



Article

Mobility-as-a-Service Personalised Multi-Modal Multi-Objective Journey Planning with Machine-Learning-Guided Shortest-Path Algorithms

Christopher Bayliss ^{1,*} Djamila Ouelhadj ¹, Nima Dadashzadeh ² and Graham Fletcher ¹

¹ School of Mathematics and Physics, University of Portsmouth, Portsmouth PO1 3DE, UK; djamila.ouelhadj@port.ac.uk (D.O.); graham.fletcher@port.ac.uk (G.F.)

² Future Mobility Centre, Huddersfield Business School, University of Huddersfield, Huddersfield HD1 3DH, UK; n.dadashzadeh@hud.ac.uk

* Correspondence: christopher.bayliss@port.ac.uk

Abstract: Mobility-as-a-service (MaaS) apps provide a single platform for journey planning, booking, payment and ticketing, and are proposed as a medium for encouraging sustainable travel behaviour. Generating sustainable-vehicle-based journey alternatives can be formulated as a multi-modal multi-objective journey-planning problem, one that is known to have a prohibitively large solution space. Building on prior insights, we develop a scalable decomposition-based solution strategy. A Pareto set of journey profiles is generated based on inter-transfer-zone objective criteria contributions. Then, guided by neural-network predictions, extended versions of existing shortest-path algorithms for open and public transport networks are used to optimise the paths and transfers of journey profiles. A novel hybrid k-means and Dijkstra's algorithm is introduced for generating transfer-zone samples while accounting for transport network connectivity. The resulting modularised algorithm knits together and extends the most effective existing shortest-path algorithms using neural networks as a look-ahead mechanism. In experiments based on a large-scale transport network, query response times are shown to be suitable for real-time applications and are found to be independent of transfer-zone sample size, despite smaller transfer-zone samples, leading to higher quality and more diverse Pareto sets of journeys: a win-win scenario.



Academic Editors: João Manuel R. S. Tavares, Teresa Galvão Dias and Marta Campos Ferreira

Received: 10 January 2025

Revised: 10 February 2025

Accepted: 12 February 2025

Published: 15 February 2025

Citation: Bayliss, C.; Ouelhadj, D.; Dadashzadeh, N.; Fletcher, G. Mobility-as-a-Service Personalised Multi-Modal Multi-Objective Journey Planning with Machine-Learning-Guided Shortest-Path Algorithms. *Appl. Sci.* **2025**, *15*, 2052. <https://doi.org/10.3390/app15042052>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Mobility-as-a-service (MaaS) is concerned with personal transportation needs in metropolitan areas [1,2]. It aims to make alternatives to private car journeys more easily accessible [3–5]. To do this, the offerings of mobility service providers (MSPs) are brought together within a single convenient platform, usually a mobile phone application, to provide integrated journey planning, booking, payment and e-ticket functionality. As outlined by [6], a multiple MSP journey planner provides the most basic functionality of any MaaS app, since booking, payment, ticketing, subscriptions, bundles and behaviour-nudging incentives can all be built on top of it.

This work tackles a multi-modal multi-objective journey-planning problem involving four distinct non-private car classes of transport modes: (i) *public transport*, such as bus, train and ferry services; (ii) *taxis services*; (iii) *micro-mobility modes*, such as hire bikes and

e-scooters; (iv) *personal transport*, such as walking and cycling. The minimisation objectives considered (and their reasons for inclusion) include the following: (i) *cost* (people prefer to pay less); (ii) *arrival time* (people typically prefer faster journeys); (iii) *CO₂ emissions* (environmentally conscious people prefer green journeys); (iv) *inconvenience* (measured as the total amount of time spent walking and waiting in a journey, which people typically prefer to avoid if possible); (v) *calorie expenditure* (some people may like to minimise the required physical exertion of their journey); (vi) *transfers* (people tend to prefer making fewer connections for convenience and reduced risk of missing connections). For any given commuter query, consisting of a departure time, origin and destination, this objective function characterises a Pareto set of journeys that are mutually non-dominated with respect to these objectives. Such a set of journeys includes the cheapest, fastest, greenest, most convenient and least strenuous. In addition, the Pareto set includes those which balance conflicting objectives. Such a set of journeys may include those attractive to private car users and prompt travel behaviour change. The problem we face is that of being able to generate a Pareto set of multi-modal journeys rapidly in response to commuter queries received through a MaaS app in real-time. A mobile MaaS application provides the means by which commuter queries are made and the query results displayed to commuters. The calculation of multi-modal routes takes place server-side, which permits the use of algorithms, such as the one presented in this work, that rely on pre-processing and on-going background update steps in order to aid a rapid response to commuter queries.

In previous work [7], we introduced the multi-modal multi-objective journey-planning problem considered in this work. In contrast to other streams of research related to multi-modal journey planning, the problem considered here is more general. It considers a wider set of transport modes, including public transport, walking, taxis, bikes and also new shared micro-mobility modes such as rentable e-scooters. Previous works have tended to focus only on cars, walking and public transport [8–11], while also placing constraints on the orders in which modes can be used in journeys, which is a solution-space-reduction approach that is avoided in this work. Ref. [8] proposed an access-node approach where the main parts of journeys are assigned to the public transport network, thereby reducing the size of the path-planning problems to and from the public transport access-nodes. Ref. [9] proposed the Round-Based Public Transit Routing (RAPTOR) algorithm, which is a dynamic programming-based approach for minimising both travel time and number of transfers via walking and public transport networks. Each iteration attempts to find faster journeys to each reachable stop involving one additional transfer. Ref. [10] proposed an A* [12] style shortest-path journey-planning algorithm (which is a goal-directed version of Dijkstra's shortest-path algorithm [13]) that only allows car usage in the first legs of journeys, as in park-and-ride journeys. Ref. [11] required journey-mode sequences as a commuter input constraint in their multi-modal travel-time minimisation approach. The problem considered here considers a unique objective function, that of the simultaneous minimisation of cost, travel time, inconvenience (measured as the sum of walking and waiting time), CO₂ emissions and calories expended. Prior to this, the most general objective function considered was that of [8], who considered cost in addition to travel time and number of transfers. Even for this objective function, acceptable query response times could not be attained without solution-space-reduction-based heuristics, by imposing structural constraints on journeys including mode-sequences. This work considers mode choice and their sequence a commuter choice issue, and the challenge addressed in this work is to rapidly generate all of the worthy alternatives.

Regarding how this work extends that of [7], the focus of the previous work was an investigation of exact and heuristic solution methodologies set within small to moderate sized problem instances such that optimal solutions could still be computed within a

reasonable amount of time. Time complexity analyses and computational results led to the conclusion that, for this new problem, the number of transport network nodes that are considered as potential multi-modal transfer points is a huge and computationally prohibitive factor influencing query response times. It was recommended that future work should investigate algorithms based on a transfer-point-sample-approach, a strategy that should most effectively reduce the size of the solution space. This work builds on the insights of [7] and proposes a methodology based on such a strategy.

The Pareto set generation process of the multi-modal multi-objective journey-planning problem is decomposed into two steps. Firstly a Pareto set of journey profiles is generated, where a journey profile is defined as the combination of a *mode sequence* and corresponding *transfer-zone sequence*. That is, solution space is reduced via a transfer-zone sampling approach. Secondly, the exact transfer locations within the transfer zones of the journey profiles are then optimised by a procedure guided by machine-learning travel-time predictions. The proposed approach is referred to as ML-TZSA, reflecting its use of machine learning and transfer-zone sampling. Figure 1 depicts the four main steps of ML-TZSA: (i) *Transfer-zone sampling*, in which the transport networks of interest are divided into a set of transfer zones (Section 4.1); (ii) *Inter-transfer-zone objective criteria contributions*, in which objective value contributions corresponding to single transport mode trips between each pair of transfer zones are calculated (Section 4.2); (iii) *Journey profile Pareto set generation*, in which a Pareto set of mutually non-dominated journey profiles, for a real-time query, is generated based on the inter-transfer-zone objective criteria contributions (Section 4.3); (iv) *Transfer-zone constrained transfer point optimisation*, in which the transfer points and paths of the journey profiles are optimised in a procedure guided by machine-learning travel-time predictions (Section 5). Phases (i) and (ii) are offline pre-processing tasks that prepare the algorithm to be able to handle any query between any pair of locations within the input transport network, while phases (ii) and (iv) are online query response tasks.

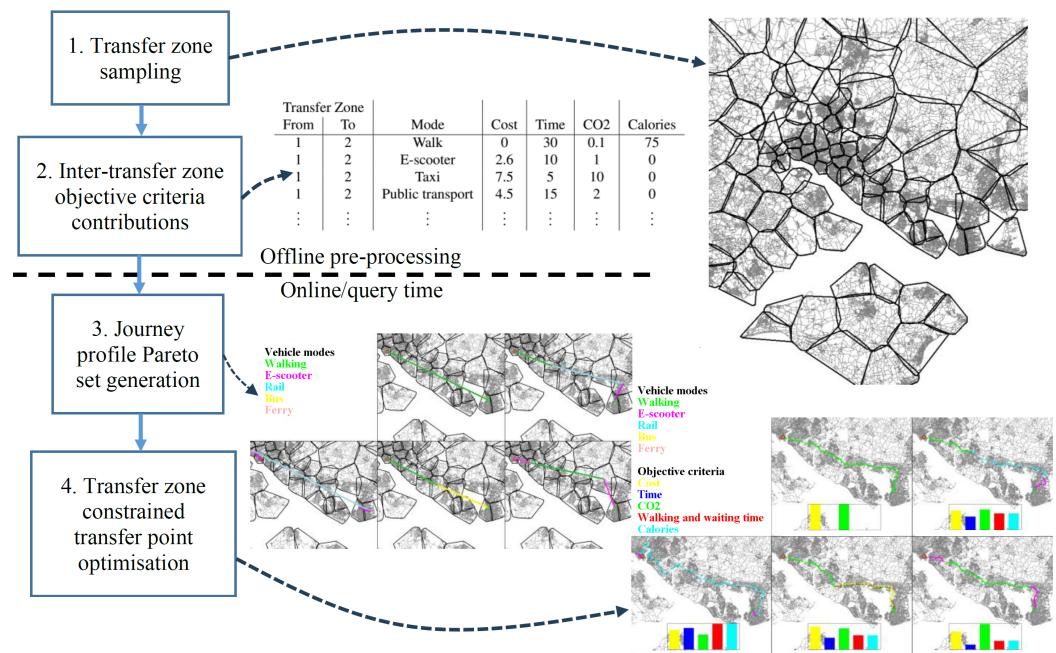


Figure 1. Overview of ML-TZSA.

On the whole, this approach side steps the issue of the very large number of possible sequences of transfer locations without ruling any out. While the number of transfer zones generated limits the size of the Pareto set of journey profiles that can be generated, this is justified by considering that, in theory, there can be an infinite number of optimal

non-dominated solutions. For instance, consider the case of two objectives, cost and time minimisation, and two modes, walking and taxi. In this case, an infinitely sized Pareto set can be created by generating the set of journeys, including those where the commuter gets out of the taxi at all points along the taxi route and walks the remainder of the journey. This is because, in such a case, journey cost and travel time are conflicting objectives. Considering this, limiting the set of possible transfer points in some way makes practical sense as a solution space pruning method. Also, it is more practical to present only a limited number of journey alternatives to commuters.

The contributions of this work are as follows: A transfer-zone sample approach is proposed and developed for greatly reducing the size of a very large journey-planning solution space without adversely effecting solution quality. A novel pedestrian transport network-based hybrid Dijkstra's algorithm and k-means clustering algorithm is introduced for selecting the sample of transfer zones—an approach which guarantees the feasibility of transfers of journeys planned via the sampled transfer zones. Existing shortest-path algorithms for open and public transport networks are adapted to the many-to-many case with numerous speed-up techniques and are used in conjunction with a machine-learning algorithm to optimise the transfer points and paths between the transfer zones of journey profiles. The new method has the beneficial feature that larger transfer-zone samples lead to large and better-quality Pareto sets of journeys without increasing query response times. Our experiment results show that our approach can generate large diverse sets of efficient journey options within seconds. Such large sets of journeys include cheap, fast, green, convenient and low effort ones which, when presented within a MaaS app, may encourage more people to opt for non-private car journeys and thereby reduce pollution and congestion.

The remainder of this paper is structured as follows. Section 2 provides a review of the relevant literature. Section 3 defines the problem addressed in this work. Sections 4 and 5 detail the proposed solution methodology. Section 6 provides experiment results evaluating the effectiveness of the proposed approach. Section 7 summarises the main findings and promising directions for future work.

2. Literature Review

This section explores the existing literature regarding MaaS in general, considering its aims and potential benefits, followed by a review of existing approaches for multi-modal journey planning.

2.1. MaaS Concept

MaaS, in general, aims to integrate the offerings of mobility service providers into a single digital platform for journey planning, booking, payment and (e-)ticketing [6,14]. They state that mobility service integration can be broken down into four hierarchical levels: (1) journey planning (which is the most basic function of a MaaS app and the theme of this work); (2) purchase and ticketing; (3) bundles and subscription plans; (4) addressing societal goals such as reducing emissions and congestion using behaviour-nudging incentives. MaaS apps can also provide live routing navigation once a journey is begun, which helps to ensure that the most efficient routes are followed. Ref. [15] shows that people do not always opt for the shortest paths without the aid of navigation.

A central goal of MaaS is to reduce high private car dependency [16]. Ref. [17] outlines a research agenda for MaaS enablement, highlighting that a move away from private car ownership comes with a natural increase in demand for convenient and seamless shared mobility offerings. According to [18], the convenience benefits of MaaS include smart ticketing, which also provides an institutional coordination mechanism for facilitating

cooperation. MaaS is also touted as a means for improving social equity. Ref. [19] proposed the MaaSINI: a MaaS inclusion index to evaluate the inclusivity and accessibility of MaaS considering vulnerable social groups' needs, such as the elderly, people with disabilities and low-income people.

Ref. [20] attempts to answer the question of whether or not MaaS can change user travel behaviour and deliver commercial and societal benefits. It is reported that MaaS may help to reduce car ownership, but this may be offset by an increase in the use of other car-based modes, such as taxi services or Uber. Evidence was found that suggests that investment in MaaS does bring about net reductions in CO₂. Regarding the demographic groups that are most likely to adopt MaaS as their alternative to private car ownership, Ref. [21] performed a latent class cluster analysis of attitudes and found that the largest cluster is most likely to adopt MaaS, while current uni-modal car users are least likely to adopt MaaS. This suggests that the natural bias against car-based transport modes of many MaaS trials may be counter-productive to the aims of the MaaS movement. Instead, by embracing car-sharing and taxi services in MaaS apps, current uni-modal car users may be more inclined to explore the many benefits of MSP integration through MaaS apps. Emerging convenient demand responsive bus services [22] may also sway private-car users and/or those undergoing life changes.

The success of MaaS trials relies on different aspects, such as business models and how MaaS can match MSP offers with a traveller's needs. Ref. [23] evaluated different business models and structures in the context of MaaS and concluded that it is important to consider different market structures since, in certain cases like intermediary settings, prices tend to be significantly higher with low profitability. Recently, Ref. [24] developed a novel mechanism for MaaS to solve matching and pricing problems between MSPs and travellers. They presented an initial approach that utilised a static offline mechanism employing a pricing scheme derived from the Vickrey–Clarke–Groves model. The proposed mechanism is designed to prioritise incentive compatibility, individual rationality and system efficiency. Additionally, they introduced an online mechanism that employs a dynamic learning algorithm to generate a solution that closely approximates optimality. A customised greedy-based algorithm is used for comparison alongside this online approach. Ref. [25] also developed incentive-compatible mechanisms for the online mobility-resource-allocation problem. The proposed polynomial-time online algorithm benefits users with the possibility to accept or reject offered MaaS bundles by comparing the associated utility obtained from MaaS with a reserve utility obtained from other travel options.

