
# Clustering de protéines

## Rapport de Projet

## INFO0606

Encadrants : Jean-Charles Boisson et Eric Hénon

Faveraux Adrien – Victor Bourquin

## Sommaire

# I. La clusterisation

## 1. Les objectifs du projet

Notre objectif est de reproduire le clustering de protéines en partant d'un logiciel appelé chimera qui sert à la modélisation de protéines en 3 dimensions, ce logiciel dispose de nombreux outils qui permettent d'interagir avec les protéines qui y sont chargées.

Pour le calcul du clustering, ce logiciel se sert de l'article : An automated approach for clustering an ensemble of NMR-derived protein structures into conformationally related subfamilies. Kelley LA, Gardner SP, Sutcliffe MJ publié en 1996.

Pour faire cela, il faudra analyser cet article pour en faire ressortir le fonctionnement de la clusterisation.

Ensuite, grâce à l'étude de l'article, créer notre propre logiciel qui calculerait le clustering avec des entrées les plus génériques possible et sortirait les informations de la façon dont on le veut pour des tâches automatiques où un clustering de protéines serait nécessaire. Le but étant d'avoir un logiciel qui peut être utilisé de la manière la plus flexible possible autant pour les entrées que pour la sortie des résultats.

Une fois le logiciel implémenter, il faudra le tester pour vérifier sont bon fonctionnement en le comparent ses résultats avec chimera avec plusieurs entrées

# 2. La Bio Informatique

## 1) La bio informatique en quelque mot

Grâce aux récents progrès de l'informatique permettant d'obtenir des machines capables d'effectuer des calculs de plus en plus complexes, de nouveaux outils ont été donnés aux biologistes. Certains de ces outils leur permettent d'effectuer de nouvelles tâches impossibles à effectuer avec des moyens classiques, ou leur permettent de faciliter des tâches que l'on pouvait déjà effectuer par le passé. Tous ces outils ne servent que dans un seul but, comprendre le fonctionnement de la vie.

Dans cette optique plusieurs recherches impossibles auparavant ont déjà été menées à bien grâce à l'utilisation d'ordinateurs. Ces recherches comprennent notamment :

- La création de médicaments
- L'identification de risques génétiques (ex trisomie 21)
- La thérapie génique
- La modification génétique d'être vivant dont l'application la plus connue et la création des OGM

## 2) La clusterisation pour quoi ?

La clusterisation consiste à former des groupes d'éléments possédants des points communs. Il existe un grand nombre de moyens de former des clusters. Ses applications vont de la biologie aux sciences sociales en passant par l'informatique. Cette discipline possède donc de nombreuses applications, mais il serait inutile de toutes les décrire ici. Nous nous contenterons donc de parler de son utilité dans le domaine des protéines, à savoir, déterminer des familles de protéines (les familles de protéines regroupent les protéines qui possèdent des structures, séquences et propriétés semblables).

L'identification de la famille de protéines est un point critique de la phylogénétique (l'étude des relations de parenté entre différents êtres vivants). Les familles sont utilisées ici aussi bien pour effectuer des analyses que pour obtenir des annotations fonctionnelles, ou bien encore permettre une exploration des fonctions des protéines dans une certaine branche phylogénétique.

Cette méthode de regroupement des protéines est aussi l'une des bases d'Enzymes Function Initiative (EFI) qui utilise des familles et superfamilles de protéines comme point de départ pour le développement afin de créer un classement global des enzymes dont on ne connaît pas les fonctions.

# II. Détails de la méthode utilisée

## 1. Principe de la méthode

Cette méthode est basée sur l'utilisation de deux approches :

La première a été créée par Mr Adzhube en 1995,il superpose toutes les structures par paire pour générer un ensemble de valeur RMSD. Après l'analyse basée sur ces distances, il faut que l'utilisateur définisse une valeur de *cut-off* qui correspond à un ensemble de clusters parmi ceux qui ont été calculés. Cette méthode requiert donc une intervention de l'utilisateur.

La deuxième approche du clustering a été faite par Mr Diamond en 1995, il superpose collectivement les structures et les transformations des corps rigides pour trouver un *cut-off point* dans un système de clustering quelconque.

