University of Groningen

Faculty of Science and Engineering

Integration Project IEM

Information Design for Transportation

Author

G. Ansaldo | S3294439

Supervisors

dr. A.K. (Ashish) Cherukuri
dr. ir. G.H. (Gerald) Jonker
J. Verbree, MSc

June 12th, 2020

Abstract

The current focus of navigation apps is to recommend drivers the fastest road to reach a specific destination. While pleasing the individual driver, this behavior contributes to higher levels of traffic congestion. In other words, navigation apps focus on the selfish needs of users and do not align with social welfare. Research has shown that concepts of Bayesian persuasion and Wardrop equilibrium allow the possibility for information design, whereby the navigation app can strategically pick what information to reveal to its users in order to achieve the desired social objective of mitigating traffic congestion.

This study aims at extending upon an existing mathematical model developed with the intention of alleviating traffic congestion. In particular, the research focuses on translating an optimization problem into a functional algorithm. In order to achieve said objective, the programming language MATLAB is used in combination with a brute force method that attempts to solve a non-convex optimization problem. Results give interesting insights into how numerically efficient methods could be constructed to find the optimum solution. In particular, a clear repetitive structure of both the optimal objective function value and the feasibility constraints at the optimum was found.

Contents

List of Figures

List of Tables

Listings

Introduction

Vehicle routing problems (VRPs) are a category of network problems that have gained eminence over the past few years (Srivatsa Srinivas and Gajanand, 2017). These issues carry various objectives, ranging from minimizing travel times and distances to minimizing pollution and fuel consumption. Navigation apps such as Google Maps, Apple Maps, and Waze have a direct affinity to VRPs. The recent advancement of smartphones and other GPS-enabled devices has allowed for improved traffic data collection and it has promoted the expansion of traffic navigation services (Herrera et al., 2010; Balakrishna et al., 2013). Issues regarding congestion are often associated with the use of navigation apps (Cabannes et al., 2017). Congestion has major impacts on economic growth and productivity, also leading to increased fuel consumption and emission. A recent phenomenon of *cut-through driving* has emerged, which is a tactic used to avoid heavy traffic and long delays where drivers take secondary routes. This phenomenon aggravates the existing congestion and contributes to other traffic jams on the cut-through streets, along with accompanying problems such as collisions, pollution from the exhaust, and road rage. Any Ministry of Transport and similar governmental institutions are constantly seeking solutions for traffic-related issues. However, navigation apps do not provide a solution to congestion and do not converge to socially optimal solutions, therefore fostering the issue of congestion (Patriksson, 2015; Nash et al., 1950).

As of now, concepts such as Bayesian persuasion and Wardrop's principles are suitable candidates in the mitigation of traffic congestion due to navigation apps. These concepts have already been utilized in the paper of Zhu and Savla (2018), where an approach focused on relieving traffic congestion has been developed in the form of a mathematical model. However, little has been done in the translation of such a model into a functional algorithm that could be implemented by navigation apps.

This project assists in finding a solution to the congestion problem; however, as mentioned previously, congestion is the cause of many other major issues, and therefore the contribution of this research and similar projects can be considerably larger. For instance, having less traffic would decrease car emissions in large cities, contributing to healthier air and less climate impact. On a deeper note, this project makes an effort at translating the mathematical model presented by Zhu and Savla (2018) into an optimization algorithm that could potentially be implemented into navigation apps. Additionally, on a wider prospective, it can be mentioned that an indirect contribution is made towards any transportation system. Such optimization algorithms could be used by any Ministry of Transport in the regulation of city buses, trolleybuses, and trams and passenger trains, therefore improving the quality of life of the city population. Finally, yet importantly, this application could have a social impact in large metropolitan areas, where traffic is a key issue.

This paper is divided into two parts. Part I, the conceptual and technical research design, provides a clear picture of how the research is designed and carried out. A problem formulation is presented, followed by the objective of the research project. Consequently, research questions alongside methods of research are presented. Part II is the implementation stage, where the objective of the research is achieved.

Part I

Conceptual & Technical Research Design

1 Problem Analysis

In this section, the problem of congestion due to the usage of navigation apps is contextualized. Later, a problem statement is provided, which will be followed by a stakeholder analysis and system description.

1.1 Problem Context

The recent advancement of technology allowed the integration of the internet and GPS functionalities into smartphones. Consequently, this advancement contributed to the introduction of navigation apps. Web mapping services such as Google, introduced in 2005, have transformed the online mapping experience (Schmidt and Weiser, 2012). As stated in the article by Hardy (2012), in February 2012, Google Maps had over 65 million viewers. Likewise, another article published by Russell (2019) indicates that more than a billion people were using Google Maps every month in 2019. Undoubtedly, this data shows how navigation apps are steadily growing in recent years and how embedded they are in daily life.

Web mapping services are commonly used for navigation purposes. It is reasonable to assume that a common behavior that travelers have is to choose routes perceived as being the shortest, under the prevailing traffic conditions (Correa et al., 2004). It is, therefore, the case that navigation apps currently suggest travel-minimizing routes. Indeed, modern navigation apps are developed to assist drivers in reaching their destination in the least time possible. However, it is imperative to mention that these apps only focus on the need of a single user. In other words, if many drivers need to get to the same destination, the navigation app suggests the same fast road to each one of them without distributing the drivers into different roads, thus contributing to higher congestion. Evidently, it can be mentioned that navigation apps focus on the selfish needs of users and do not take into account the social welfare. This phenomenon has been studied by the English mathematician J. G. Wardrop, who introduced two principles that establish different notions of equilibrium (Wardrop, 1952).

1.2 Problem Statement

After contextualizing the problem at hand, a concise problem statement has been developed.

> *The recent growth of navigation applications has aggravated the issue of congestion. This stems from the fact that the current applications provide travel-time minimizing routes to all users. This method focuses on the selfish needs of users and does not align with social welfare, thus contributing to higher congestion.*

1.3 Stakeholders Analysis

A stakeholder analysis was performed to identify stakeholders and their role in the research. Three stakeholders were determined: ODS Group, Ministry of Transport, and Navigation Apps CEOs. Fig. 1 below places the various stakeholders in a Mendelow's diagram.

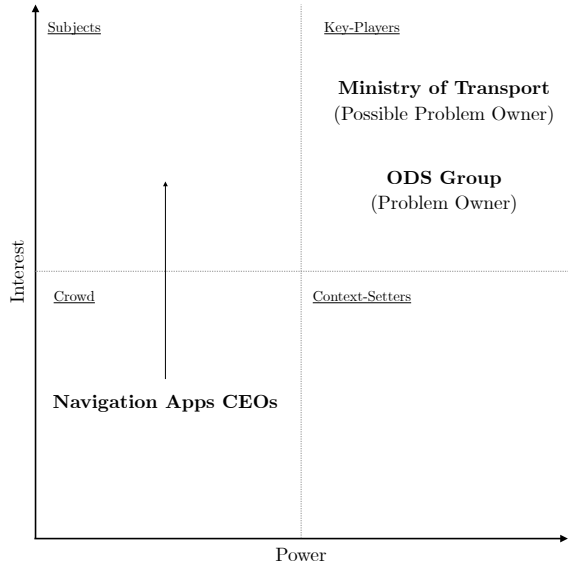

Figure 1: Mendelow's diagram

The three main stakeholders identified are further discussed.

- Optimization and Decision Systems (ODS) Group: The ODS group is also known as the problem owner. This research group has high interest and power in the project. As the commissioning party, the ODS group has a great interest in the outcome of the research. Additionally, the ODS group has a steering function, and thus remarkable power. The primary role of the ODS group is to set the scope of the project and to adopt a steering function in order to guide the research.

- Any Ministry of Transport: The Ministry of Transport is the organization that is responsible for helping maintain and develop the nation's transportation systems and infrastructure. The goal of this organization is to ensure its nation to have the safest, most efficient, and modern transportation system, which improves the quality of life for the population. Since the Ministry of Transport is often involved in solving congestion issues, it is in their interest to seek a solution to this problem. Indeed, this organization has a high interest in possible solutions that could alleviate the problem of traffic congestion. Additionally, their remarkable power could potentially steer the scope of the project towards their needs. It has to be mentioned that the Ministry of Transport is not a current problem owner, nevertheless, it is considered as a possible future problem owner.

- Navigation Apps CEOs: CEOs of navigation apps have a devoid position of power concerning the research. However, they have the potential to gain new knowledge from this project and therefore their interest could increase. It is important to mention that in the situation where an optimization algorithm that alleviates traffic congestion is developed, any Ministry of Transportation could request navigation apps to implement the algorithm in urban areas, where congestion is prominent. Consequently, navigation apps CEOs are likely to be affected by the results of this project, thus their interest in the project could potentially increase.

1.4 System Description & Scope

In this section a more detailed description of the problem is provided. Subsequently, a scope is delineated.

1.4.1 System Description

The two main elements that describe the system are the navigation app and the drivers. Navigation apps allow drivers to stay updated on the current information, and most importantly they take decisions for them. These decisions reflect in the advice of taking a specific road to reach the drivers' destination. Alternatively stated, the interactions between navigation apps and drivers follow the dynamics of Bayesian persuasion (Kamenica and Gentzkow, 2011). In Bayesian persuasion one person, the sender, wishes to persuade another, the receiver, to change her action. It is therefore clear to understand that the sender is considered to be the navigation app and the receiver is the user. This entails that the sender, the navigation app, can strategically choose what information to reveal to achieve a desired objective. The current objective utilized by navigation apps is the one of minimising the travel time of single drivers, which is the reason why navigation apps focus on the selfish needs of users, leading to higher congestion (see the next section for an example). However, in the model developed by Zhu and Savla (2018) the objective is to reduce the problem of congestion by recommending drivers to take different roads. However, this recommendation needs to persuade the driver. A driver follows a recommendation only if the recommendation is optimal with respect to her posterior belief about the state of the roads. Such constraints are formally referred to as incentive compatibility constraints. In other words, a driver follows the recommendation only if she thinks that taking the suggested road reduces her travel time, in respect to her posterior knowledge about the state of parallel roads.

1.4.2 Example

In this section an example of how the current navigation app suggestions lead to congestion is discussed. Consider a simple system where multiple users need to go from point A to point B, and two parallel routes can be chosen. Route 1 is the shortest and route 2 is the longest (refer to Figure 2).

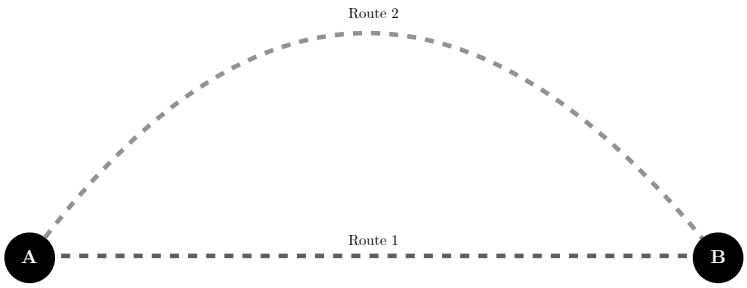

Figure 2: Example of congestion projection

Current navigation apps suggest users to take route 1 since it is the shortest option. When this happens, every user will find themselves in the same path and therefore congestion will appear on route 1. When new users enter the system, the navigation app is aware of the congested route 1, and therefore the route 2 becomes the shortest path. This means that users are advised to choose route 2. Again, this suggestion simply switches the congestion present in route 1 to route 2. As evidenced, an optimal equilibrium for the whole system is not reached, and only the selfish needs of the individual users are satisfied. Therefore, the focus of the research is to find an optimal solution that reduces congestion.

1.4.3 Scope

Once the system is described, it is possible to delineate the scope of research. Indeed, the disciplines of mathematics and programming are at the base of such research. Additionally, Bayesian persuasion theories and Wardrop's principles are key aspects of this project. Furthermore, it is imperative to mention that the model presented by Zhu and Savla (2018) applies to two different scenarios: static and dynamic. On the one hand, in the static scenario, it is assumed that every driver trusts the recommendation given by navigation apps. Additionally, no time evolution is present. Time evolution is defined as the change of state brought about by the passage of time. On the other hand, in the dynamic scenario, some drivers do not trust the advice given by the navigation app. Moreover, time evolution is present, meaning that at each time the state of drivers and perhaps traffic is different. Due to the time constraint of this research, only the static scenario is studied.

2 Research Goal

2.1 Cycle Choice

This research project has a practice-oriented nature. Therefore, it was decided to adopt the intervention cycle. The intervention cycle is a predefined set of steps to reach a solution relating to operational problems (Hermans and Schoeman, 2015). The focus of this project lies in the design stage. This is due to the fact that the problem at hand has already been identified, analyzed, and diagnosed.

