On Efficient Single-Core Execution of Agent-Based Epidemiological Models with Contact-Tracing Transmission

Vladyslav Sarnatskyi[a] *and* Igor Baklan[a]

[a] *National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", 37, Prosp. Peremohy, Kyiv, 03056, Ukraine*

Abstract

In this work, we present our research on efficiency of single-core execution of agent-based epidemiological models with contact-tracing transmission. We performed an analysis of existing epidemiological agent-based modeling tools from the perspective of their performance, which reflects computational time needed to perform a single simulation step of a model with fixed number of agents.

We developed several simulation algorithms, based on different model types, and showed some optimizations to maximize the performance of them.

We designed a metric to compare simulation step execution times on a single CPU core and used to estimate a performance of underlying simulation algorithms in the existing methods with developed one. Even though, as it will be discussed below, it is very difficult to estimate an execution time of some algorithm without actual access to it, we present the results of our method on both modern and old CPU.

Keywords 1

Modeling, epidemiology, agent-based models, individual-based models, IBMs, infectious disease spread modeling, performance, single-thread, optimization

1. Introduction

Global epidemilogical events can make significant damage to global economy [1]. In order to reduce both epidemiological and economic impact, active measures must be considered. They include, but not limited to full or partial quarantine, obligatory face mask regime, vaccinations etc. However, even though some measures like full quarantine can significantly lower disease spread rate by flattening the curve of new cases, economic impact can be unbearable for certain communities. This is why a good balance between epidemiological and economic risks must be hold. But, determining the exact threshold of disease spread countermeasures severity is problematic without modeling particular epidemics.

First research in the field of infectious disease spread modeling is dated early XX century. In their work, Anderson McKendrick and Willian Kermack [2] considered modeling an epidemic by splitting the population into several compartments with different stages of a disease. These stages can be, for example: (S)usceptible — individuals, who can be infected in the future, (I)nfectious — individuals, who were infected, show symptoms and can infect others. After assigning the initial number of individuals in each compartment, the dynamics of its change can be modeled as a system of differential equations.

Work of McKendrick and Kermack served as the foundation for the research and development of compartmental epidemiological models, which are used these days to model various diseases like dengue fever, COVID-19, etc [3-11].

[1]The Sixth International Workshop on Computer Modeling and Intelligent Systems (CMIS-2023), May 3, 2023, Zaporizhzhia, Ukraine
EMAIL: vladysarn@protonmail.com (V. Sarnatskyi); iaa@ukr.net (I. Baklan)
ORCID: 0000−0001−5231−6136 (V. Sarnatskyi); 0000−0002−5274−5261 (I. Baklan)

With the invention of computers and increase in available computation power, another type of epidemiological model appeared — agent- or individual-based. This type bears some resemblance to compartmental by considering the number of individuals in each compartment. But, instead of modeling it via equations, each individual is modeled separately. It means, that each individual is a separate entity, which can move inside an environment and spread the disease by contacting other individuals. Such significant increase in model granularity allows the users to model complex stochastic behaviors, common for the real world. But, on the other hand, required computational resources, needed to run the model, also increased rapidly. This can be explained easily by the following example: in a community of individuals, which can be considered a medium-sized model, a single simulation step requires computation of agent-to-agent contact events.

In this work, we explore the possibilities to optimize both time and space algorithmic complexity of simulation algorithm.

2. Problem statement

The present study aims to address the challenges of infectious disease modeling through the development of a mathematical framework for a generalized agent-based epidemiological model. As opposed to compartmental epidemiological modeling, agent-based considers modeling population as a set of distinct entities – agents. These agents can perform some activities and interact with each other. This agent-to-agent interaction is the main way of infection disease transmission. However, analysis of it requires iterating through all of agents pairs, making its algorithmic complexity quadratic, which is a problem for a large models with significant number of agents. For example, single simulation of FluTE model with number of agents equal to the population of United States of America takes about 6 hours to run on a cluster with 32x CPUs [12]. The ultimate goal of this research is to enhance the computational efficiency of this process. Specifically, the study has the following objectives:

Firstly, we aim to formulate a mathematical apparatus for a general agent-based epidemiological model. To achieve this, we will identify critical factors that influence disease transmission and develop a mathematical framework that accurately captures these dynamics.

Secondly, we aim to explore ways to optimize the complexity of the simulation algorithm, which can be computationally expensive, especially when dealing with large populations. The optimization of the simulation algorithm will involve identifying ways to reduce its computational complexity while maintaining its accuracy.

Thirdly, we aim to compare the performance of the model, based on optimized algorithm with previously developed models. By conducting a comparative analysis, we aim to demonstrate the effectiveness of the proposed optimization strategies in enhancing the efficiency of simulations. We will use standard evaluation metrics, such as the time required for simulation, to compare the performance of the optimized model with that of existing models.

3. Literature review

In this section, we describe our findings on existing tools and methods of agent-based epidemiological modeling. These tools can be divided into three distinct groups:

- Programming language modules/libraries;
- General-purpose modeling environments;
- Specialized software for epidemiological modeling.

We review each group separately.

3.1. Programming language modules/libraries

Swarm [13] is a set of libraries for Java programming languages which defines a model as a set of interacting agents, dynamics of their evolution and the schedule of events. This package is no longer maintained and has its successor — Ascape [14], which was simplified from the perspective of

programming interface. Ascape also delivers a simple graphical interface for model adjustments and inspection and spreadsheet data export.

3.2. General-purpose modeling environments

NetLogo [15] is modeling environment, consisting of the following elements:

- Subject-oriented programming language, used for describing agents behavior, their inner states evolution and interactions between each other;
- Tool set for the analysis of experiments data as the result of modeling.

Overall, NetLogo allows one to flexibly define models, but this flexibility comes with certain limitations. First of all, as showed in [16], the overall performance of NetLogo simulation engine is significantly lower, compared to similar models implemented using other tools. NetLogo features fixed modeling space geometry in the form of rectangular grid, where each agent is ``attached'' to a certain cell of it and can interact with agent on adjacent cells. Despite the interaction of agents positioned far from each other can be implemented, it is not supported internally, can be cumbersome and limits computational performance of a simulation.

