PLOS COMPUTATIONAL BIOLOGY

RESEARCH ARTICLE

Analyzing greedy vaccine allocation algorithms for metapopulation disease models

Jeffrey Keithley, Akash Choudhuri, Bijaya Adhikari, Sriram V. Pemmaraju*

Department of Computer Science, University of Iowa, Iowa City, Iowa, United States of America

* sriram-pemmaraju@uiowa.edu

OPEN ACCESS

Citation: Keithley J, Choudhuri A, Adhikari B, Pemmaraju SV (2025) Analyzing greedy vaccine allocation algorithms for metapopulation disease models. PLoS Comput Biol 21(7): e1012539. https://doi.org/10.1371/journal.pcbi.1012539

Editor: Jennifer A. Flegg, The University of Melbourne Faculty of Science, AUSTRALIA

Received: October 4, 2024

Accepted: July 6, 2025

Published: July 21, 2025

Data availability statement: Our experimental framework, all data processing and algorithm code, and output analysis are available on Zenodo at link http://doi.org/10.5281/zenodo.13882892.

Funding: Funding for this research was provided as part of the CDC MInD Healthcare group under cooperative agreement U01CK000594 and associated Covid19 supplemental funding. Authors BA and SVP were awarded the grant and all 4 authors were

Abstract

As observed in the case of COVID-19, effective vaccines for an emerging pandemic tend to be in limited supply initially and must be allocated strategically. The allocation of vaccines can be modeled as a discrete optimization problem that prior research has shown to be computationally difficult (i.e., NP-hard) to solve even approximately. Using a combination of theoretical and experimental results, we show that this hardness result may be circumvented. We present our results in the context of a metapopulation model, which views a population as composed of geographically dispersed heterogeneous subpopulations, with arbitrary travel patterns between them. In this setting, vaccine bundles are allocated at a subpopulation level, and so the vaccine allocation problem can be formulated as a problem of maximizing an integer lattice function $g : \mathbb{Z}_+^K \to \mathbb{R}$ subject to a budget constraint $\|\boldsymbol{x}\|_1 \leq D$. We consider a variety of simple, well-known greedy algorithms for this problem and show the effectiveness of these algorithms for three problem instances at different scales: New Hampshire (10 counties, population 1.4 million), Iowa (99 counties, population 3.2 million), and Texas (254 counties, population 30.03 million). We provide a theoretical explanation for this effectiveness by showing that the approximation factor (a measure of how well the algorithmic output for a problem instance compares to its theoretical optimum) of these algorithms depends on the *submodularity ratio* of the objective function *g*. The submodularity ratio of a function is a measure of how distant *g* is from being submodular; here submodularity refers to the very useful "diminishing returns" property of set and lattice functions, i.e., the property that as the function inputs are increased the function value increases, but not by as much.

Author summary

Strategic and timely allocation of vaccines is crucial in combating epidemic outbreaks. Developing strategies to allocate vaccines over subpopulations rather than to individuals leads to policy recommendations that are more feasible in practice. Despite this,

supported by the grant https://www.cdc.gov/hai/research/MIND-Healthcare.html. Additional funding was provided by NSF Award Number 1955939. Author SVP was awarded this grant and authors JK and SVP were supported by the grant. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

vaccine allocation over subpopulations has only received limited research interest, and the associated computational challenges are relatively unknown. To address this gap, we study vaccine allocation problems over geographically distinct subpopulations in this paper. We formulate our problems to reduce either **i)** the total infections or **ii)** the sum of peak infections over meta-population disease models. We first demonstrate that these problems are computationally challenging even to approximate and then show that a family of simple, well-known greedy algorithms exhibit provable guarantees. We conduct realistic experiments on state-level mobility graphs derived from real-world data in three states of distinct population levels: New Hampshire, Iowa, and Texas. Our results show that the greedy algorithms we consider are **i)** scalable and **ii)** outperform both state-of-the-art and natural baselines in a majority of settings.

Introduction

In the early stages of a pandemic like COVID-19, the demand for vaccinations far exceeds supply [1,2] and it is critical to strategically allocate vaccines [3,4]. The vaccine allocation problem can be modeled in a variety of ways, including as discrete optimization problems [5–9].

However, all of these problems are computationally hard, even to solve approximately (see [10], for a specific example). Despite these obstacles, we need to be able to solve vaccine allocation problems at scale and have confidence that the obtained solutions are close to being optimal. In this paper, we take steps towards this goal.

We consider the metapopulation-network model for disease-spread [11,12], which allows for heterogeneity among geographically distinct subpopulations and arbitrary travel patterns between them. Vaccine allocation within this model consists of allocating some number of bundles of vaccines to each subpopulation while satisfying an overall budget constraint. The resulting family of problems, which we call the *Metapopulation Vaccine Allocation (MVA)* problems, can be formalized as maximizing an objective function (e.g., number of cases averted) defined over an integer lattice domain subject to a budget constraint. Here, the integer lattice refers to the set of all possible vaccine allocations, where each allocation assigns an integer number of vaccines to each subpopulation. Not surprisingly, we show specific problems obtained via realistic instantiations of the metapopulation-network model and objective function in MVA are not just NP-hard (i.e., computationally intractable in general), but even hard to approximate. We show these hardness results for two instantiations, which we call MaxCasesAverted and MaxPeaksReduced, of MVA over SEIR (**S**usceptible-**E**xposed-**I**nfected-**R**ecovered) metapopulation models [11,12]. These hardness of approximation results imply that worst-case approximation guarantees are not attainable for natural instantiations of MVA. However, for a family of simple, well-known greedy algorithms, we show positive theoretical and experimental results for both MaxCasesAverted and MaxPeaksReduced. These simple and natural greedy algorithms lend themselves to the machinery from submodular function optimization for in-depth analysis. There is a rich literature of methods for submodular set function optimization [13–18] that has subsequently been extended to submodular optimization over the integer lattice [19–22]. In general, submodularity is a valuable property for a function being maximized, as it imposes structured behavior when elements are added to its input, enabling us to bound how far the value of a greedy solution is from the optimal value. Furthermore, in the last few years, researchers have attempted to extend some of the aforementioned results for submodular set and lattice function optimization to functions that are not submodular, by using the notion of

submodularity ratio of a function, which is a measure of how distant that function is from being submodular [23–25]. All of this literature is foundational to our approach to analyzing vaccine allocation algorithms in a metapopulation model setting [11,12].

In our main theoretical result, we show that simple greedy algorithms provide worst-case approximation guarantees for MaxCasesAverted and MaxPeaksReduced that become better as the submodularity ratio of their objective functions approaches 1. The submodularity ratio [23–26] of a set or lattice function is a measure (between 0 and 1) of how close the function is to being submodular, with values closer to 1 corresponding to functions that are closer to being submodular. We complement this theoretical result with experimental results indicating that the objective functions for MaxCasesAverted and MaxPeaksReduced might have relatively high submodularity ratios.

We then experimentally evaluate the performance of greedy vaccine allocation algorithms at three scales; we use New Hampshire (10 counties, population 1.4 million) for our small scale experiments, Iowa (99 counties, population 3.2 million) for our medium scale experiments, and Texas (254 counties, population 30.03 million) for our large scale experiments. We compare the performance of the greedy methods with a set of trivial baselines, such as allocating vaccines according to population sizes. We also compare against a randomized algorithm called Pareto Optimization for Subset Selection (POMS) [24]. POMS works by expanding a random pareto-optimal frontier (i.e., interpreting solution size as a second objective function and finding solutions which balance the quality of both objective functions), and was designed to compete against greedy algorithms for small scale problems. We show the greedy algorithms we consider outperform POMS for our experimental settings, while scaling more readily. Our experiments demonstrate that **i)** simple greedy vaccine allocation algorithms outperform the natural baseline algorithms substantially (up to 9M more individuals saved than the worst-performing baseline in some settings), **ii)** for both MaxCasesAverted and MaxPeaksReduced, greedy algorithms perform near-optimally for most problem instances we evaluate for New Hampshire (and recover similar approximation guarantees to those of submodular functions for experiments in Iowa and Texas), and **iii)** the fastest of our greedy algorithms are feasible even for large scale instances such as the state of Texas.

Materials and methods

Background

Notation. We use $\mathbb{Z}_+$ to denote the set of non-negative integers and for any positive integer n, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$.

Metapopulation disease-spread models. A *metapopulation* disease-spread model [11] generalizes the classic homogeneous-mixing compartmental models [27] by allowing geographically-diverse subpopulations. Let $K \in \mathbb{Z}_+$ denote the number of subpopulations in the metapopulation model. For each subpopulation $i \in [K]$, let n_i denote the size of the subpopulation and let $\mathbf{n}$ denote the vector $(n_1, n_2, \ldots, n_K)$ of subpopulation sizes. For each pair $(i, j) \in [K] \times [K]$, let $w_{ij} \in \mathbb{Z}_+$ denote the number of individuals moving from subpopulation i to subpopulation j daily. Thus, each w_{ij} is a static (i.e., time independent) quantity. Let $\mathbf{W}$ denote the $K \times K$ mobility matrix induced by the w_{ij} values.

Our goal is to decrease the spread of disease by allocating a total of $D \in \mathbb{Z}_+$ bundles of vaccines to individuals over all subpopulations; here D is the *vaccine budget*. A bundle can be viewed as the smallest "shipment" of vaccines that can be allocated to a subpopulation and we assume that each bundle consists of an integer $\Delta > 0$ number of individual vaccines. Let $\mathbf{x} = (v_1, \ldots, v_K) \in \mathbb{Z}_+^K$ denote a vaccine allocation, where v_i is the number of bundles of

vaccines allocated to subpopulation i. For simplicity, we assume that vaccination is pre-emptive, i.e., occurs at time 1, with knowledge of initial infected, but before the disease has started to spread. It is straightforward to generalize this to a setting in which vaccine allocation occurs later in the progression of the disease. Let $\mathbf{I} = (I_1^0, I_2^0, \ldots, I_K^0) \in \mathbb{Z}_+^K$, where $0 \le I_i^0 \le n_i$, denote the number of initial infections in subpopulation i. Let $f(\boldsymbol{x}|\mathcal{M}, \mathbf{I})$ denote some measure of disease-spread according to the metapopulation model $\mathcal{M}$ starting with initial infection vector $\mathbf{I}$, expressed as a function of the vaccine allocation vector $\boldsymbol{x}$. For example, $f(\mathbf{x}|\mathcal{M}, \mathbf{I})$ could denote the total number of infected individuals over some time window. Let $g(\mathbf{x}|\mathcal{M}, \mathbf{I})$ denote $f(\mathbf{0}|\mathcal{M}, \mathbf{I}) - f(\mathbf{x}|\mathcal{M}, \mathbf{I})$, representing the reduction in disease-spread due to vaccine allocation $\boldsymbol{x} \in \mathbb{Z}_+^K$, relative to the no-vaccine setting. Note that both f and g are defined over the integer lattice $\mathbb{Z}_+^K$ and our goal is to maximize the integer lattice function $g(\boldsymbol{x}|\mathcal{M}, \mathbf{I})$ subject to the cardinality constraint $||\boldsymbol{x}||_1 \le D$.

Submodularity of lattice functions For $K \in \mathbb{Z}_+$, let $g : \mathbb{Z}_+^K \to \mathbb{R}$ be a function defined on an integer lattice domain. The function g is said to be *submodular* if for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_+$

$$g(\boldsymbol{x}) + g(\boldsymbol{y}) \ge g(\boldsymbol{x} \vee \boldsymbol{y}) + g(\boldsymbol{x} \wedge \boldsymbol{y}) \tag{1}$$

Here $(\boldsymbol{x} \vee \boldsymbol{y})_i = \max\{\boldsymbol{x}_i, \boldsymbol{y}_i\}$ and $(\boldsymbol{x} \wedge \boldsymbol{y})_i = \min\{\boldsymbol{x}_i, \boldsymbol{y}_i\}$.

Below we provide an alternate "diminishing returns" notion of submodularity that is easier to work with. Here $\boldsymbol{e}_i$ denotes the unit vector with 1 in coordinate i.

