An agent-based parallel geo-simulation of urban mobility during city-scale evacuation

Kashif Zia, Katayoun Farrahi, Andreas Riener and Alois Ferscha

SIMULATION published online 13 May 2013
DOI: 10.1177/0037549713485468

Simulation

Simulation: Transactions of the Society for Modeling and Simulation International
0(0) 1–31

DOI: 10.1177/0037549713485468
sim.sagepub.com
SAGE SCS

An agent-based parallel geo-simulation of urban mobility during city-scale evacuation

Kashif Zia, Katayoun Farrahi, Andreas Riener and Alois Ferscha

Abstract

The simulation of urban mobility is a modeling challenge due to the complexity and scale. The complexity in modeling a social agent is due to three reasons: (i) the agent is behaviorally complex itself due to several interrelated/overlapping modeling aspects; (ii) the setting in which a social agent operates usually demands a multi-resolution approach; and (iii) the consideration of real spatial and population data is the underpinning that has to be realized. In this paper, we propose an agent-based parallel geo-simulation framework of urban mobility based on necessary modeling aspects. The aspect-oriented modeling paradigm relates the models vertically as well as horizontally and highlights the situations requiring multi-resolution interfacing. The framework takes into consideration the importance of technological footprints embedded with social behavior along with essential space and mobility features keeping focus on the importance of the city-scale scenario. We have used a real, high-quality raster map of a medium-sized city in central Europe converting it into a cellular automata (CA). The fine-grained CA readily supports pedestrian mobility and can easily be extended to support other mobility modes. The urban mobility simulation is performed on a real parallel and distributed hardware platform using a CA compatible software platform. Considering city-wide mobility in an emergency scenario, an analysis of the simulation efficiency and agent behavioral response is presented.

Keywords

City-scale evacuation, parallel and distributed simulation, urban mobility, large-scale agent-based modeling, geo-simulation, computational social science.

1. Introduction

Agent-based modeling (ABM) and simulation is increasingly being used in many disciplines where large-scale modeling at the entity level is desirable; for example, molecules in physics,[1] cells in biological sciences,[2] livestock in ecology,[3] natural objects in the environment,[4] households in urban planning,[5] and vehicles in traffic engineering.[6] Beyond the scientific and economic domains, modeling of humans in the social sciences, termed computational social science (CSS),[7] has also been a focus of ABM.[8] CSS is a branch of sociology that uses computationally intensive methods to analyze and model social phenomena. Using computer simulations, artificial intelligence, complex statistical methods, and analytic approaches such as social network analysis, CSS develops and tests theories of complex social processes through bottom-up modeling of social interactions.[9]

Modeling a CSS application is a challenging task due to following reasons:

- The social dimension of an individual cannot be explained without understanding many interwoven and overlapping modalities rooted in several knowledge disciplines such as human psychology, social behavior, cognitive science, and economics. Hence, modeling such a system requires significant social science knowledge, an often unfamiliar domain for computer scientists.
- Humans have enormous heterogeneity, ultimately culminating into an extreme where each human is different from another. This results in two difficulties: (i) it is nearly impossible to model social rules having a majority consensus of the theorists; and

Pervasive Computing Institute, Johannes Kepler University, Linz, Austria

Corresponding author:
Kashif Zia, Pervasive Computing Institute, Johannes Kepler University, Altenberger Straße 69, A-4040 Linz, Austria.
Email: kashif@pervasive.jku.at

(ii) it is absolutely impossible to introduce as much heterogeneity in behavior as we observe in daily life.

- The complexity of understanding social behavior due to the above two reasons often results in entirely unpredictable outcomes. Hence, modeling should be flexible enough to accommodate a phenomena with randomness and unpredictability.

In spite of the challenges faced by CSS, ABM as a platform to investigate and predict a social behavior has realistic potential. This is because ABM allows modeling at an individual level, allowing as much heterogeneity as required. The agents adapt their behavior based on local interactions, possibly culminating into unforeseeable emergence of properties at the population level, such as self-organization,[10] adaptation,[11] and social emergence.[12]

Traditionally, sociologists have modeled social processes as interactions among variables. With the increased popularity of ABM, the simulation is now focusing on actors rather than factors.[13] The notion of an actor describes a paradigm shift of modeling social life focusing on adaptive agents rather than the aggregation of variables. Agents are behaviorally adaptive in a sense that they influence others in response to the influence they receive. This is contrary to the sociologists' earlier understanding of social life as a hierarchical system of norms and influences. In fact, social behavior is mostly complex and usually not governed by a global system or a macroscopic model.[14] ABM provides a mechanism to model such a system at the microscopic level,[13] solely based on local interactions among adaptive agents. This can serve two purposes: (i) it prevents the need for the complex task of precisely modeling a nonlinear global system; (ii) it provides an opportunity to model in a bottom-up manner which is closer to reality. Moreover, a more precise behavior model of an individual can be obtained, whereas a model of collective behavior is less precise due to the unknown individual-level behavior producing it.

The following strengths of ABMs make them suitable for CSS:

- According to Gilbert,[14] the main focus of agent-based models is theoretical development and explanation rather than precise prediction. To do so, the agents' behavior must be abstracted to explore possible explanations that can describe observed phenomena. Thus, one should not be concerned about realistic assumptions. In fact, if the goal is to understand the fundamentals of a process, ''then simplicity of the assumptions is important and realistic representation of all the details of a particular setting is not''.[15] If agents are more behaviorally sound, a situation often occurs where we are not able to explain the results of a model thus undermining the usefulness of the ABM.
- According to Macy,[13] ABMs are most suitable to systems that do not require central coordination imposing an order in a top-down manner. The most suitable systems are those that can be described by bottom-up, simple, and predictable local interactions generating ''familiar but highly intricate and global patterns, such as the diffusion of information, emergence of norms, coordination of conventions, or participating in collective action''. It is possible that unexpected patterns emerge and then vanish quickly, such as market crashes and panic in a crowd. The most suitable systems for ABMs can be categorized as exhibiting emergence of social order and presenting social self-organization.

Sociologists have now started to realize the importance of ABM for exploring the effects of individuals on the collective.[15] However, there are three issues worth noting.

1. First, the exposure to new possibilities and enthusiasm of exploring more interesting emerging properties of such a system, have often resulted in neglect on the models of agents themselves. That is why in many CSS studies, the microscopic models used are simple and do not correspond to reality.
2. Second, the restriction of strictly following the bottom-up order of the influence is not justified in many cases. For many multi-resolution scenarios,[16] simulation requires models of micro as well as macro scale. This is particularly true for scenarios in which technology is considered to be part of or related to social behavior.
3. Last, the historical standing of ABM to understand process fundamentals has resulted in scenario simplification, particularly in terms of agent populations and environmental modeling. However, with availability of detailed spatial data (e.g. AutoCAD designs, Geographic Information Systems (GIS), satellite imagery), it is now possible to simulate a realistic space and agent population. Advancements in parallel and distributed multi-agent simulation have contributed to this.

In this paper, we focus on the three considerations stated above. Considering a population of citizens evacuating the city in case of an emergency, the (social) behavioral ''ingredients'' influencing such an urban mobility scenario are condensed into *aspects*. The modeling of several inter-related aspects of human behavior (both individual and collective) is defined as aspect-oriented modeling (AoM). The AoM paradigm helps to categorize the interesting dimensions which can/should be modeled in the given scenario. The AoM approach also provides a

structured way of distinguishing between microscopic and macroscopic resolution of an aspect and a mechanism to resolve this for a hybrid modeling situation. The AoM paradigm take into consideration the importance of technological influence on modern living, justifying a case of agents' interactions at different scales. Simulation is performed at a real physical and demographic scale after converting a high-resolution city map into grids of cells. The CA describing the space lets the mobile agents on top be maneuvered. It also provides a natural way to link an agent with the real space. However, a city-scale simulation of this scale in terms of space and number of agents cannot be handled without explicitly employing a parallel and distributed simulation (PDS) hardware and software platform.

In this paper, we have taken up a system which can be categorized as a socio-technical system (STS) for urban mobility with geo(graphic)-simulation capabilities, which we refer to as geo-socio-technical-urban-mobility simulation. The simulation framework (Section 4) supports distribution of space for parallel execution and handles a large population of agents. We consider an city-scale emergency scenario for applying the framework. The urban evacuation simulation is performed on a shared-memory multiprocess hardware platform using a fine-grained (1.25 m^2) grid-based space incorporated from a real map. The software platform is a full multi-agent system compatible with grid based space. The AoM paradigm (section 3) provides design principles for the agent model. Section 6 gives a detailed account of the agent model. The hardware and software platforms used to perform the simulation are presented in Section 7. Section 8 is devoted to the simulated scenario and a discussion on the simulation outcome, focusing both on agents' behavior and performance of the PDS.

2. Related work

2.1. Motivation

When comparing with the simplistic scenario of evacuating from a building,[11,17] simulating a geo-socio-technical-urban-mobility system has many additional challenges. There are many types (each considered as a different **population**) of interacting entities, where each entity can have its own character and behavior. The social and technological interaction may have different modalities as well, resulting in varying **information dispersion** based on extent and periodicity of interaction. Essentially, each such entity/agent has its **individuality** which would be reflected in **social** relations (both technological and humanistic). In addition, inclusion of geographical information as an underpinning **space** to which each such agent is attached to adds a new dimension to the level of complexity. In addition to space being a direct influence to the **mobility** of agents, the fact that space may **change** its state has consequences on every possible behavior including mobility. The AoM paradigm presented in Section 3 not only captures the social-technical dimensions of a crowd-mobility-based phenomena in its entirety, but also focuses on a complex embodiment of space at which this phenomena may occur, i.e. the scale of a city. The modeling framework (see Section 4) describes a formalism where model variety would be integrated into an agent-based PDS.

Different approaches are used to model different aspects in a population model (e.g. spared of panic) may focus on the whole system by collecting observable behavior at the crowd level. Crowd models are often termed macroscopic crowd models. However, the same behavior can be modeled on a microscopic level, where individuals are influenced only by local impact and as a result, a panic situation emerges. At the macroscopic level, the microscopic level interactions are overlooked entirely. Instead crowd dynamics are treated as a fluid, with characteristics such as flow rate, concentration and average speed are all being functions of 2D space and time. There are advantages and disadvantages to both methods. For example, modeling a panicked population using a microscopic strategy may involve change in individual behavior due to social impact (e.g. an individual transiting his phase from altruistic to a selfish state) which then needs to be locally ''spread'' in an epidemic fashion. Hence, calculating such changes, and its impact on surroundings is computationally extensive. If we are only interested in dynamics of panic in a system, a macroscopic methodology focusing on global states can be much more efficient. The macroscopic models may fall short in those cases where the system behavior is inherently stochastic, can be influenced by the occurrence of single events, is sensitive to the action of a few individuals, or whenever the average behavior is not of much interest. Moreover, the modeling aspects are often vertically and/or horizontally dependent. To resolve these dependencies and overlapping, it is almost impossible to combine two or more self-contained macroscopic models. If all of the models are microscopic, its easier to combine or interface these. The issue of microscopic modeling lack of efficiency is becoming less of an issue due to rapid developments of high-performance computing facilities.

2.2. Multi-aspect modeling

Although the term STS has originated from organizational structure and work force management,[18] it is now being used for many other domains (including urban activities[19]) where there is ''effective blending of both the technical and social systems''.[20] Fox[20] emphasizes the need to deal with ''social'' and ''technical'' aspects inter-dependently because ''arrangements that are optimal for one may not be optimal for the other and trade-offs are often required''. However, like many other STS studies, the focus of the

study was only on organizational work flow. For a geo-socio-technical-urban-mobility simulation, the aspects we consider are more extensive and specific.

Pan et al.[11,17] have argued that individual and social behavior of most crowd agent models in egress is unjustifiably oversimplified, inconsistent, and incorrect. While presenting a framework of human decision making and social interaction for emergency egress analysis, they advocate the necessity of studying and modeling human and social behavior in more depth. However, in addition to ''individual'', ''social'', and ''population'' consideration, we also focus on, e.g. ''information dispersion'' and ''space/mobility'' aspects.

2.3. Multi-resolution modeling

As highlighted in Section 1, the restriction of strictly following the bottom-up order of the influence is not justified in many realistic settings. Instead there is a trend to have both methodologies in one simulation. Xiong et al.[16] have presented a simulation framework where microscopic and macroscopic models co-exist. However, the simulator switches between the resolutions based on the situation and the co-existence is not consistent. Xiong et al.[21] have used the concepts of aggregation and disaggregation to pass on the state from microscopic to the macroscopic level and from macroscopic to the microscopic level, respectively. However, the concept cannot be applied to a large population of agents in a distributed simulation. Chen et al.[22] have extended the concept of aggregation/disaggregation towards a crowd over a hierarchal grid architecture. This work has two limitations: (i) the behavioral models are simple (e.g. pedestrians and vehicles mobility); and (ii) internet based computation distribution technologies such as HLA_Grid_Repast framework[23] and grid-aware time wrap kernel[24] are not suitable for fine-grained models due to limited bandwidth. The framework presented in this paper eliminates both limitations.

2.4. Urban simulation

A city-scale, fine-grained, agent-based evacuation simulation is an achievement in its own right. To the best of the authors' knowledge, a CA-based city-scale evacuation simulation with millions of agents has not been attempted before. In contrast, many large crowd simulation studies focus only on efficient simulation of only agents ignoring the spatial dimension of the scenario. Either the agents have no spatial significance at all[25] or these are considered as networks of agents irrespective of space.[26]

3. AoM paradigm

In Section 2.1, the aspects important for a geo-socio-technical-urban-mobility system were introduced, namely, individual, social, population, information dispersion, space, mobility, and dynamics of evolution. Here we discuss these categories under three headings adhering to how an ABM is formalized, namely, ''heterogeneity'', ''interaction and adaptation'', and ''space and mobility''. The discussion given below covers the most important dimensions of the system required to simulate the city-scale agent-based model for evacuation scenarios.

3.1. Heterogeneity

3.1.1. Agent behavior. Heterogeneity is related with the **individual** aspect. This ranges from the *physical* dimensions of a human being (e.g. body, age, gender, agility etc.) to *personal* attributes which can be ''biological'' and ''cognitive''. The biological aspects include human perceptions (sense of vision and hearing), whereas, the cognitive aspects include emotions (e.g. models addressing trust and belief of the agents).

The physical characteristics of individual humans have significant effect on individual and crowd behavior, thus mobility. For example, elderly individuals are generally less agile than younger individuals.[27] However, the most important are personal biological characteristic of an agent. The sensory perception is the ability of an individual to sense the environment. These may include the sense of vision, hearing or touch. In a technologically equipped urban environment, technology plays a vital role in decision-making.[28–31] That is why we also consider a shared/public (e.g. an exit sign or a public display) or a private/personal (e.g. a smart phone) device as an agent. The interaction capabilities of such a device is considered a conceptual equivalent of human sensory perceptions.

The personal cognitive characteristics are as important as the biological ones. Some examples of individual cognitive characteristics are:

Instinct: Instinct refers to internal patterns of behavior in response to specific stimuli. Executing an instinct does not require a conscious thought process. Examples of human instinct are fear (hope), survival, smiling. When there is a need to make decisions under high stress, following one's instincts is the most intuitive way.[32]

Experience/knowledge: An individual often relies heavily on personal experiences in decision making. Many life events are highly repetitive. An individual usually develops a set of relatively standard routines over time and then applies them to similar situations in the future.

Trust and belief: Belief is individual's judgment against an option. Trust is an attitude of an agent towards an information source that determines the extent to which information received by the agent from the source influences agent's beliefs. The trust to a source builds up over time based on the agent's experience with the source. In particular, when the agent has a positive (negative)

experience with the source, the agent's trust in the source increases (decreases). Trust and belief have an umbrella effect on all of the behavioral patterns an individual may have.

3.1.2. Collections and containers. The fact that ABM provides the capability to create one-to-one correspondence of real-world actors with virtual agents, defining an agent itself describes an individual. However, the granularity of atomicity of an individual is an issue to consider. For example, taking space description also as an agent, should we represent a building as an agent or a square cell constituting a building as an agent? We opted for the second option. The minimally resolvable description, if opted as an agent, would not hinder representing a collection of these as a collective structure. For example, in many simulations that we have conducted, we represented functionally collective structures (e.g. an exit) as a collection of space agents of type *exit* having assigned the same *exit ID*.

