AMSI VACATION RESEARCH SCHOLARSHIPS 2017-2018

Lot Sizing on a Cycle

Riley Cooper
Supervised by Hamish Waterer
University of Newcastle

2018-03-08

Vacation Research Scholarships are funded jointly by the Department of Education and Training and the Australian Mathematical Sciences Institute.

Australian Government
Department of Education and Training

AMSI AUSTRALIAN MATHEMATICAL SCIENCES INSTITUTE

1 Overview

Production planning and scheduling problems are a key component in many supply chains. A recurring model in these problems is the multi-item lot sizing model in which a schedule of production is determined for each item being produced so that the demand for each item is satisfied subject to constraints on production, inventory and the operation of the machinery. In order to better understand how to effectively solve mixed-integer programming formulations of multi-item lot sizing models, much of the work in the literature has focused on different variants of the single-item problem. See, for example, Pochet and Wolsey (2006) for a comprehensive review of the lot sizing literature.

In the classical, deterministic, single-item lot sizing problem, the underlying network is a path. Recent work (Guan, Ahmed, Nemhauser, and Miller, 2006; Guan and Miller, 2008; Di Summa and Wolsey, 2008) considers the lot sizing problem on a tree which has applications in stochastic lot sizing. In this project we will investigate a variant of the deterministic single-item lot sizing problem in which the underlying network is a cycle. Such a variant is motivated by strategic production planning and scheduling problems in which it is often convenient to assume that the planning horizon wraps around on itself, thereby eliminating the need to specify boundary conditions and avoiding possible end effects. Such an approach can be viewed as a form of steady state model.

In this project we will introduce the single-item lot sizing problem on a cycle and relate it to the existing problems in the literature. We will propose a mixed-integer programming formulation for the problem, explore the structural properties of the optimal solutions, and use them to establish the computational complexity of the problem and develop efficient algorithms for its solution. Finally we will investigate the polyhedral structure of the convex hull of the set of feasible solutions to the problem and propose extended formulations and strong reformulations for the problem.

2 Notation

- Given $k, l \in \mathbb{Z}$, let $[k, l]$ denote the set $\{k, k+1, \ldots, l\}$ if $k \leqslant l$, and $\{k, k+1, \ldots, n\} \cup \{1, 2, \ldots, l\}$ if $k > l$. Let $[n]$ denote the set $[1, n]$.
- Given a parameter α indexed by the periods $t \in [n]$, let α_{kl} denote the summation $\sum_{t \in [k,l]} \alpha_t$ if $k \leqslant l$ and $\sum_{t \in [k,n]} \alpha_t + \sum_{t \in [1,l]} \alpha_t$ if $k > l$.
- Following the three-field identifier PROB-CAP-VAR classification scheme for canonical single-item lot sizing problems (see, for example, Pochet and Wolsey, 2006), let LS-C-PATH denote the classical capacitated single-item lot sizing problem when the underlying network is a path, LS-C-TREE denote this problem when the underlying network is a tree, and LS-C-CYCLE denote this problem when the underlying network is a cycle.
- For a graph $G = (N, A)$, where N is the set of nodes and A the set of arcs, a *cut* is a partition of the set of nodes N into two parts, S and $\bar{S} = N \setminus S$. Each cut defines a set of arcs consisting of

those arcs that have an endpoint in S and another end point in $\bar{S}$. The outward flowing capacity $u[S, \bar{S}]$ of partition $(S, \bar{S})$ is given by

$$u[S, \bar{S}] = \sum_{(i,j) \in (S, \bar{S})} u_{ij},$$

where u_{ij} is the capacity of arc (i, j).

- For a graph $G = (N, A)$, we will let $b(i)$ represent the supply/demand of node i. If $b(i) > 0$ then i is a supply node and if $b(i) < 0$ then i is a demand node.

We will also introduce a theorem to aid a proof later in this report.

Theorem 1 (Ahuja, Magnanti, and Orlin (1993))**.** *The minimum cost network flow problem has a feasible solution if and only if for every subset* $S \subseteq N$*,* $b(S) - u[S, \bar{S}] \leq 0$ *where* $b(S) = \sum_{i \in S} b(i)$*.*

3 Problem Definition

An instance of LS-C-CYCLE consists of n time periods, a production capacity C_t and a demand d_t for a single item in each time period $t \in [n]$. Once produced, the item can be held in inventory to satisfy demand in a later period. A schedule is a set of setup time periods and production levels. A feasible schedule is a schedule such that production does not exceed capacity, production only occurs when the item is set up, and the demand for the item is satisfied in each time period. The problem is to find a feasible schedule that minimises the fixed setup costs and per unit production and inventory holding costs.

