



Traffic optimization using a coordinated route updating mechanism

Di Mei, I-An Huang, Anita Raja, Mohammad Rashedul Hasan & Ana L.C. Bazzan

To cite this article: Di Mei, I-An Huang, Anita Raja, Mohammad Rashedul Hasan & Ana L.C. Bazzan (2022): Traffic optimization using a coordinated route updating mechanism, Journal of Intelligent Transportation Systems, DOI: [10.1080/15472450.2022.2074791](https://doi.org/10.1080/15472450.2022.2074791)

To link to this article: <https://doi.org/10.1080/15472450.2022.2074791>



Published online: 29 May 2022.



Submit your article to this journal [↗](#)





View related articles [↗](#)



View Crossmark data [↗](#)



Traffic optimization using a coordinated route updating mechanism

Di Mei^a, I-An Huang^b, Anita Raja^c , Mohammad Rashedul Hasan^d, and Ana L.C. Bazzan^e 

^aNorthwestern University, Evanston, IL, USA; ^bThe Cooper Union, New York, NY, USA; ^cHunter College, City University Of New York, New York, NY, USA; ^dUniversity of Nebraska-Lincoln, Lincoln, NE, USA; ^eUniversidade Federal do Rio Grande do Sul (UFRGS), P. Alegre, RS, Brazil

ABSTRACT

Traffic congestion is ubiquitous in cities across the globe resulting in great economic and environmental costs. Although real-time traffic updates are now available, the tendency of drivers to make uncoordinated routing decisions exacerbates the known problems of selfish routing including traffic congestion and flow oscillation. Existing solutions, in both private and public domains, do not necessarily provide efficient mechanisms for creating a socially optimal traffic distribution (i.e., the one that minimizes the total travel time, rather than those that are individualistic and uncoordinated) to overcome the congestion problem. In this article, we present a decentralized multi-agent systems-based framework that harnesses a coordinated route recommendation algorithm while measuring the influence of coordinated decision making to improve the efficiency of the entire vehicular network. We study how our approach affects performance in synthetic traffic networks and abstractions of real-world networks. Extensive simulation results show that our approach is able to establish near socially optimal traffic distribution in networks with varying scales and price of anarchy values. They also reveal that network complexity not only accounts for network size and demand but also how this demand is distributed. Our approach produces a near-optimal traffic distribution even when up to 30% of all vehicles are not coordinated, regardless of network type. We show that in non-trivial networks, the ability of a subset of vehicles to coordinate, improves the total travel time of all the vehicles on the network while alleviating congestion and oscillation.

ARTICLE HISTORY

Received 22 January 2021
Revised 27 April 2022
Accepted 4 May 2022

KEYWORDS

Traffic congestion; selfish routing; multi-agent systems; connected vehicles; coordinated routing; distributed traffic coordination algorithm

Introduction

The traffic assignment problem (TAP), which deals with the computation of the vehicle flow on each traffic route, has been explored extensively for many years. One issue with the TAP is that drivers¹ in a traffic network are usually selfish, seeking to minimize their own travel time by choosing the quickest route to destinations without considering the outcomes of their choices, thus leading to traffic congestion. The effect of selfish routing on traffic congestion is explained by two static equilibrium states in a traffic network formulated by Wardrop (1952): the user equilibrium (UE) and the system optimum (SO). These two concepts are closely associated with the drivers' travel time on a route, which is determined by the total number of drivers per time unit (flow) using that route. The UE is a state in which every driver is selfish and the time for each driver traveling along any route between a source and destination is the same. The SO is a state in which the total travel

time is minimized. Roughgarden and Tardos (2002) view the selfish drivers of such a network that lacks coordination as independent agents in a non-cooperative game. They argue that this type of selfish routing leads to route choices that form the Nash Equilibrium (NE). The NE does not necessarily optimize the social welfare or SO.

Figure 1 illustrates the difference between the SO and UE. It contains one origin-destination (OD) pair with two possible routes. While the SO is considered the ideal traffic distribution, real-world traffic networks tend to converge to the UE because selfish drivers tend to select the route with the minimum travel time (Li et al., 2017; Wang et al., 2015). Assume that the network has a flow of 8 vehicles. Vehicle travel times are computed by latency functions $l(f)$, where f is the vehicle flow on an edge. The latency function of the upper route is $l(f) = f$, and that of the lower route is a constant $l(f) = 8$. If all drivers are selfish and there is no coordination between them, then they will

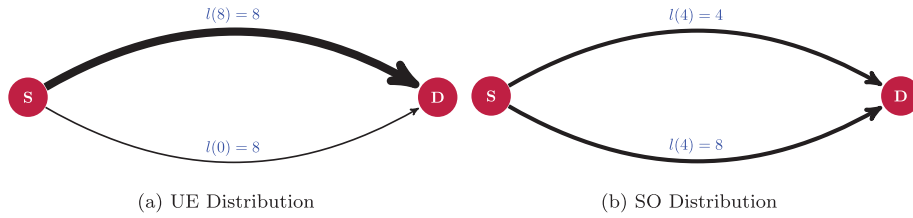


Figure 1. A Simple Network. (a) UE Distribution and (b) SO Distribution

always choose the route with the lowest travel time. Each vehicle will keep selecting the upper route until the individual travel time of the upper route reaches 8. No driver has the incentive to change its route because the expected travel time for both routes is the same. In Figure 1a, the UE distribution is reached and the total travel time is $8 * 8 + 8 * 0 = 64$. If the drivers were to coordinate, then both routes would be used evenly. In Figure 1b, each route has four vehicles and the total travel time is $4 * 4 + 4 * 8 = 48$ which happens to be the lowest possible value of total travel time in this traffic network. Thus, the SO is reached. Drivers often believe that the UE distribution is an efficient state, but in fact, it not only increases the vehicle's total travel time but also easily leads to traffic congestion (at any point in time, most vehicles are on the same route) (Roughgarden & Tardos, 2002).

To measure the influence of coordinated decision-making, we use the concept of price of anarchy (PoA) (Koutsoupias & Papadimitriou, 1999). The PoA for a traffic network is defined as the ratio of average travel time under the UE and SO distributions and is usually dependent on the type of latency function and the scale of a traffic network. To reduce the PoA and achieve the socially optimal distribution in a traffic network, we propose that an effective mechanism for coordinating vehicle route choices is required.

Popular traffic navigation mobile applications (apps) such as Google Maps or Waze ("Google Map", n.d.; "Waze", n.d.) that are designed to help drivers to choose the fastest routes do not fare well with combating the *selfish routing* problem. Their (apparently) greedy route selection based strategies send vehicles to the fastest routes, thus increasing congestion and oscillations in traffic distribution (R. Liu et al., 2016; Buscema et al., 2009). However, since the route selection algorithms used by these navigation apps are not public, it is not possible to investigate ways to mitigate the congestion problem.

In this work, we model a typical traffic network with multiple origin-destination pairs as a multiagent system (MAS) consisting of a set of independent vehicles connected by a GPS-enabled mobile app. The vehicles are connected to an app-specific server that

has access to each connected vehicle's location at any point in time. The server uses the location information of the vehicles to execute a centralized route recommendation algorithm. Each driver that uses the app can choose to use the route recommended by the centralized authority or not. The MAS also consists of drivers that make selfish and uncoordinated decisions to determine their routes. The proposed framework is (a) a *multiagent system* due to the autonomous and heterogeneous vehicles (coordinated and selfish agents) participating in it; (b) *coordinated*, since the subset of agents that use the app are connected to a central server. The central server takes information from multiple agents, computes policies based on this joint information and imparts them to the connected subset of app-enabled vehicles in the traffic network, and (c) *decentralized*, since the route choice decisions are distributed among the connected subset of app-enabled vehicles (which have the choice to follow or reject the recommended route) and those not connected to the app (which can choose from a variety of self-interested choices). Using this MAS framework, we highlight the advantage of harnessing a connected subset of vehicles that is capable of coordination and favors the socially optimal distribution to improve the efficiency of the entire vehicular network.

To our best knowledge, there is a gap in the literature with regard to minimizing the PoA of a traffic network; in other words aligning the UE to the SO of the network. Traditionally, this is investigated by focusing on tolling, and the similar road pricing mechanisms discussed in the section "Background and related work." Even so, there has been a surge in studies investigating the use of the driver's communication networks as well as online social networks to coordinate drivers (T. Liu et al., 2017; Li et al., 2017; New Cities Foundation, 2012; Pathania & Karlapalem, 2015). However, these works have the following limitations: some deal with simple traffic networks (usually with two routes only); others (T. Liu et al., 2017; Li et al., 2017) use resource-intensive centralized techniques; while the social network based approach in Pathania and Karlapalem (2015) is limited to train network systems where the route choices are more

constrained than in road networks. Against this background, we consider more realistic scenarios, as well as investigate how communication and location technologies can be put into service of the aforementioned alignment between UE and SO. We also note that it is not trivial to access the necessary data; while network topology can be easily exported from platforms such as OpenStreetMap (“Open Street Map”, n.d.), the demand data (origin–destination matrices) are seldom available for such maps.

