Scheduling arc shut downs in a network to maximize flow over time with a bounded number of jobs per time period*

Natashia Boland Thomas Kalinowski Simranjit Kaur
University of Newcastle, Australia.

July 11, 2013

Abstract

We study the problem of scheduling maintenance on arcs of a capacitated network so as to maximize the total flow from a source node to a sink node over a set of time periods. Maintenance on an arc shuts down the arc for the duration of the period in which its maintenance is scheduled, making its capacity zero for that period. A set of arcs is designated to have maintenance during the planning period, which will require each to be shut down for exactly one time period. In general this problem is known to be NP-hard, and several special instance classes have been studied. Here we propose an additional constraint which limits the number of maintenance jobs per time period, and we study the impact of this on the complexity.

Keywords: network models, complexity theory, maintenance scheduling, mixed integer programming
AMS Classification: 90C10, 90B10, 68Q25

Introduction

Motivated by the annual maintenance planning for a coal export supply chain [4], in which maximizing the annual throughput is a key concern, Boland *et al.* [2, 3] introduced a general network optimization problem in which arc maintenance jobs need to be scheduled so as to maximize the total flow in the network over time. A simplified version of the problem in which all jobs have unit processing time was studied in [1], and the complexity was determined taking into account certain instance characteristics, such as special network structures and restrictions on the set of jobs. In the present paper we extend this model by adding the constraint that the number of jobs scheduled in any time period is bounded by a number K which is given as part of the input. The problem is defined over a network $N = (V, A, s, t, u)$ with node set V, arc multiset A (i.e. we admit parallel arcs having the same start and end nodes), source $s \in V$, sink $t \in V$ and nonnegative integral capacity vector $u = (u_a)_{a \in A}$. By $\delta^-(v)$ and $\delta^+(v)$ we denote the set of incoming and outgoing arcs of node v, respectively. We consider this network over a set of T time periods indexed by the set $[T] := \{1, 2, \ldots, T\}$, and our objective is to maximize the total flow from s to t. We are also given a subset $J \subseteq A$ of arcs that have to be shut down for exactly one time period in the time horizon. In other words, there is a set of maintenance jobs, one for each arc in J, each with unit processing time. In addition, there is a parameter K such that the number of maintenance jobs scheduled in any time period must not exceed K. From a practical point of view, this is a natural variation of the model. In many real world network maintenance scheduling problems, there are resource and budget constraints which do not allow to do too many jobs at the same time. Of course, in practice there can be complicated rules about the combinations of jobs that are allowed. Disregarding these complications, we propose a simple extension of the basic model which allows us to study the effect of such job limit constraints in a simplified setting. The optimization problem is to choose the outage time periods in such a way that the total flow from s to t is maximized.

*This research was supported by the ARC Linkage Grant no. LP0990739 and HVCCC P/L.

More formally, this can be written as a mixed binary program as follows:

$$\max\ z = \sum_{i=1}^{T} \sum_{a \in \delta^+(s)} x_{ai} \tag{1}$$

$$\text{s.t.} \quad x_{ai} \leqslant u_a y_{ai} \qquad a \in J,\ i \in [T], \tag{2}$$

$$x_{ai} \leqslant u_a \qquad a \in A \setminus J,\ i \in [T], \tag{3}$$

$$\sum_{i=1}^{T} y_{ai} = T - 1 \qquad a \in J, \tag{4}$$

$$\sum_{a \in \delta^-(v)} x_{ai} = \sum_{a \in \delta^+(v)} x_{ai} \qquad v \in V \setminus \{s,t\},\ i \in [T], \tag{5}$$

$$\sum_{a \in J} y_{ai} \geqslant |J| - K \qquad i \in [T], \tag{6}$$

$$x_{ai} \geqslant 0 \qquad a \in A,\ i \in [T], \tag{7}$$

$$y_{ai} \in \{0,1\} \qquad a \in A,\ i \in [T], \tag{8}$$

where x_{ai} for $a \in A$ and $i \in [T]$ denotes the flow on arc a in time period i, and $y_{ai} \in \{0,1\}$ for $a \in A$ and $i \in [T]$ indicates when the arc a is available in time period i, i.e. $y_{ai} = 0$ in the period i in which the outage for arc a is scheduled. The problem is to schedule the maintenance jobs so that the total flow of the network over the time horizon T is maximized.

The reduction from 3-Partition in [3] shows that the general problem is strongly NP-complete for the class of instances with $K = 3$. In [1] several instance classes for the problem without the job limit per time period were analyzed. In order to classify instances we introduce the following notation. Let $\mathcal{C}$ be the class of all instances of the problem (1) to (8). With an upper index K we denote the class of all instances with an upper bound of K on the number of jobs scheduled per time period, and a lower index indicates additional restrictions as introduced in [1].

- Let $\mathcal{C}_{\text{sp}}$ be the class of instances where the underlying network is series-parallel.
- Let $\mathcal{C}_{\text{bal}}$ be the class of instances where the underlying network is *balanced*, i.e. for each transshipment node $v \in V \setminus \{s,t\}$ the capacity into this node equals the capacity out of this node.
- Let $\mathcal{C}_{uc}$ be the class of instances where the capacities are $u_a = 1$ for every arc $a \in A$.
- Let $\mathcal{C}_{\text{aa}}$ be the class of instances where *a*ll *a*rcs have a job associated, i.e. $J = A$.

For instance $\left(\mathcal{C}^3_{\text{sp}} \cap \mathcal{C}^3_{\text{aa}}\right) \setminus \mathcal{C}^3_{\text{bal}}$ is the set of all instances with a series-parallel network which is not balanced, a job associated with every arc, and the constraint that at most 3 jobs can be scheduled per time period. In general, K is not constant, and we also consider instance classes with varying K, but imposing some restrictions on how K can vary relative to other instance parameters. For instance, $\mathcal{C}^{|J|}_{\text{sp}}$ is the class of instances with a series-parallel network and no limit on the number of jobs per time period, and $\mathcal{C}^{|J|/3}$ contains the instances in which at most one third of all jobs can be scheduled per time period. As proved in [1], the classes $\mathcal{C}^{|J|}_{\text{aa}}$ and $\mathcal{C}^{|J|}_{\text{sp}} \cap \mathcal{C}^{|J|}_{\text{bal}}$ are trivial: it is always optimal to schedule all jobs at the same time. In contrast, the restriction of the problem to $\mathcal{C}^{|J|}_{\text{bal}}$ is still strongly NP-hard, and the restriction to $\mathcal{C}^{|J|}_{\text{sp}}$ is NP-hard, but for fixed T it can be solved in pseudopolynomial time using dynamic programming. Our new complexity results are summarised in Table 1.

Note that the problem is solvable in polynomial time if both T and K are bounded, say $T \leqslant T_0$ and $K \leqslant K_0$ for some absolute constants T_0 and K_0. Then $|J| \leqslant K_0 T_0$ for any feasible instance, and we can enumerate all partitions of J into at most T sets of size at most K of which there are at most

$$C = \prod_{i=0}^{T_0} \binom{K_0(T_0 - i)}{K_0}.$$

Instance class	Complexity
$\mathcal{C}^3_{\text{sp}} \cap \mathcal{C}^3_{\text{bal}} \cap \mathcal{C}^3_{\text{aa}}$	strongly NP-complete (Proposition 3)
$\mathcal{C}^{\lvert J\rvert-1}_{\text{sp}} \cap \mathcal{C}^{\lvert J\rvert-1}_{\text{bal}} \cap \mathcal{C}^{\lvert J\rvert-1}_{\text{aa}}$	NP-complete (Proposition 4)
$\mathcal{C}_{\text{uc}}$	NP-complete (Proposition 5)
$\mathcal{C}^2$	$O(\lvert J\rvert^3)$ (Proposition 1)

Table 1: Complexity results.

