





A Carbon-Neutral, Community-Based, Reactive and Scalable Ride-Sharing Service

Avinash Nagarajan¹^a, Alan McGibney²^b, Pio Fenton³^c and Ignacio Castiñeiras¹^d

¹Department of Computer Science, Munster Technological University, Cork, Ireland

²NIMBUS Centre, Munster Technological University, Cork, Ireland

³Teaching & Learning Unit, Munster Technological University, Cork, Ireland

Keywords: Ride-Sharing, Smart Energy Communities, Green Computing, Optimisation.

Abstract: This paper presents the design, implementation and evaluation of a community-based, reactive ride-sharing service that promotes carbon-neutral management of city commutes using autonomous vehicles. The problem is formulated as a variant of the classical Dynamic Vehicle Routing problem with Time Windows, considering both dynamic resources and requests over a simulated time horizon. A solution approach is provided, based on an algorithm following a reactive-based simulation on top of a greedy decision-making process. A parameterised instance generator is developed to align existing benchmarks (i.e. Google HashCode) and public datasets (i.e. NYC taxis) to the proposed problem formulation and to enable testing of the solution under various configurations. The ride-sharing service is proven to scale well, when applied to very large instances, it provides fast and competitive results, both in terms of trip petitions satisfied and overall distance traversed.

1 INTRODUCTION


The European Green Deal has set the vision for a climate neutral continent by 2050 (EU Green Deal, 2020). For transportation systems to meet the aforementioned goal, a sustained transition to mass electric vehicles (EV) and a fully Renewable Energy Sources (RES)-based energy market to power them is critical (Xiang et al., 2016). There is an onus on all citizens to play a role in this green transition, for example Smart Energy Communities (SEC) are emerging to enable distributed energy generation and storage at a local level, representing a step further to the realisation of the term ‘Energy Citizen’ (Diamantoulakis et al., 2015).


Ride-sharing has presented new opportunities for the SEC sector to be productive with energy usage (Agatz et al., 2012). Dynamic ride-sharing systems aim to match riders and drivers with similar itineraries and time schedules on short-notice (Furuhata et al., 2013). These systems can reduce the number of cars used for personal travel (improving the utilisation


of available seat capacity) as well as the total distance for such travels, thus reducing the environmental impact. The research presented in this paper includes the design, implementation and evaluation of a ride-sharing service operating in a Smart City environment to align with the ambition of the EU Green Deal goals, and therefore envisioning an alternative carbon-neutral, community-based, reactive transportation approach for managing city commutes using autonomous vehicles. Emphasis is placed on individual citizens (rather than on haulage/commercial transportation) that collectively aim to minimise their impact on the environment through the use of ride-sharing commutes relying on 100% EVs operating solely on RES.


Specifically, the paper includes the following contributions:

- The formulation of a ride-sharing simulation service with the aforementioned conditions, presented as a variant of the classical Dynamic Vehicle Routing Problem with Time Windows (Jaillet and Wagner, 2008). The dynamic resources are provided by a number of SECs spread across a city, each of them owning a number of autonomous EVs and using its own RES generation function to dynamically charge (and release) them

^a <https://orcid.org/0000-0002-4533-0527>

^b <https://orcid.org/0000-0002-0665-2005>

^c <https://orcid.org/0000-0002-1673-9737>

^d <https://orcid.org/0000-0002-3875-4460>

over time. The dynamic requests are provided via trip petitions released over time, each of them with its own location and pick-up/drop-off times. The formulation of the ride-sharing service integrates energy generation, allocation and reactive re-routing constraints, with the objective function of maximising the overall number of trip petitions being served.

- The design and implementation of a ride-sharing service, based on an algorithm following a reactive-based simulation approach on top of a greedy-based decision-making process. The algorithm favours scalability over optimality, therefore evaluating its applicability to real-world instances based on the transportation of large cities.
- The evaluation of the solution approach using a parameterised instance generator, allowing for the fine-grained customisation of a Vehicle Routing Problem benchmark and of a public transportation-based dataset to the specific requirements and format of the ride-sharing problem. A problem-specific benchmark is generated, testing the performance and scalability of the algorithm over a number of configurations.

The structure of the paper is as follows: Section 2 positions the paper w.r.t. existing ride-sharing approaches and solutions. Section 3 formalises the problem for the carbon-neutral, community-based reactive ride-sharing service being proposed, Section 4 presents the solution approach and Section 5 its evaluation. Finally, Section 6 provides the main conclusions and future work.

2 RELATED WORK

In (Hasan et al., 2018) the authors argue the viability of a ride-sharing service is dependent on six core principles: spatial and temporal proximity of the riders, low coordination costs, guaranteed to ride back home, low trust concerns and clear commuter roles. Spatial and temporal proximity reduces per-trip costs by matching customers with similar schedules and locations. The other four principles come down to psychological factors that make the service reliable. Their approach focuses on one of the passengers owning the vehicle, and therefore is explicitly attached to the trips of the vehicle like taxis. The optimisation criteria centres around maximising the incentives of key stakeholders.

The authors in (Agatz et al., 2012) focus on dynamic ride-sharing from the point of view of its sharing cost mechanism. Uncertainty in trip petitions

is considered, with trip applications ranging from a few minutes to a few hours before departure time. The algorithm then evaluates different re-optimisation frequencies for finding ride-share arrangements each time a new trip petition arrives or at fixed time intervals. Drivers are considered private entities, thus enabling vehicle availability only when aligned to their trip needs. The ride-sharing service focuses on optimising the total distance or the total travel time of each individual ride.

In (Hasan and Hentenryck, 2020) the authors focus on dynamic ride-sharing from the point of view of enabling the scheduling of ride-sharing trips in real-time. Their approach reuses the idea of re-assessment over a time horizon for matching trip petitions requested just a few minutes before its departure. Although the authors focus on clustering drivers and passengers of the same neighbourhood together (to provide a set of possible feasible trips), it is intended to address the return trip of a population for which an outbound trip has already been provided. The optimisation criteria is the quality of service where once a passenger has been assigned to a trip session, they are guaranteed a ride back.

In (Serra et al., 2019) the authors focus on dynamic ride-sharing for the last-mile from the point of view of integrating it as part of a multi-modal transportation system. The paper considers the routing of passengers who have nearly completed their commute after having reached a central station by using public transport and are now in a position of scattering to their final destination points. The goal is to reduce the number of vehicles on road by efficient passenger-vehicle allocation.

The authors of (Al-Abbasi et al., 2019) focus on ride-sharing from the point of view of trust and privacy. They propose a co-util framework based on Game Theory, which requires all involved parties to benefit by being an active part of the solution. In the proposed ride-sharing service, the passengers benefit from low travel costs. The goal is to reduce the number of vehicles on the road and the cost of mitigating traffic congestion.

The study of community-based reactive ride-sharing systems has shown significant potential in mitigating the impact of conventional transportation on the environment and promoting the use of RES. The key factor contributing to this potential is the scalability, robustness and uncertainty of such systems. By leveraging the power of autonomous electric vehicles, community-based reactive ride-sharing systems can be tailored to meet the diverse and evolving transportation needs of modern society while reducing their carbon footprint.