The main contributions of this article are as follows:

- We present an MAS coordination framework that leverages the drivers’ real-time information to help establish the optimal traffic distribution and overcome the inefficiency of the UE.
- We provide a comprehensive discussion of the related work in this area and discuss why our work is novel.
- We show that our framework is able to produce a near-SO distribution even when not all drivers are coordinated.
- We show that our framework creates more stable traffic distribution as the flow of coordinated drivers’ increases.
- We explore the effect of different types of latency functions on our framework resulting in varying traffic performance characteristics.

The remainder of this article is organized as following. First, we discuss relevant literature in the section “Background and related work.” Then, we present our proposed approach in the section “Approach” followed by an extensive empirical evaluation in the section “Experiments.” Finally we conclude with a summary of our observations and discussion of future work in the section “Conclusion and future work.”

Background and related work

This article addresses the demand perspective of transportation systems. Specifically, it deals with the assignment of the demand to the infrastructure (the traffic network), i.e., we deal with the TAP. There is an extensive literature (Ortúzar & Willumsen, 2011) that deals with assignment of demand. The concepts of route guidance, ATIS (advanced travellers’ information systems) and road pricing (which we discuss ahead) are related to the assignment problem, and also to issues regarding how to inform and divert the users of the transportation system. Multiagent systems

have been used to deal with both ITS and ATIS as described in (Bazzan & Klügl, 2013a, 2013b; Chen & Cheng, 2010; Jin, Hui, & Ping, 2003).

In transportation network research, the problem of *selfish routing* among a group of non-cooperative drivers is modeled via congestion games (Lim & Rus, 2012). In a congestion game, the cost for each route is determined by the flow of all drivers using that route. Each driver chooses a route to minimize their travel cost (e.g., travel time), which may result in congestion. This game has at least one pure strategy Nash equilibrium in which none of the drivers can reduce her travel cost by unilaterally changing a route. As discussed in section “Introduction”, the Nash equilibrium is known as user equilibrium (UE), while a socially optimal distribution of traffic is represented by the so-called system optimum (SO) (Wardrop, 1952). At the SO, traffic should be arranged in congested networks in such a way that the total travel cost is minimized. It has been shown the Nash equilibrium or UE in congestion games could be inefficient (resulting in larger total travel cost) and a socially optimal (SO) distribution is desirable (Li et al., 2017).

Most of the aforementioned works that use multiagent techniques aim to investigate how agents reach the UE. The quest for an approach, including the use of MAS, to align the UE and the SO is still an open research question. There is prior research on using a framework called collective intelligence (COIN) (Agogino & Tumer, 2008; Tumer, Welch, & Agogino, 2008) which show that, during reinforcement learning, multiagent coordination can be used to reduce congestion in road traffic as well as in air traffic, respectively. While the former has a similar goal as ours, their approach is based on selecting departure time, not routes. Also, Klein et al. (2018) considered a traffic information system that can persuade drivers to cooperate. However, they use a simple scenario with just two routes.

With respect to congestion games, the price of anarchy (PoA) (Koutsoupias & Papadimitriou, 1999) is a metric used in this article to measure how the efficiency of a system degrades due to selfish behavior of its agents. It has been shown that for any generalized routing problem with linear latencies, the PoA is at most $4/3$ (Roughgarden & Tardos, 2002), and the PoA of a small-scale network is generally higher than that of a large-scale network (Youn et al., 2008).

Several mechanisms based on road pricing (Fleischer et al., 2004; Xiao et al., 2013) or learning (Bazzan, 2019) have been proposed to improve the UE in traffic networks. A more realistic version uses a

scaled marginal-cost road pricing technique (Wang et al., 2015) to improve the PoA. While the algorithm guarantees that the PoA will reach one for two group players, it fails to do so in more general cases, which limits the applicability of this work. Moreover tolls and road pricing mechanisms have societal costs and associated objections that make them less appealing.

In Hasan et al. (2016), the Traffic Route Preference Function (TRPF) is introduced as a measure that indicates whether the number of drivers on a route is higher than the theoretical optimal. This measure is used to help direct the traffic network to converge to the appropriate SO distribution. The TRPF approach assumes that human drivers can estimate the difference between the number of current drivers on a particular route and the optimal number of drivers for that route and the TRPF value is updated after one round of cars pass through the network. These assumptions are unrealistic for real traffic environments because human drivers are incapable of exactly knowing the optimal number of drivers on a route. Moreover, updates after an entire round can cause large oscillation of the number of drivers on a given route when the proportion of drivers using the TRPF is high, because in each round, TRPF users choose the same route with the highest TRPF value. The framework described in this article seeks to address these general problems.

It can be seen that these works address a variety of issues related to alleviating traffic congestion and/or use different methods. Some works do address alignment of UE to the SO. However, they deal with different problems: air traffic control (Tumer et al., 2008), departure time (Agogino & Tumer, 2008), communication bandwidth (T. Liu et al., 2017), flexible mobility (Bucchiarone, 2019), re-routing to bypass congestion (Falek et al., 2022; Wang et al., 2014). Some use classical mechanisms to reach such alignment, such as toll (Fleischer et al., 2004) or other road pricing schemes (T. Liu et al., 2017). With respect to centralization, works stemming from the optimization community tend to assume a central entity in charge of giving some incentive for drivers to align with the SO (Fleischer et al., 2004; Li et al., 2017; Lim & Rus, 2012; T. Liu et al., 2017; Zhang & Nie, 2018). Others can be considered partially centralized as not all routes are determined or imposed by the central entity. For instance, agents may learn autonomously in a decentralized fashion (Agogino & Tumer, 2008; Tumer et al., 2008; Xiao et al., 2013); choose not to follow the recommended route in the present paper or while using the TRPF approach

(Hasan et al., 2016); or use persuasion mechanisms, but with just two autonomous agents (Klein et al., 2018). The partial decentralization in our multiagent approach facilitates agent autonomy. Specifically, we use two probabilistic parameters as metaphors for autonomy.

As is common in the literature related to the PoA (e.g., Roughgarden and Tardos (2002) and works that followed it), we assume that the flow of vehicles per OD pair remains nearly constant in a given time frame. This time frame could be the morning or afternoon peaks, or the inter peaks scenario. This is commonly known as a static traffic assignment problem (STAP). Hence, the constant demand (flow) of the entire traffic network ensures that the SO distribution which is a known parameter in the CRUM model remains unchanged. We note that constant demand does not mean that the flow per edge remains constant; this is because each driver has several options to travel from its origin to its destination making the flow per edge dynamic. While we consider dynamically changing flows for the entire network outside the scope of this work, stacking of time windows where the demand remains constant to handle dynamic situations is one possible solution. In recent work, (Shynkar et al., 2022), we investigate the role of meta-level control mechanisms capable of reasoning about dynamic traffic situations in a non-myopic fashion.

In short, while there is a plethora of works addressing TAP and congestion management, this article presents a novel approach where a subset of vehicles on the road network coordinate to improve the total travel time of all the vehicles on the network while alleviating congestion.

Approach

We address the traffic congestion problem using the MAS experimental framework described in Figure 2. The framework consists of vehicles in the connected subset of app-enabled vehicles (red) and selfish (yellow) vehicles. As described in section “Introduction”, the vehicles in the connected network use a GPS-enabled mobile application to connect to a route recommendation server. The server uses the location of the vehicles that are connected to it via the app along with an estimate of the location of the others (see discussion ahead), as input to the route-computation algorithm, we have developed. We call it the *Coordinated Route Updating Mechanism (CRUM)*. Vehicles in the connected subset are thus coordinated; henceforth, we call the connected subset of app-

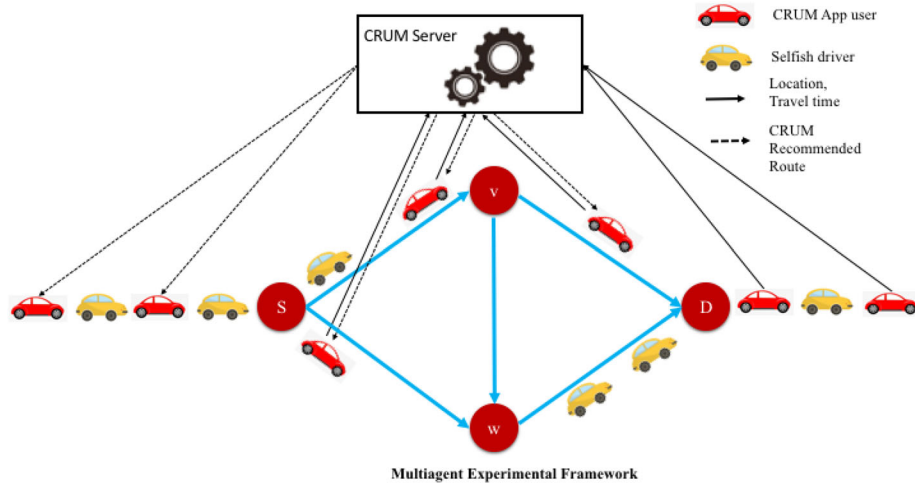


Figure 2. MAS Experimental Framework with centralized CRUM server for vehicles on connected subset of app-enabled vehicles (red vehicles) and selfish routing by selfish vehicles (yellow vehicles).

enabled vehicles the CRUM network; the server that manages the app is called the CRUM server; and the vehicles in this network are called CRUM vehicles. A CRUM vehicle upon entering the road network will have access to the CRUM server's most recent route recommendation at the beginning of its journey and can choose to use the recommended route or ignore it.

