MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE ABDELHAMID IBN BADIS MOSTAGANEM

Faculté des Sciences Exactes et d'Informatique
Département de Mathématiques et d'Informatique
Filière Informatique

Deuxième Année Master Ingénierie des Systèmes d'Information

Implémentation d'algorithme de recherche d'un cycle dans un graphe.

Etudiants :

Dekhoukh Sarah
Benallou Souad

Encadrant :
Ghania Khensous

Année Universitaire 2014/ 2015

Résumé

Ce projet consiste en l'implémentation d'un algorithme de recherche d'un cycle dans un graphe. L'objectif est de détecter un cycle dans un graphe moléculaire en temps record ainsi qu'il est bien important d'optimiser la vitesse des algorithmes correspondant en particulier pour les systèmes de perception de cycle appliqués à des graphes complexes.

Afin de réaliser notre projet, notre application est une implémentation C++ de deux algorithmes à savoir :

- L'algorithme de réduction de P-Graph.
- L'algorithme de parcours en largeur.

SOMMAIRE

Introduction Générale

L'implémentation des algorithmes de recherche d'un cycle dans un graphe permet de manipuler des graphes non orientés. Les graphes sont souvent utilisés pour représenter des objets structurés : les sommets représentent les composants de l'objet et les arcs représentent les relations binaires entre ces composants. Les graphes sont par exemple utilisés pour représenter des images, des objets de conception, des molécules ou des protéines, des emplois du temps [1].

Depuis près de deux siècles, les chimistes à la suite des cristallographes ont rivalisé d'imagination afin de représenter les molécules et les cristaux pour les besoins de l'enseignement et de la recherche [2].

Les progrès accomplis en informatique depuis une vingtaine d'années permettent à tout chimiste de visualiser des molécules en s'aidant d'un micro-ordinateur grâce à des programmes dont certains appartiennent au domaine public ou sont libres d'utilisation. Loin de se limiter à une simple représentation statique, ces programmes permettent de déplacer la molécule en trois dimensions et d'afficher les principaux paramètres moléculaires tels que : distances interatomiques, angles entre les liaisons [2].

Notre projet consiste à trouver la meilleure méthode pour trouver les cycles dans un graphe
Pour ce faire, notre projet est organisé comme suit :

Nous avons abordé certains algorithmes de recherche de cycles dans un graphe notamment le parcours en profondeur, en largeur et l'algorithme de Dijikstra qui permettent de rechercher un cycle dans un graphe de manière différente
Pour les algorithmes de recherche d'un cycle dans une molécule on a présenté un aperçu du fonctionnement de l'algorithme de RP-Chemin et l'algorithme de réduction de P-Graphe
Afin d'implémenté des algorithmes de réduction P-Graph et le parcours en largeur nous avons utilisé le langage C++.

LISTE DES FIGURES

Chapitre 1 : Algorithme de recherche d'un cycle dans un graphe

I. Introduction

Chercher un cycle dans un graphe est un problème vague, qui peut avoir de nombreuses complexités différentes, les différents algorithmes qui sont programmés et en particulier les parcours de graphes le sont à partir de deux opérations de bases sur le graphe. La première opération est le fait de pouvoir parcourir les sommets du graphe. La seconde opération est le fait de parcourir les sommets adjacents à un sommet donné. Ces deux opérations sont fournies par le graphe sous forme d'itinéraires. [3]

II. Quelque algorithme de la recherche d'un cycle dans un graphe

II.1. Parcours dans les graphes

On peut rencontrer deux types de problèmes dans les parcours de graphes:

1) Il se peut que tous les autres sommets ne soient pas accessibles à partir du sommet de départ. Tel est le cas des graphes non connexes (voir tableau ci-dessous pour définition).

2) Le graphe peut contenir des cycles, et on doit s'assurer que l'algorithme ne rentre pas dans une boucle infinie.

Pour remédier aux deux problèmes cités, les algorithmes de parcours de graphes maintiennent en général un bit de marquage pour chaque sommet du graphe. Initialement, le bit est à 0 pour tous les sommets du graphe. Le bit de marquage est mis à 1 quand un sommet est visité pendant le parcours. Si un sommet marqué est rencontré à nouveau pendant le parcours, il n'est pas visité une deuxième fois. Ceci permet d'éviter les cycles. Une fois l'algorithme de parcours terminé, on peut vérifier si tous les sommets ont été visités en consultant le tableau de marquage. La stratégie de parcours d'un graphe est alors comme suit [4]Nous allons discuter, dans ce qui suit, deux procédures: exploration en profondeur d'abord etexploration en largeur d'abord.

II.1.1 Algorithme de parcours en profondeur

L'algorithme de parcours en profondeur ou DFS (Depth-First Search) est un algorithme de parcours de graphe qui se décrit naturellement de manière récursive. Son application la plus simple consiste à déterminer s'il existe un chemin d'un sommet à un autre.
Pour les graphes non orientés, il correspond à la méthode intuitive qu'on utilise pour trouver la sortie d'un labyrinthe sans tourner en rond.
L'algorithme de parcours en profondeur peut être étendu pour résoudre d'autres problèmes sur les graphes :

- Trouver un chemin entre 2 sommets.
- Trouver un cycle dans un graphe.

Exemple [4]

Ordre de visite : 1, 2, 4, 3, 5 ;

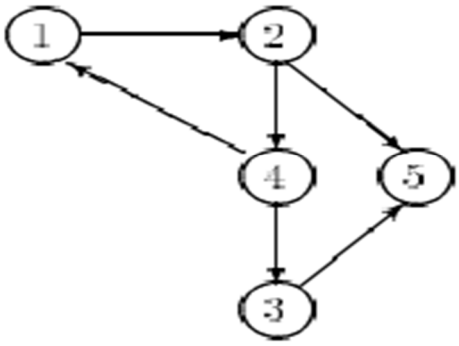


Figure1:Graphe représentant le parcours en profondeur.

II.1.1.1Principe

DFS c'est un algorithme de recherche qui progresse à partir d'un sommet S en s'appelant récursivement pour chaque sommet voisin de S.
Le nom d'algorithme en profondeur est dû au fait que, contrairement à l'algorithme de parcours en largeur, il explore en fait « à fond » les chemins un par un : pour chaque sommet, il marque le sommet actuel, et il prend le premier sommet voisin jusqu'à ce qu'un sommet n'ait plus de voisins (ou que tous ses voisins soient marqués), et revient alors au sommet père.
Si G n'est pas un arbre, l'algorithme pourrait tourner indéfiniment, c'est pour cela que l'on doit en outre marquer chaque sommet déjà parcouru, et ne parcourir que les sommets non encore marqués.
Dans le cas d'un arbre, le parcours en profondeur est utilisé pour caractériser l'arbre.

Enfin, il est tout à fait possible de l'implémenter itérativement à l'aide d'une pile LIFO(Last In First Out) contenant les sommets à explorer : on dépile un sommet et on empile ses voisins non encore explorés.

II.1.1.2Pseudo-Code

Initialement, aucun sommet n'est marqué.

```
DFS (graphe G, sommet s)
{
  Marquer(s);
  POUR CHAQUE élément s_fils de Voisins(s) FAIRE
     SI NonMarqué(s_fils) ALORS
DFS(G,s_fils);
     FIN-SI
  FIN-POUR
}
```

Voisins(s) : renvoie la liste des sommets adjacents à s.
Marquer(s) : marque un sommet s comme exploré, de manière à ne pas le considérer plusieurs fois.

II.1.1.3. Applications

Comme les autres algorithmes de parcours de graphe, l'algorithme de parcours en profondeur trouve l'ensemble des sommets accessibles depuis un sommet donné s, c'est-à-dire ceux vers lesquels il existe un chemin partant de s. Il s'agit précisément des sommets marqués par l'algorithme. Ceci s'applique à un graphe orienté ou non orienté. Sur un graphe non orienté, on peut utiliser cette propriété pour le calcul des composantes connexes.
Dans le cas d'un graphe orienté acyclique, le parcours en profondeur peut servir à calculer un tri topologique des sommets.
Lors d'un parcours en profondeur, on applique la règle "Dernier marqué, premier exploré"
L'algorithme de Kosaraju effectue un double parcours en profondeur pour calculer les composantes fortement connexes d'un graphe orienté quelconque.

II.1.2 Algorithme de parcours en largeur

L'algorithme de parcours en largeur ou BFS (Breadth-First Search) permet le parcours d'un graphe de manière itérative, en utilisant une file. Il peut par exemple servir à déterminer la connexité d'un graphe

L'algorithme de parcours en largeur peut être étendu pour résoudre d'autres problèmes sur les graphes :

- Trouver le plus court chemin entre 2 sommets.
- Trouver un cycle simple dans un graphe.

