Blampain Maxime
Fourny Nicolas

Projet Algorithmique Avancée

Algorithmes gloutons pour le placement de bâtiments

Master 1 I2L

2020-2021

Sommaire

I. Description du problème

Nous allons dans un premier temps reformuler le problème : Nous devions à l'aide d'algorithmes gloutons replacer tous les bâtiments sur un terrain, les valeurs sont initialisées à travers une instance. Nous avions plusieurs instances à mettre en place pour analyser les résultats.

Pour ce faire, nous avons découpé ce problème en plusieurs parties. Nous commencerons par la lecture de l'instance, l'implémentation des algorithmes puis par l'évaluation des algorithmes et enfin l'affichage de la solution.

a. Lecture de l'instance

Dans un premier temps, nous avons dû interpréter le fichier d'instance donné pour pouvoir commencer le travail sur les algorithmes. Pour cela, nous avons implémenté une méthode nommée "read" pour lire ce fichier. Le principe de cette méthode est simple :

- Si on peut ouvrir le fichier Alors …
    - On ouvre le fichier
    - Tant qu'il reste une ligne dans ce fichier Alors ...
        - Si la ligne sélectionnée est la 1ère ligne Alors …
            - On récupère la taille du terrain
        - Si la ligne sélectionnée est la 2ème ligne Alors …
            - On l'ignore
        - Sinon ...
            - On instancie un objet "Building" avec comme paramètres un id unique ainsi que la taille en X et en Y récupéré depuis le ligne actuelle
            - On ajoute cet objet dans le tableau dédié à cet effet
        - Fin de boucle SI
    - Fin de boucle TANT QUE
    - On ferme le fichier
- Sinon …
    - On informe que le fichier n'est pas ouvrable
- Fin de boucle SI

Cette méthode permet donc d'initialiser la taille du terrain, ainsi qu'un tableau contenant tous les bâtiments avec leurs dimensions et leurs id pour pouvoir les reconnaître.

b. Evaluation de l'algorithme

Pour calculer l'efficacité des algorithmes, nous avons compté chaque case vide ou chaque case occupée par une route pour se rendre compte si l'algorithme permettait d'organiser au mieux les bâtiments en comptabilisant le moins de route et/ou de vide possible. Pour ce faire, voici le principe de cette fonction d'évaluation :

- Initialisation du compteur d'évaluation à 0
- Pour chaque ligne du terrain, Alors …
    - Pour chaque colonne de la ligne, Alors …
        - Si la case du terrain est un bâtiment, Alors …
            - On implémente de 1 le compteur d'évaluation
        - Fin de boucle SI
    - Fin de boucle POUR
- Fin de boucle POUR
- On affecte le compteur d'évaluation à la fitness de l'algorithme

c. Affichage de la solution

Après s'être chargés d'interpréter les instances de données et calculer l'efficacité des algorithmes, nous nous sommes penchés sur la façon dont nous allions la faire apparaître dans la console. Pour cela, nous avons créé la méthode "print" qui peut être résumée comme ceci :

- Si la fitness est différente de 0, Alors …
    - Afficher la fitness
- Fin de boucle SI
- Pour chaque ligne du terrain, Alors …
    - Pour chaque colonne de la ligne, Alors …
        - Afficher l'id du bâtiment pour qui occupe la case du terrain
    - Fin de boucle POUR
    - Faire un retour à la ligne.
- Fin de boucle POUR

Grâce à cette méthode, le terrain est modélisé depuis la console pour pouvoir visualiser la disposition de chaque bâtiment sur le terrain.

II. Les algorithmes implémentés

Pour résoudre le problème de placement des bâtiments, il a fallu implémenter plusieurs algorithmes pour trouver le plus efficace et ainsi avoir tous les bâtiments reliés avec le moins de route et de vide possible.

a. L'ordre généré par défaut

Ce premier algorithme consiste à placer les bâtiments selon leur position initiale dans l'instance. Nous avons apporté une amélioration, dans ce cas si le bâtiment ne passe pas dans sa position d'origine nous effectuons une rotation de 90° pour savoir s'il est positionnable ou non. Voici la description de l'algorithme :

- Initialiser l'index à 0
- Pour chaque ligne du terrain, Alors …
    - Pour chaque colonne de la ligne, Alors …
        - Si il reste un bâtiment à placer, Alors …
            - Si il n'y a pas assez de place pour le bâtiment, Alors …
                - Tourner le bâtiment à 90°
            - Fin de boucle SI
            - Si il y a assez de place pour le bâtiment et si on peut placer une route, Alors …
                - Placer le bâtiment sur le terrain
                - Modifier l'état du bâtiment à "Positionné"
                - Changer de bâtiment
            - Fin de boucle SI
        - Fin de boucle SI
    - Fin de boucle POUR
- Fin de boucle POUR

b. L'ordre selon l'air maximum et minimum

Dans un second temps, en rapport avec une des questions du sujet, nous nous sommes intéressés à modifier l'ordre en commençant par les bâtiments selon le bâtiment avec l'aire la plus grande jusqu'à la plus petite. Nous les avons placés avec la même logique que dans le précédent algorithme. Voici le pseudo code pour cette autre algorithme :

