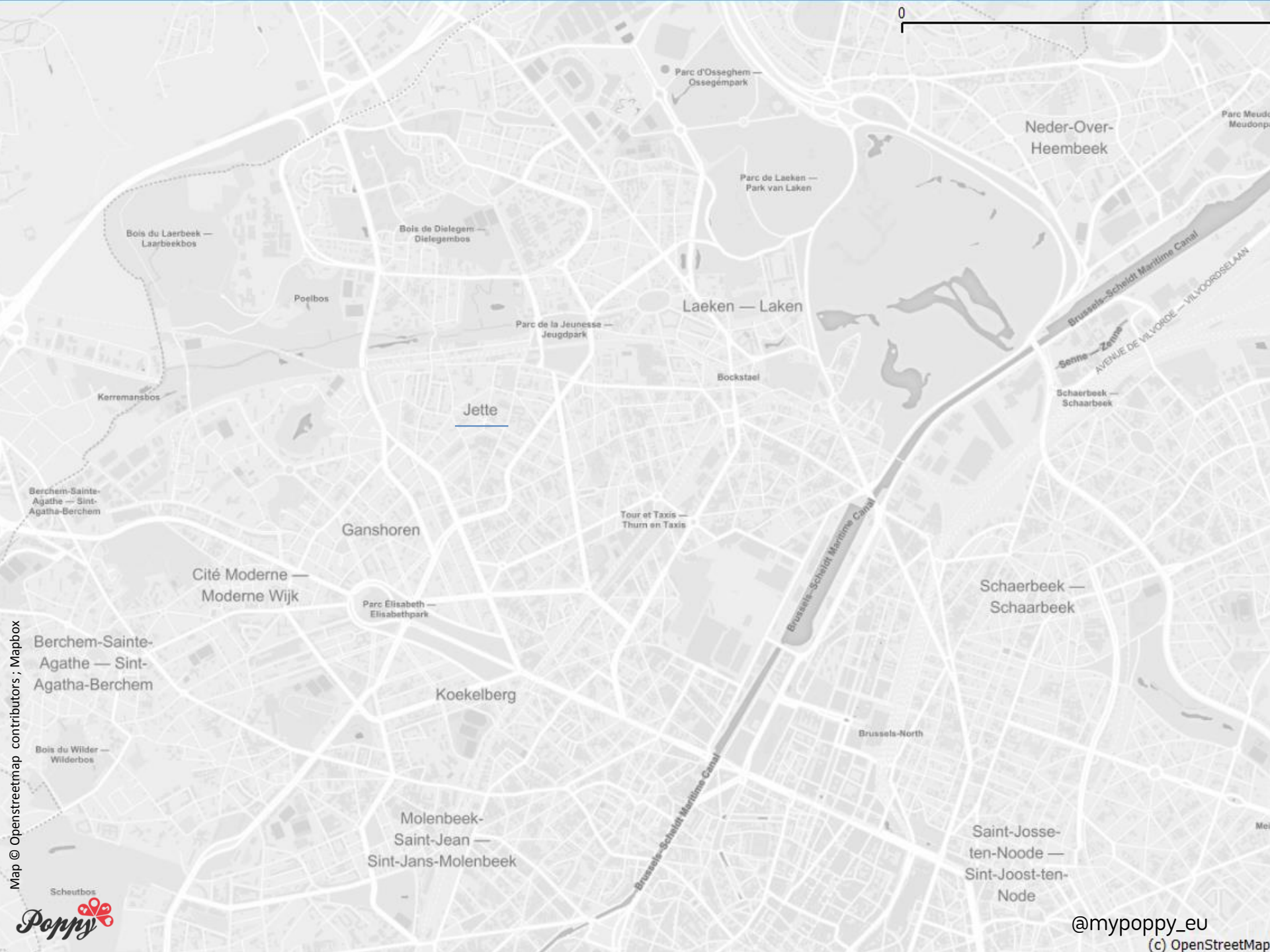
Optimiser
la collecte des déchets (connectés)
et le déneigement
avec des outils libres ?