II.1.2.1Principe

Cet algorithme diffère de l'algorithme de parcours en profondeur par le fait que, à partir d'un sommet S, il liste d'abord les voisins de S pour ensuite les explorer un par un. Ce mode de fonctionnement utilise donc une file dans laquelle il prend le premier sommet et place en dernier ses voisins non encore explorés.

Lorsque l'algorithme est appliqué à un graphe quelconque, les sommets déjà visités doivent être marqués afin d'éviter qu'un même sommet soit exploré plusieurs fois. Dans le cas particulier d'un arbre, ce n'est pas nécessaire.

Les étapes de l'algorithme sont :

1. Mettre le nœud de départ dans la file.
2. Retirer le nœud du début de la file pour l'examiner.
3. Mettre tous les voisins non examinés dans la file (à la fin).
4. Si la file n'est pas vide reprendre à l'étape 2.

Note : l'utilisation d'une pile au lieu d'une file d'attente transformerait cet algorithme en un algorithme de parcours en profondeur.

Exemple [4]

Ordre de visite : 1, 2, 4, 5,3 ;

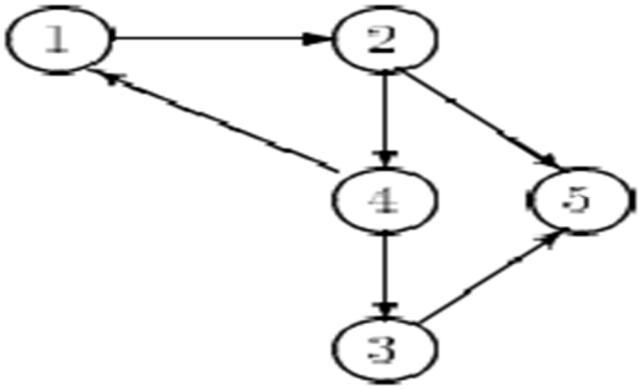


Figure2: Graphe représentant le parcours en largeur.

II.1.2.2 Pseudo-Code

BFS (Sommet s)

```
{
  f = CreerFile();
f.enfiler(s);
marquer(s);
  TANT-QUE NON f.vide() FAIRE
s = f.defiler();
afficher(s);
POUR TOUT voisin de s FAIRE
        SI voisin non marqué FAIRE
f.enfiler(voisin);
marquer(voisin);
        FIN SI
    FIN POUR TOUT
  FIN TANT QUE
 }
```

II.2. Algorithme de Dijkstra

En théorie des graphes, l'algorithme de Dijkstra sert à résoudre le problème du plus court chemin. Il permet, par exemple, de déterminer le plus court chemin pour se rendre d'une ville à une autre connaissant le réseau routier d'une région. Il s'applique à un graphe connexe dont le poids lié aux arêtes est un réel positif.
L'algorithme porte le nom de son inventeur, l'informaticien néerlandais EdsgerDijkstra, et a été publié en 1959.Cet algorithme est de complexité polynomiale.

II.2.1 Pseudo-code

```
Fonction Dijkstra (nœuds, fils, distance, début, fin)
    Pour n parcourant nœuds
        n.parcouru = infini   // Peut être implémenté avec -1
        n.précédent = 0
    Fin pour
    début.parcouru = 0
pasEncoreVu = nœuds
    Tant que pasEncoreVu != liste vide
        n1 = minimum(pasEncoreVu)    // Le nœud dans pasEncoreVu avec parcouru le plus petit
pasEncoreVu.enlever(n1)
        Pour n2 parcourant fils(n1)   // Les nœuds reliés à n1 par un arc
            Si n2.parcouru > n1.parcouru + distance(n1, n2)    // distance correspond au poids de l'arc reliant n1 et n2
                n2.parcouru = n1.parcouru + distance(n1, n2)
                n2.précédent = n1   // Dit que pour aller à n2, il faut passer par n1
            Fin si
        Fin pour
    Fin tant que
chemin = liste vide
n = fin
    Tant que n != début
        chemin.ajouterAvant(n)
        n = n.précédent
    Fin tant que
chemin.ajouterAvant(début)
    Retourner chemin
Fin fonction Dijkstra
```

II.2.2. Application

L'algorithme de Dijkstra trouve son utilité dans le calcul des itinéraires routiers. Le poids des arcs pouvant être la distance (pour le trajet le plus court), le temps estimé (pour le trajet le plus rapide), la consommation de carburant et le prix des péages (pour le trajet le plus économique).

II.2.3.Avantage et Inconvénients

L'algorithme de Dijkstra, très puissant, admet quand même quelques limites.
Il demande une certaine masse de traitements quand le graphe devient grand. Quand la puissance matérielle est limitée, ou que l'on souhaite traiter rapidement le problème du chemin le plus court, on peut utiliser d'autres algorithmes, tel A* (qui se prononce A étoile, est un algorithme de recherche de chemin dans un graphe)qui a la même complexité dans le pire des cas, mais qui est plus rapide en moyenne.
On ne peut pas affecter aux liaisons un poids négatif (boucle infinie dans le cas d'un cycle de poids négatif).

Cependant il a plusieurs avantages :

- contrairement à l'algorithme A*par exemple, il trouve toujours le chemin ayant le poids le plus faible **;**
- il n'est pas si dur à comprendre et à mettre en place.
- il peut être utilisé pour d'autres applications que la distance la plus courte d'un point à un autre.

III. Conclusion

Dans ce chapitre nous avons abordé certains algorithmes de recherche de cycles dans un graphe notamment le parcours en profondeur, en largeur et l'algorithme de Dijikstra qui permettent de rechercher un cycle dans un graphe de manière différente.
Il existe bien sûr d'autres algorithmes traitant ce problème et on peut constater que la recherche en graphes, en général, est un domaine de recherche en plein essor.

Chapitre 2 : Algorithme de recherche d'un cycle dans un graphe moléculaire.

I. Introduction

La perception de tous les cycles dans le graphe est une tâche coûteuse en temps de calcul. Il est important d'optimiserle temps d'exécution des algorithmes correspondant en particulier pour les systèmes de perception de cycle appliqués à des graphes ayant un grand nombre de sommets. [5]

II. L'algorithme de réduction de P-Graphe

II.1.Théorie

La première étape de l'algorithme de réduction consiste à convertir le graphe moléculaire (M-Graphe) en un graphe de chemin (P-Graphe), cela se fait de la manière suivante[5]:

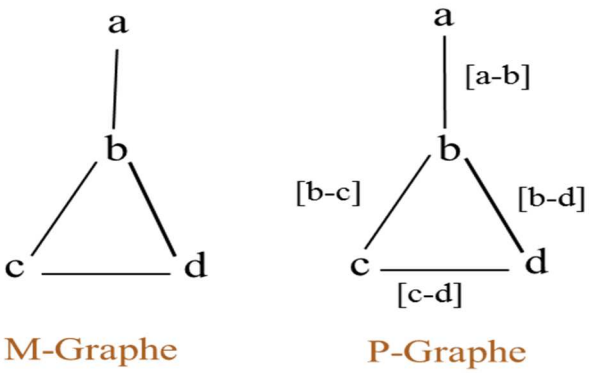


Figure 3 : Graphe moléculaire et Graphe de chemin.

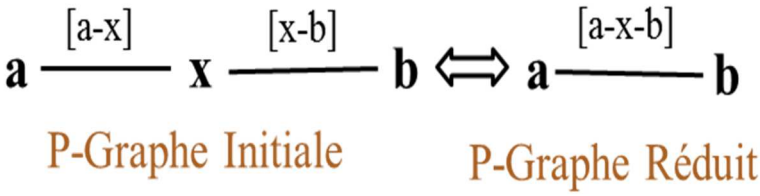


Figure 4 : Réduction de graphe de chemin.

- Pour chaque atome dans le M-Graphe correspond à un sommet de P-Graphe qui a la même étiquette. Pour chaque liaison de M-Graphe, il y'a une arête entre les mêmes sommets dans le P-Graphe. [5]
- L'étiquette d'une arrête a-b au P-Graphe décrit une liaison possible de a à b dans le M-Graphe et est définie par l'ensemble des arêtes de M-Graphe qui sont impliqués dans le

chemin. L'étiquette initiale des arrêtes P-Graphe correspond donc aux arrêtes M-Graphe associé.

