On the Use of Network Flow Techniques for Assigning Evacuees to Exits.*

Daniel Dressler[2], Martin Groß[2], Jan-Philipp Kappmeier[2], Timon Kelter[1], Joscha Kulbatzki[1], Daniel Plümpe[1], Gordon Schlechter[1], Melanie Schmidt[1], Martin Skutella[2], and Sylvie Temme[1]

[1] TU Dortmund, Dep. of Computer Science, Germany,
icem09@zet-evakuierung.de
[2] TU Berlin, Institute of Mathematics, Germany,
{dressler,kappmeie,gross,skutella}@math.TU-Berlin.DE

Abstract. We apply network flow techniques to find good exit selections for evacuees in an emergency evacuation. More precisely, we present two algorithms for computing exit distributions using both classical flows and flows over time which are well known from combinatorial optimization. The performance of these new proposals is compared to a simple shortest path approach and to a best response dynamics approach by using a cellular automaton model.

1 Introduction

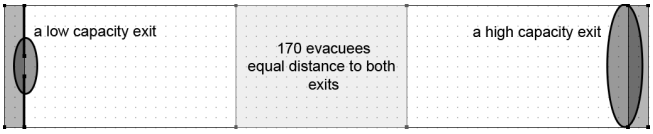
Emergency evacuations are a very important and complex field that has been heavily discussed lately. Still, there are a lot of open questions such as how to model human behavior in dangerous situations, how to design a building in a manner that supports fast evacuations or how to optimize evacuation strategies. We make a contribution to the latter question.

A central step towards developing a good evacuation plan for a given building is to find an appropriate assignment of each evacuee to the exit through which he shall leave the building. In the following we call this an *exit assignment*. The intuitive idea would be sending everybody to his nearest exit. As depicted in Fig. 1(a), this can lead to significantly bad results even if the structure of the building is very simple. More advanced techniques, based on approaches from game theory, have been presented in [5] and lead to better results. We present a completely new approach that uses network flow algorithms, well known from combinatorial optimization. In particular, we analyze the possibilities to use *earliest arrival flows* that provide an accurate model of evacuations because they consider their temporal aspect. Additionally, we investigate an approach using classical static network flows, namely *maximum flows* and *minimum cost flows*.

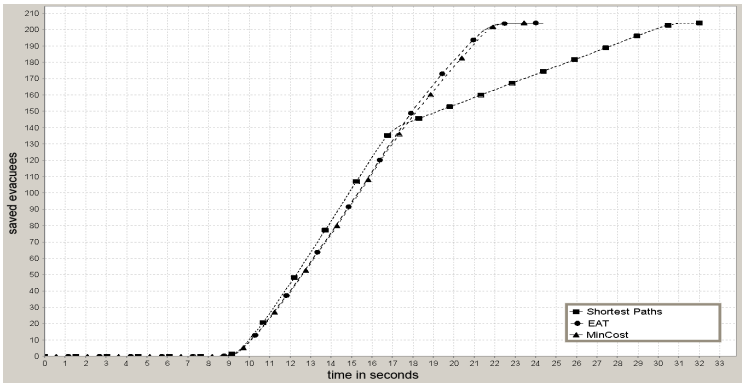
*This work was supported by DFG Research Center Matheon "Mathematics for key technologies" in Berlin.

We evaluate the new approaches with a cellular automaton based simulation. The computational results show a clear advantage of the new techniques. Moreover it turns out that they are practically fast.

All tests are made using the ZET evacuation tool[3] that implements all approaches and the cellular automaton.



(a) A very simple example building.



(b) The computational results: The diagram depicts the number of safe evacuees for each point in time. The curve of the shortest path approach flattens after 17 seconds.

Fig. 1. For about half of the evacuees the narrow exit is nearer than the wide exit so that a shortest path approach will send them there and a jam emerges.

2 Network Flows

We begin by introducing the basic definitions and concepts of network flows required by the exit assignment calculations presented in the sections below. Note that the following definitions are tailored to our purposes to some extent as flow representing evacuees has additional requirements not shared by more general settings.

