Dynamic traffic resources allocation under elastic demand of users with space–time prism constraints

Keiichiro Hayakawa\textsuperscript{a}, Eiji Hato\textsuperscript{b}

\textsuperscript{a}Toyota Central R&D Labs., Inc., 41-1, Yokomichi, Nagakute Aichi 480-1192, Japan
\textsuperscript{b}The University of Tokyo, Hongo, Bunkyo-ku, Tokyo, Japan

Abstract

We present a conceptual framework for the dynamic traffic resources allocation problem in a situation of elastic demand among customers. We introduce an activity-based model to express customers’ successive actions and transfers in order to capture the essential aspect of transfers as a derived demand. We focus on the decision-making of customers, in that they only use a mobility service when their space–time prism constraints represent the worst case. Under the setting with such elastic demand, we characterize a class of dynamic traffic resources allocation mechanisms that strictly keep space–time prism constraints of users and capacity constraints of traffic resources. Within the class of mechanisms, we show the optimal mechanism that maximizes discounted social welfare and show an exact solution algorithm for both, myopic and non-myopic settings, using the zero-suppressed binary decision diagram. We also present approximation algorithms that are still included in the class of mechanisms. In numerical studies, we showed that our proposed algorithm works effectively in settings with high rejection rates, meaning that this algorithm can be used to focus on the behavior of latent customers who have not used mobility services so far. The proposed non-myopic algorithm is suitable to design services for limited small number of customers, for example a car-sharing services for prescribed members, while the proposed myopic algorithm is suitable for designing services for a large unspecified number of customers, for example, a ride-sharing service in a large city.

Keywords: Activity-based; Dynamic traffic allocation; Space–time prism constraints; Strict capacity constraints; System optimal

1. Introduction

In many settings, traffic operators have limited traffic resources, such as airplanes, buses, trains, and taxis. Conventionally, these traffic resources were controlled under a traffic operator and customers had to buy tickets from the operator. When users traveled, they had to arrange for tickets from various traffic operators to complete their series of trips. However, recently, there are often cases in which multiple operators share traffic resources and provide a combinatorial set of services to customers, and this is often called Mobility as a Service (MaaS). The operators of these services aim to make an efficient match between limited traffic resources and heterogeneous customers. In many cases, the booking systems for such kinds of services are developed as a mobile app, and are implemented using mobile phones that are strongly connected to each user. In fact, there are many mobile apps that recommend restaurants, driving routes, and even the time taken to rest based on the habitual behavior of the users that are logged into the app within a long-term history. A traffic service operator can allocate traffic resources efficiently if they can use information on the heterogeneity of customers. In contrast, customers also benefit greatly from such services. Using such services, they can not only make arrangements for total trips by booking only once, but can also receive recommendations for activities and transfers, considering their habitual activities. Receiving such recommendations, users may change their behavior, which may enrich their lives. Considering that, the demand for such services is considered to be elastic, customers may change their destination on a trip depending on the recommendations they

\textsuperscript{*}Corresponding author: Keiichiro Hayakawa. E-mail: kei-hayakawa@mosk.tytlabs.co.jp

Preprint submitted to Elsevier

June 29, 2018
find, and may even make a decision on whether to go out or stay at home, while availing the service. Thus, the demand greatly depends on the quality of the mobile services. In this paper, we discuss the mechanisms and algorithms of such mobility services.

In such applications, customers’ space–time prism constraints (Hägerstrand, 1970) on their activities have a significant meaning. Customers have a series of constraint sets of time and location that they have to keep. For instance, in many cases, people have to return home before or at an appropriate time at night, every day. Workers have to go to work during their prescribed working hours on weekdays and thus, all activities other than the work itself, for example, going shopping or eating dinner with friends, should be assigned to either their morning or evening commutes on weekdays, while this constraint vanishes in the weekends. In the real world setting, the space–time prism constraints of users are often quite strict. For example, users cannot be late for work because being late imposes a great penalty on the users. To address such constraints, the activity-based travel analysis (Kitamura et al., 1996) is effective, which regards travel demand as a derived demand to realize a series of activities that have occurred at different locations. Using this approach, customers can be considered as agents that seek to maximize their utility under the given space–time prism constraints.

Based on this policy, Lam and Yin (2001) proposed an activity-based dynamic traffic assignment model. They define activity utility function for each activity of the users and consider the users as utility maximizers, based on the utility function and travel costs involved. Given these factors, they formulated the steady-state user equilibrium (UE) under the combined activity and route choices of users and proposed a heuristic solution algorithm using the space–time expanded network. This approach handles the activity-based elastic demand at points in which the users select a set comprising the location of the activity and the route to realize the activity, simultaneously. However, both, space–time prism constraints of users and the capacity constraints of traffic resources, are expressed by cost functions, and are thus, not strictly treated. From the viewpoint of the service operator, it is worthwhile to know the system optimal (SO) state under strict constraints.

In contrast, a considerable strand of the literature related to the dial-a-ride problem (DARP) (Cordeau and Laporte, 2007) has discussed traffic resources allocation that aims to minimize costs of the service operator under strict capacity constraints. In these works, the optimization problems are typically formulated as an integer linear programming (ILP) and a wide range of solution algorithms are provided (Ho et al., 2018). However, the problems are formulated as optimization problems of the operator, given the deterministic nature of stochastic trip-based demand of customers, and thus, they neither consider the connectivity of a series of transfers, nor the change in demand reacting to the service quality.

In this paper, we consider the setting in which user agents on mobile apps make reports on the demands and constraints of customers to the central server of the service operator that allocates a series of trips and activities to customers. We especially focus on the situation in which the demand of customers is elastic, specifically, where customers make a decision to use the service only when their demands and constraints are satisfied within the entire series of trips and activities. For example, a user uses the mobility service to attend an event held at a museum on a certain time, to go shopping after that, and to return home by a certain deadline, and does not use the service otherwise. The operator executes a combined activity and trip (e.g., route, traffic mode, etc.) assignment considering the elastic demand of customers. To address this, we introduce a framework that aims to maximize social welfare under elastic demand. The framework is based on a state transition model with normalized time-step. Thus, sequential decision-making of the operator is described consistently, on the basic economic theory. Within the framework, we characterize a class of mechanisms that strictly keep space–time prism constraints of users and capacity constraints of traffic resources. By using this class of mechanisms, the floating booking system of mobility services can be realized. Unlike common first-come first-served booking systems, by which early users have priority over any later arrivals, it rationally reallocates previously assigned traffic resources to late-coming high-value users, while guaranteeing the worst case service quality for all customers that it once accepts. It also provides users options to change their trips on the way. To realize this, we propose algorithms that use the zero-suppressed binary decision diagram (ZDD) (Minato, 1993, 2001) that provides a framework with which a sparse choice-set in a discrete state space derived from various combinatorial problems, is represented compactly and set operations related to the choice-set are computed efficiently. Radivojevic and Brewer (1996) shows that graph algorithms are superior to algorithms based on ILP in settings where resource-constrained scheduling problems with non-linear constraints and the computation cost are significantly improved using the ZDD. Bjorkman (2013) shows various applications that can be solved efficiently by using the ZDD. We take this approach to solve the traffic resources allocation problem in MaaS settings.
Overall, we make the following novel contributions.

- We characterize, for the first time, a class of dynamic activity-based traffic resources allocation mechanisms named RC (Resources Customers)-feasible mechanisms that guarantee the satisfaction of both, space–time prism constraints of customers and capacity constraints of traffic resources at any time, in its sequential decision-making. Within this class of mechanisms, we introduce the RC-optimal mechanism that has the objective function that maximizes the discounted social welfare each time.

- We propose a floating booking system that allocates traffic resources efficiently by rationally reallocating previously assigned traffic resources to late-coming high-value customers. We provide exact solution algorithms for the RC-optimal mechanism using the ZDD, and provide approximation algorithms that are still RC-feasible.

- We numerically show that our proposed solution algorithm can keep more than trillions of combinatorial trip options of current and future agents in rational computational time. We also numerically explore the trade-off between allocation efficiency and computational costs of our proposed wide range RC-feasible algorithms.

The remainder of this paper is organized as follows. In Section 2 we organize the related works in the literature. In Section 3, we show the system overview of the considered mobility service, describing the model of users and the service operator. In Section 4, we formulate the sequential decision-making of the operator that aims to maximize social welfare and characterize the RC-feasible and RC-optimal mechanisms. In Section 5, we provide the solution algorithm for both, myopic and non-myopic settings. Using ZDD, we show the exact algorithm for the RC-optimal mechanism, as well as show the approximation algorithms that are still RC-feasible. In Section 6, we numerically evaluate our proposed mechanism and discuss the selection method of myopic or non-myopic algorithm and approximation algorithms depending on the applications. Finally, in Section 7, we conclude the work and discuss the potential direction of the extension of this work.

2. Literature review

In this section, we organize the related works in literature. Specifically, we focus on four fields, namely, the dial-a-ride problem (DARP), traffic assignment with capacity constraints, the multi-agent path finding problem (MAPF), and the activity-based model.

2.1. Dial-a-ride problem (DARP)

A considerable strand of the literature has discussed traffic resources allocation that aims to minimize costs of the service operator under strict capacity constraints. For scheduling fleet management, a problem in which scheduling a set of tasks and routes for vehicles based on fully given information of demand and strict time constraints, is well-known as the Pickup and Delivery Problem with Time Windows (PDPTW) (Dumas et al., 1991). In this framework, Horn (2002) proposed myopic solutions in dynamic situations in which real-time outputs should be determined based on real-time inputs. For share-ride systems, DARP (Cordeau and Laporte, 2007) is well-known, which is a generalization of PDPTW, that includes additional constraints related to transport passengers rather than the fleet. Specifically, it aims to provide a cost-minimum traffic operation, given multiple users’ trip requests, and the related aspects of origin, destination, and time-window under limited traffic resources. This problem is typically formulated as ILP and the exact solution is proved to be NP-hard. Thus, a noticeable strand of the literature has proposed heuristic algorithms, which are extensively reviewed by Ho et al. (2018). As pointed out by Ho et al. (2018), many works with regard to DARPs discuss static situations and thus, much effort is required to address dynamic DARPs. On the other hand, dynamic ridesharing systems have been paid much attention, recently (Pelzer et al., 2015). This problem is similar to DARPs, but differ in some ways, as pointed out by Agatz et al. (2012).

Many works in these fields have proposed myopic solutions in dynamic situations, on the lines of (Horn, 2002) in PDPTW. A trivial base-line benchmark of such myopic algorithms is the first-come first-served (FCFS) algorithm, in which the system myopically determines the resources allocation each time, upon accepting user requests. To achieve more efficient allocation, requests asked within a certain time slice are collected in order to explore efficient matching within these requests in many works. Some of these have proposed graph-based algorithms to handle non-linear constraints of combinatorial requests (Kamar and Horvitz, 2009; Santi et al., 2014; Alonso-Mora et al., 2017). In
contrast, Sayarshad and Chow (2015) proposed non-myopic heuristic algorithms to solve the dynamic dial-a-ride and pricing problems by employing the Markov decision process (MDP) to explore the look-ahead policy. However, all of these works consider trip-based demands, and thus, cannot treat the correlation of successive trips. In addition, this class of problems cannot represent situations where the demand may change, depending on the service quality determined by the operator. They assume fixed demand or stochastic demand for each time and location, and do not consider the elastic demand derived from the quality of mobility services, and the achievable activities resulting from the services.

2.2. Traffic assignment with capacity constraints

In contrast to the DARP that optimizes traffic resources under given demand, the field of traffic assignment discusses customer behavior under given traffic resources. Specifically, many works discuss UE states under given traffic networks, assuming the stochastic discrete choice models of customers. In these works, the cost of transferring each link in the network is expressed by a link cost function that is (weakly) monotonically increasing with the number of customers using the same link simultaneously. The maximum capacity of the link is not strictly considered. To address this problem, Nie et al. (2004) characterized a traffic assignment problem with capacity constraints that considers the UE state on networks with link capacity constraints.

