Fabian Richardt

Evolutionary Policy Optimization in Small Communities with a Location-Based Epidemic Model

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

INF FAKULTÄT FÜR INFORMATIK

Intelligent Cooperative Systems
Computational Intelligence

Evolutionary Policy Optimization in Small Communities with a Location-Based Epidemic Model

Master Thesis

Fabian Richardt

November 19, 2021

Supervisor: Prof. Dr.-Ing. habil. Sanaz Mostaghim

Advisor: M.Sc. Dominik Fischer

Fabian Richardt: *Evolutionary Policy Optimization in Small Communities with a Location-Based Epidemic Model*
Otto-von-Guericke Universität
Intelligent Cooperative Systems
Computational Intelligence
Magdeburg, 2021.

Abstract

In the modern interconnected world epidemics like the Severe Acute Respiratory Syndrome Coronavirus-2 outbreak of 2019 are a global risk. Decision makers face the trade-off of containing infection spread versus the economic and social costs of containment measures. Simulation models to help with this decision often focus on either of two aspects: The individual level with contact between people or national scales to help policy makers.

We argue that small communities - universities, hospitals, villages - form an important middle ground between these extremes. They are a part of emergent infection spread on larger scales. At the same time small communities exhibit more social structures that the homogeneity of individuals assumed by classical models.

Using Otto-von-Guericke Universität Magdeburg as example community, we develop and implement a location-based epidemic model including costs for containment measures. The bipartite graph models individuals visiting locations according to simple generation rules. We optimize containment policy parameters using evolutionary multi-objective optimization for the goals of low cumulative infections, infection peak and cost .

Through exploratory experiments we find such optimization to produce valid parameter sets with interesting fitness characteristics. Along with in-depth analysis of simulation runs the optimization helps understand the model, which produces results in line with classical approaches and scenario assumptions. We conclude that the location-based model is a valid approach for scenarios like small communities where locations matter. The model is easy to extend with e.g. testing or vaccination and the modeling can be incrementally improved to better match the chosen scenario. It further helps with identifying locations serving as infection hot-spots. Lastly, a location-based graph can be transformed into a contact graph for individuals, making it viable as first step for other modeling approaches.

Contents

List of Figures

List of Tables

List of Acronyms

OvGU	Otto-von-Guericke Universität Magdeburg
SARS-CoV	Severe Acute Respiratory Syndrome Coronavirus
MERS-CoV	Middle East Respiratory Syndrome Coronavirus
HIV	Human Immunodeficiency Virus
A/H1N1	Influenza A virus subtype H1N1
SARS-CoV-2	Severe Acute Respiratory Syndrome Coronavirus-2
SIR model	Susceptible-Infected-Recovered model
ODE	ordinary differential equation
OSN	online social network
LBSN	location-based social network
ABM	agent-based modeling
EA	evolutionary algorithm
NN	neural network
EMO	evolutionary multi-objective optimization
NSGA-II	Nondominated Sorting Genetic Algorithm II
NSGA-III	Nondominated Sorting Genetic Algorithm III
MOEA/D	Multi-objective Evolutionary Algorithm Based on Decomposition
DEAP	Distributed Evolutionary Algorithms in Python

1. Introduction

The specter of facing a large scale catastrophe is as old as humanity - be it wars, natural catastrophes or plagues. Plagues are especially relevant considering the current Severe Acute Respiratory Syndrome Coronavirus-2 (SARS-CoV-2) pandemic, which started in December 2019 [74] and is still ongoing in 2021 [75]. It lead to upheavals in large parts of society through the medical sector trying to get ready for it and containment measures from wearing masks [47] to a full "lockdown" [15, 86]. At the same time scientists work to overcome the crisis: Researchers characterize SARS-CoV-2 symptoms [74] and recommend countermeasures to the disease [41, 85, 47]. They also work to create effective vaccines [65, 75].

On the other side, scientists employ models and simulations for understanding the epidemic [34, 25]. These models allow prognoses about the epidemic spread on national and international scales - to predict necessary containment measures [66, 27]. Real world policy makers face another problem: Most countermeasures taken come at a cost, be it economical [46, 2] or social [71]. Epidemic models need to consider this trade-off if they are to help policy makers [64].

While there is a wide array of epidemic models, most rely similar basic assumptions: Susceptible-Infected-Recovered models (SIR models) [40] for example assume the population to be homogeneous and that every susceptible individual has the same chance to infect themselves (the so called perfect mixing assumption). Such models often fit real world data well and are simple to build [34, 51]. However, they cannot capture all interaction patterns between people. For that reason there are graph- and actor-based models [63, 38] that map out the interactions between people or groups explicitly. Even those are not always an intuitive model, especially if there is no prior data to use for contact graphs [58].

We argue that there are scenarios - namely small communities - where neither modeling approach is the optimal choice. As examples for small communities, consider small towns, hospitals, factories, ships [85] or universities [76, 32, 57]. These examples form a step between modeling interaction on an individual level and nation-scale simulations. Nations and other large-scale populations consist in turn of smaller communities. For this reason we expect the characteristics of epidemic spread in smaller populations to give insight into large-scale behavior.

Small communities not only exhibit a smaller number of people compared to nation-scale models. Due to the smaller size, the interaction dynamics at play are easier to understand and interaction structures often shaped by locations and group habits. Thus, homogeneous models with their perfect mixing assumption are a poor fit. At the same time individual contact graphs are counter-intuitive. Consider a student at university: Their daily routine consists of, for example, going to lectures, eating in the cafeteria or exercising in a sport course. Instead of deciding to meet specific people, they often visit locations or attend events.

This observation leads us to propose a *location-based epidemic graph model.* We model interactions via a bipartite graph of *people visiting locations*[1] at specific times. Our chosen example scenario is a simplified version of the Otto-von-Guericke Universität Magdeburg (OvGU) campus over the course of one semester. On one hand, a university campus is the small community we are most familiar with, helping us build the model and gauge experiment results. On the other hand, university puts the trade-off between gains and costs of epidemic containment policies [64] into stark relief. Students and researchers pay the same social and emotional toll for a "lockdown" as the population at large [76, 32]. At the same time they are the people society expects to find solutions to current and future epidemics.

Finding good trade-offs to this dilemma forms the second half of this thesis's contribution: We use evolutionary multi-objective optimization (EMO) [44] to optimize a set of three containment policy parameters. These are the maximal number of attendees for a lecture ($k_{lecture}$), the number of people allowed into cafeteria at the same time (k_{food}) and maximal size of sport courses (k_{sport}). The optimization goal is to find parameter sets (also called individuals) with the best fitness. In our case this means a minimal number of cumulative infections ($f_{cumulative}$), a low infection peak (f_{peak}) and least overall cost of disallowed visits (f_{cost}). Aside from maybe finding good policies for use in the real

[1]"Locations" can in this case be logical groupings like study groups or other events as well.

world, the main desired outcome of the optimization is a deeper understanding of the simulation model.

First, Chapter 2 offers an overview over related work. It explains approaches to epidemic simulation, relevant concepts of EMO and graph theory. Chapter 3 describes the model, from the basic graph model and the spread of infections over time, to modeling policies with their costs and goals. Then follows Chapter 4 with the actual implementation of the model and optimization framework.

In Chapter 5 we lay out the experiments conducted in three stages. The first stage focuses on initial optimization runs in order to choose a good EMO algorithm for further experiments. Next, the second stage expands on this different model sizes and objective combinations. Lastly, there is a third stage with more extensive simulation runs for policy parameters that showed interesting characteristics in the previous stages.

The next Chapter then evaluates the experiment data, following the structure of the previous Chapter with a Section for every stage followed by a summary in Section 6.4.[2] Lastly, Chapter 7 wraps up the thesis with a summary and an overview over possible future work using the proposed model.

After experiments and their evaluation we find that - for the limited number of experiments run - the model produces results in line with classical models and our scenario assumptions. The simple structure of the location-based graph model allows for a clear road to model extensions (e.g. testing, vaccinations) or improvements where modeling turns out to be imprecise (e.g. surprisingly few infections in the cafeteria). We further find that the gathered data on infected and susceptible people visiting each location yields valuable insights into epidemic spread. In the simulation runs analyzed in-depth we did not find "superspreading" [49] locations. Instead we found the infection wave driven by the sheer amount of meetings with infected distributed over the different locations types. This observation matches the assumptions for the university scenario, which did not include events expected to function as superspreaders.

For the optimization experiments we choose Nondominated Sorting Genetic Algorithm III (NSGA-III) [24] as the main EMO algorithm. We find that optimization yields valuable data on the model, as well as policies to take a closer look at. In deciding the fitness of a policy parameter set we find running

[2]Due to the structure of Chapters 5 and 6 we recommend reading them interleaved by alternating sections from each.

the simulation 23 times per individual to yield dependable results. Any lower sampling rate distorts the Pareto front (see Section 2.2 for a definition) of the experiment. Furthermore, both $f_{cumulative}$ and f_{peak} conflict with the cost objective f_{cost}, making EMO a suitable approach for finding optimal policies. Despite that we also conclude that there is no gain in using all three objectives at the same time.

In summary the location-based epidemic model and optimizing policies via EMO works well together. Starting from simple rules the model can be improved and understood incrementally. This advantage makes further work on the model valuable and the approach transferable to other scenarios. Beyond that the location-based model can be used as a first step for classical models by using it to generate contact graphs between individuals.

2. Related Work

2.1. Epidemics, their Modeling and Simulation

The modern world is more interconnected than ever before. Instant communication and the internet enable ideas and cultures to mix across the globe [53]. Cheap travel is even more important for the topic of epidemiology [28]. Trade goods are shipped across vast distances and vacations in different countries are affordable for many people. Along with that the reach of highly infectious diseases increases dramatically [63, 1]. Pandemics - global epidemics - thus are a serious threat to human lives.

Looking at the last decades there are several significant epidemics [62]: Human Immunodeficiency Virus (HIV) (1981) for which there is still no complete cure; Severe Acute Respiratory Syndrome Coronavirus (SARS-CoV) (2002), Influenza A virus subtype H1N1 (A/H1N1) (2009), Middle East Respiratory Syndrome Coronavirus (MERS-CoV) (2012) and Ebola (2013); SARS-CoV-2 (2019) leading to a still ongoing crisis with a race between vaccinations and virus mutations in 2021 [65, 75].

Looking at these epidemics [62] we note that so far the more lethal infections (Ebola with $>$50% fatality rate [62]) were limited in the size of their outbreaks. SARS-CoV-2 on the other hand is much more infectious, but with a lower fatality rate [38]. The need to prepare for the next pandemic becomes obvious when considering the possibility of a disease combining these two problems. Aside from political or pharmaceutical measures this means creating and evaluating models to understand and predict epidemic spread and the effectiveness of countermeasures [15, 38, 81, 19, 34]. This fact remains true from fitting these models to real-world data to understanding an ongoing epidemic. However, researchers often face the problem of sparse real-world data [78, 56]. In addition to that there is the problem of choosing the right model, with different approaches having different trade-offs. A good model then allows to ask "What if?" by e.g. increasing infection or fatality rates.

Throughout this thesis we use the current SARS-CoV-2 outbreak as the main example. We keep modeling more general, but with infection and fatality rates similar to SARS-CoV-2. A lot of new papers regarding models of epidemic spread are published (e.g. [15, 38, 81]) due to the current pandemic. These publications include pitfalls like draft-stage papers showing up online or duplicated research through similar approaches. We provide structure and context to the fast-expanding literature by separating the next Subsections by modeling approach.

2.1.1. The Classic SIR Model and Epidemic Spread

The SIR model is the most basic epidemic model. Kermack et al. [40] first proposed it in 1927. The basic idea of the model starts with a population of susceptible people (denoted S). Some sick individuals are introduced, which form a group of infected I. It is assumed that people are homogeneous and mix perfectly [58], leading to the same infection chance for every susceptible individual. After some time infected stop being sick, filling the recovered group R.

There are several common variants to this model. By adding a group of dead people, who did not recover from the infection, one gets a SIRD model (Bailey, 1975 [7]). Instead of recovering after an infection, there is the option to become susceptible again like with the seasonal flu. Last but not least, most sicknesses have an incubation period, which is usually modeled via an Exposed group E. The last two versions are called SIS and SEIR respectively [13].

There are several examples of SIR models regarding SARS-CoV-2: Chatterjee et al. (2020)[19] build a SIRD model to predict infection peaks. Starting with cumulative infection data from India and other states, they fit their model to this data. Then Chatterjee et al. use the SIRD model to make predictions regarding the effects of relaxing lockdown too early for the states in question. Yang et al. (2020)[81] look at the original SARS-CoV-2 outbreak in China. It coincided with a national holiday including a lot of travel [81]. They check if the Chinese containment measures were effective using a modified SEIR model that can represent travelers. Lastly, not all cases of SARS-CoV-2 show symptoms [74]. Ivorra et al. (2020)[34] incorporate this into their model via a H group of hidden, unreported infections. Using this model allows them to attempt predictions for needed beds in hospitals, as well as gain insights into epidemic spread by varying the rate of hidden infections.

The mathematics underlying SIR models is a system of ordinary differential equations (ODEs), which has both advantages and disadvantages [40, 58]: Evaluating such a system via numerical algorithms is usually rather fast. However, the basic model lacks many details that become relevant in the real world. Adding those details makes the ODE system much more complex. A total lockdown is a simple example that shows this problem. It can be implemented by cutting the infection path from S to I, making the equations actually simpler. But total lockdown is not feasible for long, making consideration of partial lockdown scenarios necessary. Compartments with different infection rates are one potential solution to modeling different levels of contact graph connectivity [58]. Vrugt et al. (2020)[70] go even further and model social distancing and quarantine using dynamic density functional theory. This highlights that any significant extension of the SIR model complicates the underlying mathematics. Despite that Vrugt et al. [70] claim to prefer the extension to the SIR model to alternatives due to its fast evaluation speed.

Aside from classical numerical methods for solving ODE systems (like the Runge-Kutta methods [17]), there are more specialized solvers for SIR models. Chemistry inspired several algorithms for this, most prominently the algorithm by Gillespie and its derivatives [31, 73, 45]. They work for SIR models, because ideal chemical reactions work on a perfect, homogeneous mixing assumption as well [31]. Special about Gillespie's algorithm is picking the size of the next time-step dynamically depending on how fast the underlying population changes currently. Another example is Block et al. (2020) [15], who use a Monte-Carlo method where one individual randomly meets one other each (small) time-step. For our university scenario, neither approach makes much sense. Instead the real world gives us intuitive time-steps in the form of lecture time-slots. For more details see Section 3.2.

2.1.2. Social Networks and the Topology of the Spread

Other modeling approaches offer different trade-offs and can be more intuitive depending on the problem definition [58, 80]. Looking at people and their interaction - which are how infections spread - both homogeneity and perfect mixing are flawed assumptions with regards to a population's real-world behavior [38, 20]. Instead a real population is composed of heterogeneous individuals interacting with other individuals [38]. One common way to model this behavior is via a *contact graph*, often called a *social network* in literature [77, 18].

The field known as social network analysis has grown a lot in the last two decades [18]. Online social networks OSNs are the main reason for this, because they grew into an important part of society and provide ample data for different kinds of research. To give an example of the use of simulated OSNs [18]: Gatti de Bayser et al. (2014) [30] model the spread of information via micro-blogging. Their model allows heterogeneous individuals by learning behaviors via individual Markov chains. Very similar to the spread of general information is analyzing dissemination of fake-news [18].

Despite not thematizing epidemic spread directly, this research is relevant to our problem. This is because both normal contact graphs between people and an OSN form clusters between individuals in the same manner [55]. Imagine for example groups of university students with similar interests meeting on the campus lawn in the sun or in an OSN of their choice if it's raining. Research into different kinds of networks confirms this observation: Newman et al. (2003) [55] find that social networks are very different from other networks appearing in the natural sciences. Most importantly they observe that a social network forms clusters significantly more often then randomness would suggest.

However, small world networks [9] and other network types typically used for abstract contact graphs [82] are often insufficient to model the complexities of the real world [20]. Resorting to more expressive network types helps to keep a better overview over different kinds of social interactions. Multiplex networks - basically a set of several overlapping networks that form a greater whole [20, 42] - are one example. Chung et al. (2020) [20] use them to better model SARS-CoV-2 spread in a SEIR model. Their main idea is to connect every significantly different form of social interaction as a separate sub-network. This approach works for our university example as well: Student households, lectures, tutorials, cafeteria use and similar then form their own smaller social networks.

When modeling small communities, we observe another effect: People do not only hang out with their social circles, but gather depending on locations. Thus, even without using complex network types lectures, eating at the cafeteria and similar will form clusters on the contact graph. In the case of OSNs such a network would be called a location-based social network (LBSN) as described by Kavak et al. (2019) [39]. They mention that such networks form a bridge between digital networks and the physical world via image tagging, geolocation and similar applications. Since they allow peer-to-peer interactions and spatial relations between locations, the network model of Kavak et al. [39] is more complex that the one in Chapter 3. Using location data in OSN analysis faces problems with acquisition of the relevant data and privacy concerns [39]. Because of that we model the university scenario more abstractly.

Similar to solvers for the ODE representation of a SIR model, algorithms for evaluating epidemic spread on contact graphs exist. Simple approaches look at single time-steps. For every infected that meets a susceptible individual during that time-step, propagate the infection with a fixed infection rate. After that each infected recovers with a recovery rate as probability. Antulov-Fantulin et al. (2018)[3] call this approach NaiveSIR. It can be easily parallelized via running several simulations in at the same time and works on *any* kind of contact graph.

Beyond that Antulov-Fantulin et al. suggest a faster algorithm, which they dub FastSIR [3]. It exploits the underlying probabilistic variables of the NaiveSIR approach by noticing that the number of individuals infected by one carrier is binomially distributed. Thus, instead of checking every meeting between infected and susceptible, they sample the distribution once for every infected. The result of sampling determines the number of new infected, which are sampled from the node's neighbors. When the distributions are cached, this method is much faster than NaiveSIR. This approach even allows an easier visualization of the "generation of each infected. Note that the number of meetings between individuals needs to be known for FastSIR [3]. In case of a dynamic graph (esp. for the bipartite graph we model in Chapter 3) the speed gains would likely be invalidated by having to re-calculate the number of meetings every time-step.

More examples include Tolić et al. (2018) [72] who use weighted shortest paths to estimate the likely spread of an infection through the network. Like FastSIR, however, they assume a static contact graph.

2.1.3. Agent-based Modeling - Autonomous Heterogeneous Actors

Going further from SIR and graphs agent-based modeling (ABM) forms the other end of the modeling spectrum. See Macal et al. (2010) [48] for an introduction to the topic. Autonomous agents that interact with each other to form complex group behavior characterize ABM [48]. Such agents are often simple, but heterogeneous [48]. Relationships between them (similar to social networks) and the environment in which they act often influence their behavior [48]. In complexity these models range from relatively simple to very large and complex. To mention some of the examples mentioned by Macal et al. [48]: Modeling the stock market, supply chains, ancient civilizations and of course epidemic spread.

Bankes (2002) [8] makes a case that ABM is a natural ontology for many social problems superior to ODE modeling approaches. He further mentions emergent behavior of groups of agents as a distinct advantage of ABM. One example in regards to epidemic spread is the ability of identifying super-spreaders [69] - agents that infect significantly more susceptible people than average [49]. Compared to SIR models and (to a lesser degree) social networks there is another advantage: We do not need real-world data or known contact graphs to model realistic behavior [69], since it emerges from simple local agent behaviors.

