The representation and implementation of time-dependent inundation in large-scale microscopic evacuation simulations

Gregor Lämmel*, Dominik Grether, Kai Nagel

Transport Systems Planning and Transport Telematics, Berlin Institute of Technology, 10587 Berlin, Germany

Abstract

Multi Agent Simulation has increasingly been used for transportation simulation in recent years. With current techniques, it is possible to simulate systems consisting of several million agents. Such Multi Agent Simulations have been applied to whole cities and even large regions. In this paper it is demonstrated how to adapt an existing multi agent transportation simulation framework to large-scale pedestrian evacuation simulation. The underlying flow model simulates the traffic based on a simple queue model where only free speed, bottleneck capacities, and space constraints are taken into account. The queue simulation, albeit simple, captures the most important aspects of evacuations such as the congestion effects of bottlenecks and the time needed to evacuate the endangered area. In the case of an evacuation simulation the network has time dependent attributes. For instance, large-scale inundations or conflagrations do not cover all the endangered area at once. These time dependent attributes are modeled as network change events. Network change events are modifying link parameters at predefined points in time. The simulation framework is demonstrated through a case study for the Indonesian city of Padang, which faces a high risk of being inundated by a tsunami.

Key words: multi agent simulation, large-scale evacuation simulation, time dependent network

*Corresponding author. fon: +49 30 31421376; fax: +49 30 31426269
Email address: laemmel@vsp.tu-berlin.de (Gregor Lämmel)

1. Introduction

Disaster and evacuation planning has become an important topic in science and politics. In principle there are two different situations: evacuation of whole cities or even regions on the one hand, and evacuation of buildings, ships and airplanes or the like on the other hand. The former involves normally the by car, while the latter is rather associated with the evacuation of pedestrians. Corresponding to the two different types of problems, there are two different basic approaches for simulating the traffic flow:

- Methods of dynamic traffic assignment (DTA) have been applied to evacuation simulation on the city or regional scale. Some examples are MITSIM [20], DYNASMART [25] or VISSIM [14]. The DYNASMART approach is based on the analogy between traffic and hydrodynamic characteristics of fluids. On state of the art hardware it is possible to handle even large-scale scenarios with this approach. MITSIM and VISSIM are microscopic, meaning that every vehicle is individually resolved. They allow a more realistic representation of the traffic dynamics, but their computational performance is currently too slow for large-scale simulations. All current implementations of DTA approaches have in common that, they take traffic streams rather than individual travelers or vehicles as input: Their typical input are time-dependent origin-destination matrices, which, in turn, are based on zones. For computational reasons, it is normally not possible to have more than approximately 5 000 zones, meaning that it is not possible to resolve the starting points of the evacuation to a higher level than those 5 000 zones.

- In the area of pedestrian evacuation simulation, there has been considerable research in the last 20 years. A good overview about models and software for pedestrian evacuation simulation can be found in the proceedings of the conference "Pedestrian and Evacuation Dynamics" [33, 6, 7]. Pedestrian evacuation simulations are usually microscopic, using a Cellular Automata (CA) technique [23], discretized differential equations ("molecular dynamics (MD) technique") [16, 15], or movement rules based on random utility modelling [1]. In these models each evacuee is designed as an individual; therefore it is possible to simulate population structures where people have different speeds or ranges, or more complex behavior. Neither of these approaches is applicable for

large-scale scenarios: For a large city with hundreds of thousands inhabitants and an area of several hundred square kilometers a CA-based model would consist of more then 10^9 cells, leading to rather long computing times. For the MD approach, the problem are the sub-second time resolutions that are typically used [5].

One way to deal with large-scale scenarios but to retain persons as individual agents is to use a model with deliberately large time steps and to computationally concentrate on those areas (links) where the pedestrian movement actually takes place [10]. Another, even faster approach is based up on a modified queuing model [8, 34]. The queuing model simplifies streets to edges and crossings to nodes; the difference to standard queuing theory is that agents (particles) are not dropped but spill back, causing congestion. This graph-oriented model is defined by lengths/widths, free speed and flow capacity of the edges. This simplification leads to a major speedup of the simulation while keeping results realistic, and it is the approach used in this paper.

Once the pedestrian movement model is selected, it is necessary to define the evacuation directions. For more complex geometries, this is no longer a single movement towards one or two exits, but may involve rather complex movments in a building or in a street network. The arguably simplest solution is a grid-based potential function where the "uphill direction" leads to the nearest exit [31]. The same can be done using essentially continous spatial variables, at the expense of much larger computing times [18]. Alternatively, routing can be done along graphs [13, 11], which is a much faster technique when the abstraction to a graph is possible.

In the case of an evacuation simulation the network has time dependent attributes. For instance, large-scale inundations or conflagrations do not cover all the endangered area at once. One solution is to model this as a time dependent network. Time dependent networks have been applied to evacuation planning [28] and are often modeled as time expanded graphs [21, 32, 24]. In a time expanded graph every node/edge is replicated for every time step $t = 0, 1, \ldots, T$, and additional links are connecting nodes at every time step. So-called time-aggregated graphs [9] omit the explicit time expansion, and rather use a time-dependent look-up of the link cost. Their notation implies T time intervals which apply uniformly to all links; their $O(\log T)$ time complexity of the link cost lookup implies that this time intervals need not to be equally spaced. Yet, in the case of a tsunami inundation, the

structure of the link cost changes is quite different: There is, at least in an abstract interpretation, only a switch from "passable" to "non-passable", but that switch can happen at arbitrary times. With the above techniques, one would either need as many time intervals T as there are such switches, or several switches would need to be combined into one time bin. This paper introduces the following approach:

- The *interface*, containing the calls to the network attributes, in particular link speeds, link capacities, and link widths, is made time-dependent, allowing the implementation of arbitrary time-dependent functions behind the interface. This corresponds to the time-aggregated graph technique.

- The *implementation* presented here uses so-called network change events. The change events are applied to the edges, modifying their attributes (free speed, flow capacity) at arbitrary points in time. A change event is valid until the next change event will be applied.

In this paper we give a description of the simulation framework with a focus of the time dependent aspects. The performing of the simulation framework is demonstrated through a case study on the Indonesian city of Padang that faces a high risk of being inundated by an earthquake-triggered tsunami. This work is part of the numerical last-mile tsunami early warning and evacuation information system project ("Last-Mile–Evacuation") [2, 27].

