Duncan Cavens, Christian Gloor, Johannes Illenberger, Eckart Lange, Kai Nagel, W. A. Schmid

Distributed intelligence in pedestrian simulations

Conference paper | Submitted manuscript (Preprint)
This version is available at https://doi.org/10.14279/depositonce-8265

Cavens, D.; Gloor, C.; Illenberger, J.; Lange, E.; Nagel, K.; Schmid, W. A. (2007). Distributed intelligence in pedestrian simulations. Pedestrian and Evacuation Dynamics 2005, 201–212.
https://doi.org/10.1007/978-3-540-47064-9_18

Distributed intelligence in pedestrian simulations

D. Cavens[1], C. Gloor[2], J. Illenberger[3], E. Lange[4], K. Nagel[3], W. A. Schmid[1]

[1]ETH Zürich, Switzerland
[2]Gloor Consulting, Switzerland
[3]Technical University of Berlin, Germany
[4]University of Sheffield, United Kingdom

In order to accurately simulate pedestrian behaviour in complex situations, one is required to model both the physical environment and the strategic decision-making of individuals. We present a method for integrating both of these model requirements, by distributing the computational complexity across discrete modules. These modules communicate with each other via XML messages. The approach also provides considerable flexibility for changing and evolving the model. The model is explained using an example of simulating hikers in the Swiss Alps.

1 Introduction and Motivation

An important question in pedestrian simulation systems is the determination of the direction in which the pedestrians are heading. For the investigation of simple geometrical structures, it is sufficient to give the pedestrians pre-computed and fixed directions, which translate to a desired velocity vector for each pedestrian which is constant in time. Somewhat more advanced are evacuation simulations, which are solved either by using potentials or by using simple rules that combine searching and herd behaviour.

As one moves towards more complex spatial environments and social situations, a correspondingly more complex approach is required to modelling agents' desired direction. This is required for models that simulate, for example, how pedestrians explore a museum or a department store, or how they move around in a crowded urban park at lunchtime. In these situations, agents, like the individuals they represent, need to be able to adapt their desired directions in response to their surrounding environment and the activities of other agents.

In general, a mobility simulation consists of at least two components: the simulation of the agents' interactions with the physical world, and the simulation of the agents' strategic or mental decision-making [1]. The first component, what we call the physical simulation, deals with how a pedestrian adapts its movement to accommodate obstacles and physical constraints in its immediate environment (i.e. strategies to avoid a group of other pedestrians that are between the agent and its destination.) The second component, the agents' decision-making, models the agents' goals and strategies at a broader and temporal scale (i.e. the selection of an agent's destination from a set of similar alternatives.) While there is some overlap between the two components, for the purposes of this paper the agents interaction While both of these components, plus their interplay, are important to making a realistic pedestrian simulation, there has been comparatively little research into how to make the two components work together [2].

While the primary purpose of the work presented here seeks to integrate the modelling of pedestrians' physical movements with their strategic decision-making, it was also triggered by a research project that simulates the reaction of hikers to changes in the landscape. This created additional demands on the described system, requiring that the system be able to model and simulate the following aspects:

- **Large scale:** The study area is typically used for extensive day hikes. This implies an area of at least 25 km x 25 km, and requires the simulation of several thousand pedestrians per day.
- **Sophisticated mental models:** The evaluation of a landscape (both aesthetically and from a functional perspective) by recreational users is a process that is not well understood. This implies the use of a flexible method in which very different mental models can be tested.
- **Distributed Computation:** Since variability of experiences over the course of a day seems to have a strong influence on hiker satisfaction, a computational method that automatically evaluates sequences of views is needed. Since this is a time-consuming computation, this implies the use of distributed computing where several view analyzers can run on different computers.

We present an approach that satisfies these goals.

2 Overview of the Approach

Our method consists of dividing the simulation into distinct modules. These modules interact with each other via network messages. Each module has a distinct role in the overall simulation system, but can be classified into the following broad categories:

- **Mental modules** simulate the processes that go on in peoples' heads. These modules determine how an agent can best fulfil its goals and expectations, based on their experience on previous simulation runs. These modules also receive events from the other modules, in order to refine their knowledge of the area being simulated.
- The **physical simulation** (in this case a pedestrian simulation) executes the plans of all involved agents simultaneously. The module is responsible for modelling how the agents react to their physical environment such as slow-downs due to congestion or path characteristics. While the mobility

simulation is running, it constantly emits messages (called events) stating the status of each agent. Most of these events are simply status messages (containing the agent's location), but some messages contain additional information about the agent's surrounding environment.

