Optimize Data-Driven Multi-Agent Simulation for COVID-19 Transmission

Chao Jin (✉ jinchaoahpc@gmail.com)

National Supercomputing Center in Shenzhen

Hao Zhang

Chinese Academy of Sciences

Ling Yin

Chinese Academy of Sciences

Yong Zhang

Chinese Academy of Sciences

Sheng-zhong Feng

National Supercomputing Center in Shenzhen

Research Article

Keywords: multi-agent simulation, case-focused method, hash table

Posted Date: March 2nd, 2022

DOI: <https://doi.org/10.21203/rs.3.rs-1378622/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

RESEARCH

Optimize Data-Driven Multi-Agent Simulation for COVID-19 Transmission

Chao Jin^{1*}, Hao Zhang^{2,3}, Ling Yin², Yong Zhang² and Sheng-zhong Feng¹

*Correspondence:

jinchaohpc@gmail.com

¹National Supercomputing Center in Shenzhen, Shenzhen 518055, Guangdong, People's Republic of China

Full list of author information is available at the end of the article

Abstract

Background: Multi-Agent Simulation is an essential technique for exploring complex systems. In researches of contagious diseases, it is widely exploited to analyze their spread mechanisms, especially for preventing COVID-19. Nowadays, transmission dynamics and interventions of COVID-19 have been elaborately established by this method, but its computation performance is seldomly concerned. As it usually suffers from inadequate CPU utilization and poor data locality, optimizing the performance is challenging and important for real-time analyzing its spreading.

Results: This paper explores approaches to optimize multi-agent simulation for COVID-19 disease. The focus of this work is on the algorithm and data structure designs for improving performance, as well as its parallelisation strategies. We propose two successive methods to optimize the computation. We construct a case-focused iteration algorithm to improve data locality, and create a thread-safe data-mapping paradigm called hierarchical hash table to accelerate hash operations.

Conclusions: Our performance results demonstrate capabilities of these methods exhibiting significant improvements of system performance. The case-focused method degrades $\sim 90\%$ cache references and achieves $\times 4.3$ speedup. Hierarchical hash table can further boost computation speed by 47%. And parallel implementation with 20 threads on CPU achieves $\times 81$ speedup consequently.

Keywords: multi-agent simulation; case-focused method; hash table

Background

Multi-agent simulation (MAS) is an essential technique for exploring system phenomena in which the overall behaviour is determined by the constituent autonomous entities [1]. It provides an effectively tool to modeling systems with complex organization and non-linear interactions. As agents can be applied with different disciplines, it is widely used in studying social, economic, organization and epidemiology sciences [2, 3, 4]. Domínguez et al [5] propose multi-agent modeling for complex supply chains to overcome limitations from classical methods. McArthur et al [6] investigate MAS technology in power industry with complex scalable power networks. Moreover, it is also evaluated efficient in smart city, such as controlling city congestion, pollution and delivery time [7]. Many other applications can be found in [8].

An significant application of MAS is studying infectious diseases. Classical methods use mathematical model to emulate the transmission of infectious diseases. Some classical examples include Susceptible-Infect-Susceptible (SIS) epidemic model [9],

Susceptible-Infect-Recovered (SIR) epidemic model [10], etc. However, these simple models are not afford to analyze complex and fine-grained systems. MAS integrating evolution and phylogeny helps to understand emerging infectious diseases in complex systems. Dion et al [11] leverage it to study the landscape epidemiology of foot-and-mouth disease in South Africa. Yergen et al [12] propose IDESS for rapidly constructing MAS models of Avian Flu (H5N1) virus spreading.

Outbreaks of COVID-19 raise concerns about effectively preventing spread of infectious diseases, and MAS is widely adopted for studying this issue. Its fine-grained spreading dynamics is established through many efforts. Castro et al [13] analyze the spread processes of COVID-19 epidemics in open regions by considering effects from different environments. Vyklyuk et al [14] propose modeling its spread in large regions by simulating a set of autonomous multi-agent systems. Nanna et al [15] extend MAS to dynamically verify influences on diseases spread from government strategies. For COVID-19 preventions, both non-pharmaceutical interventions (NPI) and pharmaceutical interventions (PI) have been elaborately invested [16] with this paradigm. Yin et al. [17] proposed a data-driven NPI MAS model to suppress the diseases in Shenzhen and evaluate strategies including contact tracing, mask wearing and prompt testing. Zhou et al [18] examine the spatial heterogeneity of the disease transmission and optimize vaccine distribution strategies considering spatial prioritization.