Definition 1. *[21] **(DR-Submodularity)** A function $g : \mathbb{Z}_+^K \to \mathbb{R}$ is said to be diminishing returns submodular (DR-submodular, in short) if $g(\boldsymbol{x} + \boldsymbol{e}_i) - g(\boldsymbol{x}) \ge g(\boldsymbol{y} + \boldsymbol{e}_i) - g(\boldsymbol{y})$ for all $i \in [K]$ and $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_+^K$, where $\boldsymbol{x} \le \boldsymbol{y}$.*

For set functions, submodularity and DR-submodularity are equivalent. However, it is known [20] that if a lattice function is DR-submodular then it is submodular, but the converse is false. Thus, DR-submodularity is a stronger notion compared to submodularity. However, [24] presents a DR-type characterization of submodular lattice functions that is quite useful for our analysis.

Lemma 2. *[24] A function $g : \mathbb{Z}_+^K \to \mathbb{R}$ is submodular if and only if for any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_+^K$, $\boldsymbol{x} \le \boldsymbol{y}$ and $i \in [K]$ with $\boldsymbol{x}_i = \boldsymbol{y}_i$, $g(\boldsymbol{x} + \boldsymbol{e}_i) - g(\boldsymbol{x}) \ge g(\boldsymbol{y} + \boldsymbol{e}_i) - g(\boldsymbol{y})$.*

Note that according to this lemma, for submodular lattice functions, the DR property is only required to hold at identical coordinates of $\boldsymbol{x}$ and $\boldsymbol{y}$. The computational complexity of maximizing a submodular lattice function $g : \mathbb{Z}_+^K \to \mathbb{R}$ subject to a cardinality constraint, namely $\max_{||\boldsymbol{x}||_1 \le D} g(\boldsymbol{x})$, is well understood. [20] extend the result for set functions from [28] to lattice functions and show that greedy approaches yield a $(1 - \frac{1}{e})$-approximation for this problem for both submodular and DR-submodular lattice functions (an algorithm is said to achieve an α-approximation for a maximization problem if it always produces a solution whose objective value is at least an α fraction ($0 \le \alpha \le 1$) of the optimal value). These approximation guarantees are optimal due to the inapproximability result of [29], meaning that no approximation algorithm can achieve a higher constant-factor guarantee.

The SEIR metapopulation model

The SEIR equations are governed by parameters λ, η, and δ, where λ is the *infectivity*, $1/\eta$ is the *latency period*, and $1/\delta$ is the *infectious period*. Let r_i denote a multiplier that scales λ to allow for county differences in contact rates. Let T be a positive integer denoting the size of the time window under consideration. For $t \in [T] \cup \{0\}$, each subpopulation is split into

compartments S_i^t, E_i^t, I_i^t, and R_i^t representing the number of susceptible, exposed, infected, and recovered individuals within subpopulation i at time t. We assume the initial conditions $E_i^0 = R_i^0 = 0$, I_i^0 is an arbitrary non-negative number satisfying $I_i^0 \leq n_i$, and $S_i^0 = n_i - I_i^0$. The evolution of S_i^t, E_i^t, I_i^t, and R_i^t over time t is respectively governed by Eqs 2–5. The term q_i^t that appears in these equations is called the *force of infection*. When $q_i^t = \lambda r_i \frac{I_i^t}{n_i}$, Eqs 2–5 represent the spread of disease in a single subpopulation i with a homogeneous mixing assumption.

$$S_i^{t+1} = S_i^t - q_i^t S_i^t \tag{2}$$

$$E_i^{t+1} = E_i^t + q_i^t S_i^t - \eta E_i^t \tag{3}$$

$$I_i^{t+1} = I_i^t + \eta E_i^t - \delta I_i^t \tag{4}$$

$$R_i^{t+1} = R_i^t + \delta I_i^t \tag{5}$$

We use the following expression for the force of infection term q_i^t that takes the infection incidence within subpopulation i along with flows of individuals into and out of subpopulation i. The derivation of q_i^t is inspired by a similar derivation in [5,12] and is included in Section A in S1 Text.

$$q_i^t = \lambda \left[r_i \left(1 - \sum_j \frac{w_{ij}}{n_i} \right) \frac{\hat{I}_i^t}{\hat{n}_i} + \sum_j \frac{w_{ij} r_j}{n_i} \frac{\hat{I}_j^t}{\hat{n}_j} \right] \tag{6}$$

$\hat{n}_i$ denotes the *effective* population of subpopulation i at time t, describing the number of individuals present in subpopulation i after a daily commute has occurred, and $\hat{I}_i^t$ denotes the effective number of infected individuals in subpopulation i after a commute. The first term in the right hand side of Eq 6 is the proportion of individuals leaving subpopulation i for their commute, and the second term is the proportion of individuals arriving.

The SEIR metapopulation model $\mathcal{M}$ described above is completely specified by the vector $(\mathbf{n}, \mathbf{r}, \mathbf{W}, T, \lambda, \eta, \delta)$. In our experiments, each subpopulation represents a county within a state (e.g., $K = 99$ for Iowa) and the mobility matrix $\mathbf{W}$ is obtained from two independent sources, FRED [30] and SafeGraph [31]. By instantiating a specific disease-spread model for each subpopulation and describing its interaction with mobility matrix $\mathbf{W}$, we can obtain a completely specified metapopulation model.

Table 1 summarizes the notation introduced in this section.

Problem formulations

We are now ready to state the *Metapopulation Vaccine Allocation* (MVA) family of problems.

Table 1. Metapopulation model notation.

Variable	Definition
K,T	Number of subpopulations, size of time window
r_i	Population density correlated λ-multiplier for subpopulation i
n_i	Size of subpopulation i
w_{ij}	Mobility from subpopulations i to j
q_i^t	Force of infection in subpopulation i at time t
$\lambda, 1/\eta, 1/\delta$	Infectivity, latency period, infectious period

https://doi.org/10.1371/journal.pcbi.1012539.t001

MVA
Given a metapopulation model $\mathcal{M}$, initial infected vector $\mathbf{I} = (I_1^0, I_2^0, \ldots, I_K^0) \in \mathbb{Z}_+^K$, where $0 \le I_i^0 \le n_i$, and a vaccine budget $D \in \mathbb{Z}_+$, find a vaccine allocation $\mathbf{x} \in \mathbb{Z}_+^K$, satisfying $\|\boldsymbol{x}\|_1 \le D$ such that $g(\boldsymbol{x}|\mathcal{M}, \mathbf{I}) := f(\mathbf{0}|\mathcal{M}, \mathbf{I}) - f(\mathbf{x}|\mathcal{M}, \mathbf{I})$ is maximized.

SEIR metapopulation vaccine allocation problems. For illustrative purposes, we instantiate the general metapopulation model $\mathcal{M}$ with an SEIR model for disease spread within each subpopulation. Our framework is general and the SEIR model that we use within subpopulations can be replaced by any other homogeneous-mixing disease spread model.

Using the SEIR metapopulation model described above, we obtain specific instances of the MVA problem. But before we can describe these specific instances, we need to describe how vaccination affects disease spread in the SEIR metapopulation model. For simplicity, we assume that vaccine uptake and vaccine effectiveness are both perfect, and thus allocating a vaccine bundle $\mathbf{x} = (v_1, \ldots, v_K)$ implies that $\Delta \cdot v_i$ individuals in subpopulation i are vaccinated and removed from S_i^0. Thus the vaccine allocation $\mathbf{x} = (v_1, \ldots, v_K)$ updates the initial susceptible to $S_i^0 = \max(0, n_i - I_i^0 - \Delta \cdot v_i)$ for all $i \in [K]$. The assumptions of perfect uptake and effectiveness are easily relaxed; lowering the vaccine uptake or effectiveness is equivalent to allocating fewer vaccines.

We now present two illustrative problems that maximize the impact of vaccines according to different disease spread metrics. In the problem MaxCasesAverted, the metric is the total number of infections averted across all subpopulations, and in the problem MaxPeaksReduced, the metric is the decrease in the sum of all infection peaks across all subpopulations (both taken over the entire simulation time). More precisely, given an SEIR metapopulation model $\mathcal{M} = (\mathbf{n}, \mathbf{r}, \mathbf{W}, T, \lambda, \eta, \delta)$, initial infected vector $\mathbf{I} = (I_1^0, I_2^0, \ldots, I_K^0) \in \mathbb{Z}_+^K$, where $0 \le I_i^0 \le n_i$, and a vaccine allocation $\mathbf{x} = (v_1, \ldots, v_K) \in \mathbb{Z}_+^K$, we define the metric

$$\textsc{totBurden}(\mathbf{x}|\mathcal{M}, \mathbf{I}) := \sum_{k \in [K]} (R_k^T + I_k^T),$$

which is simply the total number of individuals who became infected in the time window $[0,T]$. Another natural disease spread metric for the SEIR metapopulation model is

$$\textsc{maxBurden}(\mathbf{x}|\mathcal{M}, \mathbf{I}) := \sum_{k \in [K]} \max_{0 \le t \le T} I_k^t,$$

which is the total number of individuals infected during "peak" infection time over all the subpopulations. This metric is motivated by the fact that even small peaks are challenging in low-resource counties (typically in low-population counties), because healthcare infrastructure is often limited in such counties. So even a small spike in the number of infected individuals can quickly overwhelm local resources. Thus we seek to reduce the likelihood that local healthcare systems will be overwhelmed with the maxBurden metric.

Given metapopulation model $\mathcal{M}$, initial infection vector $\boldsymbol{I}$, and budget D, we define the following discrete optimization problems:

MaxCasesAverted
Find a vaccine allocation $\mathbf{x} \in \mathbb{Z}_+^K$, satisfying $\|\boldsymbol{x}\|_1 \le D$ such that the following is maximized.

$$\textsc{totBurden}(\mathbf{0}|\mathcal{M}, \mathbf{I}) - \textsc{totBurden}(\mathbf{x}|\mathcal{M}, \mathbf{I})$$

MaxPeaksReduced
Find a vaccine allocation $\mathbf{x} \in \mathbb{Z}_+^K$, satisfying $\|\boldsymbol{x}\|_1 \leq D$ such that the following is maximized.

$$\textsc{maxBurden}(\mathbf{0}|\mathcal{M}, \mathbf{I}) - \textsc{maxBurden}(\mathbf{x}|\mathcal{M}, \mathbf{I})$$

Hardness of MaxCasesAverted and MaxPeaksReduced

As with many resource allocation problems, both MaxCasesAverted and MaxPeaksReduced are not only NP-hard, but even hard to efficiently approximate (additional background on NP-hardness may be found in [32]). The purpose of this section is to formally establish that the MVA problem is too hard to solve exactly in general, making it necessary to use approximation algorithms. We show this by a reduction from the *Maximum k-Subset Intersection* (Max κ-SI) problem [33]. The input to Max κ-SI consists of a collection $\mathcal{C} = \{P_1, P_2, \ldots, P_m\}$ of sets, where each set P_j is a subset of a universe $\mathcal{U} = \{p_1, p_2, \ldots, p_n\}$, and a positive integer k . The problem seeks to find k subsets $P_{j_1}, P_{j_2}, \ldots, P_{j_k}$ from $\mathcal{C}$, whose intersection has maximum size. The following theorem from [33] shows that Max κ-SI is highly unlikely to have an efficient approximation algorithm, even with an inverse polynomial approximation factor.

Theorem 3. *[33] Let $\epsilon > 0$ be an arbitrarily small constant. Assume that SATISFIABILITY does not have a probabilistic algorithm that decides whether a given instance of size n is satisfiable in time $2^{n^{\epsilon}}$. Then there is no polynomial time algorithm for Max κ-SI that achieves an approximation ratio of $1/N^{\epsilon'}$, where N is the size of the given instance of Max κ-SI and ϵ' only depends only on ϵ.*

We now show a reduction from Max κ-SI to both MaxCasesAverted and MaxPeaksReduced, thereby establishing the inapproximability of both of these problems.

