Earliest Arrival Flows on Series-Parallel Graphs

Stefan Ruzika[1]*,**, Heike Sperber[1]**, and Mechthild Steiner[1]

Department of Mathematics, University of Kaiserslautern, P.O.Box 3049, 67653 Kaiserslautern, Germany.
{ruzika, sperber, steiner}@mathematik.uni-kl.de

Abstract. We present an exact algorithm for computing an earliest arrival flow in a discrete time setting on series-parallel graphs. In contrast to previous results for the earliest arrival flow problem this algorithm runs in polynomial time.

Keywords: dynamic network flows, earliest arrival flows, universally maximal dynamic flows, discrete time setting, series-parallel graphs, polynomial algorithms

1 Introduction

Many applications can be suitably modeled using network flows. The theory of network flows is well-understood and related problems can often be solved efficiently. Flows on networks have been extended in various ways to meet the growing requirements of demanding real-world scenarios. The assumption of a flow unit consuming time to traverse an arc in the network leads to one such extension, so-called *dynamic network flows* or *network flows over time*. Especially processes with an inherent temporal component of movement can be properly modeled by dynamic network flows. Examples are among others transportation problems, traffic flow modeling, or evacuation planning (see, e.g. [1] or [9]).

In the 1950s, Ford and Fulkerson initiate research on dynamic flows by introducing the maximum dynamic flow problem (see [5],

* The research of Stefan Ruzika has been partially supported by BMBF, Project REPKA, FKZ 13N9961.

** The research of Stefan Ruzika and Heike Sperber has been partially supported by the Center of Mathematical and Computational Modelling of the University of Kaiserslautern.

[6]). In this problem the goal is to find a shipping schedule sending a dynamic flow from source to sink within a given time horizon $T \in \mathbb{N}$. Ford and Fulkerson map this dynamic flow problem to a (static) flow problem via transformation into the so-called time-expanded network. In addition they propose an efficient algorithm based on the solution of a static minimum cost flow problem, the decomposition of the resulting flow, and the temporal repetition of the paths obtained.

In this article we focus on an extension of the maximum dynamic flow problem, the so-called *earliest arrival flow problem*. The difference between the two problems becomes obvious when considering an evacuation scenario. Modeling the egress movement of evacuees by a maximum dynamic flow problem asks for the maximum number of people reaching safety within a given time bound T. In addition, an earliest arrival flow requires the number of people reaching safety to be maximal at every time step $0 \leq \theta \leq T$.

Gale [7] shows the existence of (discrete time) earliest arrival flows for networks with a single source and a single sink. Minieka [12] and Wilkinson [15] propose exact algorithms which may need exponential running time. Hoppe and Tardos [10] develop a fully polynomial-time approximation scheme for the earliest arrival flow problem with single source and single sink. Their approximation algorithm delivers at least $(1-\epsilon)$ times the amount of flow that should have reached the sink up to time θ, for every $\theta \leq T$. In networks with multiple sources and multiple sinks with given supplies and demands, an earliest arrival flow does not necessarily exist (see [2]). For the case of multiple sources and single sink Baumann and Skutella [2] show existence and give an algorithm the running time of which is polynomial in the input and output size of the instance. Tjandra [14] proposes an algorithm for problems with time-dependent transit times and capacities. This algorithm is polynomial in the time horizon T and the maximum capacity.

Dynamic problems on networks include among others the quickest flow problem (see [4]), the dynamic minimum cost flow problem (see [11]), or the multi-commodity flow problem over time (see [8]). Klinz and Woeginger [11] show that the minimum-cost maximum dynamic flow problem on series-parallel networks is NP-hard.

Our Contribution We constructively prove the existence of a temporally repeated flow on series-parallel graphs having the earliest arrival property. Our exact greedy approach is based on a result of Bein et al. [3] for solving the minimum cost flow problem on series-parallel graphs and runs in polynomial time.

Road Map We introduce some notation and recall preliminary results in Section 2 and establish our results about earliest arrival flows on series-parallel graphs in Section 3. Afterwards we recapitulate the work in Section 4.

