Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 18 (2013) 1515 – 1524

Procedia Computer Science

International Conference on Computational Science, ICCS 2013

# HPC enhanced large urban area evacuation simulations with vision based autonomously navigating multi agents

M. L. L. Wijerathne[a,*], L. A. Melgar[a], M. Hori[a], T. Ichimura[a], S. Tanaka[a]

[a]*Earthquake Research Institute, the University of Tokyo, 1-1-1, Yayoi, Bunkyo-ku, Tokyo 113-0032, Japan*

**Abstract**

An evacuation simulation code based on Multi Agent Systems (MAS), with moderately complex agents in 2D grid environments, is developed. The main objective of this code is to estimate the effectiveness of the measures taken to smoothen and speedup the evacuation process of a large urban area, in time critical events like tsunami. A vision based autonomous navigation algorithm, which enables the agents to move through an urban environment and reach a far visible destination, is implemented. This simple algorithm enables a visitor agent to navigate through urban area and reach a destination which is several kilometers away. The navigation algorithm is verified comparing the simulated evacuation time and the paths taken by individual agents with those of theoretical. Further, a parallel computing extension is developed for studying mass evacuation of large areas; vision based autonomous navigation is computationally intensive. Several strategies like communication hiding, dynamic load balancing, etc. are implemented to attain high parallel scalability. Preliminary tests on the K-computer attained strong scalability above 94% at least up to 2048 CPU cores, with 2 million agents.


Selection and peer review under responsibility of the organizers of the 2013 International Conference on Computational Science

*Keywords:* mass evacuation; multi agent systems; autonomous navigation; parallel computing

## 1. Introduction

Tens of thousands of casualties of the tsunamis of 2004 Sumatra earthquake and 2011 Thohoku earthquake emphasized the necessity of mass evacuation studies in Earthquake engineering. Understanding evacuation dynamics is necessary to identify possible bottlenecks, find strategies to make the evacuation process smooth and reduce evacuation time, etc. Such analysis can save many lives at time critical events like tsunami. Out of a number of numerical models like cellular automata, social force model, etc., Multi Agent System (MAS) is a promising method for modeling heterogeneous and complex human behavior involved in mass evacuations [1, 2, 3, 4, 6]. There are number of studies related to tsunami triggered mass evacuation based on MAS. Most of these studies use simplified environment with simple agents [1, 2, 6]. In evacuation simulations, often the environment is modeled as a network consisting of 1D members which represent rooms, corridors, roads, open spaces, etc., each associated with a pedestrian capacity. One main advantage of these simplified models is that a modern personal computer is sufficient to reproduce the overall behavior of a medium size crowd.

*Corresponding author. Tel.: +81-3-5841-1766; fax: +81-3-5841-1766.
*E-mail address:* lalith@eri.u-tokyo.ac.jp.


Selection and peer review under responsibility of the organizers of the 2013 International Conference on Computational Science
doi:10.1016/j.procs.2013.05.319

Considering the fact that tsunami evacuation is directly concerned with tens to hundreds of thousands of human lives, more sophisticated modeling is necessary. Especially, complex MAS models are essential for real-time monitoring, predicting and guiding systems, which is realizable within next decades. Such system will: monitor each individual via mobile networks; forecast crowd behavior, considering various possibilities (i.e. Monte-Carlo simulations); choose the best possible routes for each individual based on the predictions; actively control the crowd by sending the instructions to reach a safe place avoiding congestions, damaged roads and hazards like fire. In addition, the system must analyze the human behavior to identify any blocked routes and take those into the account when predicting; changes to the environment is unknown right after an earthquake.

Working towards above mentioned real-time monitoring, predicting and guiding system, we implemented a moderately complex MAS with autonomously navigating agents in 2D urban environments. The main input for agents' autonomous navigation is the details of their visible neighborhood which they perceive with their sense of vision. Simulating millions of agents with vision based autonomous navigation is computationally intensive. To meet this high computational demand, we developed a scalable parallel extension with the Message Passing Interface (MPI)[7].

The contents of the rest of this paper are organized as follows. Section two gives a brief introduction to the developed MAS system for mass evacuation simulations. Details of the vision based autonomous navigation are given in the third section while its verification is presented in the fourth section. The fifth section presents the implemented parallel computing strategies, briefly. Finally, some concluding remarks are presented in the last section.