However, an ABM approach has downsides as well [58, 69]. Despite not needing a predetermined contact graph, agent behavior still needs to be modeled in a way that produces useful - meaning realistic - results. Without a contact graph or real world data validation of emergent behavior becomes difficult. Rahmandad et al. (2008) [58] note that additionally ABM approaches often have more and more sensitive parameters. In practice this means more work to understand the model and how the chosen parameters affect it. Rahmandad et al. further mention that there can be simulation runs where the epidemic "fizzles out" due to initially infected agents not meeting enough people [58]. Due to all these considerations together more simulation runs are needed to get dependable results for an ABM approach than for competing approaches [69].

2.1.4. Consideration of Modeling Approaches

Before starting to design a model for our university scenario in Chapter 3 we compare the modeling approaches from the last Section: On the spectrum from SIR models to ABM, we note that the contact graph and ABM approaches share many similarities. Thus, we first compare an ODE based SIR model approach to its alternatives in general.

Integrating our small community assumption into a SIR model is more complex that for the more bottom-up modeling approaches. A suitable compartment model [58] might be able to represent our university scenario. At the same time it would curtail the ability to identify superspreaders or critical locations. Rahmandad et al. (2008) [58] find that the results of between the different modeling approaches are often similar enough to not make a difference. However, they further find that the modeling approaches diverge strongest when the perfect mixing assumption of the SIR model is violated. Since the university scenario as well as other small communities (e.g. hospitals) are often clustered around specific locations (lecture halls, hospital rooms) this becomes relevant in our case. To summarize: For our model a contact graph or ABM approach offers a better trade-off than the classical SIR model.

In summary, graph-based approaches have several advantages: There are simple and efficient algorithms for evaluation [72, 3]. Some of those even generate statistically reliable data with only one iteration of the simulation [72]. Visualization of the graphs and results is straight-forward. Further, the existing topology allows for easy analysis of topological interdependencies. Disadvantages of graph-based approaches include that algorithms are built with the assumption of a peer-to-peer contact graph. Constructing such a graph does not work well with the location-centric view of our small community assumption. More importantly, almost any algorithm assumes a static contact graph [72, 3]. In our community example students visit different lectures each day and self-quarantine on getting sick, making the resulting graph highly dynamic. Integrating these changes into algorithms optimized for static graphs is too much work. Instead we focus on simple algorithms like NaiveSIR [3] for initial exploration of our model.

On the other hand, an ABM approach simulates - depending on the level of detail chosen - locations via the environment. Agents then move around in them [48]. Building such an (abstract) environment from lecture lists and other typical university locations is straightforward. However, agent behavior still needs empiric data and cannot be expressed as simple rules like movement on a graph. In the same vein simulation and visualization are more complex than for a graph-based approach. Aside from spatial relations between locations we lose the ability to analyze the connection graph's topology. Due to this we need to gather run-time data manually and aggregate it. Most importantly, containment policies can no longer depend on the graphs topology, making them more complicated.

We choose a solution that lies between the contact graph and ABM approaches. Chapter 3 outlines a bipartite graph, which connects locations and logical groups to people visiting them. This modeling allows expressing the problem with the ease of an abstract, low detail ABM model. At the same time we retain the advantages of a graph-based solution. In particular, visualization of the graph stays simple and agents in the simulation only "move" across the graph. To model infection spread we choose the Naive SIR algorithm by Antulov-Fantulin et al. (2018)[3], since it is simple and works out of the box for a dynamic graph. See Section 3.1 for details. Policies can still depend on how locations are visited (e.g. reducing the number of visitors), which is close to real-world considerations [57]. Last but not least, the implementation outlined in Chapter 4 takes advantage of this, since bipartite graphs are easier to represent and optimize than general ones.

2.1.5. Characteristics and Trade-offs of Containment Policies and Goals

In order to talk about containment measures for epidemics and their goals, we need to first define what a *policy* is. In this thesis let a policy be a measure taken to reduce epidemic spread in some form., most notably by constraining the contact graph. We give a more exacting definition in Section 3.4.

Since we want to model a university scenario, the OvGU crisis plan[57] is a good example of policies employed in the real-world. University leadership created a plan with several stages to better handle the current outbreak of SARS-CoV-2. It includes recommended measures for every stage of escalation. Without going into too much detail, we find several areas where policy decisions are made.

- Firstly, the formation of a crisis unit for coordination and management.
- General hygiene rules, room sanitation and similar are implemented. Note that such measures are difficult to fit into the more abstract model

designed in Chapter 3. Our model implements better hygiene conditions via lower infection chances.

- Policies with the most direct impact are the guidelines for home office and digital or hybrid teaching.
- Due to the missing infrastructure for the implementation of policies from the previous point, procurement and setup of communication equipment is facilitated.
- In practice, students and staff often have to register for room use via calendar entries or scanning QR-codes to enable contact tracing.

However, all of these measures are associated with drawbacks [2, 64]. Examples include straightforward monetary investments (for communication equipment) and didactic problems due to digital lectures. Also, students potentially face psychological problems due to social isolation, see Thakur et al. [71]. These costs are difficult to quantify, making a more abstract model necessary, where we assume that overall cost of a policy correlates with the edges cut from the contact graph.

In summary, containment measures have three main goals: Firstly, minimize the number of overall deaths due to the epidemic or in general keeping overall infections down. Secondly, "flattening the curve" which means reducing the amount of infections at any one point in time, thus reducing strain on the healthcare system [64, 4]. This includes delaying the infection peak as well, giving hospitals more time to prepare. Last but not least, keeping the cost of containment measures down [64]. For details and the integration of these goals into the model, see Section 3.4.

2.2. Evolutionary Multi-Objective Optimization

The containment goals for a hypothetical epidemic as outlined in the previous Section are mutually exclusive [64]. As an example: The cost optimal case of "business as usual" leads to a lot of infections. On the other hand, total lockdown reduces infection spread, but costs increase along with that. Such optimizations with conflicting objectives are typical for real-world problems [44]. One method to find a set of interesting and optimal solutions is evolutionary multi-objective optimization (EMO). An introduction to it - including the underlying evolutionary algorithm (EA) principles - can for example be found in "Computational Intelligence: A Methodological Introduction" by Kruse et al. [44]. The book further serves as basis of this Section.

A background and definition of "optimal" for this case is needed, before considering a method to optimize the problem at hand. Formally, we define a *search space* S which contains the parameters to optimize [44]. In our example this could be the maximum number of people per lecture hall or seats available for the cafeteria. See Subsection 3.4 for the final search space. For every set of parameters we can then calculate values for our objectives (see the goals in the previous Section). They form an *objective space* O.

Taking two $o_i \in O$ with their respective $s_i \in S$ the next problem is deciding which one is objectively "better". For this the concept of *dominance* is introduced [44]: o_1 dominates o_2 if it is as good as or better than o_2 in every objective and strictly better in at least one of them. For an example objective space for a minimization problem, see Fig. 2.1.

Using the definition of dominance we call every individual not dominated by any other individual *Pareto optimal* [44]. The set of all Pareto optimal solutions forms the *Pareto front*. Any EMO algorithm strives to find this front. A final decision which solutions in the front have the subjectively best trade-offs or interesting characteristics is made after the optimization. Thus, EMO methods are *a-posteriori* approaches.

However, finding optimal solutions is not the only criterion for a good EMO algorithm [44]. Imagine all solutions being clustered on top of each other. Making an informed decision becomes a lot harder in this case, because the shape of the Pareto front and thus important characteristics of the problem can be obscured. Due to this, *diversity* is another goal. It refers to the goal of solutions distributed evenly across the entire Pareto front, .

Depending on the size of the population the Pareto front can contain a lot of solutions, making it hard for decision makers to pick the right one. There are several approaches to limit the number of optimal solutions or find interesting ones (e.g. ε-Dominance [33]). The method relevant for Chapter 6 is *knee points*. Fig. 2.1 shows one such point. Informally, it is characterized by protruding from the rest of the Pareto front. If we chose any of its neighboring solutions, at least one objective would become significantly worse. One way of formulating a mathematical basis for this uses the angles between each solution and its neighbors. See for example Branke et al. (2004) [16] to get an idea of how to modify an existing EMO algorithm to use knee points.

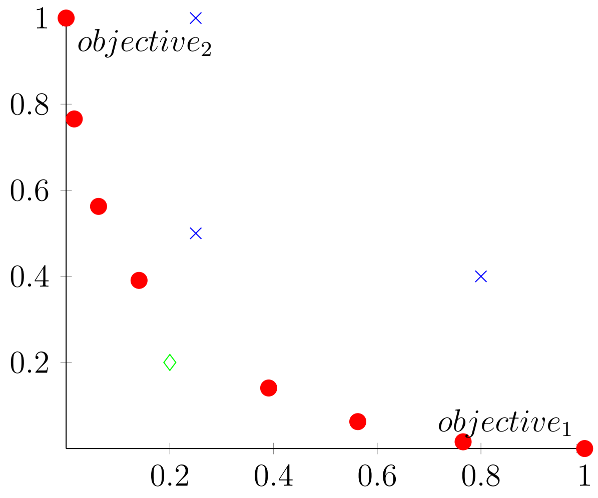


Figure 2.1.: Example objective space of a two objective optimization problem. The Pareto Front is marked with red dots, a potential knee point as part of the Pareto Front as green diamond and dominated solutions in blue crosses.

In the case of EMO approaches biological jargon is typically used [44]: We call the set of parameters of a solution the *genome* of an *individual*. Every individual has a *phenome*, which is the model used to evaluate it (e.g. epidemic simulation with the respective parameters). Phenome evaluation then yields the individual's *fitness*, which is its objective values. Optimization itself happens via creating a *population* P of individuals and performing *selection* (keeping only good solutions around), *mutation* (change genomes in small ways to get similar but different solutions) and *crossover* (combining solutions). Examples for EMO algorithms can be found in the next Subsections, starting with Subsection 2.2.1. These Subsections also give an overview over the algorithms used in Chapter 4. Their general scheme is very similar most of the time, and can be found in Algorithm 1.

Algorithm 1 Basic evolutionary algorithm

```
Initialize population $P$
while Termination condition is not met do
    Select new individuals for reproduction
    Create new population $P'$ via crossover
    Mutate individuals in $P'$
    Select new population from $P \cap P'$
end while
```

2.2.1. Nondominated Sorting Genetic Algorithm II (NSGA-II)

In Algorithm 1 both crossover and mutation depend heavily on the problem [44]. Especially, the encoding of solutions is relevant for them, while the number of objectives matters less. For this reason EMO algorithms focus mainly on the selection operators that choose which solutions are crossed over and which remain for the next generation. One of the most common EMO algorithms is NSGA-II by Deb et al. (2002) [23].

Two core concepts form the population selection operator of NSGA-II [23]: A fast algorithm for *non-dominated sorting* and *crowding distance* to preserve solution diversity. The former focuses search towards optimality by separating the population into *non-dominated fronts.* All individuals that are currently non-dominated form the first front. Subsequently, all other individuals are considered and those from the first front disregarded for the moment. The second front then consists of the non-dominated remaining individuals. This process is repeated to separate the entire population into fronts like in Fig. 2.2. Note that the implementation in the original paper [23] contains a version of this procedure optimized for speed.

After having generated new individuals via crossover and mutation the population for the next generation is formed by taking the best fronts one after the other until the population limit is reached [23]. Crowding distance comes into play if there are fewer slots left than the current front has individuals. It aims to keep solutions as diverse as possible to not converge on only one part of the search space. The metric measures the distance between a solution and its nearest neighbors in each objective. Larger crowding distances imply an "alone" solution, making it potentially interesting. In case of a solution being a minimum or maximum in one objective its crowding distance is set to

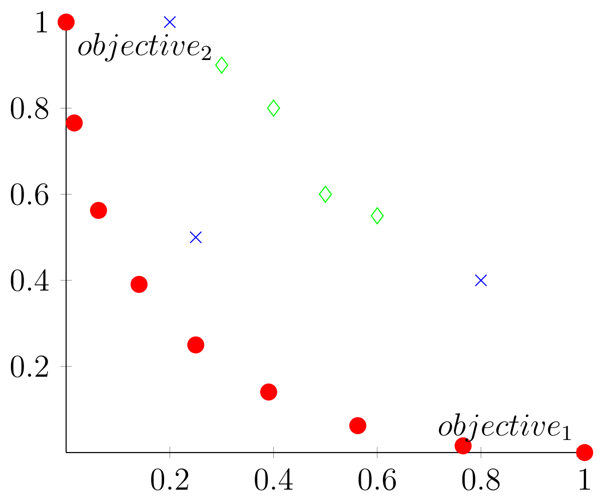


Figure 2.2.: An example objective space sorted into non-dominated fronts. The first front consists of red dots, the second of blue crosses and the last of green diamonds.

infinity. For all other solutions the normalized average distance between its neighbors is used instead.

Together these two mechanics form a selection mechanism that is effective for most basic EMO problems [23]. In summary, NSGA-II is a typical "go-to" algorithm for EMO. It balances diversity and optimality as goals, while being simple to understand and efficient.

2.2.2. Nondominated Sorting Genetic Algorithm III (NSGA-III)

If an optimization problem has more objectives, classical EMO approaches like NSGA-II start to struggle [24]. With too many objectives there are suddenly far more non-dominated solutions. Measures like crowding distance become less effective and efficient. Such problems with four or more objectives are called *many-objective optimization problems*. Research into these problems is distinct from multi-objective optimization due to the modified approach necessary. One example of changing an EMO algorithm to work on many-objective problems is NSGA-III, again by Deb et al. (2014) [24][1]. It replaces crowd-

[1]Do not be confused by the "Part I" in the paper's title. There is a follow-up paper[36] that deals with constraint handling with regards to NSGA-III.

ing distance with an approach using reference directions that works better for more objectives.

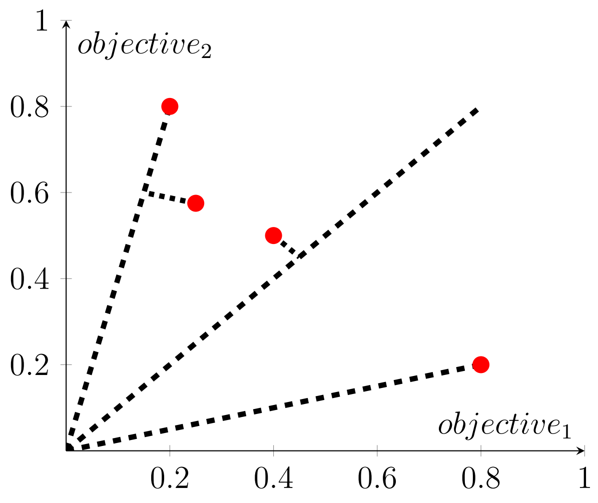


Figure 2.3.: An example with three reference lines in a problem with two objectives. In this case the reference lines are evenly distributed between the outermost solutions. Each solution (red dot) is associated with the closest reference line.

For an example how such reference lines look see Fig. 2.3. The original paper [24] also suggests the approach of reference directions distributed uniformly across all dimensions of the objective space. However, if we have special knowledge of the problem domain, other distributions can make sense as well. Each solution is then associated with the closest reference line.

Once this is done solution selection begins. Reference directions with fewer associated individuals are less explored and more interesting [24]. Thus, NSGA-III chooses the closest solution to each reference line first, starting with the least populated directions. If more individuals need to be selected, the reference lines are iterated again and again. From the second iteration the algorithm chooses random individuals associated with the selected reference in the hope of picking solutions between lines.

As an aside, the problem formulated in Chapter 3 (see esp. Subsection 3.4.1) is not a many-objective problem due to having at most three objectives. Due to the shape of the approximated Pareto front, NSGA-III [24] is still employed in the experiment Chapter 5 for reasons outlined in Section 6.1 of the evaluation.

2.2.3. Multi-objective Evolutionary Algorithm Based on Decomposition (MOEA/D)

Another approach simplifies the multi-objective problem into a scalarized single-objective one [84]. This approach looses information by only returning one solution and leaving weight choice for a linear scalarization to the user. Multiple different scalarizations can be optimized at the same time to alleviate this. One algorithm using this approach is MOEA/D by Zhang and Li (2007) [84]. Among other advantages they claim lower computational complexity than NSGA-II and diverse results even for small populations.

More precisely, MOEA/D is actually an optimization framework that works with different kinds of scalarizations [84]. Aside from a weighted sum approach, Tschebyscheffs method [84] is commonly used. For this explanation we assume the weighted sum scalarization. Using it the total fitness F of an individual x becomes $F(x) = \sum_{i=1}^{n} \lambda_i * f_i$. Here n is the number of objectives and λ_i are the weights used.

MOEA/D achieves the titular decomposition into several single-objective problems via selecting several weight vectors $\lambda^1 ... \lambda^m$ [84]. Usually, those are uniformly distributed in what can be considered weight space. Each weight vector has a neighborhood of weight vectors which are closest to it using euclidean distance. Furthermore, MOEA/D keeps an archive EP[2] of non-dominated solutions.

Initially, MOEA/D generates a population of one individual for each weight vector [84]. During each generation of the algorithm these steps are performed for each weight vector λ^i and associated individual x^i:

- Select two individuals associated with random neighbors of λ^1 and create their offspring y^i using crossover and mutation.
- For each neighbor λ^j of λ^i check if y^i is better than x^j. If yes, replace x^j with y^i.
- Update EP by removing all solutions dominated by y^i and adding y^i if it is not dominated by any remaining solution in EP.

Despite the effectiveness of MOEA/D and its low computational complexity the algorithm as one major drawback: It is difficult to parallelize. For many

[2]Stands for external population, though elitist population would be a better description.

problems the evaluation of the fitness function is the most time-intensive part of run-time - one example being the epidemic simulation proposed in Chapter 3. Parallelizing the evaluation of multiple individuals is often a necessity to get results in a reasonable time-frame. Algorithms like NSGA-II can do this trivially by first generating a batch of new individuals and then evaluating them at the same time. MOEA/D only generates one individual at a time, making parallelization more difficult [54]. One approach is parallelizing the fitness evaluation internally, which we describe in the implementation Chapter 4. Other methods take advantage of splitting the population into several groups, thus creating a parallelizable layer on top of MOEA/D [50].

2.3. General Graph Theory

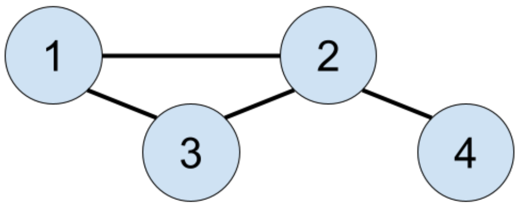


Figure 2.4.: Example of a graph $G = (V, E)$ with $V = \{1, 2, 3, 4\}$ and $E = \{\{1, 2\}, \{1, 3\}, \{2, 3\}, \{2, 4\}\}$.

Subsection 2.1.4 already hints that the next Chapters - design and implementation of the simulation model - rely on graphs. Some background on graph theory can be found in Wilson's "Introduction to Graph Theory" [79]. Mathematically speaking, a graph is a tuple $G = (V, E)$ with V being vertices and $E \subseteq V \times V$ the edges connecting them. For an example of a simple graph see Fig. 2.4.

