Proceedings of the 33rd European Modeling & Simulation Symposium (EMSS),008
18th International Multidisciplinary Modeling & Simulation Multiconference

2724-0029 © 2022 The Authors.
doi: 10.46354/i3m.2022.emss.008

Parallelization of large-scale agent-based epidemiological simulations

Steffen Fürst[1,*] and Christian Rakow[2]

[1]Zuse Institut Berlin, Takustraße 7, Berlin, 14195, Germany
[2]Department of Transport Systems Planning and Telematics, Technische Universität Berlin, Salzufer 17-19, Berlin, 10587, Germany

[*]Corresponding author. Email address: fuerst@zib.de

Abstract

Agent-based epidemiological models have been applied widely successfully during the SARS-CoV-2 pandemic and assisted policymakers in assessing the effectiveness of intervention strategies. The computational complexity of agent-based models is still challenging, and therefore it is important to utilize modern multi-core systems as good as possible.
In this paper, we are presenting our work on parallelizing the epidemiological simulation model *MATSim Episim*. Episim combines a large-scale person-centric human mobility model with a mechanistic model of infection and a person-centric disease progression model.
In general, the parallelization of agent-based models with an inherent sequential structure — in the case of epidemiological models, the temporal order of the individual movements of the agents — is challenging. Especially when the underlying social network is irregular and dynamic, they require frequent communication between the processing elements. In Episim, however, we were able to take advantage of the fact that people are not contagious on the same day they become infected, and therefore immediate health synchronization is not required. By parallelizing some of the most computationally intensive submodels, we are now able to run MATSim Episim simulations up to eight times faster than the serial version. This makes it feasible to increase the number of agents, e.g. to run simulations for the whole of Germany instead of just Berlin as before.

Keywords: Epidemics; Agent-based modeling; Parallel programming; Simulation

1. Introduction

Episim is a large-scale agent-based epidemiological model that combines a person-centric human mobility model with a mechanistic model of infection and disease progression (Müller et al., 2021).

The movements of individuals, including performed activities in which they may interact with others, can be directly taken from the data. Episim can be used to evaluate various intervention strategies, such as closing educational facilities, reducing out-of-home activities, wearing masks, or contact tracing and quarantine.

The model is regularly used to advise the German federal government (e.g. Müller et al. (2021b,a)). The current main contribution of these reports is to provide differentiated predictions of the impact of various interventions, such as reductions of activity participation, masks, or vaccinations.

Each of these reports requires between 10000-20000 simulations based on mobility data for Berlin (Germany) and taking into account the activities of 25% of Berlin's population, i.e. almost one million people. The model is implemented in Java and the initial version was single-threaded. In this paper, we are presenting our parallelization approach, which is thread-based and uses standard Java libraries.

On the one hand, the goal was to improve the utilization of the computing nodes for simulations of this configuration. On the other hand, the parallelization should also allow the simulations to be extended to the whole of Germany, or alternatively to the entire Berlin population.

Section 2 gives a brief overview of different approaches to epidemiological modeling and the specifics of agent-based models in this area. Section 3 then introduces MATSim Episim itself, explaining which parts of the code were parallelized and the ways in which this was done. The performance improvements achieved by the parallelization are then presented in section 4, and section 5 draws the conclusions from this work.

2. State of the art

Modeling infectious diseases has been a topic of interest for many decades, going back to a time before computers were invented. One of the most well-known types of models are compartmental models, with the *SIR-Model* (susceptible-infected-removed) (Kermack and Mckendrick, 1927) in its basic form.

In compartmental models, each individual belongs to exactly one exclusive group –*compartment*. The transition rates between the compartments are defined by a set of mathematical equations. This simplicity makes them straightforward to compute, and the performance depends on the underlying solver. Many variations with additional compartments have been applied throughout recent years, also in the context of modeling SARS-CoV-2 spread (Ndaïrou et al., 2020; Hou et al., 2020; Leontitsis et al., 2021).

Though, the main drawback is, that the populations within the compartments are assumed to be homogenous. They do not reflect human social structures, where interactions are typical limited to a certain contact network (Tolles and Luong, 2020). Furthermore, modeling new intervention strategies, one key interest area to contain the spread of a pandemic, is often difficult as the effect on transition rates are not known a-priori.

