DIMPLES: Distributed Influence Maximization for Pandemic pLanning on Exascale Systems

Marco Minutoli
Pacific Northwest National Laboratory
Richland, WA, USA
marco.minutoli@pnnl.gov

Reece Neff
North Carolina State University
Raleigh, NC, USA
rwneff@ncsu.edu

Naw Safrin Sattar
Oak Ridge National Laboratory
Oak Ridge, TN, USA
sattarn@ornl.gov

Hao Lu
Oak Ridge National Laboratory
Oak Ridge, TN, USA
luh1@ornl.gov

John Feo
Pacific Northwest National Laboratory
Richland, WA, USA
john.feo@pnnl.gov

Henning Mortveit
University of Virginia
Charlottesville, VA, USA
henning.mortveit@virginia.edu

Anil Vullikanti
University of Virginia
Charlottesville, VA, USA
vsakumar@virginia.edu

Dawen Xie
University of Virginia
Charlottesville, VA, USA
dx7fu@virginia.edu

Mandy L Wilson
University of Virginia
Charlottesville, VA, USA
alw4ey@virginia.edu

Gregor von Laszewski
University of Virginia
Charlottesville, VA, USA
thf2bn@virginia.edu

Parantapa Bhattacharya
University of Virginia
Charlottesville, VA, USA
pb5gj@virginia.edu

S M Ferdous
Pacific Northwest National Laboratory
Richland, WA, USA
sm.ferdous@pnnl.gov

Ananth Kalyanaraman
Washington State University
Pullman, WA, USA
ananth@wsu.edu

Michela Becchi
North Carolina State University
Raleigh, NC, USA
mbecchi@ncsu.edu

Madhav Marathe
University of Virginia
Charlottesville, VA, USA
mvm7hz@virginia.edu

Mahantesh Halappanavar
Pacific Northwest National Laboratory
Richland, WA, USA
hala@pnnl.gov

ICS '25, Salt Lake City, UT, USA

ACM ISBN 979-8-4007-1537-2/25/06
https://doi.org/10.1145/3721145.3730414

Abstract

We study exascale parallel algorithms for the selection of intervention or monitoring strategies in massive realistic socio-technical networks through scalable Influence Maximization (InfMax) algorithms. We employ novel techniques to enable efficient scaling on up to 8k nodes of OLCF Frontier, with 65k AMD GPUs and 458k AMD CPU cores. Current state-of-the-art InfMax tools are limited to networks with only a few million actors (vertices) and a few hundred million interactions (edges). By overcoming these limitations,

we show that our approach is capable of processing a realistic social contact network of the United States with 285 million nodes and about 8 billion edges. This two orders-of-magnitude improvement over the previous state-of-the-art is obtained by leveraging algorithmic advancements for the InfMax problem and designing several problem-specific approaches to overlap communication with computation, improve GPU efficiency, and lower the application's memory requirements.

We evaluate strong scaling for computing 10k most influential seeds using up to 8k nodes of an exascale system, and weak scaling from 128 to 8k system nodes for seed sets ranging from 625 to 40k seeds. We achieve the fastest-known runtime of 25 minutes while performing 48 million diffusion simulations totaling 2.31 petabytes to identify 40k influential seeds using 8k nodes, and take 5.75 minutes to identify 10k seeds while using 4k nodes.

CCS Concepts

• **Computing methodologies** → **Massively parallel algorithms**; **Massively parallel and high-performance simulations**; • **Software and its engineering** → **Massively parallel systems**; • **Mathematics of computing** → *Approximation algorithms.*

Keywords

Influence Maximization, High-Performance Computing, Parallel Graph Algorithms, Distributed Graph Algorithms

ACM Reference Format:
Marco Minutoli, Reece Neff, Naw Safrin Sattar, Hao Lu, John Feo, Henning Mortveit, Anil Vullikanti, Dawen Xie, Mandy L Wilson, Gregor von Laszewski, Parantapa Bhattacharya, S M Ferdous, Ananth Kalyanaraman, Michela Becchi, Madhav Marathe, and Mahantesh Halappanavar. 2025. DIMPLES: Distributed Influence Maximization for Pandemic pLanning on Exascale Systems. In *2025 International Conference on Supercomputing (ICS '25), June 08–11, 2025, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3721145.3730414

1 Introduction

Computational epidemiology aims to develop computer models and decision support systems to understand, predict and control the spatiotemporal diffusion of disease through populations. The models may range from descriptive, for example, static estimates of correlations within large databases, to generative, for example, computing the spread of different kinds of contagions via person-to-person interactions through a large population––these include the spread of a disease, as well as (mis)information and fear about the disease. Computational models help in understanding the space-time dynamics of epidemics.

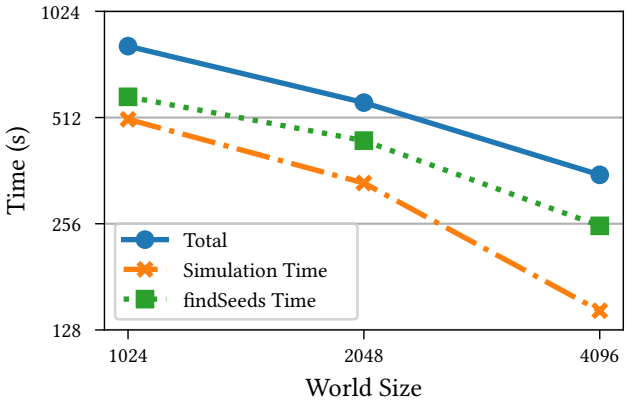

Figure 1: Strong scaling on OLCF's Frontier supercomputer up to 4,096 nodes on the USA-scale network. We fix parameters to: $k = 10k$, $\varepsilon = 0.03$, and $\delta = 0.01$. This problem instance does not fit on fewer than 1,024 nodes. Due to ranks initializing at different times, we mark the start of the program as the median start time of all nodes.

The threat and impact of infectious diseases have shaped our society and continue to be of major global importance – the COVID-19 pandemic is a reminder that pandemics will happen and cause severe social, economic, and health impacts. Despite significant advances by scientists and public health authorities, we continue to be surprised by disease outbreaks of novel or known pathogens across the world and are unable to mount a rapid and effective response. Climate change, urbanization, globalization, immuno-compromised populations, and the declining effectiveness of antibiotics in treating common diseases pose new and difficult challenges.

Compartmental mass action models have been a cornerstone of mathematical epidemiology. The basic idea is to partition a homogeneously mixing population into a small set of compartments representing the possible disease states, e.g., susceptible (S), infectious (I), and removed/recovered (R), and specifying transition rates among the states. Compartmental mass action models, often represented using ordinary differential equations (ODE), reproduce commonly-observed features of outbreaks, such as a self-limiting period of nearly exponential growth to a single peak followed by a gradual decrease as the pool of susceptible (S) population is depleted.

An alternative way to study epidemics is to explicitly represent the underlying contact structure that drives an epidemic [21, 27, 35, 43]. In this paper, we will focus on *networked models*, which consider epidemic spread on an undirected social interaction network $G(V, E)$ over a population V, each edge $e = (u, v) \in E$ implies that individuals (also referred to as nodes) $u, v \in V$ interact. The specific form of interaction depends on the disease being modeled; e.g. sexually transmitted

diseases require physical sexual contact, while influenza-like illnesses require physical proximity. Let $N(v)$ denote the set of neighbors of v. The SIR model on the graph G is a dynamical process in which each node is in one of S, I or R states. Infection can potentially spread from u to v along edge $e = (u, v)$ with a probability of $\beta(e, t)$ at time instant t after u becomes infected, conditional on node v remaining uninfected until time t — this is a discrete version of the rate of infection for the ODE model discussed earlier; we use β to refer to the vector of transmission probabilities over edges over time. We let $I(t)$ denote the set of nodes that become infected at time t. $I(0)$ denotes the set of source nodes where the infection started; in general, there would be a lot of uncertainty associated with the sources, and we assume they are drawn from a distribution.

1.1 Problem Formulation

Intuitively, our goal is to choose an intervention that controls the epidemic outbreak most effectively. Here, we focus on vaccination strategies, i.e., which subset of nodes to vaccinate, so that the expected outbreak size is minimized. If a subset X_t of nodes are vaccinated at time t, this can be modeled as removing these nodes from the network at time t, assuming that the vaccine is perfect. It is possible that some nodes of X_t were already infected and become immune; in this case, we assume the vaccine doesn't impact their state. We refer to the sequence $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ as the vaccination strategy. We assume there is a budget B_t on the number of vaccinations that can be administered at any time t; let $\mathbf{B}$ denote the vector of budgets over time. Then, $\mathbf{X}$ is feasible if $|X_t| \leq B_t$ for all t. Let #infections$(G, \beta, I(0), \mathbf{X}, \mathbf{B})$ denote the number of infections for an SIR type disease model specified by β on G, with the outbreak starting at set $I(0)$, when the vaccination strategy is $\mathbf{X}$ and budget sequence is $\mathbf{B}$. The EpiControl problem is defined as follows:

Instance. Given a contact network $G = (V, E)$, a disease model specified by β, a set $I(0)$ of initial infections, and a budget B_t at each time.

Goal. Find a vaccination strategy $\mathbf{X}$ such that $|X_t| \leq B_t$ for all t, and the expected number of infections $\mathbb{E}$[#infections($G, \beta, I(0), \mathbf{X}, \mathbf{B}$)] is minimized.

This minimization problem has been shown to have a direct relation with the maximization objective of the classical influence maximization problem [26]. More specifically, given a graph $G(V, E)$, a diffusion process M, and a budget $k > 0$, the goal of the classical influence maximization problem is to identify k vertices (as "seeds") that maximize the expected influence spread over the network. This problem was first proposed by Domingos and Richardson [26] in the context of viral marketing on online social networks, where a product given freely to certain well-connected individuals led to widespread adoption through a "word of mouth" propagation, by building on ideas of network diffusion processes in the social sciences. In [49], the authors show that the EpiControl problem of identifying a vaccination strategy $\mathbf{X}$ (under budget B) can be reduced to identifying k seeds under the influence maximization problem, where $B = k$, with the maximization objective set to the number of lives saved. This problem transformation builds on the simple intuition that for a node u to be spared from the disease, there should exist a vaccinated node along all paths from another infected source to u. In essence, this strategy aims at building herd immunity under budget constraints. Therefore, a subset of k nodes that maximize the (probabilistic) reachability to the rest of the network also signifies an optimal set of nodes to be vaccinated.

