A Fully Polynomial Time Approximation Scheme for Single-Item Stochastic Lot-Sizing Problems with Discrete Demand

N. Halman, D. Klab, Jan M. Mostagir, J. Orlin, D. Simchi-Levi
A Fully Polynomial Time Approximation Scheme for
Single-Item Stochastic Lot-Sizing Problems with Discrete
Demand

N. Halman  D. Klabjan  M. Mostagir  J. Orlin  D. Simchi-Levi

January 11, 2006

Abstract

The single-item stochastic lot-sizing problem is to find an inventory replenishment policy in the presence of a stochastic demand under periodic review and finite time horizon. The computational intractability of computing an optimal policy is widely believed and therefore approximation algorithms should be considered. To the best of our knowledge, this is the first work that develops a fully polynomial time approximation scheme for this problem. In other words, we design a tractable polynomial time algorithm that finds a policy that is arbitrarily close in the relative sense to the value of an optimal policy. In addition, we formally prove that finding an optimal policy is intractable in the standard sense.

1 Introduction

In this work we consider the standard single-item stochastic lot-sizing problem. We assume periodic time replenishment over a finite number of time periods. We assume also that the holding cost, which includes a potential penalty for backlogs, is convex. In addition, the procurement cost is convex and nondecreasing. In the case of the linear procurement cost, it is well known that the base stock policy is optimal. Since we consider the non stationary case, the base stock levels are time dependent. We point out that the policy structure in the general convex nondecreasing procurement cost case is not known.

It is widely acknowledged that even in the linear procurement cost case the computation of the base stock levels is very hard. It is for this reason that we resort to approximation algorithms. Our main result is a fully polynomial time approximation scheme (FPTAS) for the aforementioned stochastic lot-sizing problem. In order the problem to be well defined (in the standard context of Turing machines or computational complexity) we assume that the demand random variables are discrete, completely known in advance, and independent, but not necessarily identically distributed.
The standard dynamic programming approach by means of the optimality equation gives only a pseudo polynomial algorithm. This algorithm is linear with respect to the maximum demand value, which is exponential in the input size. The main problem with dynamic programming is in the fact that all possible inventory levels need to be considered, which are “too” many. To circumvent this, in each time period we carefully select only a subset of possible inventory levels and compute an approximation of the optimal value function on this subset. These sets are computed recursively for every time period. At all other inventory levels the approximate value function is interpolated.

2 Problem statement

Let $T$ be the length of the planning horizon. At the beginning of a time period we first observe the inventory level and next a replenishment decision is made. If we place an order, it arrives immediately, i.e., we assume there is no lead time. Since we allow backlogging, the inventory level can be negative. In this case its absolute value corresponds to the number of units on backlog. If the inventory level is positive, then a holding cost is charged, otherwise a backlogging cost is occurred. For ease of discussion we call both of these components the holding cost. The holding cost is accounted for at the end of the time period. For each time period $t = 1, ..., T$ we define:

\[ x_t : \text{procurement quantity in time period } t; \]
\[ I_t : \text{inventory level at the beginning of time period } t; \]
\[ \bar{I}_t : \text{inventory level at the end of period } t \text{ (i.e., } I_t = \bar{I}_{t-1}); \]
\[ c_t(x_t) : \text{procurement cost in time period } t, \text{ given an order of size } x_t; \]
\[ h_t(I_t) : \text{holding cost in time period } t, \text{ given inventory level } I_t. \]

We define $I_1$ to be 0, or in other words, we start with no inventory.

The input data for the problem consists of the number of time periods $T$, and for each time period $t = 1, ..., T$ an oracle that computes functions $c_t, h_t$, and a discrete random variable $D_t$ describing the demand in time period $t$. For each $D_t$ we are given $n_t$, the number of different values it admits with positive probability, and these demand values $d_{t,1} < ... < d_{t,n_t}$. Moreover, we are also given positive integers $q_{t,1}, ..., q_{t,n_t}$ such that

\[ \text{Prob}[D_t = d_{t,i}] = \frac{q_{t,i}}{\sum_{j=1}^{n_t} q_{t,j}}. \]
We define for every $t = 1, \ldots, T$ and $i = 1, \ldots, n_t$ the following values:

\[
\begin{align*}
  p_{t,i} &= \text{Prob}[D_t = d_{t,i}] \quad \text{probability that there is a demand of } d_{t,i} \text{ units in time period } t; \\
  n^* &= \max_t n_t \quad \text{maximum number of different values } D_t \text{ can take over the entire time horizon}; \\
  d^* &= \max_t d_{t,n_t} \quad \text{maximum demand over the entire time horizon}; \\
  D^* &= \sum_{t=1}^T d_{t,n_t} \quad \text{maximum total demand over the entire time horizon}; \\
  Q_t &= \sum_{j=1}^{n_t} q_{t,j}; \\
  M_t &= \prod_{j=1}^T Q_j; \\
  M_{T+1} &= 1.
\end{align*}
\]

We make the following assumptions.

**Assumption 2.1** All demand, procurement and inventory levels are integral. Moreover, the demand and procurement levels are nonnegative.

**Assumption 2.2** The procurement cost function $c_t$ is nondecreasing convex in $\mathbb{Z}^+$, and $c_t(0) = 0$ for every $t = 1, \ldots, T$.

**Assumption 2.3** The holding cost function $h_t$ is convex and nonnegative for every $t = 1, \ldots, T$.

**Assumption 2.4** All cost functions can be evaluated in polynomial time at any value in their domain, and are scaled such that they are integer valued.

Note that the binary input size of the problem is bounded below by $\Omega(T + n^* + \log d^*)$. Assumptions 2.1, 2.2, and 2.3 together with integrality of demand imply the integrality of procurement and inventory levels.

The objective is to minimize the total expected cost. The problem can be formulated as finding a policy $x_t = x_t(I_t)$ for $t = 1, \ldots, T$ that realizes

\[
 z^* = \min_{x_t} E_D(\sum_{t=1}^T c_t(x_t) + h_t(I_t + x_t - D_t)),
\]

subject to the system dynamics

\[
 I_{t+1} = I_t + x_t - D_t, \quad t = 1, \ldots, T.
\]

The action space requirement is $x_t \in \mathbb{Z}^+$ for $t = 1, \ldots, T$, and the initial state is $I_1 = 0$.

