ActoDemic: A Distributed Framework for Fine-Grained Spreading Modeling and Simulation in Large Scale Scenarios

Mattia Pellegrino[a], Gianfranco Lombardo[a], Monica Mordonini[a], Michele Tomaiuolo[a], Stefano Cagnoni[a] and Agostino Poggi[a]

[a]Department of Engineering and Architecture University of Parma Parma, Italy

Abstract

Agent-based modeling and simulation techniques are widely and successfully used for analyzing complex and emergent phenomena in many research and application areas. Among the many different reasons which sustain the flexibility and success of such techniques, it is important to mention the availability of a great variety of software tools, easing (1) the development of models, (2) the execution of simulations, and (3) the analysis of results. Currently, with the rapid global spread of the COVID-19 pandemic, one of the most important research area is dedicated to define algorithms and systems to support epidemic forecasting simulations, scalable on large populations. In particular, in this paper, we propose an agent-based epidemic model and a distributed architecture that can be used for the simulation of populations represented by millions of agents. Moreover, the paper presents the results of the simulations on the data of the population of Lombardy.

Keywords

Epidemic modeling, multi-agent simulation, simulation, actor model, ABMS

1. Introduction

Spreading phenomena are widely diffused in the real world; for this reason, their modeling is crucial in several domains. For example, in biological systems, it is important to model how an infectious pathogen spreads over a population [1]; in cybersecurity, it is necessary to understand how a digital virus spreads over the nodes of a network [2]; in sociology and in Social Network Analysis, it is interesting to track how opinions and behaviors spread among a community [3]; in economics, it is interesting to study the way companies in different sectors are affected by the spreading of financial chain effects. The common point of all of these systems is that they can be represented by active entities which interact following different behaviours and models that can be generalized as a sociality factor. Different modeling techniques have been proposed to model spreading phenomena in real and complex scenarios. Two widely used techniques are System Dynamics (SD) and Agent-Based Modeling (ABM). System Dynamics analyzes the modeled system at a high abstraction level, where the interacting entities are divided into compartments. A common case is the epidemiological SEIR model (Susceptible Exposed Infective Recovered) [4], where the population can move from one compartment to another according to predefined flow rates. However, the traditional SEIR model is not suitable for fine-grained modeling and not for all domains. On the other hand, agent-based approaches model the behavior of each individual agent and the interaction between agents. ABM can be used to study the system at different abstraction levels and represents an optimal choice for fine-grained simulations [5, 6].

In this paper, we propose a distributed framework (ActoDemic) that aims to facilitate the design and implementation of spreading models using Agent-Based Modeling and Simulation techniques (ABMS) for large scale scenarios. We aim to develop a general and task-independent framework. Hence, our system tries to satisfy different requirements: it has to be suitable for different domains, collaborators and computing facilities. Each agent represents an entity that is involved in some interactions in each simulation epoch, depending on its own customizable properties. Executing millions of concurrent agents could represent a bottleneck for ABMS. In order to solve this issue, we implemented the software agents as concurrent actors exploiting the ActoDeS Framework. The actors have their own behavior and change it by processing asynchronous messages received from the other actors. Finally, as a use-case to test and validate our software architecture, we present a simulation of the COVID-19 outbreaks in Italy during the early-stage of the pandemic. We model about 10 millions of agents, interacting reciprocally using a social model that we have also implemented as a custom property of ActoDemic.

Woodstock'20: Symposium on the irreproducible science, June 01–05, 2020, Woodstock, NY

mattia.pellegrino@unipr.it (M. Pellegrino); gianfranco.lombardo@unipr.it (G. Lombardo); monica.mordonini@unipr.it (M. Mordonini); michele.tomaiuolo@unipr.it (M. Tomaiuolo); stefano.cagnoni@unipr.it (S. Cagnoni); agostino.poggi@unipr.it (A. Poggi)

CEUR Workshop Proceedings (CEUR-WS.org)

To achieve such a result we have exploited the High Performance Computing facility of the University of Parma, to scale and distribute the computational load over the available resources.

2. ActoDeS Framework

ActoDeS is a software framework that simplifies the development of concurrent and distributed systems and ensures an efficient execution of applications [7]. ActoDeS is implemented in Java and takes advantage of some implementation solutions used in JADE [8], [9] [10], and in CODE [11]. ActoDeS has been mainly used for the development of applications in the areas of agent-based modeling and simulation [12], [13], evolutionary computation [14] and data analysis [15], [16], [17], [18], [19], [20], [21].

ActoDeS offers a layered architecture made up of a run-time and an application layer. The run-time layer provides the software components that implement the middle-ware infrastructures that support the development of standalone and distributed applications. The application layer provides the software components that an application developer needs to extend or directly use for implementing the specific actors of an application.

