



**HAL**  
open science

# Représentation et optimisation de maillage structuré par blocs à l'aide de systèmes multi-agents

Valentin Postat

## ► To cite this version:

Valentin Postat. Représentation et optimisation de maillage structuré par blocs à l'aide de systèmes multi-agents. Système multi-agents [cs.MA]. Université Paris-Saclay, 2024. Français. NNT : 2024UP-ASG003 . tel-04535496

**HAL Id: tel-04535496**

**<https://theses.hal.science/tel-04535496v1>**

Submitted on 6 Apr 2024

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Représentation et optimisation de maillage structuré par blocs à l'aide de systèmes multi-agents

*Representation and optimization of block-structured  
meshes using multi-agent systems*

**Thèse de doctorat de l'université Paris-Saclay**

École doctorale n°580 d'accréditation, sciences et technologies de l'information et de la  
communication (STIC)

Spécialité de doctorat : Informatique

Graduate School : Informatique et sciences du numérique. Référent : Faculté des sciences  
d'Orsay

Thèse préparée dans l'unité de recherche **LIHPC** (Université Paris-Saclay, CEA), sous la direction  
de **Franck LEDOUX**, Directeur de recherche, sous le co-encadrement de **Guillaume HUTZLER**,  
Maître de conférences à l'Université Evry-Val d'Essonne

Thèse soutenue à Évry, le 24 janvier 2024, par

**Valentin POSTAT**

## Composition du jury

Membres du jury avec voix délibérative

**Éric ANGEL**

Professeur des Universités, Université d'Evry Val d'Essonne

**Guillaume DAMIAND**

Directeur de recherche, Université de Lyon

**Gauthier PICARD**

Directeur de Recherche, Université de Toulouse

**Sylvie ALAYRANGUES**

Maître de conférences, Université de Poitiers

**Valérie CAMPS**

Maître de conférences, Université de Toulouse

Président

Rapporteur & Examineur

Rapporteur & Examineur

Examinatrice

Examinatrice

**Titre :** Représentation et optimisation de maillage structuré par blocs à l'aide de systèmes multi-agents

**Mots clés :** Maillage, Hexaèdre, Système multi-agents

**Résumé :** Ce travail de thèse porte sur la représentation et la génération de maillages hexaédriques structurés par blocs. Il n'existe pas à ce jour de méthode permettant de générer des structures de blocs satisfaisantes pour n'importe quel domaine géométrique. En pratique, des ingénieurs experts génèrent ces maillages avec des logiciels interactifs, ce qui nécessite parfois plusieurs semaines de travail. De plus, l'ajout d'opérations de modification dans ces logiciels interactifs est un travail délicat pour maintenir la cohérence de la structure de blocs et sa relation avec le domaine géométrique à discrétiser.

Afin d'améliorer ce processus, nous proposons tout d'abord de définir des opérations de manipula-

tion de maillages hexaédriques se basant sur l'utilisation du modèle des cartes généralisées. Ensuite, en considérant des structures de blocs obtenues à l'aide de la méthode des Polycubes, nous fournissons des méthodes optimisant la topologie de ces structures pour satisfaire des contraintes de nature géométrique. Nous proposons ainsi une première méthode en dimension 2, qui considère une approche locale du problème en s'appuyant sur l'expérience des ingénieurs manipulant des logiciels interactifs. Puis nous proposons une seconde méthode utilisant cette fois la méta-heuristique d'optimisation par colonie de fourmis pour la sélection de feuillets en dimension 3.

**Title :** Representation and optimization of block-structured meshes using multi-agent systems

**Keywords :** Meshing, hexahedra, multi-agent system

**Abstract :** This thesis deals with the representation and generation of block-structured hexahedral meshes. To date, there is no method for generating satisfactory block structures for any geometric domain. In practice, expert engineers generate these meshes using interactive software, which can take several weeks to complete. Moreover, adding modification operations in these interactive softwares is a delicate task to maintain the coherence of the block structure and its relationship with the geometric domain to be discretized.

In order to improve this process, we first pro-

pose to define hexahedral mesh manipulation operations based on the use of the generalized map model. Then, by considering block structures obtained using the Polycube method, we provide methods for optimizing the topology of these structures to satisfy constraints of a geometric nature. We propose a first method in dimension 2, which considers a local approach to the problem based on the experience of engineers working with interactive software. We then propose a second method, this time using ant colony optimization meta-heuristics for leaf selection in dimension 3.

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>7</b>
1.1	Du modèle physique à l'expérimentation numérique . . . . .	7
1.2	Maillage pour la simulation numérique . . . . .	8
1.3	Manipulation et optimisation de maillages structurés par blocs . . . . .	10
1.4	Positionnement et contributions . . . . .	13
1.4.1	Utilisation des cartes généralisées pour assurer la robustesse . . . . .	13
1.4.2	Approches multi-agents pour la modification des structures de blocs et pour la sélection de feuillets . . . . .	14
<b>2</b>	<b>Notion et concept</b>	<b>17</b>
2.1	Qu'est-ce qu'un maillage ? . . . . .	17
2.2	Opérations globales sur les maillages hexaédriques . . . . .	19
2.3	Classification géométrique . . . . .	20
2.4	Qualité d'un maillage . . . . .	22
2.4.1	Qualité géométrique d'un hexaèdre . . . . .	23
2.4.2	Lien entre valence, qualité géométrique et alignement au bord . . . . .	24
2.5	Représentation d'un maillage . . . . .	25
2.6	Méthodes de génération de maillage . . . . .	26
2.6.1	Overlay-grid . . . . .	27
2.6.2	Utilisation des champs d'orientation . . . . .	27
2.6.3	Polycube . . . . .	28
2.6.4	Selective padding . . . . .	29
2.6.5	Méthodes interactives . . . . .	29
<b>3</b>	<b>Représentation et opérations sur maillage structuré par blocs</b>	<b>31</b>
3.1	Représentation d'une structure de blocs à l'aide des cartes généralisées . . . . .	32
3.1.1	Cartes généralisées et topologie . . . . .	32
3.1.2	Orbites et classification géométrique . . . . .	36
3.1.3	Opérations de modifications atomiques . . . . .	37
3.2	Opérations de manipulation de structures de blocs hexaédriques . . . . .	39
3.2.1	Sélection de feuillets . . . . .	39
3.2.2	Suppression de feuillet . . . . .	41
3.2.3	Insertion de feuillet . . . . .	45
3.3	Conclusion et perspectives . . . . .	55
<b>4</b>	<b>Optimisation de polycube 2D par application de motifs et d'un système multi-agent</b>	<b>59</b>
4.1	Qu'est ce qu'un système multi-agent ? . . . . .	59
4.2	Description du problème de satisfaction de contraintes d'alignement en 2D . . . . .	60
4.3	Pourquoi utiliser un système multi-agent ? . . . . .	62

4.4	Définition de notre système multi-agent . . . . .	63
4.4.1	Définition des actions . . . . .	65
4.4.2	Définition des observations . . . . .	69
4.4.3	Politique des agents réflexes locaux . . . . .	72
4.4.4	Agent coordinateur . . . . .	72
4.5	Algorithme global et exécution pas à pas . . . . .	73
4.6	Validation expérimentale du système multi-agent . . . . .	74
4.7	Conclusion et perspectives . . . . .	76
<b>5</b>	<b>Colonie de fourmis pour piloter l'insertion de feuillets 3D</b>	<b>81</b>
5.1	Introduction à la méta-heuristique OCF pour des problèmes combinatoires . . . . .	82
5.2	Méta-heuristique OCF pour l'optimisation de structures de blocs en dimension 2 . . . . .	83
5.2.1	Définition du problème 2D à résoudre . . . . .	84
5.2.2	Qualité d'un chemin d'arêtes en 2D . . . . .	85
5.2.3	Construction d'un chemin solution par une fourmi . . . . .	86
5.2.4	Algorithme général . . . . .	87
5.3	Méta-heuristique OCF pour l'optimisation de structures de blocs en dimension 3 . . . . .	93
5.3.1	Définition du problème à résoudre . . . . .	94
5.3.2	Qualité d'une sélection de face en 3D . . . . .	94
5.3.3	Construction d'une sélection de faces par une fourmi . . . . .	95
5.3.4	Algorithme général . . . . .	96
5.4	Résultats expérimentaux en dimension 3 . . . . .	101
5.4.1	Recherche de paramètres "optimaux" . . . . .	101
5.4.2	Expérimentations sur plusieurs géométries et structures de blocs . . . . .	106
5.4.3	Limites observées expérimentalement . . . . .	112
5.5	Conclusion et perspectives . . . . .	115
<b>6</b>	<b>Conclusion</b>	<b>117</b>
6.1	Perspectives de travail . . . . .	117
6.1.1	Opérations robustes sur structures de blocs hexaédriques . . . . .	117
6.1.2	Vers des systèmes multi-agents plus intelligents . . . . .	118

## Remerciements

Premièrement je remercie Franck Ledoux et Guillaume Hutzler pour la direction de ma thèse. Guillaume pour nos échanges d'idées. Franck bien sûr pour tous tes conseils, encouragements et ton implication durant la thèse.

Je souhaite ensuite remercier Guillaume Damiaud et Gauthier Picard pour la clarté de leurs rapports, ainsi que Éric Angel, Sylvie Alayrangues et Valérie Camps qui m'ont donné l'honneur d'accepter de faire partie de mon jury.

Je tiens à remercier certains membres du laboratoire LIHPC. Agnès, Alain et Céline pour m'avoir aidé sur des questions administratives ou matérielles toujours avec patience et gentillesse. Je remercie aussi Hélène, Brigitte et Tao.

Merci à Nicolas qui m'a accompagné lors de ma première conférence à Bordeaux, avec un choix d'hôtel discutable de ma part !

Merci aussi à Simon et Romain pour des discussions parfois tardives aussi bien sur le travail que sur d'autres sujets plus farfelus...

Je remercie aussi tous les doctorants et stagiaires que j'ai pu rencontrer pour leur bonne humeur indispensable, Julie, Alexiane, Victor, Axelle, Paul, Claire, Manuel, Michael, Maël, Alexandre, Marie Corizo, Marie A., Marie T., Sébastien, Luc, Clément G., Cassandra, Clément, Maxim, Éloïse, Vanman.

Finalement, je remercie mes parents, ma soeur et Louiza pour tout le soutien qu'ils m'ont apporté durant ces années de thèse.



# 1 - Introduction

La simulation numérique est aujourd'hui un outil incontournable dans de nombreux domaines industriels : transport, énergie, aéronautique, aérospatiale, géosciences, astrophysique, etc. D'une part, elle permet de simuler des phénomènes physiques très complexes à un coût moindre que celui de l'expérimentation classique. Simuler un nouveau modèle d'avion ou de voiture nécessitera potentiellement de nombreuses heures de calcul sur un super-calculateur puissant et un temps ingénieur conséquent mais cela restera tout de même grandement inférieur au coût de la construction réelle de l'objet (avion ou voiture) et de la réalisation d'une ou plusieurs expérimentations. D'autre part, certains phénomènes sont tout simplement impossibles à réaliser expérimentalement : simuler la naissance d'une étoile en astrophysique, la rupture d'un barrage hydraulique, ou les effets d'un tremblement de terre aux abords d'une ville.

## 1.1 . Du modèle physique à l'expérimentation numérique

En pratique, pour simuler un phénomène physique à l'aide d'un ordinateur, et ce quelle que soit la complexité du phénomène, il est nécessaire de considérer trois grandes phases qui vont se succéder :

1. Tout d'abord le problème physique à étudier doit être modélisé. C'est le travail du physicien qui représente le problème en question par un ensemble de systèmes d'équations. Par exemple, un physicien en mécanique des fluides modélisera l'écoulement d'un liquide dans des tuyaux à l'aide d'équations aux dérivées partielles. Le modèle "vît" ici dans un espace continu - en dimension 3, ce sera par exemple  $\mathbb{R}^3$ . Le système précédemment défini permet de trouver de manière analytique des solutions de référence pour des géométries simples, comme un tuyau droit. Si maintenant la géométrie se complexifie, et que l'on dispose d'un réseau complet de tuyaux avec des branchements, des robinets, des rétrécissements, des élargissements, etc., la solution analytique au problème sera bien plus complexe à définir.
2. Une manière de fournir une solution numérique "expérimentale" consiste alors à approcher ce problème continu par un problème "discret" exprimable par un ordinateur. On discrétise alors à la fois le temps - l'écoulement n'est pas continu, on l'observe à différents instants - et l'espace - on découpe l'espace en éléments simples - comme des briques de Lego, ou des pièces d'un puzzle. Les méthodes numériques, comme la méthode des différences finies, celle des éléments ou celle des volumes finis, vont se baser sur ce découpage temporel et spatial pour permettre de discrétiser les modèles définis au préalable de façon continue.



- Enfin informatiquement, la complexité des problèmes modélisés, leur taille ou encore la complexité du domaine physique étudié - un moteur complet d'avion par exemple - nécessite de manipuler des données informatiques très complexes et de grande taille. La problématique est alors d'écrire des programmes informatiques exécutables sur des supercalculateurs nécessitant de maîtriser différents niveaux de parallélisme.

## 1.2 . Maillage pour la simulation numérique

Dans ce travail, nous nous intéressons à la discrétisation spatiale du problème, et plus précisément à la notion de *maillage pur*. Le maillage d'un domaine  $\Omega \subset \mathbb{R}^n$ , avec  $n > 0$ , est un découpage, ou plus précisément un *partitionnement* de  $\Omega$  en un ensemble d'éléments simples de dimension  $n$ , appelés des mailles, et qui serviront de support pour les fonctions mathématiques utilisées dans les méthodes numériques. Il existe différents types de maillages selon la nature des mailles et la structure globale du maillage.

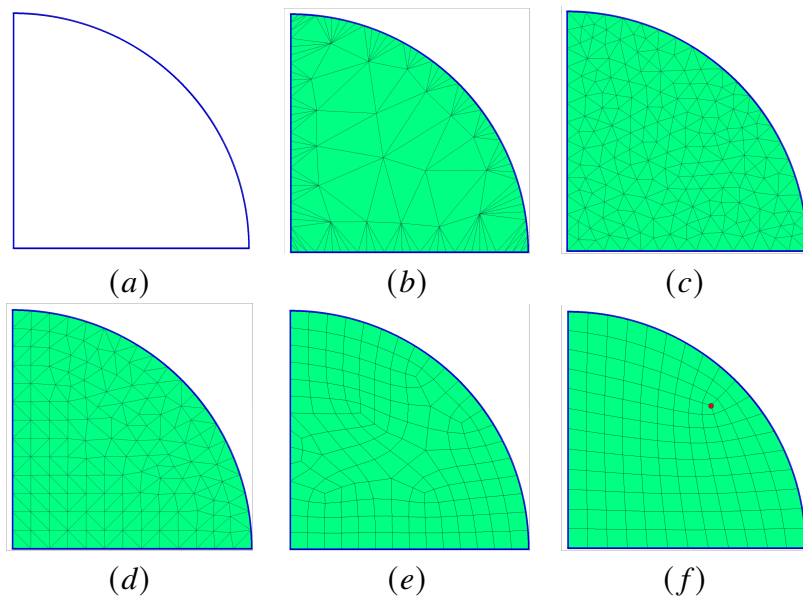


FIGURE 1.1 – Un domaine géométrique 2D en (a) est successivement discrétisé à l'aide de maillages triangulaires en (b), (c) et (d), et de maillages quadrangulaires en (e) et (f).

Considérons à titre d'exemple le domaine géométrique de dimension 2 représenté à la figure 1.1.a. Cinq maillages correspondant à ce domaine sont fournis de la figure 1.1.b à la figure 1.1.f. Les trois premiers maillages sont triangulaires mais diffèrent sur le choix des tailles des mailles (en particulier en 1.1.b) et la structuration liée à l'algorithme de création des sommets du maillage. L'algorithme pour générer le maillage de la figure 1.1.d est un algorithme frontal où les sommets sont créés par

couches successives en partant du bord du domaine. C'est pourquoi on identifie visuellement des structures quadrilatérales dans le maillage. Pour sa part, le maillage de la figure 1.1.c est le résultat d'une méthode itérative, qui partant du maillage présenté en 1.1.b va insérer des sommets dans le maillage jusqu'à avoir des triangles d'une taille prescrite et le plus isocèle possible. Enfin les maillages des figures 1.1.e et 1.1.f sont composés de quadrilatères uniquement, de tailles sensiblement équivalentes dans les deux cas, mais avec une structuration différente. On dira ainsi que le maillage de la figure 1.1.f est structuré par blocs car on peut détecter dans sa structure trois sous-grilles régulières connectées en un unique sommet singulier (le sommet rouge).

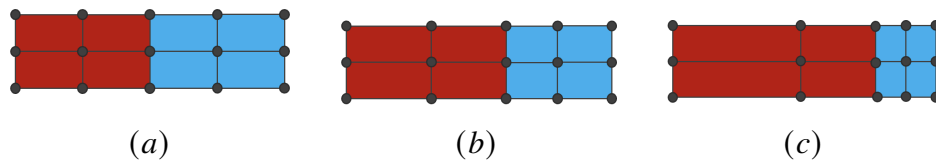


FIGURE 1.2 – Exemple de maillage pur se déformant au cours du temps pour suivre l'évolution du composant rouge et du composant bleu.

Quand au caractère *pur* du maillage, il nous indique que dans le cas où l'on partitionne un domaine géométrique composé de plusieurs composants, alors une maille fait partie d'un seul et unique composant. A titre d'exemple, considérons le cas d'une simulation dynamique, où un composant de couleur rouge va au cours du temps occuper plus d'espace et "pousser" un composant de couleur bleue. Cet exemple est illustré sur la figure 1.2 avec un maillage pur qui se déforme au cours du temps. A chaque étape, chaque maille correspond à un unique composant. Au contraire, sur la figure 1.3, le maillage est qualifié de mixte. Il reste ici statique, aucun de ses sommets ne se déplace pour suivre l'évolution des deux composants. Par contre, chaque maille contient des proportions du composant rouge et bleu qui évoluent.

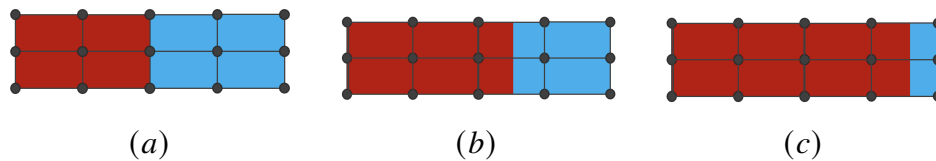


FIGURE 1.3 – Exemple de maillage mixte où les positions des sommets du maillage restent fixes au cours du temps, tandis que la proportion du composant rouge et du composant bleu dans chaque maille évolue.

Le choix des types de mailles à utiliser, de leur caractère pur ou mixte est en pratique lié aux caractéristiques des méthodes numériques utilisées par les codes de simulation numérique et par les capacités des outils de génération de maillages. Pour certaines méthodes numériques, il sera par exemple nécessaire de disposer uniquement

de maillages quadrangulaires structurés par blocs en dimension 2. Si tel est le cas, tous les outils de génération de maillage ne sont pas aujourd’hui capables de générer des maillages quadrangulaires structurés par blocs automatiquement. La personne chargée de l’étude numérique utilise alors des logiciels graphiques interactifs [1, 2, 3], et passera plusieurs heures potentiellement pour créer le maillage souhaité de manière semi-automatique. Si l’étude est menée en 3D et que le domaine doit être discrétisé par un maillage hexaédrique structuré par blocs, ce temps de création de maillage pourra dépasser plusieurs jours, voir des semaines pour des géométries aussi complexes qu’un moteur d’avion.

### 1.3 . Manipulation et optimisation de maillages structurés par blocs

Les travaux de cette thèse se positionnent justement dans le cadre de la génération de maillages hexaédriques structurés par blocs. Aujourd’hui, de nombreux domaines de simulation numérique modélisant la dynamique des fluides, des chocs, l’écrasement ou l’hydrodynamique nécessitent ou préfèrent utiliser des maillages hexaédriques structurés par blocs pour obtenir les résultats numériques et physiques escomptés. Ces maillages sont efficaces dans les simulations physiques fortement anisotropes (couches limites, ondes de choc, etc.), entre autres car la base tri-linéaire associée aux hexaèdres comporte des termes cubiques qui capturent les variations d’ordre supérieur et fournissent moins d’éléments, ce qui réduit le nombre de mailles, réduisant par là même le temps de simulation. Ils sont également très intéressants en termes de performance : contrairement aux maillages non structurés, la plupart de la connectivité d’un maillage hexaédrique structuré par blocs peut être implicitement déduite d’une structure de tableaux multidimensionnels sous-jacente. Cette structure est également utile pour faciliter l’accès rapide aux nœuds et cellules adjacentes, compte tenu de la structure indexée des tableaux.

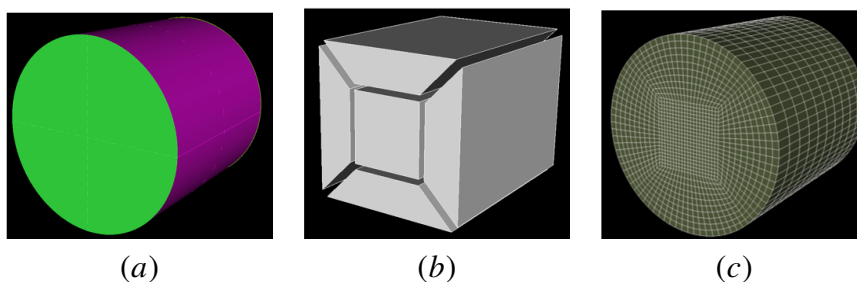


FIGURE 1.4 – Exemples d’entités dans la création d’un maillage structuré par blocs. En (a), le domaine géométrique à discrétiser - ici un cylindre ; en (b) une structure de blocs, ou macro-maillage, en O-grid ; en (c), le maillage hexaédrique final, où chaque bloc a été raffiné de manière uniforme par une grille.

Même si la recherche sur la génération de maillage hexaédrique est un domaine très actif [4], il est aujourd’hui nécessaire de disposer d’outils interactifs pour créer

les maillages hexaédriques, en particulier ceux structurés par blocs. Au CEA DAM, c'est le logiciel **MGX**, qui joue ce rôle. Il est utilisé pour générer de tels maillages et considère principalement des domaines géométriques de type CAO, qui sont représentés par leur bord. On parle alors de modélisation par les bords, ou BRep [5] pour *boundary representation*. Le maillage hexaédrique structuré par blocs est lui construit de manière hiérarchique en deux temps :

1. Tout d'abord, la structure de blocs est un maillage "grossier" composé de peu de blocs hexaédriques. On parle de macro-maillage (voir figure 1.4-b). Cette structure est générée automatiquement pour des géométries axi-symétriques simples ou des motifs élémentaires (cylindres, sphère, coques, pavés) ou assemblée de manière itérative et incrémentale par un ingénieur.
2. Une fois la structure de blocs créée, chacun des blocs est découpé en une grille régulière de mailles (voir figure 1.4-c). Le nombre et la taille des mailles dans chaque direction d'un bloc est une nouvelle fois contrôlée par l'utilisateur.

Notons que pour discrétiser correctement le domaine géométrique, chaque cellule du macro-maillage et du maillage final est associée à une entité du domaine géométrique (points, courbes, surfaces). On parle de *classification géométrique*.

La problématique principale de l'utilisation de maillages hexaédriques structurés par blocs est que leur génération n'est actuellement pas automatique ou alors pour des domaines simples et très spécifiques (obtenus par révolution ou extrusion de structures 2D). A titre d'exemple, considérons le processus illustré à la figure 1.5 où le logiciel **MGX** développé au CEA est utilisé pour générer la structure de blocs d'un objet 3D simple mais pour lequel aucune solution automatique ne donne aujourd'hui le résultat escompté. Partant du modèle illustré en 1.5-a, l'ingénieur va interactivement ajouter et modifier des blocs pour discrétiser l'objet. Le logiciel **MGX** fournira différentes aides à l'utilisateur mais ce dernier aura tout de même une succession importante d'étapes à réaliser manuellement. Sur les figures 1.5-b à 1.5-f, l'ingénieur va créer différents blocs en s'appuyant sur les courbes et surfaces du modèle initial, coller ces blocs entre eux et les découper pour bien "capturer" le domaine géométrique. En 1.5-g, une structuration, dite en O-grid<sup>1</sup> est créée dans la partie supérieure du domaine pour ensuite supprimer des blocs en 1.5-h afin d'être en conformité avec le trou cylindrique du domaine géométrique. Une fois cette étape réalisée, il reste à recoller la structure de blocs correspondant au cylindre troué avec le reste des blocs (voir fig. 1.5-j) et à régulariser la position des sommets de blocs avant de générer le maillage final (voir fig. 1.5-l). Ce processus prend une quinzaine de minutes à un ingénieur expert, ce qui est acceptable. Malheureusement, dès que la complexité de la pièce augmente, le temps nécessaire augmente énormément. La figure 1.6 illustre cela en considérant une étude menée en interne au CEA en 2018. Partant du domaine géométrique que nous avons discrétisé en blocs à la figure 1.5, le domaine est progressivement rendu plus

---

1. On parle de O-grid, car les blocs existants sont entourés chacun de quatre blocs dans une direction donnée et ces quatre blocs forment un O autour du bloc initial.

