Transfer Customization with the Trip-Based Public Transit Routing Algorithm

Vassilissa Lehoux-Lebacque[1]
NAVER LABS Europe, Meylan, France

Christelle Loiodice
NAVER LABS Europe, Meylan, France

Abstract

In the context of routing in public transit networks, we consider the issue of the customization of walking transfer times, which is incompatible with the preprocessing required by many state-of-the-art algorithms. We propose to extend one of those, the Trip-Based Public Transit Routing algorithm, to take into account *at query time* user defined transfer speed and maximum transfer duration. The obtained algorithm is optimal for the bicriteria problem of optimizing minimum arrival time and number of transfers. It is tested on two large data sets and the query times are compatible with real-time queries in a production context.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases Public transit, Route planning, Algorithms, Customization

Digital Object Identifier 10.4230/OASIcs.ATMOS.2021.15

1 Introduction

In mobility applications or websites, finding paths between an origin and a destination is a classical problem. In public transit networks, those paths can combine public transit modes and walking between the stations. In this paper, we are interested in building sets of alternative paths according to user specified transfer speed and/or maximum transfer duration. Customization of transfer times is an important feature for routing applications, as in many contexts, users have an a priori idea of the maximum duration they wish to spend on a transfer, or the speed at which they will perform it. The speed or maximum duration can be related to weather (not walking too much under the rain or walking more slowly in hot weather), to trip aim (travelling with a heavy luggage, taking small kids to an activity) or simply to the physical condition of the user. In some other contexts, a user might wish to set a large maximum duration and a high speed, for instance if long transfers at a brisk pace are perceived as an opportunity to keep fit. In addition, some modes that can be carried by the user in the public transports, like kick scooter or roller blades, have a network very similar to the walking network and can be modelled by faster walking transfers in the routing algorithm. In the following, we will refer to walking transfers for simplicity, but transfers could be done using those modes if speed customization is added.

Many efficient algorithms have been developed over the last years for mono- or bicriteria routing in public transit network (e.g. [1, 4, 5, 15, 3]). However, they mostly consider fixed transfer speed and sometimes rely on long preprocessing depending on fixed transfer times. When transitive closure of transfers is not required by the algorithm, there usually exists a limit at the application level of the duration of a transfer. This reduces the size of the routing problem and is acceptable for many users who would like to avoid long transfers. In [13], which extends the RAPTOR [4] algorithm, the authors consider higher maximum

[1] Corresponding author

21st Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2021).
Editors: Matthias Müller-Hannemann and Federico Perea; Article No. 15; pp. 15:1–15:17

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

transfer duration at the application level, and user customizable transfer speed and maximum transfer duration. In [14], the authors consider unrestricted walking transfers (no maximum duration). They find that the earliest arrival time is improved for long distance queries by allowing more walking in 75 percent of the times, compared to a limit of 8 min walking for Germany and a limit of 15 min for Switzerland. This issue of unlimited transfer times has then been studied in several recent publications [11, 2]. However, as their transfer graphs can count transfers of several hours, it is not clear that users will be willing to perform the optimal itineraries when they involve too much walking. Especially if alternatives exist with less walking, and even if those alternatives are significantly slower. We hence consider that the possibility to customize the transfer speed and maximum duration at query time will provide users with itineraries better adapted to their context and preferences.

In this article, we want to extend the Trip-Based Public Transit Routing algorithm [15] to user customized transfer speed and maximum duration at query time, while preserving the optimality of the algorithm. The outline is as follows. Section 2 describes the principle of the Trip-Based Public Transit Routing algorithm and the notations used in the paper. The proposed extension is explained in Section 3 and a proof of optimality is given. Tests on two large size data sets are discussed in Section 4. Section 5 concludes the article.

2 Notations and principle of the Trip-Based Public Transit Routing

In this section, we describe public transit networks using notations similar to that of [15], for easier reference. Public transit information contains the schedules of the transit vehicles. For each vehicle, it defines the passage times of the vehicle at the stations (also called *stops*) where its passengers can board and alight. A *trip* t corresponds to a vehicle following its *sequence of stops* $\overrightarrow{p}(t) = \langle t@0, t@1, \dots \rangle$. We denote by $\tau_{arr}(t, i)$ (resp. $\tau_{dep}(t, i)$) the *arrival time* (resp. *departure time*) of t at the i^{th} stop of $\overrightarrow{p}(t)$. We group trips of identical stop sequences that do not overtake each other into *lines*. The lines hence do not exactly represent the routes of the public transport network. Similarly to trip notations, we denote by $\overrightarrow{p}(L) = \langle L@0, L@1, \dots \rangle$ the stop sequence of line L. Note first that the partition of the trips into lines is not unique. Second, as the trips of a line do not overtake each other, they form a totally ordered set with the relation $\preceq$ and a partial order with $\prec$ defined for two trips t and u having the same sequence by:

$$t \preceq u \iff \forall i \in \{0, 1, \dots, |\overrightarrow{p}(t)| - 1\}, \quad \tau_{arr}(t, i) \leq \tau_{arr}(u, i)$$
$$t \prec u \iff t \preceq u \text{ and } \exists i \in \{0, 1, \dots, |\overrightarrow{p}(t)| - 1\}, \quad \tau_{arr}(t, i) < \tau_{arr}(u, i)$$

$t@i \to t@j$ denotes a displacement between the i^{th} and the j^{th} stops of trip t using trip t and similarly, a transfer between trip t at the i^{th} station and trip u at the j^{th} station is denoted $t@i \to u@j$. For a given stop s, $\mathbf{L}(s)$ is the set of all line-index pairs (L, i) such that $s = L@i$. Information about the transfers between the stops of network is usually represented directly by *walking transfer times* $\Delta\tau_{\text{fp}}(p, q)$ defined for any pair of stops (p, q), $p \neq q$ that are close enough from one another. When transferring between two trips at a given station $(t@i = u@j = p)$, a *minimum change time* $\Delta\tau_{\text{fp}}(p, p)$ can also be defined to represent the time needed to move within the station.

The **Trip-Based Public Transit Routing (TB) algorithm [15]** is an exact state-of-the-art algorithm for routing in public transit networks. A *bicriteria earliest arrival time query* (EAT) takes as inputs an origin, a destination and a start time. The two criteria minimized are *arrival time* and *number of transfers*.

