2011 23rd IEEE International Conference on Tools with Artificial Intelligence

Real-Time Pedestrian Evacuation Planning During Emergency

Tarik Hadzic, Kenneth N. Brown, Cormac J. Sreenan
Cork Constraint Computation Centre and Mobile and Internet Systems Laboratory
Department of Computer Science, University College Cork
Ireland

***Abstract*—We develop a set of solution techniques for real-time evacuation guidance of pedestrians during emergency, focusing on evacuation from buildings during a fire. We model the problem as an extension of a dynamic network flow by allowing for nodes and edges to expire over time. This captures evacuation situations where the spreading hazard renders parts of the network unavailable. We formally state the problem, analyze its complexity, develop a set of heuristic approaches and compare their performance against a number of most relevant alternative approaches. We experimentally demonstrate that our heuristics outperform the alternatives and are suitable for real-time use even for large networks.**

***Keywords*-evacuation planning; dynamic network flows; heuristics;**

I. Introduction

Evacuation planning involves constructing plans for rapid movement of people from an area affected by emergency, such as a city threatened by a hurricane or a building on fire. However, these plans are traditionally constructed only to *prepare* for emergency, long before more information is available about the actual threat. The same evacuation plans are to be used regardless of the origin of hazard and location of people, which might render them unsafe. However, recent sensor technology allows us to track the location of people and the spread of hazard, while intelligent direction-signalling hardware (visual, audio, tactile) and ubiquitous smart phones make it possible to communicate dynamically changing evacuation directions to evacuees. We therefore suggest computing evacuation plans *during emergency*, optimized specifically for the ongoing situation, and updated as the emergency evolves. An evacuation plan is recomputed as the new sensor input becomes available. The challenge is to ensure that 1) computed plans are of *high safety*, that 2) they can be computed and communicated to evacuees in *real-time* and that 3) they can *scale* to realistic problem sizes. This paper presents a solution approach that satisfies the above requirements.

We model our problem as a dynamic network flow [1] where network topology abstracts the building design: arcs represent corridors while nodes represent rooms, open spaces, doors and junctions. Arc transit times and capacities capture the length and width of the corridors, while node capacities capture the area of corresponding locations. Flow represents the occupants, and the movement of flow from initial nodes to a designated exit node represents the evacuation process. In addition, we model the spread of hazard through the network by estimating hazard arrival time for each node. The problem is to maximize the amount of flow that can reach an exit without exposure to hazard.

Dynamic network flows have been studied for decades as a modeling and reasoning framework for evacuation and transportation problems [1], [2], [3]. However, unlike the agent-based simulation approaches, this framework is unable to incorporate behavioral characteristics of individual evacuees or more subtle geometric interactions during the movement of people. Therefore the simulation-based frameworks are a preferred choice for evaluating the movement of people following a specific evacuation plan. However, the computation-intensive nature of simulation executions and real-time computational requirements in our setting makes it highly unlikely that any such approach would be feasible for computing evacuation plans. Therefore, the flow-based framework offers a reasonable compromise: while it approximates certain aspects of evacuation behavior, it allows for more efficient solution approaches. We could compensate for flow model inaccuracies and uncertainties in the hazard prediction by rapid plan recomputation each time the sensed data deviates from previous predictions.

The main contribution of this paper is development of a family of flow-based heuristics that produce high quality evacuation plans in real-time for large instances. Heuristics repeatedly chose a departure node, a dispatch time, a path to an exit and the number of people to send. The choices are based on a novel concept of *node* and *path priorities*, so that given multiple options, nodes and paths with higher priority are preferred. Experiments demonstrate that our heuristics are superior to alternative approaches. A classical exact approach to dynamic network flows, based on time-expanded graphs, does not scale, taking minutes to compute routes for moderately sized networks. Simple and traditionally used congestion-oblivious heuristics that ignore capacity constraints, although much faster, produce solutions of much lower quality. Our hazard-aware adaptation of the most relevant flow-based heuristic [4] also produces solutions of inferior quality. In comparison, our approach is able to evacuate more than 94% of the numbers evacuated by exact approach, while being able to communicate evacuation instructions in a fraction of a second. Therefore, our heuristics

1082-3409/11 $26.00 © 2011 IEEE
DOI 10.1109/ICTAI.2011.95

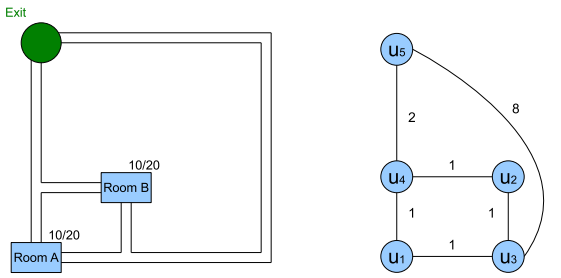
constitute a state-of-the-art solution approach to the problem of real-time evacuation planning.