2 Preliminaries and Notation

Let $G = (N, A)$ be a *directed graph*, where N is the set of all nodes, including the source s and the sink t, and A denotes the set of all arcs. For an arc $a \in A$, we denote by $t(a)$ and $h(a)$ the *tail* and *head (node)* of a, respectively. A *(directed) path* $P = (i_0, a_1, i_1, a_2, \ldots, a_k, i_k)$ is an alternating sequence of nodes and arcs with $i_\nu = t(a_{\nu+1}) = h(a_\nu)$ for all $\nu = 1, \ldots, k-1$, $i_0 = t(a_1)$, and $i_k = h(a_k)$. A *(directed) cycle* C is a path with $i_k = i_0$.

In a *dynamic network*, every arc $a \in A$ is equipped with a capacity $u_a \in \mathbb{N}_0$, corresponding to the maximal amount of flow which may enter the arc per time period, and a transit time $\tau_a \in \mathbb{N}_0$. The latter describes how long it takes one unit of flow to travel from node $t(a)$ to $h(a)$ on arc $a \in A$. Furthermore, we assume a finite time horizon $T \in \mathbb{N}$ for the flow to travel through the network and we consider discrete time steps.

A *dynamic network flow* is a function $x\colon A \times \{0, \ldots, T\} \to \mathbb{R}_0^+$. The flow which enters arc a at time θ is denoted by $x_a(\theta)$. Due to our model assumptions, this flow reaches node $h(a)$ at time $\theta + \tau_a$.

Let x be a static network flow, i.e., a function $x\colon A \to \mathbb{R}_0^+$. In the *residual network* $G(x) = (N, A^+ \cup A^-)$ of $G = (N, A)$ with respect to static flow x, the sets A^+ and A^- are defined as follows. The set A^+ contains all arcs $+a = a$ with $x_a < u_a$. To each of these arcs we assign a capacity of $u_a - x_a$ and a transit time of τ_a. The set A^- consists of all reversed arcs $-a = (h(a), t(a))$ with $x_a > 0$. Each of these arcs gets a capacity of x_a and a transit time of $-\tau_a$.

The maximal dynamic flow problem of finding a feasible dynamic flow which sends the maximal amount of flow from source s to sink t up to time T can be formulated as a linear program. Due to intended brevity we assume in the following that there are neither arcs a with $h(a) = s$ nor $t(a) = t$.

$$\max \; v \tag{1a}$$

$$\text{s.t.} \quad \sum_{\substack{a \in A \\ t(a)=s}} \sum_{\theta=0}^{T} x_a(\theta) = v \tag{1b}$$

$$\sum_{\substack{a \in A \\ h(a)=t}} \sum_{\theta=0}^{T} x_a(\theta - \tau_a) = v \tag{1c}$$

$$\sum_{\substack{a \in A \\ h(a)=i}} x_a(\theta - \tau_a) = \sum_{\substack{a \in A \\ t(a)=i}} x_a(\theta) \quad \forall i \in N \setminus \{s,t\}, \theta \in \{0,\dots,T\} \tag{1d}$$

$$0 \le x_a(\theta) \le u_a \qquad \forall a \in A, \theta \in \{0,\dots,T\} \tag{1e}$$

The objective (1a) maximizes the total flow value v which can be sent from source s to sink t within time horizon T. Constraints (1b), (1c) and (1d) ensure flow conservation at the nodes. Note that flow arriving in $h(a)$ at time θ must leave node $t(a)$ at time $\theta - \tau_a$. Constraints (1e) limit the amount of flow which may enter arc a at time $\theta \in \{0,\dots,T\}$. Every *feasible dynamic flow* has to fulfill constraints (1b) to (1e). Storage of flow in a node is not needed throughout this article.

One approach to solve the maximal dynamic flow problem is the transformation of this dynamic problem into a static maximal flow problem in the *time-expanded network* $G_T = (N_T, A_T)$. This network is obtained by an expansion of the dynamic network: Each node i of the underlying (static) graph is copied T times to obtain a node $i(\theta)$ for each $i \in N$ and each $\theta \in \{0,\dots,T\}$. For each arc $a \in A$ and $\theta \le \max\{0, T - \tau_a\}$, we introduce a copy with tail $i(\theta)$ and head $j(\theta + \tau_a)$ where i is the tail and j the head of arc a. A capacity of u_a is assigned to each of these arcs.

A dynamic flow in graph G corresponds to a static flow in this time-expanded network G_T and vice versa. Time-expanded networks are useful tools to solve dynamic network flow problems since they allow the application of solution methods for static network flow problems. However, the size of a time-expanded network depends on the time horizon T. Thus, every algorithm constructing G_T explicitly is not polynomial.