## 2. A brief overview of the multi agent system for evacuation simulations

Our multi agent system consists of two main elements; agents and the environment. The agents are the software counterparts which mimic the relevant behavior of people in a mass evacuation event. The agents interact with each other and the environment, and take spontaneous actions within the constraints of the environment. In this section, a brief overview of these two elements is given, focusing on the basic requirements of these two elements and implementations.

### *2.1. Environment*

The environment is modeled as a high resolution grid, with the maximum grid interval of $1m$. The choice of grid model for the environment, instead of the widely used vector model, is due to three main reasons; ease of modifying the environment, identifying boundary of visibility and parallel computing.

Vector models are widely used due to their versatility and compactness. In vector models, elements like buildings, roads, etc. are modeled as closed polygons, and assembled into a graph data structure, according to their connectivity. Further, detailed information like number of lanes, type of the building, number of occupants, etc. can be embedded in polygons. Route planning is fast and smooth edges of elements make the agent navigation easier. On the other hand, the grid model does not have any of these advantages: buildings, roads, etc. are no longer identifiable single entities; route planning in grid environment is time consuming; agent navigation in saw-toothed grid environment poses difficulties.

However, some advantages of grid environment make it a good candidate, provided the problem of route planning can be solved. The target problem of tsunami triggered evacuation requires real-time modification of the environment to incorporate the earthquake induced damages like collapse of buildings, damages to roads, etc. Such localized modifications can be efficiently made in grid environments. Also, identifying the boundary of the visibility of an agent is simple and fast in grid environments; working with concave polygons of vector models is complex and time consuming. Further, the vector model poses several difficulties in parallel computing; domain partitioning, dynamic load balancing and communicating graphs of vector data are complex, and have lower scalability compared to the grid data. When the region being modeled is several tens or hundreds of square kilometers, grid environment is much simpler to manage in parallel computing. The choice of grid environment is due to these advantages; how to accelerate route panning in grid environment is explained in the latter part of this section.

### 2.2. Agents

Complex models of agents have to be considered since the target mass evacuation problems concern with lives of large number of people. As mentioned in the introduction, though widely used simplified models can reproduce the overall behavior, complex models are necessary for real-time monitoring, predicting and guiding systems. It is not necessary to implement the most complex models of humans with the cutting edge artificial intelligence. Models of medium complexity with the human behaviors that have substantial infulence on the evacuation process are sufficient. The main objective of the current work is to implement vision based autonomous navigation, with or without pre-known route to a destination in an urban environment. This is an essential ability which the human counterpart possess.

Agents are implemented using object oriented paradigm with C++. An agent has two sets of data pertaining to its inherent properties, like maximum speed, age, maximum sight distance, etc., and its memory holding the information like past experiences, the routes it has taken, destination, etc. An agent has several major functionalities; *think*, *see* and *move*. Real human crowds are highly heterogeneous both in abilities and functionalities. To mimic a crowd in a given area, agents with wide range of physical abilities are generated according to a given distribution. Further, the agents are specialized by including more information and functionalities to represent relevant officials like police officers, fire fighters, volunteers, etc.

There are two main types of agents relevant to the topic of autonomous navigation; visitors and non-visitors. A visitor agent does not possess any information about the environment, except what is to be seen within its visibility, what is learnt while moving around, and the direction to a far visible destination like high ground, if any. All except visitor agents (e.g. residents, police officers, fire-fighters, volunteers, etc.) have the information of their neighborhood and have the ability to find route to a desired destination.

In this paper, all types of agents except the visitor agents are called *non-visitor agents*. A non-visitor agent has the knowledge of a limited region of few square kilometers. Once a non-visitor agent moves out of their known domain, they automatically become visitors.

#### 2.2.1. Radar vision

Vision is the main mean of perceiving the environment for human beings, hence vision is an important functionality for the agents. Just as the human counterpart, an agent must be able to identify the features in his visible neighborhood. We implemented radar based vision to make the agents identify the features like roads, obstacles, etc., and neighboring agents, within their sight distances.