Let x_t denote the amount produced in period t and s_t denote the amount of stock in inventory at the end of period t. Let $y_t = 1$ indicate that the item is setup for production in period t and $y_t = 0$ otherwise. Let p_t and h_t denote the per unit production and inventory holding costs, and f_t denote the fixed setup cost, for period $t \in [n]$.

Consider the following MIP formulation for LS-C-CYCLE:

$$\text{minimise} \sum_{t \in [n]} (p_t x_t + h_t s_t + f_t y_t) \tag{1}$$

$$\text{subject to } s_{t-1} + x_t = d_t + s_t, \qquad t \in [n] \tag{2}$$

$$x_t \leqslant C_t y_t, \qquad t \in [n] \tag{3}$$

$$s_0 = s_n \tag{4}$$

$$x_t,\ s_t \geqslant 0, \qquad t \in [n] \tag{5}$$

$$y_t \in \{0, 1\}, \qquad t \in [n] \tag{6}$$

The objective function (1) is simply minimise the costs for each period summed for all $t \in [n]$. This includes the production cost multiplied by amount of production $(p_t x_t)$, the holding cost multiplied

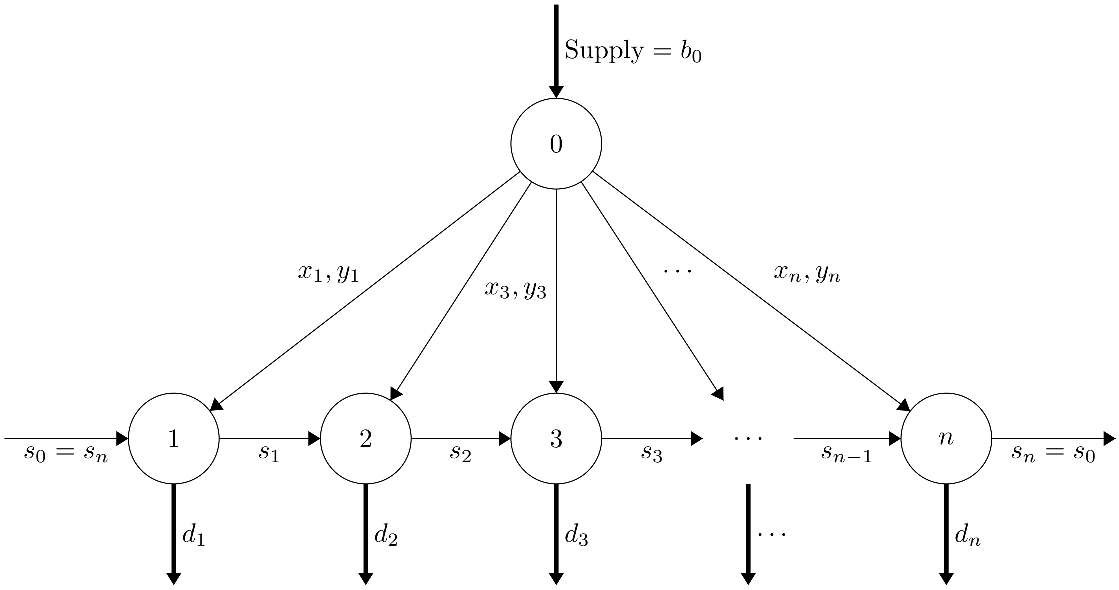

Figure 1: LS-C-CYCLE represented as a fixed charge network flow problem.

by the amount held to the next period ($h_t s_t$) and the fixed production cost multiplied by the binary variable determining whether or not production takes place in the period ($f_t y_t$). Constraint (2) is called the inventory balance constraint. This constraint ensures that the stock carried into each period in the cycle is equal to the demand of that period plus the stock carried out of the period. Constraint (3) ensures that if there is production in period t, i.e. $x_t > 0$, then the binary set-up variable $y_t = 1$. While doing this, the constraint also guarantees that the production x_t is no more than the capacity for that period C_t. If the period is not a production period, i.e. $y_t = 0$, the production is zero. Constraint (4) is the cycle constraint. This constraint ensures that the amount of stock held after period n is equal to the amount of stock carried into period 1. Constraint (5) ensures that the amount of production and amount of stock held for each period is nonnegative. Constraint (6) ensures that the y_t variables are binary.

LS-C-CYCLE is a fixed charge network flow problem, that is, a minimum cost network flow problem with fixed costs for the use of certain arcs. The network representation of LS-U-CYCLE is illustrated in Figure 1.

