CASPT 2018

Tradeoff Between Processing Time and Solution Quality for an A*-Guided Heuristic Applied to a Multi-Objective Bus Passenger Trip Planning Problem

Sylvain M. R. Fournier · Eduardo Otte Hülse · Éder Vasco Pinheiro

Abstract The Bus Passenger Trip Planning Problem is the decision problem the bus passenger faces when he has to move around the city using the bus network: how and what time will he get to the destination? Or possibly: given a fixed time to get to the destination, what time will he have to leave? We show that both questions are computationally equivalent and can be answered using an A*-guided and Pareto domination-based heuristic. The A* procedure drives the search estimating the time of arrival until the target node, even in intermediate nodes. Domination is triggered each time a new label is generated, in order to prune out labels defining subpaths with high values for the objectives we focus on: arrival time at destination, number of transfers and total walking distance. We discuss tradeoff between processing time and the solution quality through a parameter called A* speed. The tool is available for transit users on a day-to-day basis in Brazilian cities of up to about 800,000 inhabitants and returns a variety of solutions within a couple of seconds at most.

Keywords Trip Planning Problem · Pareto Border · A* algorithm

1 Context

Stimulating people to switch their means of transportation from individual to public alternatives is an important point in addressing the problem of mobility

Sylvain M. R. Fournier · Eduardo Otte Hülse · Éder Vasco Pinheiro
WPLEX Software Ltda.
Rod SC 401, 8600 - Bloco 5, Sala 101
Santo Antônio de Lisboa
Florianópolis - SC, 88050-000
Brazil
Tel.: +55-48-3239-2428
E-mail: sylvain@wplex.com.br - eduardo.hulse@wplex.com.br - eder.pinheiro@wplex.com.br

in Brazilian cities. A study by CNI RSB - Urban Mobility, 2015[1] indicates that only 24% of the Brazilian population use buses to commute. Nevertheless, bus riding is the most widespread public transportation option in the country. One of the reasons preventing more people from adopting this transport mode is the lack of clear and updated information. In this sense, powerful passenger information tools help to increase its attractiveness.

Using an online **Trip Planner**, the passenger is given a list of route options in the public transport network to go from his current place (or any other starting point) to a desired location. Each option defines values for the criteria the passenger wishes to minimize, such as the time of arrival, the number of transfers and the total walking distance. Instead of minimizing his time of arrival to his destination, the passenger can also provide a target arrival time and the tool maximizes his departure time.

WPLEX Software is a Brazilian software company that develops software for bus transit systems and companies, and provides a continuously available Trip Planning tool in several Brazilian cities such as Guarujá[2], São Bernardo do Campo[3] and Florianópolis[4].

Section 2 describes the features of our Trip Planner, while section 3 formulates it as a multi-objective problem. Section 4 shows why the Trip Planner by Departure Time and the Trip Planner by Arrival Time are equivalent problems and how they can be solved using the same algorithm. In section 5 we present an A*-based algorithm to solve the problem. Section 6 shows the influence of an A* parameter on both the algorithm performance and the solution quality and we finally draw some conclusions and perspectives for future work in section 7.

2 The Multi-Objective Bus Passenger Trip Planning Problem

In the problem tackled in this paper, the passenger expects to find some trip options for his journey from an origin to a destination, using the city bus network, all routes schedules and the current buses positions.

The data about the bus route network and the bus schedules is given by the bus company offering the service, and is considered here to be static data at the time of the passenger request.

The user needs to specify the following information:

- origin (current) location (it may be automatically picked out from the user's smartphone location),
- desired destination location,

[1] https://static-cms-si.s3.amazonaws.com/media/filer_public/7f/1d/7f1de722-455b-4a18-bc0a-6bdc5430b9a7/retratosdasociedadebrasileira_27_mobilidadeurbana.pdf

[2] https://guaruja.onibusfacil.com.br/tripplanner.jsp

[3] http://www.partiusbc.com.br

[4] https://www.floripanoponto.com.br/tripplanner.jsp

- departure (ready) time (and date, as the bus company defines specific bus schedules for different day types, such as weekday, Saturday, Sunday, etc.). By default, it is set to the present date and time.

The tool then suggests some bus trip options, each one containing the following information:

- bus routes to be used along the trip,
- for each transfer: get off time from the previous bus and boarding time onto the next bus,
- time of departure from origin location (it may be after the "ready" time given by the user),
- time of arrival at destination location,
- total walking time and distance.

For a request close to the present time, the current positions of buses (available through GPS) and traffic conditions update the static daily bus schedule considered to generate the trip plan options.See Jariyasunant et al (2010) for a similar inclusion of realtime conditions.

All passengers wish to arrive as soon as possible at their destination, but some may want to avoid transfers as much as possible or prefer short walks, even if this means a later arrival time. Therefore, the Trip Planning tool should provide a broad range of options to suit every user's needs.

In a simple Shortest Path Problem as described by Zhao et al (2008); Nannicini et al (2011); Idri et al (2017) or an Earliest Arrival Problem (see Yang et al (2012); Wang et al (2015)), the only aim is to minimize the arrival time to destination. Other criteria are also considered in this paper: the number of transfers and the total walking distance. Bast et al (2015) give a comprehensive review of similar problems and algorithms to solve them.

Several bus routes usually share common sections, which allows passengers to have backup options in case of an unpredictable event such as an accident slowing down the traffic and preventing them from performing their next transfer. Thus, it makes sense to present a variety of options for the passenger.

Figures 1, 2 and 3 depict some options of passenger trips for a request made at 21:45 between the neighbourhoods of *Córrego Grande* and *Lagoa da Conceição* in Florianópolis, Brazil. Table 1 gives a summary of the objective values for each option. Every option is the best with respect to one specific criterion. If a passenger wishes to arrive as fast as possible, he would certainly choose the figure 1 option. On the other hand, if he wants to avoid transfers and minimize the risk of delays, he will probably choose the option given in figure 3.

