Using Deep Reinforcement Learning to Coordinate Multi-Modal Journey Planning with Limited Transportation Capacity

Lara CODECÁ and Vinny CAHILL

Trinity College Dublin, School of Computer Science and Statistics, Dublin, Dublin 2, Ireland.
Lara.Codeca@tcd.ie, Vinny.Cahill@tcd.ie

Abstract

Multi-modal journey planning for large numbers of simultaneous travellers is a challenging problem, particularly in the presence of limited transportation capacity. Fundamental trade-offs exist between balancing the goals and preferences of each traveller and the optimization of the use of available capacity. Addressing these trade-offs requires careful coordination of travellers' individual plans. This paper assesses the viability of Deep Reinforcement Learning (DRL) applied to simulated mobility as a means of learning coordinated plans. Specifically, the paper addresses the problem of travel to large-scale events, such as concerts and sports events, where all attendees have as their goal to arrive on time. Multi-agent DRL is used to learn coordinated plans aimed at maximizing just-in-time arrival while taking into account the limited capacity of the infrastructure. Generated plans take account of different transportation modes' availability and requirements (e.g., parking) as well as constraints such as attendees' ownership of vehicles. The results are compared with those of a naive decision-making algorithm based on estimated travel time. The results show that the learned plans make intuitive use of the available modes and improve average travel time and lateness, supporting the use of DRL in association with a microscopic mobility simulator for journey planning.

1 Introduction

Transportation capacity in cities is a limited resource whose use needs to be optimized to achieve sustainable mobility. One approach to such optimization might be through coordination of travellers' itineraries achieved by providing them with journey plans that respect their individual goals and preferences while making the best use of available capacity. One scenario in which this might be possible, and the use case considered in this paper, is where event-goers are travelling to a large-scale event, such as a concert, conference, or sports event, each with the goal of arriving as close as possible to the event's start time. The lack of coordination of travel to such events coupled with the fact that capacity may not be sufficient to satisfy such a spike in demand is often reflected in traffic congestion present before, sometimes during, and after these events [1]. We hypothesize that the venue provides an application to the event-goers that enables them to request a journey plan to reach the event on time, considering their access to different transportation modes, and in the context of other background traffic (e.g., the 'usual' traffic in the vicinity of the venue) and the plans already offered to other event-goers. The event-goers need to specify their requirements (if any), request a trip plan, and execute it.

Among the issues to be addressed to make such a model viable, including incentivizing its uptake by event-goers and their compliance with the plans provided, designing coordinated individual plans is particularly challenging as a fundamental trade-off exists between balancing the goals and requirements of each event-goer and the optimization of the use of available capacity. In this paper, we assess the viability of using Multi-Agent Deep Reinforcement Learning (MADRL) to learn coordinated multi-modal trip plans aimed at maximizing just-in-time arrival while considering the limited capacity of the infrastructure. Each event-goer is provided with a plan to arrive at the venue on time while also minimizing waiting time at the venue before the event starts. Plans are derived from a policy learnt using Reinforcement Learning (RL) [2], in which agents (representing event-goers) learn how to achieve their goal (i.e., just-in-time arrival) through trial and error while interacting with an environment by taking actions (e.g., wait or start a journey using a particular transportation mode) and receiving a reward (or punishment) depending on the expected future benefit of performing that action with respect to achieving the goal. Deep Reinforcement Learning (DRL) [3] incorporates the use of artificial neural networks to solve tasks that require a high-dimensional feature space representation. In our case, because of the need for agents to observe the effects of their actions and those of other agents on the environment, training uses a microscopic mobility simulator in which the effects of the actions performed can be assessed.

For our experiments, we made use of the Proximal Policy Optimization (PPO) [4] algorithm provided by Reinforcement Learning library (RLlib) [5], and we implemented multiple transportation environments (based on a hand-crafted synthetic mobility scenario) offering different reward models, action spaces and state space features. We then evaluated the achievement of the agents' goals under different coordination strategies and with different combinations of available transportation modes (especially those having different observed travel times). For public transportation, we varied the capacity and reliability of services and assessed the impact of parking requirements on specific vehicle types. Finally, we explored the impact of specific hard constraints, the ownership of vehicles, on achievement, leaving the study of soft constraints (e.g., user preferences) as future work.

Of particular relevance to this scenario is how the required coordination of agents' actions is achieved. Following [6], active coordination can be achieved by agents learning to explicitly exchange relevant information and make joint decisions, incurring an overhead during learning [7]. Passive coordination (whether implicit or explicit) is usually driven by the problem to be solved and the environment at hand. In our case, the transportation capacity is limited and shared. Here, implicit coordination is learnt in response to agents' observations of the effects of other agents' actions, while explicit coordination can be achieved by observing residual demand and tuning the reward model to penalize each agent when other agents' goals are not achieved (see Section 4.2 for further details). We compare both implicit and explicit passive coordination strategies for all the experimental settings.

To combine microscopic mobility simulation using Simulation of Urban MObiltiy (SUMO) [8] and the distributed DRL infrastructure provided by RLlib [5], we implemented a general-purpose framework that enables the study of a wide variety of transportation problems. The connector between the mobility simulator and the RL infrastructure has been released to the community on GitHub[1], and it is available in the official RLlib release since v1.1.0

The results show that our agents can learn coordinated multi-modal trip plans by exploring the transportation environment, figuring out how long they need to wait before leaving for the event, and which transportation mode is more appropriate to reach the destination on time. We compared our results with a naive trip plan based solely on mode preference and estimated travel time, showing that by following the learnt trip plans, there is an improvement in the travel time and lateness, and that the impact of explicit passive coordination on the waiting time before the event varies depending on the experiment characteristics.

The main contributions of this paper are the implementation and evaluation of a MADRL framework

[1] RLlib SUMO Utils: https://github.com/lcodeca/rllibsumoutils Last access: October, 2021

to study coordinated multi-modal trip planning with capacity constraints, and the demonstration that the use of DRL with a mobility simulator is a viable option to study this class of problems. Additionally, RLlib SUMO Utils (the connector for the RLlib framework and SUMO), the mobility simulations, and the MADRL framework are available on GitHub[2].

2 Related Work

Historically, a significant amount of work has been done on the use of Artificial Intelligence (AI) and Machine Learning (ML) techniques in the context of Intelligent Transportation Systems (ITS). More recently, motivated by the large quantity of transportation-related data that is becoming available as well as their success in other application domains, the use of deep learning and now DRL to build mobility models and to optimize mobility is being studied extensively. However, to date, we are not aware of any work applying DRL to the coordinated journey planning problem with constrained capacity that we consider.

2.1 Deep Reinforcement Learning in Transportation

Two extensive surveys of ML techniques, and more precisely deep learning, applied to ITS are available in [9] and [10]. Currently, even with a large volume of traffic data, it remains challenging to build reliable (predictive) models to be used to address issues such as traffic flow forecasting, travel demand prediction, traffic signal control, transportation network representation, and autonomous driving using these techniques. Moreover, the use of deep learning is constrained due to intrinsic limitations, scalability and portability.

In our case, where we are dealing with infrequent traffic patterns (i.e., due to sporadic large-scale events), it is hard to acquire the amount of data necessary to apply the usual Deep Learning (DL) methods. We address the problem by using mobility simulation as the learning environment. A simulation of the usual mobility can be built based on data on the transportation infrastructure (e.g., OpenStreetMaps [11]), mobility data, and statistical information on the population usually provided by the government (e.g., [12, 13, 14, 15, 16]). Once realistic background traffic is available, the model can be adjusted to simulate specific requirements such as capacity constraints. Moreover, in the case of RL, the effects of agents' actions on the environment can be easily modelled. Additionally, most simulators provide heuristics for travel time estimation and routing. The use of a simulated environment facilitates the development of RL methods.

