The network maintenance problem

Parisa Charkhgard [a]**, Thomas Kalinowski** [a]**, Hamish Waterer** [a]

[a]*School of Mathematical and Physical Sciences, The University of Newcastle, Australia*
Email: parisa.charkhgard@uon.edu.au

Abstract: In this research, we describe an optimization problem motivated by the need to maintain infrastructure net-works over time. We consider infrastructure networks in which product is transported between distinct origin-destination pairs, and at the same time the infrastructure assets need to be maintained by resources moving in the network. In order to perform maintenance the assets have to be shut down from time to time thus reduc-ing the system capacity for those time periods. The objective is to maximize the total transported product by aligning the maintenance activities appropriately.

This problem combines flow maximization with maintenance scheduling capturing some important aspects of the motivating practical problem: (1) the interaction between utilization of network assets such as nodes and arcs and their maintenance demands, (2) the limited resources available to perform the maintenance, and (3) the time for moving the maintenance resources between different locations in the network. Depending on the application context, there are a number of natural ways to reflect these in a mathematical model, and this gives rise to a rich and challenging optimization problem which we call the network maintenance problem.

We formally introduce the problem, and present a mixed integer programming formulation. Next, we consider the case of a single commodity and a single maintenance resource when the network is a single path. We describe a polynomial time algorithm which, under some simplifying assumptions, solve the single path case to optimality. The problem becomes more challenging when the simplifying assumptions are dropped.

Keywords: Flow maximization, maintenance scheduling, special cases, polynomial time algorithms

1 INTRODUCTION

We consider a problem in which a network with arc capacities and a list of distinct origin-destination pairs are given. The network is used to send flow between the origin-destination pairs. At the same time, the arcs of the network need to be maintained by maintenance resources moving in the network. We are interested in scheduling maintenance activities on the arcs of the network so as to maximize the total flow over a set of time periods. Each arc has an inventory which represents the condition of that arc. The higher the level of inventory, the better the condition of the arc. Sending flow along an arc consumes its inventory which then has to be replenished by a maintenance resource. When the inventory is zero on an arc, the arc is broken, implying that no flow can be sent through that arc. Furthermore, maintenance on an arc shuts down the arc, making its capacity zero for that period. This problem is motivated by real-life applications in infrastructure networks, where the infrastructure assets are maintained over periods of time.

Boland et al. (2014) introduced a general network optimization problem in which a set of arc maintenance jobs needs to be scheduled such that the total flow in the network over time is maximized. A simplified version of the problem in which all jobs have unit processing times was studied by Boland et al. (2014). The authors establish the complexity of the problem taking into account certain instance characteristics, such as special network structures and restrictions on the set of jobs. Boland et al. (2016) extended the problem by adding a constraint to ensure that the number of jobs scheduled in any time period does not exceed a given number. Those problems are different from the network maintenance problem introduced in this paper as they assume that a set of maintenance jobs with their corresponding time windows are given a priori. The network maintenance problem decides when and how much maintenance is required for each arc such that the total flow is maximized.

We may compare some special cases of the network maintenance problem (e.g., when the network is a single arc) to the lot-sizing problem with sales (see e.g., Loparic et al., 2001). We consider each arc of the network maintenance problem as a commodity in the lot-sizing problem. Increasing the inventory of an arc by doing maintenance is equivalent to increasing the inventory of a commodity by production. In addition, sending flow through an arc decreases its inventory which is equivalent to decreasing the inventory of a commodity by selling it. Note that in the network maintenance problem, doing maintenance and sending flow cannot occur on an arc at the same period, while production and sale in a typical lot-sizing problem may occur for a commodity at the same period. The reader is referred to Pochet and Wolsey (1995), Pochet and Wolsey (2006), Van Vyve (2004), and Van Vyve (2006) for more details on lot-sizing problems.

This paper is organized as follows: In Section 2 we present a mathematical formulation for the problem. In Section 3 we study the special case of the problem in which the network is a single path. Finally, the paper is concluded with Section 4.