- There are secondary **analyzer modules** that read the event stream, compute secondary information, and re-insert that secondary information into the same event stream.
- There are additional **control modules** that coordinate communication between the other modules and keep track of the overall state of the simulation.

The simulation is designed to run over many iterations, during which the agents „learn" about their environment. Initially, the agents are assigned characteristics and non-spatially specific goals, but have no knowledge of the physical characteristics of the simulated area. These characteristics and goals are generated externally to the simulation and fed to the Agent Database.

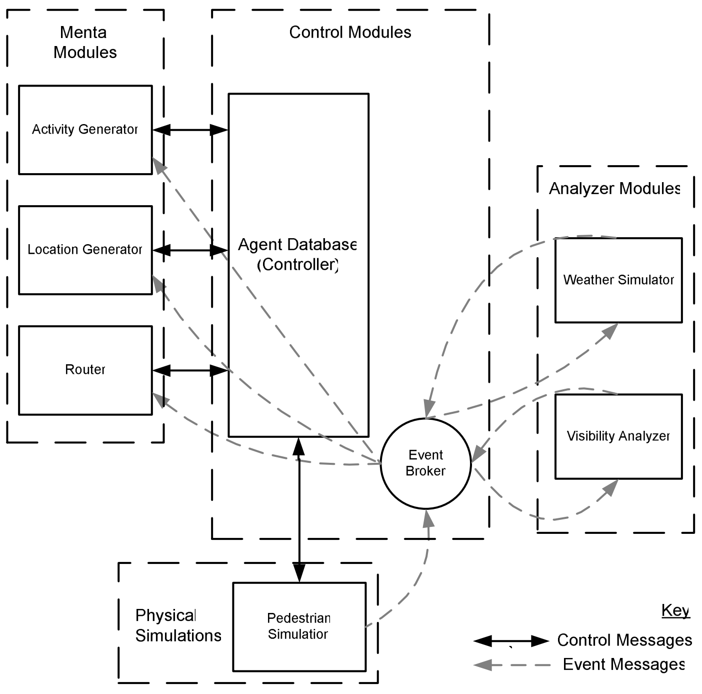

Figure 1: Overview of the Simulation System. Each Module can be implemented as a separate executable if required.

At the beginning of each simulation run (in this case representing a single day), the Agent Database, with the assistance of the mental modules, generates for each agent a plan that the simulation system expects is the most likely to fulfil the particular agent's goals and expectations. Once all plans have been elaborated, the Agent Database submits these plans to the physical simulations which simultaneously execute them.

During the model run, the physical simulations broadcast events to the rest of the simulation. These events include information about the location of the agents and any experiential information that is available (i.e. indicating that the agent has encountered a steep hill, or is in a congested area.) The mental and analyzer modules listen to events being broadcast by the physical simulation. This information is used by the mental modules to refine their knowledge of the physical environment and generate better plans in subsequent model runs. For example, they might note that an agent sees nothing but trees while the agent is interested in sunshine. On subsequent model runs, the agent will search for a different hiking path that provides more open areas.

At the end of a simulation run, the control module determines if the agents have achieved their goals and expectations using the current plan. If not, the mental modules are asked for a new plan. In order to ensure that agents are able to discover new locations, a degree of randomness is used to determine the agents' choices (the random factor decreases over many simulation runs.)

3 Modules

In order to further elucidate the major concepts, the following are descriptions of key modules in the simulation system. More complete descriptions are available in [3] and [2].

3.1 Agent Database / Controller Module

The Agent Database fulfils two major functions within the simulation system: it maintains the master list of agents in the simulation and co-ordinates the rest of the modules.

As part of the system initialization, the agent database loads in a synthetic population of agents. This population, defined in an externally generated XML file, describes each agent's individual characteristics. This includes the agents' physical constraints (such as fitness levels) as well as their goals and expectations. At this stage, the goals and expectations are non-spatial: they are simply a list of activities (in the case of the Hiking simulator, these include hiking, eating at a restaurant, etc.) and their desired durations.

