A geometric algorithm to solve the NI/G/NI/ND capacitated lot-sizing problem in O(T^2) time

Wilco van den Heuvel[a]*, Albert P.M. Wagelmans[a]†

[a] *Faculty of Economics, Erasmus University Rotterdam, Econometric Institute, P.O. Box 1738, 3000 DR Rotterdam, The Netherlands*

Econometric Institute Report EI 2003-24

Abstract

In this paper we consider the capacitated lot-sizing problem (CLSP) with linear costs. It is known that this problem is NP-hard, but there exist special cases that can be solved in polynomial time. The purpose of this paper is twofold. First, we derive a backward algorithm based on the forward algorithm by Chen et al. (1994), which solves the general CLSP. We give a problem instances that requires exponential running time using the backward algorithm. Second, we provide a new O(T^2) algorithm for the CLSP with non-increasing setup cost, general holding cost, non-increasing production cost and non-decreasing capacities over time. This is done by adapting the backward algorithm. Numerical tests show the better performance of our algorithm compared to the algorithm proposed by Chung and Lin (1988).

Keywords: Production; Capacitated lot-sizing problem; Inventory

1 Introduction

In this paper the capacitated lot-sizing problem (CLSP) is considered. The problem can be described as follows. For a finite time horizon, there is a known demand for a single product. This demand has to be satisfied each period by producing in this period or in previous periods, i.e., back-logging is not allowed. When production occurs in a period, setup cost and marginal production cost per unit production are incurred. In contrast to the uncapacitated lot-sizing problem, production in each period is limited by a certain capacity. Finally, holding costs are

*Corresponding author. Tel.: +31-10-4081321, Email: wvandenheuvel@few.eur.nl

†Email: wagelmans@few.eur.nl

incurred for carrying ending inventory from one period to the next period and it is assumed that all cost functions are non-increasing.

Florian et al. (1980) show that the CLSP with general cost functions is NP hard and they suggest a pseudo-polynomial algorithm with complexity $O(T^2\overline{c}\overline{d})$, where $\overline{c}$ and $\overline{d}$ denote the average capacity and the average demand, respectively, and T denotes the model horizon. If marginal production costs are linear, Shaw and Wagelmans (1998) show that the complexity can be improved to $O(T\overline{c}\overline{d})$.

In this paper we consider a special case of the CLSP, namely the case where holding costs and marginal production costs are assumed to be linear. In the remainder of the paper we restrict ourselves to this special case. Although the uncapacitated version of this lot-sizing problem can be solved in polynomial time (Wagner and Whitin (1958), Federgruen and Tzur (1991), Wagelmans et al. (1992) and Aggarwal and Park (1993)), Bitran and Yanasse (1982) showed that the CLSP is NP hard, even in many special cases. These authors introduced the notation $\alpha/\beta/\gamma/\delta$ for the CLSP, where α, β, γ, δ specify a certain structure for respectively setup costs, holding costs, production costs and capacity. The parameters can have values equal to the letters Z, C, NI, ND and G, which stand for zero, constant over time, non-increasing over time, non-decreasing over time and no prespecified pattern, respectively. For example a NI/ND/Z/G problem consists of non-increasing setup costs, non-decreasing holding costs, no production costs and general capacities.

We now briefly summarize some complexity results. Bitran and Yanasse (1982) show that the following cases are NP hard: (1) C/Z/NI/NI, (2) C/Z/ND/ND, (3) ND/Z/Z/ND, (4) NI/Z/Z/NI, (5) C/G/Z/NI, (6) C/C/ND/NI. Florian and Klein (1971) provided an $O(T^4)$ algorithm for the G/G/G/C case and van Hoesel and Wagelmans (1996) improved this result to $O(T^3)$. Bitran and Yanasse (1982) formulated an $O(T^4)$, $O(T^3)$, $O(T \log T)$ and $O(T)$ algorithm for the cases NI/G/NI/ND, NI/G/NI/C, C/Z/ND/NI and ND/Z/ND/NI, respectively. Chung and Lin (1988) reduced the time complexity of the NI/G/NI/ND problem by presenting an $O(T^2)$ algorithm.

Chen et al. (1994) introduced a new algorithm to solve G/G/G/G cases of the CLSP. On their test problems, this forward dynamic programming (DP) algorithm has an empirical running time that increases quadratically relative to the time horizon T for C/C/Z/C problem instances. Furthermore, they show that computation time is not effected by changing the problem instances into C/C/C/G and G/G/G/G cases, which both are NP hard. The algorithm mainly consists of updating a piecewise linear minimum cost function by efficiently finding the lower envelope of a piecewise linear function and a number of linear line segments.

The purpose of this paper is twofold. First, we derive a backward algorithm that uses similar ideas as the forward DP algorithm by Chen et al. (1994) and we give a problem instance that requires exponential running time to be solved. Second, we provide a new $O(T^2)$ algorithm for the NI/G/NI/ND problem by adapting this backward algorithm. The remainder of the paper

is organized as follows. In section 2 the backward algorithm to solve the G/G/G/G problem is introduced. In section 3 we come up with a problem instance that requires exponential running time using this backward algorithm. In section 4 we introduce a new algorithm based on the backward algorithm, which solves the NI/G/NI/ND problem in time complexity $O(T^2)$. Chung and Lin (1988) proposed an algorithm with the same time complexity, but we show that our algorithm is at least as fast empirically. The paper is ended up with the conclusion in section 5.

2 Backward algorithm to solve the G/G/G/G problem

2.1 Problem description

The following notation is used to describe the lot-sizing problem. If we denote

T = model horizon
d_t = demand in period t
c_t = capacity in period t
K_t = setup costs in period t
p_t = unit production costs in period t
h_t = unit holding costs in period t
x_t = production quantity in period t
Inv_t = ending inventory in period t
D_t = cumulative demand in period t, $D_t = \sum_{i=1}^{t} d_i$
C_t = cumulative capacity in period t, $C_t = \sum_{i=1}^{t} c_i$

then the problem can be formulated as

$$
\begin{array}{lll}
\text{(P)} \quad \min & \sum_{t=1}^{T} K_t \delta(x_t) + p_t x_t + Inv_t h_t & \\
\text{s.t.} & Inv_t = Inv_{t-1} - d_t + x_t & t = 1, \ldots, T \\
& x_t \leq c_t & t = 1, \ldots, T \\
& x_t, Inv_t \geq 0, & t = 1, \ldots, T \\
& Inv_0 = 0 &
\end{array}
$$

where

$$
\delta(x) = \begin{cases} 0 & \text{for } x = 0 \\ 1 & \text{for } x > 0. \end{cases}
$$

The assumption that starting inventory is zero can be made without loss of generality. Furthermore, it has been proven by Bitran and Yanasse (1982) that a G/G/G/G problem can always be

reformulated to a problem with the property $d_t \leq c_t$ $(t = 1, \ldots, T)$ by transforming demand into

$$d'_t = \begin{cases} c_t + \max_{\tau=1,\ldots,t-1} \left\{0, \sum_{l=t+1}^{t+\tau}(d_l - c_l)\right\} & \\ -\max_{\tau=1,\ldots,t-1} \left\{0, c_t - d_t, \sum_{l=t+1}^{t+\tau}(d_l - c_l)\right\} & \text{for } t = 1, \ldots, T-1 \\ \min\{d_t, c_t\} & \text{for } t = T \end{cases}$$

and adding the term

$$\sum_{t=1}^{T} h_t \max_{\tau=1,\ldots,T-1} \left\{0, \sum_{l=t+1}^{t+\tau}(d_l - c_l)\right\}$$

to the objective function. Finally, we may also assume that holding costs equal zero. This can easily be derived by substituting $Inv_t = \sum_{i=1}^{t}(x_i - d_i)$ in (P) (see for example Chen et al. (1994). Then problem (P) can be written as