Emerging applications of DRL in transportation are mostly limited to optimization of fleet dispatching, autonomous driving and Traffic Signal Control (TSC). For example, DeepPool [17] is a distributed model-free application for ride-sharing. It uses Deep Q-Network (DQN) to learn optimal dispatching of the fleet using a deep neural network trained with travel demand data. It has been tested using a real-world database of taxi trip records from New York. Another example of a distributed model-free approach to dispatching using DQN is FlexPool [18], where the optimization addresses transportation of both passengers and goods. A more generic study of large-scale fleet management is presented in [19], where the authors propose a contextual MADRL framework that uses both DQN and actor-critic algorithms to optimize dispatching and routing. Both approaches are extensively tested through empirical studies. Most of the tasks that comprise autonomous driving are surveyed in [20], where a taxonomy of the tasks is presented and the challenges of applying DRL are discussed. Moreover, the survey explicitly discusses the role of traffic simulators for learning and validation purposes. Finally, an extensive survey on the use of DRL applied to TSC is available in [21]. Here the authors discuss the metrics

[2]Persuasive: https://lcodeca.github.io/persuasive/ Last access: October, 2021

usually associated with intersection control, present the deep learning methods deployed and the traffic simulators used to test and evaluate them, discussing their limitations.

All the previous examples are a demonstration that DRL can be used to study ITS problems, and provide a step forward from DL. Additionally DRL has previously been used in similar planning problems, such as path planning for robots [22], unmanned aerial vehicles [23], and unmanned ships [24]. However, to the best of our knowledge, it is not usually applied to multi-modal trip planning, where the optimization needs to take into account the presence and behaviour of different transportation modes, and the limitations arising from constrained transportation capacity.

2.2 Multi-Modal Trip Planning Solutions

Although the problem implied by the need for coordination to address capacity constraints is not investigated, much work has been done on multi-modal trip planning using a multitude of methodologies.

The first issue to be addressed is the data aggregation required to build a layered interconnected graph representing all available modes of transport, the relevant transportation infrastructure, and possibly user preferences. An example of multi-modal transportation graph construction is presented in Trans2Vec [25], an analytical framework for multi-modal transportation recommendation. It uses joint representation learning with anchor embedding to capture the preferences of the users and the traffic demand, and its effectiveness has been evaluated using real-world data. Similarly, [26] presents a co-modal framework for optimal trip planning that uses multi-objective optimization based on genetic algorithms to provide a solution in terms of user (with preferences), mode of transport, and route. This system is tested with a real case study of Lille (Nord Pas de Calais, France). An example of exploration of the multi-modal transportation network is presented in [27], where graph embedding techniques are implemented to provide multi-modal trip plans to users. Additionally, the authors present a post-processing technique to filter out the inconsistency between the objective function and evaluation metric, and evaluate it using real-data provided by Baidu-Maps.

Another problem to be taken into account is the reliability of the estimation used to decide which plan is 'better'. Another graph-based approach is presented in [28], where the authors present a network search algorithm applied to a graph based on both scheduled and unregulated modes, aimed at improving the reliability of the itineraries built with multiple hops and multiple modes (planes included). Additionally, they investigate the amount and quality of data required to build a reliable model. Another project that takes reliability into account is [29], where delays are directly modelled and the problem is formalized as stochastic shortest path optimization. The proposed solution is meant to optimize just-in-time arrival using Q-Learning, and the validity of this approach is measured with experimental results based on the cities of Beijing, Munich and Singapore.

One additional issue arising is the complexity of generating these models and computing plans. [30] presents a solution based on dynamic transfer patterns, where the public transportation schedule is merged with the real-time passenger information and additional links are generated in the knowledge-graph to take delays into account. This methodology is among those implemented in OpenTripPlanner[3]. Additionally, the paper presents a discussion on the complexity of generating these plans, the amount of pre-computation required, and possible solutions to speed it up. Finally, two personalized multi-modal trip recommenders are presented in MTRecS-DLT [31] and RouteMe [32]. MTRecS-DLT combines convolutional neural networks and gradient-boosted decision trees to extract context features from the training data and then learns how to build trip plans that satisfy user needs while respecting their various preferences. The methodology has been tested using the dataset provided by the Context-Aware Multi-Modal Transportation Recommendation challenge promoted by Baidu. Similarly, RouteMe is a mobile recommender system that provides multi-modal plans to the users. Interestingly, it introduces

[3]OpenTripPlanner: https://www.opentripplanner.org/ Last access: October, 2021

the concept of collaboration, where through the ranking and evaluation of the plans provided by the users, collaborative filtering of the routes is applied to the knowledge-base, improving it over time. The methodology has been evaluated through a small user study (17 people) where a prototype application has been implemented and the users provided standardized feedback.

All the examples mentioned provide some limited solutions to the multi-modal trip planning problem by focusing on specific optimization requirements.

2.3 Coordination in Planning

Considering explicitly the issue of coordinated trip planning, COPTER [33] incorporates state-of-the-art solvers to deal with energy consumption reduction in multi-modal trip planning. In this case, both single user planning and cooperative decision making need to take into account the trade-offs between user acceptance and system-optimized routes, with the focus on energy savings. The evaluation is done through a simulation study based on Los Angeles, and a planned pilot deployment is discussed.

The specific problem we are studying is multi-modal trip planning with limited transportation capacity. More precisely, our problem of limited capacity cannot be addressed with construction works and long-term solutions. The transportation infrastructure is not designed to sustain the demand spike and consequent overload arising from a large-scale event. The usual solution comprises a mix between diverting traffic from the vicinity of the venue and enhancing public transportation capacity by increasing its frequency and adding shuttles [34].

To the best of our knowledge, we are the first to explore the use of microscopic mobility simulation and MADRL to deal with journey planning with passive cooperation, where the plans are discovered while exploring the transportation environment.

3 Reinforcement Learning with SUMO

RL is an ML approach to overcoming the limitations of manually-designed algorithms and policies for control and optimization. In Multi-Agent Reinforcement Learning (MARL), a set of agents interact with and explore an environment, based on which they optimize their behavior to achieve a goal. Many real-world problems offer high-dimensional state spaces so that finding an optimal solution is a hard problem. Often referred to as the curse of dimensionality, the complexity of a problem increases exponentially with the number of state dimensions. Deep learning is a branch of Machine Learning (ML) that uses artificial neural networks as function approximators to deal with such high dimensional state spaces. When deep learning is used with Reinforcement Learning (RL), we refer to Deep Reinforcement Learning (DRL) and Multi-Agent Deep Reinforcement Learning (MADRL).

MDP. RL problems are typically formalized as Markov Decision Processes (MDP). An MDP [2] is described by a tuple consisting of: a set of states S, a set of actions A, a state transition probability function $T(s_{t+1}|s_t,a_t)$ that defines the probability distribution of the next state s_{t+1} given the current state s_t and action a_t, a reward function $R(s_t,a_t)$ that determines the reward r_t (a scalar value) based on the action a_t performed in state s_t, a discount factor $\gamma \in [0,1]$ representing the impact of future rewards on the decision-making. The Markov property is a basic requirement for an MDP, and it assumes that the current state s_t contains all relevant information from the past states, implying that the state s_{t+1} depends only on the current state s_t and action a_t. Additionally, MDP models can be used to represent both continuous/infinite tasks or episodic tasks where a terminal state $s_{terminal}$ is defined.

Learning. In RL, the interaction with the environment is discretized in steps t. For every step, each agent receives information about the current state of the environment s_t. Based on s_t, each agent selects an action a_t to perform in the environment based on a policy that is learnt over time. The current policy π determines the behavior and the decisions made by the agents. In our case, the policy is a stochastic

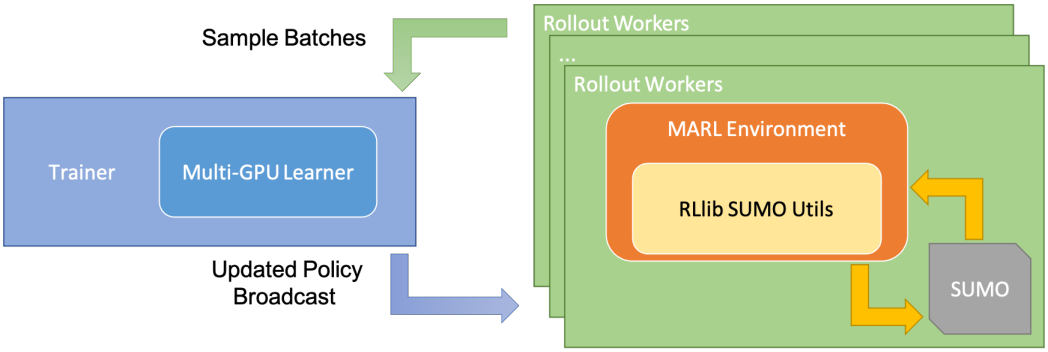


Figure 1: RLlib and SUMO Overview.

function that maps states to actions $\pi(a_t|s_t)$. The agents learn this mapping through the exploration of the environment. The learning mechanism is based on the reward r_t received from the environment after performing the action a_t. Each agent's goal is to maximize its cumulative reward in the long run.

