VISUAL ANALYSIS OF SITUATIONALLY AWARE BUILDING EVACUATIONS

by

Jackie Dean Guest Jr.

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science in
Computer Science

Charlotte

2012

ABSTRACT

JACKIE DEAN GUEST JR.. Visual analysis of situationally aware building evacuations. (Under the direction of DR. KALPATHI R. SUBRAMANIAN)

Rapid evacuation of large urban structures (campus buildings, arenas, stadiums, etc.) is a complex operation and of prime interest to emergency responders and planners. Although there is a considerable body of work in evacuation algorithms and methods, most of these are impractical to use in real-world scenarios (non real-time, for instance) or have difficulty handling scenarios with dynamically changing conditions. Our goal in this work is towards developing computer visualizations and real-time visual analytic tools for building evacuations, in order to provide situational awareness and decision support to first responders and emergency planners. We have augmented traditional evacuation algorithms in the following important ways, (1) facilitate real-time complex user interaction with first responder teams, as information is received during an emergency situation, (2) visual reporting tools for spatial occupancy, temporal cues, and procedural recommendations are provided automatically and at adjustable levels, and (3) multi-scale building models, heuristic evacuation models, and unique graph manipulation techniques for producing near real-time situational awareness. We describe our system, methods and their application using campus buildings as an example. We also report the results of evaluating our system in collaboration with our campus police and safety personnel, via a table-top exercise consisting of 3 different scenarios, and their resulting assessment of the system.

DEDICATION

This work is dedicate to my wife, Amelia and family: Jack, Ruth, Jayda, Jill, Jackie III, Joey, Rick, Tyler, Carson, Kaleigh, Marlee, Stephanie, Aidan, Asher, and Owen.

ACKNOWLEDGMENTS

This work was supported by a grant from the National Institute of Justice, Office of Justice Programs. Thanks to Jeff Baker and Josh Allen. This work is better because of their insight.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

In any emergency or incident involving large urban structures (arenas, stadiums, a college campus), the safety of the occupants is of paramount importance. Thus every building has a set of passive safety features (sprinkler systems, fire extinguishers) and evacuation plans or routing maps posted at various points within the building.

In recent years, the study of evacuation modeling has evolved using both mathematical and algorithmic approaches. There has been a large body of work, categorized as *macroscopic* and *microscopic*, as detailed below. All of these methods study the problem of evacuating the occupants from the concerned structure in the shortest possible time, also known as the egress time. What is lacking in these methods is the ability to handle real-world scenarios involving *large and complex structures that may have multiple buildings* and more important, the *ability to react in real-time to dynamic changes in the scenario*, such as blocked stairwells or hallways, and provide useful recommendations to responders who can mitigate damage, injury, or loss of life. For a commander overseeing the evacuation, the ability to clearly and unambiguously understand the dynamically changing situation is very important, being useful for optimal allocation of limited resources and personnel, and for making other timely decisions.

In this work, we address these challenges with the primary goal of responding to dynamic events in real-time during an emergency. We begin with an existing heuristic based route planning algorithm[15], make certain modifications, and embed this as part of a visual analytic system that permits complex real-time interactions during the event. This then permits dynamic changes in the building accessibility to be incorporated, and alternative routing assessed in real-time. To accomplish this, we employ an LOD graph representation of the underlying urban structures that permits real-time recommendations to be presented to the emergency planners and responders for informed decision-making. This is combined with realistic models for building occupancy and traffic flow.

Figure 1.1: Candle Light Vigil. Virginia Tech University after the massacre of 33, including the perpetrator, in April 2007. The research contained here is intended as a work toward a technical aid for Emergency Responders in the event of tragedies in large urban settings.

Interactive visual analytic tools permit quick exploration of possible impacts of the current situation, and the increased situational awareness for emergency commanders and responders permit more optimal use of scarce resources, for instance, dispatching responders to areas of need, such as congested sections of a building, handling casualties, etc.

This is more than just an interesting problem. To cite one example that is in the current news, if a suspected shooter is spotted on a college campus, the first step of the standard procedure is to *lockdown the whole campus*. How to evacuate a large campus after this happens - where, when, what part, and how - will depend on the way the scenario unfolds. This is a major concern for campus and other government public safety personnel. Situationally aware models with appropriate interactive visualization can be a central part of the tools used in this case.

Though our work is not complete, and we hope to continue the study of spatio-temporal evaluations of emergencies in urban environments we have accomplished several interesting points of analysis.

(1) We have departed from the Agent Based constructs and 3D game style visualizations that are the wisdom of the day in evacuation modeling of environments. Our direction is to create light weight data structures that can be as mobile as those who must implement them. Multiple GPU/CPU configurations that can be used for a-priori evaluation and experimentation are not a realistic infrastructure for our audience in the field. The combination of human intelligence in real time and technology as a visualization tool for the emergency responder is unique in this area of Computer Science.

(2) We have devised and implemented, in usable software, the notion of reducing the nodes in connected graphs and saving the results of the reduction for shortest path calculations while maintaining their spatial quality for visual presentation. Building interiors, university, commercial, industrial or campus streets, and walkways are a viable domain for our solution and we believe they contribute to the study of graph theory in these applications.

(3) The data model implemented in the evacuation scenario is generic enough to be implemented in any software of this kind. Any campus environment can be constructed in this way. The data driven concepts for organizing and referencing an urban environment including one or more objects for visualization is a difficult task. We combine data driven concepts with a usable interface to implement a portable way to manage these difficulties. Portability of our software was a primary concern. A mass loading of an entire campus is not possible and a simple inclusion of one object is in most cases unrealistic. The campus is organized and a permanent connectivity is applied to objects on an as needed basis. Over time the spatial definition and combination of objects and scenarios can be built into a collection that can include any possible group of objects, and even a large list of expected scenarios. Once an expected event is built and saved it can be reused, reevaluated, and adjusted in real time. It can be said that our system is trainable and grows over time.

We begin with a review of current work on evacuation algorithms, followed by a description of the techniques we use to create our 3D models and building networks. This is followed by a description of our modified capacity based route planner, followed by a detailed description of the major features of our interactive visual analytic system and its reporting and recommender functions. We describe the use of this system on typical scenarios using our campus buildings as a test case. We evaluated our system and its performance via a table-top exercise, consisting of 3 scenarios: gas leak in building, active shooter and an explosion in an adjacent structure. Performed in collaboration with our campus police and safety personnel, we report on their assessment and recommendations.

CHAPTER 2: EVACUATION PLANNING

Macroscopic approaches focus primarily on a lower bound of egress time. The evacuees are treated as a unit or group of units and moved from source to destination, or safe zone. Interaction between these units can be defined with capacity/congestion rules and the primary goal is to minimize the time it takes for all evacuees to reach an exit. Microscopic approaches use agent based modeling where each evacuee is governed by unique rules of behavior. Interaction between individuals and their environment can be defined based on spatial and social parameters.

2.1 CAPACITY BASED (MACROSCOPIC).

The inputs for this approach are a graph structure and evacuee populations. The output is a route plan with start times, and a location matrix for each evacuee for each time segment defined. This is therefore an a priori solution that is evaluated based on the time it takes for each evacuee to reach a safe zone or exit. This evaluation is applied whether the solution is optimal or sub-optimal. The evaluation of the route planning algorithm is based on processing cost. Obviously, the evaluation of a given approach simply based on minimizing the evacuation time does not consider the details of the complexities of the individual evacuees in the process of moving through their environment. For this reason, the application of network flow, capacity, and congestion restraints are applied to add reality to the challenges of moving within the environment to be evacuated.

This is the reason the Network Flow problem is referred to as optimal because it can represent the conditions of each evacuee during an egress process. Of course, the term optimal must be considered from the context of the methods for evacuation planning because flow, capacity, and congestion cannot represent all the factors that effect individual movements within an environment. Heuristic approaches such as the Capacity Constrained algorithms proposed by [15] attempt to find lower cost algorithmic solutions at the further expense of the detail of each evacuees egress. However, these approaches are interesting because they can be evaluated quickly from a user perspective. For example, the route planner can be modified, executed, and visualized in an animation relatively fast. This can yield a quicker way to canalize the algorithm for individual behaviors, at least from a spatial perspective, as the route plan moves from time step to time step.

Network Flow algorithms were applied to building evacuation planning beginning with work funded and developed under the sponsorship of the National Bureau of Standards, Center for Fire Research in the late '70s and early '80's. An application named EVACNET+ was produced by Francis and Kisko in 1984. Further development of these approaches resulted in EVACNET4 which takes the network model one provides and determines an optimal plan to evacuate the building in a minimum amount of time. This is done using an advanced capacitated network flow transshipment algorithm, a specialized algorithm used in solving linear programming problems with network structure. From the user's point of view, all the user does is supply the model, ask EVACNET4 to run it, and then examine the results. [5].

Linear programming methods based on Network Flow yield optimal solutions. Though these solutions are optimal their algorithmic cost is high. The Maximum Flow Problem is one network solution that is implemented with costs as high as $O(n^3)$[15] and yielding a best known cost of $O(nm \lg n^2/m)$. Because the processing cost of these polynomial time algorithms is so high, in general they are considered to be impractical. Over the last 20 years 'optimal' plans like the algorithms developed by Hoppe and Targus have been modified as general grid network problems, where network arcs have constant length. The work of Naoyuki Kamiyama et al.[9] applies two initial conditions to the network flow problem. For each vertex the sum of transit times of arcs on any path takes the same value, and for each vertex the minimum cut is determined by the arcs incident to it whose tails are reachable. They refer to these as having uniform path-length and being fully connected. Their work proves that for a 2d grid network the transhipment problem can be solved in $O(n \lg n)$ time.

Our model is based on a heuristic approach, the Capacity Constrained Route Planner algorithm proposed by Shekhar etal.[15]. This approach attempts to find lower cost algorithmic solutions at the expense of the detail of each evacuee's egress. These approaches are interesting because they can be evaluated quickly from a user perspective. For example, the route planner can be modified, executed, and visualized relatively fast. This can yield a quicker way to analyze the algorithm for individual behaviors, at least from a spatial perspective, as the route plan moves from time step to time step.

The work of Shekhar and Yoo[19] compares models relevant to the study of nearest neighbor paths. Also Kim, et.al[11] discuss contraflow in reconfigured networks for emergency route planning. This work gives insight into reconfiguring a network that is damaged and therefore relevant to our work. Since we have only considered multi-directional paths in buildings these methods have not been implemented.

We utilized a heuristic approach, so as to reduce the optimal network flow problem to a generalized shortest path problem. Heuristic approaches have reduced algorithmic cost but are referred to as sub-optimal; however, we find that they work reasonably well for our current scenarios. The inputs for this approach are a graph structure and evacuee populations. The output is a route plan with start times, and a location matrix for each evacuee for each defined time segment. This is therefore an a priori solution that is evaluated based on the time it takes for each evacuee to reach a safe zone or exit.

2.2 AGENT BASED (MICROSCOPIC).

Microscopic evacuation planning simulates the detailed interaction of individuals with their environment including fellow evacuees. These approaches do not exclude the deterministic path or goal that is the foundation of the route planner, but they primarily rely on behavior rules applied to each evacuee to overcome the "lack of detail" inherent in network flow or heuristics based planners. These methods are also referred to as Agent Based Models, or ABM. They integrate the routing logic and time expansion into one process, therefore they are used for visual evaluation rather than graphical or chart based evaluation. However, the goal remains to minimize the time to evacuate individuals to an exit or safe zone. Certain ABM models can exhibit more complex flow patterns (e.g., swirling or chaotic patterns) than are possible with macroscopic models.

A general description of these approaches can be found in [4]. The most important aspect of ABMs are the rules applied to each evacuee. There are no globally defined rules of behavior but there are common applications of constraints placed on the agents. Castle et. al. [4] include a detailed list of some rules or attributes.

- **Autonomy:** Agents receive and transmit information to make individual decisions.
- **Heterogeneity:** Agents with identical rules/attributes can be grouped.
- **Pro-active/Goal-directed:** Agents can be directed on specific paths.
- **Reactive/Perceptive:** Agents can be programmed with prior knowledge and awareness.
- **Bounded Rational:** Limits must be placed on agents ability to analyze their environment.

- **Communicative:** Agents are required to add and depend on information from each other.
- **Mobility:** Spatial roaming and interaction can be implemented.
- **Adaptive-Learning:** Agents can be trained.

Because of the adaptive parameter of ABM, neural networks and fuzzy-logic (if-then-else)[14] approaches are adapted to building evacuation simulation. Discrete Particle Swarm Optimization[6] can also be applied. Velocity and spatially based[7] rules of interaction between agents in a simulation are also a common approach.

Neural Networks, Fuzzy Logic. This method begins with the training of an adaptive neural network based system. The evacuation simulations conducted numerous times until a predetermined best case scenario is developed by an adaptive learning process from information stored and flowing through the network. Obviously the execution of this process will be time consuming based on the accuracy required by the prerequisite definitions of the simulation. According to [14], even though computing power has increased significantly over recent years these processes are very expensive from a processing cost perspective. Processing power must be significant for, large scale, real world situations and may be exclusive based on the size of a given scenario.

A general description of the process described in [14] illustrates an overview of the neural network fuzzy logic approach. The authors propose an Adaptive Network based Fuzzy Inference System (ANFIS), to accomplish the task of predicting pre-evacuation behavior, and a predictive back propagation model to predict evacuation response. An example of a Fuzzy Logic rule applied in their implementation is: If Age is young, fire experience is limited and fire cue is intensive flame or smoke and acquired directly, then initial reaction is to inform others. Other behaviors are defined based on the parameters age, fire experience, and fire cue. Obviously, these rules require knowledge of the demographic of the evacuees which in most cases will be unknown. For the analysis of the ANFIS system questionnaires were distributed to 150 individuals that were evacuated from 3 different fires in Hong Kong, China.

Discrete Particle Swarm Optimization. The PSO algorithms were developed from the study of bird flocks and fish schools to attempt to mathematically analyze the behavior of this phenomena during panic situations. Therefore the PSO is applied in the area of emergency evacuation to attempt to predict the actions of human evacuees during emergency situations in a building evacuation. The work of Fang[6] is an adaption of the Cellular Automata (CA) PSO with a leader function included. The intent of this study is to simulate results from an evacuation scenario which includes one source and one exit to illustrate the feasibility of the proposed approach in representing evacuation phenomena such as jamming and clogging.

In general formulas are applied to represent the velocity of evacuees in successive iterations of the PSO algorithm. The selection of variables from the leader is used as a best performing evacuee and then applied to the other evacuees in the next iteration. The leader, in the case of this example, is simply chosen as the evacuee closest to the exit of the original source destination. An ABM for the interaction of these evacuees is therefore present for the analysis of the actions and behavior of evacuees at choke points in the model. It is important to note that this is simply another mathematical solution to software analysis of evacuee behavior at the agent level. The results could be compared to the same approach using an different ABM such as a Neural Network adaptive approach.

Velocity Obstacles. These methods have their roots in robotics. The relative speed and location of nearby objects is calculated and applied to an algorithm that adjusts individual velocities to avoid collisions. This ABM is a hybrid approach related to particle behavior and collision avoidance. These methods are important if the implementing user requires a real time visual solution for analysis. This is not a goal in neural networks or PSO methods.

The following is a general description of a hybrid method from Guy, Chhugani, Kim, Satish, Lin, Manocha, Dubey [7]. The authors goal is to produce avoidance algorithms for real time simulations. If 2 planar objects are moving in the same plane they are given vector sets that contain all velocities that will result in a collision at some time after a given moment in time. These sets can be visualized as a cone in the plane. The intersection of the cones represent collision points. If there are more objects they all must contain this information about each other. The information needed by each object to related to each other object is stored and retrieved efficiently in a KD-Tree structure. Each object is then moved in its designated direction for a designation time slice and velocities are adjusted for each object in the plane to avoid collision in the next time slice.

The authors claim a 10 fold reduction in processing cost from previous applications of VO. They have also created a software package called Clearpath. The package is multi-threaded to improve the Extended VO's real time performance on multi-core processors. This allows the number of simulated objects to be increased for the best real time performance on a given multi-core processing system. Their results indicate guaranteed collision avoidance with nearly 20 visual FPS on a quad processor machine while processing 25K objects through a simulated stadium geometry.

We believe our system captures sufficient detail using a congestion model. However, because agents can be trained, we are evaluating Q-Learning/SARSA techniques to add human factors to future work.

2.3 VISUAL ANALYTICS

Visual analytics involves effectively combining interactive visual displays with computational transformation, processing and filtering of large data[21]. One focus of visual analytics is real-world problems involving situationally-aware decision support. Andrienko et al.[1] focused on automatic generation of transportation schedules for evacuation from a disaster zone; visual analytic tools were used for verification by human experts. Campbell and Weaver[3] used interactive visualization tools for hospital evacuation scenarios that involved training first responders. The work of Kim et al.[10, 12] focused on use of mobile devices for situationally aware emergency response and training, and thus their approach is similar to our work. They demonstrated their system with an evacuation simulation of the Rhode Island club fire of 2003. Our system is considerably more general and is scalable to large urban buildings and provides the means to interrupt the simulation based on new situational information or dynamic changes.

The use of linked views is an important technique to connect different representations of information within a single visualization, with applications specific to urban structures[16, 8]. Sensor networks are used within buildings to help create interactive visual analytics. Visual analytics tools such as Jigsaw[20] are available for integrating process output data from an application. We are using custom code in our system, therefore toolkits are not utilized.

The work Ivanov et. al[2] is not specific to building evacuations, however their use of data graphs to interact with maps provided inspiration for our cross platform bar chart displays that interact with our simulation. This work is from a growing segment of the visual analytics referred to as the visual analytics of movement or Geo-analytics. These scientists are focused on the problem of visualizing spatial and temporal datasets in geography and cartography. From our perspective, in visualizing building evacuations these methods are most significant and they dominate our approach. We have split our data into 2 primary sections. An animation and a cross connected interactive display. The animation display contains both spatial and temporal data and the user can view an evacuation as an animation and move back and forth in the animation. The interactive section defines the evacuation from the perspective of significant events and congestion which is a primary concern in evacuation. These are, in many ways these are Geo-analytic approaches.

Another common approach from Geo-analytics is the representation of path, congestion, and movement data along a plane perpendicular to the spatial plane. Figure 2.1 from [23] is an example of this type of presentation. The problem of visualizing movement in a single time unit is solved by expanding movement in this way.

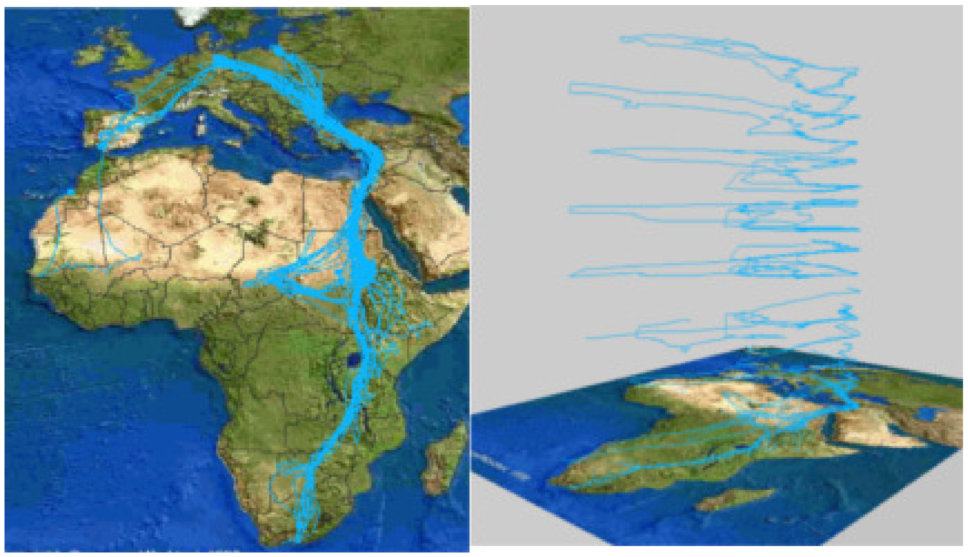


Figure 2.1: Tracks of 35 storks during 8 years, about 2,000 positions.