L'auteur précise que la deuxième approche nécessite des valeurs prédéterminées pour approcher la valeur du *cut-off point* obtenue et que l'on pourrait rater le véritable cluster en utilisant cette méthode, car il pourrait tomber sous le seuil imposé par une valeur de *cut-off* rigide.

En effet, l'auteur a vu que la distance moyenne entre deux ensembles de protéines variait entre 0,29 et 11,3. On approche donc la distance entre deux ensembles de clusters pour que le résultat tombe entre ces valeurs.

L'auteur va utiliser la première méthode pour former les clusters et utiliser la seconde avec quelques modifications pour calculer la valeur du *cut-off point* de manière automatique pour en arriver à une méthode fiable et 100 % automatique.

Il y a des avantages à utiliser cette méthode :

- Elle ne prend pas en compte le poids des atomes. La plupart des algorithmes de clusterisation prennent cette donnée en compte à la formation de leur cluster complexifiant l'algorithme. Celui-ci a vocation d'être très générique.

- Un deuxième point particulièrement intéressant de cet algorithme est qu'il ne demande en entrée qu'une suite de coordonnées, et qu'il peut, à partir de là, calculer toutes les valeurs nécessaires à la clusterisation, la rendant totalement automatique.

- Enfin, cet algorithme possède un temps d'exécution tout à fait convenable O ($n^2$) et donne des résultats tout à fait convenables pour les calculs auxquels il est voué (pas de distorsion de l'information de plus de 10E-6 A)

## 2. Détails des différentes étapes de l'article

Pour commencer l'algorithme de clusterisation, on doit déjà avoir les distances entre chaque protéine pour savoir si elles sont de la même famille ou non. Plus deux protéines sont proches, plus elles ont de chance d'être dans la même famille.

Pour chaque ensemble, on assemble les membres par paires pour calculer leur distance grâce à la méthode RMSD (Root Mean Square Déviation).

Cette méthode est utilisée, car il s'agit d'une méthode de calcul de distance permettant d'obtenir des résultats satisfaisants pour des structures comme les protéines. On obtient donc une matrice de N*N de distance RMSD (ou N est le nombre de structures en entrée)

Selon l'auteur, il ne faut prendre que les atomes lourds dans la réalisation du calcul.
La formule du RMSD est :

$$RMSD(v,w) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left((v_{ix}-w_{ix})^2+(v_{iy}-w_{iy})^2+(v_{iz}-w_{iz})^2\right)}$$

Une fois le calcul des distances effectué, on va commencer par mettre chaque structure (protéine) dans un *cluster* différent, on aura donc N *cluster.* Ensuite, notre but sera de former des groupes de protéines en fusionnant les *clusters.* À chaque fusion de cluster, on peut considérer que c'est un résultat possible du clustering, on aura donc N-1 fusion où N est le nombre de clusters.

À chaque étape, nous fusionnerons deux *clusters* qui sont les plus proches car plus deux protéines sont proches, plus elles ont de chance d'être dans la même famille. Quand un *cluster,* qui contient plusieurs protéines, a un autre *cluster* qui en contient plusieurs, c'est la distance moyenne entre toutes les protéines du premier cluster et les protéines du second cluster qui va être utilisée. Pour savoir quels *clusters* sont les plus proches, il faut utiliser cette formule :

$$dist(m,n) = \frac{\left(\sum_{i=1}^{X}\sum_{j=1}^{Y} dist(i,j)\right)}{XY}$$

Cette formule prend les deux *clusters* m et n, elle ajoute la distance de toutes les protéines du premier *cluster* au second *cluster,* divisé par le nombre de clusters, ce qui donne la distance moyenne entre les deux clusters.

Suite à la fusion des deux clusters, il faut maintenant calculer la dispersion du nouveau cluster qui a été formé. Vu que ce sont les deux clusters les plus proches, la dispersion devrait augmenter, car chaque protéine du cluster ne colle pas forcément aux autres. Cette valeur influencera le résultat final pour définir si oui ou non ils seront dans la même famille. Si des protéines sont très proches entre elles, la dispersion sera faible, au contraire si elles sont loin la valeur de dispersion sera très forte.

$$spread_m = \frac{\left(\sum_{k=1}^{N} \sum_{i=1, i<k}^{N} dist(i,k)\right)}{N(N-1)/2}$$

Dans cette formule, m correspond au cluster où l'on veut calculer la dispersion, N représente le nombre de protéines dans le *cluster*. Pour chaque protéine entre elles, on additionne leur distance et on divise par leur nombre de distances additionnées.

