Paper ID #889
Spatial Positioning Token (SPToken) for Smart Parking

Roman Overko[1]*, Rodrigo Ordóñez-Hurtado[2], Sergiy Zhuk[2], Robert Shorten[3]
1. University College Dublin, Dublin, Ireland. Email roman.overko@ucdconnect.ie
2. IBM Research, Damastown Industrial Park, Dublin 15, Ireland
3. Imperial College London, South Kensington, London SW7 2AZ, United Kingdom

Abstract
In this paper, we describe an approach to guide drivers searching for a parking space (PS). The proposed system suggests a sequence of routes that drivers should traverse in order to maximise the expected likelihood of finding a PS and minimise the travel distance. This system is built on our recent architecture SPToken, which combines both Distributed Ledger Technology (DLT) and Reinforcement Learning (RL) to realise a system for the estimation of an unknown distribution without disturbing the environment. For this, we use a number of virtual tokens that are passed from vehicle to vehicle to enable a massively parallelised RL system that estimates the best route for a given origin-destination (OD) pair, using crowdsourced information from participant vehicles. Additionally, a moving window with reward memory mechanism is included to better cope with non-stationary environments. Simulation results are given to illustrate the efficacy of our system.

Keywords:
Smart Parking, Reinforcement Learning

Introduction

A problem of immense importance to drivers is that of finding an available parking space (PS), which is also a societal problem affecting air pollution, congestion, and noise emission due to the large number of vehicles currently on the streets. For illustration, a case study of a small business district in Los Angeles [1] indicated that 730 tons of CO_2 where produced and 47,000 gal of gasoline were burned in a year by cars searching for parking. Another study by McKinsey[1] reported that the average car owners in Paris spend about four years of their lives searching for PSs. The emerging paradigm of intelligent transport systems (ITS) offers a great hope to alleviate these problems by enabling a range of parking guidance systems that are rapidly becoming an essential part of future sustainable cities. The most common approach are information boards displaying the available PSs at various locations around the city. This system provides a valuable guidance for drivers to avoid areas with potentially limited PS availability (PSA), but often leads to localised congestion around areas with the largest number of available PSs which is caused by (i) all the drivers receiving identical data, and (ii) the trend of most drivers to typically choose the areas with large PSA. More modern systems [2] provide extensive parking information directly to the drivers either to their smart devices or to the cars. The wealth of PSA information is provided to the drivers through the use of PS sensors. Such systems thus provide drivers with the exact location of currently free PSs along with e.g. their respective prices. The main disadvantage of such systems is that a large investment cost is required as sensors are expensive to deploy and maintain, thus limiting large-scale deployments. On the other hand, if moving vehicles are used to collect raw environment data (e.g. from dash cams or LiDAR sensors) to be turned into PSA information in cloud/MEC servers, then the cost and complexity of maintaining a central database of parking information built from raw environment data is also prohibitive from the perspective of the burden on the telecommunications network.

In the context of the EU, the European Commission has created a platform for the deployment of Cooperative ITS, namely the C-ITS Platform[2], with the aim of improving the efficiency of road transport at European-wide scale through a number of identified (essential) applications including *probe vehicle data* and *parking services*. In alignment with this, the smart parking solution proposed in this paper aims to use moving vehicles

[1] https://www.mckinsey.com/industries/public-and-social-sector/our-insights/the-smart-city-solution
[2] https://ec.europa.eu/transport/themes/its/c-its_en

as collectors and processors of raw environment data so that PSA can be shared as floating car data. For instance, if the ICT architecture proposed in [3] is integrated with the Spatial Potisioning Token (SPToken) framework [4, 5], then SPToken observers could include connected automated vehicles (CAVs), road-side units (RSUs), or MEC servers, and the floating car data can be transmitted via (perhaps extended) Cooperative Awareness Messages and Collective Perception Messages [6] (or a new message type to be developed).

This paper is organised as follows. First, we provide a brief summary of related work on applications in the area of smart parking. Second, we proceed to describe in detail the integration of SPToken into a smart parking solution. Third, we provide numerical validation of the proposed approach. Finally, we close the paper with some concluding remarks.

Related work

We proceed to present a short overview of relevant smart parking solutions. In [2], the authors propose a MEC-based architecture using VANETs to improve the parking experience concerning average parking cost, fuel wastes, and vehicle exhaust emissions, and the solution gives drivers the option to integrate their own parking preferences. However, this solution relies on parking information from interconnected MEC servers deployed at parking lots, thus substantially restricting its scope to such locations and not allowing for the integration of unsupervised on-street parking. Concerning privacy-preserving data management, several architectures for smart parking solutions have already been proposed. In [7], a blockchain-based descentralised parking management service is designed, which ensures anonymous authentication and resistance to data linkability, among others. In [8], the authors introduce a privacy-preserving smart parking navigation system based on the integration of cloud services and vehicular communication, that allows for identity/service authentication and traceability. Even though [7] and [8] provide means for parking data retrieval (including PSA and navigation guidance) or PS reservation, their allocation process relies on parking-related databases maintained with availability data provided by e.g. parking lot owners, and it does not involve any optimisation problem. The literature on applications of machine learning to smart parking is vast and we simply refer to a few relevant publications. In [9], the authors propose a Q-learning-based solution to find the nearest PS by minimising the total covered distance, time taken and consumed energy. This approach is evaluated only on stationary environments, which is restrictive in many real-world applications including traffic management problems. Situations with non-stationary models may lead to suboptimal policies provided by RL algorithms. In this context, only a few existing works have considered to improve RL algorithms for non-stationary environments. A recent example of such works includes [10], in which the authors have developed a model-free RL method to effectively detect changes in the rewards and transition dynamics, and validated the proposed approach on randomly generated data and traffic signal control problems as well.

SPToken for Smart Parking

In this work, we make use of the Distributed Ledger Technology (DLT) based SPToken architecture proposed in [4, 5] to design a solution for the on-street parking problem. As detailed in [4, 5], the SPToken framework allows to explore a given environment without perturbing it, this achieved through the use of (i) virtual entities referred to as *tokens* in combination with (ii) crowdsourcing approaches such as multi-agent RL, which makes SPToken suitable for a wide range of smart mobility applications including smart parking. The key idea is to use the tokens as virtual containers to be filled with floating car data, and allow tokens to “jump” from vehicle to vehicle whenever is required to complete routes determined by the underlying RL algorithm. A participating vehicle can “collect” a free token via a DLT transaction if its route coincides with the token route, after which the vehicle with the token turns into an RL agent that updates the distributed ledger with some information (e.g. spent travel time, measured roadside parking availability) whenever it passes an *observer*. The vehicle must “deposit” the token via another DLT transaction when it deviates from the token route. Since tokens are transported by participating vehicles (not forced to follow the entire token route) already present in an urban scenario, SPToken guarantees that the environment stays undisturbed during the probing process. Note that the physical presence of vehicles is imperative to collect/deposit tokens, and so a *Proof-of-Position* mechanism is also included (see [5] for details). SPToken also has all the advantages of DLT such as data privacy preservation, data ownership retention, and misuse/spamming prevention. As a base requirement for participating vehicles, we assume they can provide on-street PSA data along their routes as a result of applying analytics to (raw) real-time environment data collected by their on-board sensors (e.g. point clouds from LiDAR sensors). We assume this requirement is reasonably satisfied based on reported mechanisms for on-board detection of parking availability [11, 12].

