



GraphServer, pour les graphes de réseau

Vincent Picavet, Oslandia

En quelques mots...



Serveur de graphes

Multimodal

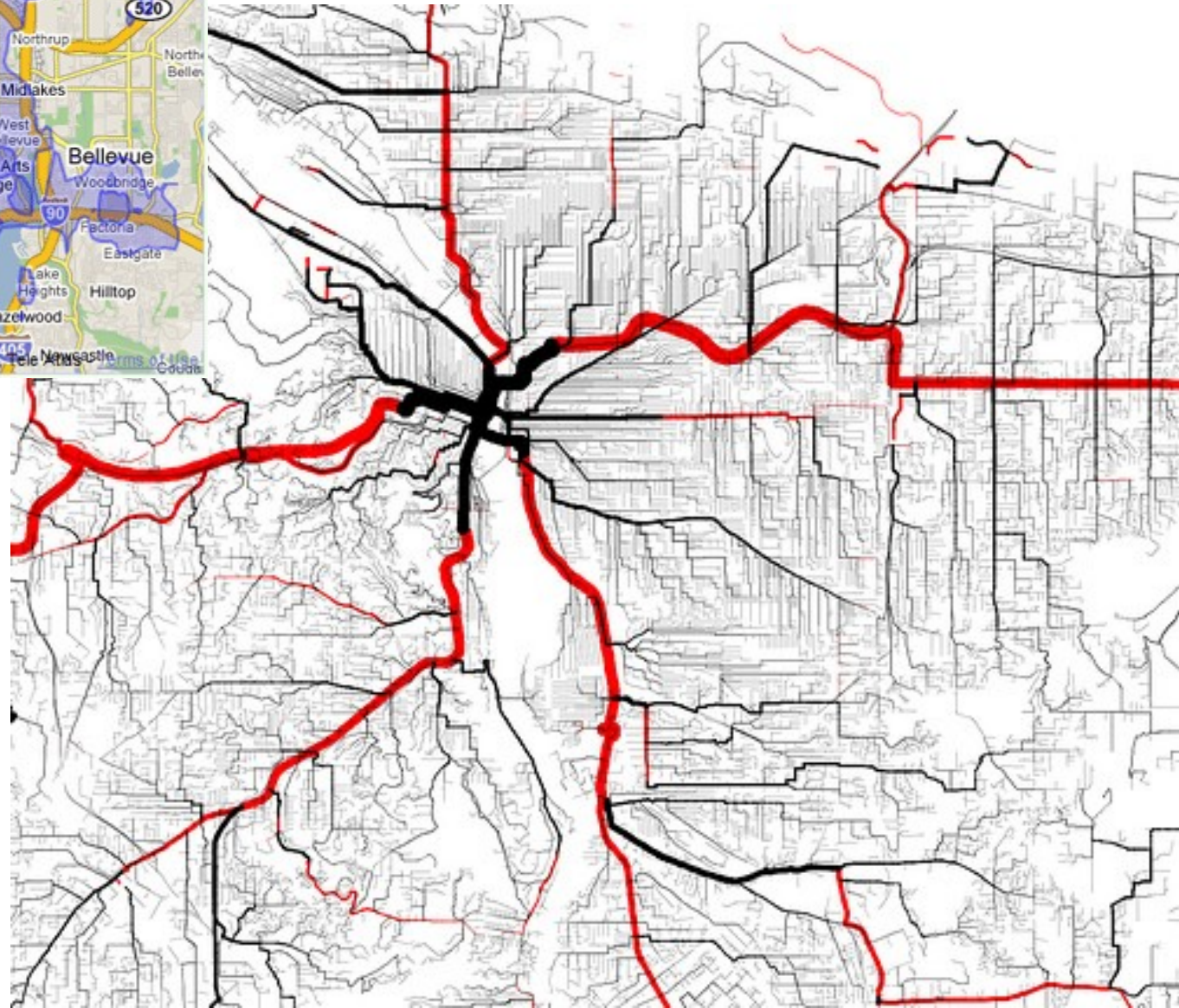
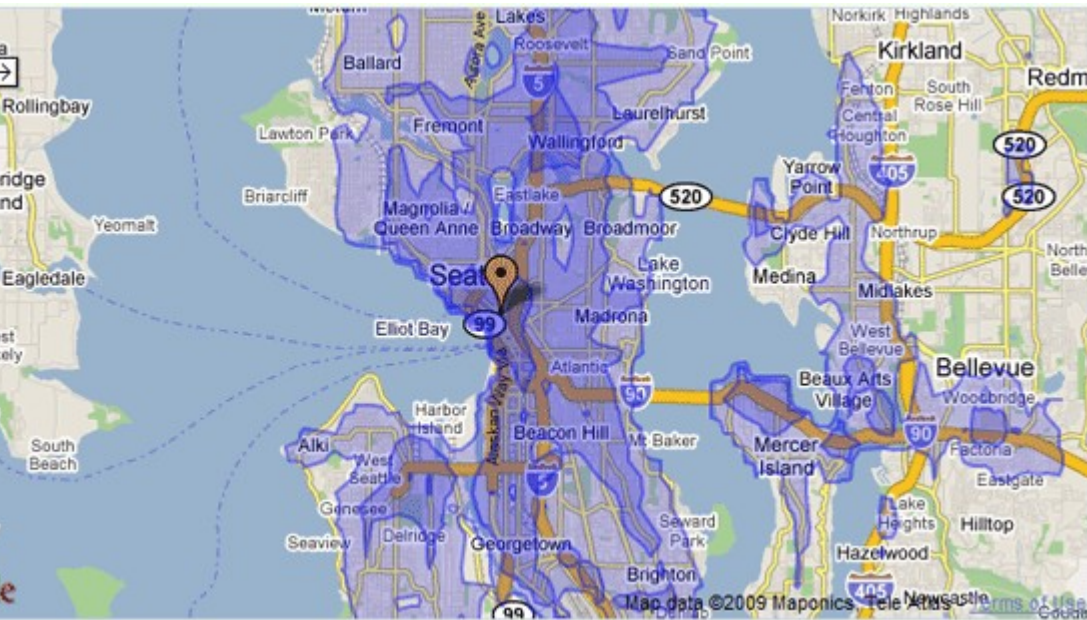
Interface web

