Modeling the COVID-19 Epidemic in a Parallelized City Simulation

24

Derek Stratton, William Garner, Terra Williams, and Frederick C. Harris Jr.

Abstract

A pandemic can arise without warning, and it is important for those in charge of managing the outbreak to understand how diseases spread. Being able to simulate the spread of a disease in varying environments can help the world be more prepared when an outbreak occurs. The COVID-19 City Simulator allows the user to test the spread of the virus under multiple different scenarios. Parallel computing can help to make these simulations more efficient by allowing data to be gathered at a faster rate on a particle simulation. This paper shows how OpenMP and MPI can improve a pandemic simulation by cutting the runtime from over 25 s to under 10 s when 4 threads and 4 boxes are used. We also find that the speed of implementing a lockdown largely impacts the amount of cases and deaths in the city.

Keywords

Parallel computing · OpenMP · MPI · Hybrid model · COVID-19 · Pandemic · Simulation · disease spread · Particle simulation · Parameter tuning

24.1 Introduction

Many mathematical models exist to represent the spread of infectious diseases in a population. These models vary widely, but all models have large limitations when generalizing a population. However, they can still be helpful to gain understanding of the biological and sociological factors that contribute to the spread of disease [1]. This knowledge can be used to advice public health policy so institutions can deal with disease outbreaks in the best way possible. This study is being conducted due to its modern relevance with the COVID-19 pandemic.

Many countries have reacted differently to this pandemic, but being able to accurately simulate the spread of a disease can help the entire world handle outbreaks more effectively. This simulation can display how different factors such as population density and amount of public interaction will affect the spread of a virus. We will be looking at lockdowns orders, which include public policies that attempt to reduce spread by reducing interactions in a city. Due to the challenge of implementing lockdowns, some countries have implemented them more strictly and more quickly than others. This study attempts to investigate the extent to which the speed at which a city locks down affects the public health outcomes of the city.

The paper will also discuss how parallel computing can make simulations more efficient, since making simulations more efficient will allow for them to be more realistic and useful for understanding the problem. Shared memory parallelism involves using multiple threads that share a common memory space to execute tasks in parallel. Message passing parallelism involves executing tasks on different boxes that don't share memory, and require messages be sent to exchange information needed for processing. Both techniques are powerful and can be used in tandem to maximize the efficiency of computing resources on a simulation.

In Sect. 24.2, the background and relevant past research are discussed. In Sect. 24.3, the detailed approach of the simulation's implementation is described. In Sect. 24.4, the results of simulation trials are reported and analyzed. Finally, Sect. 24.5 gives the conclusion of the paper's work and describes future work to expand knowledge on this topic.

D. Stratton · W. Garner · T. Williams · F. C. Harris Jr. (✉)
Department of Computer Science and Engineering, University of Nevada Reno, Reno, NV, USA
e-mail: derekstratton@nevada.unr.edu; williamgarner@nevada.unr.edu; terrawilliams@nevada.unr.edu; fred.harris@cse.unr.edu

S. Latifi (ed.), *ITNG 2021 18th International Conference on Information Technology-New Generations*, Advances in Intelligent Systems and Computing 1346,
https://doi.org/10.1007/978-3-030-70416-2_24

24.2 Related Works

The simulation discussed in this paper is based on other models used to simulate the spread of disease. The first model used as inspiration is the SIR disease model. This model splits a population into three compartments: susceptible, infected, and recovered. This model was based on Kermack-McKendrick theory from 1927 [2]. The SIR Disease model uses transition functions to move individuals of the population between the three compartments. This system of compartments and transitions is modeled with ordinary differential equations (ODEs), and it is a deterministic model. Another type of models that improve upon deterministic models are stochastic disease models. They improve upon standard compartmental models like SIR by adding some elements of randomness [3]. These stochastic models can use Continuous Time Markov Chains or Stochastic Differential Equations to simulate the spread of disease.

Britton describes a special epidemic model with site contamination [4]. This model divides a space into several different sites with a random number of particles in each site to represent the individuals of a population. The particles can move randomly between neighboring sites. When an infected particle reaches a new site, all other particles currently at that site become infected. Germann made a model that used mitigation strategies for pandemic influenza in the United States [5]. This was a complex simulation model used to study influenza in the United States. The model used data from the 2000 US Census and divided the population into seven "mixing groups" that could contact one another. The model considered various interventions such as vaccination and social distancing to determine how different factors would affect the spread of influenza.