Although many researches focus on exploring underlying dynamics through MAS modeling, fewer concern its computation performance [19, 20, 21]. However, computation performance is a significant aspect for large-scale simulations where obstructions from heavy calculation, intensive memory access and communication are inevitable. In general, approaches to improve MAS performance include parallel computing [22, 23, 24] and distributed computing [25, 26, 27] on both CPU and GPU platforms. Well known of parallel and distributed platforms contain Mason [28], Gama [29] and Symphony [30]. Meanwhile, FPGAs is remarked with fine-grained parallelism and flexible memory architecture. Some studies also focus on the acceleration on FPGA platforms [31, 32]. As communication patterns among agents are continuously changing on distributed systems, some works seek for effective agent allocation strategies to improve the performance [33, 34]. On the other hand, massive-scale simulations in serial algorithm often suffers from poor data locality, but seldom researches about this issue is established. Willem et al. [35] introduce a sorting phase of population and optimize data structure to improve system performance.

In this work, we study optimizing MAS for COVID-19 transmission in Shenzhen. Two methods are explored and evaluated including algorithm and data structure optimization. In order to improve data locality, we reconstruct the loop order of MAS algorithm, and propose a hierarchy Hash table structure leveraging cache and hardware properties. The paper is organized as follows: First, major optimization methods are elaborated. Third, we implement experiments and compare their performances. In addition, parallel computation with a share-memory approach is also considered. Finally, results and discussions are collected and presented.

Methods

We propose two methods to optimize MAS, including algorithm and hash table optimization. First, the loop order of MAS algorithm is reconstructed to improve data locality, in which data reutilization is enhanced. Second, hash table structure is modified leveraging cache properties and SIMD instructions.

Algorithm Optimization

In general, MAS algorithm for infectious diseases makes efforts to mimic its natural transmission process among population, where diseases spread as time evolves. Hence, traditional MAS algorithm is evolution-focused, where the evolution is represented calculation by time step, and is shown in Algorithm 1. It contains 3-order loops. The first loop ($k \in \text{MaxIter}$) is the time-evolving loop, where *MaxIter* is simulated max time step. The second loop is made up by three subprocesses (two of $a_i(k) \in A(k)$, and one of $b_j(k) \in B(k)$), including removing recovered agents, building up contaction networks, implementing diseases spreading calculation. A , B organized as hash table format denote infected agents and contact network respectively. *Regions* represents general venues population gather together including both private and public areas. The innerest loops ($g_i \in \text{Regions}(a_i)$ and $l_j \in b_j(k)$) scan over infected agents' appearance places daily to find contacting population and contacted candidates to decide increasingly infected agents at current time step k . These agent infomation are dynamically changed with either shrinking or expading behaviors in hash tables A and B during the iteration.

Algorithm 1 Evolution-focused MAS for COVID-19 transmission

Input: $A, B, \text{MaxIter}, \text{Regions}$.
Initialize: $A(0) \leftarrow \text{InitializeSeedAgents}()$

```

1: while  $k < \text{MaxIter}$  do
2:   for  $a_i(k) \in A(k)$  do
3:      $r_i \leftarrow \text{GetRecoverDate}(a_i)$ 
4:     if  $k > r_i$  then
5:        $A(k) \leftarrow \text{DeleteFromA}(a_i)$ 
6:   for  $a_i(k) \in A(k)$  do
7:     for  $g_i \in \text{Regions}(a_i)$  do
8:        $n_i \leftarrow \text{GetNumSampled}(g_i, a_i)$ 
9:        $l_i \leftarrow \text{SampleContactAgents}(g_i, n_i)$ 
10:       $b_i(k) \leftarrow [b_i(k), l_i]$ 
11:       $B(k) \leftarrow \text{InsertToB}(b_i(k))$ 
12:   for  $b_j(k) \in B(k)$  do
13:     for  $l_j \in b_j(k)$  do
14:        $\text{Prob}_j \leftarrow \text{GetInfectedProbability}(l_j)$ 
15:       if  $\text{rand} < \text{Prob}_j$  then
16:          $a_j(k+1) \leftarrow \text{GetDisease}(l_j)$ 
17:          $A(k+1) \leftarrow \text{InsertToA}(a_j(k+1))$ 
18:    $k += 1$ ;

```

Algorithm 2 Case-focused MAS for COVID-19 transmission

Input: $A, Regions$.