Many simulation environments define a **population** as a special agent group which falls into a class. A population describes a group of individual related spatially, temporally, socially or behaviorally. For example, in a morning rush hour, there would be a high population of school going youngsters in a train, or during a cultural event, a high population of tourists is expected in a street. On the behavioral side, a population has a common goal and movement pattern which distinguishes it from other populations. This means that members of such a population are likely to follow the same social models and may have similar individual attributes. We have used concept of *breeds* to represent behaviorally similar populations.

The fact that a notion of ''containment'' may exist between two breeds of agents, the solution is implemented using pointers. For example, dozens of people (contained agents) traveling in a bus (container agent) are contained within the container through connections. In this case, the contained agents do not lose their existence or identity. However, the certain attributes of the contained agents would be overtaken by the container agent. For example, in the above example, the travelers would not have any control over their speed and direction which would be the speed and the direction of the bus.

3.1.3. Behavioral abstraction. Owing to the existence of agents as entities at minimal possible atomicity, we face no difficulty in managing heterogeneity of agents and, hence, their individualism. However, the interrelation of model and complexity of the features described above could be problematic to comprehend if more rationality is required. As discussed in Section 1, the main idea of ABM is to explain the phenomena, not to predict precisely. To this end, the agents' behavior must be abstracted just focusing on the core features.

3.2. Interactions and adaptation

3.2.1. Social structures. Interactions between individuals constitute **social** behavior. In a STS, the interactions between agents can be humanistic and technological. The social aspects are divided into two broad categories: (i) models which address the influence of social attributes of an individual which affect the interaction among individuals; and (ii) models which describe an information flow or mechanism as a result of these socially influenced interactions.

By viewing a crowd or a group within a crowd as an emergent structure, we can identify many significant factors that may contribute to such an emergence. A group within a crowd is related with localized interactions where people may behave synchronously or may differentiate themselves from others. In either case, such a social influence defines behavior of a group of related people either moving together, or forming a queue,[33] or transforming into a herd or a jam. It also describes the movement pattern and decision making in case of agents following an agent with more knowledge (e.g. leader–followers behavior). Mostly emergence of such a group within a crowd cannot be programmed and it evolves due to localized agent interactions where different types of agents have different social rules, norms, and values.

3.2.2. Information dispersion. How an agent is ''connected'' with others influences the **information dispersion** and collection mechanisms.[34,35] We have categorized information spread into two broad categories: (i) explicit, where information is transferred explicitly, e.g. speaking, announcing, or communicating; and (ii) implicit, where information is implicitly perceived, e.g. seeing or being influenced. Due to exceedingly high proliferation of digital devices which can interact with each other without human intervention, the interaction capabilities (range and mode) of these devices is an aspect to explore.

The connectivity can be spatial and/or social. With the proliferation of personal technologies in the society, an agent can interact with others overwriting the spatial constraints. However, technologies are also spatially influenced and have limitations of their own. We use the terms ''extent'' and ''periodicity'' of interactions to harness variations in connectivity. The variations can be in spatial extent, i.e. global connectivity, local network-based connectivity, group connectivity and proximity based connectivity. The variations in extent of social connectivity can be locality, relational, or social networking. The extent also focus on probabilistic connectivity in spatial as well as social domain. This primarily affects the information dispersion mechanism as not all agents are humanistically/

technologically feasible to interact with. The interaction is also concerned with how often it happens or should happen, addressed by the concept of periodicity.

Different possibilities of information spread related with real-life situation can be explored. For example, one mode of information spread can be observation or vocal dispersion. In this case the senses are purely humanistic and individual differences can result in different behavior. On the technological side, the possibilities of agents having various modes of interaction capabilities would play an important role in deciding about dispersion level and freshness of transmitted information. Social networking is yet another paradigm highlighting the importance of connectivity.

3.2.3. Adaptation. Interaction results in agent behavior change. The obvious way of realizing the adaptation is to model the change in behavior within agent's description where an agent would be influenced by the information it receives from different channels: proximity, group, or global. For example, an agent can opt to move towards the least dense region in its surrounding based on its visual range. Or an agent can delete a destination from its list knowing that the destination is no longer accessible. The first example is related to reacting to the situation at the local level and does not require information dispersion. The second example is related to reacting to the **change dynamics** happening at the societal level and requires information dispersion.

3.3. Space and mobility

3.3.1. Related work. The representation of the physical space plays a central role in the simulation. In the domain of **crowd simulation**, there are many ways to deal with the space and consequently the mobility. The CA model of Schadschneider,[36] for example, uses a discrete structure of space. Helbing et al.,[37] in their social force mobility model, use the equation of motion to describe the change of location of the pedestrians, assuming a continuous treatment of space, similar to the multi-layer utility maximization approach proposed by Hoogendoorn et al.[38] In all of these models, the pedestrian is seen as a point or a particle in a 2D environment. In the agent-based approaches, the agent moves through a virtual environment where the movements can be discrete or continuous.[39] A completely different approach is proposed by Borgers and Timmermans.[40] They use a network representation, where each node corresponds to a city-center entry point or a departure point and each link denotes a different shopping street. In this case, the network topology represents the walkable space and the movement occurs along the links between two consecutive nodes.

One of the earliest agent-based models used CA as a representation of space.[41] In fact, cells of CA were actually agents with different states. There are other recent examples in which a cell acts not only as a space but also as an agent.[4] For more complex ABM, CA was used as an underlying space with mobile agent on top of it.[42–44] Many software programs also adopted this as one of the design principles realizing its usability and consistency with most of the real-life systems, for example, NetLogo,[45] Repast,[46] and Mason.[47] To implement the local mobility decision, these approaches adopted interaction topologies mostly based on von Neuman and Moore neighborhoods.[48] However, other interaction topologies are also important and continuously being included into simulation systems. These are Euclidean 2D/2D space, network topology, GIS, or Aspatial.[49]

The most promising progress which probably provided the impetus to the urban simulation in the last decade is open GIS data, which is increasingly becoming ubiquitous and easy to use. Several examples of integration of the GIS data in the environment modeling at a city or broader scale are already there. These simulations are categorized as **geo-simulations** due to use of GIS data for space modeling. For example GIS data has been used to simulate the land-use,[50] urban growth,[5] fire propagation,[4] and traffic flow.[6] However, the static nature of GIS data does not allow an easy integration with dynamic processes.[4] CA give us a simplistic way of performing dynamic integrated GIS modeling. CA is inherently spatial and can readily be used with available GIS sources.[51] Raster-based GIS is a simplistic way to get a CA-based space when modeling is spatial and discrete-time. It does not require complex vector manipulators and conversions.

For mobility, raster-based CA provides necessary flexibility and behavioral sophistication. Most of the (raster) GIS-based CA models concerning with land and region oriented simulations use cells as agents where change in the states of the cells brings in the dynamics on top of the static map. However, there can be layers of agents on top of cell space. For example, in traffic flow simulation,[6] the vehicles reside on top of static space agents (cells) and are mobile. This increases the complexity in modeling and execution time of the simulation.

In the case of city-scale crowd simulation, there can be multiple layers of agent groups arranged in a hierarchical order. For example, a road segment (collection of cells) can be assigned to a moving agent representing a vehicle which may further contain many mobile human agents (travelers). There are other mobility modalities that can also be imagined, either independently or in a mixed mode. There are research efforts which consider combining two or more of these modalities.[52] However, most of them are conceptual studies. To the best of the authors' knowledge, the complex task of actually performing a mixed-mode simulation on a large and real scale has never

been achieved. Our simulation framework is capable of achieving it due to its fine-grained spatial distribution after applying a suitable mechanism of augmenting the space with necessary semantics.[53] This successfully extends the current spatial abstraction in which the agents can either be static or mobile, where one of the mobile agents may occupy a static agent partially or fully, even not exclusively.

3.3.2. Space. Agents usually have simple behavior which is influenced by their interaction with other agents. The guiding principles of agents' behavior is its autonomy of making decisions. However, an agent's decision-making depends on the agents it is (has been or will be) interacting with, which many a times has spatial consideration. Particularly in urban domain, the position of an agent is very important and it is related to a certain geography. We consider a cell as a unit of space, which is treated as a static agent. The mobile agents occupy the **space** as an overlay, thus adhering to the approach suggested by Epstein,[42] Dijkstra,[43] and Kirchner.[44] Our CA-based interaction topology is based on Moore neighborhood,[48] but with possible variation of radial scale.

Technically for a space, obstacles and attractors must also be modeled. Fixed obstacles are represented by regions that no pedestrian can access. Moving obstacles are groups of pedestrians occupying space which is consequently not available any more. Attractors are useful areas with particular meanings for the individuals. Finally, origin and destination areas, where pedestrians originate and end up in a system, must be defined.

The concept of space has two meanings: semantic and physical. In the semantic case, we take space as a room, or a specific corridor connecting two rooms. The conceptual definition of space in this way guarantees a more convenient behavioral analysis granularity focusing on a conceptual basis of analysis rather than unnecessary physical (coordinates, etc.) details of a space. In the physical case, we describe the space in physical domain which is necessary to generate CA. While modeling a space based on GIS raster map, we took the physical concept of space for implementation.

3.3.3. Mobility. The **mobility** models address how individuals move within a locality and make decisions about route choice. The combination of CA with agents as an overlay makes it easy to manage the mobility.

The locomotion (small-scale movement) has two basic features.[27]

Speed: Speed is the most important feature of locomotion. In an empty space, the destination and the path are almost sufficient to reproduce the trajectory of a given pedestrian. When the environment is crowded and contains obstacles, the direction and the speed of the pedestrian may be significantly affected.[54]

Collision avoidance: The collision avoidance patterns stem automatically from a combination of the velocity vector of the other pedestrians and the density parameter. In microscopic models, an individual tries to keep a minimum distance from the others. In the social force model, this pattern is described by repulsive social forces.

The route choice is choosing a destination at a coarser level of interaction.[40] An example of route is the Google map description of route plan given a source and destination. In CA-based models, the route is chosen based on limited information in the cells, unless we augment it with macroscopic information about the decision parameters such as densities at possible destinations.

4. Agent-based parallel simulation: The framework components

The framework for agent-based parallel simulation consists of three main components, illustrated as three large blocks in Figure 1: the image processing unit, the agent-based PDS unit, and the output data processing and visualization unit.

4.1. Image processing

The massive city map, initially a raster image, is incorporated into the model by first reducing it to the relevant areas (Figure 3). The focused area was then segmented into partitions depending on the processing resources and inter-process synchronization overhead, i.e. 200 equal sized squares regions of 20 rows and 10 columns (Figure 4).

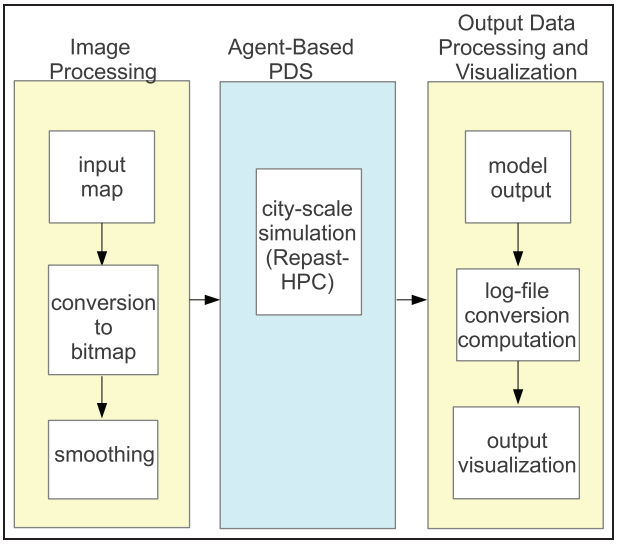

Figure 1. Overview of agent-based parallel simulation framework and the major sub-components.

These regions were first converted into a binary grid. The streets are selected to be walkable areas (agents can move here), and all other areas are considered to be non-walkable (agents cannot move here). Second, a smoothing algorithm is run, first horizontally, then vertically, to counter inconsistencies in raster. The smoothing function is a low-pass filter, with filter coefficients equal to the reciprocal of the span. The 200 files are saved to ASCII format to be used as GIS-based space.

4.2. Agent-based PDS

The simulation space is divided between two conceptual memory distributions: the local memory and the shared memory. The **local memory** is the memory place of each of the processor p_x (or process) out of total P. The **shared memory** is the memory place shared by all of the processes. The physical sharing of data is mostly managed by a high-performance software platform designed for shared memory multiprocessor (SMP) architectures, i.e. Repast for High-Performance Computing (Repast HPC).[55] The shared memory is used for synchronization of processes, simulation configuration files, and disk space for files IO (e.g. simulation log).

The simulation setup at each of the processes can be guided by simulation and model configuration files. In Repast HPC, any process, synchronization, and model-related parameters that need announcements at shared data level can be written in these configuration files. The most important are the boundaries of the space in terms of x- and y-coordinates (in CA mode), the number of processors and their vertical/horizontal distribution, the simulation time in terms of number of iterations, and synchronization information. An ID is automatically assigned to each of the process based on vertical/horizontal distribution of the processes which starts at 0 and ends with $N - 1$. It is clear that on the left-hand side of Figure 2, there are 200 processes with IDs from 0 to 199.

A process model can be divided into three components: setup, model, and synchronization.

4.2.1. Setup. The following steps are part of simulation setup:

- Update world with GIS: Initially the Repast HPC creates an inert grid of cells with default setting. This space should be updated with the GIS information provided in ASCII files.
- Set up point of attractions: Another GIS information which is not currently part of the ASCII files, i.e. the location of point of attractions or possible destinations on the map, provided through model properties files, should also be incorporated into the space.

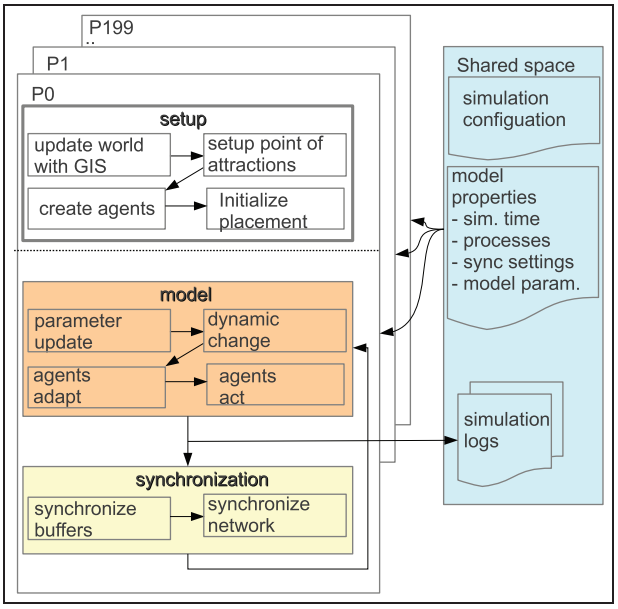

Figure 2. Agent-based city-scale PDS using Repast HPC.

- Create agents: In this step the space is populated with moving agents; i.e. agents of desired types are created mentioned in properties files.
- Initialize placement: Moving agents are placed in the space based on required density and mutual distance.

The set-up module is executed once before the start of the simulation.

4.2.2. Model. The following procedures are part of out agent model:

- Parameter update: The global variables (e.g. processes densities) are shared and updated.
- Dynamic change: In this logical step, the space dynamic is updated. This includes the floor field spread and spreading of information about change in dynamics of the space.
- Agents adapt: Each of the moving agent updates what it knows about the agents in the proximity as well as update in information at the global level parameters. This may result in adaptation of agent's behavior.
- Agents act: Finally, based on possibly adapted behavior, an agent performs the required action.

4.2.3. Synchronization. The basic purpose of these procedures is to synchronize the buffer and network space. The buffer space is set from model properties files so it is same for all processors. The buffers are managed on all sides of the adjacent processes and can be synchronized at the *i*th

iteration where i can be 1 if required. The network space shares the information between connected nodes. Each pair of such agents is synchronized at ith iteration where i can be 1 if required, if peer agents reside in the adjacent processes.

This interaction range (proximity) along with the periodicity of executing a step (update/change/adapt/act), synchronization and writing log files should be carefully managed keeping in view the trade-off between simulation efficiency and information.