This drawback of time-expanded networks can be overcome for the maximal dynamic flow problem by using a polynomial algorithm on the underlying static network $G = (N, A)$ (see Ford and Fulkerson [5]): Add an additional arc $\bar{a}$ from sink to source with infinite capacity and transit time $-(T+1)$. The transit times τ_a on the arcs of this modified network $\bar{G} = \left(N, \bar{A}\right)$ with $\bar{A} := A \cup \{\bar{a}\}$ are then interpreted as costs of arcs and a minimum cost circulation problem is solved. To obtain a maximal dynamic flow, the flow on arc $\bar{a}$ is neglected and the resulting flow from s to t is decomposed into paths. Along each path its (static) flow value, obtained in the path decomposition, is sent at each time step for which the flow can reach the sink before time T. Flows obtained by this technique are called *temporally repeated flows*. Ford and Fulkerson [5] show that there always exists a temporally repeated flow which solves the maximal dynamic flow problem.

A problem closely related to maximal dynamic flows is the *earliest arrival flow problem* asking for a feasible dynamic flow from s to t which is maximal *at all times* $0 \leq \theta \leq T$.

Obviously, every earliest arrival flow is also a maximal dynamic flow. The converse implication is not true in general and, even more, on general graphs there might not exist any temporally repeated flow having the earliest arrival property (see [13]).

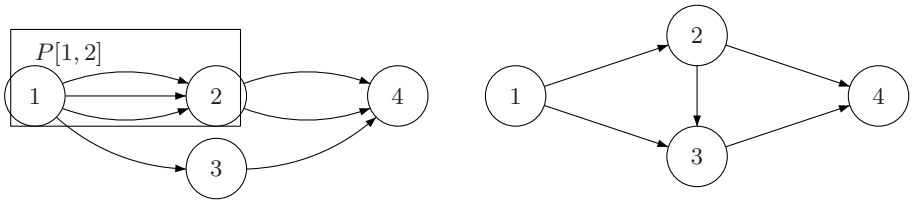
In Section 3 we show that on series-parallel graphs there always is a temporally repeated flow that is maximal and earliest arrival. Series-parallel graphs are a proper subset of acyclic digraphs and defined recursively as follows: A single arc $a = (\tilde{s}, \tilde{t})$ is series-parallel with start-terminal $\tilde{s}$ and end-terminal $\tilde{t}$ by definition. Let G_i be series-parallel with start-terminal $\tilde{s}_i$ and end-terminal $\tilde{t}_i$ $(i = 1, 2)$. Then the graph $S(G_1, G_2)$ obtained by identifying $\tilde{t}_1$ as $\tilde{s}_2$ is a series-parallel graph, with $\tilde{s}_1$ and $\tilde{t}_2$ as its terminals (series composition).

The graph $P(G_1, G_2)$ obtained by identifying $\tilde{s}_1$ as $\tilde{s}_2$ and also $\tilde{t}_1$ as $\tilde{t}_2$ is a series-parallel graph (parallel composition) with $\tilde{s}_1 (= \tilde{s}_2)$ and $\tilde{t}_1 (= \tilde{t}_2)$ as its terminals.

Figure 1a shows an example of a series-parallel graph and Figure 1b the "smallest" graph which is acyclic but not series-parallel.

For vertices i and j in a series-parallel graph, we denote by $P[i, j]$ the subgraph of G including all paths from i to j. Observe that this might be the empty set. For example in the series-parallel graph shown in Figure 1a there is no path from 2 to 3 and $P[1, 2]$ contains all three parallel edges from 1 to 2.

$P[i, j]$ is the inclusionwise maximal composition having i and j as its terminals in the iterative construction of the series-parallel graph and thus any s-t-path using an edge of this subgraph must use both i and j.



(a) Series-parallel graph

(b) Not series-parallel graph

Algorithm 1 - which is due to Bein et al. [3] - solves the minimum cost flow problem on series-parallel graphs in polynomial time $O(|N| \cdot |A| + |A| \cdot \log(|A|))$ for all flow values up to the maximal flow value $v_{\max}$. Starting with the zero flow, Algorithm 1 uses a greedy approach and iteratively finds the currently cheapest path P_k with cost $\tau(P_k) = \sum_{a \in P_k} \tau_a$ and assigns the flow value $v(P_k)$ to it. The latter is calculated as the maximal remaining capacity on the arcs of the path. Afterwards the capacity on the arcs of P_k is updated.

