

# Scheduling Maintenance Jobs in Networks

Fidaa Abed<sup>1</sup>, Lin Chen<sup>2</sup>, Yann Disser<sup>3</sup>, Martin Groß<sup>4(✉)</sup>, Nicole Megow<sup>5</sup>,  
Julie Meißner<sup>6</sup>, Alexander T. Richter<sup>7</sup>, and Roman Rischke<sup>8</sup>

<sup>1</sup> University of Jeddah, Jeddah, Saudi Arabia  
fabad@uj.edu.sa

<sup>2</sup> University of Houston, Houston, TX, USA  
chenlin198662@gmail.com

<sup>3</sup> TU Darmstadt, Darmstadt, Germany  
disser@mathematik.tu-darmstadt.de

<sup>4</sup> University of Waterloo, Waterloo, ON, Canada  
mgrob@uwaterloo.ca

<sup>5</sup> University of Bremen, Bremen, Germany  
nicole.megow@uni-bremen.de

<sup>6</sup> TU Berlin, Berlin, Germany  
jmeiss@math.tu-berlin.de

<sup>7</sup> TU Braunschweig, Braunschweig, Germany  
a.richter@tu-bs.de

<sup>8</sup> TU München, Munich, Germany  
rischke@ma.tum.de

**Abstract.** We investigate the problem of scheduling the maintenance of edges in a network, motivated by the goal of minimizing outages in transportation or telecommunication networks. We focus on maintaining connectivity between two nodes over time; for the special case of path networks, this is related to the problem of minimizing the busy time of machines.

We show that the problem can be solved in polynomial time in arbitrary networks if preemption is allowed. If preemption is restricted to integral time points, the problem is NP-hard and in the non-preemptive case we give strong non-approximability results. Furthermore, we give tight bounds on the power of preemption, that is, the maximum ratio of the values of non-preemptive and preemptive optimal solutions.

Interestingly, the preemptive and the non-preemptive problem can be solved efficiently on paths, whereas we show that mixing both leads to a weakly NP-hard problem that allows for a simple 2-approximation.

**Keywords:** Scheduling · Maintenance · Connectivity · Complexity theory · Approximation algorithm

---

This work is supported by the German Research Foundation (DFG) under project ME 3825/1 and within project A07 of CRC TRR 154. It is partially funded in the framework of MATHEON supported by the Einstein Foundation Berlin and by the Alexander von Humboldt Foundation. A version with proofs and illustrations is available on arXiv.

# 1 Introduction

Transportation and telecommunication networks are important backbones of modern infrastructure and have been a major focus of research in combinatorial optimization and other areas. Research on such networks usually concentrates on optimizing their usage, for example by maximizing throughput or minimizing costs. In the majority of the studied optimization models it is assumed that the network is permanently available, and our choices only consist in deciding which parts of the network to use at each point in time.

Practical transportation and telecommunication networks, however, can generally not be used non-stop. Be it due to wear-and-tear, repairs, or modernizations of the network, there are times when parts of the network are unavailable. We study how to schedule and coordinate such maintenance in different parts of the network to ensure connectivity.

While network problems and scheduling problems individually are fairly well understood, the combination of both areas that results from scheduling network maintenance has only recently received some attention [1, 2, 4, 11, 16] and is theoretically hardly understood.

**Problem Definition.** In this paper, we study connectivity problems which are fundamental in this context. In these problems, we aim to schedule the maintenance of edges in a network in such a way as to preserve connectivity between two designated vertices. Given a network and maintenance jobs with processing times and feasible time windows, we need to decide on the temporal allocation of the maintenance jobs. While a maintenance on an edge is performed, the edge is not available. We distinguish between **MINCONNECTIVITY**, the problem in which we minimize the total time in which the network is disconnected, and **MAXCONNECTIVITY**, the problem in which we maximize the total time in which it is connected.

In both of these problems, we are given an undirected graph  $G = (V, E)$  with two distinguished vertices  $s^+, s^- \in V$ . We assume w. l. o. g. that the graph is simple; we can replace a parallel edge  $\{u, w\}$  by a new node  $v$  and two edges  $\{u, v\}, \{v, w\}$ . Every edge  $e \in E$  needs to undergo  $p_e \in \mathbb{Z}_{\geq 0}$  time units of maintenance within the time window  $[r_e, d_e]$  with  $r_e, d_e \in \mathbb{Z}_{\geq 0}$ , where  $r_e$  is called the release date and  $d_e$  is called the deadline of the maintenance job for edge  $e$ . An edge  $e = \{u, v\} \in E$  that is maintained at time  $t$ , is not available at  $t$  in the graph  $G$ . We consider preemptive and non-preemptive maintenance jobs. If a job must be scheduled non-preemptively then, once it is started, it must run until completion without any interruption. If a job is allowed to be preempted, then its processing can be interrupted at any time and may resume at any later time without incurring extra cost.

