The Single Item Discrete Lotsizing and Scheduling Problem: Linear Description and Optimization

Roelof Kuik
Marc Salomon
Stan Van Hoesel*
Luk N. Van Wassenhove

Erasmus University
P.O.Box 1738
NL-3000 DR Rotterdam, The Netherlands

October 19, 1989

Abstract
In this paper the single item Discrete Lotsizing and Scheduling Problem (DLSP) is considered. DLSP is the problem of determining a minimal cost production schedule, that satisfies demand without backlogging and does not violate capacity constraints. We formulate DLSP as an integer programming problem and present three solution procedures. The first procedure is based on a linear description of the convex hull of DLSP, derived from a problem specific network by the so-called projection method. The second procedure is based on a reformulation of DLSP as an assignment problem, with additional restrictions to reflect the specific (setup) cost structure. For this linear programming (LP) formulation it is shown that the solution is all-integer under some additional conditions on the input parameters. Finally, the third procedure is based on a dynamic programming formulation. Using special properties of optimal solutions, the DP-based algorithm can be made to run very fast.

Keywords: Dynamic Programming, Integer Programming, Lot Sizing, Polyhedral Methods.

*Supported by The Dutch Organisation for Scientific Research (NWO) under grant no. 611-304-017.
1 Introduction

The Discrete Lotsizing and Scheduling Problem (DLSP) is the problem of stating the sequence and size of production lots for a number of different items at a single machine. The time horizon is segmented into a finite number of equal-length time periods and demand is assumed to be dynamic. There are capacity constraints stating that at most one item can be produced per period. Furthermore backlogging is not allowed. The problem is to find a feasible production schedule (with respect to capacity constraints) minimizing the sum of setup costs and inventory holding costs. In this paper we consider the single item problem, which is formulated as:

\[ \text{DLSP:} \]
\[ \min \sum_{t=1}^{T} \left( S_t \max\{0, y_t - y_{t-1}\} + h_t I_t \right) \quad (1) \]

subject to

\[ I_{t-1} + r y_t - d_t = I_t \quad \text{for } t = 1, \ldots, T \quad (2) \]
\[ I_t \geq 0 \quad \text{for } t = 1, \ldots, T \quad (3) \]
\[ y_t \in \{0, 1\} \quad \text{for } t = 1, \ldots, T \quad (4) \]

In this non-linear mixed integer model formulation, \( T \) is the number of time periods. If period \( t \) is a production period, the decision variable \( y_t \) equals one, otherwise \( y_t \) equals zero. The initial state of the machine is given by \( y_0 \). The decision variable \( I_t \) represents the inventory position at the end of period \( t \) (\( t \geq 1 \)). The starting inventory is given by \( I_0 \). Furthermore, the constants \( S_t, h_t, d_t \) represent setup cost, holding cost (per unit per period) and demand in period \( t \), respectively. The constant \( r \) is the production rate per period.

The objective is represented by (1). It must be noted that setup costs are incurred only in the first period of an uninterrupted sequence of production periods. The set of equations (2) are the so-called inventory balance equations stating that demand can be fulfilled from production or from inventory. Restrictions (3) are nonnegativity conditions on inventory and restrictions (4) state that production variables are binary.

The non-linear term \( \max\{0, y_t - y_{t-1}\} \) in (1) can easily be linearized by
introducing binary variables $v_t$, which equal one if a setup is made in period $t$, and zero otherwise. Furthermore, the inventory variables can be eliminated from the model formulation, by noting that $I_t = \sum_{r=1}^{t}(r y_r - d_r)$. The $D\bar{L}SP$ can be further simplified by assuming that (without loss of generality):

(a) demand is binary ($d_t \in \{0,1\}$),

(b) production rate equals one ($r = 1$),

(c) starting inventory is zero ($I_0 = 0$),

(d) initial machine state is idle ($y_0 = 0$).