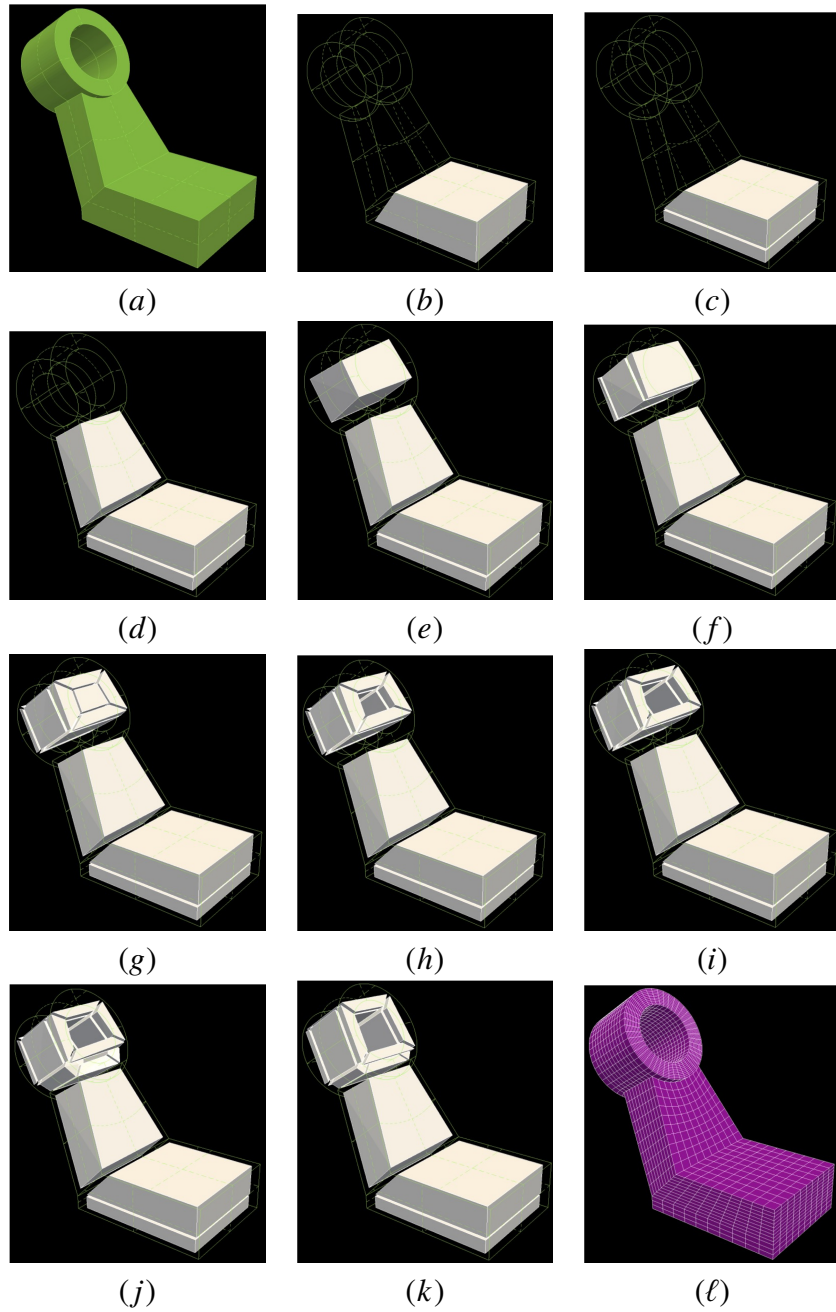


FIGURE 1.5 – Exemple de réalisation d'un maillage hexaédrique structuré par blocs à l'aide du logiciel **MGX** développé au CEA/DAM.

complexe. Comme indiqué sur la figure, le nombre de blocs augmente de manière importante, tout comme le temps passé pour obtenir le maillage. En pratique, il est courant de devoir passer plusieurs jours à plusieurs semaines pour obtenir une structure de blocs adéquate, et ce quel que soit le logiciel de maillage utilisé - **MGX**, Cubit[6], ICEM CFD-Hexa[3], etc.



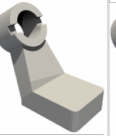
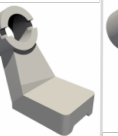
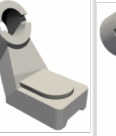






<b>GEOMETRIE</b>						
<b>TOPOLOGIE</b>						
<b> Blocs </b>	29 blocs	59 blocs	62 blocs	92 blocs	132 blocs	174 blocs
<b>Temps de création</b>	15 mins	25 mins	30 mins	1 heure	1.5 heure	2 heures

FIGURE 1.6 – Temps consacré à la réalisation d’un maillage hexaédrique structuré par blocs avec le logiciel **MGX** pour une suite de modèles de complexité croissante.

## 1.4 . Positionnement et contributions

Afin de faciliter la génération semi-automatique de structures de blocs au sein du logiciel **MGX**, une approche étudiée depuis quelques années au CEA consiste à créer automatiquement une structure non optimale en s’inspirant des méthodes de *Polycubes*. Ces méthodes permettent de discrétiser le domaine géométrique  $\Omega$  à l’aide d’une structure totalement régulière, que l’on peut voir comme étant un empilement de cubes, qui est ensuite déformée pour s’aligner avec le bord de  $\Omega$  (voir la figure 1.7).

### 1.4.1 . Utilisation des cartes généralisées pour assurer la robustesse

Afin de disposer d’opérations robustes et formellement définies, nous avons fait le choix d’utiliser le modèle des cartes généralisées [7] pour représenter rigoureusement à la fois la structure de blocs, les opérations de modification de blocs, mais aussi la relation de classification entre les cellules de la structure de blocs et les entités géométriques composant le modèle à discrétiser. Cette partie de nos travaux fait l’objet du chapitre 3.

### 1.4.2 . Approches multi-agents pour la modification des structures de

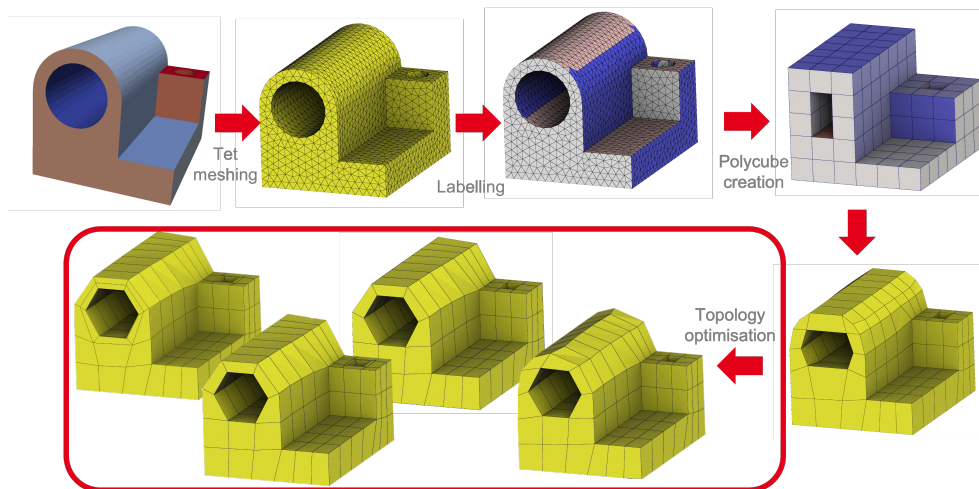


FIGURE 1.7 – Pipeline de génération de structures de blocs à l’aide de Polycubes et optimisation de la structure obtenue par contraction de cordes et insertion de feuillets.

### blocs et pour la sélection de feuillets

Pour améliorer la qualité d’une structure de blocs, nous considérons d’une part des opérateurs usuels de manipulation de blocs hexaédriques, à savoir les opérateurs d’insertion/suppression de feuillets (voir figure 1.8) et de contraction/expansion de cordes (voir figure 1.9); d’autre part l’application de motifs locaux de modification de bloc. Ces derniers sont utilisés pour des modèles géométriques en dimension deux au chapitre 4, tandis que les premiers sont utilisés sur des modèles géométriques en dimension trois au chapitre 5. Dans les deux cas, nous pilotons l’utilisation des opérateurs de modification en proposant des modèles de systèmes multi-agents. Choisir de tels modèles est original pour la communauté maillage en simulation numérique.

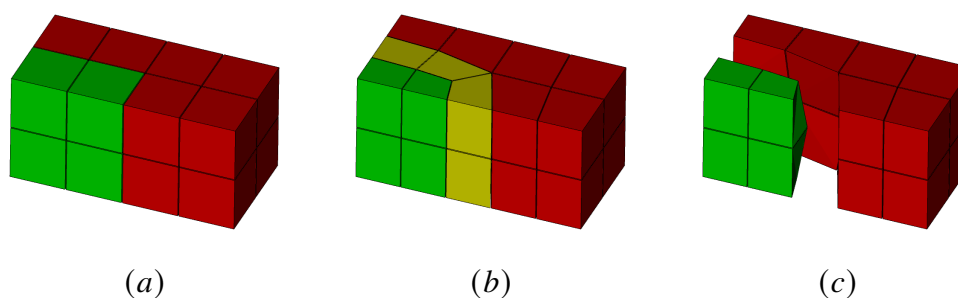


FIGURE 1.8 – Exemple d’insertion de feuillet au sein d’une structure de blocs hexaédriques. Une couche de mailles, i.e. un feuillet, est insérée entre les hexaèdres rouges et vert sur la figure (a). Après insertion, la couche jaune sépare les deux ensembles de mailles précédemment adjacents (en b et c).

Au chapitre 4, nous partons de l’hypothèse de base qu’un ingénieur sait indiquer

ce qui pose problème localement dans une structure de blocs. Par exemple, il dira qu'il souhaite remplacer tel bloc par un motif de trois autres blocs pour améliorer la qualité de la structure à cet endroit. Par contre, il ne saura pas dire comment cette résolution locale doit être étendue au reste de la structure de blocs pour que celle-ci reste valide.

Dans le chapitre 5, nous considérons l'utilisation des colonies de fourmis, il s'agit d'une métaheuristique permettant de résoudre certains problèmes combinatoires complexes. Cette fois cette méthode sera appliquée à un problème de sélection composée d'un ou plusieurs feuillets, ensuite une opération globale sur cette sélection sera effectuée.

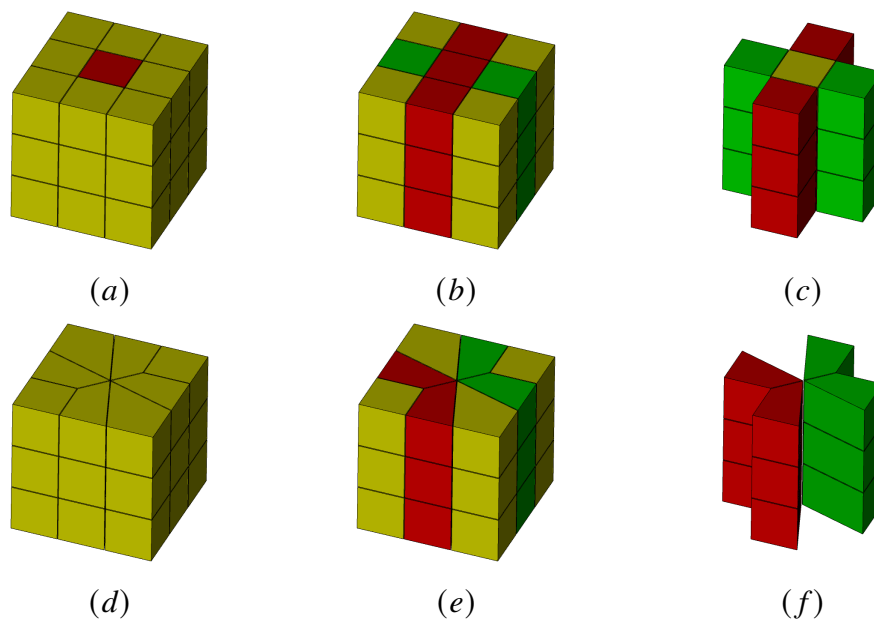


FIGURE 1.9 – Exemple de contraction de corde au sein d'une structure de blocs hexaédriques. Deux sommets opposés dans la face rouge en (a) sont sélectionnés pour contracter la colonne de mailles issue de la face rouge. Après contraction, on obtient le résultat présenté en (d), où la face rouge a disparu ainsi que la colonne de mailles correspondante. Les illustrations (b), (c), (e) et (f) permettent de voir que les feuillets qui s'intersectaient le long de la colonne de mailles sélectionnée ne s'intersectent plus. Elle se touche uniquement le long d'une ligne d'arêtes.





## 2 - Notion et concept

Dans ce chapitre nous introduisons les notions et concepts de maillage que nous jugeons nécessaires à la bonne compréhension des chapitres 3, 4 et 5 qui exposeront les contributions significatives de notre travail.

### 2.1 . Qu'est-ce qu'un maillage ?

Étant donné un domaine géométrique  $\Omega$ , un maillage est un partitionnement de  $\Omega$  en un ensemble d'entités élémentaires  $\{e_0, e_1, \dots, e_p\}$ , c'est-à-dire que :

- $\Omega$  est couvert par  $\{e_0, e_1, \dots, e_p\} : \Omega = \sum_{i=0}^n e_i$  ;
- L'intersection de tout couple d'entités élémentaires de  $\{e_0, e_1, \dots, e_p\}$  est vide :  
 $\forall (i, j) \in [0, p]^2, i \neq j \Rightarrow e_i \cap e_j = \emptyset$ .

En outre, chaque entité élémentaire  $e_i$  est bordée d'entités de dimension inférieures formant son bord. Ainsi une entité volumique est bordée d'un ensemble de faces, elles-mêmes bordées d'arêtes, elles-mêmes bordées de sommets. En dimension 3, les cellules volumiques les plus utilisées sont les tétraèdres et les hexaèdres. Sur la figure 2.1, le modèle géométrique présenté en (a) est successivement partitionné par un maillage tétraédrique en (b) et un maillage hexaédrique en (c).

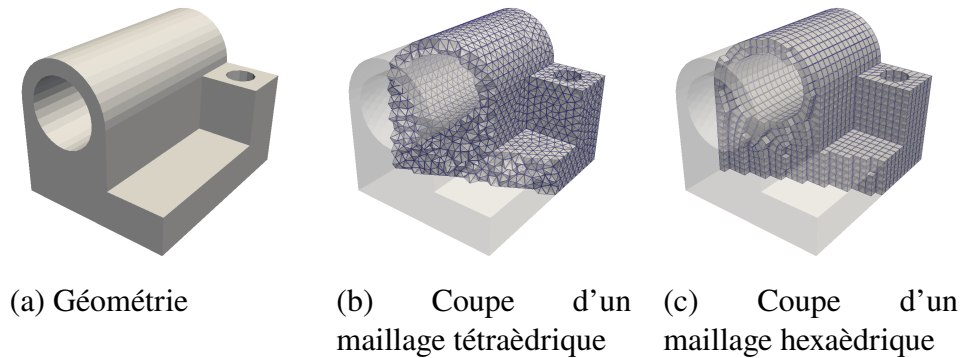


FIGURE 2.1 – Exemple de modèle géométrique en (a) partitionné par un maillage tétraédrique en (b) et hexaédrique en (c).

Nous considérons deux types de relations entre cellules d'un maillage : la relation d'incidence et la relation d'adjacence. Deux cellules  $c_1$  et  $c_2$  sont **incidentes** s'il existe un chemin de cellules, partant de la cellule de plus grande dimension vers l'autre cellule, tel que chaque cellule du chemin appartient au bord de la cellule précédente du chemin. Deux  $i$ -cellules  $c_3$  et  $c_4$  sont **adjacentes** s'il existe une cellule  $(i - 1)$  incidente à la fois à  $c_3$  et à  $c_4$ . Par exemple, considérons la figure 2.2 qui représente un maillage composé de quadrilatères. Le sommet  $s_0$  est incident aux faces  $f_0, f_1, f_2$ . L'arête  $a_2$  est incidente à  $f_2$  et  $f_3$ , et ces deux faces sont alors adjacentes.

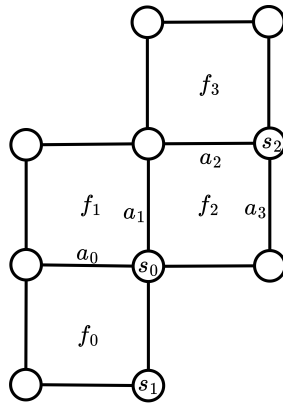


FIGURE 2.2 – Maillage composé uniquement de quadrilatères.

Dans ce document, nous considérons le cas particulier des maillages hexaédriques, et plus particulièrement les maillages hexaédriques structurés par blocs (voir figure 2.3. Mais au préalable introduisons les notions de maillages structurés et non structurés.

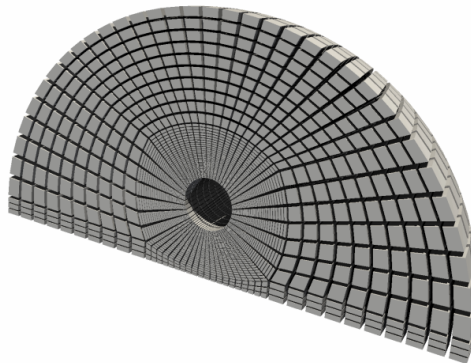


FIGURE 2.3 – Exemple de maillage hexaédrique structuré par blocs, où chaque mailles est un hexaèdre et la structure globale du maillage peut être vu comme un ensemble de grilles régulières.

Un maillage **structuré** correspond à une grille parfaite qui est déformée pour s'aligner avec le bord de la géométrie. Tous les sommets internes du maillage sont incidents à exactement 4 quadrilatères en 2D et 8 hexaèdres en 3D. De même, à l'exception des sommets de coin, c'est-à-dire les 4 coins de la grille en 2D ou les 8 en 3D, tous les sommets positionnés sur le bord de la géométrie sont incidents à exactement 2 quadrilatères en 2D et 4 hexaèdres en 3D. Cette régularité est utile pour faciliter l'écriture de schémas numériques que ce soit d'un point de vue informatique (voisinage constant et optimisation mémoire possible) que d'un point de vue mathématique (schémas de type différences finies simples à écrire, faible impact de la structuration

du maillage, celui-ci étant similaire partout dans le domaine géométrique). Le désavantage principal est qu'il n'est pas possible de discrétiser toute forme de domaine géométrique à l'aide d'un tel maillage et si tel est le cas, des mailles de mauvaise qualité géométrique peuvent apparaître près du bord.

Au contraire, un maillage **non structuré** se caractérise par une connectivité irrégulière. Elle ne peut pas être facilement exprimée sous la forme d'un tableau bidimensionnel ou tridimensionnel dans la mémoire de l'ordinateur. Cela permet à un solveur d'utiliser n'importe quel élément possible. Par rapport aux maillages structurés, pour lesquels les relations de voisinage sont implicites, ce modèle peut être très inefficace en termes d'occupation mémoire puisqu'il nécessite un stockage explicite des relations de voisinage.

Enfin, un maillage **structuré par blocs** est composé d'un ensemble de blocs quadrilatéraux en 2D et hexaédriques en 3D qui forment un macro- maillage non structuré. Chaque bloc de ce macro-maillage est alors subdivisé de manière structurée par une grille régulière. De cette façon, il est possible de discrétiser tout type de domaine géométrique par un ensemble de grilles régulières avec peu de sommets singuliers au sein du domaine.

## 2.2 . Opérations globales sur les maillages hexaédriques

Des opérations de modification topologiques sont possibles dans une structure tétraédrique de manière locale. Ainsi, il est possible d'ajouter un nouveau sommet n'importe où dans le maillage tout en maintenant une structure conforme, et ce de façon locale. Au contraire, ce n'est pas le cas pour les maillages hexaédriques. L'insertion d'un point dans ces maillages impacte directement les éléments voisins (voir figure 2.4). Ceci est dû à la structure topologique globale de tels maillages.

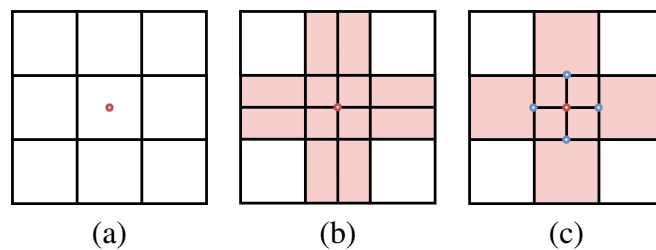


FIGURE 2.4 – (a) Insertion d'un point rouge au centre d'un quadrilatère. (b) Certains éléments voisins sont impactés en rouge. (c) Des noeuds non conformes apparaissent en bleu si la propagation de la découpe n'est pas effectuée de manière globale.

Les opérations de modification topologique de maillages hexaédriques permettent de modifier la topologie en maintenant la conformité de la structure, c'est à dire sans ajout de noeud non-conformes. Pour cela, une notion importante est la notion de **feuille**. Il s'agit d'un ensemble de mailles formant une couche de mailles comme

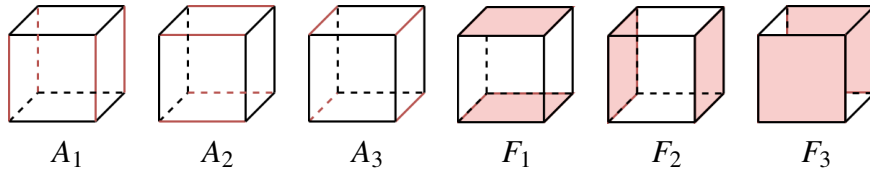


FIGURE 2.5 – Structuration des arêtes et faces d'un hexaèdre en trois quadruplés d'arêtes opposées, notés  $A_1$ ,  $A_2$ , et  $A_3$ , et trois couples de faces opposées notés  $F_1$ ,  $F_2$  et  $F_3$ .

illustré à la figure 2.6.(a) par exemple. Pour bien comprendre cette notion, considérons tout d'abord la structure individuelle d'un hexaèdre. Celui-ci est composé de 6 faces, 12 arêtes et 8 sommets. Pour chaque face, il existe une face opposée dans l'hexaèdre, et pour chaque arête, il en existe 3 autres opposées. Ceci partitionne les faces d'un hexaèdre en trois couples de faces et trois quadruplés d'arêtes (voir la figure 2.5).

Revenons maintenant à la notion de feuillet, et effectuons pour cela une sélection sur une structure hexaédrique telle que celle de la figure 2.6. En (a), une arête jaune du bord du maillage est sélectionnée. Cette arête est incidente à un hexaèdre qui dispose intrinsèquement de trois autres arêtes "opposées" à l'arête sélectionnée. Deux de celles-ci (car au bord) sont colorés en jaune en (b). De proche en proche, ces nouvelles arêtes sont également incidentes à d'autres hexaèdres, créant ainsi un ensemble d'arêtes "opposées" qui traverse le maillage, en (c), et forme ainsi aussi un ensemble d'hexaèdres traversés, en (d). Ces hexaèdres forment ce que l'on appelle un feuillet en dimension 3, et un hexaèdre peut appartenir à trois feuillets au maximum (un par quadruplé d'arêtes opposées). En sélectionnant ainsi un feuillet, une opération immédiate de modification topologique est la suppression du feuillet, où tous les hexaèdres du feuillet sont retirés du maillage (voir la figure.2.6.e).

Le contraire de supprimer un feuillet est d'en insérer un. Cette opération est souvent appelée **pillowing**, ou **padding**, et elle consiste à ajouter un *oreiller* d'hexaèdres autour d'un ensemble connexe d'hexaèdres pré-sélectionnés. Une telle opération est réalisée sur l'exemple de la figure 2.7, où l'ensemble d'hexaèdres à entourer est coloré en rouge en (a). On appelle cet ensemble le *shrinkset*. En (b), une couche de mailles vertes est insérée autour de cet ensemble.

Notons que les opérations de manipulation de feuillets (insertion et suppression) ont une propriété importante qui est de modifier le maillage en conservant sa conformité. Il est à noter aussi lors de ces opérations la modification de la topologie peut faire apparaître des sommets singuliers.