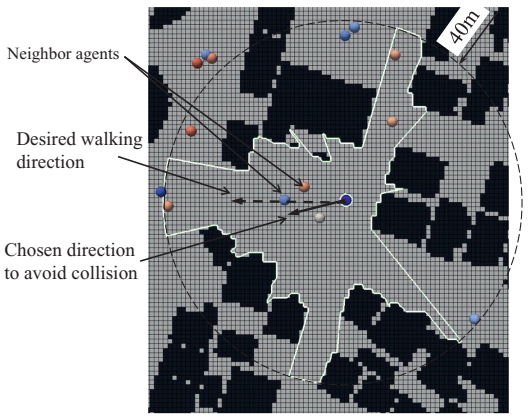



Fig. 1: Navigation based on radar vision. Shown in white is the boundary of visibility.

The vision functionality scans the environment like radar and identifies the boundary of an agent's visibility, in high resolution. The interval of radar scanning, $d\theta$, is set such that $L \times d\theta = 0.5 \times \delta l$, where $L$ is the sight distance and $\delta l$ is the length of grid cell; approximately 630 rays for $50m$ sight distance. Figure 1 shows the identified boundary of visibility and neighbor agents, by an agent with $40m$ sight distance. Analyzing the boundary of visibility an agent identifies the available paths and moves towards the chosen direction, avoiding collision.

Details of identification of features like available paths by analyzing the boundary of visibility, and a vision based autonomous navigation algorithm that enables the agents to navigate in complex urban environments are explained in the next section.

Radar vision requires ray tracing, which involves large amount of floating point operations. In order to reduce the amount of computations involved, several radar templates are prepared. For the current study, 16 sets of radar templates with their origins arranged in a $4 \times 4$ grid of $20cm$ spacing are used. Each radar templates contains the information of cells through which rays pass and the intersecting points. An agent chooses the best radar template according to their location in a grid cell. Data of each radar template is stored in a circular container so that agents can easily identify the boundary of visibility in their front and back. The maximum of sight distance of all agents is used when preparing the radar templates.

### *2.3. Route planning*

Real-time route finding is an essential for the target problem; if an agent finds the route its following is blocked by a collapsed building, etc., it should be able to find the next closest route to a nearby high ground or evacuation center. A major problem with the grid environments is the complexities of route planning algorithms. Real-time route finding in large grids is extremely slow and drastically reduces parallel scalability. This problem can be solved by using a vector map for route planning. Vector map of available routes in a grid model can be automatically constructed by abstracting the pixel skeletons obtained with thinning algorithms[8]. Figure 2a and 2b show a pixel skeleton obtained with thinning algorithms and the corresponding vector map, respectively.

(a) a pixel skeleton obtained by thinning. (b) vector map of available routes.

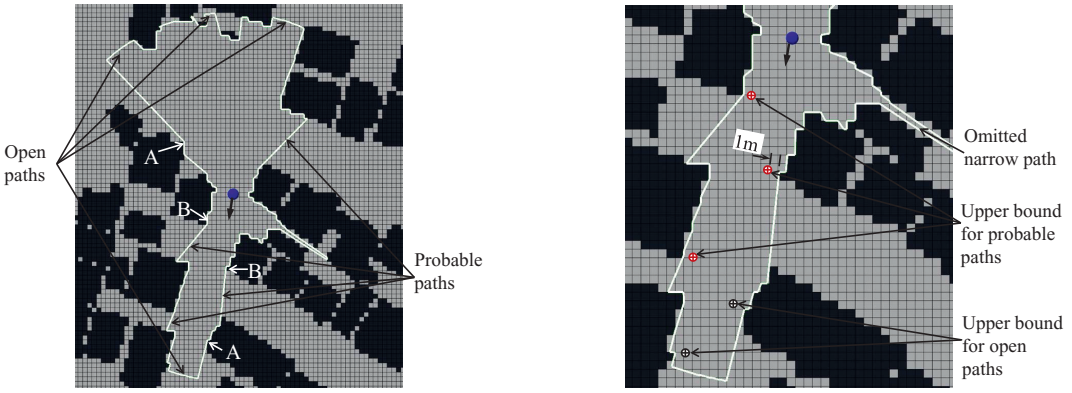
Fig. 2: Automated route map generation from grid data. Blue dots in (b) are the nodes defining the vector map.

## 3. Autonomous navigation in urban environments

For the target problem of tsunami induced large area evacuations, the agents must have good skills to autonomously navigate through a complex urban environment, which is modeled as a grid. At least the agents should have the ability of navigating to a far visible high ground or a tall building, even if an agent represents a visitor; an essential skill which any human counterpart possesses. Details of an vision based autonomous navigation algorithm are presented in this section while some verification tests are presented in next section.