Finally, we introduce constants $D_t = \sum_{r=1}^{t} d_r$ denoting cumulative demand up to period $t$, and constants $h_t, y_t = \sum_{r=1}^{t} h_r$. Straightforward application of the above mentioned assumptions and simplifications yields to the following integer linear programming formulation:

**DLSP**:

$$Z_{DLSP} = \min \sum_{t=1}^{T} (S_t v_t + h_t y_t - h_t D_t) \quad (1')$$

subject to

$$\sum_{r=1}^{t} y_r \geq D_t \quad \text{for } t = 1, \ldots, T \quad (2')$$

$$v_t \geq y_t - y_{t-1} \quad \text{for } t = 1, \ldots, T \quad (5)$$

$$y_t, v_t \in \{0,1\} \quad \text{for } t = 1, \ldots, T \quad (6)$$

Research on DLSP was initiated by Van Wassenhove and Vanderhenst [6], who discuss a hierarchical production planning problem in the chemical industry in which DLSP appeared as a subproblem. A solution procedure for the multi-item problem based on branch and bound and Lagrangean-relaxation is presented by Fleischmann [1]. Recently, Salomon et al. [5] presented problem formulations for several extensions of DLSP, including multiple machines and non-zero setup times. Magnanti and Vachani [3] suggest solution procedures based on polyhedral methods for a related problem. In this paper we follow the latter approach and focus on the single item case. In Section 2 we present a network, from which inequalities describing the convex hull of DLSP can be derived. In Section 3, we suggest an
alternative formulation for DLSP, based on the assignment problem, with additional restrictions to model the setup cost structure. In addition, we show that under non-negativity restrictions on setup and holding costs this LP formulation leads to an optimal (all-integer) solution. Section 4 describes a dynamic programming algorithm with a low order running time, which solves the single item problem (again, under non-negativity restrictions on setup and holding costs). Some conclusions and suggestions for further research are given in Section 5.
2 The Single Item DLSP Polyhedron

In this section we present a linear description of the convex hull of feasible solutions for problem DLSP. This description is implicit since the linear inequalities describing this convex hull are not stated explicitly but can be obtained by finding the generators of an explicitly defined convex cone. The method used to obtain the linear description is the so-called projection method. A theoretical foundation can be found in Nemhauser and Wolsey [4]. The following definitions will be useful:

**Definition.** We define a deadline $t_n$ as the $n$-th period in which demand equals one. More formally,

- $t_0 = 0$
- $t_n = \min\{t \mid 1 \leq t \leq T \text{ and } D_t = n\}$ for $n = 1, \ldots, N$
- $t_{N+1} = T + 1$.

Furthermore, the ordered set of deadlines $\{t_0 < \ldots < t_N < t_{N+1}\}$ will be denoted by $\mathcal{D}$. (Note that $N = D_T$.)

**Definition.** Let $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ be the graph with vertex set $\mathcal{V}$, given by

$$\mathcal{V} = \bigcup_{n=0}^{D_T} \{(s, n) \mid n \leq s \leq t_{n+1} - 1\}.$$ 

Note that $\mathcal{V} = \{(s, n) \mid \exists \text{ a feasible DLSP solution } (y, v) \text{ with } \sum_{i=1}^{T} y_i = n\}$.

In addition, the graph $\mathcal{G}$ has arc set $\mathcal{A} = \bigcup_{i=1}^{3} \mathcal{A}_i$ where the three arc sets $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}_3$ are given by,

- $\mathcal{A}_1 = \{(s_1, n_1), (s_2, n_2) \in \mathcal{V} \times \mathcal{V} \mid s_1 - s_2 = n_1 - n_2 < 0\}$

and

- $\mathcal{A}_2 = \{(s_1, n_1), (s_2, n_2) \in \mathcal{V} \times \mathcal{V} \mid n_1 = n_2 \text{ and } s_2 - s_1 = 1\}$

while $\mathcal{A}_3$ is defined by stating that there is a one-to-one correspondence between arcs in $\mathcal{A}_2$ and in $\mathcal{A}_3$ such that corresponding arcs have the same begin and end point (vertex). In Figure 1 the graph $\mathcal{G}$ is shown graphically with arcs from set $\mathcal{A}_i$ denoted by $a_i$. 

5
Definition. Let $A$ denote the $|V| \times |A|$ vertex-arc incidence matrix of the graph $G$ and define $f \in \mathbb{R}^{|V|}$ as:

$$f(s, n) = \begin{cases} 
1 & \text{if } (s, n) = (0, 0) \\
-1 & \text{if } (s, n) = (T, D_T) \\
0 & \text{otherwise.}
\end{cases}$$

Moreover, let $\mathcal{P}_G$ be the polyhedron in $\mathbb{R}^{|A|}$ defined by,

$$\mathcal{P}_G = \{ x \in \mathbb{R}^{|A|} | x \geq 0 \text{ and } Ax = f \}.$$ 

and let the polyhedron for $DLSP$ be defined by:

$$\mathcal{P}_{DLSP} = \text{Conv} \{(y, v) | (y, v) \text{ is a feasible solution for DLSP} \},$$
where Conv(. ) stands for the convex hull of a set .