Note that in the above context we have $E_D(\sum_{t=1}^T c_t(x_t) + h_t(I_t + x_t - D_t)) = \sum_{t=1}^T c_t(x_t) + \sum_{j=1}^{n_t} p_{t,j} h_t(I_t + x_t - d_{t,j})$. The standard optimality equation is presented later in Section 4.1.
3 K-approximating sets and functions

An $K$-approximation algorithm for a minimization problem guarantees its output to be no more than $K$ times the optimal solution. In this section we define $K$-approximating functions and $K$-approximation sets.

Definition 3.1 Let $K > 1$ and let $f : D \rightarrow \mathbb{R}$ be a function. We say that $\hat{f} : D \rightarrow \mathbb{R}$ is a $K$-approximation of $f$ if for all $x \in D$ we have $f(x) \leq \hat{f}(x) \leq Kf(x)$.

The following proposition follows directly from the definition of $K$-approximation.

Proposition 3.2 Let $K > 1$, let $f_1, f_2 : D \rightarrow \mathbb{R}$ be functions over domain $D$, let $\hat{f}_1, \hat{f}_2 : D \rightarrow \mathbb{R}$ be $K$-approximations of $f_1, f_2$, respectively, let $g : D \rightarrow D$, and let $\alpha, \beta \in \mathbb{R}^+$. The following properties hold:

1. $\alpha + \beta \hat{f}_1$ is a $K$-approximation of $\alpha + \beta f_1$,
2. $\hat{f}_1 + \hat{f}_2$ is a $K$-approximation of $f_1 + f_2$,
3. $\hat{f}_1(g)$ is a $K$-approximation of $f_1(g)$,
4. $\hat{f}_3(y) := \min_{x \in D} \hat{f}_1(x + y)$ is a $K$-approximation of $f_3(y) := \min_{x \in D} f_1(x + y)$.

In order to get a polynomial time approximation scheme, we consider only a subset of all possible optimal inventory values, whose cardinality is polynomially bounded in the input size. Of course this can only be done by sacrificing accuracy in the final solution. For any pair of integers $A < B$, we let $[A, ..., B]$ denote the set of integers $\{A, A + 1, ..., B\}$. Let $U = 2^x$ for some $x \in \mathbb{N}$ be given. We use the following definition.

Definition 3.3 Let $K > 1$ and let $f : [0, ..., U] \rightarrow \mathbb{Z}^+$ be a nondecreasing function. A weak $K$-approximation set of $f$ is an ordered set $S = \{i_1 < ... < i_r\}$ of integers satisfying the following 3 properties:

1. $S \subseteq \{0, ..., U\}$;
2. for each $k = 1$ to $r - 1$, if $i_{k+1} > i_k + 1$, then $f(i_{k+1}) \leq Kf(i_k)$;
3. for every nonnegative integer $x \leq U$ there is an element $i_k \in S$ such that $i_k \geq x$ and $f(x) \leq f(i_k) \leq Kf(x)$.

If in addition $0 \in S$ and $S$ satisfies that for every $k = 1, ..., r - 1$ there exists a nonnegative integer $j$ such that $i_{k+1} - i_k = 2^j$, we call $S$ a $K$-approximation set of $f$. 

4
Example 3.4 Let \( U = 15 \), \( K = 1 \frac{1}{2} \), and \( f \) be a function defined for \( i = 0,1,...,15 \) as \( f(i) = \left\lfloor \frac{i}{2} \right\rfloor \). It is easy to check that \( \{i_1 = 1, i_2 = 2, i_3 = 3, i_4 = 4, i_5 = 7, i_6 = 9, i_7 = 13, i_8 = 15\} \) is a minimal (by set inclusion) weak \( 1 \frac{1}{2} \)-approximating set of \( f \), and that \( \{i_1 = 0, i_2 = 1, i_3 = 2, i_4 = 3, i_5 = 4, i_6 = 6, i_7 = 8, i_8 = 12, i_9 = 14, i_{10} = 15\} \) and \( \{i_1 = 0, i_2 = 1, i_3 = 2, i_4 = 3, i_5 = 4, i_6 = 6, i_7 = 7, i_8 = 9, i_9 = 13, i_{10} = 15\} \) are both minimal (by set inclusion) \( 1 \frac{1}{2} \)-approximating sets for \( f \), see Table 1.

| Objects / i | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 | 11 | 12 | 13 | 14 | 15 |
|-------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| \( f(i) \)  | 0  | 0  | 1  | 1  | 2  | 2  | 3  | 3  | 4  | 4  | 5  | 5  | 6  | 6  | 7  | 7  |
| \( S_1 \)   | *  | *  | *  | *  | *  | *  | *  | *  |   |   |   |   |   |   |   |   |
| \( S_2 \)   | *  | *  | *  | *  | *  | *  | *  | *  |   |   |   |   |   |   |   |   |
| \( S_3 \)   | *  | *  | *  | *  | *  | *  | *  | *  |   |   |   |   |   |   |   |   |

Table 1: \( S_1 \) is a weak \( 1 \frac{1}{2} \)-approximating set of \( f \), while \( S_2, S_3 \) are \( 1 \frac{1}{2} \)-approximating sets.

Note that weak \( K \)-approximating sets obey monotonicity in the sense that if \( S \) is a weak \( K \)-approximating set of \( f \) and \( S \subseteq S' \subseteq \{0,...,U\} \), then \( S' \) is a weak \( K \)-approximating set of \( f \) as well. Obviously, for any \( K > 1 \), \( \{0,...,U\} \) is a \( K \)-approximation set of \( f \). We are interested in finding “small” \( K \)-approximation sets for \( f \). Let \( U \geq \max_{x=0,...,U} f(x) \) be an arbitrary upper bound for the values of \( f \).

The next lemma shows how to construct \( K \)-approximations.

Lemma 3.5 Let \( f : [0,...,U] \rightarrow \mathbb{Z}^+ \) be a nondecreasing function. For every \( K > 1 \) there exists a weak \( K \)-approximation set \( S_1 \) for \( f \) of cardinality \( O\left( \frac{1}{K} \log U \right) \) and a \( K \)-approximation set \( S_2 \) for \( f \) of cardinality \( O\left( \frac{1}{K} \log U \log U \right) \). Furthermore, it takes \( O\left( \frac{1+t_f}{K} \log U \log U \right) \) time to construct these two sets, where \( t_f \) is the time needed to evaluate \( f \).

