Operations Research Letters 48 (2020) 835–839

Contents lists available at ScienceDirect

Operations Research Letters

journal homepage: www.elsevier.com/locate/orl

A fully polynomial time approximation scheme for the Replenishment Storage problem

Dorit S. Hochbaum, Xu Rao *

University of California, Department of IEOR, Etcheverry Hall, Berkeley, CA, United States of America

ARTICLE INFO

Article history:
Received 29 September 2020
Received in revised form 13 October 2020
Accepted 14 October 2020
Available online 28 October 2020

Keywords:
Approximation algorithm
Fully polynomial time approximation scheme
Weakly NP-hard

ABSTRACT

The Replenishment Storage problem (RSP) is to minimize the storage capacity requirement for a deterministic demand, multi-item inventory system with specified individual reorder cycle lengths. The reorders can only take place at integer time units. This problem was shown to be weakly NP-hard for constant joint cycle length (the least common multiple of all individual cycle lengths). We devise here the first known FPTAS for the RSP with different individual cycle lengths and constant joint cycle length.

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. The input to the RSP consists of in 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. Here the 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.

An instance of 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 let $k = \text{lcm}(k_1, \ldots, k_n)$. 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 length, k, the problem is said to be *single-cycle*, otherwise it is said to be *multi-cycle*.

The RSP is an NP-hard problem [3,4], so there is no polynomial time algorithm unless P = NP. But a polynomial time approximation scheme may exist for the problem. An approximation scheme is a family of $(1+\epsilon)$-approximation algorithms for every $\epsilon > 0$. If the running time is polynomial in the problem size for every fixed ϵ, then this scheme is a Polynomial Time Approximation Scheme (PTAS); furthermore, if the running time is polynomial in both the problem size and $1/\epsilon$, then it is a Fully Polynomial Time Approximation Scheme (FPTAS). Hochbaum and Rao [4] gave a Fully Polynomial Time Approximation Scheme (FPTAS) for the single-cycle RSP when k is a constant [4]. For the multi-cycle case however no FPTAS has been known to date. Here, we establish for the first time an FPTAS for the multi-cycle RSP when the joint cycle length, k, is constant. We also observe here that the FPTAS of Hochbaum and Rao for the single cycle RSP is fixed-parameter tractable (FPT) and is in fact linear for a constant length of the single cycle.

1.1. Related literature

The single-cycle RSP was shown by Hall [3] to be NP-hard, even when the joint cycle length $k = 2$. Since the single-cycle RSP is a special case of the multi-cycle RSP, it implies that the multi-cycle RSP is also NP-hard, even when k is small. Hochbaum and Rao [4] investigated the complexity status of the single-cycle and the multi-cycle RSPs and showed that the problems are strongly NP-hard when k is not a constant, but weakly NP-hard when k is a constant. They further provided in [4] a pseudo-polynomial algorithm for the two problems.

These complexity results imply that there is no polynomial time algorithm for single-cycle and the multi-cycle RSPs even when k is a constant, unless P=NP. Several approximation results have been delivered for the single-cycle RSP. Hall [3] provided a linear time approximation algorithm for the single-cycle RSP, with an approximation factor of $\left(1+\frac{2}{k}\right)$, even for non-constant k. Hochbaum and Rao [4] devised for the single-cycle RSP with constant k an FPTAS, and for the single-cycle RSP with non-constant k

* Corresponding author.
E-mail addresses: hochbaum@ieor.berkeley.edu (D.S. Hochbaum), xrao@berkeley.edu (X. Rao).

https://doi.org/10.1016/j.orl.2020.10.004
0167-6377/

a PTAS. The complexity of the FPTAS for a $(1+\epsilon)$-approximation algorithm is $O(n\epsilon^{-2k})$, and the complexity of the PTAS for non-constant cycle length is $O((2\epsilon^{-1})! \cdot n\epsilon^{-2k-2})$. (We note that there was a mistake in the proof of Theorem 6 in [4], which has been corrected by replacing the original scaling factor $\epsilon^2 D$ by $\epsilon^2 kD$ in the FPTAS [5]. This affected the running time by only a constant factor, k^k, so the approximation scheme is still an FPTAS. What is more, we observe here that the running time of this FPTAS is fixed-parameter tractable for parameter k.)

