An Integer Network Flow Problem with Bridge Capacities

Anika Kinscherff[*]and Horst W. Hamacher[*]

Fachbereich Mathematik, Technische Universität Kaiserslautern, Germany

Email: {hamacher,kinscherff}@mathematik.uni-kl.de

June 12, 2017

Abstract

In this paper a modified version of dynamic network flows is discussed. Whereas dynamic network flows are widely analyzed already, we consider a dynamic flow problem with aggregate arc capacities called *Bridge Problem* which was introduced by Melkonian [Mel07]. We extend his research to integer flows and show that this problem is strongly NP-hard. For practical relevance we also introduce and analyze the hybrid bridge problem, i.e. with underlying networks whose arc capacity can limit aggregate flow (bridge problem) or the flow entering an arc at each time (general dynamic flow). For this kind of problem we present efficient procedures for special cases that run in polynomial time. Moreover, we present a heuristic for general hybrid graphs with restriction on the number of bridge arcs. Computational experiments show that the heuristic works well, both on random graphs and on graphs modeling also on realistic scenarios.

Keywords: Bridge Capacity, Maximum Dynamic Flow, Temporally Repeated Flows, Combinatorial Optimization, Evacuation Planning

1 Introduction

The concept of dynamic networks flows have become important for many applications and they are essential for modeling movement over time especially on street networks. This way they are indispensable as modeling tool [HT01] for evacuation planning. Depending on the given scenario and purpose, one can choose from many different variants of dynamic network flows. The most common dynamic flows are the quickest flow (QF), the maximal dynamic flow (MDF) and the earliest arrival flow (EAF). In the context of evacuation planning the quickest flow problem seeks the clearing of the affected area by the given population in minimal time (see e.g. [BDK93] and [FS07]). The maximal dynamic flow problem is probably the most discussed version of dynamic

[*]Partially supported by the German Ministry of Research and Technology (BMBF), project RobEZiS, grant number 13N13198.

network flows. If we have only limited time T for an evacuation, the MDF guarantees that the maximal amount of evacuees can get out early enough. Ford and Fulkerson ([FJF58], [FF10]) introduced the MDF and showed that this problem can be solved in polynomial time by temporally repeated flows (TRF). Basically for a TRF computation we determine a minimum cost flow f circulation on G modified by adding on arc from a given sink to the source with infinite capacity and duration of $-(T+1)$. On f we apply the flow decomposition theorem [AMO93] which leads to paths $P_1, ..., P_K$ with capacities depending on f, in our original network G. Then, we obtain a TRF by sending the maximal possible flow over P_i for $i = 1, ..., K$ for every time step $t \in \{0, ..., T - \tau(P_i)\}$ with $\tau(P)$ being the time need to traverse P. The earliest arrival flow is a special case of the MDF. In addition to maximize the number of evacuees in the given time T, the EAF maximizes the number of evacuees that reaches the destination at any time $t \leq T$ (see [Gal59], [Min73]). For some overviews on dynamic network flow with direct application on evacuation planning see e.g. [HT01] and [HT94].

In the following we only consider discrete models of dynamic flow problems and turn our attention to MDFs in a given network $G = (V, A)$ with node set V and arc set A. There are two mappings for each arc, $u : A \to \mathbb{N}$, with $u(a)$ or $u(v, w)$ defining the transit capacity of flow that can enter arc $a = (v, w)$ at any time step, and $\tau : A \to \mathbb{N}$ defining the time which is needed to traverse the arc. Since we are focusing on the discrete time case, τ is assumed to be integer and positive. Moreover, in evacuation problems the flow in general represents evacuees or cars, which motivates the integer values of u. Flow $f(a, t)$ with $f : A \times \mathbb{N} \longrightarrow \{\mathbb{N}, \mathbb{R}\}$ describes the flow value entering arc a at time t.

Given a network G as stated, a source s, a sink d and a time horizon T, a general dynamic flow problem can be solved by using the underlying time expanded network $G_T = (V_T, A_T)$ (see [AMO93]). Therefore, we define

$$\begin{aligned} V_T &:= \{s', d'\} \cup \{v_i | \text{ for } v \in V,\, t \in \{0, ..., T\}\} \\ A_T^1 &:= \{(s', s_t) | \text{ for } t \in \{0, ..., T\} \cup \{(d_t, d') | \text{ for } t \in \{0, ..., T\}\} \\ A_T^2 &:= \{(v_t, u_{t+\tau(v,u)}) | \text{ for } (u, v) \in A,\, t \in \{0, ..., T - \tau(v, u)\}\} \\ A_T &:= A_T^1 \cup A_T^2 \end{aligned}$$

as node and arc set of G_T for dynamic flows without waiting, i.e. each flow unit entering a node has to be sent directly to one of the outgoing arcs. The capacity of an arc $(v_t, u_{t+\tau(v,u)})$ equals the capacity of the corresponding arc (v, u) in G. If waiting is allowed we extend A_T by holdover arcs, i.e. arcs (v_t, v_{i+t}) with infinite capacity for $v \in V$ and $t \in \{0, ..., T-1\}$. Then, we can apply common static flow algorithms on G_T and the resulting flow corresponds to a dynamic flow in G. Since T is unbounded in general, algorithms on G_T do not have a polynomial running time.
Given a time horizon T, a source s and a sink d the *excess* of a flow f in node $v \in V$ at time $t \in \{0, ..., T\}$ is defined as

$$excess_f(v, t) = \sum_{a \in \delta^-(v)} \sum_{t'=0}^{t} f(a, t' - \tau(a)) - \sum_{a \in \delta^+(v)} \sum_{t'=0}^{t} f(a, t') \qquad (1)$$

where $\delta^-(v)$ and $\delta^+(v)$ denote all arcs entering and leaving node v, respectively. Using the excess of flow f, the LP of a maximum dynamic flow reduces to:

$$
\begin{aligned}
\max \quad & excess_f(d,T) & & \qquad (2)\\
s.t. \quad & excess_f(v,t) = 0 & & \forall v \in V \setminus \{s,d\}, t \in \{0,...,T\}\\
& f(a,t) \leq u(a) & & \forall a \in A, t \in \{0,...,T\}\\
& f(a,t) \geq 0 & & \forall a \in A, t \in \{0,...,T\}
\end{aligned}
$$