**Definition.** Let $B$ be the $2T \times |A|$ matrix with columns labeled by arcs from the graph $G$ and matrix elements given by

$$B_{s,a} = \begin{cases} 
1 & \text{if } a \in A_1 \text{ and } s_1 < s \leq s_2 \\
1 & \text{if } a \in A_1 \cup A_2 \text{ and } t - T = s_1 + 1 \\
0 & \text{otherwise.}
\end{cases}$$

**Proposition 1** The following two assertions hold.

a. For every feasible solution $(y, v)$ to DLSP, there exists a vector $x$, extremal in $P_G$, so that $(y, v)' = Bx$.

b. Conversely, for every extremal point $x \in P_G$ a feasible solution $(y, v)$ of DLSP is provided by $(y, v)' = Bx$.

The formal proof of this proposition can be found in Kuik et al. [2]. Here we only give some intuitive insight in the relation between a path in $G$ from $(0,0)$ to $(T,DT)$ and a feasible solution of DLSP. To this end, let $x$ be an extremal point in $P_G$. We associate the solution $(y, v)$ of DLSP to the path $x$ by,

- for $x_a = 1$ do a set up in period $s_1 + 1$ and produce in periods $s_1 + 1$ through $s_2$ for $a \in A_1$,

- for $x_a = 1$ do a set up in period $s_1 + 1$ for $a \in A_2$,

- for $x_a = 1$ leave period $s_1 + 1$ idle for $a \in A_3$.

From Proposition 1 we obtain the linear description of $P_{DLSP}$ as described in the following proposition.

**Proposition 2** We have,

$$P_{DLSP} = \{(y, v)' \in R^{2T} | \alpha f + \beta(y, v)' \geq 0 \}_{(\alpha, \beta) \in R^T \times R^T \text{ with } \alpha A + \beta B \geq 0}$$

Note that the graph $G$ can be considered as the state-space graph associated with a dynamic programming recursion for finding an optimal solution to DLSP. By allocating appropriate costs to each arc in the state-space graph, the dynamic programming algorithm corresponds to finding the shortest path in the graph. The states in the dynamic programming algorithm are
precisely the vertices of the graph $\mathcal{G}$, while the outgoing arcs from a vertex represent the options (possible decisions). More precisely, if we associate a cost $c_a$ with arc $a \in \mathcal{A}$ and if we define $C(s, n)$ as the total minimal cost to reach state $(s, n)$, the forward recursion reads:

$$C(0, 0) = 0$$

$$C(s, n) = \min_{((t, m), (s, n)) \in \mathcal{A}} \{C(t, m) + c_{((t, m), (s, n))}\} \text{ if } (s, n) \neq (0, 0).$$

Using this forward DP recursion, it can easily be seen that the single item DLSP with arbitrary cost structure is solved in $O(T^2)$.
3 A Strong LP formulation for the Single Item DLSP.

In this section we show that a solution for DLSP (under some additional assumptions on the input parameters), can be found by solving a linear programming model introduced below. This linear program will be called Reformulated DLSP (RDLSP). It is basically an assignment problem, with additional restrictions to account for setup costs.

\[ Z_{RDLSP} = \min \sum_{i \in \mathcal{D}} \sum_{s=1}^{t} (S_i u_{s,t} + h_{s,t-1} z_{s,t}) \]  

subject to

\[ \sum_{s=1}^{t} z_{s,t} = 1 \quad t \in \mathcal{D} \]  

\[ \sum_{i \geq s} \sum_{t \in \mathcal{D}} z_{s,t} \leq 1 \quad s = 1, \ldots, T \]  

\[ u_{s,t} \geq z_{s,t} - z_{s-1,t-1} + \begin{cases} \quad n = 2, \ldots, D_T; s = 1, \ldots, t_n \\ \quad t_{n-1} \geq s - 1 \geq 1 \end{cases} \]  

\[ u_{s,t} \geq z_{s,t} + \begin{cases} \quad n = 1, \ldots, D_T; s = 1, \ldots, t_n \\ \quad t_{n-1} < s - 1 \end{cases} \]  

\[ 0 \leq u_{s,t} \leq 1 \quad t \in \mathcal{D}; s = 1, \ldots, t \]  

\[ z_{s,t} \geq 0 \quad t \in \mathcal{D}; s = 1, \ldots, t \]  