- La seconde étape consiste à réduire la P-Graphe. L'idée de base de cette tâche est qu'une liaison a-x-b dans le M-Graphe peut être décrite dans le P-Graphe par une seule arrête entre a et b étiqueté [a-x-b]. En conséquence, le sommet x peut être retiré de la P-Graphe initial sans perte d'information topologique.
- Effectivement, pour enlever le sommet x, nous devons supprimer les deux arrêtes a-x et x-b et de créer une nouvelle arrête a-b. L'étiquette de a-b est tout simplement la concaténation des étiquettes de a-x et x-b.
- Pour bien clarifier, nous introduisons la notation Pxy, qui définit une arête entre les sommets x et y dans le P-Graphe. Le processus de réduction peut maintenant être décrit avec une règle unique et générale [5]:
- Pour supprimer un sommet x : (1) créer pour chaque couple de arrêtes Pxy, Pxz une arrête Pyz = Pxy⊕Pxz où ⊕ est un opérateur de concaténation; (2) supprimer le sommet x et tous les arrêtes impliquant x.
- Cette règle nous permet de réduire le nombre de sommets dans le P-Graphe sans perte d'informations de connectivité.
- Toutefois, afin de détecter que les éléments cycliques d'un graphe, on écarte les arrêtes qui ne représentent pas les chemins réels.
- La règle précédente peut être affinée afin que seuls les couples de arrêtes Pxy, $x \neq y$, Pxz, $x \neq z$, où Pxy⊗Pxz = {x} est vrai, conduisent à une nouvelle arrête Pyz.

Soient E et V l'ensemble des arêtes et sommets dans le P-Graphe. le pseudo-code correspondant peut être décrit comme suit : [5]

```
Supprimer (x,V,E)      // x :sommet, V :ensemble des sommets , E :ensemble des arrêtes //
   Pour chaque couple de chemins (Pxy,x≠y,Pxz, x≠z) ∈ a E²
        Si Pxy⊗Pxz={x} alors
        Pyz←Pxy⊕Pxz
            E←E∪ {Pyz}
   Pour chaque chemin Pxy∈ a E
        E←E -{Pxy}
   V←V -{x}
```

- La fonction Supprimer est utilisé pour percevoir tous les cycles dans un M-Graphe.

P-Graphe peut être totalement réduiten enlevant un à un les sommets.

- Ceci résulte en un effondrement général. Au cours de ce processus, le nombre de sommets diminue et le nombre d'arêtes localement peut augmenter ou diminuer en fonction de la topologie de la structure et de l'ordre dans lequel les sommets sont enlevés.
- Quand une nouvelle arrête entre elle-même et un sommet (boucle) est formée dans le P-Graphe, il correspond à un cycle dans le M-Graphe.
- En effet, la formation d'une boucle est équivalente à la fermeture d'un chemin d'accès et donc l'apparition d'un cycle.
- L'étiquette de la boucle décrit le cycle détecté. Lorsque le P-Graphe est totalement réduit (V = Φ/ E = Φ /), tous les cycles ont été détectés.

L'algorithme complet pour la perception du cycle suit exactement les deux étapes décrites ci-dessus. Le premier graphe moléculaire est converti en un graphe de chemin (défini par V et E).

Puis une boucle principale prend soin de supprimer un par un tous les sommets de P-Graphe.

La fonction Convert () s'appuie tout simplement sur les ensembles V et E de M-Graphe donné.

- La fonction Supprimer () est la même que celle présentée ci-dessus, sauf qu'un test spécial a été inclus pour la détection de boucle.

```
Anneaux (M-Graphe)
    Anneaux←∅
Convert (M-Graphe,V,E)
    Tant que V≠∅
        Choisir x dans V
        Supprimer (x,V,E,Anneaux)

Convert(M-Graphe,V,E)
    V←∅, E←∅
    Pour chaque sommet x dans M-Graphe
        V←V∪ {x}
    Pour chaque Arrête (x-y) dans M-Graphe
        Pxy←(x-y)
        E←E∪ {Pxy}

Supprimer (x, V,E,Anneaux)
    Pour chaque couple de trajets (Pxy,x≠y,Pxz,x≠z) ∈ E²
        Si Pxy⊗Pxz={x} alors
        Pyz←Pxy⊕Pxz
            E←E∪ {Pyz}
    Pour chaque trajet Pxy∈ E
        Si x=y alors Anneaux←Anneaux∪ {Pxy}
        E←E-{Pxy}
    V←V-{x}
```

Notations spéciales :

Pxy⊗Pxz: sommets communs entre chemins Pxy et Pxz.

Pxy⊕Pxz: concaténation des chemins Pxy et Pxz (joint au sommet x).

II.2 Raffinement

Les performances de l'algorithme dépendent fortement de l'ordre dans lequel les sommets sont retirés du P-Graphe. Depuis, il est préférable de commencer à jeter les chemins qui pourraient ne pas conduire à un cycle, nous pouvons supposer que les sommets appendices peuvent être directement supprimés.[5]

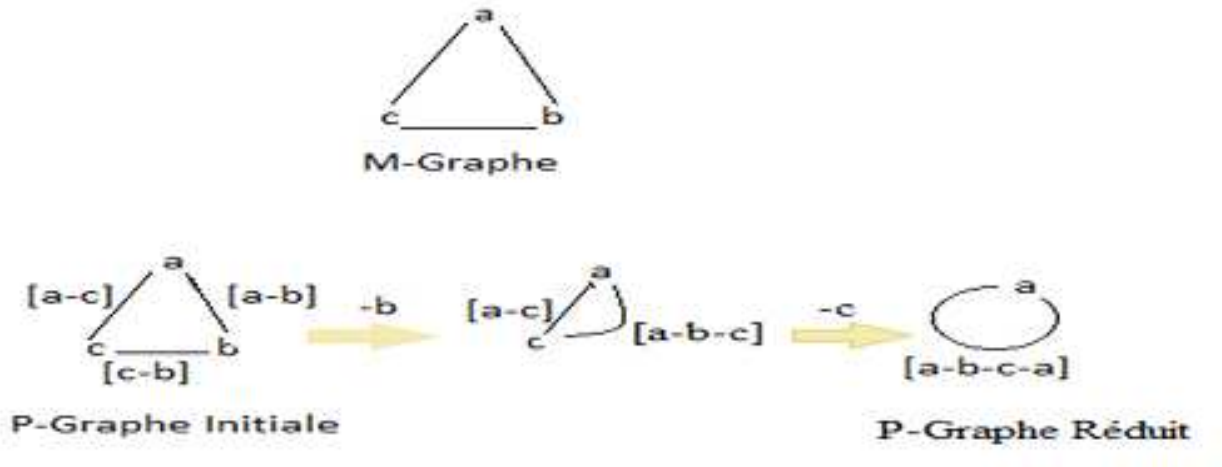


Figure 5 : Image représentant une boucle centrée sur le sommet « a » .

Une règle plus générale serait de garder la connectivité des nœuds restants dans le P-Graphe aussi bas que possible; cela permettrait d'assurer un nombre minimum de cas où Pxy⊗Pxz≠ {x} est vrai dans l'algorithme décrit précédemment. Un bon critère de sélection est donc la connectivité des sommets. Retrait des sommets selon une valeur croissante de connectivité est probablement le moyen le plus efficace pour réduire le P-Graph. Il est possible de sélectionner à chaque étape le sommet suivant à être enlevé selon le nouvel indice de connectivité induite par l'étape précédente ou de calculer, pour les premiers sommets P-Graph, un indice de connectivité étendue permet de trier les sommets et de deviner apriori une bonne séquence. [5]

II.3 Discussion

La perception de tous les cycles d'un graphe est une tâche fastidieuse. Il est important d'optimiser la vitesse d'algorithmes correspondants en particulier pour l'application de systèmes de cycle perception à un grand nombre de graphes (par exemple, la synthèse assistée par ordinateur). L'algorithme présenté dans le présent document a été conçu pour permettre une mise en œuvre effective de la vitesse en utilisant des ensembles comme une structure de données principale. Bien que les résultats de l'analyse statistique semblent très intéressant tout en traitant des valeurs moyennes, il est important de noter que la vitesse d'une perception très particulière dépend du type de structure étudiée, à savoir la " densité " et la position relative des cycles (non seulement le nombre). Graphes impliquant un grand nombre de cycles condensés peuvent conduire à des sommets avec un indice élevé con-connectivité pendant le processus effondrement et donc des performances moindres. Cependant, même pour la structure complexe, le nombre d'opérations de base nécessaires reste assez faible (B = 4519) en ce qui concerne le nombre de cycles perçus (R = 248).

