Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

The Replenishment Schedule to Minimize Peak Storage Problem: The Gap Between the Continuous and Discrete Versions of the Problem

Dorit S. Hochbaum, Xu Rao

To cite this article:
Dorit S. Hochbaum, Xu Rao (2019) The Replenishment Schedule to Minimize Peak Storage Problem: The Gap Between the Continuous and Discrete Versions of the Problem. Operations Research

Published online in Articles in Advance 27 Jun 2019

. https://doi.org/10.1287/opre.2018.1839

http://pubsonline.informs.org/journal/opre/

OPERATIONS RESEARCH
Articles in Advance, pp. 1–17
ISSN 0030-364X (print), ISSN 1526-5463 (online)

METHODS

The Replenishment Schedule to Minimize Peak Storage Problem: The Gap Between the Continuous and Discrete Versions of the Problem

Dorit S. Hochbaum,[a] **Xu Rao**[a]

[a] Department of Industrial Engineering and Operations Research, University of California, Berkeley, California 94709
Contact: hochbaum@ieor.berkeley.edu, http://orcid.org/0000-0002-2498-0512 (DSH);
xrao@berkeley.edu, http://orcid.org/0000-0003-0260-891X (XR)

Received: February 15, 2018
Revised: July 30, 2018
Accepted: December 13, 2018
Published Online in Articles in Advance: June 27, 2019

Subject Classifications: analysis of algorithms: computational complexity; dynamic programming; production/scheduling: approximations/heuristic
Area of Review: Optimization

https://doi.org/10.1287/opre.2018.1839

Abstract. The replenishment storage problem (RSP) is to minimize the storage capacity requirement for a deterministic demand, multi-item inventory system, where each item has a given reorder size and cycle length. We consider the *discrete* RSP, where reorders can only take place at an integer time unit within the cycle. Discrete RSP was shown to be NP-hard for constant joint cycle length (the least common multiple of the length of all individual cycles). We show here that discrete RSP is weakly NP-hard for constant joint cycle length and prove that it is strongly NP-hard for nonconstant joint cycle length. For constant joint cycle-length discrete RSP, we further present a pseudopolynomial time algorithm that solves the problem optimally and the first known fully polynomial time approximation scheme (FPTAS) for the single-cycle RSP. The scheme is utilizing a new integer programming formulation of the problem that is introduced here. For the strongly NP-hard RSP with nonconstant joint cycle length, we provide a polynomial time approximation scheme (PTAS), which for any fixed ϵ, provides a linear time ϵ approximate solution. The *continuous* RSP, where reorders can take place at any time within a cycle, seems (with our results) to be easier than the respective discrete problem. We narrow the previously known complexity gap between the continuous and discrete versions of RSP for the multi-cycle RSP (with either constant or nonconstant cycle length) and the single-cycle RSP with constant cycle length and widen the gap for single-cycle RSP with nonconstant cycle length. For the multi-cycle case and constant joint cycle length, the complexity status of continuous RSP is open, whereas it is proved here that the discrete RSP is weakly NP-hard. Under our conjecture that the continuous RSP is easier than the discrete one, this implies that continuous RSP on multi-cycle and constant joint cycle length (currently of unknown complexity status) is at most weakly NP-hard.

Funding: This work was supported by the Division of Civil, Mechanical and Manufacturing Innovation [Grant CMMI 1760102].

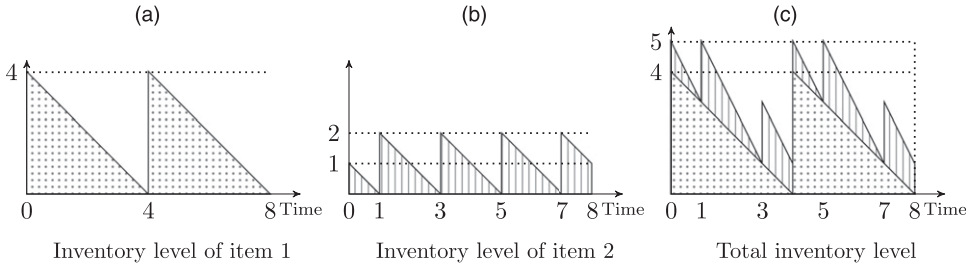
Keywords: deterministic inventory theory • complexity • approximation algorithm • weakly NP-hard • integer programming

1. Introduction

The replenishment storage problem (RSP) arises in planning a periodic replenishment schedule of multiple items so as to minimize the storage capacity required. RSP is a multi-item inventory system, where each item has deterministic demand, a given reorder size, and its own cycle length determined by its economic order quantity. We study here the *discrete* RSP, where reorders can only take place at an integer time unit within the cycle. The problem is to determine the timing of the first replenishment of each item within its cycle so that the maximum inventory level of all items over time is minimized. For the *continuous* RSP, reorders (replenishment timing) can take place at any time within the cycle of each item. RSP has been studied extensively for both the discrete and continuous versions. We clarify here the complexity status of the discrete RSP and delineate the complexity gaps between the discrete and continuous versions of the problem.

Geometrically, RSP can be viewed as a problem of shifting periodic triangle functions and then, packing them on top of each other so as to minimize the peak value required. To illustrate that, Figure 1(a) is the triangles function for the inventory level of item 1, for which the order quantity is four units and the cycle length is four periods. These correspond to the height and the width of each triangle, respectively. The replenishment timing of item 1 is at time 0. A second item, item 2, is shown in Figure 1(b). It has order

Figure 1. A Geometric View of RSP



quantity of two units and cycle length of two, and its replenishment timing is one. With this selection of the replenishment timings, the sum of the inventory levels of the two items is maximized at time 0 with a peak storage level of five, which is illustrated in Figure 1(c). This peak storage level is also the smallest possible for this discrete RSP problem. We note that the peak storage level always coincides with the reorder timing of one of the items.

An instance of discrete RSP consists of n items. Each item i is associated with an integer individual cycle length k_i and an integer reorder size s_i. Here, s_i is expressed in terms of the storage amount required for the reorder quantity. The *joint cycle length* of the n items is the least common multiple (lcm) of the lengths k_i, $i = 1, \ldots, n$. We denote $k = \text{lcm}(k_1, \ldots, k_n)$. By the cyclical nature of the problem, the total inventory levels repeat periodically every k units of time for any reorder schedule. If all items have the same cycle time, k, the problem is said to be *single-cycle*; otherwise, it is said to be *multi-cycle*.

The single-cycle discrete RSP was shown by Hall (1998) to be NP-hard, even for constant k. Because single-cycle RSP is a special case of multi-cycle RSP, it implies that multi-cycle RSP is NP-hard, even for constant joint cycle length k. Here, we prove that, for nonconstant k, both single-cycle and multi-cycle discrete RSPs are strongly NP-hard. This matches the strong NP-hardness of the continuous RSP for the multi-cycle RSP with nonconstant k (Gallego et al. 1992). Before our results here, it was conceivable that the discrete RSP for multi-cycle and nonconstant k could be easier than the continuous problem if it was shown to be weakly NP-hard. With the strong NP-hardness result here, both discrete and continuous problems are of equal complexity for this case.

For constant k, we provide here, for the first time, a pseudopolynomial time algorithm that solves discrete RSP optimally, proving that both single-cycle RSP and multi-cycle RSP for constant k are weakly NP-hard. Weakly NP-hard problems can have a fully polynomial time approximation scheme (FPTAS). Indeed, we devise for the single-cycle RSP an FPTAS. That approximation scheme utilizes a newly introduced integer programming formulation of RSP. This new formulation is of independent interest, and it is shown to be tighter that the known integer program that has been used for RSP to date. For the strongly NP-hard single-cycle RSP with nonconstant k, we devise a polynomial time approximation scheme (PTAS), which is the best approximation possible for strongly NP-hard problems. Furthermore, for a fixed parameter ϵ, that PTAS delivers an ϵ-approximate solution in linear time.

Our results narrow the complexity gap between the continuous and discrete versions of RSP for the multi-cycle RSP (with either constant or nonconstant cycle length) and the single-cycle RSP with constant cycle length. The single-cycle continuous RSP is polynomial time solvable (Homer 1966). Hence, our results widen the gap for single-cycle RSP with nonconstant cycle length. For the multi-cycle case and constant joint cycle length, the complexity status of continuous RSP is open, whereas it is proved here that the discrete RSP is weakly NP-hard. A tabular

Table 1. Summary of Complexity Results for RSP with Nonconstant Joint Cycle Length

	Continuous	Discrete	Here (discrete)
Two items	Closed form solution[a]	Closed form solution[b]	—
Single-cycle	Closed form solution[c]	$(1 + 2/k)$-approximation[d]	Strongly NP-hard and PTAS
Multi-cycle	Strongly NP-hard[e]	NP-hard[d]	Strongly NP-hard

[a]Hartley and Thomas (1982).
[b]Murthy et al. (2003).
[c]Homer (1966), Page and Paul (1976), and Zoller (1977).
[d]Hall (1998).
[e]Gallego et al. (1992).

Table 2. Summary of Algorithms and Complexity Results for RSP with Constant Joint Cycle Length

	Continuous	Discrete	Here (discrete)
Single-cycle: complexity	Polynomial[a]	NP-hard[b]	Weakly NP-hard
Single-cycle: algorithm	Closed form solution[a]	$(1+2/k)$-approximation[b]	Pseudopoly optimization algorithm and FPTAS
Multi-cycle: complexity	Open	NP-hard[b]	Weakly NP-hard
Multi-cycle: algorithm	—	—	Pseudopoly optimization algorithm

[a]Homer (1966), Page and Paul (1976), and Zoller (1977).
[b]Hall (1998).

summary of the complexity results here is provided in Tables 1 and 2.

Throughout, we will refer to the discrete RSP as RSP unless there is a risk of ambiguity.

1.1. Literature Review

Single-cycle RSP was shown to be NP-hard for $k = 2$ (Hall 1998), which implies that both single-cycle RSP and multi-cycle RSP are at least weakly NP-hard. Such proof, however, does not exclude the possibility that the problems are strongly NP-hard, and therefore, the complexity status of these two RSPs as strongly NP-hard versus weakly NP-hard has been unresolved until now. Hall (1998) also provided an approximation algorithm for single-cycle RSP, with an approximation factor of $(1+\frac{2}{k})$. Another algorithm provided by Hall (1998) delivered a two approximation, which is observed here to be satisfied by any feasible solution. The $(1+\frac{2}{k})$-approximation algorithm is of particular interest, because it is a part of the PTAS that we derive for the single-cycle RSP with nonconstant k. That algorithm links the continuous and discrete problems by rounding (down) the fractional values of the closed form solution to the respective continuous problem. A complete description of this $(1+\frac{2}{k})$-approximation algorithm is given in Section 5.

For multi-cycle RSP with only two items, Murthy et al. (2003) provided an optimal closed form replenishment solution, meaning that it is solved in constant time.

Studies of multi-cycle RSP with more than two items have been focused on the development of heuristics. These include genetic algorithms (Moon et al. 2008, Yao and Chu 2008), a smoothing procedure utilizing a Boltzmann function (Yao et al. 2008), local search procedures (Croot and Huang 2013), a simulated annealing algorithm (Boctor 2010), hybrid heuristics (Boctor 2010, Russell and Urban 2016), and an evolutionary algorithm (Boctor and Bolduc 2015). None of these heuristics were shown to deliver a guaranteed approximation bound.

The continuous single-cycle RSP was first explicitly studied by Homer (1966), who derived an optimal closed form solution. Later, Page and Paul (1976) and Zoller (1977) independently rediscovered the result of Homer (1966). Hartley and Thomas (1982) considered the continuous time multi-cycle RSP with only two items and devised an optimal closed form solution. For multiitem multi-cycle continuous RSP, the problem was proved to be strongly NP complete for nonconstant cycle lengths (Gallego et al. 1992). Anily (1991) and Hariga and Jackson (1995) obtained lower bounds on the minimum peak storage required and proposed heuristics for multi-cycle continuous RSP. Teo et al. (1998) addressed multi-cycle continuous RSP when, for all pairs of individual cycle lengths (k_i, k_j), the larger value is an integer multiple of the smaller value. For the problem with this integer ratio cycles' lengths assumption, they devised a heuristic, which is a $\frac{15}{8}$-approximation algorithm.

Continuous RSP was used as a subproblem for the problem of identifying optimal cycle lengths and replenishment schedule for multiitems so as to minimize the order and inventory holding costs without exceeding a given storage capacity. In this latter problem, each item is associated with a unit holding cost and an order cost. Under the assumption of identical cycles, an optimal closed form solution was derived from the closed form solution to the replenishment schedule for single-cycle continuous RSP (Homer 1966, Page and Paul 1976, Zoller 1977). For this problem with nonidentical cycles and only two items, a similar approach resulted in a closed form solution generated from the continuous RSP closed form solution (Hartley and Thomas 1982, Thomas and Hartley 1983). And for multi-item instances with non-identical cycles Anily (1991), Gallego et al. (1996), and Hariga and Jackson (1996) developed some heuristics based on the heuristics of multi-item multi-cycle continuous-RSP.

Summaries of the relevant known results for continuous and discrete RSPs are given in Table 1 for nonconstant k and Table 2 for constant k. These tables also include our contributions here.

1.2. Summary of Contributions

Our main contributions here are the following.

1. Resolving the complexity status of discrete RSP. We prove that, for nonconstant joint cycle length k,

the discrete RSP is strongly NP-hard regardless of whether it is single-cycle or multi-cycle. In contrast, for constant joint cycle length k, we show that the problem is weakly NP-hard and provide a pseudo-polynomial time optimization algorithm with complexity that is a polynomial function of the sum of the order sizes s_i and exponential in the joint cycle length k. These results show that, for nonconstant k, both the continuous and the discrete problems are strongly NP-hard, and that closes the gap between these two RSP problems. This provides extra evidence to support the conjecture that the discrete RSP problem can only be harder than the continuous problem. For constant k, the continuous problem is polynomial for the single-cycle case and of unknown complexity for the multi-cycle case. From our results, the problem is weakly NP-hard for both single-cycle and multi-cycle cases, which if our conjecture is true, implies that the continuous RSP for the multi-cycle case cannot be strongly NP-hard.