Ref. [26] explored possible futures for MaaS by investigating the concept of mode efficiency, where existing and emerging transport modes are categorised in terms of spatial (vehicle size and average passenger numbers) and temporal (time spent idle) dimensions. They warn that the promotion of some emerging modes can have adverse effects on the various externalities for which MaaS is sold as addressing. Along a similar line of research, ref. [27] found evidence suggesting that the introduction of car-sharing and bike-sharing (micro-mobility) as alternatives to public transport in low-density areas can improve energy efficiency. Taken together, mode efficiency, geography and demographics are key elements that should be accounted for in the provision of mobility services.

2.2. Multi-Modal Route Planning

In general terms, route-planning problems come in two main varieties: vehicle-routing problems (see [28] for a recent survey and classification scheme) and journey-planning problems. This work is concerned with the latter. Journey-planning problems themselves come in many varieties, including itinerary planning, where journeys must visit a specified

sequence of intermediate locations/transfer points [29], and shortest-path problems, which are the focus in this work.

Over the years, shortest-path-planning problems have been extended in a number of ways, including multiple objectives [30–35], transport modes [36–39] and network characteristics [40,41]. For example, Ref. [32] introduced a new exact-label-setting algorithm that returns the subset of Pareto optimal paths that satisfy a set of lexicographic goals, or a subset that minimises deviation from goals if these cannot be fully satisfied. Ref. [39] presented an algorithm that considers the price optimisation (including driving and parking costs) as a Pareto criterion in addition to travel time and the number of transfers for multi-modal two-way round trip journeys.

The term *multi-modal* route planning refers to path-planning problems where transfers between different vehicles and types of vehicles are allowed en-route. This term has often been used in cases where the available modes are all fixed-schedule transport modes [29,42,43]. More recently, *multi-modal* is most often used to refer to cases where the available modes also include what might be referred to as open-network modes, such as cars and bicycles, in addition to fixed-schedule modes, such as public transport services.

Research into multi-modal journey planning has been becoming more and more general regarding the available transport modes, the objectives considered and the rules imposed on when different modes can be used and in what order. Ref. [8] proposed a fast algorithm for a special case of a multi-modal shortest-path problem in which the starts and ends of journeys use the road network, while the main part of the journey utilises public transport. Distances to and from public transport access-nodes are pre-computed and stored, meaning that, at query time, only the smaller public transport network is subject to a path search. In this work, pre-processing tasks are also employed to reduce query times; see Section 4.2. This work focuses on the development of a general multi-modal and multi-objective journey-planning algorithm, without imposing rules regarding allowable mode sequences. Previous works have focused on other special case problems.

Ref. [44] developed a trip-planning algorithm that extends a time-minimised vehicle trip-planning system by integrating it with multi-criteria walking considerations. They argue that people are sensitive to walking for a great many reasons, including, exercise, physical capabilities, path gradients and environmental conditions at travel time. Ref. [9] introduced the RAPTOR algorithm for generating public transport and walking journeys that minimise travel time and number of transfers; it is often the case that journeys can be completed faster if commuters are willing to perform more transfers and walk more, since public transport timetables are not always conveniently synchronised with each other. More recently, ref. [10] proposed a *time-dependent inter-modal A** (TDIMA*) algorithm, which combines scheduled transport services, walking and vehicular transport networks. The objective was to minimise a weighted sum of travel time, waiting time and penalties for transfers between modes.

The general trend towards rolling out the efficiencies brought about by digitisation, internet and instant telecommunications is making route-planning algorithm functionality available to an ever expanding portion of the population, often via MaaS apps. While the basic feature of an MaaS app is that of route planning, MaaS apps can also be used as navigation aids to inform commuters of real-time hazards or delays. For example, Ref. [45] proposed a framework for an integrated multi-modal route planner in the MaaS domain. The architecture consists of two components: a dynamic journey planner that combines routes from existing planners, enhances them with innovative mobility services and transforms them into multi-modal options, and a route recommender system that filters and ranks routes based on travellers' preferences and MaaS operator requirements, such as environmental sustainability or the promotion of specific modes of transportation.

Relevant methodologies include the following. Ref. [46] propose a methodology for estimating traffic flows using historic and real-time sensor data, information that is fed into OpenTripPlanner (OTP). Ref. [47] implemented a similar approach in which real-time congestion and air quality information are integrated with the multi-modal journey-planning platform GraphHopper. Another example can be found in [48], who proposed a dynamic programming route-planning algorithm for avoiding epidemic hot spots. Ref. [49] developed a journey-planning app that combines public transport and car-pooling, called RideMyRoute, which relaxes the origin and destination constraints of purely car-pooling journeys, since the user can adjust the start and/or end of their journey by making use of public transport, thereby increasing the number of feasible car-pooling options.

Regarding the users' perspective, Ref. [50] provided 95 travellers with a multi-modal trip planner and found that participants had positive attitudes toward the trip planner as it proposes the optimum route, enhancing their comfort and reducing their stress while travelling.

3. Multi-Modal Multi-Objective Journey-Planning Optimisation Model

The multi-modal multi-objective journey-planning problem has two main sets of inputs, a graph $G = (N, E)$, which defines the nodes N (junctions) and edges E (connecting paths) of the non-public transport networks, and general transit feed specification (GTFS) data, which defines the public-transport network structure, including its routes and timetables. The graph G can be broken down further for each non-public transport mode, where $G^p = (N^p, E^p)$ denotes the pedestrian transport network, $G^b = (N^b, E^b)$ denotes the bike transport network, $G^s = (N^s, E^s)$ denotes the e-scooter network (usually the same as the bike network) and $G^c = (N^c, E^c)$ denotes the road-vehicle transport network. All nodes, and public transport stops, have a longitude and latitude, which are used to identify the nodes/stops directly connected to the nodes/stops in every other transport network, which define the set of possible transport mode transfer points. Regarding the GTFS data framework, the public transport network is composed of a set of *routes*, which define the stop sequences of each route. Each route has a set of *trips*, which correspond to the set of times each route is repeated. Each trip has a set of *stop times*, which define the stop timetable of that trip. GTFS data includes *calendar* data, which specifies the dates on which each route operates, *fare* data, specifying the costs associated with possible sub-trips, and *shape* data, specifying the exact paths of each route.

Our objective is to return, for any given commuter query (origin O , destination D , departure time α), the set of non-dominated journeys with respect to simultaneous minimisation of the objectives of (i) cost, (ii) travel time, (iii) CO₂ emissions, (iv) inconvenience (measured as the sum of walking and waiting time) and (v) calories used. Table 1 specifies the characteristics and cost structures associated with each transport mode. Travel times and waiting times of journeys are calculated from public transport timetable data and edge traversal times. CO₂ is calculated based on the distance travelled via each transport mode, while calories used are calculated based on the distances walked and cycled.

Table 1. Vehicle parameters.

	Pedestrian	E-Scooter	Taxi	Public Transport
Fixed cost (GBP)	0	1	2.5	0
Daily cost (GBP)	0	0	0	4.5
Distance cost (GBP/m)	0	0	0.00125	0
Time cost (GBP/s)	0	0.0025	0	0
Response time (s)	0	120	300	N/A
Maximum speed (m/s)	1.111	3.89	31.29	N/A
CO ₂ (g/m)	0.00011	0.007	0.12	0.0411
Calories (kCal/m)	0.06	0	0	0

While the number of transfers is, in principle, an element of inconvenience, in this work it is treated as a commuter input constraint and not an objective. The main reason for this is twofold: (i) a general investigation of multi-modal journey planning requires the consideration of transfers; (ii) transfers are the primary source of the problem's complexity that we are trying to address. More generally speaking, we avoid suppressing transfers directly for analysis purposes. We note that our objective function and corresponding solution techniques extend trivially to the case where transfers are an additional minimisation objective.

For the purpose of explaining our objective function, let y^k denote a vector defining a single (k^{th}) multi-modal journey from the origin to the destination and let $f_i(y^k)$ denote the objective value of journey y^k on objective criterion i . We are interested in finding the optimal set of mutually non-dominated journeys with respect to criteria (i) to (v). Objective (1) specifies that we want to find a Pareto set of journeys P which simultaneously minimises all objective criteria. Constraints (2) and (3) define the mutually non-dominated solutions requirement of a Pareto set. Specifically, Constraint (2) specifies that the Pareto set must not contain any solutions dominated on all criteria by any other solution in the set. Constraint (3) specifies that each solution in the Pareto set must outperform at least one other non-dominated solution on at least one objective criteria.

$$\min : f_i(y^k), \forall i \in F, \forall y^k \in P, \quad (1)$$

s.t.

$$\forall y^k \in P \nexists \left(\left(y^l \in P \right) \mid \left(f_i(y^k) < f_i(y^l), \forall i \in F \right) \right), \quad (2)$$

$$\forall y^k \in P \exists \left(\left(y^l \in P \wedge i \in F \right) \mid \left(f_i(y^k) > f_i(y^l) \right) \right). \quad (3)$$

Formally, we can characterise the validity of any given multi-modal journey in terms of a time-expanded graph model, where the non-public transport networks have copies of each edge corresponding to departures at each possible time interval t (e.g., E^{pt} for the set of pedestrian edges departing at time interval t). On the other hand, the public transport network has edges corresponding to each scheduled trip between each pair of consecutive stops in all scheduled trips. Let x_{vtij}^k denote a binary variable indicating whether or not transport mode v is used to travel from node/stop i to node/stop j departing at time t in journey k .

A feasible journey must be spatially contiguous (Constraint 4). Specifically, Constraint 4 states that each non-origin and non-destination node entered must also be exited. Note that, for the case of connections between transport modes, the edge set E^{vt} includes zero length edges to nodes in the same position but in different transport networks. Journeys begin at the origin (Constraint 5) and end at the destination (Constraint 6).

$$\sum_{v \in V} \sum_{t \in T} \sum_{(i,j) \in E^{vt} | i \neq O} x_{vtij}^k = \sum_{v \in V} \sum_{t \in T} \sum_{(j,l) \in E^{vt} | l \neq D} x_{vtjl}^k, \forall j \in N \setminus O \cup D, \forall y^k \in P, \quad (4)$$

$$\sum_{v \in V} \sum_{t \in T} \sum_{j \in N^{vt}} x_{vtOj}^k = 1, \forall y^k \in P, \quad (5)$$

$$\sum_{v \in V} \sum_{t \in T} \sum_{j \in N^{vt}} x_{vtjD}^k = 1, \forall y^k \in P, \quad (6)$$

Constraint 7 characterises time-feasible paths within this time-space network-graph-based formulation; specifically, nodes can only be exited at a time after they have been entered. τ_{vtij} denotes the travel time between nodes i and j while using transport network v departing at time interval t . d_t is the time associated with time interval t . Additionally, the journey may not begin before the earliest departure time α (Constraint 8).

$$\sum_{v \in V} \sum_{t \in T} \sum_{(i,j) \in E^{vt} | i \neq O} d_t x_{vtij}^k + \tau_{vtij} \leq \sum_{v \in V} \sum_{t \in T} \sum_{(j,l) \in E^{vt} | l \neq D} d_t x_{vtjl}^k, \forall j \in L \setminus O \cup D, \forall y^k \in P, \quad (7)$$

$$\sum_{j \in L} \sum_{v \in V} \sum_{t \in T} d_t x_{vtOj}^k \geq \alpha, \forall y^k \in P, \quad (8)$$

Commuter queries might also include budget, latest arrival time, maximum transfers and unacceptable transport mode constraints. In this work, such constraints are to be treated as commuter preferences, which they apply to filter journey options output from the proposed approach. Note that they could also be trivially addressed during the optimisation stage by pruning infeasible solutions as soon as they emerge.

4. Journey Profile Pareto Set Generation

In this work, we propose a methodology for addressing the main computational bottleneck of multi-modal multi-objective journey planning, that of the very large number of possible sequences of transfer locations, which can be between any pair of transport modes. We circumvent this bottleneck in a two-stage approach:

1. A Pareto set of multi-modal journey profiles is generated utilising a carefully selected sample of transfer zones.
2. Optimisation of the transfer points within the transfer zones, and the exact paths between them, of the journey profiles.

We propose a hybrid k-means clustering algorithm and Dijkstra's shortest-path algorithm for generating transfer zones, each consisting of inter-connected transport network nodes. Once the sample of transfer zones has been selected, static objective criteria contributions are pre-computed for each inter-transfer-zone trip via each transport mode. Then, at query time, an implicit enumeration algorithm (stage 1) is used to generate a Pareto set of journey profiles that are mutually non-dominated with respect to the minimisation of cost, arrival time, CO₂, inconvenience and calorie expenditure, where a journey profile is defined as the combination of a transport mode sequence and a corresponding transfer-zone sequence. Then, finally, the transfer points within the transfer zones and paths between them are optimised (stage 2) with respect to real-time transport network data.

4.1. Transfer-Zone Sampling (Offline)

The sample of transfer zones is selected based on applying a clustering procedure to the pedestrian transport network, while ensuring that each cluster/transfer zone forms a

connected pedestrian node sub-graph, which in turn ensures that transfers planned within transfer zones can be completed on foot.

Algorithm 1 outlines the hybrid Dijkstra's algorithm and k-means clustering algorithm procedure. The algorithm starts with randomly selected pedestrian network nodes as the seed nodes for building each cluster, while ensuring that at least one node is selected from each sub-graph (island) of the entire transport network. The sub-graphs and their number can be identified by applying the minimum spanning tree algorithm, starting a new tree/sub-graph whenever no more unconnected nodes can be added to the current tree. The relative sizes of the entire network's sub-graphs are used to determine proportional numbers of initial centroids to select for each sub-graph.

Algorithm 1 Hybrid Dijkstra's algorithm and k-means clustering algorithm for pedestrian transport network structure informed transfer-zone sampling.

```

1: Inputs: transferZoneSampleSize, iterations (number of k-means iterations),  $N^p$  (pedestrian nodes),  $E^p$  (pedestrian edges).
2: //Identify pedestrian node sub-graphs (islands).
3: //For each transfer cluster, initialise a priority list of nodes, each containing a random pedestrian node, ensuring that a pedestrian node from each sub-graph is selected, and initialise a list of priority lists containing all of the priority lists.
4: for  $i \in \{1..transferZoneSampleSize\}$  do
5:    $priorityList_i \leftarrow randomPedestrianNode()$ 
6:    $listOfPriorityLists.add(priorityList_i)$ 
7: end for
8: //Apply iterations of the pedestrian node-based clustering algorithm.
9: for  $j \in \{1..iterations\}$  do
10:    $unassignedNodes \leftarrow pedestrianNodes$ 
11:   while  $|unassignedNodes| > 0$  do
12:     //The next node to scan is that from the priority list with the non-scanned node with the shortest path time from the seed node of that priority list.
13:      $currentPriorityList \leftarrow listOfPriorityLists[0]$ 
14:      $nodeToScan \leftarrow removeAndReturn(currentPriorityList, 0)$ 
15:     //Add nodeToScan to the cluster corresponding to the current top-ranked priority list and remove from unassignedNodes.
16:     //Scan the node (fundamental step from Dijkstra's algorithm)
17:     for each node connected to an edge (edge) originating at nodeToScan do
18:       if node not assigned to a cluster then
19:         node.timeFromSeedNode = nodeToScan.timeFromSeedNode + edge.time
20:         // Add node to currentPriorityList and sort by increasing timeFromSeedNode.
21:       end if
22:     end for
23:     //Sort the priority lists of listOfPriorityLists by their minimum times from seed node.
24:   end while
25:   //
26:   if  $j < iterations$  then
27:     //Find the node in each cluster nearest to the new cluster centroids and set these as the seed nodes of each cluster priority list ready for the next iteration.
28:   end if
29: end for
30: Output: The cluster assignments of all pedestrian network nodes.

```

In contrast to a standard implementation of k-means clustering, Dijkstra's algorithm is used to determine distances to nearest cluster centroids via shortest paths in the pedestrian network. Algorithm 1 outlines an efficient approach for running a set of Dijkstra's algorithms, one for each cluster, simultaneously, which avoids having to implement Dijkstra's

algorithm in full starting from each cluster centroid individually, by avoiding “scanning” any node more than once in total during each k-means iteration. This is achieved by juggling the priority lists of each cluster’s Dijkstra’s algorithm instance, by always considering next the priority list whose first-ranked node has the minimum path length. The resulting assignments of nodes to nearest clusters reflects actual travel times in the pedestrian network, i.e., actual network topology, as opposed to “as the crow flies” distances. As in k-means clustering, cluster centroids form the cluster seed nodes for the next iteration.