This approach is partially implemented in the visualizations of the route planner activities of our application. Because we are concerned primarily about congestion(based on our audience) and this visualization is only applicable to movement it is only mentioned here. Continuing work and the use of other evacuation algorithms will result a need for the scenario build functions to incorporate path choosing analysis.

Ivanov et. al[2] also present an approach where the analysis of movement is visualized by segmenting time. A plot of multiple objects moving in space for a given period of time can cause what they refer to as worms. The longer the worm for a given object over a given amount of time on the 2D plane of a map, the faster the object is moving. Speed in our implementation is primarily controlled by the capacity and occupancy of a given area. If there is room to move to the next discretized space the evacuee moves. As our application becomes more concerned with large outdoor spaces where discretization of space is less practical from a computing perspective these approaches will be invaluable.

CHAPTER 3: METHODS

Figure 3.1 illustrates the major components of our visual analytic system for situationally aware evacuations. There are two major components to the system. Functions in the Route Planner involve a significant amount of a priori processing and initialization. The visual analytic system is a highly interactive system that is user driven and can inject and respond to dynamic changes during evacuations. It also consists of reporting and visual analysis functions that can assist an emergency planner on exploring different scenarios, as to the use and deployment of resources, dispatch responders, effect of rerouting occupants, etc.

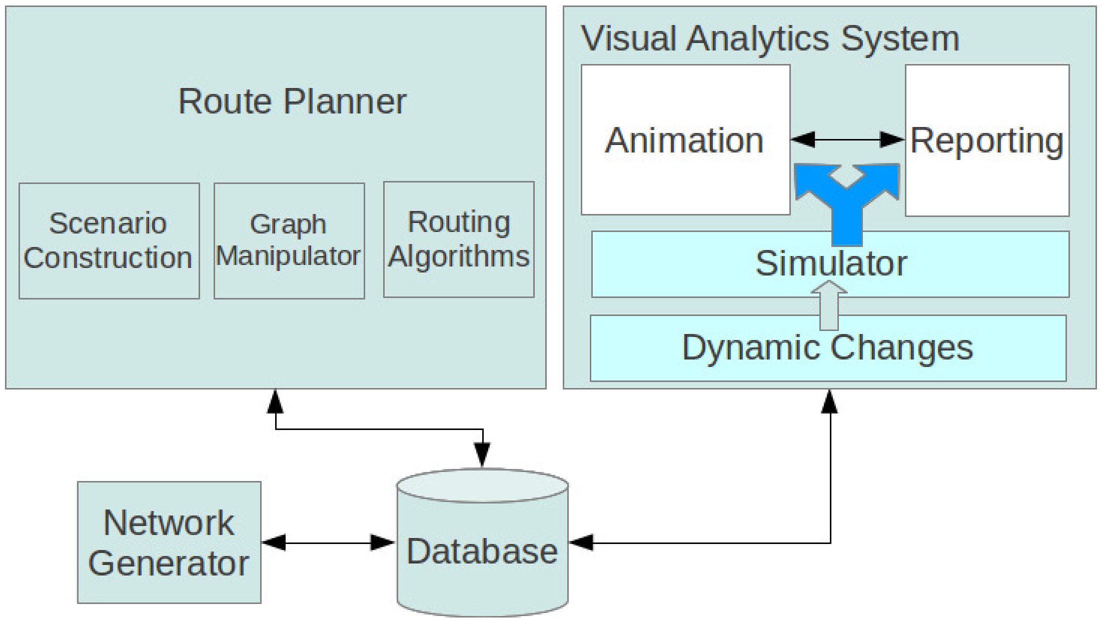


Figure 3.1: Visual Analytic System Architecture. Route planner involves preprocessing and routing calculations that serve to initialize the system, with processed results stored in a database. The visual analytic system uses visual abstraction and scalable representations that permit real-time interaction, injection of dynamic situational changes and visual analysis.

In our earlier work [13] we described a semi-automatic system that constructed a *building graph*, incorporating key elements of a geo-referenced urban structure critical to evacuations, such as hallways, stairways, elevators and entrances/exits. This building graph generator is used to process urban structures and is stored in a PostgreSQL database. We have successfully processed over 70 buildings of our campus using the graph generator.

We next describe the main components of our visual analytic system.

3.1 ROUTE PLANNER

Our route planner provides facilities for loading evacuation objects, that are combinations of urban structures, pathways, streets etc. Evacuations are built as combinations of structural objects and route planner objects and saved in the database. The Route Planner consists of the following components:

3.1.1 LOD GRAPH CONSTRUCTION/MANIPULATION

All computed egress paths are saved in the database as node to node connections. Nodes are defined based on their function, and can be rooms, hallways, stairways, elevators, and exits. The evacuation application also creates muster points, that represent locations where evacuees are ordered to congregate during an emergency. During the process of extracting centerline points the geometry of hallways is sampled at approximately two foot intervals. This process is necessary to insure accuracy, particularly at corridor elbows. However, this raises computational issues with extremely large buildings or multi-building graphs, and hinders real-time performance(expensive routing calculations) when dynamic changes need to be accommodated during an event. We address this problem by simplifying the graph with two different levels of detail.

Level 1. Remove Redundant Nodes. In this step, we simplify the original building graph by removing nodes that do not impact routing, for instance, shortest path calculations. In particular, the larger sampling rate used for accurate centerline calculations results in nodes that can be removed for use in building routes. Nodes and edges are collapsed in the process. Beginning with any node with three or more edges (or any arbitrary node), edges are followed until a node with 3 or more edges is encountered. This becomes a node of the simplified graph and a new edge is created connecting to the previous node, as can be seen in Figure 3.4. The process is continued until all nodes have been visited. Our route planner is executed on these simplified graphs for scenarios in which all original egress paths are available. In our experiments, we see a factor of 5-8 reduction in the number of nodes in the simplified graph.

The Level 1 graph reduction process attempts to create a bi-partite graph where all inner building nodes have 3 or more incident edges. The final graph produced may have nodes with 2 incidents edges because unconnected nodes remaining after algorithm execution are simply connected to their closest neighbor with no further concern for reduction. Results of processing dozens of building graphs indicates approximately 95 pct. reduction of the nodes of concern to 3 edge nodes. The following algorithms detail the process.

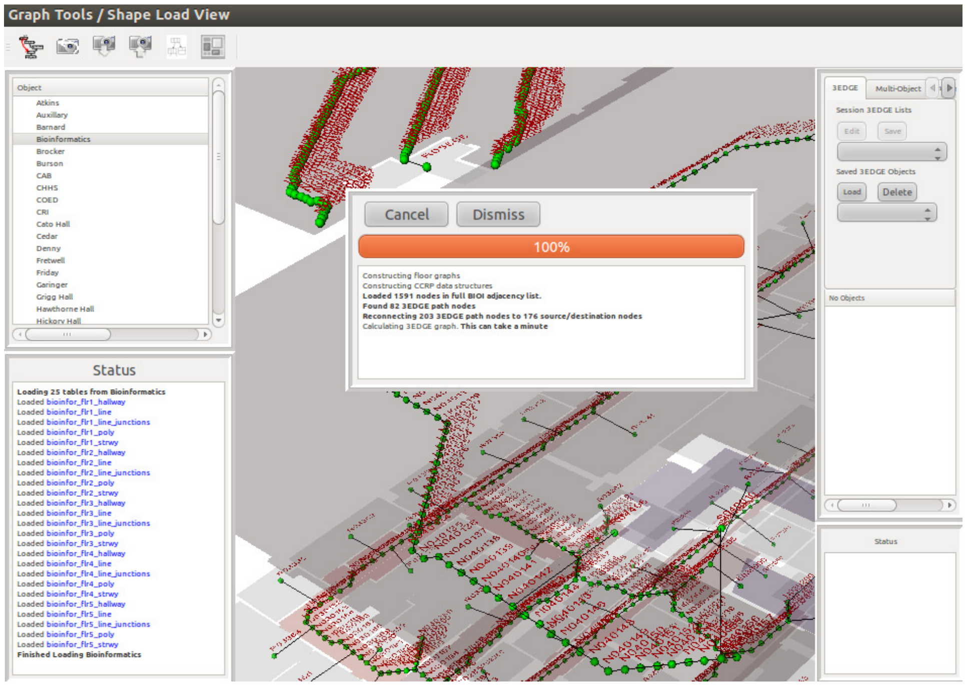
The output of the network generator produces 2D arrays stored in the PostGIS database in the form Node1, Node2, Distance in feet. For efficient manipulation we store these in adjacency lists implemented as associative arrays.

The spG structure produced by algorithm 1 serves as input to a 3 stage divide and conquer algorithm(2, 3, 4). The procedures perform the following general tasks:

- Find all nodes with 3 or more incident edges in original graph and add them to the minimal graph.
- Build a heap of all hallway nodes with less than 3 incident edges.
- Build a heap of all non hallway nodes with less than 3 incident edges.

Algorithm 1: Build an Adjacency List Graph of the building containing all Network Generator Nodes and produce Incident Edge Count Associative Array. Reference Appendix A 8 for more detailed pseudo code.

```
Input: (1) Node Properties: nIE, Count of Incident edges per node;
(2) Edge Properties: Length in Feet(LF);
(3) EAr: Array; [$Node1_{i...n}, Node2_{j...n}, LF$]]. Array list of all nodes in spG.
Result: (1) spG(N,E): Directed Graph, N nodes with 1 or more Incident
        Edges(IE), E edges. Note: This structure is an Adjacency List
        $\{N_i : \{N_j : LF, ...N_k : LF\}\}\}$;
(2) Node Properties: nIE, Count of Incident Edges per node;
1  foreach Node1; Node2; LF in EAr do
2      Initialize empty associative array Temp;
3      if Node1 not in Adjacency List spG Add It then
4          Add edge Node1-Node2-LF to Adjacency List Temp;
5          for k = IndexCounter + 1 to LENGTH(EAr) - 1) do
6              Add edge Node1-Node2-LF at EAr[k] to Temp if Node1 and Node2
               are HALLWAY-NODE then
7                  nIE[Node1] += 1;
                         /* Increment edge count for Node1 */;
8              end
9          end
10         spG[Node1] = Temp;
                 /* Assign associative array Temp to Node1 */;
11     end
12 end
```



```
Loading 25 tables from Bioinformatics
Loaded bioinfor_flr1_hallway
Loaded bioinfor_flr1_line
Loaded bioinfor_flr1_line_junctions
Loaded bioinfor_flr1_poly
Loaded bioinfor_flr1_strwy
Loaded bioinfor_flr2_hallway
Loaded bioinfor_flr2_line
Loaded bioinfor_flr2_line_junctions
Loaded bioinfor_flr2_poly
Loaded bioinfor_flr2_strwy
Loaded bioinfor_flr3_hallway
Loaded bioinfor_flr3_line
Loaded bioinfor_flr3_line_junctions
Loaded bioinfor_flr3_poly
Loaded bioinfor_flr3_strwy
Loaded bioinfor_flr4_hallway
Loaded bioinfor_flr4_line
Loaded bioinfor_flr4_line_junctions
Loaded bioinfor_flr4_poly
Loaded bioinfor_flr4_strwy
Loaded bioinfor_flr5_hallway
Loaded bioinfor_flr5_line
Loaded bioinfor_flr5_line_junctions
Loaded bioinfor_flr5_poly
Loaded bioinfor_flr5_strwy
Finished Loading Bioinformatics
```

```
Constructing floor graphs
Constructing CCRP data structures
Loaded 1591 nodes in full BIOI adjacency list.
Found 82 3EDGE path nodes
Reconnecting 203 3EDGE path nodes to 176 source/destination nodes
Calculating 3EDGE graph. This can take a minute
```

Figure 3.2: User interface for control and monitoring Reduction Algorithm. In this example 1591 nodes are being reduced to 203.

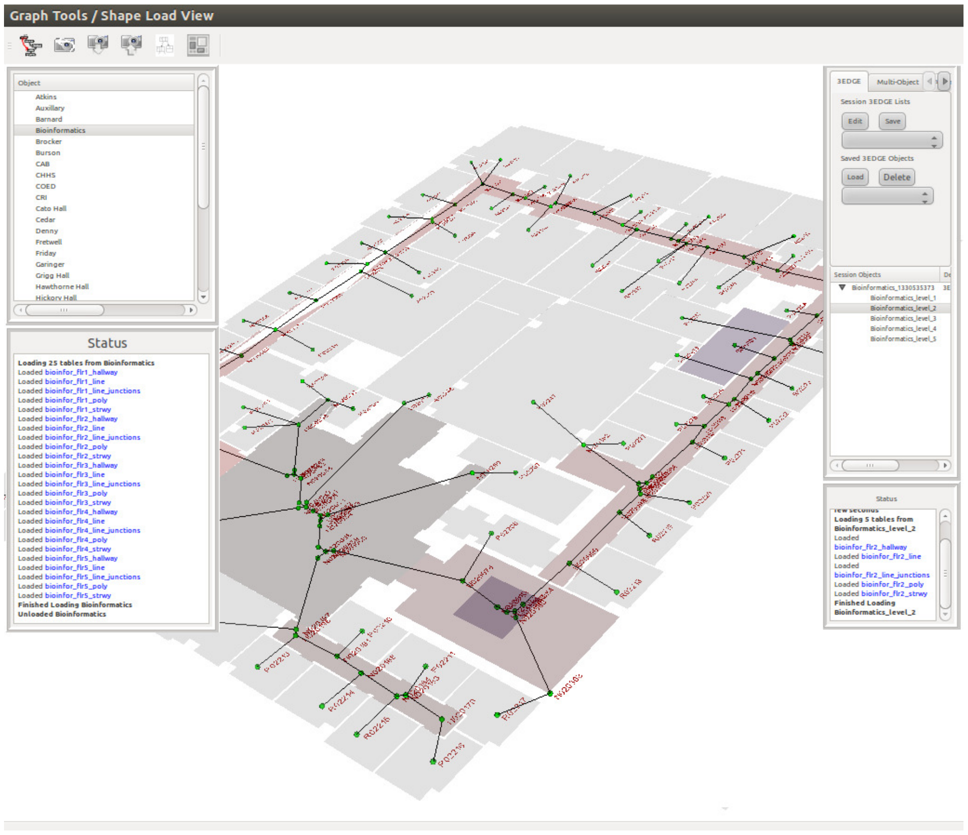
- Iterate hallway nodes combining edges between nodes until leaf node of a "3EDGE" node is found then connect them as one edge.

- Iterate NON hallway nodes combining edges between nodes until leaf node of a "3EDGE" node is found then connect them as one edge.

Table 3.1: Campus Adjacency List

ID(Int)	NodeID(Char)	AdjacencyList(VarChar)
1294290388	N010011_BIOI	"N010007_BIOI 4 N010024_BIOI 13"
1294290388	N010024_BIOI	"N010011_BIOI 13 N010025_BIOI 1"
1294290388	N010025_BIOI	"N010026_BIOI 1 N010189_BIOI 1 N010024_BIOI 1"
1294290388	N010026_BIOI	"N010043_BIOI 17 N010025_BIOI 1"

After the entire series of operations is complete the Associative Array spMinG, which is finalized in Algorithm 4, is used by the Route Planner to improve the efficiency of Single Source Shortest Path Dijkstra searches. It is also used in the Simulator for reconstruction of a real geometric path when a Zone graph is input to a Dijkstra



```
Loading 25 tables from Bioinformatics
Loaded bioinfor_flr1_hallway
Loaded bioinfor_flr1_line
Loaded bioinfor_flr1_line_junctions
Loaded bioinfor_flr1_poly
Loaded bioinfor_flr1_strwy
Loaded bioinfor_flr2_hallway
Loaded bioinfor_flr2_line
Loaded bioinfor_flr2_line_junctions
Loaded bioinfor_flr2_poly
Loaded bioinfor_flr2_strwy
Loaded bioinfor_flr3_hallway
Loaded bioinfor_flr3_line
Loaded bioinfor_flr3_line_junctions
Loaded bioinfor_flr3_poly
Loaded bioinfor_flr3_strwy
Loaded bioinfor_flr4_hallway
Loaded bioinfor_flr4_line
Loaded bioinfor_flr4_line_junctions
Loaded bioinfor_flr4_poly
Loaded bioinfor_flr4_strwy
Loaded bioinfor_flr5_hallway
Loaded bioinfor_flr5_line
Loaded bioinfor_flr5_line_junctions
Loaded bioinfor_flr5_poly
Loaded bioinfor_flr5_strwy
Finished Loading Bioinformatics
Unloaded Bioinformatics
```

```
few seconds
Loading 5 tables from
Bioinformatics_level_2
Loaded
bioinfor_flr2_hallway
Loaded bioinfor_flr2_line
Loaded
bioinfor_flr2_line_junctions
Loaded bioinfor_flr2_poly
Loaded bioinfor_flr2_strwy
Finished Loading
Bioinformatics_level_2
```

Figure 3.3: View of 2nd floor results from operation illustrated in Figure3.2.

Search. Using Zone calculated routes for visualization causes evacuees to appear to take impossible routes. Unless a building is modified the structure is permanent through the life of the application therefore it is save to the database in relation table campus_adjacency_lists. The table allows multiple lists per campus object but there should only be a need for one unless other algorithms are implemented that require a modified or different underlying structure for searches. Table 3.1 is an example representation of the campus_adjacency_list entity:

Algorithm 2 adds nodes with 3 or more incident edges to the reduced graph adjacency list spMinG. This initializes the reduced graph for processing through the later functions. At this point it contains 3 or more edge nodes that are disconnected and is not yet a complete graph. Notice that the a subordinate adjacency lists nodes are copied into heaps. There are 2 heaps utilized to accomplish the divide and conquer

Algorithm 2: Build a Partial Reduced Adjacency List Graph of unconnected node heaps of "HALLWAY" type nodes and "OTHER" type nodes. Reference Appendix A 9 for more detailed pseudo code.

```
Input: (1) spG(N,E): Directed Graph, N nodes with 1 or more Incident
       Edges(IE), E edges. Note: This structure is an Adjacency List
       $\{N_i : [N_j : LF, ...N_k : LF]]\}$;
(2) Node Properties: nIE, Count of Incident edges per nodes;
Result: (1) spMinG(N,E): Directed Graph, N nodes with 3 or more Incident
        Edges(IE), E edges as an Adjacency List $\{N_i : [N_j : LF, ...N_k : LF]]\}$;
(2) NoConnHeapPATH: Heap of path nodes with $<$ 3 Incident Edges;
(3) NoConnHeapOTHER: Heap of path nodes with $<$ 3 Incident Edges
connected to Stairwells or Exits, etc;
1  for Node in original Adjacency List spG do
2      if Node has 3 or more incident edges then
3          $spMinG[node]$ = spG[node];
                   /* Work done...add it to reduced graph */;
                   /* Node2's refer to 2nd node in Node1-Node2-LF edge
           definition */;
4          for Hallway nodes assume all Node2's are disconnected and build array
           list of these nodes do
5              Add Node2's to $NoConnHeapPATH$ Array;
6          end
7      end
8      else if NOT HALLWAY-NODE then
9          $spMinG[node]$ = spG[node];
               /* add EXIT, STAIRWELL etc, nodes to reduced graph */;
10         for NON hallway nodes assume all Node2's are disconnected and build
           array list of these nodes do
11             Add Node2's to $NoConnHeapOTHER$ Array;
12         end
13     end
14 end
```

methods. A no connection heap built from hallway nodes and a no connection heap built from non hallway nodes. The 2 categories of node heaps are created because we always know we want to connect hallway nodes adjacent to leaf nodes of 3 or more edge nodes. Nodes that are categorized as stairwells or exits for example are connected after the hallway iterations are complete. This allows for different handling in of central hallway nodes and other nodes and simplifies the logic of each segment of our procedure.

Algorithm 3 iterates the hallway nodes no connection heap. The start point is arbitrary. This is an important point because due to the complexities of real world building geometry there is no standard definition of a start point for this iteration. The procedure steps to this point only guarantee that the origin will be a leaf node of a 3 or more incident edge node and we are looking for the next one that is connected by multiple edges to another node in spG(the original graph) built in Alg. 1. Here we select a node from the heap which is known to be a leaf node of an edge in spMinG then walk through adjacency list spG until we find another node in the both heaps that we know are in spMinG and then connect them and add them to the spMinG adjacency list graph.