$$\min \sum_{t=1}^{T} K_t \delta(x_t) + p'_t x_t - h_t D_t \tag{1}$$

$$\text{s.t.} \sum_{i=1}^{t} x_t \geq D_t \qquad t = 1, \ldots, T \tag{2}$$

$$x_t \leq c_t \qquad t = 1, \ldots, T \tag{3}$$

$$x_t \geq 0 \qquad t = 1, \ldots, T \tag{4}$$

where

$$p'_t = p_t + \sum_{i=t}^{T} h_i, t = 1, \ldots, T. \tag{5}$$

2.2 The backward algorithm

In Chen et al. (1994) the CLSP is solved by a forward DP algorithm. In this section the backward version of this algorithm is examined. Although the derivation of the algorithm is almost similar to the derivation of the forward algorithm by Chen et al. (1994), we present the algorithm in full detail in order to build on this in the remainder of this paper. We define a minimum cost function $F_t(X)$, where X equals cumulative production of the first $t-1$ periods, i.e. $X = \sum_{i=1}^{t-1} x_i$. It is clear that cumulative production X should at least equal D_{t-1}, otherwise demand constraints (2) are violated. It follows from (3) that the maximum number of units produced up to period $t-1$ will never exceed C_{t-1}. It is also clear that cumulative production X will not exceed D_T. This means that the minimum cost function $F_t(X)$ is defined on the interval $I_t = [A_t, B_t]$, with $A_t = D_{t-1}$ and $B_t = \min\{C_{t-1}, D_T\}$. For $t = 1, \ldots, T$ we define the minimum cost function F_t as

$$F_t(X) = \begin{cases} \text{minimum cost for the period } t, \ldots, T \text{ production plan when total} & \\ \text{production in the first } t-1 \text{ periods equals } X = \sum_{i=1}^{t-1} x_i & \text{for } X \in I_t \\ \infty & \text{otherwise} \end{cases}$$

and let

$$F_{T+1}(X) = \begin{cases} 0 & X = D_T \\ \infty & X \neq D_T, \end{cases}$$

$$I_{T+1} = \{D_T\}.$$

The first stage corresponds to the computation of $F_t(X)$ for $t = T$ and we end up at recursion step $t = 1$, so that in recursion step t demand equals d_t. Now the minimum cost function $F_t(X)$ can be determined recursively by the recursion formula

$$F_t(X) = \min_{Y \in I_{t+1}} \{F_{t+1}(Y) + P_t(Y - X)\}, \tag{6}$$

where

$$P_t(x) = \begin{cases} 0 & \text{for } x = 0 \\ K_t + p_t x & \text{for } 0 < x \leq c_t \\ \infty & \text{otherwise} \end{cases}$$

for $t = 1, \ldots, T$ and $X \in I_t$. The interpretation of recursion formula (6) is as follows. Given the optimal production plan $F_{t+1}(Y)$ for periods $t+1, \ldots, T$, we can find the optimal production plan for some $X \in I_t$ by taking the minimum of $F_{t+1}(Y) + P_t(Y - X)$ for $Y \in I_{t+1}$, i.e. produce $Y - X$ units in period t. Note that the restriction $0 \leq Y - X \leq c_t$ is implicit in the definition of $P_t(x)$. The minimal cost of the overall optimal production plan can be found by calculating $z^* = F_1(0)$. We will call the execution of recursion formula (6) for a certain t a recursion step or an iteration. So the algorithm consists of T recursion steps or iterations.

Traditionally the variables are stored as discrete variables, but as in Chen et al. (1994) we consider continuous variables. We can show that $F_t(X)$ is actually a piecewise non-increasing linear function with a finite number of pieces. This means that for every piece of $F_t(X)$ the vertical interception, the slope and the interval have to be stored. Federgruen and Tzur (1991) and Van Hoesel et al. (1994) use this idea for the uncapacitated lot-sizing problem. In the appendix (theorem 8) it is shown that $F_t(X)$ is non-increasing. The following theorem shows that $F_t(X)$ is piecewise linear.

Theorem 1 *The minimum cost function $F_t(X)$ is piecewise linear for $t = 1, \ldots, T + 1$.*

Proof The theorem will be proven by induction. The proof for the backward algorithm is almost similar to the proof for the forward algorithm in Chen et al. (1994). For $t = T + 1$ the function $F_{T+1}(X)$ consists of one point. So the theorem is true for $t = T + 1$. Assume now that $F_{t+1}(X)$ is a piecewise linear function for some $t \leq T$. We will show that $F_t(X)$ is also a piecewise linear

function. First note that

$$F_t(X) = \min_{Y \in I_{t+1}} \{F_{t+1}(Y) + P_t(Y - X)\} \tag{7}$$

$$= \min\{\min_{Y \in I_{t+1}, Y = X} F_{t+1}(Y) + P_t(Y - X), \min_{Y \in I_{t+1}, Y \neq X} F_{t+1}(Y) + P_t(Y - X)\} \tag{8}$$

$$= \min\{F_{t+1}(X), \min_{Y \in I_{t+1}, X < Y} F_{t+1}(Y) + P_t(Y - X)\}. \tag{9}$$

Now it is sufficient to show that the second term is a piecewise linear function, because the lower envelope of two piecewise linear functions is a piecewise linear function and $F_{t+1}(X)$ is piecewise linear by assumption.

Assume now that $F_{t+1}(X)$ consists of m_{t+1} line segments $F^i_{t+1}(x) = \alpha_i - \beta_i x$ defined on the intervals $Dom_i = [a_i, b_i\rangle$ for $i = 1, \ldots, m_{t+1} - 1$ and on the interval $Dom_i = [a_i, b_i]$ for $i = m_{t+1}$.[1] Then the second term in (9) can be written as

$$\min_{1 \leq i \leq m_{t+1}} \min_{\substack{Y \in Dom_i \\ X < Y}} F^i_{t+1}(Y) + P_t(Y - X)$$

such that for a single line segment i the minimum can be written as

$$G^i_t(X) = \min_{\substack{Y \in Dom_i \\ X < Y}} F^i_{t+1}(Y) + P_t(Y - X)$$

where $X \in I_t$. The shape of $G^i_t(X)$ depends on the slopes of F^i_{t+1} and $P_t(x)$. This is illustrated in figures 1 and 2. The lines with slope $-p_t$ represent the lines $F^i_{t+1}(Y) + P_t(Y - X)$ over $Y - c_t \leq X < Y$ for $Y \in Dom_i$. The numbered lines represent the lower envelopes of the thin lines. Four different line types are distinguished: type 1, 2, 3 and 4.

It follows now that for $t = 1, \ldots, T$ and for $i = 1, \ldots, m_t$ if $p_t > \beta_i$

$$G^i_t(X) = \begin{cases} G^{1i}_t(X) = F^i_{t+1}(X) + K_t & \text{for } a_i \leq X < b_i, X \in I_t \\ G^{2i}_t(X) = F^i_{t+1}(a_i) + K_t - p_t(X - a_i) & \text{for } a_i - c_t \leq X < a_i, X \in I_t \end{cases} \tag{10}$$

and if $p_t \leq \beta_i$ then

$$G^i_t(X) = \begin{cases} G^{3i}_t(X) = F^i_{t+1}(b_i) + K_t - p_t(X - b_i) & \text{for } b_i - c_t \leq X < b_i, X \in I_t \\ G^{4i}_t(X) = F^i_{t+1}(X + c_t) + K_t - p_t c_t & \text{for } a_i - c_t \leq X < b_i - c_t, X \in I_t, \end{cases} \tag{11}$$

where we define

$$F^i_{t+1}(b_i) = \lim_{x \uparrow b_i} F^i_{t+1}(x).$$

This means that $G^i_t(X)$ is piecewise linear, which implies that the second term in (9) is piecewise linear. This completes the proof. □

[1] It is not difficult to show by induction that $F_t(X)$ is right continuous.