If a set of criteria $(c_1, c_2, \ldots, c_n)$ is to be minimized, a solution s with value $(v_1, v_2, \ldots, v_n)$ is *non-dominated in the Pareto sense* if there is no other solution s' with values $(v'_1, v'_2, \ldots, v'_n)$ such that for all $i \in \{1, 2, \ldots, n\}$, $v'_i \leq v_i$ and $\exists i \in \{1, 2, \ldots, n\}$ such that $v'_i < v_i$. Non-dominated solutions are called *optimal* and the maximum cardinality set of non-dominated solutions is denoted *Pareto set*. The *Pareto front* is the image of the Pareto set in the criterion space. As most routing algorithms, the TB algorithm doesn't compute the complete Pareto set but only one solution with this value per element in the Pareto front. As in [12], we call this family of sets *complete sets*. The TB algorithm builds a complete set of solutions for minimum arrival time and number of transfers in polynomial time. It uses a specific graph representation based on trips as vertices and feasible transfers as arcs. For each trip, a neighbourhood of reachable trips is built by a preprocessing step and is pruned while ensuring that a complete set of solutions can still be obtained. We call a preprocessing that ensure the optimality of the algorithm a *correct* preprocessing. Such a preprocessing for the TB algorithm ensures that for any optimal value, there exists an optimal solution with this value whose transfers are all in the pruned transfer set. In the search graph, an EAT query consists in a breadth-first search like exploration. Trip segments reached from the origin given the departure time form the initial current queue Q, while for every stop p from which destination can be reached and any trip t such that $p = t@j$, trips segments $t@i \to t@k$ with $i < j \leq k$ are the targets of the algorithm. Those targets can be represented by the set $\mathcal{L}$ of triplets $(L, i, \Delta\tau)$ where s is a stop from which destination can be reached, $\Delta\tau$ is the duration of walking from s to destination and $(L, i) \in \mathbf{L}(s)$. During an iteration, all the trip segments of the current queue are processed. If a trip segment is a target, best arrival time can be improved. Transfers are performed to add the reached trip segments to the queue for the next iteration.

The TB algorithm can also be used with slight modifications to compute *profile queries*, where all the optimal paths must be found for a given starting time range.

Pruning phase. Given an origin trip t and a destination trip u, transfer $t@i \to u@j$ is *feasible* if and only if

$$\tau_{arr}(t,\, i) + \Delta\tau_{\text{fp}}(t@i,\, u@j) \leq \tau_{dep}(u,\, j)$$

When considering the set of feasible transfers between a trip t at its i^{th} stop and a line L at its j^{th} stop, the order on the trips of L implies that this set is either empty or has a minimum element according to $\preceq$ and $\prec$. This element is the earliest trip such that the transfer is feasible. To construct a complete solution set for minimum arrival time and number of transfers, it is sufficient to add only this earliest transfer to the search graph.

The preprocessing phase of the TB algorithm as described in [15] first computes the set of all earliest feasible transfers and then prune the neighbourhood of each trip based on stop labels those values are the earliest arrival times at stops when transferring from the trip.

In [7], the authors modify the preprocessing to make it faster than in the original version. The key idea is to perform an additional pruning step based on trip-to-line transfers before the arrival time based pruning. The transfers between a trip t and a line L are compared using the following dominance relation. If $u, u' \in L$, a transfer $t@i \to u@j$ *is dominated* by a transfer $t@i' \to u'@j'$ if and only if

$$\begin{array}{llllll} i \leq i' & \text{and} & u' \leq u & \text{and} & j' \leq j & \text{and} \\ (i < i' & \text{or} & u' < u & \text{or} & j' < j) & \end{array}$$

We will extend both preprocessing steps to take into account transfer time customization.

3 Customization of transfers

Now we want to enable customization of transfer speed and maximum transfer duration at query time. First note that the values chosen by the user need to be bounded between realistic values defined at the application level. To modify the transfer speed, we consider that the public transit information contains transfer times for some constant chosen speed s_{std}. We can define for each query a *duration coefficient* σ corresponding to the user chosen speed s: $\sigma = s_{std}/s$. If the standard duration of a transfer is $\Delta\tau$, the application of a duration coefficient σ will result in a duration $\sigma\Delta\tau$. To avoid unrealistic fast transfer time values, a minimum application level duration coefficient can be chosen with $1 \geq \varsigma_{\min} > 0$. Similarly, a maximum transfer duration coefficient at the application level $\varsigma_{\max} \geq 1$ can be set.

3.1 Modifications of the query phase

Suppose that we obtain after preprocessing a transfer set correct for any user defined transfer duration coefficient $\sigma \in [\varsigma_{\min}, \varsigma_{\max}]$ and maximum transfer duration $\Delta\tau_{\max} \geq 0$. To avoid performing any transfer longer than $\Delta\tau_{\max}$, we can prune at query time the transfers whose duration exceeds the bound. For faster computations, and unlike in the standard algorithm, the duration must be an attribute of the transfer. Saving the maximum transfer duration coefficient for which the transfer is feasible will similarly enable faster pruning during the search phase. It would also be possible to add a minimum duration coefficient for which the transfer can be useful if above that speed the previous destination trip of the same line can be taken instead. In the case where possible speed values are from a discrete set, a speed mask can be added to transfers in order to keep only the right ones for each speed during the query phase.

Now each transfer of the transfer set is a triplet $(t@i \to u@j, \Delta\tau, \sigma_{\max})$ where $\Delta\tau$ is the standard transfer duration and $\sigma_{\max}$ the maximum duration coefficient for which the transfer is feasible. The user gives as additional inputs a maximum transfer duration $\Delta\tau_{\max}$ and transfer duration coefficient σ. They are used to prune the transfers during the search phase when exploring the neighbourhood of the trips. If a transfer $(t@i \to u@j, \Delta\tau, \sigma_{\max})$ is such that $\Delta\tau_{\max} > \sigma\,\Delta\tau$ or $\sigma > \sigma_{\max}$, it can be pruned. For concision, the pseudo-code of the modified query algorithm can be found in appendix in Algorithm 2.

Note that it would also be possible to bound the travel duration from origin to the first stop or from the last stop to destination by pruning the initial queue Q_0 and the target set $\mathcal{L}$ according to a user defined value (possibly different of $\Delta\tau_{\max}$).

It has been proven in [6] that a correct transfer set for EAT queries is correct for latest departure time queries (LDT). LDT queries could hence be modified similarly to integrate maximum duration and variable transfer speed.

3.2 Preprocessing phase

When considering multiple possible speeds, there might be several transfers of interest for a given origin trip t at stop $t@i$ toward a given destination line L' at $L'@j$, instead of a single one. The smallest destination trip to consider is the earliest trip such that the transfer is feasible with the fastest possible speed:

$$u_{\min} = \min\{u \in L' \,|\, \tau_{dep}(u,j) \geq \tau_{arr}(t,i) + \varsigma_{\min}\,\Delta\tau_{\text{fp}}(t@i, L'@j)\}$$

The latest corresponds the slowest speed:

$$u_{\max} = \min\{u \in L' \,|\, \tau_{dep}(u,j) \geq \tau_{arr}(t,i) + \varsigma_{\max}\,\Delta\tau_{\text{fp}}(t@i, L'@j)\}$$

And all the trips of L' in between could be taken, depending on the user chosen transfer speed, each corresponding to a transfer to an earliest trip for a given speed range. We call *trips of interest* of the transfer $t@i \to L'@j$ the destination trips of L' in $\{u_{\min}, \dots, u_{\max}\}$. When the set of possible speed values is finite, not all the trips of the range $\{u_{\min}, \dots, u_{\max}\}$ might be relevant, and we will consider only the earliest for each speed of the set.