Exploration vs Exploitation. The learning process introduces a trade-off between exploring the environment and exploiting the knowledge gathered previously. During exploration, the agent may perform random actions, trying something different from the last time a state was visited. During exploitation, the agent taps into the knowledge learnt, and performs actions that served it well in the past. The exploration vs exploitation trade-off brings up the additional issue of sampling, in which the state-action pairs need to be gathered from a sampling distribution, a dataset containing actual data, or in our case, a simulator. Deep learning typically requires a large number of samples to converge to a stable policy.

3.1 Distributed RL and Simulation of Urban MObility (SUMO)

Due to the complexity of the journey planning problem, scalability is an essential requirement of our architecture. For this reason, we make use of RAY [35], an open-source distributed execution framework used to scale ML applications. RAY provides a universal API for building distributed applications, and based on it, several libraries for solving problems in ML have been implemented. We use the Reinforcement Learning library (RLlib) [5], an open-source DRL library built on top of RAY. RLlib offers scalability and natively supports TensorFlow[4] and PyTorch[5], and is compatible with OpenAI Gym[6], a toolkit for developing and comparing RL algorithms. It provides a well-defined API to build and extend learning environments.

Simulation of Urban MObiltiy (SUMO) [8] is an open-source microscopic traffic simulator designed to handle large networks, that allows multi-modal mobility simulation, including pedestrians. SUMO implements the Traffic Control Interface (TraCI) API[7] providing interactive control of the mobility simulation. Using TraCI, we implemented RLlib SUMO Utils[8], an open-source general-purpose connector that extends the Gym environment provided by RLlib, and integrates the mobility simulation in the learning environment.

Figure 1 provides an overview of the training and evaluation infrastructure. The overall architecture

[4]TensorFlow: https://www.tensorflow.org/ Last access: October, 2021
[5]PyTorch: https://pytorch.org/ Last access: October, 2021
[6]OpenAI Gym https://gym.openai.com Last access: October, 2021
[7]SUMO Wiki: TraCI API https://sumo.dlr.de/docs/TraCI.html Last access: October, 2021
[8]RLlib SUMO Utils: https://github.com/lcodeca/rllibsumoutils Last access: October, 2021

is implemented by RLlib. The trainer (in blue) and the rollout workers (in green) can be deployed locally or on a distributed cluster thanks to the RAY orchestration. RLlib provides a high-level trainer class that handles all the interactions required to update a policy and evaluate it. Through the general-purpose trainer interface, one or more policies can be trained, checkpoints can be created and restored, and the policy can be evaluated. The rollout workers are an abstraction that implements the interactions with the learning environment, where based on a locally-stored policy, actions for the agents are selected and actuated in the environment, and samples (composed by s_t, a_t, r_t, and s_{t+1}) are collected in batches, and eventually sent to the trainer. Everything from the size of the batches, to the sampling policy is configured based on the learning algorithm. Additionally, a rollout worker can be deployed for training or for evaluation, depending on the trainer requirements. Based on the specific learning algorithm implementation[9], the number of rollout workers and multi-GPU learners may vary, and both sampling and policy updates can be asynchronous or synchronous. In general, each rollout worker can deploy more than one MARL environment to speed up sampling. However, due to constraints imposed by TraCI, only one RLlib SUMO Utils environment and the associated SUMO instance can be deployed for each rollout worker. The parallelization required to speed up sample gathering needs to be achieved by deploying multiple rollout workers at the same time. The MARL environment (in orange) implements the learning environment and it extends the general-purpose RLlib SUMO Utils Gym-like environment (in yellow). The MARL environment implements all of the APIs required by the trainer to understand both the state and action spaces, it implements the meaning of the actions and the reward model. The RLlib SUMO Utils environment takes care of setting up and moving forward the SUMO simulation, gathers the mobility information from TraCI and provides an interface to interact with the entities in the simulation.

3.2 Actor-Critic and Proximal Policy Optimization

For our purposes, we configured the implementation of a state-of-the-art actor-critic RL algorithm [2], Proximal Policy Optimization (PPO) [4], provided by RLlib.

The PPO algorithm collects a small batch of experiences while interacting with the environment, and it uses that batch to update the policy. After every policy update, the samples are discarded, and a new batch is collected with the newly updated policy. The main strength of PPO is that the changes between the old and the newly learnt policy are minimal by design. This reduces the variance in training at the cost of some bias, but improves the stability of learning, leading to faster training in costly simulated environments.

RLlib's implementation of PPO supports both TensorFlow and PyTorch multi-GPU optimizations, it allows both discrete and continuous actions (although we need only discrete actions), and it supports multiple deep neural network models (with attention and auto-regressive options). The PPO implementation we configured is synchronous, where for each training step, the learnt policy is sent to all the rollout workers. The policy is then used unchanged to gather all the samples required to fill the batches, and finally all the batches are sent back to the trainer, where the policy is updated.

In order to speed up the learning process, every agent in the system contributes to a single policy, sharing the knowledge acquired during the exploration. The stochastic policy represents a set of coordinated trip plans for the given number of agents operating in the same environment. To obtain its journey plan, an individual agent consults the policy (repeatedly) to choose actions based on its own circumstances and the state of the system. Based on the features representing the agent and the state of the system, an action (waiting or transportation mode selection) is picked and the trip plan is progressed. The metrics we use to compare the plans and evaluate the goodness of a policy are the observed travel

[9]RAY documentation: RLlib Learning Algorithms https://docs.ray.io/en/master/rllib-algorithms.html#rllib-algorithms Last access: October, 2021

time, the waiting time before the event, and the late arrival. Passive implicit cooperation is achieved through the transportation environment. Due to its limited capacity, the (selfish) decisions taken by each agent have a direct impact on the others, with increasing travel times and possible delays, to which other agents learn to react. Passive explicit cooperation is implemented by modifying the reward model (see Section 4.2), and directly penalizing each agent based on the number of other agents that failed in the task.

4 Experimental Setup

As explained, the experimental infrastructure is mainly composed of three parts, (i) the distributed orchestration provided by RAY [35], (ii) the RL infrastructure provided by RLlib [5], and (iii) the mobility simulation provided by SUMO [8]. In our case, we configured the PPO algorithm provided by RLlib, implemented an Open AI Gym-compatible learning environment integrating SUMO using RLlib SUMO Utils, and crafted a representative mobility scenario that provides the required transportation capacity limitations.

4.1 Learning Environment

The learning environment provides the mobility and transportation infrastructure. Agents represent event-goers, and their task is to explore trip plans to reach their destinations on time while their passive coordination is underpinned by the shared transportation infrastructure. The available actions depend on the experiment in question but are of two kinds: (i) *wait* (always available to each agent) and (ii) *select a transportation mode* and start the trip. Based on the experiment, the modes available are different (e.g., walk, bicycle, public transport, car, and/or powered two wheelers (PTW)), and the meaning of an action may vary (e.g., number or modes used together, requiring parking or not, the composition of public transport). During learning, all the agents independently explore the environment, but they feed all their observations to a common policy.

The features available from the environment include the *origin* and *destination* of each agent as latitude/longitude pairs, the *time to event* counting down the passage of time, discretized with a configurable parameter to 3 minutes in all experiments, and the current traffic situation computed using the *Estimated Travel Time (ETT) for each mode* provided by SUMO[10]. The ETT is discretized with the same parameter used for the time to the event. The *mode usage* is provided for event-goers that have already chosen a mode but have not reached their destinations. The *future demand* is defined by how many event-goers still need to make a mode choice. The *ownership by mode* is defined as a boolean flag, available only in specific experiments.

4.2 Reward Models

We implemented and evaluated three different reward models aimed at having all the event-goers arrive within a 15-minute window before the event, without being too early or too late. In all models, the agents receive a non-zero reward only at the end of their journey.

Simple In this model, used as a baseline, the rewards are 1 for achieving the goal of just-in-time arrival, -1 for arriving too early, -2 for being too late or when the agent does not arrive at the destination. Non-arrival may happen if the agent waits for too long, past the event time, or if a mode that is not usable is selected (e.g., a mode not available in the area, the means of transport not available to the

[10]SUMO Wiki: Estimated travel time computation https://sumo.dlr.de/docs/TraCI/Simulation_Value_Retrieval.html#command_0x87_find_intermodal_route Last access: October, 2021

agent, or the route associated to the mode is undefined). The simple model does not reward lower travel time experienced by agents or active coordination between them.