2. Devising the first known FPTAS for the weakly NP-hard single-cycle RSP with constant k.

3. Devising the first known PTAS for the strongly NP-hard, single-cycle RSP with nonconstant k.

4. Identifying a new integer programming formulation for RSP. This formulation enables the derivation of the FPTAS for the single-cycle RSP. We believe that this formulation is of independent interest, and it has potential applications beyond the context of RSP problems.

Summary of our results for RSP compared with the best previously known results are given in Tables 1 and 2: Table 1 for RSP with nonconstant k and Table 2 for RSP with constant k.

1.3. Paper Overview

The next section (Section 2) introduces notation, a review of the known integer programming formulation of RSP, and the derivation of our new integer programming formulation. In Section 3, we prove the strong NP-hardness of RSP for the nonconstant value of k (joint cycle length) and the weak NP-hardness of RSP for the constant value of k. These proofs apply to the complexity of both single-cycle RSP and multi-cycle RSP. The weak NP-hardness of RSP for constant k (joint cycle length) is proved here via a pseudopolynomial time dynamic programming algorithm that solves the problem optimally. This establishes that both single-cycle RSP and multi-cycle RSP with constant k are weakly NP-hard. In Section 4, we describe the new FPTAS for the single-cycle RSP for constant k. Section 5 includes the new PTAS for the single-cycle RSP with nonconstant k. That section also provides a description of the $(1+\frac{2}{k})$-approximation algorithm for the single-cycle RSP of Hall (1998). We conclude with several remarks in Section 6.

2. Preliminaries and Integer Programming Formulations

Given an instance of discrete RSP, the demand rates and inventory levels are given in terms of the respective reorder size: for item i, the demand per unit of time is $\frac{s_i}{k_i}$, and its inventory levels at each replenishment cycle of k_i time units starting at time T, $(T+0, T+1, \ldots, T+k_i-1)$, are $(s_i, \frac{k_i-1}{k_i}s_i, \frac{k_i-2}{k_i}s_i, \ldots, \frac{1}{k_i}s_i)$. The inventory level of item i on time ℓ, given that it is reordered on time j, is denoted by $V_{ij\ell}$. Thus,

$$V_{ij\ell} = s_i \cdot \left(1 - \frac{(\ell - j) \bmod k_i}{k_i}\right).$$

Recall that, because $k = \text{lcm}(k_1, \ldots, k_n)$, the inventory levels are periodic within a cycle of k time units (repeat every k time units). It is, therefore, sufficient to determine the peak storage requirement by examining a time interval of length k. In this interval, we only need to look at discrete time values from one to k, because that peak storage always coincides with the reorder timing of an item. Note that inventory level at time 0 is the same as inventory level at time k.

The decision variables in the integer programming formulations are the assignments of time periods within the k-unit timeframe to the orders of all items. This assignment of timing is given as an $n \times k$ binary matrix $\mathbf{x}$, where

$$x_{ij} = \begin{cases} 1 & \text{if item } i \text{ is ordered at time } j, \\ 0 & \text{otherwise.} \end{cases}$$

Definition 1. A $n \times k$ binary matrix $\mathbf{x}$ is said to be a *valid assignment* for a given instance if and only if each item i is replenished exactly once every k_i time units. That is,

$$\sum_{j=1}^{k_i} x_{ij} = 1 \quad i = 1, \ldots, n, \quad \text{and}$$

$$x_{ij} = x_{i,(j-k_i)} \quad i = 1, \ldots, n, \quad j = k_i + 1, \ldots, k.$$

The following listed notations denote demand rates, inventory levels, the total sum of reorder sizes at an integer time, and the optimal peak storage.

$d_i = \frac{s_i}{k_i}$: demand rate of item i for $i = 1, \ldots, n$.

$D = \sum_{i=1}^{n} d_i = \sum_{i=1}^{n} \frac{s_i}{k_i}$: total demand (aggregate stock depletion) per unit of time.

$V_{ij\ell} = s_i \cdot \left(1 - \frac{(\ell-j) \bmod k_i}{k_i}\right)$: the inventory level of item i at time ℓ given that the reorder time is j.

$V_\ell(\mathbf{x}) = \sum_{i=1}^{n} \sum_{j=1}^{k_i} V_{ij\ell} x_{ij}$: the inventory level at time ℓ for $\ell = 1, \ldots, k$.

$V(\mathbf{x}) = \max_{\ell \in \{1,\ldots,k\}} V_\ell(\mathbf{x})$: the maximum inventory level (peak storage) of a cycle.

$Q_j(\mathbf{x}) = \sum_{i=1}^{n} s_i x_{ij}$: the total sum of reorder sizes at time j for $j = 1, \ldots, k$.

$V^* = \min_{\mathbf{x} \text{ valid}} V(\mathbf{x})$: the optimal peak inventory level.

Let the following quantity, which is a constant, be denoted by C: $C = \sum_{i=1}^{n} \frac{1}{2}(1+\frac{1}{k_i})ks_i + \frac{(1+k)k}{2}D$. This quantity is used in deriving the new formulation and the FPTAS.

It is observed next that the peak inventory level for any valid assignment $\mathbf{x}$ is within twice the optimum. Hence, any replenishment schedule is a two-approximate solution.

Lemma 1. *Any valid assignment* $\mathbf{x}$ *is a* 2*-approximate solution.*

Proof. At time ℓ for $\ell = 1,\ldots,k$, the inventory level of all items is $V_\ell(\mathbf{x}) = \sum_{i=1}^{n}\sum_{j=1}^{k_i} V_{ij\ell}x_{ij}$. Because

$$V_{ij\ell} = s_i \cdot \left(1 - \frac{(\ell - j) \bmod k_i}{k_i}\right) \le s_i,$$

$$V_\ell(\mathbf{x}) \le \sum_{i=1}^{n}\sum_{j=1}^{k_i} s_i x_{ij} = \sum_{i=1}^{n}\left(s_i \sum_{j=1}^{k_i} x_{ij}\right) = \sum_{i=1}^{n} s_i.$$

Hence, the storage capacity needed is less than or equal to $\sum_{i=1}^{n} s_i$.

However, the average inventory level of item i per time unit is $\frac{1}{2}(1+\frac{1}{k_i})s_i$ for any i. Therefore, the aggregated average inventory level of all items per time unit is $\frac{1}{k}\sum_{\ell=1}^{k} V_\ell(\mathbf{x}) = \sum_{i=1}^{n}\frac{1}{2}(1+\frac{1}{k_i})s_i > \sum_{i=1}^{n}\frac{1}{2}s_i$. The peak inventory level is at least as much as this aggregated average inventory level; thus, $V^* \ge \frac{1}{2}\sum_{i=1}^{n} s_i$.

Consequently, $V(\mathbf{x}) \le \sum_{i=1}^{n} s_i \le 2V^*$, which means that $\mathbf{x}$ is a 2-approximate solution. □

The standard integer programming formulation is as follows. A straightforward formulation of RSP was used in previous studies, including Murthy et al. (2003), Boctor (2010), and Russell and Urban (2016). Let the binary variables y_{ij} be

$$y_{ij} = \begin{cases} 1 & \text{if item } i \text{ is ordered at time } j, \\ & \quad \text{for } i = 1,\ldots,n, \quad j = 1,\ldots,k_i \\ 0 & \quad \text{otherwise.} \end{cases}$$

Note that variables y_{ij} coincide with variables x_{ij} within the first k_i time periods.

Let $V_{ij\ell}$ be parameters defined as above, $V_{ij\ell} = s_i\left(1 - \frac{(\ell-j) \bmod k_i}{k_i}\right)$. The standard integer programming formulation (RSP_0) is as follows:

$$\begin{aligned} (\text{RSP}_0) \quad \min \quad & V \\ \text{subject to} \quad & \sum_{i=1}^{n}\sum_{j=1}^{k_i} V_{ij\ell}y_{ij} \le V \quad \ell = 1,\ldots,k \\ & \sum_{j=1}^{k_i} y_{ij} = 1 \qquad i = 1,\ldots,n \\ & y_{ij} \text{ binary for } i = 1,\ldots,n, \quad j = 1,..,k_i. \end{aligned}$$

2.1. A New Integer Programming Formulation

We observe that formulation (RSP_0) has multiple solutions of the same value. Specifically, for any valid assignment that attains the peak storage on day ℓ, there are $k-1$ other valid assignments of the same objective value that attain the maximum storage on any day other than ℓ. For example, to attain the peak storage on day $\ell+1$, we shift the reorder by one day forward. Our new formulation requires that any feasible assignment attains its peak storage at time k (or equivalently, at time 0). This is proved to be possible with "shift" permutations in Lemma 2. Another aspect of the new formulation is that, instead of dealing with inventory levels directly as in (RSP_0), the new formulation uses, as the main variables, the total reorder size at time j, $Q_j(\mathbf{x}) = \sum_{i=1}^{n} s_i x_{ij}$ for $j = 1,\ldots,k$. Lemmas 3 and 4 establish the relationship between inventory levels and reorder sizes.

Lemma 2. *For any valid assignment* $\mathbf{x}$*, there is a shift permutation of* $1,\ldots,k$ *denoted by* $\pi(1),\ldots,\pi(k)$ *such that the valid assignment* $\mathbf{x}'$ *with* $x'_{ij} = x_{i\pi(j)}$ *attains peak inventory level at time* k*, and this new peak inventory level equals the peak inventory level of assignment* $\mathbf{x}$*. That is,* $V_k(\mathbf{x}') = V(\mathbf{x}') = V(\mathbf{x})$.

Proof. Suppose that the peak storage for assignment $\mathbf{x}$ is attained at time h, $V_h(\mathbf{x}) = V(\mathbf{x})$.

For the following shift permutation,

$$\pi(j) = \begin{cases} (j+h), & \text{if } j+h \le k \\ (j+h)-k, & \text{if } j+h > k, \end{cases}$$

the assignment $\mathbf{x}'$ with $x'_{ij} = x_{i\pi(j)}$ is a new valid assignment, which is h time units shifted back in time compared with $\mathbf{x}$. In other words, the inventory levels induced by the new assignment $\mathbf{x}'$ form a shift permutation of the original sequence of inventory levels, $V_j(\mathbf{x}') = V_{\pi(j)}(\mathbf{x})$. Therefore, the peak inventory levels of the two assignments are the same, and $V_k(\mathbf{x}') = V_{\pi(k)}(\mathbf{x}) = V_h(\mathbf{x}) = V(\mathbf{x})$. □

With Lemma 2, we can restrict valid assignments to those attaining peak inventory level at time k without changing the optimal solution of RSP. Hence, RSP can also be formulated as minimizing the inventory level at time k such that the schedule is a valid assignment that attains peak inventory level at time k, which can be written as $V_\ell(\mathbf{x}) \le V_k(\mathbf{x})$ for $\ell = 1,\ldots,k$.

Next, we show in Lemma 3 that, for any valid assignment $\mathbf{x}$, the inventory level of time ℓ can be determined based only on $Q_j(\mathbf{x})$, the total amount ordered at time j, the inventory level at time k, $V_k(\mathbf{x})$, and D, the total sum of demand rates of all items:

Lemma 3. *For any valid assignment* $\mathbf{x}$*,*

$$V_\ell(\mathbf{x}) = V_k(\mathbf{x}) - \ell D + \sum_{j=1}^{\ell} Q_j(\mathbf{x}), \quad \ell = 1,..,k. \qquad (1)$$

Proof. For each unit of time, item i's inventory is reduced by its demand rate d_i, and the total inventory level is reduced by $D = \sum_{i=1}^{n} d_i$. Because $Q_j(\mathbf{x}) = \sum_{i=1}^{n} s_i x_{ij}$ is the reorder size at time j, the inventory level at time j is $V_j(\mathbf{x}) = V_{(j-1) \bmod k}(\mathbf{x}) - D + Q_j(\mathbf{x})$. Note that inventory level at time 0 is the same as inventory level at time k by the cyclical nature of RSP. For any integer time ℓ, we apply this equation recursively with $j = \ell, \ell-1, \ldots, 1$ to derive that $V_\ell(\mathbf{x}) = V_k(\mathbf{x}) - \ell D + \sum_{j=1}^{\ell} Q_j(\mathbf{x})$. □

Lemma 2 shows that we can formulate the RSP as minimizing the inventory level at time k such that the schedule is a valid assignment. A consequence of this lemma is that inequalities $V_\ell(\mathbf{x}) \le V_k(\mathbf{x})$ for $\ell = 1, \ldots, k$ can be rewritten using Equation (1) in Lemma 3 as

$$V_k(\mathbf{x}) \ge V_k(\mathbf{x}) - \ell D + \sum_{j=1}^{\ell} Q_j(\mathbf{x}) \text{ for } \ell = 1, \ldots, k$$

or equivalently,

$$\sum_{j=1}^{\ell} Q_j(\mathbf{x}) \le \ell D \text{ for } \ell = 1, .., k. \tag{2}$$

We refer to this set of inequalities (2) as the *cascading constraints*. These constraints enforce the peak storage at time k.

Next, we address the objective function. Let $z(\mathbf{x})$ be the following function of a valid assignment $\mathbf{x}$:

$$z(\mathbf{x}) = \sum_{j=1}^{k} (k-j+1) Q_j(\mathbf{x}) = \sum_{j=1}^{k} (k-j+1) \sum_{i=1}^{n} s_i x_{ij}.$$

In the next lemma, we prove that minimizing the inventory level of time k, $V_k(\mathbf{x})$, is equivalent to maximizing $z(\mathbf{x})$. This is proved by showing that the sum of $kV_k(\mathbf{x})$ and $z(\mathbf{x})$ is a constant: the constant C defined earlier.