While the preliminary results of [49] show the validity of the approach, EpiControl algorithms are notorious for their high computational and memory requirements [48]. The current state-of-the-art generally limits at analyzing relatively small networks with few millions of nodes and at most hundreds million edges. When larger networks are used, the experimental settings are such that they warrant reduced computational and memory costs to arrive at the solutions at the expenses of quality. Capitalizing on the key observations of [49] and recent algorithmic advancements [58], in this paper, we make the next leap towards making scalable influence maximization a practical option in implementing pandemic planning by designing a parallel algorithm that can process synthetic contact networks that are comparable in size to the population of the United States of America and in structure with its mobility patterns.

1.2 Contributions

Our key results include:

- The first parallel and distributed algorithm for the EpiControl problem based on OPIM-C [58] that can scale on large distributed systems. Our proposed approach includes communication and computation overlapping techniques through a speculative execution scheme that completely overlaps the simulation and seed selection phases of the algorithm.
- A detailed performance study on how to optimize the algorithm on the Frontier supercomputer at the Oak Ridge Leadership Computing Facility (OLCF). We present strategies to efficiently pack large graphs into the GPU memory of the system. Furthermore, we study the efficiency of pseudo-random number generation on GPUs, a vital component of diffusion simulations within Influence Maximization algorithms.
- A tool that enables the analysis of US-scale contact networks with $\approx 260M$ vertices and $\approx 8B$ edges. This

tool demonstrates unprecedented scalability, achieving nearly full machine runs on Frontier (8k nodes: 65k GPUs and 458k CPU cores). This provides a substantial advancement in large-scale model exploration with the potential to influence policy decisions, as has been observed in previous works.

2 Related Work

Beginning in 2016 [34], several efforts have been made to parallelize InfMax on a variety of computing resources ranging from multicore CPU-only servers to distributed multi-GPU compute clusters [10, 24, 33, 51, 61]. Apart from some approaches based on mathematical programming to compute InfMax better than the $(1-1/e)$-approximation bound, most efforts focus on two main approaches: reverse influence sampling and greedy hill-climbing, which we will discuss briefly.

Reverse Influence Sampling (RIS): The `Ripples` library represents the state-of-the-art InfMax implementations on distributed multi-GPU systems [47, 48], and is based on the IMM [59] algorithm that bounds the sampling effort for approaches leveraging RIS. Algorithms based on RIS generally consist of two building blocks: *(i)* Randomized breadth-first traversals collecting random reverse reachability (RRR sets) information from the diffusion process originating at randomly selected nodes, and *(ii)* a maximum k-coverage algorithm used to select influential seeds (set S) from the RRR sets. Performance analysis shows that computation is dominated by the sampling steps producing the RRR sets [48], a trait common to all RIS-based algorithms. Several techniques have been proposed to accelerate sampling, such as fusing the random traversals and collecting information as sketches (summarizations) [30–32]. The `Ripples` framework implements similar ideas and is tailored to amortize the offloading cost to GPUs as well as make better use of the memory bandwidth of both GPUs and CPUs [50]. Recent approaches such as GreediRIS [9] have also been developed to address the scaling issues of the second building block (seed selection phase) of the IMM algorithm.

The recent work of Neff et al. [50] shows the scalability of the IMM algorithm up to 4k nodes (32k GPUs) of Frontier at Oak Ridge Leadership Class Facility. However, their largest runs are limited to selecting only hundreds of seeds on networks with 4 million vertices. Modeling the memory requirements of the IMM algorithm by analyzing the `Ripples` implementation and the equations in Tang et al. [59], we find that processing a USA-scale contact network with 258M vertices would result in 223 billion RRR sets during the first iteration of the algorithm that requires **892 petabytes** of memory to store them when assuming an average of 10 vertices per RRR set. Frontier has 28 petabytes of aggregate storage when considering the entire system's DRAM and SSDs, and therefore, necessitates innovation from this work.

Greedy Hill-Climbing (GHC): The seminal work of Kempe et al. [39] introduced the sequential greedy hill climbing algorithm that is flexible for modeling different diffusion models but computationally expensive. Its complexity is: $O(kn(n+e)\theta)$, where n is the number of vertices in G, e the number of edges, θ the number of simulations of the diffusion process, and k the size of the intervention set. Minutoli et al. [49] used GHC to solve the epidemic control problem on city-scale networks with few millions of nodes; and a subsequent work by Barik et al. [10] used a graph partitioning-based heuristic to scale GHC computations. However, their approaches have scalability limitations for problem instances tackling intervention on US-scale social contact networks. Computationally efficient RIS-based approaches (trade-off between memory and compute) are better suited, if not the only viable option, for problems of the scale considered in this work. By following the established guideline in the literature using a θ value of 10k, GHC would need *10k years* to compute 10k seeds on a US-scale graph using 8k Frontier nodes. This estimate is under the reasonable assumptions that GHC simulations will be generated at the same rate measured on Frontier for RIS-based algorithms and perfect scalability. Therefore, innovation becomes necessary to solve large inputs.

Epidemic Surveillance: Some of the earliest work on epidemic surveillance strategies was by Leskovec et al. [41], who considered the problem of finding a subset $S \subseteq V$ such that testing S would help in detection of an outbreak. Some of the metrics considered include: detection probability, detection penalty (formalized as the outbreak size before detection), and detection time. It was shown by [41] that the detection probability (which is the probability that some node in set S gets infected) is submodular, and the same greedy strategy as for InfMax gives a $(1-1/e)$-approximation. The penalty is not submodular, but an alternative approach of penalty reduction can be shown to be submodular; the same approach works for detection time as well. Thus an InfMax type solution, with suitable changes for each metric gives a solution with good performance.

Models from previous works have been used to directly inform policy decisions. For instance, Reich et al. [52] highlights how the Scenario Modeling Hub (SMH), which integrated over 30 models developed by various teams of the SMH, have been used as a part of an ensemble to inform decision makers on several occasions during the pandemic. Considering the demonstrated impact of prior modeling efforts, the innovations from this work hold significant potential to help inform future policy decisions.

3 Scaling to Billion-Edge Graphs

3.1 The OPIM-C Algorithm

As discussed in §2, the need for massive numbers of simulations arises from the theoretical bounds proved for the *offline* InfMax algorithms. Recently published *online* approaches provide better sample efficiency and come with the advantage of processing information actively as simulation results are produced, and provide ways of stopping at any point of the computation to generate a solution along with an estimation of its quality. However, online algorithms arrive at solutions with the required quality from "below" and, therefore, tend to perform higher numbers of iterations to get to the final solution then offline approaches. In parallel and distributed settings, a higher number of iterations implies more communication than offline approaches. Our distributed multi-GPU implementation is based on the best known serial online Opim-C algorithm proposed by Tang et al. [58] and listed in Algorithm 1. Our analytical model of Opim-C estimated that a USA-scale contact network would require $\approx$ 43 billion random reverse reachable (RRR) sets amounting to 1.7 terabytes of memory requirements in the worst case, under the same assumptions described in §2. This requirement is within the reach of Frontier and allowed us to push the limits to compute higher quality epidemic-solutions by computing larger seed sets (k) and by getting closer to the theoretical approximation bound of $1 - 1/e$ (63% of the unknown optimal solution).

Algorithm 1: Opim-C Algorithm [58]

Input: G, k, ε, δ
Output: S

1 $\theta_0 \leftarrow$ `thetaZero`(n, k, δ)
2 $\theta_{max} \leftarrow$ `thetaMax`$(n, k, \delta, \varepsilon)$
3 $\mathcal{R}_1 \leftarrow$ Generate θ_0 RRR sets from G
4 $\mathcal{R}_2 \leftarrow$ Generate θ_0 RRR sets from G
5 **for** $i \leftarrow 1$ **to** $\lceil \log_2 \frac{\theta_{max}}{\theta_0} \rceil$ **do**
6 $\quad S \leftarrow$ `findSeeds`$(\mathcal{R}_1, k)$
7 $\quad c_1 \leftarrow$ `findCoverage`$(\mathcal{R}_1, S)$
8 $\quad c_2 \leftarrow$ `findCoverage`$(\mathcal{R}_2, S)$
9 $\quad \sigma_l \leftarrow$ `lBound`$(c_2, \delta, n, \mathcal{R}_2)$
10 $\quad \sigma_u \leftarrow$ `uBound`$(c_1, \delta, n, \mathcal{R}_1)$
11 $\quad \alpha \leftarrow \sigma_l / \sigma_u$
12 $\quad$ **if** $\alpha \geq 1 - 1/e - \varepsilon$ **then**
13 $\quad\quad$ **return** S
14 $\quad$ **else**
15 $\quad\quad$ Double the size of $\mathcal{R}_1$ and $\mathcal{R}_2$
16 **return** S

Algorithm 1 starts by determining the minimum (θ_0) and the maximum (θ_{max}) sampling effort to compute the solution S. The algorithm then generates two collections of RRR sets: $\mathcal{R}_1$ and $\mathcal{R}_2$, where $\mathcal{R}_1$ is used to compute candidate solutions (Line 6) while $\mathcal{R}_2$ constitutes a test set used to establish a lower bound on the quality of the current solution (Line 9). At every iteration of the algorithm, the size of $\mathcal{R}_1$ and $\mathcal{R}_2$ is doubled. When the estimated approximation bound (α) exceeds the user-requested bound (Line 12), the algorithm stops and returns the solution S. To the best of our knowledge, *we provide the first parallel and distributed implementation of Opim-C, which can also scale on the current generation of exascale computing systems.* Next, we will summarize the key implementation features for efficient execution on distributed multi-GPU platforms such as OLCF Frontier.

3.2 Parallel OPIM-C Algorithm

Efficient implementation of Algorithm 1 on distributed systems requires optimized implementations for the three fundamental building blocks that occur within each round of the algorithm: *(i)* generating the RRR sets using diffusion simulations (i.e., sampling); *(ii)* seed selection using a max k-coverage algorithm; and *(iii)* computing the coverage of the current solution on $\mathcal{R}_1$ and $\mathcal{R}_2$. Fig. 2 provides a high level overview of `Ripples`'s design. Specifically, the figure illustrates one iteration of the algorithm.

Each compute node (rank) extracts multiple RRR sets (represented by different colors) and generates local histograms counting vertex frequencies within those sets. Ranks then collaboratively compute a globally partitioned histogram, each finding the most frequent vertex within its partition. An all-to-all reduction identifies the globally most frequent vertex, which becomes the next seed. Histograms are updated to remove RRR sets containing the new seed, and the process repeats until all seeds are selected. Between seed selection rounds, if all remaining RRR sets fit on a single node, they are copied to rank 0, and other ranks speculate the execution of the next round of simulations (sampling) while rank 0 finishes seed selection for the remaining seeds, eschewing expensive communication overhead. At the end of each round, all ranks calculate coverage, which is the percentage of RRR sets covered by the selected seeds from that round of seed selection. Below, we describe each process in more detail, including heuristics for efficient sampling.