In RDLSP the variable \( z_{s,t} \) denotes the production quantity produced in period \( s \) to fulfil demand in period \( t (\in \mathcal{D}) \). The variable \( u_{s,t} \) equals one if a setup takes place in period \( s \) and \( z_{s,t} = 1 \). The objective function is represented by (7) and the restrictions (8) and (9) assure that demand is fulfilled without backlogging and that capacity limitations are not violated, respectively. Restrictions (10) and (11) relate setup and production variables and restrictions (12) and (13) guarantee that setup and production variables are bounded by zero and one.

With respect to the problem parameters we make the following assumptions:
Assumption 1

(a) Setup costs \((S_t)\) are non-negative and non-increasing in \(t\).

(b) Holding costs \((h_{t_1,t_2})\) are non-increasing in \(t_1\) for fixed \(t_2\).

(Note that Assumption 1 (b) holds for holding cost \(h_{t_1,t_2}\), defined through \(h_{t_1,t_2} = \sum_{\tau = t_1}^{t_2} h_\tau\) with \(h_\tau\) nonnegative.)

In what follows, we use the concept of production batches. We define a production batch as an uninterrupted sequence of production periods, that can be constructed from any solution \((z,u)\) of RDLSP in the following way:

**Step 1:** Take an arbitrary \(i\) for which \(u_{s,t_i} > 0\). Let \(\ell\) be equal to the smallest \(k \geq 0\) for which \(z_{s+k,t_i+k}\) is equal to zero. If such \(k\) does not exist, put \(\ell\) equal to \(D_T + 1\). Let \(j\) be equal to \(i + \ell\).

**Step 2:** Compute the batch amplitudes, \(\Delta_{s,i,j}\) through

\[
\Delta_{s,i,j} = \min \{ u_{s,t_i}, \min_{0 \leq k < \ell} z_{s+k,t_i+k} \}.
\]

**Step 3:** Reduce the quantities \(u_{s,t_i}\) and \(z_{s+k,t_i+k}\) \((k = 0, ..., \ell - 1)\) by an amount \(\Delta_{s,i,j}\).

The batch that we obtain in this manner starts in period \(s\) and fulfills (part of) the demand in the periods \(t_i\) until \(t_{j-1}\). In what follows, we denote this batch by \(B_{s,i,j}\).

By executing steps 1, 2, and 3 iteratively until all \(z_{s,t}\) are equal to zero, we ultimately obtain a complete split-up into batches of the solution of RDLSP. Note that, by doing so, inequalities (10) and (11) remain satisfied.

Let \(C\) be the set of pairs consisting of a solution, \((z,u)\), to RDLSP and (a set of) batch amplitudes, \(\Delta \equiv \{ \Delta_{i,j,s} \}\), such that the amplitudes, \(\Delta\), can be obtained from \((z,u)\) through successive application of Steps 1, 2, and 3. Clearly \(C\) is convex.

**Theorem 1** Under Assumption 1, it follows that every extremal optimal solution to RDLSP is all-integer.

**Proof.** The proof is presented in three steps, denoted by A, B, and C respectively.
Step A: Take a solution \((z,u)\) to RDLSP and split it into batches, using steps 1, 2, and 3. The batch amplitudes \(\Delta_{s,i,j}\) are represented as flows in a network with nodes \(i = 1, \ldots, DT + 1\) corresponding to the deadlines and arcs \((i,j)\), corresponding to the possible starting times \(s = 1, \ldots, t_i\) of batch \(B_{s,i,j}\). This network is shown in figure 2. The cost associated with an arc \((i,j)\) with starting time \(s\) equals \(S_s + \sum_{k=0}^{i-1} h_{s+k,i+k}\).

![Diagram](image)

Figure 2

First we prove that any solution of RDLSP defines through the batch amplitudes a flow of magnitude one from node 1 to node \(DT + 1\). To this end, the following two assertions must be shown to hold:

(a) outflow at node 1 = 1

(b) outflow at node \(k + 1\) = inflow at node \(k + 1\) (\(k \geq 1\)).