Framework - Boite à outils

OpenSource



Routing et isochrones



Historique

2005 : planificateur de trajets vélo

2005 : Présentation à Trimet

2007 : réimplémentation en C

2007 : Présentation à Where 2.0

2008 : Adoption par Urban Mapping Inc.

2009 : «vendu» a MapQuest

2009 : Where 2.0

2009 : Utilisation comme planificateur de trajets vélo



Caractéristiques

Une bibliothèque C

Un module Python

Un module Ruby (non maintenu)

Une boîte à outils

OSMDB : OpenStreetMap vers base de données GraphServer

GTFSDB : GTFS vers base

Un système de planification de trajets multimodal

Accepte les requêtes HTTP

Retourne des trajets

Adapté à d'autres problèmes de graphe que le routier



Planificateur de trajets

Dijkstra

Multimodal

Données OSM

Terrain (DEM)

Trafic avec horaires (GTFS)

Aérien

Bus, routier, marche, vélo...

Extensible

Adaptable

Tout est paramétrable

Rapide

Implémentation en C



Points faibles

Peu de support

Peu répandu

Manque de test sur les gros ensembles de données

Taille d'un pays, Europe...

Consomme beaucoup de ressources (RAM, CPU)

Difficulté de modéliser sous forme de graphe



Projets liés



See where it takes you.

SITI : Sistema de Informacion de transporte Intermodal

Ministère des transports espagnol

Université de Valence

MyTTC : Toronto

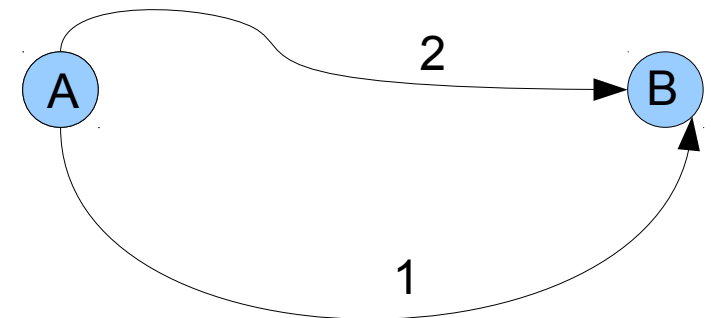
OpenTripPlanner



Exemple d'utilisation du module Python

- API de graphe
 - Bindings Python de l'API C
 - Permet la modification dynamique du graphe

```
$ python
>>> from graphserver.core import Graph, Street, State
>>> gg = Graph()
>>> gg.add_vertex("A")
>>> gg.add_vertex("B")
>>> gg.add_edge( "A", "B", Street("1", 100) )
>>> gg.add_edge( "A", "B", Street("2", 50 ) )
>>> gg.size    # un graphe très basique !
2
>>> gg.get_vertex("A").outgoing    # il est bidirectionnel
[<graphserver.core.Edge object at 0xb7b9accc>, <graphserver.core.Edge object at 0xb7b9acac>]
>>> gg.get_vertex("A").incoming
[]
```