Initialize: $A \leftarrow InitializeSeedAgents();$
 $MaxCase \leftarrow GetSize(A);$

```

1: while  $m < MaxCase$  do
2:    $a_m \leftarrow A(m)$ 
3:    $[T_{m0}, T_{m1}] \leftarrow GetInfectiousDate(a_m)$ 
4:   for  $g_i \in Regions(a_m)$  do
5:     for  $t_m \in [T_{m0}, T_{m1}]$  do
6:        $n_m \leftarrow GetNumSampled(g_m, a_m)$ 
7:        $l_m \leftarrow SampleContactAgents(g_m, n_m)$ 
8:        $b_m \leftarrow [b_m, l_m]$ 
9:     for  $l \in b_m$  do
10:       $Prob_l \leftarrow GetInfectedProbability(l)$ 
11:      if  $rand < Prob_l$  then
12:         $a_l \leftarrow GetDisease(l)$ 
13:         $A \leftarrow InsertToA(a_l)$ 
14:    $A \leftarrow DeleteFromA(a_m)$ 
15:    $MaxCase \leftarrow GetSize(A)$ 

```

As indicated by middle-order loops in this algorithm, all the elements from the constructed hash tables A and B require accessing and utilized once for each time iteration. As the amount of population are large, such a scanning mechanism wastes data fetching from memories. It will degrade system performance severely owing to this poor locality as data arrangement heavily exceeds cache capacities. For simulation in megacity with ones of millions population, this issue is inevitable in performing real-time simulation and quickly response for intervention strategies. Key point is to manage effective reutilization of data as fetching from hash tables to maintain the locality.

In order to improve data locality, we propose a case-focused method, where the algorithm loops are reorganized. As shown in Algorithm 2, the table scanning loop is elevated to the outer order, while time-evolution loop is demoted. $MaxCase$ is size of A , and changes automatically as A varies during simulation. At a primitive transmission stage of pandemic diseases, rapid spreading among population increases $MaxCase$ prominently. It will get decreased and vanished when diseases is suppressed and immunity gets common. In the reconstructed loops, infected agent a_m get accessed and transfered to cache once for each iteration. The time evolution is implemented in the inner loop $[T_{m0}, T_{m1}]$, thus a_m can be reutilized $T_{m1} - T_{m0}$ times. Meanwhile, as infection period of a_m finishes at the end of each agent m , m gets recovered naturally and we need not to query for recovered agents appearing at evolution-focused method. In addition, contacting candidates b_m are sampled independently for each infected agent a_m , and construction of B is independently. As bulky information of agents will be transferred into higher-level cache and frequently fetching from main memories are suppressed, it is expected that case-focused method will improve system performance significantly.

It should be mentioned that these two algorithms have similar computation complexity possessing 3-order loops. An interesting difference between them is causality. The evolution-focused method follows a natural process, while case-focused method reduces priority of time evolution to maintain data locality. It will affect produced distribution of infected-agent evolution. For widely spreading diseases with large overlapping susceptible population from infectious agents, the probability of a candidate b_i gets infected is boosted by both of the amount of its contacting infectious

agents and their infectious periods. The calculation order of the two factors make different effects, where the inner loop of factor takes priorities. For each time step k in evolution-focused MAS, number of contacting agents take the priority to determine candidate's healthy status. However, in case-focused MAS, each candidate agent m is dominated by infectious agent's period $[T_{m0}, T_{m1}]$. Both of them share a same infectious probability, but candidate b_m is more likely to be infected within a relatively delayed time with respect to evolution-focused method.

We formulate this difference as the systematic error from agent-focused method. It can be demonstrated and modified by a simply assumption. Assuming the delayed infected time δ follows a uniform distribution, it can be directly derived from statistic theory that the average delayed time $\bar{\delta}$ obeys a gaussian distribution $N(\mu, \sigma)$. μ denotes average delayed time and σ is its deviation, which can be regarded as super parameters tuned for real simulation. Conversion between two algorithms can be implemented by convoluting former method with a delayed-gaussian function. In other words, agent-focused MAS will generate delayed and vague results with respect to traditional method.