3 PROBLEM DEFINITION

This section formalises the problem for a carbon-neutral, community-based reactive ride-sharing service as a variant of the classical Dynamic Vehicle Routing Problem with time windows. It integrates energy generation, allocation and reactive re-routing constraints as both the trip petitions and the number of available vehicles evolve during a simulated time horizon. The ride-sharing service aims to maximise the number of trip petitions being served. It is also intended to be highly scalable to facilitate transportation in large cities.

The ride-sharing service contains the following features:

- The grid dimensions of the city c_r, c_c and the time horizon for the simulation th . W.l.o.g., Manhattan distances among the locations of the city are assumed.

$$(c_r, c_c, th)$$

- A set S of SECs, each of them with an id s_{id} , a location in the city s_x, s_y , a lexicographic-ordered list of the vehicles belonging to it s_E , the amount of vehicles ready at the start of the simulation (i.e., at time unit 0) s_R and an energy function $f_{s_{id}} : \mathbb{N} \times \mathbb{N}$ with the energy produced per time unit of the simulation, which will be used to release (in order) each new vehicle from s_E as soon as enough energy to fill its battery capacity is generated.

$$(s_{id}, s_x, s_y, s_E, s_R, f_{s_{id}}) \forall s \in S$$

- A set E of EVs, each of them with its own id e_{id} , the id of the SEC it belongs to s_{id} , its release time during the simulation e_{rt} (either 0 or when enough energy is generated) and its battery and passenger capacities (e_{bc} and e_{pc}). A mapping from E to S is assumed, such that each e_{id} belongs to one s_{id} .

$$(s_{id}, e_{id}, e_{rt}, e_{bc}, e_{pc}) \forall e \in E$$

- A set T of trip petitions (TPs), each of them with its id t_{id} , its release time during the simulation t_{rt} and its pick-up (resp. drop-off) locations t_{px}, t_{py} (resp. t_{dx}, t_{dy}) and time-windows t_{ep}, t_{lp} (resp. t_{ed}, t_{ld}).

$$(t_{id}, t_{rt}, t_{px}, t_{py}, t_{ep}, t_{lp}, t_{dx}, t_{dy}, t_{ed}, t_{ld}) \forall t \in T$$

The ride-sharing service is intended to provide an output containing the following features:

- A set $Alloc$, representing an allocation mapping from T to E .

$$\forall t \in T \text{ Alloc}[t_{id}] = \begin{cases} e_{id}, & \text{if } t_{id} \text{ is allocated to } e_{id} \\ -1, & \text{otherwise} \end{cases}$$

- A set $Sched$, representing a scheduling mapping from E to its sequence of movements (in chronological order) over the entire time horizon.

$$\forall e \in E \text{ Sched}[e_{id}] = (m_0, m_1, \dots, m_{n-1})$$

Each movement m_i of a vehicle is represented as

$$m_i \equiv \{(TA_i, TB_i, AX_i, AY_i, BX_i, BY_i, PS_i, PE_i, ES_i, EE_i, TL_i, LW_i, TD_i)\}$$

with:

- TA_i (resp. TB_i) represent the start time (resp. end time) of m_i .
- AX_i and AY_i (resp. BX_i and BY_i) represent the coordinates of the vehicle at the start (resp. end) of m_i .
- PS_i (resp. PE_i) represents the number of passengers in the vehicle at the start time (resp. end time) of m_i .
- ES_i (resp. EE_i) represents the battery left in the vehicle at the start time (resp. end time) of m_i .
- TL_i represents a label to identify the purpose of the trip.
 - * A movement for picking-up (resp. dropping off) the passenger of t_{id} is marked as $+t_{id}$ (resp. $-t_{id}$).
 - * An idle movement (resting at a given location) is marked as *idle*.
 - * A last movement, returning to its home SEC is marked as *ret*.
- LW_i represents the leeway of the movement, i.e., the maximum delay that could be applied to the movement while still accomplishing the action is intended to.
- TD_i represents the movement duration.

3.1 Instance Example

To highlight the approach taken, an example of a problem instance is presented here. A possible instance represents a city of dimensions 3×4 (w.l.o.g, $3\text{km} \times 4\text{km}$) and a simulated time horizon of 10 units (w.l.o.g., 10 minutes).

$$(c_r = 3, c_c = 4, th = 10)$$

The city contains just 1 SEC, with id SEC_1 and placed at location (1,2), with a list of 2 vehicles $S_E \equiv [EV_1, EV_2]$, one vehicle ready to go at the start of the simulation, and a constant energy generation function of 2kWh per time unit $f_{SEC_1} = 2$.

$$S \equiv \{(SEC_1, 1, 2, [EV_1, EV_2], 1, f_{SEC_1})\}$$

A constant speed is assumed, making each vehicle to traverse 1 block of the city every time unit, while consuming 1 unit of its battery capacity (w.l.o.g., 60km/h constant speed and 60kWh constant consumption). Both EVs have a battery capacity of 10kWh and room for 5 passengers. Whereas EV_1 is released at the start of the simulation (time unit 0), EV_2 is released when its battery capacity is generated by SEC_1 (i.e., given the constant energy generation of 2kWh per time unit, the vehicle is generated at time unit 5). If SEC_1 had had more vehicles in S_E , they would have been released (in order) at time units 10, 15, and so on.

$$E \equiv \{(SEC_1, EV_1, 0, 10, 5), (SEC_1, EV_2, 5, 10, 5)\}$$

The city receives 3 TPs, with ids TP_1 , TP_2 and TP_3 . TP_1 is announced at time unit 0, to pick-up the passenger at location (2,3) between time units 0 and 4, and to drop-off at location (0,0) between time units 5 and 7. TP_2 is announced at time unit 2, to pick-up the passenger at location (2,2) between time units 3 and 4, and to drop-off at location (1,1) between time units 5 and 6. TP_3 is announced at time unit 6, to pick-up the passenger at location (2,3) between time units 6 and 7, and to drop-off at location (1,3) between time units 7 and 9.

$$T \equiv \{(TP_1, (0, 2, 3, 0, 0, 0, 4, 5, 7)), \\ (TP_2, (2, 2, 1, 1, 1, 3, 4, 5, 6)), \\ (TP_3, (6, 2, 3, 1, 3, 6, 7, 7, 9))\}$$

A feasible solution to this instance allocates TP_1 and TP_2 to EV_1 , while TP_3 is not allocated.

$$Alloc \equiv \{(TP_1, EV_1), (TP_2, EV_1), (TP_3, -1)\}$$

In the allocation above, the vehicles have the following schedules:

$$Sched \equiv \{(EV_1, [(0, 2, 1, 2, 2, 3, 0, 1, 10, 8, +TP_1, 2, 2), \\ (2, 4, 2, 3, 2, 1, 1, 2, 8, 6, +TP_2, 0, 2), \\ (4, 5, 2, 1, 1, 1, 2, 1, 6, 5, -TP_2, 1, 1), \\ (5, 7, 1, 1, 0, 0, 1, 0, 5, 3, -TP_1, 0, 2), \\ (7, 10, 0, 0, 1, 2, 0, 0, 3, 0, ret, 0, 3)]), \\ (EV_2, [(5, 10, 1, 2, 1, 2, 0, 0, 10, 10, idle, 5, 0)])\}$$

Figure 1 displays the route of EV_1 and EV_2 in $Sched$. W.l.o.g., the coverage of the Manhattan distance between two points is assumed to be traversed always by covering first the distance of the x-axis followed up by the distance in the y-axis. The release of an EV is highlighted in blue, an idle movement in grey and an active movement by its starting point (green) and destination one (red).