4.3. Output data processing and visualization

The output of our system consists of a series of log files which are processed by a computer script to obtain the necessary format both for evaluation and for visualization. All of the visualizations of the city model are created by NetLogo.[45]

5. Efficiency parameters

Scalability is defined as the performance of a system as the size increases. If we wish to decide whether a simulation model will scale or not, we have to consider the architectural as well as the algorithmic side of the system. The following interrelated factors influence the efficiency of a parallel and distributed agent-based simulation setup:

- Agents' interaction: The extent (the range of interaction) and periodicity (how often the agents interact with each other).
- Agents' behavior: The complexity of agent's models and types of agents present in the system.
- Space distribution across the Processors: How the space can be distributed across the processors; the size and shape of the space.
- PDS essentials: How the synchronization mechanism between the processors is implemented and how often the synchronization takes place.

5.1. Agents' interaction

The extent (the range of interaction) and periodicity (how often the agents interact with each other) are two basic parameters affecting the PDS efficiency. In our previous study,[56] we have focused on simulation efficiency of a large-scale agent population. We designed and simulated a framework of large-scale social agent (an abstract definition of a social entity) simulation with essential social features namely cluster size (interaction range) and connectivity extent where each agent had to execute a hypothetical workload. It was concluded that while using a cluster machine, an acceleration of a factor of up to 727 is possible in one of the realistic variation in cluster size, connectivity and workload, when compared with a single CPU.

5.2. Agents' behavior

The complexity of agent's models and types of agents present in the system establish the requirement of processing time and memory utilization. In our study,[56] we realized that increase in work load decreases the efficiency of PDS. Some models are really computationally intensive in terms of process cycles and memory requirements, e.g. unconditional trust on an leader agent (as used in this paper) is far less computationally intensive then a cognitive trust evaluation model we used earlier.[57–59] The model described by Sharpanskykh[57] also requires recent cognitive state of the neighborhood agents, thus demanding frequent agent interactions.

5.3. Space distribution across the processors

As stated by Li et al.,[60] CA provides a natural mechanism of space distribution across the processors in a regular grid style. However, the decision about size and shape of the space chunks is important. It depends on the available resources on the PDS hardware. It also depends on the average population of mobile agents attached to one of those chunks.

5.4. PDS essentials

The processes can be synchronized at various levels; adjacent buffers, network of connected nodes, and space itself.[61] For CA-based mapping, buffer synchronization is essential. The modeling requirements advocate the sync periodicity and buffer size. The extent of effort required in synchronization is linked with space distribution. The space should not be distributed at a very fine level. It would cost a lot of resources on synchronization alone. In contrast, it should not be very coarse either resulting in utilization of only a few of the available processors thus degrading the efficiency.

The optimization of these factors to guarantee the best efficiency is a challenging task. In this paper (in Section 8), we have reported the important aspects we learned from our experience.

6. Models

6.1. Map distribution

The space is an extract of a raster map of the city of Linz with dimension roughly equal to $10{,}000 \times 15{,}000$ cells. Each of these cells is a square and equal to 1 m a side (1.25 m to be exact). We cropped sides of the map and focused on more interesting central region of the city (see Figure 3). This reduced us to a space equal to $5000 \times 10{,}000$ cells. We divided this space into 200 sectors where a *sector* is the space distribution unit of parallel execution. The 200 sectors were distributed as $m \times n$ along

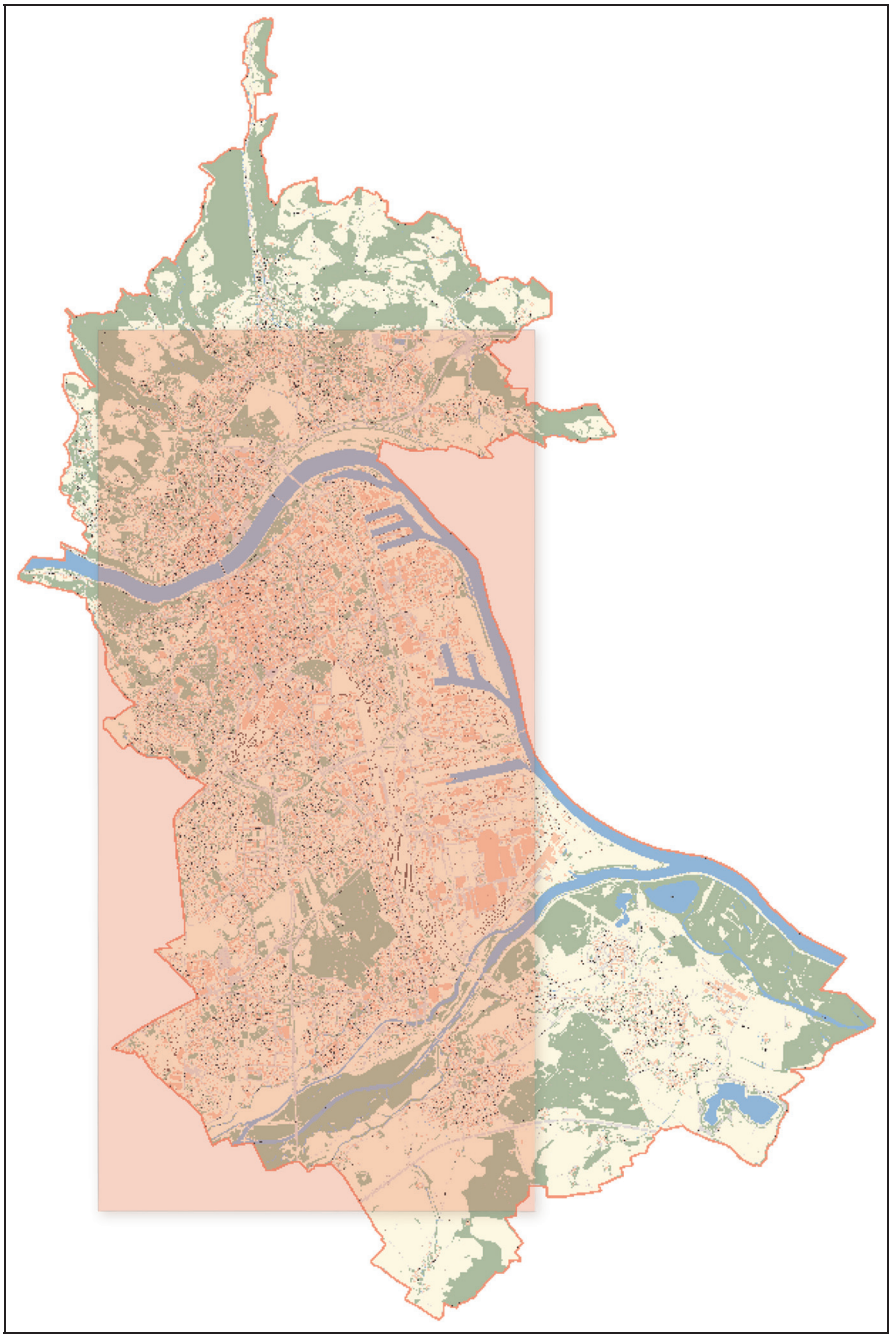

Figure 3. Original raster map of a city. Our agent-based model is simulated for the highlighted area.

horizontal and vertical axis, respectively, where m is equal to 10 and n is equal to 20. Each of these 200 sectors was executed on a single processor as a process p_i, where i is process ID from 0 to 199 (see Figure 1). Each process is responsible for a space equal to $x \times y$ cells (a sector) where x and y are equal to 500 cells each. It is important to note that x and y are equal whereas it is not mandatory. Figure 4 shows a compact visualization of the city space which is simulated, whereas Figure 5 provides a closeup view of one sector (highlighted as sector 33) used for small-scale debugging and test runs before conducting a full-scale simulation.

6.1.1. Repast patch. A patch is a single unit of space. In CA it is equivalent to a cell. Each patch/cell is represented by a integer coordinate anchored at the center of the patch. The most important GIS information each patch has is the ''structure type'', i.e. type of the structure the patch is constituted of. At a city scale, the examples are street, motorway, building, green areas, water, etc. The type thus can

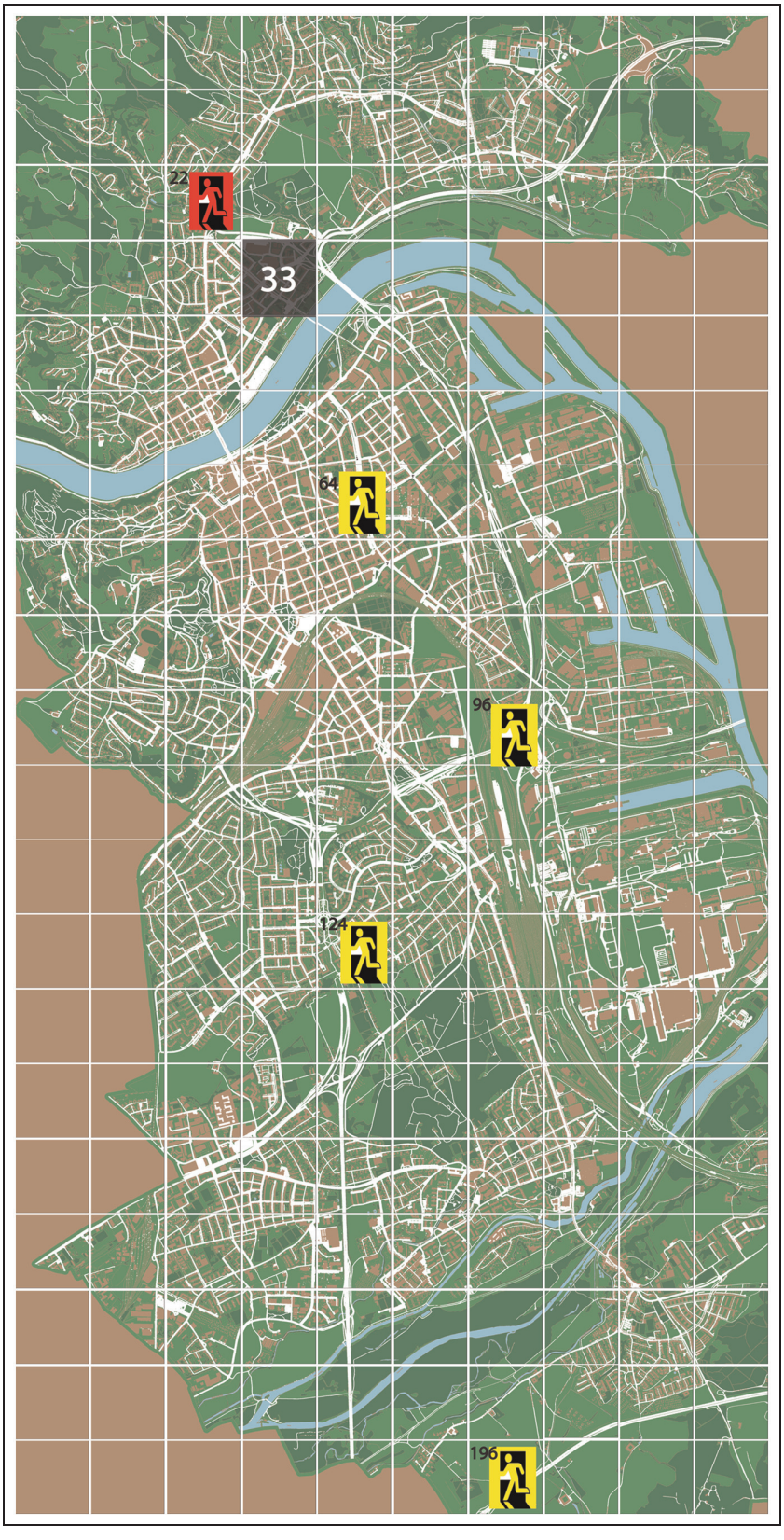

Figure 4. The simulated city map, showing the 200 sectors as squares. Each sector was simulated by one processor. For the city-scale evacuation simulations, the exits (PoAs) are highlighted at sectors 22, 64, 96, 124, and 196. PoA at sector 22 is flooded during the simulation. A closer view of sector 33 is given in Figure 5.

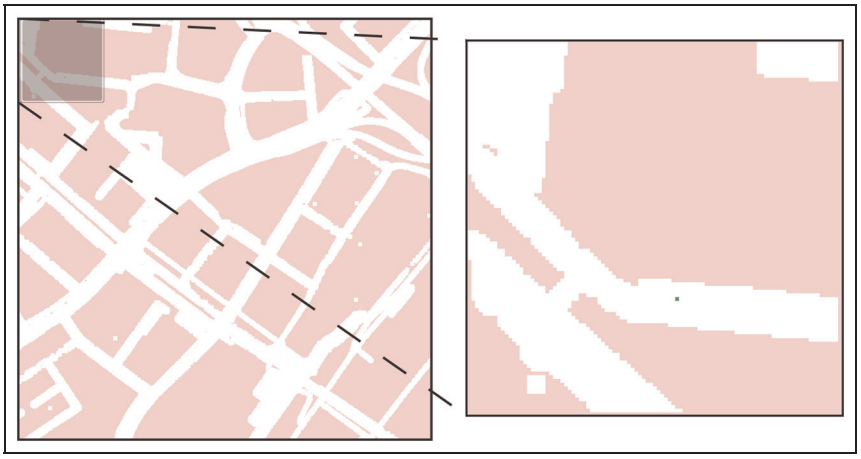

Figure 5. Left: A bitmap view of the test space consisting of 5×5 ($= 25$) processes representing one of the city sectors which is roughly 1/500th of the raster map (Figure 3) and 1/200th of the simulated map (highlighted sector 33 in Figure 4). **Right:** One (top-left) of the 25 processes. All white patches represent streets and are walkable by agents. All others are non-walkable by agents.

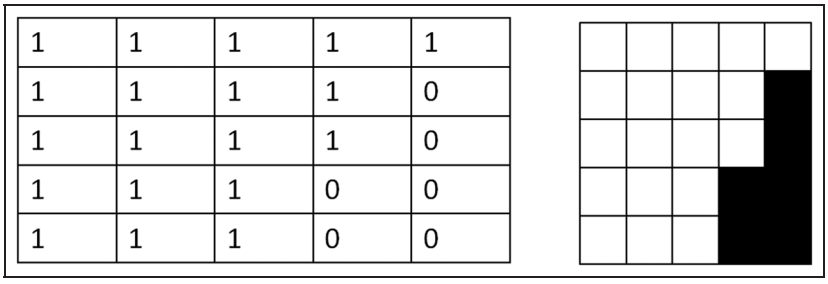

Figure 6. Example grid of patches (right) created from a sample file (left). Patches colored white are walkable whereas patches colored black are non-walkable.

be mapped onto functionality, e.g. along with vehicles, a street is where pedestrians can walk whereas a motorway is where they cannot.

Functionally, we have started with pedestrians only. However, traffic can also be incorporated in a CA-based space.[6] To achieve a mixed mobility, the inter-model behavior should also be modeled.[52] For now, we have simplified the space of having only binary states (as structure); walkable (1) and non-walkable (0). Except for streets, the other walkable areas (e.g. greens, open space etc.) are ignored. Further a walkable patch may be a PoA having a significance for decision making and mobility. Figure 5 shows different structure types at a process level. Figure 6 shows a part of the GIS ASCII file and resulting space.

6.2. Floor field: modeling information dispersion through space

Using floor field concept of CA, the information which relates to the space can be propagated and stored as a field. For example, space should know the navigation information relating to PoA(s). The (movable) agents knowledge and memory is another layer of navigational information which can overlay the space information. In this paper, we do not present results of information sharing between moving agent (the aspect of information dispersion between agents).

The processes and coordinates of the PoAs are provided manually, but these can be purely random. Each patch has three variables for decision making:

- Direction: Directions of PoA(s) from current patch - DOM.
- Distance: Distances of PoA(s) from current patch - HOPC.
- Route: The sequence of processes towards PoA(s) - ROUTE.

A typical structure of the variables at a patch level is shown in Figure 7.