For the multi-cycle RSP with only two items, Murthy et al. [7] provided an optimal closed-form replenishment solution, meaning that it is solved in constant time. Studies of algorithmic results for the multi-cycle RSP with more than two items have been focused on the development of heuristics. These include genetic algorithms [6,9]), a smoothing procedure utilizing a Boltzmann function [10], local-search procedures [2], a simulated-annealing algorithm [1] and a hybrid heuristic [1,8]. No algorithm with guaranteed approximation bound has been known for the multi-cycle RSP.

1.2. Contributions

A weakly NP-hard problem can have an FPTAS and it was shown in [4] that for constant k the RSP problem is weakly NP-hard. For constant parameter k, Hochbaum and Rao [4] devised for the single-cycle RSP an FPTAS, which we observe here is fixed-parameter tractable (FPT). We devise here an FPTAS for the multi-cycle RSP with constant joint cycle length for the first time. Unlike the case of the single-cycle (in [4]), the running time of this FPTAS for the multi-cycle RSP is not fixed-parameter tractable for parameter k.

A summary of the complexity results for RSP that includes our contributions here is given in Table 1.

1.3. Paper overview

The next section, Section 2, introduces the notation, an integer programming formulation as well as a pseudo-polynomial algorithm for the RSP which is relevant to the approximation scheme. In Section 3 we describe the new fully polynomial-time approximation scheme (FPTAS) for the multi-cycle RSP for constant joint cycle length k.

2. Preliminaries

Our approximation scheme utilizes a dynamic programming algorithm for the RSP derived by Hochbaum and Rao [4]. That dynamic programming algorithm uses an integer programming (IP) formulation of the RSP that was introduced in [4]. Since this algorithm and IP formulation are crucial for our FPTAS, we sketch them here.

We first present necessary notation. For an instance of 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 $(s_i, \frac{k_i-1}{k_i}s_i, \frac{k_i-2}{k_i}s_i, \ldots, \frac{1}{k_i}s_i)$. Recall that since $k = \text{lcm}(k_1, \ldots, k_n)$, 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. This is because each item must be reordered at least once in such interval, and the 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 time frame 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 2.1. 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,

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

The following lists the notation for 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.

$V_\ell(\mathbf{x})$: the inventory level at time ℓ according to assignment $\mathbf{x}$ 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.

2.1. The integer programming formulation of the RSP

The IP formulation of Hochbaum and Rao [4] is based on three lemmas derived in their paper, which are included for the sake of completion. Lemma 2.2 shows that valid assignments can be restricted to those attaining peak inventory level at time k without changing the optimal solution of the RSP. Lemma 2.3 establishes the relation between the inventory levels $V_\ell(\mathbf{x})$ for $\ell = 1, \ldots, k$ and the total amount ordered at time j, $Q_j(\mathbf{x})$ for $j = 1, \ldots, k$.

Lemma 2.2 ([4]). *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, $V_k(\mathbf{x}') = V(\mathbf{x}') = V(\mathbf{x})$.*

Lemma 2.3 ([4]). *For any valid assignment $\mathbf{x}$,*

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

Let the following quantity, which is a constant, be denoted by C:

$$C = \sum_{i=1}^n \tfrac{1}{2}(1 + \frac{1}{k_i})ks_i + \frac{(1+k)k}{2}D.$$

Let $z(\mathbf{x})$ be the following function of a valid assignment $\mathbf{x}$:

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

The next lemma shows that minimizing the inventory level of time k, $V_k(\mathbf{x})$, is equivalent to maximizing $z(\mathbf{x})$.

Lemma 2.4 ([4]). *For any valid assignment $\mathbf{x}$, $kV_k(\mathbf{x}) + z(\mathbf{x}) = C$.*

Table 1
Summary of complexity and algorithmic results for the RSP.

Problem	Non-constant joint cycle	Constant joint cycle
Single-cycle	Strongly NP-hard $(1+2/k)$-approximation [3], PTAS [4]	Weakly NP-hard $(1+2/k)$-approximation [3], pseudo-poly algorithm [4] & FPTAS (here FPT in k) [4]
Multi-cycle	Strongly NP-hard –	Weakly NP-hard pseudo-poly algorithm [4] & FPTAS (here)

By the above lemmas, the RSP can 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 $\ell = 1, \ldots, k$. These inequalities, according to Lemma 2.3, are equivalent to,

$$\sum_{j=1}^{\ell} Q_j(\mathbf{x}) \le \ell D \text{ for } \ell = 1, \ldots, k. \tag{2.1}$$

This set of inequalities (2.1) is referred to as the *cascading constraints*. From Lemma 2.4, the integer programming formulation below has the same optimal solution as the RSP.

