AMSI VACATION RESEARCH SCHOLARSHIPS 2020–21

Get a Thirst for Research this Summer

Lot Sizing on a Cycle, with Start-Up Costs

Bowen Parnell

Supervised by Hamish Waterer

University of Newcastle

February 28, 2021

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

AMSI Australian Mathematical Sciences Institute

Australian Government
Department of Education, Skills and Employment

1 Abstract

In this report the production planning problem of lot sizing on a cycle, with start-up costs is the main focus. The formulation of this problem as an optimisation problem with an objective function and constraints is first given, alongside an introduction to the problems underlying network. The properties of feasible solutions to the version of the problem with capacities is first described, which is that the total demand being considered must be at least zero and at most the total capacity for production. In the following sections a similar version of the problem is considered, made simpler by ignoring the capacities on how much can be produced. For this case the structure of its optimal solutions is shown to be a sequence of regeneration intervals wherein production in one time period satisfies the demand for some length of time. Finally a polynomial time algorithm is detailed which splits the problem into many instances of a path problem for which there already exists a polynomial time algorithm.

2 Introduction

Production planning and scheduling is a key problem for many manufacturers and supply chains. Problems of a similar form also appear in a variety of other fields. A common way to frame these types of problems is as a multi-item lot sizing problem in which a schedule of production needs to be determined for each item being produced. This schedule should ensure that the demand for each item is satisfied, subject to constraints on production, inventory and scheduling. Furthermore, the solution it proposes should be the cheapest of all options, according to however the cost or objective function has been specified. In order to better understand how to effectively solve mixed-integer programming (MIP) formulations of multi-item lot sizing problems, much of the work in the literature has focused on different variants of the single-item problem. This is because the multi-item problem is typically approached by splitting it up into many separate, single-item problems. See, for example, Pochet and Wolsey (2006) for a comprehensive review of the lot sizing literature.

A common way of describing lot sizing problems is to view them as a network design problem, wherein production decisions in each time period determine which set-up and start-up arcs exist, and what the flow of production and stock will be to satisfy demand. The classical, deterministic, single-item lot sizing problem usually has a specific type of underlying network: a path, that flows from the start to the end of the planning horizon. This project investigates a variant of this problem in which the underlying network is a cycle. Use of the cycle variant is typically motivated by longer timescale, strategic concerns where a more long term, steady state solution to the problem is desired. In this variant the problem's behaviour is assumed to wrap around on itself or display highly cyclical behaviour such that demand and other problem data repeats itself in cycles that are the length of the planning horizon. Solving just one of these cycles then gives a solution that can then be applied to all following cycles. This also means that the end of the schedule naturally and optimally connects to the next time period beyond the planning horizon, assuming no change in problem data between the cycles. As an example, imagine

a factory produces some widget, and has orders or demand for that widget that must be satisfied each month. If that demand is highly seasonal, which is often the case, such that the amount that must be provided in January is similar to what was demanded in the previous January. Then it would be a good idea to plan to have similar stock levels at the end of this year to those that would have been best for the start of this year. The explicit assumption of lot sizing on a cycle is that the ideal initial and final stock levels of the planning horizon are the same, so they are constrained to be equal, but this single value is still a decision variable free to be chosen. This patently ignores the fact that there may already be some stock at the present moment, instead focusing on what might be ideal in the future, leaving the intermediary problem of getting that initial stock. This informs how much initial stock it is desirable to have (for example when first starting a factory or when considering the long term) while making the reasonable assumption that just as much product would be wanted in the next cycle. Another benefit of this model is that it avoids possible end effects, in which a poorer quality solution is found than what could be achieved in the long term because the initial or final stock levels at either end of the planning horizon are artificially constrained to a particular stock value.

This report builds on the single-item lot sizing problem on a cycle introduced by Cooper (2018) and considers a further variation, in which start-up costs are also considered. Start-up costs are incurred whenever a new run of multiple batches of production is started. Whenever a set-up occurs in a time period but did not occur in the previous, a start-up cost must be incurred. This can be used to better model a variety of effects, like the costs of materials or lost productivity to clean out paint mixing machinery at the end of a run, changing over production back to the product being considered, and starting up and tuning machinery that aren't able to start quickly or at peak efficiency.

In this report a formulation of the problem is presented, the properties of it's feasible and optimal solutions are explored, and then used to establish the computational complexity of the problem and develop efficient algorithms for its solution.

2.1 Statement of Authorship

The results presented in this report are original, extending the work of Pochet and Wolsey (2006) on path problems with start-up costs and Cooper (2018) on cycle problems without start-up costs. The new results stem from the addition of start-up costs to the problem, and are presented with a similar structure to both works. Much of the work in this report was reached through frequent consultation with my supervisor, Hamish Waterer.

3 Problem Notation

To make referring to and distinguishing between different problems clear and concise, this report will follow a classification scheme for canonical single-item lot sizing problems observed in the literature (see, for example, Pochet and Wolsey, 2006). The scheme uses a three-field identifier PROB-CAP-VAR

(Problem-Capacity-Variant), such that LS-C denotes the lot sizing (LS) problem with capacities (C) on how much can be produced. This traditional version of the problem has an underlying network which is a path. The problem first encountered in this report is LS-C-CYCLE,SC, which adds two variants to the above, the underlying network becomes a cycle, and start-up costs (SC) are included. As Cooper (2018) noted for the case without start-ups, different versions of the problem depend on the capacity for production. In the problem introduced above the production facility has variable capacity (C) such that the upper limit on how much can be produced could differ in each time period, and is denoted LS-C-CYCLE,SC. If the capacity were instead the same in each time period, the problem would be considered to have constant capacity (CC), and would be denoted LS-CC-CYCLE,SC. If the capacity was so large, if not infinite, that all demand could be satisfied by production in a single time period, the problem would be considered uncapacitated (U), and would be denoted LS-U-CYCLE,SC.

For convenience, notation is introduced to make referring to and summing over values that exist in a number of time periods more concise. First note that n is the last time period included in the problem. Given two time periods $k, l \in \mathbb{Z}$, let $[k, l]$ denote the set $\{k, k+1, \ldots, l\}$ if $k \leqslant l$. Furthermore, if $k > l$ let it denote the set $\{k, k+1, \ldots, n\} \cup \{1, 2, \ldots, l\}$, allowing the set to wrap back around to the start as if time were in a loop or cycle. This reflects the fact that in the cycle case there is a way for events in the future to affect the past. This is an important difference from the usual path case where $[k, l] = \emptyset$ when $k > l$. A more concise notation for summing is also introduced, which makes use of this method of referring to sequences of time on a cycle. Given a parameter α indexed by the periods $t \in [n]$, let α_{kl} denote the summation $\sum_{t \in [k,l]} \alpha_t$. Note that this means the sum can also wrap around when $k > l$.

When the underlying network of the problem is referred to and described as either a cycle or a path, this is because the problem can be represented as a literal network, from which insight can be gained. As such it is useful to introduce some relevant notation to ensure work later on is understood. Note that network refers specifically to a directed graph, where arcs only allow flow in one direction. This report assumes a basic understanding of these ideas, refer to Ahuja, Magnanti, and Orlin (1993) for a more detailed explanation of graphs or network flow. As it has been noted that this problem has an underlying network it is worthwhile introducing some notation in this regard as well.

For a network $G = (N, A)$, 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 outgoing, directed arcs that start in S and end 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 $G = (N, A)$, $b(i)$ represents the supply of node i. If $b(i) > 0$ then i is a supply node and if $b(i) < 0$ then i is a demand node. Thus demand in this problem is treated as negative supply that must be fulfilled. With this understanding the following theorem can be understood, which will help aid a proof of what makes a problem feasible later in this report. It essentially means that supply must be at most equal to outgoing capacity for all subsets of the graph.

