# Branch-and-Price for Prescriptive Contagion Analytics

by

Martin Ramé

Diplôme d'Ingénieur, CentraleSupélec (2023)

Submitted to the Sloan School of Management
in partial fulfillment of the requirements for the degree of

Master of Science in Operations Research

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2023



Author .....
Sloan School of Management
May 12, 2023

Certified by .....
Alexandre Jacquilat
Assistant Professor of Operations Research and Statistics
Thesis Supervisor

Accepted by .....
Georgia Perakis
William F. Pounds Professor of Management Science
Co-director, Operations Research Center

**Branch-and-Price for Prescriptive Contagion Analytics**

by

Martin Ramé

Submitted to the Sloan School of Management
on May 12, 2023, in partial fulfillment of the
requirements for the degree of
Master of Science in Operations Research

## Abstract

Contagion models are ubiquitous in epidemiology, social sciences, engineering, and management. This thesis formalizes prescriptive contagion analytics problems where a centralized decision-maker allocates shared resources across multiple segments of a population, each governed by contagion dynamics. We define four real-world problems under this umbrella: distributing vaccines, deploying vaccination centers, mitigating urban congestion, and promoting online content. Prescriptive contagion problems involve mixed-integer non-convex optimization models with constraints governed by ordinary differential equations, thus combining the challenges of combinatorial optimization, non-linear optimization, and continuous-time system dynamics. This thesis develops a branch-and-price methodology for these problems based on: (i) a set partitioning reformulation; (ii) a column generation decomposition; (iii) a novel state clustering algorithm for discrete-decision continuous-state dynamic programming; and (iv) a novel tri-partite branching scheme to circumvent non-linearities. Extensive experiments show that the algorithm scales to large and otherwise- intractable instances, significantly outperforming state-of-the-art benchmarks. Our methodology provides a novel decision-making tool to support resource allocation in contagion systems. In particular, its application can increase the effectiveness of vaccination campaigns by an estimated 50-70%, resulting in 12,000 extra saved lives over 12 weeks in a situation mirroring the COVID-19 pandemic.

Thesis Supervisor: Alexandre Jacquilat
Title: Assistant Professor of Operations Research and Statistics

# Acknowledgments

Throughout my time in the Operations Research Center at MIT, I am extremely fortunate to have been advised by Alexandre Jacquillat. I would like to express my sincere thanks to him for his unwavering guidance and support over the past two years. Alex, thank you for pushing me to work on such interesting problems, for always raising insightful and challenging questions, and for believing in my success even when I did not. Thank you for not only being an incredibly inspiring academic advisor but also a caring mentor. I would also like to thank Kai Wang very warmly. Thank you for all the hours spent discussing research, and giving me help anytime I needed some on any topic, from mathematical understanding to coding errors. I also address a very sincere thank you to Michael Li who helped me a lot through the development of my thesis and has always been available to help. I wish the three of you an outstanding academic career.

My time at MIT would not have been the same without all the friends I met during my stay. I would like to thank the team of companions with whom I undertook this trip: Jean-Guillaume, Aly, Amine, Guillaume, Henri, Gaspard, Elea, Raphaël, Ariel, Jean, Gauthier, Victor, Matthieu, and Amandine. You are all very inspiring individuals to me. I would also like to address a very special thank you to my friends from undergraduate studies in France: Nassim, Pierre-Louis, Théophile, François, and Jean-Baptiste for pushing me to achieve my best and making those years full of laughter. Acknowledging my friends from high school goes without saying. Valentin, Adrien, Martin, Anaëlle, Émile, Bastien, Charlotte, Hugo, Chiara, Paul, Alexandre, Louise, Jean-Baptiste, and Tom, thank you for being a support for so long, and especially during those last two years. Your presence and friendship are invaluable.

Finally, I would like to thank all that my family has done and continues to do for me. To my parents, thank you for giving me both the opportunity and the will to study and work in fields I am passionate about, for always being a supportive

and loving presence, and for pushing me to become a better person. To my sisters, Camille and Joséphine, thank you as well for all the lively discussions, laughter, and adventures we've had together. I am infinitely grateful for having grown up in such a loving family.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Motivation and Focus

Epidemiological models have played a central role throughout the COVID-19 pandemic. In the United States, the Center for Disease Control maintained an ensemble forecast based mainly on compartmental contagion models (63). At their core, these models rely on susceptible-infected (SI) dynamics, which express the number of new infections proportionally to the number of infected and susceptible individuals. These dynamics have been embedded into more complex models, capturing immunization upon recovery (SIR models), time lags from exposure to infection (SEIR models), immunization from vaccinations, asymptomatic cases, quarantine, hospitalization, mortality, etc. Collectively, these models have shown significant success toward guiding societal response to the COVID-19 pandemic (see 2, 41, 29, 79, 49, 15, among many others).

Contagion models have a long history, well beyond the COVID-19 pandemic. Dating back to (43), SI models have been extensively used to model infectious epidemics, such as influenza (25) and Ebola (16). In marketing, the seminal model of product adoption from (13) relies on similar dynamics to capture network externalities between existing "infected" users and potential "susceptible" adopters. This model has been applied to multi-generational products (48), movie box office revenues (27), the diffusion of online content (77), etc. Other applications of SI-based contagion models

include the propagation of urban congestion (66), of habits (75), of religious beliefs (44), of online rumors (71), etc. Beyond SI-based contagion models, dynamical systems have been leveraged to model climate change mitigation (76), firm performance (61), employee compensation (62), etc. Thus, dynamical systems are ubiquitous in the natural sciences, the social sciences, engineering, and management.

Motivated by the success of *predictive* contagion models, this thesis proposes *prescriptive* methods to optimize spatial-temporal resource allocation decisions in dynamical contagion systems—and in broader dynamical systems. Spatial-temporal allocation refers to a setting where decisions are coupled across multiple segments of a population, each governed by contagion dynamics. To demonstrate the broad applicability of the methodology, we define four problems under this umbrella. The first one involves distributing a vaccine stockpile to combat an epidemic. The second one deploys mass vaccination centers, adding a facility location structure to the first one (21). The third one optimizes content promotion on social media to maximize product adoption (50). The last one deploys emergency vehicles to curb urban congestion, based on a new contagion model of traffic congestion developed with real-world data from Singapore.

Prescriptive dynamical systems—in particular, prescriptive contagion models—involve a mixed-integer non-convex optimization structure with constraints governed by ordinary differential equations (ODE). They are therefore challenging to even formulate, let alone to solve to optimality. The spatial-temporal resource allocation component, by itself, involves a mixed-integer optimization structure, but the problem is complicated by three complexities at the core of dynamical systems:

1. Continuous time dynamics: dynamical systems are governed by ODEs of the form $\frac{d\boldsymbol{M}_i(t)}{dt} = f_i(\boldsymbol{M}_i(t), \boldsymbol{x}_i(t))$, where the vector $\boldsymbol{M}_i(t)$ denotes a state variable in segment $i = 1, \cdots, n$; the vector $\boldsymbol{x}_i(t)$ denotes a control variable; and $f_i(\cdot, \cdot)$ refers to a continuous-time transition function. An easy workaround involves time discretization, to approximate the dynamics by $\frac{\boldsymbol{M}_i(t+\Delta t)-\boldsymbol{M}_i(t)}{\Delta t} = f_i(\boldsymbol{M}_i(t), \boldsymbol{x}_i(t))$. However, this can lead to extensive computational requirements if the time increment $\Delta t$ is too small, or to large approximation errors if

it is too large—or both. This is particularly salient in contagion systems, which can be highly non-linear and therefore highly sensitive to even small perturbations in initial conditions or in model parameters. A well-known incarnation of these non-linearities lies in a disease spreading if the basic reproduction number $R_0$ satisfies $R_0 > 1$ but not if $R_0 < 1$.

2. Non-linear interactions: dynamical systems may involve a non-linear transition function $f_i(\cdot, \boldsymbol{x}_i(t))$ for a given control variable $\boldsymbol{x}_i(t)$. As a prime example, contagion systems are driven by bilinear interactions of the form $\frac{dS_i(t)}{dt} = \alpha S_i(t) I_i(t)$, where $S_i(t)$ and $I_i(t)$ refer to the susceptible and infected populations. So, even with time discretization, the optimization problem would exhibit a mixed-integer non-convex structure. Accordingly, prescriptive contagion models have typically been solved via approximations and heuristics (see Section 1.3), but no exact method has been devised to handle resource allocation in non-linear dynamical systems.

3. Non-linear effects of interventions: system dynamics depend endogenously on resource allocation decisions. This dependency can be linear. In epidemiology, for instance, immunizations can be safely assumed to be proportional to vaccination rates. But interventions can also induce a non-linear transition function $f_i(\boldsymbol{M}_i(t), \cdot)$ for a given state variable $\boldsymbol{M}_i(t)$. In drug addiction, for instance, (14) model the effect of prevention interventions via an exponential decay and assume that treatment interventions exhibit diminishing returns. Again, these non-linearities create significant complexities in the optimization.

## 1.2 Contributions and outline

In response, this thesis develops a branch-and-price methodology for prescriptive contagion analytics. This approach separates the coupled resource allocation decisions in a master problem formulated via mixed-integer linear optimization, and segment-specific dynamics in a pricing problem formulated via dynamic programming. To our knowledge, it provides the first exact methodology to formulate and solve spatial-

temporal resource allocation problems to optimality in contagion systems—and, more generally, in non-linear dynamical systems.

Specifically, we first formalize a class of spatial-temporal resource allocation problems with endogenous continuous-time non-linear dynamics (Section 2). We consider a finite-horizon setting with multiple segments of a population, each constituting a dynamical system governed by non-linear ODEs. The decision-maker optimizes the discrete allocation of a shared resource in each period, which impacts the dynamics in each segment. This modeling framework allows a variety of objective functions and constraints in each segment—even non-linear ones—along with polyhedral constraints coupling resource allocations across segments (e.g., a shared budget). We apply this general-purpose formulation to define our four problems: distributing vaccines, deploying mass vaccination centers, promoting online content, and mitigating urban congestion. A byproduct of this research is a new data-driven SIR-based model of traffic congestion that yields significant improvements in predictive performance against state-of-the-art benchmarks.

Next, this thesis provides an exact branch-and-price algorithm to solve the resulting mixed-integer non-linear optimization problems with ODE constraints (Section 3). The proposed methodology handles the three complexities at the core of prescriptive contagion analytics: continuous-time dynamics, bilinear interactions, and non-linear interventions. It consists of four components:

1. *A set partitioning reformulation.* This formulation optimizes a resource allocation plan in each segment over the entire planning horizon, as opposed to natural resource allocation decisions for each period. By pre-processing the system dynamics into the definition of plan-based variables, this formulation eliminates the continuous-time dynamics and the non-linearities of the problem. However, it comes at the cost of an exponential number of plan-based variables.

2. *A scalable column generation scheme to solve its linear relaxation by generating plan-based variables iteratively.* A master problem solves a coupled resource allocation problem based on a subset of plan-based variables. A pricing problem

adds new plans of negative reduced cost or proves that none exists. Thanks to the structure of the problem, the pricing problem can be decomposed into segment-specific dynamic programming models. Yet, the system dynamics lead to a continuous state space—a notorious challenge in dynamic programming.

3. *A clustering algorithm for discrete-decision continuous-state dynamic programming.* Forward-enumeration backward-induction algorithms can solve the pricing problem to optimality, but remain intractable in even small instances. Instead, we develop a linear-time state-clustering algorithm that exploits the natural concentration of states in dynamical systems, without relying on the cost function (which varies from one column generation to the next). The reduction in the size of the state space considerably enhances the scalability of the pricing problem to large instances arising in practice at small costs in terms of approximation error.

4. *A novel tri-partite branching scheme to circumvent the non-linearities of the system.* We embed the column generation procedure into a branch-and-price structure to restore the integrality of resource allocation decisions. We adopt the typical approach to branch on natural resource allocation variables, as opposed to plan-based variables. Because of the non-linear dynamics, however, integral resource allocation decisions may not map into equivalent integral plan-based variables. We therefore propose a tri-partite branching scheme that retains a natural branching structure, while guaranteeing finite convergence and optimality.

Finally, this thesis demonstrates the scalability of our methodology to otherwise-intractable prescriptive contagion analytics problems (Section 4). In the vaccine allocation, content promotion, and congestion mitigation problems, the algorithm returns provably optimal in manageable computational times, significantly outperforming state-of-the-art benchmarks. For instance, our algorithm can tackle vaccine allocation instances involving 20 decisions in each of 51 regions and 12 weeks, with $\mathcal{O}(20^{600})$ possible decisions overall. The vaccinations centers problem features a more

challenging facility location structure with linking constraints; still, the branch-and-price methodology generates high-quality solutions and strong solution guarantees in manageable computational times. From a technical standpoint, the methodology significantly outperforms off-the-shelf implementations based on mixed-integer bilinear solvers or discretization-based linearizations, and can even provide benefits as compared to a tailored coordinate descent heuristic. From a practical standpoint, the methodology can have a significant impact on the management of contagion-based systems. In the vaccine allocation case, for instance, the optimized solution can increase the number of lives saved by an estimated 50-70%, as compared to simpler epidemiological benchmarks. Ultimately, our prescriptive contagion analytics approach can deliver significant practical benefits across domains, by fine-tuning resource allocations based on spatial-temporal system dynamics.

## 1.3 Related Work

### 1.3.1 Prescriptive contagion analytics

The prevalence of contagion modeling has motivated prescriptive methods to optimize interventions. In epidemiology, (33) showed the benefits of social planning interventions against user equilibrium behaviors. In a two-region environment, (64) found, using an SIS model, that medical interventions should target the region with fewer infections, whereas (57) prescribed, using an SIRS model, to prioritize the region with more infections. (80) designed incentivization schemes for influenza vaccinations by embedding SI dynamics into an equilibrium model of vaccination behaviors. In the context of COVID-19, several models traded off health impacts and economic costs to design differentiated lockdowns across age tranches (1), to optimize the timing and duration of lockdowns (26), to optimize the intensity of lockdowns along with test-tracing quarantine (6), and to adjust the timing and intensity of lockdowns based on initial conditions (11).

In management, the Bass model has been extensively used to guide operational and

marketing decisions surrounding product innovation, spanning sales plans and time-to-market strategies (40); pricing, production and inventory decisions (72); multi-product pricing (47); free access and premium subscriptions (53); dynamic pricing (28); dynamic pricing under data-driven demand learning (82, 3); etc. The Bass model has also been used in related areas to design drug prevention and treatment policies (14), advertising campaigns (45), crowdfunding campaigns (81), etc. This body of work relies on dynamic programming to characterize optimal interventions in a single dynamical system (or two systems).

In contrast, this thesis considers spatial-temporal resource allocation decisions with coupling constraints across multiple contagion systems. Such coupling constraints appeared in the ventilator sharing problem from (55) and (20). However, since ventilator availability does not impact the dynamics of the pandemic, this problem could be formulated via mixed-integer linear optimization by decoupling upstream epidemiological predictions from downstream ventilator allocation decisions. In sharp contrast, vaccinations impact infection dynamics, requiring to integrate predictive epidemiological models into prescriptive resource allocation models. Thus, the problem considered in this thesis combines the difficulties of spatial-temporal resource allocation (with a mixed-integer optimization structure) and the difficulties of prescriptive contagion modeling (with a non-linear dynamic programming structure).

This thesis, therefore, falls into the optimization literature with endogenous contagion dynamics. In product adoption, (4) used a heuristic algorithm to optimize the deployment of mobile healthcare units, and (50) designed an approximation algorithm for online content promotion with a $1 - 1/e$ guarantee. In epidemiology, (52) used myopic linear optimization and approximate dynamic programming to allocate Ebola treatment units. (21) optimized where to open COVID-19 mass vaccination facilities and how to allocate vaccines in the United States, where the pandemic in each state follows the SI-based model from (49). (31) incorporated uncertainty in epidemiological forecasts into the problem, using robust optimization. To handle the bilinear interactions between susceptible and infected populations, existing methods include a coordinate descent heuristic (21) and linear approximations based on discretized

numbers of infections or McCormick reformulations (31). This thesis contributes to this literature with an exact optimization approach that does not rely on finite difference approximations of the continuous-time ODE dynamics and that does not rely on approximations and heuristics to handle the non-linearities of contagion dynamics.

### 1.3.2 Mixed-integer non-linear optimization.

Our problem exhibits a mixed-integer non-linear optimization (MINLO) structure with ODE constraints. Even with a finite difference approximation of ODEs, the problem, therefore, remains highly challenging due to its MINLO structure (see 24, for a detailed review of these problems). The special case of bilinear SI-based contagion models relates to recent work on mixed-integer non-convex quadratic optimization, such as the reformulation-linearization technique (RLT) (73), semi-definite relaxations (32, 9), disjunctive programming (69, 70), perspective reformulations (36, 10), etc. General-purpose methods for mixed-integer non-linear optimization include spatial branch-and-bound (46), branch-and-reduce (65, 78), and $\alpha$-branch-and-bound (8). Recent versions of (37) tackle mixed-integer non-convex quadratic optimization problems by combining RLT and spatial branch-and-bound, which we use as a benchmark in Section 4.

The ODE constraints in our problem link to the mixed-integer optimal control literature. Solution approaches include time discretization, continuous relaxations, adaptive discretization, and integer rounding (39, 67). When the feasible region can be enumerated exhaustively, partial outer approximation and sum-up rounding can converge to the optimal solution as time discretization becomes infinitely granular (68, 38, 54). In practice, however, very granular time discretization may be required to generate high-quality solutions; moreover, optimality is not guaranteed in the presence of control costs or of combinatorial constraints—two important features of our problem. (42) decomposed the problem into a continuous optimal control problem and a combinatorial integral approximation problem (CIAP) to restore integer feasibility, using Lagrangian relaxation; (34) separated the mixed-integer optimal control part and the time-coupling combinatorial constraints; and (22) restored integral feasibility

via a shortest path formulation. This thesis departs from this literature by considering a class of problems with spatial coupling across multiple dynamical systems, and by proposing a new and exact branch-and-price methodology to solve it.

Since its introduction in (12), branch-and-price has been widely applied to mixed-integer linear optimization. (7) used it to solve a non-linear reliable $h$-paths problem, where the non-linearities stem from failure probabilities and are handled via a label-setting shortest path algorithm. (58) proposed an inner-and-outer-approximation of non-convex mixed-integer optimization problems by combining column generation with non-linear optimization. (5) synthesized a generic branch-and-price algorithm for non-convex mixed-integer optimization. This thesis contributes a tailored branch-and-price decomposition to solve spatial-temporal resource allocation problems over continuous-time dynamical systems—including, notably, a novel tri-partite branching disjunction to handle non-linearities.

Finally, one of the main bottlenecks of our methodology lies in the discrete-decision continuous-state dynamic programming structure of the pricing problem—a notoriously challenging class of problems. This relates to the approximate dynamic programming and reinforcement learning literature, which employs cost-to-go and policy approximations to avoid backward induction (see 18, 59, for reviews). Our continuous-state structure, in particular, can be tackled via state discretization. Static state discretization converges to the true optimum as grids become increasingly granular (17). However, the number of states can grow prohibitively large in practice. Adaptive discretization approaches therefore aim to build a fine grid where the cost function changes rapidly but to maintain a coarse grid elsewhere (35, 23). Starting with (19), several studies combined adaptive discretization with reinforcement learning to aggregate states with similar cost-to-go functions (see, e.g. 60, 51, 74). In our problem, however, the cost function changes at each column generation iteration, motivating our state-clustering approach that exploits the fact that the state space does not change across iterations.

# Chapter 2

# Prescriptive contagion analytics: definition and formulation

## 2.1 General model formulation

We consider a general problem of spatial-temporal resource allocation in dynamical systems, depicted in Figure 2-1. The "spatial" component refers to resource allocation across $n$ segments of a population, which correspond to different locations (e.g., state or countries) or to other entities (e.g., products). The temporal component refers to $S$ decision epochs throughout a planning horizon of length $T$. We denote by $\tau_s$ the time stamp of epoch $s = 1, \cdots, S$, with $\tau_1 = 0$ and $\tau_{S+1} = T$.

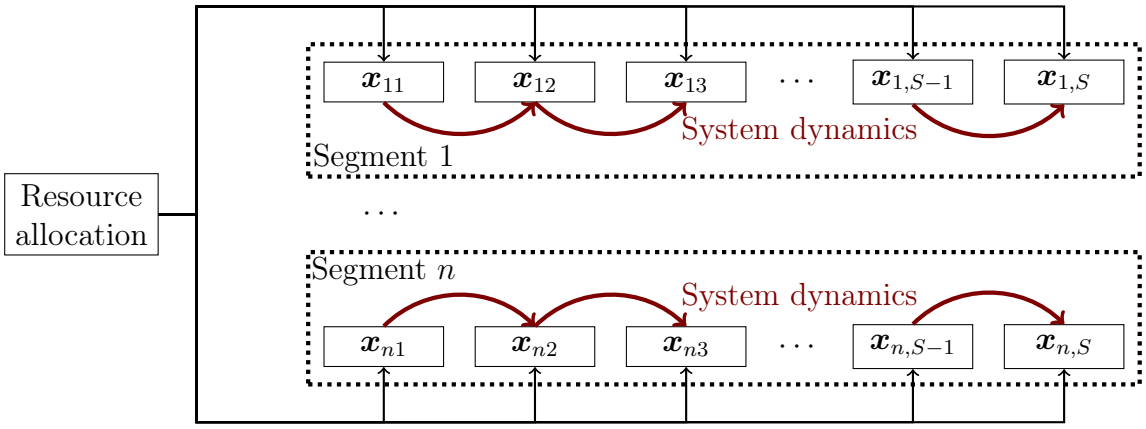



Figure 2-1: Schematic representation of the spatial-temporal resource allocation problem in dynamical systems.

We characterize centralized resource allocation decisions via variables $\boldsymbol{x}_{is} \in \mathcal{F}_{is} \subseteq \mathbb{R}^{d_{is}}$ for each segment $i = 1, \cdots, n$ and each epoch $s = 1, \cdots, S$. These variables are defined as vectors to allow for multi-dimensional resource allocation (e.g., treatment vs. prevention resources, in our traffic mitigation problem). We also introduce another variable $\boldsymbol{y} \in \mathbb{Z}^q \times \mathbb{R}^r$ to capture any other decision (e.g., facility location in our vaccination centers problem). Let $\Delta_s = \sum_{i=1}^{n} d_{is}$ and let $\boldsymbol{X}_s \in \mathbb{R}^{\Delta_s}$ denote the concatenated resource allocation variable at epoch $s = 1, \cdots, S$. Resource allocation in segment $i = 1, \cdots, n$ at epoch $s = 1, \cdots, S$ comes at a cost $C_{is}(\boldsymbol{x}_{is})$, and other decisions come at a linear cost $\boldsymbol{d}^\top \boldsymbol{y}$. We make no restriction on the form of the feasible regions $\mathcal{F}_{is}$ and the cost functions $C_{is}(\cdot)$ governing resource allocation decisions. Importantly, we focus on discrete resource allocation decisions, where each region $\mathcal{F}_{is}$ has finite cardinality $D_{is} = |\mathcal{F}_{is}| < \infty$. In many cases, resources are naturally restricted to discrete quantities (e.g., emergency vehicles, in congestion mitigation). Even when the decision space is infinite, operational constraints often lead to a discretized number of possible decisions. For example, (56) shipped COVID-19 vaccines in pallets of around 20,000 vaccines. Still, the resource allocation problem under consideration remains high-dimensional due to coupled spatial-temporal decisions—for instance, if $D$ decisions are available in each segment at each epoch, the decision space grows in $\mathcal{O}(D^{n \times S})$.

Besides segment-specific constraints captured in $\mathcal{F}_{is}$, global constraints are characterized by a polyhedral set $\left\{(\boldsymbol{X}, \boldsymbol{y}) \in \mathbb{R}^{\Delta_s + q + r} : \boldsymbol{U}_s \boldsymbol{X} + \boldsymbol{V}_s \boldsymbol{y} \geq \boldsymbol{w}_s\right\}$, where $\boldsymbol{U}_s \in \mathbb{R}^{m_s \times \Delta_s}$, $\boldsymbol{V}_s \in \mathbb{R}^{m_s \times (q+r)}$ and $\boldsymbol{w}_s \in \mathbb{R}^{m_s}$. In other words, resource allocation decisions are subject to the following linear constraints, where $\boldsymbol{u}_{sji} \in \mathbb{R}^{d_{is}}$ and $\boldsymbol{v}_{sj} \in \mathbb{R}^{q+r}$ denote the vectors corresponding to the $j^{\text{th}}$ constraint and the $i^{\text{th}}$ segment in $\boldsymbol{U}_s$ and to the $j^{\text{th}}$ constraint in $\boldsymbol{V}_s$, respectively:

$$\left(\sum_{i=1}^{n} \boldsymbol{u}_{sji}^\top \boldsymbol{x}_{is}\right) + \boldsymbol{v}_{sj}^\top \boldsymbol{y} \geq w_{sj}, \ \forall s = 1, \cdots, S, \ \forall j = 1, \cdots, m_s$$

Each population segment $i = 1, \cdots, n$ constitutes a dynamical system, governed by a continuous-time state variable $\boldsymbol{M}_i(t) \in \mathbb{R}^{r_i}$. We assume that the dynamics are

independent across the $n$ segments. In the epidemiological context, for instance, that means that the population interacts within each state (or each country) but not across regions (see, e.g. 41, 79, 49, 15, for similar assumptions). Specifically, in each segment $i = 1, \cdots, n$, the state variable $\boldsymbol{M}_i(t)$ is determined by an initial condition $\boldsymbol{M}_i^0$, and varies according to the following system of ODEs, where $f_i(\cdot, \cdot)$ denotes the transition function:

$$\frac{d\boldsymbol{M}_i(t)}{dt} = f_i(\boldsymbol{M}_i(t), \boldsymbol{x}_{is}),\ \forall t \in [\tau_s, \tau_{s+1}]\,.$$

The dynamical system is also associated with a cost function, which comprises a continuous-time component $g_{it}(\boldsymbol{M}_i(t))$ and a terminal component $h_i(\boldsymbol{M}_i(T))$. By design, we impose no restrictions on the structure of the transition function $f_i(\cdot, \cdot)$ and the cost functions $g_{it}(\cdot)$ and $h_i(\cdot)$, so our framework is sufficiently general to encompass a large class of non-linear dynamical systems.