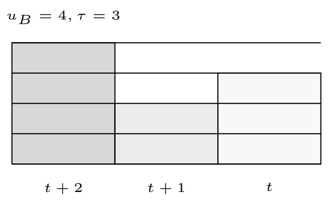
Solving LP 2 yields a maximum dynamic flow without waiting. However, as we stated earlier for MDFs Ford and Fulkerson [FJF58] developed a polynomial procedure using path flows only. The constraint set of (2) is the same for QF and EAF since the main difference occurs in the objective function. However, in this paper we want to extend the work on another variation of the MDF which differs only in one constraint.
Melkonian [Mel07] introduced a dynamic network problem with aggregate arc capacities called *Bridge Problem* (BP). In contrast to the general dynamic flow problem in which the capacity limits the amount of flow that can enter an arc every time step, the corresponding bridge capacity limits the overall value of flow that can be on an arc at any given time. As the name suggests, the BP is important whenever bridges are involved in the modeling. In practice, bridges should be built such that they do not need a special consideration, i.e. the bridge capacity should be large enough to handle any possible traffic of cars that are allowed to pass them. However, there are many hazards that might have a negative impact on the stability of the bridge (flooding, earth quakes, storm, etc.). For that reason, an appropriate modeling is even more important. Whereas Melkonian [Mel07] focuses on the *pure* linear BP, i.e. networks where every arc is labeled with bridge capacities and without an integer constraint on the flow, we also consider a *hybrid* version of the BP and restrict the flow to be integer. Therefore, we have given a network G where each arc can either be labeled with a bridge capacity or with a general capacity. For this problem we give some heuristics for general networks and deduce polynomial procedures for special classes of graphs. Moreover, we show that the pure integer BP is strongly NP-hard. Highlighting the difference between the bridge problem and the general dynamic flow problem, we talk in the remaining of this paper about bridge- and highway problem, respectively. Moreover, with respect to bridge - and highway problem we call the corresponding capacities, bridge - and highway capacities.

In the next section we give a formal definition of the pure BP and an overview of existing results. Strengthening a complexity result of [Mel07] we show that the integer pure BP is strongly NP-hard. Section 3 deals with the hybrid version of the bridge problem for which we discuss a heuristic similar to the one presented by Melkonian [Mel07], but working for integer flows as well. Also, we propose procedures for special classes of graphs that run in polynomial time. Computational results in Section 4 demonstrate the quality of the new heuristic.

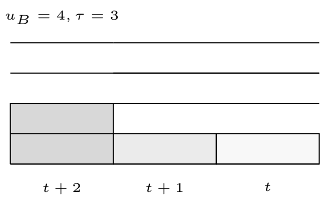
2 The Pure Bridge Problem

Note that in this version of the problem every arc in the network has a bridge capacity. We call network flow problem of this type *pure bridge problem.* Following Melkonian [Mel07], we consider the same network $G = (V, A, u, \tau)$ as before, but replace u by a bridge capacity $u_B : A \to \mathbb{N}$. Recall that $u(a)$ gives an upper bound on the flow quantity that can enter arc a at any time step. This model fails when considering a bridge which can only carry a certain amount of weight. Rather than restricting the capacity of inflow per time, here the bridge capacity limits the overall load on the "bridge" at any time. The difference should become clear in the following example.

Example 1
Figure 1 shows the difference between the two kinds of capacity considered in this paper. Assuming we are considering a road network of which Figure 1a illustrates one segment e.g. a highway with four lanes. Hence, every time unit four cars are able to access this segment of the highway. For Figure 1b we assume the indicated segment stands for a bridge. Because of its construction, this bridge is not able to take more than 4 car units at the same time. When at time t four car units enter the bridge no other car is allowed to get on it until time $t + 3$, since we need three time units to traverse it. Obviously, different distributions of flow over time as in Figure 1b are also possible.



(a) Using general flow capacities



(b) Using bridge capacity

Figure 1: Two ways of interpreting capacities with $u = u_B = 4$ and $\tau = 3$

From this example one can see what motivates the names of both problems. Melkonian [Mel07] introduced an LP to solve the bridge problem. Basically, it is the same as LP (2) but we replace the general capacity constraint by

$$\sum_{t=t'}^{t'+\tau(a)-1} f(a,t) \leq u_B(a) \quad \forall a \in A, t' \in \{0, ..., T - \tau(a).\} \tag{3}$$

Although, bridge and highway problem only differ in one constraint, there is no combinatorial algorithm for BP yet. Melkonian proved that BP is weakly $\mathcal{NP}$-hard, even for series parallel graphs with unit capacities. We can only solve it in pseudopolynomial time using the LP formulation. Moreover, in contrast to the highway problem, the solution of the LP does, in general, not yield an integer optimal solution to the BP since the total unimodularity of the coefficient matrix is destroyed.
Dressler et. al. [DS11] present an FPTAS for the bridge problem. They use an appropriate discretization of time to speed up the computation time.

A more intuitive way to tackle the bridge problem by a heuristic approach was introduced in [Mel07]. The idea is to transfer bridge capacities into highway capacities by setting $u_H := \frac{u_B}{\tau}$ and solving the problem with temporally repeated flows in (V, A, u_H, τ). One can easily see that the resulting flow is also feasible for the bridge problem. However, as the following example shows, it is in general not optimal.

Example 2
We consider a network that consists of a single arc a, with labels $u_B = 4$ and $\tau = 2$. Given a time horizon $T = 6$, we can send the following amount of flow and fulfilling the bridge constraints:

$$f_B = \{0 \to 4, 2 \to 4, 4 \to 4\}$$

which leads to a flow value $val(f_B) = 12$. Here, $\{\tau_1 \to x_1, ..., \tau_l \to x_l\}$ represents the computed flow by mapping each time step τ_i where flow is sent, to the corresponding flow value x_i provided $x_i \neq 0$. By sending the maximum amount of flow at the earliest time and keeping the bridge satisfied until the end, this flow is optimal for a bridge problem. Next, we compute the maximum dynamic flow of the highway problem using $u_H = 2$ by definition. Hence, at each time unit we can send 2 units of flow:

$$f_H = \{0 \to 2, 1 \to 2, 2 \to 2, 3 \to 2, 4 \to 2\}$$

This only leads to a flow value of $val(f_H) = 10$.

Even though this heuristic, which we call subsequently *highway heuristic*, leads to a lower bound and is feasible for the bridge LP, it is not necessary feasible for its IP. By using non-integer u_H as capacities for the highway problem the integral property does not need to be satisfied. This results in solutions which might not be feasible for the integer BP.

In the context of applications, the integer version of the bridge problem is of major importance. Unfortunately, the following result shows that it is unlikely that we will be able to solve this problem in polynomial time.

Theorem 1
The integer pure BP is strongly $\mathcal{NP}$-hard even for unit capacities.

Proof. The claim can be shown by reduction from 3-Partition (see [GJ90]) by following an idea from [PSY93].
Consider an instance of the 3-Partition: Given three sets of n integers $A = \{a_1, ..., a_n\}$, $B = \{b_1, ..., b_n\}$ and $C = \{c_1, ..., c_n\}$, and $L \in \mathbb{N}$ with $L = \frac{1}{n} \sum_{i=1}^{n} (a_i + b_i + c_i)$. Moreover, we have $a_i, b_i, c_i \geq \frac{L}{4}$ for $i = 1, ..., n$. Then we want to know if there is a partition of n disjoint sets S_j of $A \cup B \cup C$, respectively, with S_j containing at least one element of A, B and C, with $\sum_{s \in S_j} s = L$ for $j = 1, ..., n$.
Given this instance of 3-Partition we construct an instance for the bridge problem. We have four nodes s, u, v and d. Between s and u we add n parallel arcs with unit capacities and durations a_i for $i = 1, ..., n$. We do the same between