Note that the proposed solution is not a straightforward application of the SPToken architecture to smart parking, and a number of extensions (also to the underlying RL algorithm [4]) were required: (i) a new mechanism called *moving window with reward memory* (MWRM) for MUBEV [5] to improve the performance in non-stationary environments; (ii) a more advanced action selection strategy to increase exploration at early stages of the learning process; (iii) an improved design of the reward function for the RL algorithm to account for a reduced number of tuning parameters; and (iv) a design parameter α is introduced as a means for users to provide their preferences in terms of the their preferred optimisation objectives. Additionally, the application of SPToken to the on-street parking problem required dealing with a non-trivial multi-objective optimisation problem: our solution involves the investigation of optimal routes connecting origin-destination (OD) pairs so that in such routes (a) travel distance is minimised, and (b) PSA is maximised. While (a) involves a (widely studied) shortest path (SP) problem, the solution for (b) requires solving a longest path problem of NP-hard nature. Therefore, we decided to integrate these two problems into a graph-based, multi-objective optimisation approach by linearly combining two directed weighted graphs: a travel Distance Graph (DG), and a Parking Availability Graph (PAG). Both graphs come from the road network, where each vertex is a road junction, each edge is a collection of roads, weights Λ in DG represent the physical (time invariant) length of given roads, and weights K in PAG are in function of the (time-variant) number of unoccupied on-street PSs along given roads at certain time. As a result, the graph weights of the convex linear combination $G = (\alpha)\Lambda + (1-\alpha)K$ are computed as

$$\mathcal{W}_G(\alpha, s, \lambda_s, \nu_s(t)) = (\alpha) * \lambda_s + (1-\alpha) * \kappa_s\left(\nu_s(t)\right), \tag{1}$$

where: α is a user-defined parameter that satisfies $\alpha_{min} \leq \alpha \leq 1$ and represents the priority given to the distance cost; λ_s is the length of edge s in G; $\kappa_s(\nu_s(t)) = -\nu_s(t) * L_{max}/C_{max}$, with $\nu_s(t)$ being the current roadside PSA along the road links included in edge s at time t, L_{max} the maximum edge length in the road network, and C_{max} the maximum edge parking capacity in the road network. While SP routing is widely used as the *de-facto* default solution (DS) in most navigation systems, SP does not always guarantees maximum PSA. However, the total PS capacity along road links is indeed time-invariant, and thus in our approach we compute DSs using Equation 1 with the value of α provided by the users. In addition, α_{min} is determined empirically such that the graph G does not contain negative cycles for weights computed as $\mathcal{W}_G(\alpha, s, \lambda_s, c_s)$, where c_s is the total (time-invariant) roadside parking capacity along the road segments included in edge s. Consequently, if $\alpha \in [\alpha_{min}, 1]$, then we can use an SP algorithm to compute DSs to be used as the initial policy of the RL algorithm. Clearly, DSs degenerate into SPs when $\alpha = 1$ (as per Equation 1).

Optimal policy search algorithm

As previously mentioned, we use an RL strategy to solve our target problem. A full RL problem is usually modeled as a Markov Decision Process (MDP). Our decision problem is a stationary Finite Horizon MDP (FHMDP), which is a discrete-time stochastic control process defined by a tuple $\langle \mathcal{S}, \mathcal{A}, P, \mathcal{R}, H \rangle$, where: $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, P is the tensor of transition probabilities, $\mathcal{R}$ is the reward matrix, and H is the length of the time horizon. Let $S = |\mathcal{S}|$, $A = |\mathcal{A}|$ and $A_s = |\mathcal{A}_s|$, that is, S is the number of states, A is the total number of actions, and A_s is the number of actions allowable in state $s \in \mathcal{S}$. We assume that all the states are fully observable. We also adopt this general approach: each state s has a different set of allowable actions $\mathcal{A}_s$, hence $\mathcal{A} = \bigcup_{s \in \mathcal{S}} \mathcal{A}_s$. Finally, $P(s'|s, a)$ is the probability of transition to state s' if action a is chosen in state s, and $\mathcal{R}(s,a)$ is the reward of playing action a in state s.

An RL agent, i.e. the decision maker, interacts with the environment in a sequence of episodes, each episode with length H. Within an episode k at each time step $t \in [H]$ (where $[\bullet] = \{1, 2, ..., \bullet\}$ with $\bullet$ an integer number), the agent plays action a based on the observation of state s and receives a reward $r(s,a)$. The next state s' is drawn from a distribution P which defines the trajectory of the MDP, that is $s' \sim P(\cdot|s,a)$. Over each episode k, the agent selects actions according to a policy π_k, which maps states and time steps to actions. π_k is updated after H interactions with the environment, i.e. at the end of each episode. Note that in stationary MDPs, the transition probabilities and reward distribution do not vary with time step t. The policy π_k, however, is generally time-step-dependent for FHMDPs. The expected return until the end of an episode is represented by a *value function* (for state s, time step t, and policy π_k) defined as

$$V_t^{(\pi_k)}(s_t) \coloneqq \mathbb{E}\left[\sum\nolimits_{i=t}^{H} \mathcal{R}\big(s_i, \pi_k(s_i, i)\big)\right], \tag{2}$$

where the expectation is taken with respect to states s' encountered in the MDP. The quality of a policy at episode k is characterised by the *total expected reward* defined as $u^{(\pi_k)} = p_0^\top V_1^{(\pi_k)}$, where p_0 is the distribution of the initial states. The value function represented in Equation 2 can be rewritten as follows:

$$V_t^{(\pi_k)}(s_t) = \mathcal{R}\big(s_t, \pi_k(s_t, t)\big) + P\big(\cdot\,|s_t, \pi_k(s_t, t)\big)^\top V_{t+1}^{(\pi_k)}. \tag{3}$$

Equation 3 is called the *Bellman equation* or the optimality equation, which is often solved using the *backward induction* process with boundary condition $V_{H+1}^{(\pi_k)} \coloneqq 0$. The goal of an RL agent is then to find an optimal trajectory which maximises the total expected reward. Thus, the optimal policy π^* is calculated through the backward induction procedure as follows:

$$\pi^*(s,t) = \operatorname*{argmax}_{a \in \mathcal{A}_s} \left\{\mathcal{R}(s,a) + P\big(\cdot\,|s,a\big)^\top V_{t+1}^{\pi^*}\right\}, \;\; \pi^*(s,H) = \operatorname*{argmax}_{a \in \mathcal{A}_s} \mathcal{R}(s,a). \tag{4}$$

We are now in a position to present the **M**odified **UBEV** for **S**tationary FHMDPs with **M**oving window (MUBEV-SM) algorithm, which is the tool we use in our approach. Algorithm 1 represents a new variant of the MUBEV algorithm introduced in [13] (see [4] for the most recent version of MUBEV), which also includes the key modifications of the UBEV-S algorithm presented in [14] in order to achieve a better performance on stationary MDPs. Namely, UBEV-S incorporates a slightly modified exploration bonus and produces tighter regret bounds in the settings of time-independent rewards and transition dynamics [14].