Algorithm 4 iterates the non hallway nodes no connection heap. The same procedures are used as in Algorithm 3 the only difference being that if an origin has not yet been added to the spMinG is a dead end and must be added to insure no dangling nodes. This happens when a node is at the end of a building section or hallway and is connected to a source node. Source nodes are simply connected to there associated hallway node before database insertion. It is not necessary to process them through the graph reduction procedures. Note that a default path length equal to the maximum occupancy for an edge is used. This insures the path from a source to an egress can contain the maximum set by the application.

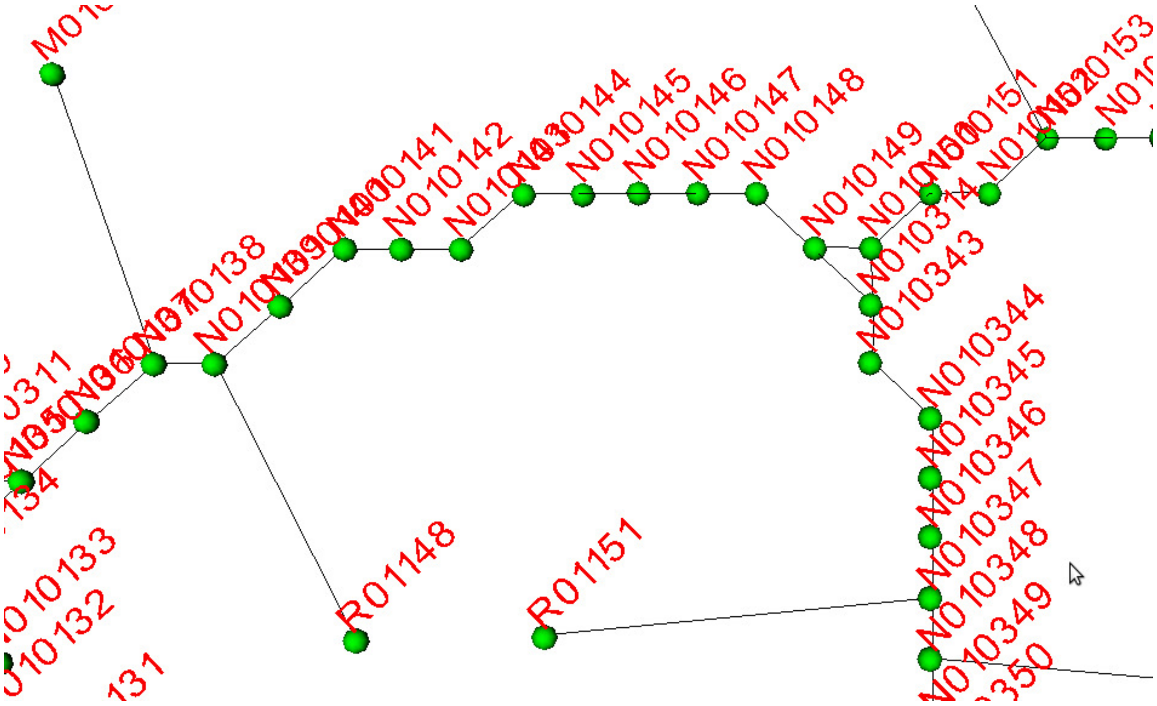
Level 2. Zone Graphs. For rapid computation of paths when dynamic changes are

Algorithm 3: Build partial reduced adjacency List graph by walking original graph adjacency list until node is found in No Connection Heap of "HALLWAY" nodes. Reference Appendix A 10 for more detailed pseudo code.

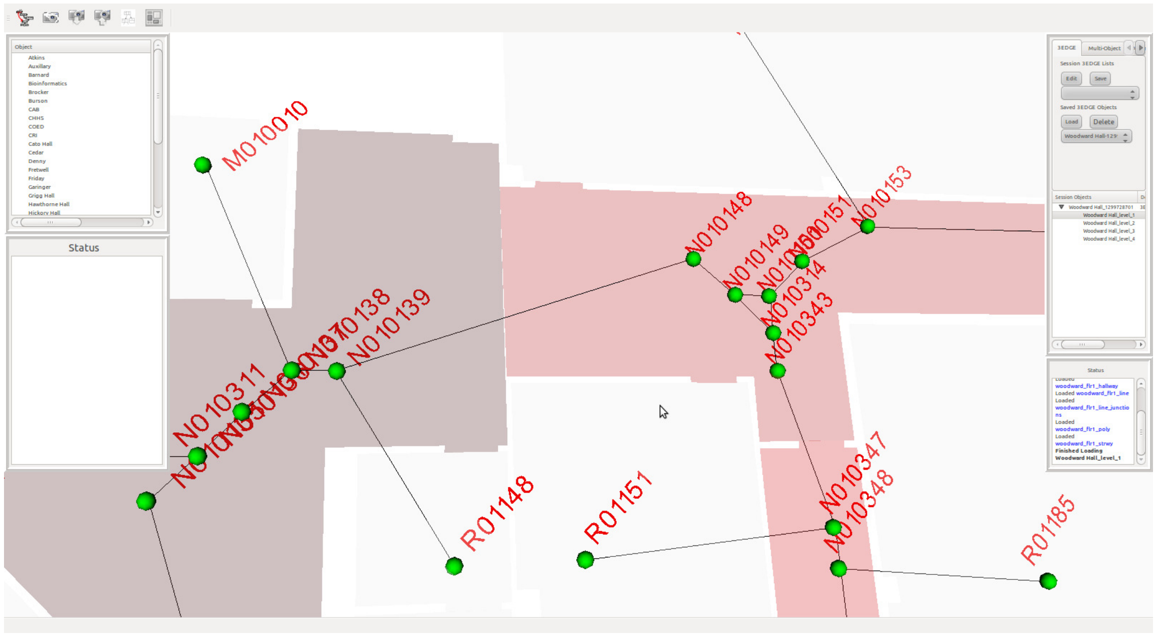
```
Input: (1) spG(N,E): Directed Graph, N nodes with 1 or more Incident
       Edges(IE), E edges. Note: This structure is an Adjacency List
       ${N_i : [N_j : LF, ...N_k : LF]]}$;
(2) Node Properties: nIE, Count of Incedent edges per nodes;
(3) spMinG(N,E): Directed Graph, N nodes with 3 or more Incedent
Edges(IE), E edges as an Adjacency List ${N_i : [N_j : LF, ...N_k : LF]]}$;
(4) NoConnHeapPATH: Heap of path nodes with $<$ 3 Incident Edges;
(5) NoConnHeapOTHER: Heap of path nodes with $<$ 3 Incident Edges
connected to Stairwells or Exits;
Result: spMinG(N,E): Directed Graph, N nodes with Maximum Incident
        Edges(IE) and Minimum Nodes, E edges as an Adjacency List
        ${N_i : [N_j : LF, ...N_k : LF]]}$;
1  foreach Node in NoConnHeapPATH do
2      Get an adjacent node next of Node in original Graph spG;
3      while keeplooking do
4          if next is in No Connection Heaps then
                   /* Add edge between Node in Heap and Leaf Node in
               original Graph spG */;
5              Add Node to Reduced Graph Adjacency List;
6              keeplooking = false;
                /* Found a connection move to next node in No Connection
               Heap of HALLWAY Nodes */;
7          end
8          else
                 /* Keep walking through NON reduced Adjacency Graph spG
               */;
9              pathlength = pathlength + Length between Node and Next;
10             next = Get an adjacent node to next in spG;
11         end
12     end
13 end
```

Algorithm 4: Build **Final** Reduced Adjacency List Graph by iterating No Connection heap built from NON HALLWAY Nodes. Reference Appendix A 11 for more detailed pseudo code.

```
Input: (1) spG(N,E): Directed Graph, N nodes with 1 or more Incident
       Edges(IE), E edges. Note: This structure is an Adjacency List
       $\{N_i : [N_j : LF, ...N_k : LF]]\}$;
(2) Node Properties: nIE, Count of Incident edges per nodes;
(3) spMinG(N,E): Directed Graph, N nodes with 3 or more Incident
Edges(IE), E edges as an Adjacency List $\{N_i : [N_j : LF, ...N_k : LF]]\}$;
(4) NoConnHeapPATH: Heap of path nodes with $<$ 3 Incident Edges;
(5) NoConnHeapOTHER: Heap of path nodes with $<$ 3 Incident Edges
connected to Stairwells or Exits;
Result: (1) spMinG(N,E): Directed Graph, N nodes with Maximum Incident
        Edges(IE)and Minimum Nodes, E edges as an Adjacency List
        $\{N_i : [N_j : LF, ...N_k : LF]]\}$;
1  foreach Node in NoConnHeapOTHER do
2      Get an adjacent node next of Node in original Graph spG;
3      while keeplooking do
4          if next is in No Connection Heaps then
               /* Add edge between Node in Heap and Leaf Node in
               original Graph spG */;
5              Add Node to Reduced Graph Adjacency List;
6              keeplooking = false;
               /* Found a connection move to next node in No Connection
               Heap of HALLWAY Nodes */;
7          end
8          else
               /* Keep walking through NON reduced Adjacency Graph spG
               */;
9              pathlength = pathlength + Length between Node and Next;
10             next = Get an adjacent node to next in spG;
11         end
12     end
13 end
```



(a)



(b)

Figure 3.4: Building graph Simplification. Removing redundant nodes. (a) a section of a building floor with labeled building elements, (b) simplified graph.

injected during an event, the simplified graphs can still be large, especially in multi-building evacuations. In these circumstances, we further simplify the building graphs into *zone graphs*, by segmenting the building into evacuation sensitive zones: for instance, stairwells, elevators and exits form the critical elements of any egress path. An example zone graph is illustrated in Figure 3.6, involving stairwells, hallways and exits(in red)

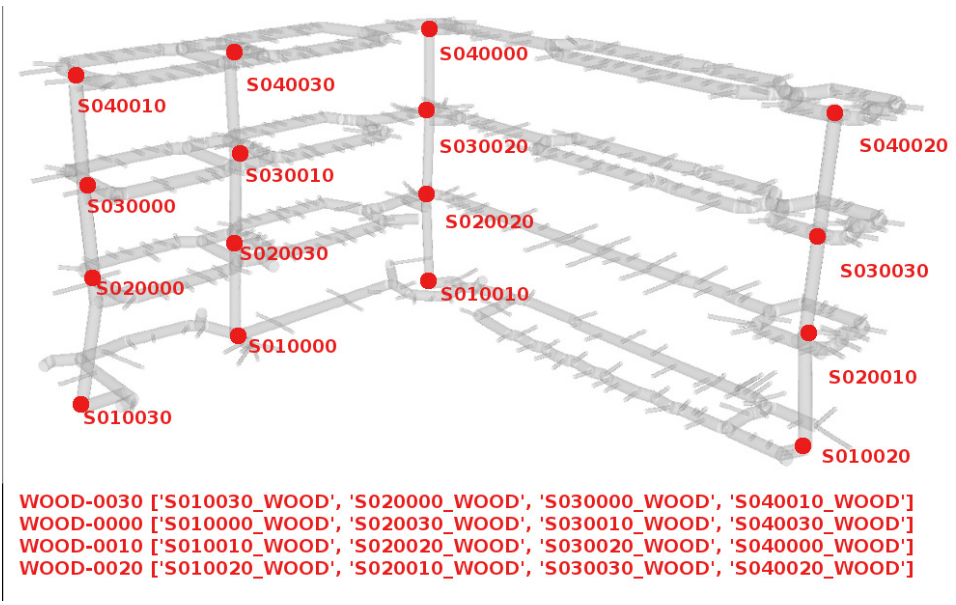
Zone Graph Construction Overview: To compute the zone graph, we use the precomputed evacuee paths to first associate each node with a zone that is closest to it, using an iterative procedure. In the second pass, the graph connectivity is established by keeping track of the zone of related objects that are encountered in these paths (adjacency lists are maintained). Intermediate nodes(the yellow spheres in Figure 3.6(b)) are also identified by paths that cross multiple zones and are further used to complete the graph construction. Zone objects span floors in multi-story structures, with appropriate floor identification for proper path determination during routing calculations. The number of nodes in the resulting zone graph depends on the number of zones and the number of floors in the building. In our experiments, a further factor of 5-7 reduction in the number of graph nodes was seen. Table 3.2 compares reroute times based on these reductions.

Table 3.2: Rerouting comparisons(full,reduced,zone) in seconds.

Evacuees	Full	Reduced	Zone
1350	98	4	.18
2885	120	6	.22
5130	330	11	.43
7264	460	14	.6

Zone graphs are saved in the database as adjacency lists and imported into responder processes for fast situationally aware rerouting. They are also used in building the reporting tools.

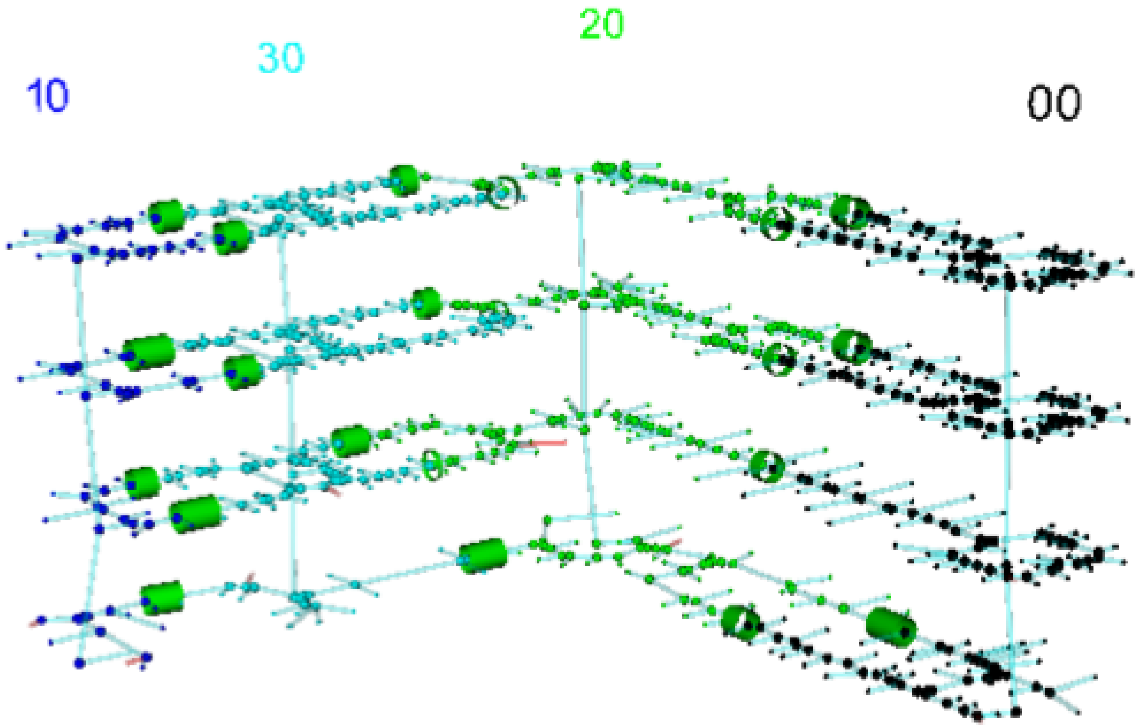
Zone Graph Construction Detail. The most significant factor here is the problem



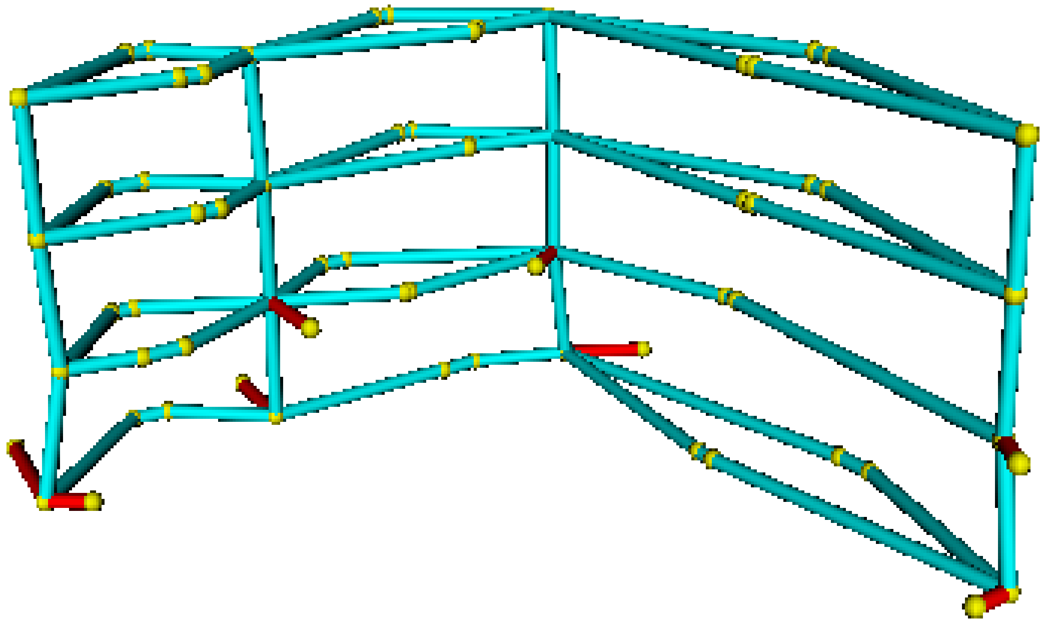
```
WOOD-0030 ['S010030_WOOD', 'S020000_WOOD', 'S030000_WOOD', 'S040010_WOOD']
WOOD-0000 ['S010000_WOOD', 'S020030_WOOD', 'S030010_WOOD', 'S040030_WOOD']
WOOD-0010 ['S010010_WOOD', 'S020020_WOOD', 'S030020_WOOD', 'S040000_WOOD']
WOOD-0020 ['S010020_WOOD', 'S020010_WOOD', 'S030030_WOOD', 'S040020_WOOD']
```

Figure 3.5: Backbone with associative data structure to form key nodes in the building Zone Graph.

set itself. We are manipulating connectivity graphs for building egress. If we are to construct subgraphs of the overall graph of a building we need to define what constitutes a node and an edge in the subgraph. We chose the vertical edges(stairwells) as the backbone of our zone network. Figure 3.5 shows this concept visually. The building has 4 vertical stairwells which create 16 backbone nodes that will serve as the basis for construction of zone graphs. In single floor buildings the backbone will be formed by exits only. Exits are part of the multiple floor building graphs also but will be added later. The associative array structure in Figure 3.5 is built as edges are loaded from the campus_adjacency_list table before a route plan is built. The next step is to identify the nodes in the overall building graph that are associated with each stairwell node and the extents(outer most nodes) for each zone. This is done in the Route Planner after a plan has been constructed. Algorithm 5 builds 2 associative arrays that define all the nodes in each zone and the points where 1 zone ends and another zone begins. ZoneNodes and ZoneExtents respectively. Figure 3.7 expands on Figure 3.5 by showing these zone subset structures colorized with all their



(a)



(b)

Figure 3.6: Building graph simplification to a zone graph. (a) building graph of a campus building, (b) transformation to zone graph representation. Yellow spheres are nodes and cyan tubes represent edges. The red tubes represent paths to exits

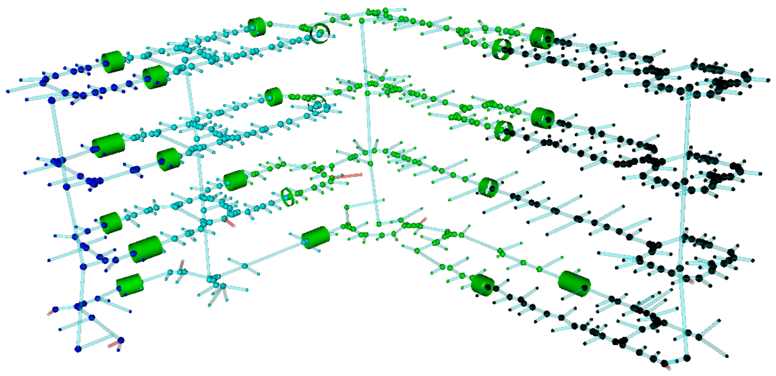


Figure 3.7: Backbone with associated nodes colorized by stairwell. The large green tubes show the location of extents.