RePast [17] is a software tool set, designed for modeling of complex systems. There are several implementations of it, based on different programming languages such as Java, C++. Repast also features an implementation, suitable for high-parallel environments [18].

3.3. Specialized software for epidemiological modeling

As each model with software implementation can be viewed as specialized software for epidemiological modeling, we review only those, offering some amount of configurability.

A Framework for Reconstructing Epidemic Dynamics (FRED) [19] is a software for agent-based epidemiological modeling, which uses a synthetic population database to represent every individual in a specific geographic region. The database contains geographically located synthetic households, group quarters, schools, and workplaces that reflect the actual spatial distribution of the area. Each agent has associated demographic and socioeconomic information and locations for their activities. The synthetic population closely matches the available census data for the United States with high spatial resolution. The model runs a discrete-time simulation with a time step of one day, allowing agents to interact with other agents at their shared activity locations, potentially transmitting diseases. The simulation records each infection transmission event to evaluate control measures and their impact on sub-populations. Agents can dynamically alter their daily activities. The fixed simulation step of 1 day allows for performance optimizations and is not a severe limitation for diseases with long latency periods, but it may be a limitation for diseases with short latency and infectious periods or short-term interventions. FRED was originally developed to study influenza, but it can be customized to investigate other infectious diseases, such as measles, by making adjustments to the configuration files that describe the natural history of the disease.

Unfortunately, to our knowledge, standalone software for agent-based epidemiological modeling with wide range of configurability options similar to GLEaMviz [20] doesn't exist.

4. Method

4.1. General notation

In this subsection, we describe a general mathematical apparatus of agent behavior within an epidemiological model with contact-tracing transmission.

Simulation is done in discrete steps, which can be, for example, days. These steps form the simulation step set $T = 1 \ldots N_T$.

Agents, representing individuals, who are somehow related to a modeled disease (can be infected, serve as vectors etc.) create the finite set A. Each agent has its schedule, dictating the geographical position of it. This position shouldn't be though of as coordinates in some space, as it would limit the

model, but rather an occupancy of some entity, for example the household or the school. In order to model the change of agents' location during the day, we split each simulation step into a finite number of smaller time steps from the set $H = 1 \dots N_H$. Thus, the schedule function can be defined as:

$$\phi: A \times T \times H \to P, \tag{1}$$

where P is a finite set of possible agents' location[2] (henceforth, we assume each function can depend on some external model parameters, even though it's not displayed in the notation).

Instead of using the compartments, agent-based models assign infection states to each agent, often mirroring respective compartments of compartmental models. This assignment can be expressed as infection state function:

$$\xi: A \times T \to S, \tag{2}$$

where S is a finite set of infection states.

Naturally, the infection state of an agent can change by two separate processes: intrinsic and extrinsic. Intrinsic one considers events, occurring inside the organism, which is modeled by the agent. These events can be driven by an immunological response or nature of the disease. We call this process *infection progress*. Extrinsic process involves inter-agent communication and majorly represents the transmission of a disease. We call this process *infection spread*.

Mathematically, these processes can be described by infection progress (Equation 3) and infection transmission (Equation 4) functions.

$$\Phi_p: A \times S \to [0,1] \tag{3}$$

$$\Phi_t: A^2 \times S \to [0,1] \tag{4}$$

For a given agent $a \in A$ and infection state $s \in S$, Φ_p describes the probability, that at next simulations step agent's a infection state will become equal to s as the result of intra-agent processes.

Similarly, for given agents $r, d \in S$ and infection state $s \in S$, Φ_t describes the probability, that at next simulation step agent's r infection state will become equal to s as the result of inter-agent communication with agent d. Discussing transmission function Φ_t, we refer to r as recipient, to d as donor.

Thus, the goal of an agent-based epidemiological model at simulation step $t \in T$ is to estimate

$$\xi(a, t+1), \forall a \in A \tag{5}$$

This can be done by evaluating the probability of each agent $a \in A$ to change its state to $s \in S, s \neq \xi(a,t)$[3], which can be expressed by the following expression:

$$\underset{t}{Pr}(\xi(a,t+1) = s) = 1 - \Big(1 - \Phi_p(a,s)\Big) \prod_{h \in H} \prod_{d \in A: \phi(a,t,h) = \phi(d,t,h)} \Big(1 - \Phi_t(a,d,s)\Big) \tag{6}$$

Given this equation, a trivial algorithm for computing $Pr_t(\xi(a,t+1) = s)$ is presented in Algorithm 1. As one may notice, this evaluation involves iteration over A, which has to be done for each agent as nested loop. Optimization of computation of this expression is the main concern of this work.

4.2. Optimizations

A significant part of literature discuss models, with transmission function independent of donor's and recipient's inner states, which can be formulated as following constraints of function Φ_t:

$$\begin{gathered} \forall a, a_1, a_2 \in A, s \in S: \xi(a_1, t) = \xi(a_2, t) \Rightarrow \\ \Phi_t(a, a_1, s) = \Phi_t(a, a_2, s) \wedge \Phi_t(a_1, a, s) = \Phi_t(a_2, a, s) \end{gathered} \tag{7}$$

This assumption can be used to perform a significant optimization. The main idea of it is to estimate «contagiousness» of each location at each time step by tracking infectious agents' movements, which is defined as:

$$\forall t \in T: I(p, h, s_1, s_2) = 1 - \prod_{a \in A, \phi(a,t,h) = p} \Big(1 - \overline{\Phi_t}(s_1, \xi(a,t), s_2)\Big), \tag{8}$$

[2]Henceforth, we assume each function can depend on some external model parameters, even though it's not displayed in the notation

[3]$P_t\big(\xi(a,t+1) = \xi(a,t)\big)$ can be computed as $1 - \sum_{s \in S, s \neq \xi(a,t)} Pr_t\,(\xi(a,t+1) = s)$

where:

$$\forall r, d \in A, s \in S, t \in T: \Phi_t(r, d, s) = \overline{\Phi_t}(\xi(r, t), \xi(d, t), s) \tag{9}$$