The remainder of the paper is organized as follows. Section 2 gives an overview of the simulation framework with a detailed description of the time dependent aspects. In Section 3, we describe the evacuation scenario that was chosen to demonstrate the simulation framework. Section 4 presents the results of the simulation with respect of the computing performance and the evacuation results. After a short discussion in Section 5, we conclude our work and give some information about future work.

2. Simulation framework

The simulation framework is based on the MATSim framework for transport simulation [29]. MATSim is implemented in Java. Since its current implementation is focused on simulation of daily motorized traffic, several adaptations were necessary. The key elements are:

- The agent database, where every agent represents one evacuee.

- The simulation network, based on links and nodes.
- The traffic flow simulator, where all the agents' plans are executed.
- The plans generator, which generates an escape plan for every agent.
- There is a mechanism that allows improving the performance of the agents' plans by repeatedly trying to find faster evacuation routes.

2.1. Synthetic population, plans, agent database

MATSim always starts with a synthetic population. A synthetic population is a randomly generated population of individuals which is based as much as possible on existing data such as census data. For evacuation, the synthetic population is the collection of all synthetic individuals that are involved in the evacuation.

Every synthetic individual possesses one or several plans. These plans are "intentions" of the synthetic individuals, to be tested in the traffic flow simulation (described later), and scored afterwards. For evacuation, the plans are evacuation strategies. For example, such a strategy may be to leave the building 5 minutes after a second warning, and follow a predetermined route to safety. The collection of agents together with their plans is sometimes called an agent database.

People can have different positions within the city when a warning occurs. For example, they can be at home or at work. Therefore, also in the evacuation context it makes sense to consider MATSim plans in their more conventional meaning, as a description on what a synthetic traveller intends to do during a normal day. One can then run a regular traffic flow simulation with these plans, stop it at the time of an evacuation warning, and use the positions of all agents at the time of that warning as the initial condition to the evacuation. This will be the subject of future work.

2.2. Time-varying network

The simulation network represents the area that is accessible by the evacuees. In the case of a vehicular evacuation this network consists of all accessible streets. Each street segment defines a link. The parameters of the links are the length, capacity and the free flow speed. For a pedestrian evacuation the links in the simulation network also consist of squares and sidewalks. The flow capacity is given by the width of a link as described in the next section.

A good way of creating the simulation network is by extracting the needed information from satellite imagery [36].

For evacuation simulations the network has two kinds of time-varying aspects. First there are the disaster related aspects (i.e. time dependent blocking of links). And second there is the information about (experienced) link travel times, caused by congestion. The time dependent congestion effects are important for the agent replanning (discussed in Sec. 2.5), while the disaster related aspects are also important for the initial plans generation (see Sec. 2.4) and the traffic flow simulator (see Sec. 2.3). Since the object of this work is to develop an evacuation simulation framework for large-scale scenarios, the time-varying aspects should be handled in an efficient way.

2.2.1. Network change events

The disaster related time-varying aspects are modeled as network change events. A network change event modifies parameters of a link in the network (e.g. free speed or flow capacity). As soon as a link is no longer passable its free speed will be set to zero.

Listing 1 Sample network change event: valid from 03:06 AM, applied to links with id 12487, 12489 and 12491. The change event sets the free speed of the corresponding links to zero.

```
<networkChangeEvent startTime="03:06:00">
            <link refId="12487"/>
            <link refId="12489"/>
            <link refId="12491"/>
            <freespeed type="absolute" value="0.0"/>
 </networkChangeEvent>
```

The network change events are stored independent from the network in a separate XML file. Listing 1 shows a sample network change event. The event manipulates the links 12487, 12489 and 12491 at 03:06 AM by setting the free speed to 0 m/s. The change values are in SI based units.

The network change events file will be loaded at simulation start up and the network change events will be applied to the corresponding links. The attributes of the links that can be influenced by the change events are accessed by time dependent getter methods (e.g. `getFreespeed(double time)`). These getter methods retrieve the value that is valid for the query time. If there

exists a network change event for the given query time, then the value of this network change event will be returned. But, as discussed in Sec. 1, it is not expected that a network change event exists for every link and every time step. If for a given query time no network change event exists, then the value of network change event with the next smaller event time will be returned. The time dependent getter methods give the flexibility to query the attributes in arbitrary chronology.

As a consequence of this flexibility, the retrieval mechanism is not straight forward. Two different approaches have been tested. The first approach that has been tested is to store the network change events in a Java TreeMap. The Java TreeMap implements a red-black tree with a complexity of $O(log\ n)$ for get operations. The second approach that has been tested is to store the network change events in an array in chronological order and use a binary search to find the corresponding entry. The complexity for a binary search is also a $O(log\ n)$. However, the Java TreeMap cannot operate on primitives (double, int) but the primitives have to be converted into objects (Double, Integer). Since Java 5 this conversion is done automatically (*autoboxing*). The *autoboxing* mechanism gives the software developer more flexibility, but produces an overhead that increases the run time. A small benchmark scenario illustrates this issue. For both approaches the retrieval time for network change events was measured. The number of network change events was successively increased. To get a robust result, the time was taken over 10 000 000 queries. The test was performed on a 2.33 GHz CPU. Fig. 1 shows that the TreeMap implementation is about 20% slower than the implementation using arrays and binary search. Even if the share of the overall runtime for the simulation framework is negligible, the array approach was chosen to implement the time dependent network.[1]

2.2.2. Link travel times

MATSim was originally developed for the simulation of vehicular traffic for large cities or even regions. For this kind of simulation a temporal resolution of 15 *min* for the link travel times is used: The link travel times are aggregated in 15 *min* time bins and the travel time values are stored in arrays (one array for each link). For a network consisting of 100 000 links,

[1] In fact, we and others consistently find that in situations with few or no insertions or removals after initialization, the array-based approach is faster.

