Towards efficient large scale epidemiological simulations in EpiGraph

Gonzalo Martín[a,*], David E. Singh[a], Maria-Cristina Marinescu[b], Jesús Carretero[a]

[a] *Universidad Carlos III de Madrid, Computer Science Department, Leganés, Madrid 28911, Spain*
[b] *Barcelona Supercomputing Center, Computer Applications in Science & Engineering, Barcelona 08034, Spain*

Abstract

The work we present in this paper focuses on understanding the propagation of flu-like infectious outbreaks between geographically distant regions due to the movement of people outside their base location. Our approach incorporates geographic location and a transportation model into our existing region-based, closed-world EpiGraph simulator to model a more realistic movement of the virus between different geographic areas. This paper describes the MPI-based implementation of this simulator, including several optimization techniques such as a novel approach for mapping processes onto available processing elements based on the temporal distribution of process loads. We present an extensive evaluation of EpiGraph in terms of its ability to simulate large-scale scenarios, as well as from a performance perspective.
Keywords:

*Corresponding author
Email addresses: gmcruz@arcos.inf.uc3m.es (Gonzalo Martín), desingh@arcos.inf.uc3m.es (David E. Singh), maria.marinescu@bsc.es (Maria-Cristina Marinescu), jcarrete@arcos.inf.uc3m.es (Jesús Carretero)

Simulation; computational epidemiology; parallel algorithms; MPI implementation; resource allocation

1. Introduction

We live in a world that is rapidly becoming predominantly urban and highly connected—virtually and physically. Transportation networks whose original role was simply to connect isolated regions are now serving a new purpose; to connect them faster and more reliably. The emergence of a mobility network that is more strongly connected, and in which more people migrate towards the nodes from rural areas, creates a big vulnerability to contagious threats. It is not just a matter of the area of dissemination, but also of propagation speed. Recent decades show the occurrence of global influenza pandemics originated in Asia (1957, A/H3N2 strain) and Latin America (2009, A/H1N1 strain) as examples of outbreaks related to the movement of people between continents [1, 2]. Understanding the patterns that viruses, such as influenza, follow when they propagate among the population of widely-spread geographic regions is fundamental for an agile response of public health authorities.

Our epidemiological simulator (EpiGraph) can predict the evolution of infections over short to medium time frames within single urban areas and was validated against the data from the 2004-2005 New York State Department of Health Report [3]. It is implemented as a scalable, fully distributed application based on MPI. The simulator we are describing in this paper is an extension of the original EpiGraph that overcomes the limitations coming from the assumption of a closed world, where new individuals could not be

introduced. To do this we capture the introduction of a virus in a population at different times and by individuals that travel between urban regions—and which, voluntarily or not, get in contact with the local population. Modeling this type of contacts is crucial to understand the effect that travel and commute have on the evolution of epidemics at a global level.

The original contributions of this paper are (1) an extension of EpiGraph which enables an efficient simulation of virus propagation between arbitrarily far apart urban areas interconnected via transportation networks, and (2) a set of several performance optimization methods complete with a thorough analysis of their effect of EpiGraph's performance. We describe how we partition data efficiently to exploit data locality, how we optimize the communications, and finally we introduce a process-to-processor mapping technique which brings significant performance improvements. We also present an experimental evaluation of EpiGraph when simulating 92 urban regions in Spain consisting of 21,320,965 inhabitants. The results show that we can scale our simulations to run efficiently over large areas.

The rest of the paper is organized as follows. Section 2 discusses the related work. Section 3 summarizes the main features of EpiGraph, including the transportation model that captures the spatial transmission of flu-like infectious diseases between cities. Section 4 discusses the parallel implementation of EpiGraph and includes an efficient process-to-processor mapping algorithm. Results and performance evaluation are presented in Section 5. Section 6 concludes the paper.

2. Related work

Epidemiological simulators have become a powerful tool for understanding and predicting the dynamics of the propagation of infectious diseases. The dynamics of infectious diseases are usually modeled using stochastic or deterministic compartment models. This simplest such model for influenza is the classic *Susceptible-Infectious-Recovered* (SIR) epidemic model [4]. The SIR model considers that the population is divided into a set of groups according to the health state of the individual. Two approaches are most popular to simulate the spreading of the infection throughout a population: using a deterministic mathematical model based on differential equations, or using a social contact network-based model.

Simulators based on the deterministic approach [5, 6] use a set of differential equations to model the process by which individuals pass between the different stages of the infection. These models assume that the population is homogeneously mixed and the social contacts are highly structured—an unrealistic assumption. In reality, each individual has specific interaction patterns and the contacts between the individuals within a social contact network are unstructured. This makes the interconnection network be heterogeneous [7, 8].

The propagation dynamic of infectious diseases is tightly related to the structure and the characteristics of the network of connections between the individuals within a population [9, 10]. For this reason, epidemiological simulators based on social contact network models are becoming increasingly more popular. These simulators model the evolution of the epidemics as a stochastic process in which the movement of the individuals between the

infectious stages is driven by probabilistic variables.

Several works have studied the implications of the social contact network on the spreading of epidemics [11, 12]. One of their shortcomings is that they do not consider propagation at a global level, but rather consider a restricted number of urban regions. In case of an epidemic outbreak the results provided by the epidemiological simulator are critical during the first days, when the quality of decision making is vital. A simulator must therefore be able to efficiently handle highly detailed simulations and process huge amounts of data in a timely manner—which requires high computational power.

EpiFast [13] is an MPI-based simulator which implements an SIR-like model for simulating the evolution of epidemics in heterogeneous social contact networks. These contact networks are generated randomly and do not use either demographic nor geographic information. The parallel implementation of EpiFast is based on the master-slave model, which makes the communications more complex and reduces the scalability of the algorithm when executing on many processors.

FluTE [14] is an individual-based simulation model for Influenza. The hierarchical structure of communities within the population is based on data extracted from the census. The social contacts within each community are randomly generated as uniformly mixing groups. FluTE is able to simulate large-scale scenarios and implements a transportation model based on information extracted from the air traffic routes in the U.S. The complexity of their MPI-based parallel algorithm increases non-linearly with the community size, which reduces the performance of FluTE for large-scale scenarios.

EpiSimdemics [15] is an epidemiological simulator which implements an

efficient MPI-based parallel algorithm to simulate very large populations of up to 100 million people, although it requires massive computing resources to do so. The population is modeled based on demographic data extracted from the census. It does not consider a transportation model. In contrast, EpiGraph requires lower computing resources to simulate large-scale scenarios with a more sophisticated epidemic model and a more realistic social model which captures the transmission of an infection across different urban regions due to the movement of the population.

InFlusim [16] implements an extension of the SEIR compartmental model which includes hospitalization and home confinement. InFlusim is a deterministic model that does not take into account individual characteristics or spatial distribution of the population. The Centers for Disease Control and Prevention of United States offers a set of tools for pandemic simulations—Flu Preparedness [17]—which includes CommunityFlu. This tool performs simulations at individual level for influenza propagation over a community of 2,500 persons and it is able to model the effect of different interventions like vaccinations, school closings, and patient isolation. A similar approach can be found in [18], where stochastic models are used to evaluate the effects of school closings and reductions in contacts of ill persons by means of confinement. EpiGraph follows a similar approach to these two tools, but it has the advantage of considering larger communities modeled from actual social networks and connected by both long and short-range transportation.

