Ripples: Exascale-Enabled Influence Maximization on Massive Social Networks with Implications to Pandemic Planning

Marco Minutoli*, Reece Neff†*, Naw Sattar§, Hao Lu§, John Feo*, Henning Mortveit‡, Anil Vullikanti‡, Dawen Xie‡, Amanda Wilson‡, Gregor von Laszewski‡, Parantapa Bhattacharya‡, Ananth Kalyanaraman¶*, Michela Becchi†, Madhav Marathe‡, Mahantesh Halappanavar*
*Pacific Northwest National Laboratory, Richland, WA, USA
†North Carolina State University, Raleigh, NC, USA
‡University of Virginia, Charlottesville, VA, USA
§Oak Ridge National Laboratory, Oak Ridge, TN, USA
¶Washington State University, Pullman, WA, USA

***Abstract*—We study exa-scale parallel algorithms for Influence maximization in massive realistic socio-technical networks. We employ novel techniques to enable efficient scaling on up to 8k nodes of OLCF Frontier, with 65k AMD GPUs and 524k AMD CPUs. To test the efficacy of our methods we use a realistic social contact network of the US with about 300 million nodes and about 15 Billion edges. We demonstrate the up to 1000× speedup relative to two nodes of Frontier. The best known methods for influence maximization have been demonstrated on graphs with at least one order of magnitude smaller. We briefly discuss how such methods can be used to address important problems arising in the context of pandemic planning and prediction. Max 150 words.**

I. Justification for ACM Gordon Bell Prize (50 word max)

For the first-of-its-kind discovery of 40k most influential seeds in a US national-scale contact network model consisting of over 250M nodes and 15B edges in under 15 minutes (6.65 minutes for 10k seeds) while performing 23 billion diffusion simulations using 8k nodes of OLCF Frontier with over 64k GPUs and 400k CPUs.

II. Performance Attributes

Performance Attribute	Value
Category of achievement	Time-to-solution, size of the solution, approximation guarantee of the solution
Type of method used	OPIM-C[1] algorithm for Influence Maximization.
Results reported on the basis of	Whole application time including IO in strong and weak scaling studies. Kernel execution time for performance analysis.
Precision reported	Integer precision
System scale	8192 nodes (524k CPU-cores and 64k GPUs)
Measurement mechanism	Timers

III. Overview of the Problem (1 p max)

The *influence maximization problem* (InfMax) was first proposed by Domingos and Richardson [2] 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. The problem was formalized as a combinatorial optimization problem in the seminal paper by Kempe *et al.* [3], which has over 10,000 citations as of April, 2024. In this setting, a directed graph $G = (V, E, \omega)$ is used to represent individuals (referred to as nodes) in a population V, with an edge $(x, y) \in E$ if node x's choice impacts that of node y. Each node will be either "active" or "inactive", with all nodes being initially inactive; the edge weights $\omega(x, y)$ represent the probability of node x to influence node y for edge $(x, y) \in E$. In the simplest diffusion model, if a node x becomes active at time t, it attempts to activate each neighbor y (which never got activated so far) with probability $w(x, y)$. If the activation succeeds, node y would become activated at time $t + 1$. Node x stops participating in the process after time t. This diffusion model, denoted by M, corresponds to the well known *independent cascades* model; many other extensions of this model have been studied, including those rely on neighbor thresholds (e.g., the *linear threshold* model) to achieve activation.

The process of diffusion works as follows. First, an initial subset of nodes $S \subseteq V$, referred to as the *seed set*, is activated. Since nodes in S need incentives to become activated (e.g., they need to be given free products in the viral marketing example), there is typically a limited budget for how large such a set can be in practice (we use k to denote the budget). Once the initial activation of k vertex seeds is complete, the activation then spreads stochastically over a series of discrete time steps, as dictated by the underlying diffusion model M. Let $\sigma(S)$ denote the number of nodes which ever get activated in this process. Consequently, *the objective of the influence maximization problem is to select an optimal seed set S of size k that leads to the maximum expected influence ($E[\sigma(S)]$).*

The InfMax problem is known to be NP-hard (combinatorial explosion of search space) but efficient $(1-1/e)$-approximation algorithms exist in literature [3]. Most algorithms for InfMax exploit the submodularity (equivalently, diminishing marginal returns) property of $\sigma(\cdot)$, which enables a greedy algorithm to provide a $(1-1/e)$-approximation guarantee, as exemplified by the greedy hill-climbing algorithm of Kempe *et al.* [3]. However, a direct implementation of the greedy algorithm is computationally expensive, and does not scale beyond small networks, unless edges with really small probabilities are used or low quality solutions are computed. The literature on influence maximization has grown exponentially in the last few years. While a detailed exploration of the topic is beyond the scope of this work, we propose a categorization of InfMax algorithms as follows: *(i)* Classical algorithms on static graphs (the topic of this work) that provide an approximation bound of $(1-1/e-\varepsilon)$. Two key approaches for scaling InfMax to much larger networks are based on a greedy hill-climbing (GHC) and reverse influence sampling (RIS) techniques (§IV); *(ii)* InfMax on static graphs with partial observability, where co-optimization of sampling and influence-maximization are considered, with or without the approximation guarantee; *(iii)* InfMax on streaming data where strict constraints on memory exist and approximation guarantees are relaxed by a factor of α, for a user-defined value of $0 < \alpha \leq 1$; *(iv)* online influence maximization where strict time constraints on when the decision for seed selection in a streaming context exists, with and without approximation bounds; and, *(v)* dynamic influence maximization where changes to the graph are batched and the focus shifts to algorithmic efficiency in updated the seed sets. Numerous variants of the problem exist, such as, topic-aware, budget-aware, competitive, and target protection versions of InfMax. A body of learning-based approaches have also been proposed in literature [4]. Finally, several papers on parallelizing classical algorithms exist (§IV). Several excellent surveys provide an overview of the field [5]–[7].