From the viewpoint of the service operator of mobility services, it is interesting to discuss optimal traffic resource allocation to maximize social welfare, or profit, considering customer behavior that has been described in the traffic assignment problem. For instance, Allsop (1974) formulate traffic signal control as a bi-level problem in which the upper level problem expresses the traffic signal control that optimizes the objective function under given demand, while the lower level problem expresses the user equilibrium states under the given traffic signal control. Solving this bi-level problem iteratively, we can obtain the solution called Nash-Cournot Equilibrium state. On the other hand, Fisk (1984) proposed to solve this bi-level problem simultaneously. Specifically, a traffic manager optimizes their policy, incorporating the reaction of the driver to the traffic control system as constraints. This approach is classified in the Stackelberg game, and formulated as a mathematical programming problem with equilibrium constraints (MPEC). The solution to this form is considered as SO states given the behavioral constraints of customers. Akamatsu and Wada (2017) proposed a demand management for static conditions as a combinatorial auction in which the SO state on networks with link capacity constraints coincide with UE states under a well-defined incentive design. The discussion is based on the Vickrey–Clarke–Groves (VCG) mechanism (Vickrey, 1961; Clarke, 1971; Groves, 1973) that consists of a traffic control achieving SO state and a well-defined pricing algorithm. However, Akamatsu and Wada (2017) only considers static settings and the extension to the dynamic settings is not trivial. Moreover, it assumes trip-based demands, and thus, cannot treat the correlation of successive trips and the related sequential decision-making of customers.

To address these problems, in this paper, we first focus on the SO states in dynamic settings, given the elastic demand with behavioral constraints of customers expressed by space–time prism constraints. Specifically, we pursue the mechanism that maximizes expected discounted social welfare, which is considered the SO state in dynamic settings in the basic economic theory (Rust, 1994). In combination with a well-defined incentive design, our proposed mechanism can also provide SO states as a result of Stackelberg game, which we discuss in Section 7.

2.3. Multi-agent Path Finding Problem (MAPF)

The MAPF problem is a problem seeking paths of multiple agents that minimize the sum of the path cost of all agents without collision (Silver, 2005). The MAPF is regarded as a generalization of the single-agent path finding problem and thus, the graph-based search algorithm such as A* is commonly used as a solution algorithm, while ILP or mixed integer program (MIP) are commonly chosen in DARP and traffic assignment problems, as stated earlier. In the MAPF problems, the state transition of multiple agents is described with normalized time. Specifically, the state consists of position and the time-step, where the space is divided into grids, and each grid can be occupied at most by one agent at a time step to avoid collision. If we regard this as a strict capacity constraint of traffic resources, the problem that we consider in this paper can be regarded as a generalization of the MAPF. Indeed, Ma et al. (2017) propose a lifelong algorithm within the online pickup and delivery setting based on MAPF. As pointed out by Amir et al. (2015), the MAPF is formulated consistently on basic economic theory, and thus, can be generalized to combinatorial auction that includes not only the common cooperative MAPF, but also the non-cooperative MAPF. To
obtain the optimal states in the setting, Amir et al. (2015) propose an algorithm using multi-valued decision diagrams (MDDs) (Srinivasan et al., 1990) to represent a massive set of plans compactly, which is a family of the binary decision diagrams (Akers, 1978), that are similar to the ZDD that we adopt in this paper.

Although the MAPF is interesting in its consistency in the basic economic theory, and is suitable to describe sequential decision-making on part of agents, many works related to MAPF only consider the myopic algorithms under fixed objectives. In this paper, we describe the dynamic traffic allocation problem in a similar manner, with MAPF, and discuss the sequential decision-making of the service operator aiming to maximize social welfare.

2.4. Activity-based model

There are earlier studies on activity-based travel analysis that regard a trip as the derived demand of users’ activities, and try to model travelers’ time usage within their daily activities. Axhausen and Gärling (1992) classified activity analysis into two conceptual frameworks, namely, utility-maximization and electric. The former assumes that people choose to spend their time in a way that maximizes utility within their space–time prism constraints, whereas the latter addresses the scheduling process explicitly. Although the problems discussed in these frameworks are indeed difficult to come to grips with, finding solutions for them are essential in enabling policy makers to manage traffic demand generated from household activities appropriately. Kitamura et al. (1996) proposed an activity-based utility model that replicates adaptive time-of-day dynamics, and introduced a simulator that offers dynamic and integrated forecasting of elements, such as transportation and land use.

The space–time expanded network is key to solving various problems that are formulated by the activity-based utility model. Such approaches were originally employed to solve dynamic traffic assignment (DTA) problems in many dynamic user equilibrium (DUE) models (Drissi-Kaitouni, 1993; Yang and Meng, 1998). This method is well-suited to activity-based travel analysis, because it expressly considers the space–time prism constraints. Indeed, Lam and Yin (2001) formulated the combined activity/route choice problem as the ideal DUE and provided a method to solve this problem using a space–time extended network, as we have already stated in Section. 1. Moreover, Arentze and Timmermans (2004) introduced the multistate supernetwork, which can represent the multimodal transport system with sequential activities. This model was inspired by the supernetwork concept as introduced by Sheffi (1985), which aims to enrich network representations in order to model traffic mode choices. In the multistate supernetwork, each node expresses a combination of an activity state and a vehicle state, and each edge expresses a transition between the states. Thus, the choice of sequential activities is expressed as a trajectory in the supernetwork and the complete trip chains that involve multiple transport modes can be obtained as a least-cost path. Liu et al. (2015) later formalized dynamic activity travel assignment as a discrete-time DUE problem on a multistate supernetwork. Oyama and Hato (2017) proposed a dynamic activity-based traffic assignment by considering users’ sequential decision-making with a recursive logit model (Fosgerau et al., 2013). However, these works use travel time as costs of edges, and consider that decisions are made at each node. Some works even consider congestion by changing the edge cost while keeping the space–time extended network as it is. This approach takes advantage of decreasing computational costs, but is not economically consistent, if we consider a sequential decision-making based on the model. On the other hand, Hara and Hato (2017) considers the optimal user and vehicle assignments for a car-sharing service, using space–time expanded network with normalized time, and considers the willingness to pay as the edge-rewards, in a manner similar to Lam and Yin (2001). They provide algorithms that maximize the social welfare by introducing the temporal and spatial connection of users within the space–time expanded network. Thus, an activity-based analysis using the space–time expanded network is not only used to express multimodal and multistate behavior, but also to develop urban planning or traffic service policies that consider user heterogeneity.

Recker (2001) discussed the relationship between trip-based and activity-based travel analysis and demonstrated that activity-based travel analysis can be formalized with mathematical programming based on traditional trip-based modeling methodologies with the addition of temporal and spatial constraints. Although we formulate problems that aim to maximize social utility under rational utility-maximizing customers, the discussion can be applied to problems that aim to minimize total travel time under rational cost-minimizing customers.

3. System Overview

We consider a MaaS system, which is implemented by a service operator. We show the system overview in Fig. 1. The traffic allocation algorithm is implemented separately from the central server of the operator and app on the
users’ mobile phones. A service request generator is implemented on the mobile app. It generates a set of candidate trip plans based on the users’ direct requests as well as information generated from the activity loggers and individual utility estimator. This is then reported to the operator. In the following part of this paper, we call this service request generator as user agent, or agent for short. Receiving requests from a set of agents, the traffic resources optimizer implemented in the operator makes a decision as to whether it allocates or rejects each request, and allocates traffic resources efficiently to the agents it accepts. The decision is notified to users through the mobile app and the traffic resources are operated as the decision, such as for example, taxis are dispatched. The operator may use the information of estimated future demand in its decision. In this paper, we focus on the traffic resources optimizer, and call it the operator. Thus, we propose algorithms implemented for the operator and the agents that appropriately allocate traffic resources to users.

We assume that decisions are made at discrete time steps \( t \in T = \{1, 2, \ldots, \bar{T}\} \), where \( T \) is the set of all time steps. The traffic network is expressed by a directed graph \( G = (N, E) \). It can represent multiple traffic modes by a concept of supernetwork (Sheffi, 1985; Arentze and Timmermans, 2004). The service operator, having a limited capacity of traffic resources for each edge and each time in the network, allocates the resource for each user. Since the traffic capacity constraints are strictly kept in the allocation, we do not consider congestion and assume that the travel time of each edge remains constant. \(^1\) We use \( \tau_e \) to denote the travel time of edge \( e \in E \). Some facilities on node \( n \in N \) may have an active time duration, such as for example, the opening time of a restaurant. We use \( b(n) \subset T \) to denote the active time duration of node \( n \).

In the following part, we introduce the model of user agent and the service operator, which is used in Section 4, where dynamic traffic allocation mechanisms are characterized.

3.1. User Agent Model

We use \( I = \{1, 2, \ldots, \bar{I}\} \) to denote a set of user agents. A type of user \( i \in I \) is expressed by origin \( O_i \in N \), destination \( D_i \in N \) and active time duration \( T_i = [t_{iB}, t_{iE}] \subset T \), where \( t_{iB} \) denotes the earliest possible time

\(^1\)This assumption is adopted in Lam and Yin (2001) as well, which discusses an activity-based traffic assignment model that aims to express time dependent UE states.
for starting activities and \( t_i^E \) denotes the strict deadline of finishing all the activities. Here, destination \( D_i \) means the final destination after visiting several locations for various activities. Typically, both, origin \( O_i \) and destination \( D_i \) are the user’s home. The service provides a series of trips from leaving his/her home after \( t_i^0 \) and until returning his/her home before \( t_i^F \).

3.1.1. Space–time prism constraints

In generating trip-plans, including activities and transfers, agent \( i \) has strict space–time prism constraints (Hagerstrand, 1970) to guarantee that the user must be on \( O_i \) at \( t_i^0 \) and on \( D_i \) at \( t_i^F \). Here, we consider a sample network shown in Fig. 2. The network has 4 nodes \( \mathcal{N} = \{A, B, C, D\} \) and 10 directed edges. The numbers in brackets show the required travel time on each edge. Specifically, two time steps are required to move between Nodes C and D, and one time step is required for the other edges.

Now, we consider an agent \( i \) with origin \( O_i = A \), destination \( D_i = D \). Using space–time expanded network (Lam and Yin, 2001), the candidate trip trajectories with given three different active time durations \( T_i \) are shown as in Fig. 3(a). In this figure, solid lines express the transfer along links and dashed lines express the staying at nodes. In cases (i), (ii), and (iii), the final time step \( t_i^E \) is set to 2, 3, and 4 and the number of candidate trajectories are 1, 5, and 15, respectively. The space–time prism constraints are tightened if the agent has another constraint related to its activity. For instance, assume that the active time duration of a facility on node B is limited as \( b(B) = \{1\} \), and the above agent \( i \) has to visit Node B during its active time duration, the candidate trajectories are decreased as shown in Fig. 3(a). Under this assumption, in case (i) with \( t_i^E = 2 \), the trip cannot be generated. In cases (ii) and (iii) with \( t_i^E = 3 \) and \( t_i^E = 4 \), the number of candidate trajectories are 2 and 1, respectively. Many users have non-linear constraints on their series of activities, such as, for example, constraints stated by if-then-else rule, and the candidate trip-plans are tightened by such constraints.

3.1.2. State transition

We introduce \( s_{ij} \in S_i \) to denote the state of agent \( i \in I \) at time \( t \in T \), that is, the state is a multi-dimensional index including for example, location \( y \), traffic mode \( m \), and so on. \( S_i \) denotes a set of states that agent \( i \) can take. We introduce the location function \( A(\cdot) \), namely the location \( y \) of an agent with state \( s_{ij} \) is obtained by \( y = A(s_{ij}) \). Note that \( y \) can be the middle of an edge when \( r_e \geq 2 \). The action that agent \( i \) with state \( s_{ij} \) takes at time \( t \) is denoted by \( a_{ij} \in \Gamma(s_{ij}) \subset \mathcal{A}_i \), where \( \Gamma(s_{ij}) \) is a set of actions that the agent can take at that time and \( \mathcal{A}_i : S_i \rightarrow S_i \) denotes a set of actions that agent \( i \) can take under all possible states. Taking action \( a_{ij} \in \Gamma(s_{ij}) \), the state of the agent transits from state \( s_{ij} \) to state \( s_{ij+1} \in S_i \). We use \( S = \bigcup_{i \in I} S_i \) to denote a set of states for all agents and \( \mathcal{A} = \bigcup_{i \in I} \mathcal{A}_i \) to denote a set of actions taken by all agents.

