Available online at www.sciencedirect.com

ScienceDirect

Transportation Research Procedia 00 (2018) 000–000

Transportation Research Procedia

www.elsevier.com/locate/procedia

International Symposium of Transport Simulation (ISTS'18) and the International Workshop on Traffic Data Collection and its Standardization (IWTDCS'18)

Scalable HPC enhanced agent based system for simulating mixed mode evacuation of large urban areas

Lalith Wijerathne[a,*], Wasuwat Petprakob[b], Leonel Aguilar[c], Muneo Hori[a], Tsuyoshi Ichmura[a]

[a]*Earthquake Research Institute, The University of Tokyo, 1-1-1, Yayoi, Bunkyo, Tokyo, 113-0032, Japan*
[b]*Department of Civil Engineering, The University of Tokyo, 7-3-1, Hongo, Bunkyo, Tokyo, 113-8654, Japan*
[c]*Department of Humanities, Social and Political Science, ETH, Clausiusstrasse 50, 8092 Zurich, Switzerland*

Abstract

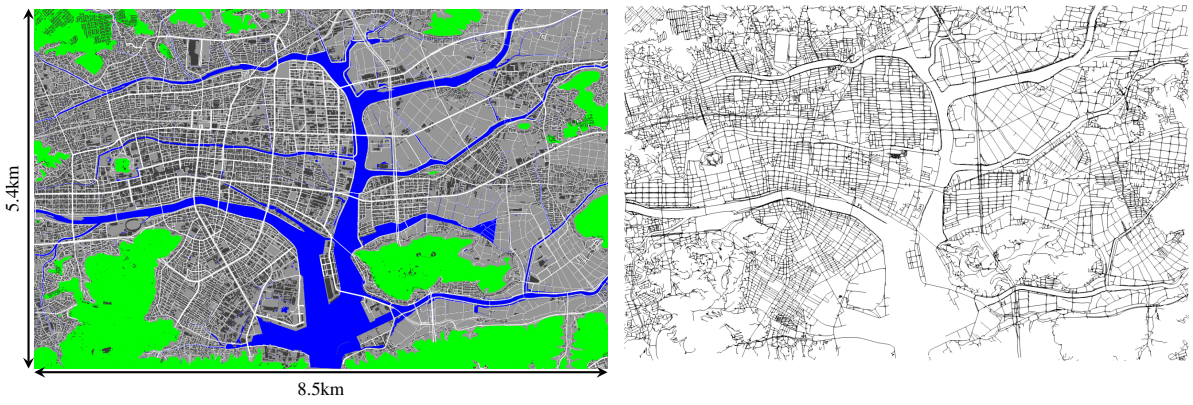
This short paper presents an HPC enhanced Agent Based Model (ABM) developed with the aim of quantitatively estimating the strategies for accelerating emergency mass evacuations, like tsunami evacuation. In order to facilitate inclusion of various influencing factors, such as localized congestion, multi-modes, pedestrian vehicle interactions, fallen debris from damaged buildings, visibility, etc., which demand detailed models, the developed system includes a 1m×1m resolution model of environment, and agents capable of perceiving and autonomously interact with this high resolution environment and visible agents. In order to meet the computational demand of large scale simulations with complex agents, a scalable high performance extension was implemented. Short introductions to the agent based model and the HPC extension are presented in this paper. In order to demonstrate the scalability of the system, both in problem size and computational capability, a 588 km^2 region with 10 million agents is simulated in K computer. It is demonstrated that the system has high strong scalability up to 2048 computing nodes, which is equivalent to 16,384 CPU cores.