Figure 2 shows the result of applying Algorithm 1 to the pedestrian network of the Solent area, for a transfer-zone sample size of 500. The identified transfer zones are highlighted via different colours and their convex hulls. It can be observed that the resulting zones capture geographical features and also reflect pedestrian node density.

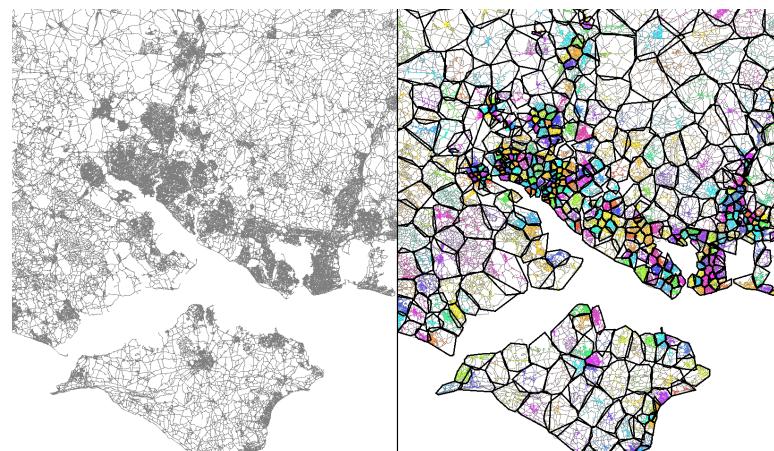


Figure 2. Transfer zones generated by Algorithm 1 for a transfer-zone sample size of 500.

4.2. Inter-Transfer-Zone Objective Contributions (Offline)

This section outlines how objective criteria contributions are estimated for each journey leg between each pair of transfer zones via each transport mode.

For the case of open transport networks, the method is as follows. For each transfer zone, each node in that zone is set as an origin node with an initial arrival time of 0. Dijkstra’s algorithm is executed to find all of the shortest paths from the origin transfer zone to each node in every other transfer zone. The average times and distances to every other transfer zone are used to calculate the required objective criteria contributions. Let $time(i, j, m)$, $distance(i, j, m)$ and $speed(i, j, m)$ denote the average shortest path times, distances and speeds, respectively, from transfer zone i to transfer zone j via transport mode m . Letting T denote the set of transfer zones and O the set of open transport networks, the open-network inter-transfer-zone objective criteria contributions are calculated as follows.

$$cost(i, j, m) = \left(\begin{array}{l} (time(i, j, m) \times timeCost(m, speed(i, j, m))) \\ + (distance(i, j, m) \times distanceCost(m, speed(i, j, m))) \\ + fixedCostPerUse(m) \\ + fixedCostPerDay(m) \\ + tripCost(i, j, m) \end{array} \right), \forall (i, j) \in T^2 | i \neq j, \forall m \in O, \quad (9)$$

$$time(i, j, m) = time(i, j, m) + responseTime(m), \forall (i, j) \in T^2 | i \neq j, \forall m \in O, \quad (10)$$

$$CO_2(i, j, m) = distance(i, j, m) \times distanceCO_2(m, speed(i, j, m)), \forall (i, j) \in T^2 | i \neq j, \forall m \in O, \quad (11)$$

$$inconvenience(i, j, pedestrian) = time(i, j, pedestrian), \forall (i, j) \in T^2 | i \neq j, \quad (12)$$

$$\text{Calories}(i, j, m) = \text{distance}(i, j, m) \times \text{distanceCalories}(m, \text{speed}(i, j, m)), \forall (i, j) \in T^2 | i \neq j, \forall m \in O, \quad (13)$$

where $\text{timeCost}(m, s)$ is the cost per unit time while using transport mode m at an average speed of s . $\text{distanceCost}(m, s)$ is the cost per unit distance while using transport mode m at an average speed of s . $\text{fixedCostPerUse}(m)$ is the fixed cost per usage of transport mode m . $\text{responseTime}(m)$ is the response time/set up time to begin using transport mode m . $\text{distanceCO}_2(m, s)$ is the CO₂ emission rate per unit distance while using transport mode m at an average speed of s . $\text{distanceCalories}(m, s)$ is the calories used per unit distance while using transport mode m at an average speed of s . $\text{fixedCostPerDay}(m)$ is the fixed daily cost associated with transport mode m . Finally, $\text{tripCost}(i, j, m)$ is the average fare associated with a public transport trip between transfer zones i and j via transport mode m . In this work, we assume a fixed daily cost model for public transport usage; see Section 6.1.

The pre-calculated inter-transfer-zone objective contributions are based on static estimate average speeds of each vehicle type on each edge. Typically, transport network travel-time data come in two varieties, daily averages and real-time data. Once Pareto sets of journey profiles have been generated based on static estimates, the transfer point optimisation procedure that follows is based on real-time speed estimates accounting for current traffic conditions and disruptions. Similarly, the pre-calculated inter-transfer-zone objective contributions for public transport journey legs are based on the scheduled public transport services for the given day, and real-time disruptions can be accounted for in the transfer point optimisation procedure. Note that the inter-transfer-zone objective criteria contributions can also be updated in response to real-time changes in traffic conditions and disruptions in an ongoing update procedure. For simpler exposition purposes, the pre-processing step is described here as an offline process relative to the query response aspect of the proposed scheme.

The public transport inter-transfer-zone objective contributions are calculated in a similar way as those for the open transport networks; however, all of the public transport trip paths are pre-defined. For each transfer zone, average journey times, distances and speeds, to every other transfer zone, are calculated based on all scheduled public transport trips (pairs of origin and destination stops) for the same day. Then, Equations (9)–(13) are used to calculate expected inter-transfer-zone objective criteria contributions.

For the case of journey profiles with consecutive public transport legs, we account for minimum possible connection times. For each transfer zone, we identify the minimum connection times required to access the public transport services that lead to each of the other transfer zones. The intention is to generate optimistic journey profiles that the subsequent transfer point optimisation procedure can use to generate the best possible multi-modal journeys. Some journey profiles may turn out to lead to low-quality journeys due to time-dependent features that could not be accounted for based on static estimates regarding inter-transfer-zone journeys. However, many journey profiles are to be generated, so we should still be left with many options that can be used to generate a final set of fully defined mutually non-dominated journeys, making the overall approach fairly fail-safe.

4.3. Implicit Enumeration of Non-Dominated Journey Profiles (Online)

At query time, a multi-modal path tree of non-dominated journey profiles is built from the origin transfer zone to all other transfer zones. The tree is built in increasing number of transfers order. Iteration 1 generates journeys from the origin transfer zone to every other transfer zone via each available transport mode. Iteration 2 extends the journeys with journey legs to every other transfer zone via each available transport mode, while at the same time maintaining lists of non-dominated journey profiles to each transfer

zone. Subsequent iterations are analogous while involving one extra transfer built on top of those from the previous iteration. As a speed-up technique, we apply *target pruning* [9]. Whenever considering the extension of a partial journey profile with a journey leg to a non-destination transfer zone, we eliminate the new partial profile if it is dominated by any profile to the destination transfer zone. The effectiveness of *target pruning* can be maximised by generating all of the new non-dominated journey profiles to the destination transfer zone first at the beginning of each transfer-iteration. This maximises the quality of the destination transfer zone's Pareto set as early as possible and increases the total amount of pruning, which in turn reduces the number of journey profile extensions to be considered in subsequent transfer-iterations. For the case that the maximum number of transfers is predefined, the final transfer iteration need only consider extensions leading to the destination transfer zone, which is useful because, generally, each subsequent transfer-iteration will involve the consideration of more-and-more partial journey profiles to extend with additional journey legs.

5. Machine-Learning-Based Transfer Point Optimisation of Journey Profiles (Online)

Once a journey profile Pareto set has been generated for a commuter's query, the task is to define precise paths and transfer points for each journey profile while taking into account real-time transport network traffic flows and disruptions, i.e., using real-time transport network graphs and GTFS data. Real-time transport network data capture traffic conditions by specifying how long it takes to traverse each transport network arc at any given time of the day; therefore, the path calculations outlined in this section account for varying traffic conditions. Figure 3 outlines the proposed procedure for this, which is applied to each non-dominated journey profile in turn. The approach is to cycle through the individual legs of a journey profile one at a time. Each journey leg-iteration builds a many-to-many shortest-path tree from the previous transfer zone (*origin(s)*) to the nodes in the current transfer zone (*reachedNodes/Stops*) of the journey profile's transfer-zone sequence. This tree is built within the transport network defined by the journey profile's mode sequence. If the current mode corresponds to an open transport network, the relevant many-to-many shortest-path algorithm is Dijkstra's algorithm with numerous speed-up techniques (see Section 5.1), which calculates and stores the shortest paths and arrival times at each node in the current transfer zone. Conversely, if the current mode is public transport, the relevant many-to-many shortest-path algorithm is a specially adapted version of RAPTOR, which calculates and stores shortest time public transport journeys from the previous transfer zone to all of the stops in the current transfer zone (see Section 5.2). Once a many-to-many shortest-path tree has been built to the current transfer-zone, machine-learning travel time predictions for the next leg of the journey profile are used to select one of the *origins* of the current many-to-many shortest-path tree as a transfer point. At which point, the shortest path leading to that transfer point is locked into the journey solution. If the current transfer zone is the last one, machine-learning predictions are not required since we just commit to the transfer point corresponding to the origin of the shortest path to the actual destination node. Section 5.3 describes the input features and training procedure of the neural-network approach for predicting travel times between any pair of locations in each open transport network. Section 5.4 details the machine-learning-based transfer point selection procedure.

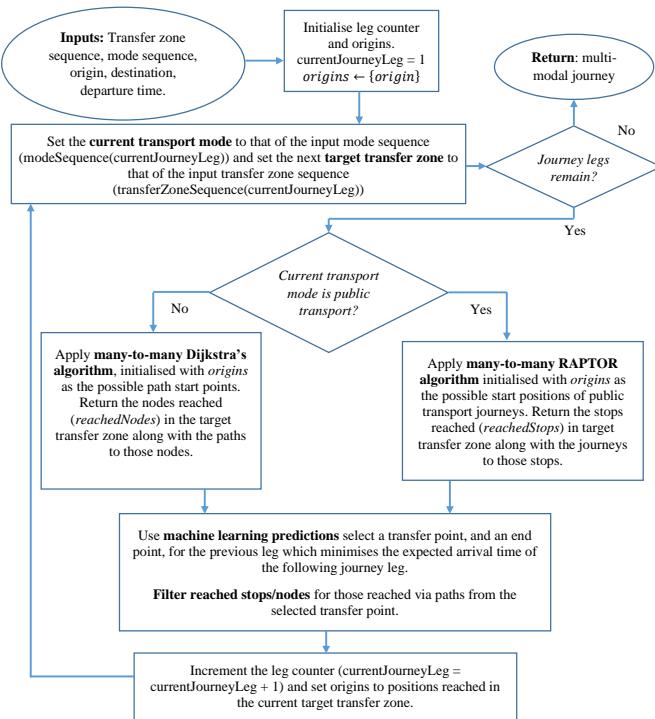


Figure 3. Flow chart for the machine-learning prediction guided transfer point optimisation procedure.

5.1. Many-to-Many Open-Network Shortest-Path Algorithm

Algorithm 2 presents our target transfer-zone-directed many-to-many shortest-path algorithm with speed ups for open transport networks. The algorithm is designed to grow a many-to-many shortest-path tree from specified nodes in the origin transfer zone (*origins*) to nodes in the destination transfer zone (*reachedNodes*). The algorithm is based on the main steps of Dijkstra's algorithm (lines 6–10, 20–21, 38–45), but also makes use of three different speed-up techniques. Both of the first two speed-up techniques are based on partitioning transport network nodes into clusters, each with defined boundary nodes; for this, Algorithm 1 is used. Firstly, the *arc flag* speed up (lines 25–30) is used to avoid the consideration of paths with edges known not to lie on any shortest paths from *arc flag boundary nodes* to any *arc flag boundary regions* overlapping with the target transfer zone (*endTransferZone*). Secondly, the *cross-region edge label* speed up (lines 32–36) is used to avoid the consideration of paths with edges, not in either the origin or destination transfer zone, that are known not to be in any *cross-region edge label* boundary zone shortest paths; Section 5.1.1 provides further details. Thirdly, the algorithm is provided with a dummy target destination location (*dummyDestination*) somewhere within the target destination transfer zone, which is used to simulate the goal-directed A* speed up of Dijkstra's algorithm, in which the candidate list is sorted according to lower bound estimates (“as the crow flies”) of the arrival times at the dummy destination from the given node's location. This has the effect of prioritising the resolution of shortest paths closer to the target transfer zone. The algorithm terminates when it is known that the shortest paths to all of the target transfer-zone nodes have been resolved.

Algorithm 2 Target transfer-zone-directed many-to-many Dijkstra's with arc flag and cross region edge label speed ups.

```

1: Inputs: origins, startTransferZone, endTransferZone, dummyDestination, maxSpeed,  

   numberOfNodesInTargetTransferZone.
2: //Initialise candidate list, the set and number of target nodes reached.
3: candidateList  $\leftarrow$  origins
4: reachedNodes  $\leftarrow \emptyset$ 
5: nodesInTargetTransferZoneScanned = 0
6: while  $|candidateList| > 0$  do
7:   //Resolve and scan the unresolved node with the lowest heuristic score and store  

   the origin associated with the resolved path.
8:   nodeToScan  $\leftarrow$  candidateList(1)
9:   candidateList  $\leftarrow$  candidateList \ {nodeToScan}
10:  nodeToScan.currentLegStartNode  $\leftarrow$  nodeToScan.parentNode.currentLegStartNode
11:  if transferZone(nodeToScan) = endTransferZone then
12:    nodesInTargetTransferZoneScanned = nodesInTargetTransferZoneScanned + 1
13:    reachedNodes  $\leftarrow$  reachedNodes  $\cup$  {nodeToScan}
14:  end if
15:  //Check if termination criterion is satisfied.
16:  if nodesInTargetTransferZoneScanned = numberOfNodesInTargetTransferZone then
17:    Return reachedNodes
18:  end if
19:  //Consider whether shorter paths to nodes linked to nodeToScan can be found.
20:  edgesOut  $\leftarrow$  nodeToScan.edgesOut
21:  for edge  $\in$  edgesOut do
22:    //Apply speed ups.
23:    edgePruned = false
24:    //Arc flag speed up.
25:    if arcFlagBoundaryNode(nodeToScan) then
26:      edgePruned = true
27:      if edge.onAShortestPathToSomeWhereInATargetArcFlagBoundaryZone  

   (endTransferZone) then
28:        edgePruned = false
29:      end if
30:    end if
31:    //Cross region edge label speed up.
32:    if  $(transferZone(nodeToScan) \neq startTransferZone) \vee$   

    $(transferZone(nodeToScan) \neq endTransferZone)$  then
33:      if  $\neg edge.crossRegionShortestPathEdge$  then
34:        edgePruned = true
35:      end if
36:    end if
37:    if  $\neg edgePruned$  then
38:      nextNode = edge.endNode
39:      newPathTime = nodeToScan.pathTime + edge.time
40:      if newPathTime < nextNode.pathTime then
41:        //Shorter path to nextNode found, update candidate list and store new parent  

        node.
42:        nextNode.pathTime  $\leftarrow$  newPathTime
43:        nextNode.parentNode  $\leftarrow$  nodeToScan
44:        nextNode.heuristicScore = newPathTime +  

         crowFliesTime(nextNode, dummyDestination, maxSpeed)
45:        addAndSort(nextNode, candidateList, 'heuristic score increasing')
46:      end if
47:    end if
48:  end for
49: end while
50: reachedNodes

```

5.1.1. Speed-Up Techniques

Our speed-up techniques build upon the same logic as *arc flags* and *geometric containers* [51], in which the nodes of a transport network graph are partitioned into a set of distinct regions. Within each region, boundary nodes are identified as those which have edges leading to other regions. For each boundary node, a Dijkstra's search to all other nodes is performed, the results of which are used to define a bit vector for each edge leaving a boundary node, which specifies each region that can be reached via a shortest path including that edge. Then, future Dijkstra's searches can prune edges leaving boundary nodes which are known not to be included on a shortest path to anywhere in the final target region. In our case, we define our set of regions using Algorithm 1.