The GLEaMviz tool [19] is a software that is able to simulate the spread of epidemics at large scale. GLEaMviz employs a stochastic SEIR compartmental model in combination with a metapopulation approach based on simulat-

ing a collection of groups interconnected by transportation. The simulation granularity is by groups of individuals with a resolution of approximately 25x25 kilometers. The geographical extension of each group is determined by applying Voronoi decomposition techniques and it is given a value based on census data. Epidemic propagation occurs differently within groups and between groups, inter-group dissemination happening due to travel using both short and long-range transportation.

STEM [20] is an open source project for modeling the spread of infectious diseases using a metapopulation approach. Its modular structure includes different disease models like seasonal Influenza [21], malaria [22], and dengue fever. In [21] STEM is used to simulate the seasonal influenza using the SIR(S) compartmental models with a seasonally modulated transmission coefficient. This tool is supported by an open community of contributors and considers spatially structured populations and transportation effects obtained from GIS data sets. STEM does not consider single individual characteristics but rather works with groups at the granularity of US counties. A similar approach is follower in GEM [23], which implements an extended SEIR stochastic model (including non-susceptible states) and simulates the effect of an influenza outbreak on the major cities in the world taking into account different travel restriction levels and vaccination policies. All these tools based on a metapopulation approach are able to perform large scale simulation but they do not model interactions between individuals. This makes it more difficult to evaluate the effectiveness of interventions like school closings, confinement of infected families in their homes, or selective vaccination policies based on single individual characteristics like age, gender, or occupa-

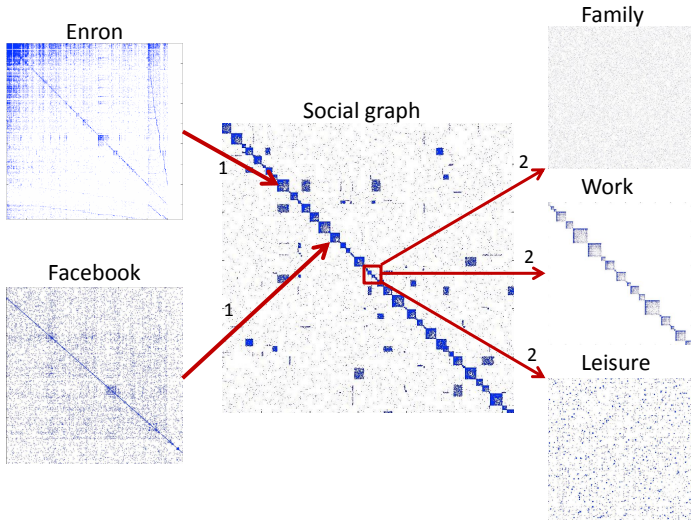


Figure 1: Enron, Facebook and social graphs displayed as matrix patterns.

tion. In contrast, EpiGraph considers individual characteristics, their related connections, and the traveling patterns between different regions.

3. EpiGraph structure

EpiGraph is designed as a scalable tool which simulates the propagation of influenza in scenarios that cover extended geographic areas. EpiGraph consists of three main components: the social model—based on the contact network of the individuals in a population—, the epidemic model, and the transportation model—which captures the movement of individuals between different regions—. The following sections describe in detail each one of these models, as well as the simulator implementation.

3.1. The social interconnection model

The social interconnection model is represented via an undirected connection graph that captures heterogeneity features at the level of both the

individual and each of his interactions. Each individual is represented as a node and has specific characteristics such as age, gender, race, and occupation. We represent the interactions with edges which capture a time-dependent interaction between two individuals. Social interaction patterns are modeled using real information extracted from on-line social networks. We use real demographic information obtained from the National Institute of Statistics of Spain [24] to represent the characteristics of the individuals and communities. The basic community is the *group*. A group is a collection of individuals connected by one of the following relationships: school-age children and students, workers, stay-home parents, and retired individuals.

We use the Albatross [25] algorithm to generate each group by sampling the graphs obtained from Facebook and the Enron email corpus. The graph extracted from the Enron database consists of 70,578 nodes and 312,620 edges and is used to model worker and retired groups. Facebook has 250,000 nodes and 3,239,137 edges and is used for school and stay-home groups. Figure 1 shows an example of these graphs as well as a fraction of the social graph. Arrows labeled 1 show two groups from the social graph that are generated by sampling Facebook and Enron graphs. Note that the sampling algorithm generates different group patterns and sizes every time that it is applied.

In addition to the interactions within a given group we represent interactions between members of the same family and between individuals of different groups. These reflect the fact that at different times individuals may interact with each other in different environments: during work, at home, during leisure time, or via spontaneous contacts. Additionally, we consider the changes in the time patterns during the weekends and holidays. For

instance, a given percentage of the companies (work connections) are not active on these days. We use the values of the social graph to code the type of connection. Each of these kinds of interactions is assigned to a specific daily time frame depending on the schedule for the main activity, leisure, and family time. Arrows labeled 2 in Figure 1 show the active edges for different times for a social graph portion. This approach allows us to store in the same connection graph different types of interactions which became active at different times of the day.

3.2. The epidemic model

The epidemic model is specific to the infectious agent under study, in our case, the Influenza virus. We start from the SEIR epidemic model [26], a variation of the classic SIR model [4] that takes into account an additional state (E, exposed) representing the latent phase. We extended the SEIR model to include additional states such as asymptomatic, dead, and hospitalized [27]. The hospitalized state is important when simulating realistic cases where this may be needed and the cost associated with this measure must be predicted. We consider that the infective period consists of three phases with different characteristics: (1) pre-symptomatic infection where individuals are infectious but symptoms are not yet present; (2) primary stage of symptomatic infection where symptoms are present and it is possible to initiate an antiviral therapy; (3) second stage of symptomatic infection where symptoms are present but viral therapy is no longer effective. A more detailed description of this model can be found in [3].

Figure 2 illustrates the epidemic model. It consists of two subgraphs. Let us consider the upper one: susceptible individuals (state S) who become

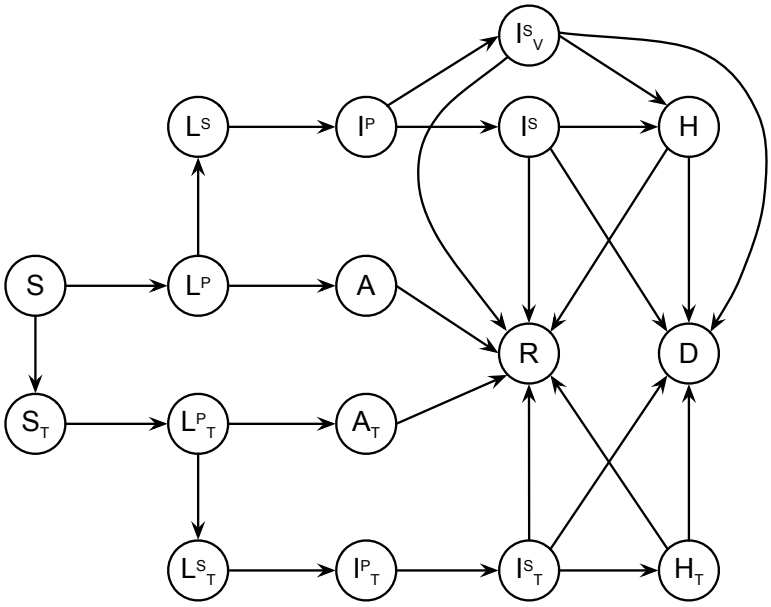


Figure 2: State diagram for the epidemic model.

infected incubate the infection in the primary latent state (L^P). In this state there are not symptoms or possibility of infecting the others. After that, a fraction of cases remains asymptomatic (A) and the rests of them go to a secondary latent state (L^S) where symptoms are still not present but it is not possible to became asymptomatic. What follows is the primary state of symptomatic infection (I^P), in which symptoms are present and, depending on the individual characteristics (age and risk group), a fraction of the clinically ill cases will seek medical care and initiate an antiviral therapy. Instead of using a fixed duration for the window of opportunity in which the antiviral therapy is effective, such specified in [28], we assume that every individual may have a slightly different one by using a probability distribution. To what extent the antiviral treatment will have an effect depends on the time within the window when an individual seeks medical care. If an individual

is treated with antivirals and the treatment has an effect then he moves to state I_V^S. Otherwise he remains in I^P and then passes to I^S in which the antiviral treatment is no longer effective. An infected individual can recover, get hospitalized, or die (states R, H and D). The lower part of the graph (involving the T-subscript states) reflects the case of vaccinated individuals in which the basic reproduction numbers are different from the non-vaccinated cases.