As actions of agents, we consider Moving and Staying. For instance, an agent located on Node C in the sample network shown in Fig. 2 at time \( t \) can take one of the following actions, namely, staying at Node C, or start moving toward Node A, B, or D. Formally, we use \( \mathcal{A}^M \subset \mathcal{A} \) to denote a set of Moving actions of agent \( i \) while \( \mathcal{A}^S \subset \mathcal{A} \) to denote a set of Staying actions, where \( \mathcal{A}_i = \mathcal{A}^M \cup \mathcal{A}^S \). A set of staying actions \( \mathcal{A}^S \) expresses a stay of agent \( i \) on any node \( n \in N \), potentially, doing some activity. We assume that the staying actions are taken only on any node \( n \in N \) in the traffic network \( \mathcal{G} \) and are not taken anywhere else such as, for example, in the middle of a edge. In contrast,
(i) \( T_i = \{0, 1, 2\} \)  
(ii) \( T_i = \{0, 1, 2, 3\} \)  
(iii) \( T_i = \{0, 1, 2, 3, 4\} \) 

Figure 3: A set of trip trajectories that keep the space–time prism constraints
moving action \( a_{ij} \in A_i \) is taken along any edge \( e \in E \) in the traffic network \( G \). We introduce the edge function \( \eta() \). The edge \( e \in E \), related to action \( a_{ij} \in A_i \) of an agent, is obtained by \( e = \eta(a_{ij}) \). For any staying action \( a_{ij} \in A_i \), \( \eta(a_{ij}) = 0 \).

Given that, we introduce a function \( \delta : S \times A \times E \to \{0, 1\} \) to express the relationship between actions and edges, as follows:

\[
\delta(s_{ij}, a_{ij}, e) = \begin{cases} 
1 & \text{if } \lambda(s_{ij}) \in \mathcal{N}, \quad \text{and } \eta(a_{ij}) = e \\
0 & \text{otherwise.} 
\end{cases} 
\]  

(1)

Specifically, \( \delta(s_{ij}, a_{ij}, e) = 1 \) if an agent \( i \) that is located on any node in the traffic network at time \( t \) take a moving action related to edge \( e \) at that time, and otherwise \( \delta(s_{ij}, a_{ij}, e) = 0 \). Given that, the traffic volume \( F_{e,t} \) that flows into edge \( e \) at time \( t \) is expressed as:

\[
F_{e,t} = \sum_{i \in I} \delta(s_{ij}, a_{ij}, e).
\]  

(2)

We assume that an agent starting the moving action at time \( t \in T \) along a edge \( e \) continues the action until it reaches the end of the edge at time \( t + \tau_e \) and does not change the action in the middle of the edge.

3.1.3. Trip plan

As shown in Section 3.1.1, actions of each agent are bounded by non-linear space–time prism constraints. We use \( L_i \) to denote a set of executable state-action trajectories that keep the constraints. A series of state-action transition \( l_i \) during the active time duration \( T_i = [t_i^B, t_i^E] \) is expressed as below:

\[
l_i = \{s_{ij}^0, a_{ij}^0, \ldots, s_{ij}^r, a_{ij}^r, s_{ij}^{r+1}\} \in L_i.
\]  

(3)

By the constraints about origin and destination, \( \lambda(s_{ij}^0) = O_i \) and \( \lambda(s_{ij}^r) = D_i \) are satisfied. The state-action trajectory \( l_i \) provides a plan for the trip-chain from the origin to the destination including multiple activities and transfers. In the following parts of the paper, we call the trajectory a trip-plan. Considering a time \( t \) such that \( t_i^B < t < t_i^E \), we use \( l_i^{(o)} \) to denote a trip-plan after time \( t \), such that:

\[
l_i^{(o)} = \{s_{ij}^{(o)}, a_{ij}^{(o)}, \ldots, s_{ij}^{r^{(o)}}, a_{ij}^{r^{(o)}}, s_{ij}^{r^{(o)+1}\}} \in L_i^{(o)},
\]  

(4)

where \( L_i^{(o)} \) denotes a set of executable trip-plans after time \( t \) while keeping the space–time prism constraints.

3.1.4. Time dependent reward function

The utility that the agents receive from activities are often dependent on the time of day. Thus, we introduce time dependent reward functions, in a manner that is similar to Axhausen and Gärling (1992); Supernak (1992); Lam and Yin (2001). Formally, we define the agents’ reward function by \( R_i : S_i \times A_i \times T \to \mathbb{R} \). Thus, the utility \( U_i(l_i) \) that agent \( i \) derives upon completing a trip-plan \( l_i = \{s_{ij}^0, a_{ij}^0, \ldots, s_{ij}^r, a_{ij}^r, s_{ij}^{r+1}\} \) is expressed as:

\[
U_i(l_i) = \sum_{i \in I} R_i(s_{ij}, a_{ij}, t_i).
\]  

(5)

The time-discounted utility \( DU_i(l_i^{(o)}) \) of agent \( i \) that takes a trip-plan \( l_i^{(o)} = \{s_{ij}^{(o)}, a_{ij}^{(o)}, \ldots, s_{ij}^{r^{(o)}}, a_{ij}^{r^{(o)}}, s_{ij}^{r^{(o)+1}\}} \) at time \( t \) is expressed as:

\[
DU_i(l_i^{(o)}) = \sum_{r=0}^{r^{(o)}-1} \beta^{t-r} R_i(s_{ij}^r, a_{ij}^r, t_i'),
\]  

(6)

where \( \beta \) is time discount rate. We assume that \( \beta \) is common for all users and the operator.

3.1.5. Reports to the operator

In the previous parts, we have defined non-linear space–time constraints and the reward function of agents. In this paper, we assume that this information is given exactly, but is available only after the beginning of the active time duration of agents. That is, information about the constraints and the reward function related to agent \( i \) is available only after \( t_i^B \). This assumption is reasonable because many users input their demands into mobile apps at the time of starting their trips. At time \( t_i^B \), user agents generate a set of executable trip-plans \( L_i \) and report the information \( \theta_i = [R_i, L_i] \), namely the reward function as well as a set of trip-plans, to the central server of the operator.
3.1.6. Decision-making on participation

Now, we consider the decision-making of users on participation with respect to mobility services. We consider users behaving based on minmax strategy to decide whether they use a service or not. Specifically, a user uses the service only when his/her space–time prism constraints are kept and the discounted utility $DU_i$ is larger than $h_i$ in the worst case scenario, where $h_i$ is the discounted utility that the user can obtain without using the service. In the following parts in this paper, we set $h_i = 0$ for all agent $i \in I$, without loss of generality.

The agent makes this decision at its start time $t_i^B$. If an agent decides to use the service, the agent accepts the trip-plan that the operator assigns to the agent. In contrast, if an agent decides not to use the service, the agent does not use any traffic resource of the operator at any time $t \geq t_i^B$, and the operator is not concerned with the agent. We use $l_i^C = \{s_i, t_i, a_i^{cancel}\}$ to denote the trip-plan of agent $i$ that decides not to use the service at $t_i^B$, where $a_i^{cancel}$ denotes the action that the agent decides not to participate.

3.1.7. Flexible behavior of users

In the previous part, we assumed static agent-type settings, where an agent reports its type $\theta_i = \{R_i, L_i\}$ at the beginning of the trip and it remains unchanged until it finishes the trip. However, agents may change their minds, for example, they may want to return home earlier, or may want to visit another place. Our proposed mechanism can easily be extended to such situations. We call this the dynamic-agent type settings. In these settings, an agent can report a new type $\theta_i' = \{R_i', L_i\}$ to the operator in an arbitrary timing.

However, to show the essential property of our proposed mechanism in simple terms, in the following part of this paper, we consider the static agent-type setting, except in Section 4.3 in which we shortly state the mechanism for the dynamic-agent type setting, and in Section 6.1.3 and 6.3.4 in which we numerically explore the effectiveness of our proposed mechanism in the dynamic-agent type setting.

3.2. Service Operator Model

As stated in the previous part, the service operator sequentially receives the agents’ report and is required to allocate traffic resources for each user adequately at each time. In this part, we focus on the operator and introduce the model of the traffic resources capacity and its allocation.

3.2.1. Capacity of traffic resources

In this part, we explain the capacity of traffic resources. We use $C_{e,t}$ to denote the capacity of edge $e \in E$ at time $t \in T$, namely at most $C_{e,t}$ users can flow into edge $e$ at time $t$. Here, the capacity of traffic resources is a generalized concept of the road capacity for private vehicles, transportation capacity of scheduled bus services, acceptable volume of users of ride-share services, and so on. The traffic volume $F_{e,t}$ on edge $e$ at time $t$ is defined as $F_{e,t} = \sum_{i \in I} \delta(s_{i,t}, a_{i,t}, e)$ by Eq. 2. For example, given the space-time trajectories of private vehicles as shown in
Fig. 4(a), five vehicles flow into link $e$ at time $t$, and thus the traffic volume is obtained as $F_{e,t} = 5$, assuming that each vehicle is occupied by only one person. In this case, the traffic capacity $C_{e,t}$ coincides with the link capacity, that is, the maximum number of vehicles that can flow into the link within a unit time, and is constant over time. In the case of scheduled buses, the space–time trajectories of which are shown in Fig. 4(b), the traffic volume is obtained as $F_{e,t} = 10$, since one bus with the capacity of 10 flows in to the link at time $t$. In this case, the traffic capacity $C_{e,t}$ is controlled by the service operator by changing the frequency and size of buses, and can be time-dependent. Finally, in the example of ride-share services, the space–time trajectories of which are shown in Fig. 4(c), the traffic volume is obtained as $F_{e,t} = 3$ since three users share the vehicles that starts moving along the link $e$ at time $t$. In this case, the traffic capacity $C_{e,t}$ is also time-dependent and is determined by the combinatorial matching of trips by users. The method of bundling the trips of users are studied in many works (Santi et al., 2014; Alonso-Mora et al., 2017; Hara and Hato, 2017).

The assignment of the spatial road capacity for each traffic mode is also an important problem to be studied. For example, traffic operators can assign road capacity to private vehicles, scheduled buses or ride share services, by setting bus priority lanes or high-occupancy vehicle (HOV) lanes. With the widened reach of automated vehicles (AVs) as a result of which road capacity can possibly be increased, the assignment of priority lanes for AVs are also considered (Chen et al., 2016). To make the discussion simple, in this paper, we assume that the traffic capacity of each traffic mode is given and represented by the capacity of edge $C_{e,t}$, note that an edge represents the transfer made by a traffic mode. Specifically, our proposed framework discusses the allocation of limited traffic resources to users with heterogeneous space–time prism constraints. However, this framework can be generalized to the settings where the capacity $C_{e,t}$ of each traffic mode is not given and only the constraints, such as road capacities, the number of vehicles, or total budgets, are given, which we discuss in Section 7.

3.2.2. Allocation of traffic resources

Given the agents’ report $\theta = \{R_i, L_i\}$ at each time, the operator dynamically makes decisions on whether it accepts or rejects the requests and also makes decisions on the allocation of traffic resources to the accepted agents. This problem is formulated as a bi-level problem in which the upper problem describes the decisions of the operator aiming to maximize social welfare, while the lower problem describes the decisions of agents regardless of whether they use the service or not. We provide an optimization model to solve this bi-level problem simultaneously, by introducing constraints for the operator, to guarantee that any agent $i$ can finish its trip within its time-space prism constraints in any situation, once the request is accepted. Specifically, one of the reported set of trip-plans $L_i$ is allocated ex-post to agent $i$ if accepted.