As an extra pruning step, we also introduce a *cross-region shortest-path pruning* technique. Since we know that if the current “resolved/scanned node” in a Dijkstra's search is not in the origin or destination region, the completed shortest path must include a shortest path between a pair of boundary nodes of the current region, because all sub-paths of shortest paths are also shortest paths. Based on this, we monitor which edges are included in shortest paths across each region between each pair of its boundary nodes, which is information that can be yielded during the pre-processing step of the arc flags speed up referred to above. If, in future Dijkstra's searches, the current “resolved/scanned node” is not in the origin or destination region, we can prune edges which are never in shortest paths across that region. We refer to this speed-up technique as *cross-region edge labels*.

Figure 4 displays the boundary nodes (green dots) of partitioned regions and cross-region shortest paths (blue); all of the grey edges can be pruned in a Dijkstra's search when the current “resolved/scanned node” is not in the origin or destination region.

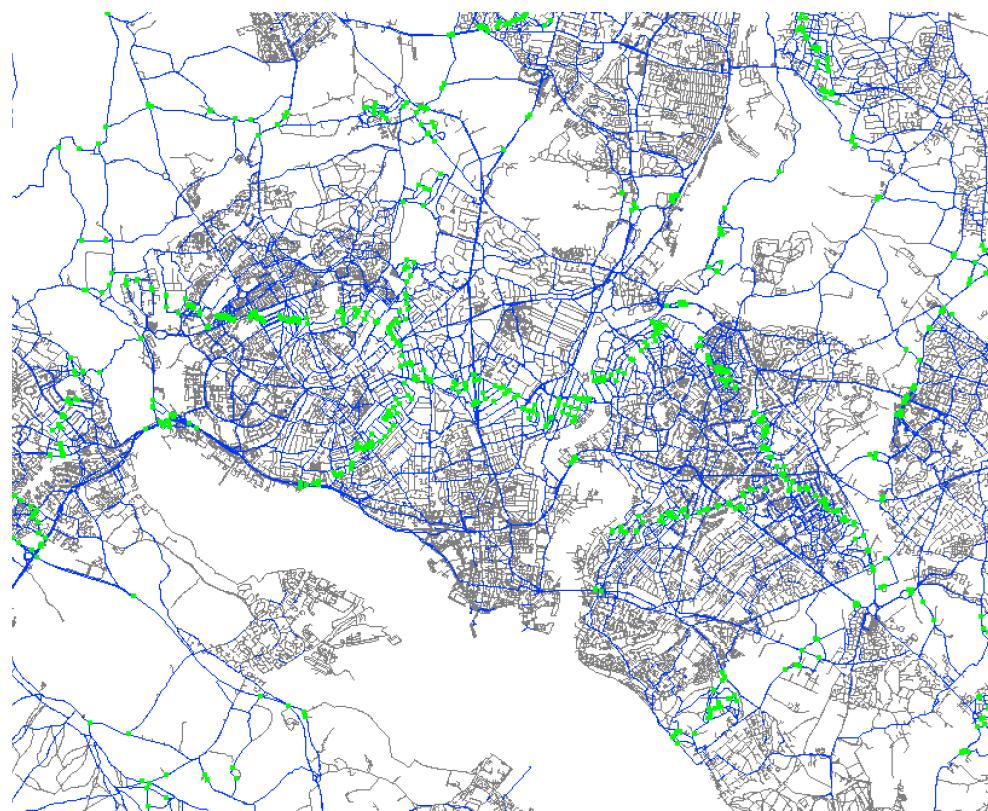


Figure 4. Cluster boundary nodes (green squares) and cross-region shortest-path edges (blue lines) of the *cross-region edge labels* speed-up technique applied to a pedestrian network.

Appendix A contains experiment results for these speed-up techniques. In summary, the pre-processing requirements of these techniques increase with transfer-zone sample size, never exceeding an hour. The *arc flags* technique alone halves query times on average, while the *cross-region edge labels* technique alone reduces query times by two-thirds on average. When both techniques are applied simultaneously, query times can be reduced by 93%.

5.2. Many-to-Many Public Transport Shortest-Path Algorithm

Algorithm 3 outlines our transfer-zone sequence constrained many-to-many adaptation of RAPTOR. As with standard RAPTOR, the algorithm works in transfer-iterations where each subsequent iteration attempts to find earlier arrival times to stops involving itineraries with one additional transfer (line 13). Within each transfer-iteration, the standard steps of RAPTOR are step 1 (line 14), step 2 (line 15) and step 3 (line 16). Step 1 gathers routes whose stops were reached at earlier times in the previous transfer-iteration, or routes serving origin stops in the first transfer-iteration. Step 2 cycles through those routes, finding earlier arrival times to the stops along it. Step 3 searches for stops that can be reached more quickly via walking connections from the stops reached more quickly via routes in the current transfer-iteration. The differences between our algorithm and RAPTOR are highlighted in bold italics and include steps 4–6. As specified on lines 15 and 16, RAPTOR target pruning no longer applies because we have multiple targets. In contrast to RAPTOR, our algorithm stores full itinerary information for the routes found to all stops. Our proposed approach for storing all itineraries is to store a path string for each stop and number of transfers which specifies the entire itinerary of the corresponding public transport journey. Each time an earlier arrival time to a stop is found, the corresponding path string is set as that of the predecessor stop in the route concatenated with details of how the stop was reached from the predecessor stop. In [9], the authors focus on finding earliest arrival times as quickly as possible, and only state that it is possible to recover itineraries from the algorithm. The path strings of origin stops (*origins*) specify which stop it is and the arrival time, so that the specific origin, of the many possible, is information that gets carried through the algorithm through the string concatenation itinerary storage approach. This itinerary storage approach was found to be more convenient than an approach based on storing references to parent stops, which is an approach that works well for open-network journey planning using Dijkstra-based algorithms.

Our adaptation has the added feature that it stores and returns all public transport journeys to stops in the final target transfer zone (*reachedStops*). For the case where journey profiles contain consecutive public transport journey legs, Algorithm 3 builds a many-to-many shortest-path tree through the specified transfer-zone sequence for that multi-leg public transport journey. Step 6 applies transfer-zone sequence pruning so that the specified transfer-zone sequence is adhered to, which can be regarded as a speed-up technique.

The adaptation also includes two fail safe mechanisms for cases in which the planned transfer-zone sequence of a journey leg cannot be followed due to real-time schedule information and disruptions. We waive the pruning of stops not in target transfer zones (Step 6) if none are found in the target transfer zones. Furthermore, extra transfer iterations, up to a maximum number, are added when no routes are found to stops in final target transfer zone.

Algorithm 3 Transfer-zone sequence-constrained many-to-many RAPTOR.

```

1: Inputs: origins, transferZoneSequence, maximumTransfers, maximumExtraIterations.
2: reachedStops  $\leftarrow \emptyset$ 
3: markedStops  $\leftarrow \text{origins}$ 
4: extraIterationsPerformed = 0
5: Reset earliest arrival times and full itineraries associated with all stops and number of
   transfers except for the stops in origins.
6: for  $k \in \{1.. \text{maximumTransfers}\}$  do
7:   for  $i \in \text{stops} \setminus \text{origins}$  do
8:      $\text{arrivalTime}(i, k) \leftarrow \infty$ 
9:      $\text{pathString}(i, k) = \emptyset$ 
10:    end for
11:   end for
12: Perform the specified number of transfer iterations (information derived from a journey
   profile).
13: for  $k \in \{1.. \text{maximumTransfers}\}$  do
14:   Step 1. Gather routes reached in the previous round.
15:   Step 2. Hop onto routes earlier than previously from stops reached in the previous
      iteration while finding earlier arrival times at stops along those routes. No target
      pruning and store full journey itinerary corresponding to each earlier stop arrival
      time found (i.e., pathString(i, k) values).
16:   Step 3. Find stops that can be reached earlier by walking from stops reached earlier
      via routes in this iteration. No target pruning and store full journey itinerary
      corresponding to each earlier stop arrival time found (i.e., pathString(i, k) values).
17:   Step 4. Add marked stops in the final public transport leg target transfer zone to
      reachedStops.
18:   Step 5. Allow extra transfer iterations in the event that no stops have been reached
      in the final public transport leg target transfer zone. (i.e., allow for failure due to
      real-time disruptions.)
19:   if  $(k = \text{maximumTransfers}) \wedge (\text{extraIterationsPerformed} < \text{maximumExtraIterations})$ 
      then
20:     if markedStops does not include stops in transfer zone transferZoneSequence(k)
      then
21:        $\text{maximumTransfers} = \text{maximumTransfers} + 1$ 
22:        $\text{extraIterationsPerformed} = \text{extraIterationsPerformed} + 1$ 
23:     end if
24:   end if
25:   Step 6. Perform transfer-zone sequence pruning, provided that public transport
      journeys to this iterations target transfer zone have been found; this exception is
      another mechanism for allowing for failure due to real-time disruptions.
26:   if markedStops includes stops in transfer zone transferZoneSequence(k) then
27:     Filter markedStops retaining only those in transferZoneSequence(k).
28:   end if
29: end for

```

5.3. Neural-Network Travel-Time Predictions

In ML-TZSA, feed-forward neural networks are used to rapidly approximate shortest time paths from anywhere in an origin transfer zone to anywhere in a destination transfer zone in any transport network. As such, we train a separate neural network for each pair of origin and destination transfer zones and each transport mode. Regarding the training of each neural network, we firstly generate the neural-network structure according to the specified number of input features, output responses and the numbers of neurons in each layer. In this case, there are 13 input features and two output responses (travel time and distance). The input features for making a prediction include origin and destination longitudes and latitudes (4 features); origin and destination longitudes and latitudes relative to their respective transfer-zone centroids (4 features); the angles and distances of

the origin and destination relative to their respective transfer-zone centroids (4 features) and the “as-the-crow-flies” distance between the origin and destination (1 feature). These features can be calculated much more rapidly than the exact shortest path between an origin and destination can be, as can a forwards pass of a neural network for making a prediction based on these features. Figure 5 illustrates the structure of the neural networks, which consists of a layer by layer fully-connected feed-forward structure. The hyperbolic tangent activation function is used for all neurons, and the synapse weights were calculated using the back propagation algorithm. The number of neurons in the layers between the input signal and output layer varies smoothly between the input vector size and output vector size (13-[10-6]-2), which is a common heuristic that helps to reduce the risk of overfitting. For each neural network, a sample of training data is generated, consisting of pairs of input features and output responses for randomly selected origins and destinations within the transfer zones of interest. The A^* shortest-path algorithm is used to calculate the desired output responses. The back propagation algorithm is used to train the weights of the neurons of the neural network. The training procedure involves repeatedly selecting random instances from the training data and performing a neural-network forwards pass to make a travel-time prediction based on the feature data, followed by a backward pass for updating the neuron weights to reduce prediction error in the future. This procedure requires a learning rate scheme which determines how aggressively the neuron weights are updated in response to prediction errors. The learning rate starts small and becomes lower in order to prevent over fitting to individual training data instances. Appendix B provides neural-network prediction accuracy and time results. In summary, travel-time prediction errors are around 3% on average and tend to decrease with increasing transfer-zone sample sizes. Prediction response times are always a tiny fraction of the time required to calculate travel times using shortest-path algorithms.

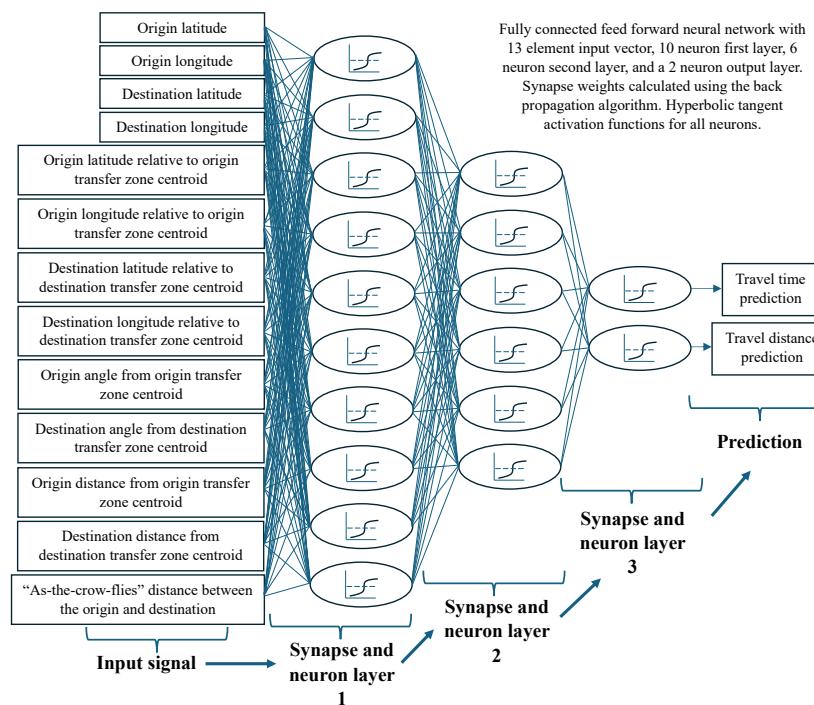


Figure 5. Neural network structure.

5.4. Transfer Point Optimisation

Using an example journey profile including the transfer-zone sequence 1,2,3 and corresponding mode sequence *walk, bike, taxi*, Figure 6 illustrates the ML-TZSA methodology for optimising the transfer points and path of a journey profile. Figure 6 depicts the

steps performed in one journey leg-iteration of Figure 3 (main loop) for the case where the current transport mode is *bike*. Firstly, the *origins*, which are the inputs of the many-to-many Dijkstra's algorithm, are the nodes in cluster 1 that can be reached along the walking path tree (black lines) from the origin. The many-to-many Dijkstra's algorithm generates cycling shortest-path trees (green lines) from cluster 1 to cluster 2. The next step is to commit to one of the cycling sub-trees, and thereby fix the transfer point of cluster 1 and also a single walking path within the walking path tree. For this, machine-learning predictions (depicted as curved dashed lines) are used to estimate arrival times at cluster 3's sampled transfer point (centroid), via taxi, starting from each possible cluster 2 transfer point. In the example, a transfer from bike to taxi at node 9 is found to minimise the expected arrival time at cluster 3's sampled transfer point, which corresponds to the cycling path sub-tree starting at node 7 (thick green lines), which is then fixed as the transfer point in cluster 1. As a result, nodes 9, 10, 13 and 15 become the origins for the next iteration of Figure 3, as well as being the candidate locations for the next transfer point in cluster 2. In summary, machine-learning predictions for the next journey leg are used to decide on a path through the previous leg's many-to-many shortest-path tree and commit to a single path sub-tree of the current leg's many-to-many shortest-path tree. The issue that machine-learning predictions can exhibit inaccuracy is not too critical given that the machine-learning predictions are only used to choose a sub-tree from a many-to-many shortest-path tree.