Agent-based (or individual-based) models have been widely successful in this regard as they allow capturing a more complex disease process, individual behavior, age and household structures, as well as geospatial structures.

Many agent-based models, including the one evaluated in this paper, have been used to predict the effects of intervention strategies such as mobility restrictions, masks, contact tracing and testing, as well as pharmaceutical interventions (vaccination) (e.g. Müller et al. (2021); Mahdizadeh Gharakhanlou and Hooshangi (2020); Hinch et al. (2021); Kerr et al. (2021); Bitencourt (2021)).

This complexity makes agent-based models much more computational expensive than compartmental models. Given that modern computers are equipped with an increasing number of cores, it is crucial to utilize as many cores in parallel as possible.

Parallelization is challenging, especially when the underlying social network is irregular and dynamic, and the many interactions between agents require frequent synchronization.

During the early phases of the COVID-19 pandemic, most research has naturally focused on predictions and strategies to contain the pandemic. Some authors report the performance of their models and scalability in terms of problem size (Kerr et al., 2021). Nevertheless, it seems that parallelization has not been a key focus area. This has been different before the pandemic, as several highly concurrent models to simulate the spread of infectious diseases have been developed and applied (Barrett et al., 2008; Grefenstette et al., 2013; Bhatele et al., 2017); most often to simulate the spread of seasonal influenza. In the case of COVID-19, it appears that more often new models have been developed instead of using existing frameworks for infectious diseases. One reason could that it required

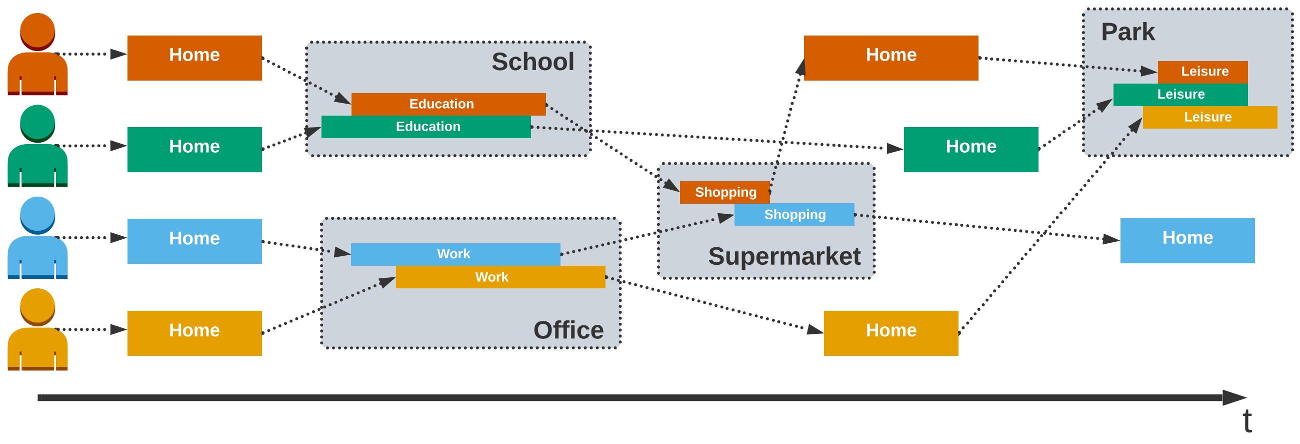

Figure 1. Illustrative daily plans of persons and their activities; Colored boxed represent an activity at a specific location. Gray boxes indicate an overlap, where persons may have a contact and could possibly infect each other.

new methods of interventions and restrictions that have not been seen and modeled before. Likewise, Episim has also been specially develop for the COVID-19 pandemic and started with a single-threaded implementation.

In this paper, we present our parallelization approach and show results on scenarios of different sizes up to the whole country of Germany. The main idea is similar to the approach presented by Barrett et al. (2008). For instance, they also divide the computational workload by the geographical locations of agents. Compared to other agent-based COVID-19 simulations, Episim is one of a few that provides a detailed study about parallelization.

3. Materials and Methods

MATSim Episim builds on the multi-agent transport simulation (MATSim) Axhausen et al. (2016), an activity-based, extensible, multi-agent simulation framework implemented in Java. MATSim simulations produce event files that describe the movement and activities of each individual person. Such plans are illustrated in Figure 1. The idea of Episim is to attach a contact- and infection model to this information in order to simulate large scale virus spreading. After each performed activity, a contact model determines with whom an agent had a contact and an infection model determines the probability of an infection if any of the agents was carrying the virus.