Theorem 1 (Ahuja et al. (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* $b(S) = \sum_{i \in S} b(i)$.

With some of the preliminary introductions to the notation out of the way, the problem LS-C-CYCLE,SC can be properly defined.

4 Problem definition

The capacitated, single-item lot sizing problem on a cycle, and with start up costs, is the first focus of this report. For convenience, this problem is denoted LS-C-CYCLE,SC as established in the previous section. A specific instance of the LS-C-CYCLE,SC problem consists of a number of time periods of the same length, which each have some instance specific data and decisions to be made. In each time period there is some known demand, in this case for just a single item, that must be satisfied either by producing in that time period or taking from stock built up in the past. However there is a maximum amount that can be produced in each time period, the production capacity. Even without considering costs, this may mean that periods of high demand can't be satisfied just by producing however much is demanded. Instead items can be produced in previous time periods and held between periods to satisfy later demand, this is the stock, the items leftover after satisfying demand. Whenever there is production, the machinery or production facility must be set-up to enable that production. Furthermore, with the inclusion of start-up costs, whenever there wasn't a set-up in the previous time period, and one is desired in the current period a start-up must occur. This can sometimes mean it is better to set up and produce nothing in a time period, rather than having to start up again later on.

A schedule is the amount to be produced in each time period. A feasible schedule is one wherein production is less than capacity, has only occurred when a set-up and start-up have occurred, and ensures that demand is satisfied. Each of these variables have some associated cost, and so the problem is to not only find a feasible schedule, but one that minimises the sum of fixed setup and start-up costs, and per unit production and stock holding costs.

Before presenting the full problem formulation, notation for these decisions and their associated data are introduced below. First note that anything with subscript t, like the cost of producing in a time period p_t, can differ from period to period. Since this cost often isn't constant for all time periods in a problem, it would be difficult for someone to make the best or even good production decisions. This is part of what makes optimisation worthwhile, as the optimal solution will make the absolute most of a situation that would otherwise be difficult for a production planner to tackle. Also recall that $[n]$, which itself is a shortening of $[1, n]$, is the set of time periods $\{1, \ldots, n\}$.

An instance of LS-C-CYCLE,SC consists of n time periods, and a demand d_t for a single item in each time period $t \in \{1, \ldots, n\}$. Furthermore, there are four decision variables to be determined for each time period, that together form the schedule or solution to the problem. Each decision variable has an associated cost, must be non-negative, and is subject to further constraints shown in the formulation

below and explained underneath that.

- The amount produced in a time period is denoted x_t, and incurs a variable cost per unit of p_t. There is also a limit on how much can be produced, C_t.
- Stock in inventory at the end of a time period is denoted s_t, and incurs a variable cost per unit of h_t. It is assumed that there is no limit on how much can be held in inventory.
- The decision to set-up is represented by y_t, where $y_t = 1$ indicates that a set-up must occur in that time period, and $y_t = 0$ means no set-up. Setting up incurs a fixed cost q_t. A set-up must occur if production is to occur in that time period.
- A start-up is similarly indicated by $z_t = 1$, and is otherwise zero. Starting up incurs a fixed cost g_t. A start-up must occur if a sequence of consecutive set-ups is to be started in this time period. Interestingly note that since this problem exists on a cycle, a start-up need not occur in the first time period if a set-up did occur in the last time period. Calling back to the earlier example of producing widgets for monthly demand that's the same each year, a start-up need not occur every January if a set-up is going to occur the month before in December, even if the planning horizon starts at January and ends in December.

With the notation now established the problem formulation for LS-C-CYCLE,SC can now be considered below:

$$\text{minimise} \sum_{t=1}^{n}(p_t x_t + h_t s_t + q_t y_t + g_t z_t) \tag{1}$$

subject to:

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

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

$$z_t \geqslant y_t - y_{t-1} \qquad t \in [n] \tag{4}$$

$$z_t \leqslant y_t \qquad t \in [n] \tag{5}$$

$$z_t \leqslant 1 - y_{t-1} \qquad t \in [n] \tag{6}$$

$$s_0 = s_n \tag{7}$$

$$y_0 = y_n \tag{8}$$

$$x_t,\ s_t \geqslant 0, \qquad t \in [n] \tag{9}$$

$$y_t, z_t \in \{0, 1\}, \qquad t \in [n] \tag{10}$$

The objective function (1) of this optimisation problem is to minimise the total cost of the schedule, the sum of production, holding, set-up and start-up costs. Constraint (2) is the inventory balance constraint that ensures the stock from the previous time period and amount produced is equal to the

demand and stock. Constraint (3) is the set-up and capacity constraint that ensures $y_t = 1$ if there is production in that period, and that production is less than capacity. Constraint (4) is the start-up constraint that ensures $z_t = 1$ if a set-up occurs in that time period but didn't occur in the previous. Note that the use of y_{t-1} requires defining y_0, the set-up that occurs before the first time period, which is done in (8). Constraints (5-6) ensure that a start-up only occurs at the start of a sequence of consecutive set-ups, when a negative cost could otherwise encourage start-up at an incorrect time, like when a set-up had occurred in the previous time period or hadn't occurred in the current time period. Interestingly, note that a start-up need not occur at all if a set-up has occurred in every time period. This is because the problem exists on a cycle, and y_{t-1} in the start-up constraints can wrap back around in the first time period when $t = 1$ such that $y_{1-1} = y_n$. Constraint (7) is the cycle constraint that ensures the stock in the last period s_n is carried back around so its available in the first time period. Constraint (8) is the additional cycle constraint relevant to this problem with start-up costs. It ensures that set-up in the last period y_n can wrap around to maintain a sequence of consecutive set-ups and potentially avoid needing a start-up in the first time period. The addition of start-up costs and their constraints requires defining y_0, which is normally zero in the path case. Constraint (9) ensures the amount produced and held in stock isn't negative. Constraint (10) ensures the set-up and start-up decision variables are binary.

Notice that, just as Cooper (2018) observed for the case without start-ups, the following observation can be made.

Observation 1. LS-C-CYCLE,SC is a generalisation of LS-C-PATH,SC since the path case requires $s_0 = s_n = 0$.

This observation is drawn from the work of Pochet and Wolsey (2006), who show that even if the path problem is first approached with $s_0 \neq 0$ and/or $s_n \neq 0$, the problem can be reformulated with modified demands so that the initial and final stocks are zero. This is a fundamental difference to the cycle case, where the final stock level, that flows between cycles, is a decision variable. The cycle case is solving for the ideal plan for a cycle, not necessarily what should be done in the next time period.

Pochet and Wolsey (2006) in their work with the LS-C-PATH case, and Cooper (2018) in his work with the LS-C-CYCLE case both realised that the set-up costs, q_t, could be assumed to be non-negative. A similar assumption cannot be made for the start-up case, for the set-up costs q_t, or start-up costs g_t. This assumption worked for those simpler cases because if the cost of setting up was negative $q_t < 0$ it is trivially optimal to set up $y_t = 1$. However the set-up isn't forced with $y_t = 1$ in such cases until after a solution has been found, to avoid affecting the structure of the solution. Instead any negative set-up costs are first set to zero in a pre-processing step, making the assumption of non-negativity possible, and once a solution is found all such set-ups would be forced on. With LS-C-CYCLE,SC the decision to start up or set up in a time period forces other decision variables on or off as well. A set-up must also occur for a start-up to occur, and a start-up or preceding set-up must occur for a set-up to occur. The knock-on effects of these decisions, such that neighbouring start-ups can no longer occur, affects how demand might be satisfied and the value of the objective function.

