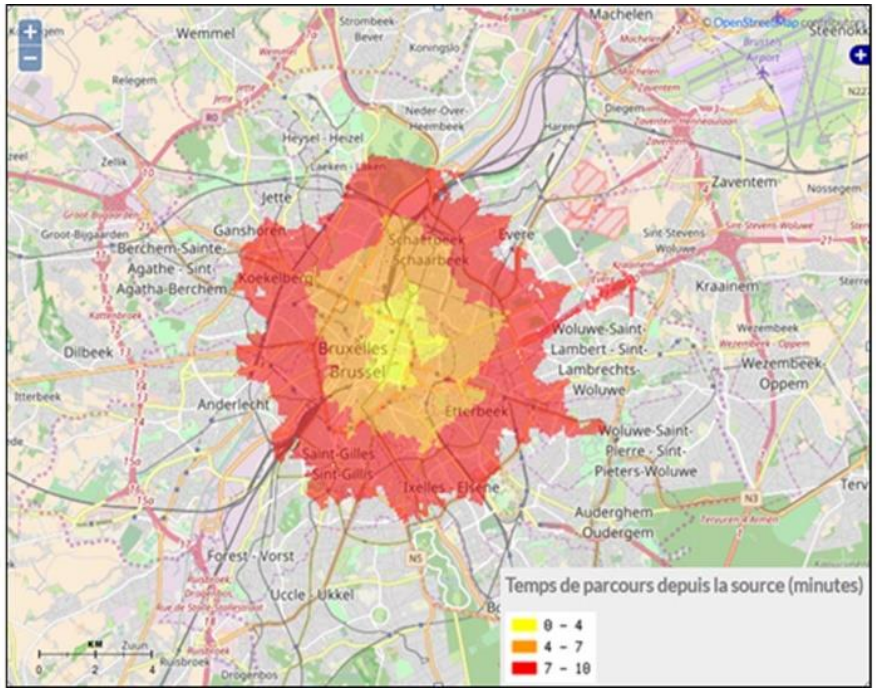
Analyse du plus court chemin



1. Théorie

– Définitions

» **Routage** = mécanisme par lequel des chemins sont sélectionnés dans un réseau pour acheminer les données d'un expéditeur jusqu'à un ou plusieurs destinataires

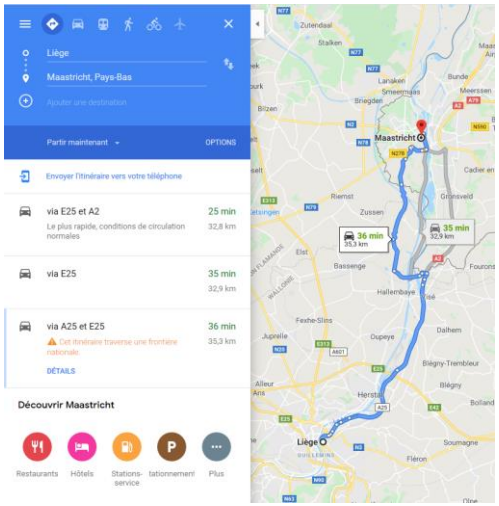
- Ex: réseau téléphonique, réseau de données (internet), réseaux de transports, …

» Application **géographique** : **recherche du plus court chemin** entre une source et une ou plusieurs destination(s) à travers un réseau dans l'espace