Selfish vehicles, on the other hand, are not part of the CRUM network and seek to minimize their own travel time by choosing the quickest route to destinations without considering the (potentially negative) outcomes of their choices on the average travel time of the entire network.

This MAS experimental framework mentioned above is hosted on a server called the MAS experimental server which is used for evaluation purposes only. The MAS experimental server has access to the location of every vehicle on each of these routes (and hence current flow) on each route. The CRUM server requires location information about CRUM vehicles and other information that we elaborate below. We now provide a formal description of the problem.

Problem model

The traffic network is represented by a directed network $\mathcal{G} = (V, E)$, as defined by Roughgarden (Roughgarden & Tardos, 2002), where V is the vertex set (locations), E the edge set (roads) and origin-destination (OD) pairs $\{o_1, d_1\} \dots \{o_k, d_k\}$. Each OD pair can contain multiple routes denoted by the set $R_i = \{r_i\}$ and $R = \cup_i R_i$. A route $r_i \in R_i$ is defined as the set of edges in a simple path that travels from the origin o_i to the destination d_i . The networks considered

in this article (see ahead) have one or more OD pairs. A flow is a function $f: R \rightarrow \mathbb{R}^+$, and the flow of a route is defined as $f_r = f(r)$. For a fixed flow f , the flow of an edge is defined as $f_e = \sum_{r: e \in r} f_r$. Each edge $e \in E$ has an associated load-dependent latency $l_e(\cdot)$. We assume the latency functions to be non-negative, differentiable and non-decreasing.

The latency function $l_e(f_e)$ calculates the travel time for a driver on an edge. It is dependent on the load or flow f_e and can be linear or nonlinear and is measured in some unit of time, which depends on the units of t_e and other constants and variables. For example, a linear latency function is given in Equation (1), where t_e is the free-flow travel time of edge e , d_e is a multiplicative constant, and f_e is the flow on e .

$$l_e(f_e) = t_e + d_e * f_e \quad (1)$$

The Bureau of Public Roads (BPR) function is widely used as a nonlinear latency function. It depends on free-flow travel time and vehicle flow capacity y_e of an edge e as shown in Equation (2), where l_e is the travel time at edge e given the edge traffic flow f_e , t_e is the free flow travel time on edge e per unit time, y_e the edge capacity, and a_e and b_e are parameters for calibration.

$$c_e = l_e(f_e) = t_e * \left(1 + a_e \left(\frac{f_e}{y_e} \right)^{b_e} \right) \quad (2)$$

The free flow times for the linear and BPR latency functions have similar implications for the physical characteristics of a road, and they often share the same value when referring to the same edge in a graph. The latency of a route r with respect to the flow f is the sum of the latencies of the edges in the route, denoted by $l_r(f) = \sum_{e \in r} l_e(f_e)$.

Our model is viewed as an instance of the TAP (see Section “ ”). The *theoretical* UE flow along a route is found by solving a convex optimization using the method of successive averages (MSA) (Sbayti et al., 2007). The theoretical SO can be found using IBM’s CPLEX for linear latency functions and CVXOPT² that handles nonlinear convex optimization such as the BPR latency function.

The core challenge we address in our model is the design of an effective coordination mechanism that enables traffic distribution to converge to the SO and eliminate congestion caused by the UE. We also study the effect of the linear and BPR latency functions on our model. Using the linear latency function in a traffic simulation framework usually comes with higher efficiency due to its simple form, but the BPR latency function provides a more realistic model of a traffic simulation because of its non-linearity and higher sensitiveness to vehicle flow. By utilizing both functions in our experimental framework, we are able to explore the effect of different latency functions with various parameter settings on traffic performance.

We introduce two important vehicle-related parameters in our model: G is the probability of a vehicle changing its route decision (instead of using its most recent route) in every round and p is the fraction of CRUM vehicles. The parameter G exerts influence on the stability of vehicle flow distribution, and p explicitly reflects the number of coordinated vehicles in a distributed traffic system. The manipulation of these parameters allows us to evaluate the effectiveness of our approach from a comprehensive perspective.

We make the following assumptions in our model:

- Only a fraction of vehicles in the MAS simulation framework are connected by the GPS-enabled mobile application, which offers vehicles route recommendations while not expecting drivers to interactively use the app to provide real-time updates about their travel experience.
- The centralized algorithm for the CRUM network is modeled as an instance of STAP in a transportation network where the total demand of the network with selfish drivers and coordinated drivers remains constant.
- Each selfish vehicle is viewed as a selfish agent; after performing experimentation (exploration), these drivers choose routes with the minimum flow (travel time).
- There exists a set of routes that are used on a regular basis by selfish vehicles, for instance the commute route from the suburbs to center city; this enables

these vehicles to accumulate traffic experience which assists them to make greedy decisions of routes.

Algorithms

In the CRUM network, each vehicle is modeled as an agent. A centralized server continuously collects information from the agents to compute a route recommendation policy at each state. However, each CRUM agent makes an independent decision on whether to use the CRUM-recommended route or the most recently used route, in other words, the decision whether to use the CRUM-recommended route is a distributed decision among the CRUM vehicles.

Algorithm 1 is the pseudocode description of our multi-agent traffic simulation framework. The input consists of the selected network’s topology, information about traffic network’s demand (consisting of both CRUM and selfish vehicles) and the number of simulation rounds. Each simulation round involves execution of Lines 1.1 – 1.9 and each vehicle in the simulation executes Lines 1.2 – 1.8. Each vehicle first makes a route choice in Line 1.3 before entering the traffic network. We present details of the route choice function in Algorithm 2. After choosing the route, the vehicle travels through that route and the corresponding travel time is recorded in Line 1.4. For CRUM vehicles, the travel time is recorded by the mobile app, which is then used to update CRUM server’s computation of the recommended route in Lines 1.5 – 1.7. If a CRUM vehicle leaves the traffic network at time t , then CRUM’s recommended route is simultaneously updated at time t . This new route recommendation is only available to the next CRUM driver entering the network at timestep $t + 1$ or later.

Algorithm 1: TrafficSimulation(V, NG, n)

input: V : Set of vehicles from the CRUM and selfish subnetworks NG : Network graph consisting of vertices, edges, and routes n : number of rounds

output: route_flows_avg, route_flows_std, route_time_avg

```

for  $i \leftarrow 1, n$  do
  for  $v \in V$  do
    chooseRoute( $v$ )
    updateTravelTime( $v$ )
    if CRUMVehicle( $v$ ) then
      updateCRUMServer( $v$ )
    end
  end
end
return (route_flows_avg, route_flows_std, route_time_avg)

```

Algorithm 2: chooseRoute(*v*)

input: *v*: vehicle
output: *k*: route selected by the driver
 initialize *a*, *b* \leftarrow random value between 0 and 1
 G = probability of driver *d* changing route
 δ_r = set of difference values
 R_v = set of routes available to the vehicle
 $last_{choice}_v$ = last route selected by the vehicle
 $historical_{tim}_v$ = historical travel time of vehicle
 $travel_{count}_v$ = number of times that the vehicle traveled through each route
if *a* < *G* **then**
 k \leftarrow *last_choice*
else if CRUMVehicle(*v*) **then**
 k \leftarrow chooseCRUMRoute(R_v, δ_r);
else if *b* < ϵ **then**
 k \leftarrow chooseRandomRoute(R_v);
else
 k \leftarrow chooseRouteByExperience($R_v, historical_{tim}_v, travel_{count}_v$);
end
 return(*k*)

For selfish vehicles, the travel time in Line 1.4 is recorded as part of their traffic experience. As discussed earlier, selfish agents in our model use their individual prior traffic experience to determine the route choice with minimum travel time. As the number of rounds in Algorithm 1 increases, it is expected that each selfish driver gradually becomes familiar with the traffic network and knows which route has the lowest traffic cost for them. The simulation server records all the routes selected by both CRUM and selfish vehicles in the previous rounds and the corresponding travel times for evaluation purposes.

The CRUM server has knowledge of both the number of CRUM vehicles and the travel times of each CRUM vehicle using each of the routes (Line 1.4). It also has access to the theoretical SO, which we call DN_r^* for each route *r* since the latency function is known. Finally the CRUM server requires flow estimates of other (selfish) vehicles in the network. Such estimates can be obtained in a variety of ways: (i) extrapolation from the data on the CRUM vehicles; (ii) historical data; and (iii) data acquisition from external providers. As we mention in the discussion about future work, the extrapolation from the data on CRUM vehicles can be realized by using machine learning techniques. In this work, we assume that selfish drivers, having explored routes, have settled on routes that have proven to be efficient for them. In

other words, the selfish drivers are assumed to have already converged in terms of route choices.

Using the location information of CRUM vehicles and flow estimates of non-CRUM vehicles, the CRUM server is thus able to compute DN_r , which is the estimate of the total number of vehicles on each route *r*. We introduce the delta vector δ_r which represents the numerical difference between the estimated current traffic distribution and the theoretical SO distribution:

$$\delta_r = DN_r - DN_r^* \quad (3)$$

The CRUM recommended route is geared to redirect traffic to the route which has the highest margin to take on new vehicles until the socially optimal allocation for that route is met. In other words, it will be the route associated with the lowest flow difference between the current estimated distribution and the SO (selected route denoted by *k*):

$$k \leftarrow \underset{r_i \in R_i}{\operatorname{argmin}} \{ \delta_{r_i} \} \quad (4)$$

Selfish drivers, on the other hand, make route decisions based on their experience when possible, which is reflected by average costs of their traveled routes.