Hash Table Optimization

As indicated in algorithms 1 and 2, hash table is an important participant in the simulation, and its performance affects system significantly. In general, hash tables consume the majority of cycles on many key applications such as databases [36], networking [37] and genomics [38]. But they suffer from inefficiencies in current systems owing to poor core utilization and poor spatial locality [39]. Hash tables spread key-value pairs uniformly and sparsely across allocated memory to reduce mapping conflicts. In the MAS simulation, each agent's index and its bulky information are formed together into the key-value pair filled in the table as shown in Fig.1 A. Due to sampling for agents is a random process during the simulation, same-line neighbors of frequently accessed agents may be rarely accessed. This leads to a significant waste of cache capacity.

There are many researches focus on reducing hash table overheads. Data-level parallelism of inter keys is introduced in prior works to optimize the throughput [40, 36]. Near-memory [41] and near-storage [42, 43] acceleration bypass the cache hierarchy entirely to avoid spatial locality problems. Exploiting hierarchical memory layout and characteristics of caches [39] is found improving spatial locality prominently.

In the MAS computation, the key-value pair of agent information is made up by agent index (key, k) and bulky data (value, d) derived from simulation, such as infected and recover date. These pairs stored by traditional (software) hash tables are allocated sparsely across memory as demonstrated in Fig. 1 A. Lookup and update operators of hash table have to access bulk memory in units $k + d$, which generate larger memory occupancy and heavier accessing burden. Hence, cross-line and misses of caches are frequently confronted. It is essential to allocate hash table elements across memory into a dense alignment for better spatial locality.

We reorganize hash table as a hierarchical structure and leverage caches to optimize spatial locality. The agent table constructed during the simulation is organized as three-level hierarchical tables in Fig. 1 B. This hierarchical hash table (HHT) is

composed with three parts: Cacheline Hash Table (CHT), Software Hash Table (SHT) and Bulk-Info Table (BIT). Agent indices and bulky data are stored separately, while extra projection index i is introduced to connect them. CHT and SHT store (k, i) pairs, in which i indicates address offset to locate d . BIT is a compact array where agents' information data ds are aligned contiguously and accessed by i transferred from former hash tables. Although projection indices possess extra memory, compact alignment of agent information data saves prominent memory occupancy. In addition, MAS computation queries the agent's existence frequently leaving its information alone, such a key-value separating architecture is expected to save memory occupancies and boosts key-only lookup speed.

CHT leverages the characteristics of cache to accelerate hash operations. Data is transferred in fixed block size between cache and main memory, namely cache line. In nowadays processors, a typical cache line size is 64-bytes. We construct cache line as the basic CHT element, where hash values derived from keys are located to address of cache line. In a line, data-level parallelism is implemented, where first 32 bytes make up 8-key group and the rest is corresponding index group. As elements within a cache line share the same hash value, collision occurs when lookup CHT. We use SIMD instructions to handle this issue and accelerate lookup and update in the line. SHT is used as a victim table, which holds pairs overflowing from CHT. It has a traditional structure and occupies subtle space with respect to CHT to maintain CHT priority.

Major interfaces of HHT contains three hash-table operations: find, insert and delete. Find operator decomposes key-value querying request into two parts. CHT has the priority to search for the key. If CHT's lookup is not resolved, i.e., the key is not found and the line is full, it continues searching across SHT. Key searching operation returns corresponding projection address i of k . If i is meaningful and bulky data is required, specific values are extracted as located at i in BIT. For implementing insert and delete operations, two extra labels are introduced representing their states. Insert operator fills content at empty seat of the line in CHT, and turns to SHT if an cache-line overflow occurs. Storage of the bulky value makes continuously increment at the end of BIT. Moreover, delete operation is achieved by replacing target content with the delete label. As a result, CHT and BIT make dense alignment for key-value storages and can reduce overhead for accessing cache.

Results and Discussion

The megacity Shenzhen with population larger than ten million is adopted as a study case. We implement the optimization methods on spatial explicit MAS system without NPIs proposed by Yin [17]. In the system, 11.2 million agents with demographic characteristics are synthesized in assistance of cross-referencing census data and house-hold travel survey [44]. Agents' hourly movements are formulated from mobile phone trajectory records or house-hold travel survey. Each individual is anchored onto different types of buildings for a daily trajectory, including living, working, studying and performing other activities. The synthesized agents are modeled to contact with each other when staying at the same location within one hour.