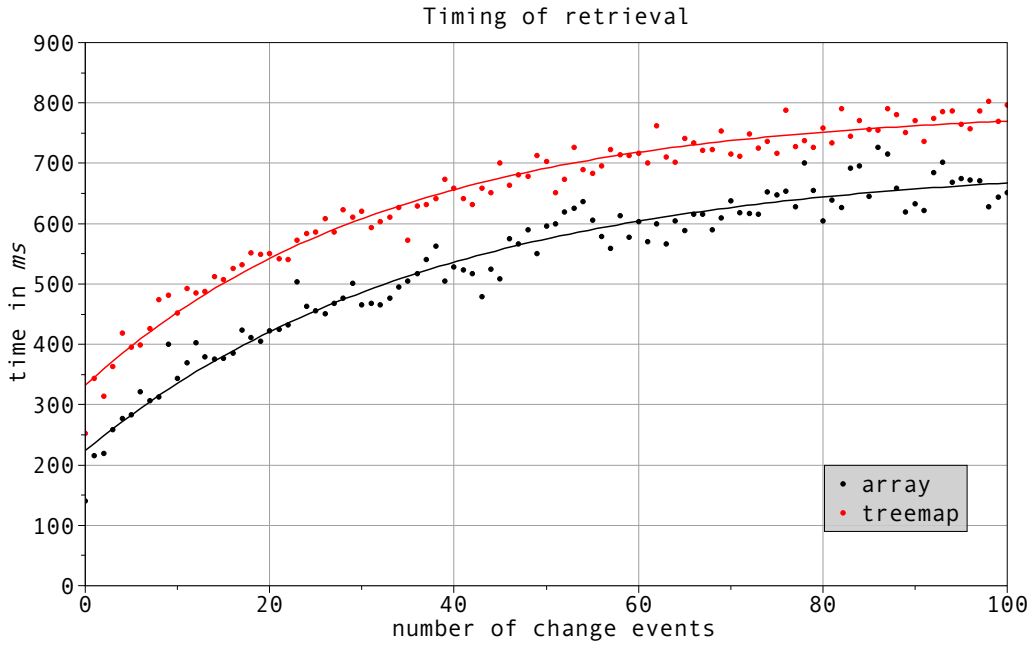

Figure 1: Comparison of the runtime performance for two different implementation of the network change event retrieval. The array based approach using binary search is about 20% faster then the approach using a Java TreeMap. Even if the complexity of both approaches is $O(log\ n)$.

this means a memory consumption of about 75 MB[2] for the link travel time values.

For a pedestrian evacuation simulation with an expected evacuation time of one or two hours, a resolution of 15 *min* is too coarse. However, a finer resolution increases the amount of memory that is needed. To overcome this problem, the array based implementation was replaced by an implementation using Java HashMap. There is a HashMap for each link that stores the link travel times. But only for those travel time bins where at least one agent has entered the link, a travel time value will be stored. This means that if for a given travel time bin no agent enters the corresponding link, then nothing will be stored (and the free speed travel time will be used). Depending on the scenario this can save a lot of memory.

An analytic appraisal of the memory usage or the needed excution time in Java is difficult and depends, besides others, on the virtual machine and its garbage collection mechanism, and on the specific scenario. Therefore it was decided to compare the HashMap based approach with the array based approach through a benchmark scenario. In the benchmark scenario, all link travel times of the simulation described in Sec. 3 have been recorded and aggregated for different travel time bin sizes. Beginning with a travel time bin size of 15 *min* and ending with a travel time bin size of 1 *min*, the memory consumption, the time for storing and aggregating and the time for the retrieval of the link travel times have been measured.

The results of this benchmark test are shown in Fig. 2. At the top there is a diagram comparing the memory usage of both approaches. It is clearly shown that the HashMap based approach consumes considerable less memory than the array based implementation. In particular, at 1-min time resolution the array-based implementation consumes about 650 MB of memory, which, together with the memory requirements of the remainder of the package, would make simulations on today's ordinary desktop computers impossible. The diagram at the bottom compares the runtime for storage and aggregation (write) and for retrieval of the link travel times (read). The time needed for write operations is almost equal for both approaches. For read operations the array approach is faster. Nevertheless, the execution time for a read operation is much smaller than for a write operation. The underlying simulation run

[2]For 24 *h* day there are $4 * 24$ time bins, each time bin holds one double value (64 bit) and there are 100 000 links ($4 * 24 * 100\,000 * 64$ bit $\approx$ 75 MB).