Episim uses the MATSim infrastructure for configuration, setting up dependencies, and its data schemas for reading or writing files. Apart from that, the Episim simulation loop and models are standalone new code, also developed in Java. In its original form, it is single-threaded, apart from the garbage collector. Recent versions of Episim consist of many more submodels, e.g for disease progression, contact tracing and quarantine, vaccinations, interventions such as mask or curfew hours, or even geographical restrictions. It also supports resuming from previous simulation state, which can save a lot of time, as we are more than two years into the pandemic, and simulating all of this history is not always necessary.

The Episim model is regularly used for reports on the current SARS-CoV-2 situation Müller et al. (2021b,a). Model details and results can be found in Müller et al. (2021) or Nagel et al. (2021). The remainder of the paper will rather focus on our parallelization approach, that is in general also applicable to other agent based models.

Figure 2 gives an overview of the parallelization approach and core simulation loop. For each day, Episim runs all submodels iteratively, until the desired number of simulation days have been reached. Disease progression only happens at the end of one day, meaning one person can not get infectious on the same day it was infected.

The focus is on parallelizing the processing of movement profiles, as well as contact and infection models, since this requires the most computing time. To parallelize this, partitioning could be done over the set of persons or the locations. However, since at the time a person leaves a location, we need to know which other people were also there during their stay. Partitioning the set of people would require a large synchronization effort.

On the other hand, the locations can be handled very well independently of each other. Even in the case that a person has actually already been infected at another location, this person can only infect other people a few days later, so that no immediate synchronization is necessary here.

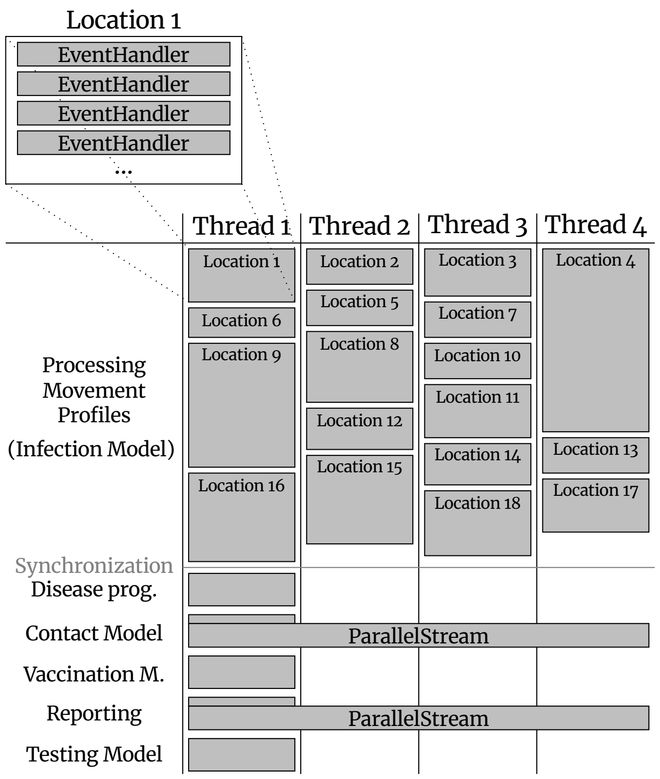

Figure 2. Parallelization approach and inner simulation loop, which will be executed iterativly for the desired number of simulation days.

When the movement profiles of individuals are aggregated and sorted in time, the individual events can be easily assigned to a thread depending on the location. Fortunately, MATSim already provides exactly the movement profiles in this aggregated, sorted form.

In order to achieve good load balancing, it is important to consider how crowded the locations are, since, for example, a shopping mall has to process many more events than an apartment, and therefore the runtime for iterating all the events belonging to a location can be very different. The threading implementation itself is done using Java8 CompletableFutures. Additional synchronization was necessary for contact tracking, since the same list of contacts can be modified from different threads.

Through profiling, it was known that about 80% of the runtime is required for the processing of these movement profiles in the single threaded version. In the remaining part of the code, we found two time-consuming loops that could be parallelized using ParallelStreams.