### *3.1. Identification of features of the visible neighborhood*

For autonomous navigation, an agent must be able to identify the features of its visible neighborhood and choose the correct direction to go. Analyzing the boundary of visibility, agents identify two types of available paths; *open-paths* and *probable-paths*. The regions visible up to an agents sight distance are categorized as open-paths. The length of some pairs of neighboring radar rays have large differences. These sudden jumps are generated by the presence of empty spaces right behind a corner of a building, like roads, parking areas, etc. The sudden jumps larger than $5m$ are categorized as probable paths (see Fig. 3a).




(a) Available open and probable paths.

(b) Target points for preventing obstruction of vision.

Fig. 3: Identification of available paths for the agent shown in blue.

Correct identification of probable-paths is necessary to make an agent follow a road with sharp bends or junctions. Some care has to be taken when identifying the available probable-paths since, in grid environments, saw-toothed boundary of visibility often produce sudden jumps similar to those produce by roads at a junction. Moving towards these false signals not only produces longer evacuation time compared to that of the human counterparts, but also extra efforts are necessary to bring the agents back to their original route. In order to eliminate these false signals, only the jumps greater than certain number of cells are considered; greater than $5m$ jumps are used in this study.

When an agent moves towards a probable or open path, care has to be taken not to let it move too close to any corner which gives rise to sudden jumps (e.g. points shown with A and B in Fig. 3a). Moving too close to a corner of an obstacle significantly lowers an agents field of vision, often forcing it to turn backwards. In order to prevent this problem, associated with each open or probable paths are a set of points defining the closest they can move towards the corresponding corners of obstacles. Figure 3b shows some of the points defining the closest an agent can move to corners giving rise to sudden jumps in boundary of visibility. It is found that setting these upper bound points around $0.5m \sim 1m$ away from the obstacle corners solves the problem of view obstruction. Considering the fact that half shoulder width of a person is around $0.4m$, this restriction is not unrealistic.

### *3.2. Navigation of non-visitor agents*

Non-visitor agents have the ability to plan a route to a desired destination in their known environment. The routes are defined by a few landmark points so as to minimize the amount of agents' data size and eventually the message sizes in parallel computing. A digraph (directed graph) of all the avaialable routes, which is mentioned in the section 2.3, are used for path planning.

To move between the landmark points defining a route, which can be a few to several hundred meters apart, a non-visitor agent uses a simple navigation algorithm. As mentioned in the section 3.1, an agents identifies, available open/probable paths in its front and back. Being aware of their living or working environment, a non-visitor agent considers all the available open or probable paths in his front and back with equal priority and choose the closest open or probable path towards the next landmark point of its pre-planned route.

### *3.3. Navigation of visitor agents*

Unlike the non-visitor agents, the visitor agents neither have the ability to find paths nor well-defined destinations. Like the human visitors, in most evacuation scenarios, the visitor agents have to either get information from a non-visitor agent, map, mass-media, etc. or follow a group of residents.

However, in the case of tsunami evacuation, safe evacuation places like tall buildings and high grounds are abundant. A human visitor has the ability navigate through an urban environment and reach a far visible safe place like a tall building or a high ground. Being tsunami evacuation is one target application, it is essential for the visitor agents have this skill of human visitors to autonomously navigate to far visible destinations.

A human visitor's ability to make an intelligent selection of a path out of all the available in his limited visibility and discover a route to far visible destination must be sophisticated and challenging to be modeled. Though it does not include all the capabilities of a real human visitor, a certain order of priority for choosing the best out of all the visible paths found to be effective in mimicking real human visitors' navigation ability. The pseudo code given in Algorithm 1 shows the proposed order of priority for selecting open/probable paths in the front and back of visitor agent. According to this algorithm, visitor agents have high priority for the open paths in the front. On the contrary, non-visitor agents have equal priority for any available paths. The visitor agents' algorithm is not too artificial in the sense that even human visitors tend to have similar preferences: in an unknown environment, people prefer taking roads in their front and prefer taking roads with longest visible stretch instead of those hidden behinds bends or obstacles.