For the simulation, multiple methods of computer parallelization of particle simulators were researched. We had to decide which method of parallelization would be most appropriate for the project. One source of research discussed many different methods and platforms for parallel computing [6]. From this paper as well as our former experiences, we decided that MPI was the most appropriate method for parallelizing the simulation. Our research into parallelization methods also showed the difference in performance for a program using pure MPI against a program using a hybrid of MPI and OpenMP [7]. The hybrid uses of both MPI and OpenMP was found to improve performance over using MPI alone. The combination of these two sources lead us to test the simulation using three different methods of parallelization: OpenMP, MPI, and a hybrid model using both OpenMP and MPI.

24.3 Approach

The city is represented by a particle simulation. The particles exist in a finite 2D space where particles represent people living in the city and the space represents the city. The simulation moves forward at discrete time steps, where each time step t represents transitioning an hour forward in real time.

To create a simulated epidemic model of the city, various properties are added to both the particles and the organization of the city to help simulate how a real city would experience an epidemic.

24.3.1 Particles with Disease States

Each particle in the simulation will always exist in a disease state. The set of states, $\mathcal{S}$, includes Susceptible (S), Infected (I), Recovered (R), and Deceased (D), based off the SIR compartmental model (24.1).

$$\mathcal{S} \in \{S, I, R, D\} \tag{24.1}$$

We let N_t represent the number of total particles in the simulation at a given time step. All living individuals will be represented as particles in the simulation (24.2). Since deceased individuals will be removed from the simulation, N can decrease over time.

$$N_t = |S_t| + |I_t| + |R_t| - |D_t| \tag{24.2}$$

The number of initial particles in the simulation, N_0, is a variable that will be examined as a parameter to determine how larger population size (and thus higher population density) affects the disease transmission statistics. The simulation begins with all individuals except for 1 being susceptible (24.3), 1 infected individual, and no recovered or deceased individuals.

$$|S_0| = N_0 - 1 \tag{24.3}$$

Susceptible individuals have the chance to transition to the infected state at every time step. Infected individuals can spread the disease to susceptible individuals. The chance of infection from an infected individual to a susceptible one at any given time step is given by the probability of infection α. However, infectious particles can only affect susceptible particles if they are within a certain radius of infection β (24.4). Individuals must also be in the same area to infect each other, which will be discussed more in section B on City Organization.

$$P(\text{x infecting y}) = \begin{cases} \alpha & dist(x, y) \leq \beta \\ 0 & otherwise \end{cases} \tag{24.4}$$

Infected individuals can transition to either the recovered or deceased state with probabilistic functions based on a recovery factor γ and death factor δ, respectively (24.5), (24.6). These functions are also based on the time since infection, with a higher chance of either recovering or perishing the longer they are infected. These probabilities are tested at each time step.

$$P(\text{recovery, i steps after infection}) = max(\gamma i, 1) \tag{24.5}$$

$$P(\text{death, i steps after infection}) = max(\delta i, 1) \tag{24.6}$$

Recovered individuals are still part of the population but cannot transition out of the recovered state. Deceased individuals are fully removed from the population and simulation. The simulation ends either when time step t reaches some stopping point T or when there are no infected individuals remaining in the population, whichever happens first.

24.3.2 City Organization

The total space of the city is partitioned into units we call areas. Areas represent discrete regions where people stay, and they are each represented as equal size, square regions. There are both public and personal areas. Personal areas represent homes, and every person will have exactly one personal area that they visit. Public areas represent places that many people visit, such as schools, workplaces, or stores.

The number of areas is proportional to the number of particles in the simulation. For every 2 initial particles in the simulation there will be 1 personal area, which is a proportion that reflects housing to population data for United States cities. For every 4 personal areas in the city, there will be 1 public area. Figure 24.1 shows an example of the box distributions. The relative locations of areas do not matter in this simulation.

At the beginning of the simulation, a set of areas are assigned to each person. Personal areas are assigned such that every person belongs to 1 personal area and every personal area has 2 people. Public areas are assigned with more variability, with each being assigned to $N_0/10$ random people. These values are treated as constant controls as other parameters are investigated for their effects.

Each area that a person has assigned to them will also come with a probability p of transitioning there at any given frame.