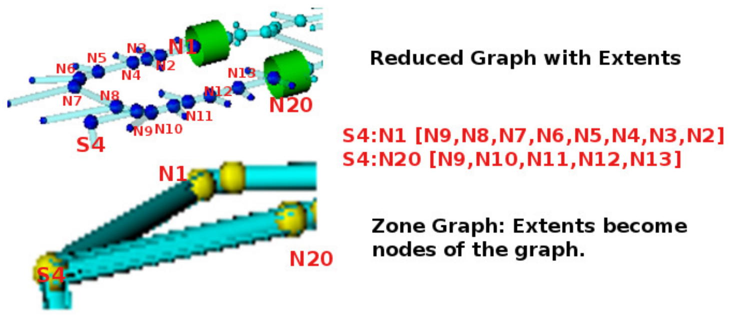
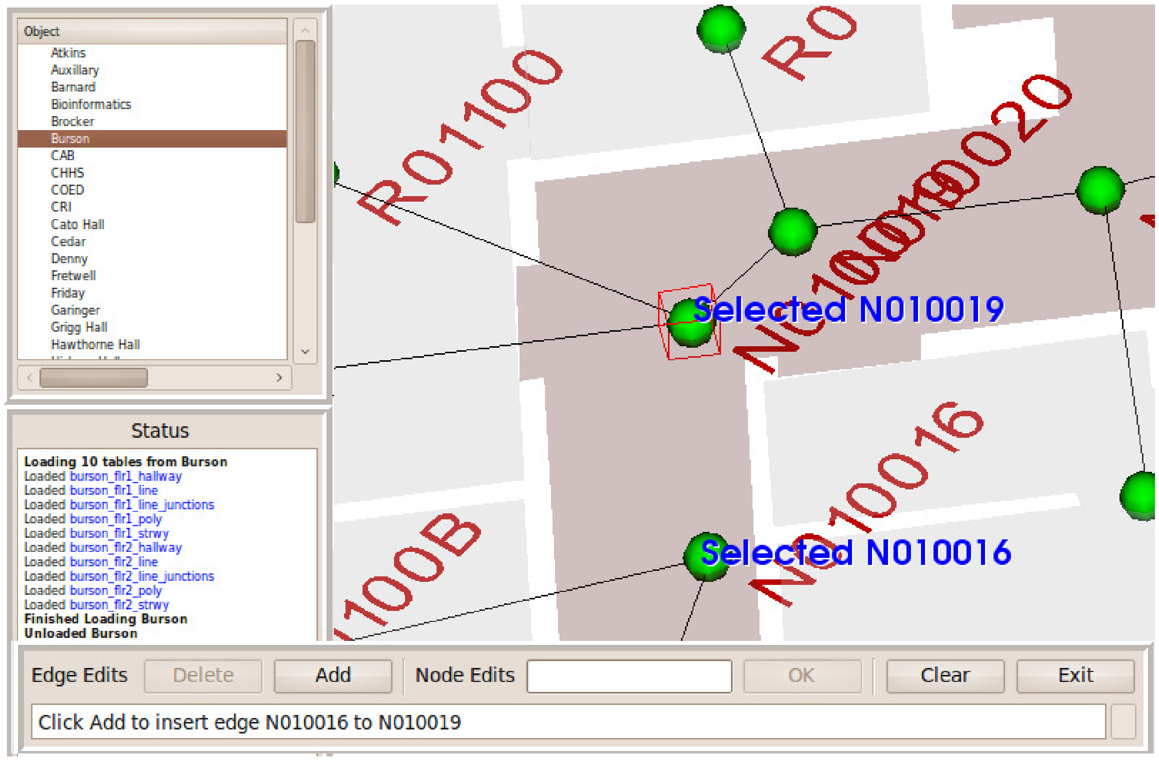

Reduced Graph with Extents
N1
N2
N3
N4
N5
N6
N7
N8
N9
N10
N11
N12
N13
N20
S4
S4:N1 [N9,N8,N7,N6,N5,N4,N3,N2]
S4:N20 [N9,N10,N11,N12,N13]
N1
N20
S4
Zone Graph: Extents become nodes of the graph.

Figure 3.8: When a zone graph is used for rerouting in the Simulator the results of the shortest path search return a geometric path that does not resemble that of the building. The evacuees are assigned the zone nodes and the nodes between them for a more realistic visualization.

associated nodes. Extents are important to building an adjacency list Zone Graph. They define where zones will be connected by edges and they serve as pointers to the nodes between them in real-time reroute mode. The final step is to add exits to the zone graphs and connect them directly to stairwells if available, or otherwise to the closest hallway node.

Algorithm 5: Associate Overall Graph Hallway Nodes with a Stairwell Zone and find Extents for each Zone.

```
Input: (1) ZoneBackbone: Array See Figure 3.5;
(2) EvacueeRoute: Array [EvacueeID, Path[0...n]];
(3) AllEdgeArray: Array of all edges in Reduced Building Graph: [Node1,
Node2, LF]
Result: (1) ZoneNodes: Associative Array: [ZoneID, Nodes[0...n]];
(2) ZoneExtents: Associative Array [ZoneID, Nodes[0...x]];
1  for EvacueeID j in EvacueeRoute do
2      Path = EvacueeRoute[j];
3      for i = LENGTH(Path) - 1 to 1 do
                                 /* Start at exit */;
4          if Path[i] is STAIRWELL_NODE then
5              ZoneID = FindZoneID(ZoneBackbone);
6              Add Node Path[i] to ZoneNodes[ZoneID]; Move to next
               EvacueeID;
7          end
8      end
9  end
                          /* Find Zone Extents */;
10 for Node1 N1, Node2 N2 in AllEdgeArray do
11     Z1 = GetZoneID(N1);
12     Z2 = GetZoneID(N2);
13     if Z1 not equal Z2 then
                 /* Nodes in different zones.  Found Extents */;
14         Add Node N1 to ZoneExtents[Z1];
15         Add Node N2 to ZoneExtents[Z2];
16     end
17 end
```



```
Loading 10 tables from Burson
Loaded burson_flr1_hallway
Loaded burson_flr1_line
Loaded burson_flr1_line_junctions
Loaded burson_flr1_poly
Loaded burson_flr1_strwy
Loaded burson_flr2_hallway
Loaded burson_flr2_line
Loaded burson_flr2_line_junctions
Loaded burson_flr2_poly
Loaded burson_flr2_strwy
Finished Loading Burson
Unloaded Burson
```

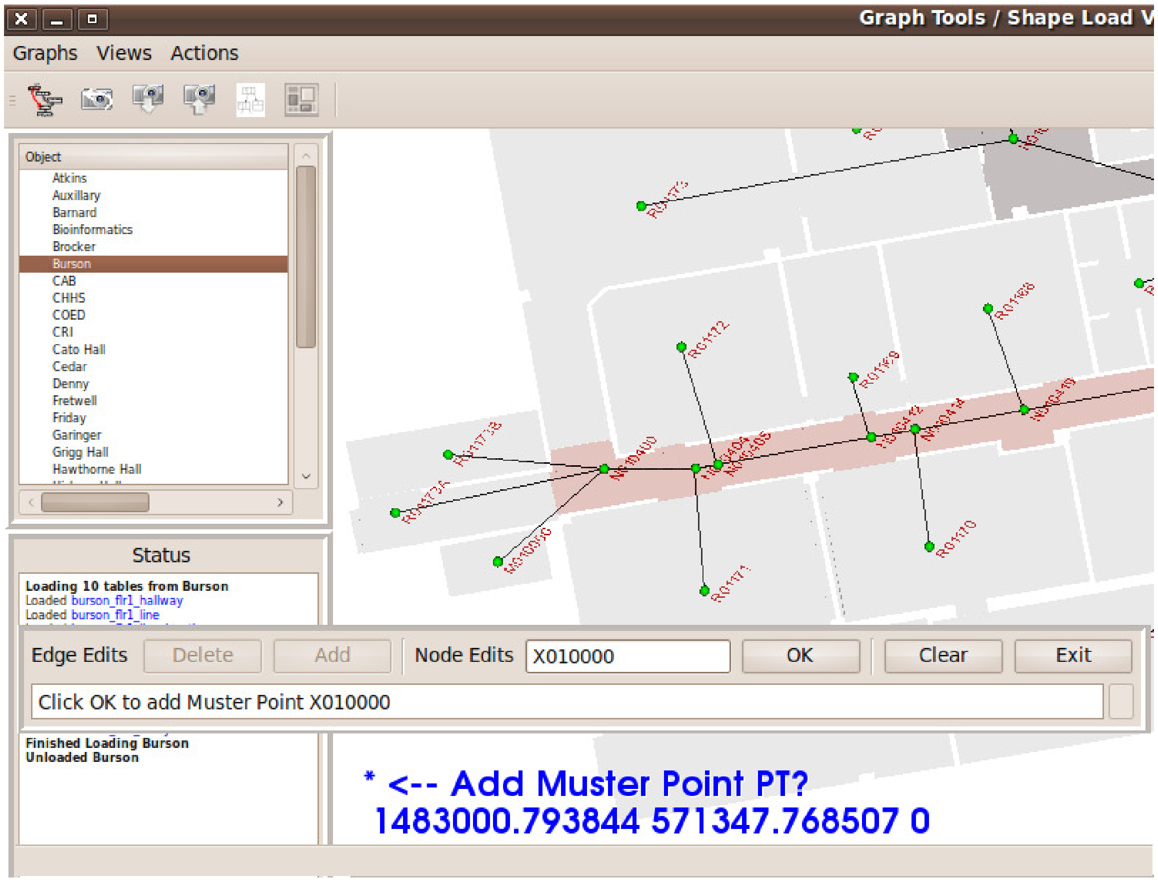
Click Add to insert edge N010016 to N010019

Figure 3.9: User adds an edge. Notice that there are also functions for deleting edges and adding and deleting nodes.

3.1.2 MANUAL GRAPH MANIPULATION

There are several steps required to create an efficient graph for evacuation evaluations. Our process, from initial data acquisition to the final graph includes, CAD conversion, geo-referencing, centerline geometry calculations, PostGIS SQL conversion, and then the processes described above. The process is robust and quality control is done at every step, however in producing the final graphs here approximately 2-3 nodes per building are left disconnected or missing.

Rather than assuming that this is always produced in the graph manipulation steps here we have included a manual function to visually inspect graphs and provide another step in quality control provided by the user. Figure 3.9 shows and inspection of the output of the route manipulation algorithms above. An edge is missing between a "3EDGE" node N010019 and hallway node N010016. In this particular case this is



Loading 10 tables from Burson
Loaded burson_flr1_hallway
Loaded burson_flr1_line

Node Edits: X010000

Click OK to add Muster Point X010000

Finished Loading Burson
Unloaded Burson

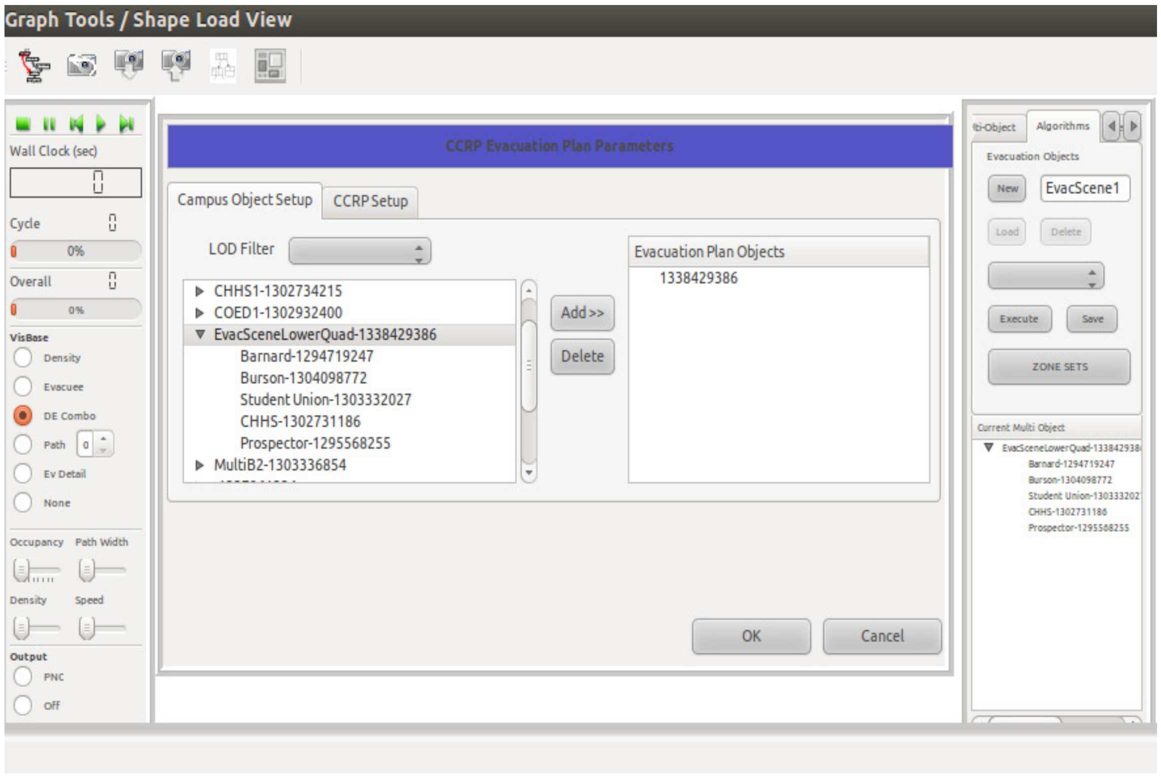
* <-- Add Muster Point PT?
1483000.793844 571347.768507 0

Figure 3.10: User adds a Muster Point. X010000 is automatically associated with M010000, visible above the user widget.

a disconnection artifact or Algorithm 4. A user clicks the ”Add” button and an edge is inserted. Another manual function of the graph manipulation process is to build a default set of outdoor muster points associated with each exit. Currently mustering is a mater of campus procedure but locations are not identified for geo-referencing. We add these manually in our application.

3.1.3 SCENARIO CONSTRUCTION

Scenario construction includes loading single or multi-building evacuation objects, selecting a route planning algorithm (currently limited to our modified capacity constrained route planner[15], and setting visualization modes and evacuation parameters, such as building capacity, egress width, evacuee speed and density, and stairway



- CHHS1-1302734215
- COED1-1302932400
- EvacSceneLowerQuad-1338429386
  - Barnard-1294719247
  - Burson-1304098772
  - Student Union-1303332027
  - CHHS-1302731186
  - Prospector-1295568255
- MultiB2-1303336854

Evacuation Plan Objects: 1338429386

Evacuation Objects: EvacScene1

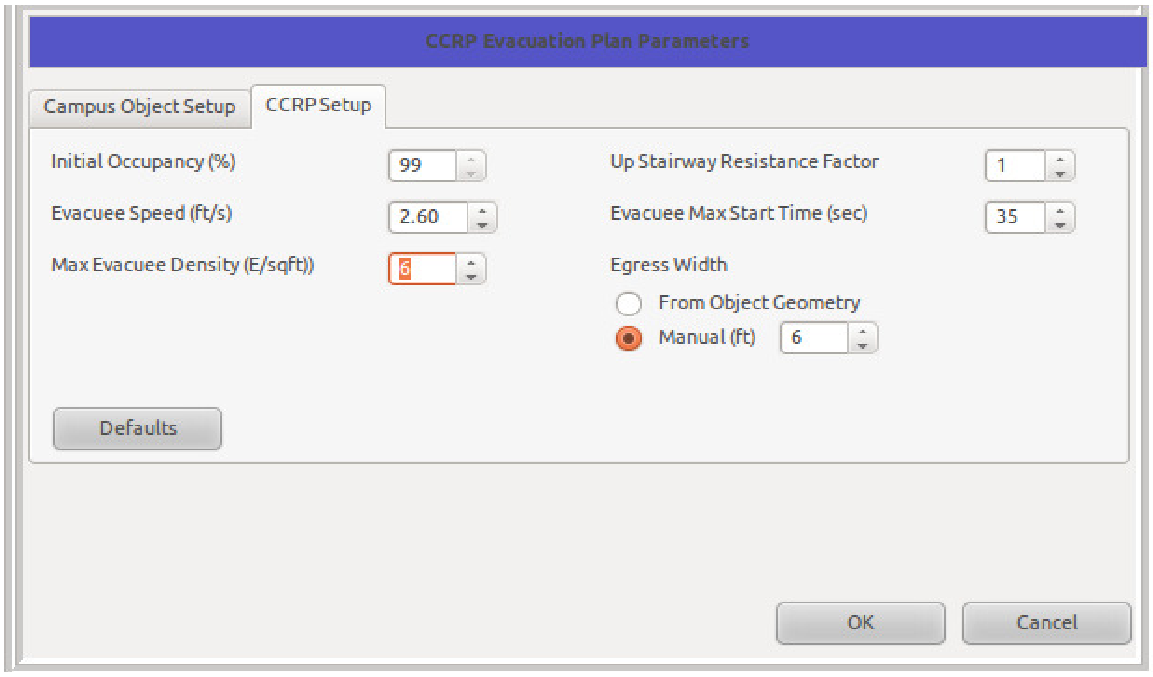
Figure 3.11: User interface for evacuation object selection.

up resistance.

When an evacuation scenario is built the objects in the evacuation are combined into a structure that includes all the attributes of each character in the event. Evacuees are loaded and placed in commonly occupied areas. Pointers are assigned to the egress graph to be used for routing. The graphs are saved as ”3EDGE” or Zone graphs in the database. Note however, a zone graph is not saved until at least one a-priori execution of the Routing Algorithm for an evacuation object has been executed.

3.1.4 ROUTING ALGORITHMS

We have implemented a modified version of the Capacity Constrained Route Planner(CCRP)[15], which is illustrated in Algorithm 6. Given a directed graph with node and edge capacities, the algorithm repeatedly computes the shortest path for each evacuee with available capacity. If a path is found, then it assigns as many evacuees as possible through that path, i.e., until the capacity of any of node or edge



Initial Occupancy (%) 99
Evacuee Speed (ft/s) 2.60
Max Evacuee Density (E/sqft)) 6
Up Stairway Resistance Factor 1
Evacuee Max Start Time (sec) 35
Egress Width
From Object Geometry [✓ Manual (ft)] 6

Figure 3.12: CCRP parameters setup.

along the path is exceeded. This is followed by moving all the evacuees at that time step. The process repeats until all evacuees have found paths to exit the structure. The final step is to evacuate the remaining evacuees in the building(who already have paths, but not exited the building).

We have augmented the CCRP algorithm by specifying the movement of the evacuees(in addition to finding the paths) at each iteration. Secondly, the algorithm is modified to work with our simplified graphs; in the simplified graphs, weights of the collapsed edges are accumulated and assigned to the new simplified edges. Running the routing algorithm on the simplified graphs makes it more scalable to larger urban structures as well as facilitating dynamic changes to the graph that will require rerouting occupants around blockages or other hazards caused by the emergency event. Finally, although each evacuee has a set average speed (3 ft. per second), evacuees cannot exceed the set density threshold. Thus, as congestion builds up, evacuee movements are naturally slowed down. Additional data structures are maintained to make these computations efficient.