We discuss the implications of scalable InfMax on important epidemic surveillance problems in §VIII. As evident during the early stages of the recent COVID pandemic, testing for when an outbreak is imminent is critically important and expensive [8]. This motivated the problem of finding a bounded subset S of nodes such that testing them would help in early detection of an impending epidemic outbreak. Other important applications for InfMax in this area include identification of early vaccination targets [9], and to determine importation times within a region R, which is the time when the outbreak within the region exceeds a threshold.

IV. Current State of the Art (1 p max)

A. SOTA in Epidemic Surveillance

Some of the earliest work on epidemic surveillance strategies was by Leskovec *et al.* [10], 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 [10] 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.

B. Parallel Influence Maximization Algorithms

Reverse Influence Sampling (RIS): The `Ripples` library represents the state-of-the-art InfMax implementations on distributed multi-GPU systems [11], [12], and is based on the IMM [13] 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 show that computation is dominated by the sampling steps producing the RRR sets [11], 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) [14]–[16]. The `Ripples` framework implements similar ideas and is tailored to amortize the offloading cost to GPUs as well as make better use of memory bandwidth of both GPUs and CPUs [17]. This recent work of Neff *et al.* [17] shows 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.* [13], we find that processing an USA-scale contact network with 258M vertices would result in 223 billion RRR sets during the first iteration, assuming an average of 10 vertices per RRR set, requiring **892 petabytes** of memory to store them. Frontier has 28 petabyte 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.* [3] introduced the greedy hill algorithm that is flexible for modeling different diffusion models but computationally expensive. Its complexity is: $\mathcal{O}(kn(n+e)\theta)$, where n in 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.* [9] 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.* [18] use a graph partitioning based heuristic to scale GHC computations. However, their approaches have scalability limitations for problem instances tackling intervention on nation-scale 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. For a θ value of 10k, GHC would need *10k years* to compute 10k seeds on 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. Consequently, innovations in both RIS- and GHC-based approaches are necessary.

V. Innovations Realized (2 pp max)

A. Algorithmic Innovation

As discussed in §IV, the need for massive numbers of simulations arises from the *offline* nature of current InfMax algorithms. This limitation is effectively addressed by *online* approaches that process information actively, and can be stopped at any point of the computation to generate a solution along with an estimation of its quality. However, online algorithms require substantially 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.* [1] and listed in Algorithm 1. Our analytical model of OPIM-C estimated that a USA-scale contact network would require 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 §IV. 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 a hypothetical optimal solution).

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.

Algorithm 1: OPIM-C Algorithm

```
Input: $G$, $k$, $\varepsilon$, $\delta$
Output: $S$
1  $\theta_0 \leftarrow$ thetaZero($n, k, \delta$)                 // Eq. (17) in [1]
2  $\theta_{max} \leftarrow$ thetaMax($n, k, \delta, \varepsilon$)       // Eq. (16) in [1]
3  $\mathcal{R}_1 \leftarrow$ Generate $\theta_0$ RRR sets from $G$
4  $\mathcal{R}_2 \leftarrow$ Generate $\theta_0$ RRR sets from $G$
5  for $i \leftarrow 1$ to $\lceil \log_2 \frac{\theta_{max}}{\theta_0} \rceil$ do
6      $S \leftarrow$ findSeeds($\mathcal{R}_1, k$)
7      $c_1 \leftarrow$ findCoverage($\mathcal{R}_1, S$)
8      $c_2 \leftarrow$ findCoverage($\mathcal{R}_2, S$)
9      $\sigma_l \leftarrow$ lBound($c_2, \delta, n, \mathcal{R}_2$)       // Eq. (5) in [1]
10     $\sigma_u \leftarrow$ uBound($c_1, \delta, n, \mathcal{R}_1$)       // Eq. (8) in [1]
11     $\alpha \leftarrow \sigma_l / \sigma_u$
12     if $\alpha \geq 1 - 1/e - \varepsilon$ then
13         return $S$
14     else
15         Double the size of $\mathcal{R}_1$ and $\mathcal{R}_2$
16 return $S$
```

B. Implementation Innovations

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 $\mathcal{R}_2$.

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.* [17]. 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 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 GPU to store the RRR sets produced during the simulations. Therefore, we adopt a more compact representation of the graph through quantization of edge probabilities into 16-bit integers. This compact representation reduces the graph storage requirements to $\approx 48GB$. Our sampling engine dynamically dispatches tasks to the CPUs and the