2 MATHEMATICAL FORMULATION

We are given the following data:

1. a network $G = (V, A)$ with node set V arc set A, and a capacity u_a every arc $a \in A$,

2. a list of origin-destination pairs $(O_i, D_i) \in V \times V$, $i = 1, \ldots, K$,

3. a time horizon $[T] = \{1, \ldots, T\}$, and

4. for every arc $a \in A$ a set $R(a) \subseteq A$ of arcs that are reachable from a. We assume that $a \in R(a)$ for all a, i.e. it is always an option to stay at the current arc. Natural choices for $R(a)$ could be:

   - $R(a) = A$, i.e. the travel time of the resource is negligible,
   - $R(a) = \{a' \in A : a' \text{ is adjacent to } a\}$ (Two arcs are adjacent if they share a node)
   - $R(a) = \{a' \in A : \text{dist}(a, a') \leqslant r\}$ for some positive integer r, where $\text{dist}(\cdot, \cdot)$ is an appropriately defined distance function on the arc set.

The basic idea is that in each period we are sending flows from the origins to the destinations, and sending flow along an arc consumes an inventory which then has to be replenished by a resource which moves from an arc a to an arc in $R(a)$ between two consecutive time periods. There are two rates describing how the inventories of arcs change. The wear rate ρ^{w} measures the decrease of the inventory for each unit of flow that is sent along

an arc a, and the repair rate ρ^{r} measures by how much the inventory of an arc increases when the resource occupies the arc for one time period. Without loss of generality, we scale everything such that $\rho^{\mathrm{w}} = 1$. The interaction between the capacities and the resource are governed by the following rules:

1. If the inventory of an arc is empty then no flow can be sent along the arc.

2. If the resource occupies arc a in period t then it is an option to increase the inventory by ρ^{r}.

3. If in period t the resource adds a positive amount to the inventory of arc a, then no flow can be sent along arc a in period t.

4. If the inventory of an arc is not empty and there is no resource doing maintenance on that arc, then the flow on the arc can be positive.

5. In period t the inventory of arc a decreases by $\rho^{\mathrm{w}} x_{at} = x_{at}$, where x_{at} is the flow on arc a in period t.

6. The flow on arc a at period t cannot exceed the minimum of u_a and the inventory of arc a at period $t-1$.

We introduce the following decision variables:

1. Continuous flow variables x_{akt} for $a \in A$, $k \in [K]$, $t \in [T]$

2. Continuous inventory variables s_{at} for $a \in A$, $t \in [T]$ (initial inventory values s_{a0} are part of the input)

3. Binary variables z_{at} indicating that the resource is on arc a in period t (the initial conditions are given by $z_{a0} = 1$ for a unique arc a, and $z_{a'0} = 0$ for all $a' \in A \setminus \{a\}$)

4. Binary variables y_{at} indication the the resource makes a delivery to arc a

5. Bookkeeping variables θ_{kt} for $k \in [K]$, $t \in [T]$ (the total flow of commodity k in period t)

The problem can be formulated as follows.

$$\text{Maximise} \sum_{t=1}^{T} \sum_{k=1}^{K} \theta_{kt} \quad \text{subject to} \tag{1}$$

$$\sum_{a \in N^{\mathrm{out}}(i)} x_{akt} - \sum_{a \in N^{\mathrm{in}}(i)} x_{akt} = \begin{cases} 0 & \text{if } i \notin \{O_k, D_k\} \\ \theta_{kt} & \text{for } i = O_k \\ -\theta_{kt} & \text{for } i = D_k \end{cases} \quad \text{for } i \in V,\ k \in [K],\ t \in [T], \tag{2}$$

$$\sum_{k=1}^{K} x_{akt} \leqslant u_a (1 - y_{at}) \qquad \text{for } a \in A,\ t \in [T], \tag{3}$$