Lemma 4. $kV_k(\mathbf{x}) + z(\mathbf{x}) = \sum_{i=1}^{n} \frac{1}{2}(1 + \frac{1}{k_i}) k s_i + \frac{(1+k)k}{2} D.$

Proof. For item i, the sum of storage space that it takes up during its k_i time units of reorder cycle is $\sum_{j=1}^{k_i} \cdot \frac{k_i+1-j}{k_i} s_i = \frac{1}{2}(1 + \frac{1}{k_i}) k_i s_i$. There are $\frac{k}{k_i}$ reorder cycles for item i during the timeframe of length k so that item i takes a total of $\frac{1}{2}(1 + \frac{1}{k_i}) k s_i$ units of storage space. The sum of all items' inventory levels over the k integer times is hence $\sum_{\ell=1}^{k} V_\ell(\mathbf{x}) = \sum_{i=1}^{n} \frac{1}{2}(1 + \frac{1}{k_i}) k s_i$. Using Equation (1) in the statement of Lemma 3, $\sum_{\ell=1}^{k} V_\ell(\mathbf{x})$ can be rewritten as

$$\begin{aligned}
\sum_{\ell=1}^{k} V_\ell(\mathbf{x}) &= \sum_{\ell=1}^{k} \left(V_k(\mathbf{x}) - \ell D + \sum_{j=1}^{\ell} Q_j(\mathbf{x}) \right) \\
&= kV_k(\mathbf{x}) - \frac{(1+k)k}{2} D + \sum_{j=1}^{k} (k-j+1) Q_j(\mathbf{x}) \\
&= kV_k(\mathbf{x}) - \frac{(1+k)k}{2} D + z(\mathbf{x}).
\end{aligned}$$

It follows that

$$\begin{aligned}
kV_k(\mathbf{x}) + z(\mathbf{x}) &= \sum_{\ell=1}^{k} V_\ell(\mathbf{x}) + \frac{(1+k)k}{2} D \\
&= \sum_{i=1}^{n} \tfrac{1}{2}(1 + \frac{1}{k_i}) k s_i + \frac{(1+k)k}{2} D,
\end{aligned}$$

and the right-hand side is a constant for every problem instance. □

From Lemma 4, it follows that minimizing $V(\mathbf{x})$ can be replaced by maximizing $z(\mathbf{x})$. This with the cascading constraints leads to the new integer programming formulation (RSP_1). For presentation simplicity, we use $Q_j(\mathbf{x}) = \sum_{i=1}^{n} s_i x_{ij}$:

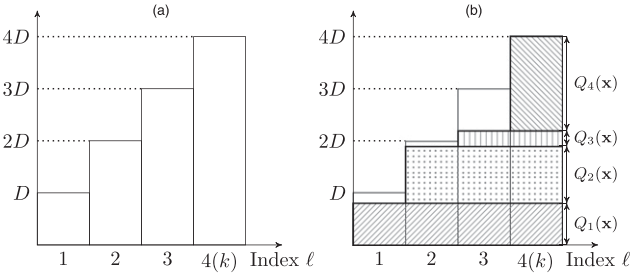
$$\begin{aligned}
(\text{RSP}_1) \quad \max \quad & z(\mathbf{x}) = \sum_{j=1}^{k} (k-j+1) Q_j(\mathbf{x}) \\
\text{subject to} \quad & \sum_{j=1}^{\ell} Q_j(\mathbf{x}) \le \ell D \quad \ell = 1, .., k \\
& \sum_{j=1}^{k_i} x_{ij} = 1 \qquad i = 1, \ldots, n \\
& x_{ij} = x_{i,(j-k_i)} \; i = 1, \ldots, n, \; j = k_i + 1, \ldots, k \\
& x_{ij} \text{ binary for } i = 1, \ldots, n, \quad j = 1, .., k_i.
\end{aligned}$$

2.1.1. A Graphical Visualization of (RSP_1). To provide the intuition behind the cascading constraints and objective function $z(\mathbf{x})$ of (RSP_1), we present a graphical illustration in Figure 2. In the example illustrated, we let the cycle length be $k = 4$. In Figure 2(a), there are $k = 4$ columns with heights ℓD for time $\ell = 1, \ldots, k$, which represent the right-hand sides of the cascading constraints.

In Figure 2(b), there is a rectangle with height $Q_1(\mathbf{x})$ that extends from day 1 to day k. A second rectangle with height $Q_2(\mathbf{x})$ is stacked on the first rectangle, extending from day 2 to day k. The third rectangle of height $Q_3(\mathbf{x})$ extends from day 3 to day k, and the fourth and last rectangle of height $Q_4(\mathbf{x})$ extends only over time period (day) $k = 4$. With this construction, the sum of heights of all rectangles within the column corresponding to time ℓ is exactly the left-hand side of the ℓth cascading constraint, $\sum_{j=1}^{\ell} Q_j(\mathbf{x})$.

The area of the first rectangle is $kQ_1(\mathbf{x})$, the second is $(k-1)Q_2(\mathbf{x})$, and in general, the jth rectangle covers

Figure 2. A Graphical Visualization of (RSP_1)



an area of $(k+1-j)Q_j(\mathbf{x})$. Therefore, the sum of areas of all rectangles is $z(\mathbf{x}) = \sum_{j=1}^{k}(k-j+1)Q_j(\mathbf{x})$, which is the objective function of (RSP_1).

Therefore, geometrically, (RSP_1) seeks to find the valid assignment $\mathbf{x}$ that maximizes the area of the rectangles representing $Q_j(\mathbf{x})$ subject to the constraints that the total height of the rectangles in column ℓ does not exceed the column height ℓD.

We call the area not covered in each column the remainder. The remainders are the spaces between the column heights in Figure 2 and the packed rectangles of area $z(\mathbf{x})$. Obviously, minimizing those is identical to maximizing $z(\mathbf{x})$, because the total area of the remainders plus the area covered by the rectangles $z(\mathbf{x})$ is a constant, $\sum_{\ell=1}^{k}\ell D = \frac{k+1k}{D}$. It is important to observe that the area of the remainders is not fully available for additional items. That is because if an item j is added to the reorder in day $p(j)$, its size s_j will diminish the remainder not only on day $p(j)$ but also, on days $p(j)+1,\ldots,k$. Instead, the critical forms of remainders are the *adjusted remainders* defined in Section 4.1, which are added to the objective function $z(\mathbf{x})$ for the derivation of the FPTAS.

2.1.2. Comparing the New and Standard Formulations (RSP_0) and (RSP_1). As noted before, (RSP_1) removes multiple optima that differ in the ordering present in (RSP_0) and always selects an optimal solution with peak storage attained at time k. In that sense, any solution to (RSP_1) and its Linear Programming (LP) relaxation is contained in the set of feasible solutions to (RSP_0) or its LP relaxation. Yet, there are other features on which the two problems differ. The new presentation of the problem as a maximization problem opens up the option, when there are multiple solutions to (RSP_1), of selecting the solution that minimizes the "adjusted remainders," a concept described in Section 4.1 that is crucial in the derivation of the FPTAS and the PTAS. A key result shown in Section 4.1 is that the optimal solution to the new integer programming formulation when the sum of the adjusted remainders is added to the objective is "close" to the optimal solution.

Another difference between the two formulations is the form of the constraints' matrix coefficients. The coefficients of binary variables in the constraints of (RSP_0) vary for each i,j,ℓ, because the coefficient $V_{ij\ell}$ follows the triangular line. By contrast, in the cascading constraints of (RSP_1), the coefficients of binary variable x_{ij} can only be zero or s_i for each i,j, because they represent the packing of rectangles. For single-cycle RSP, the coefficients of x_{ij} in (RSP_1) are all 0 in the first $j-1$ cascading constraint and all s_i for the remaining ones. We believe that, as a result, the LP relaxation of (RSP_0) is likely to have more fractional variables in an optimal solution than the LP relaxation of (RSP_1).

3. The Complexity of RSP for Constant and Nonconstant Joint Cycle Length

RSP has been known to be NP-hard for both the single-cycle and multi-cycle versions and for either constant or nonconstant joint cycle length k (Hall 1998). We show in this section that, for constant k, both the single-cycle RSP and the multi-cycle RSP are actually weakly NP-hard. We derive for both problems a pseudopolynomial time algorithm that solves these problems optimally. For nonconstant k, we show here that both the single-cycle RSP and the multi-cycle RSP are strongly NP-hard (and hence, there cannot be a pseudopolynomial time algorithm for nonconstant k unless NP = P). We first prove the strong NP-hardness for nonconstant k.

3.1. Nonconstant Joint Cycle RSP Is Strongly NP-Hard

The single-cycle RSP is a special case of multi-cycle RSP. It is, therefore, sufficient to prove that the nonconstant joint cycle-length single-cycle RSP is strongly NP-hard because that would imply that multi-cycle RSP is also strongly NP-hard for k nonconstant. The reduction is from the *three-partition* problem, a well-known strongly NP-hard problem (Gary and Johnson 1979).

The three-partition problem is as follows. Given integers $a_1,a_2,\ldots,a_{3m}$ and integer b such that $\frac{1}{4}b < a_i < \frac{1}{2}b$ for each i and $\sum_{i=1}^{3m}a_i = mb$, can $\{1,2,\ldots,3m\}$ be partitioned into m disjoint sets $A_1,A_2,\ldots,A_m$ such that, for $1 \le j \le m$, $\sum_{i\in A_j}a_i = b$?

Theorem 1. *The single-cycle* RSP *with nonconstant joint cycle length is strongly NP-hard.*

Proof. Given an instance of three partition, we define an instance of single-cycle RSP with $n = 3m$, $k = m$, and $s_i = a_i$ for $1 \le i \le 3m$. The decision problem for RSP is stated as follows. Is there a valid assignment $\mathbf{x}$ with peak inventory level that is less than or equal to $V = \frac{1+m}{2m}\sum_{i=1}^{3m}a_i$?

We observe that the sum of inventory levels at integer times over a cycle equals $\sum_{i=1}^{3m}(1+\frac{m-1}{m}+\ldots+\frac{1}{m})a_i = \sum_{i=1}^{3m}\frac{m+1}{2}a_i = mV$. Having $\sum_{\ell=1}^{k}V_\ell(\mathbf{x}) = mV$, the peak inventory level $\max_\ell V_\ell(\mathbf{x}) \le V$ is equivalent to $V_\ell(\mathbf{x}) = V$ for all ℓ. This means that the inventory levels at all integer time units are the same. From Lemma 3, it follows that $V_1(\mathbf{x}) = V_2(\mathbf{x}) = \ldots = V_m(\mathbf{x})$ if and only if $Q_1(\mathbf{x}) = Q_2(\mathbf{x}) = \ldots = Q_m(\mathbf{x}) = D$, where $D = \sum_{i=1}^{3m}\frac{a_i}{m} = b$. Let $A_j = \{i | x_{ij} = 1\}, j = 1,\ldots,m$; then, $Q_j(\mathbf{x}) = \sum_{i=1}^{n}a_i x_{ij} = \sum_{i\in A_j}a_i$. Therefore, the decision problem corresponding to single-cycle RSP has a yes answer if and only if there is a partition $A_j, j = 1,\ldots,m$ such that $\sum_{i\in A_j}a_i = b$ for $1 \le j \le m$. □

3.2. Constant Joint Cycle-Length RSP Is Weakly NP-Hard

We prove the weak NP-hardness of RSP with constant k by devising a dynamic programming algorithm solving RSP optimally in pseudopolynomial time. The complexity of this dynamic programming algorithm depends on the value of the order sizes (rather than the length/logarithm of these sizes) and is exponential in k. Such complexity is considered pseudopolynomial for constant k. Because both single-cycle RSP and multi-cycle RSP with constant k are NP-hard (Hall 1998), the existence of such an algorithm implies that RSP for constant k is weakly NP-hard for both single-cycle and multi-cycle.

The dynamic programming algorithm presented in this paper is associated with the integer programming formulation (RSP_1). It is also possible to devise a dynamic programming algorithm of the same complexity based on the standard formulation (RSP_0). We comment, however, that, to generate the FPTAS presented later, it is essential to use the dynamic programming procedure that is based on (RSP_1).

For h, an integer such that $0 \le h \le n$, let $\mathbf{x}^h$ denote the assignment of reorders for the first h items. Let the function $f_h(q_1, q_2, \ldots, q_k)$ be the maximum of $z(\mathbf{x}^h)$, with the cumulative reorder sizes at time ℓ being restricted to less than or equal to q_ℓ for $\ell = 1, \ldots, k$. Here, $(q_1, \ldots, q_k)$ is an integer array with $q_\ell \in [0, \ell D]$. Formally,

$$
\begin{aligned}
&f_h(q_1, q_2, \ldots, q_k)\\
&\quad = \max \quad \sum_{j=1}^{k} (k-j+1) Q_j(\mathbf{x}^h)\\
&\text{subject to} \quad \sum_{j=1}^{\ell} Q_j(\mathbf{x}^h) \le q_\ell \quad \ell = 1, .., k\\
&\qquad \sum_{j=1}^{k_i} x_{ij} = 1 \quad i = 1, \ldots, h\\
&\qquad x_{ij} = x_{i(j-k_i)} \quad i = 1, \ldots, h,\\
&\qquad\qquad\qquad\qquad j = k_i + 1, \ldots, k\\
&\qquad x_{ij} \text{ binary for } i = 1, \ldots, h,\ j = 1, .., k_i,
\end{aligned}
$$

where $Q_j(\mathbf{x}^h) = \sum_{i=1}^{h} s_i x_{ij}$. We set $f_h(q_1, q_2, \ldots, q_k) = -\infty$ if the above integer programming problem is infeasible. The optimal solution being sought is $f_n(D, 2D, \ldots, kD)$.

The values of the function $f_h(q_1, q_2, \ldots, q_k)$ are evaluated for every $0 \le h \le n$ and any integer array $(q_1, \ldots, q_k)$, where $q_j \in [0, jD]$ with a dynamic programming recursion. The boundary conditions are $f_0(q_1, q_2, \ldots, q_k) = 0$ for any $(q_1, q_2, \ldots, q_k)$. The recursive derivation of $f_h(q_1, q_2, \ldots, q_k)$ from $f_{h-1}(\cdot)$ is required to determine the timing to replenish item h within the first k_h time units so as to maximize the objective $\sum_{j=1}^{k} (k-j+1) Q_j(\mathbf{x}^h)$.

To see how the recursion works, we split the objective function into the terms involving item h and the terms involving items $1, \ldots, h-1$:

$$
\begin{aligned}
\sum_{j=1}^{k} (k-j+1) Q_j(\mathbf{x}^h) &= \sum_{j=1}^{k} (k-j+1) s_h x_{hj}\\
&\quad + \sum_{j=1}^{k} (k-j+1) Q_j(\mathbf{x}^{(h-1)}). \qquad (3)
\end{aligned}
$$

If time τ, for $1 \le \tau \le k_h$, is selected to be the first reorder timing of item h, the following reorder timings of item h are $\tau + k_h, \tau + 2k_h, \ldots, \tau + (\frac{k}{k_h} - 1)k_h$. The first part of (3) is then

$$
\begin{aligned}
\sum_{j=1}^{k} (k-j+1) s_h x_{hj} &= \sum_{t=0}^{k/k_h - 1} (k + 1 - \tau - t k_h) s_h\\
&= \left(\frac{k + k_h}{2} + 1 - \tau \right) \frac{k}{k_h} s_h.
\end{aligned}
$$

Let $q'_\ell(\tau) = q_\ell - \sum_{j=1}^{\ell} s_h x_{hj} = q_\ell - \lfloor \frac{\ell - \tau + k_h}{k_h} \rfloor s_h$; the second term in (3), $\sum_{j=1}^{k} (k-j+1) Q_j(\mathbf{x}^{(h-1)})$, has a maximum of $f_{h-1}(q'_1(\tau), \ldots, q'_k(\tau))$ if $q'_\ell(\tau) \ge 0$ for all ℓ and minus infinity otherwise.