For each feasible transfer described above, we save in the transfer set the t-uple $(t@i \to u@j, \Delta\tau_{\text{fp}}(t@i, u@j), \sigma_{\max})$ where $\sigma_{\max}$ is the maximum duration coefficient such that the transfer is feasible.

Note that the obtained transfer set is a correct transfer set. However, as explained, the query times would be impacted by the unnecessary transfers. We will hence consider both the line-based and the arrival time-based prunings and explain how to modify them to take into account a customizable transfer speed and maximum transfer duration.

3.2.1 Pruning based on lines

The line-based pruning is based on a dominance relation between transfers. Since we want to customize the maximum transfer duration, a transfer $(t@i \to u@j, \Delta\tau, \sigma_{\max})$ cannot be dominated by a transfer $(t@i \to v@j, \Delta\tau', \sigma'_{\max})$ such that $\Delta\tau' > \Delta\tau$. Indeed, the second transfer could be forbidden by the custom maximum transfer duration, while the first is not. Similarly, as $\sigma_{\max}$ is the maximum duration coefficient such that the first transfer is feasible, if $\sigma'_{\max} < \sigma_{\max}$, the second transfer cannot dominate the first. We hence obtain the following dominance relation. A transfer $(t@i \to u@j, \Delta\tau, \sigma_{\max})$ *is dominated* by a transfer $(t@i' \to u'@j', \Delta\tau', \sigma'_{\max})$ if and only if

$$\begin{array}{lllllllllll} i \leq i' & \text{and} & u' \leq u & \text{and} & j' \leq j & \text{and} & \Delta\tau' \leq \Delta\tau & \text{and} & \sigma_{\max} \leq \sigma'_{\max} & \text{and} \\ (i < i' & \text{or} & u' < u & \text{or} & j' < j & \text{or} & \Delta\tau' < \Delta\tau & \text{or} & \sigma_{\max} < \sigma'_{\max}) & & \end{array}$$

Using this condition, it is possible to prune the transfer set as before. However, it is expected that the percentage of pruned transfers will be lower, as the dominance condition is stronger and that preprocessing will be longer, as additional comparisons need to be performed. Corresponding pseudo-code can be found in appendix in Algorithm 4 describing the modified preprocessing that builds the search graph arc set.

3.2.2 Pruning based on arrival times

Remember that in the original TB algorithm, a transfer is removed from the set of possible transfers if previously scanned transfers allow for reaching the same stops at the same or an earlier time. As the transfers are scanned starting from the end of the origin line, later transfers are kept in case of identical arrival times. Now, we want to consider the possibility to disable some transfers at query time according to maximal duration or if speed customization makes the transfer time too long to reach the destination trip before it leaves. Applying the same pruning will not be correct, as a transfer can be removed because of previously checked transfers with longer duration. As a consequence, we consider for each tentative arrival time at a stop the transfer time and the maximum duration coefficient for which the transfer is feasible. Also, comparing arrival times is made more difficult by the speed variability. All arrival times, in this preprocessing, have a speed independent component corresponding to the arrival time of a trip at one stop of its sequence. Then, when reaching additional stations by footpaths, the duration is dependent of speed. Obviously, simply comparing the sum of the two is not correct, as the variable part will be multiplied by a duration coefficient.

We hence label the stops with a bag of t-uples instead of a single value. Each t-uple indicates arrival time, fixed and variable parts, standard transfer duration and maximum duration coefficient for the transfer to be feasible. We denote $(arr_f, arr_v, \Delta\tau, \sigma_{\max})$ such a label, with arr_f the fixed arrival time part, arr_v the variable arrival time part with standard speed, $\Delta\tau$ the standard duration, and $\sigma_{\max}$ the maximum duration coefficient. A transfer is removed from the set if it doesn't improve any of the label bags of the reached stops (i.e. its labels are dominated at each stop). If we compare the labels $(arr_f, arr_v, \Delta\tau, \sigma_{\max})$ and $(arr'_f, arr'_v, \Delta\tau', \sigma'_{\max})$, $(arr_f, arr_v, \Delta\tau, \sigma_{\max})$ is dominated if and only if:

(a) $\sigma_{\max} \leq \sigma'_{\max}$
(b) $\Delta\tau' \leq \Delta\tau$
(c) $\forall\sigma \in [\varsigma_{\min}, \varsigma_{\max}], \quad arr'_f + arr'_v \times \sigma \leq arr_f + arr_v \times \sigma$

Conditions (a) and (b) correspond to classical Pareto dominance between criterion values. Condition (c) corresponds to arrival time dominance, but must be true for all possible speeds. It is equivalent to:

$$\forall\sigma \in [\varsigma_{\min}, \varsigma_{\max}], \quad \frac{arr'_f - arr_f}{\sigma} \leq arr_v - arr'_v \tag{1}$$

In particular, inequation (1) is true if it is true for the minimum value $\varsigma_{\min}$ that duration coefficient σ can take, obtaining the following conditions for label dominance:

(a) $\sigma_{\max} \leq \sigma'_{\max}$
(b) $\Delta\tau' \leq \Delta\tau$
(c) $\frac{arr'_f - arr_f}{\varsigma_{\min}} \leq arr_v - arr'_v$

We maintain for each stop a bag of all the non-dominated labels to compare with new entries. We keep a transfer when it updates at least one label bag.

Note that in the case where possible speeds are only within a small discrete set (for instance slow, standard, fast), it is possible to save one label bag per stop and speed and use simpler labels with arrival time (computed for the given speed) and standard transfer duration (or transfer duration computed for the given speed). Each transfer is feasible for a subset of the speeds and can hence update label bags for each of those speeds.

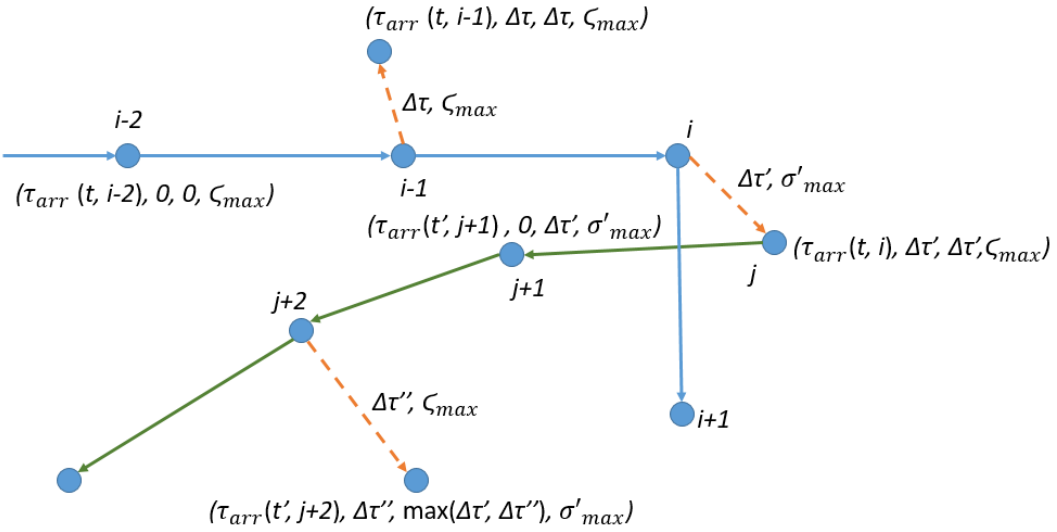

Figure 1 Arrival time labels for a given transfer $t@i \to t'@j$.