$$y_{at} \leqslant z_{at} \qquad \text{for } a \in A,\ t \in T \tag{4}$$

$$\sum_{a \in A} z_{at} = 1 \qquad \text{for } a \in A,\ t \in [T], \tag{5}$$

$$z_{a,t-1} \leqslant \sum_{a' \in R(a)} z_{a't} \qquad \text{for } a \in A,\ t \in [T], \tag{6}$$

$$s_{at} = s_{a,t-1} - \sum_{k=1}^{K} x_{akt} + \rho^{\mathrm{r}} y_{at} \qquad \text{for } a \in A,\ t \in [T], \tag{7}$$

$$x_{akt},\ s_{at},\ \theta_{kt} \geqslant 0 \qquad \text{for } a \in A,\ k \in [K],\ t \in [T], \tag{8}$$

$$y_{at},\ z_{at} \in \{0, 1\} \qquad \text{for } a \in A,\ t \in [T]. \tag{9}$$

The objective (1) is to maximize the total throughput. Constraints (2) and (3) are flow conservation and capacity constraints, respectively. Constraint (4) ensures that maintenance on arc a at time period t occurs only if the resource occupies that arc at that period. Constraints (5) and (6) captures the movement of the resource. Constraint (7) ensures inventory balance at each arc. Finally, constraints (8) and (9) specify the domain of decision variables.

3   SINGLE PATH CASE

Definition 1. For $x \in \mathbb{R}$ let x_+ or $(x)_+$ denote $\max\{0, x\}$.

Problem 1. Solve the original problem (1)–(9) for the case that $K = 1$, $R(a) = A$ for all $a \in A$ and the network is a single $O - D$ path.

Observation 1. In the single path case, for all $t \in [T]$ and all $a \in A$, $y_{at} = 1 \implies \theta_t = 0$.

Observation 2. In the single path case, without loss of generality we can assume $u_a = u$ for all $a \in A$.

We assume that the arcs in the set $A = \{a_1, \cdots, a_m\}$ are indexed according to the order in which they are traversed on the path, and for simplicity we write y_{it} and s_{it} for $y_{a_i t}$ and $s_{a_i t}$, respectively. Using Observations 1 and 2, the single path problem, denoted by SINGLEPATHMTCSCHED, can be formulated as follows:

$$\text{Maximise} \sum_{t=1}^{T} \theta_t \quad \text{subject to} \tag{10}$$

$$\theta_t \leqslant u\bar{z}_t \qquad\qquad \text{for } t \in [T], \tag{11}$$

$$\bar{z}_t + \sum_{i=1}^{m} y_{it} = 1 \qquad\qquad \text{for } t \in [T], \tag{12}$$

$$s_{it} = s_{i,t-1} - \theta_t + \rho^{\mathrm{r}} y_{it} \qquad\qquad \text{for } t \in [T],\ i \in [m], \tag{13}$$

$$\theta_t, s_{it} \geqslant 0 \qquad\qquad \text{for } t \in [T],\ i \in [m], \tag{14}$$

$$y_{it}, \bar{z}_t \in \{0, 1\} \qquad\qquad \text{for } t \in [T],\ i \in [m]. \tag{15}$$

We also introduce ℓ_i as the number of maintenance periods for arc $a_i \in A$:

$$\ell_i = \sum_{t \in [T]} y_{it} \qquad i = 1, \ldots, m.$$

Remark 1. For every feasible solution, $\ell_1 + \cdots + \ell_m \leqslant T$.