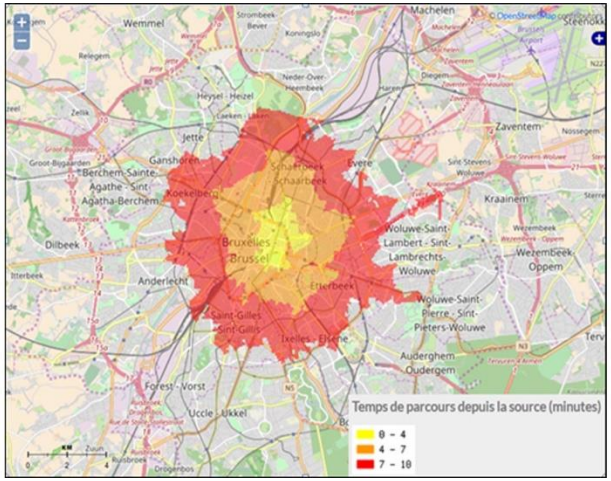
- Exemples:
  - chemin en voiture dans le réseau routier,
  - chemin dans un réseau de transports en commun (métro, bus, tram, …)
- Plusieurs types de distances peuvent être considérés
  - Distance métrique (ex: km parcourus en voiture)
  - Distance-temps (ex: temps de parcours en voiture en heure de pointe)
  - Distance-coût (ex: coût pour rejoindre une destination via transports en commun)

– Analyse du plus court chemin et SIG

» Les calculs de plus court chemin font souvent partie des fonctionnalités demandées d'un **SIG** pour des **besoins**:

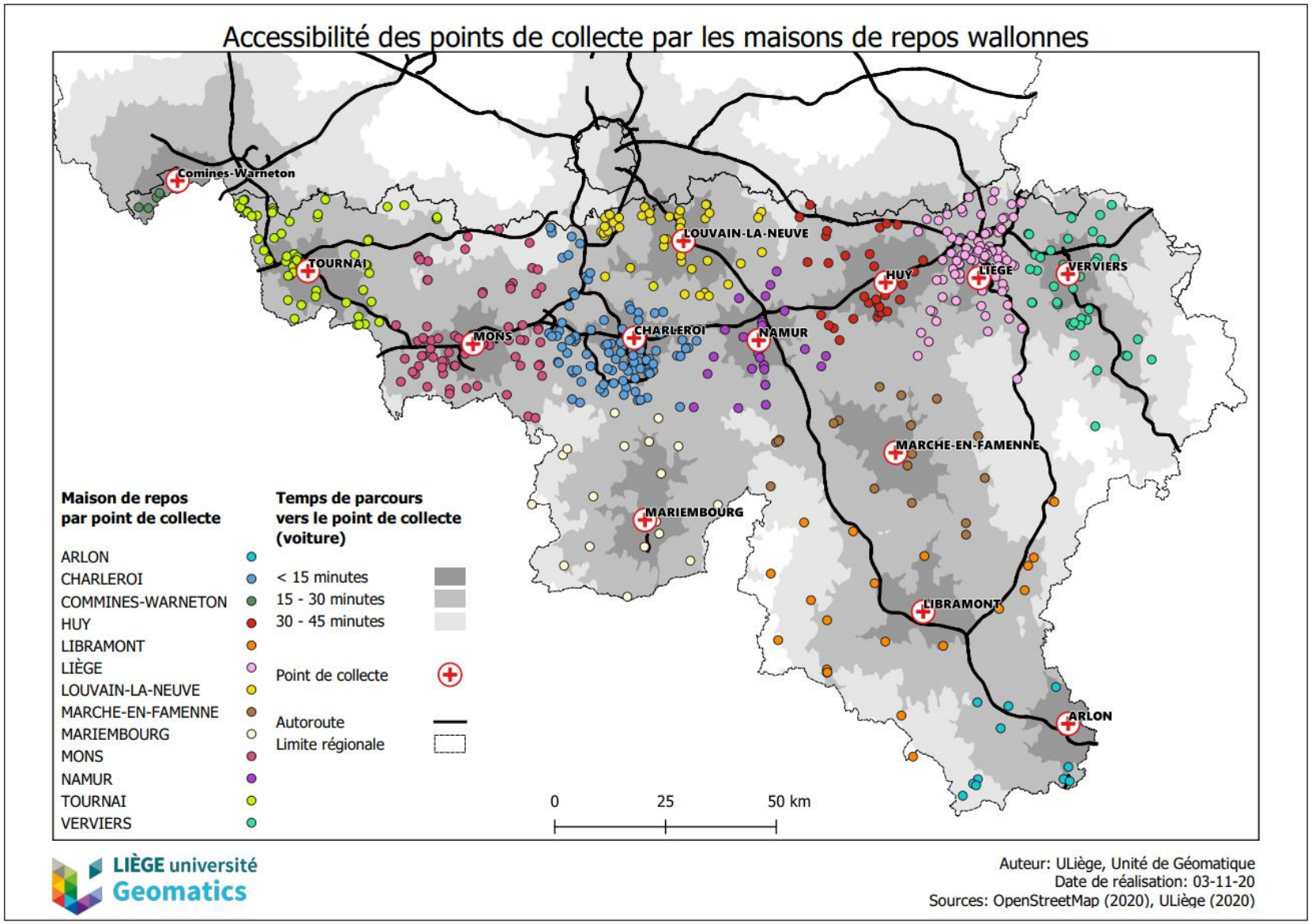


- **Opérationnels**:
  - Calculs d'itinéraires pour « monsieur tout le monde » (ex: google maps)
    - « Quel est le trajet pour atteindre ma destination? »
  - « Dispatching » pour les services d'urgence (pompiers ou aides médicales)
    - « Quel véhicule sur le terrain peut atteindre le plus rapidement le lieu d'incident? »
- **Analytiques**
  - Production de surfaces d'accessibilité ou cartes isochrones (ex: geomarketing)
    - « Combien de résidents peuvent atteindre un de mes magasins en moins de 15 min en voiture? »
  - Recherche d'une source sur base de lieux visités (ex: « crime mapping »)
    - « Où se situe le lieu de résidence d'un criminel en série? »



Analyse du plus court chemin

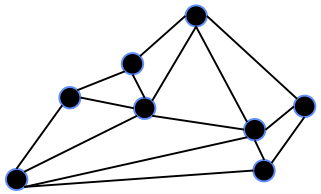
JP Kasprzyk & JP Donnay – 2019



Exemple d'application analytique exploitant le plus court chemin en distance-temps

– Méthodes et algorithmes

» Méthodes **vectorielles**



- Exploitation d'un **graphe valué** = structure arc-nœud où les arcs sont associés à au moins une valeur d'attribut (distance)
  - Algorithme de **Dijkstra**, A*, …
- **Avantage** principal:
  - Prise en compte des sens uniques
- **Inconvénient** principal:
  - La valeur de distance calculée ne peut porter que sur un nœud (pas au milieu d'une arête, par exemple)

» Méthodes **raster**

5	0	0	5	0
0	4	0	4	0
0	0	3	0	0
0	0	2	0	0
0	0	1	0	0

↑

- Exploitation d'une **surface de coût** = raster où chaque pixel est associé à un coût (distance) pour être traversé
  - Algorithme de **propagation** sur surface de coût
- **Avantage** principal:
  - La valeur de distance calculée peut porter sur n'importe quel pixel
- **Inconvénients**:
  - Non-prise en compte des sens uniques
  - Précision du résultat tributaire de la résolution de la surface de coût

– **Algorithme de Dijkstra**

» Algorithme en n-1 itérations, pour un graphe d'ordre n, fournissant les plus courts chemins entre un sommet d'origine et tous les autres sommets.

- À chaque itération, le chemin le plus court vers 1 sommet du graphe, au moins, est fixé.