Christophe Cloquet
christophe@my-poppy.eu
@mypoppy_eu

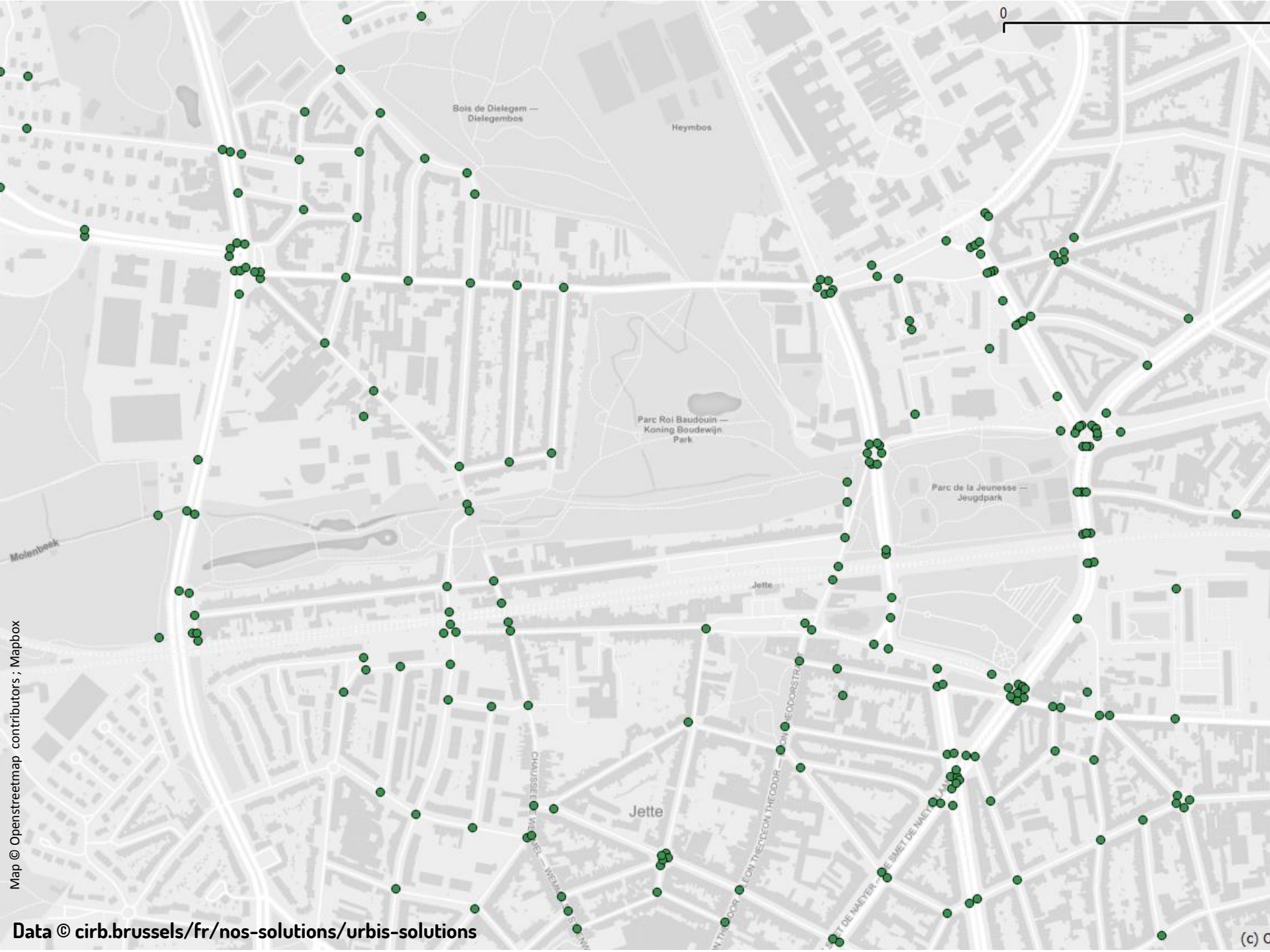

0
England
Amsterdam
The Netherlands
Magdeburg
Naturpark Teutoburger Wald / Eggegebirge
London
Germany
Brussels
Belgium
Lille
Parc Naturel Marin des Estuaires Picards et de la Mer d'Opale
Naturpark Hohes Venn-Eifel
Parc naturel régional de l'Avesnois
Frankfurt am Main
Luxembourg
Luxembourg
Le Havre
Naturpark Frankenhöhe
Parc naturel régional de la Montagne de Reims
Parc naturel régional de Lorraine
Naturpark Altmühltal
Paris
Stuttgart
Parc naturel régional de la forêt d'Orient
Parc naturel régional des Ballons des Vosges
Munich
Orléans
France
Tours
Zurich
Innsbruck
Liechtenstein
Parc naturel régional du Morvan
Bern
Switzerland
Parc naturel régional de la Brenne
Geneva
Map © Openstreetmap contributors ; Mapbox
@mypoppy_eu

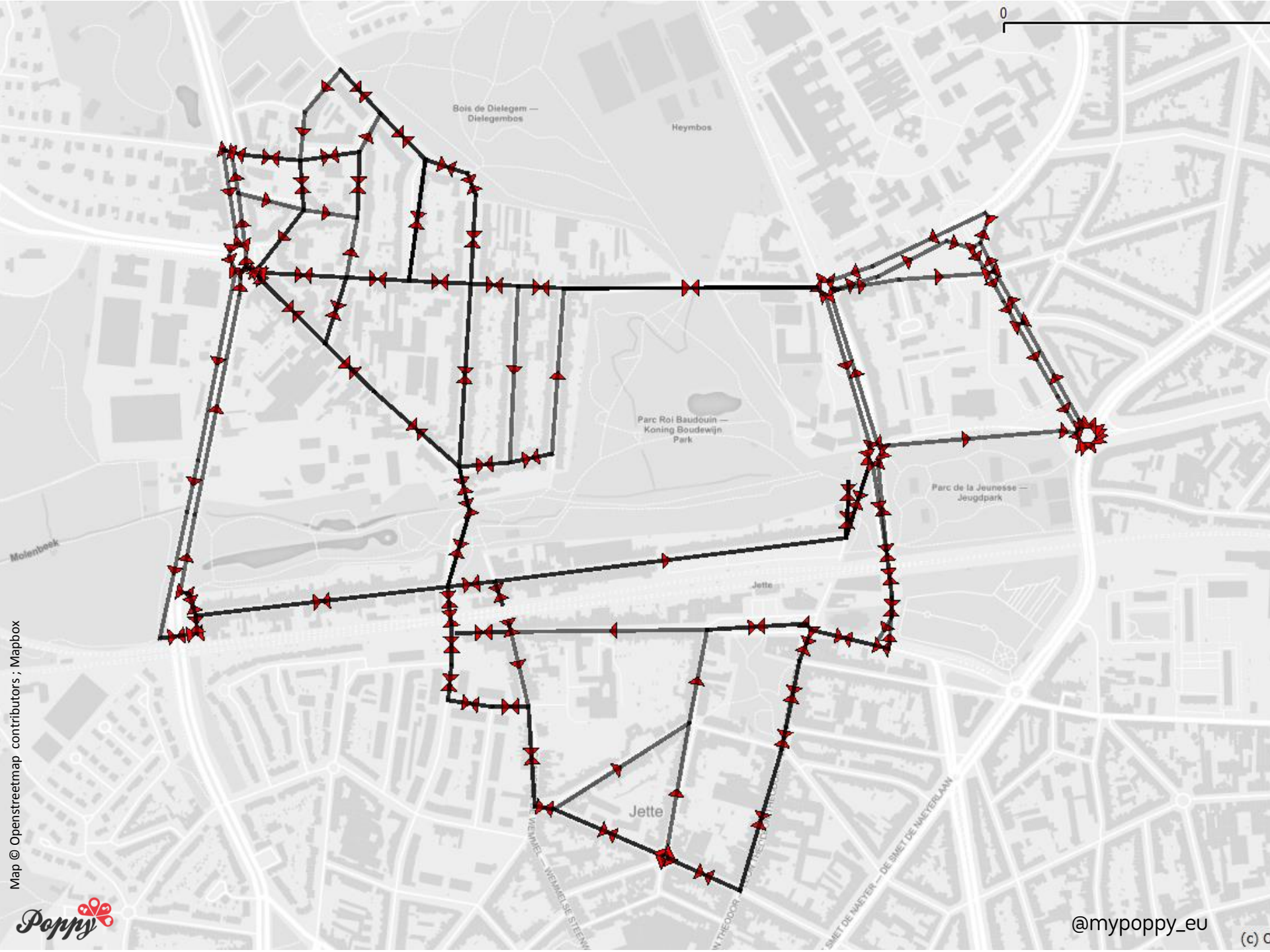

0
Parc d'Osseghem — Ossegempark
Neder-Over-Heembeek
Parc de Laeken — Park van Laken
Bois du Laerbeek — Laarbeekbos
Bois de Dielegem — Dielegembos
Poelbos
Laeken — Laken
Parc de la Jeunesse — Jeugdpark
Brussels-Scheldt Maritime Canal
Senne — Zenne
AVENUE DE VILVORDE — VILVOORDSELAAN
Bockstael
Schaerbeek — Schaarbeek
Kerremansbos
Jette
Berchem-Sainte-Agathe — Sint-Agatha-Berchem
Ganshoren
Tour et Taxis — Thurn en Taxis
Cité Moderne — Moderne Wijk
Parc Elisabeth — Elisabethpark
Schaerbeek — Schaarbeek
Berchem-Sainte-Agathe — Sint-Agatha-Berchem
Koekelberg
Bois du Wilder — Wilderbos
Brussels-North
Molenbeek-Saint-Jean — Sint-Jans-Molenbeek
Saint-Josse-ten-Noode — Sint-Joost-ten-Node
Scheutbos
Map © Openstreetmap contributors ; Mapbox
@mypoppy_eu
(c) OpenStreetMap