It is worth noting that MUBEV-SM includes all the modifications introduced in the MUBEV algorithm [5, 13]. That is, (i) MUBEV-SM does not require the estimation of transition probabilities due to the specific design of the state-action space; (ii) in MUBEV-SM, observation of the environment is achieved using multi-agent approach; and (iii) MUBEV-SM incorporates a better action selection strategy. However, we introduce a few new modifications to the MUBEV-SM algorithm as explained below.

First, the action selection strategy is slightly improved now. Specifically, MUBEV is forced to use the default policy (e.g. SP policy) whenever all the entries of Q_s are equal to each other, where Q_s denotes a vector of Q-function values for state s. In MUBEV-SM, we explicitly compute and use the maximum value of Q_s, i.e. Q_{max} (Algorithm 1, line 14). When Q_{max} appears multiple times in Q_s, the selection strategy is as follows: if one of the repetitions corresponds to the default action, then action is chosen according to the default policy; otherwise, an entry is randomly selected from those repetitions using uniform distribution, and the associated action is chosen (Algorithm 1, line 16). Such a randomness boosts the exploration at the beginning of the learning process.

Second, MUBEV-SM involves a new mechanism not used in [4, 5, 13], referred to as MWRM, which is introduced as a mitigation method against stagnation that may occur when a change in the environment happens after a long period of stationarity, such as a long sequence of episodes with similar traffic conditions. For the calculation of the estimated reward $\hat{r}$ in UBEV and UBEV-S [14, 15], all reward values from the first episode are accumulated, while in MUBEV-SM this only happens in the context of a moving window. That is, in MUBEV-SM, a maximum of past J rewards are accumulated (with J the length of the moving window), and the oldest reward is dropped every time a new reward is collected. Additionally, the accumulation is done using weighted rewards, and all the collected statistics are reset once the confidence bound reaches a certain minimum value. As a result, MUBEV-SM can better cope with environments whose reward distribution dynamically changes or includes long periods of stationarity[3]. However, the MWRM mechanism can be disabled if required ($MWRM$ = False in Algorithm 1).

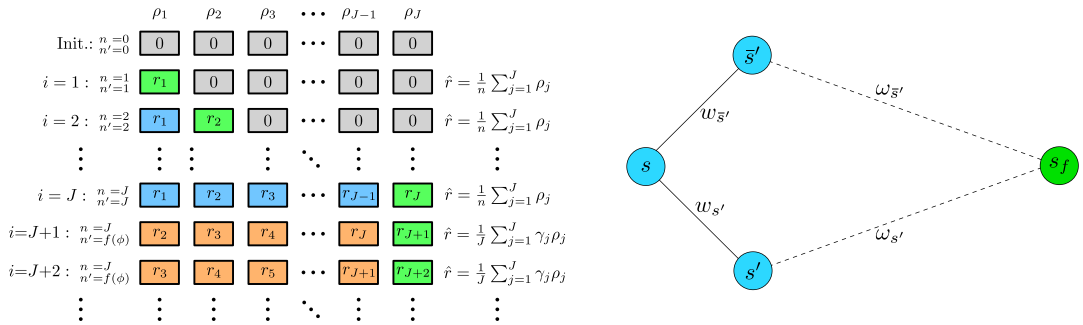
ρ_1 ρ_2 ρ_3 $\cdots$ ρ_{J-1} ρ_J
Init.: $n=0$, $n'=0$
$i=1$: $n=1$, $n'=1$
$i=2$: $n=2$, $n'=2$
$i=J$: $n=J$, $n'=J$
$i=J+1$: $n=J$, $n'=f(\phi)$
$i=J+2$: $n=J$, $n'=f(\phi)$
0
r_1 r_2 r_3 r_4 r_5 r_{J-1} r_J r_{J+1} r_{J+2}
$\hat{r}=\frac{1}{n}\sum_{j=1}^{J}\rho_j$
$\hat{r}=\frac{1}{J}\sum_{j=1}^{J}\gamma_j\rho_j$
$\bar{s}'$
s
s'
s_f
$w_{\bar{s}'}$
$w_{s'}$
$\omega_{\bar{s}'}$
$\omega_{s'}$

Figure 1 - Left: The MWRM mechanism used in MUBEV-SM. Gray cells represent initial values of ρ. A green cell represents the current observed reward. Rewards observed at previous interactions are stored sequentially (blue cells). Values of ρ in orange and green cells (if moving window is reached) are subject to discounts. Right: Schematic representation of the state weight w and the route weight ω used for computation of rewards. $w_{s'}$ and $w_{\bar{s}'}$ are the state weights for nodes s' and $\bar{s}'$, respectively. $\omega_{s'}$ and $\omega_{\bar{s}'}$ are the route weights to the destination node s_f from nodes s' and $\bar{s}'$, respectively.

For a better understanding of the MWRM mechanism, see Figure 1-Left in which we illustrate this technique for a given state-action pair (s, a) observed at the i-th interaction with the environment (without loss of generality). Recall that J is the length of the moving window. For each (s, a) pair, we introduce an additional transition count n', and a *reward memory vector* ρ of length J to store the collected rewards. Initial values of n, n', and ρ_j (gray cells) are set to zero for all (s, a) pairs and $\forall j \in [J]$, where n and n' represent the number of times action a is played in state s. Note that both n and n' are equal to each other and incremented by 1 only if $i < J$. At interaction i, reward r_i is observed and ρ is updated such that $\rho_i = r_i$ (green cells) if $i \leq J$, i.e. within the moving window, and the estimated reward $\hat{r}$ is the mean of the collected rewards. When the moving window length is reached, i.e. $n = n' = J$, we stop incrementing n. Thus, $n(s, a) \leq J$ for all state-action pairs (s, a). The estimated reward $\hat{r}$ at interaction $i = J$ is the mean of ρ. For further interactions $i > J$, the count n' is determined as a function of the confidence bound ϕ as follows: $n'(s, a)$ is reset to 1 for all $a \in \mathcal{A}_s$ whenever the confidence bound for state-action pair (s, a) decreases below some predefined threshold ϕ_{min} (which ensures a faster return to the default policy). Additionally, at interactions $i > J$, the oldest value of the collected reward is dropped, all other entries of ρ are shifted one position to the left, and the new collected reward (green cell) is inserted at the J-th position. Then, $\hat{r}$ is computed as the weighted mean of ρ using the corresponding weighting coefficients γ_i. We refer to γ as the vector of "forgetting" factors (or *discount vector*), and it is a tuning parameter along with ϕ_{min}.