For each of these partitions we have to solve T maximum flow problems, hence the run-time is bounded by CT_0nm, since the maximum flow problem can be solved in $O(mn)$ time [7, 8]. Consequently, for the asymptotic analysis we are interested in instance classes where at least one of the parameters T and K is unbounded.

The paper is organized as follows. In Section 1 we show that the case $K = 2$ can be solved in polynomial time. In addition we provide an explicit description of an optimal solution for $K = 2$ and a network with a single transshipment node which leads to a significantly better run-time bound for this case. The hardness results are proved in Section 2. In Section 3 we present a fully polynomial time approximation scheme for series-parallel networks with fixed time horizon. We also provide a polynomial time approximation scheme for series parallel networks in general when $K = |J|$.

1 The case $K = 2$

In this section we consider the case $K = 2$. In Section 1.1 we show that this case can be reduced to a maximum weighted matching problem and thus is solvable in polynomial time, and in Section 1.2 we give an explicit description of an optimal solution for the case that the network has only a single transshipment node.

1.1 General networks

We reduce the problem to a maximum weight perfect matching problem. Let F_0 denote the maximum flow value in the whole network, for $a \in J$ let F_a denote the maximum flow when arc a is shut, and for distinct $a, b \in J$ let F_{ab} be the maximum flow when arcs a and b are shut. We set $p = \max\{0, |J| - T\}$ and define an auxiliary graph whose vertex set contains two vertices for every arc $a \in J$ and two sets W and W' of dummy vertices with $|W| = 2p$ and $|W'| = 2(\lfloor |J|/2 \rfloor - p)$. The two vertices for $a \in J$ are denoted by a and a', and the weighted edge set of the auxiliary graph is defined as follows:

- For distinct arcs $a, b \in J$ there is an edge $\{a, b\}$ with weight $F_{ab} + F_0$.
- For $a \in J$ there is an edge $\{a, a'\}$ of weight F_a.
- There are all edges of the form $\{a', w\}$ for $a \in J$ and $w \in W \cup W'$. All these edges have zero weight.
- The vertex set W' induces a matching consisting of zero weight edges.

There is a correspondence between perfect matchings in the auxiliary graph and outage schedules. Let M be a perfect matching in the auxiliary digraph. The corresponding schedule has

- for every edge $\{a, b\} \in M$ with $a, b \in J$ one time period with arcs a and b shut,
- for every edge $\{a, a'\} \in M$ with $a \in J$ one time period with only arc a shut,
- all other time periods without shut arcs.

This construction is illustrated in Figure 1 for $J = \{a, b, \ldots, h\}$ and $T = 6$. The bold edges form a perfect matching corresponding to scheduling the following outage of schedule: period 1: $\{a, d\}$, period 2: $\{c, f\}$, period 3: $\{g, h\}$, period 4: $\{b\}$, period 5: $\{e\}$, period 6: $\varnothing$.

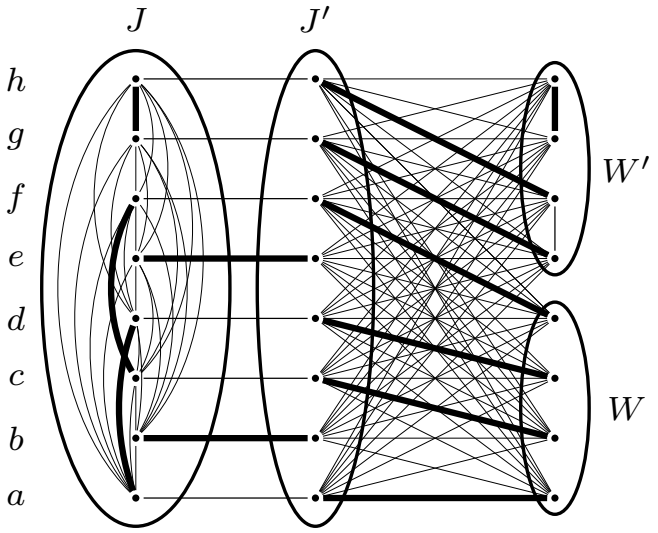

Figure 1: A perfect matching in the auxiliary graph.

For a perfect matching M we define subsets $M_1 \subseteq M$ and $M_2 \subseteq M$ by

$$M_1 = \{\{a,b\} \in M \ : \ a,b \in J\}, \qquad\qquad M_2 = \{\{a,a'\} \in M \ : \ a \in J\}.$$

Note that the $2p$ nodes in W must be matched to nodes a', hence

$$|M_2| \leqslant |J| - 2p \leqslant |J| - 2(|J| - T) = 2T - |J|,$$

and with $|M_1| = \frac{1}{2}(|J| - |M_2|)$ this implies

$$|M_1| + |M_2| = \frac{1}{2}(|J| - |M_2|) + |M_2| = \frac{1}{2}(|J| + |M_2|) \leqslant T.$$

The total throughput for the schedule corresponding to the matching M is

$$\begin{aligned}\sum_{\{a,b\}\in M_1} F_{ab} + \sum_{\{a,a'\}\in M_2} F_a + (T - |M_1| - |M_2|)F_0 \\ = \sum_{\{a,b\}\in M_1} (F_{ab} + F_0) + \sum_{\{a,a'\}\in M_2} F_a + (T - 2|M_1| - |M_2|)F_0 = \omega(M) + (T - |J|)F_0,\end{aligned}$$

where $\omega(M)$ is the weight of M. Thus the original problem is equivalent to finding a maximum weighted perfect matching in the auxiliary graph, and with an efficient implementation [6] of the blossom algorithm [5] we have proved the following proposition.

Proposition 1. *For $K = 2$ the problem (1) to (8) can be solved in $O(|J|^3)$ time.*

1.2 The single node case

Consider a network with a single transshipment node v, a job set J, a time horizon T and $K = 2$. We use the notation $J^- = \delta^-(v) \cap J$ and $J^+ = \delta^+(v) \cap J$ and assume without loss of generality that $|J^-| \leqslant |J^+|$. We order the arcs in J^- and J^+ such that the capacities are non-increasing, i.e. $J^- = \{a_1, \ldots, a_r\}$ and $J^+ = \{b_1, \ldots, b_s\}$ $(s \geqslant r)$ with

$$u_{a_1} \geqslant u_{a_2} \geqslant \cdots \geqslant u_{a_r}, \qquad\qquad u_{b_1} \geqslant u_{b_2} \geqslant \cdots \geqslant u_{b_s}.$$

Note that it is necessary for the feasibility that $r + s \leqslant 2T$, and in particular $r \leqslant T$. We will show that an optimal solution can be obtained as follows.