A *schedule*  $S$  for  $G$  assigns the maintenance job of every edge  $e \in E$  to a single time interval (if non-preemptive) or a set of disjoint time intervals (if preemptive)  $S(e) := \{[a_1, b_1], \dots, [a_k, b_k]\}$  with  $r_e \leq a_i \leq b_i \leq d_e$  for  $i \in [k]$  and  $\sum_{[a,b] \in S(e)} (b - a) = p_e$ . If not specified differently, we define  $T := \max_{e \in E} d_e$  as our *time horizon*. We do not limit the number of simultaneously maintained edges.

For a given maintenance schedule, we say that the network  $G$  is *disconnected at time  $t$*  if there is no path from  $s^+$  to  $s^-$  in  $G$  at time  $t$ , otherwise we call the network  $G$  *connected at time  $t$* . The goal is to find a maintenance schedule for the network  $G$  so that the total time where  $G$  is disconnected is minimized (MINCONNECTIVITY). We also study the maximization variant of the problem, in which we want to find a schedule that maximizes the total time where  $G$  is connected (MAXCONNECTIVITY).

**Our Results.** For *preemptive* maintenance jobs, we show that we can solve both problems, MAXCONNECTIVITY and MINCONNECTIVITY, efficiently in arbitrary networks (Theorem 1). This result crucially requires that we are free to preempt jobs at arbitrary points in time. Under the restriction that we can *preempt* jobs only at *integral points in time*, the problem becomes NP-hard. More specifically, MAXCONNECTIVITY does not admit a  $(2 - \epsilon)$ -approximation algorithm for any  $\epsilon > 0$  in this case, and MINCONNECTIVITY is inapproximable (Theorem 2), unless  $P = NP$ . By inapproximable, we mean that it is NP-complete to decide whether the optimal objective value is zero or positive, leading to unbounded approximation factors.

This is true even for unit-size jobs. This complexity result is interesting and may be surprising, as it is in contrast to results for standard scheduling problems, without an underlying network. Here, the restriction to integral preemption typically does not increase the problem complexity when all other input parameters are integral. However, the same question remains open in a related problem concerning the busy-time in scheduling, studied in [7, 8].

For *non-preemptive* instances, we establish that there is no  $(c\sqrt[3]{|E|})$ -approximation algorithm for MAXCONNECTIVITY for some constant  $c > 0$  and that MINCONNECTIVITY is inapproximable even on disjoint paths between two nodes  $s$  and  $t$ , unless  $P = NP$  (Theorems 3 and 4). On the positive side, we provide an  $(\ell + 1)$ -approximation algorithm for MAXCONNECTIVITY in general graphs (Theorem 6), where  $\ell$  is the number of distinct latest start times (deadline minus processing time) for jobs.

We use the notion *power of preemption* to capture the benefit of allowing arbitrary job preemption. The power of preemption is a commonly used measure for the impact of preemption in scheduling [6, 10, 18, 19]. Other terms used in this context include *price of non-preemption* [9], *benefit of preemption* [17] and *gain of preemption* [12]. It is defined as the maximum ratio of the objective values of an optimal non-preemptive and an optimal preemptive solution. We show that the power of preemption is  $\Theta(\log |E|)$  for MINCONNECTIVITY on a path (Theorem 7) and unbounded for MAXCONNECTIVITY on a path (Theorem 8). This is in contrast to other scheduling problems, where the power of preemption is constant, e. g. [10, 18].

On paths, we show that *mixed* instances, which have both preemptive and non-preemptive jobs, are weakly NP-hard (Theorem 9). This hardness result is of particular interest, as both purely non-preemptive and purely preemptive instances can be solved efficiently on a path (see Theorem 1 and [14]).

Furthermore, we give a simple 2-approximation algorithm for mixed instances of MINCONNECTIVITY (Theorem 10).

**Related Work.** The concept of combining scheduling with network problems has been considered by different communities lately. However, the specific problem of only maintaining connectivity over time between two designated nodes has not been studied to our knowledge. Boland et al. [2–4] study the combination of non-preemptive arc maintenance in a transport network, motivated by annual maintenance planning for the Hunter Valley Coal Chain [5]. Their goal is to schedule maintenance such that the maximum  $s$ - $t$ -flow over time in the network with zero transit times is maximized. They show strong NP-hardness for their problem and describe various heuristics and IP based methods to address it. Also, they show in [3] that in their non-preemptive setting, if the input is integer, there is always an optimal solution that starts all jobs at integer time points. In [2], they consider a variant of their problem, where the number of concurrently performable maintenances is bounded by a constant.

