DISTRIBUTED GENERATION OF FASTEST PATHS

THOMAS KÄMPKE and MARKUS SCHAAL
Forschungsinstitut für anwendungsorientierte Wissensverarbeitung (FAW),
Helmholtzstr. 16, 89081 Ulm, Germany
e-mail: {kaempke, schaal}@faw.uni-ulm.de

ABSTRACT

Distributed generation of fastest paths is an important issue in future environments of distributed information systems.

Motivated by the project DELFI ("Durchgängige ELektronische FahrplanInformation" / continuous electronic time table information), which involves the design of such an environment for selecting routes provided by public transportation means, we investigate labeling algorithms for fastest path generation and describe a distributed approach for finding fastest paths using an aggregated structure.

Keywords: Labeling algorithms, Shortest paths, Distributed computation, Aggregated structure.

1 INTRODUCTION

This work is motivated by the project DELFI ("Durchgängige ELektronische FahrplanInformation" / continuous electronic time table information), which involves the design of a traffic information system that selects connections provided by public transportation.

Passenger transportation service is typically offered by different companies and authorities and so is information thereabout. Various independent information systems which cover different regions or different means of transportation (trains, buses, airplanes) are to be merged.

The individual systems' independence and the individual system administrators' responsibility for data are to be preserved. We start from the hypothesis that merging is to be facilitated by a high level system, a so-called search controller which extracts information from the individual systems in order to answer queries beyond the scope of each individual system. This is illustrated in figure 1.

The objective considered here is that of finding a connection from a user-specified origin to a user-specified destination so that the transition takes a minimum amount of time from a certain moment onward. Though other objectives clearly exist, this serves as a reasonable proxy.

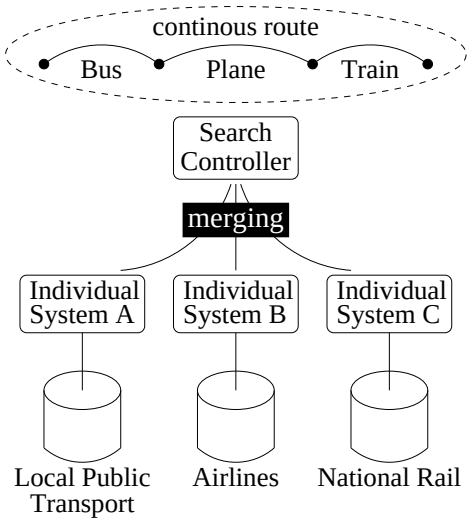

Figure 1: Search controller

In case of agreeability in terms of a certain closing condition specified below, optimal solutions can be found in a two step approach: an aggregated structure shrinks the search space for fastest connections so that subsequent computations on the complete structure provide the fastest overall solution without searching the complete structure.

2 THE FASTEST PATH PROBLEM

2.1 GENERAL ASSUMPTIONS

A directed graph $G = (V, E)$ is assumed to be given over a finite, non-void set V of vertices and a set $E \subseteq V \times V$ of directed edges. The graph G is allowed to contain cycles and it is supposed to be simple meaning that there is at most one arc from vertex i to vertex

j and there are no loops, i.e. no arcs from any vertex i to itself. Hence $|E| = O(n^2)$ for $|V| = n$. G is assumed to be connected meaning that for each $v, w \in V$ there is a path of vertices $p(v,w) = p_G(v,w) = (v = i_0, i_1, \ldots, i_k = w)$ so that $(i_0,i_1), \ldots, (i_{k-1}, i_k) \in E$. Typically, the graph G contains the inverse edge for each of its edges, i.e. $(i,j) \in E \Longrightarrow (j,i) \in E$. All edges are labeled. The label of edge (i,j) is a cost or transition function $c_{ij} : T \to \mathbb{R}_{\geq}$ with index set T denoting time, $T \in \{\mathbb{R}, \mathbb{R}_{\geq}\}$.

The traversal along edge (i,j) starting at time t_0 takes time $c_{ij}(t_0)$. We assume the function $c_{ij}(t)$ being start-arrival-monotone, i.e. $t_a \leq t_b \Rightarrow t_a + c_{ij}(t_a) \leq t_b + c_{ij}(t)$.