Figure 1 shows some tentative labels for a single transfer between a trip t (above) and a trip t' (below). First the stops of $\overrightarrow{p}(t)$ are marked with a fixed part equal to the trip arrival time, and as no transfer is performed, a null variable part and maximum transfer time and maximum duration coefficient $\varsigma_{\max}$. After transferring from t to t', the stops of $\overrightarrow{p}(t')$ are marked by the arrival time of t', a null variable arrival time part, standard transfer duration and maximum duration coefficient from the transfer between t and t'. When performing transfers from the stops of the trips' stop sequences to reach additional stops, we do not know which trips will be taken later. As a consequence, we use $\varsigma_{\max}$ as a bound. A path with several transfers is feasible for a given duration coefficient only if all its transfers' maximum coefficients are higher. Similarly, if a maximum transfer duration value is provided, the path is feasible only if all the transfer times are bellow the given bound. We hence take the maximum of the successive transfer durations and the minimum of the maximum transfer duration coefficient to mark additional stops reached from t'. As in [15], we check also minimum change times (giving them similar labels and multiplying them by a duration coefficient when speed varies). Algorithm 1 describes this pruning phase.

3.3 Correctness

To prove that the preprocessing steps build correct transfer sets, we need to prove that for any value in the Pareto front, there is an optimal solution with this value such that all its transfers are in the computed transfer set. For each preprocessing step, we prove it by constructing such a solution from any optimal solution.

▶ **Proposition 1.** *The modified line-based preprocessing (Algorithm 4) computes a correct set $\mathcal{T}$ of transfers for earliest arrival time and minimum number of transfers.*

Proof. Consider an optimal solution s for a given duration coefficient σ and a maximum transfer duration $\Delta\tau_{\max}$ with at least one transfer. It can be described by its trip segment sequence: $s = \langle t_1@j_1 \to t_1@i_1, t_2@j_2 \to t_2@i_2 \ldots, t_{k+1}@j_{k+1} \to t_{k+1}@i_{k+1}\rangle$ with L_i the line of the trip t_i, for $i \in \{1, \ldots, k+1\}$. Consider the first transfer $t_1@i_1 \to t_2@j_2$ of s. If it belongs to the transfer set $\mathcal{T}(t_1, L_2)$ of t_1 to L_2 obtained at the end of the pruning, we can move to the next transfer.

Otherwise, if t_2 is not in the set T_2 of trips of interest of transfer $t_1@i_1 \to L_2@i_2$, we can replace it with a transfer to the maximum trip u of T_2 such that $u \leq t_2$ as it can only improve arrival time at $L_2@i_2$ while keeping the transfer feasible for the same speed range, including duration coefficient σ.

Now, we suppose that $t_2 \in T_2$ but that $(t_1@i_1 \to t_2@j_2, \Delta\tau_1, \sigma^1_{\max}) \notin \mathcal{T}(t_1, L_2)$, which means that it has been pruned. Since pruned transfers are dominated, there exists a transfer $(t_1@i \to t@j, \Delta\tau, \sigma_{\max})$ of $\mathcal{T}(t_1, L_2)$ such that $i \geq i_1$, $j \leq j_2$, $t \leq t_2$, $\Delta\tau \leq \Delta\tau_1$ and $\sigma^1_{\max} \leq \sigma_{\max}$. If $k > 1$, transfer $t@i_2 \to t_3@j_3$ is feasible, since transfer $t_2@i_2 \to t_3@j_3$ is feasible and $t \leq t_2$. In solution s, we can hence replace $t_1@j_1 \to t_1@i_1$ by $t_1@j_1 \to t_1@i$, and $t_2@j_2 \to t_2@i_2$ by $t@j \to t@i_2$ to obtain a new solution s'.

As the new solution uses a transfer $(t_1@i \to t@j, \Delta\tau, \sigma_{\max})$ such that $\Delta\tau \leq \Delta\tau_1 \leq \Delta\tau_{\max}$ and $\sigma \leq \sigma^1_{\max} \leq \sigma_{\max}$, it is feasible for custom speed and custom maximum transfer time. It also has an at least as good arrival time as s, and the same number of transfers. They are hence both optimal with the same value.

Processing the transfers of s one after the other, we iteratively replace all the transfers that do not belong to the pruned transfer set $\mathcal{T}$ by transfers belonging to it. The optimal solution obtained is equivalent to s while using only transfers of $\mathcal{T}$, which completes the proof. ◀

Algorithm 1 Modifications of arrival time based pruning.

Input: Timetable data, footpath data, transfer set $\mathcal{T}$
Input: System maximum duration coefficient $\varsigma_{\max}$
Output: Reduced transfer set $\mathcal{T}$
for each trip t **do**
  $\tau_A(.) \leftarrow \emptyset$ ▷ Label bag with earliest arrival time at stops
  $\tau_C(.) \leftarrow \emptyset$ ▷ Label bag with earliest change time at stops
  for $i \leftarrow |\overrightarrow{p}(t)| - 1, \ldots, 1$ **do**
    Update $\tau_A(t@i)$ with label $(\tau_{arr}(t,i), 0, 0, \varsigma_{\max})$
    Update $\tau_C(t@i)$ with label $(\tau_{arr}(t,i), \Delta\tau_{\text{fp}}(t@i, t@i), \tau_{\text{fp}}(t@i, t@i), \varsigma_{\max})$
    for each stop $q \neq t@i$ such that $\Delta\tau_{\text{fp}}(t@i,\ q)$ is defined **do**
      Update $\tau_A(q)$ with label $(\tau_{arr}(t,i), \Delta\tau_{\text{fp}}(t@i,\ q), \Delta\tau_{\text{fp}}(t@i,\ q), \varsigma_{\max})$
      Update $\tau_C(q)$ with label $(\tau_{arr}(t,i), \Delta\tau_{\text{fp}}(t@i,\ q), \Delta\tau_{\text{fp}}(t@i,\ q), \varsigma_{\max})$
    end for
    for each transfer $(t@i \to u@j, \Delta\tau, \sigma_{\max}) \in \mathcal{T}$ **do**
      $keep \leftarrow$ **false**
      for each stop $u@k$ on trip u with $k > j$ **do**
        if $(\tau_{arr}(u,k), 0, \Delta\tau, \sigma_{\max})$ is not dominated in $\tau_A(u@k)$ **then**
          Update $\tau_A(u@k)$ with $(\tau_{arr}(u,k), 0, \Delta\tau, \sigma_{\max})$
          $keep \leftarrow$ **true**
        end if
        $lab_C \leftarrow (\tau_{arr}(u,k), \Delta\tau_{\text{fp}}(u@k,\ u@k), \max\left(\Delta\tau, \Delta\tau_{\text{fp}}(u@k,\ u@k)\right), \sigma_{\max})$
        if lab_C is not dominated in $\tau_C(u@k)$ **then**
          Update $\tau_C(u@k)$ with lab_C
          $keep \leftarrow$ **true**
        end if
        for each stop $q \neq u@k$ such that $\Delta\tau_{\text{fp}}(u@k,\ q)$ is defined **do**
          $lab \leftarrow (\tau_{arr}(u,k), \Delta\tau_{\text{fp}}(u@k,\ q), \max\left(\Delta\tau, \Delta\tau_{\text{fp}}(u@k,\ q)\right), \sigma_{\max})$
          if lab is not dominated in $\tau_A(q)$ **then**
            Update $\tau_A(q)$ with lab
            $keep \leftarrow$ **true**
          end if
          if lab is not dominated in $\tau_C(q)$ **then**
            Update $\tau_C(q)$ with lab
            $keep \leftarrow$ **true**
          end if
        end for
      end for
      if $\neg keep$ **then**
        $\mathcal{T} \leftarrow \mathcal{T} \setminus \{(t@i \to u@j, \Delta\tau, \sigma_{\max}\}$ ▷ No improvement: remove the transfer
      end if
    end for
  end for
end for