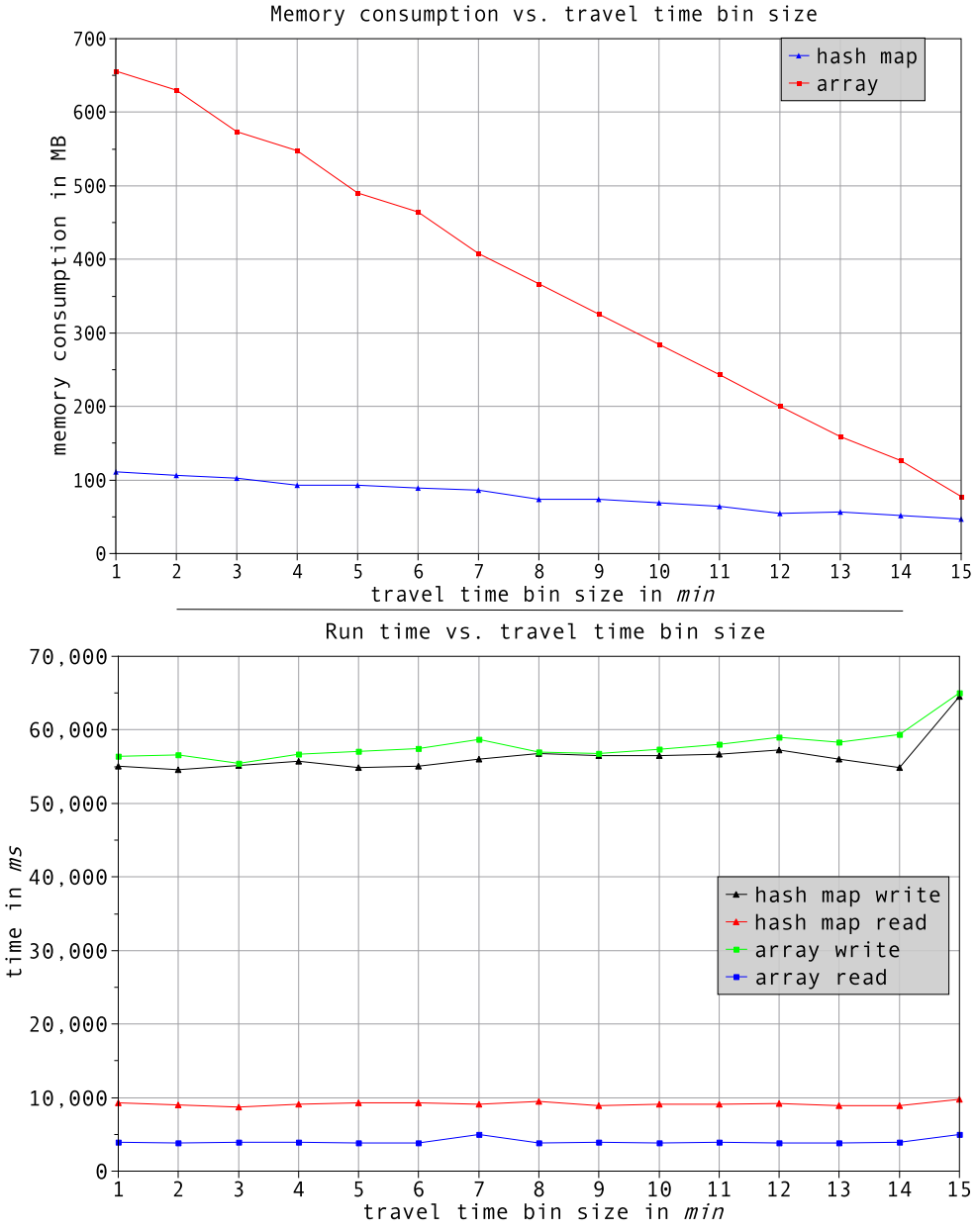

Figure 2: Comparison of two different implementations of the storage and retrieval mechanism for the time dependent link travel times. Top: comparison of the memory consumption. The Java HashMap based approach consumes considerable less memory than an array based implementation. Bottom: comparison of the runtime performance. For write operations both approaches have almost equal execution times. The array based approach is faster for read operations.

simulates the evacuation of about 320 000 agents. On average each agent had to traverse 23.5 links before she reached the safe area. This means there were approximately 75 000 000 different link travel times that had to be aggregated and stored. For the benchmark scenario there were approximately 250 000 000 synthetic read operations[3]. For a real scenario it is expected that the number of write operations is much lower than the number of read operations. That is why the small disadvantage regarding the runtime can be tolerated for the sake of much lower memory consumption.

2.3. Traffic flow simulator

The traffic flow simulation is implemented as a queue simulation, where each street (link) is represented as a FIFO (first-in first-out) queue with three restrictions [8]. First, each agent has to remain for a certain time on the link, corresponding to the free speed travel time. Second, a link flow capacity is defined which limits the outflow from the link. If, in any given time step, that capacity is used up, no more agents can leave the link. Finally, a link storage capacity is defined which limits the number of agents on the link. If it is filled up, no more agents can enter this link. The difference to standard queueing theory is that agents (particles) are not dropped but spill back, causing congestion. An illustration of the queue model is shown in Fig. 3 a). The parameters of the model are:

- Link minimum width w
- Link area A
- Link length l
- Flow capacity $FC = w * C_{max} = w * 1.3\frac{p}{m*s}$
- Free flow speed $v_{max} = 1.66\frac{m}{s}$
- Storage capacity $SC = A * D_{max} = A * 5.4\frac{p}{m^2}$

where C_{max} is the maximum flow capacity per unit width, and D_{max} is the maximum density per unit area. The parameters have been chosen to approximate Weidmann's fundamental diagram [37]. For more details, see [26].

[3]The network for the scenario consists of 16 978 links. For each link the travel time was queried 20 times for 720 different time steps. ($16\,978 * 20 * 720 \approx 250\,000\,000$)

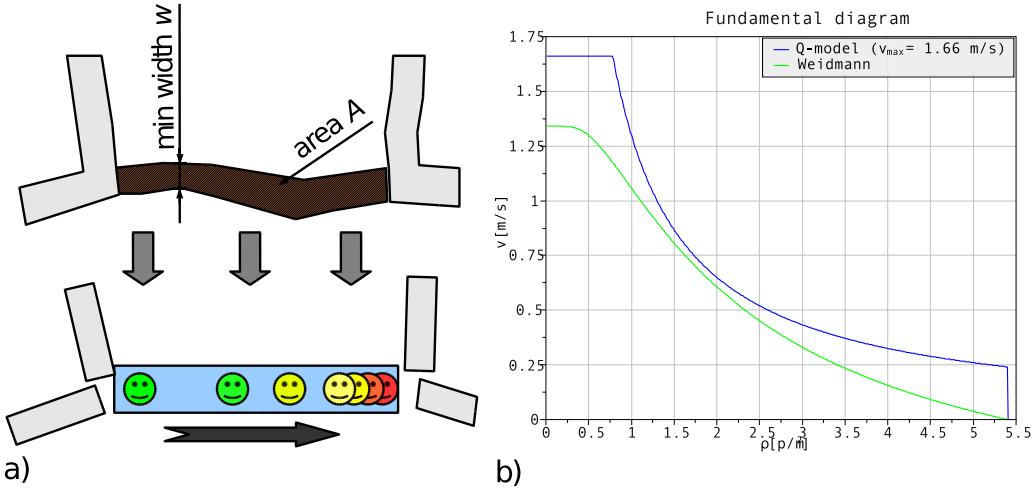

Figure 3: Functioning of the queue model is shown in (a) and its corresponding fundamental diagram in (b).

2.4. Plans generation

Initial plans use the shortest path (according to free speed travel time) out of the evacuation area for all agents. Within the MATSim framework a shortest path router based on Dijkstra's shortest path algorithm [4] has been implemented. This router finds the shortest path in a weighted graph from one node to any other, whereby the actual weights for a link are defined by a time-dependent cost function. Since the city has to be evacuated as fast as possible, the weights represent the (expected) travel time.