In particular, an actor is an autonomous and concurrent object, characterized by a state and a behavior, that exhibits the ability to interact with other actors through the exchange of asynchronous messages [22]. The communication between the actors is buffered: the incoming messages are stored in a mailbox until the actor is ready to process them; moreover, an actor can set a timeout for waiting for a new message and can then execute specific actions if the timeout is reached. Moreover, after the analysis of its incoming messages, an actor can send more messages to itself or to others, create new actors, update its state, change its behaviors and, finally, terminate its own execution. Each behavior can define a policy for handling incoming messages, through handlers called "cases". Each case can only process messages corresponding to a specific pattern. Therefore, if an unexpected message arrives, then the actor mailbox maintains it until another behavior is able to process it.

As introduced above, ActoDeS can be used for developing distributed applications. In fact, depending on the complexity of the application and on the availability of computing and communication resources, an application can involve one or more computational nodes. In ActoDeS, each computational node maintains an actor space that acts as a "container" for a subset of the actors of the application and provides them with the services necessary for their execution. In particular, an actor-space contains a set of actors (application actors) that perform the tasks specific to the current application and two special actors called executor and service provider. The executor manages the concurrent execution of the actors of the actor space. The service provider enables the actors of an application to perform new kinds of actions (e.g., to broadcast a message or to move from an actor space to another one).

3. ActoDemic

ActoDemic aims to facilitate the design and development of spreading phenomena in large-scale scenarios. The base unit is represented by actors that, depending on the target application, represent the entities of the system to model and simulate. If a fine-grained detail is required by the simulation, it is clear that a large number of concurrent actors is required too. Thus, we have used ActoDeS as a backbone to support concurrent agents, whose computational load can be distributed over several nodes. Moreover, since ActoDes is a Java-based framework, ActoDemic is able to support different operating systems and to enable fast development and prototyping, thanks to Java's built-in features such as automated serialization and extensive libraries. ActoDemic defines at least one actor space in each computational node involved in the simulation. Indeed, one of the major problems to deal with, when designing a large scale ABMS, is that the entire set of agents may not fit in a single cluster node. When a spreading phenomenon has to be modeled, the developer should define the behavior of the entities, the way they interact, and, finally, the characteristics of the spreading phenomenon. To support a wide range of spreading phenomena, ActoDemic provides a base version of the actors that can be customized by defining new behaviors in the form of new Java classes. The base actor to be used for modeling the system entities is called Base Spreader (BS). For example, if we want to model the spreading of a pathogen among people, every person can be modeled and implemented as a BS. A Base Spreader has a set of default attributes that can be enabled, configured and modified to best suit the model that is going to be realized:

- **Identification number**: Unique id that discriminates each individual.
- **Belonging to a community or cluster of entities**: As a general point in a spreading simulation, we might want to distinguish entities in different clusters and give a higher bias to the interactions among the cluster and lower outside. For example, we might want to divide entities by geographical position or define a social community. If this aspect is not required, the entities can be part of a whole cluster of entities.
- **Interaction level**: Different interaction ratios can be defined among the entities to model more

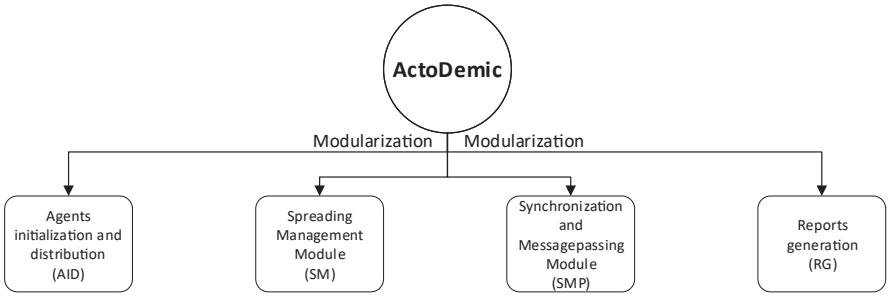

Figure 1: Framework modules

complex and fine-grained scenarios;
- **Current phase**: An indicator that specifies in which contagion phase the entity is;
- **Spreading reducer**: A damper that can reduce the spreading. For example, we might want to model a vaccination, a protective device or the use of an Anti-malware in a computer network.

3.1. A Modular Software architecture

ActoDemic emphasizes software modularity for all the aspects related to the execution and management of the simulation. Moreover, a modular structure can better adapt to any proposed epidemic model and be easier to configure. The modular architecture is presented in Figure 1. The architecture is built around four modules: Agents initialization and distribution module (AID), Spreading Management (SM), Synchronization and Message passing (SMP) and, finally, a utility to generate reports and evaluate the simulation (RG).

3.1.1. Agents initialization and distribution module

The AID module enables the distribution of the simulation over different nodes. It manages the initialization of all the ActoDeS entities to support the actors and the exchange of the messages. As a first step, AID initializes the required actor-spaces. Every created actor-space owns a thread and shares it with the agents that live in it. Hence, every individual is a passive actor and shares its thread with the other actors in the same space. Moreover, AID creates the base spreaders and distributes them over the nodes, according to different criteria that the developer can tune to match the system requirements:

a. Partitioning entities according to their cluster or community, if more than the default one are defined;
b. Splitting the population in equal-size subsets, depending on the number of actor spaces involved in the simulation.