The time an individual spends in a given state is generated following a normal distribution based on the model parameters from the existing literature [29, 30, 31, 28]. Each individual will be assigned a different time, generated based on a normal distribution. Table 1 shows the basic reproduction numbers for the infective states and the parameters used in the normal distributions. The state transition algorithm is shown in Algorithm 1. For a given individual l, the $EvalTransition$ function evaluates whether the time in the current state has expired. If so, he moves to the new state, which is obtained via the $UpdateStatus$ function (which considers the states shown in Figure 2). If there are different possible next states, the transition is probabilistic. An infected individual stops transitioning when he has reached the immune, recovered, or dead state.

The algorithm that computes the dissemination of the infectious agent is shown in Algorithm 2. For every infected individual l, the algorithm selects all individuals that he is connected to at this time (`L1`). For each one of those in the susceptible state, the probability that they will get infected is given by the $EvalInfection$ function (`L3`). This probability depends on the basic reproduction number of the infected individual, the type of connection,

Algorithm 1 UpdateStatus function for state transition of infected individuals.

Input: $(l, status_n, parameters(l))$ where l is the considered individual, $status_n$ contains characteristics and health status of each individual of the nth urban region; and $parameters(l)$ are parameters of the epidemic model for individual l.

Output: $(status_n(l))$ updated status of lth individual.

```
1: if status_n(l) is not Susceptible then
2:    if EvalTransition(parameters(l)) is True then
3:       status_n(l) = UpdateStatus(status_n(l))
4:    end if
5: end if
```

Algorithm 2 ComputeSpread function for the generation of new infected individuals.

Input: $(l, social_n, status_n, parameters(l))$ where l is the considered individual, $social_n$ is the set of graphs describing the social network of the nth urban region; $status_n$ contains characteristics and health status of each individual of the nth urban region; and $parameters(l)$ are parameters of the epidemic model for individual l.

Output: $(status_n)$ where updated status of individuals.

```
1: if status_n(l) is Infected then
2:    Connections = EvalConnections(social_n, l)
3:    for each individual i ∈ Connections do
4:       if status_n(i) is Susceptible && EvalInfection(parameters(l)) is True
         then
5:          status_n(i) = Primary latent
6:       end if
7:    end for
8: end if
```

the time of the day, and the specific characteristics of the individual. Note that both the $EvalTransition$ and the $EvalInfection$ functions use probability values and random-generated numbers, which give EpiGraph stochastic features.

A useful feature of EpiGraph is that it is possible to evaluate the effect of intervention strategies—such as vaccination, school closing, and social

Table 1: Parameters of the epidemic model for each state: R_0 is the basic reproduction number of the state, T_μ is the average time (in days) in each state and T_σ is its standard deviation.

State	R_0	T_μ **(days)**	T_σ **(days)**
Infective	1.3730	2	0.25
Latent	0.6850	0.25	0
Asymptomatic	0.6850	4.1	0.5
Infected Treated	0.0470	2	0.25
Latent Treated	0.0235	0.25	0
Asymptomatic Treated	0.0235	4.1	0.5

distancing—on the propagation. Social distancing restricts the interaction of individuals by retaining them at home and reflects measures of closing public facilities to mitigate the spreading of the disease.

3.3. The transportation model

The transportation model reflects the movement of people between cities for work, study, or vacation, and it is based on the gravity model proposed by Viboud *et al.* [2]. The number of individuals $\Delta P_{i,j}$ who move between locations i and j depends on the population size at both locations (P_i and P_j), as well as the distance between them ($d_{i,j}$). Equation 1 applies for travel distances of less than 120Km—which reflects the daily commute of students and workers to neighbouring cities. Equation 2 applies for the long-distance commute of workers that need to reside at a different location for several days in a row. Additionally, we consider people from any group type that move at any distance for several days for vacation purposes. Once the volume of inter-city commuters is calculated, we randomly select individuals from specific group types within the populations and move them for a specific period of time to other locations. In our experiments, of the short distance commuters, 85% are workers and 15% are students; for the long-distance commuters the

percentages are 50% workers, 30% students, 15% retired individuals, and 5% unemployed people.

$$(d_{i,j} < 120Km) \quad \Delta P_{i,j} = \frac{P_i^{0.30} Pj^{0.64}}{d_{i,j}^{3.05}} \tag{1}$$

$$(d_{i,j} \geq 120Km) \quad \Delta P_{i,j} = \frac{P_i^{0.24} Pj^{0.14}}{d_{i,j}^{0.29}} \tag{2}$$

The geographical information that EpiGraph takes into account includes latitude, longitude, and distance between urban regions, and was extracted from the Google Maps web service using the Google Distance Matrix API [32]. Although this work simulates urban regions that are spatially co-located within the same country, EpiGraph can be used to simulate very large-scale scenarios in which regions spawn different countries or continents.

3.4. The EpiGraph algorithm

Algorithm 3 shows the pseudocode of EpiGraph's simulator [33]. The iterative algorithm has four phases which execute every time step for each one of the simulated urban regions. The first phase (`L4`) updates the infectious status of every local individual l based on the epidemic model (in Algorithm 1). The second phase (`L6`) computes the dissemination of the infectious agent using the social model (Algorithm 2).

The third phase (`L9`) evaluates both pharmaceutical and non-pharmaceutical interventions in order to mitigate the propagation of the infectious disease. Non-pharmaceutical interventions—such as closing schools or social distancing—are triggered when the number of infected individuals in the population surpasses a threshold. The fourth phase computes the propagation of the

Algorithm 3 Spatial transmission algorithm.

Input: $(regions, social, status, distance, parameters)$ where $regions$ are the urban regions considered in the simulation, $social$ is the set of graphs describing the social network of each urban region; $status$ contains characteristics and health status of each individual for each urban region; $distance$ stores the distance for every pair of urban regions; and $parameters$ are parameters of the epidemic model for each individual)

Output: $(status)$ where $status$ is the updated status of individuals.

for $timestep = 1 \rightarrow simulation\_time$ **do**
  for each $region\ n\ \in\ regions$ **do**
    for each $individual\ l \in social_n$ **do**
      $UpdateStatus(l, status_n(l), parameters(l))$
      if $status_n(l)$ is $infectious$ **then**
        $ComputeSpread(l, social_n, status_n, parameters(l))$
      end if
    end for
    $Interventions(status_n)$
    for each $region\ m \in\ urban\_regions$, $(m\ \neq\ n)$ **do**
      $Transportation(social_m, social_n, distance_{m,n})$
    end for
  end for
end for

infection via the transportation model (`L11`) once a day for each pair of urban regions. Each subset of processes corresponding to a region compute the number of individuals which move from this region to another region depending on the size of the populations and the geographical distance between them.