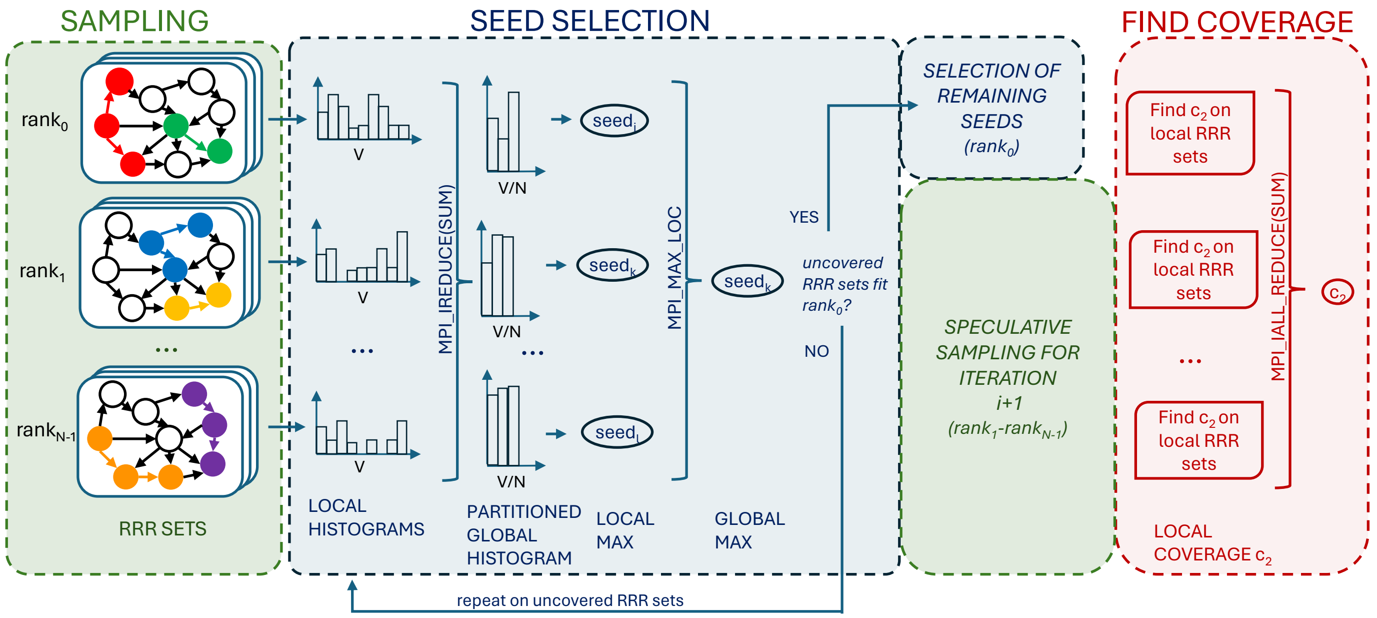
SAMPLING
SEED SELECTION
FIND COVERAGE
$rank_0$
$rank_1$
$rank_{N-1}$
RRR SETS
V
V/N
$seed_i$
$seed_k$
$seed_l$
MPI_IREDUCE(SUM)
MPI_MAX_LOC
LOCAL HISTOGRAMS
PARTITIONED GLOBAL HISTOGRAM
LOCAL MAX
GLOBAL MAX
YES
uncovered RRR sets fit $rank_0$?
NO
repeat on uncovered RRR sets
SELECTION OF REMAINING SEEDS ($rank_0$)
SPECULATIVE SAMPLING FOR ITERATION i+1 ($rank_1$-$rank_{N-1}$)
Find c_2 on local RRR sets
MPI_IALL_REDUCE(SUM)
c_2
LOCAL COVERAGE c_2

Fig. 1. Ripples overview.

GPUs participating in the construction of $\mathcal{R}_1$ and $\mathcal{R}_2$ [12]. To amortize the cost of offloading tasks to the GPUs, as well as to avoid redundant work during the simulation, we adopt the task-fusion technique proposed in [17].

Scalable Seed Selection: RIS-based INF-MAX 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 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. In order 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. Our implementation monitors the size of the uncovered RRR sets and, as soon as the cumulative size of $\mathcal{R}_1$ fits in a single machine, we move all the uncovered sets on rank 0 and complete seed selection from a single machine. To further improve the throughput of the algorithm, we relax the correctness of the implementation and we avoid updating the histograms once each selected vertex covers only one RRR set.

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 [1]. 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 effectiveness of this approach is illustrated in Figs. 2 and 3. Both figures show the timeline of execution of the application running at different scales. 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.

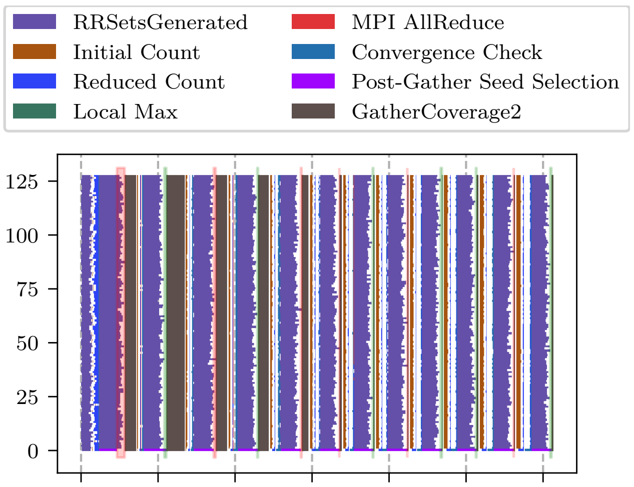

Fig. 2. Timeline of performing inf-max on 128 nodes with $k = 625, e = 0.03, delta = 0.01$. This includes the rank 0 gather and speculative execution. Red boxes indicate seed selection finished before sampling, and green boxes indicate the opposite.

Contributions: We make the following key innovations:

- Present the first distributed multi-GPU implementation of OPIM-C [1] on an exascale system and demonstrate scaling up to 64k GPUs and 200k CPUs using a largest-ever input with 250M nodes and 7B edges. This input is two-orders-of-magnitude larger than previous studies while producing 400× seeds with constant approximation factor of 60% ($\varepsilon = 0.03$).
- Introduce novel techniques to scale communication, a bottleneck for the online approach of OPIM-C, along with innovations such as speculative sampling to maximize the overlap of computation and communication.

