Integrating evacuation planning into an octree-based CSCW framework for structural engineering

R.-P. Mundani & H.-J. Bungartz
Institut für Informatik, Technische Universität München, Germany

S. Giesecke
IPVS, Universität Stuttgart, Germany

ABSTRACT: In addition to classical topics such as CAD or statics' simulations, design processes in structural engineering often deal with aspects related to the later usage of buildings. One important and not negligible point of view to be examined and to be considered deals with the suitability of buildings during emergency situations as the evacuation of people in case of fire or terrorist activities. Deriving a connection graph from any arbitrary CAD model to be used for shortest-path algorithms, the graph can also be coupled to cellular automata for the simulation of people streaming to the building's exits for evacuation. These kind of simulation reveals bottlenecks of the architectural design. As geometric alterations might interfere with the building's statics, for instance, by integrating aspects of peoples' evacuation into a CSCW framework global consistency between all experts and all different tasks can be assured.

1 MOTIVATION

Within design processes from the field of structural engineering a lot of effort has to be invested to achieve global consistency for shared data, i.e. geometric models, in cooperative working environments. Beside the necessity of sufficient representations for all participating tasks, geometric models fulfilling the specifications of a statics' (CSD) or climate simulation (CFD) might fail according to the requirements for a safe evacuation of persons in case of emergencies, for instance.

In (Mundani et al. 2003, 2004a, b) and (Niggl et al. 2004) we presented a framework for CSCW and process integration for applications from the field of structural engineering, where global consistency among all participants is achieved by octree-based methods. Solving different tasks like CAD, CSD, CFD and visualisation – shortest-path algorithm for guidance systems in buildings – by this framework, we could also show that by embedding CSD into the framework any necessary computations related to modifications of the geometric model, such as the translation of a column, e.g., can be reduced and, thus, the results can be computed much faster due to a hierarchical approach (Mundani et al. 2005).

In this paper, we present the integration of evacuation planning into the framework mentioned above. When considering aspects of peoples' evacuation already during the design process, any modifications initiated to avoid bottlenecks of the underlying geometric model, such as too small doors or columns standing in the way of emergency exits, for instance, can be immediately examined for impacts on other tasks like CSD and CFD. Hence, assuring global consistency between all experts and all applications not only allows a faster processing of the entire tasks, it also provides the necessary preliminaries for setting up customized solutions for specific kinds of problems, so called problem solving environments.

2 GRAPH-DERIVATION FROM CAD MODELS FOR SHORTEST-PATH ALGORITHMS

2.1 *Derivation of a connection graph from arbitrary CAD models*

A general problem for shortest-path algorithms is the derivation of a sufficient graph from any arbitrary CAD model, if possible without manual post-processing. In our approach, we use the graphics card's z-buffer for detecting walls, corridors, and rooms before, finally, a connection graph can be derived from that information. Therefore, the model is rendered while being moved forward along the z-axis (up-vector), using an orthogonally projection to generate a stack of slices, depending on the chosen discretisation (step) width.

Regarding these slices, walls and, thus, rooms can be found easily by edge detection algorithms. Once all rooms have been identified a graph can be generated, applying distance skeleton algorithms well known from the field of image processing (Jähne

1997). If the discretisation width is chosen "fine" enough, even stairs and ramps can be detected automatically. Manual corrections aren't necessary, only elevators and escalators are difficult to identify. Here, adding an edge between the corresponding nodes and marking it with an attribute such as ELEVATOR or ESCALATOR, resp., for further processing is everything to be done. For further information see (Drexl 2003).

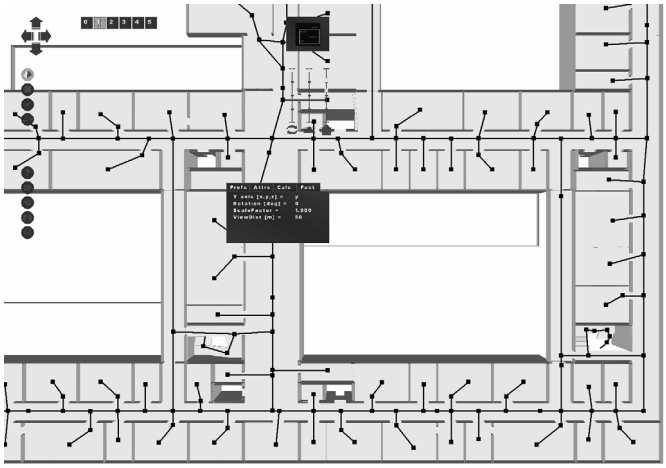

Figure 1. A sample graph – displayed within a tool for graph manipulation – as automatically derived from the CAD model of the computer science building of Universität Stuttgart.