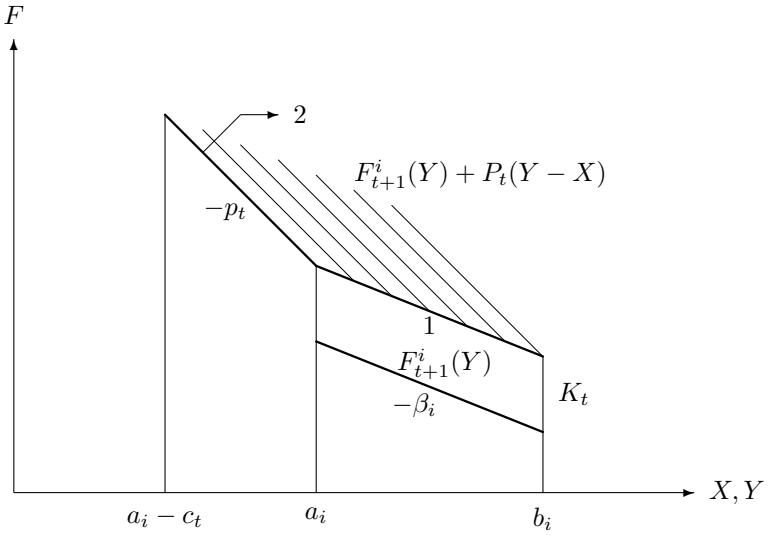

Figure 1: Case $p_t > \beta_i$

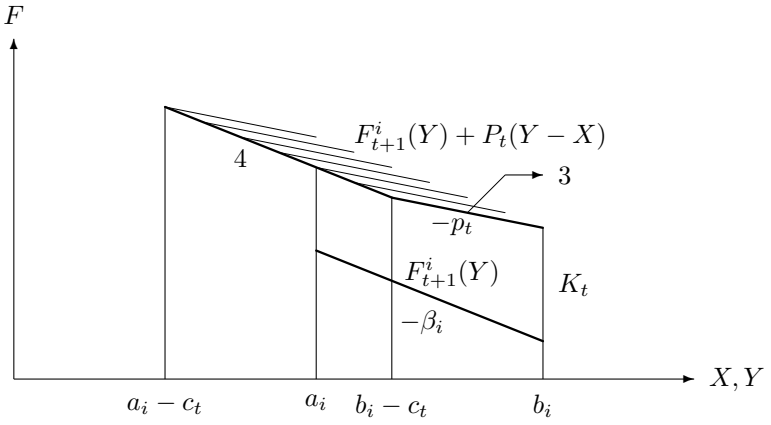

Figure 2: Case $p_t \leq \beta_i$

It follows from the previous paragraphs that the recursion formula consists of updating a piecewise linear function by adding new linear line segments. But not all line segments in (10) and (11) have to be considered. This is shown by propositions 9 and 10 in the Appendix. It follows from these propositions that $G^i_t(X)$ can be simplified to

$$G^i_t(X) = \begin{cases} G^{2i}_t(X) & \text{if } p_t > \beta_i \\ G^{4i}_t(X) & \text{if } p_t \leq \beta_i \text{ and } i = 1, \ldots, m_t - 1 \\ G^{4i}_t(X) \bigcup G^{3i}_t(X) & \text{if } p_t \leq \beta_i \text{ and } i = m_t. \end{cases}$$

In this way the minimum cost function can be constructed recursively. But we also have to keep track of the production quantities corresponding to the line segments. It follows from figures 1 and 2 that for type 2 and 3 line segments production is below capacity and that for type 4 line

segments production is at full capacity. Furthermore, if in some period $t+1$ a line segment i in $F_{t+1}(X)$ is not replaced by some line segment in period t (i.e. $F_{t+1}(X)$ is the minimum in (9)) production will be zero. Therefore we construct a function $Q_t(X)$ which keeps track of the production quantities in each period. For each period t we define

$$Q_t(X) = \begin{cases} -D_T & \text{if line segment } i \text{ is in } F_{t+1} \\ a_i & \text{if line segment } i \text{ is of type 2} \\ b_i & \text{if line segment } i \text{ is of type 3} \\ D_T + c_t & \text{if line segment } i \text{ is of type 4} \end{cases} \tag{12}$$

and X is in the domain of line segment i. The production quantities corresponding to the production lines can then be found as follows. If cumulative production in period $1, \ldots, t-1$ equals $X = \sum_{i=1}^{t-1} x_i$, then production in period t equals

$$x_t = \begin{cases} 0 & \text{if } Q_t(X) < 0 \\ c_t & \text{if } Q_t(X) - X \geq c_t \\ Q_t(X) - X & \text{otherwise.} \end{cases}$$

Note that in the dynamic programming algorithm the line segments of $F_t(X)$ and $Q_t(X)$ have to be stored, but that storage space can be saved by noting that consecutive line segments of $F_t(X)$ with the same value in (12) will require only one segment of $Q_t(X)$.

3 A NI/Z/Z/NI problem that requires exponential running time

Because the DP algorithm solves the G/G/G/G problem (which is an NP hard problem), we expect that there exist problem instances on which the algorithm requires exponential time. Chen et al. (1994) do not carry out a detailed complexity analysis, but they show empirically that their algorithm works reasonably well. In this section we present an NI/Z/Z/NI problem instance that requires at least 2^{T-1} line segments to be solved by the backward algorithm, which implies that the running time is exponential.

Consider a problem instance with

$$\begin{aligned}
d_t &= 2^{T-t-1} && \text{for } t = 1, \ldots, T-1 \\
d_T &= 1 \\
K_t &= 2^{T-t-1} && \text{for } t = 1, \ldots, T-1 \\
K_T &= 1 \\
h_t &= 0 && \text{for } t = 1, \ldots, T \\
p_t &= 0 && \text{for } t = 1, \ldots, T \\
c_1 &= 2^{T-1} \\
c_t &= d_t && \text{for } t = 2, \ldots, T
\end{aligned}$$

so that

$$A_t = \sum_{i=1}^{t-1} d_i = \sum_{i=1}^{t-1} 2^{T-i-1} = 2^{T-1} - 2^{T-t}$$

and

$$B_t = \min\{C_{t-1}, D_T\} = D_T = 2^{T-1}$$

for $t = 2, \ldots, T$, because

$$D_T = \sum_{t=1}^{T} d_t = c_1 \leq \sum_{i=1}^{t-1} c_i = C_{t-1}.$$

This implies that the lengths of the intervals, denoted by L_t, double in each iteration, because

$$L_t = B_t - A_t = 2^{T-t} = 2 \cdot 2^{T-(t+1)} = 2L_{t+1}.$$

For $t = 1$ we have $A_1 = B_1 = 0$.

It may be clear that for $t = T$ the minimum cost function F_T consists of two line segments

$$\begin{aligned}
f_0(x) &= 0 && \text{for } x = D_T = 2^{T-1} \\
f_1(x) &= K_T = 1 && \text{for } 2^{T-1} - 1 \leq x < 2^{T-1},
\end{aligned}$$

because one type 3 line segment is added to F_{T+1}. In the remainder of this section we number the line segments from the right to the left for notational convenience and we start with line segment 0. The following proposition states that F_t consists of $2^{T-t} + 1$ line segments for $t = 2, \ldots, T$.

Proposition 2 *The minimum cost function $F_t(x)$ consists of the line segments*

$$\begin{aligned}
f_0(x) &= 0 && \textit{for } x = 2^{T-1} \\
f_i(x) &= i && \textit{for } 2^{T-1} - i \leq x < 2^{T-1} - i + 1,\ i = 1, \ldots, 2^{T-t}
\end{aligned}$$

for $t = 2, \ldots, T$.

Proof We will prove the proposition by induction. We have already shown that the proposition holds for $t = T$. Assume now that the proposition holds for some $t \leq T$. We will show that the proposition also holds for $t - 1$.

