Algorithms for unbounded and varied capacitated lot-sizing problems with outsourcing

Ping Zhan *

Abstract

Lot-sizing problems have been extensively researched for more than half a century ([1]). There are relatively small number of papers on lot-sizing models with outsourcing, despite its important applications in operations research. Recently several papers related to outsourcing models are published ([11, 2, 9, 10, 8]). When there is no bound on production capacity, linear algorithm (totally square) is possible. Period varying capacitated lot-sizing model is known as NP-hard even outsourcing is not allowed. In this manuscript, we treat these two extreme cases, we give an efficient algorithm for former case, and propose a pseudo-polynomial scheme for period varying production capacities.

Keywords: Unbounded and period varying capacitated lot-sizing models, Outsourcing, Dynamic programming

1 Introduction

Outsourcing, as an alternative choice, becomes more and more important in lots of field, because of high speed change of environment or demands, limits of production capacities, concentrating resources on more competing productions and services. Motivated by these, outsourcing is added to classic lot-sizing model with constant production capacity, dynamic programming algorithm ([9, 11, 8]) and greedy algorithm ([10]) were also proposed. A possible application of expensive products related to lot-sizing model with outsourcing is mentioned in [2], a wide range of algorithms to solve lot-sizing models are summarized also there.

Lot-sizing problems as typical mixed-integer programming, the reformulation as compact linear form is a challenge. In last decade, various reformulations of single item lot-sizing sets have been successful achieved ([6, 5, 4, 3]).

The basic form of single-item lot-sizing model with outsourcing ($X^{LS} - C - O$) is formulated as:
\begin{equation}
\min \sum_{t=1}^{n} (p'_t x_t + g'_t z_t + h'_t s_t + q_t y_t) + h_0 s_0
\end{equation}

\begin{align}
& s_{t-1} + x_t + z_t = d_t + s_t \quad \text{for } 1 \leq t \leq n \\
& x_t \leq c_t y_t \quad \text{for } 1 \leq t \leq n \\
& x_t, s_t, z_t \in R_+, \ y_t \in \{0, 1\} \quad \text{for } 1 \leq t \leq n,
\end{align}

where, \(q_t\) is production set up cost, \(p'_t\), \(h'_t\) (including \(h'_0\)) and \(g'_t\) are unit production, holding and outsourcing costs respectively, \(d_t\) is demand and \(c_t\) is the production capacity in period \(t \ (1 \leq t \leq n)\). The variables are production \(x_t\), stock \(s_t\) (including \(s_0\)), outsourcing \(z_t\) and setup \(y_t\) respectively in period \(t \ (1 \leq t \leq n)\). If production capacity is constant, i.e., \(c_t = C\) for all \(t\), we denote such problem as \((X^{LS-CC-O})\). And if production capacity is unbounded, we express it as \((X^{LS-U-O})\).

Since the algorithms related to \((X^{LS-CC-O})\) and its variation has been researched in [9, 10, 11, 8], we treat two other cases, \((X^{LS-U-O})\) and \((X^{LS-C-O})\) here.

2 Unbounded Production Capacity

We begin with simple case. Production capacity is larger than cumulative demands, or unbounded, i.e., \(X^{LS-U-O}\). The structure of extreme solutions is shown in Figure 1.

![Figure 1. Extreme solution of X^{LS-U-O}](image)

Note the structure of solutions is the result of minimum cost flow problems ([6]), more precisely, demand at any period is not met partial by stock, partial by production or outsourcing. The similar property is a base for many algorithms related to lot-sizing problems.

We are now ready for DP (Dynamic Programming). Note, by the flow balance equalities in constraint, stock variables, or production variables can be canceled (or omitted) ([9]).