The recursive equation using the notation $q'_\ell(\tau) = q_\ell - \lfloor \frac{\ell - \tau + k_h}{k_h} \rfloor s_h$ is

$$
\begin{aligned}
&f_h(q_1, q_2, \ldots, q_k)\\
&= \begin{cases}
\max\limits_{\tau = 1, \ldots, k_h} \{ \left(\frac{k + k_h}{2} + 1 - \tau \right) \frac{k}{k_h} s_h \\
\qquad + f_{h-1}(q'_1(\tau), \ldots, q'_k(\tau)) \}, & \text{if } q'_\ell(\tau) \ge 0 \text{ for all } \ell \\
-\infty & \text{otherwise.}
\end{cases}
\end{aligned}
$$

All function values are evaluated recursively for $h = 1, \ldots, n$ and all integer values of $(q_1, \ldots, q_k)$, where each $q_j \in [0, jD]$ and q_j integer. Each function evaluation is associated with a choice of $\tau(h)$, which is the timing of the replenishment of item h within the k_h cycle. The optimal objective value is then $f_n(D, 2D, \ldots, kD)$.

To recover the optimal valid assignment, we record the choices of the replenishment timings within the k cycle for each function value evaluation.

Because there are $O(n)$ possible values of h and $O(\ell D)$ possible values of q_ℓ for each $1 \le \ell \le k$, the total number of function evaluations is $O(n \cdot 1D \cdot 2D \cdot \ldots \cdot kD) = O(n \cdot k! D^k)$. Each evaluation for item h enumerates the $O(k_h)$ choices of τ, and for each choice τ, it takes $O(k)$ to compute $q'_1(\tau), \ldots, q'_k(\tau)$. Therefore, the run time of this dynamic programming procedure is $O(k^2 \cdot k! D^k \cdot n)$, which is $O(nD^k)$ for constant k. This complexity is pseudopolynomial, and hence, RSP is weakly NP-hard for constant k. This weak NP-hardness applies for both single-cycle RSP and multi-cycle RSP problems.

4. A Fully Polynomial Time Approximation Scheme for Single-Cycle RSP with Constant k

An approximation scheme is a family of $(1+\epsilon')$-approximation algorithms for every $\epsilon' > 0$. For any given $\epsilon' > 0$, we let $\epsilon = \frac{\epsilon'}{3+\epsilon'} > 0$, and therefore, $O(\frac{1}{\epsilon}) = O(\frac{3+\epsilon'}{\epsilon'}) = O(\frac{1}{\epsilon'})$. The $(1+\epsilon')$-approximation algorithm for RSP works by first assigning reorder timings of "large" items, where large items are those with reorder size greater than ϵD. This large assignment is determined by solving a variant of (RSP_1) that involves adjusted remainders. All items that are not large, which are considered "small," are assigned in a greedy fashion to any "adjusted" cascading constraint that still has a positive slack. It is shown that such a solution is within a factor of $1+\epsilon'$ of the optimal solution. The run time of this approximation algorithm is polynomial in n and $\frac{1}{\epsilon}$. Because $O(\frac{1}{\epsilon}) = O(\frac{1}{\epsilon'})$, this run time is also polynomial in $\frac{1}{\epsilon'}$, and hence, this family of algorithms is a fully polynomial approximation scheme.

Because we address here the single-cycle RSP, we use a formulation referred to as (k-RSP_1), which is the integer programming (RSP_1) for a single-cycle of length k RSP:

$$
\begin{aligned}
(k\text{-RSP}_1) \quad \max\ & z(\mathbf{x}) = \sum_{j=1}^{k}(k-j+1)Q_j(\mathbf{x}) \\
\text{subject to}\ & \sum_{j=1}^{\ell} Q_j(\mathbf{x}) \le \ell D \quad \ell = 1,..,k \\
& \sum_{j=1}^{k} x_{ij} = 1 \qquad i = 1,\dots,n \\
& x_{ij} \text{ binary for } i = 1,\dots,n,\ j=1,..,k.
\end{aligned}
$$

We next define the classification of the n items as large or small. For a given value of $\epsilon > 0$, let the set of large items be $I_{\text{L}} = \{i | s_i > \epsilon D\}$ and the set of small items be $I_{\text{S}} = \{i | s_i \le \epsilon D\}$. Let $n_{\text{L}} = |I_{\text{L}}|$ denote the number of large items and $n_{\text{S}} = |I_{\text{S}}|$ denote the number of small items. It is observed that, because $kD = \sum_{i=1}^{n} s_i \ge \sum_{i\in I_{\text{L}}} s_i > n_{\text{L}} \cdot \epsilon D$, the number of large items n_{L} is bounded by $\frac{k}{\epsilon}$.

Let an $n \times k$ binary matrix $\mathbf{x}^{\text{L}}$ determine the assignment of large items as

$$
x^{\text{L}}_{ij} = \begin{cases} 1 & \text{if } i \in I_{\text{L}}, \text{ and item } i \text{ is ordered on time } j \\ 0 & \text{otherwise.} \end{cases}
$$

Similarly, an $n \times k$ binary matrix $\mathbf{x}^{\text{S}}$ is the assignment of small items, where

$$
x^{\text{S}}_{ij} = \begin{cases} 1 & \text{if } i \in I_{\text{S}}, \text{ and item } i \text{ is ordered on time } j \\ 0 & \text{otherwise.} \end{cases}
$$

Definition 2. An assignment of large (small) items $\mathbf{x}^{\text{L}}$ ($\mathbf{x}^{\text{S}}$) is a *valid large (small) assignment* for a given single-cycle RSP instance if and only if any large (small) item i is replenished exactly once in a joint cycle. That is,

$$
\sum_{j=1}^{k} x^{\text{L}}_{ij} = 1 \qquad i \in I_{\text{L}} \qquad \left(\sum_{j=1}^{k} x^{\text{S}}_{ij} = 1 \qquad i \in I_{\text{S}}\right).
$$

By Definitions 1 (of valid assignment) and 2, it is clear that the sum of any valid large assignment $\mathbf{x}^{\text{L}}$ and any valid small assignment $\mathbf{x}^{\text{S}}$, $\mathbf{x} = \mathbf{x}^{\text{L}} + \mathbf{x}^{\text{S}}$, is a valid assignment of all items.

Definition 3. For any assignment $\mathbf{x}$ and $I \subseteq \{1,\dots,n\}$, the $n \times k$ binary matrix $P_I(\mathbf{x})$ is said to be the *projection* of the $\mathbf{x}$ onto I, where

$$
P_I(\mathbf{x})_{ij} = \begin{cases} x_{ij} & \text{if } i \in I \\ 0 & \text{if } i \notin I. \end{cases}
$$

It follows that, for a valid assignment $\mathbf{x}$, its projection on the large (small) item set $\mathbf{x}^{\text{L}} = P_{I_{\text{L}}}(\mathbf{x})$ ($\mathbf{x}^{\text{S}} = P_{I_{\text{S}}}(\mathbf{x})$) is a valid large (small) assignment, and furthermore, $z(\mathbf{x}) = z(\mathbf{x}^{\text{L}}) + z(\mathbf{x}^{\text{S}})$; $Q_j(\mathbf{x}) = Q_j(\mathbf{x}^{\text{L}}) + Q_j(\mathbf{x}^{\text{S}})$.

The FPTAS devised here for single-cycle RSP consists of two stages. In the first stage, we determine an assignment of the large items, $\hat{\mathbf{x}}^{\text{L}}$, and in the second stage, we assign the small items, $\hat{\mathbf{x}}^{S}$. For a given $\epsilon' > 0$ and $\epsilon = \frac{\epsilon'}{3+\epsilon'}$, we show that the valid assignment $\hat{\mathbf{x}} = \hat{\mathbf{x}}^{\text{L}} + \hat{\mathbf{x}}^{\text{S}}$ is a $(1+\epsilon')$-approximate solution.

4.1. The Assignment of Large Items

The procedure for assigning the large items addresses a modified (k-RSP_1) in which the objective function is changed by adding terms dependent on a form of slacks in the cascading constraints, the adjusted remainders. We call this problem (modified k-RSP_1). The modified problem is then scaled in that the order sizes are scaled by $\epsilon^2 D$. The resulting scaled problem (scaled modified k-RSP_1) is then shown to be solvable using the dynamic programming procedure of Section 3.2 and generating an assignment of large items that has objective function value close to the optimal value of (k-RSP_1).

4.1.1. The Adjusted Remainders.

The cascading constraints are equivalent to $\sum_{j=1}^{\ell} Q_j(\mathbf{x}^{\text{S}}) \le \ell D - \sum_{j=1}^{\ell} Q_j(\mathbf{x}^{\text{L}})$ for $\ell = 1,..,k$. Hence, the remaining "space" for small items in the first ℓ integer times is no more than $R_\ell(\mathbf{x}^{\text{L}}) = \ell D - \sum_{j=1}^{\ell} Q_j(\mathbf{x}^{\text{L}})$. We refer to $R_\ell(\mathbf{x}^{\text{L}})$ as the *remainder* of time ℓ. The cascading constraints are then equivalently written as

$$
\sum_{j=1}^{\ell} Q_j(\mathbf{x}^{\text{S}}) \le R_\ell(\mathbf{x}^{\text{L}}) \text{ for } \ell = 1,..,k. \tag{4}
$$

For any integer ℓ_1, ℓ_2 such that $1 \le \ell_1 < \ell_2 \le k$, the constraint of time ℓ_2 in the form of (4) implies

$$
\sum_{j=1}^{\ell_1} Q_j(\mathbf{x}^{\text{S}}) \le R_{\ell_2}(\mathbf{x}^{\text{L}}) - \sum_{j=\ell_1+1}^{\ell_2} Q_j(\mathbf{x}^{\text{S}}) \le R_{\ell_2}(\mathbf{x}^{\text{L}}).
$$

Hence, these constraints (4) are equivalent to $\sum_{j=1}^{\ell} \cdot Q_j(\mathbf{x}^S) \le \min_{j\ge\ell} R_j(\mathbf{x}^L)$ for $\ell = 1,\ldots,k$. We refer to the right-hand sides of these inequalities, $\bar{R}_\ell(\mathbf{x}^L) = \min_{j\ge\ell} \cdot R_j(\mathbf{x}^L)$, as the adjusted remainders. Obviously, the adjusted remainder for any inequality ℓ can only be smaller than the respective remainder, $\bar{R}_\ell(\mathbf{x}^L) \le R_\ell(\mathbf{x}^L)$.

To illustrate the concepts of remainders and adjusted remainders, we provide an instance of a single-cycle RSP with five items for $k = 4$ (see Table 3). In this instance, $D = 10$, and we take $\epsilon = 0.5$. Then, three items are determined as large, and two are determined as small.

Consider the assignment of large items: item 1 to time 1, item 2 to time 2, and item 3 to time 4. Figure 3 visualizes the four cascading constraints for this assignment of large items: the columns heights indicate the right-hand side of the four cascading constraints, and rectangles with patterns in each column represent the large items that are replenished before and on the indexed time. Therefore, the remainders are represented by the white space in each column. Figure 4 shows the corresponding remainders and adjusted remainders.

4.1.2. The Modified Objective of (Modified *k*-RSP$_1$).

We modified the objective function $z(\mathbf{x})$ by adding the adjusted remainders. Specifically, let the objective function $g(\mathbf{x}^L)$ be defined for any valid assignment of large items $\mathbf{x}^L$:

$$g(\mathbf{x}^L) = z(\mathbf{x}^L) + \sum_{\ell=1}^{k} \bar{R}_\ell(\mathbf{x}^L) = \sum_{j=1}^{k}(k-j+1)Q_j(\mathbf{x}^L) + \sum_{\ell=1}^{k} \bar{R}_\ell(\mathbf{x}^L).$$

An important property of function $g(\cdot)$ is that, for $\mathbf{x}$ feasible for (k-RSP$_1$) and $\mathbf{x}^L$, its projection to the large items set, $g(\mathbf{x}^L)$, is an upper bound of $z(\mathbf{x})$ as proved next.

Lemma 5. *For any feasible solution* $\mathbf{x}$ *of* (k-RSP$_1$) *and* $\mathbf{x}^L = P_{I_L}(\mathbf{x})$, $g(\mathbf{x}^L) \ge z(\mathbf{x})$.

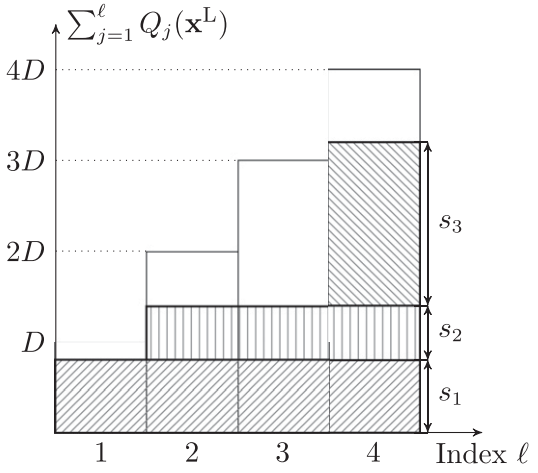
Proof. To show $g(\mathbf{x}^L) = z(\mathbf{x}^L) + \sum_{\ell=1}^{k} \bar{R}_\ell(\mathbf{x}^L) \ge z(\mathbf{x})$, it suffices to prove that, for $\mathbf{x}^S = P_{I_S}(\mathbf{x})$, $z(\mathbf{x}^S) = z(\mathbf{x}) - z(\mathbf{x}^L) \le \sum_{\ell=1}^{k} \bar{R}_\ell(\mathbf{x}^L)$.