EpiGraph uses a large portion of memory to store the infectious status and the connections of each individual. For example, a simulation of an area with 92 cities and an overall population of 21,320,965 inhabitants requires 31.3 GB of memory. This amount of data requires parallel data distribution and processing. The next section describes the parallel design and imple-

mentation of EpiGraph, including different optimization techniques applied for exploiting locality, improving communications and load balancing, and reducing the execution time by an efficient process placement.

4. Enhancing EpiGraph's performance

4.1. Data partitioning

We implemented EpiGraph as a parallel application based on the *Single Program Multiple Data* (SPMD) paradigm. SPMD applications require a workload partitioning strategy to distribute the data to the processes that execute in parallel. The interaction graph for each urban region is stored internally as a sparse matrix called interconnection matrix. In our case, this data is of the order of millions of individuals and interactions. To use the memory efficiently we do not replicate data structures between processes, but rather distributed them. We use a one-dimensional data decomposition strategy with block partitioning to assign different subsets of the data structures to the processes. The interaction graph is partitioned by dividing the population in equal-size blocks assigned to the subset of processes involved in the simulation of a specific urban region. Other data structures such as individuals' status—health status, age, or race—are partitioned using the same methodology.

Each process is responsible for simulating the virus propagation through the individuals assigned to it only. Communication and synchronization operations use MPI, which enables an efficient execution both on shared memory, as well as on distributed memory architectures.

Figure 3 illustrates an example of data partitioning for a simulation consisting of three urban regions: Madrid, Barcelona, and Valencia. Each of

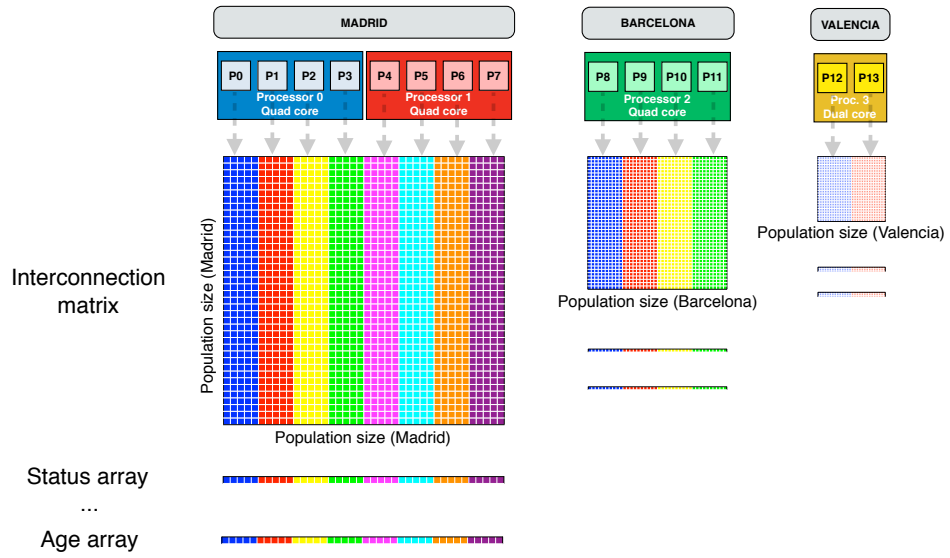


Figure 3: Example of data partitioning strategy in EpiGraph for Madrid, Barcelona, and Valencia.

them is simulated on a different subset of processes—8 processes for Madrid, 4 processes for Barcelona, and 2 processes for Valencia. Note that the interconnection matrices are sparse. *ComputeSpread* analyzes the contacts of infected individuals, which are stored as consecutive entries in the sparse matrix. An individual will more probably infect another individual of the same group based on the fact that there is a higher probability that they can come in close vicinity. As a consequence of our storage schema—which uses a block partition of the graph and assigns consecutive blocks of the sparse matrix to the same process—this individual is stored in a memory location close by. This fact allows us to exploit spatial data locality.

4.2. Communications

ComputeSpread and *Transportation* in Figure 3 are responsible for implementing the propagation behaviors within each urban region and among

different regions. If during the execution of *ComputeSpread* a local individual is infected by another one belonging to the same process, his state is locally updated. However, if this individual is not local and belongs to the same urban region then his new state is communicated to the remote process responsible for him via an *intra-region* communication. Intra-region communications are designed to overlap in time to minimize the communication overhead. Each process uses point-to-point `MPI_Send` and `MPI_Recv` to communicate those newly infected individuals who are local to each remote process within the same urban region. In addition, the *Interventions* function (Figure 3) performs a collective `MPI_Allgather` operation to collect statistics about the number of infected individuals in each region. These statistics are used to adopt intervention policies at region-level.

Inter-region communications are performed by the *Transportation* function to propagate the infection between individuals belonging to different regions. Each process from an urban region communicates with every remote region according to the flow of individuals which move between them. We use MPI point-to-point operations to transfer these individuals between processes.

EpiGraph is a communication-intensive application in which most of the execution time is spent in performing intra- and inter-region communications. As a result, our focus is on optimizing the communication between processes to reduce its overhead. All processes executing an MPI application are by default grouped into the global communicator `MPI_COMM_WORLD`. Any collective operation that uses this global communicator—such as those performed by the *Interventions* function—will block until all processes com-

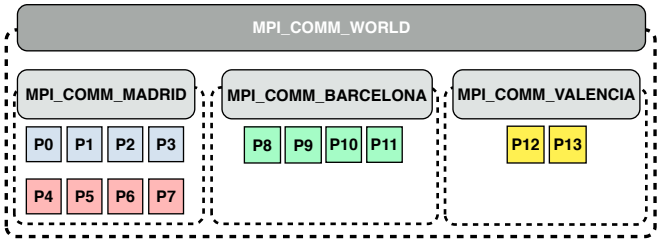


Figure 4: Two-level schema of MPI communicators.

plete. To overcome this performance bottleneck and reduce the communication overhead of collective operations we implement a communicator model based on a two-level schema: on the first level we have the default communicator—called *global communicator*, and on the second level we have ad-hoc communicators—called *local communicators*—that work at the granularity of each urban region. Figure 4 shows the two-level schema for a scenario consisting of the urban regions of Madrid, Barcelona, and Valencia.

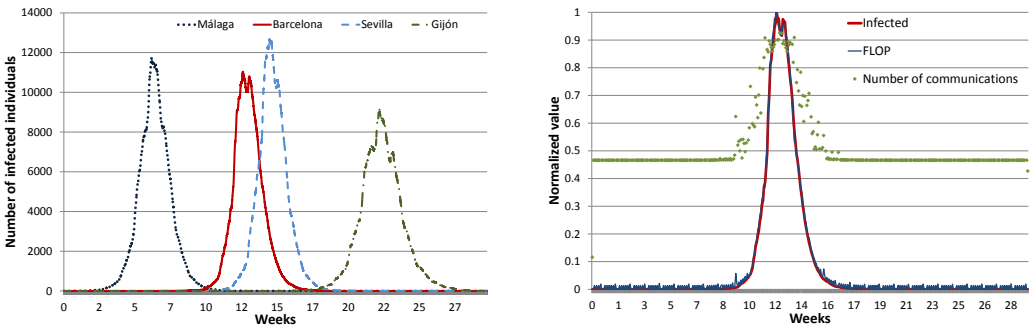
The global communicator is used for communicating between processes in behalf of the transportation model. Each process is identified by a `global_rank`. Intra-region communications involve both point-to-point messages to transmit new infectious states, and collective operations to gather the number of infected individuals. Once the algorithm has divided urban regions into groups of processes, the local communicators group together the subsets of processes involved in the computation of each specific region. The ad-hoc local communicators `MPI_COMM_[REGION]` enable the decoupled execution between those subsets of processes that are associated with each urban region. This improves the performance of intra-region collective operations and reduces the synchronization overhead. Processes are identified in the `MPI_COMM_[REGION]` communicator by a `local_rank`.

