Enhancing Radiation Emergency Simulations Using FLAMEGPU2: Performance and Scalability Improvements

Minho Hwang, Geon Kim, Joonseok Lim, Sungmin Han, Gyunyoung Heo*
Department of Nuclear Engineering, Kyung Hee Univ., Deogyeong-daero, Giheung-gu, Yongin-si, Gyeonggi-do 17104, Republic of Korea
**Corresponding author: gheo@khu.ac.kr*

***Keywords :** Radiological emergency, Agent-based simulation, Flame GPU 2, Parallelization

1. Introduction

A radiological emergency involves numerous variables, making the rapid and accurate formulation of evacuation strategies crucial for saving many lives. To address this, the previously introduced PRISM (Platform for Radiological Emergency Agent-Based Integrated Simulation Model)[1, 2] employs agent-based modeling (ABM) to simulate evacuation processes with precision, aiming to optimize evacuation routes and reduce evacuation time. While PRISM can analyze various scenarios by considering real-time changes, the integration of multiple models leads to increased computational load, resulting in long simulation completion times. Therefore, optimization is needed to enhance the simulation's performance and scalability.

The original PRISM has been developed on NetLogo, which had the advantage of offering GIS data and a user-friendly interface. However, due to the speed limitations of the Logo language, extensive computations required significant time, forcing the extension of the time step to secure speed at the cost of simulation accuracy. To address this issue, we transitioned PRISM to FLAMEGPU, significantly improving its performance. FLAMEGPU utilizes GPU parallel processing and the C++ language to offer much faster computation than NetLogo and supports Python, making model updates and maintenance easier. With this upgrade, PRISM has overcome the previous limitation of handling only 20,000 agents, now enabling simulations of radiological emergency evacuations involving hundreds of thousands of agents. This study presents a methodology for enhancing the accuracy, scalability, and computational speed of radiological emergency simulations through the improved performance of PRISM, thereby enabling the formulation of more effective and reliable evacuation strategies.

2. Overview of PRISM

2.1 Agent-Based Modeling

PRISM utilizes ABM techniques to simulate the evacuation process of residents under different scenarios during a radiological emergency. The ABM approach demonstrates complex macro-level phenomena through the interaction of agents, which act based on simple algorithms. In PRISM, residents, who serve as the agents, possess key variables such as initial location, speed, and radiation exposure levels. The interactions between these agents are determined by the interactions among these variables and are calculated according to simple behavioral rules. For instance, in PRISM's traffic model, the current speed of an agent is calculated by considering factors such as lane position, distance to other agents, and speed, which then determines the agent's location in the subsequent time step. By leveraging these characteristics, macro-level phenomena can be observed by adjusting factors such as the number of agents or their choice of evacuation routes. PRISM has been utilized to conduct studies on the optimal placement of evacuation shelters and staged evacuation strategies[3].

2.2 Model Description and Operational Sequence

PRISM is an integrated platform that incorporates various models, including a traffic model, infrastructure model, panic model, pathfinding model, and dispersion model. The traffic and panic models are executed by algorithms within PRISM, while the infrastructure and pathfinding models are executed by external Python code. The dispersion model is implemented with connection to the results of an external software called HYSPLIT (Hybrid Single-Particle Lagrangian Integrated Trajectory).

The traffic model calculates the current speed by considering factors such as inter-vehicle distance and speed differences, using the Panic integrated Intelligent Driver Model (P-IDM), and then moves the agent a distance corresponding to the speed in the direction the agent is facing. The panic model calculates the panic level based on the distance to the destination, the distance between agents, and speed differences; this panic value is then used as a parameter in the traffic model (P-IDM). The infrastructure model[1, 4], designed to facilitate smooth evacuation for agents, defines infrastructure entities labeled as "police" and "medical." The police infrastructure smooths traffic flow, while the medical infrastructure adjusts the supply speed for residents arriving at evacuation shelters. Specifically, the police infrastructure improves speed by removing a certain proportion of (invisible) agents,

which exist to correct speeds on pre-existing routes. The medical infrastructure adjusts the speed at which supplies are distributed to evacuation shelters with large crowds. The total capacity of this infrastructure is capped, and reinforcement learning is employed to activate the most appropriate infrastructure at optimal times.