» Définition de deux paramètres pour chaque sommet :

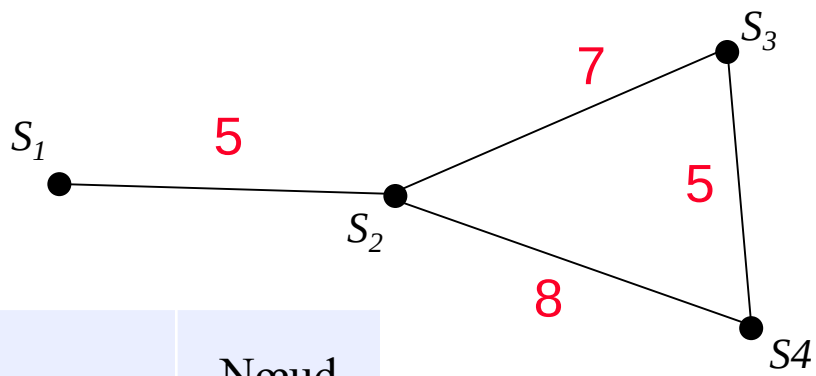
- Le numéro du dernier sommet traversé sur le chemin de l'origine.
- La longueur (valeur) cumulée des arêtes traversées depuis l'origine (initialisation à l'infini).

» À chaque itération :

- Essai d'amélioration de la distance cumulée de tous les chemins non définitifs en **passant par un sommet adjacent marqué définitivement**.
- Marquage définitif du sommet présentant la plus petite longueur cumulée.
  - Plusieurs sommets présentant la même valeur minimale sont marqués en même temps.

» À la fin de la procédure:

- Tous les nœuds du graphe ont comme valeur le plus court chemin depuis la source (accessibilité depuis la source)
- Les chemins (ordre des sommets traversés depuis l'origine vers chaque sommet du graphe) sont retrouvés par récurrence (premier paramètre).



Algorithme de Dijkstra

Origine : S_1

Etape	Nœud marqué	S_2	S_3	S_4
1	S_2	[S_1] 5	[-] ∞	[-] ∞
2	S_3	[S_1] 5	[S_2] 12	[S_2] 13
3	S_4	[S_1] 5	[S_2] 12	[S_2] 13 [S_3] 17

- **Propagation raster**
  - » Conversion du réseau en mode maillé :
    - Superposition du territoire d'analyse par une grille de pixels :
      - Résolution (taille des pixels) choisie selon l'échelle d'analyse, la précision du réseau, la taille de l'image et/ou le temps de traitement.
    - Rastérisation du réseau valué dans l'image :
      - Tous les pixels traversés par les arêtes du réseau sont marqués :
        - Par la valeur 1 en présence d'un graphe non valué.
        - Par la valeur associée à l'arête traversant le pixel dans les autres cas, **éventuellement pondérée par la longueur de traversée d'un pixel**.
      - Les autres pixels sont marqués d'une valeur signifiant l'absence de données (**pas** la valeur 0 si l'attribut des arêtes peut être nul).
  - » Définition de l'origine :
    - Construction d'une image identique (superposable) à celle contenant le réseau.
    - L'entité ponctuelle ou zonale constituant l'origine doit se situer "sur" le réseau et est identifiée par un ou plusieurs pixels non nuls dans l'image.
    - Plusieurs origines sont possibles dans la même image (pixels voisins ou non).

» Algorithme de propagation raster:

- Principe : report (**cumul**) de la valeur du pixel courant sur les valeurs des pixels voisins (connexité 8).
- Application au réseau rastérisé :
  - Propagation interdite sur les pixels extérieurs au réseau (**effet de barrière**).
  - Propagation interdite sur le dernier pixel traversé (**pas de retour**).

» Résultat : valeur d'accessibilité en chaque pixel appartenant au réseau.

