Compression en diamants de maillages tetrahédriques

Gabriel Beauplet, Luca Castelli Aleardi

Stage MPRI
1er août 2019

1 Abstract

2 Introduction

Un maillage représente un domaine géometrique en le discretisant en formes simples. Les maillages permettent de représenter des objets géométriques en 1, 2 ou 3D à des fins scientifiques ou industrielles par exemple. En 2D, les maillages representent des surfaces et sont constitués de polygones (triangles, carrés...) reliés deux à deux par une arête. En 3D, les maillages représentent des volumes à l'aide de polyèdres (tétraèdres, pyramides...) partageant une face commune. Les maillages sont très utilisés pour la visualisation de volume, calculs de solutions pour des équations aux dérivées partielles... Cependant, les maillages sont des structures complexes qui peuvent devenir très volumineuses et dont on essaye de réduire la taille.
La compression de données est omniprésente en informatique, avec des formats compressés génériques comme *gzip* mais aussi dédiés comme *mp3* pour les fichiers audios. Ce besoin de compresser les données est grandissant car de plus en plus de fichiers sont stockés à distance sur des serveurs et la moindre économie de stockage a d'importantes répercussions. Néanmoins, sous certains formats compressés, les données originales deviennent inutilisables (ex : *rar*). Cela pose problème quand l'on souhaite accéder aux données sans passer par l'étape de décompression.
Une structure de données est une manière d'organiser les données pour faciliter leur traitement. Les listes, arbres et graphes sont des exemples de structures de données. Leur but n'est pas de limiter l'usage mémoire mais seulement de faciliter l'utilisation des données. Ainsi, pour certains formats volumineux, des structures de données compactes ont été inventées. Ce sont des structures de données compressées, des structures de données dont l'utilisation de la mémoire est limitée.
Si l'on revient aux maillages, les algorithmes de compression limitent au maximum l'usage mémoire du maillage et il devient inutilisable sous forme compressé tandis que la structure de données pré-traite le maillage en réduisant l'usage mémoire mais celui-ci reste utilisable. Les maillages en deux dimensions sont majoritairement utilisés car ils sont plus léger, et permettent de représenter implicitement des volumes (la frontière). Par conséquent, de nombreuses structures de données compactes ont été créees afin de faciliter leur utilisation. Les maillages 3D étant beaucoup moins utilisés pour l'instant, peu de structure de données compactes leur sont dévolues. Néanmoins, leur utilisation croissante incite à procéder de même. La suite de ce rapport sera principalement consacrée aux structures de données compactes pour maillages 3D tétraèdriques.

De manière générale, un structure de données pour maillage stocke trois types d'informations :
- La géometrie, c'est à dire les positions des sommets
- La connectivité, les relations d'adjacence entre les tétraèdres
- Des attributs (par sommet, par arête, par face, par tétraèdre)

La structure de données doit supporter des requêtes simples :
- Quels sont les sommets de la ième face ?
- Quel est le degré du ième sommet ?
- Quelles sont les tétraèdres adjacents au ième tétraèdre ?
- ...

Par ailleurs, suivant l'utilisation ciblée, la structure de données devra être en mesure de satisfaire des opérations de modification :

— Ajouter/Enlever un sommet
— Ajouter/Enlever/Séparer un tetra

On évalue une telle structure de données en analysant le temps nécessaire à sa construction, à l'exécution d'une requête, à une opération de modification et surtout en observant la quantité de stockage nécessaire pour l'utiliser.

Contributions. Dans ce rapport, nous présentons une structure de données permettant de représenter la connectivité d'un maillage tetrahedrique en utilisant en moyenne 2.4 références par tétraèdre. Notre structure s'appelle *Tétraèdres en diamant*. Elle permet par ailleurs l'accès au ième sommet, au ième tétrahèdre et à l'étoile d'un sommet en temps constant. Son implémentation est simple et l'utilisation d'un tableau d'entiers afin de représenter les références permet une interopérabilité entre les languages de programmation.
Nous allons d'abord définir les principaux termes utilisés et rapeller les algorithmes de compression et structures de données déjà développés en deux et trois dimensions. Puis nous présenterons le fonctionnement, les avantages et inconvénients de notre structure de données. Par ailleurs, nous comparerons notre structure de données avec d'autres structures en terme de stockage mémoire ou de coût de calcul.