Proposition 2. *An optimal solution for the single node problem with $K = 2$ is given by the following schedule.*

- *For $i = 1, 2, \ldots, r$ take arc a_i out in time period i.*
- *For $i = 1, 2, \ldots, \min\{T, s\}$ take arc b_i out in time period i.*
- *For $i = T + 1, T + 2, \ldots, s$ take arc b_i out in time period $2T - i + 1$.*

The basic idea for proving Proposition 2 is to start with any schedule and show that, as long as it is not the described schedule, there is a small modification that does not decrease the objective value and such that the modified schedule is in some sense closer to the one described in the proposition. This process must terminate in a finite number of steps, and since we can assume that we start with an optimal schedule this proves the proposition. In order to justify the modification steps we will need the following notation.

$$\begin{aligned} X &= \sum_{a \in \delta^-(v)} u_a, & Y &= \sum_{a \in \delta^+(v)} u_a, \\ X_a &= X - u_a \text{ for } a \in J^-, & Y_a &= Y - u_a \text{ for } a \in J^+, \\ X_{ab} &= X - u_a - u_b \text{ for distinct } a, b \in J^-, & Y_{ab} &= Y - u_a - u_b \text{ for distinct } a, b \in J^+. \end{aligned}$$

Lemma 1. *For $a, b \in J^-$ and $c, d \in J^+$ we have*

$$\min\{X_{ab}, Y\} + \min\{X, Y_{cd}\} \leqslant \min\{X_a, Y_c\} + \min\{X_b, Y_d\}.$$

Proof. If $Y \leqslant X_{ab}$ then

$$\min\{X_{ab}, Y\} + \min\{X, Y_{cd}\} = Y + Y_{cd} = Y_c + Y_d = \min\{X_a, Y_c\} + \min\{X_b, Y_d\}.$$

If $X \leqslant Y_{cd}$ then

$$\min\{X_{ab}, Y\} + \min\{X, Y_{cd}\} = X_{ab} + X = X_a + X_b = \min\{X_a, Y_c\} + \min\{X_b, Y_d\}.$$

For the remaining case $X_{ab} < Y$ and $Y_{cd} < X$, i.e. $\min\{X_{ab}, Y\} + \min\{X, Y_{cd}\} = X_{ab} + Y_{cd}$, we check that this is upper bounded by each of the four possible expressions on the RHS:

$$\begin{aligned} X_{ab} + Y_{cd} &\leqslant X_{ab} + X = X_a + X_b, \\ X_{ab} + Y_{cd} &\leqslant X_a + Y_d, \\ X_{ab} + Y_{cd} &\leqslant X_b + Y_c = Y_c + X_b, \\ X_{ab} + Y_{cd} &\leqslant Y + Y_{cd} = Y_c + Y_d. \end{aligned}$$

□

Lemma 2. *For $a, b \in J^-$ and $c, d \in J^+$ with we have*

$$\min\{X_{ab}, Y\} + \min\{X, Y_c\} + \min\{X, Y_d\} \leqslant \min\{X_a, Y_c\} + \min\{X_b, Y_d\} + \min\{X, Y\}.$$

Proof. Without loss of generality we assume $u_c \geqslant u_d$, i.e. $Y_c \leqslant Y_d$. If $X_{ab} \geqslant Y$ then

$$\begin{aligned} \min\{X_{ab}, Y\} + \min\{X, Y_c\} + \min\{X, Y_d\} = Y + Y_c + Y_d &= Y_c + Y_d + Y \\ &= \min\{X_a, Y_c\} + \min\{X_b, Y_d\} + \min\{X, Y\} \end{aligned}$$

If $X \leqslant Y_c$ then

$$\begin{aligned} \min\{X_{ab}, Y\} + \min\{X, Y_c\} + \min\{X, Y_d\} = X_{ab} + X + X &= X_a + X_b + X \\ &= \min\{X_a, Y_c\} + \min\{X_b, Y_d\} + \min\{X, Y\} \end{aligned}$$

If $X_{ab} < Y$ and $X \geqslant Y_d$, then $\min\{X_{ab}, Y\} + \min\{X, Y_c\} + \min\{X, Y_d\} = X_{ab} + Y_c + Y_d$, and we check that this is upper bounded by every possible expression on the RHS:

$$\begin{aligned}
&X_{ab} + Y_c + Y_d \leqslant X_{ab} + X + Y_d = X_a + X_b + Y_d \leqslant X_a + X_b + \min\{X, Y\},\\
&X_{ab} + Y_c + Y_d \leqslant X_a + Y_d + \min\{X, Y\},\\
&X_{ab} + Y_c + Y_d = Y_c + X_{ab} + Y_d \leqslant Y_c + X_b + \min\{X, Y\},\\
&X_{ab} + Y_c + Y_d = Y_c + Y_d + X_{ab} \leqslant Y_c + Y_d + \min\{X, Y\}.
\end{aligned}$$

In the remaining case $Y_c < X < Y_d$ we have $\min\{X_{ab}, Y\} + \min\{X, Y_c\} + \min\{X, Y_d\} = X_{ab} + Y_c + X$, and again we check that this is upper bounded by every possible expression on the RHS:

$$\begin{aligned}
&X_{ab} + Y_c + X = X_{ab} + X + Y_c = X_a + X_b + Y_c \leqslant X_a + X_b + \min\{X, Y\},\\
&X_{ab} + Y_c + X = Y_c + X_b + X_a \leqslant Y_c + X_b + \min\{X, Y\}.
\end{aligned}$$

□

Lemma 3. *For $a \in J^-$ and $b, c \in J^+$ we have*

$$\begin{aligned}
&\min\{X_a, Y\} + \min\{X, Y_b\} \leqslant \min\{X_a, Y_b\} + \min\{X, Y\}, \qquad \textit{and}\\
&\min\{X_a, Y\} + \min\{X, Y_{bc}\} \leqslant \min\{X_a, Y_b\} + \min\{X, Y_c\}.
\end{aligned}$$

Proof. If $Y \leqslant X_a$ then

$$\min\{X_a, Y\} + \min\{X, Y_b\} = Y + Y_b = Y_b + Y = \min\{X_a, Y_b\} + \min\{X, Y\}.$$

If $X \leqslant Y_b$ then

$$\min\{X_a, Y\} + \min\{X, Y_b\} = X_a + X = \min\{X_a, Y_b\} + \min\{X, Y\}.$$

In the remaining case we have $X_a < Y$ and $Y_b < X$, hence

$$\min\{X_a, Y\} + \min\{X, Y_b\} = X_a + Y_b = \min\{X_a, Y_b\} + \max\{X_a, Y_b\} \leqslant \min\{X_a, Y_b\} + \min\{X, Y\}.$$

This proves the first inequality. For the second inequality, if $Y \leqslant X_a$ then

$$\min\{X_a, Y\} + \min\{X, Y_{bc}\} = Y + Y_{bc} = Y_b + Y_c = \min\{X_a, Y_b\} + \min\{X, Y_c\}.$$

If $X \leqslant Y_{bc}$ then

$$\min\{X_a, Y\} + \min\{X, Y_{bc}\} = X_a + X = \min\{X_a, Y_b\} + \min\{X, Y_c\}.$$

In the remaining case we have $X_a < Y$ and $Y_{bc} < X$, hence $\min\{X_a, Y\} + \min\{X, Y_{bc}\} = X_a + Y_{bc}$ and we check that this is upper bounded by every possible expression on the RHS:

$$\begin{aligned}
&X_a + Y_{bc} \leqslant X_a + \min\{X, Y_c\},\\
&X_a + Y_{bc} \leqslant Y + Y_{bc} = Y_b + Y_c,\\
&X_a + Y_{bc} \leqslant X + Y_b = Y_b + X.
\end{aligned}$$

□

Lemma 4. *For $a \in J^-$ and $b, c, d \in J^+$ with $u_b \geqslant u_c$ we have*

$$\begin{aligned}
&\min\{X_a, Y_c\} + \min\{X, Y_b\} \leqslant \min\{X_a, Y_b\} + \min\{X, Y_c\}, \qquad \textit{and}\\
&\min\{X_a, Y_c\} + \min\{X, Y_{bd}\} \leqslant \min\{X_a, Y_b\} + \min\{X, Y_{cd}\}.
\end{aligned}$$

Proof. If $X \leqslant Y_b$ then

$$\min\{X_a, Y_c\} + \min\{X, Y_b\} = X_a + X = \min\{X_a, Y_b\} + \min\{X, Y_c\}.$$

