Generalised Time-Dependent Graphs for Fully Multimodal Journey Planning

Jan Hrnčíř[1] and Michal Jakob[1]

***Abstract*— We solve the fully multimodal journey planning problem, in which journey plans can employ any combination of scheduled public transport (e.g., bus, tram and underground), individual (e.g., walk, bike, shared bike and car), and on-demand (e.g., taxi) transport modes. Our solution is based on a generalised time-dependent graph that allows representing the fully multimodal earliest arrival problem as a standard graph search problem and consequently using general shortest path algorithms to solve it. In addition, to allow users to express their journey planning preferences and to speed up the search process, flexible journey plan templates can be used in our approach to restrict the transport modes and mode combinations permitted in generated journey plans. We have evaluated our solution on a real-world transport network of the city of Helsinki and achieved practically usable search runtimes in the range of hundreds of milliseconds.**

I. Introduction

The growing number of transport options available in modern cities raises the importance of tools that support travellers in finding journey itineraries that make the best use of available transport services while respecting traveller's individual needs. Existing journey planners fulfil such requirements only to a limited degree, in particular as they often only consider a certain subset of transport modes and their combinations, and as they only provide limited ways for users to express their preferences.

The journey planning problem is most often formalised as the *earliest arrival problem (EAP)*, i.e., the problem of finding the earliest arrival at a destination given a departure date and time from an origin. The earliest arrival problem has been widely studied and numerous algorithms and speed-up techniques exist for solving it on road network graphs and networks of public transport (PT) services. However, very limited work has been done on solving the earliest arrival problem for journey plans allowing general combinations of individual and public transport modes, the work of Horn [7] and Yu and Lu [13] being notable exceptions.

In this paper, we aim to address this gap. More specifically, we focus on solving the *fully multimodal* variant of the EAP. We use the term fully multimodal in order to stress that we consider modes and combinations thereof that go beyond what is supported in existing multimodal journey planners. In our approach, a journey can consist of any combination of scheduled PT modes (e.g., bus, tram and underground), individual modes (e.g., walk, bike, shared bike and car), and on-demand (e.g., taxi) modes.

[1]{hrncir,jakob}@agents.fel.cvut.cz, Agent Technology Center, Dept. of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, Praha, Czech Republic

We adopt a representation-centric approach to solving the fully multimodal EAP. Thus, instead of providing complex, purpose-specific journey planning algorithms, we introduce a *generalised time-dependent (GTD) graph* that allows representing the fully multimodal EAP as a standard graph search problem and consequently use general shortest path algorithms to solve it. We treat the problem in a deterministic setting assuming no uncertainty in any of the attributes of the planning graph.

Along with the GTD graph representation, we also introduce the concept of *journey plan templates*. Journey plan templates provide a powerful way of parameterising the operation of the planner and allow the user or the administrator of the journey planner to obtain plans that best meet their constraints and preferences. In addition, the templates constrain the search space and therefore speed up journey planning.

II. Related Work

As already indicated, the earliest arrival problem is a widely studied problem when considered separately for planning on road networks and for planning on networks of scheduled PT services. Existing work covers the whole spectrum from formal models of the problem, through solution algorithms up to practical consumer-oriented planning tools and services.

The road network variant of the EAP typically employs the direct graph representation of the road network. The road network is represented as a weighted directed graph $G = (V, E, \rho)$ where the set of nodes V represents junctions and the set of edges E represents roads. Each edge $(u, v) \in E$ is assigned a weight $\rho((u, v))$ specifying the time needed to travel across this edge. The road network graph is very sparse, almost planar, and usually has hierarchical properties. These properties are used to enhance basic shortest-path algorithms, such as A*, with speed-up techniques that accelerate graph search. The best known speed-up techniques include *SHARC* [2], *Landmark A** (ALT) [6], *highway hierarchies* [11], and *transit-node routing* [1].

For the scheduled PT variant of the EAP, there are two main ways to represent public transport timetables as a search graph. In the *time-expanded approach* [9], each event at a stop, e.g., the departure of a train, is modelled as a node in the graph; in the *time-dependent approach* [3], the graph only contains one node for each stop. To accelerate the search process, many speed-up techniques for basic shortest-path algorithm, e.g., Dijkstra's algorithm, have been proposed, including the *multi-level graph* approach [12], *access-node*

routing [4], and *core-ALT* [8]. Many of these techniques have been implemented as part of public online travel planning services.

In contrast to the EAP for road networks and for networks of scheduled PT services, only very limited research exists on the fully multimodal variant of the EAP. One of few exceptions is the planner proposed by Horn [7] which supports combinations of scheduled PT and on-demand transport services. A limitation of the Horn's approach is that the on-demand mode can only appear as the first or the last non-walk leg of a journey, i.e., the on-demand mode can only serve as a feeder service. The second attempt at solving the fully multimodal EAP is provided by Yu and Lu [13] who use a genetic algorithm to construct the sequence of transport modes in a journey plan. In their experiments, Yu and Lu permit walk, bus, underground, and taxi modes. However, the individual modes of transport (bike, shared bike and car) are not used.

Despite the limited research on the fully multimodal EAP, several online services exist capable of planning journeys employing non-trivial combinations of transport modes. For example, the AnachB[1] planner supports the combination of car and scheduled PT services. A major weakness is that the parking place (P+R) is not chosen optimally by the planner but needs to be selected manually by the user. The OpenTripPlanner[2] supports the combination of walking and riding a shared bike borrowed and later returned to one of the many city's bike sharing stations. However, the technical approaches behind these services have not been published and no guarantees about their optimality are known.

