Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

FRANK PFEUFFER
AXEL WERNER

Adaptive telecommunication network operation with a limited number of reconfigurations

This research has been supported by the German Federal Ministry of Education and Research (BMBF) within the project SASER (www.celticplus.eu/project-saser)

ZIB Report 15-36 (July 2015)

Herausgegeben vom
Konrad-Zuse-Zentrum für Informationstechnik Berlin
Takustraße 7
D-14195 Berlin-Dahlem

Telefon: 030-84185-0
Telefax: 030-84185-125

e-mail: bibliothek@zib.de
URL: http://www.zib.de

ZIB-Report (Print) ISSN 1438-0064
ZIB-Report (Internet) ISSN 2192-7782

Adaptive telecommunication network operation with a limited number of reconfigurations

Frank Pfeuffer* Axel Werner*

July 10, 2015

Rising traffic in telecommunication networks lead to rising energy costs for the network operators. Meanwhile, increased flexibility of the networking hardware may help to realize load-adaptive operation of the networks to cut operation costs. To meet network operators' concerns over stability, we propose to switch network configurations only a limited number of times per day. We present a method for the integrated computation of optimal switching times and network configurations by alternating solution of a mixed-integer program and computation of a constrained shortest cycle in a certain graph. The algorithm can act as a framework to be adapted and applied to suitable problems of different origin.

1 Introduction

Due to a forecasted increase in data traffic, telecommunication network operators are faced with rising operation costs, of which energy consumption is one of the main contributing factors. Conventionally, networks are operated statically, even though the traffic volumes change considerably over the day. The emergence of flexible hardware operation modes (sleep mode, flexible bit-rate) enables immediate adaptation of capacities to actual network loads and saves resources during low-demand times. As long as such flexible networks are not yet an established technology, network operators likely prefer to choose from a limited number of thoroughly tested configurations for employment at carefully chosen time points instead of freely and incessantly reconfiguring the network.

*Zuse Institute Berlin (ZIB), Takustr. 7, D-14195 Berlin, Germany, {pfeuffer,werner}@zib.de

Some efforts to reduce energy-consumption through load-adaptiveness have been taken recently, see, e.g., [2] and the references therein. While it is often assumed that networks can be reconfigured freely, the need to consider constraints on the dynamism of the applied network configurations has also been recognized by some: In [1], a partitioning of the time horizon into intervals in which the network should not be reconfigured is part of the input for computing energy-minimal network configurations. A model based on random graph theory and a simple traffic profile for a day is used in [3] to choose a limited number of reconfiguration time points and to estimate the optimal power consumption between reconfigurations. Here, we address the problem of finding such reconfiguration time points and network configurations, optimal w.r.t. energy consumption, simultaneously by an integrated approach.

More precisely, the problem addressed is the following: Given a demand vector for each time point in a finite time period $[0, \tau^{\max}[$ and a minimum time τ^{wait} to wait between reconfigurations of the network, find at most N ($N \leq \frac{\tau^{\max}}{\tau^{\text{wait}}}$) disjoint time intervals $[\tau_i, \tau_{i+1}[$ with duration $\Delta(\tau_i, \tau_{i+1}) \geq \tau^{\text{wait}}$ partitioning $[0, \tau^{\max}[$, as well as associated network configurations with power consumption $P(\tau_i, \tau_{i+1})$ that are able to route all demands within the time interval $[\tau_i, \tau_{i+1}[$ such that total energy consumption is minimized:

$$E^{\text{opt}} := \min_{\substack{n \leq N \\ \tau_0 < \cdots < \tau_{n-1} \\ \tau_n = \tau_0}} \sum_{0 \leq i < n} P(\tau_i, \tau_{i+1})\, \Delta(\tau_i, \tau_{i+1}). \tag{1}$$

In the following we consider the discretized problem with time points τ_i chosen from the set $T := [0, \tau^{\max}[\cap g\,\mathbb{Z}$, with time granularity $g \in \mathbb{R}_+$. We also use an extended notation of intervals and interval lengths suitable for the periodic setting: For $\tau \geq \tau'$ define

$$[\tau, \tau'[:= [0, \tau'[\cup [\tau, \tau^{\max}[\quad \text{and} \quad \Delta(\tau, \tau') := \begin{cases} \tau' - \tau & \text{if } \tau < \tau', \\ \tau' - \tau + \tau^{\max} & \text{if } \tau \geq \tau'. \end{cases}$$

In Section 2 we explain how to compute the minimal power consumption $P(\tau, \tau')$ of the network for a time interval $[\tau, \tau'[$ and in Section 3 we use the power consumption values to obtain optimal time points for switching the network to minimize energy consumption. In Section 4 we describe an approach to compute time points and power consumption simultaneously and present results from computations using the method.