0
Bois de Dielegem — Dielegembos
Heymbos
Parc Roi Baudouin — Koning Boudewijn Park
Parc de la Jeunesse — Jeugdpark
Molenbeek
Jette
Jette
CHAUSSEE DE WEMMEL — WEMMELSE STEENWEG
LEON THEODOR — LEON THEODOR — LEON THEODORSTRAAT
DE SMET DE NAEYER — DE SMET DE NAEYERLAAN
Map © Openstreetmap contributors ; Mapbox
Poppy
@mypoppy_eu
(c) O

www.jette.irisnet.be/fr/ma-commune/jette-en-images-def/les-bois-et-les-parcs/parc-baudouin/image_preview

https://www.resto.be/restaurant/bruxelles/1090-jette/35669-brasserie-le-central

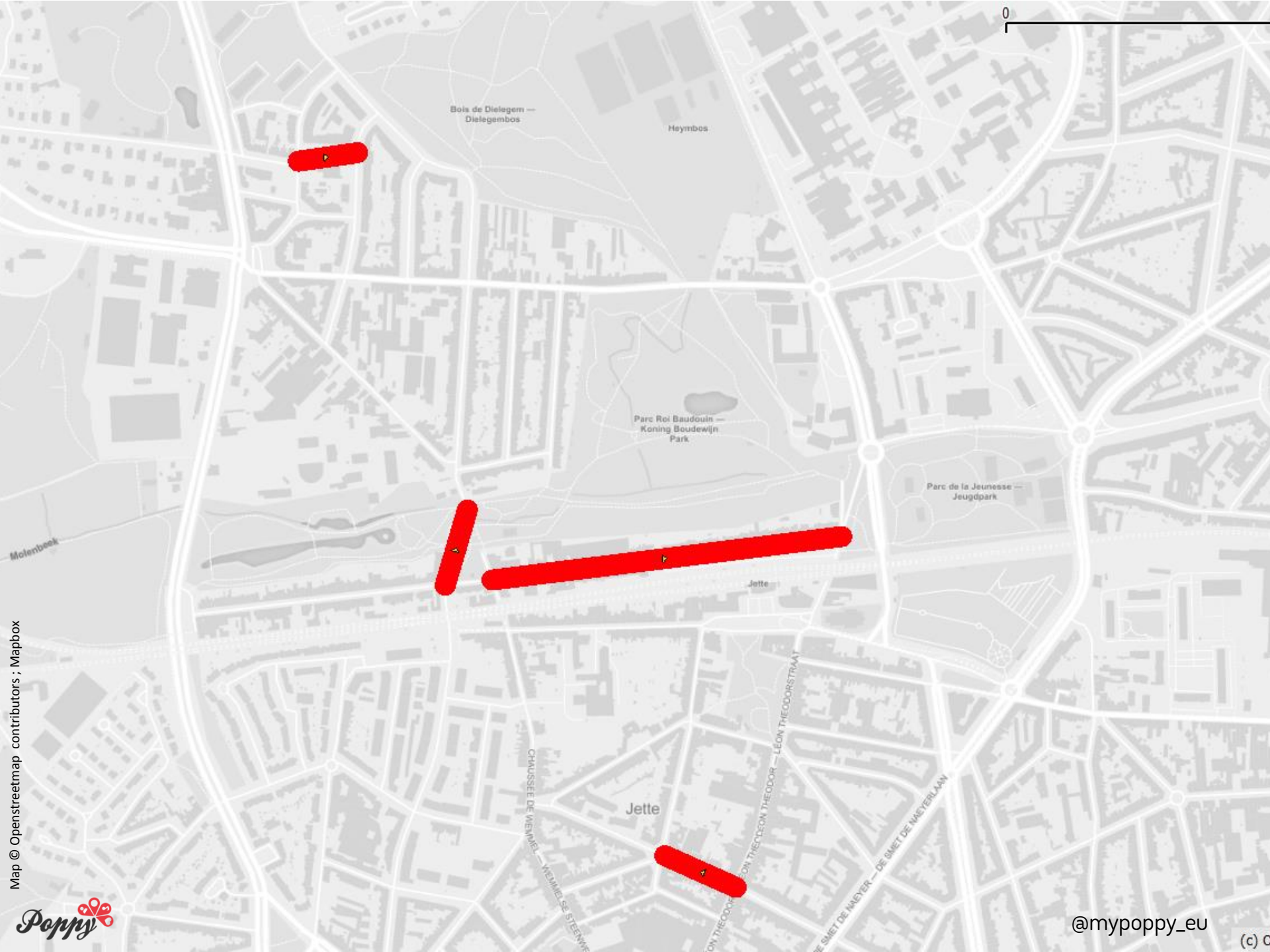

0
Bois de Dielegem — Dielegembos
Heymbos
Parc Roi Baudouin — Koning Boudewijn Park
Parc de la Jeunesse — Jeugdpark
Molenbeek
Jette
Jette
Map © Openstreetmap contributors ; Mapbox
Data © cirb.brussels/fr/nos-solutions/urbis-solutions
(c) C


0
Bois de Dielegem — Dielegembos
Heymbos
Parc Roi Baudouin — Koning Boudewijn Park
Parc de la Jeunesse — Jeugdpark
Molenbeek
Jette
Jette
Map © Openstreetmap contributors ; Mapbox
@mypoppy_eu
(c) O

1. Comment collecter les poubelles efficacement ?

2. Comment optimiser le salage des routes ?


0
Bois de Dielegem — Dielegembos
Heymbos
Parc Roi Baudouin — Koning Boudewijn Park
Parc de la Jeunesse — Jeugdpark
Molenbeek
Jette
Jette
CHAUSSEE DE WEMMEL — WEMMELSE STEENWEG
LEON THEODOR — LEON THEODORSTRAAT
DE SMET DE NAEYER — DE SMET DE NAEYERLAAN
Map © Openstreetmap contributors ; Mapbox
Poppy
@mypoppy_eu
(c) O


0
Bois de Dielegem — Dielegembos
Heymbos
Parc Roi Baudouin — Koning Boudewijn Park
Parc de la Jeunesse — Jeugdpark
Molenbeek
Jette
Jette
CHAUSSÉE DE WEMMEL — WEMMELSE STEENWEG
LÉON THÉODORSTRAAT
DE SMET DE NAEYERLAAN
Map © Openstreetmap contributors ; Mapbox
@mypoppy_eu
(c) O