▶ **Proposition 2.** *The modified arrival time based preprocessing (Algorithm 1) computes a correct set $\mathcal{T}$ of transfers for earliest arrival time and minimum number of transfers.*

Proof. Consider again an optimal solution s' with at least one transfer for an origin stop org, a target stop tgt, a duration coefficient σ and a maximum transfer duration $\Delta\tau_{\max}$. We consider both the cases where line-based pruning is applied to the set of transfers of interest and where the set of transfers of interest is pruned directly without line-based pruning. From the proof of Proposition 1, we can construct in both cases another optimal solution s from s' (possibly equal to s') such that all its transfers are in the input transfer set given to the arrival time based pruning as input.

We again describe s by its trip segment sequence, but we add the origin and target stops at the beginning and the end of the sequence:
$s = \langle org, t_1@j_1 \to t_1@i_1, t_2@j_2 \to t_2@i_2 \ldots, t_{k+1}@j_{k+1} \to t_{k+1}@i_{k+1}, tgt\rangle$
with L_i the line of the trip t_i, for $i \in \{1, \ldots, k+1\}$.

Suppose that the first transfer $(t_1@i_1 \to t_2@j_2, \Delta\tau, \sigma_{\max})$ of s is not in $\mathcal{T}$. If it is the last transfer ($k = 1$), it means that there exists a transfer $(t_1@i'_1 \to t'_2@j'_2, \Delta\tau', \sigma'_{\max})$ of $\mathcal{T}$ such that $i'_1 \geq i_1$ (as later transfers are scanned first) and target is reachable from $t'_2@i'_2$ for an index $i'_2 > j'_2$ and $l' = (\tau_{arr}(t'_2, j'_2), \Delta\tau_{\text{fp}}(t'_2@j'_2, tgt), \max\{\Delta\tau', \Delta\tau_{\text{fp}}(t'_2@j'_2, tgt)\}, \sigma'_{\max})$ is dominating $l = (\tau_{arr}(t_2, j_2), \Delta\tau_{\text{fp}}(t_2@j_2, tgt), \max\{\Delta\tau, \Delta\tau_{\text{fp}}(t_2@j_2, tgt)\}, \sigma_{\max})$ for the arrival time label bag $\tau_A(tgt)$. As $\Delta\tau' \leq \max\{\Delta\tau, \Delta\tau_{\text{fp}}(t_2@j_2, tgt)\} \leq \Delta\tau_{\max}$ and $\sigma \leq \sigma_{\max} \leq \sigma'_{\max}$, this transfer is feasible for custom parameters $\Delta\tau_{\max}$ and σ. Note that target could not be reached directly from one of the stops of t and arrival time be at least as good as that of s since s is optimal and has hence the minimum number of trips for its arrival time. The solution $\hat{s} = \langle org, t_1@j_1 \to t_1@i'_1, t'_2@j'_2 \to t'_2@i'_2, tgt\rangle$ has hence the same arrival time as s but its transfers belong to $\mathcal{T}$.

Now, consider the case where transfer $t_1@i_1 \to t_2@j_2$ is not the last transfer of s. As transfer $(t_1@i_1 \to t_2@j_2, \Delta\tau, \sigma_{\max})$ has been pruned, there exist a transfer $t_1@i'_1 \to t'_2@j'_2$ of $\mathcal{T}$ such that $i'_1 \geq i_1$, $t_3@j_3$ can be reached from the trip segment $t'_2@j'_2 \to t'_2@i'_2$ and the label $l' = (\tau_{arr}(t'_2, j'_2), \Delta\tau_{\text{fp}}(t'_2@j'_2, t_3@j_3), \max\{\Delta\tau', \Delta\tau_{\text{fp}}(t'_2@j'_2, t_3@j_3)\}, \sigma'_{\max})$ is dominating $l = (\tau_{arr}(t_2, j_2), \Delta\tau_{\text{fp}}(t_2@j_2, t_3@j_3), \max\{\Delta\tau, \Delta\tau_{\text{fp}}(t_2@j_2, t_3@j_3)\}, \sigma_{\max})$ for the change time label bag $\tau_C(t_3@j_3)$. As previously, this transfer exists since arriving at $t_3@j_3$ directly from t_1 at a time at least as good as that of s without performing a transfer would mean that s is not optimal. The transfer is also feasible for custom parameters $\Delta\tau_{\max}$ and σ. From dominance condition (c), the change time at $t_3@j_3$ is identical or improved for all duration coefficients, including σ. It will hence be possible to board trip t_3 at index j_3 after performing the transfer. We can hence replace $t_1@i_1 \to t_2@j_2$ by $t_1@i'_1 \to t'_2@j'_2$ in solution s.

Repeating this procedure for the transfers of s in order leads to build a solution $\hat{s}$ with the same number of transfers as s, the same arrival time and all its transfers in $\mathcal{T}$. ◀

4 Experiments

To evaluate the computation time performances, we implemented the proposed algorithms in rust and ran our experiments on a 64 2.7 GHz CPU Intel(R) Xeon(R) CPU E5-4650 server with 20 M of L3 cache and 504 GB of RAM. We used two large size data sets. The first covers the Région Île-De-France and is provided by IDFM [8] (Île-De-France Mobilités). The footpaths are computed with an OSRM [10] monomodal routing server using OSM [9] road data with a standard speed of 4 kph. To compare the impact of different maximum transfer times, two footpath sets are generated: one with a maximum of 10 min between two adjacent stops and one with a maximum of 30 min. The second data set is provided by Naver Map

Table 1 Data sets.

