Scalable and Memory-Efficient Algorithms for Controlling Networked Epidemic Processes Using Multiplicative Weights Update Method

Prathyush Sambaturu[1], **Marco Minutoli**[2], **Mahantesh Halappanavar**[2,3], **Ananth Kalyanaraman**[3,2] and **Anil Vullikanti**[1]
[1]University of Virginia, Charlottesville, VA, USA
[2]Pacific Northwest National Laboratory, Richland, WA, USA
[3]Washington State University, Pullman, WA, USA
{pks6mk,vsakumar}@virginia.edu, {marco.minutoli, mahantesh.halappanavar}@pnnl.gov, ananth@wsu.edu

Abstract

We study the problem of designing scalable algorithms to find effective intervention strategies for controlling stochastic epidemic processes on networks. This is a common problem arising in agent based models for epidemic spread. Previous approaches to this problem focus on either heuristics with no guarantees or approximation algorithms that scale only to networks corresponding to county-sized populations, typically, with less than a million nodes. In particular, the mathematical-programming based approaches need to solve the Linear Program (LP) relaxation of the problem using an LP solver, which restricts the scalability of this approach. In this work, we overcome this restriction by designing an algorithm that adapts the multiplicative weights update (MWU) framework, along with the sample average approximation (SAA) technique, to approximately solve the linear program (LP) relaxation for the problem. To scale this approach further, we provide a memory-efficient algorithm that enables scaling to large networks, corresponding to country-size populations, with over 300 million nodes and 30 billion edges. Furthermore, we show that this approach provides near-optimal solutions to the LP in practice.

1 Introduction

Due to the complex factors underlying disease spread, mathematical models based on the Susceptible-Infected-Recovered (or SIR) process are used extensively to understand the trade-off between the cost of interventions (e.g., vaccination and social distancing), and the benefit (e.g., the number of infections averted), e.g., [Medlock and Galvani, 2009; Marathe and Vullikanti, 2013; Halloran *et al.*, 2008; Eubank *et al.*, 2004; Germann *et al.*, 2006]. As an example, the CDC Scenario Hub [Truelove *et al.*, 2021] involves the use of a variety of models—deterministic models based on differential equations, e.g., [Medlock and Galvani, 2009; Anderson and May, 1991], and stochastic models based on networks, referred to as networked SIR models, e.g., [Marathe and Vullikanti, 2013; Halloran *et al.*, 2008; Eubank *et al.*, 2004; Germann *et al.*, 2006; Chen *et al.*, 2021]—in order to evaluate the benefits of different interventions, and find most effective ones.

We refer to the problem of designing an intervention to minimize the number of infections as EpiControl (note that this is a very general and complex problem, and we only study a very specific form of the problem, which is defined formally in Section 2). Such epidemic analyses implicitly try to find near-optimal solutions (or solutions better than current policies) to EpiControl to understand (a) how much benefit they can provide, or (b) structural properties of near-optimal solutions, which can give insights for more implementable policies, e.g., [Medlock and Galvani, 2009; Chen *et al.*, 2021; Sambaturu *et al.*, 2020]. We note that though the problem, in reality, is much more complex than the simplified version we study, even that is open, in general networks and disease model, making this a natural starting point.

Near-optimal solutions to EpiControl in models based on differential equations can be computed by local search methods, as in [Medlock and Galvani, 2009]. However, solving the EpiControl problem in network or agent based models is very hard [Eubank *et al.*, 2006; Hayrapetyan *et al.*, 2005]. A number of heuristics have been considered, e.g., prioritizing based on degree, centrality, or spectral properties e.g., [Cohen *et al.*, 2003; Miller and Hyman, 2007; Saha *et al.*, 2015; Chen *et al.*, 2021], but it is possible to show that they have $\Omega(n)$ worst case bounds, in general, where n is the number of nodes in the network (we summarize other related work in Section 6). An approach that has been effective for some versions of the problem is using linear programming (LP) and randomized rounding [Hayrapetyan *et al.*, 2005] (for the special case where the transmission probability is 1). This approach, combined with the *sample average approximation* (SAA) technique [Kleywegt *et al.*, 2002] (which is a powerful tool from stochastic optimization, and reduces the problem to solving a deterministic problem on a set of samples) has been used for handling the more realistic regime of EpiControl with transmission probability less than 1 [Sambaturu *et al.*, 2020; Babay *et al.*, 2022]. [Babay *et al.*, 2022] obtain the first rigorous approximation bounds for some settings; [Sambaturu *et al.*, 2020] show that this approach works well in practice.

Method	Runtime	Space
SAAROUND [Sambaturu *et al.*, 2020]	$O((n+nM)^{2.5})$	$O((n+nM)mM)$
MWUROUND	$\tilde{O}(\epsilon^{-2}nmM)$	$O(nM+mM)$
MWUROUND-SCALABLE	$\tilde{O}(\epsilon^{-2}nmq)$	$O(m+nM)$

Table 1: Runtime and space requirements.