We perform the simulation on an Intel Core i9-10850K CPU platform with 10 physical cores and 20 threads. It is configured with 64 GB main memory, and cooperates with 32 KB L1 DCache, 256 KB L2 Cache and 20 MB L3 Cache. As agent-focused MAS is convenient to be implemented in data-parallel manner, we perform multi-thread computation in assistant of C++11's **thread**, **mutex** and **atomic** libraries. **thread** is used to create and detach threads for agent-parallel processing. **mutex** and **atomic** help to avoid changing memory values simultaneously among threads.

First, performances between Algorithm 1 and Algorithm 2 are present on single-thread mode. As shown in Tab. 1, both of these methods generate almost same amount of total infected agents. But their time evolution diverges as demonstrated in Fig. 2. Convolution with gaussian function is implemented to eliminate divergences, and parameters are tuned as $N(6, 13)$. It results that average delayed time is 6 days with 13-day deviation. In the statistic perspective, agent-focused MAS generates delayed and vague results with time resolution ~ 13 days.

As mentioned above, agent-focused iteration has potential to significantly improve system performance by maintaining data locality. We use **perf** provided by Linux to evaluate cache accesses during simulation. As shown in Tab. 1, results indicate that almost 94% cache references are saved and cache misses are reduced by 90%. This optimized algorithm accelerates simulation prominently and achieves $\times 4.4$ speedup. As a result, it maintains the data locality well and improves calculation speed significantly, but sacrificing part of precision.

Next, proposed hash table HHT is implemented for further optimizing system performance under the agent-focused method. As SHT is provided by system and holds victims overflowing from CHT, frequently accessing to SHT may affect HHT's querying performance. We set its member occupancies is 1% to CHT capacity to minimize the influence. Algorithm 1 with libstdc++'s C++11 unordered_map under single-thread operation is adopted as benchmark. The single-thread computation is shown in the first ticks along thread axis of Fig. 3. HHT achieves $\times 6.4$ speedup which leverages both algorithm and hash table optimization. Comparing the result with unordered_map implemented under the same MAS algorithm, HHT is still faster by 47%.

Traditional hash table is a thread-unsafe data mapping paradigm. It requires multiple memory duplication and migration during the table expands or shrinks. As hash **insert** and **find** encounter from different threads, while the **insert** causes re-hashing and expands the table's capacity by factor 2, another thread may lose the actual access to target and a program fault occurs. We use atomic operator on HHT to maintain a thread-safe manner across multi-thread operations. As demonstrated by Fig. 3, this parallel implementation prominently boosts the program, where original 40-min process is compressed within ~ 30 s. Consequently, parallel acceleration with 20 threads achieves $\times 81$ speedup to the benchmark, and processor efficiency is derived roughly 64% accordingly.

Conclusions

The MAS is an essential method for studying epidemiology sciences, especially for intervention for COVID-19 transmission. Traditional methods needs querying over

population as time evolves and suffers from poor-locality issues. In this work, we focus on optimizing MAS for infectious diseases, and propose two successive processes to accelerate computation. First, we reconstruct the iteration order of MAS and propose the agent-focused method. It can improve data locality significantly, where 90% cache references are saved. Consequently, the program is accelerated by $\times 4.3$. Next, we design a thread-safe and high performance hash table HHT for managing intermediate products from simulation. HHT leverage cache characteristics and SIMD intructions to optimize hash operations. It can further get a faster speed by 47% comparing with classical hash table, and a 20-thread implementation achieves $\times 81$ speedup finally.

It should be noticed that major MAS researches adopted evolution-focused method as simulation backbone [45, 46, 47]. Our proposed agent-focused method overcomes their disadvantages with poor data locality, but takes a trade-off with blurred and delayed results. Although the results can be recovered by deconvoluting from a mixing kernel, kernel parameters need pre-calibrated under a specific scenario. Carefully studies for this mechanism will be implemented in next works. Moreover, NPI and PI methods are crucial for intervenning spreading of infectious diseases. Studies of fast calculation with these interventions is left in the future.