L'algorithme effondrer P-Graph est très générale et peut être appliquée à tout graphe. En outre, il n'y a aucune exigence de connexité, chaque fragment dans un graphe nonconnexe sera traité indépendamment. [5]

Conçu à l'origine pour la perception de l'anneau dans les graphes moléculaires, le même algorithme peut également être utilisé pour énumérer tous les chemins entre deux sommets d'un graphe. En effet, ces chemins seront générés au cours du processus effondrement Chaque bord restant correspond à un chemin existant. Bien que ce soit probablement pas le moyen le plus efficace d'énumérer les chemins entre les sommets de remorquage, il montre le potentiel de la méthode graphe effondrement et nous permet d'envisager de nouvelles applications.[5]

III.L'algorithme de RP-Chemin

La perception des anneaux dans les graphes est largement utilisée dans de nombreux domaines de la science et de l'ingénierie. Algorithmes développés dans la communauté de la chimie, appelés plus petit ensemble de petits anneauxSSSR (Smallest Set of Smallest Rings), applicables que pour les graphes simples ou des structures chimiques. Dans les algorithmes de contraste développés par la communauté de la science informatique, appelé base de cycle minimum MCB (Minimum Cycle Basis) sont identiques à SSSR encore présenter une plus grande robustesse. Algorithmes basés-MCB peuvent correctement révéler tous les anneaux dans des graphes complexes. Cependant, ils sont lents lorsqu'il est appliqué à de grands graphes complexes en raison des limites inhérentes aux algorithmes utilisés. RP-chemin est une méthode de recherche robuste, simple et rapide avec $O(n^3)$ d'exécution qui identifie correctement le SSSR de l'ensemble des cas de test de graphes complexes en utilisant approche différente de la méthode de la MCB. Tant la robustesse et l'amélioration de la vitesse sont obtenues en utilisant une matrice de distance de trajet inclus et la description des caractéristiques de cycles de la matrice. Cette méthode est précise et plus rapide que les autres méthodes et peut trouver de nombreuses applications dans divers domaines de la science et de l'ingénierie qui utilisent des graphes complexes avec des milliers de nœuds.
Systèmes complexes, par exemple, les réseaux biologiques, réseaux de télécommunication, et les circuits électriques sont décrits avec des graphes pour la manipulation mathématique. Plus petit ensemble de bague la plus petite (SSSR) est le trait caractéristique de ces graphes. Par conséquent, il est crucial d'avoir la méthode de calcul SSSR robuste et rapide pour une analyse précise et la manipulation d'un graphe.Tous les algorithmes développés signalés par la communauté de la chimie pour trouver la SSSR, sont basées sur deux méthodes principales, une table de connexion de plain-pied de travers, et les manipulations de graphes. Certains aspects théoriques sont résumés. Bien que diverses méthodes pour trouver SSSRsont été développés en utilisant des algorithmes basés sur les deux méthodes, Du point de vue de la science informatique, identifier SSSRs, connus comme base de cycle minimum (MCB), est également intéressant. En plus de sa capacité d'identification SSSR, le temps de calcul d'une méthode particulière représentée par le temps polynôme est une considération importante vers sa performance. Horton a proposé un algorithme de temps polynomial robuste pour trouver la MCB d'un graphe. Dans cette approche, l'algorithme crée un sur-ensemble de la MCB, puis extrait comme les plus courts cycles linéairement indépendants du sur-ensemble en utilisant l'élimination de Gauss. De Pinaa proposé une méthode avec une autonomie de $O(m^3\ mn^2 logn)$ Plusieurs méthodes ont été développées sur la base de méthodes Horton et De Pina et leur puissance de calcul sont résumées.[6]

III.1. Résultat

RP-Chemin une méthode de calcul robuste et rapide pour la perception de l'anneau et l'identification SSSR de graphe complexes.
Au lieu d'utiliser un SDM, RP-Cheminutilise une matrice de PID qui contient des informations sur les sentiers et la taille des anneaux. Avec cette information, nous avons développé une méthode de perception de la bague robuste considérablement réduit le temps de calcul.
Cette méthode peut avoir une influence considérable sur les différents domaines de la science et de l'ingénierie, y compris les tests de circuit électrique, la planification de calendrier périodique, face à la rigidité de la structure de cadre, la perception de l'anneau chimique, et analyse des voies biologiques complexes avec des milliers de nœuds de réseau.[6]

IV.Conclusion

Ce chapitre a été consacré à la présentation d'un aperçu du fonctionnement de l'algorithme de RP-Cheminet l'algorithme de réduction de P-Graphe, de même, nous avons traité théoriquement les approches abordées lors de la conversion d'un M-Graphe en un P-Graphe.

Chapitre 3 : Implémentation d'algorithme de réduction P-Graph et parcours en largeur.

I.Introduction

Pour effectuer une recherche d'un cycle dans un graphe nous avons implémenté l'algorithme de réduction P-Graph et l'algorithme de parcours en largeur afin de nous détecter un cycle on optimisant le temps d'exécution.

II.Implémentation de l'algorithme de réduction P-Graph et de parcours en largeur

II.1.Implémentation de l'algorithme de réduction P-Graph

Cet algorithme est basé sur une réduction progressive de la courbe de trajet associée au graphe moléculaire étudié[5]. Le graphe de la trajectoire est une image du graphe moléculaire dans laquelle chaque sommet correspond à un atome du graphe moléculaire et chaque arrête a-b décrit une liaison reliant les atomes a et b dans le graphe moléculaire.
Lors de la réduction, les nœuds du graphe de chemin sont supprimés, et les informations relatives à l'apparition du cycle sont concentrées dans l'étiquette de nouvelles arêtes entre les sommets restants. Chaque boucle formée dans le P-graphe s'effondre au cours de ce processus correspond à un cycle dans le graphe moléculaire. Une fois le graphe moléculaire a totalement effondré, tous les anneaux dans le graphe moléculaire ont été perçus.

II.2.Implémentation de l'algorithme de parcours en largeur

Le parcours en largeur explore tous les sommets accessibles depuis le sommet initial. Il permet de calculer les composantes connexes du graphe avec une complexité linéaire.

Lors d'un parcours en largeur, on applique la règle "premier Marqué-premier exploré".

De plus, lors de ce parcours, les sommets sont explorés par distance croissante au sommet de départ. Grâce à cette propriété, on peut utiliser l'algorithme pour résoudre le plus simple des problèmes de cheminement : calculer le plus court chemin entre deux sommets dans un graphe non pondéré.

```
void TForm2::CreerFile()
{
     File    = new int[1];
     File[0] = 0;
     nbf=1;
}

void TForm2::Enfiler(int s)
{
     int *Tmp;
     Tmp = new int[nbf];
     for(int i=0;i<nbf;i++) Tmp[i] = File[i];

     File   = new int[nbf+1];

     for(int i=0;i<nbf;i++) File[i] = Tmp[i];
     File[nbf+1] = s;
     nbf++;

}

void TForm2::Marquer(int s)
{
     File[nbf] = s;
     nbf++;
}
```

Figure 6: Image représentant le code source de l'algorithme BFS.

III. Outil VMD

VMD (acronyme de Visual Molecular Dynamics) est un programme permettant d'étudier en 3 dimensions la structure d'une molécule complexe [7]. C'est un programme de visualisation moléculaire pour afficher, animer et analyser de grands systèmes biomoléculaires. [8]

III.1. Avantage de VMD

• Prise en charge de toutes les principales plates-formes informatiques.

• Prise en charge une variété de formats de fichiers .pdb, .mol2 etc.

• Prise en chargeplus de 60formats de fichiersmoléculaireset types de donnéesgrâce àune vaste bibliothèque defichiers intégrépluginslecteur /écrivainettraducteurs.

•Beaucoup d'excellentstutorielsVMDdéveloppéeslocalement etpar la communautédes chercheurs en général.[9]

III.2.Les fenêtres de VMD:

Lorsqu'on utilise VMD plusieurs fenétres apparaissent. Voici les principales et leurs fonctions:

III.2.1. la fenêtre console VMD

Figure 7 : Image représentant la fenêtre console VMD.

La *console VMD* affiche diverses informations notamment lorsqu'on affiche le label d'un atome ou d'une liaison. Si on ferme. [10]

III.2.2.La fenêtre Main:

La console VMD s'arrête sans sauvegarder. Lafenêtre*Main* est celle à partir de laquelle on peut faire afficher toutes les autres. C'est également celle à partir de laquelle on à accès à tous les outils de VMD. [10]

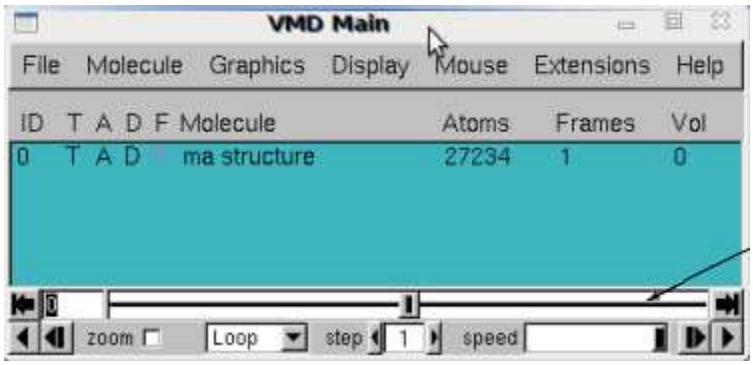


Figure 8 : Image représentant la fenêtre Main.