III. Generalised Time-dependent Graph

As mentioned in the introduction, our approach to solving the fully multimodal EAP relies on the newly proposed generalised time-dependent (GTD) graph, which allows representing the combined road network (for individual and on-demand modes) and PT network (for PT modes) in a single structure. The GTD graph is a generalisation of the time-dependent graph with constant transfer times defined by Pyrga et al. [9] (the time needed to make a transfer between two lines at a stop is defined as a constant for each stop). The generalised time-dependent graph G is constructed from the following three structures: (1) *time-dependent graph* G^T for the PT network; (2) *network graph* G^N for the network of pavements, cycleways, and roads; (3) *graph connector* D of the time-dependent graph G^T and the network graph G^N. The GTD graph's structure is shown in Figure 1. Below, we describe each part of the construction in detail.

A. Time-dependent Graph

To model the network of scheduled PT services (e.g., bus, tram, underground), we use a time-dependent graph $G^T = (V^T, E^T, \rho^T)$ with constant transfer times [9]. We have chosen this model for its better performance than the time-expanded model [10]. Let S be the set of *stop nodes*

[1] http://anachb.at/

[2] http://emtvalencia.es/geoportal/?lang=en_otp

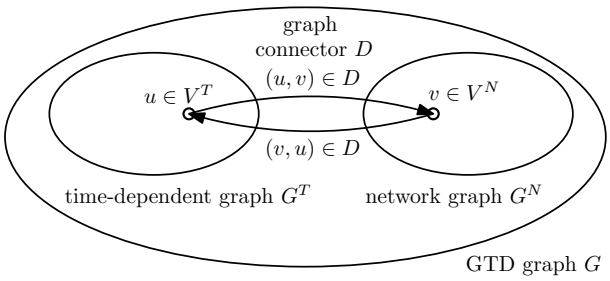

Fig. 1: The structure of a GTD graph

corresponding to the stops that are physically present in the PT network. A stop node can be served by one or more routes. A *route* is a set of PT vehicle trips that are known to the public under the same route number identifier, e.g., the tram line number 3. Assuming that n is the number of routes using a stop $u \in S$, then n *route nodes* $R_u = \{r_1^u, \ldots, r_n^u\}$, one for each route, are associated with stop u. Route nodes are virtual nodes without corresponding counterparts in the real world and they are used to model constant transfer times. Without route nodes, it would not be possible to model non-zero transfer times between different routes at the same stop. The set of all route nodes is denoted as $R = \cup_{u \in S} R_u$. The set of nodes V^T of the time-dependent graph G^T is then defined as $V^T = S \cup R$.

The set of edges E^T of the time-dependent graph G^T is defined as $E^T = A \cup B \cup C$ where A denotes the set of edges between route and stop nodes, B denotes the set of edges between stop and route nodes, and C denotes the set of *route edges* between route nodes of the same route. Edges $(v, w) \in A \cup B$ are called *transfer edges*. Formally, the sets are defined as follows:

$$
\begin{aligned}
A = &\ \cup_{u \in S} \{(r^u, u) | r^u \in R_u\} \\
B = &\ \cup_{u \in S} \{(u, r^u) | r^u \in R_u\} \\
C = &\ \cup_{u,v \in S} \{(r^u, r^v) | r^u \in R_u \wedge r^v \in R_v\} \text{ where } r^u \\
&\text{and } r^v \text{ are visited successively by the same route}
\end{aligned}
$$

The *link-traversal function* ${f'}_{(v,w)} : \mathbb{N} \to \mathbb{N}$ is associated with each edge $(v, w) \in C$ and defined as ${f'}_{(v,w)}(t) := t'$ where t is the departure time from v and $t' \geq t$ is the earliest possible arrival time at stop w. We assume that overtaking of vehicles on edges of the same route is not permitted. This means that the earliest arrival of a PT vehicle to a route node r_j^w corresponds to the earliest departure from an adjacent departure route node r_i^v.

Let the function g_v return the constant transfer time at stop v. For example in Figure 2, the transfer from a route node r_0^v to r_1^v and vice versa takes time g_v. Then the travel duration ${\rho^T}_{(v,w)} : \mathbb{N} \to \mathbb{N}$ of traversing an edge $(v, w) \in E^T$ from v at the departure time t is defined as

$$
{\rho^T}_{(v,w)}(t) := \begin{cases} 0 & \text{if } (v, w) \in A \\ g_v & \text{if } (v, w) \in B \\ {f'}_{(v,w)}(t) - t & \text{if } (v, w) \in C \end{cases}
$$

B. Network Graph

To model the network for individual modes of transport (e.g., walk, bike, shared bike and car) and on-demand modes

of transport (e.g., taxi), we use the *network graph* $G^N = (V^N, E^N, \rho^N)$ defined as a weighted directed graph, where the set of nodes V^N represents junctions and the set of edges E^N represents roads, pavements, and cycleways. The length of each edge $(v, w) \in E^N$ is given by the weight function $\rho^N : E^N \to \mathbb{R}_0^+$.

C. Graph Connector

In order to plan multimodal journeys using combinations of individual, on-demand, and PT modes of transport, the time-dependent graph G^T and the network graph G^N need to be interconnected. Let $\theta : S \to \mathcal{P}(\mathcal{V}^{\mathcal{N}})$ be a mapping that associates with each stop $v \in S$ a set of nodes $\theta(v) \in \mathcal{P}(\mathcal{V}^{\mathcal{N}})$ from the network graph. For the underground stops and large PT stations, the mapping assigns a stop a set of corresponding entrances from the network graph G^N. For the other PT stops, the mapping assigns to a stop a nearest pavement node from the network graph G^N.