VI. How Performance Was Measured (2 pp max)

A. 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

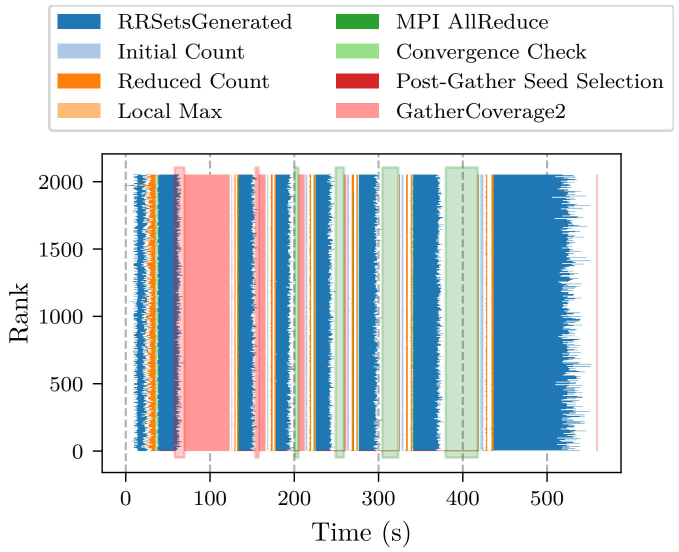

Fig. 3. Timeline of performing inf-max on 2048 nodes with $k = 10,000, e = 0.03, delta = 0.01$. This includes the rank 0 gather and speculative execution. Red boxes indicate seed selection finished before sampling, and green boxes indicate the opposite.

through a methodology using synthetic populations developed and refined by the team members over three decades (see, e.g. [19]–[24]). These and their associated populations have had broad application in computational epidemiology [25]–[30], transportation analysis [31]–[33], resilience assessments of socio-technical systems [34], evacuation studies [35]–[37], and planning scenarios addressing options for renewable energy [38]–[41].

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 [42]. *(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

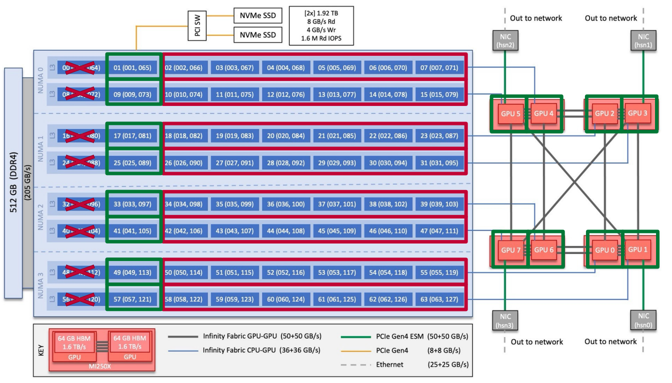

Fig. 4. Illustration of a Frontier node (Image credit: OLCF User Guide[43]) with an overlay showing how our application maps on the hardware. Crossed cores are reserved to perform OS services with the low-noise configuration; the red highlighted nodes execute tasks on the CPU; the green highlighted nodes are the closest to the GPUs and are used to perform GPU offloading.

the contact networks G used in this work.

The resulting synthetic populations statistically match the real populations of each block group 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.

B. 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 9408 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, 512GB of DDR4 memory, and four AMD MI250X (Figure 4). Each AMD MI250X includes two Graphics Computing Dies (GCDs) with 64GB of HBM2E memory. From a user perspective, one can think of GCDs as separate GPUs. The GCDs are connected among each other and to the CPU on the host through Infinity Fabric links. Each of the compute nodes on Frontier has two 1.92TB 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 8192 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 runtime 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 the memory allocator to use interleaving across NUMA domains. Our performance analysis showed that the application is memory bound and, therefore, we avoid using hyper-threading by setting the number of OpenMP threads to 56. 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 [12]. 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 [17] and we balanced the CPU and GPU task-binding rates of the dynamic scheduler through preliminary tuning studies on the machine as detailed in [17]. CPU workers are organized in 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 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. However, GPUs will benefit for smaller inputs that do not necessitate swapping from the memory.

We report the results from two ablation studies in §VII: *(i)* a benchmarking and detailed performance analysis; and *(ii)* a scalability study. The benchmarking and performance analysis present the different design options that we have guided the development of our final implementation by considering: *(i)* different graph representations; *(ii)* parallel pseudo-random number generators using integer of floating point operations; *(iii)* different reduction strategies for the `findSeeds` algorithm; *(iv)* different data movements 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 [44]: *(i)* Barabasi-Albert [45], *(ii)* LFR [46], *(iii)* RMAT [47], and *(iv)* Watts-Strogatz [48]. We report average execution time from multiple runs of a given feature being tested in the micro-benchmarks.

We present performance analysis of Ripples with strong and weak scaling studies using a USA-scale contact network as the input (§VI-B). 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 2,048 nodes up to 8,192 nodes, as our problem instance won't fit on smaller allocations. We study the weak scaling of our application with respect to the size of the solution set S. We study the weak scaling behavior between 128 nods with $k = 625$ and 8,192 nods with $k = 40,000$. The remaining parameters stay unchanged. All our experiments report wall-clock time from when the program starts reading the input file to its completion.

