

Éléments de recherche opérationnelle 1

- Agoulif Youssef
- Cardi Julien
- Mélanger Alexandre
- Ferroni Sandro



Plan

- Introduction
- Réflexion
- Super Drone
- Déneigeuses
- Limites
- Quel aurait été la suite ?
- Conclusion
- Annexes

Introduction



Réflexion

Parcours (non-orienté) :

- DFS: parcours en profondeurs
- Fleury
- Hierholzer

Déneigeuses :

- Commencer avec une déneigeuse de type 1
- Comparer avec type 2
- plusieurs déneigeuses du même type
- plusieurs déneigeuses de type différent

=> Solution plus rapide, solution la moins chère et un juste-milieu

Super Drone

► Parcours : non-orienté



► Constantes :

Super Drone :

- Coût fixe : 100 €/jour
- Coût kilométrique : 0.01 €/km

Longueur des routes de la ville

► Variables :

Longueur parcourue par le drone

Prix d'utilisation du drone

Le parcours du drone

Super Drone

```
graph = ox.graph from place( "Outremont,  
Montréal, Canada", network type='drive')
```

```
graph temp = graph copy.to undirected()  
graph euler = nx.eulerize(graph temp)  
circuit = list(nx.eulerian circuit(graph euler))
```



Chemin affiché :

```
[(29239079, 4504112773),  
(4504112773, 4504112778),  
... ,  
(31700818, 124524582),  
(124524582, 29239079)]
```



Super Drone

Lieux :	Temps d'exécution :	Longueur totale des routes :	Kilomètre parcouru par le drone :	Prix d'utilisation :
Outremont	5 secondes	44 km	56 km	100.57 €
Verdun	5 secondes	72 km	82 km	100.82 €
Saint-Léonard	30 secondes	182 km	231 km	102.32 €
Rivière-des-prairies-p ointe-aux-trembles	13 minutes	428 km	545 km	105.45 €
Le Plateau-Mont-Royal	10 secondes	134 km	156 km	101.56 €

Super Drone

Montréal :

- Temps d'exécution : + 48 h
- Longueur totale des routes de Montréal : 4299 km
- Projection des km parcourus par le drone : 5460 km
- Prix : 154.60€

⇒ Liée à la fonction : eulerize

⇒ Un réel problème ?

Résultats :

- Outremont
- Verdun
- Saint-Léonard
- Rivière-des-prairies-pointe-aux-trembles
- Le Plateau-Mont-Royal

Déneigeuses

► Parcours : orienté

► Constantes :

- Véhicule 1 :
 - Coût fixe : 500 €/jour
 - Coût kilométrique : 1.1 €/km
 - Coût horaire les 8 premières heures : 1.1 €/h puis 1.3 €/h
 - Vitesse moyenne : 10 km/h
- Véhicule 2 :
 - Coût fixe : 800€/jour
 - Coût kilométrique : 1.3 €/km
 - Coût horaire les 8 premières heures : 1.3 €/h puis 1.5 €/h
 - Vitesse moyenne : 20 km/h
- Longueur des routes des quartiers
- Distance et nombre de déneigeuses dans les entrepôts

► Variables :

- Longueur parcourue par les déneigeuses
- Prix d'utilisation des déneigeuses
- Le temps de parcours des déneigeuses
- Le nombres de déneigeuses
- Le parcours des déneigeuses

Déneigeuses

Comme le drone :

```
circuit_non_oriente = list(nx.eulerian_circuit(graph_euler))
```

Nœud de départ : choisi en fonction de l'entrepôt le plus proche

Entrepôts	Saint-Dominique	Biodôme/Planétarium	Sherbrooke	Des Trinitaires
Nombre de déneigeuses	43	400	113	735

Déneigeuses

```
north = 45.530  
south = 45.480  
east = -73.570  
west = -73.640  
graph montreal = ox.graph from bbox(north, south,  
east, west, network_type='drive')
```

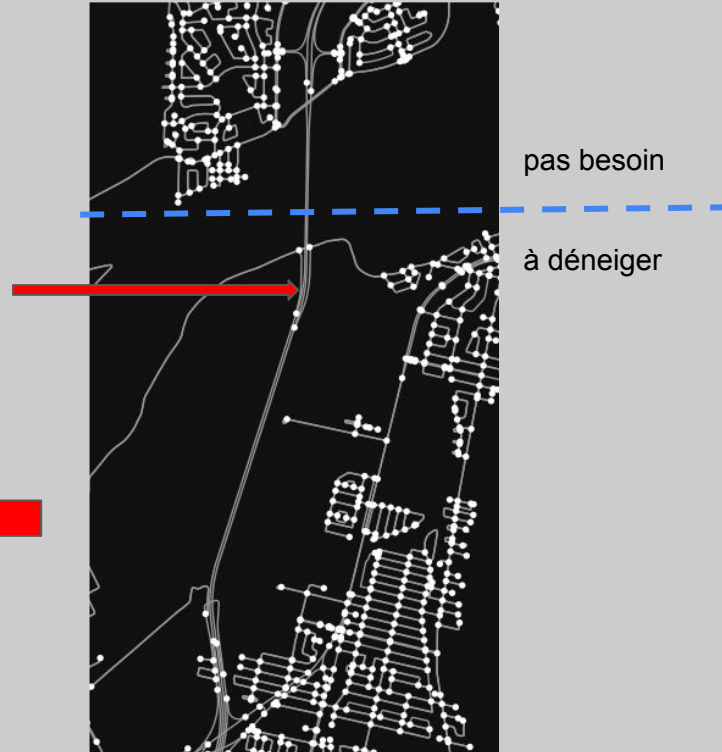
```
route = nx.shortest_path(graph montreal,  
source=depart, target=arrive, weight='length')
```