Then the *graph connector* D of graphs G^T and G^N is defined as a set of interconnecting edges:

$$D = \{(v, w) | (v \in S \wedge w \in \theta(v)) \vee (v \in \theta(w) \wedge w \in S)\}$$

A length in metres $\rho_d((v, w)) = |v, w|$ is assigned to each $(v, w) \in D$ (the Euclidean distance between v and w is used).

D. GTD Graph

Finally, we can use the described structures to construct a unified network graph that supports multimodal journeys that use *any combination* of PT, individual, and on-demand modes of transport. Before defining the GTD graph, we define the edge weight ρ, the permitted modes function μ, and the permitted mode change predicate χ.

Firstly, let t be the departure time from node $v \in V$ and $vel \in \mathbb{R}^+$ the travel speed in metres per second. Then the *edge weight* $\rho_{(v,w)} : \mathbb{N} \times \mathbb{R}^+ \to \mathbb{N}$ returns the travel duration (in seconds) of traversing the edge $(v, w) \in E$ at time t using travel speed vel:

$$\rho_{(v,w)}(t, vel) := \begin{cases} \rho^T((u, w), t) & \text{if } (v, w) \in E^T \\ \rho^N((v, w))/vel & \text{if } (v, w) \in E^N \\ \rho_d((v, w))/vel & \text{if } (v, w) \in D \end{cases}$$

Secondly, assuming $M = \{m_1, \ldots, m_l\}$ is the set of all l supported modes of transport, the function $\mu : E \to \mathcal{P}(\mathcal{M})$ returns the set of permitted modes of transport $\mu((v, w)) \in \mathcal{P}(\mathcal{M})$ at an edge $(v, w) \in E$. In our approach, we currently use the following modes of transport: *walk (W), bike (I), shared bike (S), car (C), taxi (X), bus (B), tram (T)*, and *underground (U)*. Especially in the network graph G^N, there are usually several modes of transport permitted to use a given edge, e.g., car, taxi, and bike.

Thirdly, we need to capture the fact that certain changes of mode of transport are possible only at some nodes. For example, changing from walk to shared bike or vice versa is only possible at bike sharing stations. Formally, the *permitted mode change* predicate $\chi_v : M \times M$ is associated with each node $v \in V$ and $\chi_v(m_1, m_2)$ returns true if it is possible to change the mode of transport from m_1 to m_2 at node v.

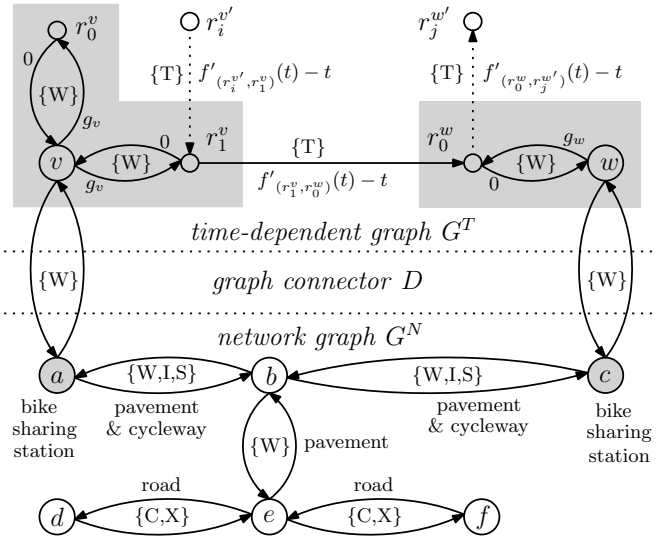

Fig. 2: An example of the GTD graph. Edges are annotated with the permitted modes of transport. Stop nodes $v, w \in S$ represent two tram stops that are connected by one tram route connecting four route nodes $(r_i^{v'}, r_1^v, r_0^w, r_j^{w'})$. Route nodes $R_v = \{r_0^v, r_1^v\}$ and $R_w = \{r_0^w\}$ associated with the respective stop nodes v and w are highlighted with grey background. Edges from the time-dependent graph G^T are also annotated with their weight (edge traversal time).

As an example, let $S^N \subset V^N$ be the set of bike sharing stations and $P^N \subset V^N$ be the set of park and ride (P+R) parking places. For the modes of transport currently used, the predicate $\chi_v(m_1, m_2)$ for each $v \in V$ and $m_1, m_2 \in M$ is defined as follows (`t` denotes true, `f` denotes false):

$$\chi_v(m_1, m_2) := \begin{cases} \texttt{t} & \text{if } v \in V^T & (1) \\ \texttt{t} & \text{if } v \in S^N \wedge ((m_1 m_2 = \text{WS}) \\ & \qquad \vee (m_1 m_2 = \text{SW})) & (2) \\ \texttt{t} & \text{if } v \in P^N \wedge ((m_1 m_2 = \text{CW}) & (3) \\ \texttt{t} & \text{if } m_1 = m_2 & (4) \\ \texttt{f} & \text{otherwise} & (5) \end{cases}$$

The defined predicate captures the five following rules: (1) change of mode of transport is not restricted for any stop $v \in V^T$; (2) change from walk to shared bike or vice versa is possible only at bike sharing stations $v \in S^N$; (3) change from car to walk is possible only at P+R parking places; (4) change from m_1 to $m_2 = m_1$ is not a change (it is always permitted to continue with the same mode of transport); (5) change of modes is not permitted in all other cases.

Finally, we define the *generalised time-dependent graph* as a weighted directed graph $G = (V, E, \rho, \mu, \chi)$ where $V = V^T \cup V^N$ and $E = E^T \cup E^N \cup D$. An example of a GTD graph is shown in Figure 2.