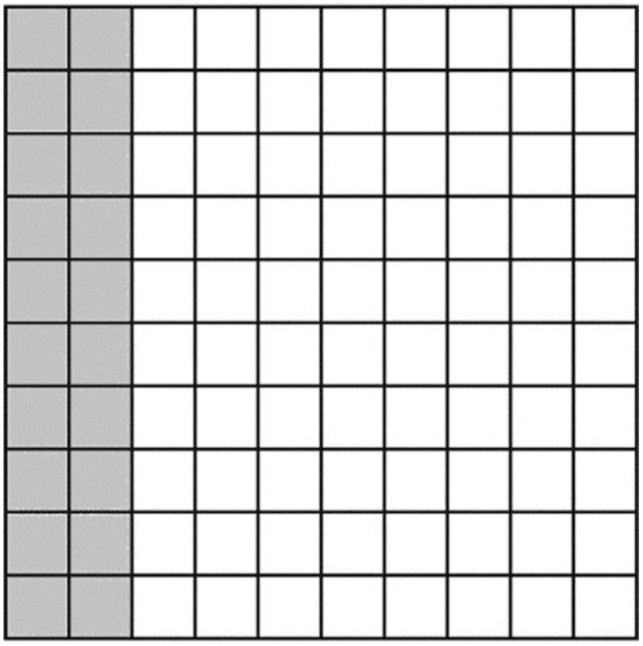

Fig. 24.1 Example 10×10 grid layout for a city with $N_0 = 160$. The shaded regions represent public areas and the unshaded represent personal areas

24.3.3 Simulating Movement of People

There are two types of movements that particles have in the simulation: movement between areas and movement within an area.

Movement between areas is considered an instantaneous jump from one area to another. At each time step, every person will have a chance p to jump to one of their assigned areas (assuming they are not already in that area). A random real value between 0 and 1 is uniformly drawn for every person's possible areas to jump to, and if the value is less than the probability, the jump will occur. The particle's position will be randomly chosen in the new area.

It is worth noting that since these jump timings are random, people do not have set "schedules" and can stay in areas for short and long amounts of time. There is also no notion of day and night. The values are chosen to best simulate people on average dividing their time evenly between time in public and their time at home.

An intervention policy that is tested to affect these probabilities is the lockdown. With a lockdown, the goal is to limit the number of jumps to public areas. This is accomplished by reducing the probability that any particle will jump to a public area. A parameter that will be tested to find the effect of this is the time of intervention implementation, τ. Changing this

Table 24.1 Parameters that will be investigated and tuned in the simulations

Symbol	Description
N_0	The number of initial particles
α	The probability of infection at a time step
β	The radius of infection for 2 particles
γ	The factor for recovery probability
δ	The factor for death probability
p	The probability of jumping to a new area
τ	The time of lockdown intervention

will be used to determine the success of implementing the lockdown early as to opposed to late.

Movement within an area occurs when a particle is not jumping between areas. The movement occurs by randomly picking an x and y value, each with a maximum magnitude. This type of movement is used to simulate people coming in proximity with one another to potentially spread the disease.

24.3.4 Parameter Estimation

To create a useful model, different parameters of the system must be chosen that help to describe phenomena that occur in real life. Table 24.1 shows a list of these parameters that will need to be decided. They will be decided using both logic and with respect to other parameters in the simulation. For example, the probability of infection is to be chosen by using the current knowledge that COVID-19 has a R_0 (Reproduction Number) number of approximately 2.2. A value for α will be decided so that the trend shows that around 2.2 particles are infected by a single infectious particle.

24.3.5 Parallelization Strategy

The simulation is implemented in C++ while utilizing the OpenMP and MPI libraries for both shared memory and message passing parallelization. Testing will be done on the Bridges supercomputing environment where we will access up to 16 nodes to test with at a time.

To distribute work evenly across the nodes used in the simulation, every node will get the same number of public and personal areas. Since the particles can jump to an area on any node, the communication of jumping particles must be done sequentially with each node having a chance to send its jumping particles while all others must be ready to receive.

Since relative position between areas doesn't matter, each area can be viewed as its own 2D space, and the relative (x, y) coordinate of each particle in the area, and the id of the area where the particle exists is sufficient information for processing.

24.3.6 Strategy for Measuring Success

Two approaches are used to measure the success of our simulation. The first is to analyze the infections, recoveries, and deaths that occur in the simulation. The second is to measure the impact of parallelization on the simulation's performance.