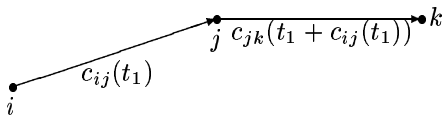

Figure 2: Labeled path without waiting

Let a vertex path $p = (i_1, \ldots, i_r)$ with $(i_1,i_2), \ldots, (i_{r-1}, i_r) \in E$ be given without cycles meaning that $i_s \neq i_t$ for $s \neq t$. This leads to arrival time t_k in vertex i_k with iteration law

$$t_k = t_{k-1} + c_{i_{k-1} i_k}(t_{k-1}) \text{ for } k = 2, \ldots, r.$$

Obviously, $t_k = t_1 + c_{i_1 i_2}(t_1) + \ldots + c_{i_{k-1} i_k}(t_{k-1})$ for $k = 2, \ldots, r$ so that the transition time from vertex i_1 to i_r starting at time t_1 along path $p_G(v,w;t_1) = (v = i_1, \ldots, i_r = w; t_1)$ is

$$t_r - t_1 = c_{i_1 i_2}(t_1) + \ldots + c_{i_{r-1} i_r}(t_{r-1}) =: \vartheta(p_G(v,w;t_1)).$$

Figure 2 sketches the situation with $r = 3$ and $i_1 = i$, $i_2 = j$, and $i_3 = k$. The time dependence of the edge labels has the consequence that an arc being part of different paths and traversed with different departure times may contribute with different traversal times to the overall path traversal times.

Definition 1 *A fastest path $p^0_G(v,w;t_1)$ from v to w starting at time t_1 is one solving the optimization problem*

$$\min_{p_G(v,w;t_1)} \vartheta(p_G(v,w;t_1)).$$

For time independent labels $c_{ij}(t) \equiv c_{ij}$ the fastest path problem obviously reduces to the shortest path problem.

Saw tooth functions are of special importance for edge labels. This type of function is exclusively considered from now on. Suppose, as in a scheduled transportation system there are connections from i to j departing only at fixed times $t^1 < \ldots < t^{N_i}$ requiring proper transition times $d^1, \ldots, d^{N_i} > 0$. Then the cost function c_{ij} is given by

$$c_{ij}(t) = \begin{cases} t^1 - t + d^1, & \text{if } t \leq t^1 \\ t^2 - t + d^2, & \text{if } t^1 < t \leq t^2 \\ \vdots & \vdots \\ t^{N_i} - t + d^{N_i}, & \text{if } t^{N_i - 1} < t \leq t^{N_i} \\ \infty, & \text{if } t^{N_i} < t. \end{cases}$$

The appearance of c_{ij} is sketched in figure 3.

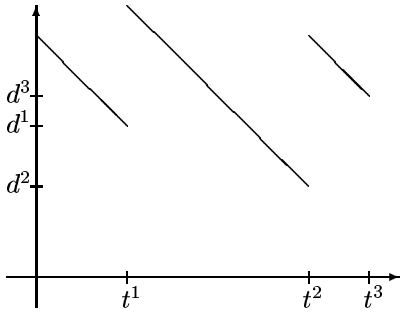

Figure 3: Saw tooth cost function

To make $c_{ij}(t)$ be defined for values $t > t^{N_i}$, ∞ is there replaced by a sufficiently large value M.

An obvious lower bound for a fastest path is given by shortest paths, where all edges receive a positive real-valued label denoting the minimum of all proper transition times along the edge. The length of a path $p_G(v,w) = (v = i_1, \ldots, i_r = w)$ with respect to the time independent labels is denoted by $\lambda(p_G(v,w)) = \lambda(v = i_1, \ldots, i_r = w) = \sum_{j=1}^{r-1} c_{i_j i_{j+1}}$ so that $\lambda(p_G(v,w)) \leq \vartheta(p_G(v,w;t))$ for all $t \geq 0$ and a shortest path is denoted by $p^0_G(v,w) = argmin_{p_G(v,w)} \lambda(p_G(v,w))$.

2.2 RELATED APPROACHES

With respect to aggregation we account for the irregularity of a real world constellation. Thus, regular hierachical graph constructions such as developed for VLSI design are not appropriate here. Rather, present local replacement techniques are loosely related to algebraic graph and order decomposition [7] and separation approaches to decomposition [3].