Algorithm 6: Modified Capacity Based Route Planner

```
   Input: ;
   (1) G(N,E): Directed Graph, N nodes, E edges;
   (2) Node Properties: capacity, occupancy;
   (3) Edge Properties: capacity, length in feet;
   (4) Set of Source Nodes;
   (5) Set of Destination Nodes;
   (6) Set of evacuee objects;
   Result: Evacuation Plan : Routes with schedules of evacuees on each route
1  foreach evacuee i at each source node s do
2      path_found = find shortest path $p$ from $s$ to all destinations with
       available capacity;
3      if path_found then
4          while $p <$ max_capacity do
               /* can route evacuee via p */ evacuees[i].path = p;
5          end
6      end
7      move all evacuees;
8  end
9  while evacuees not at destination nodes do
10     move all evacuees;
11 end
```

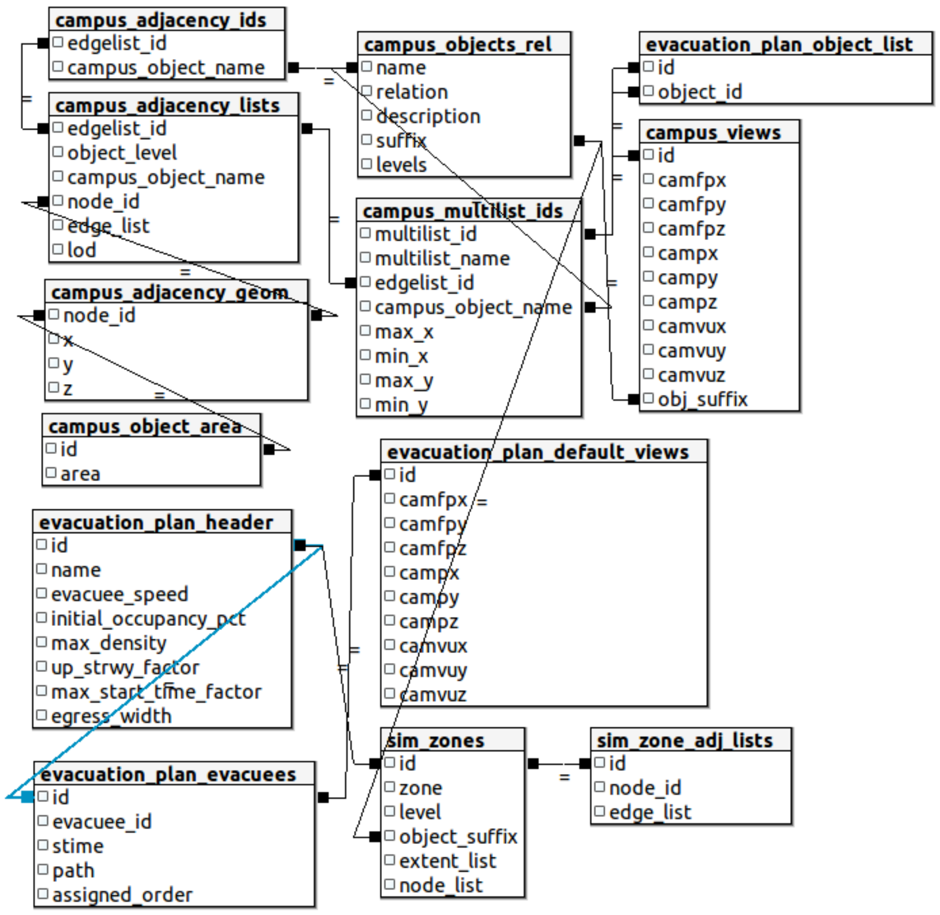


Figure 3.13: ER Diagram for Route Planner output and Simulator Input.

3.1.5 EVACUATION SCENARIO DATA MODEL

- **campus_objects_rel:** Maps readable names for campus objects to table name in the database. This improves the data driven application model by allowing the developer to add objects to the table and access them in a standard way rather than hard coding application relationships.
- **campus_adjacency_lists:** Assign an ID to Campus Object Adjacency Lists. Unless a building is modified the application produces reduced graph structures once and saves them for further use by multiple applications.
- **campus_multilist_ids:** Combinations of objects that connect multiple graphs together for multi-object scenarios.
- **campus_adjacency_geom:** Saves the location of an objects attributes.
- **campus_object_area:** Saves the area of an objects attributes.
- **campus_adjacency_ids:** Relation table for joins on scenario save/load.
- **campus_views:** Saves default camera position for an object.
- **evacuation_plan_header:** Save overall parameters for a scenario.
- **evacuation_plan_evacuees:** Evacuation plan table produced by the route planner and used as input to the simulator.
- **evacuation_plan_object_list:** Relation table for joins on scenario save/load.
- **evacuation_plan_default_views:** Provide multiple object views within a multi-object scenario. Produced as a function of the route planner and utilized by the simulator.
- **sim_zones:** Building zones adjacency lists.

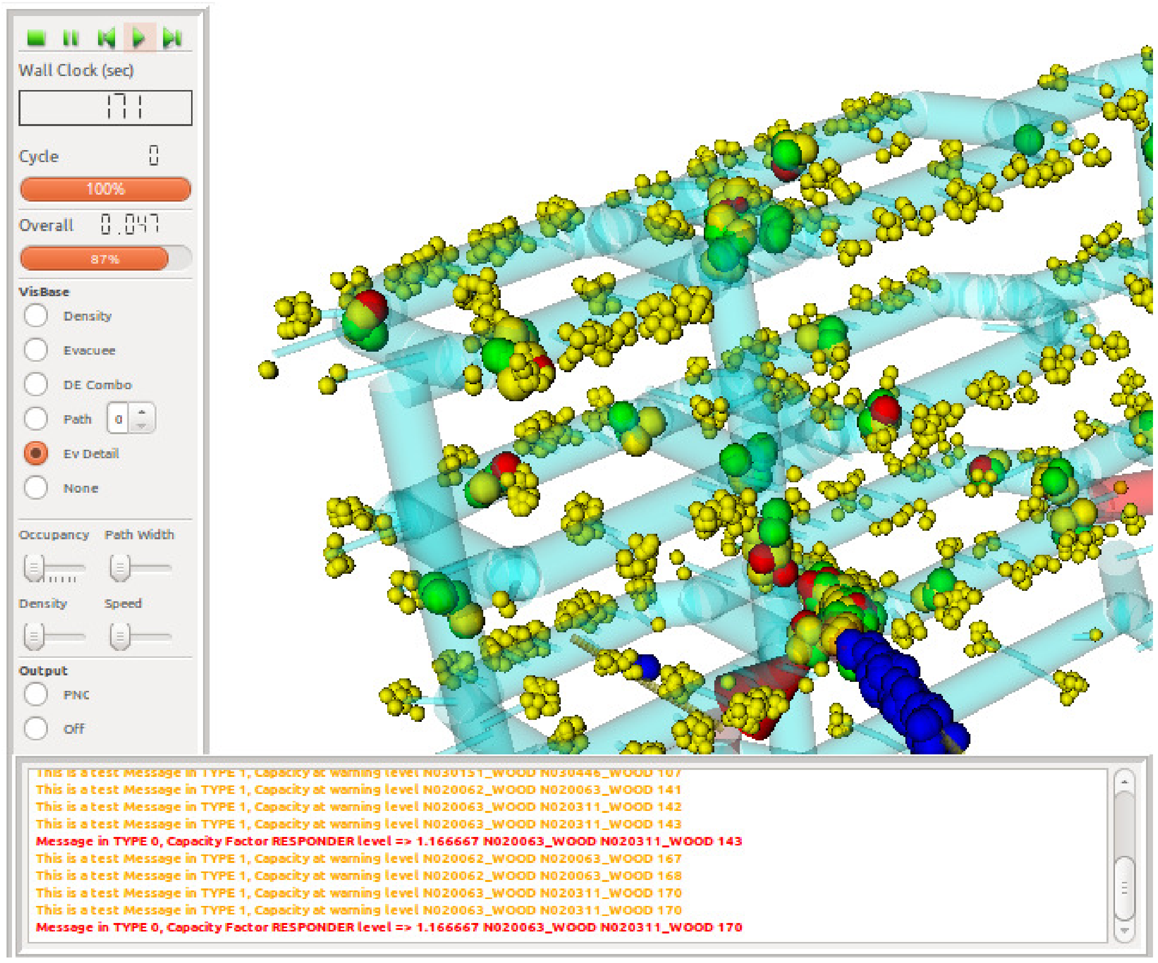
- **sim_zone_adj_lists:** Relation table for joins on scenario save/load.

3.1.6 ROUTE PLANNER EXECUTION AND VISUALIZATION

The application functionality for route planning and emergency evacuation simulation are 2 distinct units of work. The emergency evacuation visual analytics system is discussed in the following section. Here we describe in detail the route planner visualization. 3.15 shows a building in density visualization mode. Density only evacuations are done the first time a scenario is executed for route planning purposes. The tubes are colored green to red and enlarged based on maximum density thresholds set by the user(see Figure: 3.12). The widget to the left shows:

- Simulated wall clock time, based on evacuee speed setting.
- Cycle and overall progress sections.
- VisBase radio buttons for selecting the evacuee and density visualization modes.
- Sliders to adjust initial occupancy, path width, density and speed. These adjustments can be set during execution and effect the entire algorithm execution. If adjusted the route planner is reset.
- By selecting the PNG output button a movie is produced. This is necessary for fast visualization of the relatively long running route planner.

Different visualization modes provide multiple interpretations. Density Mode is a fast running mode where the execution time of the route planner is nearly the same as the visualization time. Since the graphics production is light weight it produces minimal delays. This mode is executed on the first production of the plan for a scenario. More detailed visualization modes are used when a scenario has been reloaded in the application because all the work done by the shortest path calculations of Algorithm 6 are complete, therefore reducing run time for evaluation. Ev Detail Mode provides greater detail. The spheres are adjusted in by size and color. The blue spheres are outside the building.

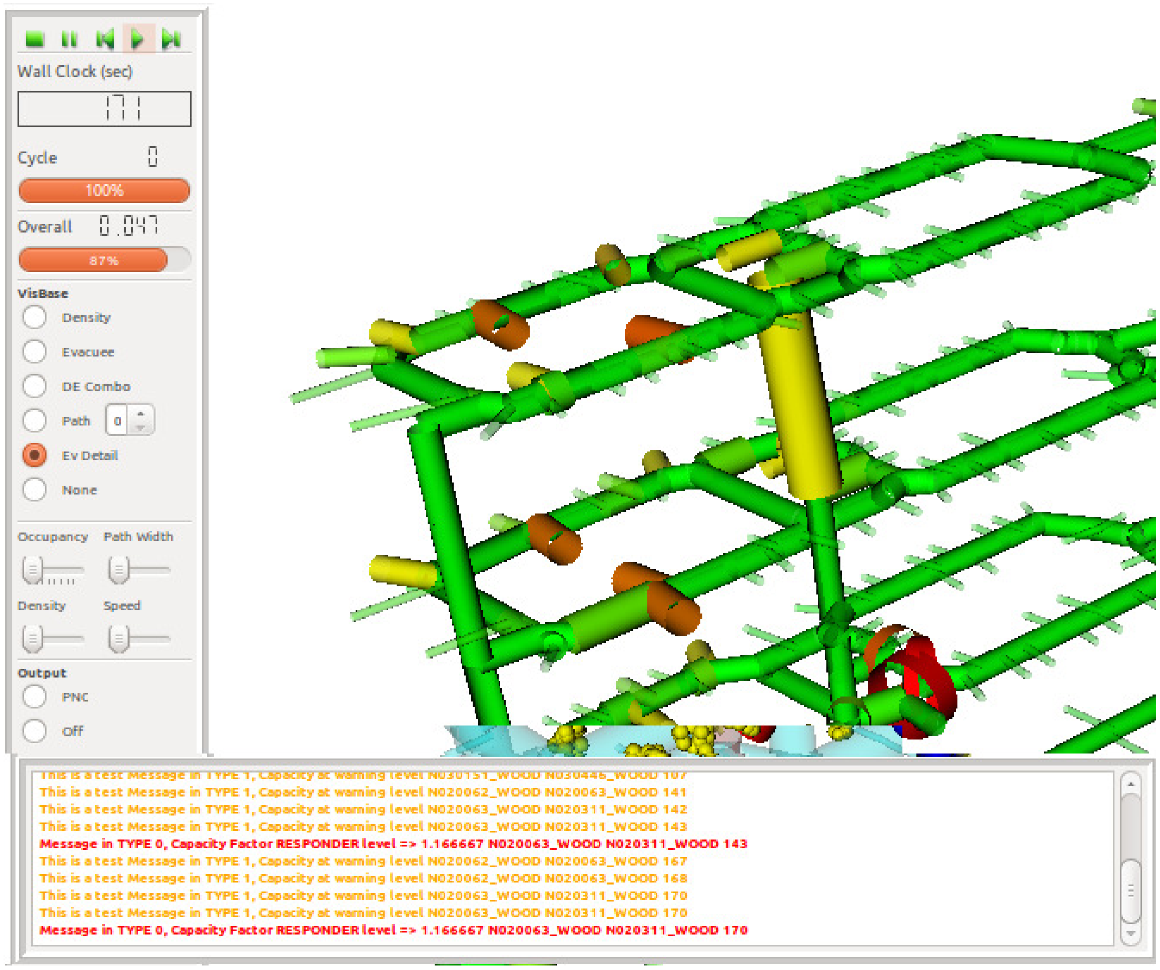


VisBase: Density Evacuee DE Combo Path [✓ Ev Detail] None

```
This is a test Message in TYPE 1, Capacity at warning level N030151_WOOD N030446_WOOD 107
This is a test Message in TYPE 1, Capacity at warning level N020062_WOOD N020063_WOOD 141
This is a test Message in TYPE 1, Capacity at warning level N020063_WOOD N020311_WOOD 142
This is a test Message in TYPE 1, Capacity at warning level N020063_WOOD N020311_WOOD 143
Message in TYPE 0, Capacity Factor RESPONDER level => 1.166667 N020063_WOOD N020311_WOOD 143
This is a test Message in TYPE 1, Capacity at warning level N020062_WOOD N020063_WOOD 167
This is a test Message in TYPE 1, Capacity at warning level N020062_WOOD N020063_WOOD 168
This is a test Message in TYPE 1, Capacity at warning level N020063_WOOD N020311_WOOD 170
This is a test Message in TYPE 1, Capacity at warning level N020063_WOOD N020311_WOOD 170
Message in TYPE 0, Capacity Factor RESPONDER level => 1.166667 N020063_WOOD N020311_WOOD 170
```

Figure 3.14: Zoom in on visualizing Route Planner on a campus building in EvDetail Mode. Evacuee Detail with density represented by coloring evavuees and larger sphere representing a group of evaucees being assigned a path. See Algorithm 6.

To explain the use of the movie function a multi-building evacuation is presented. See Figure 3.16. The "PNG" function produces a folder of sequentially numbered images that can be imported into a typical movie maker application and presented on any system and even placed on a server for remote download. The results, of course, are static and cannot be modified under changing conditions. There advantage is speed, and portability. A 10000 evacuee Route Planning evacuation can be viewed in less than 15 seconds and on any system with a standard movie player.



VisBase: Ev Detail [✓]

```
This is a test Message in TYPE 1, Capacity at warning level N030151_WOOD N030446_WOOD 107
This is a test Message in TYPE 1, Capacity at warning level N020062_WOOD N020063_WOOD 141
This is a test Message in TYPE 1, Capacity at warning level N020063_WOOD N020311_WOOD 142
This is a test Message in TYPE 1, Capacity at warning level N020063_WOOD N020311_WOOD 143
Message in TYPE 0, Capacity Factor RESPONDER level => 1.166667 N020063_WOOD N020311_WOOD 143
This is a test Message in TYPE 1, Capacity at warning level N020062_WOOD N020063_WOOD 167
This is a test Message in TYPE 1, Capacity at warning level N020062_WOOD N020063_WOOD 168
This is a test Message in TYPE 1, Capacity at warning level N020063_WOOD N020311_WOOD 170
This is a test Message in TYPE 1, Capacity at warning level N020063_WOOD N020311_WOOD 170
Message in TYPE 0, Capacity Factor RESPONDER level => 1.166667 N020063_WOOD N020311_WOOD 170
```

Figure 3.15: Zoom in on visualizing Route Planner on a campus building in density mode.

3.2 SITUATIONALLY AWARE VISUAL ANALYTICS SYSTEM

The visual analytic system (Figure 3.17) consists of a simulator that accepts user input during an emergency, a 3D interactive animated display of the ongoing evacuation, and reporting and analytic modules. All these views accept direct input and the views are linked to update automatically, resulting in presenting the user with the most current information.

3.2.1 SIMULATOR

The simulator loads evacuation plans that were saved for scenarios under normal static conditions. Since dynamic changes affect only a small part of the structure,

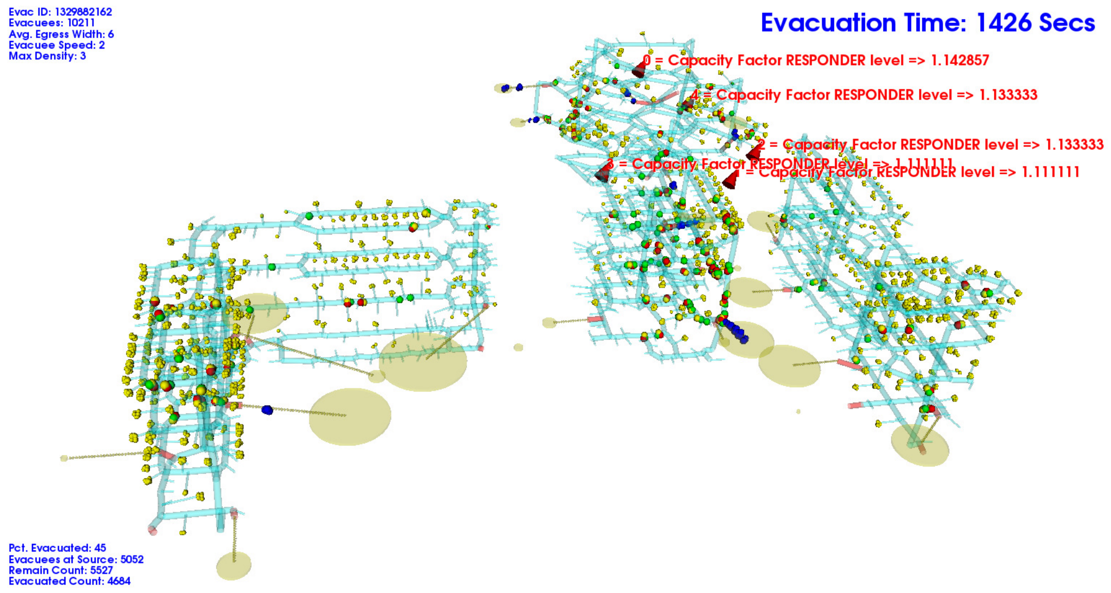
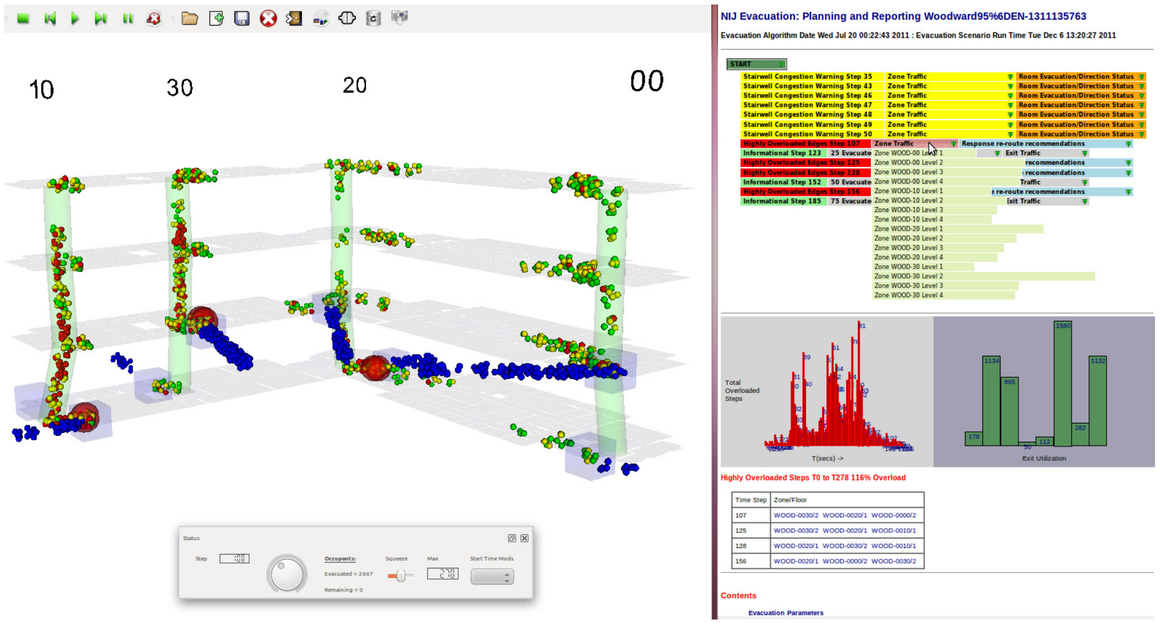

Evac ID: 1329882162
Evacuees: 10211
Avg. Egress Width: 6
Evacuee Speed: 2
Max Density: 3
Evacuation Time: 1426 Secs
0 = Capacity Factor RESPONDER level => 1.142857
4 = Capacity Factor RESPONDER level => 1.133333
2 = Capacity Factor RESPONDER level => 1.133333
3 = Capacity Factor RESPONDER level => 1.111111
1 = Capacity Factor RESPONDER level => 1.111111
Pct. Evacuated: 45
Evacuees at Source: 5052
Remain Count: 5527
Evacuated Count: 4684

Figure 3.16: All evacuation parameters are available on the top and bottom left. Total evacuation time is available top right. The density messages are available at "Responder level" congestion thresholds and muster point/exit utilization circles remain available to the First Responder.

a *large amount of preprocessed data can be reused, contributing to our real-time performance*. The simulator accepts user defined dynamic changes (specification of blockages or casualty reports through the 3D display or the report modules) and modifies computed paths, and reroutes occupants. In addition it updates the generated reports and responder recommendations or action plans.

