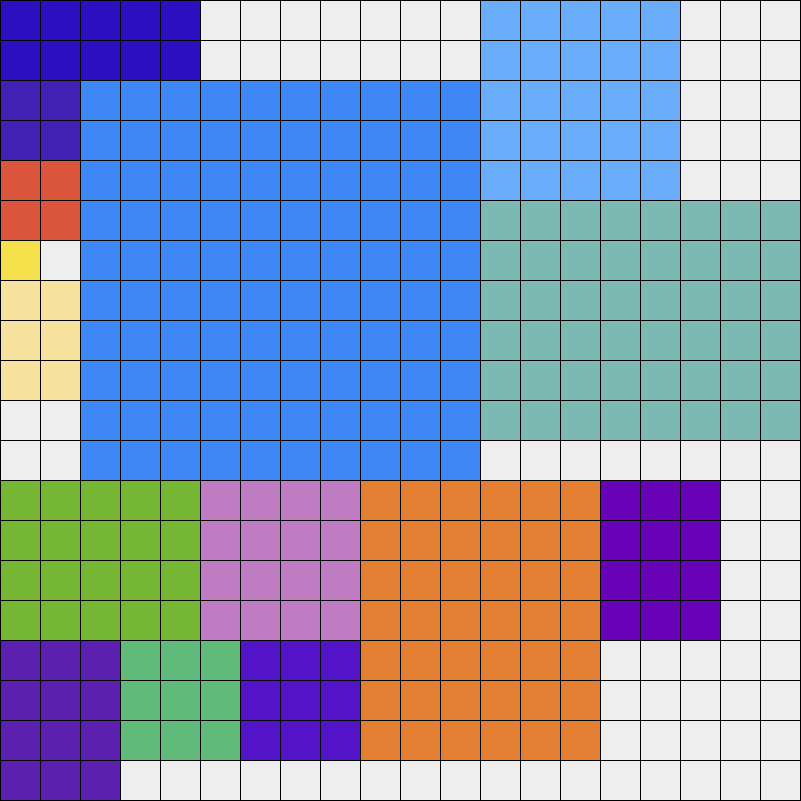
Rapport optimisation linéaire et combinatoire

Projet :

Implémentation d’algorithmes d’optimisation des placements de bâtiments dans une ville.



Etudiants :

-Maxence Marot

-Quentin Maignan

Tables des matières :

Table des matières

[1 – Présentation du problème et de l’objectif à optimiser 3](#_Toc69648189)

[A – Notre problèmes 3](#_Toc69648190)

[B – Notre objectif 3](#_Toc69648191)

[2 – Présentation des algorithmes de résolution 3](#_Toc69648192)

[A – Links 3](#_Toc69648193)

[B – Glouton 3](#_Toc69648194)

[C – Les variantes du glouton 3](#_Toc69648195)

[D – Best glouton 3](#_Toc69648196)

[3 – Etudes des résultats 3](#_Toc69648197)

[A – les résultats du glouton « de base » 3](#_Toc69648198)

[B – les résultats du best glouton 3](#_Toc69648199)

[4 – Discussion / Conclusion 3](#_Toc69648200)

[A – Ce qu’il nous manque 3](#_Toc69648201)

[B – Ce qu’on a apprit 3](#_Toc69648202)

# 1 – Présentation du problème et de l’objectif à optimiser

## A – Notre problèmes

Ce projet d’optimisation consiste à placer des bâtiments, qui sont représentés par des positions de départs en X et Y et de longueur et largeur prédéfini, dans une ville, représenté par une matrice d’entier ou pour chaque nous avons le numéro du bâtiment qui occupe cette case ou alors 0 si aucun bâtiment n’est sur la case. Nous avons pour contrainte que tous les bâtiments soit relié avec le bâtiment numéro 1 qui représente la mairie de la ville.

Nous disposons de 5 instances de ville dans lesquelles la taille de la ville ainsi que tous les bâtiments nous sont imposées.

## B – Notre objectif

Notre objectif est de maximiser l’air total des bâtiments dans la ville en veillant à ce que tous les bâtiments soient bien relier avec la marie. Nous allons donc pouvoir calculer un score de ville qui sera la somme des airs des bâtiments placés dans la ville. Plus ce score sera élevé plus notre algorithme aura été performant.

# 2 – Présentation des algorithmes de résolution

## A – Links

Ce premier algorithme n’est pas un algorithme de résolution mais il va nous permettre de vérifier que tous les bâtiments sont bien reliés à la mairie. Cet algorithme va explorer les cases adjacentes à notre mairie pour voir si tous les bâtiments qui sont placés sur les cases de la ville sont atteignable par la route ou alors directement collé à notre mairie.