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

2.2. The dynamic programming algorithm for the RSP

We present here the dynamic programming algorithm of Hochbaum and Rao [4], which is associated with the integer program (IP). 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,

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

where $Q_j(\mathbf{x}^h) = \sum_{i=1}^{h} s_i x_{ij}$. The function $f_h(q_1, q_2, \ldots, q_k)$ is set to $-\infty$ 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 $f_{h-1}(\cdot)$ requires to determine the timing to replenish item h within the first k_h time units so as to maximize the objective $\sum_{j=1}^{k}(k-j+1)Q_j(\mathbf{x}^h)$. The recursive equation, using the notation $q'_\ell(\tau) = q_\ell - \lfloor \frac{\ell-\tau+k_h}{k_h} \rfloor s_h$, is:

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

All function values are evaluated recursively for $h = 1, \ldots, n$ and for 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.

The running time of this algorithm is $O(nD^k)$ for constant k [4], which is pseudo-polynomial as it depends on the value D.

3. A fully polynomial-time approximation scheme for the RSP with constant joint cycle length

As the RSP is strongly NP-hard when the joint cycle length is not a constant, there is no fully polynomial-time approximation scheme assuming that P≠NP. However, when the joint cycle length k is constant, it is possible to obtain a fully polynomial-time approximation scheme for this problem. Hochbaum and Rao [4] showed an FPTAS for the single-cycle RSP but no FPTAS has been known for the multi-cycle case when k is constant. In this section, we establish the first known FPTAS for the multi-cycle RSP for constant joint cycle length.

Here we derive a family of $(1+\epsilon')$-approximation algorithms for the multi-cycle RSP for every $\epsilon' > 0$. The $(1+\epsilon')$-approximation algorithm works by applying the dynamic programming algorithm in Section 2.2 with scaled reorder sizes with some scaling factor T. We show in this section that the output of the dynamic programming algorithm using the scaled sizes is within a factor of $1+\epsilon'$ of the optimal solution. The run time of this approximation algorithm is polynomial in n and ϵ'^{-1}, and hence this family of algorithms is a fully polynomial approximation scheme.

The approximation algorithm solves a modified integer program of (IP), (scaled IP), in which the order sizes are scaled by a factor T. The scaled problem is solvable using the dynamic programming procedure of Section 2.2 and the solution of it is a valid assignment that has objective function value close to the optimal value of (IP).

3.1. The scaling of (IP), (scaled IP)

For any $\epsilon' > 0$, we let $\epsilon = \epsilon'/2$ and we scale the reorder sizes by the factor $T = \frac{\epsilon D}{kn}$ as follows. Let $s'_i = \lfloor \frac{s_i}{T} \rfloor$ be the scaled sizes of items $i = 1, \ldots, n$ and $D' = \frac{D}{T}$ be the scaled demand. Let $Q'_j(\mathbf{x})$ and $z'(\mathbf{x})$ denote the "scaled" replenishment sizes at time j and the objective function for the scaled sizes s'_i: $Q'_j(\mathbf{x}) = \sum_{i=1}^{n} s'_i x_{ij}, \quad j = 1, \ldots, k$; $z'(\mathbf{x}) = \sum_{j=1}^{k}(k-j+1)Q'_j(\mathbf{x})$.

We refer to the resulting integer program by substituting $z(\mathbf{x})$ with $z'(\mathbf{x})$, D with D', and $Q_j(\mathbf{x})$ with $Q'_j(\mathbf{x})$ for each j. The optimal solution for (scaled IP) is found by applying the dynamic programming procedure in Section 2.2 with scaled sizes D' and $s'_1, \ldots, s'_n$. The running time of finding the optimal solution for (scaled IP) with the dynamic programming procedure, is $O(nD'^k) = O(n^{k+1}\epsilon^{-k})$.

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

3.2. The ϵ-relaxed RSP

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

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

We refer to the constraints $\sum_{j=1}^{\ell} Q_j(\mathbf{x}) \le \ell D + \epsilon D$ 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 ϵD.