2.1 Basic Definitions

A *directed graph* $G = (V, A)$ consists of a set of nodes V and a set of arcs $A \subseteq V \times V \setminus \{(v, v) \mid v \in V\}$. For a directed graph $G = (V, A)$ and a node

[3] www.zet-evakuierung.de

$v \in V$, $\delta^+(v)$ and $\delta^-(v)$ denote the set of outgoing and ingoing arcs of node v, respectively. A *network* consists of a directed graph $G = (V, A)$ and a capacity function $u : A \cup V \to \mathbb{N}$ and either a set of sources $S^+ \subseteq V$ and a set of sinks $S^- \subseteq V$ or a balance function $b : V \to \mathbb{Z}$. If a balance function is used, we refer to nodes v with $b(v) > 0$ as sources and to nodes v with $b(v) < 0$ as sinks. For a node v we call $|b(v)|$ its supply if $b(v) > 0$ and its demand if $b(v) < 0$. A *dynamic network* is a network with a transit time function $\tau : A \to \mathbb{N}$ and a time horizon T.

A static flow x assigns a flow value $x(e)$ to each edge $a \in A$. For reasons of brevity, we define the following notations for a flow function $x : A \to \mathbb{N}_0$:

$$x^+(v) := \sum_{e \in \delta^+(v)} x(e) \qquad x^-(v) := \sum_{e \in \delta^-(v)} x(e) \qquad \forall v \in V$$

A dynamic flow assigns a flow value $x(e, t)$ to each arc $a \in A$ at each time $t \in \mathbb{N}_0$. Likewise, we define for a dynamic flow function $x : A \times \mathbb{N}_0 \to \mathbb{N}_0$:

$$x^+(v, t) := \sum_{e \in \delta^+(v)} \sum_{\theta=0}^{t} x(e, \theta) \quad x^-(v, t) := \sum_{e \in \delta^-(v)} \sum_{\theta=0}^{t-\tau(a)} x(e, \theta) \quad \forall v \in V$$

A *flow* on a network (G, u, S^+, S^-) with sources and sinks is a function $x : A \to \mathbb{N}_0$ that satisfies the capacity constraints $x(a) \leq u(a)$ for all $a \in A$ as well as the flow conservation constraints $x^+(v) - x^-(v) = 0$ for all $v \in V \setminus (S^+ \cup S^-)$. A *transshipment* on a network (G, u, b) with a balance function is a function $x : A \to \mathbb{N}_0$ fulfilling the capacity constraints $x(a) \leq u(a)$ for all $a \in A$ and the balance constraints $x^+(v) - x^-(v) = b(v)$ for all $v \in V$.

A *dynamic flow* or *flow over time* on a dynamic network $(G, u, \tau, S^+, S^-, T)$ with sources and sinks is a function $x : A \times \mathbb{N}_0 \to \mathbb{N}_0$ that satisfies

- the time horizon constraints $x(a, t) = 0$ for all $a \in A$ and all $t \geq T - \tau(a)$,
- the edge capacity constraints $x(a, t) \leq u(a)$ for all $a \in A$ and all $t \in \mathbb{N}_0$,
- the node capacity constraints $x^-(v, t) - x^+(v, t) \leq u(v)$ for all $v \in V$, $t \in \mathbb{N}_0$,
- the flow conservation constraints $x^+(v, t) \leq x^-(v, t)$ for all $v \in V \setminus S^+$, $t \in \mathbb{N}_0$.

The value $x(a, t)$ assigned to an edge a at time t can be interpreted as the amount of flow entering the edge a at time t. Note that we allow storage in nodes as long as the capacity of the node is not exceeded.

A *dynamic transshipment* or *transshipment over time* on a dynamic network (G, u, τ, b, T) with a balance function is a function $x : A \times \mathbb{N}_0 \to \mathbb{N}_0$ satisfying the time horizon and the edge capacity constraints as specified for flows over time. Furthermore, it satisfies

- the node capacity constraints $\max\{b(v), 0\} + x^-(v, t) - x^+(v, t) \leq u(v)$ for all $v \in V$ and all $t \in \mathbb{N}_0$,
- the flow conservation constraints $x^+(v, t) - x^-(v, t) \leq \max\{b(v), 0\}$ for all $v \in V$ and all $t \in N_0$,
- the balance constraints $x^+(v, T) - x^-(v, T) = b(v)$ for all $v \in V$.