IV. Journey Planning Problem

In this section, we first describe the notions of a journey leg and a journey plan. Then, we define the fully multimodal earliest arrival problem.

A. Journey Plan

Let the journey leg be a part of a journey plan that is either covered by the traveller on foot or by a movement by one and only one vehicle from one location to another. Formally, the *journey leg* $L = ((v_1, w_1), \ldots, (v_k, w_k))$ is defined as a sequence of $|L| = k$ edges $(v_j, w_j) \in E$. Edges are a finer-grained decomposition of a journey leg and represent the lowest-level, atomic parts of any journey plan. Then the *journey plan* is a quadruple $\pi = (P, \sigma, \phi, \psi)$:

- $P = (L_1, \ldots, L_n)$ is a sequence of $|P| = n$ journey legs L_i.
- Function σ denotes the mode of transport $\sigma(L_i) \in M$ that is used for journey leg L_i.
- Function $\phi : E \to \mathbb{N}$ returns the departure time from v for each edge $(v, w) \in E$.
- Function $\psi : E \to \mathbb{N}$ returns the arrival time at w for each edge $(v, w) \in E$.

Let $L[j]$ be the j-th element of a sequence of elements L and $|L|$ be the number of elements in L. Let $\xi(P)$ be the *flattened plan* constructed as the concatenation of all edges in all journey legs $L_i \in P$:

$$\xi(P) = (L_1[1], \ldots, L_1[|L_1|], \ldots, L_n[1], \ldots, L_n[|L_n|])$$

B. Fully Multimodal Earliest Arrival Problem

The *fully multimodal earliest arrival problem* is a pair $J = (G, r)$, where:

- $G = (V, E, \rho, \mu, \chi)$ is a *GTD graph*
- $r = (o, d, t)$ is a *journey request* specifying an origin $o \in V$, a destination $d \in V$, and a time of departure $t \in \mathbb{N}$

A *journey plan* $\pi = (P, \sigma, \phi, \psi)$, where $P = (L_1, \ldots, L_n)$, is then a solution of the fully multimodal earliest arrival problem $J = (G, r)$ if and only if all the following conditions hold:

1) Journey plan starts at the origin:
   $o = v$ where $(v, w) = L_1[1]$
2) Journey plan ends at the destination:
   $d = w$ where $(v, w) = L_n[|L_n|]$
3) All edges are present in the GTD graph:
   $\forall (v, w) \in \xi(P) : (v, w) \in E$
4) Edges form a path in the GTD graph:
   $\forall j \in \{1, \ldots, |\xi(P)| - 1\} :$
   $(v, w) = \xi(P)[j] \wedge (w, x) = \xi(P)[j + 1]$

V. Journey Planning Problem with Templates

For the fully multimodal earliest arrival problem with templates, we introduce the notion of a *journey plan template*. As mentioned in the introduction, journey plan templates give users and journey planner administrators a powerful way of parameterising the journey planner to obtain plans that best meet their constraints and preferences. For instance, a journey plan template that prefers environmentally friendly modes of transport can be designed by a journey planner administrator (e.g., a combination of walk and shared bike).

A. Journey Plan Template

A journey plan template constrain the journey plan in the permitted combination of modes on the level of journey legs. A *journey plan template* τ is defined as a regular expression over the transport modes alphabet M. As an example, we list three templates[3]:

- Taxi only: `^X$`
- Walk and PT: `^W((B|T|U)W)*$`
- Walk and shared bike: `^W(SW)?$`

We define several notions related to the journey plan templates. Let the word $\sigma(L_1) \ldots \sigma(L_n)$ be the *mode sequence* $\kappa(P)$ of a sequence of journey legs $P = (L_1, \ldots, L_n)$. Empty mode sequence $\kappa(\emptyset) = \epsilon$. We say that a sequence of journey legs P match a journey plan template τ if and only if the mode sequence $\kappa(P)$ matches the regular expression τ. Next, let $modes(\tau)$ be the set of modes of transport that are present in a template τ. Finally, the binary operator $\|$ over a mode sequence $m_1 \ldots m_n$ and a mode of transport $m \in M$ is defined as follows:

$$m_1 \ldots m_n \| m := \begin{cases} m_1 \ldots m_n & \text{if } m = m_n \\ m_1 \ldots m_n m & \text{otherwise} \end{cases}$$

B. Fully Multimodal EAP with Templates

The fully multimodal EAP with templates adds the notion of journey plan template to the fully multimodal EAP. Thus, the *fully multimodal earliest arrival problem with templates* is a triple $J = (G, r, \tau)$, where:

- $G = (V, E, \rho, \mu, \chi)$ is a *GTD graph*
- $r = (o, d, t)$ is a *journey request*
- τ is a *journey plan template*

A *journey plan* $\pi = (P, \sigma, \phi, \psi)$ is then a solution of the fully multimodal earliest arrival problem with templates $J = (G, r, \tau)$ if and only if all the following conditions hold:

1) Journey plan π is a solution of the fully multimodal earliest arrival problem $J = (G, r)$.
2) Journey legs $P = (L_1, \ldots, L_n)$ match the journey plan template, i.e., $\sigma(L_1) \ldots \sigma(L_n)$ matches τ.

VI. Solution Method

In this section, we present a method to solve the fully multimodal earliest arrival problem with templates using the GTD graph representation. The method uses a contextual view over the underlying GTD graph in order to use general shortest path algorithms to find the journey plans in the search space. This is enabled by storing the node context, i.e., the time of arrival and the modes of transport sequence used, in the contextual GTD graph.