If $Y_c \leqslant X_a$ then

$$\min\{X_a, Y_c\} + \min\{X, Y_b\} = Y_c + Y_b = Y_b + Y_c = \min\{X_a, Y_b\} + \min\{X, Y_c\}.$$

In the remaining case we have (using $Y_b \leqslant Y_c$)

$$\min\{X_a, Y_c\} + \min\{X, Y_b\} = X_a + Y_b \leqslant \min\{X_a, Y_b\} + \min\{X, Y_c\}.$$

This proves the first inequality. For the second inequality, if $X \leqslant Y_{bd}$, then

$$\min\{X_a, Y_c\} + \min\{X, Y_{bd}\} = X_a + X = \min\{X_a, Y_b\} + \min\{X, Y_{cd}\}.$$

If $Y_c \leqslant X_a$, then

$$\min\{X_a, Y_c\} + \min\{X, Y_{bd}\} = Y_c + Y_{bd} = Y_b + Y_{cd} = \min\{X_a, Y_b\} + \min\{X, Y_{cd}\}.$$

In the remaining case we have $X > Y_{bd}$ and $Y_c > X_a$, hence $\min\{X_a, Y_c\} + \min\{X, Y_{bd}\} = X_a + Y_{bd}$, and we check that this is upper bounded by every possible expression on the RHS:

$$\begin{aligned} &X_a + Y_{bd} \leqslant X_a + \min\{X, Y_{cd}\}, \\ &X_a + Y_{bd} \leqslant Y_c + Y_{bd} = Y_b + Y_{cd}, \\ &X_a + Y_{bd} \leqslant X + Y_b = Y_b + X. \end{aligned}$$

□

Lemma 5. *For $a, b \in J^-$ and $c, d \in J^+$ with $u_a \geqslant u_b$ and $u_c \geqslant u_d$ we have*

$$\min\{X_a, Y_d\} + \min\{X_b, Y_c\} \leqslant \min\{X_a, Y_c\} + \min\{X_b, Y_d\}.$$

Proof. Note that $X_a \leqslant X_b$ and $Y_c \leqslant Y_d$. If $X_b \leqslant Y_c$ then

$$\min\{X_a, Y_d\} + \min\{X_b, Y_c\} = X_a + X_b = \min\{X_a, Y_c\} + \min\{X_b, Y_d\}.$$

If $X_a \geqslant Y_d$ then

$$\min\{X_a, Y_d\} + \min\{X_b, Y_c\} = Y_d + Y_c = Y_c + Y_d = \min\{X_a, Y_c\} + \min\{X_b, Y_d\}.$$

Finally, if $Y_c < X_b$ and $X_a < Y_d$ then

$$\min\{X_a, Y_d\} + \min\{X_b, Y_c\} = X_a + Y_c = \min\{X_a, Y_c\} + \max\{X_a, Y_c\} \leqslant \min\{X_a, Y_c\} + \min\{X_b, Y_d\}. \quad \square$$

Lemma 6. *For $a, b, c \in J^+$ with $u_a \geqslant u_b, u_c$ we have*

$$\begin{aligned} &\min\{X, Y_{ab}\} + \min\{X, Y\} \leqslant \min\{X, Y_a\} + \min\{X, Y_b\}, \text{ and} \\ &\min\{X, Y_{ab}\} + \min\{X, Y_c\} \leqslant \min\{X, Y_a\} + \min\{X, Y_{bc}\}. \end{aligned}$$

Proof. If $X \leqslant Y_{ab}$ then

$$\min\{X, Y_{ab}\} + \min\{X, Y\} = X + X = \min\{X, Y_a\} + \min\{X, Y_b\}.$$

If $Y \leqslant X$ then

$$\min\{X, Y_{ab}\} + \min\{X, Y\} = Y_{ab} + Y = Y_a + Y_b = \min\{X, Y_a\} + \min\{X, Y_b\}.$$

Finally, if $Y_{ab} < X < Y$ then $\min\{X, Y_{ab}\} + \min\{X, Y\} = Y_{ab} + X$ and it is easy to see that this is upper bounded by any of the possible expressions on the RHS: $X + X$, $X + Y_b$, $Y_a + X$ and $Y_a + Y_b$. This proves the first inequality, and the second one is proved similarly. □

Lemma 7. *For $a, b, c, d \in J^+$ with $u_a \geqslant u_b \geqslant u_c \geqslant u_d$ we have*

$$\min\{X, Y_{ab}\} + \min\{X, Y_{cd}\} \leqslant \min\{X, Y_{ad}\} + \min\{X, Y_{bc}\}.$$

Proof. Note that $Y_{ab} = \min\{Y_{ab}, Y_{cd}, Y_{bc}, Y_{ad}\}$ and $Y_{cd} = \max\{Y_{ab}, Y_{cd}, Y_{bc}, Y_{ad}\}$. If $X \leqslant Y_{ab}$ then

$$\min\{X, Y_{ab}\} + \min\{X, Y_{cd}\} = X + X = \min\{X, Y_{ad}\} + \min\{X, Y_{bc}\}.$$

If $X \geqslant Y_{cd}$ then

$$\min\{X, Y_{ab}\} + \min\{X, Y_{cd}\} = Y_{ab} + Y_{cd} = Y_{ad} + Y_{bc} = \min\{X, Y_{ad}\} + \min\{X, Y_{bc}\}.$$

Finally, if $Y_{ab} < X < Y_{cd}$ then $\min\{X, Y_{ab}\} + \min\{X, Y_{cd}\} = Y_{ab} + X$ and it is easy to see that this is upper bounded by the possible expressions on the RHS: $X + X$, $X + Y_{bc}$, $Y_{ad} + X$ and $Y_{ad} + Y_{bc} = Y_{ab} + Y_{cd}$. □

Lemma 8. *For $a, b, c, d \in J^+$ with $u_a \geqslant u_b \geqslant u_c \geqslant u_d$ we have*

$$\min\{X, Y_{ac}\} + \min\{X, Y_{bd}\} \leqslant \min\{X, Y_{ad}\} + \min\{X, Y_{bc}\}.$$

Proof. Note that $Y_{bd} \geqslant Y_{bc} \geqslant Y_{ac}$ and $Y_{bd} \geqslant Y_{ad} \geqslant Y_{ac}$. If $X \leqslant Y_{ac}$ then

$$\min\{X, Y_{ac}\} + \min\{X, Y_{bd}\} = X + X = \min\{X, Y_{ad}\} + \min\{X, Y_{bc}\}.$$

If $X \geqslant Y_{bd}$ then

$$\min\{X, Y_{ac}\} + \min\{X, Y_{bd}\} = Y_{ac} + Y_{bd} = Y_{ad} + Y_{bc} = \min\{X, Y_{ad}\} + \min\{X, Y_{bc}\}.$$

Finally, if $Y_{ac} < X < Y_{bd}$ then $\min\{X, Y_{ac}\} + \min\{X, Y_{bd}\} = Y_{ac} + X$ and we check that this is upper bounded by any of the possible expressions on the RHS:

$$\begin{aligned}
&Y_{ac} + X \leqslant X + X,\\
&Y_{ac} + X = X + Y_{ac} \leqslant X + Y_{bc},\\
&Y_{ac} + X \leqslant Y_{ad} + X,\\
&Y_{ac} + X \leqslant Y_{ac} + Y_{bd} = Y_{ad} + Y_{bc}.
\end{aligned}$$

□

With the help of Lemmas 1 to 8 we can now prove Proposition 2.

