Exploration algorithmique d’un problème

Analyse comparative des performances d'algorithmes sur différents environnements

JAUFRET BOURGUET

Jaufret Bourguet

Algorithme

Algorithme de recherche local :

L'algorithme de recherche locale a un fonctionnement unique. Il consiste à rechercher toujours une meilleure tournée que celle initiale parmi les voisins de la tournée. Si la tournée voisine est meilleure que la tournée initiale alors on inverse la tournée voisine avec la tournée initiale.

Complexité : $O(n^3)$

Algorithme Heuristique de plus proche voisin :

L'heuristique du plus proche voisin consiste à visiter tous les sommets qui ne sont pas encore visités puis de refaire l'opération suivante se rendre au sommet le plus proche pas encore visité

Complexité : $O(n^2)$

Insertion proche et loin

Insertion au plus proche :

On commence par trouver les deux lieux les plus éloignés l'un de l'autre. Tant qu'il existe des lieux non visités, on cherche le lieu (non visité) dont la distance est la plus petite. On insère ce lieu dans la tournée là où cela augmente le moins la longueur de la tournée.

Complexité : $O(n^3)$

Insertion au plus loin :

Cette heuristique possède une variante qui consiste à ajouter à chaque étape non pas le lieu le plus proche de la tournée mais celui le plus loin de la tournée.

Complexité : $O(n^3)$

L'heuristique du plus proche voisin :

L'heuristique du plus proche voisin est une heuristique gloutonne. Elle consiste à choisir un sommet de départ et visiter son voisin le plus proche, qui n'a pas encore été visité, jusqu'à l'obtention d'une solution complète.

Complexité : $O(n^2)$

Heuristique personnelle :

Algorithme génétique :

La première tournée est créée aléatoirement, ensuite nous créons entre 1 et 10 tournées aléatoire qui vont nous servir de base pour cette heuristique génétique. Nous prenons la meilleure tournée parmi les 10, nous l'ajoutons à la génération suivante. Pour créer une tournée de la génération suivante nous allons prendre les 5 premiers

lieux d'une tournée I et les fusionnées avec les 5 derniers lieux d'une tournée I+1.

Complexité : $O(n^4)$

Graphes choisi :

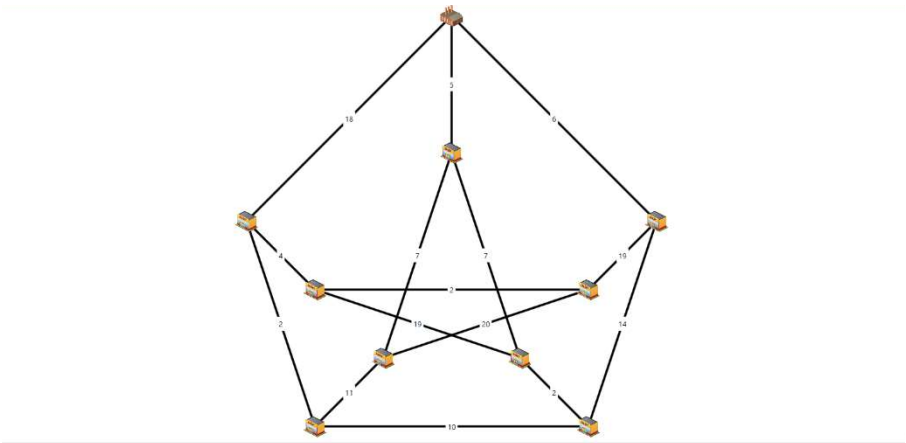

Figure : graphe de Petersen

Le graphe de Petersen est régulier, symétrique, connexe de plus c'est un graphe simple.

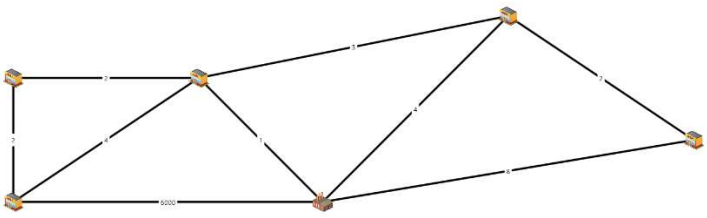

Figure : Graphe simple n°1

Ce graphe est simple et permet de voir l'efficacité des différentes heuristiques sur un parcours moins long.