2.2 Flow Problems with or without time dimension

In this section we will consider several problems based on flows and transshipments (over time). We will begin with the simpler flow and transshipment problems that do not use transit times.

Maximum Flow The *maximum flow problem* proposed by Ford and Fulkerson [6] consists of finding a flow x of maximal value for a given network (G, u, S^+, S^-) with sources and sinks. The value of a flow x is defined as the number of flow units sent from the sources to the sinks, or more formally:

$$value(x) := \sum_{v \in S^-} x^-(v) - \sum_{v \in S^-} x^+(v)$$

We use the preflow-push algorithm with the highest-label selection rule and the global- and gap-heuristics described by Cherkassky and Goldberg [3] to compute a maximal flow.

Note that the maximum flow problem does not take distances or transit times into account. Nonetheless, it is able to provide useful information for our exit assignments: we can use the maximum flow problem to identify bottlenecks and gauge their capacities. For example, we can use the areas where a group of evacuees resides as sources and the safe regions for them as sinks. By computing a maximum flow based on this we are able to locate bottlenecks slowing the evacuation of these people down. Furthermore, it provides us with the number of evacuees that can pass through the bottleneck at the same time, which assists us in determining capacities of escape routes.

Minimum Cost Flow Given a network (G, u, b) and a cost function $c : A \to \mathbb{Z}$, which assigns costs to the edges, the *minimum cost flow problem* consists of finding a transshipment x of minimal cost. A transshipment pays for each flow unit sent through an edge the price determined by the cost function. Therefore, the cost of a transshipment x is defined as follows:

$$cost(x) := \sum_{a \in A} c(a)x(a)$$

Several approaches exist to compute a solution to this problem, e.g. cycle cancelling and successive shortest path based techniques. We found successive shortest path based algorithms to be particularly well suited to our instances due to the fact that the number of evacuees is usually much smaller than the number of nodes and edges in the network, which favors the successive shortest path based techniques.

If we consider the transit times of our building networks as costs we can use minimum cost flow problems to incorporate the transit times in our flow calculations without having time as an explicit dimension in our calculations. On the one side this makes our calculations significantly easier but on the other

side it is a rather large idealization. Still, in conjunction with the maximum flow problem, the minimum cost flow problem is able to provide interesting exit assignments: by estimating the capacities of the building's exits we can derive demands for each exit and use these to compute a minimum cost flow whose result can then be used to generate an exit assignment, as we will see in the next section.

Until now, we have only considered flow problems without an explicit time dimension. In the following we will abandon this restriction and examine flow over time problems that have an explicit time dimension. These problems are typically harder to solve than their counterparts without a time component, but they are able to model evacuation scenarios much more closely.

Maximum Flow Over Time Problem The *maximum flow over time problem* consists of finding a flow over time x of maximal value for a given dynamic network $(G, u, \tau, S^+, S^-, T)$ with sources and sinks. The value of a flow over time x is defined as the number of flow units sent from the sources to the sinks during the specified time interval, or more formally:

$$value(x) := \sum_{v \in S^-} x^-(v, T) - \sum_{v \in S^-} x^+(v, T)$$

We do not use this problem directly, though.

Quickest Transshipment The *quickest transshipment problem* asks for the minimal time horizon T necessary to find a transshipment that sends all supplies to the demands in a dynamic network (G, u, τ, b, T). Expressed differently, this problem basically asks for the time needed to evacuate a certain number of people from a building. While this is certainly an interesting question, the next problem leads to an even more interesting one.