However, such LP based approaches (which use standard solvers, such as Gurobi) only scale to small county-sized networks (with up to 10^5 nodes). To put this in perspective, the CDC Scenario Hub models [1] run on state level network based models with millions of nodes. Further, such studies involve large experimental designs (due to the number of parameters), which requires solving intervention design problems on thousands of network models. Therefore, algorithms which can scale to networks with millions of nodes are necessary for supporting such policy analyses.

Key Contributions. We design algorithms MWUROUND and MWUROUND-SCALABLE for finding near-optimal vaccination strategies in networked SIR models in large networks with hundreds of millions of nodes (Section 3); see summary in Table 1. Our algorithms adapt the multiplicative weights update (MWU) [Arora *et al.*, 2012] to run the sample average approximation and LP rounding steps of [Sambaturu *et al.*, 2020]. More specifically,

- We show that our problem can be reduced to a covering type problem, for which prior MWU based algorithms exist [Arora *et al.*, 2012; Fleischer, 2000]; we show that by exploiting the problem structure, we can get an improvement of more than a factor of M (the number of samples) over the standard use of MWU (Section 3). This also improves both the running time and space of SAAROUND by more than a factor of mM—this is many orders of magnitude in massive networks.
- We design a more scalable algorithm MWUROUND-SCALABLE, by running the MWU computations on random samples computed on-the-fly, leading to almost a factor M improvement in running time, and a factor M improvement in space. This allows MWUROUND-SCALABLE to run on national scale networks.
- We show that our approach can be extended to the FAIREPICONTROL problem, which incorporates fairness constraints with respect to the budget. Our results show that such fairness constraints have a significant impact on epidemic outcomes.

These performance gains (runtime and memory) are achieved without compromising on the approximation quality. In Section 5, we present a detailed experimental evaluation of the proposed algorithms on a number of real-world and synthetic networks; the largest is a country-scale contact network containing over 334M nodes. Though the version of EPICONTROL we consider is admittedly the simplest possible (which itself has not been fully resolved), e.g., compared with other models, our results show that the sample average approximation, linear programming and rounding, and MWU techniques can help in scaling interventions in networked SIR models.

The supplementary information, including proofs and further details, can be found in the full version [2].

2 Preliminaries

Notation	Definition
$G=(V,E)$	Contact network
p	transmission probability
$H'_j=(V_{H'_j},E_{H'_j})$	A sampled graph of G
x_u	Indicator for u getting vaccinated
y_{vj}	Indicator for v getting infected in H'_j
$H_j=(V_{H_j},E_{H_j})$	Augmented sampled graph
$a(u,j)$	Copy/stub of node u attached to u in H_j
$A(j)$	Set of stub nodes in H_j
$\mathcal{P}_{v,j}$	Set of paths from S to stub v in H_j
$\mathcal{P}_j$	Set of paths from S to all stubs in $A(j)$
$\mathcal{P}$	$=\bigcup_{j\in[M]}\mathcal{P}_j$, i.e., set of all paths
$\ell(u)$	Length of node/stub u
$\ell(P)$ for $P\in\mathcal{P}$	Sum of lengths of nodes in P
$z(P)$	Flow on path P

Table 2: Summary of notation.

Let $G=(V,E)$ be a contact network, where V is the set of people or nodes, and $e=(u,v)\in E$ when nodes $u,v\in V$ come into direct contact. Let $|V|=n$ and $|E|=m$. Let us assume the following simple SIR model of disease spread.

SIR Epidemic Model on Networks. An agent based SIR model of disease spread can be viewed as a diffusion process on a network [Marathe and Vullikanti, 2013; Halloran *et al.*, 2008]. Each node in the network is in one of the following three states: Susceptible (S), Infectious (I), or Recovered/Removed (R). In its simplest form, we have a discrete time model, and in each time step, an infected node spreads infection to each of its susceptible neighbors with a probability p — called transmission probability. An infected node recovers after its infectious duration (and doesn't get infected after that). We assume that the disease starts at a fixed set of externally infected nodes $S\subseteq V$ and $|S|=k$ (our results extend to more general starting conditions, e.g., a random source). We assume that interventions have 100% efficacy. Let #infections$(\mathbf{X})$ denote the number of infections that result if set $\mathbf{X}$ of nodes is vaccinated; this is a random variable, and our goal is to minimize $\mathbb{E}[\text{\#infections}(\mathbf{X})]$.

EPICONTROL problem. Given a contact network $G=(V,E)$, a simple SIR model of disease spread, a fixed set of sources of infection S where $|S|=k$, and a budget B on the number of interventions. Find a set of nodes $\mathbf{X}\subseteq V$ such that $|\mathbf{X}|\leq B$ and the expected number of infections $\mathbb{E}[\text{\#infections}(\mathbf{X})]$ is minimized.

Our results extend to more general settings with non-uniform transmission probabilities, and other starting conditions, such as random sources. However, the EPICONTROL stated above is admittedly much simpler than the ones that

[1]https://viz.covid19scenariomodelinghub.org

[2]Link to full version: https://tinyurl.com/mwuroundfullversion

need to be considered in public health analyses, e.g., requiring adaptive interventions, which have limited efficacy, and with a lot of uncertainty in the model.