Lemma 3.1. *The peak inventory level of any feasible solution* $\mathbf{x}$ *to (ϵ-relaxed IP) is at most* $V_k(\mathbf{x}) + \epsilon D$.

Proof. Any feasible solution $\mathbf{x}$ for (ϵ-relaxed IP) is a valid assignment, so Lemma 2.3 applies. That is, $V_\ell(\mathbf{x}) = V_k(\mathbf{x}) + \left(\sum_{j=1}^{\ell} Q_j(\mathbf{x}) - \ell D\right)$ for $\ell = 1, \ldots, k$. The ϵ-relaxed cascading constraints state that $\sum_{j=1}^{\ell} Q_j(\mathbf{x}) - \ell D \le \epsilon D$ for all ℓ. So when $\mathbf{x}$ is a feasible solution of (ϵ-relaxed IP), $V_\ell(\mathbf{x}) \le V_k(\mathbf{x}) + \epsilon D$ for all ℓ, and hence, $V(\mathbf{x}) = \max_\ell V_\ell(\mathbf{x}) \le V_k(\mathbf{x}) + \epsilon D$.

The next lemma proves that any feasible solution for (scaled IP), including $\hat{\mathbf{x}}$, is feasible for (ϵ-relaxed IP).

Lemma 3.2. *Any assignment* $\mathbf{x}$ *that is feasible for (scaled IP) is feasible for (ϵ-relaxed IP).*

Proof. In both problems $\mathbf{x}$ is required to be a valid assignment. It remains to show that $\mathbf{x}$ satisfies the ϵ-relaxed cascading constraints, that is, $\sum_{j=1}^{\ell} Q_j(\mathbf{x}) \le \ell D + \epsilon D$ for $\ell = 1, \ldots, k$.

By definition, $s'_i = \lfloor \frac{s_i}{T} \rfloor$. So $s_i < T(s'_i + 1)$ and thus,

$$
\begin{aligned}
\sum_{j=1}^{\ell} Q_j(\mathbf{x}) &= \sum_{j=1}^{\ell}\sum_{i=1}^{n} s_i \mathbf{x}_{ij} \le \sum_{j=1}^{\ell}\sum_{i=1}^{n} T(s'_i + 1)\mathbf{x}_{ij} \\
&= T\left(\sum_{j=1}^{\ell}\sum_{i=1}^{n} s'_i \mathbf{x}_{ij} + \sum_{j=1}^{\ell}\sum_{i=1}^{n} \mathbf{x}_{ij}\right). \qquad (3.1)
\end{aligned}
$$

Since $\mathbf{x}$ is feasible for (scaled IP), the scaled cascading constraints are satisfied. And as $D' = \frac{D}{T}$,

$$
\sum_{j=1}^{\ell}\sum_{i=1}^{n} s'_i \mathbf{x}_{ij} = \sum_{j=1}^{\ell} Q'_j(\mathbf{x}) \le \ell D' = \frac{\ell D}{T}. \qquad (3.2)
$$

For $\ell = 1, \ldots, k$,

$$
\sum_{j=1}^{\ell}\sum_{i=1}^{n} \mathbf{x}_{ij} \le \sum_{j=1}^{k}\sum_{i=1}^{n} \mathbf{x}_{ij} \le nk. \qquad (3.3)
$$

Hence from inequalities (3.1), (3.2) and (3.3),

$$
\sum_{j=1}^{\ell} Q_j(\mathbf{x}) \le T\left(\frac{\ell D}{T} + nk\right) = \ell D + \epsilon D. \quad \square
$$

Using the relationship between reorder sizes s_i and the scaled sizes s'_i, we show that for any feasible solution of (scaled IP), $\mathbf{x}$, the objective with original sizes $z(\mathbf{x}) = \sum_{j=1}^{k}(k-j+1)Q_j(\mathbf{x})$ is closely approximated by the objective with scaled sizes $z'(\mathbf{x}) = \sum_{j=1}^{k}(k-j+1)Q'_j(\mathbf{x})$, corrected for the scaling factor T:

3.3. The approximation property of the solution to (scaled IP)

Lemma 3.3. *For any assignment of items* $\mathbf{x}$ *feasible for (scaled IP), the values of the objective function with original and scaled sizes,* $z(\mathbf{x})$ *and* $z'(\mathbf{x})$ *respectively, satisfy,*

$$
Tz'(\mathbf{x}) \le z(\mathbf{x}) \le Tz'(\mathbf{x}) + \epsilon kD.
$$

Proof. Recall that $s'_i = \lfloor \frac{s_i}{T} \rfloor$, so $Ts'_i \le s_i < T(s'_i + 1)$. We derive the lower bound on $z(\mathbf{x})$ as follows:

$$
\begin{aligned}
z(\mathbf{x}) &= \sum_{j=1}^{k}(k-j+1)Q_j(\mathbf{x}) \\
&= \sum_{j=1}^{k}(k-j+1)\sum_{i=1}^{n} s_i x_{ij} \\
&\ge T \cdot \sum_{j=1}^{k}(k-j+1)\sum_{i=1}^{n} s'_i x_{ij} \\
&= T \cdot \sum_{j=1}^{k}(k-j+1)Q'_j(\mathbf{x}) \\
&= Tz'(\mathbf{x}).
\end{aligned}
$$

The upper bound on $z(\mathbf{x})$ can be derived as follows:

$$
\begin{aligned}
z(\mathbf{x}) &= \sum_{j=1}^{k}(k-j+1)Q_j(\mathbf{x}) \\
&= \sum_{j=1}^{k}(k-j+1)\sum_{i=1}^{n} s_i x_{ij} \\
&\le T \cdot \sum_{j=1}^{k}(k-j+1)\sum_{i=1}^{n}(s'_i + 1)x_{ij} \\
&= T \cdot \left[\sum_{j=1}^{k}(k-j+1)Q'_j(\mathbf{x}) + \sum_{j=1}^{k}(k-j+1)\sum_{i=1}^{n} x_{ij}\right] \\
&\le Tz'(\mathbf{x}) + Tk^2 n \\
&= Tz'(\mathbf{x}) + \epsilon kD. \quad \square
\end{aligned}
$$

Lemma 3.3 leads to the following lower bound on $z(\hat{\mathbf{x}})$ for $\hat{\mathbf{x}}$ being an optimal solution of (scaled IP):

Theorem 3.4. *For any feasible solution* $\mathbf{x}$ *of (IP),* $z(\hat{\mathbf{x}}) \ge z(\mathbf{x}) - \epsilon kD$.

Proof. By Lemma 3.3, we know $z(\hat{\mathbf{x}}) \ge Tz'(\hat{\mathbf{x}})$. Since any feasible solution of (IP), $\mathbf{x}$, must be also feasible for (scaled IP), we use the

upper bound of $z(\mathbf{x})$ from Lemma 3.3 to get:

$Tz'(\mathbf{x}) \geq z(\mathbf{x}) - \epsilon kD$.

Because $\hat{\mathbf{x}}$ is optimal for (scaled IP), it follows that $z'(\hat{\mathbf{x}}) \geq z'(\mathbf{x})$. Combining the three inequalities, we get

$z(\hat{\mathbf{x}}) \geq Tz'(\hat{\mathbf{x}}) \geq Tz'(\mathbf{x}) \geq z(\mathbf{x}) - \epsilon kD$. □

Consequently, the optimal solution $\hat{\mathbf{x}}$ for (scaled IP) attains a objective value $z(\hat{\mathbf{x}})$ that is at least as much as the optimal objective of (IP) minus ϵkD.

3.4. The $(1+\epsilon')$-approximation bound

From the discussion above, we know that the optimal solution for (scaled IP) $\hat{\mathbf{x}}$ is a valid assignment whose inventory levels at time k approximates that maximum inventory level, and the value $z(\mathbf{x})$ approximates the optimal objective value of (IP). We will prove here that $\hat{\mathbf{x}}$ is an $(1+\epsilon')$-approximation solution for $\epsilon' = 2\epsilon$ and any $\epsilon > 0$.

Theorem 3.5. *The optimal solution $\hat{\mathbf{x}}$ for (scaled IP) is a $(1+\epsilon')$-approximation solution for the RSP.*

Proof. Assignment $\hat{\mathbf{x}}$ is valid as it is feasible for (scaled IP). So we just need to prove the approximation factor for the peak inventory level.

Let $\mathbf{x}^*$ be an optimal solution of (IP), and V^* the corresponding peak inventory level.