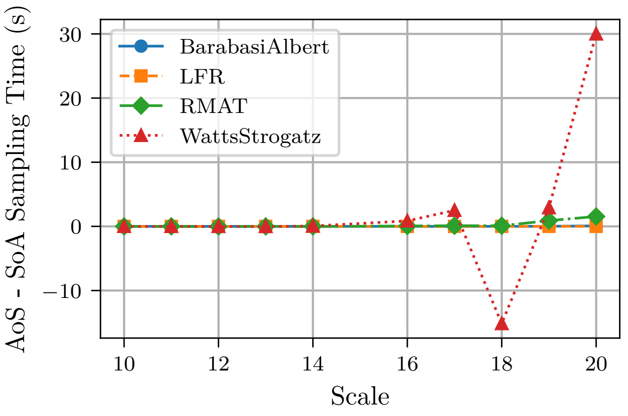

Fig. 5. Difference in execution time of the simulations algorithms on synthetic networks between the graph layout in Struct of Arrays (SoA) and Array of Structures (AoS) on CPUs. Anything above zero means SoA is better.

VII. Performance Results (2 pp max)

A. 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 storing the input graph. The first stores the graph in an Array of Structures (AoS) while the second uses 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 population of synthetically generated graphs showed that the performance of our simulation benefits 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 graph, with AoS generally performing better from scale 16 forward.

Pseudo-Random Number Generation: The performance of the simulations performed by our application are greatly influenced by the performance of the Pseudo-Random Number Generator (PRNG). We use the TRNG4 library that provides highly performing PRNGs [49]. While well performing approaches on CPUs are well-established in the practice[50], the same cannot be said for the simulations using GPUs. We mentioned in § V that we quantize the weights of the edges in input graph preserve memory to store the results of the simulations required by the application. Considering that integer operations are generally faster than floating point operations, one might expect that using a PRNG that uses solely integer operations might provide the best performance. Surprisingly, our roofline analysis (Fig. 6) shows that by using a floating-point PRNG we get a substantial increase in performance (triangles). We explain this behavior by noting that the our workload is in the memory-bound region of the roofline. Therefore, using floating point units to generate PNRG reduces the pressure on the integer units to generate more memory references. A similar boost can be observed when replacing the graph edge weight from float to 16 bits integers (dots). This is explained by the more compact representation of 16 bits integers making more efficient use of the available memory bandwidth. Finally, the roofline analysis shows that the best performing configuration is obtained by using floating point edge weights and a PRNG using floating point operations. However, the USA-scale contact network that we target would required a distributed CSR representation if we were to adopt floating point edge weight as the graph wouldn't leave enough space on the GPUs to perform the simulations. To avoid the communication costs of that a distributed CSR representation would require, we find that using 16 bits edge weights with a floating point PRNG a good trade-off. Concluding, we want to note that, while the inner working of the PRNG uses floating point numbers, its output is still an integer in the appropriate range and, therefore, our application is still working with integer precision and not in a mixed-precision regimen.

Communication in `findSeeds`: We previously noted(§ V) that `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. We benchmark two different approaches to solve the same problem. The first approach simply performs a reduce operation of vector of size n, the number of vertices in the input graph. The second approach block-distributes the global histogram across the P participating machines and performs P reduce of size n/P. Our micro-benchmarks show that the second approach is on average $1.1x$ time faster than the other. While this gain might seem slim at first, it gets amplified

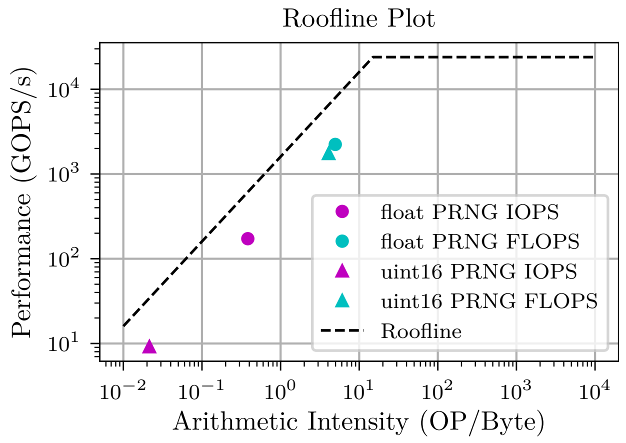

Fig. 6. GPU Roofline of each PRNG method showing both floating-point operation (FLOP) and integer operation (IOP) performance.

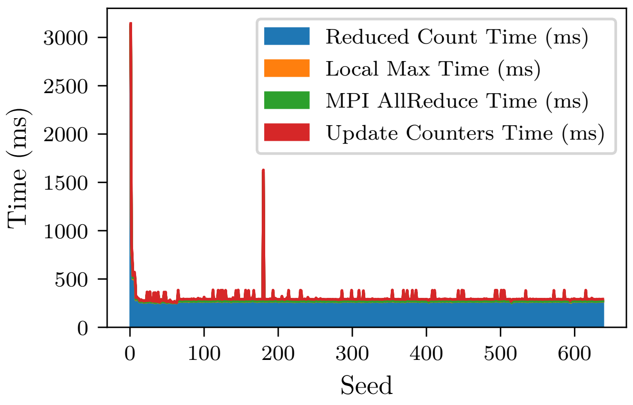

Fig. 7. Breakdown of max time across ranks for each step in selecting a seed.