Earliest Arrival Transshipment The *earliest arrival transshipment problem* consists of finding a transshipment over time for a given dynamic network with balances (G, u, τ, b, T) that has the earliest arrival property, which is defined as follows: for every time $t \in \mathbb{N}_0$ the value of the flow units sent from the sources to the sinks (as defined by the balance function) in the interval $[0, t]$ is maximal. The value of the flow units sent in the interval $[0, t]$ is defined analogously to the maximum flow over time problem:

$$value(x, t) := \sum_{v \in S^-} x^-(v, t) - \sum_{v \in S^-} x^+(v, t)$$

In order to solve this problem we use the algorithm proposed by Tjandra [9]. The earliest arrival property guarantees that we have evacuated the maximal amount of people at every point in time. This is especially helpful if the amount of time available for the evacuation is not known — regardless of the amount of time we actually have, the earliest arrival property guarantees that we have saved as many people as possible.

3 Our Approach

Our primary goal in this section is to compute a mapping of evacuees to exits (an *exit assignment*) that minimizes the simulated evacuation time. We approach this problem by reducing the building to a network and apply results from network flow theory.

3.1 Reducing a Building to a Network Model

To reduce the building to a network we essentially partition the building into almost-square rectangles and treat every rectangle as a node. We use transit times and capacities that are proportional to the euclidian distance and to the shared borders of the rectangles, respectively.

By computing a transshipment on this network we can now map the supplies on the sources to the sinks, i.e. we can find a mapping of evacuees to exits by transfering the results of an arbitrary (static or dynamic) transshipment algorithm to an exit assignment. This improves on the intuitive approach that assigns each evacuee to his nearest exit.

3.2 Exit Assignments by Maximum Flows & Minimum Cost Flows

Our first approach transfers the results from a *minimum cost transshipment*. We first calculate a *maximum flow* for every sink in the network to estimate the capacities of the sinks. The network for the following minimum cost computation uses these estimated capacities for the sinks and has infinite capacity on all edges. More precisely, our procedure is as follows: Let $(G = (V, A), u, \tau, b)$ be a network, with S^+ and S^- denoting the sources and sinks, respectively, as defined by the balance function. We proceed in five steps:

1. Calculate the shortest path with respect to transit times between each source-sink pair $(s, t) \in S^+ \times S^-$ in G, i.e. look for paths with minimal transit time. This can e.g. be achieved by using Dijkstra's algorithm [4]. We denote the distance with respect to transit times between s and t by $dist(s, t)$.
2. Estimate the capacity $capacity(t)$ of each exit $t \in S^-$ by calculating a maximum flow from all sources to t. As noted above, we recommend the preflow-push algorithm with the highest-label selection rule and the global- and gap-heuristics for this purpose.
3. Create a complete bipartite graph $G' := (V', A')$ using S^+ and S^- as first and second subset of the partition, respectively:

   $$V' := S^+ \cup S^- \qquad A' := \left\{ (s, t) \mid s \in S^+, t \in S^- \right\}$$

   Add costs $c' : A' \to \mathbb{N}_0$, capacities $u' : A' \to \mathbb{N}_0$ and a balance function $b' : V' \to \mathbb{Z}$ to G':

   $$c'(a) := distance(s, t) \quad u'(a) := \infty \quad \forall a = (s, t) \in A'$$

   $$b'(v) := \begin{cases} b(v), & \text{if } v \in S^+ \\ \frac{capacity(v)}{\sum_{w \in S^-} capacity(w)} \left(\sum_{w \in S^+} b(w) \right), & \text{else} \end{cases}$$

4. Calculate a minimum cost flow on G'. To do so, we suggest using a successive shortest path based technique. We use a slightly relaxed minimum cost flow problem to achieve better results: we treat the demand of a node as an upper bound for the amount of flow sent to it. Thereby we can impose a limit to the number of people evacuated through a specific exit without imposing too rigid constraints.
5. Extract the exit assignment from the computed transshipment.

3.3 Exit Assignments by Earliest Arrival Transshipments

Our second approach uses an earliest arrival transshipment to compute an exit distribution. Contrary to the minimum cost approach, we do not place demands on the sinks: Instead, we add a super sink to our network and connect all sinks to it using edges with infinite capacity and zero transit time. Finally, we set the demand of the super sink to the number of all evacuees. Since the earliest arrival transshipment problem uses capacities, transit times and has a concept of time we can use the network instances we are deriving from a building directly as an earliest arrival transshipment problem.