2.2 *Shortest-path algorithms and visualisation*

Once the graph has been generated, applying DIJKSTRA's algorithm (Dijkstra 1959) to the weights stored to the graph's edges – representing the distance between the two nodes bordering an edge – calculates all shortest routes between any two arbitrary nodes. An online visualisation, as VRML application (Fig. 2), for instance, can now be used as guidance system to find any room or location inside a building.

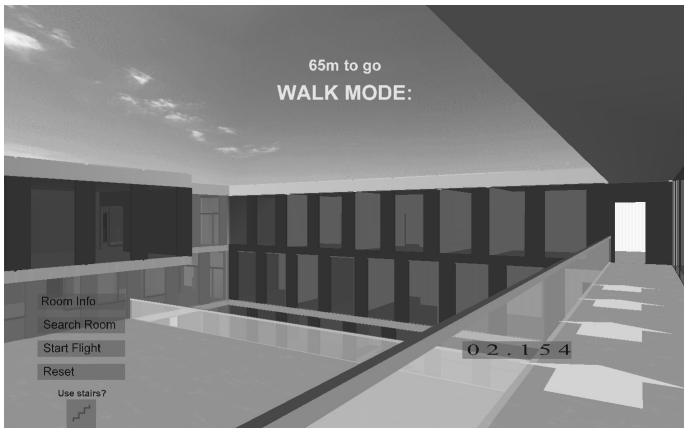

Figure 2. A sample guidance system – here shown as VRML application – based on the graph derived from the computer science building of Universität Stuttgart.

Some more sophisticated routing can be achieved when allowing a dynamic (such as time-dependent) change of the weights – setting them to infinity, e.g. – to prevent the usage of certain routes. Sometimes the route through the canteen might be the shortest, whereas it's not a good idea to choose it at noon during the week. In combination with databases further information retrieval is possible, such as the usage of rooms or names and phone numbers of persons sitting in an office.

The automatic derivation of a graph is also the basis for our evacuation planning where people should find their own way to the nearest emergency exit. As visualisation and shortest-path algorithms are already integrated into the framework, a graph-based pedestrian flow can be easily integrated, too. Thus, peoples' evacuation can be made a part to be considered when dealing with a building's global consistency, allowing some more detailed studies of design processes.

3 EVACUATION SIMULATION

3.1 *Cellular automata*

A lot of research has been done in the field of pedestrian flow and evacuation dynamics (Schreckenberg et al. 2002). For the simulation of flow two different approaches have evolved: continuous and discrete simulation. While the first one (in general) is based on a precise physical model, it often lacks due to large computation times and, thus, is only of limited usage for an evacuation planning in real time. Beside this, for simulating pedestrian flow a precise model is not always necessary. In (Thiele 2001) we have shown that the continuous simulation of pedestrian flow based on the NAVIER-STOKES-equations is far beyond the usability in real time or online applications. Here, online means interactive, thus, an expert has the possibility to change the geometric model while a simulation for studying any impacts related to this manipulation runs.

To overcome that problem, discrete event simulation (Banks et al. 2000) – less precise than continuous simulation but nevertheless sufficient enough for describing pedestrian flow – allows faster computations and, thus, the processing in real time or online applications. Another technique, meanwhile state-of-the-art, are cellular automata that allow fast and efficient computations of peoples' movement. Each cell of such an automaton represents the place for a person – if not empty – being "moved" to one of the neighbouring cells by the state transition function

$$\delta: Q^N \rightarrow Q \qquad (1)$$

with a set of states Q and a finite amount of neighbours N. Applying this transition function to all cells simultaneously and removing all persons that have reached a final state, i.e. a node representing an exit, an evacuation simulation is done by successively repeating this step until all cells are empty. For further information on cellular automata see (Wolfram 1994), for evacuation simulation based on

cellular automata see (Hanisch et al. 2003), for instance.