The route of EV_1 is composed of 5 movements:

1. From time unit 0 to time unit 2, it leaves the SEC_1 to serve the pick-up of TP_1 . On doing so, it goes from (1,2) to (2,3), increases its number of passengers from 0 to 1, and reduces its battery capacity from 10 to 8. As the pick-up of TP_1 must be within time units 0 and 4 and EV_1 arrives at time unit 2, a leeway of 2 time units is associated to the movement, as it could have been re-scheduled by delaying it up to 2 time units in case the vehicle had needed any re-routing.
2. From time unit 2 to time unit 4, the vehicle serves the pick-up of TP_2 . On doing so, it goes from (2,3) to (2,1), increases its number of passengers from 1 to 2, and reduces its battery capacity from 8 to 6. As the pick-up of TP_2 must be within time units 3 and 4 and EV_1 arrives at time unit 4, a leeway of 0 time units is associated to the movement, as it cannot be further delayed by any re-routing.
3. From time unit 4 to time unit 5, the vehicle serves the drop-off of TP_2 . On doing so, it goes from (2,1) to (1,1), decreases its number of passengers from 2 to 1, and reduces its battery capacity from 6 to 5. As the drop-off of TP_2 must be within time units 5 and 6 and EV_1 arrives at time unit 5, a leeway of 1 time unit is associated to the movement.
4. From time unit 5 to time unit 7, the vehicle serves the drop-off of TP_1 . On doing so, it goes from (1,1) to (0,0), decreases its number of passengers from 1 to 0, and reduces its battery capacity from 5 to 3. As the drop-off of TP_1 must be within time units 5 and 7 and EV_1 arrives at time unit 7, a leeway of 0 time units is associated to the movement.
5. From time unit 7 to time unit 10, the vehicle returns to SEC_1 before the end of the simulation. On doing so, it goes from (0,0) to (1,2), stays at 0 passengers, and reduces its battery capacity from 3 to 0. As the end of the time horizon is at time unit 10, and the vehicle arrives to SEC_1 at time unit 10, a leeway of 0 time units is associated to the movement.

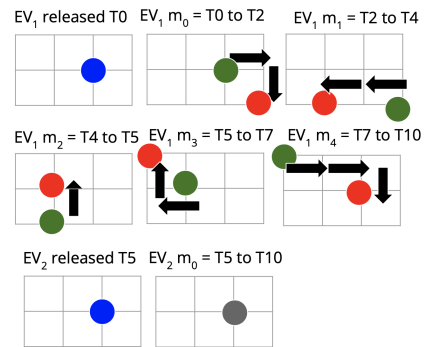


Figure 1: $Sched$ Outputted as Solution.

The route of EV_2 is composed of just 1 movement, as the vehicle remains idle since it is released at time unit 5 until the end of the time horizon.

4 SOLUTION APPROACH

This section presents an initial algorithm to solve the problem formalised in Section 3. Subsections 4.1 and 4.2 present the algorithm and analyse its complexity, respectively.

4.1 Algorithm Overview

Given a valid instance of a city with its SECs, vehicles, energy generation and trip petitions, the algorithm aims to maximise the number of trip petitions being served, outputting both the allocation of each trip to a vehicle and the routing of each vehicle over the entire simulated time horizon. For doing so, the algorithm implements a reactive-based simulation approach on top of a greedy decision making process for trip allocations. Algorithm 1 presents the pseudocode of the solution approach, which is explained in detail next.

Algorithm 1: Ride-Sharing.

```

function allocate( $e, t, Alloc[t], Sched[e]$ )
   $is\_allocated \leftarrow false$ 
   $m \equiv [m_0, \dots, m_{k-1}] \leftarrow copy(Sched[e])$ 
  for  $i \in 0, \dots, k-1$  do
    if pick_up( $m, t, i$ ) then
       $m \leftarrow [m_0, \dots, m'_i, m'_{i+1}, \dots, m'_{r-1}]$ 
      for  $j \in i+1, \dots, r-1$  do
        if drop_off( $m, t, j$ ) then
           $m \leftarrow [m_0, \dots, m'_j, m''_{j+1}, \dots, m''_{w-1}]$ 
          if ret_sec( $m, w-1$ ) then
             $m \leftarrow [m_0, \dots, m''_{w-1}, m''_w]$ 
            ( $Alloc[t], Sched[e]$ )  $\leftarrow (e, m)$ 
             $is\_allocated \leftarrow true$ 
          Break
      Break
  return  $is\_allocated$ 

```

```

function reactive_simulation( $S, E, T, th$ )
  ( $Alloc, Sched$ )  $\leftarrow init(S, E, T, th)$ 
  for  $t \in sorted(T)$  do
    for  $e \in E$  do
      if allocate( $e, t, Alloc[t], Sched[e]$ ) then
        Break
  return ( $Alloc, Sched$ )

```

4.1.1 Reactive-Based Simulation

First, the algorithm initialises $Alloc$, considering each trip as initially not allocated. It also initialises $Sched$,

considering the route of each vehicle as, initially, resting at its home SEC from its release time e_{rt} to the end of the time horizon of the simulation th . Therefore, even at the beginning of the simulation (time unit 0), all vehicles have an associated schedule (even if such schedule does not start until a release time unit e_{rt} in the future).

$$\forall t \in T \quad Alloc[t_{id}] = (e_{id}, -1)$$

$$\forall e \in E \quad \exists s_{id} \in S \text{ with } s_x, s_y, s_E, e \in s_E \text{ s.t. } Sched[e_{id}] = [(e_{rt}, th, s_x, s_y, s_x, s_y, 0, 0, e_{bc}, e_{bc}, idle, th - e_{rt}, 0)]$$

For example, given the instance of Section 3.1, the initial value of $Alloc$ and $Sched$ is as follows:

$$Alloc \equiv \{(TP_1, -1), (TP_2, -1), (TP_3, -1)\}$$

$$Sched \equiv \{(EV_1, [(0, 10, 1, 2, 1, 2, 0, 0, 10, 10, idle, 10, 0)]) \\ (EV_2, [(5, 10, 1, 2, 1, 2, 0, 0, 10, 10, idle, 5, 0)])\}$$

The algorithm then simulates a reactive-based ride-sharing service by simply sorting all trip petitions by their increasing release time (referred to as $sorted(T)$), thus attempting to serve each trip petition as soon as it is released. For each $t \in sorted(T)$, the algorithm iterates for each $e \in E$, attempting to allocate the trip to the vehicle by dynamically re-routing its schedule (i.e., by fitting both the pick-up and drop-off of t as new movements into the existing schedule of the vehicle $Sched[e]$). As soon as the algorithm successfully allocates t to a vehicle e , the search stops (i.e., the trip is not attempted to be allocated to any other vehicle). On the other hand, if no vehicle can allocate t , then the attempt to serve the trip petition is considered as not successful, and no further attempt for allocating it is made for the rest of the simulation.