III.2.3. La fenêtre*Molécule File Browser:*

La fenêtre *Molecule File Browser* permet de recherche un fichier et de charger une structure ou encore de rajouter des données dans une structure déjà existante. Le rectangle vert affiche un aperçut du fichier ou des informations le concernant.[10]

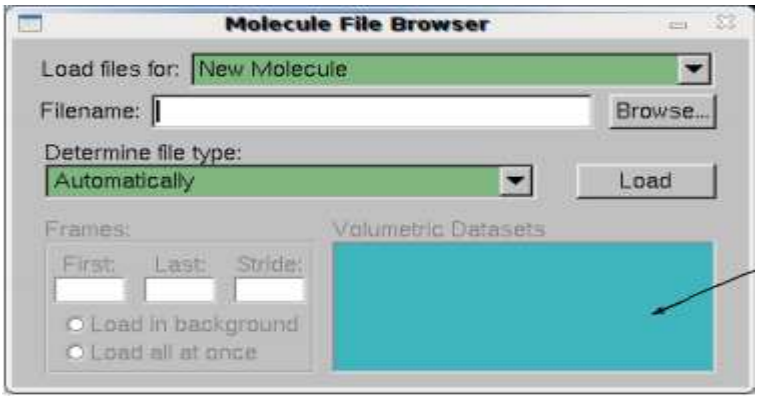


Figure 9 : Image représentant la fenêtre *Molécule File Browser.*

III.2.4. La fenêtre*GraphicalRepresentations :*

La fenêtre*GraphicalRepresentations*permet de sélectionner ce que l'on souhaite voir etde changer la façon dont on le voit : la couleur, le type de représentation etc ... Pour l'afficher dans la fenêtre*Main* cliquer sur *Graphics/Representations*.[10]

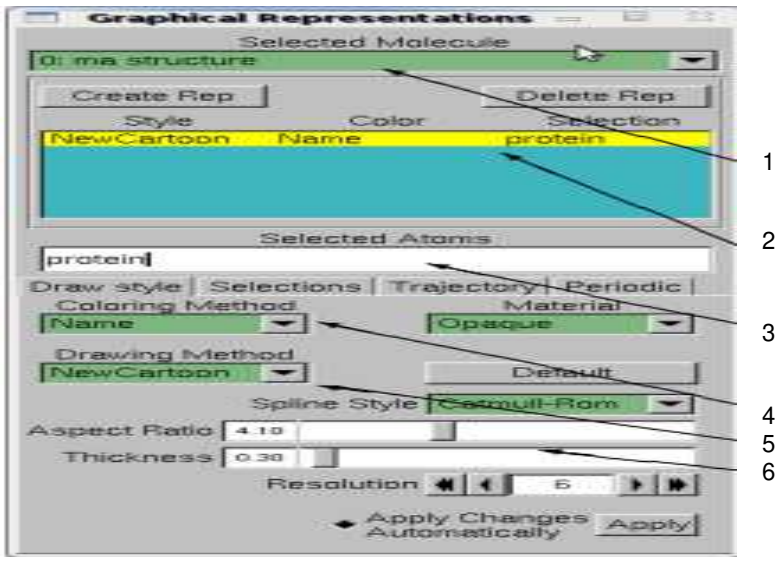


Figure 10 : Image représentant la fenêtre*GraphicalRepresentation*

1--> Sélection de la molécule ou la structure sur laquelle agir.

2--> Création/Suppression/Liste des représentations. Chaque représentation correspond à une

sélection d'une partie de la structure et `a son mode de vue.

3 -->Sélection des résidus groupe de résidus etc...[10]

L'onglet *Draw style* permet de choisir :

4 -->Le mode de couleur.

5 -->Le type de représentation.

6 -->Des options liées à la couleur ou à la représentation. [10]

III.3. Formats de fichiers de représentation des molécules

De nombreux formats de fichiers sont utilisés en modélisation moléculaire. En voici quelques-uns:

- **pdb**: Les caractéristiques du format .pdb sont décrites en détail sur le site de la Protein Data Bank[11].
- **mol** : ce format a été introduit par Molecular Design Limited[12]. Il permet la représentation de molécules ou de fragments disjoints tels que les composés ioniques.
- **xyz** : (Minnesota SupercomputerCenter's) est surtout utilisé pour gérer les animations.

Parmi ces formats on a choisi Mol2 pour représenter notre molécule.

IV. Le fichier MOL2

Un fichier Tripos MOL2 (.mol2) est une représentation complète, portable d'un SYBYL molécule [12]. C'est un fichier qui contient toutes les informations nécessaires pour reconstruire une molécule SYBYL.

Cette section décrit le format de fichier MOL2 détaillé de sorte que nous pouvons écrire nos propres programmes pour lire et écrire des fichiers MOL2.

IV.1. Avantage de MOL2

Les fichiers MOL2 sont écrits dans un format libre pour éviter les restrictions créées par les fichiers de format fixe. Les lignes d'un fichier .mol2 ont été numérotées pour faciliter la consultation [12].

IV.2. Exemple de fichier MOL2

```
#
#     Creating user name:       user InsightII
#     Creation time:            Fri Mar 29 22:58:26 2013

@<TRIPOS>MOLECULE
LIG_ORIG_3BFC
   48    48     1     0     1
SMALL
USER_CHARGES
@<TRIPOS>ATOM
      1 C1         -0.0599    1.5099    0.1879 C.3       1 TEMP1     -0.1590 ****
      2 C2         -0.0100   -0.0159    0.0862 C.3       1 TEMP1      0.0603 ****
      3 H2          0.4690   -0.3022   -0.8501 H         1 TEMP1      0.0530 ****
      4 O4          0.7378   -0.5423    1.1843 O.3       1 TEMP1     -0.5374 ****
      5 L41         1.7561   -0.1286    1.1404 L         1 TEMP1      0.0000 ****
      6 L42         0.7621   -1.6383    1.0935 L         1 TEMP1      0.0000 ****
      7 C5         -1.4332   -0.5762    0.1231 C.3       1 TEMP1     -0.0530 ****
      8 H5         -1.9122   -0.2900    1.0594 H         1 TEMP1      0.0530 ****
      9 C7         -2.2219   -0.0211   -1.0349 C.2       1 TEMP1      0.3508 ****
     10 O8         -3.2188    0.6288   -0.8301 O.2       1 TEMP1     -0.3964 ****
     11 L81        -3.5512    0.8156    0.2017 L         1 TEMP1      0.0000 ****
     12 L82        -3.7944    1.0341   -1.6753 L         1 TEMP1      0.0000 ****
     13 C9         -1.3834   -2.1021    0.0215 C.3       1 TEMP1      0.1578 ****
     14 H9         -0.8051   -2.5184    0.8463 H         1 TEMP1      0.0530 ****
     15 C11        -2.8074   -2.6863    0.0158 C.3       1 TEMP1     -0.1060 ****
     16 C12        -2.9210   -3.3963   -1.3137 C.2       1 TEMP1     -0.0120 ****
     17 S13        -4.3400   -4.2648   -1.8941 S.3       1 TEMP1     -0.0530 ****
     18 L131       -4.2110   -4.6156   -2.9286 L         1 TEMP1      0.0000 ****
     19 L132       -5.2411   -3.6340   -1.8791 L         1 TEMP1      0.0000 ****
     20 C14        -4.3976   -5.6139   -0.6889 C.3       1 TEMP1     -0.0410 ****
```

Figure 11 : Image représentant la partie TRIPOS MOLECULE et ATOM de fichier MOL2.

```
@<TRIPOS>BOND
     1     1     2 1
     2     1    33 1
     3     1    34 1
     4     1    35 1
     5     2     3 1
     6     2     4 1
     7     2     7 1
     8     4    36 1
     9     4     5 1
    10     4     6 1
    11     7     8 1
    12     7     9 1
    13     7    13 1
    14     9    10 2
    15     9    37 1
    16    10    11 1
    17    10    12 1
    18    13    14 1
    19    13    27 1
    20    13    15 1
    21    15    16 1
    22    15    38 1
    23    15    39 1
    24    16    17 1
    25    16    26 2
    26    17    20 1
    27    17    18 1
    28    17    19 1
    29    20    21 1
```

Figure 12 : Image représentant la partie TRIPOS BOND de fichier MOL2.

IV.3. Format de fichier MOL2

Chacune des sections suivantes dans cet exemple consiste en une description, des champs dans l'enregistrement de données, une description détaillée des données.

IV.3.1. Les types d'enregistrements dans les fichiers MOL2

Parmi les types d'enregistrement on trouve :

IV.3.1.1.Type d'enregistrement Indicateur (RTI)

Une chaîne de caractères imprimables par une arobase(@) Dans la colonne 1, terminée par une fin de ligne. Il est utilisé pour indiquer le type de données qui suit dans un fichier .mol2[13].

- ❖ Quelque type d'enregistrementd'indicateur :

• @<TRIPOS>ATOM:

- ➢ Chaque enregistrement de données associé à cette RTI se compose d'une seule ligne de données. Cette ligne de données contient toutes les informations nécessaires pour reconstruire un atome. [13]

- ❖ **Dans ce type de RTI :**

- La colonne 1 définit le numéro de l'identificateur.