Bipartite graphs [79] are a form of graph especially relevant for the model in Chapter 3. Instead of one set of vertices, this graph has two and takes the form $G = (V_1, V_2, E)$. In a bipartite graph every vertex from V_1 can only be connected to vertices from V_2 and vice versa. Thus, $E \subset V_1 \times V_2$. Matching problems are a classical application of bipartite graphs. In them e.g. prospective hires are assigned to open job positions in an optimal way. Another advantage of bipartite graphs is that they are more restricted than a general graph. We can use this structure to speed up solving graph problems and make implementation easier. See Chapter 4 for details.

There are several common approaches to implement graphs, notably adjacency matrices, adjacency lists (also called vertex lists) and edge lists [79]. We use the later in Chapter 4 to build the final graph, so they warrant a short explanation. Look again at the description of Fig. 2.4, because edge lists are close to the mathematical formulation: E can be implemented as an array of tuples or as two arrays (containing the start and end of each edge respectively). In theory, V then becomes redundant in some cases, since we can reconstruct the list of (connected) vertices from E. However, if data attributes are required on vertices (like in Chapter 4), the list of vertices cannot be discarded. For more details regarding graph frameworks used in social networks, see Camacho (2020) [18].

We can consider the view-point of breaking large graphs into manageable parts for either modeling or implementation. Multiplex Networks are an example of this. Chung et al. (2020) [20] offers an introduction and we already mentioned the concept in Subsection 2.1.2. Many more types of such networks exist. Ultimately, they only serve as a lens for viewing the model in the next Chapter. A more detailed overview and consideration would lead to far afield in this thesis. For an introduction to different types of multilayer networks see Kivela et al. 2014 [42].

2.3.1. Dynamic Graphs

A contact graph like in Subsection 2.1.2 can have another relevant property: People do not meet the same individuals every day, so a contact graph changes over time. Dynamic graph theory (see Kochkarov et al. 2015 [43]) deals with such graphs.

It defines a dynamic graph Γ as a sequence of classical graphs without loops or parallel edges [43]. A $G_{t+1} \in \Gamma$ is obtained from G_t by applying graph operations like edge or node addition or removal. While a contact graph can in theory be such a dynamic graph, any real-world example will contain loops as soon as three people meet up at the same time. On the other hand, we build the graph sequence introduced in Section 3.2 on a graph that can be considered a directed mapping of visitors to locations. Thus, dynamic graph theory offers a viable view-point to keep in mind when considering the epidemic graph model in Chapter 3.

2.3.2. Graph Generation

When working on graph problems (like a contact graph for epidemic spread) we need a lot of different graphs, often with specific properties [21]. Procedural generation helps with such simulation efforts, because it enables easy generation of more graphs fitting the necessary scheme [21]. When working on generating graphs, we need to keep in mind that the generation should be reproducible. Otherwise, the tooling used is inconvenient and more importantly experiment results irreproducible. Generally, there are two ways to approach the problem: Top-down, which leads to black-box generation algorithms or bottom-up using rules.

Examples for a top-down approach are graph generation using EAs by Bach et al. (2013) [6] or neural networks (NNs) by Bacciua et al. (2020) [5]. These methods main advantage is the ability to generate a lot of graphs fast and mimic graphs with which the model was trained. In turn, if there are no previous graphs to mimic, the algorithms can not be trained properly. Due to their black-box nature we may have to validate necessary graph properties afterwards as well. In absence of understandable generation rules reproducible graphs require fixed random seeds for the algorithms. See also Section 4.1.

On the other hand, we can use simple rules to create a graph bottom-up: In our university scenario there are n lectures with a specific time-slot. Each lecture has one lecturer and m to k students attending. This method still requires fixing the random seed for repeatability and cross-checking the final graph for strange rule interactions. At the same time we can understand the building blocks easily. For a more formalized approach see graph grammars [26].

Regardless of approach, care needs to be taken, since a bad or misunderstood generation method can lead to wrong data and thus invalid results [21].

3. Model Design

As mentioned in Chapter 1 our goal is a model adaptable to diverse small communities. While we choose the university scenario for reasons of familiarity, it does represent a community that was directly impacted by the recent SARS-CoV-2 outbreak [76, 32]. See Subsection 2.1.5 for examples. To keep the model simple, we only use the main OvGU campus as basis for the model Furthermore, we build the model to be self-contained and no contact with the outside model is represented - though the background mechanic in Section 3.3 can be said to include this.

We want to end up with a reusable model and not overspecialize. Thus, the trade-off between relying on empiric data and extrapolation from simple assumptions is especially relevant. On one hand, we can model the university scenario by taking the lecture list and attendance - anonymized for privacy - from the relevant system. Aside from avoiding the need to generate random graphs, this approach creates a model very close to the actual problem. Reliance on empiric data faces the drawback of a lot of time spent on gathering said data. In the other extreme, we can completely forgo empiric data. Such a model is without actual relevance, since it does not model the university scenario. A bottom-up generation method built on simple rules as mentioned in Subsection 2.3.2 is a more sensible approach on this side of the spectrum. Rule-based generation allows keeping the structures typical for the chosen scenario while reducing the need for gathering empiric data.

We choose such a graph generation method for the final model. An actual generation algorithm is part of the implementation Chapter and can be found in Section 4.1. The current Chapter focuses on the theoretical foundations of the model instead.

3.1. The Graph Model

The trade-offs of different simulation approaches are discussed in Subsection 2.1.4. As a result of these deliberations we choose a bipartite graph model. In principle, the idea is to model individuals and the locations (or logical groups) they visit at specific times.

For a first look at the model, we assume a fixed point in time. Every such time-step (details in Section 3.2) is represented via a bipartite graph of the form $G = (P, L, E_{visits})$. The first entry of the graph's tuple, P, is the set of people we model. In our university scenario these are students, lecturers (professors and their teams) and staff (esp. cafeteria workers). On the other hand, L denotes the locations every person can visit. We include lecture halls, cafeterias, shared student flats and sport courses in this set. Both of these sets can be extended and modified for different scenarios and L is not constrained to physical locations. One example for this is study groups with fixed participants, but no fixed location.

$E_{visits} \subset P \times L$ is the relation of which individual visits which location, e.g. students attending a course. The elements of E_{visits} form the edges of the bipartite graph. Since a person cannot be in two places at once, the graph could be constrained to only allow one location to be visited by one individual. However, the time-steps described in the next Section are not atomic, but represent a time interval. A student could leave a lecture early to grab a bite to eat, creating two connections. Due to this and for the sake of simplifying the graph generation in Section 4.1 we drop the constraint.

Different viewpoints on the graph are possible: The graph can be separated on location or person type lines to get sub-graphs. Those form a multiplex network as mentioned in Subsection 2.1.2 and Section 2.3. More importantly, we can transform the bipartite graph into a classical contact graph by fully connecting all individuals that visit a location at a given point in time. We show an example of this transformation in Fig. 3.1.

The graph G in this simple form does not contain all data needed for simulating an epidemic. First, we give every person an infection status like in a SIR model [40]. Mathematically speaking, we represent this via a function $state : P \to \{S, E, I, R, D\}$. The practical implementation in Section 4.1 represents *state* and similar functions as node and edge attributes. Recapitulating the states used in the SIR model as already mentioned in Subsection 2.1.1: S is healthy

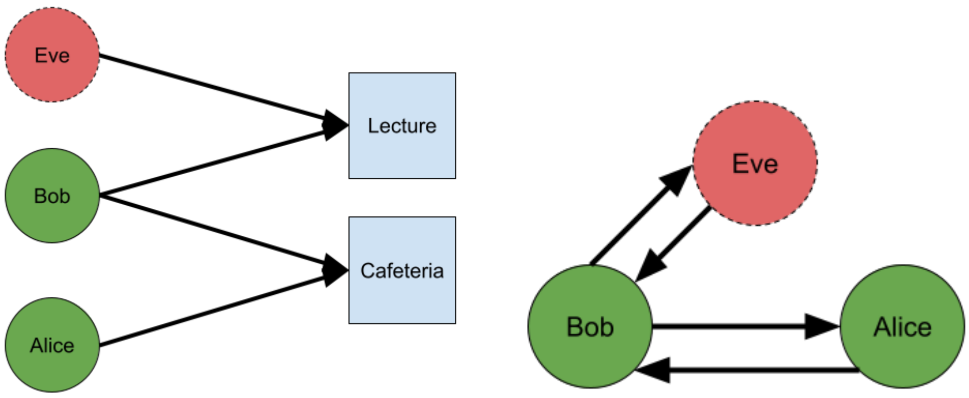


Figure 3.1.: A simple bipartite graph and its transformed contact graph counterpart. In this example Eve is infected, while Bob and Alice are susceptible.

and susceptible, E is exposed but not yet infectious, I is infected and infectious, R is recovered and D dead.

Further details about members of P and L can be represented via similar functions One example is the type of location, $type_L : L \rightarrow \{foodplace, lecture, flat, ...\}$ or infection rate for the location.

In order to find out how an epidemic spreads in our (still hypothetical) fixed time-step, we look at each $l \in L$. The set of infected visiting the location during the time-step are $I_l = \{p \in P | state(p) = I, (p, l) \in E_{visits}\}$. Using the number of ill visitors $|I_l|$ we can then calculate the infection chance for every $S_l = \{p \in P | state(p) = S, (p, l) \in E_{visits}\}$ depending on the location's infection rate. We do this by assuming that for each infected a susceptible person has one chance to infect themselves with probability $p_{type(l)}$ Iterating through all locations yields a list of newly infected.

A simplified view of the simulation steps during a time-step can be:

- Every exposed individual becomes infectious with rate $p_{E \rightarrow I} = \frac{1}{Number of incubation time-steps}$.
- An infection mechanic as described in the last paragraph.
- Every infected person may die with a chance of $p_{I \rightarrow D}$.
- Lastly, they may recover instead with rate $p_{I \rightarrow R}$

3.2. Modeling Time-steps

After modeling infection spread in one time-step, we simulate consecutive time-steps next. The overall spread of the epidemic can then be tracked by observing people's changing states. However, individuals visit different places during different time-steps. Thus, the graph G is more accurately described as $G_t = (P, L, E_{visits,t})$. If the next time-step is t', we use a different graph $G_{t'}$. Ideally, we want to generating a single graph and not new ones for every time-step. Because of this the final implementation in Section 4.1 uses a different representation. In it $E_{visits,T} \subset P \times L \times T$ is the relation of visits over time, with T the set of all possible time-steps. The infection mechanic then uses $E_{visits,t} = \{(p, l)|t' = t, (p, l, t') \in E_{visits,T}\}$

In the university scenario we find two natural subdivisions for time: Days of the week and lecture time-slots, called *blocks* from now on. In the case of OvGU these blocks are two hours long, which is the subdivision used for the model as well. The first block starts at 7:00 and the last one at 19:00. Due to that the final implementation (see Section 4.1) uses two edge attributes in visits instead of the one attribute implied in the last paragraph. However, the principle of only using edges matching the current time remains.

For reasons of simplicity, we set the maximal run-time of the simulation to one semester, which is 27 weeks. The visit relation is further assumed to not change between weeks, $E_{visits,t} = E_{visits,t+7}$. Alternatively, we terminate the simulation if the epidemic has run its course, instead of waiting for the end of the simulated semester. We consider infection spread over if there are no more individuals with an E or I state.

3.3. Representing Sporadic Infections via a Global Background Mechanic

Arguably, the infection model presented so far is incomplete and too simple. There are especially two concerns: What happens with spontaneous interactions between people outside the modeled locations (e.g. meeting on the lawn)? Further, how do we represent interactions with the outside world introducing new infections?

In order to keep the model simple, a global infection mechanic similar to the one used by Karaivanov (2020) [38] is introduced. We assume a small random chance for meetings between infected and susceptible people. This assumption leads to a fuzzier epidemic spread and reduces the chance of the infection running into dead ends in the graph. It covers both concerns from the last paragraph while keeping the model simple.

Each time-step this mechanic calculates the expected value of global meetings between infected and susceptible people. For this it uses a global meeting chance r_{global} as well as a infection rate for such meetings p_{global}. We then calculate the expected value of new infections due to background meetings via $E_{global}(I, S) = p_{global} * r_{global} * \frac{|I_t| * |S_t|}{|P|}$. This value is rounded and the newly infected sampled from the set of susceptible people.

3.4. Modeling of Epidemic Containment Policies

Subsection 2.1.5 talks about containment policies for epidemic spread in the real world. A clearer way of modeling such policies and their costs is needed to model such containment adequately and be able to draw conclusions from the model. For the remaining discussion we need to give a more precise definition of policies and their types. We use these definitions in the rest of this thesis:

- Let a *quarantine policy* be any algorithm or rule set that blocks visits between people and places (e.g. no food for students).
- Such policies can be further differentiated into *location-based and people-based quarantine policies* with respective definitions.
- Another important kind of policy for containing epidemics is a *testing policy*, which decides if an individual should be tested or not.
- Last but not least, a kind of policy that is rather controversial during the current SARS-CoV-2 outbreak [61, 65]: *Vaccination policies.* It is similar to a testing policy, but takes people out of the epidemic model for good similar to recovery.

Note that a model using all of these policies ends up with three additional states for people, inspired by Ivorra et al. (2020)[34]. T would mark tested but healthy individuals, while H would be untested, but infected people . Lastly, V would stand for vaccinated individuals.

However, the model so far (including policy definition) is still an extreme simplification of real world processes. The SARS-CoV-2 virus for example could possibly be recontracted even after having recovered from it or being vaccinated - after enough time has passed [11, 67]. Neither does it consider suboptimal vaccination efficiency or people with prior health problems. While implementing a SIS model like mentioned in Subsection 2.1.1 is possible, our chosen time-frame for the simulation makes it far less relevant. We only simulate 27 weeks at most. The expected period of immunity for SARS-CoV-2 is longer than that [35]. Thus, we do not model reinfection.

Before talking about which policies to explore and optimize, we assume three fixed policies:

- If a person shows sufficient symptoms (which is assigned with a certain chance on getting infected, see Table A.2), they self-quarantine and stay at home.
- If a person shows symptoms, we assume they are tested positive. Thus, the people living in the same flat are quarantined as well.
- If a lecturer shows symptoms, the lecture is canceled.

The model hard-codes these policies because they are ubiquitous whenever policies are made in the real world [75].

A simulation of epidemic spread only serves as a starting point to actually find an optimal way to contain it. In order to do that, we need to a look at which policies make sense to optimize. For this we take a view away from university for a second. Think about a small city with retail shops, barbers and other small businesses. A policy-maker for such a community faces several main questions, roughly separated by policy type:

- Which rules of thumb doe we give people to keep contact and spread risk minimal? This question leads to person-based policies.
- Which businesses are closed (and when)? Which get strict hygiene rules? These questions lead to location-based policies.
- Which people do we test? If someone is positive, who else should be tested around them? (Testing policies)

- Once we have medical personnel vaccinated, how do we distribute the limited doses available? Do we force people to be vaccinated? (Vaccination policy)

While all of these are important questions, most go beyond the bounds of our initial model. In turn they do not help with understanding the model. Thus. we ignore person-based quarantine policies, since they do not interact as nicely with our graph due to lacking person-person interaction. We further assume people in general to be sensible and keep to adequate hygiene standards without policy optimization offering significant gains. Testing and vaccination are not modeled at this point either to keep the model simple. While implementing them can be done via adding more attributes to people, the additional complexity breaks the scope of this thesis - even if the resulting data would be interesting.

After discarding several types of policies, we remain with one ideal starting point for policy optimization in our model: A location-based quarantine policy that decides if a location can be open and how many people may go there during each time-step. Limiting locations by number of visitors allows for the advantage of using hard numbers. Using visitor density (people per square meter) yields no gain over the plain visitor limit. Both methods can be transformed into each other, but the density approach needs additional attributes on locations. In practice, the chosen method leads to three policies:

- Closing lectures with more than $k_{lecture}$ attendees.
- Letting at most k_{food} visitors into each cafeteria per time-slot. If $n > k_{food}$ people want to eat, sample k_{food} from the n visitors to simulate random arrival times.
- Closing sports courses above k_{sport} attendees.

We optimize these parameters via EMO as described in Section 2.2, with the model itself serving as a black-box fitness function. These three policies look simple on first glance. Despite that, finding optimal parameters for them is a relevant problem, because similar policies are common in the real world [75, 57]. Thus, the results of a sufficiently detailed model can be used as basis for decision making. The Pareto front of the resulting problem can give further insights into its overall shape and other avenues of research. Last but not least, a simple optimization problem allows for tweaking model parameters and learning more about the model itself more easily.

3.4.1. Optimization Goals

We already discuss the three main goals of epidemic containment in Subsection 2.1.5. With a model on hand we can conceive several variations of these goals:

- We can view the number of cumulative infections either as the number of deaths or the combined number of recovered and dead individuals. Due to the model not considering prior health problems or higher susceptibility to a lethal infection, both are equal. To be more precise: For an infinite population the rate of recovered to dead people is fixed by their respective unchanging probabilities.

- The infection peak is defined as the highest number of infections in any one time-step. Similar to the number of cumulative infections we can consider either the number of infected or the number of infected and exposed people.

- The time-step of the infection peak, which is a distinct possible fitness from the size of the peak.

- The cumulative cost of all policies enforced, together with the cost of self-quarantine and similar measures.

Ideally, we choose no more than three criteria to guide optimization. The reason for this is EMO algorithms being optimized for these numbers of criteria [44, 24]. Visualization of the Pareto front beyond three dimensions is more difficult as well. Furthermore, too many criteria can obscure insights gained into the model.

We discard peak time as a goal due to it being less important than peak size for the scenario considered - the health care system is simply assumed to be as prepared as possible. The three other objectives are left. For both peak size and cumulative infections we choose the second variant. While peak size and cumulative infections do not necessarily depend on each other directly, they might not conflict. Because of this we cannot be sure adding both objectives is beneficial without running tests. In the worst case both objectives can together marginalize the cost objective. As a result three combinations will be evaluated starting in Chapter 5: We pair the cost objective with either peak size, cumulative infections or both.

Equations 3.1 to 3.3 contain a more precise formulation of these objectives. The first one defines the cumulative objective $f_{cumulative}$ via the population state in the last time-step.

$$f_{cumulative} = |\{p | p \in P, state_{t_{last}}(p) \in \{E, I, R, D\}\}| \tag{3.1}$$

Our informal description of this objective in the beginning of the current section only considers recovered and dead people. On the other hand, Equation 3.1 counts exposed and infected as well. The reason for this is the simulation of the university scenario being cut off after one semester (see Section 3.2) without guarantee that the epidemic has run its course. Equation 3.2 defines f_{peak} in a similar way, but takes the maximum of infected and exposed people over all time-steps.

$$f_{peak} = \max_{t} |\{p | p \in P, state_t(p) \in \{E, I\}\}| \tag{3.2}$$

Lastly, we describe an idealized f_{cost} in Equation 3.3. Breaking it apart, φ denotes a policy (or multiple) as described in Section 3.4.2 and $\varphi(E_{visits,t})$ is the set of allowed visits in time-step t. Then $E_{visits,t} \setminus \varphi(E_{visits,t})$ is the set of disallowed visits under the current policies in the current time-step. We average the cost for each disallowed visit (as described in Section 3.4.3) over all time-steps, with T being the total number of time-steps. The actual, more complex version used in the implementation can be found in Subsection 3.4.3, Equation 3.4.

$$f_{cost} = \frac{\sum_{t \in T} \sum_{e \in E_{visits,t} \setminus \varphi(E_{visits,t})} cost(e)}{T} \tag{3.3}$$

3.4.2. A Policy Model for the Graph Representation

Integration of the policies discussed in this Section so far happens by considering them as filters for E_{visits}, the visit relation of the graph. We consider $\varphi(E_{visits}) = E'_{visits} \subseteq E_{visits}$ a policy, which returns an "active" sub-set of the current visits. Any relation elements thrown out by policies are no longer considered for the time-step. This approach is still a simplification. The policy

φ can use all attributes and other information of the graph (infection states, location types, ...) - some of which are hidden in real-world scenarios. We gain the final sub-set of active visits for each time-step by applying policies after another. Filtering visits for the correct day of the week and lecture time-slot works like this as well. However, these two special policies do not come with an associated cost unlike the normal policies described in the next Subsection.