Ensuite, une fois le *spread* du nouveau cluster fusionné calculé, on va recalculer la moyenne des dispersions de tous les *clusters. C*ela permettra d'établir un bilan global de la situation dans le clustering car si la moyenne de dispersion est très haute, cela signifie qu'il y a beaucoup de clusters avec des protéines éloignées et qu'il y a donc trop peu de familles et certaines protéines ne sont pas dans la bonne famille. Plus la moyenne de dispersion est faible en augmentant le nombre de clusters, plus les protéines ont bien été attribuées à la bonne famille, si la moyenne de dispersion ne diminue plus mais augmente. Cela signifie que l'on est en train de fusionner des clusters éloignés et qu'on est allé trop loin. L'auteur s'appuie sur cette valeur pour définir le *cutoffpoint* mais l'auteur va normaliser cette valeur pour éviter des faux positifs en cherchant le *cutoffpoint*.

$$AvSp_i = \frac{\left(\sum_{m=1}^{cnum_i} spread(m)\right)}{cnum_i}$$

Dans ce calcul *i* correspond à l'étape où nous en sommes dans le clustering. À chaque fusion de deux clusters, *i* est incrémenté de 1 et on peut considérer que c'est une solution possible au clustering.
*cnumi* correspond au nombre de clusters à l'étape i en excluant les clusters d'un seul élément.
Pour résumer, on additionne toutes les valeurs de dispersion de tous les *clusters* que l'on divise par le nombre de *clusters*.

Après avoir récupéré les différentes AvsP, on les normalisera pour les faire entrer dans une plage allant de 1 à N-1 (où N est le nombre de structures original). La normalisation sera effectuée dans le but d'avoir un poids égal pour chaque AvsP dans l'étape suivante (sans cette normalisation des cas exceptionnels où, par exemple deux protéines seraient très proches et auraient par conséquent un AvsP proche de 0 fausserait les calculs)

$$AvSp\left(norm\right)_i=\left(\frac{N-2}{Max\left(AvSp\right)-Min\left(AvSp\right)}\right)\left(AvSp_i-Min\left(AvSp\right)\right)+1$$

Pour chaque étape de l'algorithme de liaison moyenne, on calculera une « *penalty value* ». Cette valeur représentera le rapport entre l'AsvP et le nombre de *clusters.* Plus elle sera petite, plus le résultat sera satisfaisant (créant des cluster très peuplés, avec une faible dispersion).

$$P_i=AvSp\left(norm\right)_i+nclus_i$$

On comparera les différentes *penalty value* obtenues et on choisira la plus faible comme étant le point de coupe (cut-off value).

C'est à l'étape qui produit cette valeur que l'on trouvera les clusters les plus probables et les plus corrects (plus les clusters sont peuplés, moins il y a de chance d'oublier une protéine qui aurait dû en faire partie et moins il y a de dispersion, plus ses membres seront conformes). Ce sera cette étape qui sera choisie et donnée comme résultat par notre algorithme.

On peut voir le calcul comme ceci :

$$P_{icut}=Min\left(P\right)$$

Le résultat sera le *cut-off point* et se base sur la plus petite valeur de la *penalty value*.

# III. Modélisation du projet

## 1. Besoin du projet

Lors de l'analyse de l'article scientifique pour la clusterisation, il nous a fallu dégager un schéma pour commencer le développement. Voici les grandes lignes de ce qu'il nous fallait :

→ Un moyen de stockage pour le RMSD.

→ Un moyen de représenter une protéine dans son ensemble, protéine, chaîne de protéine, atome. Pour faire cela, nous avons pensé à des classes hiérarchisées.

→ Un moyen de lire différents types d'entrées pour les protéines, sous quelle forme les placer en mémoire, c'est là d'où est venue l'idée de créer un lecteur PDB.

→ Une classe pour pouvoir stocker plusieurs protéines de façon dynamique : c'est là que l'on a crée la classe cluster. Dans l'article, un cluster peut recalculer sa dispersion quand on le lui demande, on a donc ajouté cette fonction dans la classe cluster.