As stated in Theorem 3.4, $z(\hat{\mathbf{x}}) \geq z(\mathbf{x}) - \epsilon kD$ for any $\mathbf{x}$ that is feasible of (IP), including $\mathbf{x}^*$. From Lemma 2.4, 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,

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

From Lemma 3.1 it follows that the peak inventory level for $\hat{\mathbf{x}}$ satisfies $V(\hat{\mathbf{x}}) \leq V_k(\hat{\mathbf{x}}) + \epsilon D$. Since $\mathbf{x}^*$ is a solution of (IP), the peak inventory level for $\mathbf{x}^*$ is $V^* = V_k(\mathbf{x}^*)$. Hence,

$V(\hat{\mathbf{x}}) \leq V_k(\hat{\mathbf{x}}) + \epsilon D \leq V^* + 2\epsilon D$.

That is, for the optimum peak storage of the RSP, V^*, and for the optimal solution of (scaled IP) $\hat{\mathbf{x}}$, the ratio $V(\hat{\mathbf{x}})/V^*$ is at most $1 + 2\epsilon D/V^*$. Observe that V^* must be at least the per unit time demand D, it follows that $2\epsilon D/V^* \leq 2\epsilon$.

Therefore, the ratio $V(\hat{\mathbf{x}})/V^*$ is at most $1+2\epsilon = 1+\epsilon'$. Hence, $\hat{\mathbf{x}}$ is a $(1+\epsilon')$-approximate solution to the RSP. □

The complexity of this approximation procedure is $O(n^{(k+1)}\epsilon^{-k})$ for constant k. As $\epsilon' = 2\epsilon$, this complexity is polynomial in n and ϵ'^{-1}. Therefore, we indeed have an FPTAS.

4. Concluding remarks

Both the single-cycle and the multi-cycle RSPs are weakly NP-hard but an FPTAS was known only for the single cycle RSP, in [4]. Here we devise an FPTAS for the *multi-cycle* RSP with constant joint cycle length. The running time of our FPTAS here is not fixed-parameter tractable as compared to the running time of the FPTAS for the single-cycle case. We leave the existence of a fixed-parameter tractable FPTAS for the multi-cycle RSP as an open question. The question of whether there exists a PTAS for the multi-cycle RSP when the joint cycle length is not constant remains open as well.

Acknowledgment

The research of the authors was supported in part by NSF, United States of America, award No. CMMI-1760102. We thank the anonymous referees and the area editor for a number of helpful comments and suggestions.

References

[1] F.F. Boctor, Offsetting inventory replenishment cycles to minimize storage space, European J. Oper. Res. 203 (2) (2010) 321–325.
[2] E. Croot, K. Huang, A class of random algorithms for inventory cycle offsetting, Int. J. Oper. Res. 18 (2) (2013) 201–217.
[3] N.G. Hall, A comparison of inventory replenishment heuristics for minimizing maximum storage, Amer. J. Math. Management Sci. 18 (3–4) (1998) 245–258.
[4] D.S. Hochbaum, X. Rao, The replenishment schedule to minimize peak storage problem: The gap between the continuous and discrete versions of the problem, Oper. Res. 67 (5) (2019) 1345–1361.
[5] D.S. Hochbaum, X. Rao, Errata and simplification for Hochbaum and Rao OR 2019, 2020, https://hochbaum.ieor.berkeley.edu/html/pub/RSP-errata.pdf. (Accessed 25 September 2020).
[6] I.K. Moon, B.C. Cha, S.K. Kim, Offsetting inventory cycles using mixed integer programming and genetic algorithm, Int. J. Ind. Eng. Theory Appl. Pract. 15 (3) (2008) 245–256.
[7] N.N. Murthy, W. Benton, P.A. Rubin, Offsetting inventory cycles of items sharing storage, European J. Oper. Res. 150 (2) (2003) 304–319.
[8] R.A. Russell, T.L. Urban, Offsetting inventory replenishment cycles, European J. Oper. Res. 254 (1) (2016) 105–112.
[9] M. Yao, W. Chu, A genetic algorithm for determining optimal replenishment cycles to minimize maximum warehouse space requirements, Omega 36 (4) (2008) 619–631.
[10] M. Yao, W. Chu, Y. Lin, Determination of replenishment dates for restricted-storage, static demand, cyclic replenishment schedule, Comput. Oper. Res. 35 (10) (2008) 3230–3242.