The spatial-temporal resource allocation problem, referred to as Problem $(\mathcal{P})$, minimizes total costs subject to resource allocation constraints and the system's dynamics. It is written as follows:

$$(\mathcal{P}) \quad \min \quad \sum_{i=1}^{n} \left( \int_0^T g_{it}(\boldsymbol{M}_i(t))dt + h_i(\boldsymbol{M}_i(T)) \right) + \sum_{i=1}^{n} \sum_{s=1}^{S} C_{is}(\boldsymbol{x}_{is}) + \boldsymbol{d}^\top \boldsymbol{y} \tag{2.1}$$

$$\text{s.t.} \quad \left( \sum_{i=1}^{n} \boldsymbol{u}_{sji}^\top \boldsymbol{x}_{is} \right) + \boldsymbol{v}_{sj}^\top \boldsymbol{y} \geq w_{sj},\ \forall s = 1, \cdots, S,\ \forall j = 1, \cdots, m_s \tag{2.2}$$

$$\frac{d\boldsymbol{M}_i(t)}{dt} = f_i(\boldsymbol{M}_i(t), \boldsymbol{x}_{is}),\ \forall i = 1, \cdots, n,\ \forall s = 1, \cdots, S,\ \forall t \in [\tau_s, \tau_{s+1}] \tag{2.3}$$

$$\boldsymbol{M}_i(0) = \boldsymbol{M}_i^0,\ \forall i = 1, \cdots, n \tag{2.4}$$

$$\boldsymbol{x}_{is} \in \mathcal{F}_{is},\ \forall i = 1, \cdots, n,\ \forall s = 1, \cdots, S \tag{2.5}$$

$$\boldsymbol{y} \in \mathbb{Z}^q \times \mathbb{R}^r \tag{2.6}$$

This formulation remains intractable due to continuous time dynamics (Equation (2.3)), possible non-convex system dynamics (functions $f_i(\cdot, \cdot)$, $g_{it}(\cdot)$ and $h_i(\cdot)$) and possible non-convex segment-wise decisions (functions $C_{is}(\cdot)$, regions $\mathcal{F}_{is}$). One workaround would be to model $(\mathcal{P})$ directly via dynamic programming. At each de-

cision epoch, the state variable would need to store *all* variables $\boldsymbol{M}_1(\tau_s), \cdots, \boldsymbol{M}_n(\tau_s)$ as well all required information to enforce the resource allocation constraints (Equation (2.2)). This approach, however, would scale in $\mathcal{O}(D^{n\times S})$ with $D$ possible decisions in each segment at each epoch, thus quickly growing into the curse of dimensionality (59). Instead, we propose in Section 3 a branch-and-price method that separates the combinatorial difficulties of managing coupled resource allocations across segments (via a master problem) and the difficulties of controlling continuous-time dynamics in each segment (via a pricing problem).

Before proceeding, let us underscore an important characteristic of the model related to the state variables $\boldsymbol{M}_i(t)$. Following (59), we define it as a necessary and sufficient function of history to compute the cost function, the constraints, and the transition function. Accordingly, the dynamics are separable over time in each segment, which we exploit in our algorithm. For example, if a decision-maker faced inter-temporal budget constraints of the form $\sum_{s=1}^{S} \boldsymbol{x}_{is} \leq \boldsymbol{\gamma}_i$, the state variable would need to be expanded to embed the budget used up to epoch $s$. In contrast, our model accommodates budget constraints across segments of the form $\sum_{i=1}^{n} \boldsymbol{x}_{is} \leq \boldsymbol{\beta}_s$. The key feature of our modeling approach, however, is the segment-wise decomposition induced by the segment-specific state variables $\boldsymbol{M}_i(t) \in \mathbb{R}^{r_i}$, as opposed to relying on a single higher-dimensional state variable of the form $(\boldsymbol{M}_1(\tau_s), \cdots, \boldsymbol{M}_n(\tau_s)) \in \mathbb{R}^{r_1} \times \cdots \times \mathbb{R}^{r_n}$ as would be required by a full dynamic programming formulation. We still use dynamic programming in the pricing problem, but our segment-wise decomposition enables it to scale in $\mathcal{O}(D^S)$, as opposed to $\mathcal{O}(D^{n\times S})$.

## 2.2 Prescriptive contagion model and applications

An important motivation involves spatial-temporal resource allocation in contagion systems. We derive this formulation from Problem $(\mathcal{P})$, where each segment is broken down into a susceptible state $S(t) \in \mathbb{R}$, an infected state $I(t) \in \mathbb{R}$, and other states generically represented via $\boldsymbol{R}(t) \in \mathbb{R}^V$. The key feature lies in bilinear interactions between the susceptible and infected populations. The dynamics are as follows, with

an infection rate $\alpha$ and transition functions $f_i^S(\cdot)$, $f_i^I(\cdot)$ and $f_i^R(\cdot)$:

$$\frac{dS_i(t)}{dt} = -\alpha S(t)I(t) + f_i^S(S(t), I(t), \boldsymbol{R}(t)) \tag{2.7}$$

$$\frac{dI_i(t)}{dt} = +\alpha S(t)I(t) + f_i^I(S(t), I(t), \boldsymbol{R}(t)) \tag{2.8}$$

$$\frac{dR_{iv}(t)}{dt} = f_i^R(S(t), I(t), \boldsymbol{R}(t)), \quad \forall v = 1, \cdots, V \tag{2.9}$$

Similarly, we define functions $g_{it}^{SIR}(\cdot)$ and $h_i^{SIR}(\cdot)$ to characterize the costs of the system dynamics in each segment. We then formulate the prescriptive contagion model as follows:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} \left( \int_0^T g_{it}^{SIR}(S(t), I(t), \boldsymbol{R}(t))dt + h_i^{SIR}(S(T), I(T), \boldsymbol{R}(T)) \right) \\ & + \sum_{i=1}^{n} \sum_{s=1}^{S} C_{is}(\boldsymbol{x}_{is}) + \boldsymbol{d}^\top \boldsymbol{y} \end{aligned} \tag{2.10}$$

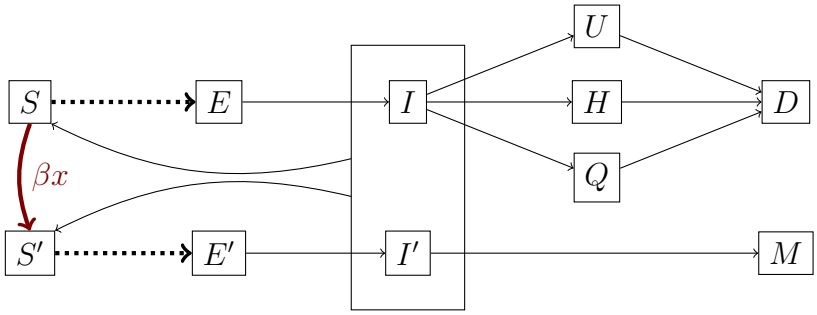
s.t. Resource allocation constraints (Equation (2.2) and Equations (2.5)–(2.6))

Contagion dynamics (initial conditions, and Equations (2.7)–(2.9))

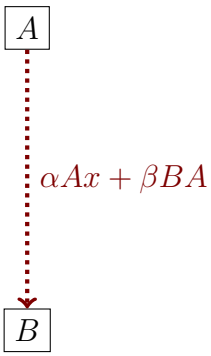
We define our four problems from this generic formulation, all inspired from real-world problems and data-driven contagion models.

**Vaccine allocation.** The problem involves allocating a weekly stockpile of vaccines across 51 regions (50 US states plus Washington, DC) to minimize the death toll of COVID-19. We model the pandemic via the DELPHI-V model shown in Figure 2-2a, with 7 non-vaccinated states (susceptible (S), exposed (E), infected (I), undetected (U), hospitalizations (H), quarantine (Q), death (D)) and 4 vaccinated states (susceptible (S'), exposed (E'), infected (I'), immunized (M)).

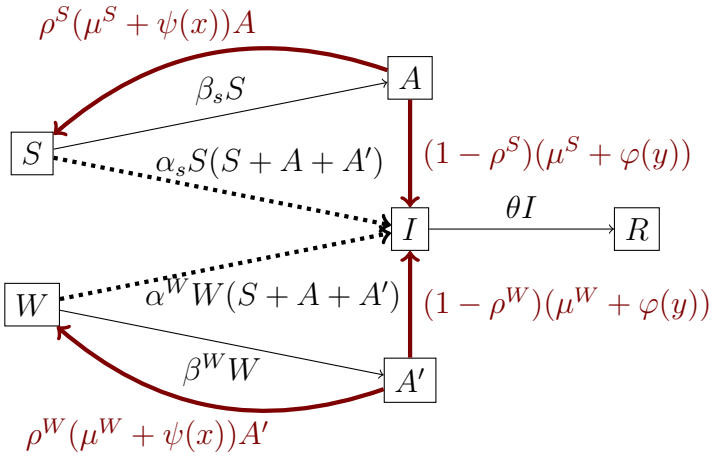
A centralized decision-maker distributes $B_s$ vaccine doses each week, captured via the decision epochs $s = 1, \cdots, S$. Each segment represents one of 51 US jurisdictions (50 states plus Washington, DC). The dynamics in each jurisdiction are characterized by a contagion model, shown in Figure 2-2a. We refer to (49) and (21) for details on the model and its empirical performance.




(a) Vaccine allocation and vaccination centers (21).




(b) Content promotion (50).




(c) Traffic congestion (this thesis).

Figure 2-2: Contagion models for the four problems. Thick red lines indicate transitions that are endogenous to resource allocation. Dotted lines indicate bilinear susceptible-infected interactions.

The decision variable $x_{is}$ denotes the number of vaccines allocated to jurisdiction $i = 1, \cdots, n$ at epoch $s = 1, \cdots, S$. For convenience, let us denote by $\overline{x}_i(t)$ a piece-wise constant function such that $\overline{x}_i(t) = \frac{x_{is}}{\tau_{s+1} - \tau_s}$ for all $\tau_s \leq t \leq \tau_{s+1}$. Vaccinations induce a transfer of the susceptible population to the alternative susceptible state, proportionally to an effectiveness parameter $\beta$. This model assumes conservatively that vaccinations do not prevent infections but immunize patients. The system dynamics are then governed by the following ordinary differential equations:

$$\frac{dS_i(t)}{dt} = -\alpha\gamma(t)\left(S_i(t) - \beta\overline{x}_i(t)\right) I_i(t) - \beta\overline{x}_i(t) \tag{2.11}$$

$$\frac{dE_i(t)}{dt} = \alpha\gamma(t)\left(S_i(t) - \beta\overline{x}_i(t)\right) I_i(t) - r^I E_i(t) \tag{2.12}$$

$$\frac{dI_i(t)}{dt} = r^I E_i(t) - r^d I_i(t) \tag{2.13}$$

$$\frac{dU_i(t)}{dt} = r_k^U(t) I_i(t) - r^D U_i(t) \tag{2.14}$$

$$\frac{dH_i(t)}{dt} = r_k^H(t) I_i(t) - r^D H_i(t) \tag{2.15}$$

$$\frac{dQ_i(t)}{dt} = r_k^Q(t) I_i(t) - r^D Q_i(t) \tag{2.16}$$

$$\frac{dD_i(t)}{dt} = r^D\left(U_i(t) + H_i(t) + Q_i(t)\right) \tag{2.17}$$

$$\frac{dM_i(t)}{dt} = \beta\overline{x}_i(t) \tag{2.18}$$

The vaccine allocation problem is defined as follows. Equation (2.19) minimizes the number of deaths (an absorbing state) along with the terminal number of hospitalized, quarantined and exposed people. Constraint (2.20) applies the budget of vaccines in each period.

$$\min \quad \sum_{i=1}^{n} \left(c_D D_i(T) + c_H H_i(T) + c_Q Q_i(T) + c_E E_i(T)\right) \tag{2.19}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} x_i(s) \leq B_s \quad \forall s = 1, \cdots, S \tag{2.20}$$

$$\text{Equations (2.11)–(2.18)} \tag{2.21}$$

$$\text{Initial conditions} \tag{2.22}$$

$$\boldsymbol{S}, \boldsymbol{E}, \boldsymbol{I}, \boldsymbol{U}, \boldsymbol{H}, \boldsymbol{Q}, \boldsymbol{D}, \boldsymbol{M}, \boldsymbol{x} \geq \boldsymbol{0} \tag{2.23}$$

**Vaccination centers.** The problem involves setting up mass vaccination centers and allocating a weekly stockpile of vaccines (21) We still model the dynamics of the pandemic via the DELPHI-V model; we limit the distance that people will be willing to travel to access a vaccination center; and we maximize the number of saved lives through the vaccination campaign. We consider the same system dynamics as in the vaccine allocation problem, but impose a facility location structure governing the deployment of mass vaccination centers. We assume consider a set of $K$ candidate vaccination centers, and select a subset of $F$ vaccination centers. At each epoch $s$, we have a budget of $B_s$ vaccines to allocate. Finally, we impose a threshold on the maximum distance that a resident can travel to reach a vaccination center. Let $P_{ij}$ denote the number of residents in region $i$ that can be served by vaccination centers $j$. We then define the following variables:

$$w_j = \begin{cases} 1 & \text{if vaccination centers } j = 1, \cdots, K \text{ is built} \\ 0 & \text{otherwise} \end{cases}$$

$$v_{ijs} = \text{number of vaccines sent to region } i = 1, \cdots, n \text{ from center } j = 1, \cdots, K$$
$$\text{at epoch } s = 1, \cdots, S$$

The problem minimizes the number of deaths (Equation (2.24)). Constraint (2.25) ensures that exactly $F$ vaccination centers are built. Constraint (2.26) applies the

total budget of vaccines. Constraint (2.27) applies the capacity of each vaccination center, along with a logical constraint coupling facility location decisions and subsequent vaccine allocation decisions. Finally, constraint (2.28) ensures that vaccine allocation decisions are consistent with the coverage of each vaccination center.

$$\min \quad \sum_{i=1}^{n} \left(c_D D_i(T) + c_H H_i(T) + c_Q Q_i(T) + c_E E_i(T)\right) \tag{2.24}$$

$$\text{s.t.} \quad \sum_{j=1}^{K} w_j = F \tag{2.25}$$

$$\sum_{i=i}^{n} \sum_{j=1}^{K} v_{ijs} \leq B \quad \forall s = 1, \cdots, S \tag{2.26}$$

$$\sum_{i=1}^{n} v_{ijs} \leq C_j w_j \quad \forall s = 1, \cdots, S, \quad \forall j = 1, \cdots, K \tag{2.27}$$

$$\sum_{s=1}^{S} v_{ijs} \leq P_{ij} w_j \quad \forall i = 1, \cdots, n, \quad \forall j = 1, \cdots, K \tag{2.28}$$

$$\text{Equations (2.11)–(2.18)} \tag{2.29}$$

$$\text{Initial conditions} \tag{2.30}$$

$$\boldsymbol{S}, \boldsymbol{E}, \boldsymbol{I}, \boldsymbol{U}, \boldsymbol{H}, \boldsymbol{Q}, \boldsymbol{D}, \boldsymbol{M}, \boldsymbol{v} \geq \boldsymbol{0} \tag{2.31}$$

$$\boldsymbol{w} \in \{0, 1\}^K \tag{2.32}$$

**Content promotion.** This problem optimizes the share of the population to which each product shall be promoted online, subject to a sparsity constraint, a cover constraint, a linking constraint, and contagion dynamics. Each segment corresponds to a product, and product adoption follows a Bass model with susceptible users (A) and adopters (B) (Figure 2-2b). A social media platform needs to promote a portfolio of $n$ products. We define the following variables:

$$y_{is} = \begin{cases} 1 & \text{if product } i = 1, \cdots, n \text{ is promoted at epoch } s = 1, \cdots, S \\ 0 & \text{otherwise} \end{cases}$$

$x_{is}$ = population share to which product $i = 1, \cdots, n$ is promoted at epoch $s = 1, \cdots, S$

Again, we define a piece-wise constant function $\overline{x}_i(t) = x_{is}$ for all $\tau_s \leq t \leq \tau_{s+1}$. The contagion model captures product adoption dynamics via a Bass model, comprising susceptible users (A) and adopters (B). Product promotion increases adoption from external sources, which, in turn, increases positive network externalities between adopters and the susceptible population (Figure 2-2b). This model follows the one from (50), developed using data from a social media platform. The dynamics are defined as follows:

$$\frac{dA_i(t)}{dt} = -\alpha \overline{x}_i(t) A_i(t) - \frac{\beta}{m} S_i(t) B_i(t) \tag{2.33}$$

$$\frac{dB_i(t)}{dt} = \alpha \overline{x}_i(t) A_i(t) + \frac{\beta}{m} S_i(t) B_i(t) \tag{2.34}$$

The problem maximizes total adoption (Equation (2.35)), subject to a sparsity constraint ensuring that the platform promotes at most $B_s$ products per period (Constraint (2.36)), to a cover constraint ensuring that every user is shown a product (Constraint (2.37)), to a linking constraint ensuring consistency between the $\boldsymbol{x}$ and $\boldsymbol{y}$ variables (Constraint (2.38)), and to the contagion dynamics.

$$\max \quad \sum_{i=1}^{n} B_i(T) \tag{2.35}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} y_{is} \leq B_s \quad \forall s = 1, \cdots, S \tag{2.36}$$

$$\sum_{i=1}^{n} x_{is} = 1 \quad \forall s = 1, \cdots, S \tag{2.37}$$

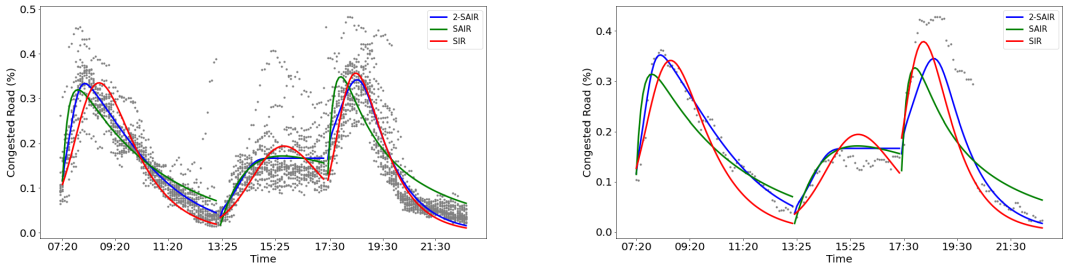
$$x_{is} \leq y_{is} \quad \forall s = 1, \cdots, S, i = 1, \cdots, n \tag{2.38}$$

$$\text{Equations (2.33)–(2.34)} \tag{2.39}$$

$$\text{Initial conditions} \tag{2.40}$$

$$\boldsymbol{S}, \boldsymbol{B}, \boldsymbol{x} \geq \boldsymbol{0}, \ \boldsymbol{y} \in \{0,1\}^{n \times S} \tag{2.41}$$

**Congestion mitigation.** This problem deploys emergency vehicles across neighborhoods to mitigate urban congestion. Prevention resources respond to minor accidents and prevent congestion, whereas treatment resources respond to major accidents




(a) In-Sample (b) Out-of-Sample

Figure 2-3: Comparison of the congestion level predicted by our three models. In-Sample is trained on data from a whole month. Out-of-Sample only shows the performance on one day.

and ease recovery. SIR models have been used recently to model the propagation of urban congestion from congested "infected" roads to free-flow "susceptible" roads (66). In this thesis, we leverage new data sources from Singapore on traffic speeds, road work, and traffic accidents to propose a richer six-state model, shown in Figure 2-2c: susceptible roads (S), road work (W), accidents (A), road work and accidents ($A'$), congested (I), and recovered (R). To come up with this model, we collected data in Singapore that tracks the average speed on each road every five minutes, the log of incidents in the city, and the road works in the city. We divide the city into five regions, and the population in each segment characterizes all the road segments. In each five-minute interval, a road segment is labeled as congested if the average speed recorded is less than 30% of the average speed on that road. We considered three contagion models: a basic Susceptible-Infected-Recovered (SIR) model inspired from (66), an SAIR model with an "accident" state, and the 2-SAIR model depicted in Figure 2-2c where we separate road segments with and without road work. We divided each day into a morning period (7am to 1pm), an afternoon period (1pm to 5:30pm) and an evening period (5:30pm onward). We trained our contagion models on those three periods separately, using historical data from March 2022. We then tested the models on data from April 2022 The results are reported in Figure 2-3 and Table 2-4.

Note that the 2-SAIR model consistently outperforms the two benchmarks, both in sample and out of sample. Its parameters can be found in Table 2-5. These param-

| | In sample | | | Out of sample | | |
|---|---|---|---|---|---|---|
| Model | Morning | Afternoon | Evening | Morning | Afternoon | Evening |
| SIR | 0.0551 | 0.0546 | 0.0586 | 0.0526 | 0.1028 | 0.2196 |
| SAIR | 0.0506 | 0.0502 | 0.0702 | 0.0434 | 0.1002 | 0.205 |
| 2-SAIR | **0.0389** | **0.0501** | **0.0533** | **0.0406** | **0.0998** | **0.1885** |

Figure 2-4: Root-mean-square error (RMSE) for each model on the three periods of the day.

| Time Period | Region | $\alpha^F$ | $\beta^F$ | $\rho^F$ | $\mu^F$ | $\alpha^W$ | $\beta^W$ | $\rho^W$ | $\mu^W$ | $\theta$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Morning | West | 0.531 | 0.578 | 1.126 | 0.569 | 3.966 | 1.585 | 0.411 | 4.835 | 10.354 |
| | Central | 0.000 | 0.048 | 0.034 | 0.076 | 1.224 | 0.875 | 1.180 | 1.544 | 2.135 |
| | Northeast | 0.000 | 0.137 | 0.345 | 0.078 | 1.240 | 0.198 | 1.023 | 2.273 | 4.742 |
| | East | 0.052 | 0.041 | 0.432 | 0.086 | 2.745 | 3.086 | 1.201 | 1.934 | 3.950 |
| | North | 0.492 | 0.562 | 1.450 | 2.043 | 1.396 | 0.896 | 2.162 | 1.878 | 1.113 |
| Afternoon | West | 0.897 | 0.180 | 1.913 | 3.180 | 2.852 | 1.321 | 1.208 | 3.444 | 2.512 |
| | Central | 0.862 | 0.070 | 3.259 | 1.144 | 1.098 | 1.026 | 0.338 | 1.831 | 3.477 |
| | Northeast | 0.782 | 0.024 | 5.786 | 0.283 | 1.215 | 2.360 | 0.002 | 3.555 | 1.381 |
| | East | 0.838 | 0.115 | 2.143 | 1.260 | 0.000 | 0.288 | 0.000 | 0.137 | 1.283 |
| | North | 1.538 | 0.196 | 1.856 | 1.656 | 0.987 | 1.085 | 0.996 | 0.930 | 1.111 |
| Evening | West | 0.091 | 0.448 | 1.249 | 1.176 | 0.568 | 0.990 | 2.194 | 1.264 | 3.775 |
| | Central | 0.024 | 0.331 | 1.176 | 0.793 | 0.068 | 0.733 | 1.873 | 1.335 | 3.350 |
| | Northeast | 0.018 | 0.304 | 1.247 | 1.326 | 0.116 | 1.514 | 1.728 | 1.123 | 2.861 |
| | East | 0.603 | 0.191 | 2.255 | 1.141 | 0.962 | 0.985 | 1.527 | 1.517 | 6.799 |
| | North | 0.001 | 0.580 | 0.905 | 0.698 | 1.148 | 1.110 | 1.136 | 1.142 | 4.264 |

Figure 2-5: Parameters of 2-SAIR for every region of the city.

eters provide some insights into the underlying dynamics; for instance, the accident rate and the congestion rate are both higher on road segments with road work than on road segments without road work. Ultimately, the model provides an adequate estimation of the congestion dynamics observed in the city, as illustrated in Figure 2-3. We can then move on to the problem formulation. We consider $n$ neighborhoods, and each population corresponds to the set of road segments in each neighborhood. The centralized decision-maker can deploy resources for prevention purposes (namely, to respond to minor accidents and prevent the formation of congestion) or for treatment purposes (namely, to respond to major accidents in order to clear the roads faster).

Accordingly, we define the following variables:

$x_{is}$ = number of treatment vehicles allocated to neighborhood $i = 1, \cdots, n$ at epoch $s = 1, \cdots, S$

$y_{is}$ = number of prevention vehicles allocated to neighborhood $i = 1, \cdots, n$ at epoch $s = 1, \cdots, S$

Again, we define piece-wise constant functions $\overline{x}_i(t) = x_{is}$ and $\overline{y}_i(t) = y_{is}$ for all $\tau_s \leq t \leq \tau_{s+1}$. The system in each neighborhood is governed by the following equations, where $f(\cdot)$ and $g(\cdot)$ denote generic functions reflecting the effects of the prevention and treatment interventions:

$$\frac{dS_i(t)}{dt} = -\alpha^S S_i(t)(I_i(t) + A_i(t) + A_i'(t)) - \beta^S S_i(t) + \rho^S(\mu^S + \psi(\overline{x}(t), \mu^S))A_i(t) \quad (2.42)$$

$$\frac{dA_i(t)}{dt} = \beta^S S_i(t) - \rho^S(\mu^S + \psi(\overline{x}(t), \mu^S))A_i(t) - (1 - \rho^S)(\mu^S + \phi(\overline{y}(t), \mu^S))A_i(t) \quad (2.43)$$

$$\frac{dW_i(t)}{dt} = -\alpha^W W_i(t)(I_i(t) + A_i(t) + A_i'(t)) - \beta^W W_i(t) + \rho^W(\mu^W + \psi(\overline{x}(t), \mu^W))A_i'(t) \quad (2.44)$$

$$\frac{dA_i'(t)}{dt} = \beta^W W_i(t) - \rho^W(\mu^W + \psi(\overline{x}(t), \mu^W))A_i'(t) - (1 - \rho^W)(\mu^W + \phi(\overline{y}(t), \mu^W))A_i'(t) \quad (2.45)$$

$$\frac{dR_i(t)}{dt} = \theta I_i(t) \quad (2.46)$$

$$\frac{dI_i(t)}{dt} = -\left(\frac{dS_i(t)}{dt} + \frac{dW_i(t)}{dt} + \frac{dA_i(t)}{dt} + \frac{dA_i'(t)}{dt} + \frac{dR_i(t)}{dt}\right) \quad (2.47)$$

In these equations $\phi(\cdot, \cdot)$ and $\psi(\cdot, \cdot)$ reflect the effects of the prevention and treatment interventions:

$$\phi(\overline{y}(t), \mu) = \frac{n\mu}{2D_y}(\overline{y}(t) - \frac{D_y}{n}) \quad (2.48)$$

$$\psi(\overline{x}(t), \mu) = \frac{n\mu}{2D_x}(\overline{x}(t) - \frac{D_x}{n}), \quad (2.49)$$

where $D_x$ and $D_y$ represent the maximum number of vehicles available. The problem minimizes the costs of congestion and accidents subject to a budget constraint and the contagion dynamics:

$$\min \quad \sum_{i=1}^{n} \int_0^T \left(c_I I_i(t) + c_A A_i(t) + c_{A'} A_i'(t)\right) dt \tag{2.50}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} (x_{is} + y_{is}) \leq B_s \quad \forall s = 1, \cdots, S \tag{2.51}$$

$$\text{Equations (2.42)–(2.47)} \tag{2.52}$$

$$\text{Initial conditions} \tag{2.53}$$

$$\boldsymbol{S}, \boldsymbol{W}, \boldsymbol{A}, \boldsymbol{A'}, \boldsymbol{I}, \boldsymbol{R} \geq \boldsymbol{0}, \ \boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_+^{n \times T} \tag{2.54}$$

**Discussion.** These problems complement each other. The vaccine allocation problem has the largest state space, reflecting the complexities of the COVID-19 pandemic. The vaccination centers problem features a discrete facility location structure due to linking constraints between a vector switching variable $\boldsymbol{y}$ (facility location) and resource allocation variables $\boldsymbol{x}_{is}$ (vaccine distribution). The other two problems include multi-dimensional resource allocation decisions $\boldsymbol{x}_{is}$. Finally, the traffic congestion problem involves a novel contagion model. Collectively, these problems cover a range of applications of prescriptive contagion analytics, with different optimization structures.