It seems that there could still be a negative enough cost that starting up or setting up would always be optimal, such that it could outweigh whatever complicated effects or opportunity costs a start-up or set-up might cause. However this is overly complicated and brings little value, as it is Pochet and Wolsey (2006) and Cooper (2018) don't seem to make use of this assumption, and no such similar one appears in literature on start-up costs. Being able to make this assumption may make finding solutions to the problem easier or faster, by essentially simplifying the problem such that highly negative start-up or set-up costs need not be considered. Another complication is that even if one were to replace the costs with zero or some negative value, those costs could only apply if the solution didn't preclude the possibility of being able to start-up in the required time period. So the start-ups and at least their immediate set-ups would have to be forced on before solving the problem, likely disturbing the structure of the optimal solution and preventing it from taking the extreme form discussed in later sections.

As has already been mentioned, LS-C-CYCLE,SC has an underlying network, specifically one that wraps around or exists on a cycle. It is common, and helpful, to examine the network of lot sizing problems. This problem, and many others, can be re-framed as a network design problem, wherein one can choose which arcs should exist, representing the decision to start up or set up in a time period. Once these arcs are chosen, the problem on the network that has been designed becomes that of a minimum cost network flow problem, wherein supply must flow to satisfy the demand in each time period as cheaply as possible. Though these lenses the problem can be better understood and solved.

Before seeing this network in Figure 1 some introduction is worthwhile. Much like other directed graphs or networks, this one shows flow travelling from a source or supply node, through arcs and intermediary transshipment nodes, to the sink or demand nodes. Each column of nodes in the network represent a time period, while each row of nodes represents each of the decisions to be made in a time period, first whether to start up, then whether to set-up, and finally how much to produce. The supply node, 0, is an artificial inclusion with supply d_{1n} at the very top of this hierarchy, it represents all the product that must be produced to satisfy demand for each time period. The start-up and set-up decision variables represent the decision to include an arc in the solution to the problem, and make up the design portion of the network design problem. They are coloured in Figure 1 to highlight this distinction. In any solution only some of these arcs would exist. To represent the fact that a start-up need not occur if a set-up already occurred in the previous time period, there are permanent arcs, labelled a_t, that connect the start-up and set-up nodes. This allows supply to flow to the current time period without the start-up arc, as long as the previous set-up arc exists.

The supply of node i, is denoted $b(i)$ as was introduced earlier and is here represented by the thicker arrows, labelled with some demand d_t, that only connect to one node. If they flow into a node, this is a positive supply $b(i) > 0$, while if they flow out this is a negative supply $b(i) < 0$ or a demand that must be satisfied. Only node 0 has positive supply. The bottom level of production nodes have negative supply, their demand that must be fulfilled. The intermediary, transhipment nodes through which the flow passes have zero supply, $b(i) = 0$, since the supply just flows through them. A cut that partitions

the graph into subsets of nodes S and $\bar{S}$ is so called because it draws a line through or cuts through arcs to do so. The outgoing capacity of this partition, $u[S, \bar{S}]$, is the sum of capacities on the outgoing arcs from S, arcs which start but don't end in S, those that have been cut. Note that only the flow on the arcs from set-up nodes to production nodes, which will have a flow of the amount produced x_t, actually have a capacity, one already defined as C_t. The flow on all other arcs have no defined capacity, or equivalently infinite capacity. The cyclical nature of this network is visible in the arcs that seem to flow out of the network on the right and into it on the left. It isn't clearly visible in Figure 1 but they are the same arc on each level, these two arcs a_n and s_n connect the last and first time periods. This represents how initial stock and set-up values for time $t = 0$ come from their final values where $t = n$. With the network thoroughly introduced it can be seen and understood in Figure 1.

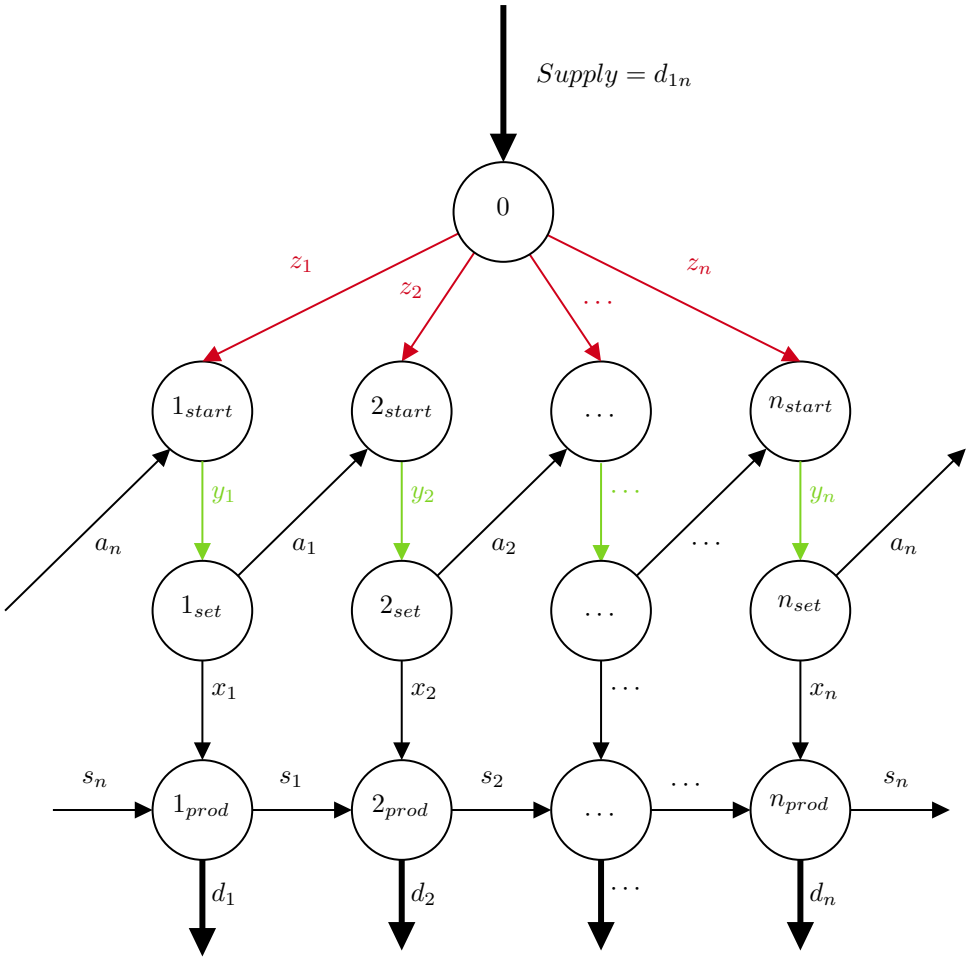

Figure 1: A network representation of LS-C-CYCLE,SC, as a network design problem.

Suppose the values of y and z are known, such that it is known when a set-up or start-up occurs. The optimal production plan is then a minimum cost flow problem in the network of Figure 1 where the coloured arcs $(0, t_{start})$ or (t_{start}, t_{set}) are suppressed if $z_t = 0$ or $y_t = 0$ respectively.

Now that the problem, its notation, formulation as an optimisation problem, and representation as a network have been firmly established, a better understanding of this problem, and how to solve it can be developed.