For example, given the instance of Section 3.1, the trips TP_1 , TP_2 and TP_3 are released in time units 0, 2 and 6, respectively. Therefore:

1. The algorithm simulates that TP_1 is attempted to be allocated first, at time unit 0. On that moment, the schedule of EV_1 and EV_2 is the initial one described above. If EV_1 can be successfully re-routed to fit TP_1 , then it is allocated to it. Otherwise EV_2 is attempted. If TP_1 can not be serviced by either EV_1 or EV_2 , then TP_1 is not allocated. In any case, these allocation attempts lead to a new updated state $Sched'$ and $Alloc'$.
2. The algorithm continues by attempting to allocate TP_2 over EV_1 first and EV_2 otherwise with their current updated schedules $Sched'$ at time unit 2,

further leading to a new updated state $Sched''$ and $Alloc''$.

3. Finally, the algorithm continues by attempting to allocate TP_3 over EV_1 first and EV_2 otherwise with their current updated schedules $Sched''$ at time unit 6, leading to the final state (the one reported as output) $Sched'''$ and $Alloc'''$.

4.1.2 Trip Petition Allocation

The allocation of t to e (specifically, to its schedule $Sched[e]$) is defined to be successful iff:

1. The updated $Sched[e]'$ incorporates one new movement ensuring the pick-up (resp. drop-off) of t within its defined time window t_{ep} and t_{lp} (resp. t_{ed} and t_{ld}). The functions $pick_up$ (resp. $drop_off$) ensure this (cf. Algorithm 1).
2. The updated $Sched[e]'$ contains a very last active movement, labelled as ret , ensuring the vehicle returning to its home SEC before the end of the time horizon th . The functions ret_sec ensures this (cf. Algorithm 1).
3. The addition of these new pick-up, drop-off and ret movements to $Sched[e]'$ (to serve t) might delay the actual pick-up and drop-off times of any other previous trips t_z the vehicle e had previously committed to. For t to be successfully allocated to e , such these additional delays on serving t_z must not break the pick-up and drop-off time windows of t_z . In other words, once a vehicle e commits to a trip t_z , no further trip allocation t can delay e enough to make it not serving t_z in time. The functions $pick_up$ (resp. $drop_off$) ensure this (cf. Algorithm 1).
4. The updated $Sched[e]'$ contains no movement where the number of passengers exceeds its capacity e_{pc} or where the battery capacity e_{bc} goes below 0.

To fit t into $Sched[e] = [m_0, \dots, m_{k-1}]$ the algorithm iterates through its movements (which are in chronological order). When considering a movement $m_a \equiv (TA_a, TB_a, \dots)$, the algorithm only considers its starting time unit TA (i.e., the decision of whether to re-route or not e to serve t is considered just at TA , not at any other given time in the interval $(TA, \dots, TB]$).

First, the algorithm searches for a movement m_i fitting the pick-up of t . As soon as such movement m_i is found, no further movement m_j with $j > i$ of e is considered for picking-up t , and the algorithm moves on into searching for a movement m_j fitting the drop-off of t . Again, as soon as such movement m_j is found, no further movement m_k with $k > j$ of e is considered for dropping-off t .

Needless to say, both policies of (1) considering just the starting time TA of each movement and (2) considering just one valid pair (m_i, m_j) of picking-up and dropping-off movements search to be incomplete. That is, the search is not considering other (m_i, m_j) valid combinations and any other valid re-routing times for such combinations other than $TA_{i'}$ and $TA_{j'}$. Any such alternatives might lead not only to the allocation of t , but to an overall higher number of trips allocated, which is the goal aimed by the algorithm. However, these policies are designed to favour scalability over optimality for the algorithm (more detail in Section 4.2). Moreover, although not leading to optimality, the above non-complete search still leads to a very competitive trip allocation rate when applied to real-world very large instances, as it is discussed in more detail in Section 5.2.

Given the instance of Section 3.1, the following analyses the attempt to fit TP_2 into EV_1 at time unit 2.

$$TP_2 \equiv (2, 2, 1, 1, 1, 3, 4, 5, 6)$$

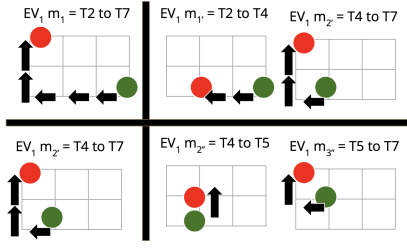
As described previously, at this stage in the simulation, TP_1 had been successfully allocated to EV_1 , making its $Sched[EV_1]$:

$$Sched[EV_1] = [\\ m_0 \equiv (0, 2, 1, 2, 2, 3, 0, 1, 10, 8, +TP_1, 2, 2), \\ m_1 \equiv (2, 7, 2, 3, 0, 0, 1, 0, 8, 3, -TP_1, 0, 5), \\ m_2 \equiv (7, 10, 0, 0, 1, 2, 0, 0, 3, 0, ret, 0, 3)]$$

The window of opportunity for allocating the pick-up of TP_2 is from $TP_2_{rt} = 2$ to $TP_2_{lp} = 4$.

The algorithm first attempts to fit the pick-up on m_0 , but fails, as the time of the movement TA_0 is 0, even before TP_2 is announced, and therefore outside of its window of opportunity.

The algorithm then attempts m_1 , and it succeeds. m_1 spans from time unit $TA_1 = 2$ to $TB_1 = 7$, going from (2,3) to (0,0) for dropping-off TP_1 . The total distance covered is 5. It must be at (0,0) at time unit 7 at the latest ($TP_1_{ld} = 7$), so it cannot be further delayed. Re-routing m_1 to serve pick-up of TP_2 involves (1) going from (2,3) to (2,1); (2) arrive there between time units 3 and 4; (3) going from (2,1) to (0,0); (4) still arrive there on time (i.e., time unit $7 + 0$ leeway = 7). The total distance of re-route to pick-up TP_2 while still dropping-off TP_1 from there is $2 + 3 = 5$; if leaving straight-away at time unit 2, it will reach (2,1) to pick-up TP_2 at time unit 4, still within the window of opportunity, and leading to a leeway of 0, as this new movement cannot be any further delayed.


 Figure 2: $Sched[EV_1]$ when Fitting TP_2 .

The new state of $Sched'[EV_1]$ is:

$$\begin{aligned}
 Sched'[EV_1] = [\\
 m'_0 &\equiv (0, 2, 1, 2, 2, 3, 0, 1, 10, 8, +TP_1, 2, 2), \\
 m'_1 &\equiv (2, 4, 2, 3, 2, 1, 1, 2, 8, 6, +TP_2, 0, 2), \\
 m'_2 &\equiv (4, 7, 2, 1, 0, 0, 2, 1, 6, 3, -TP_1, 0, 3) \\
 m'_3 &\equiv (7, 10, 0, 0, 1, 2, 0, 0, 3, 0, ret, 0, 3)]
 \end{aligned}$$

The window of opportunity for allocating drop-off of TP_2 is from the next time unit to its pick-up (5) to $TP_{2ld} = 6$.