Data set	Nb stops	Nb trips	Nb lines	Nb connections	Nb footpaths (10 min)	Nb footpaths (30 min)
IDFM	42.3 K	319.2 K	1.9 K	103.8 M	846.2 K	7.186 M
Korea	180.9 K	446.7 K	31.7 K	241.9 M	4.196 M	–

Table 2 Preprocessing for IDFM with maximum 10 min and 30 min transfer time.

Version	IDFM (10 min)			IDFM (30 min)		
	# kept transfers	# removed transfers	Mean duration (s)	# kept transfers	# removed transfers	Mean duration (s)
Standard	98.1 M	1 314.8 M	44	135.9 M	8 382.9 M	692
Variable speed	153.0 M	1 443.9 M	94	320.0 M	11 786.6 M	1 373
Max. duration and var. speed	242.9 M	1 353.9 M	1 892	732.8 M	11 374 M	96 616

and contains public transit information for Korea and footpaths those maximum value is 10 min. We illustrate on this one the impact of the arrival time based preprocessing compare to line-based pruning only. Table 1 gives the respective sizes of the two networks.

To test the proposed algorithms in a standard context, we allow for 3 different speeds (slow: 2 kph, standard: 4 kph and fast: 6 kph). We hence have $\varsigma_{\max} = 2$ and $\varsigma_{\min} = 2/3$. We compare 3 versions of the code: the standard version without customization, a version with speed customization and a version with speed and maximum transfer duration customization.

4.1 Preprocessing

As explained, with speed customization, there might be several transfers of interest from each origin trip-index pair to each reachable line-index pair. The total number of feasible transfers before pruning is hence increased (see Table 2 and Table 3) and the preprocessing is more computationally expensive. Enabling the maximum transfer duration constraint also increases the number of kept transfers as conditions for removal are harder to fulfil. The final number of transfers for each speed is indicated in appendix (see Table 6 and Table 7).

The preprocessing times for maximum duration and variable speed are considerably increased compare to the standard version, while variable speed only multiply them by 2.33. Indeed, label bag updating is much more expensive than taking the minimum between two arrival times. As we use a straightforward implementation for those label bag updates and as the number of labels can be large for one stop, the computation times are significantly impacted for arrival time based pruning. However, they remain in an acceptable range for public transit data update made every two or three days, which is often the case. On the

Table 3 Preprocessing for Korea with maximum 10 min transfer time.

Version	Line based pruning			All prunings		
	nb kept transfers	nb removed transfers	Mean duration (s)	nb kept transfers	nb removed transfers	Mean duration (s)
Standard	608.6 M	2192.2 M	89	238.1 M	3 251.9 M	170
Variable speed	1 085.5 M	2773.9 M	116	463.8 M	4 106.3 M	490
Max. duration and var. speed	1 520.1 M	2.339.2 M	140	658.1 M	3 912.1 M	14 773

other hand, line-based pruning is less impacted in terms of computation times since less transfers are compared at once (only those to the same line) and the (c) condition of arrival time based pruning is not necessary. It can hence be considered as an alternative when more frequent updates are needed, at the price of slower query times.

4.2 Query phase

For each data set, we generated uniformly at random 100 origin-destination pairs from stop to stop. We run earliest arrival time queries and one-hour profile queries starting at 8.30 am (rush hour is the densest in term of number of trips and transfers).

Table 4 presents the mean execution times and number of solutions for EAT queries with the different versions of the algorithm given a selected speed. As expected, using appropriate transfer structure with speed mask, the execution times are not much impacted by the existence of several speeds instead of one. They are increased compare to that of the standard code without any modifications as the number of transfers is larger, but not much. Remember that to divide by 3 the execution time, the number of transfers removed is 9 out of 10 in the standard version [15]. Here when the number of transfers is multiplied by 2.35 for IDFM 30 min, the mean query duration is multiplied by 1.49 compared to standard version while it includes additional transfer checking. The query times of the different speed values are similar.

When adding the possibility to set maximum transfer duration (see Table 5), the number of transfers is multiplied by 5.39 for IDFM 30 min and the computation times are multiplied by 2.04 for standard speed compare to standard version. Different values of maximum transfer time hardly impact the query times with only a few milliseconds difference between 20 min, 10 min, 5 min and no restriction.

The results are similar for the other two networks and we can conclude that although the modification does increase the query times, those remain sufficiently low for interactive queries in a production application, with at most half a second of execution time for the Korean network.

Numerical results for profile queries can be found in appendix in Tables 8 and 9, and are similar to that of EAT queries.

5 Conclusion

In this article, we extend the Trip-Based Public Transit Routing algorithm, to take into account *at query time* user defined transfer speed and maximum transfer duration, while keeping the optimality for the bicriteria problem of optimizing minimum arrival time and number of transfers. The tests on two large scale data sets show that the preprocessing steps are significantly slower, but the query times are much less increased and still compatible with real-time queries in a production context. Many other algorithms of the literature rely on preprocessing steps using fixed sets of transfers of immutable duration. It would hence be interesting to design similar extensions for those algorithms, in particular for the ones relying on unbounded transfer duration, where transfers in an optimal solution can be very long without customization.

Table 4 EAT queries at 8.30 am.

Data set	Version	Speed	Mean query time (ms)	Mean nb solutions
IDFM 30 min	Standard	-	75	1.86
IDFM 30 min	Variable speed	Standard	112	1.86
IDFM 30 min	Variable speed	Slow	108	1.76
IDFM 30 min	Variable speed	Fast	108	2.03
IDFM 30 min	Max. duration - var. speed	Standard	153	1.86
IDFM 30 min	Max. duration - var. speed	Slow	134	1.76
IDFM 30 min	Max. duration - var. speed	Fast	130	2.03
IDFM 10 min	Standard	-	91	1.71
IDFM 10 min	Variable speed	Standard	98	1.71
IDFM 10 min	Variable speed	Slow	94	1.72
IDFM 10 min	Variable speed	Fast	100	1.76
IDFM 10 min	Max. duration - var. speed	Standard	117	1.71
IDFM 10 min	Max. duration - var. speed	Slow	108	1.72
IDFM 10 min	Max. duration - var. speed	Fast	107	1.76
Korea	Standard	-	316	2.00
Korea	Variable speed	Standard	418	2.00
Korea	Variable speed	Slow	356	1.93
Korea	Variable speed	Fast	374	2.12
Korea	Max. duration - var. speed	Standard	583	2.00
Korea	Max. duration - var. speed	Slow	531	1.93
Korea	Max. duration - var. speed	Fast	543	2.12

Table 5 EAT queries at with user defined maximum transfer time and speed customization, standard speed.