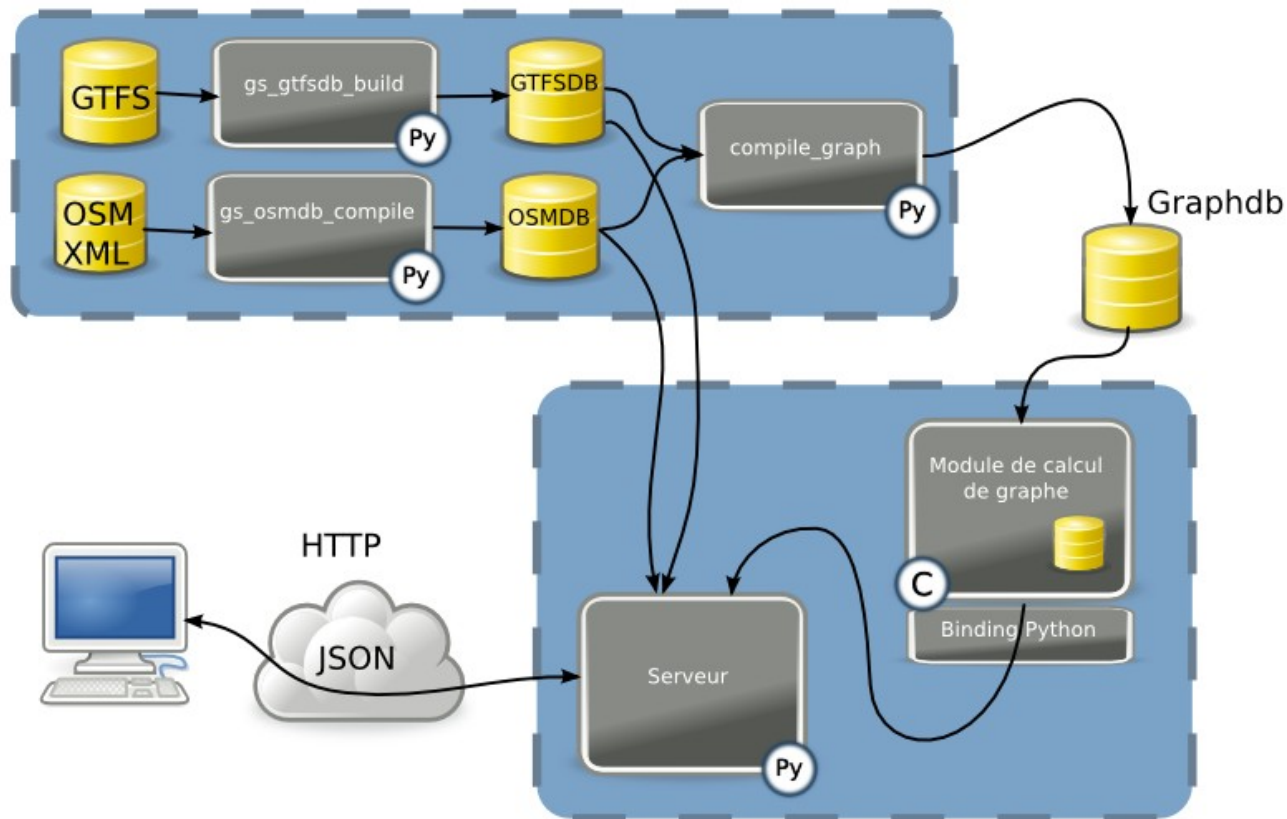
Exemple d'utilisation du module Python (suite)