A. Contextual GTD Graph

The contextual GTD graph is a *view* over an underlying GTD graph. The contextual GTD graph serves two main purposes. First, it allows filtering the available edges in the GTD graph with respect to the permitted modes of transport specified by a given journey plan template τ. Second, it

[3]POSIX Extended Regular Expression syntax is used.

Function 1 Outgoing edges of a contextual node

Input: A contextual node (v, t_a, m_s) and a template τ

Output: A set of outgoing edges from (v, t_a, m_s) given τ

```
1: function OUT((v, t_a, m_s), τ)
2:     O := ∅
3:     for all (v, w) ∈ E do
4:         for all m ∈ μ((v, w)) do
5:             m'_s := m_s ‖ m
6:             m_prev := m_n where m_s = m_1 ... m_n
7:             a := χ_v(m_prev, m)
8:             b := m'_s matches τ
9:             if m ∈ modes(τ) ∧ a ∧ b then
10:                t' := t_a + ρ_(v,w)(t_a, λ(m))
11:                O := O ∪ ((v, t_a, m_s), (w, t', m'_s))
12:            end if
13:        end for
14:    end for
15:    return O
16: end function
```

Function 2 Path to journey plan transformation

Input: A path K in G_τ

Output: A journey plan $\pi = (P, \sigma, \phi, \psi)$

```
1: function DERIVEJOURNEYPLAN(K)
2:     i := 0
3:     for all ((v, t_a, m_s), (v', t'_a, m'_s)) ∈ K do
4:         if m_s ≠ m'_s then
5:             i := i + 1
6:             L_i := ()
7:             σ(L_i) := m_j where m'_s = m_1 ... m_j
8:         end if
9:         L_i := L_i ∘ (v, v')
10:        φ((v, v')) := t_a
11:        ψ((v, v')) := t'_a
12:    end for
13:    P := (L_1, ..., L_i)
14:    return (P, σ, φ, ψ)
15: end function
```

allows checking that the current partial journey plan matches a given journey plan template τ during the search process.

Let us define the graph formally. The *contextual GTD graph* G_τ over a GTD graph $G = (V, E, \rho, \mu, \chi)$ using a journey plan template τ is defined as $G_\tau = (V_\tau, E_\tau, \rho, \mu, \chi, \lambda)$. V_τ is a set of *contextual nodes* defined as triples (v, t_a, m_s) where:

- $v \in E$ is a node in the GTD graph
- $t_a \in \mathbb{N}$ is the arrival time at v
- m_s is a mode sequence $\kappa(P')$ of a sequence of journey legs P' from origin o (taken from the input journey request r) to node v

The context of contextual nodes corresponds to a GTD graph traversal at certain time using specific modes of transport, cf. Figure 3.

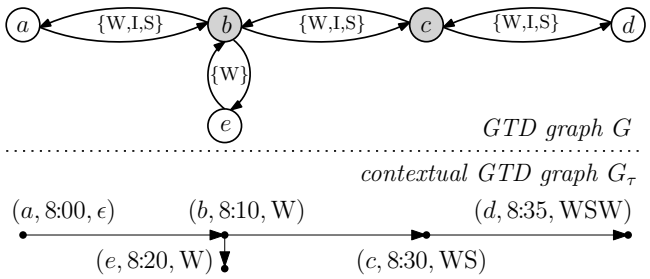

Fig. 3: An example of a GTD graph and its corresponding contextual GTD graph searched using the walk and shared bike template. Origin is set to a at 8:00; destination is set to d. Grey nodes b and c represent bike sharing stations. The bottom part of the figure shows how the contextual information is represented using the contextual nodes in the contextual GTD graph G_τ.

Let the function $\lambda : M \to \mathbb{R}^+$ returns the travel speed $\lambda(m)$ for a mode of transport $m \in M$. Then the set of contextual nodes V_τ and the set of edges E_τ is constructed using the origin contextual node (o, t, ϵ) and the function $\text{OUT}((v, t_a, m_s), \tau)$ (cf. Function 1) that returns the outgoing edges for a contextual node (v, t_a, m_s) and a template τ. At line 9 of Function 1, it is checked that a mode of transport m is present in the template τ, that a mode change from m_{prev} to m is permitted and that the current mode sequence m'_s matches the journey plan template τ.

The advantage of the contextual GTD graph is that unmodified general shortest path algorithms (e.g., A* or Dijkstra) can be used to find journey plans. This is enabled by embedding the domain information (e.g., permitted modes of transport and checking against a journey plan template) in the contextual GTD graph.

From the implementation point of view, the contextual GTD graph can be constructed *on request*. The nodes and edges are created on request only when they are needed during the search process of the respective shortest path algorithm.

B. Algorithm Specification

Now we present how the contextual GTD graph is used to solve the fully multimodal EAP with templates $J = (G, r, \tau)$. The input of the algorithm is an instance of the problem $J = (G, r, \tau)$ and the output is a journey plan $\pi = (P, \sigma, \phi, \psi)$ that solves the problem $J = (G, r, \tau)$. The algorithm works in two phases:

1) Shortest path algorithm on contextual GTD graph
2) Journey plan derivation

In the first phase, a general shortest path algorithm (e.g., A* or Dijkstra) is used to find a path $K = ((x_1, x_2), (x_2, x_3), \ldots, (x_k, x_{k+1}))$ of length $|K| = k$ in the contextual GTD graph $G_\tau = (V_\tau, E_\tau, \rho, \mu, \lambda)$ from the origin contextual node (o, t, ϵ) to the destination contextual node $(d, \cdot, \cdot)$. The edge weight function $\rho_{(v,w)}$ at line 10 of Function 1 returns the duration of traversing an edge

TABLE I: Size of the Helsinki GTD graph

Graph name	Graph	Nodes	Edges
Time-dependent graph	G^T	50,320	112,127
Network graph	G^N	207,240	585,937
Graph connector	D	-	14,980
GTD graph	G	257,560	713,044

$(v, w) \in E$, therefore the journey plan is optimised with respect to its duration (i.e., the earliest arrival problem is solved).