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 remind that our large runs have $k = 10k$ or $k = 40k$, while I is generally between 3 and 6. Fig. 7 shows the time breakdown of a single iteration of **`findSeeds`** when $k = 625$. Two important observations emerge from Fig. 7. First, the early iterations of the algorithm take $\approx 7\times$

Speculative Execution:

Load Balancing:

B. Strong and Weak Scaling Studies

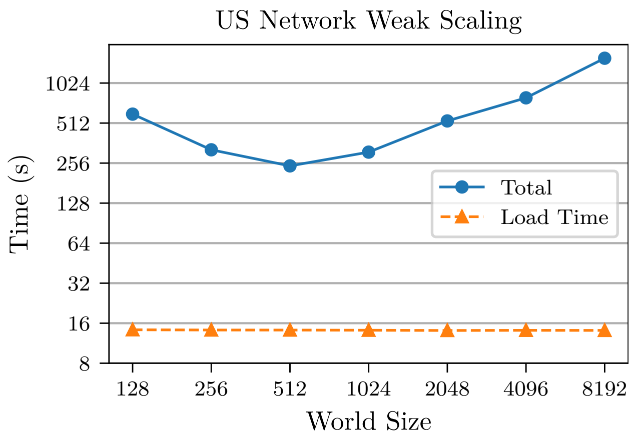

Fig. 8. Weak Scaling from 128 nodes up to 8192 nodes on the USA-scale network. We study the weak scaling fixing ration between k and the allocated nodes: $k = 625$ @ 128 nodes; $k = 40k$ @ 8,196 nodes. The other parameters are set to: $\varepsilon = 0.03$, and $\delta = 0.01$.

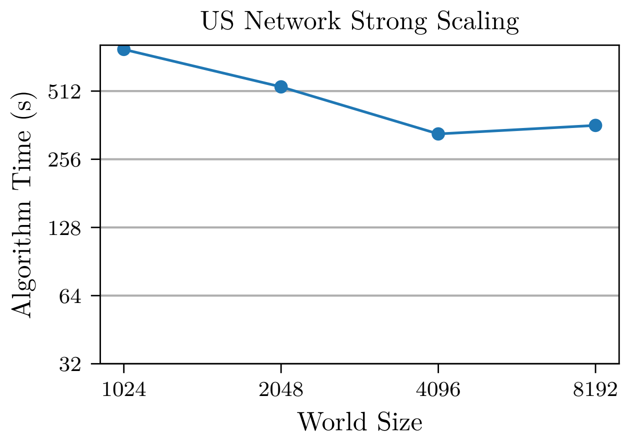

Fig. 9. Strong scaling from 1024 nodes up to 8192 nodes on the USA-scale network. We fix parameters to: $k = 10k$, $\varepsilon = 0.03$, and $\delta = 0.01$. This problem instance does not fit in less than 1,024 Frontier nodes.

VIII. Implications (1 p max)

We briefly summarize our key technical contributions and discuss how such scalable algorithms can be used for pandemic planning and response.

Influence Maximization: This work represents the first execution of InfMax on graphs that are over two orders-of-magnitude larger than before, and required new algorithm and parallelization techniques. 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.

Implications for pandemic planning and response: All of the pandemics in the last 50 years or more have arrived to the US via individuals coming to US, e.g., [51], [52]. This can include visitors, or residents who might have gone and visited other countries. In most cases today, most of these initial cases arrive at one of the international airports. Informally importation refers to arrival of first few cases of the pandemic to a region that would lead to a sustained transmission in the region. Importation time is a critical quantity for planning a more efficient response to the pandemic. It provides policy makers to start preparation, and importantly as discussed in subsequent sections put in the place a testing coupled with quarantine regimen 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 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 [10]—this is the number of infections in R by the time the outbreak is detected in R; Leskovec *et al.* [10] 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., [53], [54], 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.

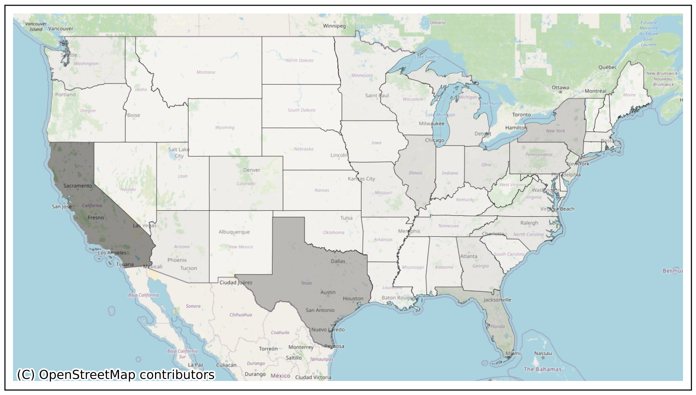

Fig. 10. A spatial breakdown of the 10k seeds by state depicted using a grayscale gradient. Seed vertices correspond to persons in the synthetic population for which the US network was generated, and accumulation is done through person's residence state. California has the largest number of seeds, accounting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

Acknowledgment

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, the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), through the Advanced Graphic Intelligence Logical Computing Environmet (AGILE) research program, contract number 77740. 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] J. Tang, X. Tang, X. Xiao, and J. Yuan, "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*, G. Das, C. M. Jermaine, and P. A. Bernstein, Eds., ACM, 2018, pp. 991–1005.

[2] P. Domingos and M. Richardson, "Mining the network value of customers," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 2001, pp. 57–66.

[3] D. Kempe, J. M. Kleinberg, and É. Tardos, "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*, 2003, pp. 137–146. [Online]. Available: https://doi.org/10.1145/956750.956769.

[4] S. Munikoti, D. Agarwal, L. Das, M. Halappanavar, and B. Natarajan, "Challenges and opportunities in deep reinforcement learning with graph neural networks: A comprehensive review of algorithms and applications," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2023.

[5] J. Sun and J. Tang, "A survey of models and algorithms for social influence analysis," in *Social Network Data Analytics*, C. C. Aggarwal, Ed. Boston, MA: Springer US, 2011, pp. 177–214.