**Algorithm 1:** Algorithm for visitor agents' path selection.

**input** : vectors of all the avaiable paths; *front/back_open_paths*, *front/back_probable_paths*, destinations within sight distance; *front/back_exits*, direction of the destination and current moving direction;$\mathbf{v}_d$, $\mathbf{v}_c$

**output**: Next moving direction: $\mathbf{v}_n$

$\mathbf{v}_n \leftarrow \mathbf{v}_c$

**if** *front_exits*.`size()` $> 0$ **then** $\mathbf{v}_n \leftarrow$ `ClosestExit(`*front_exits*`);` `/* choose closest exit */`
**else if** *back_exits*.`size()` $> 0$ **then** $\mathbf{v}_n \leftarrow$ `ClosestExit(`*back_exits*`);`
**else if** `BestPathForVisitor(`*front_open_paths, front_probable_paths,* $\mathbf{v}_n$, $\mathbf{v}_d$`)` **then** ;
**else if** `BestPathForVisitor(`*back_open_paths, back_probable_paths,*$\mathbf{v}_n$, $\mathbf{v}_d$`)` **then** ;
**else** Freeze the agent; `/* agent has no paths */`

**return** $\mathbf{v}_n$

**Function** BestPathForVisitor

**input** : Open and probable paths in the font or back; *open_paths*, *probable_paths*, $\mathbf{v}_n$, $\mathbf{v}_d$
**output**: whether a direction found and the selected direction, $\mathbf{v}_n$

`/* BestProbablePath() and BestOpenPath() find the probable and open path closest to the direction of destination, avoiding narrow paths */`

*dot_max* = *dot_max_p* $\leftarrow$ *-1* ; `/* max(`$\mathbf{v}_d$`.`$\mathbf{v}$`) of all the directions,` $\mathbf{v}$`, considered */`
*jump* $\leftarrow$ *0* ; `/* size of the jump which defines a probable path */`

**if** *open_paths*.`size()` $> 0$ **and** *probable_paths*.`size()` $> 0$ **then**
  $\mathbf{v}_n \leftarrow$ `BestOpenPath(`*open_exits,* $\mathbf{v}_d$*, dot_max*`);`
  $\mathbf{v}_p \leftarrow$ `BestProbablePath(`*probable_exits,* $\mathbf{v}_d$*, dot_max_p, jump*`);`
  **if** *dot_max_p* $>$ *dot_max* **and** *jump* $>$ *30m* **then** $\mathbf{v}_n \leftarrow \mathbf{v}_p$
**else if** *open_path*.`size()` $> 0$ **then** $\mathbf{v}_n \leftarrow$ `BestOpenPath(`*open_paths,* $\mathbf{v}_d$`)` ;
**else if** *probable_path*.`size()` $> 0$ **then** $\mathbf{v}_n \leftarrow$ `BestProbablePath(`*probable_paths,* $\mathbf{v}_d$`);`
**else return** *false*

**return** *true*

Though not perfect, this simple algorithm makes visitor agents to navigate through urban environments and reach destinations which are several kilometers away. As shown in the next section, most of the routes discovered with this algorithm are acceptable choices even for human visitors. Algorithm 1 is found to produce best results when the sight distances of agents are almost equal to the distance between two lanes in the considered urban environment. As an example, sight distances between $40m$ to $50m$ made almost all the visitor agents to reach their destination in the environment used in the next section (see Fig. 4).

Once the open or probable path to be taken is decided, an agent checks for possible collisions with neighboring

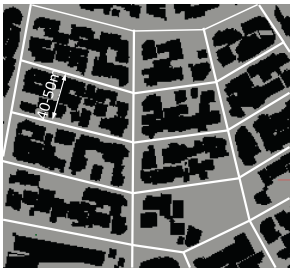



Fig. 4: Distance between parallel lanes in a residential area is found to be a good choice for the sight distance of visitor agents.

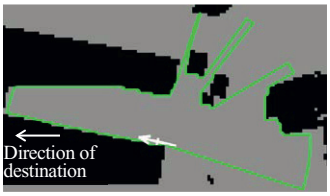



Fig. 5: An agent moving to a well deeper than it's sight distance may have a low chance of getting out of it, depending on the arrangement of the paths avaiable at the entrance.

agents and slightly adjusts the moving direction to avoid collision (see Fig. 1), sticking to the selected open or probable path.