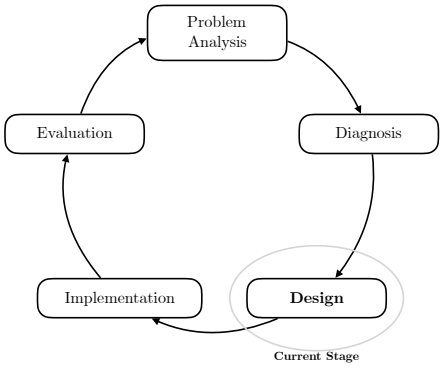

Figure 3: Intervention Cycle

For the current project, the designing phase consists in translating and adapting an existing mathematical model which alleviates the current problem of congestion into an optimization algorithm. The said mathematical model introduces both aspects of Bayesian persuasion theories and Wardrop' principles. This model has been developed by Zhu and Savla (2018) in the paper *"On the Stability of Optimal Bayesian Persuasion Strategy under a Mistrust Dynamics in Routing Games"*.

2.2 Goal Statement

After having identified a problem statement and described the system, the research objective can be formulated.

The goal of the project is to contribute towards the advancement of navigation applications by translating the mathematical model presented by Zhu and Savla (2018) into an optimization algorithm that solves the current problem of congestion caused by the misalignment of users' selfish needs and social welfare.

3 Research Questions

In order to acquire the data and knowledge necessary to achieve the design goal, a set of research questions has been formulated. Subsequently, secondary questions have been introduced to contribute to the answering of the main core question. The following questions have been formulated:

1. How can the design of an optimization algorithm be realized based upon the mathematical model proposed by Zhu and Savla (2018)?
    (a) What are the characteristics of the mathematical model presented by Zhu and Savla (2018)?
    (b) What are the current limitations of the mathematical model presented by Zhu and Savla (2018)?
    (c) What current existing methods could be used to solve the optimization model presented by Zhu and Savla (2018)?

4 Research Methodology

The research is focused on programming an optimization algorithm based on the mathematical model of Zhu and Savla (2018). In order to achieve this objective, the computer programming language MATLAB is used. Research questions are answered with the help of literature, experimentation, and programming. Table 1 shows a clear overview of the research methods that are used to carry out the project.

Question		**Deliverable**	**Strategy**	**Tool**
1	1.a	Optimization Algorithm	Experimentation	MATLAB
	1.b	based on the mathematical	Programming	
	1.c	model of Zhu and Savla (2018)	Literature	

Table 1: Overview of Research Methods

4.1 Validation

Validation is performed to assess the outcome of the research. Moreover, the validation process ensures that the stakeholders' needs are satisfied (Wheatcraft, 2012). The internal validity of the research is achieved by ensuring that the predetermined goal is reached. Simply put, it is crucial to verify the functioning of the optimization algorithm. For the purpose of this study, the algorithm is tested for diverse settings. A possible setting could consist of various routes and several drivers. For instance, a method to ensure the functioning of the algorithm is to compare the results obtained with the ones of the paper by Zhu and Savla (2018). Additionally, in order to ensure internal validity it is crucial to pay close attention to the various variables of the algorithm and what effects they have on the output.

4.2 Risk Analysis

Risk analysis is essential in order to identify the various hazards that could arise during the research. Challenges could be faced in the process of programming the algorithm. In other words, various assumptions would have to be made while programming and designing the algorithm. Assumptions are essential in order to simplify the model and easily adapting it to an algorithm. Additionally, difficulties could be encountered in solving the optimization problem since, as claimed by Zhu and Savla (2018), it is a non-convex problem.

Part II

Implementation Stage

5 Mathematical Setup

This section provides a detailed description of the model presented by Zhu and Savla (2018).

5.1 Problem Formulation

The model is based on the dynamics of Bayesian persuasion where the sender, the navigation app, gives different recommendations to specific drivers in order to reach a particular objective. In this setting, the recommendations given by the navigation app need to persuade the drivers; meaning that by following the advice the driver is better off than making a decision based only on her prior knowledge of what the state of the network might be. This is translated into an optimization problem, where an objective function is minimized under specific constraints. The problem has the following structure:

$$\begin{aligned} \min \quad & objective \\ \text{s.t.} \quad & constraint(1) \\ & constraint(2) \\ & constraint(3) \\ & constraint(4) \end{aligned} \tag{1}$$

Before diving into the specific characteristics of the objective function and each constraint, it is important to introduce some parameters.

Consider two points A and B, and let $m = \{1, 2, ..., j\}$ be the number of parallel roads that connect these two points. A network has finitely many possible states which are defined as $\Theta = \{\theta_1, ..., \theta_s\}$. These states represent the possible conditions of each parallel road; for instance a road could be congested or empty. Let q designate the probability distribution with support over Θ; q is known by both the navigation app and the various drivers (common prior). In other words, q is the likelihood that the network of roads is at a specific state θ. In the model there are two types of drivers, those who participate in persuasion, and those who do not. Simply put, some drivers use navigation apps and some do not. These drivers are referred to as p and b type drivers respectively. Therefore, v^p and v^b are the fraction of p-type and b-type drivers, where $v^p + v^b = 1$, and $v = (v^b, v^p)$. Additionally, let $f_j^b, f_j^{p,\theta} \geq 0$ be the flow induced by drivers of type b and p on road j. Let $f_j^\theta = (f_j^b, f_j^{p,\theta})$ be the compact notation. Therefore, the total flow on road j is $f_{s,j}^\theta = f_j^b + f_j^{p,\theta}$. Consequently, the delay function on road j at state θ is defined as:

$$\ell(f_j^\theta) = \alpha_j^\theta f_{s,j}^\theta + \beta_j^\theta \tag{2}$$

where β_j^θ is the constant of free flow travel time, meaning the amount of time it takes to traverse road j at state θ when no traffic is present; and α_j^θ is the sensitivity of the delay function to flow. Furthermore, let ϕ_j^θ denote the fraction of p-type drivers who are recommended to take route j when the realization of state is θ. This can be considered as the signaling scheme of the navigation app where $\Phi = \{\phi_j^\theta\}_{\theta \in \Theta, j \in [m]} \in [0, 1]^{|\Theta| \times m}$.

For the sake of clarity the variables defined above are summarized in Table 2 below.

Variable	**Description**	**Mathematical Representation**
m	Number of parallel roads that connect two points	$m = \{1, 2, ..., j, ..., m\}$
θ	Possible states of the network	$\Theta = \{\theta_1, ..., \theta_s\}$
q	Probability distribution with support over θ	$q = \{q^{\theta_1}, q^{\theta_2}, ..., q^{\theta_s}\}$
v^p	Fraction of p-type drivers	$v^p \in [0, 1]$, $v = (v^b, v^p)$
v^b	Fraction of b-type drivers	$v^b \in [0, 1]$, $v = (v^b, v^p)$
f_j^b	Flow induced by agents of type b on link j	$f_j^b \geq 0$, $f_j^\theta = (f_j^b, f_j^{p,\theta})$
$f_j^{p,\theta}$	Flow induced by agents of type p on link j	$f_j^{p,\theta} \geq 0$, $f_j^\theta = (f_j^b, f_j^{p,\theta})$
$\ell(f_j^\theta)$	Delay function on link j at state θ	$\ell(f_j^\theta) = \alpha_j^\theta f_{s,j}^\theta + \beta_j^\theta$
α_j^θ	Sensitivity of the delay function to flow	$\alpha_j^\theta \geq 0$
β_j^θ	Free flow travel time	$\beta_j^\theta \geq 0$
ϕ_j^θ	p-type drivers recommended to take route j	$\Phi = \{\phi_j^\theta\}_{\theta \in \Theta, j \in [m]} \in [0, 1]^{\lvert\Theta\rvert \times m}$

Table 2: Description of Variables

5.2 Detailed Elaboration

After having defined the main variables and ingredients of the model, it is possible to provide a detailed description of the objective function and constraints.

Objective

The objective of the navigation app is to minimize expected social cost over all feasible recommendation schemes. The social cost can be seen as the sum of the delay per road caused by the amount of drivers. Therefore, for a given state of the network θ, the corresponding social cost is:

$$J^{\theta,v}(\Phi, f^b) = \sum_{j \in [m]} f_{s,j}^\theta \ell(f_j^\theta). \tag{3}$$

Consequently, the expected social cost depends on the likelihood q of state θ to happen, and is expressed as:

$$J^{q,v}(\Phi, f^b) = \mathbb{E}[J^{\theta,v}(\Phi, f^b)] = \sum_{\theta \in \Theta} q^\theta J^{\theta,v}(\Phi, f^b). \tag{4}$$

Therefore, Equation 4 is the considered objective function.

Constraint 1

The first constraint relates to the signaling scheme Φ performed by the navigation app. Since ϕ_j^θ denotes the fraction of p-type drivers who are recommended to take route j when the realization of state is θ, the first constraint is expressed as:

$$\sum_{j\in[m]} \phi_j^\theta = 1. \tag{5}$$

Simply put, it means that a single driver receives one and only one recommendation.

Constraint 2

The second constraint relates to the assumption that all p-type drivers follow the recommendation given by the navigation app. The drivers follow the recommendation if they are incentive compatible, i.e., if the recommendations minimize the agents' expected travel cost with respect to the posterior distribution of θ induced by the recommendations. Hence, the second constraint goes as follows:

$$f_j^{p,\theta} = v^p \phi_j^\theta. \tag{6}$$

Constraint 3

The third constraint is the incentive compatible constraint. The recommendation received by p-type drivers about following a specific road j, along with their knowledge about the likelihood q of state θ to happen, leads to the posterior probability of p-type agents. The posterior probability is defined as the probability that state θ is occurring given a specific road j to be followed, and it goes as follows:

$$\Pr(\theta|j) = \frac{\Pr(j|\theta)\Pr(\theta)}{\sum_{\omega\in\Theta}\Pr(j|\omega)\Pr(\omega)} = \frac{\phi_j^\theta q^\theta}{\sum_{\omega\in\Theta}\phi_j^\omega q^\omega}. \tag{7}$$

Therefore, the incentive compatibility condition for road j can be written as:

$$\sum_{\theta\in\Theta}\Pr(\theta|j)\ell_j^\theta(f_j^\theta) \le \sum_{\theta\in\Theta}\Pr(\theta|j)\ell_k^\theta(f_k^\theta) \qquad \forall\ k\neq j. \tag{8}$$

Upon substituting Equation 2 and Equation 7, the incentive compatibility constraints over all roads can be collectively written as:

$$\sum_{\theta\in\Theta} q^\theta \alpha_j^\theta \phi_j^\theta f_{s,j}^\theta + \sum_{\theta\in\Theta} q^\theta \beta_j^\theta \phi_j^\theta \le \sum_{\theta\in\Theta} q^\theta \alpha_k^\theta \phi_j^\theta f_{s,k}^\theta + \sum_{\theta\in\Theta} q^\theta \beta_k^\theta \phi_j^\theta \qquad \forall\ k\neq j,\ j\in[m] \tag{9}$$

Constraint 4

The fourth constraint is related to the distribution of b-type drivers on the various roads. These drivers find themselves choosing what road j to follow depending only on their prior knowledge of the likelihood q of state θ to occur. This constraint is related to the notion of Bayesian Wardrop equilibrium, and it goes as follows:

$$\mathbb{E}[\ell_j^\theta(f_j^\theta)] \leq \mathbb{E}[\ell_k^\theta(f_k^\theta)] \qquad \forall \;\; k \neq j, \; j \in [m]. \tag{10}$$

Upon substituting Equation 2, constraint 4, also known as the *BWE-P* constraint, can be written as:

$$\sum_{\theta\in\Theta} q^\theta \alpha_j^\theta f_{s,j}^\theta + \sum_{\theta\in\Theta} q^\theta \beta_j^\theta \leq \sum_{\theta\in\Theta} q^\theta \alpha_k^\theta f_{s,k}^\theta + \sum_{\theta\in\Theta} q^\theta \beta_k^\theta \qquad \forall \;\; k \neq j, \; j \in [m] \tag{11}$$

Constraint 4 can be considered to be a dynamic constraint as it depends on the behavior of b-type drivers. The b-type drivers are believed to be 'selfish', meaning that, after the p-type drivers have made their routing choices, the b-type flow drivers simply take the road with the smallest possible expected travel time. One of the consequences of such behavior is that all roads with b-type flow on it give the same expected travel time, and there are no roads available that have lower expected travel time than the roads with b-type flow on it. Simply put, if one road with or without b-type flow on it has less expected travel time than another road with b-type flow on it, then some of the flow on the second road moves to the first road, since it is faster, until both roads have the same expected travel time, or the second road has no more b-type flow. Mathematically speaking the *BWE-P* constraint behaves as follows:

- In the case where there is no b-type flow, constraint 4 does not exist.
- In the case where b-type flow is present on all roads then the *BWE-P* constraint is an equality constraint, and goes as follows:

$$\sum_{\theta\in\Theta} q^\theta \alpha_j^\theta f_{s,j}^\theta + \sum_{\theta\in\Theta} q^\theta \beta_j^\theta = \sum_{\theta\in\Theta} q^\theta \alpha_k^\theta f_{s,k}^\theta + \sum_{\theta\in\Theta} q^\theta \beta_k^\theta \qquad \forall \;\; k \neq j, \; j \in [m] \tag{12}$$

- In the case where b-type flow is present and is zero on some roads, then the roads where the b-type flow is non-zero need to have smaller expected travel time than the ones where the b-type flow is zero. Therefore, the constraint goes as in Equation 11, where road j has b-type flow, and road k has none.