Efficient Parallel Simulations: A simulation (or a sample) begins by randomly selecting a vertex as the source of a random breadth-first traversal (BPT) using edge weights as the probabilities and one diffusion model. In this work, we build on the fused BPT approach of Neff et al. [50]. Memory requirements for Opim-C are dominated by the two collections of RRR sets $\mathcal{R}_1$ and $\mathcal{R}_2$ (about $1.7TB$ for the US-scale

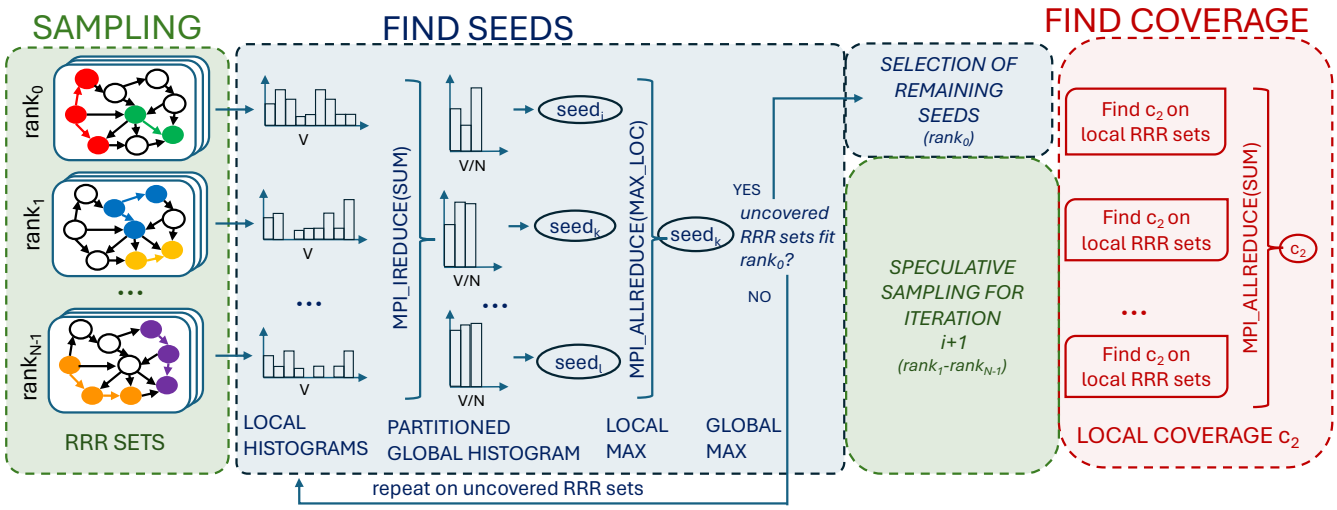

Figure 2: Illustration of the key building blocks of computation in InfMax using the Opim-C algorithm: *Sampling* (N/θ RRR sets per rank, where N is the number of MPI ranks), *Find Seeds* (counting to select k seeds), and *Find Coverage* (evaluating seed set coverage to determine if another round is required). During *Find Seeds*, if the remaining uncovered RRR sets are small enough, they are gathered to rank 0 while other ranks speculatively sample for the next round.

network). We implement our distributed sampling process so that $\mathcal{R}_1$ and $\mathcal{R}_2$ are partitioned evenly among the P machines involved in the execution. The input graph (US-scale network) requires about $60GB$ of memory when using 32-bit floating-point edge weights representing the infection probability. Therefore, to avoid communication during the sampling process, we store separate copies of the input graph on the host and on each GPU to prevent inter-GPU memory operations. We note that the GPU memory available on MI250X is $64GB$ per computing die. Our graph storage leaves limited space on the GPU to store the RRR sets produced during the simulations. Therefore, we adopt a more compact representation of the graph through the quantization of edge probabilities into 16-bit integers. This compact representation reduces the graph storage requirements to $\approx 48GB$. The application's sampling engine dynamically dispatches tasks to CPUs and GPUs participating in the construction of $\mathcal{R}_1$ and $\mathcal{R}_2$ [47]. We adopt the task-fusion techniques proposed in [50] to amortize the cost of offloading tasks to GPUs and avoid redundant work during the simulations.

Scalable Seed Selection: RIS-based InfMax algorithms perform seed selection using a maximum k-cover algorithm, where greedy approaches provide an approximation of $1-1/e$. At each iteration, our algorithm first computes the global histogram of the occurrences of the vertices in the graph over the collection of RRR sets ($\mathcal{R}_1$) and adds the most frequent vertex v to the solution set S. Once v is added to the final solution, all the RRR sets containing the vertex v are

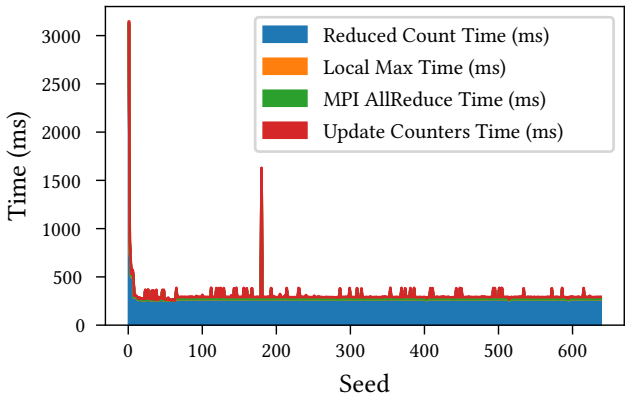

Figure 3: Breakdown of time (max) spent by all ranks for each of the phases of `findSeeds` when $k = 625$, where k is the number of seeds to find. The Local Max Time (orange) is barely visible as it takes much less time.

marked as covered and are prevented from being considered in computing the histogram during the subsequent iterations. The algorithm stops when the solution S is of size k. Since the collection of RRR sets $\mathcal{R}_1$ is partitioned over P ranks, building the global histogram to select each of the seeds in S requires a global reduction of the local histograms on P ranks. We note that the histogram of a USA-scale contact

network is $\approx 1GB$ on each rank, so at full scale we would need to reduce $\approx 9TB$ of data into a single histogram for each selected seed. To scale this step, we introduce a novel scheme where a subset of $|V|/P$ nodes are reduced on each rank, where $|V|$ is the number of vertices in the graph. The top seed is then reduced from locally-owned partitions by an all-reduce step computing the seed with max coverage (through `MPI_MAXLOC`).

Moreover, we leverage the submodular structure of the problem to avoid unnecessary communication. Figure 3 shows the time spent during the early iterations of of `findSeeds`. Here, we see the first several seeds incur an expensive update cost, but quickly shrink to a constant $\approx 350ms$ for inter-node reduction which, at 10k seeds, would require almost 1 hour for each round, which is infeasible considering OPIM-C typically requires iterations to reach the specified quality. To address this communication bottleneck, our implementation monitors the size of the remaining uncovered RRR sets and, as soon as the cumulative size of these sets is small enough to fit in a single machine, we gather all the uncovered sets to rank 0 and complete seed selection from a single machine. Note: this does not restrict the total size of RRR sets to a single machine. To further improve the throughput of the algorithm, we safely avoid updating the histograms once each selected vertex covers only one RRR set. For histogram updates, once a newly selected seed covers one RRR set, all remaining seeds will cover no more than one RRR set. At this point, instead of an expensive update operation, we can pick one vertex from each remaining RRR set as a new seed until $|S| = k$. Each seed is guaranteed to be unique and non-overlapping.

Find Coverage: The `findCoverage` algorithm is used as an estimator of the influence function. This function computes the number of RRR sets that are covered in $\mathcal{R}_1$ and $\mathcal{R}_2$ by the vertices included in the current solution S. The coverage value is used to compute the upper and lower bounds as described by the theoretical analysis in [58]. We note that our implementation distributes $\mathcal{R}_1$ and $\mathcal{R}_2$ over P ranks, and therefore, our implementation requires reduction of the local coverage into a global value on each rank. We fuse the computation of c_1 (Line 7) into the seed-selection step; only c_2 requires explicit computation.

Speculative Execution: Moving the seed selection process to a single machine as soon as the computation fits in memory provides the best application performance. However, this approach alone is wasteful of computing resources at scale, and therefore, we complement it by introducing *speculative sampling*. While rank 0 is busy completing the solution for iteration i of Algorithm 1, the remaining $P-1$ ranks work on expanding $\mathcal{R}_1$ and $\mathcal{R}_2$ that will determine the solution of iteration $i+1$. Our ablation study shows that growing $\mathcal{R}_1$ and $\mathcal{R}_2$ by doubling their sizes does not provide enough work in the early iterations to keep a large system busy for the entire duration of seed selection, and also delays completion in later iterations. To address this load balancing problem, we train a predictor of the sampling effort at each iteration that aims at balancing the execution time of the speculative sampling and the seed selection process, and use the predictor to determine the growth rate of $\mathcal{R}_1$ and $\mathcal{R}_2$. The doubling strategy in Algorithm 1 is suggested by [58] to make the OPIM-C algorithm comparable to the rest of the offline INFMAX algorithms literature. However, due to the online nature of the approach, the doubling is not a requirement to preserve the approach's approximation guarantees. Empirical evidence of the effectiveness of adapting the growth rate of $\mathcal{R}_1$ and $\mathcal{R}_2$ to load balance the overlapping sampling and seed selection phases is presented in §5.

4 Experimental Setup

4.1 Application Used for Experiments

The application is rooted in highly resolved, individual-based, computational epidemiology at the US national level. For this, we have constructed a network capturing all the people of the US and their contacts throughout a typical day. These contact networks are constructed through a methodology using synthetic populations developed and refined by the computational epidemiologists at the University of Virginia over three decades (see, e.g. [2, 7, 14, 16, 21, 28, 64]). These and their associated populations have had broad application in computational epidemiology [3, 20–23, 36, 53], transportation analysis [15, 29, 55], resilience assessments of socio-technical systems [11], evacuation studies [18, 37, 38], and planning scenarios addressing options for renewable energy [45, 46, 57, 60].

The network construction, which is based on an extensive collection of public and commercial data sources, uses a broad range of techniques (e.g., data fusion, data modeling, and machine learning) in conjunction with high-performance computing to construct and validate the following data components for a given region (e.g., the US) at a specific spatial resolution (e.g., block group): *(i)* A population P partitioned into a collection H of households, all with associated demographic and application-specific attributes. *(ii)* A detailed representation of individual residential and non-residential locations captured as a set L. *(iii)* A mapping Λ_R of households to residential locations of L. *(iv)* For each person $p \in P$, an assignment of a daily activity sequence $a_p = (a_{p,1}, a_{p,2}, \ldots, a_{p,k})$ where activities a_i have type (e.g., home, work, school), start time, and duration. *(v)* A mapping Λ that for each person p and each activity $a_{p,i}$ assigns a location $\ell \in L$. The maps Λ and Λ_R coincide when restricted to activities of type home. We remark that assignment of

work activities is done to match the ACS commuting flows in the case of the US [1]. *(vi)* Using the location assignment Λ, a modified Erdős-Rényi random graph model is applied at each location $\ell \in L$ to infer which pairs of simultaneous visitors p and p' come in contact and are connected by an edge $\{p, p'\}$. The union of these location graphs G_ℓ form the contact networks G used in this work.