In order to manage the base spreaders, AID module initializes the ActoDeS schedulers and managers in each actor-space. In particular, each manager creates the subset of agents for its computational node and synchronizes the simulation execution on that node with the others. Moreover, the last created manager assumes the role of "Master". The default algorithm 1 distributes the whole population of actors over the N available actor spaces according to the identification number of each actor. Every subset includes, generically, the agents that go from $(n-k)\cdot p-1$ to $(n-k+1)\cdot p-1$, where n is the number of partitions, k identifies the actual partition and $p = \frac{population}{N}$ is a constant.

Algorithm 1 Pseudo-code for distribute the whole agents' set across N actor-spaces

```
1:  𝒩 ← GetTotalActorSpaceNumbers()
2:  ℛ ← GetAllReferences()
3:  function BuildPopulation(𝒩, ℛ)
4:      master ← amITheMaster()
5:      Begin ← ∅
6:      End ← ∅
7:      if master is True then
8:          for (i = 0; i < 𝒩; i++) do
9:              𝒮 ← ((𝒩 − i) · p − 1)
10:             ℰ ← ((𝒩 − i + 1) · p − 1)
11:             SendMessage(ℛ[i], (𝒮, ℰ))
12:         end for
13:     end if
14:     loop
15:                 ▷ Wait a message from the master Actor-space
16:     end loop
17:     Begin, End ← ProcessMessageFromMaster()
18:     CreatePopulation(Begin, End)
19: end function
```

where:

- $\mathcal{N}$ is the number of total Actor-Spaces that we

want to create;
- $\mathscr{R}$ is a set that contains all the Actor-Space references (unique system-wide id that we need to reach an actor and communicate with it);
- *master* is a Boolean value that specifies whether an Actor-space acts as a master or not;
- *SendMessage*() is a function that communicates to an Actor-space which population subset it has to manage.
- *Begin*, *End* define the range of the actors to be created and managed ;
- *CreatePopulation*() divides the population into subsets

Algorithm 2 Pseudo-code to distribute X actor-space across N computational nodes

```
1: 𝒩 ← GetNodesNumber()
2: 𝒯 ← GetTaskPerNodesNumber()
3: function SpreadActorSpaces(𝒩, 𝒯)
4:     N_JOB ← 𝒩 · 𝒯
5:     BrokerIp ← ∅
6:     for (i = 0; i < N_JOB; i++) do
7:
8:         if i == 0 then
9:             BrokerIp ← SetBrokerIp()
10:            LaunchActoDemic(broker, BrokerIp)
11:        else if i == (N_JOB − 1) then
12:            LaunchActoDemic(initiator, BrokerIp)
13:        else
14:            LaunchActoDemic(node, BrokerIp)
15:        end if
16:    end for  ▷ The For cycle is managed by MPI on SLURM
17: end function
```

where:

- $\mathscr{N}$: Number of computational nodes;
- $\mathscr{T}$: Number of tasks using a single CPU on every node;
- *N_JOB* Number of actor-spaces that will be created;
- *LaunchActoDemic*() is the function that launches an ActoDemic instance and it takes two arguments.

We can distinguish three entities that play a fundamental role in ActoDemic and are inherited from ActoDeS: the Broker, the generic node and the Initiator. The Broker receives messages from producers and sends messages to consumers. The Initiator acts like the master node and coordinates the agent creation process across the various actor-spaces. AID provides also a sub-module that enables the use of ActoDemic with the Slurm Workload Manager (SLURM) and the MPI protocol [23]. SLURM is an open source, fault-tolerant, and highly scalable cluster management and job scheduling system for large and small Linux clusters. The MPI protocol is useful to distribute and manage the actor-spaces across the computational nodes. These capabilities enable the use of ActoDemic in High Performance Computing scenarios where SLURM and MPI are often a standard. An explanation of this sub-module is given by algorithm 2.

3.1.2. Spreading Management Module

ActoDemic exploits some concepts from Network Science to model the interactions among the system entities. The Spreading Management module (SM) defines the way the interactions should occur. Every actor-space manages its own actors, as well as their own different interactions. It assumes that a generic actor represents a node in a generic graph and the outgoing and incoming links represent, respectively, the entities with which the node interacts and vice versa. Graph theory allows us to study the distribution of interactions, by studying the node degree distribution. The distribution of interactions is a crucial factor that affects the spreading of an epidemic phenomenon. ActoDemic provides a simple way for tuning this distribution when defining the interactions. It allows the user to choose a distribution and set it as default. The available distributions that the framework provides are: power-law, log-normal, exponential, Gaussian and Poisson. The SM module also allows to partition the entire agents' set into different clusters and give a bias to the interactions such that they are higher inside the cluster and lower outside.