Ultimately, the problem of computing the optimal Bayesian persuasion strategy for the navigation app is formulated as:

$$
\begin{aligned}
\min \quad & J^{q,v}(\Phi, f^b) \\
\text{s.t.} \quad & \sum_{j\in[m]} \phi_j^\theta = 1 \\
3 \qquad & f_j^{p,\theta} = v^p \phi_j^\theta \\
& \sum_{\theta\in\Theta} q^\theta \alpha_j^\theta \phi_j^\theta f_{s,j}^\theta + \sum_{\theta\in\Theta} q^\theta \beta_j^\theta \phi_j^\theta \leq \sum_{\theta\in\Theta} q^\theta \alpha_k^\theta \phi_j^\theta f_{s,k}^\theta + \sum_{\theta\in\Theta} q^\theta \beta_k^\theta \phi_j^\theta \qquad \forall\ k \neq j,\ j \in [m] \\
& \text{Dynamic } \textit{BWE-P} \text{ constraint}
\end{aligned}
\tag{13}
$$

From the optimization problem presented in Equation 13 it is clear to see that the variables of interest are Φ and f^b. The goal is to find the set of variables Φ and f^b such that the objective function is minimum and its constraints are satisfied. The remaining variables summarized on Table 2 are known constants regarding the network of study.

5.3 Current Limitations

The optimization problem derived on Equation 13 has various properties which are further discussed. Given a network with more than one road, thus $m \geq 2$, the following properties hold for Equation 13:

(i) The objective function, Equation 4, is convex.

(ii) Equation 13 has a non-empty feasible set, which is generally non-convex.

(iii) When $v^p = 1$ then Equation 13 is a convex optimization problem, with quadratic objective function and quadratic constraints.

Noteworthy, is that the non-convex feasible set is the union of a few convex sets.

6 Implementation of the Model by Zhu and Savla (2018)

In this section, the translation of the mathematical model into an optimization algorithm is performed. For the sake of this research, the computer programming language MATLAB is used. The problem in Equation 13 is first solved for the convex simplification and then a non-convex solution is considered, in both cases a 2-roads 2-states setting is examined. In order to validate the functioning of the model, the two examples shown in the paper by Zhu and Savla (2018) are studied.

6.1 Convex Simplification

It was shown by Zhu and Savla (2018) that the optimization problem presented on Equation 13, is convex when $v^p = 1$. This entails, that all drivers utilize and follow the advice of the navigation app. Therefore, when every driver participates in the persuasion game, constraint 4 (Equation 10) does not apply. This is due to the fact that every driver receives additional information about the state of the network, and b-type drivers are not present. Consequently, the optimization problem can be simplified as follows:

$$
\begin{aligned}
\min \quad & J^{q,v}(\Phi) \\
\text{s.t.} \quad & \sum_{j\in[m]} \phi_j^\theta = 1 \\
& f_j^{p,\theta} = v^p \phi_j^\theta \\
& \sum_{\theta\in\Theta} q^\theta \alpha_j^\theta \phi_j^\theta f_{s,j}^\theta + \sum_{\theta\in\Theta} q^\theta \beta_j^\theta \phi_j^\theta \le \sum_{\theta\in\Theta} q^\theta \alpha_k^\theta \phi_j^\theta f_{s,k}^\theta + \sum_{\theta\in\Theta} q^\theta \beta_k^\theta \phi_j^\theta \qquad \forall\ k \neq j,\ j \in [m]
\end{aligned}
\tag{14}
$$

In the process of programming the optimization algorithm, constraint 2 (Equation 6) was not directly considered as a constraint. However, it was decided to implement this relation within each equation. In other words, the flow of p-type agents, $f_j^{p,\theta}$, is considered to be equal to $v^p \phi_j^\theta$ in every equation of the model. The simulation codes can be viewed in the Appendix. In order to verify the simulations, the example presented in the paper by Zhu and Savla (2018) was applied. After running the code, the same solution of $(\phi_1^{\theta_1}, \phi_1^{\theta_2}, \phi_2^{\theta_1}, \phi_2^{\theta_2}) = (0.7455, 0.9475, 0.2545, 0.0525)$ obtained in the paper was achieved. Therefore, this suggests that the code performs as intended.

6.2 Non-Convex Problem: 2-Roads 2-States Case

After having analyzed the convex simplification of the optimization problem, it is possible to dive into the non-convex case. The problem becomes non-convex when the flow of b-type agents is non zero. Since non-convex optimization problems are often challenging to solve, a brute force method has been used in order to find an optimal solution. Referring to Equation 13, it was decided to simulate the problem for all the possible distributions of b-type drivers within each road. In other words, the value of f_j^b becomes an input, and for a specific set of f_j^b a solution Φ is obtained. As an example, in the case where the network consists of two roads, $m = 2$, and the fraction of b-type agents is $v^b = 0.4$, the brute force method solves the problem of Equation 13 for each input f_j^b where $f_j^b = (f_1^b, f_2^b) \rightarrow$

$\{(0, 0.4), (0.01, 0.39), (0.02, 0.38), ..., (0.4, 0)\}$. The rate of increment and decrease of f_1^b and f_2^b respectively is defined to be r; in the example above $r = 0.01$.

The simulation code utilized to run the brute force method for the non-convex case is available in the Appendix. The following results where obtained for a 2-roads 2-states example, where the rate $r = 0.004$, and $v^b = 0.4$. In Figure 4 below the value of the objective function at its optimal solution Φ is plotted against the various combinations of $f_j^b = (f_1^b, f_2^b)$. For the sake of clarity, set 1 corresponds to $(f_1^b, f_2^b) = (0, 0.4)$, set 2 to $(f_1^b, f_2^b) = (0.004, 0.396)$ and so forth. Additionally, Figure 4 shows the behaviour of the optimization constraints.

From the graphs depicted on Figure 4, the flat region of the optimal value corresponds to the set of optimal feasible solutions. This is true as the objective function is minimum in that region. These solutions are optimal for a range of $f_j^b = (f_1^b, f_2^b)$. Outside that range, the value of the objective function increases. The range of $f_j^b = (f_1^b, f_2^b)$ that gives the optimal solution is approximately between $(f_1^b, f_2^b) = (0, 0.4)$ and $(f_1^b, f_2^b) = (0.333, 0.067)$. In other words, the solution to the problem is optimal for various Φ each related to a set of f_j^b which belongs to said range. From the navigation app prospective, these results entail that the recommendation given by the navigation app to p-type drivers results in an optimal minimization of traffic in the case where b-type drivers are distributed between the two roads in the same manner as the said range.

6.3 Graph Notation

In this section, an explanation of how to read the graphs of Figure 4, Figure 5, and Figure 6 is given. For the sake of simplicity, in the programming process, equality constraints are reorganized so that each constraint is equal to zero, while inequality constraints are rearranged such that they have to be smaller or equal than zero. For instance, constraint 1 is rewritten as follows:

$$\sum_{j\in[m]} \phi_j^\theta - 1 = 0 \tag{15}$$

and constraint 3 is rearranged as follows:

$$\sum_{\theta\in\Theta} \Pr(\theta|j)\ell_j^\theta(f_j^\theta) - \sum_{\theta\in\Theta} \Pr(\theta|j)\ell_k^\theta(f_k^\theta) \leq 0 \qquad \forall\ k \neq j. \tag{16}$$

The other constraints are rearranged similarly.

Subplot 1: Optimal Value
In this graph, the optimal value is plotted for each f_b set number. A blue star ($*$) indicates that the solution found is feasible, while a red star ($*$) implies that the solution is non-feasible. A solution is feasible if all its constraints are satisfied, and a solution becomes non-feasible if at least one of its constraints is not satisfied. Additionally, the green star ($*$) highlights the optimal feasible solution, while a yellow star ($*$) represents the optimal non-feasible solution.

Subplot 2: Constraint 1 (ϕ *per road*)
This graph shows the behavior of constraint 1 (Equation 15) with respect to each f_b set number. It can be seen that the number of points per f_b set number varies depending on the number of states. For a 2-states example, the number of values of constraint 1 per f_b set number is 2, while for a 3-states case it will be 3. In other words, the number of points per f_b set is equal to the number of states. To be noted that often these values are very similar and therefore the points on the graph might overlap. A blue star ($*$) indicates that the constraint is satisfied, while a red star ($*$) implies that the constraint is not satisfied. Additionally, a green star ($*$) highlights the constraints that belong to the optimal feasible solution; these constraints are indeed always satisfied. Moreover, a yellow star ($*$) indicates the constraints belonging to the optimal non-feasible solution that are not being satisfied; however, if these constraints are satisfied, a blue star ($*$) is kept instead of a yellow one. Noteworthy that this colored notation also applies for subplots 3, 4, and 5.

Subplot 3: Constraint 3 (*I.C.C.*)
This graph shows the behavior of constraint 3 (Equation 9) with respect to each f_b set number. It can be seen that the number of points per f_b set number varies depending on the number of roads. The number of points per f_b set can be found as follows:

$$\frac{\text{number of links!}}{(\text{number of links} - 2)!}. \tag{17}$$

This is due to the nature of constraint 3.

Subplot 4 & 5: Constraint 4 (*BWE-P*)

These last two subplots show the behavior of constraint 3 (Equation 10). As discussed previously, the *BWE-P* constraint is a dynamic constraint that depending on the distribution of b-type drivers can become an inequality or equality constraint. Subplot 4 shows the equality constraint. It can be seen that the number of points per f_b set number varies depending on the number of roads. For instance, for a 2-roads example, the number of values per f_b set number is 2, while for a 4-roads case it will be 6. This is due to the nature of constraint 4 (Equation 10). Subplot 5 shows the inequality constraint. Yet again, the number of points per f_b set number varies depending on the number of roads. Additionally, it can be seen that for some f_b sets this constraint does not hold. This is due to the fact that the inequality constraint only applies when the flow of b-type is zero on at least one road. For instance, in a 2-roads case, the inequality constraint only holds twice: when all b-type flow is on one road and when all b-type flow is on the other road.

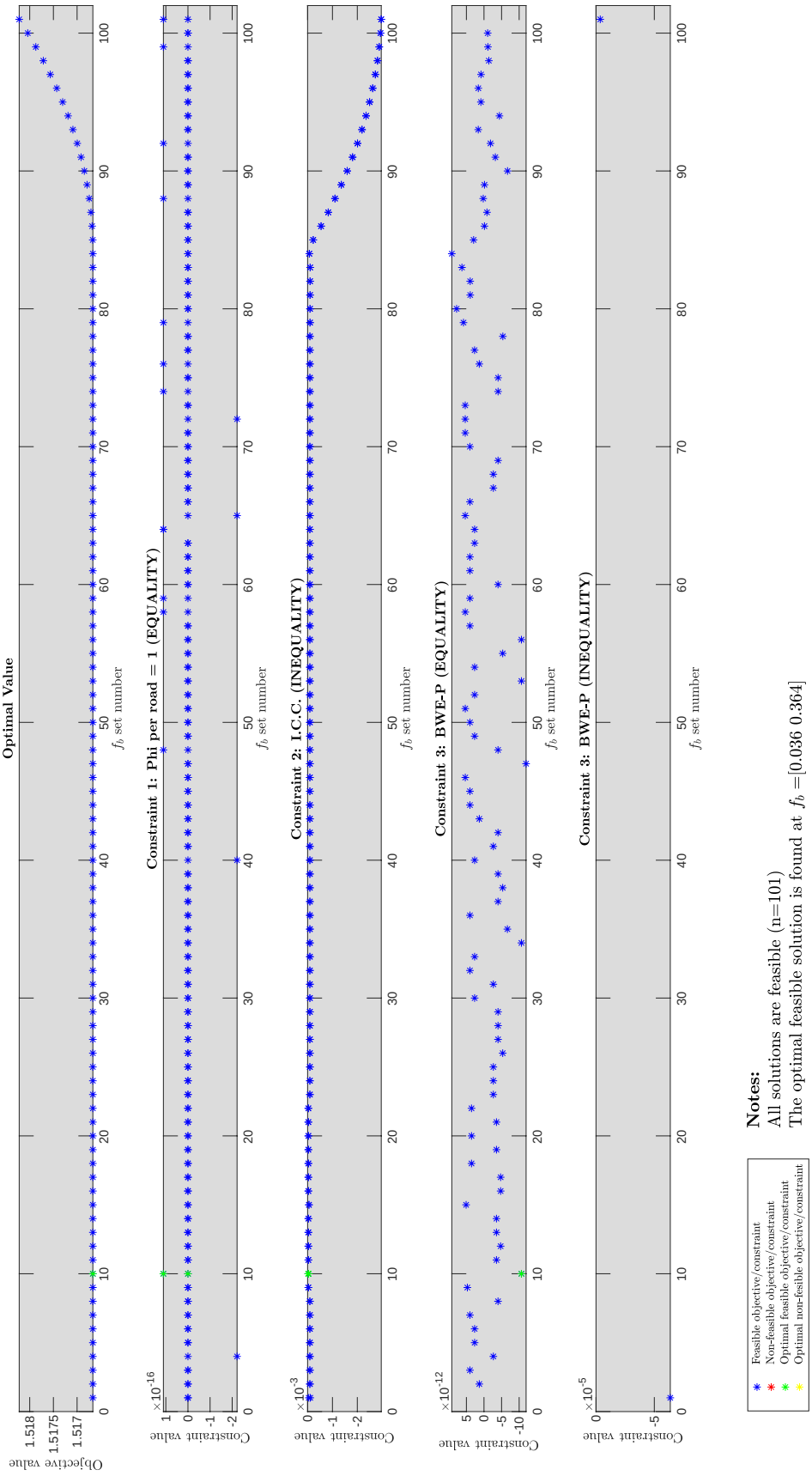

Figure 4: Objective Function Evaluation for Different Sets of $f_j^b = (f_1^b, f_2^b)$