Note that initially these variables assume the role of an obstacle representation (see Figure 8(d)) with no value

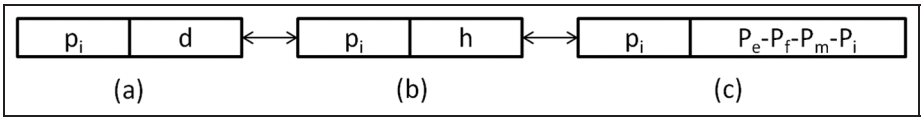

Figure 7. Structure of one patch's floor field: The information in all three cases is indexed by PoA's ID. (a) Direction: *d* ranges from 0 to 355.99 calculated as relative angle between sender and receiver patch. (b) Distance: *h* is number of hops the information traveled before reaching the patch. (c) Route: the sequence of processes propagating the information.

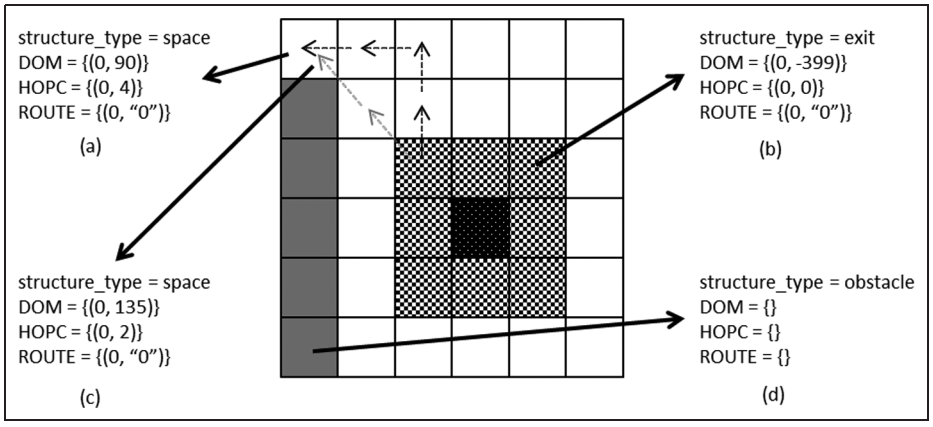

Figure 8. An example PoA, the hatched area, on a process.

assigned. A PoA is the originator of the floor field (see Figure 8(b)). A PoA is represented by assigning impossible values to DOM (−399) and HOPC (0). Each walkable patch **p** other than PoA itself iterates through its Moore's neighborhood to update its knowledge about PoAs. If a neighboring patch (source) has information about a new PoA in its DOM collection or is reporting lesser hop count for an existing PoA, **p** creates/updates the PoA information of its own with reference to the source (compare Figure 8(a) with Figure 8(c)). This involves calculating the relative angle towards the source, incrementing the hop count of source by 1 and concatenating (represented by ".") the process ID (if different from process ID of **p**) in ROUTE process sequence. For a more frequent case (when a PoA does not reside on the same process as that of the patch), the route may have the following formation: {(15, 2.12.13.14.15)(0, 2.1.0)} where the patch is residing on process 2 and there are two possible PoAs at process 15 and 0. An example of dispersion of the floor field is shown in Figure 9, and the complete pseudo-code is given in Appendix A.

The information flow and decision making is integrated, i.e. we do not run the floor field generation prior to actual simulation run. This is possible but does not correspond to real-world situations. During the simulation run, a patch may have no or partial information which is as valuable as full information as agents represent humans who perform some action in any situation. Each patch would ultimately have information about all of the PoAs if there is no disconnection (a series of non-walkable patches disrupting the information flow) after sometimes which can be considered as rate of flow of information and can be controlled.

6.3. Interaction modalities and modeling multi-resolution

6.3.1. Synchronization. Since each sector is executed by a unique process, the information synchronization mechanism must be invoked across the processes wherever it is needed. This applies both to space and mobile agents.

Thinking about propagation of patch fields (both floor field and change dynamics) is simplistic in a single process scenario. But special attention is needed in multiple processors. The same is true for mobile agents when having proximity spread across the processes or having to "migrate" to new process due to mobility. Two modes of synchronizations are supported by Repast HPC; buffer and network. Both modes work with space (patch) and mobile (turtle) agent.

Buffer synchronization: In a grid space (CA), the processes may be synchronized across the boundaries with specification of buffers of a certain size. The buffered would be shared between the processes as a read-only copy (both patch and turtle). In Figure 10(a), the buffer synchronization of size 1 for process 64 is shown. Process 64 synchronizes the patch and turtle data with process 54, 63, 74, and 65. The processes at the diagonals are also synchronized which are not shown. Some patches of

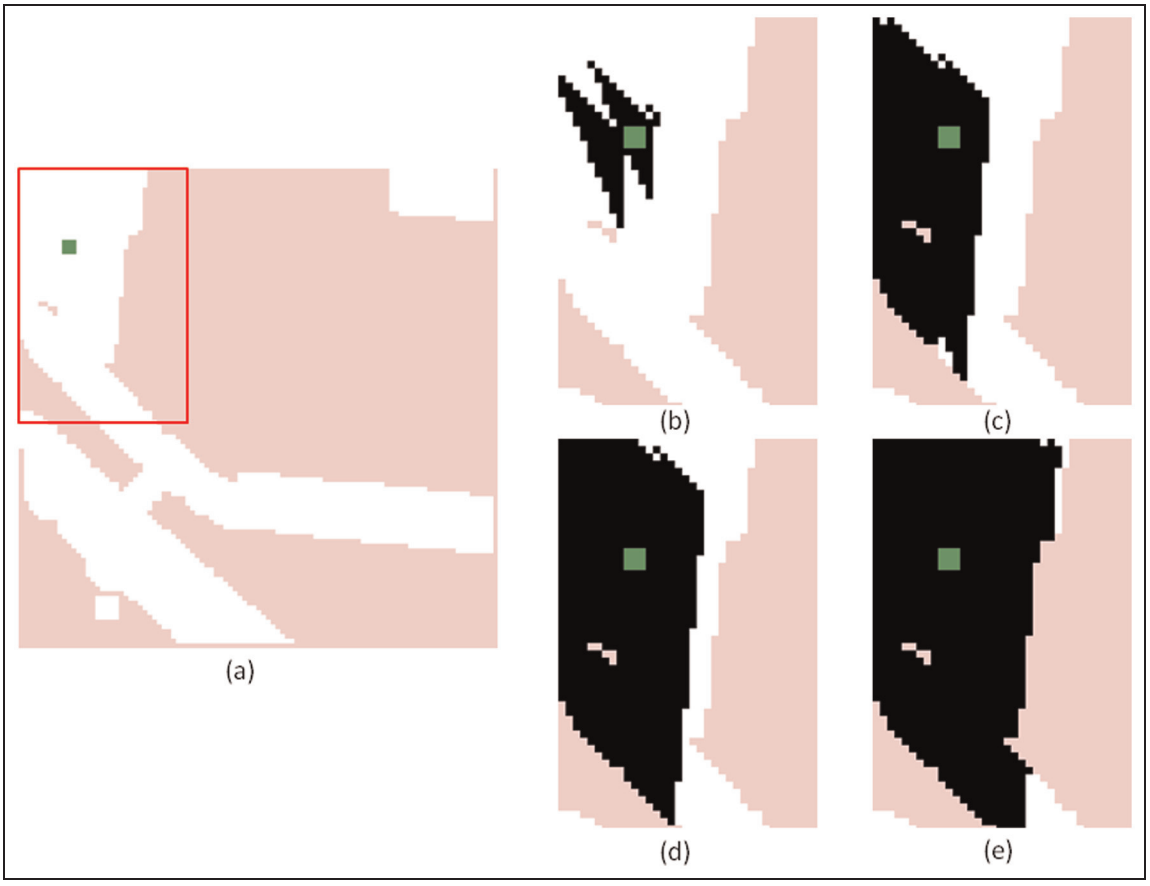

Figure 9. Floor field dispersion using Repast HPC supported mechanism. PoA is shown (a) inside the bounding box. Spread after iteration (b) 1, (c) 3, (d) 6 and (e) 10.

process 64 are shared between more than one processes (shown with black color).

Network synchronization: The network space shares the information between connected nodes (both patch and turtle), if peer agents reside in the adjacent processes. An example is shown in Figure 10(a) where two patches (linked with each other) are networked with each other residing on adjacent processes 64 and 74.

Patch and turtle classes are both treated as an agent and inherit the same field of ''agent id'', ''process id'', and ''agent type''. These three fields are essential to synchronize the buffer of networked space of adjacent processes both in case of reading or migrating. The other fields are specific to the functionalities of agents. For example, Table 1 lists the variables that are synchronized.

6.3.2. Process parameters. There can be situations in which information at the process level must be shared. For example, the ''density per unit walkable area''

$$process - unit - density = pop/wal \quad (1)$$

(where ''*pop*'' is equal to count of moving agents on the process and ''*wal*'' is the count of walkable patches at the process) is stored at process level and accessed by all of the agents to calculate the speed (see Figure 10(b)).

6.3.3. Global parameters. The inter-process parameters sharing is an expensive task but cannot be replaced by local/process-level data. For example, the density at each process is an absolute requirement for route decision during mobility (see Figure 10(c)). However, it cannot be executed in each iteration of the simulation due to high cost of execution.

6.4. Modeling agents behavior and adaptation

All agents are of type pedestrians. There are two types of pedestrians:

- Ambient intelligence (AmI)-assisted: the agents who have access to all of the information; process level as well as across the process.
- Non-AmI-assisted: the agents who have access to only local information (process level).

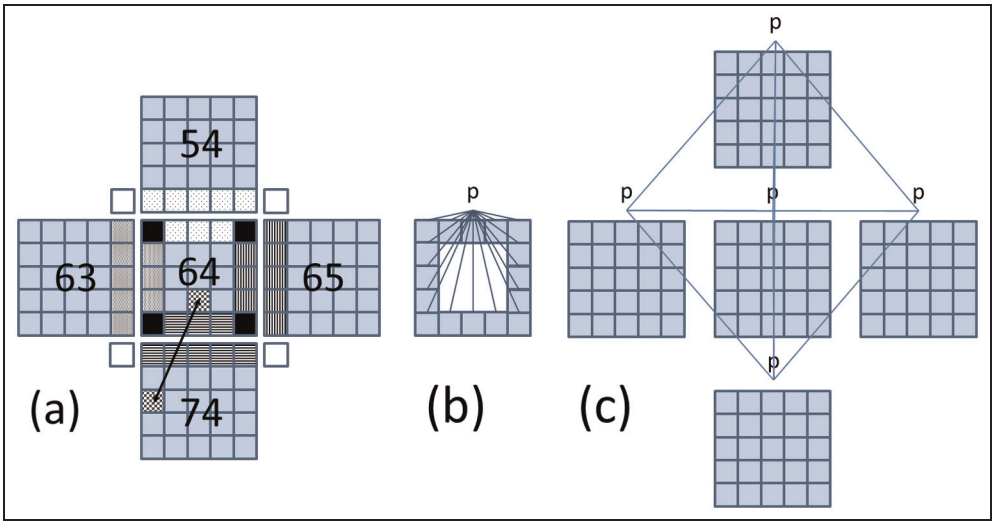

Figure 10. Data sharing across processes: (a) buffer and network synchronization; (b) process-level parameters; (c) inter-process parameters sharing.

Table 1. Variables synchronized across buffers.

Variable name	Agent type	Description
agent_id	Both	Unique ID of the agent
process_id	Both	Process ID where agent is currently residing
agent_type	Both	Type or class of the agent
structure_type	Patch	Type of the GIS space
DOM	Patch	Floor field
HOPC	Patch	Floor field
ROUTE	Patch	Floor field
orientation	Turtle	The current angle of direction which is a value for a PoA from patch's DOM
curr_exit	Turtle	The current PoA destination of the agent
wait	Turtle	How long an agent has been waiting to move (in terms of iterations)
speed	Turtle	The speed at which the agent would make the current move
following?	Turtle	Is the agent following an agent with better information than itself?

All agents can recognize the type of any other. All agents can access any information available at process and in the buffer zone.

At high level, randomly placed agents are destined towards one of the PoA as soon as they are able to do it. Agents use information present in the patch underneath, information available at process level and information synchronized at global level.

6.4.1. Agents creation and random placement. Agents are created based on fractions where each process gets a fraction equal to its walkable count as

$$count_{Type} = (Perc_{Type}/100) * ((ProccessW/TotalW) * TotalP) \quad (2)$$

where $Perc_{Type}$ is the percentage of agents of a particular type, $ProccessW$ is the count of walkable patches on the current process, $TotalW$ is sum of all walkable patches in the simulation space and $TotalP$ is total population of movable agents. Each agent is initialized with the following variables values: orientation(−999), current-exit(−1), speed(0), wait(−1), confidence(−1) and following(false), irrespective of agent type. An agent is generated on a patch not already occupied by another agent and is of the type walkable.

6.4.2. Next-step decision: locomotion and collision avoidance. There are two general-purpose procedures common to mobility strategies:

- Before proceeding with an actual move, an agent if already residing on a PoA would destroy itself with an increment of 1 in save count at the process level.
- For an actual move, a neighborhood-based ''next-step decision'' would be made by selecting the appropriate direction with a certain speed (step distance).

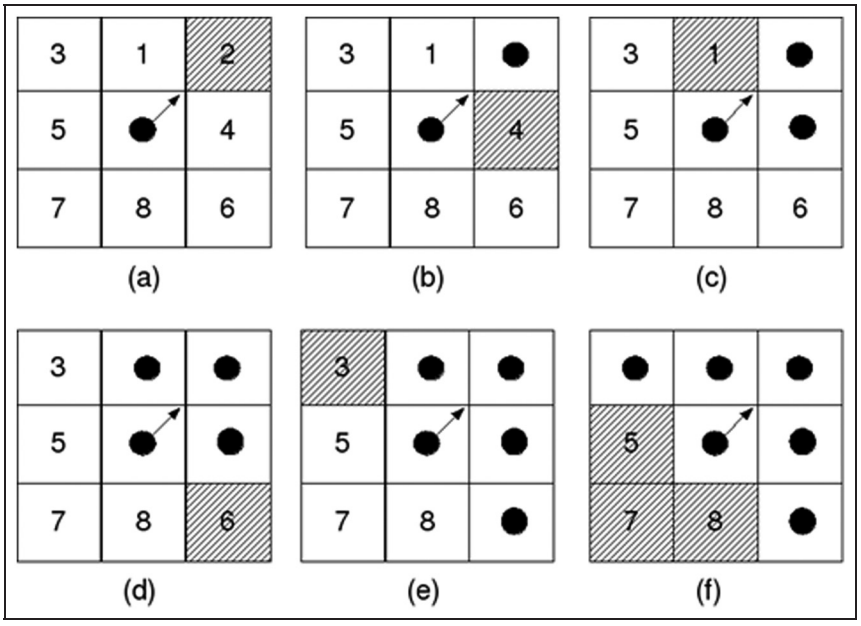

Figure 11. Possible configurations of an object moving at angle 45°. Black circles represent the cells occupied by an object. Hatched cells show the adopted choice. It is evident that as a strategy, we adopted a random choice between the last three cells (5, 7, and 8) if none of the first five cells are available.

6.4.3. Speed of agents. One aspect of proximic influence is the ''next-step decision'' seeking availability of one of the neighbors as described before. The sequence of preferences as described in Figure 11 are not intuition-based. We collected evidences to reach to this strategy.[62,63] In our previous work[62], we also introduced the notion of local density-based next cell selection considering a specific observation range. In the current simulation, we have not used local density-based direction decision. However, the speed of the agents is density based, where local observation range of density is that of local process. The formulation for speed is based on free flow speed and is given in the following equations:

$$speed\_on\_density = v_o * (1 - count(agents)/count(walkables)/max\_density\_per\_patch) \quad (3)$$

$$speed = max\{v_{min}, speed\_on\_density\} \quad (4)$$

where $v_o = 1.36$, max_density_per_patch is the maximum allowed pedestrians per patch ($= 1$) and $v_{min} = 0.0136$.

The procedure of the next-step behavior of an agent as soon as it has a valid value for orientation is given in Appendix B.

6.5. Modeling social influence

The true influence of social awareness on agents' decision making is described as the following behavior of agents (non-AmI-assisted) due to the presence of more informed (AmI-assisted) agents in the vicinity. Different possibilities can be considered starting from very basic where AmI-assisted agents are informed about densities of all of the sections of the city so that they make an *optimal* route decision and adhere to it. Any non-AmI-assisted agent in the interaction range of an AmI-assisted agent can then follow the AmI-assisted agents overwriting its own understanding about PoAs. The other possibilities are related with dispersion mechanism which can be varied in terms of how the information spreads or extent of connectivity between AmI-assisted agents.