4. Results and Discussion

The performance results presented in this section refer to simulations performed on the Lise cluster of the North German Supercomputing Alliance (HLRN). The nodes of the cluster are equipped with 2 Intel Cascade Lake Platinum 9242 (CLX-AP) CPUs and 384 GB Ram.

The simulations used in the reports for the German federal government (e.g., Müller et al. (2021b,a)) were performed for the state of Berlin. To save computation time, the number of agents in these simulations is reduced to a quarter of the individuals they represent. This configuration is referred to as "berlin25" in the following figures, where 25 stands for 25%, as only 25% of the population is represented by an agent.

In addition to this configuration, we will also examine two other configurations. The "berlin100" configuration refers again to the area of the federal state of Berlin, but this time the activities of the complete Berlin's population is taking into account.

And the "germany25" configuration extends the scope of the study to the area of Germany, whereby as in the "berlin25" configuration again only 25% of the population is considered.

(in Mio.)	berlin25	berlin100	germany25
Locations	0.9	1.3	13.3
Agents	1.2	4.8	20.6
Events per Week	75.8	295.6	973.1

Table 1. The number of locations, persons and events for the different configurations. An event refers to the movement profiles of the agents and is either the arrival or the departure of a person from a location.

Table 1 shows the number of places, people and events for these configurations. For each activity there are two events. The first determines when a person reaches the location and starts the activity, the second when a person leaves the location. The probability of an infection is calculated when the person leaves the place, both for the person himself and for all contacts of the person.

4.1. Single simulation on node

First, we consider the speedup achieved when only a single simulation is started on a node, and the resources are artificially limited to *n* threads.

The overall performance increase achieved by the parallelization approach described in section 3 is shown in Figure 3, where the simulations went through 50 iterations of the inner simulation loop. This is the maximum number of iterations possible for the Germany25 configuration before it exceeds the 12-hour wall time limit, which is the maximum job length on the standard nodes of the Lise cluster.

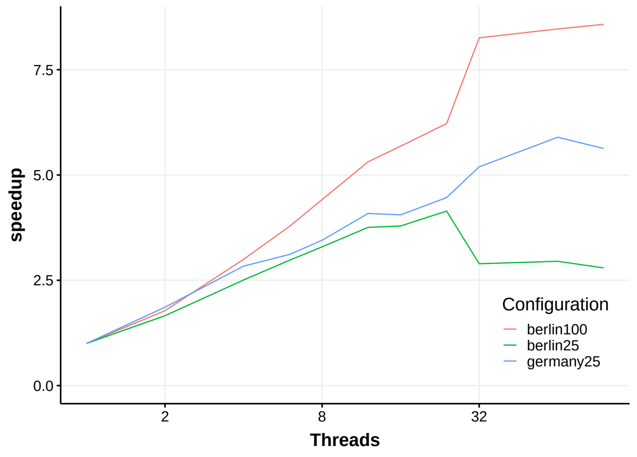

Figure 3. Speedup compared to the single threaded runs for the complete inner simulation loop.

We see that compared to the "berlin25" configuration increasing the number of agents per location as in the "berlin100" configuration increases the speedup for high thread counts by a factor of three, and increasing the number of locations themselves as in the "germany25" configuration only by a factor of two. This can be explained by the fact that the increasing the number of agents reduces the proportion of time spent in serial code, as Table 2 shows.

	berlin25	berlin100	germany25
time (ms) in serial section	1617	8566	90541
time (ms) in parallelized section	12615	140092	720997
maximal speedup (96 cores)	8.81	16	8.85
maximal speedup (∞ cores)	9.6	19.1	9.64

Table 2. Average runtime of the inner simulation loop for the serial and parallelized parts. The runtimes are given for simulations utilizing only a single thread. From Amdahl's law follows the maximal speedup possible for simulation runs on a single Lise node with 96 cores and the theoretical maximal speedup in the case of unlimited resources.

When we compare the "berlin25" and "berlin100" configuration in Table 2 we see that the serial runtime increases more or less linearly with the number of agents, but the parallel one takes 11 times longer when we quadruple the number of agents. This is due to the way the Movement Profiles are processed and explains why the runtime portion of the serial code decreases in the "berlin100" configuration and thus the speedup is better than in the other configurations.

When we compare the maximal possible speedup for the given fraction of time used for the serial section of the code to the real speedup achieved as shown in Figure 3, we can see that only the "berlin100" configuration comes near to this limit.