Their model generalizes ours in two ways – it has capacities and the objective is to maximize the total flow value. As a consequence of this, their IP-based methods carry over to our setting, but these methods are of course not efficient. Their hardness results do not carry over, since they rely on the capacities and the different objective. However, our hardness results – in particular our approximation hardness results – carry over to their setting, illustrating why their IP-based models are a good approach for some of these problems.

Bley et al. [1] study how to upgrade a telecommunication network to a new technology employing a bounded number of technicians. Their goal is to minimize the total service disruption caused by downtimes. A major difference to our problem is that there is a set of given paths that shall be upgraded and a path can only be used if it is either completely upgraded or not upgraded. They give ILP-based approaches for solving this problem and show strong NP-hardness for a non-constant number of paths by reduction from the linear arrangement problem.

Nurre et al. [16] consider the problem of restoring arcs in a network after a major disruption, with restoration per time step being bounded by the available work force. Such network design problems over time have also been considered by Kalinowski et al. [13].

In scheduling, minimizing the busy time refers to minimizing the amount of time for which a machine is used. Such problems have applications for instance in the context of energy management [15] or fiber management in optical networks [11]. They have been studied from the complexity and approximation point of view in [7, 11, 14, 15]. The problem of minimizing the busy time is equivalent to our problem in the case of a path, because there we have connectivity at a time point when no edge in the path is maintained, i. e., no machine is busy.

Thus, the results of Khandekar et al. [14] and Chang et al. [7] have direct implications for us. They show that minimizing busy time can be done efficiently for purely non-preemptive and purely preemptive instances, respectively.

## 2 Preemptive Scheduling

In this section, we consider problem instances where all maintenance jobs can be preempted.

**Theorem 1.** *Both MAXCONNECTIVITY and MINCONNECTIVITY with preemptive jobs can be solved optimally in polynomial time on arbitrary graphs.*

*Proof.* We establish a linear program (LP) for MAXCONNECTIVITY. Let  $TP = \{0\} \cup \{r_e, d_e : e \in E\} = \{t_0, t_1, \dots, t_k\}$  be the set of all *relevant time points* with  $t_0 < t_1 < \dots < t_k$ . We define  $I_i := [t_{i-1}, t_i]$  and  $w_i := |I_i|$  to be the length of interval  $I_i$  for  $i = 1, \dots, k$ .

In our linear program we model connectivity during interval  $I_i$  by an  $(s^+, s^-)$ -flow  $x^{(i)}$ ,  $i \in \{1, \dots, k\}$ . To do so, we add for every undirected edge  $e = \{u, v\}$  two directed arcs  $(u, v)$  and  $(v, u)$ . Let  $A$  be the resulting arc set. With each edge/arc we associate a capacity variable  $y_e^{(i)}$ , which represents the fraction of availability of edge  $e$  in interval  $I_i$ . Hence,  $1 - y_e^{(i)}$  gives the relative amount of time spent on the maintenance of edge  $e$  in  $I_i$ . Additionally, the variable  $f_i$  expresses the fraction of availability for interval  $I_i$ .

$$\max \quad \sum_{i=1}^k w_i \cdot f_i \quad (1)$$

$$\text{s.t.} \quad \sum_{u:(v,u) \in A} x_{(v,u)}^{(i)} - \sum_{u:(u,v) \in A} x_{(u,v)}^{(i)} = \begin{cases} f_i & \forall i \in [k], v = s^+, \\ 0 & \forall i \in [k], v \in V \setminus \{s^+, s^-\}, \\ -f_i & \forall i \in [k], v = s^-, \end{cases} \quad (2)$$

$$\sum_{i: I_i \subseteq [r_e, d_e]} (1 - y_e^{(i)}) w_i \geq p_e \quad \forall e \in E, \quad (3)$$

$$x_{(u,v)}^{(i)}, x_{(v,u)}^{(i)} \leq y_{\{u,v\}}^{(i)} \quad \forall i \in [k], \{u, v\} \in E, \quad (4)$$

$$f_i \leq 1 \quad \forall i \in [k], \quad (5)$$

$$x_{(u,v)}^{(i)}, x_{(v,u)}^{(i)}, y_{\{u,v\}}^{(i)} \in [0, 1] \quad \forall i \in [k], \{u, v\} \in E. \quad (6)$$

Notice that the LP is polynomial in the input size, since  $k \leq 2|E|$ . We show in Lemma 1 that this LP is a relaxation of preemptive MAXCONNECTIVITY on general graphs and in Lemma 2 that any optimal solution to it can be turned into a feasible schedule with the same objective function value in polynomial time, which proves the claim for MAXCONNECTIVITY. For MINCONNECTIVITY, notice that any solution that maximizes the time in which  $s$  and  $t$  are connected also minimizes the time in which  $s$  and  $t$  are disconnected – thus, we can use the above LP there as well.  $\square$

**Lemma 1.** *The given LP is a relaxation of preemptive MAXCONNECTIVITY on general graphs.*

**Lemma 2.** *Any feasible LP solution can be turned into a feasible maintenance schedule at no loss in the objective function value in polynomial time.*

The statement of Theorem 1 crucially relies on the fact that we may preempt jobs arbitrarily. However, if preemption is only possible at integral time points, the problem becomes NP-hard even for unit-size jobs. This follows from the proof of Theorem 3 for  $t_1 = 0$ ,  $t_2 = 1$ , and  $T = 2$ .