When two individuals interact with each other, a procedure that simulates an infection is started. We implemented an epidemic diffusion model starting from the compartments of the SEIR mathematical model (Susceptible-Exposed-Infective-Recovered)[4]. SEIR is based on a series of dynamic differential equations that consider the amount of the population subject to contagion, the trend over time of the number of individuals who recover after infection, and of the casualties. A limit of this model is its coarse-grain nature with respect to individual behaviors. Moreover, the SM module provides a method to control the contagion power through a variable called "Transmission Probability" (TP). This variable can take values between 0 and 1, to express the probability for a contagion operation between two entities to be successful.

However, the compartments are fully customizable. A single compartment can be disabled, changing in this way the development of the infection. Moreover, intermediate compartments can also be added to add steps to

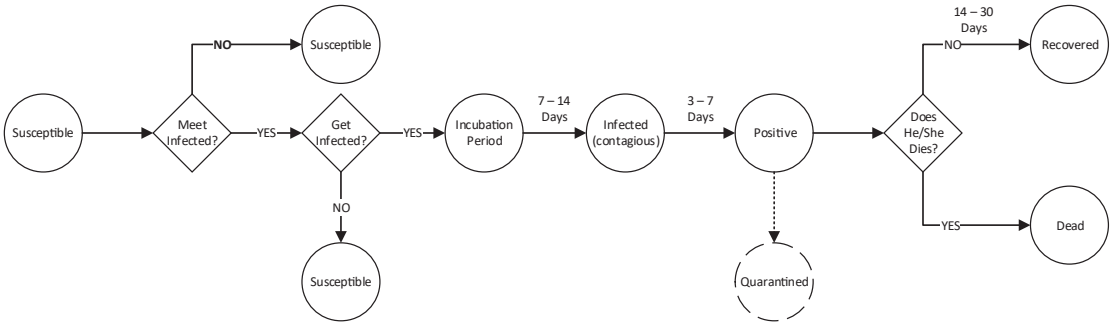

Figure 2: Example of a generic infection Cycle

the simulation process and even the duration between the various phases can be tuned. ActoDemic also makes it possible to enable and specify the behavior in case of a re-infection event after a recovery.

An example of a full infection cycle is shown in Figure 2. This scheme represents the COVID-19 infection cycle. We shall deepen this aspect in section 4.

Finally, the SM module is responsible for the initialization and the implementation of the damper that is able to scale the transmission probability.

3.1.3. Synchronization and Message passing Module

The SMP module manages the message exchange and time synchronization.

ActoDemic uses a very simple scheduler called "*CycleScheduler*" provided by ActoDeS. This tool can be used in a wide variety of applications; more specifically, also in ABMS applications. Furthermore, this scheduler manages the passive actors within its actor-space and cyclically repeats the same actions, until the simulation ends:

1. Send a "step" message to all agents and increment the "step" value; this operation triggers the transition from one epoch to the next.
2. Perform an execution step of all agents.

Every actor-space is an actor and an actor cannot access the internal state of another actor. Therefore, if one or more Base Spreader that belongs to an actor-space interact with some other Base Spreaders of a different actor-space then we must inform both actor-spaces of that interaction. This can only be done with a messages exchange system.

Another typical property of the actor model is that the exchange of messages is asynchronous. This property creates a synchronization problem, because an actor-space could move to the next epoch without receiving all the necessary information from all the other actor-spaces. Moreover, an actor-space needs to retrieve the information from all other actor-spaces before triggering the transaction of its actors to the next epoch. SMP solves this problem by creating a mechanism that acts as a "barrier" and does not allow all the actor-spaces to switch to the next epoch if the message exchange between all the actor-spaces has not been completed.

Information routing and management may differ according to the target model the user wants to implement. However, there is a basic scheme that we report here. Some information is sent across every partition type, while some information can only be sent or received by the master. The content of a generic message can be schematized in this way:

1. Information sent to all partitions:
   - Information about Base Spreaders' interactions: it informs the actor-space of the interaction between two or more of its Base Spreaders;
   - Base Spreaders that have to change their status: it specifies which Base Spreaders have to change their status due a trigger situation;
   - Base Spreaders that have been closed: actors that have to end their cycle;
   - Statistical Data: data needed to generate reports;
   - Synchronization message: message that acts like a "barrier". When an actor-space receives all the synchronization messages from all the actor-spaces involved in the simulation, then it can move to the next epoch;
   - End signal: a message that allows the end of simulation process;
2. Information sent to master only:
   - Partial summary report: data needed to generate the final reports;
3. Information sent only by the master:

– Base spreaders that have to shutdown themselves: the master decides who should end their execution, based on the user's criteria.

3.1.4. Reports generation module

The RG module enables the generation of reports during the simulation process. The user can decide whether to enable, or not, the generation of reports. Moreover, the user can decide how often a report is to be generated.