5 Properties of Feasible Solutions to LS-C-CYCLE,SC

In this section the focus will be on understanding what makes an instance of the problem LS-C-CYCLE,SC feasible, or equivalently what properties its feasible solutions have. Cooper (2018) established some properties of feasible solutions to the similar problem LS-C-CYCLE. Since the network with start-up costs is larger and more complex it seems necessary to confirm these properties still hold. In so proving that similar properties hold for the problem with start-up costs, a more concise proof has also

been discovered. Let $X^{LS-C-CYCLE,SC} = \{(x, y, s, z) \in \mathbb{R}^n \times \{0,1\}^{n+1} \times \mathbb{R}^{n+1} \times \{0,1\}^n : (2-10)\}$ denote the set of feasible solutions to the formulation of LS-C-CYCLE,SC. For a given problem to be feasible, such that feasible solutions exist and $X^{LS-C-CYCLE,SC} \neq \emptyset$, certain conditions must be met. Determining these conditions can provide insight into the properties of feasible solutions.

According to Theorem 1, introduced earlier, since this problem is a minimum cost network flow problem once the network has been designed, it only has a feasible solution if, for every subset of the nodes that make up the network, the supply is at most equal to the outflowing capacity.

Observation 2. For any subset S that has at least one outgoing arc with infinite capacity, $u[S, \bar{S}] = \infty$ and the inequality from Theorem 1, $b(S) - u[S, \bar{S}] \leq 0$, is trivially satisfied.

With this observation it will be much simpler to draw insights from our problems interaction with Theorem 1.

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

In Claim 1, the conditions under which an instance of LS-C-CYCLE,SC has feasible solutions are stated, and proved with the assistance of Theorem 1 below.

Proof. First, note that the supply of node 0 is by definition equal to the sum of all demands. This can be shown by summing the network flow constraint (2) for all $t \in [n]$. This gives $s_0 + x_{1n} = d_{1n} + s_n$. Since $s_0 = s_n$ from the cycle constraint (7), the stock variables can be removed leaving $x_{1n} = d_{1n}$. In the network representation the flow on the x_t arcs must originate from the supply of node 0. By summing the flow balance constraints for the t_{set} and t_{start} nodes, which have not been introduced but similarly ensure the flow into a node is equal to the flow out, it can be shown that $b(0) = d_{1n}$.

Now according to Theorem 1, for every subset of nodes in the network $S \subseteq N$, where N is all the nodes in the network, the constraint $b(S) - u[S, \bar{S}] \leq 0$ must hold for the network flow problem to have a feasible solution. Rather than writing out and checking the constraint for every subset, note that there are a limited number of ways in which these subsets can be taken for them to give different results. Meaning that the inequality $b(S) - u[S, \bar{S}] \leq 0$ may hold, in the same way, for many seemingly different subsets. This is most evident in the case where a subset has infinite outgoing capacity, since all such subsets are trivially satisfied, as was noted in Observation 2. Only a few subsets with finite outgoing capacity are left that still need to be considered.

1. $S = \{0, 1_{start}, \ldots, n_{start}, 1_{set}, \ldots, n_{set}\}$, all of the supply, start-up and set-up nodes.
2. $S = \{1_{prod}, \ldots, n_{prod}\}$, $S = \{1_{prod}, \ldots, n_{prod}, 1_{set}, \ldots, n_{set}\}$, or

   $S = \{1_{prod}, \ldots, n_{prod}, 1_{set}, \ldots, n_{set}, 1_{start}, \ldots, n_{start}\}$, all production nodes in the cycle and optionally all the nodes in the layer or two layers above it.
3. $S = N$, all nodes are included in S.

From these we can determine the conditions under which the inequality from Theorem 1, $b(S) - u[S, \bar{S}] \leqslant 0$ is satisfied.

For case 1:

$$\begin{gathered} S = \{0, 1_{start}, \ldots, n_{start}, 1_{set}, \ldots, n_{set}\},\ b(S) = d_{1n} \text{ and } u[S, \bar{S}] = C_{1n} \\ \implies b(S) - u[S, \bar{S}] = d_{1n} - C_{1n} \leqslant 0 \\ \implies \text{The inequality holds for this case if and only if } d_{1n} \leqslant C_{1n}. \end{gathered}$$

For case 2:

$$\begin{gathered} S = \{1_{prod}, \ldots, n_{prod}\},\ S = \{1_{prod}, \ldots, n_{prod}, 1_{set}, \ldots, n_{set}\}, \text{ or} \\ S = \{1_{prod}, \ldots, n_{prod}, 1_{set}, \ldots, n_{set}, 1_{start}, \ldots, n_{start}\},\ b(S) = -d_{1n}, \text{ and } u[S, \bar{S}] = 0 \\ \implies b(S) - u[S, \bar{S}] = -d_{1n} - 0 \leqslant 0 \\ \implies \text{The inequality holds for this case if and only if } d_{1n} \geqslant 0. \end{gathered}$$

For case 3:

$$\begin{gathered} S = N,\ b(S) = d_{1n} - d_{1n} = 0, \text{ and } u[S, \bar{S}] = 0 \\ \implies b(S) - u[S, \bar{S}] = 0 - 0 \leqslant 0 \\ \implies \text{The inequality holds for this case.} \end{gathered}$$

Therefore Theorem 1 holds for all cases if and only if $d_{1n} \leqslant C_{1n}$ and $d_{1n} \geqslant 0$. This implies that the set $X^{LS-C-CYCLE,SC} \neq \emptyset$ if and only if $0 \leqslant d_{1n} \leqslant C_{1n}$. □

The conditions have been found that an instance of LS-C-CYCLE,SC must satisfy for there to exist feasible solutions to the problem. Now it can be shown that the set of feasible solutions is unbounded. The following results were shown by Cooper (2018) for the similar problem without start-up costs LS-C-CYCLE. For completeness these are restated in the context of this problem, LS-C-CYCLE,SC.

Observation 3. The set $X^{LS-C-CYCLE,SC}$ is unbounded. For example, if $(x, y, s, z) \in X^{LS-C-CYCLE,SC}$ then $(x, y, s', z) \in X^{LS-C-CYCLE,SC}$ where $s'_t = s_t + 1$ for all $t \in [n]$.

This is entirely because s_0 or s_n is a decision variable, that could be chosen to be infinitely large. While the initial stock s_0 could be used to satisfy demand it would have to be replenished by production to ensure $s_n = s_0$. As such the schedule for production would be quite similar, but could have any amount of stock passing between time periods.

As the set of feasible solutions to LS-C-CYCLE,SC is unbounded, it would be valuable to establish if at least the set of optimal, feasible solutions is bounded. Cooper (2018) already proved a similar claim to the following, for the case without start-up costs.

Claim 2. *If LS-C-CYCLE,SC is feasible then the optimal feasible solution is finite valued if and only if* $h_{1n} \geqslant 0$.

If $h_{1n} < 0$ then it would be optimal to have as much stock flowing through the network as possible. Since there is no limit to how much stock could be chosen to flow through the network, it would be optimal to have an infinite amount of stock.

In what follows we will consider the uncapacitated version of the problem, LS-U-CYCLE,SC. This means there is no restriction on how much can be produced in any time period. Equivalently C_t is assumed to be infinity or at least greater than the sum of all demands, $C_t \geqslant d_{1n}$, as previously introduced. This is a simplification of the problem that will make gaining further insights easier.

6 Structure of Bounded, Optimal, Feasible Solutions to LS-U-CYCLE,SC

Some knowledge of the properties of feasible solutions to LS-C-CYCLE,SC has been developed and is still valid for LS-U-CYCLE,SC. This includes Claim 2, so it's assumed that $h_{1n} \geqslant 0$ to avoid the impractical special case where the optimal solutions are unbounded. In this section knowledge of the cyclical problem with start-up costs can be developed by discovering the structure of optimal, feasible solutions to the uncapacitated problem.