**Theorem 2.** *MAXCONNECTIVITY with preemption only at integral time points is NP-hard and does not admit a  $(2 - \epsilon)$ -approximation algorithm for any  $\epsilon > 0$ , unless  $P = NP$ . Furthermore, MINCONNECTIVITY with preemption only at integral time points is inapproximable.*

### 3 Non-preemptive Scheduling

We consider problem instances in which no job can be preempted. We show that there is no  $(c\sqrt[3]{|E|})$ -approximation algorithm for MAXCONNECTIVITY for some  $c > 0$ . We also show that MINCONNECTIVITY is inapproximable, unless  $P = NP$ . Furthermore, we give an  $(\ell + 1)$ -approximation algorithm, where  $\ell := |\{d_e - p_e \mid e \in E\}|$  is the number of distinct latest start times for jobs.

To show the strong hardness of approximation for MAXCONNECTIVITY, we begin with a weaker result which provides us with a crucial gadget.

**Theorem 3.** *Non-preemptive MAXCONNECTIVITY does not admit a  $(2 - \epsilon)$ -approximation algorithm, for  $\epsilon > 0$ , and non-preemptive MINCONNECTIVITY is inapproximable, unless  $P = NP$ . This holds even for unit-size jobs.*

*Proof (Sketch).* This is shown by a reduction from 3SAT. We construct a network such that connectivity is possible only within two disjoint time slots  $[t_1, t_1 + 1]$  and  $[t_2, t_2 + 1]$ .

We show that this network admits a schedule with total connectivity time greater than one if and only if the 3SAT-instance is a YES-instance. Furthermore, we show that if the total connectivity time is greater than one, then there is a schedule with maximum total connectivity time of two. For this, we distinguish between *variable paths* and *clause paths*. By construction, variable paths exist only in  $[t_2, t_2 + 1]$  and clause paths only in  $[t_1, t_1 + 1]$ . These paths walk through variable gadgets which encapsulate the decision whether to set a variable to TRUE or FALSE. A variable path ensures that we have a valid variable assignment, and a clause path sets literals in a clause to TRUE. If and only if both types of paths exist, then the 3SAT-instance is a YES-instance.

For  $t_1 = 0$ ,  $t_2 = 1$ , and  $T = 2$ , this construction uses only unit-size jobs, and in the MINCONNECTIVITY case YES-instances have an objective value of 0 and NO-instances a value of 1.  $\square$

We reuse the construction in the proof of Theorem 3 repeatedly to obtain the following improved lower bound.

**Theorem 4.** *Unless  $P = NP$ , there is no  $(c\sqrt[3]{|E|})$ -approximation algorithm for non-preemptive MAXCONNECTIVITY, for some constant  $c > 0$ .*

*Proof (Sketch).* We show this by reduction from 3SAT. Let  $n$  be the number of variables in the given 3SAT instance. Using the construction from Theorem 3 repeatedly allows us to construct a network that has maximum connectivity time  $n$  if the given 3SAT instance is a YES-instance and maximum connectivity time 1 otherwise. This implies that there cannot be an  $(n - \epsilon)$ -approximation algorithm for non-preemptive MAXCONNECTIVITY, unless  $P = NP$ . Notice that the construction in the proof of Theorem 3 has  $\Theta(n)$  maintenance jobs and we will introduce  $\Theta(n^2)$  copies of the construction, yielding  $|E| \leq c \cdot n^3$  for some  $c > 0$ . Hence, we have  $n \geq c' \sqrt[3]{|E|}$  for some  $c' > 0$ .

For the construction, we use  $n^2 - n$  copies of the 3SAT-network from the proof of Theorem 3, where each copy uses *different*  $(t_1, t_2)$ -combinations with  $t_1, t_2 \in \{0, \dots, n-1\}$  and  $t_1 \neq t_2$ . Considering special  $(s^+, s^-)$ -paths, a path labeled with  $k$  allows connectivity only during  $[k, k+1]$ ,  $k = 0, \dots, n-1$ , and passes through every 3SAT-network with  $t_1 = k$  or  $t_2 = k$ . Notice that within a 3SAT-network we have connectivity during both time slots if and only if the corresponding 3SAT-instance is a YES-instance. Also, we know due to [3] that there is an optimal solution which starts all jobs at integral times. Now, if the 3SAT-instance is a YES-instance, there is a global schedule such that its restriction to every 3SAT-network allows connectivity during both intervals. Thus each path with label  $k \in \{0, \dots, n-1\}$  allows connectivity during  $[k, k+1]$ . This implies that the maximum connectivity time is  $n$ .