We use $I_{\text{reported}}^t = \{1,2, \ldots, I\} \subset I$ to denote the set of all agents that report to the operator by time $t \in T$, and use $\Theta_t = \{\theta_1, \theta_2, \ldots, \theta_I\}$ to denote all the reports that the operator has received from the agents until time $t$. In $\Theta_t$, the information about the reward functions $R_t = \{R_1, R_2, \ldots, R_I\}$ and the information about the sets of trip-plan $L_t = \{L_1, L_2, \ldots, L_I\}$ are included. The operator makes all decisions based on the information it has at that time, specifically, the information $\Theta_t$ and the observed states $s_t = \{s_{1,t}, s_{2,t}, \ldots, s_{I,t}\}$. The actions of all agents determined by the operator are denoted by $\pi_t = \{\pi_{1,t}, \pi_{2,t}, \ldots, \pi_{I,t}\}$, where $\pi_{i,t} \in \Gamma(s_{i,t})$ denotes the action that the operator allocates to agent $i$ at time $t$. Specifically, $\pi_{i,t} = l_{i,t}^t$ if the request of an agent $i$ is rejected. We use $I_t$ to denote the set of agents using the service at time $t$, meaning all agents with $t^b \leq t \leq t^e - 1$, except agents that are rejected before time $t$. We assume that the operator knows the state $s_{i,t}$ of agent $i \in I_t$ at time $t$, which is natural under the assumption that the operator provides mobility services for all accepted users.

4. Mechanism

Given the agent and the operator model stated in the previous section, we now show the traffic allocation mechanism of the service operator that guarantees to keep capacity constraints of traffic resources and space–time constraints of customers, even in the worst case, while aiming to maximize social welfare. We first show the offline optimal mechanism in a static setting and then discuss sequential mechanisms in a dynamic setting.

4.1. Offline optimal mechanism

First, we show the offline-optimal problem in which the operator knows all the information $\Theta = \{R, L\}$ of all agents beforehand, where $\Theta = \{\theta_1, \theta_2, \ldots, \theta_I\}$, $R = \{R_1, R_2, \ldots, R_I\}$, and $L = \{L_1, L_2, \ldots, L_I\}$. We define the social welfare as
follows.

**Definition 1 (Social welfare).** Social welfare is defined as the summation of rewards obtained by all agents over all time horizons.

Formally, social welfare $SW$ is given by:

$$SW(\pi|s_1) = \sum_{i \in I} \sum_{t=1}^{T-1} R_i(s_{i,t}, \pi_{i,t}, t)$$

where $\pi = \{\pi_1, \ldots, \pi_T\}$ denotes the allocation to all agents at all times and $R_i(s_{i,t}, \pi_{i,t}, t)$ denotes the sum of the rewards of all agents given the allocation $\pi_t$ under the state $s_t$. Here, $s_1$ denotes the given initial state.

Then, we define the constraints in static settings. First, we define space–time prism constraints of users as follows.

**Definition 2 (Space–time prism constraints in static settings).** We say space–time prism constraints are satisfied in static settings, if the allocated trip plans for all accepted agents are included in a set of executable trip-plans for each agent.

Formally, it is given by:

$$\forall i \in I : (\pi_i|s_{i,t}) = \{s_{i,t-1}, \pi_{i,t-1}, \ldots, s_{i,1}, \pi_{i,1}\} \in \{L_i \cup L_i\}$$

Specifically, the trip-plan allocated to agent $i$ is selected within a set of executable trip-plans $L_i$ or the request is rejected, otherwise. We also define the capacity constraints as follows.

**Definition 3 (Capacity constraints in static settings).** We say capacity constraints are satisfied in static settings, if the allocated trip plans for all agents do not violate the capacity constraints of traffic resources at any time and at any edge in the network.

Formally, it is given by:

$$\forall e \in E, \forall t \in T : \sum_{i \in I} \delta(s_{i,t}, \pi_{i,t}, e) \leq C_{e,t},$$

The left term of these constraints expresses the traffic volume that flows into the edge $e$ at time $t$, given by Eq. 2.

**Definition 4 (Offline optimal mechanism).** Offline optimal mechanism in our setting has an objective function that maximizes the social welfare $SW$ as defined by Definition 1 under space–time prism constraints defined by Definition 2 and capacity constraints defined by Definition 3.

Owing to the assumption that the operator has the perfect knowledge of future agents, the social welfare achieved by the offline optimal mechanism is the upper bound of all mechanisms in settings with limited information of future agents.

4.2. Sequential mechanisms

Then, we show the sequential mechanism in which the operator can use reports $\theta_t = \{\theta_1, \theta_2, \ldots, \theta_{I_t}\}$ from a limited set of agents $I_t^{\text{reported}} = \{1, 2, \ldots, \tilde{I}\} \subset I$ that has already reported, to decide the allocation $\pi_t = \{\pi_{1,t}, \pi_{2,t}, \ldots, \pi_{I,t}\}$ at time $t$. First, we introduce the definition of discounted social welfare based on dynamic programming (Rust, 1994), as follows.

**Definition 5 (Discounted social welfare).** The discounted social welfare at time $t$ is defined by the summation of discounted rewards obtained by all agents at and after time $t$. 

12
Formally, the discounted social welfare $DSW_t$ at time $t$ is given by:

$$DSW_t = DS W(\pi_t, \pi_{t+1}, \ldots, \pi_T|s_t) = \sum_{t=0}^{T} \beta^{t-t} R(s_t, \pi_t).$$  \hspace{1cm} (10)$$

Then, we define the constraints in dynamic settings. First, we define space–time prism constraints of users as follows.

**Definition 6** (Space–time prism constraints in dynamic settings). We say that space–time prism constraints are satisfied under the decision $\pi$ at time $t$, if there exists at least one executable trip plan for all existing agents, except agents that are rejected at time $t$.

Formally, it is given by:

$$\forall i \in I_t, \exists Q^{(i)} \in [L_0^{(i)} \cup L_{\pi}^{(i)}], \pi_{i,t} \in I_t.$$  \hspace{1cm} (11)

We also define the capacity constraints in dynamic settings as follows.

**Definition 7** (Capacity constraints in dynamic settings). We say that capacity constraints are satisfied at time $t$, if there are joint trip plans across all accepted agents, that do not violate the capacity of traffic resources at any time in the future.

Formally, it is given by:

$$\exists \Omega_{I_t} \in L_{I_t}^{(i)}, \forall e \in E, \forall t' \geq t : \sum_{i \in \Omega_{I_t}} \sum_{(s_{i,t'}, \pi_{i,t'}) \in \Omega_{I_t}^{(i)}} \delta(s_{i,t'}, \pi_{i,t'}, e) \leq C_{e,t'},$$  \hspace{1cm} (12)

where $I_t'$ denotes all agents under the service at time $t$ except agents that are rejected at that time. $I_t^{(i)}$ denotes a combination of trip plans by $I_t'$, and $L_{I_t}^{(i)}$ denotes a set of executable combinations of trip plans by $I_t'$.

Given these definitions, we define a class of mechanisms that always keep both, space–time prism constraints and capacity constraints in our dynamic settings, and call it Resources-Customers(RC)-feasible mechanism.

**Definition 8** (RC-feasible mechanism). We say the mechanism is RC-feasible in our dynamic settings, if the mechanism satisfies the space–time prism constraints as defined by Definition 6 and capacity constraints as defined by Definition 7 at any time.

It is the class of mechanisms that have arbitral objective functions and strictly keep the constraints given by Eqs. 11 and 12. Moreover, we define Resources-Customers(RC)-optimal mechanism as follows:

**Definition 9** (RC-optimal mechanism). The RC-optimal mechanism in our dynamic setting has an objective function that maximizes the discounted social welfare $DSW_t$ as defined by Definition 5 and satisfies the space–time prism constraints as defined by Definition 6 and capacity constraints as defined by Definition 7.

To maximize the discounted social welfare $DSW_t$ given by Eq. 10, we introduce the $V$-value function $V(s_t)$ that represents the expected discounted social welfare under the state $s_t$ such that;

$$V(s_t) = \mathbb{E}\left[\sum_{t=0}^{T} \beta^{t-t} R(s_t, \pi_t)\right].$$  \hspace{1cm} (13)

where $\mathbb{E}[\cdot]$ denotes the mathematical operator that expresses the expectation. We also introduce Q-value function $Q(s_t, \pi_t)$ defined as;

$$Q(s_t, \pi_t) = R(s_t, \pi_t) + \beta \cdot V(T(s_t, \pi_t)).$$  \hspace{1cm} (14)

where $T(s_t, \pi_t)$ denotes the state at time $t+1$ given the state $s_t$ and action $\pi_t$ at time $t$. This Q-value function $Q(s_t, \pi_t)$ represents the expected discounted social welfare $DSW_t$ given the allocation $\pi_t$ under the state $s_t$. By Bellman’s principle of optimality (Bellman, 1957), the objective function of the RC-optimal mechanism as defined by Definition 9 is reformulated in a recursive-form such that:

$$\max_{\pi_t \in \Omega(s_t)} Q(s_t, \pi_t).$$  \hspace{1cm} (15)
The RC-optimal mechanism maximizes the Q-value function under the constraints given by Eqs. 11 and 12 at each time. This mechanism is an instance of the RC-feasible mechanism. Another instance of the RC-feasible mechanism is the FCFS mechanism that allocates the traffic resource myopically to the early coming agents and rejects in case the request violates the capacity constraints, which is common in this kind of a setting. In addition to these two extreme examples, there exists a wide range of RC-feasible algorithms that reasonably gives priority to later-coming high-value agents in its decision of allocation, unlike the FCFS mechanism that always gives priority to the early coming agents. We call a booking system operated by a RC-feasible algorithms as the Floating booking system, in the sense that it does not totally fix the trip-plan of agents at the time of accepting the booking, but has some flexibility to accept any other high-valued agent, while guaranteeing to keep the constraints of all agents and traffic resources. We show algorithms for establishing such systems in Section 5.

4.3. RC-feasible algorithm considering flexible behaviors of agents

In this part, we briefly present an example of the RC-feasible algorithm in dynamic agent-type settings stated in Section 3.1.7, assuming that the agent can report a new type \( \theta_i' = [R_i', L_i'] \) to the operator in an arbitrary timing. We consider an agent \( i \) whose request \( \theta_i = [R_i, L_i] \) has been accepted and is on the trip at time \( t \). At this point, an RC-feasible algorithm guarantees that this agent can reach its destination \( D \) by time \( t_i^e \), within a set of trip-plan \( L_i^{(0)} \) based on its space–time constraints. Assume that this agent changes its mind and reports a new type \( \theta_i' = [R_i', L_i'] \) to the operator. Receiving this report, the mechanism makes the following decision.

- (1) Case 1, where \( L_i' \supseteq L_i^{(0)} \), namely the space–time constraints are relaxed, the mechanism accepts the new type of agents immediately and makes a decision based on this new type \( \theta_i' = [R_i', L_i'] \).

- (2) Case 2, where \( L_i' \nsubseteq L_i^{(0)} \), namely a new set of trip-plans are not a superset of the original set of trip-plans, the mechanism makes a judgment on whether it accepts this request or not. Specifically, the mechanism temporarily processes the decision at that time based on new type \( \theta_i' = [R_i', L_i'] \) and reports by other agents. If the mechanism could find a solution that does not violate the capacity constraints, the operator accepts the new request and makes the following decisions based on the new type \( \theta_i' = [R_i', L_i'] \). Otherwise, the mechanism makes the following decisions based on the original type \( \theta_i = [R_i, L_i] \).

The algorithm shown above is RC-feasible in the dynamic agent-type settings. Note that if the agent’s new request is \( L_i' = t_i^e \), that is the agent want to renounce the trip and not use any traffic resources in the future, the request is always accepted and thus, users can drop off the trip at any time.

5. Solution Algorithm

In the previous sections, we show a MaaS model and characterize the RC-feasible mechanism for establishing the floating booking system in the service. The mechanism solves two problems simultaneously, namely, deciding a set of accepted agents and allocating traffic resources. In this section, we propose solution algorithms to realize such a system that is implemented to the central server of the operator and the distributed servers connecting to users’ mobile apps. As shown in Eqs. 11 and 12, this problem includes non-linear constraints. Thus, we propose using the ZDD (Minato, 1993) to solve this problem.