3.2 *Objectives for evacuation dynamics*

In most cases, engineers are primarily interested to check if buildings, planes, or ships fulfill certain security requirements in case of emergencies when processing an evacuation simulation. That means, it has to be proven that it's possible for all people to leave within a given amount of time. In (Klüpfel et al. 2000) it is shown how security requirements concerning peoples' evacuation can be proven for passenger ships, for instance.

In our approach, the main focus does not tend into that direction of proving time-related security requirements, but to reveal any kind of bottlenecks related to the underlying geometric model. By integrating our approach into the framework mentioned above, those bottlenecks can be detected in early steps. Thus, impacts due to geometric modifications of the model on tasks such as CSD or CFD, for instance, can be considered and consistency among all participating experts or applications, resp., can be achieved.

4 GRAPH-BASED PEDESTRIAN FLOW

4.1 *Basic approach*

Here, we want to simulate online pedestrian flow in arbitrary buildings, thus, information about all corridors, stairs, escalators, elevators, doors and exits is necessary. This information is almost given within the connection graph derived from the geometric model. Corridors and stairs are represented as edges, the latter ones as a combination of edges and nodes (Fig. 3). The distance between two adjacent nodes is given by their common edge's weight, automatically retrieved from the geometric model, too. Escalators and elevators can also be represented as single edges with certain attributes, but a different processing – in the sense of additional time for waiting or transportation – has to be considered for pedestrian flow. Both are not included in the current version and will follow within some later extension.

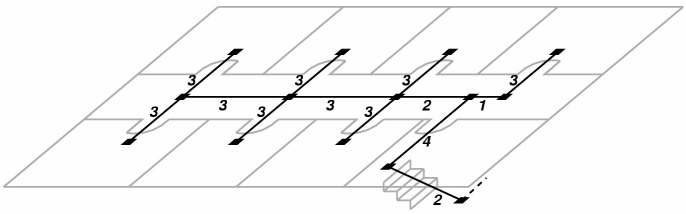

Figure 3. Connection graph with edge weights representing the distances – as measured from the CAD model – between two nodes; the edges on the lower right-hand side are from a staircase to the next floor.

The connection graph can now directly be used as a basis for cellular automata. Therefore, each edge represents one (1-dimensional) cellular automaton with n cells, where n is calculated in the following way. Assuming one person needs the space of a circle with diameter 0.8m for standing on the corridor (information about the height is not necessary as only rooms with a height of more than 2m have been considered within the graph derivation), thus, with

$$n = \lfloor (length / 0.8) + 0.5 \rfloor \qquad (2)$$

for an edge with weight 3 (length of 3 meters) this results to a discretisation of 4 cells, for instance. The corresponding cellular automaton has 4 cells where each cell can represent one person.

As persons should move through the building for evacuation they have to pass over different edges connected via common nodes. That means, tracking one person's route it has to be shifted between the different cellular automata corresponding to these edges. The transition from one cellular automaton to another is done over the common node that connects both of them or the corresponding edges, resp. Hence, a person reaching the border cell of one automaton – left or right border of the 1-dimensional array representing the automaton's cells – can decide in which direction to go (for the case this node connects more than two edges) and, thus, is removed from one cellular automation and inserted in the next one, if possible, by a state transition.

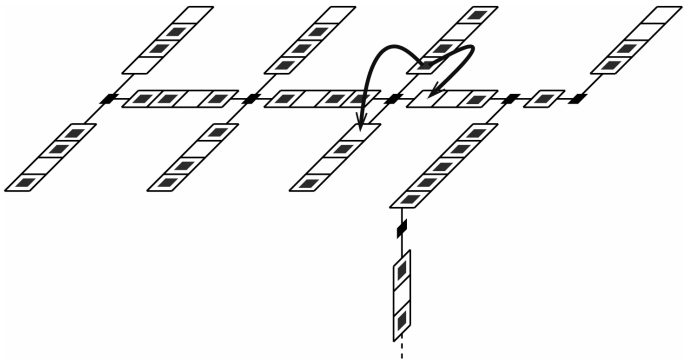

Figure 4. Sample graph-based cellular automata for the connection graph from Fig. 3 with some random states (filled squares indicate people); for one cell the possible directions for a transition are shown (arrows).