Algorithm 1: $\Pr_t(\xi(a, t+1) = s)$ computation algorithm (trivial)

Data: $A, H, S, \Phi_t, \Phi_p, t \in T,$
Result: $\Pr_t(\xi(a, t+1) = s)$
for $a \in A$ **do**
  for $s \in S$ **do**
    $\Pr_t(\xi(a, t+1) = s) \leftarrow \Phi_p(a, s)$
  end
  for $h \in H$ **do**
    $p \leftarrow \phi(a, t, h)$
    for $d \in A$ **do**
      if $p = \phi(d, t, h)$ **then**
        for $s \in S, s \neq \xi(a, t)$ **do**
          $\Pr_t(\xi(a, t+1) = s) \leftarrow 1 - (1 - \Pr_t(\xi(a, t+1) = s))(1 - \Phi_t(a, d, s))$
        end
      end
    end
  end
  $\Pr_t(\xi(a, t+1) = \xi(a, t)) \leftarrow 1 - \sum_{s \in S, s \neq \xi(a,t)} \Pr_t(\xi(a, t+1) = s)$
end

Figure 1: Description of a trivial algorithm for computation of $P_t(\xi(a, t+1) = s)$.

Subsequently, susceptible agents' movements are used together with these estimates to compute the probabilities of their state change:

$$\forall t \in T, \underset{t}{Pr}(\xi(a, t+1) = s) = 1 - \Big(1 - \Phi_p(a, s)\Big) \prod_{h \in H} \Big(1 - I(\phi(a, t, h), h, \xi(a, t), s)\Big) \tag{10}$$

Even though, there are no evidence that Equation 7 holds for real world, it can be slightly adjusted, so that the scope of possible use cases of this model increases. If Φ_t can be decomposed into functions Φ_{tr}, Φ_{td} so that:

$$\forall r \in A, s \in S: \prod_{d \in A} \Big(1 - \Phi_t(a, d, s)\Big) = \Phi_{tr}\left(r, \prod_{d \in A} \Big(1 - \Phi_{td}(d, s)\Big), s\right), \tag{11}$$

the Algorithm 2 can be used.

We don't try to solve this equation in the scope of this work, but the following solution is used in our experiments:

$$\begin{gathered} \forall f: A \to R_+, \Phi_{tr}(r, x, s) = x^{f(x)} \\ \forall \Phi_{td}(d, s): A \times S \to [0,1] \end{gathered} \tag{12}$$

Algorithm 2: $\Pr_t(\xi(a,t+1)=s)$ computation algorithm (linear)

Data: $A, H, S, P, \Phi_t, \Phi_p, t \in T,$
Result: $\Pr_t(\xi(a,t+1)=s)$
for $p \in P$ **do**
  for $h \in H$ **do**
    for $(s_1, s_2) \in S^2$ **do**
      $I(p,h,s_1,s_2) \leftarrow 0$
    end
  end
end
for $a \in A$ **do**
  for $h \in H$ **do**
    $p \leftarrow \phi(a,t,h)$
    for $(s_1, s_2) \in S^2$ **do**
      $I(p,h,s_1,s_2) \leftarrow 1-(1-I(p,h,s_1,s_2))(1-\Phi_t(s_1,\xi(a,t),s_2))$
    end
  end
end
for $a \in A$ **do**
  for $s \in S$ **do**
    $\Pr_t(\xi(a,t+1)=s) \leftarrow \Phi_p(a,s))$
  end
  for $h \in H$ **do**
    $p \leftarrow \phi(a,t,h)$
    for $s \in S$ **do**
      $\Pr_t(\xi(a,t+1)=s) \leftarrow 1-(1-\Pr_t(\xi(a,t+1)=s))(1-I(p,h,\xi(a,t),s))$
    end
  end
  $\Pr_t(\xi(a,t+1)=\xi(a,t)) \leftarrow 1-\sum_{s\in S, s\neq\xi(a,t)} \Pr_t(\xi(a,t+1)=s)$
end

Figure 2: Description of a linear algorithm for computation of $P_t(\xi(a,t+1)=s)$.

In a general case, when Algorithm 2 cannot be applied, there is still room for an optimization. Algorithm 1 considers iterating over all possible pairs of agents, even though most of them weren't even in contact. By iterating over pairs of agents, which are in the same contact group, the overall number of computation can be decreased. Here, a contact group is a subset of the agent set *A*, which were at the same location at a specified simulation step *t* and time step *h*:

$$\forall a_1, a_2 \in G_{t,h,i} \subseteq A: \phi(a_1,t,h) = \phi(a_2,t,h) \tag{13}$$

This computational optimization can be explained by the following:

$$\begin{aligned} \sum_{h\in H}\sum_{i}\left|G_{t,h,i}\right| &= |A\times H| \\ \sum_{h\in H}\sum_{i}\left|G_{t,h,i}\right|^2 &\leq |A\times H|^2 \end{aligned} \tag{14}$$

Algorithm 3: $\Pr_t(\xi(a,t+1)=s)$ computation algorithm (with grouping)

Data: $A, H, S, P, \Phi_t, \Phi_p, t \in T$,
Result: $\Pr_t(\xi(a,t+1)=s)$
$G \leftarrow \{\emptyset | \forall (p,h) \in P \times H\}$
for $a \in A$ **do**
  for $s \in S$ **do**
    $\Pr_t(\xi(a,t+1)=s) \leftarrow \Phi_p(a,s)$
  end
  for $h \in H$ **do**
    $p \leftarrow \phi(a,t,h)$
    $G_{p,h} \leftarrow G_{p,h} \cup \{a\}$
  end
end
for $(p,h) \in P \times H$ **do**
  for $(r,d) \in G_{p,h}, r \neq d$ **do**
    for $s \in S$ **do**
      $\Pr_t(\xi(r,t+1)=s) \leftarrow 1-(1-\Pr_t(\xi(r,t+1)=s))(1-\Phi_t(r,d,s))$
    end
  end
end
for $a \in A$ **do**
  $\Pr_t(\xi(a,t+1)=\xi(a,t)) \leftarrow 1-\sum_{s\in S, s\neq\xi(a,t)} \Pr_t(\xi(a,t+1)=s)$
end

Figure 3: Description of a grouping-based algorithm for computation of $P_t(\xi(a,t+1)=s)$.