There is another way of looking at applying multiple policies at the same time: The resulting sub-set is the intersection of visits allowed by each policy. Mathematically, this means $\varphi_1(\varphi_2(E_{visits})) = \varphi_1(E_{visits}) \cap \varphi_2(E_{visits}) = \varphi_2(\varphi_1(E_{visits}))$. Since set intersection is commutative, policy application is as well. Cumulative policies have the advantage of enabling us to reordering them for better performance in Chapter 4.

3.4.3. A Cost Model for Policies

After modeling location-based policies, we define their costs next. In Subsection 2.1.5 we noted that these costs come in many variations [57, 64, 76]. However, optimization of the objectives named in Subsection 3.4.1 needs quantified costs.

The easiest way to do this is assigning each edge in the graph (corresponding to a specific visits to a location) an associated cost. We model edges having different levels of importance by different assigned costs. Mathematically, let $cost_{E_{visits}} : E_{visits} \to \mathbb{R}$ denote the importance of an edge. The total cost in one time-step (see Equation 3.3) is the sum of costs over all visits that are cut off by policies.

However, we only sum costs for the three policies associated with the optimized parameters ($k_{lecture}$, k_{food}, k_{sport}). Initial tests showed that adding the costs of edges cut by static policies (self-quarantine, lecture cancellation, ...) dominates the final cost by a wide margin. Due to this optimization is skewed and ignores the actually interesting cost-factors. We work around this through special handling of edges removed by static policies.

In the final model we disregard self-quarantine cost-wise. It is observed through its ripple effects instead. The resulting quarantine of flat mates and lectures are counted and multiplied by a constant factor ($cost_{lecture-quarantine} = |lectures\_canceled| * c_{lecture-quarantine}$). For the sake of simplicity and to better see the effects of active policies, we set these constants to $c_{flat-quarantine} =$

$c_{lecture-quarantine} = 0$ for the experiments described in Chapter 5. The final modified cost objective is then given by Equation 3.4 with $cost_{static-policies} = cost_{lecture-quarantine} + cost_{flat-quarantine}$.

$$f_{cost} = \frac{\sum_{t \in T}((\sum_{e \in E_{visits,t} \setminus \varphi(E_{visits,t})} cost(e)) + cost_{static-policies})}{T} \tag{3.4}$$

Another problem with the cost model is that cost values skyrocket if summed over all time-steps. For this reason we choose the cost objective (Equations 3.4) as the average cost over simulated time-steps. This approach has the advantage of keeping cost values easier to interpret. However, we loose some information as well: Longer epidemics with high cost can be equal to shorter periods with lower costs. Initial test runs of the simulation show the epidemic infecting the entire population before cut-off time - except for a total lockdown scenario. Because of this we consider information on the total length of the epidemic less important for the formulated optimization objective.

3.5. Summary of the Complete Model

Finally, we need to integrate the modeling decisions from this Chapter into a single algorithm. To summarize: We use a bipartite graph of people visiting locations. The modeled simulation time is - in line with the university example - divided into weeks, days and blocks for a total of one semester. Infection happens via infected and susceptible people visiting the same locations with an infection rate depending on the location. At the same time, we abstract away random meetings and outside interaction via a global background infection mechanic. Last but not least, there is a cost model for containment policies along with several "common sense" quarantine measures. A summarized overview of the simulation procedure can be found in algorithm 2.

3.5.1. Parameters

Before we continue with the actual implementation in Chapter 4, we should discuss the chosen model parameters that are not implementation specific. An overview over the parameters used for the underlying graph is found in Table

Algorithm 2 Summary of the epidemic simulation model

```
Initialize graph
Initial infections
for Each week of the semester do
    for Each day of the week do
        Exposed → Infected
        $cost_{day} = 0$
        for Each lecture time-slot of the day do
            Find relevant visits for day and block under policies and quarantine
            Add their cost to the total: $cost_{day} += cost_{policies}$.
            Group visits by location and find the number of infected at each
            Roll infection chance for susceptible depending on infected at location
        end for
        Run global background infection
        Infected → Dead
        Infected → Recovered
        Gather statistics for current day
        if Termination condition is met then
            Write statistics to disk
            Stop execution
        end if
    end for
end for
```

A.1 along with reasoning for the choices. The parameters for the epidemic simulation are in Table A.2. Both can be found in Appendix A.

Some highlights include: There are 10000 people simulated of which 7500 are students, 1500 lecturers and the rest staff. We simulate slightly over 2300 lectures, 75 sport courses, 4 cafeterias and as many flats as needed - all with respective minimal and maximal sizes. Furthermore, we define possible days and times for eating at the cafeteria and sport courses.

On the epidemic side, base infection rate is set to 50% [38], with adapted higher rates for sport and lower ones for cafeteria visits. There are 25 initial infections. For exposed people the infection breaks out after roughly 5 days [38], with 50% developing symptoms [74]. The total rate of removal for infected (which is recovery and death rate together) is 20% [38].

4. Implementation

The implementation of the model[1] described in the previous Chapter is split into four parts: In a first Section we describe the data structures representing the graph as well as its generation. Next are implementation details of the algorithm shown in Section 3.5, including performance optimizations. After that, Section 4.3 describes the learning framework used for optimizing the policies. Lastly, we discuss run-time data gathered from the simulation and other utilities.

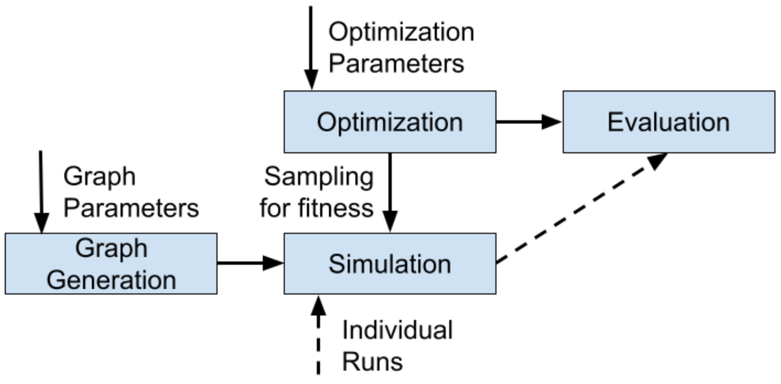


Figure 4.1.: Shows the workflow for either an optimization run with internal usage of the simulation (solid arrows) or an individual simulation run (dashed arrows). In the rest of this thesis *iterations* are multiple runs of the same optimization with the same starting parameters. *Sampling* on the other hand refers to an individual's fitness running the simulation multiple times for more accurate results.

Each of those Sections can be found in Fig. 4.1 as well. It gives an overview over the general process of running an optimization or individual simulations.

[1]Sorce code can be found at https://gitlab.com/covid-simulation/simulation-py.

Note that the parts of the implementation build on each other. The simulation needs graph generation. In turn, the optimization algorithm samples the simulation to obtain fitness values for individuals. Evaluation is based on the data gathered according to Section 4.4.

4.1. Graph Representation and Generation

The graph model described in Chapter 3 first needs to be converted into a form usable by computers. There are several possible ways to do so mentioned in Section 2.3 on graph theory. We choose edge lists for our use-case. Reasons for hand-rolling such an implementation instead of relying on existing graph software are two-fold: First, representing a bipartite graph via edge lists is straightforward and simple. General purpose frameworks optimize for many different types of graphs, making them more complex and less optimized for our specific use case. On the other hand, an implementation with basic lists at its heart offers transparency for understanding and modifying the model, as well as optimizing performance.

For the same reasons we choose Python as implementation language. It offers rapid prototyping speed with good performance through use of appropriate libraries. We use Python 3.6.8[2] due to the compute cluster used for experiments running this Python version.

Furthermore, Pandas [60] is the main implementation library. People, locations and visits are each modeled as one dataframe (the Pandas equivalent of a database table). Pandas's underlying implementation offers good performance. The library can also be used for analyzing the resulting data without switching to something else.

4.1.1. Generation Algorithm

While our model in theory supports graphs of arbitrary complexity, we first need to verify the validity of the chosen approach. For this we need a simpler initial graph. Thus, we choose a bottom-up graph generation approach with simple rules. See Section 2.3.2 for a recap of pros and cons this approach to graph generation offers.

[2]https://documentation.help/Python-3.6.8/index4.html

First, we generate the list of people. In order to avoid naming individuals as "person-1", we use a list of names[3]. Generation shuffles the list and then samples the needed amount of names from it. The 10000 initial names are split by type according to the numbers in Table A.1. We insert each person into the people dataframe with name, type, a state of S and the boolean flag for having symptoms set to false.

Locations work slightly different (except for cafeterias, which are read from a short name file as well). They are generated together with the relevant visits by applying a generation rule. For example lectures: For each lecture number, we choose a lecturer and time (day and slot). Their visit is added to the visits list with

- the lecturer's name,
- the location's name (e.g. lecture-1),
- day and time-slot,
- cost of removing the edge and
- inverse infection rate ($1 - infection\_rate$).

We prefer inverse infection rates for performance reasons. See Section 4.2 for details. In the same vein, we pick a random number of lecture attendees from the students. As a last step, the lecture location is inserted into its table, with name, type and degree (the number of people visiting it). This degree helps with enacting policies later.

Overall, visits are generated according to these rules:

- Lectures as described in the last paragraph.
- We generate sport courses the same way, but they have no fixed "leader".
- For each cafeteria every student and lecturer picks a random time during "eating hours" to eat there. Collisions with other events are not checked. See Section 3.1 for reasons.
- Staff works in a cafeteria during eating hours.

[3]The file for them taken from https://www.usna.edu/Users/cs/roche/courses/s15si335/proj1/files.php%3Ff=names.txt.html on the 19th of February 2021.

Table 4.1.: Graphs used for experiments in this thesis. Note that during prototyping we used different random seeds as well.

Graph Seed	Graph Scaling	Cap Scaling
0	0.15	0.5
0	0.5	0.75
0	1	1

- Students share a flat in the last time-slot (after hours). We sample the number per flat from a predefined range similar to lectures and sport courses.

4.1.2. Graph Scaling and Parameters

The full graph (10,000 people, 4500 locations, 240,000 visits) is quite large for rapid prototyping and model exploration. For this reason we introduce two scaling parameters used to modify graph generation without having to define even more parameter sets. Firstly, $graph\_scaling \in (0, 1]$ proportionally scales down the number of people of all types, lectures, sport courses and cafeterias (with a minimum of 1 of each remaining). Next, $cap\_scaling \in (0, 1]$ decides the maximal number of attendees for lectures and sport courses.

Thus, three values uniquely specify each graph generated as described in this Section: The two scaling parameters and a fixed random seed used to initialize the random number generator before producing the graph. During experiments in Chapter 5 we employ the graphs in Table 4.1.

4.2. Implementation of the Epidemic Simulation

This Section is an extension of the algorithm summary in Section 3.5. The actual implementation of the epidemic simulation is very close to this description. Thus, we only highlight some implementation specific details here.

As mentioned in the previous Section, we use three Pandas dataframes for people, locations and visits. Since their use is very similarly to relational database tables, we can think of the implementation in terms of database operations: For a transition of exposed people to infected we "select" all people with a state of E. A random number decides for each person if they transition or not. In the same vein, infected people visiting each location are found with a "join" and policies reduce visits by returning a "view" with less permissive "where" clause.

We further built several wrappers around the central *simulate* function. Several simulations can be run in parallel for in-depth evaluation of containment parameter sets as in Sections 5.3 and 6.3. Optimization instead uses a wrapper as fitness function which samples the simulation multiple times and returns the (independent) medians of each objective. The necessity of this approach appears in the end of Subsection 2.1.3 and we discuss it in Chapters 5 and 6 as well.

4.2.1. Performance Optimizations for the Implementation

Parallel execution is the first go-to for a fast simulation. Beyond that we need to avoid other trivial slowdowns. Thus, we implement several performance optimizations after profiling the simulation's execution.

We already introduced use of $1-infection\_rate$ in Subsection 4.1.1. Its mathematical background considers the total infection chance for a susceptible individual visiting a location: For one infected, this infection rate is the location specific $infection\_rate$. With more infected individuals we can more easily calculate the chance for a susceptible to avoid infection: $(1-infection\_rate)^n$, with n being the number of infected at the location. Thus, the final infection rate is $total\_infection\_rate = 1 - (1 - infection\_rate)^n = 1 - inverse\_infection\_rate^n$.

This alternative formula for infection further takes better advantage of vectorized Pandas operations. In a similar vein, proper library usage (correct indices to speed up joins) and advantageous ordering of policy and quarantine application speed up execution. Further enhancements include caching of unchanging values (e.g. which lectures are banned by policy) and - relevant for the next Section - caching of fitness values across threads.

4.3. Parameter Optimization with jMetalPy

Several python libraries offer implementations of EMO algorithms with custom problem definitions and genetic operators. Examples include DEAP [59], pyMOO [14] or jMetalPy [12]. All contenders fulfill most of our criteria, including parallel implementations of the respective algorithms - NSGA-II [23], NSGA-III [24, 36] and MOEA/D [84]. Thus, we choose jMetalPy due to a preference for its interface.

This Section covers the problem definition in jMetalPy, crossover and mutation operators, parameters for the EMO algorithms and implementation of a simple experiment wrapper.

4.3.1. Problem Definition

Our basic problem definition (*PolicyParameterProblem*) inherits from jMetalPy's *IntegerProblem*. It specifies the number of variables to optimize with an upper and lower bound for each. Lectures and sport courses are shut down completely, if the lowest value is chosen. The highest value allows all visits instead. Cafeterias have a lower bound of 0. We make an educated guess for the upper bound of 500 based on the average number of visits. EMO algorithms optimize away too high values for cafeteria attendance at the cost of a larger total search space.

Depending on the experiment (see Subsection 4.3.4) we pass to the problem, either two or three (float) objectives are used. Valid objective combinations are explained in Subsection 3.4.1. The fitness function for optimization (Section 4.2) depends on the samples desired and EMO algorithm used. Especially, if we choose MOEA/D, fitness evaluation runs in parallel, since the algorithm itself is sequential.

4.3.2. Simple Crossover and Mutation Operators

Our implementation prefers simple crossover and mutation operators.

As mutation operator we choose an integer neighborhood mutation. It picks one of the three variables of the solution (called v for now). With a chance of $p = 0.3$ a new value for v is picked from the allowed range. In the other case, we add a random integer offset to the variable by doing $v = v + n; n \in \{-5, ..., -1, 1, ..., 5\}$.

Crossover uses a single point crossover implementation: Each child inherits the first $k \in \{1, 2\}$ of its variables from one parent. The remaining $3 - k$ variables come from the other. Due to the small number of variables, any other value for k will lead to both children being identical to their parents. Furthermore, the MOEA/D implementation offered by jMetalPy uses three parent crossover. We handle this via applying our single point crossover pairwise, leading to six total offspring.

Both operators may seem overly simple on first glance. Since the graph model proposed in this thesis is simple, this is intentional. Our model offers a maximal search space consisting of $95 * 35 * 500 = 1\,662\,500$ possible individuals. Compared to other - often continuous [29] - EMO problems this is comparatively small [44]. Thus, our chosen operators present a trade-off between implementation speed and retaining effectiveness.

4.3.3. Algorithm Settings

Many of the settings used for the three EMO algorithms are set via the experiment settings detailed in the next Subsection. For all other values we prefer sensible defaults. We consider any optimization of the algorithm's hyperparameters out of scope for this thesis.

While NSGA-II [23] needs no further parameters, NSGA-III [24] requires reference directions. The jMetalPy framework offers an implementation for uniform reference directions in all objective dimensions. Both two and three objective runs use 12 total reference directions.

Similarly, MOEA/D [84] needs (evenly distributed) weight vectors. For two objectives jMetalPy has a built in implementation. Three objectives require manual generation of a weight vector file in a format the framework can read. Importantly, this generation constrains the population size for MOEA/D. Our implementation expects a population size with an integer square root for easier weight distribution in three objective problems.

We use the implementation of Tschebyscheff scalarization [84] provided by jMetalPy as an aggregation function for MOEA/D. The neighborhood size is 20, with a chance of 0.9 for picking a mating partner from the neighborhood.

Conversely, the chance of picking the second crossover individual from the whole population is 0.1. Lastly, a new solution may at most replace two individuals.

4.3.4. Experiment Definition

Wrapping up the implementation of the optimization algorithms, experiment setup deserves a mention. Similar to graph specification via a random seed and the two scaling parameters (see Subsection 4.1.2), we define a datatype to fully describe an optimization experiment. It consists of these variables:

- Size of the population for the EMO algorithms.
- Number of generations to run the optimization. Note that jMetalPy algorithms take number of evaluations as stopping criterion. We get these as the product of number of generations times population size.
- How many samples to take for deciding an individual's fitness. See Section 4.2 for details and Section 6.2 for an evaluation.
- A string deciding which algorithm to use.
- Which set of objectives to use.
- The specification for the graph.
- How many times to run the whole experiment. This parameter becomes relevant in Chapter 5 on experiment planning.

We build a wrapper to run specific experiments via an experiment list as well. Every experiment stage in the next Chapter is defined by such a list. The script executes one of the experiments if supplied with list name and experiment number[4]. A similar setup exists for many simulations of a single individual. Section 6.3 uses it to better understand the model by looking at specific individuals.

4.4. Tracking Experiment Results

Experiments need to gather data about their execution and write it to disk for later evaluation. Optimization experiments do this by logging individuals

[4]Along with execution details like maximal number of CPU cores to use.

and their fitness for every generation of the algorithm to a sub-folder. The final population, its fitness values and a plot for easier sighting of the results end up in the experiment's main folder. This experiment folder needs to be unique for every experiment to allow running several of them in parallel. We facilitate this by using the experiment specification as a basis for generating the path. As added benefit important experiment parameters become obvious on the file system's folder structure without having to consult the experiment list.

The simulation itself gathers run-time data to better understand epidemic spread for the model as well. "Basic" logging includes the number of people in every state (I, S, ...), as well as cumulative infections up to the time-step. These values determine the fitness of a set of policy parameters. If the simulation runs to gather data about specific parameter sets (e.g. knee points from the approximated Pareto front), we can set a flag to write these values to disk for later analysis.

In the same vein, we implement advanced statistics. With them we log the names of all infected and susceptible people for every lecture time-slot and location. While this amounts to a lot of data, identifying locations critical to epidemic spread becomes possible by analyzing it.

Aggregation scripts for the acquired data, plotting helpers and statistics are implemented along with the simulation. Details can be found in Chapter 6 on evaluation of obtained results. Notably, we use pymoo's [14] hypervolume calculation method for evaluation, since jMetalPy does not offer an easy to use alternative.

5. Experiment Planning and Results

Experiments in this thesis have the main goal of a better understanding of characteristics and viability of the defined model. Thus, we use the EMO algorithms as described in the last Chapter with different graphs and settings. The resulting Pareto front approximations are then analyzed. In a last step, "interesting" single individuals are picked from these fronts and their simulation data looked at in more depth. Interesting means in this case individuals at extremes, gaps, knee points or with other characteristics we consider worth looking at.