Before each simulation run, the Agent Database determines if the agent has a plan that meets its expectations. If not, the Agent Database requests that the Mental Modules (Activity Generator, Location Generator and Router) provide suggested routes that potentially fulfil the agent's goals. In these transactions, the Agent Database acts as an "ignorant" broker: it contains very little knowledge about the simulated environment or agent logic.

At the end of this elaboration process, the Agent Database contains a plan for each agent has a plan that represents the overall system's current best solution to the agent's goals and expectations. (Over the course of many simulation runs, this solution will generally improve as the agents have the opportunity to explore the simulated landscape and discover more appropriate solutions.)

A simplified representation of an agent's plan is contained in figure 2.

Once the Agent Database has received elaborated plans for each agent, they are submitted to the Physical Simulation for execution. At this point, the Agent Database assumes more of a "controller" role, primarily ensuring that the various modules are able to keep up with each other. It does this by throttling the entire simulation (by requesting that the physical simulation wait after each time step) if some of the modules not able to process events and/or requests quickly enough.

```
<plan agent =”1” plan_id=”1” >
    <activity id=”1-1” type=”enter_simulation” time=”324000”>
        <location id=”1-1-1” type=”parking_lot” x=”512432.2” y=”508343.5” />
    </activity>
    <activity id=”1-2” type=”hike” suggested_duration=”3600”  >
        <waypoint id=”1-2-1” type=”node” node_id=”1246” x=”512438.5 y=”5078334.3” />
        <waypoint id=”1-2-2” type=”node” node_id=”1247” x=”512436.0 y=”507820.9” />
        (...)
        <location id=”1-2-1” type=”hike_waypoint” x=”512450.0” y=”508012.3” />
        <waypoint id=”1-2-12” type=”node” node_id=”1281” x=”512470.5 y=”507950.3” />
        <waypoint id=”1-2-13” type=”node” node_id=”1284” x=”512322.5 y=”507912.8” />
        (...)
    </activity>
    <activity id=”1-3” type=”eat” duration=”1800” >
        <location id=”1-3-1” type=”restaurant” x=”514432.0” y=”505323.0” />
    </activity>
</plan>
```

Figure 2: Simplified XML Plan. The simulation system dynamically generates a new plan for each agent every day. The plan is used by the Physical Simulation Module to direct the agent's movements over the course of a simulation run.

3.2 Physical Simulations - Pedestrian Simulation Module

The Pedestrian Simulation Module models how the agents interact with the physical environment. This includes interactions with other agents (such as avoiding collisions) and interactions with the physical world (i.e. slowing down when climbing up steep hills.)

Because of the need for realistic arbitrary movement, the pedestrian simulation module uses a hybrid approach adopted from Mauron [4]: the module uses a continuous representation of geographic space, but also uses a network representation of available paths as a guide for the agents' movements. This means that agents are free to move anywhere in the model, but are more likely to walk along existing paths and trails.

The pedestrian model uses a force-based approach, with strong forces along the path trajectories and weak forces toward the middle of the path which encourage agents to follow the trails. Additional forces are generated by neighbouring agents and inanimate objects near the agent. The force model was calibrated based on video data of pedestrian movement and provides very realistic movement patterns.

A continuous space implementation requires, in general, considerably more computational resources than a network-based approach, particular for areas as large as our study area (over 600 km^2). However, the particular nature of hiking areas means that the study area is very sparsely populated with agents at any given time, and they tend to congregate within a much smaller subset of the total area available to them. In order to reduce the computational demand, the pedestrian simulation module takes advantage of these features and uses lazy-initialization and caching techniques to ensure that only a small proportion of the total area is loaded into memory at any given time[3]. As a result, the physical simulation module can easily fit within the resources available on standard desktop PCs.