OTT The second model introduces time into the reward. The *observed time travelled*, the *waiting time* before the event, and the *late arrival* time are discretized and added to the reward as a penalty. The discretization is configurable, and in our experiments, it is set to 5 minutes. The reward function R_{OTT} for each agent is defined as

$$R_{OTT} = \begin{cases} 1 - ott & \text{if on time} \\ 0 - (ott + t_{wait}) * w_{wait} & \text{if too early} \\ 0 - (ott + t_{late}) * w_{late} & \text{if too late} \\ 0 - (ett_{max} * 2) * w_{late} & \text{if never arrived} \end{cases}$$

If the agent arrives on time (i.e., within the 15-minute window before the event), the only penalty is the observed travel time ott measured as the elapsed time from departure to arrival at the destination. In case the agent arrives too early, the waiting time t_{wait} (computed from the arrival at destination to the time of the event) is added to the ott and then multiplied by the waiting time modifier w_{wait}. In all our experiments $w_{wait} = 1.0$ for all agents. If the agent arrives too late, its lateness t_{late} is measured from the event time to the arrival at the destination. Similarly to the early arrival, t_{late} is added to the ott and then multiplied by the lateness modifier w_{late}. In all our experiments $w_{late} = 2.0$ for all the agents. In case an agent never arrives at the destination, the penalty computed is based on the assumption that the agent chose the worst transportation mode at the time of the event and arrived too late. In this case, the maximum estimated travel time ett_{max} is obtained by computing the ETT from origin to destination for each mode using SUMO, and then selecting the maximum. The penalty is then computed by doubling the ett_{max} and multiplying it with the lateness modifier w_{late}.

OTTCoord The last reward model is based on OTT but introducing an additional penalty based on how many of the other event-goers arrived too late. $R_{OTTCoord}$ is defined as

$$R_{OTTCoord} = R_{OTT} * w_{coord}$$

and is initially computed using R_{OTT}, and then multiplied by the penalty $w_{coord} \in]0,1]$ defined as

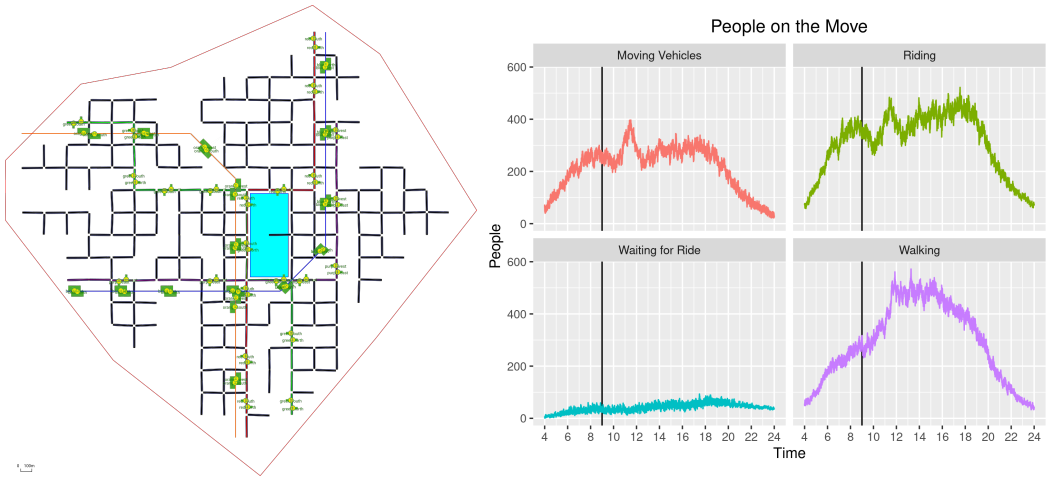
$$w_{coord} = \begin{cases} \frac{1}{\frac{1}{N_{agents}}} & if \sum A_{early} + \sum A_{ontime} = 0 \\ \frac{1}{\frac{\sum A_{early} + \sum A_{ontime}}{N_{agents}}} & otherwise \end{cases} \quad (1)$$

and representing all the agents that arrived late (or never). In the equation, N_{agents} is the number of agents, A_{early} is the number of agents that arrived early, and A_{ontime} is the number of agents that arrived on time. If every agent has a viable plan, and there are no late agents, $w_{coord} = 1$ and there is no penalty.

4.3 Mobility Scenario

The mobility models provided by SUMO allow us to configure different behaviours based on the vehicle type in use. With SUMO's sub-lane model, [36] it is possible to have smaller vehicles (e.g., bicycles and motorbikes) seeping through the rest of the traffic. For pedestrians, we use SUMO's striping model[11], where people walk on sidewalks and pedestrian crossings, and are capable of interacting with vehicles, increasing congestion, and delaying each other.

[11] SUMO Wiki: Striping Model definition https://sumo.dlr.de/docs/Simulation/Pedestrians.html#model_striping Last access: October, 2021



(a) SUMO Network

(b) Mobility plans generated with SAGA.

Figure 2: SUMO Random Grid scenario

We configured a mobility scenario based on a random grid topology that would not serve the 1000 agents we use in training without optimization of the load among the available transport modes and a staggered departure time for each agent. In Figure 2a, the roads are mapped in black, and the blue box in the middle represents the venue, with its main entrance in the middle. The yellow/green boxes are bus and metro stops. We introduced two metro lines and three bus lanes. The coverage is uneven by design, and the buses share the road with the other vehicles. We have the capability of introducing variations to the public transportation infrastructure depending on the requirements of different experiments. Figure 2b presents the background mobility generated with SAGA [37], an activity-based multi-modal traffic generator for SUMO. Based on the default activity chains provided by the tool, we generated 24-hours of background mobility for 20K people. In red, the moving vehicles are composed of bicycles, cars, taxis, PTW, and public transport. The green line shows all the people riding in or driving a vehicle (taxi drivers and public transport drivers are not included). All the people waiting for a ride are the bottom line in blue. Finally, all the people that are walking are shown in purple. As a reference, the black vertical line is at 9 AM, the start of the event. We increased or reduced the number of parking spaces depending on the experiments' requirements. Given that the cars and PTW may need to park, delays due to cruising for parking and then walking to the destination are introduced.

4.4 MARL Environment and Agent Initialization

The MARL environment is initialized for each training run, and the agents are re-initialized for each episode. Every agent has the same destination (the venue's location) and expected arrival time (the event's start time). Their origins are randomized and uniformly distributed over the map. The possible actions (including the transport modes) are the same for all the agents; however, the usable modes may differ based on their origin, and the agents need to discover which of the possible modes are viable by exploring the environment.

Given that the agents need to learn how long they should wait before departure, we evaluated three different Pareto start-time distributions. The three distributions are tuned to vary the *minimum possible competition* (hence, coordination) required to achieve the goal.

- $30min_{\alpha=3}$: every agent has a minimum 30-minute time window before the expected arrival time. With this distribution, it may not be possible to have everyone arrive on time, and the competition for transportation resources is at its highest.
- $45min_{\alpha=4}$: everyone has a minimum time window of 45-minutes to reach the event.
- $60min_{\alpha=5}$: the minimum time window is 60 minutes. With a longer time before the event, the exploration takes longer, but there is the least possible competition, and the agents have a higher chance to spread their departures over time.

5 Evaluation and Results

This section shows how agents can learn coordinated multi-modal trip plans by exploring the transportation environment. The metrics we used to compare the different experiments are *travel time*, *waiting time at destination*, *lateness*, and the *the distribution of transportation mode usage*. In the following sections we present a representative subset of the experiments we ran during the study. The complete set of graphs is available on GitHub[12]. Due to the use of SUMO, the environment reflects the differing characteristics typically observed for different transportation modes (e.g., relative speed/immunity to congestion). Intuitively, the plans discovered by the agents should respect these differences facilitating the interpretation of the results.

DRL is notoriously unstable during training [38]. Once learning is finished and the reward has converged to a stable outcome, we select the *best* policy from the last 25 training runs to collect the metrics used for comparison. Each selected policy is then evaluated by averaging its outcome, defined as the sum of the rewards collected by each agent in a single episode, over ten episodes. The *best* policy is the one with the highest average total reward.