4.3. Load balancing and process mapping

An effective process mapping needs to balance the workload of the application between the MPI processes. We consider each of the n_c cores of every compute node to be independent *processing elements* (PE) with the same performance. We balance the workload both at the internal level of each urban region (intra-region level), as well as at the top level of the simulation (inter-region level).

To balance the workload at intra-region level we take into account the workload of each region and the available computing power of the platform. We calculate the relative computing power (in $FLOPS$) of the PEs and the load (in $FLOP$) associated to each region. The relative computing power of the PEs is evaluated by running an offline microbenchmark. The load associated with each city is estimated using PAPI performance counters [34] and taking into account the population size and the number of contacts within each region. Using these values, we assign a given number of FLOP to each PE. As a result, large regions are divided over several processes while small regions can be fully executed on a single process.

At the inter-region level we balance the computation by mapping the processes involved in the execution to the available PEs. Due to the dynamic computational load and irregular computation pattern of EpiGraph, finding an efficient mapping for a multicore cluster is not trivial. The key factor for propagating the infection beyond city boundaries is the transportation model, and cities with more traveling individuals have bigger chances of being infected. This probability is higher for cities located at less than 120Km from infected cities (short-distance transport) and for large populated areas (long-



(a) (b)

Figure 5: Simulation results: (a) infection spread for 92 regions when the disseminations starts in Malaga; (b) normalized values of infected individuals, FLOP, and number of communications for Barcelona.

distance transport). Figure 5(a) shows the number of infected individuals for a selection of four processes related to the regions of Malaga, Barcelona, Seville, and Gijon (one per region) in a simulation of 92 regions with the infection starting in Malaga. We observe that Barcelona is infected 10 weeks after the beginning of the simulation, and shortly after the infection starts in Seville. In contrast Gijon, a small area located in the north, starts the infection 17 weeks later.

Figure 5(b) shows the normalized computational load (in FLOP) and the number of communications of the process that runs the simulation for Barcelona. We can see that there is a direct correlation between the number of infected individuals and both the computation intensity (defined as number of FLOP per iteration) and the number of communications. Given that the distribution of infected individuals is different for each city—both in terms of number of infections and temporal distribution of the infection spread—we conclude that EpiGraph's computational load is also different for each

process both in computation intensity and temporal distribution.

The mapping algorithm presented in this paper follows some guiding principles such as assigning processes that do not exhibit simultaneous high computational load (like Malaga and Gijon) to the same processor while avoiding resource-competing processes (like Barcelona and Seville). This strategy prevents access conflicts to shared resources (e.g. shared cache levels, memory channels, and I/O buses). The basic idea of the mapping algorithm is to construct a weighted adjacency matrix $adja$ for all the processes. For a given pair of processes i and j, $adja(i, j)$ contains the combined number of FLOP for the iterations when their loads are simultaneously greater than a given threshold. We call this interval of iterations the *overlap* between processes i and j. In the previous example Malaga's process does not overlap in time with Gijon's, and the associated entry in $adja$ is therefore zero. On the other hand, the computational load of Barcelona and Seville overlaps during 5 weeks, which means that the value stored in $adja$ is the sum of the load in FLOP of the two cities during this interval. $adja$ allows us to identify which processes do not overlap during the program execution, and therefore can be potentially executed on the same compute node without competing for resources. In addition, this matrix also quantifies the amount of combined load, which is proportional to the aggregated computation intensity of both processes during the overlap.

Figure 4 shows the load-aware process mapping algorithm. It receives the total number of processes, the number of processing elements (nodes and cores), and the weighted adjacency matrix. It returns map, a structure which specifies the PE assigned to each process p. We identify each PE as

Algorithm 4 Load-aware process mapping algorithm.

Input: $(n_p, n_n, n_c, adja)$ where n_p is the number of processes, n_n and n_c the number of nodes and cores of the platform, and adja the weighted adjacency matrix.

Output: (map) is the core and core mapping assigned to each process.

```
1:  for process p = 1, n_p do
2:     for node n = 1, n_n do
3:        {procs} = take_processes(PE_{n,*}, map)
4:        Load_n = take_load(adja, {procs}, p)
5:     end for
6:     n_min = take_minimum_load(Load_{1:n_n})
7:     for core c = 1, n_c do
8:        {procs} = take_processes(PE_{n_min,c}, map)
9:        Load_c = take_load(adja, {procs}, p)
10:    end for
11:    c_min = take_minimum_load(Load_{1:n_c})
12:    map(p) = {n_min, c_min}
13: end for
```

$PE_{\{n,c\}}$, where c is the core belonging to the compute node n. The first step of the algorithm computes the overlap of each process with all the processes assigned to each compute node. In line (`L3`), function *take_processes* obtains the list of processes that are already mapped to node n. The function receives as an argument the list of cores associated to the node and the mapping of the previously assigned processes. Function *take_load* (`L4`) subsequently computes the accumulated load for the overlap between the process p and all the processes running on node n by using the adjacency matrix. In line `L6` the algorithm selects the node with the minimum accumulated load.

The second step is to find the core which is best suited to host the process under consideration. Lines `L7-L11` of the algorithm evaluate individually each core of the node n_{min} already selected. Using the adjacency matrix we analyze the load during the overlap between the process p and the processes

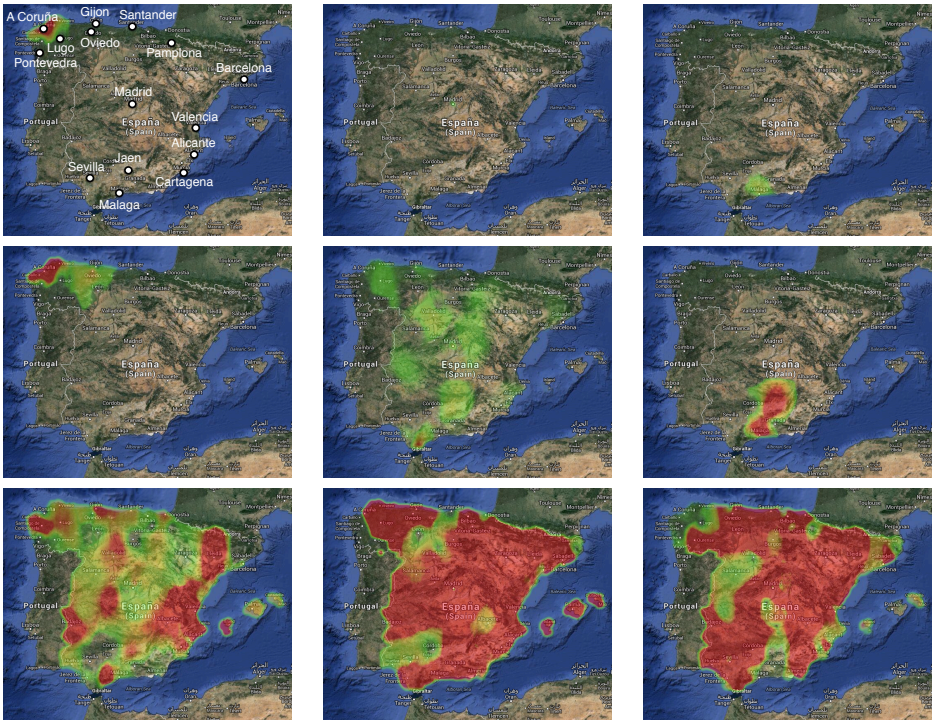
assigned to each core (line `L8`). Then, in line `L11` we select the core with the smallest overlap in load. Finally, the algorithm updates *map* with the new mapping. The next section presents a detailed evaluation of the performance improvement achieved with this technique.