Next, we consider fairness with respect to the budget. We have a partition of V into groups $V_1, \dots, V_c$, and a budget B_i for each group (e.g.,proportional to the size of each V_i).

FAIREPICONTROL problem. Given a contact network $G = (V, E)$, a simple SIR model of disease spread, a fixed set of sources of infection S where $|S| = k$, a partition of $V = \cup_{i=1}^{c} V_i$, and a budget B_i for each group V_i. Find a set of nodes $\mathbf{X}_i \subseteq V_i$ such that $|\mathbf{X}_i| \leq B_i$, for each i, and the expected number of infections $\mathbb{E}[\#\text{infections}(\mathbf{X}_1, \dots, \mathbf{X}_c)]$ is minimized.

Note that there are many other more complex fairness models [Barocas *et al.*, 2019], but the EPICONTROL is challenging even in the simplest setting, e.g., [Dinitz *et al.*, 2022].

2.1 SAA Based Algorithm

We briefly summarize the stochastic optimization based algorithm of [Sambaturu *et al.*, 2020], referred to as SAAROUND, which is our starting point. It involves the following steps (we only give a short description here, and refer to [Sambaturu *et al.*, 2020] for details).

- Construct a sampled graph $H'_j = (V_{H'_j}, E_{H'_j})$, for $j = 1, \dots, M$, by picking each edge $e \in E$ to be in $E_{H'_j}$ with a probability p.
- Solve the following linear program (LP_{saa}): $\min \frac{1}{M} \sum_j \sum_v y_{vj}$, subject to: (1) $\forall j, \forall u \in V : y_{uj} \leq 1 - x_u$, (2) $\forall j, \forall u \in V,\ (w, u) \in E_j : y_{uj} \geq y_{wj} - x_u$, (3) $\forall s \in S : y_{sj} = 1$, (4) $\sum_{u \in V} x_u \leq B$ (5) All variables $\in [0, 1]$
- Let x, y be the optimal fractional solution to LP_{saa}. For each v, set $X_v = 1$ with probability $\min\{1, 2x_v \log(4nMN)\}$, where N is the maximum number of paths from S to any node v in H'_j.
- $X = \{v : X_v = 1\}$ is the set of nodes vaccinated.

[Sambaturu *et al.*, 2020] show that using $M = \Omega(n^2 \log n)$ samples, the above approach gives a bicrteria approximation, vaccinating $O(\log(4nMN)B)$, while ensuring that the expected number of infections is at most six times the optimal. Solving the LP in SAAROUND is the main bottleneck, as it has $n + nM$ variables and $\sum_j |E_j|$ constraints—they report that this only scales to contact networks with up to 10^5 nodes.

3 Algorithm

We improve the approach of [Sambaturu *et al.*, 2020] by bypassing the need to use a solver to directly solve the LP. Instead, we adapt the multiplicative weights update technique [Arora *et al.*, 2012] to find a near-optimal solution to the LP. It will be easier to present the LP in a slightly different form. Let y_{vj} be an indicator whether the node v gets infected in the sampled outcome H'_j. Let $\mathcal{P}'_v$ be the set of paths from S to node v in any outcome H'_j for $j \in [M]$.

$$(LP_{path}) \qquad Z_{LP} = \min \frac{1}{M} \sum_j \sum_{v \in V_{H'_j}} y_{vj} \ s.t. \tag{1}$$

$$\forall v \in V_{H'_j} \setminus S, \forall P \in \mathcal{P}'_v\ , \sum_{u \in P} x_u + y_{vj} \geq 1 \tag{2}$$

$$\sum_{u \in V} x_u \leq B; \ \ x_u, y_{vj} \in [0, 1] \tag{3}$$

By adding up the constraints for each edge of a path of LP_{saa}, it can be verified that we get the constraint (2) of LP_{path}, which is summarized below.

Observation 1. *The above LP is equivalent to* LP_{saa}.

The main ideas and steps are described below.

1. Lagrangian multiplier for budget. The dual of LP_{path} is complicated due to a negative coefficient associated with the budget constraint (3). We simplify it by changing the objective to $\frac{1}{M} \sum_j \sum_{v \in V_{H'_j}} y_{vj} + \lambda \sum_{u \in V} x_u$, with the multiplier λ for the cost of the solution. The budget constraint is dropped; we refer to this LP as LP_{LM}. This simplifies the resulting LP, since it only has covering constraints. As λ increases, $\sum_u x_u$ will decrease in the optimal solution. Since values for λ are not known a priori, a binary search can be employed to find a suitable value λ' such that $\sum_u x_u \leq B$, which is done in Algorithm LSEARCH-SAA. The x', y' values returned by Algorithm 1 for λ' provides an approximate solution to LP_{saa}.