Table 1 Objective values for the Trip Planning request example

	Fig. 1	Fig. 2	Fig. 3
arrival time at destination	**22:28**	22:33	22:44
number of transfers	1	1	**0**
total walking distance	610m	**481m**	491m

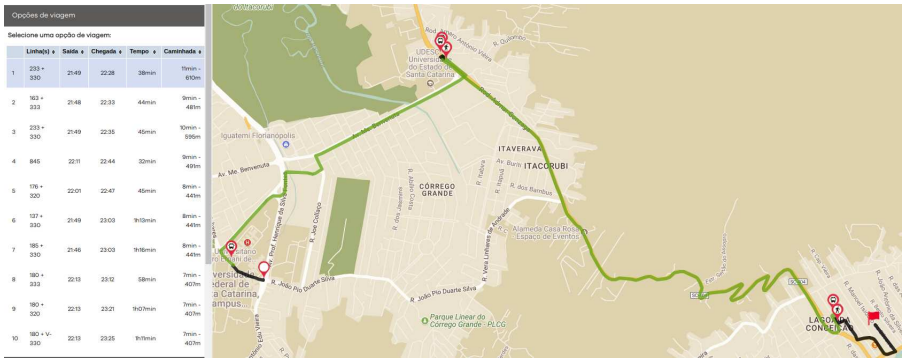

Fig. 1 Earliest arrival option, with one transfer

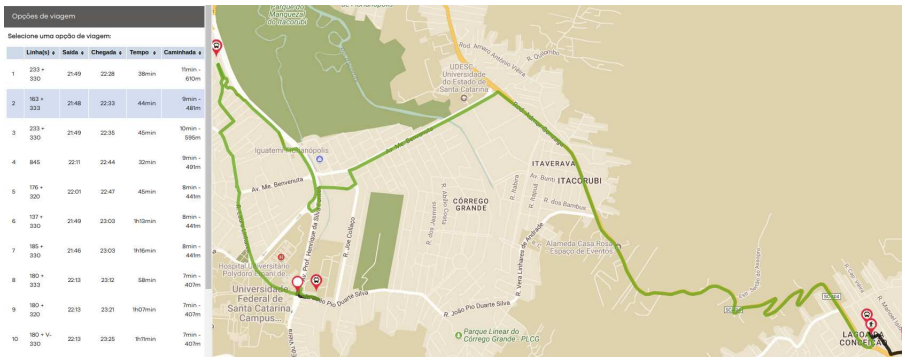

Fig. 2 Option with a short walking distance and one transfer

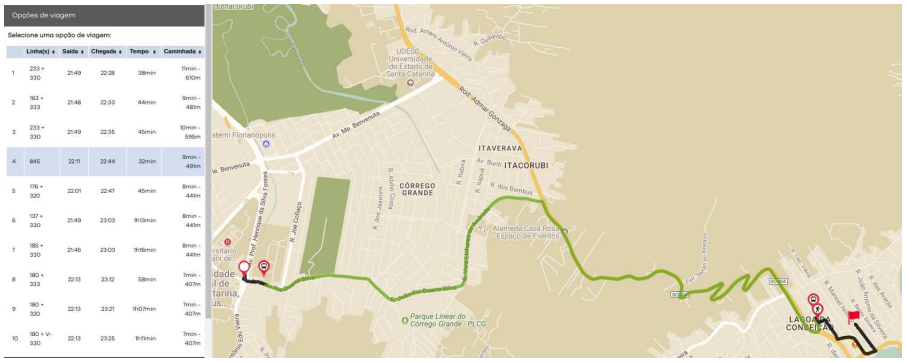

Fig. 3 Direct ride option, arriving after both previous options

3 Time-dependent formulation with timetable

In this section, the problem introduced previously is described formally. For a matter of clarity, we chose Greek letters to denote all the time-related variables or constants.

The problem formulated in sections 3.1, 3.2 and 3.3 is called **Trip Planning by Departure Time** (or TPDT). The later sections (3.4 and 3.5) describe possible additional features of this problem.

3.1 Graph and timetable

The graph defined to perform the search is usually either a time-expanded graph (as for Jariyasunant et al (2010); Wang et al (2015)) or a time-dependent graph. Time-expanded graphs can be too large and memory-consuming for the Trip Planner to be able to deal with several requests per second (Pyrga et al (2008)).

Among the time-dependent approaches, Cooke and Halsey (1966) and Dreyfus (1969) propose a generalized Dijkstra algorithm to solve the mono-objective time-dependent shortest path problem. Zhao et al (2008) present an A* algorithm where the heuristic function takes the current time as a parameter. Nannicini et al (2011) introduce a bidirectionnal A* search on large-scale networks. Idri et al (2017) use an A* algorithm on an heuristically restricted search space. Brodal and Jacob (2004) model the timetable through a time-dependent network so that their problem can be solved using Dijkstra-like methods. Mandow and De La Cruz (2010) present a multi-objective A*-based heuristic and Sanders and Mandow (2013) find all Pareto-optimal paths using a parallel label-setting algorithm.

The network in this paper is **time-dependent** where the transit-time function is given by the bus timetables stored as distinct data structures and not included directly in the graph.

The bus route network is defined in a directed graph $G = (V, E)$ where V is the set of vertices (all the possible reference locations and bus stops) and E is the set of directed edges, made of:

- a set E_R of **route edges**, and
- a set E_W of **walking edges**.

Sets E_R and E_W are disjoint and such that $E = E_R \cup E_W$. Every edge e in E_R is a street link between two successive bus stops in at least one route path. The set of all the bus routes that have edge e inside their path are denoted by $R(e)$, and R is the set of all bus routes. In addition, every walking edge e in E_W is such that $R(e) = \emptyset$ and is associated with a walking distance $d(e)$ and a walking duration $\delta(e)$.

A **path** is a sequence of nodes in V: $P = (v_1 v_2 \cdots v_i v_{i+1} \cdots)$ such that $\forall i \geq 1, (v_i v_{i+1}) \in E$. A **route path** is such that $\forall i \geq 1, (v_i v_{i+1}) \in E_R$, whereas a **passenger path** can be made of edges from both E_R and E_W. By

extension, path P can also be seen as a sequence of edges. Formally, for any edge $e \in P, \exists i, e = (v_i v_{i+1})$.

Each **bus route** r of the company is defined by its path of length m_r along the geographical network, which is the sequence of m_r nodes in V through which it goes : $P_r = (v_1^r v_2^r \cdots v_{m_r}^r)$, where for each i such that $1 \leq i \leq m_r - 1$, $(v_i^r v_{i+1}^r) \in E_R$ and $r \in R(v_i^r v_{i+1}^r)$. Each bus route $r \in R$ is associated with a sequence of n_r bus trips. Each **bus trip** is a bus ride along the bus route at a certain time of the day. It is defined by the list of the m_r times at which the bus arrives at each bus stop along the route path related with the trip. Formally, if r is a bus route, its u-th trip can be defined by the times $\theta^r_{u,v_1^r}, \theta^r_{u,v_2^r}, \cdots, \theta^r_{u,v_{m_r}^r}$ where $\forall i, 2 \leq i \leq m_r, \theta^r_{u,v_i^r}$ is the time the bus is at bus stop i. We suppose here that the **bus layover** is zero at every intermediate stop, meaning that the bus leaves just after it arrives.