u and v with durations b_i and between v and d with duration c_i (see Figure 2). The time horizon T is set to L. In the corresponding decision problem of BP we ask if there is a flow with value M that can be pushed through the network within the given time horizon T. Thus, we set M to n.
First, we assume there is a solution for 3-partition. Given n solution sets $S_j = \{a_{j_1}, b_{j_2}, c_{j_3}\}$ we obtain a feasible flow for the bridge problem by sending one unit of flow over the corresponding arcs of our network. By construction the flow is feasible and the flow value equals to $n = M$.
Now we assume that we have a feasible flow for the bridge problem with value M. For each unit of flow that reaches d we know by feasibility that the sum over the durations of traversed arcs is less than or equal to $T = L$. Assume we sent two units of flow over one arc, say arc a. After sending the first unit at time zero we have to wait $\tau(a) = a_i$ time units until we can send again. However, by construction all durations are greater than $\frac{L}{4}$. The next flow cannot be sent before time $t = \frac{L}{4}$ and needs more than $3 \cdot \frac{L}{4}$ time units and reaches d after time $L = T$, which leads to a contradiction. Thus, we can only send flow once over an arc. For each unit of flow that reaches d we build a set S_i by including the integer durations of the arcs which were passed. Since n units of flow reach the destination we obtain n subsets with sum of included elements smaller or equal to L. Equality is reached since $L = \frac{1}{n}\sum_{i=1}^{n}(a_i + b_i + c_i)$ holds. $\square$

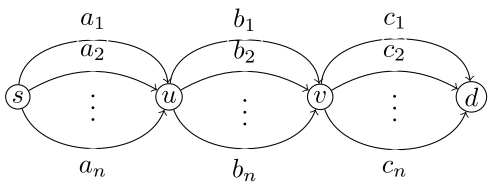


Figure 2: Reduction from the 3-Partition with unit capacities and labeled durations.

3 The Integer Hybrid Bridge Problem

In practice, a street network does not only consist of bridges. Thus, for practical relevance it is important to consider a hybrid version of BP. Given a network $G = (V, A, u, \tau)$ with arc labels for capacity and duration. However, we have some arcs with highway capacity u known from the general maximum dynamic flow problem (highway version), and some with label u_B, which means those arcs are representing bridges. Without loss of generality we assume that arcs $s \in A$ have either a highway capacity $u(a)$ or a bridge capacity $u_B(a)$, but not both. For those arcs with bridge capacity we add constraint (3) into the IP. With A_B being the set of all bridge arcs we can model the integer hybrid bridge problem by the following IP.

$$\begin{aligned}
\max \quad & excess_f(d,T) && && (4)\\
s.t. \quad & excess_f(v,t) = 0 && \forall v \in V \setminus \{s,d\}, t \in \{0,...,T\}\\
& f(a,t) \leq u(a) && \forall a \in A \setminus A_B, t \in \{0,...,T\}\\
& \sum_{t'=t}^{t+\tau(a)-1} f(a,t') \leq u_B(a) && \forall a \in A_B, t \in \{0,...,T-\tau(a)\}\\
& f(a,t) \geq 0 && \forall a \in A, t \in \{0,...,T\}\\
& f(a,t) \in \mathbb{Z} && \forall a \in A, t \in \{0,...,T\}
\end{aligned}$$

This hybrid problem can be solved heuristically by the highway heuristic transferring the bridge capacities to u_H and use a temporally repeated flow. However, we do have the same issue as before when it comes to the integer version, since we cannot guarantee that the resulting solution is feasible or brings any boundary condition.
In this section we will focus on a heuristic for the integer hybrid BP and special cases for which we can compute the flow value in polynomial time. We first develop a polynomial algorithm for the most basic graph class, namely a *Path Graph*, which becomes an important tool in the following sections. Thus, we have given a network that only consists of consecutive arcs and only one of those arcs is labeled with a bridge capacity (see Figure 3 for an example). The arc representing the bridge is denoted by a_B.

For these kind of networks Algorithm 1 outputs the optimal flow and flow

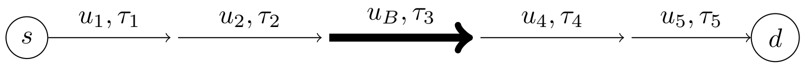


Figure 3: Example for a path graph containing one bridge only

value in polynomial time. In the first case, when the bridge arc carries the lowest capacity (i.e. $u_B \leq u^* := \min_{a \in A \setminus A_B} u(a)$), we cannot send more than u_B units of flow every τ_B time units, without violating the bridge constraint. If a highway arc carries the lowest capacity (i.e. $u^* < u_B$), we have to distinguish two cases: If the bottleneck capacity u^* is smaller or equal than $\frac{u_B}{\tau_B}$, by construction we can send u^* units of flow every time step without ever violating the bridge capacity. Then, the solution results in a TRF. In case $u^* > \frac{u_B}{\tau_B}$ we have to determine how often we can send u^* along a_B without violating the bridge constraint (Step 7). By sending r units of flow (i.e. the rest which still fits on the bridge arc) one time step later we guarantee optimality. After sending r units we stop sending flow until time τ_B. From there we start over again and continue until time $T - \tau(P)$. The comparison of y and n in Step 14 helps to determine the objective value. Since the sent flow is repeated every τ_B time steps we call the resulting flow a *modified temporally repeated flow*.

By construction the output flow fulfills all constraints and is optimal.

Proposition 1
Algorithm 1 outputs an optimal flow for series graphs with one bridge.

Algorithm 1 An algorithm solving the max dynamic flow problem for serial networks with one bridge

INPUT: G path graph with capacities u_a $\forall a \in A \setminus \{a_B\}$ and u_B, durations τ_a $\forall a \in A$
OUTPUT: maximum dynamic flow f^* and $val(f^*)$.

```
1: $u^* := \min_{a \in A \setminus \{a_B\}} u_a$
2: $\tau(P) := \sum_{a \in A} \tau_a$
3: if $u^* \geq u_B$ then
4:     $f^* = \{0 \to u_B, \tau_B \to u_B, 2\tau_B \to u_B, ..., \lfloor \frac{T-\tau(P)}{\tau_B} \rfloor \tau_B \to u_B\}$
5:     $\text{val}(f^*) = (\lfloor \frac{T-\tau(P)}{\tau_B} \rfloor + 1)u_B$
6: else
7:     Determine $n, r$ s.t. $u_B = n \cdot u^* + r$
8:     if $\tau_B \leq n$ then
9:         $f^* = \{0 \to u^*, 1 \to u^*, 2 \to u^*, ..., T - \tau(P) \to u^*\}$
10:        $\text{val}(f^*) = (T - \tau(P) + 1)u^*$
11:    else
12:        $f^* = \{l\tau_B + k \to u^* : l\tau_B + k \leq T - \tau(P), k \leq n-1, k, l \in \mathbb{N}\} \cup$
           $\{l\tau_B + n \to r : l\tau_B + n \leq T - \tau(P), l \in \mathbb{N}\}$
13:        Determine $x, y$, s.t. $T - \tau(P) = x \cdot \tau_B + y$
14:        if $y \leq n$ then
15:            $\text{val}(f^*) = u_B \cdot x + y \cdot u^*$
16:        else
17:            $u_B(x+1)$
18:        end if
19:    end if
20: end if
```