# Chapter 3

# The method: branch-and-price for prescriptive contagion analytics

We propose a branch-and-price algorithm to solve Problem ($\mathcal{P}$), using a set partitioning reformulation (Section 3.1). We circumvent the exponential number of variables via a scalable column generation procedure (Section 3.2). The pricing problem is formulated as a discrete-decision continuous-time dynamic programming model, for which we develop a state-clustering algorithm (Section 3.3). Finally, we embed the column generation procedure into a branch-and-price structure, with a novel and exact tri-partite branching disjunction to circumvent the non-convexity of the problem (Section 3.4). Section 3.5 summarizes the algorithm and establishes its exactness.

## 3.1 Set partitioning formulation

The set partitioning formulation relies on a Dantzig-Wolfe decomposition approach to optimize over plan-based variables in each segment (which characterize resource allocations over the full planning horizon), as opposed to natural variables (which characterize resource allocation in each period). Specifically, we define the set of feasible plans in segment $i = 1, \cdots, n$ as follows:

$$\mathcal{P}_i = \mathcal{F}_{i1} \times \cdots \times \mathcal{F}_{iS},\ \forall i = 1, \cdots, n$$

Specifically, each plan $p \in \mathcal{P}_i$ defines a sequence of resource allocation decisions in segment $i$ over the full planning horizon. Let us also define the following parameters:

$\boldsymbol{\alpha}_{is}^p$ = resource allocation in segment $i = 1, \cdots$ at epoch $s = 1, \cdots, S$ under plan $p \in \mathcal{P}_i$

$C_i^p$ = total cost in segment $i = 1, \cdots$ achieved under resource allocation plan $p \in \mathcal{P}_i$

By design, plan-based variables map into resource allocation variables, and the resulting resource allocation satisfies segment-based constraints. A plan $p \in \mathcal{P}_i$ is equivalent to a sequence of resource allocation decisions $(\boldsymbol{\alpha}_{i1}^p, \cdots, \boldsymbol{\alpha}_{iS}^p) \in \mathcal{F}_{i1} \times \cdots \times \mathcal{F}_{iS}$. The cost is governed by the system's dynamics:

$$C_i^p = \int_0^T g_{it}(\boldsymbol{M}_i(t))dt + h_i(\boldsymbol{M}_i(T)) + \sum_{s=1}^{S} C_{is}(\boldsymbol{x}_{is}) \tag{3.1}$$

$$\text{s.t.} \quad \frac{d\boldsymbol{M}_i(t)}{dt} = f_i(\boldsymbol{M}_i(t), \boldsymbol{\alpha}_{is}^p), \ \forall s = 1, \cdots, S, \ \forall t \in [\tau_s, \tau_{s+1}] \tag{3.2}$$

$$\boldsymbol{M}_i(0) = \boldsymbol{M}_i^0 \tag{3.3}$$

Accordingly, we define the following plan-based variables:

$$z_i^p = \begin{cases} 1 & \text{if plan } p \in \mathcal{P}_i \text{ is selected for segment } i = 1, \cdots, n, \\ 0 & \text{otherwise.} \end{cases}$$

The set partitioning formulation, referred to as ($\mathcal{SP}$), minimizes total costs (Equation (3.4)), while enforcing the coupling constraints (Equation (3.5)) and ensuring that one plan is selected in each segment (Equation (3.6)).

$$(\mathcal{SP}) \quad \min \quad \sum_{i=1}^{n} \sum_{p \in \mathcal{P}_i} C_i^p z_i^p + \boldsymbol{d}^\top \boldsymbol{y} \tag{3.4}$$

$$\text{s.t.} \quad \left( \sum_{i=1}^{n} \sum_{p \in \mathcal{P}_i} \boldsymbol{u}_{sji}^\top \boldsymbol{\alpha}_{is}^p z_i^p \right) + \boldsymbol{v}_{sj}^\top \boldsymbol{y} \geq w_{sj}, \ \forall s = 1, \cdots, S, \ \forall j = 1, \cdots, m_s \tag{3.5}$$

$$\sum_{p \in \mathcal{P}_i} z_i^p = 1 \quad \forall i = 1, \cdots, n \tag{3.6}$$

$$z_i^p \in \{0, 1\} \quad \forall i = 1, \cdots, n, \ \forall p \in \mathcal{P}_i \tag{3.7}$$

$$\boldsymbol{y} \in \mathbb{Z}^q \times \mathbb{R}^r \tag{3.8}$$

**Proposition 1.** *The set partitioning formulation* $(\mathcal{SP})$ *is equivalent to Problem* $(\mathcal{P})$*.*

*Proof.* Let $((\widehat{z}_{ip})_{i=1,\ldots,n;\, p \in \mathcal{P}_i}, \widehat{\boldsymbol{y}})$ be a feasible solution to $(\mathcal{SP})$. We define:

$$\widehat{\boldsymbol{x}}_{is} = \sum_{p \in \mathcal{P}_i} \boldsymbol{\alpha}_{is}^p z_i^p \quad \forall i = 1, \ldots, n, \ \forall s = 1, \ldots, S$$

By definition of the costs in $(\mathcal{SP})$ and in $(\mathcal{P})$, the costs of these two solutions are equal and constraints (2.3) and (2.4) are satisfied. Moreover, because of constraint (3.6) and by definitions of the $\boldsymbol{\alpha}$, we know that, for all $i = 1, ..., n$, there exists a single plan $p \in \mathcal{P}_i$ such that $\widehat{z}_i^p = 1$ and that $\boldsymbol{\alpha}_{is}^p \in \mathcal{F}_{is}$ for all $s = 1, \ldots, S$.Therefore, $\widehat{\boldsymbol{x}}_{is} \in \mathcal{F}_{is}$ for all $i = 1, \ldots, n, \ s = 1, \ldots, S$. Finally, using constraint (3.5), we have:

$$\sum_{i=1}^{n} \boldsymbol{u}_{sji}^\top \widehat{\boldsymbol{x}}_{is} + \boldsymbol{v}_{sj}^\top \widehat{\boldsymbol{y}} = \sum_{i=1}^{n} \sum_{p \in \mathcal{P}_i} \boldsymbol{u}_{sji}^\top \boldsymbol{\alpha}_{is}^p \widehat{z}_i^p + \boldsymbol{v}_{sj}^\top \widehat{\boldsymbol{y}} \geq w_{sj} \quad \forall s = 1, \ldots, S \ j = 1, \ldots, m_s$$

Therefore, this yields a feasible solution to $(\mathcal{P})$ with the exact same cost.

Conversely, let us consider a solution to $(\mathcal{P})$, denoted by $\widehat{\boldsymbol{x}}_{is}$ for $i = 1, \ldots, n; \ s =$

$1, \ldots, S$ and $\widehat{\boldsymbol{y}}$. We define the following plan-based variable:

$$\begin{cases} \widehat{p}_i = [\widehat{\boldsymbol{x}}_{i1}, \ldots, \widehat{\boldsymbol{x}}_{iS}] \quad \forall i = 1, \ldots, n \\ \boldsymbol{\alpha}_{is}^{\widehat{p}_i} = \widehat{\boldsymbol{x}}_{is} \quad \forall i = 1, \ldots, n, \ \forall s = 1, \ldots, S \end{cases}$$

Now we consider the following plan-based solution, for all $i = 1, \cdots, n$:

$$z_i^p = \begin{cases} 1 & \text{if } p = \widehat{p}_i \\ 0 & \text{otherwise} \end{cases}$$

By construction, constraints (3.6) and (3.7) are satisfied. Moreover:

$$\begin{aligned} \sum_{i=1}^{n} \sum_{p \in \mathcal{P}_i} \boldsymbol{u}_{sji}^\top \boldsymbol{\alpha}_{is}^p z_i^p + \boldsymbol{v}_{sj}^\top \widehat{\boldsymbol{y}} &= \sum_{i=1}^{n} \boldsymbol{u}_{sji}^\top \boldsymbol{\alpha}_{is}^{\widehat{p}_i} z_i^{\widehat{p}_i} + \boldsymbol{v}_{sj}^\top \widehat{\boldsymbol{y}} \\ &= \sum_{i=1}^{n} \boldsymbol{u}_{sji}^\top \widehat{\boldsymbol{x}}_{is} + \boldsymbol{v}_{sj}^\top \widehat{\boldsymbol{y}} \geq g_j^s \quad \forall s = 1, \ldots, S, \ \forall j = 1, \ldots, m_s \end{aligned}$$

By definitions of the costs in $(\mathcal{P})$ and $(\mathcal{SP})$, the costs of these two solutions are equal and constraints (3.2) and (3.3) are satisfied. And so we obtained a feasible solution to $(\mathcal{SP})$ with the exact same cost. This proves that Problem $(\mathcal{P})$ and Problem $(\mathcal{SP})$ are equivalent. □

For completeness, we reformulate our four problems:

**Vaccine Allocation** A plan $p$ is a list of $S$ elements $\alpha_{is}^p$, where $\alpha_{is}^p$ represents the number of vaccines allocated to jurisdiction $i$ at epoch $s$ in plan $p$. The set partitioning formulation is given as follows:

$$\min \quad \sum_{i=1}^{n} \sum_{p \in \mathcal{P}_i} C_i^p z_i^p \tag{3.9}$$

$$\text{s.t} \quad \sum_{i=1}^{n} \alpha_{is}^p z_i^p \leq B_s, \quad \forall s = 1, \cdots, S \tag{3.10}$$

$$\sum_{p \in \mathcal{P}_i} z_i^p = 1 \tag{3.11}$$

$$z_i^p \in \{0, 1\} \quad \forall p \in \mathcal{P}_i \tag{3.12}$$

By denoting the dual variables as $\lambda_s$ and $\mu$, we obtain the following pricing problem:

$$\min \quad (c_D D_i(T) + c_H H_i(T) + c_Q Q_i(T) + c_E E_i(T)) - \sum_{s=1}^{S} \lambda_s x_{is} - \phi_i \tag{3.13}$$

$$\text{s.t.} \quad x_{is} \leq B_s \quad \forall s = 1, \cdots, S \tag{3.14}$$

$$\text{Equations (2.11)–(2.18)} \tag{3.15}$$

$$\text{Initial conditions} \tag{3.16}$$

$$\boldsymbol{S}, \boldsymbol{E}, \boldsymbol{I}, \boldsymbol{U}, \boldsymbol{H}, \boldsymbol{Q}, \boldsymbol{D}, \boldsymbol{M}, \boldsymbol{x} \geq \boldsymbol{0} \tag{3.17}$$

**Vaccination centers** A plan $p$ is a list of $S$ elements $\alpha_{is}^p$, where $\alpha_{is}^p$ represents the number of vaccines allocated to region $i$ at epoch $s$ in plan $p$. The set partitioning formulation is given as follows:

$$\min \quad \sum_{i=1}^{n} \sum_{p \in \mathcal{P}_i} C_i^p z_i^p \tag{3.18}$$

$$\text{s.t} \quad \sum_{j=1}^{K} w_j = F \tag{3.19}$$

$$\sum_{i=i}^{n} \sum_{j=1}^{K} v_{ijs} \leq B \quad \forall s = 1, \cdots, S \tag{3.20}$$

$$\sum_{i=1}^{n} v_{ijs} \leq C_j w_j \quad \forall s = 1, \cdots, S, \quad \forall j = 1, \cdots, K \tag{3.21}$$

$$\sum_{s=1}^{S} v_{ijs} \leq P_{ij} w_j \quad \forall i = 1, \cdots, n, \quad \forall j = 1, \cdots, K \tag{3.22}$$

$$\sum_{p \in \mathcal{P}_i} \alpha_{is}^p z_i^p = \sum_{j=1}^{K} v_{i,j,s} \quad \forall s = 1, \cdots, S \quad \forall i = 1, \cdots, n \tag{3.23}$$

$$\sum_{p \in \mathcal{P}_i} z_i^p = 1 \quad \forall i = 1, \cdots, n \tag{3.24}$$

$$z_i^p \in \{0, 1\} \quad \forall p \in \mathcal{P}_i, \ \forall i = 1, \cdots, n \tag{3.25}$$

Note that the traditional budget constrained associated to decision variables $z$ is now ensured by the variable $v$. To ensure consistency between the solutions, we add constraint (3.23) to ensure consistency between the number of vaccines sent to

a region $i$ and the plan-based variables. By denoting the dual variables associated to Constraint (3.23) $\lambda_{is}$ and to Constraint (3.24) $\mu$, we obtain the following pricing problem:

$$\min \quad (c_D D_i(T) + c_H H_i(T) + c_Q Q_i(T) + c_E E_i(T)) - \sum_{s=1}^{S} \lambda_{is} x_{is} - \phi_i \tag{3.26}$$

$$\text{s.t.} \quad x_{is} \leq B_s \quad \forall s = 1, \cdots, S \tag{3.27}$$

$$\text{Equations (2.11)–(2.18)} \tag{3.28}$$

$$\text{Initial conditions} \tag{3.29}$$

$$\boldsymbol{S}, \boldsymbol{E}, \boldsymbol{I}, \boldsymbol{U}, \boldsymbol{H}, \boldsymbol{Q}, \boldsymbol{D}, \boldsymbol{M}, \boldsymbol{x} \geq \boldsymbol{0} \tag{3.30}$$

**Online Content Promotion** A plan $p$ is a list of $S$ tuples $(\alpha_{is}^{xp}, \alpha_{is}^{yp})$ where $\alpha_{is}^{yp}$ is equal to 1 if plan $p$ promotes product $i$ at epoch $s$ and $\alpha_{is}^{xp}$ is the share of the population to which product $i$ is shown at epoch $s$ under plan $p$. The set partitioning formulation is given by:

$$\min \quad \sum_{i=1}^{n} \sum_{p \in \mathcal{P}_i} C_i^p z_i^p \tag{3.31}$$

$$\text{s.t} \quad \sum_{i=1}^{n} \alpha_{is}^{yp} z_i^p \leq B_s, \quad \forall s = 1, \cdots, S \tag{3.32}$$

$$\sum_{i=1}^{n} \alpha_{is}^{xp} z_i^p = 1 B_s, \quad \forall s = 1, \cdots, S \tag{3.33}$$

$$\sum_{p \in \mathcal{P}_i} z_i^p = 1 \tag{3.34}$$

$$z_i^p \in \{0, 1\} \quad \forall p \in \mathcal{P}_i \tag{3.35}$$

By denoting the dual variables as $\lambda_s^y$, $\lambda_s^x$ and $\mu$, we obtain the following pricing

problem:

$$\max \quad B_i(T) - \sum_{s=1}^{S} \lambda_s^y y_{is} - \sum_{s=1}^{S} \lambda_s^x x_{is} - \mu \tag{3.36}$$

$$x_{is} \leq y_{is} \quad \forall s = 1, \cdots, S \tag{3.37}$$

$$\text{Equations (2.33)–(2.34)} \tag{3.38}$$

$$\boldsymbol{S}, \boldsymbol{B}, \boldsymbol{x} \geq \boldsymbol{0}, \ \boldsymbol{y} \in \{0,1\}^S \tag{3.39}$$

**Traffic Congestion** A plan $p$ is a list of $S$ tuples $(\alpha_{is}^{xp}, \alpha_{is}^{yp})$ where $\alpha_{is}^{xp}$ and $\alpha_{is}^{xp}$ are the amounts of resources allocated used for prevention and treatment purposes, respectively, in segment $i$ at epoch $s$ under plan $p$. The set partitioning formulation is given by:

$$\min \quad \sum_{i=1}^{n} \sum_{p \in \mathcal{P}_i} C_i^p z_i^p \tag{3.40}$$

$$\text{s.t} \quad \sum_{i=1}^{n} \alpha_{is}^{xp} z_i^p + \sum_{i=1}^{n} \alpha_{is}^{yp} z_i^p \leq B_s, \quad \forall s = 1, \cdots, S \tag{3.41}$$

$$\sum_{p \in \mathcal{P}_i} z_i^p = 1 \tag{3.42}$$

$$z_i^p \in \{0,1\} \quad \forall p \in \mathcal{P}_i \tag{3.43}$$

By denoting the dual variables as $\lambda_s$ and $\mu$, we obtain the following pricing problem:

$$\min \quad \int_0^T \left(c_I I_i(t) + c_A A_i(t) + c_{A'} A_i'(t)\right) dt - \sum_{s=1}^{S} \lambda_s (x_{is} + y_{is}) - \mu \tag{3.44}$$

$$\text{s.t.} \quad x_{is} + y_{is} \leq B_s \quad \forall s = 1, \cdots, S \tag{3.45}$$

$$\text{Equations (2.42)–(2.47)} \tag{3.46}$$

$$\boldsymbol{S}, \boldsymbol{W}, \boldsymbol{A}, \boldsymbol{A'}, \boldsymbol{I}, \boldsymbol{R} \geq \boldsymbol{0}, \ \boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}_+^{n \times T} \tag{3.47}$$

This reformulation eliminates all structural complexities of Problem $(\mathcal{P})$ by pre-processing the continuous-time dynamics and the non-linear dynamics into the plan-

based variables and the corresponding cost parameters. The ($\mathcal{SP}$) formulation then exhibits a mixed-integer linear optimization structure. However, it comes at the cost of an exponential number of plan-based variables. In an instance with $D$ decisions available in each segment and at each epoch, the number of plans scales in $\mathcal{O}(D^{n\times S})$. It therefore remains intractable to even enumerate the full set of feasible plans, let alone to estimate the cost parameters by running a dynamical model for each one (Equations (3.1)–(3.3)). In response, we propose a column generation approach to generate plans iteratively.

## 3.2 Solving the linear optimization relaxation via column generation

The column generation algorithm decomposes the problem into a restricted master problem (RMP) and a pricing problem (PP). The RMP solves the ($\mathcal{SP}$) relaxation over subsets of plans $\mathcal{P}_i^0 \subseteq \mathcal{P}_i$:

$$(RMP) \quad \min \quad \sum_{i=1}^{n} \sum_{p\in\mathcal{P}_i^0} C_i^p z_i^p + \boldsymbol{d}^\top \boldsymbol{y} \tag{3.48}$$

$$\text{s.t.} \quad \left( \sum_{i=1}^{n} \sum_{p\in\mathcal{P}_i^0} \boldsymbol{u}_{sji}^\top \boldsymbol{\alpha}_{is}^p z_i^p \right) + \boldsymbol{v}_{sj}^\top \boldsymbol{y} \geq w_{sj}, \ \forall s = 1, \cdots, S, \ \forall j = 1, \cdots, m_s \tag{3.49}$$

$$\sum_{p\in\mathcal{P}_i^0} z_i^p = 1 \quad \forall i = 1, \cdots, n \tag{3.50}$$

$$z_i^p \geq 0 \quad \forall i = 1, \cdots, n, \ \forall p \in \mathcal{P}_i^0 \tag{3.51}$$

$$\boldsymbol{y} \in \mathbb{R}^{q+r} \tag{3.52}$$

Let $\lambda_{sj} \in \mathbb{R}_+$ denote the dual variables associated with Constraints (3.49) and $\mu_i \in \mathbb{R}$ the dual variable associated with Constraints (3.50). The pricing problem generates new plan-based variables or a certificate of optimality, by seeking the plan

in $\mathcal{P}_i, i = 1, \cdots, n$ with the minimal reduced cost:

$$(PP_i) \quad \min \quad \left( \int_0^T g_{it}(\boldsymbol{M}_i(t))dt + h_i(\boldsymbol{M}_i(T)) + \sum_{s=1}^{S} C_{is}(\boldsymbol{x}_{is}) - \sum_{s=1}^{S} \sum_{j=1}^{m_s} \lambda_{sj} \boldsymbol{u}_{sji}^{\top} \boldsymbol{x}_{is} - \mu_i \right) \tag{3.53}$$

$$\text{s.t.} \quad \boldsymbol{x}_{is} \in \mathcal{F}_{is}, \ \forall s = 1, \cdots, S \tag{3.54}$$

$$\frac{d\boldsymbol{M}_i(t)}{dt} = f_i(\boldsymbol{M}_i(t), \boldsymbol{x}_{is}), \ \forall s = 1, \cdots, S, \ \forall t \in [\tau_s, \tau_{s+1}] \tag{3.55}$$

$$\boldsymbol{M}_i(0) = \boldsymbol{M}_i^0. \tag{3.56}$$

The pricing problem exhibits a continuous-time non-linear optimization structure, due to the system dynamics in each segment (as well as, possibly, to complicated segment-specific constraints). The fundamental difference with the original problem $(\mathcal{P})$, however, is that the pricing problem is decomposable across segments. We exploit this structure to formulate it via dynamic programming. By definition, the state variable $\boldsymbol{M}_i(t)$ encapsulates all of the system's history that is necessary to compute the cost function, the constraints in $\mathcal{F}_{is}$, and the transition function $f_i(\cdot, \cdot)$, so the system follows Markovian dynamics. At each epoch $s = 1, \cdots, S$, the state variable is denoted by $\boldsymbol{N}_s^i$ and equal to $\boldsymbol{M}_i(\tau_s)$. The decision is $\boldsymbol{x}_{is} \in \mathcal{F}_{is}$, and the transition function is governed by the dynamical system between times $\tau_s$ and $\tau_{s+1}$, with an initial condition at time $\tau_s$ determined by the state variable. The problem involves a per-period cost characterizing the continuous-time cost $g_{it}(\cdot)$ accrued between $\tau_s$ and $\tau_{s+1}$, the cost of resource allocation $C_{is}(\cdot)$, and the dual price associated with the polyhedral constraints. In addition, the problem involves a terminal cost characterizing the end-state cost $h_i(\cdot)$ and the dual price of the set partitioning constraints. The Bellman equation is given as follows, using $J_s(\cdot)$ to define the cost-to-go function at epoch

$s = 1, \cdots, S$:

$$J_s\left(\boldsymbol{N}_s^i\right) = \min_{\boldsymbol{x}_{is} \in \mathcal{F}_{is}} \left\{ \int_{\tau_s}^{\tau_{s+1}} g_{it}(\boldsymbol{M}_i(t))dt + C_{is}(\boldsymbol{x}_{is}) - \sum_{j=1}^{m_s} \lambda_{sj} \boldsymbol{u}_{sji}^\top \boldsymbol{x}_{is} + J_{s+1}\left(\boldsymbol{N}_{s+1}^i\right) \right\} \tag{3.57}$$

where $\boldsymbol{M}_i(\tau_s) \leftarrow \boldsymbol{N}_s^i$; $\dfrac{d\boldsymbol{M}_i(t)}{dt} = f_i(\boldsymbol{M}_i(t), \boldsymbol{x}_{is}),\ \forall t \in [\tau_s, \tau_{s+1}]$; and $\boldsymbol{N}_{s+1}^i \leftarrow \boldsymbol{M}_i(\tau_{s+1})$

$$J_{S+1}\left(\boldsymbol{N}_{S+1}^i\right) = h\left(\boldsymbol{N}_{S+1}^i\right) - \mu_i \tag{3.58}$$

The column generation procedure solves the linear relaxation of the $\mathcal{SP}$ formulation iteratively. Starting with initial sets $\mathcal{P}_i^0$ of plan-based variables, the RMP provides a feasible primal solution at each iteration, along with the dual variables $\lambda_{js}$ and $\mu_i$. We then solve $(PP_i)$ for each segment $i = 1, \cdots, n$; if its optimal value is negative, we expand the sets $\mathcal{P}_i^0$ with the new solution and go back to the RMP. Otherwise, the incumbent RMP solution is optimal for the $(\mathcal{SP})$ relaxation.

This column generation procedure effectively separates the two complexities of the problem: the RMP handles the combinatorial optimization component via its mixed-integer optimization structure (Equations (3.48)–(3.51)), and the PP handles the system dynamics via a finite-horizon dynamic program in each segment (Equations (3.57)–(3.58)). Due to the continuous-state dynamics of the system, however, the pricing problem exhibits a continuous-state dynamic programming structure—a notoriously challenging class of problems. Given the need to solve it repeatedly throughout column generation, we therefore devise an acceleration algorithm based on state space clustering.

## 3.3 Solving the subproblem via a state-clustering dynamic programming algorithm

Since the pricing problem is separable across segments $i = 1, \cdots, n$, we ignore the index $i$ in this section; for instance, $\boldsymbol{N}_s$ refers to the state, and $\boldsymbol{x}_s \in \mathcal{F}_s$ to the decision with $D_s = |\mathcal{F}_s|$.

### 3.3.1 Exact algorithm

Any finite-horizon discrete-decision continuous-state dynamic programming model can be solved via forward enumeration and backward induction:

1. *Forward enumeration*: From the initial state $\boldsymbol{M}^0$, evaluate all possible decisions $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_S$, and store all possible states at each epoch $s = 1, \cdots, S$ in a set $\mathcal{N}_s$.

2. *Backward induction*: Starting from $s = S$, derive the optimal policy $\boldsymbol{\pi}_s^*(\cdot)$ at epoch $s = S, \cdots, 1$ and update the cost-to-go function $J_s(\cdot)$, as follows:

$$\boldsymbol{\pi}_s^*\left(\boldsymbol{N}_s\right) = \underset{\boldsymbol{x}_s \in \mathcal{F}_s}{\arg\min} \left\{ \int_{\tau_s}^{\tau_{s+1}} g_{it}(\boldsymbol{M}_i(t))dt + C_s(\boldsymbol{x}_s) - \sum_{j=1}^{m_s} \lambda_{sj} \boldsymbol{u}_{sji}^\top \boldsymbol{x}_s + J_{s+1}\left(\boldsymbol{N}_{s+1}\right) \right\} \tag{3.59}$$

$$J_s\left(\boldsymbol{N}_s\right) = \left\{ \int_{\tau_s}^{\tau_{s+1}} g_{it}(\boldsymbol{M}_i(t))dt + C_s\left(\boldsymbol{\pi}_s^*\left(\boldsymbol{N}_s\right)\right) - \sum_{j=1}^{m_s} \lambda_{sj} \boldsymbol{u}_{sji}^\top \boldsymbol{\pi}_s^*\left(\boldsymbol{N}_s\right) + J_{s+1}\left(\boldsymbol{N}_{s+1}\right) \right\} \tag{3.60}$$

where $\boldsymbol{M}_i(\tau_s) \leftarrow \boldsymbol{N}_s$; $\dfrac{d\boldsymbol{M}_i(t)}{dt} = f_i(\boldsymbol{M}_i(t), \boldsymbol{\pi}_s^*\left(\boldsymbol{N}_s\right)),\ \forall t \in [\tau_s, \tau_{s+1}]$

and $\boldsymbol{N}_{s+1} \leftarrow \boldsymbol{M}_i(\tau_{s+1})$

This approach is detailed in Algorithm 1.

Per the Bellman principle of optimality, the optimal path is optimal at every decision point. We obtain the following result.

**Proposition 2.** *Algorithm 1 returns a feasible and optimal solution to Equations* (3.57)–(3.58).

However, Algorithm 1 is highly expensive due to the exhaustive state enumeration in both the forward enumeration and backward induction loops. Complexity grows in $\mathcal{O}\left(\prod_{s=1}^{S} D_s\right)$, which can quickly become very large as the planning horizon and the decision space increase. Even for a simple eight-period problem with two resource allocation decisions that can each take four values, $\prod_{s=1}^{S} D_s = 16^8$ is on the order of 1 billion, which severely hinders exhaustive enumeration.