### 2.3 . Classification géométrique

Nous considérons dans ce document qu'une structure de blocs peut être définie par le quadruplé  $B_h = (N, A, F, H)$  tel que  $N$  est l'ensemble de ses noeuds,  $A$  est

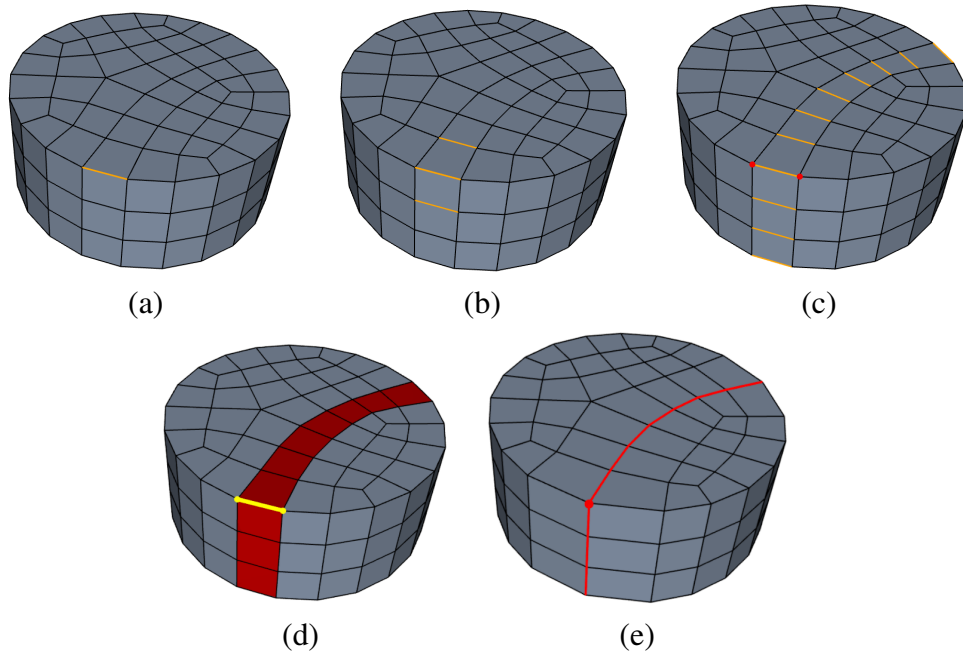


FIGURE 2.6 – Sélection d’une couche de mailles à partir de la sélection d’une arête en (a). En (b) et (c), l’ensemble des arêtes opposées à l’arête initialement sélectionnée sont représentés. En (d), les hexaèdres incidents à au moins quatre de ces arêtes sont représentés en rouge et forment le feuillet défini à partir de l’arête sélectionnée en (a). En (e), le feuillet complet d’hexaèdres est retiré du maillage, qui reste conforme.

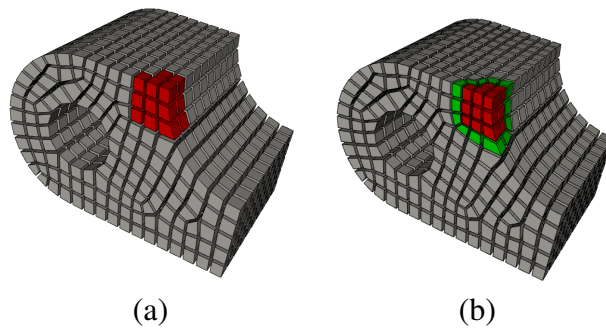


FIGURE 2.7 – Opération de *pillowing* en dimension 3, où il s’agit d’insérer une couche de mailles hexaédrique (b) entourant un ensemble d’hexaèdres pré-sélectionnés (a).

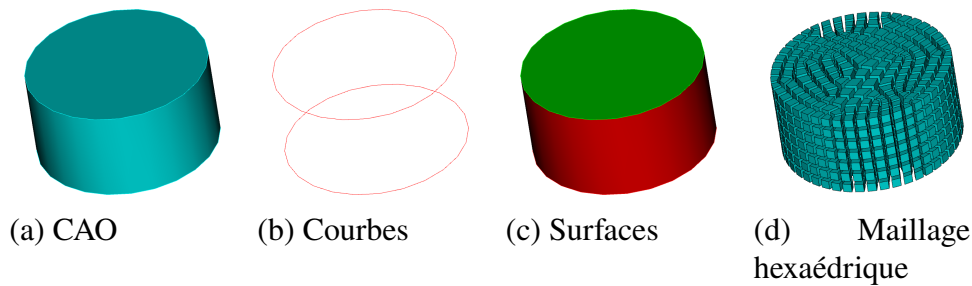


FIGURE 2.8 – De la CAO au maillage. En (a) un cylindre dont on représente son volume, en (b), les courbes de ce cylindre, en (c), ses surface, et en (d) un maillage associé, ici de nature hexaédrique.

l'ensemble de ses arêtes,  $F$  l'ensemble de ses faces, et  $H$  l'ensemble de ses blocs hexaédriques. On notera ces différentes entités, des  $i$ -cellules, avec  $i$  la dimension de la cellule en question. Ainsi une 3-cellule est un bloc, une 2-cellule est une face, une 1-cellule est une arête, et une 0-cellule est un nœud.

Outre la structure de blocs, nous devons représenter le modèle géométrique. En l'occurrence, un modèle géométrique de type CAO, pour *conception assistée par ordinateur* peut être représenté par son bord  $(V, C, S)$  (voir figure 2.8) où  $V$  est un ensemble de sommets,  $C$  un ensemble de courbes (voir figure 2.8.b), et  $S$  un ensemble de surfaces (voir figure 2.8.c). Une  $i$ -cellule sera alors *classifiée* sur une entité géométrique de dimension  $j$  avec  $0 \leq i \leq 3$  et  $i \leq j \leq 3$ . Une fonction de classification géométrique permet d'indiquer à quel élément géométrique correspond un cellule du maillage.

Dans le domaine de la simulation numérique, disposer d'une fonction de classification géométrique apporte de nombreux avantages. Elle permet d'abord de faciliter l'initialisation de conditions au bord pour des méthodes de type éléments ou volumes finis directement sur le modèle géométrique. Un autre intérêt est le lissage géométrique, c'est-à-dire un déplacement des coordonnées des sommets constituant le maillage en tenant compte de la nature et de relation avec les entités géométriques.

## 2.4 . Qualité d'un maillage

Un bon maillage dépend de la simulation et de critères physiques et numériques. Si on ne peut pas déterminer de manière générale un critère de qualité, il existe tout de même certaines caractéristiques géométriques usuelles pour les maillages hexaédriques comme le fait que le maillage soit structuré par blocs (voir figure 2.3), que la taille des cellules soit contrôlée à certains endroit d'intérêt, que les mailles soient alignées avec le bord du domaine et que chaque maille soit géométriquement proche d'un parallélépipède rectangle. On parle de distorsion des cellules (voir figure 2.9).

### 2.4.1 . Qualité géométrique d'un hexaèdre

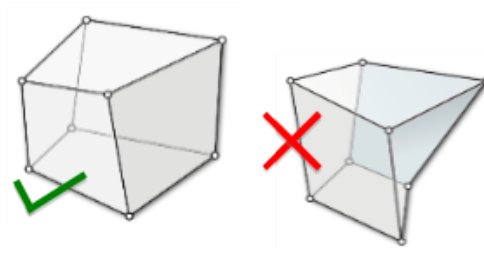


FIGURE 2.9 – Exemple de cellules hexaédriques dont la qualité géométrique est acceptable (à gauche) ou non (à droite). Dans le second cas, on peut noter que la cellule est non convexe.

Pour déterminer la qualité géométrique d'une cellule hexaédrique, on utilise fréquemment la mesure du *scaled Jacobian*. Pour introduire cette mesure, considérons que la géométrie de n'importe quel hexaèdre peut être transformée en l'élément de référence via une fonction de forme. L'élément de référence est le cube unité. Nous notons  $(x, y, z)$  (respectivement  $(x, y)$  en dimension 2) les coordonnées d'un point dans l'espace physique dans lequel vit notre hexaèdre et nous notons  $(\xi, \eta, \zeta)$  (respectivement  $(\xi, \eta)$  en dimension 2) les coordonnées dans l'espace de référence (ou espace logique).

Une fonction de forme est bijective en chaque point de l'élément de référence et de son bord<sup>1</sup> si la matrice jacobienne  $J$  de la transformation correspondant au changement de variable entre l'espace logique, dans lequel vit l'élément de référence, et l'espace physique n'est pas singulière. Cette condition peut être vérifiée en considérant que le déterminant de  $J$  n'est jamais nul en tout point du cube de référence. En d'autres termes, il est strictement positif. En dimension 2, la matrice jacobienne  $J_2$  est définie comme suit :

$$\begin{pmatrix} \frac{\partial T}{\partial \xi} \\ \frac{\partial T}{\partial \eta} \end{pmatrix} = \begin{pmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{pmatrix} \begin{pmatrix} \frac{\partial T}{\partial x} \\ \frac{\partial T}{\partial y} \end{pmatrix} = J \begin{pmatrix} \frac{\partial T}{\partial x} \\ \frac{\partial T}{\partial y} \end{pmatrix}. \quad (2.1)$$

De même, en dimension 3, on obtient la définition de  $J_3$  :

$$\begin{pmatrix} \frac{\partial T}{\partial \xi} \\ \frac{\partial T}{\partial \eta} \\ \frac{\partial T}{\partial \zeta} \end{pmatrix} = \begin{pmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{pmatrix} \begin{pmatrix} \frac{\partial T}{\partial x} \\ \frac{\partial T}{\partial y} \\ \frac{\partial T}{\partial z} \end{pmatrix} = J \begin{pmatrix} \frac{\partial T}{\partial x} \\ \frac{\partial T}{\partial y} \\ \frac{\partial T}{\partial z} \end{pmatrix}. \quad (2.2)$$

Quelle que soit la dimension (2 ou 3), la matrice jacobienne d'un simplexe  $T$ , peut être calculée en fonction des valeurs en ses sommets  $\{s_i(x_i, y_i, z_i)\}_{i \in [0, n]}$ , avec  $n = 2$  ou 3 :

$$|J_2| = \begin{vmatrix} x_1 - x_0 & y_1 - y_0 \\ x_2 - x_0 & y_2 - y_0 \end{vmatrix} \quad \text{et} \quad |J_3| = \begin{vmatrix} x_1 - x_0 & y_1 - y_0 & z_1 - z_0 \\ x_2 - x_0 & y_2 - y_0 & z_2 - z_0 \\ x_3 - x_0 & y_3 - y_0 & z_3 - z_0 \end{vmatrix} \quad (2.3)$$

1. À chaque point de l'élément de référence correspond un et un seul point de l'hexaèdre situé dans l'espace physique.



Dans le cas d'une cellule hexaédrique, une méthode usuelle pour calculer sa qualité géométrique est de considérer les huit tétraèdres issus de chacun de ses sommets et des trois arêtes incidentes. Cette approximation est usuelle même si elle ne garantit pas que l'on ait traité tous les points contenus dans l'hexaèdre. Le déterminant le plus petit parmi les huit matrices issues de ses sommets définit la métrique *Jacobienne* de l'hexaèdre. Une autre métrique utilisée est celle qui nous intéresse. C'est la version *scaled* avec des longueurs normalisées afin de disposer de valeurs comprises entre -1 et 1. Une valeur négative indique que l'élément est non convexe alors qu'une valeur de 1 indique que tous les angles internes sont des angles droits.

#### 2.4.2 . Lien entre valence, qualité géométrique et alignement au bord

Les sommets, arêtes et faces d'une structure de blocs sont classifiées sur des entités géométriques. Cette classification va jouer un rôle prépondérant pour détecter des problèmes d'alignement au bord, mais aussi de qualité géométrique des éléments sans affecter les qualités géométriques préalablement introduits. On note  $v(x)$  la valence de la cellule  $x$ ,  $x$  pouvant être une arête ou un sommet dans les exemples qui suivent. Nous définissons la **valence** d'une  $i$ -cellule  $x$  comme le nombre de  $i + 1$ -cellules incidentes à  $x$  avec  $i \in [1, 2]$ . (voir la figure 2.10 pour des exemples en dimension 2).

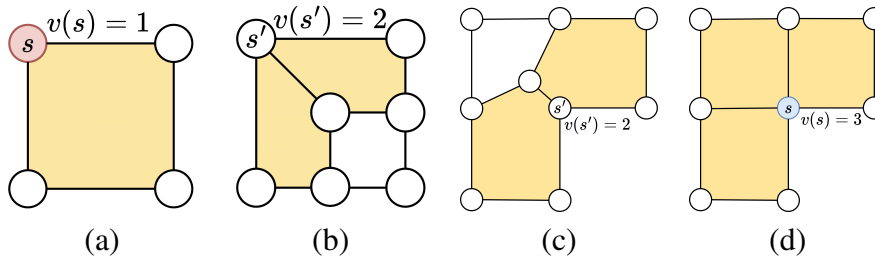


FIGURE 2.10 – Illustration de valences possibles pour un sommet en dimension 2. En (a), le sommet est de valence 1 ; en (b) et (c), il est de valence 2 ; en (d), il est de valence 3.

Considérons le cas d'une structure de blocs en dimension 2. Lorsqu'un sommet est interne au domaine, i.e. classifié sur une surface, l'angle idéal dans chaque face autour du sommet est de  $90^\circ$ . Il faut donc une valence de 4. Il en est de même en fait où que l'on soit au bord du domaine : en tout coin d'une face, l'angle idéal reste de  $90^\circ$ . Lorsqu'un sommet est classifié sur une courbe convexe comme sur la figure 2.11.a en noir, la valeur idéale de la valence est de 2 pour obtenir deux faces, et donc des coins d'angle  $90^\circ$  en ce point. Pour le cas (a), on n'a qu'une valence de 1 et donc il est nécessaire d'augmenter son nombre de voisins. Dans le cas d'une classification sur une courbe concave en revanche, on veut toujours deux faces incidentes, il faut donc réduire la valence de trois à deux.

Le raisonnement est identique en 3D, mais on considère les arêtes et non plus les

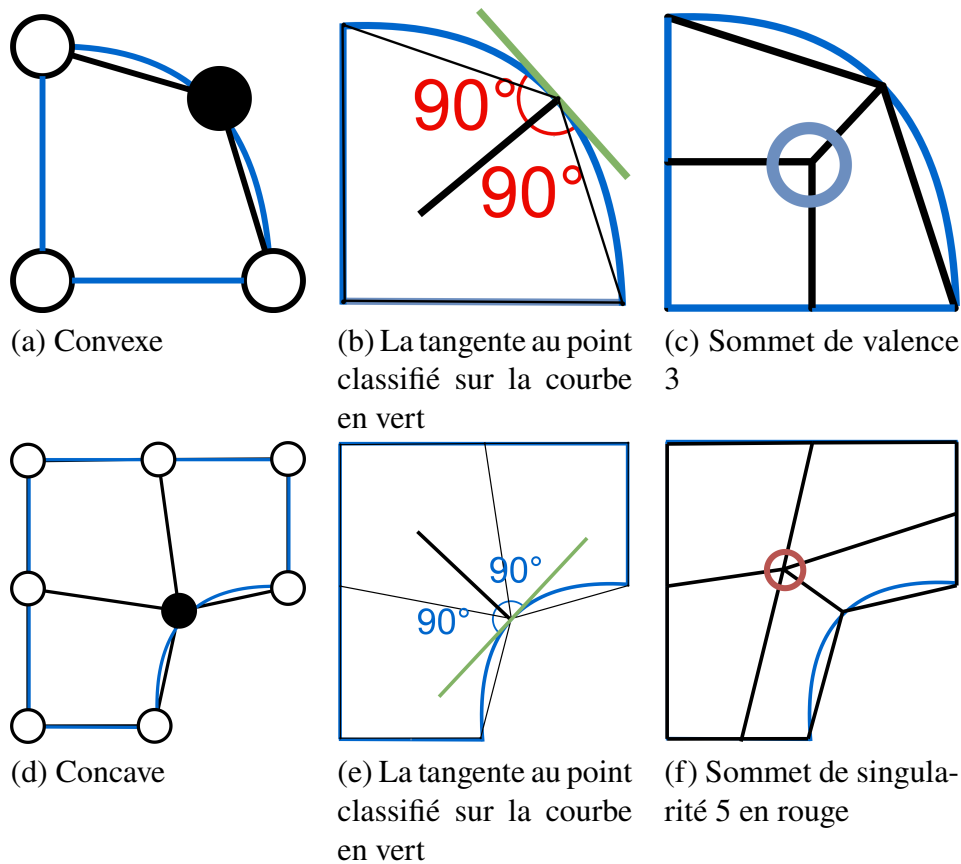


FIGURE 2.11 – Différentes configurations de sommets présents sur le bord du domaine.

sommets. L'angle diédral d'une arête au sein d'une cellule hexaédrique incidente doit être là aussi de  $90^\circ$  et on peut appliquer le même raisonnement à l'intérieur du domaine et le long de surfaces convexes et concaves.

## 2.5 . Représentation d'un maillage

La représentation d'une structure en blocs est à mi-chemin entre la représentation d'un maillage, qui peut contenir des millions de cellules, et la représentation de la topologie d'un modèle géométrique composé de quelques milliers de volumes au maximum. Une approche directe de la représentation d'un maillage à  $n$  dimensions consiste à le définir comme un  $n$ -tuple d'ensembles de cellules, chaque ensemble contenant les cellules d'une dimension donnée, et les relations d'incidence/adjacence utiles pour l'application à développer. Ceci est fourni par des bibliothèques génériques de maillage [8, 9, 10, 11, 12, 13] qui **représentent explicitement** les cellules du maillage. Si l'on considère les représentations traditionnelles utilisées en géométrie informatique, les représentations utilisées dans ces bibliothèques dérivent du modèle

classique des *graphes d'incidence* [14] figure 2.13.a.

D'autres bibliothèques sont basées sur des entités abstraites (voir la figure 2.12) qui représentent implicitement les cellules du maillage, telles que les arêtes ailées [15], la liste des arêtes doublement chaînées [16], la structure de données des demi-arêtes [17], les maillages de surface [18] ou les cartes combinatoires [19, 20] (figure 2.13.b). Ces dernières peuvent être considérées comme une généralisation des maillages à demi-bords qui peuvent prendre en charge des complexes cellulaires de plus grande dimension. D'autres représentations vont plus loin dans la division des cellules et fournissent des entités centrales abstraites relatives aux sommets. En particulier, nous considérons l'utilisation de cartes généralisées (figure 2.13.c) à  $n$  dimensions. Finalement, les cartes généralisées à  $n$  dimensions [19] représentent des subdivisions non orientables et ouvertes. Ce qui n'est pas possible avec les cartes combinatoires.

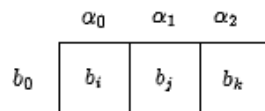


FIGURE 2.12 – Exemple d'une entité abstraite sur laquelle reposent en partie les cartes généralisées et combinatoires. Il s'agit d'un brin pointant vers d'autres brins  $b_i, b_j, b_k$  permettant de se déplacer dans la topologie avec les liens  $\alpha_0, \alpha_1, \alpha_2$ .

D'après l'ouvrage [20], les cartes combinatoires sont une bonne base pour représenter la topologie lorsque les cellules doivent être manipulées que cela soit pour la nature des opérations ou pour les besoins de représentation naturelle de l'objet. Par contre, la définition des cartes généralisées est homogène peu importe la dimension ce qui en pratique simplifie l'écriture des algorithmes. Quant aux cartes combinatoires, elles ont l'avantage de nécessiter deux fois moins de mémoire que les cartes généralisées. Dans le cas des structures de blocs qui ont "peu" de cellules [21], nous privilégierons les cartes généralisées.

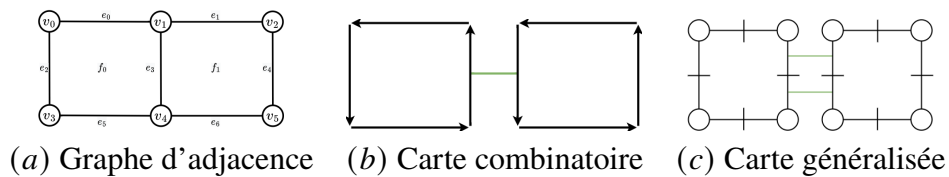


FIGURE 2.13 – Exemples de représentation de la structure topologique de deux mailles quadrangulaires partageant deux sommets et une arête.

## 2.6 . Méthodes de génération de maillage

La méthode de génération de maillage la plus simple consiste à générer un maillage tétraédrique puis à découper chaque tétraèdre en quatre hexaèdres, un par coin du té-

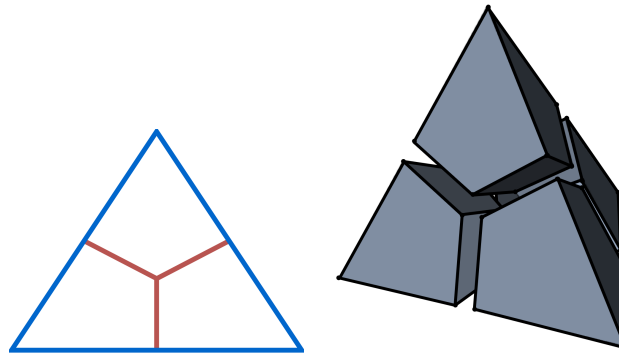


FIGURE 2.14 – Illustration de la génération du découpage d'un triangle ou d'un tétraèdre pour générer un maillage hexaédrique.

traèdre (voir la figure 2.14). Le problème est que cette méthode génère de nombreuses singularités dans le maillage, ce qui n'en fait pas une méthode utilisée en pratique.

Les méthodes les plus étudiées aujourd'hui pour générer un maillage hexaédrique structuré par blocs s'appuient sur la génération de polycubes [22, 23] ou l'utilisation conjointe de champs d'orientations et de méthodes de paramétrisation globale [24, 25, 26]. Les premières sont plus robustes mais ne fournissent que très rarement la structure escomptée, tandis que les secondes produisent le résultat attendu mais uniquement pour des domaines simples. Dans les deux cas, les approches suivies reposent sur la résolution approchée de problèmes d'optimisation linéaire ou quadratique.

### 2.6.1 . Overlay-grid

Les méthodes par overlay-grid ou mesh-first considèrent un maillage initial dont les mailles situées au cœur du modèle de CAO restent inchangées et celles au bord vont être déformées pour capturer le modèle géométrique. Les premiers algorithmes d'overlay-grid [27] ont été améliorés afin de permettre une meilleure capture de la géométrie en adaptant le maillage utilisé [28, 29, 30]. Dans le cas où le modèle de CAO n'est pas connu explicitement, des méthodes peuvent utiliser des maillages portant des informations comme par exemple des fractions de présence [31, 32, 33].

Par construction, cette famille de méthodes est intrinsèquement robuste, dans le sens où l'on dispose en permanence d'un maillage hexaédrique. Par contre, elle présente les inconvénients de mal capturer le modèle et de fournir des maillages dont les mailles de plus mauvaise qualité sont localisées sur les bords du domaine. Si l'on regarde les valences des arêtes de tel maillages, elles sont "parfaites" au sein du domaine (valence 4) et le plus souvent mauvaises au bord.

### 2.6.2 . Utilisation des champs d'orientation

La méthode des champs de d'orientation, ou *frame fields*, [24] est illustrée sur la figure 2.15. Le principe de la méthode est de partir d'un maillage tétraédrique puis de calculer un champ de croix lisse et orthogonal au bord. Ce champ de croix fournit

en tout point du domaine trois directions orthogonales qui indique l'orientation que devrait avoir un hexaèdre en ce point. En calculant une paramétrisation globale qui soit la plus alignée avec le champ d'orientation, on en déduit une structure de grilles proche du maillage hexaédrique par blocs que l'on souhaite avoir.

Bien que cette méthode produise des structures de blocs avec un alignement au bord et des hexaèdres de bonne qualité, elle ne fournit des structures de blocs valides que dans le cas de modèles géométriques simples.

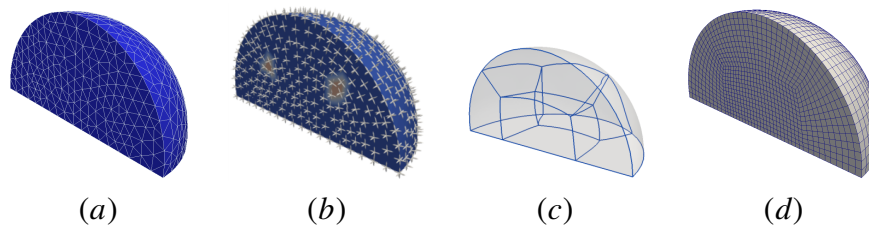


FIGURE 2.15 – Obtention d'un maillage par la méthode des champs d'orientation. Un champ d'orientation est calculé sur les sommets d'un maillage tétraédrique qui discrétise la géométrie (a), le graphe de singularité est extrait de ce champ (b), ce qui permet de construire un « squelette » d'une structure de blocs (c), enfin le maillage hexaédrique est généré en subdivisant la structure de blocs(d).