2.1 Définitions

Simplexe. Un simplex σ^p de dimension p est l'enveloppe convexe de $p+1$ points $\{v_0, v_1, ...v_p\}$, où $v_i \in R^n$ et les vecteurs $v_1 - v_0, v_2 - v_0...$ sont linéairement indépendants. Les simplexes de dimensions 0, 1, 2 et 3 sont respectivement les sommets, arêtes, triangles et tétraèdres.

Complexe simplicial. Un complexe simplicial est un ensemble K de simplexes d'un espace affine tel que toutes les faces de chaque simplexe de K appartiennent aussi à K et si deux simplexes σ et τ de K sont adjacents alors $\sigma \cap \tau \neq \emptyset$.

Variété. Une variété topologique (manifold en anglais) M de dimension n est un espace topologique connexe séparé localement homéomorphe à un ouvert de $\mathbb{R}^n$. C'est à dire que chaque point de M admet un voisinnage homéomorphe à un ouvert de $\mathbb{R}^n$.

Variété à bord. Une variété à bord est un sous-espace topologique dont les points admettent un voisinage homéomorphe à $\mathbb{R}^n$ (les points intérieurs) ou un voisinage homéomorphe à $\mathbb{R}^{n-1}x\mathbb{R}^+$ (les points bordants). L'ensemble des points bordants constitue le bord de la variété.

Frontière. Les (k-1)-simplexes d'une k-variété M qui sont incidents à seulement un k-simplexe sont les simplexes frontières. L'ensemble des simplexes frontières est dénoté ∂M.

Maillage. Un maillage est un complexe simplicial representant un objet géométrique. Il à la même dimension que l'objet qu'il représente. Ainsi, pour tout objet en 1, 2 ou 3D, les maillages respectifs seront en dimensions 1, 2 ou 3. Dans un maillage de dimension d, les simplexes de dimensions (d-1) sont appelées des facettes. Ainsi, les facettes d'un tétraèdre sont ses faces, les facettes d'une face sont ses arêtes et les facettes d'une arête sont ses sommets.

Le degré. Le degré d'un k-simplexe est le nombre de (k+1)-simplexes adjacents. Ainsi le degré d'un sommet est le nombre d'arêtes adjacentes et le degré d'une arête est le nombre de faces adjacentes.

Etoile. L'étoile d'un sommet est l'ensemble des k-simplexes adjacents à un sommet. C'est l'ensemble des triangles (resp. tétraèdres) adjacents à un sommet dans le cas surfacique (resp. volumique).

2.2 Combinatoire

En mathématique et en optimisation combinatoire, la caracéristique d'Euler χ est un invariant topologique décrivant la forme d'un objet géométrique.

$$\chi = \sum_{i=0} (-1)^i \, |dim(H_i)| = 2 - 2g \tag{1}$$

— H_i est l'ensemble des faces de dimension i
— g est le genre (le nombre de trou de l'objet étudié)

Ainsi, pour un polytope de dimension 4, la formule d'Euler devient :

$$\chi = |V| - |E| + |F| - |T| \tag{2}$$

— V est l'ensemble des sommets
— E est l'ensemble des arêtes
— F est l'ensemble des faces (ie. polygone)
— T est l'ensemble des tétraèdres

Sachant que chaque face qui n'est pas sur les bords du volume est partagée par deux tétraèdres, alors :

$$|F| \simeq 2|T| \tag{3}$$

Ainsi :