- La colonne 2 définit le nom de l'atome.

- La colonne 3 définit où l'atome est situé.

- La colonne 4 définit le type de l'atome.

- La colonne 5 est appartient à la sous-structure avec une ID1.

- La colonne 6 définit La charge associée à l'atome.

• @<TRIPOS>MOLECULE:

- ➢ Chaque enregistrement de données associé à cette RTI se compose de six lignes de données :

- La première ligne de données est le nom de la molécule.

- La seconde ligne de données contient le numéro des atomes, des obligations, des sous-structures, les caractéristiques et les ensembles associés à la molécule.

- La troisième ligne de données est le type de molécule.

- La quatrième ligne de données indique le type de frais associés à la molécule.

- La cinquième ligne de données contient l'interne SYBYL bits d'état associés à la molécule.

- La dernière ligne de données contient le commentaire qui peut être associé avec la molécule. [13]

• **@<TRIPOS>BOND:**

- Chaque enregistrement de données associé à cette RTI se compose d'une seule ligne de données qui contient toutes les informations nécessaires pour reconstruire une obligation dans la molécule. [13]
- **Exemple:**

@<TRIPOS>BOND

1 1 2 1

- La liaison dans l'exemple a le numéro d'identification 1 et relie les atomes 1 et 2.

- Il s'agit d'une liaison simple (1).

- ✓ Apres la lecture du fichier mol2 de l'exemple ci-dessus en utilisant l'outil VMD, on a obtenu ce qui suit:

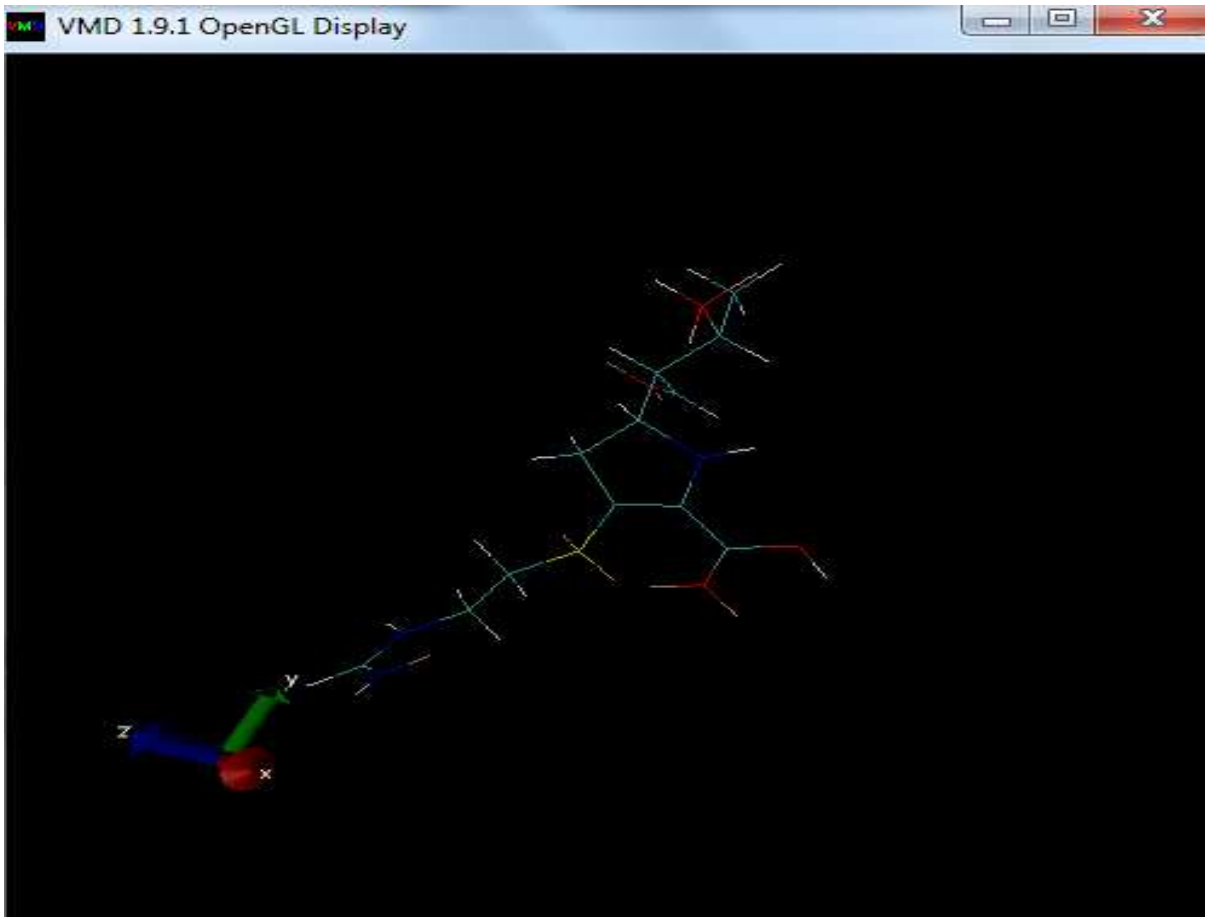


Figure 13 : Image représentant la fenêtre OpenGlL Display de l'outil VMD.

- ❖ Le logiciel VMD affiche la molécule en 3D, Cette molécule contient un seul cycle.

V.Borland C++ Builder

C++Builder est un **IDE**(un environnement de développement intégré). Il regroupe tout un ensemble d'outils permettant d'effectuer un maximum de tâches de développement au sein du même environnement de travail.[14]

C++ Builder est de plus un environnement de développement visuel C++ RAD (Rapid Application Development). Il permet de construire rapidement des applications en utilisant des composants et simplifie au maximum l'écriture du code et la réalisation de l'interface. On peut ainsi très rapidement se consacrer à la partie "métier" du code (le code réellement utile de l'application). [14]

C++ Builder permet le développement multi-plateformes (Windows et Linux) au moyen d'une librairie spéciale, la CLX. Si l'on ne se soucie que de Windows,la VCL, utilisée pardéfaut,estégalement disponible. [14]