The pathfinding model uses the A* algorithm and employs Python's network library. The road network in PRISM is modeled using GIS data, accurately replicating the actual roads within approximately 5 km of the nuclear plant site. The dispersion model is calculated using HYSPLIT and utilizes meteorological data provided by the NOAA (National Oceanic and Atmospheric Administration) Air Resources Laboratory. This model predicts and backtracks the trajectory of radioactive materials. Although it takes longer to compute than the Gaussian puff model, it provides more accurate trajectory predictions. The results are extracted as shp files and integrated into PRISM.

Figure 1 illustrates the operational sequence of the model. As PRISM is a multi-agent simulation, computational speed is of paramount importance. The most time-consuming component is the traffic model, as every agent calculates the distance and speed differences with nearby agents and then moves a corresponding distance in an iterative process. This time requirement increases exponentially as the number of agents increases. To address this issue, we propose the use of FLAMEGPU.

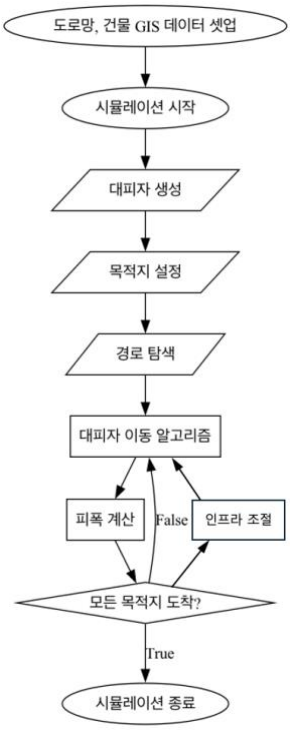

Fig. 1. Model Algorithm Flowchart.

3. FLAMEGPU2 Integration

3.1 What is FLAMEGPU?

FLAMEGPU (Flexible Large-scale Agent Modelling Environment for Graphics Processing Units) is a GPU (Graphics Processing Unit) based parallel processing framework designed for high-performance agent-based simulations[5]. This framework enables the modeling of complex systems by breaking them down into individual agents and simulating the overall behavior of the system as these agents interact with one another. The key strength of FLAMEGPU lies in its ability to leverage NVIDIA's CUDA platform to fully utilize the parallel processing power of GPUs, allowing it to perform large-scale simulations much faster than traditional CPU-based simulations. For instance, it can efficiently handle simulations involving millions to billions of agents.

This framework is designed to be applicable across various research fields, making it an essential tool for researchers who need to process and analyze large-scale data in domains such as ecology, social sciences, economics, and physics. FLAMEGPU2 was developed by the Computer Science department at the University of Sheffield in the United Kingdom and is a significantly enhanced version of the initial FLAME GPU, with improved performance and functionality.

3.2 How Parallel Processing Works

Traditional agent-based simulation tools like NetLogo primarily operate on CPUs, and in most cases, handle simulations in a single-threaded manner. As the scale of the simulation increases—meaning more agents and more complex interactions—the computation time increases exponentially, creating significant performance bottlenecks.

FLAMEGPU addresses this issue by actively utilizing the parallel processing capabilities of GPUs to efficiently perform large-scale agent-based simulations. This system is built on NVIDIA's CUDA platform, allowing millions of agents to perform computations simultaneously. Each agent is treated as an individual computation unit, and thousands of cores within the GPU execute these units in parallel. As a result, FLAMEGPU greatly enhances the overall computation speed by processing each agent's state updates and interactions with other agents in parallel.

Agent interactions are primarily handled through message passing. FLAMEGPU parallelizes this message passing process as well, ensuring that all agents can send and receive data at the same time. In doing so, FLAMEGPU efficiently organizes and filters the messages generated by agents, optimizing the access and processing of data distributed across GPU memory.

FLAMEGPU also effectively manages synchronization and asynchronization processes. While synchronization between agents is necessary at certain computational stages, other stages can be processed asynchronously to maximize the benefits of parallel processing. For example, synchronization is required during the message-passing phase between agents, but subsequent state updates can be processed asynchronously. This approach allows FLAMEGPU to optimize the performance of parallel processing while ensuring accurate simulation results.