In summary, all the bus trips of a given route r define a matrix θ^r of n_r lines and m_r columns of trip times from the first node of the route path to the last node. This matrix is the complete daily timetable for route r, and strictly increases linewise and columnwise: for each u such that $1 \leq u \leq n_r$ and for each i such that $1 \leq i \leq m_r$,

$$\text{if } i \leq m_r - 1 \text{ then } \theta^r_{u,v_{i+1}^r} > \theta^r_{u,v_i^r} \tag{1}$$

$$\text{if } u \leq n_r - 1 \text{ then } \theta^r_{u+1,v_i^r} > \theta^r_{u,v_i^r} \tag{2}$$

Inequality (1) states that the duration between two successive stops is positive, whereas inequality (2) ensures that the trips in the matrix are sorted in increasing order of their first stop time. This inequality also ensures that the trips don't overlap: if a bus departs before another one, it won't arrive after the other in any subsequent bus stop, which is a reasonable practical assumption. The contrary would mean that the buses overtake one another, which is operationally avoided.

Figure 4 and table 2 depict an example of graph and timetable on a simple network, with two bus routes performing three trips overall.

Table 2 Example of timetable for the previous graph, including 2 trips for route r_1 and one single trip for route r_2

Route	Trips		
r_1	v_1	v_2	v_3
	08:00	08:03	08:10
	08:50	08:54	09:00
r_2	v_2	v_1	v_3
	08:30	08:35	08:55

3.2 Solution description

The user defines a source node s and a target node t. He also defines a time Γ which is the date and time he is ready to start from node s.

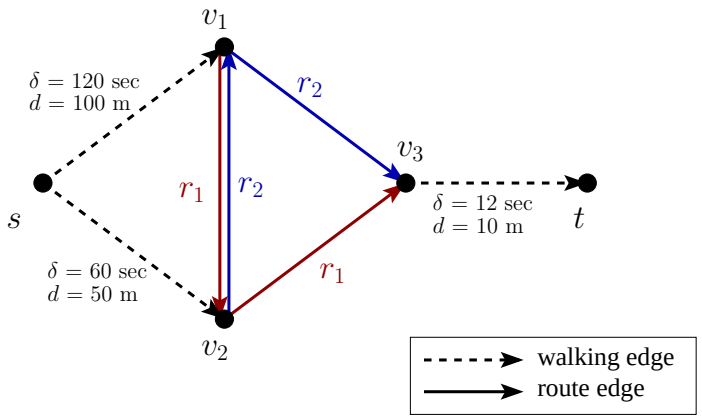

Fig. 4 Example of Trip Planner graph with 3 intermediate nodes and 2 bus routes

A **Trip Planning option** (or solution option) is composed of:

- a passenger path of q nodes $P = (v_1 v_2 \cdots v_q)$,
- the arrival time at each node of the path $\gamma_1, \gamma_2, \cdots, \gamma_q$,
- the index of the route taken at each step $r_1, r_2, \cdots, r_{q-1}$ (r_i is the route between v_i and v_{i+1}),
- the index of the trip taken at each step $u_1, u_2, \cdots, u_{q-1}$.

In the following, the expression "passenger path" sometimes refer to a solution option, meaning that this path also contains all the other attributes (arrival times, routes and trips).

These option attributes are subject to the following constraints:

- $v_1 = s$, $v_q = t$, $\gamma_1 = \Gamma$,
- $\forall i, 1 \leq i \leq q-1, \gamma_i < \gamma_{i+1}$,
- $\forall i, 1 \leq i \leq q-1$, if edge $(v_i v_{i+1}) \in P \cap E_W$ (walking edge), then we have $\gamma_{i+1} = \gamma_i + \delta(v_i v_{i+1})$ and by convention $r_i = u_i = 0$.
- $\forall i, 1 \leq i \leq q-1$, if edge $(v_i v_{i+1}) \in P \cap E_R$ (route edge) then:
  - $r_i \in R(v_i v_{i+1})$ (r_i is one of the routes going through edge $(v_i v_{i+1})$),
  - $1 \leq u_i \leq n_{r_i}$ (u_i is one of route r_i's trips),
  - $\gamma_{i+1} \geq \theta^{r_i}_{u_i, v_{i+1}}$ (arrival at node v_{i+1}),
  - $\gamma_i \leq \theta^{r_i}_{u_i, v_i}$ (departure from node v_i),

The arrival time at the next node in the case of a route edge could also be stated as exactly $\gamma_{i+1} = \theta^{r_i}_{u_i, v_{i+1}}$ because there is no point in chosing a higher value for γ_{i+1}. It is written here as an inequality for reasons of symmetry with the departure time constraint.

A passenger **boarding** can be seen as an index i along path P such that $r_i \neq 0$ and $r_i \neq r_{i-1}$: the path is going through a route edge on i and this route is not the same as the previous one (which can possibly be a walk). The **number of boardings** $b(P)$ of a passenger path $P = (v_1 v_2 \cdots v_q)$ is the number of times a route r_i is such that $r_i \neq 0$ and $r_i \neq r_{i-1}$ along the path.

Let $R^*(P)$ be the sequence of all bus routes along P: $R^*(P) = (r_i)_{0 \neq r_i \neq r_{i-1}}$. Then $b(P)$ is simply the size of $R^*(P)$: $b(P) = |R^*(P)|$.

3.3 Objectives

The **number of transfers** is exactly $b(P) - 1$ (or 0 if $b(P) = 0$, but pure walking paths are disconsidered) so minimizing the number of transfers is the same as minimizing $b(P)$.

Given a final passenger path $P = (v_1 v_2 \cdots v_q)$, the objective is to minimize several criteria:

- the arrival time $\gamma(P) = \gamma_q$,
- the number of boardings $b(P)$,
- the total walking distance $d(P) = \sum_{e \in P \cap E_W} d(e)$.

It is also possible to minimize the **total trip fare**, which depends on the bus types covering the bus routes along the passenger's path. For example, a passenger will pay an extra fare in a so-called executive bus with air conditioning. Moreover, in some cities, when a passenger buys a ticket, he may be able to ride several trips within a time window with a single fare. After this time window, he will need to pay another fare if he has to take another bus. The fare calculation will not be considered here so as to maintain the formulation concise.

3.4 Transfer delay constraint

When switching buses, the passenger could be prompted to arrive earlier than a minimum time (e.g. at least three minutes) before the next bus. This gives the passenger convenient get off and get on times and helps decreasing the risk of not catching the next bus if the current bus gets late. This minimum time is called **transfer delay duration** and is denoted by T.

It is possible to enforce that, whenever $(v_i v_{i+1}) \in P \cap E_R$ (route edge) and $r_i \neq r_{i-1}$ (passenger boards onto a new bus at v_i), the following constraint holds: $\gamma_i \leq \theta^{r_i}_{u_i, v_i} - T$.