Proof. We provide an algorithm to construct both \( S_1 \) and \( S_2 \). We initially let \( S_1 = \emptyset \). We add elements to \( S_1 \) as follows. Let \( T_1 = \{i \mid 0 < i \leq U, f(i) \leq K f(0)\} \). If \( T_1 \) is empty, we set it to be \( T_1 = \{0\} \) and say that it is created by Rule 2, otherwise we say it is created by Rule 1. We let the first element in \( S_1 \) be the maximal element in \( T_1 \), i.e., \( i_1 = \max T_1 \). If \( i_1 < U \), we let \( T_2 = \{i \mid i_1 < i \leq U, f(i) \leq K f(i_1)\} \) (Rule 1) and again, if \( T_2 \) is empty, we correct it to be \( T_2 = \{i_1 + 1\} \) (Rule 2). We let \( i_2 = \max T_2 \). If \( i_2 < U \) we continue and construct the elements \( i_3, ..., i_r \) in \( S_1 \) in a similar way until we reach \( U \).

Due to the construction, the sequence \( i_1, i_2, ..., i_r \) is strictly increasing, so the algorithm terminates after at most \( U \) steps. Moreover, we have \( i_{k+2} \geq i_{k+1} + 1 \), so due to the monotonicity of \( f \) we get that \( f(i_{k+2}) \geq f(i_{k+1} + 1) > K f(i_k) \). Hence the cardinality of \( S \) is bounded above by \( O(\log_K U) = O\left( \frac{1}{K} \log U \right) \). Since \( f \) is monotone, we can perform binary
search in order to determine the maximum element in each of the $T_j$’s in $O((1 + t_k) \log U)$ time. Hence we can construct $S_1$ in $O(\frac{1 + t_f}{K} \log U \log U)$ time.

It remains to show that $S_1$ is a $K$-approximation set of $f$. Clearly $S_1 \subseteq \{0, ..., U\}$, so the first property in Definition 3.3 holds. If $T_{k+1}$ is created by Rule 1, then $f(i_{k+1}) \leq Kf(i_k)$. If $T_{k+1}$ is created by Rule 2, then $i_{k+1} = i_k + 1$ and $f(i_{k+1}) > Kf(i_k)$. Hence the second property holds as well. Let $0 \leq x \leq U$, and let $i_k$ be the smallest element in $S_1$ which is greater than or equal to $x$. If $i_k = x$, then $f(x) = f(i_k) < Kf(x)$. Otherwise $i_k > i_{k-1} + 1$, so due to the monotonicity of $f$ and the second property we get $f(x) \leq f(i_k) \leq Kf(i_{k-1}) \leq Kf(x)$, so the third property is satisfied.

We construct $S_2$ from $S_1 = \{i_1, ..., i_r\}$ in the following way. We initially let $S_2 = S_1$. If $0 \notin S_2$, we add it to $S_2$. If $S_2$ is still not a $K$-approximating set of $f$, there exists $1 \leq k \leq r - 1$ such that $i_{k+1} - i_k$ is not a power of 2. For every such pair of successive indices $i_k$ and $i_{k+1}$ we add to $S_2$ a set $J_k$ of elements $i_k < j < i_{k+1}$ such that the differences between consecutive indices in $J_k \cup \{i_k, i_{k+1}\}$ are all powers of 2. It is easy to see that there exists such a set $J_k$ with cardinality at most $O(\log U)$. Hence the cardinality of $S_2$ is $O(\frac{1}{K} \log U \log U)$, and the time needed to construct it is $O(\frac{1 + t_f}{K} \log U \log U)$. □

We note that sets $S_1$ and $S_2$ in Table 1 are built by the algorithm described in the above proof.

We use $K$-approximation sets to construct approximations in the following way.

**Definition 3.6** Let $K > 1$ and let $f : [0, ..., U] \rightarrow \mathbb{Z}^+$ be a nondecreasing function. Let $S$ be a $K$-approximation set of $f$. A function $\hat{f}$ defined as follows is called the approximation of $f$ corresponding to $S$. For any integer $0 \leq x \leq U$ and successive elements $i_k, i_{k+1} \in S$ with $i_k \leq x \leq i_{k+1}$ let

$$\hat{f}(x) := \frac{x - i_k}{i_{k+1} - i_k} f(i_k) + \frac{i_{k+1} - x}{i_{k+1} - i_k} f(i_{k+1}).$$

Note that $f|_S \equiv \hat{f}|_S$. Also note that if we calculate the values of $f$ on $S$ in advance, then any query for the value of $\hat{f}(x)$, for any $x$, can be calculated in $O(\log |S|)$ time. This is done by performing binary search on $S$ to find the consecutive elements $i_k, i_{k+1} \in S$ such that $i_k \leq x \leq i_{k+1}$.

In the next proposition we show that an approximation of a convex function $f$, corresponding to a given $K$-approximation set $S$ of $f$, is a $K$-approximation of $f$.

**Proposition 3.7** Let $K > 1$, let $f : [0, ..., U] \rightarrow \mathbb{Z}^+$ be a nondecreasing function, and let $S$ be a $K$-approximation set of $f$. If $\hat{f}$ is the approximation of $f$ corresponding to $S$, then $\hat{f}$ is a piecewise-linear nonnegative nondecreasing function satisfying $\hat{f}(x) \leq K f(x)$ for any $0 \leq x \leq U$, and $U \hat{f}$ is an integer valued function. Moreover, if $f$ is convex, then $\hat{f}$ is convex and $\hat{f} \geq f$, and therefore $\hat{f}$ is a $K$-approximation of $f$.
Proof. Due to $f$ being nonnegative and nondecreasing and due to the third property of $K$-approximating sets, we get that there exist indices $i_k, i_{k+1} \in S$ such that $i_k \leq x \leq i_{k+1}$ and

$$\hat{f}(x) = \frac{x - i_k}{i_{k+1} - i_k} f(i_k) + \frac{i_{k+1} - x}{i_{k+1} - i_k} f(i_{k+1}) \leq \frac{x - i_k}{i_{k+1} - i_k} f(x) + \frac{i_{k+1} - x}{i_{k+1} - i_k} K f(x) \leq K f(x),$$

as needed. If in addition $f$ is convex, then

$$f(x) \leq \frac{x - i_k}{i_{k+1} - i_k} f(i_k) + \frac{i_{k+1} - x}{i_{k+1} - i_k} f(i_{k+1}) = \hat{f}(x),$$

and consequently $\hat{f}$ is a $K$-approximation of $f$. Since $S$ is a $K$-approximation set of $f$, $i_{k+1} - i_k$ divides $U$ and therefore $U \hat{f}$ is an integer valued function. \qed

We extend the definition of $K$-approximation sets to nonnegative nonincreasing functions and to unimodal functions in the following way.