7 Non-Convex Simulations

In this section, the properties of the model presented by Zhu and Savla (2018) are further discussed for two different cases: 3-roads 2-states and 4-roads 2-states. For both cases, the constants depicted in Table 2 are generated randomly since no collected data was available. The MATLAB code utilized to solve these two cases is shown in the Appendix. Indeed, a brute force method was utilized where all possible permutations of f_j^b are inputted. For the sake of clarity, an input example of f_j^b for a 3-roads network is provided in Table 3 below. For this example $v_b = 0.3$ and the rate of increment $r = 0.1$.

f_j^b set number	$f_j^b =$	$(f_1^b$	f_2^b	$f_3^b)$
1		(0	0	0.3)
2		(0	0.1	0.2)
3		(0	0.2	0.1)
4		(0	0.3	0)
5		(0.1	0	0.2)
6		(0.1	0.1	0.1)
7		(0.1	0.2	0)
8		(0.2	0	0.1)
9		(0.2	0.1	0)
10		(0.3	0	0)

Table 3: Example of f_j^b sets

As seen from the table above the various sets of f_j^b are ordered based on the values in the first entry f_1^b. In the case where the first entry is similar for various sets of f_j^b, the sorting method sorts according to the values in the next entry f_2^b. This behavior is repeated for succeeding equal values. This example is crucial in order to properly understand the graphs obtained in the simulations.

7.1 3-Roads 2-States Case

A 3-roads 2-states example is performed using randomly generated constants shown in Table 4 below.

Constants for the 3-Roads 2-States Case

$$
\begin{array}{ccccc}
q^{\theta} & = & \begin{pmatrix} q^{\theta_1} & q^{\theta_2} \end{pmatrix} & = & \begin{pmatrix} 0.1690 & 0.8310 \end{pmatrix} \\
v & = & \begin{pmatrix} v_b & v_p \end{pmatrix} & = & \begin{pmatrix} 0.7218 & 0.2782 \end{pmatrix} \\
\alpha_j^{\theta} & = & \begin{pmatrix} \alpha_1^{\theta_1} & \alpha_2^{\theta_1} & \alpha_3^{\theta_1} \\ \alpha_1^{\theta_2} & \alpha_2^{\theta_2} & \alpha_3^{\theta_2} \end{pmatrix} & = & \begin{pmatrix} 0.4735 & 0.3411 & 0.1917 \\ 0.1527 & 0.6074 & 0.7384 \end{pmatrix} \\
\beta_j^{\theta} & = & \begin{pmatrix} \beta_1^{\theta_1} & \beta_2^{\theta_1} & \beta_3^{\theta_1} \\ \beta_1^{\theta_2} & \beta_2^{\theta_2} & \beta_3^{\theta_2} \end{pmatrix} & = & \begin{pmatrix} 0.2428 & 0.2691 & 0.1887 \\ 0.9174 & 0.7655 & 0.2875 \end{pmatrix}
\end{array}
$$

Table 4: Randomly Generated Constants for a 3-Roads 2-States Case

The results of the simulation are displayed in Figure 5 on the next page. It is evident from the graph that a repetitive structure is present for the optimal values of ϕ as well as in the way the constraints are violated or satisfied. However, it is not possible to confirm that the non-convex feasible set is the union of finite convex sets. This is due to various reasons, such that the f_b set numbers could be positioned in a different order and therefore resulting in a different shape. However, a more plausible justification stems from the fact that the graph only shows the optimal values of ϕ for different values of f_b, and therefore only limited information is available about the feasible set of ϕ Furthermore, it is noteworthy that the *BWE-P* equality and inequality constraints are the constraints that are violated most often causing solutions not to be feasible. Another interesting behavior can be seen by looking at the graph of both the *I.C.C* and *BWE-P* constraint. Indeed, the graphs suggest that these constraints behave in a monotonic way, on subsets of the possible flows f_b. The behavior of the *BWE-P* equality constraint and the *I.C.C* is also similar for the 4-roads 2-states case explained next.

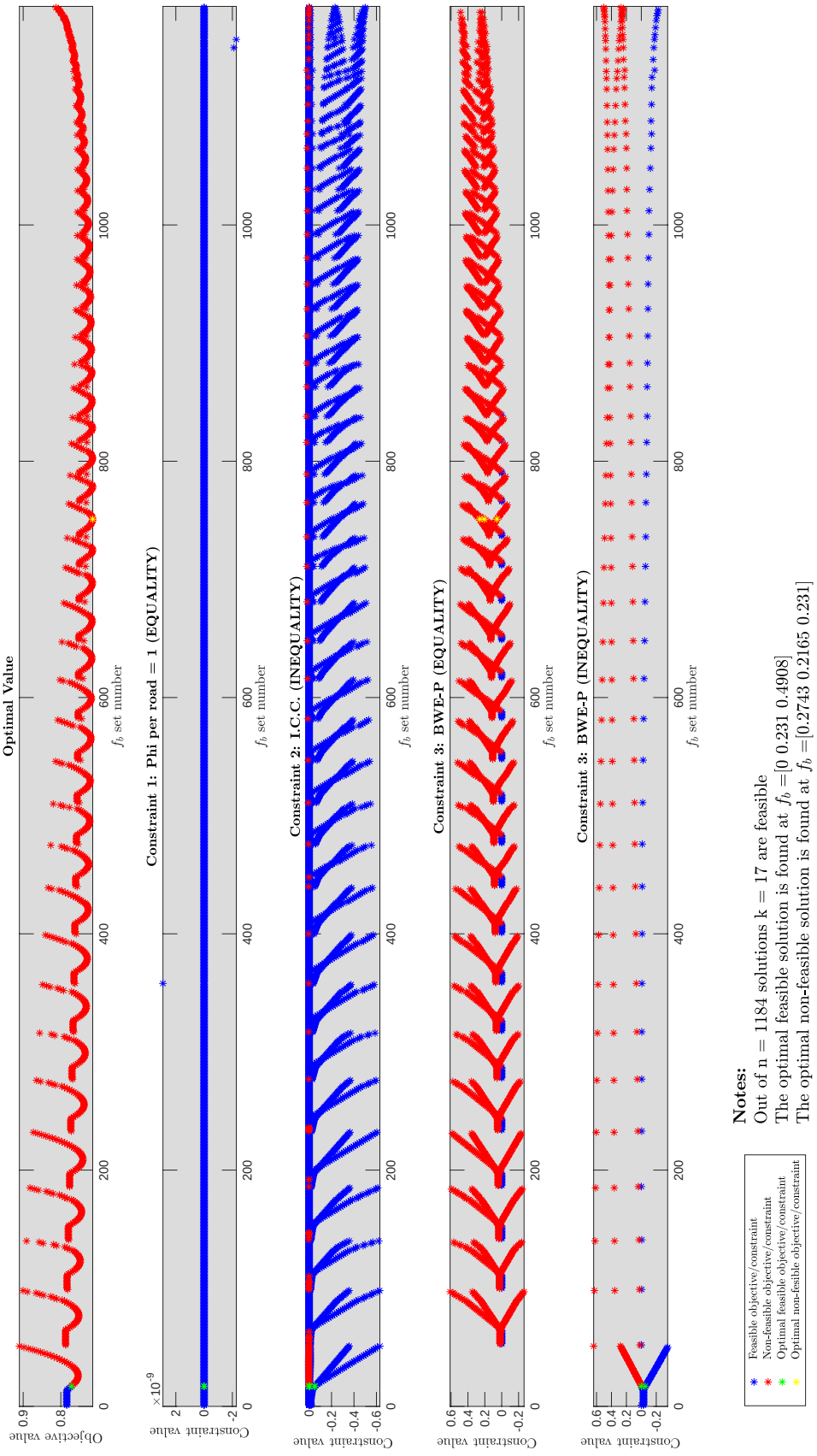

Figure 5: Objective Function Evaluation for a 3-Roads 2-States Case

7.2 4-Roads 2-States Case

Randomly generated constants required to perform a 4-roads 2-states example are shown in Table 5 below.

Constants for the 3-Roads 2-States Case

$$q^{\theta} = \begin{pmatrix} q^{\theta_1} & q^{\theta_2} \end{pmatrix} = \begin{pmatrix} 0.0803 & 0.9197 \end{pmatrix}$$

$$v = \begin{pmatrix} v_b & v_p \end{pmatrix} = \begin{pmatrix} 0.7757 & 0.2243 \end{pmatrix}$$

$$\alpha_j^{\theta} = \begin{pmatrix} \alpha_1^{\theta_1} & \alpha_2^{\theta_1} & \alpha_3^{\theta_1} & \alpha_4^{\theta_1} \\ \alpha_1^{\theta_2} & \alpha_2^{\theta_2} & \alpha_3^{\theta_2} & \alpha_4^{\theta_2} \end{pmatrix} = \begin{pmatrix} 0.4868 & 0.4468 & 0.5085 & 0.8176 \\ 0.4359 & 0.3063 & 0.5108 & 0.7948 \end{pmatrix}$$

$$\beta_j^{\theta} = \begin{pmatrix} \beta_1^{\theta_1} & \beta_2^{\theta_1} & \beta_3^{\theta_1} & \beta_4^{\theta_1} \\ \beta_1^{\theta_2} & \beta_2^{\theta_2} & \beta_3^{\theta_2} & \beta_4^{\theta_2} \end{pmatrix} = \begin{pmatrix} 0.6443 & 0.8116 & 0.3507 & 0.8759 \\ 0.3786 & 0.5328 & 0.9390 & 0.5502 \end{pmatrix}$$

Table 5: Randomly Generated Constants for a 4-Roads 2-States Case

The results of the simulation can be seen in Figure 6 on the next page. Yet again, by looking at the graph displaying the optimal value against the various sets of f_j^b, it can not be confirmed that the feasible set of Equation 13 is the union of finite convex sets. Indeed, it is evident from Figure 6 that a repetitive structure is present for the optimal values of $/phi$ as well as in the way the constraints are violated or satisfied; however, for the same reason as for the 3-roads 2-states case the claim made by Zhu and Savla (2018) can not be validated. Consistent with the results of the 3-roads 2-state, it can be noticed that the *BWE-P* constraint is a particularly rigid constraint that causes infeasibility. Additionally, the graphs for the *BWE-P* constraint and *I.C.C.* suggests again that these constraints behave monotonically, on subsets of the possible flows f_b.