The resulting synthetic populations statistically match the real populations of each block on their demographic attributes and household structure distributions. The same holds for the county-county commute flows which govern the assignment of people's activities to locations. The network was validated using real-world information that provided details about typical contact rates (such as typical number of interactions), degree distributions, size of a giant component and various metrics related to connectedness, see [21] (SI-Section F), and are consistent with other published observations on contact networks. The US-scale graph used for empirical evaluation has 284, 567, 557 vertices and 7, 544, 209, 380 directed edges, and each edge weight (infection probability) is a simple linear mapping of the contact duration from 0.0 (no duration) to 0.25 (12 hours or more).

4.2 System and Environment for Experiments

We perform all our benchmarking and scaling studies in this work on the US Department of Energy's Frontier located at the Oak Ridge Leadership Computing Facility (OLCF). The full system comprises 74 cabinets each with 128 compute nodes, for a total of 9,408 nodes. The nodes are connected through an HPE Slingshot network with each node having 4 NICs. Each compute node includes a 64-core AMD Optimized 3rd Gen EPYC CPU, 512*GB* of DDR4 memory, and four AMD MI250X GPUs (Figure 4). Each AMD MI250X includes two Graphics Computing Dies (GCDs) with 64*GB* of HBM2E memory. From a user perspective, one can think of GCDs as separate GPUs. The GCDs are connected to each other and to the CPU on the host through Infinity Fabric links. Each of the compute nodes on Frontier has two 1.92*TB* Non-Volatile Memory (NVMe) storage devices. The full system has 75k AMD GPUs and 600k CPUs, amounting to over half-a-billion stream processors (14,080 stream processors per MI250X). We use up to 8,192 nodes in our work.

Our application is developed in C++ using MPI + OpenMP + HIP. All our experiments were compiled using PrgEnv-amd/8.3.3, which includes HIPCC 5.3.22061-e8e78f1a and AMD clang 15.0.0. We use the ROCm runtime version 5.3.0, MPICH runtime version cray-mpich/8.1.23, TRNG version 4.25 for random number generation, and ROCm-Thrust version 5.3.0. The application code and all the benchmarks are compiled with optimization flags `-m64 -O3` and to instruct

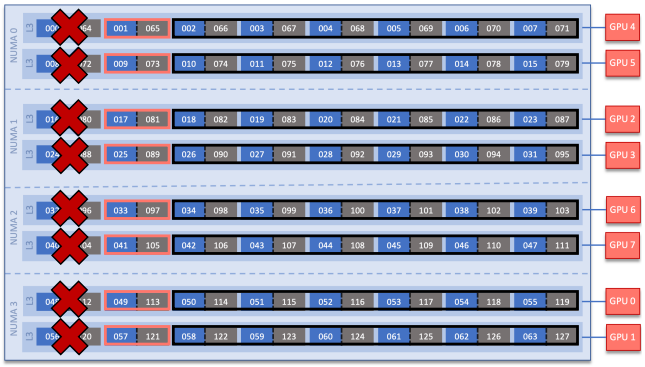

Figure 4: Simplified illustration of a Frontier node (Image credit: OLCF User Guide[6]) with an overlay showing how our application maps on the hardware. Crossed out cores are reserved to perform OS services with the low-noise configuration; the nodes outlined in black execute tasks on the CPU; the cores outlined in salmon are used to perform GPU offloading to the GPU closest to it.

the memory allocator to use interleaving across NUMA domains. Our performance analysis showed that the application is memory bound and, therefore, we avoid using hyperthreading by setting the number of OpenMP threads to 56 and `OMP_PLACES` to `CORES`. We unset `MPICH_OFI_NIC_POLICY` and, for the 8,192 node strong scaling run, we set `FI_CXI_OFLOW_BUF_SIZE` and `FI_CXI_OFLOW_BUF_COUNT` to 4× their default value as advised by the facility.

Experimental Setup: Our application implements custom engines to dynamically dispatch work to CPU cores and GPUs so that *every* processing element on the node contributes to the global solution as suggested in [47]. The engine has two types of threads: *(i)* CPU workers consuming tasks on the CPU cores, and *(ii)* GPU workers offloading tasks to the 8 GPUs. CPU and GPU workers leverage fused probabilistic traversals as described in [50] and we balanced the CPU and GPU task-binding rates of the dynamic scheduler through preliminary tuning studies on the machine as detailed in [50]. CPU workers are organized into teams of 6 and consume tasks at a rate of 14 simulations at a time. Contrarily, GPU workers operate independently and push work on the 8 GPUs at the rate of 64 simulations at a time. All our experiments use the low-noise configuration in Fig. 4. In low-noise mode, the cores that are crossed out in Fig. 4 are reserved to perform operating system tasks while the rest are available for the user application. We map one core per L3 cache region to its corresponding closest GPU. The custom application engine maps the GPU workers to the green-marked cores and the CPU workers to the red-marked cores in Fig. 4.

Although available in `Ripples`, we disabled the use of GPUs for the seed-selection step: (`findSeeds`) in our experiments. Enabling the use of GPUs requires swapping the input graph (USA-scale network) in/out of the GPU memory at each iteration of the algorithm. The additional data movement negates any potential performance boost we would get from leveraging GPUs during seed selection for graphs of the US-scale. Finally, this choice enables the speculative sampling approach that is central to our approach.

We report the results from two ablation studies in §5: *(i)* a benchmarking and detailed performance analysis; and *(ii)* a scalability study. The benchmarking and performance analysis allowed us to evaluate various design options, providing critical insights that informed the development of our approach. Specifically, we considered factors such as: *(i)* different graph representations; *(ii)* parallel pseudo-random number generators using integer or floating point operations; *(iii)* different reduction strategies for the `findSeeds` algorithm; *(iv)* different data movement strategies to aggregate the RRR sets on rank 0; *(v)* the effect of speculative sampling on performance; and *(vi)* the effects of the load balancing between the `findSeeds` algorithm and the simulation process. Our micro-benchmarks use the US-scale contact network as input when the scope of the inference requires a large input. However, for node-level measurements that do not require large inputs, we use different synthetic networks generated by NetworKit [5]: *(i)* Barabasi-Albert [8], *(ii)* LFR [40], *(iii)* RMAT [19], and *(iv)* Watts-Strogatz [62]. We report average execution time from multiple runs of a given feature being tested in the micro-benchmarks.

We present a performance analysis of our approach as implemented in the Ripples framework with strong and weak scaling studies using a USA-scale contact network as the input (§4.2). We used a very simple analytical model to estimate the memory requirements so we could fit the problem instance on 1024/128 nodes for the strong/weak scaling studies, respectively. We prioritized configurations with better approximation guarantees. The algorithm has a constant approximation factor of $1 - 1/e - \epsilon$ and we used setting $\epsilon = .03$ (previous SOTA was $\epsilon = .13$) that provides 60% approximation with probability 99% ($\delta = 0.01$). For the strong scaling study, we use parameters: $k = 10,000$, $\varepsilon = 0.03$, and $\delta = 0.01$. We study the strong scaling of our application from 1,024 nodes up to 8,192 nodes, as our problem instance won't fit on smaller allocations on Frontier. We study the weak scaling of our application relative to the solution set size (S), specifically evaluating performance from 128 nodes ($k = 625$) to 8,192 nodes ($k = 40,000$). The remaining parameters stay unchanged. All our experiments report wall-clock time from when the program starts reading the input file to its completion.

5 Performance Results

5.1 Micro-Benchmarks and Ablation Studies

Optimal Graph Data Structure: Our application represents the input graph in Compressed Sparse Row (CSR) format. To improve the memory behavior of our application, we have experimented with two different memory layouts of the CSR format to store the input. While the first layout stores the graph in an Array of Structures (AoS), the second layout employs a Structure of Arrays (SoA). SoA are generally preferred when using GPUs, while AoS tend to be favored on CPUs. Our application originally used two different representations for CPUs (AoS) and GPUs (SoA). However, this strategy incurs the additional cost of converting the CPU data layout into an SoA. Our micro-benchmarks (Fig. 5) on a collection of synthetically generated graphs showed that the performance of diffusion simulations benefit from using the SoA also on CPUs. Figure 5 plots the difference in execution time between AoS and SoA when varying the scale of the input, with AoS generally performing better from scale 16 onward.

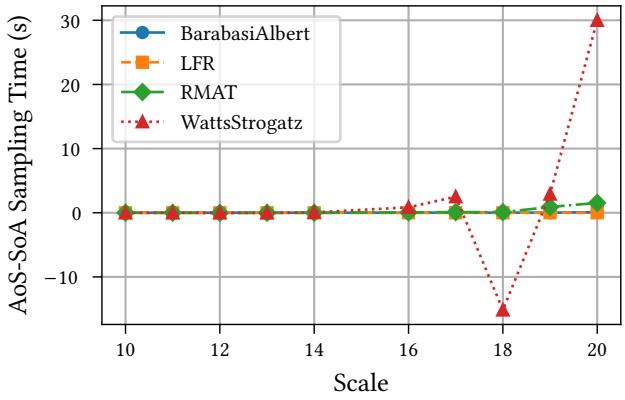

Figure 5: Difference in execution time of the simulations algorithms on synthetic networks between the graph layout in Struct of Arrays (SoA) and Array of Structures (AoS) on CPUs. Times above zero means SoA is better.