Observation 1. LS-C-CYCLE is a generalisation of LS-C-PATH in the sense that LS-C-PATH is the special case of LS-C-CYCLE in which $s_0 = s_n = 0$.

Definition 1.

(a) LS-C-CYCLE is said to have *constant capacity* when $C_t = C$ for all $t \in [n]$. We denote this variant of the problem as LS-CC-CYCLE.

(b) LS-C-CYCLE is said to be *uncapacitated* when $C_t \geqslant d_{1n}$ for all $t \in [n]$. We denote this variant of the problem as LS-U-CYCLE.

Assumption 1. $q_t \geq 0$ *for all* $t \in [n]$.

As is the case for LS-C-PATH, we may assume that for instances of LS-C-CYCLE with negative fixed costs, it suffices to replace the fixed costs q_t by $q_t^+ = max[q_t, 0]$ for $t \in [n]$ and add the constant term $\sum_t min[q_t, 0]$ to the objective function. This is because if $q_t < 0$ then $y_t = 1$ in any optimal solution. So replacing the term $q_t y_t$ by $0y_t + q_t$ does not affect the optimal value of the problem. This result may also apply to LS-U-CYCLE.

4 Structure of Feasible Solutions to LS-C-CYCLE

In this section we will discuss the structure of feasible solutions to LS-C-CYCLE. We will let $X^{\text{LS-C-CYCLE}} = \{(x, y, s) \in \mathbb{R}^n \times \mathbb{B}^n \times \mathbb{R}^{n+2} : (2\text{–}6)\}$ denote the set of feasible solutions to the MIP formulation of LS-C-CYCLE. Clearly LS-C-CYCLE is feasible when $X^{\text{LS-C-CYCLE}} \neq \emptyset$. Below we claim conditions under which $X^{\text{LS-C-CYCLE}} \neq \emptyset$. To aid in our proof of Claim 1, we will use Theorem 1.

Claim 1. *The set* $X^{\textit{LS-C-CYCLE}} \neq \emptyset$ *if and only if* $0 \leqslant d_{1n} \leqslant C_{1n}$.

Proof. Firstly, observe that the supply of node 0 is equal to the sum of all demands. We can show this by simply summing constraint (2) for all $t \in [n]$. This gives $s_0 + x_{1n} = d_{1n} + s_n$. Using constraint (4) we can eliminate s_0 and s_n giving us $x_{1n} = d_{1n}$. The flow balance at node 0 implies that the supply of node 0 equals d_{1n}. Now, imagine we take a subset of nodes $S \subseteq N$ where $N = [n] \cup 0$. There are five ways in which this can be done.

1. $S = \{0\}$, only the supply node is included.
2. $S = \{0, t_1, \ldots, t_m\}$, the supply node and any other combination of m nodes from the set of nodes in the cycle $[n]$, where $m < n$.
3. $S = \{t_1, \ldots, t_m\}$, any combination of m nodes from the set of nodes in the cycle $[n]$, where $m < n$.
4. $S = [n]$, all nodes in the cycle.
5. $S = N$, all nodes are included in S.

We will find the conditions under which the inequality from theorem 1, $b(S) - u[S, \bar{S}] \leq 0$, holds. For case 1:

$$S = \{0\},\ b(S) = d_{1n} \text{ and } u[S, \bar{S}] = C_{1n}$$

$$\implies b(S) - u[S, \bar{S}] = d_{1n} - C_{1n} \leq 0$$

$$\implies \text{The inequality holds for this case if and only if } d_{1n} \leq C_{1n}.$$

For case 2:

$$S = \{0, t_1, \ldots, t_m\},\ b(S) = d_{1n} - d_{t_1} - \ldots - d_{t_m} \text{ and } u[S, \bar{S}] = \infty$$

$$\implies b(S) - u[S, \bar{S}] = -\infty < 0$$

$$\implies \text{The inequality holds for this case.}$$

For case 3:

$$S = \{t_1, \ldots, t_m\},\ b(S) = -d_{t_1} - \ldots - d_{t_m} \text{ and } u[S, \bar{S}] = \infty$$

$$\implies b(S) - u[S, \bar{S}] = -\infty < 0$$

$$\implies \text{The inequality holds for this case.}$$

For case 4:

$$S = [n],\ b(S) = -d_{1n} \text{ and } u[S, \bar{S}] = 0$$

$$\implies b(S) - u[S, \bar{S}] = -d_{1n} \leq 0$$

$$\implies \text{The inequality holds for this case if and only if } d_{1n} \geq 0.$$