Assertion (a) holds since

outflow at node 1 = \(\sum_{s,j} \Delta_{s,i,j}\) = total production for \(t_1 = 1\)

and assertion (b) holds since

\[
1 = \text{production for period } t_k
\]

\[
= \sum_{i \leq j \leq j-1} \Delta_{s,i,j} + \sum_{i \leq k < j-1} \Delta_{s,i,j,k+1}
\]

11
and
\[ 1 = \text{production for period } t_{k+1} \]
\[ = \sum_{i \leq k+1} \Delta_{s,i,j} = \sum_{i < k+1} \Delta_{s,i,j} + \sum_{i,j} \Delta_{s,k+1,j}. \]

From this, it follows that
\[ \text{the outflow at node } k+1 = \sum_{s,j} \Delta_{s,k+1,j} \]
\[ = \sum_{s,i} \Delta_{s,i,k+1} \]
\[ = \text{inflow at node } k+1, \]

proving assertion (b).

**Step B:** Secondly, we prove that there exists an extremal optimal solution to the minimal cost flow problem that corresponds to an binary optimal solution of RDLSP. Every extremal solution to the minimal cost flow problem is integer (with a flow equal to 1). Furthermore, an optimal solution of this minimal cost flow problem exists with the property that the flow uses at most one arc from \( i \) to \( j \), namely an arc with the lowest cost. This lowest cost arc can be taken as the arc corresponding to a batch that starts in \( s = t_i \). Constructing a (tentative) \((z, u)\) solution to RDLSP from this flow \((= \text{production batches})\) by inverting the Steps 1, 2, and 3 then leads to a feasible solution to RDLSP. Since \( \Delta_{s,i,j} \) is binary, it follows that \( u_{s,t} \) and \( z_{s,t} \) are binary.

Combining **Step A** and **Step B** we conclude that RDLSP has an optimal solution that is all-integer.

**Step C:** Let \((z, u)\) be an extremal optimal solution and consider \(((z, u), \Delta) \in C\). Then \( \Delta \) is an optimal solution to the min cost flow problem. Let \( \Delta = \sum_{n=1}^{N} \alpha_n \Delta^{(n)} \) be a convex decomposition of \( \Delta \) into extremal optimal solutions \( \Delta^{(n)} \) to the min cost flow problem (with \( \alpha_n > 0 \)). One verifies that there exists a set of solutions \(((z, u)^{(n)})_{n=1}^{N} \in \text{RDLS}\) such that \(((z, u)^{(n)}, \Delta^{(n)}) \in C\) for \( n = 1, \ldots, N \) and \( \sum_{n=1}^{N} \alpha_n (x, u)^{(n)} = (z, u) \). But as \((z, u)\) is extremal we must then have \((x, u)^{(n)} = (z, u)\) for \( n = 1, \ldots, N \).
Hence \(((z, u), \Delta^{(n)}) \in C\) and as \(\Delta^{(n)}\) is integer it follows that \((z, u)\) is integer. \(\square\)

Consider the map \(\mathcal{M}\) from the set of RDLSP solutions to the set of DLSP solutions given by \(y_s = \sum_t z_{s,t}\) and \(v_s = \sum_t u_{s,t}\).

**Corollary 1** Let \(h_{t_1, t_2} = \sum_{r=t_1}^{t_2} h_r\). Then, under \(\mathcal{M}\) an optimal solution of RDLSP is mapped onto an optimal solution of DLSP and every optimal solution of DLSP is the image under \(\mathcal{M}\) of an optimal solution of RDLSP.

**Proof.** The only non evident assertion is that every optimal solution of DLSP is the image under \(\mathcal{M}\) of an optimal solution of RDLSP. However this follows readily by observing that an optimal \((y, v)\) solution can be transformed into an optimal \((z, u)\) solution, using the following transformation for the \(z\) variables:

\[
z_{s,t} = \begin{cases} 
1 & \text{if } s = s_k \text{ and } t = t_k \\
0 & \text{otherwise.}
\end{cases}
\]

where \(s_k = \min\{s \mid \sum_{r=1}^{s} y_r = k\}\).

The variables \(u_{s,t}\) can be obtained in a similar way. For the constructed solution \((z, u)\) one easily verifies that application of \(\mathcal{M}\) yields \((y, v)\). \(\square\)

**Remark.** Instances of RDLSP, not satisfying Assumption 1, can be constructed such that there exists an extremal optimal solution that is non integer.
4 A Dynamic Programming based algorithm.