- Copier la liste de bâtiment dans une autre liste “from”
- supprimer tous les bâtiments de la liste de base
- Pour chaque bâtiment de la liste “from”, Alors …
    - Récupérer le bâtiment qui a l’air la plus grande
    - Ajouter le bâtiment dans la liste de base
    - Supprimer le bâtiment de la liste “from”
- Fin de boucle POUR
- Effectuer le placement comme pour le 1er algorithme

c. L’ordre d’encombrement selon le maximum et le minimum

Selon le même principe que pour l’air, cette fois-ci la variable de décision est l'encombrement du bâtiment sur le terrain, auquel on affecte le placement sur le terrain de la même manière que le premier algorithme. Le pseudo code ne sera pas détaillé car il réside dans le même principe que celui de l’air maximale.

d. L’ordre aléatoire

Pour la variante aléatoire, nous avons simplement fait une fonction qui vient placer les bâtiments dans un ordre aléatoire puis nous les plaçons de la même manière que les précédents algorithmes. Le principe de cette fonction est simple, il faut créer une copie de la liste des bâtiments, puis aléatoirement nous allons récupérer un id puis nous les ajoutons l’un à la suite de l’autre dans une nouvelle liste pour ensuite la placer sur le terrain dans le nouvel ordre.

e. L’ordre de l’air selon le minimum et le maximum

Pour la dernière variante glouton, il s’agit de faire l’inverse que l’aire maximale tout simplement en parcourant la liste et en la classant du bâtiment avec l'aire minimale jusqu’au bâtiment avec l’aire maximale.

III. Résultats pour chaque algorithme

Pour faire suite à la partie d'analyse des instances, chacune des instances a été testée 5 fois pour réaliser une moyenne approximative pour chaque algorithme. Sachant que l'instance reste la même, seul l'algorithme aléatoire possède un résultat différent à chaque essai. Les résultats ont été répertoriés dans un fichier Google Sheet disponible ici pour la version numérique de ce rapport.
Voici un aperçu des résultats obtenus pour les différentes instances :

Execution	Type d'algo	Evaluation	Nb Batiment
1	Défaut	43	5
1	MaxAire	40	3
1	MaxEncombrement	24	2
1	Random	38	4
1	MinAire	32	5
2	Défaut	43	5
2	MaxAire	40	3
2	MaxEncombrement	24	2
2	Random	34	4
2	MinAire	32	5
3	Défaut	43	5
3	MaxAire	40	3
3	MaxEncombrement	24	2
3	Random	43	4
3	MinAire	32	5
4	Défaut	43	5
4	MaxAire	40	3
4	MaxEncombrement	24	2
4	Random	44	4
4	MinAire	32	5
5	Défaut	43	5
5	MaxAire	40	3
5	MaxEncombrement	24	2
5	Random	34	4
5	MinAire	32	5

	Défaut	MaxAire	MaxEncombrement	Random	MinAire
Moyenne éval	43	40	24	38,6	32
Moyenne nbBatiment	5	3	2	4	5

Classement:	
	1
	2
	3
	4
	5

Nous pouvons observer le classement à l'aide des couleurs allant du vert pour le meilleur algorithme pour le test, jusqu'au rouge pour le pire algorithme.

Comme indiqué sur la capture d'écran juste au dessus, voici la moyenne des résultats pour chaque instance :

- Instance 0 :

	Défaut	MaxAire	MaxEncombrement	Random	MinAire
Moyenne éval	43	40	24	38,6	32
Moyenne nbBatiment	5	3	2	4	5

- Instance 1 :

	Défaut	MaxAire	MaxEncombrement	Random	MinAire
Moyenne éval	56	64	62	55,6	46
Moyenne nbBatiment	7	4	4	6,2	10

- Instance 2 :

	Défaut	MaxAire	MaxEncombrement	Random	MinAire
Moyenne éval	30	58	58	41	55
Moyenne nbBatiment	2	2	2	4,2	8

- Instance 3 :

	Défaut	MaxAire	MaxEncombrement	Random	MinAire
Moyenne éval	258	266	276	223,8	225
Moyenne nbBatiment	10	6	7	11	15

- Instance 4 :

	Défaut	MaxAire	MaxEncombrement	Random	MinAire
Moyenne éval	120	135	128	134	117
Moyenne nbBatiment	7	5	5	8,2	11

- Instance 5 :

	Défaut	MaxAire	MaxEncombrement	Random	MinAire
Moyenne éval	112	80	80	128	92
Moyenne nbBatiment	6	2	2	7	11

IV. Conclusion

Comme vu précédemment avec la moyenne des résultats de chaque instance, on peut remarquer qu'il n'y a aucun algorithme qui se démarque des autres, leur efficacité respective dépend essentiellement de l'instance de données qu'on vient injecter au programme. Nous avons ajouté un deuxième critère de classement en comptabilisant le nombre de bâtiments placés sur le terrain en plus du critère de remplissage. On se rend alors compte que les deux critères ne sont pas liés. Ce qui signifie que l'algorithme qui place le plus de bâtiments sur le terrain n'est pas l'algorithme qui occupe le plus d'espace, et inversement.

On peut donc en conclure qu'il n'y a pas un algorithme plus efficace qu'un autre sur ce problème, tout dépendra des différents paramètres implémentés dans l'instance de données.