II. Background and Related Work

Evacuation and transportation problems are normally modeled using *dynamic network flows* [1], an extension of classical network flow theory. The term "dynamic" refers to the fact that flow takes time to traverse the edges and does not imply that the underlying graph is changing. A building (or a road network) is modeled as a *dynamic network*, i.e. a directed graph $G(V, E)$ where nodes $u_i \in V$ represent locations, rooms and junctions, while edges represent corridors and roads. A directed edge between nodes u_i and u_j is denoted as (u_i, u_j) or just e_{ij}. Each edge is labeled with *transit time* $d(u_i, u_j)$ (or d_{ij}) and *capacity* $c(u_i, u_j)$ (or c_{ij}) to represent speed of movement and congestion constraints. A unit of flow (a person) entering edge e_{ij} at time t will reach node u_j at time $t + d_{ij}$. At most c_{ij} units of flow can enter the edge e_{ij} at the same time. Nodes are labeled with a capacity $a(u_i)$ to denote the maximum number of people that can occupy the node. All nodes have an initial occupancy at $t = 0$ of $s(u_i)$. $S \subseteq V$ is the set of source nodes, where $s(u_i) > 0$. We assume a single *exit* node $e \in V$. Multiple exits can be modeled by linking them to a superexit with 0 transit time and infinite capacity. The problem is to find the flow (allocation of people to edges over time) that minimizes evacuation time while respecting the flow constraints.

Consider an example in Figure 1. A floor topology is shown in Figure 1(a) where rooms A and B contain 10 people each (while having capacity of 20 people). The width of each corridor allows 5 people to enter simultaneously. The corresponding dynamic network is shown in Figure 2(b). It consists of nodes $V = \{u_1, \ldots, u_5\}$. Nodes u_1 and u_2 correspond to rooms A and B so that $s(u_1) = s(u_2) = 10$, $a(u_1) = a(u_2) = 20$. Nodes u_3 and u_4 model the junctions, that are empty initially $s(u_3) = s(u_4) = 0$ and at most 8 people can fit into each junction simultaneously, $a(u_3) = a(u_4) = 8$. Node u_5 is an exit node, it is initially empty $s(u_5) = 0$, and there is no restriction on the number of people that can enter it, $a(u_5) = \infty$. The capacity of each edge $c(u_i, u_j) = 5$ while transit times are proportional to the length and are illustrated on the graph: $d_{13} = 1, d_{14} = 1, d_{23} = 1, d_{24} = 1, d_{45} = 2, d_{35} = 8$. A corresponding solution to the minimal-time evacuation problem is presented in Table I: column *Time* indicates the departure time, column *Flow* the amount of flow and column *Path* the path traversed by the flow. The last unit of flow reaches exit at time 6.

The standard approach to solving dynamic flow problems [1] is to transform the graph $G = (V, E)$ into a *time-expanded network* [2]. For a given time horizon T, a time expanded network $G_T(V_T, E_T)$ is constructed by creating a copy of each node in V for each time unit ($V_T = \{u(t) \mid u \in V, t \in \{0, 1, \ldots, T\}\}$). An edge $(u(t), w(t + k))$ with



(a) Floor Topology (b) Dynamic Network

Figure 1. Floor topology and the corresponding dynamic network.

Table I
An evacuation plan for the problem in Figure 1.

Time	*Flow*	*Path*
0	5	(u_1, u_4, u_5)
1	5	(u_1, u_4, u_5)
2	5	(u_2, u_4, u_5)
3	5	(u_2, u_4, u_5)

capacity c is added to E_T if there is an edge $(u, w) \in E$ with transit time k and capacity c. E_T also contains *holdover* edges $(u(t), u(t + 1))$ with capacity $a(u)$ to designate flow that waits at node u from time t to $t+1$. This time-expanded network allows all variations of dynamic flow problems to be solved by polynomial static flow algorithms over the expanded graph. However, the expanded graph is larger by a factor of T which makes the entire solution process *pseudo-polynomial*. The major computational bottlenecks are the time and memory needed to construct the expanded network [2], [3]. The quickest evacuation problem can be seen as a transhipment problem with multiple sources and a single sink. [5], [6] proved the polynomial time complexity of the quickest transhipment problem by an approach based on temporally repeated chain-decomposable flows. However, their solution employs a highly expensive submodular function minimization algorithm as a subroutine, which makes it unsuitable for practical use. There are a number of evacuation models based on dynamic network flows [2], [3]. However, none explicitly model the spread of a hazard throughout the network. The most relevant approach is based on *minimal lexicographical flows* [7] that divides a building into priority zones, and searches for a solution that evacuates highest-priority zones in quickest time.

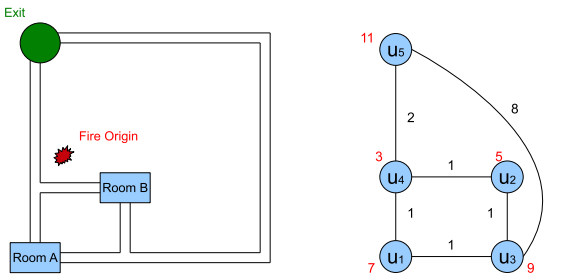
Other researchers have suggested using *hazard* spread to inform the evacuation plans. For example, in [8] the hazard forecast is based on a *hazard graph*, which estimates the time a hazard needs to reach nodes in the building. However, they ignore flow constraints. Finally, [4] consider *capacity constrained routing* heuristics, which keeps reserving evacuation paths and departure times from source nodes to the exit as long as there is unreserved flow left. The heuristic

observes the capacity constrains by reserving capacity at edges at future time points.