The decision which direction to choose in case there are several possibilities is driven by a shortest-path algorithm. As in our scenario people want to evacuate the building, a state transition always looks for the neighbouring cellular automaton or edge, resp., that lies in the direction of the shortest exit—always under the assumption people are following the exit signs, chaotic behaviour is not considered so far. If this transition is not possible as the designated cell is not empty the state transition looks for the next possibility. Only in case none of them is applicable the state transition decides either to wait or to "run back" and try another exit. It's obvious that a wait causes all other people behind also to wait, thus, a congestion appears. Congestions also appear

at nodes that are connected to several edges, such as a T-crossing, e.g., as always only from one side a state transition can be done. To prevent the "starvation" of one ore more cellular automata, a round robin principle assures an equal distribution of state transitions to all cellular automata connected to a node. In reality this can often be seen at road works on highways where cars (should) merge together from several lanes to one lane like a zipper.

4.2 *Further geometric considerations*

So far, no geometric information about the dimensions of corridors or the width of exits have been taken into account for the evacuation simulation. For a realistic pedestrian flow further data is necessary. Beside the length of an edge (given by its weight) and, thus, the amount of cells n also the "capacity" and "throughput" of an edge are important values. While the first one determines the amount of people that can walk next to each other on the corridor represented by the corresponding edge, the latter one specifies the amount of people that can pass over that edge in a certain amount of time. The capacity can easily be retrieved from the geometric model by examining the width of corridors. Again, Equation 2 computes the correct amount of cells m substituting the width instead of the length (Fig. 5). To simplify the cellular automaton it is still represented by a 1-dimensional array. In contrast to before, each cell can now store more than one person, thus, the cellular automation has n cells where each of them can store m_i persons—$i \in [0, n-1]$.

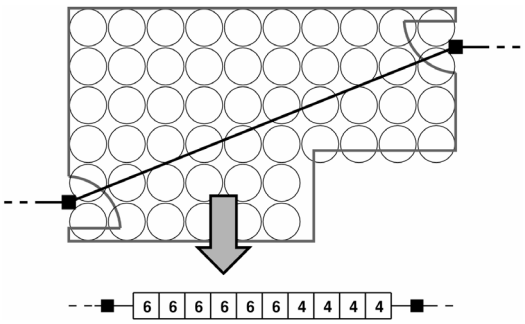

Figure 5. For a sample room the possible amount of persons – according to Equation 2 – is indicated (circles) as well as the corresponding cellular automaton for the drawn edge. It consists of 10 cells where the ones representing the left part of the room can store up to 6 persons and the ones representing the right part up to 4 persons.

Calculating the throughput is a bit more complicated. The amount of people that can enter or leave a room depends on its exits. An office room, for instance, has one exit in the general case, thus, one person can enter or leave. A hallway might have large exits on two opposite sides, thus, several persons can enter or leave at the same time. Hence, for calculating the throughput all available information according to a room's exits has to be considered. From the geometric model the width of a door can easily be obtained. Unfortunately, the geometric model doesn't tell anything about the direction a door can be opened (to the left or right and, thus, in direction of an escape route or against it) or if it's locked at certain times. Here, some context-based data such as attributes for parts of the geometric model could be helpful. For the current state, our algorithm only considers the width of doors or exits, resp., to calculate the throughput. More sophisticated approaches will be examined in future works.

The throughput is expressed as persons/time that can enter or leave. In the sense of a cellular automaton that means how many persons can be moved from one automaton to a neighbouring one over a common node within one state transition. It's value is determined by Equation 2 when substituting the width of a door or exist as obtained from the geometric model. For the example shown in Fig. 5 the throughput is 1 person/transition for both doors. Obviously, this not too much because it would take at least 52 state transitions to evacuate the full room.