**Algorithm 1** Dynamic programming algorithm for pricing problem

**procedure** StateSpaceCreation($\mathcal{N}_0$, $\boldsymbol{\lambda}_s$, $\mu$)
  **for** $s \leftarrow 0$ to $S$ **do** ▷ Forward enumeration
    $\mathcal{N}_{s+1} \leftarrow \emptyset$
    **for** $\boldsymbol{N}_s \in \mathcal{N}_s,\ \boldsymbol{x}_s \in \mathcal{F}_s$ **do**
      Transition: $\boldsymbol{M}(\tau_s) \leftarrow \boldsymbol{N}_s$; $\frac{d\boldsymbol{M}(t)}{dt} = f(\boldsymbol{M}(t), \boldsymbol{x}_s),\ \forall t \in [\tau_s, \tau_{s+1}]$; $\mathcal{N}_{s+1} \leftarrow \mathcal{N}_{s+1} \cup \boldsymbol{M}(\tau_{s+1})$
      Cost: $c_s(\boldsymbol{N}_s, \boldsymbol{x}_s) \leftarrow \int_{\tau_s}^{\tau_{s+1}} g(\boldsymbol{M}_i(t))dt + C_s(\boldsymbol{x}_s) - \sum_{j=1}^{m_s} \lambda_{sj} \boldsymbol{f}_j^s \boldsymbol{x}_s$
    **end for**
  **end for**
  **return** state space $\mathcal{N}_1, \cdots, \mathcal{N}_S$, cost $c_s(\boldsymbol{N}_s, \boldsymbol{x}_s)$ for all $\boldsymbol{N}_s \in \mathcal{N}_s,\ \boldsymbol{x}_s \in \mathcal{F}_s,\ s \in \{1, \cdots, S\}$
**end procedure**
**procedure** DynamicProgram($\mathcal{N}_1, \cdots, \mathcal{N}_S,\ c_1(\cdot,\cdot), \cdots, c_S(\cdot,\cdot)$)
  Initialization: $c^{opt}(\boldsymbol{N}_s) = +\infty;\ \boldsymbol{\pi}_s^{opt}(\boldsymbol{N}_s) \leftarrow \emptyset$
  **for** $s \leftarrow S$ to 0 **do** ▷ Backward induction
    **for** $\boldsymbol{N}_s \in \mathcal{N}_s$ **do**
      $c^{temp} \leftarrow +\infty$; $\boldsymbol{x}^{temp} \leftarrow \emptyset$
      **for** $\boldsymbol{x}_s \in \mathcal{F}_s$ **do**
        If $c(\boldsymbol{N}_s, \boldsymbol{x}_s) < c^{temp}$, update $c^{temp} \leftarrow c(N_s, \boldsymbol{x}_s)$; $\boldsymbol{x}^{temp} \leftarrow \boldsymbol{x}_s$
      **end for**
      $c^{opt}(\boldsymbol{N}_s) = c^{temp};\ \boldsymbol{\pi}_s^{opt}(\boldsymbol{N}_s) \leftarrow \boldsymbol{x}^{temp}$
    **end for**
  **end for**
  Initialization: $\boldsymbol{N}_0^{opt} \leftarrow \boldsymbol{M}^0$, $OPT \leftarrow 0$
  **for** $s \leftarrow 0$ to $S$ **do** ▷ Forward evaluation
    Find optimal decision $\boldsymbol{x}_s^{opt} \leftarrow \boldsymbol{\pi}_s^{opt}(\boldsymbol{N}_s^{opt})$ and update cost $OBJ \leftarrow OBJ + c(\boldsymbol{N}_s^{opt}, \boldsymbol{x}_s^{opt})$
    Find next state: $\boldsymbol{M}(\tau_s) \leftarrow \boldsymbol{N}_s^{opt}$; $\frac{d\boldsymbol{M}(t)}{dt} = f(\boldsymbol{M}(t), \boldsymbol{x}_s),\ \forall t \in [\tau_s, \tau_{s+1}]$; $\boldsymbol{N}_{s+1}^{opt} \leftarrow \boldsymbol{M}(\tau_{s+1})$
  **end for**
  **return** cost $OBJ$, decisions $\boldsymbol{x}_s\ \forall s \in \{0, \cdots, S\}$
**end procedure**

As noted in Section 1.3, most approximate dynamic programming and reinforcement learning methods make use of cost-to-go approximations and policy approximations that depend on the cost function. In our problem, however, the cost function changes from one column generation iteration to the next. Thus, traditional approaches would need to be re-started at every iteration, resulting in significant inefficiencies. We therefore propose a state-clustering algorithm that does not depend on the cost function, but instead exploits the natural concentration of states in dynamical systems.

### 3.3.2 A state-clustering acceleration.

In dynamical systems, many states in $\mathcal{N}_s$ are close to each other, especially as $S$ increases. In vaccine allocation, for example, there are only mild differences between a no-vaccine baseline and an allocation of just a few vaccines; similarly, two allocations that merely swap decisions between epochs $s$ and $s+1$ may yield similar outcomes down the line. Accordingly, we design a state-clustering approach independent of the cost function.

Specifically, at each epoch $s = 1, \cdots, S$, we evaluate all decisions $\boldsymbol{x}_s \in \mathcal{F}_s$ from all states $\boldsymbol{N}_s \in \mathcal{N}_s$. We group the resulting $|\mathcal{N}_s| \times |\mathcal{F}_s|$ states into $\Gamma_{s+1}$ clusters, and treat the cluster centroids as the representative states at epoch $s+1$, and apply the backward induction algorithm. By design, this procedure reduces the state space significantly as long as the number of clusters is small (i.e., $\Gamma_s \ll |\mathcal{N}_s| \times |\mathcal{F}_s|$). Obviously, the resulting dynamic program is an approximation of the full one; yet, the loss of optimality can be small if the full state space does admit a clustered structure.

The next question is which clustering algorithm to apply through this procedure. As it turns out, many standard methods are not suitable in our context:

- $k$-means clustering relies on a pre-determined number of clusters $\Gamma_s$, which is difficult to set for complex dynamical systems. Instead, our algorithm creates clusters dynamically.

- Global similarity clustering algorithms often have quadratic or worse complex-

ity, due to the computation of the entire distance matrix. In our setup, given that $|\mathcal{N}_s| \times |\mathcal{F}_s|$ may already be large, a complexity of $\mathcal{O}\left(|\mathcal{N}_s|^2 \times |\mathcal{F}_s|^2\right)$ is likely to lead to intractability.

– Many clustering algorithms cannot bound the cluster diameter. This is problematic in our non-linear dynamical systems, as large deviations between a cluster centroid and the "actual" state could lead to significant drift through the differential equations, hence suboptimal decisions.

We therefore propose a linear-time clustering algorithm with guarantees on cluster diameter in an $\ell_\infty$-space. For each cluster $\gamma$, we record: (i) the number of data points $\eta(\gamma)$, (ii) the sum of all states $\boldsymbol{N}^\Sigma(\gamma)$, (iii) the element-wise minimum state $\underline{\boldsymbol{N}}(\gamma)$, (iv) the element-wise maximum state $\overline{\boldsymbol{N}}(\gamma)$, (v) the set of decisions that lead to the cluster $\mathcal{X}(\gamma)$, and (vi) the total cost $c(\gamma)$. The first and second elements are used to define the centroid of each cluster; the third and fourth elements are used to bound the cluster diameter in our dynamic clustering procedure; and the last two elements define state transitions and cost functions in the clustered state space. Specifically, for a diameter $\varepsilon$, we cluster $Q$ decision states $(\boldsymbol{N}^1, \boldsymbol{x}^1), \cdots, (\boldsymbol{N}^Q, \boldsymbol{x}^Q)$ at epoch $s$ as follows:

1. We assign the first state $\boldsymbol{N}^1$ to a new cluster $\gamma_1$. We initialize:

$$\begin{aligned} \boldsymbol{N}^\Sigma(\gamma_1) &\leftarrow \boldsymbol{N}^1 \qquad & \underline{\boldsymbol{N}}(\gamma_1) &\leftarrow \boldsymbol{N}^1 \qquad & \overline{\boldsymbol{N}}(\gamma_1) &\leftarrow \boldsymbol{N}^1 \\ \eta(\gamma_1) &\leftarrow 1 \qquad & \mathcal{X}(\gamma_1) &\leftarrow \{\boldsymbol{x}^1\} \qquad & c^\Sigma(\gamma_1) &\leftarrow \int_{\tau_s}^{\tau_{s+1}} g_{it}(\boldsymbol{M}^1(t))dt, \end{aligned}$$

2. For any state $\boldsymbol{N}^q$, we first test if $\boldsymbol{N}^q$ should belong to an existing clusters $\gamma_1, \cdots, \gamma_k$. We do so by calculating the maximal $\ell_\infty$ distance to the minimum and maximum states:

$$\delta(\boldsymbol{N}^q, \gamma_l) = \min_{l=1,\cdots,k} \max(\|\boldsymbol{N}^q - \underline{\boldsymbol{N}}(\gamma_l)\|_\infty, \|\boldsymbol{N}^q - \overline{\boldsymbol{N}}(\gamma_l)\|_\infty)$$

Let $l^*$ denote the index that attains the minimum. If $\delta(\boldsymbol{N}^q, \gamma_{l^*}) \leq \varepsilon$, $\boldsymbol{N}^q$ is assigned to $\gamma_{l^*}$:

$$\boldsymbol{N}^{\Sigma}(\gamma_{l^*}) \leftarrow \boldsymbol{N}^{\Sigma}(\gamma_{l^*}) + \boldsymbol{N}^q \quad \underline{\boldsymbol{N}}(\gamma_{l^*}) \leftarrow \min(\underline{\boldsymbol{N}}(\gamma_{l^*}), \boldsymbol{N}^q) \quad \overline{\boldsymbol{N}}(\gamma_{l^*}) \leftarrow \max(\overline{\boldsymbol{N}}(\gamma_{l^*}), \boldsymbol{N}^q)$$
$$\eta(\gamma_{l^*}) \leftarrow \eta(\gamma_{l^*}) + 1 \qquad \mathcal{X}(\gamma_{l^*}) \leftarrow \mathcal{X}(\gamma_{l^*}) \cup \boldsymbol{x}^q \qquad c^{\Sigma}(\gamma_{l^*}) \leftarrow c^{\Sigma}(\gamma_{l^*}) + \int_{\tau_s}^{\tau_{s+1}} g_{it}(\boldsymbol{M}^q(t))dt$$

where $\boldsymbol{M}^q(\tau_s) \leftarrow \boldsymbol{N}^q$ and $\frac{d\boldsymbol{M}^q(t)}{dt} = f_i(\boldsymbol{M}^q(t), \boldsymbol{x}^q)$. Otherwise, if $\delta(\boldsymbol{N}^q, \gamma_{l^*}) > \varepsilon$, then $\boldsymbol{N}^q$ is assigned to a new cluster $\gamma_{k+1}$, initialized as follows:

$$\boldsymbol{N}^{\Sigma}(\gamma_{k+1}) \leftarrow \boldsymbol{N}^q \quad \underline{\boldsymbol{N}}(\gamma_{k+1}) \leftarrow \boldsymbol{N}^q \quad \overline{\boldsymbol{N}}(\gamma_{k+1}) \leftarrow \boldsymbol{N}^q$$
$$\eta(\gamma_{k+1}) \leftarrow 1 \quad \mathcal{X}(\gamma_{k+1}) \leftarrow \{\boldsymbol{x}^q\} \quad c^{\Sigma}(\gamma_{k+1}) \leftarrow \int_{\tau_s}^{\tau_{s+1}} g_{it}(\boldsymbol{M}^q(t))dt.$$

3. Upon termination, we retrieve the average state $\boldsymbol{N}(\gamma_l) = \boldsymbol{N}^{\Sigma}(\gamma_l)/\eta(\gamma_l)$, the decisions $\mathcal{X}(\gamma_l)$ leading to the cluster, and the average cost of transitioning to the cluster $c(\gamma_l) = c^{\Sigma}(\gamma_{l^*})/\eta(\gamma_l)$.

Pseudo-code is given in Algorithm 2.

The algorithm makes a single pass through all the states, thus terminating in linear time (assuming that the number of clusters is much smaller than the number of states). Still, the algorithm controls the maximum pair-wise distance within each cluster, guaranteeing an $\ell_\infty$ diameter of $\varepsilon$. This is formalized in Proposition 3.

**Proposition 3.** *For any states $\boldsymbol{N}^1, \cdots, \boldsymbol{N}^Q \in \mathbb{R}^r$, the clusters $\gamma_1, \cdots, \gamma_k$ satisfy:*

$$\|\overline{\boldsymbol{N}}(\gamma_l) - \underline{\boldsymbol{N}}(\gamma_l)\|_\infty \leq \varepsilon, \quad \forall l \in \{1, \cdots, k\}. \tag{3.61}$$

*Proof.* Let $\gamma_l^{(q)}$ denote the $l^{\text{th}}$ cluster after $q$ states have been assigned to it. Let us denote by $\Phi_l$ the size of the $l^{\text{th}}$ cluster upon termination of the algorithm. The ending cluster is $\gamma_l^{\Phi_l}$.

We proceed by contradiction. Assume that there exists a cluster $l$ such that:

$$\|\overline{\boldsymbol{N}}(\gamma_l^{(\Phi_l)}) - \underline{\boldsymbol{N}}(\gamma_l^{(\Phi_l)})\|_\infty > \varepsilon$$

**Algorithm 2** State-clustering acceleration of dynamic programming algorithm for pricing problem

**procedure** ClusteredDP($\mathcal{N}_0$, $\lambda_i(s)$, $\mu$, $\varepsilon$)
  **for** $s \leftarrow 0$ to $S$ **do** ▷ Forward pass
    Initialization: $\mathcal{N}_{s+1} \leftarrow \emptyset$, $k \leftarrow 0$
    **for** $\boldsymbol{N}_s \in \mathcal{N}_s,\ \boldsymbol{x}_s \in \mathcal{F}_s$ **do**
      Next state: $\boldsymbol{M}(\tau_s) \leftarrow \boldsymbol{N}_s$; $\frac{d\boldsymbol{M}(t)}{dt} = f(\boldsymbol{M}(t), \boldsymbol{x}_s),\ \forall t \in [\tau_s, \tau_{s+1}]$; $\boldsymbol{N}_{s+1} \leftarrow \boldsymbol{M}(\tau_{s+1})$
      $l^* \leftarrow \arg\min_{l \in \{0,1,\cdots,k\}} \max(\|\boldsymbol{N}_{s+1} - \underline{\boldsymbol{N}}(\gamma_l)\|_\infty, \|\boldsymbol{N}_{s+1} - \overline{\boldsymbol{N}}(\gamma_l)\|_\infty)$
      **if** $\max(\|\boldsymbol{N}_{s+1} - \underline{\boldsymbol{N}}(\gamma_{l^*})\|_\infty, \|\boldsymbol{N}_{s+1} - \overline{\boldsymbol{N}}(\gamma_{l^*})\|_\infty)$ **then**
        Cluster update (1): $\boldsymbol{N}^\Sigma(\gamma_{l^*}) \leftarrow \boldsymbol{N}^\Sigma(\gamma_{l^*}) + \boldsymbol{N}_{s+1}$ , $\eta(\gamma_{l^*}) \leftarrow \eta(\gamma_{l^*}) + 1$, $\mathcal{X}(\gamma_{l^*}) \leftarrow \mathcal{X}(\gamma_{l^*}) \cup \boldsymbol{x}_s$
        Cluster update (2): $\underline{\boldsymbol{N}}(\gamma_{l^*}) \leftarrow \min(\underline{\boldsymbol{N}}(\gamma_{l^*}), \boldsymbol{N}_{s+1})$, $\overline{\boldsymbol{N}}(\gamma_{l^*}) \leftarrow \max(\overline{\boldsymbol{N}}(\gamma_{l^*}), \boldsymbol{N}_{s+1})$
        Cluster update (3): $c^\Sigma(\gamma_{l^*}) \leftarrow c^\Sigma(\gamma_{l^*}) + \int_{\tau_s}^{\tau_{s+1}} g(\boldsymbol{M}_i(t))dt$
      **else**
        New cluster (1): $k \leftarrow k+1$, $\boldsymbol{N}^\Sigma(\gamma_k) \leftarrow N_{s+1}$, $\underline{\boldsymbol{N}}(\gamma_k) \leftarrow N_{s+1}$
        New cluster (2): $\overline{\boldsymbol{N}}(\gamma_k) \leftarrow N_{s+1}$, $\eta(\gamma_k) \leftarrow 1, \mathcal{X}(\gamma_k) \leftarrow \boldsymbol{x}_s$
      **end if**
    **end for**
    **for** $l \leftarrow 0$ to $k$ **do**
      **for** $\boldsymbol{x}_s \in \mathcal{X}(\gamma_l)$ **do**
        $\mathcal{N}_{s+1} \leftarrow \mathcal{N}_{s+1} \cup \boldsymbol{N}^\Sigma(\gamma_l)/\eta(\gamma_l)$
        $c_s(\boldsymbol{N}_s, \boldsymbol{x}_s) \leftarrow c^\Sigma(\gamma_l)/\eta(\gamma_l) + \mathbf{1}\{s = S\}h(\boldsymbol{N}^\Sigma(\gamma_l)/\eta(\gamma_l)) + C_s(\boldsymbol{x}_s) - \sum_{j=1}^{m_s} \lambda_{sj} \boldsymbol{f}_j^s \boldsymbol{x}_s$
      **end for**
    **end for**
  **end for**
**end procedure**
Run DynamicProgram($\mathcal{N}_1, \cdots, \mathcal{N}_S,\ c_1(\cdot,\cdot), \cdots, c_S(\cdot,\cdot)$)

Note that, by construction (Step 2.), the quantity

$$\|\overline{\boldsymbol{N}}(\gamma_l^{(i)}) - \underline{\boldsymbol{N}}(\gamma_l^{(i)})\|_\infty$$

is non decreasing in $i$. There must exist $i \in \{1, \cdots, \Phi_l\}$ such that:

$$\begin{aligned}\|\overline{\boldsymbol{N}}(\gamma_l^{(i)}) - \underline{\boldsymbol{N}}(\gamma_l^{(i)})\|_\infty &\leq \varepsilon \\ \|\overline{\boldsymbol{N}}(\gamma_l^{(i+1)}) - \underline{\boldsymbol{N}}(\gamma_l^{(i+1)})\|_\infty &> \varepsilon\end{aligned}$$

By definition of the $\ell_\infty$ norm, we know that there exists coordinate $j = 1, \cdots, r$ such that:

$$\begin{aligned}|(\overline{\boldsymbol{N}}(\gamma_l^{(i)}))_j - (\underline{\boldsymbol{N}}(\gamma_l^{(i)}))_j| &\leq \varepsilon \\ |(\overline{\boldsymbol{N}}(\gamma_l^{(i+1)}))_j - (\underline{\boldsymbol{N}}(\gamma_l^{(i+1)}))_j| &> \varepsilon.\end{aligned}$$

Denote the $i+1^{\text{th}}$ sample that got clustered into $\gamma_l$ as $\boldsymbol{N}^*$. By construction (Step 2.), at most one of $(\overline{\boldsymbol{N}}(\gamma_l^{(i)}))_j$ and $(\underline{\boldsymbol{N}}(\gamma_l^{(i)}))_j$ can change after $\boldsymbol{N}^*$ gets added to the cluster. Without loss of generality, assume that $\boldsymbol{N}^*$ changed the minimum state, so we have:

$$(\boldsymbol{N}^*)_j = (\underline{\boldsymbol{N}}(\gamma_l^{(i+1)}))_j \neq (\underline{\boldsymbol{N}}(\gamma_l^{(i)}))_j \qquad (\overline{\boldsymbol{N}}(\gamma_l^{(i+1)}))_j = (\overline{\boldsymbol{N}}(\gamma_l^{(i)}))_j$$

However, this implies:

$$\begin{aligned}\max(\|\boldsymbol{N}^* - \underline{\boldsymbol{N}}(\gamma_l^{(i)})\|_\infty, \|\boldsymbol{N}^* - \overline{\boldsymbol{N}}(\gamma_l^{(i)})\|_\infty) &\geq \|\boldsymbol{N}^* - \overline{\boldsymbol{N}}(\gamma_l^{(i)})\|_\infty \\ &\geq |(\boldsymbol{N}^*)_j - (\overline{\boldsymbol{N}}(\gamma_l^{(i)}))_j| \\ &= |(\underline{\boldsymbol{N}}(\gamma_l^{(i+1)}))_j - (\overline{\boldsymbol{N}}(\gamma_l^{(i+1)}))_j| \\ &> \varepsilon.\end{aligned}$$

This implies that $\boldsymbol{N}^*$ would not be accepted in the cluster (Step 3.), a contradiction.

Therefore, we have:

$$\|\overline{\boldsymbol{N}}(\gamma_l^{(\Phi_l)}) - \underline{\boldsymbol{N}}(\gamma_l^{(\Phi_l)})\|_\infty \leq \varepsilon \ \ \forall l$$

This completes the proof. □

This guarantee allows us to bound the error of state approximation. Specifically, Proposition 4 shows that our global approximation error depends linearly on the cluster diameter $\varepsilon$ and exponentially in $t$. The latter dependence on $t$ is typical in dynamical systems (see e.g. 68). Our results in Section 4.1 show that, in practice, our state-clustering algorithm induces moderate error in small-scale instances that can be fully enumerated—and scales to much larger instances.

**Proposition 4.** *Assume that the transition function $f_i(., \boldsymbol{x}_{is})$ is $L_{is}$-Lipschitz in the state variables in the $l_\infty$ metric, for every $i \in \{1, \cdots, n\}$ and $s \in \{1, \cdots, S\}$. For every segment $i$ and epoch $s$, denote the true states (from Algorithm 1) by $\boldsymbol{N}_*^{is1} \cdots, \boldsymbol{N}_*^{isP} \in \mathcal{N}_*^{is}$, and the clustered states (from Algorithm 2) by $\boldsymbol{N}^{is1}, \cdots, \boldsymbol{N}^{isQ} \in \mathcal{N}^{is}$. Then, for all $\boldsymbol{N}^{isq} \in \mathcal{N}^{is}$, there exists $\boldsymbol{N}_*^{isp} \in \mathcal{N}_*^{is}$ such that:*

$$\|\boldsymbol{N}_*^{isp} - \boldsymbol{N}^{isq}\|_\infty \leq \begin{cases} 0 & \text{if } s = 1 \\ \varepsilon \sum_{\sigma=3}^{s+1} \exp\left(\sum_{\nu=\sigma}^{s} L_{i(\nu-1)}(\tau_\nu - \tau_{\nu-1})\right), & \forall s \geq 2 \end{cases} \tag{3.62}$$

*Proof.* Because the clustering process is done separately for each segment $i = 1, \cdots, n$, we overlook the $i$ subscript for conciseness. We first note that by the definition of Algorithm 2, in Epoch $s = 1$ (the initial state), there is no difference between the clustered and true states, and by Proposition 4, the error in Epoch $s = 2$ is $\leq \epsilon$.

Therefore we would focus on proving the result for Epoch $s = k + 1 \geq 3$ assuming the result is true for $s = k$. At Epoch $s = k$, we have that for all $\boldsymbol{N}^{kq} \in \mathcal{N}^k$, there exists $\boldsymbol{N}^{kp} \in \mathcal{N}_*^k$ such that:

$$\min_{p \in \{1, \cdots, P\}} \|\boldsymbol{N}_*^{kp} - \boldsymbol{N}^{kq}\|_\infty \leq \epsilon \sum_{\sigma=3}^{k+1} \exp\left(\sum_{\nu=\sigma}^{k} L_{i(\nu-1)}(\tau_\nu - \tau_{\nu-1})\right) \tag{3.63}$$

Then we introduce Gronwall's inequality: Let $L$ be a constant and $\boldsymbol{u}(t) \in \mathbb{R}^n$ be a real-valued $n$-dimensional continuous function defined on $[a, b]$. If $u$ is differentiable

on the interior of $(a, b)$ and satisfies for all $t$:

$$\|\boldsymbol{u}'(t)\|_\infty \leq L\|\boldsymbol{u}(t)\|_\infty.$$

Then, we have that for all $t \leq b$:

$$\|\boldsymbol{u}(t)\|_\infty \leq \|\boldsymbol{u}(a)\|_\infty e^{L(t-a)}$$

Now for every $\boldsymbol{N}^{kq} \in \mathcal{N}^k$, denote $\boldsymbol{N}_*^{kp}$ as the corresponding true state that satisfies the inequality in Equation 3.63. Then, we have that for $\boldsymbol{M}_1(t), \boldsymbol{M}_2(t)$ defined by:

$$\begin{aligned} \boldsymbol{M}_1(\tau_{k+1}) &= \boldsymbol{N}_*^{kp} & \frac{d\boldsymbol{M}_1(t)}{dt} &= f_i(\boldsymbol{M}_1(t), \boldsymbol{x}_{ik}) \\ \boldsymbol{M}_2(\tau_{k+1}) &= \boldsymbol{N}^{kq} & \frac{d\boldsymbol{M}_2(t)}{dt} &= f_i(\boldsymbol{M}_2(t), \boldsymbol{x}_{ik}) \end{aligned}$$

we have, by the assumption in Proposition 4:

$$\|f_i(\boldsymbol{M}_1(t), \boldsymbol{x}_{i(k+1)} - f_i(\boldsymbol{M}_2(t), \boldsymbol{x}_{ik})\|_\infty \leq L_{ik}\|\boldsymbol{M}_1(t) - \boldsymbol{M}_2(t)\|_\infty \ \forall t \in [\tau_k, \tau_{k+1}]$$

Then we apply Gronwall's Lemma to $\boldsymbol{M}_1(t) - \boldsymbol{M}_2(t)$ with $a = \tau_k$ and $b = \tau_{k+1}$ to get that:

$$\begin{aligned} \|\boldsymbol{M}_1(\tau_{k+1}) - \boldsymbol{M}_2(\tau_{k+1})\| &\leq \exp(L_{ik}(\tau_{k+1} - \tau_k))\|\boldsymbol{M}_1(\tau_k) - \boldsymbol{M}_2(\tau_k)\| \\ &= \epsilon \sum_{i=3}^{k+1} \exp(\sum_{\nu=i}^{k+1} L_{i(\nu-1)}(\tau_\nu - \tau_{\nu-1})) \end{aligned}$$

Now note that $\boldsymbol{M}_1(\tau_{k+1}) \in \mathcal{N}_*^{(k+1)}$ by definition of the full dynamic programming algorithm. On the other hand, we have, by Proposition 2, we have that for some $\boldsymbol{N}^{(k+1)q} \in \mathcal{N}^{(k+1)}$, we have:

$$\|\boldsymbol{M}_2(\tau_{k+1}) - \boldsymbol{N}^{(k+1)q}\|_\infty \leq \epsilon$$

Therefore, we have that for some $p, q$:

$$\|\boldsymbol{N}_*^{(k+1)p} - \boldsymbol{N}^{(k+1)q}\|_\infty \leq \epsilon \sum_{\sigma=3}^{k+2} \exp(\sum_{\nu=\sigma}^{k+1} L_{i(\nu-1)}(\tau_\nu - \tau_{\nu-1}))$$

Then, note that when we loop over all possible decisions $\boldsymbol{x}_{i(k+1)}$, we must have covered all possible clustered states within $\mathcal{N}^{(k+2)}$ by construction of Algorithm 2. Therefore we have that for every $\boldsymbol{N}^{(k+2)q} \in \mathcal{N}^{(k+2)}$, there exists $\boldsymbol{N}_*^{(k+2)p} \in \mathcal{N}_*^{(k+2)}$ such that:

$$\|\boldsymbol{N}_*^{(k+1)p} - \boldsymbol{N}^{(k+1)q}\|_\infty \leq \epsilon \sum_{\sigma=3}^{k+2} \exp(\sum_{\nu=\sigma}^{k+1} L_{i(\nu-1)}(\tau_\nu - \tau_{\nu-1})) \tag{3.64}$$

And the induction step is proved. □

## 3.4 A tri-partite branching strategy to solve the full problem via branch-and-price

The column generation algorithm solves the linear relaxation of $(\mathcal{SP})$. The resulting solution, however, is not guaranteed to be integral. Therefore, we embed the column generation procedure into a branch-and-price structure to retrieve a feasible, and optimal, solution to the $(\mathcal{SP})$ formulation. We proceed with multi-phase branching: we first branch on the "other" variables $\boldsymbol{y}$ distinct from resource allocation; we then branch on the natural resource allocation variables $\boldsymbol{x}_{is}$; and we finally restore integrality of the plan-based variables $z_i^p$ via a novel tri-partite branching disjunction.