For the first approach of analyzing disease metrics, graphs of infections, recoveries, and deaths over time will be produced. Graphs will also be produced that measure the total number of infections, recoveries, and deaths that are caused by different values. This provides useful intervention insight in the case of altering τ, since we can measure how many lives can be potentially saved by how quickly the lockdown is implemented following the disease onset.

The second approach is to analyze the speedup induced by the parallelization of the simulation. Experiments for both strong and weak scaling will be performed by changing the number of total particles in the city to show that the hybrid approach scales well compared to a sequential implementation.

24.4 Results

One of our goals when running the simulation was to determine the effectiveness of parallelization using OpenMP, MPI, and then a hybrid of the two. Having a simulation run efficiently is important for its usefulness as a tool to better understand the spread of disease. First, we tested the application using only OpenMP for parallelization. We ran a simulation of 100,000 people using 1, 2, 4, 8, and 16 threads. We ran each of these simulations 10 times and used the average run time to determine the speedup of the simulation with OpenMP. Figure 24.2 shows the average time taken to run the simulation for each number of threads. Figure 24.3 shows the speedup of the simulation with each number of threads.

Next, we tested the simulation using only MPI for parallelization. Similarly to the OpenMP tests, we tested the MPI simulation with 100,000 people and 1, 2, and 4 processors. Unfortunately, due to resource restrictions with the Bridges supercomputer at the time of these tests, we were not able to run the simulation with more than 4 processors. We ran each of the simulations 10 times and used the average runtime to determine the speedup, strong scaling efficiency, and weak scaling efficiency. Figure 24.4 shows the average runtime of the simulation with different numbers of processors. Figure 24.5 shows the speedup of the simulation with an increasing number of processors.

We were able to see a much larger speedup when using MPI than when using OpenMP. Even with 16 OpenMP threads, the speedup was below 2.5 while 4 processors with

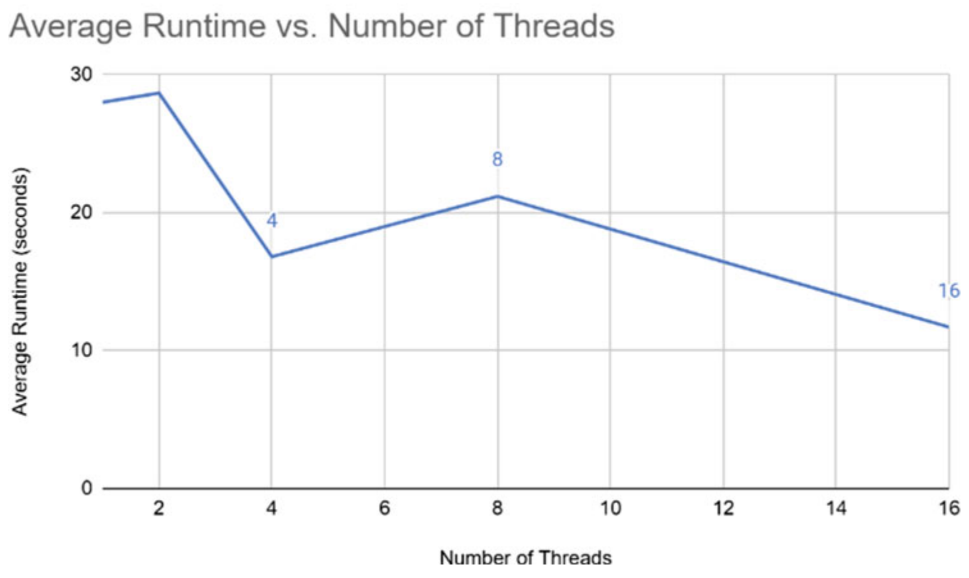

Fig. 24.2 The average runtime of the simulation tends to decrease with additional OpenMP threads