Theorem 4. *Let $\epsilon > 0$ be an arbitrarily small constant. Assume that SATISFIABILITY does not have a probabilistic algorithm that decides whether a given instance of size n is satisfiable in time $2^{n^{\epsilon}}$. Then there is no polynomial time algorithm for MaxCasesAverted or for MaxPeaksReduced that achieves an approximation ratio of $1/N^{\epsilon'}$, where N is the size of the given instance of MaxCasesAverted or MaxPeaksReduced and ϵ' only depends only on ϵ.*

Proof: To prove the portion of this theorem pertaining to MaxCasesAverted, we show the following lemma.

Lemma 5. *Suppose there is a polynomial-time algorithm $\mathcal{A}$ that yields an α-approximation for MaxCasesAverted. Then there is a polynomial-time $\alpha/2$-approximation algorithm $\mathcal{A}'$ for Max κ-SI.*

Proof of Lemma 5. Given an instance $(\mathcal{C}, \mathcal{U}, k)$ of Max κ-SI, we construct the graph G with $m + n + 1$ nodes. For each subset $P_j \in \mathcal{C}$ and each $p_i \in \mathcal{U}$, there is a node in G, for a total of $m + n$ nodes. There is an extra node I that is connected to every P_j-node. There are edges between the P_j-nodes and the p_i-nodes connecting an P_j-node to an p_i-node iff $p_i \notin P_j$.

To each node v in G, we assign a population n_v as follows: $n_I = m$, $n_{P_j} = 2n$ for all $j \in [n]$, and $n_{p_i} = M$ for all $i \in [n]$, where M is a large integer whose value will be specified later.

We then interpret each undirected edge in G as a pair of directed edges pointing in opposite directions and assign a flow to each directed edge. We assign flow 1 to each edge from I to

P_j and to each edge from P_j to p_i. To all other edges, i.e., the edges pointing "backwards", we assign flow 0. This construction is illustrated in Fig 1. This specifies the vectors $\boldsymbol{n}$ and $\boldsymbol{w}$ of the instance of MaxCasesAverted.

We set the contact rate r_v and infectivity λ such that the force of infection q_v^t is always at least 1. This corresponds to "perfect infectivity", meaning that if a subpopulation contains some infected and some susceptible individuals at a time step, then *all* the susceptible individuals in the subpopulation will transition to the exposed state at the next time step. We then set $\eta = \delta = 1$ so that the latency period and recovery period are both 1. With this setting of the parameters, the infection will completely die out in 5 time steps, i.e., every individual will either be susceptible or recovered. So we set the size of the time window $T = 5$.

Finally, we set the vaccination budget $D = (m-k) \cdot 2n$ and initialize the entire population of m individuals at node I to be infected and all other individuals to be susceptible. This completes the specification of the problem instance $\mathcal{I}$ of MaxCasesAverted.

We now make 2 simple observations that follow from the construction of $\mathcal{I}$ and depend on the notion of being "unprotected" with respect to a vaccine allocation. Let $\mathbf{x}$ be an arbitrary, feasible allocation for $\mathcal{I}$. A subpopulation P_j is called *unprotected for* $\boldsymbol{x}$ if $\mathbf{x}_{P_j} < 2n$; otherwise, P_j is called *protected for* $\boldsymbol{x}$. A subpopulation p_i is called *unprotected for* $\boldsymbol{x}$ if $\mathbf{x}_{p_i} < M$ and for some subpopulation P_j that is unprotected for $\mathbf{x}$, the edge $\{P_j, p_i\}$ is in G; otherwise, p_i is called *protected for* $\boldsymbol{x}$.

Observation 1: In every unprotected subpopulation P_j, $j \in [m]$, $2n - \mathbf{x}_{P_j}$ individuals will become exposed in time step 1 and infected in time step 2.

Observation 2: In every unprotected subpopulation p_i, $i \in [n]$, $M - \mathbf{x}_{p_i}$ individuals will become exposed in time step 3 and infected in time step 4.

These 2 observations immediately lead to the following 3 claims.

Claim i) Consider a vaccine allocation $\mathbf{x} \in \mathbb{Z}_+^{m+n+1}$ that is feasible for $\mathcal{I}$ and satisfies $\mathbf{x}_{p_i} > 0$. Let $\mathbf{x}' \in \mathbb{Z}_+^{m+n+1}$ be an allocation obtained from $\mathbf{x}$ by reallocating all vaccines from the subpopulation p_i to subpopulations P_j, $j \in [m]$. Then $\mathbf{x}'$ is feasible for $\mathcal{I}$ and

$$\text{totBurden}(\mathbf{x}'|\mathcal{M}, \mathbf{I}) \leq \text{totBurden}(\mathbf{x}|\mathcal{M}, \mathbf{I}).$$

Claim ii) Consider a vaccine allocation $\mathbf{x} \in \mathbb{Z}_+^{m+n+1}$ that is feasible for $\mathcal{I}$ and satisfies $0 < \mathbf{x}_{P_j}, \mathbf{x}_{P_{j'}} < 2n$ for two subpopulations $P_j, P_{j'}, j \neq j'$. Let $\mathbf{x}' \in \mathbb{Z}_+^{m+n+1}$ be an allocation obtained

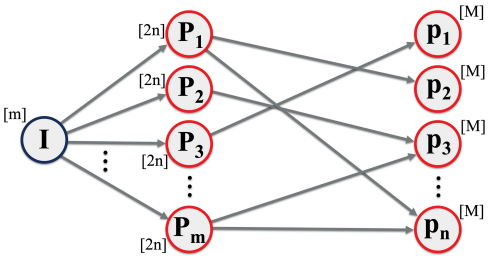

Fig 1. The instance of MaxCasesAverted and MaxPeaksReduced is a graph G constructed from the given instance $(\mathcal{C} = (P_1, P_2, \ldots, P_m), \mathcal{U} = (p_1, p_2, \ldots, p_n), k)$ of Max κ-SI. Each node represents a subpopulation, with the size of the subpopulation shown in square brackets next to it. The directed edges permit 1 unit flow. The unit flows from nodes P_j to p_i encode non-membership. For example, the flow from P_1 to p_2 implies that $p_2 \not\in P_1$.

https://doi.org/10.1371/journal.pcbi.1012539.g001

from $\mathbf{x}$ by reallocating as many vaccines as possible from the subpopulation $P_{j'}$ to the subpopulation P_j, until $\mathbf{x}_{P_j} = 2n$ or $\mathbf{x}_{P_{j'}} = 0$ (or both). Then $\mathbf{x}'$ is feasible for $\mathcal{I}$ and

$$\textsc{totBurden}(\mathbf{x}'|\mathcal{M},\mathbf{I}) \leq \textsc{totBurden}(\mathbf{x}|\mathcal{M},\mathbf{I}).$$

Claim iii) Consider a vaccine allocation $\mathbf{x} \in \mathbb{Z}_+^{m+n+1}$ that is feasible for $\mathcal{I}$ and satisfies $||x||_1 = D = (m-k)\cdot 2n$. Then using the reallocations from Claims (i) and (ii), it is possible to transform $\mathbf{x}$ into $\mathbf{x}' \in \mathbb{Z}_+^{m+n+1}$ in polynomial time such that $\mathbf{x}'$ is feasible for $\mathcal{I}$, $\mathbf{x}'_{P_j} = 2n$ for exactly $(m–k)$ subpopulations P_j, $\mathbf{x}'$ is 0 for all other subpopulations, and

$$\textsc{totBurden}(\mathbf{x}'|\mathcal{M},\mathbf{I}) \leq \textsc{totBurden}(\mathbf{x}|\mathcal{M},\mathbf{I}).$$

Claim (iii) allows us to assume that any α-approximation algorithm $\mathcal{A}$ for MaxCasesAverted returns an allocation $\mathbf{x}'$ for the problem instance $\mathcal{I}$, that picks exactly $(m–k)$ subpopulations P_j and vaccinates these subpopulations entirely, while allocating no vaccines to any of the remaining subpopulations. Similarly, Claim (iii) implies that there is an optimal allocation $\mathbf{x}^*$ for $\mathcal{I}$ that picks exactly $(m–k)$ subpopulations P_j and vaccinates these subpopulations entirely, while allocating no vaccines to any of the remaining subpopulations.

Let $\mathcal{S}(\mathbf{x}')$ be the set of subpopulations P_j unprotected for $\mathbf{x}'$. Similarly, define $\mathcal{S}(\mathbf{x}^*)$. Note that $|\mathcal{S}(\mathbf{x}')| = |\mathcal{S}(\mathbf{x}^*)| = k$. Let $\mathcal{E}(\mathbf{x}')$ be the set of subpopulations p_i that are protected for $\mathbf{x}'$. Similarly, define $\mathcal{E}(\mathbf{x}^*)$. By the construction of edges from subpopulations P_j to subpopulations p_i in $\mathcal{I}$, it follows that $\mathcal{E}(\mathbf{x}') = \cap_{P_j \in \mathcal{S}(\mathbf{x}')} P_j$. Similarly, $\mathcal{E}(\mathbf{x}^*) = \cap_{P_j \in \mathcal{S}(\mathbf{x}^*)} P_j$

The objective function value of MaxCasesAverted for the optimal allocation $\mathbf{x}^*$, which is $\textsc{totBurden}(\mathbf{0}|\mathcal{M},\mathbf{I}) - \textsc{totBurden}(\mathbf{x}^*|\mathcal{M},\mathbf{I})$, can be simplified to

$$\begin{aligned} & (2n\cdot m + n\cdot M) - \textsc{totBurden}(\mathbf{x}^*|\mathcal{M},\mathbf{I}) \\ = \;& (2n\cdot m + n\cdot M) - (k\cdot 2n + M(n - |\mathcal{E}(\mathbf{x}^*)|)) \\ = \;& 2n(m-k) + M\cdot|\mathcal{E}(\mathbf{x}^*)| \\ = \;& 2n(m-k) + M\cdot|\cap_{P_j\in\mathcal{S}(\mathbf{x}^*)} P_j| \end{aligned} \tag{7}$$

Similarly, the objective function value of MaxCasesAverted for the α-approximate allocation $\mathbf{x}'$ is $2n(m-k) + M\cdot|\cap_{P_j\in\mathcal{S}(\mathbf{x}')} P_j|$. Since $\mathbf{x}^*$ maximizes the objective function value of MaxCasesAverted, Eq 7 implies that $|\cap_{P_j\in\mathcal{S}(\mathbf{x}^*)} P_j|$ has largest possible cardinality. Since $|\mathcal{S}(\mathbf{x}^*)| = k$, this implies that $\mathcal{S}(\mathbf{x}^*)$ is an optimal solution to the Max κ-SI problem. Using $OPT_{\text{Max }\kappa\text{-SI}}$ to denote the optimal objective function value of Max κ-SI, we can rewrite the expression (7) as $2n(m-k) + M\cdot OPT_{\text{Max }\kappa\text{-SI}}$. Since $\mathbf{x}'$ is an α-approximate solution to MaxCasesAverted,

$$2n(m-k) + M\cdot|\cap_{P_j\in\mathcal{S}(\mathbf{x}')} P_j| \geq \alpha\left(2n(m-k) + M\cdot OPT_{\text{Max }\kappa\text{-SI}}\right).$$

Rearranging terms we get

$$\begin{aligned} |\cap_{P_j\in\mathcal{S}(\mathbf{x}')} P_j| &\geq \alpha\cdot OPT_{\text{Max }\kappa\text{-SI}} - \frac{(1-\alpha)\cdot 2n(m-k)}{M} \\ |\cap_{P_j\in\mathcal{S}(\mathbf{x}')} P_j| &\geq \alpha\cdot OPT_{\text{Max }\kappa\text{-SI}} - \frac{2nm}{M} \end{aligned} \tag{8}$$

Picking M large enough so that $\frac{2nm}{M} \leq \frac{\alpha}{2}$ and using $OPT_{\text{Max } \kappa\text{-SI}} \geq 1$, we obtain

$$\left| \cap_{P_j \in \mathcal{S}(\mathbf{x}')} P_j \right| \geq \frac{\alpha}{2} \cdot OPT_{\text{Max } \kappa\text{-SI}}.$$

This implies that the allocation $\mathbf{x}'$ can be used to obtain an $\alpha/2$-approximation to Max κ-SI.

We now prove a similar lemma for the MaxPeaksReduced problem.