Acknowledgements

Not applicable

Funding

National Key Research and Development Program of China (Grant No. 2020YFB0204802)

Abbreviations

MAS: Multi-agent Simulation; COVID-19: Coronavirus Disease 2019; HHT: Hierarchical Hash Table; CHT: Cacheline Hash Table; SHT: Software Hash Table; BIT: Bulk-Info Table; CPU: Central Processing Unit; GPU: Graphics Processing Unit; FPGA: Field Programmable Gate Array

Availability of data and materials

Text for this section. . .

Ethics approval and consent to participate

Not applicable

Competing interests

The authors declare that they have no competing interests.

Consent for publication

Not applicable

Authors' contributions

C.J. contributed this study and wrote the manuscript. C.J., H.Z. and L.Y. developed the model. Y.Z. and S.F. revised this manuscript. All authors read and approved the manuscript.

Authors' information

Not applicable

Author details

¹National Supercomputing Center in Shenzhen, Shenzhen 518055, Guangdong, People's Republic of China.

²Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, Guangdong, People's Republic of China. ³University of Chinese Academy of Sciences, , Beijing 100049, People's Republic of China.

References

1. Sansores, C., Pavón, J.: Agent-based simulation replication: A model driven architecture approach. In: Mexican International Conference on Artificial Intelligence, pp. 244–253 (2005). Springer
2. Davidsson, P.: Agent based social simulation: A computer science view. *Journal of artificial societies and social simulation* 5(1) (2002)
3. Wolf, S., Fürst, S., Mandel, A., Lass, W., Lincke, D., Pablo-Marti, F., Jaeger, C.: A multi-agent model of several economic regions. *Environmental modelling & software* 44, 25–43 (2013)
4. Chen, D.: Modeling the spread of infectious diseases: A review. *Analyzing and modeling spatial and temporal dynamics of infectious diseases*, 19–42 (2014)