3.5 Transfer location preference

In case of transfer between two routes sharing a common section (same sequence of stops), passengers usually choose to switch buses as early as possible to avoid the situation when the next bus of the transfer gets past the current one. In addition, **bus terminals** provide facilities (clear information, platforms for the disabled, etc.) to improve the comfort, speed and safety for the waiting. Therefore, passengers may prefer to wait for their bus at the terminal to perform their scheduled transfer.

This transfer location preference is modelled as follows: consider a passenger path $P = (v_1 v_2 \cdots v_q)$. Suppose that for two points g and h in the path such that $1 < g < h < q$, there are two routes x and y such that $\forall i, g \leq i < h, \{x, y\} \subset R(v_i v_{i+1})$. That is, routes x and y share the same subpath between nodes v_g and v_h. Figure 5 depicts a common section for routes x and y.

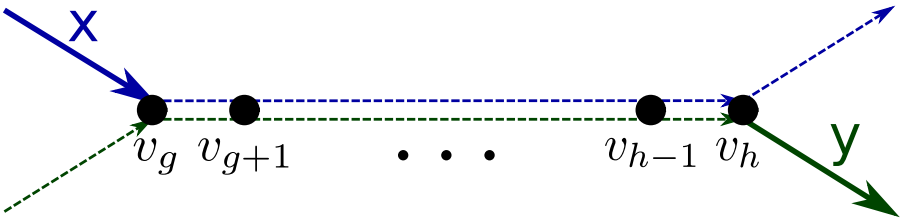

Fig. 5 Possible transfer locations incoming through route x and outgoing through route y

Suppose further that $r_{g-1} = x$ and $r_h = y$, meaning that the passenger has to switch between a bus of route x and a bus of route y at some node between v_g and v_h, and suppose that such a change is possible in any of those nodes. For this, we suppose that trips u_{g-1} on route x and u_h on route y are such that **at any node in the common subpath**, trip u_{g-1} arrives before trip u_h, considering the transfer security threshold T (see section 3.4 for its definition), meaning that the passenger has enough time to switch between both buses. Formally, $\forall i, g \leq i \leq h, \theta^x_{u_{g-1}, v_i} \leq \theta^y_{u_h, v_i} - T$.

Then the following holds:

- If the sequence of nodes $v_g \cdots v_h$ contains no bus terminal, the transfer between routes x and y must be performed at node v_g. In other words: $\forall i, g \leq i < h, r_i = y$ and $u_i = u_h$.
- If the sequence of nodes $v_g \cdots v_h$ contains at least one bus terminal (e.g. $v_{z_1}, v_{z_2}, \cdots$) , the transfer between routes x and y is performed at the **first bus terminal** in the sequence, denoted by v_{z_1}:
  - $\forall i, g \leq i \leq z_1 - 1, r_i = x$ and $u_i = u_{g-1}$.
  - $\forall i, z_1 \leq i < h, r_i = y$ and $u_i = u_h$.

4 Trip Planning by Arrival Time

In the **Trip Planning by Arrival Time** (TPAT), instead of being available at a given time and minimizing his arrival time at his destination (see the TPDT problem described above), the passenger may be interested in finding out when he'll have to leave from his current place to be able to arrive at some other place at a given time. For example, what time do I have to leave home so that I can get to work at 08:00 AM?

Formally, if the date and time the passenger wishes to arrive at node t is Γ, the path arrival time is constrained to $\gamma_q = \Gamma$ (with the same notations as previously), and the **departure time** γ_1 should be **maximized** (which is the only difference to the TPDT objectives).

Theorem 1 *The TPDT and the TPAT are computationally equivalent: from any instance of one problem, we can build an instance of the other problem such that any optimal solution in the problem can be matched with an optimal solution in the other problem.*

The idea of the proof is similar as the forward and backward search introduced by Wu and Hartley (2004). Here both the graph and the route timetables are reverted, and the trip plan is requested from the original target to the original source. We show here that TPDT $\leq_p$ TPAT (TPDT reduces polynomially to TPAT) by mapping this instance of the TPDT into an instance of the TPAT. We could show similarly that TPAT $\leq_p$ TPDT.

Figure 6 and table 3 depict the TPAT graph and timetable corresponding to the TPDT graph and timetable given in figure 4 and table 2.

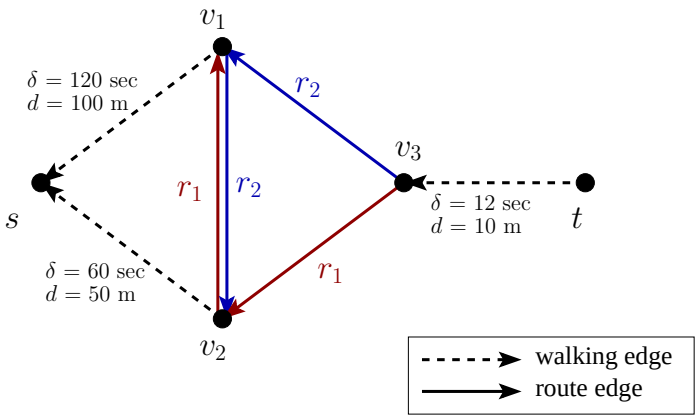

Fig. 6 Definition of the TPAT graph with reversed arcs

Table 3 Definition of the TPAT timetable, with reversed and negated times

Route	Trips		
r_1	v_3	v_2	v_1
	-08:10	-08:03	-08:00
	-09:00	-08:54	-08:50
r_2	v_3	v_1	v_2
	-08:55	-08:35	-08:30

Proof. Consider an instance of the TPDT described above and formulated in section 3.

The TPDT is described by its graph $G = (V, E)$ where $E = E_R \cup E_W$ and by its set of bus routes R, each of which ($r \in R$) is defined by its path P_r and its timetable θ^r. As stated above, a solution option between nodes s and t is defined by its path $P = (v_i)_{i=1}^{q}$, its arrival times at each node $(\gamma_i)_{i=1}^{q}$, its route at each step $(r_i)_{i=1}^{q-1}$ and its trip at each step $(u_i)_{i=1}^{q-1}$.

From this TPDT we build a TPAT instance as follows: let $G' = (V', E')$ be its graph where $E' = E'_R \cup E'_W$ is such that:

- $V' = V$ (same nodes as in the TPDT graph),
- $|E'| = |E|$ and $\forall e = (ij) \in E, e' = (ji) \in E'$ (TPDT arcs are reverted), where $R(e') = R'(e)$ (the set of routes R' is defined just below), $e \in E_W \Leftrightarrow e' \in E'_W$ and if $e \in E_W$ then $\delta(e') = \delta(e)$ and $d(e') = d(e)$ (same walking duration and distance for both arcs).