4 Computational Results

In this section we discuss the performance of the different approaches to calculate exit assignments. To gain computational results we use a simulation that computes the behavior of the evacuees when they get a certain exit assignment as input. The following subsections describe this simulation in detail and discuss the quality of the solutions computed by our network flow based approaches.

4.1 Cellular Automaton Model

In order to evaluate our new approaches we use a *cellular automaton* to simulate the behavior of the evacuees. This approach is very efficient and can produce a lot of effects of pedestrian dynamics with only local interactions of the simulated persons. The model is very similar to cellular automatons presented in [2], [7]. We implemented a very simple model, so that no special simulation features can influence the result.

Discretization of the Building The cellular automaton basically consists of a matrix of squared cells with the dimenson 40cm $\times$ 40cm. Each cell represents room that can be occupied by at most one evacuee thus leading to a maximum density of 6.25 persons per square meter which is also used by [7] and seems to be quite practical in most cases according to [10]. The building is rasterized so that it can be represented by the cells and therefore the model is discrete in space.

The cells are of different types: beside general cells an *exit cell* represents an evacuation area which is a safe place and should be reached from the starting position. Other cells can be used to simulate obstacles blocking the evacuees.

Movement The cellular automaton is a $v_{max} = 1$ model, that means it is discrete in time and every evacuee can move to one of the 8 neighbor cells in each step. Different walking speeds are realized by compulsory breaks after each step, the evacuees also have to wait to compensate longer covered distances in the case they move diagonally.

The evacuees use a potential assigned to the cells to decide in which direction they move, as explained in [7]: the cells with the lowest potential are the exit cells and the potential is increasing if the distance is larger. The evacuees always try to reach cells with a lower potential value and thus use shortest paths implicitly. It is decided randomly with respect to the potential difference to which neigbored cell an evacuee moves.

Each cell has a potential for every reachable block of exit cells, so an evacuee can have a special exit assigned that it wants to reach. Thus the evacuees do not necessary follow the shortest path to *any* exit, but the shortest path to a specified exit.

During the simulation the persons do not move simultaneously, as it would create some problems with evacuees wanting to use the same cell, but one after the other. In order not to favor any evacuee, the order in which the persons move is randomized in each step of the cellular automaton.

A special case occurs, if two persons want to switch positions. The simulation algorithm can detect such situations and allows the two evacuees to swap positions. As the space for the swap is very limited, the swap will lead to a longer waiting period before the next move can happen.

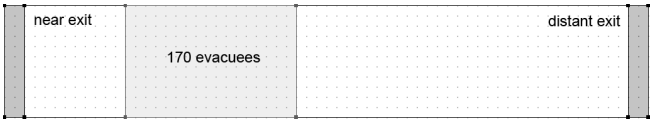
Simulation Algorithm The simulation runs a set of local rules for each person in every step. The set of rules includes rules for finding the target cell and to remove persons from the simulation if they have reached an exit cell. This design contributes to the modularity of our cellular automaton because single rules can easily be replaced or extended if necessary.