2.3 STRUCTURAL PROPERTIES

The fastest path problem satisfies the dynamic principle for minimum arrival times. Let therefore u_k denote the minimum arrival time at vertex $w = i_k$ starting at vertex v from t_1 onward.

Theorem 1 *The objective values of the fastest path problem satisfy*

$$\begin{aligned} u_k &= \min_{\{l \in V | (l,k) \in E\}} \{u_l + c_{lk}(u_l)\}, \forall k \in V - \{v\} \\ u_v &:= t_1. \end{aligned}$$

The proof is analogue to the one for shortest paths as given for example in [4].

3 COMPUTATIONS

3.1 GENERAL CASE

Fastest path problems can be solved by a modification of the Dijkstra algorithm. This modification is given as **W1**. It makes use of a list of tentatively labeled vertices L and labels $m(k)$ denoting an upper bound on the arrival time at k. The sets $S(i) = \{j \in V | (i,j) \in E\}$ denote immediate successors of $i \in V$. Instead of considering only one starting point v, the algorithm takes the set $A \subseteq V - \{w\}$ of candidate departure vertices into account with availability times $t_v, v \in A$. The optimal candidate $v \in A$ is the one for which the fastest path from v to w starting at time t_v has minimal arrival time in w.

W1

1. (Initialization). Set $L = V$, $m(l) = \infty \; \forall \, l \in V - A$, and $m(v) = t_v$ and $pred(v) = v$ for all $v \in A$.
2. (Iteration). While $L \neq \emptyset$ do:
   (a) Select $i \in L$ with $i = argmin_{l \in L} m(l)$.
   (b) $L = L - \{i\}$.
   (c) $\forall \, j \in L \cap S(i)$ do:
   if $c_{ij}(m(i)) < m(j)$, then $m(j) = c_{ij}(m(i))$ and $pred(j) = i$.
3. (Termination). Output $m(w)$ and $w, pred(w)$, $pred(pred(w)), \ldots, pred(\ldots(w)\ldots) = v$.

Paths generated by Dijkstra like algorithms such as **W1** will always be specified backwards for notational ease. After having finished the last iteration of step 2, the algorithm does not yet explicitly hold the departure vertex v of the optimal path. This is found by $O(n)$ successive evaluations of the predecessor function. The optimal path may run from v through one or several other vertices of A. Therfore it is essential to start with all vertices – not all except some from A – being in the list L of tentatively labeled vertices.

Theorem 2 *Algorithm* **W1** *computes a fastest path from $v \in A$ to w starting at time t_v.*

The time complexity of **W1** is $O(n^2)$ if computing each value of $c_{ij}(\cdot)$ is organized by an array to take $O(1)$ steps.

3.2 SPECIAL GRAPHS

For linear graphs which are graphs with $O(n)$ edges the algorithm **W1** can be refined to have a less than quadratic run time. This is done by a heap containing the values $m(k)$. Since the minimum of a heap can be computed in $O(\log n)$, the n minimum computations in step 2(a) require $O(n \log n)$ computation effort over all iterations. The operations from step 2(b) require n computations and the operations of step 2(c) require $O(n)$ edge related updates for all iterations combined. Each update of the heap requiring effort $O(\log n)$ gives an overall run time of $O(n \log n)$, comp. [2, p. 1005]. The same complexity applies to computing all fastest paths that emanate from v.

4 DISTRIBUTED COMPUTATIONS

Distributed computing requires local computations as well as a communication overhead. The local computations adhere to region covers.

Definition 2 *Let $G = (V, E)$ be connected. A collection $\mathcal{C} = \{C_1, \ldots, C_\mu\}$ of subsets or classes C_i of vertices from V is called a region cover of V if it satisfies all subsequent conditions:*

1. $\bigcup_{i=1}^{\mu} C_i = V$.
2. $\bigcup_{i=1, \, i \neq k}^{\mu} C_i \neq V \;\; \forall \, k = 1, \ldots, \mu$.
3. *Each induced subgraph $G_i = (C_i, E(C_i))$ with $E(C_i) = \{(a,b) \in E | \, a, b \in C_i\}$ is connected, $i = 1, \ldots, \mu$.*
4. $\bigcup_{i=1}^{\mu} E(C_i) = E$.