Furthermore, for each TPDT route $r \in R$ we associate a TPAT route $r' \in R'$ such that:

- its path $P_{r'} = (v^r_{m_r} v^r_{m_r-1} \cdots v^r_1) = (v^r_{m_r-i+1})_{i=1}^{m_r}$ where route r's path is $P_r = (v^r_1 v^r_2 \cdots v^r_{m_r})$ (TPDT route paths are reverted),
- each trip $u, 1 \leq u \leq n_r$ of route r is associated to a trip $u' = n_r - u + 1$ of route r' such that: $\forall i, 1 \leq i \leq m_r$, let $i' = m_r - i + 1$. Then $1 \leq i' \leq m_r$ and $\theta^{r'}_{u',v^r_{i'}} = -\theta^r_{u,v^r_i}$ (TPDT timetable is reverted and negated).

The timetable matrix then maintains its linewise and columnwise time-increase property:

- $\theta^{r'}_{u'+1,v^r_{i'}} - \theta^{r'}_{u',v^r_{i'}} = -\theta^r_{u-1,v^r_i} + \theta^r_{u,v^r_i} > 0$,
- $\theta^{r'}_{u',v^r_{i'+1}} - \theta^{r'}_{u',v^r_{i'}} = -\theta^r_{u,v^r_{i-1}} + \theta^r_{u,v^r_i} > 0$.

From a TPDT s-to-t solution option, we define a TPAT t-to-s solution option as follows:

- its path $P' = (v'_1 v'_2 \cdots v'_q) = (v_q v_{q-1} \cdots v_1)$ (reversed TPDT path),
- its arrival times at each node $\forall i, 1 \leq i \leq q, \gamma'_i = -\gamma_{q-i+1}$,
- its route and trip at each step $\forall i, 1 \leq i \leq q-1, r'_i = r_{q-i}$ and $u'_i = u_{q-i}$.

We show that the TPAT solution option complies with the rules described in section 3.2, except the specific TPAT requirement that $\gamma'_q = \gamma_1 = \Gamma$ is the maximum expected arrival time at the last TPAT node s:

- $v'_q = v_1 = s$ (node s is the destination of the TPAT we defined)
- $v'_1 = v_q = t$
- $\forall i, 1 \leq i \leq q-1, \gamma'_{i+1} - \gamma'_i = -\gamma_{q-i} + \gamma_{q-i+1} > 0$
- $\forall i, 1 \leq i \leq q-1$, if $(v'_i v'_{i+1})$ is a walking edge then $(v_{q-i} v_{q-i+1})$ is a walking edge in the original TPDT graph, therefore:

$$\begin{aligned}\gamma'_{i+1} &= -\gamma_{q-i} = -(\gamma_{q-i+1} - \delta(\gamma_{q-i}\gamma_{q-i+1})) = -\gamma_{q-i+1} + \delta(v'_i v'_{i+1}) \\ &= \gamma'_i + \delta(v'_i v'_{i+1})\end{aligned}$$

Besides, $r'_i = r_{q-i} = 0$ and $u'_i = u_{q-i} = 0$.

Suppose now that $(v'_i v'_{i+1})$ is a route edge. So is $(v_{q-i} v_{q-i+1})$ in the TPDT graph. Then:

- $r_{q-i} \in R(v_{q-i} v_{q-i+1})$, which means $r'_i \in R(v'_i v'_{i+1})$. Also, as u_{q-i} is a trip from route r_{q-i}, u'_i is a trip from route r'_i (in other words, $1 \leq u'_i \leq n_{r'_i}$).
- $\gamma_{q-i} \leq \theta^{r_{q-i}}_{u_{q-i}, v_{q-i}}$ (departure from v_{q-i}). Given that, by definition: $\gamma'_{i+1} = -\gamma_{q-i}$ and $\theta^{r_{q-i}}_{u_{q-i}, v_{q-i}} = -\theta^{r'_i}_{u'_i, v'_{i+1}}$, we finally have $\gamma'_{i+1} \geq \theta^{r'_i}_{u'_i, v'_{i+1}}$, which is the arrival time constraint at v'_{i+1}.
- $\gamma_{q-i+1} \geq \theta^{r_{q-i}}_{u_{q-i}, v_{q-i+1}}$ (arrival at v_{q-i+1}). Similarly: $\gamma'_i = -\gamma_{q-i+1}$ and $\theta^{r_{q-i}}_{u_{q-i}, v_{q-i+1}} = -\theta^{r'_i}_{u'_i, v'_i}$, which yields $\gamma'_i \leq \theta^{r'_i}_{u'_i, v'_i}$ (departure time constraint from γ'_i).

We still need to show that a solution option of the TPAT instance is optimal with respect to a given criterion if and only if the corresponding TPDT solution option is optimal with respect to the same criterion. In fact, we show that both the number of boardings and the walking distance are the same values in both problems. Regarding the departure time (in the TPAT) or arrival time (in the TPDT), they are opposite one another, so maximizing one will minimize the other one. Recall that every criterion is described in section 3.3.

First consider the number of boardings in the TPDT $b(P)$ and the number of boardings in the TPAT $b(P')$. Recall that, by definition, $b(P) = |R^*(P)|$ where $R^*(P) = (r_i)_{0 \neq r_i \neq r_{i-1}}$ is the sequence of all the routes along P. Then similarly $b(P') = |R^*(P')|$. As by definition $R^*(P')$ has the same elements as $R^*(P)$ in reverse order, its size is the same as $R^*(P)$. Then $b(P) = b(P')$.

The walking distance is also simple to calculate: in the TPAT instance its definition is $\sum_{e \in P' \cap E'_W} d(e)$. As the arcs in $P' \cap E'_W$ are the same as those in $P \cap E_W$ but they are just reversed and with the same walking distance, $\sum_{e \in P' \cap E'_W} d(e) = \sum_{e \in P \cap E_W} d(e)$.

To complete the proof, we finally show that the departure time in the TPAT and the arrival time in the TPDT have opposite values. Recall that $\forall i, 1 \leq i \leq q, \gamma'_i = -\gamma_{q-i+1}$. Especially for $i = 1$, $\gamma'_1 = -\gamma_q$, which means that maximizing γ'_1 is the same as minimizing γ_q.

□

This theorem and proof also introduce a way to solve the TPAT using an already implemented TPDT algorithm. By reverting the input data (graph and timetable) and applying the TPDT algorithm, and finally reverting the solution option back to the original values, we are able to solve the TPAT as efficiently as the TPDT, using the algorithm described in the next section.