As one may notice, the worst-case computational complexity of Algorithm 3 is $O(n^2)$, depending on the number of agents when $|P|=1$. But, when using with real-world models, it's easy to show that it's equal to $O(n)$. In the case of real world, we can assume, the number of places is proportional to the number of agents:

$$|A| \propto |P| \tag{15}$$

This can be explained by the fact, that, for example, the average number of residents of a household is independent of the total number of agents in the model[4].

Given this assumption, the model can be divided into a set of contact groups types, where i-th type has a number of associated contact groups, proportional to the number of agent's with coefficient k_i and number of agents in each group equal to x_i from some probability distribution X_i. Given this, computational complexity of Algorithm 3 can be estimated as:

$$O\left(g_T(n) + E\left[\sum_i n\, k_i x_i^2\right]\right) = O\left(g_T(n) + n\sum_i k_i\, E\left[x_i^2\right]\right) = O(g_T(n)+n), \tag{16}$$

where $g_T(n)$ is algorithmic complexity of the grouping algorithm itself.

Both space and time complexity of the grouping algorithm depends on the choice and implementation of the underlying data structure which holds values of function Φ. It must support the following operations:

- Add tuple (a,h,p) into collection $(a \in A, h \in H, p \in P)$;
- For tuple (h,p) return all a_i, so that (a_i,h,p) was added into collection before;
- Clear all data from the collection.

[4]There can be some variation for the models, differing in size by orders of magnitude. For example, rural settlement with population of 10^3 cannot be modeled by scaling down the model of a large city with 10^6 citizens — the economical and sociodemographic gap will be too significant.

Moreover, the following constraints are present:

- Overall number of tuples (a, h, p) is equal to $|H| \cdot |A|$;
- Number of unique tuples (h, p) in the collection is at most $|H| \cdot |P|$;
- For a given tuple (h, p), maximum number of a_i, so that (a_i, h, p) was added into the collection is at most $|A|$.

Such a data structure is a multimap, which is an extension of the associative array for the case of storing several values by one key. Being an abstract data type, a multimap can be implemented in several different ways, however, having a common feature: dividing the structure into two components. The first component is a data structure that maps a key to a reference to a container containing a list of values; the second is the type of this container. So, for example, the first data structure can be a hash table [21], and the second — a linked list [22]. Based on this, the computational algorithm complexity g_T can be given as:

$$\begin{gathered} g_T(n) = g_{T1}(n) + g_{T2}(n) \\ g_{T1}(n) = O\big(n \cdot ind_1(n) ins_2(n)\big) \\ g_{T2}(n) = O\big(itr_1(n) itr_2(n)\big), \end{gathered} \tag{17}$$

where ind_1 is the algorithmic complexity of searching by the index in the first data structure, ins_2 is the algorithmic complexity of inserting at the end of the second data structure, itr_1,itr_2 is the algorithmic complexity of iterating the elements of the first and second data structures, respectively.

Based on the aforementioned constraints and the fact that all keys can be numbered from 1 to $|P| \cdot |H|$, the obvious candidates for the first data structure are the static array and the list because it is valid for them: $ind_1(n)=O(1)$,$itr_1(n)=O(n)$. However, due to the use of the CPU cache, in practice the use of a static array is more efficient (Figure 4).

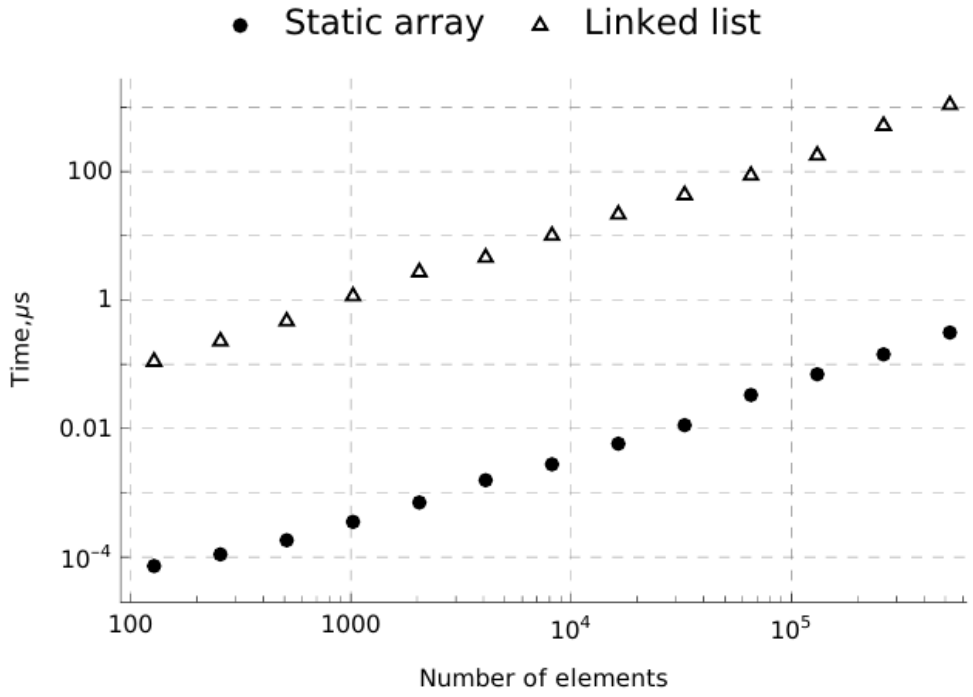

Figure 4: Comparison of container iteration speed. Cargo 1.56.0 compiler, compilation optimizations disabled

In addition to implementing a multiset using a container of containers, an approach based on hash tables can be used. In this case, it involves two data structures, the first of which maps the value of the key to the number of elements by this key stored in the structure, and the second - the key and the index of the element to its value.