3.1 Heuristic

For general networks with restriction of the bridge number per path we develop a recursive heuristic (see Algorithm 2 for the main procedure) similar to the one of Melkonian [Mel07], which additionally ensures an integer output.
We start with the recursive flow computation in step 1 of Algorithm 2. After the initialization of some parameters we build the highway graph G_H of the input graph G by replacing all bridge capacities by their corresponding highway capacities. On G_H we compute a TRF and store $P_1, ..., P_L$ from the resulting paths decomposition. For each path we transform the highway capacity of the bridge arc back to a bridge capacity by considering the relative allocation of flow on $P_1, ..., P_L$ that uses the bridge (Step 8). This might also change the possible flow value that can traverse that path. By applying the same modification to non-bridge arcs (Step 11), we ensure that the impact of an earlier reduction of capacity on the bridges is not that big. This way we use the information of the TRF which is optimal for the corresponding highway graph. Among the new capacities we determine the minimal non-bridge capacity and apply Algorithm 1 with the respective values.
By rounding up in Step 8 and 11 we might waste some leftover capacity. Thus, after an update of the capacities of G we check if there is still some path that

can carry flow and start Algorithm 3 again. The additional testing if val(f) > 0 guarantees that we will not get stuck in a loop. By rounding down in steps 8 and 11 we might send zero flow which can result in val(f) $= 0$ although there is a flow increasing path.
In case there is such a flow increasing path we will deal with it after returning to the main procedure. We determine again some paths $P_1, ..., P_L$. Then for every path we use directly the original capacities of G and apply Algorithm 1. By updating the capacities before continuing with the next path we ensure that the resulting flow is still feasible with respect to the capacity constraints.
Since there are no theoretical results on the computational accuracy of the LP or IP heuristic, Section 4 describes computational tests for both methods.

Algorithm 2 Main procedure: Heuristic for solving the Hybrid Bridge Problem on general graph with maximal one bridge on each paths

INPUT: $G = (V, A)$ capacities u_a, $\forall a \in A$, durations τ_a, $\forall a \in A$
OUTPUT: feasible dynamic flow f^* and $val(f^*)$.

```
1:  Apply Algorithm 3 on $G$. Output: flow $f$, value val($f$).
2:  if path exists in $G$ then
3:    Build highway graph $G_H$ of $G$ with $u_a^H = \frac{u_a}{\tau_a}$ if a represents a bridge and
      $u_a^H = u_a$ else.
4:    Compute TRF on $G_H$ with output $(P_i, \text{val}(P_i), \tau(P_i))$ for $i = 1, .., L$ and
      $\hat{f}$
5:    for $i = 1$ to $L$ do
6:      mincapap= "$inf$", bridgeduration= 0, bridgecap= "$inf$"
7:      if all capacities are $> 0$ on $P_i$ then
8:        for $a \in P_i$ do
9:          if a is a bridge then
10:           bridgeduration=$\tau_a$
11:           bridgecap= $u_a$
12:         else
13:           if $u_a <$ mincap then
14:             mincap=$u_a$
15:           end if
16:         end if
17:       end for
18:       Update capacities in $G$
19:       Apply Algorithm 1 with $u^* =$ mincap, $\tau(P) = \tau(P_i)$, $u_B =$ bridgecap
          and $\tau_B =$ bridgeduration. Add flow to solution storage $f$ and flow
          value to val($f$).
20:     end if
21:   end for
22: end if
23: return $f$ and val($f$).
```

3.2 Bridges on SP-Graphs

In this subsection we extend the result on path graphs to special case of series parallel graphs and present a polynomial algorithm. We assume that there is

Algorithm 3 Recursive flow computation

```
INPUT: $G = (V, A)$ capacities $u_a$, $\forall a \in A$, durations $\tau_a$, $\forall a \in A$
OUTPUT: feasible dynamic flow $f^*$ and $val(f^*)$.
 1: bridgeduration= 0, modbridgecap= "$inf$", mincap= "$inf$"
 2: Build highway graph $G_H$ of $G$ with $u_a^H = \frac{u_a}{\tau_a}$ if a represents a bridge and
    $u_a^H = u_a$ else.
 3: Compute TRF on $G_H$ with output $(P_i, \text{val}(P_i), \tau(P_i))$ for $i = 1, .., L$ and $\hat{f}$
 4: for  $i = 1$ to $L$ do
 5:   for $a \in P_i$ do
 6:     if a is a bridge then
 7:       bridgeduration=$\tau_a$
 8:       modbridgecap=$\left( \left\lfloor u_a \frac{val(P_i)}{\hat{f}} \right\rfloor \right)$
 9:     else
10:       if $u_a \frac{val(P_i)}{\hat{f}} <$ mincap then
11:         mincap=$\left( \left\lfloor u_a \frac{val(P_i)}{\hat{f}} \right\rfloor \right)$
12:       end if
13:     end if
14:   end for
15:   Apply Algorithm 1 with $u^* =$ mincap, $\tau(P) = \tau(P_i)$, $u_B =$ modbridgecap
      and $\tau_B =$ bridgeduration. Add flow to solution storage $f$ and flow value
      to val$(f)$.
16:   Update available capacities in $G$
17: end for
18: if path exists in $G$ and val$(f) > 0$ then
19:   Apply Algorithm 3 on G. Output: $f'$ and val$(f')$.
20:   Merge $f$ and $f'$ and add val$(f')$ to val$(f)$.
21: end if
22: return  $f$ and val$(f)$.
```

only one bridge at the end of the network. Before we discuss our algorithm for SP-graph, we first outline the basic idea of this procedure which works for general graphs as well. Be aware that waiting is allowed for the following discussion. Given a graph G (not necessary series parallel) with one bridge at the end. This way, we can consider the highway part (all arcs with highway capacity, in Figure 4 denoted by G') and the bridge separately. We first compute and earliest arrival flow on the highway part for the reduced time horizon $T - \tau_B$, with τ_B being the duration on the bridge arc and T the given time horizon. This way, we know that the maximum amount of flow reaches the bridge at every time step. Next, we send flow along the bridge whenever the bridge is not working to capacity and there is flow that is able to be send before $T - \tau_B$. By using an EAF we make sure that there is no other way flow can get over the bridge earlier which guarantees the optimality. Algorithm 4 states this procedure.
Since can compute a EAF only on pseudo-polynomial time ([HT94]) this holds for Algorithm 4, too.

Theorem 2
Algorithm 4 computes a MDF for graphs with one bridge at the end in pseudo-

polynomial time.

Algorithm 4 Maximum dynamic flow on general graphs with on bridge and waiting

```
INPUT: $G = (V, A, u, \tau)$ with one bridge $a_B$ at the end and time horizon $T$.
OUTPUT: feasible dynamic flow $f^*$ and $val(f^*)$.
1: Compute an EAF on the highway part og $G$ with time horizon $T - \tau_B$.
2: for $i = 1$ to $T - \tau_B$ do
3:     If a flow amount of $x$ is available to send along $a_B$ and $y$ flow units still
       fit on the bridge we send $\min(x, y)$ units of flow along $a_B$
4: end for
5: Delete flow that was send along $G$ but could not get on the bridge.
```

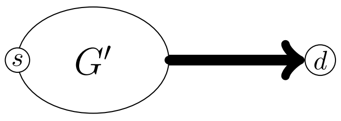


Figure 4: A graph G' with one bridge at the end.