To help understand the work in this section let the definition of an acyclic graph be recalled. An acyclic graph is a graph that contains no cycles. A cycle is a path, a connected sequence of nodes, where the first and last nodes are also connected with an arc. For example, the path $0 - 1_{start} - 1_{set} - 2_{start}$, together with the arc $(0, 2_{start})$. Some definitions like those given by Ahuja et al. (1993) state that an acyclic graph only contains no directed cycles. Let it be made clear that the definition of an acyclic graph used here excludes any cycle, including undirected cycles. Furthermore an acyclic graph can still include the cyclical behaviour that gives this problem its name, such that the n nodes can connect to the 1 nodes. With this understanding developed, a fundamental property of minimum cost network flow problems can be stated below.

Observation 4 (Pochet and Wolsey (2006))**.** 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 an acyclic graph.

Note that in this newly uncapacitated problem the upper bounds are infinite, while the lower bounds are still zero. So this means if all the arcs with no flow are ignored or suppressed in an extreme feasible solution, this forms an acyclic graph. As is the case for LS-U-PATH and LS-U-CYCLE, shown by Pochet and Wolsey (2006) and Cooper (2018) respectively, this provides an important insight about the structure of optimal solutions to LS-U-CYCLE,SC.

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

The proof given by Cooper (2018) still holds for the case with start-up costs, and it should be noted extends beyond that given by Pochet and Wolsey (2006) since the inclusion of the cycle constraint allows

demand in a time period to be satisfied by producing before or after and then holding in stock until then. Essentially since production is uncapacitated, demand will be completely satisfied by either producing now, or producing at some other time and holding, whichever is cheapest. There is no reason not to satisfy all of a time periods demand with the cheapest of these two options.

Even when a set-up or start-up could occur without production, because of negative costs or to avoid a later start-up, the extreme feasible solution will still form an acyclic graph. The constraints on start-ups ensure they cannot occur after a set-up, preventing a loop from forming. Even if a sequence of a start-up and consecutive set-up(s) occurs without any production, the resulting network will still be acyclic and the demand in those time periods can be satisfied by stock. Thus it can be seen that any optimal or extreme feasible solution will indeed form an acylic graph. This structure of an extreme feasible solution has now been described and an example of such a solution is shown in Figure 2 below.

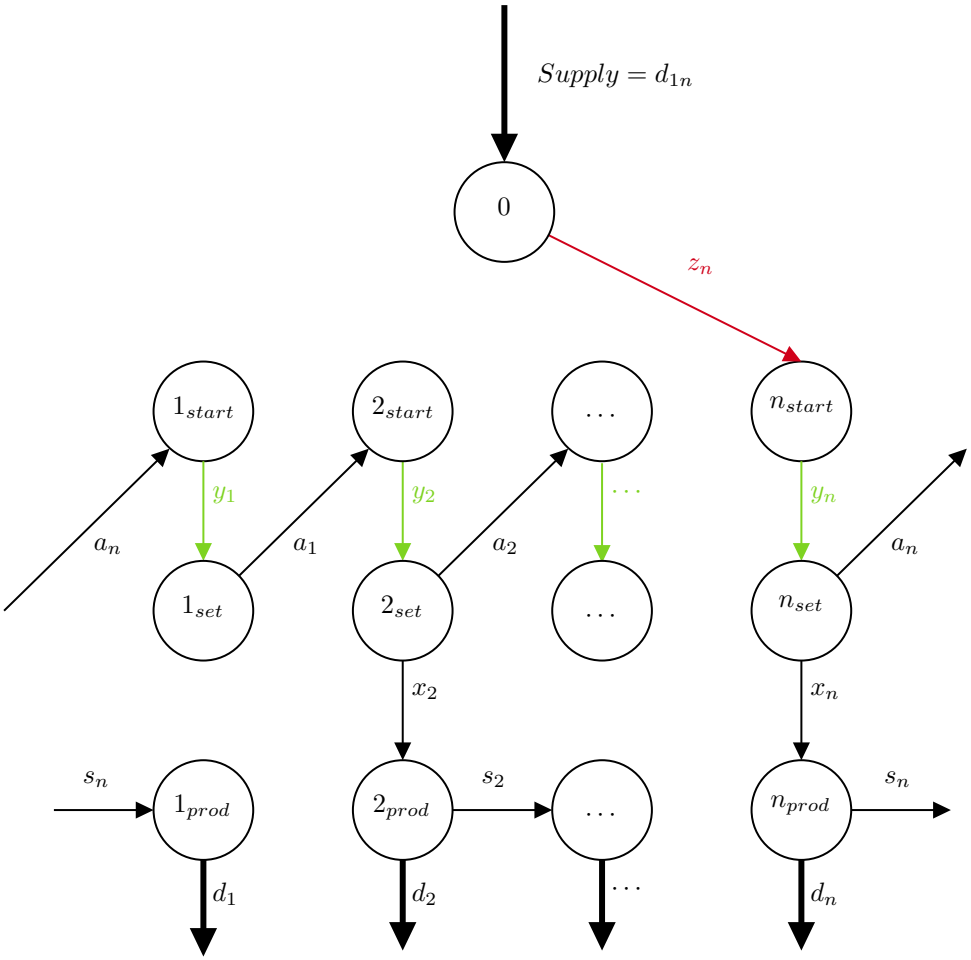

Figure 2: Structure of an extreme, feasible solution to LS-U-CYCLE,SC.

The example of an extreme, feasible solution given in Figure 2, which includes the optimal solutions of interest to this report, illustrates a number of important features of the structure of such solutions. The cyclic nature of this problem can be seen in both the s_n and a_n arcs that wrap around the time horizon

from period n to period 1. In this solution the flow of supply that will satisfy demand all travels through the z_n and y_n arcs and then splits at n_{set}. The flow in arc x_n or the amount produced is $x_n = d_n + d_1$, which makes use of the cyclic stock arc to satisfy demand in the first time period by producing in the last time period. Supply for all other demand flows through a_n and is realised by production in at least time period 2 and possibly other times. Here we can see an example of the case where it might be optimal to set-up but not produce, as in time period 1. Since production occurs in the next time period it must have been cheaper to set-up then, rather than start-up in period 2. If it is assumed that no other production occurs except in periods n and 2, then time period 3 and possibly more that were condensed into the ellipsis node serve as an example of setting up and not producing simply because the cost is negative. These set-ups do not prevent a future start-up, and it is assumed they don't enable production, so they would occur simply because they have a negative cost. What is not shown here is a similar case where a start-up and a sequence of set-ups could occur without any production, simply because the total cost of doing so is negative enough.

With this understanding of the structure of an optimal solution a similar claim can be made to that made by Pochet and Wolsey (2006) and Cooper (2018) for LS-U-PATH and LS-U-CYCLE respectively. Note that some potentially confusing notation is used here, so a brief explanation is first given. Recall that the time periods are denoted $1, 2, \ldots, n$, and here a subset of these periods $\{l_1, l_2, \ldots, l_r\} \subseteq [n]$ is taken and given a subscript to denote the order they occur in. The subscript does not indicate the time period, but whether this is the first, second, j^{th}, or last time that production occurs. If production occurred in every time period, then the subset would have length n with subscripts $[n]$, while if production only occurred once there would only be one value with subscript 1.