Lemma 1. *Let $(\theta, y, s, \bar{z})$ be a feasible solution for the problem SINGLEPATHMTCSCHED with objective value Z. Then*

$$Z \leqslant \left(T - \sum_{i=1}^{m} \left\lceil \frac{(Z - s_{i0})_+}{\rho^r} \right\rceil \right) u.$$

Proof. Using constraint (13), for each $i \in [m]$ we have:

$$Z = \sum_{t \in [T]} \theta_t = s_{i0} + \rho^{\mathrm{r}} \sum_{t \in [T]} y_{it} - s_{iT} \leqslant s_{i0} + \rho^{\mathrm{r}} \sum_{t \in [T]} y_{it},$$

which implies that for each $i \in [m]$,

$$\sum_{t \in [T]} y_{it} \geqslant \left\lceil \frac{(Z - s_{i0})_+}{\rho^{\mathrm{r}}} \right\rceil.$$

Summing over i, we obtain

$$\sum_{i=1}^{m} \sum_{t \in [T]} y_{it} \geqslant \sum_{i=1}^{m} \left\lceil \frac{(Z - s_{i0})_+}{\rho^{\mathrm{r}}} \right\rceil. \tag{16}$$

Now

$$Z = \sum_{t \in [T]} \theta_t \overset{(11)}{\leqslant} u \sum_{t \in [T]} \bar{z}_t \overset{(12)}{=} u \sum_{t \in [T]} \left(1 - \sum_{i=1}^{m} y_{it} \right) = u \left(T - \sum_{i=1}^{m} \sum_{t \in [T]} y_{it} \right)$$

$$\overset{(16)}{\leqslant} \left(T - \sum_{i=1}^{m} \left\lceil \frac{(Z - s_{i0})_+}{\rho^{\mathrm{r}}} \right\rceil \right) u. \quad \square$$

Lemma 2. *Let Z be an integer with*

$$Z \leqslant \left(T - \sum_{i=1}^{m} \left\lceil \frac{(Z - s_{i0})_+}{\rho^r} \right\rceil \right) u.$$

Then there exists a feasible solution $(\theta, y, s, \bar{z})$ for the problem SINGLEPATHMTCSCHED *with objective value*

$$\sum_{t=1}^{T} \theta_t = Z.$$

Proof. Let $\ell_i = \lceil (Z - s_{i0})_+ / \rho^r \rceil$, $K = \lfloor Z/u \rfloor$ and $L = \ell_1 + \cdots + \ell_m$. The assumption in the lemma can be written as $Z \leqslant (T - L)u$, and consequently $Ku \leqslant Z \leqslant (T - L)u$, which implies

$$L + K \leqslant \begin{cases} T & \text{if } Z = Ku, \\ T - 1 & \text{if } Z > Ku. \end{cases}$$

Now we can define a feasible solution with objective value Z as follows.

$$\theta_t = \begin{cases} 0 & \text{if } 1 \leqslant t \leqslant L, \\ u & \text{if } L + 1 \leqslant t \leqslant L + K, \\ Z - Ku & \text{if } t = L + K + 1, \\ 0 & \text{if } t \geqslant L + K + 2, \end{cases} \qquad t = 1, \ldots, T$$

$$\bar{z}_t = \begin{cases} 0 & \text{if } 1 \leqslant t \leqslant L, \\ 1 & \text{otherwise}, \end{cases} \qquad t = 1, \ldots, T$$

$$y_{it} = \begin{cases} 1 & \text{if } \ell_1 + \cdots + \ell_{i-1} + 1 \leqslant t \leqslant \ell_1 + \cdots + \ell_i, \\ 0 & \text{otherwise}. \end{cases} \qquad i = 1, \ldots, m, \ t = 1, \ldots, T$$

Note that the values for the s_{it} for $t \geqslant 1$ are determined by these values and (13). $\qquad \square$

Combining Lemmas 1 and 2, we obtain the following proposition.