The basic framework of Algorithm 4 we can apply to SP-graphs with one bridge at the end and without waiting. From general dynamic network flows we know how to obtain a maximal dynamic flow in G' efficiently (e.g. see Ford et. al. [FJF58]) using the concept of temporally repeated flows. Moreover, since we are dealing with series parallel (SP) graphs, we can compute the paths decomposition and the resulting TRF even more efficiently by using the results of Ruzika et. al. [RSM11]. The resulting maximal dynamic flow additionally fulfills the earliest arrival property. This way we are in the same set up as for algorithm 4 and can guarantee that at any time step the maximal possible amount of flow enters the bridge.

Let $P_1, ..., P_L$ be the flow paths from the TRF with τ_i the time needed to traverse P_i, for $i = 1, .., L$. Without loss of generality, we assume $\tau_i \geq \tau_{i+1}$, for $i = 1, ..., L-1$. Moreover, we denote by x_i the flow value that is sent along P_i every time step $t \in \{0, ..., T\}$. Next, we define a function that computes the value of flow that is on the bridge at a given time t and was sent along path P_i, for $i = 1, ..., L$.

$$f_i : [0, T - \tau_B] \to \mathbb{R}$$

$$f_i(t) = \begin{cases} 0 & \text{if } t \leq \tau_i - 1 \\ (t - \tau_i + 1)x_i & \text{if } t \in [\tau_i, \tau_i + \tau_B - 1] \\ \tau_B x_i & \text{else} \end{cases}$$

Since we are interested in the entire amount of flow on the bridge at a given time step $t \in \{0, ..., T\}$, we aggregate over all flow paths. This yields a new function $f : [0, T - \tau_B] \to \mathbb{R}$, with

$$f(t) = \sum_{i=1}^{L} f_i(t).$$

The linearity of f_i for $i = 1, ..., L$ implies that f is also piecewise linear, and hence, it is easy to determine the smallest $t \in \mathbb{N}$ such that $f(t) > u_B$ and denote $\lceil t \rceil$ by t^*. Additionally, we compute the largest τ_j and τ_k such that $\tau_j + \tau_B \leq t^*$ and $\tau_k \leq t^*$.
By construction $r := u_b - f(t^* - 1) > 0$ units of flow can be sent over the bridge at time t^* without violating the bridge capacity. Since $\tau_j + \tau_B \leq t^*$ we send a TRF as usual at every time step until $T - \tau_i - \tau_B$ over P_i, for $i = 1, ..., j$. For P_i with $i = j+1, ..., k$ we are dealing with a modified TRF, which is again τ_B periodic. Thus, we push x_i units of flow over P_i for

$$t \in \{l\tau_B, ..., l\tau_B + (t^* - 1) - \tau_i\}$$

as long as $t \leq T - \tau_i - \tau_B$ and for $l \in \mathbb{N}$. Finally, to exploit the bridge capacity completely we send r units of flow such that it reaches the bridge at time $t = l\tau_B + t^*$. Therefore, we use $P_{j+1}, ..., P_l$ with $l \leq k$ that together can carry the flow amount of r. Since for $i = 1, ..., j$ we use the general TRF and for $i = j+1, ..., k$ the we send the same amount of flow every τ_B period besides the last one, this results in a method that solves the bridge problem on SP-networks with one bridge at the end in polynomial time.

Algorithm 5 Maximum dynamic flow on SP-graphs with on bridge

INPUT: $G = (V, A, u, \tau)$ SP-graph with one bridge a_B at the end and time horizon T.
OUTPUT: feasible dynamic flow f^* and $val(f^*)$.

```
1: Compute a TRF the highway part of $G$ with flow paths $P_1, ..., P_L$ with $\tau_i \leq \tau_{i+1}$
2: Determine $f(t)$ for $t \in [0, T - \tau_B]$
3: Compute $t' = \min\{t | f(t) > u_B\}$ and set $t^* := \lceil t' \rceil$
4: Determine indexes $j$ and $k$ such that $\tau_j = \max\{\tau_i | \tau_i + \tau_B \leq t^*, i = 1, ..., L\}$ and $\tau_k = \max\{\tau_i | \tau_i \leq t^*, i = 1, ..., L\}$
5: For $i = 1, ..., j$, $f^*$ sends $x_i$ units of flow for $t = 0, ..., T - \tau_B - \tau_i$ along $P_i$.
6: For $i = j+1, ..., k$, $f^*$ sends $x_i$ units of flow for $t \in \{l\tau_B, ..., l\tau_B + (t^* - 1) - \tau_i\}$ as long as $t \leq T - \tau_B - \tau_i$ and for $l \in \mathbb{N}$.
7: For $i = j+1, ..., l$, $f^*$ sends the rate of $r$ using $P_i$ at time $t = l\tau_B + t^* - \tau_i$ for $t \leq T - \tau_B - \tau_i$ and $l \in \mathbb{N}$.
```

Theorem 3
Algorithm 5 computes a maximal integer bridge flow for SP-graphs with one bridge at the end.

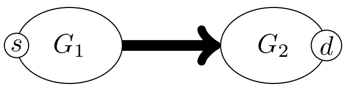
Proof. By construction the flow is feasible. Optimality follows from the fact that we use an EAF on the highway part and thus, there is no other flow for which the maximum bridge capacity is reached before t^*. Moreover, from that point on the bridge capacity is completely exhausted until $T - \tau_B$, where the last flow units can enter the bridge. Hence, there cannot be send more flow along the bridge arc and the procedure leads to an optimal flow. □

Fortunately, we can use the same procedure for graphs with one bridge at the beginning or somewhere in the middle. Before we show the details on this work, we need two more definitions.

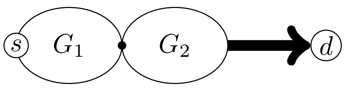
Definition 1

(a) *We call a graph G with one bridge,* bridge separable *if by deleting the bridge arc, G decomposes into two non-connected components. One contains the starting node of the bridge whereas the other one contains the end node.*

(b) *Consider two graphs G and $\overline{G}$ with one bridge, which are bridge separable. Then, we call G and $\overline{G}$* bridge equivalent *if they result in the same graphs when contracting the bridge. In case the bridge is somewhere in the middle, we merge the starting node of the bridge with its end node to obtain the new graph (see Figure 5).*



(a) Bridge separable graph with bridge in between.



(b) Bridge separable graph with bridge at the end.

Figure 5: Two graphs that are bride dependent equal.

Note that the graph in Figure 5b is bridge separable since it decomposes into the concatenation of G_1 and G_2 and the separate node d.

Theorem 4
Given an SP-graph with one bridge arc at the end. Then, the optimal solution obtained by Algorithm 5 is feasible for all bridge equivalent graphs.