By the definition of adjusted remainders, any $\mathbf{x}$ feasible for (k-RSP$_1$) satisfies $\sum_{j=1}^{\ell} Q_j(\mathbf{x}^S) \le \bar{R}_\ell(\mathbf{x}^L)$ for $\ell = 1,\ldots,k$. Therefore, $z(\mathbf{x}^S) = \sum_{j=1}^{k}(k-j+1)Q_j(\mathbf{x}^S) = \sum_{j=1}^{k}\sum_{\ell=j}^{k} Q_j(\mathbf{x}^S) = \sum_{\ell=1}^{k}\sum_{j=1}^{\ell} Q_j(\mathbf{x}^S) \le \sum_{\ell=1}^{k} \bar{R}_\ell(\mathbf{x}^L)$. □

Table 3. A Problem Instance of Single-Cycle RSP with $k = 4$

i	1	2	3	4	5
s_i	8	6	18	3	5
Large/small	Large	Large	Large	Small	Small

Figure 3. An Assignment of Large Items for the Example in Table 3



4.1.3. The Scaling of (Modified k RSP$_1$) (Scaled Modified k RSP$_1$).

The data are scaled by the factor $\frac{1}{\epsilon^2 D}$ as follows. Let $s'_i = \lfloor \frac{s_i}{\epsilon^2 D} \rfloor$ be the scaled sizes of items $i = 1,\ldots,n$ and $D' = \frac{D}{\epsilon^2 D} = \frac{1}{\epsilon^2}$ be the scaled demand. Let $Q'_j(\mathbf{x}^L)$, $\bar{R}'_\ell(\mathbf{x}^L)$, and $g'(\mathbf{x}^L)$ denote the "scaled" replenishment sizes at time j, the adjusted remainder at time ℓ, and the objective function for the scaled sizes s'_i:

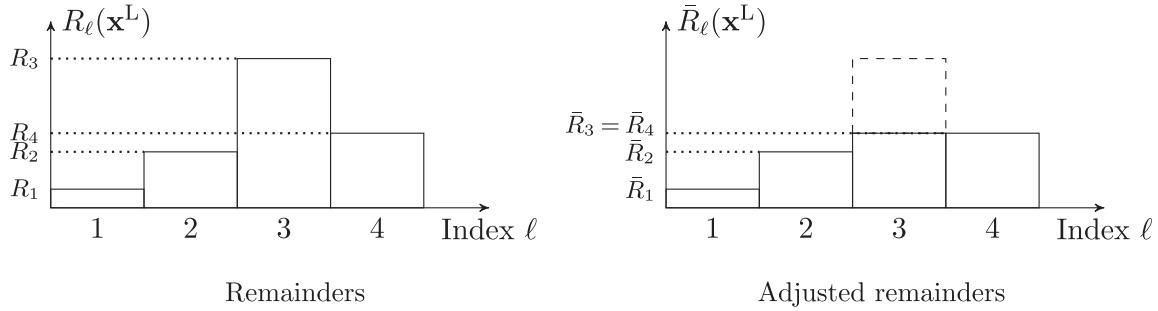
$$Q'_j(\mathbf{x}^L) = \sum_{i\in I_L} s'_i x^L_{ij}, j = 1,..,k;$$
$$R'_\ell(\mathbf{x}^L) = \ell D' - \sum_{j=1}^{\ell} Q'_j(\mathbf{x}^L), \ell = 1,..,k;$$
$$\bar{R}'_\ell(\mathbf{x}^L) = \min_{j\ge\ell} R'_j(\mathbf{x}^L), \ell = 1,..,k;$$
$$g'(\mathbf{x}^L) = \sum_{j=1}^{k}(k-j+1)Q'_j(\mathbf{x}^L) + \sum_{\ell=1}^{k} \bar{R}'_\ell(\mathbf{x}^L).$$

Let the large items be reindexed from 1 to n_L. The scaled problem that is solved to determine the assignment of the large items is (scaled modified k RSP$_1$) formulated as follows:

$$\begin{aligned}
\text{(scaled modified k RSP}_1\text{)} \quad \max\ & g'(\mathbf{x}^L) \\
\text{subject to}\ & \sum_{j=1}^{\ell} Q'_j(\mathbf{x}^L) \le \ell D' \\
& \ell = 1,..,k \\
& \sum_{j=1}^{k} x^L_{ij} = 1 \\
& i = 1,\ldots,n_L \\
& x^L_{ij} \text{ binary for} \\
& i = 1,\ldots,n_L, \\
& j = 1,..,k.
\end{aligned}$$

The optimal solution for (scaled modified k RSP$_1$) is found by applying the dynamic programming procedure in Section 3.2 with scaled sizes.

Figure 4. The Remainders and the Adjusted Remainders Corresponding to Figure 3



Algorithm 1. (Large assignment $(D', s'_1, \ldots, s'_{n_L})$)
Initialize: $f_0(q_1, q_2, \ldots, q_k) \leftarrow 0$ for any $(q_1, q_2, \ldots, q_k)$ with $q_j \in [0, jD']$ and q_j integer.
for $h = 1, \ldots, n_L$, **do**
for $j = 1, \ldots, k$ and for each $q_j \in \{0, 1, \ldots jD'\}$, **do**
$f_h(q_1, q_2, \ldots, q_k) \leftarrow$

$$\begin{cases} \max\limits_{\tau=1,\ldots,k} \{(k+1-\tau)s'_h \\ \quad + f_{h-1}(q'_1(\tau), \ldots, q'_k(\tau))\}, & \text{if } q'_\ell(\tau) \geq 0 \text{ for all } \ell \\ -\infty & \text{otherwise,} \end{cases}$$

where $q'_\ell(\tau) = q_\ell - \left\lfloor \dfrac{\ell - \tau + k_h}{k_h} \right\rfloor s_h$.
end for
end for
Output $\hat{\mathbf{x}}^L$, the solution attaining the value $\max_{(q_1,\ldots,q_k)} \cdot f_{n_L}(q_1, \ldots, q_k) + \sum_{\ell=1}^{k} \bar{r}_j$, where $\bar{r}_\ell = \min_{j \geq \ell}(jD' - q_j)$.

The complexity of algorithm Large assignment, with the scaled sizes, is $O(k^2 \cdot k! \cdot D'^k \cdot n_L) = O(\frac{n}{\epsilon^{2k}})$.

Next, we define the (ϵ-relaxed RSP) and then prove that any feasible solution for (scaled modified k RSP$_1$), including $\hat{\mathbf{x}}^L$, is feasible for (ϵ-relaxed RSP).

4.2. The ϵ-Relaxed RSP

The (ϵ-relaxed RSP) formulation allows the cascading constraints to be violated by up to ϵkD as follows:

$$\begin{aligned} (\epsilon\text{-relaxed RSP}) \qquad \max\ & z(\mathbf{x}) = \sum_{j=1}^{k} (k-j+1) Q_j(\mathbf{x}) \\ \text{subject to } & \sum_{j=1}^{\ell} Q_j(\mathbf{x}) \leq \ell D + \epsilon k D \qquad \ell = 1, .., k \\ & \sum_{j=1}^{k} x_{ij} = 1 \qquad i = 1, \ldots, n \\ & x_{ij} \text{ binary for } i = 1, \ldots, n, \quad j = 1, .., k. \end{aligned}$$

We refer to the constraints $\sum_{j=1}^{\ell} Q_j(\mathbf{x}) \leq \ell D + \epsilon k D$ as the *ϵ-relaxed cascading constraints*. We next show that the effect of the ϵ-relaxed cascading constraints on the optimal solution is at most ϵkD.

Lemma 6. *The peak inventory level of any feasible solution* $\mathbf{x}$ *to* (ϵ-relaxed RSP) *is at most* $V_k(\mathbf{x}) + \epsilon kD$.

Proof. Any feasible solution $\mathbf{x}$ for (ϵ-relaxed RSP) is a valid assignment, and therefore, Lemma 3 applies. That is, $V_\ell(\mathbf{x}) = V_k(\mathbf{x}) + \left(\sum_{j=1}^{\ell} Q_j(\mathbf{x}) - \ell D\right)$ for all ℓ. The ϵ-relaxed cascading constraints state that $\sum_{j=1}^{\ell} Q_j(\mathbf{x}) - \ell D \leq \epsilon k D$ for all ℓ. Therefore, when $\mathbf{x}$ is a feasible solution of (ϵ-relaxed RSP), $V_\ell(\mathbf{x}) \leq V_k(\mathbf{x}) + \epsilon kD$ for all ℓ, and hence, $V(\mathbf{x}) = \max_\ell V_\ell(\mathbf{x}) \leq V_k(\mathbf{x}) + \epsilon kD$. □

The next lemma proves that any feasible solution for (scaled modified k RSP$_1$), including $\hat{\mathbf{x}}^L$, is feasible for (ϵ-relaxed RSP).

Lemma 7. *Any valid assignment of large items* $\mathbf{x}^L$ *that is feasible for* (scaled modified k RSP$_1$) *satisfies* $\sum_{j=1}^{\ell} \cdot Q_j(\mathbf{x}^L) \leq \ell D + \epsilon k D$ *for* $\ell = 1, .., k$.

Proof. By definition, $s'_i = \lfloor \frac{s_i}{\epsilon^2 D} \rfloor$. Therefore, $s_i < \epsilon^2 D(s'_i + 1)$, and thus,

$$\begin{aligned} \sum_{j=1}^{\ell} Q_j(\mathbf{x}^L) &= \sum_{j=1}^{\ell} \sum_{i=1}^{n_L} s_i \mathbf{x}^L_{ij} \leq \sum_{j=1}^{\ell} \sum_{i=1}^{n_L} \epsilon^2 D(s'_i + 1) \mathbf{x}^L_{ij} \\ &= \epsilon^2 D \left(\sum_{j=1}^{\ell} \sum_{i=1}^{n_L} s'_i \mathbf{x}^L_{ij} + \sum_{j=1}^{\ell} \sum_{i=1}^{n_L} \mathbf{x}^L_{ij} \right). \end{aligned} \tag{5}$$

Because $\mathbf{x}^L$ is feasible for (scaled large packing) and $D' = \frac{1}{\epsilon^2}$,

$$\sum_{j=1}^{\ell} \sum_{i=1}^{n_L} s'_i \mathbf{x}^L_{ij} = \sum_{j=1}^{\ell} Q'_j(\mathbf{x}^L) \leq \ell D' = \frac{\ell}{\epsilon^2}. \tag{6}$$

Because the reorder size for any large item is greater than ϵD and the sum of reorder sizes of large items is bounded by the total reorder sizes kD, we can infer that the number of large items is bounded by $n_L < \frac{k}{\epsilon}$. From feasibility of x^L for (scaled large packing), we also know that $\sum_{j=1}^{k} x^L_{ij} = 1$ for any large item i. Hence, for $\ell = 1, \ldots, k$,

$$\sum_{j=1}^{\ell} \sum_{i=1}^{n_L} \mathbf{x}^L_{ij} \leq \sum_{j=1}^{k} \sum_{i=1}^{n_L} \mathbf{x}^L_{ij} = \sum_{i=1}^{n_L} \left(\sum_{j=1}^{k} \mathbf{x}^L_{ij} \right) = n_L < \frac{k}{\epsilon}. \tag{7}$$

Hence, from inequalities (5), (6), and (7),

$$\sum_{j=1}^{\ell} Q_j(\mathbf{x}^L) \leq \epsilon^2 D \left(\frac{\ell}{\epsilon^2} + \frac{k}{\epsilon} \right) = \ell D + \epsilon k D. \quad \square$$

Using the relationship between reorder sizes s_i and the scaled sizes s'_i, we show that, for any feasible solution of (scaled large packing), $\mathbf{x}^{\mathrm{L}}$, the objective with original sizes $g(\mathbf{x}^{\mathrm{L}})$ is closely approximated by the objective with scaled sizes $g'(\mathbf{x}^{\mathrm{L}}) = \sum_{j=1}^{k}(k-j+1)\cdot Q'_j(\mathbf{x}^{\mathrm{L}}) + \sum_{\ell=1}^{k}\bar{R}'_\ell(\mathbf{x}^{\mathrm{L}})$ corrected for the scaling factor $\epsilon^2 D$.

4.2.1. The Approximation Property of the Solution to (Scaled Modified k RSP$_1$).

Theorem 2. *For any assignment of large items $\mathbf{x}^{\mathrm{L}}$ feasible for (*scaled modified k RSP$_1$*), the values of the objective function with original and scaled sizes $g(\mathbf{x}^{\mathrm{L}})$ and $g'(\mathbf{x}^{\mathrm{L}})$, respectively, satisfy*

$$\epsilon^2 D g'(\mathbf{x}^{\mathrm{L}}) - \frac{k^2(k+1)D}{2}\cdot\frac{\epsilon}{1-\epsilon} \le g(\mathbf{x}^{\mathrm{L}}) \le \epsilon^2 D g'(\mathbf{x}^{\mathrm{L}}) + \frac{k(k+1)(k+2)D}{6}\cdot\frac{\epsilon}{1-\epsilon}.$$

Proof. Recall that $s'_i = \lfloor \frac{s_i}{\epsilon^2 D} \rfloor$, and therefore, $\epsilon^2 D s'_i \le s_i < \epsilon^2 D(s'_i+1)$.