5 A*-guided algorithm with Pareto domination-based elimination

The solution process for the problem described in the previous section is guided by an A* procedure over the graph defined by the bus route network. For that, we use a **trip time matrix** similar as the one introduced by Delling et al

(2012). As in Dijkstra's shortest path algorithm, labels are selected among a set of open labels and expanded over the graph during the solving process. Each label L is related to a given node v_l of the network and its path $P_l = (v_1 v_2 \cdots v_l)$ of length l (with $v_1 = s$) and contains the same information as for a solution option path P, namely: the time of arrival at the current node γ_l, the number of transfers $b(L)$ and the total walking distance $d(L)$.

During its processing, the algorithm maintains a priority queue of open labels and in the main loop, an open label is selected and expanded, creating new open labels defined from the label node's outgoing edges that are in turn inserted in the priority queue.

5.1 Label expansion

A **label expansion** is the generation of new labels from the current label and all the neighbour edges, using all the bus routes available on these edges as well as the walking edges.

The selection for the next label to be expanded is based on the **minimum expected arrival time** at the target node t and considers the objective of minimizing the arrival time γ_q as a driver for the search. One of the consequences is that the first solution options found are one of the earliest arriving ones overall, regardless of the values of the other objectives.

The **expected arrival time** at the target node t from label L is defined as $\lambda(L) = \gamma(L) + \tau(v_l, t)$, where:

- $\gamma(L) = \gamma_l$ is the arrival time at the label node v_l (see its definition in section 3.2),
- $\tau(v_l, t)$ is an estimate of the duration of a ride between v_l and t.

The time on the path's first node is $\gamma_1 = \Gamma$ and if $l \geq 2$, γ_l depends on the incoming edge kind:

- if $(v_{l-1} v_l) \in E_W$ then γ_l is computed using the value γ_{l-1} from the father label: $\gamma_l = \gamma_{l-1} + \delta(v_{l-1} v_l)$,
- if $(v_{l-1} v_l) \in E_R$ then γ_l is the time read in the timetable matrix corresponding to a route r_{l-1} and a trip u_{l-1} on node v_l: $\gamma_l = \theta^{r_{l-1}}_{u_{l-1}, v_l}$ where $r_{l-1} \in R(v_{l-1} v_l)$ and $1 \leq u_{l-1} \leq n_{r_{l-1}}$.

The duration of a ride between any node v and the target node t is estimated as $\tau(v, t) = \frac{d(v,t)}{\sigma}$, where:

- $d(v, t)$ is the geographical distance between nodes v and t,
- σ is a speed parameter called A* speed.

The value of parameter σ should be chosen carefully: unless this value is sufficiently high, the expected arrival time at the target $\lambda(L)$ is not garanteed to be a lower bound of the actual arrival time at t of any s-t path generated from label L. For example, if we set σ as the maximum possible speed for a city bus (e.g. 80 km/h), $\lambda(L)$ will always be a lower bound for the actual arrival

time at t for any label generated from L which proves the admissibility of our A*-based heuristic. However, chosing a too high value can provide a poor lower bound in most cases as a bus will not always be available at once at the node and it will rarely go straight to the target node at maximum speed. Tradeoff between the algorithm's processing time and the solution quality through choices of values for σ will be discussed in section 6.

When creating a new label from the current label and all the current node's neighbour edges, we need to determine which will be the first possible trip of each outgoing route at the time given by the label. Recall that for each route $r \in R$, each column of matrix θ^r (sequence of trip departure times in the same node) is sorted in increasing order, as stated in inequality (2). In case of a transfer, the next possible trip of each route going through a given node can be found solving a binary search over the trips in the route timetable. Otherwise, the trip of the next label can be the same as the current label's trip.

5.2 Stopping conditions

Unlike usual A* algorithms, our algorithm doesn't stop at the first solution option it finds. Instead, it carries on the search until it finds some acceptable solution options that, for each criterion, have a better value or similar as the first option found. For example, independently of the values on the other criteria, it will eventually reject an option with five transfers when the first solution option found has a single one.

The algorithm main loop stops whenever:

- no more labels are available to be expanded (empty priority queue),
- the number of solution options (s-t paths) found so far have reached a predefined quantity (e.g. 10 options),
- the number of expanded labels have reached a predefined quantity (e.g. 100,000 expanded labels).

The last condition is similar to a time limit condition but the fact of using the number of expanded labels instead of an actual time limit (e.g. one second) allows the algorithm to remain deterministic, i.e. to always return the same answer given the same input. The value for the maximum number of expanded labels is tuned beforehand according to each city bus network, so that the processing time for most requests is less than one second.

5.3 Heuristic label pruning

Beside the stopping conditions, some labels are pruned during the algorithm processing if one of the following values of the label is much higher compared to the value of the first option found:

- its estimated time to target $\lambda(L)$ (e.g. over 1h30),

- its number of boardings $b(L)$ (e.g. over 2 more boardings),
- its total walking distance $d(L)$ (e.g. over 1 km).

The first solution option found is used as a reference for this heuristic label pruning as it is one of the most likely to be chosen by the passenger, being one of the earliest arriving solution options. Once the label selection in the main loop is guided by the minimum expected arrival time to target, meaning that the expected arrival time to target is expected to rise along the successive label selections, the first condition can also be triggered as a stopping condition of the main loop.

In order to avoid pruning labels with just slightly worse criterion values and which can turn out to be very good options later, we introduce an **equality threshold** for two criteria: the arrival time (90 seconds) and the walking distance (30 seconds). For instance, two labels are considered to have a similar arrival time if the difference between their respective arrival times is lower than 90 seconds. In practice, when deciding between two possible solution options arriving with a difference of less than 90 seconds, the passenger will doubtlessly consider the other criteria (number of transfers, walking distance) to make up his mind.

5.4 Pareto domination

Each time a new label is generated through its predecessor's expansion, it is submitted to a **Pareto domination**-based elimination on its node. This label is compared, for every criterion, to all the previously generated non-dominated labels on the same node and coming from the same bus route. When comparing each objective value, the same thresholds as the ones described previously are considered. Whenever a label is not worse than another one on every criterion but is better for at least one criterion, it is said to dominate the other label. Every dominated label is discarded from the graph.

Figure 7 plots feasible options or labels (squares and circles).

The green squares are the Pareto-optimal options of this set of points, and the solid line depicts the Pareto frontier. Every option in the upper right corner (red circles) is dominated by at least one Pareto-optimal option. When considering the threshold to compare the options, the Pareto frontier becomes the strip between the original Pareto frontier and the dashed line to its right. In this case, the option depicted by the red square becomes Pareto-optimal.

If two non-dominated labels have the same sequence of bus routes (recall the definition of $R^*(P)$ in section 3.3), their last transfer are compared according to the transfer location preference introduced in section 3.5. If one of the transfers is preferable (at a terminal while the other one is not, or at a previous node), the label corresponding to the other one is discarded.