Denote $g_i(x)$ by a line segment created from line segment $f_i(x)$. It follows from section 2 equation (11) that one type 3 and one type 4 line segment are created from line segment 0, so that

$$g_0(x) = K_{t-1} = 2^{T-t} \text{ for } 2^{T-1} - 2^{T-t} \leq x \leq 2^{T-1},$$

because the lines coincide because of equal slopes. This means that $g_0(x) = 2^{T-t} \geq f_i(x)$ for $i = 1, \ldots, 2^{T-t}$ and that $g_0(x)$ will not contribute to $F_{t-1}(x)$.

Now 2^{T-t} type 4 line segments are created from line segments $i = 1, \ldots, 2^{T-t}$ and $g_i(x)$ is defined on $[a'_i, b'_i\rangle = [a_i - c_{t-1}, b_i - c_{t-1}\rangle = [2^{T-1} - i - 2^{T-t}, 2^{T-1} - i + 1 - 2^{T-t}\rangle$, so that

$$g_i(x) = K_{t-1} + i = 2^{T-t} + i \text{ for } a'_i \leq x < b'_i,\ i = 1, \ldots, 2^{T-t}.$$

Note that $[a'_i, b'_i\rangle \cap [a'_{i+1}, b'_{i+1}\rangle = \emptyset$ for $i = 1, \ldots, 2^{T-t} - 1$, $[a'_i, b'_i\rangle \subset [A_{t-1}, B_{t-1}]$ and $[a'_i, b'_i\rangle \cap [A_t, B_t] = \emptyset$ for $i = 1, \ldots, 2^{T-t}$. This means that all line segments $g_i(x)$ will contribute to $F_{t-1}(x)$. Furthermore, line segments $f_i(x)$ for $i = 0, \ldots, 2^{T-t}$ (these are the line segments of $F_t(x)$) remain in the minimum cost function $F_{t-1}(x)$, because $B_t = D_T$. If we now define

$$f_{i+2^{T-t}}(x) = g_i(x) = i + 2^{T-t},\ \text{for } a'_i \leq x < b'_i,\ i = 1, \ldots, 2^{T-1},$$

then

$$\begin{aligned} f_0(x) &= 0 \quad \text{for } x = 2^{T-1} \\ f_i(x) &= i \quad \text{for } 2^{T-1} - i \leq x < 2^{T-1} - i + 1,\ i = 1, \ldots, 2^{T-(t-1)}, \end{aligned}$$

which completes the proof. □

A graphical representation of the proof is shown in figure 3. We see that in the first recursion step F_T consists of $2 = 1 + 2^0$ line segments, in the second recursion step F_{T-1} consists of $3 = 1 + 2^1$ line segments, in the third recursion step F_{T-2} consists of $5 = 1 + 2^2$ line segments and in the fourth recursion step F_{T-3} consists of $9 = 1 + 2^3$ line segments. This means that the number of line segments increases at an exponential rate.

Now the total number of line segments required to solve the above problem by the backward algorithm can be computed.

Theorem 3 *The total number of line segments required in the backward algorithm to solve the NI/Z/Z/NI problem is* $T + 2^{T-1}$.

Proof It follows from proposition 2 that the the minimum cost function consists of $m_t = 2^{T-t} + 1$ line segments in recursion step $t = 2, \ldots, T$. The algorithm starts with one line segment and in recursion step 1 one line segment is needed, because $F_1(x)$ is defined on $[A_1, B_1] = \{0\}$. So the total number of line segments required in the backward algorithm equals

$$\sum_{t=1}^{T+1} m_t = 2 + \sum_{t=2}^{T} (2^{T-t} + 1) = T + 1 + \sum_{t=0}^{T-2} 2^t = T + 2^{T-1},$$

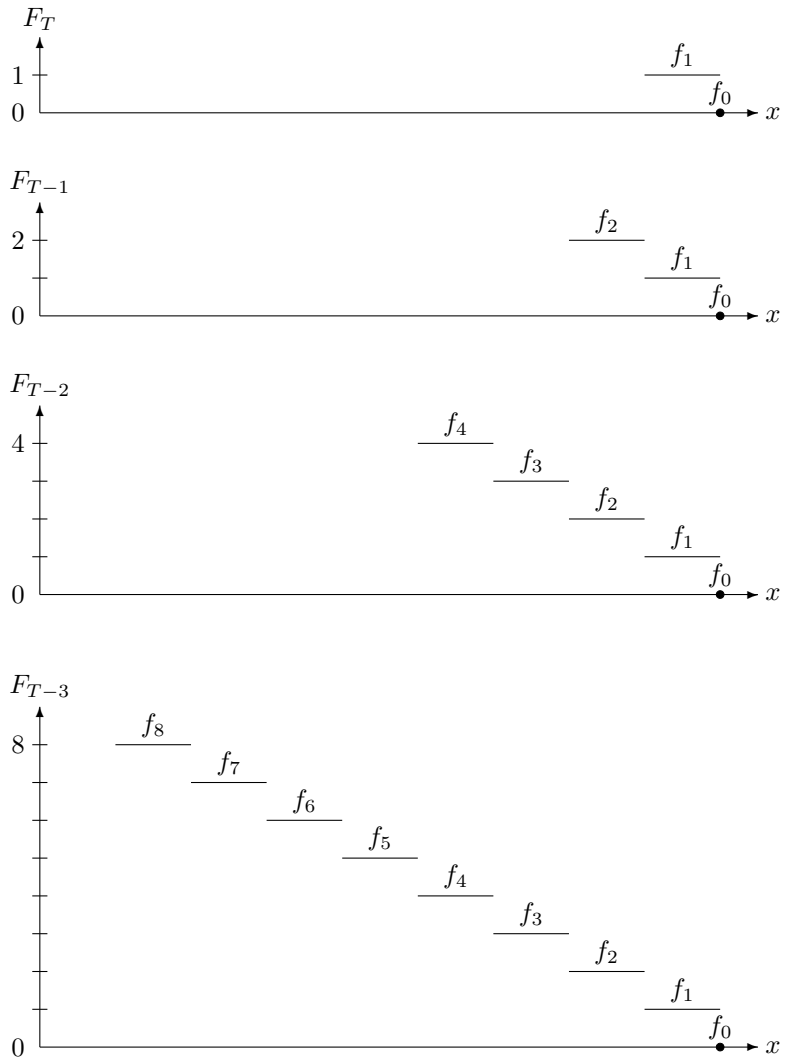

Figure 3: The first 4 recursion steps for the NI/Z/Z/NI problem

which completes the proof. □

Note that the above CLSP is actually an easy problem to solve by hand, because $c_1 = D_T$. It is easy to verify that the optimal solution for the above problem is to produce all demand in the first period. It is also interesting to note that the forward algorithm solves this problem in $O(T)$ time. This means that it does make a difference which algorithm is used to solve a CLSP. However, for the forward algorithm we can also find problem instances that are solved in exponential time.

4 A new O(T^2) algorithm to solve the NI/G/NI/ND problem

4.1 The algorithm

In this section we present a new algorithm to solve the NI/G/NI/ND problem based on the backward algorithm presented in section 2. Bitran and Yanasse (1982) developed an O(T^4) algorithm to solve this problem class and Chung and Lin (1988) improved the time complexity to O(T^2). Our algorithm has the same time complexity, but we can show it requires less operations than Chung and Lin's. It is based on the following geometric argument.

Proposition 4 *In a NI/G/NI/ND problem line segments in $F_t(X)$ are not replaced by new line segments.*

Proof Let $f_i(x)$ be a line segment of $F_t(X)$. Because p_t is non-increasing and $h_t \geq 0$, it follows from (5) that p'_t is non-increasing. Because only type 2 line segments are created in each recursion step (see section 2), line segments created from $f_i(x)$ in recursion step r and s with $r < s < t$ are defined as

$$g_i^r(x) = f_i(a_i) + K_r + p'_r(a_i - x) \text{ for } a_i - c_r \leq x < a_i$$
$$g_i^s(x) = f_i(a_i) + K_s + p'_s(a_i - x) \text{ for } a_i - c_s \leq x < a_i.$$

Note that the domain of g^r is a subset of g^s because $c_r \leq c_s$ and $g^r(x) \geq g^s(x)$ because $K_r \geq K_s$ and $p'_r \geq p'_s$. This shows that an existing line segment will not be replaced by a line segment created in a previous iteration.