Claim 4. *There exists an extreme, feasible solution to LS-U-CYCLE,SC characterised by:*

1. *A subset of time periods $\{l_1, l_2, \ldots, l_r\} \subseteq [n]$ where $1 \leqslant l_1 < \cdots < l_r \leqslant n$ in which production occurs. The amount produced in period l_j is the sum of demand from that period up to the period before whenever production occurs again. As this problem exists on a cycle, when production occurs for the r^{th} and final time, the amount produced will cover all demand until the first time production occurs. Each sequence of periods of demand covered by a single production can be denoted $[l_j, l_{j+1} - 1]$ for $j \in [r]$, recalling that these are the times production occurs up until the last and r^{th} time. The amount produced is $x_{l_j} = d_{l_j, l_{j+1}-1}$ for $j \in [r]$ and $l_j \in \{l_1, l_2, \ldots, l_r\}$. There is no production in other periods, such that $x_t = 0$ for $t \in [n] \setminus \{l_1, l_2, \ldots, l_r\}$. Note that to enable this production set-up must also occur in all of these time periods $\{l_1, l_2, \ldots, l_r\}$.*

2. *A subset of time periods that see no production $R \subseteq [n] \setminus \{l_1, l_2, \ldots, l_r\}$ but still have a set-up occur, such that there is a set-up in periods $\{l_1, l_2, \ldots, l_r\} \cup R$. Thus R consists of any time periods where it is optimal to set-up but not produce, such as to avoid a future start-up cost. The first time period of each sequence of consecutive set-ups, $\{l_1, l_2, \ldots, l_r\} \cup R$, will also have a start-up.*

Note that if $l_1 = 1$ production occurs in the first time period, so there must be no stock wrapping

around such that $s_0 = s_n = 0$. The last interval would be from l_r to $l_1 - 1 = (1) - 1 = n$. If there was also no set-up in the last time period, so $l_r \neq n$, then no set-up would occur in n and the a_n variable wouldn't wrap around either. This would make the problem the same as the traditional path problem, as no stock or set-ups would wrap around, such that $s_0 = s_n = 0$ and $y_0 = 0$.

The extreme feasible solution, an example of which is shown in Figure 2, is comprised of *regeneration intervals* (Pochet and Wolsey, 2006). These intervals are consecutive sequences of time periods wherein demand is satisfied by production at the start of the interval, in the example shown in Figure 2 the regeneration intervals are $[l_1 = 2, l_2 - 1], \ldots, [l_r = n, l_1 - 1 = 1]$. Each of these intervals has no stock entering or leaving it, and have production in the first period to satisfy demand for the whole interval. In the network they appear as connected production nodes. Note that the interval of time being focused on, $[l_j, l_{j+1} - 1]$, is when demand is being satisfied, such that a start-up and set-ups can occur in periods outside of this interval as can be seen in Figure 2. Claim 4 shows that an optimal solution can be decomposed into a sequence of regeneration intervals, plus possibly some additional time periods with set-ups but no production. Each of the time periods where production occurs $\{l_1, l_2, \ldots, l_r\}$ described in Claim 4 marks the start of a regeneration interval.

7 Polynomial Time Algorithm for LS-U-CYCLE,SC

In this section a polynomial time algorithm for LS-U-CYCLE,SC is described that uses an already existing dynamic programming algorithm for the solving of LS-U-PATH,SC. An instance of LS-U-CYCLE,SC can be solved by splitting into n^2 instances of LS-U-PATH,SC and a single transportation problem or a constrained instance of LS-U-CYCLE.

Observation 5. Consider the problem LS-U-CYCLE,SC and observe that if the constraints $z_k = 1$ and $x_l > 0$ are imposed for some period $k = l \in [n]$ then the problem reduces to an instance of the LS-U-PATH,SC problem. For example, if $z_1 = 1$ and $x_1 > 0$, then the preceding set-up must be zero $y_n = 0$ from Constraint 6 and the preceding stock variable must be zero $s_0 = 0 = s_n$ from Claim 3. The LS-U-CYCLE,SC instance with these constraints reduces to an instance of the classical LS-U-PATH,SC problem with time horizon $[n]$. Similarly, if $z_k = 1$ and $x_l > 0$ are imposed for some $k = l \in [n]$, then this LS-U-CYCLE,SC instance reduces to an instance of LS-U-PATH,SC with time horizon $[k, l - 1]$.

This is the natural extension of an observation made by Cooper (2018) for the case without start-ups. The LS-U-CYCLE problem could be broken up into n different LS-U-PATH problems, as there were only n different times that production could occur and no need to consider start-ups. The example extreme feasible solution seen in Figure 2 could be arrived at with the constraints $z_n = 1$ and $x_n > 0$ applied to the cycle case such that it reduces to a path problem. In that example $k = l = n$ is the start of a regeneration interval, which also had a start-up occur in the same time period. However for the LS-U-CYCLE,SC problem, Observation 5 doesn't cover all the possible ways it can be broken up. A start-up at the beginning of an interval, such that it enables the production at the start of a regeneration

interval, can occur in a time period before the actual production occurs. This will just require set-ups when the start-up occurs, in any intermediary time periods, and when production occurs at the start of the regeneration interval. It is certainly possible for this to occur in the cycle case, but can't be captured by any of the n instances of LS-U-PATH,SC described in Observation 5.

Let $k \in [n]$ denote a time period when start-up occurs and $l \in [n]$ denote a time period when production subsequently occurs. If $z_k = 1$ and $x_l > 0$ is imposed when $k \neq l$, then the way in which this cycle instance reduces to a path instance is not so clear. To reduce the cycle problem of this form, it must first be condensed into a shorter time horizon, without the time periods $[k, l-1]$. This can be done in such a way that start-up and production occur in the same time period, in this condensed horizon, which allows the problem to be reduced to an instance of the path problem.

When $k \neq l$, there are some time periods from start-up to before production $[k, l-1]$ when production does not occur and so demand must be satisfied by stock in their preceding time period s_{k-1}. This demand can be shifted to the demand in period $k-1$ such that $d'_{k-1} = d_{k-1,l-1}$, without changing the solution. Where d' and any other parameters later denoted with an apostrophe $'$ means the new or updated value of the parameter. The constant holding cost $H_{k,l-1} = \sum_{t=k}^{l-1} h_{t-1} d_{t,l-1}$ can be added to the objective function to ensure it has the right value.

Furthermore, the cost of starting up in period k and setting up until period l can be shifted to the cost of starting up in period l. The new cost of starting up would then be $g'_l = g_k + q_{k,l-1}$. With these two changes, the time periods $[k, l-1]$ when $k \neq l$ can be removed, for any $k, l \in [n]$.

The problem defined by having a start-up in period k and production occurring in period l, once these changes have been made, now has start-up and production occuring on period l and only has a time horizon from l to $k-1$. This reformulation of the problem then reduces to an instance of the classical path problem in much the same way as it does in Observation 5 when $k = l$.

The especially keen reader may have noticed one other, rather unique, case not already covered. If set-up occurs in every time period, then start-up need not occur at all. This unique possible solution to the LS-U-CYCLE,SC problem could not be reached through any of the reductions to path problems mentioned thus far. One way of dealing with this case is reducing the problem to an instance of LS-U-CYCLE where all set-ups are forced on. As Cooper (2018) described, the LS-U-CYCLE problem is itself solved by reducing it to n different instances of the path problem. This specific problem, of just determining how much to produce without any concern for set-ups and start-ups, can more accurately be called a transportation problem. In this report, this problem will be treated as a constrained instance of the LS-U-CYCLE problem for convenience, but it should be noted that handling it as a transportation problem may provide a faster method to solve the problem.

This leaves three different ways for the problem to reduce to an instance of a more easily solved problem. When $k = l$ the problem reduces to an instance of the LS-U-PATH,SC problem with a time horizon of the same length, just shifted to $[k, k-1]$. When $k \neq l$ the problem must first be condensed to the sequence $[l, k-1]$ and then this reduces to an instance of the LS-U-PATH,SC problem on a

shorter horizon. When a start-up doesn't occur at all the problem can be reduced to a transportation problem or a constrained instance of LS-U-CYCLE, whichever is easier. To help make referring to these different cases more clear and concise, and following the example set by Cooper (2018), some notation is introduced below. With this ability to separate the problem into many instances of more easily solved problems a divide and conquer approach can be devised to solve the cycle problem.