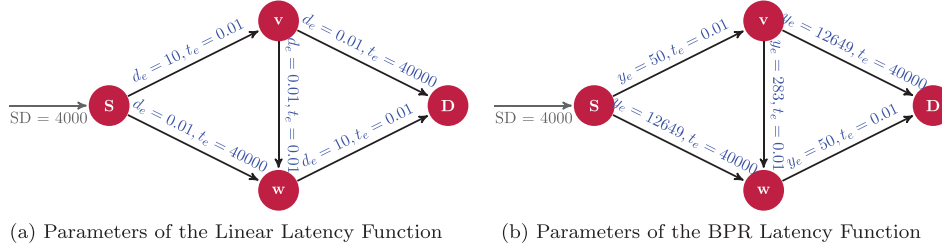
Algorithm 2 is the pseudocode of the chooseRoute(*d*) in Algorithm 1 Line 1.3. It determines the route choices of both CRUM and selfish vehicles. We introduce two parameters, *a* and *b*, represented as random floats between 0 and 1 that are initialized in Line 2.1. These parameters are vectors corresponding to each vehicle. They are used to determine the decision choices of the vehicles including whether to use a prior route or the CRUM-recommended route or a random route.

Lines 2.8 through 2.17 show how different types of drivers can choose different routes. If *a* is smaller than *G*, the vehicle chooses the same route it used in the last round. Else, if *a* is greater than *G* and the vehicle is a CRUM vehicle, the vehicle will use the CRUM server's recommended route updated by all the preceding CRUM vehicles.

If *a* is greater than *G*, then the vehicle is a selfish vehicle and if the parameter *b* is also smaller than 0.05 (Lines 2.12 – 2.13), the vehicle will randomly select a route. The threshold of 0.05 for *b* represents the small fraction of drivers that can select a route randomly. This encourages vehicles to explore new routes, which will allow them to have a more diversified experience and choose routes that truly minimize their travel time. Such exploratory behavior allows selfish vehicles to explore routes with the minimum travel time.

Table 1. Experimental network characteristics summary.

	No. of OD pairs	No. of drivers	Max. No. of routes	linear PoA	BPR PoA
Braess	1	4000	3	1.33	1.94
ND Network	4	2000	25	1.01	1.06
Sioux Falls	528	360600	inf	1.00	1.04

**Figure 3.** Braess' Network. (a) Parameters of the Linear Latency Function. (b) Parameters of the BPR Latency Function

Selfish vehicles with a greater than G and b greater than 0.05 (Lines 2.14 – 2.15) will have their expected route determined by the average travel time of selfish vehicles for each route calculated using the traffic data from the previous rounds. We reiterate that the expected route computation for selfish vehicles is independent of the CRUM server.

The conditional statements described above are summarized as follows:

- if $a < G$, all vehicles, selfish or CRUM, then use their individual previous route.
- if $a > G$, and a CRUM vehicle, then use CRUM recommendation.
- if $a > G$, and a selfish vehicle, then use random choice if $b < \epsilon$; use the experience information if $b > \epsilon$.

Experiments

Networks

We study the performance of the CRUM-based traffic optimization model on three networks of increasing complexity. Our use of these networks is due to the fact that they have both topology and demand (full OD matrix plus latency functions) published in the literature (Braess, 1968; Nguyen & Dupuis, 1984; “Transportation problems hub”, n.d.). Moreover, they are frequently used as benchmarks. We provide a summary of the characteristics of the three networks in Table 1.

Besides these benchmarks, we also discuss results on modifications of these networks, such as what happens when different latency functions are used.

Braess network

The first network (depicted in Figure 3) is inspired by the Braess paradox (Braess, 1968), where there is an incentive for drivers to greedily choose a route with the lower cost even though it leads to an increase in average travel time. Braess is a synthetic network with a relatively simple structure. It consists of one OD pair and five edges that are associated with either a linear latency function (Figure 3(a)) or a BPR latency function (Figure 3(b)).

Nguyen Dupuis (ND) network

The second network is a medium-scale network (Nguyen & Dupuis, 1984), with four OD pairs that are combinations of two origins (nodes 1 and 4) and two destinations (nodes 2 and 3) as shown in Figure 4. By enumerating all possible routes for these OD pairs, we find 25 routes in total. The overall demand of this network is 2000 drivers. The PoA is around 1.01 with a linear latency function and 1.06 with a BPR latency function. Compared to the Braess' network, the ND network has a lower PoA.

Sioux falls network

The third network is an abstraction of the central portion of the city of Sioux Falls, South Dakota, depicted in Figure 5. Unlike the first two networks, the roads in this network are represented by bi-directional edges going in the opposite directions. There are 24 nodes and every distinct node-to-node pair is considered an OD pair with positive demand. A total of 360600 drivers are assumed to travel through this network and the PoA is even smaller than that of the ND network given the very large scale and complex network structure. The maximum number of routes is infinite due to the existence of loops. We use standardized network data³ for our evaluation. The parameters for the BPR latency function (Equation (2)) were set to

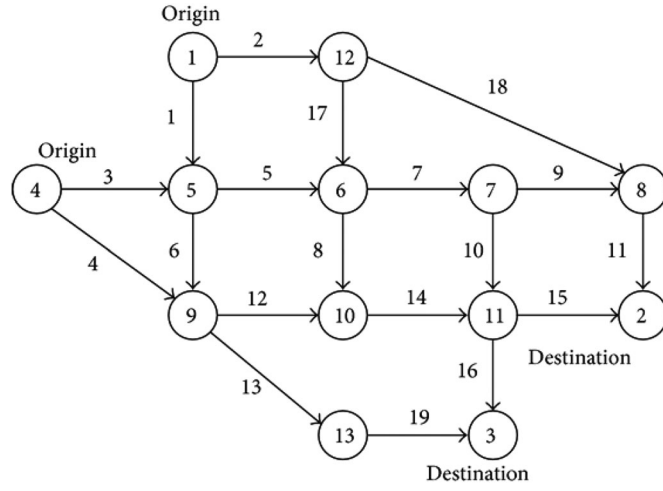


Figure 4 Nguyen Dupuis (ND) Network.

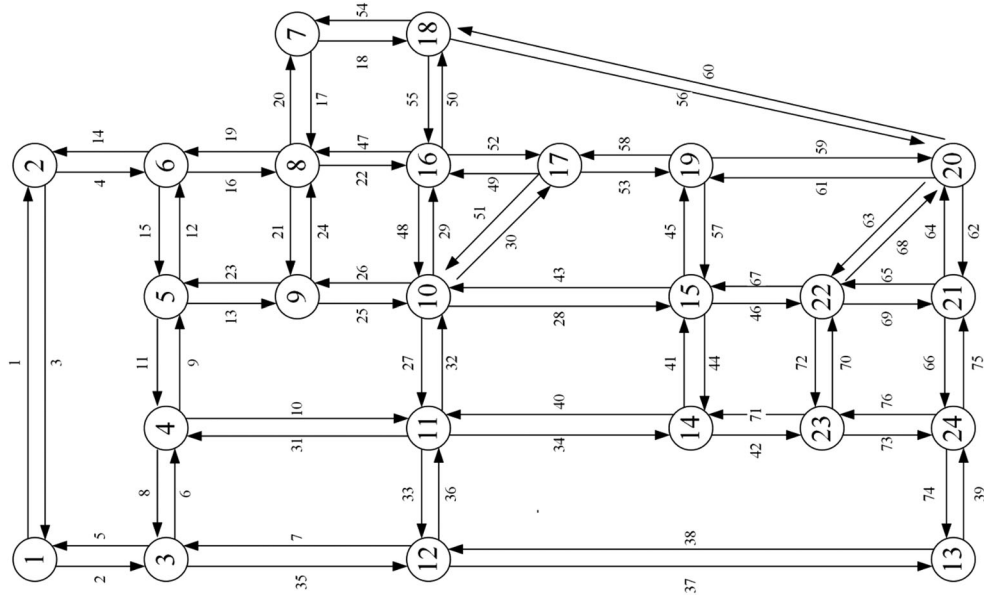


Figure 5. Sioux Falls Network.

$a_e = 0.15$ and $b_e = 4$ for the latency function as is common in the literature (e.g., (Petrik et al., 2014)). It is based on expert judgment for a medium-sized city.

In summary, the Braess, ND, and Sioux Falls networks are chosen primarily because with their varying number of routes, OD pairs and latency functions, they are representative of a broad range of traffic networks for the STAP model.

Simulation setting

We conduct simulations on the above three networks with the following goals:

- *Goal 1:* Show that the CRUM-based MAS framework overcomes the inefficiency caused by selfish

drivers in networks of different scales by creating a near SO distribution of flows.

- *Goal 2:* Show that the CRUM-based MAS framework is able to generate a more stable traffic distribution than the traffic model that does not use any explicit coordination.
- *Goal 3:* Study the robustness of the CRUM-based MAS framework with respect to efficiency and stability under various parameter (G and p) settings.
- *Goal 4:* Study the robustness of the CRUM-based MAS framework with respect to performance when different latency functions are applied.