Pseudo-Random Number Generation: The performance of diffusion simulations is significantly influenced by the performance of the Pseudo-Random Number Generator (PRNG). We use the TRNG4 library, which provides highly performing PRNGs [13] and is used in the `Ripples` framework [47, 48]. While performant approaches on CPUs are well-established in practice [12], performant PRNGs on GPUs are still evolving. We mentioned in §3 that we quantize the edge weights to conserve memory in order to store the results of the simulations required by InfMax. We compare two approaches to

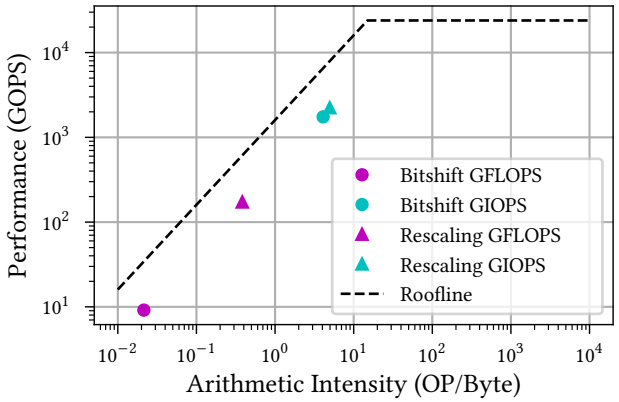

Figure 6: GPU Roofline comparing two methods of random number generation in our simulations: Bitshifting all integer operation, and Rescaling using floating-point operations. We report operation intensity for both integer (IOP) and floating-point (FLOP) operations during simulation. The roofline shows rescaling outperforms bitshifting in both performance metrics.

generate pseudo random numbers. The first approach (Fig. 6 Bitshift) consists of generating a 64-bit number and then taking the high 16 bits, as they are higher quality random numbers. The second approach (Fig. 6 Rescaling) instead uses rescaling through floating-point operations to convert the 64-bit number in a proper range and then casts the value to integer type. Considering that integer operations are generally faster than floating-point operations, one might expect that using the first approach relying solely on integer operations might provide the best performance. Surprisingly, our roofline analysis (Fig. 6) shows that for the rescaling method, we get a substantial increase in performance. In fact, we measure an increase of arithmetic intensity of 488 GIOPS (1.25×) and 163 GFLOPS (19×) when moving from the bitshifting to the rescaling method. We note that our workload is in the "memory-bound" region of the roofline (below the inclined solid line). Therefore, using floating-point units to generate pseudorandom numbers reduces the pressure on the integer units to generate more memory references, and consequentially, improves application performance. The workload and device architecture must be taken into careful consideration, as other PRNG approaches may be more suitable in different scenarios.

`findSeeds` Communication: In §3, we noted `findSeeds` needs to reduce the local histograms storing the frequency of each vertex in the collection of RRR on each machine into a global histogram (Fig. 2). We benchmark two different approaches to solve this problem. The first approach simply performs a reduce operation on a vector of size n, the number of vertices in the input. The second approach block-distributes the global histogram across the P participating machines and performs P reductions of size n/P. Our micro-benchmarks show that the second approach is on average 1.1× time faster than the other. While this gain might seem slim at first, it gets amplified by a factor of $I \cdot k$, where k is the number of seeds selected over the single execution of `findSeeds`, and I is the number of rounds of `findSeeds` that the application will run to find the final solution. We note that our large runs use k values from 10k to 40k, and I is generally between 3 to 6.

Fig. 3 shows the time breakdown of a single iteration of `findSeeds` when $k = 625$. Two important observations emerge from Fig. 3. First, the early iterations of the algorithm take $\approx 7\times$ more than the following iterations, which implies that seeds selected early cover many of RRR sets, and coverage quickly decays. The rate at which the decay happens for our network is justified by the theoretical results of Kempe et al. [39], who showed that the influence function, estimated through coverage, is submodular (has diminishing returns). The second important observation is that communication costs are fixed. At the specific scale of this experiment, each round of the algorithm incurs a cost of $\approx 350ms$ in network communication cost per selected seed. These observations motivated our implementation, which moves the data, necessary to select the remaining seeds, to rank 0 as soon as it is profitable to do so.

Speculative Execution and Load Balancing: Speculative execution alone would cause under utilization of resources, as illustrated in Fig. 7. The top and the bottom figures show the timeline of execution of the first 600s of the application running at on 2048 nodes before (top) and after (bottom) introducing the speculative sampling and predictor to balance the simulations with the selection of seeds. The timeline highlights when the predictor determining the growth rate of $\mathcal{R}_1$ and $\mathcal{R}_2$ successfully hides the sampling process by overlapping it with the `findSeeds` step (green), and when the predictor overshoots and the sampling process lasts more than `findSeeds` (red). In the ideal case, we want to balance the two tasks and we should observe only very thin red or green boxes between iterations.

5.2 Strong and Weak Scaling Studies

The high watermark we achieve in this study is performing InfMax on a US-scale network to compute 40k seeds in 25 minutes while performing 48 million diffusion simulations using 8k nodes of Frontier with 65k GPUs and 458k CPUs. We take 5.75 minutes for 10k seeds using 4k nodes. In contrast to a normal scaling studies, our work is motivated by executing InfMax on a US-scale realistic network that posed

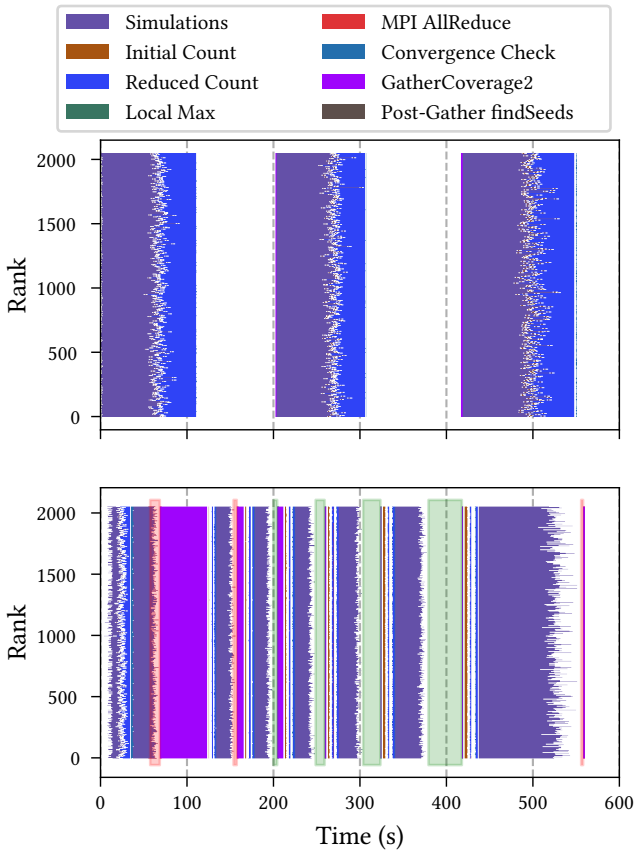

Figure 7: Timeline of performing InfMax on 2048 nodes with $k = 10$k, $\varepsilon = 0.03$, and $\delta = 0.01$, where k is the number of seeds, ε is the approximation factor, and δ is the failure probability. The top timeline shows the first 600 seconds of the initial implementation of `Ripples` after the gather optimization. The bottom plot shows the final implementation with all applied optimizations, including the rank 0 gather and speculative execution, where red boxes indicate seed selection finished before sampling, and green boxes indicate the opposite. The optimized implementation shown in the bottom timeline better utilizes all compute nodes, completing the computation in under 600 seconds, unlike the initial implementation. (Note: This description was generated with the help of Google Gemini 2.0 Flash.)

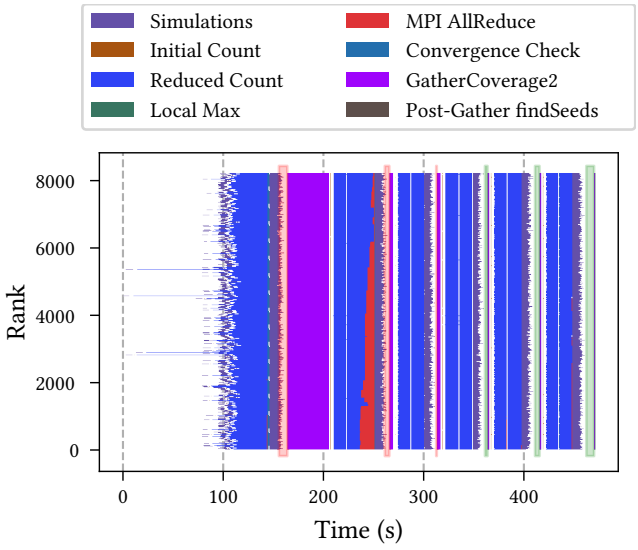

Figure 8: Timeline of performing InfMax on 8192 nodes with $k = 10k$, $\varepsilon = 0.03$, and $\delta = 0.01$, where k is the number of seeds, ε is the approximation factor, and δ is the failure probability. Red boxes indicate seed selection finished before sampling, and green boxes indicate the opposite. Note the large startup delay and time spent on `Reduced Count`, which uses all-to-all communication, compared to previous timelines.

significant challenges at smaller number of nodes due to memory requirements.

We observe linear scalability in our strong scaling studies from 1024 to 4096 nodes. After that we observe a slight inflection at 8192 nodes. There are two factors that cause this inflection: (1) Our job experienced an abnormal startup time (and we report total execution time) with many of the nodes taking up to 100 seconds to start the execution (Fig. 8). Even when accounting for this anomaly, our total execution time would be at around 5 minutes which is close to what the execution time of our 4096 nodes run. (2) In Fig. 9, we extract the time waiting for `MPI_IReduce`, which is the same culprit communication cost described in § 5.1, to reduce counts between all nodes during `findSeeds`. This poor scaling behavior ultimately limits the scalability of the 8k node setup, as shown by the purple line where the total time without the MPI_IReduce overhead is plotted as a reference.

Exactly controlling the amount of work performed by our application is extremely challenging. We chose to double k at each step of the study because it is in a linear relationship with the *initial* number of simulations performed. However, the speculation and the predictor vary the total number of simulation performed at each configuration. Therefore, to account for this variability we normalized our execution times by the total number of simulations performed. The weak scaling study (Fig. 10) shows almost *perfect* weak scaling for the simulation phase, while the `findSeeds` algorithm show good weak scaling up to 2048 nodes. The degradation is caused by the larger number of simulations between 4096 and 8192 preventing the use of our heuristic that speeds up the final phases of seed selection in the rest of the runs.

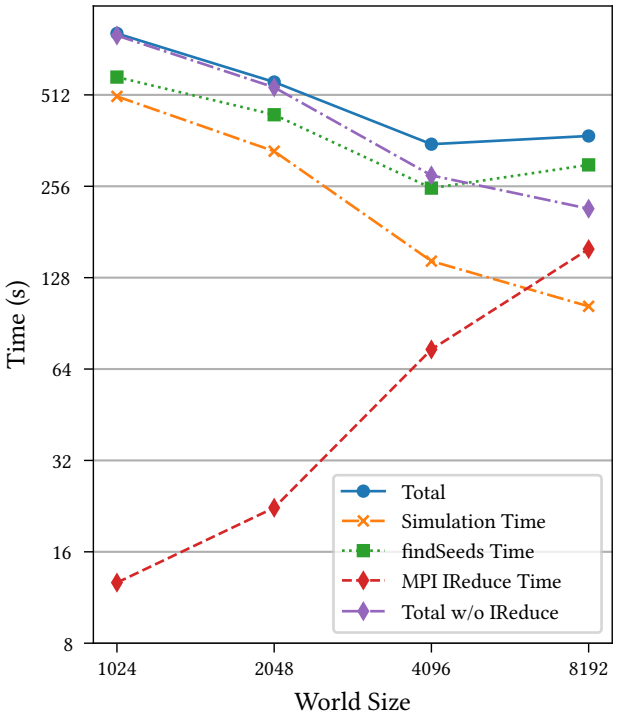

Figure 9: Strong scaling from 1024 nodes up to 8192 nodes on the USA-scale network. We fix parameters to: $k = 10$k, $\varepsilon = 0.03$, and $\delta = 0.01$, where k is the number of seeds, ε is the approximation factor, and δ is the failure probability. This problem instance does not fit on fewer than 1,024 Frontier nodes. Due to ranks initializing at different times, we mark the start of the program as the median start time of all nodes. Strong scaling reverses after 4,096 nodes, indicated by the increasing cost of `MPI_IReduce`.