This approach is appropriate, because the operations of adding and obtaining a value by key in such a structure will have an algorithmic complexity equal to $O(1)$. This is justified due to the fact that

similar operations on the hash table also have a constant complexity (in the absence of collisions) [23].

To evaluate the time performance of each approach, a number of experiments were conducted, which involved the calculation of the execution time of the grouping algorithm for each approach with different numbers of agents and the ratio of the number of agents to the number of places. The results are shown in figures 5, 6, 7.

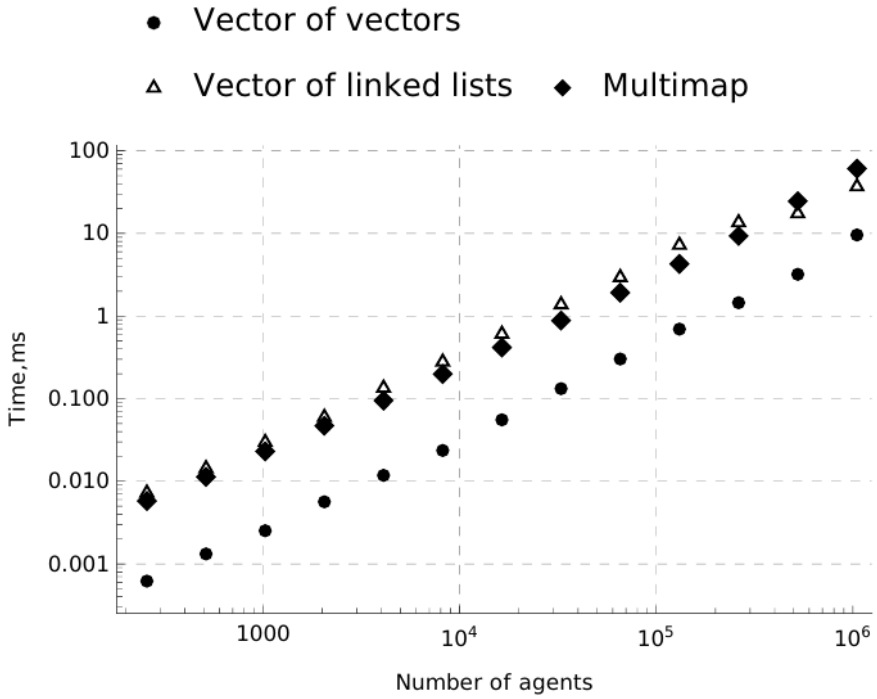

Figure 5: The execution time of the grouping algorithm for the average ratio of the number of agents to the number of places equal 16:1

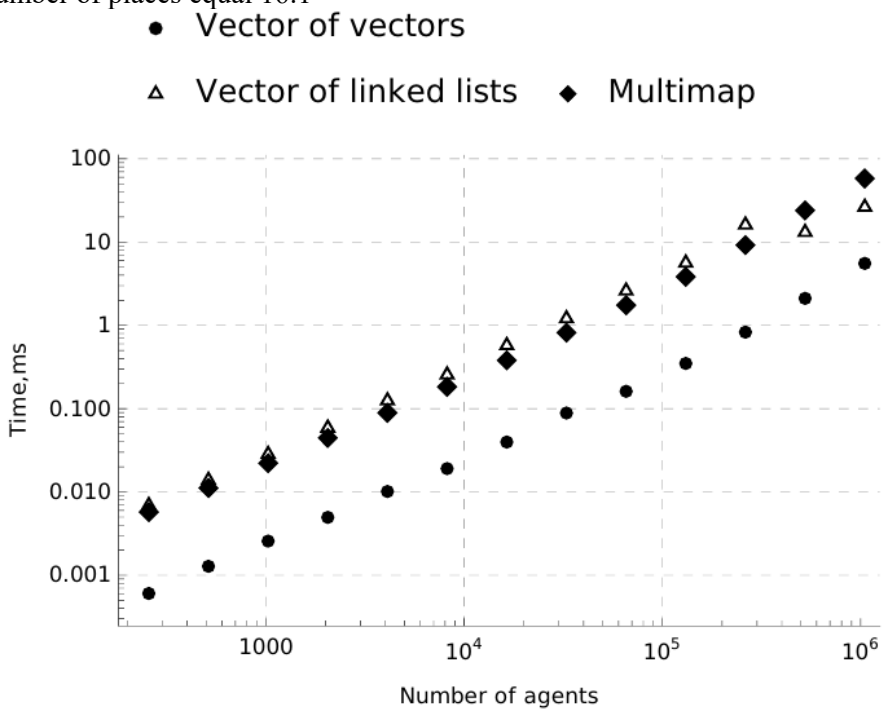

Figure 6: The execution time of the grouping algorithm for the average ratio of the number of agents to the number of places equal 256:1