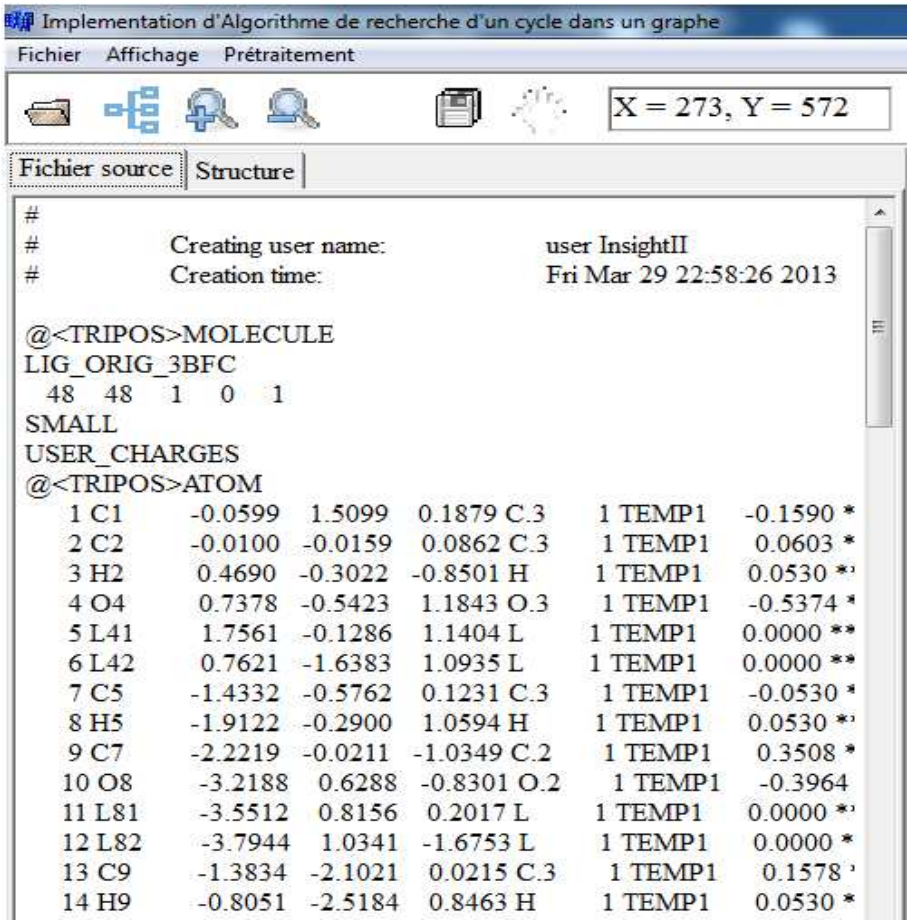
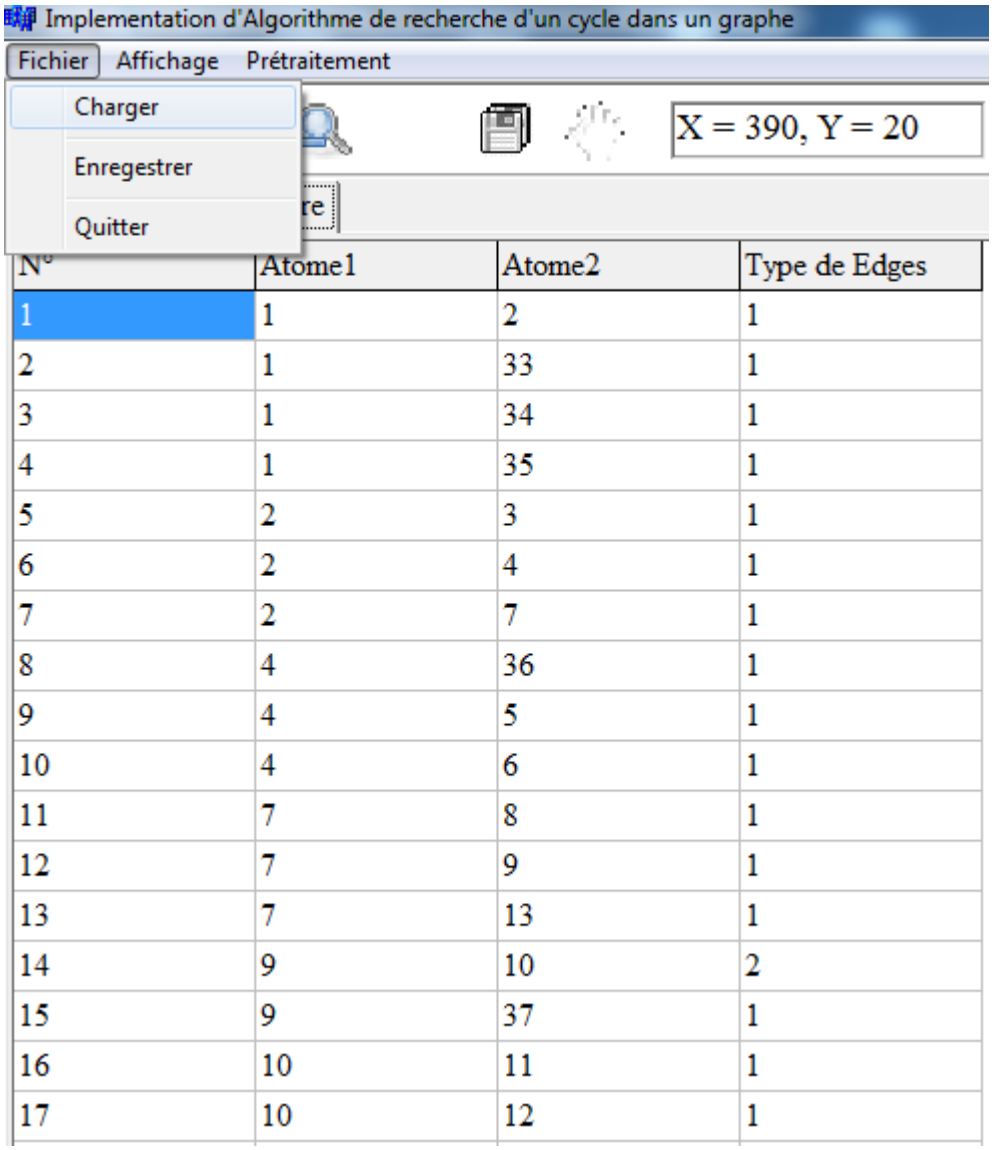


Figure14 : Image représentant l'interface.



N°	Atome1	Atome2	Type de Edges
1	1	2	1
2	1	33	1
3	1	34	1
4	1	35	1
5	2	3	1
6	2	4	1
7	2	7	1
8	4	36	1
9	4	5	1
10	4	6	1
11	7	8	1
12	7	9	1
13	7	13	1
14	9	10	2
15	9	37	1
16	10	11	1
17	10	12	1

Figure 15 :Image pour charger le fichier source de lig9.

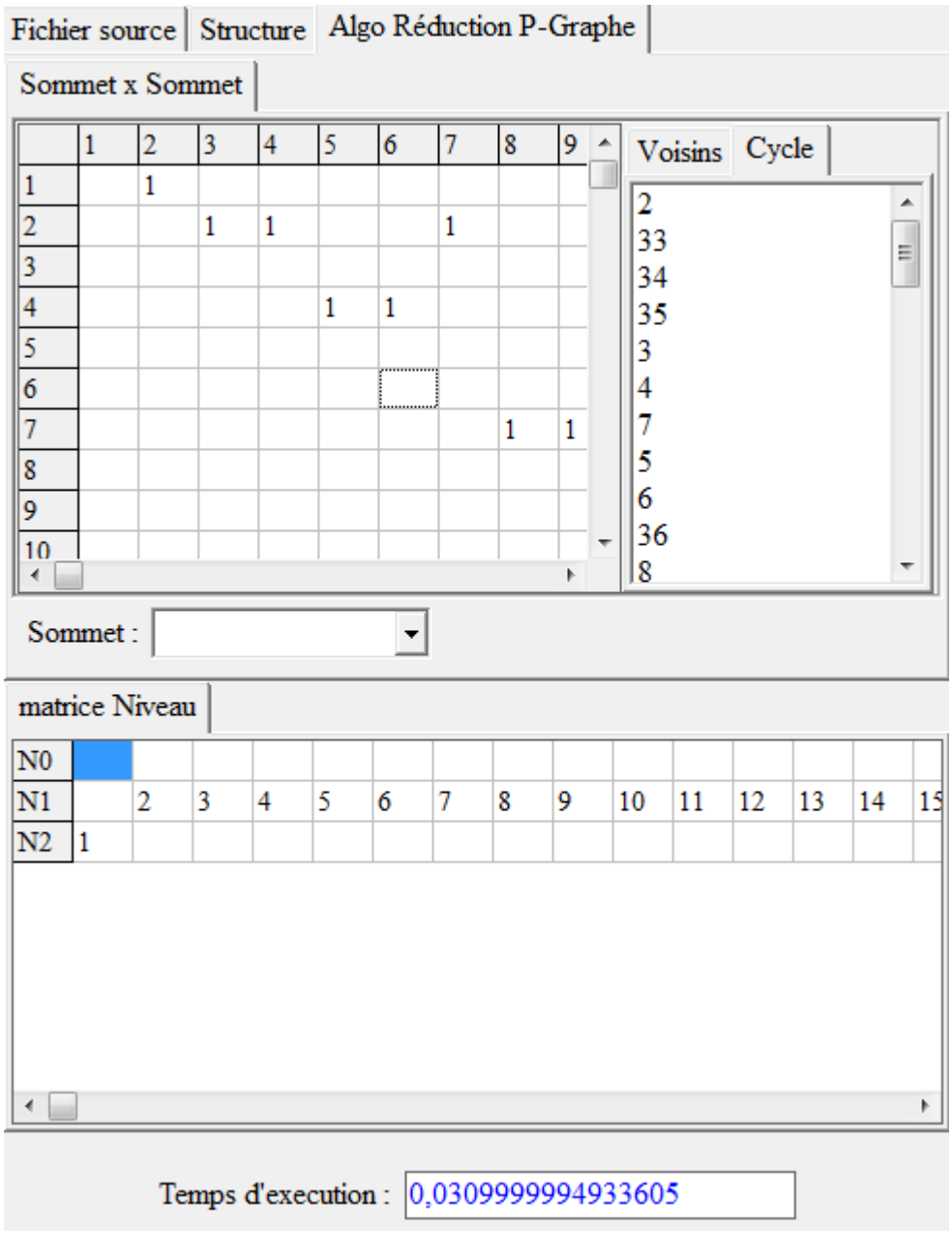


Figure 16 : Image représentant le temps d'exécution d'algorithme de Réduction P-Graphe.