Condition 1 means that $\mathcal{C}$ is a cover of V and condition 2 means that each class C_i contains a vertex which is not contained in all other classes united. The last condition implies that a region cover is a clutter, which is a system of pairwise $\subseteq$-incomparable subsets of V. Condition 4 means that a path's transition from one class to another occurs only at vertices that are common to both classes. Hence from condition 4 alone follows that a region cover can never be a partition of V except in the trivial case $\mu = 1$. Condition 3 is not necessary for the sequel but it avoids artificially complicated constellations. A non-empty intersection of two classes need not be connected.

The classes C_i of a region cover are supposed to be those sets for which fastest paths can be computed

locally. Transitions from one class to another should be sparse and usually will be for classes derived from real world problems. However, the fastest path may make $\Theta(n)$ changes from one class to another even in the case of only $\mu = 2$ classes as in the next example.

Example 1 *Let G have a cover with $\mu = 2$ as given in figure 4, where arc directions are omitted. Each traversal along a horizontal edge is long compared to each traversal along a slanted edge. The fastest path is unique with $n/3$ class changes.* ◇

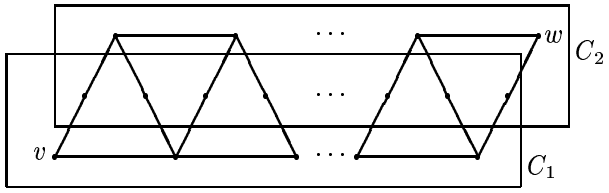

Figure 4: $\Theta(n)$ class transitions

Cardinalities of intersections between different classes of a region cover should be minimal. The reason is twofold. First, this simplifies updating and consistency routines of transition functions which are subject to change over long periods of time. Second, the computational effort is reduced with each reduction of sizes $|C_i \cap C_j|$ for classes $C_i, C_j \in \mathcal{C}$, see below.

Definition 3 *Let $\mathcal{C} = \{C_1, \ldots, C_\mu\}$ be a region cover of $G = (V, E)$.*

1. *A class C_i is called (fastest path) closed if at least one fastest path from any $v \in C_i$ to any $w \in C_i$ starting from an arbitrary moment onward consists only of vertices in C_i.*
2. *The region cover is called (fastest path) closed if all its classes are (fastest path) closed.*

Let C be a connected subset of V, $A \subseteq C$, and (t_a) denoting the vector of availability times. $W1(C, A, t)$ then denotes the algorithm **W1** computing the fastest paths from the respective optimal candidate $v \in A$ to vertices in C starting at time t_v. More precisely, $W1(C, A, (t_a)) \rightarrow (m(x))\ \forall x \in C'$, $C' \subseteq C$ with set $A \subseteq C$ denotes the execution of **W1** with given initial conditions until all $x \in C'$ have received permanent labels $m(x)$. Also, we make use of the so-called trace.

Definition 4 *A trace or $\mathcal{C}$-trace of a path is a sequence of classes containing the vertices of a path in the order of their appearance where a class is only changed if necessary.*

For example, a path $i_0, \ldots, i_k$ has trace (C_7, C_8, C_3) if $i_0, \ldots, i_{s_1} \in C_7$, $i_{s_1+1}, \ldots, i_{s_2} \in C_8 - C_7$, and $i_{s_2+1}, \ldots, i_k \in C_3 - C_8$. A trace of a path is generally not unique, but it is in a two region cover due to minimality of description length. Computing the trace of a fastest or reasonably fast path based on aggregated information appears to be the core problem of distributed path computations.

In the sequel, a fastest path will be concatenated from partial paths stemming from different computations steps (over different classes).

4.1 FIXED TRACES

The fastest path from v to w starting from time t_1 onward under the constraint of using a prespecified trace $(C_1, \ldots, C_\nu)$, $v \in C_1, w \in C_\nu$, can be computed by the following algorithm. The trace may contain repetitions accounting for the case of region covers being not fastest path closed.

W2

1. $W1(C_1, \{v\}, t_1) \rightarrow m_0(x)$ for all $x \in C_1 \cap C_2$.
2. For $i = 2, \ldots, \nu - 1$ do:
   $W1(C_i, C_{i-1} \cap C_i, (m_{i-2}(x))) \rightarrow m_{i-1}(x)$, for all $x \in C_i \cap C_{i+1}$.
3. $W1(C_\nu, C_{\nu-1} \cap C_\nu, (m_{\nu-2}(x))) \rightarrow m_{\nu-1}(w)$ and output $m_{\nu-1}(w)$ and $w, pred_{\nu-1}(w), \ldots, x_{\nu-1}, pred_{\nu-2}(x_{\nu-1}), \ldots, x_1, pred_0(x_1), \ldots, v$.