$$\chi \simeq |V| - |E| + |T| \tag{4}$$

3 Etat de l'art

Les maillages sont la pluspart du temps stockés sous forme indexés. Dans un premiers temps, on énumère pour chaque sommet ses coordonnées géométriques. Puis pour chaque face (resp. tétraèdre), les indices de ses 3 (resp. 4) sommets. D'autres attributs peuvent être stockés (normales, couleurs...) mais nous n'en discuterons pas ici. Les formats indexés ne sont pas les formats les plus concis pour sauvegarder des maillages. En effet, la connectivité occupe une place très importante. Tandis que dans les maillages 2D, le degré moyen des sommets est de 6, il est de 22 dans les maillages tétraèdriques. Par conséquent, dans ce type de format, un sommet apparaitra dans 22 tetras différents.
La mesure utilisée pour évaluer la qualité d'une compression est le nombre de bits par sommets (bits per vertex ou bpv). Tandis que la mesure utilisée pour évaluer la qualité d'une structure de données compacte est le nombre de références par triangle (resp. tétraèdre) rpt pour les maillages surfaciques (resp. volumiques).
On peut compresser un maillage en le simplifiant (supprimer des sommets...), en encodant la géométrie et/ou la connectivité du maillage. Nous ne nous intéresserons dans cette partie qu'à la compression de la connectivité puisque c'est la plus gourmande en mémoire. Par ailleurs, nous détaillerons d'abord le travail effectué en 2D puis en 3D. Bien que notre travail soit uniquement centré sur les maillages 3D, l'essentiel des travaux effectués jusqu'à présent est en 2D.

3.1 Maillages 2D

3.1.1 Compression

Bande de triangle. Les bandes de triangles ("triangles strips") et les eventails de triangles ("triangle fans") sont des représentations utilisées pour transférer les maillages de la mémoire centrale du PC vers la mémoire du GPU. Une bande de triangle est une séquence de sommet où chaque nouveau sommet défini un triangle avec les deux précedent sommets. En ce qui concerne les bande de triangles, le but est de trouver de très longues bandes. Si les bande de triangles sont suffisament longue alors, cette representation permet de passer de 3N references aux sommets à N+2. L'algorithme de Deering utilise ces bandes de triangles et utilise entre 3.3 et 9.8 bpv [1].

Traversée de triangles. La Cut border Machine [2] est un algorithme de Gumhold qui encode

la conectivité en parcourant le graphe en largeur. L'algorithme étend la frontière défini par un triangle initial en travesant itérativement des triangles adjacents. Sept symboles sont utilisés pour préciser si la frontière a été étendu en insérant un nouveau sommets, si la frontière a été séparée ou si deux frontière se sont jointes. Le schéma peut compresser des variétés avec 4 bpv. En revanche, ce résultat est seulement valide pour des maillages réguliers. En effet, quand une jointure est effectuée, un décallage doit etre fait pour désigner les sommets concernés. Par conséquent, il n'y a pas de borne supérieure garantie pour la compression avec cet algorithme.
L'algorithme EdgeBreaker de Rossignac [4] traverse lui aussi le graphe d'un triangle adjacent à un autre et enregistre la connectivité d'un maillage en produisant les sympboles C,L,R,E,S. Cependant, il garantit un cout de 4bpv.

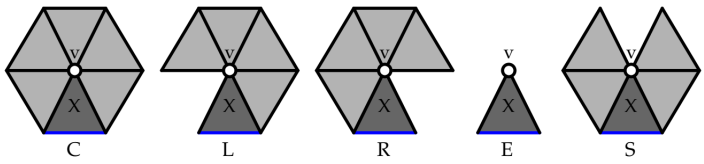

FIGURE 1 – Les cinq configurations dans l'algorithme Edgebreaker. v est le sommet central de la configuration et X est le triangle cible

Codage de la Valence. Une manière de décrire la connectivités de sommets est à travers leurs valence. Le premier travail sur la valence des sommets est le travail de Touma et Gotsman [6]. Le principe est de considérer la frontière d'un triangle initial et de l'étendre en ajoutant itérativement de nouveaux sommets. La connectivité est encodée en utilisant la valence des nouveaux sommets (concentrée autour de 6). Ainsi, La liste de valence des sommets peut être efficacement compressée par un encodeur d'entropie (2.3 bpv). C'est toujours aujourd'hui l'une des méthodes les plus efficace.