2 Optimization of network configurations

Although our approach will also work with other models, we briefly state the network configuration model we use: a network design problem for an IP-over-WDM core network with capacity modules for IP routers and optical

channels. Let V be the set of IP router sites, L a set of possible IP links, and $(d_k(\hat{\tau}))_{k\in K}$ the demand vector at time $\hat{\tau} \in [\tau, \tau'[\cap g\,\mathbb{Z}$. The IP layer is modeled as

$$\begin{aligned} \sum_{\ell\in L} (f_\ell^k - \bar{f}_\ell^k) &= \delta_i^k \qquad \forall i \in V, k \in K, \\ \sum_{k\in K} d_k(\hat{\tau})\,(f_\ell^k + \bar{f}_\ell^k) &\le t_\ell \qquad \forall \ell \in L, \hat{\tau} \in [\tau, \tau'[\cap g\,\mathbb{Z}, \end{aligned} \tag{2}$$

where $\delta_i^k = 1$ if i is the source of demand $k \in K$, -1 if it is the target, and 0 otherwise. The variables $f_\ell^k, \bar{f}_\ell^k \in \mathbb{Z}_+$ indicate the IP flow and $t_\ell \in \mathbb{R}_+$ the maximum traffic on IP link $\ell \in L$ within the time interval $[\tau, \tau'[$. Let E be a set of possible optical links, P a set of paths through E on which optical channels can be established, and N_p a set of possible modulations for optical channels on path p. C_n denotes the capacity of module n and W_n its spectral width. Then the optical layer is modeled by

$$\sum_{p\in P_\ell, n\in N_p} C_n\, y_{p,n} \ge t_\ell \quad \forall \ell \in L \quad \text{and} \quad \sum_{p\in P_e, n\in N_p} W_n\, y_{p,n} = z_e \quad \forall e \in E,$$

where we denote by P_ℓ the set of paths in P having the same end points as ℓ and by P_e the set of paths in P using edge e. The variables $y_{p,n} \in \mathbb{Z}_+$ count the number of modules of type n used on path p and $z_e \in \mathbb{Z}_+$ the consumed spectral width on optical link e. Let M_i be a set of IP router modules for site $i \in V$, Y_m the number of line cards provided by module m, A_i the number of line cards at i connecting the non-core network parts, and S the spectrum available on one fiber. The IP hardware and fibers are then modeled as

$$\sum_{m\in M_i} Y_m\, x_{i,m} \ge \sum_{p\in P_i, n\in N_p} y_{p,n} + A_i \quad \forall i \in V \quad \text{and} \quad S\,\zeta_e \ge z_e \quad \forall e \in E,$$

where we denote the set of paths in P with one end point in i by P_i. The variables $x_{i,m} \in \{0, 1\}$ indicate the IP router module used, and variables $\zeta_e \in \mathbb{Z}_+$ indicate the number of fibers used on optical link e. We minimize total power consumption of the used hardware:

$$P(\tau, \tau') = \min \sum_{i\in V, m\in M_i} E_m\, x_{i,m} + \sum_{p\in P, n\in N_p} E_n\, y_{p,n} + \sum_{e\in E} E_e\, \zeta_e, \tag{3}$$

where E_m, E_n, and E_e denote the power consumption of the corresponding modules m, n and link e, respectively. Computing $P(\tau, \tau')$ then amounts to solving a mixed-integer program (MIP).

The following helpful property is a consequence of Eq. (2):

$$[\tau, \tau'[\subseteq [\bar{\tau}, \bar{\tau}'[\quad \text{implies} \quad P(\tau, \tau') \le P(\bar{\tau}, \bar{\tau}'), \tag{4}$$

since the optimization problem for computing $P(\bar{\tau}, \bar{\tau}')$ relaxes the problem for computing $P(\tau, \tau')$. We also remark that

$$P(\tau, \tau') \geq 0 \quad \forall \tau, \tau' \in T, \tag{5}$$

since the coefficients in the objective are power consumption values, which are non-negative.

3 Optimization of time intervals

We assume in this section that the power consumption of an optimal network configuration for the time interval $[\tau, \tau'[$, $P(\tau, \tau')$, is known for all $\tau, \tau' \in T$. We construct a directed graph $G = (V, A)$ with node set $V = T$ and arc set $A = \{(\tau, \tau') \in T \times T : \Delta(\tau, \tau') \geq \tau^{\text{wait}}\}$, and we define arc weights

$$w_P : (\tau, \tau') \mapsto P(\tau, \tau')\, \Delta(\tau, \tau').$$

Next, we relate energy-optimal partitions of the time horizon T to cycles in this graph, i.e., to sequences $(\tau_0, \tau_1, \ldots, \tau_n)$ of at least two nodes where consecutive nodes are connected by an arc, $(\tau_i, \tau_{i+1}) \in A$ for $i = 0, \ldots, n-1$, and which is closed, $\tau_n = \tau_0$.