**Branching on the "other" variables.** In the case where the formulation involves discrete variables $\boldsymbol{y}$, we first branch on those variables whenever one of its components is fractional, as follows:

$$\underbrace{(y_\ell \leq \lfloor \widehat{y}_\ell \rfloor)}_{\text{left branch}} \vee \underbrace{(y_\ell \geq \lceil \widehat{y}_\ell \rceil)}_{\text{right branch}}, \text{ where } \widehat{y}_\ell \text{ denotes the } \ell^{\text{th}} \text{ component of the incumbent solution} \tag{3.65}$$

**Branching on the natural variables.** To avoid building deep and one-sided trees and maintain the structure of the pricing problem, a common approach in branch-and-price involves branching on the natural variables (i.e., on the resource allocation decisions $\boldsymbol{x}_{is}$ in our case) as opposed to branching on the plan-based variables (i.e., on the variables $z_i^p$). For convenience, let us denote by $\boldsymbol{x}_{is}(\boldsymbol{z}) \in \mathbb{R}^{d_{is}}$ the natural resource allocation variable in segment $i = 1, \cdots, n$ and epoch $s = 1, \cdots, S$ for a plan-based solution $\boldsymbol{z}$; and let $x_{is}^k(\boldsymbol{z}) \in \mathbb{R}$ denote its $k^{\text{th}}$ component, for $k = 1, \cdots, d_{is}$. Let $\widehat{\boldsymbol{z}}$ be the solution of the column generation procedure in a given node. The resource allocation decisions may be infeasible if $\boldsymbol{x}_{is}(\widehat{\boldsymbol{z}}) \notin \mathcal{F}_{is}$. In that case, we exploit the discreteness of the feasible region $\mathcal{F}_{is}$ to create a valid disjunction. Since the feasible region $\mathcal{F}_{is}$ does not necessarily comprise contiguous integers, we denote by $\lfloor a \rfloor_{is}$ and $\lceil a \rceil_{is}$ the nearest elements of $\mathcal{F}_{is}$ that are smaller and larger than $a$, respectively, i.e., $\lfloor a \rfloor_{is} = \max\{b \in \mathcal{F}_{is} : b \leq a\}$ and $\lceil a \rceil_{is} = \min\{b \in \mathcal{F}_{is} : b \geq a\}$.

Armed with these notations, we can then define the following valid branching disjunction:

$$\underbrace{\left(x_{is}^k(\boldsymbol{z}) \leq \left\lfloor x_{is}^k(\widehat{\boldsymbol{z}}) \right\rfloor_{is}\right)}_{\text{left branch}} \vee \underbrace{\left(x_{is}^k(\boldsymbol{z}) \geq \left\lceil x_{is}^k(\widehat{\boldsymbol{z}}) \right\rceil_{is}\right)}_{\text{right branch}}, \text{ with } \boldsymbol{x}_{is}(\widehat{\boldsymbol{z}}) = \sum_{p \in \mathcal{P}_i} \boldsymbol{\alpha}_{is}^p \widehat{z}_i^p \quad (3.66)$$

Out of these disjunctions, we select one that corresponds to a variable $x_{is}^k(\widehat{\boldsymbol{z}})$ with the largest value of $\min\left\{x_{is}^k(\widehat{\boldsymbol{z}}) - \left\lfloor x_{is}^k(\widehat{\boldsymbol{z}}) \right\rfloor_{is}, \left\lceil x_{is}^k(\widehat{\boldsymbol{z}}) \right\rceil_{is} - x_{is}^k(\widehat{\boldsymbol{z}})\right\}$. This branching strategy is an analog to the *most fractional* branching strategy in integer optimization. Importantly, this disjunction preserves the structure of the pricing problem, by merely restricting the search from the full feasible regions $\mathcal{F}_{is}$ to their subregions defined by the corresponding lower-bound and upper-bound constraints.

**Tri-partite branching.** By design, the above branching scheme provides an enumerative algorithm to solve the $\mathcal{SP}$ formulation, while guaranteeing that the plan-based variables define feasible resource allocation decisions. However, it may not guarantee integral plan-based variables $z_i^p$; for instance, it may be the case that $\alpha_{is}^{p_1} = 4$, $\alpha_{is}^{p_2} = 8$, $z_i^{p_1} = z_i^{p_2} = 0.5$, and $6 \in \mathcal{F}_{is}$. In linear problems, this solu-

tion can be brought into an equivalent feasible solution by considering the "average plan" $p^*$ with $\alpha_{is}^{p^*} = 6$. In our problem, however, the non-convex system dynamics break the equivalence between the fractional solution and the average plan, because $C_i^{p^*} \neq 0.5C_i^{p_1} + 0.5C_i^{p_2}$ in general. As a result, there is no more guarantee that the plan $p^*$ will form an optimal solution to Problem $(\mathcal{P})$.

It is therefore critical to also enforce the integrality of the plan-based variables in the branch-and-price algorithm. A direct way to do so would be to create disjunctions of the form $(z_i^p = 0) \vee (z_i^p = 1)$. However, as noted earlier, this disjunction can lead to weak and imbalanced tree structures. Moreover, this disjunction breaks the structure of the pricing problem—notably, it is difficult to seek the "second-best" solution in the branch associated with the $z_i^p = 0$ disjunction.

Instead, we devise a new tri-partite branching disjunction to handle non-linearities. Consider a node with a solution $\widehat{\boldsymbol{z}}$. Assume that $\boldsymbol{x}_{is}(\widehat{\boldsymbol{z}}) \in \mathcal{F}_{is}$ for all $i, s$ but that there exists a fractional plan-based variable $z_i^p \in (0, 1)$. Let $\alpha_{is}^{pk}(\boldsymbol{z}) \in \mathbb{R}$ denote the $k^{\text{th}}$ component of $\boldsymbol{\alpha}_{is}^p$ for $k = 1, \cdots d_{is}$. There must exist a segment $i$, an epoch $s$, a component $k$ and a plan $p_0$ such that:

$$\boldsymbol{x}_{is}(\widehat{\boldsymbol{z}}) = \sum_{p \in \mathcal{P}_i} \boldsymbol{\alpha}_{is}^p \widehat{z}_i^p \in \mathcal{F}_{is} \text{ and } \alpha_{is}^{k,p_0} \neq x_{is}^k(\widehat{\boldsymbol{z}}) \text{ and } \widehat{z}_i^{p_0} > 0. \tag{3.67}$$

Intuitively, $\boldsymbol{x}_{is}(\widehat{\boldsymbol{z}})$ is the "promising" resource allocation decision for segment $i$ at epoch $s$. Accordingly, we create a corresponding branch in the tree. To retain a mutually exclusive and collectively exhaustive tree, we create two additional branches with lower and higher resource allocations. Formally, our tri-partite branching disjunction is defined via the following disjunction:

$$\underbrace{\left(x_{is}^k(\boldsymbol{z}) < x_{is}^k(\widehat{\boldsymbol{z}})\right)}_{\text{left branch}} \vee \underbrace{\left(x_{is}^k(\boldsymbol{z}) = x_{is}^k(\widehat{\boldsymbol{z}})\right)}_{\text{middle branch}} \vee \underbrace{\left(x_{is}^k(\boldsymbol{z}) > x_{is}^k(\widehat{\boldsymbol{z}})\right)}_{\text{right branch}}, \text{ with } \boldsymbol{x}_{is}(\widehat{\boldsymbol{z}}) = \sum_{p \in \mathcal{P}_i} \boldsymbol{\alpha}_{is}^p \widehat{z}_i^p \tag{3.68}$$

This disjunction makes the incumbent solution infeasible. To see this, let $\widehat{\mathcal{P}}_i = \{p \in \mathcal{P}_i : \widehat{z}_i^p > 0\}$ denote the set of plans included in the incumbent solution for

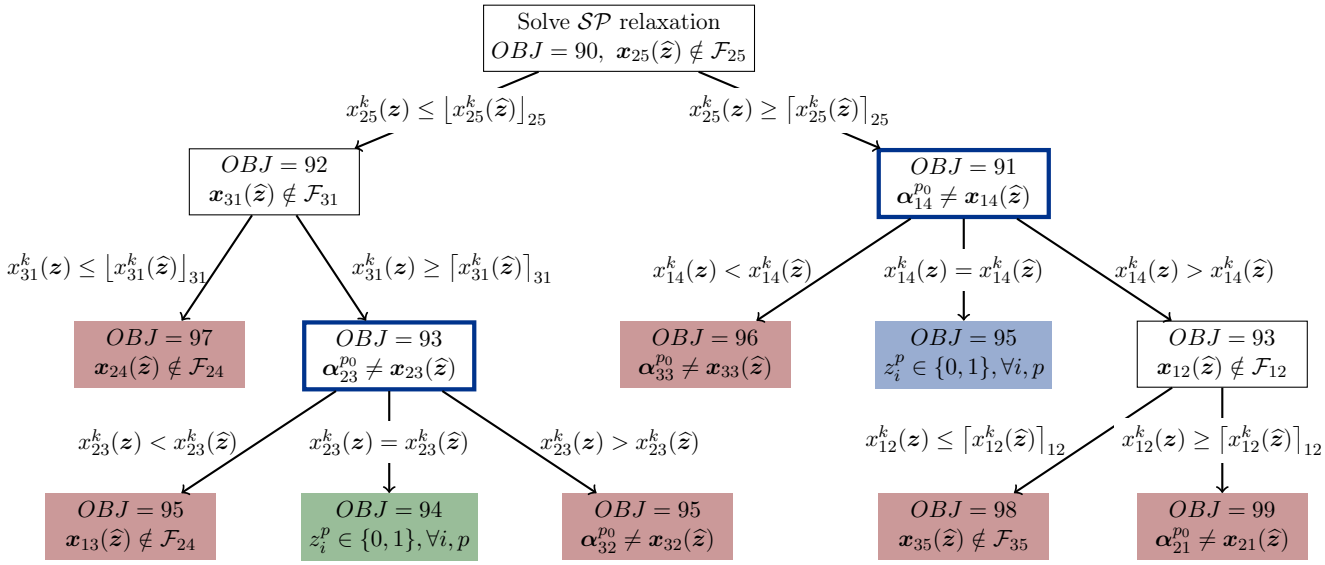



Figure 3-1: Illustration of the branching structure, combining bi-partite branching and tri-partite branching. Red nodes are pruned by bound; blue nodes are pruned by feasibility; white nodes trigger bi-partite branching; blue squares trigger tri-partite branching; the green node indicates the optimal solution.

segment $i$, partitioned into $\widehat{\mathcal{P}}_i^{<} = \{p \in \widehat{\mathcal{P}}_i : \alpha_{is}^{pk} < x_{is}^k(\widehat{\boldsymbol{z}})\}$, $\widehat{\mathcal{P}}_i^{=} = \{p \in \widehat{\mathcal{P}}_i : \alpha_{is}^{pk} = x_{is}^k(\widehat{\boldsymbol{z}})\}$, and $\widehat{\mathcal{P}}_i^{>} = \{p \in \widehat{\mathcal{P}}_i : \alpha_{is}^{pk} > x_{is}^k(\widehat{\boldsymbol{z}})\}$. Since $\alpha_{is}^{k,p_0} \neq x_{is}^k(\widehat{\boldsymbol{z}})$, we know that $\widehat{\mathcal{P}}_i^{<} \neq \emptyset$ *or* $\widehat{\mathcal{P}}_i^{>} \neq \emptyset$. But since, in addition, $\sum_{p \in \mathcal{P}_i} \alpha_{is}^{pk} \widehat{z}_i^p = x_{is}^k$, we actually know that $\widehat{\mathcal{P}}_i^{<} \neq \emptyset$ *and* $\widehat{\mathcal{P}}_i^{>} \neq \emptyset$. The incumbent solution is therefore infeasible in the left branch, which eliminates the plans in $\widehat{\mathcal{P}}_i^{>}$; it is infeasible in the right branch, which eliminates the plans in $\widehat{\mathcal{P}}_i^{<}$; and it is infeasible in the middle branch, which eliminates the plans in both.

A final question lies in variable selection to determine which $x_{is}^k(\widehat{\boldsymbol{z}})$ to branch upon. We extend the *most fractional* principle, by selecting the variable with the largest weighted difference between the plan-based allocations and the resource allocations from the column generation solution. Formally, we select $x_{is}^k(\widehat{\boldsymbol{z}})$ such that the $(i,k,s)$ tuple maximizes $\sum_{p \in \mathcal{P}_i} \widehat{z}_i^p |\alpha_{is}^{pk} - x_{is}^k(\widehat{\boldsymbol{z}})|$.

Again, this branching disjunction does not impact the structure of the pricing problem, which can easily accommodate lower-bounding and upper-bounding constraints on the decision variables.

## 3.5 Summary of the branch-and-price algorithm

Algorithm 3 summarizes our solution approach. In each node, the algorithm solves the linear relaxation of $\mathcal{SP}$ via column generation (Step 1), iterating between the RMP (Step 1.1) and the dynamic programming algorithm for the pricing problem (Step 1.2), until convergence (Step 1.3). It then proceeds to bi-partite branching when the variables $\boldsymbol{y}$ are not integral or when the natural resource allocation variables $\boldsymbol{x}$ are infeasible (Steps 2 and 3), and to tri-partite branching to restore integrality in plan-based variables (Step 4). This branching strategy is illustrated in Figure 3-1. Step 5 corresponds to the node selection; we have implemented breadth-first and depth-first search strategies, and found comparable performance. The algorithm uses bounding and feasibility rules to prune leaves (Step 1.3) and terminates when the list of active leaves is empty (Step 5).

Proposition 5 establishes the convergence and exactness of the algorithm, as long as the pricing problem is solved to optimality (Step 1.2). This follows directly from the facts that the algorithm maintains a valid lower bound and a valid upper bound, and that the branching disjunction eliminates any infeasible solution (as proved in Section 3.4). The finite convergence of the algorithm follows from the finiteness of the decision space. Note that the pricing problem does not need to be solved to optimality at each iteration but merely upon convergence; therefore, an exact acceleration strategy would involve applying the state-clustering dynamic programming heuristic in initial iterations and then turning to the exact dynamic programming algorithm until convergence.

**Proposition 5.** *The branch-and-price algorithm with the tri-partite branching disjunction converges in a finite number of iterations and returns an optimal solution to the $\mathcal{SP}$ formulation.*

**Algorithm 3** Branch-and-price algorithm for Problem $(\mathcal{P})$.

1: **Initialization:** node list $\Phi = \{0\}$ of the branch-and-bound tree; parent node $cur = 0$; initial sets of plans $\mathcal{P}_1^0, \cdots, \mathcal{P}_n^0$ and corresponding RMP (3.48)–(3.51); upper bound $UB = +\infty$; incumbent solution $\boldsymbol{z}^* = \emptyset$.
2: ***Step 1.*** Solve parent node *cur* via column generation (Section 3.2); store objective $OBJ$, solution $\widehat{\boldsymbol{z}}$, and corresponding resource allocation decisions $\boldsymbol{x}_{is}(\boldsymbol{z}) = \sum_{p \in \mathcal{P}_i} \boldsymbol{\alpha}_{is}^p \widehat{z}_i^p$; remove *cur* from node list $\Phi$.
3: ***Step 1.1.*** Solve RMP relaxation with $\mathcal{P}_1^0, \cdots, \mathcal{P}_n^0$; store solution $\widehat{\boldsymbol{z}}$, objective $OBJ$, and dual costs $\boldsymbol{\lambda}, \boldsymbol{\mu}$.
4: ***Step 1.2.*** For each segment $i = 1, \cdots, n$, solve PP using exact algorithm (Algorithm 1) or approximate state-clustering acceleration (Algorithm 2). If the objective is negative, add optimal plan to $\mathcal{P}_i^0$.
5: ***Step 1.3.*** If the objective of every pricing problem is positive, `BREAK`. Otherwise, go to Step 1.1.
6: – If $OBJ < UB$ and $\widehat{y}_\ell \notin \mathbb{Z}$ for some $\ell = 1, \cdots, q$; go to Step 2.
7: – If $OBJ < UB$ and $\boldsymbol{x}_{is}(\widehat{\boldsymbol{z}}) \notin \mathcal{F}_{is}$ for some $i, s$; go to Step 3.
8: – If $OBJ < UB$ and $\boldsymbol{x}_{is}(\widehat{\boldsymbol{z}}) \in \mathcal{F}_{is}$, $\forall i, s$ and $z_i^p$ is fractional for some $i, p$; go to Step 4.
9: – If $OBJ < UB$ and $\boldsymbol{x}_{is}(\widehat{\boldsymbol{z}}) \in \mathcal{F}_{is}$, $\forall i, s$ and $\widehat{\boldsymbol{z}}$ is integral; update $UB \leftarrow OBJ$, $\boldsymbol{z}^* \leftarrow \widehat{\boldsymbol{z}}$; go to Step 5.
10: – If $OBJ \geq UB$; go to Step 5.
11: ***Step 2:*** **branching on $y$ variables.** Add two children nodes to $\Phi$ (Equation (3.65)); go to Step 5.
12: ***Step 3:*** **bi-partite branching.** Add two children nodes to $\Phi$ (Equation (3.66)); go to Step 5.
13: ***Step 4:*** **tri-partite branching.** Solve RMP via integer optimization; store solution $\widehat{\boldsymbol{z}}$ and objective $IO$; if $IO \leq UB$, $UB \leftarrow IO$ and $\boldsymbol{z}^* \leftarrow \widehat{\boldsymbol{z}}$. Add three children nodes to $\Phi$ (Equation (3.68)); go to Step 5.
14: ***Step 5.*** If $\Phi = \emptyset$, `STOP`; return solution $\boldsymbol{x}^*$ and objective $UB$. Otherwise, choose $cur \in \Phi$; go to Step 1.

# Chapter 4

# Experimental results

We evaluate our methodology on the four problems described in Section 2. For each one, we design a real-world experimental setup at practical scale, by setting the number of segments $n$, the planning horizon $S$, and the total number of decisions per epoch $D_{is}$. Specifically, we define:

- a vaccine allocation problem based on (21) with up to 51 states, 12 decision epochs and 21 decisions per epoch. This setup mirrors US data in the midst of the COVID-19 pandemic, with an allocation of a weekly stockpile of 2.5 to 7 million vaccines, discretized into 20 increments, over a three-month horizon from February to April 2021.

- a mass vaccination center case with up to 7 groups of states and 20 candidate facilities per group. Due to the complexity of this problem, we break down the United States into 10 groups defined by the Centers for Disease Control and Prevention (Table 4.1). We consider a budget of 20 and 30 facilities across the country, again mirroring the situation in the United States in 2021. For equity purposes, each of the 10 groups receives 2-3 facilities. Each facility can serve residents within 100 miles (with 30 facilities) or 150 miles (with 20 facilities). Each region receives a budget of vaccines proportionally to its share of the US population, discretized into pallets of 25,000 vaccines, plus some buffer to allow for flexibility in vaccine allocations.

Table 4.1: Composition of the groups for the vaccination centers problem.

| Group | States | % US Population | Vaccines | Discretization |
|---|---|---|---|---|
| A | Connecticut, Maine, Massachusetts, New Hampshire, Rhode Island, Vermont, New York | 11.8% | 300K | 12 |
| B | Delaware, District of Columbia, Maryland, Pennsylvania, Virginia, West Virginia, New Jersey | 9.8% | 250K | 10 |
| C | North Carolina, South Carolina, Georgia, Florida | 12.7% | 325K | 13 |
| D | Kentucky, Tennessee, Alabama, Mississippi | 4.9% | 125K | 5 |
| E | Illinois, Indiana, Michigan, Minnesota, Ohio, Wisconsin | 15.6% | 400K | 16 |
| F | Arkansas, Louisiana, New Mexico, Oklahoma, Texas | 12.7% | 325K | 13 |
| G | Iowa, Kansas, Missouri, Nebraska | 3.9% | 100K | 4 |
| H | Colorado, Montana, North Dakota, South Dakota, Utah, Wyoming | 3.9% | 100K | 4 |
| I | Arizona, California, Hawaii, Nevada | 19.6% | 500K | 20 |
| J | Alaska, Idaho, Oregon, Washington | 4.9% | 125K | 5 |

– a content promotion case inspired by (50) with up to 20 products, 10 decision epochs, 21 decisions per epoch, and a sparsity constraint of 2-6 products per epoch.

– a congestion mitigation case, with 5 urban neighborhoods and up to 8 time periods, and 12 prevention and treatment vehicles to allocate, mirroring real-world situations in Singapore.

All experiments were run on Julia v1.5.2 optimization package JuMP (30) on a single core of a 10-core i9-10900k CPU. We make the instances and code available online.

## 4.1 Benefits of the state-clustering dynamic programming algorithm

Our methodology relies on the ability to solve the pricing problem efficiently. To compare the state-clustering dynamic programming algorithm to the exact algorithm

based on exhaustive enumeration, Table 4.2 reports the total number of states and computational times, broken down into the initialization time to create the state space and the dynamic programming time. For all instances solved by the exact algorithm, we also report the Median Average Percentage Error (MAPE) defined as the median relative error between all clustered states and the true states under exhaustive enumeration.

Table 4.2: State-clustering dynamic programming against exhaustive enumeration (vaccine allocation case).

| | | | | | | **Time (sec.)** | | **MAPE (%)** | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | $S$ | $D$ | Method | Tolerance | $\|S\|$ | Init. | DP | $s=1$ | $s=2$ | $s=3$ | $s=4$ |
| 51 | 4 | 6 | Enumeration | — | 79,305 | 20.01 | 1.33 | — | — | — | — |
| | | | Clustering | $\varepsilon = 0.002$ | 2,350 | 3.33 | 0.08 | 0.0 | 0.0 | 0.5 | 0.7 |
| | | | Clustering | $\varepsilon = 0.005$ | 1,458 | 2.20 | 0.04 | 0.0 | 0.0 | 1.3 | 2.1 |
| | | | Clustering | $\varepsilon = 0.01$ | 982 | 1.57 | 0.03 | 0.0 | 0.0 | 7.2 | 9.4 |
| 51 | 4 | 11 | Enumeration | — | 821K | 194 | 17.1 | — | — | — | — |
| | | | Clustering | $\varepsilon = 0.002$ | 4,971 | 13.3 | 0.33 | 0.0 | 0.1 | 0.6 | 0.9 |
| | | | Clustering | $\varepsilon = 0.005$ | 2,533 | 6.74 | 0.19 | 0.0 | 0.1 | 2.9 | 3.7 |
| | | | Clustering | $\varepsilon = 0.01$ | 1,521 | 4.29 | 0.10 | 0.0 | 0.1 | 6.3 | 10.0 |
| 51 | 4 | 21 | Enumeration | — | 10.4M | 2950 | 234 | — | — | — | — |
| | | | Clustering | $\varepsilon = 0.002$ | 10,615 | 74.6 | 1.09 | 0.0 | 0.1 | 0.9 | 1.1 |
| | | | Clustering | $\varepsilon = 0.005$ | 4,477 | 26.9 | 0.58 | 0.0 | 0.1 | 3.5 | 5.1 |
| | | | Clustering | $\varepsilon = 0.01$ | 2,473 | 14.8 | 0.29 | 0.0 | 0.2 | 7.2 | 13.6 |
| 51 | 6 | 21 | Enumeration | — | 4.59B | n/a | n/a | — | — | — | — |
| | | | Clustering | $\varepsilon = 0.002$ | 29,090 | 370 | 7.41 | n/a | n/a | n/a | n/a |
| | | | Clustering | $\varepsilon = 0.005$ | 9,212 | 75.0 | 1.48 | n/a | n/a | n/a | n/a |
| | | | Clustering | $\varepsilon = 0.01$ | 4,873 | 33.3 | 0.55 | n/a | n/a | n/a | n/a |
| 51 | 8 | 21 | Enumeration | — | 2.06T | n/a | n/a | — | — | — | — |
| | | | Clustering | $\varepsilon = 0.002$ | 56,635 | 1057 | 13.6 | n/a | n/a | n/a | n/a |
| | | | Clustering | $\varepsilon = 0.005$ | 15,076 | 147 | 3.50 | n/a | n/a | n/a | n/a |
| | | | Clustering | $\varepsilon = 0.01$ | 6,341 | 52.4 | 1.47 | n/a | n/a | n/a | n/a |
| 51 | 10 | 21 | Enumeration | — | 893T | n/a | n/a | — | — | — | — |
| | | | Clustering | $\varepsilon = 0.002$ | 89,685 | 2586 | 32.8 | n/a | n/a | n/a | n/a |
| | | | Clustering | $\varepsilon = 0.005$ | 21,308 | 238 | 5.50 | n/a | n/a | n/a | n/a |
| | | | Clustering | $\varepsilon = 0.01$ | 8,353 | 77.9 | 2.13 | n/a | n/a | n/a | n/a |

Median Absolute Percentage Error (MAPE) is defined against the exact solution.
"n/a" means that the exact dynamic programming algorithm does not terminate due to memory limitations.

Note, first, that the number of states grows very quickly, reflecting the "curse of dimensionality" in dynamic programming. Even by exploiting the segment-based decomposition in the pricing problem, the state space scales in $\mathcal{O}(n \times D^S)$, with trillions of states in our largest instances. Exhaustive enumeration does not scale to even small instances, requiring minutes to converge with 4 segments and 4 decision

epochs. The state-clustering algorithm considerably reduces the state space—by up to a factor of $10^{10} - 10^{11}$. Thus, it accelerates convergence by two orders of magnitude in small instances (seconds versus minutes), and scales to the largest instance with 51 states, 10 epochs, and 21 decisions in minutes when the exact algorithm failed to terminate.

Moreover, the state-clustering algorithm provides a high-quality approximation of the full state space. With $\varepsilon = 0.01$, the algorithm results in a median absolute error up to $4.9 \times 10^{-4}$ and a median absolute percentage error up to 13.6%. A tolerance of $\varepsilon = 0.002$ further reduces the MAE within $6 \times 10^{-5}$ and the MAPE within 1.1%, at some computational cost. Note that the error is significantly smaller than the allowed tolerance. Importantly, though, the error increases over time: the clustering output introduces only minor perturbations in the first two periods, but the quality of the approximation deteriorates thereafter, reflecting the propagation of errors in dynamical systems. Whereas we cannot estimate the error over longer time horizons because of the limitations of the enumerative algorithm, this observation could motivate dynamic implementations of our methodology in practice—for instance, by re-evaluating the system's dynamics at each epoch and re-optimizing resource allocations accordingly. Nonetheless, these results suggest that the drift in state approximation remains moderate, underscoring the critical role of the state-clustering algorithm to enhance the tractability of the pricing problem at small costs in terms of accuracy.