Figure 3.18 has a scroll slider to move around in the simulation and informational status presentations and a "Squeeze" slider. The Start Time Mods dialog is for future implementations. At this point it is necessary to explain in detail the "Squeeze" function.

Squeeze is an acronym for the function that sets the limit on the start time for each evacuee. Referring back to the heuristic based route planning algorithm[15] of Alg. 6 it is important to emphasize that when a free path all the way to an exit cannot be found for an evacuee the evacuee remains at the source. This is not a realistic assumption in an actual evacuation. The Squeeze in implemented when the Scenario



NIJ Evacuation: Planning and Reporting Woodward95%6DEN-1311135763

Evacuation Algorithm Date Wed Jul 20 00:22:43 2011 : Evacuation Scenario Run Time Tue Dec 6 13:20:27 2011

START

Stairwell Congestion Warning Step 35 | Zone Traffic | Room Evacuation/Direction Status
Stairwell Congestion Warning Step 43 | Zone Traffic | Room Evacuation/Direction Status
Stairwell Congestion Warning Step 46 | Zone Traffic | Room Evacuation/Direction Status
Stairwell Congestion Warning Step 47 | Zone Traffic | Room Evacuation/Direction Status
Stairwell Congestion Warning Step 48 | Zone Traffic | Room Evacuation/Direction Status
Stairwell Congestion Warning Step 49 | Zone Traffic | Room Evacuation/Direction Status
Stairwell Congestion Warning Step 50 | Zone Traffic | Room Evacuation/Direction Status
Highly Overloaded Edges Step 107 | Zone Traffic | Response re-route recommendations
Informational Step 123 | 25 Evacuate | Zone WOOD-00 Level 1 | Exit Traffic
Highly Overloaded Edges Step 125 | Zone WOOD-00 Level 2 | recommendations
Highly Overloaded Edges Step 128 | Zone WOOD-00 Level 3 | recommendations
Informational Step 152 | 50 Evacuate | Zone WOOD-00 Level 4 | Traffic
Highly Overloaded Edges Step 156 | Zone WOOD-10 Level 1 | re-route recommendations
Informational Step 185 | 75 Evacuate | Zone WOOD-10 Level 2 | Exit Traffic

Zone WOOD-10 Level 3
Zone WOOD-10 Level 4
Zone WOOD-20 Level 1
Zone WOOD-20 Level 2
Zone WOOD-20 Level 3
Zone WOOD-20 Level 4
Zone WOOD-30 Level 1
Zone WOOD-30 Level 2
Zone WOOD-30 Level 3
Zone WOOD-30 Level 4

Highly Overloaded Steps T0 to T278 116% Overload

Time Step	Zone/Floor
107	WOOD-0030/2 WOOD-0020/1 WOOD-0000/2
125	WOOD-0030/2 WOOD-0020/1 WOOD-0010/1
128	WOOD-0020/1 WOOD-0030/2 WOOD-0010/1
156	WOOD-0020/1 WOOD-0000/2 WOOD-0030/2

Contents

Evacuation Parameters

Figure 3.17: Visualization Design. The upper left panel is the 3D view of the building undergoing an evacuation. Spheres encapsulate evacuee population densities, permitting easy identification of congestion points in the building. Vertical tubes (light green) represent stairways/elevators. Cubes on the first and second floor indicated exits. Lower left indicates the status bar for animation control. The upper right panel is for displaying reports of significant events, that can be drilled down. Lower right panels(bar graphs) show aggregate information on exit occupancy as well as events arranged on a timeline. The report views and the 3D views are linked for immediate updates.

is loaded into the simulator at an evacuation scene. The start time for each evacuee saved in the route planner is divided by a factor from 1 to 9 based on the slider(in the case illustrated approximately 5).

The effect of Squeeze on the relationship between the evacuation presented by the route planner and that presented by the simulator is that the congestion levels are raised to 10 to 20 percent greater in the simulation. Realism is maintain however because the simulator moves evacuees based on congestion even though their paths were loaded statically from the route planner.

Algorithm 7 defines the procedure for loading a previously saved route plan into the simulator. Notice the Shortest path step is removed and the Squeeze is implemented. Since the evacuees are moved in the same fashion the simulation is identical

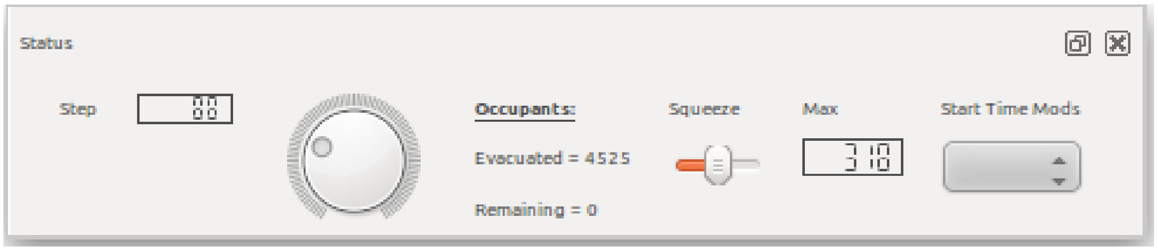


Figure 3.18: Floating Control Widget. From bottom left of Figure 3.17

to the route plan except as explained above.

Algorithm 7: Simulation Load Algorithm

```
Input: Evacuation Plan : Routes with schedules of evacuees on each route
       refered to internally as array SavedEvacueeList;
Result: Animation Structure: Associative array AnimationStep
1  for i = 0 to i = Length(SavedEvacueeList) do
2      sm = SqueezedMove(SavedEvacueeList[i][StartTime]);
3      while sm <= self.moves do
4          j = self.SavedEvacueeList[i][EvacueeId];
5          AssignRoutingInfo(Evacuees[j]);
6      end
7      MoveEvacuees();
8      SetAnimationStep();
9  end
10 while evacuees not at destination nodes do
11     MoveEvacuees();
12     SetAnimationStep();
13 end
```

The AnimationStep associative array is used to iteratively step through a scenario and provides a structure for the user to move back and forth in a simulation. This is also the primary control structure for the analytic and report structures defined below.

3.2.2 VISUALIZATION DESIGN

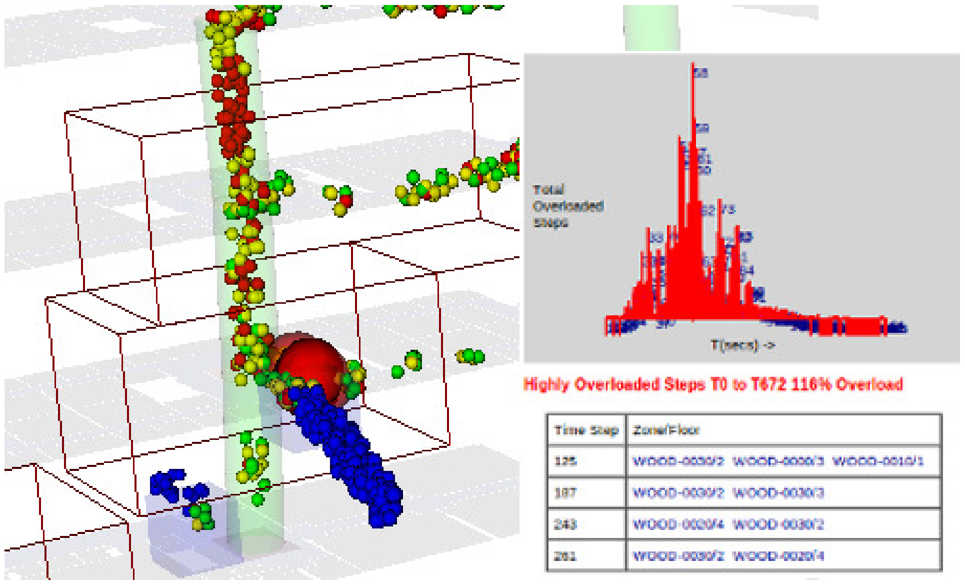
Figure 3.17 presents our interactive visualization system. Almost all of the interaction are via *direct manipulation*. There are three components that make up the design, (1) a 3D animated view of the urban structure, where the user can load and play evacuation simulations. Operations to manipulate the evacuation include play, pause, step forward and back, and restart the simulation. Evacuees are represented as spheres and colored green, yellow, or red based on low to high congestion. Partially transparent light green tubes represent stairwells, while. purple cubes are exits. Blue polygons represent areas that can be occupied. Large red spheres of varying opacity represent edge (capacity) congestion. On the bottom left is the *status tool widget*, that allows moving around in the animation with a slider, indicating current step, evacuee counts and total simulation time. The top right panel is the *significant event* window. The rectangular bars are menus with varying levels of detail of the simulation report. The bottom right panel is a scrolling widget with interactive charts and graphs for interacting with the simulation and visual analysis.

Following are notable features of our visualization design:

Visual Cues. Congestion is the primary concern of each evacuee and predictions of future congestion and mitigation is of concern to the emergency response teams. In our system, congestion levels range from green to red (low to high). In earlier work[7] it was more common to represent evacuees as 3D human models with detailed walking and turning patterns. We believe this is unnecessary in emergency evacuation scenarios, where the evacuee densities and their locations are of prime concern to first responders.

Temporal Cues. The color and size of spheres is modified at significant event times. For example sphere size is enlarged when the simulator starts moving evacuees from a source location. The resultant pulsing in the animation yields important spatial and temporal information to the user about where a new source of traffic will originate and likely areas of future congestion.

Animation Cueing Based on Congestion vs. Time Graphs. Congested step graphs are rolled over and clicked to move around in the scenario. The temporal information gathered in this way has revealed interesting points of comparison between time steps in a simulation. In fact, comparing a congestion graph and a paused frame in the simulation reveals information about what will occur before and after the given time steps in view. We have found no other work that attempts this useful manipulation. Figure 3.19 illustrates what is available to the user in any scenario when the rising edge of the primary congestion curve bell graph is clicked by the mouse. We have found in our analysis that with limited resources the commander could select this point and visualize the situation in the building that would be most important to act upon.



Time Step	Zone/Floor
125	WOOD-0030/2 WOOD-0000/3 WOOD-0010/1
187	WOOD-0030/2 WOOD-0030/3
243	WOOD-0020/4 WOOD-0030/2
261	WOOD-0030/2 WOOD-0020/4

Figure 3.19: Rollover on highly overloaded steps at Time Step 126 highlights sections in 3d cubes. The Total overloaded steps graph has been selected at the rising edge of the primary bell curve at step 126. The result is a cue for responder action in the evacuation scenario.

Details on Demand Display. The 'Significant Event' window in the reporting tool uses a colorized layered menu so that the visualization of significant information is presented as needed by the user. This allows a maximum amount of reporting while allowing for quick event scanning in the event report. A user sees a limited top level distribution of data unless there is a reason to drill down deeper into the event for details. In the top right of Figure 3.17 the user has selected a *Heavy Zone Congestion* item (in red) with its time step. The user can explore further via a mouseover operation. to reveal a bar graph that shows the relative congestion of each zone in the building.

Direct Interaction On Linked Views. The simulation is manipulated by direct (mouse and keyboard) interaction over the 3D animation and report views. For instance, a blockage can be introduced via the 3D view, and simulation rerun to generate new (rerouted) paths for impacted evacuees; the report view is updated to reflect the situational change. Similarly the interaction with the reports menus, charts, etc. temporally updates the 3D animation view. All such operations are performed in near real-time as both the visual representation (spheres representing collections of evacuees) and the underlying simplified graph structure allows these operations to be rapidly performed, facilitating interactive visual analysis.

3.3 IMPLEMENTATION

Our system is built utilizing open source toolkits on a linux system, using Python 2.7[22]. 3D rendering is done with the Visualization Toolkit(VTK)[18] and the GUI is produced with QT4[17]. We use a PostgreSQL database with the PostGIS extensions. The reports section is an HTML/Javascript window inside a QT4 widget. This allows automatic porting to mobile device browsers. VTK allows us to provide 3D interaction with their custom window interactor without additional coding required to do so in a lower level graphics package such as OpenGL. The database is accessed running on the local machine with direct package calls.

CHAPTER 4: EXAMPLE SCENARIO

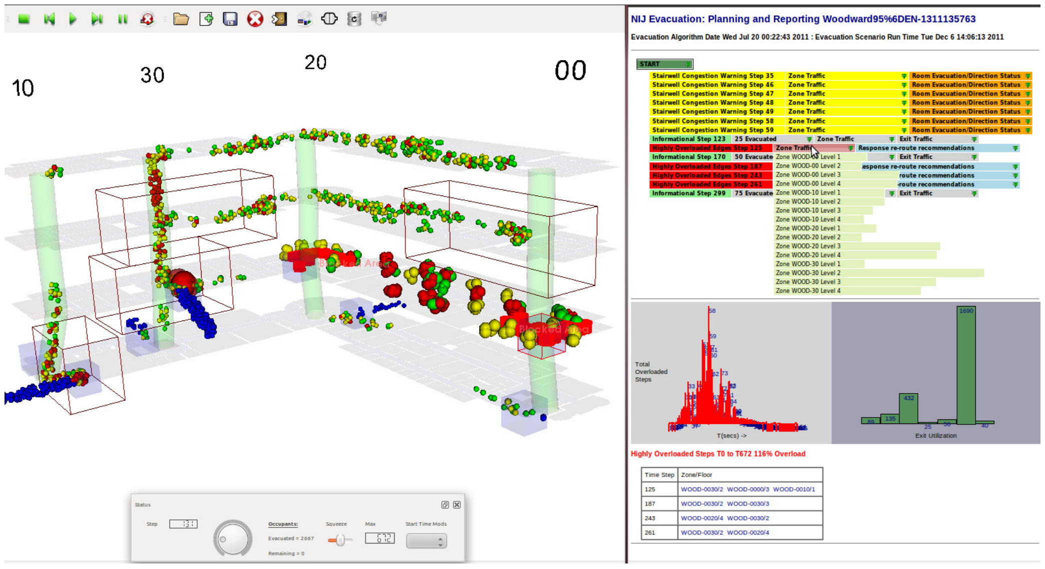
Here we describe three experimental scenarios to illustrate the use of our application. In this scenario, there are 3500 evacuees. The maximum egress capacity is set to 1 evacuee per cubic foot and navigation speed is set at 3ft/sec (congestion can slow down or halt evacuees during a simulation). The building is loaded to 95 percent capacity.

4.1 SITUATION 1: NO BLOCKAGES

Figure 3.17 is a screen shot of our visual analytic system loaded with a campus building with no inaccessible areas. The 3D animation view is a direct representation of the evacuation scenario which was built in the scenario construction phase of the process. Evacuees are represented by spheres, clustered visually using an algorithm which arranges them based on egress width. A small red sphere indicates an evacuee cluster on a congested step. Large red spheres of varying opacity indicate congestion greater than 116 percent of rated capacity. In the timestep shown in Figure 3.17, the user has clicked on the reporting panel(upper right), representing a highly loaded event at timestep 108 sec. The user has also rolled over the zone congestion bar to reveal the detailed zone congestion graph. As indicated by the bar at Zone 30 Level 2 this is the most traveled and congested route. The application suggests that a responder be dispatched to this area. As the user rolls over the associated ‘Response Reroute Recommendations’ menu bar in the list the recommended action is visible.

Easy access for responders and special evacuation needs can be found by looking at the green exit utilization bars(lower right of Figure 3.17). If an area has become too congested for a responder, he or she can be dispatched to exits with lower utilization, and then can direct evacuees toward less congested areas. The zone exits in this scenario that are not utilized are on the 1st level. This type of information can be a powerful dispatch tool. The emergency commander has what he/she needs to make a decision: the amount and location of the congestion in the context of the overall evacuation and where related less congested paths are located.

Each bullet in the significant event list (upper right panel in Figure 3.17) serves as a visual clue to the overall execution of the scenario. The list covers the entire evacuation. The yellow stairway warnings can be drilled deeper to see which areas are becoming congested. These cues are important for responders to quickly react during the beginning of an event or if a campus lock down has been released. The room evacuation, direction status options can be rolled over to indicate the direction from which the traffic is proceeding, which might result in congestion at a later point.



NIJ Evacuation: Planning and Reporting Woodward95%6DEN-1311135763

Evacuation Algorithm Date Wed Jul 20 00:22:43 2011 : Evacuation Scenario Run Time Tue Dec 6 14:06:13 2011

START

Stairwell Congestion Warning Step 35 Zone Traffic Room Evacuation/Direction Status
Stairwell Congestion Warning Step 46 Zone Traffic Room Evacuation/Direction Status
Stairwell Congestion Warning Step 47 Zone Traffic Room Evacuation/Direction Status
Stairwell Congestion Warning Step 48 Zone Traffic Room Evacuation/Direction Status
Stairwell Congestion Warning Step 49 Zone Traffic Room Evacuation/Direction Status
Stairwell Congestion Warning Step 58 Zone Traffic Room Evacuation/Direction Status
Stairwell Congestion Warning Step 59 Zone Traffic Room Evacuation/Direction Status
Informational Step 123 25 Evacuated Zone Traffic Exit Traffic
Highly Overloaded Edges Step 125 Zone Traffic Response re-route recommendations
Informational Step 170 50 Evacuate Zone WOOD-00 Level 1 Exit Traffic
Highly Overloaded Edges Step 187 Zone WOOD-00 Level 2 esponse re-route recommendations
Highly Overloaded Edges Step 243 Zone WOOD-00 Level 3 route recommendations
Highly Overloaded Edges Step 261 Zone WOOD-00 Level 4 -route recommendations
Informational Step 299 75 Evacuate Zone WOOD-10 Level 1 Exit Traffic
Zone WOOD-10 Level 2
Zone WOOD-10 Level 3
Zone WOOD-10 Level 4
Zone WOOD-20 Level 1
Zone WOOD-20 Level 2
Zone WOOD-20 Level 3
Zone WOOD-20 Level 4
Zone WOOD-30 Level 1
Zone WOOD-30 Level 2
Zone WOOD-30 Level 3
Zone WOOD-30 Level 4

Highly Overloaded Steps T0 to T672 116% Overload

Time Step	Zone/Floor
125	WOOD-0030/2 WOOD-0000/3 WOOD-0010/1
187	WOOD-0030/2 WOOD-0030/3
243	WOOD-0020/4 WOOD-0030/2
261	WOOD-0030/2 WOOD-0020/4

Figure 4.1: Two blockages have been placed in the building at 45 seconds into an emergency evacuation. (1) Floor 2 at zone 20 stairwell and (2) floor 2 at zone 00 stairwell. Individuals are shown trapped between them by enlarged spheres. Wire-frame cubes indicate traffic flow areas, obtained by rolling over the bar chart at the bottom of the report window.