Keywords: Agent based modeling; tsunami evacuation; high resolution environment; high performance computing

1. Introduction

Anticipating major earthquakes in Tokai, Tonankai and Nankai regions of Japan are predicted to bring major tsunami hazards to the neighboring coastal regions. While short arrival time and high tsunami heights are major threats in many areas, different regions have various other problems, like long distance to the evacuation areas, insufficient

* Corresponding author. Tel.: +81-03-5841-5699 ; fax: +81-03-5841-5699.
E-mail address: lalith@eri.u-tokyo.ac.jp



(a) 1m×1m resolution grid. (b) Topological graph

Figure 1: Grid and topological graph of Kochi city environment. Shown in green are the regions above 30m elevation.

capacity of evacuation shelters, large elderly population, narrow roads, car usage, car-pedestrian interaction, etc. Numerical simulations are invaluable to the disaster mitigation authorities in seeking various strategies to address these problems, accelerating evacuation process, identifying unforeseen problems, etc. Motivated by the ease of use, less effort in developing, and limited computational resources available, simple queue models with one dimensional networks are widely used in mass evacuation studies. While such simplified models provide useful information, they have strong limitations. Related to mass evacuation in large congested urban cities, there are many scenarios which require the use of accurate high resolution model of the environment and complex models of individuals: people with different behaviors and responsibilities, multiple evacuation modes and their interaction, dynamic changes like a progressive inundation, visibility, etc. With the aim of simulating such demanding scenarios, we developed an Agent Based Model (ABM) which includes a high resolution model of the environment and complex agents which are capable of perceiving it (Melgar et al. (2014), Wijerathne et al. (2013)). In order to meet the high computational demand of complex agents in high resolution environment, we implemented a scalable parallel computing extension so that millions of agents in several hundreds of square kilometers can be simulated by utilizing high performance computing facilities (Maddegedara et al. (2013), Aguilar et al. (2017)). In this short paper, we present the details of the implemented agent based model and its parallel computing extension.

2. Agent based model

The developed system is a time step driven agent based model which consists of a hybrid environment and cognitive agents. In order to model heterogeneous evacuating crowds, agents are assigned with different physical and behavioral parameters, different roles (i.e., evacuee, volunteer, officials, etc.), different level of access to information. Most systems model mass evacuation as agent flows on 1D networks and set the agents' speeds according to the average density in each link and suitable fundamental diagrams. On the contrary, the parameters of the autonomous agents of the developed system are tuned such that agents' individual actions reproduce characteristics like fundamental diagrams as an emergent behavior (Melgar et al. (2013), Melgar et al. (2014)).

2.1. Hybrid environment

With the aim of modeling evacuation process in microscopic details, we model the environment as a hybrid of a high resolution grid and a topological graph, see Fig. 1. The traversable 2D domain is modeled with a 1m×1m resolution grid, and updated at desired time intervals to model dynamic changes like tsunami inundation, fallen debris due to the earthquake, etc. The grid updates are generated based on the physics based simulators or observations. Connectivity between the traversable spaces in the grid is abstracted by a topological graph. The graph contains topological connectivity of an ordinary day (i.e., not updated according to dynamic changes of the grid). The graph

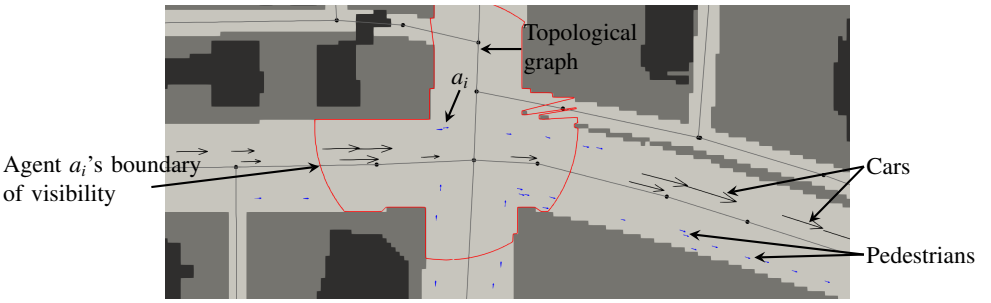


Figure 2: Snapshot of agents' movements at a junction. Blue and black arrows indicate instantaneous velocities of pedestrians and cars, respectively. Pedestrian agents walk along the edges, if the road can accommodate vehicles.