There is, however, no particular node as the target of the shortest path calculation, as the evacuees have more than one safe place to run to. Instead, in the underlying domain every node outside the evacuation area is a possible destination for an agent that is looking for an escape route. To resolve this, the standard approach (e.g. [28]) is to extend the network in the following way: All links which lead out of the evacuation area are connected, using virtual links with infinite flow capacity and zero length, to a special evacuation node. Doing so, Dijkstras algorithm will always find the shortest route from any node inside the evacuation area to this evacuation node.

2.5. Agent learning

During the simulation, each evacuee optimizes his/her personal evacuation route to find the fastest escape route. At this point two different routing

solutions are considered: (1) An "shortest path" routing solution, where every evacuee follows the path that would be fastest in an empty network. (2) A "Nash equilibrium" approach, where, via iterations every evacuating person attempts to find a route that is optimal for him-/herself under the given circumstances including congestion. Both approaches can be considered as benchmarks: the first as one where congestion effects are not taken into account in the path choice; the second one as one which might be achieved by appropriate training or guidance while maintaining acceptability in the sense that no person could gain by deviating from this solution. This will be discussed in more detail in Sec. 5.

At the end of every iteration, every agent will score the performed plan. The score of a plan is the negative of its execution time (i.e. of the needed time to evacuate). The scored plans remain in the agents' memory for further executions. Two different learning strategies have been applied for the learning procedure.

- The ReRoute strategy generates new plans with new evacuation routes based on the information of the experienced travel times from the last run. This uses the router described in the previous section, but using time-variant link travel times as link costs. The link travel times are aggregated into 3 *min* time bins.

- The other strategy is called ChangeExpBeta. This strategy decides if the just performed plan should be used again, or if a random plan out of the memory should be selected for the next iteration. The probability to change the selected plan is calculated as:

$$p_{change} = min(1, \alpha * e^{\beta * (s_{other} - s_{current})/2}) \ , \tag{1}$$

  with:

  - α: The probability to change if both plans have the same score
  - β: A sensitivity parameter
  - $s_{\{other,current\}}$: The score of the other/current plan

  In the long run this model is equivalent to the following probabilistic discrete choice model:

$$p_j = \frac{e^{\beta * s_j}}{\sum_i e^{\beta * s_i}} \ ,$$

where p_i is the probability for plan i to be selected and s_i its current score. Eq. 1 thus describes a process which slowly converges towards a logit model probability distribution. This makes convergence to a steady state smoother than when all choices are made in every iteration.

A strategy selector decides for every agent which of the strategies (ReRoute or ChangeExpBeta) will be used. Each strategy is selected with a certain probability. These probabilities are assigned before the simulation starts, but they can be varied during the iterations.

After re-planning every agent has a selected plan that will be executed in the next iteration. Repeating this iteration cycle of learning, the agents' behavior will move towards a Nash equilibrium. If the system were deterministic, then a state where every agent uses a fixed plan that is always a best response to the last iteration would be a fixed point of the iterative dynamics, and at the same time a Nash Equilibrium since no agent would have an incentive to unilaterally deviate. Since, however, the system is stochastic, this statement does not hold, and instead we look heuristically at projections of the system, e.g. the average evacuation time. From experience it is enough to run 100 iterations until the iterative dynamics has reached a steady state. In most (but not all) evacuation situations, the Nash equilibrium leads to a shorter overall evacuation time than when everybody moves to the geographically nearest evacuation point.

3. Scenario

The aim of this work is to find feasible solutions for the evacuation of the city of Padang in the case of a tsunami. There are several aspects that have to be taken into consideration. At first one needs a synthetic population for the city. In this case study it is assumed that all people are at home. The information about the distribution of the population was derived from existing census data [3]. Approximately 320 000 people are living in the endangered area. This means that 320 000 agents have been created for the simulation.

As discussed in Sec. 2.3, the traffic flow simulation is performed on a network representing the walkable area of the city. The street map was extracted from satellite imagery by remote sensing technologies [35] and converted into a graph [27]. The resulting simulation network consists of 6 289 nodes and 16 978 unidirectional links.

Another important aspect is the information about safe places. In future it is planned to identify buildings that are suitable for a vertical evacuation. For the time being we use a simpler approach: All areas with an elevation of more the 10 m above sea level are defined as safe. Fig. 4 shows an image of the city with the safe area. However, based on models of small-scale flooding and inundation dynamics of the tsunami [12] it is not expected that all the area below 10 m will be flooded. Based on these simulations, one also learns that the estimated time between the earthquake and the inundation of the city is about 28 min. The results are backed by the results of large-scale tsunami simulations for the west coast of Sumatra Island [30]. Adding this to the simulation, the agents were made to learn a more risk averse behavior: they are not only trying to reach the safe area as fast as possible, but they also try to increase the distance to the endangered areas. In some places the flooding will reach locations that are more than 2 km away from the shoreline.

The inundation data was provided by [12] as a time series of flooding heights for (x, y)-coordinates with a temporal resolution of 1 min and a spatial resolution of 3 m. The queue model reproduces the inundation as a time dependent network by varying the free speed parameter of the links. The free speed, as discussed in Sec. 2.2, for the not inundated link is 1.66 m/s and the free speed for an inundated link is 0 m/s. This means as soon as a link is flooded its free speed will be set to 0 m/s. The router will, in consequence, try avoid the link at these times. And if, nevertheless, an agent happens to be on this link at such a time, it will remain stuck there forever.

As explained above, we applied two different strategies for learning to the simulation. The ReRoute strategy finds a new evacuation route for an agent, based on the experienced travel times of the former iteration. The ChangeExpBeta strategy implements a discrete-choice model that assigns a plan from the agent's memory with a probability depending on the score of the plan.

In the current setup the probability of being chosen for the ReRoute strategy is 10% and 90% for ChangeExpBeta. This setup gives a fair arrangement between exploration and exploitation. If the probability for ReRoute (i.e. exploration) is too low, then it could be that some promising routes will never be discovered. On the other hand, if the probability for ReRoute is very high, the system tends to fluctuate and will not convert to a steady state. The system would change so fast that the agents would not get a chance to exploit their knowledge about the system.

