MUSE: une planification d'itinéraires inspirée de Séparateurs Multimodaux

Amine M. Falek[1,2], Cristel Pelsser[2], Sebastien Julien[1], Fabrice Theoleyre[2]

[1]*Technology & Strategy group, 4 rue de Dublin, 67300 Schiltigheim, France.*
[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 connait 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 prétraitement, 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-clefs : 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.

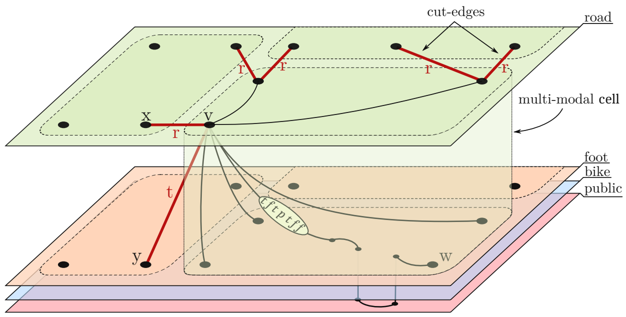
2 Modeling Multimodal Networks

Transportation networks are usually modeled with graphs for their intuitiveness and the extensive algorithmic toolbox of graph theory. We use a directed Graph model $G(V,E)$ 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 τ. The cost is given by a periodic positive piece-wise linear function $f : \Pi \to \mathbb{R}^+$ where $\Pi = [0,p] \subset \mathbb{R}$ with a period $p \in \mathbb{N}$. A Path $P = \{v_0, v_1, .., v_k\}$, also written $P_{v_0 v_k}$, is an ordered sequence of vertices $v_i \in V$.

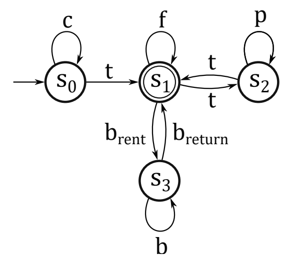
Road Network: for each segment, we collect a set of speed values over a time window Π sampled at a fine-grained rate Δt, and we construct its speed profile as a piece-wise linear function f_{vw}.

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 $v \in V^{\text{rent}} \subseteq V$ for each rental station and an edge (v,w) between the station and its closest junction in the bike network.



(a) Cut-edges are shown in bold labeled segments



(b) NFA combining car, foot, public transit, and rental bicycles.

Figure 1: Multimodal graph partitioning and NFA

Transit Network: is based on a timetable $\mathcal{T} = (\mathcal{Z}, \mathcal{S}, \mathcal{C})$ which consists of a set of shuttle vehicles $\mathcal{Z}$, a set of stations $\mathcal{S}$, 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 $\{C_0, C_1, .., C_k\}$. 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_{RegLC} [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 $\{C_i\}_{i\in[1,k]}$ 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 $w \in C_j | i \neq j$. 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 $P_i = \{v, .., w\} \subset P$ 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 $|T| \simeq 1/2|V_c|$. To obtain k partitions, the initial partitions are then recursively partitioned $log_2(k)$ times.

- **Uncoarsening and Refining:** at each iteration i, a less coarse graph G_i is obtained by expanding G_{i-1} (the pair of vertices that have been merged at the i^{th} step with coarsening, are here split again).

We have then to process each cell C_i to compute the set of all shortest paths traversing C_i. We have to consider all the paths P_{vw}, where $v,w \in V_i^b \subset C_i$ are the boundary vertices. This problem is tractable since the number of boundary vertices is practically small in a cell (by construction). We construct a *clique* $H_i(V_i^b, E_i)$ for each cell C_i by creating a *virtual* edge $(v,w) \in E_i \mid v,w \in V_i^b$ whose cost $c(v,w) = c(P_{vw})$. It is worth noting that we can easily reconstruct a shortest path during the query phase since we have its cost.

In essence, we construct the complete graph induced by boundary vertices whose edge costs correspond to the shortest paths. Combining all cliques, we build an *overlay* $H(V,E)$ whose vertex-set $V = \cup_{i=1}^k V_i^b$ contains all boundary vertices, and edge-set $E = E_c \cup_{i=1}^k E_i$ consists of all edge-cuts and clique edges.

Label Constrained Cliques: For a multimodal network, however, we must ensure that the cliques implement modes that are allowed by the user. Figure 1a illustrates a multimodal partition where each cell C_i spans across all layers of uni-modal networks with, possibly, boundary vertices at each level. Since a user can specify the accepted sequence of modes, we must solve the Label Constrained Shortest Path Problem (LCSPP). An edge may or may not be considered, depending on where the user comes from: we must verify the acceptability of the sequence of modes.