## 4.2 Benefits of the branch-and-price algorithm

Armed with the state-clustering algorithm, we now solve our four prescriptive contagion analytics with our branch-and-price algorithm. Table 4.3 reports computational times, broken down into state space initialization, restricted master problems, and pricing problems throughout the branch-and-price tree. Recall that the state space of the pricing problem does not change across iterations, and can thus be performed only once throughout the algorithm. The table also reports the solution and the lower bound obtained via column generation, via bi-partite branching (i.e., a branch-

and-price algorithm that only branches on the natural variables), and with the full branch-and-price algorithm with tri-partite branching. The column generation and bi-partite branching solutions are obtained by solving the restricted master problem via integer optimization upon convergence—a heuristic sometimes referred to as "price-and-branch". For brevity, the table reports results for "easy", "medium" and "hard" instances and full results are deferred to Appendix A. Yet, all results yield similar observations, thus showing the robustness of the findings derived from Table 4.3.

Note, first, that the column generation algorithm is highly scalable: it solves the set partitioning relaxation in seconds even in the harder instances. The restricted master problem is virtually instantaneous but the pricing problem is more time-consuming due to the non-linear system dynamics—reinforcing the need for an efficient dynamic programming algorithm (Section 4.1). In fact, the state-clustering algorithm shifts the complexity away from the dynamic programming algorithm itself toward—enabling the online column generation algorithm to consistency terminate in seconds—toward the offline pre-processing of the state space—which can take up to minutes.

Moreover, the column generation heuristic derives high-quality solutions in most instances of the vaccine allocation, content promotion, and congestion mitigation problems. Even without resorting to the branching scheme, the "price-and-branch" heuristic makes use of the tight set partitioning reformulation to generate high-quality solutions in seconds. In the vaccine allocation case in particular, the column generation algorithm by itself yields solutions within a 1-2% optimality gap. For the other two problems, however, column generation can leave a larger optimality gap in other instances, due to the sparsity constraint in the content promotion case and to the coordination of prevention and treatment resources in the congestion mitigation case. This limitation of the column generation heuristic motivates our branch-and-price algorithm as an exact solution method.

Turning to our main observation, the branch-and-price algorithm solves every instance of the three pure resource allocation problems to near-optimality. The number of nodes remains moderate in some instances but can grow larger for harder instances, leading to higher computational requirements. As a result, the algorithm terminates

Table 4.3: Performance evaluation of the branch-and-price algorithm across use cases.

| | | | | CPU times (s) | | | | Solution quality | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Problem | Instance | Method | Nodes | Init. | RMP | PP | Alg. | UB | Solution | Gap |
| Vaccine allocation | Easy | CG | 1 | 21 | 0.02 | 0.22 | 0.29 | 93,695 | 92,261 | 1.53% |
| | | 2-BP | 28 | 21 | 0.27 | 1.43 | 7.11 | 93,379 | 93,266 | 0.12% |
| | | 3-BP | 46 | 21 | 0.42 | 2.2 | 9.6 | 93,353 | 93,263 | 0.09% |
| | Medium | CG | 1 | 43 | 0.022 | 0.81 | 0.92 | 156,465 | 155,624 | 0.54% |
| | | 2-BP | 4,778 | 43 | 53 | 426 | 1,064 | 156,183 | 156,026 | 0.1% |
| | | 3-BP | 4,778 | 43 | 55 | 420 | 1,058 | 156,183 | 156,027 | 0.1% |
| | Hard | CG | 1 | 144 | 0.03 | 4.07 | 4.32 | 225,323 | 222,592 | 1.21% |
| | | 2-BP | 9,990 | 144 | 101 | 1,765 | 3,475 | 225,048 | 224,495 | 0.24% |
| | | 3-BP | 9,990 | 144 | 103 | 1,808 | 3,535 | 225,048 | 224,495 | 0.24% |
| Vaccination centers | Easy, F=2 | CG | 1 | 39 | 0.03 | 0.02 | 0.46 | 785 | 652 | 55.5% |
| | | 2-BP | 1,936 | 39 | 45 | 10 | 351 | 764 | 763 | 0.09% |
| | | 3-BP | 1,936 | 39 | 45 | 10 | 351 | 764 | 763 | 0.09% |
| | Easy, F=3 | CG | 1 | 44 | 0.02 | 0.03 | 0.54 | 677 | 652 | 3.75% |
| | | 2-BP | 53,443 | 44 | 1,328 | 122 | 10,826 | 675 | 673 | 0.23% |
| | | 3-BP | 53,590 | 44 | 1,417 | 118 | 10,801 | 675 | 673 | 0.23% |
| | Hard, F=2 | CG | 1 | 45 | 0.14 | 0.42 | 1.05 | 8,200 | 7,133 | 18.5% |
| | | 2-BP | 13,338 | 45 | 235 | 202 | 10,851 | 8,141 | 7,917 | 2.75% |
| | | 3-BP | 13,683 | 45 | 228 | 198 | 10,811 | 8,134 | 7,917 | 2.66% |
| | Hard, F=3 | CG | 1 | 45 | 0.17 | 0.45 | 1.11 | 8200 | 6822 | 40.8% |
| | | 2-BP | 12,304 | 45 | 268 | 187 | 10,832 | 8,061 | 7,724 | 4.15% |
| | | 3-BP | 13,451 | 45 | 294 | 193 | 11,072 | 8,059 | 7,725 | 4.14% |
| Content promotion | Easy | CG | 1 | 79 | 0.04 | 4.6 | 4.8 | 1.013 | 0.98 | 3.86% |
| | | 2-BP | 696 | 79 | 9.3 | 198 | 327 | 1.013 | 1.009 | 0.53% |
| | | 3-BP | 142 | 79 | 1.6 | 52 | 74 | 1.013 | 1.0125 | 0.09% |
| | Medium | CG | 1 | 58 | 0.04 | 5.01 | 5.4 | 1.52 | 1.44 | 5.66% |
| | | 2-BP | 17,960 | 58 | 294 | 1,949 | 7,201 | 1.52 | 1.51 | 0.83% |
| | | 3-BP | 17,819 | 58 | 3,380 | 2,0570 | 7,201 | 1.52 | 1.517 | 0.31% |
| | Hard | CG | 1 | 134 | 0.08 | 15 | 16 | 2.02 | 1.87 | 7.08% |
| | | 2-BP | 12,606 | 134 | 229 | 3,221 | 10,801 | 2.009 | 1.99 | 1.04% |
| | | 3-BP | 11,864 | 134 | 229 | 3,279 | 10,801 | 2.009 | 1.99 | 1.04% |
| Congestion mitigation | Easy | CG | 1 | 10 | 0.012 | 0.08 | 0.11 | 0.82 | 0.81 | 1.23% |
| | | 2-BP | 2 | 10 | 0.018 | 0.126 | 0.56 | 0.82 | 0.81 | 0.16% |
| | | 3-BP | 8 | 10 | 0.053 | 0.23 | 0.8 | 0.82 | 0.82 | 0% |
| | Medium | CG | 1 | 185 | 0.024 | 6.2 | 6.27 | 2.20 | 2.05 | 6.61% |
| | | 2-BP | 48 | 185 | 0.34 | 32 | 34 | 2.19 | 2.19 | 0.21% |
| | | 3-BP | 57 | 185 | 0.43 | 39 | 42 | 2.19 | 2.19 | 0.06% |
| | Hard | CG | 1 | 459 | 0.044 | 20 | 20 | 3.32 | 2.93 | 11.9% |
| | | 2-BP | 206 | 459 | 4.14 | 350 | 389 | 3.31 | 3.31 | 0.2% |
| | | 3-BP | 243 | 459 | 4.9 | 396 | 443 | 3.31 | 3.31 | 0.09% |

Vaccine allocation: $S = 8$, $D = 5$ (easy); $S = 10$, $D = 10$ (medium); $S = 12$, $D = 20$ (hard).
Vaccination centers: Group H (easy) and Group A (hard), with 2 or 3 facilities per group, $S = 6$.
Content promotion: $n = 20$; $D_y = 2$ ; $S = 6$, $D_x = 20$, (easy); $S = 8$, $D_x = 10$, (medium); $S = 8$, $D_x = 20$ (hard).
Congestion mitigation: $n = 5$; $D_x = 6$; $D_y = 4$; $S = 2$ (easy); $S = 4$ (medium); $S = 6$ (hard).
CG (Column Generation), 2-BP (Bi-partite Branch-&-Price), 3-BP (Tri-partite Branch-&-Price).

Table 4.4: Performance evaluation of the branch-and-price algorithm for Facility. Full country, F=3

| | | | | | | | CPU times (s) | | | | Solution quality | | % |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Group | n | S | D | F | Method | Nodes | Init. | RMP | PP | Alg. | UB | Solution | Gap |
| A | 7 | 6 | 12 | 3 | CG | 1 | 45 | 0.17 | 0.45 | 1.1 | 8,200 | 6,822 | 40.8% |
| | | | | | 3-BP | 13K | 45 | 294 | 193 | 11,072 | 8,059 | 7,725 | 4.14% |
| B | 7 | 6 | 10 | 3 | CG | 1 | 38 | 0.041 | 0.24 | 0.7 | 3,658 | 3,334 | 29.5% |
| | | | | | 3-BP | 12K | 38 | 249 | 135 | 10,815 | 3,604 | 3,460 | 4.01% |
| C | 4 | 6 | 13 | 3 | CG | 1 | 68 | 0.03 | 0.23 | 0.94 | 3,880 | 3,170 | 18.3% |
| | | | | | 3-BP | 9K | 68 | 375 | 164 | 10,810 | 3,880 | 3,692 | 4.84% |
| D | 4 | 6 | 5 | 3 | CG | 1 | 47 | 0.02 | 0.04 | 0.55 | 1,056 | 732 | 45.4% |
| | | | | | 3-BP | 25K | 47 | 632 | 64 | 10,813 | 1,051 | 1,000 | 4.95% |
| E | 6 | 6 | 16 | 3 | CG | 1 | 48 | 0.07 | 0.65 | 1.19 | 8,331 | 7,113 | 14.6% |
| | | | | | 3-BP | 8K | 48 | 151 | 192 | 10,816 | 8,330 | 8,018 | 3.75% |
| F | 5 | 6 | 13 | 3 | CG | 1 | 41 | 0.06 | 0.18 | 0.63 | 3,593 | 2,273 | 89.6% |
| | | | | | 3-BP | 16K | 41 | 687 | 154 | 10,809 | 3,512 | 3,334 | 5.08% |
| G | 4 | 6 | 4 | 3 | CG | 1 | 59 | 0.02 | 0.03 | 0.72 | 1,985 | 1,869 | 5.83% |
| | | | | | 3-BP | 42K | 59 | 733 | 109 | 10,803 | 1,958 | 1,921 | 1.91% |
| H | 6 | 6 | 4 | 3 | CG | 1 | 44 | 0.03 | 0.03 | 0.6 | 678 | 652 | 3.75% |
| | | | | | 3-BP | 54K | 44 | 1,417 | 118 | 10,801 | 675 | 674 | 0.23% |
| I | 4 | 6 | 20 | 3 | CG | 1 | 49 | 0.06 | 0.2 | 0.79 | 3,573 | 2,587 | 34.9% |
| | | | | | 3-BP | 41K | 49 | 5,056 | 471 | 11,252 | 3,422 | 3,316 | 3.09% |
| J | 4 | 6 | 5 | 3 | CG | 1 | 43 | 0.02 | 0.02 | 0.5 | 626 | 565 | 9.73% |
| | | | | | 3-BP | 21K | 43 | 314 | 36 | 10,805 | 624 | 565 | 9.46% |
| Full | 51 | 6 | — | 3 | CG | — | — | — | — | — | 35,580 | 29,118 | 18.2% |
| | | | | | 3-BP | — | — | — | — | — | 35,117 | 33,704 | 4.02% |

in minutes to hours. Nonetheless, it consistently improves the solution from the column generation heuristic. In fact, it reaches optimality within a 0.1% tolerance in most instances, and leaves a small optimality gap otherwise. In the content promotion and congestion mitigation cases, the branch-and-price solution improves the column generation solution by up to 6-12%. Even in the vaccine allocation case, where the column generation algorithm leaves a small optimality gap, the branch-and-price solution yields a 1% improvement, amounting to an extra 2,000 lives saved over a three-month horizon.

Finally, the standard bi-partite branching scheme is not sufficient to guarantee convergence to an optimal solution. For instance, it leaves a 1% optimality gap in content promotion instances and a 0.2% gap in the congestion mitigation instances. This result underscores the impact of the non-linearities of the system dynamics on the branch-and-price algorithm. In turn, the tri-partite branching scheme developed in this thesis is instrumental to ensure finite convergence to an optimal solution, while retaining an efficient branching tree and an efficient pricing problem structure.

Let us now turn to the vaccination centers problem. Recall that this problem features an upstream facility location structure with a downstream resource allocation structure, leading to a much looser linear relaxation of the set-partitioning formulation. As a result, column generation leaves large optimality gaps, up to 40-50%. Restoring integrality via branch-and-price is therefore more challenging. Interestingly, the master problem becomes much more challenging and ends up being more time-consuming than the pricing problem—therefore highlighting the joint complexities of the problem arising from combinatorial optimization and non-linear system dynamics. Yet, the branch-and-price algorithm leads to a significant gap reduction: it returns an optimal solution in the "easy" two-facility instance, a near-optimal solution in the "hard" two-facility instances, and still moderate optimality gaps in the three-facility cases (2-4% versus 20-40% with column generation). In summary, the column generation heuristic is not sufficient in the vaccination centers problem but the branch-and-price algorithm provides critical solution improvements.

To shed more light into these findings, Table 4.4 reports results for the entire

country, broken down into the ten groups defined by the CDC. Recall that each group comprises 4 to 7 states, so the problem's complexity can vary significantly across groups. Nonetheless, our core observations are highly robust, in that the column generation algorithm leaves consistently large optimality gaps and the branch-and-price algorithm significantly improves the solution and tightens the gap. Ultimately, the branch-and-price algorithm results in an estimated 4,500 extra lives saved across the country over a 6-week period—a 16% improvement over the column generation algorithm.

## 4.3 Practical impact of the methodology

This thesis has proposed a general methodology to optimize critical resource allocation decisions in contagion systems—and in dynamical systems, more generally. A lingering question is whether this methodology provides sizeable benefits as compared to easily-implementable heuristics and existing methods from the literature. Tables 4.5–4.7 answer positively, by comparing our solution to various practical and technical benchmarks as well as a do-nothing baseline (i.e., no vaccines, or no vehicle intervention). To provide a fair assessment, all costs are estimated from the full continuous-state contagion models, as opposed to the state-clustering approximation.

Specifically, for the vaccine allocation and congestion mitigation problems we compare our methodology to the following practical benchmarks (we omit the content promotion problem in this section because these benchmarks do not have direct application due to the sparsity constraint):

- **uniform allocation**: each region receives the same amount of at each epoch.
- **myopic allocation**: at each epoch, each region receives a fraction of resources proportionally to the cost incurred in the next period under a do-nothing baseline—a greedy benchmark.
- **cost-based allocation**: each region receives a share of resources at each epoch proportionally to the total cost incurred across the planning horizon under a do-nothing baseline—a rule-based decision-making benchmark based on available

epidemiological information.

For the vaccination centers problem, we consider a "**Top** $F$" benchmark that selects the $F$ facilities that can serve the most people within access distance restrictions. We test it both with uniform vaccine allocations across the resulting $F$ facilities and with optimized vaccine allocations.

Last, we define three technical benchmarks in the vaccine allocation and vaccination centers cases:

- **direct implementation**: This benchmark directly implements Problem ($\mathcal{P}$) as mixed-integer bilinear optimization problem, using the latest release of (37).

- **discretization**: This benchmark approximates Problem ($\mathcal{P}$) via mixed-integer linear optimization, using time discretization to eliminate the continuous-time dynamics, and a staircase approximation of the infected population to handle the bilinearities. This benchmark provides an easily-implementable optimization approach, employed by (31) for example in a robust optimization setting. To optimize its performance, we divide the $[0, 2\%]$ interval into sub-intervals of length $\delta$ (since the infected population never exceeded 2% of the population) and loosen the termination criterion for acceleration. We consider coarser and more granular discretization of the infected population via values of $\delta = 0.001$ and $\delta = 0.002$.

- **coordinate descent**: This benchmark applies an iterative heuristic to solve Problem ($\mathcal{P}$), by alternatively optimizing vaccine allocations for a given number of infections and estimating the resulting number of infections for a given vaccine allocation plan (21).

Results show that our solution consistently outperforms all benchmarks, with significant practical benefits. Let us start with the vaccine allocation results (Table 4.5). In the absence of vaccinations, the pandemic would lead to an estimated 80,000 fatalities in region I of the United States over one month (our $n = 4$ setting), and to 650,000 fatalities in the full country over 3 months. Note that these numbers are adjusted to account for undetected deaths. Uniform vaccine allocations can prevent 0.13% of

Table 4.5: Performance comparison versus benchmarks (vaccine allocation).

| | | $n = 4,\ S = 4,\ D = 11$ | | $n = 4,\ S = 4,\ D = 21$ | |
|---|---|---|---|---|---|
| Method | Tolerance | Time (sec.) | Deaths | Time (sec.) | Deaths |
| Do nothing | — | — | 79.98K | — | 79.98K |
| Uniform allocation | — | — | -0.11K | — | -0.11K |
| Myopic allocation | — | — | -0.11K | — | -0.11K |
| Cost-based allocation | — | — | -0.11K | — | -0.11K |
| Direct | — | 1,000* | -0.22K | 1,000* | -0.22K |
| Discretization | $\delta = 0.002$ | 13.8 | -0.24K | 19.5 | -0.24K |
| Discretization | $\delta = 0.001$ | 131 | -0.24K | 110 | -0.24K |
| Coordinate descent | | 3.53 | **-0.25K** | 3.84 | **-0.25K** |
| Branch-and-price | $\varepsilon = 0.01$ | 1.25 | **-0.25K** | 3.34 | **-0.25K** |
| Branch-and-price | $\varepsilon = 0.002$ | 3.75 | **-0.25K** | 9.13 | **-0.25K** |
| Branch-and-price | Enumeration | 20.53 | **-0.25K** | 257 | **-0.25K** |

| | | $n = 51,\ S = 6,\ D = 11$ | | $n = 51,\ S = 6,\ D = 21$ | |
|---|---|---|---|---|---|
| Method | Tolerance | Time (sec.) | Deaths | Time (sec.) | Deaths |
| Do nothing | — | — | 573.37K | — | 573.37K |
| Uniform allocation | — | — | -5.91K | — | -5.91K |
| Myopic | — | — | -6.47K | — | -6.47K |
| Cost-based | — | — | -6.59K | — | -6.59K |
| Direct | — | n/a | n/a | n/a | n/a |
| Discretization | $\delta = 0.002$ | 1,000* | -5.54K | 1,000* | -5.03K |
| Discretization | $\delta = 0.001$ | n/a | n/a | 1,000* | -5.56K |
| Coordinate Descent | | 5.17 | **-11.02K** | 5.43 | **-11.36K** |
| Branch-and-price | $\varepsilon = 0.01$ | 12.5 | -10.94K | 26.1 | -11.10K |
| Branch-and-price | $\varepsilon = 0.002$ | 65.4 | **-11.05K** | 426.2 | **-11.28K** |
| Branch-and-price | Enumeration | n/a | n/a | n/a | n/a |

| | | $n = 51,\ S = 10,\ D = 21$ | | $n = 51,\ S = 12,\ D = 21$ | |
|---|---|---|---|---|---|
| Method | Tolerance | Time (sec.) | Deaths | Time (sec.) | Deaths |
| Do nothing | — | — | 629.40K | — | 649.75K |
| Uniform | — | — | -16.40K | — | -21.42K |
| Myopic | — | — | -19.77K | — | -26.88K |
| Cost-based | — | — | -20.28K | — | -27.64K |
| Direct | — | n/a | n/a | n/a | n/a |
| Discretization | $\delta = 0.002$ | n/a | n/a | n/a | n/a |
| Discretization | $\delta = 0.001$ | n/a | n/a | n/a | n/a |
| Coordinate descent | | 5.17 | **-30.16K** | 6.03 | -39.01K |
| Branch-and-price | $\varepsilon = 0.01$ | 82.44 | -29.30K | 136.3 | -38.91K |
| Branch-and-price | $\varepsilon = 0.002$ | 2466.2 | **-30.21K** | 3671.4 | **-39.41K** |
| Branch-and-price | Enumeration | n/a | n/a | n/a | n/a |

$*$ and "n/a": the algorithm does not return an optimal and feasible solution, respectively, within the time limit.
Bold fonts indicate solutions within 1% of the best-found solution in terms of number of deaths.

Table 4.6: Performance comparison versus benchmarks (vaccination centers, full country, $S = 6$).

| | | | $F = 2$ | | $F = 3$ | |
|---|---|---|---|---|---|---|
| Facility Selection | Allocation | Tolerance | Time (sec.) | Deaths | Time (sec.) | Deaths |
| — | No vaccine | — | — | 573.37K | — | 573.37K |
| Top F | Uniform | — | — | -6.3K | — | -6.98K |
| Top F | Optimized | — | — | -7.99K | — | -7K |
| Discretization | | $\delta = 0.001$ | 10,372* | -4.5K | 10,275* | -4.3K |
| Direct | | — | n/a | n/a | n/a | n/a |
| Coordinate Descent | | $\epsilon = 0.01$ | 206 | -8.2K | 254 | -7.7K |
| Branch-and-price | | $\epsilon = 0.01$ | 11,072 | **-8.3K** | 10,978 | **-8.6K** |

$*$ and "n/a": the algorithm does not return an optimal and feasible solution, respectively, within the time limit.
Bold fonts indicate solutions within 1% of the best-found solution in terms of number of deaths.

Table 4.7: Performance comparison versus benchmarks (congestion mitigation, $D_x = 6$, $D_y = 4$).

| | $S = 4$ | | $S = 6$ | |
|---|---|---|---|---|
| Method | Time (sec.) | Congestion (%) | Time (sec.) | Congestion (%) |
| Do nothing | — | 14.736 | — | 18.976 |
| Uniform | — | -1.077% | — | -1.797% |
| Myopic | — | -0.988% | — | -1.705% |
| Cost-based | — | -1.084% | — | -1.791% |
| Branch-and-price | 42 | **-2.160%** | 443 | **-3.316%** |

$*$ and "n/a": the algorithm does not return an optimal and feasible solution, respectively, within the time limit.
Bold fonts indicate the best-found solution.

these fatalities in small instances (110 vs. 80,000) and 3.3% in large instances (21,000 vs. 650,000). The myopic and cost-based benchmarks save up to an extra 30% of lives, by allowing for temporal re-optimization and accounting for epidemiological information, respectively. Yet, our solution provides significant benefits, by increasing the number of saved lines by over 100% in small instances and by 50-70% in larger instances. Stated differently, without increasing vaccine capacity or vaccine efficacy, an optimized management of a vaccine stockpile can increase the effectiveness of the vaccination campaign by a factor of 1.5 to 1.7. In absolute terms, these improvements represent an increase from 27,000 saved lives (or 4.1% of all fatalities) to nearly 39,000 lives (or 6.0%). These results demonstrate the role of vaccine distribution as a critical lever to combat an epidemic and the edge of optimization to guide resource allocation in complex and non-linear contagion systems over simple epidemiological proxies.

For robustness, we ran experiments with a larger vaccine budget—specifically, with 1 million vaccines a day as opposed to 2.5 million vaccines a week. This experiment captures the vaccine supply that was actually available during the period under consideration, as opposed to the supply that was planned at the beginning of the horizon. Results are reported in Appendix B, and confirm the benefits of our optimization methodology to guide spatial-temporal vaccine allocation. One difference is that, as vaccine supply becomes less constrained, the benefits of optimization become smaller. Nevertheless, our methodology still consistently generates a $12 - 25\%$ improvement compared with the strongest benchmark, resulting in an extra 5,000 lives saved.

We obtain similar findings in the other two problems. In the congestion mitigation problem *Table 4.7), the relative benefits of optimization are even more significant, estimated at 80-160% against cost-based allocation. In vaccination centers problem (Table 4.6), the optimized solution also provides significant improvements against the simple benchmark that merely consists of selecting the most attractive facility locations. Specifically, the benefit of optimization amounts to 23-31% under uniform vaccine allocations and to 3-23% under optimized vaccine allocations. Exactly like epidemiological proxies are not sufficient to guide resource allocation, simple demo-

graphic proxies are not sufficient to guide strategic planning in complex contagion systems.

Next, these results underscore the impact of our methodology to reap the benefits of optimization. First, our state-clustering dynamic programming algorithm is instrumental to reliably generate high-quality vaccine allocation solutions in manageable computational times. Indeed, the exhaustive state space enumeration does not scale beyond small instances, whereas our state-clustering algorithm enables the branch-and-price methodology to terminate in minutes in the largest instance (full country, 3 months). Most importantly, our methodology significantly outperforms simpler optimization benchmarks. Direct implementation of the mixed-integer bilinear problem does not even terminate for small instances. The discretization benchmark generates near-optimal results in small instances of the vaccine allocation problem, but solution quality drops significantly thereafter. In fact, this benchmark fails to even outperform uniform allocation in medium instances, and fails to return a feasible solution in large instances of the vaccine allocation problem and for the vaccination centers problem. This finding demonstrates the limitations of simple workarounds to circumvent the non-convexities of contagion dynamics in an optimization framework— coarse discretization leads to extensive computation time and granular discretization leads to poor-quality solutions at just medium scales. In comparison, our algorithm achieves a Pareto improvement over this benchmark: faster computational times, stronger scalability, and better practical performance. Finally, our methodology can also outperform the coordinate descent heuristic. For instance, it yields a 1% improvement in the largest vaccine allocation instance. In the vaccination centers problem, the benefits can be much more significant when local optimal induced by the coordinate descent heuristic leads to a different set of selected facilities. In the $F = 3$ instance, our methodology results in an extra 900 lives saved over a 6-week horizon—a 12% improvement.