We first show the algorithm of user agents that is rather simple. The user agents make a set of executable trip-plans \( L_i \) under space–time prism constraints, based on the users’ request, or the information of the active logger. To do this, agents execute a full search of the routes in the space–time expanded network, such as for example, as shown in Fig. 3. This process is computed efficiently by using ZDD (Minato, 1993). For example, when we consider the network with only 2 nodes, Node A and Node B, and a link AB with the link cost \( \tau_{AB} = 1 \), the trip-plans of an agent that start from Node A at \( t = 0 \) and arrive at Node B at \( t = 2 \) is enumerated as shown in Fig. 5(a), using the binary decision tree. In the figure, for example, \( e^{AB} \) express the action moving from Node A to Node B at \( t = 0 \), and \( \bar{e}^{AB} \) express the action staying at Node A at \( t = 0 \). The arrows express the binary-decisions, specifically, the solid arrows mean that the action is taken while the dashed arrows mean that the action is not taken. The numbers in \([0, 1]\) shown in the bottom express whether the plan is executable or not. For example, a trip plan displayed in red in the figure shows a plan that an agent moves from Node A to Node B at \( t = 0 \) and stays at Node B at \( t = 1 \). In this case, two trip-plans shown
in red and green in Fig. 5(a) are executable. By using the ZDD, this binary-decision tree is represented in a compact form as shown in Fig. 5(b). Various manipulations such as subset, union, and cross-product are efficiently computed while keeping the compact form. As another example, in Fig. 6, we show the enumerated trip-plans in the setting as shown in Fig. 3(a-iii). By using the ZDD, all of the 15 trip-plans considered in the settings where \( T_i = \{0, 1, 2, 3, 4\} \) are compactly enumerated. If the agent has to stay at Node B at \( t = 1 \), as considered in Fig. 3(b-ii), the trip-plans have to include the arrow shown in red in Fig. 6 and thus only 2 plans are extracted. This process is computed on distributed small servers that communicate with users’ mobile apps, and the obtained set of trip-plans are held in the compact form. The user agent sends this set of plans \( L_t \) as well as the reward function \( R_t \) to the central server.

In the following parts, we present the algorithm of the central server. We first show the algorithm of the RC-optimal mechanism in myopic and non-myopic settings. Then, we show the approximation algorithms to solve it in a computationally low cost manner, which are classified into the RC-feasible mechanism. After that, we discuss the trade-off among the wide range of algorithms that we provide. In this section, we assume static agent-type settings, where an agent reports its type \( \theta_i = [R_i, L_i] \) at the beginning of the trip and it remains unchanged until it ends, but the discussion can easily be extended to dynamic agent-type settings.

5.1. Myopic algorithm

First, we discuss the myopic algorithm that myopically makes decisions without using any information on future agents. The overall algorithm is shown in Algorithm 1.

**Algorithm 1 Main (Myopic)**

*Require:* \( T, \mathcal{G} = (\mathcal{N}, \mathcal{E}), I, \theta = [R, L] \)

*Ensure:* \( \pi \)

1: \( t \leftarrow 0 \)
2: \( \mathcal{Z} \leftarrow \emptyset \)
3: for \( t \in T \) do
4: \( I_t, \theta_t \leftarrow \text{RENEWAGENT}(I, \theta, t) \)
5: \( \mathcal{Z} \leftarrow \text{MYOPICUPDATE}(\mathcal{Z}, L_t) \)
6: \( \pi_t \leftarrow \text{MYOPTIMALALLOCATION}(\mathcal{Z}, R_t, s_t) \)
7: \( \mathcal{Z} \leftarrow \text{NARROWDOWN}(\mathcal{Z}, \pi_t) \)
8: end for

The initial inputs to the algorithm are control time duration \( T = \{1, 2, \ldots, \bar{T}\} \) and spatial network \( \mathcal{G} = (\mathcal{N}, \mathcal{E}) \). In addition, while receiving reports from agents at each time, the information on a set of agents \( I \) and their type \( \theta = [R, L] \) are dynamically updated (Line 4). At each time, a set of executable trip-plans for all agents are updated based on \( L_t \).
Figure 6: Enumerated trip-plans for the setting in Fig. 3(a - iii), by using the ZDD
and saved to $\mathcal{Z}$, using ZDD (Minato, 1993) (Line 5). We use $|\mathcal{Z}|$ to express the number of plans saved in $\mathcal{Z}$. Based on the reward functions $R_t$ and the state $s_t$, the algorithm determines an allocation $\pi_t$ within a set of executable trip-plans $\mathcal{Z}$ (Line 6). Given the fixed allocation $\pi_t$ at each time, the set of executable plans $\mathcal{Z}$ is narrowed down (Line 7). In the following parts, we highlight each part of this algorithm in detail.

5.1.1. Function MyopicUpdate

Algorithm 2 MyopicUpdate

Require: $\mathcal{Z}, L_t = \{L_1, L_2, \ldots, L_k\}$
Ensure: $\mathcal{Z}$
for $i \in I_t$ do
    $Z \leftarrow \text{CROSSPRODUCT}(Z, L_i)$
    $Z \leftarrow Z \setminus \text{CAPACITYVIOLATIONSET}(Z)$
end for
return $Z$

We show the pseudo code of Function MYOPICUPDATE in Algorithm 2. This function updates the set of plans $\mathcal{Z}$ based on the reported sets of trip-plans $L_t = \{L_1, L_2, \ldots, L_k\}$. Given a set of trip-plans $L_i$ of agent $i$, the algorithm first takes the cross-product of $L_i$ and $\mathcal{Z}$, which is a set of plans for agents that have already been processed. At this point, all trip-plans included in $\mathcal{Z}$ keeps the space–time prism constraints given by Eq. 11 for all agents. However, some plans obtained by this process violate the capacity constraints given by Eq. 12, so the algorithm extracts such plans by the function CAPACITYVIOLATIONSET($\mathcal{Z}$) and removes these plans from $\mathcal{Z}$. This process is repeated for all agents. Thus, we can get the full-searched executable trip-plans that keep constraints, as given by Eqs. 11 and 12. The algorithm that is described above can be computed efficiently using ZDD.

5.1.2. Function MyopicOptimalAllocation

Algorithm 3 MyopicOptimalAllocation

Require: $\mathcal{Z}, R_t, s_t$
Ensure: $\pi_t$
$\Gamma(s_t) \leftarrow \text{JOINTACTION}(s_t)$
$\mathcal{Z}^w \leftarrow \text{WEIGHT}(\mathcal{Z}, R_t)$
$\pi_t \leftarrow \text{OPTIMIZE}(s_t, \Gamma(s_t), \mathcal{Z}^w)$
return $\pi_t$

We show the pseudo code of Function MYOPICOPTIMALALLOCATION in Algorithm 3. First, the algorithm enumerates a set of joint actions $\Gamma(s_t)$ that could be taken by all agents based on state $s_t$ at the current time step $t$. Then, the algorithm weights the reward $R_t$ to the set of plans $\mathcal{Z}$. The obtained $\mathcal{Z}^w$ is a weighted graph saved in the ZDD structure. Using this, the algorithm determines the best allocation $\pi_t$ based on Eq. 15. These processes are executed by search-based optimization using the ZDD.

5.1.3. Function NarrowDown

Algorithm 4 NarrowDown

Require: $\mathcal{Z}, \pi_t$
Ensure: $\mathcal{Z}$
$\mathcal{Z} \leftarrow \text{INCLUDING}(\mathcal{Z}, \pi_t)$
return $\mathcal{Z}$

Finally, we show the pseudo code for the NARROWDOWN function in Algorithm 4. As the joint action of all agents is determined as $\pi_t$ by the mechanism at each time, this function extracts the executable plans, including the determined joint action from $\mathcal{Z}$. This process can also be computed efficiently using the ZDD.
5.2. Non-myopic algorithm

Then, we discuss the non-myopic algorithm that makes decisions using information of estimated future demand. We assume that the stochastic and elastic demand $\tilde{D}$ is given to the operator in advance although the actual demand is not revealed until the reports of user agents are received. For instance, in a car-sharing service for a small number of predescribed members, the space–time constraints of each member can be well estimated by observing the daily-activity of the member.

Algorithm 5 Main (Non-myopic)
Require: $\tilde{D}, T, G = (N, E), I, \theta = (R, L)$
Ensure: $\pi$
1: $t \leftarrow 0$
2: $Z, \tilde{L} \leftarrow \text{ENUMERATE}(\tilde{D}, T, G)$
3: for $t \in T$ do
4: $I_t, \theta_t \leftarrow \text{RENEWAGENT}(I, \theta, t)$
5: $Z \leftarrow \text{NONMYOPICUPDATE}(Z, \tilde{L}_t, L_t)$
6: $\pi_t \leftarrow \text{NONMYOPTIMALALLOCATION}(\tilde{D}, Z, R_t, s_t)$
7: $Z \leftarrow \text{NARROWDOWN}(Z, \pi_t)$
8: end for

We show an overview of the algorithm in Algorithm 5. Basically, this algorithm is similar to the myopic algorithm shown in Algorithm 1. The differences are that the non-myopic algorithm enumerates an estimated set of plans based on the given stochastic demand $\tilde{D}$ (Line 2), it uses the constraints $\tilde{L}_t$ derived from $\tilde{D}$ when updating the set of plans (Line 5), and it uses $\tilde{D}$ when making decisions on the allocation (Line 6). In the following parts, we show these processes.

5.2.1. Function ENUMERATE

Algorithm 6 Enumerate
Require: $\tilde{D}, T, G = (N, E)$
Ensure: $Z$
$L \leftarrow \emptyset$
$\tilde{L} \leftarrow \text{SAMPLECONSTRAINTS}(\tilde{D}, T, G)$
$Z \leftarrow \text{MYOPICUPDATE}(Z, \tilde{L})$
return $Z, \tilde{L}$

We show the pseudo code for the ENUMERATE function in Algorithm 6. This function aims to enumerate executable plans by possibly attending future agents in control time $T$. First, it generates space–time constraints of future agents using the stochastic demand model $\tilde{D}$, and enumerates the sets of trip-plans $\tilde{L}$ for each agent that keeps the constraints. We call this set of plans $L$ as virtual plans. Then, it generates executable trip-plans $Z$ of all agents by applying the process of function MYOPICUPDATE that we have shown in Algorithm 2.

5.2.2. Function NONMYOPICUPDATE

Algorithm 7 NonMyopicUpdate
Require: $Z, L_t, L_t = \{L_1, L_2, \ldots, L_t\}$
Ensure: $Z$
$L' \leftarrow L_t \cap \tilde{L}_t$
$Z \leftarrow \text{REMOVE}(\tilde{L}_t \setminus L')$
$Z \leftarrow \text{MYOPICUPDATE}(Z, L_t \setminus L')$
return $Z$
We show the pseudo code of Function \textsc{NonMyopicUpdate} in Algorithm 7. This function updates the set of executable trip-plans $Z$ by using the information of a set of trip-plans $L_t$ that are actually reported at time $t$. It first takes an intersection $L' = L_t \cap \hat{L}$, of the actual plans and virtual plans, and then changes the unrealized virtual plans $L_t \setminus L'$ to the unexpected actual plans $L_t \setminus L'$ in the executable trip-plans $Z$.

This function does not execute anything if the estimated virtual demands are precise, that is $L_t = \emptyset$, while it coincides with the Function \textsc{MyopicUpdate} if the virtual demands are not provided, that is $L_t = \emptyset$. Thus, it is reasonable to divide the computational efforts into two functions, \textsc{Enumerate} that can be processed in advance and \textsc{NonMyopicUpdate} that requires adaptive processes to the real-time requests. In cases in which the operator estimates the constraints of users precisely, the computational efforts in real-time processes can be decreased by using the Function \textsc{Enumerate} effectively.