4.3 *A prototypical implementation*

To achieve some first results and to test our approach of a graph-based pedestrian flow we have made a prototypical implementation. Our test scenario is the model of a small office building with two floors (basement and first floor). Within this scenario we start from a random distribution of persons inside the building, placing some of them in the office rooms and some on the hallways. There exists one exit to leave the building and two staircases to come down from the first floor to the basement.

When starting the simulation, in each step the algorithm determines for all persons the shortest route to the exit according to their current positions. Once it is known in which direction to go (forward or backward) or on which edge to change at crossings (nodes), the corresponding state transitions for all cellular automata are initiated. Finally, all persons that have reached a final state – i.e. a node representing an exit – are removed and all steps are successively repeated until all persons have left the building (see Fig. 6).

In some cases it might not be possible to evacuate all persons as parts of the building are destroyed and cannot be accessed. This can easily happen if one thinks about fire or – unfortunately not negligible in our times – terrorist activities. To prevent the algorithm from using certain parts of the building during the simulation the corresponding edges or weights, resp., have to be set to an infinite value, thus, the shortest-path algorithm doesn't take these edges into account when calculating the shortest route. As we are interested in an online simulation, in a future release users will be able to select single edges during the evacuation simulation and to disable them manually to pretend an emergency – fire or the detonation

of a bomb – to perform studies about the impact on security related issues.

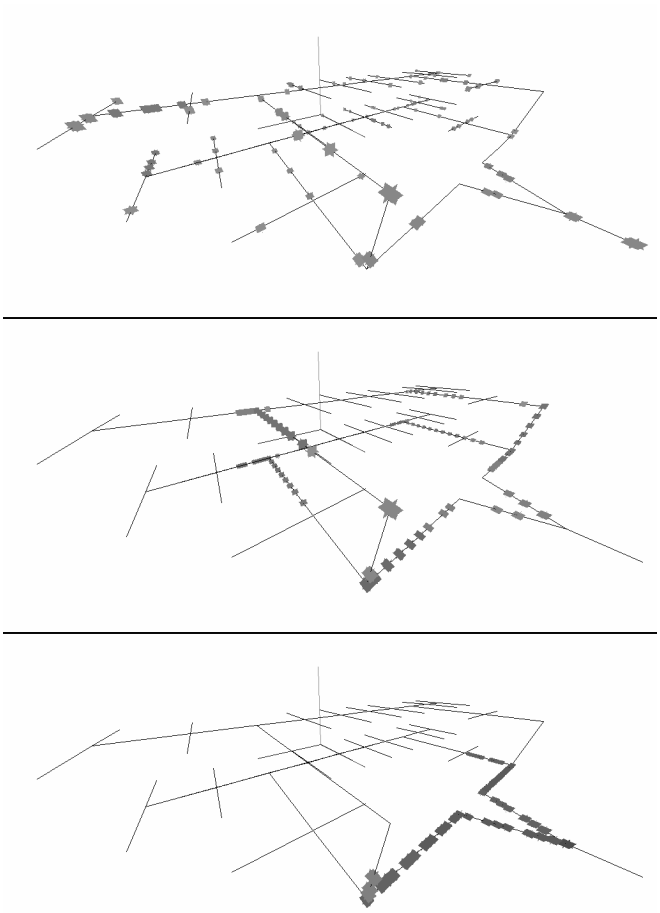

Figure 6. Sample evacuation simulation for an office building with two floors: The topmost picture shows the initial setup (squares are indicating people), the middle picture the state after some transitions, and the lowermost picture the state close to the end. The darker squares are drawn, the more people have to wait before they can move on as it can be seen near the exit (right-hand side).

Nevertheless, disabling an edge and, thus, making parts of the building not accessible might result to isolated blocks where people are running around but they are not able to leave. Such a block has to be removed from the current computations, so the simulation can stop if the rest of the building is empty. Otherwise it would loop forever, waiting that all people have been evacuated. The problem is to automatically identify an isolated block and to stop computations of the corresponding cellular automata.