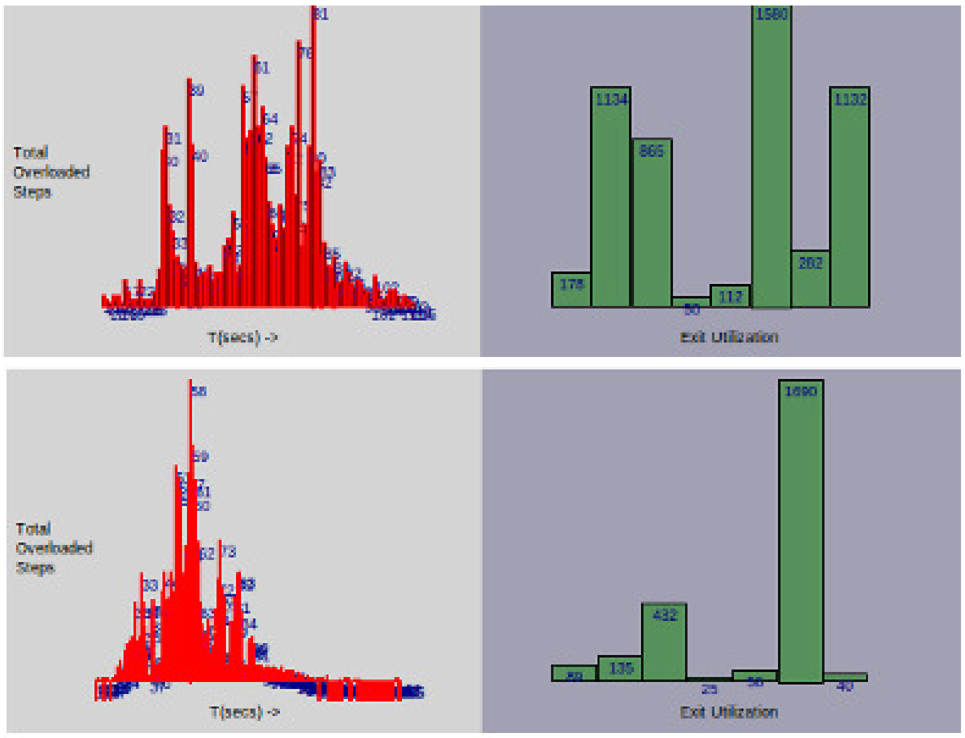


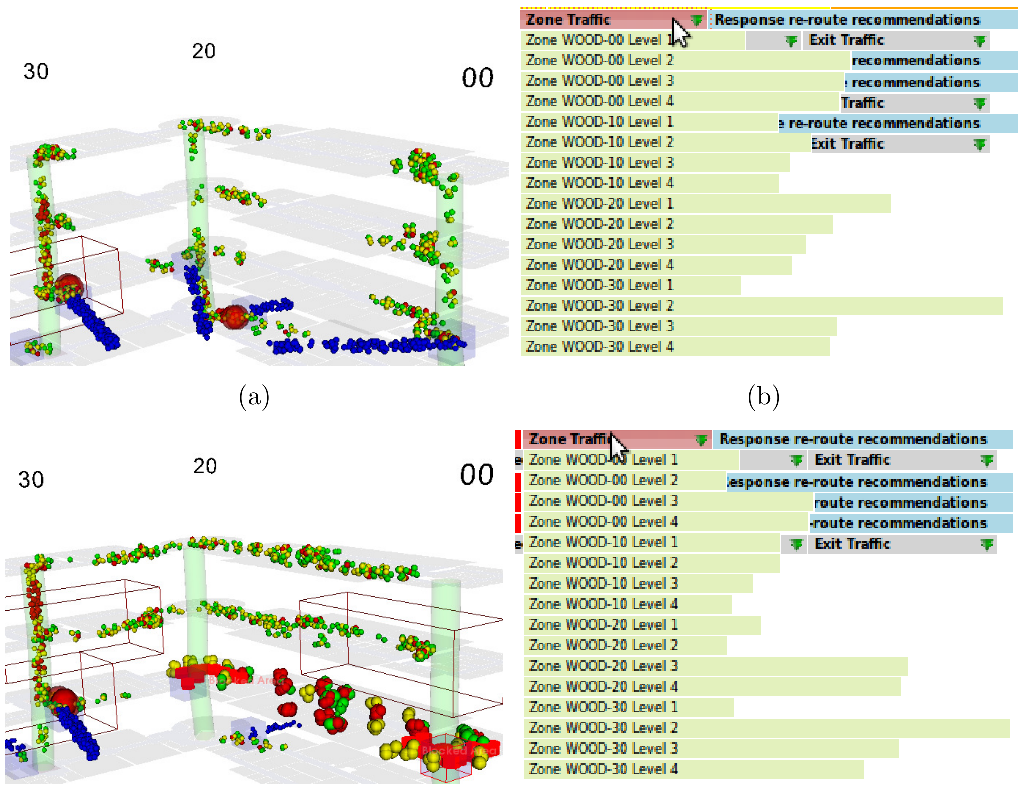
Figure 4.2: Before and after congestion and exit charts.

4.2 SITUATION 2: INDUCED BLOCKAGES

As illustrated in Figure 4.1, two blockages have been introduced into the scenario of Figure 3.17. The blocked areas represented by red squares spread out over several square feet on the second floor at zone 00 and zone 20. In this example, a total of 3100 evacuees were rerouted and all reporting re-calculated in 2.8sec. Note that in the control bar at the bottom of the 3D animation view the maximum evacuation time has increased to nearly 7 minutes from less than 3 minutes.

Figure 4.2 shows the drastic shifts in the movement of people from a standard evacuation of the building. This example serves to show that evacuation modeling of normal (non-blocked) scenarios is considerably different than when blockages are introduced. In particular, notice the difference in the utilization of the exits (right panel). The left panel shows the congestion amounts as a function of time. In the blocked case, the congestion occurs earlier and is steeper, with a longer tail, resulting in longer times for all evacuees to exit the building. Probing these congestion points can immediately bring up the 3D view at that instant of time to see the evacuee locations, thus providing a powerful means to quickly understand the situation.

Mouse rollover on the significant event list view shows that the third and fourth floors are getting backed up above the exit at zone 20 floor 2 which is loaded heavily even during a normal scenario. Because the event occurred early there were a number of evacuees occupying the upper floors.



(a) (b)

(c) (d)

Figure 4.3: Contrast between normal(unblocked) vs. blocked scenarios. Two blockages have been introduced. (a,b) 3D view at 108 seconds into the evacuation, with top floors mostly evacuated, (c,d) 3D view at 139 sec. Floors 3 and 4 are still heavily occupied. Zone traffic (right panels) confirm and illustrate the aggregate picture of the traffic across the entire evacuation.

Figure 4.3 further contrasts the blocked and non-blocked cases. Here the top row shows the normal(unblocked) case and the bottom row shows the blocked case. We compare the zone traffic via the light green bar charts. The bar charts show total zone traffic from the beginning to the end of the evacuation scenario. The snapshot of the building in Figure 4.3a,b is at 108 seconds and Figure 4.3c,d at 131 seconds. Even though the bar chart for the blocked case is the total picture it is still clear from both the building and zone traffic charts that the traffic on the top two floors is heavier in the blocked case (Figure 4.3c,d), and in particular, is shown by the size of the bars labeled Zone WOOD-20 Level 3, Zone WOOD-20 Level 4, Zone WOOD-30 Level 3, and Zone WOOD-30 Level 4. The total picture of the bar chart and the single step picture of the building reinforce each other. Each can stand on its own but more insight is gained if they are viewed together. Comparison of non blocked and blocked evacuation visualizations based on these factors can give previously unavailable clues on critical areas of the building across multiple scenarios.

4.3 SITUATION 3: REROUTING EVACUEES

When certain parts of a building are blocked, our system will automatically reroute all evacuees in that area to other nearby exits. Additionally, our system provides rerouting functions that allow a selected number of evacuees to be rerouted; for instance, in congested areas it might be desirable to move evacuees to other exits. This operation is performed in real-time and the simulation played through to evaluate the traffic or congestion patterns resulting from such an intervention. This could then let a commander make a more informed decision and dispatch responders to the affected areas in the building.

Figure 4.4 illustrates an example evacuation from a cluster of 4 academic buildings. The original evacuee densities are illustrated in panel 2. Exits 8 and 16 are heavily used, as indicated by the area of their exit circles. The red cubes in panel 1 have been interactively selected(panel 3 shows a zoomed-in view of these areas) for rerouting occupants within those areas. This is followed by specifying the number of evacuees to be rerouted to specific exits (here exits 2 and 23 were chosen). Panel 4 shows the results of these actions, leading to reduced densities at exits 8 and 23.

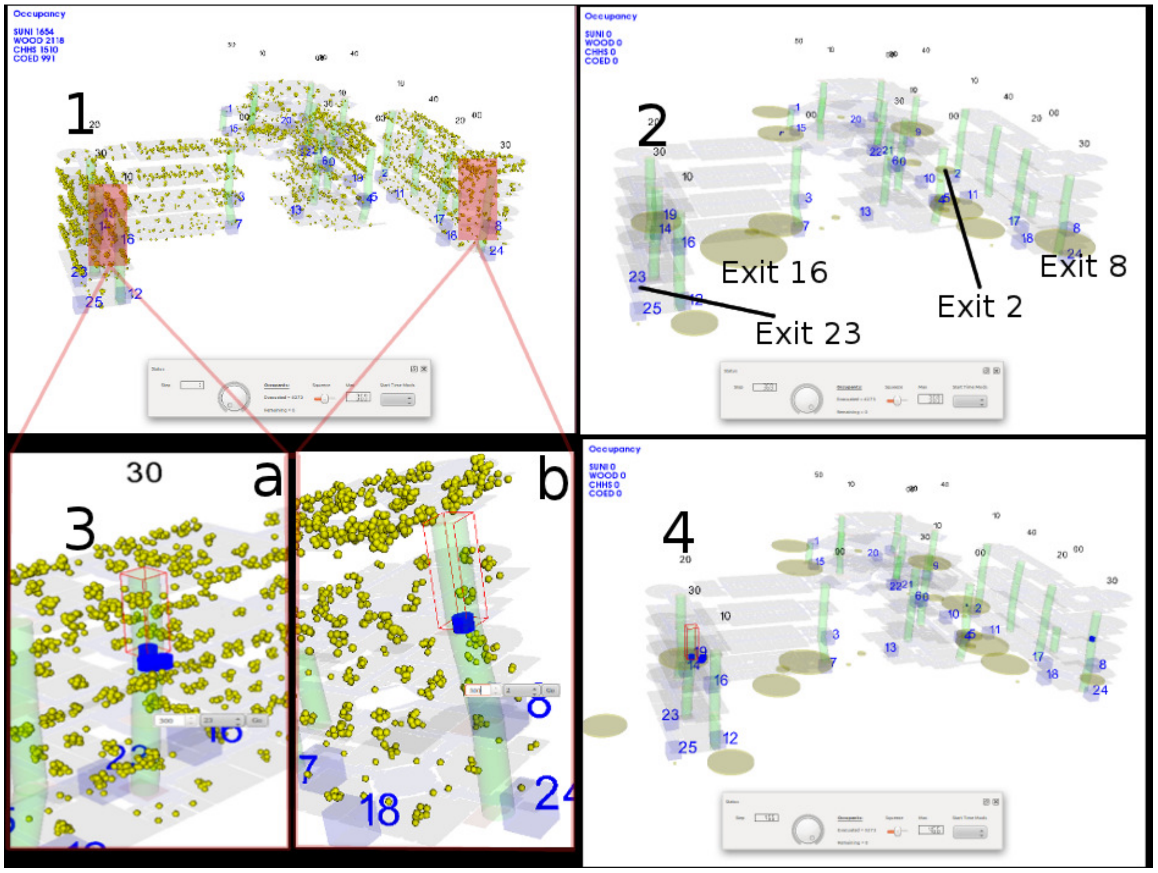


Figure 4.4: Rerouting evacuees from congested areas in a 4 building evacuation simulation. **1.** Four building scenario. **2.** Resulting evacuee density circles after simulation. **3a and 3b)** User added rerouting flags as indicated by the blue discs and associated with the opaque red rectangles in 1. **4.** Resulting evacuee density circles after modified simulation, changing the routes of evacuees to exit 2 and exit 23 in their respective buildings.

This function has value for planning and training. Building lockdown and release operations can benefit from such 'what-if' style scenarios that brings together rich spatio-temporal information into the hands of first responders. This is facilitated by the responsiveness of the system and thus its effectiveness in real-time decision support. In this example, running the entire scenario from initiation to results and analysis took approximately 2 minutes. When large collections of buildings are involved with traffic routed to the adjacent street networks, such tools can be invaluable for effective and timely evacuation as well as optimal asset deployment decisions.

CHAPTER 5: EVALUATION: TABLETOP EXERCISE

The development of our application has included regular feedback and demonstrations with campus emergency and safety personnel, including the chief of police, other safety officers, and campus business continuity staff. This partnership has resulted in continuous improvement of the system based on the knowledge and experience of stakeholders in the area of public safety. This process also led to the scheduling of a table top exercise with these campus officers. We ran the application through three different scenarios to determine our system's usability, effectiveness, and to assess needs for improvements. A business continuity office staff member (an expert in running training exercises) designed the scenarios. The campus police chief, a senior police officer, and the software team participated in the exercise. Well-designed tabletop exercises are essential when dealing with policing and other experts who have little time or for whom physical exercises would be disruptive.

All three scenarios involved a cluster of four campus buildings and a base scenario for the evacuation of approximately 5000 evacuees; preprocessing was performed in accordance with the description in previous sections. The preprocessing step was timed at approximately 8 Minutes. All simulations used this base evacuation object. Video of each of the 3 exercises were recorded for analysis, followed by feedback from the emergency personnel (excerpts in the submitted video). The system was operated by a member of the software team while instructions/commands were received from the police chief.

Figure 5.1: Tabletop System Evaluators.

5.1 SCENARIO 1. GAS LEAK IN BUILDING.

Figure 5.2 shows time sequenced snapshots of a simulated gas leak somewhere in the exercise area. Initially the gas leak was reported as “near Woodward Hall”. The police chief requested a simulation start. As seen in Figure 5.2(a) the buildings are being evacuated as expected with all exits being utilized. Several seconds into the simulation (sim time: 7:26:38) a report is received that the leak was reported to be in the “courtyard”, as shown in the red ellipse(and known to campus personnel) in the figure. The simulation is halted and reset. The police chief instructed first responders to be dispatched to the building exits facing the courtyard. Also, entrance/exits into the courtyard were to be blocked from further use.

The simulation was restarted based on the new situation. We interacted with the software by placing blocks at the requested areas from 7:27:27 until 7:28:19 (Figs. 5.2(a), 5.2(b)). At this point, the software began to recalculate the 5000 evacuee paths. At 7:29:08 calculations were completed and the reporting process rebuilt, including the scenario timeline and the temporal congestion and exit utilization charts. The police chief requested to see the simulation based on the new situation.

Evacuees are confirmed to be exiting the buildings away from the hazard, as seen in Figs. 5.2(c), 5.2(d). The simulated time to exit all buildings increased from 318 seconds to 687 seconds. There were large evacuee populations in the areas of Woodward hall opposite the hazard and it is noted that due to the blockages in the second floor that evacuees are trapped.

5.2 SCENARIO 2. ACTIVE SHOOTER IN BUILDING.

Figure 5.3 shows a time sequenced video snapshots of a simulated active shooter exercise in the Woodward hall. First, the police chief ordered a campus lock down and the building to be evacuated. At this point we switched from the base evacuation scenario of the lower quad (building cluster) and open a base scenario of the Woodward hall. The reason for this is that the scenarios are built as objects and run apriori. We could have placed blockages in the locked down buildings but chose to open a single building scenario for the purposes of the exercise.

Some highlights in this exercise include: building rerouting and reporting occurs in 49 seconds (3 seconds for evacuee rerouting and 46 seconds for report generation). The total time here is similar to the multi-building evacuation because our base scenario included 3600 evacuees. This simulates a highly overloaded building to exercise the software for testing.

5.3 SCENARIO 3. EXPLOSION IN UTILITY PLANT

An explosion in the RUP(regional utility plant) building created a scenario where the four building evacuation simulation of Figure 5.2 was also used. This scenario also found evacuees blocked in the upper floors and the explosion created a hazard in the building courtyard. As reports were received the building floors were blocked in the application and the simulation was started. As more reports were received it

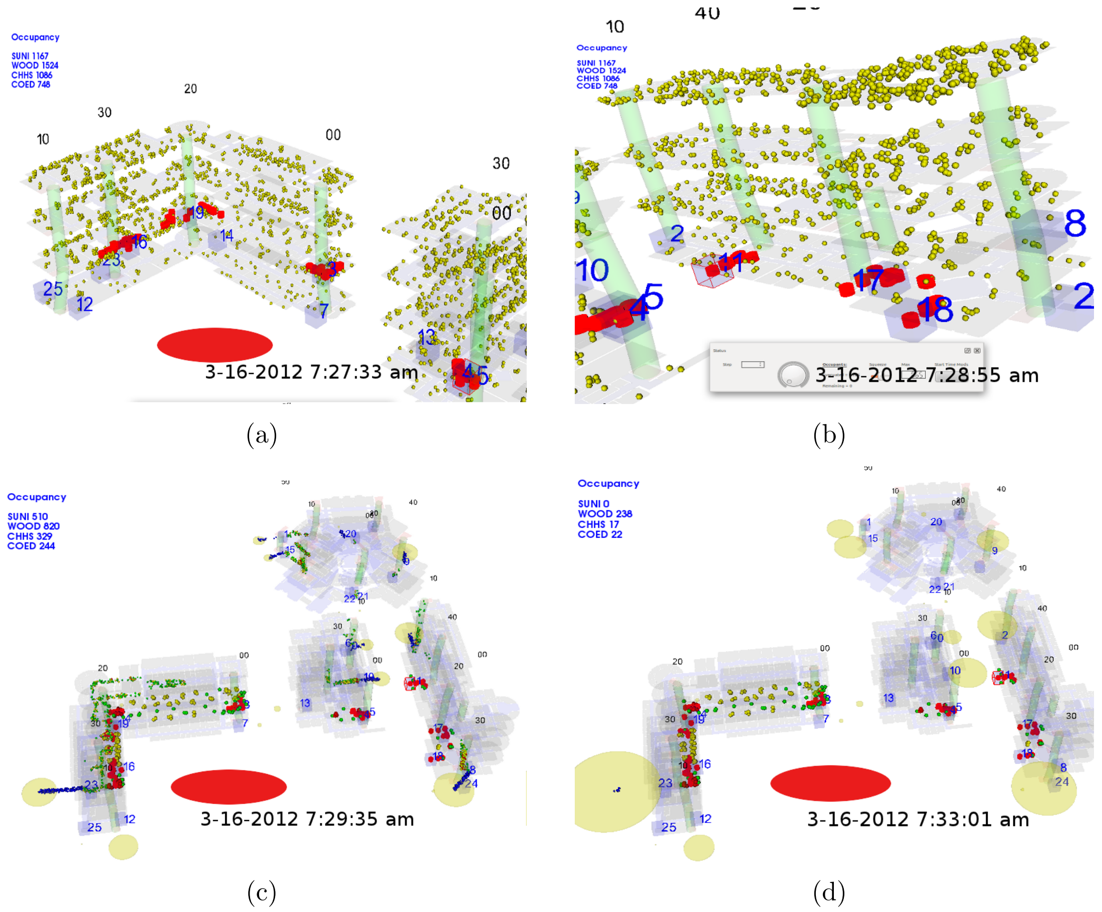
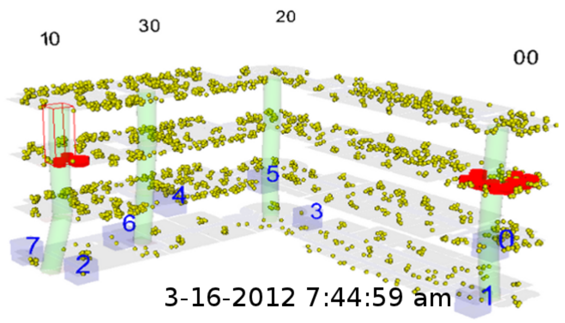
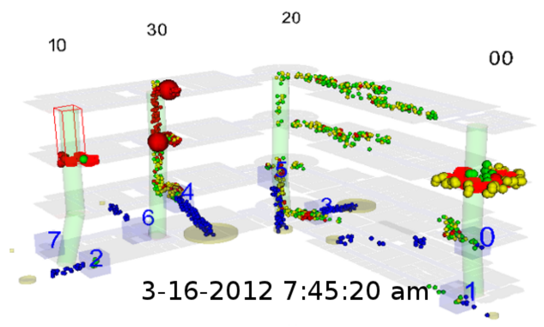


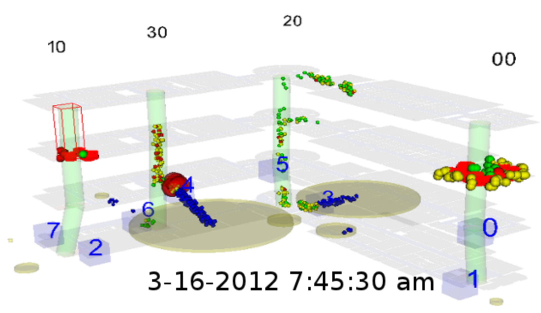
Figure 5.2: Tabletop Exercise: Gas Leak in Building. At approximately 7:20 a gas leak is reported. Lower quad campus possibly affecting four buildings. Evacuation simulation begins. At **Sim. Time: 7:26:38.**, leak confirmed near red ellipse. Simulation suspended, assets are deployed to prevent evacuation into the hazard. Views modified to simulate asset activities at building exits. (a)**Sim. Time; 7:27:33.** Blocking building exits into quad (b)**(5). Sim. Time; 7:28:55.** Blocking building exits from adjacent buildings into quad complete. Signal sent for application to perform situationally aware rerouting, (c) **Sim Time; 7:29:35.** Visualization of new simulation, evacuation in progress, simulating responders interaction at exits to affected areas, (d) **Sim. Time; 7:33:01.** Evacuees are avoiding hazard and and exiting to safe zones, evacuee densities are indicated by areas of yellow circles.