The notation for MUBEV-SM (Algorithm 1) is as follows. $\mathcal{S}$ is the set of states; $\mathcal{A}_s$ is the set of allowable actions in state s, and thus, $\mathcal{A} = \bigcup_{s\in\mathcal{S}} \mathcal{A}_s$ is the total action set; S, A_s and A denote cardinality of finite sets $\mathcal{S}$, A_s and $\mathcal{A}$, respectively; P is a 3-dimensional tensor of predefined transition probabilities; Π_{DP} is the

[3] Note that reward is independent of time step t in stationary MDPs, although it can vary from one episode to another.

Algorithm 1 **M**odified **U**pper **B**ounding the **E**xpected Next State **V**alue (**UBEV**) Algorithm for **S**tationary FHMDPs with **M**oving Window—**MUBEV-SM**

```
Input: $\mathcal{S}$, $\mathcal{A}$, $P$, $\Pi_{DP}$; $\varepsilon, \delta \in (0,1]$; $H$, $J$, $M \in \mathbb{N}$; $r_{max}, \phi_{min} \in \mathbb{R}$; $\gamma \in \mathbb{R}^J$; $MWRM \in$ {True, False}.
$n(s,a) = n'(s,a) = \hat{r}(s,a) = R(s,a) = Q(s,a) = \phi(s,a) = \hat{V}(s,t) = \rho(s,a,j) = 0 \ \forall s \in \mathcal{S},\ a \in \mathcal{A}_s,\ t \in [H+1],\ j \in [J]$.
$\delta' = \delta/9$, $V_{max} = H * r_{max}$, $\phi_+ = 0$, $\pi = \Pi_{DP}$.
for $k = 1, 2, 3...$ do
    for $t = H$ to $1$ do                                                  ▷ Optimistic planning loop
        $\hat{V}_{t+1} = \hat{V}(\cdot, t+1);\ \tilde{V}_{max} = \min\big(\max(\hat{V}_{t+1}), V_{max}\big)$
        for $s \in \mathcal{S}$ do
            $\eta = \ln\big(27 * S * A_s * H/\delta'\big)$
            for $a \in \mathcal{A}_s$ do
                $r = r_{max};\ EV = \tilde{V}_{max}$
                if $n'(s,a) > 0$ then
                    $\phi(s,a) = \varepsilon * \sqrt{\frac{2\ln\ln\big(\max\big(e, n'(s,a)\big)\big) + \eta}{n'(s,a)}};\ \hat{V}_{next} = P(\cdot, s, a) \times \hat{V}_{t+1};\ \bar{v} = \max(\hat{V}_{t+1}) - \min(\hat{V}_{t+1})$
                    $\bar{v}' = \min\big((H-t) * r_{max}, \bar{v} + \phi_+\big);\ r = \min\big(r_{max}, \hat{r}(s,a) + \phi(s,a)\big);\ EV = \min\Big(\tilde{V}_{max}, \hat{V}_{next} + \bar{v}' * \phi(s,a)\Big)$
                $Q(s,a) = r + EV$
            $Q_{max} = \max_{a \in \mathcal{A}_s}\big(Q(s,a)\big)$
            if $Q_{max}$ duplicates in $Q(s,\cdot)$ then
                if $Q_{max} = Q\big(s, \Pi_{DP}(s,t)\big)$ then $\tilde{a} = \Pi_{DP}(s,t)$ else $\tilde{a} \sim \mathcal{U}\big(\{a \,|\, Q(s,a) = Q_{max}\}\big)$
            else $\tilde{a} = \mathrm{argmax}_{a \in \mathcal{A}_s} Q(s,a)$
            $\pi_k(s,t) = \tilde{a};\ \hat{V}(s,t) = Q(s,\tilde{a})$
            if $n'(s,\tilde{a}) > 0$ then $\tilde{\phi} = \varepsilon * \sqrt{\frac{2\ln\ln\big(\max\big(e, n'(s,\tilde{a})\big)\big) + \eta}{n'(s,\tilde{a})}};\ \phi_+ = \max\big(4 * \sqrt{S} * H^2 * \tilde{\phi}, \phi_+\big)$
    $\tilde{s} = \big[s_1^{(1)}, ..., s_1^{(M)}\big] \sim \mathcal{U}\big(1, S\big),\ s_1^{(i)} \neq s_1^{(j)} \ \forall i,j \in [1, M]$
    for $m = 1$ to $M$, $t = 1$ to $H$ do                                      ▷ Execute policy for one episode
        $a_t^{(m)} = \pi_k(s_t^{(m)}, t);\ \bar{a}_t^{(m)} = \Pi_{DP}(s_t^{(m)}, t);\ s_{t+1}^{(m)} \sim P(\cdot|s_t^{(m)}, a_t^{(m)});\ \bar{s}_{t+1}^{(m)} \sim P(\cdot|s_t^{(m)}, \bar{a}_t^{(m)})$
        $r_t^{(m)} = \mathcal{R}\big(s_t^{(m)}, s_{t+1}^{(m)}, \bar{s}_{t+1}^{(m)}\big);\ R\big(s_t^{(m)}, a_t^{(m)}\big) += r_t^{(m)}$      ▷ Call to Function 1, accumulate rewards
        if $MWRM =$ True then                                              ▷ Employ moving window
            $j_m := n\big(s_t^{(m)}, a_t^{(m)}\big);\ j'_m := n'\big(s_t^{(m)}, a_t^{(m)}\big)$
            if $j_m \leq J$ then
                $\rho\big(s_t^{(m)}, a_t^{(m)}, j_m\big) = r_t^{(m)};\ n\big(s_t^{(m)}, a_t^{(m)}\big)++;\ n'\big(s_t^{(m)}, a_t^{(m)}\big)++;\ \hat{r}\big(s_t^{(m)}, a_t^{(m)}\big) = R\big(s_t^{(m)}, a_t^{(m)}\big)/n\big(s_t^{(m)}, a_t^{(m)}\big)$
            else
                $R\big(s_t^{(m)}, a_t^{(m)}\big) = R\big(s_t^{(m)}, a_t^{(m)}\big) - \rho\big(s_t^{(m)}, a_t^{(m)}, 1\big)$                   ▷ “Forget” the oldest reward
                $\rho\big(s_t^{(m)}, a_t^{(m)}, \cdot\big) = \rho\big(s_t^{(m)}, a_t^{(m)}, \cdot\big)_{2:J} +\!\!+ \{r_t^{(m)}\}$                 ▷ Incorporate the newest reward
                $\hat{r}\big(s_t^{(m)}, a_t^{(m)}\big) = \Big(\gamma \times \rho\big(s_t^{(m)}, a_t^{(m)}\big)\Big)/J$
                if $\phi\big(s_t^{(m)}, a_t^{(m)}\big) \leq \phi_{min}$ then $n'\big(s_t^{(m)}, a\big) = 1 \ \forall a \in \mathcal{A}_s$ else $n'\big(s_t^{(m)}, a_t^{(m)}\big) ++$
        else                                                               ▷ Do not apply moving window
            $n\big(s_t^{(m)}, a_t^{(m)}\big) ++;\ n'\big(s_t^{(m)}, a_t^{(m)}\big) ++;\ \hat{r}\big(s_t^{(m)}, a_t^{(m)}\big) = R\big(s_t^{(m)}, a_t^{(m)}\big)/n\big(s_t^{(m)}, a_t^{(m)}\big)$
```