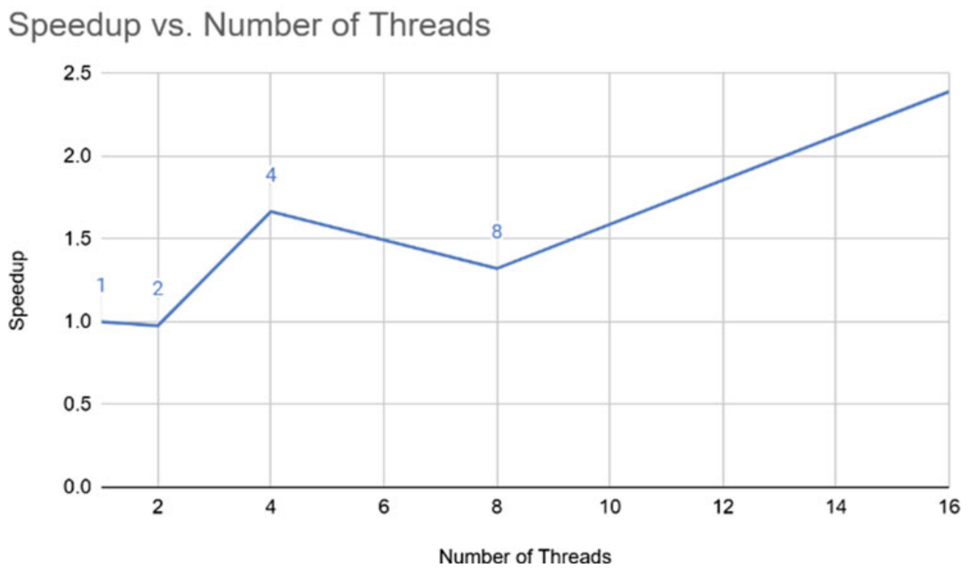

Fig. 24.3 The speedup of the program tends to increase with more OpenMP threads

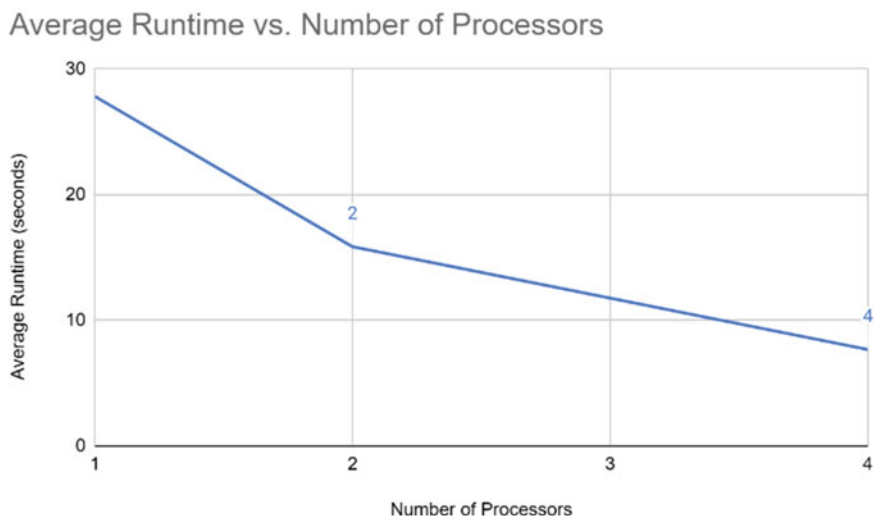

Fig. 24.4 The average runtime of the simulation decreases when more processors are added with MPI

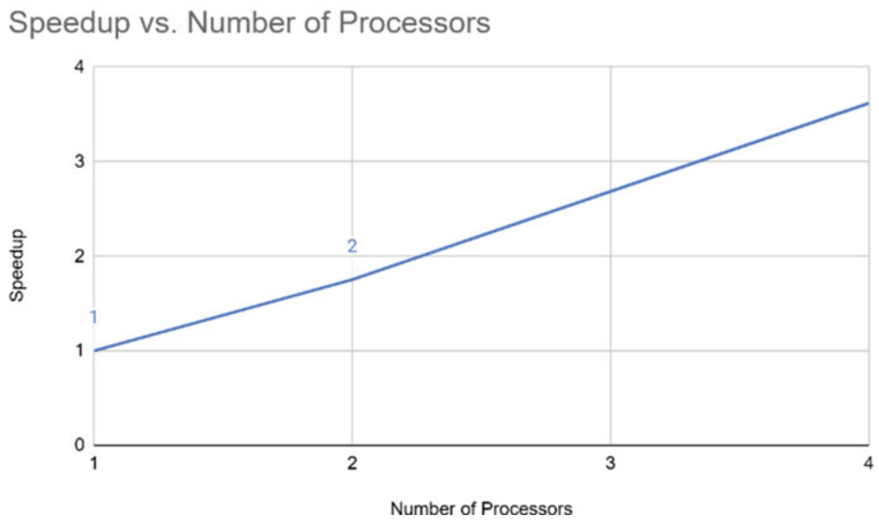

Fig. 24.5 The speedup of the program increases when more processors are added with MPI