In general, at the end of every epoch every actor-space generates a report that regards only its partition and its actors: these are called "intermediate reports". Furthermore, the intermediate reports are also used to support a "Save&Load" functionality. Thanks to this mechanism it is possible to restart the simulation process from a specific epoch. However, in order to do so, reports need to include certain information for each Base Spreader within the actor-space:

- The Identification Number
- The belonging to any type of social cluster
- The interaction level
- The epidemic phase it is currently going through
- How much the epidemic phases must last
- If the Base Spreader has an active Spreading reducer
- A set that maintains the information about the past interactions with the other Base Spreaders

Finally, at the end of the simulation, the master node generates a final report that summarizes the most important and relevant information. In particular, it generates a summary for each simulation epoch showing how many people belong to each specific epidemic compartment.

The RG module is also the one responsible for the initial configuration. Thanks to this, ActoDemic supports an external configuration file, in which it is possible to customize and specify all the properties explained in the article.

4. Use-Case: COVID-19 spreading in Lombardy

4.1. Modelling

After building and modeling the framework in all its features, we needed a real use case to test it. Therefore, to validate ActoDemic, we focused our attention on COVID-19 spreading. The social interactions, which are the major cause of COVID-19 spreading, can be easily modeled by properly setting the parameters on which ActoDemic depends.

The proposed model simulates about ten million independent agents that reproduce the social behaviour of the inhabitants of Lombardy, a region in northern Italy. We have decided to simulate the COVID-19 epidemic spread in Lombardy for several reasons. It was the first region, in Italy, to be affected by the virus and currently it is the first for number of infections; therefore, it is the Italian region that offers the most abundant statistical data, that can be used to compare and validate our model. This represents an interesting use-case to test the robustness of our methodology.

To make ActoDemic suitable for our use-case, we have customized every framework module. Every person involved in the simulation process is represented by a Base Spreader and every epoch represents a generic day in a real-life situation.

The ten million people living in Lombardy are subjected to a partition by districts. Accordingly, we have divided the entire population geographically respecting the number of Lombardy provinces and the distribution of their inhabitants, creating several social communities. We have assigned an additional feature to each Base Spreader: its age. The alter parameter is crucial, because it adds information in our design and can be used to better model the social interactions. We have also respected the Lombardy age distribution [24]. To model social interactions even more efficiently, we have defined three interaction ratios: high, medium and low. These have been introduced to increase or decrease the average number of the subjects' daily contacts with other people, based on their age. To estimate the average number of a Base Spreader's contacts, we have used data from the Italian National Institute of Health [25] and [26].

To correctly model the pathogen spread, we have customized the SM module. We have used the base SEIR model, adding two extra compartments: Positive and Quarantine. These phases are typical in the COVID-19 infection cycle. Initially, all people are in the susceptibility stage. In this compartment, every subject can be infected by another one who is contagious. An individual who is infected moves from a susceptibility phase to an incubation phase and remains in this stage for a certain time, before moving into an infection stage. A subject in this condition can infect other people. When this phase ends, the person becomes positive. After a certain time, a positive will either heal or die. There is no death probability, but deaths follow the real death curve trend in Lombardy. When an individual heals, it cannot be infected any more. In particular, the incubation phase lasts from 7 to 14 days, the infectious phase from 3 to 7 days, and the positive phase from 14 to 30 days [27] [28]. Figure 2 shows a diagram that represents the infection cycle.

To model COVID-19 compartments we have used preliminary data collected by [27], [28] and the age suscep-

tibility to COVID-19 virus. Moreover, the SM module supports a damper to mitigate the contagion spread; we have used this particular feature to simulate the adoption of protective devices. We have collected data about the percentage of the population that was using protective devices [29] and their effectiveness [30]. The last property expected by the SM module concerns the distribution of social interactions. For this reason, we have evaluated various hypotheses, but, in the end, we have decided to focus our studies on a power-law distribution. We assume a common hypothesis in network science that asserts that social networks commonly have a power-law distribution with an exponent between 2 and 3, also known as the scale-free property [2]. Contact networks are usually modeled with a power-law distribution [31]. We have exploited these interaction ratios to also model the lockdown policy and the contagion containment strategy adopted in Italy in the first pandemic months. Modeling the Italian lockdown has required different pieces of information about the set of "essential workers" [32]. Remember that the only people who were not subjected to limitations were those who worked in the so-called "essential sectors".

The information that the actor-spaces exchange with each other need to be modified to make the simulator work. Accordingly, we have customized the SMP module to modify the message contents:

1. Information sent to all partitions:
   - Information about people's meeting
   - People who have to change their infection phase to "Incubated" due an infection
   - Number of people expected to die in that partition
   - Statistical Data
   - Synchronization message
   - End signal
2. Information sent to master only:
   - Currently positive people
   - Currently infected people
   - Partial summary report
3. Information sent only by the master:
   - Total people expected to die

The last module we have customized is the RG module. In our use-case, we generate an intermediate report at the end of every epoch. Obviously, the information included in each intermediate report has been customized ad-hoc for our use-case. It contains the following information:

- Id
- Age
- Province of Residence
- Essential worker (Boolean value)
- Mask wearing (Boolean value)
- Incubation Days period
- Infection Days period
- Positive Days period
- Usual Contacts' List

At the end of the simulation process, a summary report is generated, containing the following information:

- Epoch Number
- Positive People
- Infected People
- Susceptible People
- Recovered People
- Dead People

Once every customization has been realized, the framework is ready to be tested and to reproduce some results. In the next section, we report the results we have obtained in our use-case.