Proof. Given the solution flow f computed by Algorithm 5. Let index j and k be as in the algorithm. Then, f pushes x_i units of flow over P_i at every time step $t \in \{0, ..., T - \tau_i - \tau_B\}$ for $i = 1, ..., j$. For $i = j + 1, ..., k$ f sends flow at times $t \in \{l\tau_B, ..., l\tau_B + (t^* - 1) - \tau_i\}$ for $l \in \mathbb{N}$ and as long as $t \leq T - \tau_i - \tau_B$. By construction of the flow we know that it is feasible on the highway part. Hence, it remains to show that f is feasible with respect to the bridge capacity on bridge equivalent graphs.
Therefore, we consider the maximum flow that can be on the bridge of a bridge equivalent graph. Over $P_1, ..., P_j$ we send flow every time step. Thus, the maximum amount of flow on a bridge sent along P_i for $i = 1, ..., j$ equals $\tau_B x_i$. For $i = j + 1, ..., k$ the flow sending pattern is τ_B periodic. This limits the amount of flow on the bridge which was sent along P_i by $(t^* - \tau_i)x_i$ which does not include the remaining amount. This is sent at t^* and from there on also in a τ_B periodic manner. The aggregated flow value on the bridge is bounded by

$$\sum_{i=1}^{j}(\tau_B x_i) + \sum_{i=j+1}^{k}((t^* - \tau_i)x_i) + r.$$

This is exactly what fits on the bridge on our original graph for which we computed f and thus, we have proven the claim. □

Next, we want to show why this solution is not only feasible but also optimal for bridge equivalent graphs. We consider the two cases of the bridge being at the beginning or at the middle of the graph. Figure 6 gives an example for a graph that is bridge equivalent to a graph as in Figure 4, where the bridge is at the beginning or at the end respectively. Similarly, both graphs in Figure 5 are also bridge equivalent, where Figure 5a illustrates a graphs with a bridge in the middle. For those kind of networks the following theorem holds:

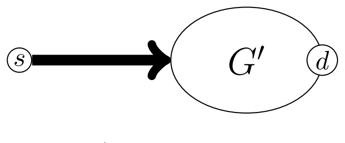


Figure 6: SP-graph G' with one bridge at the beginning

Theorem 5

(a) Solution f computed by Algorithm 5 is optimal for a bridge equivalent SP-graph with one bridge at the beginning.

(b) Solution f computed by Algorithm 5 is optimal for a bridge equivalent SP-graph with one bridge in the middle

Proof.

(a) We prove this claim by showing that for each feasible flow of Figure 6 there is an equivalent flow for Figure 4 with the same objective value.
We assume flow f' is feasible for the graph depicted in Figure 6 with first flow leaving s at time 0 and a time horizon T. Hence, f' is feasible for the same graph with shifted starting time $-T$ and time horizon 0. Using the arrival times of f' in the shifted version as starting times in Figure 4 we obtain an equivalent flow with the same objective value. Together with Theorem 4 this proves the claim.

(b) Assume we have a graph $\overline{G}$ as depicted in Figure 5a. Let f be the optimal solution of the bridge equivalent graph with one bride at the end (see Figure 5b) computed by Algorithm 5. Since the concatenation of G_1 and G_2 is series parallel, so is G_1. Thus, the corresponding path flows of f are also earliest arrival flows on G_1. Hence, it is sufficient to prove that no more flow can be sent over any path used by f and the bridge is completely working to capacity.
We cannot send any more flow along $P_1, ..., P_j$, since they are used consistently until $T - \tau_i - \tau_B$. Over P_i, for $i = j+1, ..., k$, we push flow only at time steps $t \in \{l\tau_B, ..., l\tau_B + (t^* - 1) - \tau_i\}$ for $l \in \mathbb{N}$ and then wait until the next τ_B period. Thus, if it is possible to send more flow in $\overline{G}$ than f it has to travel along any of those paths.
Let τ'_i denote the duration from s to the starting point of the bridge along P_i in G_1 for $i = 1, ..., k$.
For $P_{j+1}, ..., P_k$ we know that

$$\begin{aligned} 0 &< (\tau_{j+1} + \tau_B) - t^* && (5)\\ &\leq (\tau_{j+1} + \tau_B) - (\tau_k + x) && (6)\\ &\leq (\tau'_{j+1} - \tau'_k) + (\tau_B - x) && (7)\\ &= (\tau'_{j+1} + \tau_B) - (\tau'_k + x) && (8)\end{aligned}$$

where x denotes how often we send flow over P_k in one period minus one. Inequality (5) follows from the choice of index j in our procedure. Since no more flow from P_k can reach the bridge after t^*, (6) is true. The next inequality holds because the difference of reaching the bridge with respect to P_{j+1} and P_k can only get larger in the concatenation of G_1 and G_2 than just in G_1 (note that $\tau_{j+1} \leq \tau_k$). The reformulation in (8) states that before the first flow of P_{j+1} leaves the bridge, the last flow of P_k enters the bridge in a given period. Since $\tau_h \leq \tau_i$ for $h < i$ we have $\tau'_h \leq \tau'_i$ and thus, $\tau'_j + \tau_B \leq \tau'_{j+1} + \tau_B$.
Hence, before the first flow that travels along P_{j+1} leaves the bridge, we have $\tau_B x_i$ flow units on the bridge for $i = 1, ..., j$ and all the flow which is send over any other path P_i for $i = j+1, ..., k$. This yields a bridge load of

$$\sum_{i=1}^{j}(\tau_B x_i) + \sum_{i=j+1}^{k}((t^* - \tau_i)x_i) + r,$$

which is just the maximum of flow that fits on the bridge and proves that no more flow can be sent along any paths of f.

□

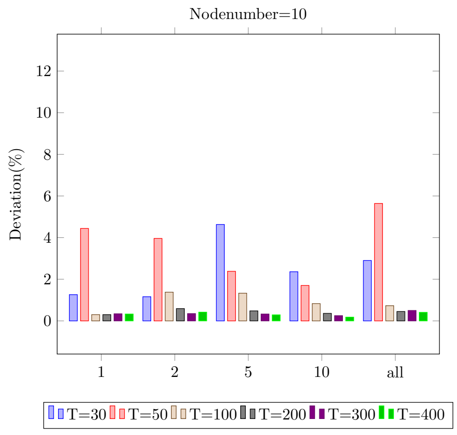
4 Numerical Examples

In this section we consider random networks and compare the output of the presented heuristics to the optimal value computed by the integer program (4). For a given node number and arc probability, we construct two random graphs with the same number of nodes and connect them by b arcs, where b denotes the number of bridges that are allowed. This is one way to ensure that on every path is at most one bridge. In this particular case we can even guarantee that there is exactly one. Durations as well as highway capacities are also assigned randomly by a uniform distribution given an upper bound. On these kind of graphs we tested the general highway heuristic proposed by Melkonian [Mel07] and the heuristic presented in Algorithm 2 and compared the results with the optimal solution of IP (4). All test are done on a 64 bit Linux compute server equipped with two Intel Xeon E5-2690 (single processor specification: nominal speed 2.9 gigahertz, boost up to 3.8 gigahertz, 8 cores, 16 threads, 20 megabyte Cache) and 192 gigabyte DDR-3 ECC-RAM at 1333 megahertz, making use of python 2.7.11, networkx 1.6. ([HSS08]) and Gurobi solver 6.5.1 ([GO15]).