### 2.6.3 . Polycube

La méthode des Polycube [22] se décompose en plusieurs étapes complexes représentées sur la figure 2.16. Elle se base elle-aussi sur une triangulation du domaine d'entrée ou tout du moins de la surface le bordant. La première étape de cette méthode est l'étape de coloration. Les six directions  $\{+X, -X, +Y, -Y, +Z, -Z\}$  représentent les normales sortantes d'un cube unité et une couleur leur est attribuée. Une approche naïve consiste à colorier chaque triangle avec la couleur de la normale la plus proche de sa normale sortante. Cependant, cette méthode n'offre aucune garantie de qualité.

La deuxième étape est l'étape de déformation où le but est de s'aligner avec les contraintes issues de la coloration le long des 6 directions  $\{+X, -X, +Y, -Y, +Z, -Z\}$ .

La troisième étape est la quantification. L'objectif est ici d'extraire une grille. Une approche naïve consiste à prendre des valeurs du bord, de les arrondir et enfin de recalculer une nouvelle carte. Enfin la quatrième et dernière étape consiste à appliquer la déformation inverse qui permet de retrouver la géométrie initiale discrétisée.

Cette méthode fournit une structure de blocs avec une bonne qualité sauf au bord où l'alignement n'est pas réellement contrôlé. Elle a l'avantage de permettre de fournir des structures de blocs pour des géométries plus nombreuses et plus complexes que celles traitées aujourd'hui par les méthodes basées sur les champs d'orientations.

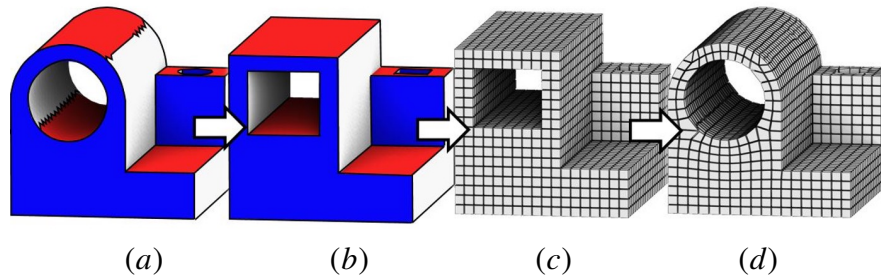


FIGURE 2.16 – Illustration de la méthode de maillage par polycube. (a) L'étape de coloration a été effectuée puis le maillage tétraédrique est déformée (b). Le polycube obtenu est ensuite discrétisé par une grille régulière d'hexaèdres (c), la déformation inverse est appliquée à la grille pour générer le maillage final de la géométrie de base (d) ou une extraction des hexaèdres peut être effectuée directement.

#### 2.6.4 . Selective padding

La méthode du *selective padding* ne génère pas à proprement parler une structure de blocs. Elle prend en entrée un maillage obtenu à l'aide de la méthode polycube et insère des couches de mailles au bord pour améliorer la qualité géométrique des cellules dans ces zones.

Une méthode utilisée jusqu'ici pour améliorer la qualité était d'ajouter une couche de mailles au bord du domaine améliorant ainsi les pires éléments mais réduisant aussi la qualité moyenne de la structure de bloc. De plus la structure ainsi obtenue se complexifie de part les singularités insérées mais aussi par le nombre d'éléments. Dans [34], les auteurs proposent une méthode d'optimisation qui génère une sélection d'un feuillet puis insère une couche de maille en ciblant les mauvais éléments du bord. Leur formulation globale est basée sur la résolution d'un problème binaire ce qui leur permet de contrôler la balance  $\lambda$  entre le nombre d'éléments insérés et le nombre de singularités.

La première étape de cette méthode consiste à déterminer les éléments de mauvaise qualité à l'aide de plusieurs métriques. Puis ils attribuent une valeur de distorsion à chaque face selon des critères géométriques et propagent cette distorsion aux faces voisines pour éviter des sélections initiales non valides.

La limite de cette méthode est qu'ils utilisent un paramètre  $\lambda$ , qui permet de gérer la balance entre le nombre d'éléments et le nombre de singularités insérées. Or ce paramètre est inconnu et fixé par l'utilisateur. Pour une instance, un grand nombre de valeurs de  $\lambda$  devraient être testées afin de trouver celle qui améliore le plus la structure de blocs initiale.

#### 2.6.5 . Méthodes interactives

Comme nous venons de le voir, il n'existe pas de méthode automatique fonctionnant dans toutes les situations. Des travaux ayant pour but de faciliter l'utilisation

des outils interactifs de CAO-maillage ont été menés, que ce soit via des interfaces homme-machine permettant à l'ingénieur de dessiner ce qui sera traduit en commandes pour ces outils [35], ou bien la capacité à suggérer des opérations, l'ingénieur choisissant lesquelles il souhaite effectuer [36, 37].

Concernant les champs d'orientations, des outils interactifs ont été adaptés pour aider à la décomposition en blocs en 3D. Dans [38] la méthode requiert de l'utilisateur de créer les cordes au bord depuis lesquelles les feuillet duaux vont se propager, et dans [39] un feuillet dual peut se propager à partir d'un point sélectionné par l'ingénieur (voir figure 2.17) pour ensuite en itérant former la structure de blocs souhaitée.

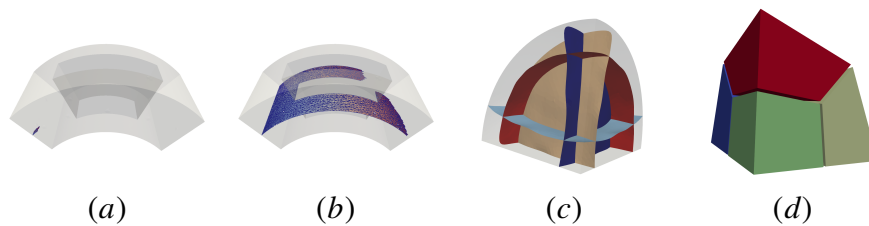


FIGURE 2.17 – Après sélection d'un point de départ (a) un feuillet dual se propage (b). Une fois les feuillet duaux construits (c) une structure de blocs est extraite (d).

## 3 - Représentation et opérations sur maillage structuré par blocs

Dans ce chapitre, nous nous intéressons à la façon de représenter une structure de blocs, et à son lien avec la représentation par le bord d'un modèle géométrique de type CAO. C'est la structure principale d'un logiciel de maillage hexaédrique tel que le logiciel **MGX**, développé au CEA DAM, et une attention toute particulière doit donc être apportée à son choix, que ce soit en termes de robustesse, de fiabilité ou d'expressivité. Par ce dernier point, on entend que la représentation choisie doit permettre de facilement écrire des fonctionnalités de gestion et de modification de structures complexes de blocs. On pensera en particulier à des opérations telles que la suppression de blocs, l'insertion de blocs, l'insertion de grilles, la propagation de découpage des blocs, la propagation des paramètres de maillage le long des couches de blocs, etc. Ces opérations sont sujettes aux erreurs et la modification ou l'ajout d'une opération est un travail très fastidieux.

Afin de fournir des structures de données et des opérations fiables et robustes pour manipuler et modifier les structures de blocs hexaédriques dans les logiciels interactifs de maillage, nous utilisons le modèle des cartes généralisées de dimension  $n$  [19]. Ce modèle fournit des invariants topologiques et un traitement systématique des données géométriques qui nous permettent de garantir au logiciel la robustesse attendue. Dans ce contexte, les principales contributions de ce travail sont les suivantes :

1. Nous définissons formellement comment représenter la topologie de la structure en blocs et le lien automatique avec un modèle géométrique de type BRep (voir Section 3.1) ;
2. Nous définissons formellement l'ensemble minimal d'opérations pour éditer les structures en blocs hexaédriques (voir Section 3.2). Ces opérations sont la sélection de feuillets, la suppression de feuillets et l'insertion de feuillets ;
3. Nous fournissons des algorithmes en pseudo-code pour chaque opération dans la section 3.2 ;

### **Publications et communications relatives à ce chapitre :**

- Conférence internationale avec comité de lecture et publication des actes : V. Postat, N. Le Goff, S. Calderan, F. Ledoux, et G. Hutzler, Formal definition of hexahedral blocking operations using  $n$ -G-maps, *SIAM International Meshing Roundtable Workshop 2023* (SIAM IMR23), 6-9 mars, 2023.

### **3.1 . Représentation d'une structure de blocs à l'aide des cartes géné-**



## ralisées

Au contraire des maillages, les structures de blocs contiennent un nombre limité de cellules, i.e. les blocs, qui dépasse rarement quelques milliers d'éléments, de sorte que la mémoire et les performances d'accès aux données ne constituent pas un facteur limitant dans la pratique. Le choix de la bonne représentation des données pour les structures de blocs soulève donc d'autres contraintes que celles considérées usuellement en maillage pour la simulation numérique. En l'occurrence, on considère qu'il est nécessaire de prendre en compte les points suivants :

1. **Classification géométrique.** Chaque cellule d'une structure de blocs  $B$  est associée/classifiée sur une entité du modèle géométrique  $G$ . Plus précisément, chaque cellule de dimension  $i$  de  $B$ , avec  $0 \leq i \leq 3$ , est classifiée sur une entité géométrique de  $G$  de dimension  $j$ , avec  $i \leq j \leq 3$ . En outre, lors des opérations d'édition de la structure  $B$ , la classification doit être mise à jour selon certaines règles, et la classification peut empêcher une opération de modification topologique d'être effectuée dans le cas où elle ferait perdre de l'information géométrique. La classification géométrique est essentielle pour le processus de maillage.
2. **Paramètres des algorithmes de maillage.** Les cellules de  $B$  portent les paramètres de maillage de chaque bloc. Parmi ceux-ci, certains sont locaux à un côté d'une face. En d'autres termes, pour une face partagée par deux blocs, un paramètre peut être donné à chaque côté de la face ; il peut s'agir de paramètres de lois de discrétisation géométrique, de conditions d'orthogonalité. Les représentations implicites, comme les cartes généralisées de dimension  $n$  sont alors une meilleure option que les représentations explicites qui ne représentent généralement pas de telles demi-faces.
3. **Représentation géométrique des blocs.** Il est simple et traditionnel de représenter les blocs comme des complexes de cellules linéaires, mais cette représentation rencontre des limitations. En particulier, le logiciel **MGX** s'oriente aujourd'hui vers une représentation non linéaire des blocs, où chaque bloc pourra être représenté par une cellule d'ordre élevée.

Dans la suite de cette section, nous introduisons des définitions usuelles relatives au modèle des cartes généralisées. Le lecteur intéressé pourra se référer à [20] pour plus d'explications et de détails.

### 3.1.1 . Cartes généralisées et topologie

Les cartes généralisées de dimension  $n$ , ou  $n$ -G-cartes en abrégé, reposent sur la notion de *brins* comme éléments de base. Les brins sont des entités purement abstraites qui n'intègrent aucune information géométrique. Intuitivement, les  $n$ -G-cartes proviennent de la décomposition d'objets de dimension  $n$  en cellules topologiques. Considérons l'objet 2D de la figure. 3.1.a, qui est d'abord décomposé en faces à la figure. 3.1.b, faces reliées entre elles le long de leurs bords communs par un lien nu-

méroté 2. Le lien est noté 2 car il relie deux faces, qui sont des cellules de dimension 2. De même, chaque face est décomposée en arêtes connectées par le lien 1 (voir la figure 3.1.c), elles-mêmes chacune décomposée selon leurs sommets extrémités à la figure 3.1.d. Il en résulte à la fin de la décomposition que les sommets ainsi obtenus n'ont pas la même signification que ceux des représentations explicites. Ce sont les brins et en 2D, chacun d'eux correspond à un "sommets localement à une arête, elle-même localement à une face". En d'autres termes, un brin 2D est un triplet (sommets, arête, face)<sup>1</sup>.

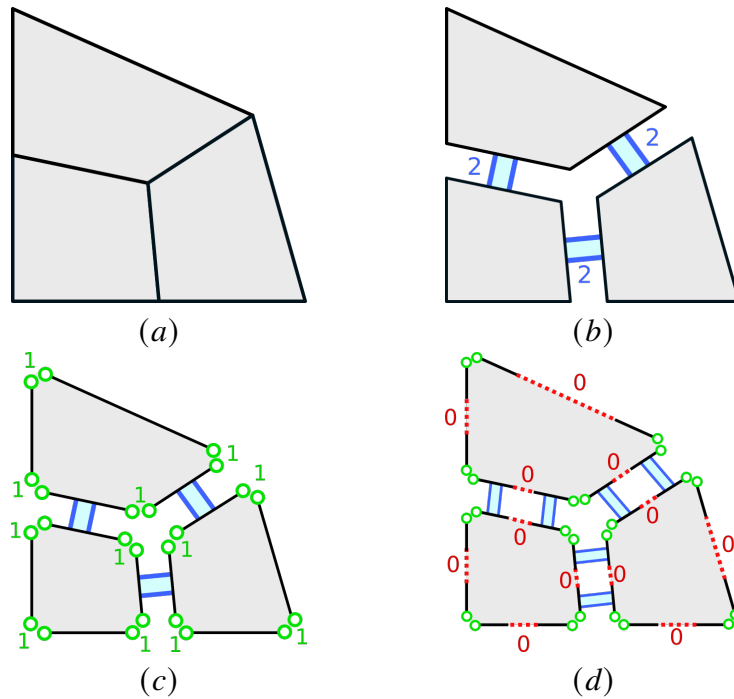


FIGURE 3.1 – Décomposition d'un complexe cellulaire de dimension 2 en un ensemble de brins : En (a), le complexe initial; En (b), il est décomposé en faces; En (c), chaque face est décomposée en arêtes; Enfin, chaque arête localement à une face est décomposée en ses deux extrémités, qui sont des sommets (d).

Les différents liens  $i$  sont des arcs étiquetés, désignés  $\alpha_i$ , où  $i$  représente une dimension qui appartient à  $[[0; n]]$  pour une  $n$ -G-carte. Chaque lien  $\alpha_i$  est une fonction agissant sur les brins et participant à retrouver les notions habituelles de cellules. Définissons maintenant formellement les cartes généralisées de dimension  $n$ , qui permettent de représenter des variétés de dimension  $n$ , orientables ou non, avec ou sans bord.

**Definition 3.1.1 (Carte généralisée)** Soit  $n \geq -1$ , une carte généralisée de dimension  $n$ , ou  $n$ -G-carte, est une algèbre  $(D, \alpha_0, \dots, \alpha_n)$  telle que  $D$  est un ensemble fini

1. Un brin est équivalent à un tuple de cellules tel que défini dans [40].

de brins et :

$$\forall i, 0 \leq i \leq n, \alpha_i^2 = id, \quad (3.1)$$

$$\forall (i, j), 0 \leq i < i+2 \leq j \leq n, (\alpha_i \circ \alpha_j)^2 = id. \quad (3.2)$$

La définition précédente mérite quelques explications. Tout d'abord, elle commence par considérer la dimension  $-1$ , ceci afin de représenter la carte  $G$  vide qui ne contient que des brins non connectés. Ensuite, la propriété 3.1 indique que les fonctions de lien  $\alpha_i$  sont des involutions<sup>2</sup>, tandis que la propriété 3.2 ajoute une contrainte supplémentaire qui garantit de ne représenter que les complexes cellulaires de type variété. Par exemple, en dimension 2, cette propriété impose que  $\alpha_0 \circ \alpha_2$  est une involution. Cela signifie que si deux brins  $d$  et  $d'$  sont reliés par  $\alpha_2$ , alors les brins  $\alpha_0(d)$  et  $\alpha_0(d')$  sont également reliés par  $\alpha_2$ . De cette manière, deux faces adjacentes partagent une arête complète et pas seulement une partie (voir la figure 3.2).

Introduisons maintenant quelques notions utiles pour les sections à venir. Un brin  $d \in D$  est  $i$ -**libre** si  $\alpha_i(d) = d$ . Dans notre utilisation des cartes généralisées, tous les brins de la frontière d'une  $n$ -G-carte sont  $n$ -libres. Dans les différentes figures du document, nous avons fait les choix suivants de représentation graphique :

- Les liens  $\alpha_0$  sont représentés par des lignes de points entre deux brins (⋯);
- Les liens  $\alpha_1$  sont représentés par deux cercles (⊖);
- Enfin les liens  $\alpha_2$  sont représentés par deux lignes de segments (◊).

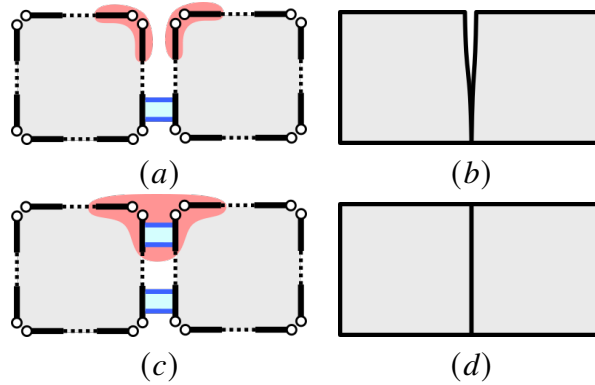


FIGURE 3.2 – Sur la première ligne en (a) et (b), nous n'imposons pas que  $\alpha_0 \circ \alpha_2$  soit une involution ; les faces quadrangulaires sont partiellement connectées, elles ne partagent qu'un seul sommet. Sur la deuxième ligne en (c) et (d), la composition  $\alpha_0 \circ \alpha_2$  est une involution et les faces quadrangulaires sont entièrement collées le long du bord.

Si les brins sont les éléments atomiques d'une  $n$ -G-carte, une structure de blocs est constituée de cellules que l'on souhaite récupérer pour écrire de nombreux algorithmes

2. Une fonction  $f$  est une involution si et seulement si  $f^2 = id$ .

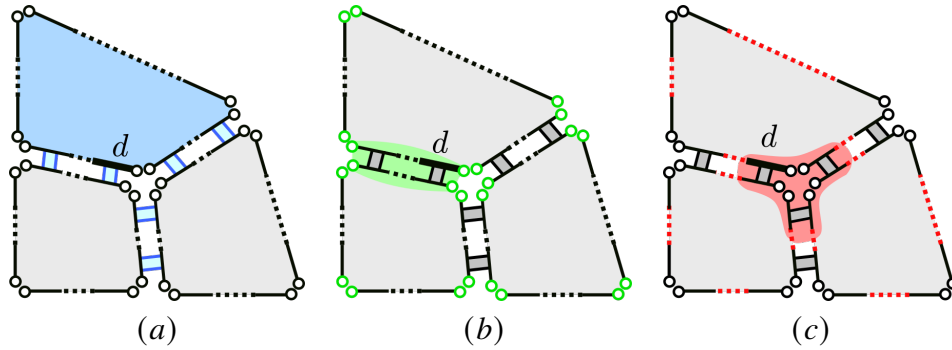


FIGURE 3.3 – Définition des  $i$ -cellules. En (a), la 2-cellule qui contient le brin  $d$  est l'orbite  $\langle \alpha_0, \alpha_1 \rangle (d)$ ; En (b), la 1-cellule qui contient le brin  $d$  est l'orbite  $\langle \alpha_0, \alpha_2 \rangle (d)$ ; En (c), la 0-cellule qui contient le brin  $d$  est l'orbite  $\langle \alpha_1, \alpha_2 \rangle (d)$ .

et applications. Dans notre cas, il est obligatoire d'accéder aux sommets, ou 0-cellules, aux arêtes, ou 1-cellules, aux faces, ou 2-cellules, et aux régions, ou 3-cellules. Dans la représentation  $n$ -G-carte, les  $i$ -cellules sont un cas particulier d'orbites.

**Definition 3.1.2 (orbite)** Soit  $\Phi = \{f_1, \dots, f_n\}$  un ensemble de permutations <sup>3</sup> sur un ensemble  $E$ . L'orbite de  $e \in E$  relativement à  $\Phi$  est le sous ensemble  $\langle \Phi \rangle (e)$  de  $E$  tel que

$$\langle \Phi \rangle (e) = \{\phi(e) \mid \phi \in \langle \Phi \rangle\}.$$

L'orbite d'un élément  $e$  dans  $E$  est constituée des éléments de  $E$  qui peuvent être atteints par toute composition de permutations de  $\phi$  et de leurs inverses. Dans le modèle des  $n$ -G-cartes, les fonctions  $\alpha_i$  sont des involutions, c'est-à-dire un cas particulier de permutations. Et les sommets, les arêtes, les faces et les régions sont des cas particuliers d'orbites. Par exemple, si l'on considère la 2-G-carte représentée sur la figure 3.3(a) et le brin  $d$ , l'orbite  $\langle \alpha_0, \alpha_1 \rangle (d)$  rassemble tous les brins qui appartiennent à la face bleue. Sur la figure 3.3(b), l'orbite  $\langle \alpha_0, \alpha_2 \rangle (d)$  correspond aux brins d'une arête, tandis qu'en 3.3(c), la 0-cellule qui contient le brin  $d$  est l'orbite  $\langle \alpha_1, \alpha_2 \rangle (d)$ . Formellement, dans une  $n$ -G-carte, les  $i$ -cellules sont définies comme suit.

**Definition 3.1.3 (i-cellule)** Soit  $G = (D, \alpha_0, \dots, \alpha_n)$  une  $n$ -G-carte,  $d$  un brin de  $D$  et  $i \in \{0, \dots, n\}$ , la  $i$ -cellule qui contient  $d$  est l'orbite

$$\langle \widehat{\alpha}_i \rangle (d) = \langle \alpha_0, \dots, \alpha_{i-1}, \alpha_{i+1}, \dots, \alpha_n \rangle (d).$$

On constate que l'application d'une involution  $\alpha_i$  sur un brin permet de passer d'une  $i$ -cellule à une autre  $i$ -cellule. Au contraire, l'application d'une involution  $\alpha_j$ ,  $j \neq i$  permet de rester dans la cellule  $i$ .

3.  $f$  est une permutation sur  $E$  si  $\forall e \in E, \exists k > 0 / f^k(e) = e$ .

**Notation :** Par souci de lisibilité, on note  $\alpha_{ij\dots k}$  la composition de fonctions. Par extension, nous notons

$$d\alpha_{ij\dots k} \cong (\alpha_k \circ \dots \circ \alpha_j \circ \alpha_i)(d).$$

Cette notation nous permet de lire de gauche à droite l'application successive des involutions sur un brin  $d$ . Par exemple le brin  $d\alpha_{12102}$  sera le brin obtenu en appliquant successivement  $\alpha_1, \alpha_2, \alpha_1, \alpha_0$  et  $\alpha_2$  à partir de  $d$ .

### 3.1.2 . Orbites et classification géométrique

Les sommets, les arêtes et les faces tels que définis précédemment ne sont que des entités topologiques. Pour écrire des algorithmes, mais aussi pour associer une représentation géométrique, ils doivent "porter" des données. En particulier, dans le contexte de la génération de blocs pour les modèles géométriques de type CAO, nous devons :

1. Intégrer/représenter le modèle topologique des  $n$ -G-cartes dans un espace géométrique et,
2. Attribuer des cellules de bloc aux entités géométriques. Une telle affectation de données peut être gérée dans le modèle des cartes généralisées en définissant une fonction de "mapping" pour chaque type d'orbites.

Pour notre propos, nous limitons ces associations aux cellules et ne généralisons pas sur les orbites. Nous nous limitons aussi au cas 3D. Considérons une 3-G-carte  $G = (D, \alpha_0, \dots, \alpha_3)$  et un modèle géométrique représenté par son bord  $M = (V, S, C, P)$ , qui est un ensemble de volumes  $V$ , délimités par des surfaces de  $S$ , des courbes de  $C$  et des points de  $P$ . Cette définition d'ensemble est très faible et permet de considérer des modèles non étanches. Pour chaque dimension  $i \in \llbracket 0; n \rrbracket$ , nous définissons une fonction de *classification géométrique* comme suit.