For a final list of experiments, we need to consider combinations of these points:

- Graph sizes. Here we use the three graphs mentioned in Table 4.1 for experiments.
- All three possible objective combinations (f_{cost} with either $f_{cumulative}$, f_{peak} or both) from Subsection 3.4.1 need an evaluation.
- We compare the three algorithms introduced in Section 2.2. This comparison is relevant, because we need to choose one of the algorithms for further analysis.
- How many samples are needed for fitness evaluations to remove "lucky" individuals from the results? A lucky individual has all its fitness samples at the lower end of the real distribution. Thus, optimization thinks it is much fitter than it acutally is. For evaluating this effect, strategies with 1, 7 and 23 samples are considered.
- Graphs with different random seeds. However, in the scope of this thesis we consider only one graph seed (see Table 4.1).

Even with only one graph seed, these points lead to a total of 81 combinations. To reduce this number, we split experiments into three phases which are explained in detail in the next Subsections. The idea is to first make an algorithm choice on a reduced graph. Both phases after that then deal with the main experiments using the chosen algorithm and individual evaluation.

5.1. Stage 1 - Algorithm Selection

For an informed choice between NSGA-II, NSGA-III and MOEA/D, this experiment phase focuses on the medium graph (Table 4.1) with 5000 people. We choose this graph for being the middle trade-off between run-time of experiments (due to graph size) and representing our university scenario better. For the medium graph we run all combinations of algorithms and objectives with 1-sampled and 23-sampled fitness. This leads to 18 experiments.

Note that the low sample rate proved subpar during development, with "lucky" (as defined in the beginning of the Chapter) individuals dominating the front. See Section 6.1 for details. For this reason we run the optimization only once for the 1-sampling in order to get comparison data and explore if one of the algorithms can mitigate the problem. Experiments for the high sample rate run 3 times. However, we canceled the MOEA/D experiment after one iteration on account of excessive run-time. Multiple iterations are aggregated and we consider the randomness of the resulting fronts in the evaluation, Section 6.1.

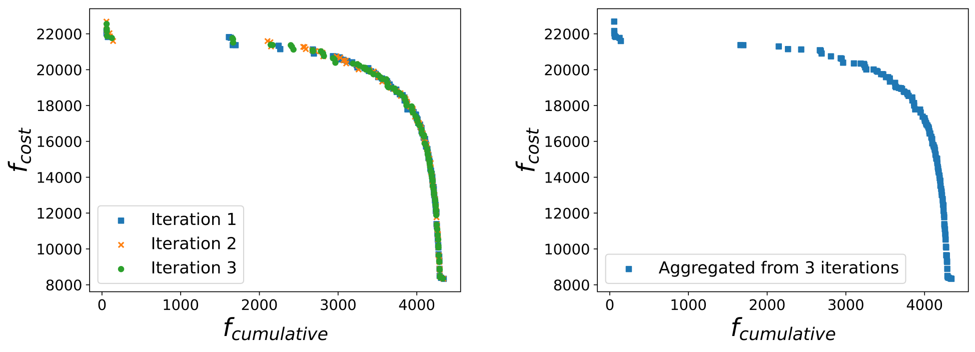
5.1.1. Results for Stage 1

After running the entire experiment list for the first stage we sight the results. The chosen algorithms take a vastly different time to complete. Thus, an overview over their averaged run-times can be found in Table 5.1.

Table 5.1.: The rounded run-time values for each EMO algorithm and sampling, averaged over objectives. Note that the three objective runs are typically slower that their counterparts.

Algorithm	1-sampling Time	23-sampling Time
NSGA-II	0d 2:04h ± 0:30h	2d 21:58h ± 17:22h
NSGA-III	0d 3:44h ± 0:07h	2d 7:06h ± 11:11h
MOEA/D	10d 16:18h ± 9:30h	**17d 18:39h** ± 2d 19:21h

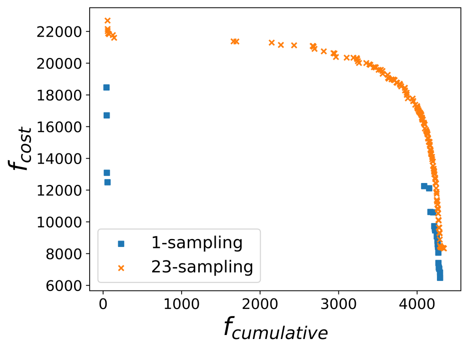
In the case of experiments running multiple times, we aggregate iterations into a combined front. We obtain these aggregated fronts by taking all individuals of the chosen iterations together and only keeping non-dominated ones. An evaluation of the validity of this aggregation can be found in Section 6.1. Figure 5.1 offers an example aggregation for the NSGA-II algorithm with the $f_{cumulative}$ and f_{cost} objectives. Within the rest of this Chapter and the next graphs display aggregated runs (except where the aggregation method is discussed explicitly).



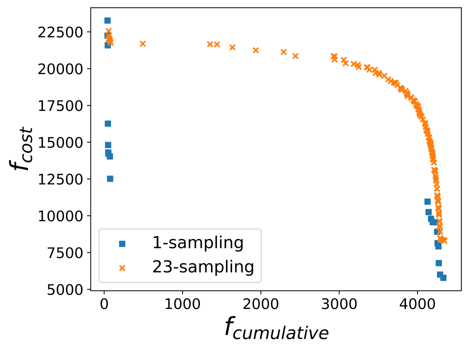
(a) Individual fronts (b) Aggregated front

Figure 5.1.: Comparison of individual fronts to their aggregated counterpart. The fronts on the left are the iterations for NSGA-II on the medium graph with 23 samples. They use the cumulative infection and cost objectives. The right plot shows the aggregated front obtained by taking all individuals and only keeping non-dominated ones.

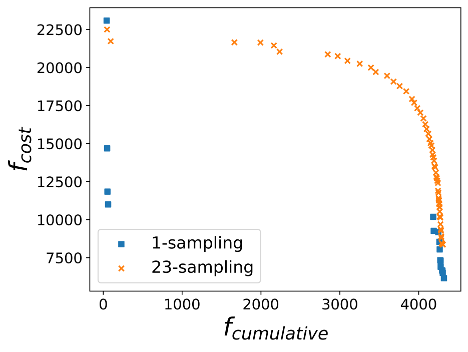
To provide a basis for the discussion on merits of the different EMO algorithms and the chosen objectives, see two more data sets: Figure 5.2 shows both sampling rate comparisons for the algorithms, as well as all algorithms side by side. In addition, Fig. 5.3 offers example results for the different possible combinations of objectives.



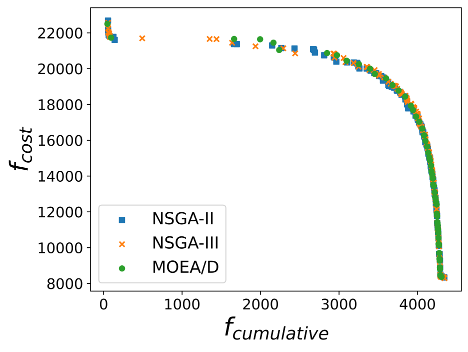
(a) Sampling comparison for NSGA-II



(b) Sampling comparison for NSGA-III

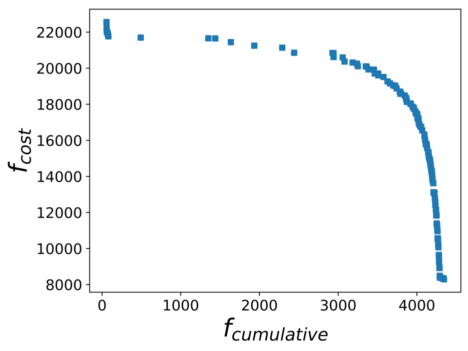


(c) Sampling comparison for MOEA/D

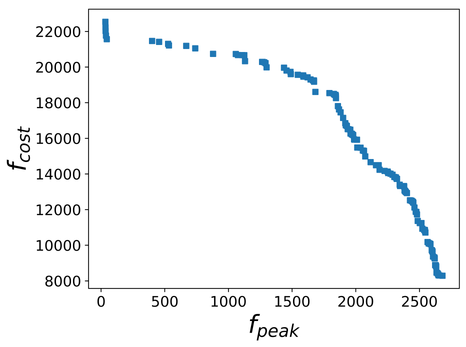


(d) Comparison of 23-sampling for all three algorithms

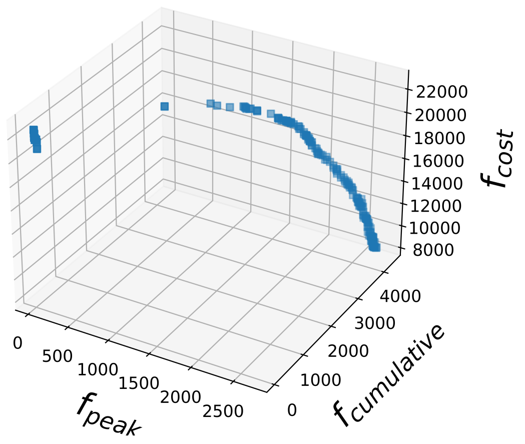
Figure 5.2.: Comparisons of NSGA-II, NSGA-III and MOEA/D algorithms. All subplots show the cumulative infection and cost objectives on the medium graph for low and high sampling rates. Note that due to its much longer run-time, MOEA/D only ran for one iteration.



(a) Cumulative infections and cost



(b) Infection peak and cost



(c) All three objectives

Figure 5.3.: Comparison between using cumulative infection, infection peak or both in combination with cost as objectives. The subplots show NSGA-III runs with high sampling on the medium graph.

5.2. Stage 2 - Optimization Experiments on Different Graph Sizes

First stage experiment results indicate NSGA-III being a good algorithm choice for further exploration. It offers both a good Pareto front and adequate run-time performance - with the caveat of having run too few iterations for statistically solid results. Thus, we choose it as main algorithm for this stage. For details see Section 6.1.

In the second stage we run the missing 7-sample fitness evaluation for the medium graph. More importantly, this stage contains the full set of objectives and sampling rates for both the small and large graphs from Table 4.1. In total the experiment plan consist of 21 experiments, with the NSGA-III results from the first stage completing the set.

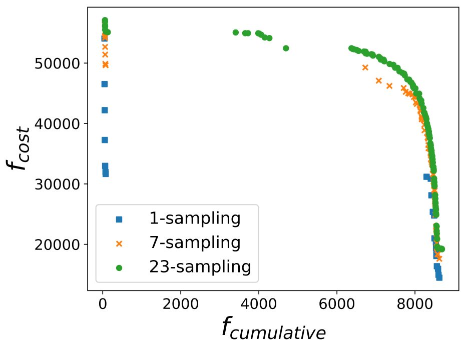
Our main goal for this stage is finding interesting individuals (as defined in the beginning of the Chapter) for the last experiment stage. During this stage we aim to understand the impact graph size and objective choice have on the optimization. Also, we investigate if the 7-sampling approach represents a valid alternative to the high sampling variant for cutting down algorithm run-time.

5.2.1. Results for Stage 2

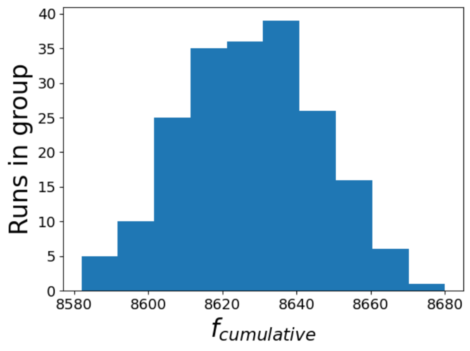
We want to compare all tested sampling strategies in Section 6.2. For this Fig. 5.4 displays an example plot with all three sampling variants for $f_{cumulative}$ and f_{cost} objectives on the large graph. It shows a histogram for $f_{cumulative}$ of one individual as a visual aid for consideration of sample rates. While this histogram is helpful for deciding on the number of samples, its distribution can vary depending on where an individual is in the search space.

Fig. 5.5 serves as comparison basis for the different graph sizes and objectives is Section 6.2. Fitness values for each individual are normalized in relation to the largest objective values for each graph size to make the data comparable. This approach leads to three plots, one for each objective combination, which contain the condensed results for all three graphs using the high sample rate.

Lastly, objective comparison and choosing individuals for the third stage profit from looking at the individuals in both search and objective space. We show



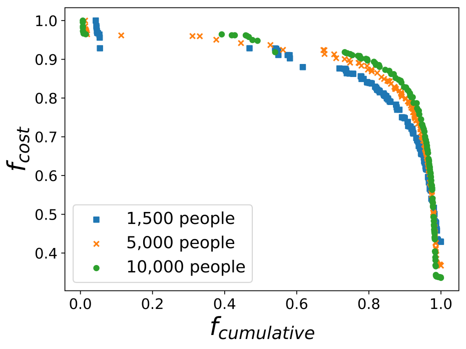
(a) Comparison of sampling strategies



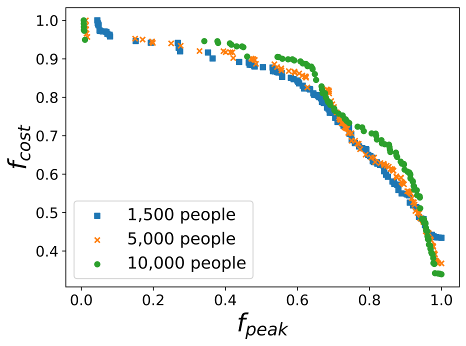
(b) Histogram for cumulative infections

Figure 5.4.: Compare the different sampling strategies on the left. This uses NSGA-III on the large graph and the cumulative infection and cost objectives. A histogram for 199 runs of the simulation for individual $(21, 474, 4)$ is shown on the right. It serves as a visual aid for considering how many samples need to be taken for dependable fitness values. Note that the median of the histogram data is at 8627.

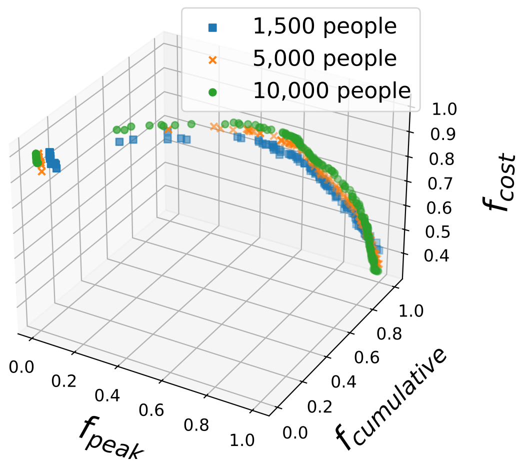
a side-by-side view for both the cumulative infection and infection peak objective, using the large graph, in Fig. 5.6.



(a) Cumulative infections and cost

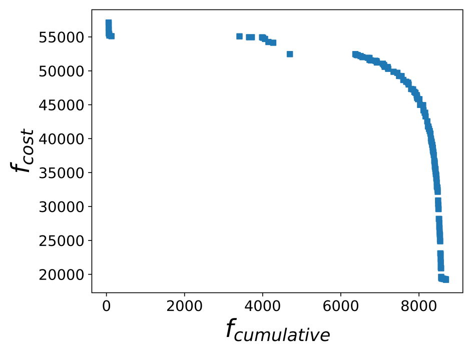


(b) Infection peak and cost

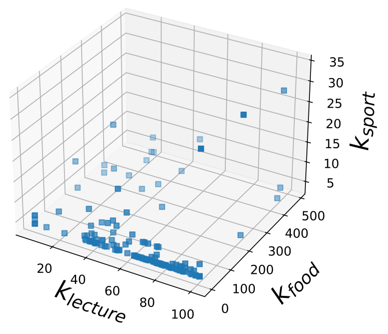


(c) All three objectives

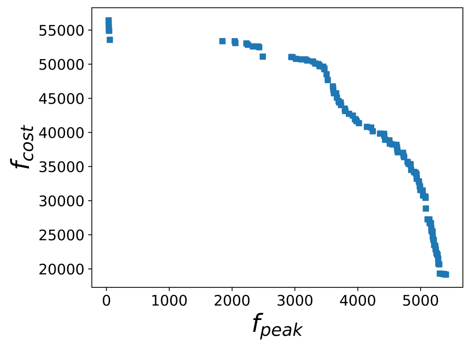
Figure 5.5.: Comparison between the three graph sizes used for NSGA-III experiments. Values are independently normalized by maximal objective values for each graph to make them easier to compare. Note that the right shift on the smaller graphs is likely due to the number of initial infections not scaling along with the graph.



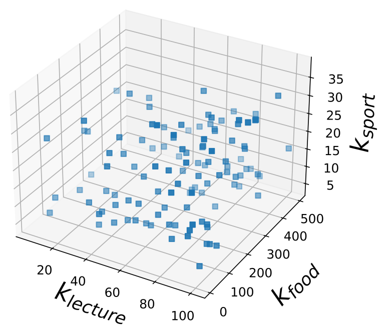
(a) Objective space for cumulative infection and cost objectives



(b) Search space for cumulative infection and cost objectives



(c) Objective space for infection peak and cost objectives



(d) Search space for infection peak and cost objectives

Figure 5.6.: Comparison between the search and objective space for both the cumulative infections and infection peak combined with the cost objective. The plots use the NSGA-III experiments on the large graph as basis.

5.3. Stage 3 - Infection Spread for Specific Policy Sets

This Section aims to answer the core questions arising from both model definition and previous experiment stages: Does the policy optimization return sensible parameter sets? Exhibit different parts of the Pareto front show distinct behavior for epidemic spread? Is the model able to represent epidemic spread in small communities accurately? Do results point toward deficiencies of the model for our university scenario? Can we improve the model incrementally to remove these deficiencies?

The simulation is run for the individuals chosen in Section 6.2 to answer these questions. A list of the final individuals and their policy parameters can be found in Table 5.2. For each individual we run the simulation 199 times[1] on the large graph. We enable advanced run-time logging as described in Section 4.4.

Table 5.2.: Individuals from the large graph experiments picked out for further analysis. For an explanation of the choices see Section 6.2. A description of the infection gap is in Section 6.1.

Individual	$k_{lecture}$	k_{food}	k_{sport}
No lockdown	101	500	40
Total lockdown	0	0	0
Least cumulative infections	5	1	5
Most cumulative infections	101	219	33
Left of gap	12	1	4
Right of gap	20	0	9
Only Lectures open	101	0	4
Low peak curve point	45	408	10
Mid peak curve point	61	284	14
High peak curve point	66	257	21
Highest peak	101	418	34
Close only lectures (Added after an initial sighting of data in Subsection 5.3.1)	0	500	40

[1]See also the probabilities in Table 6.3.

5.3.1. Results for Stage 3

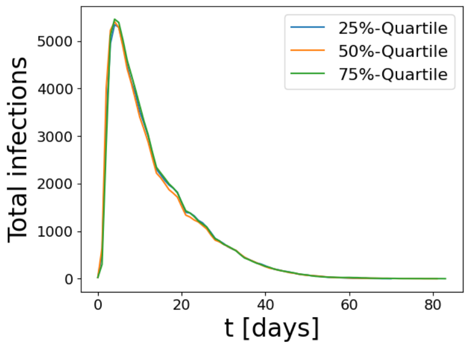
Running the simulation for all individuals in Table 5.2 results in a lot of data. This Subsection focuses on the data relevant to the evaluation in Section 6.3 by highlighting specific scenarios. Our first visualization of infection spread displays curves with the number of infected over time. These plots show quartile runs, picked using the height of the infection peak as measure. Fig. 5.7 shows a comparison of the upper extreme scenarios - no lockdown, the individual with most cumulative infections and the one with the highest infection peak. After that, Fig. 5.8 shows a similar comparison for the policy sets on each side of the infection gap[2].