**Definition 3.8** Let $K > 1$ and let $f : [-U, ..., 0] \rightarrow \mathbb{Z}^+$ be a nonincreasing function. Let $\hat{f}(x) := f(-x)$, for $x$ in $[0, ..., U]$. A $K$-approximation set of $f$ is an ordered set $S = \{i_1 > ... > i_r\}$ of integers such that $\{-i_1, ..., -i_r\}$ is a $K$-approximation set of $\hat{f}$.

**Definition 3.9** Let $K > 1$ and let $f : [-U, ..., U] \rightarrow \mathbb{Z}^+$ be a unimodal function, which attains its minimum at 0. Let $f, \hat{f}$ be the restriction of $f$ on $[-U, ..., 0], [0, ..., U]$, respectively. A $K$-approximation set of $f$ is an ordered set $S = \{k_{\ell} < ... < k_1 \leq k_1 < ... < k_{\ell}\}$ such that $\{k_1, ..., k_{\ell}\}, \{\tilde{k}_1, ..., \tilde{k}_{\ell}\}$ is a $K$-approximation set of $f, \hat{f}$, respectively.

We note that a convex function is in particular a unimodal function, and we can extend in a natural way Definition 3.6 to convex functions $f : [-U, ..., U] \rightarrow \mathbb{Z}^+$. If $f$ attains its minimum at $x^*$ (if $x^*$ is not unique, we set $x^* = \min\{\arg\min_{x \in [-U, ..., U]} f(x)\}$ to be the smallest such minimizer), by defining $g(x) = x - x^*$, using property 3 in Proposition 3.2 and using similar arguments to those used in the proof of Lemma 3.5, we get that Lemma 3.5 and Proposition 3.7 hold for such functions as well.

**Lemma 3.10** Let $f : [-U, ..., U] \rightarrow \mathbb{Z}^+$ be a convex function. For every $K > 1$ there exists a weak $K$-approximation set $S_1$ for $f$ of cardinality $O(\frac{1}{K} \log U)$, and a $K$-approximation set $S_2$ for $f$ of cardinality $O(\frac{1}{K} \log U \log U)$. Furthermore, it takes $O(\frac{1+\tau_f}{K} \log U \log U)$ time to construct such sets, where $t_f$ is the time needed to evaluate $f$.

**Proposition 3.11** Let $K > 1$, let $f : [-U, ..., U] \rightarrow \mathbb{Z}^+$ be a convex function, and let $S$ be a $K$-approximation set of $f$. The approximation $\hat{f}$ of $f$ corresponding to $S$ is a piecewise-linear nonnegative convex function. Moreover, $\hat{f}$ is a $K$-approximation of $f$, and $U \hat{f}$ is an integer valued nonnegative convex function.
4 An FPTAS

In this section we develop an FPTAS for the lot-sizing problem defined in Section 2.

4.1 Preliminaries

Let \( g_t(I_t) \) denote the optimal total expected cost for periods \( t, \ldots, T \) starting in period \( t \) with an inventory of \( I_t \). Therefore our goal is to calculate \( z^* = g_1(0) \). Let \( r_j(\bar{I}_t) \) be the expected cost for periods \( t, \ldots, T \) if the inventory at the end of period \( t \) is \( \bar{I}_t \). It follows that for \( t = 1, \ldots, T \) we have

\[
    r_t(\bar{I}_t) = h_t(\bar{I}_t) + g_{t+1}(\bar{I}_t),
\]

and

\[
    g_t(I_t) = \min_{x_t \in \mathbb{Z}^+} \{ c_t(x_t) + \sum_{j=1}^{n_t} p_{t,j} r_t(I_t + x_t - d_{t,j}) \},
\]

where \( g_{T+1}(y) = 0 \) for any \( y \).

The inventory \( I_t \) in the beginning of time period \( t \) following an optimal policy satisfies

\[
    -D^* \leq -\sum_{j=1}^{t-1} d_{j,d_j} \leq I_t \leq D^* - \sum_{j=1}^{t-1} d_{j,d_j} \leq D^* \text{ for every } t = 1, \ldots, T.
\]

Moreover, for every time period \( t \), the inventory \( I_t + x_t \) after the procurement decision has been made satisfies

\[
    -D^* \leq -\sum_{j=1}^{t-1} d_{j,d_j} \leq I_t + x_t \leq D^* - \sum_{j=1}^{t-1} d_{j,d_j} \leq D^*.
\]

As a result in (2), \( x_t \) can be restricted to take values between 0 and \( D^* \). We point out that the running time for computing the values of \( g_t \) and \( r_t \) by dynamic programming, for all possible optimal inventory levels and for every period \( t = 1, \ldots, T \), is \( O(TD^2) \), i.e., pseudo-polynomial in the input size.

We next state a basic property of functions \( r_t \) and \( g_t \).

**Proposition 4.1** For every \( t = 1, \ldots, T \) functions \( r_t \) and \( g_t \) are convex over \( \mathbb{Z} \).

It is straightforward to prove this proposition by induction, the definition of convexity and the following well known result,

**Proposition 4.2** Let \( a, b, c : A \rightarrow \mathbb{R} \) be functions over a convex set \( A \) and let \( b, c \) be convex over \( A \). For all \( y \in A \) if we define \( a(y) = \min_{x \in A} \{ b(x) + c(y + x) \} \), then \( a \) is a convex function over \( A \) as well.

In order to develop an approximation scheme we first show that all the values of \( g_t(\cdot) \) and \( r_t(\cdot) \) over all inventory levels and time periods are rational numbers with a bounded least common multiple of their denominators.

