MUSE: une planification d’itinéraires inspirée de Séparateurs Multimodaux

Amine M. Falek\textsuperscript{1,2}, Cristel Pelsser\textsuperscript{2}, Sebastien Julien\textsuperscript{1}, Fabrice Theoleyre\textsuperscript{2}

\textsuperscript{1}Technology & Strategy group, 4 rue de Dublin, 67300 Schiltigheim, France.
\textsuperscript{2}ICube Lab, CNRS / University of Strasbourg, Pole API, Boulevard Sebastien Brant, 67412 Illkirch Cedex, France.

Le domaine des algorithmes de calcul de plus courts chemins connaît un essor important avec le développement du cloud. Quelques solutions, dites multimodales, sont conçues pour combiner divers modes de transports, mais au prix d’une augmentation significative de la complexité. Nous proposons ici MUSE, un algorithme basé sur les séparateurs de graphes, mais adapté au cas multimodal. Dans une phase de prêtraitement, nous découpons tout d’abord le graphe en partitions indépendantes (ou cellules), chacune découpée en modes de transport afin de pouvoir plus tard répondre à n’importe quelle requête. Ensuite, nous précalculons toutes les plus courtes routes, sur ce petit nombre de cellules, en tenant compte des labels (modes) de chaque arête. Nous pouvons ainsi répondre à une requête très rapidement dans la phase online : l’utilisateur spécifie les séquences de mode qu’il autorise, et exploite les plus courtes routes pré-calculées.

Mots-clés : multimodal algorithm; route planning; graph partitioning; cells;

1 Introduction

Transportation is an intricate network deeply rooted in our society. Thereby, efficient route planning is a pressing necessity to accommodate modern travelers. As a research topic, route planning has tremendously evolved and branched into a myriad of sub-problems. In its purest form, the goal is to identify a shortest route to reach a given destination in the network. Graph separators significantly accelerate queries when applied to road networks [DGPW11]. With independent partitions, the pre-processing becomes parallelizable, and only partitions that have been affected by traffic congestion or delays are ought to be updated.

The multimodal routing problem [ZA08] consists of computing a shortest route constrained by a sequence of transportation modes. Unfortunately, most of the proposed solutions are tailored for either road networks or public transit networks in isolation. Thereby, to address current limitations, we tackle here the multimodal routing problem using a different approach based on graph separators.

2 Modeling Multimodal Networks

Transportation networks are usually modeled with graphs for their intuitiveness and the extensive algorithmic toolbox of graph theory. We use a directed Graph model $(V,E)$ that consists of a set of vertices $v \in V$, and a set of directed edges $(v,w) \in E$ connecting vertices $v,w \in V$. The Edge Cost $c(v,w,\tau)$ represents the required travel-time to reach vertex $w$ when departing from vertex $v$ at time $\tau$. The cost is given by a periodic positive piece-wise linear function $f : \Pi \rightarrow \mathbb{R}^+$ where $\Pi = [0,p] \subset \mathbb{R}$ with a period $p \in \mathbb{N}$. A Path $P = \{v_0,v_1,\ldots,v_k\}$, also written $P_{v_0v_k}$, is an ordered sequence of vertices $v_i \in V$.

Road Network: for each segment, we collect a set of speed values over a time window $\Pi$ sampled at a fine-grained rate $\Delta t$, and we construct its speed profile as a piece-wise linear function $f_{vw}$.

Foot Network: is a time-independent graph $G(V,E)$, with footpaths including sidewalks and stairs.

Bicycle Network: all cycling lanes in addition to rental stations. We insert a vertex $v \in V_{rent} \subseteq V$ for each rental station and an edge $(v,w)$ between the station and its closest junction in the bike network.
Transit Network: is based on a timetable $T = (\mathcal{Z}, S, C)$ which consists of a set of shuttle vehicles $\mathcal{Z}$, a set of stations $S$, and a set of elementary connections $\mathcal{C}$.

The multimodal network combines all of the road, foot, bike, and public transit networks within a single data structure: a labeled directed graph $G^\Sigma(V, E, \Sigma)$. To distinguish each network, we attach a unique label $\sigma \in \Sigma = \{c, f, b, p, t\}$ to each edge, where $c$, $f$, $b$, and $p$ stand for car, foot, bike, and public respectively. Link edges labeled $t$, are used to transit from the foot network to all other networks. We rely on edge labels to constrain a shortest path by a sequence of acceptable modes.

3 MUSE: Multimodal Separators with Label Constraints

MUSE is a speedup technique to Dijkstra’s algorithm for multimodal route planning inspired by graph separators and label constraints. The user provides a set of transportation modes (e.g., private car/bike, public transit), and MUSE computes a shortest path, restricted to the authorized modes.