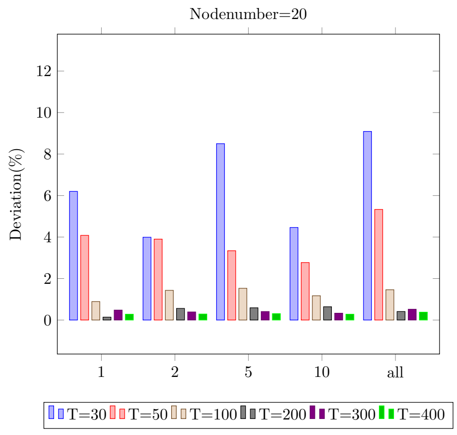
4.1 Highway Heuristic vs. LP Solution

We consider instances of various input data in node number, time horizon and number of bridges. For each instance we evaluated the test on ten different graphs.

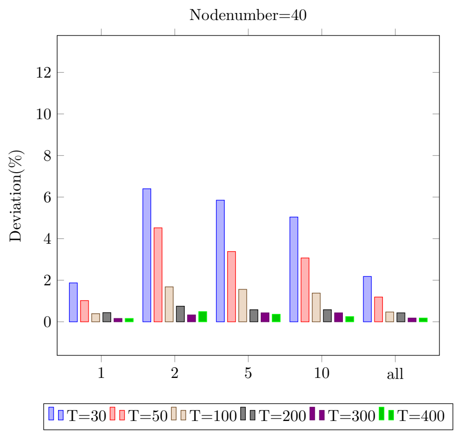
Figures 7a-7d illustrate the mean values of the relative errors for each group of instance with equal input data. Each graphic stands for a different node number. Time horizon and bridge number are distinguished by different colors and ordering along the x-axis, respectively. Additionally to bridge number one, two, five and ten we analyze when all arcs represent bridges.



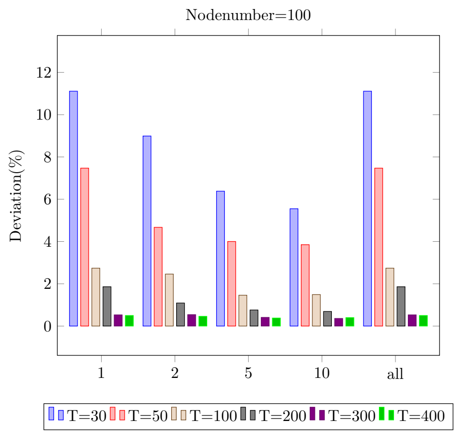
(a) Several instance data with node number equal to 10.



(b) Several instance data with node number equal to 20.



(c) Several instance data with node number equal to 40.



(d) Several instance data with node number equal to 100.

Figure 7: Relative error in comparison to the optimal solution of the LP with respect the different instance data.

What is evident in almost all four graphics is the decreasing error with increasing time horizon, even though there are a few outliers in Figure 7a. This result seems to be quite useful, since especially for a large time horizon T an IP solver becomes very slow. Moreover, the graphics indicate that the heuristic works reasonable well. In the mean value the relative error is for all instances below 12%, for larger time horizons it differs by not more than 2%.

Obviously a big advantage of the heuristic is the computation time compared to solving the corresponding LP. We consider one random graph with one hundred nodes and two bridges. Then for several time horizons we run the LP solver (Gurobi) and the LP heuristic. The time difference is depicted in Figure 8. One can see the computation time of the heuristic is not visible increasing with growing time horizon, whereas the LP computation consumes more and more time. Especially for larger instances the heuristic seems to be a promising

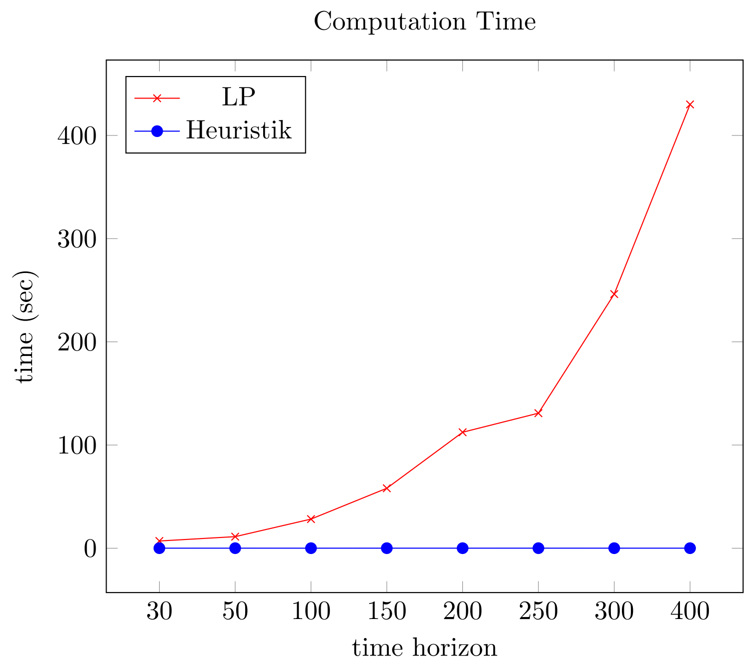


Figure 8: Time which is needed for the LP solver and heuristic in seconds for different time horizons.

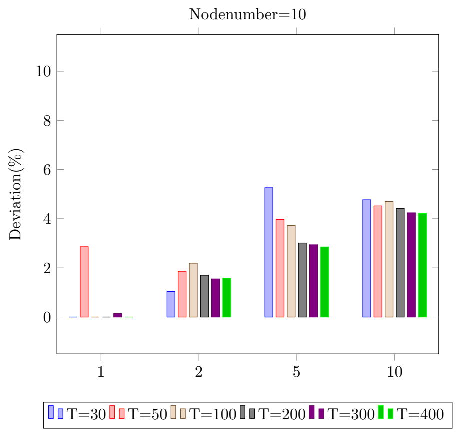
method.

4.2 Modified Highway Heuristic vs. IP Solution

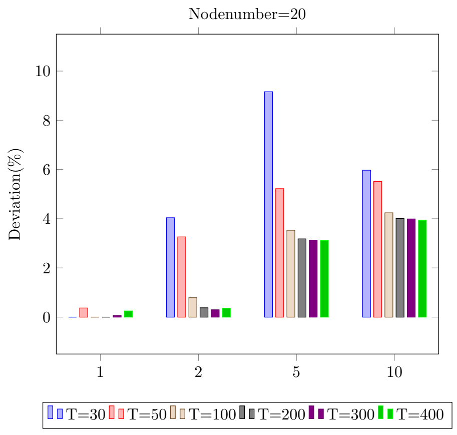
Next we did the same experiments with the same instances applying Algorithm 2 instead of the highway heuristic. Since Algorithm 2 makes use of Algorithm 1 which only works for serial graphs with one bridge, our heuristic is strictly limited to networks that fulfill our assumption of maximum one bridge per path. Hence, there will be mo results for graphs which contains bridge arcs only.