serves as the base map of the agents' memory supporting agents' decision making functions. Scanning the grid in high resolution, agents recognize the features of their visible surrounding in the grid and any mismatches between the information in graph and grid. These mismatches, like a blocked road, and other experiences, like getting support from an official, are stored with reference to the graph so that experiences and new information can be easily taken into account in their decision making processes. The graph is equipped with a number of functions to find paths with various requirements (e.g. minimize the use of narrow roads in night time evacuation scenarios). The environment evolves in time, and its state at time t is denoted by E^t.

The grid and graph are automatically generated from GIS data. Traversable spaces can be generated from any of the following three data: center line and width of roads, road boundaries, or outline of the buildings. While generating the topological graph is straightforward with road center lines, we use thinning algorithm (Beeson et al. (2005)) to generate graph when the given input data is grid environment. The hybrid system is scalable to accommodate several hundreds of square kilometers in $1\times1\text{m}^2$ resolution. An accurate model of the environment, including data like buildings, contours, water bodies, etc., can be generated in a short time, when GIS data is available. As an example, the model of $8.5\times5.4\text{km}^2$ region shown in Fig. 1 was generated in 90 seconds.

2.2. Agents

An agent, a_i , which mimics the behavior of a person during an emergency mass evacuation, consists of its state at time t, s_i^t, and a set of constituent functions, g^i, defines its role. s_i^t consists of agent's private data (e.g. destination, sight distance, decision, etc.) and information which are deducible by neighboring agents (e.g. speed, moving direction, etc.). The surrounding, with which an agent interacts at time t, consists of animate and inanimate parts. Inanimate surrounding consists of a_i's visible surrounding, $E_i^{visible,t} \subset E^t$, and any remote sources of information, $E_i^{remote,t} \subset E^t$, like the current travel time of the road segments along its current path. Animate surrounding consists of agents visible to a_i's, $A_i^{visible,t} \subset A$, where the set of agents $A = \{a_i | i = 1, 2, \ldots\}$. Figure 2 shows the boundary of visibility of an agent a_i with 30m sight distance; the agents inside this visibility boundary are a_i's $A_i^{visible,t}$.

The discrete time evolution of the agent based system, $A \cup E^t$, is governed by the functions updating the environment, and functions defining individual agent's role. Evolution of environment is modeled by updating with suitable functions, λ_j's , to mimic events like tsunami inundation, damage propagation, etc.; $E^{t+\Delta t} = \lambda_1 \circ \lambda_2 \ldots \circ \lambda_m(E^t)$. Autonomous actions of an agent a_i are defined by f_i, which updates the agent's state as $s_i^{t+\Delta t} = f_i(s_i^t, A_i^{visible,t}, E_i^{visible,t}, E_i^{remote,t})$. f_i is composed of suitable subset of the available set of constituent functions, $G = \{g^j | j = 1, 2, \ldots\}$, to mimic the role of corresponding agent; $f_i = g^{move} \circ g^{\cdots} \circ \ldots \circ g^{see}$.

The system provides a number of predefined constituent functions to perceive the high resolution environment, do complex path planning for different scenarios, identify whether a path is blocked, follow someone, etc. Each agent uses g^{see} function to scan its visible surrounding in high resolution (every 0.5°) like a radar; each agent has its own eyesight distance, which is typically set within 30m to 100m. This high resolution scanning enables agents to detect dynamic changes in the environment, move avoiding collision obeying restrictions like walk along road edges or drive on road lanes.