III. Problem Formulation

We now extend the dynamic flow model by associating *expiry times* with nodes and edges to represent the progress of a hazard. The topology of the underlying graph is therefore not constant as nodes and edges become unavailable. Each node $u \in V$ and each edge $(u_i, u_j) \in E$ have an expiry time, $T(u)$ and $T(u_i, u_j)$ respectively, denoting the last time points at which they can be safely used for flow traversal. In this paper we assume that only the expiry times of nodes are given, since the edge expiry times can be inferred as: $T(u_i, u_j) = \min\{T(u_i), T(u_j) - d(u_i, u_j)\}$, $(u_i, u_j) \in E$, i.e. the edge (u_i, u_j) cannot be safely traversed at time t if the start node u_i is unsafe before time t or if the end node u_j becomes unsafe before the flow reaches it.

Figure 2 presents an extension of our example from Figure 1 where a presence of the fire is detected. The projected expiry times of nodes $u_1, \ldots, u_5$ are $7, 5, 9, 3$ and 11 respectively. The evacuation plan from Table I is *not safe* since the evacuation route departing at time 3 and sending 5 units of flow through the path (u_2, u_4, u_5) is traversing node u_4 at time 4, i.e. after its expiry time $T(u_4) = 3$.



(a) Floor Topology (b) Dynamic Network

Figure 2. Floor topology and its dynamic network in the presence of hazard.

In such scenarios we want to evacuate as many people as possible, as safely as possible. We introduce the *maximal safe evacuation* (MSE) problem, that captures a core algorithmic challenge associated with achieving the above goals.

Problem 1 (MSE): Given a dynamic network $G(V, E)$, with edge transition times $d(u, v)$, edge capacities $c(u, v)$, node capacities $a(u)$, node supplies $s(u)$ and node expiration times $T(u)$, $u \in V, (u, v) \in E$, find an evacuation plan that sends the maximal amount of flow from sources $u \in S$ to exit e traversing only the edges and nodes that are not expired.

To formally state the problem we will use notation from [3]. Let $x_{ij}(t)$ denote the flow moving through the edge (u_i, u_j), starting at time t and reaching node u_j at time $t + d_{ij}$. Furthermore, let $y(t+1)$ denote the amount of flow waiting at node u_i from time t to time $t + 1$. Let $pred(u)$ denote the set of nodes v such that $(v, u) \in E$ and $succ(u)$ denote nodes v such that $(u, v) \in E$. Let time horizon T denote the expiration time of the exit, $T = T(e)$. The MSE solves the following problem:

$$\max \sum_{t=0}^{T} \sum_{u \in pred(e)} x_{ue}(t) \tag{1}$$

$$y_i(t+1) - y_i(t) = \sum_{u_k \in pred(u_i)} x_{ki}(t - d_{ki}) - \sum_{u_j \in succ(u_i)} x_{ij}(t),\ 0 \le t \le T,\ u_i \in V \tag{2}$$

$$y_i(0) = s(u_i), u_i \in V \tag{3}$$

$$0 \le y_i(t) \le a(u_i),\ 1 \le t \le T \tag{4}$$

$$0 \le x_{ij}(t) \le c_{ij},\ 0 \le t \le T - d_{ij},\ (u_i, u_j) \in E \tag{5}$$

$$x_{ij}(t) = 0,\ T(u_i, u_j) < t \le T - d_{ij},\ (u_i, u_j) \in E \tag{6}$$

The objective function (1) states that we want to maximize the amount of flow reaching an exit within time horizon T. Equation (2) is a set of flow-preservation constraints. Equation (3) sets the initial flow at each node, while equation (4) and equation (5) enforce the node and edge capacity constraints. The constraints (2)-(5) state the classical problem of maximizing the dynamic flow within a time horizon T. The main difference is the addition of the *expiration constraints* in equation (6) that forbid flow traversal over expired edges.

Observation 1: The *MSE* problem is solvable in polynomial time.

Proof Outline: This can be seen by a reduction to the *quickest transhipment problem* with *mortal edges* with start time α_u and end time β_v, proven by [5] to be polynomially solvable. By taking $\alpha_u = 0$ and $\beta_v = T(u, v)$ for all $(u, v) \in E$, and by setting the demand of the exit node e to a given flow value F, the solution to an MSE problem sends at least F units of flow if and only if the quickest transhipment problem has a solution. By changing the demand F through a binary search, we can find the maximal amount of flow for which the quickest transhipment problem returns a solution. ■

Unfortunately, the solution method of [5], [6] requires solving highly expensive (but polynomial) optimization subproblems, and is too slow for practical use. Since this is the best known solution algorithm for the quickest transhipment problem, and quickest transhipment can be encoded as MSE, we suspect that any exact solution to MSE will also be too slow for practical use on large problems.

IV. Flow-Based Heuristic Framework

Given that it is unlikely that an exact polynomial solution approach would be efficient in practice, we focus on development of efficient and safe *heuristic* approaches. The main contribution of this paper is a development of a family of flow-based and hazard-aware heuristic approaches to construct evacuation routes that respect expiry times based on the notions of node priority and path priority.