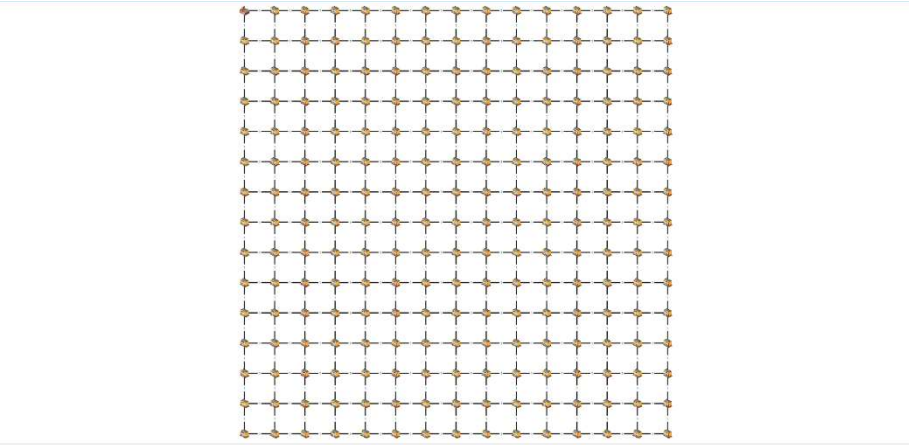

Figure : Graphe simple n°2

Ce graphe est simple et régulier et permet de voir l'efficacités des différentes heuristiques sur un parcours long.

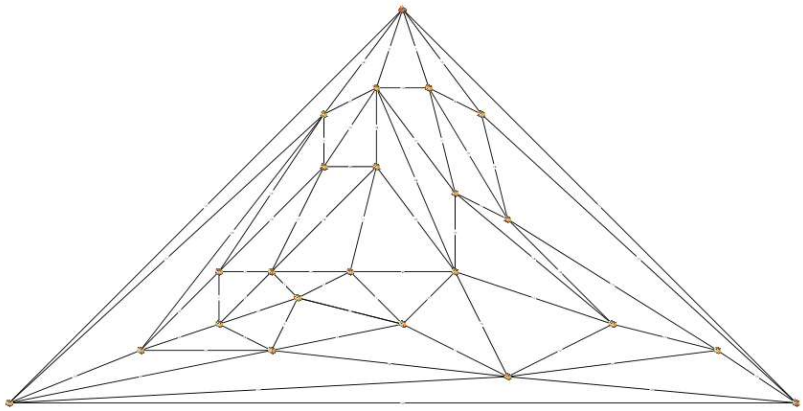

Figure : Graphe de kittel

Deuxième graphe possédant un parcours long mais cette fois non régulier pour tester l'efficacité des différentes heuristiques.