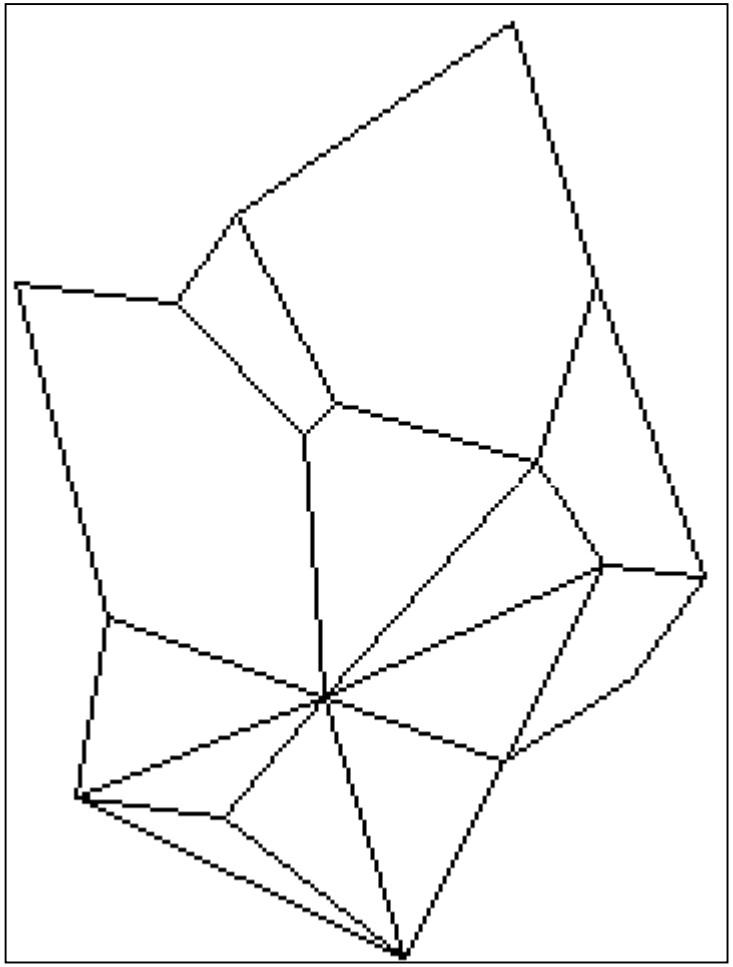
2.8	2.4	2	2.4	2.8
2.4	1.4	1	1.4	2.4
2	1	0	1	2
2.4	1.4	1	1.4	2.4
2.8	2.4	2	2.4	2.8

À gauche : propagation isotrope de la distance (unité = taille des pixels) depuis le pixel central (connexité 8)

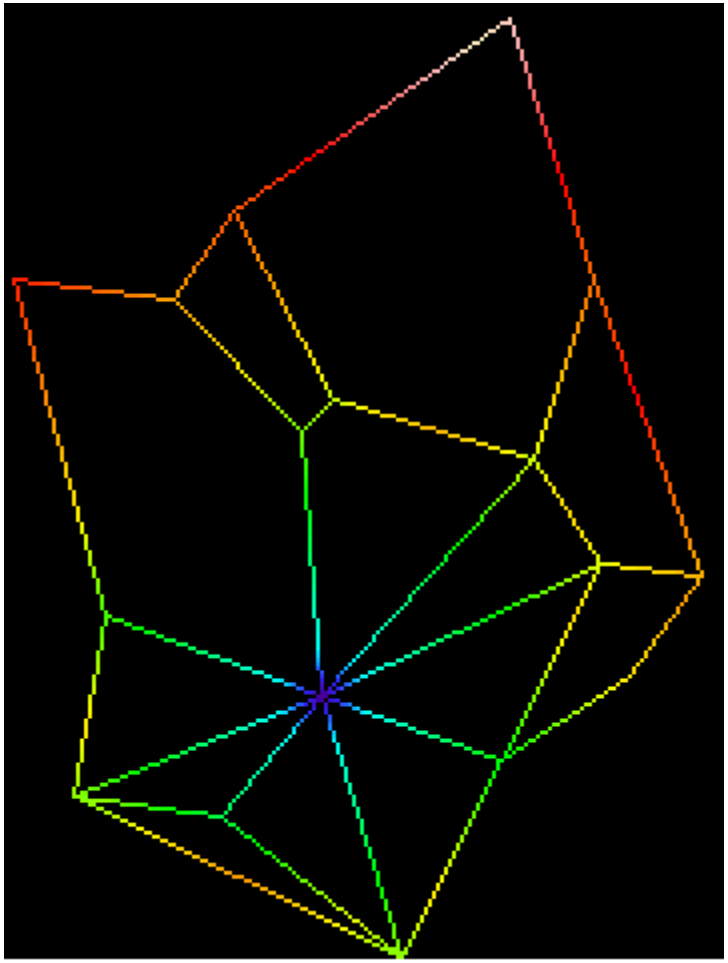
À droite : propagation d'un poids unitaire à travers un réseau rastérisé (espace anisotrope)