It remains to show that a line segment created from $g_j^t(x)$ will not replace $g_i^t(x)$, where $g_k^t(x)$ denotes a line segment created in iteration t with domain $[a_k, b_k\rangle$. We will prove this by induction. Assume that an existing line segment will not be replaced. It is clear that the assumption holds for $t = T$. Furthermore, we may assume without loss of generality that $b_i \leq a_j$. Note that it is sufficient to show that the line segment created from $g_j^t(x)$ lies above $g_i^t(b_i)$, because p_t is non-increasing. Assume that line segment $g_i^t(x)$ is created from some line segment $f_k(x)$ in iteration t. Because existing line segments are not replaced by the induction assumption, it must hold that

$$g_j^t(a_j) \geq f_k(a_k) + (a_k - a_j)p_t.$$

Furthermore, because line segment $g_i^t(x)$ is created from line segment $f_k(x)$

$$g_i^t(b_i) = f_k(a_k) + K_t + (a_k - b_i)p_t.$$

But the line segment created from $g_j^t(x)$ in iteration $t' < t$ is defined as

$$h_j^{t'}(x) = g_j^t(a_j) + K_{t'} + (a_j - x)p_{t'},$$

so that

$$\begin{aligned} h_j^{t'}(b_i) &= g_j^t(a_j) + K_{t'} + (a_j - b_i)p_{t'} \\ &\geq f_k(a_k) + (a_k - a_j)p_t + K_{t'} + (a_j - b_i)p_{t'} \\ &= g_i^t(b_i) - K_t - (a_k - b_i)p_t + (a_k - a_j)p_t + K_{t'} + (a_j - b_i)p_{t'} \\ &= g_i^t(b_i) + (K_{t'} - K_t) + (a_j - b_i)(p_{t'} - p_t) \\ &\geq g_i^t(b_i). \end{aligned}$$

This means that line segment $g_i^t(x)$ will not be replaced by $g_j^t(x)$. □

Proposition 4 implies that only line segments are added to the left of F_{t+1} in each recursion step (namely in the interval $[A_t, B_t] \subset [A_t, A_{t+1}]$) in the case of a NI/G/NI/ND problem. Using this property we can reduce the length of the intervals. This is the key point of the algorithm. Because line segments are added to the left of an existing line segment and the domain length of the newly created lines is at most c_t, only line segments created from lines in the domain $[A_{t+1}, A_{t+1} + c_t]$ will contribute to F_t in recursion step t. This implies that line segments in the interval $[A_{t+1} + c_t, B_{t+1}]$ do not have to be considered and that B_t can be changed into $B'_t = \min\{A_{t+1}, D_T, C_{t-1}\} = \min\{D_t, C_{t-1}\}$ so that $L_t = B'_t - A_t \leq d_t \leq c_t$. This reduces the domain length of F_t considerably.

Note that for this algorithm we do not have to store the slopes of the line segments. Because we know that only type 2 line segments are created, we only have to keep track of the points $f_i(a_i)$ for $i = 1, \ldots, m_t$ in recursion step t. This will save storage space and running time. We also used the line segments of $F_t(X)$ to determine the optimal production quantities and not the back-tracking method presented in section 2.2. The former back-tracking method saves some storage space, but by the observed linear behavior of storage space in the new algorithm, it is not necessary to use this method.

4.2 Numerical example

In this section we give a numerical example of our algorithm. Consider the following 4-period example which has been taken from Chung and Lin (1988):

$$\begin{aligned} d &= (30, 40, 70, 30) \\ c &= (60, 70, 80, 90) \\ K &= (50, 40, 30, 20) \\ p &= (5, 5, 5, 4) \\ h &= (1, 2, 3, 4). \end{aligned}$$

After rewriting this problem as a problem without holding cost (see section 2.1), we have that

$$p' = (14, 14, 12, 8), D = (30, 70, 140, 170) \text{ and } \sum_{t=1}^{T} h_t D_t = 1270.$$

So after solving the problem with the new production cost, we have to subtract 1270 from the optimal value to obtain the minimal cost for the original problem.

A graphical representation of our algorithm applied to this problem instance is shown in figure 4. In the first iteration ($t = 4$) we start with one line piece. The setup cost of 20 plus the production of 30 items ($x_4 = 30$) at a cost of 8 leads to a total cost of 260. In the second iteration ($t = 3$) line piece (2) created from line piece (1) covers the whole interval $[70, 140]$. However, there is another line piece which may contribute to (part) of this interval. This is line (3) which corresponds to full production in period 3. (Note that this line is created from the point (170,0).) We see that this line does not contribute to the minimum cost function. So production of 50 units in period 3 ($x_3 = 50$) and 30 units in period 4 ($x_4 = 30$) with total cost 890 is preferred over full production in period 3 ($x_3 = 80$), which has cost $30 + 80 \cdot 12 = 990$.

In iteration three ($t = 2$) only line (4), which is created from line (3), contributes to the interval [30,70]. The line created from line (1) is not considered, because it can not contribute to this interval since production capacity is too small. In the last iteration the same happens and line (5) (created from line (4)) is the only line which contributes to the minimum cost function. Now the total costs equal 2230 corresponding to production plan $x = (30, 40, 70, 30)$. So the costs of the original problem equal $2230 - 1270 = 960$, which is equal to the solution in Chung and Lin (1988).

4.3 Time complexity of the algorithm

In this section we examine the time complexity of the algorithm. Note that in each recursion step we have to create a line segment from an existing line segment, we have to check if this line segment will contribute to F_t and we have to add it to F_t if necessary. The above steps will require constant time, so by determining the total number of line segments considered in each recursion step we can find the time complexity of the algorithm. Note that for the algorithm in section 2 which solves G/G/G/G problems, we have to search where a new line segment is added and/or if the line intersects other lines of F_t. For the NI/G/NI/ND problem, we know exactly that a new line segment is added to the left and that it does not intersect other lines. Therefore, adding a line segment requires constant time in our algorithm. Define