We vary the two model parameters $G = [0.4, 0.6]$ (probability of a vehicle changing its route in each round), $p = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]$

(fraction of drivers that use CRUM to choose routes) to observe the effect of CRUM-based algorithms on traffic distribution (average number of drivers in each route), the drivers' average travel time, and the standard deviation of flow for each route. The range for G ⁴ was chosen to study the effect of when most of the drivers change routes versus only a few of the total number of drivers change routes in each round. The value of ϵ in Algorithm 2 is set to 0.05.

Each experiment reports the results of 200 rounds. A round represents a single trip of a vehicle from its origin to destination during which the vehicle chooses a route exactly once. The results are generated by taking the average of the last 50 rounds once the traffic distribution stabilizes.

Experimental results

The experiments below investigate the effectiveness of CRUM-based MAS framework on the three networks: Braess (Figure 3), ND (Figure 4), and the Sioux Falls network (Figure 5), with a constant flow of 4000, 2000, and 360,600 drivers, respectively. Each network when combined with the linear latency function is called the *linear* model of the network and when combined with the BPR latency function is called the *non-linear* model. The experiment results include the average travel time and flow oscillation, presented in Figures 6 and 7 respectively. For each plot in Figure 6, the dashed line represents the drivers' average travel time at the theoretical UE (upper bound) and the dotted line represents the theoretical SO (lower bound). We use the theoretical UE and SO as baselines since comparison of the CRUM-based MAS algorithms to WAZE or similar apps is not practically possible (the route selection algorithms used in those apps are not available to the public). Table 2 describes the theoretical SO and UE average travel time computed using MSA (Sbayti et al., 2007) and optimizers for the linear and nonlinear models, respectively, with respect to the three traffic networks.

Comparison by latency functions

We evaluate our model using both linear and BPR latency functions. In all plots of Figure 6, as p increases (i.e., the number of CRUM vehicles increases), the average travel time decreases in all traffic networks. With the linear models, the average travel time decreases in almost a linear fashion, but this trend is not observed with the nonlinear models. Compared to results obtained from the linear model,

the average travel time is generally closer to the socially optimal travel time in the nonlinear model.

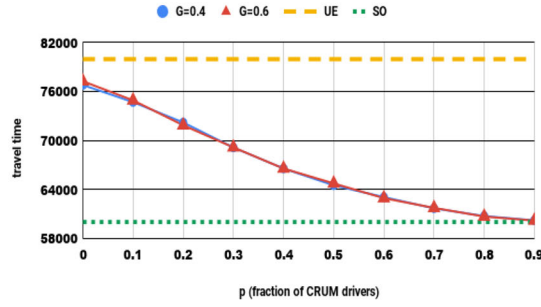
In Figure 6b, the traffic distribution for the Braess network reaches the near-SO distribution when 60% of the total vehicles are CRUM vehicles ($p = 0.6$), while with the linear model of the Braess network, Figure 6a shows that the average travel time gets close to the SO when 80% are CRUM vehicles ($p = 0.8$). Similarly, for both ND and Sioux Falls' networks in Figure 6c–f, traffic distributions reach the SO at a lower p value in the nonlinear model than those in a linear model. As indicated by the plots in Figure 6, in general, while the traffic distribution when using the linear model does not converge as fast as compared to using the nonlinear model, it can still be close to the SO distribution at $p > 0.7$. We have shown that our proposed framework is more efficient for networks using the BPR model which as mentioned earlier, is the latency function that is more representative of real traffic. We have also shown that the framework results in improved performance for networks in which UE and SO varies significantly (i.e., Braess and ND) as well as networks in which UE and SO are closer (i.e., Sioux Falls in 6(e)). Thus, Goal 1 which states that the CRUM-based MAS framework overcomes the inefficiency caused by selfish vehicles in networks of different scales by creating a near-SO distribution of flows is achieved.

Comparison while varying G and p :

All plots in Figure 6 indicate that the variation of G does not necessarily influence the vehicle's average travel time. Algorithm 2 ensures that while G does not affect the convergence point of the traffic distribution, it influences the vehicle's route choices and alters the rate of convergence. When p is equal to 0, nearly all the vehicles are selfish and the average travel time of Braess' and ND networks is very close to the average travel time at UE. When p is high (e.g. $p = 0.9$), almost all the vehicles are coordinated by the CRUM server and the average travel time of Braess' and ND networks also becomes close to the average travel time at SO. However, this phenomenon is not observed in the Sioux Falls network. While the average travel time decreases smoothly as p increases in the Sioux Falls network, the individual travel times are actually higher than the theoretical UE for most values of p in the case of both latency functions. We argue that these results are caused by the complexity of a network. Among the three networks used in our simulation, the Sioux Falls network has (a) the most complicated topology of the three representative networks studied,

Average Travel Time

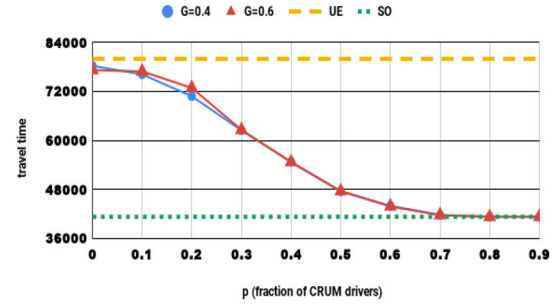
Braess' Network (linear latency function)



(a)

Average Travel Time

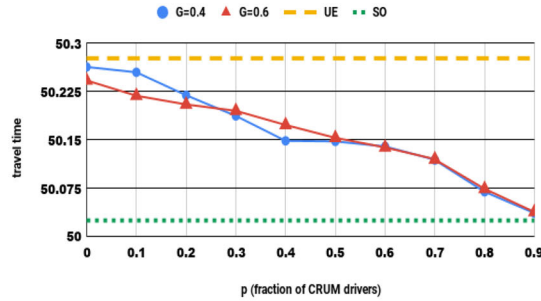
Braess' Network (BPR latency function)



(b)

Average Travel Time

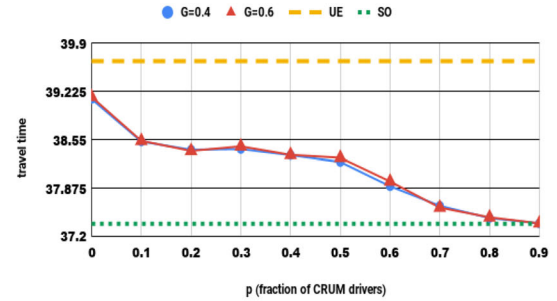
ND Network (linear latency function)



(c)

Average Travel Time

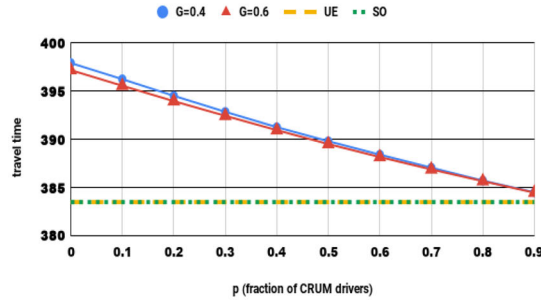
ND Network (BPR latency function)



(d)

Average Travel Time

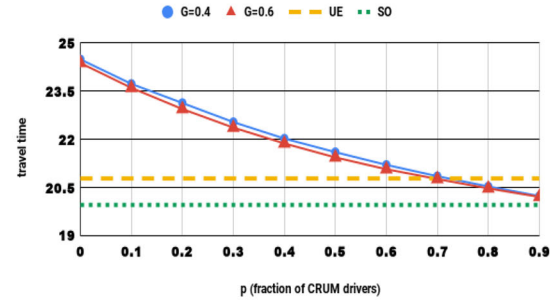
Sioux Falls (linear latency function)



(e)

Average Travel Time

Sioux Falls (BPR latency function)



(f)

Figure 6. Average Travel Time with $G = [0.4, 0.6]$ and p varying from 0.1 to 0.9; Figs. (a), (c) and (e) use linear latency function while Figs. (b), (d) and (f) use BPR latency function for Braess, ND and Sioux Falls networks respectively.

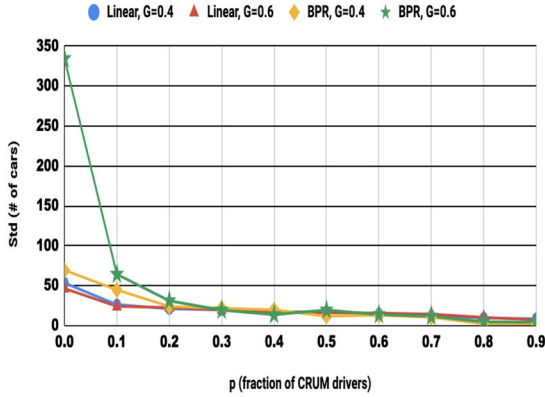
(b) a low PoA and also (c) the highest number of vehicles. Finding the UE distribution involves nonlinear programming, or convex optimization. In a large scale network, any unexpected behavior of vehicles (even it is a small-extent data oscillation) will cause traffic distribution to deviate from the convergence to UE and leads to average travel time that is higher than the theoretical UE travel time.