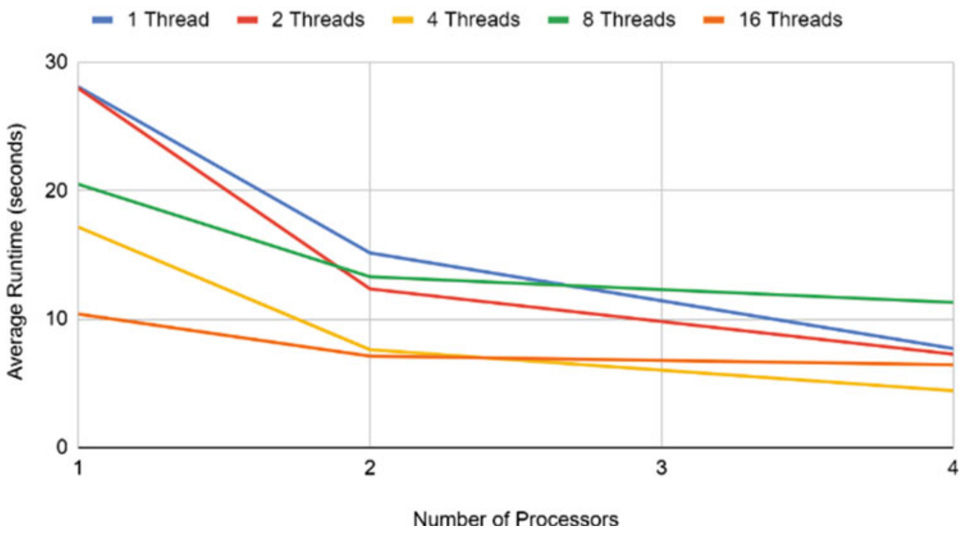

Fig. 24.6 The runtime of the simulation tends to decrease with more threads and more processors

MPI yielded a speedup of about 3.6. The strong scaling efficiency with MPI also stayed above 85% while the strong scaling efficiency of OpenMP dropped to about 42% with 4 threads and to about 15% with 16 threads.

The final test conducted on the efficiency of different parallelization methods was to use a hybrid model of OpenMP and MPI. We ran these tests with 100,000 people in the simulation. We ran the tests with 1, 2, 4, 8, and 16 threads, and 1, 2 and 4 processors. Figure 24.6 shows the resulting runtimes with various configurations of threads and processors. We found the best speedup with 4 threads and 4 processors, but overall, the runtime tended to decrease when more threads and processors were used.

Besides running tests on the efficiency of the simulation, we also ran tests to determine the effectiveness of a lockdown during a disease outbreak. In the simulation, a lockdown can be set to occur during a particular time step. After the lockdown occurs, people will be much less likely to jump to areas outside of their personal space. Figure 24.7 shows the number of people susceptible, infected, recovered, and deceased for each time tick when no lockdown was implemented. Figure 24.8 shows the same statistics but in this case, a lockdown was implemented about one quarter of the way through the simulation. Figure 24.9 shows the results of implementing a lockdown one tenth of the way through the simulation.

We can see from Figs. 24.7, 24.8, and 24.9 how a lockdown slows the spread of disease. With no lockdown, about 91% of the simulation population became infected at some time. Even when a lockdown occurred a quarter of the way through the simulation, about 91% of the population was infected with the disease. We saw the most dramatic difference when a lockdown was implemented one tenth of the way through the simulation. In this instance, only 20% of the population became infected with the disease.