The output $(P_k, \tau(P_k), v(P_k))$ for $k = 1, \ldots, q$ for some $q \leq |A|$ describes a continuous, piecewise linear and convex function where $\tau(P_k)$ are the slopes of the linear parts and $v(P_k)$ indicates the length of the interval on which the function is linear. This function describes the objective function value of the minimum cost flow prob-

lem depending on the flow value. The maximal flow value is given by $v_{\max} = \sum_{k=1,\dots,q} v(P_k)$.

Algorithm 1 Minimum Cost Flows on Series-Parallel Graphs [3]

Input: Series-parallel graph $G = (N, A)$ with source s and sink t, costs τ_a and capacities u_a on each arc $a \in A$.
Output: Triples $(P_k, \tau(P_k), v(P_k))$ for $k = 1, \dots, q$ for some $q \leq |A|$.

1. $k := 1$ // initialization
2. **while** there exists a path connecting s and t **do**
3. Find a minimal cost s-t-path P_k and its costs $\tau(P_k)$ // path, cost
4. $v(P_k) := \min\{u_a \,|a \in P_k\}$ // flow on path
5. **for all** $a \in P_k$ **do**
6. $u_a := u_a - v(P_k)$ // decrease capacity on used arcs
7. **if** $u_a = 0$ **then**
8. then $A := A \setminus \{a\}$ // delete full arcs
9. **end if**
10. **end for**
11. $k := k + 1$
12. **end while**

3 Earliest arrival flows on series-parallel graphs

In this section we restrict our analysis to series-parallel graphs as the recursive structure of these graphs is crucial for our results. We modify Algorithm 1 to obtain a polynomial time algorithm solving the maximal dynamic flow problem using a temporally repeated flow. Then, we show that the output of the algorithm has the earliest arrival property.

Algorithm 1 solves the minimum cost flow problem by successive computations of shortest paths. Our modification takes into account that the cost of these paths should not exceed the time horizon T as flow being sent over longer paths would not arrive in time. Note that the introduction of the arc $\bar{a}$ in the algorithm of Ford and Fulkerson [5] for solving the maximal dynamic flow problem serves the same purpose. Thus, we add a condition on the cost of the paths retrieved by Algorithm 1: If the cost of path P_k is less than $T + 1$, then this path is considered as before in Algorithm 1. Otherwise the path is neglected and the algorithm stops.

Formally, we replace Line 4 in Algorithm 1 by Procedure 2 in order to get our modified Algorithm 2.

Procedure 2 Replacement of Line 4 in Algorithm 1

```
if τ(P_k) − (T + 1) < 0 then
        v(P_k) := min {u_a | a ∈ P_k }
else
        Stop the algorithm.                    // additional stopping criterion
end if
```

Checking this additional condition needs only constant time for each path. The algorithm might stop before all possible s-t paths are found. Thus, the running time of Algorithm 2 is $O(|A| \cdot |N| + |A| \log(|A|))$.

The output $(P_k, \tau(P_k), v(P_k))$ with $k = 1, \dots, q'$ for some $q' \in \mathbb{N}$ of Algorithm 2 induces a temporally repeated flow sending $v(P_k)$ flow units on path P_k at times $0, \dots, T - \tau(P_k)$. We show that this is indeed a maximal dynamic flow.

Theorem 1. *Algorithm 2 yields a maximal dynamic flow on a series-parallel graph $G = (N, A)$ with source s, sink t, and time horizon T.*

Proof. We show that Algorithm 2 solves the minimum cost circulation problem introduced by Ford and Fulkerson [5] for the maximal dynamic flow problem (see Section 2).

Let $(P_k, \tau(P_k), v(P_k))_{k=1,\dots,q}$ and $(P_k, \tau(P_k), v(P_k))_{k=1,\dots,q'}$ with $q' \leq q$ be the output of Algorithm 1 and Algorithm 2 on an arbitrary series-parallel graph, respectively. The notation is justified as both algorithms yield exactly the same paths except that Algorithm 2 might stop earlier.