Algorithm **W2** runs in $O(\nu \cdot \max\{|C_1|^2, \ldots, |C_\nu|^2\})$.

Accounting for the fact that a subtrace of a trace could result in an earlier overall arrival time, algorithm **W2** can be modified, so that already found optimal labels will not be overwritten by subsequent computations.

4.2 COMPUTING TRACES FROM AN AGGREGATED VIEWPOINT

The intersection graph $G_{\mathcal{IC}}$ of all non-void intersections of regions is introduced to enable the computation of traces which ideally coincide with traces of fastest paths. The intersection graph allows to calculate candidate traces for fastest paths on the basis of estimates for travel times.

Definition 5 *The intersection graph $G_{\mathcal{IC}} := (V_{\mathcal{IC}}, E_{\mathcal{IC}})$ is defined by having the vertex set $V_{\mathcal{IC}} := \{v_{C_i \cap C_j} | \ C_i, C_j \in \mathcal{C}$ with $C_i \cap C_j \neq \emptyset\}$ and the edge set $E_{\mathcal{IC}} := \{(v_{C_i \cap C_j}, v_{C_k \cap C_l}) | \ \{C_i, C_j\} \cap \{C_k, C_l\} \neq \emptyset$ and $\{C_i, C_j\} \neq \{C_k, C_l\}\}$.*

The intersections of distinct classes receive different vertices even if the intersections should be equal. The edge set specification contains the condition "$\{C_i, C_j\} \neq \{C_k, C_l\}$" ensuring that the intersection graph contains no loops. Each class of $\mathcal{C}$ is represented in $G_{\mathcal{IC}}$ by the clique of all its intersections with other classes.

$G_{\mathcal{IC}}$ receives constant edge labels by assigning edge $(v_{C_i \cap C_j}, v_{C_k \cap C_l})$ the value $\min_{v_0 \in C_i \cap C_j, w_0 \in C_k \cap C_l} \lambda(p^0_G(v_0, w_0))$. Thereby labels within G are given for each edge (i, j) by $\min_{1 \leq k \leq N_i} d^k$, where d^k are the transition times belonging to $c_{ij}(\cdot)$; comp. section 2.1.

A candidate trace for a fastest path from $v \in C_1$ to $w \in C_\nu$ can be computed as a shortest path by the following procedure which extends the intersection graph by making connections to v and w, comp. example 2.

T1

1. Vertices $v \in C_1$ and $w \in C_\nu$ are inserted into $V_{\mathcal{IC}}$, all edges $(v, v_{C_1 \cap C_i})$ with $C_1 \cap C_i \neq \emptyset$ and all edges $(v_{C_i \cap C_\nu}, w)$ with $C_i \cap C_\nu \neq \emptyset$ are inserted into $E_{\mathcal{IC}}$ resulting in the extended intersection graph $G^{(v,w)}_{\mathcal{IC}}$. The extra edges are labeled in the same way as previous edges in $E_{\mathcal{IC}}$.
2. Computation of a shortest path $p^0_{G^{(v,w)}_{\mathcal{IC}}}(v, w) = (v, C_{i_1} \cap C_{i_2}, \ldots, C_{i_{k-1}} \cap C_{i_k}, w)$ with indexing chosen so that k is even.
3. Induction of trace
   (a) (Initialization). $c = (C_1)$ and $last(c) = 1$.
   (b) For $j = 4, 6, \ldots, k$ do
   If $C_{i_{j-1}} \cap C_{i_j} \not\subseteq C_{last(c)}$ then update $c = (c, C_s)$ with $\{C_s\} = \{C_{i_{j-1}}, C_{i_j}\} \cap \{C_{i_{j-3}}, C_{i_{j-2}}\}$ and $last(c) = s$.
   (c) If $w \notin C_{last(c)}$ then $c = (c, C_\nu)$ and $last(c) = \nu$.
   (d) Output of induced trace $ind(p^0_{G^{(v,w)}_{\mathcal{IC}}}(v, w)) = c$.