2. Constructing augmented sampled graphs. For simplifying the presentation, we construct M sampled graphs $H_j = (V_{H_j}, E_{H_j})$ in the following manner: H_j is initially the same as H'_j, constructed as in the first step of SAAROUND. Let $A(j) = \{a(u, j) : u \in V_{H'_j} - S\}$, where $a(u, j)$ denotes a copy of node u in H_j, and is referred to as a *stub*. Let $\mathbf{A} = \bigcup_{j \in [M]} A(j)$ be set of all stubs. Each *stub* $a(u, j)$ is attached to u by an edge $(u, a(u, j))$. Overloading the definitions, let $\mathcal{P}_{v,j}$ denote the set of paths from S to a *stub* node $v = a(u, j) \in A(j)$ in H_j. Let $\mathcal{P}_j = \bigcup_{v \in A(j)} \mathcal{P}_{v,j}$ and $\mathcal{P} = \bigcup_{j \in [M]} \mathcal{P}_j$.

3. Variables and costs. We associate a length to each node $u \in (V - S) \bigcup \mathbf{A}$ denoted by $\ell(u)$. For a node $u \in V$, $\ell(u)$ will correspond to the variable x_u, while for a node $v = a(u, j) \in \mathbf{A}$, $\ell(v)$ will correspond to the variable y_{uj}. The length of any path in $P \in \mathcal{P}$ is given by the sum of lengths of nodes on this path. Let $\ell = \langle \ell(u) : u \in V \bigcup \mathbf{A} \rangle$ denote the vector of length variables. Let $c(u) = \lambda$ for $u \in V \setminus S$ denote its capacity, whereas $c(u) = 0$ for $u \in S$. Let $c(v) = \frac{1}{M}$ for $v = a(u, j) \in \mathbf{A}$ denote the capacity of a *stub* v. We will keep track of flows on the network; let $z(P)$ denote the flow on the path $P \in \mathcal{P}$. Let $\mathbf{z} = \langle z(P) : P \in \mathcal{P} \rangle$ denote the vector of flow variables.

Algorithm 1 MwuSaa (λ)
Input: parameter λ (we assume the network $G = (V, E), S$, subgraphs $H_1, \ldots, H_M$, ϵ are fixed, $\delta = (1+\epsilon)((1+\epsilon)L)^{-\frac{1}{\epsilon}}$ where L is the max. number of nodes on any path in G
Output: ℓ

```
1: Initialize $\ell(u) = \delta$ for all $u \in (V - S) \cup \mathbf{A}$,
   $z(P) = 0$ for all $P \in \mathcal{P}$.
2: Set $c(u) = \lambda$ for $u \in V - S$ and $c(v) = 1/M$ for $v \in \mathbf{A}$
3: for $r = 1$ to $\lfloor \log_{1+\epsilon} \frac{1+\epsilon}{\delta} \rfloor$ do
4:   for $j = 1$ to $M$ do
5:     while there exists path $P \in \mathcal{P}_j$ such that $\ell(P) < \delta(1+\epsilon)^r$ do
6:       Let $c(P) = \min_{u \in P} c(u)$
7:       Let $d \geq 1$ be the smallest integer such that
         $\sum_{v \in P-S} \ell(v) \left(1 + \frac{\epsilon c(P)}{c(v)}\right)^d \geq \delta(1+\epsilon)^r$
8:       $z(P) \leftarrow z(P) + d \cdot c(P)$
9:       For $v \in P - S$, $\ell(v) \leftarrow \ell(v) \left(1 + \frac{\epsilon c(P)}{c(v)}\right)^d$
10:    end while
11:  end for
12: end for
13: for each $v \in V - S$, $\ell(v) = \frac{\ell(v)}{\ell_{max}}$ where $\ell_{max} = \max_{v \in V \setminus S} \ell(v)$
14: Return $\ell$
```

4. Simplified LP. The above discussion reduces our LP to the following:

$$LP_\ell(\lambda) : \mathrm{Z}_{LR}(\lambda) = \min \sum_u \ell(u)c(u) \ s.t.$$

$$\forall P \in \mathcal{P} : \sum_{u \in P-S} \ell(u) \geq 1; \qquad \forall u : \quad \ell(u) \geq 0$$

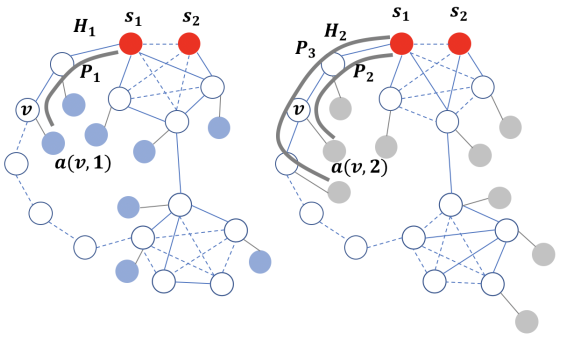

Figure 1: Example with two samples H_1, H_2, and stub nodes.