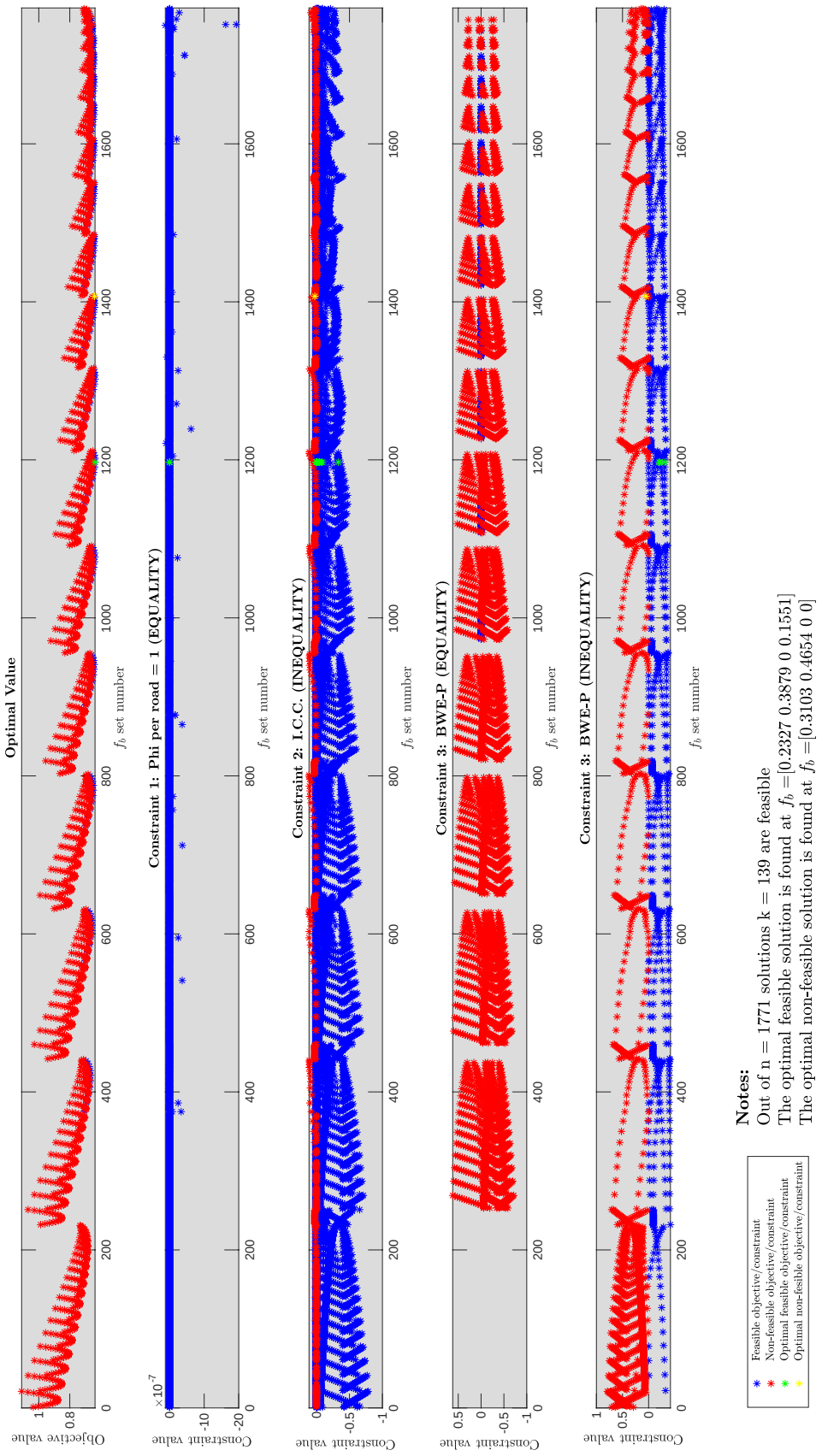

Figure 6: Objective Function Evaluation for a 4-Roads 2-States Case

7.3 General Remarks

After having analyzed the results of the simulations some conclusions can be drawn.
The initial intent of the research was to gain insights into the feasible set of ϕ and demonstrating the claim made by Zhu and Savla (2018) that the non-convex feasible set is the union of finite convex sets. Although the results obtained do give a perception of how the optimization problem behaves, they do not allow to draw reliable conclusions on the feasible set of ϕ. However, insights were identified on how the feasibility at the optimal solution depends on the b-type drivers. Indeed, it can be concluded that the introduction of b-type drivers strongly influences the feasibility of solutions.
Moreover, the clear repetitive structure of both the optimal objective function value and the feasibility constraints lend themselves for exploitation, giving insight into how numerically efficient methods could be constructed to find the optimum solution.
On a deeper note, further research is required in finding a proper way of representing the feasible set of ϕ. The current method found in the research lacks a proper representation of the feasible set. Indeed, this poses a major issue in implementing the mathematical model in any real-life scenario. Nevertheless, the current research may be considered as a first investigation into how to address the problem concerning the mathematical model presented by Zhu and Savla (2018).

8 Future Perspectives

In the following sections the limitations and future possible applications of the studied mathematical model are discussed.

8.1 Shortcomings

The model presented by Zhu and Savla (2018) is indeed a promising solution to the issue of congestion. However, this model contains many shortcomings that are further discussed.
The optimization problem at the core of the mathematical model is based on two game theory concepts: Wardrop Bayesian equilibrium and incentive compatibility. While game theory provides the most satisfying, conclusive information and analysis in simpler games, it has limited practical applications in real life (Cornell University, 2014).
As previously mentioned the model considers two types of drivers: those who use navigation apps and those who do not, p and b type drivers, respectively. However, these drivers can have additional characteristics. For instance, local and foreign drivers could be identified. Local drivers are knowledgeable about the network of roads and the possible states the network can assume. Foreign drivers do not have any prior knowledge about the network. Therefore, by adding these two realistic characteristics about the drivers it is possible to see how the Bayesian equilibrium and incentive compatibility have some limitations.
On the one hand, Bayesian equilibrium does not entirely portray how the b-type drivers behave. In the case where these drivers aim at minimizing their travel time, their decisions are based on their prior knowledge of the state of the road. Despite that, many other rational decisions may be taken by b-type drivers. For instance, the prior knowledge these drivers have could differ from driver to driver. Moreover, instead of minimizing the travel time, a driver's preference could be following roads with better views. Indeed, it could be argued that Bayesian equilibrium still arises in the long run; however, it is best suited for long-term games and not immediate decisions. Regarding the characteristic of being local or foreign, an argument could be made that the most of the b-type drivers are local. Hence, it could happen that foreign drivers do not use navigation apps and decide to move around the network based on their personal preferences.
On the other hand, the incentive compatibility constraint also does not completely portray the behavior of p-type drivers. These are the drivers that use the navigation app. It is reasonable to assume that those who uses navigation apps are foreign drivers since they often have little knowledge about the available roads and the possible states of the network. Therefore, any suggestion given by the navigation app would be followed by these drivers as this is the only source of information the driver has. In contrast, local p-type drivers could also use a navigation app to have more accurate information about how to minimize their travel time. In this case, it is reasonable to say that to make these drivers follow the advice of the navigation app the incentive compatibility constraint has to be satisfied.
It is clear to see how the mathematical model and its constraints have serious limitations. The goal of the drivers is not always the one of minimizing travel time and their knowledge about the network is not only based on the advice they receive from the navigation app. Many other factors need to be taken into account. Game theory is based on the assumption that every player behaves rationally; however, people are not always rational

and unpredictable behaviors could arise.
From a technical view, if the model was to be used by navigation apps, it would require that every navigation app were to share the same database of information of the network. Such a model only works if the information about all drivers is available. Indeed, these could pose a major challenge as navigation apps would need to collaborate.
Additionally, the approach of utilizing a brute force method to find an optimal solution to the optimization problem has its limitations. In the case where the networks become larger, it is required to run the optimization problem a significant number of times, making it numerically intensive.

8.2 Applicability

Traffic flow problems have been investigated for decades (Bando et al., 1995). In particular, one of the most prominent issue is traffic congestion, which does not only result in long delays for individual drivers but also in a large accident rate (Smulders, 1990). Increasing the capacity of roads by adding additional lanes is not always an adequate or preferable solution to traffic congestion. However, alternative approaches focusing on controlling the flow of vehicles by means of signals and information have been considered (Smulders, 1990). Along with prolonged travel time and higher risk of accidents, traffic congestion brings other adverse effects such as carbon dioxide emissions. Indeed, the transportation sector plays a significant role in carbon emissions. In their paper, Barth and Boriboonsomsin (2008), show how in a typical traffic condition in Southern California, carbon emissions could be reduced by up to almost 20% through reducing traffic congestion and allowing traffic to flow at better speeds. Undoubtedly, traffic congestion decreases the efficiency of the transportation sector and increases air pollution. However, if drivers were provided with accurate information, the road network efficiency could be improved significantly (Yuan et al., 2014).
Besides its negative effects on the drivers and environment, traffic congestion also has a financial impact on nations. The European UNITE project estimated the costs of traffic congestion in the UK to be £15 billion/year, equivalent to 1.5% of GDP. Furthermore, for France and Germany these estimates were of 1.3% and 0.9% of GDP respectively (Nash et al., 2003). Additionally, according to the Texas Transportation Institute, in 2007, traffic congestion in the USA caused an estimated 4.2 billion hours of travel delay and 2.8 billion gallons of extra fuel consumption with a total cost of $87 billion (Schrank et al., 2009).
The use of information as a solution to traffic congestion has already been studied (Arnott et al., 1991). Indeed, navigation apps help relieving congestion; however, as discussed in the previous chapters, their current behaviour only focuses on the individual driver and does not take into account the overall state of the network. Therefore, navigation systems have the potential of further improving traffic congestion by adopting approaches that reduce travel time for individual drivers while relieving traffic congestion. Such an approach is the one presented by Zhu and Savla (2018), which has the potential of being implemented in navigation apps.
Since the primary goal of any Ministry of Transport is to reduce congestion, carbon emissions, and improve road safety, the introduction of such a model into navigation systems would be extremely beneficial to any nation (Government of the Netherlands, 2021; U.S.

Department of Transportation, 2020; U.K. Department for Transport, 2021). Governments could urge the CEOs of navigation apps to utilize the model in urban areas where the issue of congestion is frequent. Metropolitan areas are a perfect fit for the model as many parallel roads are available to reach the same destination. Instead of expanding transportation capacity by introducing new roads, utilizing information as a way of coordinating traffic could benefit countries on different sides such as reducing the nation environmental impact, improving traffic flow of vehicles in urban areas, and reducing the major costs resulting from traffic congestion.

Conclusion

This research aimed at translating the mathematical model presented by Zhu and Savla (2018) into an optimization algorithm that supports the advancement of navigation apps. Such a model intends to reduce traffic congestion by routing drivers on various roads. The complexity of the mathematical model comes from the non-convexity of the optimization problem presented. To investigate the non-convex feasible set, a brute force method was implemented. Even though brute force methods are a programming style that does not include any shortcuts to improve performance, it has to be mentioned that these methods always return the correct result, albeit slowly. Therefore, implementing a brute force approach in solving the optimization problem developed by Zhu and Savla (2018) may be considered as the first step in developing an efficient algorithm that could solve the non-convex optimization problem of study.

The initial technical goal of this research was to gain insights into the feasible set of the optimization problem and confirm the claim made by Zhu and Savla (2018) that the non-convex feasible set is the union of few convex sets. As seen from the research, although the results give some insights on how the feasibility at the optimal solution depends on b-type drivers, it was not possible to draw the desired conclusions. To better understand the implications of these results, future studies could address different ways of representing the feasible set.

Additionally, possible limitations of the mathematical model have been discussed. The main shortcoming introduced regards the characteristics of drivers. Two types of drivers are distinguished by Zhu and Savla (2018), p-type and b-type drivers; however, many additional characteristics could be introduced in order to obtain a more realistic model. For example, drivers could be identified as local or foreign, where local drivers have higher knowledge about the state of the network than foreign drivers who have little knowledge.

Despite the complications found at representing the feasible set at its optimal, this research contributes towards finding a solution to the issue of congestion by the use of mathematical models such as the one of Zhu and Savla (2018). The current algorithms utilized by navigation apps focus on minimizing travel time for single users. Indeed, modern navigation apps are developed to assist drivers in reaching their destination in the least time possible. However, such an approach only focuses on the selfish needs of users and does not take into account the social welfare. Better algorithms could be implemented in order to satisfy single users and mitigate traffic congestion. Such algorithms would not only help traffic flow, but also enable a larger, positive impact on the issues that traffic congestion brings, such as the higher risk of accidents, carbon dioxide emissions, and prolonged travel time.

Bibliography

Arnott, R., De Palma, A., and Lindsey, R. (1991). Does providing information to drivers reduce traffic congestion? *Transportation Research Part A: General*, 25(5):309–318.

Balakrishna, R., Ben-Akiva, M., Bottom, J., and Gao, S. (2013). Information impacts on traveler behavior and network performance: State of knowledge and future directions. In *Advances in dynamic network modeling in complex transportation systems*, pages 193–224. Springer.

Bando, M., Hasebe, K., Nakayama, A., Shibata, A., and Sugiyama, Y. (1995). Dynamical model of traffic congestion and numerical simulation. *Physical review E*, 51(2):1035.

Barth, M. and Boriboonsomsin, K. (2008). Real-world carbon dioxide impacts of traffic congestion. *Transportation Research Record*, 2058(1):163–171.