4.2. Experimentation

In order to evaluate our simulator, we have considered two different COVID-19 outbreaks in Lombardy (Italy). In particular, we are interested in modeling the first wave form January to April 2020 and the second one between August and December 2020. The combination of these two waves has been taken in consideration to validate our simulator and for modelling its parameters.

When the simulation process starts, all agents are in the susceptibility status, as previously reported. In this way, no one can start a hypothetical contagion. Hence, at the beginning of the simulation we have to choose randomly which Base Spreader will start directly from an incubation phase. In addition, to do this, we respect the number of positives people between 20 and 29 February in Lombardy on a provincial basis.

We have calibrated the transmission probability with a random-search over the probability space. We have estimated this value trying to chase up the contagion curve until the pre-lockdown date, March 8th, 2020. The lockdown is an Italian policy to prevent the contagion spread that implies the closure of non-essential activities, social distancing and some rules for limiting the movement of people. The value satisfying these hypothesis is 0.3. A summary histogram is shown in figure 3.

4.3. Results

In this section, we present the results that we have obtained simulating different scenarios and considering each time an average of 10 different runs, since the entire simulation process is stochastic in most of its steps. For each case, we have measured the simulation quality using the Pearson correlation and the Root-mean-square

error (RMSE) between simulated data and real data from the Italian government [33]. The Person correlation expresses any linear relationship between two statistical variables. This value ranges from −1 to 1, where 1 corresponds to a strong positive linear correlation and −1 corresponds to a strong negative linear correlation. In our case, it explains how much the trend of the simulated contagion curve resembles the real one. The Root-mean-square error is computed between the predicted values and the real data. Figure 4 shows the results of the simulation in the early-stage of the pandemic between January and April 2020 with the COVID-19 Transmission probability (CTP) equal to 0.3. The blue curve represents the real contagion data, while the red curve represents the simulated data. Pearson correlation and RMSE referred to Figure 4 until April 30th are equal to 0.992 for Pearson correlation and 38818, respectively. The number of the total positives obtained using the simulator in that date, exceeds by about 53,000 units the number of actual positives (Figure 4).

The results in Figure 4 seem to support our thesis, but it is widely conceivable that the real data measured over that period were underestimated.

A comparison with the *ISTAT*'s serological investigation reveals a very different situation [34]. This study shows that, on July 15th, the actual number of COVID-19 cases in Lombardy was about 7.92 time greater than the data form COVID-19 tests. In addition, the study shows that about 7.5% of the Lombard population had developed antibodies for the COVID-19. The population of Lombardy is about 10,060,000 people, 7.5% of which is therefore equivalent to about 754,500. This strengthens the hypothesis that the spring data were underestimated. Assuming that this ratio is constant over time, we retro-projected this data and observed how many positive people could be estimated.

Therefore, we have used the data of the second wave (August to December 2020) to perform a fine tuning procedure. We have used the previously obtained CTP to verify whether the simulated data properly followed the real data generated by the second wave. The values ob-

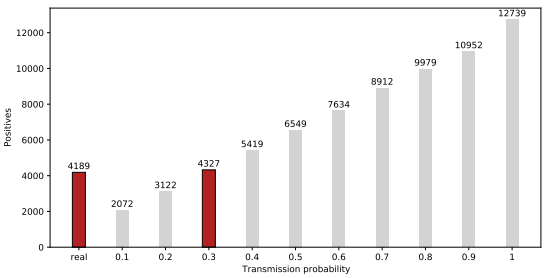

Figure 3: Positive people on 8th March 2020 with different transmission rates

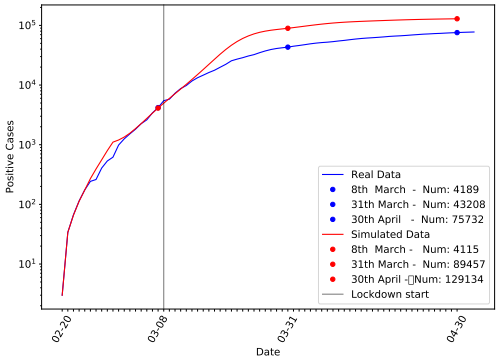

Figure 4: Simulation results with real data with starting assumptions (CTP=0.3) - Incremental positives representation

tained greatly underestimated the actual data. For this case, the Pearson correlation is 0.988 and the RMSE is 27415.