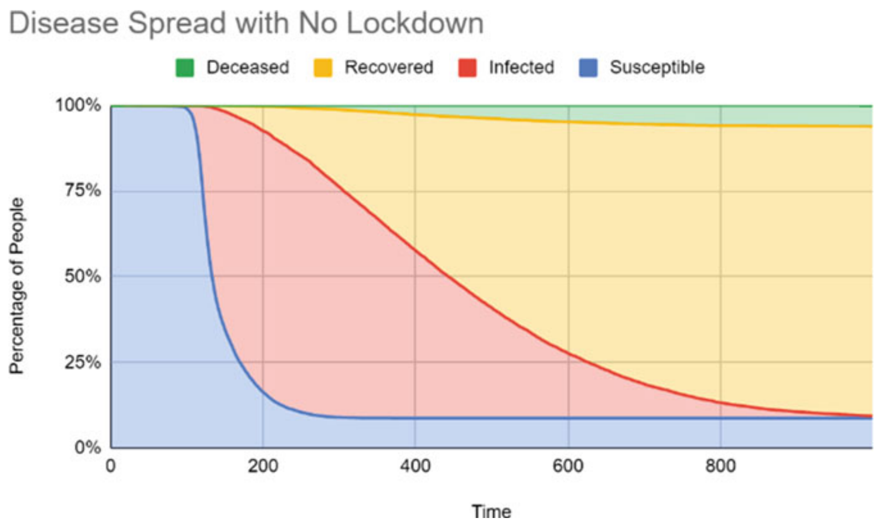

Fig. 24.7 State of the disease over time when no lockdown is implemented

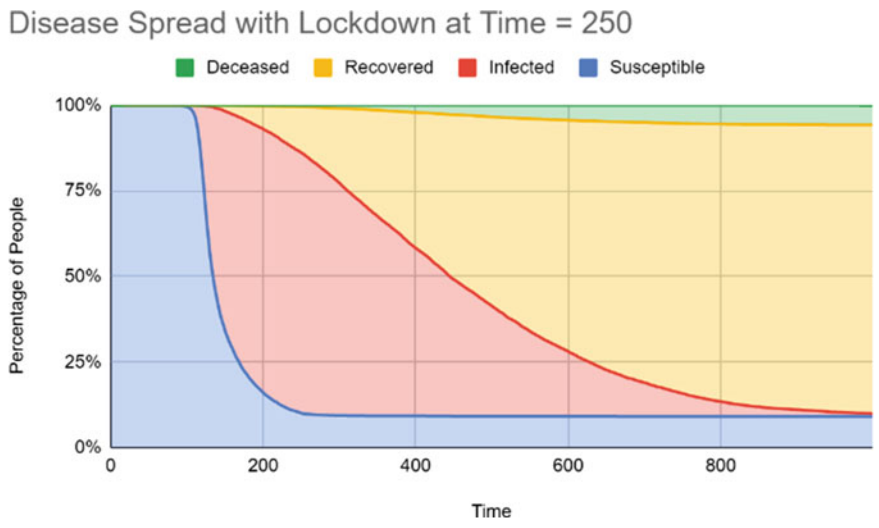

Fig. 24.8 Spread of disease when a lockdown is implemented one quarter of the way into the simulation

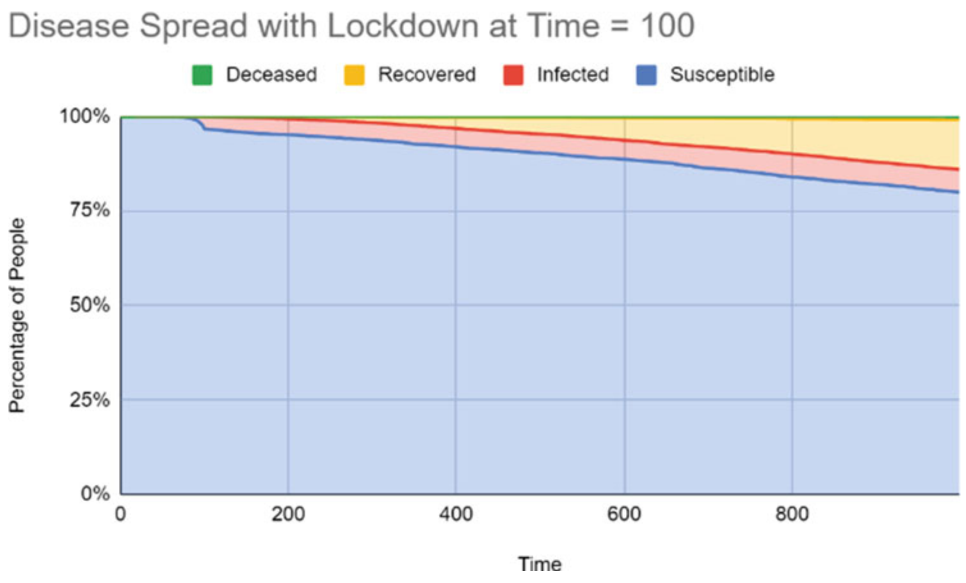

Fig. 24.9 Spread of the disease when a lockdown is implemented early in the simulation

24.5 Conclusions and Future Work

In conclusion, we found that a hybrid parallel model using both OpenMP and MPI yielded the best performance when running the simulation. We also found that using only MPI produced a much better efficiency than using only OpenMP. OpenMP however did improve the runtime over using the base serial implementation.

