VRPLib: a library of instances for time-dependent routing problems

Tommaso Adamo[a], Gianpaolo Ghiani[a], Emanuela Guerriero[a]

[a]*Department of Engineering for Innovation, Università del Salento, Via Monteroni 73100 Lecce, Italy*

1. Introduction

For an in depth review on the time-dependent routing problems we refer the reader to Gendreau et al. [1] and Adamo et al. [2]. Details on construction of road graph with realistic travel times in Ghiani et al. [3].

2. Travel time model

Let's define a graph $G = (V, A)$, in which $V = \{1, \ldots, n\}$ is a set of vertices and A is a set of arcs. Each arc $(i, j) \in A$ has a non-negative length L_{ij} constant over time.

The time horizon $[0, T]$ is partitioned into a finite number H of time slots $[T_h, T_{(h+1)}]$ $(h = 0, \ldots, H-1)$, where $T_0 = 0$ and $T_H = T$. Differently from the contributions prior to Ichoua et al. [4], where the traversal time on arc $(i, j) \in A$ was constant during each time slot h, i.e. $\tau_{ij}(t) = \frac{L_{ij}}{v_{ijh}}$ (see Hill and Benton [5]) or linear piecewise, i.e. $\tau_{ij}(t) = \frac{L_{ij}}{v_{ij}(t)}$ (see Horn [6]), we assume that the speed v_{ijh} is constant during each time interval and the travel time functions $\tau_{ij}(t)$ is continuous piecewise linear. Each $\tau_{ij}(t)$ can be computed through Algorithm 1, firstly introduced by Ichoua et al. [4](IGP model).

Algorithm 1 Travel time $\tau_{ij}(t)$ calculation procedure in the IGP model

$t_0 \leftarrow t$
$h \leftarrow p : T_p \leq t \leq T_{p+1}$
$d \leftarrow L_{ij}$
$t' \leftarrow t + d/v_{ijp}$
while $t' > T_{h+1}$ **do**
    $d \leftarrow d - v_{ijh}(T_{h+1} - t)$
    $t \leftarrow T_{h+1}$
    $h \leftarrow h + 1$
    $t' \leftarrow t + d/v_{ijh}$
end while
return $t' - t_0$

The main reason to adopt the above model is to satisfy the first-in-first-out (FIFO) property, that is, the arrival time is a strictly monotonic function of the starting time. In particular, as stated by Fleischmann et al.

Email addresses: tommaso.adamo@unisalento.it (Tommaso Adamo), gianpaolo.ghiani@unisalento.it (Gianpaolo Ghiani), emanuela.guerriero@unisalento.it (Emanuela Guerriero)

[7], the FIFO property is assured if the slope of $\tau_{ij}(t)$ is strictly greater than -1. Any continuous piecewise linear travel time model can be derived from a suitable IGP model. In particular, if the traversal time respects the FIFO property, then speed values are non negative. In addition, some lower bounding procedures can be derived for many routing problems.

It is possible to generate an IGP model starting from a suitable continuous piecewise linear $\tau_{ij}(t)$. As described by Ghiani and Guerriero [8], the procedure to calculate the IGP model is composed by two phases: the determination of the speed breakpoints and the computation of the associated speeds. Given a start time t and path composed by a sequence of k nodes, i.e. $p_n = (i_0, i_1, \ldots, i_n)$ with $i_k \in V$ and $k = 0, \ldots, n$, the time needed to arrive from node i_0 to node i_n can be recursively defined as:

$$z(p_k, t) = z(p_{k-1}, t) + \tau_{i_{k-1}, i_k}(t + z(p_{k-1}, t) \quad k = 1, \ldots, n$$

and initialization $z(p_0, t) = 0$. The main results by Ghiani and Guerriero [8] relies on the proof that the IGP model is suitable for both a road-network as well as a customer based graph. As stated in [2], starting from raw data (most often, GPS vehicle traces), forecasting methods aim to predict travel time functions over road networks and customer-based graphs. This is done through the three-stage process depicted in Figure 1: travel speed predictions; arc travel time queries, quickest path queries. In the following we propose data input format underlying the second and third stage of process.