5. Domínguez, R., Cannella, S., Framinan, J.M.: Scope: a multi-agent system tool for supply chain network analysis. In: IEEE EUROCON 2015-International Conference on Computer as a Tool (EUROCON), pp. 1–5 (2015). IEEE
6. McArthur, S.D., Davidson, E.M., Catterson, V.M., Dimeas, A.L., Hatziaargyriou, N.D., Ponci, F., Funabashi, T.: Multi-agent systems for power engineering applications—part i: Concepts, approaches, and technical challenges. *IEEE Transactions on Power systems* **22**(4), 1743–1752 (2007)
7. Wangapisit, O., Taniguchi, E., Teo, J.S., Qureshi, A.G.: Multi-agent systems modelling for evaluating joint delivery systems. *Procedia-Social and Behavioral Sciences* **125**, 472–483 (2014)
8. Dorri, A., Kanhere, S.S., Jurdak, R.: Multi-agent systems: A survey. *Ieee Access* **6**, 28573–28593 (2018)
9. Kermack, W.O., McKendrick, A.G.: A contribution to the mathematical theory of epidemics. *Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character* **115**(772), 700–721 (1927)
10. Bailey, N.T.: The mathematical theory of infectious diseases and its applications. In: *The Mathematical Theory of Infectious Diseases and Its Applications*, pp. 413–413 (1975)
11. Dion, E., VanSchalkwyk, L., Lambin, E.F.: The landscape epidemiology of foot-and-mouth disease in south africa: A spatially explicit multi-agent simulation. *Ecological Modelling* **222**(13), 2059–2072 (2011)
12. Yergens, D., Hiner, J., Denzinger, J., Noseworthy, T.: Multiagent simulation system for rapidly developing infectious disease models in developing countries. In: *Proceedings of the 2nd International Workshop on Multi-Agent Systems for Medicine and Computational Biology*, Hakodate, Japan, pp. 104–116 (2006)
13. Castro, B.M., de Melo, Y.d.A., Dos Santos, N.F., da Costa Barcellos, A.L., Choren, R., Salles, R.M.: Multi-agent simulation model for the evaluation of covid-19 transmission. *Computers in Biology and Medicine* **136**, 104645 (2021)
14. Vyklyuk, Y., Manylich, M., Škoda, M., Radovanović, M.M., Petrović, M.D.: Modeling and analysis of different scenarios for the spread of covid-19 by using the modified multi-agent systems—evidence from the selected countries. *Results in Physics* **20**, 103662 (2021)
15. Nanna, G.A., Quatraro, N.F., De Carolis, B.: A multi-agent system for simulating the spread of a contagious disease. In: *WOA*, vol. 1613, p. 119 (2020)
16. Lorig, F., Johansson, E., Davidsson, P.: Agent-based social simulation of the covid-19 pandemic: A systematic review. *JASSS: Journal of Artificial Societies and Social Simulation* **24**(3) (2021)
17. Yin, L., Zhang, H., Li, Y., Liu, K., Chen, T., Luo, W., Lai, S., Li, Y., Tang, X., Ning, L., et al.: A data driven agent-based model that recommends non-pharmaceutical interventions to suppress coronavirus disease 2019 resurgence in megacities. *Journal of the Royal Society Interface* **18**(181), 20210112 (2021)
18. Zhou, S., Zhou, S., Zheng, Z., Lu, J.: Optimizing spatial allocation of covid-19 vaccine by agent-based spatiotemporal simulations. *GeoHealth* **5**(6), 2021–000427 (2021)
19. Eubank, S., Guclu, H., Kumar, V.A., Marathe, M.V., Srinivasan, A., Toroczkai, Z., Wang, N.: Modelling disease outbreaks in realistic urban social networks. *Nature* **429**(6988), 180–184 (2004)
20. Ferguson, N.M., Cummings, D.A., Cauchemez, S., Fraser, C., Riley, S., Meeyai, A., Iamsirithaworn, S., Burke, D.S.: Strategies for containing an emerging influenza pandemic in southeast asia. *Nature* **437**(7056), 209–214 (2005)
21. Germann, T.C., Kadau, K., Longini, I.M., Macken, C.A.: Mitigation strategies for pandemic influenza in the united states. *Proceedings of the National Academy of Sciences* **103**(15), 5935–5940 (2006)
22. Barceló, J., Ferrer, J.L., García, D., Florian, M., Le Saux, E.: Parallelization of microscopic traffic simulation for att systems analysis. In: *Equilibrium and Advanced Transportation Modelling*, pp. 1–26. Springer, ??? (1998)
23. Aydt, H., Xu, Y., Lees, M., Knoll, A.: A multi-threaded execution model for the agent-based semsim traffic simulation. In: *Asian Simulation Conference*, pp. 1–12 (2013). Springer
24. Saprykin, A., Chokani, N., Abhari, R.S.: Large-scale multi-agent mobility simulations on a gpu: towards high performance and scalability. *Procedia Computer Science* **151**, 733–738 (2019)
25. Rao, D.M.: Accelerating parallel agent-based epidemiological simulations. In: *Proceedings of the 2nd ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, pp. 127–138 (2014)
26. Cameron, G.D., Duncan, G.I.: Paramics—parallel microscopic simulation of road traffic. *The Journal of Supercomputing* **10**(1), 25–53 (1996)
27. Rickert, M., Nagel, K.: Dynamic traffic assignment on parallel computers in transims. *Future generation computer systems* **17**(5), 637–648 (2001)
28. Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K.: Mason: A new multi-agent simulation toolkit. In: *Proceedings of the 2004 Swarmfest Workshop*, vol. 8, pp. 316–327 (2004). Michigan, USA
29. Amouroux, E., Chu, T.-Q., Boucher, A., Drogoul, A.: Gama: an environment for implementing and running spatially explicit multi-agent simulations. In: *Pacific Rim International Conference on Multi-Agents*, pp. 359–371 (2007). Springer
30. North, M.J., Collier, N.T., Ozik, J., Tatara, E.R., Macal, C.M., Bragen, M., Sydelko, P.: Complex adaptive systems modeling with repast symphony. *Complex adaptive systems modeling* **1**(1), 1–26 (2013)
31. Cui, L., Chen, J., Hu, Y., Xiong, J., Feng, Z., He, L.: Acceleration of multi-agent simulation on fpgas. In: *2011 21st International Conference on Field Programmable Logic and Applications*, pp. 470–473 (2011). IEEE
32. Zhou, X., Fu, W.: A multi-agent simulation method of urban land layout structure based on fpga. *Mobile Networks and Applications* **25**(4), 1572–1581 (2020)
33. Jang, M.-W., Agha, G.: Dynamic agent allocation for large-scale multi-agent applications. *A PARAMETRIC MODEL FOR LARGE SCALE AGENT SYSTEMS*, 252 (2005)
34. Jang, M.-W., Agha, G.: Adaptive agent allocation for massively multi-agent applications. In: *International Workshop on Massively Multiagent Systems*, pp. 25–39 (2004). Springer
35. Willem, L., Stijven, S., Tijssens, E., Beutels, P., Hens, N., Broeckhove, J.: Optimizing agent-based transmission models for infectious diseases. *BMC bioinformatics* **16**(1), 1–10 (2015)
36. Kocberber, O., Grot, B., Picorel, J., Falsafi, B., Lim, K., Ranganathan, P.: Meet the walkers accelerating index traversals for in-memory databases. In: *2013 46th Annual IEEE/ACM International Symposium on*