Lemma 6. *Suppose there is a polynomial-time algorithm $\mathcal{A}$ that yields an α-approximation for MaxPeaksReduced. Then there is a polynomial-time α-approximation algorithm $\mathcal{A}'$ for Max κ-SI.*

Proof of Lemma 6. This uses the same argument as the lemma above. Claims (i) and (ii) hold for $\text{maxBurden}(\mathbf{x}|\mathcal{M}, \mathbf{I})$ as well and from these two claims, Claim (iii) follows. Furthermore,

$$\text{maxBurden}(\mathbf{0}|\mathcal{M}, \mathbf{I}) - \text{maxBurden}(\mathbf{x}^*|\mathcal{M}, \mathbf{I})$$

simplifies exactly to expression (7), from which inequality (8) follows. From this, the lemma immediately follows, as shown above. □

Algorithmic approach and analysis

We present the following four greedy algorithms for MVA:

1. **ℓ-EnumGreedy**: Enumerates all feasible solutions with at most ℓ subpopulations allocated a positive number of vaccine bundles. Then each of these solutions is iteratively extended greedily, one subpopulation at a time, with a variable number of additional vaccine shipments. Because of the potentially large number of initial feasible solutions, this is only suitable for small and medium-scale problems.
2. **SingletonGreedy**: Finds one solution by extending the empty solution greedily, one subpopulation at a time, until D shipments are allocated. Compares this solution to the K solutions by allocating the entire budget D to each of the K subpopulations. Returns the best of these $K + 1$ solutions.
3. **FastGreedy**: Runs a "relaxed" version of greedy that stops its search as soon as it finds a "good enough" additional allocation of vaccine shipments. The threshold for a "good enough" allocation is adaptive, i.e., may change from iteration to iteration. This algorithm trades off solution quality for speed and is suitable for large-scale problems.
4. **UnitGreedy**: Starting from the empty allocation, greedily allocates just one vaccine shipment at a time. Searching over a space of just one additional vaccine shipment per subpopulation, speeds up each iteration. This algorithm is suitable for large-scale problems.

In the following, we first describe these algorithms in further detail and then we establish approximation guarantees for ℓ-EnumGreedy, SingletonGreedy, and UnitGreedy based on how close the objective function is to being submodular. These algorithms and their accompanying analyses also apply to the general budget-constrained maximization problem on an integer lattice: $\max_{\|\boldsymbol{x}\|_1 \leq D} g(\boldsymbol{x})$, where $g : \mathbb{Z}_+^K \to \mathbb{R}$ is an arbitrary, monotone function defined on an integer lattice.

We start by defining the LatticeGreedySubroutine, whose search space is the entire lattice $\mathbb{Z}_+^K$ in each iteration. This subroutine forms the basis for two greedy algorithms ℓ-EnumGreedy and SingletonGreedy [20] (detailed below). Algorithms based on LatticeGreedySubroutine are prohibitively expensive for large problem instances, so we also consider FastGreedy [25], which is a relaxation of LatticeGreedySubroutine, based on a threshold greedy algorithm [34]. In addition, we evaluate and further analyze an approach which considers the lattice as a multiset and runs the greedy algorithm for set functions over it, which we call UnitGreedy (Algorithm 3). In this section, we describe each algorithm we evaluate and their associated approximation guarantees, some of which we derive.

Greedy algorithm descriptions

LatticeGreedySubroutine description We first describe LatticeGreedySubroutine, which is the core component of ℓ-EnumGreedy and SingletonGreedy. The foundation of LatticeGreedySubroutine is to repeatedly apply a "locally optimal" approach, where the subpopulation and number of vaccine shipments is applied that would improve the objective function the most.

As shown in the Algorithm 1 pseudocode, LatticeGreedySubroutine selects a (k^*,s^*) pair that maximizes the marginal gain of $g(\cdot)$ in each iteration, where $k^* \in [K]$ is a subpopulation and $s^* \in \mathbb{Z}_+$ is the number of bundles to allocate to subpopulation k^*. To compute the highest marginal gain among all possible $(k,s) \in [K] \times \mathbb{Z}_+$ pairs in each iteration of the algorithm, we assume that the algorithm has access to a "value oracle" that returns the value of the objective function $g(\cdot)$ at any point in its domain. It is possible that the selected pair (k^*,s^*) is not feasible because adding it to the solution causes the budget constraint to be violated. Such an iteration is said to have *failed*, and we remove the (k^*,s^*) pair from the search space Q, which is a list that LatticeGreedySubroutine maintains of all feasible allocations. Otherwise, the iteration is *successful* and the (k^*,s^*) pair is used to update the allocation. It is useful for our analysis to state the algorithm in this manner, allowing for failed iterations. However, to obtain an efficient implementation we can, in Line 4, prune the search space Q so as to guarantee that the condition in Line 5 is always satisfied. Such an implementation runs in $O(K \cdot D^2 \cdot T_g)$ time in the worst case, where T_g is the worst case running time of the value oracle. However, the at most $K \cdot D$ pairs in Q can all be evaluated in parallel, and assuming full parallelism with no overhead, the running time of LatticeGreedySubroutine can also be reduced to $O(D \cdot T_g \nu \log(K \cdot D))$ in the PRAM model (even with exclusive read and exclusive write). We note that LatticeGreedySubroutine and the algorithms based on it come from [20].

ℓ-EnumGreedy description We further allow LatticeGreedySubroutine to start with an arbitrary initial allocation $\hat{\boldsymbol{x}}_{\mathbf{0}}$, and not just $\mathbf{0}$ (see Line 1). This is so that we can use LatticeGreedySubroutine as the completion step for an algorithm that enumerates solutions of bounded size. Specifically, let $\ell \geq 1$ be a fixed integer and let $\mathcal{S}$ be the set of all feasible solutions of size ℓ or less. Thus each element in $\mathcal{S}$ is a subset of at most ℓ subpopulations, each allocated some number of vaccine bundles so that the overall allocation is of size at most D. Note that $|\mathcal{S}| = O(K^\ell \cdot D^\ell)$. We then iterate over all elements of $\mathcal{S}$ and call LatticeGreedySubroutine with $\hat{\boldsymbol{x}}_{\mathbf{0}}$ set to each element in $\mathcal{S}$. We call this entire algorithm ℓ-EnumGreedy. Later in this section, we analyze 3-EnumGreedy.

SingletonGreedy description While 3-EnumGreedy runs in polynomial time (specifically, $O(K^4 \cdot D^5 \cdot T_g)$ time), it is expensive and not practical for large instances. A cheaper algorithm based on LatticeGreedySubroutine computes one solution by starting LatticeGreedySubroutine with $\mathbf{0}$ as the initial allocation and then computes K additional

"singleton" solutions by allocating the entire budget to each of the K subpopulations. The final solution returned is the best of these $K + 1$ solutions. We call this the SingletonGreedy algorithm. Note that the running time of SingletonGreedy is dominated by LatticeGreedySubroutine.

Algorithm 1 LatticeGreedySubroutine $(\mathcal{M}, \mathbf{I}, \hat{\boldsymbol{x}}_0)$

$\hat{\boldsymbol{x}} \leftarrow \hat{\boldsymbol{x}}_0$
$Q := \{(k,s) : k \in [K], 1 \le s \le \lceil \frac{n_k}{\Delta} \rceil - \hat{\boldsymbol{x}}_k\}$
while $\|\hat{\boldsymbol{x}}\|_1 < D$ and $Q \neq \varnothing$ **do**
  $k^*, s^* \leftarrow \underset{(k,s)\in Q}{\operatorname{argmax}} \frac{g(\hat{\boldsymbol{x}}+s\cdot\boldsymbol{e}_k|\mathcal{M},\mathbf{I}) - g(\hat{\boldsymbol{x}}|\mathcal{M},\mathbf{I})}{s}$
  if $\|\hat{\boldsymbol{x}} + s^* \cdot \boldsymbol{e}_{k^*}\|_1 \le D$ **then**
    $\hat{\boldsymbol{x}} \leftarrow \hat{\boldsymbol{x}} + s^* \cdot \boldsymbol{e}_{k^*}$
    $Q \leftarrow Q \setminus \{(k,s) : s + \hat{\boldsymbol{x}}_k > \lceil \frac{n_k}{\Delta} \rceil\}$
  else
    Remove (k^*, s^*) from Q
  end if
end while
return $\hat{\boldsymbol{x}}$

FastGreedy description FastGreedy [25] is a more flexible version of LatticeGreedySubroutine that maintains a threshold value to determine how strict the algorithm is in choosing allocations. FastGreedy starts with a high threshold (τ_f) value that determines the minimum benefit required from a (k,s) pair to select it, and τ_f is relaxed as FastGreedy progresses. In each iteration, any (k,s) pair that provides benefit above the current threshold gets selected, allowing multiple selections per iteration (unlike LatticeGreedySubroutine, which picks only one). At the end of each iteration, the threshold is lowered according to the rate parameters κ_f and δ_f. More specifically, κ_f controls how much τ_f decreases in each iteration, where a higher value of κ_f will result in higher standards for selection in each iteration. δ_f controls how quickly β_f approaches β_f^*, which is an upper-bound on the DR-submodularity ratio (see Eq 7 and [25]). The parameter ε_f determines the minimum progress FastGreedy must make in order to terminate early (i.e., before the budget is met).

FastGreedy differs from LatticeGreedySubroutine in two ways that make it more efficient: i) FastGreedy allows allocation to multiple subpopulations in a single iteration instead of only one per iteration, and ii) FastGreedy determines the best number of bundles through a binary search subroutine (included in Appendix B) instead of searching through every (k,s) pair (which LatticeGreedySubroutine does).

UnitGreedy description. On the problem instances we consider, in practice, 3-EnumGreedy, and SingletonGreedy elect to allocate one bundle at a time for a majority of iterations. With this in mind, we consider another more efficient algorithm, UnitGreedy. As shown in the Algorithm 3 pseudocode, UnitGreedy allocates one vaccine bundle to a subpopulation $k \in [K]$, each time selecting a subpopulation that yields the highest marginal gain in the objective function - this is equivalent to converting the lattice into a multiset and running a set greedy algorithm on it (such as the one in [26]). The algorithm continues until the vaccine budget D is met.

The running time of this algorithm is $O(K \cdot D \cdot T_g)$. Note that the marginal gains for the various bundles can be computed (Line 3 in Algorithm 3) in parallel in a straightforward manner, and if we ignore overhead for parallelization, the running time reduces to $O(D \cdot T_g)$.

Algorithm 2 FastGreedy ($\mathcal{M}, \mathbf{I}, \kappa_f, \delta_f, \varepsilon_f \in (0,1)$)

$\boldsymbol{x} \leftarrow \mathbf{0}, M \leftarrow \max_{k\in[K]} g(\boldsymbol{e_k}), m \leftarrow M, m' \leftarrow M/\kappa_f, \beta_f \leftarrow 1$
while $m \geq M\varepsilon_f/D$ **do**
  $m \leftarrow \max_{k\in[K]} g(\boldsymbol{x}+\boldsymbol{e_k}) - g(\boldsymbol{x})$
  if $m > \kappa_f m'$ **then**
    $\beta_f \leftarrow \beta_f\delta_f$
  end if
  $m' \leftarrow m$
  $\tau_f \leftarrow \beta_f\kappa_f m$
  for $k \in [K]$ **do**
    $\ell \leftarrow$ `BinarySearchPivot`$(\mathcal{M}, g, \boldsymbol{x}, k, D, \tau_f)$
    $\boldsymbol{x} \leftarrow \boldsymbol{x} + \ell e_k$
    if $\|\boldsymbol{x}\|_1 = D$ **then return** $\boldsymbol{x}$
    end if
  end for
end whilereturn $\boldsymbol{x}$

Algorithm 3 UnitGreedy ($\mathcal{M}, \mathbf{I}$)

$\hat{\boldsymbol{x}} \leftarrow \mathbf{0}$
while $\|\hat{\boldsymbol{x}}\|_1 < D$ **do**
  $k^* \leftarrow \operatorname{argmax}_{k\in[K]} g(\hat{\boldsymbol{x}} + \boldsymbol{e_k}|\mathcal{M}, \mathbf{I}) - g(\hat{\boldsymbol{x}}|\mathcal{M}, \mathbf{I})$
  $\hat{\boldsymbol{x}} \leftarrow \hat{\boldsymbol{x}} + \boldsymbol{e_{k^*}}$
end while
return $\hat{\boldsymbol{x}}$

Approximation guarantees

The performance of our algorithms depends on how close to submodularity their objective functions are. In this section, we i) formally define the notion of "distance to submodularity" and ii) connect these definitions to the quality of output of our algorithms.