We first sort the nodes according to node-priority, so that sending flow from the most critical nodes is scheduled first. Then, for each node, and earliest possible departure time, we search for an evacuation path and the departure time that satisfies the expiration and the flow constraints while taking into account the previously reserved flow. We search for a path of the highest path-priority among all such paths. Once the path p is found, its capacity $c(p)$ is computed as the largest amount of flow that can be sent though p without violating the constraints. Finally, the $c(p)$ amount of flow is reserved for a given path and departure time. We keep sending the flow over such paths until either there are no paths left (in which case not all the flow from the source can be sent) or until all the flow has been allocated to evacuation paths. Algorithm 1 illustrates our overall heuristic approach.

Algorithm 1: `Heuristic Approach`

```
sort nodes u ∈ V according to node-priority;
for each source node u ∈ S in priority order do
    //schedule flow from u;
    departure time t = 0;
    while t ≤ T(u) do
        Find feasible path p from u to terminal, departing at
        t, with highest path priority;
        if no such p exists then
            t++;
            continue;
        c(p) ← minimal edge capacity on path p;
        s(u) ← s(u) − c(p);
        reserve c(p) flow;
        if s(u) = 0 then
            break;
```

We rely on *flow reservation*, a mechanism used originally in [4]. To reserve g units of flow over an edge (u_i, u_j) at time t means to increase the value of variable $x_{ij}(t)$ to $x_{ij}(t)+g$ from the system of equations (1)-(6). The capacity of the edge (u_i, u_j) at time t with respect to reserved flow is $c(u_i, u_j) - x_{ij}(t)$. Hence, our algorithm finds only paths that satisfy the edge capacity constraints (5). It is important to note that the algorithm finds only paths that *do not allow waiting* at intermediate nodes, i.e. as soon as flow arrives at intermediate node it enters the following edge on the path. This guarantees that the flow preservation constraints (2) are observed during the execution and that the node-capacity constraints (4) are observed, since no unit of flow is allowed to wait and therefore increase the node occupancy. Furthermore, our path-finding procedure ensures that the path never traverses an expired edge. Hence we can claim the following:

Observation 2 (Correctness): Algorithm 1 finds flows that are solutions to the MSE problem.

Note that restricting evacuation paths to *no-waiting paths* (that do not allow waiting at intermediate nodes) does not restrict the generality of our approach. It suffices to update the dynamic network by adding for every node u_i a *loop edge* (u_i, u_i) with capacity $c(u_i, u_i) = a(u_i)$ and transit time $d(u_i, u_i) = 1$ to ensure that any path that allows waiting on original dynamic network corresponds to a no-waiting path on the updated network.

Our algorithm in worst case computes a path for each unit of flow $F = \sum_{u \in V} s(u)$. Complexity of computing an evacuation path in worst case corresponds to linear-time traversal over implicitly-represented time-expanded graph $O(|V||E|T^2)$. This leads to the overall worst-case time complexity of $O(|V||E|FT^2)$.

Early-Notification Variant. Algorithm 1 can be modified based on the following insight: an evacuation planning algorithm is suitable for real-time use as long as it produces evacuation routes for evacuees prior to their scheduled departure time. That is, for people that are scheduled to start moving later it is acceptable to receive the evacuation instructions later. We therefore suggest a modification to Algorithm 1 which computes all evacuation paths that depart at time t before computing paths for departure time $t+1$. To implement this modification it suffices to switch the two iteration loops in Algorithm 1: outer loop iterates over departure time t while the inner loop iterates over nodes according to node-priority order. The critical advantage of this approach is that the portion of evacuation plan computed for departure time t can be immediately communicated to corresponding evacuees. This gives raise to an additional metric by which to evaluate the algorithms, the *communication delay*, which expresses the maximal time difference between computation time of an evacuation route, and its corresponding departure time. If the communication delay is small, the algorithm is adequate for real-time use even if the total computation time is large. Note that the communication delay of Algorithm 1 corresponds closely to the total computation time since the evacuation routes departing at time 0 can be allocated even during computations for the last node in priority order.

A. Node and Path Priorities

We will now describe the various versions of the flow-based and expiration-aware heuristics. The heuristics only differ in the choice of the *node-priority* and *path-priority* criteria and can be easily integrated into Algorithm 1 and its early-notification variant. We denote with $u_1 \prec u_2$ the fact that u_1 will be processed before u_2 with respect to the ordering criteria. Analogously, for paths p_1, p_2 starting at the

same time, we denote with $p_1 \prec p_2$ the fact that p_1 will be selected before p_2.

1) Path Priorities: We consider two *path priorities*, the *Max-Safety* and *Min-Distance*.

The *Max-Safety* priority prefers evacuation routes with the highest estimated *lead-time*, where lead-time denotes the minimal time-distance between traversal of a node on the path and its expiration. For a given evacuation route $p(u_1(t_1), u_2(t_2), \ldots, u_{k+1}(t_k))$ which traverses each node u_i at time t_i, we define *lead-time*$(p) = \min\{t_i - T(u_i) \mid i = 1, \ldots, k+1\}$ and

$$p_1 \prec p_2 \Leftrightarrow \textit{lead-time}(p_1) \geq \textit{lead-time}(p_2).$$

The selection of the exit paths based on maximal lead-time was proposed in [8], where the authors referred to it as *safety*. However, it was used in settings where capacity constraints were ignored.