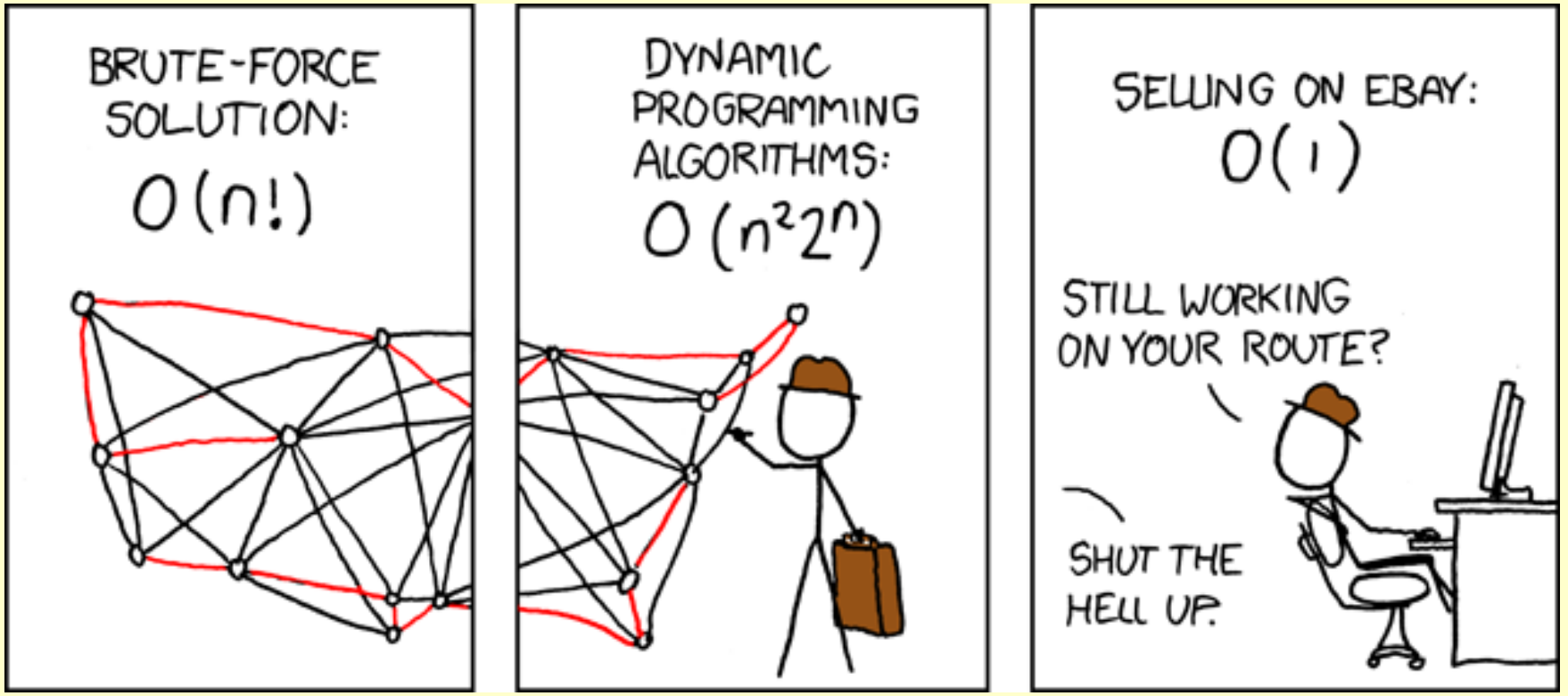
http://www.math-info.univ-paris5.fr/~moisan/gtnum/data/recuit/car54.jpg

Collecte de poubelles

- **voyageur de commerce**
- pg_routing : `pgr_tsp()`

-> fin de l'histoire ?

fr.wikipedia.org/wiki/Probl%C3%A8me_du_voyageur_de_commerce



Qu'optimiser ?

- un max de points dans un temps donné
- un **min de temps pour un nombre de points donnés**
- ...

Quid des
sens uniques ?

https://upload.wikimedia.org/wikipedia/commons/e/e2/NYC_-_Greene_St_and_Prince_St_One_Way_signs_-_0209.jpg

Outils

https://www.flickr.com/photos/peterhellberg/4572432746/sizes/o/

/ données

PostgreSQL + PostGIS +

pgrouting.org

+ plugin

python™

pgRouting library contains following features:

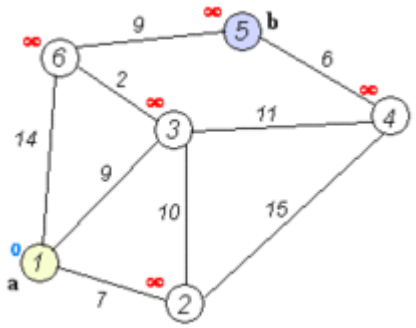
- All Pairs Shortest Path, Johnson's Algorithm
- All Pairs Shortest Path, Floyd-Warshall Algorithm
- Shortest Path A*
- Bi-directional Dijkstra Shortest Path
- Bi-directional A* Shortest Path
- Shortest Path Dijkstra
- Driving Distance
- K-Shortest Path, Multiple Alternative Paths
- K-Dijkstra, One to Many Shortest Path
- Traveling Sales Person
- Turn Restriction Shortest Path (TRSP)

pgr_dijkstra()

pgr_tsp()

Pour optimiser, on doit connaître les **distances** entre les noeuds (poubelles)

Les distances **sur le réseau**, pas les distances euclidiennes…




```
pgr_dijkstra(…, i, j, false, false)
```

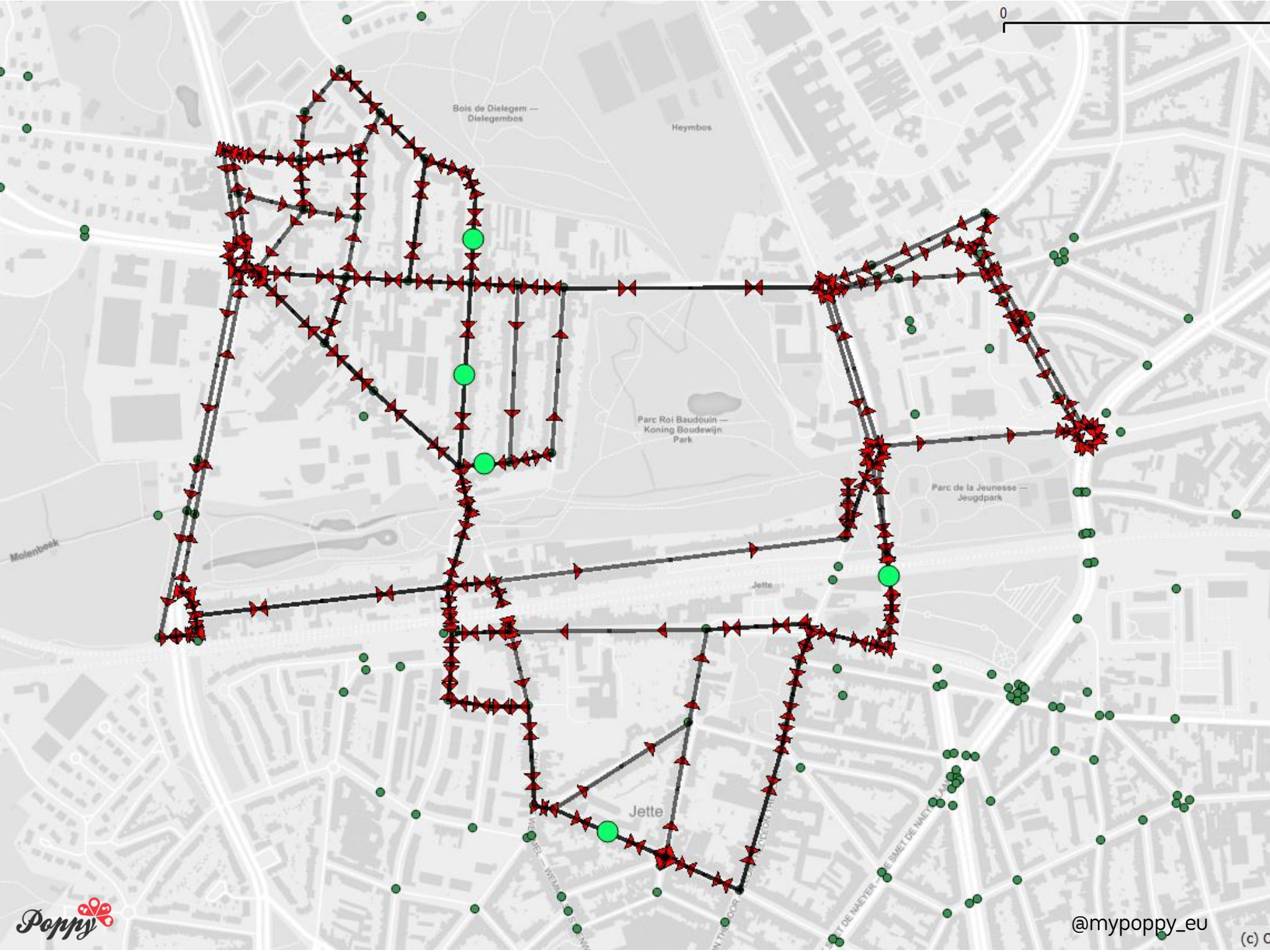
À cause des **sens uniques,** les distances ne sont pas nécessairement les mêmes dans un sens et dans l'autre