4. Benchmark Results

The simulation environment for PRISM is shown in Table 1. In the scenario, all evacuees depart simultaneously and move towards evacuation shelters located within approximately 5 km from the nuclear power plant (Figure 2). The computer specifications used are as follows: CPU: AMD Ryzen Threadripper PRO 3995WX 64-Cores, RAM: 128 GB, GPU: NVIDIA GeForce RTX 3080ti. While NetLogo's visualization features require the performance of a graphics card, this does not impact the calculations. Additionally, although NetLogo offers a multi-processor feature called BehaviorSpace, each model typically operates on a single processor.

When programming the traffic model in ABM, the agent perception algorithm is critically important. The presence or absence of obstacle recognition changes the vehicle's speed and, consequently, the simulation results. In the case of the pre-update PRISM, the obstacle recognition algorithm used NetLogo's 'in-cone' function, which allowed agents to recognize other agents as obstacles within a certain angle and distance in front of them. If there were fewer obstacles than lanes, they were not recognized as obstacles. With the update to FLAMEGPU, the obstacle recognition algorithm was improved. Using the collected messages, the position and heading direction of other agents are used to determine if they are in front and facing the same direction. An improved model that considers lane changes for IDM has been proposed in recent studies [6], but the current version of PRISM does not take lane changes into account. Lane changes can cause congestion, and modeling such decision-making remains a limitation. In the future, the decision-making model will be updated to include changes in destination, route selection, and lane changes during the evacuation process. At merge points, vehicles will merge in a random order. Therefore, only vehicles in the same lane are recognized as obstacles, resulting in more accurate outcomes compared to the pre-update PRISM.

Table I: Simulation parameters

Name	Value
Number of Evacuees	20,000 #
Evacuation start time	0 sec
Maximum speed	50 km/h
Time step	10 sec

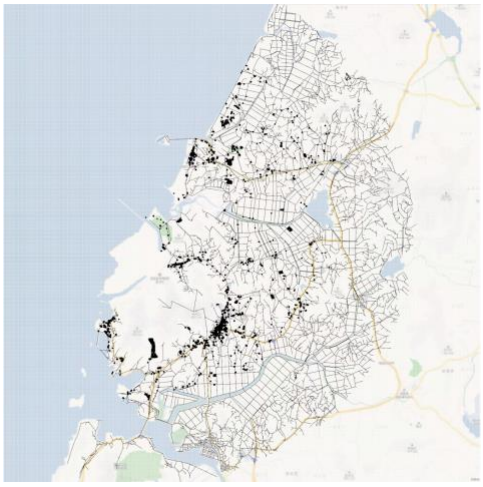

Fig. 2. PRISM simulation screen

4.1 Result with NetLogo

Figure 3 shows the results from the pre-update PRISM. The actual simulation time was approximately 2 hours and 3 minutes. It took 483 steps for all 20,000 evacuees to reach their destination. The average speed was 11.90 km/h.

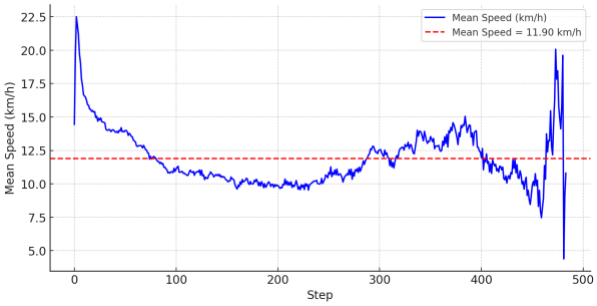

Fig. 3. Average speed of evacuees over simulation steps (Pre)

4.2 Result with FLAMEGPU2

Figures 4 shows the results of PRISM executed using FLAMEGPU2. The actual simulation time was 2 minutes, which is 61.5 times faster than the pre-update PRISM. It took 334 steps for all 20,000 evacuees to reach their destination, with an average speed of 8.21 km/h.

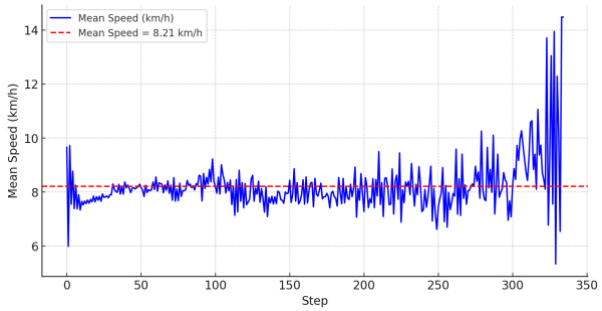

Fig. 4. Cumulative number of arrived evacuees over simulation steps (time step=10 sec)

With the improved execution speed provided by FLAMEGPU2, the time step could be reduced from 10 seconds to 1 second. Figures 5 shows the results when the simulation was run with a 1-second time step. The actual time required was 31 minutes. It took 6,668 steps for all 20,000 evacuees to reach their destination, with an average speed of 5.66 km/h. While the pre-update PRISM increased the time step for speed at the cost of some accuracy, these results can be considered more accurate. As more models are added, it is expected that this improved version will be useful for analyzing various cases.