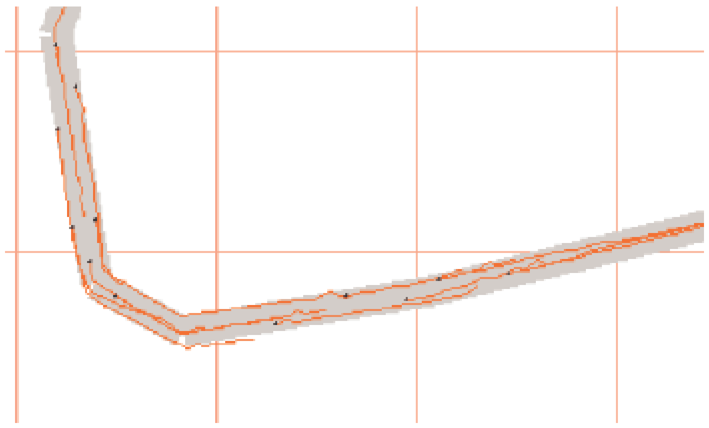

Figure 3: Hybrid Continuous Space Model: Traces of Simulated Pedestrians following a path while avoiding each other.

From the perspective of the Pedestrian Simulation Module, the agent's plan consists of a series of waypoints that it needs to traverse over the course of a simulated day. The plan also indicates where and when the agent should enter the simulation, and if it should wait at any given waypoint (such as at a bench for a rest). Once the simulation module has received all of the agent plans, it simultaneously executes these plans for all agents in the simulation.

While the simulation is being executed, the module broadcasts messages describing the agents' interactions with the physical world to the event stream. These messages include:

- the location and orientation of each agent,
- if the agent has encountered congestion,
- information about the steepness of the terrain, and
- trail condition information.

The physical simulation uses additional GIS data, provided as a series of raster layers, to provide information such as the steepness and trail conditions. These two particular kinds of information are also used by the simulation, in conjunction with the agent's particular characteristics (such as agent fitness), to determine the agents' velocity. This is calibrated based on hiker data collected in other recreational areas[5].

3.3 Mental Modules

As described in section 2**,** part of the role of the mental modules is in elaborating plans. More importantly, however, is the mental module's key roles in observing and interpreting the agents' environment. The Mental Modules are where all agent learning takes place: the modules receive events from the physical simulations, which describe the agents' experiences, and use them to inform their suggested agent plans.

Each mental module is responsible for a different spatial and temporal scale in the plan-generating phase:

- The *activity generator* generates an ordered chain of activities based on the agent's goals and expectations.
- The *location generator* assigns specific locations to this activity chain, including key points in the middle of mobile activities such as hiking.
- The *router* generates specific routes between the locations specified by the location generator.

The three mental modules share a lot of similarities (they are implemented as closely related software classes). Each:

- maintains an internal representation of all possible agent choices at their respective spatial scales (for the router and location generators these representations are akin to a geographic map of nodes and links, while the location generator's is simply a list of possible permutations, in keeping with its non-spatial nature.)
- listens to the event stream generated by the physical simulation and summarizes this information into distinct "experiences". These experiences are stored based on the spatial and temporal scale of the module (i.e. per

activity pair in the case of the activity generator, per location pair by the location generator and per node-pair for the router).

- Contains an evaluator function that scores these previous experiences based on a particular agent's expectations (i.e. while a hike may be too steep for another hiker, it might be exactly what another hiker is seeking.)

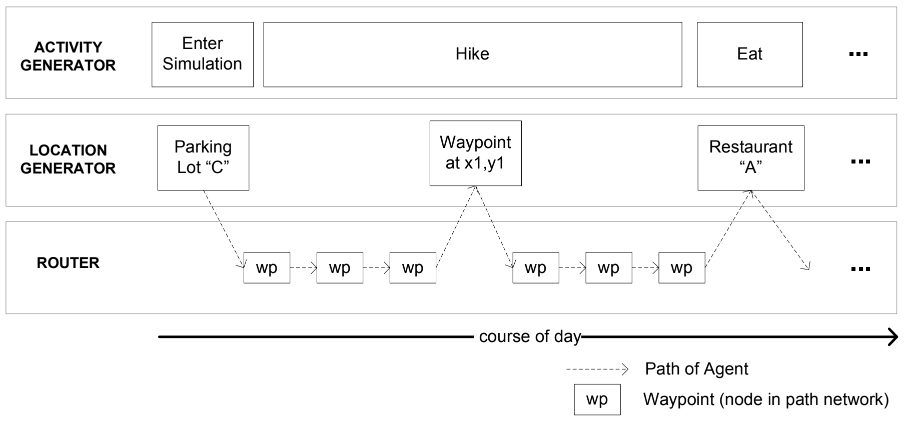

Figure 4: Schematic representation of each Mental Module's contribution to the plan generation process. As the Agent Database queries the Modules from top to bottom, the agent's plan gains increasing resolution.