[6] S. Peng, Y. Zhou, L. Cao, S. Yu, J. Niu, and W. Jia, "Influence analysis in social networks: A survey," *Journal of Network and Computer Applications*, vol. 106, pp. 17–32, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804518300195.

[7] S. Banerjee, M. Jenamani, and D. K. Pratihar, "A survey on influence maximization in a social network," vol. 62, no. 9, pp. 3417–3455, Sep. 2020. [Online]. Available: https://doi.org/10.1007/s10115-020-01461-4.

[8] N. D. Goldstein and I. Burstyn, *On the importance of early testing even when imperfect in a pandemic such as covid-19*, Mar. 2020. [Online]. Available: osf.io/9pz4d.

[9] M. Minutoli, P. Sambaturu, M. Halappanavar, A. Tumeo, A. Kalyanaraman, and A. Vullikanti, "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, 2020, p. 55.

[10] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, "Cost-effective outbreak detection in networks," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2007, pp. 420–429.

[11] M. Minutoli, M. Halappanavar, A. Kalyanaraman, A. V. Sathanur, R. S. McClure, and J. E. McDermott, "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, 2019, pp. 1–12.

[12] M. Minutoli, M. Drocco, M. Halappanavar, A. Tumeo, and A. Kalyanaraman, "cuRipples: influence maximization on multi-GPU systems," in *ICS '20: 2020 International Conference on Supercomputing, Barcelona Spain, June, 2020*, ACM, 2020, 12:1–12:11.

[13] Y. Tang, Y. Shi, and X. Xiao, "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*, 2015, pp. 1539–1554. [Online]. Available: https://doi.org/10.1145/2723372.2723734.

[14] G. Göktürk and K. Kaya, "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, 2022, pp. 908–918.

[15] G. Göktürk and K. Kaya, "Fast and error-adaptive influence maximization based on count-distinct sketches," *Inf. Sci.*, vol. 655, p. 119 875, 2024.

[16] G. Göktürk and K. Kaya, "Boosting parallel influence-maximization kernels for undirected networks with fusing and vectorization," *IEEE Trans. Parallel Distributed Syst.*, vol. 32, no. 5, pp. 1001–1013, 2021.

[17] R. Neff, M. E. Zarch, M. Minutoli, *et al.*, "Fused breadth-first probabilistic traversals on distributed GPU systems," *CoRR*, vol. abs/2311.10201, 2023. arXiv: 2311.10201.

[18] R. Barik, M. Minutoli, M. Halappanavar, and A. Kalyanaraman, "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, 2022, pp. 125–134.

[19] R. J. Beckman, K. A. Baggerly, and M. D. McKay, "Creating synthetic baseline populations," *Transportation Research Part A: Policy and Practice*, vol. 30, no. 6, pp. 415–429, 1996.

[20] S. Eubank, V. S. A. Kumar, M. Marathe, A. Srinivasan, and N. Wang, "Structure of social contact networks and their impact on epidemics," in *Discrete Methods in Epidemiology*, ser. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 70, 2006.

[21] H. Xia, J. Chen, M. Marathe, H. Mortveit, and M. Salathe, "Synthesizing social proximity networks by combining subjective surveys with digital traces," in *2013 IEEE 16th International Conference on Computational Science and Engineering*, Dec. 2013, pp. 188–195.

[22] K. Atkins, C. Barrett, R. Beckman, *et al.*, *Synthetic data products for societal infrastructures and proto populations: Data set 1.0.* NDSSL Technical Report No. 06-006, 2006.

[23] A. Adiga, R. Beckman, K. Bisset, *et al.*, *Synthetic populations for epidemic modeling*, International conference on computational social sciences (ICCSS), June 8-11, Helsinki, Finland, 2015, 2015.

[24] P. Bhattacharya, J. Chen, S. Hoops, *et al.*, "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*, vol. 37, no. 1, pp. 4–27, 2023, Gordon Bell Award finalist. Online version. eprint: https://doi.org/

10.1177/10943420221127034. [Online]. Available: https://doi.org/10.1177/10943420221127034.

[25] J. Chen, A. Vullikanti, J. Santos, *et al.*, "Epidemiological and economic impact of COVID-19 in the US," *Scientific Reports*, vol. 11, no. 1, p. 20 451, 2021. [Online]. Available: https://doi.org/10.1038/s41598-021-99712-z.

[26] A. Adiga, J. Chen, M. Marathe, H. Mortveit, S. Venkatramanan, and A. Vullikanti, "Data-driven modeling for different stages of pandemic response," *Journal of the Indian Institute of Science*, vol. 100, pp. 901–915, 2020.

[27] J. Chen, A. Vullikanti, S. Hoops, *et al.*, "Medical costs of keeping the US economy open during COVID–19," *Nature Scientific Reports*, vol. 10, p. 18 422, 1 2020.

[28] J. Chen, S. Hoops, A. Marathe, *et al.*, "Effective social network-based allocation of covid-19 vaccines," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ser. KDD '22, Washington DC, USA: Association for Computing Machinery, 2022, pp. 4675–4683. [Online]. Available: https://doi.org/10.1145/3534678.3542673.

[29] E. Howerton, L. Contamin, L. C. Mullany, *et al.*, "Evaluation of the us covid-19 scenario modeling hub for informing pandemic response under uncertainty," *Nature Communications*, vol. 14, 1 2023. [Online]. Available: https://doi.org/10.1038/s41467-023-42680-x.