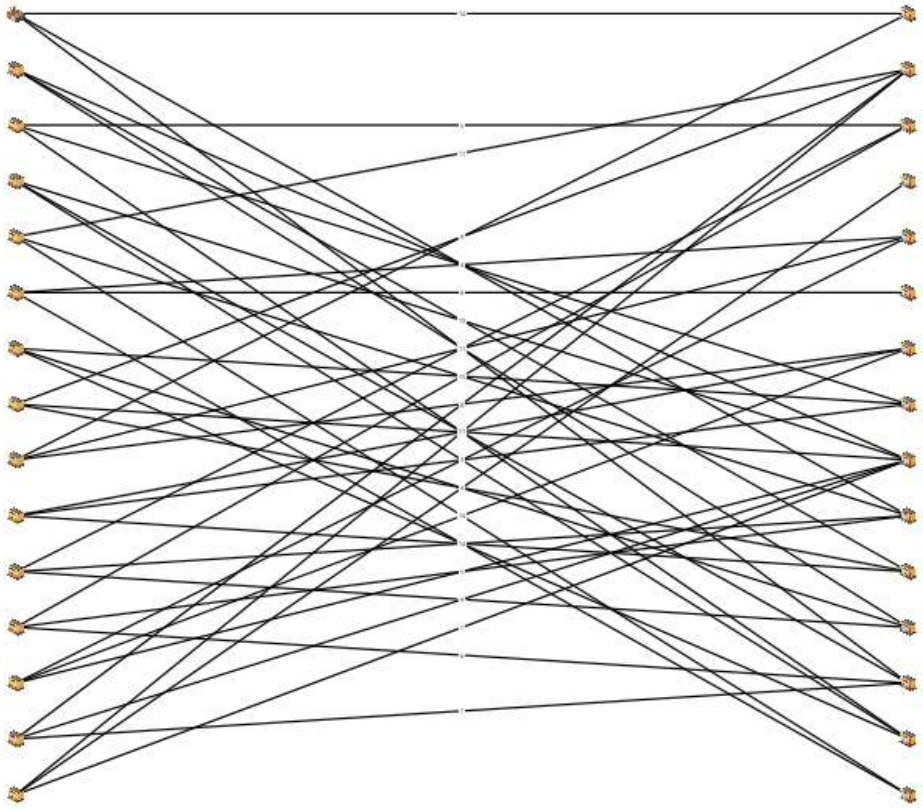

Figure : Graphe Bipartie

Ce graphe bipartie est fait en sorte que les points du premier ensemble soi relié avec les points du second ensemble. Ce graphe est utilisé dans le cas d'optimisation ferroviaires. L'objectif est de trouver un ensemble de gares le plus petit qu'il soit tel que chaque train visite au moins une des gares sélectionnées.

Figure : graphe anti croissant

Le but de ce graphe est de montré que l'heuristique croissant n'est pas optimal dans tous les cas.

Figure : graphe locale

Ce graphe a pour objectif de montrer que l'heuristique de recherche locale peut être l'heuristique la plus efficace.

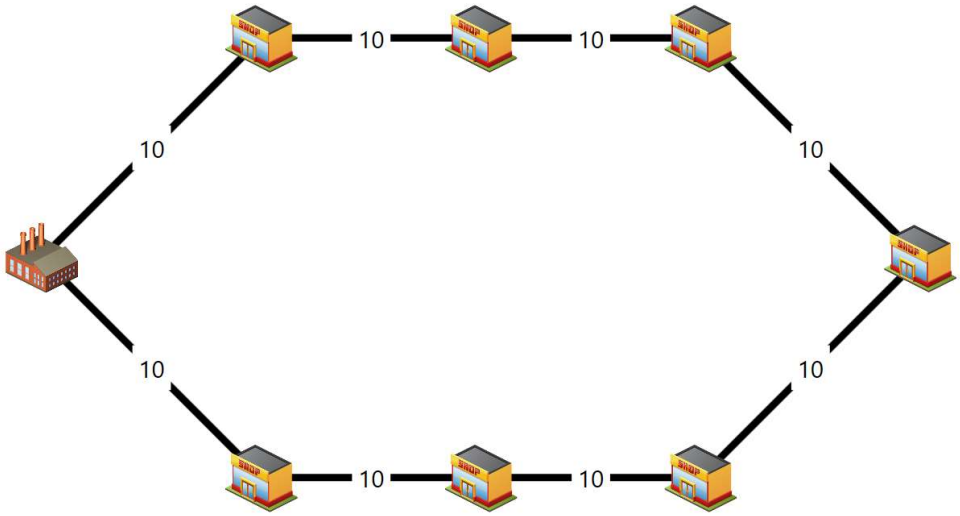

Figure : anti insertion

Le but de ce graphe est de montré que l'heuristique insertion au plus proche et au plus loin n'est pas optimal dans tous les cas.

Etude comparative des heuristiques

Temps d'exécution des algorithmes sur le Graphe de Peterson :

Algorithme	Temps d'exécution	Score
Tournée croissante	0 ms	141
Plus proche voisin	0 ms	73
Insertion au plus proche	3 ms	92
Insertion au plus loin	3 ms	177
Recherche local	24 ms	117
Génétique	2048 ms	≈120