default path (DP) policy, computed using an SP algorithm as the initial estimate for π; ε is the exploration bonus, and δ is the failure probability (see [15] for details); H is the length of the MDP's time horizon; J is the size of moving window; M is the number of MUBEV-SM tokens; r_{max} is the maximum reward an agent can receive per transition; ϕ_{min} is the minimum value of the confidence bound which determines when to reset the collected statistics (i.e. counts n'); γ is a vector of discounts for the rewards stored in the reward memory ρ; $MWRM$ is a boolean flag to enable/disable MWRM; $n(s,a)$ and $n'(s,a)$ count the times action a is played in state s (see previous paragraph for details); $\hat{r}(s,a)$ and $R(s,a)$ are the normalised and accumulated rewards in state s under action a, respectively; $Q(s,a)$ is the Q-function [15] for state s and action a; $\phi(s,a)$ is the confidence bound for state s and action a; $\hat{V}(s,t)$ is the value function from time step t for state s; $\rho(s,a,j)$ represents a reward stored in a 3-dimensional tensor of reward memory ρ for state s and action a at memory epoch j; δ' represents a scaled failure tolerance (see [15] for details); V_{max} is the maximum value of the value function for next states; ϕ_+ is the correction term which keeps track of the largest confidence bound for the least visited state-action pair under the policy π [14]; $\hat{V}(\cdot, t+1)$ and $P(\cdot, s, a)$ denote vectors of length S, and $Q(s,\cdot)$ is interpreted as a vector of length A_s; e is the Euler's number; η, r, EV, $\bar{v}$ and

$\bar{v}'$ are auxiliary variables; $\bar{v}$ represents the range of vector $\hat{V}_{t+1}$ [14]; vector $\tilde{s}$ is a vector of initial states of MUBEV-SM tokens, which is uniformly sampled in range from 1 to S with no repeated entries; m is the index of an agent (a vehicle with a token) that interacts with the environment each time step t, and receives reward $r_t^{(m)}$ determined by the reward function defined in Function 1.

Reward function

The reward function (Function 1) returns the total reward r_t at time t, which implicitly incorporates two objectives, namely travel distance and parking availability.

Function 1 The Reward Function

```
Input: $\alpha \in [\alpha_{min}, 1]$ ; $s_t, s_{t+1}, \bar{s}_{t+1} \in \mathcal{S}$; $\beta_1, \beta_2, r_{max} \in \mathbb{R}$.
Output: $r_t$.
 1: $s_t := s, s' := s_{t+1}, \bar{s}' := \bar{s}_{t+1}$
 2: function $\mathcal{R}(s, s', \bar{s}')$
 3:     Get the state lengths $\lambda_{s'}$ and $\lambda_{\bar{s}'}$, and the total roadside parking capacity $c_{\bar{s}'}$.
 4:     Observe the current roadside parking availability $\nu_{s'}$.
 5:     $w_{\bar{s}'} = \mathcal{W}_G(\alpha, \bar{s}', \lambda_{\bar{s}'}, c_{\bar{s}'}),\ w_{s'} = \mathcal{W}_G(\alpha, s', \lambda_{s'}, \nu_{s'})$    ▷ Compute weights for $\bar{s}'$, $s'$, respectively, using Equation 1
 6:     if $w_{\bar{s}'} \neq w_{s'}$ then    ▷ Uncertainty has been encountered
 7:         Get route total weights $\omega_{s'}$ and $\omega_{\bar{s}'}$. $\Omega_{s,\bar{s}'} = \omega_{\bar{s}'} + w_{\bar{s}'};\ \Omega_{s,s'} = \omega_{s'} + w_{s'};\ r_t = 1 - \frac{\Omega_{s,s'}}{\Omega_{s,\bar{s}'}}$
 8:         if $s' = \bar{s}'$ then $r_t = \beta_1 * r_t$    ▷ Agent takes the DP action
 9:         else $r_t = \beta_2 * r_t$    ▷ Agent takes an alternative action
10:     else $r_t = r_{max}$    ▷ There is no uncertainty
    return $r_t$
```

Function 1 receives three arguments: (i) s_t, the previous state of an agent; (ii) s_{t+1}, the current state; and (iii) $\bar{s}_{t+1}$, the state to which the agent would "jump" if the default action in state s_t were taken. To begin with, $\lambda_{s'}$ and $\lambda_{\bar{s}'}$ are the lengths of road links included in states s' and $\bar{s}'$, respectively, $c_{\bar{s}'}$ is the total on-street parking capacity for state $\bar{s}'$, and $\nu_{s'}$ is the current parking availability in state s'. Note that the state length λ can be easily computed for a given road network. Also, as we assume that total roadside PS capacity for the road network is available (e.g. via digital map provider), and thus the total state capacity c is known for each state. However, PSA is subject to real-time measurements (e.g. obtained via [11, 12]). $w_{\bar{s}'}$ is the state weight if the agent would transit from state s to state $\bar{s}'$ in which all the on-street parking stops are treated as available. In contrast, $w_{s'}$ represents the state weight of the actual transition from state s to state s' at time t. If uncertainties occur in the environment, then the weights computed in line 7 are not equal to each other, and thus the DP policy has to be amended. In this case, the reward is computed using the total route weight of two (in general different) routes. Further, $\Omega_{s,\bar{s}'}$ and $\Omega_{s,s'}$ are the total weights of routes from s to the destination state s_f via $\bar{s}'$ and s', respectively, and $\omega_{\bar{s}'}$ and $\omega_{s'}$ are the total weights of routes from $\bar{s}'$ and s' to the destination s_f, respectively (see Figure 1-Right). As a result, $\Omega_{s,\bar{s}'} = \omega_s$, and since $\Omega_{s,\bar{s}'} < \Omega_{s,s'}$, reward computed in line 7 always satisfies $r_t < 0$. Finally, β_1 and β_2 are used to speed up the learning process of searching for a detour and returning to the DP route respectively, and their values may depend on many factors including values of α and r_{max} or the road network itself. To achieve a better and faster performance of the algorithm, β_1 and β_2 need to be increasing functions of α as illustrated in the following section. Finally, if a given state is not affected by uncertainties, the agents receive the maximum reward r_{max} available per transition.

Numerical evaluation

In this section we are interested in evaluating a route recommender system for smart parking, for which a number of MUBEV-SM tokens is distributed to emulate real vehicles probing the uncertain environment. These tokens are passed from vehicle to vehicle using the same DLT framework introduced in [4, 5], where vehicles in possession of tokens are permitted to write data to the DLT. The token passing mechanism is dictated by both the MUBEV-SM algorithm and the DLT architecture and can be implemented using a MEC/cloud-based service (also described in [4, 5]). Once the unknown environment has been ascertained, route recommendations can be accessed via a smart parking app by a variety of users interested in optimal routing.