$$q = \min\{t : \sum_{i=T-t}^{T} d_i \geq c_{T-t}\}.$$

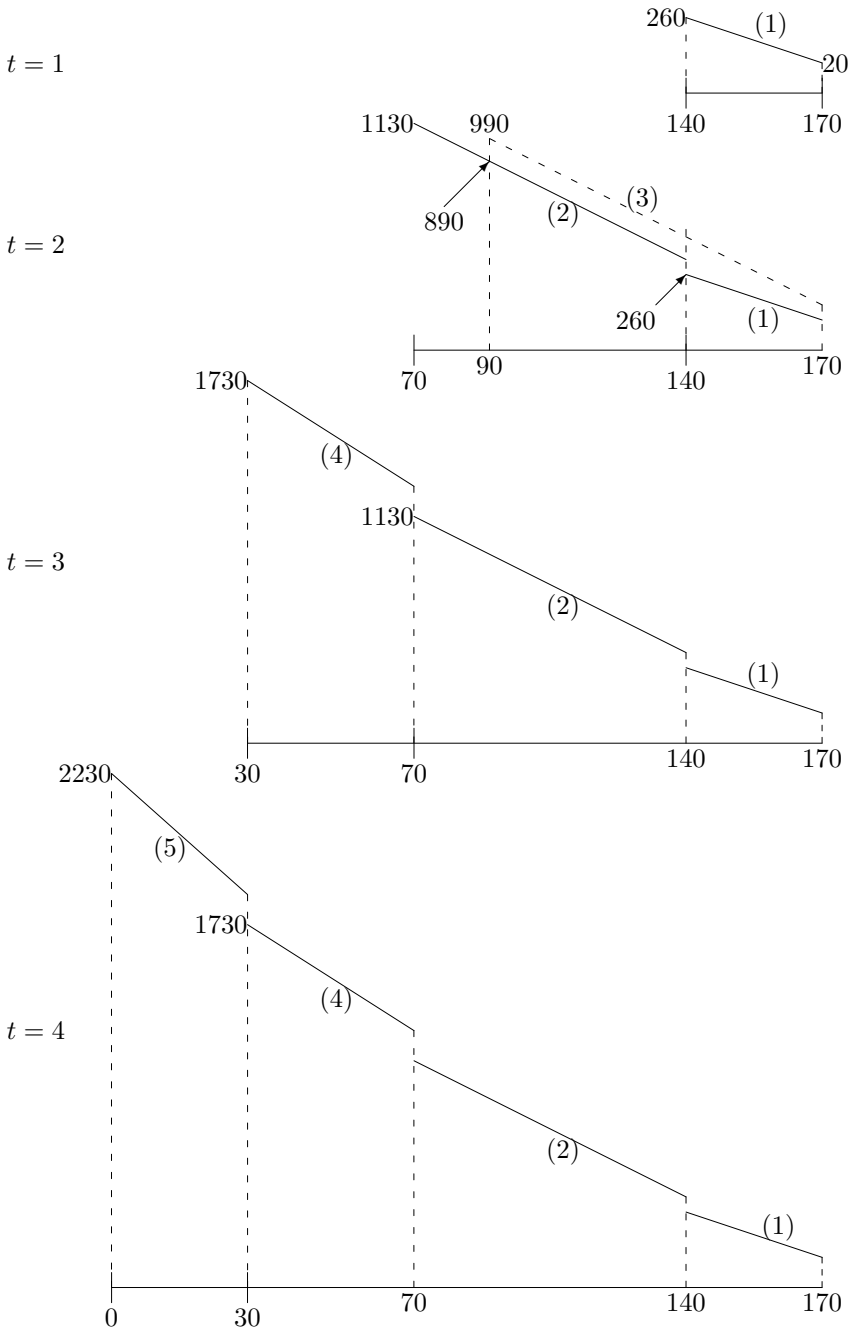

Figure 4: Numerical example

This means that q is the smallest number for which the sum of last $q+1$ demands exceeds capacity in period $T-q$. This implies for $t = T-q+1, \ldots, T$ that

$$\sum_{i=t}^{T} d_i < c_t.$$

We use the following two lemmas to derive the complexity of the algorithm.

Lemma 5 *In the first q iterations the number of line segments to be considered will increase by at most one in each iteration.*

Proof First note that in iterations $t = T - q + 1, \ldots, T$ (these are the first q iterations) $B_t = \min\{D_t, C_{t-1}\} \leq D_T$. This means that in the first q iterations $[A_t, B_t] \subset [D_{t-1}, D_T]$. Now in recursion step t the new line segments are created from line segments in the interval $[D_t, D_T]$ and new line segments end up in the interval $[D_{t-1}, D_t]$, because line segments are added to the left of F_{t+1}. Assume there is some line segment i in F_{t+1} defined on $[a_i, b_i\rangle$ for $D_t \leq a_i < b_i \leq D_T$. Now the type 2 line segment created from this line is defined on $[a_i - c_t, a_i\rangle$. Note that $a_i - c_t < D_T - c_t < D_T - \sum_{i=t}^{T} d_i = D_{t-1}$ and $a_i \geq D_t$ so that $[D_{t-1}, D_t] \subset [a_i - c_t, a_i\rangle$ for all lines created from line $i = 1, \ldots, m_{t+1}$. Because all new type 2 line segments have the same slope, the lines do not intersect, which implies that exactly one line segment is added to F_{t+1}. If $d_t = 0$ then the interval is not extended to the left and the number of line segments does not increase. This means that at most one new line segment is added to the existing minimum cost function F_{t+1} and the number of line segments to be considered increases at most by one in each iteration, which completes the proof. □

Lemma 6 *In the last $T - q$ iterations the number of line segments to be considered will increase by at most one in each iteration.*

Proof We show that the number of line segments to be considered in iteration t and $t - 1$ does not differ more than one for $t = 1, \ldots, T - q$. Note that it is sufficient to show that the number of line segments in the interval $[A_t, A_t + c_{t-1}] = [D_{t-1}, D_{t-1} + c_{t-1}]$ is at most one more than the number of line segments in the interval $[A_{t+1}, A_{t+1} + c_t] = [D_t, D_t + c_t]$.

Define $s = D_t + c_t - d_t$ and note that $s \in [D_t, D_t + c_t]$ because $d_t \leq c_t$. Furthermore, line segments with $a_i \leq s$ are denoted by $1, \ldots, n_s$ and line segments with $a_i > s$ are denoted by $n_s + 1, \ldots, m_t$ b. A type 2 line segment created from some line segment $i \in \{1, \ldots, n_s\}$ is defined on $[a_i - c_t, a_i\rangle$ with $a_i - c_t \leq s - c_t = D_t - d_t = D_{t-1}$ and $a_i \geq D_t$, so that $[D_{t-1}, D_t] \subset [a_i - c_t, a_i\rangle$. So if there is some new line segment created from some line segment $i = 1, \ldots, n_s$, exactly one line segment is added to the left of F_t, because the slopes are equal. Note that here we use the same argument as in lemma 5.

Now assume that some line segment is added to F_t, which is created from some line segment $i \in \{n_s + 1, \ldots, m_t\}$. This line segment is defined on $[a_i - c_t, a_i\rangle$ with $a_i - c_t > D_t - d_t = D_{t-1}$ and $a_i > D_t$. This means that the interval $[D_{t-1}, D_t]$ can not be totally covered by line segments created from line segments $i = n_s + 1, \ldots, m_t$ and at most $m_t - n_s$ line segments are created from those lines. This also implies that there must be a line segment created from lines i with

$1 \leq i \leq n_s$ which contributes to F_t. But it follows from the previous paragraph that exactly one line segment is created from these line segments. This implies that the number of line segments added to the left of F_t will not exceed $m_t - n_s + 1$.

Now we look at the line segments that are considered in iteration $t-1$. These are the line segments in the interval $[A_t, A_t + c_{t-1}] = [D_{t-1}, D_{t-1} + c_{t-1}] \subset [D_{t-1}, D_{t-1} + c_t = [D_{t-1}, s]$, because $c_{t-1} \leq c_t$. But line segments $1, \ldots, n_s$ lie in the interval $[D_{t-1}, s]$. This means that the total number of line segments considered in iteration $t-1$ equals at most $(m_t - n_s + 1) + n_s = m_t + 1$, which completes the proof. □

A visual representation of the proof is shown in figure 5. The horizontal lines represent the

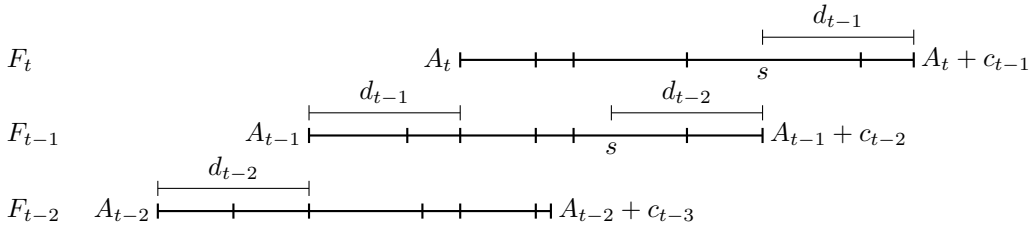

Figure 5: Visual representation of Lemma 6

intervals on which F_t is defined, so F_t consists of five line segments. In recursion step $t-1$ a part of length d_{t-1} (consisting of two line segments) is 'copied' from F_t to F_{t-1}. Note that in the worst case at most two line segments are added to the left of F_t in iteration $t-1$. We also see that the number of line segments to be considered in iteration $t-1$ has increased by one.