Conversely, suppose there exists a global schedule with connectivity during two time slots. Then there must exist two paths  $P_1, P_2$  from  $s^+$  to  $s^-$  with two distinct labels, each realizing connectivity during one of both intervals. By construction there is one 3SAT-network they both use. This implies by the proof of Theorem 3, that the global schedule restricted to this 3SAT-network corresponds to a satisfying truth assignment for the 3SAT-instance.  $\square$

The results above hold for general graph classes, but even for graphs as simple as disjoint paths between  $s$  and  $t$ , the problem remains strongly NP-hard.

**Theorem 5.** *Non-preemptive MAXCONNECTIVITY is strongly NP-hard, and non-preemptive MINCONNECTIVITY is inapproximable even if the given graph consists only of disjoint paths between  $s$  and  $t$ .*

We give an algorithm that computes an  $(\ell + 1)$ -approximation for non-preemptive MAXCONNECTIVITY, where  $\ell \leq |E|$  is the number of different time points  $d_e - p_e, e \in E$ . The basic idea is that we consider a set of  $\ell + 1$  feasible maintenance schedules, whose total time of connectivity upper bounds the maximum total connectivity time of a single schedule. Then the schedule with maximum connectivity time among our set of  $\ell + 1$  schedules is an  $(\ell + 1)$ -approximation.

The schedules we consider start every job either immediately at its release date, or at the latest possible time. In the latter case it finishes exactly at the deadline. More precisely, for a fixed time point  $t$ , we start the maintenance of all

edges  $e \in E$  with  $d_e - p_e \geq t$  at their latest possible start time  $d_e - p_e$ . All other edges start maintenance at their release date  $r_e$ . This yields at most  $\ell + 1 \leq |E| + 1$  different schedules  $S_t$ , as for increasing  $t$ , each time point where  $d_e - p_e$  is passed for some edge  $e$  defines a new schedule. Algorithm 1 formally describes this procedure, where  $E(t) := \{e \in E : e \text{ is not maintained at } t\}$ .

---

**Algorithm 1.** Approx. Algorithm for Non-preemptive MAXCONNECTIVITY

---

- 1: Let  $t_1 < \dots < t_\ell$  be all different time points  $d_e - p_e, e \in E$ ,  $t_0 = 0$  and  $t_{\ell+1} = T$ .
  - 2: Let  $S_i$  be the schedule, where all edges  $e$  with  $d_e - p_e < t_i$  start maintenance at  $r_e$  and all other edges at  $d_e - p_e$ ,  $i = 1, \dots, \ell + 1$ .
  - 3: For each  $S_i$ , initialize total connectivity time  $c(t_i) \leftarrow 0$ ,  $i = 1, \dots, \ell + 1$ .
  - 4: **for**  $i = 1$  to  $\ell + 1$  **do**
  - 5:   Partition the interval  $[t_{i-1}, t_i]$  into subintervals such that each time point  $r_e, r_e + p_e, d_e, e \in E$ , in this interval defines a subinterval bound.
  - 6:   **for all** subintervals  $[a, b] \subseteq [t_{i-1}, t_i]$  **do**
  - 7:     **if**  $(V, E(1/2 \cdot (a + b)))$  contains an  $(s^+, s^-)$ -path for  $S_i$  **then**
  - 8:       Increase  $c(t_i)$  by  $b - a$ .
  - 9: **return** Schedule  $S_i$  for which  $c(t_i), i = 1, \dots, \ell + 1$ , is maximized.
- 

Algorithm 1 considers finitely many intervals, as all (sub-)interval bounds are defined by a time point  $r_e, r_e + p_e, d_e - p_e$  or  $d_e$  of some  $e \in E$ . As we can check the network for  $(s^+, s^-)$ -connectivity in polynomial time, and the algorithm does this for each (sub-)interval, Algorithm 1 runs in polynomial time.

**Theorem 6.** *Algorithm 1 is an  $(\ell + 1)$ -approximation algorithm for non-preemptive MAXCONNECTIVITY on general graphs, with  $\ell \leq |E|$  being the number of different time points  $d_e - p_e, e \in E$ .*

## 4 Power of Preemption

We first focus on MINCONNECTIVITY on a path and analyze how much we can gain by allowing preemption. First, we show that there is an algorithm that computes a non-preemptive schedule whose value is bounded by  $O(\log |E|)$  times the value of an optimal preemptive schedule. Second, we argue that one cannot gain more than a factor of  $\Omega(\log |E|)$  by allowing preemption.