In this section we formulate a Dynamic Programming (DP) algorithm for the single item DLSP problem. The algorithm is inspired by the $O(T \log T)$ algorithm for the Economic Lotsizing Problem by Wagelmans, Van Hoesel and Kolen [7]. Before we present a detailed description of our DP algorithm, we state the following theorem:

**Theorem 2 (Zero-Switch property)** Under Assumption 1, there exists an optimal solution $(y, v)$ to DLSP, such that:

$$v_t(i_t - 1 + (1 - d_t)) = 0.$$  

**Proof.** This theorem follows directly from the proof of Theorem 1 (Section 3), since it was shown there that there exists an optimal solution to RDLSP consisting only of batches $B_{s,i,j}$ with $t_i, t_j \in \mathcal{D}$ that start in periods $s = t_i$. □

**Definition** Let $C(t)$ $(t \in \mathcal{D})$ be the minimal total cost to fulfill demand in periods $t$ until $T$ (under the assumption of zero inventory at the start of period $t$).

The Zero-Switch property enables us to find an optimal solution to DLSP by using the following backward DP recursion:

$$C(t_{N+1}) = 0$$

$$C(t_n) = \min_{t \in \mathcal{F}_n} \{S_{t_n} + H(t_n, t) + C(t)\} \text{ for } n = N, ..., 1.$$  \hspace{1cm} (14)

where

$$H(t_n, t) = \sum_{k=0}^{D_t - D_{t_n} - 1} h_{t_n + k, T}$$

and $\mathcal{F}_n \equiv \{t_k, t_{k+1}, ..., t_{N+1}\}$, with $k$ such that

$$k = \min\{j > n \mid d_{t_{j-1}} = 0 \text{ and } d_{t_j} = 1\}.$$  

In the following we will establish how this DP can be streamlined to become very efficient. The streamlining is implemented through replacement at each stage $n$ in the DP of $\mathcal{F}_n$ by an ordered set $\mathcal{S}_n \subset \{t_{n+1}, ..., t_{N+1}\}$, such that the minimum in (14) occurs for the last element in $\mathcal{S}_n$. The key to the
efficiency of the streamlined DP, as we will show, is that $S_{n-1} \subseteq \{t_n\} \cup S_n$ and that the ordered set $S_{n-1}$ can be constructed fast from the ordered set $S_n$ (and some extra information).

**Definition:** For $t, t' \geq \tau$, we say that $t \in D \setminus \{0\}$ dominates $t' \in D \setminus \{0\}$ with respect to $\tau \in D \setminus \{0\}$, if

$$S_\tau + H(\tau, t) + C(t) \leq S_\tau + H(\tau, t') + C(t')$$

or equivalently,

$$C(t) - C(t') \leq H(\tau, t') - H(\tau, t).$$

In what follows, we denote dominance of $t$ over $t'$ with respect to $\tau$ by $t \triangleright t'$, or by $t' \triangleleft t$.

**Lemma 1** For $m_1 < m_2$ it holds that $H(t_n, t_{m_2}) - H(t_n, t_{m_1})$ is non-increasing in $n$, for $n = 1, \ldots, m_1 - 1$.

**Proof.** After simple calculations we obtain that

$$H(t_n, t_{m_2}) - H(t_n, t_{m_1}) - [H(t_{n+1}, t_{m_2}) - H(t_{n+1}, t_{m_1})] =$$

$$\sum_{k=0}^{D_{t_{m_2}}-D_{t_{m_1}}-1} (h_{t_n} + D_{t_{m_2}} - D_{t_{m_1}} + k, T - h_{t_{n+1}} + D_{t_{m_2}} - D_{t_{m_1}} + k, T).$$

Since $h_{t,T}$ is nonincreasing in $t$, it is sufficient to show that $t_n + D_{t_{m_1}} - D_{t_n} \leq t_{n+1} + D_{t_{m_2}} - D_{t_{n+1}}$. But this is obvious, because demand is binary so that $D_{t_{n+1}} - D_{t_n} \leq t_{n+1} - t_n$. $\square$

**Corollary 2** For $t_{m_1}, t_{m_2} \in F_n$ with $t_{m_1} < t_{m_2}$ it holds that if $t_{m_1} \triangleright t_{m_2}$, then $t_{m_1} \overset{t}{\mapsto} t_{m_2}$ for $t > t_n \in D$.