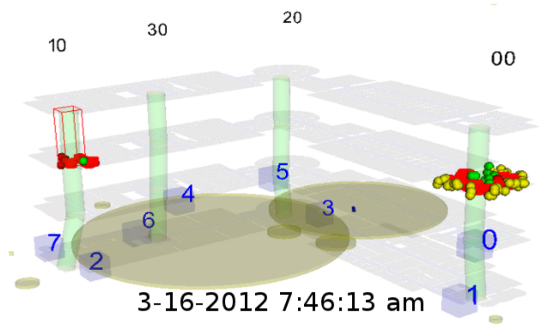
(a)



(b)



(c)



(d)

Figure 5.3: Tabletop Exercise:Active Shooter. At approximately 7:40 a shooter is reported at Woodward hall. Campus is locked down. Reports are received that 3rd floor stairwells are blocked at each end of the building. At 7:44:10 blockages are placed in Woodward by the operator and the simulation is recalculated. (a)**Sim. Time; 7:44:59.** Processing for new evacuation simulation is complete, commander orders visualization of new simulation, (b) **(Sim. Time; 7:45:20.** Trapped evacuees noted at 3rd floor zone 00 and zone 10, (c)**Sim. Time; 7:45:30.** Extreme congestion noted at stairwells in floors 2, 3, and 4 at zone 30. (d)**Sim. Time; 7:46:13.** Simulated evacuation complete. Evacuee populations are indicated by approximate area occupied circles.

became obvious that personnel would exit toward the hazard in the courtyard. The exit density circles alerted the police chief to this problem and emergency personnel were dispatched to redirect these evacuees. At this point the police chief requested the exits facing the courtyard to be blocked. The simulation was restarted and evacuation times and exit results were evaluated as in previous scenarios.

At 8:04:00AM the explosion was reported. At 8:06:18AM we inserted blocks on three stairways on the fourth floor because of a report from Woodward hall. The simulation was started. As the simulation was running, responders were discussing logistics of fire and other emergency personnel deployment in the exercise. The simulation ran to the end. At 8:09:52 the police chief noted the large exit circles in an area near the explosion in the utility building. He requested we reset the simulation and block the exits toward the utility building. At 8:10:30 all second floor exits are blocked in the simulator. The simulation was reset and restarted at 8:11:02.

5.4 ANALYSIS AND SYSTEM ASSESSMENT

We detail below both the observations from first responders as well as the important features and current limitations of our evacuation system, as noted from the table top exercise.

5.4.1 FIRST RESPONDER OBSERVATIONS

Overall, the feedback from the chief of police and his officers was positive and consisted of the following observations:

- The ability to see the layout of the buildings and surrounding areas and get a sense of the current situation was considered most valuable.
- The near real-time responsiveness of the system and the ability to see the evacuation under blockages gave the ability to get a quick assessment for taking appropriate action, dispatch first responders, etc.

Figure 5.4: Tabletop Discussions.

- Immediate knowledge and visualization of the clearly marked exits that faced a hazard was only possible with the visualization. The number of exits and required assets to deploy was possible due to the accuracy of the building representation.

- In the gas leak exercise, a review of the evacuation helped the commander quickly size up the situation (number of evacuees, exit routes, etc) and order a building evacuation. Once the hazard was located, evacuees were routed away from it by injecting suitable blockages at key points in the building.

- A redistribution of the typical evacuee populations outside the buildings resulting from rerouting of personnel is an important issue for crowd or traffic control. The ability to view the simulation and the evacuee densities (red spheres in Figs. 5.2,5.3, is thus important for planning (dispatch officers to these areas) after the event.

- In the active shooter scenario, the police chief noted that the exit utilization and congestion reports would be an invaluable tool for first responders to analyze the condition of a building and dispatch personnel.

5.4.2 EVACUATION SYSTEM EFFECTIVENESS

Overall, the system performed well and was responsive. A number of desirable features and potential improvements will be needed as we move towards real-world exercise training scenarios, and actual deployment.

- Additional work on the user interface will be needed to ensure minimum delay during a dynamically changing situation; for instance, blockages are specified one at a time vs. a lasso style interface to include multiple exits spanning a region around multiple buildings.

- A limitation of the current system is its inability to localize blockages to the exits themselves which can trap evacuees in the vicinity. Additionally, blockages need to be flexible enough to be removed when not needed; early reports might be unreliable, for instance, requiring exits to be reopened.

- The current system runs on a quad processor laptop at 2GHz. More optimal implementation and a faster system should further improve the performance with a better response time. It is to be noted that the system's near real-time response exploits the simplified building graph representations: this also makes it scalable to large collections of buildings.

- A visualization issue that is common to visual analytic systems is visual clutter and the ability to unambiguously visualize critical information. As we extend our system to incorporate tens of buildings in evacuation scenario, these issues will require careful design and representation choices. Working with actual responders will be valuable in this design process.

CHAPTER 6: CONCLUSION

In this paper we have presented a visual analytic system for situationally aware evacuations of large urban structures. The goals of this work are to provide visual analytic tools that can be used in real-world scenarios (large urban structures, and dense collections of buildings) and more importantly, be able run evacuation scenarios in the context of dynamically changing conditions. We have developed and used an LOD representation of building graphs that can be used as part of a visual analytic system for near real-time response. This in turn permits situational changes to be incorporated into the underlying models and evacuees rerouted. Finally, our visual analytic system provides recommendations through the reporting functions that can be used for effective use of scarce resources in dispatching responders to areas of need during the emergency. We evaluated our system with first responders, including the campus police chief, a senior police officer and public safety and business continuity/planning personnel. Input from these experienced personnel is invaluable. A tabletop exercise was performed with three different scenarios(gas leak in building, active shooter, and explosion) overseen by the police chief, acting as the situation commander. Overall, the system performed well, as evidenced by direct feedback from the first responders, with valuable suggestions to improve the system.

BIBLIOGRAPHY

[1] G. Andrienko, N. Andrienko, and U. Bartling. Interactive visual interfaces for evacuation planning. In *Working Conference on Advanced Visual Interfaces(AVI) 2008 Proceedings*, pages 472–473. ACM Press, 2008.

[2] N. Andrienko, G. Andrienko, and P. Gatalsky. Towards exploratory visualization of spatio-temporal data. In *3rd AGILE Conference on Geo-graphic Information Science*, 2000.

[3] B.D. Campbell and C. Weaver. Rimsim response hospital evacuation: Improving situation awareness and insight through serious games play and analysis. *Journal of Information Systems for Crisis Response and Management*, 3(3):1–15, Jul-Sept 2011.

[4] Christian J. E. Castle and Andrew T. Crooks. Principles and concepts of agent-based modelling for developing geospatial simulations. In *Centre for Advanced Spatial Analysis. University College London*, volume 110, pages 1–52, 2007.

[5] EVACNET4. http://www.ise.ufl.edu/kisko/files/evacnet/.

[6] G. Fang. Swarm interaction-based simulation of ocucupant evacuation. *2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application*, 2008.

[7] Stephen. J. Guy, Jatin Chhugani, Changkyu Kim, Nadathur Satish, Ming Lin, Dinesh Manocha, and Pradeep Dubey. Clearpath: Highly parallel collision avoidance for multi-agent simulation. In E. Grinspun and J. Hodgins, editors, *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2009)*, 2009.

[8] Y.A. Ivanov, C.R. Wren, A. Sorokin, and I. Kaur. Visualizing the history of living spaces. *IEEE Transactions on Visualization and Computer Graphics*, 110:1153–1159, November 2007.

[9] Naoyuki Kamiyama, Naoki Katoh, and Atsushi Takizawa. An efficient algorithm for the evacuation problem in a certain class of networks with uniform path-lengths. *Discrete Applied Mathematics*, pages 3665–3677, 2009.

[10] S. Kim, R. Maciejewski, K. Ostmo, E.J. Delp, T.F. Collins, and D.S. Ebert. Mobile analytics for emergency response and training. *Information Visualization*, 7(1):77–88, 2008.

[11] S. Kim, S. Shekhar, and M. Min. Contraflow transportation network reconfiguration for evacuation route planning. *IEEE Trans. on Knowl. and Data Eng.*, 20:1115–1129, August 2008.

[12] S. Kim, Y. Yang, A. Mellama, D.S. Ebert, and T. Collins. Visual analytics on mobile devices for emergency response. In *IEEE Symposium on Visual Analytics Science and Technology (VAST)*, pages 35–42, 2007.

[13] J. Liu, K. Lyons, K. Subramanian, and W. Ribarsky. Semi-automated processing and routing within indoor structures for emergency response applications. In *Proceedings of SPIE Defense Security and Sensing*, 2010.

[14] S. M. Lo, M. Liu, and Richard K. K. Yuen. An artificial neural-network based predictive model for pre-evacuation human response in domestic building fire. *Fire Technology*, pages 431–449, September 2009.

[15] Qingsong Lu, Betsy George, and Shashi Shekhar. Capacity constrained routing algorithms for evacuation planning: A summary of results. *Springer-Verlag Berlin Heidelberg 2005*, pages 291–307, September 2005.

[16] B.S. Meiguins and A.S.G. Meiguins. Multiple coordinated views supporting visual analytics. In *Proceedings of the ACM SIGKDD Workshop on Visual Analytics and Knowledge Discovery: Integrating Automated Analysis with Interactive Exploration*, pages 40–45, New York, NY, USA, 2009. ACM.

[17] Qt. http://qt.nokia.com.

[18] W. Schroeder, K. Martin, and B. Lorensen. *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*. Prentice Hall Inc., 4th edition, 2006. www.vtk.org.

[19] Shashi Shekhar and Jin Soung Yoo. Processing in-route nearest neighbor queries: a comparison of alternative approaches. In *Proceedings of the 11th ACM international symposium on Advances in geographic information systems*, 2003.

[20] J. Stasko, C. Gorg, Z. Liu, and K. Singhal. Jigsaw: Supporting investigative analysis through interactive visualization. In *Proceedings of the 2007 IEEE Symposium on Visual Analytics Science and Technology*, pages 131–138. IEEE Computer Society, 2007.

[21] J.J. Thomas and K.A. Cook. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE Press, 2005.

[22] G. van Rossum and F.L. Drake. *An Introduction to Python*. Network Theory Ltd., 2003. WWW: www.python.org.

[23] GeoVA(t): AGILE 2010 workshop + IJGIS spec.issue. http://geoanalytics.net/GeoVA(t)2010.

APPENDIX A: DETAILED PSEUDO CODE.

Algorithm 8: Build an Adjacency List Graph of the building containing all Network Generator Nodes and produce Incident Edge Count Associative Array

```
Input: (1) Node Properties: nIE, Count of Incident edges per node;
(2) Edge Properties: Length in Feet(LF);
(3) EAr: Array; [$Node1_{i...n}, Node2_{j...n}, LF$]. Array list of all nodes in spG.
Result: (1) spG(N,E): Directed Graph, N nodes with 1 or more Incident
        Edges(IE), E edges. Note: This structure is an Adjacency List
        $\{N_i : \{N_j : LF, ...N_k : LF\}\}$;
(2) Node Properties: nIE, Count of Incident Edges per node;
$IndexCounter = 0$;
foreach Node1 $N1$; Node2 $N2$; LF $lf$ in EAr do
    Initialize empty associative array $Temp$;
    if $N1$ not in spG then
        $Temp[EAr[N1][N2]] = lf$;
                    /* Add edge Node1-Node2-LF to Temp */;
        $key\_node = N1$;
        for $k$ = IndexCounter + 1 to LENGTH(EAr) - 1) do
            $edge\_node = EAr[k]$[Node2];
            $travel\_time = EAr[k]$[LF];
            if $key\_node == N1$ and $edge\_node$ not in $Temp[N1]$ then
                        /* Add edge Node1-Node2-LF at EAr[$k$] to Temp */
                $Temp[key\_node][edge\_node]] = EAr[k]$[Node2];
                if $key\_node$ and $edge\_node$ are HALLWAY-NODE then
                    nIE[$key\_node$] += 1;
                            /* Increment edge count for Node1 */;
                end
            end
        end
        spG[key-node] = tempG;
    end
    $IndexCounter$ += 1;
end
```

Algorithm 9: Build a Partial 3-edge Adjacency List Graph and unconnected nodes heaps of HALLWAY type nodes and OTHER type nodes.

```
Input: (1) spG(N,E): Directed Graph, N nodes with 1 or more Incident
       Edges(IE), E edges. Note: This structure is an Adjacency List
       $\{N_i : [N_j : LF, ...N_k : LF]]\}$;
(2) Node Properties: nIE, Count of Incident edges per nodes;
Result: (1) spMinG(N,E): Directed Graph, N nodes with 3 or more Incident
        Edges(IE), E edges as an Adjacency List $\{N_i : [N_j : LF, ...N_k : LF]]\}$;
(2) NoConnHeapPATH: Heap of path nodes with < 3 Incident Edges;
(3) NoConnHeapOTHER: Heap of path nodes with < 3 Incident Edges
connected to Stairwells or Exits, etc;
1  for Node node in Adjacency List spG do
2      if nIE[node] >= 3 then
               /* Node has 3 or more incident edges */;
3          $spMinG[node]$ = spG[$node$];
               /* Work done...add it to reduced graph */;
4          for Nodes node2 in Adjacency List spMinG[node] do
5              if HALLWAY-NODE then
                       /* For hallway nodes assume all Node2's are
                   disconnected and build arrray list of these nodes */;
6                  Add $node2$ to $NoConnHeapPATH$ Array;
7              end
8          end
9      end
10     else if NOT HALLWAY-NODE then
11         $spMinG[node]$ = spG[$node$];
               /* Work done...add it to reduced graph */;
12         for Nodes node2 in Adjacency List spMinG[node1] do
13             if HALLWAY-NODE then
                   /* For all NON hallway nodes assume all Node2's are
                   disconnected and build arrray list of these nodes */;
14                 Add $node2$ to $NoConnHeapOTHER$ Array;
15             end
16         end
17     end
18 end
```

Algorithm 10: Partial Minimum Adjacency List

```
Input: (1) spG(N,E): Directed Graph, N nodes with 1 or more Incident
        Edges(IE), E edges. Note: This structure is an Adjacency List
        $\{N_i : [N_j : LF, ...N_k : LF]]\}$;
(2) Node Properties: nIE, Count of Incedent edges per nodes;
(3) spMinG(N,E): Directed Graph, N nodes with 3 or more Incedent
Edges(IE), E edges as an Adjacency List $\{N_i : [N_j : LF, ...N_k : LF]]\}$;
(4) NoConnHeapPATH: Heap of path nodes with $<$ 3 Incident Edges;
(5) NoConnHeapOTHER: Heap of path nodes with $<$ 3 Incident Edges
connected to Stairwells or Exits;
(6) Same3EdgeGroup: List of nodes in a 3edge group around a central node;
Result: spMinG(N,E): Directed Graph, N nodes with Maximum Incident
        Edges(IE) and Minimum Nodes, E edges as an Adjacency List
        $\{N_i : [N_j : LF, ...N_k : LF]]\}$;
foreach $origin$ $in$ $NoConnHeapPATH$ do
    $node = origin$;
    $keeplooking$ = true;
    while $keeplooking$ do
        $next$ = Get Node from Adjacent Nodes to $spG[node]$;
        if $next$ $in$ $NoConnHeapPATH$ $or$ $NoConnHeapOTHER$ then
            $pathlength$ += $spG[node][next]$;
            $spMinG$ = AddNodeToReducedGraph($spG$, $spMinG$, $pathlength$,
            $node$);
             /* Found a connection move to next node in No Connection
            Heap Path Nodes */;
            $keeplooking$ = false;
        end
        else
            $pathlength$ += $spG[node][next]$;
              /* Keep walking through NON reduced Adjacency Graph spG
            */;
            $node = next$;
        end
    end
end
```

Algorithm 11: Build **Final** Reduced Adjacency List Graph by iterating No Connection heap built from NON HALLWAY Nodes

```
Input: (1) spG(N,E): Directed Graph, N nodes with 1 or more Incident
       Edges(IE), E edges. Note: This structure is an Adjacency List
       ${N_i : [N_j : LF, ...N_k : LF]]}$;
(2) Node Properties: nIE, Count of Incident edges per nodes;
(3) spMinG(N,E): Directed Graph, N nodes with 3 or more Incident
Edges(IE), E edges as an Adjacency List ${N_i : [N_j : LF, ...N_k : LF]]}$;
(4) NoConnHeapPATH: Heap of path nodes with $< 3$ Incident Edges;
(5) NoConnHeapOTHER: Heap of path nodes with $< 3$ Incident Edges
connected to Stairwells or Exits;
(6) Same3EdgeGroup: List of nodes in a 3edge group around a central node;
Result: (1) spMinG(N,E): Directed Graph, N nodes with Maximum Incident
        Edges(IE)and Minimum Nodes, E edges as an Adjacency List
        ${N_i : [N_j : LF, ...N_k : LF]]}$;
1  foreach origin in NoConnHeapOTHER do
2      node = origin;
3      keeplooking = true;
4      while keeplooking do
5          next = Get Node from Adjacent Nodes to spG[node];
6          if next in NoConnHeapPATH or NoConnHeapOTHER then
7              if origin not in spMinG then
8                  spMinG = AddNodeToReducedGraph(spG, spMinG,
                   maxedgecap, origin);
                           /* Found a connection move to next node in No
                   Connection Heap Path Nodes */;
9                  keeplooking = false;
10             end
11             else
12                 pathlength += spG[node][next];
13                 spMinG = AddNodeToReducedGraph(spG, spMinG,
                   pathlength, node);
                   /* Connect here to prevent unconnected edge; ie if the
                   node is already in reduced graph and not connected
                   already force it.  */;
14                 keeplooking = false;
15             end
16         end
17         else
18             pathlength += spG[node][next];
                 /* Keep walking through NON reduced Adjacency Graph spG
               */;
19             node = next;
20         end
21     end
22 end
```