5.1 Preliminary Experiments

We conducted extensive initial experiments to gather the insights used to direct the study and to produce the baselines required for comparison purposes.

Naive solution We implemented an algorithm for naive trip planning based on people's common behavior [39]. Depending on the origin and destination, the estimated travel time for each of the (preferred) modes is computed using SUMO. The estimation is used to decide the departure time, taking into account a possible delay due to unforeseen congestion. The naive solution baseline is computed averaging over 100 simulations with a population of 1000 people, each with a random origin and the same destination, the venue. For each simulation, the preferred transportation mode distribution is selected randomly from a pool of six. One distribution presents balanced use of all the available modes, and the other distributions are skewed towards a preference for one specific mode. The actual weight associated with the preference is initialized each time, drawing from a Gaussian distribution. Additionally, each person has a personal buffer time estimation used to allow for unforeseen delays. This buffer time is randomly initialized and uniformly distributed between 5 and 15 minutes. For each simulation, we collect the same metrics we use in training and evaluation: travel time, waiting time, and lateness. These metrics are then averaged and used in all the relevant figures as the naive baseline for comparison.

DRL Baselines We initially evaluated the three reward models (Simple, OTT, and OTTCoord), with all of the actions (wait, walk, bicycle, public transport, car, and PTW) being available to every agent. Each of the above-mentioned start-time distributions ($30min_{\alpha=3}$, $45min_{\alpha=4}$, and $60min_{\alpha=5}$) are compared, and the resulting transportation mode distributions are shown in Figure 3. The Simple reward

[12]Results: https://github.com/lcodeca/results/Persuasive-training Last access: October, 2021

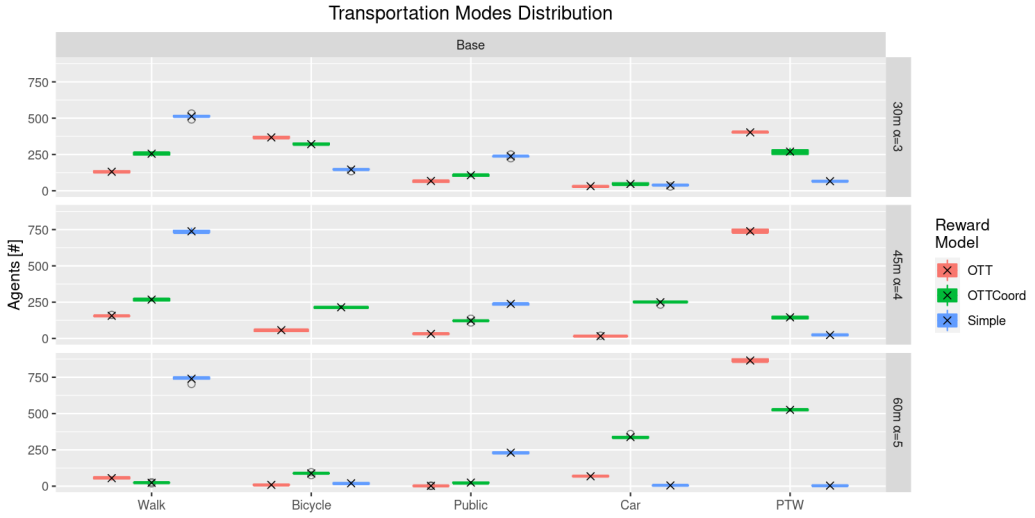


Figure 3: Mode Distribution for the base experiments.

model, in blue, does not take into account observed travel times, and the outcome is that it rewards the reliability provided by walking and using public transport (in a world practically devoid of other means of transportation generating congestion and delays). Once we introduce the observed travel time penalty (OTT, in red, and OTTCoord, in green), we can see that walking and public transport are not the most suitable choice anymore, and smaller and faster vehicles such as bicycles and PTW become more appealing. PTW are as fast as cars, and can filter through traffic congestion. Bicycles, although slower, have similar filtering capabilities in traffic, and being smaller, they seep through other vehicles more efficiently than PTW. Additionally, the start time distribution (thus, the competition among the agents) has a major impact on the mode choice. With $30min_{\alpha=3}$, where the competition is higher and the possibility of distributing departure over time is very limited, bicycles and PTW are the clear winners. In $60min_{\alpha=5}$, where the departure time can easily be spread, the clear winners are the faster modes: cars and PTW. Nonetheless, the presence of explicit coordination seems to favour the car and balance between the modes. The balanced mode usage is more explicit in $45min_{\alpha=4}$, where the mode choice is spread with explicit coordination (OTTCoord, in green), but the mode choice is completely skewed in favour of PTW where the only coordination is implicit and due to the limited resources (OTT, in red).

Background traffic For completeness, we run some experiments with the background mobility generated using SAGA and tuned to present a peak in demand around 8:30 AM, with the event starting at 9 AM. However, due to the complexity of the scenarios, we ran all other variations without background traffic in order to isolate the specific behaviors that we wanted to study. A selection of those run with background traffic is shown in Figure 4. Here we compare the OTTCoord model implementation used in the DRL Baseline, with (in blue) and without (in red) background traffic, to investigate if the results are significantly different. For these scenarios the start time distribution changed over the course of learning, starting with a 30-minute window and moving to the 60-minute one. The left plot in Figure 4 shows that the lateness and the travel time distributions with (in blue) and without (in red) traffic are comparable. The main difference is in the waiting time before the event. Due to the rush-hour traffic, the agents needed to depart in advance to arrive on time, resulting in increased waiting time before the event. The right plot in Figure 4 shows the transportation mode distribution. Although the trends are comparable, the few differences are intuitive. The public transportation capacity is more limited due to

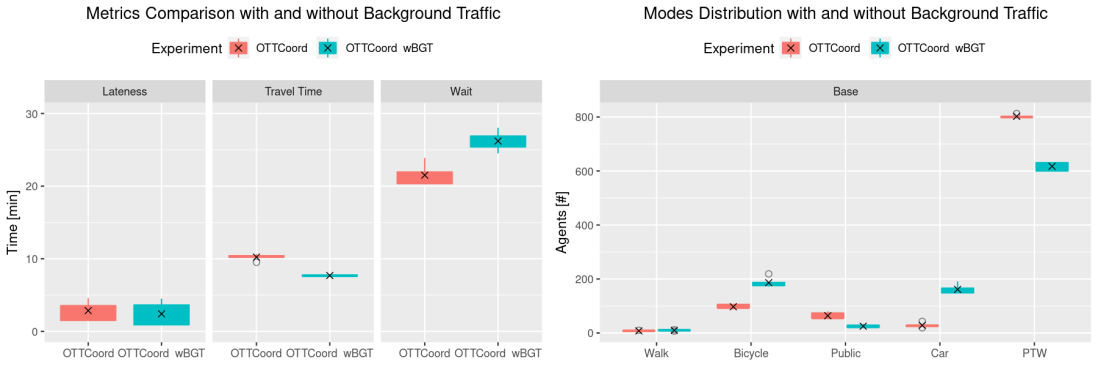


Figure 4: Impact of background traffic on the DRL Baseline experiment based on OTTCoord reward model.

Figure 5: Mode Distribution for the starting distribution of 60m $\alpha = 5$.

the background traffic, and the additional vehicles on the road are slowing the buses. PTW filtering is impacted by the additional congestion, making it less appealing and spreading the load to other modes such as cars and bicycles.

5.2 Ablation Study

To study the capabilities and limitations of the DRL approach, we performed an ablation study where starting from the insights provided by the DRL baselines, we systematically removed or changed modes of transport (actions) to assess whether the resulting policies matched our intuitions. Additionally, we evaluated the impact of uncertainty over the learned policy by forcing cars and PTW to find a parking spot, and we added hard constraints based on vehicle ownership. The impact of all the above-mentioned experiments on the transportation mode distributions is shown in Figure 5.