Our observations seem to indicate that network complexity not only takes into account network size and demand but also how this demand is distributed, number of OD pairs and so on. In Table 1, we show

the PoA (indicating the distance between the Nash-based UE and the SO) of the 3 networks is lower for both the ND network and Sioux Falls than the Braess network. In other words, the PoA—which indicates the potential for performance improvement—is high for Braess and low for Sioux Falls. Our coordination-based approach provides an appreciable performance improvement for the ND network despite the lower PoA. However our results for Sioux Falls, which not only has a lower PoA but also is a larger network with many more drivers than the ND and Braess networks, as shown in see Table 1) do not show a

Flow Oscillation

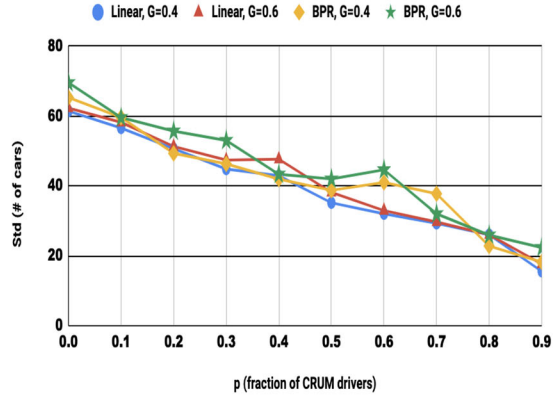
Braess' Network



(a)

Flow Oscillation

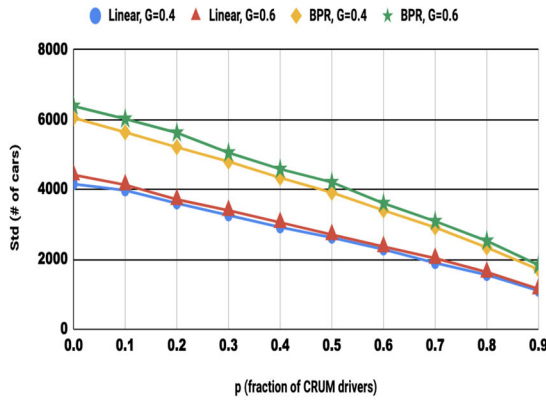
ND Network



(b)

Flow Oscillation

Sioux Falls



(c)

Figure 7. Vehicle Flow Oscillation for Networks.**Table 2.** Average travel time (given in time units) at UE and SO for different networks.

	Braess' network		ND Network		Sioux Falls' network	
	UE	SO	UE	SO	UE	SO
linear	79960	60020	50.2766	50.0249	383.4985	383.4893
BPR	79975	41326	39.6501	37.3760	20.7816	19.9598

performance advantage. As seen above, the individual travel times are higher than the theoretical UE for $p < 0.7$. In the literature (or in the available testbed repositories), almost all networks have a low PoA and in general, we expect that for low PoA networks and lower p (fraction of CRUM vehicles), the average travel time of the vehicles will be high. That said, based on the Sioux Falls results, we expect that as p increases, our approach will give a performance advantage over the Nash-based approach in larger networks with complex topologies and higher number of drivers.

Oscillation in vehicle flow:

Figure 7 describes the oscillation in vehicle flow, which is the sum of the standard deviations of the flow for each route, for different network models.

For the Braess' network shown in Figure 7(a), the flow oscillation decreases relatively quickly from $p = 0$ to $p = 0.2$. A *significant* reduction of flow oscillation is observed within this interval when a nonlinear model with $G = 0.6$ is used. *In general, the nonlinear model results in more oscillation than a linear model.* A smaller value of p decreases the number of coordinated vehicles, and a larger value of G decreases the probability of the vehicle using the CRUM-recommended route. Both parameters play a critical role in the decrease in coordination and increase in flow oscillation. Thus, when the nonlinear model is used with $G = 0.6$, the flow oscillation is much higher than those in other models in Figure 7a. When $p > 0.2$, the oscillation decreases slowly and variations in the

latency function type and parameter G do not affect the oscillation curve too much.

For the ND network (Fig. 7(b)), oscillation curves tend to decrease linearly, and no specific trend with respect to the effect of the latency function type and parameter G on the flow oscillation is observed. Note that the y-axis scale for the ND network is much smaller than the other two plots in Fig. 7 which explains the more *irregular* trends.

For the Sioux Falls network (Figure 7(c)), the oscillation curves decrease in a smooth linear fashion. Compared with the other two plots in Figure 7, there is obvious distinction with respect to oscillations in flow when models with different latency functions are used. The traffic distribution modeled with a nonlinear and a higher G value is more unstable than that with a linear model and a lower G value. From Figure 7, flow oscillation in networks with different scales and demands, exhibits different trends with varied models and G , but they all have a decreasing trend.

In summary, as the number of CRUM vehicles increases, we note that the flow oscillation for every network decreases. This supports our Goal 2 which is the CRUM-based MAS framework is able to generate a more stable traffic distribution than the traffic model that does not use any explicit coordination.

Combined with our analysis of vehicle's average travel time above, these observations enable us to achieve our robustness goals described in Goal 3 and Goal 4 which are, respectively, to study the efficiency and stability of the CRUM-based MAS framework under various parameter (G and p) settings and the performance of the CRUM-based MAS framework when different latency functions are applied.

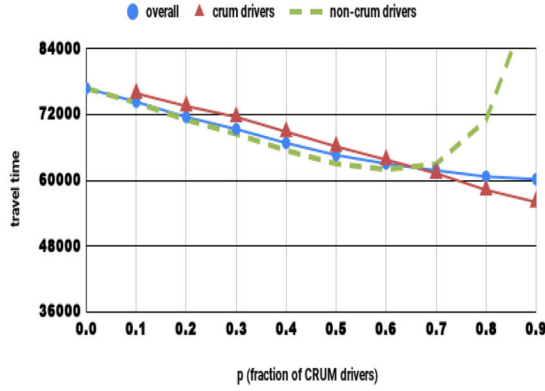
Braess network deep dive:

Of the three networks, the Braess network has the highest PoA and thus the highest potential for improvement in performance. We use this case study to further investigate the effectiveness of the CRUM-based coordinated approach to overcome the negative effects of selfish routing. As aforementioned, in Figure 6a and b, as p increases (i.e. the number of vehicles that use CRUM increases), the average travel time of all the vehicles (CRUM and selfish vehicles) in the Braess network decreases. However, the variation of G does not necessarily influence the average travel time of all the vehicles.

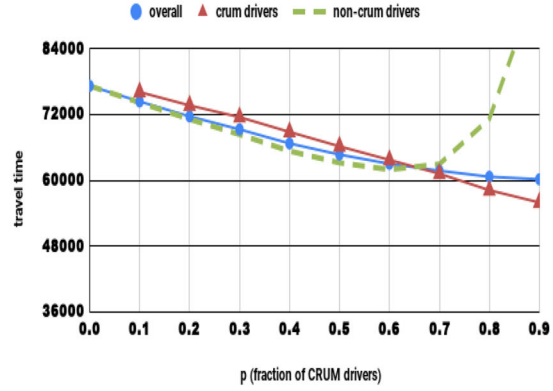
We now investigate the effect of this coordination on CRUM vehicles as opposed to selfish vehicles. Figure 8a and b focus on the average travel time of CRUM vehicles and selfish vehicles when $G = 0.4$ and 0.6 , respectively, with p ranging from 0.1 to 0.9 with a

linear latency function, while Figure 8c and d show the average travel time under the same conditions with the BPR latency function. The CRUM vehicles in both plots show the similar behavior where the average travel time is monotonically decreasing as p increases but is slightly above average travel time for the entire network for $p < 0.6$. The selfish drivers also experience a decreasing trend in their average travel time for $p < 0.6$ but then exhibit an exponential increase in average travel time as p increases. As p increases, the number of CRUM vehicles in the network increases implying the CRUM network is more coordinated and the CRUM vehicles are using the optimal routes to reduce their travel time. The number of selfish vehicles, on the other hand, decreases making them more isolated and greedy in their choices, thereby resulting in a significant increase in their travel time. We argue that CRUM vehicles would be willing to take on the initial small cost of slightly higher average travel time to receive the advantage of lower average travel times as more vehicles join the CRUM network. This motivates the incentive for drivers to join the CRUM network and shows that Goal 1 is achieved in the case of the Braess' network. In the previous section, we discussed how the coordination in the CRUM-based MAS framework leads to a more stable traffic distribution in the Braess network than using a selfish routing framework (Goal 2). We also showed the robustness of the CRUM-based is robust with respect to efficiency and stability for various parameter settings (Goal 3) and under different latency functions (Goal 4).