For case 5:

$$S = N,\ b(S) = d_{1n} - d_{1n} = 0 \text{ and } u[S, \bar{S}] = 0$$

$$\implies b(S) - u[S, \bar{S}] = 0 \leq 0$$

$$\implies \text{The inequality holds for this case.}$$

Therefore we have shown that the inequality holds for all cases if and only if $d_{1n} \leq C_{1n}$ and $d_{1n} \geq 0$. This implies that the set $X^{\text{LS-C-CYCLE}} \neq \emptyset$ if and only if $0 \leqslant d_{1n} \leqslant C_{1n}$. □

We have found the conditions that LS-C-CYCLE must satisfy in order for there to exist feasible solutions to the problem. Now we show that this set of feasible solutions is in fact unbounded.

Observation 2. The set $X^{\text{LS-C-CYCLE}}$ is unbounded. For example, if $(x, y, s) \in X^{\text{LS-C-CYCLE}}$ then $(x, y, s') \in X^{\text{LS-C-CYCLE}}$ where $s'_t = s_t + 1$ for all $t \in [n]$.

As the set of feasible solutions to LS-C-CYCLE is unbounded, we are interested in whether the set of optimal feasible solutions is bounded.

Claim 2. *If LS-C-CYCLE is feasible then there exists a finite valued optimal feasible solution if and only if* $h_{1n} \geq 0$.

Proof. Assume that $h_{1n} < 0$ and we will show that an optimal feasible solution (x^*, y^*, s^*) is not finitely valued. From observation 2 we can see that $(x^*, y^*, s') \in X^{\text{LS-C-CYCLE}}$ where $s'_t = s^*_t + 1$ for all $t \in [n]$. This new s' will add h_{1n} to the objective function. As $h_{1n} < 0$ this would reduce the objective function. This contradicts our assumption and thus we have shown that if we have a finite valued optimal feasible solution then $h_{1n} \geq 0$. Now to complete the proof we must show that if $h_{1n} \geq 0$ then any optimal feasible solution will be finitely valued. Assume that $h_{1n} \geq 0$ and we have an optimal feasible solution (x^*, y^*, s^*). In this solution x^* is bounded above by C_t and y^* is bounded above by 1. Therefore the contribution to the objective function from variables x and y will be finite. As $h_{1n} \geq 0$, the variables s^* will be minimised to reduce any extra holding costs. This implies that the contribution to the objective function from variables s will also be finite. Therefore we have shown that if $h_{1n} \geq 0$ then we have a finitely valued optimal feasible solution. Combining results we have proven that LS-C-CYCLE has a finite valued optimal feasible solution if and only if $h_{1n} \geq 0$. □

In what follows we will consider the uncapacitated variant of the problem LS-U-CYCLE.

5 Structure of Bounded Optimal Feasible Solutions to LS-U-CYCLE

For this section we will assume that $h_{1n} \geq 0$ to avoid the impractical special case when the problem LS-U-CYCLE is unbounded.

We now know about the structure of feasible solutions to LS-C-CYCLE. This knowledge is valid for LS-U-CYCLE. In this section we will increase our knowledge of the problem by discovering the structure of optimal feasible solutions to LS-U-CYCLE. Suppose that we know $y \in \{0,1\}^n$, i.e. we know in which periods a set up occurs. The optimal production plan is now a minimum cost flow problem in the network of Figure 1 where the arcs (0,t) are suppressed if $y_t = 0$. Below we will recall a property of minimum cost flow networks. This property is a well known and fundamental property.

Observation 3. In an extreme feasible solution of a minimum cost network flow problem, the arcs corresponding to variables with flows strictly between their lower and upper bounds form a spanning tree.

As is the case for LS-U-PATH, this tells us something important about the structure of optimal solutions to LS-U-CYCLE.

Claim 3. *There exists an optimal solution to LS-U-CYCLE in which $s_{t-1}x_t = 0$ for all $t \in [n]$.*

Proof. Given $y \in \{0,1\}^n$, consider an optimal basic feasible solution. Suppose that $s_{t-1} > 0$. The stock flowing through arc $(t-1,t)$ corresponding to s_{t-1} must originate from production in some period k. Unlike LS-U-PATH, in LS-U-CYCLE this period k can be before or after period t. If $k < t$ then flows on arcs $(0,k),(k,k+1),\ldots,(t-1,t)$ must be positive. On the other hand if $k > t$ then the flows on arcs $(0,k),(k,k+1),\ldots,(n-1,n),(n,1),(1,2),\ldots,(t-1,t)$ must be positive. In both cases, if $x_t > 0$, the addition of arc $(0,t)$ will form a cycle. It follows from Observation 3 that $x_t = 0$. □