3.1.2 Structure de données compacte

Plusieurs structures de données permettent une utilisation très facile des maillages et se focalisent sur l'utilisation des arêtes du graphe. C'est le cas d'Half-Edge, Winged-Edge et Quad-Edge qui stockent respectivement $18n$ et $9n_t$ références. Elles permettent facilement de naviguer dans le maillage et opèrent des requêtes d'adjacence en temps constant. Cependant, elles occupent trop de place pour être considérées comme compactes.
La Corner Table (CT) est à la base de plusieurs structures de données. Elle utilise deux listes V et O de $3|F|$ entiers chacune. La table V stocke les incidence triangle/sommet tel que les 3 sommets bordant un triangle t sont consécutifs (V[3t],V[3t+1],V[3t+2]) et sont listés dans un ordre consistent avec le maillage. Ainsi, V[c] représente un coin c associé avec une face f et un sommet. La table O stocke la référence entière du coin oposé. Le coin opposé o(c) au coin c est un coin dans un triangle adjacent qui partage la même arête opposée.

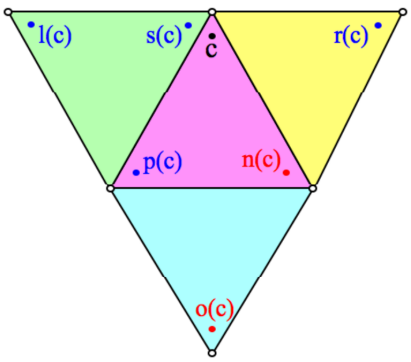

FIGURE 2 – Les opérateurs utilisant les coins pour un maillage triangulaire

VOT. La structure de données VOT (Vertex Opposite Table) est la première structure de données à utiliser cette "Corner Table". Elle permet une répresentation simple et efficace des maillages avec 6 références par triangle (3 références pour les sommets dans la table V et 3 référence pour les coins dans la table O).
SOT. SOT, développée par Rossignac [9] est une amélioration de VOT où la table O est réordonnée et la table V supprimée. Néanmoins, l'accès au coin d'un sommet et à l'étoile d'un sommet sont toujours en temps constant. Cette dernière structure de données utilise 3 rpt en moyenne.

3.2 Maillages 3D

3.2.1 Compression

Grow&Fold. L'algorithme Grow&Fold [7] combine les idées de l'algorithme Topological Surgery [5] de Taubin et EdgeBreaker [4] de Rossignac. Il construit un arbre couvrant de tétraèdre et un folding string. L'arbre couvrant début à une face arbitraire et grandit en ajoutant des tétraèdres aux faces externes de l'actuel arbre couvrant. Pour chaque ajout de tétraèdre, 3 bits encodes si d'autres tétraèdres seront attchés aux 3 faces extérieures de ce tétraèdre. Le folding string contient pour chaque triangle externe de l'arbre couvrant un code sur 2 bits permettant de retrouver les relations d'incidences absentes de l'arbre couvrant. L'abre couvrant contient $|T|$ tétraèdres et il y a $2|T|$ faces externes. Par conséquent, l'usage mémoire est de 7 bpt.
Cut Border Machine. La Cut Border Machine pour les maillages volumiques [2] est directement inspirée de celle pour les maillages surfaciques de Gumhold [3]. L'algorithme étend la frontière défini par une face initiale en traversant des tétraèdres adjacents. Dix symboles sont utilisés pour décrire l'entourage de la frontière lors de l'ajout d'un nouveau sommet pour la construction d'un tétraèdre. Leur algorithme permet de compresser les maillages tétraèdriques en utilisant 2.4 bpt et s'adapte aux non-variétés.

3.2.2 Structure de données compacte

VOT. La Corner Table a été adapté par Rossignac aux maillages tétraèdriques (VOT). Elle demande 8 rpt (4 pour les sommets et 4 pour les coins opposés). Un index dans ces listes identifie un coin particulier à un tétraèdre. Ainsi, les tables O et V ont toutes les deux $4|T|$ entrées. Les coins de chaque tetrahèdre sont consécutifs dans les deux listes (les quatres coins du ième tetra sont stockées aux entrées 4i+j, where j = 0,1,2,3) et sont listés dans un ordre consistent avec l'orientation du tétrahèdre (les sommets des coins j=1,2,3 apparaissent dans le sens inverse des aiguilles d'une montre depuis le sommet du coin 0).
Bande de Triangles. Weiler et al. [8] encode les tétrahèdres en bandes. L'inclusion d'une petite quantité d'informations d'adjacence leur permet d'acceder aux faces voisines en temps constant. Leur algorithme stocke en moyenne 5.1 rpt.
SOT. La dernière structure de données développée est SOT [9] par Rossignac. Elle améliore sa première structure de données VOT en triant la table O et en supprimant la table V. La structure de données utilise 4 références et 9 bits par tétraèdre en moyenne et permet l'accès à l'étoile d'un sommet en temps constant.