Let $d(p)$ denote the length of the path with respect to transition times. For two paths p_1, p_2 between the node u and the exits the *Min-Distance* heuristic prefers the shorter route:

$$p_1 \prec p_2 \Leftrightarrow d(p_1) \leq d(p_2).$$

2) Node Priorities: We consider three *node priorities*, the *Smallest Expiration*, *Smallest Max-Safety* and *Largest Min-Distance*.

The *Smallest Expiration* heuristic first processes nodes with the *smallest expiration time*, i.e.

$$u_1 \prec u_2 \;\Leftrightarrow\; T(u_1) \leq T(u_2).$$

A node might be further from the expiration point, but the routes from the node to an exit might inevitably lead over nodes at times close to their expiration. Let the *lead-time of a node* be the lead-time of its safest exit route: *lead-time*$(u) = \max\{\textit{lead-time}(p) \mid p \textit{ is an exit route from } u\}$. The *Smallest Max-Safety* heuristic prefers nodes with smaller lead-time, i.e.

$$u_1 \prec u_2 \Leftrightarrow \textit{lead-time}(u_1) \leq \textit{lead-time}(u_2).$$

The *Largest Min-Distance* heuristic first processes nodes that are furthest away from the exit, using the shortest feasible paths. Formally, let $sp(u, v)$ denote the length of the shortest path between u and v. Let $sp(u) = \min\{sp(u, e) \mid e \textit{ is a terminal}\}$, denote the shortest distance to a terminal from the node u. The heuristic first processes nodes so that:

$$u_1 \prec u_2 \Leftrightarrow sp(u_1) \geq sp(u_2).$$

3) The Selection of Flow-Based Heuristics: We focus here on three particular combinations of the priorities, leading to the heuristics described in Table II. Other combinations are of course possible as well.

Table II
SELECTION OF FLOW HEURISTICS.

Heuristic	*Node Priority*	*Path Priority*
H_1	Smallest Expiration	Max-Safety
H_2	Smallest Max-Safety	Max-Safety
H_3	Largest Min-Distance	Min-Distance

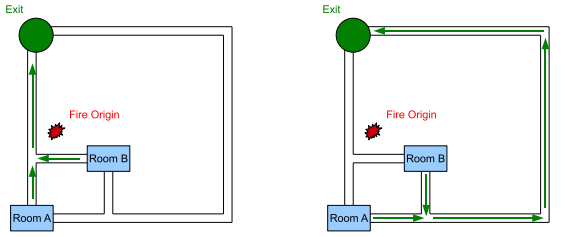
V. ALTERNATIVE SOLUTION APPROACHES

To evaluate our heuristics we compare them against the most relevant alternative solution approaches in the literature. In this section we describe these alternatives.

A. Congestion-Oblivious Approaches

We first consider the class of *congestion-oblivious* heuristics, which compute evacuation routes based on a distance to an exit or a proximity to a hazard, while ignoring the number of evacuees. These heuristics are easier to deploy from technological point of view and so we consider them to investigate whether these simpler approaches are sufficient, thus making our more sophisticated heuristics redundant.

The most commonly used is the *shortest-distance* heuristic. In most buildings, evacuation maps that are displayed to occupants typically indicate the shortest route to an exit. They are extremely easy to implement - already in design phase of a building the shortest routes are known. However, they ignore the number of evacuees and location of hazard. They can therefore steer people towards the fire or cause congestion. Figure 3(a) illustrates shortest-distance evacuation routes while Table I (on page 2) represents a corresponding solution that observes flow capacities..



(a) Shortest Path Evacuation (b) Safest Path Evacuation

Figure 3. Path-based solution approaches.

We also consider the *safest-path* heuristic proposed in [8]. For each node u it computes a max-safety path to an exit. It does take into account the spread of hazard but ignores the number of evacuees, thus possibly causing congestion. Figure 3(b) illustrates safest-path evacuation routes, while Table III shows a corresponding solution that observes flow capacities.

Please note that neither Table I nor Table III represent a *safe* solution. For example, the fourth evacuation path in Table III, departing at time 3 and sending 5 units of flow

Table III
AN EVACUATION PLAN FOR THE PROBLEM IN FIGURE 3(B).

Time	*Flow*	*Path*
0	5	(u_1, u_3, u_5)
1	5	(u_1, u_3, u_5)
2	5	(u_2, u_3, u_5)
3	5	(u_2, u_3, u_5)

along the path (u_2, u_3, u_5) reaches exit u_5 at time 12 which is after the exit expiry time, $T(u_5) = 11$.

B. Flow-Based Exact Approach

We also consider the *exact* approach which solves flow problems to optimality. If exact approach produces evacuation plans of significantly higher quality with comparable efficiency, then our solution approach would be inadequate.

We adapted the solution approach based on time-expanded graphs (described on page 2). We represent the expiration constraints by deleting all edges and nodes after their expiry time (delete all $(u_i(t), u_j(t + d_{ij}))$ where $t > T(u_i, u_j)$), and we select a time horizon based on the expiry of the exit, $T = T(e)$. By solving the maximal flow problem over the resulting static network using the Edmonds-Karp algorithm [9] we compute the maximum amount of flow that can reach the exit by time T.