Definition 1. Given an instance of LS-U-CYCLE,SC with the additional constraints $z_k = 1$ and $x_l > 0$, let LS-U-PATH,SC(k, l) denote the instance of LS-U-PATH,SC the cycle problem reduces to. LS-U-PATH,SC$(1, 1)$ is the classical LS-U-PATH,SC problem arrived at when production and start-up are assumed to occur in the first time period. This would force $s_0 = s_n = y_0 = 0$ in the cycle case, which makes it equivalent to a path problem on the time horizon $[n]$. For LS-U-PATH,SC(k, l) for $k, l \in [n]$ when $k = l$ the time horizon is $[k, k-1]$ while when $k \neq l$ the time horizon is $[l, k-1]$.

To see the network of an instance of LS-U-PATH,SC(k, l), where $k \neq l$, before it has been condensed to the shorter time horizon, see Figure 3.

It is now possible to separate the problem LS-U-CYCLE,SC into one constrained instance of LS-U-CYCLE and n^2 instances of LS-U-PATH,SC, where there are n sets of n instances that have lengths from the set $[n]$. Such that n instances have horizon length 1, another n instances have horizon length 2, all the way up to the last set of n instances that have horizon length n. Each of these instances of a path problem can be solved in polynomial time with an already existing dynamic programming algorithm from Pochet and Wolsey (2006). The constrained LS-U-CYCLE problem can also be solved in polynomial time with the algorithm by Cooper (2018) with the additional constraints that all set-ups must occur.

Definition 2. Let PATH(k, l) denote an algorithm that solves the instance LS-U-PATH,SC(k, l) using the dynamic programming algorithm from Pochet and Wolsey (2006), where $k, l \in [n]$. The output of PATH(k, l) will be the pair $(v^*, (x^*, y^*, s^*, z^*))$ where v^* is the optimal objective function value, and (x^*, y^*, s^*, z^*) is the optimal solution. It is assumed that the algorithm maps the time horizon to the path case, as discussed below, and that if $k \neq l$ it computes and assigns the new values g'_l and d'_{k-1}.

As noted in Observation 4, the extreme, feasible solution to LS-U-CYCLE,SC forms an acyclic graph. This includes the optimal solution being searched for here. In such a solution to LS-U-CYCLE,SC there exists at least one pair of $k, l \in [n]$ such that $z_k = 1$ and $x_l > 0$. Other than the one exception where there are no start-ups and set-ups occur in every time period, which can still form an acyclic graph. By considering all instances LS-U-PATH,SC(k, l) and their optimal solutions, this event where the cycle is cut at least once, will be captured in one of them. The optimal solution to LS-U-CYCLE,SC will simply be equal to the solution PATH(k, l) with smallest optimal objective function value, v^*.

For each $k, l \in [n]$ the following mapping can be defined:

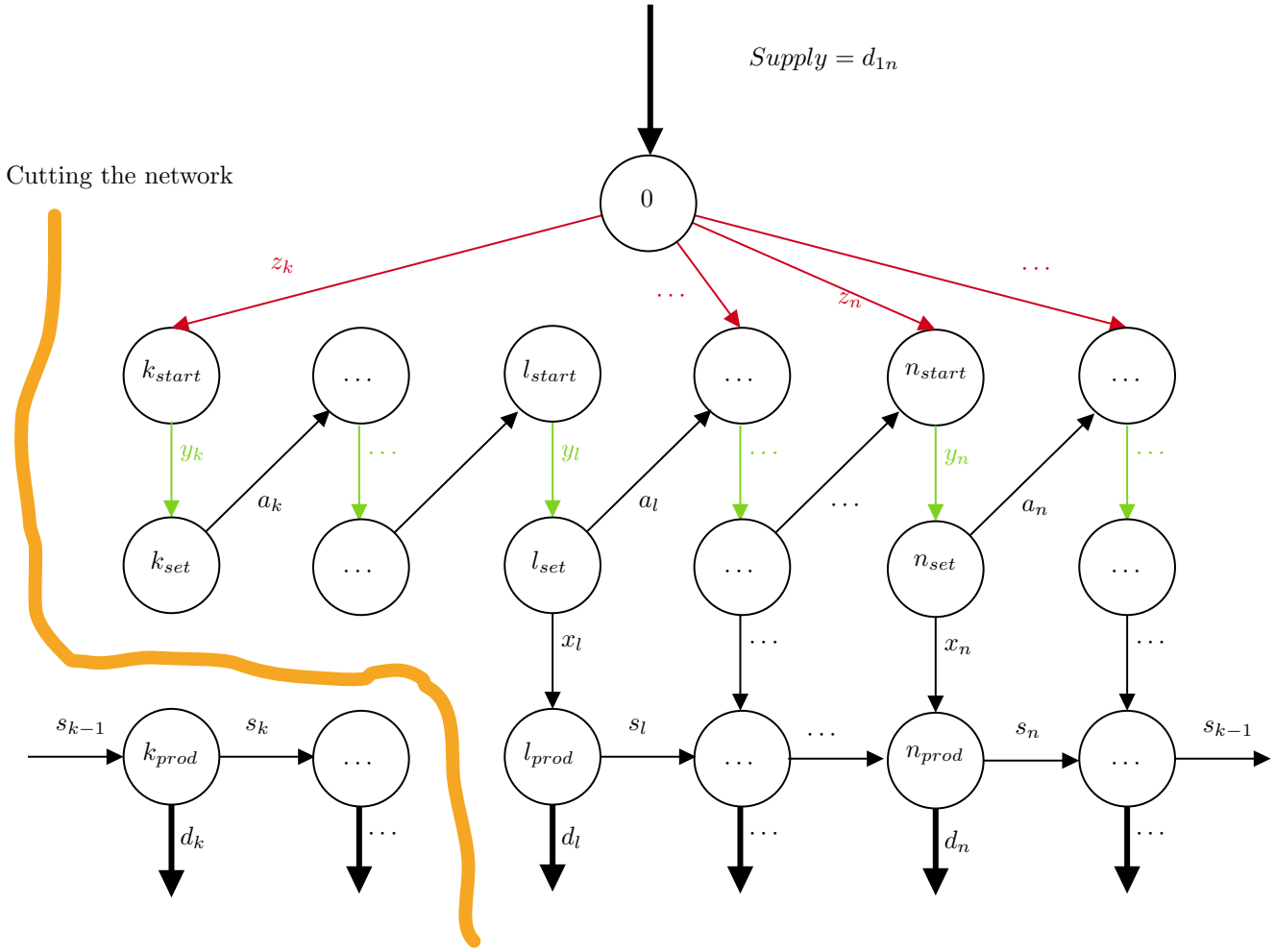

Figure 3: Network representation of LS-U-PATH,SC(k,l) before it has been condensed to the shorter time horizon.

$$\pi_{k,l}(i) = \begin{cases} i-l+1, & i \in [l,n], \\ i-l+1+n, & i \in [1,l-1]. \end{cases}$$

This mapping simply transforms the time horizon set $[l, l-1]$ to the set $[n]$. To solve LS-U-PATH(k,l) for $k,l \in [n]$ where $k = l \neq 1$, this mapping will be used. When $k \neq l$ the problem condenses to a shortened horizon $[l, k-1]$ and the last few time periods shouldn't be included. This can be accomplished simply by not passing the last few parameters and having the path problem exist till $\pi_{k,l}(k-1)$ rather than n. Imagine an instance of LS-U-CYCLE,SC with demand and cost data (d,p,f,h,g), to map the data to the horizon of LS-U-PATH,SC(k,l) the following can be done.