The reorder size s_i of any large item i satisfies $s_i > \epsilon D$. Therefore, $s'_i > \lfloor \frac{\epsilon D}{\epsilon^2 D} \rfloor = \lfloor \frac{1}{\epsilon} \rfloor \ge \frac{1}{\epsilon} - 1$, and for any $\ell = 1,..,k$, $\sum_{j=1}^{\ell} Q'_j(\mathbf{x}^{\mathrm{L}}) = \sum_{j=1}^{\ell}\sum_{i=1}^{n_{\mathrm{L}}} s'_i x^{\mathrm{L}}_{ij} > \sum_{j=1}^{\ell}\sum_{i=1}^{n_{\mathrm{L}}}(\frac{1}{\epsilon}-1)x^{\mathrm{L}}_{ij}$. However, the feasibility of $\mathbf{x}^{\mathrm{L}}$ for (scaled large packing) implies that $\sum_{j=1}^{\ell} Q'_j(\mathbf{x}^{\mathrm{L}}) \le \ell D' = \frac{\ell}{\epsilon^2}$ for any ℓ. Together, we have that $\sum_{j=1}^{\ell}\sum_{i=1}^{n_{\mathrm{L}}}(\frac{1}{\epsilon}-1)x^{\mathrm{L}}_{ij} < \frac{\ell}{\epsilon^2}$ for any ℓ. Dividing both sides of this inequality by $(\frac{1}{\epsilon}-1)$ results in

$$\sum_{j=1}^{\ell}\sum_{i=1}^{n_{\mathrm{L}}} x^{\mathrm{L}}_{ij} < \frac{\ell}{\epsilon(1-\epsilon)} \quad \text{for } \ell = 1,\dots,k. \tag{8}$$

Because $s_i < \epsilon^2 D(s'_i+1)$ for any large item i, we have, for any integer time j,

$$\begin{aligned} Q_j(\mathbf{x}^{\mathrm{L}}) &= \sum_{i=1}^{n_{\mathrm{L}}} s_i x^{\mathrm{L}}_{ij} \\ &< \epsilon^2 D \sum_{i=1}^{n_{\mathrm{L}}}(s'_i+1)x^{\mathrm{L}}_{ij} \\ &= \epsilon^2 D\left(\sum_{i=1}^{n_{\mathrm{L}}} s'_i x^{\mathrm{L}}_{ij} + \sum_{i=1}^{n_{\mathrm{L}}} x^{\mathrm{L}}_{ij}\right) \\ &= \epsilon^2 D\left(Q'_j(\mathbf{x}^{\mathrm{L}}) + \sum_{i=1}^{n_{\mathrm{L}}} x^{\mathrm{L}}_{ij}\right). \end{aligned} \tag{9}$$

The second term in the parentheses, $\sum_{i=1}^{n_{\mathrm{L}}} x^{\mathrm{L}}_{ij}$, is less than or equal to the left-hand side of inequality (8) for any $\ell \ge j$. Therefore, we derive the inequality

$$\sum_{i=1}^{n_{\mathrm{L}}} x^{\mathrm{L}}_{ij} < \frac{j}{\epsilon(1-\epsilon)} \quad \text{for } j = 1,\dots,k. \tag{10}$$

Using inequality (10) in inequality (9), we get

$$\begin{aligned} Q_j(\mathbf{x}^{\mathrm{L}}) &< \epsilon^2 D\left(Q'_j(\mathbf{x}^{\mathrm{L}}) + \frac{j}{\epsilon(1-\epsilon)}\right) \\ &= \epsilon^2 D Q'_j(\mathbf{x}^{\mathrm{L}}) + jD\cdot\frac{\epsilon}{1-\epsilon} \quad \text{for } j = 1,\dots,k. \end{aligned} \tag{11}$$

As $s_i \ge \epsilon^2 D s'_i$ for any i,

$$Q_j(\mathbf{x}^{\mathrm{L}}) = \sum_{i=1}^{n_{\mathrm{L}}} s_i x^{\mathrm{L}}_{ij} \ge \epsilon^2 D \sum_{i=1}^{n_{\mathrm{L}}} s'_i x^{\mathrm{L}}_{ij} = \epsilon^2 D Q'_j(\mathbf{x}^{\mathrm{L}}) \quad \text{for } j = 1,\dots,k. \tag{12}$$

Recall that adjusted remainder of time τ is $\bar{R}_\tau(\mathbf{x}^{\mathrm{L}}) = \min_{\ell\ge\tau} R_\ell = \min_{\ell\ge\tau}\left(\ell D - \sum_{j=1}^{\ell} Q_j(\mathbf{x}^{\mathrm{L}})\right)$ and that the scaled adjusted remainder of time τ is $\bar{R}'_\tau(\mathbf{x}^{\mathrm{L}}) = \min_{\ell\ge\tau}\left(\ell D' - \sum_{j=1}^{\ell} Q'_j(\mathbf{x}^{\mathrm{L}})\right)$. We derive from inequality (11) that, for any time τ,

$$\begin{aligned} \bar{R}_\tau(\mathbf{x}^{\mathrm{L}}) &= \min_{\ell\ge\tau}\left(\ell D - \sum_{j=1}^{\ell} Q_j(\mathbf{x}^{\mathrm{L}})\right) \\ &\ge \min_{\ell\ge\tau}\left[\ell D - \sum_{j=1}^{\ell}\left(\epsilon^2 D Q'_j(\mathbf{x}^{\mathrm{L}}) + jD\cdot\frac{\epsilon}{1-\epsilon}\right)\right] \\ &= \min_{\ell\ge\tau}\left[\ell D - \epsilon^2 D\sum_{j=1}^{\ell} Q'_j(\mathbf{x}^{\mathrm{L}}) - \frac{\ell(\ell+1)}{2}\cdot D\cdot\frac{\epsilon}{1-\epsilon}\right] \\ &\ge \min_{\ell\ge\tau}\left[\ell D - \epsilon^2 D\sum_{j=1}^{\ell} Q'_j(\mathbf{x}^{\mathrm{L}})\right] \\ &\quad - \max_{\ell\ge\tau}\left[\frac{\ell(\ell+1)}{2}\cdot D\cdot\frac{\epsilon}{1-\epsilon}\right] \\ &= \min_{\ell\ge\tau}\left[\epsilon^2 D\left(\ell D' - \sum_{j=1}^{\ell} Q'_j(\mathbf{x}^{\mathrm{L}})\right)\right] \\ &\quad - \frac{k(k+1)}{2}\cdot D\cdot\frac{\epsilon}{1-\epsilon} \\ &= \epsilon^2 D\bar{R}'_\tau(\mathbf{x}^{\mathrm{L}}) - \frac{k(k+1)}{2}\cdot D\cdot\frac{\epsilon}{1-\epsilon}. \end{aligned} \tag{13}$$

Additionally, we derive from inequality (12) that, for any time τ,

$$\begin{aligned} \bar{R}_\tau(\mathbf{x}^{\mathrm{L}}) &= \min_{\ell\ge\tau}\left(\ell D - \sum_{j=1}^{\ell} Q_j(\mathbf{x}^{\mathrm{L}})\right) \\ &\le \min_{\ell\ge\tau}\left(\ell D - \epsilon^2 D\sum_{j=1}^{\ell} Q'_j(\mathbf{x}^{\mathrm{L}})\right) \\ &= \min_{\ell\ge\tau}\left[\epsilon^2 D\left(\ell D' - \sum_{j=1}^{\ell} Q'_j(\mathbf{x}^{\mathrm{L}})\right)\right] \\ &= \epsilon^2 D\bar{R}'_\tau(\mathbf{x}^{\mathrm{L}}). \end{aligned} \tag{14}$$

Using the inequalities (11) and (14), we prove the upper bound on $g(\mathbf{x}^L)$ as follows:

$$\begin{aligned} g(\mathbf{x}^L) &= \sum_{j=1}^{k}(k-j+1)Q_j(\mathbf{x}^L) + \sum_{\tau=1}^{k}\bar{R}_\tau(\mathbf{x}^L) \\ &< \sum_{j=1}^{k}(k-j+1)\left(\epsilon^2 DQ'_j(\mathbf{x}^L) + jD\cdot\frac{\epsilon}{1-\epsilon}\right) \\ &\quad + \sum_{\tau=1}^{k}\epsilon^2 D\bar{R}'_\tau(\mathbf{x}^L) \\ &= \left[\sum_{j=1}^{k}(k-j+1)\epsilon^2 DQ'_j(\mathbf{x}^L) + \sum_{\tau=1}^{k}\epsilon^2 D\bar{R}'_\tau(\mathbf{x}^L)\right] \\ &\quad + \sum_{j=1}^{k}(k-j+1)jD\cdot\frac{\epsilon}{1-\epsilon} \\ &= \epsilon^2 Dg'(\mathbf{x}^L) + \left[(k+1)\cdot\sum_{j=1}^{k}j - \sum_{j=1}^{k}j^2\right]D\cdot\frac{\epsilon}{1-\epsilon} \\ &= \epsilon^2 Dg'(\mathbf{x}^L) + \left[\frac{k(k+1)^2}{2} - \frac{k(k+1)(2k+1)}{6}\right]D \\ &\quad \cdot\frac{\epsilon}{1-\epsilon} \\ &= \epsilon^2 Dg'(\mathbf{x}^L) + \frac{k(k+1)(k+2)D}{6}\cdot\frac{\epsilon}{1-\epsilon}. \end{aligned}$$

The lower bound on $g(\mathbf{x}^L)$ follows from inequalities (12) and (13):

$$\begin{aligned} g(\mathbf{x}^L) &= \sum_{j=1}^{k}(k-j+1)Q_j(\mathbf{x}^L) + \sum_{\tau=1}^{k}\bar{R}_\tau(\mathbf{x}^L) \\ &\geq \sum_{j=1}^{k}(k-j+1)\epsilon^2 DQ'_j(\mathbf{x}^L) \\ &\quad + \sum_{\tau=1}^{k}\left(\epsilon^2 D\bar{R}'_\tau(\mathbf{x}^L) - \frac{k(k+1)}{2}\cdot D\cdot\frac{\epsilon}{1-\epsilon}\right) \\ &= \left[\sum_{j=1}^{k}(k-j+1)\epsilon^2 DQ'_j(\mathbf{x}^L) + \sum_{\tau=1}^{k}\epsilon^2 D\bar{R}'_\tau(\mathbf{x}^L)\right] \\ &\quad - \sum_{\tau=1}^{k}\frac{k(k+1)}{2}\cdot D\cdot\frac{\epsilon}{1-\epsilon} \\ &= \epsilon^2 Dg'(\mathbf{x}^L) - \frac{k^2(k+1)D}{2}\cdot\frac{\epsilon}{1-\epsilon}. \end{aligned}$$

This completes the proof of the statement of the theorem. □

Theorem 2 leads to the following lower bound on $g(\hat{\mathbf{x}}^L)$ for $\hat{\mathbf{x}}^L$ being an optimal solution of (scaled modified k RSP_1).

Theorem 3. *For any feasible solution* $\mathbf{x}$ *of* (k-RSP_1) *and* $\mathbf{x}^L = P_{I_L}(\mathbf{x})$, $g(\hat{\mathbf{x}}^L) \geq g(\mathbf{x}^L) - \delta(\epsilon)$, *where* $\delta(\epsilon) = \frac{k(k+1)(2k+1)D}{3}\cdot\frac{\epsilon}{1-\epsilon}$.

Proof. By Theorem 2, we have the lower bound of $g(\hat{\mathbf{x}}^L)$:

$$g(\hat{\mathbf{x}}^L) \geq \epsilon^2 Dg'(\hat{\mathbf{x}}^L) - \frac{k^2(k+1)D}{2}\cdot\frac{\epsilon}{1-\epsilon}.$$

Because for any feasible solution of (k-RSP_1), the projection to the large items set, $\mathbf{x}^L$, is also feasible for (scaled modified k-RSP_1), we use the upper bound of $g(\mathbf{x}^L)$ from Theorem 2 to get

$$\epsilon^2 Dg'(\mathbf{x}^L) \geq g(\mathbf{x}^L) - \frac{k(k+1)(k+2)D}{6}\cdot\frac{\epsilon}{1-\epsilon}.$$

Because $\hat{\mathbf{x}}^L$ is optimal for (scaled modified k-RSP_1), it follows that $g'(\hat{\mathbf{x}}^L) \geq g'(\mathbf{x}^L)$. Combining the three inequalities, we get

$$\begin{aligned} g(\hat{\mathbf{x}}^L) &\geq \epsilon^2 Dg'(\hat{\mathbf{x}}^L) - \frac{k^2(k+1)D}{2}\cdot\frac{\epsilon}{1-\epsilon} \\ &\geq \epsilon^2 Dg'(\mathbf{x}^L) - \frac{k^2(k+1)D}{2}\cdot\frac{\epsilon}{1-\epsilon} \\ &\geq g(\mathbf{x}^L) - \frac{k(k+1)(k+2)D}{6}\cdot\frac{\epsilon}{1-\epsilon} - \frac{k^2(k+1)D}{2} \\ &\quad \cdot\frac{\epsilon}{1-\epsilon} \\ &= g(\mathbf{x}^L) - \frac{k(k+1)(2k+1)D}{3}\cdot\frac{\epsilon}{1-\epsilon} \\ &= g(\mathbf{x}^L) - \delta(\epsilon). \quad \square \end{aligned}$$

For any feasible solution $\mathbf{x}$ of (k-RSP_1) and $\mathbf{x}^L = P_{I_L}(\mathbf{x})$, Lemma 5 states that $g(\mathbf{x}^L) \geq z(\mathbf{x})$. Hence, a corollary of Theorem 3 and Lemma 5 is as follows.

Corollary 1. *For any feasible solution* $\mathbf{x}$ *of* (k-RSP_1), $g(\hat{\mathbf{x}}^L) \geq z(\mathbf{x}) - \delta(\epsilon)$, *where* $\delta(\epsilon) = \frac{k(k+1)(2k+1)D}{3}\cdot\frac{\epsilon}{1-\epsilon}$.

Consequently, the algorithm Large assignment yields an output $\hat{\mathbf{x}}^L$ such that $g(\hat{\mathbf{x}}^L)$ is at least as large as the optimal objective of (k-RSP_1) minus $\delta(\epsilon)$.

4.3. The Assignment of Small Items

Given the assignment of large items $\hat{\mathbf{x}}^L$ and the corresponding adjusted remainders $\bar{R}_\ell(\mathbf{x}^L)$, we derive a valid assignment of the small items so that the joint large and small items assignment is a feasible solution to (ϵ-relaxed RSP).

As stated in Section 4.1, the cascading constraints are equivalent to

$$\sum_{j=1}^{\ell_1} Q_j(\mathbf{x}^S) \leq \bar{R}_\ell(\mathbf{x}^L) \qquad \text{for} \quad \ell = 1,\ldots,k. \tag{15}$$

The following greedy procedure assigns any yet unassigned small item to the lowest integer index time with positive adjusted remainder. For simplicity, the

small items are reindexed 1 to n_S in the procedure's description provided below.