C. CCRP-Based Heuristic Approach

The most related approach in the literature is the heuristic solution technique used in the *capacity-constrained routing planner* (CCRP) [4]. The authors construct evacuation plans with a goal to *minimize egress time* by repeatedly reserving *earliest arrival paths*. Just like in our work, the path reservation is handled implicitly using (what authors refer to as) time-series data structures. The authors do not provide sufficient details about the procedure they use to compute earliest arrival paths so exact reproduction of their work is not possible. However, we made the best effort to ensure a fair comparison by implementing their techniques using the same data structures that are used for our own heuristics. Furthermore, we implemented a hazard-aware version of the CCRP algorithm, where earliest evacuation paths are computed only over safe edges.

VI. EXPERIMENTS

We ask the following question in our empirical evaluation: Can the flow-based heuristics provide high-safety evacuation plans in real-time for large realistic instances? Can any of the alternative approaches provide comparable performance?

We execute our experiments over a set of grid instances. For a given dimension n_1 we create a square grid with $n = n_1 \cdot n_1$ nodes. For each pair of neighboring nodes $u_1(i_1, j_1), u_2(i_2, j_2)$ we introduce two directed edges $e(u_1, u_2), e(u_2, u_1)$. For each node u, we randomly select its capacity $a(u) \in [1, maxNodeCapacity]$. Initial supply $s(u)$ is selected to reflect an uneven distribution of people that can be found in a university building, with 5% of large lecture halls of random occupancy of up to 200, 30% of medium lecture halls with random occupancy of up to 50, 25% of meeting rooms with random occupancy of up to 10 and remaining 40% of office spaces with occupancy of up to 3 people. For each edge, $e(u, v)$ we randomly select its capacity $c(u, v) \in [0, maxEdgeCapacity]$ and its traversal time $d(u, v) \in [1, maxEdgeDistance]$. We select an exit node e to be in the lower right corner of the grid. We then generate expiration times $T(u), u \in V$. We first select a node $s \in V$ placed in the center of the grid to be the origin of a fire, and then assign expiration times $T(u)$ proportional to their shortest-path distance from the origin, $sp(s, u)$: we set $T(u) = sp(s, u) \cdot \mathit{fireSpeed}$, where $\mathit{fireSpeed}$ denotes how slowly fire spreads in comparison to flow traversal. We use $\mathit{fireSpeed} = 5$. All experiments are executed as a single process on a dual Quad core Intel Xeon processor running at 2.66 GHz, with the Fedora 9 operating system.

A. Comparison Against Alternative Approaches

In the first set of experiments we compare the performance of our heuristics H_1, H_2, H_3 (defined on page 5) using Algorithm 1 against the alternative solution approaches from Section V: congestion-oblivious approaches, flow-based exact approach and the CCRP-based heuristic. First, we gradually increase the grid size. The results are reported in the upper part of Table IV and represent average values over 100 trials. The input parameters for the random generation were $maxNodeCapacity = 50$, $maxEdgeCapacity = 10$, $maxEdgeDistance = 20$. Column **Grid** is the grid dimension n_1; **T** is the average time horizon T for the time-expanded graph, chosen to be the expiry of the exit node e; **Supply** is average total initial flow (number of people). $\mathbf{F_{ex}}$, $\mathbf{F_{h1}}$, $\mathbf{F_{h2}}$, $\mathbf{F_{h3}}$ and $\mathbf{F_{ccrp}}$ denote the amount of flow that reaches the exit within the time horizon **T** for the exact, our heuristic approaches and the CCRP heuristic respectively. Columns $\mathbf{t_{ex}}$, $\mathbf{t_{h1}}$, $\mathbf{t_{h2}}$, $\mathbf{t_{h3}}$ and $\mathbf{t_{ccrp}}$ denote the corresponding computation times in seconds. Finally, $\mathbf{F_{ob1}}$ and $\mathbf{F_{ob2}}$ report the amount of flow that safely reaches the exit using the congestion-oblivious shortest-distance and safest-path heuristics respectively. We do not report their running times as they execute within a few milliseconds.

The results clearly confirm that the exact approach is too slow to be used in real-time settings, even for instances of moderate size, taking about 10 minutes to compute optimal evacuation plans for instances with 225 nodes [1]. Secondly, the oblivious heuristics generate plans of inadequate quality. For small instances they evacuate over 85% of the number evacuated by the exact approach, but this quickly deteriorates

[1]This is despite our best efforts to optimize implementation of the time-expanded graph to exploit the sparsity of the network. Initial versions of the same algorithm, that used the incidence-matrix representation were orders of magnitude slower.

Table IV
COMPARING EXACT, FLOW-BASED HEURISTIC AND CONGESTION-OBLIVIOUS HEURISTIC APPROACHES.