The algorithm first attempts m'_2 (as it does not start attempting the drop-off allocation any earlier than the pick-up movement), and it succeeds. m'_2 spans from time unit $TA'_2 = 5$ to $TB'_2 = 7$, going from (2,1) to (0,0) for dropping-off TP_1 . The total distance covered is 3. It must be at (0,0) at time unit 7 at the latest ($TP_{1ld} = 7$), so it cannot be further delayed. Re-routing m'_2 to serve drop-off of TP_2 involves (1) going from (2,1) to (1,1); (2) arrive there between time units 5 and 6; (3) going from (1,1) to (0,0); (4) still arrive there on time (i.e., time unit 7 + 0 leeway = 7). The total distance of re-route to drop-off TP_2 while still dropping-off TP_1 from there is $1 + 2 = 3$; if leaving straight-away at time unit 4, it will reach (1,1) to drop-off TP_2 at time unit 5, still within the window of opportunity and leading to a leeway of 1 as the time window for dropping TP_2 ends at time unit 6

The new state of $Sched''[EV_1]$ is:

$$\begin{aligned}
 Sched''[EV_1] = [\\
 m_0'' &\equiv (0, 2, 1, 2, 2, 3, 0, 1, 10, 8, +TP_1, 2, 2), \\
 m_1'' &\equiv (2, 4, 2, 3, 2, 1, 1, 2, 8, 6, +TP_2, 0, 2), \\
 m_2'' &\equiv (4, 5, 2, 1, 1, 1, 2, 1, 6, 5, -TP_2, 1, 1) \\
 m_3'' &\equiv (5, 7, 1, 1, 0, 0, 1, 0, 5, 3, -TP_1, 0, 2) \\
 m_4'' &\equiv (7, 10, 0, 0, 1, 2, 0, 0, 3, 0, ret, 3, 0)]
 \end{aligned}$$

Figure 2 displays the re-routing of EV_1 to fit TP_2 . Whereas its top-left (resp. bottom-left) display the movements before the re-routing for the pick-up (resp. drop-off), the top-right (resp. bottom-right) display the movements after the re-routing.

Finally, if a re-routing movement m_i or m_j serving t implies a delay of d units, this delay must be supported by any further movement in the schedule. Idle

movements use their own leeway to reduce d (until eventually reduce it to 0), and any non further movement serving t_z must have a leeway of, at least, the remaining delay when reaching it.

4.2 Complexity Analysis

An evaluation of the algorithm reveals its complexity to be, at worst, $O(n^2)$, where n is the number of trips, m the number of vehicles and $n > m$.

A movement-centric reasoning is used. Given m vehicles, at the start of the simulation each of them has a schedule with a single (very large) idle movement (c.f., 4.1.1). From then on-wards, each trip is attempted to be allocated, by iterating through the vehicles and, for each vehicle, by iterating through its movements. This means that, in the worst case, first trip will need to explore $m \times 1 = m$ movements. If successful, two new movements will be added to the schedule of the vehicle allocating it (one for pick-up and one for drop-off). Thus, after allocating 1 trip, the overall amount of movements of all vehicles will be $m + 2$ (the m movements being there before and the 2 new ones). The same rationale applies over the next trips attempted to be allocated; for the second trip, a worst-case scenario of $m + 2$ movements will be attempted before allocating it, leading to a new overall amount of $m + 4$ movements, and so on. Given n trips, this leads to an overall amount of

$$\sum_{n=0}^{n-1} m + 2n$$

which can be decomposed as

$$\sum_{n=0}^{n-1} m + \sum_{n=0}^{n-1} 2n = (nm) + \frac{(n^2) - 3n - 2}{2}$$

$n > m$, the above expression is dominated by the term $\frac{n^2}{2}$, which leads to a worst-case scenario of $O(n^2)$.

5 EVALUATION

Section 5.1 presents a parameterised instance generator. The parametric generation enables fine-grained customisation of the instances, which is then used in sections 5.2 and 5.3 for respectively analysing the performance and scalability of the solution approach defined in Section 4.

Section 5.4 applies the parametric instance generator to a wider context, in this case customising the well-known, real-world-based data-set of NYC Taxis to valid ride-sharing problem-based instances. This includes a mapping process from the areas and timeline of the original data-set to the grid dimensions and

simulation time horizon required by the ride-sharing problem defined in Section 3. The newly generated benchmark is then used in Section 5.5, analysing the performance of the ride-sharing service from the perspective of the amount of distance covered and the number of vehicles used compared to more classical taxi-based services or even individual private vehicle transportation-based systems.

5.1 Parameterisable Instance Generator

Google HashCode (Google HashCode, 2018) is perhaps the most popular team-based programming competition in the world, with more than 100,000 programmers participating on each edition. On the qualification round of 2018, a self-driven-based transportation system (a variant of the classical Vehicle Routing Problem with Time Windows) was proposed as a challenge. Its very-large complex instances make it an ideal candidate for testing the performance and scalability of the solution approach described in Section 4. However, although the challenge was also based on a grid-based city and a simulated time horizon, the following features of the ride-sharing problem of Section 3 were not included in the challenge:

- The model does not include a set S of SECs, nor an ownership mapping from E to S . Instead, a single depo at location (0,0) is placed.
- The model does not include dynamic availability of requests T and resources E over time. Instead, all vehicles and trip petitions are available (e_{ri} and t_{ri}) at time unit 0.
- The vehicles have unlimited battery capacity e_{bc} and room for just one passenger e_{pc} .
- The pick-up and drop-off windows of the trips contain t_{ep} and t_{ld} , but not t_{lp} nor t_{ed} .

Therefore, all the above must be incorporated when customising the instances of Google HashCode to the ride-sharing problem. To enable a fine-grained customisation allowing the solution approach to be tested under different configurations, an instance generator is implemented, including:

Num SECs. A configurable integer number of SECs $|S|$ is created across the city, distributed uniformly among its areas. The vehicles E are also distributed uniformly across S , with each SEC_{id} having a same amount of vehicles $|S_E|$.

Energy Generation Factor. Let sum_dist be the sum of the distance from pick-up to drop-off locations for all trips in T . A float value is passed to specify the $total_energy$ generated, collectively, by all SECs as a factor of sum_dist . Thus, if factor is 2.0, then $total_energy = sum_dist * 2$. A full use of

$total_energy$ is assumed, as well as its uniformly distribution among all SECs and vehicles. Therefore, the factor ultimately impacts the battery capacity of all vehicles, setting $e_{bc} = \frac{total_energy}{|E|}$.