Let \(G(t)\) be the minimum cost of solving the problem over the first \(t\) periods, and let \(\phi (k, t)\) be the minimum cost of solving the problem over the first \(t\) periods subject to the additional condition that the last production period is \(k\) for some \(k \leq t\). From the definition we have:

\[ G(t) = \min_{k:k \leq t} \phi (k, t). \]

By extreme optimal solution structure shown in Figure 1, no edge between nodes \(k-1\) and \(k\) exists \((s_{k-1} = 0)\), hence separation principle of DP is satisfied and \(\phi (k, t)\) can be calculated as
the best that one we can hope for. The algorithm proposed in this section is based on the
A forward DP recursion for \( X^{LS-U-O} \)
\[
G(0) = 0 \\
G(t) = \min_{k: k \leq t} \left\{ G(k - 1) + \min [q_k + p_k d_{kt}, g_k d_{kt}] \right\}, \quad \text{for } t = 1, \ldots, n.
\]
Every recursion can be carried out in \( O(n) \), and total time complexity for \( X^{LS-U-O} \) is \( O(n^2) \). The complexity is same as the one of unbounded lot-sizing problem without outsourcing.

3 General Capacity Models

In this section, we treat the model \( X^{LS-C-O} \), i.e., general production capacity. It is \( NP \)-hard even in some special cases ([5]). Therefore, a pseudo-polynomial complexity is the best that one we can hope for. The algorithm proposed in this section is based on the model without outsourcing \( X^{LS-C} \), and its extension with backlogging \( X^{LS-C-B} \) in [7], here production capacities \( c_k \) and demands \( d_k \) are non-negative integers.

For any period \( k \) and stock level \( s \in \{0, 1, 2, \ldots, d_k\} \), we denote \( F_k(s) \) as the minimum cost incurred in period \( k \) to \( n \), when the starting stock in period \( k \) is equal to \( s \). Since outsourcing is unbounded, the demands can always been satisfied. Therefore \( F_k(s) \) is feasible.

By denition of \( F_k(s) \), we have the following backward recursive formulas:
\[
F_k(s) = \min_{1 \leq x_k \leq c_k, z_k \geq 0} \left\{ q_k + p_k x_k + g_k z_k + h_k(s - d_k + x_k + z_k) \right\}
\]
\[
+ F_{k+1}(s - d_k + x_k + z_k) \right\} (3)
\]
To simplify notation, let
\[
G_k(s) = \min_{1 \leq x_k \leq c_k, z_k \geq 0} \left\{ q_k + p_k x_k + g_k z_k + h_k(s - d_k + x_k + z_k) + F_{k+1}(s - d_k + x_k + z_k) \right\}
\]
We can now specifically give,
\[
F_k(s) = \begin{cases} 
G_k(s), & s = 0, 1, 2, \ldots, d_k - 1, \\
\min \{h_k(s - d_k) + F_{k+1}(s - d_k), G_k(s)\}, & s = d_k, d_k + 1, \ldots, D_k - 1 \\
h_k(d_{kn}) + F_{k+1}(d_{kn}), & s = D_k.
\end{cases}
\]
(4)
It is reasonable to make assumption
\[
g_k \geq p_k.
\]
(5)
Then outsourcing occurs if and only if no production or production at full capacities.

\[
G_k(s) = \min_{0 \leq z_k} \min_{0 \leq x_k \leq c_k} \min_{0 \leq z_k} \left( g_k z_k + h_k (s - d_k + z_k) + F_{k+1} (s - d_k + z_k) + q_k + p_k x_k + h_k (s - d_k + x_k) + F_{k+1} (s - d_k + x_k) + q_k + p_k c_k + g_k z_k + h_k (s - d_k + c_k + g_k) + F_{k+1} (s - d_k + c_k + z_k) \right)
\]

In (6), production \( x_k \) varies from 1 to \( \min \{ c_k, d_{\text{in}} \} - s \), and outsourcing \( z_k \) varies from 0 to \( d_{\text{in}} - s \) or from \( c_k \) to \( d_{\text{in}} - s \). While \( s \) varies from 0 to \( d_{\text{in}} \). Hence, the total time complexity for \( G_k(x) \) is \( O(d_{\text{in}}^5) \).

Since no special property for different values in each iteration, some calculations are repeated in other ones, the above complexity can be improved by keeping minimum values in Queue/Stack. See details in [7] for similar discussions. Although more two cases are needed to be treated in lot-sizing model including outsourcing, while the complexity order is also \( O(n d_{\text{in}}) \).

**Proposition 3.1** There is an \( O(n d_{\text{in}}) \) algorithm for \( \text{X}^{\text{LT-C-O}} \).

We should point out that the DP suggested for \( \text{X}^{\text{LT-C-O}} \) here is based on the fact that products are discrete parts, not continuous variable as something like liquid are included.

References