In the next iteration we see that a part consisting of two line segments is copied from F_{t-1} to F_{t-2}. Because $c_{t-3} < c_{t-2}$, the interval to be considered in iteration $t-3$ ($[A_{t-2}, A_{t-2} + c_{t-3}]$) has decreased relative to the interval in iteration $t-2$ ($[A_{t-1}, A_{t-1} + c_{t-2}]$). In this example the number of line segments to be considered in those two iterations is equal. So if c_t is increasing, it is more likely that the number of line segments will not increase. Note that if $c_{t-3} = c_{t-2}$, the number of line segments could have increased by one again.

Now we can state our main theorem.

Theorem 7 *The time complexity of the algorithm for solving a NI/G/NI/ND problem is O(T^2).*

Proof By the previous lemmas it follows that the number of line segments considered in each recursion step increases by at most one. This means that $m_t \leq m_{t+1} + 1$ for $t = 1, \ldots, T-1$. Furthermore, we know that $m_T = 1$, so that $m_t \leq T - t + 1$ for $t = 1, \ldots, T$. Summing up all line

segments yields

$$\sum_{t=1}^{T} m_t \leq \sum_{t=1}^{T} T - t + 1 = \sum_{t=1}^{T} t = \frac{1}{2}T(T+1),$$

which implies that the time complexity of the algorithm is $\mathrm{O}(T^2)$. □

We can also prove that this result cannot be improved upon. To this end we show that the following problem instance requires a quadratic number of line segments to be solved. Let

$$\begin{aligned}
&c_t = C = 2T \\
&d_1 = 1 \\
&d_t = 2T - 1 = C - 1 \qquad \text{for } t = 2, \ldots, T-1 \\
&d_T = T = \tfrac{1}{2}C \\
&K_t = T - t + 1 \qquad \text{for } t = 1, \ldots, T \\
&p_t = 0 \\
&h_t = 0.
\end{aligned}$$

We will briefly describe why this problem instance needs a quadratic number of line segments. In the first iteration ($t = T$) a line segment with domain length $T = \frac{1}{2}C$ is created. Furthermore, it can be shown that $A_T = D_{t-1}$ and $B_t = \min\{A_{t+1}, C_{t-1}\} = A_{t+1} = D_t$ for $t = 2, \ldots, T-1$, so that the interval $[D_t, D_t + c_t]$ is considered in iteration t. Because demand is almost equal to capacity in the following iterations, the point s is forced to be in the domain of the left most line segment, so that all line segments of the previous iteration are potential candidates for the minimum cost function. Because K_t is strictly decreasing all candidates are added to the minimum cost function and the number of line segments increases by one in each iteration. The minimum cost function F_t looks like a step function as in section 3.

If we denote the number of line segments created in iteration t by n_t, then it can be shown that $n_{T+1} = 1$, $n_T = 1$ and $n_t = n_{t+1} + 1 = T - t + 1$ for $t = 2, \ldots, T-1$. In the last iteration ($t = 1$) only one line segment is added, because $[A_1, B_1] = \{0\}$ so that $n_1 = 1$. This is the maximum number of line segments which can be created in each recursion step. Now the total number of line segments equals

$$\sum_{t=1}^{T+1} n_t = 2 + \sum_{t=2}^{T} (T - t + 1) = 2 + \sum_{t=1}^{T-1} t = \frac{1}{2}T(T-1) + 2,$$

which is clearly of quadratic order.

4.4 Computational tests

We have tested our new algorithm empirically using the same problem instances as in Chen et al. (1994). That is, the demand pattern is generated by the formula

$$d_t = \mu + \sigma x_t + a \sin\left[\frac{2\pi}{c}(t + c/2)\right]$$

for $t = 1, \ldots, T$, where

μ = average demand

σ = standard deviation of demand

x_t = i.d.d. standard normal random variable

a = amplitude of the seasonal component

c = cycle length of the seasonal component.

If demand is negative for some period, demand is set to zero and only feasible problem instances are generated.

Four different types of demand are generated: (1) $\sigma = 67$, $a = 0$, (2) $\sigma = 237$, $a = 0$, (3) $\sigma = 67$, $a = 125$, $c = T$, (4) $\sigma = 67$, $a = 125$, $c = 12$ and for each type of demand $\mu = 200$. For each type of demand 5 problem instances are generated, so that we have 20 test instances for some parameter setting of K_t, c_t, p_t and h_t. We set $K_t = K = 100$, 900 and 3600, $h_t = h = 1$, $p_t = p = 0$ and $c_t = C = 250$, 700 and 1200 and the time horizon T is set to 96, 192, 384 and 768 periods. So in total we generated 720 test problems.

We do not present running times of the new algorithm, because all problem instances are solved within less than 0.02 seconds. Table 1 shows the total number of line segments used in the algorithm (worst case in parentheses). Note that the number of line segments determines the running time (see also Chen et al. (1994)). If the total number of line segments increase in linear way, then the running time will also increase in a linear way. We observe that the behavior of the total number of line segments is almost linear relative to T. In particular for small values of K this linear behavior is observable. In the paper of Chen et al. (1994) a quadratic behavior is observed for the same problem instances. This is not surprising, because their algorithm solves more general instances of the CLSP. The reason why their algorithm is slower than ours for NI/G/NI/ND problem instances, is that larger intervals are considered as already mentioned in section 4.1. Furthermore, we observe that the total number of line segments is considerably smaller than the worst possible number of line segments $\frac{1}{2}T(T-1)+2$.

We also implemented the algorithm proposed by Chung and Lin (1988) and we found that their algorithm also exhibits a quadratic behavior. We will now show why this quadratic behavior is observed. To this end we will explain the main ideas of their algorithm first. In their paper a subplan s_{uv} $(1 \leq u \leq v \leq T)$ is defined as the portion of the solution that covers period u trough v. Here periods $u-1$ and v are two consecutive regeneration points, i.e., $Inv_{u-1} = Inv_v = 0$. Because

Table 1: Total number of line segments

K	C	$T = 96$	$T = 192$	$T = 384$	$T = 768$
$\frac{1}{2}T(T-1)+2$		4562	18338	73538	294530
100	250	98	196	388	780
		(103)	(206)	(401)	(808)
	700	91	182	361	721
		(97)	(193)	(385)	(769)
	1200	91	182	360	721
		(97)	(193)	(385)	(768)
900	250	231	455	914	1897
		(416)	(744)	(1424)	(3178)
	700	116	240	480	960
		(127)	(262)	(515)	(992)
	1200	91	181	362	724
		(97)	(193)	(385)	(769)
3600	250	467	1036	2263	4672
		(1059)	(2145)	(4582)	(9319)
	700	269	558	1158	2342
		(294)	(621)	(1315)	(2490)
	1200	135	271	547	1089
		(145)	(291)	(589)	(1150)

a NI/G/NI/ND problem satisfies the property $Inv_{t-1}x_t(x_t - C_t) = 0$ for all t, it must hold that in each period of a subplan s_{uv} there is full or zero production except for period u.

Now Chung and Lin (1988) show that the following subplans are candidates for the optimal solution. A subplan $s_{ut} = (i_1, \ldots, i_k)$ with $1 \leq u = i_1 < i_2 < \cdots < i_k \leq t$ is a candidate subplan for demand in periods $u, \ldots, t$ if

$$i_n = \max\{i : i < i_{n+1} \text{ and } c_i < M_n - D_{i-1}\},$$

where $M_n = D_t - (c_{i_{n+1}} + \ldots + c_{i_k})$ and $i_{k+1} = t+1$. So i_n is the largest period for which demand in periods $i_n, \ldots, t$ cannot be satisfied by $c_{i_n}, \ldots, c_{i_k}$. This demand will be satisfied by a previous production period. If we denote the set of candidate subplans s_{ut} with $1 \leq u \leq t$ by V_t, the cost associated with candidate subplan s_{ut} by $C(s_{ut})$ and the minimal cost for period 1 through t by $f(t)$, then

$$f(t) = \min_{u=1,\ldots,t}\{f(u-1) + C(s_{ut}) : s_{ut} \in V_t\},$$

with $f(0) = 0$.