Cabannes, T., Vincentelli, M. A. S., Sundt, A., Signargout, H., Porter, E., Fighiera, V., Ugirumurera, J., and Bayen, A. M. (2017). The impact of gps-enabled shortest path routing on mobility: a game theoretic approach 2. *University of California, Berkeley*.

Cornell University (2014). Why study game theory when it has limited practical applications in real life? Available at `https://blogs.cornell.edu/info2040/2014/09/26/why-study-game-theory-when-it-has-limited-practical-applications-in-real-life/`.

Correa, J. R., Schulz, A. S., and Stier-Moses, N. E. (2004). Selfish routing in capacitated networks. *Mathematics of Operations Research*, 29(4):961–976.

Government of the Netherlands (2021). Mobility, public transport and road safety. Available at `https://www.government.nl/topics/mobility-public-transport-and-road-safety`.

Hardy, Q. (2012). Facing fees, some sites are bypassing google maps. *New York Times*. Available at `https://archive.nytimes.com/query.nytimes.com/gst/fullpage-9904E4D61E3AF933A15750C0A9649D8B63.html`.

Hermans, C. and Schoeman, W. (2015). Practice-oriented research in service of designing interventions. *Acta Theologica*, pages 26–44.

Herrera, J. C., Work, D. B., Herring, R., Ban, X. J., Jacobson, Q., and Bayen, A. M. (2010). Evaluation of traffic data obtained via gps-enabled mobile phones: The mobile century field experiment. *Transportation Research Part C: Emerging Technologies*, 18(4):568–583.

Kamenica, E. and Gentzkow, M. (2011). Bayesian persuasion. *American Economic Review*, 101(6):2590–2615.

Nash, C., Sansom, T., and Matthews, B. (2003). Final report for publication. *UNITE (UNIfication of accounts and marginal costs for Transport Efficiency). Leeds: University of Leeds*.

Nash, J. F. et al. (1950). Equilibrium points in n-person games. *Proceedings of the national academy of sciences*, 36(1):48–49.

Patriksson, M. (2015). *The traffic assignment problem: models and methods*. Courier Dover Publications.

Russell, E. (2019). 9 things to know about google's maps data: Beyond the map. *Google Cloud*. Available at `https://cloud.google.com/blog/products/maps-platform/9-things-know-about-googles-maps-data-beyond-map`.

Schmidt, M. and Weiser, P. (2012). Web mapping services: development and trends. In *Online maps with APIs and WebServices*, pages 13–21. Springer.

Schrank, D., Lomax, T., et al. (2009). 2009 urban mobility report.

Smulders, S. (1990). Control of freeway traffic flow by variable speed signs. *Transportation Research Part B: Methodological*, 24(2):111–132.

Srivatsa Srinivas, S. and Gajanand, M. (2017). Vehicle routing problem and driver behaviour: a review and framework for analysis. *Transport reviews*, 37(5):590–611.

U.K. Department for Transport (2021). Available at `https://www.gov.uk/government/organisations/department-for-transport/about`.

U.S. Department of Transportation (2020). Available at `https://www.transportation.gov/about`.

Wardrop, J. G. (1952). Road paper. some theoretical aspects of road traffic research. *Proceedings of the institution of civil engineers*, 1(3):325–362.

Wheatcraft, L. S. (2012). Thinking ahead to verification and validation. *Requirements Experts*.

Yuan, Q., Liu, Z., Li, J., Zhang, J., and Yang, F. (2014). A traffic congestion detection and information dissemination scheme for urban expressways using vehicular networks. *Transportation Research Part C: Emerging Technologies*, 47:114–127.

Zhu, Y. and Savla, K. (2018). On the stability of optimal bayesian persuasion strategy under a mistrust dynamics in routing games. In *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 92–99. IEEE.

Appendices

MATLAB Optimization Algorithm

The MATLAB code used to implement the mathematical model is shown below and is also available for download on GitHub. This code solves the convex simplification of the optimization problem and the non-convex case. Ten MATLAB files have been created to collect the objective function, constraints, constants, executable main file, plots and more. In order to clarify the functioning of the algorithm, each file contains useful comments. In order to set and execute the algorithm the file *main.m* is used.

Executable File

```
%%% START
clear all

%%% STEP 1: Select the constants to use (uncomment the line)

% Example vb = 0 from the paper by Zhu and Savla (2018) --> convex
% [n_links, n_states, constants] = constants_ex_1();

% Example vb ~= 0 from the paper by Zhu and Savla (2018)--> non-convex
% [n_links, n_states, constants] = constants_ex_2();

% Generate random constants by choosing number of links and states
% n_links = 3;
% n_states = 2;
% [constants] = constants_rdm(n_links, n_states);


%%% STEP 2: Generate an initial guess
phi0 = rand(n_states, n_links);


%%% STEP 3: Choose precision of solutions
%%% (for fast solutions 20 < precision < 60)
precision = 25;


%%% STEP 5: Choose tolerances for constraints
tol_eq = 1e-4;
tol_ineq = 1e-4;


%%% STEP 4: Choose options for fmincon.m
options = optimoptions('fmincon');
options.MaxFunctionEvaluations = 3.0e+06;
options.Display = 'off';
options.Algorithm = 'sqp';


%%% STEP 6: Compute solutions
tic
[phi, objective, fb, const_phi, const_icc, const_bwep] ...
    = algo(phi0, precision, n_states, n_links, constants, ...
    tol_eq, tol_ineq, options);
toc
```

Listing 1: main.m

Constants Convex Example

```
function [n_links, n_states, constants] = constants_ex_1()

% Number of links and states
n_links = 2;
n_states = 2;

% q
q1 = 0.1481;
q2 = 0.8519;
q = [q1, q2];

% vb, vp
vb = 0;
vp = 1;

% alpha
a11 = 0.0011;
a12 = 0.0055;
a21 = 0.003;
a22 = 0.0043;
a = [a11, a12; a21, a22];

% beta
b11 = 2.5485;
b12 = 2.1929;
b21 = 1.5363;
b22 = 2.2671;
b = [b11, b12; b21, b22];

% Constant
constants = {q, vb, vp, a, b};
```

Listing 2: constants_ex_1.m

Constants Non-Convex Example

```
function [n_links, n_states, constants] = constants_ex_2()

% Number of links and states
n_links = 2;
n_states = 2;

% q
q1 = 0.1;
q2 = 0.9;
q = [q1, q2];

% vb, vp
vb = 0.4;
vp = 0.6;

% alpha
a11 = 2;
a12 = 1;
a21 = 1;
a22 = 1;
a = [a11, a12; a21, a22];

% beta
b11 = 1;
b12 = 1;
b21 = 1;
b22 = 1;
b = [b11, b12; b21, b22];

% Constant
constants = {q, vb, vp, a, b};
```

Listing 3: constants_ex_2.m

Random Generation of Constants

```
function [constants] = constants_rdm(n_links, n_states)

% q
q = rand(1, n_states);
q = q / sum(q);

% vb, vp
vb = rand(1);
vp = 1 - vb;

% alpha
a = rand(n_states,n_links);

% beta
b = rand(n_states,n_links);

% Constant
constants = {q, vb, vp, a, b};
```

Listing 4: constants_rdm.m

Brute Force Algorithm

```
function [phi, objective, fb, const_phi, const_icc, const_bwep] ...
    = algo(phi0, precision, n_states, n_links, constants, tol_eq, tol_ineq, options)

% This algorithm attemps at solving the optimization problem presented in
% the paper by Zhu and Savla (2018) "On the stability of optimal bayesian
% persuasion strategy under a mistrust dynamics in routing games".
%
% Since the non-convex problem is the union of convex sets, a brute force
% method is used.
%
%
%
%                INPUTS
%       phi0       -->     Initial guess
%       precision  -->     Number of possible f_b sets
%       n_states   -->     Number of states
%       n_links    -->     Number of links
%       constants  -->     It is a 1x5 cell containing {q, vb, vp, a, b}
%       tol_eq     -->     Tolerance for equality constraints
%       tol_ineq   -->     Tolerance for inequality constraints
%       options    -->     Options for the solver fmincon.m
%
%                OUTPUTS
%       phi        -->    1x2 cell with feasible and non feasible solutions
%                         the first entry contains the feasible solutions
%                         and the second entry the non feasible solutions
%       objective  -->    1x2 cell similar to phi, feasible solutions in
%                         the first entry and non feasible in the second
%       fb         -->    Feasible pairs of fb related to each solution
%       const_phi  -->    1x2 cell containing the values of the first
%                         constraint. First entry are the feasible values
%                         and second entry are the non feasible values
%       const_icc  -->    1x2 cell containing the values of the second
%                         constraint. First entry are the feasible values
%                         and second entry are the non feasible values
%       const_bwep -->    1x2 cell containing the values of the third
%                         constraint. First entry are the feasible values
%                         and second entry are the non feasible values
%
%       This function also outputs some graphes.
%       For more info see myplots.m

% ALGORITHM
% Assign constants
[~, vb, ~, ~, ~] = constants{:};

% Define upper and lower bound for fmincon.m
lb = 0 * ones(n_states, n_links);
ub = 1 * ones(n_states, n_links);

% Create all possibilities of fb
if vb == 0
    fb = zeros(1,n_links);
```

```
else
    step_size = vb/precision;
    f = 0:step_size:vb;
    f = permn(f,n_links);
    fb = f(sum(f,2)==vb,:);
end

% Solve for each set of fb ---> brute force method
n = size(fb,1);
k = 0;

phi_opt = cell(1,n);
phi_no_opt = cell(1,n);

obj_opt = NaN(1,n);
obj_no_opt = NaN(1,n);

c = cell(1,n);
ceq = cell(1,n);

% Setting up wait bar
h = waitbar(0,'Please wait ...','Name','Calculating Solutions',...
    'CreateCancelBtn','setappdata(gcbf,''canceling'',1)');

setappdata(h,'canceling',0);

for i = 1:n
    % Check for clicked Cancel button
    if getappdata(h,'canceling')
        break
    end
    waitbar(i/n, h);

    % Assign functions
    obj_fun = @(phi) obj(phi, fb(i,:), constants);
    nlcon_fun = @(phi) nlcon(phi, fb(i,:), n_links, n_states, constants);

    % Find optimal set phi
    phi_opt{i} = fmincon(obj_fun, phi0, [], [], [], [], lb, ub, ...
        nlcon_fun, options);
    phi_no_opt{i} = NaN(n_states, n_links);

    % Find objective optimal
    obj_opt(i) = obj(phi_opt{i}, fb(i,:), constants);
    obj_no_opt(i) = NaN;

    % Collect feasible and non feasible solutions with objective
    [c_opt, ceq_opt] = nlcon(phi_opt{i}, fb(i,:), n_links, n_states, ...
        constants);
    if any(c_opt > tol_ineq) || any(abs(ceq_opt) > tol_eq)
        obj_no_opt(i) = obj_opt(i);
        obj_opt(i) = NaN;

        phi_no_opt{i} = phi_opt{i};
```

```
        phi_opt{i} = NaN(n_states, n_links);
    else
        k = k + 1;
        phi0 = phi_opt{i};
    end

    % Collect the values of constraints
    c{i} = c_opt;
    ceq{i} = ceq_opt;

end
delete(h);

% Transform c and ceq into matrix form for plotting
% If cell has only one entry
if length(c) ==  1
    c = c{1};
    ceq = ceq{1}';
else
    c = padcat(c{:});
    ceq = padcat(ceq{:})';
end

% Save the feasible and non feasible solutions
phi = {phi_opt, phi_no_opt};

% Save the feasible and non feasible objectives
objective = {obj_opt, obj_no_opt};

% Save the feasible and non feasible constraints
c_yes = c;
c_no = c;
c_yes(c_yes > tol_ineq) = NaN;
c_no(c_no <= tol_ineq) = NaN;

ceq_yes = ceq;
ceq_no = ceq;
ceq_yes(abs(ceq_yes) > tol_eq) = NaN;
ceq_no(abs(ceq_no) <= tol_eq) = NaN;

% Constraint 1: Phi per road
ceq_const_1_yes = ceq_yes(:, 1:n_states);
ceq_const_1_no = ceq_no(:, 1:n_states);

const_phi = {ceq_const_1_yes, ceq_const_1_no};

% Constraint 3: I.C.C.
n_const_2 = factorial(n_links)/(factorial(n_links - 2));
c_const_2_yes = c_yes(:, 1:n_const_2);
c_const_2_no = c_no(:, 1:n_const_2);

const_icc = {c_const_2_yes, c_const_2_no};

% Constraint 4: BWE-P
```

```
ceq_const_3_yes = ceq_yes(:, n_states+1:end);
ceq_const_3_no = ceq_no(:, n_states+1:end);

c_const_3_yes = c_yes(:, n_const_2+1:end);
c_const_3_no = c_no(:, n_const_2+1:end);

const_bwep = {ceq_const_3_yes, ceq_const_3_no, c_const_3_yes, c_const_3_no};

% Plot results
myplots(phi, fb, objective, const_phi, const_icc, const_bwep, ...
    n, k, n_links, n_states);
```