We conclude by illustrating the drivers of vaccine allocation in Figure 4-1. As expected, the states where the pandemic is more costly (e.g., those with a larger population, more infections) generally receive more vaccines. Yet, the relationship

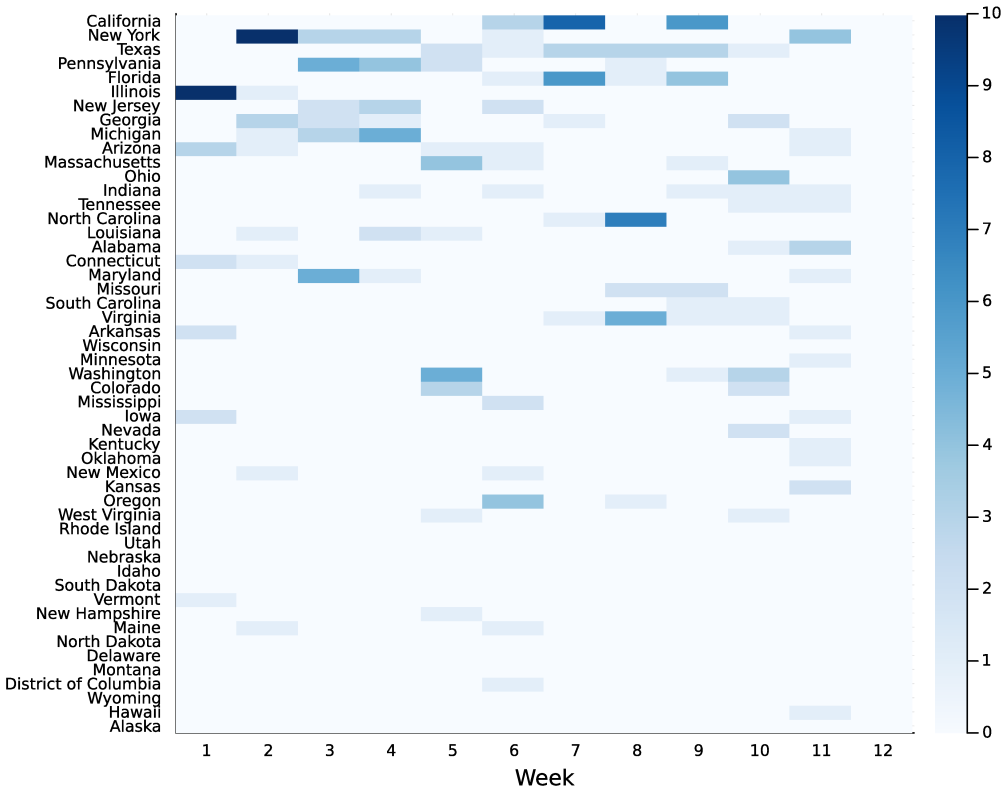


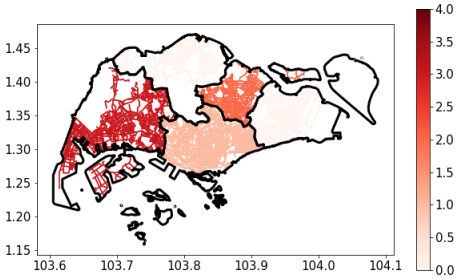

Figure 4-1: Spatial-temporal vaccine allocation over a 12-week horizon. States are ordered from top to bottom in decreasing order of the total cost under no vaccine—a proxy for the prevalence of the pandemic.

is not monotonic. Moreover, vaccine allocations exhibit different temporal patterns; for instance, New York and Illinois receive most of their vaccines at the beginning of the horizon; California and Texas receive most of their vaccines later on; and Florida stands in between. In other words, there is no one-size-fits-all allocation strategy. Instead, the benefits of optimization stem from fine-tuning vaccine allocations based on the spatial-temporal dynamics of the epidemic, alternating between treatment interventions to prevent fatalities where the pandemic is prevalent and prevention interventions to minimize its future growth.

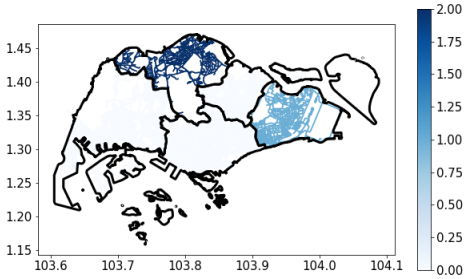
Similarly, Figure 4-2 reports the spatial-temporal allocation of treatment and prevention vehicles in Singapore. Note that treatment vehicles are mainly sent to central regions that concentrate on dense business centers and the airport, whereas prevention vehicles are primarily sent to suburban regions. Intuitively, this solution prioritizes clearing large accidents in denser areas to ease congestion recovery, while preventing

the spread of congestion to other areas. Again, the solution adopts different strategies depending on the contagion dynamics in each region.

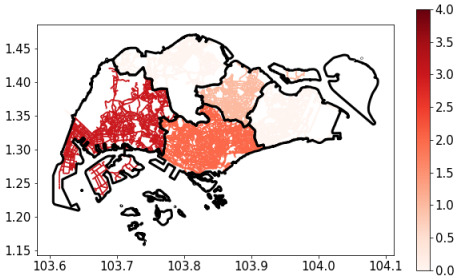
In summary, the benefits of our methodology stem from our set-partitioning formulation to model non-convex system dynamics, of our column generation algorithm with the state-clustering dynamic programming algorithm to enable scalability to large instances, and of our tri-partite branch-and-price algorithm to solve the discrete non-convex decision-making problem to near-optimality. Ultimately, this methodology generates consistently high-quality solutions across problem instances, outperforming all benchmarks and yielding practical benefits across dynamic systems.



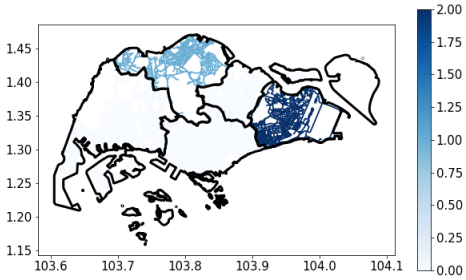

(a) Treatment vehicles, s=1



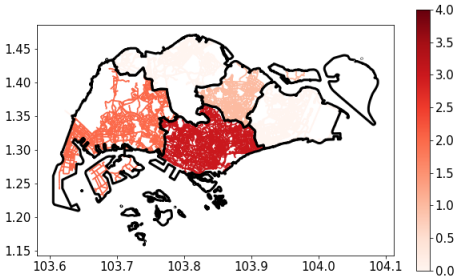

(b) Prevention vehicles, s=1



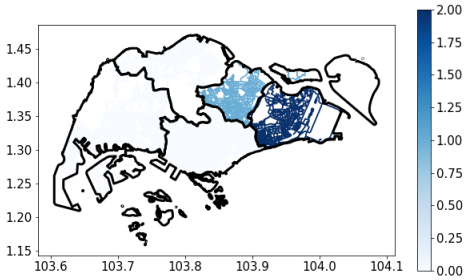

(c) Treatment vehicles, s=2



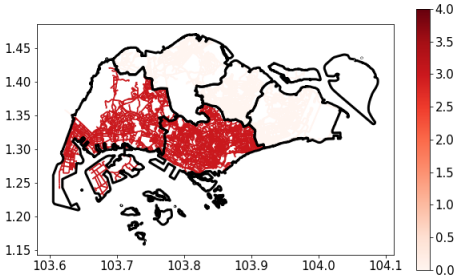

(d) Prevention vehicles, s=1



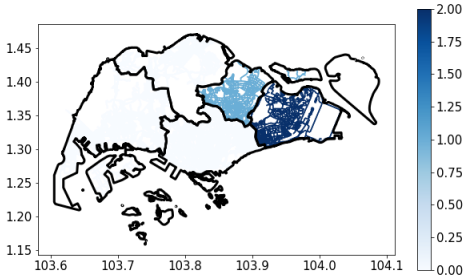

(e) Treatment vehicles, s=3




(f) Prevention vehicles, s=1




(g) Treatment vehicles, s=4




(h) Prevention vehicles, s=1

Figure 4-2: Allocation of treatment vehicles (left) and prevention vehicles (right) in Singapore over a 4-hour horizon.

# Chapter 5

# Conclusion

Predictive contagion systems have shown considerable success in epidemiology and other domains of science, engineering, and management. This thesis developed a prescriptive algorithm to optimize spatial-temporal resource allocation decisions in systems governed by contagion dynamics—and in more general dynamical systems. This class of problems, however, combines the difficulties of combinatorial optimization and those of dynamical systems—namely, continuous-time dynamics, non-linear system dynamics (e.g., bilinear interactions between susceptible and infected populations), and possibly non-linear effects of interventions. In response, this thesis developed a branch-and-price algorithm for prescriptive contagion analytics, using (i) a set partitioning reformulation to eliminate non-linearities via plan-based variables; (ii) column generation to separate the combinatorial optimization components in a restricted master problem from the non-linear dynamics in a pricing problem; (iii) a novel state-clustering algorithm for discrete-decision continuous-state dynamic programming to solve the pricing problem; and (iv) a tri-partite branching scheme to circumvent the non-linearities without branching on plan-based variables.

We implemented the proposed methodology on four prescriptive contagion problems: vaccine allocation, deployment of mass vaccination centers, online content promotion, and traffic congestion mitigation. The branch-and-price algorithm scales to realistic instances, for instance to problems with 20 decisions, 50 regions, and 12 de-

cision epochs, hence $\mathcal{O}(20^{600})$ possible decisions. From a practical standpoint, the methodology significantly outperforms easily-implementable benchmarks based on demographic or epidemiological proxies, as well as simpler discretization-based optimization approaches. In the vaccine allocation context, the prescriptive contagion analytics method developed in this thesis can result in an extra 30-70% extra lives saved in a situation mirroring the midst of the COVID-19 pandemic in the United States, therefore enhancing the effectiveness of vaccination campaigns without increasing vaccine capacity or vaccine efficacy. Ultimately, our prescriptive contagion analytics approach can deliver significant benefits by fine-tuning resource allocations based on spatial-temporal system dynamics.

# Bibliography


[1] Daron Acemoglu, Victor Chernozhukov, Iván Werning, and Michael D Whinston. Optimal targeted lockdowns in a multigroup sir model. *American Economic Review: Insights*, 3(4):487–502, 2021.

[2] David Adam. Special report: The simulations driving the world's response to covid-19. *Nature*, 580(7802):316–319, 2020.

[3] Shipra Agrawal, Steven Yin, and Assaf Zeevi. Dynamic pricing and learning under the bass model. In *Proceedings of the 22nd ACM Conference on Economics and Computation*, pages 2–3, 2021.

[4] Andres Alban, Philippe Blaettchen, Harwin de Vries, and Luk N Van Wassenhove. Resource allocation with sigmoidal demands: Mobile healthcare units and service adoption. *Manufacturing & Service Operations Management*, 24(6):2944–2961, 2022.

[5] Andrew Allman and Qi Zhang. Branch-and-price for a class of nonconvex mixed-integer nonlinear programs. *Journal of Global Optimization*, 81(4):861–880, 2021.

[6] Fernando Alvarez, David Argente, and Francesco Lippi. A simple planning problem for covid-19 lock-down, testing, and tracing. *American Economic Review: Insights*, 3(3):367–82, 2021.

[7] April K Andreas, J Cole Smith, and Simge Küçükyavuz. Branch-and-price-and-cut algorithms for solving the reliable h-paths problem. *Journal of Global Optimization*, 42(4):443–466, 2008.

[8] Ioannis P Androulakis, Costas D Maranas, and Christodoulos A Floudas. $\alpha$bb: A global optimization method for general constrained nonconvex problems. *Journal of Global Optimization*, 7:337–363, 1995.

[9] Kurt M Anstreicher. Semidefinite programming versus the reformulation-linearization technique for nonconvex quadratically constrained quadratic programming. *Journal of Global Optimization*, 43:471–484, 2009.

[10] Kurt M Anstreicher and Samuel Burer. Quadratic optimization with switching variables: the convex hull for n= 2. *Mathematical Programming*, 188(2):421–441, 2021.

[11] Rocío Balderrama, Javier Peressutti, Juan Pablo Pinasco, Federico Vazquez, and Constanza Sánchez de la Vega. Optimal control for a sir epidemic model with limited quarantine. *Scientific Reports*, 12(1):1–26, 2022.

[12] Cynthia Barnhart, Ellis L Johnson, George L Nemhauser, Martin WP Savelsbergh, and Pamela H Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations research*, 46(3):316–329, 1998.

[13] Frank M Bass. A new product growth for model consumer durables. *Management science*, 15(5):215–227, 1969.

[14] Doris A Behrens, Jonathan P Caulkins, Gernot Tragler, and Gustav Feichtinger. Optimal control of drug epidemics: prevent and treat—but not at the same time? *Management Science*, 46(3):333–347, 2000.

[15] Amine Bennouna, Joshua Joseph, David Nze-Ndong, Georgia Perakis, Divya Singhvi, Omar Skali Lami, Yannis Spantidakis, Leann Thayaparan, and Asterios Tsiourvas. Covid-19: Prediction, prevalence, and the operations of vaccine allocation. *Manufacturing & Service Operations Management*, 2022.

[16] T Berge, JM-S Lubuma, GM Moremedi, Neil Morris, and R Kondera-Shava. A simple mathematical model for ebola in africa. *Journal of biological dynamics*, 11(1):42–74, 2017.

[17] Dimitri Bertsekas. Convergence of discretization procedures in dynamic programming. *IEEE Transactions on Automatic Control*, 20(3):415–419, 1975.

[18] Dimitri P Bertsekas. Dynamic programming and optimal control 4th edition, volume ii. *Athena Scientific*, 2015.

[19] Dimitri P Bertsekas, David A Castanon, et al. Adaptive aggregation methods for infinite horizon dynamic programming. 1988.

[20] Dimitris Bertsimas, Leonard Boussioux, Ryan Cory-Wright, Arthur Delarue, Vassilis Digalakis, Alexandre Jacquillat, Driss Lahlou Kitane, Galit Lukin, Michael Li, Luca Mingardi, et al. From predictions to prescriptions: A data-driven response to covid-19. *Health care management science*, 24(2):253–272, 2021.

[21] Dimitris Bertsimas, Vassilis Digalakis Jr, Alexander Jacquillat, Michael Lingzhi Li, and Alessandro Previero. Where to locate covid-19 mass vaccination facilities? *Naval Research Logistics (NRL)*, 69(2):179–200, 2022.

[22] Felix Bestehorn, Christoph Hansknecht, Christian Kirches, and Paul Manns. Mixed-integer optimal control problems with switching costs: a shortest path approach. *Mathematical Programming*, 188(2):621–652, 2021.

[23] Conrado Borraz-Sánchez and Dag Haugland. Minimizing fuel cost in gas transmission networks by dynamic programming and adaptive discretization. *Computers & Industrial Engineering*, 61(2):364–372, 2011.

[24] Samuel Burer and Adam N Letchford. Non-convex mixed-integer nonlinear programming: A survey. *Surveys in Operations Research and Management Science*, 17(2):97–106, 2012.

[25] Renato Casagrandi, Luca Bolzoni, Simon A Levin, and Viggo Andreasen. The sirc model and influenza a. *Mathematical biosciences*, 200(2):152–169, 2006.

[26] Jonathan Caulkins, Dieter Grass, Gustav Feichtinger, Richard Hartl, Peter M Kort, Alexia Prskawetz, Andrea Seidl, and Stefan Wrzaczek. How long should the covid-19 lockdown continue? *Plos one*, 15(12):e0243413, 2020.

[27] Don M Chance, Eric Hillebrand, and Jimmy E Hilliard. Pricing an option on revenue from an innovation: An application to movie box office revenue. *Management Science*, 54(5):1015–1028, 2008.

[28] Koray Cosguner and PB Seetharaman. Dynamic pricing for new products using a utility-based generalization of the bass diffusion model. *Management Science*, 68(3):1904–1922, 2022.

[29] Jonas Dehning, Johannes Zierenberg, F Paul Spitzner, Michael Wibral, Joao Pinheiro

Neto, Michael Wilczek, and Viola Priesemann. Inferring change points in the spread of covid-19 reveals the effectiveness of interventions. *Science*, 369(6500):eabb9789, 2020.

[30] Iain Dunning, Joey Huchette, and Miles Lubin. JuMP: A modeling language for mathematical optimization. *SIAM review*, 59(2):295–320, 2017.

[31] Chenyi Fu, Melvyn Sim, and Minglong Zhou. Robust epidemiological analytics. *Available at SSRN 3869521*, 2021.

[32] Tetsuya Fujie and Masakazu Kojima. Semidefinite programming relaxation for nonconvex quadratic programs. *Journal of Global optimization*, 10:367–380, 1997.

[33] Steven Goldman and James Lightwood. Cost optimization in the sis model of infectious disease with treatment. *Topics in Economic Analysis & Policy*, 2(1):1007, 2002.

[34] Simone Göttlich, Falk M Hante, Andreas Potschka, and Lars Schewe. Penalty alternating direction methods for mixed-integer optimal control with combinatorial constraints. *Mathematical Programming*, 188(2):599–619, 2021.

[35] Lars Grüne and Willi Semmler. Using dynamic programming with adaptive grid scheme for optimal control problems in economics. *Journal of Economic Dynamics and Control*, 28(12):2427–2456, 2004.

[36] Oktay Günlük and Jeff Linderoth. Perspective reformulations of mixed integer nonlinear programs with indicator variables. *Mathematical programming*, 124:183–205, 2010.

[37] Gurobi. Non-convex quadratic optimization. https://www.gurobi.com/wp-content/uploads/2020-01-14_Non-Convex-Quadratic-Optimization-in-Gurobi-9.0-Webinar.pdf?x41151, 2019.

[38] Falk M Hante and Sebastian Sager. Relaxation methods for mixed-integer optimal control of partial differential equations. *Computational Optimization and Applications*, 55:197–225, 2013.

[39] Michael Hinze, René Pinnau, Michael Ulbrich, and Stefan Ulbrich. *Optimization with PDE constraints*, volume 23. Springer Science & Business Media, 2008.

[40] Teck-Hua Ho, Sergei Savin, and Christian Terwiesch. Managing demand and sales dynamics in new product diffusion under supply constraint. *Management science*, 48(2):187–206, 2002.

[41] Solomon Hsiang, Daniel Allen, Sébastien Annan-Phan, Kendon Bell, Ian Bolliger, Trinetta Chong, Hannah Druckenmiller, Luna Yue Huang, Andrew Hultgren, Emma Krasovich, et al. The effect of large-scale anti-contagion policies on the covid-19 pandemic. *Nature*, 584(7820):262–267, 2020.

[42] Michael N Jung, Gerhard Reinelt, and Sebastian Sager. The lagrangian relaxation for the combinatorial integral approximation problem. *Optimization Methods and Software*, 30(1):54–80, 2015.

[43] William Ogilvy Kermack and Anderson G McKendrick. A contribution to the mathematical theory of epidemics. *Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character*, 115(772):700–721, 1927.

[44] Andriy Koval, William Howard Beasley, Oleksandra Hararuk, and Joseph Lee Rodgers. Social contagion and general diffusion models of adolescent religious transitions: A tutorial, and emosa applications. *Journal of Research on Adolescence*, 33(1):318–343, 2023.

[45] Trichy V Krishnan and Dipak C Jain. Optimal dynamic advertising policy for new products. *Management Science*, 52(12):1957–1969, 2006.

[46] Sangbum Lee and Ignacio E Grossmann. A global optimization algorithm for nonconvex generalized disjunctive programming and applications to process systems. *Computers & Chemical Engineering*, 25(11-12):1675–1697, 2001.

[47] Hongmin Li. Optimal pricing under diffusion-choice models. *Operations Research*, 68(1):115–133, 2020.

[48] Hongmin Li, Dieter Armbruster, and Karl G Kempf. A population-growth model for multiple generations of technology products. *Manufacturing & Service Operations Management*, 15(3):343–360, 2013.

[49] Michael Lingzhi Li, Hamza Tazi Bouardi, Omar Skali Lami, Thomas A Trikalinos, Nikolaos Trichakis, and Dimitris Bertsimas. Forecasting covid-19 and analyzing the effect of government interventions. *Operations Research*, 2022.

[50] Yunduan Lin, Mengxin Wang, Zuo-Jun Max Shen, Heng Zhang, and Renyu Philip Zhang. Content promotion for online content platforms with network diffusion effect. *Available at SSRN 3863104*, 2021.

[51] Konstantinos Lolos, Ioannis Konstantinou, Verena Kantere, and Nectarios Koziris. Adaptive state space partitioning of markov decision processes for elastic resource management. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 191–194. IEEE, 2017.

[52] Elisa F Long, Eike Nohdurft, and Stefan Spinler. Spatial resource allocation for emerging epidemics: A comparison of greedy, myopic, and dynamic policies. *Manufacturing & Service Operations Management*, 20(2):181–198, 2018.

[53] Yunke Mai and Bin Hu. Optimizing free-to-play multiplayer games with premium subscription. *Management Science*, 2022.

[54] Paul Manns and Christian Kirches. Improved regularity assumptions for partial outer convexification of mixed-integer pde-constrained optimization problems. *ESAIM: Control, Optimisation and Calculus of Variations*, 26:32, 2020.

[55] Sanjay Mehrotra, Hamed Rahimian, Masoud Barah, Fengqiao Luo, and Karolina Schantz. A model of supply-chain decisions for resource sharing with an application to ventilator allocation to combat covid-19. *Naval Research Logistics (NRL)*, 67(5):303–320, 2020.

[56] Moderna. Storage & handling: Moderna covid-19 vaccine (eua). https://eua.modernatx.com/covid19vaccine-eua/providers/storage-handling, 2020.

[57] Martial L Ndeffo Mbah and Christopher A Gilligan. Resource allocation for epidemic control in metapopulations. *PLoS one*, 6(9):e24577, 2011.

[58] Ivo Nowak, Norman Breitfeld, Eligius MT Hendrix, and Grégoire Njacheun-Njanzoua. Decomposition-based inner-and outer-refinement algorithms for global optimization. *Journal of Global Optimization*, 72(2):305–321, 2018.

[59] Warren B Powell. *Reinforcement Learning and Stochastic Optimization: A Unified Framework for Sequential Decisions*. John Wiley & Sons, 2022.

[60] Larry D Pyeatt, Adele E Howe, et al. Decision tree function approximation in reinforcement learning. In *Proceedings of the third international symposium on adaptive systems: evolutionary computation and probabilistic graphical models*, volume 2, pages 70–77. Cuba, 2001.

[61] Hazhir Rahmandad, Rebecca Henderson, and Nelson P Repenning. Making the num-

bers?"short termism" and the puzzle of only occasional disaster. *Management Science*, 64(3):1328–1347, 2018.

[62] Hazhir Rahmandad and Zeynep Ton. If higher pay is profitable, why is it so rare? modeling competing strategies in mass market services. *Organization Science*, 31(5):1053–1071, 2020.

[63] Evan L Ray, Nutcha Wattanachit, Jarad Niemi, Abdul Hannan Kanji, Katie House, Estee Y Cramer, Johannes Bracher, Andrew Zheng, Teresa K Yamana, Xinyue Xiong, et al. Ensemble forecasts of coronavirus disease 2019 (covid-19) in the us. *MedRXiv*, 2020.

[64] Robert E Rowthorn, Ramanan Laxminarayan, and Christopher A Gilligan. Optimal control of epidemics in metapopulations. *Journal of the Royal Society Interface*, 6(41):1135–1144, 2009.

[65] Hong S Ryoo and Nikolaos V Sahinidis. A branch-and-reduce approach to global optimization. *Journal of global optimization*, 8:107–138, 1996.

[66] Meead Saberi, Homayoun Hamedmoghadam, Mudabber Ashfaq, Seyed Amir Hosseini, Ziyuan Gu, Sajjad Shafiei, Divya J Nair, Vinayak Dixit, Lauren Gardner, S Travis Waller, et al. A simple contagion process describes spreading of traffic jams in urban networks. *Nature communications*, 11(1):1–9, 2020.

[67] Sebastian Sager. *Numerical methods for mixed-integer optimal control problems*. Der Andere Verlag Lübeck, 2005.

[68] Sebastian Sager, Hans Georg Bock, and Moritz Diehl. The integer approximation error in mixed-integer optimal control. *Mathematical programming*, 133(1-2):1–23, 2012.

[69] Anureet Saxena, Pierre Bonami, and Jon Lee. Convex relaxations of non-convex mixed integer quadratically constrained programs: extended formulations. *Mathematical programming*, 124(1-2):383–411, 2010.

[70] Anureet Saxena, Pierre Bonami, and Jon Lee. Convex relaxations of non-convex mixed integer quadratically constrained programs: projected formulations. *Mathematical programming*, 130:359–413, 2011.

[71] Devavrat Shah and Tauhid Zaman. Finding rumor sources on random trees. *Operations research*, 64(3):736–755, 2016.

[72] Wenjing Shen, Izak Duenyas, and Roman Kapuscinski. Optimal pricing, production, and inventory for new product diffusion under supply constraints. *Manufacturing & Service Operations Management*, 16(1):28–45, 2014.

[73] Hanif D Sherali and Warren P Adams. *A reformulation-linearization technique for solving discrete and continuous nonconvex problems*, volume 31. Springer Science & Business Media, 2013.

[74] Sean R Sinclair, Siddhartha Banerjee, and Christina Lee Yu. Adaptive discretization in online reinforcement learning. *Operations Research*, 2022.

[75] Daniel A Sprague and Thomas House. Evidence for complex contagion models of social contagion from observational data. *PloS one*, 12(7):e0180802, 2017.

[76] John Sterman, Thomas Fiddaman, Travis Read Franck, Andrew Jones, Stephanie McCauley, Philip Rice, Elizabeth Sawin, and Lori Siegel. Climate interactive: the c-roads climate policy model. *System Dynamics Review*, 28(3):295–305, 2012.

[77] Anjana Susarla, Jeong-Ha Oh, and Yong Tan. Social networks and the diffusion of user-

generated content: Evidence from youtube. *Information systems research*, 23(1):23–41, 2012.

[78] Mohit Tawarmalani and Nikolaos V Sahinidis. Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. *Mathematical programming*, 99(3):563–591, 2004.

[79] Patrick GT Walker, Charles Whittaker, Oliver J Watson, Marc Baguelin, Peter Winskill, Arran Hamlet, Bimandra A Djafaara, Zulma Cucunubá, Daniela Olivera Mesa, Will Green, et al. The impact of covid-19 and strategies for mitigation and suppression in low-and middle-income countries. *Science*, 369(6502):413–422, 2020.

[80] Dan Yamin and Arieh Gavious. Incentives' effect in influenza vaccination policy. *Management Science*, 59(12):2667–2686, 2013.

[81] Jiding Zhang, Sergei Savin, and Senthil Veeraraghavan. Revenue management in crowdfunding. *Manufacturing & Service Operations Management*, 2022.

[82] Mengzhenyu Zhang, Hyun-Soo Ahn, and Joline Uichanco. Data-driven pricing for a new product. *Operations Research*, 70(2):847–866, 2022.

# Appendix A

# Additional Tables for section 4.2

Those tables are additional results for discussion in sections 4.2 and 4.3. The tables presented in those sections are drawn from the tables in this appendix. The same comments can be made for the rest of the other instances so only snippets of them were added to the main text for conciseness.