Grid	T	Supply	$\mathbf{F_{ex}}$	$\mathbf{F_{h1}}$	$\mathbf{F_{h2}}$	$\mathbf{F_{h3}}$	$\mathbf{F_{ccrp}}$	$\mathbf{t_{ex}}$	$\mathbf{t_{h1}}$	$\mathbf{t_{h2}}$	$\mathbf{t_{h3}}$	$\mathbf{t_{ccrp}}$	$\mathbf{F_{ob1}}$	$\mathbf{F_{ob2}}$
5	145	296	266	258	259	258	236	0.73	0.0	0.0	0.0	0.0	184	231
7	214	611	582	569	572	571	527	6.05	0.01	0.01	0.01	0.01	373	450
9	263	1099	1047	1011	1017	1008	894	27.52	0.03	0.03	0.03	0.03	581	702
11	317	1683	1573	1501	1517	1503	1332	86.78	0.09	0.08	0.08	0.07	772	956
13	384	2292	2094	1993	2011	1975	1709	244.75	0.25	0.22	0.23	0.14	956	1129
15	438	3074	2769	2581	2607	2514	2136	595.96	0.68	0.56	0.6	0.25	1038	1374
10	255	440	440	440	440	440	437	38.87	0.2	0.03	0.06	0.14	341	437
10	255	870	870	870	870	870	858	40.27	0.02	0.02	0.06	0.03	591	833
10	255	1105	1105	1105	1105	1085	1063	40.93	0.03	0.03	0.03	0.04	590	893
10	255	1420	1420	1406	1420	1381	1145	40.67	0.04	0.04	0.05	0.04	743	1054
10	255	1750	1750	1664	1687	1658	1198	41.25	0.07	0.07	0.08	0.04	798	1163
10	255	2335	1999	1723	1759	1854	1330	41.39	0.12	0.14	0.12	0.04	853	1240

to about 50% for the instance with 225 nodes. However, the *safest-path* oblivious heuristic performs noticeably better than the *shortest-distance* heuristic. The CCRP-based heuristic performs significantly better than oblivious heuristics, but still suffers from inadequate quality, which deteriorate to 77% for the largest instance. The flow-based heuristics, however, achieve significantly higher quality results with small running times. On average, our best-performing flow-based heuristics produce plans achieving more than 94% of the quality of the exact approach, but computing almost 1000 times faster. A computation time of under 1 second clearly allows for frequent replanning as events happen for moderately sized instances. All three heuristics have comparable performance, with heuristic H_1 being slightly outperformed by the other two.

The first experiment discussed above considered instances where, on average, it was not possible to get all the flow out before the exit expires, even for the exact approach. This might bias our results, as some approaches (in particular the congestion-oblivious heuristics) might perform better in looser scenarios where all the initial flow can escape. We therefore fixed one of the 100 node instances, and we varied the initial supply of flow by setting for each node u its initial supply $s(u) \in [0, maxNodeSupply]$, where $maxNodeSupply$ parameter is gradually increased from 10 to 50. The results appear in the lower part of Table IV. We can see that the *safest-path* congestion-oblivious heuristic does perform better for low concentrations of people, but for concentrations of 10 or more people per room suffers from significant quality deterioration even if the exact approach is capable of evacuating all the people. We also notice a gradual increase in computation time for our flow-based heuristics. While the performance of the exact approach is still too slow, it does not deteriorate with the increase in supply. This is consistent with the performance guarantees provided by the Edmonds-Karp algorithm which do not depend on the amount of initial supply.

B. Scalability Study and Early Notification Version

In the first set of experiments we concluded that the exact approach is too slow for real-time use while the congestion-oblivious approaches and CCRP-heuristics suffer from poor quality, thus leaving our flow-based heuristics as the preferred approach. In the second set of experiments we perform a scalability study of our heuristics H_1, H_2, H_3 using Algorithm 1 and their *early-notification variants*, denoted as H_{1v}, H_{2v} and H_{3v} respectively. For each heuristic we report the number of evacuated people $\mathbf{F_1}$, $\mathbf{F_{1v}}$, $\mathbf{F_2}$, $\mathbf{F_{2v}}$, $\mathbf{F_3}$, $\mathbf{F_{3v}}$, and their corresponding running times $\mathbf{t_1}, \mathbf{t_{1v}}, \ldots, \mathbf{t_{3v}}$. In addition, we report the *communication delay* for early-notification variants of our heuristics (described on page 4), denoted as $\mathbf{t_{1vd}}$,$\mathbf{t_{2vd}}$,$\mathbf{t_{3vd}}$ assuming that 1 time unit in an evacuation plan corresponds to 1 second. We report results over grid instances with dimensions $5, 10, \ldots, 35$, i.e. for graphs of up to 1225 nodes.

The experimental results are presented in Table V and represent average values over 100 trials. The average flow values are rounded to a nearest integer value. We can observe the following:

1) Both the standard flow-based heuristics and their early-notification counterparts compute in roughly the same time and produce the results of comparable quality.
2) The standard flow-based heuristics are too slow for real-time use for instances greater than 400 nodes where computation takes more than couple of seconds. This is mainly because all the grid nodes are junctions with high degree. We demonstrate in the next subsection that in realistic instances we have many more nodes of smaller degree and thus faster performance.
3) Early-notification heuristics *are suitable* for real-time use even for the largest instances of over 1200 nodes. Their *communication delays* are a fraction of a second for all the instances.

C. Scalability Over Realistic Instances

Finally, we investigate scalability of our heuristics over instances that exhibit a real-world structure. Buildings typically have backbone corridors, a number of rooms which correspond to nodes of degree 1 or 2 while multi-story buildings are connected through staircases which could cause a congestion during evacuation. We evaluated our

Table V
SCALABILITY STUDY COMPARING FLOW-BASED HEURISTICS AND THEIR EARLY-NOTIFICATION VARIANTS.