Figure 4: Satellite imagery of the city shows the safe area (light green) and some preliminary results of the flooding simulation (blue area). Satellite imagery by the German Aerospace Center, Oberpaffenhofen (2007)

4. Results

The simulation run was stopped after 200 iterations of learning. The overall runtime was about 15 hours on a 3 GHz CPU using up to 2 GB of RAM. After 200 iterations of learning, the evacuation time is about 75 min. This is the time that is needed to evacuate all the area with an elevation lower than 10 m. An interesting aspect is the time that is needed to evacuate all the area that is expected to be inundated. Fig. 5 compares the evacuation progress of the proposed routing solutions. The three snapshots on the left side of the figure show the evacuation progress for the "shortest path" solution and the three snapshots on the right side of the figure for the "Nash equilibrium" approach. It is clearly shown that the "shortest path" solution does not leave enough time to evacuate the costal strip. The results from the "Nash equilibrium" approach seem to be more feasible. But not only the evacuation of the coastal strip is much faster, but also the overall evacuation of all the area below 10 m. Fig. 6 shows the evacuation progress for the first and the last iteration. After 200 iterations of learning, the overall evacuation time is about 75 min. This is much better compared to the first iteration, where only 75% of the evacuees manage to escape within 75 min.

5. Discussion

The simulations concentrate on two types of agent behaviors: One where every agent follows the shortest path to the safe area; one where a Nash equilibrium is reached. Both can be considered as benchmarks:

- The first as one where agents are rational about their path choice, but unaware of congestion effects.
- The second as a solution that could be reached by training, assuming that agents follow the training solution also in the real situation.

In panic situations, people tend to be irrational and to display herd behavior [17]. Still, if even the Nash equilibrium solution does not leave enough time, then this would be a strong indicator that major measures would need to be taken to rectify the situation.

It should also be stated that Nash equilibrium and system optimum do not need to coincide – i.e. that solutions even better than the Nash equilibrium might be possible. Such solutions would, however, be unstable in the sense

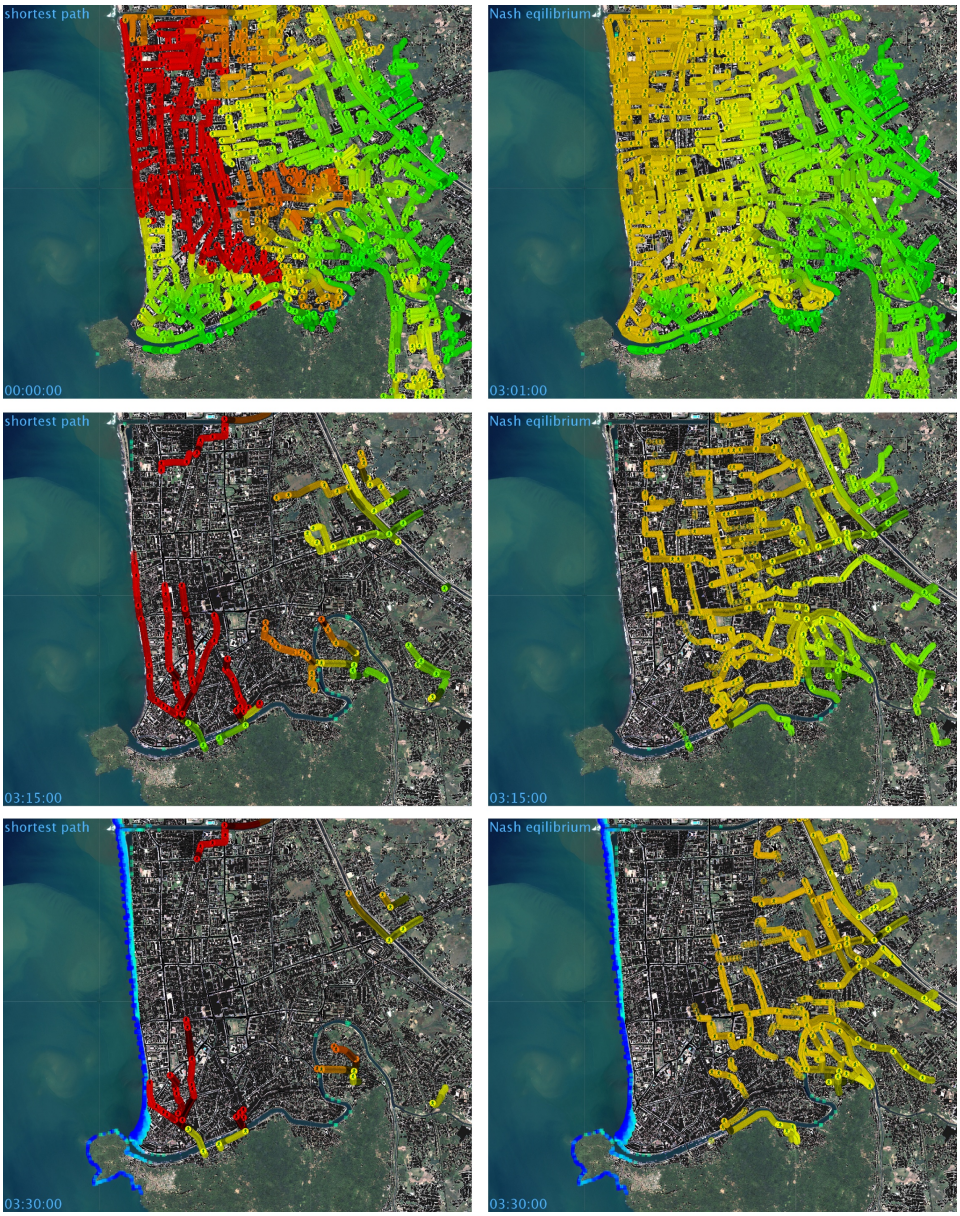

Figure 5: Visualizer snapshots of the evacuation progress. The evacuation starts at 03:00 AM and the snapshots are taken after 1 *min*, 15 *min* and 30 *min*. The three snapshots at the left side shows the evacuation progress for the "shortest path" solution and the three snapshots on the right side the "Nash equilibrium" approach. The agents are colorized with respect to the time they need to evacuate. The evacuation time increases as the color moves from green to yellow to red. Note in particular some highly endangered agents in the shortest path solution due to congestion.