The effects of availability and freshness of information to AmI-assisted agents is important. However, there is another question that is equally important: Why should a non-AmI-assisted agent follow an AmI-assisted agent? Currently we have only implemented ''unconditional following'' which is essentially simple and without any social consideration. However, in our earlier papers[57,64] we investigated the role of emotions in social decision making particularly following a leader or the formation of a group. The computational cognitive agent model anchored on trust and belief was proposed which integrates existing neurological and cognitive theories of affective decision making. Based on this model several variants of a large-scale crowd evacuation scenario with technically assisted agents were simulated. By analysis of the simulation results it was established that spread of emotions in a crowd increases resistance of agent groups to opinion changes and supports continuity of decision making in a group with technically assisted agents. The

general outcome is that in a system with scarce and uncertain information, AmI technology can be used to stimulate emergence of leaders and groups to increase the efficiency of evacuation.

6.6. Modeling mobility

In mobility we are considering the route selection strategies based on agent types, information extent (full, partial, or none) available as a floor field, information update rate/pattern, and principle of unconditional following of AmI-assisted agents. Numerous variations can be applied such as controlling the information dispersion mechanism, varying interaction ranges and population sizes, and introducing more sophisticated social behavior models. Currently we are taking the nearest measure to populate the DOM, HOPC, and ROUTE collections of the patches. We are not considering multiple routes for one PoA yet. There are two possibilities in which nearest can be inferred. One where we do nothing after a PoA x has a value in DOM, i.e. based on the information which is received first. Owing to the typical behavior of an instruction sequence in Repast HPC, this may not be exactly the nearest. However, this can be a good approximation. In the other we explicitly compare the hop count of PoA information and update the PoA if fresh information has hop count less than what a patch already had.

6.6.1. Mobility factors. In the mobility strategies there are three factors which are important: (i) population type; (ii) information granularity; and (iii) approximation considerations.

Population type. Mobility strategies are dependent on population (agent type/breed). We have considered three variations:

- With all agents having non-AmI-assisted type: Strategy 1 and Strategy 4.
- With all agents having AmI-assisted type: Strategy 2.
- With agents with a percentage of non-AmI-assisted as well as AmI-assisted type: Strategy 3 and Strategy 5.

Information granularity. Integrated with all mobility strategies, this is an assisting feature helpful in deciding the extent of information (full, partial, and none) an agent has while choosing one of the PoAs.

Approximation consideration. While using approximate information, the only motivation here is to save processing time if there is not much difference in quality. The approximations are realized using Strategy 4 instead of Strategy 1 and Strategy 5 instead of Strategy 3.

6.6.2. Mobility strategies.

Strategy 1. If all of the agents are of type non-AmI-assisted, only the local movement strategy would work where the selected PoA would be PoA with minimum hop count. The strategy would direct the agents towards the nearest PoA. The notion of nearest is exact here which means that we make sure that hop count in minimum irrespective of when the information is received. See Appendix C for the full pseudo-code of Strategy 1.

Strategy 2. If all of the agents are of type AmI-assisted, a decision would be made between available options by all of the agents. See Appendix E for the full pseudo-code of Strategy 2, described below.

If we define a PoA, poA, as a series of process identifiers forming a route (R), we can formalize this as $R = \{ID_{j_1}, ID_{j_2}, \ldots, ID_{j_N}\}$ where ID_{j_i} is the identifier for the process j_i. The subscript j denotes the index of the process in the route j. We assume that N processes form a given route. We compute the average density for each route as

$$\rho(poA_j) = \sum_{e=1}^{N} \rho(ID_{j_e})/N \tag{5}$$

The PoA selected, $poA*$, is chosen to be that with the minimum average density over the route. Formally,

$$poA* = poA_j \quad \text{with } min(\rho(poA_j)) \tag{6}$$

Strategy 3. In the case of a hybrid population of agents of type AmI-assisted and non-AmI-assisted, a mix of Move_to_Best_Available (AmI-assisted agents) and Move_to_Nearest_with_Follow (non-AmI-assisted agents) would be executed. A non-AmI-assisted agent would always seek for an AmI-assisted agent in the proximity and follow. The procedure shown in Appendix D is executed by each agent with type non-AmI-assisted. The following principle is unconditional and random chosen (in the case of more than 1 AmI-assisted agent in the surroundings). The AmI-assisted agent would adhere to Strategy 2 as described before.

Strategy 4. When compared with Strategy 1, this strategy considers the first information about a PoA as optimal in terms of hop count. The only motivation here is to save processing time if there is not much difference in quality.

Strategy 5. This is another variation of original strategy (Strategy 3 in this case) to save processing time if there is not much difference in quality. While non-AmI-assisted agents are performing lookup for AmI-assisted agents within interaction range, line 1 of the procedure Move_to_Nearest_with_Follow (Appendix D) would not be executed in each iteration but after a specific gap.

6.7. Dynamics over time

A trademark of a complex social system is evolving dynamics. An ABM should be able to handle unpredictability in agents' behavior and environmental conditions. We have handled change in states of the environment and updated mobility strategies to handle it; in particular, the situation in which the availability of PoA varies with time. Based on the possible real-world situations, we have taken three cases:

- Case 1: One of the PoAs x abruptly becomes unavailable but readily becomes available again. The information about its non-availability reaches everywhere instantly. Information about its re-availability disperses as the PoA x is new and has no history.
- Case 2: One of the PoAs x abruptly becomes unavailable and remains as it is. The information about its non-availability reaches everywhere instantly.
- Case 3: One of the PoA x abruptly becomes unavailable and remains as it is. The information about its non-availability disperses in step-by-step fashion.

We used Strategy 5 of mobility to incorporate these changes. Taking i as the simulation iteration when the floor field across the map has been populated with all possible PoAs, we described the following strategies according to the three cases given above.

Strategy 6. At iteration i, the contents against PoA x are erased from DOM, HOPC, and ROUTE collections of all of the patches. At iteration $i+1$, the information dispersion mechanism shown in Appendix A starts again for PoA x.

Strategy 7. At iteration i, the contents against PoA x are erased from DOM, HOPC, and ROUTE collections of all of the patches.

Strategy 8. At iteration $i+1$, a modified version of procedure shown in Appendix A disperses the information about non-availability of PoA x throughout the space. This means that unless the patches receive this information, they would consider PoA x valid.

7. Software and hardware considerations

To change over from a single CPU system to a SMP environment not only allows us to perform simulations on larger models (space geometry, number of agents) within reasonable time, but also allows us to simulate more complex behavior. For example, it would enable us to simulate behavioral aspects of agents, such as trust, belief, decision making, etc. and with all of its mutual influences on a city-scale level. It has to be pointed out that the factor of speed-up actually achievable with a model executed on a SMP system is highly dependent on inter-agent interactions. While execution time scales almost linearly with the number of cores for mainly independent agents, it gets extremely complex and difficult to model for highly interdependent agents.[65] Furthermore, parallel processing, in contrast to execution on single CPU systems, requires issues such as synchronization (between spatially adjacent processes) and mutual influence (e.g. the density of agents in one space fragment or process has an influence on other space fragments), to be dealt with. Where most SMP frameworks are able to provide built-in mechanisms to achieve synchronization, the global parameters exchange across all of the processes has to be explicitly managed by the programmer.

7.1. Repast HPC

Repast HPC[55] is an ABM and simulation framework for high-performance distributed computing platforms written in C++ and using MPI[66] for parallel operations. It is designed for parallel environments where many processes are running in parallel and where the agents themselves are distributed across processes. Shared, synchronized spaces are used for passing an agent from one process over to another, or to gather information such as agent density, blocked exits, etc. from the neighboring processes. In addition to writing ''pure'' C++ applications, Repast HPC also allows simulation models to be developed in a Logo-like language similar to the commands used by the NetLogo framework (http://ccl.northwestern.edu/netlogo/).

7.2. Systems

To discover variety in the different modeling approaches and to gain experience in potentials, problems, etc. the different platforms might have the full range of environments, starting with a single CPU system, continuing with a fine-grained parallel processor (GPU), a SMP system with 768 cores (SGI Altix 4700), and finally a SMP with 2048 cores (SGI Altix UV) was utilized to challenge different evacuation (i.e. agent movement enriched with individual and group behavior) models. Subsequently, we give a list of the hardware/software systems used:

- GPU: NVidia GeForce 9700M GT, G96 PU (625 MHz), 32 stream processors, 512 MB GDDR3, DirectX 10, Shader 4.0;
- SMP768: SMP, SGI Altix ICE 8200, 768 cores, type Intel Xeon 2.5 GHz, 1.45 TB shared memory;
- SMP2048: SMP, SGI Altix UltraViolet 1000, MIMD, 2048 cores (256 Intel Xeon E78837 (WestmereEX) CPUs, 2.66 GHz, 24MB L3 Cache); 16 TB shared memory (ccNUMA), 21.3 TFlops, 192 TB SAS HDD.

7.3. Comparison between systems

First, possible improvements (i.e. performance speed-up) between an atomic (single) processing unit and the different parallel hardware architectures listed above were evaluated separately for different facets of a model. Related work has revealed that the highest potential for speed-up is offered by the next-step decision of individual (independent) agents. In this case, possible improvements are almost unbounded, only limited by the number of CPUs/cores (and their computation power) available. Using for example GPU, speed-up in the range of 100 to 250 can be easily achieved (reason is the highly parallel pipelining architecture).[67] If the decision making of an agent is rather independent from individual data of other agents or by using optimization methods for models based on proximity relations (i.e. neighborhood), potential acceleration is still in the range of a factor of 100.[68,69] Using a simulation model with a (global) mailbox system for the synchronized exchange of messages, a speed of factor 50 is realistic.[70] The more the behavior of a single agent is influencing other agents, the less is the potential of speed-ups. Even using sophisticated branch prediction schemes might not bring that much, as synchronization and rollback are the dominant routines.

What we have found out on executing a simulation model with 10^7 agents processing hypothetical workload (''move'', ''adapt'', and ''interact'' procedures) is that: (i) the level of speed-up is constant within a cluster of given size (number of agents assigned to a process) independent from the interaction extent with the cluster (varying from 0 to 100%); (ii) the speed-up increases on a logarithmic scale with the size of the clusters; and (iii) that (at least for the special case processed) model execution on the single GPU outperforms the cluster machine (using 128 cores). For more details we refer to Moser.[56]

To assess the difference (in terms of execution performance) of the two available cluster machines (SMP768, SMP2048), different variants of the simulation models were later executed on both systems in a similar way. Dependent on the simulation strategy, the speed-up achieved for SMP2048 as compared with SMP768 on a (rather small) simulation model involving 25 processes with 250 agents each (6250 agents), and after 500 iterations is in the range between two and more than three.

Hence, both the test and real-scale simulation uses SMP2048.

8. Evacuation simulation and selected results

8.1. Scenario

The simulation is performed at two scales: small scale and full city scale. The small-scale simulation serves to evaluate the methodologies and models, connecting the simulation efficiency with that of agents' behavior. Based on the small-scale simulation results, the most suitable mobility strategy in each category of behaviorally ''similar'' strategies is chosen based on simulation efficiency and tolerance to functional equality. Only these chosen strategies are then simulated at the scale of the city.

At both scales, the raster-map-based space is distributed across multiple processes. Each patch of the space is either walkable or not. The space is also augmented with PoAs where agents need to move to based on the strategy they are following. The agents can be of two types: AmI-assisted and non-AmI-assisted.

The exit strategies can be categorized into three types: (i) nearest (Strategy 1 and 4); (ii) following (Strategy 3 and 5); and (iii) following with evolving space (Strategy 6, 7, and 8). In category (i), there is no AmI assistance and all of the agents move to the ''nearest'' PoA based on the floor field at time. In category (ii), there is a hybrid population of AmI-assisted (5%) and non-AmI-assisted (95%) agents. The AmI-assisted agents calculate the ''optimal'' PoA based on: (a) the floor field at time; (b) distances to available PoAs; and (c) accumulated regional densities along the ROUTE of available PoAs. The non-AmI-assisted agents, if possible, just follow one of the AmI-assisted agents in proximity. In category (iii), one of the variations in (ii) acts as a base case (Strategy 5). In addition, after a specified time, one PoA becomes unavailable which affects the agents' subsequent choices. Strategies 6, 7, and 8 explore different possibilities of state change of PoAs with respect to how the information about non-availability of one of the PoAs is dispersed throughout the floor field.

It is important to note that floor field is evolving with time. It would predominantly not affect Strategy 1 or 4 where it can safely be assumed that the CA-based field spreading mechanism would guarantee that a patch receiving the information about the first PoA would generally be the ''nearest'' PoA. However, it can affect the relative behavior of strategies in categories (ii) and (iii). The (AmI-assisted) agents would choose an ''optimal'' PoA out of the available few (starting with 1) at the start of the simulation. At that time, an optimal PoA would be more ''nearest'' than ''optimal''.

8.2. Small-scale evaluation study

The test space consists of $5 \times 5\ (=25)$ processes and equals a dimension of 500×500 cells. This space represents one section of the city which is roughly $1/500$th of the city map (highlighted in Figure 3). Each process (out of 25) is equal to 100×100 cells. The small-scale simulated space can be seen in Figure 12. At this scale, the PoAs are not realistic and are intuitively chosen. The PoAs are labeled with ID of the process they reside in and PoA 12 represents the exit which changes its state from ''available'' to ''unavailable'' after a specified time.

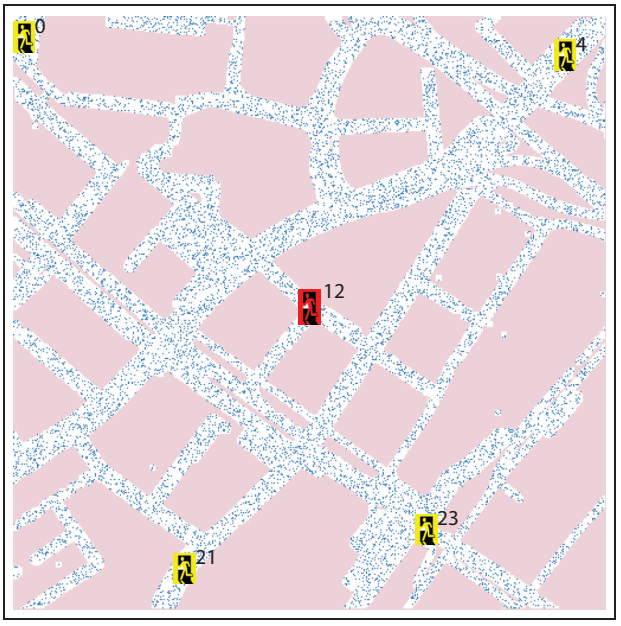

Figure 12. The small-scale scenario distributed across 25 (5×5) sectors. Each sector was simulated by one processor. The exits (PoAs) are highlighted at sectors 0, 4, 12, 21, and 23. The PoA at sector 12 is flooded during the simulation.

The space in Figure 12 also shows the density of agents at the start of the simulation, which is 12,500 agents distributed across 25 processes based on the availability of walkable space. All 8 mobility strategies were simulated over 500 iterations. The PoA 12 changes its state from ''available'' to ''unavailable'' at iteration 150.

Discussion. Since the overall purpose of simulation is evacuation here, the analysis of the results is anchored at evacuation patterns and efficiency. The purpose of the small-scale simulation was to evaluate the execution time of the strategies with the behavioral difference. We noted that there was a significant difference in execution time when comparing Strategy 1 with Strategy 4 and that of Strategy 5 with Strategy 2 (see the table in Figure 15), whereas the behavioral difference is acceptable.

While comparing Strategy 1 and Strategy 4 based on graphs shown in Figure 13 and Table 2, it is evident that the notion of ''nearest'' is affected by the floor field spreading mechanism due to spatial features of PoAs. PoAs 0 and 21 have narrow scope when compared with PoAs 4 and 23 (see Figure 12). That is the reason the percentage of agents reaching PoAs 0 and 21 has dropped when comparing Strategy 4 with Strategy 1. The exact opposite of that has happened with PoAs 4 and 23 where PoA 4 is broader in a spatial sense and receives the best rise. Although the PoA 12 is almost in the center, it does not mean that it is also at the best place with respect to spread of CA-based information. That is the reason, the percentage of agents reaching PoA 12 has dropped when compared with Strategy 1 (representing more realistic nearest). Another important factor is to consider the overall efficiency of the two strategies. It is evident from Figure 14 that due to agents moving towards ''real'' nearest almost all of the time, Strategy 1 is able to evacuate much faster than Strategy 4; 20% more efficient to be exact. However, there is no difference in total number of agents evacuated in 500 iterations. In addition, the execution time of Strategy 1 is nearly seven times that of Strategy 4 (see Figure 15). Hence, for full-scale simulation, instead of Strategy 1 we opted for Strategy 4.