5.2.3. Function \textsc{NonMyopicOptimalAllocation}

\begin{algorithm}
\caption{NonMyopicOptimalAllocation}
\begin{algorithmic}
\Require $D$, $Z$, $R_t$, $s_t$
\Ensure $\pi_t$
\State $\Gamma(s_t) \leftarrow \text{JointAction}(s_t)$
\ForAll{$a \in \Gamma(s_t)$} $Q_a \leftarrow 0$
\For{$m = 1$ to $M$}
\State $\hat{R} \leftarrow \text{SampleRewards}(D)$
\State $Z^w \leftarrow \text{Weight}(Z, (R_t \cup \hat{R}))$
\For{$a \in \Gamma(s_t)$} $Q_a \leftarrow Q_a + \text{GetQValue}(s_t, a, Z^w)$
\EndFor
\EndFor
\State $\pi_t \leftarrow \text{argmax}_{a \in \Gamma(s_t)} Q_a$
\State \Return $\pi_t$
\end{algorithmic}
\end{algorithm}

We show the pseudo code of Function \textsc{NonMyopicOptimalAllocation} in Algorithm 8. Comparing it to the Function \textsc{MyopicOptimalAllocation}, this function takes a multi-scenario approach (Chang et al., 2000) to calculate the efficient allocation that maximizes the discounted social welfare based on given estimated demands $D$.

It samples $M$ sets of reward functions $\hat{R}$ for future agents based on the demand model $D$. Then, it unifies those samples with already-reported reward functions $R_t$ of agents until the current time step and weights them to the set of plans $Z$. The obtained $Z^w$ is a weighted graph that is saved in the ZDD structure. Using this, the algorithm calculates the Q-value given by Eq. 14 for each joint action $a \in \Gamma(s_t)$. After repeating this process for all $M$ sample scenarios, the algorithm selects the action that has the maximum Q-value on average as the efficient allocation at the current time step.

Since the function \textsc{GetQValue} needs optimized calculation, the number of optimized calculations that this function needs is the number of sample scenario times the number of joint actions, including in $\Gamma(s_t)$. However, the graphs structure of $Z$ is not changed throughout these processes, and only the weight is changed for each scenario. Thus, it can be calculated efficiently by using the ZDD.

5.3. Approximation algorithm

Although the algorithm proposed in the previous part can be treated in a compact form by using ZDD, it requires large computation effort as the number of agents increases, because of the combinatorial nature of the problem. In this part, we show approximation algorithms to decrease the computational burden.

5.3.1. Making decisions for each agent

In the Function \textsc{MyopicOptimalAllocation} (Algorithm 3) or the Function \textsc{NonMyopicOptimalAllocation} (Algorithm 8), the joint actions $\Gamma(s_t)$ of all agents existing at time $t$ are considered. However, the size of $\Gamma(s_t)$ suffers from the combinatorial explosion in the number of agents. To avoid this, we propose an approximate
algorithm that makes decisions for each agent separately. The pseudo code of this approximate algorithm for myopic allocation is shown in Algorithm 9. The approximation algorithm for non-myopic allocation can also be considered similarly. We call this approximation algorithm as Per agent.

Algorithm 9 MyopicAllocation - PerAgent

Require: $Z, R_t, s_t$
Ensure: $\pi_t$

\[ Z^w \leftarrow \text{WEIGHT}(Z, R_t) \]
for $i \in I$ do

\[ \Gamma(s_{ij}) \leftarrow \text{JOIN\!ACTION}(s_{ij}) \]
\[ \pi_{ij} \leftarrow \text{OPTIMIZE}(s_{ij}, \Gamma(s_{ij}), Z^w) \]
\[ \pi_i \leftarrow \text{SET}(\pi_{ij}) \]
\[ Z^w \leftarrow \text{NARROW\!DOWN}(Z^w, \pi_{ij}) \]
end for

return $\pi_t$

Although this algorithm makes decisions for each agent’s action separately, it collects the information of the reward function $R_t$ for all agents that reports at time $t$ beforehand and uses it to achieve efficient allocation.

5.3.2. Branch cutting

By introducing the limitation on the number of options, we can decrease the computational effort reasonably. We introduce the number of maximum branches $N_{\text{branch}}^\text{max}$ in our mechanism. Namely, the number of options that the enumerated plan $Z$ always keeps;

\[ |Z| \leq N_{\text{branch}}^\text{max}. \] (16)

If the number of options exceeds $N_{\text{branch}}^\text{max}$, the only $N_{\text{branch}}^\text{max}$ options are left and the rest are cut off. There are various ways to select $N_{\text{branch}}^\text{max}$ options, for example, to select the highest $N_{\text{branch}}^\text{max}$ options in terms of rewards, to select $N_{\text{branch}}^\text{max}$ options randomly, or to use a combination of these two methods. We call this approximate algorithm as Branch cutting.

5.4. Property of proposed solution algorithms

In this part, we show the property of algorithms stated in the previous parts.

Proposition 1. The exact algorithm shown in Algorithm 1 is RC-optimal in myopic settings without any information related to future agents.

Proof. The function MYOPICUPDATE shown in Algorithm 2 provides a complete set of executable trip-plans for all agents that have reported, which satisfies the space–time constraints as defined by Eq. 11 and capacity constraints as defined by Eq. 12. Then, the decisions are made by the function MyopicOptimalAllocation as shown in Algorithm 3 that optimize the discounted social welfare based on Eq. 15 by a full-search using ZDD. Thus, the solution of the Algorithm 1 is RC-optimal, in myopic settings.

Proposition 2. Assuming that the set of estimated trip-plans $\tilde{L}$ obtained by the function ENUMERATE is precise, namely $\tilde{L} = L$, and the number of sample scenarios $M$ is set as infinitely large, the exact algorithm shown in Algorithm 5 is RC-optimal in non-myopic settings under the given stochastic demand $\tilde{D}$.

Proof. If the set of estimated trip-plans $L$ obtained by the function ENUMERATE is precise, the function NONMYOPICUPDATE does nothing. The function NONMYOPICALLOCATION with infinitely large number of samples maximizes the expected discounted social welfare as defined by Eq. 10 precisely, by maximizing Q-value given by Eq. 14 using ZDD. The solution is obtained within the set of executable trip plans $Z$ that satisfies the space–time constraints as defined by Eq. 11 and capacity constraints as defined by Eq. 12. Thus, this mechanism is RC-optimal under the given stochastic demand $\tilde{D}$.
Table 1: Summary of the proposed framework of the solution algorithms

| Myopic/Non-Myopic | Reports at a time-step | Maximum options to keep | Computational complexity | Efficiency of allocation | Algorithm |
|-------------------|------------------------|-------------------------|--------------------------|-------------------------|------------|
| Myopic            | Sequential             | \(1\) / \(N\) \(\infty\) | small                     | low                     | FCFS       |
|                   | Simultaneous           | \(1\) / \(N\) \(\infty\) | \(\uparrow\)             | \(\uparrow\)            |            |
| Non-Myopic        | Sequential             | \(1\) / \(N\) \(\infty\) | middle                    | middle                  |            |
|                   | Simultaneous           | \(1\) / \(N\) \(\infty\) | \(\uparrow\)             | \(\uparrow\)            | Extra High |

**Proposition 3.** All algorithms introduced in this section, specifically, myopic algorithm shown in Algorithm 1 and non-myopic algorithm shown in Algorithm 5, arbitrarily combined with Per agent and Branch cutting approximation is RC-feasible. The number of samples \(M\) and the size of initially enumerated samples in non-myopic algorithm can also be set arbitrarily.

**Proof.** In all algorithms, including the approximation ones, the solution is obtained within the set of executable trip plans \(Z\) that satisfies the space–time constraints as defined by Eq. 11 and capacity constraints as defined by Eq. 12. Thus, all of these mechanisms are RC-feasible.

We summarize the proposed framework of the solution algorithms, including the approximation algorithms in Table 1. We provide myopic/non-myopic algorithms that process the information reported from agents at the same time-step sequentially or simultaneously and keep the pre-defined maximum options. The framework provides a wide range of trade-offs between the computational complexity and the efficiency of allocation. At the simplest, the myopic algorithm that sequentially processes reports at a time-step and keeps only one option coincides with the FCFS algorithm that requires a low computational complexity, but is not efficient. In contrast, a non-myopic algorithm that simultaneously processes reports at a time-step and does not limit the number of options, achieves the optimal allocation that maximizes the discounted social welfare given by Eq. 10 but requires extremely high computational efforts. As shown in Table 1, we provide wide-range algorithms between these two extreme cases. All of these algorithms are RC-feasible and can be interrelatedly implemented by a graph algorithm such as the ZDD.

6. Numerical Study

Although we provide a wide-range of RC-feasible algorithms in the previous section, they have trade-offs between efficiency and computational time. In this section, we explore these trade-offs using numerical studies. In a numerical analysis, we consider a simple network representing the setting where the trip demands of tourists interferes with the background trip demand of commuters. In the study, we use Graphillion\(^2\), a Python software package on search, optimization, and enumeration using ZDD.

6.1. Experimental setup

First, we introduce a simple experimental setting to evaluate the basic performance of the proposed algorithm. In this setting, we set \(T = 8\). Specifically, we consider discrete time \(T = \{1, 2, \ldots, 8\}\) in this analysis. We also assume that the time discount rate is \(\beta = 1\). We show the experimental setup for the static agent-type setting in Table 2, which we state in detail in the following parts.

\(^2\text{https://github.com/takemaru/graphillion}\)
### Table 2: Experimental setup in the static agent-type setting

| Network | Set of nodes | \([A, B, C, D] \in N\) |
|---------|--------------|-------------------------|
|         | Set of edges | \((AB, BA, AC, CA, BC, CB, BD, DB, CD, DC) \in E\) |
|         | Travel time  | \(\tau_{AB} = \tau_{BA} = \tau_{AC} = \tau_{CA} = \tau_{BC} = \tau_{CB} = \tau_{BD} = \tau_{DB} = 1, \tau_{CD} = \tau_{DC} = 2\), as shown in Fig. 2. |
|         | Traffic resource capacity | \(\forall t: C_{AB}, t = C_{BA}, t = C_{BC}, t = C_{CB} = 1, \forall t: C_{AC}, t = C_{CA} = C_{CD}, t = C_{DC} = 3\), as shown in Fig. 7. |
|         | Active time for nodes | \(b(B) = \{3, 4, 5\}\) |

| Agents  | Passing agents \(I_{PA}^{AD}\) | Origin | Node A |
|---------|---------------------------------|--------|--------|
|         | Destination                      | Node D |
|         | Value of time and location       | \(v_i^A = 0, \ v_i^B = 0, \ v_i^C = 0, \ v_i^D \sim N(250, 10000)\) |
|         | Departure time                   | \(t_i^B\) is distributed uniformly in \([0, 1, 2]\) |
|         | Deadline                         | \(t_i^E = t_i^B + 4\) |

| Agents  | Passing agents \(I_{PA}^{DA}\) | Origin | Node D |
|---------|---------------------------------|--------|--------|
|         | Destination                      | Node A |
|         | Value of time and location       | \(v_i^A \sim N(250, 10000), \ v_i^B = 0, \ v_i^C = 0, \ v_i^D = 0\) |
|         | Departure time                   | \(t_i^B\) is distributed uniformly in \([2, 3, 4]\) |
|         | Deadline                         | \(t_i^E = t_i^B + 4\) |

| Agents  | Cruising agents \(I_C\) | Origin | Node A |
|---------|--------------------------|--------|--------|
|         | Destination              | Node A |
|         | Value of time and location| \(v_i^A \sim N(50, 400), \ v_i^B \sim N(200, 6400), \ v_i^C \sim N(100, 1600), \ v_i^D \sim N(50, 400)\) |
|         | Departure time            | \(t_i^B\) is distributed uniformly in \([0, 1, 2]\) |
|         | Deadline                  | \(t_i^E = t_i^B + 6\) |

| Additional constraints | Spend at least one time step at Node B during active time \(b(B)\) |

### 6.1.1. Network

We use the simple network that has already been shown in Fig. 2. The numbers in brackets in the figure show the required travel time on each edge. The capacity of traffic resources, that is, the possible number of agents using the edge simultaneously at one time step is shown in Fig. 7. We assume that the capacity is constant over time. In this numerical study, we consider Nodes A and D to be residential and office areas, and Nodes B and C to be amusement areas. We set the active time for the facility on Node B as \(b(B) = \{3, 4, 5\}\), while the facilities on other nodes are set to be active for the entire time duration in \(T\).
6.1.2. Passing and cruising agents

We consider two types of agents, passing and cruising agents, denoted by $I_P$ and $I_C$, respectively, on the condition that $I_P \cup I_C = I$. The main objective of passing agents is to move between places. The origin and destination of passing agents are different. In contrast, the main objective of cruising agents is spending time at some places, enriching their experience by cruising around. The origin and destination of cruising agents are the same. A key motivating example for this setting is the traffic services in tourist sites. Passing agents express the background traffic demand that is represented by morning or evening commuters, and the cruising agents express tourists cruising multiple areas in the network. In the following section, we describe the agents in this analysis in detail.