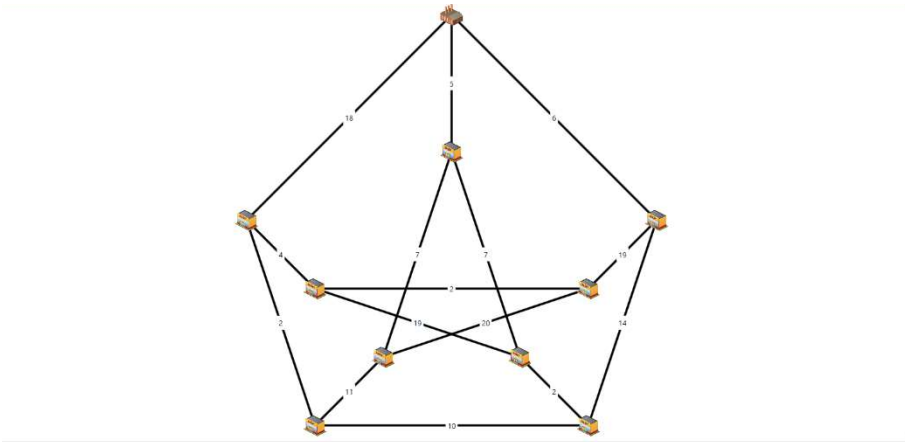

Figure : graphe de Petersen

Pour le graphe de Petersen l'insertion au plus proche voisin est l'heuristique la plus optimal.

Temps d'exécution des algorithmes sur le Graphe simple 1 :

Algorithme	Temps d'exécution	Score
Tournée croissante	0 ms	20
Plus proche voisin	0 ms	20
Insertion au plus proche	0 ms	20
Insertion au plus loin	0 ms	26
Recherche local	0 ms	20
Génétique	679 ms	≈ 20

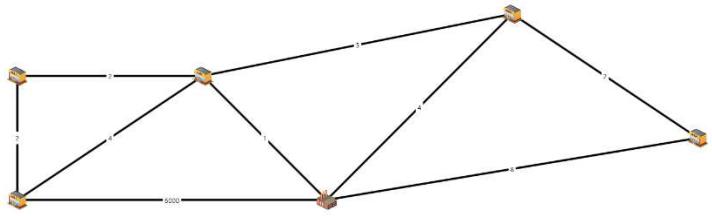

Figure : Graphe simple n°1

Pour le graphe simple n°1 l'insertion au plus proche voisin, l'insertion au plus proche, et la recherche locale sont les heuristiques les plus optimal.

Temps d'exécution des algorithmes sur le Graphe simple 2 :

Algorithme	Temps d'exécution	Score
Tournée croissante	2723 ms	448
Plus proche voisin	2900 ms	252
Insertion au plus proche	3422 ms	448
Insertion au plus loin	2000 ms	1384
Recherche local	1666 ms	448
Génétique	<60 000 ms	No data

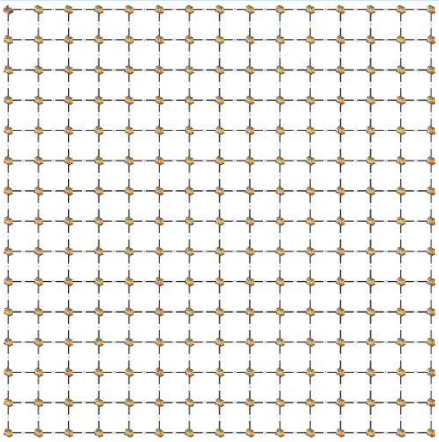

Figure : Graphe simple n°2

Lors de l'exécution de l'algorithme génétique le résultat n'a pas pu aboutir, une exception a été levé car la durée des calculs était supérieure à 60 000 ms.

Le meilleur heuristique dans ce graphique en terme de score est celle du plus proche voisin en un temps raisonnable.

Temps d'exécution des algorithmes sur le graphe de kittel :

Algorithme	Temps d'exécution	Distance
Tournée croissante	2 ms	303
Plus proche voisin	6 ms	165
Insertion au plus proche	6 ms	227
Insertion au plus loin	6 ms	447
Recherche local	6 Ms	261
Génétique	15 538 Ms	≈ 303