Déneigeuses

Problème pour :
Rivière-des-prairies-pointe-aux-trembles

Autoroute à déneiger ? OUI



Déneigeuses

1 déneigeuse type 1 :

Lieux :	Temps d'exécution :	Longueur totale des routes :	Kilomètre parcouru : (entrepôt compris)	Prix d'utilisation :	Temps de déneigement :
Outremont	5 secondes	44 km	124 km	652.12 €	13 h
Verdun	6 secondes	72 km	180 km	720.13 €	18 h
Saint-Léonard	45 secondes	182 km	531 km	1152.14 €	53 h
Rivière-des-prairies-pointe-aux-trembles	14 minutes	428 km	1507 km	2353.11 €	150 h
Le Plateau-Mont-Royal	10 secondes	134 km	528 km	1148.17 €	53 h

Déneigeuses

Comparaison type 1 et type 2 :

Lieux :	Prix d'utilisation : Type 1	Prix d'utilisation : Type 2	Temps de déneigement : Type 1	Temps de déneigement : Type 2
Outremont	652.12 €	970.59 €	13h	6h15
Verdun	720.13 €	1047.86 €	18h	9h
Saint-Léonard	1152.14 €	1524.69 €	53h	26h30
Rivière-des-prairies-pointe-aux-trembles	2353.11 €	2954.77 €	150h	78h30
Le Plateau-Mont-Royal	1148.17 €	1539.48 €	53h	27h

Déneigeuses

circuit_euler = [(a,b) , (c,d) , (e,f) , ... , ... , (w,x) , (y,z) , (z,a)]


L1
L2
L3
L4


circuit_machine2 = chemin(a,e) + L2 + chemin(L2[-1],a)

Comparaison 1 type 1 et 4 type 1 :



Lieux :	Prix d'utilisation :	Prix d'utilisation :	Temps de déneigement :	Temps de déneigement :	km parcouru :	km parcouru :
Outremont	652.12 €	2173.65 €	13h	4h	124 km	143 km


 × 3,33


 ÷ 3,33

Limites

- Modélisation de la réalité abusive :
- Complexité temporelle de la solution :
 - Réel problème ?
- Un parcours de déneigeuses à améliorer :

Quel aurait été la suite ?

Script shell :

- Implémenter correctement plusieurs déneigeuses
- Les meilleures solutions

Drone :

- Implémenter un algorithme plus rapide à l'exécution

Déneigeuses :

- Meilleur algorithme (spécifique à l'orienté)
- Meilleur découpage pour multi-déneigeuses (sous-graphes Hierholzer)
- Déneigeuses de types différents sur un même quartier

Conclusion



Annexes

```
def eulerize(G):
    if G.order() == 0:
        raise nx.NetworkXPointlessConcept("Cannot Eulerize null graph")
    if not nx.is_connected(G):
        raise nx.NetworkXError("G is not connected")
    odd_degree_nodes = [n for n, d in G.degree() if d % 2 == 1]
    G = nx.MultiGraph(G)
    if len(odd_degree_nodes) == 0:
        return G

    # get all shortest paths between vertices of odd degree
    odd_deg_pairs_paths = [
        (m, {n: nx.shortest_path(G, source=m, target=n)})
        for m, n in combinations(odd_degree_nodes, 2)
    ]

    # use the number of vertices in a graph + 1 as an upper bound on
    # the maximum length of a path in G
    upper_bound_on_max_path_length = len(G) + 1

    # use "len(G) + 1 - len(P)",
    # where P is a shortest path between vertices n and m,
    # as edge-weights in a new graph
    # store the paths in the graph for easy indexing later
    Gp = nx.Graph()
    for n, Ps in odd_deg_pairs_paths:
        for m, P in Ps.items():
            if n != m:
                Gp.add_edge(
                    m, n, weight=upper_bound_on_max_path_length - len(P), path=P
                )
```

```
def to_undirected(self, as_view=False):
    graph_class = self.to_undirected_class()
    if as_view is True:
        return nx.graphviews.generic_graph_view(self, graph_class)
    # deepcopy when not a view
    G = graph_class()
    G.graph.update(deepcopy(self.graph))
    G.add_nodes_from((n, deepcopy(d)) for n, d in self._node.items())
    G.add_edges_from(
        (u, v, deepcopy(d))
        for u, nbrs in self._adj.items()
        for v, d in nbrs.items()
    )
    return G
```

```
# find the minimum weight matching of edges in the weighted graph
best_matching = nx.Graph(list(nx.max_weight_matching(Gp)))

# duplicate each edge along each path in the set of paths in Gp
for m, n in best_matching.edges():
    path = Gp[m][n]["path"]
    G.add_edges_from(nx.utils.pairwise(path))
return G
```


Annexes

```
def eulerian_circuit(G, source=None, keys=False):
    if not is_eulerian(G):
        raise nx.NetworkXError("G is not Eulerian.")
    if G.is_directed():
        G = G.reverse()
    else:
        G = G.copy()
    if source is None:
        source = arbitrary_element(G)
    if G.is_multigraph():
        for u, v, k in _multigraph_eulerian_circuit(G, source):
            if keys:
                yield u, v, k
            else:
                yield u, v
    else:
        yield from _simplegraph_eulerian_circuit(G, source)
```

Utilisation OSMnx :

Boeing, G. 2017. [OSMnx: New Methods for Acquiring, Constructing, Analyzing, and Visualizing Complex Street Networks](#). *Computers, Environment and Urban Systems* 65, 126-139.
doi:10.1016/j.compenvurbsys.2017.05.004