5. Incremental computation of $\ell(\cdot)$. Algorithm MwuSaa computes an approximate solution to LP_ℓ, using the multiplicative weight update technique [Arora *et al.*, 2012]. It starts by initializing the length $\ell(v) = \delta$ for each $u \in (V \setminus S) \bigcup \mathbf{A}$, where δ has a very small value determined in the analysis. The $\ell(v)$ for $v \in S$ is initialized to zero. Also, for each $u \in V - S$, we set a capacity $c(v) = \lambda$, whereas for $v \in \mathbf{A}$ the capacity $c(v) = \frac{1}{M}$. In each iteration r of the algorithm, and for each augmented sampled graph H_j, we update the lengths of nodes on the paths in $\mathcal{P}_j$ corresponding to an augmented sampled graph H_j until all the paths in $\mathcal{P}_j$ have length at least $\delta(1+\epsilon)^r$ — this value is referred to as $threshold(r)$ for the r^{th} iteration. The algorithm terminates after $r_{max} = \lfloor \log_{1+\epsilon} \frac{1+\epsilon}{\delta} \rfloor$ iterations. Since, the $threshold(r_{max})$ for the r_{max} iteration is in $[\log_{1+\epsilon} \frac{1+\epsilon}{\delta} - 1, \log_{1+\epsilon} \frac{1+\epsilon}{\delta}]$, we are guaranteed that, at termination, all paths in $\mathcal{P}$ are of length in range $[1, 1+\epsilon]$, thereby satisfying the constraints of the linear program.

Subroutine. LSearch-Saa(M, B) starts with $\lambda = \frac{1}{MB}$, uses binary search to find largest λ that has $\sum_{u \in V \setminus S} \ell(u) \leq B$; compute $\ell =$ MwuSaa(λ), and increase λ if the condition on $\sum_u \ell(u)$ is still satisfied (see full version for details).

Algorithm 2 MwuRound$(G, S, M, B, p, \epsilon)$

```
1: $\ell$ = LSearch-Saa(M, B)
2: Using the randomized rounding in [Sambaturu et al., 2020], round the fractional solution $\ell$ to an integral solution $X$
3: $\mathbf{X} = \{u : u \in V \setminus S$ and $X(u) = 1\}$ is the set of nodes picked for intervention
4: return $\mathbf{X}$
```

Analysis. We show below that Algorithm MwuSaa gives an approximate solution to $LP_\ell(\lambda)$. Theorem 1 summarizes its running time. Note that this is a factor M better than directly using the MWU technique of [Fleischer, 2000].

Theorem 1. MwuSaa *can be implemented using* $O(n \log_{1+\epsilon} \frac{1+\epsilon}{\delta})$ *shortest path computations, and has a total running time of* $\tilde{O}(\epsilon^{-2}nmM)$.

3.1 Incorporating Fairness

To adapt our approach to solve FairEpiControl, we have a separate lagrangian multiplier λ_i for group V_i, and the objective becomes $\frac{1}{M}\sum_j \sum_{v \in V_{H'_j}} y_{vj} + \sum_{i=1}^{c} \lambda_i \sum_{u \in V_i} x_u$. As in Algorithm MwuSaa, we search over the space of the λ_i's, till the budgets are all within the required bounds.

4 Algorithm MwuRound-Scalable

The main bottleneck in MwuSaa is that in each phase, it has to iterate over all the M sampled graphs, thereby needing to stores all the M graphs in memory. This can be handled by storing the sampled graphs in files and loading only one graph into memory at any time, however, this affects the total runtime. Our next algorithm MwuScalable, whose main ideas are presented below, addresses these issues.

1. Generate random stubs. The intuition behind this approach is that the actual samples do not matter, as long as we are able to generate the paths that would appear in these samples in each iteration of the algorithm. Let the probability that u is reachable from S in a sampled graph be denoted by $sp(u)$, and is referred to as stub probability. This can be estimated from our sampling process as follows: $sp(u) \approx \frac{reachable(u,S,M)}{M}$, where $reachable(u, S, M)$ denotes the number of samples in the M sampled graphs in which u is reachable from sources S. At the start of MwuScalable, we generate the random set of stubs $A_{sp}(u)$ for each node u as follows: for each $u \in V$ and $j \in 1, \cdots, M$, the stub $a(u, j)$ is generated with probability $sp(u)$. Let $\mathbf{A}_{sp} = \bigcup_u A_{sp}(u)$. The initial length $\ell(v) = \delta$ for each $v \in \mathbf{A}_{sp}$.

2. Generate sampled graphs on the fly. In every iteration r, we generate only $q \ll M$ sampled graphs, $H'_j = (V_{H'_j}, E_{H'_j})$ for $j \in [1, q]$. The algorithm then works on one sampled graph at a time, therefore, at any time stores only one sampled graph in memory.

3. Phases and iterations of the algorithm. In each phase, q iterations are performed. In each iteration q of phase r, the algorithm generates a random sampled graph H'_j. Then, independently attaches a random stub for each node reachable from sources in H'_j to form H_j. Then, it iteratively updates lengths of paths in the sampled graph H_j until there is no path of length less than $threshold(r)$. The algorithm terminates after $r_{max} = \lfloor \log_{1+\epsilon} \frac{1+\epsilon}{\delta} \rfloor$ phases (same as MwuSaa). The pseudocode of this algorithm is provided in the full version.