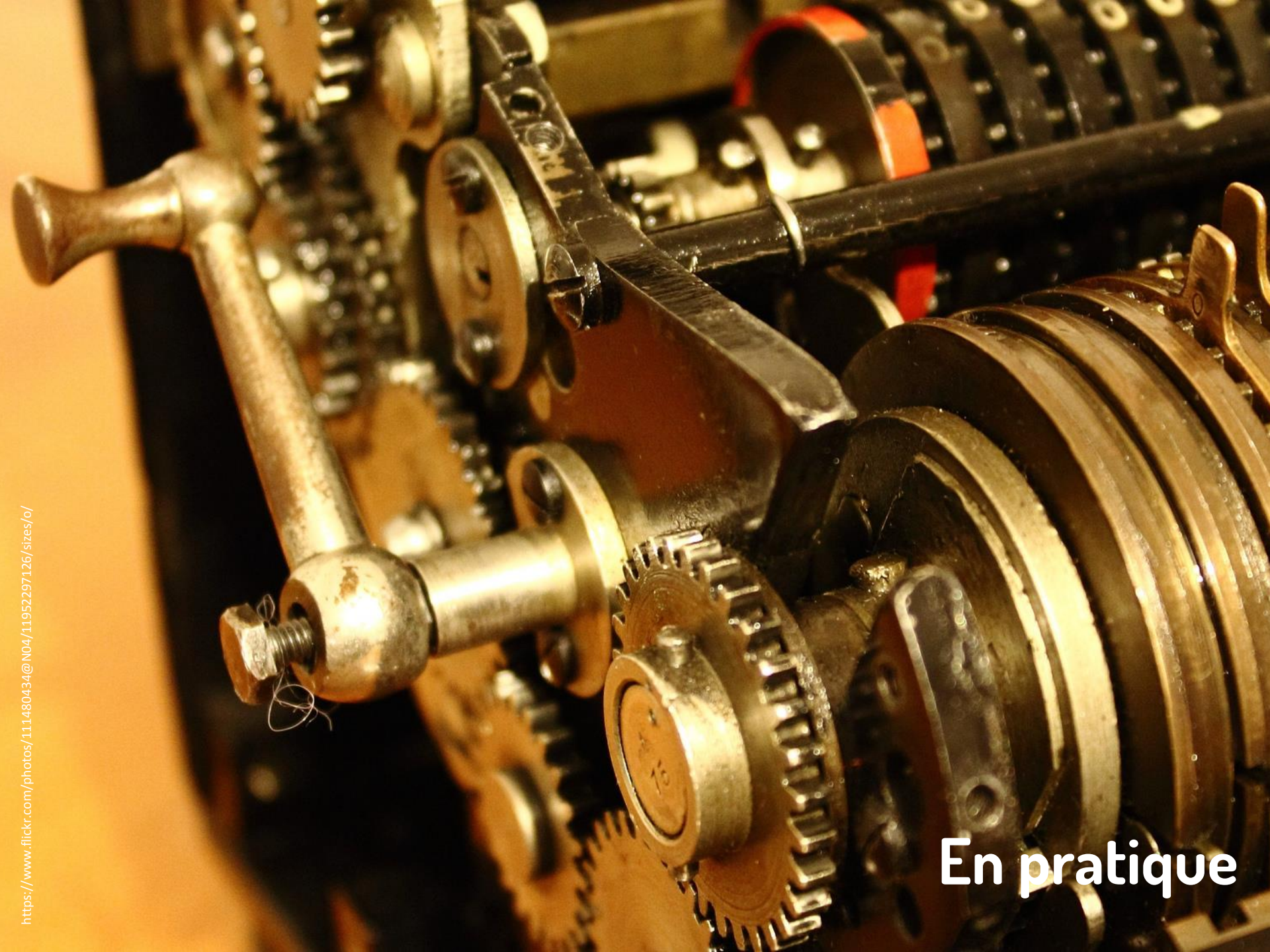
```
pgr_dijkstra(…, i, j, true, false)
```

S'il y a une colonne **reverse_cost** (cf OSM)

```
pgr_dijkstra(…, i, j, true, true)
```

https://fr.wikipedia.org/wiki/Algorithme_de_Dijkstra


0
Bois de Dielegem —
Dielegembos
Heymbos
Parc Roi Baudouin —
Koning Boudewijn
Park
Parc de la Jeunesse —
Jeugdpark
Molenbeek
Jette
Jette
Poppy
@mypoppy_eu
(c) C



En pratique

https://www.flickr.com/photos/111480434@N04/11952297126/sizes/o/

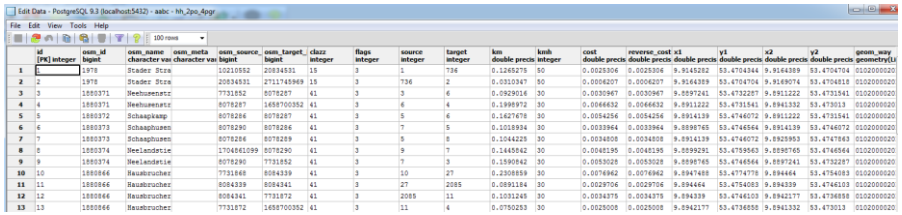
Charger des données à partir d'OpenstreetMap

Avec osm2po

http://osm2po.de/

- ```
  java -Xmx1g -jar osm2po-core-5.1.0-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
  ```

- ```
  psql.exe -U myusername -d mydatabase -q -f hh_2po_4pgr.sql
  ```



```
CREATE TABLE hh_2po_4pgr
(
  id integer NOT NULL,
  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,
  target integer,
  km double precision,
  kmh integer,
  cost double precision,
  reverse_cost double precision,
  x1 double precision,
  y1 double precision,
  x2 double precision,
  y2 double precision,
  geom_way geometry(LineString,4326),
  CONSTRAINT pkey_hh_2po_4pgr PRIMARY KEY (id)
)
```

https://anitagraser.com/2011/12/18/osm2po-part-2-pgrouting-on-osm-the-easy-way/

Avec données “privées”