Data set	Max transfer time (min)	Mean query time (ms)	Mean nb solutions
IDFM 30 min	-	153	1.86
IDFM 30 min	20	150	1.83
IDFM 30 min	10	158	1.87
IDFM 30 min	5	155	2.02
IDFM 10 min	-	145	1.71
IDFM 10 min	5	150	1.96
Korea	-	565	2.00
Korea	5	538	1.97

References

1 Hannah Bast, Erik Carlsson, Arno Eigenwillig, Robert Geisberger, Chris Harrelson, Veselin Raychev, and Fabien Viger. Fast routing in very large public transportation networks using transfer patterns. In *Proceedings of the 18th Annual European Conference on Algorithms: Part I*, ESA'10, pages 290–301, Berlin, Heidelberg, 2010. Springer-Verlag. URL: http://dl.acm.org/citation.cfm?id=1888935.1888969.

2 Moritz Baum, Valentin Buchhold, Jonas Sauer, Dorothea Wagner, and Tobias Zündorf. UnLimited TRAnsfers for Multi-Modal Route Planning: An Efficient Solution. In Michael A. Bender, Ola Svensson, and Grzegorz Herman, editors, *27th Annual European Symposium on Algorithms (ESA 2019)*, volume 144 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:16, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ESA.2019.14.

3 Daniel Delling, Julian Dibbelt, Thomas Pajor, and Renato F. Werneck. Public Transit Labeling. In Evripidis Bampis, editor, *Experimental Algorithms*, pages 273–285. Springer International Publishing, 2015. doi:10.1007/978-3-319-20086-6_21.

4 Daniel Delling, Thomas Pajor, and Renato F. Werneck. Round-based public transit routing. In *Proceedings of the Fourteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 130–140, 2012. doi:10.1137/1.9781611972924.13.

5 Julian Dibbelt, Thomas Pajor, Ben Strasser, and Dorothea Wagner. Intriguingly simple and fast transit routing. In Vincenzo Bonifaci, Camil Demetrescu, and Alberto Marchetti-Spaccamela, editors, *Experimental Algorithms. SEA 2013*, volume 7933 of *Lecture Notes in Computer Science*, pages 43–54, Berlin, Heidelberg, 2013. Springer. doi:10.1007/978-3-642-38527-8_6.

6 Vassilissa Lehoux and Darko Drakulic. Mode Personalization in Trip-Based Transit Routing. In Valentina Cacchiani and Alberto Marchetti-Spaccamela, editors, *19th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2019)*, volume 75 of *OpenAccess Series in Informatics (OASIcs)*, pages 13:1–13:15, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/OASIcs.ATMOS.2019.13.

7 Vassilissa Lehoux and Christelle Loiodice. Faster Preprocessing for the Trip-Based Public Transit Routing Algorithm. In Dennis Huisman and Christos D. Zaroliagis, editors, *20th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2020)*, volume 85 of *OpenAccess Series in Informatics (OASIcs)*, pages 3:1–3:12, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. doi:10.4230/OASIcs.ATMOS.2020.3.

8 Île De France Mobilités. Open data. URL: https://www.iledefrance-mobilites.fr.

9 OSM. Open street map. URL: https://www.openstreetmap.org.

10 OSRM. Open source routing machine. URL: http://project-osrm.org/.

11 Duc-Minh Phan and Laurent Viennot. Fast public transitrouting with unrestrictedwalking through hub labeling. In *Proceedings of the Special Event on Analysis of Experimental Algorithms (SEA2019)*, volume 11544 of *Lecture Notes in Computer Science*. Springer, Cham, 2019. doi:10.1007/978-3-030-34029-2_16.

12 Andrea Raith, Marie Schmidt, Anita Schöbel, and Lisa Thom. Extensions of labeling algorithms for multi-objective uncertain shortest path problems. *Networks*, 72(1):84–127, 2018. doi:10.1002/net.21815.

13 Luis Ulloa, Vassilissa Lehoux, and Frédéric Roulland. Trip Planning Within a Multimodal Urban Mobility. *IET Intelligent Transport Systems*, 12(2):87–92, 2018. doi:10.1049/iet-its.2016.0265.

14 Dorothea Wagner and Tobias Zündorf. Public Transit Routing with Unrestricted Walking. In Gianlorenzo D'Angelo and Twan Dollevoet, editors, *17th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2017)*, volume 59 of *OpenAccess Series in Informatics (OASIcs)*, pages 7:1–7:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/OASIcs.ATMOS.2017.7.

15 Sacha Witt. Trip-based public transit routing. In N. Bansal and I. Finocchi, editors, *ESA 2015*, volume 9294 of *Lecture Notes in Computer Science*, Berlin, Heidelberg, 2015. Springer. doi:10.1007/978-3-662-48350-3_85.

A Algorithms

Algorithm 2 Earliest arrival query.

input Timetable data, transfer set $\mathcal{T}$
input Source stop p_{src}, destination stop p_{tgt}, start time τ
input Maximum transfer duration $\Delta\tau_{\max}$, transfer duration coefficient σ
output Result set J
$J \leftarrow \emptyset$, $\mathcal{L} \leftarrow \emptyset$
$Q_n \leftarrow \emptyset$ for $n = 0, 1, \ldots$
$R(.) \leftarrow \infty$ for all trips t

INITIALIZATION()
$\tau_{\min} \leftarrow \infty$ ▷ The current minimum arrival time at target
$n \leftarrow 0$
while $Q_n \neq \emptyset$ **do**
  for each $t@b \rightarrow t@e \in Q_n$ **do**
    ▷ Checking if a target is reached
    for each $(L_t, i, \Delta\tau) \in \mathcal{L}$ with $b < i$ and $\tau_{arr}(t, i) + \Delta\tau < \tau_{\min}$ **do**
      $\tau_{\min} \leftarrow \tau_{arr}(t, i) + \Delta\tau$
      $J \leftarrow J \cup \{(\tau_{\min}, n)\}$, removing dominated entries
    end for

    if $\tau_{arr}(t, b+1) < \tau_{\min}$ **then** ▷ Filling the queue for the next round
      for each transfer $(t@i \rightarrow u@j, \Delta\tau, \sigma_{\max}) \in \mathcal{T}$ with $b < i \leq e$ and
          $\sigma \times \Delta\tau \leq \Delta\tau_{\max}$ and $\sigma \leq \sigma_{\max}$ **do**
        ENQUEUE(u, j, $n+1$)
      end for
    end if
  end for
  $n \leftarrow n + 1$
end while

Algorithm 3 Auxiliary procedures.

procedure INITIALIZATION
  for each stop q such that $\Delta\tau_{\text{fp}}(q, p_{tgt}$ is defined **do**
    $\Delta\tau \leftarrow 0$ if $p_{tgt} = q$, else $\Delta\tau = \sigma \times \Delta\tau_{\text{fp}}(q, p_{tgt})$
    for each $(L, i) \in \mathbf{L}(q)$ **do**
      $\mathcal{L} \leftarrow \mathcal{L} \cup \{(L, i, \Delta\tau\}$
    end for
  end for

  for each stop q such that $\Delta\tau_{\text{fp}}(p_{src}, q)$ is defined **do**
    $\Delta\tau = 0$ if $p_{src} = q$, else $\Delta\tau = \sigma \times \Delta\tau_{\text{fp}}(p_{src}, q)$
    for each $(L, i) \in \mathbf{L}(q)$ **do**
      $t \leftarrow$ earliest trip of L such that $\tau + \Delta\tau \leq \tau_{dep}(t, i)$
      ENQUEUE$(t, i, 0)$
    end for
  end for
end procedure
procedure ENQUEUE(trip t, index i, nb transfers n)
  if $i < R(t)$ **then**
    $Q_n \leftarrow Q_n \cup \{t@i \rightarrow t@R(t)\}$
    for each trip u with $t \leq u$ and $L_t = L_u$ **do**
      $R(u) \leftarrow \min(R(u), i)$
    end for
  end if
end procedure

Algorithm 4 Modification of transfer set building.