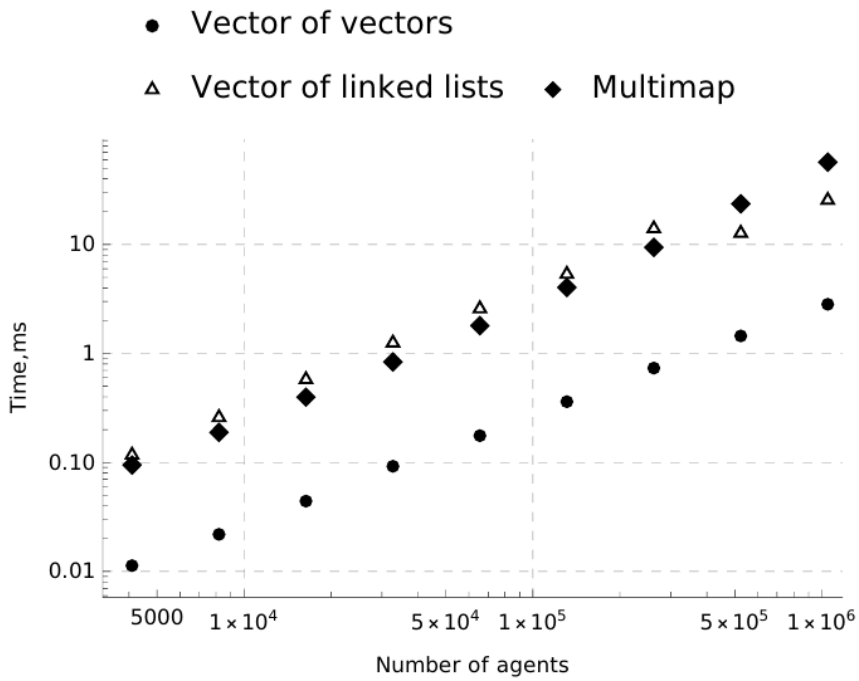

Figure 7: The execution time of the grouping algorithm for the average ratio of the number of agents to the number of places equal 4096:1

The results show that the implementation of a multiset through a vector of vectors is much more effective. This can be explained by the fact that the number of memory allocation can be reduced to a minimum with an optimally configured schedule for changing the size of the vector. And although it is impossible to predict the number of elements for each key in advance without having prior information about it, in the case of using a multiset to count visits of agents to places in the framework of infection modeling, it can be assumed that this number remains approximately the same. That is:

$$\forall p \in P, \forall t \in T, \forall h \in H: \sum_{a} 1\left(p = \phi(a,t,h)\right) \approx \sum_{a} 1\left(p = \phi(a,t+1,h)\right) \tag{18}$$

Obviously, in the real world, this statement is generally not true. For example, it is valid to assume that the number of visitors of shopping malls increases significantly on Saturdays, compared to Fridays in countries where the working week ends on Friday. However, this can be partially solved by keeping the size of the buffers equal to at least the last number of elements multiplied by some constant.

In the general case, one can achieve a minimum number of memory allocations equal to zero if set the size of each buffer equal to the number of agents. However, due to the assumption 15, the number of buffers is proportional to the number of agents. This makes the space complexity of such an algorithm equal to $O(n^2)$, which is unacceptable.

Apart from aforementioned algorithmic optimizations, computational optimizations can also be applied. One may notice, the evaluation of state change probabilities involves large amount of multiplication operations if the form:

$$\underbrace{x_1 x_1 \dots x_1}_{p_1} \dots \underbrace{x_n x_n \dots x_n}_{p_n} \tag{19}$$

Computation of such expression can be reformulated as:

$$\prod_{i=1}^{n} x_i^{p_i} = exp\left(\sum_{i=1}^{n} p_i ln x_i\right) \tag{20}$$

If we denote by $t_x, t_+, t_{exp}, t_{ln}$ the calculation time of the product, sum, exponential function, and natural algorithm, respectively, the calculation time of the left part will be equal to $t_x(\sum p_i - 1)$, the right one - $(t_{ln} + t_+)\sum p_i - t_+ + t_{exp}$. In the general case, the latter value is greater because the calculation of the natural logarithm is a time-consuming operation. However, in the case where the values of x_i are constants known during the compilation, the computation time of the right-hand side

will be equal to $t_+(\sum p_i - 1) + t_{exp}$. Because computing the product of two floating-point numbers requires more CPU cycles (for most architectures) than computing the sum [24], computing the optimized version is more efficient when:

$$t_* - t_+ > \frac{t_{exp}}{\sum p_i - 1}, \tag{21}$$

which is valid for sufficiently large values of $\sum p_i$. Thus, if the disease distribution function depends only on the resulting state and the recipient and donor states, the specified optimization can be used. To check the effect of this optimization, an experiment was conducted, during which execution times with and without it were measured for various model sizes. We observed some difference between optimized and default approach on the experimental machine for small number of agents (Fig. 5). As, the test was done on a single machine with Intel Core i7-8700K, for more general results, additional experiments should be performed. We leave this problem for a future research.