**Proposition 4.3** For every \( t = 1, \ldots, T \) functions \( M_{t+1}r_t \) and \( M_t g_t \) are nonnegative, integer valued and convex over \( \mathbb{Z}^+ \).
Proof. The proof is by induction. Since $r_T \equiv h_T$, and $g_{T+1} \equiv 0$, $r_T$ is an integer function. Considering $g_T$ we have

$$g_T(I_T) = \frac{1}{Q_T} \max_{x_T} \{Q_Tc_T(x_T) + \sum_{j=1}^{n_T} q_{T,j}r_T(I_T + x_T - n_{T,j})\}.$$ 

Since $c_T$ and $r_T$ are integer valued functions, $Q_Tg_T$ is an integer valued function as well. Assuming by induction that the statement holds for $t = k + 1$, we get immediately from (1) that the statement holds for $r_k$ as well. From (2) we have

$$g_k(I_k) = \frac{1}{Q_k} \max_{x_k} \{Q_kc_k(x_k) + \sum_{j=1}^{n_k} q_{k,j}r_k(I_k + x_k - n_{k,j})\}.$$ 

Since $c_k$ and $M_{k+1}r_k$ are both integer valued functions, by the induction hypothesis, $M_kg_k$ is an integer valued function as well. We conclude the proof by applying Proposition 4.1. ∎

In the rest of the paper we set $U = 2^k$, where $k$ is the unique integer number satisfying $2^{k-1} < D^* \leq 2^k$. By the discussion proceeding Proposition 4.1 we can restrict the range of $r_t$ and $g_t$ to be $[-U,...,U]$. Note that $k = O(\log D^*)$.

### 4.2 Algorithm

In this section we develop an approximating scheme for $r_t$ and $g_t$ for $t = 1,...,T$. The algorithm proceeds backwards from $t = T$ down to $t = 1$. Given $g_{t+1}$, we show how to approximate $r_t$ and $g_t$. For $t = 1,...,T-1$ the algorithm actually uses an approximation of $g_{t+1}$. Recall that we need to approximate $g_0(0)$.

Let

$$y_t(z) = \sum_{j=1}^{n_t} p_{t,j}r_t(z - d_{t,j}), \quad Y_t(z) = M_{t+1}y_t(z).$$

Here $y_t(z)$ is the expected total cost for periods $t,...,T$ after a procurement decision has been made in period $t$, and the procurement plus the inventory level is $z$. Recall from Proposition 4.3 that $M_{t+1}r_t$ is a nonnegative integer convex function over $[-U,...,U]$. Note that by the definition of $Q_t$ and the $p_{t,j}$’s it follows that $Q_tY_t$ is an integer nonnegative convex function.

Let $\psi_t$ be a $K$-approximating set of $M_{t+1}r_t$. Let $\hat{R}_t$ be its corresponding $K$-approximation (instead of the more accurate notation $\hat{M}_{t+1}r_t$) and let $\hat{r}_t = \frac{\hat{R}_t}{M_{t+1}}$. Due to Proposition 3.2 we have that

$$\hat{Y}_t(z) = \sum_{j=1}^{n_t} p_{t,j}\hat{R}_t(z - d_{t,j})$$

9
is a $K$-approximation of $Y_t$, and therefore
\[
Y_t(z) \leq \hat{Y}_t(z) \leq K Y_t(z). \tag{3}
\]

Due to Proposition 3.11, $U\hat{R}_t$ is an integer valued nonnegative convex function, and since $\hat{Y}_t$ is a convex combination of the $\hat{R}_t$'s, $UQ_t\hat{Y}_t$ is a nonnegative integer convex function. We can use the same argument in order to compute a $K$-approximating set $\mathcal{Y}_t$ for $UQ_t\hat{Y}_t$.

Let $\bar{Y}_t$ denote its corresponding $K$-approximation (instead of the more accurate notation $\hat{UQ_t}\hat{Y}_t$). Due to (3) we have
\[
Y_t(z) \leq \hat{Y}_t(z) \leq K \bar{Y}_t(z) \leq K^2 Y_t(z). \tag{4}
\]

Let
\[
\bar{y}_t(z) = \frac{\hat{Y}_t(z)}{UM_t}.
\]

From (4) we get that for every $z$
\[
y_t(z) \leq \bar{y}_t(z) \leq K^2 y_t(z), \tag{5}
\]
and therefore $\bar{y}_t(z)$ is a $K^2$-approximation of $y_t(z)$.

We now proceed with an approximation to $g_t$. For $A \subset \mathbb{R}$ and $x \in \mathbb{R}$ let $A - x := \{a - x \mid a \in A\}$. First we invoke Lemma 3.10 to compute a $K$-approximating set $\Phi_t$ for the procurement cost function $c_t$. Let
\[
\bar{G}_t(z) = \min_{x \in \Phi_t \cup (Y_t - z)} \{UM_t c_t(x) + \hat{Y}_t(z + x)\}, \quad \bar{g}_t(z) = \frac{\bar{G}_t(z)}{UM_t}.
\]

We now show that $\bar{g}_t(z)$ is a $K^2$-approximation of $g_t(z)$ for every $z$. Let us fix $z$. Let $\bar{x} \in \Phi_t \cup (Y_t - z)$ attain the minimum in the definition of $\bar{g}_t(z)$, and let $x^*$ be the smallest value attaining the minimum in the definition of $g_t(z)$. From (5) we get that $\bar{g}_t(z) = c_t(\bar{x}) + \bar{y}_t(z + \bar{x}) \geq c_t(\bar{x}) + y_t(z + \bar{x}) \geq g_t(z)$. It remains to show that $\bar{g}_t(z) \leq K^2 g_t(z)$. By the construction of $\Phi_t$ there exists $x' \in \Phi_t$ such that $x' \geq x^*$ and $c_t(x^*) \leq c_t(x') \leq K c_t(x^*)$.

We consider two cases.

**Case 1.** Suppose that $\bar{g}_t(z + x' - 1) \geq \bar{g}_t(z + x')$, see Figure 1. From (5) we get that $\bar{g}_t(z + x') \leq \bar{y}_t(z + x^*) \leq K^2 y_t(z + x^*)$. Hence in this case $\bar{g}_t(z) \leq c_t(x') + \bar{y}_t(z + x') \leq K c_t(x^*) + K^2 y_t(z + x^*) \leq K^2 g_t(z)$.