Input: Timetable data, footpath data
Input: Maximum and minimum transfer duration coefficients $\varsigma_{\max}$ and $\varsigma_{\min}$
Output: Reduced transfer set $\mathcal{T}$
$\mathcal{T} \leftarrow \emptyset$
for each line L **do**
  $\mathcal{T}(L) \leftarrow LINE\_TRANSFERS(L)$
  for each trip t of L **do**
    $T \leftarrow \emptyset$ ▷ Transfer set for each target line
    $L_{prev} \leftarrow$ **null**
    for each transfer $(i, L'@j, \Delta\tau)$ of $\mathcal{T}(L)$ **do**
      if $L_{prev} \neq L'$ **then**
        $\mathcal{T} \leftarrow \mathcal{T} \cup T$
        $T \leftarrow \emptyset$, $L_{prev} = L'$
      end if
      $t'_{\min} \leftarrow$ earliest trip of L' at j such that $\tau_{dep}(t'_{\min}, j) \geq \tau_{arr}(t, i) + \Delta\tau \times \varsigma_{\min}$
      $t'_{\max} \leftarrow$ earliest trip of L' at j such that $\tau_{dep}(t'_{\max}, j) \geq \tau_{arr}(t, i) + \Delta\tau \times \varsigma_{\max}$
      $Labs \leftarrow \emptyset$
      for each trip t', $t'_{\min} \leq t' \leq t'_{\max}$ **do**
        $\sigma_{\max} \leftarrow$ maximum value $\sigma \leq \varsigma_{\max}$ such that $\tau_{dep}(t', j) \geq \tau_{arr}(t, i) + \Delta\tau \times \sigma$
        $Labs \leftarrow Labs \cup \{(t@i \rightarrow t'@j, \Delta\tau, \sigma_{\max})\}$
      end for
      if $T = \emptyset$ **then**
        $T(L') \leftarrow Labs$
      else
        for each $lab \in Labs$ **do**
          if lab is not dominated by an element of T **then**
            Update T with lab
          end if
        end for
      end if
    end for
    $\mathcal{T} \leftarrow \mathcal{T} \cup T$
  end for
end for
return $\mathcal{T}$

procedure LINE_TRANSFERS(line L, footpath data) ▷ Builds the line neighbourhood
  for $i \leftarrow |\overrightarrow{p}(L)| - 1, \ldots, 1$ **do**
    for each stop q such that $\Delta\tau_{\text{fp}}(L@i, q)$ is defined **do**
      for each (L', j) such that $q = L'@j$ **do**
        $\mathcal{T} \leftarrow \mathcal{T} \cup (i, L'@j, \Delta\tau_{\text{fp}}(L@i, L'@j))$
      end for
    end for
  end for
  Sort $\mathcal{T}$ first by target line, then by decreasing origin line index, then by increasing target line index, then by chosen sorting in case of tides
  return $\mathcal{T}$
end procedure

B Experiments

Tables 6 and 7 describe the number of transfers for each speed level for speed customization only and for maximum transfer duration and speed customization. We can observe that the number of transfers are similar for each speed in all configurations.

Table 6 Preprocessing for IDFM with maximum 10 min and 30 min transfer time - Number of transfers for each speed level in millions.

Version	IDFM (10 min)			IDFM (30 min)		
	Fast	Standard	Slow	Fast	Standard	Slow
Variable speed	126.6	114.5	99.5	623.2	639.6	699.0
Max. duration and var. speed	209.4	196.9	179.0	2 364.9	2 328.5	2 268.4

Table 7 Preprocessing for Korea with maximum 10 min transfer time - Number of transfers for each speed level in millions.

Version	Line based pruning			All prunings		
	Fast	Standard	Slow	Fast	Standard	Slow
Variable speed	608.7	631.6	678.9	379.8	324.8	248.1
Max duration and variable speed	1 126.0	1 107.5	1 067.2	548.4	486.7	392.0

Tables 8 and 9 describe the performances of profile queries.

Table 8 One-hour profile queries at 8.30 am with user defined speed.

Data set	Version	Speed	Mean query time (ms)	Mean nb solutions
IDFM 30 min	Standard	-	125	5.26
IDFM 30 min	Variable speed	Standard	202	5.26
IDFM 30 min	Variable speed	Slow	210	4.91
IDFM 30 min	Variable speed	Fast	155	5.6
IDFM 30 min	Max. duration - var. speed	Standard	347	5.26
IDFM 30 min	Max. duration - var. speed	Slow	229	4.91
IDFM 30 min	Max. duration - var. speed	Fast	237	5.6
IDFM 10 min	Standard	-	139	2.1
IDFM 10 min	Variable speed	Standard	146	2.1
IDFM 10 min	Variable speed	Slow	118	2.04
IDFM 10 min	Variable speed	Fast	126	2.28
IDFM 10 min	Max. duration - var. speed	Standard	144	2.1
IDFM 10 min	Max. duration - var. speed	Slow	137	2.04
IDFM 10 min	Max. duration - var. speed	Fast	126	2.28
Korea	Standard	-	586	3.96
Korea	Variable speed	Standard	698	3.96
Korea	Variable speed	Slow	672	3.83
Korea	Variable speed	Fast	682	4.25
Korea	Max. duration - var. speed	Standard	976	3.96
Korea	Max. duration - var. speed	Slow	941	3.83
Korea	Max. duration - var. speed	Fast	950	4.25

Table 9 One-hour profile queries at 8.30 am with user defined maximum transfer time and speed.

Data set	Speed	Max transfer time (min)	Mean query time (ms)	Mean nb solutions
IDFM 30 min	Standard	-	125	5.26
IDFM 30 min	Standard	20	347	5.26
IDFM 30 min	Standard	10	250	5.15
IDFM 30 min	Standard	5	262	5.33
IDFM 10 min	Standard	-	140	2.1
IDFM 10 min	Standard	5	136	2.14
Korea	Standard	-	976	3.96
Korea	Standard	5	989	3.9