Due to the large estimation error, we have decided to estimate again the transmission probability parameter, using the data of the second wave. We have searched for a value that follows the contagion curve correctly. The best value that satisfies our hypothesis is 0.53. In this final case the Pearson correlation is 0.996 while the RMSE is 6405.

4.4. Final Projections

In the light of the previous considerations, we have decided to simulate again the first wave with the new CTP value. Moreover, we have added also the data obtained from the comparison with the national screening activity. The whole process is shown in Figure 5. The blue curve represents the actual data, the green curve represents the serological data projection on the real data and the red curve represents the simulated data with transmission probability equal to 0.53. The second estimation of the COVID-19 Transmission Probability using the autumn data is confirmed as a better choice to validate our model. The difference with the serological data projection on April 30th, corresponding to the last simulation day, is only 83,369 units. Seroprevalence analysis is much more reliable than the data collected during the months of March and April, because it also takes into account asymptomatic people, which is a very crucial factor. Considering the cumulative curves in Figure 5, the Pearson correlation is 0.996 and the RMSE is 249,529. In the comparison between simulated and serological data Pearson correlation is again 0.996 while the RMSE is 56,009. Matching data confirms the validity of our hypothesis and of our simulation model.

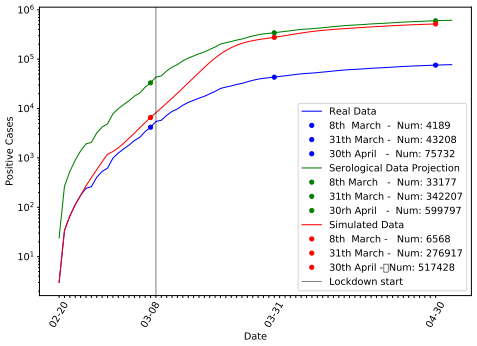

Figure 5: Simulation results with real data with transmission probability equal to 0.53 - Spring case - Incremental positives representation

5. Conclusion

In this paper, we have presented a framework that aims to combine a fine-grained spreading model with a large-scale scenario. This result is achieved by exploiting an efficient multi-agent system that can be run on a distributed architecture. The proposed framework has been designed with a modular structure to be user-friendly and easy to customize. We have validated our framework simulating the outbreaks of COVID-19 in Lombardy (Italy) in 2020. The results have proven that our framework is able to simulate and accurately reproduce a spreading phenomenon. Future works are related to improving the framework by adding more features and modules to make the framework faster and more customizable.

References

[1] P. C. Silva, P. V. Batista, H. S. Lima, M. A. Alves, F. G. Guimarães, R. C. Silva, Covid-abs: An agent-based model of covid-19 epidemic to simulate health and economic effects of social distancing interventions, Chaos, Solitons & Fractals 139 (2020) 110088.

[2] A.-L. Barabási, Network Science, Cambridge University Press, 2016.

[3] G. Lombardo, P. Fornacciari, M. Mordonini, L. Sani, M. Tomaiuolo, A combined approach for the analysis of support groups on facebook-the case of patients of hidradenitis suppurativa, Multimedia Tools and Applications 78 (2019) 3321–3339.

[4] H. W. Hethcote, The mathematics of infectious diseases, SIAM review 42 (2000) 599–653.

[5] G. P. Figueredo, P.-O. Siebers, M. R. Owen, J. Reps, U. Aickelin, Comparing stochastic differential equations and agent-based modelling and simulation for early-stage cancer, PloS one 9 (2014) e95150.

[6] H. V. D. Parunak, R. Savit, R. L. Riolo, Agent-based modeling vs. equation-based modeling: A case study and users' guide, in: International Workshop on Multi-Agent Systems and Agent-Based Simulation, Springer, 1998, pp. 10–25.

[7] F. Bergenti, E. Iotti, A. Poggi, M. Tomaiuolo, Concurrent and distributed applications with actodes, in: MATEC Web of Conferences, volume 76, EDP Sciences, 2016, p. 04043.

[8] A. Poggi, M. Tomaiuolo, P. Turci, Extending jade for agent grid applications, in: 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, IEEE, 2004, pp. 352–357.

[9] A. Negri, A. Poggi, M. Tomaiuolo, P. Turci, Dynamic grid tasks composition and distribution through agents, Concurrency and Computation: Practice and Experience 18 (2006) 875–885.

[10] F. Bergenti, G. Caire, S. Monica, A. Poggi, The first twenty years of agent-based software development with jade (2020).

[11] F. Bergenti, A. Poggi, M. Tomaiuolo, An actor based software framework for scalable applications, Lecture Notes in Computer Science 8729 (2014) 26–35.

[12] A. Poggi, Agent based modeling and simulation with actomos., in: WOA, 2015, pp. 91–96.

[13] G. Angiani, P. Fornacciari, G. Lombardo, A. Poggi, M. Tomaiuolo, Actors based agent modelling and simulation, in: International Conference on Practical Applications of Agents and Multi-Agent Systems, Springer, 2018, pp. 443–455.