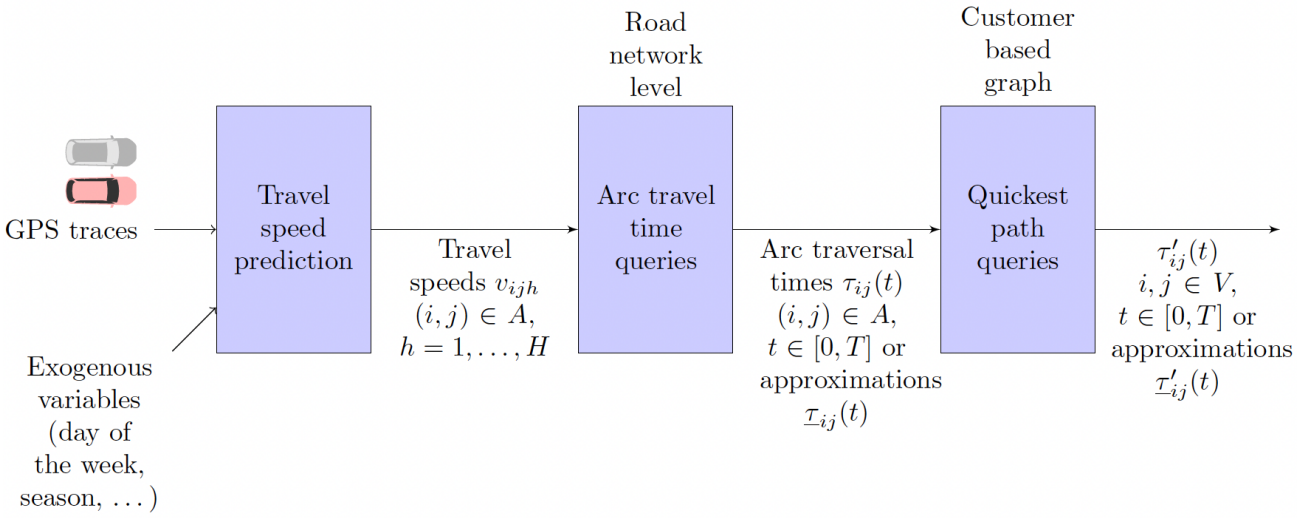

Figure 1: From GPS vehicle traces to quickest path queries proposed in [2]

3. Road graph data format

The road graph is described in plain text (*"rg.txt"*). The first two lines form a preamble reporting the number of vertices and arcs in the graph.

`VERTICES <NUMBER_OF_VERTICES>\n`

ARCS <NUMBER_OF_ARCS>\n

Subsequent lines contain information describing each arc in the format:

<TAIL_ID> <HEAD_ID> <LENGTH> <MAX_SPEED> <H> <X_0> <Y_0> <X_1> <Y_1> ... <X_H> <Y_H>\n

Each component is separated by a blank space, while each line is terminated by a carriage return. The items are as follows:

- <TAIL_ID> is an integer value identifying a node in the graph that is the tail of the current arc;
- <HEAD_ID> is an integer identifying a node in the graph that is the head of the current arc;
- <LENGTH> is a floating point number representing the length of the arc;
- <MAX_SPEED> is a floating point number representing the free-flow speed of a time-dependent arc;
- <H> the number of pieces composing the speed step function;
- <X_h> a floating point value representing a speed breakpoint, for $h = 0, \ldots, H$;
- <Y_h> a floating point value representing the jam factor for the speed value at time X_h, for $h = 0, \ldots, H$.

4. Customer-based complete graph data format

On top of the road graph, a customer-based graph (a directed complete graph) is generally constructed. A customer-based graph instance consists in a folder (or a deflated ZIP file) containing different plain text files:

- *"topology.txt"*: a mandatory graph topology file,
- *"tw.txt"*: an optional time windows file, and
- *"approximations.txt"*: an optional approximations file.

Additionally, we can find the IGP parameters (constant length and speed step function for each arc) as provided by Ghiani and Guerriero [8] in an optional file (*"igp.txt"*) with the same structure of a road graph (see Section 3).

4.1. Graph Topology File

This file reports a detailed description of the graph structure: customers, arcs, and the traversal time function associated to each arc.

We describe the graph using a plain text file reporting the following information:

CUSTOMERS <N>\n

The subsequent $N^2 - N$ lines contain information describing an arc in the format:

<TAIL_ID> <HEAD_ID> <H> <X_0> <Y_0> <X_1> <Y_1> ... <X_H> <Y_H>\n

Each component is separated by a blank space, while each line is terminated by a carriage return. The items are as follows:

- <N> the number of customers in the graph.
- <TAIL_ID> is an integer value identifying a node in the graph that is the tail of the current arc;
- <HEAD_ID> is an integer identifying a node in the graph that is the head of the current arc;
- <H> the number of pieces composing the traversal time function;
- <X$_h$> a floating point value representing a time breakpoint, for $i = 0, \ldots, H$;
- <Y$_h$> a floating point value representing the traversal time $\tau(\texttt{X}_h)$, for $h = 0, \ldots, H$.

4.2. Time Windows

A set of time windows $[a_i, b_i]$ for any customer $i = 1, \ldots, N$ can be provided in a file reporting N lines using the syntax:

<a$_i$> <b$_i$>\n

where:

- <a$_i$> is the opening time of customer i,
- <b$_i$> is the closing time of customer i.

4.3. Approximated travel times

Given a customer-based topology file and a time windows file, a Python module can be used to compute the approximated travel time defined by Adamo et al. [10]. The output will be a plain text file reporting:

COMMON_SPEED_PROFILE <H> <X$_0$> <Y$_0$> <X$_1$> <Y$_1$> ... <X$_H$> <Y$_H$>\n

where:

- <H> is the number of pieces composing the speed step function;
- <X$_h$> a floating point value representing a speed breakpoint, for $h = 0, \ldots, H$;
- <Y$_h$> a floating point value representing the speed value at time X$_h$, for $h = 0, \ldots, H$.

The subsequent $N^2 - N$ lines contain the minimum and maximum length for each arc:

<TAIL_ID> <HEAD_ID> <MIN_LENGTH> <MAX_LENGTH>\n

Moreover, the last N lines report the scaled time windows:

<A$_i$> <B$_i$>\n

where:

- <A$_i$> is the scaled opening time of customer i computed as:

$$A_i = \int_0^{a_i} y(\mu) d\mu$$

- <B$_i$> is the scaled closing time of customer i computed as:

$$B_i = \int_0^{b_i} y(\mu) d\mu$$

where $y(t)$ is the common speed profile.