In the current implementation, only the location generator and router have been fully implemented: as an interim measure the activity generator uses some simple heuristic rules to create plausible activity chains.

Example Mental Module: Router

The router operates at the smallest spatial scale- it suggests routes between locations provided by the location generator. In order to do so, the router is preloaded with an internal network of nodes and links which represent the available paths within the simulated area. During a simulation run, every time an event is received indicating that an agent has passed a node (and therefore entered a link), the router begins collecting the events and stores them until a subsequent event indicates that the agent has left the link. These events, which indicate landscape features (such as the quality of a view, type of landscape or terrain difficulty) or human factors (congestion, trail closed etc.) are then summarized by the router and stored, along with the time the agent entered the link and how long it took for the agent to walk the link. Over time, as other agents walk the same route, their experiences are also associated with that particular link.

When asked to elaborate an agent's plan, the router parses the given plan, and extracts all of the location pairs. For each of these pairs, it computes the best available route along the path network according to the agent's individual characteristics. It computes this by first converting each link's set of experiences to a numerical value using a generalized cost function calibrated to the agent's goals and expectations (as the precise implementation of this function is part of ongoing research, details are to follow in a forthcoming publication.) An optimal path between the two locations is then computed using a modified Djikstra's algorithm[6]. Although our standard implementation uses the typical Djikstra algorithm, its heritage as a shortest path algorithm means that it is unsuited for modelling recreational activity: as at least part of the attraction for recreational users is the "getting there", a more complex algorithm is required and is currently under development.

3.4 Analyzer Modules

One particular strength of the described modelling framework is the ability to create new modules that model external factors and/or interpret the agents' environment in different ways. One does this by creating new analyzer modules that listen to the event stream (broadcast either by the physical simulations or by other analyzer modules). The analyzer modules can then insert additional information as events into the event stream, where they can be interpreted by the various Mental Modules, if appropriate. While the mental modules do need to be modified to be able to react to any additional information provided by the Analyzer Module, the overall simulation approach means that only minor changes need to be made (i.e. in the Evaluator and Summarizing functions of a single Mental Module). Two analyzer modules have already been implemented: a weather simulator, and a Visibility / Visual Quality Model.

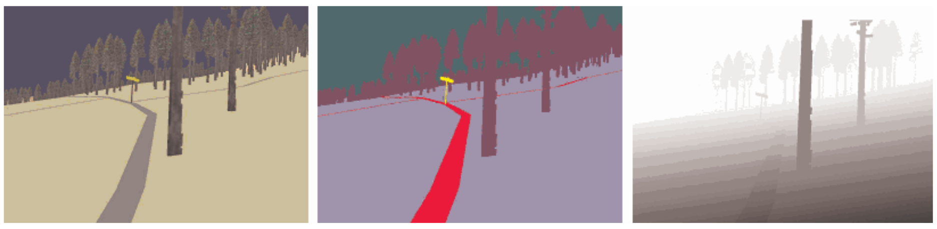

Figure 6: The visibility analyzer module calculates what can be seen by each agent in the simulation. Using positional data generated by the physical simulation, the module uses 3D rendering techniques to render false colour images and depth maps of the agent's field of view. These images are analyzed and information about what is seen is broadcast back to the event stream.

Example Analyzer Module: Visibility / Visual Quality Module

The visibility module is an example of a secondary analyzer module: it listens to the main event stream, and based on the current agent locations, it calculates what is visible to that particular agent. It then broadcasts this visibility to the main event stream, where it can be "heard" and interpreted by one of the mental modules. The visibility module uses a 3D representation of the landscape being modelled to calculate what can be seen from any location in the model [7]. Depending on the needs of the questions asked of the simulation, the visibility calculations information can be pre-computed or done in real-time as the simulation in running (useful if the visibility of other agents is important.) As interpreting the results of the visibility calculations are rather computationally intensive, a further Analyzer Module was developed that interprets what an agent sees and returns an aggregated visual quality score. This means that the mental modules need not be further complicated by this interpretation. Like any other analyzer module, the Visual Quality Module can be inserted almost transparently into the event stream.