4. Computation of y values. Some of the $\ell(u)$ for $u \in V$ could have a value in $[1, 1+\epsilon]$. Therefore, to make the solution $\mathbf{x}$ feasible, we scale $\ell(u) = \frac{\ell(u)}{\ell_{max}}$ for $u \in V$ where $\ell_{max} = max_{u \in V} \ell(u)$. Since the sampled graphs H_r generated in each iteration are a combination of paths from many sampled graphs, the $\ell(v)$ variables for $v \in \mathbf{A}_{sp}$ will not be meaningful. Therefore, we use the $\ell(u)$ for $u \in V$ obtained after scaling, and re-compute $\ell(v)$ (corresponding to y_{vj} variables) for $v = a(u, j)$ as follows: for each sampled graph H'_j, find a shortest path tree of H'_j with S as sources using $\ell(u)$ for $u \in V \setminus S$ as weights. For each $u \in V_{H'_j}$, let P_{uj} be a shortest path to node u that has length $\ell(P_{uj})$. Then, for the stub $v = a(u, j)$, we set $\ell(v) = \min\{0, 1 - \ell(P_{uj})\}$.

Lemma 1. *The solution ℓ computed by* MwuScalable *is a feasible solution to $LP_\ell(\lambda)$.*

The search over λ values is an embarrassingly parallel task, as these searches are independent of each other. Therefore, we implement a parallel version of this algorithm.

5 Experiments

We address the following questions in our experiments:

1. Performance. What are the empirical guarantees of our methods? How does the performance of our approach compare to the baselines for this problem?

2. Impact of parameters. How do the runtime and solution quality of our methods vary with changes in transmission probability p and error parameter ϵ?

3. Scaling. How does the runtime of our approach grow with that of size of the network?

4. Parallelism. What is the throughput of our parallel implementation?

5. Cost of fairness. What is the cost of incorporating fairness constraints?

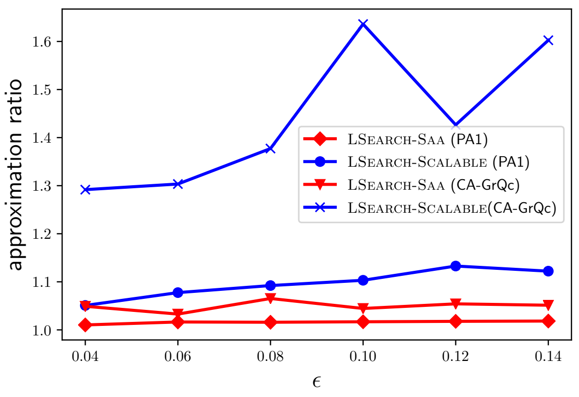

Figure 2: Comparison of approximation ratios of fractional solutions obtained by MwuRound and MwuRound-Scalable. $B = 50$.

5.1 Datasets and Methods

In our experiments, we considered networks of different classes and varying sizes (Table 3) for evaluating the performance and scalability of our approach—these range from random networks (PA1) based on the preferential attachment model [Barabási and Albert, 1999], collaboration networks, such as CA-GrQc and CA-HepTh [Leskovec *et al.*, 2007]), synthetic contact networks for Montgomery county VA and Portland city from [Sambaturu *et al.*, 2020], and for Virginia from [Chen *et al.*, 2021], and networks Regional and US-size — which are generated by connecting many copies (5 and 44 respectively) of the Virginia network using random edges, in order to study scaling.

Dataset	Nodes	Edges
Preferential1 (PA1)	1000	1996
CA-GrQc	5242	14496
CA-HepTh	9877	25998
Montgomery	75457	648667
Portland	2336693	8307767
Virginia	7605430	165533061
Regional	35024319	2068241728
US-size	334638920	32740251903

Table 3: Description of datasets.

Methods and Baselines. We consider the following methods: (1) saaRound [Sambaturu *et al.*, 2020], (2) Degree: select top B nodes in $V \setminus S$ for intervention, (3) No-Action: Baseline with no interventions, (4) MwuRound, and (5) MwuRound-Scalable and its parallel implementation.

5.2 Performance

The full version provides additional details and plots.
Figure 2 shows that the approximation ratio of fractional solutions obtained by MwuRound is within 1.2, even for $\epsilon = 0.15$. In comparison, the approximation ratio obtained by MwuRound-Scalable (for $q = 1$) is at most 1.3 for $\epsilon = 0.04$, and goes up to 1.7 for $\epsilon = 0.15$, which is within a factor of $(1 + 5\epsilon)$. MwuRound-Scalable has significantly better performance for small values of ϵ and on higher values of p (full version).

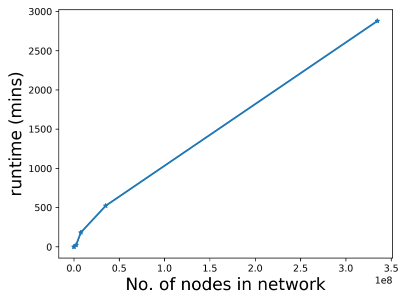

Figure 3: Runtime of MwuRound-Scalable for a fixed λ and a medium attack rate (10-20% infections in population).