Traces generated by **T1** may repeatedly visit classes of the region cover $\mathcal{C}$. The induced traces serve as candidate traces for a fastest path from v to w when fed to **W2**. An induced trace from a shortest path in the extended intersection graph need not be the trace of a fastest path.

Even a nonplanar graph G may lead to a planar modified intersection graph as in the following example which also illustrates the intersection graph and the extended intersection graph.

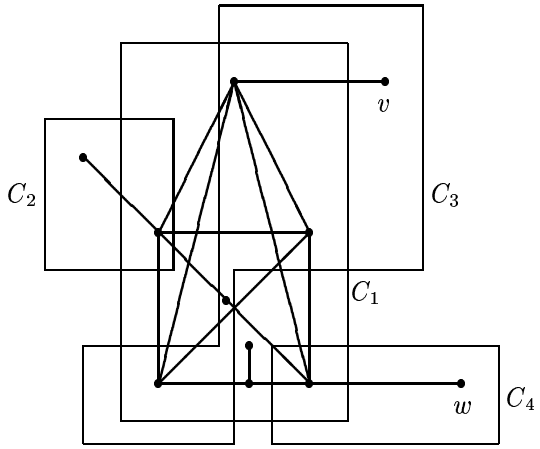

Figure 5: Nonplanar graph with a two region cover

Example 2 *Let a nonplanar graph G be given with $n = 11$ vertices by figure 5 together with a region cover with $\mu = 4$ classes. Edge directions and labels are omitted in all cases. Both the (planar) intersection graph $G_{\mathcal{IC}}$ and the extended intersection graph $G^{(v,w)}_{\mathcal{IC}}$ are given in figure 6. The intersection graph contains for example the edge between $C_1 \cap C_2$ and $C_1 \cap C_4$ because $\{C_1, C_2\} \cap \{C_1, C_4\} = \{C_1\} \neq \emptyset$ and $\{C_1, C_2\} \neq \{C_1, C_4\}$, comp. the definition of $G_{\mathcal{IC}}$.* $\diamond$

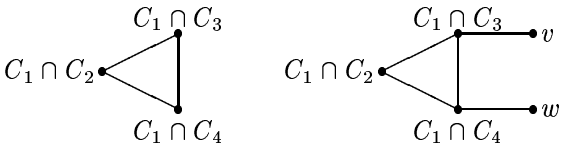

Figure 6: Intersection graph $G_{\mathcal{IC}}$ and extended intersection graph $G^{(v,w)}_{\mathcal{IC}}$

4.3 RELATIONS OF THE ORIGINAL GRAPH TO AGGREGATED GRAPHS

4.3.1 SHORTEST PATHS

Lemma 1 *Let G have region cover $\mathcal{C}$. $\lambda(p_{G^{(v,w)}_{\mathcal{IC}}}(v, w)) \leq \lambda(p_G(v, w))\ \forall\, v, w \in V$, where $p_G(v, w)$ has the same intersection sequence as $p_{G^{(v,w)}_{\mathcal{IC}}}(v, w)$.*

Particular interest deserves the case of the inequality of lemma 1 becoming an equality.

Lemma 2 *All intersections of regions are assumed to be singletons or having transition cost 0 from each vertex to each other within the same intersection. Then $\lambda(p_{G^{(v,w)}_{\mathcal{IC}}}(v, w)) = \lambda(p_G(v, w))\ \forall\, v, w \in V$, where $p_G(v, w)$ has the same intersection sequence as $p_{G^{(v,w)}_{\mathcal{IC}}}(v, w)$.*

The sequence of classes traveled along by a path $p_{G_{\mathcal{IC}}^{(v,w)}}(v,w)$ is called induced trace and denoted by $ind(p_{G_{\mathcal{IC}}^{(v,w)}}(v,w))$. The trace $ind(p^0_{G_{\mathcal{IC}}^{(v,w)}}(v,w))$ need not imply a shortest path $p^0_G(v,w)$ by simply using this trace even in case $\mathcal{C}$ is shortest path closed. Shortest path closedness for graphs with time independent edge labels is the analog to fastest path closedness. The reason for the intended implication to fail is that transition costs within class intersections may sum up to amounts which destroy the order of lengths of paths in $G_{\mathcal{IC}}^{(v,w)}$. However, this order will be maintained to a sufficient degree under an additional "discrimination" or "filtering" condition. This condition will be provided by k-shortest paths, where the 1-shortest path is the usual shortest path and a k-shortest path for $k \geq 2$ is a shortest path among all paths longer than a $k-1$-shortest path. The length of a k-shortest path from v to w in $G_{\mathcal{IC}}^{(v,w)}$ is denoted by $\lambda(p_{G_{\mathcal{IC}}^{(v,w)}}(v,w);k)$.