Proof of Proposition 2. Suppose we have an optimal schedule that is not the one described in the proposition. As long as there is any time period with two inbound jobs we can use Lemmas 1 and 2 to reduce the number of time periods with two inbound jobs without decreasing the objective value. After these modifications we can assume without loss of generality that the jobs on the arcs $a_1, a_2, \ldots, a_r$ are scheduled in the time periods $1, 2, \ldots, r$. If in time period $i \leqslant r$ no outbound job is scheduled then by Lemma 3 we can move an outbound job to period i without decreasing the objective function. Hence we can assume that in each of the time periods $1, 2, \ldots, r$ exactly one inbound and one outbound job are scheduled. If a job on one of the arcs $b_1, \ldots, b_r$ is scheduled in in a time period $i > r$, then by Lemma 4 it can be exchanged with an outbound job in a time period $j \leqslant r$ without decreasing the objective value. Hence we can assume that the jobs on the arcs $b_1, \ldots, b_r$ are scheduled in the time periods $1, \ldots, r$. If they are not scheduled in the right order we can swap outbound jobs using Lemma 5, hence we may assume that, for $i = 1, 2, \ldots, r$, the job on arc b_i is scheduled for time period i. If a job on an arc b_i with $r + 1 \leqslant i \leqslant 2T - s$ is scheduled together with a job on an arc b_j, $j > i$, then by Lemma 6 we can reschedule so that the job on arc b_i is the only job in its time period. If $s \leqslant T$ we are done at this point. So we may assume that $s > T$, for $i = r + 1, r + 2, \ldots, 2T - s$ the job on arc b_i is the only job in time period i, and in each of the time periods $2T - s + 1, \ldots, T$ there are two outbound jobs. Using Lemma 7 we can ensure that the jobs on arcs b_i for $i = 2T - s + 1, \ldots, T$ are scheduled in distinct time periods, hence we may assume that, for these i, the job on arc b_i is scheduled in time period i. Finally, Lemma 8 asserts that the jobs on arcs $b_{T+1}, \ldots, b_s$ can be reordered as described in the proposition without decreasing the objective value, and this concludes the proof. □

Since sorting the arcs dominates the run-time of the algorithm to find the solution described in Proposition 2 we obtain the following stronger run-time bound for the single-node case.

Corollary 1. *For $K = 2$ and a single transshipment node the problem can be solved on time $O(|J| \log |J|)$.*

2 Hardness results

Before proving the hardness results we make precise the definition of *series-parallel network*. In the present paper this term refers to a *two-terminal series-parallel network*: a network that has a single source and single sink and is constructed by a sequence of series and parallel compositions starting from single arcs. For two networks N_1 and N_2 the *parallel composition* of N_1 and N_2 is obtained by identifying the source node s_1 and sink node t_1 of N_1 with the source node s_2 and sink node t_2 of N_2, respectively. The *series composition* of N_1 and N_2 is obtained by identifying the sink node t_1 of N_1 with the source node s_2 of N_2. The construction of a series parallel network can be encoded into a tree, the so-called SP-tree, whose leaves are the arcs of the network. This is illustrated in Figure 2.

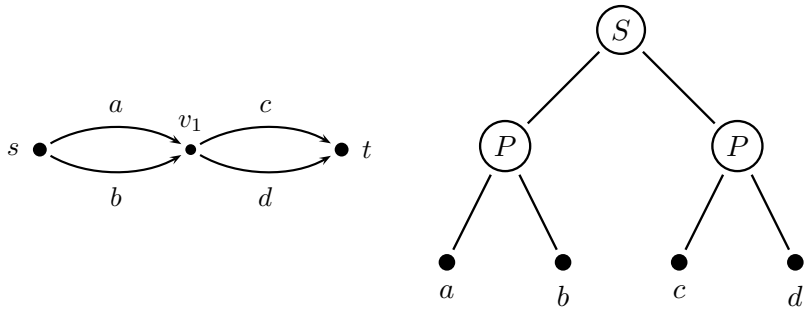

Figure 2: A series-parallel network and the corresponding SP-tree.

Proposition 3. *The restrictions of problem (1) to (8) to the instance class $\mathcal{C}^3_{sp} \cap \mathcal{C}^3_{bal} \cap \mathcal{C}^3_{aa}$ is strongly NP-complete.*

Proof. We use reduction from 3-Partition. Let a 3-Partition instance be given by an integer B and a set $\{u_1, \ldots, u_{3n}\}$ of integers with $B/4 < u_j < B/2$ for all j and $\sum_{j=1}^{3n} u_j = nB$. The problem is to decide if there is a partition of the set $\{u_1, \ldots, u_{3n}\}$ into n triples such that the sum of each triple equals B. We define new numbers u'_i for $i = 1, \ldots, 3n$ by $u'_i = 3u_i - B$. Note that $u'_i + \cdots + u'_{3n} = 0$ and

$$u_i + u_j + u_k = B \iff u'_i + u'_j + u'_k = 0.$$

Without loss of generality we can assume that for some integer r, we have $u_i \geqslant 0$ for $i \leqslant r$ and $u_i < 0$ for $i > r$. We define an instance of our problem with $K = 3$, $T = n$, a single transshipment node v and the following arcs:

- For $i = 1, 2, \ldots, r$ there is an arc into v having capacity u'_i, and
- for $i = r + 1, \ldots, 3n$ there is an out of v arc of capacity $-u'_i$.

This is illustrated in Figure 3, where the arc labels represent capacities and all arcs have an associated job, i.e. $J = A$. It follows from $u'_i + \cdots + u'_{3n} = 0$ that this network is balanced. Let X be the capacity of the network. The upper bound of $(n-1)X$ for the total throughput can be achieved if and only if the set $\{u'_i \ : \ i = 1, \ldots, 3n\}$ can be partitioned into triples that sum up to zero, or equivalently, the set $\{u_i \ : \ i = 1, \ldots, 3n\}$ can be partitioned into triples that sum up to B. □

Proposition 4. *The restriction of problem (1) to (8) to the instance class $\mathcal{C}^{|J|-1}_{sp} \cap \mathcal{C}^{|J|-1}_{bal} \cap \mathcal{C}^{|J|-1}_{aa}$ is NP-complete.*

Proof. We use reduction from Partition. Let a Partition instance be given by an integer B and a set $\{u_1, \ldots, u_n\}$ of integers with $\sum_{j=1}^{n} u_j = 2B$. The problem is to decide if there is a partition of the set $\{u_1, \ldots, u_n\}$ into two parts such that the sum of each part equals B. The network used for the reduction is shown in Figure 4, where the arc labels represent capacities and all arcs have an associated job, i.e. $J = A$. Consider this network for the time horizon $T = 2$ and with $K = n + 1$. The upper bound of $2B$ can be achieved if and only if the Partition instance is a YES instance. □

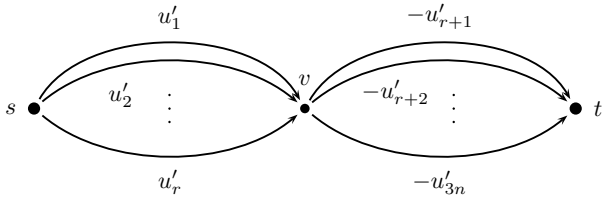

Figure 3: The network for $\mathcal{C}^3_{\text{sp}} \cap \mathcal{C}^3_{\text{bal}} \cap \mathcal{C}^3_{\text{aa}}$.

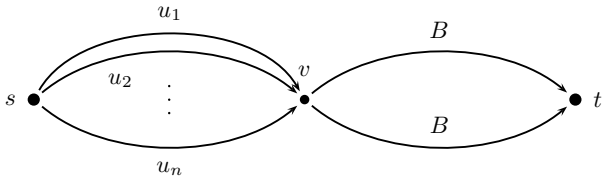

Figure 4: The network for $\mathcal{C}^{|J|-1}_{\text{sp}} \cap \mathcal{C}^{|J|-1}_{\text{bal}} \cap \mathcal{C}^{|J|-1}_{\text{aa}}$.