**Definition 3.1.4 (classification géométrique)** *Soient une 3-G-carte  $G = (D, \alpha_0, \dots, \alpha_3)$ , un modèle géométrique  $M = (V, S, C, P)$ , et  $i \in \llbracket 0; n \rrbracket$ , alors  $gc_i : D \rightarrow S_M$ , avec  $S_M = V \cup S \cup C \cup P \cup \emptyset$ , est une fonction de classification si et seulement si :*

$$\forall d \in D, \forall d' \in \langle \widehat{\alpha}_i \rangle (d), gc_i(d') = gc_i(d), \quad (3.3)$$

$$\forall d \in D, i \leq \dim(gc_i(d')). \quad (3.4)$$

Dans cette définition, la propriété 3.3 garantit que la fonction  $gc_i$  affecte tous les brins d'une  $i$ -cellule à la même entité géométrique. Et avec la propriété 3.4, on indique que la dimension de cette entité géométrique est supérieure à  $i$ . Par exemple, une face peut être classifiée sur une courbe, une surface ou un volume de  $M$  mais pas sur un point.

Dans le cadre de cette thèse, nous représentons linéairement les blocs, c'est-à-dire que nous attribuons un point de  $\mathbf{R}^n$  à chaque 0-cellule, et nous en déduisons linéairement la géométrie des arêtes, des faces et des volumes. Plus formellement,

pour une  $n$ -G-carte  $G = (D, \alpha_0, \dots, \alpha_n)$  nous définissons une fonction de plongement  $e : D \rightarrow \mathbf{R}^n$  telle que :

$$\forall d \in D, \forall d' \in \langle \widehat{\alpha_0} \rangle (d), e(d') = e(d). \quad (3.5)$$

Cette propriété est similaire à l'équation 3.3 pour les fonctions de classification. Nous nous assurons de la cohérence du plongement : tous les brins d'une 0-cellule sont affectés sur le même point géométrique. A noter que la classification géométrique permet uniquement de mettre à jour le lien vers le modèle géométrique lors des opérations sur les blocs. Il ne gère pas intrinsèquement les inexactitudes du modèle géométrique (espaces, intersections de surfaces, courbes de taille nulle, etc.).

### 3.1.3 . Opérations de modifications atomiques

Nous considérons quatre opérations atomiques qui peuvent être appliquées sur une  $n$ -G-carte  $G = (D, \alpha_0, \dots, \alpha_n)$ . Notons que chacune de ces opérations a un impact potentiel sur la classification géométrique.

#### Création de brins

La première opération consiste à ajouter un nouveau brin  $d$  à l'ensemble  $D$ . Lors de l'ajout, le brin est libre à toutes les dimensions ( $\forall i \in [[0; n]], \alpha_i = d$ ), la classification est inconnue ( $g_i(d) = \emptyset$ ) et le plongement est à l'origine ( $e(d) = (0, 0)$  en 2D et  $(0, 0, 0)$  en 3D). En d'autres termes,  $d$  est totalement indépendant des autres brins.

#### Suppression de brins

Supprimer un brin  $d$  de  $D$  aura un impact sur les involutions  $\{\alpha_i\}_{i=0..n}$ . Tout d'abord, pour tout  $i \in [[0; n]]$ , le brin qui est  $i$ -lié à  $d$  devient  $i$ -libre. Deuxièmement, comme  $G$  reste une  $n$ -G-carte après la suppression de  $d$ , la deuxième propriété de la définition 3.1.1 induit que d'autres brins seront  $i$ -libres après la suppression de  $d$ . Par exemple, supprimer un brin  $d$  d'une 2-G-carte implique que  $d\alpha_2$  deviendra 2-libre, mais aussi les brins  $d\alpha_0$  et  $d\alpha_02$ .

#### Découpage d'un brin

Nous effectuons une  $i$ -découpe d'un brin  $d$  lorsque nous posons  $d\alpha_i = d$ . On dira que aussi que  $d$  est découpé pour l'involution  $\alpha_i$ . De la même manière que pour la suppression de brin, comme  $G$  reste une  $n$ -G-carte après la  $i$ -découpe de  $d$ , la deuxième condition de la définition 3.1.1 implique que d'autres brins sont  $i$ -libres après avoir découpé  $d$  pour  $\alpha_i$ .

## Couture de brins

Si nous considérons deux brins  $(d, d') \in D \times D$   $i$ -libres, alors  $i$ -coudre  $d$  et  $d'$  signifie avoir  $d\alpha_i = d'$ . La première condition de la définition 3.1.1 implique que  $d'\alpha_i = d$  et la deuxième condition que d'autres liens sont créés.

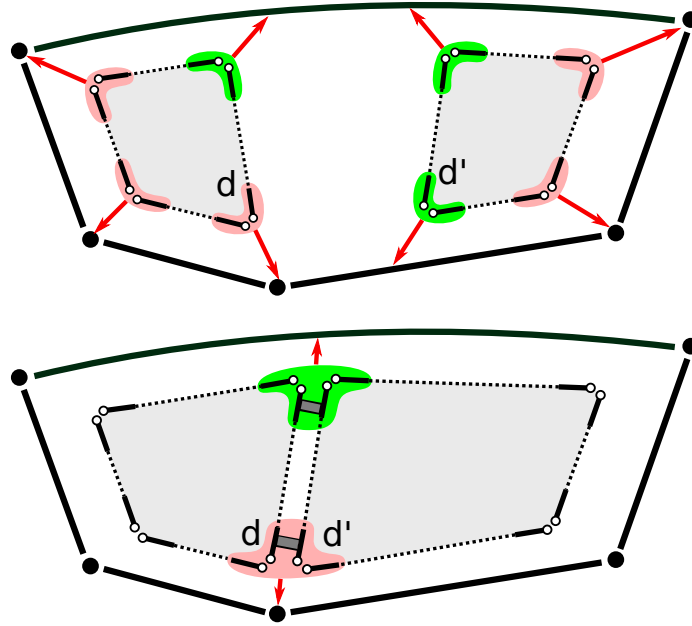


FIGURE 3.4 – Une 2-G-carte dont les cellules sont classifiées sur un modèle géométrique. En haut, les brins des 0-cellules qui sont classifiés sur les points sont colorés en rouge alors que ceux classifiés sur des courbes sont colorés en vert. En bas, les brins  $d$  and  $d'$  sont 2-cousus et leurs 0-cellules sont de fait fusionnées.

La couture de brins a également un impact très fort sur les fonctions de classification  $gc_i$  et de plongement  $e$ . Les propriétés 3.3 et 3.5 imposent à ces fonctions de renvoyer la même valeur pour les brins appartenant aux mêmes cellules. Par exemple, coudre deux brins  $d$  et  $d'$  par  $\alpha_2$  sur la figure 3.4 induit de fusionner les cellules  $\langle \widehat{\alpha_0} \rangle (d)$  et  $\langle \widehat{\alpha_0} \rangle (d')$  en une seule. Ici, cela a un impact direct à la fois sur les fonctions  $e$  et  $gc_0$ . Les deux fonctions donnaient des valeurs différentes avant de coudre pour les brins des cellules  $\langle \widehat{\alpha_0} \rangle (d)$  et  $\langle \widehat{\alpha_0} \rangle (d')$ . Après les opérations de couture, tous ces brins appartiennent à la même 0-cellule. Les fonctions  $e$  et  $gc_0$  doivent maintenant retourner les mêmes valeurs pour ces brins.

L'assurance de la cohérence des fonctions  $gc_i$  et  $e$  lors de l'exécution des opérations de couture se fait automatiquement en fournissant des *règles de fusion* qui dépendent de la sémantique intrinsèque de  $gc_i$  et  $e$ . Nous appliquons les règles suivantes. Lors de la fusion de deux  $i$ -cellules  $c_i^1$  et  $c_i^2$  pour créer la cellule  $c_i^f$ , avec  $d \in c_i^1$  et  $d' \in c_i^2$ , nous avons :

- Si  $gc_i(d) = gc_i(d')$  alors pour tout  $d'' \in c_i^f$ ,  $gc_i(d'') = gc_i(d)$  et  $e(d'')$  est une projection de  $\frac{e(d)+e(d')}{2}$  sur l'entité géométrique  $gc_i(d)$  ;
- Si  $\dim(gc_i(d)) < \dim(gc_i(d'))$  alors pour tout  $d'' \in c_i^f$ ,  $gc_i(d'') = gc_i(d)$  et  $e(d'') = e(d)$  ;
- Si  $\dim(gc_i(d)) > \dim(gc_i(d'))$  alors pour tout  $d'' \in c_i^f$ ,  $gc_i(d'') = gc_i(d')$  et  $e(d'') = e(d')$ .

Sinon cela signifie que les deux  $i$ -cellules sont classifiées sur des entités géométriques distinctes de même dimension et les opérations de couture sont donc impossibles.

### 3.2 . Opérations de manipulation de structures de blocs hexaédriques

La structure des blocs que nous manipulons est entièrement hexaédrique. Par conséquent, la mise à jour de la structure des blocs consiste à modifier la topologie et la géométrie tout en préservant que l'ensemble des cellules de dimension 3 resteront des hexaèdres. Pour cela, nous considérons que nous ne manipulons que des cartes généralisées contenant des quadrilatères uniquement en dimension 2 et des hexaèdres en dimension 3.

**Definition 3.2.1 ( $n$ -G-carte structurée)** Soit  $G = (D, \alpha_0, \dots, \alpha_n)$  une  $n$ -G-carte, avec  $n = 2$  ou  $3$ , alors  $G$  est qualifiée de **structurée** si et seulement si toutes ses  $n$ -cellules sont des quadrilatères pour  $n = 2$  et des hexaèdres pour  $n = 3$ .

Nous considérons aussi deux types d'opérations de modification uniquement, qui sont la suppression et l'insertion de feuillets [41]. Dans cette section, nous présentons chaque opération, la définition dans le modèle des 3-G-cartes, et l'algorithme en pseudo-code correspondant. Mais au préalable, nous commençons par définir la sélection de feuillets qui nous fournira toutes les cellules appartenant à un feuillet.

#### 3.2.1 . Sélection de feuillets

Nous considérons indifféremment les cas 2D et 3D où un feuillet est respectivement composé d'un ensemble de quadrilatères et d'hexaèdres. Nous définissons un feuillet  $S$ , ou couche de cellules, comme un sous-ensemble de cellules (quadrilatères en 2D, hexaèdres en 3D). En partant d'une arête  $e$  du maillage, nous définissons  $E_e$  comme étant le plus petit sous-ensemble d'arêtes qui vérifie :

$$e \in E_e \text{ et } \forall e_i \in E_e \Rightarrow E_{e_i}^{\parallel} \subseteq E_e,$$

avec  $E_{e_i}^{\parallel}$  l'ensemble des arêtes opposées à  $e_i$  dans les cellules qui sont incidentes à  $e_i$ . Le feuillet  $S_H$  est l'ensemble des cellules qui sont incidentes à une arête de  $E_e$  au moins. Des exemples de feuillets 3D sont donnés sur la figure 3.5 où en figurent trois, l'un pouvant être qualifié de simple en jaune, un deuxième s'intersectant lui-même le long d'une corde complète de cellules hexaédriques en rouge et le dernier se touchant lui-même le long de plusieurs faces en vert. Les deuxième et troisième feuillets sont respectivement qualifiés comme étant **auto-intersectant** et **auto-touchant**.



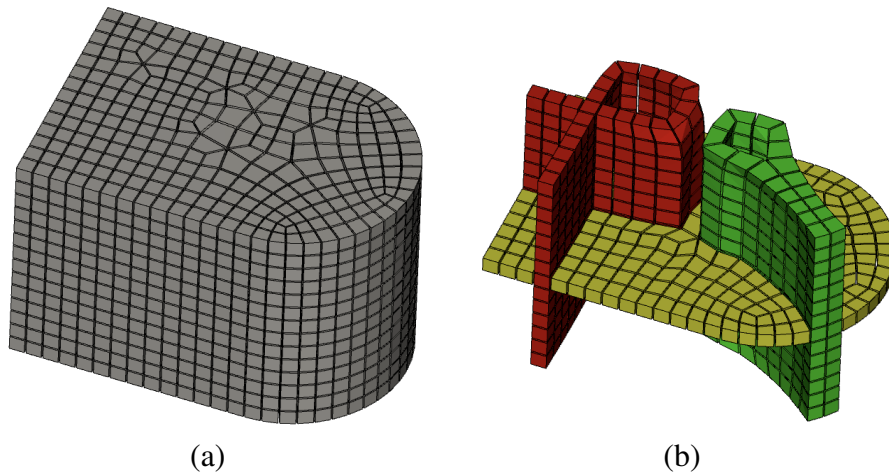


FIGURE 3.5 – Exemple de feuillets 3D dans un maillage hexaédrique. En (a), le maillage complet, en (b) trois feuillets sont représentés : un feuillet régulier (en jaune), un feuillet qui s'auto-intersecte (en rouge) et un feuillet qui se touche lui-même le long de plusieurs faces (en vert).

Dans le modèle des cartes généralisées, le processus de sélection des arêtes consiste à sélectionner un brin  $d$ , qui appartient à une arête, c'est-à-dire à la cellule  $\langle \widehat{\alpha}_1 \rangle (d)$ , puis à sélectionner l'ensemble des arêtes qui sont topologiquement "opposées" à  $\langle \widehat{\alpha}_1 \rangle (d)$ . Cet ensemble d'arêtes se déduit de l'ensemble de brins, appelé **ensemble de brins opposés** et noté  $S_d$ , tel que défini à la définition 3.2.2. Cet ensemble de brins consiste en une orbite spécifique en utilisant les liens  $\alpha_0, \alpha_2$  en 2D, respectivement les liens  $\alpha_0, \alpha_2, \alpha_3$  en 3D, et le lien  $\alpha_{(n-1)\dots 0\dots (n-1)}$  pour "sauter" d'une arête à une autre.

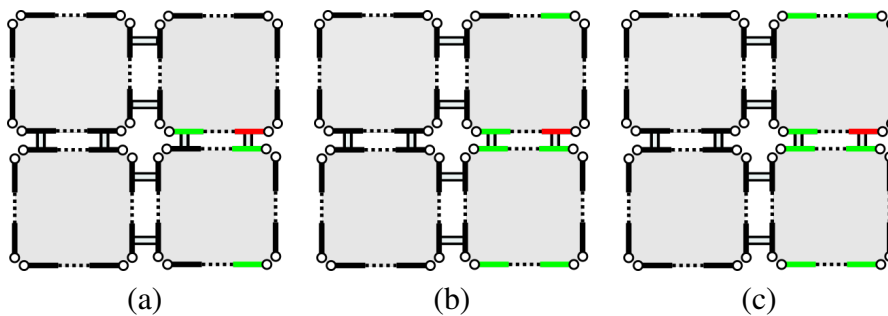


FIGURE 3.6 – Construction de l'ensemble de brins opposés  $S_d$  (voir la définition 3.2.2 et l'algorithme 1). A partir d'un premier brin marqué (en rouge), le front  $F$  est propagé via les liens  $\alpha$  décrits lignes 5, 6 et 8 et les brins sont ensuite ajoutés à  $S_d$ . Les images (a), (b) et (c) représentent  $S_d$  à différentes itérations de l'algorithme.

**Definition 3.2.2 (Ensemble de brins opposés)** Soit  $G = (D, \alpha_0, \dots, \alpha_n)$  une  $n$ -G-carte, avec  $n = 2$  ou  $3$ , et  $d \in D$ , l'ensemble de brins du feuillet opposé défini par  $d$  est l'orbite  $\langle \alpha_{n(n-1)\dots 0\dots(n-1)}, \alpha_0, \alpha_2 \rangle (d)$ .

Dans cette définition et dans la suite, toutes les  $n$ -cellules d'une  $n$ -G-carte sont des quadrilatères pour  $n = 2$  et des hexaèdres pour  $n = 3$ . L'ensemble des brins appartenant aux  $n$ -cellules du feuillet développé à partir d'un brin  $d$  est donné en obtenant les  $n$ -cellules des brins appartenant à  $S_d$ . En partant d'un brin  $d$ , cela indique en 2D que le brin  $d\alpha_{2101}$  appartient également à l'ensemble du feuillet. En 3D, c'est le lien  $\alpha_{321012}$  qui permet d'atteindre un bord opposé. L'algorithme 1 illustré par la figure 3.6 nous donne une méthode naïve mais directe de sélection de brins.

---

**Algorithme 1 : Sélection d'un ensemble de brin opposés.**

---

**Data :** Une  $n$ -G-carte  $G = (D, \alpha_0, \dots, \alpha_n)$  formant un maillage structuré et un brin  $d \in D$

**Result :**  $S_d$  un ensemble de brins

```

1  $F \leftarrow \{d\};$ 
2 while  $F \neq \emptyset$  do
3    $d_i \leftarrow F.first();$  // Pop up the head of  $F$ ;
4   if  $d_i \notin S_d$  then  $S_d \leftarrow S_d + \{d_i\};$ 
5   if  $d_i\alpha_0 \notin S_d$  then  $F \leftarrow F + \{d_i\alpha_0\};$ 
6   if  $d_i\alpha_2 \notin S_d$  then  $F \leftarrow F + \{d_i\alpha_2\};$ 
7   if  $d_i\alpha_{n(n-1)\dots 0\dots(n-1)} \notin S_d$  then
8      $F \leftarrow F + \{d_i\alpha_{n(n-1)\dots 0\dots(n-1)}\};$ 
9   end
10 end

```

---

### 3.2.2 . Suppression de feuillet

La suppression de feuillet consiste à le retirer entièrement de la structure de blocs. La figure 3.7 illustre la procédure globale dans un cas simple où le feuillet représenté par des faces quadrilatérales rouges figure 3.7(a) sera totalement supprimé dans la figure 3.7(f). Nous définissons la  $n$ -G-carte résultante de la suppression d'un feuillet comme suit :

**Definition 3.2.3 (Suppression de feuillet)** Soit  $G = (D, \alpha_0, \dots, \alpha_n)$  une structure  $n$ -G-carte structurée, avec  $n = 2$  ou  $3$ ,  $d \in D$ , et  $S_d$ , l'ensemble des brins opposés du feuillet défini à partir du brin  $d$ . La suppression du feuillet déduit de  $d$  produit la  $n$ -G-carte  $G' = (D', \alpha'_0, \dots, \alpha'_n)$  telle que :

1.  $D' = D - \langle \widehat{\alpha_n} \rangle (S_d)$ ,
2.  $\forall i \in \llbracket 0; n-1 \rrbracket, \forall d' \in D', d'\alpha'_i = d'\alpha_i$ ,

$$3. \forall d' \in D', d' \alpha'_n = \begin{cases} d' \alpha_n & \text{if } d' \alpha_n \notin \langle \widehat{\alpha}_n \rangle (S_d), \\ d_k \alpha_n & \text{autrement,} \end{cases}$$

avec

$$d_k = d'(\alpha_{n(n-1)..0..(n-1)})^k$$

et  $k$  le plus petit entier positif tel que

$$d'(\alpha_{n(n-1)..0..(n-1)})^k \alpha_n \notin \langle \widehat{\alpha}_n \rangle (S_d).$$

Cette définition mérite quelques commentaires. Le point 1 indique que l'ensemble de brins constituant la carte généralisée  $G'$  est composé des brins de  $G$  sans les brins de  $\langle \widehat{\alpha}_n \rangle (S_d)$  tel que défini par la définition 3.2.2. Le point 2 indique que les liens  $\alpha_0$  à  $\alpha_{n-1}$  demeurent inchangés. Seul le lien  $\alpha_n$  est modifié pour les brins qui appartiennent à  $D - \langle \widehat{\alpha}_n \rangle (S_d)$  et qui sont liés par  $n$  à un brin de  $\langle \widehat{\alpha}_n \rangle (S_d)$  (voir la figure 3.7). Après la suppression du feuillet, chacun de ces brins est  $n$ -lié au premier brin qui n'appartient pas à  $\langle \widehat{\alpha}_n \rangle (S_d)$  et que nous pouvons atteindre en appliquant une composition de liens  $(\alpha_{n(n-1)..0..(n-1)})^k \alpha_n$  avec  $k > 0$ . Cette composition de liens est obligatoire pour supprimer les feuillets qui se touchent. Notons que le feuillet à supprimer peut être un feuillet au bord (voir figure 3.8-a). Dans ce cas, le brin  $d'$  atteint en appliquant  $(\alpha_{n(n-1)..0..(n-1)})^k \alpha_n$  à partir d'un brin  $d$  est le brin  $d$  lui-même :  $d' = d$ . Cela ne pose pas de problème d'un point de vue topologique mais nécessite une mise à jour prudente de l'attribut du brin gérant la classification géométrique ainsi celui gérant le plongement géométrique.

Afin d'éviter un cas spécifique de gestion des attributs lorsque nous réduisons un feuillet au bord, l'algorithme que nous développons suit une approche en deux étapes après la phase d'initialisation, illustré dans l'algorithme 2 :

- **Initialisation** - Nous définissons d'abord tous les brins qui seront utilisés dans la suppression du feuillet (ligne 1). Nous obtenons ainsi les brins opposés en vert sur la figure 3.7. Des lignes 6 à 10, nous marquons les brins des  $n$ -cellules qui sont dans le feuillet. Ensuite, nous récupérons les brins rouges en appliquant  $\alpha_1$  sur les brins verts et nous conservons les brins rouges opposés dans chaque 1-cellule. A partir des brins rouges, nous obtenons les brins bleus de la figure 3.7(b) dans la  $n$ -cellule voisine en appliquant l'involution  $\alpha_2$  en 2D et la composition  $\alpha_{232}$  en 3D. Le brin bleu opposé est alors conservé en appliquant  $(\alpha_{n(n-1)..0..(n-1)})^k \alpha_n$  jusqu'à atteindre un brin non marqué dans le feuillet. Cela correspond au point 3 de la définition 3.2.3 (voir les lignes 12 à 19).
- **Phase (1)** - Commence alors la première étape de l'opération de suppression du feuillet. Nous commençons par retirer les brins du feuillet opposé (c'est-à-dire les brins verts) comme l'illustre la figure 3.7(c). Cela libère le lien  $\alpha_1$  des brins rouges restants (ligne 20). Nous pouvons maintenant coudre par  $\alpha_1$  les brins rouges opposés et créer ainsi une  $n$ -cellule à la ligne 21 et montré figure 3.7(d).
- **Phase (2)** - La deuxième étape débute avec la suppression des  $n$ -cellules contenant des brins rouges sur la figure 3.7(e). Cette étape nous permet de vérifier le

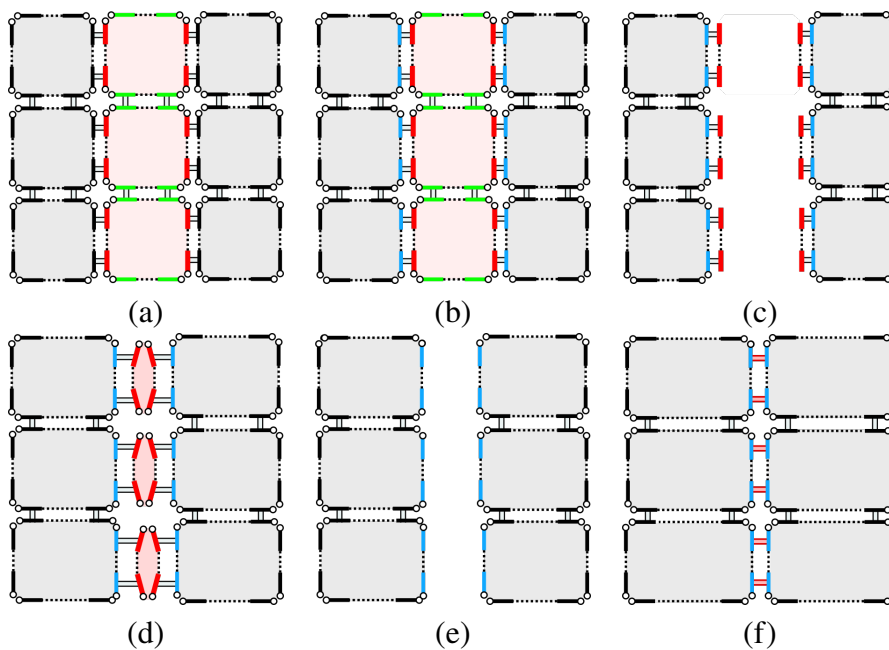


FIGURE 3.7 – Les brins de  $S_d$  sont colorés en vert puis étendus aux brins rouges pour obtenir tous les brins de  $\langle \widehat{\alpha}_2 \rangle (S_d)$  en (a). A partir de cet ensemble de brins, nous pouvons atteindre les brins bleus mis en évidence en (b) en utilisant l'involution  $\alpha_2$ ; les brins verts sont ensuite retirés de la G-carte et les brins rouges deviennent 1-libres en (c); Les brins rouges opposés sont alors 1-cousus en (d) avant d'être retirés en (e). Les brins bleus sont alors cousus par  $\alpha_2$  entre brins opposés en (f).

point 1 de la définition 3.2.3 à la ligne 22. Comme nous avons supprimé tous les brins du feuillet à supprimer, les brins bleus sont  $n$ -libres. La dernière étape consiste à les coudre à leurs brins bleus opposés (lignes 23) pour obtenir le résultat montré sur la figure 3.7(f).