And also if we only compare the time needed for processing the movement profile, as shown in Figure 4, we can see similar patterns.

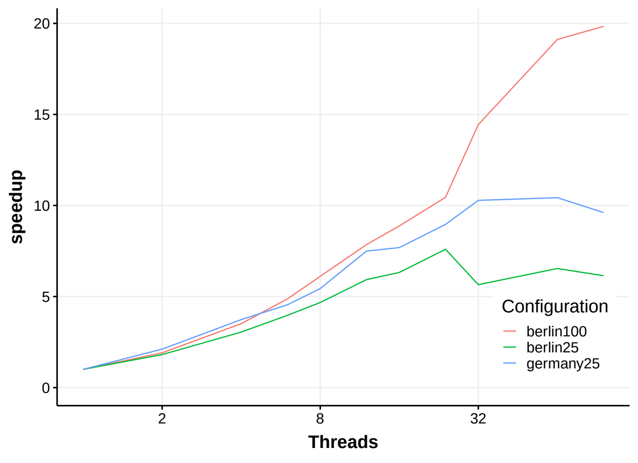

Figure 4. Speedup of the Infection Submodel compared to the single threaded runs.

This is a bit surprising, since it would be expected that the load balancing works better the more locations there are.

	berlin25	berlin100	germany25
load	0.8	0.77	0.99

Table 3. Load balancing the infection submodel for runs with 96 cores. In the optimal case, where all cores take the same amount of time for the calculation, the value would be 1.

Table 3 confirms this expectation. The values given there correspond to the longest runtime of a core for the processing of the movement profiles divided by the mean value of the runtimes of all cores. The reciprocal value thus corresponds to the maximum possible improvements in the case that all cores require exactly the same amount of time.

Why the achieved speedup results do not match the balancing needs further research. The current hypothesis is that the agents to locations ratio plays a role in the synchronization required to make the infection submodel thread safe.

4.2. Multiple simulation on node

For the results discussed so far, a simulation allocated always a whole node. However, since different scenarios are to be investigated for the reports (e.g. Müller et al. (2021b,a)) and this is also done for different seeds due to the stochastic behavior of the simulation, several simulations are usually started in parallel on a node.

In this section, we focus on the “berlin25” configuration as it is used for the reports. Since each simulation run requires about 11 GB of memory, it is not possible to start a simulation per core on a standard compute node of HLRN's Lise cluster. Instead, only 32 simulations can be started in parallel on a single node, although it has 96 cores. So the serial version of Episim left a lot of resources unused.

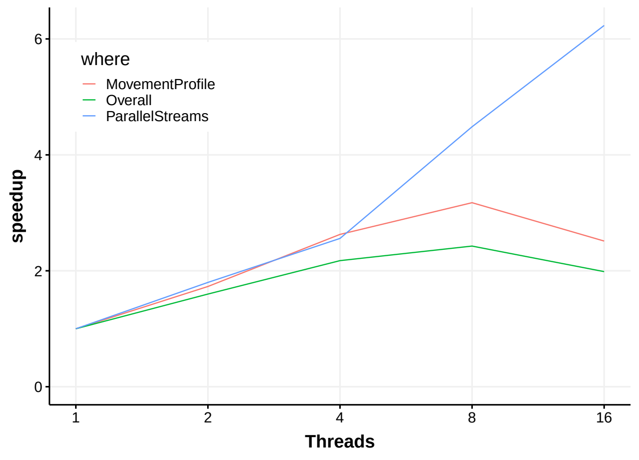

Figure 5. Speedup compared to the single threaded runs for different code sections in the “berlin25” configuration, when 32 simulations are performed simultaneously on a node.

Figure 5 shows the speedup of the overall simulation and also separately the speedup of the multi-threaded infection model (MovementProfile) and the loops parallelized with Java's ParallelStreams for the case where 32 simulation runs of the “berlin25” configuration were performed simultaneously on a single node.

Not surprisingly, the maximum achievable speedup of 2.43 with 8 threads is lower than the maximum achievable speedup of 4.14 with 24 threads for a single run. However, it is interesting to see that the best result is achieved with an overcommitted number of 256 threads on nodes with 192 virtual cores, although hyperthreading was counterproductive for the single runs. But since the amount of runtime in serial segments is relatively high, and therefore the 256 threads are not always active at the same time, the “gaps”, which are caused by the serial code, can be used well by the other simulations.