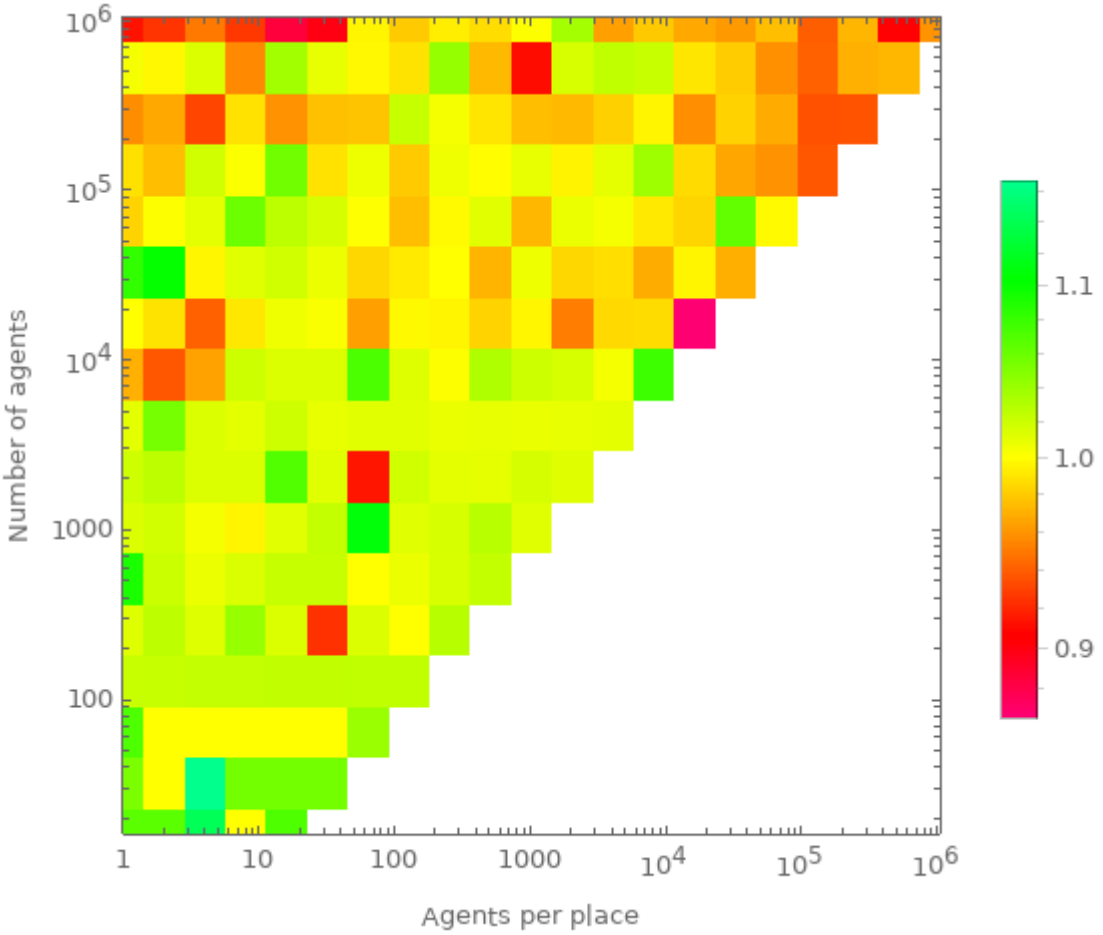

Figure 8: Simulation step performance increase (higher values are better). White areas correspond to absent data points.

4.3. Performance estimation

Comparing the performance of algorithms across different machines is not a trivial task. Modern CPUs are able to reach significant performance boost by using caching, instruction pipelines and other technologies, the impact of which on a specific algorithm is difficult to predict [25]. So, to compare out approach to existing ones, we perform two sets of measurements: first one on a desktop machine with Intel Core i7-8700K and second — on a 12 years old laptop with Intel Core i3-2330M.

In order to account for various model sizes and CPU core numbers, we use the following metric:

$$\frac{T}{10^{-6} N_a * N_d * N_s * N_c}, \quad (22)$$

where T - simulation execution time in seconds, N_a,N_d,N_s - number of agents, simulation steps and time steps in a model respectively, N_c - number of CPU cores. This metric is only suitable for algorithms with linear computational complexity as function of number of agents.

5. Results

We compared several models and their implementation approaches with CTrace [26] modeling language, whose translator implements aforementioned algorithms. Our test model didn't captured all details of each model, but instead was based around their common features like agent activity scheduling, several different place types, etc. To compute the aforementioned metric for it, a number of experiments were conducted with various model sizes.

Table 1
Performance comparison of various agent-based epidemiological models/approaches

Method	Metric, s	Hardware
FluTe[12]	4.00 — 13.71	32x Intel Core2 Duo T9400
AvilovNetLogo[16]	511.36 — 10960.15	Intel Core i3 330M 2.13Ghz
AvilovOracle[16]	5.75 — 13.71	
FRED[19]	1.20 — 7.29	SGI Altix UV supercomputer
MontañolaNetLogo[27]	2.47 — 24.71	Intel Core i5 3.20 GHz
GiacopelliLombardy[28]	240	AMD 3900X 12-core
CTrace[26]	**0.67**	**Intel Core i7-8700K 3.70 GHz**
	1.38	**Intel Core i3-2330M 2.20 GHz**

To be able to compare the developed approach to previously developed, we measured it's performance on both modern CPU and 12-years old one. Even though, the CPU generation range is significantly wide, we can assume that our approach is at least as performant as previously developed, offering the most flexible model specification. This flexibility is explained by CTrace's independence of modeling space structure, number and types of places (and other modeling entities), dynamics of a disease, etc. Given these results, we can conclude that usage of the aforementioned optimization approaches can provide significant benefits for agent-based epidemiological model computational performance on a single CPU core.

6. Conclusions

This paper presents a discussion of the challenges encountered in the implementation of agent-based modeling for infectious diseases. Specifically, the focus is on the time complexity of the simulation algorithm, which is identified as a key issue. To address this problem, a mathematical framework for agent-based epidemiological modeling is developed. This framework enables exploration of various optimization techniques for the simulation process. Several algorithms for contact tracing, which have linear time complexity, are proposed, depending on the nature of the infectious disease being modeled. These algorithms are tested and compared with existing models. Results demonstrate that the proposed approach is at least as effective as other implementations, while offering increased model flexibility through the use of the CTrace language interface. Described approach scales up by running multiple models, each per CPU core, which is often desirable for this kind of models as it leads to statistically more significant results. The question of single model parallelization we leave for a future research

7. References

[1] R. Dandekar, G. Barbastathis, Quantifying the effect of quarantine control in Covid-19 infectious spread using machine learning, medRxiv, 2020, pp. 1-13. doi: 10.1101/2020.04.03.20052084.

[2] W. O. Kermack, A. G. McKendrick, A contribution to the mathematical theory of epidemics, Proceedings of the royal society of London, Vol. 115, 1927, pp. 700–721. doi: 10.1098/rspa.1927.0118.

[3] K. S. Sharov, Creating and applying SIR modified compartmental model for calculation of COVID-19 lockdown efficiency, Chaos, Solitons & Fractals, Vol. 141, 2020, pp. 1-14. doi: 10.1016/j.chaos.2020.110295.

[4] S. Nie, W. Li, Using lattice SIS epidemiological model with clustered treatment to investigate epidemic control, Biosystems, Vol. 191, 2020, doi: 10.1016/j.biosystems.2020.104119.

[5] A. Ceria, K. Köstler, R. Gobardhan, H. Wang, Modeling airport congestion contagion by heterogeneous SIS epidemic spreading on airline networks, PLoS One, Vol. 16, 2021. doi: 10.1371/journal.pone.0245043.

[6] W. Sanusi, N. Badwi, A. Zaki, S. Sidjara, N. Sari, M. I. Pratama, S. Side, Analysis and Simulation of SIRS Model for Dengue Fever Transmission in South Sulawesi, Indonesia, Journal of Applied Mathematics, Vol. 2021, 2021, pp. 1-8. doi: 10.1155/2021/2918080.