5	0	0	5	0
0	4	0	4	0
0	0	3	0	0
0	0	2	0	0
0	0	1	0	0

Analyse du plus court chemin



Réseau rastérisé



Accessibilité des pixels du réseau par propagation

– **Données routières**

» **Open Street Map** (open data)

- Open Street Map propose plusieurs couches thématiques de données vectorielles (**routes**, bâtiments, occupations du sol, entités administratives, …) basées sur le crowdsourcing
- La géométrie et topologie sont généralement correcte
- Pour les routes, l'aspect sémantique se limite à quelques classes
  - Pas de vitesse légale ou réelle

» **Données propriétaires**

- Google, Tomtom, …
- Tronçons associés à des vitesses de traversées calculées sur base des données « utilisateur »

type character varying(16)	count bigint
bus stop	2
no	2
construction	14
traffic island	14
tertiary link	23
secondary link	25
bridleway	31
primary link	78
trunk link	109
living street	112
trunk	128
motorway	168
motorway link	190
platform	192
track	307
pedestrian	526
unclassified	531
cycleway	542
steps	694
primary	1104
path	1269
secondary	1298
tertiary	2544
service	4949
footway	5914
residential	8671

Nombre d'arcs par type de route OSM pour la région de Bruxelles-Capitale

- **Outils**
  - » **PGRouting**
    - Extension open source de PostGIS pour la gestion de graphes routiers vectoriels
      - Calculs de plus courts chemins
      - Implémentation de l'algorithme de Dijkstra
  - » **ArcGIS** et **QGIS** proposent plusieurs outils pour l'analyse de réseau et la propagation raster

2. Exercice (PGRouting)

– **Création d'un graphe routier de la Belgique à partir d'Open Street Map**

» OSM2PO

- Outil écrit en Java permettant la conversion d'un fichier PBF d'Open Street Map en graphe (structure arc-nœud) compatible avec PgRouting (SQL)
- Installation préalable de **Java Runtime Environment** (JRE)
- Téléchargement d'OSM2PO: https://osm2po.de/
- Une fois téléchargé, décompresser le fichier « osm2po-5-2-43.zip » dans le répertoire de votre choix
- Le fichier « demo.bat » contient un exemple de commande DOS pour convertir un graphe routier de la ville d'Hamburg en exploitant OSM2PO

Instructions Java

```
java -Xmx1g -jar osm2po-core-5.2.43-signed.jar prefix=hh tileSize=x
http://download.geofabrik.de/europe/germany/hamburg-latest.osm.pbf
postp.0.class=de.cm.osm2po.plugins.postp.PgRoutingWriter
```

Adresse du fichier pbf

- Remplacer le fichier pbf d'Hamburg par celui de la Belgique:
  - https://download.openstreetmap.fr/extracts/europe/belgium.osm.pbf
  - Note: d'autres données au niveau mondial sont disponibles ici https://download.openstreetmap.fr/extracts/
- Exécution du fichier « demo.bat » modifié (cela prend quelques minutes…)

» Au terme de l'exécution du fichier « demo.bat », un fichier « **hh_2po_4pgr.sql** » a été créé dans le sous-répertoire « hh » d'OSM2PO

» Ce fichier comprend les instructions SQL pour importer le graphe routier de la Belgique dans une base de données PostGIS (plus de 600 000 arcs!)