We model the user's constraint with a regular expression, accepted by a Non-deterministic Finite State Automata (NFA). Conveniently, an NFA denoted $\mathcal{A}$, can be implemented as a directed labeled graph $G^{\mathcal{A}}$. Thus, we can solve the LCSPP with a Dijkstra variant D_{RegLC}, deployed on the *product graph* $G^{\times} = G^{\Sigma} \oplus G^{\mathcal{A}}$ merging the underlying graph G^{Σ} and the automaton graph $G^{\mathcal{A}}$. A vertex $\langle v,s \rangle \in V^{\times}$ is a combination of a vertex $v \in V$ and a state $s \in S$. An edge $(\langle v,s_i \rangle, \langle w,s_j \rangle) \in E^{\times}$ is added *iff* there exists an edge $(v,w) \in E$ such that $s_i \times label(v,w) \rightarrow s_j$ is a valid transition of $\delta \in \mathcal{A}$. The space complexity of D_{RegLC} is $O(V \times S)$ but we implement an improved version reducing the complexity to $O(V+S)$.

Furthermore, to avoid computing large cliques, we construct the NFA such that each state corresponds to a unique transportation layer in the multimodal graph (Figure 1b). Thereby, each boundary vertex is potentially combined to only a subset of states sharing the same label.

3.2 Query

During the query phase, we run D_{RegLC} on the graph $G^q = G_r \cup H \cup G_t$, where H is the overlay graph, and C_r and C_t are the root and target cells respectively (*i.e.,* the cells that contain the point of departure and arrival). Therefore, we achieve a significant speedup by skipping most of the underlying graph G^{Σ} as most of the computational effort is spent exploring G_r and G_t.

4 Experimental Evaluation

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

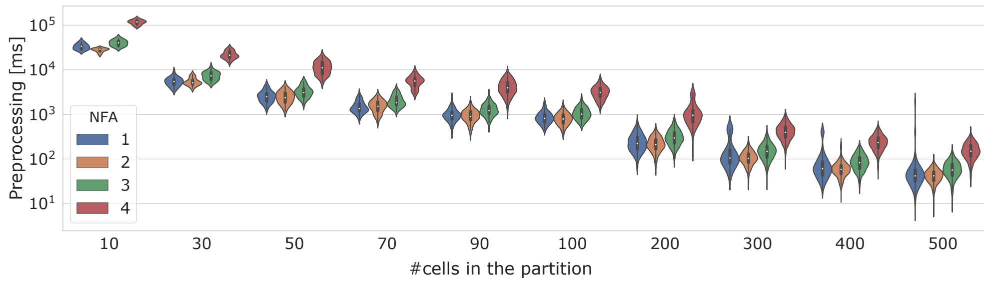


Figure 2: Preprocessing time to compute the constrained cliques for different graph partitions and NFA.

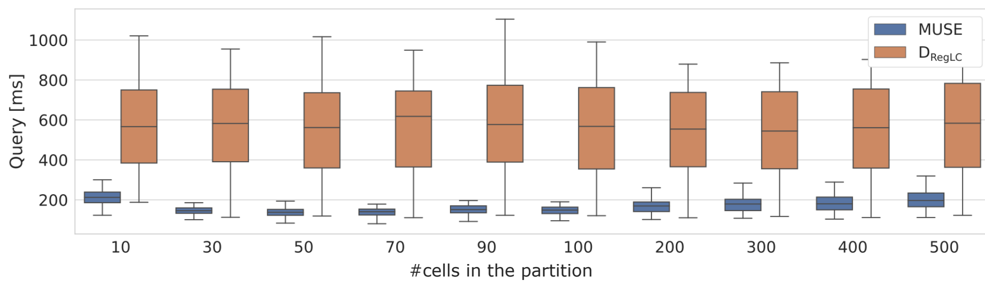


Figure 3: Query time for 1000 queries using D_{RegLC} and MUSE constrained by the NFA in figure 1b.

using both MUSE and D_{RegLC} (constrained by the NFA in figure 1b). As shown in figure 3, query-time depends on the partition size, mainly, we achieve a significant speedup over D_{RegLC} with a partition of 100 cells which offers an excellent computational trade-off between preprocessing and query.

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

[BBH+08] Chris Barrett, Keith Bisset, Martin Holzer, Goran Konjevod, Madhav Marathe, and Dorothea Wagner. Engineering label-constrained shortest-path algorithms. In *Algorithmic Aspects in Information and Management*, pages 27–37, 2008.

[DGPW11] Daniel Delling, Andrew V Goldberg, Thomas Pajor, and Renato F Werneck. Customizable route planning. In *International Symposium on Experimental Algorithms*, pages 376–387. Springer, 2011.

[Dji82] Hristo Nicolov Djidjev. On the problem of partitioning planar graphs. *SIAM Journal on Algebraic Discrete Methods*, 3(2):229–240, 1982.

[KK98] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392, 1998.

[MM+17] Peter Mooney, Marco Minghini, et al. A review of openstreetmap data. In *Mapping and the Citizen Sensor*. Ubiquity Press, 2017.

[ZA08] K. G. Zografos and K. N. Androutsopoulos. Algorithms for itinerary planning in multimodal transportation networks. *IEEE Transactions on Intelligent Transportation Systems*, 9(1):175–184, 2008.