Theorem 3 *("Filtering for shortest paths in intersection graphs")*
Let $p_{G,0}(v,w)$ be a shortest path among all those having the same intersection sequence like one of the $k-1$-shortest or shorter paths $p_{G_{\mathcal{IC}}^{(v,w)}}(v,w)$ and let $\lambda(p_{G,0}(v,w)) < \lambda(p_{G_{\mathcal{IC}}^{(v,w)}}(v,w);k)$ for some $k \geq 2$.

1. *No path having an intersection sequence like one of the k-shortest or longer paths from v to w in $G_{\mathcal{IC}}^{(v,w)}$ is shorter than $p_{G,0}(v,w)$.*
2. *$\mathcal{C}$ being shortest path closed implies $p_{G,0}(v,w) = p^0_G(v,w)$.*

4.3.2 FASTEST PATHS

Edge labels of the intersection graph which are lower bounds of the transition time of the original graph can be improved by a refined consideration of time dependency. Therefore, an edge $(v_{C_i \cap C_j}, v_{C_i \cap C_k}) \in E_{\mathcal{IC}}$ receives the label $\min_{t \in T} \min_{v_0 \in C_i \cap C_j, w_0 \in C_i \cap C_k} \vartheta(p^0_G(v_0,w_0;t))$. Then $\vartheta(p^0_G(v_0,w_0;t)) \geq \lambda(p^0_G(v_0,w_0))$, see section 2.1. The filtering for shortest paths then extends to fastest paths.

Theorem 4 *("Filtering for fastest paths in intersection graphs")*
Let $p_{G,0}(v,w;t)$ be a fastest path among all those having the same intersection sequence like one of the $k-1$-shortest or shorter paths $p_{G_{\mathcal{IC}}^{(v,w)}}(v,w)$ and let $\vartheta(p_{G,0}(v,w;t)) < \lambda(p_{G_{\mathcal{IC}}^{(v,w)}}(v,w);k)$ for some $k \geq 2$.

1. *No path having an intersection sequence like one of the k-shortest or longer paths from v to w in $G_{\mathcal{IC}}^{(v,w)}$ is faster than $p_{G,0}(v,w;t)$.*
2. *$\mathcal{C}$ being fastest path closed implies $p_{G,0}(v,w;t) = p^0_G(v,w;t)$.*

5 CONCLUSIONS

The core buisness of computing fastest paths in a distributed manner is computation of traces. An approach for trace computation has been shown based upon shortest path computation on the aggregated structure.

Fastest paths can be used for distributed computation of passenger travel connections even in case of other objectives, serving as lower bounds for transition times.

REFERENCES

[1] Ahuja, R., Magnanti, T.L., Orlin, J.B., *Network flows*, Prentice Hall, Englewood Cliffs, 1993.

[2] Frederickson, G.N., Fast algorithms for shortest paths in planar graphs, with applications, *SIAM Journal on Computing 6*, 1987, p. 1004-1022.

[3] Kämpke, T., A separation decomposition for orders, *Networks 24*, 1994, p. 185-194.

[4] Lawler, E., *Combinatorial optimization networks and matroids*, Holt, Rinehart and Winston, New York, 1976.

[5] Martins, E.Q.V. et al., An algorithm for the ranking of shortest paths, *European Journal of Operational Research 69*, 1993, p. 97-106.

[6] Martins, E.Q.V., Santos, J.L.E., A new shortest paths ranking algorithm, *manuscript*, 1996.

[7] Möhring, R.H., Radermacher, F.J., Substitution decomposition for discrete structures and connections with combinatorial optimization, *Annals of Discrete Mathematics 19*, 1984, p. 257-356.

[8] Thulasiraman, K., Swamy, M.N.S., *Graphs: theory and algorithms*, Wiley, New York, 1992.