Theorem 1. *There is an N-hop constrained shortest cycle in (G, w_P) that corresponds to a set of time points that is energy optimal in the sense of* (1).

Proof. Since time intervals correspond to arcs in G the correspondence between sets of time points as limits of intervals covering T and cycles is obvious. Clearly, for an N-hop constrained shortest cycle $C = (\tau_0, \tau_1, \ldots, \tau_{n-1}, \tau_n)$ with $n \leq N$ to induce a partition (and not just a cover) of T, it has to satisfy $|\{i : \tau_i \geq \tau_{i+1}\}| = 1$. Assume that $|\{i : \tau_i \geq \tau_{i+1}\}| \geq 2$. Then

$$\sum_{i=0}^{n-1} \Delta(\tau_i, \tau_{i+1}) = |\{i : \tau_i \geq \tau_{i+1}\}|\, \tau^{\max} \geq 2\, \tau^{\max}$$

implies that $\Delta(\tau_j, \tau_{j+1}) \geq \frac{2}{n}\, \tau^{\max} \geq 2\, \tau^{\text{wait}}$ for some j; by periodic shift of the order of the arcs in C and of the demands we can w.l.o.g. assume that $j = 0$, $\tau_0 = 0$ and consequently $\tau_1 \geq 2\, \tau^{\text{wait}}$.

Let $\hat{\imath} < n - 1$ be the smallest index such that $\tau_{\hat{\imath}} \geq \tau_{\hat{\imath}+1}$. Define $\bar{\tau} := \min\{\tau_{\hat{\imath}+1}, \lfloor \frac{1}{2}\, \tau_1 \rfloor\}$ and the cycle $C' := (\bar{\tau}, \tau_1, \ldots, \tau_{\hat{\imath}}, \bar{\tau})$ with $\hat{\imath} + 1 < n$ hops. The cycle C' only uses arcs of C except for $(\bar{\tau}, \tau_1)$ and $(\tau_{\hat{\imath}}, \bar{\tau})$; the definition of $\bar{\tau}$ makes sure that $\Delta(\bar{\tau}, \tau_1) \geq \tau^{\text{wait}}$ and $\Delta(\tau_{\hat{\imath}}, \bar{\tau}) \geq \tau^{\text{wait}}$, and thus all arcs

of C' lie in G. For the weight of the new cycle holds

$$\begin{aligned} w_P(C') &= w_P(\bar{\tau}, \tau_1) + \sum_{i=1}^{\hat{\imath}-1} w_P(\tau_i, \tau_{i+1}) + w_P(\tau_{\hat{\imath}}, \bar{\tau}) \\ &\leq w_P(\tau_0, \tau_1) + \sum_{i=1}^{\hat{\imath}-1} w_P(\tau_i, \tau_{i+1}) + w_P(\tau_{\hat{\imath}}, \tau_{\hat{\imath}+1}) \\ &\leq \sum_{i=0}^{n-1} w_P(\tau_i, \tau_{i+1}) \\ &= w_P(C) \end{aligned}$$

since $[\bar{\tau}, \tau_1[\subseteq [\tau_0, \tau_1[$ and $[\tau_{\hat{\imath}}, \bar{\tau}[\subseteq [\tau_{\hat{\imath}}, \tau_{\hat{\imath}+1}[$ imply, with the help of Eq. (4), $w_P(\bar{\tau}, \tau_1) \leq w_P(\tau_0, \tau_1)$ and $w_P(\tau_{\hat{\imath}}, \bar{\tau}) \leq w_P(\tau_{\hat{\imath}}, \tau_{\hat{\imath}+1})$, and since by Eq. (5) $w_P(\tau_i, \tau_{i+1}) \geq 0$ for all i with $\hat{\imath} + 1 \leq i < n$.

Thus, we have verified that C' uses only arcs of G, that it has less hops as C and therefore also satisfies the N-hop constraint, and that it is at most as long as C. Finally, the construction makes sure that only one arc (τ, τ') on C' has $\tau' \geq \tau$ such that C' corresponds to a partition of T. □

4 Integrated optimization approach

To obtain all arc weights $w_P(\tau, \tau')$ of G, we would have to solve $|A| = O(|T|^2)$ $\mathcal{NP}$-hard network design problems. To avoid this, we need some mechanism to distinguish relevant edges from those that are not needed in an optimal solution or for proving its optimality. To this end, let L and U be lower and upper bounds of P: $L(\tau, \tau') \leq P(\tau, \tau') \leq U(\tau, \tau')$ for all $\tau, \tau' \in T$. Then w_L and w_U can be defined analogous to w_P in Section 3 as

$$w_L : (\tau, \tau') \mapsto L(\tau, \tau')\, \Delta(\tau, \tau') \quad \text{and} \quad w_U : (\tau, \tau') \mapsto U(\tau, \tau')\, \Delta(\tau, \tau').$$