Lattice function submodularity ratios To analyze the greedy algorithms described above, we utilize the notion of *submodularity ratio* defined in [24]. The submodularity ratio of a function g is a quantity between 0 and 1 that is a measure of g's "distance" to submodularity. Since there are two distinct notions of submodularity for lattice functions, as defined in the Background section, there are two associated notions of submodularity ratios, which we now present. To simplify notation, we drop $\mathcal{M}$ and $\mathbf{I}$ and simply use $g(\boldsymbol{x})$ for our objective function.

Definition 7. DR-Submodularity Ratio. *[24] The DR-submodularity ratio of a function* $g : \mathbb{Z}_+^K \to \mathbb{R}$ *is defined as*

$$\beta(g) = \min_{\boldsymbol{y}\leq\boldsymbol{x}, k\in[K]} \frac{g(\boldsymbol{y}+\boldsymbol{e_k}) - g(\boldsymbol{y})}{g(\boldsymbol{x}+\boldsymbol{e_k}) - g(\boldsymbol{x})} \tag{9}$$

In this definition (and in the next definition below) we designate $\frac{0}{0}$ to be 1 and $\frac{n}{0}$ to be ∞ for any positive integer n. From this definition it is clear that $\beta(g) \leq 1$ because $\boldsymbol{x} = \boldsymbol{y}$ is included in the space that is being minimized over. Furthermore, this definition along with the definition of DR-submodularity (Definition 1) implies that $\beta(g) = 1$ iff g is DR-submodular. Thus, the "distance" $1 - \beta(g)$ indicates how far the function g is from being DR-submodular. Below we present a similar definition that captures the notion of "distance" of a function g from being submodular.

Definition 8. Submodularity Ratio. *[24] The submodularity ratio of a function* $g : \mathbb{Z}_+^K \to \mathbb{R}$ *is defined as*

$$\alpha(g) = \min_{\boldsymbol{y} \leq \boldsymbol{x}, k \in [K]: \boldsymbol{x}_k = \boldsymbol{y}_k} \frac{g(\boldsymbol{y} + \boldsymbol{e_k}) - g(\boldsymbol{y})}{g(\boldsymbol{x} + \boldsymbol{e_k}) - g(\boldsymbol{x})} \tag{10}$$

Like $\beta(g)$, the submodularity ratio $\alpha(g)$ also satisfies $\alpha(g) \leq 1$ and $1 - \alpha(g)$ indicates how far the function g is from being submodular. Since submodularity is a weaker notion than DR-submodularity, an arbitrary lattice function will be "closer" to submodularity than DR-submodularity. Correspondingly, $\alpha(g) \geq \beta(g)$.

We now present approximation guarantees for 3-EnumGreedy (Theorem 9a), SingletonGreedy (Theorem 9b), and UnitGreedy (Theorem 10). The approximation guarantee associated with FastGreedy can be found in [25].

Guarantees for 3-EnumGreedy and SingletonGreedy Theorem 9 provides a guarantee for 3-EnumGreedy and SingletonGreedy. Previously, [20] established approximation guarantees for these algorithms over submodular objective functions, whereas we establish them for more general objective functions.

Theorem 9. *Let* $g : \mathbb{Z}_+^K \to \mathbb{R}$ *be an arbitrary monotone function. Let OPT denote the optimal solution to the problem* $\max_{\|\boldsymbol{x}\|_1 \leq D} g(\boldsymbol{x})$.

(a) If $\hat{\boldsymbol{x}}$ *is the solution returned by* 3-EnumGreedy *then*

$$g(\hat{\boldsymbol{x}}) \geq (1 - e^{-\alpha(g)}) \cdot OPT$$

(b) If $\hat{\boldsymbol{x}}$ *is the solution returned by* SingletonGreedy *then*

$$g(\hat{\boldsymbol{x}}) \geq \frac{\alpha(g)}{2} \cdot (1 - e^{-\alpha(g)}) \cdot OPT$$

The proof is included in Section B.1 in S1 Text.

Guarantee for UnitGreedy Here, we provide a version of the approximation guarantee found in [26], which is dependent on the submodularity ratio for set functions [23] and generalized curvature [26]. Their guarantee is applicable to UnitGreedy when we consider the lattice over which we allocate to be a multiset.

Theorem 10. *Let* $g : \mathbb{Z}_+^K \to \mathbb{R}$ *be an arbitrary monotone function. Let OPT denote the optimal solution to the problem* $\max_{\|\boldsymbol{x}\|_1 \leq D} g(\boldsymbol{x})$. *If* $\hat{\boldsymbol{x}}$ *is the solution returned by* UnitGreedy *then*

$$g(\hat{\boldsymbol{x}}) \geq (1 - e^{-\beta(g)}) \cdot OPT$$

The above results show that the approximation guarantees shown in the literature [20,28] for greedy algorithms when g is a submodular function (on sets or lattices) are more general and apply to arbitrary monotone integer lattice functions.

Note that the theorems above also provide a trade-off between approximation-factor and running time. UnitGreedy is the fastest algorithm, but this provides an $(1 - e^{-\beta(g)})$-approximation, which is no better than the $(1 - e^{-\alpha(g)})$-approximation provided by the more expensive algorithm 3-EnumGreedy.

We remark that ℓ-EnumGreedy, SingletonGreedy, FastGreedy, and UnitGreedy are well known algorithms for maximizing a submodular function over sets or lattices subject to a cardinality constraint (e.g., [19,20,25,26]). Our main contribution here is to show that 3-EnumGreedy and SingletonGreedy provide approximation guarantees even when the

Table 2. Summary of greedy algorithms presented in this section. Details of approximation factors for FastGreedy and UnitGreedy may be found in [26] and [25], respectively.

Algorithm	Query Complexity	Approximation Factor	Prior Work
3-EnumGreedy	$O(K^4 \cdot D^5)$	$1 - e^{-\alpha(g)}$ [this paper]	Analysis for lattice submodular functions [19,20]
SingletonGreedy	$O(K^2 \cdot D^2)$	$\frac{\alpha(g)}{2}(1 - e^{-\alpha(g)})$ [this paper]	
FastGreedy	$O((\log_{\delta_f}(\gamma_d) \cdot \log_{\kappa_f}(\gamma_d) + \log_{\kappa_f} \varepsilon^2/D) \cdot K \log D)$	$1 - e^{-\kappa_f \beta^* \gamma_s} - \varepsilon$ [prior work]	Analysis for lattice non-submodular functions [25]
UnitGreedy	$O(K \cdot D)$	$\frac{1}{\alpha_g}(1 - e^{-\alpha_g \gamma_{dk}})$ [prior work] $1 - e^{-\beta(g)}$ [this paper]	Analysis for multiset non-submodular functions [26]

https://doi.org/10.1371/journal.pcbi.1012539.t002

objective function is not submodular and these guarantees degrade gracefully as the objective function becomes less submodular, as measured by the submodularity ratio. We also derive a lattice function based approximation guarantee for UnitGreedy, extending from the set function guarantee provided in [26].

Finally, we note that the POMS algorithm in [24] achieves a $\max((1 - e^{-\beta(g)}), \alpha(g)/2 \cdot (1 - e^{-\alpha(g)}))$ -approximation. Our results show that simple, well-known, and faster greedy algorithms achieve these same approximation factors.

Results

Next, we present a variety of experiments that collectively show that **i)** greedy methods outperform various baseline vaccine allocation algorithms for both MaxCasesAverted and MaxPeaksReduced objectives, **ii)** greedy methods are very close to optimal for all instances for which this comparison was feasible, and **iii)** the greedy methods are considerably faster than POMS [24] (when requiring all algorithms to run until their approximation factors can be guaranteed).

We run our experiments at 3 different scales: (i) small-scale experiments: New Hampshire (10 counties, population 1.4 million), (ii) medium-scale experiments: Iowa (99 counties, population 3.2 million), and (iii) large-scale experiments: Texas (254 counties, population 30.03 million).

Our code and processed data are made available. Experiments were run on AMD EPYC 7763 CPUs with 2 TB RAM.

Baselines

Our baselines include natural vaccine allocation strategies such as Population, Out-Mobility, In-Mobility, and Random, which assign vaccines to each county proportional to the population, the total mobility originating in the county, the total mobility terminating in the county, and uniformly at random respectively. We also compare our approaches against POMS [24], which works by expanding a random pareto-optimal frontier.

Data

Our experimental test-beds consist of simulated outbreaks over inter-county mobility graphs for New Hampshire, Iowa and Texas constructed from two separate sources:

FRED (Framework for Reconstructing Epidemic Dynamics) [30] (open source) includes a census-based synthetic population with high-resolution social, familial, demographic, and behavioral details. We infer a daily-commute mobility network from home and work locations.

- **Movement**: Daily inter-county commute statistics.
- **Coverage**: Uniform coverage ensuring that home and work locations for every county (i.e., subpopulation) are modeled to an equal degree.
- **Strengths**: Captures essential work-based mobility and is consistent across counties.
- **Limitations**: Lacks recreational, shopping, and irregular mobility patterns.

SafeGraph [31] (open source for academics) provides aggregated and anonymized mobility data from mobile device GPS signals, which provides inferred 'home' locations and visits to places of interest (POIs).

- **Movement**: All types of travel including work, leisure, shopping, social visits, and other activities.
- **Coverage**: Broad coverage for urban areas with comprehensive mobile and internet infrastructure.
- **Strengths**: Captures a comprehensive view of mobility including irregular patterns.
- **Limitations**: Potentially unreliable coverage for rural subpopulations due to lower mobile phone infrastructure.

Mobility graphs. We derive state-level directed mobility graphs from both data sources, where nodes correspond to counties and directed weighted edges correspond to movement from the source county to the target county, with edge weights representing the number of commutes between county pairs. The mobility graphs constructed using FRED and SafeGraph are similar for New Hampshire and Iowa (the SafeGraph mobility graphs have a slightly higher density). The New Hampshire graphs are nearly complete due to the state's small size (it can be crossed by vehicle in about 45 minutes). In contrast, the mobility graphs for Texas reveal significant differences between FRED and SafeGraph data sources. The density of the FRED mobility graph is an order of magnitude lower than that of SafeGraph. This difference occurs because SafeGraph captures more irregular travel patterns, including instances where individuals travel long distances across Texas. Such cross-state travel is relatively rare compared to New Hampshire, where short distances make travel between any two points feasible, but it takes nearly 10 hours to cross Texas by vehicle. A more comprehensive description of mobility graph construction and a table of their properties appears in Section C.2 and Table A in S1 Text.

Parameters

We select values of λ (infectivity) at approximately 0.347 and 0.535 to result in 20% and 70% of each population becoming infected without vaccination, respectively. We conducted experiments with a wider range of λ values (in general, we observed that problem instances with lower values of λ are more easily solved by more vaccine allocation methods) and chose two values that represent significantly different levels of infectivity. We performed experiments for New Hampshire, Iowa, and Texas, with a vaccine budget of 10% through 60% of each state's total population in 10% increments. Each vaccine budget refers to the total percent of the population for which vaccines are available, and we assume that the entire vaccine

budget will be used up for each budget amount. The parameters k, n_i, and w_{ij} are instantiated according to the data when we constructed the mobility graphs. The parameters r_i scale the infectivity parameter λ for each county, and is set in proportion to the population density of each county. We set the initially infected vector $\mathbf{I}$ to be 1 for each county. The choice in $\mathbf{I}$ does not make a difference in our setting due to the deterministic nature of our model and the small diameter of our mobility graphs (at most 4). η and δ are set according to [35]. For FastGreedy in New Hampshire and Iowa, we set $\kappa_f = \delta_f = 0.96$, and in Texas, we set $\kappa_f = \delta_f = 0.93$. For all FastGreedy experiments, we set $\varepsilon_f = 0$. We run each simulation for at least 200 timesteps and terminate the simulation when the disease dies out.