Strategy 3 (and 5) disperses the agents more evenly across the PoAs when compared with Strategy 1 where load is shed from PoA 12 towards PoAs 0, 4, and 23. This means that even a small percentage of AmI-assisted agents are able to make an impact. While comparing Strategy 3 and Strategy 5 based on graphs shown in Figure 13 and Table 2, it is evident that there is not much difference in PoA utilization and evacuation speed. Whereas, the execution time of Strategy 3 is nearly three times that of Strategy 5 (see Figure 15). Hence, for full-scale simulation, instead of Strategy 3 we opted for Strategy 5.

Both Strategy 2 and Strategy 6 are not realistic settings at the scale of the city. That is why we have not considered these for city-scale simulation. Strategy 7 (and 8) is able to reroute the agents towards new destination if PoA 12 becomes unavailable during the simulation. It can be seen in Figure 13 and Table 2 that the agents previously destined for PoA 12 are rerouted towards PoAs 4, 21, and 23. This ''transfer'' is dependent on the location of the PoAs where PoAs 23 and 4 get more attention than PoA 21, whereas PoA 0 remains unaffected. Strategy 6 and Strategy 7 look similar in terms of PoAs usage and execution time. However, Strategy 7 has much better speed of evacuation than Strategy 8 due to abrupt knowledge being spread about unavailability of PoA 12. This diminishes the possibility of unproductive moves towards PoA 12. We chose Strategy 7 over Strategy 8 based on these benefits.

8.3. City-scale evacuation simulation

For city-scale simulation, the map was divided into 200 sectors (see Figure 4) distributed as $m \times n$ along horizontal and vertical axis, respectively, where m is equal to 10 and n is equal to 20. Each of these 200 sectors (or processors) consisted of a space equal to $x \times y$ cells where x and y are equal to 500 cells each. A total of 200,000 agents were simulated which corresponded to the population of the city. Each sector got its share of agents based on fraction of walkable space it has. At this scale, the PoAs are realistic and represent crowd attractions such as city center (PoA 64), industrial area (PoA 96), city stadium (PoA 124), suburban center (PoA 22), and motorway exit (PoA

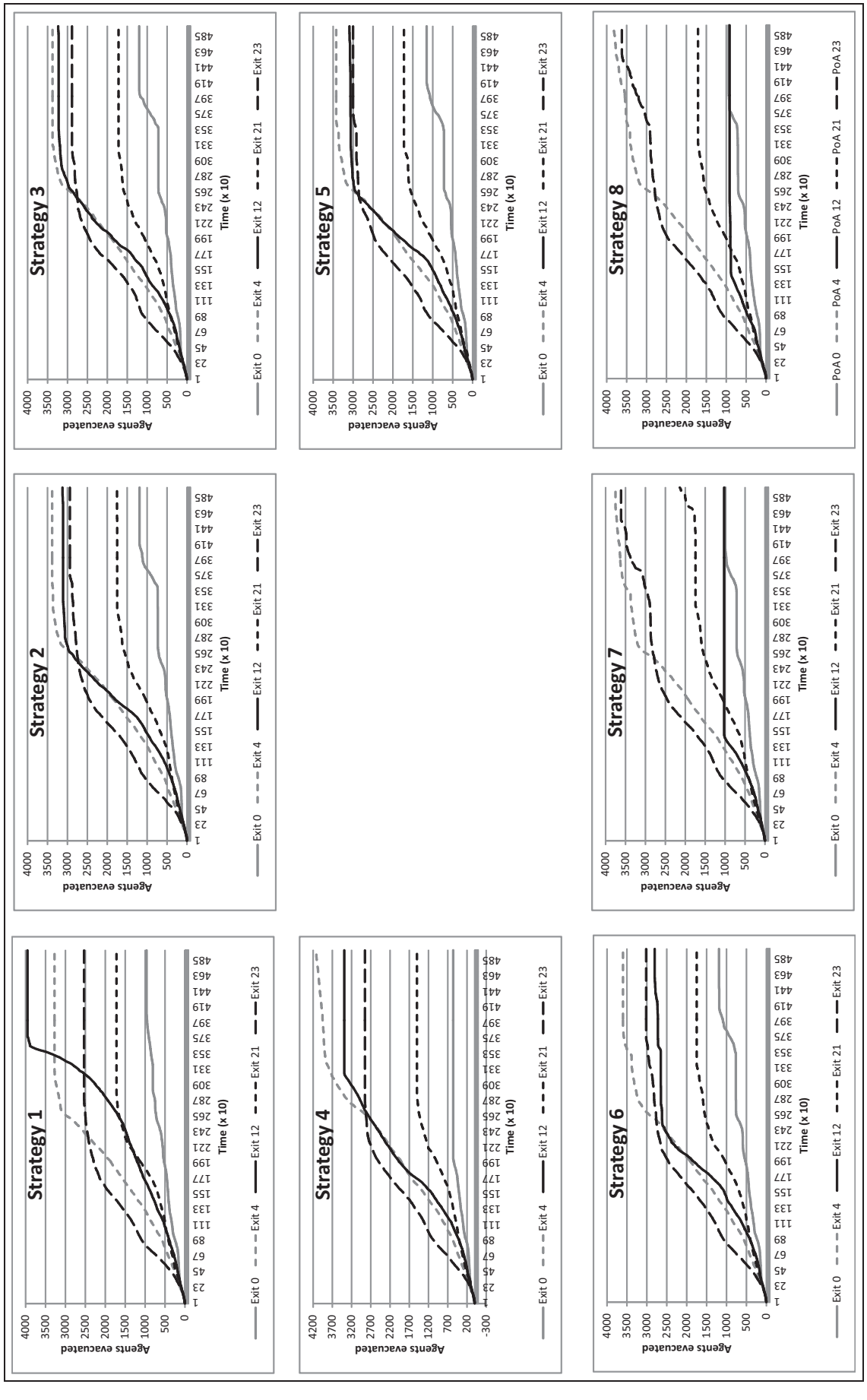

Figure 13. Small-scale results: individual PoAs plots. The graphs show time series of evacuating agents through the PoAs (i.e. exit 0, 4, 12, 21, and 23) for strategies 1–8.

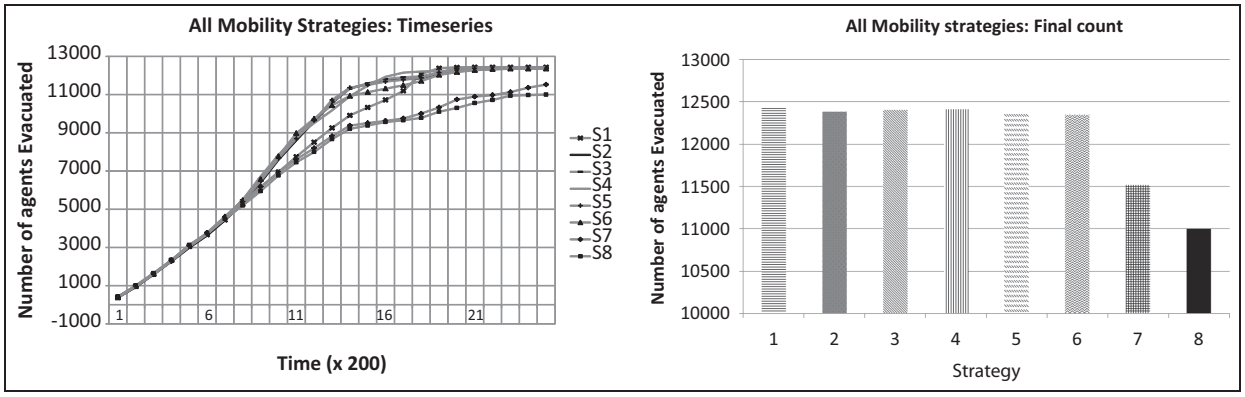

Figure 14. Small-scale results: comparing strategies irrespective of PoAs usage. Strategies 1–8 are labeled "S*n*" where *n* represents the strategy number.

Mobility Strategy	**CPU Time** *hh:mm:ss*	**Memory** *kb*	**Virtual Mem.** *kb*	**Wall Time** *hh:mm:ss*
Strategy 1 (0 % AmI-assisted, 100 % Non-AmI-assisted)	11:32:22	1068744	20562304660	00:27:58
Strategy 2 (100 % AmI-assisted, 0 % Non-AmI-assisted)	01:56:21	1067968	20562303596	00:04:53
Strategy 3 (5 % AmI-assisted, 95 % Non-AmI-assisted)	06:56:40	1067512	20562302904	00:16:57
Strategy 4 (0 % AmI-assisted, 100 % Non-AmI-assisted)	01:57:28	1065856	20562301428	00:04:54
Strategy 5 (5 % AmI-assisted, 95 % Non-AmI-assisted)	02:24:3	1068116	20562303472	00:06:02
Strategy 6 (5 % AmI-assisted, 95 % Non-AmI-assisted)	02:27:33	1067432	20562302588	00:06:37
Strategy 7 (5 % AmI-assisted, 95 % Non-AmI-assisted)	01:52:50	1052036	20562287592	00:04:42
Strategy 8 (5 % AmI-assisted, 95 % Non-AmI-assisted)	02:13:37	1061136	20562296924	00:05:33

Figure 15. Small-scale evaluations: comparative PDS performance between mobility strategies.

Table 2. Small-scale simulation: comparison between percentage of agents evacuating from PoAs.

Strategy	PoA 0	PoA 4	PoA 12	PoA 21	PoA 23
1	8%	26%	32%	14%	20%
4	4%	33%	27%	12%	22%
3	10%	27%	26%	14%	23%
5	9%	27%	25%	14%	24%
7	9%	33%	10%	19%	31%
8	8%	35%	8%	16%	33%

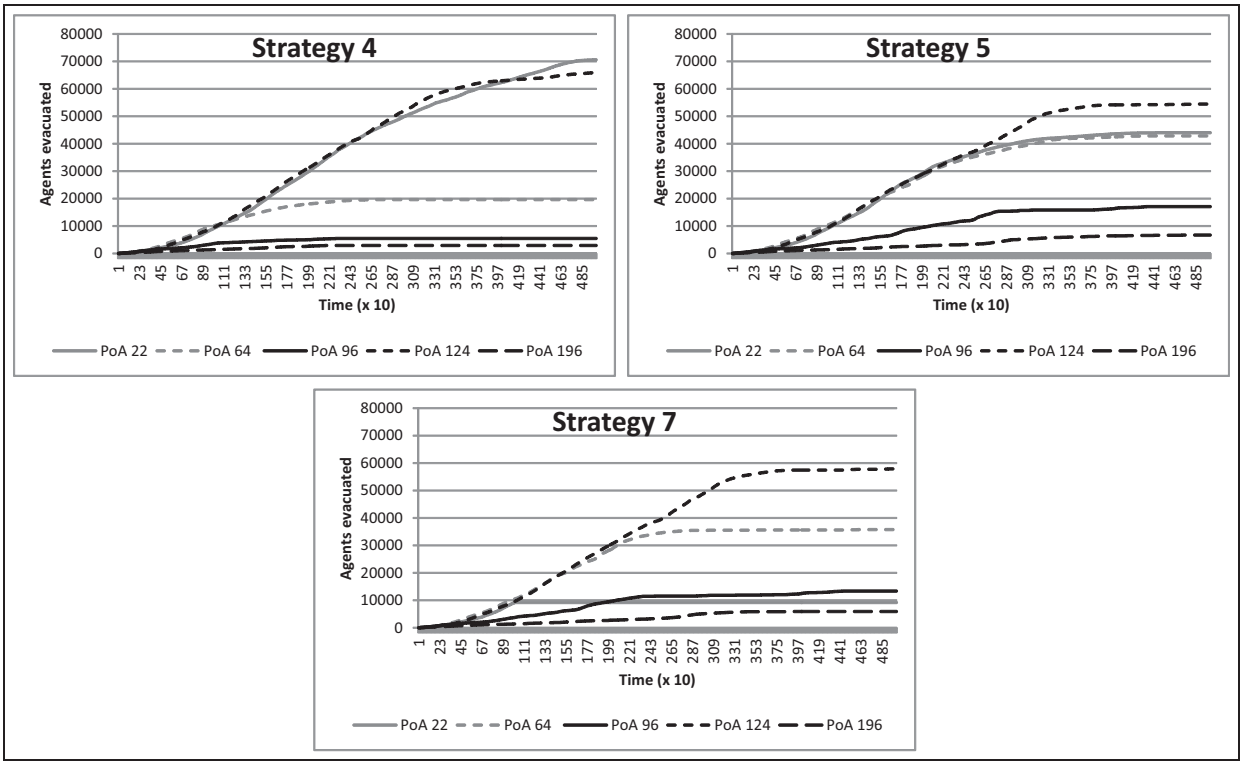

Figure 16. City-scale results: individual PoAs plots. The graphs show time series of evacuating agents through PoAs 22, 64, 96, 124, and 196 for Strategies 4, 5, and 7.

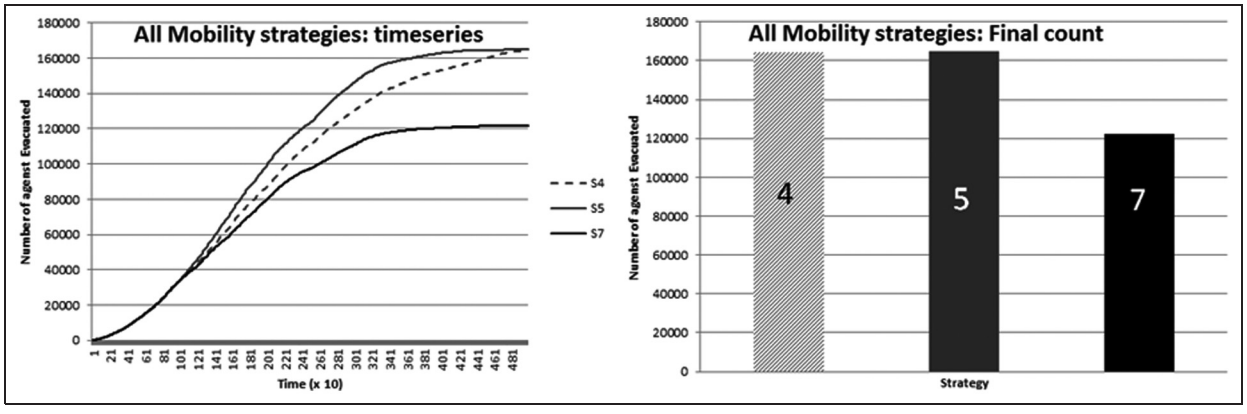

Figure 17. City-scale results: comparing strategies irrespective of PoAs usage. Strategies 4, 5, and 7 are labeled "S*n*" where *n* represents the strategy number.

196). The simulation was run for 5000 iterations. The PoA 22 changes its state from ''available'' to ''unavailable'' at time 1000.

Discussion. We present the simulation results by computing the total number of agents exited over time, shown in terms of iterations for Strategies 4, 5, and 7. In Figure 17, we plot the total number of exited agents over time, irrespective of PoAs. We can see that with Strategy 5, more agents exit at an earlier time: 80% of the agents exited by iteration 3690 in Strategy 5 whereas 80% of the agents exited by iteration 4540 in Strategy 4. Obviously Strategy 7 cannot compete with Strategy 5 as transfer of agents from PoA 22 towards other PoAs would require more time. That is the reason Strategy 7 was only able to evacuate 60% of the agents in a given time.

Next we look more closely into the results by considering the number of exited agents per PoA (see Figure 16 and Table 3). Overall, we can see that in Strategy 5 the PoAs are more evenly used, i.e. the least used exits from

Table 3. City-scale simulation: comparison between percentage of agents evacuating from PoAs.