4.2 Results

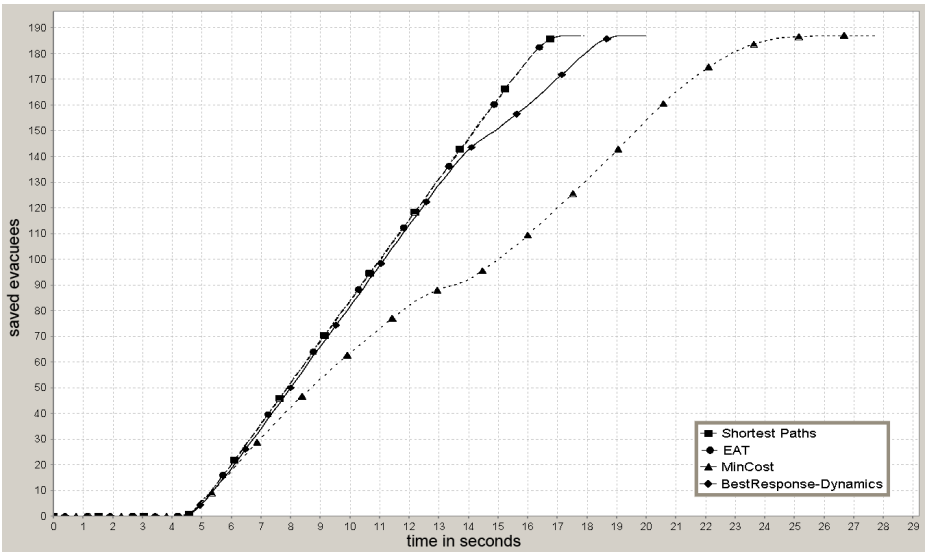
In this section we briefly discuss the advantages and disadvantages of the proposed approaches. As we have already seen in the introduction, the main disadvantage of using the nearest exit for all evacuees is the fact that exit capacities are ignored. The minimum cost approach does not only consider the exits but also computes the width of the bottleneck when reaching an exit from a particular starting point. Thus it can not only handle examples as described in Fig. 1(a) but also buildings that include structures such as small staircases.

However, the minimum cost approach does not care about the distances between evacuees and exits and thus may fail if there are wide exits very far away. It also struggles if a lot of exits share the same bottleneck because it does not take into account that all people that want to go to one of these exits may meet while passing the bottleneck.

The earliest arrival approach uses an optimal flow over time and thus does not suffer from these problems. By taking the temporal aspect into account it



(a) Exits in different distances: The exit on the left is much closer to the evacuees.

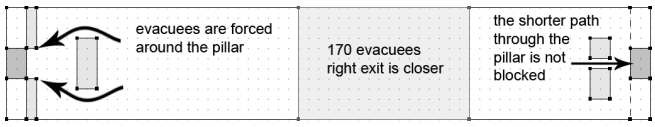


(b) Computational results: The min cost approach loses.

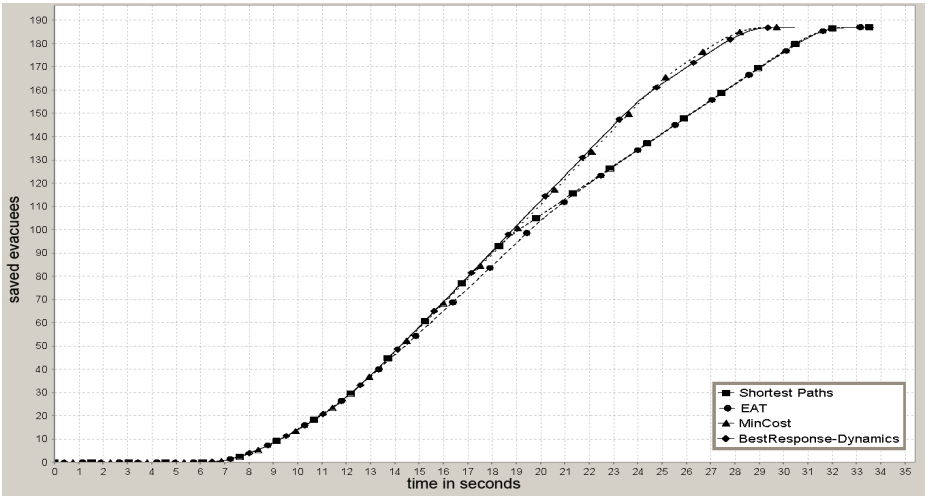
Fig. 2. This example demonstrates the influence of distances on the minimum cost approach. As both exits have the same capacity, the min cost approach sends about half of the people to the right exit although they could reach the left one in significantly shorter time.

detects bottlenecks and exactly balances between the capacity and the distance of an exit. Figure 4 and 2 show the superior performance of the earliest arrival approach in the situations described as critical for the minimum cost approach.