Proposition 1. In the single path case with $R(a) = A$, the optimal objective value is

$$Z^* = \max \left\{ Z \ : \ \left(T - \sum_{i=1}^{m} \left\lceil \frac{(Z - s_{i0})_+}{\rho^r} \right\rceil \right) u \geqslant Z \right\}.$$

Corollary 1. *Problem* (10)–(15) *always has an optimal solution* $(\theta^*, y^*, \bar{z}^*)$ *such that for some L,*

$$\bar{z}_t^* = \begin{cases} 0 & \text{for } 1 \leqslant t \leqslant L, \\ 1 & \text{for } L + 1 \leqslant t \leqslant T. \end{cases}$$

Corollary 2. *Let $\ell = (\ell_i)_{1 \leqslant i \leqslant m} \in \mathbb{Z}_{\geqslant 0}$ be a vector with $\ell_1 + \cdots + \ell_m \leqslant T$, and consider the problem* (10)–(15) *with the additional constraints*

$$\sum_{t \in [T]} y_{it} = \ell_i \qquad \text{for all } i \in \{1, \ldots, m\}.$$

The optimal objective value for this problem is

$$\min \left\{ (T - \sum_{a \in A} \ell_a) u, \ \min_{1 \leqslant i \leqslant m} s_{i0} + \ell_i \rho^r \right\}.$$

Corollary 3. *In order to solve a single path case with $R(a) = A$, it is sufficient to solve problem* (17)–(21).

   Maximise Z subject to $\hfill (17)$

$$Z \leqslant \left(T - \sum_{i=1}^{m} \ell_i \right) u \tag{18}$$

$$Z \leqslant s_{i0} + \ell_i \rho^r \qquad \qquad for\ i \in [m], \tag{19}$$

$$Z \geqslant 0 \tag{20}$$

$$\ell_a \in \mathbb{Z}_{\geqslant 0} \qquad \qquad for\ i \in [m]. \tag{21}$$

Algorithm 1 is based on Proposition 1 to solve single path cases with $R(a) = A$ to optimality. It uses binary search to find the optimal value for the total flow, Z^*, and the number of times that each arc should be visited for maintenance, ℓ_i^* for $i \in [m]$. Let $K^* = \lfloor Z^*/u \rfloor$ and $L^* = \ell_1 + \cdots + \ell_m$. After finding Z^* and ℓ_a^*, the following is going to be an optimal solution:

$$\theta_t^* = \begin{cases} 0 & \text{for } 1 \leqslant t \leqslant L^*, \\ u & \text{for } L^* + 1 \leqslant t \leqslant L^* + K^*, \\ Z^* - K^*u & \text{for } t = L^* + K^* + 1 \text{ (if } L^* + K^* + 1 \leqslant T) \\ 0 & \text{otherwise.} \end{cases}$$

$$\bar{z}_t^* = \begin{cases} 0 & \text{for } 1 \leqslant t \leqslant L^*, \\ 1 & \text{otherwise.} \end{cases}$$

$$y_{it}^* = \begin{cases} 1 & \text{for } \ell_1^* + \cdots + \ell_{i-1}^* + 1 \leqslant t \leqslant \ell_1^* + \cdots + \ell_i^*, \\ 0 & \text{otherwise.} \end{cases}$$

Algorithm 1 Solving single path case with $R(a) = A$, finding Z^* (SPZ)

1: **Input:** (s_{a0}, u, ρ^r, T)
2: $Z_L = 0, Z_R = Tu$
3: **if** $\left(T - \sum_{a \in A} \left\lceil \frac{(Z_R - s_{a0})_+}{\rho^r} \right\rceil \right) u \geqslant Z_R$ **then**
4: $\quad Z^* = Z_R = Tu, \ell_a^* = 0.$ **return** $(\ell_a^*, Z^*).$
5: **else**
6: $\quad$ **while** $Z_L + 1 \neq Z_R$ **do**
7: $\quad\quad Z_M = \left\lceil \frac{Z_L + Z_R}{2} \right\rceil$
8: $\quad\quad$ **if** $\left(T - \sum_{a \in A} \left\lceil \frac{(Z_M - s_{a0})_+}{\rho^r} \right\rceil \right) u \geqslant Z_M$ **then**
9: $\quad\quad\quad Z_L = Z_M$
10: $\quad\quad$ **else**
11: $\quad\quad\quad Z_R = Z_M$
12: $\quad Z^* = Z_L, \ell_a^* = \left\lceil \frac{(Z_L - s_{a0})_+}{\rho^r} \right\rceil.$ **return** (ℓ_a^*, Z^*)