We aggregate the data gathered on infected and susceptible at each location to better understand epidemic spread across locations. In general, we sum up the visitor amounts for all time-steps in one simulation run. Multiple runs are then averaged. For a first overview, we group different location types together. Fig. 5.9 shows infection curves and infected per location beside each other. It compares the scenario without any lockdown to one without lectures. Then we do a similar comparison in Fig. 5.10 for the individuals above and below the infection gap. Lastly, Fig. 5.11 offers a histogram showing the internal distribution of susceptible visitors for the different lectures.

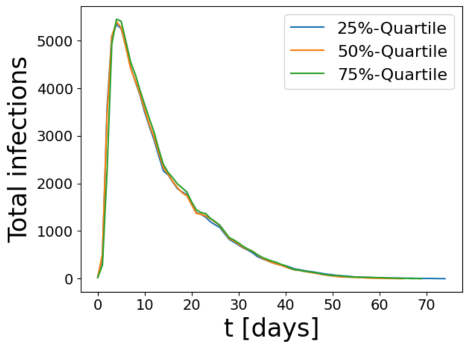
Several entries from Table 5.2 and alternative aggregations are not shown in this Subsection. First of all, infection curves in this Section display the total infections - infected and exposed persons both. Graphs with only the currently active infections are similar enough to need no showcase. However, be aware of their infection peak being more rounded due to the stochastic model of incubation time.

Beyond that, the scenario with only lectures open is less interesting than assumed initially due to data shown in Fig. 5.9. The three points taken from the Pareto front of the infection peak objective yield no new insights into the model either. For actual policy makers These would be more interesting by helping find points with similar infection numbers but lower costs. Lastly, we do not aggregate cost data, since the dynamic cost factors described in Subsection 3.4.3 are set to zero for this thesis. This configuration leads to costs being constant over time for each day of the week.

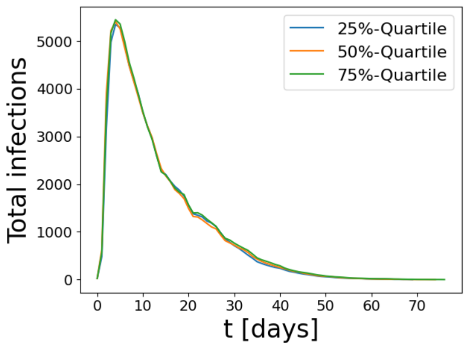
[2]Refer back to Fig. 5.5a for a visualization of this gap in the Pareto front.



(a) No lockdown ($k_{lecture} = 101$, $k_{food} = 500$, $k_{sport} = 40$)

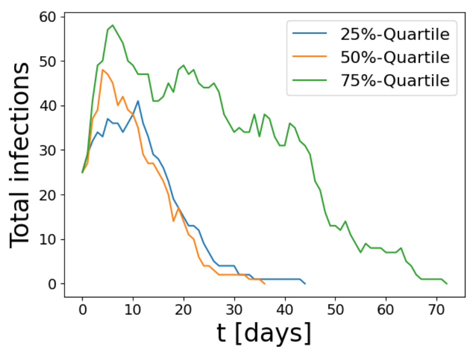


(b) Most cumulative infections ($k_{lecture} = 101$, $k_{food} = 219$, $k_{sport} = 33$)

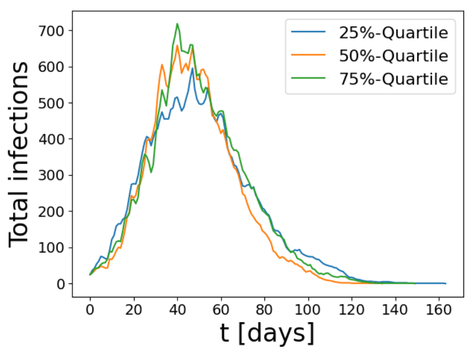


(c) Highest infection peak ($k_{lecture} = 101$, $k_{food} = 418$, $k_{sport} = 34$)

Figure 5.7.: Compare the individuals with most total infections (infected and exposed) for both infection objectives to an artificial policy with no restrictions. The comparison shows the quartiles by infection peak for 199 simulation runs on the large graph.

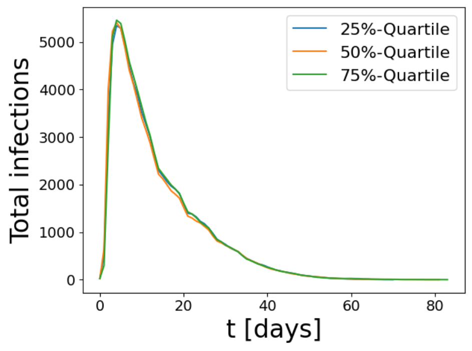


(a) Less infections/below gap ($k_{lecture} = 12$, $k_{food} = 1$, $k_{sport} = 4$)

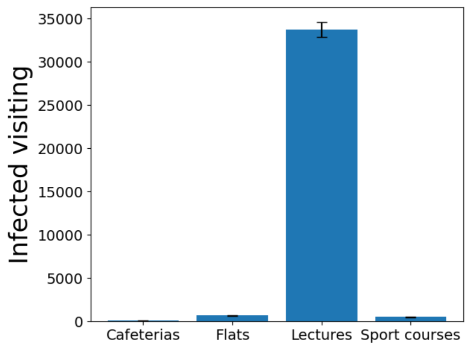


(b) More infections/above gap ($k_{lecture} = 20$, $k_{food} = 0$, $k_{sport} = 9$)

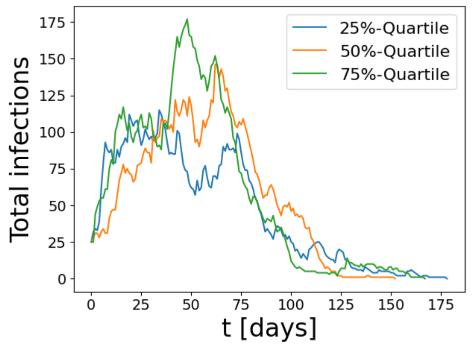
Figure 5.8.: Compare the individuals directly above and below (meaning more and less cumulative infections) the infection gap. For a definition of the infection gap see Section 6.1. The comparison shows the quartiles by infection peak for 199 simulation runs on the large graph. Note that the scale for the y-axis differs between the plots.



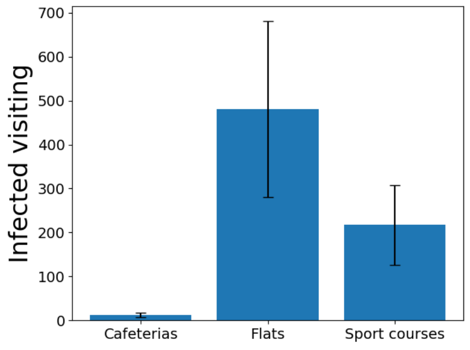
(a) Infection curves without lockdown ($k_{lecture} = 101$, $k_{food} = 500$, $k_{sport} = 40$)



(b) Infected per location type without lockdown

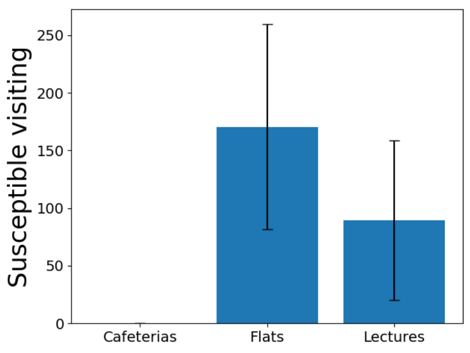


(c) Infection curves for no lectures ($k_{lecture} = 0$, $k_{food} = 500$, $k_{sport} = 40$)

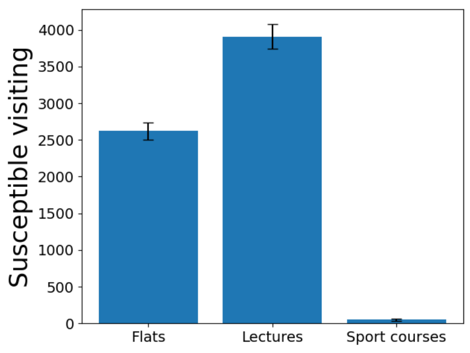


(d) Infected per location type for no lectures

Figure 5.9.: Compare total infection curves on the left with the amount of infected individuals per location type on the right. The location data is averaged over all individual simulation runs, with one σ error range on the bars. Within one run, the values for all time-steps are summed up. Note that infected people are only counted, if the location they visited contained at least one susceptible individual.



(a) Susceptible per location type, below gap ($k_{lecture} = 12$, $k_{food} = 1$, $k_{sport} = 4$)



(b) Susceptible per location type, above gap ($k_{lecture} = 20$, $k_{food} = 0$, $k_{sport} = 9$)

Figure 5.10.: Compare the amount of susceptible individuals per location type for individuals above and below the infection gap (see Section 6.2). The location data is averaged over all individual simulation runs, with one σ error range on the bars. Within one run, the values for all time-steps are summed up. Note that susceptible people are only counted, if the location they visited contained at least one infected.

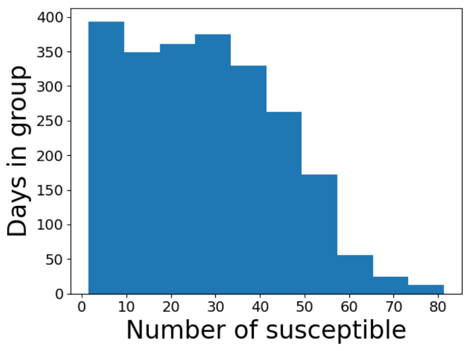


Figure 5.11.: Histogram of average susceptible individuals visiting lectures each day. For each day the visits are summed up over all time-slots. The data displayed here is gathered from 199 runs without lockdown ($k_{lecture} = 101$, $k_{food} = 500$, $k_{sport} = 40$) on the large graph.

6. Evaluation and Discussion

This Chapter refines and interprets the data gathered in the previous Chapter. We separate it by experiment stage as well. Since we use them to decide on further experiments, the conclusions reached in each Section of this Chapter in turn influence the results of the next Section.[1]

Section 6.1 begins by discussing front aggregation from multiple runs of an experiment. After that we do a preliminary comparison of low and high sampling rates and the problems of low sample rates. Deciding on a primary EMO algorithm rounds out the first Section. The second Section concludes the sampling analysis. Then it discusses the different graphs and objectives. Lastly, Section 6.3 investigates interesting individuals (as defined in Chapter 5) in depth. We conclude it with a discussion about the validity of our chosen modeling approach.

6.1. Evaluation of Stage 1 - Aggregation, Sampling and Algorithm Choice

As mentioned in Section 5.1 we run some experiments multiple times. Consequently, we need a method to aggregate their results and describe it in Subsection 5.1.1. First consider our approach - only keeping non-dominated solutions - under the assumption of perfect fitness values: Each individual is simulated as many times as necessary to return the true median of its fitness distribution. In this case our aggregation method yields a front equal to or better than its constituent iterations. The resulting front is less cluttered compared to plotting individual iterations. Due to these advantages the aggregation is closer to the real Pareto front and easier to interpret.

[1]This Chapter and the previous one are interleaved. For each Section, refer back to the corresponding Section in Chapter 5 for details.

Fig. 5.1 shows an example for the experiments run with 23 samples per fitness evaluation. With the high-fidelity fitness values the plot resembles the ideal scenario we described. Individual iterations in the plot look very similar, which is true for the other algorithms and objective combinations as well. Additionally, we compute hypervolumes for the different objectives (and NSGA-III). They can be found in Table 6.1 and lead to the same conclusion: For our small number of runs we find only small differences between iterations. Thus, we can assume the aggregation offers a very slight advantage over the individual iterations. Consequently, further evaluation takes advantage of this aggregation method to reduce visual clutter.

Table 6.1.: Comparing hypervolumes of the aggregated front to the individual iterations for NSGA-III and all objective combinations. Fitness values are normalized using values of 5000 for the infection objectives and 25000 for the cost.

Objectives	Iteration 1	Iteration 2	Iteration 3	Aggregated
$f_{cumulative}, f_{cost}$	0.245	0.242	0.242	0.247
f_{peak}, f_{cost}	0.447	0.448	0.448	0.450
$f_{cumulative}, f_{peak}, f_{cost}$	0.199	0.216	0.201	0.218

Under less ideal conditions aggregation using the non-dominated set has a disadvantage: With low-fidelity fitness values and few samples an individual can get "lucky". A simple example: We take three fitness samples for an individual. For all three the initial infected do not pass on the sickness (due to chance). Thus, we assume the individual's fitness to be very good. Using more samples would instead have shown a higher infected count in the median.

In practice, this problem appears especially for individuals with less stringent containment measures. Normally, those have less overall cost, but higher infections as a trade-off. Due to the randomness inherent in the model, initial infected can fail to infect anyone else as described in the previous paragraph. If this happens, the relevant individual will have both low cost and low infections.

We see this effect clearly in Fig. 5.2. It show an *infection gap* regardless of the algorithm used, which is more pronounced for the low sample rate. In theory several phenomena can cause such a gap:

- Low sample rates lead to "lucky" individuals as mentioned before. We see this effect in the difference the sample rate makes in Fig. 5.2.

- The EMO algorithm might not explore the space of the infection gap due to a suboptimal diversity mechanic.
- A full lockdown limits infection spread to the living arrangements of the initial infected. Once the epidemic first spreads beyond that, total infections make a jump upwards. Thus, the modeling of interactions between people and infection mechanics form a minimal (natural) infection gap.

The last point is important, because individuals at the edges of this natural gap are interesting to explore in more depth. Because of that the infection gap is relevant at the end of Section 6.2 when we pick out individuals for further analysis. We need to investigate sampling rates as a result of the first point mentioned above. Section 6.2 deals with the topic, because the additional 7-sample evaluation offers more data for interpretation.

For the moment we consider the low sample rate a negative example. It highlights the inherent difficulties of fitness evaluation. The similarities between algorithms (Fig. 5.2) and experiment iterations (Fig. 5.1) for the high sample rate imply a much higher fidelity for 23 samples. Furthermore, Fig. 5.3 shows a more pronounced gap for the $f_{cumulative}$ objective compared to f_{peak}.

Thus, we need to choose an algorithm that minimizes the impact of point two above. As a baseline we compare NSGA-III and MOEA/D to NSGA-II. The former algorithms both use reference directions of some kind to improve diversity, which should help with closing the gap. Since the infection gap is most pronounced for the $f_{cumulative}$ objective, we choose is as the basis for comparison. Fig. 5.2 shows a similar performance for the aggregated fronts of all three algorithms.

Table 6.2.: Comparing hypervolumes of the different EMO algorithms for different objectives. Fitness values are normalized using values of 5000 for the infection objectives and 25000 for the cost. Note that MOEA/D has no aggregated front due to there being only one iteration.

Objectives	NSGA-II	NSGA-III	MOEA/D
$f_{cumulative}, f_{cost}$	**0.248**	0.247	0.243
f_{peak}, f_{cost}	**0.450**	**0.450**	0.447
$f_{cumulative}, f_{peak}, f_{cost}$	0.207	**0.218**	0.194

Table 6.2 shows the hypervolumes for the comparison to further back this observation up. Again all three algorithms show similar results. MOEA/D having slightly smaller hypervolumes across the board can be explained with there being only one iteration for the algorithm. Consequently, algorithm performance was not improved via non-dominated aggregation.

Much more important is the MOEA/D run-time of over 17 days: Table 5.1 shows the algorithm taking a lot longer for each experiment. Likely, its manually parallelized implementation is not as efficient as the others. For this reason we do not consider MOEA/D for further experiments in the second stage.

The choice between NSGA-II and NSGA-III proves more difficult. Both show similar run-times and hypervolumes. While the infection gap for NSGA-III in Fig. 5.2 looks smaller, three iterations are not enough to make that observation with statistical significance. Thus, we face a trade-off between NSGA-II being slightly faster - again with too few data points for significance - and NSGA-III providing a potentially smaller gap via reference directions.

Our final choice for the experiments in Section 5.2 falls to NSGA-III. We prefer it for two reasons: The difference in run-time is small enough to not matter for the number of experiments run. Furthermore, the large graph from the next stage showed a more pronounced gap during development. This fact makes NSGA-III's reference directions more attractive.

6.2. Experiments Stage 2 - Evaluation of Optimization and Finding Interesting Individuals

The data resulting from the second round of experiments can be found in Section 5.2.1. We need to conclude the sampling discussion started in the previous Section before talking about the differences between the different graphs and objectives. Relevant to this is Fig. 5.4. It shows a comparison between the different sampling rates for the $f_{cumulative}$ and f_{cost} objectives on the large graph. This comparison show 7-sampling between the low and high sample rates quality wise.

According to this data (Fig. 5.4) both low and medium sample rates do not produce adequate fitness values. Far more importantly: Can we trust the results from the high sample rate. To answer this Subfig. 5.4b shows an example histogram for one individual's cumulative infection fitness. Individuals at the infection gap might have a different, bi-modal distribution for their fitness values. Such distributions do not invalidate our considerations, but would make them more complicated mathematically.

We already laid out the worst case for an optimization experiment in Section 6.1: More that half of an individual's fitness samples have values a lot lower than the true median. We can approximate the likelihood for this via treating the histogram data as a normal distribution. Table 6.3 lists the probabilities of more than half of the fitness samples being 1σ or 2σ below the average (which is close enough to the median for our use-case).

Table 6.3.: This Table lists the chance of an individual's fitness being significantly different from the median of the underlying distribution depending on the sampling rate. The data of the histogram in Fig. 5.4 forms the basis for this calculation. Its average value is 8 627.2 and $\sigma = 17.95$.

	1-sampling	7-sampling	23-sampling	199-sampling
Outside 1σ	15.87%	$6.34 * 10^{-4}\%$	$2.54 * 10^{-10}\%$	$1.11 * 10^{-80}\%$
Outside 2σ	2.28%	$2.68 * 10^{-7}\%$	$1.92 * 10^{-20}\%$	$4.99 * 10^{-165}\%$

One optimization experiment has 10^4 fitness evaluations. The Table shows clearly that we can expect such divergent individuals for 1-sampling (10^4 samples) and 7-sampling ($7*10^4$ samples). At the same time, the expected number of individuals with anomalous fitness (1σ) for the high sample ($23*10^4$ samples) rate is only $2.54*10^{-6}$. Thus, fitness values for 23 samples are trustworthy and our experiment results not invalid on these grounds. Since individual simulations in the third stage, Section 6.3, run 199 times we added the last column of Table 6.3 .

After making sure the sample rate used is high enough, we consider the different graph sizes. For this Fig. 5.5 contains normalized and condensed data for all graphs and objectives. The first difference we notice is a right shift for both f_{peak} and $f_{cumulative}$ on the smaller graphs. This effect occurs purely due to value normalization. Because the number of initial infections does not scale along with the graph, the minimal number of infected is relatively larger on the small graph. For reasons to avoid downscaling of initial infections, see the end of Subsection 2.1.3.