Algorithm 2 (SMALL ASSIGNMENT ($\bar{R}_1(\hat{\mathbf{x}}^L), \ldots, \bar{R}_k(\hat{\mathbf{x}}^L), s_1, \ldots, s_{n_S}$))

$\tilde{R}_\ell(0) \leftarrow \bar{R}_\ell(\hat{\mathbf{x}}^L) \quad \ell = 1, \ldots, k.$
for $i = 1, \ldots, n_S$, **do**
  $j(i) \leftarrow \min\{\ell | \tilde{R}_\ell(i-1) > 0\}$.
  Assign item i to time $j(i)$.
  $\tilde{R}_\ell(i) \leftarrow \tilde{R}_\ell(i-1) \quad \ell = 1, \ldots, j(i)-1.$
  $\tilde{R}_\ell(i) \leftarrow \tilde{R}_\ell(i-1) - s_i \quad \ell = j(i), \ldots, k.$
end for

Let $\hat{\mathbf{x}}^S$ denote the output assignment of the small items of SMALL ASSIGNMENT.

Lemma 8. *Algorithm* SMALL ASSIGNMENT *is correct, and its complexity is linear,* $O(n_S)$.

Proof. To prove correctness, one needs to prove that the algorithm terminates only after all small items were assigned, meaning that there is always a positive adjusted remainder available for each small item.

The notation used in SMALL ASSIGNMENT, $\tilde{R}_\ell(h)$, is the slack of cascading constraint (15) at time ℓ after iteration h. That is, $\tilde{R}_\ell(h) = \bar{R}_\ell(\hat{\mathbf{x}}^L) - \sum_{j=1}^{\ell}\sum_{i=1}^{h} s_i \hat{x}_{ij}^S$. Also, $\tilde{R}_\ell(n_S) = \bar{R}_\ell(\hat{\mathbf{x}}^L) - \sum_{j=1}^{\ell}\sum_{i=1}^{n_S} s_i \hat{x}_{ij}^S = \bar{R}_\ell(\hat{\mathbf{x}}^L) - \sum_{j=1}^{\ell} Q(\hat{\mathbf{x}}^S)$.

By the definition of remainders and adjusted remainders, it follows that the initial slack corresponding to day k is $\tilde{R}_k(0) = \bar{R}_k(\hat{\mathbf{x}}^L) = R_k(\hat{\mathbf{x}}^L) = kD - \sum_{i\in I_L} s_i = \sum_{i\in I_S} s_i$. In each iteration of SMALL ASSIGNMENT, the slack corresponding to day k is reduced by the reorder size of the small item assigned at that iteration. Therefore, the slack of day k is positive until all small items have been assigned. That is, SMALL ASSIGNMENT guarantees an assignment of all of the small items.

The complexity of SMALL ASSIGNMENT is linear, because there are n_S iterations, each of which runs in $O(1)$ steps when k is a constant as assumed here. □

As an illustrative example, consider the adjusted remainders in Figure 4 and the small items in Table 3. SMALL ASSIGNMENT assigns item 4 to time 1 and item 5 to time 2. Figure 5 visualizes this assignment example of small items. Columns heights correspond to the values of the adjusted remainders. The rectangles with patterns in each column are the small items that are replenished on or before the indexed time. The combined assignment of all items is shown by Figure 6. In Figure 6(a), the column heights are truncated by the same amount as the difference between remainder and adjusted remainders. Figure 6(b) combines the assignment of large items and small items in the truncated columns.

The update of the slacks $\tilde{R}_\ell(h)$ in the algorithm is such that the following properties hold.

Property 1. *The slacks before the assigning timing* $j(h)$ *are unaffected.*

Property 2. *The slacks of time* $j(h)+1, \ldots, k$ *are reduced by the same amount as the slack of assigning timing* $j(h)$.

We note that Property 1 is not satisfied by the (RSP_0) formulation and that Property 2 does not hold for multi-cycle RSP.

From the two properties, it follows that the positive slacks at each iteration are nondecreasing in the integer indices of time from one to k.

Lemma 9. *For any* $1 \le h \le n_S$, $0 < \tilde{R}_{j(h)}(h-1) \le \tilde{R}_{j(h)+1}(h-1) \le \ldots \le \tilde{R}_k(h-1)$.

Proof. We prove this by induction on the iteration index.

Base: Because the adjusted remainders are defined as $\bar{R}_\ell(\hat{\mathbf{x}}^L) = \min_{j\ge\ell} R_\ell(\hat{\mathbf{x}}^L)$, we have $\bar{R}_1(\hat{\mathbf{x}}^L) \le \bar{R}_2(\hat{\mathbf{x}}^L) \le \ldots \le \bar{R}_k(\hat{\mathbf{x}}^L)$. Additionally, as $\tilde{R}_\ell(0) = \bar{R}_\ell(\hat{\mathbf{x}}^L)$ for all ℓ, we also have $\tilde{R}_1(0) \le \tilde{R}_2(0) \le \ldots \le \tilde{R}_k(0)$. The algorithm SMALL ASSIGNMENT assigns $j(1)$ to be the smallest integer index such that $\tilde{R}_{j(1)}(0) > 0$. Therefore, $0 < \tilde{R}_{j(1)}(0) \le \tilde{R}_{j(1)+1}(0) \le \ldots \le \tilde{R}_k(0)$, and the statement is true for $h = 1$.

Inductive step: Assume that the statement holds for h, $1 \le h \le n_S - 1$; now, we prove for $h+1$. Because $j(h+1) \ge j(h)$, the assumption implies that $\tilde{R}_{j(h+1)}(h-1) \le \tilde{R}_{j(h+1)+1}(h-1) \le \ldots \le \tilde{R}_k(h-1)$. Because $\tilde{R}_\ell(h) = \tilde{R}_\ell(h-1) - s_h$ for all $\ell \ge j(h)$, $\tilde{R}_{j(h+1)}(h) \le \tilde{R}_{j(h+1)+1}(h) \le \ldots \le \tilde{R}_k(h)$. Moreover, by the algorithm SMALL ASSIGNMENT, we have $\tilde{R}_{j(h+1)}(h) > 0$. Therefore, $0 < \tilde{R}_{j(h+1)}(h) \le \tilde{R}_{j(h+1)+1}(h) \le \ldots \le \tilde{R}_k(h)$. □

Figure 5. (a) Adjusted Remainders and (b) Assignment of Small Items for the Example in Table 3 by SMALL ASSIGNMENT

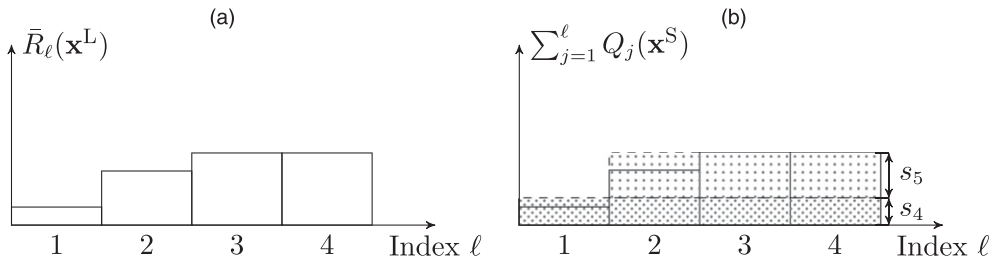
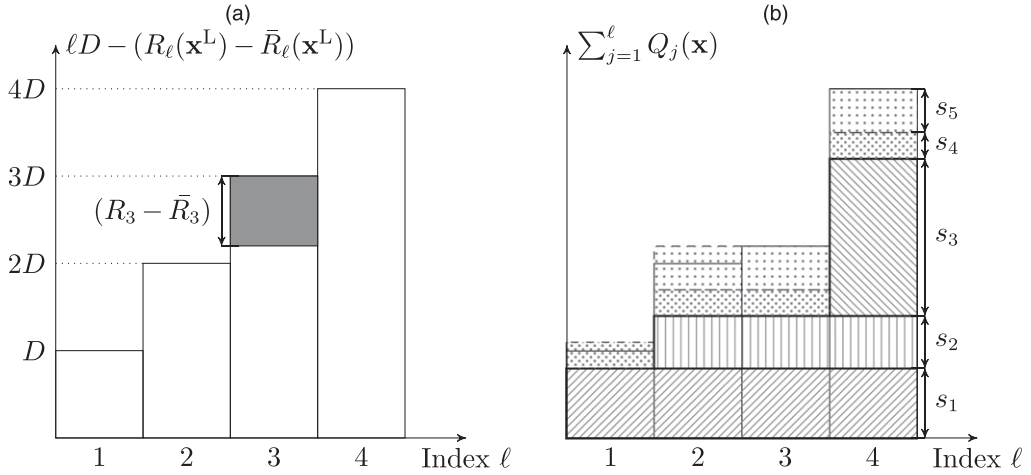


Figure 6. Assignment of All Items in Table 3



Next, we show that the joint assignment of large items with the output assignment of small items $\hat{\mathbf{x}}^S$ yields a feasible solution for (ϵ-relaxed RSP).

Lemma 10. *The assignment* $\hat{\mathbf{x}} = \hat{\mathbf{x}}^L + \hat{\mathbf{x}}^S$ *is feasible for* (ϵ-relaxed RSP).

Proof. It is easy to see that $\hat{\mathbf{x}}$ is valid. It remains to show that $\hat{\mathbf{x}}$ satisfies the ϵ-relaxed cascading constraints.

By Lemma 7, we know that $\sum_{j=1}^{\ell} Q_j(\mathbf{x}^L) \le \ell D + \epsilon k D$, and therefore, $R_\ell(\hat{\mathbf{x}}^L) = \ell D - \sum_{j=1}^{\ell} Q_j(\mathbf{x}^L) \ge -\epsilon k D$ for any ℓ. Thus, $\bar{R}_\ell(\hat{\mathbf{x}}^L) = \min_{j \ge \ell} R_j(\hat{\mathbf{x}}^L) \ge -\epsilon k D$ for any ℓ.

By Lemma 9, only the positive slacks could be reduced at each iteration. Additionally, because the reorder sizes of small items are less than or equal to ϵD, the reduction is at most ϵD. Thus, $\tilde{R}_\ell(h) \ge -\epsilon k D$ for any ℓ and h, meaning that $\sum_{j=1}^{\ell} Q(\hat{\mathbf{x}}^S) = \bar{R}_\ell(\hat{\mathbf{x}}^L) - \tilde{R}_\ell(n_S) \le \bar{R}_\ell(\hat{\mathbf{x}}^L) + \epsilon k D$.

By definition of $\bar{R}_\ell(\hat{\mathbf{x}}^L)$, for any ℓ, $\bar{R}_\ell(\hat{\mathbf{x}}^L) \le R_\ell(\hat{\mathbf{x}}^L) = \ell D - \sum_{j=1}^{\ell} Q(\hat{\mathbf{x}}^L)$, and therefore, $\sum_{j=1}^{\ell} Q(\hat{\mathbf{x}}^S) + \sum_{j=1}^{\ell} Q(\hat{\mathbf{x}}^L) \le \ell D + \epsilon k D$. □

Lemma 11. $\tilde{R}_\ell(n_S) \le 0$ *for* $\ell = 1, \ldots, k$.

Proof. By contradiction, suppose that, at the conclusion of the algorithm Small assignment, there exists an index ℓ so that $\tilde{R}_\ell(n_S) > 0$. This implies that $j(h) \le \ell$ for any h, meaning that all small items are assigned to the first ℓ integer times. In that case, $\sum_{j=1}^{\ell} Q(\hat{\mathbf{x}}^S) = \sum_{h \in I_S} s_h$. Therefore, $\tilde{R}_\ell(n_S) = \bar{R}_\ell(\hat{\mathbf{x}}^L) - \sum_{j=1}^{\ell} Q(\hat{\mathbf{x}}^S) = \bar{R}_\ell(\hat{\mathbf{x}}^L) - \sum_{h \in I_S} \cdot s_h \le \bar{R}_k(\hat{\mathbf{x}}^L) - \sum_{i=1}^{n_S} s_i = \left(kD - \sum_{h \in I_L} s_h\right) - \sum_{h \in I_S} s_h = 0$, contradicting the assumption. □

Theorem 4. *Given the assignment of large items* $\hat{\mathbf{x}}^L$ *that is optimal for* (scaled modified k RSP$_1$), Small assignment *will generate an output* $\hat{\mathbf{x}}^S$ *such that* $z(\hat{\mathbf{x}}^S) \ge \sum_{\ell=1}^{k} \bar{R}_\ell(\hat{\mathbf{x}}^L)$.

Proof. From Lemma 11, $\sum_{j=1}^{\ell} Q(\hat{\mathbf{x}}^S) = \bar{R}_\ell(\hat{\mathbf{x}}^L) - \tilde{R}_\ell(n_S) \ge \bar{R}_\ell(\hat{\mathbf{x}}^L)$. Therefore,

$$z(\hat{\mathbf{x}}^S) = \sum_{j=1}^{k} (k - j + 1) Q(\hat{\mathbf{x}}^S) = \sum_{\ell=1}^{k} \sum_{j=1}^{\ell} Q(\hat{\mathbf{x}}^S) \ge \sum_{\ell=1}^{k} \bar{R}_\ell(\hat{\mathbf{x}}^L). \quad \square$$

Theorem 4 together with Corollary 1 implies the following theorem.

Theorem 5. *Let* $\hat{\mathbf{x}} = \hat{\mathbf{x}}^L + \hat{\mathbf{x}}^S$, *where* $\hat{\mathbf{x}}^L$ *and* $\hat{\mathbf{x}}^S$ *are the outputs of algorithms* Large assignment *and* Small assignment, *respectively. Then, for any* $\mathbf{x}$ *that is feasible for* (k-RSP$_1$), $z(\hat{\mathbf{x}}) \ge z(\mathbf{x}) - \delta(\epsilon)$.

Proof. By Theorem 4, $z(\hat{\mathbf{x}}^S) \ge \sum_{\ell=1}^{k} \bar{R}_\ell(\hat{\mathbf{x}}^L)$, and therefore, $z(\hat{\mathbf{x}}) = z(\hat{\mathbf{x}}^L) + z(\hat{\mathbf{x}}^S) \ge z(\hat{\mathbf{x}}^L) + \sum_{\ell=1}^{k} \bar{R}_\ell(\hat{\mathbf{x}}^L) = g(\hat{\mathbf{x}}^L)$. Additionally, by Corollary 1, $g(\hat{\mathbf{x}}^L) \ge z(\mathbf{x}) - \delta(\epsilon)$ for any $\mathbf{x}$ that is feasible of (k-RSP$_1$). Therefore, $z(\hat{\mathbf{x}}) \ge g(\hat{\mathbf{x}}^L) \ge z(\mathbf{x}) - \delta(\epsilon)$ for any $\mathbf{x}$ that is feasible of (k-RSP$_1$). □

Therefore, assignment $\hat{\mathbf{x}}$ described in Theorem 5 attains an objective value $z(\hat{\mathbf{x}})$ that is at least as much as the optimal objective of (k-RSP$_1$) minus $\delta(\epsilon)$.