The constituent function for collision avoidance, g^{coll_av}, is implemented adopting the Optimal Reciprocal Collision Avoidance (ORCA) (Berg et al. (2011)) with minor modifications and additional parameters to model interactions among agents (Melgar et al. (2013), Melgar et al. (2014)). The parameters of g^{coll_av} were tuned to model collision avoidance among pedestrians and pedestrians, cars and cars, and cars and pedestrians. In network flow based simplified models, within each road link, the rate of evacuees' flow is controlled according to a relevant fundamental diagram (i.e., relation between speed and flow density which is obtained from field observations). In contrast, in the developed system, the agents' speeds are not controlled externally. Instead, each agent autonomously controls its moving speed and direction using g^{coll_av}, which is tuned to reproduce observed fundamental diagrams. Several predefined update functions, f's, which are composed of suitable constituent functions are available to mimic roles of residents, visitors, officials who advise people to evacuate, cars, pedestrians.

3. High Performance Computing Extension

In order to meet the high computational demand of the autonomous agents, a scalable High Performance Computing (HPC) extension was developed. Some of the basic constituent functions, like g^{see} and $g^{identify\_road\_blocks}$, are computationally expensive. g^{see}, which is based on a memory intensive ray tracing algorithm, scans the visible neighborhood in high resolution. $g^{identify\_road\_blocks}$ is used to identify road blockages due to fallen debris or inundation and detect whether it is possible to continue avoiding debris. When encountered with road blockages and congestion, agents replan their paths or even start looking for new destinations. The processor intensive and memory intensive nature of these functions demand a significant amount of computational resources. As an example, a simulation with 90,000 agents with the environment shown in Fig. 1 required 33 node hours in K Computer; a computing node of K-computer consists of an 8-core SPARC64 VIIIfx processor with 16GB of RAM. In order to address this high computational demand, a shared and hybrid parallel extension was developed. This section presents some details of the implemented HPC extension.

3.1. Domain decomposition

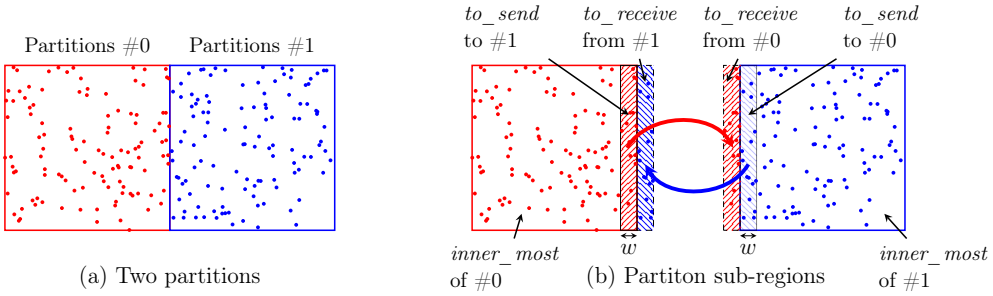


Figure 3: Domain decomposed for 2 MPI processes. In order to maintain the continuity, a region of width w from the neighbor partition #0 is included along the boundary of the partition #1

In order to utilize computational power of multiple computing nodes in a computer cluster, the domain of the environment is partitioned such that each computing node is assigned nearly equal workload. Almost all the computational workload is due to agents, and execution time related to the environment is mostly memory bound. Since the execution time of the heterogeneous agents highly depends on their type, visible surrounding, the density of other agents in the neighborhood, etc., we use measured execution time of each agent as a weight in creation partitions with equal work load. Kd-tree based 2D partitions are generated using agents' execution time as weight, and each partition is assigned to an MPI process. A shared memory compute node is designated as an MPI process, and computations within a node are accelerated using OpenMP threads. Specifically, we use OpenMP's *task* level parallelism to accelerate computations.