1. Créer une table de liens (le réseau)

Edit Data - PostgreSQL 9.3 (localhost:5432) - aabc - urbadm_sn

File Edit View Tools Help

100 rows

	gid [PK] serial	id numeric	snft character var	snlv character var	versionid numeric	geom geometry(Point)
1	1	3007639.000000000000000	I	0	5160919.000	0101000000AE47E17ACE0B02415EBA490CB52E0541
2	2	3007572.000000000000000	I	0	5161183.000	0101000000E3A59BC44CEA0141A69BC4205D470541
3	3	3007727.000000000000000	I	0	5161189.000	01010000007368 91ED06D40141D9CEF753F96F0541
4	4	3007675.000000000000000	I	0	5161231.000	01010000008012 8DD82AE801411FEEFC18144D0541

Edit Data - PostgreSQL 9.3 (localhost:5432) - aabc - urbadm_edges

File Edit View Tools Help

100 rows

	uniqueid [PK] bigint	id1 bigint	id2 bigint	oneway boolean
1	1	3007547	3007640	FALSE
2	2	3007640	3007641	FALSE
3	3	3007641	3007642	FALSE
4	4	3007642	3007643	FALSE
5	5	3007643	11100623	FALSE
6	6	11100623	11100619	FALSE
7	7	11100619	3007644	FALSE
8	8	3007644	15750296	FALSE
9	10	3007644	3007643	FALSE
10	11	3007644	11100627	FALSE
11	12	11100627	3007545	FALSE
12	13	3007545	3007641	FALSE
13	14	3007545	3007546	FALSE
14	15	3007640	3007546	TRUE
15	16	3007546	3007912	FALSE
16	17	3007912	3007547	TRUE

```
CREATE TABLE urbadm_edges_with_geom
AS
(
            SELECT e.uniqueid as gid, e.id1 as source, e.id2 as target,
n1.geom as startpoint, n2.geom as endpoint, ST_MakeLine(n1.geom, n2.geom),
ST_Distance(n1.geom, n2.geom) as dist
            FROM urbadm_edges as e
            INNER JOIN urbadm_sn as n1 on n1.id = e.id1
            INNER JOIN urbadm_sn as n2 on n2.id = e.id2

            UNION

            SELECT -e.uniqueid as gid, e.id2 as source, e.id1 as target,
n2.geom as startpoint, n1.geom as endpoint, ST_MakeLine(n2.geom, n1.geom),
ST_Distance(n2.geom, n1.geom) as dist
            FROM urbadm_edges as e
            INNER JOIN urbadm_sn as n1 on n1.id = e.id1
            INNER JOIN urbadm_sn as n2 on n2.id = e.id2
            WHERE oneway = false
);
```

2. Créer une table de noeuds (ici, 5 poubelles, au hasard, au milieu des rues)

```
CREATE TABLE urbadm_sn_half AS
(
            SELECT id::integer, geom FROM urbadm_sn
            UNION
            SELECT -row_number() OVER (ORDER BY 1), ST_CENTROID(st_makeline)
            FROM (SELECT DISTINCT st_makeline FROM urbadm_edges_with_geom) AS a
);
```

2. Créer une table de noeuds (ici, 5 poubelles, au hasard, au milieu des rues)

```
CREATE TABLE urbadm_sn_half AS …

CREATE TABLE urbadm_edges_half_with_geom AS
(
            SELECT row_number() OVER (ORDER BY 1) as gid, source, target, startpoint, endpoint,
ST_MakeLine(startpoint, endpoint) , ST_Distance(startpoint, endpoint) as dist
            FROM
            (
                        SELECT source, startpoint, b.id as target, st_centroid(st_makeline) as
endpoint    FROM urbadm_edges_with_geom as a INNER JOIN urbadm_sn_half as b ON
st_centroid(a.st_makeline) = b.geom
                        UNION
                        SELECT b.id as source, st_centroid(st_makeline) as startpoint, target,
endpoint    FROM urbadm_edges_with_geom as a INNER JOIN urbadm_sn_half as b ON
st_centroid(a.st_makeline) = b.geom
            ) as a
);
```

2. Créer une table de noeuds (ici, 5 poubelles, au hasard, au milieu des rues)

```
CREATE TABLE urbadm_sn_half AS …

CREATE TABLE urbadm_edges_half_with_geom AS …

CREATE TABLE urbadm_nodes_half AS   -- construction d'une liste des noeuds
(
            SELECT DISTINCT id, x, y, dist
            FROM
            (
                        SELECT DISTINCT source as id, ST_X(startpoint) as x, ST_Y(startpoint) as y
                        FROM urbadm_edges_half_with_geom
                        UNION
                        SELECT DISTINCT target as id, ST_X(endpoint) as x, ST_Y(endpoint) as y
                        FROM urbadm_edges_half_with_geom
            ) as a
);
```

2. Créer une table de noeuds (ici, 5 poubelles, au hasard, au milieu des rues)

```
CREATE TABLE urbadm_sn_half AS …

CREATE TABLE urbadm_edges_half_with_geom AS …

CREATE TABLE urbadm_nodes_half AS …

CREATE TABLE random_nodes -- sélection de 5 noeuds au hasard (doivent être au milieu (<=0) de
tronçons de plus de 30 m (> 30)
AS
(
            SELECT * FROM (
                        SELECT id
                        FROM urbadm_nodes_half WHERE id <= 0 AND dist > 30
                        order by random() LIMIT 5
            ) as a ORDER BY id
);
```

3. Calculer la matrice de distances entre les noeuds

- entre deux noeuds
- matrice complète

Cas particuliers :

- tenir compte de la position de la/des poubelles dans la rue

- **sens uniques**

4. pgr_tsp()

docs.pgrouting.org/v2.1.0-beta/src/common/doc/convenience/vids_to_dmatrix2.html

S'il y a des sens uniques : étendre la matrice

TSP – Infrastructure for the Traveling Salesperson Problem

Michael Hahsler
Southern Methodist University

Kurt Hornik
Wirtschaftsuniversität Wien

A different manipulation is to reformulate an asymmetric TSP as a symmetric TSP. This is possible by doubling the number of cities (Jonker and Volgenant 1983). For each city a dummy city is added. Between each city and its corresponding dummy city a very small value (e.g., $-\infty$) is used. This makes sure that each city always occurs in the solution together with its dummy city. The original distances are used between the cities and the dummy cities, where each city is responsible for the distance going to the city and the dummy city is responsible for the distance coming from the city. The distances between all cities and the distances between all dummy cities are set to a very large value (e.g., ∞) which makes these edges infeasible. An example for equivalent formulations as an asymmetric TSP (to the left) and a symmetric TSP (to the right) for three cities is:

$$\begin{pmatrix} 0 & d_{12} & d_{13} \\ d_{21} & 0 & d_{23} \\ d_{31} & d_{32} & 0 \end{pmatrix} \Longleftrightarrow \begin{pmatrix} 0 & \infty & \infty & -\infty & d_{21} & d_{31} \\ \infty & 0 & \infty & d_{12} & -\infty & d_{31} \\ \infty & \infty & 0 & d_{13} & d_{23} & -\infty \\ -\infty & d_{12} & d_{13} & 0 & \infty & \infty \\ d_{21} & -\infty & d_{23} & \infty & 0 & \infty \\ d_{31} & d_{32} & -\infty & \infty & \infty & 0 \end{pmatrix}$$

Instead of the infinity values suitably large negative and positive values can be used. The new symmetric TSP can be solved using techniques for symmetric TSPs which are currently far more advanced than techniques for ATSPs. Removing the dummy cities from the resulting tour gives the solution for the original ATSP.