In some sense, the exit assignment computed by the earliest arrival approach is even optimal: if all people in the simulation (and in reality, too) would behave exactly as the earliest arrival transshipment assumes then the evacuation would be as fast as possible. Anyway, the only information the agents in our simulation receive is the exit they should take. In the cellular automaton model (and probably also in reality) people try to get to their desired exit as fast as possible and choose the shortest path to get there. If the earliest arrival transshipment has decided that a lot of people should go to an exit reachable by different ways of different length, all people will take the shortest one. That can take significantly longer than the earliest arrival transshipment expected. Figure 3 shows that the minimum cost approach should be preferred for situations like this.



(a) Different positions of pillars.



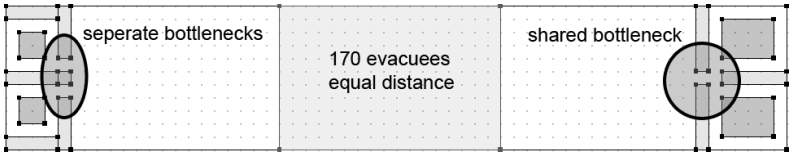
(b) Computational results: The earliest arrival transshipment counts too much on the foresight of the evacuees.

Fig. 3. Evacuees going to the right exit will jam at the bottleneck between the pillars because they do not take the longer but more clever way around the pillars. The earliest arrival approach does not take this into accout and does not perform as well as before.

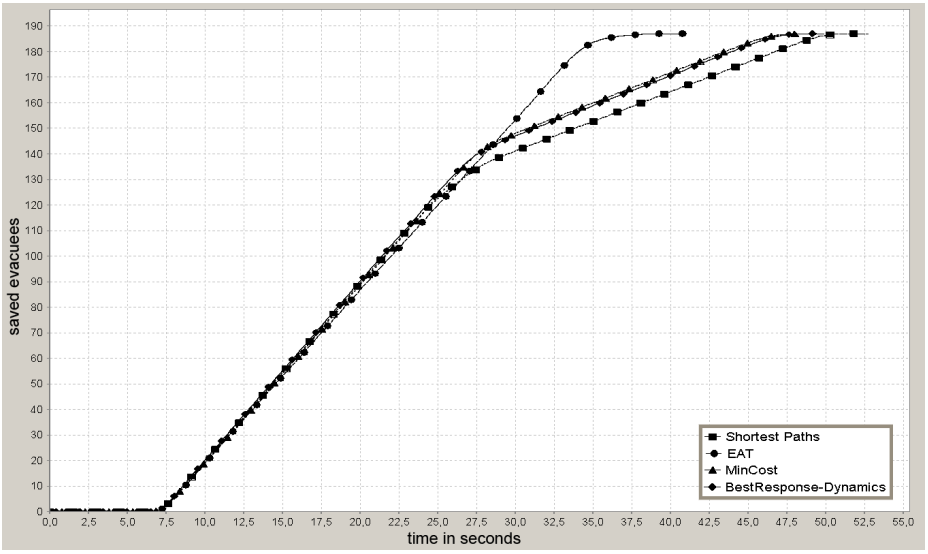
5 Discussion

We have seen that network flows greatly help to compute good exit assignments but the approaches are still in an early state of development and thus have some disadvantages. To reach the best results, one should apply all approaches and choose the best exit assignment by testing all assignments in a simulation. The gained exit assignments can also be used as start distributions for iterative heuristics like the best response dynamic approach to decrease the number of necessary iterations. Another idea to improve the performance of existing approaches by using network flow techniques is to include the capacity calculation described in section 3.2 to gain better approximations of the real capacity of exits. In our tests with the best response dynamics approach this significantly improved the results.