Table A.1: Performance evaluation of the branch-and-price algorithm for the vaccine allocation problem.

| | | | | | CPU times (s) | | | | Solution quality | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | $S$ | $D$ | Method | Nodes | Init. | RMP | PP | Alg. | UB | Solution | Gap |
| 51 | 6 | 5 | CG | 1 | 15 | 0.02 | 0.14 | 0.2 | 45,995 | 45,563 | 0.94% |
| | | | 2-BP | 14 | 15 | 0.11 | 0.69 | 5.25 | 45,821 | 45,780 | 0.09% |
| | | | 3-BP | 14 | 15 | 0.12 | 0.59 | 5.61 | 45,821 | 45,780 | 0.09% |
| 51 | 6 | 10 | CG | 1 | 27 | 0.02 | 0.55 | 0.61 | 48,240 | 48,108 | 0.27% |
| | | | 2-BP | 14 | 27 | 0.14 | 1.24 | 5.94 | 48,208 | 48,208 | 0% |
| | | | 3-BP | 14 | 27 | 0.17 | 1.5 | 7.01 | 48,208 | 48,208 | 0% |
| 51 | 6 | 20 | CG | 1 | 58 | 0.01 | 0.75 | 0.8 | 50,649 | 50,143 | 1% |
| | | | 2-BP | 9,990 | 57 | 52 | 710 | 1,249 | 50,542 | 50,437 | 0.21% |
| | | | 3-BP | 9,990 | 57 | 51 | 709 | 1,250 | 50,542 | 50,437 | 0.21% |
| 51 | 8 | 5 | CG | 1 | 21 | 0.02 | 0.22 | 0.29 | 93,694 | 92,261 | 1.53% |
| | | | 2-BP | 28 | 21 | 0.27 | 1.43 | 7.11 | 93,378 | 93,263 | 0.12% |
| | | | 3-BP | 46 | 21 | 0.42 | 2.22 | 9.62 | 93,353 | 93,263 | 0.09% |
| 51 | 8 | 10 | CG | 1 | 34 | 0.02 | 0.58 | 0.68 | 99,296 | 98,801 | 0.5% |
| | | | 2-BP | 732 | 34 | 7.8 | 52 | 137 | 99,081 | 98,969 | 0.11% |
| | | | 3-BP | 744 | 34 | 7.4 | 53 | 139 | 99,068 | 98,969 | 0.1% |
| 51 | 8 | 20 | CG | 1 | 87 | 0.02 | 2.04 | 2.13 | 103,992 | 103,668 | 0.31% |
| | | | 2-BP | 1,868 | 87 | 20 | 346 | 557 | 103,771 | 103,668 | 0.1% |
| | | | 3-BP | 1,868 | 87 | 20 | 346 | 555 | 103,771 | 103,668 | 0.1% |
| 51 | 10 | 5 | CG | 1 | 23 | 0.02 | 0.31 | 0.38 | 146,281 | 145,698 | 0.4% |
| | | | 2-BP | 16 | 23 | 0.17 | 1.39 | 6.22 | 145,947 | 145,810 | 0.09% |
| | | | 3-BP | 16 | 23 | 0.18 | 1.4 | 6.32 | 145,947 | 145,810 | 0.09% |
| 51 | 10 | 10 | CG | 1 | 43 | 0.02 | 0.81 | 0.92 | 156,464 | 155,624 | 0.54% |
| | | | 2-BP | 4,778 | 43 | 53 | 426 | 1,064 | 156,183 | 156,026 | 0.1% |
| | | | 3-BP | 4,778 | 43 | 55 | 420 | 1058 | 156,183 | 156,026 | 0.1% |
| 51 | 10 | 20 | CG | 1 | 118 | 0.02 | 2.55 | 2.78 | 164,927 | 163,111 | 1.1% |
| | | | 2-BP | 9,990 | 118 | 62 | 1,262 | 2,495 | 164,698 | 164,065 | 0.38% |
| | | | 3-BP | 9,990 | 118 | 62 | 1,251 | 2,486 | 164,698 | 164,065 | 0.38% |
| 51 | 12 | 5 | CG | 1 | 27 | 0.03 | 0.5 | 0.6 | 198,566 | 196,855 | 0.86% |
| | | | 2-BP | 1,224 | 27 | 17 | 63 | 238 | 197,468 | 197,272 | 0.01% |
| | | | 3-BP | 1,302 | 27 | 19 | 68 | 257 | 197,468 | 197,272 | 0.01% |
| 51 | 12 | 10 | CG | 1 | 51 | 0.03 | 0.95 | 1.04 | 212,653 | 211,412 | 0.58% |
| | | | 2-BP | 122 | 51 | 1.37 | 18 | 35 | 212,027 | 211,978 | 0.02% |
| | | | 3-BP | 122 | 51 | 1.33 | 18 | 35 | 212,027 | 211,978 | 0.02% |
| 51 | 12 | 20 | CG | 1 | 144 | 0.03 | 4.1 | 4.32 | 225,323 | 222,592 | 1.21% |
| | | | 2-BP | 9,990 | 144 | 101 | 1,765 | 3,475 | 225,048 | 224,495 | 0.25% |
| | | | 3-BP | 9,990 | 144 | 103 | 1,808 | 3,535 | 225,048 | 224,495 | 0.25% |

Table A.2: Performance evaluation of the branch-and-price algorithm for vaccination centers problem.

| | | | | | | | | CPU times (s) | | | | Solution quality | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Group | n | S | D | F | Flexibility | Method | Nodes | Init. | RMP | PP | Alg. | UB | Solution | Gap |
| A | 7 | 6 | 12 | 2 | 0 | CG | 1 | 58 | 0.12 | 0.33 | 0.96 | 7,224 | 6,109 | 18.66% |
| | | | | | | 3-BP | 36,017 | 58 | 794 | 440 | 10,813 | 7,094 | 6,665 | 6.05% |
| A | 7 | 6 | 12 | 2 | 1 | CG | 1 | 46 | 0.14 | 0.42 | 1.06 | 8,200 | 7,134 | 18.52% |
| | | | | | | 3-BP | 13,683 | 45 | 228 | 199 | 10,811 | 8,134 | 7,917 | 2.66% |
| A | 7 | 6 | 12 | 3 | 0 | CG | 1 | 46 | 0.18 | 0.27 | 0.86 | 8,200 | 6,703 | 39.32% |
| | | | | | | 3-BP | 16,586 | 45 | 383 | 241 | 10,839 | 7,867 | 7,511 | 4.53% |
| A | 7 | 6 | 12 | 3 | 1 | CG | 1 | 46 | 0.18 | 0.46 | 1.12 | 8,200 | 6,822 | 40.82% |
| | | | | | | 3-BP | 13,451 | 45 | 294 | 194 | 11,072 | 8,059 | 7,726 | 4.14% |
| A | 7 | 8 | 12 | 2 | 0 | CG | 1 | 79 | 0.30 | 1.34 | 2.17 | 16,562 | 12,471 | 48.54% |
| | | | | | | 3-BP | 24,453 | 78 | 1,138 | 701 | 10,827 | 16,381 | 15,842 | 3.29% |
| A | 7 | 8 | 12 | 2 | 1 | CG | 1 | 54 | 0.33 | 1.01 | 2.02 | 18,662 | 16,417 | 17.69% |
| | | | | | | 3-BP | 33,034 | 53 | 979 | 754 | 10,803 | 18,628 | 18,443 | 0.99% |
| A | 7 | 8 | 12 | 3 | 0 | CG | 1 | 54 | 0.41 | 1.02 | 1.87 | 18,652 | 14,619 | 27.66% |
| | | | | | | 3-BP | 8,638 | 53 | 289 | 305 | 10,891 | 17,740 | 16,929 | 4.57% |
| A | 7 | 8 | 12 | 3 | 1 | CG | 1 | 53 | 0.45 | 1.05 | 1.93 | 18,662 | 14,918 | 31.59% |
| | | | | | | 3-BP | 12,287 | 53 | 343 | 375 | 10,939 | 18,274 | 17,187 | 5.94% |
| H | 6 | 6 | 4 | 2 | 0 | CG | 1 | 49 | 0.03 | 0.04 | 0.55 | 786 | 652 | 16.97% |
| | | | | | | 3-BP | 4,678 | 49 | 147 | 24 | 1,026 | 764 | 763 | 0.09% |
| H | 6 | 6 | 4 | 2 | 1 | CG | 1 | 39 | 0.03 | 0.03 | 0.46 | 786 | 652 | 55.49% |
| | | | | | | 3-BP | 1,936 | 39 | 46 | 10 | 351 | 764 | 763 | 0.09% |
| H | 6 | 6 | 4 | 3 | 0 | CG | 1 | 42 | 0.03 | 0.02 | 0.48 | 678 | 652 | 3.75% |
| | | | | | | 3-BP | 45,144 | 42 | 1,278 | 113 | 10,802 | 675 | 674 | 0.23% |
| H | 6 | 6 | 4 | 3 | 1 | CG | 1 | 45 | 0.03 | 0.03 | 0.55 | 678 | 652 | 3.75% |
| | | | | | | 3-BP | 53,590 | 44 | 1,417 | 118 | 10,802 | 675 | 674 | 0.23% |
| H | 6 | 8 | 4 | 2 | 0 | CG | 1 | 45 | 0.04 | 0.08 | 0.68 | 1,978 | 1,671 | 15.55% |
| | | | | | | 3-BP | 62 | 44 | 1.43 | 0.89 | 6.99 | 1,978 | 1,978 | 0.00% |
| H | 6 | 8 | 4 | 2 | 1 | CG | 1 | 42 | 0.08 | 0.07 | 0.58 | 1,978 | 1,753 | 49.08% |
| | | | | | | 3-BP | 20,280 | 41 | 2,002 | 76 | 10,818 | 1,886 | 1,865 | 1.11% |
| H | 6 | 8 | 4 | 3 | 0 | CG | 1 | 43 | 0.05 | 0.05 | 0.53 | 1791 | 1780 | 0.60% |
| | | | | | | 3-BP | 14,149 | 42 | 298 | 71 | 1,835 | 1,787 | 1,787 | 0.05% |
| H | 6 | 8 | 4 | 3 | 1 | CG | 1 | 45 | 0.03 | 0.05 | 0.54 | 1,791 | 1,780 | 0.60% |
| | | | | | | 3-BP | 11,927 | 45 | 231 | 63 | 1,506 | 1,787 | 1,787 | 0.05% |

Table A.3: Performance evaluation of the branch-and-price algorithm for the online promotion problem.

| | | | | | | CPU times (s) | | | | Solution quality | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | $S$ | $X$ | $Y$ | Method | Nodes | Init. | RMP | PP | Alg. | UB | Solution | Gap |
| 20 | 6 | 10 | 4 | CG | 1 | 49 | 0.03 | 2.17 | 2.35 | 1.015 | 1.004 | 1.06% |
| | | | | 2-BP | 8 | 49 | 0.08 | 4.04 | 5.9 | 1.015 | 1.015 | 0% |
| | | | | 3-BP | 8 | 49 | 0.09 | 4.11 | 6.1 | 1.015 | 1.015 | 0% |
| 20 | 6 | 10 | 6 | CG | 1 | 56 | 0.02 | 1.76 | 2.01 | 1.017 | 1.007 | 0.96% |
| | | | | 2-BP | 25,958 | 56 | 417 | 2,430 | 7,200 | 1.017 | 1.016 | 0.1% |
| | | | | 3-BP | 26,194 | 56 | 426 | 2,407 | 7,201 | 1.017 | 1.016 | 0.1% |
| 20 | 6 | 20 | 2 | CG | 1 | 79 | 0.04 | 4.62 | 4.84 | 1.017 | 0.978 | 3.9% |
| | | | | 2-BP | 696 | 79 | 9.35 | 198 | 327 | 1.015 | 1.009 | 0.53% |
| | | | | 3-BP | 142 | 79 | 1.67 | 52 | 74 | 1.013 | 1.013 | 0.09% |
| 20 | 6 | 20 | 4 | CG | 1 | 72 | 0.03 | 3.77 | 4.06 | 1.052 | 1.04 | 1.13% |
| | | | | 2-BP | 11,400 | 72 | 179 | 1,861 | 7,201 | 1.051 | 1.047 | 0.35% |
| | | | | 3-BP | 5,908 | 72 | 106 | 1,158 | 3,913 | 1.049 | 1.049 | 0.06% |
| 20 | 6 | 20 | 6 | CG | 1 | 61 | 0.03 | 4.85 | 5.08 | 1.069 | 1.056 | 1.29% |
| | | | | 2-BP | 19,660 | 61 | 223 | 2,761 | 7,202 | 1.067 | 1.065 | 0.17% |
| | | | | 3-BP | 19,640 | 61 | 222 | 2,766 | 7,201 | 1.067 | 1.065 | 0.17% |
| 20 | 8 | 10 | 2 | CG | 1 | 58 | 0.04 | 5.01 | 5.37 | 1.527 | 1.44 | 5.7% |
| | | | | 2-BP | 17,960 | 58 | 294 | 1,949 | 7,201 | 1.523 | 1.511 | 0.82% |
| | | | | 3-BP | 17,819 | 58 | 338 | 2,057 | 7,201 | 1.52 | 1.515 | 0.31% |
| 20 | 8 | 10 | 4 | CG | 1 | 57 | 0.03 | 3.80 | 4.12 | 1.545 | 1.5 | 2.93% |
| | | | | 2-BP | 18,876 | 57 | 158 | 1,845 | 10,801 | 1.545 | 1.533 | 0.74% |
| | | | | 3-BP | 19,016 | 57 | 156 | 1,783 | 10,801 | 1.545 | 1.53 | 0.74% |
| 20 | 8 | 10 | 6 | CG | 1 | 79 | 0.05 | 5.64 | 6.05 | 1.551 | 1.529 | 1.41% |
| | | | | 2-BP | 21,122 | 79 | 187 | 2,686 | 10,802 | 1.55 | 1.542 | 0.51% |
| | | | | 3-BP | 23,670 | 79 | 196 | 2,588 | 10,802 | 1.55 | 1.542 | 0.51% |
| 20 | 8 | 20 | 4 | CG | 1 | 103 | 0.08 | 12 | 13 | 1.575 | 1.528 | 3.01% |
| | | | | 2-BP | 6,546 | 103 | 77 | 1,642 | 10,802 | 1.572 | 1.564 | 0.45% |
| | | | | 3-BP | 6,536 | 103 | 74 | 1,685 | 10,805 | 1.572 | 1.564 | 0.45% |
| 20 | 8 | 20 | 6 | CG | 1 | 79 | 0.05 | 8.61 | 10 | 1.602 | 1.587 | 0.91% |
| | | | | 2-BP | 5,462 | 79 | 58 | 1,156 | 10,805 | 1.598 | 1.595 | 0.23% |
| | | | | 3-BP | 5,252 | 79 | 59 | 1,423 | 10,806 | 1.598 | 1.595 | 0.23% |
| 20 | 10 | 10 | 2 | CG | 1 | 78 | 0.08 | 8.64 | 9.08 | 2.078 | 1.977 | 4.82% |
| | | | | 2-BP | 16,446 | 78 | 369 | 3,337 | 10,803 | 2.07 | 2.045 | 1.22% |
| | | | | 3-BP | 15,688 | 78 | 347 | 3,627 | 10,803 | 2.07 | 2.045 | 1.22% |
| 20 | 10 | 10 | 4 | CG | 1 | 82 | 0.06 | 6.85 | 7.24 | 2.098 | 2.070 | 1.31% |
| | | | | 2-BP | 11,450 | 82 | 131 | 1,521 | 10,802 | 2.098 | 2.09 | 0.35% |
| | | | | 3-BP | 11,304 | 82 | 130 | 1,570 | 10,802 | 2.097 | 2.09 | 0.35% |
| 20 | 10 | 10 | 6 | CG | 1 | 90 | 0.08 | 7.87 | 8.65 | 2.105 | 2.088 | 0.81% |
| | | | | 2-BP | 10,394 | 90 | 122 | 1,435 | 10,802 | 2.105 | 2.097 | 0.41% |
| | | | | 3-BP | 10,536 | 90 | 120 | 1,384 | 10,802 | 2.105 | 2.097 | 0.41% |
| 20 | 10 | 20 | 2 | CG | 1 | 134 | 0.08 | 15 | 16 | 2.017 | 1.874 | 7.08% |
| | | | | 2-BP | 12,606 | 134 | 229 | 3,221 | 10,801 | 2.009 | 1.988 | 1.04% |
| | | | | 3-BP | 11,864 | 134 | 229 | 3,279 | 10,801 | 2.009 | 1.988 | 1.04% |
| 20 | 10 | 20 | 4 | CG | 1 | 111 | 0.09 | 19 | 32 | 2.115 | 2.058 | 2.7% |
| | | | | 2-BP | 1,050 | 111 | 14 | 454 | 10,823 | 2.112 | 2.099 | 0.61% |
| | | | | 3-BP | 1,042 | 111 | 14 | 527 | 10,827 | 2.112 | 2.099 | 0.61% |
| 20 | 10 | 20 | 6 | CG | 1 | 124 | 0.08 | 15 | 18 | 2.164 | 2.147 | 0.69% |
| | | | | 2-BP | 1,016 | 124 | 13 | 331 | 10,818 | 2.162 | 2.151 | 0.5% |
| | | | | 3-BP | 1,016 | 124 | 14 | 347 | 10,821 | 2.162 | 2.151 | 0.5% |

Table A.4: Performance evaluation of the branch-and-price algorithm for the congestion mitigation problem.

| | | | | | | CPU times (s) | | | | Solution quality | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $n$ | $S$ | $X$ | $Y$ | Method | Nodes | Init. | RMP | PP | Alg. | UB | Solution | Gap |
| 5 | 2 | 4 | 2 | CG | 1 | 7.54 | 0.05 | 0.02 | 0.1 | 0.50 | 0.50 | 0% |
| | | | | 2-BP | 1 | 7.54 | 0.01 | 0.02 | 0.41 | 0.50 | 0.50 | 0% |
| | | | | 3-BP | 1 | 7.54 | 0.01 | 0.02 | 0.41 | 0.50 | 0.50 | 0% |
| 5 | 2 | 4 | 4 | CG | 1 | 8.37 | 0.01 | 0.04 | 0.05 | 0.63 | 0.63 | 0% |
| | | | | 2-BP | 1 | 8.37 | 0.03 | 0.04 | 0.48 | 0.63 | 0.63 | 0% |
| | | | | 3-BP | 1 | 8.37 | 0.01 | 0.04 | 0.45 | 0.63 | 0.63 | 0% |
| 5 | 2 | 6 | 2 | CG | 1 | 8.04 | 0.01 | 0.03 | 0.06 | 0.68 | 0.67 | 0.6% |
| | | | | 2-BP | 2 | 8.04 | 0.04 | 0.05 | 0.54 | 0.68 | 0.68 | 0.2% |
| | | | | 3-BP | 5 | 8.04 | 0.02 | 0.07 | 0.8 | 0.68 | 0.68 | 0.02% |
| 5 | 2 | 6 | 4 | CG | 1 | 10 | 0.01 | 0.09 | 0.11 | 0.82 | 0.81 | 1.23% |
| | | | | 2-BP | 2 | 10 | 0.02 | 0.13 | 0.56 | 0.82 | 0.81 | 0.16% |
| | | | | 3-BP | 8 | 10s | 0.05 | 0.23 | 0.8s | 0.82 | 0.82 | 0% |
| 5 | 2 | 8 | 2 | CG | 1 | 8.85 | 0.01 | 0.26 | 0.28 | 0.87 | 0.87 | 0% |
| | | | | 2-BP | 1 | 8.85 | 0.01 | 0.06 | 0.48 | 0.87 | 0.87 | 0% |
| | | | | 3-BP | 1 | 8.85 | 0.01 | 0.06 | 0.51 | 0.87 | 0.87 | 0% |
| 5 | 2 | 8 | 4 | CG | 1 | 12 | 0.01 | 0.17 | 0.19 | 1.01 | 1.00 | 0.72% |
| | | | | 2-BP | 8 | 12 | 0.06 | 0.36 | 0.95 | 1.01 | 1.01 | 0.27% |
| | | | | 3-BP | 14 | 12 | 0.08 | 0.74 | 1.49 | 1.01 | 1.01 | 0.05% |
| 5 | 4 | 4 | 2 | CG | 1 | 14 | 0.02 | 0.41 | 0.44 | 1.34 | 1.34 | 0.31% |
| | | | | 2-BP | 1 | 14 | 0.02 | 0.4 | 0.83 | 1.34 | 1.34 | 0.31% |
| | | | | 3-BP | 6 | 14 | 0.05 | 0.92 | 1.74 | 1.34 | 1.34 | 0% |
| 5 | 4 | 4 | 4 | CG | 1 | 49 | 0.03 | 2.3 | 2.35 | 1.61 | 1.55 | 4.16% |
| | | | | 2-BP | 28 | 49 | 0.2 | 8.7 | 10.8 | 1.61 | 1.59 | 1.06% |
| | | | | 3-BP | 73 | 49 | 0.79 | 19.4 | 25 | 1.60 | 1.60 | 0.02% |
| 5 | 4 | 6 | 2 | CG | 1 | 33 | 0.02 | 1.35 | 1.39 | 1.90 | 1.90 | 0.19% |
| | | | | 2-BP | 4 | 33 | 0.04 | 2.12 | 2.73 | 1.90 | 1.90 | 0.11% |
| | | | | 3-BP | 7 | 33 | 0.08 | 3.38 | 4.07 | 1.90 | 1.90 | 0.04% |
| 5 | 4 | 6 | 4 | CG | 1 | 185 | 0.02 | 6.21 | 6.27 | 2.20 | 2.05 | 6.61% |
| | | | | 2-BP | 48 | 185 | 0.34 | 32 | 34 | 2.19 | 2.19 | 0.21% |
| | | | | 3-BP | 57 | 185 | 0.43 | 39 | 42 | 2.19 | 2.19 | 0.06% |
| 5 | 4 | 8 | 2 | CG | 1 | 83 | 0.03 | 4.02 | 4.1 | 2.47 | 2.44 | 1.11% |
| | | | | 2-BP | 2 | 83 | 0.05 | 5.4 | 5.92 | 2.47 | 2.44 | 1.09% |
| | | | | 3-BP | 63 | 83 | 0.63 | 29 | 33 | 2.46 | 2.46 | 0.07% |
| 5 | 4 | 8 | 4 | CG | 1 | 550 | 0.03 | 12 | 12 | 2.82 | 2.80 | 0.95% |
| | | | | 2-BP | 4 | 550 | 0.049 | 18 | 18 | 2.82 | 2.80 | 0.89% |
| | | | | 3-BP | 70 | 550 | 0.63 | 96 | 100 | 2.81 | 2.81 | 0.07% |
| 5 | 6 | 4 | 2 | CG | 1 | 24 | 0.03 | 2.02 | 2.09 | 2.05 | 1.94 | 5.32% |
| | | | | 2-BP | 136 | 24 | 1.76 | 24 | 38 | 2.04 | 2.03 | 0.46% |
| | | | | 3-BP | 203 | 24 | 2.9 | 32 | 56 | 2.03 | 2.03 | 0.002% |
| 5 | 6 | 4 | 4 | CG | 1 | 111 | 0.04 | 6.9 | 6.94 | 2.36 | 2.30 | 2.8% |
| | | | | 2-BP | 724 | 111 | 20 | 328 | 656 | 2.36 | 2.33 | 1.02% |
| | | | | 3-BP | 964 | 111 | 42 | 442 | 760 | 2.34 | 2.34 | 0.09% |
| 5 | 6 | 6 | 2 | CG | 1 | 74 | 0.04 | 5.9 | 6.07 | 2.94 | 2.80 | 4.72% |
| | | | | 2-BP | 16 | 74 | 0.21 | 17 | 18 | 2.94 | 2.94 | 0.005% |
| | | | | 3-BP | 16 | 74 | 1.06 | 18 | 20 | 2.94 | 2.94 | 0.005% |
| 5 | 6 | 6 | 4 | CG | 1 | 459 | 0.04 | 20 | 20 | 3.32 | 2.93 | 11.86% |
| | | | | 2-BP | 206 | 459 | 4.1 | 350 | 389 | 3.31 | 3.31 | 0.19% |
| | | | | 3-BP | 243 | 459 | 4.94 | 396 | 443 | 3.31 | 3.31 | 0.1% |
| 5 | 6 | 8 | 2 | CG | 1 | 209 | 0.04 | 12 | 12 | 3.93 | 3.80 | 3.39% |
| | | | | 2-BP | 556 | 209 | 18 | 512 | 770 | 3.92 | 3.90 | 0.36% |
| | | | | 3-BP | 757 | 209 | 31 | 676 | 1,088 | 3.91 | 3.90 | 0.1% |

# Appendix B

# Additional Table for section 4.3

Table B.1 reports our vaccine allocation solution, as compared to the practical and technical benchmarks, with a larger budget of 1 million vaccines per day, using the same nomenclature as Table 4.5. These results confirm the benefits of our methodology as compared to all benchmarks. In this case, the benefits are smaller, percentwise, due to the larger overall number of vaccines. At one extreme, very small vaccine budgets would not make a dent in the dynamics of the epidemic; at the other extreme, very large vaccine budgets would manage to successfully alleviate the impact of the pandemic even under sub-optimal allocation strategy. In between, optimized vaccine distribution provides the strongest benefits by managing spatial-temporal allocation as an extra lever to combat the pandemic.

Table B.1: Vaccine allocation against benchmarks with a budget of 7M vaccines per week.

| | | $n = 4,\ S = 4,\ D = 11$ | | $n = 4,\ S = 4,\ D = 21$ | |
|---|---|---|---|---|---|
| Method | Tolerance | Time (sec.) | Deaths | Time (sec.) | Deaths |
| No vaccine | — | — | 79.98K | — | 79.98K |
| Uniform | — | — | -0.30K | — | -0.30K |
| Cost-based | — | — | -0.30K | — | -0.30K |
| Dynamic | — | — | -0.30K | — | -0.30K |
| Direct | — | n/a | n/a | n/a | n/a |
| Discretization | $\delta = 0.002$ | 1,000* | -0.50K | 1,000* | -0.47K |
| Discretization | $\delta = 0.001$ | 1,000* | -0.51K K | 1,000* | -0.48K |
| Coordinate Descent | | 4.02 | **-0.52K** | 4.02 | **-0.52K** |
| Branch-and-price | $\varepsilon = 0.01$ | 1.50 | **-0.52K** | 3.62 | -0.41K |
| Branch-and-price | $\varepsilon = 0.005$ | 1.94 | -0.51K | 4.93 | **-0.50K** |
| Branch-and-price | Enumeration | 24.6 | **-0.55K** | 257 | **-0.55K** |

| | | $n = 51,\ S = 6,\ D = 11$ | | $n = 51,\ S = 6,\ D = 21$ | |
|---|---|---|---|---|---|
| Method | Tolerance | Time (sec.) | Deaths | Time (sec.) | Deaths |
| No vaccine | — | — | 573.32K | — | 573.32K |
| Uniform | — | — | -13.22K | — | -13.22K |
| Cost-based | — | — | -17.44K | — | -17.44K |
| Dynamic | — | — | -17.20K | — | -17.20K |
| Direct | — | n/a | n/a | n/a | n/a |
| Discretization | $\delta = 0.002$ | 1,000* | -6.96K | 1,000* | -5.88K |
| Discretization | $\delta = 0.001$ | 1,000* | -0.92K | 1,000* | -2.88K |
| Coordinate Descent | | 4.88 | **-21.26K** | 4.88 | **-21.26K** |
| Branch-and-price | $\varepsilon = 0.01$ | 8.43 | -19.66K | 23.1 | **-20.19K** |
| Branch-and-price | $\varepsilon = 0.005$ | 11.1 | -19.97K | 34.5 | **-21.31K** |
| Branch-and-price | Enumeration | n/a | n/a | n/a | n/a |

| | | $n = 51,\ S = 10,\ D = 21$ | | $n = 51,\ S = 12,\ D = 21$ | |
|---|---|---|---|---|---|
| Method | Tolerance | Time (sec.) | Deaths | Time (sec.) | Deaths |
| No vaccine | — | — | 629.39K | — | 649.75K |
| Uniform | — | — | -34.20K | — | -43.60K |
| Cost-based | — | — | -46.05K | — | -58.71K |
| Dynamic | — | — | -46.05K | — | -57.32K |
| Direct | — | n/a | n/a | n/a | n/a |
| Discretization | $\delta = 0.002$ | n/a | n/a | n/a | n/a |
| Discretization | $\delta = 0.001$ | n/a | n/a | n/a | n/a |
| Coordinate Descent | | 5.32 | **-49.41K** | 5.72 | **-61.06K** |
| Branch-and-price | $\varepsilon = 0.01$ | 46.1 | **-47.58K** | 50.6 | **-59.06K** |
| Branch-and-price | $\varepsilon = 0.005$ | 131.5 | **-51.41K** | 194 | **-64.31K** |
| Branch-and-price | Enumeration | n/a | n/a | n/a | n/a |

$*$ and "n/a": the algorithm does not return an optimal and feasible solution, respectively, within the time limit.

Bold fonts indicate the top methods in terms of minimizing the number of deaths.