Calcul de distances

```
CREATE OR REPLACE FUNCTION p_distance(i integer, j integer)
RETURNS double precision AS $$
        DECLARE out double precision;
        BEGIN
                SELECT   sum(cost)
                INTO     out
                FROM     pgr_dijkstra(
                                'SELECT  gid AS id,
                                         source::int4 AS source,
                                         target::int4 AS target,
                                         dist::float8 AS cost
                                FROM     urbadm_edges_half_with_geom',
                                i,
                                j,
                                false, false);
        RETURN out;
        END;
$$ LANGUAGE plpgsql;
```

Calcul de distances

```
CREATE OR REPLACE FUNCTION p_distance(i integer, j integer)
RETURNS double precision AS $$
        DECLARE out double precision;
        BEGIN
                SELECT   sum(cost)
                INTO     out
                FROM     pgr_dijkstra(
                                'SELECT  gid AS id,
                                         source::int4 AS source,
                                         target::int4 AS target,
                                         dist::float8 AS cost
                                FROM     urbadm_edges_half_with_geom',
                                i,
                                j,
                                true, false);
        RETURN out;
        END;
$$ LANGUAGE plpgsql;
```

asym

reverse_cost

Créer une matrice de distances

```
CREATE OR REPLACE FUNCTION m_distance( i integer[], j integer[] ) RETURNS double precision[][]

        DECLARE nut double precision[];         DECLARE zero double precision[];
        DECLARE out double precision[][];       DECLARE qut double precision[][];

        DECLARE k integer;  DECLARE m integer;  DECLARE a integer;  DECLARE b integer;

        BEGIN

        FOREACH k IN ARRAY i
        LOOP
                nut = array[]::double precision[];

                FOREACH m IN ARRAY j
                LOOP
                        IF (k=m) THEN nut = array_append(nut, 0::double precision);
                        ELSE          nut = array_append(nut, p_distance(k,m));
                        END IF;

                END LOOP;
                nut = nut;
                out := out || ARRAY[nut];
        END LOOP;

        RETURN out;
        END;
$$ LANGUAGE plpgsql;
```

Cf aussi **pgr_apspWarshall** (pas pgr_apspJohnson si asym)

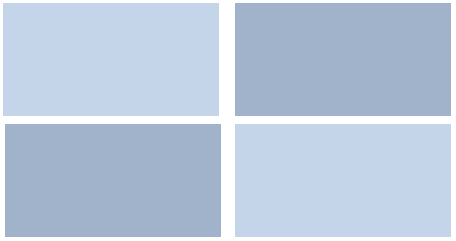
Créer une matrice de distances : cas asymétrique

Remplacer le RETURN out par:

```
zero = array[]::double precision[];
FOREACH k IN ARRAY i
LOOP
          zero = array_append(zero, 1000000000::double precision);
END LOOP;

a=0;
b=0;

FOREACH k IN ARRAY i
LOOP
          nut = array[]::double precision[];
          a   = a+1;
          b   = 0;
          FOREACH m IN ARRAY j
          LOOP
                    b   = b+1;
                    nut = array_append(nut, out[a][b]);
          END LOOP;
          nut  = zero || nut;
          qut := qut || ARRAY[nut];
END LOOP;
```

Créer une matrice de distances : cas asymétrique

```
a=0;
b=0;

FOREACH k IN ARRAY i
LOOP
          nut = array[]::double precision[];
          a   = a+1;
          b   = 0;
          FOREACH m IN ARRAY j
          LOOP
                    b   = b+1;
                    nut = array_append(nut, out[b][a]);
          END LOOP;
          nut  = nut || zero;
          qut := qut || ARRAY[nut];
END LOOP;

RETURN qut;
```



Doublage de la liste (pour affichage du path dans le cas asymétrique)

```
CREATE TEMPORARY TABLE duplicated_random_nodes
AS
(
          SELECT id, s
          FROM
          (
                    SELECT id, 1 as s FROM random_nodes
                    UNION
                    SELECT id, 2 as s FROM random_nodes
          ) as a
          ORDER BY s, id
);
```

Optimisation proprement dite

```
CREATE TABLE urbadm_half_tsp AS
(
            SELECT a.seq, a.id as a_id, b.id, n1.geom
            FROM pgr_tsp(m_distance(ARRAY(SELECT id FROM random_nodes)::integer[],
                                    ARRAY(SELECT id FROM random_nodes)::integer[]),
                    4 ) AS a
            INNER JOIN
            (
                        SELECT id, -1+row_number() over(ORDER BY s,id) as r
                        FROM duplicated_random_nodes
             ) as b
            ON b.r = a.id

            INNER JOIN urbadm_sn_half as n1 on n1.id = b.id
            ORDER BY a.seq
);
```

partant du dernier noeud de la liste !!!! SHOULD BE (N-1) !!!!! BEGINS WITH 0

Edit Data - PostgreSQL 9.3 (localhost:5432) - aabc - urbadm_half_ts

File Edit View Tools Help

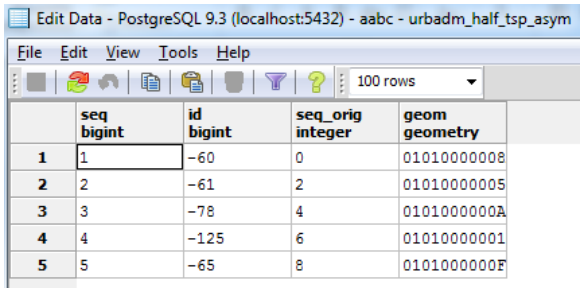
100 rows

	seq integer	a_id integer	id bigint	geom geometry
1	0	4	-60	01010000008
2	1	9	-60	01010000008
3	2	3	-61	01010000005
4	3	8	-61	01010000005
5	4	1	-78	0101000000A
6	5	6	-78	0101000000A
7	6	0	-125	01010000001
8	7	5	-125	01010000001
9	8	2	-65	0101000000F
10	9	7	-65	0101000000F

! **row_number** commence à 1, tandis que **a.id** commence à 0 (= premier noeud, tel que retourné par pgr_tsp) !