In the same vein, the (leftmost) infection gap for the $f_{cumulative}$ objective scales slightly with graph size. There is a second gap in the front, separating policy sets with about half the people infected from those with almost total epidemic spread. Furthermore, the f_{peak} objective shows more structure on the larger graphs. A clear curve around 0.75 for the normalized f_{peak} marks these individuals as interesting for further study.

In summary graph scaling accomplishes its purpose: The small graphs are similar enough to the full version for efficient development and early testing. On the other hand, the large graph offers more structures within and therefore a more interesting Pareto front for in-depth work.

Next, we compare the possible objective combinations. We highlighted some differences between the $f_{cumulative}$ and f_{peak} objectives during graph comparison. So far, we did not consider optimization using all three objectives. Looking back to Fig. 5.5, the three objective experiments seem suspiciously similar to the other combinations. Arguably, we can remove either infection objective and get the plot with the other in return. Thus, we investigate a possible correlation between $f_{cumulative}$ and f_{peak}. As a first step, Fig. 6.1 offers a plot containing $f_{cumulative}$ and f_{peak} plotted against each other.

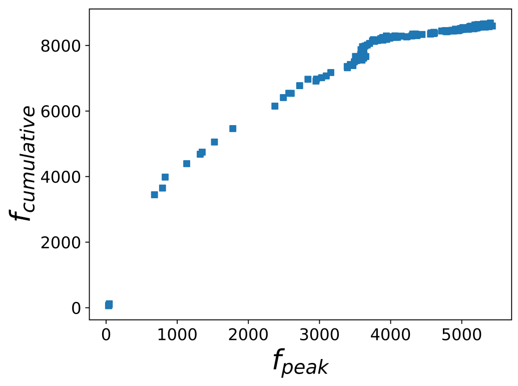


Figure 6.1.: Scatterplot that plots the infection peak and cumulative infections objectives against each other. The data is from the large graph and using all three objectives, then stripping out the cost objective.

The Figure implies a relationship between the objectives, though not a linear one. For this reason we calculate Spearman's [22] correlation for the data set. As a rank-based correlation method it does not require objective values to be distributed in any specific way[2]. Furthermore, Spearman's correlation tries to find monotonous relationships between variables. Such a relationship between the objectives seems reasonable when looking at Fig. 6.1. An overview of the resulting correlation values between objectives can be found in Table 6.4.

Table 6.4.: Spearman correlation matrix [22] for all three objectives. Data is from the large graph and aggregated experiment runs with all three objectives.

	$f_{cumulative}$	f_{peak}	f_{cost}
$f_{cumulative}$	1	0.971	-0.981
f_{peak}	0.971	1	-0.997
f_{cost}	-0.981	-0.997	1

The Table is based on a small number of experiments and thus the results not really statistically significant. Still, we can conclude that - for our specific model and the experiments run - a monotonous relationship between the infection objectives exists. Thus, using all three objectives at the same time is unlikely to yield gains.

On the other hand, Table 6.4 shows either infection objective to be in conflict with f_{cost}. Due to this, the resulting two-objective problems actually produce interesting Pareto fronts. We can pick the infection objective that fits our current goals better.

A further comparison of $f_{cumulative}$ and f_{peak} in Fig. 5.6 shows search and objective space beside each other. Together with Fig. 5.5 both objectives display advantages and disadvantages. For the $f_{cumulative}$ objective infection gaps are more pronounced. At the same time, the search focuses more strongly on a part of the search space where eating in the cafeteria is locked down strongly. This implies a stronger impact of the cafeteria policy on the total infections.[3] The infection peak objective on the other hand explores the search space more uniformly. It offers a front in objective space with a more interesting shape including knee points.

To conclude this stage's evaluation, only choosing individuals (from Fig. 5.5) to evaluate in the next stage remains. Setting all policy values to their mini-

[2]Note that the $f_{cumulative}$ objective is not normally distributed, making Pearson's [22] correlation useless.

[3]In contrast to Section 6.3, where we find cafeteria visits to account for few meetings between infected and susceptible people.

mum and maximum is the baseline. We compare these parameter sets to the individuals with highest and lowest $f_{cumulative}$ found by the optimization. Furthermore, we want to investigate the infection gap (see Section 6.1) and choose individuals on both sides of it for the same objective. Since optimization of the $f_{cumulative}$ objective (in Fig. 5.6) focuses on keeping lectures open and shutting everything else down, we pick an individual from this corner. Points from the curves of the f_{peak} objective as well as the highest peak individual are chosen to round the set of investigated individuals out .

6.3. Experiments Stage 3 - Evaluation of the Location-Based Model

This Section starts with the first question formulated in Section 5.3: Are the policies found by the optimization in the second stage of experiments valid and diverse? For this see Fig. 5.7. It compares the upper extremes (in terms of infected) found by each infection objective to an artificial scenario without lockdown. As the plots clearly show, these are basically identical. Data on the other side of the objective space looks the same, though we omit a plot of it.

Thus, the EMO search finds the edges of the Pareto front we expect. Looking back to the fronts in Fig. 5.5, diversity between these points is decent on first glance. This further underlines the conclusion of Section 6.2: Using EMO approaches to find policy parameters to contain epidemic spread in our model works and returns useful data.

At the same time the overall question of model validity remains. One example of this is the infection gap we first discuss in Section 6.1. We argue that for a high enough sampling rate this gap is an inherent part of the epidemic model and not a deficit of the search approach. Section 6.2 further backs this up with the analysis that our sample rate for an individual's fitness is high enough. With the data from experiment stage three, we can confirm this gap as a result of the model.

Fig. 5.8 shows a clear difference between policy parameter sets above and below the infection gap. Note that in contrast to the infection curves we discussed so far, the policy parameters below the gap (Subfig. 5.8a) exhibit a less pronounced peak. Here the epidemic tapers off, because only the flatmates of the initial infected contract the disease. In a more typical scenario (Fig.

5.8b) a wave of new infections leads to a much higher and clearer peak. A representation of this mapped to locations is in Fig. 5.10: Below the gap flats dominate for susceptible meeting an infected person. Above that lectures lead to more meetings, kicking of more infections. The epidemic turns out longer and has much more overall infected.

From the data on the infection gap, we conclude that the structure of the modeled graph has a large impact on the front. Different graph structures lead to break points, where the epidemic behavior changes suddenly [38]. Through looking for such breakpoints in optimization we can gain knowledge about the model. Another example of an interesting region is the points we pick out from the infection peak front in Section 5.3.

Overall, the resulting infection curves (see Subsection 5.3.1) are similar to those of classical epidemiological models [38, 27][4]. We can explain divergences like infections not propagating beyond initial infected with full lockdown within the model. Thus, we conclude that the location-based modeling approach used in this thesis is suited to model epidemic spread.

For a location-based model the next question is, if we can identify specific locations central to epidemic spread. In real-world epidemic events the concept of superspreaders matters - individuals that infect a lot of others on their own [49, 38]. How do we stop such superspreaders? If we want to take real-world action depending on predictions about individual people, we face serious ethical questions [52, 68]. We consider superspreading locations a safer topic: Aside from highly infectious individuals [49, 38], epidemic spread at a location depends on total visitors and population density [87]. Our model represents the later via specific infection rates. A more precise modeling could for example treat this infection rate as a function of visitor numbers.

Actually finding superspreading locations is possible via evaluating the number of infected visiting each type of location on average. An example of this can be seen in Fig. 5.9. Susceptible individuals visiting each location type exhibit a very similar distribution. According to the plot lectures dominate epidemic spread by a wide margin. Lectures accounting for most new infections coincides with our assumptions for the university scenario. However, we find the stark difference to the remaining location types concerning. Subfig. 5.9d on the other hand shows the distribution without lectures. While it shows more reasonable

[4]Most papers will plot "active" infections. See the end of Subsection 5.3.1.

ratios for the different locations, the total infections are much lower. Thus, without lectures infection is limited to certain clusters (sport courses) of people. Overall, the question arises, whether we face a discrepancy in our modeling of the university scenario.[5]

Two main error sources can be responsible for this effect: Either a part of the model is wrong or modeled with insufficient detail (or not at all). Despite the simpler structures of small communities (refer back to Chapter 1), the later error is still relevant for them. University is no closed system (unlike an island community). It consists of far more moving parts than our simple rules in Section 4.1 suggest. In this light, we can bring the meetings with infected over location types in line with our assumptions if we extend the modeling. Flats adequately model students' living arrangements, but do not include other activities in their social circle (parties, study groups, ...). Sport courses similarly account for only a small fraction of the hobbies a university student might pursue.

For locations the distribution of susceptible individuals that meet an infected person is reasonable: Fig. 5.11 shows a relatively uniform distribution, which tapers off towards large lectures. This Figure shows no specific superspreading locations - as expected, since we model neither really large lectures (>> 100 attendees), nor large scale events like parties. Lectures dominate infections by sheer number of overall meetings between students. Comparison with our basic assumptions (as detailed the paragraph before last) makes a fault in lecture modeling unlikely.

On the other hand, we consider the number people - regardless of infected or susceptible - in the cafeteria locations too low. In order to make an informed evaluation of our modeling, we take a short look at real-world studies on infections at restaurants and cafeterias: Zuber et al. (2020) [87] find a correlation between eating out and contracting SARS-CoV-2 that is even higher than for public transport. Aerosols serve as the main infection vector for restaurants, instead of close contact [83].

Cafeterias in our university example are situated in larger spaces than the average lecture. Additionally, students sometimes eat outside if the weather is good. Because of this we set the infection rate for eating lower that for

[5]To be clear: We refer to a discrepancy between the model and our assumptions. A comparison between model and real university is difficult due to a lack of empiric data.

lectures (Appendix A). The driving force behind infections via cafeterias is thus the number of visitors. At the least we would expect more meetings between infected and susceptible people than in sport courses.

As a more faithful modeling we suggest two improvements compared to the current model: Cafeteria attendance focuses on midday instead of a completely uniform distribution across opening hours. At the same time both lecturers and students often go eating in small groups, which our model does not represent. Lastly, current generation rules create too few visits overall. These suggestions taken together would further help us to avoid lectures being the only vector for epidemic spread throughout the modeled population.

Such a modeling flaw degrades our model's direct usability for deriving real world conclusions and containment measures. However, this problem highlights an advantage of the location-based modeling approach: Since we generate the graph from simple, scenario-specific rules we can improve it incrementally. Using different objectives in policy optimization helps this workflow of building a preliminary model and then improving it. For example, the f_{peak} objective finds results more evenly distributed in the search space. This observation hints at the objective being more sensitive to non-lecture locations than $f_{cumulative}$.

6.4. Summary of the Evaluation

The previous Section shows that our location-based approach to modeling epidemic spread is feasible. We will highlight similarities and differences with two similar relevant papers during this summary. These comparisons are meant to help readers place our approach among classical epidemic models. First is "Optimal Control Policies to Address the Pandemic Health-Economy Dilemma" by Salgotra et al. (2021) [64]. They focus on the trade-off between containing epidemic spread and its cost in a modified SEIR model, using EMO to find optimal policies as well. Next, "A social network model of COVID-19" by Karaivanov (2020) [38] describes a (peer-to-peer) graph version of the SIR model. Notably, Karaivanov uses the same number of people in his model as in our large graph (Table 4.1) and a lot of our model parameters (Appendix A) are derived from his paper.

Aside from its general feasibility we claim that the location-based approach is an intuitive modeling for certain scenarios, namely small communities. Comparisons with other approaches back this up: Karaivanov [38] uses a variant of scale-free network [82]. This approach is a simplification relative to real-world intricacies [20] and more importantly not easily tailored to specific scenarios. On the other hand, the model of Salgotra et al. [64] misrepresents populations with a lot of structure through the perfect mixing assumption of its SIR model derivative [38]. Our own model works according to simple rules of people visiting locations. For small communities we can often derive such rules (see Section 4.1) easily.

In Section 6.3 we noticed epidemic break points, where the spread changes drastically due to the graph structure (Figs. 5.8 and 5.9). According to Karaivanov [38] peer-to-peer contact graphs exhibit similar behavior. He highlights the graph structures responsible as good targets for precise lockdowns and links them to the concept of superspreaders [49]. We see practical and ethical concerns with containing and especially predicting superspreading individuals (see Section 6.3). Instead our model uses the concept of superspreading locations. Due to data we gather about locations at run-time, we can find and reason about locations that form hot-spots for epidemic spread. These considerations are made easier by the location model because we can tailor it more precisely to a scenario. Thus, we can bring more domain knowledge to bear than in a more generic model.

Another advantage of our location-based model is adaptability. We can improve and extend it incrementally until a sufficient level of detail is reached. The university scenario we model in this thesis (as discussed in Section 6.3) is a good example for this: First we look at the data gathered during simulation runs to understand how our model behaves for the scenario. We notice that lectures account for a majority of meetings between infected and susceptible individuals. This observation is in line with our assumptions. However, cafeteria visits account for too few meetings between people and consequently have to little impact on epidemic spread. Once we identify this mismatch between the model and our assumptions, we suggest changes to the generation rules (Section 6.3).

Furthermore, we conclude that using EMO to optimize policy parameters works. Salgotra et al. [64] arrive at similar conclusions. In their evaluation of different EMO algorithms they find NSGA-II and NSGA-III to work well for their problems, while MOEA/D performs worse. Our analysis in Section 6.1 matches this, backing our choice of NSGA-III for further experiments in Section 5.2. Additionally, Section 6.2 shows that we must take care when determining the fitness of a set of policy parameters in our model. We need to generate enough fitness samples to get an accurate value, since individual simulation runs yield different fitness values.

The choice of optimization goals matters as well. Coupling either f_{peak} or $f_{cumulative}$ with the f_{cost} objective creates different fronts with their own focus areas. In Section 6.2 we show that using all three objectives together brings no direct benefits. All Pareto fronts found by our optimization (see Fig. 5.5) show structure (infection gap, knee points) and are no simple anti-proportional relations. Optimization results for f_{peak} and f_{cost} show visually similar fronts to the results of Salgotra et al. [64]. Knee points found in their experiments are more defined. Possible reasons for this are a differing definition of the cost objective or a different speed of epidemic spread between graph and SIR models [38]. We also reproduce the clear conflict of reducing infections versus minimizing costs Salgotra et al. [64] mention. Overall, we consider optimization via EMO a valid approach to both find interesting policy parameters and guide further model design.

So far our university scenario model is too simplistic to help decision makers in the real world. What steps are necessary to improve the model to this point? Firstly, we need to revise graph generation rules (see Section 6.3) to be more in line with our assumptions regarding our university scenario. The dynamic part of the cost model introduced in Section 3.4 requires exploration. We can refine location dependent infection rates using empiric data or even define them dynamically depending on visitor numbers. Real policies change over time and are often much more complex than our three policy parameters [38, 64, 86]. Chapter 7 ends with some ideas on how we could model dynamic location-based policies.

Beyond that, our university model needs to be evaluated on more and different graphs. We need to review EMO algorithms more in depth than in Section 6.1 to find the best one for optimizing policy parameters. Potential other objectives like time of infection peak are worth exploration. Lastly, we recommend modeling another small community to see how well the conclusions reached in this thesis transfer to new scenarios.

7. Conclusion and Future Work

This thesis describes a location-based approach to epidemic modeling for small communities. Small "communities" like islands, schools, hospitals or similar fill a gap between clinical trials finding the characteristics of a virus and large scale models simulating epidemic spread for entire societies. Such a community offers richer interaction structures and potentially different emergent behavior from society at large. As an example, we model a university scenario in this thesis and optimize policy parameters using EMO.

After considering related work (Chapter 2) we decide on building a location-based graph model. Chapter 3 describes the theory behind it - people visiting locations at different times represented via a bipartite graph. We define the goals of low infection peaks (f_{peak}), few cumulative infections ($f_{cumulative}$) and minimal cost of containment measures (f_{cost}). With different EMO algorithms we optimize three policy parameters for our scenario: The maximal allowed lecture size, people allowed into cafeterias and sport course size. Implementation details (Chapter 4) are followed by a description of our experiments in Chapter 5.

Our evaluation starts with choosing NSGA-III as primary EMO algorithm (Section 6.1) after initial experiments. We continue with experiments on different graph sizes and with different objective combinations. 23 simulation runs prove a good choice for deciding the fitness of a parameter set (Section 6.2), with lower sample rates showing much worse results. Lastly, we look at specific individuals in details to better understand the proposed model (Section 6.3).

In Section 6.4 we conclude that both the location-based model and policy optimization using EMO are viable and intuitive ways to build and understand an epidemiological model for small communities. Furthermore, we highlight the advantages of our model: Graph generation from simple rules, adaptability and through that incremental improvement of the model. Where our university model diverges from our assumptions we are able to make clear recommendations for better generation rules (Section 6.3). Policy optimization complements model analysis by finding interesting sets of policy parameters on the structure-rich Pareto front created from the conflict between keeping infections low and minimizing costs. Regarding optimization goals we find both $f_{cumulative}$ and f_{peak} objectives offering different advantages when paired with f_{cost}. However, using all three objectives at the same time yields no real benefit (Section 6.2).

Aside from exploring the existing university model further as described in Section 6.4 there is a plethora of possible extensions: Testing of infected without symptoms; vaccinations; modeling risk groups with a higher chance of dying to infection; diversification of symptom modeling beyond "yes" and "no". Regarding policy optimization we can introduce more parameters to represent more complex policies. Beyond that, a policy could actually decide dynamically on lockdowns for each location and time-step. One example of a possible approach to dynamic policies is training a decision tree using EMO [37, 10]. Lastly, we can easily integrate the location-based approach into other epidemiological models by generating a classical contact graph from the location-based representation.

Bibliography

[1] Adeshina Adekunle, Michael Meehan, Diana Rojas-Alvarez, James Trauer, and Emma McBryde. Delaying the COVID-19 epidemic in Australia: evaluating the effectiveness of international travel bans. *Australian and New Zealand Journal of Public Health*, 44(4):257–259, aug 2020.

[2] Jorge Andrés Sánchez-Duque, Juan Pablo Orozco-Hernández, Daniel Stiven, Marín Medina, and Alfonso J Rodriguez-Morales. Economy or Health, Constant Dilemma in Times of Pandemic: The Case of Coronavirus Disease 2019 (COVID-19). *researchgate.net*, 14(1):717–720, 2020.

[3] Nino Antulov-Fantulin, Alen Lančić, Hrvoje Štefančić, and Mile Šikić. FastSIR algorithm: A fast algorithm for the simulation of the epidemic spread in large networks by using the susceptible-infected-recovered compartment model. *Information Sciences*, 239:226–240, 2013.

[4] N Arshed, MS Meo, and F Farooq. Empirical assessment of government policies and flattening of the COVID19 curve. *researchgate.net*, 20(4), nov 2020.

[5] D Bacciua, A Michelia, M Poddaa Stat, and undefined 2020. Edge-based sequential graph generation with recurrent neural networks. *researchgate.net*, 2020.

[6] Benjamin Bach, Andre Spritzer, Evelyne Lutton, and Jean Daniel Fekete. Interactive random graph generation with evolutionary algorithms. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 7704 LNCS, pages 541–552. Springer, Berlin, Heidelberg, 2013.

[7] NTJ Bailey. *The mathematical theory of infectious diseases and its applications.* 1975.

[8] Steven C. Bankes. Agent-based modeling: A revolution?, may 2002.

[9] A. Barrat and M. Weigt. On the properties of small-world network models. *European Physical Journal B*, 13(3):547–560, feb 2000.

[10] Rodrigo C Barros, Márcio P Basgalupp, André C P L F de Carvalho, and Alex A Freitas. A Survey of Evolutionary Algorithms for Decision Tree Induction. Technical report.