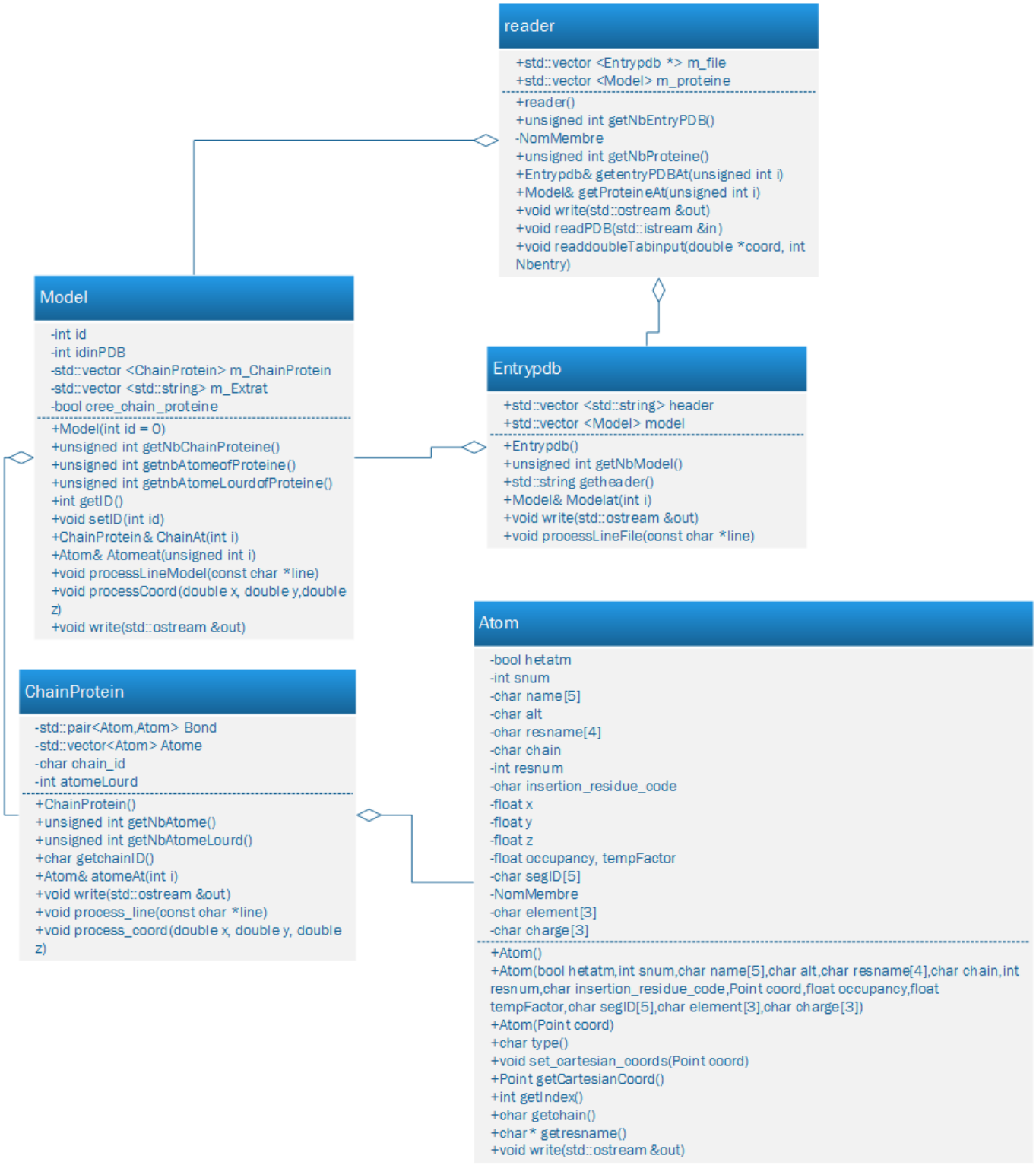
→ Il nous fallait un moyen de représenter chaque étape du clustering. Dans l'article, chaque étape est une modification d'un ensemble de cluster, on a donc créé une classe Ensemble_Cluster qui va être créée à chaque étape du clustering. Selon l'article, un ensemble de clusters doit pouvoir fusionner les deux clusters les plus proches ainsi que pouvoir calculer la dispersion globale des clusters.

