**INF8775 – Analyse et conception d’algorithmes**

TP3 – Automne 2022

|  |  |
| --- | --- |
| **Nom, prénom, matricule des membres** | Dansereau, Charles, 1897705  El Fakhry, Farid, 1875036 |
| **Note finale / 25** | **0** |

# Informations sur la correction

* Répondez directement dans le document DOCX.
* La correction se fait sur ce même rapport.
* Vous devez faire une remise électronique sur Moodle avant le 6 Décembre 2022 en suivant les instructions suivantes :
  + Vos fichiers doivent être remis dans une archive zip à la racine de laquelle on retrouve :
    - Le rapport au format DOCX.
    - Un script nommé *tp.sh* servant à exécuter les différents algorithmes du TP. L’interface du script est décrite à la fin du rapport.
    - Le code source et les exécutables.
    - Si le langage que vous utilisez nécessite une phase de compilation, veuillez joindre un Makefile afin que nous puissions le compiler en cas de problème avec vos exécutables. Si nous ne sommes pas en mesure de tester votre code, vous perdrez des points de respect d’interface et de qualité de code !
* Vous avez le choix du langage de programmation utilisé mais vous devrez utiliser les mêmes langage, compilateur et ordinateur pour toutes vos implantations. Le code et les exécutables soumis devront être compatibles avec les ordinateurs de la salle L-4714.
* Si vous utilisez des extraits de codes (programmes) trouvés sur Internet, vous devez en mentionner la source, sinon vous serez sanctionnés pour plagiat.

# Notations utilisées dans le rapport:

* m : nombre de circonscription
* n : nombre de municipalités
* muni : municipalités
* h,b : hauteur, largeur

N.B. Un readme contenant les instruction de compilation est fournis.

# Q1 – Description de votre algorithme

*Décrivez en quelques phrases votre algorithme. Soyez clair et concis. Donnez les noms des patrons de conception utilisés.*

|  |  |
| --- | --- |
| **0** | / 4 pt |

Notre algorithme se fait en deux étapes:

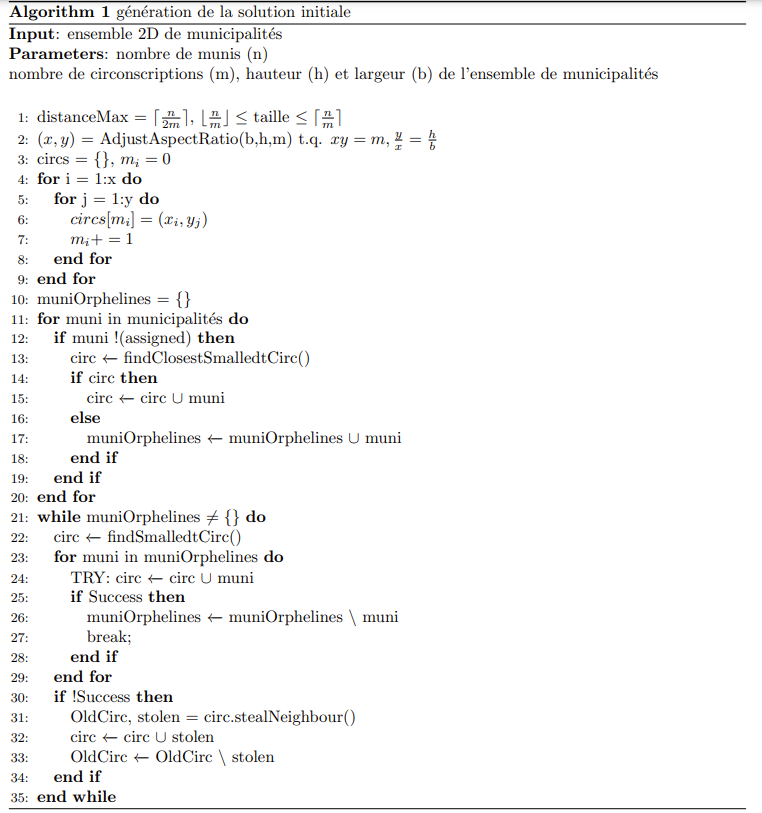
En premier lieu, nous commençons par un algorithme **probabiliste** qui créera une solution initiale. On crée m circonscriptions contenant initialement une municipalité chacune, de sorte qu’elles soient équidistantes et uniformément distribuées sur l’ensemble des munis. Par la suite, on assigne les municipalités à l’aide d’un algorithme **glouton** avec comme critère une distance croissante à la municipalité de départ des circonscriptions. S’il reste un groupe de municipalités qui ne sont assignées à aucune circonscription, on nomme ces dernières orphelines. En utilisant un algorithme “**Sherwood**” nous essayons de remplir les munis orphelines dans les circonscriptions. Il choisit une de municipalités voisines de la plus petite circonscription, et l’assigne à celle-ci, l’enlevant de sa circonscription originelle si besoin. Puis il vérifie si une des orphelines peut maintenant être assignée. Cela est répété jusqu’à convergence.