In the second phase, the path K found in the contextual GTD graph G_τ is transformed into a journey plan $\pi = (P, \sigma, \phi, \psi)$. This is done using the DERIVEJOURNEYPLAN(K) function, cf. Function 2. The function iterates over edges $((v, t_a, m_s), (v', t'_a, m'_s)) \in K$. Every time the mode sequence is changed, a new journey leg L_i is created and its mode $\sigma(L_i)$ set. The edge $(v, v') \in E$ is then added to the current journey leg L_i using the operator $\circ$ that appends an element to a sequence and the departure $\phi(v, v')$ and arrival $\psi(v, v')$ is set.

It is important to note that if the shortest path algorithm used in the first phase of the algorithm is optimal, then the solution of the fully multimodal EAP with templates is optimal with respect to journey plan duration and the journey plan template τ.

VII. EVALUATION

Our proposed approach has been evaluated on real-world PT and road network data for Helsinki. The main purpose of the evaluation was to confirm that the GTD graph representation is flexible enough to allow successfully planning fully multimodal journeys with a variety of mode combinations. We were also interested in measuring how fast the GTD graph can be searched using standard algorithms – the runtimes results should, however, be treated as preliminary because we have not yet applied any speed-up techniques or other optimisation methods.

A. Data

Helsinki covers the area of 600 square kilometres. Kalkati.net XML database dump[4] provided by the Helsinki Regional Transport Authority (HSL) has been used as the data source for scheduled PT services. The data has been converted to the widely used General Transit Feed Specification (GTFS)[5] data format which is then used to construct the time-dependent graph G^T. OpenStreetMap[6] has been used as a data source for the network graph G^N. Basic statistics about the size of the GTD graph and its components are given in Table I. A fragment of the GTD graph is visualised in Figure 4. Note that in Helsinki, there are currently no bike sharing stations. For experimentation purposes, 150 bike sharing stations have therefore been added – the locations of the stations were chosen randomly with the uniform distribution over the nodes V^N of the network graph G^N. In addition, P+R parking places are not properly set in the OpenStreetMap data. For experimentation purposes, 10 P+R parking places were manually inserted into the map at the border of the Helsinki city centre.

[4] http://developer.reittiopas.fi/pages/en/kalkati.net-xml-database-dump.php

[5] https://developers.google.com/transit/gtfs/

[6] http://openstreetmap.org/

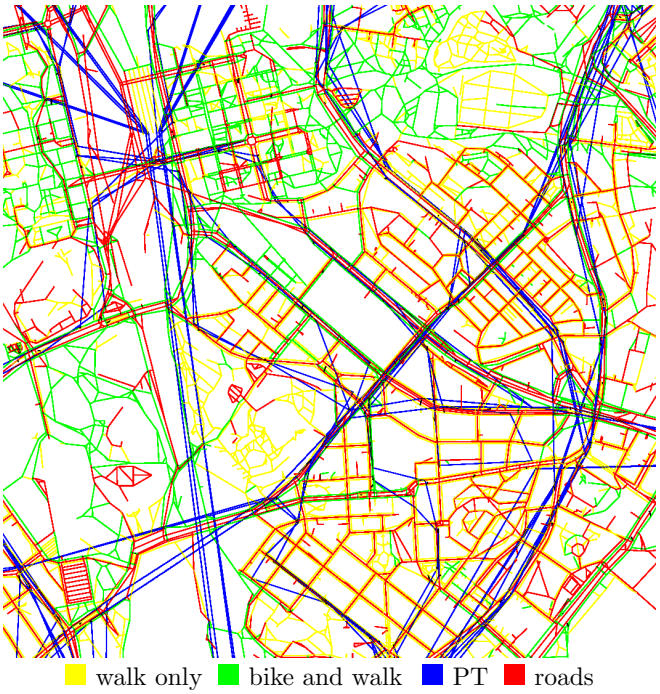

Fig. 4: Visualisation of a 2.4 km by 2.4 km fragment of the Helsinki GTD graph. Edge colours denote the modes of transport permitted at each edge, cf. legend. All other combinations of modes (e.g., car and taxi, bike only) are marked red. There are approximately 9,500 nodes and 27,700 edges in the visualisation.

B. Experiment Settings

We used seven journey plan templates $\tau \in T_7$ for the evaluation, cf. Table II. The templates have been chosen to reflect the typical combinations of modes used in modern multimodal transport systems. To allow a unified description of the results, we treat the fully multimodal EAP (without templates) as equivalent to the fully multimodal EAP with templates using the *empty template* permitting any combination of modes.

A* and Dijkstra's algorithms have been used to find a journey plan π given $J = (G, r, \tau)$. A* uses a duration heuristic $h(v)$ calculated as $h(v) = |v, d|/vel_{max}$ where $|v, d|$ is the Euclidean distance between current node v and destination node d, vel_{max} is the speed of the underground set to 120 km/h.

Following initial experiments, the better algorithm of the two has been chosen for each template $\tau \in T_7$. The templates $\tau \in T_7$ and their corresponding chosen algorithms are listed in Table II. Consequently, all templates use A* except the walk and PT template and car, walk and PT template where the Euclidean distance heuristic slows-down the A* search process [6] so the Dijkstra's algorithm is used.