Pendant toute la durée de l'algorithme, les seules modifications apportées aux liens des brins de la  $n$ -G-carte qui ne sont pas dans le feuillet à effondrer sont les  $n$ -liens des brins n'appartenant pas aux cellules du feuillet. Ceci vérifie le point 2 de la définition 3.2.3.

---

**Algorithme 2 : Sheet Collapse**

---

**Data :**  $n$ -G-map  $H = (D, \alpha_0, \dots, \alpha_n)$ , a dart  $d \in D$

```

1  $S_d \leftarrow \text{sheetSelection}(d)$ ; /* Algorithm 1 */
2  $\text{in\_sheet}(x) : \{d \in D\} \rightarrow \{\text{false}, \text{true}\}$ ;
3  $\forall d \in D, \text{in\_sheet}(d) \leftarrow \text{false}$ ;
4  $\text{collapse} \leftarrow \emptyset$ ;
5  $\text{to\_collapse}(x) : \{d \in D\} \rightarrow \{d' \in D\}$ ;
6 for  $d \in S_d$  do
7    $\text{in\_sheet}(\langle \widehat{\alpha}_n \rangle (d)) \leftarrow \text{true}$ ;
8    $\text{collapse} \leftarrow \text{collapse} + \{d\alpha_1\}$ ;
9    $\text{to\_collapse}(d\alpha_1) \leftarrow d\alpha_0$ ;
10 end
11  $\text{sew} \leftarrow \emptyset$ ;  $\text{to\_sew}(x) : \{d \in D\} \rightarrow \{d' \in D\}$ ;
12 for  $d_c \in \text{collapse}$  do
13    $d_X \leftarrow d_c\alpha_X$ ; //  $X = 232$  in 3D,  $X = 2$  in 2D;
14   if  $d_X \notin \text{collapse}$  then
15      $\text{sew} \leftarrow \text{sew} + \{d_X\}$ ;  $k \leftarrow 1$ ;
16     while  $\text{in\_sheet}(d_X(\alpha_{n(n-1)..0..(n-1)})^k \alpha_n)$  do  $k \leftarrow k + 1$ ;
17      $\text{to\_sew}(d_X) \leftarrow d_X(\alpha_{n(n-1)..0..(n-1)})^k \alpha_n$ ;
18   end
19 end
20 for  $d \in S_d$  do  $\text{remove}(d)$ ;
21 for  $d_c \in \text{collapse}$  do  $1\text{-sew}(d_c, \text{to\_collapse}(d_c))$ ;
22 for  $d_c \in \text{collapse}$  do  $\text{remove}(\langle \widehat{\alpha}_n \rangle (d_c))$ ;
23 for  $d_c \in \text{sew}$  do  $n\text{-sew}(d_s, \text{to\_sew}(d_s))$ ;

```

---

Cette approche en deux étapes est conçue pour traiter des cas particuliers de configurations de feuillets comme ceux illustrés par la figure 3.8 avec la gestion de feuillets auto-intersectants en (a), de feuillets auto-touchants en (b) et de feuillets au bord en (c). Les configurations de feuillets auto-intersectants et auto-touchants sont traitées durant la phase d'initialisation. La ligne 14 garantit qu'aucun brin rouge intérieur n'est considéré comme étant un brin bleu et la ligne 16 permet d'associer

chaque brin bleu à son brin opposé à travers plusieurs couches de feuillets auto-touchants. La définition de l'opération de couture associée aux règles de fusion des fonctions  $gc_i$  et  $e$  (voir la section 3.1.3) garantit la modification de la géométrie. En particulier, nous ne perdons pas les entités géométriques de dimension inférieure, ce qui nous permet de traiter le cas de la suppression d'un feuillet au bord du domaine, comme le montre la figure 3.8(c), où la couture des brins rouges raccroche le côté droit au bord à gauche.

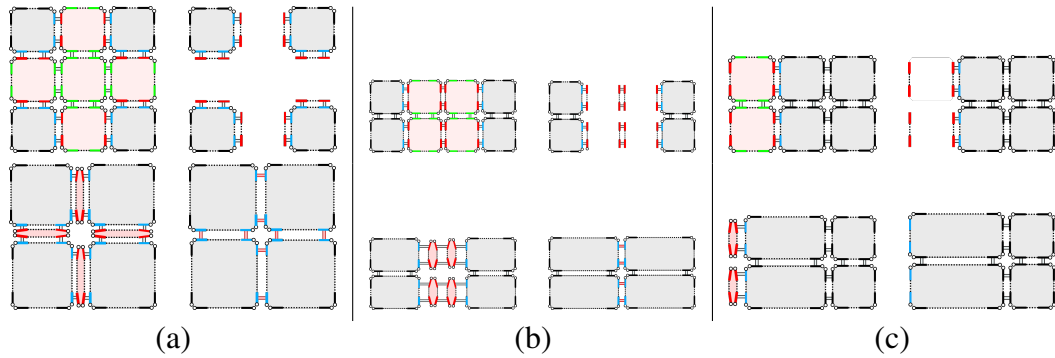


FIGURE 3.8 – Les quatre étapes principales de la suppression pour les feuillets auto-intersectants (a), les feuillets auto-touchants (b), et les feuillets au bord (c). En haut à gauche, nous définissons les brins à enlever (verts, rouges) et à coudre (bleus); en haut à droite, nous enlevons les brins verts; en bas à gauche, nous cousons les brins rouges et en bas à droite, nous cousons finalement les brins bleus.

L'algorithme 2 s'applique aussi bien pour supprimer un feuillet en dimension 2 qu'en dimension 3 (voir la figure 3.9 pour un cas en dimension 3). Dans le second cas, le seul changement se trouve à la ligne 13 de l'algorithme lorsque nous voulons obtenir le brin de la  $n$ -cellule voisine. Nous définissons  $X$  où  $X = 2$  en 2D et  $X = 232$  en 3D et nous appliquons donc  $\alpha_X$ , c'est-à-dire  $\alpha_2$  ou  $\alpha_{232}$ . Cela nous permet d'obtenir les brins bleus de la figure 3.9(b) en 3D de la même manière qu'en 2D.

### 3.2.3 . Insertion de feuillet

L'insertion de feuillet est beaucoup plus complexe à définir que la suppression de feuillet. La façon traditionnelle d'insérer un feuillet est d'effectuer une opération appelée *pillowing* en anglais [41]. Après avoir sélectionné un ensemble  $P$  de  $n$ -cellules, appelé un *pillow set*, l'opération de *pillowing* consiste à insérer une nouvelle couche de  $n$ -cellules, c'est-à-dire à ajouter un *oreiller*, qui entoure  $P$  et l'isole du reste de la structure de blocs (voir la figure 3.10 pour un exemple). Le fait d'avoir un ensemble d'éléments hexaédriques en paramètre d'entrée de cette opération ne permet pas d'insérer des feuillets qui s'auto-intersectent et se touchent. Pour remédier à cela, nous proposons une opération plus générique, où nous allons sélectionner un ensemble connexe de  $(n - 1)$ -cellules, ou *hyperplan*, qui vérifie plusieurs conditions. Pour exprimer ces conditions, nous introduisons d'abord la notion de carte généralisée

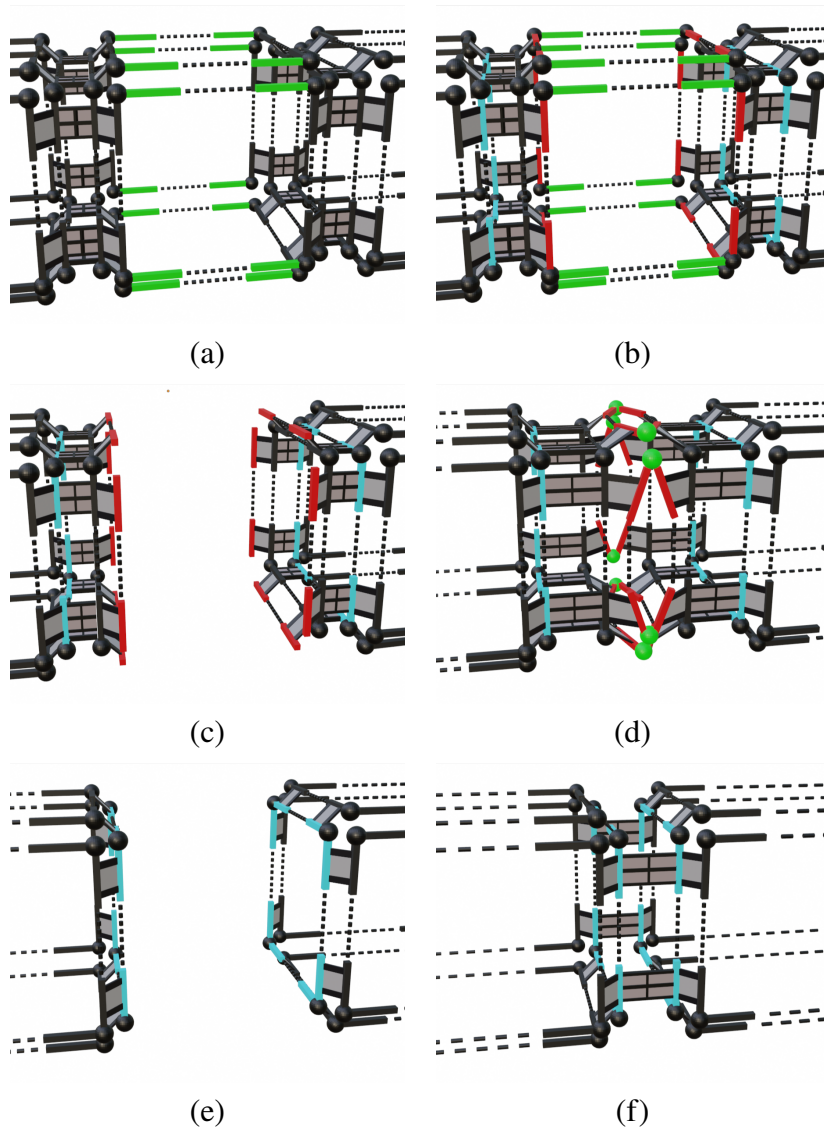


FIGURE 3.9 – Une vue approchée de l'opération d'effondrement d'un feuillet en dimension 3, qui suit les même étapes que celles illustrées en dimension 2 sur la figure 3.7.

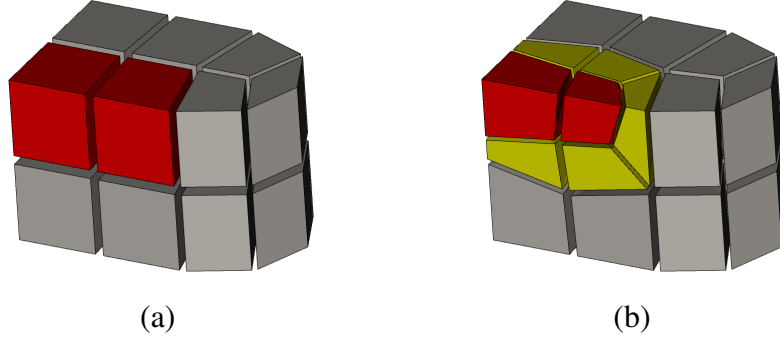


FIGURE 3.10 – Exemple d’opération de *pillowing* où les blocs rouges sont sélectionnés en (a). L’opération de *pillowing* isole ces deux blocs des autres blocs et insère un feuillet complet en jaune (b).

fantôme. Ce nom est choisi par analogie avec la notion de couche fantôme de cellules utilisée dans les simulations de calcul haute performance (HPC) pour dupliquer les cellules frontières à travers les unités de calcul. Cette duplication permet d’éviter des communications inter-processus inutiles. Dans notre cas, nous enrichissons une  $n$ -G-carte structurée avec des brins supplémentaires sur son bord. Ces brins nous permettent d’éviter de traiter des cas particuliers pour les éléments du bord.

**Definition 3.2.4 (n-G-carte fantôme)** Soit  $G = (D, \alpha_0, \dots, \alpha_n)$  une  $n$ -G-carte structurée, avec  $n = 2$  ou  $3$ , et  $\partial_n D$  un ensemble de brins  $n$ -libre de  $G$ . Soit  $\{f_i : \partial_n \rightarrow D_i\}_{i \in \llbracket 1; n \rrbracket}$   $n$  fonctions d’association telles que

$$D_i \cap \partial_n D = \emptyset \text{ and } D_i \cap D_j = \emptyset, \forall (i, j) \in \llbracket 1; n \rrbracket^2$$

avec  $i \neq j$ . La  $n$ -G-carte fantôme de  $G$ , notée  $G^s$  est la  $n$ -G-carte  $(D^s, \alpha_0^s, \dots, \alpha_n^s)$  telle que :

1.  $D^s = D + D_1 + \dots + D_n$ ,
2.  $\forall d \in D, \forall i \in \llbracket 0; n-1 \rrbracket, d\alpha_i^s = d\alpha_i$ ,
3.  $\forall d \in D^s, d\alpha_n^s = \begin{cases} d\alpha_n & \text{if } d \in D - \partial_n D, \\ df_n & \text{if } d \in \partial_n D, \\ d & \text{if } d \in f_i(\partial_n D), i < n \end{cases}$
4.  $\forall d \in \partial_n D, \forall i \in \llbracket 1; n-1 \rrbracket, df_i\alpha_i^s = df_{i+1}$ ,
5.  $\forall d \in f_1(\partial_n D), d\alpha_n^s = d'$ , avec  $d' \in \partial_n D$  et

$$\exists k > 0 / d(\alpha_{n-1}^s \alpha_n^s)^k \alpha_{n-1}^s = d'$$

La  $n$ -G-carte fantôme  $G^s$  enrichit la  $n$ -G-carte  $G$  en ajoutant une couche de cellules  $n$  partielles autour de  $G$ . La figure 3.13.(b) montre l’ensemble des brins qui ont été ajoutés. Pour chaque brin au bord  $d$  de  $G$ , deux brins  $d_1$  et  $d_2$  sont ajoutés de



telle sorte que  $d_1 = d\alpha_2$  et  $d_2 = d_1\alpha_1$  (points 1, 2, 3 et 4 de la définition 3.2.4). Le point 5 assure de zipper les sommets contenant les nouveaux brins en dimension 2 (et les arêtes en dimension 3). Nous pouvons maintenant définir les conditions qu'un ensemble connexe de  $(n - 1)$ -cellules doit vérifier afin d'être utilisé pour l'insertion de feuillet.

**Definition 3.2.5 (Hyperplan admissible)** Soit  $G = (D, \alpha_0, \dots, \alpha_n)$  une  $n$ - $G$ -carte structurée, avec  $n = 2$  ou  $3$  et  $\mathcal{H}$  un sous-ensemble de  $D$ . Soit  $G^g = (D^g, \alpha_0^g, \dots, \alpha_n^g)$  la  $n$ - $G$ -carte fantôme de  $G$ . Alors  $\mathcal{H}$  définit un hyperplan admissible de  $G$  si et seulement si :

1.  $d \in \mathcal{H} \Rightarrow \langle \alpha_0, \dots, \alpha_{n-2} \rangle (d) \subset \mathcal{H}$ ,
2.  $\forall d \in \mathcal{H}, |\langle \widehat{\alpha_{n-2}^g} \rangle (d) \cap \mathcal{H}| = 2$  ou  $4$
3.  $\forall d \in \mathcal{H}, |\langle \widehat{\alpha_{n-2}^g} \rangle (d) \cap \mathcal{H}| = 2 \Rightarrow |\langle \widehat{\alpha_{n-1}^g} \rangle (d) \cap \mathcal{H}| = 0$ .

Le premier point de la définition 3.2.5 assure que si un brin d'une demi-arête en dimension 2, ou d'une demi-face en dimension 3, est dans  $\mathcal{H}$  alors tous les brins de l'ensemble de la demi-arête en dimension 2, respectivement de la demi-face en dimension 3, sont aussi inclus dans  $\mathcal{H}$ . Par demi, nous entendons ici un "côté", c'est-à-dire une arête vue d'une face en dimension 2 et une face vue d'un bloc hexaédrique en dimension 3. Les figures 3.11(a) et 3.11(b) l'illustrent en dimension 2 : les brins de quatre "demi"-arêtes sont sélectionnés. Le deuxième point indique qu'autour d'un sommet en dimension 2, respectivement d'une arête en dimension 3, l'hyperplan peut se croiser au plus une fois. Si  $|\langle \widehat{\alpha_{n-2}^g} \rangle (d) \cap \mathcal{H}| = 2$ , il ne se croise pas au niveau de la  $(n - 2)$ -cellule en question. Au contraire, si  $|\langle \widehat{\alpha_{n-2}^g} \rangle (d) \cap \mathcal{H}| = 4$ , il se croise au niveau de cette  $(n - 2)$ -cellule. C'est le cas pour les figures 3.11(a) et 3.11(b) mais pas pour les figures 3.11(c) et (d). Le dernier point est nécessaire pour éviter le cas particulier où deux demi-cellules auraient été sélectionnées alors qu'elles appartiennent à la même cellule (voir la figure 3.11(d) par exemple).

Soulignons qu'en limitant  $|\langle \widehat{\alpha_{n-2}^g} \rangle (d) \cap \mathcal{H}|$  à 4 dans la condition 2 de la définition 3.2.5, nous ne considérons que l'intersection de quatre  $(n - 1)$ -cellules qui créeront un seul quadrilatère en dimension 2, ou un hexaèdre en dimension 3. Un hyperplan admissible  $\mathcal{H}$  dans une  $n$ - $G$ -carte  $H$  est défini en utilisant l'extension fantôme de  $H$ . Les définitions 3.2.6 à 3.2.7 nous donnent les notions essentielles pour définir formellement le processus d'insertion de feuillet détaillé dans l'algorithme 3 et qui correspond à la définition 3.2.9. Tout d'abord, pour effectuer l'insertion de feuillet, nous étendons l'ensemble des brins de  $\mathcal{H}$  dans la carte étendue.

**Definition 3.2.6 (Hyperplan admissible fantôme)** Soit  $G = (D, \alpha_0, \dots, \alpha_n)$  une  $n$ - $G$ -carte structurée, avec  $n = 2$  ou  $3$  et  $\mathcal{H}$  un hyperplan admissible dans  $H$ . On étend  $\mathcal{H}$  dans la  $n$ - $G$ -carte fantôme  $G^g$  comme suit :

1.  $H^g = \mathcal{H}$  sur  $D$  ;

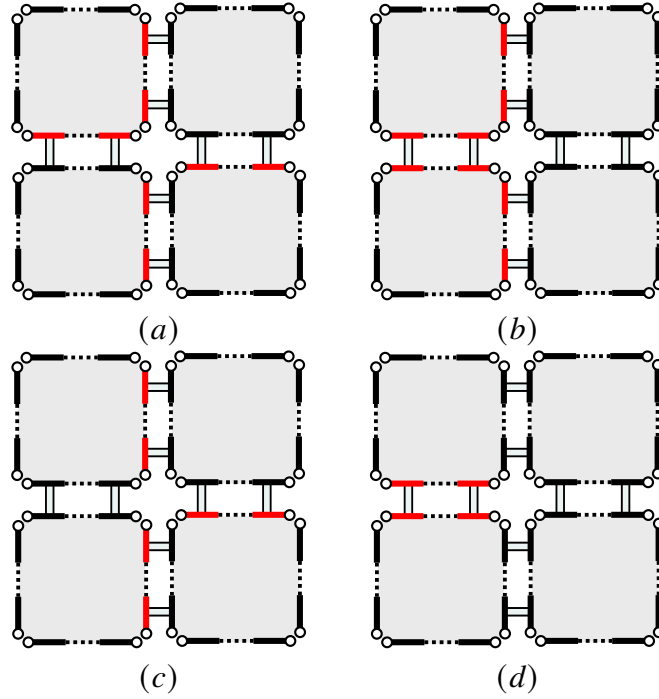


FIGURE 3.11 – Exemples 2D d’hyperplans admissibles en (a) et (b) et d’hyperplans non admissibles en (c) et (d).

2.  $\forall d \in H^g$ , si il existe  $d' \in \langle \widehat{\alpha}_0^g \rangle (d)$  tel que  $d' \in \mathcal{H}$ , alors  $d \in \mathcal{H}^g$ .

La deuxième propriété de la définition 3.2.6 garantit que si un brin  $d$  appartient à l’hyperplan, alors tous les brins du sommet contenant  $d$  sont également dans l’hyperplan. Cela nous donne une configuration homogène pour appliquer le même motif à chaque brin de  $\mathcal{H}^g$ . Nous disposons d’un motif spécifique en dimension 2 et d’un autre en dimension 3 (voir la figure 3.12). Les deux définitions suivantes introduisent formellement ces motifs.

**Definition 3.2.7 (Motif d’insertion 2D)** Nous définissons  $P_2 = \{d_0, \dots, d_5\}$  le motif fait de 6 brins tel que tous ces brins sont 0, 1 et 2-libres et que  $d_1 = d_0\alpha_1$ ,  $d_2 = d_0\alpha_{10}$ ,  $d_3 = d_0\alpha_{12}$ ,  $d_4 = d_0\alpha_{102}$  et  $d_5 = d_0\alpha_{101}$  (voir la figure 3.12(a)).

**Definition 3.2.8 (Motif d’insertion 3D)** Nous définissons  $P_3 = \{d_0, \dots, d_{11}\}$  le motif fait de 12 brins tel que tous ces brins sont 0, 1, 2 et 3-libres et que  $d_{11} = d_0\alpha_{21012}$ ,  $d_1 = d_0\alpha_2$ ,  $d_2 = d_0\alpha_{21}$ ,  $d_3 = d_0\alpha_{210}$ ,  $d_4 = d_0\alpha_{2101}$ ,  $d_5 = d_0\alpha_{2131}$ ,  $d_6 = d_0\alpha_{213}$ ,  $d_7 = d_0\alpha_{2103}$ ,  $d_8 = d_0\alpha_{21013}$ ,  $d_9 = d_0\alpha_{2132}$  et  $d_{10} = d_0\alpha_{21032}$  (voir les figures 3.12(b) et 3.12(c)).

Nous pouvons maintenant formellement définir l’opération d’insertion. Nous notons  $p(d)$  l’ensemble de brins qui correspond au motif  $P_n$  pour le brin  $d$ . La notation

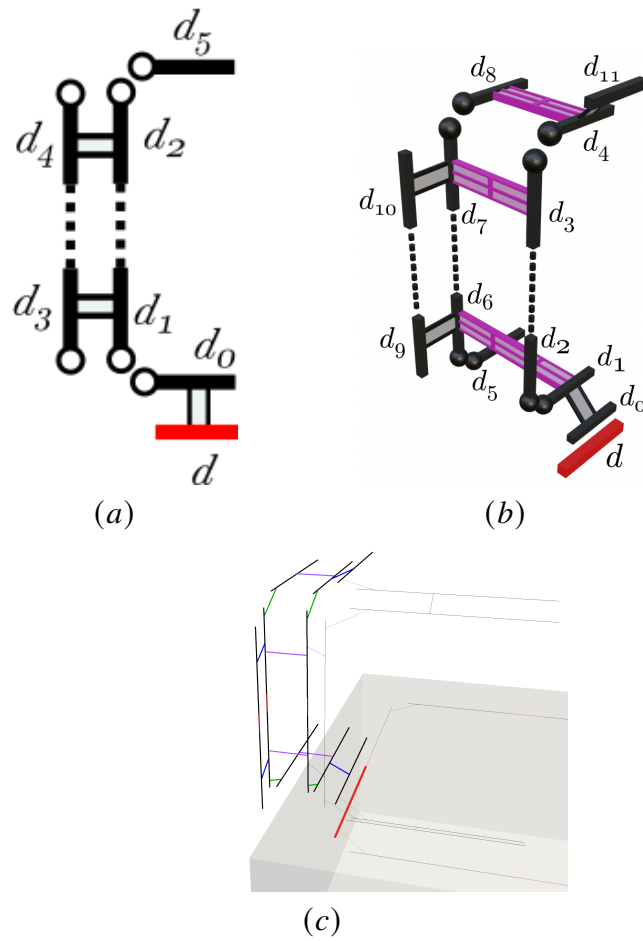


FIGURE 3.12 – Modèles d’insertion en dimension 2 (a) et en dimension 3 (b). Pour le brin rouge, nous insérons une configuration de 6 brins en dimension 2 et de 12 en dimension 3 (avec des liens  $\alpha_3$ ). En (c), est illustrée la configuration 3D où les liens  $\alpha_0$  vers les brins produits par d’autres brins marqués sont établis sur la base de  $\alpha_0(d)$  ( $\alpha_1$  respectivement).