Figure 3 shows the domain decomposed for two MPI processes. It is necessary to exchange information among those to ensure the continuity of the problem, since the compute nodes cannot directly access each other's memory. Along the boundary of each partition, we add dummy copies of a w wide region from the neighbor partitions (see Fig. 3(b)). In HPC literature, this dummy layers are referred as *halo*, or *ghost*. At the end of each iteration step, the information of the dummy agents in ghost layers are updated according to the latest states of the originals in the corresponding neighbor partitions, using MPI. By using the information of the agents in the ghost regions, when executing the agents of its own partition, MPI process can eliminate the effect of artificial boundaries introduced by partitioning (i.e. maintain the continuity). Since agents' actions are affected by the information within their visible surrounding, the thickness, w, of the ghost region is set to the longest eyesight of the agents.

3.1.1. Communication hiding

The inter-node communication, which is required for maintaining the continuity, is an extra overhead, and this overhead increases with the number of MPI processes. Communication hiding (i.e. doing useful work during communication) is a standard technique to minimize the performance degradation due to communication overhead. In order to hide communications, we group the agents in an MPI process into three mutually exclusive sets. *to_send* includes all the agents to be sent to neighbor MPI processes, while *to_recv* group includes all the agents in ghost regions. The rest are included in *inner_most*.

In order to hide the communications, first the agents in the set *to_send* is executed, and non-blocking messages to send the latest state of these agents to the corresponding neighbor MPI processes are initialized using non-blocking *MPI_Isend* and *MPI_Irecv* functions. While the communications proceed, the agents in *inner_most* set are executed. As long as execution of *inner_most* set takes longer than the time to complete the communication, we can effectively eliminate the communication overhead. Once, execution of *inner_most* is completed, the non-blocking messages are finalized, which updates the agents in ghost regions to latest states.

3.2. Dynamic load balancing

With time, agents move from the domain of one partition to another, which we call *agent migration* in this paper. Unlike the above discussed ghost region updates, permanently moving agents to neighbor partitions is expensive. Movement of a large number of agents in or out from a partition leads to imbalances of workloads assigned to MPI processes, lowering the computation efficiency. When the load imbalance reaches a critical state, domain is re-partitioned to re-assign equal workloads. Figure 3 (a), (b) and (c) show how the partitions are rearranged according to the agent distribution, at early stages of a simulation with 80,000 agents. The process of this dynamic load balancing is the heaviest in communication and can consume a significant time depending on the total number of agents being simulated. Figure 4 shows the time history of run-time with 10 million agents in a 588 km^2 region. The small spikes indicate *migration*, while the nearly 60s long spikes at 200 intervals indicate dynamic load balancing.