4 Diamond Compression

4.1 Le principe

Dans citation, ils regroupaient les tetrahedres deux à deux afin d'économiser une reference d'un tetrahedre à l'autre. L'idée ici est de regrouper les tetrahedres partageant une même arrete. On appelle un tel regroupement un diamant. Au sein d'un diamant, les tetrahedres sont ordonnés. On peut donc oublier les références de voisinage entre deux tetrahedres du même diamant (le tetrahedre 7 est connecté au tetra 6 et 8). Ainsi, pour chaque tetrahedre d'un diamant, il est connecté à deux tetras du même diamant (un avant et un après). Ainsi, seul, pour chaque tetra du diamant, seul deux references sont necessaires afin de savoir quels sont les deux tetras adjacents n'appartenant pas au diamant. Ainsi, dans le cas parfait, si tous les tetras étaient regroupés en diamants, alors la structure de données n'utiliserait que deux references par tetra.

4.2 Appariement tetras en diamants

La première étape est de regrouper les tetras en diamants. C'est à dire, trouver un ensemble maximum d'arêtes tel qu'aucune arête n'appartienne au même tetra. Les tetras n'ayant aucune arête dans cet ensemble seront appelés des tetras isolés. Pour trouver cet ensemble d'arêtes, nous avons essayés plusieurs algorithmes.

mettre image du cas de figure que l'on ne veut pas (deux aretes qui appartiennent au même tetra)

4.2.1 Choisir l'arête la plus pentu pour chaque sommet

La première méthode consiste à prendre pour chaque sommet, l'arête étant la plus pentu. Néanmoins, cette méthode a deux incovénients majeurs : deux arêtes peuvent être choisi et appartenir au même tetra, elle utilise la géometrie du domaine (et donc peut sembler moins générique).

4.2.2 BFS

La seconde approche consiste à parcourir le maillage en largeur. On choisit un tetrahedre au début de l'algorithme puis on regarde pour chacune de ses arêtes si elle forme un diamant et si aucun des tetrahedres n'est deja dans un diamant. Si c'est le cas, on ajoute ce nouveau diamant, sinon, on ajoute les tetras adjacents à la file. On continue cet algorithme tant que tous les tetras n'ont pas été parcouru.

Choix du Tetrahedre de départ Le choix du tetrahedre de départ n'affecte que très légerement la qualité de l'algorithme.

mettre image de l'incidence du tetra de début

BFS ou DFS DFS semble donner des résultats moins bon que BFS.

Inconvénient En utilisant un outil de visualisation de maillages 3D réalisé par Luca Castelli, on s'appercoit que les tetra isolés sont essentiellement sur les bords du domaine. Par conséquent, nous avons essayé de constituer les diamants d'abord sur les bords mais sans reussite.

4.3 Algorithme randomisé

L'idée est d'utiliser un algorithme randomisé afin de selectionner les arêtes centrales des diamants. On assigne donc un 0 ou 1 à chaque arête suivant si la solution la contient ou non. Puis on calcule la qualité de la solution en utilisant une fonction de fitness qui assigne +1 pour chaque tetrahedre dans un diamant et -1 si deux arêtes appartiennent au même tetrahedre.

L'avantage de cet algorithme est qu'il permet de trouver des solutions de meilleures qualités. Cependant, son execution est plus longue et ne garantie aucune propriété sur la solution finale.

4.4 BFS+algorithme randomisé

La solution finale est donc un mélange de BFS et d'algorithme randomisé. BFS offre une solution convenable en un temps très court. Néanmoins, sa solution peut etre améliorer avec un algorithme randomisé.

Nous avons donc sélectionné un ensemble d'arêtes representant les diamants. Tous les tetrahedres ne possedant pas une de ces aretes sont dit isolés.