Listing 5: algo.m

Objective Function

```
function [J] = obj(phi, fb, constants)

% Assign constants
[q, ~, vp, a, b] = constants{:};

% fp
fp = vp * phi;

% fs
fs = 1*(fb + fp);   %instead of 1 use 202.6 for example (1)

% Delay function l
l = a .* fs + b;

% Social cost J1
J1 = fs .* l;
J1 = sum(J1,2);

% Expected social cost J
J = (q * J1);
```

Listing 6: obj.m

Constraints

```
function [c, ceq] = nlcon(phi, fb, n_links, n_states, constants)

% Assign constants
[q, vb, vp, a, b] = constants{:};

% fp
fp = vp * phi;

% fs
fs = 1*(fb + fp);   %instead of 1 use 202.6 for example (1)

% Constraint 1 --> Phi = 1
ceq = 1 - sum(phi,2);

% Constraint 3 --> I.C.C.
index_c = 0;

for j=1:n_links
    A1 = q * ( a(:,j) .* phi(:,j) .* fs(:,j) );
    A2 = q * ( b(:,j) .* phi(:,j));
    for k=1:n_links
        if k ~= j
            index_c = 1 + index_c;
            B1 = q * ( a(:,k) .* phi(:,j) .* fs(:,k) );
            B2 = q * ( b(:,k) .* phi(:,j));

            c(index_c) = (A1 + A2) - (B1 + B2);
        end
    end
end

% Constraint 4 --> BWE-P
if vb ~=0
    index_ceq = n_states;
    if any(fb==0) %There is some zero fb flow
        % Find index of where fb is zero and non zero
        idx_zero = find(fb==0);
        idx_no_zero = find(fb~=0);

        % Calculate how many zeros and non zeros there are in fb
        n_zero = length(idx_zero);
        n_no_zero = length(idx_no_zero);
        if n_no_zero > 1 % If there are at least 2 non zero
            for i = 1:n_no_zero
                j = idx_no_zero(i);

                A1 = q * ( a(:,j) .* fs(:,j) );
                A2 = q * ( b(:,j) );
                for w = 1:n_no_zero
                    k = idx_no_zero(i);
                    if k ~= j && k < j
                        index_ceq = index_ceq + 1;

```

```
                        B1 = q * ( a(:,k) .* fs(:,k) );
                        B2 = q * ( b(:,k) );

                        ceq(index_ceq) = (A1 + A2) - (B1 + B2);
                    end
                end
            end
        end

        for i = 1:n_no_zero
            j = idx_no_zero(i);

            A1 = q * ( a(:,j) .* fs(:,j) );
            A2 = q * ( b(:,j) );
            for w = 1:n_zero
                k = idx_zero(w);

                index_c = 1 + index_c;

                B1 = q * ( a(:,k) .* fs(:,k) );
                B2 = q * ( b(:,k) );

                c(index_c) = (A1 + A2) - (B1 + B2);
            end
        end

    else % There is b flow on every road
        for j = 1:n_links
            A1 = q * ( a(:,j) .* fs(:,j) );
            A2 = q * ( b(:,j) );
            for k = 1:n_links
                if j < k
                    index_ceq = 1 + index_ceq;

                    B1 = q * ( a(:,k) .* fs(:,k) );
                    B2 = q * ( b(:,k) );

                    ceq(index_ceq) = (A1 + A2) - (B1 + B2);
                end
            end
        end
    end
end
```

Listing 7: nlcon.m

Plotting Function

```
function [fig1, fig2] = myplots(phi, fb, objective, const_phi, ...
    const_icc, const_bwep, n, k, n_links, n_states)

% This function is used to plot results obtained by algo.m
%
% Figure 1: this figure contains four different plots
%           Plot 1     : the objective function is plotted for each
%                        possible set of fb
%           Plot 2     : The value of constraint 1 (Phi = 1) is plotted for
%                        each set of fb
%           Plot 3     : The value of constraint 2 (I.C.C) is plotted for
%                        each set of fb
%           Plot 4 & 5 : The value of constraint 3 (BWE-P) is plotted for
%                        each set of fb.
%
% Figure 2: this figure is used to plot the results of phi for each state.
%           This figure is created only for the 2-links and 3-links case

% Setting up wait bar
h = waitbar(0,'Please wait ...','Name','Plotting Graphs');

% FIGURE 1: Objective values and constraints
fig1 = figure('Name','Objective Values & Constraints','NumberTitle', ...
    'off','units','normalized','outerposition',[0 0 1 1], ...
    'DefaultAxesFontSize', 12);

x = 1:n;

[~, idx_opt] = min(objective{1});
[~, idx_no_opt] = min(objective{2});


%%% Plot 1: Objective
obj_opt = objective{1};
obj_no_opt = objective{2};

subplot(6,1,1)
plot(x, obj_opt,'b*')
hold on
plot(x, obj_no_opt,'r*')
hold on
plot(idx_opt, obj_opt(idx_opt),'g*')
hold on
plot(idx_no_opt, obj_no_opt(idx_no_opt),'y*')
xlim([0 n+1])
xlabel("$f_b$ set number" + newline + "   ",'Interpreter','latex')
ylabel('Objective value','Interpreter','latex')
title('\textbf{Optimal Value}','Interpreter','latex')
set(gca,'Color','#DCDCDC')


%%% Plot 2: Constraint 1 --> Phi
const_phi_yes = const_phi{1};
```

```
const_phi_no = const_phi{2};

subplot(6,1,2)
plot(x, const_phi_yes,'b*')
hold on
plot(x, const_phi_no,'r*')
hold on
plot(idx_opt, const_phi_yes(idx_opt, :),'g*')
hold on
plot(idx_no_opt, const_phi_no(idx_no_opt, :),'y*')
xlim([0 n+1])
xlabel("$f_b$ set number" + newline + "   ",'Interpreter','latex')
ylabel('Constraint value','Interpreter','latex')
title('\textbf{Constraint 1: Phi per road = 1 (EQUALITY)}', ...
    'Interpreter','latex')
set(gca,'Color','#DCDCDC')


% Plot 3: Constraint 2 ---> I.C.C.
const_icc_yes = const_icc{1};
const_icc_no = const_icc{2};

subplot(6,1,3)
plot(x, const_icc_yes,'b*')
hold on
plot(x, const_icc_no,'r*')
hold on
plot(idx_opt, const_icc_yes(idx_opt, :),'g*')
hold on
plot(idx_no_opt, const_icc_no(idx_no_opt, :),'y*')
xlim([0 n+1])
xlabel("$f_b$ set number" + newline + "   ",'Interpreter','latex')
ylabel('Constraint value','Interpreter','latex')
title('\textbf{Constraint 2: I.C.C. (INEQUALITY)}','Interpreter','latex')
set(gca,'Color','#DCDCDC')


% Plot 4 & 5: Constraint 3 ---> BWE-P
ceq_const_bwep_yes = const_bwep{1};
ceq_const_bwep_no = const_bwep{2};

c_const_bwep_yes = const_bwep{3};
c_const_bwep_no = const_bwep{4};


% Plot 4
if ~all(fb==0)
    subplot(6,1,4)
    plot(x, ceq_const_bwep_yes,'b*')
    hold on
    plot(x, ceq_const_bwep_no,'r*')
    hold on
    plot(idx_opt, ceq_const_bwep_yes(idx_opt, :),'g*')
    hold on
```

```
    plot(idx_no_opt, ceq_const_bwep_no(idx_no_opt, :),'y*')
    xlim([0 n+1])
    xlabel("$f_b$ set number" + newline + "   ",'Interpreter','latex')
    ylabel('Constraint value','Interpreter','latex')
    title('\textbf{Constraint 3: BWE—P (EQUALITY)}','Interpreter','latex')
    set(gca,'Color','#DCDCDC')


    % Plot 5
    subplot(6,1,5)
    plot(x, c_const_bwep_yes,'b*');
    hold on;
    plot(x, c_const_bwep_no,'r*');
    hold on;
    plot(idx_opt, c_const_bwep_yes(idx_opt, :),'g*');
    hold on;
    plot(idx_no_opt, c_const_bwep_no(idx_no_opt, :),'y*');
    xlim([0 n+1])
    xlabel("$f_b$ set number" + newline + "   ",'Interpreter','latex')
    ylabel('Constraint value','Interpreter','latex')
    title('\textbf{Constraint 3: BWE—P (INEQUALITY)}','Interpreter','latex')
    set(gca,'Color','#DCDCDC')
end


% Notes Figure 1
subplot(6,1,6);
plot(NaN,NaN,'b*');
hold on;
plot(NaN,NaN,'r*');
hold on;
plot(NaN,NaN,'g*');
hold on;
plot(NaN,NaN,'y*');
hold on;
legend('Feasible objective/constraint', ...
    'Non—feasible objective/constraint', ...
    'Optimal feasible objective/constraint',...
    'Optimal non—fesible objective/constraint');
legend('Location','westoutside', 'Interpreter','latex')
hold on;

if n == k
    fb_opt = mat2str(fb(idx_opt,:),4);
    msg(1) = "All solutions are feasible (n="+n+")";
    msg(2) = "The optimal feasible solution is found at $f_b =$"+ fb_opt;
elseif k == 0
    msg(1) = "There are no feasible solutions";
else
    fb_opt = mat2str(fb(idx_opt,:),4);
    fb_no_opt = mat2str(fb(idx_no_opt,:),4);
    msg(1) = "Out of n = " + n + " solutions k = " + k + " are feasible";
    msg(2) = "The optimal feasible solution is found at $f_b =$"+ fb_opt;
    msg(3) = "The optimal non—feasible solution is found at $f_b=$"+ ...
```

```
        fb_no_opt;
end

notes = msg;

set(gca,'XColor', 'none','YColor','none', 'Color', 'none')
text(0,0.5, ['\textbf{Notes:}', notes], 'Interpreter','latex',...
    'FontSize',14)


% FIGURE 2
if n_links == 2
    fig2 = figure('Name','Phi Solutions','NumberTitle','off', ...
        'units','normalized','outerposition',[0 0 0.8 0.8], ...
        'DefaultAxesFontSize', 12);

    for j=1:n_states
        for i=1:n
            x_phi_opt(i) = phi{1}{i}(j,1);
            y_phi_opt(i) = phi{1}{i}(j,2);

            x_phi_no_opt(i) = phi{2}{i}(j,1);
            y_phi_no_opt(i) = phi{2}{i}(j,2);
        end
        subplot(1,n_states,j)
        plot(x_phi_opt, y_phi_opt, 'b*');
        hold on;
        plot(x_phi_no_opt, y_phi_no_opt, 'r*');
        hold on;
        title(sprintf('\\textbf{State %d: $\\phi^{\\theta_%d}_j$}', j, ...
            j) ,'Interpreter','latex')
        xlabel(sprintf('$\\phi^{\\theta_%d}_1$', j),'Interpreter','latex')
        ylabel(sprintf('$\\phi^{\\theta_%d}_2$', j),'Interpreter','latex')
        set(gca,'Color','#DCDCDC')
    end
    axP = get(gca,'Position');
    legend('Feasible solution', 'Non-feasible solution','Interpreter', ...
        'latex')
    legend('Location','northwestoutside')
    set(gca, 'Position', axP)

elseif n_links == 3
    fig2 = figure('Name','Phi Solutions','NumberTitle','off', ...
        'units','normalized','outerposition',[0 0 0.8 0.8], ...
        'DefaultAxesFontSize', 12);

    for j=1:n_states
        for i=1:n
            x_phi_opt(i) = phi{1}{i}(j,1);
            y_phi_opt(i) = phi{1}{i}(j,2);
            z_phi_opt(i) = phi{1}{i}(j,3);

            x_phi_no_opt(i) = phi{2}{i}(j,1);
            y_phi_no_opt(i) = phi{2}{i}(j,2);
```

```
            z_phi_no_opt(i) = phi{2}{i}(j,3);
        end
        subplot(1,n_states,j)
        plot3(x_phi_opt, y_phi_opt, z_phi_opt, 'b*');
        hold on;
        plot3(x_phi_no_opt, y_phi_no_opt, z_phi_no_opt, 'r*');
        hold on;
        title(sprintf('\\textbf{State %d: $\\phi^{\\theta_%d}_j$}', j,...
            j) ,'Interpreter','latex')
        xlabel(sprintf('$\\phi^{\\theta_%d}_1$', j),'Interpreter','latex')
        ylabel(sprintf('$\\phi^{\\theta_%d}_2$', j),'Interpreter','latex')
        zlabel(sprintf('$\\phi^{\\theta_%d}_3$', j),'Interpreter','latex')
        set(gca,'Color','#DCDCDC')
    end
    axP = get(gca,'Position');
    legend('Feasible solution', 'Non-feasible solution','Interpreter', ...
        'latex')
    legend('Location','northwestoutside')
    set(gca, 'Position', axP)
end
delete(h);
```