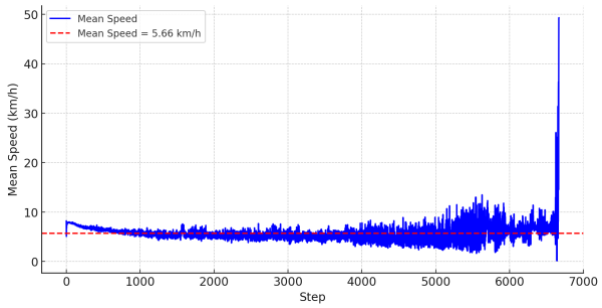

Fig. 5. Average speed of evacuees over simulation steps (time step=1 sec)

5. Conclusion

In this study, we have demonstrated the significant performance and scalability improvements achieved by integrating FLAMEGPU2 into the PRISM framework. By leveraging the parallel processing capabilities of GPUs, FLAMEGPU2 allowed us to drastically reduce simulation time while simultaneously enhancing the accuracy of the results. The ability to handle a much larger number of agents and perform simulations at finer time steps without sacrificing speed represents a substantial advancement over the pre-update PRISM, which was constrained by the limitations of CPU-based processing.

The benchmark results highlight the effectiveness of FLAMEGPU2 in optimizing computational speed and handling complex, large-scale scenarios in radiological emergency simulations. By reducing the time step from 10 seconds to 1 second, we were able to model agent behaviors and interactions more precisely, allowing for a more detailed analysis of agent behavior and interactions during the evacuation process. Additionally, the improved obstacle recognition algorithm in FLAMEGPU2 contributed to more accurate traffic modeling, further enhancing the realism of the simulation.

The increased computational efficiency enables the exploration of more complex scenarios and the incorporation of additional models, which can lead to more robust and comprehensive evacuation strategies in radiological emergencies. As a result, PRISM with FLAMEGPU2 serves as a powerful tool for researchers and emergency planners, facilitating the development of more effective evacuation plans that can save lives in real-world emergencies.

ACKNOWLEDGMENT

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korean government(MSIP:Ministry of Science, ICT and Future Planning) (No. NRF-2021M2D2A1A02044210).

REFERENCES

[1] Hwang, Y., & Heo, G. (2021). Development of a radiological emergency evacuation model using agent-based modeling. Nuclear Engineering and Technology, 53(7), 2195-2206.
[2] Kim, G., & Heo, G. (2023). Agent-based radiological emergency evacuation simulation modeling considering mitigation infrastructures. Reliability Engineering and System Safety, 233, 109098.
[3] Han, S., Lim, J., Hwang, M., & Heo, G. (In-press). Enhancing Radiological Emergency Response through Agent-Based Model Case 1: Effectiveness of Staged Evacuation. Korean Journal of Chemical Engineering.
[4] Hwang, M., Kim, G., Lim, J., & Heo, G. (2023). Platform for radiological emergency agent-based integrated simulation model to optimize evacuation planning. Asian Symposium on Risk Assessment and Management 2023.
[5] Richmond, P., Chisholm, R., Heywood, P., Leach, M., & Kabiri Chimeh, M. (2023). FLAME GPU 2: A framework for flexible and performant agent-based simulation on GPUs. *Software: Practice and Experience, 53*(8), 1659–1680. https://doi.org/10.1002/spe.3207
[6] Holley, D., D'sa, J., Nourkhiz Mahjoub, H., Ali, G., Chalaki, B., & Moradi-Pari, E. (2023). Merge Reactive Intelligent Driver Model: Towards Enhancing Laterally Aware Car-following Models. *arXiv preprint arXiv: 2305.12014v1*. https://doi.org/10.48550/arXiv.2305.12014