[14] G. Petrosino, F. Bergenti, G. Lombardo, M. Mordonini, A. Poggi, M. Tomaiuolo, S. Cagnoni, Island model in actodata: an actor-based implementation of a classical distributed evolutionary computation paradigm, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2021, pp. 1801–1808.

[15] F. Bergenti, E. Franchi, A. Poggi, Agent-based social networks for enterprise collaboration, in: 2011 IEEE 20th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, IEEE, 2011, pp. 25–28.

[16] G. Lombardo, P. Fornacciari, M. Mordonini, M. Tomaiuolo, A. Poggi, A multi-agent architecture for data analysis, Future Internet 11 (2019) 49.

[17] E. Franchi, A. Poggi, M. Tomaiuolo, Blogracy: A peer-to-peer social network, in: Censorship, Surveillance, and Privacy: Concepts, Methodologies, Tools, and Applications, IGI global, 2019, pp. 675–696.

[18] E. Franchi, A. Poggi, M. Tomaiuolo, Social media for online collaboration in firms and organizations, in: Information Diffusion Management and Knowledge Sharing: Breakthroughs in Research and Practice, IGI Global, 2020, pp. 473–489.

[19] M. Tomaiuolo, G. Lombardo, M. Mordonini, S. Cagnoni, A. Poggi, A survey on troll detection, Future internet 12 (2020) 31.

[20] G. Lombardo, A. Poggi, Actornode2vec: An actor-based solution for node embedding over large networks, Intelligenza Artificiale 14 (2020) 77–88.

[21] G. Lombardo, A. Poggi, A scalable and distributed actor-based version of the node2vec algorithm., in: WOA, 2019, pp. 134–141.

[22] G. A. Agha, Actors: A model of concurrent computation in distributed systems., Technical Report, Massachusetts Inst of Tech Cambridge Artificial Intelligence Lab, 1985.

[23] L. Clarke, I. Glendinning, R. Hempel, The MPI message passing interface standard, in: Programming Environments for Massively Parallel Distributed Systems, Birkhäuser Basel, 1994, pp. 213–218. URL: https://doi.org/10.1007/978-3-0348-8534-8_21. doi:10.1007/978-3-0348-8534-8_21.

[24] Popolazione per età, sesso e stato civile 2020, https://www.tuttitalia.it/lombardia/statistiche/popolazione-eta-sesso-stato-civile-2020/, 2020.

[25] C. T. Scientifico, Valutazione di politiche di riapertura utilizzando contatti sociali e rischio di esposizione professionale (2020).

[26] J. Mossong, N. Hens, M. Jit, P. Beutels, K. Auranen, R. Mikolajczyk, M. Massari, S. Salmaso, G. S. Tomba, J. Wallinga, et al., Social contacts and mixing patterns relevant to the spread of infectious diseases, PLoS Med 5 (2008) e74.

[27] Covid-19: nuovo studio sull'incubazione del sars-cov-2, https://www.osservatoriomalattierare.it/news/ricerca-scientifica/15771-covid-19-nuovo-studio-sull-incubazione-del-sars-cov-2, 2020.

[28] Epidemia covid-19, https://www.epicentro.iss.it/coronavirus/bollettino/Bollettino-sorveglianza-integrata-COVID-19_12-marzo-2020.pdf, 2020.

[29] D. Perrotta, A. Grow, F. Rampazzo, J. Cimentada, E. D. Fava, S. Gil-Clavel, E. Zagheni, Behaviours and attitudes in response to the COVID-19 pandemic: Insights from a cross-national facebook survey (2020). URL: https://doi.org/10.1101/2020.05.09.20096388. doi:10.1101/2020.05.09.20096388.

[30] S. E. Eikenberry, M. Mancuso, E. Iboi, T. Phan, K. Eikenberry, Y. Kuang, E. Kostelich, A. B. Gumel, To mask or not to mask: Modeling the potential for face mask use by the general public to curtail the COVID-19 pandemic, Infectious Disease Modelling 5 (2020) 293–308. URL: https://doi.org/10.1016/j.idm.2020.04.001. doi:10.1016/j.idm.2020.04.001.

[31] S. Eubank, H. Guclu, V. A. Kumar, M. V. Marathe, A. Srinivasan, Z. Toroczkai, N. Wang, Modelling disease outbreaks in realistic urban social networks, Nature 429 (2004) 180–184.

[32] Coronavirus, la demografia del lockdown: ecco l'identikit di chi lavora e di chi sta a casa, https://www.ilfattoquotidiano.it/2020/04/13/coronavirus-la-demografia-del-lockdown-ecco-lidentikit-di-\chi-lavora-e-di-chi-sta-a-casa/5763773/, 2020.

[33] Github protezione civile, https://github.com/pcm-dpc/COVID-19, Real Time.

[34] Studio della sieroprevalenza in italia, http://www.salute.gov.it/portale/news/p3_2_1_1_1.jsp?lingua=italiano&menu=notizie&p=dalministero&id=4998, 2020.