5. Performance evaluation

For our experiments we consider the propagation of Influenza throughout the most populated cities of Spain. We evaluated EpiGraph by simulating the spatial transmission both on a distributed memory system and on a shared memory system. The distributed platform is a cluster with 19 compute nodes, each of them having one Intel Quad Core Xeon E5405 processor running at 2.00GHz and 4GB of memory. The shared memory system consists of a single compute node which has four Intel Xeon E7-4807 processors with Hyper-Threading support and 6 cores each, running at 1.87GHz and 128GB of memory. All the compute nodes run a Linux Ubuntu Server 10.10 with the 2.6.35-32 kernel and are interconnected by a Gigabit Ethernet network. We use the MPICH-2 v1.4.1 implementation of MPI. The problem instances are available at [35], including the executable version for running the experiments.

5.1. Large-scale area simulations

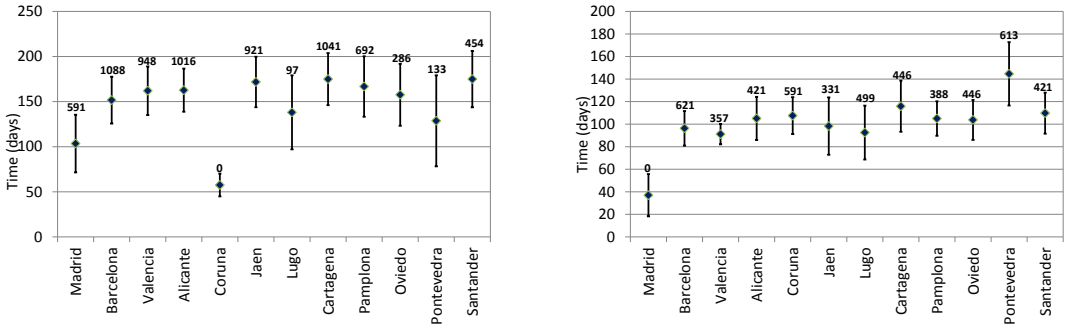
We simulated the virus propagation for the 92 most populated cities in Spain [24] and a simulated time span of one year. We executed the scenario on the cluster using 76 processes—4 processes per compute node—and we compare the spatio-temporal propagation of the infectious disease when the outbreak originates in different regions.



(a) A Coruña (b) Madrid (c) Malaga

Figure 6: Comparison of the spatio-temporal propagation of Influenza since the outbreak of epidemics originated in (a) A Coruña, (b) Madrid and (c) Malaga.

Figure 6 illustrates the spatio-temporal propagation of the Influenza epidemics started in A Coruña, Madrid, and Malaga. We observe that the infectious disease propagates rapidly when the epidemic originates in a highly populated, well connected region (Madrid) compared to a smaller, more isolated region (A Coruña). When the epidemic starts in Madrid the disease propagates quickly not only to neighboring regions, but also to distant regions—due to the higher travel volume and the more frequent long distance travel. When the epidemic starts in an isolated region such as A Coruña the virus takes several weeks to reach far away regions. Medium connected cities like Malaga obtain intermediate results. Using the transportation model to perform large



(a) A Coruña scenario (b) Madrid scenario

Figure 7: Comparison of the peak infection time for (a) A Coruña and (b) Madrid and scenarios. The standard deviation is represented as vertical error lines and the numerical values on top of them are the distance in kilometers to the source city of the infection.

scale simulations allows us to predict not only the impact of the infection on the population but also the temporal evolution of the propagation.

5.2. Statistical analysis

We performed a statistical evaluation of the results produced by EpiGraph by running each scenario of the large-scale simulation 20 times. We analyzed the variability of the infection spread pattern measuring the *peak infection time* for each execution. This time is defined as the time when the maximum number of infected individuals is reached considering both the primary and secondary infection states. For instance, in Figure 5(a), the peak infection time is around week 6 for Malaga and week 12 for Barcelona. For each city in each scenario we obtained the average and standard deviation values of the peak infection time. Without loss of generality, Figure 7 shows the results for A Coruña and Madrid scenarios. The standard deviation is represented as vertical error lines and the numerical values on top of them are the distance

in kilometers to the source city of the infection. In Figure 6 you can see the geographical location of all the cities mentioned in the rest of the paper. If we compare Figures 7(a) and (b) we see that the peak propagation times are smaller for Madrid. This is because A Coruña is a medium-size city located on the north-west coast of Spain, far away from many of the rest of the cities in the country. In contrast, Madrid is the largest city located in the center of the country. According to the transportation model, Madrid generates many more traveling individuals than A Coruña (both because of the distance to Malaga and A Coruña and to the volume of population involved), which produces a faster spread of the infection. In the case of the infection starting in A Coruña we can observe that the cities that first reach the peak infection time are large cities (like Madrid or Barcelona) and medium-size cities that are close to the initial infection point (like Lugo or Pontevedra). In contrast, in case of the infection starting in Madrid, this reaches its peak in most of the cities around the same time because of Madrid‘s central location and large size.

5.3. Performance analysis

The following experiment evaluates the performance of EpiGraph when executing on distributed and shared parallel architectures. We simulated the propagation of the virus for a medium-scale scenario which consists of a subset of 4 urban regions: Madrid, Barcelona, Valencia, and Seville. This configuration allows us to evaluate the scalability of EpiGraph by increasing linearly the number of processes. The execution of the medium-scale simulations requires a minimum of 2 compute nodes (running 4 processes each) when executing on the cluster due to the large memory footprint of

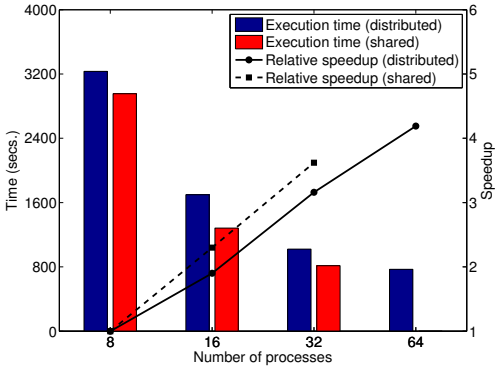


Figure 8: Process mapping for EpiGraph simulations, where NP stands for the number of processes. For 8 processes we used 4 processes for Madrid, 2 for Barcelona, 1 for Valencia and 1 for Seville. Doubling the number of processes implies doubling the number of processes for each city.

the simulator. We take the execution with 8 processes as base, both for the distributed and the shared configurations. The experiment is bounded from above by 32 processes in the shared memory system because it has 48 logical PEs supported by Hyper-Threading and it is not possible to map 64 processes. Figure 8 shows how the application scales on both parallel architectures with respect to the performance of the base execution. EpiGraph scales well up to 32 processes in the cluster and almost linearly in the shared memory system due to a low intra-node communication overhead.