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

Then limit the horizon of the problem to $[\pi_{k,l}(k-1)]$.

Now the instance data $(d,p,f,h,g)'$ are in the correct form to be solved using the dynamic pro-

gramming algorithm for LS-U-PATH,SC. The single constrained LS-U-CYCLE problem is already in the correct form as well. So a polynomial time algorithm CYCLE-SC for LS-U-CYCLE,SC can be written and seen below as Algorithm 1. This algorithm, the following claim, and its proof, follow the structure and divide and conquer strategy laid out by Cooper (2018) who tackled the similar problem LS-U-CYCLE by breaking it into n instances of LS-U-PATH.

Claim 5. *The algorithm CYCLE-SC finds an optimal solution to LS-U-CYCLE,SC in* $O(n^3 \log(n) + O(n^2 \log(n)))$ *time.*

Proof. From Pochet and Wolsey (2006) it is known that PATH(k, l) can be computed in $O(n \log(n))$ and the LS-U-CYCLE problem can be solved in $O(n^2 \log(n))$. To solve LS-U-CYCLE,SC a single LS-U-CYCLE problem and at most n^2 instances of LS-U-PATH,SC must be solved. □

As Cooper (2018) further noted in the proof of his similar claim, the order of the problem is very much an upper bound. Depending on the instance of the cycle problem it could be solved much faster. This is because in the solution of LS-U-PATH,SC(k, l) there may exist some other pair of start-up and production times $(i, j) \in [n] \setminus \{(k, l)\}$ such that $z_i = 1$ and $x_j > 0$ for $j = \pi_{k,l}^{-1}(\arg\min\{m \in [\pi_{k,l}(i), \pi_{k,l}(k-1)] : x_{\pi_{k,l}(m)} > 0\})$. Essentially the next start-up and the soonest production that either occurs at the same time or follows after it. The inverse of the $\pi_{k,l}$ mapping is also employed here, to map the soonest production back to the original cycle problem. Let the set of all other such pairs be defined as $S^{k,l} = \{(i, j) \in [n] \setminus \{(k, l)\} : z_i = 1, x_j > 0, j = \pi_{k,l}^{-1}(\arg\min\{m \in [\pi_{k,l}(i), \pi_{k,l}(k-1)] : x_{\pi_{k,l}(m)} > 0\})\}$ in the solution of LS-U-PATH,SC(k, l). If $S^{k,l} \neq \emptyset$ for any $(k, l) \in [n]$ then there is no longer a need to solve all n^2 instances of LS-U-PATH,SC. This implies that the algorithm for LS-U-CYCLE,SC can be computed even faster than the time given in Claim 5 as an upper bound.

Furthermore, this report presents a new improvement to the algorithm not based on the past work of Cooper (2018) that could even be generalised to the case considered there without start-up costs. The essential idea is to perform a simple check to see if solving a particular instance of the path problem is worthwhile. When performing the preprocessing for an instance LS-U-PATH,SC(k, l) where $k \neq l$, the new start-up cost g'_l must be computed. Interestingly if this cost of starting up is greater than what it was previously $g'_l > g_l$, then it can be immediately ignored as it is a less optimal solution. This is because the instance LS-U-PATH,SC(k, l) where $k \neq l$ assumes a start-up occurs in period k and all demand in periods $[k, l-1]$ is satisfied by stock. The instance with the same production time l where $k = l$ could still satisfy all demand in periods $[k, l-1]$ with stock. The only difference is when the start-up occurs and how much it will cost. So if it's cheaper to start-up in period l than start-up in period k and set-up in the intermediary time periods, then there is no point considering the case where $k \neq l$.

However the reverse is not true if $g'_l \leqslant g_l$ then the case where $k = l$ should still be considered. For example if g'_l was only marginally smaller than g_l because g_k was large but was counterbalanced by a highly negative set-up cost, this set-up could potentially be better captured when $k = l$ by setting up in every time period until then. This extends nicely to the n times start-up can occur, $k \in [n]$, for a

production that occurs in period l. The new start-up cost g'_l for starting up in period k can be denoted $g'_l(k)$ and computed for all $k \in [n]$. If any new start-up cost is greater than one that would be imposed by having a smaller sequence from starting up to producing, such that $g'_l(k) > g'_l(t)$ for any t in $[k, l]$, then it can be ignored. Since start-up and set-up costs can be negative, all n possible values of $g'_l(k)$ would have to be evaluated. However not all n equivalent instances of the path problem would necessarily have to be solved.

Algorithm 1 Polynomial Time Algorithm for LS-U-CYCLE,SC

procedure CYCLE-SC
  $T \leftarrow \emptyset$
  $v^* \leftarrow \infty$
  for $l \in [n]$ **do**
    $G \leftarrow \emptyset$
    for $k \in \{l, l-1, l-2, \ldots, l+2, l+1\}$ **do**
      if $(k, l) \notin T$ **then**
        if $k = l$ **then**
          $g' \leftarrow g_k$
        else
          $g' \leftarrow g_k + q_{k,l-1}$
        end if
        if $g' < \max(G)$ or $k = l$ **then**
          $G \leftarrow G \cup g'$
          $(v', (x', y', s', z')) \leftarrow \text{PATH}(k, l)$
          $T \leftarrow T \cup \{(i, j) \in [n] \setminus \{(k, l)\} : z_i = 1, x_j > 0, j = \pi_{k,l}^{-1}(\arg\min\{m \in [\pi_{k,l}(i), \pi_{k,l}(k-1)] : x_{\pi_{k,l}(m)} > 0\})\}$
          if $v' < v^*$ **then**
            $v^* \leftarrow v'$
            $(x^*, y^*, s^*, z^*) \leftarrow (x', y', s', z')$
          end if
        end if
      end if
    end for
  end for
  return $(v^*, (x^*, y^*, s^*, z^*))$
end procedure

It should also be noted that if there's no demand in time period l then the solution mightn't start-up or produce in the first time period like it would be expected to. It could do so in a later period when

demand is first positive and find an optimal solution. This solution would still be valid, it just can't be assumed that $z_k = 1$ or $x_l > 0$ in the actual solution. Instead the full problem solution would have to be taken, and mapped back to the correct time horizon. In the $k \neq l$ case again it can't be assumed that $z_k = 1$, and care must be taken to stretch the condensed horizon of the path problem back to the original horizon of the cycle problem. It can still be assumed that stock is held for the final time periods as the solution couldn't avoid satisfying the modified demand d_{k-1}.

8 Discussion and Conclusion

In this report the problem of lot sizing on a cycle, with start-up costs, has been introduced, showing its formulation as an optimisation problem with objective function and constraints and its underlying network. The properties of feasible solutions to the problem with capacities have been described, namely that the sum of demands must be at least zero and at most the sum of all capacity for production. Furthermore, for the uncapacitated problem, the structure of its bounded, optimal, feasible solutions is shown to be a sequence of regeneration intervals where production in one time period satisfies the demand for some length of time. Finally a polynomial time algorithm was detailed which split the problem into many instances of a path problem for which there already exists a polynomial time algorithm.

Some possible future directions of this research include improvements to the algorithm, as it seems possible that only the case where $k = l$ needs to be solved, and then if relevant different values of g'_l could be compared to see if an improvement on this solution could be made. Furthermore, a clear route for further work can be seen in the work of Cooper (2018) and Pochet and Wolsey (2006), as the valid inequalities for LS-U-CYCLE,SC could be investigated. This would then lead to a description and proof of the convex hull of the problem, which could be extended back to the LS-U-CYCLE problem as well.

References

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

R Cooper. Lot sizing on a cycle. Report, AMSI VRS, 2018.

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