- **Passing Agents**
  
  In our analysis, we introduce passing agents as background traffic. We consider two types of passing agents, namely, traveling from Node A to Node D and vice versa, representing morning and evening commuters. Sets of these agents are denoted by $I_{PD}^D$ and $I_{PD}^A$, respectively, on the condition that $I_{PD}^D \cup I_{PD}^A = I_P$. For each agent $i \in I_{PD}^D$ traveling from Node A to Node D with the value $v_i = [v_i^A, v_i^B, v_i^C, v_i^D]$, we set the values as follows:
  
  \[ v_i^A = 0, \quad v_i^B = 0, \quad v_i^C = 0, \quad v_i^D \sim N(250, 10000), \]

  where $N(\mu, \sigma^2)$ means the normal distribution with the mean of $\mu$ and the variance of $\sigma^2$. Similarly, for each agent $i \in I_{PD}^A$, we set the values as follows:
  
  \[ v_i^A \sim N(250, 10000), \quad v_i^B = 0, \quad v_i^C = 0, \quad v_i^D = 0. \]

  We assume that the departure time $t_i^B$ of each agent $i \in I_{PD}^D$ is distributed uniformly in $[0, 1, 2]$ whereas the departure time $t_i^B$ of each agent $i \in I_{PD}^A$ is distributed uniformly in $[2, 3, 4]$. The agents cannot stay at the origin, in that they have to start immediately after they request the trip. We further assume that the deadline for the trip $t_i^D$ is set by $t_i^D = t_i^B + 4$ for all passing agents $i \in I_P$, meaning that passing agents have an upper-limit travel time of 4. However, they have to spend the final time-step to stay at the destination.

- **Cruising Agents**
  
  Unlike passing agents, we introduce cruising agents as visitors that have a large amount of flexibility on their trips. The efficiency of traffic services can be improved by using this flexibility effectively. We assume that the origin and destination of all cruising agents is Node A, representing the visitor staying at Node A and intending to cruise the area nearby. The main purpose of the cruising agents is visiting the facility on Node B. Thus, we set the constraints so that all the cruising agents must spend at least one time step at Node B during active time $b(B)$, except when the agents cannot obtain a permit and thus, cancel their trips. We set the value of these agents as follows:
  
  \[ v_i^A \sim N(50, 400), \quad v_i^B \sim N(200, 6400), \quad v_i^C \sim N(100, 1600), \quad v_i^D \sim N(50, 400). \]

  We assume that the departure time $t_i^B$ of each agent $i \in I_C$ is distributed uniformly in $[0, 1, 2]$ and the deadline for trip $t_i^D$ is set by $t_i^D = t_i^B + 6$, meaning that the cruising agents have an upper-limit travel time of 6.

  In this setting, passing agents traveling from Node A to Node D interfere with the cruising agents when they depart from Node A; passing agents traveling from Node D to Node A interfere with cruising agents when they arrive at Node D. Thus, appropriate control of traffic resources is desired.

6.1.3. Experimental setup for the dynamic agent-type setting

As stated in Section 3.1.7, we basically consider the static agent-type setting. However, in Section 6.3.4, we explore our proposed mechanism in the dynamic agent-type setting, where the reward and constraints of agents are changed by the situation. In the numerical study in Section 6.3.4, we specifically assume that cruising agents that visit Node B change their mind with a probability of 1/2, such that, $v_i^C = 3 \times v_i^C$, and $t_i^D = t_i^B + 2$. Namely, the value $v_i^C$ of staying at Node C becomes three times as high as the previous value $v_i^C$ and the deadline of the trip $t_i^D$ extends by two time steps, when compared to the previous deadline $t_i^B$. This captures the setting where a part of the visitors who visit Node B are eager to visit Node C, additionally.

---

3This setting can be expanded to demand response services at tourist sites, by introducing congestion pricing. We discuss this in Section 7.
6.2. Benchmark Mechanism

In our numerical analysis, we adopted the offline-optimal and FCFS mechanisms as benchmarks. The former is the optimal assignment stated in Definition 4, assuming that all future agents are given to the mechanism in advance. The social welfare achieved by the offline-optimal mechanism is considered the upper-limit of the given situation. FCFS is commonly introduced in settings that are considered in this study. It is not efficient in that traffic resources that were once allocated to users cannot be reallocated.

We evaluate a wide-range of RC-feasible algorithms over these benchmarks. In myopic settings, our proposed exact algorithm shown in Algorithm 1, which maximizes the discounted social welfare is called as Proposed (myopic) in this section. We also evaluate the approximate algorithms. Our proposed algorithm with Per agent approximation as shown in Algorithm 9 is called as Proposed (myopic). The algorithm with Branch cutting approximation is stated by the maximum number of options, like Per agent (myopic) with maximum branch of $N_{\text{branch}}$. Similarly, in non-myopic settings, our proposed algorithm shown in Algorithm 5 is called as Proposed (non-myopic). The approximation algorithms Per agent and Branch cutting are also considered and called Per agent (non-myopic) with maximum branch of $N_{\text{branch}}$. The number of sample scenarios used in these algorithms is set as $M = 10$.

We assume that the offline-optimal mechanism has perfect knowledge of the space–time constraints and reward functions of future agents, while our proposed non-myopic algorithms knows the stochastic demand $\tilde{D}$ that includes the deterministic space–time constraints and stochastic model of the reward function as expressed by Eqs. 17, 18, and 19. In contrast, FCFS and our proposed myopic algorithms have no knowledge of future agents at all. For dynamic agent-type setting, we assume that the offline-optimal mechanism additionally has perfect knowledge of the changing mind of agents, specifically in terms of the individual’s identity and how that agent changes in the future. In contrast, the FCFS mechanism and our proposed myopic and non-myopic algorithms have no knowledge of the changing minds of agents.

6.3. Results

In this part, we present the results of the numerical analysis and discuss the advantage of our proposed mechanism over the benchmark mechanism.

6.3.1. Efficiency

First, we evaluate the efficiency of our proposed algorithms in static agent-type setting. Here, we define efficiency as the total social welfare achieved by each mechanism as a proportion of the offline-optimal welfare. We consider
two cases, with the ratio of 10% or 20% of cruising agents over all agents, and vary the number of agents from 10 to 40. The experiment was repeated 10 trials for each setting. The results of Proposed and Per agent algorithms in myopic and non-myopic settings are shown in Fig. 8. The results in settings where the fraction of cruising agents is 0.1 and 0.2 are shown in Fig. 8(a) and Fig. 8(b), respectively. The efficiency of the FCFS mechanism is also shown in the same figure. The plots in these figures show the mean value of the trials and the error bars express 95% confidence.

As seen in Fig. 8, the efficiency of FCFS decreases with the interference of agents. On the other hand, Proposed (myopic) achieves better efficiency than FCFS does, and Proposed (non-myopic) retains almost the same quality as the optimal allocation does. While the proposed mechanism guarantees the minimum service quality to the early-coming agents with space–time prism constraints, it takes later-coming high-valued agents into consideration in its decision-making and thus, achieves high efficiency.

As for the Per agent approximation, Per agent (myopic) results in losses in the efficiency that is almost near the level of FCFS, while Per agent (non-myopic) does not result in losses in efficiency. Although the decision-making process in Per agent algorithm shown in Algorithm 9 is divided for each agent, the information of all agents reporting at the same time-step is taken into consideration in the non-myopic algorithm and thus, the efficiency loss is suppressed.

In Fig. 9, we show the number of enumerated executable plans that keep space–time constraints and capacity constraints considered in the numerical analysis stated above. As the number of agents increases, the number of executable plans also increases extensively, owing to the combinatorial nature of the problem. It reaches $1.13 \times 10^{17}$ when the number of agents is 40 and the fraction of cruising agents is 0.2. Proposed (non-myopic) offers a full-search of all these plans for making decisions and thus achieves the highest efficiency. On the other hand, Proposed (myopic) makes decisions myopically, but keeps all executable plans in the future, under the given myopic states with all agents reported until that time. This results in higher efficiency when compared to the FCFS.

Then, we evaluate the performance of Branch cutting approximation as introduced into Per agent (myopic) and Per agent (non-myopic) algorithms, running a numerical experiment in the same condition as in Fig. 8. For each algorithm, we vary the maximum branch $N_{\text{max}}^{\text{branch}}$ from 1 to 100, and evaluate the performance. In this analysis, we choose the highest $N_{\text{max}}^{\text{branch}}$ branches in terms of the expected discounted social welfare at each step of the mechanism. The results are shown in Fig. 10. Note that, in this figure, the Per agent (myopic) algorithm with maximum branch $N_{\text{max}}^{\text{branch}} = 1$ coincides with the FCFS mechanism. As we can see from this figure, the efficiency decreases as the maximum branch decreases. In these settings, the performance of the Branch cutting algorithm with maximum branch $N_{\text{max}}^{\text{branch}} = 10$ is fairly close to the performance, without the limitation of the maximum branch in myopic settings, but it still has a gap in non-myopic settings. In the setting where the fraction of cruising agents is 0.1 (Fig. 10(a)), the results
with maximum branch $\mathcal{N}_{\text{branch}}^{\text{max}} = 3$ outperform the results of FCFS in myopic settings. It implies that our proposed algorithms may achieve considerably higher efficiency than FCFS, by keeping just a few options, depending on the situation.

6.3.2. Rejection rate

In Fig. 11, we show the rejection rate as the fraction of the number of rejected agents over the number of all agents, by each algorithm. In Fig. 11(a), where the fraction of cruising agents is 0.1, the rejection rates of our proposed algorithms are lower than the FCFS. In contrast, in Fig. 11(b), where the fraction of cruising agents is 0.2, the rejection rates of our proposed non-myopic algorithms are lower, but those of our proposed myopic algorithms are higher than FCFS. In both cases, the rejection rate of our proposed non-myopic algorithms are higher than myopic algorithms when the number of agents is small, in the range from 20 to 25 in Fig. 11(a) or from 15 to 20 in Fig. 11(b).
We can see from this, that the non-myopic mechanism makes decisions to reject early-coming agents to keep traffic resources for later-coming high-value agents in order to achieve high efficiency.

As we can seen from Fig. 11, in our experimental settings, the rejection rate is considerably high and exceeds 0.4 in the largest settings. It seems to be too high if the discussion is based on an emerged demand, for example, the actual records of taxi services in the real world. However, in this paper, we aim to consider hidden demands that have not emerged because users give up on traveling before they request the service, making judgments that the trip would possibly suffer their space–time constraints. Considering that such potential users may become customers if the mobility services are improved, the rejection rates in Fig. 11 are rational. In such settings with high rejection rates, it is difficult to solve the combined problems of finding a combination of feasible agents and allocating traffic resources for those agents using ILP. Graph-based algorithms, including our proposed algorithms using the ZDD, are appropriate in these settings.