Note that the algorithm from [1] for series-parallel networks and $K = |J|$ which is pseudopolynomial for fixed T can be adapted to the case $K = |J| - 1$. This algorithm computes a list of T-dimensional vectors for each node of the SP-tree. The vectors at a node v of the SP-tree represent the possible throughputs for the corresponding subnetwork: $(z_1, \ldots, z_T)$ is in the list at node v if and only if the jobs for arcs in the subnetwork can be scheduled such that the maximum flow value for the subnetwork in time period i is z_i $(i = 1, \ldots, T)$. In each node of the tree we flag a vector that can only be achieved by scheduling all jobs at the same time (which is at most one per node in the tree). Finally, when we scan the list at the root node in order to determine the optimal solution, we exclude the flagged vector.

In [1], the class $\mathcal{C}_{\text{uc}}$ of instances where every arc has unit capacity was shown to be tractable when there is no limit for the number of jobs per time period. We finish this section with a proof that this class becomes NP-complete when such a limit is introduced.

Proposition 5. *The restriction of problem (1) to (8) to the instance class $\mathcal{C}_{uc}$ is NP-complete.*

Proof. We use reduction from 3-Partition. Let a 3-Partition instance be given by an integer B and a set $\{u_1, \ldots, u_{3n}\}$ of integers with $B/4 < u_j < B/2$ for all j and $\sum_{j=1}^{3n} u_j = nB$. This can be reduced to the instance presented in Figure 5, where every arc has unit capacity and the set J is represented by dashed arcs. Since 3-Partition is strongly NP-hard we may assume that the numbers u_i are bounded by a polynomial in

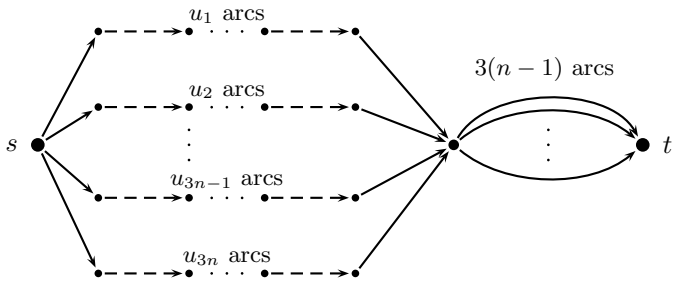

Figure 5: Instance for the reduction in the proof of Proposition 5. The dashed arcs indicate the set J of arcs with an associated job.

the input size, and this ensures that the network size is polynomial in the size of the 3-Partition instance. We consider this network with a time horizon $T = n$ and a bound of $K = B$ jobs per time period. From $|J| = nB$ it follows that exactly B jobs have to be scheduled in each time period. The proof is finished by the observation that the upper bound of $3n(n-1)$ on the total throughput can be achieved if and only if the 3-Partition instance is a YES instance. □

3 An FPTAS for series-parallel networks with fixed T

In this section we restrict our attention to series-parallel networks. We modify the algorithm from [1] such that the bound K can be taken into account. For fixed time horizon T, this algorithm runs in pseudopolynomial time, and we use it together with scaling and rounding [9] to design an FPTAS.

The algorithm presented in [1] starts at the leaves of the SP-tree and computes a list of vectors $z = (z_1, \ldots, z_T)$ for each node of the SP-tree, where the list at a node v in the SP-tree contains exactly the vectors z such that there exists some schedule for which the subnetwork corresponding to v can carry flow z_i in time period i for $i = 1, \ldots, T$. In the problem variant studied in [1] there is no restriction on the number of arcs that can be shut in a period, so it is sufficient to keep track of the possible flow vectors at the nodes of the SP-tree. But the same capacity vector can be realised through different schedules. For instance, for the network shown in in Figure 6, there are three possibilities to get the flow vector $(7, 0)$, i.e. 7 units in the first time period and zero flow in the second period:

- shut 2 arcs in period 1 (arcs with capacities 1 and 2), and 2 arcs in period 2 (arcs with capacities 8 and 7); or
- shut 1 arc in period 1 (arc with capacity 1 or 2), and 3 arcs in period 2 (arcs with capacities 8, 7 and (2 or 1)); or
- shut no arc in period 1, and all four arcs in period 2.

Thus with a limit K for the number of shut arcs per time period it becomes important to keep track of the number of arcs shut in each period along with maximum flow that can be sent in that period. Let j_i represent the number of arcs shut in the i^{th} period. We determine a lists of *job-capacity* vectors of the form $z = ((j_1, z_1), (j_2, z_2), \ldots, (j_T, z_T))$ at each node of the SP-tree. The interpretation of such a vector z in the list of node N is that there is a solution in which, for $i = 1, \ldots, T$, in time period i exactly j_i arcs from the subnetwork corresponding to N are shut, and this subnetwork has capacity z_i. Due to the symmetry with respect to the time periods it is no loss of generality to require the job-capacity vectors to be ordered. Hence we consider only vectors that satisfy, for $i = 1, \ldots, T-1$, either $z_i > z_{i+1}$ or $z_i = z_{i+1}$ and $j_i \geqslant j_{i+1}$. We say that a vector with this property is in *standard form*, and we note that for every job-capacity vector there is a unique vector in standard form which can be obtained by a permutation of the entries. The list at a leaf node of the tree, corresponding to an arc a of the network, consists of the unique vector $((0, u_a), (0, u_a), \ldots, (0, u_a), (1, 0))$ if $a \in J$ or $((0, u_a), (0, u_a), \ldots, (0, u_a), (0, u_a))$ if $a \notin J$. As in [1], let $\mathcal{L}$ and $\mathcal{W}$ denote the sets of leaves and internal nodes of the SP-tree, and let $\mathcal{W}_i$ $(i = 0, \ldots, d)$ be the set of internal nodes at distance i from the root. The lists of job-capacity vectors are computed as described in the Algorithm 1.

Example 1. Consider the series-parallel graph in Figure 6 where arc labels indicate capacities, all arcs need maintenance for a period over a time horizon of 2 periods. Suppose that $K = 3$. In Figure 7, we show how job-capacity vectors are computed in the SP-tree.

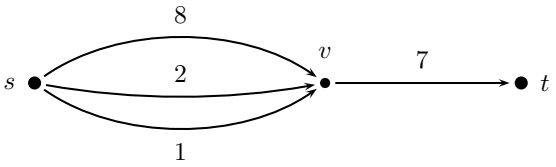

Figure 6: Example network.

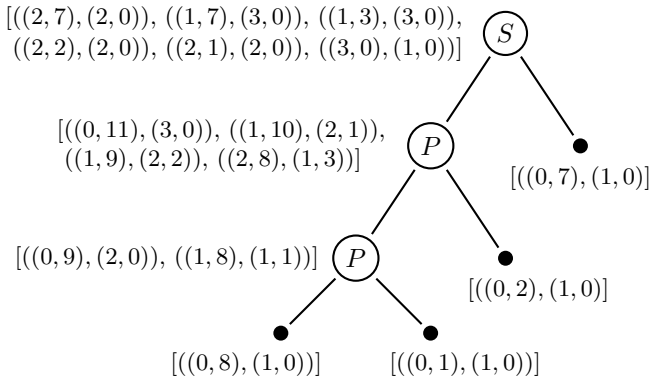

Figure 7: Computation of job-capacity vectors.