Now we can fully describe optimal solutions. The structure of an optimal solution is shown in Figure 2 and Figure 3. There are two cases, $t_1 = 1$ and $t_1 \neq 1$. If $t_1 = 1$ then period 1 is the first set up period, shown in Figure 2. If $t_1 \neq 1$ then period 1 is not the first set up period, this means that period t_r, the last set up period, will produce enough stock the satisfy demand in period 1 up to period $t_1 - 1$, as well as for period t_r up to period n, shown in Figure 3.

Claim 4. *There exists an optimal solution to LS-U-CYCLE characterised by:*

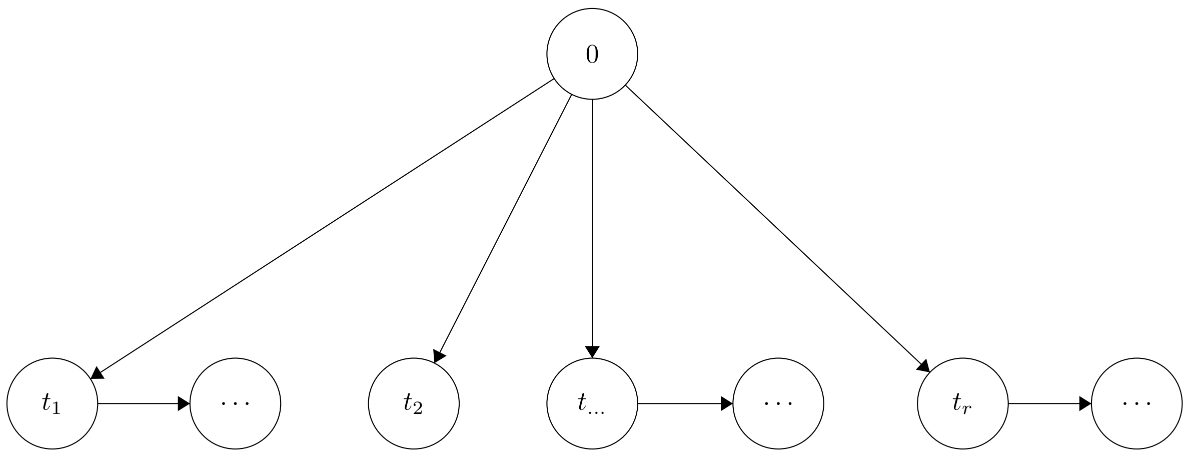

Figure 2: Structure of an optimal solution to LS-U-CYCLE, $t_1 = 1$.

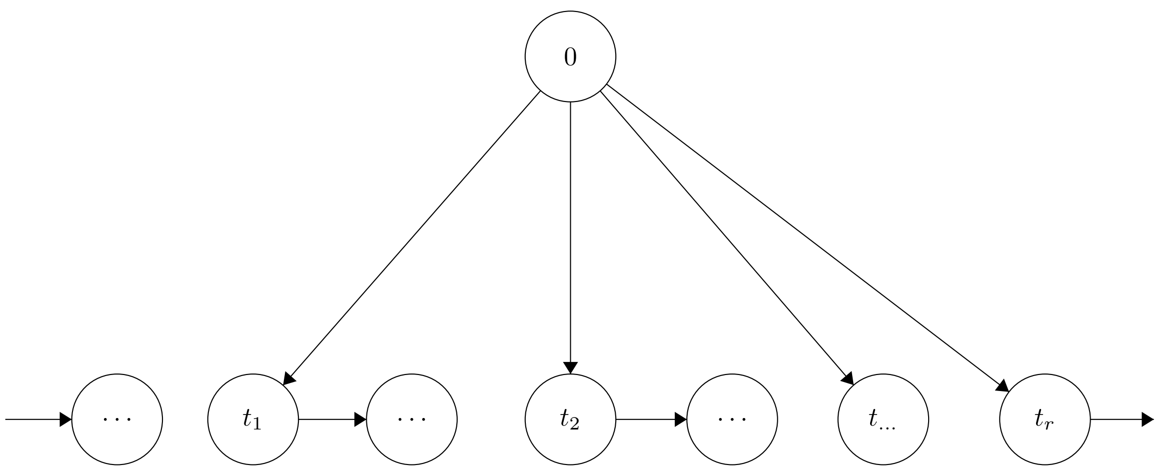

Figure 3: Structure of an optimal solution to LS-U-CYCLE, $t_1 \neq 1$.