**Theorem 7.** *The power of preemption is  $\Theta(\log |E|)$  for MINCONNECTIVITY on a path.*

*Proof.* Observe that if at least one edge of a path is maintained at time  $t$ , then the whole path is disconnected at  $t$ . We give an algorithm for MINCONNECTIVITY on a path that constructs a non-preemptive schedule with cost at most  $O(\log |E|)$  times the cost of an optimal preemptive schedule.

We first compute an optimal preemptive schedule. This can be done in polynomial time by Theorem 1. Let  $x_t$  be a variable that is 1 if there exists a job  $j$



that is processed at time  $t$  and 0 otherwise. We shall refer to  $x$  also as the *maintenance profile*. Furthermore, let  $a := \int_0^T x_t dt$  be the active time, i.e., the total time of maintenance. Then we apply the following *splitting procedure*. We compute the time point  $\bar{t}$  where half of the maintenance is done, i.e.,  $\int_0^{\bar{t}} x_t dt = a/2$ . Let  $E(t) := \{e \in E \mid r_e \leq t \wedge d_e \geq t\}$  and  $p_{\max} := \max_{e \in E(t)} p_e$ . We reserve the interval  $[\bar{t} - p_{\max}, \bar{t} + p_{\max}]$  for the maintenance of the jobs in  $E(\bar{t})$ , although we might not need the whole interval. We schedule each job in  $E(\bar{t})$  around  $\bar{t}$  so that the processing time before and after  $\bar{t}$  is the same. If the release date (deadline) of a jobs does not allow this, then we start (complete) the job at its release date (deadline). Then we mark the jobs in  $E(\bar{t})$  as scheduled and delete them from the preemptive schedule.

This splitting procedure splits the whole problem into two separate instances  $E_1 := \{e \in E \mid d_e < \bar{t}\}$  and  $E_2 := \{e \in E \mid r_e > \bar{t}\}$ . Note that in each of these sub-instances the total active time in the preemptive schedule is at most  $a/2$ . We apply the splitting procedure to both sub-instances and follow the recursive structure of the splitting procedure until all jobs are scheduled.  $\square$

**Lemma 3.** *For MINCONNECTIVITY on a path, the given algorithm constructs a non-preemptive schedule with cost  $O(\log |E|)$  times the cost of an optimal preemptive schedule.*

*Proof.* The progression of the algorithm can be described by a binary tree in which a node corresponds to a partial schedule generated by the splitting procedure for a subset of the job and edge set  $E$ . The root node corresponds to the partial schedule for  $E(\bar{t})$  and the (possibly) two children of the root correspond to the partial schedules generated by the splitting procedure for the two sub-problems with initial job sets  $E_1$  and  $E_2$ . We can cut a branch if the initial set of jobs is empty in the corresponding subproblem. We associate with every node  $v$  of this tree  $B$  two values  $(s_v, a_v)$  where  $s_v$  is the number of scheduled jobs in the subproblem corresponding to  $v$  and  $a_v$  is the amount of maintenance time spent for the scheduled jobs.

The binary tree  $B$  has the following properties. First,  $s_v \geq 1$  holds for all  $v \in B$ , because the preemptive schedule processes some job at the midpoint  $\bar{t}_v$  which means that there must be a job  $e \in E$  with  $r_e \leq \bar{t}_v \wedge d_e \geq \bar{t}_v$ . This observation implies that the tree  $B$  can have at most  $|E|$  nodes and since we want to bound the worst total cost we can assume w.l.o.g. that  $B$  has exactly  $|E|$  nodes. Second,  $\sum_{v \in B} a_v = \int_0^T y_t dt$  where  $y_t$  is the maintenance profile of the non-preemptive solution.

The cost  $a_v$  of the root node (level-0 node) is bounded by  $2p_{\max} \leq 2a$ . The cost of each level-1 node is bounded by  $2 \cdot a/2 = a$ , so the total cost on level 1 is also at most  $2a$ . It is easy to verify that this is invariant, i.e., the total cost at level  $i$  is at most  $2a$  for all  $i \geq 0$ , since the worst node cost  $a_v$  halves from level  $i$  to level  $i + 1$ , but the number of nodes doubles in the worst case. We obtain the worst total cost when  $B$  is a complete balanced binary tree. This tree has at most  $O(\log |E|)$  levels and therefore the worst total cost is  $a \cdot O(\log |E|)$ . The total cost of the preemptive schedule is  $a$ .  $\square$

We now provide a matching lower bound for the power of preemption on a path.