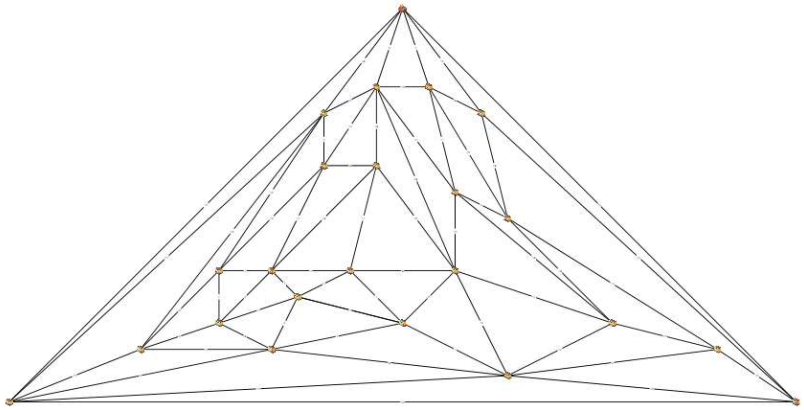

Figure : Graphe de kittel

Le meilleur heuristique dans ce graphique en terme de score est encore celle du plus proche voisin, cependant l'heuristique ayant effectué le meilleur temps est la tournée croissante avec un score de 303 soit presque deux fois plus grands que celle du plus proche voisin.

Temps d'exécution des algorithmes sur le graphe biparti :

Algorithme	Temps d'exécution	Score
Tournée croissante	6ms	687
Plus proche voisin	4ms	274
Insertion au plus proche	4ms	321
Insertion au plus loin	5ms	716
Recherche local	4 ms	604
Génétique	37499 ms	≈ 570

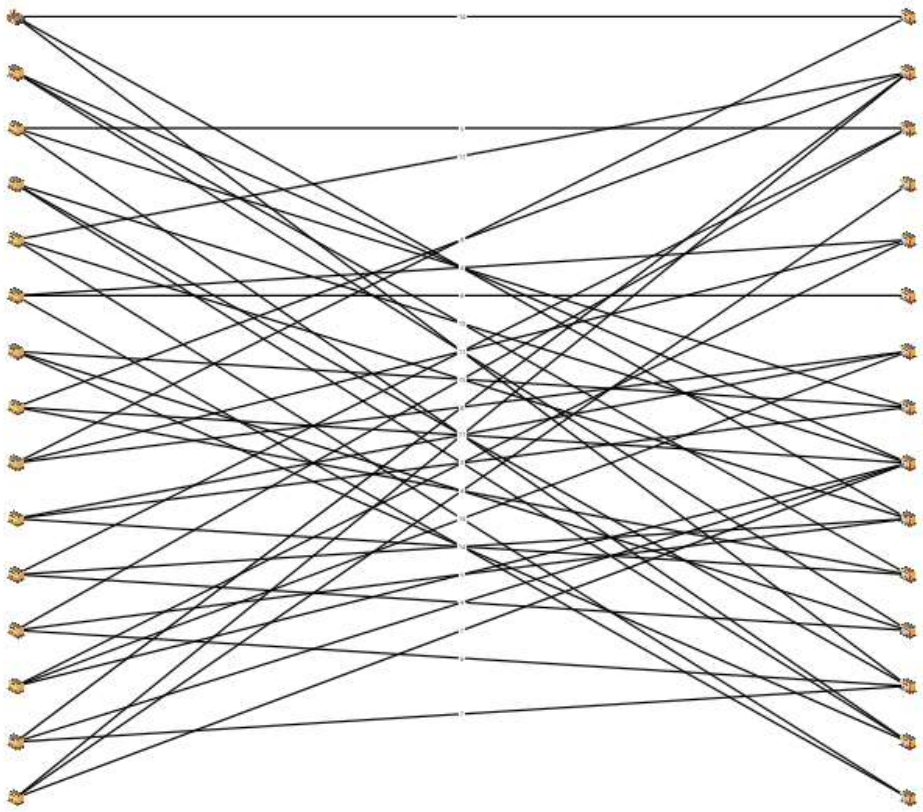

Figure : Graphe Bipartie

Le meilleur heuristique pour ce graphe est l'insertion au plus proche avec le meilleur score et faisant partie de celle ayant réalisé le meilleur temps.

Temps d'exécution des algorithmes sur le graphe anti croissant :

Algorithme	Temps d'exécution	Score
Tournée croissante	0 ms	240
Plus proche voisin	0 ms	120
Insertion au plus proche	0ms	120
Insertion au plus loin	0 ms	240
Recherche local	0 ms	140
Génétique	872 ms	≈ 120

Figure : graphe anti croissant

Ici en terme de score trois algorithme ont réussi a trouvé le score optimal :

- Insertion au plus proche
- Insertion au plus proche voisin
- Et heuristique génétique

Néanmoins l'heuristique génétique n'atteint pas le temps optimal.