In terms of improving the performance of the simulation, future work could include implementing different approaches to parallelization. Other technologies such as CUDA could be used to implement the simulation, and then the results could be compared to our implementation. There are still many methods and platforms for parallelization that could be explored with this project [6]. This future work could help to discover if using MPI and OpenMP was the most appropriate way to implement the simulation. If allowed more resources in the future, we would also like to be able to test our simulation using more than 4 processors with MPI. We would like to be able to see how using 8 and 16 processors would improve the speedup of the simulation.

The results of the simulation showed just how important early action is when a possible pandemic is at hand. Early preventative measures proved to be very effective at reducing the number of people to be infected by a disease. Our data showed that waiting too long to try to stop the spread of infection can allow the disease to reach many people.

In the future, more work could be done to make the simulation a better representation of the world's population as our Coronavirus simulation was conducted on a homogeneous population. Possible future work into this simulation could include introducing genetic differences such as age or preexisting health conditions that would make a person more or less likely to die after being infected. Immunities could be introduced so that some people within the simulation are unable to be infected. Areas of different population densities could be introduced to see how the spread of a disease differs in a large community versus a smaller one. Another factor that could be added to the simulation is hospitalization. Infected persons would be isolated to one location therefore making them less likely to infect others and more likely to recover. Another interesting addition to the simulation would be to add the creation of a vaccine to the disease. This would allow us to see how a vaccine would help to lessen the spread of the disease as well as see how the lifetime of the disease is shortened. There exists a lot of future work that could be done to make this simulation a more accurate representation of the world's population.

Acknowledgments This material is based in part upon work supported by the National Science Foundation under grant number IIA-1301726. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

1. N. Becker, The uses of epidemic models. Biometrics **35**(1), 295–305 (1979). ISSN: 0006341X, 15410420 [Online]. Available: http://www.jstor.org/stable/2529951
2. W.O. Kermack, A.G. McKendrick, G.T. Walker, A contribution to the mathematical theory of epidemics. Proc. R. Soc. Lond. Ser. A Contain. Pap. Math. Phys. Char. **115**(772), 700–721 (1927). https://doi.org/10.1098/rspa.1927.0118. eprint: https://royalsocietypublishing.org/doi/pdf/10.1098/rspa.1927.0118 [Online]. Available: https://royalsocietypublishing.org/doi/abs/10.1098/rspa.1927.0118.
3. L.J. Allen, A primer on stochastic epidemic models: Formulation, numerical simulation, and analysis. Infect. Dis. Modell. **2**(2), 128–142 (2017). ISSN: 2468-0427. https://doi.org/10.1016/j.idm.2017.03.001 [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2468042716300495
4. T. Britton, M. Deijfen, F. Lopes, A spatial epidemic model with site contamination (2017). arXiv: 1705.07448 [math.PR]
5. T.C. Germann, K. Kadau, I.M. Longini, C.A. Macken, Mitigation strategies for pandemic influenza in the United States. Proc. Natl. Acad. Sci. **103**(15), 5935–5940 (2006). https://doi.org/10.1073/pnas.0601266103. eprint: https://www.pnas.org/content/103/15/5935.full.pdf [Online]
6. P. Czarnul, J. Proficz, K. Drypczewski, Survey of methodologies, approaches, and challenges in parallel programming using high-performance computing systems. Sci. Program. **2020**, 4176794:1–4176794:19 (2020)
7. M. Hipp, W. Rosenstiel, Parallel hybrid particle simulations using MPI and OpenMP, in *Euro-Par 2004 Parallel Processing*, ed. by M. Danelutto, M. Vanneschi, D. Laforenza (Springer, Berlin, Heidelberg, 2004), pp. 189–197. ISBN: 978-3-540-27866-5