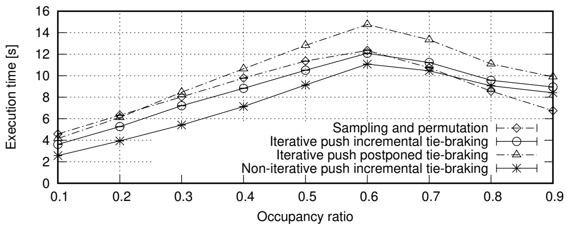

Figure 3: Simulation runtime with global movement.

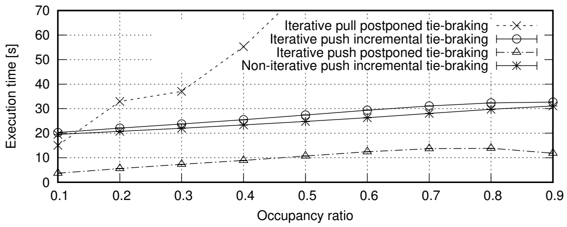

Figure 4: Simulation runtime with limited neighborhood.

figure. As the non-iterative approaches depend on incremental tie-breaking, combinations with postponed tie-breaking are excluded as well. We varied the occupancy ratio, i.e., the ratio of populated cells, from 0.1 to 0.9. The simulation space was a grid of about 16 million (4096×4096) cells. The agents' happiness threshold was set to 5. The simulation was terminated after 100 simulation cycles. The results show a minor runtime reduction of non-iterative push over iterative push. The sampling and permutation approach performs worse under low occupancy ratios. However, for occupancy ratios above 0.7, sampling and permutation outperforms the push approaches. The reason is that at high occupancy ratios, with the push approaches, agents will perform many retries until an open cell is found and potential conflicts are won. Sampling and permutation avoids these situations by directly mapping agents to the open cell set. At a smaller grid size of 1024×1024 cells, the relative performance of the push approaches remained roughly the same. However, sampling and permutation was outperformed by a factor of about 2.9 to 5.2. Generally, the performance is affected by the number of moveable agents and conflicts. The largest number of conflicts was generated at an occupancy ratio of 0.6, coinciding with the longest observed execution times.

The performance of iterative push is affected by the number of iterations required to resolve all conflicts, which can be computed by iteratively determining the number n of agents that lose a conflict according to the birthday paradox [4]: $n = |A| - |C| + |C|(1 - \frac{1}{|C|})^{|A|}$, where |A| is the number of agents and |C| is the number of cells.

Figure 4 shows the execution times of the various approaches when the agent movement is limited to a 3×3 neighborhood on a grid of 4096×4096 cells. For incremental tie-breaking, the approach described in Section 4 to avoid bias is used. As discussed in Section 3.3, sampling and permutation cannot be apply to restricted neighborhoods. In contrast to the unrestricted movement, the incremental tie-breaking approaches require the computation of random permutations in order to avoid bias. If an agent cannot find an unoccupied cell, its neighborhood is enlarged in steps of 3 cells after 10 trials each. We observed neighborhoods up to 63×63 cells at occupancy ratio 0.9. Since in the pull approach, the cells scan for interested agents, they have to consider the largest neighborhood used by any agent in the current iteration. Accordingly, iterative pull was substantially slower than the push approaches, requiring up to 1127 seconds at an occupancy ratio of 0.9. Given these results, we did not implement iterative pull with incremental tie-breaking. Iterative and Non-iterative push with incremental tie-breaking requires the computation of a random permutation for each simulation cycle. Since postponed tie-breaking does not require this step, it consistently achieved the lowest execution times.

6 CONCLUSIONS AND OUTLOOK

We systematized and evaluated conflict resolution approaches for agent-based simulation on GPUs from the literature and proposed two new variants. Our measurements indicate that if agents compete for resources globally without restriction to a certain neighborhood, a non-iterative approach achieves best performance. If the numbers of agents and conflicts are both large, a direct computation of a random mapping between agents and resources performed the best. If agents consider resources within a limited neighborhood, a postponed tie-breaking between competing agents substantially outperformed the alternatives. We further discussed ways to avoid bias in the conflict resolution. Our current observations were made only based on the classic segregation model, which is simple but often used to illustrate the power of agent-based modeling. Future work could extend our observations to more practical applications such as traffic simulation.

7 ACKNOWLEDGMENTS

This work was financially supported by the Singapore National Research Foundation under its Campus for Research Excellence And Technological Enterprise (CREATE) programme.

REFERENCES

[1] Joshua M Epstein and Robert Axtell. 1996. *Growing Artificial Societies: Social Science from the Bottom up.* Brookings Institution Press.

[2] Mikola Lysenko and Roshan M. D'Souza. 2008. A Framework for Megascale Agent Based Model Simulations on Graphics Processing Units. *Journal of Artificial Societies and Social Simulation* 11, 4 (2008), 10.

[3] Thom McLean and Richard Fujimoto. 2000. Repeatability in Real-Time Distributed Simulation Executions. In *Proceedings of the Workshop on Parallel and Distributed simulation.* IEEE, 23–32.

[4] Michael Mitzenmacher and Eli Upfal. 2005. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis.* Cambridge University Press.

[5] Kai Nagel, Dominik Grether, Ulrike Beuck, Yu Chen, Marcel Rieser, and Kay W Axhausen. 2008. Multi-Agent Transport Simulations and Economic Evaluation. *Jahrbücher für Nationalökonomie und Statistik* 228, 2-3 (2008), 173–194.

[6] Kalyan S Perumalla and Brandon G Aaby. 2008. Data Parallel Execution Challenges and Runtime Performance of Agent Simulations on GPUs. In *Proceedings of the Spring Simulation Multiconference.* SCS, 116–123.

[7] Paul Richmond. 2014. Resolving Conflicts Between Multiple Competing Agents in Parallel Simulations. In *European Conf. on Parallel Processing.* Springer, 383–394.

[8] Robert Rönngren and Michael Liljenstam. 1999. On Event Ordering in Parallel Discrete Event Simulation. In *Proceedings of the Workshop on Parallel and Distributed Simulation.* IEEE, 38–45.

[9] Peter Sanders. 1998. Random Permutations on Distributed, External and Hierarchical Memory. *Inform. Process. Lett.* 67, 6 (1998), 305–309.

[10] Peter Sanders, Sebastian Lamm, Lorenz Hübschle-Schneider, Emanuel Schrade, and Carsten Dachsbacher. 2018. Efficient Parallel Random Sampling – Vectorized, Cache-Efficient, and Online. *ACM Trans. Math. Softw.* 44, 3 (2018), 29:1–29:14.

[11] Thomas C Schelling. 1971. Dynamic Models of Segregation. *Journal of Mathematical Sociology* 1, 2 (1971), 143–186.

[12] David Strippgen and Kai Nagel. 2009. Using Common Graphics Hardware for Multi-Agent Traffic Simulation with CUDA. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques.* ICST, 62.