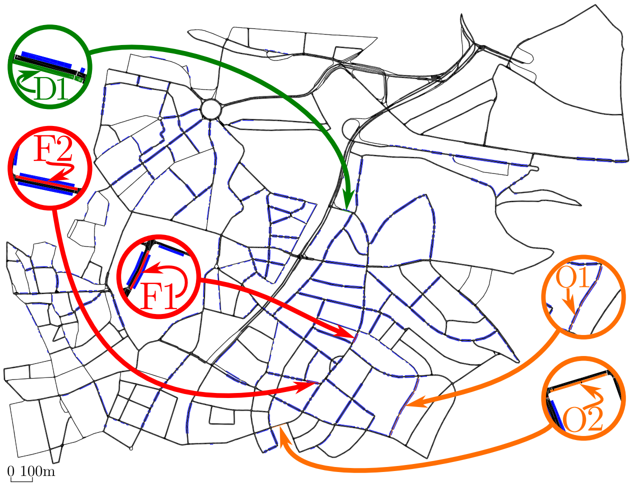

Figure 2 - The road network used in the experiments comes from an urban area close to the city centre in Würzburg, Germany. In the OpenStreetMap network of this area[4], PSs have been tagged, and thus on-street PSs can be easily imported to a SUMO road network file. Some road segments of interest are highlighted, representing either origins (O), destinations (D), or sources of full parking areas (F).

To evaluate MUBEV-SM, a number of numerical experiments was designed using the traffic simulator SUMO [16]. Interactions with running simulations (including those to measure roadside parking occupancy) are accomplished using the Python programming language and SUMO packages *Sumolib* and *TraCI*. The generic settings for our simulations are as follows:

- The urban area in Würzburg, Germany shown in Figure 2 is used for traffic simulations in all our experiments. Once all the U-turns are removed[5] from the network file, the road network results in a graph with 861 nodes (states).
- A number of road links are selected as origins, destinations, and sources of full on-street parking areas (see Figure 2 for details). Specifically, we use: origin $O1$ and full parking area (Full-PA) $F1$ in Experiment 1 and 2; origin $O2$ and Full-PA $F2$ in Experiment 3; destination $D1$ in all our experiments.
- In all our simulations, we create and use a vehicle type based on the default SUMO vehicle type with maximum speed equal to 118.8 km/h and impatience[6] equal to 0.5. When these vehicles are in possession of a token, they become *virtual* MUBEV-SM *vehicles*. In Experiment 1 and 2, these vehicles have no parking stops assigned. In Experiment 3, every 100th vehicle added to the simulation is assigned to a random parking stop along its route. For the generation of Full-PAs, we release a number of cars of the aforementioned vehicle type and populate the selected on-street parking lanes with them.
- For all our experiments, we add an average of 250 background cars with random routes, which can carry tokens if required.
- DPs are calculated using Dijkstra's SP algorithm (SciPy[7] implementation).
- A state of an RL agent is considered to be a set of road links joined into one state using the merging technique introduced in [4]. We refer to a token trip as an RL episode. Before policy execution, the tokens are uniformly distributed on the road network. The length of the MDP's time horizon H is set to 50. H determines a maximum number of allowed road links that each token can traverse over a given episode. If tokens do not reach a specified destination within this restriction, their trips will be declared incomplete (i.e. unsuccessful).

[4] `https://www.openstreetmap.org/#map=14/49.7830/9.9553&layers=T`

[5] U-turns may lead to undesirable recurrent attempts to use DP policy.

[6] `https://sumo.dlr.de/wiki/Definition_of_Vehicles_Vehicle_Types_and_Routes`.

[7] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.csgraph.dijkstra.html`

Regarding the input parameters of Algorithm 1 and Function 1, in all our experiments we set $\varepsilon = \delta = r_{max} = 1$, $J = 10$, $\phi_{min} = 1.25$. The components of vector γ are computed as follows: $\gamma_j = 1 - (J - j) * (1 - \gamma_{min})/(J - 1)$, $j \in [J]$, with γ_{min} empirically set to 0.1. In Experiment 1, we set $\beta_1 = \beta_2 = 20$ for simulations with $\alpha = 0.66$, and $\beta_1 = \beta_2 = 60$ for $\alpha = 0.80$. In Experiment 2 and 3, we set $\beta_1 = \beta_2 = 20$ and $\alpha = \alpha_{min} = 0.66$. Any specific additional setting for each individual experiment will be described in the corresponding subsection below.

Experiment 1: Optimal route estimation under uncertainty using different values of α
The objectives of this experiment is to illustrate that our DLT-enabled RL approach is able to determine a simple uncertain environment using a single token for different values of α. For a chosen OD pair, we artificially simulate a Full-PA on some road links along DP at various time intervals. In this scenario, our goal is to show that the token-enabled MUBEV-SM algorithm can distinguish between Free Parking Intervals (Free-PIs) and Full Parking Intervals (Full-PIs), and, in the latter case, find the next optimal route for the selected OD pair.

Particularly, this illustrative experiment is designed as follows. A single MUBEV-SM token is used over each episode of the learning process to collect data used to update the MDP's policy. This token has a fixed OD pair, namely $\{O1, D1\}$ as marked in Figure 2. Additionally, we choose $F1$ (a set of road segments which belong to DP for the specified OD pair) and generate a Full-PA in it at different time intervals. Over each episode, we start the token from $O1$ and ask it to travel to $D1$, keeping a record of its performance in terms of route characteristics, such as the total weight (computed using Equation 1), the number of available on-street PSs, and the travel distance (route length), regardless of its success in attempting to reach $D1$. During Full-PIs, the total weight of the token route enlarges while parking availability decreases (this is annotated with "DP full" for the first Full-PI in Figure 3). MUBEV-SM is expected to provide an alternative path (AP) which is an optimal solution leading to a minimum possible value of total weight during Full-PIs (as also annotated in Figure 3), and that DP routing would eventually be advised in all other situations.