In Figure 6 is shown, how the improvements are distributed among the serial and the different parallelized sections. As can already be seen in Figure 5, the loops parallelized with the parallel streams scale better than the parallelization of the Infection Model (MovementProfile), but the runtime of the Infection Model dominates the overall runtime. The overcommitment of threads increases the runtime of the serial sections, but with up to 8 threads this is more than compensated by the runtime improvements of the parallel sections, until increasing the number of threads is counterproductive for the infection model. The runtime increase seen here can be explained by thread overcommitment, since this reduces the load balancing of the movement profile processing, which is quite sensitive to an uneven thread scheduling.

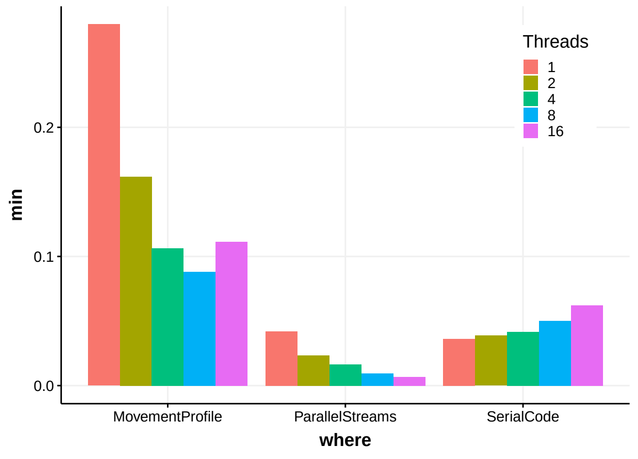

Figure 6. Comparing the average iteration runtime of different code sections in the "berlin25" configuration for different numbers of threads per Simulation. 32 simulations are performed simultaneously on a node.

5. Conclusions

As we have shown, it can be useful to examine an existing serial implementation of a model for parallelization capabilities, although the performance result certainly cannot match an implementation designed from the ground up for parallelization, as for example Barrett et al. (2008).

Also, the approach presented here exploits some features of MATSim Episim which are not found in many epidemiological simulations. Especially the fact that the movement profiles are predefined by exogenous data and not randomly generated at runtime facilitates the load balancing, since the required cpu-time per location is more or less deterministic and therefore the locations can be reasonably distributed among the threads.

Two additional and more common conditions were very helpful, namely that the runtime was dominated by the infection submodel and that changes in a person's health status do not need to be synchronized within one iteration — which corresponds to one day of simulated time — since an exposed person cannot become infectious within one day.

Converting all submodels from their sequential structure to a parallel one would be very labor-intensive, but since many (32) simulations can be started in parallel in the "berlin25" configuration, this can be partially compensated by starting more threads than there are cores. After all, a speedup of 2.4 could be achieved in the computations for the reports to the German government, even though only two-thirds of the cores were not used in the serial case.

Furthermore, thanks to parallelization, it is now possible to extend the area from Berlin to the whole of Germany. Before parallelization, only 50 days could be calculated within the maximum walltime of 12 hours, whereas now the simulation of a whole year can be performed in 10 hours.

6. Funding

The work on the paper was funded by the Ministry of research and education (BMBF) Germany (01KX2022A) and TU Berlin; regular reports can be found through this search: `https://depositonce.tu-berlin.de/simple-search?query=modus-covid`. Zuse Institute Berlin (ZIB) provided CPU time.

References

Axhausen, K. W., Nagel, K., and Horni, A. (2016). *The Multi-Agent Transport Simulation MATSim*. Ubiquity Press.

Barrett, C. L., Bisset, K. R., Eubank, S., Feng, X., and Marathe, M. V. (2008). Episimdemics: An efficient algorithm for simulating the spread of infectious disease over large realistic social networks. *2008 SC - International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–12.

Bhatele, A., Yeom, J.-S., Jain, N., Kuhlman, C. J., Livnat, Y., Bisset, K. R., Kale, L. V., and Marathe, M. V. (2017). Massively parallel simulations of spread of infectious diseases over realistic social networks. In *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 689–694.

Bitencourt, J. (2021). Analyzing the impact of vaccination on covid-19 spread and hospitalizations: A multi-paradigm simulation modeling approach. In *Proceedings of the 33rd European Modeling & Simulation Symposium*, page nil.