4 Communication and Coordination

As the individual modules are implemented in most cases as separate executables, communication and co-ordination between the modules is a crucial part of the overall system design. The modules communicate with each other via TCP network messages, which are formatted as XML. There are two major message types in the system:

- **Control messages:** these messages are used for communication between the control modules and the mental modules or physical simulations. They consist of XML "requests" from the control modules and "responses" from the other modules.
- **Event messages:** these messages are used to broadcast information about agents' current location and state to the entire simulation system. The events are sent by the physical simulation and analyzer to the Event Broker module, which re-broadcasts them to all interested modules. The events indicate when an agent has started a specific activity (such as hiking), reached a specific location (such as a path intersection), encountered congestion, etc.

A key issue is timing: in order to keep all modules synchronized during a model run, messages are sent to identify which modules are ready to receive additional input. We use a variation of the Time Warp algorithm [8], whereby modules inform the control module at which temporal resolution they are operating (some modules, such as the physical simulation might need to react every 10 seconds "real-time", whereas others, such as the weather simulator, might only need to re-compute every 15 minutes) and if they are ready for the simulation to proceed.

One of the advantages of using XML messages over TCP is that it is relatively trivial to distribute the various modules across multiple computing nodes. While this requires some configuration changes in the control modules, and perhaps in the modules being distributed, those modules receiving messages generally do not need to be modified to accommodate this. The current implementation has the visibility analyzer distributed transparently across multiple hosts, as it requires a fair amount of computing resources.

Another advantage of this approach is that the modular nature allows one to test different implementation approaches for different modules without needing to rewrite the entire system.

4.1 Within-Simulation Replanning

One example of using the modular structure to test different approaches was the implementation of replanning during the simulation run. In the typical implementation, during a model run the mental modules only observe the event stream. They use this data to make decisions for the next model run. However, with a simple modification to the mental modules, the system was modified to accommodate changing the agents' plans in the middle of a simulation run. As the mental modules realized that an agent's plan was not appropriate for the day's weather (modelled by the weather simulator), it sends a revised plan to the control module, which forwards it to the pedestrian simulation.

5 Outlook

While at first glance the system might seem rather over-complicated, the modular structure now in place allows for it to be easily extended and tweaked without extensive rewriting of software code. A particular strength of the framework is that modelling the agents' physical interaction is completely separate from modelling the agents' mental processes, which is an area which requires extensive research before the simulation of pedestrian behaviour will be entirely plausible.

Although the current implementation is still some steps away from a real-world applicability in the tourism industry, our prototype nevertheless demonstrates that all these features can indeed be implemented into a computational system. Future work will include to make the system more robust, and to include better behavioural models.

6 Acknowledgements

Parts of this work were funded by the Swiss National Research Foundation's research Programme "NFP 48: Habitats and Landscapes of the Alps."

References

1. Ferber, J., *Mutli-agent Systems. An Introduction to distributed artificial intelligence*. 1999: Addison-Wesley.

2. Gloor, C., *Distributed Intelligence in Real World Mobility Simulations*, in *Unpublished Doctoral Thesis, Department of Computer Science*. 2005, ETH Zürich: Zürich.

3. Gloor, C., et al., *A Pedestrian Simulation for Very Large Scale Applications*, in *Multi-Agenten-Systeme in der Geographie*, A. Koch and P. Mandl, Editors. 2003, Institut für Geographie und Regionalforschung der Universität Klagenfurt.

4. Mauron, L., *Pedestrian simulation methods*, in *Unpublished Diploma Thesis, Department of Computer Science*. 2002, ETH Zürich: Zürich.

5. van Wagtendonk, J.W. and J.M. Benedict, *Travel Time Variation on Backcountry Trails.* Journal of Leisure Research, 1980. **12**: p. 99-104.

6. Dijkstra, E.W., *A note on two problems in connexion with graphs.* Numerische Mathematik, 1959. **1**: p. 269–271.

7. Cavens, D., et al. *Integrating Visual Quality Modeling within an Agent-Based Hiking Simulation for the Swiss Alps.* in *The Second International Conference on Monitoring and Management of Visitor Flows in Recreational and Protected Areas*. 2004. Rovaniemi, Finland.

8. Jefferson, D.R., *Virtual Time.* ACM Transactions on Programming Languages and Systems, 1985. **7**(3): p. 404-25.