Performance of greedy methods compared to baselines

In our first experiment, we compare the performance of greedy vaccine allocation algorithms to baseline algorithms using both the FRED and SafeGraph mobility graphs, for both the MaxCasesAverted and MaxPeaksReduced problems. For our small-scale experiment (New Hampshire), we run all four greedy algorithms. For our medium-scale experiment (Iowa), we drop our slowest greedy algorithm 3-EnumGreedy because the initial enumeration step required by 3-EnumGreedy too prohibitively expensive the number of subpopulations grows. For our large-scale experiment (Texas), we drop our slowest two greedy algorithms (3-EnumGreedy and SingletonGreedy) because exploring every possible number of shipments to each subpopulation in a reasonable amount of time is infeasible for this scale. For this comparison, we always run POMS for the same amount of time as UnitGreedy. We seek to demonstrate how close the performance of POMS gets to that of UnitGreedy in a simple wall clock time based comparison. We repeat these experiments for six different budgets (expressed as a percentage of the population of the state) for two different values of λ. The results for a high infectivity value of λ are summarized in Figs 2, 3, and 4. The same experiments for lower infectivity parameter values can be found in Section C.3 in S1 Text.

Fig 2 shows that, for our small-scale experiments, the baselines never outperform the greedy methods. Population and POMS perform on-par with the greedy methods in some instances, particularly in MaxPeaksReduced. We see the performance of baselines relative to the greedy methods decline as the scale of our experiments become larger.

As observed in Fig 3, even for our medium-scale experiments, the greedy algorithms outperform each baseline in several settings, while no baseline outperforms the greedy methods. Fig 4 demonstrates that, for our large-scale experiment, UnitGreedy and FastGreedy outperform the baselines by a wider margin than our small and medium-scale experiments over the FRED dataset. This margin is more narrow (with UnitGreedy and FastGreedy still in the lead) over the SafeGraph mobility graph. For SafeGraph, UnitGreedy and FastGreedy perform on-par with the same methods over the FRED data - the difference is primarily in the increased performance of the baselines over SafeGraph. We note that for our large-scale experiment (Texas), the mobility graph constructed using SafeGraph data is far more dense than the one constructed using FRED data, and as a result, disease flows much more freely over the subpopulations. This highlights how mobility graph structure can impact the effectiveness of simple vaccine allocation methods. Similar results hold for a lower value of λ, which can be found in Figs B, C, and D in S1 Text.

UnitGreedy performs at least on-par with the other greedy methods, all of which employ larger search spaces. In all experiments, after the greedy methods, the Population heuristic performs well, followed by POMS, other baselines, and finally Random. The relatively poor performance of POMS could be attributed to the fact that it requires a long running

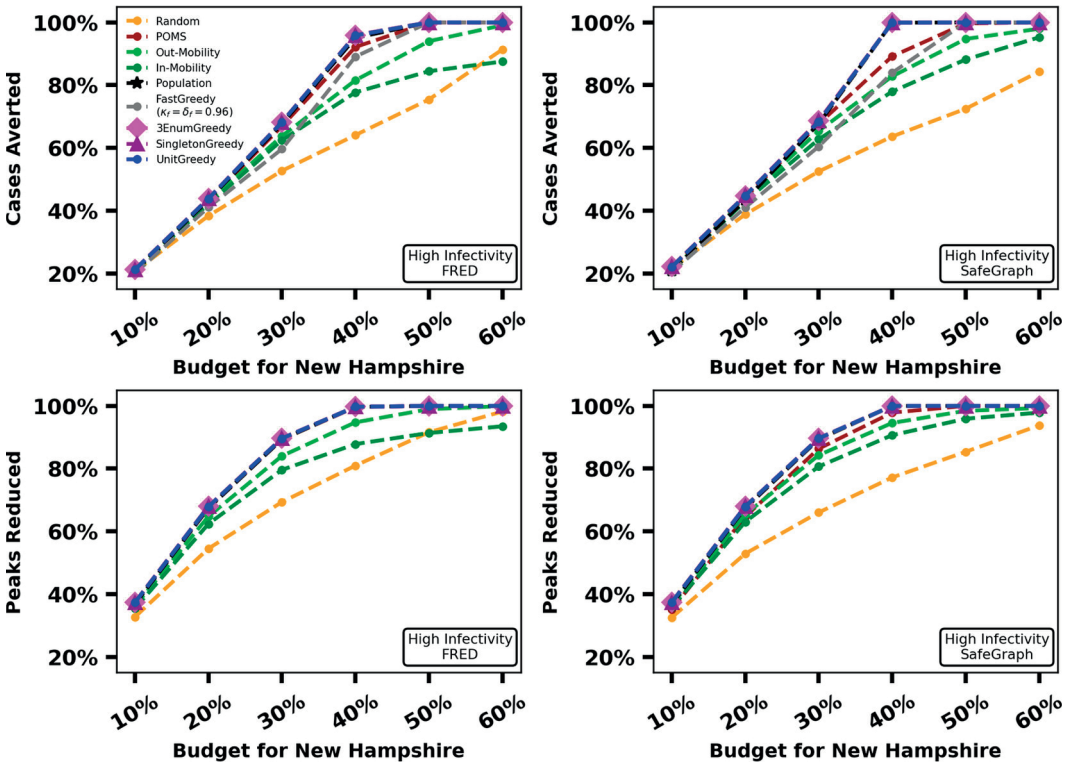

Fig 2. Percentage totBurden and percentage maxBurden reduced by all approaches for $\lambda = 0.5345$ in New Hampshire for FRED (first column) and SafeGraph (second column).

https://doi.org/10.1371/journal.pcbi.1012539.g002

time to achieve its theoretical guarantee (see Performance-Time Trade-off). The surprisingly good performance of the Population heuristic suggests that it might be a good on-the-field strategy in the absence of mobility data for small problem instances. UnitGreedy substantially outperforms Population and FastGreedy for our large-scale experiment on Texas over FRED data, with totBurden reduced by up to 8% of the population, which translates to almost 2 million additional cases avoided.

Near-optimality of greedy algorithms

In this section, we demonstrate that in practice, the greedy algorithms we evaluate return allocations whose objective function value is close to optimal for both MaxCasesAverted and MaxPeaksReduced. First, we directly compute optimal solutions for our small-scale experiment (New Hampshire) using mobility derived from FRED data, which would be prohibitively expensive for our medium-scale and large-scale settings. We consider 4 problem instances for each of MaxCasesAverted and MaxPeaksReduced, obtained by setting λ to 0.347 and 0.5345 and the budget D to 2 values (10% and 40% of the population). For these problem instances we compute an optimal solution by exhaustive search and compare the results to that of 3-EnumGreedy, SingletonGreedy, UnitGreedy, and FastGreedy.

For each problem and problem instance, let OPT denote the objective function value of an optimal solution. Table 3 shows the performance relative to OPT of problem instances for 10% and 40% budgets, high and low infectivity, and both objective functions for each greedy method.

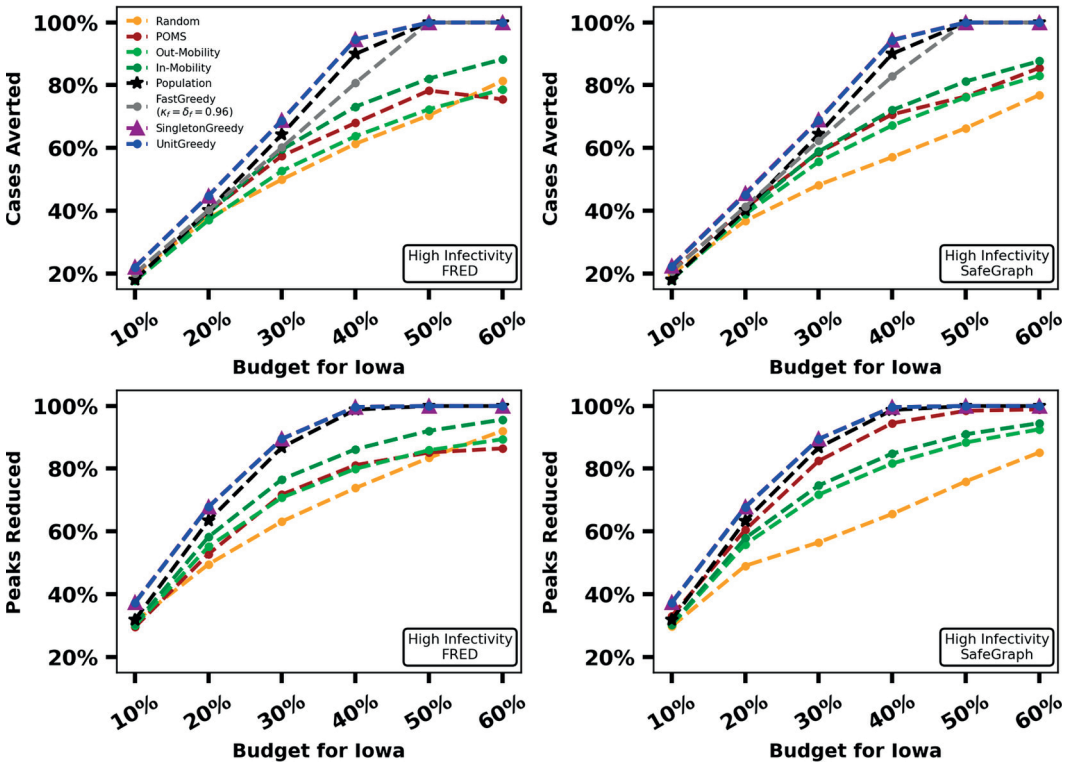

Fig 3. Percentage totBurden and percentage maxBurden reduced by UnitGreedy, SingletonGreedy, FastGreedy and baselines for $\lambda = 0.535$ in Iowa for FRED (first column) and SafeGraph (second column).

https://doi.org/10.1371/journal.pcbi.1012539.g003

Problem instances for the state of Iowa are much larger and it is not feasible to compute OPT to make a direct comparison. To circumvent this problem, we first note that it is possible to obtain improved versions of Theorems 9(a), 9(b), and 10 by defining "per instance" versions of the DR-submodularity ratio and submodularity ratio. To be specific, let $\hat{\boldsymbol{x}}_i$ denote the allocation after iteration i of UnitGreedy, let $\boldsymbol{x}^*$ be an optimal solution, and let $\boldsymbol{y}^* = \boldsymbol{0} \vee (\boldsymbol{x}^* - \hat{\boldsymbol{x}}_i)$. Define

$$\beta(g, \hat{\boldsymbol{x}}_i) := \frac{\sum_{j=1}^{K} g(\boldsymbol{y}_j^* \boldsymbol{e}_j + \hat{\boldsymbol{x}}_i) - g(\hat{\boldsymbol{x}}_i)}{g(\boldsymbol{y}^* + \hat{\boldsymbol{x}}_i) - g(\hat{\boldsymbol{x}}_i)} \tag{11}$$

The numerator is the total marginal gain of independently increasing each individual subpopulation's allocation to the optimal allocation. The denominator is the marginal gain of increasing $\hat{\boldsymbol{x}}_i$ to the optimal solution all at once. If g were submodular, it would follow that $\beta(g, \hat{\boldsymbol{x}}_i) \geq 1$, but this guarantee does not hold for an arbitrary $g(\cdot)$. It is possible to show that the bound stated in Theorem 10 holds for $\beta(g, \hat{\boldsymbol{x}})$, i.e., $g(\hat{\boldsymbol{x}}) \geq (1 - e^{-\beta(g,\hat{\boldsymbol{x}})}) \cdot OPT$ (more on this may be found in Section C.4 in S1 Text).