4.5 Choisir l'ancre

Il est necessaire d'acceder au ième sommet facilement. Afin de ne pas ajouter de references, on associe à chaque edge central d'un diamant un sommet. Ainsi, pour chaque diamant, il y a un sommet affecté à celui-ci. Si un sommet n'a aucun diamant voisin (que des tetras isolés) alors, on associe le sommet à un des tetras isolés. Si un sommet n'a aucun diamant de libre dans son voisinnage alors, on explose un diamant (on supprime le diamant, et on fait comme si il n'y avait que des tetras isolés) et on affecte le sommet à un de ces tetras isolés.

montrer la figure représentant tous les edges centraux

4.6 La structure

La structure est un tableau à une dimension dont la taille est le nombre de faces extérieures des diamants ou tetras isolés. Ainsi un diamant contenant 4 tetrahedres occupera 8 cellules dans le tableau (car il a 8 faces extérieures) et un tetra isolés en occupera 4. Afin de pouvoir acceder au sommet rapidement, on reordonner l'ordre des diamants afin que le ième vertex soit adjacent au ieme diamant. Par ailleurs, pour savoir quand on passe d'un diamant à un autre, nous utilisons des bits de service (1 ou 0) pour chaque face. Une face contient un 1 si c'est la première face d'un diamant, 0 sinon. Finalement, histoire de pouvoir tourner facilement autour d'un sommet, nous encodons dans des bits de service les permutations entre deux faces.

Pour résumer, voici notre structure de données :

- un tableau 1D de tailles —F— où F est l'ensemble des faces exterieures du maillage. - un bit de service afin de savoir si une face est la première d'un diamant - 3 bits de services afin de représenter les permutation entre deux faces

mettre image d'un tableau 1D mettre image d'une permutation

4.6.1 Ordre des sommets et tetrahedres

4.6.2 Calculer les permutations

mettre image de deux diamants qui se touchent et partagent une face

Tout comme les tetrahedres, les sommets peuvent etre ordonnés au sein d'un diamant. Un diamant de taille —D— (contenant D tetras) contient —D—+2 sommets. On ordonne d'abord les sommets se trouvant sur "l'équateur" du diamant puis les deux sommets adjacents à tous les tetras. Le premier sommet est alors le sommet adjacent au premier tetra et au dernier, le second sommet est celui adjacent au premier tetra et au deuxieme... Pour calculer les permutations entre deux faces, il suffit alors de comparer l'ordre des sommets des deux faces.

4.7 Les methodes possibles

Notre structure de données doit être à la fois compacte (nécessite moins de stockage que la version standard) et doit permettre de réaliser plusieurs opérations.

4.8 Acceder au ieme vertex

La première opération doit être de pouvoir acceder au ième sommet. Bien que nous ne stockions pas les sommets de manière explicites (nous ne stockons que les faces). Nous sommes en mesures de localiser le ième sommet car il est adjacent au ième diamant. Il nous suffit donc de parcourir les faces et de s'arreter après avoir parcouru i face avec un bit de service=1. Cet accès se fait en temps O(n).

4.9 Acceder au ieme tetra

Lors du regroupement des tetras en diamants, l'ordre des diamants n'est plus le même que l'ordre initial (à la lecture du fichier OFF). Néanmoins, on peut re-ordonner les tetras dans le fichier original afin que l'ordre des tetras soit les mêmes. De cette manière on peut acceder au ième tetra en O(n).

4.10 Degré d'un sommet et Hypersphere d'un sommet

Calculer le degré d'un sommet est plus compliqué. On sait que le le ième sommet est adjacent au ième tetra. Par conséquent, on si le ième tetra est un diamant alors toutes les faces paires de celui-ci sont adjacentes au sommet ciblé. Si le diamant est un diamant isolé, alors le sommet cible est opposé à la première face et est donc adjacent aux trois autres faces. Puis, il suffit de suivre les diamant adjacents à chacune de ces faces.

Quant à l'hypersphere d'un sommet, c'est exactement le même procédé que pour calculer le degré.

4.11 Evaluation de notre structure de donneés

4.11.1 Comment evaluer notre structure

4.12 Resultats

comparaison avec les autres structures

4.13 Borne inférieur théorique

4.14 Encoder les diamants en off

Il est possible de sauvegarder notre structure dans un format succins. Pour rappel, dans notre tableau, le ieme sommet est adjacent au ième diamant. Il suffit pour cela de calculer l'hypersphere pour chaque sommet. De cette manière, pour chaque sommet, on assigne un ensemble de diamant adjacents à ce sommet. Il suffit alors de renverser la structure, ou pour chaque diamant on obtient les sommets adjacents (les sommets qui le compose).