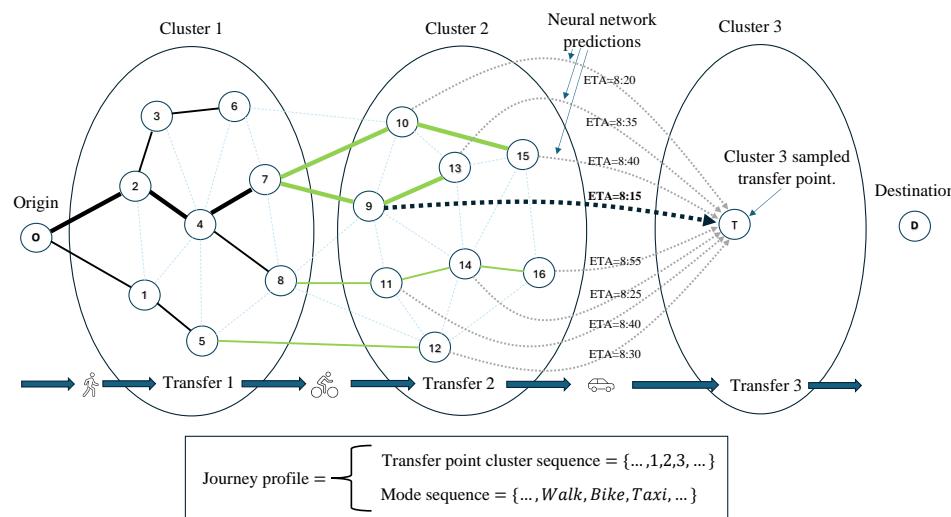


Figure 6. Diagram illustrating how machine-learning predictions of travel times of the subsequent leg of a multi-modal journey, starting from different possible transfer zones, are used to select the transfer point of the previous leg.

To conclude the presentation of the proposed methodology, we note that the aim of this approach is to generate a large number of alternative journeys with attractive features regarding cost, time, CO₂, convenience and calorie expenditure. So, while our journey profile transfer point optimisation procedure is driven by time minimisation, constraining the transfer points to limited size transfer zones provides the mechanism by which the objective criteria characteristics of the original journey profiles are preserved. On a case-by-case basis, it is also possible to further limit the candidate transfer points to those that do not impact the objective values too much. In any case, the proposed approach is intended to augment the offerings of MaaS apps rather than replace their existing ones, be they optimal or not. Regarding the issue of presenting too many journey options to commuters, the alternatives can be ordered according to different preference criteria (possibly listed in a drop-down menu). Another option is to present those optimising each individual objective

in addition to a selection of those balancing conflicting criteria, perhaps identified using a clustering algorithm.

6. Results

In this section, experiment results are presented for the offline pre-processing step of ML-TZSA and the online query responses of ML-TZSA. ML-TZSA is also compared with two versions of RAPTOR in order illustrate the superior diversity of the outputs of ML-TZSA, as well as to provide some validation of the quality of the journeys generated by ML-TZSA. ML-TZSA is also analysed in terms of a number of different Pareto set quality metrics for a variety of transfer-zone sample sizes. The experiment results reported were all carried out using a 3.9 GHz processor with 8 Gb RAM.

6.1. Experiment Inputs

While our methodology can be applied to any sets of open and public transport networks, our experiment results are based on the transport networks of the Solent region of the UK, which can be summarised as follows. The Solent transport network is composed of pedestrian, bike and road open networks, as well as bus, ferry and train public transport services. Ferries connect the Isle of Wight with the mainland and connect the adjacent peninsulas of the coast of the mainland. Trains connect the main suburbs, towns and cities on the same islands, while the bus network provides the same plus a more extensive range of connections between local conurbations. The pedestrian transport network is defined by 202,816 nodes and 523,580 edges. The road network is defined by 141,743 nodes and 313,373 edges. The e-scooter transport network is defined by 204,037 nodes and 524,917 edges; however, e-scooters are restricted by geofence constraints to the cities of Portsmouth and Southampton. The public transport network consists of bus, ferry and train routes, of which there is a total of 1272 unique routes and 5830 stops. GTFS data define the timetables of the public transport networks as well as the dates that different public transport trips are available. Regarding the modelling of connections between different transport networks, nodes are generated in each open transport network corresponding to each public transport stop along with edges connecting those nodes to the nodes of the nearest existing edges in those networks. For connections between open transport networks, connection nodes are similarly generated if the nearest edges in the other networks are within some specified small distance. These additional connection nodes and edges are included in the totals reported above.

Table 1 shows the parameters associated with each transport mode, which were collected from various sources, including www.carbonindependent.org (accessed on 4 April 2024), and capture the main characteristics and relative differences between each transport mode. The CO₂ contribution for public transport is based on that of buses (822 g/km) and is divided by 20 as an attempt to make the CO₂ accounted for directly attributable to each individual commuter. Walking is assumed to consume 60 calories per kilometre. The taxi cost (GBP/metre) is based on 40p per fifth of a mile and a fixed cost of GBP 2.50 per trip. The e-scooter fixed and variable costs are representative of those of a current app-based e-scooter trial (Voi Portsmouth and Southampton). The vehicle maximum speeds have been extracted from the static open transport network graphs and are used for the goal-directed heuristic scores used in the A* shortest-path algorithm.

A day ticket public transport cost model has been selected because of its real-life relevance and simplicity, and because it leads to a special exception when checking whether or not one partial solution dominates another. The issue is that a partial solution without public transport may need public transport for its completion. Therefore, we need to avoid eliminating a partial solution which already includes public transport on the basis of its

cost exceeding that of a partial solution without public transport by less than the cost of a day ticket; we must instead retain both partial solutions. Another exception is that we must avoid comparing and eliminating partial solutions if they do not end at the exact same transfer point within the same transfer point cluster. Partial solutions ending with public transport legs may end at different stops within the same transfer point cluster.

6.2. ML-TZSA Pre-Processing Results

Figure 7 (left) shows the pre-processing times required to complete the tasks of Sections 4.1 and 4.2 for different transfer-zone sample sizes. For transfer-zone sample sizes of 50 and above, pre-processing times increase linearly with transfer-zone sample size. This is because the number of calls of Dijkstra's algorithm required for calculating open transport network inter-transfer-zone objective criteria contributions is equal to the number of transfer zones. The task of generating the transfer zones is only dependent on the size of the pedestrian transport network and not the number of transfer zones. For low transfer-zone sample sizes, the task of calculating minimum connection times for consecutive public transport legs takes a longer amount of time because there are more possible public transport connections in larger transfer zones. The calculation of inter-transfer-zone objective criteria contributions in the public transport network takes the least amount of the total pre-processing time because public transport routes are pre-defined and the public transport network is generally a fraction of the size of any of the open transport networks.

Figure 7 (right) shows the portions of all inter-transfer-zone journeys that are actually possible in each transport network. These values provide a measure of the connectivity of each transport network. For the case of pedestrian, bike and car networks, approximately 0.8 of all inter-transfer-zone trips are possible, independent of transfer-zone sample size. This reflects the fact that the Isle of Wight is not connected to mainland Britain via these networks. The e-scooter network, which is a subset of the bike network, has the lowest level of connectivity, which is because of the city-based geofence constraints on the use of hireable e-scooters. For the public transport network, the connectivity appears to decrease with increasing transfer-zone sample size, which is because smaller transfer-zones each contain fewer public transport stops and therefore fewer ways to reach other transfer zones. The connectivity of the public transport network is least accurately judged on this criterion because the real connectivity of the public transport network relies on transfers, where that of the open transport networks does not. Multi-modal journeys facilitate a connectivity level greater than that of any of the individual constituent transport networks.

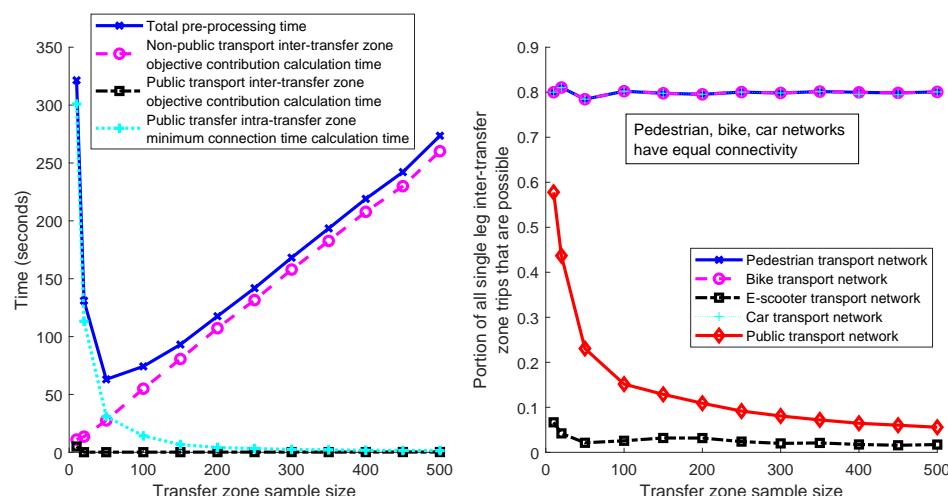


Figure 7. (Left) Pre-processing times and their components. (Right) The connectivity of different transport networks measured as the portion of all inter-transfer-zone trips that are possible.

6.3. ML-TZSA Query Response Results

Figure 8 provides average query response information based on 2000 commuter queries, each varying by origin, destination and departure time. The top row of bar charts provides the average number of non-dominated journey profiles generated and average number of non-dominated multi-modal journeys generated from those journey profiles, for a range of different transfer-zone sample sizes. The bottom row of bar charts provides corresponding average query times expressed in orders of magnitude of seconds ($1 = 10\text{ s}$, $0 = 1\text{ s}$, $-1 = 0.1\text{ s}$, $-2 = 0.01\text{ s}$, $-3 = 0.001\text{ s}$). The first column limits the available transport modes to walking and public transport; for this we include results for RAPTOR [9]. The second column limits the available transport modes to walking, public transport and e-scooters. For this, we include results for RAPTOR-ES, which is RAPTOR where e-scooters can be used via e-scooter racks at the beginning and/or end of public transport journeys as well as to make connections between consecutive public transport services. The third column limits the available transport modes to walking, public transport, e-scooters and taxis.

Figure 8 reveals several general trends. The average journey profile Pareto set sizes increase the more transport modes there are available, as do average total query times. On average, half of the journey profiles lead to non-dominated journeys, which can be attributed to the aggregated and approximate nature of the journey profile generation procedure with respect to exact query departure times, origin and destination locations, but which are accounted for in the transfer point and path optimisation procedure. Average total query times remain between 0.1 s and 10 s in all cases, i.e., acceptable for real-time applications. The largest average total query times are 3.34 s for RAPTOR-ES (for the case of walking, public transport and e-scooters being available) and 3.21 s for ML-TZSA (10) (for the case of walking, public transport, e-scooters and taxis being available). Figure 8 also shows that ML-TZSA average total query times are somewhat independent of transfer-zone sample size. Although the number of solutions generated increases with transfer-zone sample size, the time required to optimise the paths and transfer points of individual journey profiles decreases. The transfer point solution space, constrained by the transfer-zone sequences, decreases within increasing transfer-zone sample size. Average individual journey profile transfer point optimisation times range between 0.575 s for the case of 10 transfer zones and 0.0189 s for the case of 500 transfer zones. This is a positive result, meaning that we need not worry about finding a compromise transfer-zone sample size and simply make it as large as possible. The bottleneck regarding the maximum transfer-zone sample size is the memory space required to store the pre-processed information, which becomes unmanageable for a current standard desktop computer for transfer-zone sample sizes in excess of 500. For the case of walking, public transport and e-scooters being available, ML-TZSA can generate Pareto sets of journey profiles in a time ranging from 1.4 to 10.8 ms. Transfer point optimisation accounts for the vast majority of total query time.

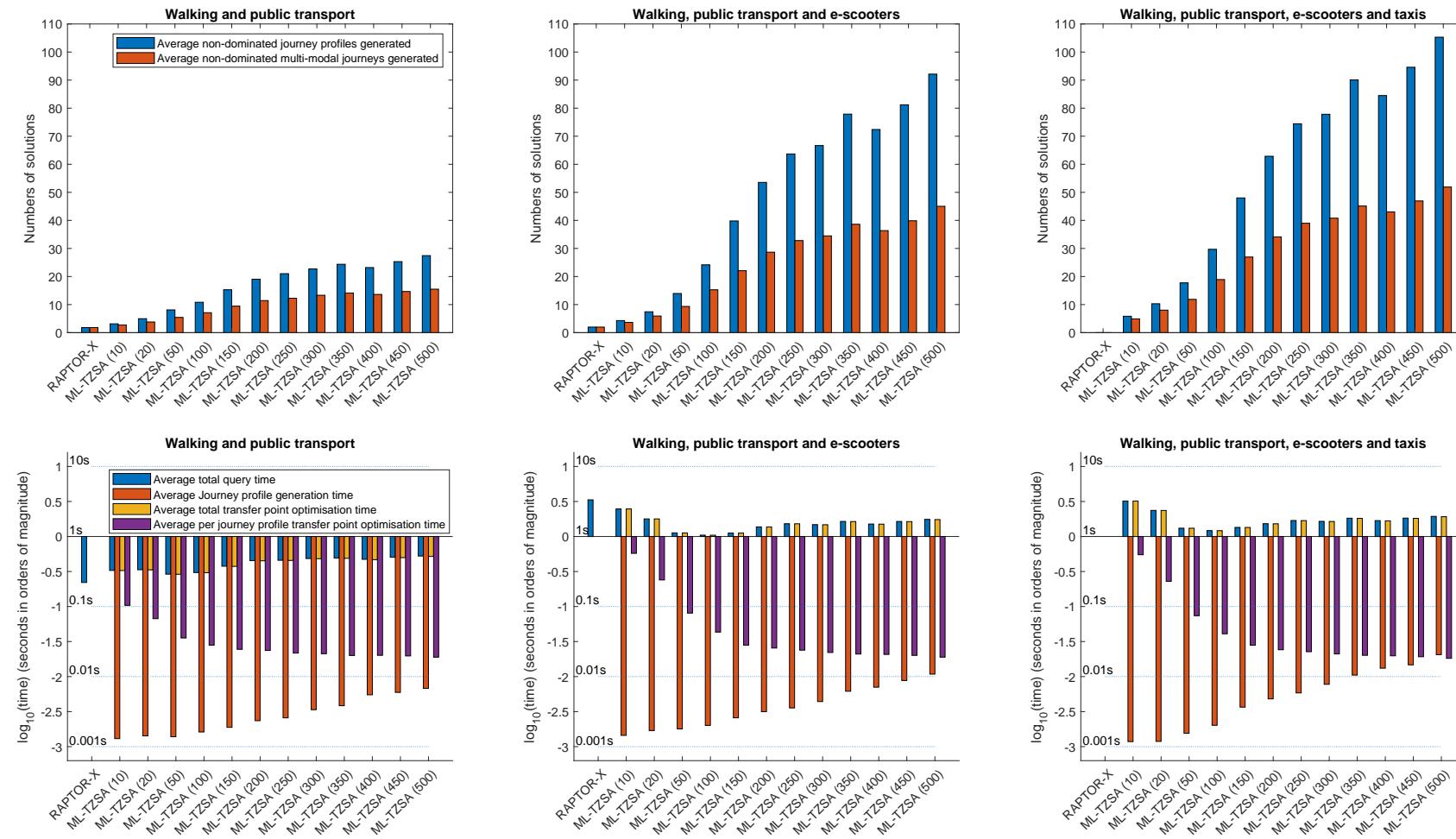


Figure 8. Average numbers of solutions and query times for ML-TZSA and RAPTOR methodologies for different sets of available modes, based on 2000 example commuter queries.

Figure 8 showed that ML-TZSA can generate many more solutions than two different versions of RAPTOR in roughly equal query times. The main reason for this is that RAPTOR's objective is to minimise travel time and transfers, which is an objective function that generally characterises a smaller Pareto set of journeys. Figure 9 provides results for a comparison between ML-TZSA and RAPTOR based on the best objective values found for each criteria for each of the 2000 test queries. The results are presented as relative values, where values below 1 indicate that ML-TZSA finds better objective values for particular criteria compared to RAPTOR on average. When only walking and public transport are available, RAPTOR finds slightly faster and less strenuous (in terms of calorie expenditure) journeys on average. This perhaps is not so surprising considering that ML-TZSA's RAPTOR component is relatively constrained by the maximum transfers and transfer-zone sequence imposed by journey profiles. This reinforces the idea that ML-TZSA is best considered an option for augmenting the offerings of MaaS apps rather than being a replacement for their existing ones. However, in all other cases, ML-TZSA finds better best solutions than RAPTOR on average. This is true for all criteria when the available modes are walking, public transport and e-scooters, and for the criteria of walking and waiting time, cost and CO₂ emissions, when the available modes are walking and public transport. That is to say, ML-TZSA offers the possibility to directly address a wider range of objectives within a query response time suitable for real-time applications.