For the sake of brevity we call an N-hop constrained shortest cycle with respect to w_P an *N-hop P-shortest cycle* (analogous for L and U). For an N-hop L-shortest cycle C_L and an N-hop U-shortest cycle C_U, together with an N-hop P-shortest cycle, which achieves the optimum E^{opt} in (1) by Theorem 1, we have

$$w_L(C_L) \leq w_L(C_P) \leq w_P(C_P) = E^{\text{opt}} \tag{6}$$

and

$$E^{\text{opt}} = w_P(C_P) \leq w_P(C_U) \leq w_U(C_U). \tag{7}$$

Proposition 2. *An N-hop L-shortest cycle gives a lower bound on E^{opt} and, analogously, an N-hop U-shortest cycle gives an upper bound on E^{opt}.*

This justifies Algorithm 1 for simultaneously finding time intervals and network configurations solving (1) up to a relative target gap ε. It maintains upper and lower bounds for w_P on the arcs of G together with the N-hop U-shortest and L-shortest cycles. The algorithm successively improves up to the target gap ε the upper and lower bounds on arcs that belong to the shortest cycles. If the weights of the shortest cycles with respect to the updated bounds have a gap below ε, the algorithm returns the upper bound cycle; the time intervals corresponding to the cycle are optimal up to ε.

Algorithm 1 Integrated optimization of intervals and configurations

Input: number of configurations N, relative target gap ε, arc set A, data necessary to compute $P(\tau, \tau')$ for all $(\tau, \tau') \in A$ (see Section 2)
Output: at most N time intervals and configurations

for all $(\tau, \tau') \in A$ **do**
  $U(\tau, \tau') :=$ generic upper bound on $P(\tau, \tau')$, e.g., from a heuristic
  $x(\tau, \tau') :=$ the network configuration realizing $U(\tau, \tau')$
  $L(\tau, \tau') :=$ generic lower bound on $P(\tau, \tau')$, e.g., from LP relaxation
end for
loop
  $C_U :=$ N-hop U-shortest cycle
  $C_L :=$ N-hop L-shortest cycle
  if the relative gap between lengths of C_U and C_L is below ε **then**
    return C_U, $x(\tau, \tau')$ for all $(\tau, \tau') \in C_U$
  end if
  Choose a pivot element $(\tau, \tau') \in C_U \cup C_L$
  Improve $U(\tau, \tau')$, $L(\tau, \tau')$ for (τ, τ') up to relative gap ε
  (e.g., by using an integer programming solver)
  and let $x(\tau, \tau')$ be the network configuration realizing $U(\tau, \tau')$
end loop

4.1 Implementation details and improvements

Algorithm 1 is formulated rather generically and several issues are subject to further, more precise specification. The critical subtasks are the initialization and improvement of the bounds. Every upper or lower bound on an arc (τ, τ'), implies an upper or lower bound, respectively, on other arcs as well due to Eq. (4):

Proposition 3. *A lower bound for $P(\tau, \tau')$ is also a lower bound for $P(\bar{\tau}, \bar{\tau}')$ for all $[\bar{\tau}, \bar{\tau}'[\supseteq [\tau, \tau'[$. An upper bound for $P(\tau, \tau')$ is also an upper bound for $P(\bar{\tau}, \bar{\tau}')$ for all $[\bar{\tau}, \bar{\tau}'[\subseteq [\tau, \tau'[$.*

In other words, in the partially ordered set build from the arc set A and ordered by the set inclusion relation of the associated time intervals,

lower bounds propagate from minimal towards maximal elements while upper bounds propagate from maximal towards minimal elements.

Initialization. Due to Proposition 3, for obtaining lower bounds on all arcs, it is sufficient to initialize lower bounds on arcs whose corresponding time interval does not contain the time interval of any other arc (minimal elements). Analogously, for upper bounds on all arcs, initialization is sufficient for those arcs whose time interval is not contained in the time interval of any another arc (maximal elements). Likewise, the improvement of the bounds on one arc may allow to update the bounds on other arcs, which helps to avoid to explicitly compute bounds on some arcs.

In our implementation we obtain initial upper and lower bounds on an arc by solving the root node of the associated mixed-integer program (3) with a start solution computed by a shortest path heuristic. In a first step, only the arcs whose associated time intervals are minimal w.r.t. set inclusion are initialized this way. By propagation, this implicitly initializes lower bounds on all arcs, but upper bounds only on these minimal arcs. In fact, it may yet be impossible for the algorithm to find an initial N-hop U-shortest cycle C_U with finite weight assuming arcs with uninitialized upper bounds have the trivial upper bound $+\infty$. To avoid this situation, we have to make sure a cycle with finite U-weight exists. This can be done by simply solving the root node (with heuristic start solution) on the arc corresponding to the whole time horizon, which naturally contains the time intervals of all other arcs and implies upper bounds there. The disadvantage is that all cycles C end up with the same weight $w_U(C) = U(\tau, \tau')\,\Delta(\tau, \tau')$, which results in a completely arbitrary initial choice of C_U.