Dispatch Mode. A unique energy generation function fs is assumed to be applied to all SECs, with a constant energy generation per time unit. A boolean value is passed to specify such function. If *false* is passed (let's call it *START*), then all vehicles of a SEC are released at time unit 0, and $fs = 0$, producing no further energy afterwards. Otherwise, if *true* is passed (let's call it *SPREAD*), a constant energy production $fs = \frac{e_{bc} \times |S_E|}{th}$ is used, releasing a first vehicle per SEC at time unit 0, and each of the remaining $|S_E| - 1$ vehicles at a constant pace across the time horizon, with one new vehicle each $\frac{th}{|S_E|}$ time units.

Trip Flexibility Factor. As discussed, each trip petition includes t_{ep} and t_{ld} , but not t_{ri} , t_{lp} nor t_{ed} . In the case of t_{ed} , it is simply computed as $t_{ep} + dist$, where $dist$ is the Manhattan distance from pick-up to drop-off location. A float value between 0 and 1 $flex$ is passed, representing the flexibility percentage for computing $t_{ri} = t_{ep} - (t_{ep} * flex)$ and $t_{lp} = t_{ep} + ((t_{ed} - t_{ep}) * flex)$. If flexibility factor tends to 1.0 (i.e., 100% flexible), then t_{ri} tends to 0 (early announcement) and t_{lp} tends to t_{ed} (late pick-up far from early pick-up). In other words, plenty of time to re-route vehicles to accommodate the trip. On the other hand, if the flexibility factor tends towards 0.0 (i.e., 0% flexible), then both t_{ri} and t_{lp} tend to t_{ep} (with very little time to react from the announcement of the trip and a very short time window for its pick-up.)

5.2 Analysis - Performance

The instance *d.metropolis.in* from Google HashCode is selected for its customisation to the ride-sharing problem. The instance is very-large, representing a city of 10,000 x 10,000 distance units, a simulation time horizon of 50,000 time units, 400 vehicles and 10,000 trip petitions. The instance generator is used for the fine-grained customisation of *d.metropolis.in*, leading to a new benchmark of 72 instances, coming from the 72 different combinations of values for *NumSECs*, *EnergyGenerationFactor*, *DispatchMode* and *TripFlexibilityFactor* below:

- **Num SECs:** 1, 4, 16.
- **Energy Generation Factor:** 0.5, 1.0, 2.0.
- **Dispatch Mode:** false (*START*), true (*SPREAD*).
- **Trip Flexibility Factor:** 0.02 (2%), 0.10 (10%), 0.25 (25%), 0.50 (50%).

The 72 instances are run using the solution approach of Section 4 to evaluate the performance of

the algorithm (i.e., the number of trip petitions being allocated to vehicles).

Interestingly, a wide range of results are reported for the 72 instances, from some successful with up to 8,969 trip petitions allocated (an 89,69% of the petitions) to unsuccessful with 0 trip petitions allocated (a 0% of the petitions). The set of results is discussed below, to identify the factors (configurations) having a bigger impact in the performance of the algorithm.

The best configuration is the one with < maximum flexibility, max energy, earlier release of EV > and, given that the number of SECs might have an impact, the more SECs the better.

1. (F - 50%; SECs - 16; ST; EF - 2.0) \equiv 89.69%
2. (F - 50%; SECs - 4; ST; EF - 2.0) \equiv 89.55%
3. (F - 50%; SECs - 1; ST; EF - 2.0) \equiv 82.53%

The next 3 top results have the same configuration as before, but now with energy factor 1.0. Therefore, it seems that the best configuration is the one with < maximum flexibility, earlier release of EV >, and given that, the energy factor does have an impact, with the more energy the better. After that, once again, the number of SECs seems irrelevant.

4. (F - 50%; SECs - 1; ST; EF - 1.0) \equiv 77.69%
5. (F - 50%; SECs - 4; ST; EF - 1.0) \equiv 76.95%
6. (F - 50%; SECs - 16; ST; EF - 1.0) \equiv 76.40%

The next 6 results confirm that, among the duo < maximum flexibility, earlier release of EV >, the earlier release of EVs seems to be the top factor, as all the 11 top results have START release. But, it can also be observed the crucial importance of flexibility, as by reducing it from 50% to 25% the algorithm passes from the previous range of TPs satisfied of [76.40, 89.69] to a reduced performance result of [49.45, 61.84].

7. (F - 25%; SECs - 1; ST; EF - 1.0) \equiv 61.84%
8. (F - 25%; SECs - 1; ST; EF - 2.0) \equiv 61.67%
9. (F - 25%; SECs - 16; ST; EF - 2.0) \equiv 56.26%
10. (F - 25%; SECs - 4; ST; EF - 2.0) \equiv 55.20%
11. (F - 25%; SECs - 16; ST; EF - 1.0) \equiv 51.19%
12. (F - 25%; SECs - 4; ST; EF - 1.0) \equiv 49.45%

The next 8 results confirm that, among the duo < maximum flexibility, earlier release of EV >, the earlier release of EVs seems to be the top factor, followed by the flexibility. If top flexibility is maintained at 50%, but release the EVs over the time horizon, the decrease in total TPs satisfied continues, from the previous range [49.45, 61.84] to a new one of [41.73, 49.09]. If flexibility decreases again back to 25%, then the range decreases even a bit more, to 35.57% of the petitions satisfied.

13. (F - 50%; SECs - 16; SP; EF - 2.0) \equiv 49.09%
14. (F - 50%; SECs - 1; SP; EF - 2.0) \equiv 48.82%
15. (F - 50%; SECs - 4; SP; EF - 2.0) \equiv 47.76%
16. (F - 50%; SECs - 4; SP; EF - 1.0) \equiv 45.02%
17. (F - 50%; SECs - 16; SP; EF - 1.0) \equiv 42.06%
18. (F - 50%; SECs - 1; SP; EF - 1.0) \equiv 41.73%
19. (F - 25%; SECs - 1; SP; EF - 2.0) \equiv 35.86%
20. (F - 25%; SECs - 1; SP; EF - 1.0) \equiv 35.57%

The next 16 results show that, all the above analysis state when there is enough energy being produced. In the case of scarcity of energy, then this turns out to be the most important factor, as the range of TPs satisfied decreases dramatically, to [15.38, 30.31]. That is, if there is a scarcity of energy, the best configuration satisfies the 30.31% of the TPs. However, when looking only at SPREAD, there is a configuration leading to 49.09% of TPs satisfied. Likewise, when looking only at flexibility 25%, there is a configuration leading to 61.84% of TPs satisfied.

21. (F - 50%; SECs - 1; ST; EF - 0.5) \equiv 30.31%
22. (F - 25%; SECs - 1; ST; EF - 0.5) \equiv 29.94%
35. (F - 50%; SECs - 16; SP; EF - 0.5) \equiv 17.07%
36. (F - 25%; SECs - 16; SP; EF - 0.5) \equiv 15.38%

The next 24 results show that, all the above analysis state when there is enough flexibility. If flexibility becomes very low, with the TPs having very little margin (2% or even 10%), then the vast majority of the TPs cannot be satisfied, even with an earlier release of EV and plenty of energy.