## B – Glouton

Cet algorithme dit glouton est le cœur de notre projet, En effet c’est celui-ci qui va réaliser le placement des bâtiments dans la ville en respectant la contrainte de connexion entre les bâtiments et la marie.

Pour ce faire cet algorithme nous allons itérer sur toutes les cases de notre matrice. Dès qu’une case sera marquée comme vide matrice[y][x] == 0, nous allons essayer de placer un bâtiment (le premier de notre liste de bâtiments qui puisse rentrer dans cette case, c’est-à-dire qu’il ne sorte pas des bornes de la ville et qu’il ne soit pas non plus sur un autre bâtiment). Une fois un bâtiment placé nous allons regarder s’il n’a pas détruit de liaisons avec d’autre bâtiments précédemment placés, (s’il a détruit des liens alors nous allons juste passer à un autre bâtiment). Si le bâtiment à bien été placé alors nous passons simplement à la case suivante. De fait aucun bâtiment bien placé ne sera déplacé ou retiré.

## C – Les variantes du glouton

Nous avons 4 variantes du glouton. C’est 4 variantes sont appelées dans la même méthode glouton. Pour les différencier nous avons rajouté un attribut de méthode qui va pouvoir effectuer une trie spécifique avant de réellement commencer le glouton.

**if**(sortType.equals("air")) : alors nous trions nos bâtiments pour que les bâtiments qui compte le plus de cases soit en premier dans la liste des bâtiments. De ce fait les plus gros bâtiments seront placé en premier dans la ville.

**else** **if**(sortType.equals("congestion")) :alors nous trions nos bâtiments par congestion c’est-à-dire que les bâtiments dont la longueur + largueur est la plus élevé seront placé en premier

**else** **if**(sortType.equals("random")) et finalement un dernier trie dit random, pour que les bâtiments soit trié de façon aléatoire.

## D – Best glouton

Cet algorithme est dit le meilleur car il va itérer sur tous les placements possibles de la mairie avant de faire le glouton pour voir avec quelle position de la mairie le placement est le meilleur pour obtenir un score le plus élevé possible.

Nous pouvons appeler best Glouton avec les 4 types de glouton précédemment défini.

# 3 – Etudes des résultats

## A – les résultats du glouton « de base »

## 

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Tri=none | Test 1 | Test 2 | Test 3 | Test 4 | Test 5 | av. | Best Glouton |
| Instance 1 max 251 | 231 | 217 | 225 | 231 | 215 | 223,8 | 251 |
| Instance 2 max 456 | 320 | 326 | 309 | 322 | 326 | 320,6 | 344 |
| Instance 3 max 804 | 796 | 796 | 782 | 819 | 796 | 797,8 | 829 |
| Instance 4 max 579 | 382 | 368 | 355 | 358 | 368 | 366,2 | 385 |
| Instance 5 max 620 | 358 | 358 | 349 | 358 | 358 | 356,2 | 367 |

Dans ce tableau vous pouvez voir nos scores pour l’algorithme du glouton en mode « sans tri ». Nous avons 5 tests pour chaque instance pour avoir une moyenne de plusieurs runs de notre algorithme vu que la marie est placée aléatoirement dans le coin supérieur gauche.

Nous pouvons voir que notre algorithme Best glouton est toujours le meilleur. En effet, c’est normal puisque le max de nos tests est une borne inferieur à notre algorithme du best glouton.

# 4 – Discussion / Conclusion

## A – Ce qu’il nous manque

Pour ce projet d’optimisation, il nous manque l’algorithme de « Branch and Bound ». Malheureusement, il nous a manqué un peu de temps pour l’implémenter.

Nous avons aussi quelques problèmes d’algorithme telle que le links qui ne marche pas comme voulu de temps en temps et nous n’avons pas su corriger le problème. Tous nos tris de bâtiments qui sont censé représenter les 3 autres variantes du glouton ne marche pas non plus, pour une mystérieuse raison, ou raison mystérieuse à vous de choisir.

Mise à part ces deux petits soucis nous avons réussi à faire un projet plus ou moins complet et une interface graphique plutôt ludique.

## B – Ce qu’on a apprit

Avec ce projet nous avons pu, pour la première fois, réaliser un algorithme d’optimisation. Nous nous sommes rendu compte que dans un contexte simple comme celui de projet ce genre d’algorithme peut être facile à mettre en place et à la fois puissant. Demain si nous avons un problème de ce genre nous serions plus enclins à implémenter cet algorithme glouton, et pourquoi pas si le temps le permet cette fois ci, le « Branch and Bound »