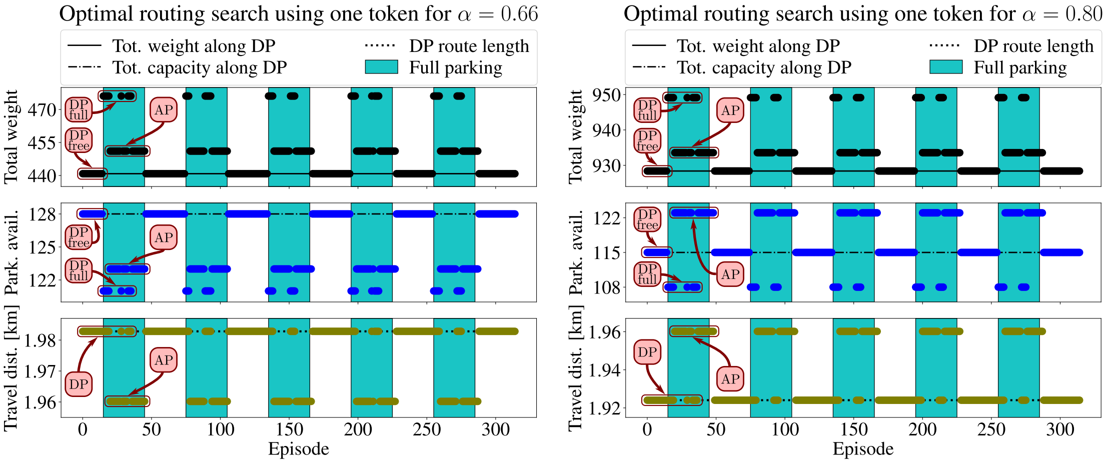

Figure 3 - Experiment 1: Performance indices of MUBEV-SM with MWRM during the learning process on a changing environment using one token and fixed OD pair $\{O1, D1\}$ for $\alpha = 0.66$ (Left) and $\alpha = 0.80$ (Right). Annotations are only provided for the first part of the learning process (DP: default path, AP: alternative path).

The results of a single realisation of the previously described experiment are depicted in Figure 3 for different values of α. As it can be noted, the token succeeds in avoiding Full-PAs once they are created (which can be better observed in the travel distance plots), and learns an AP route with a smaller total weight than that of DP during Full-PIs (annotated as “DP full”). The token also successfully returns to DP once Full-PAs are cleared. Both actions, avoiding and returning, happen within a reasonably small number of episodes. Additionally, thanks to MWRM, this rapid adaptation is present even at the beginning of the learning process, and the performance of MUBEV-SM is nearly uniform as time passes. These observations validate our expectations about MUBEV-SM concerning its ability to adapt rapidly to uncertain environments.

When comparing the right and left panels of Figure 3, we can notice that the length of DP, and the total parking capacity along it, are larger for a smaller value of α (left panel) than the corresponding values for a larger α (right panel). This indicates, as expected, that smaller values of α generally lead to longer DP routes. Consequently, it would be impractical to consider an extremely small value of α (even if it satisfies $\alpha \in [\alpha_{min}, 1]$), as it could lead to a dramatic increase in travel distances/times. As it can also be seen in Figure 3, our MUBEV-SM performs as expected for both values of α.

Experiment 2: Comparative analysis of MUBEV-SM with and without MWRM

In this second experiment, we want to evaluate the performance of MUBEV-SM with and without MWRM using a single token and non-uniform Full-PIs. For this, we use a similar setup as in Experiment 1 (i.e. same OD pair, Full-PA on $F1$, and one token probing the environment). To begin with, the token starts at $O1$ and is asked to travel to $D1$ each episode, keeping a record of the performance indices as mentioned in the previous subsection. We intentionally do not introduce a Full-PA for a long period of time, allowing the token to collect lots of “positive” rewards while traveling along DP, after which we generate a Full-PA over a reasonable short time interval. Afterwards, we allow a short Free-PI followed by a Full-PA on $F1$ again, but over a significantly longer period of time. The results of this comparative analysis for a single realisation are shown in Figure 4.

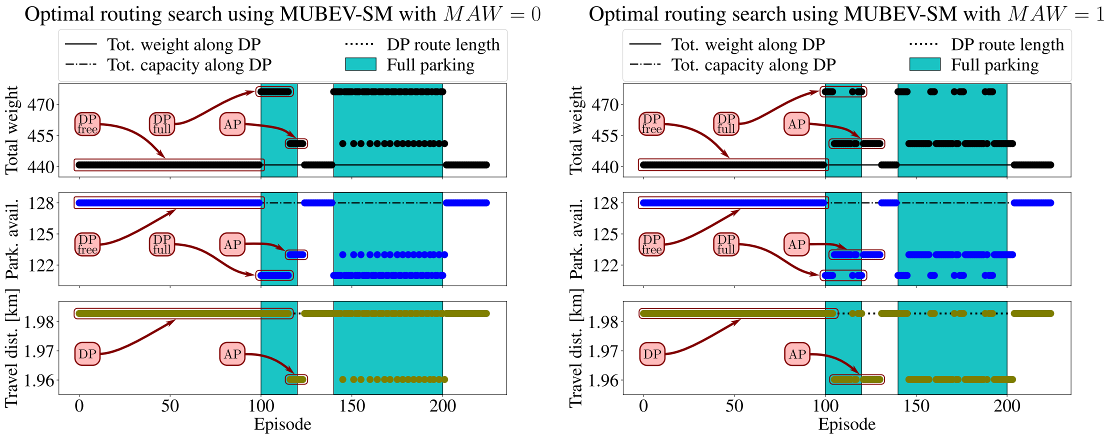

Figure 4 - Experiment 2: Comparative performance of MUBEV-SM without (Left) and with (Right) MWRM on a changing environment with non-uniform Full-PIs, using a single token and fixed OD pair. Annotations are only provided for the first part of the learning process (DP: default path, AP: alternative path).

From Figure 4-Left, we can notice that the learning process using MUBEV-SM without MWRM is notably slow during the first Full-PI (better observed in the travel distance plot, left panel), meaning that a large number of episodes are required to learn an AP. On the contrary, the algorithm is able to provide an optimal detour AP much faster (in terms of fewer episodes) if MWRM is applied as seen in Figure 4-Right. The latter configuration also allows for a better performance during the second Full-PI: MUBEV-SM is able to learn an AP route in a small number of episodes, and such an AP solution is preferred over the Full-PI. It must be noted that without MWRM, the system mostly selects DP routing over the whole learning process, which can be seen in the lowest subplot in Figure 4-Left. The reason of such performance is that the system without MWRM has collected lots of “positive” rewards and statistics during the first (long) Free-PI.

Finally, it can be observed that returning to DP (once the Full-PAs are cleared) takes slightly longer with MWRM than without. However, with MWRM, the system does not require much tuning of its parameters for each particular uncertainty, and performs significantly better in general. Note that experiments with a single token are useful to analyse the performance of MUBEV-SM in uncertain environments, but more elaborated experiments must be performed for a comprehensive validation. Experiment 3 addresses this.

Experiment 3: Route recommendations from a MUBEV-SM-based system, and speedup in learning
Along with an illustration of how MUBEV-SM route recommendations can be delivered to a number of drivers, this experiment is also designed to evaluate the performance of the system as a function of the number of tokens used to collect the data to update the MDP’s policy. The setup for this experiment is as follows. Over each episode, we release a set of tokens starting at different origins according to a uniform spatial distribution. All these tokens have a common destination $D1$. In parallel, we analyse the performance of a *test* (non-MUBEV-SM) *vehicle* starting at origin $O2$ every single episode. The test vehicle is asked to travel to destination $D1$ using recommendations provided by a MUBEV-SM-based routing system, whose recommendations are defined as follows: (i) a route connecting $\{O2, D1\}$ is calculated based on the estimated MDP’s policy; and (ii) whenever a complete route for $\{O2, D1\}$ cannot be calculated using the MUBEV-SM policy (e.g. the destination is not reached within the length horizon H), the latest valid recommendation is reused. A Full-PA is generated on the set of road links $F2$ (which belong to the DP for $\{O2, D1\}$) during episodes 25 to 45. It is worth noting that a new test car is released at the end of each episode as soon as the policy is updated. Figure 5 depicts the results for this experiment obtained from 50 different realisations.