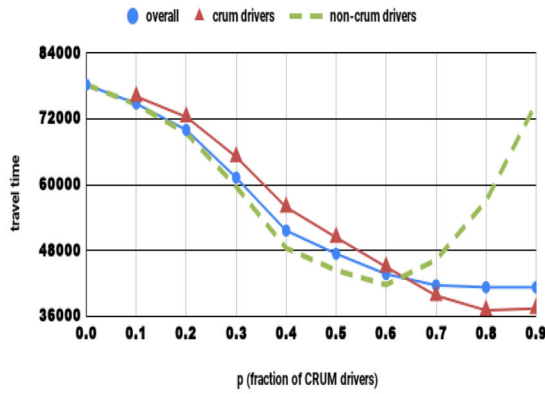
We now discuss the effect of the number of rounds on the average travel time in the Braess network. The previously described experiments are determined over 200 rounds, with the average travel time being computed from the observed car flow in the last 50 rounds when the traffic distribution on each route has stabilized. We thus argue that the rate of car flow can be viewed as a proxy for the average travel time of the vehicles. Figure 9 illustrates the convergence rate of car flow in the Braess network for varying parameter settings. The first plot Figure 9a shows the car flow for the 3 routes (SvD, SwD, and SvwD) when using the linear latency function with G taking on the values 0.4 and 0.6 and $p = 0$. It can be observed that as the number of rounds increases, the car flow on each route varies quite a bit for the first 20 rounds but then gradually stabilizes leading to the total travel time reaching its minimum value. The average travel time as well as flow oscillation are calculated from the last 50 rounds. This shows that as the number of

Average Travel TimeBraess' Network (linear latency function, $G=0.4$)

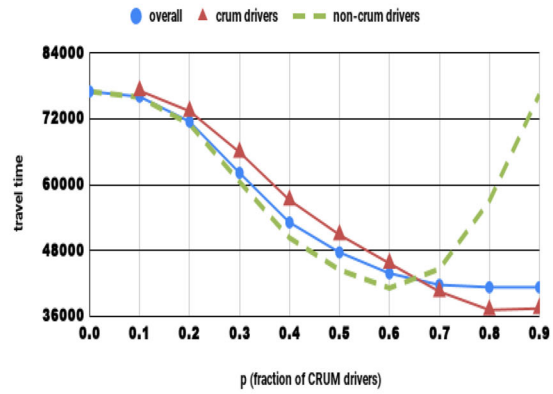
(a)

Average Travel TimeBraess' Network (linear latency function, $G=0.6$)

(b)

Average Travel TimeBraess' Network (BPR latency function, $G=0.4$)

(c)

Average Travel TimeBraess' Network (BPR latency function, $G=0.6$)

(d)

Figure 8. Average Travel time of CRUM vs Selfish Vehicles for Braess network using linear vs BPR latency functions.

rounds increases, the car flow and thus the average travel time improve until it stabilizes. The other plots Figure 9b and c shows similar trends of the car flow for the linear latency function $G=0.4$ and $G=0.6$ with $p=0.5$ and $p=0.9$, respectively. The plots Figure 9d–f show convergence of the car flow for the BPR latency function for $G=0.4$ and $G=0.6$ with $p=0.1$, 0.5 and $p=0.9$, respectively. We note that the second set of plots with the BPR show a slightly better convergence to the SO than those with the linear latency function (once route flow becomes stable) as evidenced in the previously results comparing the latency functions.

Conclusion and future work

In this article, we presented a coordinated technique to address the problem of selfish routing in transportation networks by leveraging the vehicles' real-time

data. We developed a Coordinated Route Updating Mechanism (CRUM) for coordinating vehicles by automating traffic reports and allowing real-time updates of route recommendation. The CRUM server uses real-time GPS-based data taken from the vehicles instead of the driver's reports to give route recommendations to CRUM vehicles. The CRUM model is updated after each CRUM driver travels through the network enabling the CRUM model to calculate flow along the route using the travel time as measured by the GPS. Using extensive simulations, and some instances that are representative of a broad range of traffic networks, we have demonstrated that the CRUM-enabled MAS framework is able to establish coordination among vehicles for choosing alternative routes.

While our partially decentralized approach has been designed to handle environments requiring static routing and specific time frames (e.g. morning or

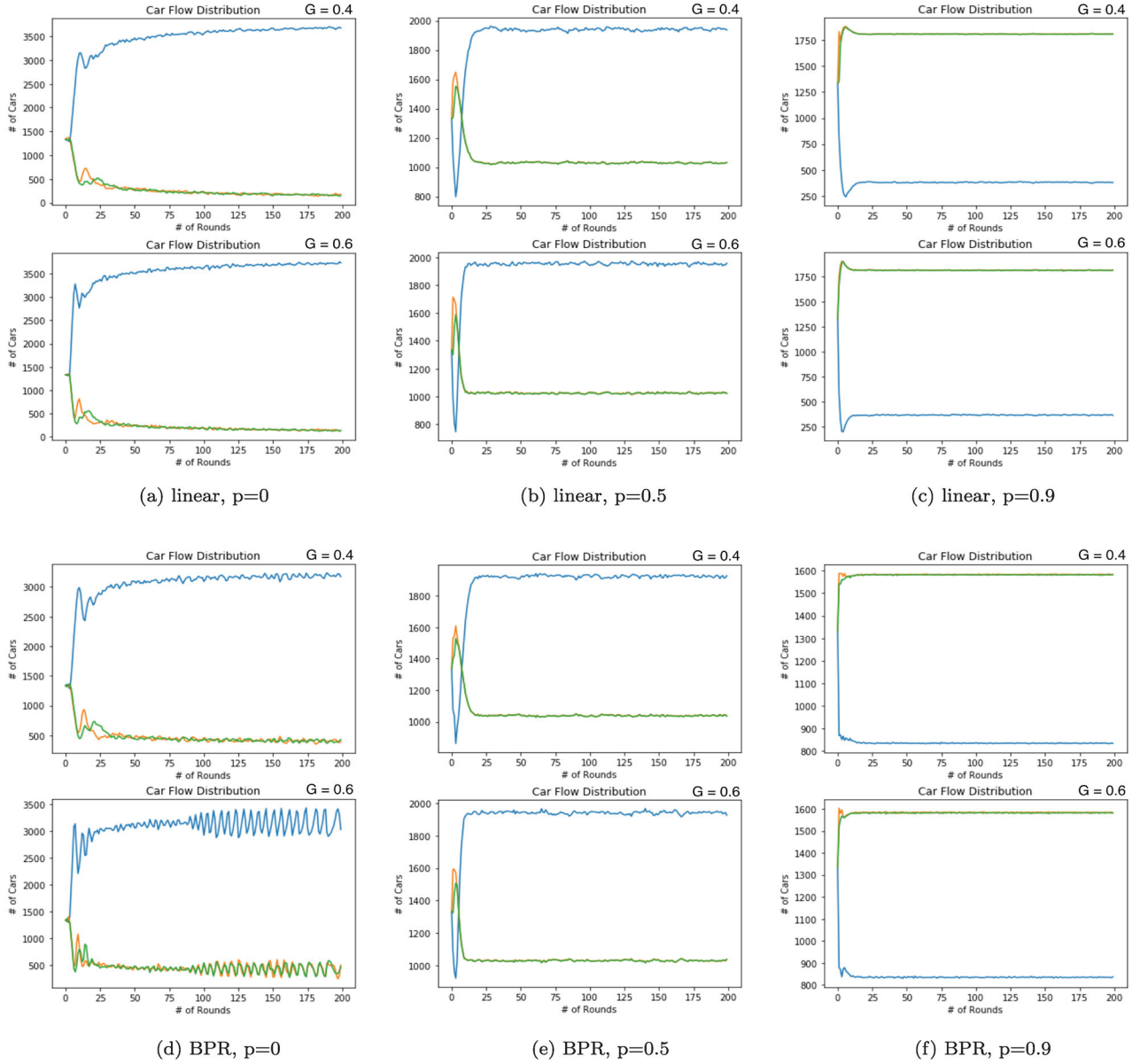


Figure 9. Plots showing Route Flow Convergence in the Braess network with Parameter settings $G = [0.4, 0.6]$ and $p = [0.0, 0.5, 0.9]$ where the orange curve represents flow on route SvD, green curve is flow on SwD, and blue curve is flow on route SvD; Figs. (a), (b) and (c) use linear latency function while Figs. (d), (e) and (f) use BPR latency function for the Braess network.

afternoon peaks) in which the demand per OD pair remains nearly constant, we argue that our approach could be extended to address the dynamic traffic assignment problem. We have recently explored (Shynkar et al., 2022) the use of meta-level control mechanisms to handle dynamic traffic situations. In this work, we use stacking of time windows where the demand remains constant within a window but differs across windows (as is commonly done in the literature) to capture the dynamics. To summarize, our main findings in this article are as follows:

- The CRUM-enabled MAS framework achieves the near SO traffic distribution by reducing congestion

and decreasing the average travel time in complex networks with a range of PoAs.

- Average travel time improves linearly with the fraction of CRUM vehicles when using the linear model of latency function.
- Traffic distribution becomes more stable as the number of CRUM vehicles increases.
- For the same network, the nonlinear model generates better traffic convergence to the SO than the linear model.

In our future work, we plan to extend our work to further investigate how network structure and latency functions affect congestion in a transportation

network. Also, we plan to augment the MAS framework with learning capabilities to dynamically estimate the flow of selfish drivers per edge which will then serve as an input to the CRUM server.

Notes

1. Agent, driver and vehicle are used interchangeably throughout the document
2. <https://cvxopt.org/>
3. <https://github.com/DavidMei99/CRUMNetlists>
4. We used 2 different values for G to study the effects of a higher fraction ($G=0.6$) of the total number of vehicles changing routes in each round versus a lower fraction ($G=0.4$). In work discussed in (Hasan et al., 2016), the authors performed experiments with a range of G values when G was 0.5 or above as opposed to when G was below 0.5. We were able to observe a similar difference in performance of this cutoff of G at $G=0.5$ by doing extensive experiments as well.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was supported by the CNPQ/CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil); PSC-CUNY jointly funded by the Professional Staff Congress and The City University of New York.