Since we cannot calculate the optimal solution $\boldsymbol{x}^*$ directly (and $\beta(g, \hat{\boldsymbol{x}}_i)$ depends on $\boldsymbol{x}^*$) we cannot calculate $\beta(g, \hat{\boldsymbol{x}}_i)$ directly either. Instead, we use a sampling method (described in Section C.4 in S1 Text) to find an estimate of $\beta(g, \hat{\boldsymbol{x}}_i)$, which we denote as $\hat{\beta}(g, \hat{\boldsymbol{x}}_i)$. We calculate $\hat{\beta}(g, \hat{\boldsymbol{x}}_i)$ 5000 times for each experiment to estimate $\beta(g, \hat{\boldsymbol{x}}_i)$. Our key finding is that $\hat{\beta}(g, \hat{\boldsymbol{x}})$ is very close to (or even larger than) 1 for most of our experimental instances,

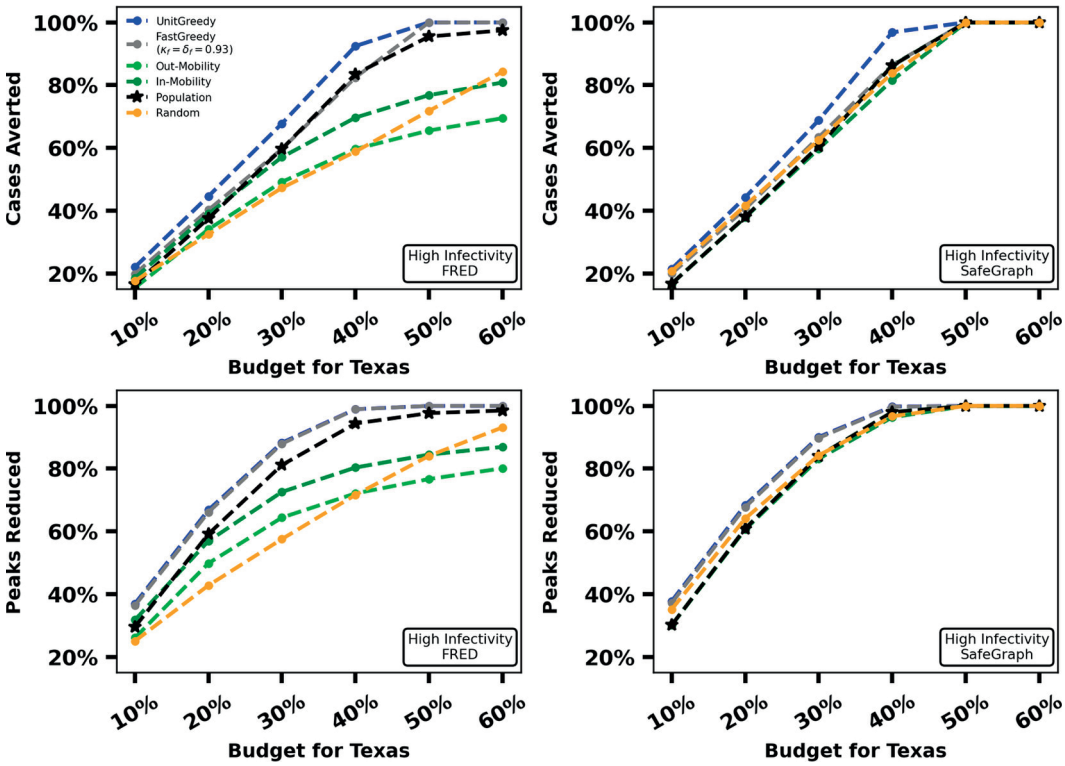

Fig 4. Percentage totBurden and percentage maxBurden reduced by UnitGreedy, FastGreedy and baselines for $\lambda = 0.525$ in Texas for FRED (first column) and SafeGraph (second column).

https://doi.org/10.1371/journal.pcbi.1012539.g004

Table 3. Approximation factors for each problem instance.

NH (FRED) 10% Budget	Cases Averted		Peaks Reduced	
	$\lambda = 0.3475$	$\lambda = 0.5355$	$\lambda = 0.3475$	$\lambda = 0.5355$
3-EnumGreedy	99.84%	98.53%	99.71%	96.33%
SingletonGreedy	79.02%	99.04%	99.71%	99.33%
FastGreedy	79.02%	95.29%	92.29%	95.21%
UnitGreedy	79.02%	99.04%	99.71%	96.33%
NH (FRED) 40% Budget	Cases Averted		Peaks Reduced	
	$\lambda = 0.3475$	$\lambda = 0.5355$	$\lambda = 0.3475$	$\lambda = 0.5355$
3-EnumGreedy	100%	99.86%	100%	99.97%
SingletonGreedy	100%	99.86%	100%	99.97%
FastGreedy	100%	99.26%	100%	99.97%
UnitGreedy	100%	99.86%	100%	99.97%

https://doi.org/10.1371/journal.pcbi.1012539.t003

implying that g might be close to being submodular in practice. This suggests that the allocation $g(\hat{\boldsymbol{x}}) \geq (1 - 1/e) \cdot OPT \approx 0.63 \cdot OPT$.

Estimates for $\beta(g, \hat{\boldsymbol{x}})$ can be found in Table 4. These values indicate worst-case approximation factors for performance on-par (and some exceeding) that of submodular functions for our problem formulations and experimental settings.

Table 4. Estimates of $\beta(g, \hat{x})$ for each problem instance.

NH (FRED) 60% Budget	Cases Averted		Peaks Reduced	
	$\lambda = 0.3475$	$\lambda = 0.5355$	$\lambda = 0.3475$	$\lambda = 0.5355$
3-EnumGreedy	1.03	1.01	1.74	1.02
SingletonGreedy	1.41	1.01	1.53	1.02
FastGreedy	1.04	1.01	1.05	1.01
UnitGreedy	1.51	1.02	2.66	1.12
IA (FRED) 60% Budget	Cases Averted		Peaks Reduced	
	$\lambda = 0.3475$	$\lambda = 0.5355$	$\lambda = 0.3475$	$\lambda = 0.5355$
SingletonGreedy	1.02	1.01	1.06	1.01
FastGreedy	1.04	1.01	1.05	1.01
UnitGreedy	1.02	1.02	1.07	1.04
TX (FRED) 60% Budget	Cases Averted		Peaks Reduced	
	$\lambda = 0.3475$	$\lambda = 0.5355$	$\lambda = 0.3475$	$\lambda = 0.5355$
FastGreedy	1.12	1.03	1.04	1.03
UnitGreedy	1.02	1.02	1.06	1.03

https://doi.org/10.1371/journal.pcbi.1012539.t004

Performance and running-time trade-offs

Here, we compare the performance and running time trade-offs for 3-EnumGreedy, SingletonGreedy, FastGreedy, UnitGreedy and POMS. Let $c_{\max} = \max\{n_i | i \in [K]\}$. The approximation guarantee for POMS requires $T = 2ec_{\max}D^2K$ queries [24]; this makes POMS significantly more expensive to run compared to the greedy methods. The term "query" refers to an evaluation of the objective function $g(\cdot)$; here, that evaluation entails running a disease simulation conditioned on a vaccine allocation. Compared to POMS, UnitGreedy requires relatively fewer $T = K \cdot D$ queries. In addition, UnitGreedy is much faster in practice (by Wall Clock Time) than POMS since UnitGreedy is embarrassingly parallel, whereas POMS is much more inherently sequential. These comparisons are presented in Table 5, where we list required iterations and practical run time (extrapolated from 12 hours for POMS). FastGreedy introduces an approximation guarantee parameterized by a value which upper bounds the DR-submodularity ratio. Their input parameters can be adjusted to determine the quality required of potential allocation in each iteration, effectively trading

Table 5. FastGreedy, UnitGreedy, SingletonGreedy, 3-EnumGreedy, and POMS comparison with respect to practical running time (estimated for POMS) to achieve approximation guarantee for New Hampshire with 20% and 60% budgets.

NH (FRED) 20% Budget	Queries Required		Wall Clock Time	
	$\lambda = 0.3475$	$\lambda = 0.5355$	$\lambda = 0.3475$	$\lambda = 0.5355$
FastGreedy	1567	70	3.9 Minutes	7.4 Seconds
UnitGreedy	$6.67 \cdot 10^3$	$6.67 \cdot 10^3$	16.7 Minutes	7.6 Minutes
SingletonGreedy	$6.43 \cdot 10^5$	$7.57 \cdot 10^5$	1 Hour	37.1 Minutes
3-EnumGreedy	$6.53 \cdot 10^5$	$7.58 \cdot 10^5$	1 Hour	37.6 Minutes
POMS	$3.63 \cdot 10^{15}$	$3.63 \cdot 10^{15}$	$\sim 1.5 \cdot 10^8$ Years	$\sim 1.2 \cdot 10^8$ Years
NH (FRED) 60% Budget	Queries Required		Wall Clock Time	
	$\lambda = 0.3475$	$\lambda = 0.5355$	$\lambda = 0.3475$	$\lambda = 0.5355$
FastGreedy	4761	2025	7.7 Minutes	3.3 Minutes
UnitGreedy	$2 \cdot 10^4$	$2 \cdot 10^4$	33.2 Minutes	30 Minutes
SingletonGreedy	$3.54 \cdot 10^6$	$2.7 \cdot 10^6$	3.9 Hours	3.4 Hours
3-EnumGreedy	$3.55 \cdot 10^6$	$2.66 \cdot 10^6$	5.1 Hours	1.5 Hours
POMS	$3.27 \cdot 10^{16}$	$3.27 \cdot 10^{16}$	$\sim 1.1 \cdot 10^9$ Years	$\sim 1.1 \cdot 10^9$ Years

https://doi.org/10.1371/journal.pcbi.1012539.t005

performance for speed. When the input parameters to FastGreedy are set so that the performance is maximized, the resulting approximation guarantee is similar to that of UnitGreedy, 3-EnumGreedy, and SingletonGreedy.

Discussion

Through a combination of theoretical and experimental results, we have shown that even though metapopulation model vaccine allocation problems are inapproximable in the worst case, simple greedy algorithms can be both effective and scalable for these problems. We provide a possible theoretical explanation for the effectiveness of these greedy algorithms by establishing worst case approximation guarantees in terms of the submodularity ratios of the objective functions of these problems. Specifically, we extend worst case approximation guarantees from the literature for lattice greedy algorithms [20,25,26] to the non-submodular objective function setting. Our analysis builds upon prior work on submodular set and lattice function maximization [5,10,19,20,28,34].

For specific instantiations of the metapopulation model vaccine allocation problems (e.g., MaxCasesAverted, MaxPeaksReduced) we provide some empirical evidence that the submodularity ratio of the objective functions is high enough (i.e., close enough to 1) to imply that greedy algorithms yield near-optimal solutions to these problems. The greedy algorithms we evaluate are effective across small (New Hampshire), medium (Iowa), and large (Texas) problem scales over two mobility graphs constructed from FRED [30] and SafeGraph [36] data sources. In all problem instances of MVA we evaluate, the greedy methods outperform the baselines, sometimes by quite a significant margin. This difference in performance is typically greatest for a high λ (infectivity) value, vaccinating 30% to 50% of the total state's population for each problem scale. We also demonstrate that the greedy algorithms achieve an approximation factor of over 0.79 for a 10% budget, and an approximation factor of over 0.99 with a 40% budget for both MaxCasesAverted and MaxPeaksReduced problem instances over New Hampshire.

We observe the performance of the greedy methods are on-par with each other for the Texas FRED and SafeGraph mobility graphs, but the performance of the baselines over the FRED mobility graph are much lower. Because of this, we conjecture that the MVA problem over sparse mobility graphs is harder to solve and we cannot depend on the baselines. Across all experiments, we observe that the MVA problem instances with a lower infectivity value λ - infecting approximately 20% of the population - are generally easier to achieve good performance on for all methods.

Moreover, we have parallelized our algorithms to enhance scalability. As a result, the fastest of our algorithms takes 2-3 hours to run for the state of Texas. The ability to parallelize the computation allows us to manage the computational demands of large states, ensuring that our methods remain feasible even in large-scale datasets. The query complexities for each greedy algorithm (shown in Table 2) further contributes to the feasibility and speed of the fastest two greedy algorithms we present, UnitGreedy and FastGreedy. In addition, it is quite natural to speed up greedy methods by looking not just for a locally optimal update in each iteration, but for an *approximately* optimal update, which is a main principle behind the threshold approach of FastGreedy. These features of the greedy methods present a computational advantage with respect to scalability over algorithms such as POMS, introduced in [24].