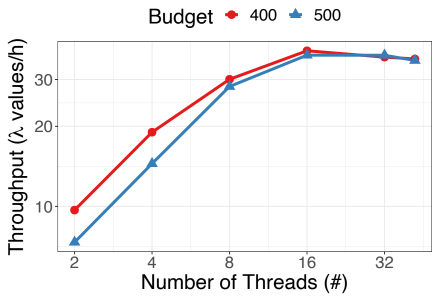

Figure 4: Number of λ values processed per hour by the parallel implementation of MwuRound-Scalable on the Virginia network.

5.3 Runtime Performance

Figure 3 presents the runtime performance (for a fixed λ) of MwuRound-Scalable on various networks. The number of λ values considered by MwuRound-Scalable algorithm determines the runtime of the algorithm. The depth of λ search decreases with the increase in the budget B. MwuRound-Scalable ran within a few minutes ($<$ 15 minutes averaged over a few runs), for a fixed λ value, on the Portland network for problem, whereas it ran in just about 2 days on the US-size network which has over 334 million nodes and 32 billion edges — for instances with a high attack rate ($> 40\%$ population infected).

5.4 Parallel Implementation

Figure 4 shows how the throughput of MwuRound-Scalable changes when increasing the number of threads and varying the budget on Virginia. The peak throughput observed, is between 16 and 32 threads, for our computing platform. At its maximum throughput, MwuRound-Scalable scales graciously with the size of the network.

5.5 Cost of Incorporating Fairness

The nodes in Montgomery network are divided into two age groups: (i) nodes with age at most 19, and (ii) nodes with age over 19. We consider two settings: (i) without fairness: budget B is allotted to all nodes, and (ii) with fairness: budget B is allotted to each group as a proportion of its size. Figure 5 shows the ratio of the expected number of infections in a fair allocation to that without fairness—observe that this ratio can be as high as 1.4, which is the price of fairness.

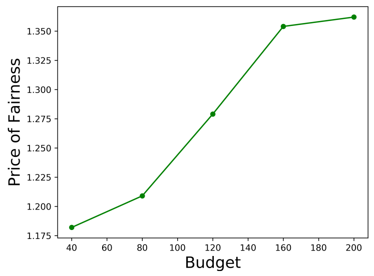

Figure 5: Cost of incorporating fairness. Montgomery.

5.6 Discussion

MwuRound-Scalable outperforms the degree baseline, and is faster than the saaRound which uses the LP solver. These runtimes can be improved using our parallel implementation. Both, MwuRound-Scalable and its parallel version, scale well for networks larger than Portland, and run even on the US-size network which has over 32 billion edges.

6 Related Work

There is a huge amount of literature on interventions for epidemic models. Due to the limited space, we only discuss network based models [Marathe and Vullikanti, 2013; Halloran *et al.*, 2008; Lofgren *et al.*, 2014; Eubank *et al.*, 2004; Germann *et al.*, 2006], which is our focus here; see full version for further details. Such models have been found to be more powerful and useful for epidemic spread on large heterogeneous populations, where the complete mixing assumptions of differential equation models are not valid. However, these are harder to set up, simulate and optimize over. Therefore, a number of heuristics have been proposed, which prioritize vaccination based on degree, centrality, or spectral properties e.g., [Cohen *et al.*, 2003; Miller and Hyman, 2007; Saha *et al.*, 2015; Minutoli *et al.*, 2020]. These do not directly give rigorous worst case guarantees; however, some of them can be computed very efficiently. The influence based approach of [Minutoli *et al.*, 2020] has been parallelized using clever hill climbing techniques.

7 Conclusions

Our algorithms are the first to give near-optimal vaccination strategies and minimize the expected number of infections in SIR models on US-scale networks. The reduced time and memory requirements result from the multiplicative weights update method adapted for percolation processes—in contrast, prior methods, which used the state-of-the-art LP solvers, were not able to scale beyond county-scale contact networks. Further improving the performance for US-scale networks is a natural question. We have focused on non-adaptive interventions; extending this to more general interventions is a challenging open problem. Finally, uncertainty is an important component of all such models. The SAA technique is easily amenable to handling some models of uncertainty. Extending our work to incorporate uncertainty is another important problem.

Acknowledgments

We would like to thank all the funding agencies for supporting this work. This work has been partially supported by the following grants: DTRA CNIMS (Contract HDTRA1-11-D-0016-0001), NSF DIBBS Grant ACI-1443054, NSF EAGER Grant CMMI-1745207, NSF BIG DATA Grant IIS-1633028, NSF Expeditions in Computing Grant CCF-1918656, NIH R01GM109718, and the Exascale Computing Project's (ECP)) (17-SC-20-SC) ExaGraph codesign center. ECP is a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. The research was in part supported by the NSF awards OAC 1910213 and CCF 1919122, and USDA NIFA award 2021-67021-35344.