ORCID

Anita Raja  <http://orcid.org/0000-0002-0735-7358>

Ana L.C. Bazzan  <http://orcid.org/0000-0002-2803-9607>

References

- Agogino, A., & Tumer, K. (2008). Regulating air traffic flow with coupled agents. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems* (pp. 535–542). Estoril, Portugal.
- Bazzan, A. L. C. (2019). Aligning individual and collective welfare in complex socio-technical systems by combining metaheuristics and reinforcement learning. *Engineering Applications of Artificial Intelligence*, 79, 23–33. Retrieved from <https://doi.org/10.1016/j.engappai.2018.12.003>
- Bazzan, A. L. C., & Klügl, F. (2013a). *Introduction to Intelligent Systems in Traffic and Transportation* 7(3), 1–137. <https://doi.org/10.2200/S00553ED1V01Y201312AIM025>
- Bazzan, A. L. C., & Klügl, F. (2013b). A review on agent-based technology for traffic and transportation. *The Knowledge Engineering Review*, 29(3), 375–403. <https://doi.org/10.1017/S0269888913000118>
- Braess, D. (1968). Über ein paradoxon aus der verkehrsplannung. In *Unternehmensforschung Operations Research – Recherche Opérationnelle*, 12(1), 258–268. <https://doi.org/10.1007/BF01918335>
- Bucchiarone, A. (2019). Collective adaptation through multi-agents ensembles: The case of smart urban mobility. *ACM Transactions on Autonomous and Adaptive Systems*, 14(2), 1–28. <https://doi.org/10.1145/3355562>
- Buscema, D., Ignaccolo, M., Inturri, G., Pluchino, A., Rapisarda, A., & Santoro, C. (2009). Sept). The impact of real time information on transport network routing through intelligent agent-based simulation [Paper presentation]. In *Science and Technology for Humanity (TIC-STH)*, 2009 IEEE Toronto International Conference, pp. 72–77.
- Chen, B., & Cheng, H. H. (2010). A review of the applications of agent technology in traffic and transportation systems. *IEEE Transactions on Intelligent Transportation Systems*, 11(2), 485–497. <https://doi.org/10.1109/TITS.2010.2048313>
- Falek, A. M., Gallais, A., Pelsser, C., Julien, S., & Theoleyre, F. (2022). To re-route, or not to re-route: Impact of real-time re-routing in urban road networks. *Journal of Intelligent Transportation Systems*, 26(2), 198–212. <https://doi.org/10.1080/15472450.2020.1807345>
- Fleischer, L., Jain, K., & Mahdian, M. (2004). Tolls for heterogeneous selfish users in multicommodity networks and generalized congestion games [Paper presentation]. 45th Symposium on Foundations of Computer Science (FOCS 2004), 17–19 October 2004, Rome, Italy, Proceedings, (pp. 277–285). <https://doi.org/10.1109/FOCS.2004.69>
- Google map [Computer software manual]. (n.d.). <https://www.google.com/maps>
- Hasan, M. R., Bazzan, A. L. C., Friedman, E., & Raja, A. (2016). A multiagent solution to overcome selfish routing in transportation networks [Paper presentation]. 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), pp. 1850–1855. <https://doi.org/10.1109/ITSC.2016.7795856>
- Jin, Z., Hui, W., & Ping, L. (2003). Towards the applications of multi-agent techniques in intelligent transportation systems. In *Proceedings of the 2003 IEEE International Conference on Intelligent Transportation Systems*, Shanghai, China (Vol. 2, pp. 1750–1754). <https://doi.org/10.1109/ITSC.2003.1252783>
- Klein, I., Levy, N., & Ben-Elia, E. (2018). An agent-based model of the emergence of cooperation and a fair and stable system optimum using atis on a simple road network. *Transportation Research Part C: Emerging Technologies*, 86, 183–201. <https://doi.org/10.1016/j.trc.2017.11.007>
- Koutsoupias, E., & Papadimitriou, C. (1999). Worst-case equilibria. In C. Meinel, & S. Tison (Eds.), *Proceedings of the 16th annual conference on Theoretical aspects of computer science (STACS 99)* (pp. 404–413). Springer-Verlag.
- Li, Y., Courcoubetis, C., & Duan, L. (2017). Dynamic routing for social information sharing. *IEEE Journal on Selected Areas in Communications*, 35(3), 571–585. Retrieved from <https://doi.org/10.1109/JSAC.2017.2659578>
- Lim, S., & Rus, D. (2012). *Stochastic distributed multi-agent planning and applications to traffic* [Paper presentation].

- IEEE International Conference on Robotics and Automation, ICRA (pp. 2873–2879). <https://doi.org/10.1109/ICRA.2012.6224710>
- Liu, T., Abouzeid, A. A., & Julius, A. A. (2017). Traffic flow control in vehicular communication networks. In *2017 American Control Conference (ACC)*, 5513–5518. <https://doi.org/10.23919/ACC.2017.7963812>
- Liu, R., Liu, H., Kwak, D., Xiang, Y., Borcea, C., Nath, B., & Iftode, L.,. (2016). Balanced traffic routing: Design, implementation, and evaluation. *Ad Hoc Networks*, 37, 14–28. Retrieved from <https://doi.org/10.1016/j.adhoc.2015.09.001>
- New Cities Foundation. (2012). *Connected commuting: Research and analysis on the new cities foundation task force in San Jose* (Tech. Rep.). New Cities Foundation.
- Nguyen, S., & Dupuis, C. (1984). An efficient method for computing traffic equilibria in networks with asymmetric transportation costs. *Transportation Science*, 18(2), 185–202. Retrieved from <http://www.jstor.org/stable/25768128> <https://doi.org/10.1287/trsc.18.2.185>
- Open street map [Computer software manual]. (n.d.). <https://www.openstreetmap.org/>
- Ortúzar, J. d. D., & Willumsen, L. G. (2011). *Modelling transport* (4th ed.). John Wiley & Sons.
- Pathania, D., & Karlapalem, K. (2015). Social network driven traffic decongestion using near time forecasting. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems* (pp. 1761–1762). International Foundation for Autonomous Agents and Multiagent Systems.
- Petrik, O., Moura, F., & Silva, J. d A. e. (2014) The influence of the volume–delay function on uncertainty assessment for a four-step model. *Advances in Intelligent Systems and Computing*, 262, 293–306.
- Roughgarden, T., & Tardos, É. (2002). How bad is selfish routing? *Journal of the ACM*, 49(2), 236–259. <https://doi.org/10.1145/506147.506153>
- Sbayti, H., Lu, C.-C., & Mahmassani, H. (2007). Efficient implementation of method of successive averages in simulation-based dynamic traffic assignment models for large-scale network applications. *Transportation Research Record: Journal of the Transportation Research Board*, 2029(1), 22–30. <https://doi.org/10.3141/2029-03>
- Shynkar, Y., Raja, A., Bazzan, A. L., & Marinov, M. (2022). Multiagent meta-level control for adaptive traffic systems: A case study. *Transportation Research Procedia*, 62, 236–244. Retrieved from <https://www.sciencedirect.com/science/article/pii/S2352146522001570> (24th Euro Working Group on Transportation Meeting) <https://doi.org/10.1016/j.trpro.2022.02.030>
- Transportation problems hub [Computer software manual]. (n.d.). <https://github.com/bstabler/TransportationNetworks/tree/master/SiouxFalls>
- Tumer, K., Welch, Z. T., & Agogino, A. (2008). Aligning social welfare and agent preferences to alleviate traffic congestion. In *Proceedings of the 7th Int. conference on autonomous agents and multiagent systems* (pp. 655–662). IFAAMAS.
- Wang, X., Xiao, N., Xie, L., Frazzoli, E., & Rus, D. (2015). Analysis of price of anarchy in traffic networks with heterogeneous price-sensitivity populations. *IEEE Transactions on Control Systems Technology*, 23(6), 2227–2237. <https://doi.org/10.1109/TCST.2015.2410762>
- Wang, S., Djahel, S., & McManis, J. (2014). A multi-agent based vehicles re-routing system for unexpected traffic congestion avoidance [Paper presentation]. 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), (pp. 2541–2548). IEEE.
- Wardrop, J. G. (1952). *Some theoretical aspects of road traffic research* (vol. 1, pp. 325–362). <https://doi.org/10.1680/ipeds.1952.11259>
- Waze [Computer software manual]. (n.d.). <https://www.waze.com/>
- Xiao, N., Wang, X., Wongpiromsarn, T., You, K., Xie, L., Frazzoli, E., et al. (2013). *Average strategy fictitious play with application to road pricing* [Paper presentation]. In American Control Conference, ACC 2013, Washington, DC, USA, June 17–19, 2013, pp. 1920–1925. <http://ieeexplore.ieee.org/document/6580116/>
- Youn, H., Gastner, M. T., & Jeong, H. (2008). Price of anarchy in transportation networks: Efficiency and optimality control. *Physical Review Letters*, 101(12), 128701. <https://doi.org/10.1103/PhysRevLett.101.128701>
- Zhang, K., & Nie, Y. M. (2018). Mitigating the impact of selfish routing: An optimal-ratio control scheme (orcs) inspired by autonomous driving. *Transportation Research Part C: Emerging Technologies*, 87, 75–90. <https://doi.org/10.1016/j.trc.2017.12.011>