On doit pouvoir ajouter un cluster à un ensemble de cluster.

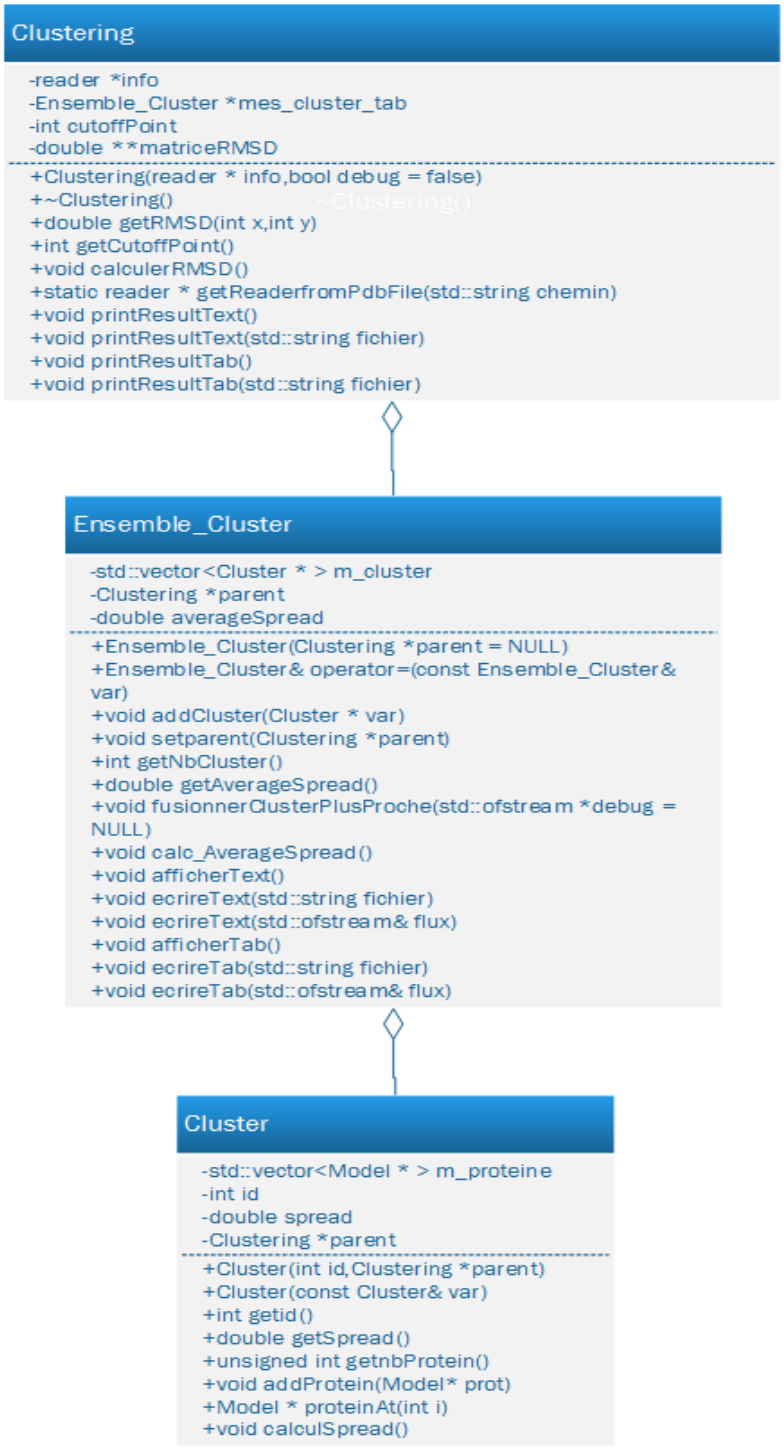
Un ensemble de cluster doit pouvoir calculer une distance entre deux clusters.

En temps qu'étape du clustering ,l'une de ces étapes va donc être sélectionnée. Il a donc fallu aussi implémenter un moyen d'afficher les informations depuis un ensemble de clusters.