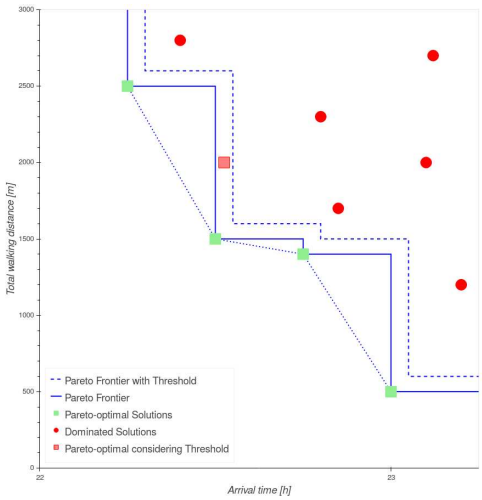

Fig. 7 Example of Pareto frontier considering two criteria: arrival time and total walking distance

6 Results and choice of value of the A* speed σ

Tests were performed in two Brazilian cities where our Trip Planner is available for the population: Florianópolis and Guarujá. Table 4 gives an insight on the size of the problems related to these cities.

Table 4 Problem size for the tested cities

City	Bus stops	Links	Routes	Daily trips
Florianópolis	2512	23575	306	7854
Guarujá	697	6855	64	2364

For both cities, we ran the algorithm for a set of previously chosen origin-destination locations over the network and for requests at a specific day and several times along the day, so as to cover the whole timetable. The origin-destination pairs in the test were chosen according to their high processing time over a big set of randomly chosen origin-destination pairs, for a request at a specific time.

Figure 8 depicts the tests carried on in Florianópolis with the speed parameter σ of the A* heuristic (see section 5) varying between 0 and $+\infty$. Each point in the chart is an average value over a set of 180 requests as described above (several origin-destination pairs and several times along the day). The average number of solution options gives an insight of the quality of the algorithm. Obviously, similar solution options, such as two options with the exact same routes and trips but distinct get on/off stops, are considered only once.

The value $\sigma = 0$ means that the next label to be expanded is chosen only according to its distance to the target node (and disconsidering its current

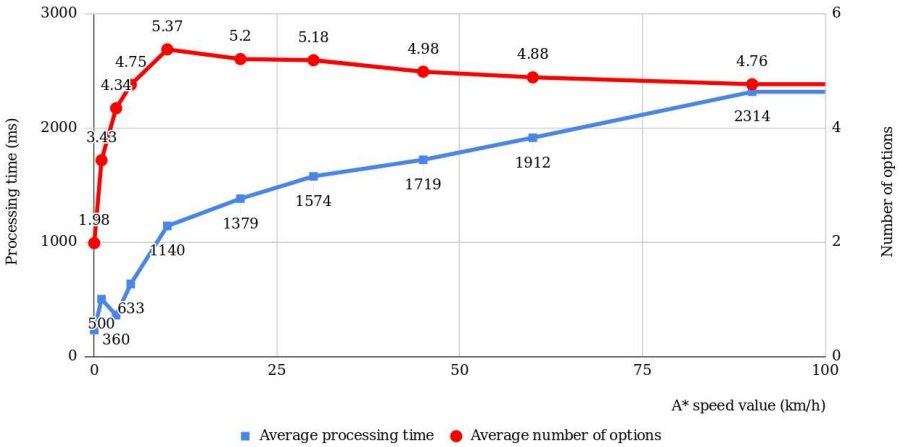

Fig. 8 Influence of the speed parameter of the A* procedure on the processing time and number of solution options in Florianópolis, Brazil

time), whereas setting $\sigma = +\infty$ means chosing according to the label's current time and disconsidering its location. Any value in between balances both attributes.

Figure 8 shows that in Florianópolis, the average processing time increases with the A* speed, but the number of solution options decreases after a maximum value obtained around $\sigma = 10$. Interestingly, this value of the A* speed doesn't make the A* heuristic function admissible because it will not always give a lower bound on the actual arrival time at the target location. This is especially the case when, considering a label location and time, there is a bus just about to arrive at this location that can take the passenger directly to the target location (as it will probably drive faster than 10 km/h). However, the tests empirically show that this value is one of the best for the variety of options and yields an acceptable processing time (around 1140 ms).

Figures 9 and 10 show maps of label expansion for the cases $\sigma = 3$ and $\sigma = 10$, respectively, for the same request as the example in section 2. The origin and destination of the request are the green and red squares, and each expanded label during the algorithm is depicted as a circle with a color gradient between yellow and purple, depending on the step in which the label was expanded (purple for a later step). The higher the value of σ, the more labels are expanded along the algorithm even if some label expansions look useless. This explains why the average processing time tends to rise with high values of the A* speed σ.

Figure 11 plots a similar chart as above for the tests run in Guarujá over a set of 650 requests for each value of the A* speed σ. In this case, both the number of solution options and the processing time remain somewhat constant for $\sigma > 20$. For $\sigma < 20$, the number of options rises continuously and the processing time increases to its maximal value (still less than 800 ms) at around $\sigma = 2$ and decreases afterwards. As for Florianópolis, $\sigma = 10$ is a

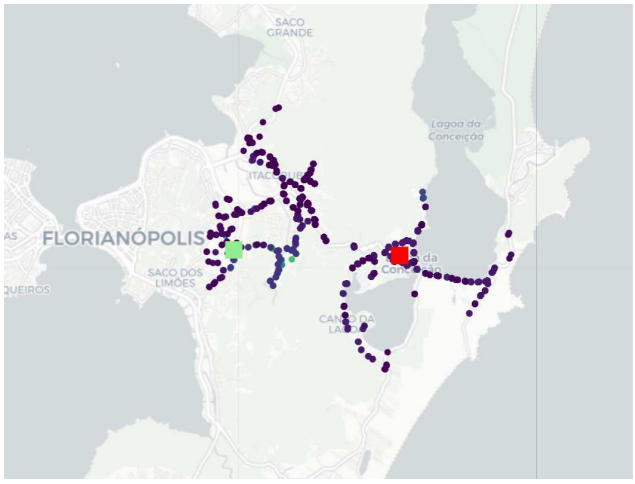

Fig. 9 Label expansion map for a request in Florianópolis and $\sigma = 3$

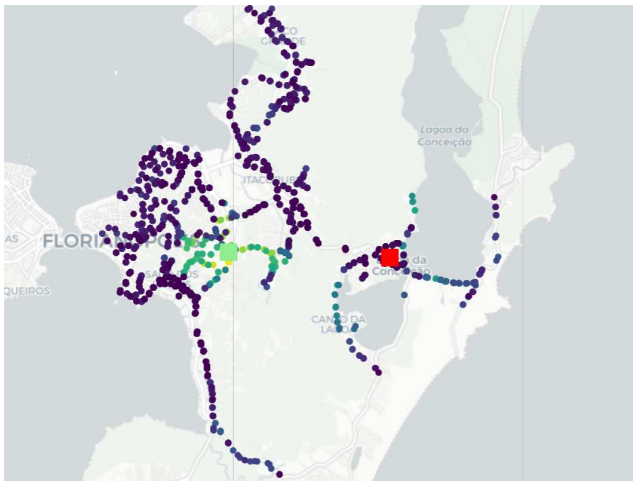

Fig. 10 Label expansion map for a request in Florianópolis and $\sigma = 10$

good setting but for Guarujá it seems that $\sigma = 20$ or higher is slightly better. These values of the A* speed parameter σ give a good balance between the processing time and the solution quality in both cases.