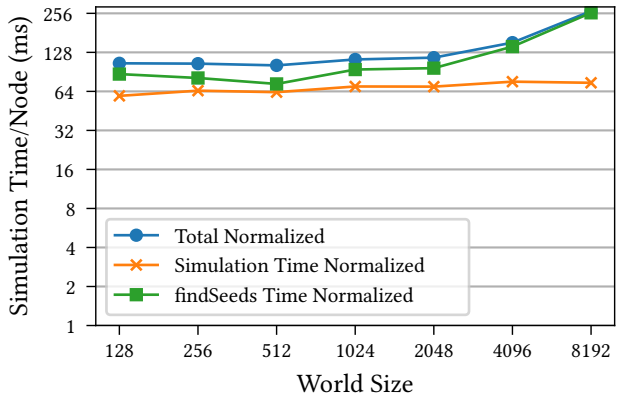

Figure 10: Weak scaling normalized with the number of simulations and nodes to highlight its impact on sampling and seed selection. While weak scaling is nearly perfect, it encounters a similar scaling issue with `MPI_IReduce` within `findSeeds` at 8,192 nodes, as seen in Fig. 9

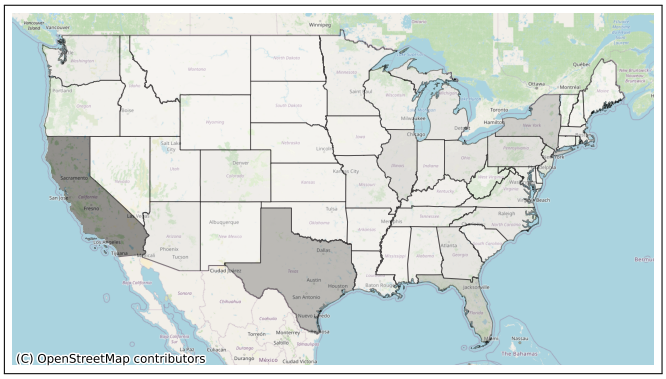

Figure 11: A spatial breakdown of the 10k seeds by state depicted using a grayscale gradient. Seed vertices correspond to persons in the synthetic population for which the US network was generated, and accumulation is done through person's residence state. California has the largest number of seeds, accounting for nearly 12.7% of the 10k seeds.

More iterations before reducing to a single node for the remaining seeds in seed selection leads to more `MPI_IReduce` rounds, making worse a similar issue shown earlier with strong scaling.

6 Implications

Influence Maximization: This work represents the first execution of InfMax on graphs that are comparable in size with the US population and over two orders-of-magnitude larger than the previous state-of-the-art. We show the results from a run with 10k seeds in Fig. 11. Our results were enabled by a new algorithm and many optimization techniques aiming at improving scaling. We believe that this work will advance both the theory and practice of graph analytics and will motivate algorithmic innovations and novel applications beyond influence maximization. Given the scalability of the codes, we can now consider deploying the software for real world applications that are sensitive to response times.

Implications for Pandemic Planning and Response: The role of major hubs as sources of importation of infectious diseases into the US has been well documented for H1N1 and more recently COVID pandemics, see, e.g., the CDC report [54]. Detailed analysis of the role of large hubs on disease propagation is undertaken in a number of papers including [4, 17, 25, 42] All of the pandemics in the last 50 years or more have arrived to the US via individuals coming

to US, e.g., [44, 63]. This can include visitors, or residents who might have gone and visited other countries. Today, most of these initial cases arrive at one of the international airports. Informally, importation refers to the arrival of the first few cases of a pandemic to a region that would lead to a sustained transmission in the region. Importation time is a critical measure for planning a more efficient response to the pandemic. It enables policy makers to start preparation and, as discussed in subsequent sections, put a testing and quarantine regimen in place to slow the importation time.

If cases arrive at one of the large international airports, an important policy question is: *How fast will pandemic spread to other parts of the country?* One way to capture the speed of spread is to use a quantity called "importation time". Informally speaking, for a region R, the importation time $imp(R)$ is the expected time for the number of infections in the region R to reach a threshold quantity R_{th}, assuming that initial infections reach the international airports at time 0. Our goal is to understand the distribution of expected importations times in all the states in the US (i.e. the regions R corresponding to the states of US). Threshold value R_{th} is used to capture the fact that once this threshold value is reached, then the epidemic can take off in that region R; we assume R_{th} to be defined in terms of a percentage of the population in R.

This notion of importation is closely related to the notion of "detection penalty" in a region R, studied by [41]— this is the number of infections in R by the time the outbreak is detected in R; Leskovec et al. [41] show that an alternative objective, namely reduction in penalty, is submodular, with similar structure as the InfMax problem. Our parallel algorithms can be used to study importation times for pandemic planning. For this, we would use both optimal strategies and random strategies for seed selection to maximize the number of infections. These numbers provide appropriate bounds (worst case and expected) for pandemic to take hold in the US.

Surveillance using different kinds of testing strategies is a fundamental component of pandemic response for detecting an outbreak, and identifying importations at incoming ports is one way to slow down pandemic spread in the early period. Testing is expensive to implement, in terms of costs of materials and staffing. For instance, testing on a large scale for accurate estimation of disease prevalence has only been done in a small number of regions during the COVID pandemic, e.g., [56, 65], due to its expense. Different kinds of metrics of public health interest are probability of detection of an outbreak, estimation of prevalence, and early detection. InfMax based methods can be highly valuable in finding solutions to optimize these metrics, which can help us understand how much testing is needed (dependent on test error rates) to slow down the spread. However, spread simulations at nation-scale and perhaps global-scale will be necessary to better prepare and plan for future pandemics—which is where our work here on scaling InfMax on the exascale OLCF Frontier platform is an important step.

Acknowledgments

This research is based upon work supported by the U.S. Department of Energy (DOE) through the Exascale Computing Project (17-SC-20-SC) (ExaGraph) at the Pacific Northwest National Laboratory (PNNL) the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), through the Advanced Graphic Intelligence Logical Computing Environment (AGILE) research program, contract number 77740, and Laboratory Directed Research and Development funds at PNNL. This work was supported in part by the following grants: University of Virginia Strategic Investment Fund (Award Number SIF160), National Science Foundation Grants CCF-1918656 (Expeditions), OAC-1916805 (CINES), IIS-1955797, VDH Grant PV-BII VDH COVID-19 Modeling Program VDH-21-501-0135, DTRA subcontract/ARA S-D00189-15-TO-01-UVA, NIH 2R01GM109718-07, NSF RAPID CNS-2028004, NSF RAPID OAC-2027541, US Centers for Disease Control and Prevention 75D30119C05935 and CDC MIND cooperative agreement U01CK000589. Washington State University: NSF CCF 1919122 and CCF 2316160. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the ODNI, IARPA, or the U.S. Government. We sincerely thank Oak Ridge Leadership Computing Facility for enabling this work.

References

[1] ACS 2024. 2016-2020 5-Year ACS Commuting Flows. https://www.census.gov/data/tables/2020/demo/metro-micro/commuting-flows-2020.html Last accessed: 05 March, 2024.

[2] A Adiga, R. Beckman, K. Bisset, J. Chen, Y. Chung Baek, S. Eubank, S. Gupta, M. Khan, C. Kuhlman, E. Lofgren, B. Lewis, A. Marathe, M. Marathe, H. Mortveit, E. Nordberg, C. Rivers, P. Stretz, S. Swarup, A. Wilson, and D. Xie. 2015. Synthetic Populations for Epidemic Modeling. International conference on computational social sciences (ICCSS), June 8-11, Helsinki, Finland, 2015.

[3] Aniruddha Adiga, Jiangzhuo Chen, Madhav Marathe, Henning Mortveit, Srinivasan Venkatramanan, and Anil Vullikanti. 2020. Data-Driven Modeling for Different Stages of Pandemic Response. *Journal of the Indian Institute of Science* 100 (2020), 901–915. doi:10.1007/s41745-020-00206-0

[4] Lijing Adiga, Aniruddha abd Wang, Adam Sadilek, Ashish Tendulkar, Srinivasan Venkatramanan, Anil Vullikanti, Gaurav Aggarwal, Alok Talekar, Xue Ben, Jiangzhuo Chen, Bryan Lewis, Samarth Swarup, Milind Tambe, and Madhav Marathe. 2020. Interplay of global multi-scale human mobility, social distancing, government interventions, and COVID-19 dynamics. doi:10.1101/2020.06.05.20123760 medRxiv preprint.

[5] Eugenio Angriman, Alexander van der Grinten, Michael Hamann, Henning Meyerhenke, and Manuel Penschuck. 2022. Algorithms for

Large-Scale Network Analysis and the NetworKit Toolkit. In *Algorithms for Big Data - DFG Priority Program 1736*, Hannah Bast, Claudius Korzen, Ulrich Meyer, and Manuel Penschuck (Eds.). Lecture Notes in Computer Science, Vol. 13201. Springer, 3–20. doi:10.1007/978-3-031-21534-6_1

[6] Leadership Class Computing Facility at Oak Ridge National Laboratory. [n. d.]. Frontier User guide. https://docs.olcf.ornl.gov/systems/frontier_user_guide.html

[7] K. Atkins, C. Barrett, R. Beckman, K. Bisset, S. Eubank, A. Marathe, M. Marathe, H. S. Mortveit, P. Stretz, and Vullikanti A. 2006. Synthetic data products for societal infrastructures and proto populations: Data set 1.0. NDSSL Technical Report No. 06-006.

[8] Albert-László Barabási and Réka Albert. 1999. Emergence of scaling in random networks. *science* 286, 5439 (1999), 509–512.