Despite these contributions, several limitations remain. Our current disease model is relatively simple and deterministic and it assumes homogeneous mixing within subpopulations. Our model can be extended in a variety of ways in order to better capture the complexities of

real-world disease spread. It would relatively easy to extend the SEIR model we use to allow for additional compartments (e.g., an infected but asymptomatic compartment). Another simple extension for future work could be to extend the model to one that captures certain demographics (e.g., age, gender, etc.) of the population, such as the one presented in [7]. Incorporating demographics of the population into the disease model would be the first step in designing a vaccine allocation method that prioritizes certain groups. A more granular approach would be to incorporate agent-based simulations within subpopulations, to better reflect heterogeneous contact patterns. The choice of a disease model in our setting is largely driven by the needs of the vaccine allocation methods. On the other hand, the granularity and sophistication of the disease model has a direct impact on the computational cost of our vaccine allocation methods. This trade-off, between sophistication of the disease models and the computational cost of the vaccine allocation methods should be carefully considered when choosing a disease model. Exploring faster, more scalable algorithms, such as sketch-based methods [25,37], could alleviate this trade-off to some extent and is a promising direction for future research. An additional limitation is that the inferred mobility data we use is based on limited sources and does not fully reflect real-world movement patterns, particularly in rural areas. Expanding to include more comprehensive mobility data, such as transportation networks, would improve accuracy. Our work focuses on preemptive vaccine allocation, i.e., vaccine allocation at the beginning of an outbreak. Expanding our work to consider vaccine allocation over time as the disease spreads and more vaccines become available is also a promising direction for future research. For this paper, we ran experiments on individual states in isolation without taking physical border effects into account, where in real-world settings, the influence of areas (especially urban) across a state border could have significant impact on vaccine allocation decisions. Finally, deriving confidence bounds for the estimated submodularity ratios would enhance the robustness of our theoretical guarantees.

Supporting information

Fig A in S1 Text. Iowa and New Hampshire mobility graphs derived from FRED data. We overlay mobility graphs over maps of Iowa and New Hampshire, where the size of each node is proportional to the population size of the subpopulation in which it is centered. Likewise, the width of each edge $e \in E_{ij}$ is proportional to its weight w_{ij} (number of individuals commuting from subpopulation i to subpopulation j).

Fig B in S1 Text. Percentage MaxCasesAverted and percentage MaxPeaksReduced for all approaches in New Hampshire under low infectivity. Most methods are able to save all individuals across all budgets for this small problem instance, with Random being the lowest performing method.

Fig C in S1 Text. Percentage MaxCasesAverted and percentage MaxPeaksReduced for UnitGreedy, SingletonGreedy, FastGreedy and baselines in Iowa under low infectivity. The effectiveness of the greedy methods is largely unchanged from that of the small problem instances (New Hampshire), but the baseline methods begin to decrease in performance.

Fig D in S1 Text. Percentage MaxCasesAverted and percentage MaxPeaksReduced for UnitGreedy, FastGreedy and baselines in Texas under low infectivity. For the SafeGraph mobility graph, all methods are able to save most individuals for all budgets, unlike for the FRED mobility graph, where the performance decreases for smaller budgets.

S1 Text. Contains Supplementary Information sections A–D, detailing model derivation, approximation guarantee proofs, descriptions of mobility graph construction, parameters, additional experiments, and related work.
(PDF)

Table A in S1 Text. Comparison of FRED and SafeGraph mobility graph properties. Contains properties of the mobility graphs constructed from FRED and SafeGraph data in New Hampshire, Iowa, and Texas.

Table B in S1 Text. System specifications for experiments. Contains information on the CPU type, memory, and storage where we run experiments.

Table C in S1 Text. Metapopulation model notation. Summary of the notation used for the metapopulation disease model.

Table D in S1 Text. Problem formulations and algorithm notation. Summary of the notation used for problem formulations and algorithms.
(PDF)

Acknowledgments

The authors acknowledge feedback from members of the Computational Epidemiology research group at the University of Iowa and the CDC MInD-Healthcare group.

Author contributions

Conceptualization: Jeffrey Keithley, Bijaya Adhikari, Sriram Pemmaraju.

Data curation: Jeffrey Keithley, Akash Choudhuri.

Formal analysis: Jeffrey Keithley, Bijaya Adhikari, Sriram Pemmaraju.

Funding acquisition: Bijaya Adhikari, Sriram Pemmaraju.

Investigation: Jeffrey Keithley, Akash Choudhuri.

Methodology: Jeffrey Keithley, Akash Choudhuri, Bijaya Adhikari, Sriram Pemmaraju.

Project administration: Bijaya Adhikari, Sriram Pemmaraju.

Resources: Bijaya Adhikari, Sriram Pemmaraju.

Software: Jeffrey Keithley, Akash Choudhuri.

Supervision: Bijaya Adhikari, Sriram Pemmaraju.

Validation: Jeffrey Keithley, Sriram Pemmaraju.

Visualization: Jeffrey Keithley, Bijaya Adhikari.

Writing – original draft: Jeffrey Keithley, Akash Choudhuri, Bijaya Adhikari, Sriram Pemmaraju.

Writing – review & editing: Jeffrey Keithley, Akash Choudhuri, Bijaya Adhikari, Sriram Pemmaraju.

References

1. Srivastava V, Priyadarshini S. Vaccine shortage dents India's coronavirus adult immunisation drive. Nature India. 2021.

2. Liu K, Lou Y. Optimizing COVID-19 vaccination programs during vaccine shortages. Infect Dis Modell. 2022;7(1):286–98.
3. Matrajt L Jr, Longini IM. Optimizing vaccine allocation at different points in time during an epidemic. PLoS One. 2010;5(11):e13767. https://doi.org/10.1371/journal.pone.0013767 PMID: 21085686
4. Mylius SD, Hagenaars TJ, Lugnér AK, Wallinga J. Optimal allocation of pandemic influenza vaccine depends on age, risk and timing. Vaccine. 2008;26(29):3742–9.
5. Kitagawa T, Wang G. Who should get vaccinated? Individualized allocation of vaccines over SIR network. J Econometrics. 2023;232(1):109–31.
6. Lemaitre JC, Pasetto D, Zanon M, Bertuzzo E, Mari L, Miccoli S, et al. Optimal control of the spatial allocation of COVID-19 vaccines: Italy as a case study. PLoS Comput Biol. 2022;18(7):e1010237. https://doi.org/10.1371/journal.pcbi.1010237 PMID: 35802755
7. Medlock J, Galvani AP. Optimizing influenza vaccine distribution. Science. 2009;325(5948):1705–8. https://doi.org/10.1126/science.1175570 PMID: 19696313
8. Zhang Y, Adiga A, Saha S, Vullikanti A, Prakash BA. Near-optimal algorithms for controlling propagation at group scale on networks. IEEE Trans Knowl Data Eng. 2016;28(12):3339–52.
9. Sambaturu P, Adhikari B, Prakash BA, Venkatramanan S, Vullikanti A. Designing effective and practical interventions to contain epidemics. In: International Conference on Autonomous Agents and MultiAgent Systems. 2020. p. 1187–95.
10. Zhang Y, Prakash BA. Data-aware vaccine allocation over large networks. ACM Trans Knowl Discov Data. 2015;10(2):1–32.
11. Grenfell B, Harwood J. (Meta)population dynamics of infectious diseases. Trends Ecol Evol. 1997;12(10):395–9. https://doi.org/10.1016/s0169-5347(97)01174-9 PMID: 21238122
12. Calvetti D, Hoover A, Rosea J, Somersalo E. Metapopulation network models for understanding, predicting, and managing the coronavirus disease COVID-19. Front Phys. 2020;8(261).
13. Fisher ML, Nemhauser GL, Wolsey LA. An analysis of approximations for maximizing submodular set functions—II. Springer; 1978.
14. Edmonds J. Submodular functions, matroids, and certain polyhedra. Combinatorial optimization—eureka, you shrink! papers dedicated to Jack Edmonds 5th international workshop. Springer; 2003. p. 11–26.
15. Iwata S. Submodular function minimization. Math Program. 2008;112:45–64.
16. Krause A, Golovin D. Submodular function maximization. Tractability. J Mach Learn Res. 2014;3(3):71–104.
17. Iyer RK, Bilmes JA. Submodular optimization with submodular cover and submodular knapsack constraints. Adv Neural InfProcess Syst. 2013;26.
18. Svitkina Z, Fleischer L. Submodular approximation: sampling-based algorithms and lower bounds. SIAM J Comput. 2011;40(6):1715–37.
19. Alon N, Gamzu I, Tennenholtz M. Optimizing budget allocation among channels and influencers. In: Proceedings of the 21st International Conference on World Wide Web. 2012. p. 381–8. https://doi.org/10.1145/2187836.2187888
20. Soma T, Kakimura N, Inaba K, Kawarabayashi K. Optimal budget allocation: theoretical guarantee and efficient algorithm. In: Proceedings of the 31st International Conference on Machine Learning. 2014. p. 351–9.
21. Soma T, Yoshida Y. A generalization of submodular cover via the diminishing return property on the integer lattice. In: Advances in Neural Information Processing Systems. 2015.
22. Zhang H, Vorobeychik Y. Submodular optimization with routing constraints. In: Proceedings of the AAAI Conference on Artificial Intelligence. 2016.
23. Das A, Kempe D. Approximate submodularity and its applications: subset selection, sparse approximation and dictionary selection. J Mach Learn Res. 2018;19(3):1–34.
24. Qian C, Zhang Y, Tang K, Yao X. On multiset selection with size constraints. In: Proceedings of the AAAI Conference on Artificial Intelligence. 2018.
25. Kuhnle A, Smith D, Crawford V, Thai M. Fast maximization of non-submodular, monotonic functions on the integer lattice. In: Proceedings of the 35th International Conference on Machine Learning, 2018. p. 2786–95.
26. Bian AA, Buhmann JM, Krause A, Tschiatschek S. Guarantees for greedy maximization of non-submodular functions with applications. In: Proceedings of the 34th International Conference on Machine Learning. 2017. p. 498–507.
27. Kermack W, McKendrick A. A contribution to the mathematical theory of epidemics. Proc Roy Soc Lond. 1927;15(772):700–21.

28. Nemhauser G, Wolsey L, Fisher M. An analysis of approximations for maximizing submodular set functions–I. Math Program. 1978;14(1):265–94.
29. Feige U. A threshold of Ln n for approximating set cover. J ACM. 1998;45(4):634–52.
30. Grefenstette JJ, Brown ST, Rosenfeld R, DePasse J, Stone NTB, Cooley PC, et al. FRED (a framework for reconstructing epidemic dynamics): an open-source software system for modeling infectious diseases and control strategies using census-based populations. BMC Public Health. 2013;13:940. https://doi.org/10.1186/1471-2458-13-940 PMID: 24103508
31. Safegraph. Places data curated for accurate geospatial analytics. 2022. https://www.safegraph.com/
32. Garey MR, Johnson DS. Computers and intractability: a guide to the theory of NP-completeness. New York: W. H. Freeman; 1979.
33. Xavier EC. A note on a maximum k-subset Intersection problem. Inf Process Lett. 2012;112(12):471–2. https://doi.org/10.1016/j.ipl.2012.03.007
34. Badanidiyuru A, Vondrák J. Fast algorithms for maximizing submodular functions. In: Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms; 2014. p. 1497–514
35. Pei S, Kandula S, Shaman J. Differential effects of intervention timing on COVID-19 spread in the United States. Sci Adv. 2020;6(49):eabd6370. https://doi.org/10.1126/sciadv.abd6370 PMID: 33158911
36. SafeGraph. Places data curated for accurate geospatial analytics. 2023. https://safegraph.com
37. Cohen E, Delling D, Pajor T, Wernack RF. Sketch-based influence maximization and computation: scaling up with guarantees. In: Proceedings of the 23rd ACM international conference on conference on information and knowledge management. 2014. p. 629–38.