We consider the graph $\bar{G} = (N, \bar{A})$ where $\bar{A} = A \cup \{\bar{a}\}$ with $\tau_{\bar{a}} = -(T + 1)$ and $u_{\bar{a}} = \infty$, i.e., we introduce an arc from sink to source and assign costs and capacity as in the minimum cost circulation network introduced by Ford and Fulkerson. Additionally, we extend the paths P_k, $k = 1, \dots, q'$ in G to cycles C_k in $\bar{G}$ by adding arc $\bar{a}$ and sending $v(P_k)$ flow units along $\bar{a}$. This yields a feasible circulation flow $\mathcal{C}$ in $\bar{G}$ and, by construction, each of the cycles C_k has negative costs.

It remains to show that every cycle in the residual network $\bar{G}(\mathcal{C})$ has nonnegative cost. Hence, the output of Algorithm 2 solves the minimum cost circulation problem due to the cycle optimality conditions (see [1]).

Consider an arbitrary cycle C in $\bar{G}(\mathcal{C})$. C either contains the arc $+\bar{a}$ or $-\bar{a}$. Otherwise there is a contradiction to Algorithm 1 solving the minimum cost flow problem for flow value $v := \sum_{k=1}^{q'} v(P_k)$.

If the cycle C contains the arc $+\bar{a}$, then the s-t-path induced by C has costs at least $T+1$ due to the revised stopping criterion of Algorithm 2, and hence C has nonnegative cost.

Suppose the cycle C contains the arc $-\bar{a}$. Denote by P the path from t to s induced by C. In the following we show that C has nonnegative cost which is equivalent to showing that P has cost at least $-(T+1)$.

> *Claim:* Without loss of generality P does not contain arcs in $\bar{A}^+$.
>
> *Proof of claim:* Assume there is an arc a in $P \cap \bar{A}^+$. Consider the longest (with respect to the number of arcs) connected subpath π of P containing arc a and only further arcs of $\bar{A}^+$. Let i and j be the end nodes of this subpath. Then, π is contained in $P[i,j]$. There is another path from j to i in the residual network $\bar{G}(\mathcal{C})$ using only arcs from $\bar{A}^-$ as $\mathcal{C}$ sends flow to j via i through $P[i,j]$.

Thus, we may assume that P consists only of arcs from $\bar{A}^-$. Algorithm 2 sends flow along some paths which contain some arcs a such that $-a \in P$. Among these paths denote by P_h the one with the largest index h. Let P_h^{rev} denote the path in the residual network $\bar{G}(\mathcal{C})$ obtained from P_h by reversing its arcs, i.e., for each arc a in P_h, P_h^{rev} contains the arc $-a$.

Consider the nodes $t = i_1, \ldots, i_l = s$ on which P and P_h^{rev} intersect. Each induced subpath π_ν of P between i_ν and $i_{\nu+1}$ for $\nu \in \{1, \ldots, l-1\}$ may coincide with the corresponding subpath of P_h^{rev}. Otherwise, π_ν uses arcs $-a$ induced by flow on paths with index smaller than h. We conclude that the cost of P in $\bar{G}(\mathcal{C})$ is not less than $\tau(P_h^{\text{rev}}) = -\tau(P_h)$ due to the greedy choice of paths in Algorithm 2. The revised stopping criterion of Algorithm 2 implies $-\tau(P_h) > -(T+1)$ and, hence, the cost of C cannot be negative.

The temporally repeated flow found by Algorithm 2 is maximal. In the following, we show that this flow has the earliest arrival property.

Theorem 2. *Algorithm 2 yields an earliest arrival flow on a series-parallel graph $G = (N, A)$ with source s, sink t, and time horizon T.*

Proof. Denote by x the temporally repeated maximal dynamic flow obtained from the output of Algorithm 2. Assume this flow does not have the earliest arrival property.

Let x^{EAF} be an earliest arrival flow in G_T. Minieka [12] shows that x^{EAF} can be obtained by adding a residual flow to x in $G_T(x)$. This residual flow must contain a flow on a path

$$P = (i_0(\theta_0), a_1, i_1(\theta_1), \dots, a_l, i_l(\theta_l)) \text{ with } \theta_0 > \theta_l$$

and $i_0(\theta_0) = t(\theta_0)$ as well as $i_l(\theta_l) = t(\theta_l)$ as there must be a time $\theta_l < T$ up to which x^{EAF} sends more flow to the sink than x.