[30] "Active sensing for epidemic state estimation using ABM-guided machine learning," in *Proceedings of the Multi-agent-based Simulation (MABS) Workshop*, Best Paper Award, 2023. [Online]. Available: https://mabsworkshop.github.io/articles/salibaEtAl2023.pdf.

[31] L. Smith, R. Beckman, K. Baggerly, D. Anson, and M. Williams, *Transims: Transportation analysis and simulation system*, Los Alamos National Laboratory technical report: LA–UR: 95-1641, Jul. 1995. [Online]. Available: https://www.osti.gov/biblio/88648.

[32] R. J. Beckman, C. L. Barrett, K. B. Berkbigler, *et al.*, *Transportation ANalysis SIMulation System TRANSIMS - The Dallas-Ft. Worth Case Study*, Los Alamos National Laboratory technical report: LA-UR: 97–4502, 1997.

[33] Federal Highway Administration, *TRANSIMS*, Online, https://www.fhwa.dot.gov/planning/tmip/resources/transims/. [Online]. Available: https://www.fhwa.dot.gov/planning/tmip/resources/transims/.

[34] C. Barrett, K. Bisset, S. Chandan, *et al.*, "Planning and response in the aftermath of a large crisis: An agent-based informatics framework," in *Proceedings of the 2013 Winter Simulation Conference*, 2013, pp. 1515–1526. [Online]. Available: https://informs-sim.org/wsc13papers/includes/files/132.pdf.

[35] A. E. Brower, B. Ramesh, K. A. Islam, *et al.*, "Augmenting the social vulnerability index using an agent-based simulation of hurricane Harvey," *Computers, Environment and Urban Systems*, vol. 105, p. 102 020, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0198971523000832.

[36] K. A. Islam, D. Q. Chen, M. Marathe, H. Mortveit, S. Swarup, and A. Vullikanti, "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*, 2023, pp. 2397–2399.

[37] K. A. Islam, D. Q. Chen, M. Marathe, H. Mortveit, S. Swarup, and A. Vullikanti, "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*, Article No.: 595, 2023, pp. 5359–5367. [Online]. Available: https://doi.org/10.24963/ijcai.2023/595.

[38] T. Swapna, M. Henning, V. Anil, M. Madhav, and S. Samarth, "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, 2024.

[39] S. Thorve, Y. Y. Baek, S. Swarup, *et al.*, "High resolution synthetic residential energy use profiles for the united states," *Scientific Data*, vol. 10, no. 1, p. 76, 2023. [Online]. Available: https://doi.org/10.1038/s41597-022-01914-1.

[40] R. Meyur, S. Thorve, M. Marathe, A. Vullikanti, S. Swarup, and H. Mortveit, "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*, L. D. Raedt, Ed., AI for Good, International Joint Conferences on Artificial Intelligence Organization, Jul. 2022, pp. 5115–5121. [Online]. Available: https://doi.org/10.24963/ijcai.2022/710.

[41] R. Meyur, A. Vullikanti, S. Swarup, *et al.*, "Ensembles of realistic power distribution networks," *Proceedings of the National Academy of Sciences*, vol. 119, e2205772119, 42 2022. [Online]. Available: https://www.pnas.org/doi/full/10.1073/pnas.2205772119.

[42] *2016-2020 5-Year ACS Commuting Flows*, Last accessed: 05 March, 2024, 2024. [Online]. Available: https://www.census.gov/data/tables/2020/demo/metro-micro/commuting-flows-2020.html.

[43] L. C. C. F. at Oak Ridge National Laboratory, *Frontier user guide.* [Online]. Available: https://docs.olcf.ornl.gov/systems/frontier_user_guide.html.

[44] E. Angriman, A. van der Grinten, M. Hamann, H. Meyerhenke, and M. Penschuck, "Algorithms for large-scale network analysis and the networkit toolkit," in *Algorithms for Big Data - DFG Priority Program 1736*, ser. Lecture Notes in Computer Science, H. Bast, C. Korzen, U. Meyer, and M. Penschuck, Eds., vol. 13201, Springer, 2022, pp. 3–20.

[45] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.

[46] A. Lancichinetti and S. Fortunato, "Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities," *Physical review E*, vol. 80, no. 1, p. 016 118, 2009.

[47] D. Chakrabarti, Y. Zhan, and C. Faloutsos, "R-mat: A recursive model for graph mining," in *Proceedings of the 2004 SIAM International Conference on Data Mining*, SIAM, 2004, pp. 442–446.

[48] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world'networks," *nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[49] H. Bauke and S. Mertens, "Random numbers for large-scale distributed monte carlo simulations," *Phys. Rev. E*, vol. 75, p. 066 701, 6 Jun. 2007. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.75.066701.

[50] H. Bauke, *Tina's random number generator library.* [Online]. Available: https://www.numbercrunch.de/trng/trng.pdf.

[51] T. F. Menkir, T. Chin, J. A. Hay, *et al.*, "Estimating internationally imported cases during the early covid-19 pandemic," *Nature communications*, vol. 12, no. 1, p. 311, 2021.

[52] C. R. Wells, P. Sah, S. M. Moghadas, *et al.*, "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*, vol. 117, no. 13, pp. 7504–7509, 2020.

[53] N. Sood, P. Simon, P. Ebner, *et al.*, "Seroprevalence of sars-cov-2–specific antibodies among adults in los angeles county, california, on april 10-11, 2020," *JAMA*, vol. 323, no. 23, pp. 2425–2427, 2020.

[54] W. Zhang, J. P. Govindavari, B. D. Davis, *et al.*, "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*, vol. 3, no. 10, e2024191, 2020.