Chung and Lin (1988) show that all candidate subplans in V_t can be constructed in O(t) time starting in period t and ending in period 1. Candidate subplans are constructed by building on previous constructed ones. If we look at the numerical example, there are two candidate subplans

for $t = 4$: $s_{44} = (4)$ (i.e., $x_4 = 30$) and $s_{24} = (2, 3)$ (i.e., $x_2 = 60$ and $x_3 = 80$). In the example s_{44} has been constructed in the first iteration (line (1) in figure 4). Whereas Chung and Lin's algorithm also finds s_{24}, our algorithm detects in an early stage that this is not an optimal candidate subplan. Namely, in iteration 2 of our algorithm we find that line (3) does not contribute to the minimum cost function and this is exactly the line that corresponds to the full production in period 3. In this way some candidate subplans are eliminated in our algorithm, whereas Chung and Lin's algorithm does find these non-optimal candidate subplans.

To summarize the algorithm of Chung and Lin (1988) exhibits a quadratic behavior, because in each period t they search for candidate subplans s_{ut} for all values of $u = 1, \ldots, t$, whereas our algorithm eliminates some candidate subplans in an early stage. This means that both algorithms have the same time complexity, but our algorithm performs better empirically.

5 Conclusion

In this paper we have presented a new $O(T^2)$ algorithm to solve the capacitated lot-sizing problem (CLSP) with non-increasing setup cost, general holding cost, non-increasing production cost and non-decreasing capacities. Our algorithm is based on the algorithm proposed by Chen et al. (1994) which solves general cases of the CLSP. Chung and Lin (1988) also proposed an $O(T^2)$ to solve this problem, but we can show that our algorithm performs at least equally well. In fact, numerical tests show that our algorithm has a linear running time, whereas we can empirically show that Chung and Lin's algorithm behaves quadratically.

A Theorems and propositions

Theorem 8 *For $t = 1, \ldots, T$ and $X \in I_t$, $F_t(X)$ is a non-increasing function.*

Proof Assume some cumulative production levels $X, X' \in I_t$ with $X < X'$. Let $F_t(X)$ the minimum cost corresponding to production level X and let $\delta = X' - X$. Let $x_t^*, \ldots, x_T^*$ the optimal production plan corresponding to cumulative production before t equal to X. From this production plan we construct a production plan corresponding to cumulative production before t equal to X' in the following way. Let $m = \{\min s : \sum_{i=t}^{s} x_t^* - \delta \geq 0$ and let the new production program equal $0, \ldots, 0, \sum_{i=t}^{m} x_i^* - \delta, x_{m+1}^*, \ldots, x_T^*$. This is a feasible production plan because starting inventory in period t (equal to δ) is large enough to cover demand in period $t, \ldots, m-1$ and $\sum_{i=t}^{m} x_i^* - \delta < x_m^* \leq c_m$. But this production plan has less production costs because P_t is non-decreasing, which implies that $F_t(X) \geq F_t(X')$. This completes the proof. □

Note that this theorem holds for all non-decreasing production functions P_t.

Proposition 9 *Line segment* $G_t^{1i}(X)$ *will not contribute to* $F_t^i(X)$ *for* $t = 1, \ldots, T$ *and* $i = 1, \ldots, m_t$.

Proof This is easily seen by the fact that

$$G_t^{1i}(X) = F_{t+1}^i(X) + K_t \geq F_{t+1}^i(X),$$

which means that $G_t^{1i}(X)$ will not contribute to $F_t^i(X)$ according to (9). □

Proposition 10 *Line segment* $G_t^{3i}(X)$ *will not contribute to* $F_t^i(X)$ *for* $t = 1, \ldots, T$ *and* $i = 1, \ldots, m_t - 1$.

Proof This can be seen by the fact that

$$\begin{aligned} G_t^{3i}(X) &= F_{t+1}^i(b_i) + K_t - p_t(X - b_i) \\ &\geq F_{t+1}^i(a_{i+1}) + K_t - p_t(X - a_{i+1}) \text{ (because } F_{t+1}^i(X) \text{ is non-increasing)} \\ &= G_t^{2,i+1}(X) \geq F_t^i(X). \end{aligned}$$

This means that for $i = 1, \ldots, m_t - 1$, $G_t^{3i}(X)$ will never contribute to $F_t^i(X)$, which completes the proof. □

B Pseudocode

Input:

A NI/G/NI/ND CLSP instance with $d_t \leq c_t$ and $h_t = 0$

Initialize:

$A_{T+1} := D_T$

Add$(D_T, 0)$

Algorithm:

For $t = T, \ldots, 1$ do

Set the domain:

$B_t := \min\{A_{t+1}, C_{t-1}\}$

$A_t := D_{t-1}$

Line segment that covers the interval $[A_t, B_t]$:

$j := \arg\min_{k=1,\ldots,m_{t+1}}\{f(a_k) + S_t + p_t(a_k - A_t) | a_k - c_t \leq A_t\}$

Add$(A_t, f(a_j) + S_t + p_t(a_j - A_t))$

Line segments that cover part of the interval $[A_t, B_t]$:

For $k = \{1, \ldots, m_{t+1} | a_k - c_t < B_t\}$ do

If $f(a_k) + S_t + p_t(a_k - B_t) < f(a_j) - p_t(B_t - a_j)$ then

Add$(a_k - c_t, f(a_k) + S_t + p_t c_t)$

j:=k

End if

End for

End for

Optimal solution:

$z^* = f(0)$

This function adds a line segment to the minimum cost function:

Function Add(a, f)

$i := i + 1$

$a_i := a$

$f(a_i) := f$

End function

References

Aggarwal, A., and Park, J.K., 1993. Improved algorithms for economic lot-size problems. Operations Research 14, 549–571.

Bitran, G.R., Yanasse, H.H., 1982. Computational complexity of the capacitated lot size problem. Management Science 28, 1174–1186.

Chen, H.-D., Hearn, D.W., Lee, C.-Y., 1994. A new dynamic programming algorithm for the single item capacitated lot size model. Journal of Global Optimization 4, 285–300.

Chung, C.S., Lin, C.H.M., 1988. An O(T^2) algorithm for the NI/G/NI/ND capacitated lot size problem. Management Science 34, 420–426.

Federgruen, A., Tzur, M., 1991. A simple forward algorithm to solve general dynamic lot sizing models with n periods in O($n \log n$) or O(n) time. Management Science 37, 909–925.

Florian, M., Klein, M., 1971. Deterministic production planning with concave costs and capacity constraints. Management Science 18, 12–20.

Florian, M., Lenstra, J.K., Rinnooy Kan, A.H.G., 1980. Deterministic production planning algorithm and complexity. Management Science 26, 669–679.

Shaw, D.X., Wagelmans, A.P.M., 1998. An algorithm for single-item capacitated economic lot sizing with piecewise linear production costs and general holding costs. Management Science 44, 831–838.

Van Hoesel, C.P.M., Wagelmans, A.P.M., 1996. An O(T^3) algorithm for the economic lot-sizing problem with constant capacities. Management Science 42, 142–150.

Van Hoesel, C.P.M., Wagelmans, A.P.M., Moerman, B., 1994. Using geometric techniques to improve dynamic programming algorithms for the economic lot-sizing problem. European Journal of Operational Research 75, 312–331.

Wagelmans, A.P.M., Van Hoesel, C.P.M., Kolen, A., 1992. Economic lot sizing: An O($n \log n$) algorithm that runs in linear time in the Wagner Whitin case. Operations Research 40, 8145–8156.

Wagner, H.M., Whitin, T.M., 1958. Dynamic version of the economic lot size model. Management Science 5, 89–96.