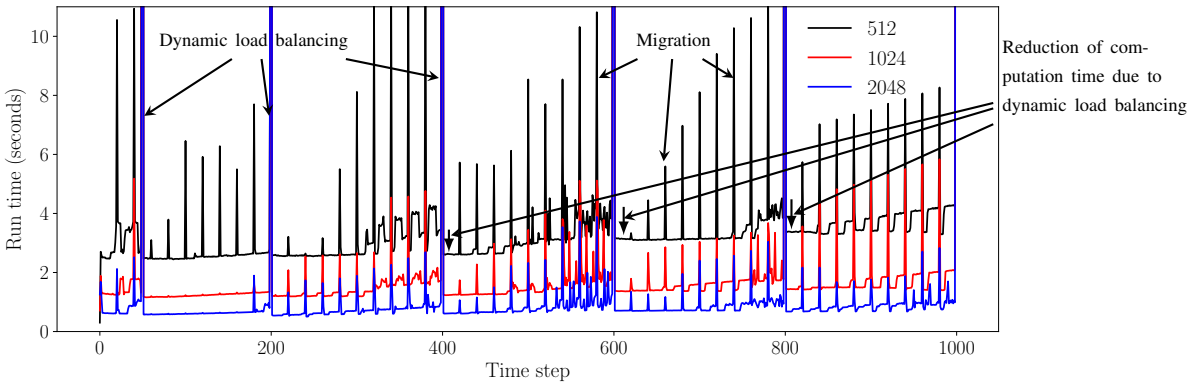
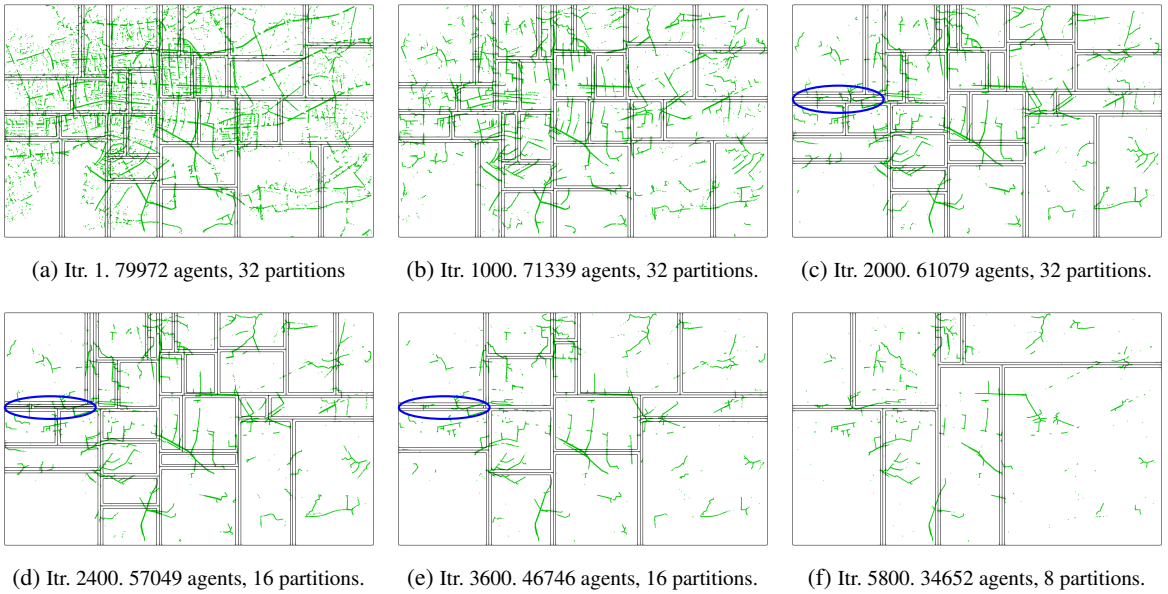


Figure 4: Run-time history for 1000 iterations with 10 million pedestrian agents. Three graphs indicate time history with 512, 1024 and 2048 nodes of K computer. Migration and dynamic load balancing are executed on a single thread.



(a) Itr. 1. 79972 agents, 32 partitions (b) Itr. 1000. 71339 agents, 32 partitions. (c) Itr. 2000. 61079 agents, 32 partitions.

(d) Itr. 2400. 57049 agents, 16 partitions. (e) Itr. 3600. 46746 agents, 16 partitions. (f) Itr. 5800. 34652 agents, 8 partitions.

Figure 5: Partition arrangements at different iteration steps with 80,000 agents in the environment of Fig. 1. Partitions are dynamically adjusted according to the distribution of agents and workload; shown in green dots are the agents. When the number of agents is too small and/or concentration of agents leads to partitions with $2w$ or lesser side length, like in (d) and (e), the problem has to be mapped to a smaller number of MPI processes.

Number of nodes	Execution time (s)	Strong scalability (%)
512	3904.0	-
1024	2067.7	94.4
2048	1258.5	82.1

Table 1: Runtime and strong scalability with 512, 1024 and 2048 nodes in K computer.