**Case 2.** Let now $\bar{g}_t(z + x' - 1) < \bar{g}_t(z + x')$, see Figure 1. Let $z + x''$ be the smallest value attaining the minimum of $\bar{g}_t$. Note that $z + x'' \leq z + x'$, i.e., $x'' < x'$. Due to the definition of $K$-approximating sets $x'' \in Y_t - z$. Hence $\bar{g}_t(z) \leq c_t(x'') + \bar{y}_t(z + x'') \leq c_t(x') + \bar{y}_t(z + x^*) \leq K c_t(x^*) + K^2 y_t(z + x^*) \leq K^2 g_t(z)$ as needed.
We summarize,
\[ g_t(z) \leq \hat{g}_t(z) \leq K^2 g_t(z), \]
and therefore \( \hat{g}_t(z) \) is a \( K^2 \)-approximation of \( g_t(z) \).

\[ \text{Figure 1: The two cases} \]

Due to Proposition 3.11, \( U^T \hat{y}_t \) is an integer nonnegative convex function, and therefore \( U^T \hat{G}_t \) is such a function as well. We use once again Lemma 3.10 in order to compute a \( K \)-approximating set \( \Gamma_t \) for \( U^T \hat{G}_t \). Let \( \hat{G}_t \) denote its corresponding \( K \)-approximation (instead of the more accurate notation \( \overline{U^T \hat{G}_t} \)), so by letting \( \hat{g}_t(z) = \frac{{\hat{G}_t}}{{U^T_M}} \) we get that
\[
G_t(z) \leq \hat{G}_t(z) \leq K^3 G_t(z), \quad g_t(z) \leq \hat{g}_t(z) \leq K^3 g_t(z).
\]

We conclude that \( \hat{g}_t(z) \) is a \( K^3 \)-approximation of \( g_t(z) \).

We wrap up this section with a description of the algorithm. In the first iteration the algorithm computes an approximation of \( r_T \) and \( g_T \). Recalling that \( g_{T+1} \equiv 0 \), (1) gives us an exact formula for \( r_T \). We proceed as described above to get \( \hat{g}_T \), a \( K^3 \)-approximation of \( g_T \). In the second iteration the algorithm computes an approximation of \( r_{T-1} \) and \( g_{T-1} \). Since \( r_{T-1} \) is unavailable, the algorithm computes \( r_{T-1} = h_{T-1} + \hat{g}_T \) instead, where \( h_{T-1} \) is given in the input data, and \( \hat{g}_T \) is the \( K^3 \)-approximation of \( g_T \) which was calculated in the first iteration. Next, again as described above, we use \( \hat{r}_{T-1} \) and \( \hat{c}_{T-1} \) to compute \( \hat{g}_{T-1} \). In general, assuming that approximations \( \hat{g}_{t+1}, ..., \hat{g}_T \) and \( \hat{r}_{t+1}, ..., \hat{r}_T \) for \( g_{t+1}, ..., \hat{g}_T \) and \( r_{t+1}, ..., r_T \), respectively, have already been obtained, based on the described procedure, we calculate \( \hat{r}_t \) and \( \hat{g}_t \). We stop when \( \hat{g}_1 \) is computed. The algorithm outputs \( \hat{g}_1(0) \).

4.3 Analysis

In this section we show that the algorithm produces an FPTAS for a suitable choice of \( K \). Let \( \epsilon > 0 \) be given, where we seek an \( \epsilon \)-approximation.
Lemma 4.4 For every time period $t = 1, \ldots, T$, $\hat{r}_t$ is a $\mathcal{K}^{3(T-t)}$-approximation of $r_t$ and $\hat{g}_t$ is a $\mathcal{K}^{3(T-t+1)}$-approximation of $g_t$.

Proof. We prove the statement by induction. The base case of time period $T$ is correct as explained in the description of the algorithm. We assume correctness for time period $t$ and proceed to time period $t - 1$. Considering (1) for $r_{t-1}$, since $g_t$ is unavailable, the algorithm uses $\hat{g}_t$ instead, i.e., it uses $h_{t-1} + \hat{g}_t$ instead of $r_{t-1} = h_{t-1} + g_t$. By the induction hypothesis $g_t \leq \hat{g}_t \leq \mathcal{K}^{3(T-t+1)} g_t$. Therefore, by the second property in Proposition 3.2, $h_{t-1} + \hat{g}_t$ is a $\mathcal{K}^{3(T-t+1)}$-approximation of $r_{t-1}$ and the inductive step for $r_{t-1}$ is completed. For $g_{t-1}$ a factor of $\mathcal{K}^{3(T-t+1)}$ is added to each of the formulas (3)-(7), and the inductive step is completed for $g_{t-1}$ as well. 

It remains to prove that the approximation procedure described above yields an FPTAS for our problem.

Theorem 4.5 The outlined approximation algorithm gives an FPTAS for the single-item discrete stochastic lot-sizing problem when $K = 1 + \frac{\epsilon}{\mathcal{K}^T}$.

Proof. By Lemma 4.4 the approximation $\hat{g}_1(0)$ of the optimal total expected cost in periods $1, \ldots, T$ starting in period 1 with an inventory of 0 satisfies $g_1(0) \leq \hat{g}_1(0) \leq \mathcal{K}^{3T} g_1(0)$. Setting $K = 1 + \frac{\epsilon}{\mathcal{K}^T}$ gives $g_1(0) \leq \hat{g}_1(0) \leq (1 + \frac{\epsilon}{\mathcal{K}^T})^{3T} g_1(0)$. From the inequality $(1 + \frac{x}{n})^n \leq 1 + 2x$, which holds for every $0 \leq x \leq 1$, and since the optimal solution is $g_1(0)$, we get that $z^* \leq \hat{g}_1(0) \leq (1 + \epsilon)z^*$.

We analyze now the running time of our algorithm. Let

$$A = \sum_{t=1}^{T} (c_t(D^*) + \max \{ h_t(D^*), h_t(-D^*) \}), \quad B = 2TK^{3T} \max_t \{ c_t(D^*), h_t(D^*), h_t(-D^*) \}.$$ 