1. *A subset of periods* $1 \leq t_1 < \ldots < t_r \leqslant n$ *in which production occurs. The amount produced in* t_j *is*

$$d_{t_j, t_{j+1}-1} \textit{ for } j \in [r].$$

2. *A subset of periods* $R \subseteq [n] \setminus \{t_1, t_2 \ldots, t_r\}$*. There is a set-up in periods* $\{t_1, t_2 \ldots, t_r\} \cup R$*.*

The intervals shown in Figures 2 and 3, $[t_1 = 1, t_2 - 1], [t_2, t_3 - 1], \ldots, [t_r, n]$ in Figure 2 and $[t_1 \neq 1, t_2 - 1], [t_2, t_3 - 1], \ldots, [t_r, t_1 - 1]$ in Figure 3, have no stock entering or leaving the interval and have production in the first period to satisfy demand for whole interval. These intervals are called *regeneration intervals* (Pochet and Wolsey, 2006). Claim 4 shows that a basic feasible solution can be decomposed into a sequence of regeneration intervals, plus possibly some additional set-ups without production.

6 Polynomial Time Algorithm for LS-U-CYCLE

In this section we will describe a polynomial time algorithm for LS-U-CYCLE that uses a dynamic programming algorithm to solve LS-U-PATH. Firstly we will show how an instance of LS-U-CYCLE can be separated into n instances of LS-U-PATH.

Observation 4. Consider the problem LS-U-CYCLE and observe that if the constraint $s_{t-1} = 0$ is imposed for some period $t \in [n]$ then the problem reduces to an instance of the LS-U-PATH problem. For example, if $s_0 = s_n = 0$ is imposed then the LS-U-CYCLE instance reduces to an instance of the classical LS-U-PATH problem with time horzion $[n]$. Similarly, if, for some $t \in [n]$, $s_{t-1} = 0$ is imposed then the LS-U-CYCLE instance reduces to an instance of LS-U-PATH with time horizon $[t, t-1]$.

We will now describe a way to separate the problem LS-U-CYCLE into n instances of LS-U-PATH. In doing so we will devise a divide and conquer mechanism to solve the problem.

Definition 2. Given an instance of LS-U-CYCLE we will let LS-U-LS-U-PATH(t) denote the instance of LS-U-PATH created when the additional constraint $s_{t-1} = 0$ is added. LS-U-PATH(1) is the classical LS-U-PATH problem with $s_0 = s_n = 0$ and time horizon $[n]$. For LS-U-LS-U-PATH(t) where $t \in [2, n]$ the time horizon is $[t, t-1]$.

From Observation 4 we can see that given an instance of LS-U-CYCLE, we can separate this problem into n distinct instances of LS-U-PATH, each of which can be solved with the dynamic programming algorithm from Pochet and Wolsey (2006).

Definition 3. We will let PATH denote an algorithm that solves an instance of LS-U-PATH using the dynamic programming algorithm from Pochet and Wolsey (2006). The output of PATH will be the pair $(z^*, (x^*, y^*, s^*))$ where z^* is the optimal objective function value, and (x^*, y^*, s^*) is the optimal solution.

Claim 3 tells us that the optimal solution to LS-U-CYCLE forms a spanning tree. In a spanning tree solution to LS-U-CYCLE there exists at least one $t \in [n]$ such that $s_{t-1} = 0$. Therefore by considering