Grid	T	Supply	F_1	F_{1v}	F_2	F_{2v}	F_3	F_{3v}	t_1	t_{1v}	t_{1vd}	t_2	t_{2v}	t_{2vd}	t_3	t_{3v}	t_{3vd}
5	124	306	236	220	238	221	244	229	0.01	0.01	0.0	0.01	0.0	0.0	0.0	0.0	0.0
10	275	1456	1144	1106	1177	1108	1196	1153	0.07	0.05	0.0	0.05	0.05	0.0	0.05	0.04	0.0
15	428	3099	2458	2413	2518	2422	2478	2448	0.51	0.3	0.01	0.38	0.31	0.01	0.44	0.4	0.01
20	584	5691	3595	3675	3698	3687	3805	3714	3.65	2.18	0.04	2.71	2.15	0.04	3.3	3.1	0.02
25	732	8597	5481	5550	5540	5572	5634	5513	12.86	8.35	0.09	9.86	8.11	0.1	12.03	11.27	0.04
30	860	12424	5695	5664	5728	5608	5912	5840	52.25	42.18	0.22	44.0	40.19	0.21	46.9	46.44	0.1
35	957	16828	8009	7898	8006	7968	8047	8019	111.13	98.13	0.38	92.35	92.45	0.35	102.7	100.17	0.15

heuristics over one such instance, a graph that models a hotel with 6 floors, each floor consisting of two parallel corridors with 30 rooms each. Each room has a maximal capacity of 5 people. Corridors are connected through a western and eastern staircase which lead to the ground level connected to three exists.

We discovered that while the exact approach is still taking unacceptably long times to compute (more than 30 minutes), our heuristics run significantly faster, probably due to a larger number of lower-degree nodes. The total number of nodes in the corresponding dynamic network was more than 1000 and yet our heuristics executed in less than 0.5 seconds. In comparison, our heuristics take on average more than 40 seconds to compute over a comparable random grid instance of 900 nodes (grid dimension 30 in Table V). Furthermore, a multi-story hotel instance with 16 floors induces dynamic network of 2892 nodes and is solved in 7.5 seconds by our heuristics while early-notification version introduces communication delays of less than 0.2 seconds. We can therefore conclude that the random instances used in our experiments are significantly more challenging than the real-world instances, and we could expect our algorithms to scale up to real-world structures with thousands of nodes.

ACKNOWLEDGMENT

This research was funded by the Higher Education Authority PRTLI-IV Scheme as part of the NEMBES project.

VII. CONCLUSIONS AND FUTURE WORK

In this paper we considered the problem of planning evacuation routes in deteriorating networks, where nodes become unavailable over time, due to the spread of a hazard. We developed a family of flow-based evacuation heuristics and compared its performance against a number of alternative approaches: the exact approach based on maximising flow over a time-expanded graph, congestion oblivious approaches and the hazard-aware adaptation of the related flow-based CCRP heuristic. In empirical evaluation we demonstrated that the exact approach does not scale while the congestion-oblivious and CCRP approach produce solutions of inferior quality. Our flow-based heuristics produce high-quality plans in small amounts of time. In particular, the early-notification variant of our heuristics is able to compute relevant portions of the evacuation plan in a real-time manner even for large instances with thousands of nodes. To the best of our knowledge the set of heuristics we present constitutes the state-of-the-art approach for the real-time evacuation planning. In future we plan to integrate our flow-based heuristics into a pedestrian-movement simulator to evaluate the performance of our algorithms in more realistic settings.

REFERENCES

[1] L. R. Ford and D. R. Fulkerson, "Constructing maximal dynamic flows from static flows," *Operations Research*, vol. 6, no. 3, pp. 419–433, May-June 1958.

[2] B. Kotnyek, "An annotated overview of dynamic network flows," INRIA, Tech. Rep. RR-4936, 09 2003.

[3] H. Hamacher and S. Tjandra, "Mathematical Modelling of Evacuation Problems: A State of Art," Fraunhofer ITWM, Tech. Rep. 24, 2001.

[4] Q. Lu, B. George, and S. Shekhar, "Capacity constrained routing algorithms for evacuation planning: A summary of results," in *Advances in Spatial and Temporal Databases*, ser. Lecture Notes in Computer Science, C. Bauzer Medeiros, M. Egenhofer, and E. Bertino, Eds. Springer Berlin / Heidelberg, 2005, vol. 3633, pp. 291–307.

[5] B. Hoppe and E. Tardos, "The quickest transshipment problem," in *Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*, ser. SODA '95. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1995, pp. 512–521.

[6] B. Hoppe and O. Tardos, "The quickest transshipment problem," *Math. Oper. Res.*, vol. 25, pp. 36–62, February 2000. [Online]. Available: http://portal.acm.org/citation.cfm?id=351180.351185

[7] H. W. Hamacher and S. Tufekci, "On the use of lexicographic min cost flows in evacuation modeling," *Naval Research Logistics*, vol. 34, no. 4, pp. 487–503, August 1987.

[8] M. Barnes, H. Leather, and D. K. Arvind, "Emergency evacuation using wireless sensor networks," in *Proceedings of the 32nd IEEE Conference on Local Computer Networks*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 851–857.

[9] J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," *J. ACM*, vol. 19, pp. 248–264, April 1972. [Online]. Available: http://doi.acm.org/10.1145/321694.321699