One possibility determining an isolated block is to try to reach an exit from both sides of the disabled edge. Therefore, from both of the edge's nodes a recursive search is initiated, either stopping when an exit was found or all edges and nodes behind that specific node have been visited. For the case no exit could have been reached all edges and nodes on that side belong to an isolated block. Thus, to prevent further computations and to assure that the simulation stops when all people from the rest of the building have been evacuated, all subsequent edges on the isolated side are disabled, too, and the cells of the corresponding cellular automata are set to zero.

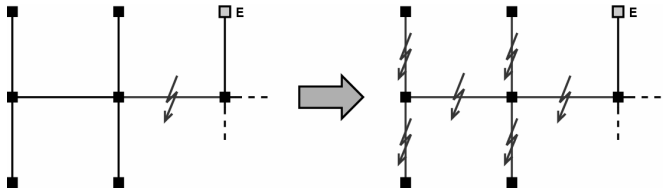

Figure 7. For both nodes of the disabled edge a search for a reachable exit (highlighted node with capital letter 'E') is performed. If no exit can be reached all subsequent edges behind that node are disabled, too.

5 INTEGRATION OF EVACUATION PLANNING INTO THE CSCW FRAMEWORK

By now, we have shown a graph-based pedestrian flow as stand-alone application for evacuation simulation. To exploit the full potential of this approach and, thus, to add a new characteristic to the cooperative working environment discussed in this paper, evacuation planning was integrated into our octree-based CSCW framework. Hence, the results of evacuation planning can be considered in other tasks, revealing any impacts of geometric modifications on the global consistency initiated due to security requirements.

Starting from a CAD building model (IFC) all geometric and related data is stored to a database server. Any access to this server is controlled via services to check-out/in parts from the geometric model to/from the local workspace. Any (local) modifications written back to the server have to pass a collision detection before stored to the database or being rejected in case of failure. Thus, global consistency among all participating experts or tasks, resp., can be assured. Further information about this framework can be found in (Mundani et al. 2004b) and (Niggl et al. 2004).

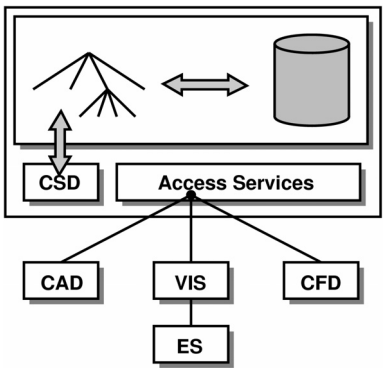

Figure 8. A sketch of our CSCW framework: All shared data is stored to a central database server assuring global consistency via octree-based methods. Already embedded into that framework is the structure analysis (CSD), all other integrated tasks can access the data via provided access services. Here, the component for visualisation and shortest-path algorithms (VIS) is extended by an evacuation simulation (ES), based on the automatically derived connection graph from the central building model.

Another important property related to global consistency deals with impacts or side effects between all participating tasks. A modification done within one application might interfere with another application – not necessarily based on the geometry itself – and, thus, might entail further modifications until all requirements – such as a room's air conditioning, for instance – are satisfied. The earlier these properties are taken into account, the easier global consistency can be achieved. Integrating evacuation planning into the design process from the start prevents additional iteration cycles at the end when the building evolved so far doesn't come up to security related expectations. Figure 8 depicts a schematic sketch of the current framework also considering evacuation planning.

Whenever one engineer now changes the central data all subsequent applications can immediately react on that new scenario and check if there occur any problems. In case of the evacuation planning another engineer responsible of that can not only check the suitability of the current geometry according to security requirements, he can also reveal bottlenecks when studying further scenarios different from that standard evacuation case. Therefore, he can block certain routes as described above and see if a secure evacuation is still possible. Any suggestions or changes according to more or wider doors, additional staircases, or the translation of columns in huge areas, for instance, can then again be considered within the other applications immediately. Thus, any inconsistent state is easy to detect and necessary measures can be initiated.

Nevertheless, reacting to a different scenario entails manual activities by some engineer. If all tasks would be embedded into the framework, the corresponding applications would run automatically and, thus, only the simulation results would have to be checked. This can safe a lot of effort, as shown in the case of statics' simulation (Mundani et al. 2005) where a much faster and much more efficient processing is possible—especially when one engineer intends to locally study the behavior of different variants without conflicting the global consistency.