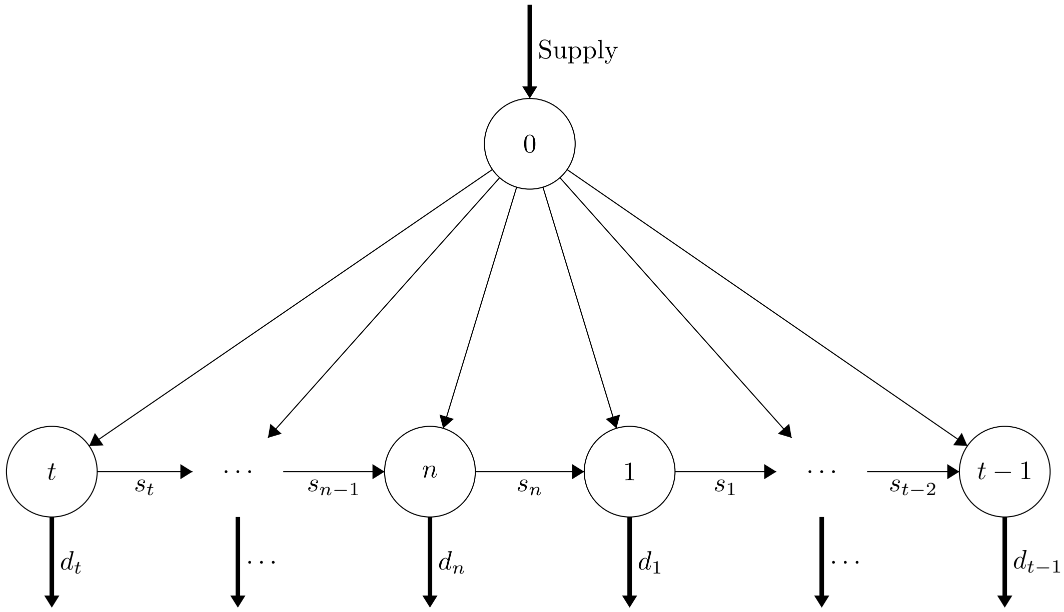

Figure 4: Network representation of LS-U-PATH(t).

all LS-U-LS-U-PATH(t), and their optimal solutions, we can find the optimal solution to LS-U-CYCLE, simply being equal to the solution PATH with smallest z^*. For the general network representation of LS-U-PATH(t), see Figure 4. For each $t \in [n]$ we will define the mapping

$$\pi_t(i) = \begin{cases} i - t + 1 + n, & \text{if } i \in [1, t-1], \\ i - t + 1, & \text{if } i \in [t, n]. \end{cases}$$

This mapping simply transforms the time horizon set $[t, t-1]$ to the set $[n]$. To solve LS-U-PATH(t) for $t \in [2, n]$, we will use this mapping. Imagine we have an instance of LS-U-PATH(t). We will map our instance data (d, p, f, h) in the following way.

$$\text{Let } (d, p, f, h)'_{\pi_t(i)} = (d, p, f, h)_i, \text{ for all } i \in [n].$$

Now the instance data $(d, p, f, h)'$ are in the correct form to be solved using the dynamic programming algorithm for LS-U-PATH. Now we can write a polynomial time algorithm CYCLE for LS-U-CYCLE, see Algorithm 6.

Claim 5. *The algorithm CYCLE finds an optimal solution to LS-U-CYCLE in $O(n^2 \log n)$ time.*

Proof. From Pochet and Wolsey (2006) we know that PATH(t) can be computed in $O(n \log n)$. Now we claim that in order to solve LS-U-CYCLE we must solve at most n instances of LS-U-PATH. Imagine we are solving LS-U-PATH(t). In the solution of LS-U-PATH(t) there may exist some $i \in [n] \setminus \{t\}$ such that $s_{i-1} = 0$. We will definte $S^{[t]} = \{i \in [n] \setminus \{t\} : s_{i-1} = 0 \text{ in solution of LS-U-PATH}(t)\}$. If $S^{[t]} \neq \emptyset$ for any $t \in [n]$ then we no longer need to solve all n instances of LS-U-PATH. If $S^{[t]} = \emptyset$ then we will need to solve all n instances of LS-U-PATH. This implies that the polynomial time algorithm for LS-U-CYCLE can be computed in $O(n^2 \log n)$. □

Algorithm 1 Polynomial Time Algorithm for LS-U-CYCLE

```
procedure CYCLE
    T ← ∅
    z* ← ∞
    for t ∈ [n] do
        if t ∉ T then
            (z', (x', y', s')) ← PATH(t)
            T ← T ∪ {i ∈ [n] \ {t} : s'_{i-1} = 0}
            if z' < z* then
                z* ← z'
                (x*, y*, s*) ← (x', y', s')
            end if
        end if
    end for
    return (z*, (x*, y*, s*))
end procedure
```

7 Valid Inequalities for LS-U-CYCLE

In this section we wish to find valid inequalities satisfied by all points in $X^{\text{LS-U-CYCLE}}$, that is, those satisfying constraints (2)-(6). In particular we are looking for valid inequalities for $X^{\text{LS-U-CYCLE}}$ that are not just obtainable as linear combinations of the constraints (2)-(6).

From Pochet and Wolsey (2006) we know that the following claim is true.

Claim 6. *Let $1 \leqslant k \leqslant l \leqslant n$, $L = [k, l]$ and $S \subseteq [k, l]$ such that $k \in S$. The (L, S) inequality*

$$s_{k-1} + \sum_{j \in L \setminus S} x_j + \sum_{j \in S} d_{jl} y_j \geqslant d_{kl},$$

is valid for $X^{\text{LS-U-PATH}}$.