Figures 9a to 9d illustrate again the relative error with the same setting as for the LP. However the resulting figures look different. We still can notice that the error decreases with a larger time horizon. However, this characteristic is less distinct as in Figure 7. Instead, the deviation from the optimum are more similar for the same node- and bridge number but with a light decreasing error with respect to the time horizon. Also the overall performance of the heuristic seems quite good since the mean error is always below 10% and for higher time horizon even below 6%.

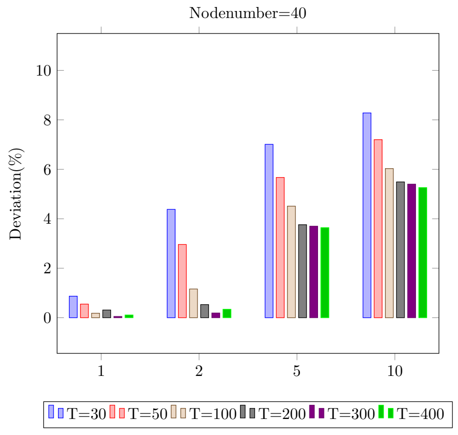
Similar as for the LP we also compare the running time of both methods. Figure 10 presents the two curves describing the computation time with respect to some chosen time horizons. As in Figure 8 this points out that the heuristic leads to a solution much faster than an IP solver (we again used Gurobi 6.5.1).



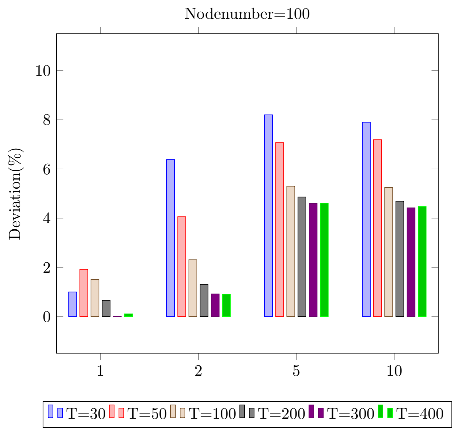
(a) Several instance data with node number equal to 10.



(b) Several instance data with node number equal to 20.



(c) Several instance data with node number equal to 40.



(d) Several instance data with node number equal to 100.

Figure 9: Relative error in comparison to the optimal solution of the IP with respect the different instance data.

5 Conclusion

The bridge problem improves the modeling of traffic whenever bridges are involved. Especially, when considering evacuation problems we should not neglect bridges, in particular damaged bridges need some special attention. Hence, we feel that controlling the hybrid bridge problem is of great importance and deserves more attention. The presented methods in this paper already give some promising results for SP-graphs. We showed that we can handle them for special cases and in fact can compute an optimal solution in polynomial time. For general networks with at most one bridge on a path we proposed a new heuristic. Computational results showed that this heuristic works well in most cases and

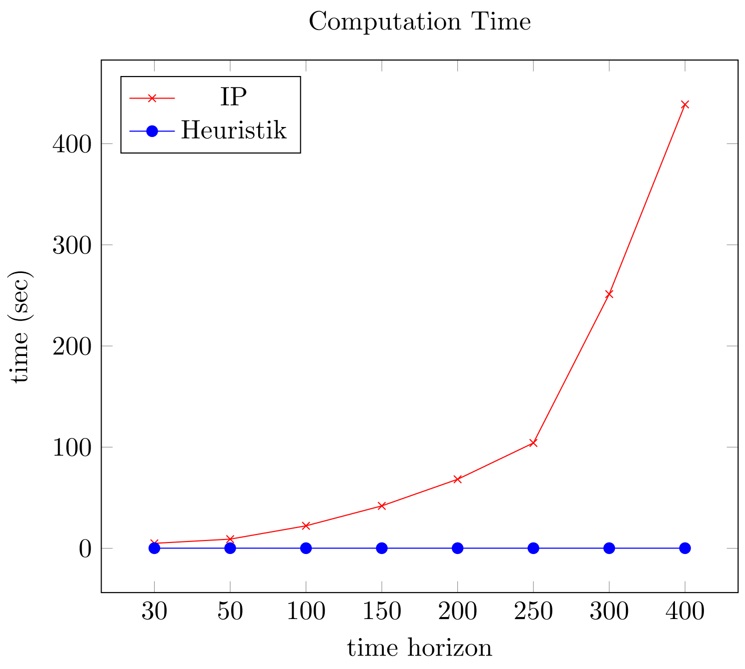


Figure 10: Time which is needed for the LP solver and heuristic in seconds for different time horizons.

require much less time than any IP solver. Hence, this approach can be rather useful for practice as well.

References

[AMO93] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. Network flows: theory, algorithms, and applications. 1993.

[BDK93] R. E. Burkard, K. Dlaska, and B. Klinz. The quickest flow problem. *ZOR Zeitschrift für Operations Research Methods and Models of Operations Research*, 37(1):31–58, 1993.

[DS11] D. Dressler and M. Skutella. An FPTAS for flows over time with aggregate arc capacities. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6534 LNCS:106–117, 2011.

[FF10] D. R. Ford and D. R. Fulkerson. *Flows in Networks.* Princeton University Press, Princeton, NJ, USA, 2010.

[FJF58] L. R. Ford Jr. and D. R. Fulkerson. Constructing maximal dynamic flows from static flows. *Operations Research*, 6(3):419–433, 1958.

[FS07] L. Fleischer and M. Skutella. Quickest Flows Over Time. *SIAM Journal on Computing*, 36(6):1600–1630, 2007.

[Gal59] D. Gale. Transient flows in networks. *Michigan Math. J.*, 6(1):59–63, 1959.

[GJ90] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, NY, USA, 1990.

[GO15] Inc. Gurobi Optimization. Gurobi optimizer reference manual, 2015.

[HSS08] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pages 11–15, Pasadena, CA USA, August 2008.

[HT94] B. Hoppe and E. Tardos. Polynomial time algorithms for some evacuation problems. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '94, pages 433–441, Philadelphia, PA, USA, 1994. Society for Industrial and Applied Mathematics.

[HT01] H. W. Hamacher and S. A. Tjandra. *Mathematical modelling of evacuation problems: A state of art.* Fraunhofer-Institut für Techno-und Wirtschaftsmathematik, Fraunhofer (ITWM), 2001.

[Mel07] V. Melkonian. Flows in dynamic networks with aggregate arc capacities. *Information Processing Letters*, 101(1):30–35, 2007.

[Min73] Edward Minieka. Maximal, lexicographic, and dynamic network flows. *Oper. Res.*, 21(2):517–527, April 1973.

[PSY93] C. H. Papadimitriou, P. Serafini, and M. Yannakakis. Computing the throughput of a network with dedicated lines. *Discrete Applied Mathematics*, 42(2):271 – 278, 1993.

[RSM11] S. Ruzika, H. Sperber, and Steiner M. Earliest arrival flows on series-parallel graphs. *Networks*, 57(2):169–173, 2011.