The algorithm consists of a preprocessing and a query phase. During the first stage of preprocessing, we execute a graph partitioning algorithm to split the multimodal graph into $k$ balanced cells $\{C_0, C_1, \ldots, C_k\}$. Partitioning is run only once, as it solely depends on the topology of the graph. The second stage of the preprocessing consists of computing an overlay graph $H$: for each cell $C_i$ in the partition, we compute a clique on its boundary vertices, while taking care of the labels (modes of transport) of each edge. We achieve this by running a label constrained Dijkstra $D_{Reg\mathcal{I}C}$ [BBH+08] from each boundary vertex.

3.1 Preprocessing Phase

Partitioning: Planar graphs can be partitioned in linear time with small separators [Dji82]. The goal is to split the graph into $k$ cells $\{C_i\}_{i \in \{1, \ldots, k\}}$ such that the number of cut-edges linking the boundary vertices of different cells is minimum. Formally, a cut-edge is an edge $(v, w)$ with $v \in C_i$ and $w \in C_j$ if $i \neq j$. Road networks, although not planar (due to overpasses and tunnels), can also be efficiently partitioned.

Let us consider a shortest path $P$ and a subpath $P_i = \{v, \ldots, w\} \subset P$ enclosed by the cell $C_i$ (accessing $C_i$ through vertex $v$ and leaving it through vertex $w$). Using graph partitioning, we can precompute, inside each cell, all the shortest paths between all pairs of its boundary vertices. We propose to apply the same approach to a multimodal network. Computing ideal partitions is NP-hard; thus we use the heuristic METIS [KK98] (Multilevel Graph Partitioning algorithm) adapted to multimodal graphs.

- **Coarsening:** by repeatedly contracting neighboring vertices in $G_{i-1}(V_{i-1}, E_{i-1})$ we obtain a graph $G_i(V_i, E_i)$ where $|V_i| < |V_{i-1}|$. At each iteration, we compute the maximal matching $M$ and contract in the same cell each pair of vertices $v, w|(v, w) \in M$. Since $k$ is the number of desired cells in the partition, $|V_i| \geq k$.

- **Partitioning:** we partition the coarsest graph $G_c(V_c, E_c)$ using Breadth-First-Search (BFS) starting from a random vertex $v \in V_c$ and growing a tree $T \subset V_c$ until $|T| \approx 1/2|V_c|$. To obtain $k$ partitions, the initial partitions are then recursively partitioned $\log_2(k)$ times.
We construct a multimodal graph with 1,444,634 vertices and 4,630,315 edges for the Ile-de-France region. The dataset was obtained with OpenstreetMap [MM+17] to model the road, cycling and pedestrian networks. The public transit network was built from GTFS timetables combining train, RER, subway, tramways, and bus transportation from the Ile-de-France mobilités open dataset. We evaluate the algorithm on four Non-deterministic Finite Automata (NFA): 1) Foot-Transit consists of two states combining walking and public transportation; 2) Car-Foot: private car is used initially, followed by walking; 3) Foot-Transit + rental Bicycle: similar to the Foot-Transit NFA with the addition of rental bicycles for faster transfers in the city center. 4) Bicycle-Foot-Transit-Car combines all modes. The private bike is used initially, then followed by any combination involving walking, public transportation, or taxi and uber services.

The experiments were run on an Intel Cascade Lake CPU with 24 cores and 128GB of memory, using Java JRE 1.8. Figure 2 reports the preprocessing time for each NFA and partition size, parallelized on the 24 cores. Preprocessing requires less time with a larger number of partitions: we have smaller cells, and fewer border vertices; thus, the algorithm computes the label constrained cliques faster. Total preprocessing time is reduced from 17 minutes (10 cells) to less than 2 minutes for partitions with 100 cells or more. The NFA impacts only slightly the preprocessing time: the partitions are the same (with a different number of modes), but the number of border vertices remains comparable. We ran 1000 queries for each partition size.
5 Conclusion and Future Work

We presented a multimodal shortest path algorithm based on graph separators. Using partitions, we can parallelize preprocessing to accommodate large graphs, and pre-compute a set of shortest paths, through each possible pair of cells. That way, we can retrieve very fast a shortest path during the query phase, for any sequence of modes. We are currently running further experiments involving much larger graphs (France and Europe) as well as exploring multi-level partitions combined with a bidirectional search during query-time for even better speedups. Moreover, we are investigating how to exploit this multimodal partition based solution with dynamic graphs, where only a subset of the modes have time-dependent travel times. The objective consists in re-running only one part of the preprocessing, to make the solution scalable.

References