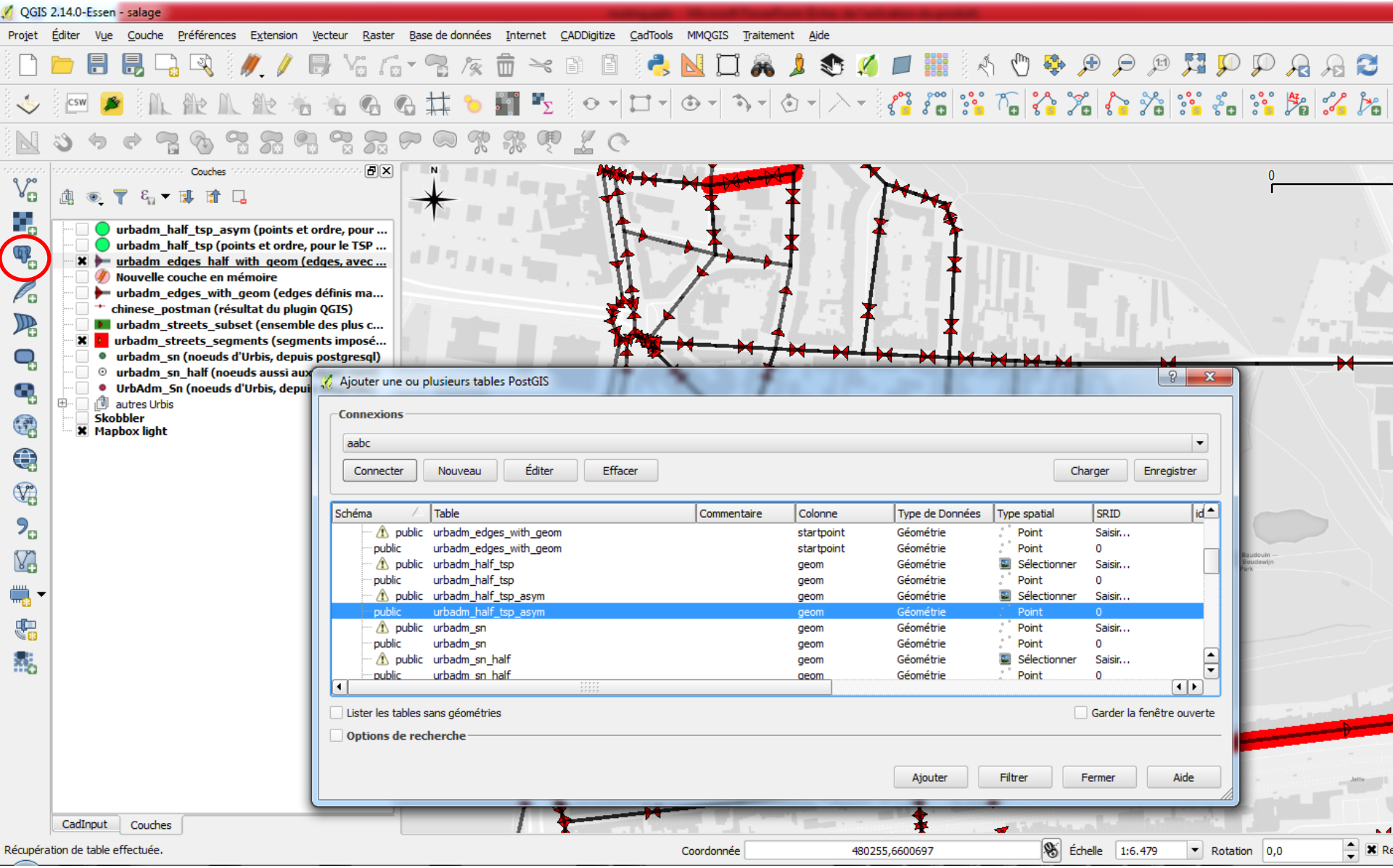
Prendre un noeud sur 2 (car avaient été dupliqués)

```
CREATE TABLE urbadm_half_tsp_asym AS
(
        SELECT row_number() over (ORDER BY seq_orig) as seq, id, seq_orig, geom
        FROM
        (
                SELECT id, geom, min(seq) as seq_orig FROM  urbadm_half_tsp
                GROUP BY id, geom
        ) as a
);
```



	seq bigint	id bigint	seq_orig integer	geom geometry
1	1	-60	0	01010000008
2	2	-61	2	01010000005
3	3	-78	4	0101000000A
4	4	-125	6	01010000001
5	5	-65	8	0101000000F

Affichage en QGIS



Et pour le déneigement, c'est pareil ?

“**Arc** routing”

Problème du **postier chinois**

« trouver un plus court chemin dans un graphe connexe non orienté qui passe au moins une fois par chaque arête du graphe et revient à son point de départ »

fr.wikipedia.org/wiki/Probl%C3%A8me_du_postier_chinois

Problème du postier chinois **rural**

idem, mais seulement une partie des arêtes

Problème du postier chinois rural **dirigé**

idem + sens uniques

Problème du postier chinois rural **mixte**

https://www-m9.ma.tum.de/games/mcpp-game/index_en.html

https://en.wikipedia.org/wiki/Route_inspection_problem

Pas de solutions clé sur porte, mais qqs pointeurs

Rien en **pg_routing** (mais ce serait cool)

QGIS : Extension : 'Chinese Postman Solver' (Ralf Kistner)

- Réseau d'arêtes non orientées
- Sélectionner avec l'outil 
- Plugins -> chinese postman

-> renvoie le chemin pour tout couvrir

[démo]

Et si orienté ?

Si le graphe est **eulérien** (il existe un circuit où chaque arête est visitée une et une seule fois), alors **OK**

https://fr.wikipedia.org/wiki/Probl%C3%A8me_des_sept_ponts_de_K%C3%B6nigsberg

- **Python + networkx**

```
import networkx as nx
import psycopg2

G = nx.DiGraph()
[…]

cur.execute("       SELECT a.gid, a.source, a.target, dist FROM urbadm_edges_with_geom as a
                    INNER JOIN urbadm_streets_subset as b ON a.gid = b.gid;")
ret=cur.fetchall()

# build the network
for r in ret:
 print (r[0], r[1], r[2], r[3])
 G.add_edge(r[1], r[2], weight=r[3])

ise = nx.is_eulerian(G)
print('IS_EULERIAN', ise)

ec = nx.eulerian_circuit(G)
```

Très (trop) restrictif !

http://orion.journals.ac.za/pub/article/viewFile/17/17

En résumé

- Import de données d'OSM : osm2po depuis geofabrik.de + psql
- Voyageur de commerce asymétrique : étendre la matrice + pgr_tsp()
- Déneigement :
  - ≠ **voyageur de commerce**
  - QGIS : Plugin Chinese Postman
  - Python + networkx si Eulérien
  - Développements à faire sur base de la littérature ?

Codes & données disponibles dès demain

http://www.my-poppy.eu/2016-foss4Gfr/2016_foss4G_FR_rtsp_cp_tuto.zip

https://www.flickr.com/photos/hitchster/5504806327/sizes/o/

Une autre ressource à examiner ?
(suite aux questions)

-> OSMR

https://www.flickr.com/photos/hitchster/5504806327/sizes/o/

Un tuto intéressant sur le routing :

http://www.postgis.fr/chrome/site/docs/workshop-routing-foss4g/docs/pgRoutingWorkshop.pdf

https://www.flickr.com/photos/hitchster/5504806327/sizes/o/

https://www.flickr.com/photos/hitchster/5504806327/sizes/o/