Note that $A$ is an upper bound for the values of the $r_t$’s and $g_t$’s. Clearly $A \leq 2T \max_t \{ c_t(D^*), h_t(D^*), h_t(-D^*) \}$. Due to Lemma 4.4, $B$ is an upper bound for the values of the $\hat{r}_t$’s and $\hat{g}_t$’s. Therefore, $\hat{U} = M_t U^2 B$ serves as an upper bound for the values of the functions for which we construct $\mathcal{K}$-approximating sets throughout the algorithm. Assumption 2.4 implies that $\hat{U}$ is a polynomial upper bound on the numbers calculated throughout the algorithm. Due to Lemma 3.10, in every time period $t = T, T - 1, \ldots, 1$ we construct the $\mathcal{K}$-approximating sets $\Psi_t, \Upsilon_t, \Phi_t,$ and $\Gamma_t$ of $M_{t+1} r_t, U Q_t \Upsilon_t, c_t,$ and $U G_t$, respectively, and calculate the values of their corresponding $\mathcal{K}$-approximations over these sets in $O(\frac{1}{\mathcal{K}} \log \hat{U} \log U)$ time and the same number of evaluations. Each evaluation of $M_{t+1} r_t, U Q_t \Upsilon_t,$ and $U G_t$ takes at most $1, n^*$ and $O(\log(\mathcal{F}) + \mathcal{Y})) = O(\log(\frac{1}{\mathcal{K}} \log \hat{U} \log U))$ queries of the corresponding $\mathcal{K}$-approximating function, respectively. Thus, the total running time of the algorithm and number of queries the algorithm performs is $O(T(n^* + \log \frac{1}{\mathcal{K}} \log \hat{U} \log U))(\frac{1}{\mathcal{K}} \log \hat{U} \log U)$.
Replacing $\bar{U}, U$ by their bounds, $K$ with $1 + \frac{e}{\sqrt{T}}$, and a query time by $t_f$ we conclude that the running time of the algorithm is

$$O((1 + t_f) T (n^* + \log(\frac{1}{K} \log(M_1 D^* B) \log D^*)) (\frac{1}{K} \log(M_1 D^* B) \log D^*)),$$

which is polynomial in the binary size of the input data. \[\square\]

5 Extensions

5.1 Capacitated version

It is easily seen that convex cost functions can be adapted to effectuate procurement capacity limits by making the procurement cost sufficiently high beyond these limits, while preserving convexity. Thus, our results hold also for single-item stochastic capacitated lot-sizing problems with discrete demands.

5.2 Discounted version

In the discounted single-item stochastic lot-sizing problem we are also given a rational discounting factor $0 < \lambda = \frac{\lambda_1}{\lambda_2} < 1$, where $\lambda_1$ and $\lambda_2$ are positive integers. In this version the objective function changes to

$$z^* = \min E_D(\sum_{i=1}^{T} \lambda^{t-1} (c_t(x_t) + h_t(\bar{I}_t + x_t - D_t))).$$

It is easy to adapt our algorithm to this case. The recursive formula (1) for $r_t$ changes to

$$r_t(\bar{I}_t) = h_t(\bar{I}_t) + \lambda g_{t+1}(\bar{I}_t), \quad (8)$$

while (2) remains intact. We need to account for the fact that $r_t(\bar{I}_t)$ is not necessarily integer valued even if $g_{t+1}(\bar{I}_t)$ is integer valued. However, by multiplying (8) by $\lambda_2$, i.e., by changing $Q_t$ to be $Q_t = \lambda_2 \sum_{j=1}^{n_t} g_{t,j}$, we recover integrality, and we can use the same algorithm. The analysis of the running time can be easily extended. The only change is that $M_1$ is multiplied by $\lambda_2^T$, and therefore the running time of the entire algorithm is multiplied by $O(T \log \lambda_2)$, i.e., remains polynomial in the binary size of the data.

5.3 Non-convex cost functions

We can also approximate a version of the problem where the cost functions need not be convex. We replace Assumption 2.2 and Assumption 2.3 by the following two assumptions.
**Assumption 5.1** The procurement cost function $c_t$ is nondecreasing in $[0, \infty]$, and $c_t(0) = 0$ for every $t = 1, \ldots, T$.

**Assumption 5.2** The holding cost function $h_t$ is nonnegative. Furthermore, it is nondecreasing in $[0, \infty]$, nonincreasing in $[-\infty, 0]$, and $h_t(0) = 0$ for all $t = 1, \ldots, T$.

Unfortunately, at the expense of losing convexity, we must add the following assumption.

**Assumption 5.3 (Free disposal assumption)** Inventory can either be held from one period to the next, or it can be disposed of at no cost.

Note that under the free disposal assumption, we can handle economies of scales in procurement, i.e., the fixed plus variable procurement cost.

Our algorithm can be modified to this version as well. We change the recursive formula (1) for $r_t$ to

$$ r_t(I_t) = \begin{cases} \min \{ r_t(I_t - 1), h_t(I_t) + g_{t+1}(I_t) \} = \min_{x \geq 0} \{ h_t(x) + g_{t+1}(x) \} & \text{for } I_t > 0, \\ h_t(I_t) + g_{t+1}(I_t) & \text{for } I_t \leq 0, \end{cases} $$

while leaving (2) intact. In this way $r_T(y) = 0$ for any $y \geq 0$. It is easy to see that $r_t$ is nonincreasing for every $t = 1, \ldots, T$, and consequently $g_t$ is such as well. In this case, instead of using Definition 3.6, we use the following alternative definition of $K$-approximation functions.

**Definition 5.4** Let $K > 1$ and let $f : [0, \ldots, U] \rightarrow \mathbb{Z}^+$ be a nondecreasing function. Let $S$ be a $K$-approximation set of $f$. A function $\hat{f}$ defined as follows is called the approximation of $f$ corresponding to $S$. For any integer $0 \leq x \leq U$ and successive elements $i_k, i_{k+1} \in S$ with $i_k \leq x \leq i_{k+1}$ let $\hat{f}(x) := f(i_{k+1})$.

Instead of Proposition 3.7 we use the following proposition.

**Proposition 5.5** Let $K > 1$, let $f : [0, \ldots, U] \rightarrow \mathbb{Z}^+$ be a nondecreasing function, and let $S$ be a $K$-approximation set of $f$. If $\hat{f}$ is the approximation of $f$ corresponding to $S$, then $\hat{f}$ is a step function, it is nonnegative and nondecreasing, and is a $K$-approximation of $f$.