We choose a different procedure, which is inspired by the observation that power consumption $P(\tau, \tau')$ roughly depends (affinely) linear on the demand sum of the component-wise maximum demand vector over the time period $[\tau, \tau'[$, see Fig. 1. Thus, we first compute the N-hop shortest cycle C_d with respect to the edge weights

$$w_d(\tau, \tau') = \sum_{k \in K} \max_{\hat{\tau} \in [\tau, \tau'[} d_k(\hat{\tau}),$$

which are trivial to obtain, and then solve the root node (with heuristic start solution) on each arc of the cycle C_d.

Choice of pivot elements. A further decision is the choice of the pivot element in Step 12 of Algorithm 1. We consider the following criteria to determine priorities for the time intervals in $C_U \cup C_L$:

1. whether $(\tau, \tau') \in C_L$;
2. whether the associated mixed-integer program's root node has not yet been solved on (τ, τ');

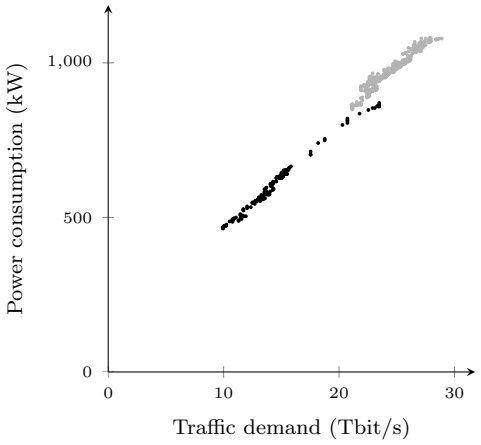

Figure 1: Power consumption $P(\tau, \tau')$ as a function of the demand sum of the maximum traffic demand vector $w_d(\tau, \tau')$ for various time intervals $[\tau, \tau'[$ in the two networks `abilene` (gray) and `geant_eu15` (black)

3. whether the relative gap

$$\gamma(\tau, \tau') := \frac{U(\tau, \tau') - L(\tau, \tau')}{U(\tau, \tau')}$$

between upper and lower bound on (τ, τ') is above ε;

4. and the magnitude of the relative gap $\gamma(\tau, \tau')$ itself.

Let $\lambda(\tau, \tau')$, $\rho(\tau, \tau')$, $\alpha(\tau, \tau') \in \{0, 1\}$ indicate with a 1 whether Criterion 1, 2, and 3, respectively, hold true. This defines a priority vector

$$\pi(\tau, \tau') := (\lambda(\tau, \tau'), \rho(\tau, \tau'), \alpha(\tau, \tau'), \gamma(\tau, \tau'));$$

we choose the arc with lexicographically largest π-vector as our pivot element. This prioritizes improving the lower bound cycle C_L and the global lower bound $w_L(C_L)$, while first improving the gap on arcs whose initial bounds came from propagation from another arc and then on the remaining arcs with high gap. To improve the bounds on arcs with yet unsolved root node we solve the root node, otherwise we use a mixed-integer programming solver to improve the bounds up to the target gap ε.

Early termination of solvers. While the algorithm is improving bounds, it becomes apparent that more and more arcs do not have to be considered further, because they either have bounds of satisfactory quality or their bounds indicate that they have become obsolete. More precisely, an arc (τ, τ') becomes obsolete if the N-hop L-shortest cycle C through this arc has a weight above the cutoff value $(1 - \varepsilon)\, w_U(C_U)$. An arc can already become obsolete

while a mixed-integer programming solver is improving the bounds on it, since changing the bounds affects the weights of C and C_U. In this case the solver can be terminated early to save some solver time, in particular during the final iterations of the algorithm.

Computation of cycles. In each iteration, Algorithm 1 needs to compute N-hop constrained shortest cycles with respect to w_L and w_U. Note that because of Eq. (5) we can assume that the arc weights w_L and w_U are non-negative. Therefore, computing the cycles can be done in polynomial time by the following procedure: First, compute $(N-1)$-hop shortest paths between all pairs of nodes with a suitable algorithm, as in [6, Ch. 3, Sec. 12], for instance. Then, for each node pair (τ, τ') add the additional arc (τ', τ) to the $(N-1)$-hop shortest path from τ to τ' and thus form cycles. Finally, choose the shortest of these cycles. This procedure completes within $O(|T|^3 \log N)$ steps.