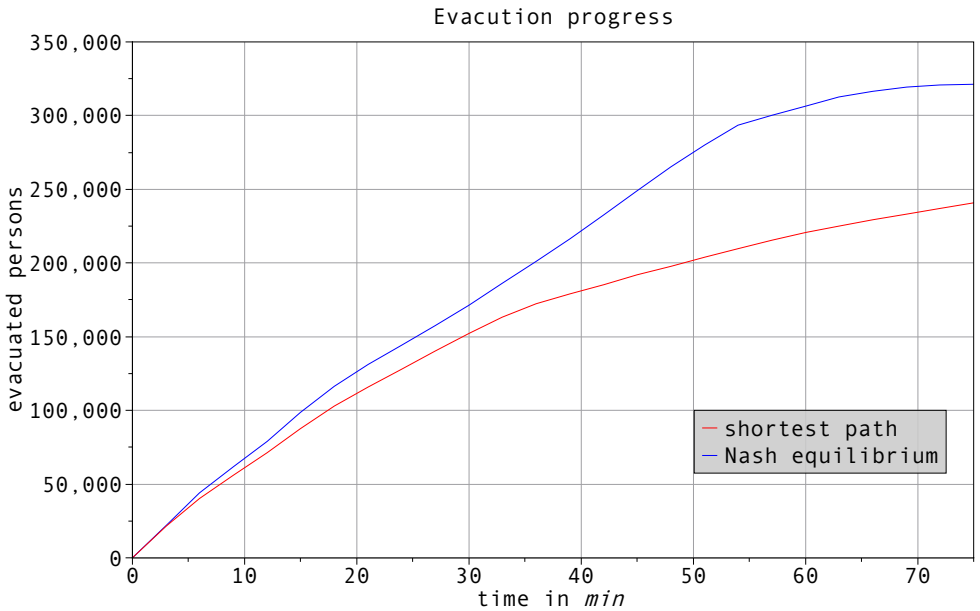

Figure 6: Comparison of the evacuation curves of the first iteration ("shortest path" solution) and the last iteration ("Nash equilibrium" approach). The curves are truncated at 75 min.

that people would have an incentive to deviate. Such solutions seem even more improbable than Nash equilibrium solutions.

Finally, one should mention that MATSim already contains the first hooks towards en-route replanning [19]. This would allow to add situation-based behavior into the simulation.

Another issue concerns the mode choice: The investigation assumes that all evacuation is done by foot while it might be reasonable to assume that some people use cars or cycles, and they might even leave vehicles in the street to continue on foot if progress by vehicle becomes too slow. For the time being, such issues are not considered. The queue model could, to a certain extent, be parameterized to deal with mixed traffic, as long as all modes move with the same speed. Beyond that, one would arguably need to switch to a true two-dimensional model such as [17] or [22]. Such models could still operate on networks [11].

In the current base case it is assumed that all people are at home. Currently we are working on more detailed picture of the population. Based on census data and the results of a survey with 1 000 households, that took place in April/Mai 2008 [2], we are developing a synthetic population with individual daily plans. From this synthetic population it will be possible to derive a model of the population distribution at any time of day. In future work it is also planned to integrate tsunami proof shelters into the simulation framework. Therefore the simulation framework could be extended in a way to find optimal locations for the tsunami proof shelters.

6. Conclusions

We introduced a microscopic pedestrian simulation framework for large-scale evacuations. It is implemented as a Multi Agent Simulation, where every agent tries to optimize its individual evacuation plan in an iterative way. The flooding information is modeled as network change events in the simulation framework, which, when such events are comparatively rare, is a much sparser representation than time-expanded or time-aggregated graphs. The network change events approach could be easily adapted to other scenarios, for example for modeling accidents.

The simulation framework is demonstrated through a case study based on a tsunami evacuation of the Indonesian city of Padang. Despite the underlying behavioral model being quite simple, the simulation gives plausible results regarding the predicted evacuation time and process. The runtime

performance shows that this approach is well suited for large scale scenarios. With state of the art hardware it is no problem to simulate much larger scenarios with over one million agents.

7. Acknowledgments

This project was funded in part by the German Ministry for Education and Research (BMBF), under grants numbers 03G0666E ("last mile") and 03NAPI4 ("Advest"). We would like to thank Hubert Klüpfel for the fruitful discussion about pedestrian evacuation simulations.

References

[1] M. Bierlaire, G. Antonini, and M. Weber. Behavioral dynamics for pedestrians. In K. Axhausen, editor, *Moving through nets: The physical and social dimensions of travel*. Elsevier, 2003.

[2] J. Birkmann, S. Dech, N. Goseberg, H. Klüpfel, G. Lämmel, F. Moder, K. Nagel, M. Oczipka, T. Schlurmann, N. Setiadi, F. Siegert, G. Strunz, and H. Taubenböck. Numerical last-mile tsunami early warning and evacuation information system ("Last-Mile – Evacuation "). In *GEOTECHNOLOGIEN Science Report No. 10: "Early Warning Systems in Earth Management"*, Osnabrück, October 2008.

[3] BPS. *Kecamatan Dalam Angka - Subdistricts in Numbers*. Satistical bureau (BPS) Kota Padang, Padang, 2005.

[4] E. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269 – 271, 1959.

[5] I. Farkas. pedsim source code, accessed 2008. See pedsim.elte.hu.

[6] E. R. Galea, editor. *Pedestrian and Evacuation Dynamics*. Proceedings of the 2nd international conference, London, 2003. CMS Press, University of Greenwich, UK, 2003.

[7] P. Gattermann, N. Waldau, and M. Schreckenberg, editors. *Pedestrian and Evacuation Dynamics*. Proceedings of the 3rd international conference, Vienna. Springer, Berlin, 2006.

[8] C. Gawron. An iterative algorithm to determine the dynamic user equilibrium in a traffic simulation model. *International Journal of Modern Physics C*, 9(3):393–407, 1998.

[9] B. George, S. Kim, and S. Shekhar. Spatio-temporal network databases and routing algotihms: A summary of results. In *Proceedings of International Symposium on Spatial and Temporal Databases (SSTD'07)*, 2007.

[10] C. Gloor, P. Stucki, and K. Nagel. Hybrid techniques for pedestrian simulations. In *Proceedings of Swiss Transport Research Conference (STRC)*, Monte Verita, CH, 2004. See www.strc.ch.