[7] H.-Y. Shin, A multi-stage SEIR(D) model of the COVID-19 epidemic in Korea, Annals of Medicine, Vol. 53, 2021, pp. 1160–1170. doi: 10.1080/07853890.2021.1949490.

[8] S. Annas, M. I. Pratama, M. Rifandi, W. Sanusi, S. Side, Stability analysis and numerical simulation of SEIR model for pandemic COVID-19 spread in Indonesia, Chaos, Solitons & Fractals, Vol. 139, 2020, pp. 1-7. doi: 10.1016/j.chaos.2020.110072.

[9] Z. Ahmad, M. Arif, F. Ali, I. Khan, K. S. Nisar, A report on COVID-19 epidemic in Pakistan using SEIR fractional model, Scientific Reports, Vol. 10, 2020, pp. 1–14. doi: 10.1038/s41598-020-79405-9.

[10] W. Li, J. Gong, J. Zhou, L. Zhang, D. Wang, J. Li, C. Shi, H. Fan, An evaluation of COVID-19 transmission control in Wenzhou using a modified SEIR model, Epidemiology & Infection Vol. 149, 2021, pp. 1-12. doi: 10.1017/S0950268820003064.

[11] M. W. Levin, M. Shang, R. Stern, Effects of short-term travel on COVID-19 spread: A novel SEIR model and case study in Minnesota, PLoS One, Vol. 16, 2021, pp. 1-16. doi: 10.1371/journal.pone.0245919.

[12] D. L. Chao, M. E. Halloran, V. J. Obenchain, I. M. Longini Jr, FluTE, a publicly available stochastic influenza epidemic simulation model, PLoS computational biology, Vol. 6, 2010, pp. 1-8. doi: 10.1371/journal.pcbi.1000656.

[13] Minar, N., Burkhart, R., Langton, C., & Askenazi, M. The Swarm Simulation System: A Toolkit for Building Multi-Agent Simulations, ACM Transactions on Modeling and Computer Simulation, 1996.

[14] P. Patlolla, V. Gunupudi, A. R. Mikler, R. T. Jacob, Agent-based simulation tools in computational epidemiology, International Workshop on Innovative Internet Community Systems, Springer, 2004, pp. 212–223. doi: 10.1007/11553762_21.

[15] Tisue, S., & Wilensky, U. (2004, May). Netlogo: A simple environment for modeling complexity. In International conference on complex systems, Vol. 21, pp. 16-21.

[16] K. Avilov, O. Y. Solovey, Agent-Based Models: Approaches and Applicability to Epidemiology, 2012, pp. 425-443. doi: 10.1146/annurev-publhealth-040617-014317.

[17] Collier, Nick. Repast: An extensible framework for agent simulation, The University of Chicago's Social Science Research, Vol. 36, 2003, pp. 1-18.

[18] N. Collier, M. North, Parallel agent-based simulation with Repast for high performance computing, Simulation, Vol. 89, 2013, pp. 1215–1235. doi:10.1177/0037549712462620. doi: 10.1177/0037549712462620.

[19] J. J. Grefenstette, S. T. Brown, R. Rosenfeld, J. DePasse, N. T. Stone, P. C. Cooley, W. D. Wheaton, A. Fyshe, D. D. Galloway, A. Sriram, et al., FRED (a Framework for Reconstructing Epidemic Dynamics): an open-source software system for modeling infectious diseases and

control strategies using census-based populations, BMC public health, Vol. 13, 2013, pp. 1–14. doi: 10.1186/1471-2458-13-940.
[20] W. Van den Broeck, C. Gioannini, B. Gonçalves, M. Quaggiotto, V. Colizza, A. Vespignani, The GLEaMviz computational tool, a publicly available software to explore realistic epidemic spreading scenarios at the global scale, BMC infectious diseases, Vol. 11 2011, pp. 1–14. doi: 10.1186/1471-2334-11-37.
[21] D. P. Mehta, S. Sahni, Handbook of Data Structures and Applications, Chapman and Hal l/CRC, 2004, p. 163. doi: 10.1201/9781420035179.
[22] D. P. Mehta, S. Sahni, Handbook of Data Structures and Applications, Chapman and Hall/CRC, 2004, p. 47. doi: 10.1201/9781420035179.
[23] T. Van Dijk, Analysing and Improving Hash Table Performance, 10th Twente Student Conference on IT. University of Twente, Faculty of Electrical Engineering and Computer Science, Citeseer, 2009.
[24] A. Fog, et al., Instruction tables: Lists of instruction latencies, throughputs and micro-operation breakdowns for Intel, AMD, and VIA CPUs, Copenhagen University College of Engineering, 2011.
[25] J. Domínguez, E. Alba, A Methodology for Comparing the Execution Time of Metaheuristics Running on Different Hardware, European Conference on Evolutionary Computation in Combinatorial Optimization, Springer, 2012, pp. 1–12. doi: 10.1007/978-3-642-29124-1_1.
[26] Sarnatskyi V., Baklan I. CTrace: Language for Definition of Epidemiological Models with Contact-Tracing Transmission, International Scientific Conference “Intellectual Systems of Decision Making and Problem of Computational Intelligence”. – Springer, Cham, 2023, pp. 426-448. doi: 10.1007/978-3-031-16203-9_25.
[27] C. Montañola-Sales, J. F. Gilabert-Navarro, J. Casanovas-Garcia, C. Prats, D. López, J. Valls, P. J. Cardona, C. Vilaplana, Modeling tuberculosis in Barcelona. A solution to speed-up agent-based simulations, in: 2015 Winter Simulation Conference (WSC), IEEE, 2015, pp. 1295–1306. doi: 10.1109/WSC.2015.7408254.
[28] G. Giacopelli, et al., A Full-Scale Agent-Based Model to Hypothetically Explore the Impact of Lockdown, Social Distancing, and Vaccination During the COVID-19 Pandemic in Lombardy, Italy: Model Development, Jmirx med, Vol. 2, 2021, pp. 1-11. doi: 10.2196/24630.