References

[Anderson and May, 1991] R.M. Anderson and R.M. May. *Infectious Diseases of Humans*. Oxford University Press, Oxford, 1991.

[Arora *et al.*, 2012] S. Arora, E. Hazan, and S. Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

[Babay *et al.*, 2022] Amy Babay, Mike Dinitz, Aravind Srinivasan, Leonidas Tsepenekas, and Anil Vullikanti. Controlling epidemic spread using probabilistic diffusion models on networks. In *Proc. AISTATS*, 2022.

[Barabási and Albert, 1999] A. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[Barocas *et al.*, 2019] Solon Barocas, Moritz Hardt, and Arvind Narayanan. *Fairness and Machine Learning*. fairmlbook.org, 2019. http://www.fairmlbook.org.

[Chen *et al.*, 2021] J. Chen, S. Hoops, et al. Prioritizing allocation of covid-19 vaccines based on social contacts increases vaccination effectiveness. *medRxiv*, 2021.

[Cohen *et al.*, 2003] R. Cohen, S. Havlin, and D. ben Avraham. Efficient immunization strategies for computer networks and populations. *Phys. Rev. Lett.*, 91:247901, Dec 2003.

[Dinitz *et al.*, 2022] Mike Dinitz, Aravind Srinivasan, Leonidas Tsepenekas, and Anil Vullikanti. Fair disaster containment via graph-cut problems. In *Proc. AISTATS*, 2022.

[Eubank *et al.*, 2004] S. Eubank, H. Guclu, V. S. Anil Kumar, M. Marathe, A. Srinivasan, Z. Toroczkai, and N. Wang. Modelling disease outbreaks in realistic urban social networks. *Nature*, 429:180–184, 2004.

[Eubank *et al.*, 2006] S. Eubank, V. S. Anil Kumar, M. V. Marathe, A. Srinivasan, and N. Wang. Structure of Social Contact Networks and Their Impact on Epidemics. In *Discrete Methods in Epidemiology*, volume 70, pages 179–200. American Math. Soc., Providence, RI, 2006.

[Fleischer, 2000] L. K. Fleischer. Approximating fractional multicommodity flow independent of the number of commodities. *SIAM J. Discret. Math.*, 13(4):505–520, October 2000.

[Germann *et al.*, 2006] T.C. Germann, K. Kadau, I.M. Longini Jr, and C.A. Macken. Mitigation strategies for pandemic influenza in the united states. *PNAS*, 103(15):5935–5940, 2006.

[Halloran *et al.*, 2008] M. E. Halloran, N. M. Ferguson, et al. Modeling targeted layered containment of an influenza pandemic in the United States. In *PNAS*, pages 4639–4644, March 10 2008.

[Hayrapetyan *et al.*, 2005] A. Hayrapetyan, D. Kempe, M. Pál, and Z. Svitkina. Unbalanced graph cuts. In *ESA 2005*, ESA'05, page 191–202, Berlin, Heidelberg, 2005. Springer-Verlag.

[Kleywegt *et al.*, 2002] A. J Kleywegt, A. Shapiro, and T. Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502, 2002.

[Leskovec *et al.*, 2007] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM TKDD*, 1(1), March 2007.

[Lofgren *et al.*, 2014] E. Lofgren, M. E. Halloran, et al. Opinion: Mathematical models: A key tool for outbreak response. *PNAS*, pages 18095–18096, 2014.

[Marathe and Vullikanti, 2013] M. Marathe and A. Vullikanti. Computational epidemiology. *Communications of the ACM*, 56(7):88–96, 2013.

[Medlock and Galvani, 2009] J. Medlock and A. P. Galvani. Optimizing influenza vaccine distribution. *Science*, 325(5948):1705–1708, 2009.

[Miller and Hyman, 2007] Joel C. Miller and James M. Hyman. Effective vaccination strategies for realistic social networks. *Physica A: Statistical Mechanics and its Applications*, 386(2):780–785, 2007. Disorder and Complexity.

[Minutoli *et al.*, 2020] M. Minutoli, P. Sambaturu, M. Halappanavar, A. Tumeo, A. Kalyananaraman, and A. Vullikanti. Preempt: Scalable epidemic interventions using submodular optimization on multi-gpu systems. In *SC20*, pages 1–15, 2020.

[Saha *et al.*, 2015] S. Saha, A. Adiga, B. A. Prakash, and A. Vullikanti. Approximation algorithms for reducing the spectral radius to control epidemic spread. In *SIAM SDM*, 2015.

[Sambaturu *et al.*, 2020] Prathyush Sambaturu, Bijaya Adhikari, B. Aditya Prakash, Srinivasan Venkatramanan, and Anil Vullikanti. Designing effective and practical interventions to contain epidemics. In *AAMAS*, AAMAS '20, page 1187–1195, 2020.

[Truelove *et al.*, 2021] S. Truelove, C. P. Smith, et al. Projected resurgence of covid-19 in the united states in july—december 2021 resulting from the increased transmissibility of the delta variant and faltering vaccination. *medRxiv*, 2021.