There exists at least one parallel component in G whose end terminal lies in P for two points of time from $\{\theta_0, \dots, \theta_l\}$ as otherwise i_l cannot be equal to $i_0 = t$. Consider an inclusionwise minimal parallel component $P[i_\nu, i_\mu]$ with this property and let $\theta_{\mu_1}, \theta_{\mu_2} \in \{\theta_0, \dots, \theta_l\}$ be the times at which P visits i_μ.

Due to the minimality of $P[i_\nu, i_\mu]$, the subpath

$$\pi := (i_\mu(\theta_{\mu_1}), a_{\mu_1+1}, \dots, a_\nu, i_\nu(\theta_\nu), a_{\nu+1}, \dots, a_{\mu_2}, i_\mu(\theta_{\mu_2}))$$

of P decomposes into $\pi_1 = (i_\mu(\theta_{\mu_1}), a_{\mu_1+1}, \dots, a_\nu, i_\nu(\theta_\nu))$ using only arcs in $A_T^-(x)$ and $\pi_2 = (i_\nu(\theta_\nu), a_{\nu+1}, \dots, a_{\mu_2}, i_\mu(\theta_{\mu_2}))$ using arcs from $A_T^+(x)$.

Due to the greedy choice of paths in Algorithm 2, no path $\tilde{\pi}$ starting in $i_\nu(\theta_\nu)$ and using only arcs from A_T^+ can reach $i_\mu(\tilde{\theta})$ with $\theta_\nu + \tau(\tilde{\pi}) = \tilde{\theta} < \theta_{\mu_1}$ as otherwise Algorithm 2 would have used this cheaper path $\tilde{\pi}$. Recalling that $\tau(\pi_1) < 0$, we conclude $\tau(\pi_2) \geq -\tau(\pi_1)$. This implies that $\tau(\pi) = \tau(\pi_1) + \tau(\pi_2) \geq 0$.

Iteratively using this argument yields a contradiction to $\theta_l < \theta_0$ and hence x is an earliest arrival flow.

4 Conclusion

Based on an algorithm by Bein et al. [3] we established a greedy algorithm that computes a maximal dynamic flow with time horizon T in a discrete time setting. The algorithm does not only run in polynomial time but additionally yields an earliest arrival flow. In contrast to more general graph topologies this additionally proves the existence of a temporally repeated flow on series-parallel graphs which has the earliest arrival property. To the best of our knowledge this is the first exact polynomial time algorithm to find an earliest arrival flow.

Bibliography

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms and Applications*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1993.

[2] N. Baumann and M. Skutella. Solving evacuation problems effeciently - earliest arrival flows with multiple sources. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 399–410, 2006.

[3] W. W. Bein, P. Brucker, and Tamir A. Minimum cost flow algorithms for series-parallel networks. *Discrete Applied Mathematics*, 10:117–124, 1985.

[4] L. Fleischer and M. Skutella. Quickest flows over time. *SIAM Journal on Computing*, 36(6):1600–1630, 2007.

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

[6] L. R. Ford, Jr. and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

[7] D. Gale. Transient flows in networks. *Michigan Mathematical Journal*, 6(1):59–63, 1959.

[8] A. Hall, S. Hippler, and M. Skutella. Multicommodity flows over time: Efficient algorithms and complexity. *Theoretical Computer Science*, 379(3):387–404, 2007.

[9] H. W. Hamacher and S. A. Tjandra. Mathematical modeling of evacuation problems: State of the art. In M. Schreckenberger and S.D. Sharma, editors, *Pedestrian and Evacuation Dynamics*, pages 227–266. Springer, 2002.

[10] B. Hoppe and E. Tardos. Polynomial time algorithms for some evacuation problems. In *Proceedings of 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 433–441, 1994.

[11] B. Klinz and G. J. Woeginger. Minmum-cost dynamic flows: The series-parallel case. *Networks*, 43(3):153–162, 2004.

[12] E. Minieka. Maximal, lexicographic, and dynamic network flows. *Operations Research*, 21(2):517–527, 1973.

[13] M. Steiner. A survey on earliest arrival flows and a study of the series-parallel case. Diplomarbeit, University of Kaiserslautern, 2009.

[14] S. A. Tjandra. *Dynamic Network Optimization with Application to the Evacuation Problem.* PhD thesis, University of Kaiserslautern, 2003.

[15] W. L. Wilkinson. An algorithm for universal maximal dynamic flows in a network. *Operations Research*, 19:1602–1612, 1971.