TABLE II: Journey plan templates used in the evaluation, along with the best performing algorithm for each template

Template name	Template regexp	Algorithm
Walk only	`^W$`	A*
Bike only	`^I$`	A*
Taxi only	`^X$`	A*
Walk and PT	`^W((B|T|U)W)*$`	Dijkstra
Car, walk and PT	`^CW((B|T|U)W)*$`	Dijkstra
Walk and shared bike	`^W(SW)?$`	A*
Empty template	N/A	A*

TABLE III: Average runtimes in milliseconds

Template name	Short	Medium	Long
Walk only	24	135	417
Bike only	15	60	178
Taxi only	31	103	239
Walk and PT	488	817	939
Car, walk and PT	384	477	504
Walk and shared bike	87	223	440
Empty template	376	758	891

The set of instances of the fully multimodal EAP with templates Q for the experiment were created in the following way. First, $n = 10{,}000$ origin-destination-departure triples $Q_t = ((o_1, d_1, t_1), \ldots, (o_n, d_n, t_n))$ were sampled using the uniform distribution over the coordinates of Helsinki area and the uniform distribution over the time interval from 8:00 to 18:00 on 17 Jan 2013. The maximum origin-destination distance was set to 40 km to exclude long trips that are not usual in the urban setting.

Then the origin and destination coordinates were converted to origin and destination nodes from graph G. Let $\delta(c, m)$ be a function that returns the nearest node in the GTD graph G given a coordinate c and a mode of transport m. For example, for the walk mode, the nearest node on a pavement is returned. Then the set of $|Q| = 70{,}000$ instances of the fully multimodal EAP with templates is constructed. Each of the origin-destination-departure triples Q_t is combined with all journey templates as follows:

$$Q = \{(G, (\delta(o, m_1), \delta(d, m_n), t), \tau) | \\ (o, d, t) \in Q_t \wedge \tau = m_1 \ldots m_n \in T_7\}$$

C. Implementation

The algorithm is implemented in JAVA 7. The results obtained are based on running the algorithm on one core of a 3.2 GHz Intel Core i7 processor of a Linux desktop computer with OpenJDK IcedTea7 2.3.7. The PostgreSQL 9.1 database spatially enabled with PostGIS 2.0.1[7] was used for storing and retrieving the data for the time-dependent graph G^T and the network graph G^N. The Osmosis 0.41[8] tool has been used to cut the Helsinki area from the OSM data dump and to put the data in the PostgreSQL database. Both the A* and Dijkstra's algorithm use the Fibonacci heap [5] implementation from the JGraphT 0.8.3[9] library.

[7] `http://postgis.net/`
[8] `http://wiki.openstreetmap.org/wiki/Osmosis`
[9] `http://jgrapht.org/`

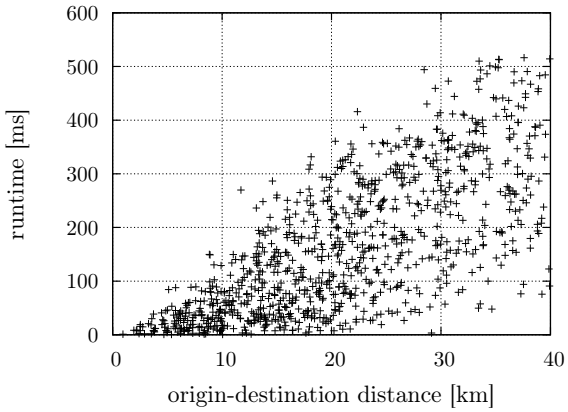

Fig. 5: Runtime against origin-destination distance (taxi only template, 1000 randomly selected requests)

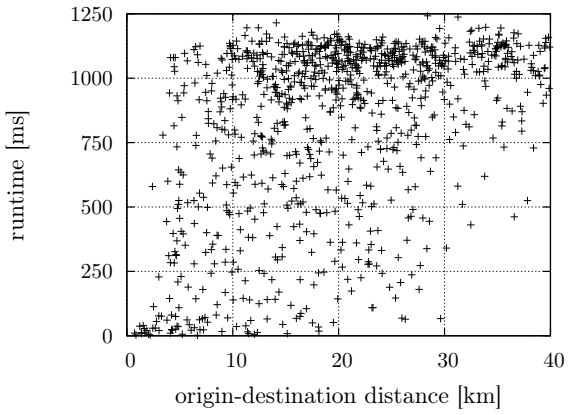

Fig. 6: Runtime against origin-destination distance (walk and PT template, 1000 randomly selected requests)

Geographical locations of all nodes in the OSM data and stops in the GTFS data are represented as their longitude and latitude values using the World Geodetic System (version WGS 84). WGS 84 is a *geographic* coordinate system type identified by SRID 4326[10] (Spatial Reference System Identifier). In order to simplify the complex calculation of the Euclidean distance between two nodes expressed in the WGS 84 coordinates (the calculation is very frequently used in the A* Euclidean distance heuristic), we use a *projected* coordinate system. The projected coordinate system is regional and projects the location from a spheroid to a plane. For locations in Helsinki, the spatial reference system "KKJ / Finland zone 2" with SRID 2392[11] is used.

D. Results

A solution for each problem instance $J \in Q$ has been computed. All instances are divided into three sets based on the distance of their origin and destination location: short (below 10 km), medium (10–20 km), and long (20–40 km). Average runtimes in milliseconds for each journey plan template and origin-destination distance interval are shown in Table III.