[9] Reet Barik, Wade Cappa, S.M. Ferdous, Marco Minutoli, Mahantesh Halappanavar, and Ananth Kalyanaraman. 2025. GreediRIS: Scalable influence maximization using distributed streaming maximum cover. *J. Parallel and Distrib. Comput.* 198 (2025), 105037. doi:10.1016/j.jpdc.2025.105037

[10] Reet Barik, Marco Minutoli, Mahantesh Halappanavar, and Ananth Kalyanaraman. 2022. IMpart: A Partitioning-based Parallel Approach to Accelerate Influence Maximization. In *2022 IEEE 29th International Conference on High Performance Computing, Data, and Analytics (HiPC)*. IEEE, 125–134.

[11] Christopher Barrett, Keith Bisset, Shridhar Chandan, Jiangzhuo Chen, Youngyun Chungbaek, Stephen Eubank, Yaman Evrenosoglu, Bryan Lewis, Kristian Lum, Achla Marathe, Madhav Marathe, Henning Mortveit, Nidhi Parikh, Arun Phadke, Jeff Reed, Caitlin Rivers, Sudip Saha, Paula Stretz, Samarth Swarup, James Thorp, Anil Vullikanti, and Dawen Xie. 2013. Planning and response in the aftermath of a large crisis: an agent-based informatics framework. In *Proceedings of the 2013 Winter Simulation Conference*. 1515–1526. doi:10.1109/wsc.2013.6721535

[12] Heiko Bauke. [n. d.]. Tina's Random Number Generator Library. https://www.numbercrunch.de/trng/trng.pdf

[13] Heiko Bauke and Stephan Mertens. 2007. Random numbers for large-scale distributed Monte Carlo simulations. *Phys. Rev. E* 75 (6 2007), 066701. Issue 6. doi:10.1103/PhysRevE.75.066701

[14] Richard J. Beckman, Keith A. Baggerly, and Michael D. McKay. 1996. Creating synthetic baseline populations. *Transportation Research Part A: Policy and Practice* 30, 6 (1996), 415–429. doi:10.1016/0965-8564(96)00004-3

[15] R. J. Beckman, C. L. Barrett, K. B. Berkbigler, K. R. Burris, B. W. Bush, S. D. Hull, J. M. Hurford, P. Medvick, D. A. Kubicek, M. Marathe, J. D. Morgeson, K. Nagel, D. J. Roberts, L. L. Smith, M. J. Stein, P. E. Stretz, S. J. Sydoriak, K. Cervenka, and R. Donnelly. 1997. Transportation ANalysis SIMulation System TRANSIMS - The Dallas-Ft. Worth Case Study. Los Alamos National Laboratory technical report: LA-UR: 97–4502.

[16] Parantapa Bhattacharya, Jiangzhuo Chen, Stefan Hoops, Machi Dustin, Bryan Lewis, Srinivasan Venkatramanan, Mandy L. Wilson, Brian Klahn, Aniruddha Adiga, Benjamin Hurt, Joseph Outten, Abhijin Adiga, Andrew Warren, Hannah Baek, Przemyslaw Porebski, Achla Marathe, Dawen Xie, Samarth Swarup, Anil Vullikanti, Henning Mortveit, Stephen Eubank, Christopher L. Barrett, and Madhav Marathe. 2023. Data-driven scalable pipeline using national agent-based models for real-time pandemic response and decision support. *The International Journal of High Performance Computing Applications* 37, 1 (2023), 4–27. doi:10.1177/10943420221127034 arXiv:https://doi.org/10.1177/10943420221127034 Gordon Bell Award finalist. Online version.

[17] Dirk Brockmann and Dirk Helbing. 2013. The Hidden Geometry of Complex, Network-Driven Contagion Phenomena. *Science* 342, 6164 (2013), 1337–1342. doi:10.1126/science.1245200 arXiv:https://www.science.org/doi/pdf/10.1126/science.1245200

[18] Anna E. Brower, Balaji Ramesh, Kazi Ashik Islam, Henning S. Mortveit, Stefan Hoops, Anil Vullikanti, Madhav V. Marathe, Benjamin Zaitchik, Julia M. Gohlke, and Samarth Swarup. 2023. Augmenting the Social Vulnerability Index using an agent-based simulation of Hurricane Harvey. *Computers, Environment and Urban Systems* 105 (2023), 102020. doi:10.1016/j.compenvurbsys.2023.102020

[19] Deepayan Chakrabarti, Yiping Zhan, and Christos Faloutsos. 2004. R-MAT: A recursive model for graph mining. In *Proceedings of the 2004 SIAM International Conference on Data Mining*. SIAM, 442–446.

[20] Jiangzhuo Chen, Stefan Hoops, Achla Marathe, Henning Mortveit, Bryan Lewis, Srinivasan Venkatramanan, Arash Haddadan, Parantapa Bhattacharya, Abhijin Adiga, Anil Vullikanti, Aravind Srinivasan, Mandy L. Wilson, Gal Ehrlich, Maier Fenster, Stephen Eubank, Christopher Barrett, and Madhav Marathe. 2022. Effective Social Network-Based Allocation of COVID-19 Vaccines. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Washington DC, USA) *(KDD '22)*. Association for Computing Machinery, New York, NY, USA, 4675–4683. doi:10.1145/3534678.3542673

[21] Jiangzhuo Chen, Stefan Hoops, Henning S. Mortveit, Bryan L. Lewis, Dustin Machi, Parantapa Bhattacharya, Srinivasan Venkatramanan, Mandy L. Wilson, Chris L. Barrett, and Madhav V. Marathe. 2025. EpiHiper – A High Performance Computational Modeling Framework to Support Epidemic Science. *PNAS Nexus* (2025). doi:10.1093/pnasnexus/pgae557

[22] Jiangzhuo Chen, Anil Vullikanti, Stefan Hoops, Henning Mortveit, Bryan Lewis, Srinivasan Venkatramanan, Wen You, Stephen Eubank, Madhav Marathe, Chris Barrett, and Achla Marathe. 2020. Medical Costs of Keeping the US Economy Open During COVID–19. *Nature Scientific Reports* 10 (2020), 18422. Issue 1. doi:10.1038/s41598-020-75280-6

[23] Jiangzhuo Chen, Anil Vullikanti, Joost Santos, Srinivasan Venkatramanan, Stefan Hoops, Henning Mortveit, Bryan Lewis, Wen You, Stephen Eubank, Madhav Marathe, Chris Barrett, and Achla Marathe. 2021. Epidemiological and economic impact of COVID-19 in the US. *Scientific Reports* 11, 1 (2021), 20451. doi:10.1038/s41598-021-99712-z

[24] Xinyu Chen, Marco Minutoli, Jiannan Tian, Mahantesh Halappanavar, Ananth Kalyanaraman, and Dingwen Tao. 2023. HBMax: Optimizing Memory Efficiency for Parallel Influence Maximization on Multicore Architectures. In *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques* (Chicago, Illinois) *(PACT '22)*. Association for Computing Machinery, New York, NY, USA, 412–425. doi:10.1145/3559009.3569647

[25] Vittoria Colizza, Alain Barrat, Marc Barthélemy, and Alessandro Vespignani. 2006. The role of the airline transportation network in the prediction and predictability of global epidemics. *Proceedings of the National Academy of Sciences* 103, 7 (2006), 2015–2020. doi:10.1073/pnas.0510525103 arXiv:https://www.pnas.org/doi/pdf/10.1073/pnas.0510525103

[26] Pedro Domingos and Matt Richardson. 2001. Mining the network value of customers. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. 57–66.

[27] Stephen Eubank. 2005. Network Based Models of Infectious Disease Spread. *Japanese Journal of Infectious Diseases* 58, 6 (2005), S9–S13. doi:10.7883/yoken.JJID.2005.S9

[28] S. Eubank, V. S. A. Kumar, M. Marathe, Aravind Srinivasan, and Nan Wang. 2006. Structure of Social Contact Networks and Their Impact on Epidemics. In *Discrete Methods in Epidemiology (DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 70)*.

doi:10.1090/dimacs/070

[29] Federal Highway Administration. [n. d.]. TRANSIMS. Online. https://www.fhwa.dot.gov/planning/tmip/resources/transims/

[30] Gökhan Göktürk and Kamer Kaya. 2021. Boosting Parallel Influence-Maximization Kernels for Undirected Networks With Fusing and Vectorization. *IEEE Trans. Parallel Distributed Syst.* 32, 5 (2021), 1001–1013. doi:10.1109/TPDS.2020.3038376

[31] Gökhan Göktürk and Kamer Kaya. 2022. Fast and High-Quality Influence Maximization on Multiple GPUs. In *2022 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2022, Lyon, France, May 30 - June 3, 2022*. IEEE, 908–918. doi:10.1109/IPDPS53621.2022.00093

[32] Gökhan Göktürk and Kamer Kaya. 2024. Fast and error-adaptive influence maximization based on Count-Distinct sketches. *Inf. Sci.* 655 (2024), 119875. doi:10.1016/J.INS.2023.119875

[33] Gökhan Göktürk and Kamer Kaya. 2022. Fast and High-Quality Influence Maximization on Multiple GPUs. In *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 908–918. doi:10.1109/IPDPS53621.2022.00093

[34] Mahantesh Halappanavar, Arun V. Sathanur, and Apurba K. Nandi. 2016. Accelerating the mining of influential nodes in complex networks through community detection. In *Proceedings of the ACM International Conference on Computing Frontiers* (Como, Italy) *(CF '16)*. Association for Computing Machinery, New York, NY, USA, 64–71. doi:10.1145/2903150.2903181

[35] M Elizabeth Halloran, Neil M Ferguson, Stephen Eubank, Ira M Longini, Jr, Derek A T Cummings, Bryan Lewis, Shufu Xu, Christophe Fraser, Anil Vullikanti, Timothy C Germann, Diane Wagener, Richard Beckman, Kai Kadau, Chris Barrett, Catherine A Macken, Donald S Burke, and Philip Cooley. 2008. Modeling targeted layered containment of an influenza pandemic in the United States. *Proc. Natl. Acad. Sci. U. S. A.* 105, 12 (March 2008), 4639–4644.