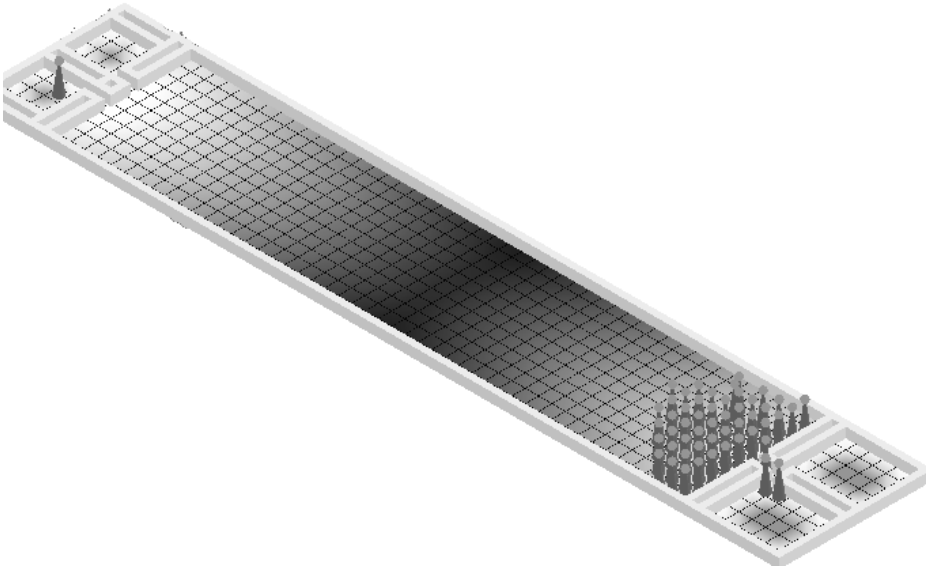
Our research on the use of network flows to set up evacuation plans is far from complete. It is a very interesting question whether our approaches could be combined to reduce the disadvantages of each. Furthermore, the approach



(a) Two exits share the same bottleneck.



(b) Computational results: The diagram depicts the number of safe evacuees for every point in time.



(c) A screenshot of the jam that emerges at the shared bottleneck when using the shortest path approach.

Fig. 4. A building with a shared bottleneck on the right exit: Only the earliest arrival approach has no problems with the shared bottleneck.

using earliest arrival flows could be improved by restricting the algorithm from choosing unlikely routes for evacuees.

Even more interesting is the question whether there is a more direct way to derive evacuation plans from network flows than to calculate exit assignments and use them to develop a good plan. A big step into this direction is the computation of personal evacuation routes, i.e. of a detailed plan telling exactly which way each person should take in a perfect evacuation. The latter can be done by computing an earliest arrival transshipment and finding a mapping of flow units to the evacuees. However, as people usually do not behave perfectly, additional ideas are needed to fully use the strength of network flow computations. A new idea which tries to respect human behavior is the use of Nash Flows over Time [8].

References

1. Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall Inc., 1993.
2. Carsten Burstedde, Kai Klauck, Andreas Schadschneider, and Johannes Zittartz. Simulation of pedestrian dynamics using a 2-dimensional cellular automaton. *Physica A*, pages 507–525, February 2001.
3. Boris V. Cherkassky and Andrew V. Goldberg. On implementing push-relabel method for the maximum flow problem. *Algorithmica*, 19:390–410, 1997.
4. Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.
5. Harri Ehtamo, Simo Heliovaara, Simo Hostikka, and Timo Korhonen. Modeling evacuees' exit selection with best response dynamics. In *Intl. Conference on Pedestrian and Evacuation Dynamics*, volume 4, February 2008.
6. Lester R. Jr. Ford and Delbert. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.
7. Hubert L. Klüpfel. *A Cellular Automaton Model for Crowd Movement and Egress Simulation*. PhD thesis, Faculty of natural sciences, University Duisburg-Essen, Germany, 2003.
8. Ronald Koch and Martin Skutella. Nash equilibria and the price of anarchy for flows over time. In Marios Mavronicolas, editor, *Proceedings of the 2nd International Symposium on Algorithmic Game Theory*, Lecture Notes in Computer Science, Berlin, 2009. Springer. To appear.
9. Stevanus A. Tjandra. *Dynamic Network Optimization with Application to the Evacuation Problem*. Shaker, 1. edition, 2003.
10. Ulrich Weidmann. Transporttechnik der Fussgänger-Transporttechnische Eigenschaften des Fussgängerverkehrs (Literaturstudie). Literature Research 90, Institut für Verkehrsplanung, Transporttechnik, Strassen- und Eisenbahnbau IVT an der ETH Zürich, ETH-Hönggerberg, CH-8093 Zürich, March 1993. In German.