Example 1. Table 1 shows all steps in Algorithm 1 for solving the network shown in Figure 1 with $R(a) = A$, $u = 10$, $T = 7$ and $\rho^r = 9$. According to the table, in the first five periods the resource works on the arcs (2 periods on arc 1, 1 period on arc 2, 1 period on arc 3 and 1 period on arc 4). At time periods 6 and 7, 18 flow can be sent from the origin to destination.
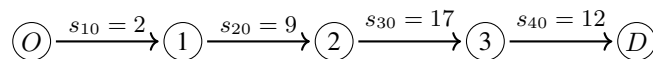
$$\textcircled{O} \xrightarrow{\ s_{10} = 2\ } \textcircled{1} \xrightarrow{\ s_{20} = 9\ } \textcircled{2} \xrightarrow{\ s_{30} = 17\ } \textcircled{3} \xrightarrow{\ s_{40} = 12\ } \textcircled{D}$$

Figure 1: A single path network.

Table 1. Solution given by Algorithm 1 for Example 1.

Step	Z_L	Z_R		Calculation
0	0	70		$-210 < 70$

Step	Z_L	Z_R	Z_M	Calculation
1	0	70	35	$-50 < 35$
2	0	35	18	$20 \geqslant 18$
3	18	35	27	$-20 < 27$
4	18	27	23	$-10 < 23$
5	18	23	21	$0 < 21$
6	18	21	20	$10 < 20$
7	18	20	19	$10 < 19$

ℓ_1^*	ℓ_2^*	ℓ_3^*	ℓ_4^*	Z^*
2	1	1	1	18

4 CONCLUSION

In this paper, we introduced a new combinatorial optimization problem called the network maintenance problem. We presented a mixed integer linear programming formulation for this problem. The formulation can be solved using any standard linear solver. Under some simplifying assumptions, we were able to solve to optimality the special case of the problem having a single path network. We presented a binary search algorithm which finds an optimal solution of the single path case, efficiently. A promising direction for future research is to develop efficient algorithms to solve the general case of the p roblem. Developing strong valid inequalities for the proposed mathematical formulation can also be a good future research direction. The network maintenance problem introduced in this paper may be extended in different ways such as considering upper bounds on the inventories, multiple resources, resource travel times, etc.

REFERENCES

Boland, N., T. Kalinowski, and S. Kaur (2016). Scheduling arc shut downs in a network to maximize flow over time with a bounded number of jobs per time period. *Journal of Combinatorial Optimization 32*(3), 885–905.

Boland, N., T. Kalinowski, H. Waterer, and L. Zheng (2014). Scheduling arc maintenance jobs in a network to maximize total flow over time. *Discrete Applied Mathematics 163*, 34–52.

Boland, N., R. Kapoor, S. Kaur, and T. Kalinowski (2014). Scheduling unit time arc shutdowns to maximize network flow over time: complexity results. *Networks 63*(2), 196–202.

Loparic, M., Y. Pochet, and L. A. Wolsey (2001). The uncapacitated lot-sizing problem with sales and safety stocks. *Mathematical Programming 89*(3), 487–504.

Pochet, Y. and L. A. Wolsey (1995). Algorithms and reformulations for lot sizing problems. *"DIMACS Series in Discrete Mathematics and Theoretical Computer Science" 20*, 245–293.

Pochet, Y. and L. A. Wolsey (2006). *Production planning by mixed integer programming.* Springer Science & Business Media.

Van Vyve, M. (2004). A solution approach of production planning problems based on compact formulations for single-item lot-sizing models. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies 2*(1), 89–91.

Van Vyve, M. (2006). Linear programming extended formulations for the single-item lot-sizing problem with backlogging and constant capacity. *Mathematical Programming 108*, 53–77.