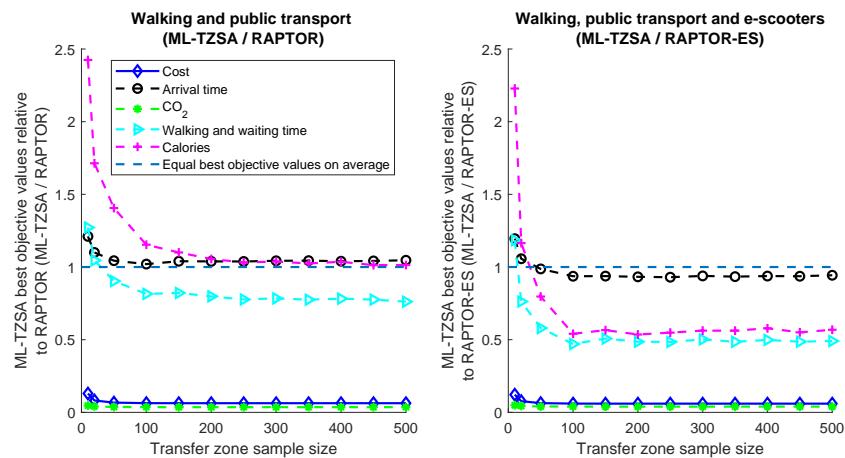


Figure 9. (Left) Average best ML-TZSA objective values relative to those of RAPTOR for the case of walking and public transport being available. (Right) Average best ML-TZSA objective values relative to those of RAPTOR-ES for the case of walking, public transport and e-scooters being available.

Figure 10 displays a sample of routes generated by ML-TZSA (500), as well as RAPTOR-ES's two non-dominated solutions (top-right) and a pure taxi route (top-middle). The bar charts indicate the overall relative quality of each solution with respect to the objectives of cost, time, CO₂, walking and waiting time and calories, respectively, with higher bars indicating better quality (lower objective values). ML-TZSA can provide numerous solutions with very similar characteristics to RAPTOR's but with variations on the modes used to access and leave the main train segment of RAPTOR's solutions. ML-TZSA's solutions include those where e-scooters are used at both the start and end of the journey (ML-TZSA 10 and 13), just at the start of the journey (ML-TZSA 3 and 11), just at the end (ML-TZSA 1) or never. ML-TZSA also provides solutions from entirely different regions of the objective space. For instance, ML-TZSA 13 shows a solution where the maximum possible usage of e-scooters is made given the geofence constraint that e-scooters can only be used within cities and cannot be used to travel between them. Such a solution has low CO₂ emissions; it is significantly faster than walking but is also one of the most expensive. Nevertheless, it is a non-dominated solution for this query. Figure 10 also shows that ML-TZSA can also generate solutions which evenly balance the trade-offs between the various objectives via a more equal division of the

total trip distance between the different transport modes. For example, ML-TZSA 1 shows a solution with moderate values for each objective criteria, achieved by travelling approximately equal distances by walking, bus and e-scooter. ML-TZSA solutions also include bus-based alternatives to RAPTOR's train-based solution (ML-TZSA 14), as well as a solution involving the use of short ferry crossings (ML-TZSA 4), which provide a faster walking-based solution than the pure walking solution (ML-TZSA 7). While none of the solutions are faster than the pure taxi journey, many relatively fast alternatives are provided which vastly reduce cost and CO₂ emissions at a small cost in terms of convenience and calorie expenditure.

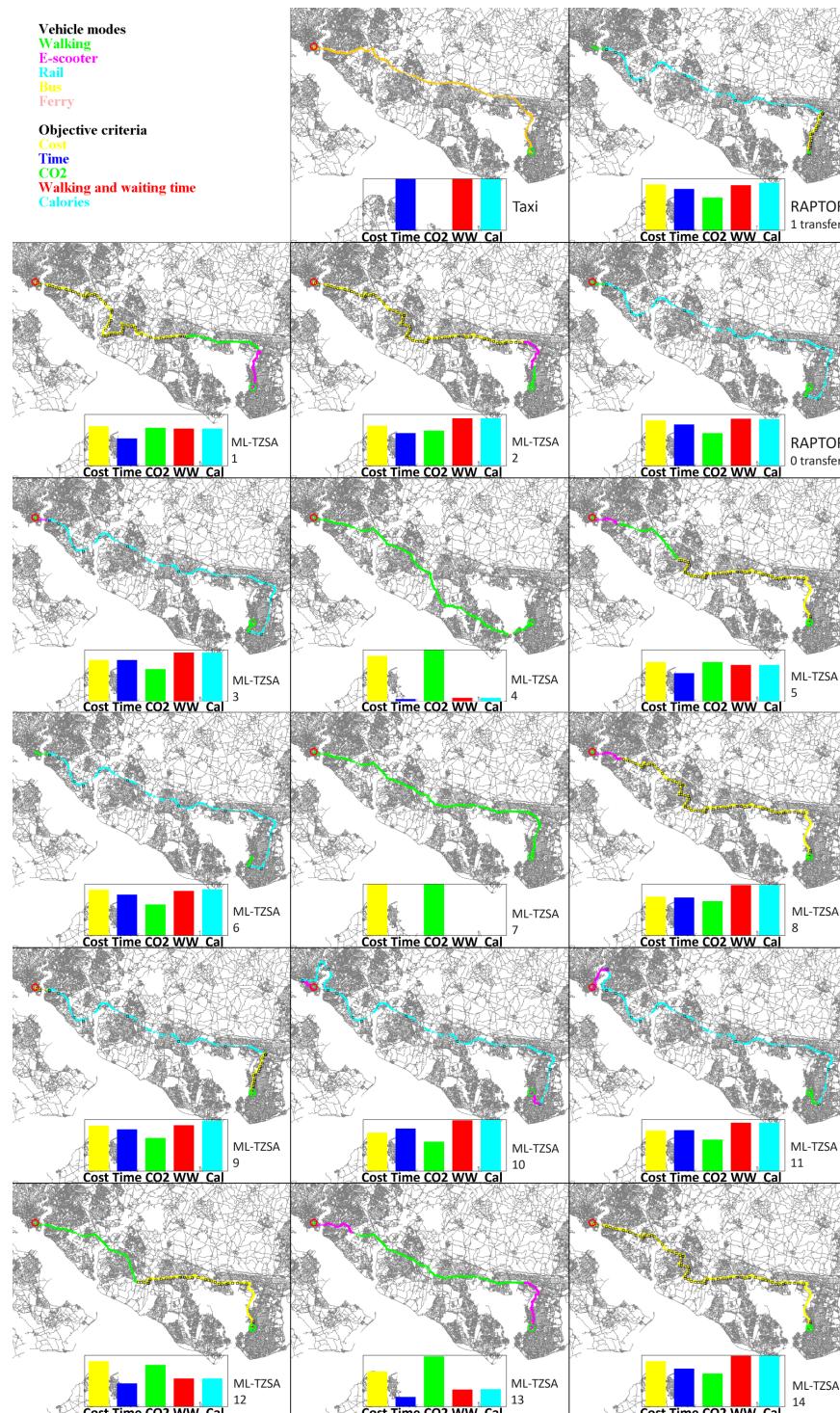


Figure 10. A sample of the 252 non-dominated journeys generated by ML-TZSA (500) and RAPTOR-ES's two non-dominated journeys, plus a taxi journey for an example query.

6.4. Pareto Set Quality Results

We now turn our attention to the quality of the Pareto sets generated by ML-TZSA and for completeness include results for those of RAPTOR solution methodologies. Our Pareto set metrics include dominance ratio, non-dominance ratio, reference set gap average and reference set gap standard deviation. Dominance ratio and non-dominance ratio are Pareto set metrics based on comparing the Pareto sets generated by two different methodologies for the same query directly against each other. The dominance ratio of one Pareto set is defined as the portion of its solutions that dominate at least one solution from the Pareto set it is being compared with. The non-dominance ratio of one Pareto set is defined as the portion of its solutions that are not dominated by any solution from the Pareto set it is being compared with. The reference set gap average of a Pareto set is defined as the average distance (in normalised objective value space) between each of a reference Pareto set's solutions and the nearest solutions in the Pareto set being measured. This measure is zero when a solution method can generate the same Pareto set as the reference set. Regarding the reference set, ideally this would be the optimal Pareto set; however, currently no algorithm scalable enough exists to calculate it within a reasonable amount of time for the large sized instances considered here. Instead, we generate a reference Pareto set for each query composed of the combined set of mutually non-dominated solutions from all available solution methodologies. The available solution methodologies in this case are ML-TZSA for all transfer-zone sample sizes considered, as well as those generated by RAPTOR solution methodologies. Wherever an average can be calculated, a standard deviation can also be calculated; the reference set gap standard deviation measures the variance in the average distances between each of a reference set's solutions and the nearest ones in the Pareto set being evaluated. Low values of this metric indicate a Pareto set consisting of a diverse set of solutions, which is a desirable feature of the solutions we would like to provide to a commuter.

In order to ensure a meaningful comparison, we restrict our attention to the set of queries of the 2000 test queries for which all methodologies found solutions. This excludes cases where query origins and destinations lie in the same transfer zone. In such cases, queries are generally of a short total distance, as such multi-modal journeys are less appropriate, and other methodologies are sufficient for trip recommendation purposes. Note that to overcome this issue it is possible to prepare the algorithm for different transfer-zone structures and to then use ones where the query origin and destination are not in the same transfer zone.

Tables 2 and 3 provide details regarding the numbers of solutions that each separate methodology contributed to the reference sets over all of the considered test queries, listed in decreasing total contribution order, for the cases of (i) walking and public transport being available and (ii) walking, public transport and e-scooters being available. Generally speaking, ML-TZSA with high transfer-zone sample sizes contributes most to the reference sets, and RAPTOR the least, as expressed by columns 2 and 3. Columns 4 and 5 state the total number of non-dominated solutions generated by each methodology and indicate that (very) roughly, a third of the non-dominated solutions generated by each method form part of the reference sets. Reference set contributions are proportional to the relative numbers of non-dominated solutions generated by each methodology. This indicates that all methodologies are generating consistently good quality solutions, but different numbers of them.

Table 2. The reference Pareto set contributions of different solution methodologies (walking and public transport).

Method Name	Reference Set Contributions		Non-Dominated Solutions per Method		Rank
	Total	Portion	Total	Portion Overall	
ML-TZSA (200)	7233	0.1167	19,479	0.0958	1
ML-TZSA (500)	6758	0.1090	26,774	0.1317	2
ML-TZSA (250)	6715	0.1083	21,048	0.1035	3
ML-TZSA (350)	6593	0.1064	24,277	0.1194	4
ML-TZSA (450)	6550	0.1057	25,298	0.1244	5
ML-TZSA (400)	6352	0.1025	23,495	0.1156	6
ML-TZSA (300)	6294	0.1015	22,866	0.1125	7
ML-TZSA (150)	5375	0.0867	15,619	0.0768	8
ML-TZSA (100)	3888	0.0627	11,286	0.0555	9
ML-TZSA (50)	2609	0.0421	7293	0.0359	10
ML-TZSA (20)	1731	0.0279	4151	0.0204	11
ML-TZSA (10)	1423	0.0230	1731	0.0085	12
RAPTOR	467	0.0075	1731	0.0085	13

Table 3. The reference Pareto set contributions of different solution methodologies (walking, public transport and e-scooters).

Method Name	Reference Set Contributions		Non-Dominated Solutions per Method		Rank
	Total	Portion	Total	Portion Overall	
ML-TZSA (500)	28,484	0.1450	77,812	0.1490	1
ML-TZSA (350)	23,269	0.1184	66,383	0.1271	2
ML-TZSA (250)	22,622	0.1151	56,294	0.1078	3
ML-TZSA (400)	22,359	0.1138	62,713	0.1201	4
ML-TZSA (450)	22,154	0.1128	68,815	0.1318	5
ML-TZSA (200)	21,930	0.1116	48,888	0.0936	6
ML-TZSA (300)	20,359	0.1036	59,183	0.1133	7
ML-TZSA (150)	14,599	0.0743	36,505	0.0699	8
ML-TZSA (100)	10,329	0.0526	24,265	0.0465	9
ML-TZSA (50)	5033	0.0256	12,537	0.0240	10
ML-TZSA (20)	3141	0.0160	6450	0.0124	11
ML-TZSA (10)	1682	0.0086	2314	0.0044	12
RAPTOR-ES	505	0.0026	2314	0.0044	13

Figures 11 and 12 show firstly that, generally speaking, ML-TZSA outperforms RAPTOR in terms of all Pareto set metrics by a significant amount, which is what ML-TZSA is designed for. However, as indicated by the moderate non-dominance ratios, RAPTOR does generate strong albeit few solutions. More interestingly, we see that ML-TZSA Pareto set metric values improve with increasing transfer-zone sample size, which can be attributed to the larger and more diverse Pareto sets generated when the transfer-zone sample size is higher. ML-TZSA always has a non-dominance ratio close to one, indicating that its Pareto sets rarely contains solutions that are dominated on all criteria by any RAPTOR solution. Regarding reference set gap measures, RAPTOR results are independent of transfer-zone sample size and are hence displayed as constant in this comparison. The ML-TZSA reference set gap metrics tend to zero as the transfer-zone sample size approaches 500, even though ML-TZSA (500) only contributes around 15% of reference set solutions.

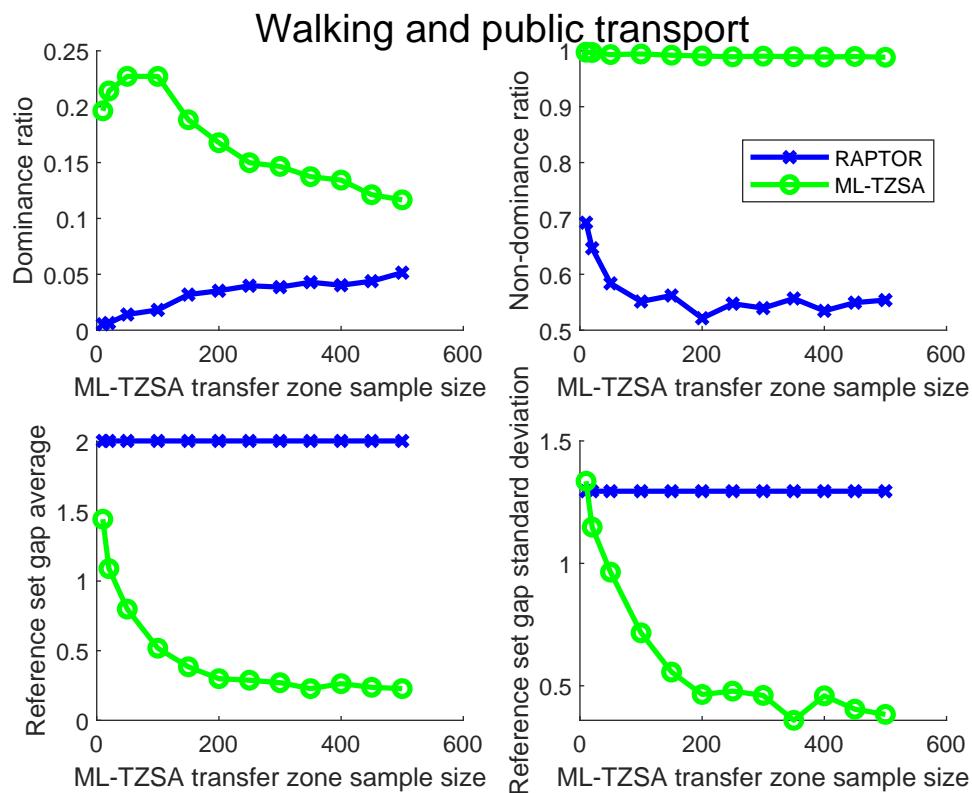


Figure 11. Dominance ratio, non-dominance ratio, reference set gap average and standard deviation results for the case of walking and public transport.