[11] Cristiane M. Batistela, Diego P.F. Correa, Átila M. Bueno, and José Roberto C. Piqueira. SIRSi compartmental model for COVID-19 pandemic with immunity loss. *Chaos, Solitons and Fractals*, 142:110388, jan 2021.

[12] Antonio Benítez-Hidalgo, Antonio J. Nebro, José García-Nieto, Izaskun Oregi, and Javier Del Ser. jMetalPy: A Python framework for multi-objective optimization with metaheuristics. *Swarm and Evolutionary Computation*, 51:100598, dec 2019.

[13] Ottar N. Bjørnstad, Katriona Shea, Martin Krzywinski, and Naomi Altman. The SEIRS model for infectious disease dynamics. *Nature methods*, 17(6):557–558, jun 2020.

[14] Julian Blank and Kalyanmoy Deb. pymoo: Multi-objective Optimization in Python. *IEEE Access*, 8:89497–89509, jan 2020.

[15] Per Block, Marion Hoffman, Isabel J. Raabe, Jennifer Beam Dowd, Charles Rahal, Ridhi Kashyap, and Melinda C. Mills. Social network-based distancing strategies to flatten the COVID-19 curve in a post-lockdown world. *Nature Human Behaviour*, 4(6):588–596, 2020.

[16] Jürgen Branke, Kalyanmoy Deb, Henning Dierolf, and Matthias Osswald. Finding knees in multi-objective optimization. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3242:722–731, 2004.

[17] JC Butcher. A history of Runge-Kutta methods. *math.uoc.gr*, 20:247–260, 1996.

[18] David Camacho, Ángel Panizo-LLedot, Gema Bello-Orgaz, Antonio Gonzalez-Pardo, and Erik Cambria. The four dimensions of social network analysis: An overview of research methods, applications, and software tools. *Information Fusion*, 63(1):88–120, 2020.

[19] Saptarshi Chatterjee, Apurba Sarkar, Swarnajit Chatterjee, Mintu Karmakar, and Raja Paul. Studying the progress of COVID-19 outbreak in India using SIRD model. *Indian Journal of Physics 2020*, pages 1–17, jun 2020.

[20] Ning Ning Chung and Lock Yue Chew. Modelling Singapore COVID-19 pandemic with a SEIR multiplex network model, jun 2020.

[21] Daniel Cordeiro, Grégory Mounié, Swann Pérarnau, Denis Trystram, Jean-Marc Vincent, and Frédéric Wagner. Random graph generation for scheduling simulations. page 10, mar 2010.

[22] Joost C.F. de Winter, Samuel D. Gosling, and Jeff Potter. Comparing the pearson and spearman correlation coefficients across distributions and sample sizes: A tutorial using simulations and empirical data. *Psychological Methods*, 21(3):273–290, sep 2016.

[23] K Deb, A Pratap, S Agarwal, T Meyarivan IEEE TRANSACTIONS ON, and Undefined 2002. NSGA-II. *cse.unr.edu*, 6(2), 2002.

[24] Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, 2014.

[25] Jonas Dehning, Johannes Zierenberg, F. Paul Spitzner, Michael Wibral, Joao Pinheiro Neto, Michael Wilczek, and Viola Priesemann. Inferring change points in the spread of COVID-19 reveals the effectiveness of interventions. *Science*, 369(6500), jul 2020.

[26] H. Ehrig, M. Pfender, and H. J. Schneider. Graph-grammars: An algebraic approach. pages 167–180, jul 2008.

[27] Jesús Fernández-Villaverde and Charles I. Jones. Estimating and Simulating a SIRD Model of COVID-19 for Many Countries, States, and Cities. may 2020.

[28] Aidan Findlater and Isaac I. Bogoch. Human Mobility and the Global Spread of Infectious Diseases: A Focus on Air Travel. *Trends in Parasitology*, 34(9):772–783, sep 2018.

[29] Antonio Gaspar-Cunha, Jose Covas, A Gaspar-Cunha, and J A Covas. A Real-World Test Problem for EMO Algorithms. *researchgate.net*, 2632:752–766, 2003.

[30] Maira Gatti de Bayser, Paulo Rodrigo Cavalin, Claudio Pinhanez, Cicero Nogueira Dos Santos, Maíra Gatti, Ana Paula Appel, Cícero dos Santos, Daniel Gribel, Paulo Cavalin, and Samuel Barbosa Neto. Large-Scale Multi-agent-Based Modeling and Simulation of Microblogging-Based Online Social Network Classification of Life Events on Social Media View project Life Events Analytics View project Large-Scale Multi-Agent-based Modeling and Simulation of Micro. *researchgate.net*, 8235 LNAI:17–33, 2014.

[31] Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. In *Journal of Physical Chemistry*, volume 81, pages 2340–2361, 1977.

[32] Philip T. Gressman and Jennifer R. Peck. Simulating COVID-19 in a university environment. *Mathematical Biosciences*, 328:108436, oct 2020.

[33] Christian Horoba and Frank Neumann. Benefits and drawbacks for the use of ϵ-dominance in evolutionary multi-objective optimization. *GECCO'08: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation 2008*, pages 641–648, 2008.

[34] B. Ivorra, M. R. Ferrández, M. Vela-Pérez, and A. M. Ramos. Mathematical modeling of the spread of the coronavirus disease 2019 (COVID-19) taking into account the undetected infections. The case of China. *Communications in Nonlinear Science and Numerical Simulation*, 88:105303, sep 2020.

[35] Parnian Jabbari and Nima Rezaei. With Risk of Reinfection, Is COVID-19 Here to Stay?, aug 2020.

[36] Himanshu Jain and Kalyanmoy Deb. An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, Part II: Handling constraints and extending to an adaptive approach. *IEEE Transactions on Evolutionary Computation*, 18(4):602–622, 2014.

[37] Dariusz Jankowski and Konrad Jackowski. Evolutionary algorithm for decision tree induction. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8838, pages 23–32. Springer Verlag, 2014.

[38] Alexander Karaivanov. A social network model of COVID-19. *PLoS ONE*, 15(10 October):e0240878, oct 2020.

[39] Hamdi Kavak, Dieter Pfoser, Joon Seok Kim, Carola Wenk, Andrew Crooks, and Andreas Züfle. Location-based social simulation. In *ACM International Conference Proceeding Series*, pages 218–221, New York, NY, USA, aug 2019. Association for Computing Machinery.

[40] W 0 Kermack and A G Mckendrick. 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, aug 1927.

[41] Kyung Hyun Kim, Eun Hwa Choi, and Seung Ki Kim. Editorial. COVID-19 outbreak and its countermeasures in the Republic of Korea. *Journal of Neurosurgery*, 133(1):29–30, apr 2020.

[42] Mikko Kivelä, Marc Barthelemy, James P Gleeson, Yamir Moreno, and Mason A Porter. Multilayer Networks. Technical report, 2014.

[43] A. A. Kochkarov, R. A. Kochkarov, and G. G. Malinetskii. Issues of dynamic graph theory. *Computational Mathematics and Mathematical Physics*, 55(9):1590–1596, 2015.

[44] R Kruse, C Borgelt, C Braune, and S Mostaghim. *Computational intelligence.*

[45] Luis F Lafuerza and Marian Bogu. Simulating non-Markovian stochastic processes. 042108:1–9, 2014.

[46] S Lasaulce, C Zhang, V Varma Frontiers in Public . . . , and undefined 2021. Analysis of the tradeoff between health and economic impacts of the Covid-19 epidemic. *ncbi.nlm.nih.gov.*

[47] Dion T.S. Li, Lakshman Perera Samaranayake, Yiu Yan Leung, and Prasanna Neelakantan. Facial protection in the era of COVID-19: A narrative review. *Oral Diseases*, 27(S3):665–673, apr 2021.

[48] C M Macal and M J North. Tutorial on agent-based modelling and simulation. *Journal of Simulation*, 4(3):151–162, 2010.

[49] Dasha Majra, Jayme Benson, Jennifer Pitts, and Justin Stebbing. SARS-CoV-2 (COVID-19) superspreader events, 2021.

[50] Andrea Mambrini and Dario Izzo. PaDe: A parallel algorithm based on the MOEA/D framework and the island model. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8672:711–720, 2014.

[51] D Caccavo MedRxiv and undefined 2020. Chinese and Italian COVID-19 outbreaks can be correctly described by a modified SIRD model. *medrxiv.org*.

[52] Richard Murphy and Gill Wyness. Minority report: the impact of predicted grades on university admissions of disadvantaged groups. *https://doi.org/10.1080/09645292.2020.1761945*, 28(4):333–350, jul 2020.

[53] Murooj Nadhom and Pavel Loskot. Survey of public data sources on the Internet usage and other Internet statistics. *Data in Brief*, 18:1914–1929, jun 2018.

[54] Antonio Jesús Nebro, Juan José Durillo, Antonio J Nebro, and Juan J Durillo. A study of the parallelization of the multi-objective metaheuristic MOEA/D. *Springer*, 6073 LNCS:303–317, 2010.

[55] M E J Newman and Juyong Park. Why social networks are different from other types of networks. Technical report, 2003.

[56] John Norrie. Some challenges of sparse data necessitating strong assumptions in investigating early COVID-19 disease, 2020.

[57] Pandemic Plan. Pandemic Plan of Otto von Guericke University Magdeburg (OVGU). page 19, 2020.

[58] Hazhir Rahmandad and John Sterman. Heterogeneity and network structure in the dynamics of diffusion: Comparing agent-based and differential equation models. *Management Science*, 54(5):998–1014, 2008.

[59] FM De Rainville, FA Fortin, MA Gardner, and M Parizeau. DEAP: A Python Framework for Evolutionary Algorithms. *GECCO'12 - Proceedings of the 14th International Conference on Genetic and Evolutionary Computation Companion*, pages 85–92, 2012.

[60] Jeff Reback, Wes McKinney, Jbrockmendel, Joris Van den Bossche, Tom Augspurger, Phillip Cloud, Gfyoung, Sinhrks, Simon Hawkins, Matthew Roeschke, Adam Klein, Terji Petersen, Jeff Tratner, Chang She, William Ayd, Shahar Naveh, Marc Garcia, Jeremy Schendel, Andy Hayden, Daniel Saxton, Vytautas Jancauskas, Ali McMaster, Pietro Battiston, Skipper Seabold, Patrick, Kaiqi Dong, Chris-b1, H-vetinari, Stephan Hoyer, and Marco Gorelli. pandas-dev/pandas: Pandas 1.1.5. dec 2020.

[61] Daniel Romer and Kathleen Hall Jamieson. Conspiracy theories as barriers to controlling the spread of COVID-19 in the U.S. *Social Science and Medicine*, 263, 2020.

[62] Shubhadeep Roychoudhury, Anandan Das, Pallav Sengupta, Sulagna Dutta, Shatabhisha Roychoudhury, Arun Paul Choudhury, A. B. Fuzayel Ahmed, Saumendra Bhattacharjee, and Petr Slama. Viral Pandemics of the Last Four Decades: Pathophysiology, Health Impacts and Perspectives. *International Journal of Environmental Research and Public Health*, 17(24):1–39, dec 2020.

[63] Zhongyuan Ruan, Chaoqing Wang, Pak Ming Hui, and Zonghua Liu. Integrated travel network model for studying epidemics: Interplay between journeys and epidemic. *Scientific Reports*, 5, 2015.

[64] Rohit Salgotra, Amiram Moshaiov, Thomas Seidelmann, Dominik Fischer, and Sanaz Mostaghim. Optimal Control Policies to Address the Pandemic Health-Economy Dilemma. In *ieeexplore.ieee.org*, pages 720–727, 2021.

[65] Seyed H. Shahcheraghi, Jamshid Ayatollahi, Alaa A.A. Aljabali, Madhur D. Shastri, Shakti D. Shukla, Dinesh K. Chellappan, Niraj K. Jha, Krishnan Anand, Naresh K. Katari, Meenu Mehta, Saurabh Satija, Harish Dureja, Vijay Mishra, Abdulmajeed G. Almutary, Abdullah M. Alnuqaydan, Nitin Charbe, Parteek Prasher, Gaurav Gupta, Kamal Dua, Marzieh Lotfi, Hamid A. Bakshi, and Murtaza M. Tambuwala. An overview of vaccine development for COVID-19. *Therapeutic Delivery*, 12(3):235–244, mar 2021.

[66] Amit Singhal, Pushpendra Singh, Brejesh Lall, and Shiv Dutt Joshi. Modeling and prediction of COVID-19 pandemic using Gaussian mixture model. *Chaos, Solitons and Fractals*, 138:110023, sep 2020.

[67] Brad Spellberg, Travis B. Nielsen, and Arturo Casadevall. Antibodies, Immunity, and COVID-19, apr 2021.

[68] S Spielberg. Minority report. 2012.

[69] Sreenivas R Sukumar and James J Nutaro. Agent-based vs. Equation-based Epidemiological Models A Model Selection Case Study. Technical report.

[70] Michael te Vrugt, Jens Bickmann, and Raphael Wittkowski. Effects of social distancing and isolation on epidemic spreading modeled via dynamical density functional theory. *Nature Communications 2020 11:1*, 11(1):1–11, nov 2020.

[71] Vikram Thakur and Anu Jain. COVID 2019-suicides: A global psychological pandemic. *Brain, Behavior, and Immunity*, 88:952, aug 2020.

[72] Dijana Tolić, Kaj Kolja Kleineberg, and Nino Antulov-Fantulin. Simulating SIR processes on networks using weighted shortest paths. *Scientific Reports*, 8(1):1–10, 2018.

[73] Christian L Vestergaard and Mathieu Génois. Temporal Gillespie Algorithm : Fast Simulation of Contagion Processes on Time- Varying Networks. pages 1–28, 2015.

[74] Pauline Vetter, Diem Lan Vu, Arnaud G. L'Huillier, Manuel Schibler, Laurent Kaiser, and Frederique Jacquerioz. Clinical features of covid-19, apr 2020.

[75] Chengdi Wang, Zhoufeng Wang, Guangyu Wang, Johnson Yiu Nam Lau, Kang Zhang, and Weimin Li. COVID-19 in early 2021: current status and looking forward. *Signal Transduction and Targeted Therapy 2021 6:1*, 6(1):1–14, mar 2021.

[76] Chongying Wang and Hong Zhao. The Impact of COVID-19 on Anxiety in Chinese University Students. *Frontiers in Psychology*, 11:1168, may 2020.

[77] S Wasserman and K Faust. Social network analysis: Methods and applications. 1994.

[78] Bryan Wilder, Michael J. Mina, and Milind Tambe. Tracking disease outbreaks from sparse data with Bayesian inference. 2020.

[79] RJ Wilson. Introduction to Graph Theory. *llrc.mcast.edu.mt*.

[80] David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[81] Zifeng Yang, Zhiqi Zeng, Ke Wang, Sook San Wong, Wenhua Liang, Mark Zanin, Peng Liu, Xudong Cao, Zhongqiang Gao, Zhitong Mai, Jingyi

Liang, Xiaoqing Liu, Shiyue Li, Yimin Li, Feng Ye, Weijie Guan, Yifan Yang, Fei Li, Shengmei Luo, Yuqi Xie, Bin Liu, Zhoulang Wang, Shaobo Zhang, Yaonan Wang, Nanshan Zhong, and Jianxing He. Modified SEIR and AI prediction of the epidemics trend of COVID-19 in China under public health interventions. *Journal of Thoracic Disease*, 12(3):165–174, mar 2020.

[82] V. N. Zadorozhnyi and E. B. Yudin. Structural properties of the scale-free Barabasi-Albert graph. *Automation and Remote Control*, 73(4):702–716, 2012.

[83] Nan Zhang, Xuguang Chen, Wei Jia, Tianyi Jin, Shenglan Xiao, Wenzhao Chen, Jian Hang, Cuiyun Ou, Hao Lei, Hua Qian, Boni Su, Jiansen Li, Dongmei Liu, Weirong Zhang, Peng Xue, Jiaping Liu, Louise B. Weschler, Jingchao Xie, Yuguo Li, and Min Kang. Evidence for lack of transmission by close contact and surface touch in a restaurant outbreak of COVID-19. *Journal of Infection*, 83(2):207–216, aug 2021.

[84] Qingfu Zhang and Hui Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, dec 2007.

[85] Sheng Zhang, Meng Yuan Diao, Wenbo Yu, Lei Pei, Zhaofen Lin, and Dechang Chen. Estimation of the reproductive number of novel coronavirus (COVID-19) and the probable outbreak size on the Diamond Princess cruise ship: A data-driven analysis. *International Journal of Infectious Diseases*, 93:201–204, apr 2020.

[86] Yi Zhang and Sanjiv Kapoor. Strategic release of lockdowns in a COVID infection model *. Technical report.

[87] Sophie Zuber and Harald Brüssow. COVID 19: challenges for virologists in the food industry. *Microbial Biotechnology*, 13(6):1689–1701, nov 2020.

A. Model and Simulation Parameters

Table A.1.: Parameters of the graph model used for the university example.

Parameter	Value	Why was the value chosen?
Total number of people	10000	Estimate for the main OvGU campus; similar size to Karaivanov (2020) [38]
Number of students	7500	Estimate from personally observed ration of students to staff
Number of lecturers	1500	Estimate from personally observed ration of students to staff
Number of remaining staff	1000	Remainder of people; could include more than cafeteria staff in next version
Maximal lecture attendees	100	Personal estimate from faculty of computer science; upwards outliers (e.g. 200 attendees) are not modeled
Minimal lecture attendees	5	Personal estimate of minimal size
Number of lectures	2337	Taken from OvGU system for winter semester 20/21
Maximal sport attendees	40	Personal estimate of good group size
Minimal sport attendees	5	Personal estimate of minimal viable size
Number of sports	75	Estimate from OvGU university sport courses
Maximal flat size	5	Personal estimate
Minimal flat size	2	Personal estimate
Number of cafeterias	4	Counted from main OvGU campus
Cafeteria opening days	Monday to Friday	
Cafeteria opening hours	9 to 15	
Sport course hours	15 to 19	

Table A.2.: Parameters used in the simulation.

Parameter	Value	Why was the value chosen?
Base infectivity rate	50%	From Karaivanov (2020) [38]; used for lectures, flats and global infection
Sport infection rate	70%	Increased for close contact between people
Cafeteria infection rate	40%	Lowered due to people doing take-out or eating outside
Initial infections	25	Found via testing; see Rahmmandad et al (2008) [58] as well
Global meeting rate	5%	Personal estimate; kept low on purpose to avoid homogeneous mixing
Incubation rate	19.2%	Incubation period is roughly 5 days according to Karaivanov (2020) [38].
Mortality rate	5%	Calculated from Karaivanov (2020) [38]
Recovery rate	15%	Calculated from Karaivanov (2020) [38]
Symptom rate	50%	Vetter et al. (2020) [74]; though still a guess, because clear data is not available
Cost for lecturer attending lecture	5	Personal estimate
Cost for student attending lecture	2	Personal estimate
Cost for attending sport course	1	Personal estimate
Cost for eating in cafeteria	1	Personal estimate
Cost for working at cafeteria	5	Personal estimate
Cost factor quarantined flat	0	See Subsection 3.4.3
Cost factor canceled lecture	0	See Subsection 3.4.3
t_{max}	180	Roughly number of days in a semester

Declaration of Authorship

I hereby declare that this thesis was created by me and me alone using only the stated sources and tools.

Fabian Richardt Magdeburg, 19.11.2021