To analyze the performance in greater detail we profiled EpiGraph by instrumenting the code with wall-clock timing functions. These collect the time spent by each process in each of the functions of Algorithm 3. Function *ComputeSpread* computes the dissemination of the virus and communicates the newly infected individuals to the processes that are responsible for them. To perform a precise profiling we split each such task in two components: *ComputeSpread_Comp* for the computation time and *ComputeSpread_Comm*

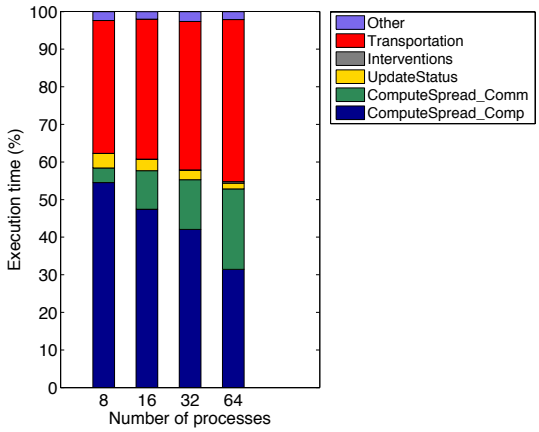


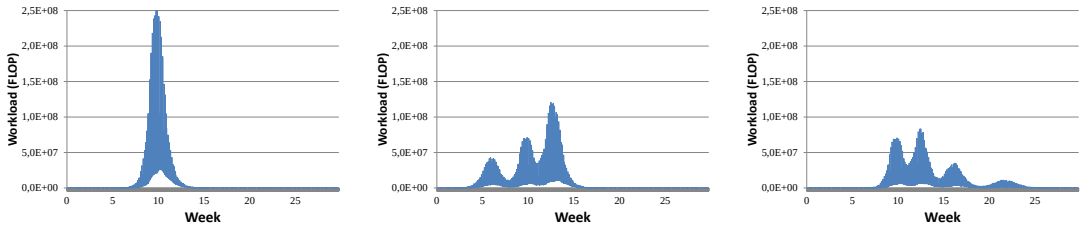
Figure 9: Time spent by EpiGraph in each of the phases of the simulation algorithm.

for the communication time. Figure 9 shows the percentage of the execution time spent in each of these functions. As expected, the percentage of the execution time spent in the computation phases (*ComputeSpread_Comp* and *UpdateStatus*) decreases and the time spent in the communication phases (*ComputeSpread_Comm*, *Interventions* and *Transportation*) increases when simulating with more processes. When executing on 32 and 64 processes more than half of the execution time is invested in communication operations. The execution time of the *Transportation* phase is significantly larger than the execution time of the *ComputeSpread_Comm* and *Interventions* phases. The simulation of the transportation model involves both collective and point-to-point communication and synchronization operations between all of the running processes, which increases the cost of inter-region communications. Communication operations in the *Interventions* phase are performed using the ad-hoc local communicators; this optimizes the cost of the collective intra-region communications to less than 1% of the execution time.

5.4. Process mapping

The load-aware process mapping algorithm relies on the use of a weighted adjacency matrix. The infection dissemination has a random component due to the fact that every time we run a simulation the transportation model propagates the infection in a slightly different way. This variability of propagation patterns affects the contents of the adjacency matrix. However, all the executions of the same scenario have a common propagation pattern which is related to the transportation model: the cities that are infected earliest are usually the largest ones or the closest to the already infected cities. Based on this property we run 20 times each one of the three city scenarios (Madrid, A Coruña, and Malaga) and use profiling to obtain the adjacency matrix for each run. For each city we build the *average adjacency matrix* by counting the number of times that each entry of the adjacency matrix appears in the matrices of the 20 runs. If this entry has several occurrences then we include it in the average adjacency matrix with a value equal to the average value of the occurrences. Otherwise the matrix receives a zero in that cell. This approach allows us to generate an adjacency matrix that contains the most common propagation patterns.

We run these experiments on the distributed memory system, for which we consider three different block-wise mappings. The first mapping assigns processes {1,2,3,4} to compute node 1 (consisting of a four-core processor), processes {5,6,7,8} to compute node 2, and so on. This is the default mapping that the Torque resource manager uses [36]. This mapping raises problems for cities whose simulation runs on several processes. These cities have several processes assigned to the same compute node, which causes contention con-



(a) Block (b) Cyclic (c) Load-aware

Figure 10: Aggregated load of compute node 3 for Malaga and a simulation with 24 cities and 16 compute nodes.

flicts. Figure 10(a) shows the aggregated load of a compute node which has been assigned four processes corresponding to Madrid and having a similar load distributions. As a result we observe a load peak which can potentially produce contention conflicts when accessing the shared resources of the processor (cache, memory and I/O buses, etc.). The second mapping –called *cyclic mapping*– alleviates this problem by using a round-robin distribution such that processes {1,2,3,4} are assigned to compute nodes 1, 2, 3 and 4, and so on. This mapping distributes the processes of the same city on different compute nodes but does not leverage the knowledge about the common propagation patterns of the processes. Figure 10(b) shows the result of this mapping for the same compute node. The node now runs processes of four different cities and the work load is more evenly distributed than in the previous example. However, two cities still overlap in time to create a load peak around the 13th week. The load-aware mapping algorithm takes into account the load distribution of all the processes involved in the execution. Figure 10(c) shows the load using the average adjacency matrix. We can see that the algorithm reduces the peak load to produce a more even distribution.

Table 2: Percentage-wise reduction in execution time of cyclic and load-aware mappings with respect of the block mapping policy for 92 cities and 131 processes.

Compute nodes	Mapping policy	Madrid	Malaga	A Coruña
19	Cyclic	12.2%	13.7%	12.7%
19	Load-aware	21.5%	16.8%	15.0%
18	Cyclic	2.2%	7.8%	6.1%
18	Load-aware	7.3%	12.8%	11.3%
17	Cyclic	5.9%	6.3%	6.1%
17	Load-aware	13.4%	10.4%	9%

Table 2 shows the percentage-wise reduction in the overall execution time of the cyclic and load aware mappings with respect to the block mapping. The simulation time for block mapping is around 5,500 secs. We evaluate each one of the three city scenarios for 19, 18 and 17 compute nodes. We can see that the load-aware mapping algorithm obtains the greatest reductions in the execution time.

6. Conclusions

EpiGraph is a parallel tool that simulates the spatial transmission of Influenza. This paper presents several techniques that improve its simulation capabilities as well as its performance. In this work we extend EpiGraph with a geographic location and transportation model which allows us to study the spatial dynamics of the virus propagation over large-scale areas. We have evaluated EpiGraph considering the 92 most populated cities of Spain and different epidemiological scenarios. EpiGraph captures features of the population model at the level of both the individual and each of his interactions. This implies large computational and memory requirements. The second contribution of this work is a set of different parallelization techniques to deal with these requirements efficiently. These include data partitioning methods

that exploit data locality and enable load balance, inter-process communication optimization based on a two-level MPI communicator schema, and a process-to-processor mapping algorithm that considers the load variability of EpiGraph. Results show that EpiGraph can be executed efficiently for large-scale scenarios both on a cluster platform and on a shared memory compute node. Additionally, the mapping algorithm reduces the execution time up to 21.5% compared with the default mapping provided by Torque.

6.1. Future work

There are several directions that deserve further investigation on our part. One of them arises from the fact that in the transportation model the actual individuals that travel out of town are randomly chosen. It would we interesting to see how fixing the pool of travelers—at least that of daily commuters—affects the stability of the propagation in terms of the order (and intensity) in which new cities become infected. In terms of process-to-processor mapping we would like to evaluate how well a dynamic mapping can do compared to our (pattern-based) load-aware mapping strategy. Finally, a less immediate extension would be the addition of a module which can capture potential (person-specific) changes in travel behavior as a result of acquiring knowledge about the cities that are already infected. This would allow us to study, qualitatively and quantitatively, the effect of people's behavior on the timing and intensity of epidemics.

7. Acknowledgments

We would like to acknowledge the assistance provided by David del Río Astorga and Alberto Martín Cajal. This work has been partially supported

by the Spanish Ministry of Science TIN2010-16497, 2010.