[10] `http://spatialreference.org/ref/epsg/4326/`
[11] `http://spatialreference.org/ref/epsg/2392/`

Runtimes for all journey plan templates (except templates containing PT) are better than the runtimes for the empty template. This empirically confirms that the journey plan templates constrain the search space of the planner, which results in lower runtimes than when the empty template, which permits any combination of modes, is used. Runtimes for the empty template are better than the runtimes for templates containing PT because the heuristic of the A* algorithm leads the planner well into the destination using the taxi mode (for the majority of requests, taxi is the fastest mode of transport with the lowest journey plan duration).

In general, the templates containing more than one mode of transport are more difficult for the planner (higher branching factor and a larger contextual GTD graph) resulting in higher runtimes than the single-mode templates. Template with the lowest runtimes is the bike only template where the average runtimes range from 15 ms for the short requests up to 178 ms for the long requests. Template with the highest average runtimes is the walk and PT template where the average runtimes ranges from 488 ms for the short requests up to 939 ms for the long requests. The runtimes of car, walk and PT template are lower than the runtimes of the walk and PT template because a significant part of the journey is covered by car and only the last part from the P+R parking place to the destination by walk and PT modes.

Figures 5 and 6 show scatter plots of the search runtime versus the origin-destination distance for 1000 randomly selected requests. It can be observed that runtimes for the taxi only template in Figure 5 are more strongly correlated on the origin-destination distance than the runtimes of the walk and PT template in Figure 6.

E. Discussion

Compared to the algorithms employing state-of-the-art speed-up techniques specifically designed for road network and public transport network variant of EAP, the search times of our method are high. There are several reasons for such a behaviour. First and most importantly, the GTD graph representation is significantly more expressive and flexible, enabling searching for plans from a much richer family of journey plans, which necessarily increases the method's computational cost. Second, no speed-up techniques have yet been applied to accelerate the search of the contextual GTD graph. Last, the algorithm is currently implemented in JAVA whereas the search algorithms employing state-of-the-art speed-up techniques are usually implemented in C++. That said, even without the use of speed-up techniques and other optimisations, our method achieves practically usable runtimes.

So far, seven journey plan templates have been used in the evaluation. In the future, we plan to add the following useful plan templates:

- Taxi and PT: `^X?W((B|T|U)W)*X?$`
  A taxi can be used for covering the first, the last, or both first and last journey legs.
- Bike and PT: `^IW(UW(IW)?)*I?$`
  A traveller uses his or her own bike to get from an origin to a destination. Where possible and beneficial, PT mode of transport that permits taking bike along is used (in this example only the underground permits it).

VIII. Conclusion

We have presented a novel method for multimodal journey planning that allows finding multi-leg journeys utilising transport modes and combinations thereof not supported by existing journey planners. At the core of our method is a novel, generalised time-dependent graph representation which allows representing the fully multimodal journey planning problem with templates as a graph search problem that can be solved by general graph search algorithms. Experiments on realistic network data about the Helsinki transport system confirmed the viability of the approach – the planner was able to find a diverse set of journey plans and achieve runtimes which, although noticeably higher compared to algorithms optimised for basic variants of the earliest arrival problem, are generally usable and are likely to be significantly improved after the preliminary implementation of the approach is optimised.

Acknowledgment

This work was supported by the European Union Seventh Framework Programme FP7/2007-2013 (grant agreement no. 289067), by the Ministry of Education, Youth and Sports of Czech Republic (grant no. LD12044 and 7E12065) and by the Czech Technical University (grant no. SGS13/210/OHK3/3T/13).

References

[1] H. Bast, S. Funke, P. Sanders, and D. Schultes. Fast Routing in Road Networks with Transit Nodes. *Science*, 316(5824):566, 2007.

[2] R. Bauer and D. Delling. SHARC: Fast and robust unidirectional routing. *ACM Journal of Experimental Algorithmics*, 14, 2009.

[3] G. S. Brodal and R. Jacob. Time-dependent Networks as Models to Achieve Fast Exact Time-table Queries. *Electronic Notes in Theoretical Computer Science*, 92(0):3–15, 2004.

[4] D. Delling, T. Pajor, and D. Wagner. Accelerating Multi-modal Route Planning by Access-Nodes. In *ESA*, volume 5757 of *Lecture Notes in Computer Science*, pages 587–598. Springer, 2009.

[5] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34(3):596–615, 1987.

[6] A. V. Goldberg and C. Harrelson. Computing the shortest path: A* search meets graph theory. In *Proceedings of the 16th annual ACM-SIAM symposium on Discrete algorithms*, Philadelphia, USA, 2005.

[7] M. Horn. Multi-modal and demand-responsive passenger transport systems: a modelling framework with embedded control systems. *Transportation Research Part A: Policy and Practice*, 36(2):167–188, 2002.

[8] T. Pajor. Multi-Modal Route Planning. Master's thesis, 2009.

[9] E. Pyrga, F. Schulz, D. Wagner, and C. Zaroliagis. Efficient models for timetable information in public transportation systems. *Journal of Experimental Algorithmics (JEA)*, 12, 2008.

[10] E. Pyrga, F. Schulz, D. Wagner, and C. D. Zaroliagis. Experimental Comparison of Shortest Path Approaches for Timetable Information. In *Proceedings of the 6th Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 88–99, 2004.

[11] P. Sanders and D. Schultes. Highway Hierarchies Hasten Exact Shortest Path Queries. In *ESA*, volume 3669 of *Lecture Notes in Computer Science*, pages 568–579. Springer, 2005.

[12] F. Schulz. *Timetable information and shortest paths*. PhD thesis, 2005.

[13] H. Yu and F. Lu. A Multi-Modal Route Planning Approach With an Improved Genetic Algorithm. In *Joint International Conference on Theory, Data Handling and Modelling in GeoSpatial Information Science*, pages 343–348, 2010.