From the Guarujá test details, we noted that distinct values of $\sigma > 20$ present a high probability to return exactly the same options, unlike the Florianópolis case. The processing time is also always lower in Guarujá than in Florianópolis (which can be explained by the problem sizes given in table 4), but the average number of options is higher in Guarujá. This is certainly because of the structure of the bus route network: in Guarujá there are lots of routes sharing subroutes one with another, whereas in Florianópolis some remote areas are served by a single route.

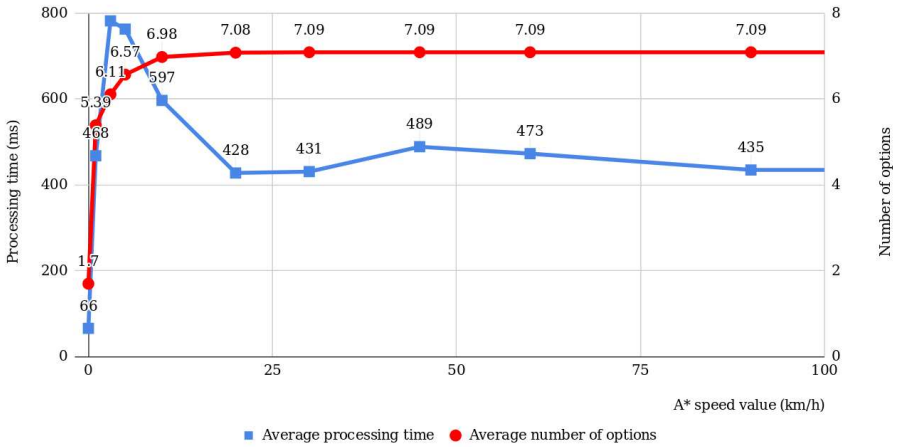

Fig. 11 Influence of the speed parameter of the A* procedure on the processing time and number of solution options in Guarujá, Brazil

7 Conclusion and perspectives

In this paper, we described a multi-objective Bus Passenger Trip Planning Problem and explained how it can be solved, either for its arrival time or departure time version, through an A*-based algorithm where Pareto-dominated labels are discarded. The tests show that using the right value for the A* speed (10 for Florianópolis, 20 for Guarujá) optimizes the algorithm performance as well as the quantity and quality of the options given to the user.

We focused on the impact of changing the A* speed value over the solution quality. Other parameters, such as the equality threshold values, also have a significant influence on the result and could be tested in the same way. We are also aware that the algorithm would probably be even more powerful with some improvements, such as preprocessed transfer options, with no need of constant lookup in the timetable. Furthermore, in the A* procedure, the estimated time from a given node to the target should be calculated using the bus route network and the timetable so as to get a better lower bound and possibly to avoid the need of an A* speed parameter.

Still, our Trip Planning tool has been continuously available in several Brazilian cities for several years and provides a wide range of solution paths within a couple of seconds or less. Whoever uses the Trip Planning tool, even longtime users of the bus transit network, can come up with some surprising route options. It is a good incentive for public transport in traffic-crowded cities like Florianópolis: being the home city of the three authors, we can only confirm the benefit of having a reliable Trip Planning tool for the better use of bus transportation.

References

Bast H, Delling D, Goldberg A, Müller-Hannemann M, Pajor T, Sanders P, Wagner D, Werneck RF (2015) Route Planning in Transportation Networks. Microsoft Research Technical Report pp 1–65

Brodal GS, Jacob R (2004) Time-dependent networks as models to achieve fast exact time-table queries. In: Electronic Notes in Theoretical Computer Science, vol 92, pp 3–15, DOI 10.1016/j.entcs.2003.12.019

Cooke KL, Halsey E (1966) The shortest route through a network with time-dependent internodal transit times. Journal of Mathematical Analysis and Applications 14(3):493–498

Delling D, Pajor T, Werneck RF (2012) Round-Based Public Transit Routing. Proceedings of the 14th Meeting on Algorithm Engineering and Experiments (ALENEX'12) pp 130–140

Dreyfus SE (1969) An Appraisal of Some Shortest Path Algorithms. Operations Research 17(3):395–412

Idri A, Oukarfi M, Boulmakoul A, Zeitouni K, Masri A (2017) A new time-dependent shortest path algorithm for multimodal transportation network. Procedia Computer Science 109:692–697

Jariyasunant J, Work DB, Kerkez B, Sengupta R, Bayen AM, Glaser S (2010) Mobile Transit Trip Planning with Real-Time Data. Transportation Research Board 89th Annual Meeting (September):1–17

Mandow L, De La Cruz JLP (2010) Multiobjective A* search with consistent heuristics. Journal of the ACM 57(5):1–25

Nannicini G, Delling D, Schultes D, Liberti L (2011) Bidirectional A* search on time-dependent road networks. Networks 59(2):240–251

Pyrga E, Schulz F, Wagner D, Zaroliagis C (2008) Efficient models for timetable information in public transportation systems. Journal of Experimental Algorithmics 12(2):1

Sanders P, Mandow L (2013) Parallel Label-Setting Multi-Objective Shortest Path Search. Proceedings - IEEE 27th International Parallel and Distributed Processing Symposium, IPDPS 2013 pp 215–224

Wang S, Lin W, Yang Y, Xiao X, Zhou S (2015) Efficient Route Planning on Public Transportation Networks. Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data - SIGMOD '15 pp 967–982

Wu Q, Hartley J (2004) Accommodating user preferences in the optimization of public transport travel. International Journal of Simulation: Systems, Science and Technology 5(3-4):12–25

Yang Y, Wang S, Hu X, Li J, Xu B (2012) A Modified K-Shortest Paths Algorithm for Solving the Earliest Arrival Problem on the Time-Dependent Model of Transportation Systems. Proceedings of The International Multi-Conference of Engineers and Computer Scientists II:1562–1567

Zhao L, Ohshima T, Nagamochi H (2008) A* algorithm for the time-dependent shortest path problem. The 11th Japan-Korea Joint Workshop on Algorithms and Computation (WAAC08) pp 36–43