The following three experiments are focused on removing specific modes or changing the meaning of an action to verify that the resulting policy reflects the intended change. We evaluated all combinations of the reward models and the start time distributions. In the following paragraphs, we focus on the OTT and OTTCoord models, without background traffic, and the $60min_{\alpha=5}$ start time distribution.

carOnly The first experiment removed access to all vehicles except for cars, limiting the available actions to wait, walk and car. The *carOnly* column in Figure 5 shows the comparison between the transportation mode distributions with implicit (in red) and explicit (in blue) coordination. The expectation is to see a preference for the car (low travel time) with the departure time spread over time. This actually happens with the OTT reward model, but the presence of explicit coordination in OTTCoord balanced out the mode usage. This effect is due to the penalty paid by everyone if *other* agents are late, combined with the reliability of walking.

noPTW In the second experiment, we removed the PTW from the transportation modes by removing the associated action. In this case, the expected outcome is an increase in the usage of the car (for their speed) and bicycles (due to filtering capabilities). The *noPTW* column in Figure 5 shows the results. With the OTT reward model (in red), the expectation is met, with more agents choosing to walk (similarly to carOnly), a balanced spread between car and bicycles, and few public transportation options. However, the explicit coordination implemented in the OTTCoord reward model (in blue) again emphasised the travel time and reliability requirements as seen in the previous experiment, reducing bicycles and public transportation to a negligible use.

metroOnly In the third experiment, we changed the public transportation composition, removing the buses and leaving only the metro. Given that the buses are sharing the road with the rest of the vehicles, removing them from the public transportation infrastructure leaves only the metro, improving its reliability. In this experiment, all the transportation modes are available to the agents. The expectation is that the additional reliability introduced to public transportation would make it a more favourable option. The resulting transportation mode distribution is shown in the *metroOnly* column of Figure 5. In this case, the expectations are not met. Leaving the option of using the PTW, in conjunction with the increased road capacity, skewed the mode selection in favour of the PTW once more. Although in the presence of explicit coordination (in blue), the rest of the modes are comparatively more used than in OTT (in red), public transportation is still not a favourable option. As future work, we intend to investigate additional changes aimed at increasing public transports reliability and usage.

5.3 Increasing Uncertainty

In reality, private vehicles need to be parked, introducing delays and decreasing the effectiveness and reliability of the mode. For this experiment, we changed the meaning of the action associated with the selection of cars and PTW. Once the mode is chosen, the target destination changes from the event location to the parking area closest to the final destination, and the 'last mile' is walked. If the closest parking area is already full, the agent will cruise for parking (based on the SUMO parking area reroute mechanism[13]). Building upon the knowledge obtained from the previous experiment, we expected to see a decrease in the use of cars and PTW in favor of other more reliable modes. More precisely, we expect to see a shift from the car and PTW to bicycles due to their capability of filtering through congested vehicles and their reliability. The results of this experiment are shown in the *wParking* column of Figure 5. In this case, the expectation is met for both OTT (in red), and OTTCoord (in blue), and the impact of implicit or explicit coordination is negligible.

5.4 Constraints

In reality, not all people have access to every transportation mode. For this reason, we introduced hard constraints regarding the ownership of a vehicle modelled as the probability of each agent owning one.

[13] Wiki SUMO: Parking Area Rerouters definition https://sumo.dlr.de/docs/Simulation/Rerouter.html#rerouting_to_an_alternative_parking_area Last access: October, 2021

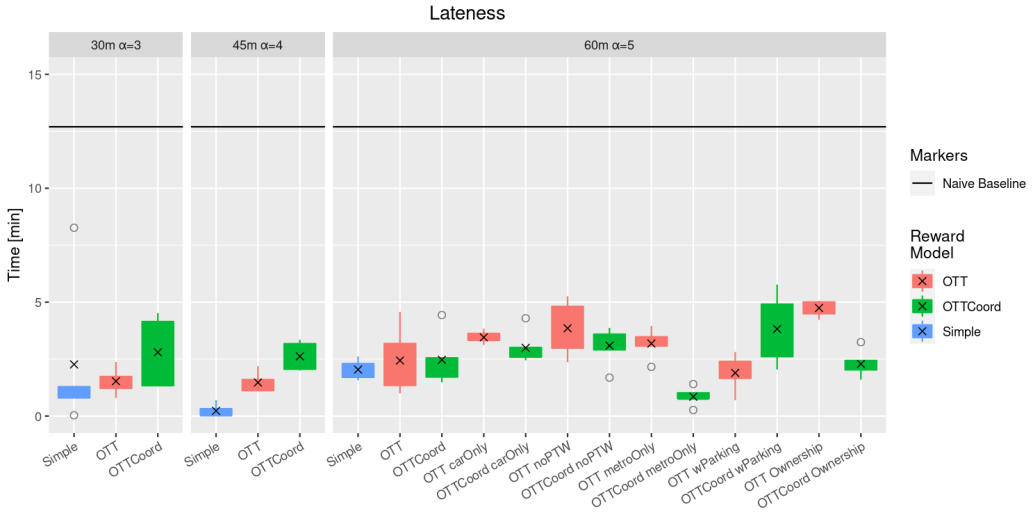


Figure 6: Lateness: late arrival distribution.

Following the National Household Travel Survey 2017 (reported on December 2018)[14] conducted by the Irish National Transport Authority (NTA), we configured the agent with the following probabilities: 89% of agents own a car, 35% own a bicycle, and only 10% own a PTW. In this experiment, all the actions are available to the agents, but they need to learn if they can use (own) a vehicle or not. With this experiment, we wanted to make sure that it was possible for the agents to learn a policy based on variable hard constraints. The expectation is that with the enforced limit over the PTW and the congestion generated by the presence of too many cars, the agents are able to learn how to distribute the load over the other modes. The comparison between the resulting mode distributions is shown in the *Ownership* column of Figure 5. In this experiment, the expectation is met with both reward models, and the hard constraints are respected. The cars are capped not by the defined ownership but by capacity constraints, and the other modes such as walking, bicycles and public transportation are used to spread the load.

5.5 Metrics

To measure the goodness of the stochastic policy the agents learnt over time, we evaluated the distribution of late arrivals, waiting time at destination, and observed travel time. Although departure time is not meaningful on its own due to its dependency on the resulting transportation mode distribution, it is useful to understand the other metrics.

Given the capacity constraints imposed by the problem at hand, finding the optimal set of trip plans able to maximize just-in-time arrival while minimizing late arrival may not be possible. Based on the reward models we implemented, the penalty for late arrival is higher than for waiting at the destination, increasing the chance that in case of unstable traffic congestion closer to the event's start time, the preferred option may incur higher waiting times.

Lateness The first metric we discuss is lateness, measuring the late arrival distribution of the agents. The late arrival is computed as the elapsed time between the event's start time and the agent's arrival

[14] National Household Travel Survey https://www.nationaltransport.ie/wp-content/uploads/2019/01/National_Household_Travel_Survey_2017_Report_-_December_2018.pdf Last access: October, 2021

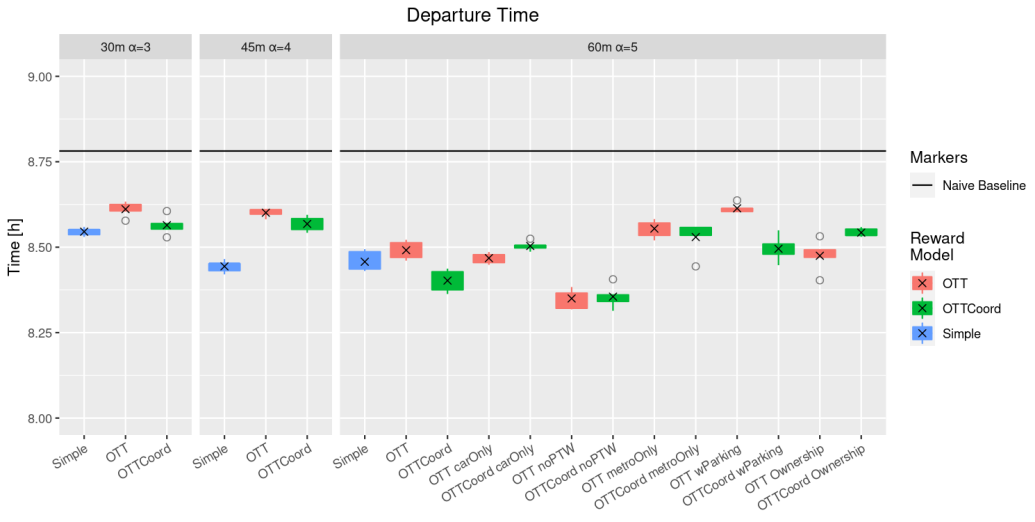


Figure 7: Departure time distribution.

at the destination. The box plots in Figure 6 show the late arrival distribution for all the experiments previously discussed. In the graph, the x in black shows the mean, and the empty black dots represent outliers. The black line is the naive solution's baseline (around 12 minutes), and it shows that every stochastic policy we learnt improves on it significantly. Nonetheless, there are no visible trends emerging from the difference between the OTT and OTTCoord reward models.