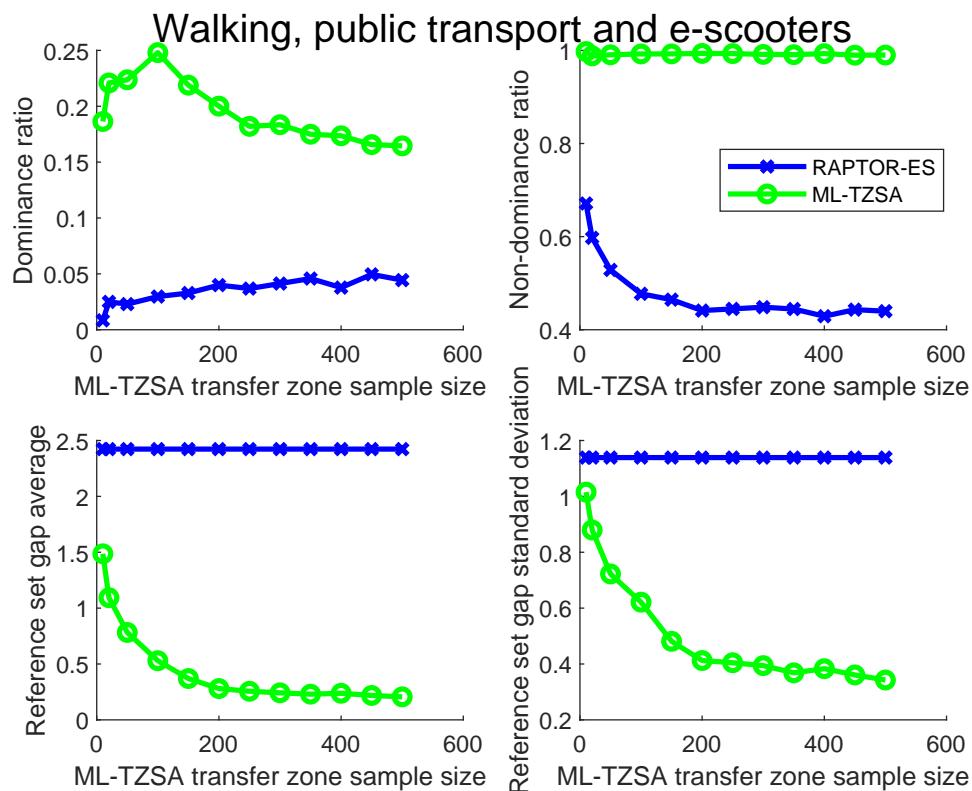


Figure 12. Dominance ratio, non-dominance ratio, reference set gap average and standard deviation results for the case of walking, public transport and e-scooters.

Figure 13 displays the objective criteria scatter plots for the reference, ML-TZSA (500) and RAPTOR Pareto sets corresponding to the query depicted in Figure 10. The reference set consists of 628 solutions (small dots), ML-TZSA (500)'s Pareto set consists of 252 solutions (circles), 115 of which are reference set solutions, while RAPTOR's Pareto set consists of 2 solutions (asterisks), neither of which are in the reference set. For this query, both ML-TZSA's and RAPTOR's dominance ratio was 0 and their non-dominance ratio was 1, indicating that, for this query, ML-TZSA (500) also found both of the solutions found by RAPTOR (note that dominance and non-dominance metrics are based on comparing ML-TZSA's solutions with RAPTOR's and not with the reference set). ML-TZSA's and RAPTOR's reference set gap averages were 0.0767 and 1.79, respectively, ML-TZSA providing an improvement of 1.71. ML-TZSA's and RAPTOR's reference set gap standard deviation values were 0.118 and 1.02, respectively, ML-TZSA providing an improvement of 0.90.

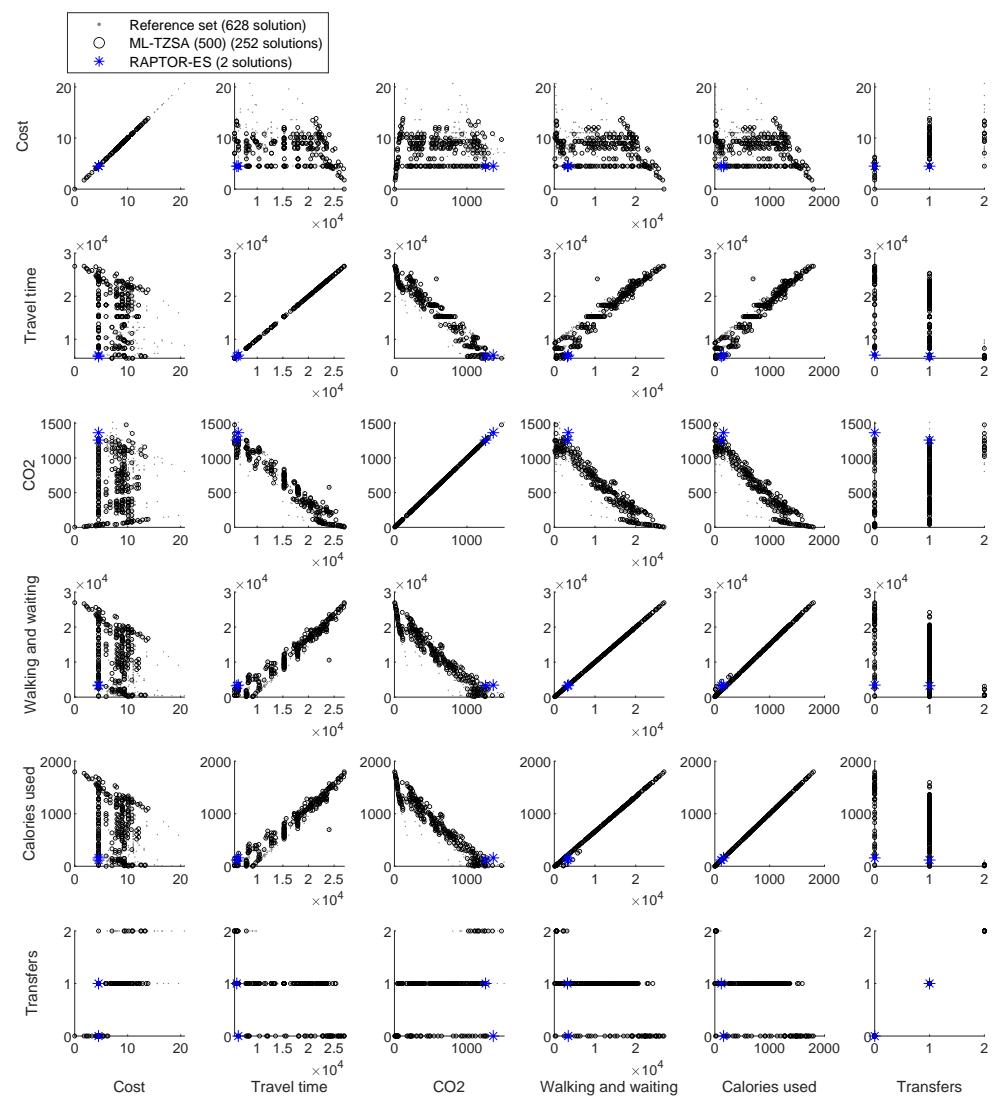


Figure 13. Pareto front objective criteria scatter plots for ML-TZSA (500), RAPTOR and the reference set for the query depicted in Figure 10, displaying the nature of the trade-offs between each criterion.

Figure 13 shows that the clearly conflicting pairs of objectives all include CO₂ emissions, which conflicts with travel time, walking and waiting time, and calorie expenditure. That is, low CO₂ emissions generally have to be accepted in exchange for higher travel times, walking and waiting times and calorie expenditure. The generally accepted conflict between the objectives of cost and travel time appears to be more convoluted than might be

expected. The explanation for this is that cost provides the main lever that enables us to use different transport modes, e-scooters in this case, that can be used to balance the conflict between CO₂ and each of travel time, walking and waiting time and calorie expenditure. Another visible pair of conflicting objectives is between travel time and transfers; generally speaking, we will only accept more transfers if total travel time is reduced.

Appendix C shows that increasing the set of available transport modes increases Pareto front quality and diversity. Additionally, it provides another perspective on some of the main sets of conflicting objectives.

7. Conclusions

This work has presented a methodology for rapidly generating many efficient alternative multi-modal journey alternatives with respect to the objectives of cost, travel time, CO₂ emissions, convenience and calorie expenditure. The aim of this methodology is to augment the offerings of MaaS apps with cheap, fast, green, convenient and energy efficient alternatives to private car journeys, which, given their uptake, can in turn help to reduce pollution and congestion on our roads.

Our methodology side steps the issue of the large solution space of possible transfer locations between pairs of transport modes using a two-step approach. Firstly, a Pareto set of journey profiles is generated based on aggregated transfer zones. Secondly, integrated and adapted versions of existing shortest-path algorithms for open and public transport networks, with various speed-up techniques, are used to optimise the transfer points and paths between transfer zones in a procedure guided by neural machine-learning predictions. A novel hybrid k-means and Dijkstra's algorithm was introduced for determining sets of transfer zones which accounts for transport network topology and local geography. In general terms, this work provides another example of how learning and optimisation can be beneficially integrated.

The proposed methodology was then tested on a real large-scale multi-modal transport network. Pre-processing times for ML-TZSA were found to be linearly increasing with increasing transfer-zone sample size. In terms of query responses, Pareto set sizes increased with transfer-zone sample size without adversely affecting query times, which were never more than a few seconds. The time required to generate journey profile Pareto sets was always very small, while the time to optimise the transfer points, and the paths between them, decreases with increasing transfer-zone sample size, because this reduces the transfer point solutions space. The result is that the recommended transfer point sample size should be as high as possible before the pre-processing times and memory requirements become too large. However, meaningful results are still obtained for moderate sample sizes.

ML-TZSA was contrasted with RAPTOR in order to illustrate how ML-TZSA can generate large diverse Pareto sets of multi-modal journey, addressing a wider variety of objective criteria in similar query times to RAPTOR. By looking at a sample of the routes generated by RAPTOR and ML-TZSA, it was shown that ML-TZSA can generate routes including those of equal quality and characteristics to those of RAPTOR's, but also that ML-TZSA can provide many solutions that minimise and balance different objectives. Based on four different Pareto set metrics, ML-TZSA consistently found high-quality and diverse Pareto sets.

While our transfer point optimisation procedure is driven by time minimisation, constraining the transfers to limited size transfer zones provided the mechanism by which the objective criteria characteristics of the original journey profiles are preserved. Quickly generating truly optimal and full Pareto sets of multi-modal journeys for the criteria considered, without relying on pre-processing and transfer-zone sampling as this work does, currently remains an open area of research. However, the issue of presenting too

many journey options to commuters warrants methods that limit the sizes of the Pareto sets generated. A related future research direction could involve trials into the influence that presenting different sets of journey alternatives has on travel behaviour for different demographics and personas. Another future direction is to investigate how shortest-path algorithms can incorporate preference criteria such as avoiding motorways, large numbers of turns and lane changes, and to use such algorithms in modular multi-modal journey-planning algorithms such as that presented here.

Author Contributions: Conceptualization, D.O., N.D. and G.F.; Methodology, C.B.; Software, C.B.; Formal analysis, D.O.; Investigation, C.B., N.D. and G.F.; Writing—original draft, C.B., D.O. and N.D.; Writing—review & editing, C.B., D.O., N.D. and G.F. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been funded by the Department of Transport (DfT) as part of the Solent Future Transport Zone (FTZ) programme led by Solent Transport, UK.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Acknowledgments: Thanks are owed to Trafi (app developers), who provided the Solent transport network data upon which this research is based.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Speed Up Technique Results

Table A1 displays the statistics associated with the pre-processing and testing of both of the goal-directed Dijkstra's algorithm (A^*) speed-up techniques presented in Section 5.1.1, applied to the pedestrian transport network of the Solent region. It shows that pre-processing times increase with the number of regions, which is because more regions mean more boundary nodes between adjacent regions (see "boundary nodes" column), and hence more corresponding calls of Dijkstra's algorithm. For 1000 regions, pre-processing takes no longer than an hour. For each number of regions, we tested 3 versions of A^* in the same set of 1000 random origin–destination queries: (i) A^* (no speed ups); (ii) A^* with arc flags; (iii) A^* with cross-region edge labels. In each case, the same shortest-path solutions are found. The final three columns of Table A1 provide average query times, given in milliseconds. All speed-up experiments do lead to reduced query times compared to A^* alone. For the case of the arc flags speed up, query times are lower the higher the number of regions, because more boundary nodes are encountered where edges can be pruned. Conversely, for the case of the cross-region edge label speed up, query times generally increase gradually the higher the number of regions, after a minimised query time for the case of 50 regions. That is, fairly large regions lead to greater portions of edges not being included in cross-region shortest paths.

The previous experiment results are based on applying both speed-up techniques separately. However, they can be applied simultaneously, along with their optimal node region definitions. When both speed ups are applied simultaneously with their respective optimal node region definitions, query times are reduced to 3 milliseconds, which is approximately a 93% query time reduction compared to A^* alone.

Table A1. Speed-up technique statistics for different numbers of regions.

Regions	Pre-		Pruned		With Arc		With Cross Region Edge Labels	
	Processing Time (s)	Boundary Nodes	Cross Region Edges	A* Query Time (ms)	Flags Query Time (ms)	Query Time (ms)		
10	112.2	787	432,023	44.76	31.08		13.66	
20	185.0	1364	404,318	44.79	29.56		10.55	
50	341.6	2496	367,501	44.69	27.28		8.95	
100	597.8	4082	335,123	44.89	25.56		10.30	
150	816.6	5382	311,209	44.76	23.43		11.60	
200	983.1	6288	291,265	44.80	22.88		12.64	
250	1162.4	7225	276,504	44.58	21.68		13.79	
300	1291.9	8014	266,260	45.17	20.38		14.45	
350	1451.0	8864	256,527	44.94	19.78		15.13	
400	1574.9	9489	248,124	45.88	19.08		15.61	
450	1703.3	10,148	240,749	45.50	18.68		16.10	
500	1873.1	10,878	234,755	45.90	17.76		16.19	
1000	3093.9	15,690	221,502	45.76	13.65		16.80	

Appendix B. Neural-Network Travel-Time Prediction Time and Accuracy Results

Table A2 provides neural-network prediction accuracy results for different transport networks and transfer-zone sample sizes based on 1000 randomly selected origin and destination queries. The root mean squared error (RMSE) of travel-time predictions are lower when there are many small transfer zones (as opposed to a few large ones), which makes sense considering that travel times between smaller zones will have a lower variance in possible shortest-path travel times. However, the storage space required for the neural networks increases with the square of the number of transfer zones. For the case of 100 transfer zones, we need to store 30,000 individual neural networks (100 zones times 100 zones times 3 open transport networks). For more than 100 transfer zones, this becomes a serious computational bottleneck for a current standard desktop computer. In terms of ML-TZSA, this means that different degrees of graph partitioning need to be used for neural-network travel-time predictions than for other journey-planning tasks when the transfer-zone sample size exceeds 100. The RMSE and mean absolute error (MAE) are always least for predictions in the car transport network. However, the relative MAE is always highest in the car transport network. This is because travel times are always lowest in the car network, so variance of travel times is naturally lower. Furthermore, predictions are more difficult to make in the car network because the roads exhibit a high variance in top speeds in comparison to paths in the other networks. The units of MAE are seconds, so for 100 transfer zones the expected prediction error lies between 30 s and 2.5 min. We except this error in exchange for the speed with which neural-network predictions can be made: small fractions of a millisecond.

Table A2. Time and accuracy measures for 1000 random test data queries (time prediction in seconds, distance predictions in metres).