3.3. Dynamically mapping to different number of MPI processes

With the progress of time, the agents start to concentrate along roads leading towards evacuation regions, and decrease in number since evacuated agents are deactivated. Both these can force to reduce the number of MPI processes. Since the minimum width of the *to_receive* regions, w, is set to the maximum eyesight, the length or width of a partition cannot be smaller than $2w$. The concentration of agents to a few number of streets, as seen in Fig. 5 (d) and (e), give rise to partitions with closer to $2w$ side lengths. When that happens, either the program has to be aborted, or the problem should be mapped to a smaller number of MPI processes. Though this mapping is a complex and time consuming process, it is necessary to prevent premature abortions and continue until the end of the desired simulation. Figure 5 (d), (e), and (f) show the partitions after mapping to 16 and 8 MPI processes.

3.4. Scalability

In order to conduct the target large scale evacuation simulations (e.g., tsunami evacuation of a long stretch of coastal region, or evacuation of large metropolis like Tokyo after a major earthquake) utilizing HPC facilities, the developed code must scale to a large number of MPI processes. Near ideal scaling (i.e., halving the execution time when doubling the number of MPI processes) to thousands of compute nodes with the complex heterogeneous agents

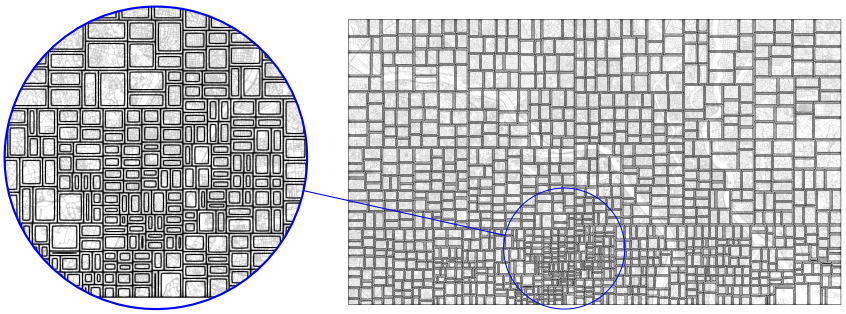


Figure 6: 2048 partitions of Tokyo domain. High concentrations of agents in the central Tokyo, encircled with a black ellipse, gives rise to partitions with smaller dimensions and poor aspect ratio. Presence of partitions with side lengths close to $2w$ makes communication hiding ineffective and significantly lower the scalability.

is a challenging task. According to autohors' literature survey, there are no records of agent based models of similar complexity which scales even to hundred compute nodes.

In order to test the scalability of the developed system, we conducted several simulations with 10 million pedestrians in a 588 km^2 area in central Tokyo. The pedestrians were set to move to the nearest park, and their walking speeds were set according to Bohannon (1997). To produce maximum computation load, all the agents were set to start evacuation immediately. 1000 iteration steps were simulated on 512, 1024 and 2048 nodes of K computer.

Table 1 shows the strong scalability obtained from the simulations. Strong scalability is defined as $({}^{T_m}/_{T_n})/({}^{n}/_{m})$, where T_i is the execution time with i number of MPI processes, and $n \geq 2m$. These simulations indicate that not only the developed scales well up to 2048 nodes (16384 CPU cores), but also it can handle millions of agents in several hundreds of square kilometers in $1{\times}1\text{m}^2$ resolution model of the environment. Figure 4 shows the time history of run-times. The tall spikes (nearly 60s tall) indicate the dynamic load balancing at 100 iteration steps intervals. The smaller spikes indicate the migration of agents from one partition to another, at 20 iteration intervals.

The developed system has potential to scale even beyond the results shown in Table 1. Currently, only the execution of agents uses shared memory parallelism to best utilize multiple threads in each compute node. The migration and dynamic load balancing functions are executed on single threads. Enhancing these time consuming functions with shared memory parallelism will further improve the scalability.