6 CONCLUSIONS

In this paper, we have shown how a connection graph for shortest-path algorithms can be coupled to cellular automata to simulate pedestrian flow in arbitrary CAD models for evacuation planning. Furthermore, this evacuation planning can be used to study different emergency scenarios by making parts of the geometric model inaccessible and, thus, revealing bottlenecks of the architectural design. By integrating the evacuation planning into a framework for cooperative work, the results of an evacuation simulation can immediately be used within other tasks to achieve global consistency among all participating experts and applications, resp.

A next step will comprise the embedding of evacuation planning into this framework as already done for the statics' simulation, thus, providing a so called problem solving environment that allows a faster and more efficient treatment of design processes from the field of structural engineering.

REFERENCES

Banks, J., Carson, J.S., Nelson, B.L. & Nicol D.M. 2000. *Discrete-Event Systems Simulation*. Prentice Hall.

Dijkstra, E.W. 1959. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik* 1: 269-271.

Drexl, T. 2003. *Entwicklung intelligenter Pfadsuchsysteme für Architekturmodelle am Beispiel eines Kiosksystems (Info-Point) für die FMI in Garching*; Diplomarbeit, Fakultät für Informatik, Technische Universität München.

Hanisch, E., Tolujew, J., Richter, K. & Schulze T. 2003. Online Simulation of Pedestrian Flow in Public Buildings. In Ferrin, D.M. et al. (eds.), *Proc. of the 35th Conf. on Winter Simulation*; New Orleans, Louisiana, 7-10 December 2003.

Jähne, B. 1997. *Practical Handbook on Image Processing for Scientific Applications*. CRC Press.

Klüpfel, H., Meyer-König, T., Wahle, J. & Schreckenberg, M. 2000. Microscopic Simulation of Evacuation Processes on Passenger Ships. In Bandini, S. et al. (eds.), *Proc. of the 4th Int. Conf. on Cellular Automata for Research and Industry*; Karlsruhe, 4-6 October 2000.

Mundani, R.-P., Bungartz, H.-J., Rank, E., Romberg, R. & Niggl. A 2003. Efficient Algorithms for Octree-Based Geometric Modelling. In Topping, B.H.V. (ed.), *Proc. of the 9th Int. Conf. on Civil and Structural Engineering Computing*; Egmond aan Zee, 2-4 September 2003.

Mundani, R.-P. & Bungartz, H.-J. 2004a. An Octree-Based Framework for Process Integration in Structural Engineering. In Callaos, N. et al. (eds.), *Proc. of the 8th World Multi-Conf. on Systemics, Cybernetics and Informatics*; Orlando, Florida, 18-21 July 2004.

Mundani, R.-P. & Bungartz, H.-J. 2004b. Octrees for Cooperative Work in a Network-Based Environment. In Beucke, K. et al. (eds.), *Proc. of the 10th Int. Conf. on Computing in Civil and Building Engineering*; Weimar, 2-4 June 2004.

Mundani, R.-P., Bungartz, H.-J., Rank, E., Niggl, A. & Romberg, R. 2005. Extending the *p*-Version of Finite Elements by an Octree-Based Hierarchy. Submitted to the *16th Int. Conf. on Domain Decomposition Methods*; New York, 12-15 January 2005.

Niggl, A., Romberg, R., Rank, E., Mundani, R.-P. & Bungartz, H.-J. 2004. Octrees for Cooperative Work in a Network-Based Environment. In Dikbas, A. et al. (eds.), *Proc. of the 5th European Conf. on Product and Process Modelling in the Building and Construction Industry*; Istanbul, 8-10 September 2004.

Schreckenberg, M. & Sharma, S.D. (eds.) 2002. *Pedestrian and Evacuation Dynamics*. Springer-Verlag.

Thiele, E. 2001. *Simulation des Verkehrsflusses in einem Straßennetz durch Strömungssimulation in zweidimensionalen Berechnungsgebieten*; Diplomarbeit, Fakultät Informatik, Elektrotechnik und Informationstechnik, Universität Stuttgart.

Wolfram, S. 1994. *Cellular Automata and Complexity*. Perseus Books Group.