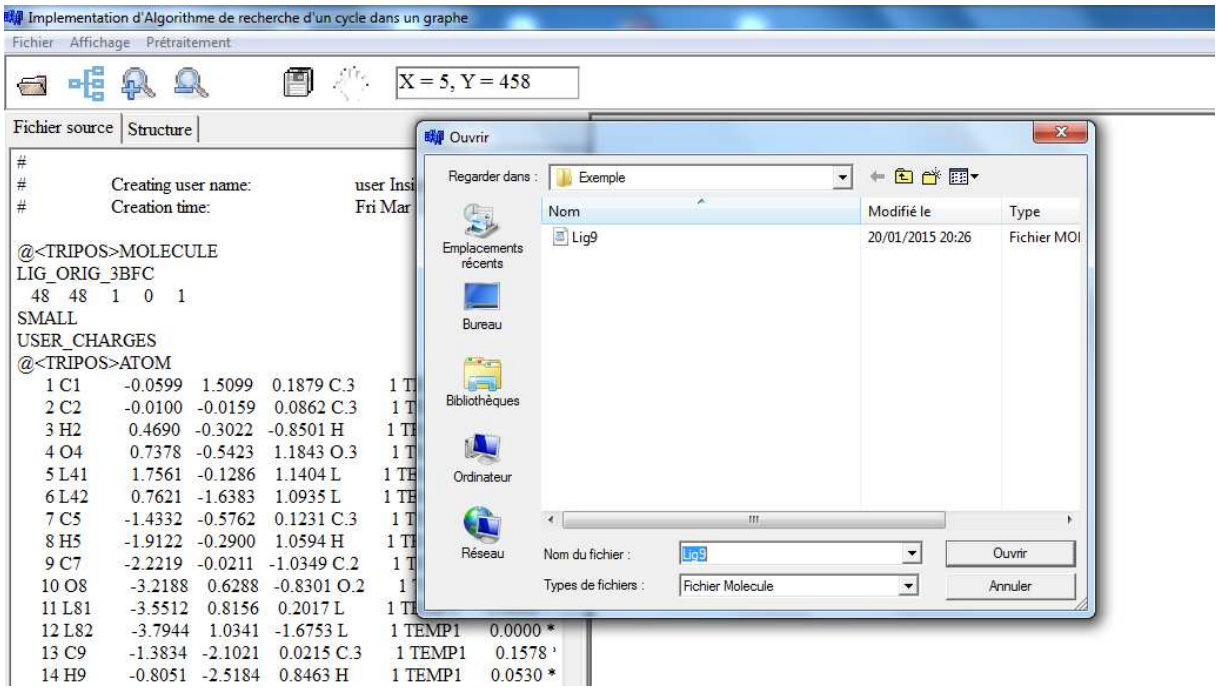


Figure 17 : Image charger un fichier lig9.

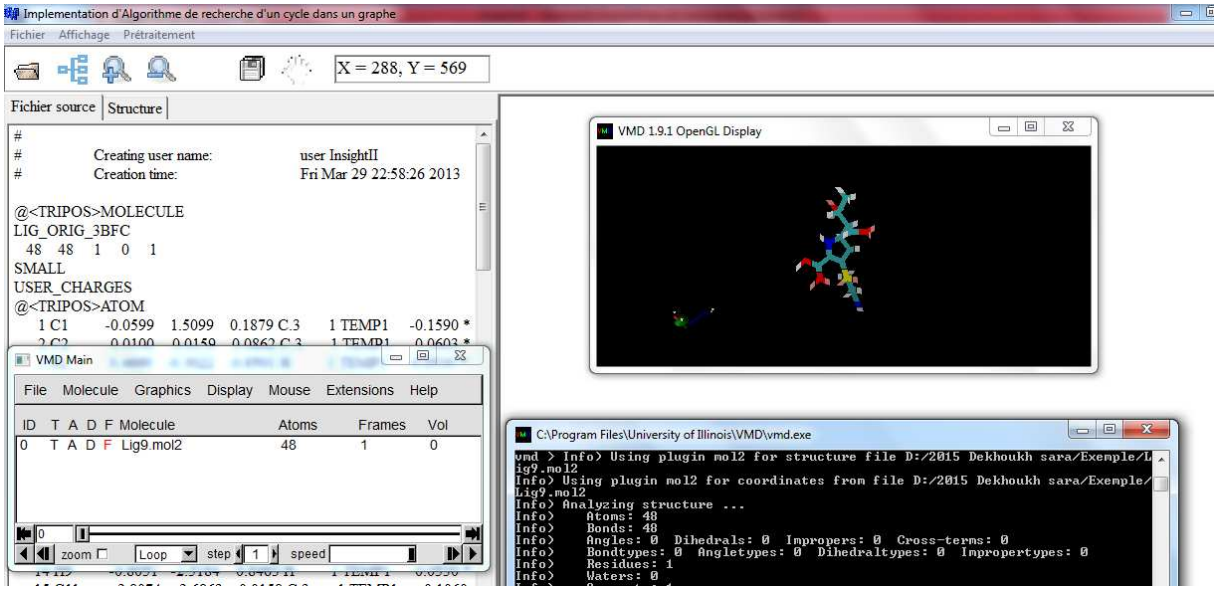


Figure 18 :Image représentant le rechargement de VMD .

```
 MGraphe  *TMG;
MGraphe mg;

  void MGraphe::creationMGraphe(int nbP, TStringGrid *T)
  {
        TMG = new MGraphe[T->RowCount-1];
        for(int i=1;i<T->RowCount;i++)
        {
            MGraphe pt;
            pt.idp       = i;
            pt.nbvoisin  = 0;
            pt.listvoisin= NULL;
            pt.x         = 0;
            pt.y         = 0;
            pt.typeLien  = StrToInt(T->Cells[3][i]);
            TMG[i-1]        = pt;
```

Figure 19 : Image représentant le code source de M-Graphe.

```
void TForm2::Extraction(TMemo * M, TStringGrid * T)
{
T->ColCount    = 4;
T->RowCount    = 2;
T->Cells[0][0]= "N°";
T->Cells[1][0]= "Atome1";
T->Cells[2][0]= "Atome2";
T->Cells[3][0]= "Type de Edges";
```

Figure 20 :Image représentant le code source de structure.

VI. Conclusion

Dans ce chapitre nous avons vu le logiciel Borland C++ Builder et nous avons également présenté quelques outils utilisés pour la représentation des molécules (le format mol2), ainsi que la visualisation de ces molécules (le logiciel VMD).

CONCLUSION GÉNÉRALE

Nous avons détaillé à travers nos chapitres les étapes à effectuer pour réaliser notre projet.

Bien que de nombreux algorithmes aient été développés pour atteindre la perception de l'anneau dans les graphes moléculaires, nous avons choisi l'algorithme de réduction de P-Graphe, nous avons choisi cet algorithme car il peut être appliqué à d'autres fins telles que l'énumération d'un trajet par exemple, et pour son principe qui est simple et qui peut être mis en œuvre de manière très efficace, de même qu'il peut être appliqué à n'importe quel graphe.

Ainsi que nous avons aussi choisi l'algorithme de parcours en largeur pour faire une comparaison entre les deux algorithmes.

Afin d'accomplir ce projet nous avons désigné le logiciel C++ Builder pour implémenter notre application, cette dernière nous permis de détecter les cycles dans un temps record à travers l'algorithme de réduction P-Graph.

Bibliographie

[1] : Liris-2387 : "http://liris.cnrs.fr/Documents/Liris-2387.pdf ", Consulté le : 12/12/2014.

[2] : Atelier A 6, programme de visualisation : "http://www.faidherbe.org/site/cours/dupuis/udpd.htm", résumé udpd.htm", Consulté le : 17/01/2015.

[3] : Site algorithme sur le graphe :" http://www.liafa.jussieu.fr/~carton/Enseignement/Algorithmique/Programmation/Graphs/", Consulté le : 14/12/2014.

[4] : Coursgraphe01.pdf : "http://www.uqac.ca/rebaine/8INF805/coursgraphe01.pdf" Consulté le : 12/03/2015.

[5]: A New Algorithm for Exhaustive Ring Perception in a Molecular Graph, Th. Hanser, Ph. Jauffret, and G. Koufmann , Consulté le :21/12/2014.

[6] : A robust method for searching the smallest set of smallest rings with a path-included distance matrix , Consulté le: 16/04/2015.

[7] : Explora :" http://www.xplora.org/ww/fr/pub/xplora/library/software/vmd.html "résume vmd.htm, Consulté le: 02/01/2015.

[8] : Molécule Dynamique :" http://www.ks.uiuc.edu/Research/vmd/", Consulté le : 05/01/2015.

[9] :Site pharmaxchange.info : "http://pharmaxchange.info/press/2011/04/visual-molecular-dynamics-vmd/", Consulté le : 09/01/2015.

[10] :Site gvallver.perso :"http://gvallver.perso.univ-pau.fr/doc/tut_vmd.pdf ",Consulté le : 12/03/2015.Conclusion et bibliographie.docx

[11] : Site Protein Data Bank : http://www.rcsb.org/pdb/home/home.do , consulté le : 17/01/2015.

[12] : Site Molecular Design Limited : "http://www.mdli.com", Consulté le :17/01/2015.

[13] : Site Tripos Mol2 File Format : "http://www.tripos.com/data/support/mol2.pdf ", consulté le : 07/01/2015.

[14] : Site :"http://cpp.developpez.com/faq/bcb/?page=Le-logiciel-Cplusplus-Builder#Qu-est-ce-que-Borland-Cplusplus-Builder",Consulté le 23/04/2015.