3.4.1. Partitions and scalability

In the central Tokyo environment, which is used for the above scalability tests, the city center had a high agent density compared to the outskirts (see Fig. 6). This high agent concentration produced smaller partitions, while, with time, agents start to further concentrate along the roads to evacuation regions, like in Fig. 5. This high concentrations of agents lead to partitions with side dimensions closer to $2w$ and poor aspect ratios, like the one encircled with a blue color ellipse in Fig. 5(c). In partitions with dimensions closer to $2w$, most of the agents are located in *to_receive* regions, and have only smaller number of agents in *inner_most* region. This makes the communication hiding ineffective, and significantly lowers the scalability. The simulation with 2048 nodes starts to produce partitions with side dimensions closer to $2w$ at iteration 600 and onward, which is why there is 12% reduction in scalability with 2048 nodes. Partitioning scheme with better aspect ratio will enable to maintain the high scalability up to a larger number of MPI processes.

4. Concluding remarks

Details of an agent based model for simulating emergency mass evacuations with complex agents and a high resolution model of environment are presented. In order to meet the computational demand of constituent functions of complex agents, a scalable HPC extension was implemented. It is demonstrated that the system has high strong scalability up to 2048 compute nodes and capable of simulating millions of agents in several hundred square kilometers. In order to deal with significant reduction of agents towards the end of a simulation, and partitions narrower than two

times the maximum eye sight of agents, real-time transition to smaller number of MPI ranks is implemented. Narrow partitions with poor aspect ratios is one of the major problem in maintaining high scalability across a large number of MPI processes. Further improvements in parallel scalability is possible by enhancing migration and re-partition with shared memory parallelism, and implementing a better portioning scheme with high quality partitions.

Acknowledgements

Most of the results are obtained using K computer at the RIKEN Advanced Institute for Computational Science (hp150283).

References

Aguilar, L., Wijerathne, L., Hori, M., Ichimura, T. and Tanaka, S., 2015. Mixed Mode Large Urban Area Tsunami Evacuation Considering Car-pedestrian Interactions. Journal of Japan Society of Civil Engineers, Ser. A2 (Applied Mechanics (AM)) 71.2, 633-641.

Beeson, P., Jong, N. K., and Kuipers, B., 2005. "Towards autonomous topological place detection using the Extended Voronoi Graph," Proceedings of the 2005 IEEE International Conference on Robotics and Automation, pp. 4373-4379.

Bohannon R. W., 1997. Comfortable Walking Speed of Adults Aged 20-79 Years: Reference Values and Determinants. Age and Ageing 26, 15-19.

Melgar, L. E. A., Wijerathne, L., Hori, M., Ichimura, T. and Tanaka, S., 2013. On the Development of an MAS Based Evacuation Simulation System: Autonomous Navigation & Collision Avoidance. Lecture Notes in Computer Science, Vol. 8291, pp. 388-395.

Melgar, L. E. A., Wijerathne, L., Hori, M., Ichimura, T. and Tanaka, S., 2013. A Scalable Workbench for Large Urban Area Simulations, Comprised of Resources for Behavioural Models, Interactions and Dynamic Environments. Lecture Notes in Computer Science, Vol. 8861, pp. 166-181.

Berg, J. V. D., Guy, S. J., Lin, M., and Manocha, D., 2011. "Reciprocal n-body collision avoidance," Robotics Research: The 14th International Symposium ISRR, Springer Tracts in Advanced Robotics, vol. 70, Springer-Verlag, May , pp. 3-19.

Maddegedara, L., Hori, M., and Ichimura, T., 2013. Parallel Scalability Enhancements of Seismic Response and Evacuation Simulations of Integrated Earthquake Simulator. Lecture Notes in Computer Science, Vol. 7851, pp. 105-117.

Leonel Aguilar, Maddegedara Lalith, Tsyoshi Ichimura, and Muneo Hori, 2017. "On the performance and scalability of an HPC enhanced MAS based evacuation simulator", Procedia Computer Science, pp. 937-947.

Wijerathne, M. L., Melgar L. A., Hori, M., Ichimura T., and Tanaka S., 2013. "HPC enhanced large urban area evacuation simulations with vision based autonomously navigating multiagents", Procedia Computer Science, pp. 1515-152.