Ensuite, nous utilisons un 2e algorithme **probabiliste** avec **heuristique** **locale** pour améliorer itérativement cette solution. Le voisinage considéré est l'échange de 2 municipalités appartenant à 2 circonscriptions différentes. Nous utilisons un algorithme de **monte Carlo** qui choisit aléatoirement un nouvel état dans le voisinage, l’évalue, et modifie la solution avec différentes probabilités selon son évaluation (ex. gagne ou perd une circonscription)

# Q2 – Présentation

*Sous forme de pseudo-code et incluant une analyse de complexité théorique des principales fonctions. Si vous préférez écrire vos équations en Latex, vous pouvez ajouter un pdf à la remise avec la réponse à cette question et le mentionner ici. Pas besoin de faire une analyse empirique de la complexité.*

|  |  |
| --- | --- |
| **0** | / 6 p |



O(n/m)

Θ(n)

(max(n/m,m))

Θ(m)

Θ(m)

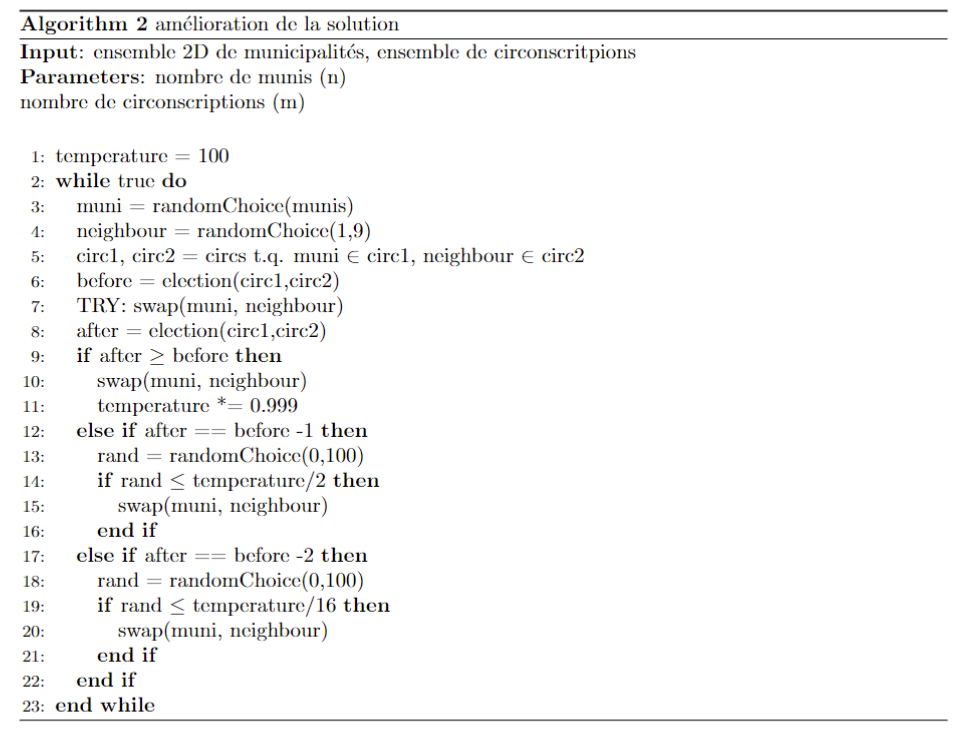
Θ(max(mn, n^2/m))

Θ(n/m)

O(n/m)

Θ(m)

Vu la nature probabiliste de cet algorithme, on ne peut déterminer la complexité réelle de cet algorithme, on peut par contre affirmer que l’encadré orange est dans Ω(max(n/m,m)) et que les encadrés bleus sont dans Ө(max(mn,n^2/m)), donc l’algo est dans **Ω(max(mn,n^2/m))**.