$p(d).f$  correspond au premier brin de  $P_n$ . La notation  $p(d).l$  correspond au dernier brin de  $P_n$  et  $p_i(d)$  est le  $i^{\text{me}}$  brin.

**Definition 3.2.9 (Insertion de feuillet)** Soit  $G = (D, \alpha_0, \dots, \alpha_n)$  une  $n$ -G-carte structurée, avec  $n = 2$  ou  $3$ , et  $\mathcal{H}$  un hyperplan admissible de  $G$ . L'insertion de feuillet caractérisée par  $\mathcal{H}$  est la  $n$ -G-carte structurée  $G' = (D', \alpha'_0, \dots, \alpha'_n)$  telle que :

1.  $D' = D + \{p(d)\}_{d \in \mathcal{H}}$ ,
2.  $\forall i \in \llbracket 0; n-1 \rrbracket, \forall d \in D, d\alpha'_i = d\alpha_i$ ,
3.  $\forall d \in D'$ ,

$$d\alpha'_n = \begin{cases} d\alpha_n & \text{if } d \in D - \widehat{\langle \alpha_{n-1} \rangle} (\mathcal{H}^g) \\ p(d).f & \text{if } d \in \mathcal{H}^g, \\ p(d').l & \text{if } d' \in \mathcal{H}^g \wedge d\alpha_n = d'. \end{cases}$$

4.  $\forall d \in \{p(d)\}_{d \in \mathcal{H}^g}, \exists k_1 > 0 / d(\alpha'_{n-1}\alpha'_n)^{k_1} = d$  et  $\exists k_2 > 0 / d(\alpha'_{n-2}\alpha'_{n-1})^{k_2} = d$ .

Pour tenter d'expliquer cette définition, considérons le cas 2D illustré sur la figure 3.13. En partant de l'hyperplan  $\mathcal{H}$  (brins rouges) passé en paramètre d'entrée (voir la figure 3.13.a), nous montrons l'extension de la couche fantôme en (b) et l'hyperplan fantôme  $\mathcal{H}^g$ . Ensuite, pour chaque brin de  $\mathcal{H}^g$ , nous montrons le motif 2D inséré (voir définition 3.2.7) sur la figure 3.13.c. Nous avons ici tous les brins de  $D'$  (point 1 de la définition 3.2.9). Le second point indique que nous préservons tous les liens  $\alpha_i$  pour les brins de  $G$ , pour tout  $i$  dans  $\llbracket 0; n-1 \rrbracket$ . Le troisième point indique que nous préservons également les liens  $\alpha_2$  pour les brins qui ne sont pas impliqués dans le processus d'insertion du feuillet (ceux qui ne sont pas connectés à un brin de  $\mathcal{H}^g$ ). Nous reconnectons également les motifs insérés via  $p(d).l$  et  $p(d).f$  (voir la figure 3.13.d). Comme  $G'$  est une 2-G-carte, certains liens  $\alpha_0$  sont implicitement effectués pour s'assurer que  $\alpha_{02}$  est une involution (voir la figure 3.13.e). Le quatrième point ferme quelques cellules ouvertes. En dimension 2, les brins  $p_3(d)$  et  $p_4(d)$  sont 1-liés avec le premier brin 1-libre  $m$  de leurs orbites  $\langle \alpha_1, \alpha_2 \rangle (d), d \neq m$  (voir la figure 3.13.d). (autour du sommet en dimension 2 et autour de l'arête en dimension 3). Cela nous procure le résultat de la figure 3.13.f.

A partir de la définition 3.2.9, nous dérivons l'algorithme 3, qui définit l'insertion d'un feuillet à la fois en dimension 2 et 3. Seule l'étape de liaison diffère entre le cas 2D et le cas 3D (ligne 6). Nous continuons à utiliser la figure 3.13 pour expliquer l'algorithme. Étant donné un ensemble de sélection  $D_e$ , qui est un hyperplan admissible (voir la définition 3.2.5), nous insérons d'abord une couche fantôme (ligne 1), comme défini à la définition 3.2.4, pour passer de la figure 3.13.a à la figure 3.13.b. Pour faciliter la suppression finale de la couche fantôme, nous marquons le brin d'un motif généré pour la couche fantôme comme devant être supprimé ultérieurement. L'utilisation des couches fantômes et de l'hyperplan fantôme (ligne 2) nous permet

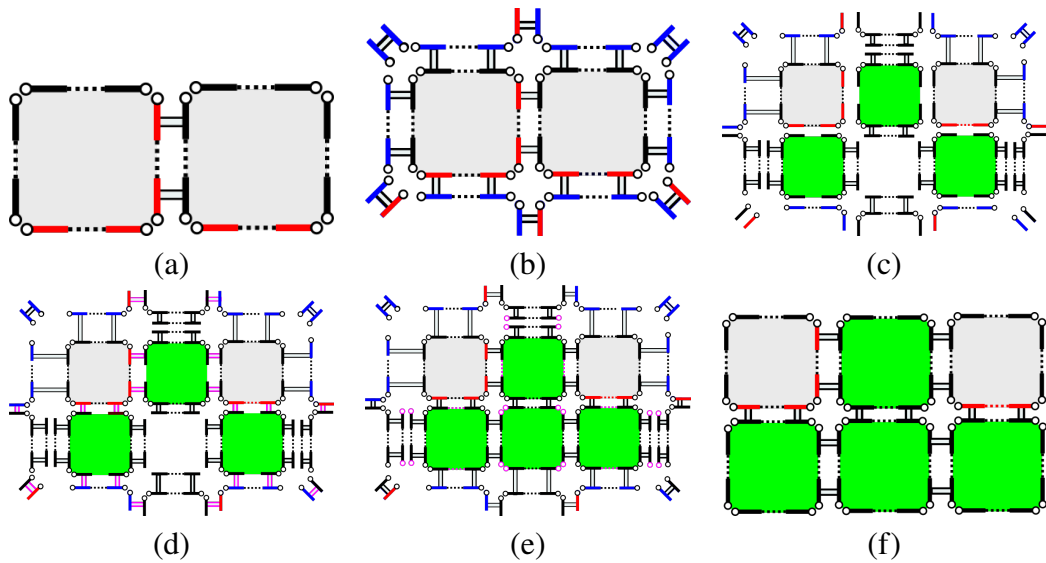


FIGURE 3.13 – Pipeline de la 2-Insertion pour les brins sélectionnés en rouge (voir l’algorithme 3). En (a), la sélection des brins passés en entrée est en rouge ; En (b), est ajoutée la couche fantôme en bleu, avec l’extension de la sélection en rouge aux brins de cette couche ; En (c), on mémorise  $\alpha_n^p$ , sélection des brins 2-décousus des brins sélectionnés (d) 2-lien  $p(d).l$  et  $p(d).f$  ; (e) 0-lien  $p(d).f, p(d).l$  et 1-lien  $p_3, p_4$  ; (f) le résultat avec fermeture, faces 2-compressés et couche fantôme supprimée.

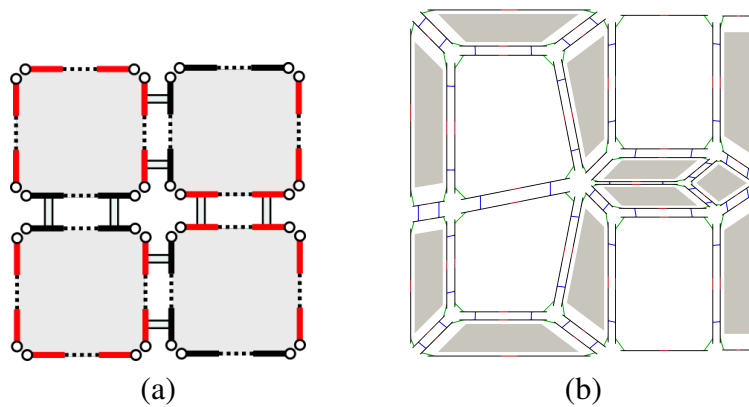


FIGURE 3.14 – (a) Brins marqués ; (b) Après une opération d’insertion du feuillet et suppression de la face insérée centrale (voir Fig. 3.15).

---

**Algorithme 3 : n-Insertion**

---

**Data :** A  $n$ -G-map  $H = (D, \alpha_0, \dots, \alpha_n)$ , a set of selected darts  $D_e$

- 1 Add a ghost-layer  $g$  on the boundary of  $H$ ;
- 2  $D_e \leftarrow D_e +$  darts selected in  $g$  (see Def. 3.2.6);  
/\* Fig. 3.13.(b) \*/
- 3 Memorize  $\alpha_n^p, \forall d \in D_e$ ;
- 4  $n - unsew$  all darts  $\forall d \in D_e$ ;
- 5 Generate local nD-pattern  $\forall d \in D_e$ ;  
/\* Fig. 3.2.7 Fig. 3.2.8, Fig. 3.13.(c) \*/
- 6 Link patterns ;  
/\* Fig. 3.13.(d)-(e) \*/
- 7 Collapse compressed n-cells ;
- 8 Remove  $g$  ;  
/\* Fig. 3.13.(f) \*/

---

d'écrire un algorithme générique sans avoir à considérer des cas spécifiques pour les brins du bord.

Notons que contrairement à la définition 3.2.9, nous mettons à jour de manière incrémentale la  $n$ -G-carte dans l'algorithme 4. Nous stockons donc la fonction initiale  $\alpha_n$ , notée  $\alpha_n^p$ , qui sera utilisée ultérieurement pour insérer les motifs (ligne 3). Après cela, nous découpons tous les brins de  $D_e$  pour  $\alpha_n$ . Ensuite, le motif est inséré pour chaque brin  $d \in D_e$  mais sans les lier à la  $n$ -G-carte initiale (voir la figure 3.13.c).

La ligne 6 de l'algorithme 3 diffère entre le cas 2D et le cas 3D. Elle correspond au quatrième point de la définition 3.2.9. Nous donnons quelques détails sur leur mise en œuvre par la suite.

À la ligne 7, nous supprimons les  $n$ -cellules compressées. Le motif d'insertion en dimension 2, donné dans la définition 3.2.7, génère des 2-cellules compressées (voir figure 3.16.a) et nous obtenons le même type de cellules compressées en dimension 3 sur la figure 3.16.b. Que ce soit en dimension 2 ou 3, nous détectons de telles cellules lorsqu'elles possèdent au moins un brin  $d$  tel que  $d\alpha_{0101} = d$ . Une fois détectées, nous supprimons chaque  $n$ -cellule compressée. En dimension 2, on supprime une 2-cellule compressée. En dimension 3, une 3-cellule compressée  $C_3$  est supprimée et nous cousons  $\alpha_{23}(d)$  avec  $\alpha_{123}(d)$  pour  $d$  un brin de  $C_3$ . Enfin, la couche fantôme est supprimée (ligne 8).

**Liens en 3D**, nous commençons par relier les brins  $\{0, 1, 4, 5, 8, 11\}$  d'un brin sélectionné  $d$  avec les brins de  $p(d\alpha_0)$ . Par exemple, le brin  $p_0(d)$  est lié au brin 0 avec  $p_0(d\alpha_0)$ . Ensuite, les brins  $p(d).f$ ,  $p(d).\ell$  et  $p_{2,3}(d)$  sont respectivement liés par  $\alpha_1$  et  $\alpha_2$  aux brins de  $p(d\alpha_1)$  (voir la figure 3.12.c).

---

4. Cela implique que les propriétés de la  $n$ -G-carte ne sont pas nécessairement vérifiées au cours de l'algorithme, mais seulement à la fin.

Nous lions maintenant les brins  $p_{5,9}(d)$  au motif engendré par le brin marqué  $d'$  trouvé sur l'orbite  $\langle \alpha_2 \rangle (d)$ , avec  $d' \neq d$ . Le brin  $p_5(d)$  est lié par  $\alpha_2$  à  $p_5(d')$  tandis que  $p_9(d)$  est lié par  $\alpha_1$  à  $p_9(d')$ ; s'il n'y a pas de  $d'$  de ce type, rien n'est fait. Nous recherchons ensuite le brin marqué  $m$  ( $m \neq d$ ) sur l'orbite  $\langle \alpha_2, \alpha_3 \rangle (d\alpha_3^p)$ . Si  $m$  est trouvé, nous lions  $p_{10}(d)$  à  $p_9(m)$  et  $p_8(d)$  à  $p_5(m)$ . Nous procédons de la même manière avec  $d'$ .

S'il n'y a pas de brin  $m$ , nous sommes dans le cas habituel illustré par la figure 3.17.e, où nous formons une 3-cellule plate entre  $p(d)$  et  $p(d')$ . La figure 3.17.f montre le cas de l'auto-intersection où les motifs s'ouvrent pour former un hexaèdre (voir les figures 3.17.a et b).

Nous fermons ensuite l'orbite  $\langle \alpha_0, \alpha_1 \rangle p_{10}(d)$  en 1-liant ensemble les deux brins 1-libres  $f$  et  $f'$ , et nous 2-lions  $f\alpha_{21}$  et  $f'\alpha_{21}$  ce qui ferme les 3-cellules plates et les hexaèdres. Après la  $n$ -liaison effectuée, nous devons enlever la  $n$ -cellule compressée. En dimension 3, nous enlevons les 3-cellules compressées et nous cousons en une corde les hexaèdres créés sur l'auto-intersection. Pour un segment libre  $d$  d'un tel hexaèdre, nous le cousons avec l'autre segment libre de  $\langle \alpha_2, \alpha_3 \rangle (d)$ . Pour traiter le cas d'une auto-intersection qui se touche (voir la figure 3.14.a) en dimension 2, nous définissons une opération qui réduit une 2-cellule lorsque deux nouvelles 2-cellules insérées sont 2-liées (voir la figure 3.15.b).

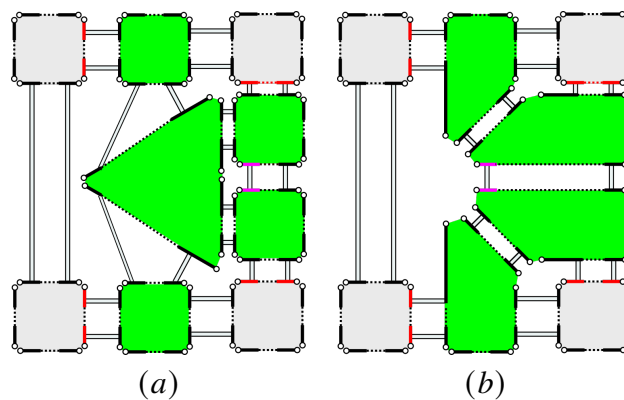


FIGURE 3.15 – Suppression d'une 2-cellule pour générer un motif auto-touchant.

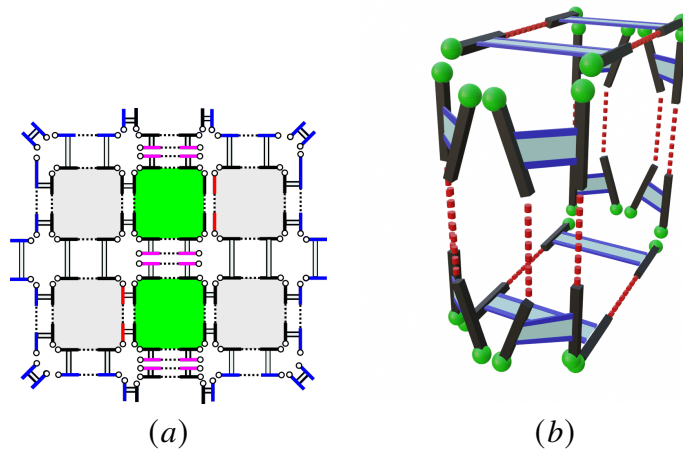


FIGURE 3.16 – (a) Simple insertion de deux 2-cellules (en vert) dont les brins en mauve forment trois 2-cellules compressées ; (b) 3-cellule compressée.

### 3.3 . Conclusion et perspectives

Le choix de la représentation des données est une étape essentielle dans le développement de tout logiciel manipulant et éditant des structures complexes telles que des modèles de CAO ou des maillages. Dans ce chapitre, nous avons considéré la gestion de structures de blocs hexaédriques pour la génération de maillages hexaédriques structurés par blocs à partir de modèles géométriques de type CAO représentés par leur bords. Pour représenter cette structure et définir des opérations d'édition de blocs, nous avons opté pour l'utilisation du modèle des cartes généralisées de dimension 2 et 3. L'utilisation de ce modèle nous apporte de nombreux avantages :

1. Tout d'abord la possibilité d'obtenir des définitions similaires en 2D et 3D pour nos opérations d'édition de feuillets. Par similaire, on entend que la structure des algorithmes est identique et que les différences sont principalement liées au composition d'involutions impliquées pour parcourir la structure de blocs.
2. La séparation claire des préoccupations entre la topologie et la géométrie. La gestion de la géométrie - plongement dans l'espace et classification sur le modèle géométrique - est assurée par un système de règles mis en oeuvre systématiquement lorsque les opérations de couture/découiture fusionnent ou séparent des cellules de la structure de blocs.
3. L'utilisation de cartes généralisées nous permet de disposer de pré- et post-conditions claires pour garantir la validité de la structure de blocs durant les opérations d'édition.
4. Enfin, la présence de la notion d'orbite, qui généralise les notions usuelles de cellules en maillage sera à l'avenir très utile pour, par exemple, attribuer efficacement des informations à des demi-faces de blocs.



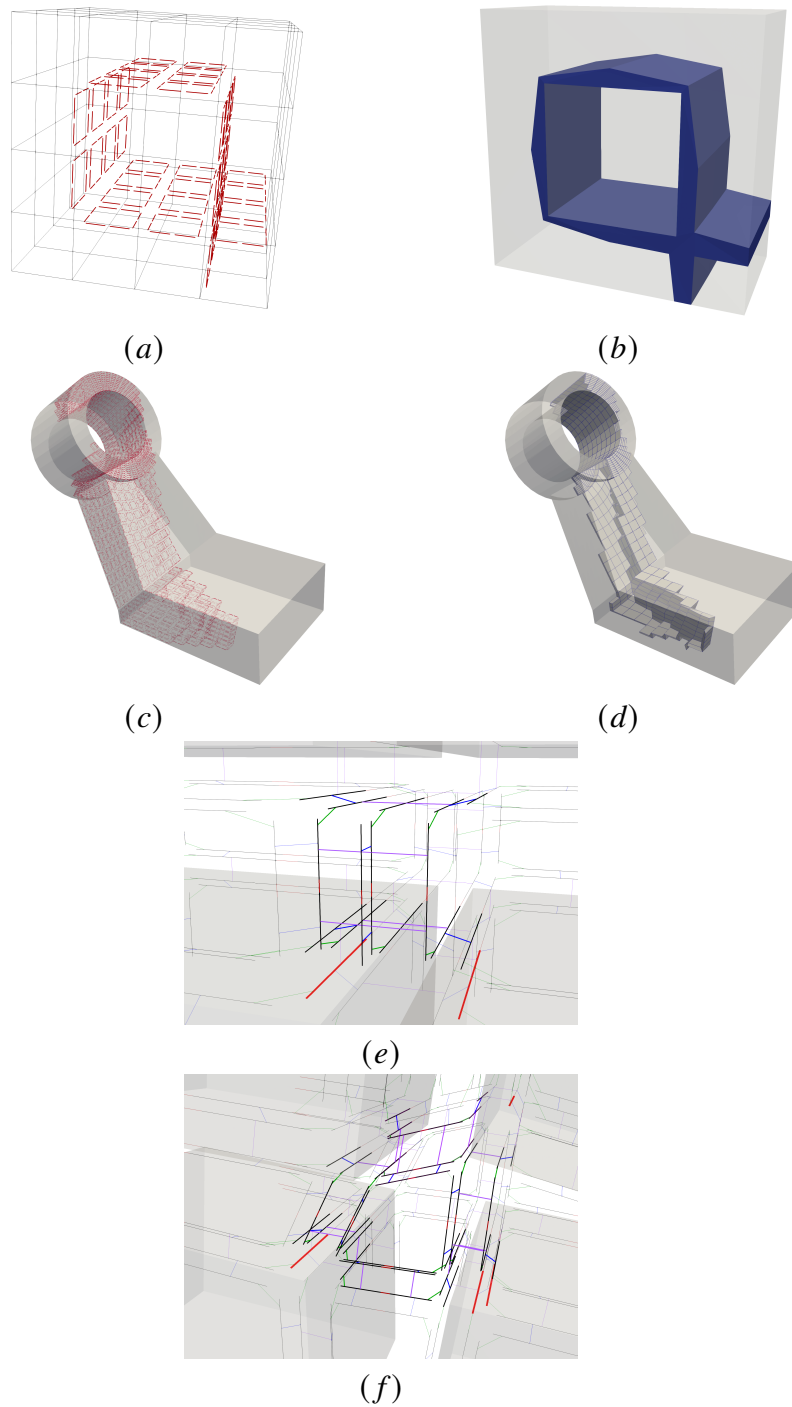


FIGURE 3.17 – Brins marqués d’une grille le long de laquelle un feuillet auto-intersecté sera inséré (a) et la grille résultante avec le feuillet surligné (b). En (c) un cas plus complexe avec (d) une coupe transversale pour montrer le feuillet inséré. Les deux motifs de 12 brins en sur-brillance engendrés par les deux brins rouges (voir la figure 3.12) forment une 3-cellule compressée en (e) (illustration du cas 2D dans la figure 3.16.a). En (f), où le feuillet s’auto-intersecte, les quatre motifs forment une cellule hexaédrique supplémentaire.

Les trois premiers points ont été très importants dans ce travail pour obtenir une implémentation propre des opérations sur les feuillets, en particulier l'insertion de feuillet. Nous sommes en mesure d'insérer des feuillets qui s'auto-intersectent et se touchent d'une manière robuste. L'implémentation a été réalisée en s'appuyant sur l'implémentation des cartes généralisées disponible au sein du projet CGAL [42, 43].

Ce travail d'utilisation des cartes généralisées pour la représentation de structures de blocs hexaédriques va se poursuivre à court et moyen terme dans deux directions complémentaires. Tout d'abord, la représentation géométrique de la structure va évoluer pour représenter des blocs courbes, en considérant entre autres une représentation des arêtes et faces par des BSplines de degré élevé (3 à 10). La dissociation topologie-géométrie nous permet d'isoler les traitements de la représentation courbe au sein des opérations de fusion et d'éclatement des cellules. En parallèle, cette représentation remplacera la représentation des blocs actuels présente au sein du logiciel **MGX**.

Un objectif à plus long terme serait de prouver formellement que nos opérations d'édition de structures de blocs produisent tout le temps des blocs purement hexaédriques. Aujourd'hui, nous pouvons juste indiquer que l'implémentation semble robuste à la vue des nombreux tests d'insertion réalisés (en particulier avec les algorithmes fournis au chapitre 5). Nous avons conscience que nos tests ne sont pas exhaustifs, en particulier car nous ne pouvons garantir avoir testé toutes les configurations représentatives de feuillets auto-intersectants ou auto-touchants. Nous ne pensons d'ailleurs pas qu'obtenir un ensemble exhaustif de tests soit atteignable. Dans l'optique de prouver formellement les propriétés souhaitées de notre structure de blocs, une voie envisagée est de formuler nos opérations d'édition au sein du logiciel Jerboa [44]) et de valider les propriétés attendues avant génération du code final.

Notons pour conclure ce chapitre que l'ensemble des algorithmes proposés aux chapitres 4 et 5 reposent sur l'implémentation des structures de blocs à l'aide des 2 et 3-G-cartes. Pour le cas 2D, nous avons ajouté des opérations non décrites ici mais utilisées pour les motifs de remplacement utilisés au chapitre 4.



## 4 - Optimisation de polycube 2D par application de motifs et d'un système multi-agent

Nous avons vu au chapitre 2 que les méthodes générant des maillages hexaédriques ou des structures de blocs hexaédriques de type Polycube sont relativement robustes, au sens où elles fournissent un résultat pour de nombreux domaines géométriques. En d'autres termes, elles échouent "rarement", mais par contre la structure obtenue n'est pas celle généralement souhaitée. En particulier, des contraintes d'alignement au bord ne sont pas respectées.

Dans ce chapitre, nous nous concentrons sur le cas 2D, et nous proposons une méthode pour optimiser les structures de blocs quadrilatéraux afin d'améliorer la qualité globale de la structure. Partant d'une structure de blocs classifiée sur un modèle géométrique, nous suivons une approche basée sur l'application de motifs locaux d'édition de la structure. Si appliquer localement un motif est "simple", il faut par contre assurer la cohérence globale de la structure. Pour cela, nous proposons une solution basée sur un système multi-agent, où un ensemble d'agents proposera d'appliquer des motifs locaux, et un agent coordinateur gèrera l'organisation globale pour obtenir un résultat cohérent.