C:\Windows\system32\cmd.exe

```
INFO  Graph ID is 962605957
INFO  Loading SourceVertex-EntryPoints
INFO  528.833 Vertices loaded. - 980M
INFO  Memory for 1.333.470 edges reserved. - 953M
INFO  1.333.470 Edges loaded. - 953M
INFO  528.833 Classes loaded. - 953M
INFO  Memory for 528.833 coords reserved. - 949M
INFO  Graph supports: +C -M -F -X -B -R -V -E
INFO  Graph is in memory - 949M free
INFO  Services started. Waiting for requests at
      http://localhost:8888/Osm2poService
```

– **Création d'une base de données postgres « td_routing »**

» Création de la BD via PGAdmin (voir chapitre 2)

» Importation des fonctions de PostGIS et PGRouting dans votre BD via les commandes SQL suivantes:

```
create extension postgis;
create extension pgrouting;
```

– **Importation du graphe SQL issu de OSM2PO dans votre BD via la fonction psql de Postgres (« invite de commande »)**

C:\Windows\System32\cmd.exe - psql -U postgres -d td_routing -h localhost -p 5432 -f hh_2po_...

```
Microsoft Windows [version 10.0.18362.535]
(c) 2019 Microsoft Corporation. Tous droits réservés.

D:\osm2po-5.2.43\hh>psql -U postgres -d td_routing -h localhost -p 5432 -f hh_2po_4pgr.sql
Mot de passe pour l'utilisateur postgres :
```

– **Analyse de la table « hh_2po_4pgr » ainsi créée**

» Un enregistrement (ligne) = un arc

» Chaque arc est associé à une géométrie de type polyligne (attribut « geom_way »)

» SRID: 4326 (WGS84 en latitude, longitude)

f_table_catalog character varying (256)	f_table_schema name	f_table_name name	f_geometry_column name	coord_dimension integer	srid integer	type character varying (30)
td_routing	public	hh_2po_4pgr	geom_way	2	4326	LINESTRING

Vue « geometry_columns » de la BD

Analyse du plus court chemin

ID de l'arc

Attributs OSM de la route comprenant l'arc

ID OSM du nœud de départ et d'arrivée

Classe de route OSM

	id [PK] integer	osm_id bigint	osm_name character varying	osm_meta character varying	osm_source_id bigint	osm_target_id bigint	clazz integer	flags integer
1	550910	368527390	[null]	[null]	1448825474	1448825478	21	1
2	550911	368527391	[null]	[null]	2483609330	266826469	21	1
3	550912	368527392	Avenue de Longwy	[null]	2483609207	2978161745	41	3

ID du nœud de départ

ID du nœud d'arrivée

Longueur de l'arc (km)

Vitesse sur l'arc (km/h) déduite de la classe de route OSM

Coût de traversée de l'arc (heures) = longueur/vitesse

Coût de traversée de l'arc en sens inverse (heures) (très grand nombre en cas de sens interdit)

source integer	target integer	km double precision	kmh integer	cost double precision	reverse_cost double precision
339770	471726	0.0147284	60	0.0002455	1000000
339767	65140	0.0234001	60	0.00039	1000000
339768	443275	0.0336356	40	0.0008409	0.0008409

Sens interdit

Extrait de la table hh_2po_4pgr exploitable par PGRouting (1/2)

Analyse du plus court chemin

Coordonnées (longitude, latitude) du nœud de départ en WGS84

Coordonnées (longitude, latitude) du nœud d'arrivée en WGS84

Géométrie de l'arc (WGS84)

x1 double precision	y1 double precision	x2 double precision	y2 double precision	geom_way geometry
5.7935428	49.5308332	5.7936891	49.5309205	0102000020E6100...
5.7936681	49.5310695	5.7933813	49.5311296	0102000020E6100...
5.7955305	49.5325282	5.7958434	49.5327479	0102000020E6100...