4.2 Computational Results

For computational studies on practical instances, we adopted two exemplary networks from SNDlib [7], `abilene` (12 nodes) and `geant`, which we transformed into a 15-node version `geant_eu15`. For both networks, measured demand curves are available, which were averaged to yield day curves with 1 h or ½ h granularity and scaled to match the capacities of today's hardware. Technical data on network devices, such as link length restrictions and power consumption, were taken from [4, 5].

Solutions for these instances with $N = 1, \ldots, 6$ and a waiting time of 4 hours as well as the maximum number of configurations allowed by the demand granularity ($N = 24$ or $N = 48$) were computed, optimal up to a target gap of $\varepsilon = 1\%$ for `abilene` and $\varepsilon = 2\%$ for `geant_eu15`. Table 1 lists for each instance the number of arcs in the corresponding graph G, the number of arcs for which initial lower and upper bounds were computed by solving only the branch-and-bound root node of the network design problem, the number of arcs for which the network design problem was solved aiming at the target gap, the number of arcs on which the solution process was terminated before reaching the target gap, the total time spent by the MIP solver[1], and the energy consumption of the computed solution.

The results show that by employing Algorithm 1, the computationally expensive network design problems have to be solved up to the target gap for only a small fraction of all arcs of G, maximally for about 2 % in the cases where $N \leq 6$ and still less than 4 % when unlimited reconfiguration is allowed. For a number of these arcs, the solver could be terminated before the target

[1]The MIP solver used was CPLEX 12.6; all computations were carried out on Ubuntu 14.04 Linux systems using Intel Xeon E3-1245 3.4 GHz quad-core CPUs, 32 GB of memory.

Network	# configurations	waiting time	# initial bounds	# target gap	# early terminated	total solver time	energy consumption
`geant_eu15`	1	–	1	1	0	98 s	21.1 MW
granularity: 1 h	2	4 h	28	1	0	1705 s	17.1 MW
# arcs: 409	3	4 h	36	2	0	1740 s	16.4 MW
target gap: 2 %	4	4 h	35	4	0	5196 s	15.7 MW
	5	4 h	33	5	0	5644 s	15.3 MW
	6	4 h	24	4	0	11420 s	15.2 MW
	24	–	24	14	1	9497 s	14.1 MW
`geant_eu15`	1	–	1	1	0	5385 s	20.7 MW
granularity: ½ h	2	4 h	62	2	1	10664 s	16.5 MW
# arcs: 1585	3	4 h	91	12	7	39643 s	15.9 MW
target gap: 2 %	4	4 h	59	4	0	26744 s	15.1 MW
	5	4 h	59	5	0	16624 s	14.8 MW
	6	4 h	49	5	1	10053 s	14.7 MW
	48	–	48	35	1	25566 s	13.5 MW
`abilene`	1	–	1	1	0	3137 s	28.4 MW
granularity: 1 h	2	4 h	99	7	4	3593 s	26.6 MW
# arcs: 409	3	4 h	64	8	3	1903 s	25.6 MW
target gap: 1 %	4	4 h	47	5	2	958 s	24.9 MW
	5	4 h	32	6	1	1167 s	24.5 MW
	6	4 h	31	8	1	1612 s	24.5 MW
	24	–	24	13	1	999 s	22.7 MW
`abilene`	1	–	1	1	0	2479 s	25.9 MW
granularity: ½ h	2	4 h	264	29	21	20631 s	24.5 MW
# arcs: 1585	3	4 h	131	21	11	13257 s	23.5 MW
target gap: 1 %	4	4 h	130	33	17	8652 s	23.1 MW
	5	4 h	90	22	7	7195 s	22.8 MW
	6	4 h	81	23	2	8492 s	22.8 MW
	48	–	48	38	1	2881 s	20.7 MW

Table 1: Computational results for four instances and different maximum number of configurations N, and minimum time to wait between reconfigurations τ^{wait}: the number of arcs on which initial upper and lower bounds were computed, the number of arcs on which the network design problem was solved aiming at the target gap, the number of arcs on which the latter was terminated before the target gap was reached, the total time spent by the MIP solver, and the optimized energy consumption