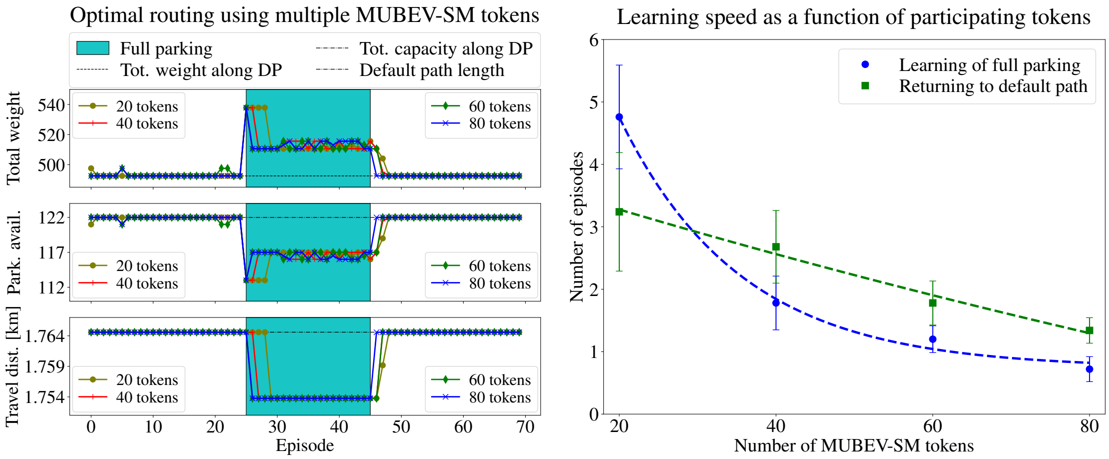

Figure 5 - Experiment 3: Left: Median values of route attributes for a test vehicle using recommendations from a MUBEV-SM-based routing system involving multiple tokens. Each datapoint represents the median value of 50 different realisations of the experiment. Right: Average learning speed using multiple tokens. The error bars depict the 95% confidence interval for the mean. Dashed lines were obtained using exponential curve fitting.

As it can be concluded from Figure 5-Left, the recommender system has a remarkable performance for different number of participating tokens. However, the number of tokens directly affects the convergence rate of the MUBEV-SM algorithm: the more tokens involved, the faster the learning process. A more conclusive relationship between the number of participating tokens and the average number of episodes required to learn a new traffic condition (either free or full parking areas) is depicted in Figure 5-Right. This relationship reflects an exponential-like decay for learning of Full-PAs, and a nearly linear decrease for returning to DP.

Conclusion

We propose a routing system design based on DLT and multi-agent RL technique to solve the on-street parking problem. The proposed approach is applied to estimate the best routes in terms of short travel distance and large number of available roadside PSs, and the design allows drivers to choose their preferred objectives. A moving window mechanism is also incorporated into the underlying RL algorithm to better cope with non-stationary environments. Experimental results have proven the efficiency of our system in improving the parking problems. For future work, we plan to include more objectives into our optimisation problem (e.g. travel time, walking distance), and extend the range of allowable values of α.

Acknowledgements

This work has been supported by Science Foundation Ireland (SFI) under grant No. 16/IA/4610. Part of this work has received funding from the EU's Horizon 2020 research & innovation programme via the ICT4CART project under grant agreement No. 768953. Content reflects only the authors' view and the European Commission is not responsible for any use that may be made of the information it contains.

References

[1] D. C. Shoup, "Cruising for parking," *Transport Policy*, vol. 13, no. 6, pp. 479–486, 2006.

[2] C. Tang *et al.*, "Towards Smart Parking Based on Fog Computing," *IEEE Access*, vol. 6, pp. 70172–70185, 2018.

[3] M. Buchholz *et al.*, "Enabling automated driving by ICT infrastructure: A reference architecture," in *Proceedings of TRA'20, Helsinki (conference cancelled)*, 2020. doi: 10.18725/OPARU-26023.

[4] R. Overko *et al.*, "Spatial Positioning Token (SPToken) for Smart Mobility," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–14, 2020.

[5] R. Overko *et al.*, "Spatial Positioning Token (SPToken) for Smart Mobility," in *Proceedings of IEEE ICCVE'19*, pp. 1–6, 2019.

[6] K. Garlichs *et al.*, "Leveraging the Collective Perception Service for CAM Information Aggregation at Intersections," 2020. Accepted at IEEE VTC2020-Fall.

[7] W. Al Amiri *et al.*, "Privacy-Preserving Smart Parking System Using Blockchain and Private Information Retrieval," in *Proceedings of SmartNets'19*, pp. 1–6, 2019.

[8] J. Ni *et al.*, "Privacy-Preserving Smart Parking Navigation Supporting Efficient Driving Guidance Retrieval," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 7, pp. 6504–6517, 2018.

[9] M. Khalid *et al.*, "A Reinforcement Learning based Path Guidance Scheme for Long-range Autonomous Valet Parking in Smart Cities," in *Proceedings of ComNet'20*, pp. 1–7, 2020.

[10] S. Padakandla *et al.*, "Reinforcement Learning in Non-Stationary Environments," *Applied Intelligence*, vol. 50, pp. 3590–3606, 2020.

[11] M.-C. Wu *et al.*, "Early detection of vacant parking spaces using dashcam videos," in *Proceedings of AAAI'19*, vol. 33, pp. 9613–9618, 2019.

[12] F. Bock *et al.*, "On-street parking statistics using lidar mobile mapping," in *Proceedings of ITSC'15*, pp. 2812–2818, IEEE, 2015.

[13] R. Overko *et al.*, "Reinforcement Learning Augmented Optimization for Smart Mobility," in *Proceedings of IEEE CDC'19*, pp. 1286–1292, 2019.

[14] A. Zanette *et al.*, "Problem Dependent Reinforcement Learning Bounds Which Can Identify Bandit Structure in MDPs," in *Proceedings of MLR'18*, vol. 80, pp. 5747–5755, PMLR, 10–15 Jul 2018.

[15] C. Dann *et al.*, "Unifying PAC and regret: Uniform PAC bounds for episodic reinforcement learning," in *Proceedings of NeurIPS'17*, pp. 5713–5723, 2017.

[16] P. A. Lopez *et al.*, "Microscopic Traffic Simulation using SUMO," in *Proceedings of ITSC'18*, pp. 2575–2582, IEEE, 2018.