6.3.3. Calculation time

We show the calculation time of the proposed mechanism in Fig. 12. All the analysis was performed on a machine with an Intel(R)Xeon(R)E5-2690v4 CPU@2.60GHz, using a single core, and 56GB of RAM. As we can see from these figures, the calculation time of the proposed non-myopic algorithm becomes large as the number of agents increases, while that of the proposed myopic algorithm stays small. However, considering that the non-myopic algorithm treats more than a trillion options (as shown in Fig. 9) of the future combinatorial behavior of agents and repeats the optimization process for each action plan and each sample scenario shown in Algorithm 3, the calculation time is still small, owing to the nature of the ZDD.

As we have shown, our proposed non-myopic algorithm takes a large amount of computation time, but has high performance, while the basic FCFS mechanism can compute faster, but results in poor performance. Within the trade-off between the performance and computation time, our proposed framework offers a wide range of options. We can achieve better performance by adopting a myopic mechanism with an appropriate maximum branch. Depending on the situation, such as for example, a car-sharing service for limited pre-described customers in which we have enough time to compute, we can introduce a non-myopic mechanism to achieve better performance.
6.3.4. Efficiency in the dynamic agent-type setting

Then, we evaluate the efficiency in the dynamic agent-type setting as stated in Section 6.1.3. The performance of Proposed, Per agent and FCFS algorithms in both, myopic and non-myopic settings, are shown in Fig. 13. The overall trends in the results are similar to the results in Static agent-type setting as shown in Fig. 8, although the non-linearity of the performance increases, because of the settings that include the non-linear mind-change on part of the users. It is noteworthy that the gap between the performance of Per agent (myopic) and that of FCFS becomes larger in a dynamic agent-type setting than in a static agent-type setting. As we can see from this result, considering the changing minds of agents, the proposed mechanism that sequentially makes a decision at each time, performs well. In a non-myopic setting, the performance of Per agent algorithm is almost as good as that of Proposed algorithm. Thus, it can be said that a Per agent approximation is effective in both, myopic and non-myopic mechanisms, used in dynamic situations.

6.3.5. Evaluation on the large number of agents

Finally, we apply our proposed framework to a setting with a large number of agents. The network considered in this analysis is the same as the network shown in Fig. 2. However, the edge capacity is enlarged. Specifically, the capacity of all edges are multiplied by the number of agents divided by 50 together with the basic capacity as shown in Fig. 7. We vary the number of agents from 50 to 250, setting the fraction of cruising agents to 0.1, and apply our proposed myopic mechanism to the approximation algorithms Per agent and Branch Cutting. We execute a numerical study and evaluate the achieved total social welfare, varying the max branch \( N_{\text{max branch}} \) from 1 to 10000. Note the case where the max branch is set to \( N_{\text{max branch}} = 1 \) coincides with FCFS. In Fig. 14, we show the social welfare achieved by each number of max branch divided by that achieved by FCFS. As we can see from this figure, higher social welfare is achieved when we set a higher branch. For example, the proposed algorithm performs about 2 to 3% higher social welfare when the max branch is set to \( N_{\text{max branch}} = 10000 \), while it reaches 2 to 5% without the limitation of the max branch.

In our computational environment shown in Section 6.3.3, it takes only 1500 seconds to calculate the settings with 250 agents. However, owing to the nature of ZDD, the computation requires a large size of RAM. Using an environment with large sized RAM, our proposed framework can be implemented for a service with hundreds of agents.
7. Conclusions and discussions

In this study, we addressed dynamic traffic resource allocation problems with strict capacity constraints and elastic demand of users with space–time prism constraints who require the guarantee of service quality in the worst cases. We characterized this problem by using activity-based user model, in which the relationship between the successive transfers generated from activities was considered. In many settings, this problem includes many non-linear constraints. Thus we take an approach that does not use the ILP, but uses graph algorithms.

In such a setting, we characterize a class of RC-feasible mechanisms that strictly keep both, space–time prism constraints of customers and capacity constraints of traffic resources. We also showed the RC-optimal mechanism that maximizes the discounted social welfare by keeping these constraints. Based on the RC-optimal and RC-feasible mechanisms, we propose a floating booking system that allocates traffic resources efficiently by rationally reallocating previously assigned traffic resources to late-coming high-value customers. We showed exact algorithms for the RC-optimal mechanism using the ZDD for both, myopic and non-myopic settings. We also showed multiple approximation algorithms that are RC-feasible. The proposed framework provides a wide range of options from a simple FCFS algorithm to the algorithm that achieves economically dynamic social optimal states in which discounted social welfare is maximized at each time. Thus, we can find an appropriate algorithm within a trade-off between computational costs and efficiency, depending on applications.

In several studies, we showed that our proposed model can keep more than trillions of combinatorial trip options of current and future agents in rational computational time, and can thus be used effectively in settings with high rejection rates, meaning that this mechanism can be used for focusing on the behavior of latent customers who have not used mobility services so far. Specifically, our proposed framework is effective as a demand prediction tool that can consider induced demands depending on the service quality.

We also showed that our proposed Per agent approximation algorithm when introduced to the non-myopic algorithm, is effective. Commonly, the operator can achieve high efficiency by treating agents reporting in certain durations simultaneously in its decision-making, but it requires high computational costs. Our proposed mechanism solves this problem by making decisions at each agent using information from multiple agents. This approach is effective in designing mobility services with a fixed small number of customers, such as car-sharing systems with specified residents. In such settings, the space–time prism constraints of customers can be highly predictable by observing the daily active patterns of specified customers and thus, the non-myopic algorithm is suitable.

Moreover, we showed that the Per agent approximation algorithm introduced to the myopic algorithm performs much better than the common FCFS mechanism does, in settings where the type of agents changes dynamically. This approach is effective in designing mobility services for unspecified many customers, such as shared taxi systems.
in large cities. In such settings, our proposed floating booking system can provide a lot of flexibility to customers. Customers can arbitrarily change the booking as far as it does not bother other users, and in case the change is rejected, the space–time constraints that the customer originally registered are still guaranteed.

Our proposed framework brings up various directions for discussion. The first is the discussion on the speculative behavior of customers. In the floating type of bookings, customers can be better-off by reporting untruthful demand on purpose and thus, the pricing algorithm that promotes customers’ truthful reports should be discussed. To address these problems, Bergemann and Välimäki (2010) proposed the pivot mechanism by which strategy-proofness of users is guaranteed, in that customers can never be better-off by misreporting. Our proposed framework that describes decision-making of all agents at a normalized time-step is consistent with this mechanism at the point that both of them are based on a basic economic model. Indeed, we can establish the pivot mechanism by combining our proposed non-myopic RC-optimal mechanism that maximizes the expected discounted social welfare and the well-defined pivot pricing (Bergemann and Välimäki, 2010). Given that, we can design a mobility system that achieves social optimal states by the best response strategy of self-interested customers (Hayakawa and Hato, 2018). In considering the pricing algorithm, it is natural to focus not only on social welfare, but also on the profit of the operator. In MaaS systems, the operator may not hold a fixed number of traffic resources but procure them flexibly. In such a setting, the combined procurement, pricing, and allocation problems should be considered, which bring up a trade-off between profit of the operator and benefit of its customers (Hayakawa et al., 2018, 2015).

The second is the discussion on the applications. Our proposed framework presents the possibilities of many emerging applications. One instance is the electric vehicle (EV) dispatch problems for car-sharing or ride-sharing services. In such systems, the space–time constraints of vehicles should also be considered. EVs have a limited cruising distance depending on their battery capacities and thus, they have to return to a charging station before the battery runs out. Thus, the executable trajectory of vehicles is expressed in space–time prism constraints based on a series of customers and charging stations. The problem is formulated as a combination of the vehicle assignment to customers, routing, and selecting charging stations problems. This setting provides a lot of non-linear constraints and thus, our proposed framework is effective. So far, many studies have discussed this type of a problem based on trip-based formulation and not based on activity-based formulation. If it only considers myopic decision-making based on the emerged deterministic demands, the myopic trip-based approach can be reasonable. However, to provide non-myopic decision-making based on the elastic and potentially hidden demands, our proposed framework with the activity-based model will play an important role.

The third is the discussion on the various types of space–time prism constraints. Hägerstrand (1970) introduced three types of space–time prism constraints, namely, “capability constraints”, “coupling constraints” and “authority constraints.” To make the discussion simple, we only refer to the “capability constraints” in this paper. However, by introducing two other types of constraints, our proposed framework becomes more resourceful. Considering “coupling constraints” between people and people, we can express the activities with families and friends. Our proposed framework can easily be extended to such situations by introducing reward functions depending on the person doing the same activities. Considering “coupling constraints” between people and objects, such as vehicles, machines, and so on, we can consider another type of application. The EV application stated above is an instance considering the “coupling constraints” between people and vehicles. In addition, we can extend the model to logistical problems by introducing this type of constraints. Finally, if we introduce “authority constraints,” we can discuss far wider problems. For example, it can be used as a tool to design premium charge for limited priority members. It may also be useful to discuss the diversity of mobility services aiming to realize services for all people, including the elderly, children, handicapped persons, and so on. Emerging technologies in the coming age, including automated vehicles, can provide mobility services for people who cannot have their own driver’s licenses, and our proposed framework can be extended to evaluate the possibility of such achievements by new services.

However, our study still has many limitations. The first and the most important problem to be discussed is the method to measure the utility of activities. In the earlier study, Kitamura and Supernak (1997) presented the concept of “temporal utility profiles” of activities and travel, and empirically discussed it using data obtained in the San Diego Zoo, and after that, many studies approached this problem. With recent progress in information technologies, a data driven approach for this problem is promising. Another limitation to be discussed is solution algorithms to solve the problems more efficiently. Depending on applications, the RC-feasible algorithms should be refined and the specific algorithms for this should be explored. For example, A∗-based algorithms considered in MAPF settings can potentially be expanded to our settings. In the future, we plan to explore these problems.
Oyama, Y. and Hato, E. (2017). A discounted recursive logit model for dynamic gridlock network analysis. *Transportation Research Part C: Emerging Technologies*, 85:509–527.

Pelzer, D., Xiao, J., Zehe, D., Lees, M. H., Knoll, A. C., and Aydt, H. (2015). A partition-based match making algorithm for dynamic ridesharing. *IEEE Transactions on Intelligent Transportation Systems*, 16(5):2587–2598.

Radijojevic, I. and Brewer, F. (1996). A new symbolic technique for control-dependent scheduling. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(1):45–57.

Recker, W. W. (2001). A bridge between travel demand modeling and activity-based travel analysis. *Transportation Research Part B: Methodological*, 35(5):481–506.

Rust, J. (1994). Structural estimation of markov decision processes. *Handbook of econometrics*, 4:3081–3143.

Santi, P., Resta, G., Szell, M., Sobolevsky, S., Strogatz, S. H., and Ratti, C. (2014). Quantifying the benefits of vehicle pooling with shareability networks. *Proceedings of the National Academy of Sciences*, 111(37):13290–13294.

Sayarshad, H. R. and Chow, J. Y. (2015). A scalable non-myopic dynamic dial-a-ride and pricing problem. *Transportation Research Part B: Methodological*, 81:539–554.

Sheffi, Y. (1985). Urban transportation networks: equilibrium analysis with mathematical programming methods. *Traffic engineering control. Prentice-Hall, ISBN 0-13-93-972-9*.

Silver, D. (2005). Cooperative pathfinding. In *Proceedings of the 1st Artificial Intelligence and Interactive Digital Entertainment Conference, AIIDE 2005*, pages 117–122.

Srinivasan, A., Ham, T., Malik, S., and Brayton, R. K. (1990). Algorithms for discrete function manipulation. In *Computer-Aided Design, 1990. ICCAD-90. Digest of Technical Papers., 1990 IEEE International Conference on*, pages 92–95. IEEE.

Supernak, J. (1992). Temporal utility profiles of activities and travel: uncertainty and decision making. *Transportation Research Part B: Methodological*, 26(1):60–76.

Vickrey, W. (1961). Counterspeculation, auctions, and competitive sealed tenders. *The Journal of finance*, 16(1):8–37.

Yang, H. and Meng, Q. (1998). Departure time, route choice and congestion toll in a queuing network with elastic demand. *Transportation Research Part B: Methodological*, 32(4):247–260.