[11] C. Gloor, P. Stucki, and K. Nagel. Hybrid techniques for pedestrian simulations. In *Cellular automata, Proceedings*, number 3305 in Lecture Notes in Computer Science, pages 581–590. Springer, 2004.

[12] N. Goseberg, A. Stahlmann, S. Schimmels, and T. Schlurmann. Highly-resolved numerical modeling of tsunami run-up and inundation scenarios in the city of Padang, West Sumatra. In *Interantional Conference on Coastal Engineering*, Hamburg, 2008. (Poster).

[13] S. Hamacher, H.W. Tjandra. Mathematical modelling of evacuation problems: A state of art. *Berichte des Fraunhofer ITWM*, 24:1–38, 2001.

[14] L. Han and F. Yuan. Evacuation modeling and operations using dynamic traffic assignment and most desirable destination approaches. Paper 05-2401, Transportation Research Board Annual Meeting, Washington, D.C., 2005.

[15] D. Helbing, L. Buzna, A. Johansson, and T. Werner. Self-organized pedestrian crowd dynamics: experiments, simulations and design solutions. *Transportation Science*, 39:1–24, 2005.

[16] D. Helbing, I. Farkas, P. Molnar, and T. Vicsek. Simulation of pedestrian crowds in normal and evacuation situations. In *Pedestrian and evacuation dynamics*, pages 21–58. Springer, Berlin, 2002.

[17] D. Helbing, I. Farkas, and T. Vicsek. Simulating dynamical features of escape panic. *Nature*, 407:487–490, 2000.

[18] S. Hoogendoorn, P. Bovy, and W. Daamen. Microscopic pedestrian wayfinding and dynamic modelling. In Schreckenberg and Sharma [33], pages 123–154.

[19] J. Illenberger, G. Flötteröd, and K. Nagel. Enhancing MATSim with capabilities of within-day re-planning. In *Proceedings of the 2007 IEEE Intelligent Transportation Systems Conference*, pages 94–99, Seattle, WA, Sep 30 – Oct 3 2007.

[20] M. Jha, K. Moore, and B. Pashaie. Emergency evacuation planning with microscopic traffic simulation. Paper 04-2414, Transportation Research Board Annual Meeting, Washington, D.C., 2004.

[21] D. Kaufman and R. Smith. Fastest paths in time-dependent networks for intelligent vehicle-highway systems application. *Journal of Intelligent Transportation Systems*, 1:1–11, 1993.

[22] H. Klüpfel. The simulation of crowd dynamics at very large events – Calibration, empirical data, and validation. In Gattermann et al. [7].

[23] H. Klüpfel, T. Meyer-König, A. Keßel, and M. Schreckenberg. Simulating evacuation processes and comparison to empirical results. In M. Fukui et al, editor, *Traffic and granular flow '01*, pages 449–454. Springer, Berlin Heidelberg New York, 2003.

[24] E. Kohler, K. Langtau, and M. Skutella. Time-expanded graphs for flow-dependent transit times. In *10th Annual European Symposium on Algorithms*, pages 599–611, 2002.

[25] E. Kwon and S. Pitt. Evaluation of emergency evacuation strategies for downtown event traffic using a dynamic network model. Paper 05-2164, Transportation Research Board Annual Meeting, Washington, D.C., 2005.

[26] G. Lämmel, M. Rieser, and K. Nagel. Bottlenecks and congestion in evacuation scenarios: A microscopic evacuation simulation for large-scale disasters. In *5th Workshop on AGENTS IN TRAFFIC AND TRANSPORTATION @ Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Estoril, Portugal, May 2008.

[27] G. Lämmel, M. Rieser, K. Nagel, H. Taubenböck, G. Strunz, N. Goseberg, T. Schlurmann, H. Klüpfel, N. Setiadi, and J. Birkmann. Emergency preparedness in the case of a tsunami – evacuation analysis and traffic optimization for the Indonesian city of Padang. In *Pedestrian and Evacuation Dynamics*, Wuppertal, Germany, 2008.

[28] Q. Lu, B. George, and S. Shekhar. Capacity constrained routing algorithms for evacuation planning: A summary of results. *LNCS*, 3633:291–307, 2005.

[29] MATSIM www page. MultiAgent Transport SIMulation. http://matsim.org/, accessed 2008.

[30] J. McCloskey, A. Antonioli, A. Piatanesi, K. Sieh, S. Steacy, S. Nalbant, M. Cocco, C. Giunchi, J. Huang, and P. Dunlop. Tsunami threat in the indian ocean from a future megathrust earthquake west of Sumatra. *Earth and Planetary Science Letters*, 265:61–81, 2008.

[31] K. Nishinari, A. Kirchner, A. Nazami, and A. Schadschneider. Extended floor field CA model for evacuation dynamics. *IEICE Transactions on Information and Systems*, E87-D(3):726–732, 2004.

[32] S. Pallottino and M. Scutella. Shortest path algorithms in transportation models: Classical and innovative aspects. In P. Marcotte and S. Nguyen, editors, *Equillibrium and Adavanced Transportation Modelling*, pages 245–281. Kluwer, 1998.

[33] M. Schreckenberg and S. D. Sharma, editors. *Pedestrian and Evacation Dynamics*. Proceedings of the 1st international conference, Duisburg, 2001. Springer, 2001.

[34] P. Simon, J. Esser, and K. Nagel. Simple queueing model applied to the city of Portland. *International Journal of Modern Physics*, 10(5):941–960, 1999.

[35] H. Taubenböck, J. Prost, R. Kiefl, A. Roth, F. A. Ismail, G. Strunz, and S. Dech. Risk and vulnerability assessment to tsunami hazard using very high resolution satellite data. In C. Jürgens, editor, *Remote Sensing - New Challenges of High Resolution*, Bochum, 2008. European Association of Remote Sensing Laboratories.

[36] H. Taubenböck and A. Roth. A transferable and stable classification approach in various urban areas and various high resolution sensors. In *Urban Remote Sensing Joint Event*, Paris, 2007.

[37] U. Weidmann. *Transporttechnik der Fussgänger*, volume 90 of *Schriftenreihe des IVT*. Institute for Transport Planning and Systems ETH Zürich, 2 edition, 1993. In German.