**Lemma 4.** *The power of non-preemption is  $\Omega(\log |E|)$  for MINCONNECTIVITY on a path.*

*Proof.* We construct a path with  $|E|$  edges and divide the  $|E|$  jobs into  $\ell$  levels such that level  $i$  contains exactly  $i$  jobs for  $1 \leq i \leq \ell$ . Hence, we have  $|E| = \ell(\ell + 1)/2$  jobs. Let  $P$  be a sufficiently large integer such that all of the following numbers are integers. Let the  $j$ th job of level  $i$  have release date  $(j - 1)P/i$ , deadline  $(j/i)P$ , and processing time  $P/i$ , where  $1 \leq j \leq i$ . Note that now no job has flexibility within its time window, and thus the value of the resulting schedule is  $P$ .

We now modify the instance as follows. At every time point  $t$  where at least one job has a release date and another job has a deadline, we stretch the time horizon by inserting a gap of size  $P$ . This stretching at time  $t$  can be done by adding a value of  $P$  to all time points after the time point  $t$ , and also adding a value of  $P$  to all release dates at time  $t$ . The deadlines up to time  $t$  remain the same. Observe that the value of the optimal preemptive schedule is still  $P$ , because when introducing the gaps we can move the initial schedule accordingly such that we do not maintain any job within the gaps of size  $P$ .

Let us consider the optimal non-preemptive schedule. The cost of scheduling the only job at level 1 is  $P$ . In parallel to this job we can schedule at most one job from each other level, without having additional cost. This is guaranteed by the introduced gaps. At level 2 we can fix the remaining job with additional cost  $P/2$ . As before, in parallel to this fixed job, we can schedule at most one job from each level  $i$  where  $3 \leq i \leq \ell$ . Applying the same argument to the next levels, we notice that for each level  $i$  we introduce an additional cost of value  $P/i$ . Thus the total cost is at least  $\sum_{i=1}^{\ell} P/i \in \Omega(P \log \ell)$  with  $\ell \in \Theta(\sqrt{|E|})$ .  $\square$

**Theorem 8.** *For non-preemptive MAXCONNECTIVITY on a path the power of preemption is unbounded.*

## 5 Mixed Scheduling

We know that both the non-preemptive and preemptive MAXCONNECTIVITY and MINCONNECTIVITY on a path are solvable in polynomial time by Theorem 1 and [14, Theorem 9], respectively. Notice that the parameter  $g$  in [14] is in our setting  $\infty$ . Interestingly, the complexity changes when mixing the two job types – even on a simple path.

**Theorem 9.** *MAXCONNECTIVITY and MINCONNECTIVITY with preemptive and non-preemptive maintenance jobs is weakly NP-hard, even on a path.*

**Theorem 10.** *There is a 2-approximation algorithm for MINCONNECTIVITY on a path with preemptive and non-preemptive maintenance jobs.*

## 6 Conclusion

Combining network flows with scheduling aspects is a very recent field of research. While there are solutions using IP based methods and heuristics, exact and approximation algorithms have not been considered extensively. We provide strong hardness results for connectivity problems, which is inherent to all forms of maintenance scheduling, and give algorithms for tractable cases.

In particular, the absence of  $c\sqrt[3]{|E|}$ -approximation algorithms for some  $c > 0$  for general graphs indicates that heuristics and IP-based methods [2–4] are a good way of approaching this problem. An interesting open question is whether the inapproximability results carry over to series-parallel graphs, as the network motivating [2–4] is series-parallel. Our results on the power of preemption as well as the efficient algorithm for preemptive instances show that allowing preemption is very desirable. Thus, it could be interesting to study models where preemption is allowed, but comes at a cost to make it more realistic.

On a path, our results have implications for minimizing busy time, as we want to minimize the number of times where some edge on the path is maintained. Here, an interesting open question is whether the 2-approximation for the mixed case can be improved, e.g. by finding a pseudo-polynomial algorithm, a better approximation ratio, or conversely, to show an inapproximability result for it.