En procedant de cette manière, on économise la sauvegarde de nombreux sommets. En effet, dans un diamant contenant 3 tetras, chaque tetra reference 1 fois les deux sommets situés aux poles et les sommets situés sur l'équateur sont référencés deux fois. Ainsi, il y a les sommets du diamant sont références au total 12 fois. En revanche, dans notre cas, chaque sommet n'est référencé qu'une seule fois. Ainsi on économise 2 fois plus de place.

4.15 Améliorations

Cette section vise à répertorier les améliorations possible à notre structure.

4.15.1 Références différentielles

Dans notre tableau T, chaque face est représenté par un indice sur 32 bits, qui est la taille miimale d'un entier en C++. Néanmoins, on peut representer l'adjacence par la sa distance entre les deux faces dans le tableau. Nous n'avons pas implémenté cette fonction car le gain semblait trop faible.

4.15.2 Rank and select

4.16 Bornes theoriques

4.17 Dynamicité

4.18 Defaults

Notre structure de données n'est pas exempte de défaults.

Le premier défaut est qu'il est nécessaire de lire l'intégralité des données afin de pouvoir lancer notre algorithme de compression. Si les tetras sont ordonnés pas proximité au sein du fichier original. Alors notre algorithme peut former les diamants en disposant d'un nombre limité de tetras. En revanche, si les tetras ne sont pas ordonnés, alors aucune garantie ne peut etre apporté sur le nombre de tetras nécessaires en cache pour former les diamants. C'est le défault principal qui constitue un vrai goulot d'étranglement.

5 NP completude

Le problème : Existe-il une couverture des tetras en diamants contenant plus de k tetras ?

Apparier les tetras en diamants est l'étape clé de notre algorithme. Malheureuseument, nous montrons dans cette section que ce problème est NP-Complet. En effet, notre problème consiste à choisir un ensemble d'aretes tel qu'aucune de ces arêtes n'appartiennent au même tetra. On peut alors créer un autre graphe où chaque arête du graphe initial est modélisé par un sommet. Deux sommets dans ce nouveau graphe sont connectés si et seulement si leur arete dans le graphe original appartiennent au même tetra. La création de ce nouveau graphe se fait en temps polynomiale (en fonction des arêtes).

Dans ce nouveau graphe, notre problème devient le même que le maximum independant set (MIS). A savoir que l'on recherche un ensemble maximum de sommets (donc d'aretes dans notre graphe initial)

tel qu'aucun de ces sommets de soient connectés (donc que leur aretes appartiennent au même tetra dans le graphe initial). En revanche, notre instance du MIS est une instance pondérée, où chaque sommet à un poids correspondant au nombre de tetra adjacent à l'arete dans le graphe initial.

6 Implémentation

Tout est implémenté en C++ natif, sans aucune bibliothèque extérieure. Le tout est compilé en C++17 sur une machine avec un processeur i5-5300U et 16Go de RAM.

6.1 Vitesse

7 Travail futur

7.1 Comment rendre la structure dynamique

8 Conclusion

Références

[1] Deering, *Geometry Compression.*

[2] Stefan Gumhold, *Improved Cut-Border Machine for Triangle Mesh Compression.*

[3] Stefan Gumhold, Stefan Guthe, Wolfgang Straßer, *Tetrahedral Mesh Compression with the Cut-Border Machine.*

[4] Jarek Rossignac, *Edgebreaker : Connectivity compression for triangle meshes.*

[5] Gabriel Taubin, Jarek Rossignac, *Geometric Compression Through Topological Surgery.*

[6] Costa Touma and Craig Gotsman, *Triangle Mesh Compression.*

[7] Andrzej Szymczak, Jarek Rossignac, *Grow&fold : compression of tetrahedral meshes.*

[8] Manfred Weiler, Paula N. Mallon, Martin Kraus, Thomas Ertl, *Texture-Encoded Tetrahedral Strips.*

[9] Topraj Gurung, Jarek Rossignac, *SOT : Compact representation for tetrahedral meshes.*