Strategy	PoA 22	PoA 64	PoA 96	PoA 124	PoA 196
1	43%	12%	3%	40%	18%
4	27%	26%	10%	33%	4%
7	8%	29%	11%	47%	5%

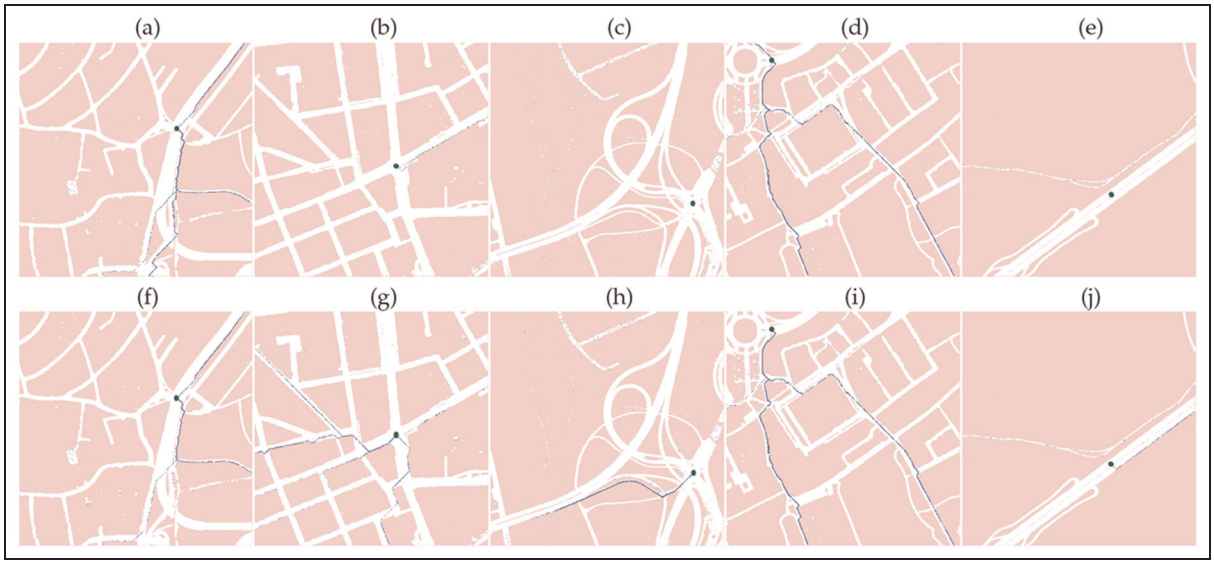

Figure 18. Visualization of 5 exits half-way through the simulation (iteration 2500) for Strategy 4 (a)–(e) and Strategy 5 (f)–(j). Agents are demonstrated as tiny dots. The point in each figure visualizes the exit. We can see the more even distribution of exit usage in Strategy 5, especially in (b) versus (g), where more paths are created towards the exit. In addition (c) versus (h) and (e) versus (j) illustrate the less central exits continue to be used in Strategy 5, although not in Strategy 4.

Strategy 4 have more usage in Strategy 5, and the highest used exits from Strategy 4 have less usage in Strategy 5. This results in a delayed flattening out of many of the curves in Strategy 5, which results in the earlier exiting by agents. In Strategy 4, only 2 out of 5 PoAs remain active whereas the others flattens out in first half of the simulation.

The actual positions of the agents in the five processes containing PoAs are shown in Figure 18. Plots (a) to (e) show the configuration at iteration 2500, exactly half way through the simulation for Strategy 4. Exactly the same frames at iteration 2500 are shown for Strategy 5 in (f) to (j). The agents are the small dots, often lined up at the exiting points. The largest noticeable difference can be seen for exit 2 ((b) versus (g)), where several paths are developed towards the exit in (g), although only one is taken in (b). This clearly shows the difference in the process densities resulting in more diverse paths towards the exit for Strategy 5. Also, at exit 3 ((c) versus (h)), in Strategy 4 the exit is no longer used half way through the process, however, it is still used in Strategy 5. A similar conclusion can be made for exit 5 ((e) vs. (j)), where this exit is quite distant and non-central, however, it is made to better use in Strategy 5 by the agents.

Visual comparison of Strategies 4, 5, and 7 is given in Figure 19. It is evident that agents start to disperse away from PoA 22 towards others PoAs around iteration 2500 when comparing Strategy 4 with Strategy 5. When comparing Strategy 5 with Strategy 7, it can be concluded that both are similar before iteration 1000. When PoA 22 becomes unavailable, we can see migration of agents from PoA 22 to other PoAs (at $t = 1100$). Later, there are no agents around PoA 22 at iteration 1500.

9. Conclusion

The potential of PDS for an agent-based geo-simulation can only be materialized if in addition to an efficient hardware architecture, the algorithmic optimization is also taken care of in order to fully utilize the ABM strength in which each agent may potentially have a unique behavior. The scale becomes a real issue if the focus is an urban space with billions of space agents in addition to millions

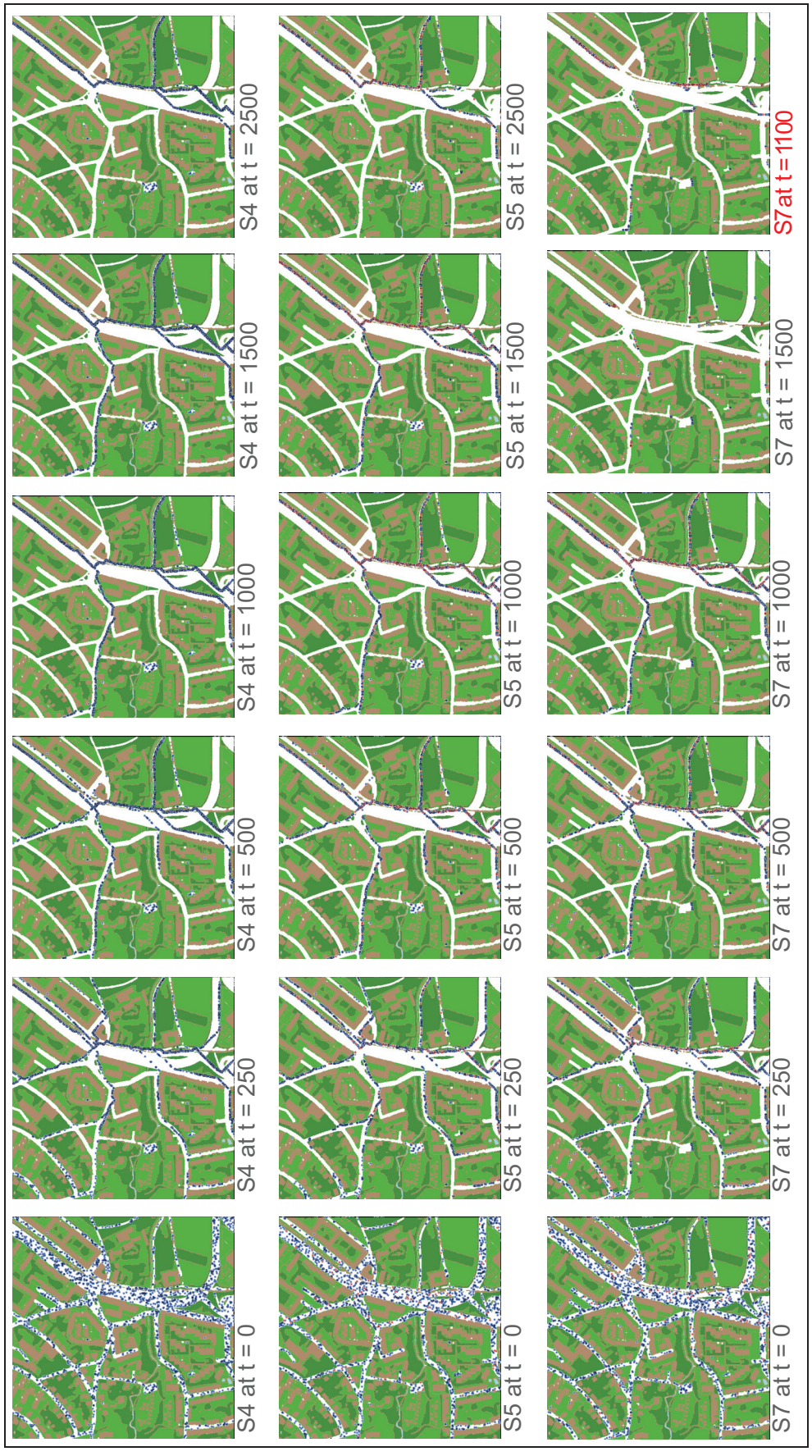

Figure 19. Visualization of the agents around PoA 22 for Strategy 4 (upper row), Strategy 5 (middle row), and Strategy 7 (lower row). Strategies 1 to 3 are labeled "S*n*" where *n* represents the strategy number and *t* represents the iteration number.

of mobile agents. Fortunately the space agents (e.g. cells space in CA) are usually not as diverse in function as moving agents can be, mainly representing spatial features of the environment which can be as basic as a bitmap representing walkable and non-walkable space in a map. However, a space like that would require a smart information dispersion mechanism to be incorporated, so that mobile agents are able to make mobility decisions. In addition to an efficient floor field spread, the interaction of moving agents with others should also be minimized within tolerable limits.

While running a simulation at the scale of a medium-sized city, one of the mobility strategies (Strategy 1) can be that agents always move towards the nearest PoA. However, this requires updates based on the neighborhood field in each of the simulation iterations. This is a choice clearly not practical as we observed that we had to abort the simulation run after 278 hours in which only the 4% of the simulation was complete.[71] By modifying Strategy 1 into Strategy 4, which considers the first information received about a PoA as the final information, a full simulation run was possible in around 12 hours. The change is strategy does not incur much difference in mobility pattern due to the large scale. Similarly we were able to run the simulation with Strategy 5 in 98 hours which was not feasible at all with Strategy 3.[71] Strategies 5 and 3 differ only in terms of moving agents interaction periodicity.

Simulation of urban mobility is a complex task with respect to variety of aspects that are important. We have tried to conceptualize these aspects into categories and designed an agent-based PDS framework to simulate. This is an ongoing research where we intend to enrich the agents' models with more data, information, and behavioral rules for future works.

Funding

This work is supported under the FP7 ICT Future Enabling Technologies program of the European Commission (grant agreement number 231288; SOCIONICAL).

References

1. Troisi WVA and Ratner M. An agent-based approach for modeling molecular self-organization. *Proc Natl Acad Sci USA* 2005; 102: 255–260.
2. Folcik V and Orosz C. The basic immune simulator: an agent-based model to study the interactions between innate and adaptive immunity. *Theoret Biol Med Model* 2007; 4: 39.
3. Mock K and Testa J. An Agent-based Model of PredatorPrey Relationships between Transient Killer Whales and Other Marine Mammals. University of Alaska Anchorage, Anchorage, AK, 2007. Available from: http://www.math.uaa.alaska.edu/Borca/.
4. Yassemi S, Dragicevic S and Schmidt M. Design and implementation of an integrated GIS-based cellular automata model to characterize forest fire behaviour. *Ecol Modell* 2008; 210: 71–84.
5. Almeida CM, Gleriani JM, Castejon EF and Soares-Filho BS. Using neural networks and cellular automata for modelling intra-urban land-use dynamics. *Intl J Geog Inform Sci* 2008; 22: 943–963.
6. Sun T and Wang J. A traffic cellular automata model based on road network grids and its spatial and temporal resolution's influences on simulation. *Sim Modell Practice Theory* 2007; 15: 864–878.
7. Sallach D and Macal C. The simulation of social agents: an introduction. *Soc Sci Comp Rev* 2001; 19: 245–248.
8. Carley K, et al. Biowar: scalable agent-based model of bioattacks. *IEEE Trans Syst Man Cybernet* 2006; 36: 252–265.
9. Wiki: Computational Sociology,. Available from: http://en.wikipedia.org/wiki/Computational_Sociology#cite_note-MW-0.
10. Helbing D, Yu W and Rauhut H. Self-organization and emergence in social systems: modeling the coevolution of social environments and cooperative behavior. *J Math Soc* 2011; 35: 177–208.
11. Pan XS, Han CS, Dauber K and Law KH. A multi-agent based framework for the simulation of human and social behaviors during emergency evacuations. *AI Society* 2007; 22: 113–132.
12. Sawyer R. *Social Emergence: Societies ad Complex Systems*. Cambridge: Cambridge University Press, 2005.
13. Macy MW and Willer R. From factors to actors: computational sociology and agent-based modeling. *Ann Rev Sociol* 2002; 28: 143–166.
14. Gilbert N. A simulation of the structure of academic science. *Sociol Res Online* 1997; 2(2).
15. Axelrod RM. *The Complexity of Cooperation: Agent-based Models of Competition and Collaboration*. Princeton: Princeton University Press, 1997.
16. Xiong M, Cai W, Zhou S, et al. A case study of multi-resolution modeling for crowd simulation. In: *Proceedings of the 2009 Spring Simulation Multiconference (SpringSim '09)*, San Diego, CA: Society for Computer Simulation International, 2009, pp. 17:1–17:8.
17. Pan CDK, Han X and Law K. Human and social behavior in computational modeling and analysis of egress. *Automat Construct* 2006; 15: 448–461.
18. Trist EL. *The Evolution of Socio-technical Systems: A Conceptual Framework and An Action Research Program* (*Issues in the Quality of Working Life: A Series of Occasional Papers*, no. 2). Toronto, ON: Ontario Quality of Working Life Centre, 1981.
19. Valderrama A and Jörgensen U. Urban transport systems in Bogot and Copenhagen: an approach from STS. *Built Environment* 2008; 34: 200–217.
20. Fox WM. Sociotechnical system principles and guidelines: past and present. *J Appl Behav Sci* 1995; 31: 91–105.
21. Xiong M, Lees M, Cai W, Zhou S and Low MYH. Hybrid modelling of crowd simulation. *Proc Comput Sci* 2010; 1: 57–65.
22. Chen D, Wang L, Wu X, et al. Hybrid modelling and simulation of huge crowd over a hierarchical Grid architecture. *Fut Gen Comput Syst* 2012; in press.