Departure The departure time metric on its own is not representative of the goodness of the policy. Nonetheless, it allows us to make some observations in conjunction with other metrics. Figure 7 depicts the departure time distributions for all the experiments discussed. The black line marks the naive solution baseline (around 8:45 AM). The figure shows that the agents learnt a more conservative strategy, expecting last-minute delays, hence leaving earlier. This is an additional reason why the lateness metrics are lower than in the naive baseline.

Waiting The higher penalty for the late arrival has a direct impact on the distribution of the waiting time at the destination. The waiting time is measured from the time of arrival at the venue until the event starts. Figure 8 shows the box plots associated with the waiting times for all the experiments. The naive baseline is marked by the black dashed line (about 7 minutes), and it shows that all our solutions require more waiting at the destination. Nonetheless, the solid black line represents the 15-minutes mark that we use to establish if the agents achieved their goal. Here we can see that all the DRL Baselines experiments performed for the initial start time distributions of $30m_{\alpha=3}$ and $45m_{\alpha=4}$ achieved the goal; for $60m_{\alpha=5}$, experiments such as Simple, OTTCoord carOnly, OTT wParking, and OTTCoord wOwnership achieved it too. Although the presence of coordination does not provide a clear trend in the results, it interesting to notice how its presence drastically decreases the waiting times in the carOnly experiment, while the departure time distributions are not significantly different. The same can be said for the noPTW and wOwnership experiments, and to a lesser extent, to metroOnly too.

Travel Time The final metric we consider is the observed travel time, measured between the departure time from the origin and the arrival time at the destination. Figure 9 shows the travel time distributions for each experiment. The solid black line marks the naive solution baseline (around 18 minutes). Apart from the Simple reward model (that does not take into consideration the travel time), in all the other experiments, we see an improvement. Similarly to the departure time, the actual travel

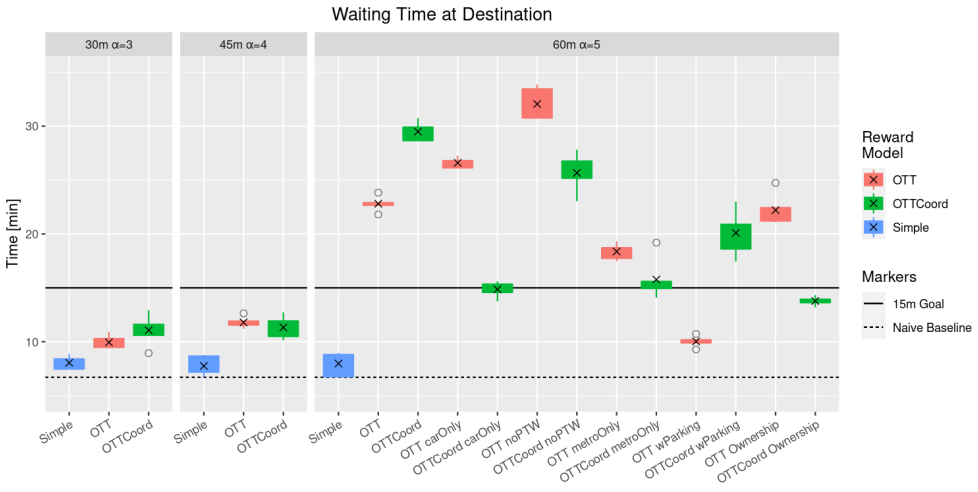


Figure 8: Waiting time at destination distribution.

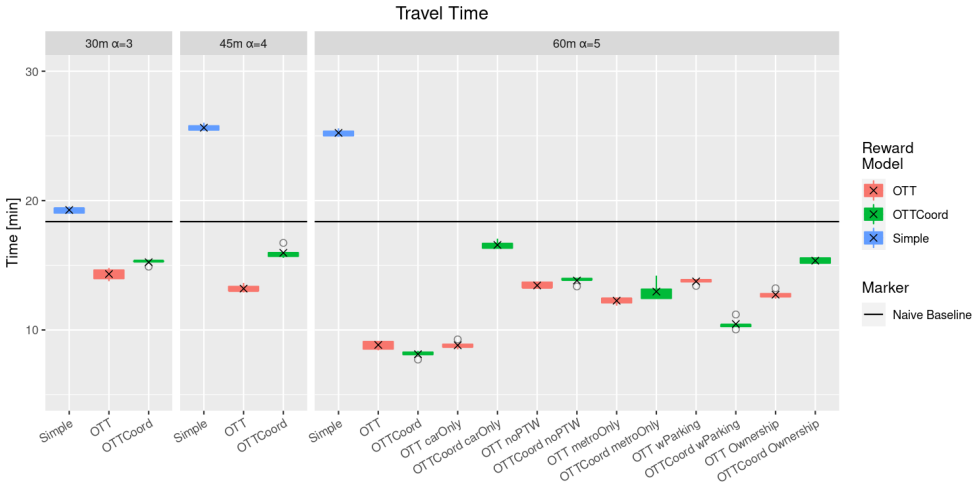


Figure 9: Travel Time distribution.

time is not a significant metric on its own due to its strong correlation with both the transportation mode distribution and the congestion levels. Like the previously discussed metrics, it is not possible to find a significant trend concerning the impact of coordination, leaving it on an experiment by experiment basis.

In summary, with the extensive evaluation presented in this section, we show that it is possible to use DRL to tackle the problem of coordinated multi-modal trip planning with limited transportation capacity. Through the exploration of a simulated transportation environment, it is possible to discover its features and learn viable plans. In all the variations discussed, the stochastic policy learnt is able to reduce late arrivals and travel times while keeping reasonable waiting times at destination for most of them. Finally, it is possible to learn a policy that takes into account hard constraints while providing reasonable trip plans.

6 Conclusions and Future Work

In this paper, we presented the implementation and evaluation of a MADRL framework to study the problem of coordinated multi-modal trip planning with constrained transportation capacity using a microscopic mobility simulator. The proposed solution leverages the distributed orchestration provided by RAY, the state-of-the-art ML algorithms implemented by RLlib, and RLlib SUMO Utils, a general-purpose connector to SUMO that enables the implementation of OpenAI Gym-like learning environments. All the software and infrastructure previously discussed is open-source and available on GitHub.

The use case presented is a large-scale social event, where the event-goers are assumed to have at their disposal an application provided by the venue that generates coordinated trip plans for just-in-time arrival at the event based on user requirements (if any). The DRL algorithm that we configured is the state-of-the-art implementation of PPO provided by Reinforcement Learning library (RLlib). Enabled by the general-purposes APIs definition, the learning environment we implemented for the case study is independent of the learning algorithm.

In this paper, we evaluated multiple learning environments with a combination of reward models (tied to travel time and coordination), availability of transportation modes (hard constraints such as ownership of a vehicle), and environmental factors (such as parking requirements).

The results showed that it is possible to use Deep Reinforcement Learning (DRL) and microscopic mobility simulations to study the problem of coordinated multi-modal trip planning with constrained transportation capacity. The stochastic policy learnt is able to reduce the late arrival and minimize the travel time. Additionally, the trip plans discovered by trial and error through the exploration of an unknown transportation environment are intuitive and easy to motivate, increasing the trust in DRL with microscopic mobility simulations as a viable option.

As future work, we intend to study the impact of soft constraints (i.e., user preferences) on coordinated trip plans. With the need to take into account user preferences concerning the transportation mode to use, the solution space increases drastically, and based on the distribution of the preferences, a solution may not exist. Finally, we intend to scale up the training from the 1000 agents in a controlled simulated environment to real numbers of event-goers in a realistic mobility scenario. Initial studies have been started using the Monaco SUMO Traffic (MoST) Scenario [14] and the Monaco stadium as the target venue.

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 713567. ENABLE is funded under Science Foundation Ireland (16/SP/3804) and is co-funded under the European Regional Development Fund.

References

[1] Zhongyu Wang, Hang Yang, and Bing Wu. Traffic flow characteristics and congestion evolution rules for urban road networks during special events. In CICTP 2017: Transportation Reform and Change—Equity, Inclusiveness, Sharing, and Innovation, pages 2368–2380. American Society of Civil Engineers Reston, VA, 2018.