4.4. The $(1+\epsilon')$-Approximation Algorithm

Combining the theorems of the previous sections, the algorithms Large assignment and Small assignment deliver an FPTAS for single-cycle RSP: that is, a $(1+\epsilon')$-approximation algorithm for any $\epsilon' > 0$. The $(1+\epsilon')$-approximation algorithm consists of following steps.

Algorithm 3. ($(1+\epsilon')$-approximation)

Step 1. (Initialize) Let $\epsilon = \frac{\epsilon'}{3+\epsilon'}$; $D = \frac{1}{n}\sum_{i=1}^{n} s_i$;
(Partition the items into large and small) Let $I_L = \{i : s_i > \epsilon D\}$ and $I_S = \{i : s_i \le \epsilon D\}$;
(Scaling) Let $s'_i = \lfloor \frac{s_i}{\epsilon^2 D} \rfloor$ for $i \in I_L$, and let $D' = \frac{1}{\epsilon^2}$;

Step 2. Call LARGE ASSIGNMENT (D', s'_i for $i \in I_L$) to get $\hat{\mathbf{x}}^L$;
Compute $R_\ell(\mathbf{x}^L) = \ell D - \sum_{j=1}^{\ell} Q_j(\mathbf{x}^L)$ for $\ell = 1, \ldots, k$;
Compute $\bar{R}_\ell(\hat{\mathbf{x}}^L) = \min_{j \geq \ell} R_j(\mathbf{x}^L)$ for $\ell = 1, \ldots, k$;
Step 3. Apply SMALL ASSIGNMENT ($\bar{R}_1(\hat{\mathbf{x}}^L), \ldots, \bar{R}_k(\hat{\mathbf{x}}^L), s_i$ for $i \in I_S$) to get $\hat{\mathbf{x}}^S$;
Step 4. Output $\hat{\mathbf{x}}^{(\epsilon)} = \hat{\mathbf{x}}^L + \hat{\mathbf{x}}^S$.

Theorem 6. *The* $(1+\epsilon')$-APPROXIMATION *algorithm is a* $(1+\epsilon')$*-approximation algorithm for single-cycle* RSP.

Proof. Let $\mathbf{x}^*$ be an optimal solution of (k-RSP$_1$) and V^* be the corresponding peak inventory level.

As stated in Theorem 5, $z(\hat{\mathbf{x}}) \geq z(\mathbf{x}) - \delta(\epsilon)$ for any $\mathbf{x}$ that is feasible of (k-RSP$_1$), including $\mathbf{x}^*$. From Lemma 4, the inventory levels at time k for $\hat{\mathbf{x}}$ and $\mathbf{x}^*$ are $V_k(\hat{\mathbf{x}}) = \frac{C}{k} - \frac{z(\hat{\mathbf{x}})}{k}$ and $V_k(\mathbf{x}^*) = \frac{C}{k} - \frac{z(\mathbf{x}^*)}{k}$, respectively. Therefore,

$$V_k(\hat{\mathbf{x}}) = \frac{C}{k} - \frac{z(\hat{\mathbf{x}})}{k} \leq \frac{C}{k} - \frac{z(\mathbf{x}^*)}{k} + \frac{\delta(\epsilon)}{k} = V_k(\mathbf{x}^*) + \frac{\delta(\epsilon)}{k}.$$

From Lemma 6, it follows that the peak inventory level for $\hat{\mathbf{x}}$ satisfies $V(\hat{\mathbf{x}}) \leq V_k(\hat{\mathbf{x}}) + \epsilon k D$. Because $\mathbf{x}^*$ is a solution of (k-RSP$_1$), the peak inventory level for $\mathbf{x}^*$ is $V^* = V_k(\mathbf{x}^*)$. Hence,

$$V(\hat{\mathbf{x}}) \leq V_k(\hat{\mathbf{x}}) + \epsilon k D \leq V^* + \frac{\delta(\epsilon)}{k} + \epsilon k D.$$

That is, for the optimum peak storage of (k-RSP$_1$), V^*, and the output of our $(1+\epsilon')$-APPROXIMATION, $\hat{\mathbf{x}}$, the ratio $V(\hat{\mathbf{x}})/V^*$ is at most $1 + \left(\frac{\delta(\epsilon)}{k} + \epsilon k D\right)/V^*$. Because $V^* \geq \frac{k+1}{2}\sum_{i=1}^{n} s_i = \frac{k(k+1)}{2} D$, it follows that

$$\left(\frac{\delta(\epsilon)}{k} + \epsilon k D\right)/V^* \leq \frac{2}{k(k+1)D} \cdot \left(\frac{(k+1)(2k+1)D}{3} \cdot \frac{\epsilon}{1-\epsilon} + \epsilon k D\right).$$

Substituting the second term in the parentheses by $\epsilon k D < k D \cdot \frac{\epsilon}{1-\epsilon}$ and canceling D, we get

$$\left(\frac{\delta(\epsilon)}{k} + \epsilon k D\right)/V^* \leq \frac{2}{k(k+1)} \cdot \left(\frac{(k+1)(2k+1)}{3} \cdot \frac{\epsilon}{1-\epsilon} + k \cdot \frac{\epsilon}{1-\epsilon}\right) = \frac{2(2k^2+6k+1)}{3k(k+1)} \cdot \frac{\epsilon}{1-\epsilon}.$$

It is easy to show that $\frac{2(2k^2+6k+1)}{3k(k+1)} < 3$ when $k \geq 2$:

$$\begin{aligned} &3 \cdot 3k(k+1) - 2(2k^2+6k+1) = 5k^2 - 3k - 2 \\ &\quad = k \cdot (5k-3) - 2 \geq 2 \cdot 7 - 2 = 12 > 0 \\ &\Rightarrow 3 \cdot 3k(k+1) > 2(2k^2+6k+1) \\ &\Rightarrow 3 > \frac{2(2k^2+6k+1)}{3k(k+1)}. \end{aligned}$$

Therefore, the ratio $V(\hat{\mathbf{x}})/V^*$ is at most $1 + \frac{3\epsilon}{1-\epsilon} = 1 + \epsilon'$ as $\epsilon = \frac{\epsilon'}{3+\epsilon'}$. Hence, $\hat{\mathbf{x}}$ is a $(1+\epsilon')$-approximate solution to RSP. □

The complexity of this $(1+\epsilon')$-APPROXIMATION procedure is dominated by the complexity of LARGE ASSIGNMENT, which is $O(\frac{n}{\epsilon'^{2k}})$ for constant k. (SMALL ASSIGNMENT takes linear time in n.) Note that $\frac{1}{\epsilon} = \frac{3+\epsilon'}{\epsilon'} = O(\frac{1}{\epsilon'})$. Therefore, the complexity of RSP $(1+\epsilon')$-approximation algorithm is $O(\frac{n}{\epsilon'^{2k}})$, which is polynomial in n and $\frac{1}{\epsilon'}$ for constant k. Additionally, a family of $(1+\epsilon')$-approximation algorithms with complexity that is polynomial in n and $\frac{1}{\epsilon'}$ is called a fully polynomial time approximation scheme.

5. A Polynomial Time Approximation Scheme for Single-Cycle RSP with Nonconstant k

In this section, we combine the $(1+\frac{2}{k})$-approximation algorithm by Hall (1998) with the FPTAS that we just presented to get a PTAS for single-cycle RSP with nonconstant k. Because this problem is strongly NP-hard, a PTAS is the best approximation possible.

The $(1+\frac{2}{k})$-approximation algorithm of Hall (1998) has a complexity of $O(n)$. Recall that the continuous single-cycle RSP has closed form solutions (Homer 1966, Page and Paul 1976, Zoller 1977). The algorithm of Hall (1998) rounds down the reorder timings of the optimal solution for the continuous problem. As the value of k increases, the resolution of the discrete problem becomes more and more refined, and the rounded solution becomes very close to the continuous solution and close to the optimum. (This is because the continuous optimal solution is a lower bound on the value of the discrete optimum.) For the sake of completeness, we include the pseudocode of the algorithm below.

Algorithm 4. ($(1+\frac{2}{k})$-APPROXIMATION (Hall 1998))
for $i = 1, \ldots, n$, **do**
$j(i) \leftarrow \left\lfloor k \cdot \frac{\sum_{h=1}^{i} s_h}{\sum_{h=0}^{n} s_h} \right\rfloor$.
Assign item i to time $j(i)$.
end for

Although this $(1+\frac{2}{k})$-approximation algorithm performs better as k increases, our FPTAS works well for small values of k, where the continuous solution is not close to the discrete solution and the approximation algorithm of Hall (1998) does not work well.

We take advantage of the complexity of Hall's algorithm and the approximation factor of our FPTAS into a PTAS, which works as follows. Ehen $k > \frac{2}{\epsilon}$, we run the algorithm of Hall (1998), which is a $(1+\epsilon)$-approximation algorithm. When $k \leq \frac{2}{\epsilon}$, we run the FPTAS in this paper, and the running time is

$O(k^2 \cdot k! \cdot \frac{n}{\epsilon^{2k}}) = O((\frac{2}{\epsilon})! \cdot \frac{n}{\epsilon^{2k+2}})$ when k is nonconstant. The overall running time $O((\frac{2}{\epsilon})! \cdot \frac{n}{\epsilon^{2k+2}})$ is linear in n for fixed ϵ, and therefore, it is a polynomial time approximation scheme.

6. Concluding Remarks

In this paper, we resolve the complexity of discrete RSP, the problem of minimizing the peak storage requirement with given reorder sizes and individual cycle lengths. Although it was known that discrete RSP is NP-hard even when the joint cycle length is constant (Hall 1998), we prove here that the problem is strongly NP-hard for nonconstant joint cycle length and that the problem is weakly NP-hard for constant joint cycle length.

For constant joint cycle-length discrete RSP, we further present a pseudopolynomial time algorithm that solves the problem optimally and the first known FPTAS for the case when individual cycles are identical (single-cycle). For nonconstant joint cycle-length discrete RSP, we devise here the first known PTAS for the single-cycle that runs in linear time for fixed ϵ. The question of whether there exists an FPTAS and a PTAS for the respective cases of the multi-cycle RSP remains open.

References

Anily S (1991) Multi-item replenishment and storage problem (MIRSP): Heuristics and bounds. *Oper. Res.* 39(2):233–243.

Boctor FF (2010) Offsetting inventory replenishment cycles to minimize storage space. *Eur. J. Oper. Res.* 203(2):321–325.

Boctor FF, Bolduc MC (2015) Inventory replenishment planning and staggering. *IFAC PapersOnLine* 48(3):1416–1421.

Croot E, Huang K (2013) A class of random algorithms for inventory cycle offsetting. *Internat. J. Oper. Res.* 18(2):201–217.

Gallego G, Queyranne M, Simchi-Levi D (1996) Single resource multi-item inventory systems. *Oper. Res.* 44(4):580–595.

Gallego G, Shaw D, Simchi-Levi D (1992) The complexity of the staggering problem, and other classical inventory problems. *Oper. Res. Lett.* 12(1):47–52.

Gary MR, Johnson DS (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness* (W. H. Freeman and Co., New York).

Hall NG (1998) A comparison of inventory replenishment heuristics for minimizing maximum storage. *Amer. J. Math. Management Sci.* 18(3–4):245–258.

Hariga MA, Jackson PL (1995) Time-variant lot sizing models for the warehouse scheduling problem. *IIE Trans.* 27(2): 162–170.

Hariga MA, Jackson PL (1996) The warehouse scheduling problem: Formulation and algorithms. *IIE Trans.* 28(2):115–127.

Hartley R, Thomas LC (1982) The deterministic, two-product, inventory system with capacity constraint. *J. Oper. Res. Soc.* 33(11): 1013–1020.

Homer E (1966) Space-limited aggregate inventories with phased deliveries. *J. Indust. Engrg.* 17(6):327–333.

Moon IK, Cha BC, Kim SK (2008) Offsetting inventory cycles using mixed integer programming and genetic algorithm. *Internat. J. Indust. Engrg. Theory Appl. Practice* 15(3):245–256.

Murthy NN, Benton WC, Rubin PA (2003) Offsetting inventory cycles of items sharing storage. *Eur. J. Oper. Res.* 150(2): 304–319.

Page E, Paul RJ (1976) Multi-production inventory situation with one restriction. *J. Oper. Res. Soc.* 27(4):815–834.

Russell RA, Urban TL (2016) Offsetting inventory replenishment cycles. *Eur. J. Oper. Res.* 254(1):105–112.

Teo CP, Ou J, Tan KC (1998) Multi-item inventory staggering problems: Heuristics and bounds. *Proc. Ninth Annual ACM-SIAM Sympos. Discrete Algorithms* (Society for Industrial and Applied Mathematics, Philadelphia), 584–593.

Thomas LC, Hartley R (1983) An algorithm for limited capacity inventory problem with staggering. *J. Oper. Res. Soc.* 34(1): 81–85.

Yao MJ, Chu WM (2008) A genetic algorithm for determining optimal replenishment cycles to minimize maximum warehouse space requirements. *Omega* 36(4):619–631.

Yao MJ, Chu WM, Lin YF (2008) Determination of replenishment dates for restricted-storage, static demand, cyclic replenishment schedule. *Comput. Oper. Res.* 35(10):3230–3242.

Zoller K (1977) Deterministic multi-item inventory systems with limited capacity. *Management Sci.* 24(4):451–455.

Dorit S. Hochbaum is a full professor and chancellor chair at UC Berkeley's department of IEOR. Her recent work focuses on efficient techniques related to network flows with applications in ranking, pattern recognition, data mining and image segmentation. Hochbaum is the author of over 160 papers that have appeared in the operations research, management science and theoretical computer science literature. Hochbaum is an INFORMS fellow, SIAM fellow, and honorary doctorate of University of Copenhagen.

Xu Rao is a doctoral candidate in the Department of Industrial Engineering and Operations Research at the University of California, Berkeley. Her research interests include combinatorial optimization and approximation algorithms.