Extrait de la table hh_2po_4pgr exploitable par PGRouting (2/2)

– **Exploitation de la BD: recherche de l'arc le plus proche du bâtiment B5a**

» Coordonnées WGS84 du B5a:

- Latitude: 50.582691°
- Longitude: 5.566298°

» Requête SQL

```
select * from hh_2po_4pgr
order by st_distance(
                geom_way,
                st_setsrid(
                                st_geomfromtext(
                                                'POINT(5.566298 50.582691)'
                                ),
                                4326
                )
)
limit 1
```

Attribut « geometry » de la table d'arcs

Géométrie PostGIS du point représentant le B5a

Système de coordonnées WGS84

Résultat: le nœud « source » de l'arc sera exploité dans la requête suivante

	id [PK] integer	osm_id bigint	osm_name character varying	osm_meta character varying	osm_source_id bigint	osm_target_id bigint	clazz integer	flags integer	source integer
1	3892	4783056	Allée du Six Août	[null]	30621951	30621773	43	3	4248

– Exploitation de la BD: calcul d'une couche isochrone

» Recherche de tous les arcs situés à moins de 30 minutes en voiture du B5a

» Requête SQL: application de l'algorithme de Dijkstra par PGRouting via la fonction pgr_drivingDistance dont les paramètres sont:

- 5 attributs de la table graphe:
  - id,
  - nœud de départ,
  - nœud d'arrivée,
  - coût,
  - coût inverse
- L'ID du nœud de départ du Dijkstra (B5a = ID 4248)
- Le coût maximal atteignable par l'algorithme (30 min = 0,5 heure)

» Note: les attributs « cost » et « reverse_cost » déterminent le type de distance de la couche isochrone (distance métrique, distance-temps, distance-coût, …)

```
CREATE view arc as
SELECT * FROM
pgr_drivingDistance(
        'SELECT id, source, target, cost, reverse_cost FROM hh_2po_4pgr',
        4248,
        0.5
)
```

Création d'une vue « arc » avec le résultat

Application de la fonction Dijkstra de PGRouting (pgr_drivingDisance)

» Résultat (vue « arc »)

Séquence de Dijkstra | Id du noeud | Id de l'arc traversé | Coût de l'arc traversé | Coût cumulé de tous les arcs traversés

	seq integer	node bigint	edge bigint	cost double precision	agg_cost double precision
1	1	4248	-1	0	0
2	2	4245	539464	0.00048	0.00048
3	3	4246	3890	0.0003553	0.0008353
4	4	4240	539462	0.0004238	0.0012591
5	5	4250	3894	0.0010766	0.0015566
6	6	4242	197899	0.0009867	0.001822
7	7	195000	197900	0.0040871	0.0059091

» Visualisation du résultat dans QGIS

- L'algorithme pgr_drivingDistance ne retournant pas les géométries des arcs, il est d'abord nécessaire de joindre le résultat à la table de base

```
CREATE TABLE arc_geom as
SELECT arc.* , hh_2po_4pgr.geom_way
FROM hh_2po_4pgr, arc
WHERE hh_2po_4pgr.id=arc.edge
```

Création d'une nouvelle table « arc_geom » avec le résultat final

Condition de la jointure

	seq integer	node bigint	edge bigint	cost double precision	agg_cost double precision	geom_way geometry
1	2	4245	539464	0.00048	0.00048	0102000020E6100...
2	3	4246	3890	0.0003553	0.0008353	0102000020E6100...
3	4	4240	539462	0.0004238	0.0012591	0102000020E6100...
4	5	4250	3894	0.0010766	0.0015566	0102000020E6100...
5	6	4242	197899	0.0009867	0.001822	0102000020E6100...

» Connexion à la table via QGIS et symbolisation du résultat sur base de l'attribut « agg_cost »