Listing 8: myplots.m

Matrix Manipulation Function

```
function [M, TF] = padcat(varargin)
% PADCAT - concatenate vectors with different lengths by padding with NaN
%
%   M = PADCAT(V1, V2, V3, ..., VN) concatenates the vectors V1 through VN
%   into one large matrix. All vectors should have the same orientation,
%   that is, they are all row or column vectors. The vectors do not need to
%   have the same lengths, and shorter vectors are padded with NaNs.
%   The size of M is determined by the length of the longest vector. For
%   row vectors, M will be a N-by-MaxL matrix and for column vectors, M
%   will be a MaxL-by-N matrix, where MaxL is the length of the longest
%   vector.
%
%   Examples:
%      a = 1:5 ; b = 1:3 ; c = [] ; d = 1:4 ;
%      padcat(a,b,c,d) % row vectors
%         % ->   1     2     3     4     5
%         %      1     2     3   NaN   NaN
%         %    NaN   NaN   NaN   NaN   NaN
%         %      1     2     3     4   NaN
%      CC = {d.' a.' c.' b.' d.'} ;
%      padcat(CC{:}) % column vectors
%         %      1     1   NaN     1     1
%         %      2     2   NaN     2     2
%         %      3     3   NaN     3     3
%         %      4     4   NaN   NaN     4
%         %    NaN     5   NaN   NaN   NaN
%
%   [M, TF] = PADCAT(..) will also return a logical matrix TF with the same
%   size as R having true values for those positions that originate from an
%   input vector. This may be useful if any of the vectors contain NaNs.
%
%   Example:
%       a = 1:3 ; b = [] ; c = [1 NaN] ;
%       [M,tf] = padcat(a,b,c)
%       % find the original NaN
%       [Vev,Pos] = find(tf & isnan(M))
%       % -> Vec = 3 , Pos = 2
%
%   This second output can also be used to change the padding value into
%   something else than NaN.
%
%       [M, tf] = padcat(1:3,1,1:4)
%       M(~tf) = 99 % change the padding value into 99
%
%   Scalars will be concatenated into a single column vector.
%
%   See also CAT, RESHAPE, STRVCAT, CHAR, HORZCAT, VERTCAT, ISEMPTY
%            NONES, GROUP2CELL (Matlab File Exchange)

% Example figure created using:
%   C = arrayfun(@(x) ones(1,randi([10 100],1,1)),1:40,'un',0) ;
%   pcolor(padcat(C{:}))

```

```
% for Matlab 2008 and up (last tested in R2018a)
% version 1.4 (dec 2018)
% (c) Jos van der Geest
% email: samelinoa@gmail.com

% History
% 1.0 (feb 2009) created
% 1.1 (feb 2011) improved comments
% 1.2 (oct 2011) added help on changing the padding value into something
%     else than NaN
% 1.3 (feb 2016) updated contact info
% 1.4 (dec 2018) fixed minor code warnings

% Acknowledgements:
% Inspired by padadd.m (feb 2000) Fex ID 209 by Dave Johnson

narginchk(1,Inf) ;

% check the inputs
SZ = cellfun(@size,varargin,'UniformOutput',false) ; % sizes
Ndim = cellfun(@ndims,varargin) ; %

if ~all(Ndim==2)
    error([mfilename ':WrongInputDimension'], ...
        'Input should be vectors.') ;
end

TF = [] ; % default second output so we do not have to check all the time

% for 2D matrices (including vectors) the size is a 1-by-2 vector
SZ = cat(1,SZ{:}) ;
maxSZ = max(SZ) ;    % probable size of the longest vector
% maxSZ equals :
%  - [1 1] for all scalars input
%  - [X 1] for column vectors
%  - [1 X] for all row vectors
%  - [X Y] otherwise (so padcat will not work!)

if ~any(maxSZ == 1)  % hmm, not all elements are 1-by-N or N-by-1
    % 2 options ...
    if any(maxSZ==0)
        % 1) all inputs are empty
        M  = [] ;
        return
    else
        % 2) wrong input
        % Either not all vectors have the same orientation (row and column
        % vectors are being mixed) or an input is a matrix.
        error([mfilename ':WrongInputSize'], ...
            'Inputs should be all row vectors or all column vectors.') ;
    end
end

if nargin == 1
```

```
    % single input, nothing to concatenate ..
    M = varargin{1} ;
else
    % Concatenate row vectors in a row, and column vectors in a column.
    dim = (maxSZ(1)==1) + 1 ;      % Find out the dimension to work on
    X = cat(dim, varargin{:}) ;    % make one big list

    % we will use linear indexing, which operates along columns. We apply a
    % transpose at the end if the input were row vectors.

    if maxSZ(dim) == 1
        % if all inputs are scalars, ...
        M = X ;   % copy the list
    elseif all(SZ(:,dim)==SZ(1,dim))
        % all vectors have the same length
        M = reshape(X,SZ(1,dim),[]) ;% copy the list and reshape
    else
        % We do have vectors of different lengths.
        % Pre-allocate the final output array as a column oriented array. We
        % make it one larger to accommodate the largest vector as well.
        M = zeros([maxSZ(dim)+1 nargin]) ;
        % where do the fillers begin in each column
        M(sub2ind(size(M), SZ(:,dim).'+1, 1:nargin)) = 1 ;
        % Fillers should be put in after that position as well, so applying
        % cumsum on the columns
        % Note that we remove the last row; the largest vector will fill an
        % entire column.
        M = cumsum(M(1:end-1,:),1) ; % remove last row
        % If we need to return position of the non-fillers we will get them
        % now. We cannot do it afterwards, since NaNs may be present in the
        % inputs.
        if nargout > 1
            TF = ~M ;
            % and make use of this logical array
            M(~TF) = NaN ; % put the fillers in
            M(TF)  = X ;   % put the values in
        else
            M(M==1) = NaN ; % put the fillers in
            M(M==0) = X ;   % put the values in
        end
    end
    if dim == 2
        % the inputs were row vectors, so transpose
        M = M.' ;
        TF = TF.' ; % was initialized as empty if not requested
    end
end % nargin == 1
if nargout > 1 && isempty(TF)
    % in this case, the inputs were all empty, all scalars, or all had the
    % same size.
    TF = true(size(M)) ;
end
```

Listing 9: padcat.m

Permutation Function

```
function [M, I] = permn(V, N, K)
% PERMN - permutations with repetition
%   Using two input variables V and N, M = PERMN(V,N) returns all
%   permutations of N elements taken from the vector V, with repetitions.
%   V can be any type of array (numbers, cells etc.) and M will be of the
%   same type as V.  If V is empty or N is 0, M will be empty.  M has the
%   size numel(V).^N-by-N.
%
%   When only a subset of these permutations is needed, you can call PERMN
%   with 3 input variables: M = PERMN(V,N,K) returns only the K-ths
%   permutations.  The output is the same as M = PERMN(V,N) ; M = M(K,:),
%   but it avoids memory issues that may occur when there are too many
%   combinations.  This is particulary useful when you only need a few
%   permutations at a given time. If V or K is empty, or N is zero, M will
%   be empty. M has the size numel(K)-by-N.
%
%   [M, I] = PERMN(...) also returns an index matrix I so that M = V(I).
%
%   Examples:
%     M = permn([1 2 3],2) % returns the 9-by-2 matrix:
%              1     1
%              1     2
%              1     3
%              2     1
%              2     2
%              2     3
%              3     1
%              3     2
%              3     3
%
%     M = permn([99 7],4) % returns the 16-by-4 matrix:
%              99     99    99    99
%              99     99    99     7
%              99     99     7    99
%              99     99     7     7
%              ...
%               7      7     7    99
%               7      7     7     7
%
%     M = permn({'hello!' 1:3},2) % returns the 4-by-2 cell array
%             'hello!'        'hello!'
%             'hello!'        [1x3 double]
%             [1x3 double]    'hello!'
%             [1x3 double]    [1x3 double]
%
%     V = 11:15, N = 3, K = [2 124 21 99]
%     M = permn(V, N, K) % returns the 4-by-3 matrix:
%     %        11  11  12
%     %        15  15  14
%     %        11  15  11
%     %        14  15  14
%     % which are the 2nd, 124th, 21st and 99th permutations
%     % Check with PERMN using two inputs
```

```
%     M2 = permn(V,N) ; isequal(M2(K,:),M)
%     % Note that M2 is a 125-by-3 matrix
%
%     % PERMN can be used generate a binary table, as in
%     B = permn([0 1],5)
%
%   NB Matrix sizes increases exponentially at rate (n^N)*N.
%
%   See also PERMS, NCHOOSEK
%            ALLCOMB, PERMPOS, NEXTPERM, NCHOOSE2 on the File Exchange

% tested in Matlab 2018a
% version 6.2 (jan 2019)
% (c) Jos van der Geest
% Matlab File Exchange Author ID: 10584
% email: samelinoa@gmail.com

% History
% 1.1 updated help text
% 2.0 new faster algorithm
% 3.0 (aug 2006) implemented very fast algorithm
% 3.1 (may 2007) Improved algorithm Roger Stafford pointed out that for some values, the floor
%   operation on floating points, according to the IEEE 754 standard, could return
%   erroneous values. His excellent solution was to add (1/2) to the values
%   of A.
% 3.2 (may 2007) changed help and error messages slightly
% 4.0 (may 2008) again a faster implementation, based on ALLCOMB, suggested on the
%   newsgroup comp.soft-sys.matlab on May 7th 2008 by "Helper". It was
%   pointed out that COMBN(V,N) equals ALLCOMB(V,V,V...) (V repeated N
%   times), ALLCMOB being faster. Actually version 4 is an improvement
%   over version 1 ...
% 4.1 (jan 2010) removed call to FLIPLR, using refered indexing N:-1:1
%   (is faster, suggestion of Jan Simon, jan 2010), removed REPMAT, and
%   let NDGRID handle this
% 4.2 (apr 2011) corrrectly return a column vector for N = 1 (error pointed
%    out by Wilson).
% 4.3 (apr 2013) make a reference to COMBNSUB
% 5.0 (may 2015) NAME CHANGED (COMBN -> PERMN) and updated description,
%   following comment by Stephen Obeldick that this function is misnamed
%   as it produces permutations with repetitions rather then combinations.
% 5.1 (may 2015) always calculate M via indices
% 6.0 (may 2015) merged the functionaly of permnsub (aka combnsub) and this
%   function
% 6.1 (may 2016) fixed spelling errors
% 6.2 (jan 2019) fixed some coding style warnings

narginchk(2, 3) ;

if fix(N) ~= N || N < 0 || numel(N) ~= 1
    error('permn:negativeN','Second argument should be a positive integer') ;
end
nV = numel(V) ;

```

```
if nargin==2
    %% PERMN(V,N) - return all permutations
    if nV == 0 || N == 0
        M = zeros(nV, N) ;
        I = zeros(nV, N) ;
    elseif N == 1
        % return column vectors
        M = V(:) ;
        I = (1:nV).' ;
    else
        % this is faster than the math trick used with 3 inputs below
        [Y{N:-1:1}] = ndgrid(1:nV) ;
        I = reshape(cat(N+1, Y{:}), [], N) ;
        M = V(I) ;
    end
else
    %% PERMN(V,N,K) - return a subset of all permutations
    nK = numel(K) ;
    if nV == 0 || N == 0 || nK == 0
        M = zeros(numel(K), N) ;
        I = zeros(numel(K), N) ;
    elseif nK < 1 || any(K<1) || any(K ~= fix(K))
        error('permn:InvalidIndex','Third argument should contain positive integers.') ;
    else
        V = reshape(V, 1, []) ; % v1.1 make input a row vector
        nV = numel(V) ;
        Npos = nV^N ;
        if any(K > Npos)
            warning('permn:IndexOverflow', ...
                'Values of K exceeding the total number of combinations are saturated.')
            K = min(K, Npos) ;
        end

        % The engine is based on version 3.2 with the correction
        % suggested by Roger Stafford. This approach uses a single matrix
        % multiplication.
        B = nV.^(1-N:0) ;
        I = ((K(:)-.5) * B) ; % matrix multiplication
        I = rem(floor(I), nV) + 1 ;
        M = V(I) ;
    end
end

% Algorithm using for-loops
% which can be implemented in C or VB
%
% nv = length(V) ;
% C = zeros(nv^N,N) ; % declaration
% for ii=1:N,
%     cc = 1 ;
%     for jj=1:(nv^(ii-1)),
%         for kk=1:nv,
%             for mm=1:(nv^(N-ii)),
%                 C(cc,ii) = V(kk) ;
```

```
161 %                 cc = cc + 1 ;
162 %             end
163 %         end
164 %     end
165 % end
```

Listing 10: permn.m