References

[1] S. B. Thacker, Spatial aspects of influenza epidemics, The Journal of the American Medical Association 258 (1987) 2593–2594.

[2] C. Viboud, O. N. Bjørnstad, D. L. Smith, L. Simonsen, M. A. Miller, B. T. Grenfell, Synchrony, waves, and spatial hierarchies in the spread of influenza, Science 312 (2006) 447–451.

[3] G. Martín, M. Marinescu, D. Singh, J. Carretero, Leveraging social networks for understanding the evolution of epidemics, BMC Syst Biol 5 (2011).

[4] W. Kermack, A. McKendrick, Contributions to the mathematical theory of epidemics, in: Proceedings of the Royal society of London. Series A, volume 115, 1927, pp. 700–721.

[5] M. Eichner, M. Schwehm, H. Duerr, S. Brockmann, The influenza pandemic preparedness planning tool influsim, BMC infectious diseases 7 (2007) 17.

[6] L. Hufnagel, D. Brockmann, T. Geisel, Forecast and control of epidemics in a globalized world, Proceedings of the National Academy of Sciences of the United States of America 101 (2004) 15124–15129.

[7] L. A. Meyers, B. Pourbohloul, M. E. Newman, D. M. Skowronski, R. C. Brunham, Network theory and SARS: predicting outbreak diversity, Journal of theoretical biology 232 (2005) 71–81.

[8] S. Bansal, B. T. Grenfell, L. A. Meyers, When individual behaviour matters: homogeneous and network models in epidemiology, Journal of the Royal Society Interface 4 (2007) 879–891.

[9] R. Anderson, R. May, B. Anderson, Infectious diseases of humans: dynamics and control, volume 28, Wiley Online Library, 1992.

[10] M. J. Keeling, K. T. Eames, Networks and epidemic models, Journal of the Royal Society Interface 2 (2005) 295–307.

[11] L. Mao, L. Bian, Spatial–temporal transmission of influenza and its health risks in an urbanized area, Computers, Environment and Urban Systems 34 (2010) 204–215.

[12] Y. Ma, K. R. Bisset, J. Chen, S. Deodhar, M. V. Marathe, Formal specification and experimental analysis of an interactive epidemic simulation framework, in: High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference on, IEEE, 2011, pp. 790–795.

[13] K. Bisset, J. Chen, X. Feng, V. Kumar, M. Marathe, Epifast: a fast algorithm for large scale realistic epidemic simulations on distributed memory systems, in: Proceedings of the 23rd international conference on Supercomputing, ACM, 2009.

[14] D. Chao, M. Halloran, V. Obenchain, I. Longini, FluTE, a publicly available stochastic influenza epidemic simulation model, PLoS computational biology 6 (2010) e1000656.

[15] C. L. Barrett, K. R. Bisset, S. G. Eubank, X. Feng, M. V. Marathe, EpiSimdemics: an efficient algorithm for simulating the spread of infectious disease over large realistic social networks, in: Proceedings of the 2008 ACM/IEEE conference on Supercomputing, IEEE Press, 2008, p. 37.

[16] M. Eichner, M. Schwehm, H.-P. Duerr, S. Brockmann, The influenza pandemic preparedness planning tool influsim, BMC Infectious Diseases 7 (2007) 17.

[17] Pandemic Flu Preparedness Tools, Centers for Disease Control and Prevention (CDC), http://www.cdc.gov/flu/tools, 2012.

[18] D. X. P. R. J. X. W. E. e. a. Haber MJ, Shay DK, Effectiveness of interventions to reduce contact rates during a simulated influenza pandemic, Emerging Infectious Diseases 13 (2007) 9.

[19] W. Broeck, C. Gioannini, B. Goncalves, M. Quaggiotto, V. Colizza, A. Vespignani, The gleamviz computational tool, a publicly available software to explore realistic epidemic spreading scenarios at the global scale, BMC Infectious Diseases 11 (2011) 37.

[20] S. Edlund, M. Davis, J. Kaufman, Extending geospatial data to support epidemiological modeling, in: Proceedings of the 20th International Conference on Advances in Geographic Information Systems, SIGSPATIAL '12, ACM, New York, NY, USA, 2012, pp. 589–592. URL: http://doi.acm.org/10.1145/2424321.2424423. doi:10.1145/2424321.2424423.

[21] S. Edlund, J. Kaufman, J. Lessler, J. Douglas, M. Bromberg, Z. Kaufman, R. Bassal, G. Chodick, R. Marom, V. Shalev, Y. Mesika, R. Ram, A. Leventhal, Comparing three basic models for seasonal influenza, Epidemics 3 (2011) 135 – 142.

[22] S. Edlund, M. Davis, J. Douglas, A. Kershenbaum, N. Waraporn, J. Lessler, J. Kaufman, A global model of malaria climate sensitivity: comparing malaria response to historic climate data based on simulation and officially reported malaria incidence, Malaria Journal 11 (2012) 331.

[23] J. Epstein, D. Goedecke, F. Yu, R. Morris, D. Wagener, G. Bobashev, Controlling pandemic flu: The value of international air travel restrictions, PLoS ONE 2 (2007) e401.

[24] National Statistics Institute (INE), http://www.ine.es/, 2013.

[25] L. Jin, Y. Chen, P. Hui, C. Ding, T. Wang, A. V. Vasilakos, B. Deng, X. Li, Albatross sampling: robust and effective hybrid vertex sampling for social graphs, in: Proceedings of the 3rd ACM international workshop on MobiArch, ACM, 2011, pp. 11–16.

[26] J. L. Aron, I. B. Schwartz, Seasonality and period-doubling bifurcations in an epidemic model, Journal of Theoretical Biology 110 (1984) 665–679.

[27] F. Brauer, P. Van den Driessche, J. Wu, L. Allen, Mathematical epidemiology, volume 1945, Springer Verlag, 2008.

[28] M. E. Alexander, C. S. Bowman, Z. Feng, M. Gardam, S. M. Moghadas, G. Roest, J. Wu, P. Yan, Emergence of drug resistance: implications

for antiviral control of pandemic influenza., Proceedings of the Royal Society 274 (2007) 1675–1684.

[29] F. Brauer, P. v. d. Driessche, J. Wu (Eds.), Mathematical Epidemiology, Springer, 2008.

[30] I. M. Longini, E. M. Halloran, A. Nizam, Y. Yang, Containing pandemic influenza with antiviral agents, American Journal of Epidemiology 159 (2004) 623–633.

[31] L. R. Elveback, J. P. Fox, E. Ackerman, A. Langworthy, M. Boyd, L. Gatewood, An influenza simulation model for immunization studies., American Journal of Epidemiology 103 (1976) 152–165.

[32] Google Maps API, https://developers.google.com/maps/, 2013.

[33] G. Martín, D. E. Singh, M.-C. Marinescu, J. Carretero, Parallel algorithm for simulating the spatial transmission of influenza in epigraph, in: Proceedings of the 20th European MPI Users' Group Meeting, EuroMPI '13, ACM, New York, NY, USA, 2013, pp. 205–210. URL: http://doi.acm.org/10.1145/2488551.2488585. doi:10.1145/2488551.2488585.

[34] P. Mucci, S. Browne, C. Deane, G. Ho, PAPI: A portable interface to hardware performance counters, in: Proceedings of the Department of Defense HPCMP Users Group Conference, 1999, pp. 7–10.

[35] EpiGraph Home Page, http://www.arcos.inf.uc3m.es/∼desingh/epigraph, 2014.

[36] G. Staples, TORQUE resource manager, in: Proceedings of the 2006 ACM/IEEE conference on Supercomputing, ACM, 2006, p. 8.