**Corollary 3** For each pair of periods $t_{m_1}, t_{m_2} \in D \setminus \{0\}$, with $t_{m_1} < t_{m_2}$, there exists a "reversal" period $r(t_{m_1}, t_{m_2}) \in D$ with $r(t_{m_1}, t_{m_2}) \leq t_{m_1}$ such that the following condition holds

$$(t_{m_1} \geq t \text{ and } t_{m_1} \overset{t}{\mapsto} t_{m_2}) \iff r(t_{m_1}, t_{m_2}) \geq t$$

for $t \in D \setminus \{0\}$. 

15
Having introduced dominance we now turn to introducing the sets $\mathcal{S}_n$.

**Definition.** For $1 \leq n \leq N + 1$ the set $\mathcal{S}_n \equiv \{t^n_{i}\}_{i=1}^{[S_n]}$ is defined as the largest ordered subset of $\{t_{n+1}, \ldots, t_{N+1}\}$ such that,

$$ t \in \mathcal{S}_n \Rightarrow \exists s \in D : t_{n+1} \leq s < t \text{ such that } s \overset{\text{in}}{\rightarrow} t. $$

We now define $\mathcal{S}_n \equiv \{t^n_{i}\}_{i=1}^{[S_n]}$ as the maximal (ordered) subset of $\mathcal{S}_n$ such that,

$$ t_j \in \mathcal{S}_n \implies \exists t_i, t_k \in \mathcal{S}_n \text{ with } t_i < t_j < t_k \text{ and } r(t_i, t_j) \geq r(t_j, t_k). $$

Note that the set $\mathcal{S}_n$ has the following structure:

1) $t_{n+1} = t^n_1 < \ldots < t^n_{[S_n]}$

2) $t_{n+1} = t^n_{i} \overset{\text{in}}{\rightarrow} \ldots \overset{\text{in}}{\rightarrow} t^n_{[S_n]}$

3) $r(t^n_i, t^n_{i+1}) < r(t^n_{i+1}, t^n_{i+2})$ for $i = 1, \ldots, [S_n] - 2$.

During the streamlined DP we will construct the sequence of subsets $\mathcal{S}_{N+1}, \mathcal{S}_N, \mathcal{S}_{N-1}, \ldots$ by recursion. First we have to establish that the recursion given as,

$$ C(t_{N+1}) = 0 $$

$$ C(t_n) = S_{t_n} + H(t_n, t^n_{[S_n]}) + C(t^n_{[S_n]}) $$

for $n = N, \ldots, 1.$

is valid. For validity to hold it is sufficient to prove the following lemma.

**Lemma 2** For each $n$ it holds

$$ H(t_n, t^n_{[S_n]}) + C(t^n_{[S_n]}) \leq H(t_n, t) + C(t) $$

for all $t = n + 1, \ldots, N + 1$.

**Proof.** Let $t^*$ be the smallest period $t$ (strictly) larger than $t_n$ for which

$$ H(t_n, t) + C(t) $$

is minimal. Then obviously $t^* \in \mathcal{S}_n$. So $t^n_{[S_n]} \overset{\text{in}}{\rightarrow} t^*$. But since $t^*$ is a "mining period" it dominates all other periods in $\{t_{n+1}, \ldots, t_{N+1}\}$ and thus a fortiori $t^n_{[S_n]}$. Therefore
\[ H(t_n, t^*) + C(t^*) = H(t_n, \tau_{|S_n|}^n) + C(\tau_{|S_n|}^n) . \]

Noting that \( \tau_{|S_n|}^n = \tau_{|S_n|}^n \) completes the proof of the lemma. \( \Box \)

We now show how \( S_n \) can be obtained from \( S_{n+1} \). In particular, suppose we have the following data (for stage \( n + 1 \))

**Data known for stage \( n + 1 \)**

- the ordered set \( S_{n+1} \),
- \( r(t_i^{n+1}, t_{i+1}^{n+1}) \) for \( i = 1, \ldots, |S_{n+1}| - 1 \),
- \( C(t_\ell) \) for \( \ell = n + 2, \ldots, N + 1 \).