### *3.4. Known limitations of visitor agent's navigation*

There is one known limitation of visitor agents' navigation; non-visitors have no known limitations. A visitor agent might get lost and start to move in a loop, if enters a region where there are no identifiable opening towards the destination, and depth of the region longer than the sight distance. Under such circumstances Algorithm 1 makes the agent to turn back and exit such deep wells. However, some complex setting at the mouth of the deep well may make the agent to re-enter the deep well, making to move in a loop eternally. Figure 5 shows an agent entering such a deep well with complex setting at its entrance. A simple solution to this problem is temporarily increasing such trapped agents' sight distance to prevent reentering the deep well.

## 4. Verification of autonomous navigation in urban environments

Verification and validation of the proposed radar vision based autonomous navigation is essential since the simulated are people, with the aim of reducing casualties. For verification, it has to be tested whether the proposed algorithm can reproduce analytically expected answer of a well-posed problem. Validation requires testing whether the numerically obtained solution satisfactorily matches with the observed behavior of a real evacuation.

In this paper, we consider only the verification by comparing the overall and agent-wise results with analytically obtained shortest paths and evacuation times. The problem for the verification is chosen so as to evaluate the proposed algorithms in navigating agents to far visible destinations.

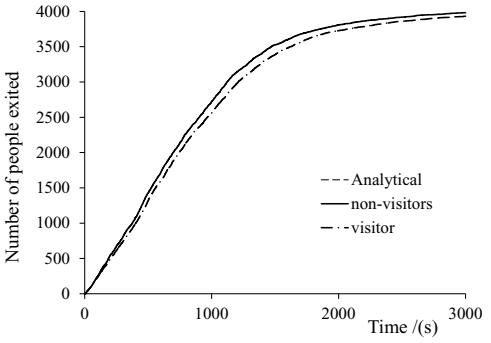
Collision avoidance or other interactions among agents are not considered in this verification test since the objective is testing the navigation algorithms. The problem becomes ill-posed when the collision avoidance is included; the total evacuation time and the routes taken by individual agents are highly dependant on the crowd density and each individual's response to dense crowds. Surely, it is important to study to which extent the agents behavior deviates when the collision avoidance is included and how it varies with the crowd density. However, it is out of the scope of the current study.

### *4.1. Problem setting*

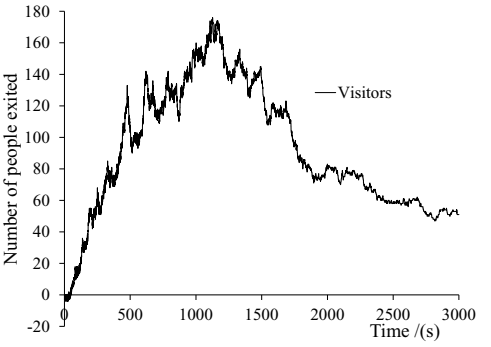
For the verification tests, 4000 agents in $1.6 \times 1.2 km^2$ regions of Kochi city environment is considered; as shown in Fig. 8 the selected area has many buildings and a few open areas. Two cases, each with either non-visitor agents or visitor agents, are considered. The agents are expected to move to a high ground, which is visible from anywhere in the selected domain, located at the left edge of the domain. All the agents are assumed to have the same sight distance of $50m$. The non-visitor agents follow the shortest routes to the left edge of the boundary from their starting location. On the other hand, visitor agents navigate only based on the information within their sight distance and the direction to the visible high ground. Apart from these, both the problems have the same settings.

### 4.2. Overall behavior

Figure 6a shows the graphs of cumulative number of agents exited versus time. The near perfect matching of non-visitor graph with that of the theoretical verifies non-visitor agents navigation algorithm; even if they rely on their vision to navigate between few landmark points defining their routes, they can reach the destinations at the theoretically expected time. As seen in the Fig. 6a, visitor agents also have surprisingly good navigation abilities; visitors' graph is very close to the theoretical. As show in the Fig 6b, the maximum difference is around 170 agents, at 1100 seconds. The authors think such deviation from the ideal is acceptable considering the complexity and size of the domain and the presence of slow moving agents with the minimum speed of $0.5ms^{-1}$.




(a) Cumulative number of agents exited




(b) Difference between the cumulative number of theoritical and vistor agents exited

Fig. 6: Time history of cumulative number of agents exited.

### 4.3. Trajectories of individual agents

In addition to the overall behavior, investigating the paths taken by each individual is important to make sure that none of the agents has taken paths which are unrealistic choices for the human counterpart. The possibility of non-visitor agents to take an unrealistic path is very low since they are guided with intermediate landmark points. Analysis of simulation results confirmed that all the non-visitor agents have followed the pre-planned shortest routes with minor deviations.