23. Chen D, Theodoropoulos GK, Turner SJ, Cai W, Minson R and Zhang Y. Large scale agent-based simulation on the grid. *Fut Gen Comput Syst* 2008; 24: 658–671.
24. Iskra KA, van Albada GD and Sloot PMA. Toward grid-aware time warp. *SIMULATION* 2005; 81: 293–306.
25. Solar R, Suppi R and Luque E. High performance distributed cluster-based individual-oriented fish school simulation. *Proc Comput Sci* 2011; 4: 76–85.
26. Perumalla KS, Aaby BG, Yoginath SB and Seal SK. Interactive, graphical processing unit-based evaluation of evacuation scenarios at the state scale. *SIMULATION* 2012; 88: 746–761.
27. Bryan J. Human behavior and fire. In: Cote A (ed.), *Fire Protection Handbook*, 19th edn, Vol. 1. National Fire Protection Association, 2003, pp. 4.3–4.32.
28. Ferscha A and Zia K. LifeBelt: crowd evacuation based on vibro-tactile guidance. *IEEE Pervasive Comput* 2010; 9(4): 33–42.
29. Ferscha A and Zia K. On the efficiency of LifeBelt based crowd evacuation. In: *13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications (DS-RT 2009)*, 25-28 October, Singapore. Los Alamitos, CA: IEEE Computer Society Press, 2009.
30. Zia K and Ferscha A. A simulation study of exit choice based on effective throughput of an exit area in a multi-exit evacuation situation. In: *13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications (DS-RT 2009)*, 25–28 October, Singapore. Los Alamitos, CA: IEEE Computer Society Press, 2009.
31. Zia K and Ferscha A. Self-organized evacuation based on LifeBelt. In: *Proceedings of the 3rd International Workshop on Self-Organizing Systems (IWSOS 2009)* (*Lecture Notes in Computer Science*, vol. 5918, pp 250–255). Berlin: Springer-Verlag, 2009.
32. Bon L. *The Crowd: A Study of the Popular Mind*. London: Unwin, 1908.
33. Lovas GC. Modeling and simulation of pedestrian traffic flow. *Transport Res* 1994; 28: 429–443.
34. Li CT, Hsieh HP, Kuo TT and Lin SD. SocioCrowd: a social-network-based framework for crowd simulation. In: *ACM SIGGRAPH 2010 Posters*, 2010.
35. Kempe D, Kleinberg J and Tardos É. Maximizing the spread of influence through a social network. In: *ACM SIGKDD 2003*, 2003.
36. Schadschneider A, Kirchner A and Nishinari K. CA approach to collective phenomena in pedestrian dynamics. In: Bandini BC and Tomassini M (eds), *5th International Conference on Cellular Automata for Research and Industry, ACRI 2001*, Geneva, Switzerland, 9–11 October 2001 (*Lecture Notes in Computer Science*, vol. 2493). Berlin: Springer, 2002, pp. 239–248.
37. Helbing D, Farkas IJ, Molnár P and Vicsek T. Simulation of pedestrian crowds in normal and evacuation simulations. In: Schreckenberg M and Sharma S (eds), *Pedestrian and Evacuation Dynamics*. Berlin: Springer, 2002, pp. 21–58.
38. Hoogendoorn SP. Pedestrian travel behavior modeling. In: *Proceedings of Travel Behavior Research*, Lucerne. Amsterdam: Elsevier, 2003, pp. 10–15.
39. Penn A and Turner A. Space syntax based agent simulation. In: Schreckenberg M and Sharma S (eds), *Pedestrian and Evacuation Dynamics*. Berlin: Springer, 2002, pp. 99–114.
40. Borgers A and Timmermans H. A model of pedestrian route choice and demand for retail facilities within inner-city shopping areas. *Geograph Anal* 1986; 18: 115–128.
41. Gardner M. The fantastic combinations of John Conway's new solitaire game ''Life''. *Sci Amer* 1970; 223: 120–123.
42. Epstein J and Axtell R. *Growing Artificial Societies Social Science from the Bottom Up*. Cambridge, MA: MIT Press, 1996.
43. Dijkstra J, Timmermans HJP and Jessurun AJ. A multi-agent cellular automata system for visualizing simulated pedestrian activity. In: *Cellular Automata for Research and Industry*. Berlin: Springer-Verlag, 2000, pp. 29–36.
44. Kirchner A, Namazi A, Nishinari K and Schadschneider A. Role of conflicts in the floor field cellular automaton model for pedestrian dynamics. In: *Proceedings of the 2nd International Conference on Pedestrians and Evacuation Dynamics (PED)*, London, UK, 2003, pp. 51–62.
45. Wilensky U. NetLogo modeling environment, 2009. http://ccl.northwestern.edu/netlogo.
46. North MJ, Collier NT and Vos JR. Experiences creating three implementations of the repast agent modeling toolkit. *ACM Trans Model Comput Sim* 2006; 16: 1–25.
47. Luke S, Cioffi-Revilla C, Panait L, Sullivan K and Balan G. MASON: A multiagent simulation environment. *Simulation* 2005; 81: 517–527.
48. Kretz T and Schreckenberg M. Moore and more and symmetry. In: *Proceedings of the International Conference in Pedestrian and Evacuation Dynamics*, 2005, pp. 297–308.
49. Macal C and North M. Tutorial on agent-based modelling and simulation. *J Sim* 2010; 4: 151–162.
50. Almeida C, Batty M, Monteiro A, et al. Stochastic cellular automata modeling of urban land use dynamics: empirical development and estimation. *Comput Environ Urban Syst* 2003; 27: 481–509.
51. White R and Engelen G. High-resolution integrated modelling of the spatial dynamics of urban and regional systems. *Comput Environ Urban Syst* 2000; 24: 383–400.
52. Zhang X and Chang GL. Optimal control strategies for massive vehicular–pedestrian mixed flows in the evacuation zone. In: *The 89th Transportation Annual Meeting of the Transportation Research Board*, Washington, DC, January 2010.
53. Moreno N, Wang F and Marceau DJ. Implementation of a dynamic neighborhood in a land-use vector-based cellular automata model. *Comput Environ Urban Syst* 2009; 33: 44–54.
54. Blue VJ and Adler JL. Cellular automata microsimulation for modeling bi-directional pedestrian walkways. *Transport Res B* 2001; 35: 293–312.
55. Collier N. Repast HPC Manual, 2010. http://repast.sourceforge.net/docs/RepastHPCManual.pdf.
56. Moser D, Riener A, Zia K and Ferscha A. Comparing parallel simulation of social agents using Cilk and OpenCL. In: *15th International Symposium on Distributed Simulation and Real Time Applications (DS-RT 2011)*, Salford, UK. Los Alamitos, CA: IEEE Computer Society Press, 2011.
57. Sharpanskykh A and Zia K. Grouping behaviour in AmI-enabled crowd evacuation. In: *Proceedings of the 2nd*

International Symposium on Ambient Intelligence (ISAmI'10). Berlin: Springer-Verlag, 2011.
58. Ferscha A, Riener A, Sharpanskykh A and Zia K. Potential of social modelling in socio-technical systems. *Proc Comput Sci* 2011; 7: 235–237.
59. Zia K, Riener A, Ferscha A and Sharpanskykh A. Evacuation simulation based on cognitive decision making model in a socio-technical system. In: *15th International Symposium on Distributed Simulation and Real Time Applications (DS-RT 2011)*. Los Alamitos, CA: IEEE Computer Society Press, 2011.
60. Li X, Zhang X, Yeh A and Liu X. Parallel cellular automata for large-scale urban simulation using load-balancing techniques. *Intl J Geograph Inform Sci* 2010; 24: 803–820.
61. Collier NT and North MJ. Repast SC++: a platform for large-scale agent-based modeling. In: Dubitzky W, Kurowski K and Schott B (eds), *Large-Scale Computing Techniques for Complex System Simulations*. New York: Wiley, 2011.
62. Ferscha A and Zia K. LifeBelt: silent directional guidance for crowd evacuation. In: *Proceedings of the 13th International Symposium on Wearable Computers (ISWC09)*, Linz, Austria, 4–7 September 2009. Los Alamitos, CA: IEEE Computer Society Press, 2009.
63. Zia K, Ferscha A, Riener A, et al. Scenario based modeling for very large scale simulations. In: *14th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications (DS-RT 2010)*, Fairfax, VA. Los Alamitos, CA: IEEE Computer Society Press, 2010.
64. Sharpanskykh A and Zia K. Emotional decision making in large crowds. In: *Proceedings of the 10th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS'12)*, March 2012. Berlin: Springer-Verlag, 2012.
65. Murphy JT. *Computational Social Science and High Performance Computing: A Case Study of a Simple Model at Large Scales*. Argonne, IL: Argonne National Laboratory, 2011.
66. Message Passing Interface. http://www.dmoz.org/Computers/Parallel_Computing/Programming/Libraries/MPI/
67. Richmond P, Coakley S and Romano DM. A high performance agent based modelling framework on graphics card hardware with CUDA. In: *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2 (AAMAS '09)*. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2009, pp. 1125–1126.
68. Garcia V, Debreuve E and Barlaud M. Fast *k* nearest neighbor search using GPU. *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*.
69. Deissenberg C, van der Hoog S and Dawid H. EURACE: a massively parallel agent-based model of the European economy. *Appl Maths Comput* 2008; 204: 541–552.
70. Mishra S and Xie P. Interagent communication and synchronization support in the DaAgent mobile agent-based computing system. *IEEE Trans Parallel Distrib Syst* 2003; 14: 290–306.
71. Zia K, Riener A, Farrahi K and Ferscha A. A new opportunity to urban evacuation analysis: very large scale simulations of social agent systems in Repast HPC. In: *ACM/IEEE/SCS 26th Workshop on Principles of Advanced and Distributed Simulation (PADS)*. Piscataway, NJ: IEEE Press, 2012, pp. 233–242.

Author biographies

Kashif Zia is a PhD candidate in the Institute for Pervasive Computing, Johannes Kelper University in Linz, Austria. His research interests revolve around STSs, particularly focusing on crowd dynamics and simulation. His PhD work is related with large-scale ABM and analysis of STSs utilizing parallel and distributed simulation. He has an MSc in computer science and a B.E from the University of Engineering and Technology in Lahore, Pakistan.

Katayoun Farrahi is a postdoctoral research fellow at the Johannes Kepler University, Linz, Austria. Her research focuses on large-scale human behavior modeling and mining, with special interest in pervasive technologies, CSSs, mobile phone sensor data, and machine learning. She received her PhD in Computer Science from the Swiss Federal Institute of Technology (EPFL) Lausanne, and the Idiap Research Institute, Switzerland. She has spent time as an intern at MIT and is a recipient of the Google Anita Borg scholarship, and the Idiap research award.

Andreas Riener is a postdoctoral research fellow at the Institute of Pervasive Computing at the University of Linz (Austria). He has more than 60 refereed publications in the broader field of (implicit) human–computer interaction and context-aware computing, considered in both user study and simulation-based approaches. His core competence and current research focus is vital state recognition from embedded sensors, multimodal sensor and actuator systems, context-sensitive data processing/context-aware computing and implicit interaction influencing the driver–vehicle interaction loop. He is and has been engaged in several EU- and industrial-funded research projects and has been a reviewer for years for conferences and journals in the broader field of pervasive/ubiquitous computing. He is member of the Austrian Computer Society (OCG) and IEEE Member.

Alois Ferscha is full professor at the University of Linz where he heads the Excellence Initiative ''Pervasive Computing'', the Department of Pervasive Computing and the Research Studio Pervasive Computing Applications. He is focused on pervasive and ubiquitous computing, networked embedded systems, embedded software systems, wireless communication, multiuser cooperation, distributed interaction, and distributed interactive simulation. He leads/has lead international EU funded projects (EU FP7, FET: SAPERE, HC2, PANORAMA, SOCIONICAL, OPPORTUNITY; EU FP6, FET: BeyondTheHorizon, InterLink, CRUISE), but also national research projects (DISPLAYS, SPECTACLES, PowerSaver, WirelessCampus, MobiLearn), and holds tight cooperation with industrial stakeholders (SIEMENS Project FACT, IBM Project VRIO). He has served on editorial

boards of renowned international scientific journals (e.g. *Pervasive and Mobile Computing, Transactions of the Society for Computer Simulation*), on steering and programme committees of several conferences such as PERVASIVE, UMBICOMP, ISWC, WWW, PADS, DS-RT, SIGMETRICS, MASCOTS, MSWiM, MobiWac, TOOLS, Euro-Par, PNPM, ICS, etc. to name a few. His activities and recognition in the pervasive computing and wearable computing research communities is expressed by, e.g., his chairing of PERVASIVE 2004 (Programme Chair) and ISWC'09 (General Chair). He is an active consultant to the IST FET group within the Commission of the European Communities, Information Society and Media Directorate-General, and to the Austrian bm-wf and bm-vit. He is Austria's representative in IFIP TC-10 (International Federation for Information Processing, TC10 - Computer Systems Technology). As an invited researcher or guest professor he was visiting the Dipartimento di Informatica, Universita di Torino, Italy, at the Dipartimento di Informatica, Universita di Genoa, Italy, at the Computer Science Department, University of Maryland at College Park, College Park, Maryland, USA, and at the Department of Computer and Information Sciences, University of Oregon, Eugene, Oregon, USA.

Appendix A: Patches' floor field generation

Pseudo-code of floor field generation for patches.

```
1   if patch p structure_Type  == "walkable" and is not a "PoA"
2       for each neighboring patch n
3       // Moore's neighborhood of 8 cells
4           for each pair p <key, value> of DOMn
5           // key, value pairs of available PoAs
6               if key does not exists in DOMp
7               // if this PoA does not exists, update
8                   DOMp [key] = relative-angle to n
9                   HOPCp [key] = value(HOPCn [key]) + 1
10                  ROUTEp [key] = value(ROUTEn [key]) . ProcessID
```

Appendix B: Proceed_to_Next_Cell

Next-step behavior of an agent. The procedure is executed for each agent which is based on the orientation information that it has to make a cell-to-choose decision according to the rules shown in Figure 11 and updates agent-level variables accordingly.

```
1   heading (orientation)
2   p = patchAtHeadingAndDistance (heading, speed)
3   // take the patch at heading and distance
4       if (p → turtlesHere.size > 0) || p → structure_Type == 3)
5       // if patch already has some agents or is not walkable
6           if (scanNeighborinOrderRight(angle))
7           // different options are scanned in a typical order
8               heading(angle)
9               // setting heading to angle of first available option
10              move(speed)
11              // a step at a speed
12              reset_wait()
13              // wait variable is reset to 0
14              confidence(x)
15              // confidence is assigned based on how different the
16              // selected option is from original angle
17          else
18              inc_wait()
19              // if there is no available neighbor, wait is incremented by 1
20      else  // if patch is available
21          move(speed)
22          reset_wait()
23          confidence(100)
24          // maximum confidence sue to adoption of desired angle
```

Appendix C: Move_to_Nearest

Pseudo-code of Strategy 1 executed by non-AmI-assisted agents. The procedure Proceed_to_Next_Cell is given in Appendix B.

```
1   p = patchHere // patch underneath this agent
2   selected_exit  = min {key(HOPC_p)}
3   // trying to assign key of minimum value (least hop count) to selected_exit.
4   // If there is only 1 element in the collection, it would be selected.
5   // If there is no element in collection, this statement would not execute.
6   if (selected_exit == null)
7       inc_wait() // increments the agent's wait variable by 1
8   else
9       find pair  <key, value> in DOM_p
10      orientation(value(DOM_p [key])) // assigning value to agent variable
11      current_exit(selected_exit) // assigning value to agent variable
12      Proceed_to_Next_Cell() // procedure call
```

Appendix D: Move_to_Nearest_with_Follow

Pseudo-code of Strategy 3 executed by non-AmI-assisted agents. The procedure Proceed_to_Next_Cell is given in Appendix B. The procedure Move_to_Nearest is given in Appendix C.

```
1   if (current_exit != -1) // if there is an actionable exit
2       p = patchHere // patch underneath this agent
3       if (size(p→ DOM_p) >= 2)
4           set_OptimalPoA()
5           // if there is enough information, set the best option as current PoA,
6            // this statement would execute at every nth iteration (e.g. n = 50)
7           find pair  <key, value> in DOM_p where key = current_exit
8           orientation(value(DOM_p [key])) // assigning value to agent variable
9           Proceed_to_Next_Cell() // procedure call
10  else
11      if (size(p→ DOM_p) >= 2)
12          set_OptimalPoA()
13          inc_wait()
14      else
15          if (size(p→ DOM_p) == 1)
16              set_AvailablePoA() // setting it for one available exit
17              inc_wait()
18          else // if no information available for any PoA
19              inc_wait()
20
21  set_OptimalPoA()
22      current_weight = 999999; // a very high value for comparison
23      aggregate_route_density = 0; // initializing densities across the route
24      selected_PoA = -1; // default value for selected PoA
25      for each pair  <key, value> in ROUTE_p
26          current_process = key
27          current_route_sequence  = value
28          for each element e in current_route_sequence
29          // hiding full details here of string manipulations
30              find pair  <key, value> in DENSITIES_process where key = e
31              current_density = value(DENSITIES_process [key]
32              aggregate_route_density = aggregate_route_density + current_density
33          aggregate_route_density = aggregate_route_density / size(current_route_sequence)
34          find pair  <key, value> in DOM_p where key = current_process
35          temp_value = aggregate_route_density × value(DOM_p [key]
36          if (temp_value <= current_weight)
37              current_weight = temp_value
38              selected_PoA = current_process
39      current_exit(selected_PoA)
40
41  set_AvailablePoA()
42  current_exit(p → DOM_p .first[key]) // assigning first key as current exit
```

Appendix E: Move_to_Best_Available

Pseudo-code of Strategy 2 executed by AmI-assisted agents. The procedure Proceed_to_Next_Cell is given in Appendix B.

```
1   if there exits an AmI-assisted agent a in interation_range
2   // randomly pick one AmI agent in interaction range (e.g. r = 10 cells)
3       if (a→current_exit != -1) // if a has already made a decision
4       following(true) // start following
5       current_exit(a→current_exit) // making a's PoA as its own
6   if (following())
7       p = patchHere // patch underneath this agent
8       find pair  <key, value> in DOM_p where key = current_exit
9       orientation(value(DOM_p [key])) // assigning value to agent variable
10      Proceed_to_Next_Cell() // procedure call
11  else
12      Move_to_Nearest() // procedure call
```