- Microarchitecture (MICRO), pp. 468–479 (2013). IEEE
37. Song, H., Dharmapurikar, S., Turner, J., Lockwood, J.: Fast hash table lookup using extended bloom filter: an aid to network processing. *ACM SIGCOMM Computer Communication Review* **35**(4), 181–192 (2005)
 38. Melsted, P., Pritchard, J.K.: Efficient counting of k-mers in dna sequences using a bloom filter. *BMC bioinformatics* **12**(1), 1–7 (2011)
 39. Zhang, G., Sanchez, D.: Leveraging caches to accelerate hash tables and memoization. In: *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 440–452 (2019)
 40. Hayes, T., Palomar, O., Unsal, O., Cristal, A., Valero, M.: Vector extensions for decision support dbms acceleration. In: *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 166–176 (2012). IEEE
 41. Lloyd, S., Gokhale, M.: Near memory key/value lookup acceleration. In: *Proceedings of the International Symposium on Memory Systems*, pp. 26–33 (2017)
 42. Tanaka, S., Kozyrakis, C.: High performance hardware-accelerated flash key-value store. In: *The 2014 Non-volatile Memories Workshop (NVMW)* (2014)
 43. Xu, S., et al.: Bluecache: A scalable distributed flash-based key-value store. PhD thesis, Massachusetts Institute of Technology (2016)
 44. Zhao, Z., Shaw, S.-L., Yin, L., Fang, Z., Yang, X., Zhang, F., Wu, S.: The effect of temporal sampling intervals on typical human mobility indicators obtained from mobile phone location data. *International Journal of Geographical Information Science* **33**(7), 1471–1495 (2019)
 45. Cuevas, E.: An agent-based model to evaluate the covid-19 transmission risks in facilities. *Computers in biology and medicine* **121**, 103827 (2020)
 46. Jalayer, M., Orsenigo, C., Vercellis, C.: Cov-abm: A stochastic discrete-event agent-based framework to simulate spatiotemporal dynamics of covid-19. *arXiv preprint arXiv:2007.13231* (2020)
 47. Kou, L., Wang, X., Li, Y., Guo, X., Zhang, H.: A multi-scale agent-based model of infectious disease transmission to assess the impact of vaccination and non-pharmaceutical interventions: The covid-19 case. *Journal of Safety Science and Resilience* **2**(4), 199–207 (2021)

Figures

Figure 1: A: Traditional (software) hash table format. B: Overview of hierarchical hash table format.

Figure 2: Evolutions of increasing cases per day among different methods. Blue solid line is calculated by evolution-focused MAS. Red solid line is calculated by case-focused MAS. Purple dashed line is the result from convoluting evolution-focused MAS outputs with delayed-gaussian function $N(6, 13)$.

Figure 3: Multi-thread speedup of HHT implementation on the agent-focused MAS comparing to benchmark (unordered.map implementation on the evolution-focused MAS).

Tables

Table 1: Comparison between Evolution-focused MAS and Agent-focused MAS on single-thread mode

	Evolution-focused MAS	Agent-focused MAS
Total Infected Agents	84%	84%
Cache References	2.5×10^{11}	1.5×10^{10}
Cache Misses	8.5×10^{10}	8×10^9
Running Time (s)	2600	600
Speedup	$\times 1$	$\times 4.3$

Figures

Figure 1

A: Traditional (software) hash table format. B: Overview of hierachical hash table format.

Figure 2

Evolutions of increasing cases per day among different methods. Blue solid line is calculated by evolution-focused MAS. Red solid line is calculated by case-focused MAS. Purple dashed line is the result from convoluting evolution-

focused MAS outputs with delayed-gaussian function $N(6; 13)$.

Figure 3

Multi-thread speedup of HHT implementation on the agent-focused MAS comparing to benchmark (unordered map implementation on the evolution-focused MAS).