As one can easily guess, visitor agents have a high possibility of taking unrealistic paths. However, detailed comparisons indicate that none of the visitor agents have taken unrealistic path (see Fig. 7). Figure 8 shows the trajectories of randomly selected visitor and non-visitor agents; each pair has identical starting point with identical properties. Though the paths taken by visitor agents are not the shortest, those are realistic choice even for a human and the time differences between most of the paths are small.

The comparison of overall behavior and individual paths indicates that the non-visitor agents have the ideally expected navigation behavior while the visitor agents have acceptable navigation behavior. Though further improvements to the navigation algorithms are possible, the moderately complex current algorithms are sufficient for the long term objective of real-time observation, prediction and guiding system.

## 5. Parallel computing enhancements

Evacuation simulation with millions of people in large urban areas requires efficient utilization of parallel computing resources. The existing MAS based pedestrian simulation studies have reported low scalability which is limited to a few tens of CPU cores. With several strategies for hiding and minimizing communications, we attained near linear scalability at least up to a few thousands of CPU cores, with several millions of agents[7]. The rest of this section provides a brief description of some difficulties in parallelization and main strategies used to attain high parallel scalability.

The major steps in parallelizing the multi agent code are the same as in other particle type simulations, like SPH. However, compared to other particle type methods, parallelization of the multi agent code involves several additional difficulties, some of which are listed below.

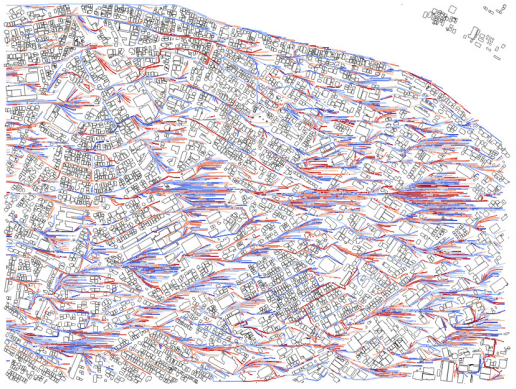


Fig. 7: Routes taken by the visitor agents.

Fig. 8: Comaprison of routes taken by some of the visitor and non-visitor agents with same properties.

1. Agents involve large amount of data some of which are dynamically growing and stored in complex data structures like graph.
2. Objects of different types of agents, like officials, residents, etc., have to be stored in non-contiguous locations in different vectors.
3. Require maintaining a wide ghost layer of thickness at least equal to the maximum visibility distance of an agent. In a dense urban area, ghost layer of $50m$ may contain a large number of people.
4. Amount of computations depends on the type and surrounding conditions of an agent.

### *5.1. Strategies for enhancing scalability*

As mentioned, the basic parallelization strategies are the same as in other particle type simulations; the domain is decomposed such that each CPU has equal work loads, ghost or overlapping layer is maintained and updated with the neighboring CPUs to preserve the continuity and domain is repartitioned when the agent movements bring significant load imbalance. For domain decomposition, kd-tree is used since its simple geometry makes it easy to manage movement of agents between CPUs. In addition to these common strategies, the following major strategies are used to deal with some of above mentioned additional difficulties.

#### *5.1.1. Hiding communications and minimizing volume of data exchanged*

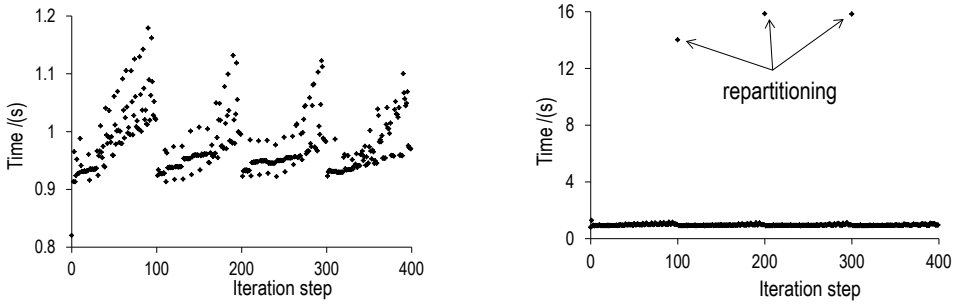
The main strategy of gaining scalability is hiding the communication overhead by first processing the agents to be communicated and posting the message. However, the presence of dynamically growing data makes it difficult to eliminate the communication overhead completely. Since exchanging all the dynamic data requires at least two messages and packing and unpacking of large amount of data, only the new contents added at a time step are exchanged when updating ghost boundaries. This, still, requires explicit packing and unpacking making it impossible to hide the communication overhead completely. However, a significant portion of communication overhead is hidden, making the code to attain high parallel scalability.