- L'API expose les fonctions de calcul de plus court chemin

```
>>> spt = gg.shortest_path_tree( "A", "B", State(1,0) )
>>> spt
<graphserver.core.ShortestPathTree object at 0xb7c45b8c>
>>> spt.get_vertex("A") # l'arbre de chemin les plus courts est une copie partielle du graphe
<graphserver.core.Vertex object at 0xb7c4a92c>
>>> spt.get_vertex("A").outgoing # qui contient tous les chemins les plus courts vers les points plus
# proches que 'B'

[<graphserver.core.Edge object at 0xb7b9adec>]
>>> vertices, edges = spt.path("B") # on récupère le plus court chemin de A à B a partir de l'arbre
>>> vertices # avec tous les noeuds
[<graphserver.core.Vertex object at 0xb7b9ae4c>, <graphserver.core.Vertex object at 0xb7b9adec>]
>>> edges # et les arcs
[<graphserver.core.Edge object at 0xb7b9accc>]
>>> edges[0].payload # sur les arcs on récupère les informations utiles (nom, longueur, etc)
<Street name='2' length=50.000000 rise=0.000000 fall=0.000000 way=0>
>>> vertices[-1].payload # sur les noeuds aussi on a des informations, notamment sur le dernier
<graphserver.core.State object at 0xb7c4a92c>
>>> vertices[-1].payload.time # il faut 58 secondes pour aller de "A" à "B"
58
>>> spt.destroy() # Nettoyage des données
>>> gg.destroy()
>>> exit()
```

Exemple avec des données réelles



Récupération des données OSM :

```
$ wget http://api.openstreetmap.org/api/0.6/map?bbox=-122.4624,40.5505,-122.2876,40.6334 -O map.osm
```

Conversion de format et création de la base du graphe :

```
$ gs_osmdb_compile map.osm map.osmdb
```

```
$ gs_import_osm map.gdb map.osmdb
```

Récupération de données de transport :

```
$ wget http://trilliumtransit.com/transit_feeds/redding/google_transit.zip -O redding_gtfs.zip
```

Conversions de format et incorporation de la base dans le graphe :

```
$ gs_gtfsdb_build redding_gtfs.zip redding.gtfsdb
```

```
$ gs_import_gtfs map.gdb redding.gtfsdb
```

Création des liens entre les deux réseaux :

```
$ gs_link_osm_gtfs map.gdb map.osmdb redding.gtfsdb
```



Récupération des itinéraires

- Par l'API Python
- Par le serveur HTTP :

[http://mygraphserver/path?origin="osm-92011649"&dest="sta-3803"&curtime=1260839444](http://mygraphserver/path?origin=)

```
{
  "performance": {"cleanup_time": 0.00016093254089355469,
    "narrative_postprocess_time": 0.073606014251708984,
    "path_query_time": 0.00079703330993652344},
  "narrative": [
    ["BoardEvent",
      {"when": "2009-12-14 17:52:00-08:00",
        "what": "Board the 4-None",
        "geom": [-122.352435, 40.598131289999998],
        "where": "Canby at Churn Creek (north side)"} ],
    ["DescribeCrossingAtAlightEvent",
      {"when": null,
        "what": "Ride trip 3880A105B166 from stop_seq 33 to stop_seq 35",
        "geom": [[-122.3524, 40.59807],...[-122.353634, 40.589312]],
        "where": null}],
    ["AlightEvent",
      {"when": "2009-12-14 17:59:00-08:00",
        "what": "Alight",
        "geom": [-122.3536326, 40.589311940000002],
        "where": "Canby Road Transfer Facility"}],
    ...
  ]
}
```



Mise en forme côté client

The screenshot displays the OpenTripPlanner web interface. The browser address bar shows the URL <http://demo.opentripplanner.org/>. The main content area is titled "Open Transit openplans.org" and features a "Trip Planner" section. The current route is "9 AV TO BARROW".

Trip Planner Summary:

- PLAN A TRIP** 9 AV TO BARROW ✕
- 1: Wed, Apr 7th 3:11PM - Wed, Apr 7th 3:26PM**
(0 transfers, 14 minutes)
- Start at 9 AV**
- Walk to 50TH STREET**
About 4 minutes - 0.2 mi
- 1. Walk southwest on 9 AV (131 ft)**
- 2. Left at W 50 ST (0.2 mi)**
- 3. Left at 8 AV (66 ft)**
- Subway E**
- 3:16PM Depart 50TH STREET (To World Trade Ctr)**
8 minutes
- 3:24PM Arrive WEST 4 ST - UPPER LEVEL - WASHINGTON SQ**
- Walk to BARROW ST & W 4 ST**
About 2 minutes - 0.1 mi
- 1. Walk northwest on W WASHINGTON PL (0.1 mi)**

The map on the right shows the route in red, starting at 9th Avenue and ending at Barrow Street. The map includes labels for the Hudson River, Chelsea, Carment District, and Murray Hill. A scale bar indicates 500 meters and 2000 feet. The coordinates at the bottom of the map are -73.9961, 40.7641.

Fin – Questions ?

Merci de votre attention

Questions, informations :
infos@oslandia.com

Oslandia :
www.oslandia.com