Proposition 6. *Let m be the number of arcs, B be an upper bound for the capacities and K be the limit on the number of arcs that can be shut in a period. For series-parallel networks the problem (1) to (8) can be solved in time $O(T \log T (KmB)^{2T} T! m)$.*

Algorithm 1 Maximizing total throughput for series-parallel networks under uniform maintenance limit K

for $v \in \mathcal{L}$ **do**
  Let $a \in A$ be the arc corresponding to v
  if $a \in J$ **then** $L_v \leftarrow [((0, u_a), (0, u_a), \ldots, (0, u_a), (1, 0))]$
  else $L_v \leftarrow [((0, u_a), (0, u_a), \ldots, (0, u_a), (0, u_a))]$
for $i = d, d-1, \ldots, 0$ **do**
  for $v \in \mathcal{W}_i$ **do**
    $L_v \leftarrow []$ {initialize empty list}
    Let u and w be the child nodes of v
    for $(z, z') \in L_u \times L_w$ and π permutation of $\{1, 2 \ldots, T\}$ **do**
      for $i \in [T]$ **do** $j''_i = j_i + j'_{\pi(i)}$
      if $j''_i \leqslant K$ for all $i \in [T]$ **then**
        if v is a parallel composition node **then**
          for $i \in [T]$ **do** $z''_i = z_i + z'_{\pi(i)}$
        else
          for $i \in [T]$ **do** $z''_i = \min\{z_i, z'_{\pi(i)}\}$
        sort z'' to get the corresponding canonical vector
        if $z'' \notin L_v$ **then** add z'' to L_v
Let v be the root node
return $\max_{z \in L_v} \sum_{i=1}^{T} z_i$

Proof. The first and second component of an entry of a vector in the list at an internal node are bounded by K and mB respectively, hence each entry can take KmB possible values. Therefore every list can contain at most $(KmB)^T$ elements. Thus the loop over $(z, z') \in L_u \times L_w$ and permutations π is over at most $T!(KmB)^{2T}$ elements. If hash tables are used for the check of $z'' \in L_v$ then the bound of $O(T \log T)$ for sorting z'' dominates the run-time of the loop. In total there are $m - 1$ internal nodes, thus the run-time of the complete algorithm is $O(T \log T (KmB)^{2T} T! m)$. □

From Proposition 6, it follows that for fixed T the problem (1) to (8) on series-parallel networks can be solved in $O(m^{2T+1} B^{2T} K^{2T})$ time where B is the maximum capacity of an arc in the network. Now we use a scaling approach to derive a fully polynomial approximation scheme (FPTAS), that is a family $(\mathcal{A}_\varepsilon)$ of algorithms, parameterized by a positive real number ε, such that algorithm $\mathcal{A}_\varepsilon$ produces a solution with objective value at least $(1 - \varepsilon)z^*$, where z^* is the optimal value, and the run-time of algorithm $\mathcal{A}_\varepsilon$ is polynomially bounded in the input size and $1/\varepsilon$.

Our approximation scheme is based on scaling the problem such that the maximum capacity becomes bounded. In order to ensure that the solution of the scaled problem is sufficiently close to the optimum we need a lower bound for the optimal objective value. If $|J| \leqslant K(T-1)$ there is a feasible solution having one time period without any outage, and the flow value for such a time period will be sufficient as lower bound for our purpose. For $|J| > K(T-1)$ the situation is more complicated, and we need a preprocessing step to transform a given instance into an equivalent one with some control on the maximum capacity. Let $\rho = \max\{0, |J| - K(T-1)\} \in \{0, 1, \ldots, K\}$, and let M be the maximum flow value with ρ arcs closed. For $\rho = 0$, M is the capacity of a minimum cut and can be computed by solving a max flow problem. For $\rho > 0$, the computation of M is described in Algorithm 2. Here, for a node v in the SP-tree and a number $j \in \{0, 1, \ldots, \rho\}$, z_v^j is the capacity of the subnetwork corresponding to node v when j arcs in the intersection of J and this subnetwork are closed. If j is larger than the size of this intersection, we put $z_v^j = -\infty$. Algorithm 2 shows that M can be computed efficiently.

Lemma 9. *The maximum flow value M subject to the constraint that ρ arcs from J carry zero flow can be determined in time $O(mK^2) = O(m^3)$.* □

No arc can carry more than M units of flow in any time period, hence we may assume w.l.o.g. that

Algorithm 2 Computing the maximum flow M with ρ outages

for $v \in \mathcal{L}$ **do**
  Let $a \in A$ be the arc corresponding to v
  $z_v^0 \leftarrow u_a$
  if $a \in J$ **then** $z_v^1 \leftarrow 0$ **else** $z_v^1 \leftarrow -\infty$
  for $j = 2, \dots, \rho$ **do** $z_v^j \leftarrow -\infty$
for $i = d, d-1, \dots, 0$ **do**
  for $v \in \mathcal{W}_i$ **do**
    for $j = 0, 1, \dots, \rho$ **do** $z_v^j \leftarrow -\infty$
    Let u and w be the child nodes of v
    for $j = 0, 1, \dots, \rho$ **do**
      for $j' = 0, 1, \dots, \rho - j$ **do**
        if v is a parallel composition node **then**
          $z_v^{j+j'} \leftarrow \max\{z_v^{j+j'}, z_u^j + z_w^{j'}\}$
        else $\{v$ is a series composition node$\}$
          $z_v^{j+j'} \leftarrow \max\{z_v^{j+j'}, \min\{z_u^j, z_w^{j'}\}\}$
Let v be the root node
return $M = z_v^\rho$

$B \leqslant M$. We also know that the optimal objective value is at least M because, we can schedule ρ jobs allowing a flow of value M in time period 1, and then continue arbitrarily. Let $L = \max\{1, \varepsilon B/(mT)\}$ and consider the scaled problem with the capacities u_a replaced by $u'_a = \lfloor u_a/L \rfloor$. The scaled instance can be solved in time

$$O(m^{2T+1}(B/L)^{2T}K^{2T}) = O(m^{4T+1}K^{2T}/\varepsilon^{2T}).$$

For any feasible vector $y = (y_{ai})_{a\in A, i\in[T]} \in \{0,1\}^{|J|\,T}$, let $F(y)$ and $F'(y)$ denote the objective values for the problem on the original network and for the scaled version, respectively. Let $y^* = (y^*_{ai})_{a\in A, i\in[T]}$ and $\tilde{y} = (\tilde{y}_{ai})_{a\in A, i\in[T]}$ denote optimal solutions of the problem on the original network and of the scaled version, respectively. In the following lemma, we study the the behaviour of the objective values for these solutions under the scaling.

Lemma 10. *We have the following estimates:*

$$L \cdot F'(y^*) \geqslant (1-\varepsilon)F(y^*), \tag{9}$$
$$F(\tilde{y}) \geqslant L \cdot F'(\tilde{y}). \tag{10}$$

Proof. Both inequalities are obvious for for $L = 1$, because in this case the original and the scaled problem coincide. So we assume $L > 1$. For $i = 1, \dots, T$ let C_i be a minimum cut in the network (V, A^*_i, s, t, u') where $A^*_i = \{a \in A : y^*_{ai} = 1\}$. Then, using $B \leqslant M \leqslant F(y^*)$, we obtain

$$L \cdot F'(y^*) = L\sum_{i=1}^{T}\sum_{a\in C_i} u'_a \geqslant L\sum_{i=1}^{T}\left(\sum_{a\in C_i}\frac{u_a}{L} - |C_i|\right) \geqslant \sum_{i=1}^{T}\sum_{a\in C_i} u_a - LmT$$
$$= \sum_{i=1}^{T}\sum_{a\in C_i} u_a - \varepsilon B \geqslant (1-\varepsilon)F(y^*).$$