#### *5.1.2. Minimizing data exchange in repartitioning*

Repartitioning for dynamic load balancing is an expensive step since agents have a large amount of data. With 2D-tree partitioning, most of the agents remain in the same CPU even after repartitioning, unless *MPI_Dist_graph_create()* maps a partition to a different CPU. The repartitioning algorithm detects whether a partition is assigned to the same CPU and exchange only the newly assign agents. This drastically reduces the communication overhead involved with repartitioning, effectively lowering any performance degeneration due to repartitioning.

### *5.2. Scalability*

Preliminary scalability tests in the K-computer, Kobe, produced 94% strong scalability with 2048 CPU cores; strong scalability is defined as $\frac{(\frac{T_m}{T_n})}{(\frac{n}{m})}$, where $T_k$ is the time with $k$ number of CPU cores and $n \geq 2m$. A problem




(a) excluding the repartitioning time (b) including the repartitioning time

Fig. 9: History of run time with 2048 CPU cores.

with 2 million agents in $18km^2$ urban area is considered in these tests. Figures 9a and 9b shows the runtime for 400 iterations, excluding and including the repartitioning time. Figure 9a clearly show the significant performance gain due to repartitioning. As is seen in Fig. 9b the major bottleneck in the current code is repartitioning, which is still a serial code.

## 6. Concluding remarks

We implemented radar vision based autonomous navigation algorithms for both the visitor and non-visitor agents. The verification test shows that the proposed navigation algorithms enable both the visitor and non-visitor agents to navigate successfully through urban environments. Although non-visitor agents do not take the optimal paths, their paths are found to be acceptable choices even for human counterparts. In order to meet the heavy computational demands of simulating millions of agents with vision based navigations, we implemented a *MPI* based parallel extension. With a number of strategies, near linear scalability is attained up to 2048 CPU cores in the K computer.

## Acknowledgements

This work was supported by JSPS KAKENHI Grant Number 24760359. Parts of the results are obtained with early access to the K computer at the RIKEN Advanced Institute for Computational Science.

## References

[1] S. Gwynne, E. Galea, M. Owen, J. Lawrence, and L. Filippiddis, *A Review of the MethodologiesUsed in Evacuation Modeling. Fire and Materials* , 1999, 23, pp. 383–388.

[2] D. Helbing, I. J. Farkas, P. Molnr, T. Vicsek, *Simulation of pedestrian crowds in normal and evacuation situations*, Pedestrian and Evacuation Dynamics, Springer, 2002, pp. 21–58.

[3] M. Hori, H. Miyajima, Y. Inukai and K. Oguni, *Agent simulation for prediction of post-earthquake mass evacuation*, Journal of Japan Society of Civil Engineers, Ser. A, 2008, vol. 64, pp. 1017–1036.

[4] R. Dulam, M. Lalith, M. Hori, T. Ichimura and S. Tanaka, Development of HPC enhanced multi agent simulation code for tsunami evacuation, Journal of Applied Mechanics, JSCE, 2012, vol. 15.

[5] A. Nash, K. Daniel, S. Koenig and A. Felner, *Theta*: any angle path planning on grids*, Proceedings of the AAAI Conference on Artificial Intelligence, 2007, pp. 1177–1183.

[6] Takashi Saito and Hiroshi Kagami, *Simulation of evacuation behavior from tsunami untilizing multi agent simulation*, Proceedings ot the 13-th World Conf. on earthquake Engineering, Vancouver, 2004.

[7] Maddegedara Lalith, Hori Muneo and Ichimura Tsuyoshi, *Parallel scalability enhancements of seismic response and evacuation simulations of IES*, Proceedings of the 10th International Meeting on High-Performance Computing for Computational Science, Kobe 2012.

[8] Beeson, Jong, and Kuipers, *Towards autonomous topological place detection using the Extended Voronoi Graph*. IEEE International Conference on Robotics and Automation (ICRA-05), 2005, pp. 4373 – 4379.