37. (F - 10%; SECs - 4; ST; EF - 0.5) \equiv 12.77%
38. (F - 10%; SECs - 16; ST; EF - 0.5) \equiv 12.61%
39. (F - 10%; SECs - 16; SP; EF - 0.5) \equiv 9.95%
58. (F - 10%; SECs - 4; SP; EF - 1.0) \equiv 4.30%
59. (F - 10%; SECs - 16; SP; EF - 2.0) \equiv 3.93%
60. (F - 2%; SECs - 16; SP; EF - 1.0) \equiv 3.93%

Interestingly, the lack of flexibility can even lead to 0 TPs satisfied. This is the case when there is just 1 SEC, placed in the centre of the city. Irrespective of the early or spread release of the EVs, a very tight flexibility in the TPs can make it impossible for an EV to leave from the centre of the city, and reach both the pick-up and drop-off in time. This situation does not happen at all when there are 4 or even 16 SECs, as this the EV is closer to extreme pick-up or drop-off points, making it possible to reach to the destination even with a very tight flexibility.

61. (F - 10%; SECs - 1; ST; EF - 2.0) \equiv 0.06%
72. (F - 2%; SECs - 1; SP; EF - 0.5) \equiv 0.00%

All in all, the analysis above leads to the following conclusions:

- The most important factor is to have enough flexibility in the TPs (of at least 25%).

- This is followed by having enough energy (of at least 1.0 times the one required to accomplish all the TPs).
- With enough flexibility and energy, the most relevant factor is to have earlier release of the EVs.
- On top of that, the more flexibility the better (50% better than 25%).
- If all SECs are uniform in terms of their number of vehicles and energy production, then the number of SECs does not have an impact in the results, except for the extreme cases with very few flexibility and a unique SEC placed in the centre of the city.

5.3 Analysis - Scalability

The same 72 instances of the benchmark of Section 5.2 (referred to as d_{10000_400} due to its 10,000 trip petitions and 400 vehicles) are considered again to evaluate the scalability of the algorithm. Also, smaller versions by factor f of such benchmark are achieved by simply removing $\frac{1}{f}$ trips and vehicles from the instances of the original d_{10000_400} . Table 1 shows the total running time (resp. average running time per instance) in seconds as the size of the instances scale from 1,000 trips and 40 vehicles to 10,000 trips and 400 vehicles. Benchmarks have been run in a Mac OS Ventura, Intel i7 Quad Core 2.3Ghz processor and 16GB RAM memory. The algorithm has been implemented in Python and run with Python 3.10.5.

The results confirm the scalability of the solution approach of Section 4, as it is able to solve very large instances (as the ones of d_{10000_400}) in less than 2 minutes. Together with the previous ones of Section 5.2, it is concluded that the algorithm provides both good performance and scalability.

5.4 NYC Taxi Instances

The data-set (NYC Green Taxi data, 2018) covers the entire fleet of taxi operators in NYC and comprises a total of 1,048,575 trips. The records include fields to capture pick-up and drop-off dates/times, pick-up and drop-off locations and trip distances. The data used were collected and provided by the NYC Taxi and

Table 1: Scalability of the Algorithm.

Benchmark	Total Time	Avg. Time
d_40_1000	40.82	0.54
d_100_2500	221.29	2.91
d_200_5000	933.2	12.28
d_400_10000	9038.0	118.92

Limousine commission by technology providers authorised under the Taxi cab and Livery management enhancements program. The NYC taxi data models a wide array of attributes such as *tip_amount*, *payment_type*, and *fare_amount*. However, for the experiments conducted, the attributes considered are mentioned in Table 2.

The ride-sharing service allows various vital parameters to be dynamically adjusted, as presented in Section 5.1. On top of the configuration described then, the customisation of the NYC taxis uses a fixed battery capacity per vehicle, and then allows to parameterise the total amount of vehicles as a factor of such battery capacity and the total energy being produced.

5.4.1 Grid-Generation

A grid-based mapping approach is used. This approach significantly decreases the time required to analyse data compared to a spatial resolution of near-continuous coverage data (Ramsdale et al., 2017). Grid-based mapping provides a consistent and standardised approach to spatial data collection. A grid map is a set of cells where the users can customise their geometries, and visual representations (Ferreira et al., 2013). An example of a grid map of taxi zones in NYC is shown in Figure 3. To divide the spatial polygon of the NYC map with 262 taxi zones (NYC Taxi Zones, 2023) into a grid, Quantum Geospatial Information System(QGIS) is used. The OGC coordinate reference system (OGC: EPSG7326-WGS84), proposed in the world geodetic system 1984 ensemble (EPSG:7326), is used to form the grids. The average trip distance from the NYC taxi data set is 2.7 miles, with the shortest trip being 0.2 miles. Therefore, the horizontal and vertical spacing is set at 0.01 miles, and a ‘420X556’ grid map is generated. A GeoJSON file is generated for the grid layout of the NYC map. The instance generator computes the pick-up and drop-off locations of passengers based on the ‘*PULocationID*’, the ‘*DOLocationID*’ and *trip_distance* of the passengers from Table 2.

Table 2: NYC Attributes.

Attribute	Eg.Value
<i>!pep_pickup_datetime</i>	01/01/2018 12:18:50 AM
<i>!pep_dropoff_datetime</i>	01/01/2018 12:24:39 AM
<i>PULocationID</i>	236
<i>DOLocationID</i>	236
<i>trip_distance</i>	0.7

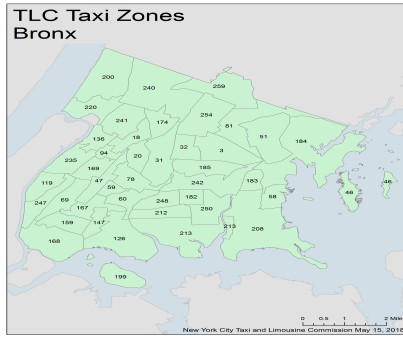


Figure 3: Taxi zones in Bronx, New York (NYC Green Taxi data, 2018).

5.5 Analysis - Ride-Sharing Vehicles and Distance Traversed

Interestingly, when running a large-scale instance for the NYC taxis, the best configuration turns out to be the very same one as reported in Section 5.2. This confirms the configuration observations reported then, in this case observed when applying the ride-sharing service over a real-world data-set:

$$(F - 50\%; SECs - 16; ST; EF - 2.0)$$

This time, the experiments are focused on analysing the performance of the ride-sharing service when compared to both a more classical taxi-based service and to an individual, private vehicle transportation-based service. In this case, both the amount of distance covered and the number of vehicles used are analysed.

- **Taxi Mode:** In this mode, EVs aim to pick-up and drop-off passengers sequentially, as opposed to the ride-sharing mode, where several passengers might, temporarily, share a vehicle during their trips.
- **Individual Mode:** This mode represents a private mode of transportation. Each passenger uses a privately owned EV to commute from the pick-up to the drop-off point.

The overhead of both taxi and ride-sharing modes w.r.t. individual mode is for the extra distance traversed by the EV from the drop-off point to the pick-up point of subsequent requests. Whereas in ride-sharing mode the possibility of having multiple passengers sharing the vehicle might reduce this extra distance, in the case of the taxi mode the extra distance becomes non-avoidable.