Prediction Clusters	Transportation Mode	Time RMSE (s)	Time MAE (s)	Time Relative MAE	Prediction Time (ms)	A* Time (ms)
10	Walking	398.2	296.3	0.052	0.019	5.46
10	E-scooter	287.9	209.6	0.063	0	4.58
10	Car	74.5	50.9	0.079	0	2.29
20	Walking	310.6	227.3	0.038	0	5.60
20	E-scooter	222.6	136.5	0.053	0	3.66
20	Car	62.6	41.5	0.064	0	2.45
50	Walking	248.1	177.5	0.030	0	5.62
50	E-scooter	126.5	73.2	0.038	0	5.85
50	Car	44.7	31.4	0.040	0	2.41
100	Walking	211.1	150.8	0.019	0.020	5.27
100	E-scooter	103.3	64.8	0.031	0	2.56
100	Car	48.1	29.5	0.038	0	2.42

Appendix C. The Impact of the Set of Available Transport Modes on Pareto Set Diversity

Figure A1 displays spider plots of ML-TZSA (200) Pareto fronts for three different available transport mode cases. It can be seen that, as the set of available transport modes increases, the Pareto sets becomes larger and more diverse, and also that the best solutions according to the criteria of time, walking and waiting time and calorie expenditure improve. Generally speaking, different transport modes allow us to address different objectives better, and that combinations of transport modes allow us to generate more trade-off solutions that balance conflicting objectives. Each individual solution in each Pareto set has its own unique colour; the shades of the solutions in each Pareto set change uniformly according to travel time. This makes some of the trade-offs between conflicting criteria visible. The conflict between travel time and both calorie expenditure and walking and time is most clearly visible. For the cases involving each of public transport, walking and e-scooters, a trade-off between cost and travel time is also quite visible. The trade-offs relating to CO₂ are more convoluted, which is because public transport introduces non-linearities due to their discrete timetables, and that their routes are predefined and some routes are more efficient than others for different journey requirements. Journey planning has limited control over CO₂ emissions due to public transport usage.

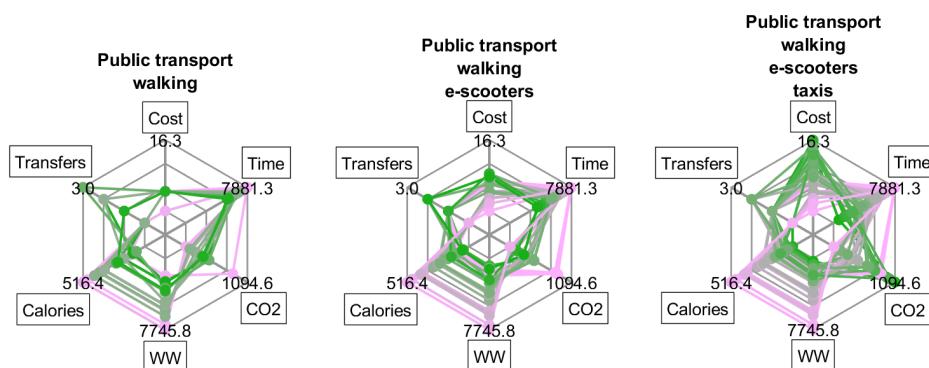


Figure A1. Spider plots of pareto fronts for different sets of available transport modes, with individual solutions coloured according to travel time.

References

- Heikkilä, S. Mobility as a Service—a Proposal for Action for the Public Administration, Case Helsinki. Master’s Thesis, Aalto University, Otaniemi, Finland, 2014.
- Kamargianni, M.; Li, W.; Matyas, M.; Schäfer, A. A critical review of new mobility services for urban transport. *Transp. Res. Procedia* **2016**, *14*, 3294–3303. [[CrossRef](#)]
- Matyas, M.; Kamargianni, M. The potential of mobility as a service bundles as a mobility management tool. *Transportation* **2019**, *46*, 1951–1968. [[CrossRef](#)]
- Sucu Sagmanli, S.; Dadashzadeh, N.; Ouelhadj, D.; Woods, L. Mobility as a Service (MaaS): A conceptual framework for travel behaviour change analysis. In Proceedings of the EURO 2022 Conference, Aalto University, Espoo, Finland, 3–6 July 2022; pp. 37–38.
- Dadashzadeh, N.; Sucu Sagmanli, S.; Ouelhadj, D.; Woods, L. Mobility as a Service and Travel Behaviour Change: Longitudinal Data Collection and Analysis Approach. In Proceedings of the 54th Annual Conference of Universities Transport Group Studies (UTSG), Edinburgh, UK, 4–6 July 2022.
- Hensher, D.A.; Mulley, C.; Nelson, J.D. Mobility as a service (MaaS)—Going somewhere or nowhere? *Transp. Policy* **2021**, *111*, 153–156. [[CrossRef](#)]
- Bayliss, C.; Ouelhadj, D. Mobility as a Service: An exploration of exact and heuristic algorithms for a new multi-modal multi-objective journey planning problem. *Appl. Soft Comput.* **2024**, *162*, 111871. [[CrossRef](#)]
- Delling, D.; Pajor, T.; Wagner, D. Accelerating Multi-Modal Route Planning by Access-Nodes. In *Proceedings of the 17th Annual European Symposium on Algorithms (ESA’09)*; Lecture Notes in Computer Science; Fiat, A., Sanders, P., Eds.; Springer: Berlin, Germany, 2009; Volume 5757, pp. 587–598.
- Delling, D.; Pajor, T.; Werneck, R.F. Round-based public transit routing. In Proceedings of the 14th Meeting on Algorithm Engineering and Experiments (ALENEX), Kyoto, Japan, 16 January 2012; pp. 130–140.
- Verbas, Ö.; Auld, J.; Ley, H.; Weimer, R.; Driscoll, S. Time-Dependent Intermodal A* Algorithm: Methodology and Implementation on a Large-Scale Network. *Transp. Res. Rec.* **2018**, *2672*, 219–230. [[CrossRef](#)]
- Dibbelt, J.; Pajor, T.; Wagner, D. User-Constrained Multi-Modal Route Planning. In Proceedings of the Meeting on Algorithm Engineering and Experiments (ALENEX), Kyoto, Japan, 16 January 2012; pp. 118–129. [[CrossRef](#)]
- Hart, P.; Nilsson, N.; Raphael, B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107. [[CrossRef](#)]
- Dijkstra, E. A Note on Two Problems in Connexion with Graphs. *Numer. Math.* **1959**, *1*, 269–271. [[CrossRef](#)]
- Ho, C.Q.; Hensher, D.A.; Reck, D.J.; Lorimer, S.; Lu, I. MaaS bundle design and implementation: Lessons from the Sydney MaaS trial. *Transp. Res. Part A* **2021**, *149*, 339–376. [[CrossRef](#)]
- Sun, D.J.; Zhang, C.; Zhang, L.; Chen, F.; Peng, Z.R. Urban travel behavior analyses and route prediction based on floating car data. *Transp. Lett.* **2014**, *6*, 118–125. [[CrossRef](#)]
- Pritchard, J. MaaS to pull us out of a car-centric orbit: Principles for sustainable Mobility-as-a-Service in the context of unsustainable car dependency. *Case Stud. Transp. Policy* **2022**, *10*, 1483–1493. [[CrossRef](#)]
- Casady, C.B. Customer-led mobility: A research agenda for Mobility-as-a-Service (MaaS) enablement. *Case Stud. Transp. Policy* **2020**, *8*, 1451–1457. [[CrossRef](#)]
- Bushell, J.; Merkert, R.; Beck, M.J. Consumer preferences for operator collaboration in intra- and intercity transport ecosystems: Institutionalising platforms to facilitate MaaS 2.0. *Transp. Res. Part A* **2022**, *160*, 160–178. [[CrossRef](#)]
- Dadashzadeh, N.; Woods, L.; Ouelhadj, D.; Thomopoulos, N.; Kamargianni, M.; Antoniou, C. Mobility as a Service Inclusion Index (MaaSINI): Evaluation of inclusivity in MaaS systems and policy recommendations. *Transp. Policy* **2022**, *127*, 191–202. [[CrossRef](#)]
- Ho, C.Q. Can MaaS change users’ travel behaviour to deliver commercial and societal outcomes? *Transp. Res. Part A* **2022**, *165*, 76–97. [[CrossRef](#)]
- Alonso-González, M.J.; Hoogendoorn-Lanser, S.; van Oort, N.; Cats, O.; Hoogendoorn, S. Drivers and barriers in adopting Mobility as a Service (MaaS)—A latent class cluster analysis of attitudes. *Transp. Res. Part A* **2020**, *132*, 378–401. [[CrossRef](#)]
- Qiu, G.; Song, R.; He, S.; Xu, W.; Jiang, M. Clustering Passenger Trip Data for the Potential Passenger Investigation and Line Design of Customized Commuter Bus. *IEEE Trans. Intell. Transp. Syst.* **2019**, *20*, 3351–3360. [[CrossRef](#)]
- van den Berg, V.A.; Meurs, H.; Verhoef, E.T. Business models for Mobility as an Service (MaaS). *Transp. Res. Part B* **2022**, *157*, 203–229. [[CrossRef](#)]
- Ding, X.; Qi, Q.; Jian, S.; Yang, H. Mechanism design for Mobility-as-a-Service platform considering travelers’ strategic behavior and multidimensional requirements. *Transp. Res. Part B Methodol.* **2023**, *173*, 1–30. [[CrossRef](#)]

25. Xi, H.; Liu, W.; Waller, S.T.; Hensher, D.A.; Kilby, P.; Rey, D. Incentive-compatible mechanisms for online resource allocation in mobility-as-a-service systems. *Transp. Res. Part B Methodol.* **2023**, *170*, 119–147. [[CrossRef](#)]
26. Wong, Y.Z.; Hensher, D.A.; Mulley, C. Mobility as a service (MaaS): Charting a future context. *Transp. Res. Part A* **2020**, *131*, 5–19. [[CrossRef](#)]
27. Becker, H.; Balac, M.; Ciari, F.; Axhausen, K.W. Assessing the welfare impacts of Shared Mobility and Mobility as a Service (MaaS). *Transp. Res. Part A* **2020**, *131*, 228–243. [[CrossRef](#)]
28. Zajac, S.; Huber, S. Objectives and methods in multi-objective routing problems: A survey and classification scheme. *Eur. J. Oper. Res.* **2021**, *290*, 1–25. [[CrossRef](#)]
29. Androutsopoulos, K.N.; Zografos, K.G. Solving the multi-criteria time-dependent routing and scheduling problem in a multi-modal fixed scheduled network. *Eur. J. Oper. Res.* **2009**, *192*, 18–28. [[CrossRef](#)]
30. Martins, E.Q.V. On a multicriteria shortest path problem. *Eur. J. Oper. Res.* **1984**, *16*, 236–245. [[CrossRef](#)]
31. Artigues, C.; Huguet, M.J.; Gueye, F.; Schettini, F.; Dezou, L. State-based accelerations and bidirectional search for bi-objective multi-modal shortest paths. *Transp. Res. Part C Emerg. Technol.* **2013**, *27*, 233–259. [[CrossRef](#)]
32. Pulido, F.J.; Mandow, L.; de la Cruz, J.L.P. Multiobjective shortest path problems with lexicographic goal-based preferences. *Eur. J. Oper. Res.* **2014**, *239*, 89–101. [[CrossRef](#)]
33. Yao, X.; Li, W.; Pan, X.; Wang, R. Multimodal multi-objective evolutionary algorithm for multiple path planning. *Comput. Ind. Eng.* **2022**, *169*, 108145. [[CrossRef](#)]
34. Ahern, Z.; Paz, A.; Corry, P. Approximate multi-objective optimization for integrated bus route design and service frequency setting. *Transp. Res. Part B Methodol.* **2022**, *155*, 1–25. [[CrossRef](#)]
35. Cervantes-Sanmiguel, K.I.; Chavez-Hernandez, M.V.; Ibarra-Rojas, O.J. Analyzing the trade-off between minimizing travel times and reducing monetary costs for users in the transit network design. *Transp. Res. Part B Methodol.* **2023**, *173*, 142–161. [[CrossRef](#)]
36. Horn, M.E. An extended model and procedural framework for planning multi-modal passenger journeys. *Transp. Res. Part B* **2003**, *37*, 641–660. [[CrossRef](#)]
37. Liu, L.; Yang, J.; Mu, H.; Li, X.; Wu, F. Exact algorithms for multi-criteria multi-modal shortest path with transfer delaying and arriving time-window in urban transit network. *Appl. Math. Model.* **2014**, *38*, 2613–2629. [[CrossRef](#)]
38. Kumar, A.A.; Kang, J.E.; Kwon, C.; Nikolaev, A. Inferring origin-destination pairs and utility-based travel preferences of shared mobility system users in a multi-modal environment. *Transp. Res. Part B Methodol.* **2016**, *91*, 270–291. [[CrossRef](#)]
39. Gündling, F.; Hoch, P.; Weihe, K. Multi Objective Optimization of Multimodal Two-Way Roundtrip Journeys. In Proceedings of the RailNorrköping 2019. 8th International Conference on Railway Operations Modelling and Analysis (ICROMA), Norrköping, Sweden, 17–20 June 2019; Volume 69, pp. 350–360.
40. Liao, F.; Arentze, T.; Timmermans, H. Incorporating space–time constraints and activity–travel time profiles in a multi-state supernetwork approach to individual activity-travel scheduling. *Transp. Res. Part B Methodol.* **2013**, *55*, 41–58. [[CrossRef](#)]
41. Chen, B.Y.; Li, Q.; Lam, W.H. Finding the k reliable shortest paths under travel time uncertainty. *Transp. Res. Part B Methodol.* **2016**, *94*, 189–203. [[CrossRef](#)]
42. Redmond, M.; Campbell, A.M.; Ehmke, J.F. Reliability in public transit networks considering backup itineraries. *Eur. J. Oper. Res.* **2022**, *300*, 852–864. [[CrossRef](#)]
43. Häme, L.; Hakula, H. Dynamic journeying under uncertainty. *Eur. J. Oper. Res.* **2013**, *225*, 455–471. [[CrossRef](#)]
44. Socharoentum, M.; Karimi, H.A. Multi-modal transportation with multi-criteria walking (MMT-MCW): Personalized route recommender. *Comput. Environ. Urban Syst.* **2016**, *55*, 44–54. [[CrossRef](#)]
45. Georgakis, P.; Almohammad, A.; Bothos, E.; Magoutas, B.; Arnaoutaki, K.; Mentzas, G. MultiModal route planning in mobility as a service. In Proceedings of the WI '19: IEEE/WIC/ACM International Conference on Web Intelligence (WI '19 Companion), Thessaloniki, Greece, 14–17 October 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 283–291.
46. Liebig, T.; Piatkowski, N.; Bockermann, C.; Morik, K. Dynamic route planning with real-time traffic predictions. *Inf. Syst.* **2017**, *64*, 258–265. [[CrossRef](#)]
47. Choudhary, R.; Ratna, S.; Agarwal, A. Multimodal routing framework for urban environments considering real-time air quality and congestion. *Atmos. Pollut. Res.* **2022**, *13*, 101525. [[CrossRef](#)]
48. Wang, R.; Luo, Y.; Zou, Y.; Liu, S. Travel route planning method to avoid epidemic hot-spots in the post-epidemic era. *Transp. Res. Interdiscip. Perspect.* **2023**, *19*, 100802. [[CrossRef](#)]
49. Wright, S.; Nelson, J.D.; Cottrill, C.D. MaaS for the suburban market: Incorporating carpooling in the mix. *Transp. Res. Part A* **2020**, *131*, 206–218. [[CrossRef](#)]

50. Portouli, E.; Lytrivis, P.; Theodoropoulos, T.; Pantazopoulos, P.; Amditis, A. Can multimodal trip planning alter the travelers' mobility behavior? A cross-European study. In Proceedings of the IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), Rio de Janeiro, Brazil, 1–4 November 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 2119–2124.
51. Bast, H.; Delling, D.; Goldberg, A.; Müller-Hannemann, M.; Pajor, T.; Sanders, P.; Wagner, D.; Werneck, R.F. Route planning in transportation networks. In *Algorithm Engineering*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 19–80.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.