Grefenstette, J. J., Brown, S. T., Rosenfeld, R., DePasse, J., Stone, N. T., Cooley, P. C., Wheaton, W. D., Fyshe, A., Galloway, D. D., Sriram, A., Guclu, H., Abraham, T., and Burke, D. S. (2013). 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*, 13(1):940.

Hinch, R., Probert, W. J., Nurtay, A., Kendall, M., Wymant, C., Hall, M., Lythgoe, K., Bulas Cruz, A., Zhao, L., Stewart, A., Ferretti, L., Montero, D., Warren, J., Mather, N., Abueg, M., Wu, N., Legat, O., Bentley, K., Mead, T., Van-Vuuren, K., Feldner-Busztin, D., Ristori, T., Finkelstein, A., Bonsall, D. G., Abeler-Dörner, L., and Fraser, C. (2021). OpenABM-Covid19-An agent-based model for non-pharmaceutical interventions against COVID-19 including contact tracing. *PLoS Computational Biology*, 17(7):e1009146.

Hou, C., Chen, J., Zhou, Y., Hua, L., Yuan, J., He, S., Guo, Y., Zhang, S., Jia, Q., Zhao, C., Zhang, J., Xu, G., and Jia, E. (2020). The effectiveness of quarantine of Wuhan city against the Corona Virus Disease 2019 (COVID-19): A well-mixed SEIR model analysis. *Journal of Medical Virology*, 92(7):841–848.

Kermack, W. . and Mckendrick, A. G. (1927). 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.

Kerr, C. C., Stuart, R. M., Mistry, D., Abeysuriya, R. G., Rosenfeld, K., Hart, G. R., Núñez, R. C., Cohen, J. A., Selvaraj, P., Hagedorn, B., George, L., Jastrzębski, M., Izzo, A. S., Fowler, G., Palmer, A., Delport, D., Scott, N., Kelly, S. L., Bennette, C. S., Wagner, B. G., Chang, S. T., Oron, A. P., Wenger, E. A., Panovska-Griffiths, J., Famulare, M., and Klein, D. J. (2021). Covasim: An agent-based model of COVID-19 dynamics and interventions. *PLoS Computational Biology*, 17(7):e1009149.

Leontitsis, A., Senok, A., Alsheikh-Ali, A., Nasser, Y. A., Loney, T., and Alshamsi, A. (2021). SEAHIR: A Specialized Compartmental Model for COVID-19. *International Journal of Environmental Research and Public Health*, 18(5):1–11.

Mahdizadeh Gharakhanlou, N. and Hooshangi, N. (2020). Spatio-temporal simulation of the novel coronavirus (COVID-19) outbreak using the agent-based modeling approach (case study: Urmia, Iran). *Informatics in Medicine Unlocked*, 20:100403.

Müller, S. A., Balmer, M., Charlton, W., Ewert, R., Neumann, A., Rakow, C., Schlenther, T., and Nagel, K. (2021). Predicting the effects of COVID-19 related interventions in urban settings by combining activity-based modelling, agent-based simulation, and mobile phone data. *PLOS ONE*, 16(10):1–32.

Müller, S. A., Charlton, W., Conrad, N. D., Ewert, R., Jefferies, D., Rakow, C., Wulkow, H., Conrad, T., Schütte, C., and Nagel, K. (2021a). Modus-covid bericht vom 09.04.2021. Technical report, Technische Universität Berlin.

Müller, S. A., Charlton, W., Conrad, N. D., Ewert, R., Jefferies, D., Rakow, C., Wulkow, H., Conrad, T., Schütte, C., and Nagel, K. (2021b). Modus-covid bericht vom 19.03.2021. Technical report, Technische Universität Berlin.

Nagel, K., Rakow, C., and Müller, S. A. (2021). Realistic agent-based simulation of infection dynamics and percolation. *Physica A: Statistical Mechanics and its Applications*, 584:126322.

Ndaïrou, F., Area, I., Nieto, J. J., and Torres, D. F. (2020). Mathematical modeling of COVID-19 transmission dynamics with a case study of Wuhan. *Chaos, Solitons and Fractals*, 135:109846.

Tolles, J. and Luong, T. (2020). Modeling Epidemics with Compartmental Models.