Similarly, let C'_i be a minimum cut in the network $(V, \tilde{A}_i, s, t, u)$ where $\tilde{A}_i = \{a \in A : \tilde{y}_{ai} = 1\}$. Then

$$F(\tilde{y}) = \sum_{i=1}^{T}\sum_{a\in C'_i} u_{ai} \geqslant L\sum_{i=1}^{T}\sum_{a\in C'_i} u'_{ai} \geqslant LF'(\tilde{y}). \qquad \square$$

Proposition 7. *For fixed T, the class $\mathcal{C}_{sp}$ of instances with a series-parallel network has an FPTAS with run-time $O(m^{2T+1}(B/L)^{2T}K^{2T}) = O(m^{4T+1}K^{2T}/\varepsilon^{2T}) = O(m^{6T+1}/\varepsilon^{2T})$.*

Proof. The run-time bound for the scaled problem is a consequence of Proposition 6, and the approximation guarantee follows from (9) and (10): $F(\tilde{y}) \geqslant LF'(\tilde{y}) \geqslant LF'(y^*) \geqslant (1-\varepsilon)F(y^*)$. □

Remark 1. The problem can be generalized by allowing the bound on the number of jobs to vary over time. In other words, the parameter K is replaced by a vector $(K_1, \ldots, K_T)$ and constraints (6) are replaced by

$$\sum_{a \in J} y_{ai} \geqslant |J| - K_i \qquad \text{for all } i \in [T].$$

Algorithm 1 can be modified to solve this more general problem, and with

$$\rho = \max\left\{0,\ |J| - \sum_{i=1}^{T} K_i + \min_{i \in [T]} K_i\right\}$$

we obtain an FPTAS of runtime $O(m^{6T+1}/\varepsilon^{2T})$ for this problem.

For $K = |J|$, it was shown in [1] that the method corresponding to Algorithm 7 runs in time $O(m^{2T-1}B^{2T-2})$, and using the same argument as above, we obtain the following approximation result.

Proposition 8. *For fixed T, $K = |J|$ and series-parallel networks, the problem (1) to (8) has an FPTAS with run-time $O(m^{2T-1}(B/L)^{2T-2}) = O(m^{4T-3}/\varepsilon^{2T-2})$.* □

Since for $K = |J|$ shutting all arcs in the job set J at the same time gives an approximation ratio of $(1 - 1/T)$, we have the following result for arbitrary T.

Corollary 2. *For $K = |J|$ and series-parallel networks, the problem (1) to (8) has a PTAS with run-time*

$$O\left(f(1/\varepsilon)m^{4/\varepsilon-3}\right)$$

where $f(x) = x^{5x-5/2}e^x \log x$.

Proof. Let $\varepsilon > 0$ be fixed. If $1/T \leqslant \varepsilon$ we schedule all jobs at time 1. Otherwise $T < 1/\varepsilon$ and we run the ε-approximation algorithm for T. By the argument in [1], the run-time is bounded by

$$\begin{aligned} O\left(T\log(T)T!(mB/L+1)^{2(T-1)}m\right) &= O\left(T\log(T)T!\left(\frac{m^2T}{\varepsilon}+1\right)^{2(T-1)}m\right) \\ &= O\left((mT)^{4T-3}\log(T)T!\left(\frac{1}{\varepsilon T}+\frac{1}{(mT)^2}\right)^{2(T-1)}\right). \quad (11) \end{aligned}$$

We have

$$\left(\frac{1}{\varepsilon T}+\frac{1}{(mT)^2}\right)^{2(T-1)} = \left(\frac{m^2T+\varepsilon}{\varepsilon(mT)^2}\right)^{2(T-1)} = \left(1+\frac{m^2T+\varepsilon-\varepsilon(mT)^2}{\varepsilon(mT)^2}\right)^{2(T-1)}$$

With $\alpha = m^2T + \varepsilon - \varepsilon(mT)^2$ and $\beta = \varepsilon(mT)^2$ we obtain

$$\left(\frac{1}{\varepsilon T}+\frac{1}{(mT)^2}\right)^{2(T-1)} = \left[\left(1+\frac{\alpha}{\beta}\right)^{\beta/\alpha}\right]^{2(T-1)\alpha/\beta} \leqslant e^{2(T-1)\alpha/\beta}.$$

Now

$$2(T-1)\frac{\alpha}{\beta} \leqslant 2T\cdot\frac{m^2T+\varepsilon-\varepsilon(mT)^2}{\varepsilon(mT)^2} = 2\cdot\frac{m^2T(1-\varepsilon T)+\varepsilon}{\varepsilon m^2T} \leqslant 2/\varepsilon+1,$$

and this implies

$$\left(\frac{1}{\varepsilon T}+\frac{1}{(mT)^2}\right)^{2(T-1)} = O(e^{2/\varepsilon}).$$

Substituting into (11) yields a run-time bound of

$$O\left((mT)^{4T-3}\log(T)T!e^{2/\varepsilon}\right),$$

and since all terms are increasing in T, we get with $T < 1/\varepsilon$ and using Stirling's formula to bound the factorial, that the run-time is bounded by

$$\begin{aligned} O\left((1/\varepsilon)^{4/\varepsilon-3}\log(1/\varepsilon)\lceil 1/\varepsilon\rceil!e^{2/\varepsilon}m^{4/\varepsilon-3}\right) &= O\left((1/\varepsilon)^{4/\varepsilon-3}\log(1/\varepsilon)(1/\varepsilon)^{1/\varepsilon}e^{-1/\varepsilon}\sqrt{1/\varepsilon}e^{2/\varepsilon}m^{4/\varepsilon-3}\right) \\ &= O\left((1/\varepsilon)^{5/\varepsilon-5/2}\log(1/\varepsilon)e^{1/\varepsilon}m^{4/\varepsilon-3}\right) \quad \square \end{aligned}$$

References

[1] N. Boland, T. Kalinowski, R. Kapoor, and S. Kaur. Scheduling unit time arc shutdowns to maximize network flow over time: complexity results. *Networks*, to appear. `arXiv:1306.4917`.

[2] N. Boland, T. Kalinowski, H. Waterer, and L. Zheng. Mixed integer programming based maintenance scheduling for the Hunter Valley Coal Chain. *Journal of Scheduling*, 2012. in press, `doi:10.1007/s10951-012-0284-y`.

[3] N. Boland, T. Kalinowski, H. Waterer, and L. Zheng. Scheduling arc maintenance jobs in a network to maximize total flow over time. *Discrete Applied Mathematics*, 2012. in press, `doi:10.1016/j.dam.2012.05.027`.

[4] N. Boland and M. W. P. Savelsbergh. Optimizing the Hunter Valley coal chain. In H. Gurnani, A. Mehrotra, and S. Ray, editors, *Supply Chain Disruptions: Theory and Practice of Managing Risk*. Springer-Verlag London Ltd., 2011.

[5] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17(3):449–467, 1965.

[6] H. N. Gabow. Data structures for weighted matching and nearest common ancestors with linking. In *Proc. 1st ACM-SIAM Symp. on Discrete Algorithms, SODA 1990*, pages 434–443, 1990.

[7] V. King, S. Rao, and R. Tarjan. A faster deterministic maximum flow algorithm. *Journal of Algorithms*, 17(3):447–474, 1994.

[8] J. B. Orlin. Max flows in $O(nm)$ time or better. *to appear in Proc. 45th ACM Symp. on Theory of Computing (STOC 2013)*, online: `http://jorlin.scripts.mit.edu/docs/papersfolder/O(nm)MaxFlow.pdf`, 2013.

[9] D. P. Williamson and D. B. Shmoys. *The design of approximation algorithms*. Cambridge University Press, 2011.