**Acknowledgements.** We thank the anonymous reviewers for their helpful comments.

## References

1. Bley, A., Karch, D., D’Andreagiovanni, F.: WDM fiber replacement scheduling. *Electron. Notes Discret. Math.* **41**, 189–196 (2013). <http://www.sciencedirect.com/science/article/pii/S1571065313000954>
2. Boland, N., Kalinowski, T., Kaur, S.: Scheduling arc shut downs in a network to maximize flow over time with a bounded number of jobs per time period. *J. Comb. Optim.* **32**(3), 885–905 (2016)
3. Boland, N., Kalinowski, T., Kaur, S.: Scheduling network maintenance jobs with release dates and deadlines to maximize total flow over time: Bounds and solution strategies. *Comput. Oper. Res.* **64**, 113–129 (2015). <http://www.sciencedirect.com/science/article/pii/S0305054815001288>
4. Boland, N., Kalinowski, T., Waterer, H., Zheng, L.: Scheduling arc maintenance jobs in a network to maximize total flow over time. *Discret. Appl. Math.* **163**, 34–52 (2014). <http://dx.doi.org/10.1016/j.dam.2012.05.027>
5. Boland, N.L., Savelsbergh, M.W.P.: Optimizing the hunter valley coal chain. In: Gurnani, H., Mehrotra, A., Ray, S. (eds.) *Supply Chain Disruptions: Theory and Practice of Managing Risk*, pp. 275–302. Springer, London (2012). doi:10.1007/978-0-85729-778-5\_10
6. Canetti, R., Irani, S.: Bounding the power of preemption in randomized scheduling. *SIAM J. Comput.* **27**(4), 993–1015 (1998). <http://dx.doi.org/10.1137/S0097539795283292>

7. Chang, J., Khuller, S., Mukherjee, K.: LP rounding and combinatorial algorithms for minimizing active and busy time. In: Blelloch, G.E., Sanders, P. (eds.) Proceedings of the 26th SPAA, pp. 118–127. ACM, New York (2014). <http://doi.acm.org/10.1145/2612669.2612689>
8. Chang, J., Khuller, S., Mukherjee, K.: Active and busy time minimization. In: Proceedings of the 12th MAPSP, pp. 247–249 (2015). <http://feb.kuleuven.be/mapsp.2015/Proceedings%20MAPSP%202015.pdf>
9. Cohen-Addad, V., Li, Z., Mathieu, C., Milis, I.: Energy-efficient algorithms for non-preemptive speed-scaling. In: Bampis, E., Svensson, O. (eds.) WAOA 2014. LNCS, vol. 8952, pp. 107–118. Springer, Cham (2015). doi:10.1007/978-3-319-18263-6\_10
10. Correa, J.R., Skutella, M., Verschae, J.: The power of preemption on unrelated machines and applications to scheduling orders. *Math. Oper. Res.* **37**(2), 379–398 (2012). <http://dx.doi.org/10.1287/moor.1110.0520>
11. Flammini, M., Monaco, G., Moscardelli, L., Shachnai, H., Shalom, M., Tamir, T., Zaks, S.: Minimizing total busy time in parallel scheduling with application to optical networks. *Theor. Comput. Sci.* **411**(40–42), 3553–3562 (2010). <http://www.sciencedirect.com/science/article/pii/S0304397510002926>
12. Ha, S.: Compile-time scheduling of dataflow program graphs with dynamic constructs. Ph.D. thesis, University of California, Berkeley (1992). <http://www.eecs.berkeley.edu/Pubs/TechRpts/1992/ERL-92-43.pdf>
13. Kalinowski, T., Matsypura, D., Savelsbergh, M.W.: Incremental network design with maximum flows. *Eur. J. Oper. Res.* **242**(1), 51–62 (2015). <http://www.sciencedirect.com/science/article/pii/S0377221714008078>
14. Khandekar, R., Schieber, B., Shachnai, H., Tamir, T.: Real-time scheduling to minimize machine busy times. *J. Sched.* **18**(6), 561–573 (2015). <http://dx.doi.org/10.1007/s10951-014-0411-z>
15. Mertzios, G.B., Shalom, M., Voloshin, A., Wong, P.W.H., Zaks, S.: Optimizing busy time on parallel machines. In: Proceedings of the 26th IPDPS, pp. 238–248. IEEE (2012). <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6267839>
16. Nurre, S.G., Cavdaroglu, B., Mitchell, J.E., Sharkey, T.C., Wallace, W.A.: Restoring infrastructure systems: an integrated network design and scheduling (INDS) problem. *Eur. J. Oper. Res.* **223**(3), 794–806 (2012). <http://www.sciencedirect.com/science/article/pii/S0377221712005310>
17. Parsons, E.W., Sevcik, K.C.: Multiprocessor scheduling for high-variability service time distributions. In: Feitelson, D.G., Rudolph, L. (eds.) JSSPP 1995. LNCS, vol. 949, pp. 127–145. Springer, Heidelberg (1995). doi:10.1007/3-540-60153-8\_26
18. Schulz, A.S., Skutella, M.: Scheduling unrelated machines by randomized rounding. *SIAM J. Discret. Math.* **15**(4), 450–469 (2002). <http://dx.doi.org/10.1137/S0895480199357078>
19. Soper, A.J., Strusevich, V.A.: Power of preemption on uniform parallel machines. In: Proceedings of the 17th APPROX. LIPIcs, vol. 28, pp. 392–402. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2014). <http://drops.dagstuhl.de/opus/volltexte/2014/4711>