The algorithm described in Subsection 4.2 constructs an FPTAS for this case as well, using the updated recursive formula for $r_t$. However, since $\hat{G}_t$ is not a convex function in this case, we cannot perform binary search on the values of $\Phi_t \cup Y_t$ in order to calculate the value of $G_t(z)$ for fixed $t$ and $z$. We perform all $O(|\Phi| + |\mathcal{T}|)$ evaluations of $c_t$ and $\hat{Y}_t$ instead. This slightly increases the total running time of the algorithm to $O((1 + t_f)T(n^* + \frac{1}{\mathcal{R}} \log(M_1D^*B) \log D^*)^2 + \frac{1}{\mathcal{R}} \log(M_1D^*B) \log D^*)$, while preserving polynomial running time in the binary size of the input data. Note that in this case the proof that $\hat{g}_t$ is a $K^2$-approximation of $g_t$ becomes simpler: Since $\hat{g}_t$ is nonincreasing, only the first case applies.
6 Hardness result

In this section we show that the single-item stochastic lot-sizing problem is NP-hard even in the case of linear procurement and holding costs. We make a transformation from the $K$-th largest subset sum problem, which is known to be NP-hard (see for example problem SP20 in page 225 in 3).

Problem: $K$-th largest subset sum

Input: A finite set $A = \{a_1, ..., a_n\}$ of nonnegative integers and a positive integer number $K \leq 2^n$.

Output: A subset of $A$ where the sum of its elements is the $K$’s largest.

Theorem 6.1 The single-item stochastic lot-sizing problem is NP-hard.

Proof. Given an instance of the $K$-th largest subset sum problem we transform it into the following instance of the single-item stochastic lot-sizing problem. Let $M = 2^n \max \{2^n, \sum_{j=1}^{n} a_i\}$. We have

- $T = n$ time periods, labeled 1, ..., $n$.
- Demand in time period $t$ is $d_t = \begin{cases} 0 & \text{with probability } \frac{1}{2}\text{ for all } t; \\ a_j & \text{with probability } \frac{1}{2}\text{ for all } t. \end{cases}$
- Production cost in time period $t$ is $c_t(x) = \begin{cases} (2^n - K)x & \text{for } t = 1; \\ Mx & \text{for } t = 2, ..., n. \end{cases}$
- Holding cost in time period $t$ is $h_t(x) = \begin{cases} 0 & \text{for } x \geq 0 \text{ and all } t; \\ 0 & \text{for } x < 0 \text{ and } t = 1, ..., n-1; \\ K & \text{for } x < 0 \text{ and } t = n. \end{cases}$

Since $c_t(x) \gg c_1(x)$ for $t = 2, ..., n$, it is clear that the optimal policy is to order only in time period 1. Let $S$ be the number of units an optimal policy orders in time period 1. If $S$ is larger than the realized demand $D$ from time periods 1 to $n$, then the cost is $(2^n - K)S$, due to production cost in time period 0, and no backlogging in time period $n$. If $S$ is smaller than the realized demand $D$, then the cost is $(2^n - K)S + K(D - S)$, due to production cost in time period 1, and backlogging cost in time period $n$.

This reduces to the newsvendor problem, where the unit cost of ordering one item too many is $m = 2^n - K$, and the cost of ordering one item too few is $l = K$. Therefore, the optimal decision is to produce the minimum amount $S$ such that $\Pr(D \leq S) \geq \frac{l}{m+l} = \frac{K}{2^n}$, where $D$ is a random variable describing the demand (see, e.g., [6], Section 8.2.1). Note that in our case we have

$$P(D \leq S) = \frac{\text{number of subsets } I \subseteq \{1, ..., n\} \text{ such that } \sum_{i \in I} a_i \leq S}{2^n}.$$
Therefore, $S$ equals to the $K$’th largest subset. Since it is NP-hard to determine the $K$’th largest subset, it is NP-hard to solve the single-item stochastic lot-sizing problem. □

We point out that we do not know how to show that the single-item stochastic lot-sizing problem is in $NP$. Note that the $K$-th largest subset sum problem is also $\#P$-hard$^1$, and therefore the single-item stochastic lot-sizing problem is $\#P$-hard as well. Most $\#P$-complete problems exhibit a randomized FPTAS, e.g., counting Hamiltonian cycles in dense graphs [2], counting knapsack solutions [1], counting Eularian orientations of a graph [5], counting perfect matching in a bipartite graph ???, and computing the permanent [4]. To the best of our knowledge$^2$, our algorithm is the first deterministic FPTAS (without using derandomization) for a $\#P$-hard problem.

7 Conclusions and Future Research

We presented the first FPTAS for the single-item stochastic lot-sizing problem. Differently from recent developments in approximation algorithms for stochastic dynamic and multistage programs, which are based on gradients or sampling and either offer a non polynomial algorithm or cannot be applied to lot-sizing, our framework is based on the notion of approximating sets and functions. We still use the standard optimality equation or recursion, however, we consider only polynomially many states. Our algorithm relies on either convexity or monotonicity of the value function.

Some of the extensions to the basic model, which do not require substantial modifications to the algorithm, were already presented in Section 5. Another extension, which is work in progress, is about relaxing the assumption of zero lead time. Under general lead times the value function is multivariate. We note that the standard transformation to the single variate value function does not preserve the approximation ratio since it offsets the optimal value. We are currently working on an extension to the presented framework to multivariate convex value functions. Yet another interesting extension and work in progress is the consideration of the infinite time horizon problem under stationary data.

We believe a challenging problem is the incorporation of economies of scale. It is well known that if economies of scale are present in the procurement cost, then the value function is no longer convex. We are not sure if the presented framework can be applied under such a case. It is interesting to note that the sampling based techniques rely heavily on convexity as well.

$^1$Note from Nir: $K$-th largest subset sum is $\#P$-complete.
$^2$Note from Nir: To check out with “Stata building people”
References

[1] M. Dyer, *Approximate counting by dynamic programming*, Proceedings of the 35'th Annual ACM Symposium on the Theory of Computing, 693-699, 2003.

[2] M. Dyer, A. Frieze, and M. J., *Approximately counting Hamilton cycles in dense graphs*, Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms, 336-343, 1994.

[3] M.R. Garey and D.S. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*, W. H. Freeman, San Francisco, 1979.

[4] M. Jerrum, A. Sinclair and E. Vigoda, *A polynomial-time approximation algorithm for the permanent of a matrix with non-negative entries*, Proceedings of the 33'th Annual ACM Symposium on Theory of Computing, 712-721, 2001.

[5] M. Mihail and P. Winkler, *On the number of Eularian orientations of a graph*, Proceedings of the 3th Annual ACM-SIAM Symposium on Discrete Algorithms, 138 -145, 1992.

[6] D. Simchi-Levi, Xin Chen and Julien Bramel, *The logic of logistics: theory, algorithms, and applications for logistics and supply chain management*, Springer series in operations research, second edition, New York, 2005.