O(n/m)

O(n/m)

O(1)

O(n/m)

O(n/m)

Encore une fois, on ne peut pas prédire le nombre d’itérations nécessaires avant convergence. Une itération par contre évolue avec la taille des circonscriptions O(n/m), cela est dû à la fonction swap qui doit itérer à travers les municipalités deux circonscriptions. on a donc **Ω(n/m)**.

# Q3 – Justification de l’originalité de vote algorithme

*La conception de votre algorithme sera jugée avec les critères suivants :*

* *Lien avec le contenu du cours*
* *Originalité*

*Si vous vous êtes inspirés d’un algorithme existant, citez sa source.*

|  |  |
| --- | --- |
| **0** | / 4 pt |

Notre algorithme est aussi original que l'énoncé du problème. Bien que la notion de gerrymandering est bien documenté dans la littérature, les contraintes sont assez différentes :

* Les exemple réel de gerrymandering sont souvent fait avec des municipalités connexes
* La distance entre deux municipalités de la même circonscriptions n’est pas prise en compte

Nous avons recherché plusieurs sources ou l’utilisation de programmation dynamique assure une réponse optimale: [Cours de northeastern university (Boston)](https://www.youtube.com/watch?v=h7i5FHf0Mxg&t=1184s). Nous avons opté de ne pas utiliser cette approche vu qu’elle ne tient pas compte des contraintes.

Notre algorithme applique ainsi les notions du cours suivantes: algorithmes probabilistes (Sherwood et monte Carlo), algorithme glouton et heuristique locale.

Une partie particulièrement intéressante et importante de l’algorithme est le travail fait pour initialiser les 1ères municipalités des circonscriptions de façon intelligente (le plus réparti possible) on peut comparer cela à un algorithme de las-vegas hybride (sherwood hybride dans notre cas), car une partie des municipalités sont assignées de façon déterministe et une autre partie de façon aléatoire. La différence est que la partie déterministe est réalisée avant, et la partie aléatoire peut modifier cette solution si elle ne conduit pas à une solution légale, garantissant la convergence. De plus, une solution simplement probabiliste n’arriverait pas à converger dans les temps requis du fait que le nombre d’échanges aléatoires requis pour assigner une municipalité orpheline à une circonscription lointaine croît très rapidement dans la taille de l’exemplaire. Le travail de répartition glouton permet en effet de limiter le nombre de municipalités orphelines à 0 pour les formes régulières, et moins de 10% (en pratique) pour les cas plus difficiles.

# Q4 – Votre algorithme est-il assuré de trouver la solution optimale ?

*Répondez simplement “oui” ou “non”. Aucune justification requise.*

|  |  |
| --- | --- |
| **0** | / 1 pt |

non

# Autres critères de correction

## Qualité de l’algorithme

*Les points de cette question sont répartis comme suit :  
Nous allons exécuter votre code sur 3 exemplaires de notre choix. Pour chaque exemplaire la sortie de votre code sera envoyée vers le script de vérification, et nous classerons sur chaque exemplaire les différents binômes.  
Il y a 2 points par exemplaire. Le premier s’obtient en dépassant une baseline obtenue avec un algorithme basique. Le deuxième dépend de votre classement. Si votre algorithme donne une solution valide mais est classé dans le dernier quart vous aurez 0.25pt, si il est classé dans le 3eme quart vous aurez 0.5pt, si il est classé dans le 2eme quart, vous aurez 0.75pt. Enfin, si il est classé dans le meilleur quart, vous obtiendrez 1pt.*

*Ainsi si votre algorithme retourne des solutions valides dépassant la baseline, alors vous avez au moins 3,75 pts, indépendamment du classement.*

|  |  |
| --- | --- |
|  | / 6 pt |

## Respect de l’interface tp.sh

|  |  |
| --- | --- |
|  | / 1 pt |

## 

## Qualité du code

|  |  |
| --- | --- |
|  | / 2 pt |

## Présentation générale (concision, qualité du français, etc.)

|  |  |
| --- | --- |
|  | / 1 pt |

## Pénalités

|  |
| --- |
| 0 |

* Retard : -15 % / journée de retard, arrondi vers le haut. Les TPs ne sont plus acceptés après 3 jours.
* Autres : Le correcteur peut attribuer d’autres pénalités (par exemple si les exécutables sont manquants, etc.)