We will now claim that an equivalent inequality is valid for $X^{\text{LS-U-CYCLE}}$.

Claim 7. *Let $k, l \in [n]$, $L = [k, l]$ and $S \subseteq L$ such that $k \in S$. Then the (L, S) inequality*

$$s_{k-1} + \sum_{j \in L \setminus S} x_j + \sum_{j \in S} d_{jl} y_j \geqslant d_{kl}$$

is valid for $X^{\text{LS-U-CYCLE}}$.

Proof. Consider a point $(x, y, s) \in X^{\text{LS-U-CYCLE}}$. If $\sum_{j \in S} y_j = 0$, then $\sum_{j \in S} d_{jl} y_j = 0$ and $x_j = 0$ for all $j \in S$. From this and the flow balance equations we can see that

$$\sum_{j \in L \setminus S} x_j = d_{kl} + s_k - s_{k-1}.$$

Rearranging this gives

$$s_{k-1} + \sum_{j \in L \setminus S} x_j = d_{kl} + s_k.$$

As $s_k \geq 0$ for all $k \in [n]$, this implies that

$$s_{k-1} + \sum_{j \in L \setminus S} x_j \geq d_{kl},$$

which is the inequality required. Otherwise let $u = \min\{j \in S : y_j = 1\}$. Then

$$\sum_{j \in S} x_j \leqslant \sum_{j=u}^{l} x_j \leqslant d_{ul} + s_l \leqslant \sum_{j \in S} d_{jl} y_j + s_l.$$

Now we have that

$$\sum_{j \in S} x_j \leqslant \sum_{j \in S} d_{jl} y_j + s_l.$$

Using $s_{k-1} + \sum_{j \in L} x_j = d_{kl} + s_l$ we have

$$\sum_{j \in S} x_j \leqslant \sum_{j \in S} d_{jl} y_j + s_{k-1} + \sum_{j \in L} x_j - d_{kl}.$$

This implies that

$$s_{k-1} + \sum_{j \in L} x_j - \sum_{j \in S} x_j + \sum_{j \in S} d_{jl} y_j \geq d_{kl},$$

and finally

$$s_{k-1} + \sum_{j \in L \setminus S} x_j + \sum_{j \in S} d_{jl} y_j \geq d_{kl}$$

as required. Therefore the (L, S) inequality holds for LS-U-CYCLE. □

As the (L, S) inequalities hold for LS-U-CYCLE, we can use these inequalities to describe the set of solutions to LS-U-CYCLE.

8 Conclusion

We have extended the lot sizing problem from the classical implementation on a path to a cycle. For the problem LS-U-CYCLE we have shown that many of the properties of LS-U-PATH also hold true for LS-U-CYCLE. Firstly we have introduced the problem definition, and have shown that this is a generalisation of the classical path problem. We have described the set of feasible solutions to the problem by showing that the problem is feasible if $0 \leqslant x_{1n} \leqslant C_{1n}$, the set of solutions is unbounded and the set of feasible solutions is bounded if and only if $h_{1n} \geq 0$. Similar to the problem LS-U-PATH we have shown that the structure of an optimal solution to LS-U-CYCLE forms a spanning tree. This result has been very useful throughout the report.

Once the problem was introduced and the structure of solutions described we introduced a polynomial time algorithm and valid inequalities for LS-U-CYCLE. Using a dynamic programming algorithm for LS-U-PATH and a separation of the problem LS-U-CYCLE into n instances of LS-U-PATH we have shown that, using the algorithm introduced, LS-U-CYCLE can be solved in $O(n^2 \log n)$ time. Finally we described valid inequalities for LS-U-CYCLE which used to describe the set of solutions to the problem LS-U-CYCLE.

References

R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall, 1993.

M. Di Summa and L.A. Wolsey. Lot-sizing on a tree. *Operations Research Letters*, 36:7–13, 2008.

Y. Guan and A.J. Miller. Polynomial-time algorithms for stochastic uncapacitated lot-sizing problems. *Operations Research*, 56:1172–1183, 2008.

Y. Guan, S. Ahmed, G.L. Nemhauser, and A.J. Miller. A branch-and-cut algorithm for the stochastic uncapacitated lot-sizing problem. *Mathematical Programming*, 105:55–84, 2006.

Y. Pochet and L.A. Wolsey. *Production Planning by Mixed Integer Programming.* Springer Series in Operations Research and Financial Engineering. Springer, 2006.