Temps d'exécution des algorithmes sur le graphe local :

Algorithme	Temps d'exécution	Score
Tournée croissante	0 ms	160
Plus proche voisin	0 ms	120
Insertion au plus proche	0 ms	120
Insertion au plus loin	0 ms	240
Recherche local	0 ms	120
Génétique	837 ms	≈ 120

Figure : graphe locale

En terme de score quatre algorithme ont réussi a trouver le score optimal :

- Insertion au plus proche
- Insertion au plus proche voisin
- Et heuristique génétique
- Recherche local

Néanmoins l'heuristique génétique n'atteint pas le temps optimal.

Temps d'exécution des algorithmes sur le graphe anti insertion :

Algorithme	Temps d'exécution	Score
Tournée croissante	0 ms	200
Plus proche voisin	0 ms	80
Insertion au plus proche	0 ms	200
Insertion au plus loin	0 ms	240
Recherche local	0 ms	140
Génétique	1337 ms	≈ 140

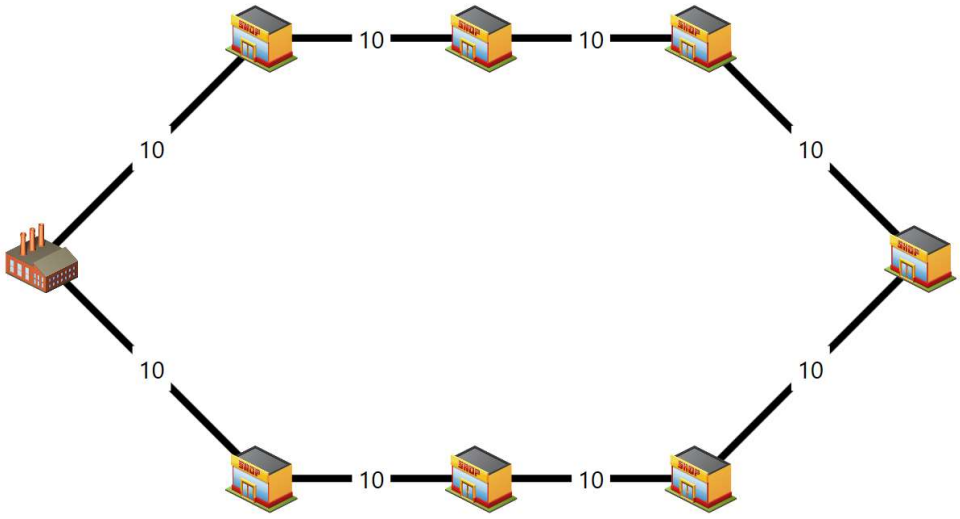

Figure : anti insertion

Sur ce graphe l'algorithme le plus efficace est celui du plus proche voisin avec un score de 80.

Comme prévu les insertions rencontre des difficultés lorsque le graphe contient des cycles.

Conclusion :

Au cours de cette SAE nous avons pu explorer l'efficacité de différentes heuristiques sur différents graphes. Nous avons remarqué que l'heuristique du plus proche voisin est le plus optimal sur tous les graphes. Sur tous les graphes avec ou sans cycle, elle a obtenu les meilleurs scores ou un temps d'exécution plus faible.

On ne peut pas en dire autant pour toute les heuristiques : les heuristiques d'insertion au plus proche et au plus loin, donne des résultats incohérents lorsque le graphe possède des cycles.

La tournée croissante dépend des placements des lieux. Si le nom des lieux est placé de manière aléatoire alors le résultat le saura aussi.

L'heuristique de recherche locale obtient des résultats moyens avec un score moyen mais ne dépends pas des noms des points. C'est un algorithme plutôt stable en terme de résultat, il se place juste après le plus proche voisin.

L'heuristique génétique doit fonctionner dans un graphe petit pour d'assurer de son efficacité et de son bon fonctionnement. Comme les valeurs initiales sont créés aléatoirement le résultat n'est pas fixe.