Then, the data for stage \( n \) can be computed using the following algorithm:

**Algorithm for stage \( n \):**

**Step 1** \( C(t_{n+1}) = S_{n+1} + H(t_{n+1}, t_{n+1}^{n+1}) + C(t_{n+1}^{n+1}) \),

**Step 2** \( S'_n := S_{n+1} \),

**Step 3** if \( r(t_{|S_{n+1}|-1}^{n+1}, t_{|S_{n+1}|}^{n+1}) = t_n \) then delete \( t_{|S_{n+1}|}^{n+1} \) from \( S'_n \),

**Step 4** \( S'_n := \{ t_{n+1} \} \cup S'_n \),

**Step 5** determine \( i^* \) as the largest \( i \) for which \( t_{n+1} \rightharpoonup t_i \leftarrow t_{i+1}^{n+1} \).

\[ S'_n := S'_n \setminus \bigcup_{i=1, \ldots, i^*} \{ t_{i+1}^{n+1} \} \]

**Step 6** renumber \( S'_n \) and put \( S'_n = \{ t'_1, \ldots, t'_{|S'_n|} \} \).

**Step 7** compute \( i^* \) as the largest \( i \) for which \( r(t'_i, t'_i) \geq r(t'_i, t'_{i+1}) \).

\[ S'_n := S'_n \setminus \bigcup_{i=2, \ldots, i^*} \{ t'_{i+1}^{n+1} \} ; \]

note that the reversal periods are computed efficiently using binary search.

**Step 8** renumber \( S'_n \).

The following lemma is easily verified and a formal proof is left to the reader.
Lemma 3 $S_n = S'_n$

Note that indeed by this lemma the computation preceding it shows how to compute the data for stage $n$ from the data for stage $n + 1$.

Before we assert the complexity of our streamlined DP-algorithm, we first explain how to compute $H(t_n,t_m)$ for arbitrary $n$ and $m$ in constant time. To do so, we need to perform the following preprocessing step:

**Preprocessing Step:** First, compute $h_{t,T}$ for all $t = 1,..,T$. Since $h_{t,T} = h_t + h_{t+1,T}$, this can be done in $O(T)$. Secondly, compute $\sum_{\tau=t}^T h_{\tau,T}$ for $t = 1,..,T$. This can also be done by backward recursion in $O(T)$.

Since $H(t_n,t_m) = \sum_{\tau=t_n}^T h_{\tau,T} - \sum_{\tau=t_n+D_{tn}}^{t_m+D_{tn}} h_{\tau,T}$, it is clear that computation can be performed in constant time.

**Theorem 3** The complexity of the streamlined DP-algorithm is $O(T + |D| \log |D|)$

**Proof.** We make the following observations:

(a) Preprocessing requires $O(T)$.

(b) Step 1, Step 2, Step 3, Step 4, Step 6 and Step 8 require constant time and must be executed $|D|$ times.

(c) At stage $n$ Step 5 has a running time of $O(f_n + 1)$, where $f_n$ is the number of deletions performed during the computations at stage $n$. Since $\sum_n f_n \leq |D|$, this step requires in total a running time of $O(|D|)$.

(d) Step 7 for stage $n$ has a running time of $O(f_n' \log |D|)$, where $f_n'$ is the number of deletions. Since $\sum_n f_n' \leq |D|$, the overall running time of this step is $O(|D| \log |D|)$.

From the four observations made above, it follows that the running time of the DP-based algorithm is $O(T + |D| \log |D|)$.
Remark. If setup costs are constant, the DP-algorithm can be adapted, such that the running time becomes $O(T + |D_0|\log|D_0|)$, where $D_0 = \{t \mid d_t = 1 \text{ and } d_{t-1} = 0\}$. 
5 Summary and Conclusions

In this paper the single item Discrete Lotsizing and Scheduling (DLSP) problem is considered. A network is presented from which a linear description of the convex hull of DLSP can be obtained implicitly, using the projection method. Furthermore, a LP formulation, which yields all-integer solutions under certain restrictions on the input parameters is given. Finally, we present a Dynamic Programming algorithm which uses properties of optimal DLSP solutions and requires $O(T + |D| \log |D|)$ computation time.

Directions for further research are to develop a cutting plane algorithm for the multi-item problem, using the network described in this paper. In this context, it is also worthwhile to search for an explicit linear description of the convex hull of DLSP. Furthermore, it is interesting to extend the DLSP formulations and algorithms to allow for non-zero setup times.
References