Table 3 shows the results when running a large instance with thousands of trip petitions. The ride-sharing service is configured to use 704 vehicles, and it manages to serve 4,514 trip petitions. When compared to individual mode, the use of the ride-sharing

Table 3: Ride-share vs Taxi vs Individual.

Mode	Vehicles	Distance(miles)
Ride-share	704	70360
Taxi	718	71164
Individual	4514	57909

service would have allowed to put $4,514 - 704 = 3,810$ vehicles out of the road, a reduction of an 84.4%.

The algorithm is then re-run for taxi mode. This time it is set to contain: only the 4,514 petitions that were served above; an unlimited amount of vehicles (to ensure that all 4,514 trip petitions are indeed satisfied); and a single passenger capacity on each vehicle (to ensure that no two trip petitions might share a vehicle while being served). The focus is now on the actual number of vehicles used by the algorithm, which turns out to be slightly higher than before (718, an increase of 1.9% w.r.t. ride-sharing mode).

In terms of the overall distance traversed, the ride-sharing mode has an overhead of $70,360 - 57,909 = 12,451$ extra miles, representing an increase of 21.5% in the distance on the individual mode. In the case of taxi-mode, this increase is even higher, confirming the benefits of the ride-sharing mode vs. the taxi simulation also.

6 CONCLUSIONS AND FUTURE WORK

This paper has presented the design, implementation and evaluation of a carbon-neutral, community-based, reactive ride-sharing service for a Smart City with a 100% EVs operating solely over RES.

The ride-sharing service has been formulated as a variant of the classical Dynamic Vehicle Routing Problem with Time Windows, with both dynamic resources and requests. The objective function has been set to maximise the overall number of trip petitions being served. The problem has been solved via a reactive-based simulation approach on top of a greedy-based decision-making process.

An instance generator has been developed to create configurable scenarios to evaluate the proposed approach. It was used to extend existing existing benchmarks (Google HashCode) and public datasets (NYC taxis) to the problem formulation and testing the proposed algorithm under different settings.

The ride-sharing service has proven to scale well, with a quadratic algorithm complexity over its number of trips, allowing to solve an instance of about 1,000 trips in less than a second, and an instance of about 10,000 trips in less than two minutes. Although

favouring scalability over optimality, the service has proven to be able to solve up to 90% of the instances for some configurations of *d.metropolis.in*, a complex instance from the Google HashCode. When applied to the real-world data-set of the NYC taxis and compared to an individual private transportation, the ride-sharing approach has been proven to offer a good trade-off between the amount of vehicles reduced (84%) and its overhead in terms of distance traversed (21%). The complete code can be accessed via GitHub (Nagarajan, 2023).

Future work will include the analysis of the ride-sharing service under diversity on its SECs specification, i.e., diversity in the number of vehicles and energy produced by each SEC. Coupled with the association of each trip petition to a given SEC, this will allow to evaluate the ride-sharing service under unbalanced resources and requests for different SECs. Research will then investigate the possibility of trading services among SECs, ultimately exploring a decentralised solution approach where each SEC competes for serving a trip petition. This decentralised setting will be explored in both reactive and proactive modes, simulating that all trips were known beforehand. Finally, it is intended to integrate the ride-sharing service as part of an existing simulation-tool, as Sumo (Alazzawi et al., 2018).

ACKNOWLEDGEMENTS

This research work is supported by Science Foundation Ireland Centre for Research Training focused on Future Networks and the Internet of Things (AdvanceCRT), under Grant number 18/CRT/6222.

REFERENCES

- Agatz, N. A. H., Erera, A. L., Savelsbergh, M. W. P., and Wang, X. (2012). Optimization for dynamic ride-sharing: A review. *Eur. J. Oper. Res.*, 223(2):295–303.
- Al-Abbasi, A. O., Ghosh, A., and Aggarwal, V. (2019). Deeppool: Distributed model-free algorithm for ride-sharing using deep reinforcement learning. *IEEE Trans. Intell. Transp. Syst.*, 20(12):4714–4727.
- Alazzawi, S., Hummel, M., Kordt, P., Sickenberger, T., Wieseotte, C., and Wohak, O. (2018). Simulating the impact of shared, autonomous vehicles on urban mobility – a case study of milan. In *SUMO'18*, volume 2 of *EPiC Series in Engineering*, pages 94–110. Easy-Chair.
- Diamantoulakis, P. D., Kapinas, V. M., and Karagiannidis, G. K. (2015). Big data analytics for dynamic energy management in smart grids. *Big Data Res.*, 2(3):94–101.
- EU Green Deal (2020). <https://ec.europa.eu/info/strategy/priorities-2019-2024/european-green-deal.en>.
- Ferreira, N., Poco, J., Vo, H. T., Freire, J., and Silva, C. T. (2013). Visual exploration of big spatio-temporal urban data: A study of new york city taxi trips. *IEEE Trans. Vis. Comput. Graph.*, 19(12):2149–2158.
- Furuhata, M., Dessouky, M., Ordóñez, F., Brunet, M.-E., Wang, X., and Koenig, S. (2013). Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B: Methodological*, 57:28–46.
- Google HashCode (2018). <https://codingcompetitions.withgoogle.com/hashcode/archive/2018>.
- Hasan, M. H. and Hentenryck, P. V. (2020). The flexible and real-time commute trip sharing problems. *Constraints An Int. J.*, 25(3-4):160–179.
- Hasan, M. H., Hentenryck, P. V., Budak, C., Chen, J., and Chaudhry, C. (2018). Community-based trip sharing for urban commuting. In *AAAI'18*, pages 6589–6597. AAAI Press.
- Jaillet, P. and Wagner, M. (2008). *Online Vehicle Routing Problems: A Survey*, volume 43, pages 221–237.
- Nagarajan, A. (2023). https://github.com/Nasheor/reactive_rideshare.git.
- NYC Green Taxi data (2018). <https://data.cityofnewyork.us/Transportation/2018-Green-Taxi-Trip-Data/w7fs-fd9i>.
- NYC Taxi Zones (2023). <https://data.cityofnewyork.us/Transportation/NYC-Taxi-Zones/d3c5-ddgc>.
- Ramsdale, J., Balme, M., Conway, S., Gallagher, C., Gasselt, S., Hauber, E., Orgel, C., Séjourné, A., Skinner Jr, J., Costard, F., Johnsson, A., Losiak, A., Reiss, D., Swirad, Z., Kereszturi, A., Smith, I., and Platz, T. (2017). Grid-based mapping: A method for rapidly determining the spatial distributions of small features over very large areas. *Planetary and Space Science*, 140.
- Serra, T., Raghunathan, A. U., Bergman, D., Hooker, J. N., and Kobori, S. (2019). Last-mile scheduling under uncertainty. In *CPAIOR'19*. LNCS 11494, Springer.
- Xiang, Y., Liu, J., Li, R., Li, F., Gu, C., and Tang, S. (2016). Economic planning of electric vehicle charging stations considering traffic constraints and load profile templates. *Applied Energy*, 178:647–659.