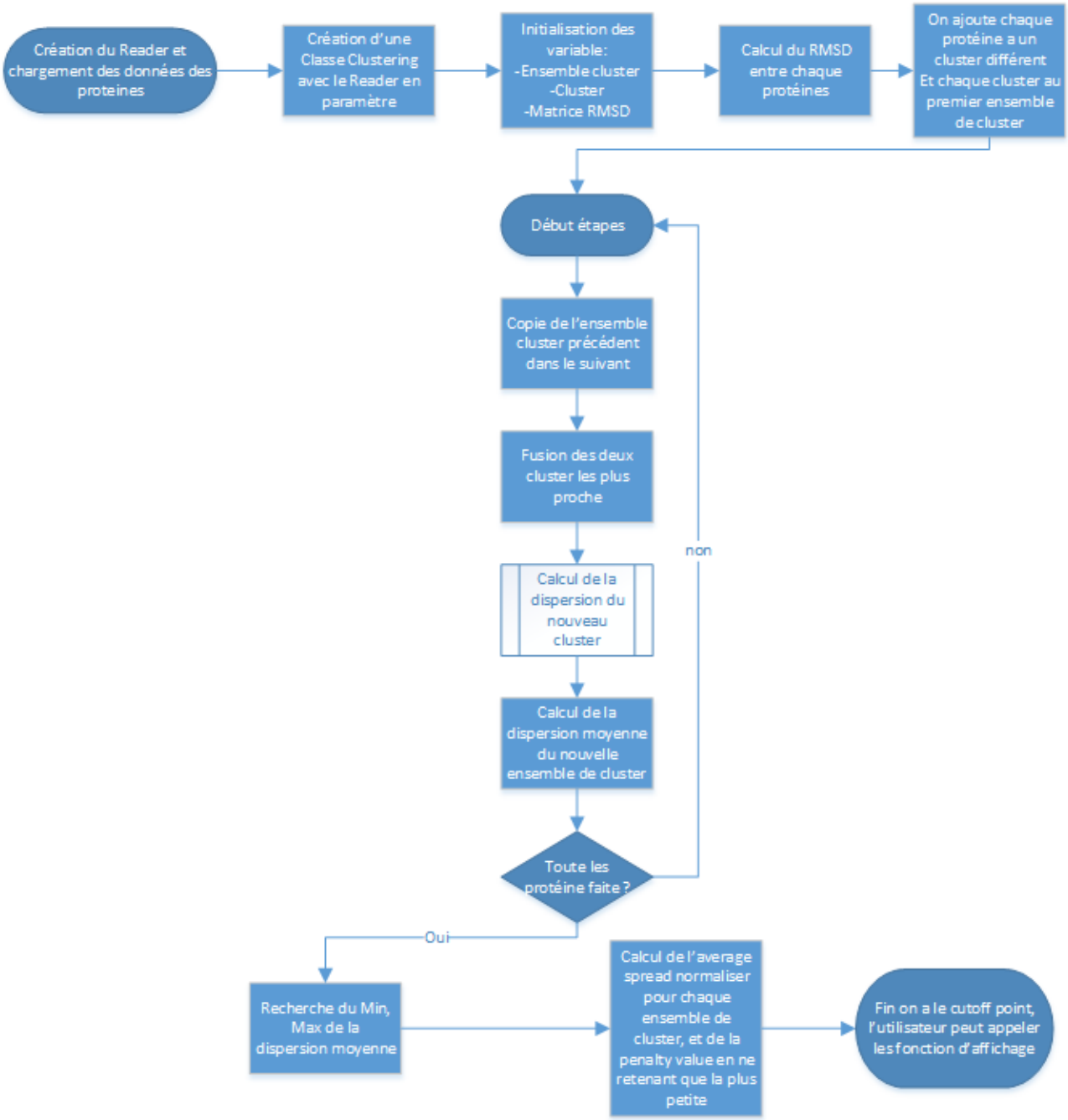
## 2. Diagramme de classe du lecteur



## 3. Diagramme de classe du clustering



## 4. Diagramme d'exécution



# IV. Implémentation de l'algorithme

## 1. Le lecteur d'entrée :

La classe *reader* sert à lire les informations envoyées au programme. Elle peut lire les informations d'un fichier PDB ou bien un tableau de coordonnées.