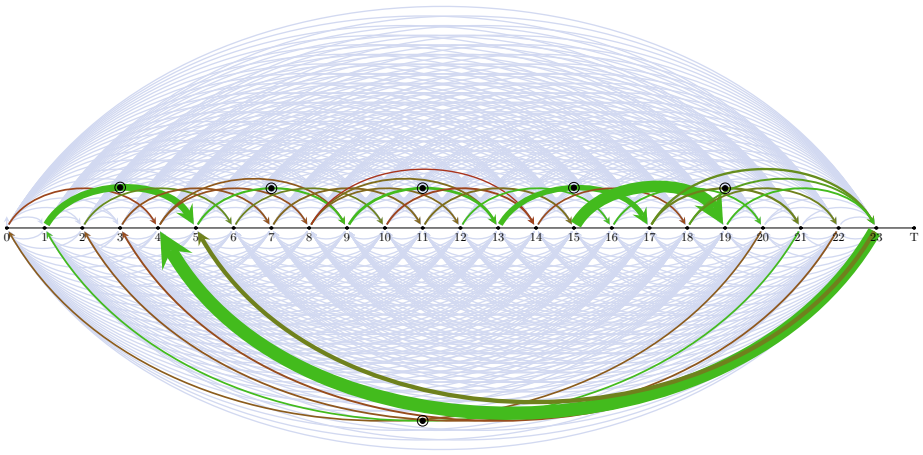

Figure 2: Computational effort to solve the various subproblems in the graph G for network `abilene` (granularity 1 h, 6 configurations, waiting time 4 h, target gap 1 %); broader arcs indicate longer solution times for the respective subproblems, red arcs indicate subproblems that were solved up to large gap, green arcs such with gaps near or below the target gap ε, shaded arcs represent subproblems that need not be considered at all; the final upper bound cycle is marked by dots, the final lower bound cycle by circles

gap was reached due to the arc becoming obsolete while the solver was improving the bounds; this is particularly true for the `abilene` instances, where about one third of the solver runs were terminated early. The less expensive computation of lower and upper bounds by heuristics and LP relaxations was necessary for maximally 9 % of all arcs for the `geant_eu15` network and up to 25 % of all arcs for `abilene`. As for the total computation time, no clear trend can be derived. It is obvious, however, that refining the granularity results in substantially higher computation times. For the slightly larger `geant_eu15` network, the algorithm consumes more time on average than for `abilene`, even though aiming at a higher target gap.

Figure 2 visualizes the effort spent on solving the individual problems associated with different arcs of the graph G for a typical run. During the total running time of 1612 s, less than 10 % of all arcs in the graph had to be touched and the time for solving the respective subproblem exceeded one minute only for five arcs. For a number of arcs, a solution with large gap (drawn in red), mostly from solving only the root node, was sufficient to obtain an overall near-optimal solution.

Figure 3 shows the power consumption of the optimal solutions (w.r.t.

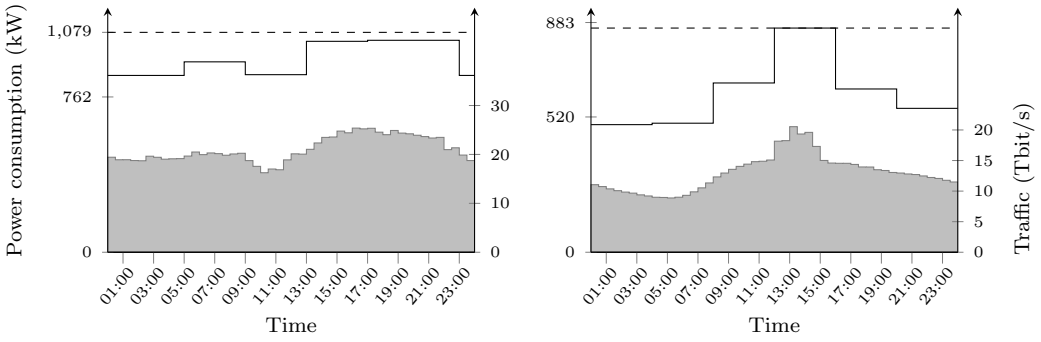

Figure 3: Traffic demands (shaded) and total power consumption over time for static network (dashed) and at most 6 reconfigurations with a waiting time of 4 h (solid); `abilene` (left) and `geant_eu15` (right)

target gap) for $N = 6$ for `abilene` and `geant_eu15` with ½ h granularity, as well as the traffic curves in these networks. Compared to the total energy consumption of a static network over the day, dynamic reconfiguration allows for considerable energy savings, given by the difference of the integrals of the dashed and solid lines. For `abilene` this amounts to savings of about 12 % and for `geant_eu15` of about 29 %. Table 1 lists the energy consumption of all considered instances. With increasing number of configurations N the energy savings increase, where the biggest increase in savings happens already for small N. Comparing the fully dynamic networks ($N = 24$ or $N = 48$) to those with the highest possible number of configurations with a 4 hour waiting time ($N = 6$), it turns out that the benefit of having to obey no waiting times amounts to about 5–8 percentage points of additional savings.

5 Outlook

In Algorithm 1, the specific nature of the optimization problems providing the values $P(\tau, \tau')$ is more or less irrelevant. Hence the algorithm can be applied not only in network design, but also for other subproblems that satisfy certain conditions. More precisely, Theorem 1 and Algorithm 1 are applicable for all problems, for which the total objective is expressible as the sum of the objectives of the subproblems and these objectives satisfy the conditions in Eqs. (4) and (5).