[36] Emily Howerton, Lucie Contamin, Luke C. Mullany, Michelle Qin, Nicholas G. Reich, Samantha Bents, Rebecca K. Borchering, Sung-mok Jung, Sara L. Loo, Claire P. Smith, John Levander, Jessica Kerr, J. Espino, Willem G. van Panhuis, Harry Hochheiser, Marta Galanti, Teresa Yamana, Sen Pei, Jeffrey Shaman, Kaitlin Rainwater-Lovett, Matt Kinsey, Kate Tallaksen, Shelby Wilson, Lauren Shin, Joseph C. Lemaitre, Joshua Kaminsky, Juan Dent Hulse, Elizabeth C. Lee, Clifton D. McKee, Alison Hill, Dean Karlen, Matteo Chinazzi, Jessica T. Davis, Kunpeng Mu, Xinyue Xiong, Ana Pastore y Piontti, Alessandro Vespignani, Erik T. Rosenstrom, Julie S. Ivy, Maria E. Mayorga, Julie L. Swann, Guido España, Sean Cavany, Sean Moore, Alex Perkins, Thomas Hladish, Alexander Pillai, Kok Ben Toh, Ira Longini, Shi Chen, Rajib Paul, Daniel Janies, Jean-Claude Thill, Anass Bouchnita, Kaiming Bi, Michael Lachmann, Spencer J. Fox, Lauren Ancel Meyers, Ajitesh Srivastava, Przemyslaw Porebski, Srini Venkatramanan, Aniruddha Adiga, Bryan Lewis, Brian Klahn, Joseph Outten, Benjamin Hurt, Jiangzhuo Chen, Henning Mortveit, Amanda Wilson, Madhav Marathe, Stefan Hoops, Parantapa Bhattacharya, Dustin Machi, Betsy L. Cadwell, Jessica M. Healy, Rachel B. Slayton, Michael A. Johansson, Matthew Biggerstaff, Shaun Truelove, Michael C. Runge, Katriona Shea, Cécile Viboud, and Justin Lessler. 2023. Evaluation of the US COVID-19 Scenario Modeling Hub for informing pandemic response under uncertainty. *Nature Communications* 14 (2023). Issue 1. doi:10.1038/s41467-023-42680-x

[37] Kazi Ashik Islam, Da Qi Chen, Madhav Marathe, Henning Mortveit, Samarth Swarup, and Anil Vullikanti. 2023. Simulation-Assisted Optimization for Large-Scale Evacuation Planning with Congestion-Dependent Delays. In *IJCAI '23: Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. 5359–5367. doi:10.24963/ijcai.2023/595 Article No.: 595.

[38] Kazi Ashik Islam, Da Qi Chen, Madhav Marathe, Henning Mortveit, Samarth Swarup, and Anil Vullikanti. 2023. Towards Optimal and Scalable Evacuation Planning Using Data-driven Agent Based Models. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*. 2397–2399.

[39] David Kempe, Jon M. Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24 - 27, 2003*. 137–146. doi:10.1145/956750.956769

[40] Andrea Lancichinetti and Santo Fortunato. 2009. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical review E* 80, 1 (2009), 016118.

[41] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. 2007. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. 420–429.

[42] Chi Li, Linhao Yu, Jianfeng Mao, Wei Cong, Zibin Pan, Yuhao Du, and Lianmin Zhang. 2024. How did international air transport networks influence the spread of COVID-19? A spatial and temporal modeling perspective. *Transportation Research Part C: Emerging Technologies* 165 (2024), 104730. doi:10.1016/j.trc.2024.104730

[43] Madhav Marathe and Anil Kumar S. Vullikanti. 2013. Computational epidemiology. *Commun. ACM* 56, 7 (July 2013), 88–96. doi:10.1145/2483852.2483871

[44] Tigist F Menkir, Taylor Chin, James A Hay, Erik D Surface, Pablo M De Salazar, Caroline O Buckee, Alexander Watts, Kamran Khan, Ryan Sherbo, Ada WC Yan, et al. 2021. Estimating internationally imported cases during the early COVID-19 pandemic. *Nature communications* 12, 1 (2021), 311.

[45] Rounak Meyur, Swapna Thorve, Madhav Marathe, Anil Vullikanti, Samarth Swarup, and Henning Mortveit. 2022. A Reliability-aware Distributed Framework to Schedule Residential Charging of Electric Vehicles. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, Lud De Raedt (Ed.). International Joint Conferences on Artificial Intelligence Organization, 5115–5121. doi:10.24963/ijcai.2022/710 AI for Good.

[46] Rounak Meyur, Anil Vullikanti, Samarth Swarup, Henning S. Mortveit, Virgilio Centeno, Arun Phadke, H. Vincent Poor, and Madhav V. Marathe. 2022. Ensembles of realistic power distribution networks. *Proceedings of the National Academy of Sciences* 119 (2022), e2205772119. Issue 42. https://www.pnas.org/doi/full/10.1073/pnas.2205772119

[47] Marco Minutoli, Maurizio Drocco, Mahantesh Halappanavar, Antonino Tumeo, and Ananth Kalyanaraman. 2020. cuRipples: influence maximization on multi-GPU systems. In *ICS '20: 2020 International Conference on Supercomputing, Barcelona Spain, June, 2020*. ACM, 12:1–12:11. doi:10.1145/3392717.3392750

[48] Marco Minutoli, Mahantesh Halappanavar, Ananth Kalyanaraman, Arun V. Sathanur, Ryan S. McClure, and Jason E. McDermott. 2019. Fast and Scalable Implementations of Influence Maximization Algorithms. In *2019 IEEE International Conference on Cluster Computing, CLUSTER 2019, Albuquerque, NM, USA, September 23-26, 2019*. IEEE, 1–12. doi:10.1109/CLUSTER.2019.8890991

[49] Marco Minutoli, Prathyush Sambaturu, Mahantesh Halappanavar, Antonino Tumeo, Ananth Kalyanaraman, and Anil Vullikanti. 2020. Preempt: Scalable epidemic interventions using submodular optimization on multi-GPU systems. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2020, Virtual Event / Atlanta, Georgia, USA, November 9-19, 2020*. IEEE/ACM, 55. doi:10.1109/SC41405.2020.00059

[50] Reece Neff, Mostafa Eghbali Zarch, Marco Minutoli, Mahantesh Halappanavar, Antonino Tumeo, Ananth Kalyanaraman, and Michela Becchi. 2024. FuseIM: Fusing Probabilistic Traversals for Influence

Maximization on Exascale Systems. In *Proceedings of the 38th ACM International Conference on Supercomputing*. 38–49.

[51] Hung T. Nguyen, My T. Thai, and Thang N. Dinh. 2017. A Billion-Scale Approximation Algorithm for Maximizing Benefit in Viral Marketing. *IEEE/ACM Trans. Netw.* 25, 4 (8 2017), 2419–2429. doi:10.1109/TNET.2017.2691544

[52] Nicholas G. Reich, Justin Lessler, Sebastian Funk, Cecile Viboud, Alessandro Vespignani, Ryan J. Tibshirani, Katriona Shea, Melanie Schienle, Michael C. Runge, Roni Rosenfeld, Evan L. Ray, Rene Niehus, Helen C. Johnson, Michael A. Johansson, Harry Hochheiser, Lauren Gardner, Johannes Bracher, Rebecca K. Borchering, and Matthew Biggerstaff. 2022. Collaborative Hubs: Making the Most of Predictive Epidemic Modeling. *American Journal of Public Health* 112, 6 (2022), 839–842. doi:10.2105/AJPH.2022.306831 arXiv:https://doi.org/10.2105/AJPH.2022.306831 PMID: 35420897.

[53] Sami Saliba, Faraz Dadgostari, Stefan Hoops, Henning S Mortveit, and Samarth Swarup. 2023. Active Sensing for Epidemic State Estimation Using ABM-guided Machine Learning. In *Proceedings of the Multi-agent-based Simulation (MABS) Workshop*. https://mabsworkshop.github.io/articles/salibaEtAl2023.pdf Best Paper Award.

[54] Anne Schuchat. 2020. Public Health Response to the Initiation and Spread of Pandemic COVID-19 in the United States. 551–556 pages. Issue 18. CDC report; was posted online as an MMWR Early Release.

[55] L Smith, R Beckman, K Baggerly, D. Anson, and M Williams. 1995. TRANSIMS: Transportation analysis and simulation system. doi:10.2172/88648 Los Alamos National Laboratory technical report: LA–UR: 95-1641.

[56] Neeraj Sood, Paul Simon, Peggy Ebner, Daniel Eichner, Jeffrey Reynolds, Eran Bendavid, and Jay Bhattacharya. 2020. Seroprevalence of SARS-CoV-2–specific antibodies among adults in Los Angeles County, California, on April 10-11, 2020. *JAMA* 323, 23 (2020), 2425–2427.

[57] Thorve Swapna, Mortveit Henning, Vullikanti Anil, Marathe Madhav, and Swarup Samarth. 2024. Assessing fairness of residential dynamic pricing for electricity using active learning with agent-based simulation. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multi-Agent Systems*. Status: accepted.

[58] Jing Tang, Xueyan Tang, Xiaokui Xiao, and Junsong Yuan. 2018. Online Processing Algorithms for Influence Maximization. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, Gautam Das, Christopher M. Jermaine, and Philip A. Bernstein (Eds.). ACM, 991–1005. doi:10.1145/3183713.3183749

[59] Youze Tang, Yanchen Shi, and Xiaokui Xiao. 2015. Influence Maximization in Near-Linear Time: A Martingale Approach. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*. 1539–1554. doi:10.1145/2723372.2723734

[60] Swapna Thorve, Young Yun Baek, Samarth Swarup, Henning Mortveit, Achla Marathe, Anil Vullikanti, and Madhav Marathe. 2023. High resolution synthetic residential energy use profiles for the United States. *Scientific Data* 10, 1 (2023), 76. doi:10.1038/s41597-022-01914-1

[61] Letong Wang, Xiangyun Ding, Yan Gu, and Yihan Sun. 2023. Fast and Space-Efficient Parallel Algorithms for Influence Maximization. *Proc. VLDB Endow.* 17, 3 (11 2023), 400–413. doi:10.14778/3632093.3632104

[62] Duncan J Watts and Steven H Strogatz. 1998. Collective dynamics of 'small-world'networks. *nature* 393, 6684 (1998), 440–442.

[63] Chad R Wells, Pratha Sah, Seyed M Moghadas, Abhishek Pandey, Affan Shoukat, Yaning Wang, Zheng Wang, Lauren A Meyers, Burton H Singer, and Alison P Galvani. 2020. Impact of international travel and border control measures on the global spread of the novel 2019 coronavirus outbreak. *Proceedings of the National Academy of Sciences* 117, 13 (2020), 7504–7509.

[64] H. Xia, J. Chen, M.V. Marathe, H.S. Mortveit, and M. Salathe. 2013. Synthesizing Social Proximity Networks by Combining Subjective Surveys with Digital Traces. In *2013 IEEE 16th International Conference on Computational Science and Engineering*. 188–195. doi:10.1109/CSE.2013.38

[65] Wenjuan Zhang, John Paul Govindavari, Brian D Davis, Stephanie S Chen, Jong Taek Kim, Jianbo Song, Jean Lopategui, Jasmine T Plummer, and Eric Vail. 2020. Analysis of genomic characteristics and transmission routes of patients with confirmed SARS-CoV-2 in Southern California during the early stage of the US COVID-19 pandemic. *JAMA network open* 3, 10 (2020), e2024191.