Pour lire un fichier au format PDB, le programme va lire le fichier ligne par ligne à chacune des sous-classes qui le compose.
Chaque sous-classe qui la compose vérifiera si la ligne la concerne. Le cas échéant, elle passera la ligne à sa sous-classe, et ce, en suivant ce chemin :
→ Reader → EntrerPdb → Model → ChainProtein → Atome

Lorsque le lecteur reçoit un tableau de coordonnées en entrée, celui-ci doit être accompagné du nombre de cases dans le tableau. Le lecteur commencera alors par vérifier que le nombre de coordonnées du tableau est correct (multiple de 3). Dans ce cas, il lira les coordonnées du tableau trois par trois et les placera dans un vecteur de la classe Model qui se chargera de créer une chaîne de protéines et les atomes associés.

À chaque lecture d'un fichier PDB ou d'un tableau de coordonnées, les données précédentes sont gardées et additionnées aux nouvelles.

## 2. Explication des différentes classes

La classe « ensemble de cluster » sert à garder en mémoire un résultat possible de la clusterisation. Elle contient les clusters à une étape de fusion donnée. Un opérateur de copies a été créé afin de dupliquer l'ensemble de clusters à chaque copie. Chaque cluster qu'il contient est lui aussi copié pour faire deux entités distinctes.

La classe « cluster » permet représenté un cluster. Elle contient des protéines. Le cluster peut calculer sa dispersion en fonction des protéines qu'il stocke comme l'article le conçoit. Vu le nombre de manipulations sur les clusters notamment les copies, il ont été créés de la manière la plus légère possible en évitant les variables et en prenant des pointeurs vers une protéine (car une protéine ne change pas, peu importe le nombre de manipulations de cluster).

La classe « Clusterisation » est la classe principale du programme. Lors de sa création, elle effectue une clusterisation et l'utilisateur n'a rien à faire. Une fois la clusterisation terminée par le constructeur, l'utilisateur peut récupérer le résultat sous forme de tableau ou sous forme textuelle. Ce résultat peut être récupéré sous forme de flux , être écrit dans un fichier, ou être envoyé vers la sortie standard.

# 3. Tests du programme

## 1) Test 1 - 60 protéines

*Description :*

Le premier test du programme est un test de clustering sur un fichier pdb de 60 protéines fourni par Mr Boisson dont la plupart ont un tronc commun et seuls les 4 derniers atomes changeaient entre les protéines.

*Résultat de chimera :*

Chimera trouve 4 clusters et notre logiciel 4 également

## 2) Test en données aléatoires

Nous avons conçu un générateur de cube dans l'espace pour tester notre clusterisation en comparent les résultats de chimera et celle de notre application :

Sur les 5 tests de clusterisation notre application a réussi les 5 avec 20 et 40 cubes aléatoires

Bien sûr les résultats dépendent de la manière dont son placer les cubes et en étant aléatoire des tests plus pousser sur de vraie protéine seront nécessaire pour confirmer intégralement la bonne implémentation.

## V. Conclusion et évolution

Pour conclure, on peut dire que notre projet est réussi, car nous sommes parvenus à comprendre le principe de la clusterisation via le logiciel chimera et grâce à l'article associé à sa méthode de réalisation.

Suite à l'étude de cet article, nous avons pu comprendre les rouages du clustering , les différentes étapes qui la composent et la raison pour laquelle la distance entre les protéines est si importante. Nous avons également saisi la façon de calculer la dispersion,  nous amenant à savoir quelle étape du clustering choisir et en quoi ces étapes sont essentielles à sa bonne réalisation.

Grâce à cela, nous avons créé notre propre programme en suivant les étapes de l'article, tout en gardant à l'esprit que notre logiciel devait être flexible et rapide afin d'être utilisé de manière autonome et efficace.

Puis grâce à des séries de tests avec des données aléatoires, nous pouvons affirmer de son bon fonctionnement dans un environnement avec des données quelconques.

Évolution possible de notre programme :

-Multi-threading car le calcul du RMSD se fait en $O(n^2)$ se qui est l'étape la plus longue du clustering , on peut donc tirer partie de l'architecture multicœur des ordinateurs récents en parallélisant cette tache.

-Optimisation on peut optimiser le programme en retournant le cut off point avant que toute les penalty value soit calculer, car quand la penalty value augment c'est que l'on peut retourner le résultat.

-Reconnaissance de plus d'entrer en prenant en entrée d'autre fichier que le fichier de type PDB.