To generalize this even further, assume the overall objective can be expressed as

$$\min_{\substack{n \le N \\ \tau_0 < \cdots < \tau_{n-1} \\ \tau_n = \tau_0}} \quad \Phi(\tau_0, \ldots, \tau_n)$$

for a function $\Phi : T^{\leq N+1} \to \mathbb{R}$, where $T^{\leq N+1}$ is the set of all n-tuples, with $n \leq N+1$, of time points in T. If we define the length of a cycle by its value under Φ then Theorem 1 holds, provided that the following monotonicity condition is satisfied for all $(\tau_0, \ldots, \tau_n), (\tau'_0, \ldots, \tau'_{n'}) \in T^{\leq N+1}$ with $n \leq n'$:

$$\tau_0 \geq \tau'_0,\ \tau_1 = \tau'_1,\ \ldots,\ \tau_{n-1} = \tau'_{n-1},\ \tau_n \leq \tau'_n \\ \implies \quad \Phi(\tau_1, \ldots, \tau_n) \leq \Phi(\tau'_1, \ldots, \tau'_{n'}). \quad (8)$$

For the objective considered in this paper,

$$\Phi(\tau_0, \ldots, \tau_n) := \sum_{i=0}^{n-1} P(\tau_i, \tau_{i+1})\, \Delta(\tau_i, \tau_{i+1}),$$

Condition (8) is satisfied due to Eqs. (4) and (5).

Another setting in which Algorithm 1 can be applied, originates from the context of network fault tolerance. Assume that we want to design a network configuration schedule with a limited number of reconfigurations and minimum waiting times between these that minimizes the probability of a network failure. Suppose that with each network configuration there is associated a *normalized failure probability*, which represents the probability that a failure occurs in said configuration within a unit time interval of duration 1. For a given time period $[\tau, \tau'[$, let $p(\tau, \tau')$ be the minimal normalized failure probability of all network configurations that are able to route all demands in $[\tau, \tau'[$. Then the total failure probability of a schedule over the time horizon is

$$\Phi(\tau_0, \ldots, \tau_n) := 1 - \prod_{i=0}^{n-1} (1 - p(\tau_i, \tau_{i+1}))^{\Delta(\tau_i, \tau_{i+1})}$$

where we assume that failure probabilities for different configurations and intervals are independent. Since a configuration for a time interval $[\bar{\tau}, \bar{\tau}'[$ is also able to route the demands of any contained time interval $[\tau, \tau'[\subseteq [\bar{\tau}, \bar{\tau}'[$, we have $p(\tau, \tau') \leq p(\bar{\tau}, \bar{\tau}')$. Hence, Φ again satisfies the monotonicity condition (8).

Other objectives that would satisfy the necessary condition might be of interest, such as bottleneck problems minimizing the value of $\Phi(\tau_0, \ldots, \tau_n) = \max_{i=0}^{n-1} v(\tau_i, \tau_{i+1})$ for some suitable function v. Objectives like these may arise in multilayer network design when the number of hops of logical links in the physical network or the number of used lightpaths on optical links should be minimized.

Algorithm 1 provides a framework for computing an optimal collection of solutions to the subproblem considered, and as such it can be extended with all kinds of heuristics and efficient methods to compute lower bounds and (close-to) optimal solutions for the problem at hand. Additionally, the proposed procedure for choosing the pivot arc (τ, τ') in Step 12 is only one possibility; various other strategies for this step are devisable. Such improvements might speed up the general algorithm considerably.

References

[1] B. Addis, A. Capone, G. Carello, L. Gianoli, and B. Sanso. Energy management through optimized routing and device powering for greener communication networks. *Transactions on Networking*, 22, 2014.

[2] A. Betker, I. Gamrath, D. Kosiankowski, C. Lange, H. Lehmann, F. Pfeuffer, F. Simon, and A. Werner. Comprehensive topology and traffic model of a nationwide telecommunication network. *Journal of Optical Communications and Networking*, 6, 2014.

[3] L. Chiaraviglio, A. Cianfrani, E. Le Rouzic, and M. Polverini. Sleep modes effectiveness in backbone networks with limited configurations. *Computer Networks*, 57, 2013.

[4] Gwang-Hyun Gho, L. Klak, and J. M. Kahn. Rate-adaptive coding for optical fiber transmission systems. *Journal of Lightwave Technology*, 29(2):222–233, Jan 2011.

[5] W. Van Heddeghem and F. Idzikowski. Equipment power consumption in optical multilayer networks – source data. Technical report, Ghent Univ., 2012. `http://powerlib.intec.ugent.be`.

[6] Eugene Lawler. *Combinatorial Optimization: Networks and Matroids*. Saunders College Publishing, Fort Worth, 1976.

[7] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski. SNDlib 1.0—Survivable Network Design library. *Networks*, 55(3), 2010. `http://sndlib.zib.de`.