[2] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. MIT press, 2018.

[3] Hao Dong, Hao Dong, Zihan Ding, Shanghang Zhang, and Chang. Deep Reinforcement Learning. Springer, 2020.

[4] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.

[5] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. Rllib: Abstractions for distributed reinforcement learning. In International Conference on Machine Learning, pages 3053–3062. PMLR, 2018.

[6] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. PloS one, 12(4):e0172395, 2017.

[7] Alejandro Torreno, Eva Onaindia, Antonín Komenda, and Michal Štolba. Cooperative multi-agent planning: a survey. ACM Computing Surveys (CSUR), 50(6):1–32, 2017.

[8] Pablo Alvarez Lopez and Michael Behrisch and Laura Bieker-Walz and Jakob Erdmann and Yun-Pang Flötteröd and Robert Hilbrich and Leonhard Lücken and Johannes Rummel and Peter Wagner and Evamarie Wießner. Microscopic Traffic Simulation using SUMO. In The 21st IEEE International Conference on Intelligent Transportation Systems. IEEE, 2018.

[9] Hoang Nguyen, Le-Minh Kieu, Tao Wen, and Chen Cai. Deep learning methods in transportation domain: a review. IET Intelligent Transport Systems, 12(9):998–1004, 2018.

[10] Zahra Karami and Rasha Kashef. Smart transportation planning: Data, models, and algorithms. Transportation Engineering, 2:100013, 2020.

[11] Mordechai Haklay and Patrick Weber. OpenStreetMap: User-generated street maps. Pervasive Computing, IEEE, 2008.

[12] Sandesh Uppoor and Marco Fiore. Large-scale urban vehicular mobility for networking research. In Vehicular Networking Conference (VNC), 2011 IEEE, pages 62–69. IEEE, 2011.

[13] Luca Bedogni, Marco Gramaglia, Andrea Vesco, Marco Fiore, Jerome Harri, and Francesco Ferrero. The Bologna Ringway dataset: improving road network conversion in SUMO and validating urban mobility via navigation services. Vehicular Technology, IEEE Transactions on, 64(12):5464–5476, 2015.

[14] Lara Codeca and Jérôme Härri. Monaco SUMO Traffic (MoST) Scenario: A 3D Mobility Scenario for Cooperative ITS. In SUMO 2018, SUMO User Conference, Simulating Autonomous and Intermodal Transport Systems, Berlin, GERMANY, 05 2018.

[15] Marco Rapelli, Claudio Casetti, and Giandomenico Gagliardi. Tust: from raw data to vehicular traffic simulation in turin. In 2019 IEEE/ACM 23rd International Symposium on Distributed Simulation and Real Time Applications (DS-RT), pages 1–8. IEEE, 2019.

[16] Maxime Gueriau and Ivana Dusparic. Quantifying the impact of connected and autonomous vehicles on traffic efficiency and safety in mixed traffic. In 23rd IEEE International Conference on Intelligent Transportation Systems, 2020.

[17] Abubakr O. Al-Abbasi, Arnob Ghosh, and Vaneet Aggarwal. Deeppool: Distributed model-free algorithm for ride-sharing using deep reinforcement learning. IEEE Transactions on Intelligent Transportation Systems, 20(12):4714–4727, 2019.

[18] Kaushik Manchella, Abhishek K. Umrawal, and Vaneet Aggarwal. Flexpool: A distributed model-free deep reinforcement learning algorithm for joint passengers and goods transportation. IEEE Transactions on Intelligent Transportation Systems, 22(4):2035–2047, 2021.

[19] Kaixiang Lin, Renyu Zhao, Zhe Xu, and Jiayu Zhou. Efficient large-scale fleet management via multi-agent deep reinforcement learning. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 1774–1783, 2018.

[20] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A. Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. IEEE Transactions on Intelligent Transportation Systems, pages 1–18, 2021.

[21] Ammar Haydari and Yasin Yilmaz. Deep reinforcement learning for intelligent transportation systems: A survey. IEEE Transactions on Intelligent Transportation Systems, pages 1–22, 2020.

[22] Jing Xin, Huan Zhao, Ding Liu, and Minqi Li. Application of deep reinforcement learning in mobile robot path planning. In 2017 Chinese Automation Congress (CAC), pages 7112–7116, 2017.

[23] Ursula Challita, Walid Saad, and Christian Bettstetter. Deep reinforcement learning for interference-aware

path planning of cellular-connected uavs. In 2018 IEEE International Conference on Communications (ICC), pages 1–7, 2018.

[24] Siyu Guo, Xiuguo Zhang, Yisong Zheng, and Yiquan Du. An autonomous path planning model for unmanned ships based on deep reinforcement learning. Sensors, 20(2):426, 2020.

[25] Hao Liu, Ting Li, Renjun Hu, Yanjie Fu, Jingjing Gu, and Hui Xiong. Joint representation learning for multi-modal transportation recommendation. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 1036–1043, 2019.

[26] Mariagrazia Dotoli, Hayfa Zgaya, Carmine Russo, and Slim Hammadi. A multi-agent advanced traveler information system for optimal trip planning in a co-modal framework. IEEE Transactions on Intelligent Transportation Systems, 18(9):2397–2412, 2017.

[27] Yang Liu, Cheng Lyu, Zhiyuan Liu, and Jinde Cao. Exploring a large-scale multi-modal transportation recommendation system. Transportation Research Part C: Emerging Technologies, 126:103070, 2021.

[28] Michael Redmond, Ann Melissa Campbell, and Jan Fabian Ehmke. Data-driven planning of reliable itineraries in multi-modal transit networks. Public Transport, 12(1):171–205, 2020.

[29] Zhiguang Cao, Hongliang Guo, Wen Song, Kaizhou Gao, Zhenghua Chen, Le Zhang, and Xuexi Zhang. Using reinforcement learning to minimize the probability of delay occurrence in transportation. IEEE Transactions on Vehicular Technology, 69(3):2424–2436, 2020.

[30] Thomas Liebig, Sebastian Peter, Maciej Grzenda, and Konstanty Junosza-Szaniawski. Dynamic transfer patterns for fast multi-modal route planning. In The Annual International Conference on Geographic Information Science, pages 223–236. Springer, 2017.

[31] Ayat Abedalla, Ali Fadel, Ibraheem Tuffaha, Hani Al-Omari, Mohammad Omari, Malak Abdullah, and Mahmoud Al-Ayyoub. Mtrecs-dlt: Multi-modal transport recommender system using deep learning and tree models. In 2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS), pages 274–278, 2019.

[32] Daniel Herzog, Hesham Massoud, and Wolfgang Wörndl. Routeme: A mobile recommender system for personalized, multi-modal route planning. In Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization, pages 67–75, 2017.

[33] Filip Dvorak, Shiwali Mohan, Victoria Bellotti, and Matthew Klenk. Collaborative optimization and planning for transportation energy reduction. In ICAPS Proceedings of the 6th Workshop on Distributed and Multi-Agent Planning (DMAP, 2018.

[34] Tanzina Afrin and Nita Yodo. A survey of road traffic congestion measures towards a sustainable and resilient transportation system. Sustainability, 12(11):4660, 2020.

[35] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I Jordan, et al. Ray: A distributed framework for emerging AI applications. In 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18), pages 561–577, 2018.

[36] Erdmann, Jakob. Lane-changing model in SUMO. Proceedings of the SUMO User Conference 2014: Modeling mobility with open data, 2014.

[37] Lara CODECA, Jakob ERDMANN, Vinny CAHILL, and Jérôme HARRI. Saga: An activity-based multi-modal mobility scenario generator for sumo. SUMO User Conference 2020 - From Traffic Flow to Mobility Modeling, 2020.

[38] Evgenii Nikishin, Pavel Izmailov, Ben Athiwaratkun, Dmitrii Podoprikhin, Timur Garipov, Pavel Shvechikov, Dmitry Vetrov, and Andrew Gordon Wilson. Improving stability in deep reinforcement learning with weight averaging. In Uncertainty in artificial intelligence workshop on uncertainty in Deep learning, 2018.

[39] Anne Durand, Lucas Harms, Sascha Hoogendoorn-Lanser, and Toon Zijlstra. Mobility-as-a-Service and changes in travel preferences and travel behaviour: a literature review. KiM— Netherlands Institute for Transport Policy Analysis, 2018.