Notons que l'ensemble des résultats proposés dans ce chapitre repose sur les choix de représentation de données effectués au chapitre 3 et sur l'implémentation associée.

### Publications et communications relatives à ce chapitre :

- Poster aux journées des doctorants du CEA DAM DIF, *Maillage structuré par blocs à l'aide d'un système multi-agent réflexe coordonné*, en juin 2022.
- Poster à la conférence SIAM IMR2023, *Block structured mesh using a system of coordinated reflex agents*, SIAM International Meshing Roundtable Workshop 2023, 6-9 mars, 2023.
- Communication orale aux journées des doctorants du CEA DAM DIF, *Représentation et optimisation de maillage structuré par blocs*, en juin 2023.
- Communication orale aux JFIG (Journées Françaises d'Informatique Graphique), *Maillage structuré par blocs à l'aide d'un système multi-agent réflexe coordonné*, en novembre 2022.
- Communication orale au *Symposium on Trends in Unstructured Mesh Generation* à Albuquerque, USA, en juillet 2023.

### 4.1 . Qu'est ce qu'un système multi-agent ?

Dans ce travail, nous considérons un système multi-agent tel que défini par J. FERBER [45]. Les Systèmes Multi-Agents, ou SMA, ont pour fondement que « *les activités simples ou complexes, telles que la résolution de problèmes, l'établissement*

*d'un diagnostic, la coordination d'action ou la construction de systèmes sont le fruit d'entités relativement autonomes et indépendantes, appelées agents, qui travaillent au sein de communautés selon des modes, parfois complexes, de coopération, de conflit et de concurrence, pour survivre et se perpétuer. De ces interactions émergent des structures organisées qui, en retour, contraignent et modifient les comportements de ces agents » [45]. Ainsi un agent dans un SMA est une entité physique - un robot - ou virtuelle - un logiciel - principalement définie par les caractéristiques suivantes proposées par J. FERBER dans [45] :*

1. il est capable d'agir dans un environnement,
2. il peut communiquer directement avec d'autres agents,
3. il est mu par un ensemble de tendances, définies sous la forme d'objectifs individuels, ou d'une fonction de satisfaction, voire de survie, qu'il cherche à optimiser,
4. il possède des ressources propres,
5. il est capable de percevoir (mais de manière limitée) son environnement,
6. il ne dispose que d'une représentation partielle de cet environnement (et éventuellement aucune),
7. il possède des compétences et offre des services,
8. il peut éventuellement se reproduire,
9. son comportement tend à satisfaire ses objectifs, en tenant compte des ressources et des compétences dont il dispose, et en fonction de sa perception, de ses représentations et des communications qu'il reçoit.

Pour FERBER, « *les recherches dans le domaine des systèmes multi-agents poursuivent deux objectifs majeurs : Le premier concerne l'analyse théorique et expérimentale des mécanismes d'auto-organisation qui ont lieu lorsque plusieurs entités autonomes interagissent. Le second s'intéresse à la réalisation d'artefacts distribués capables d'accomplir des tâches complexes par coopération et interaction.* »

Nos travaux se positionnent en considérant le second objectif : nous allons chercher à utiliser les SMAs d'une part pour modéliser les problèmes de manière informatique, et d'autre part, pour résoudre un problème par coordination d'un ensemble d'agents.

#### **4.2 . Description du problème de satisfaction de contraintes d'alignement en 2D**

Notre problème consiste à optimiser une structure de blocs obtenue à l'aide d'un algorithme de type Polycube. De telles structures sont illustrées en dimension 2 et 3 sur la figure 4.1.

Dans les deux cas, les cellules des structures de blocs sont classifiées sur leurs modèles géométriques respectifs. Sur la figure 4.1(a), chaque sommet de la structure

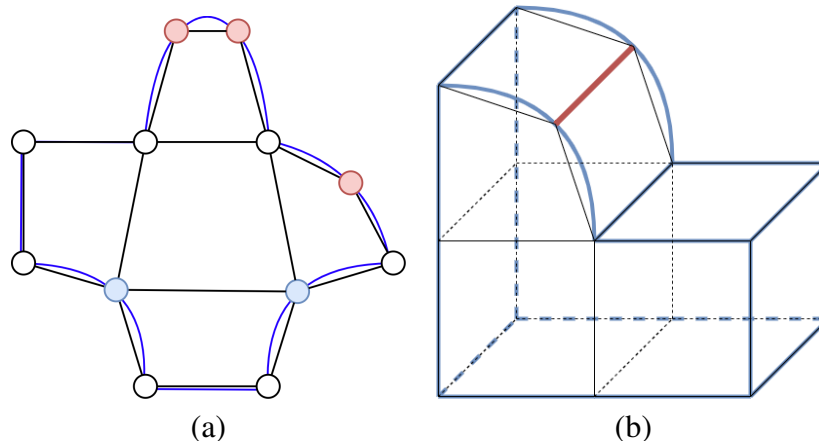


FIGURE 4.1 – Contraintes sur des structures de type Polycube. En (a), en dimension 2, les contraintes d’alignement sont portées par les sommets (en bleu et en rouge); En (b), les contraintes en dimension 3 sont portées au niveau des arêtes - ici l’arête rouge.

est coloré en blanc, bleu ou rouge. La couleur blanche indique que le sommet est *satisfait*, au sens où sa valence est en adéquation avec sa classification géométrique. Pour rappel, la valence d’un sommet en dimension 2, respectivement une arête en dimension 3, est le nombre de faces, respectivement de blocs, incidents à ce sommet, respectivement cette arête.

Pour revenir à notre exemple, contrairement à la couleur blanche, les couleurs bleues et rouges sont caractéristiques d’une configuration problématique. Ainsi un sommet de couleur rouge correspond à un sommet dont la valence devrait être augmentée pour satisfaire les conditions d’alignement au bord. En l’occurrence, il faudrait ici que chacun des sommets roses soit incident à deux blocs et non un seul. Au contraire, un sommet de couleur bleue correspond à un sommet dont la valence devrait être diminuée : chacun des sommets bleus devrait être ici incident à deux blocs et non trois. Ce critère de satisfaction se retrouve en dimension 3 au niveau des arêtes de la structure de blocs. Sur la figure 4.1(b), l’arête en rouge devrait être incidente à deux blocs et non un seul.

Nous nous concentrons dans la suite de ce chapitre uniquement sur le cas 2D<sup>1</sup>, et notre objectif est de modifier la structure de blocs afin de n’obtenir à la fin que des sommets de couleur blanche, c’est-à-dire que des configurations locales satisfaisantes. Pour cela, nous allons successivement appliquer des séries de motifs à l’aide d’agents ayant une vue locale à chaque face.

### 4.3 . Pourquoi utiliser un système multi-agent ?

1. Nous évoquerons la possibilité d’étendre les travaux proposés au cas 3D en conclusion de ce chapitre.

Notre postulat de départ est qu'un ingénieur « sait » comment modifier la structure polycube pour des cas simples comme ceux de la figure 4.2. Ici, pour chaque exemple, un ingénieur habitué à réaliser des maillages structurés par blocs identifie aisément comment insérer une couche de mailles dans la structure de blocs pour satisfaire tous les sommets problématiques. Par contre, il est beaucoup plus dur de déterminer comment insérer un ou plusieurs feuilletés dans une structure qui comporterait des dizaines, voire des centaines de faces, et qui aurait un grand nombre de sommets insatisfaits.

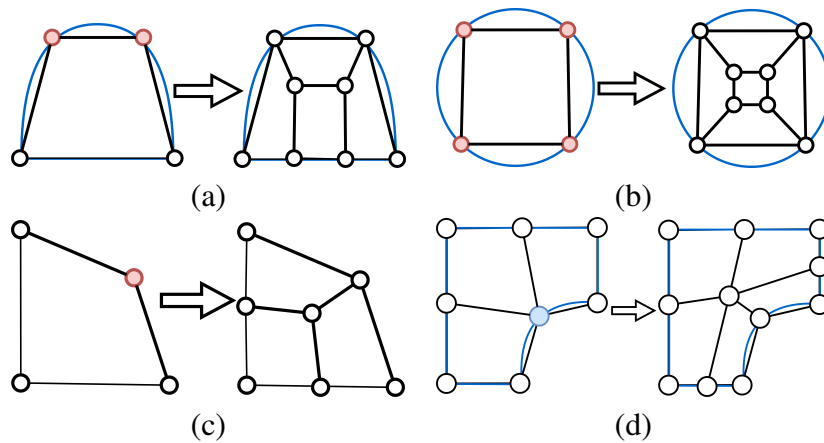


FIGURE 4.2 – Exemple de polycubes 2D qu'un ingénieur sait améliorer par l'insertion d'une couche de mailles. En (a), une couche de mailles composée de trois quadrilatères est ajoutée le long d'une partie du bord pour satisfaire les deux sommets roses ; En (b), ajouter une couche qui entoure l'unique bloc de la structure satisfait les quatre sommets roses ; En (c), seuls deux quadrilatères sont ajoutés pour satisfaire le sommet rose ; Enfin, en (d), le sommet bleu est lui aussi satisfait après l'insertion de deux quadrilatères.

En considérant notre postulat initial de manière extrême, nous posons l'axiome suivant : *un ingénieur est capable de modifier localement un bloc de la structure*. Dès lors, pour une structure de blocs composée de  $n$  blocs, nous pouvons demander à  $n$  ingénieurs de chacun travailler indépendamment sur un bloc de la structure. Par exemple, considérons le cas de la figure 4.3, où le domaine géométrique représenté en (a) peut être discrétisé par une structure composée de 5 blocs (voir 4.3.b et 4.3.c). Dès lors, si on assigne chacun de ces 5 blocs à un ingénieur, deux diront qu'ils n'ont rien à faire car deux blocs sont localement "satisfaisants", et trois voudront effectuer une action pour améliorer la qualité du bloc qui leur a été confié. Le bloc du bas et le bloc à gauche sont satisfaits car ils ne sont pas placés de manière à pouvoir résoudre les contraintes en bleu. Sur la figure 4.4, on peut voir trois actions possibles proposées indépendamment par chaque ingénieur :

- L'action 1 permet de satisfaire les deux sommets rouges en augmentant leurs valences respectives. Par contre, l'opération étant locale, deux sommets non

conformes - les carrés rouges - seront introduits le long de l'une des arêtes du blocs.

- L'action 2 permet de satisfaire un sommet bleu sur les deux en diminuant sa valence. On introduit au passage un sommet non conforme là aussi <sup>2</sup>.
- L'action 3 permet de satisfaire un sommet rouge en augmentant sa valence mais introduit deux sommets non conformes sur deux arêtes différentes du bloc initial.

Si ces trois actions peuvent être appliquées indépendamment et qu'elles permettent pour chacun des ingénieurs de fournir une solution à son problème, la structure obtenue sur la droite de la figure 4.4 n'est pas cohérente : des blocs ont des arêtes coupées, des blocs ont l'air d'être triangulaires <sup>3</sup>. Il est nécessaire de coordonner les agents pour qu'ils fournissent une solution globalement viable. C'est ce que nous proposons de faire en considérant l'ajout d'un agent coordinateur.

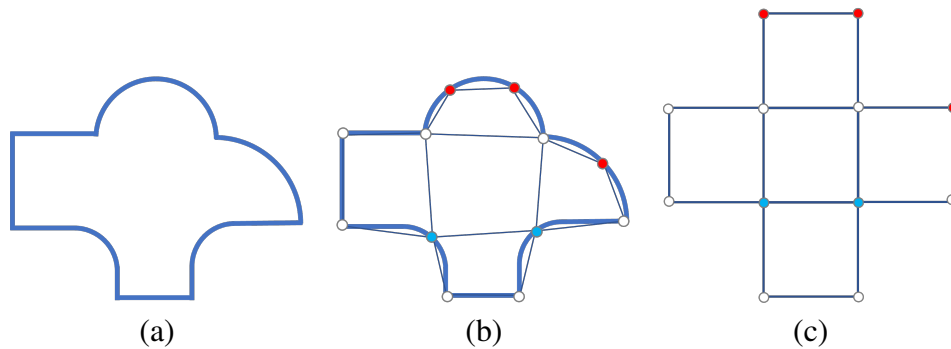


FIGURE 4.3 – Exemple de domaine géométrique en (a) et de structures de blocs associées représentée projetée sur le bord du domaine géométrique en (b), et représentée de manière cartésienne en (c).

#### 4.4 . Définition de notre système multi-agent

Les données d'entrée de notre problème sont un domaine géométrique  $\Omega$  et une structure de blocs quadrangulaires associée  $B_H$  (voir figures 4.5.a, 4.5.b et 4.5.c). Pour chaque sommet de cette structure, nous considérons deux indicateurs :

1. La **valence**  $v(s)$ , qui est le nombre de blocs incidents à  $s$  ;
2. La **criticité**, qui indique une insuffisance ou un excès de blocs incidents afin de satisfaire une contrainte d'alignement au bord.

Une criticité positive  $c^+(s)$  ● signifie qu'une augmentation de valence est nécessaire alors qu'une criticité négative  $c^-(s)$  ● indique le besoin d'une diminution. Notons que

2. Ce cas de motif est très particulier et peu intuitif à ce stade du chapitre. Nous reviendrons ultérieurement sur sa description.

3. Ce n'est qu'une impression qui sera expliquée par la suite.



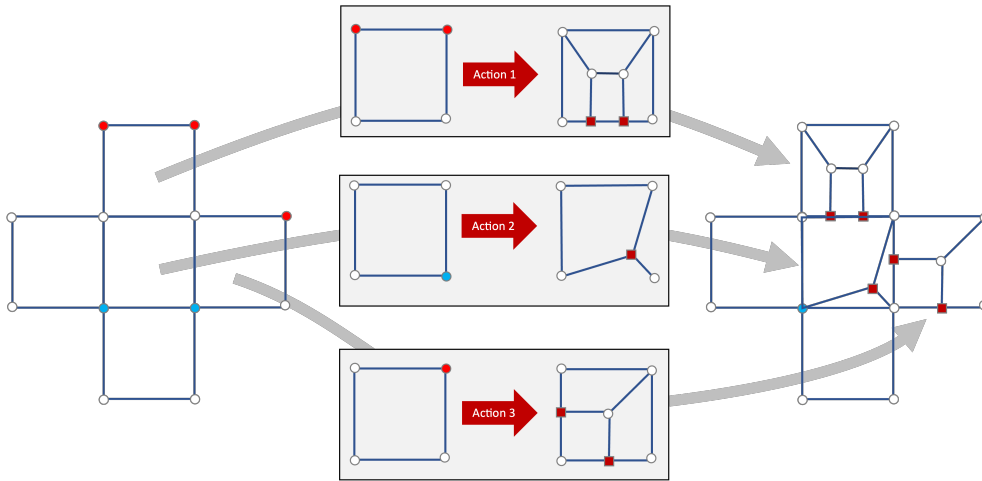


FIGURE 4.4 – Trois actions appliquées indépendamment sur des blocs de la structure.

durant le processus de modification de la structure, nous réalisons des opérations locales qui introduisent des sommets non conformes (■). Nous prenons pour hypothèse que les sommets critiques sont toujours placés au bord du domaine. Cette hypothèse est réaliste, la criticité étant dans notre cas liée à l'inadéquation de la valence d'un sommet avec la courbure locale<sup>4</sup> de la géométrie au bord.

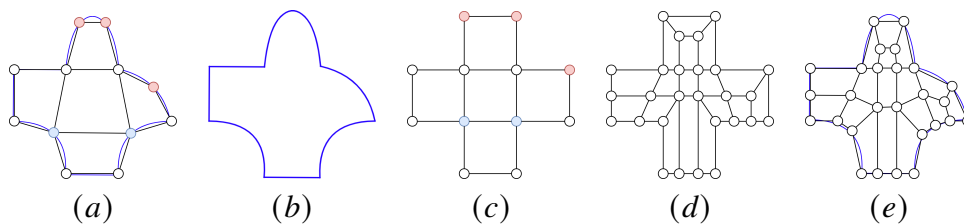


FIGURE 4.5 – Pipeline d'amélioration d'une structure de blocs. En (a), une structure de blocs associée à un domaine géométrique ; en (b), le domaine géométrique seul et en (c), la structure seule représentée sous forme cartésienne ; En (d) la structure cartésienne après modification, et, en (e), la structure finale associée au domaine.

Nous modélisons notre problème à résoudre par le système  $(E, AL_i, AC)$  où :

- $E$  est l'environnement constitué de la structure de bloc  $B_H$  et d'un domaine géométrique  $\Omega$ .
- $AL_i$  est un ensemble d'**agents réflexes locaux**, avec  $0 \leq i \leq N$ , où chaque agent est assigné à un bloc de  $B_H$ . La valeur  $N$  varie à chaque itération.
- $AC$  est un **agent coordinateur** responsable de la collaboration des agents  $AL_i$ .

De manière traditionnelle, le système  $(E, AL_i, AC)$  résout le problème qui lui est confié de manière itérative jusqu'à obtenir une solution satisfaisante. A chaque itération, les agents réflexes **observent** localement le problème qu'ils doivent résoudre

4. voir chapitre 2, section 2.4.2, page 24.

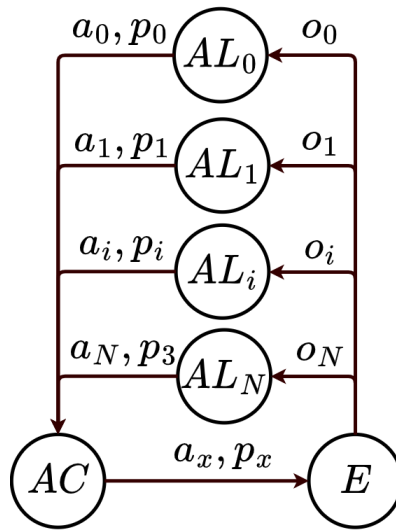


FIGURE 4.6 – Schéma général du système multi-agent utilisé pour optimiser une structure de blocs en dimension 2.

et proposent une **action** à effectuer à l'agent coordinateur. Ce dernier va **sélectionner** les actions à réaliser et l'environnement sera modifié en conséquence. Ce processus est répété jusqu'à la résolution de tous les problèmes locaux. Dans la suite de cette section, nous donnons des éléments de détail sur la façon dont nous gérons les actions, les observations, les agents locaux et l'agent coordinateur. Le schéma figure 4.6 illustre les relations entre les agents et l'environnement.

#### 4.4.1 . Définition des actions

Dans le système que nous proposons, chaque agent local propose une action à effectuer. Le choix de l'action se fait de manière *réflexe*, c'est-à-dire que selon l'observation locale qu'un agent fait du bloc  $B_i$  qui lui est affecté, il va proposer systématiquement la même opération à appliquer. Une opération va consister à remplacer le bloc  $B_i$  par un motif de blocs qui fera évoluer la valence des sommets de  $B_i$ . Potentiellement, des sommets non-conformes émergeront au bord de  $B_i$ . Des exemples d'actions sont présentés à la figure 4.7. Ces actions modifient la topologie d'un bloc de la manière la plus locale possible. Il existe trois types d'actions :

1. Les **actions critiques**, qui ont pour objectif de satisfaire des sommets insatisfaits à cause de la contrainte d'alignement au bord. Ces opérations sont donc appliquées sur des blocs ayant un ou plusieurs sommets positionnés sur le bord du domaine. Ces actions sont illustrées sur la figure 4.7.
2. Les **actions de propagation simple** s'appliquent à des blocs ayant un ou plusieurs sommets non-conformes le long de leurs arêtes. Elles découpent le bloc en propageant les sommets non-conformes vers l'arête opposée (voir les figures 4.8, et 4.9, 4.10 et 4.11).

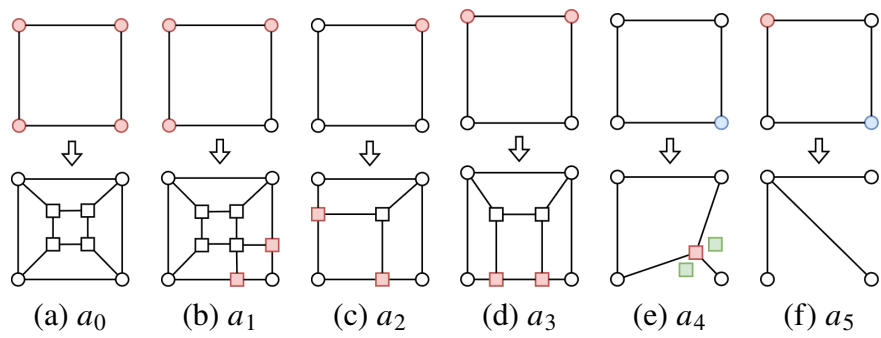


FIGURE 4.7 – Actions critiques. Les blocs sont modifiés de façon à satisfaire les sommets critiques.

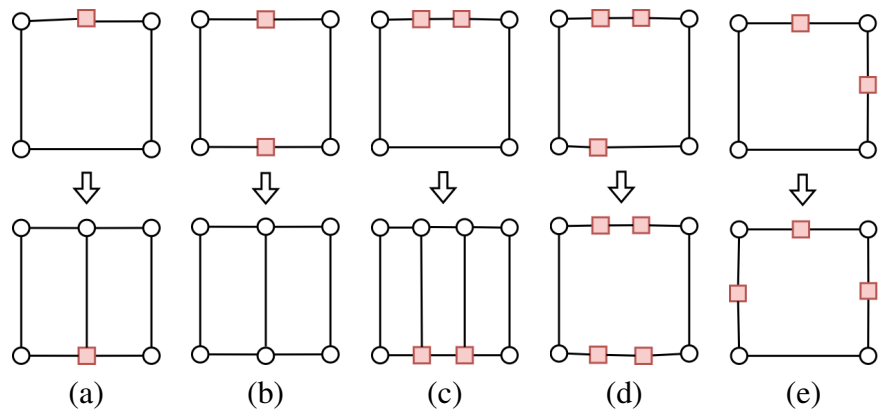


FIGURE 4.8 – Actions de propagations simples (1/4).

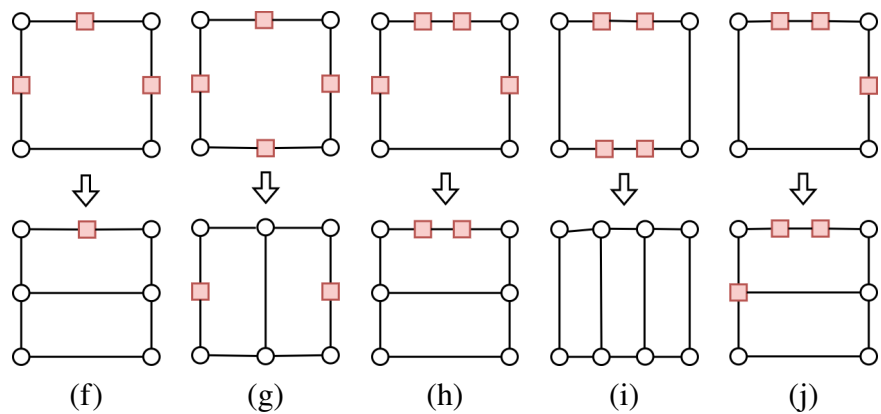


FIGURE 4.9 – Actions de propagations simples (2/4).

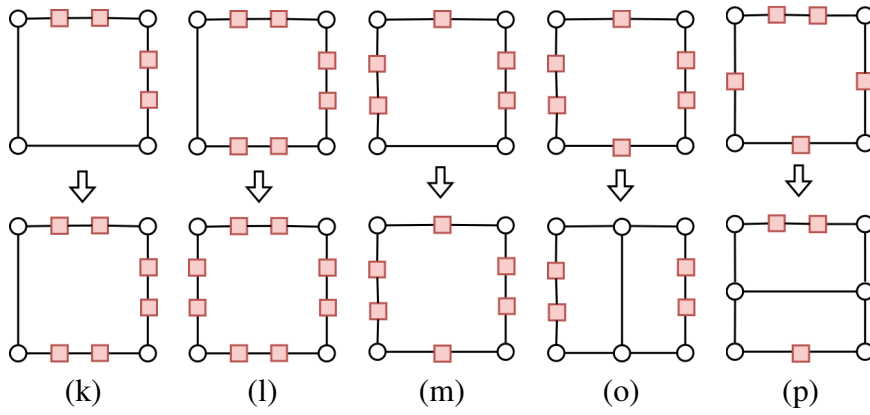


FIGURE 4.10 – Actions de propagations simples (3/4).

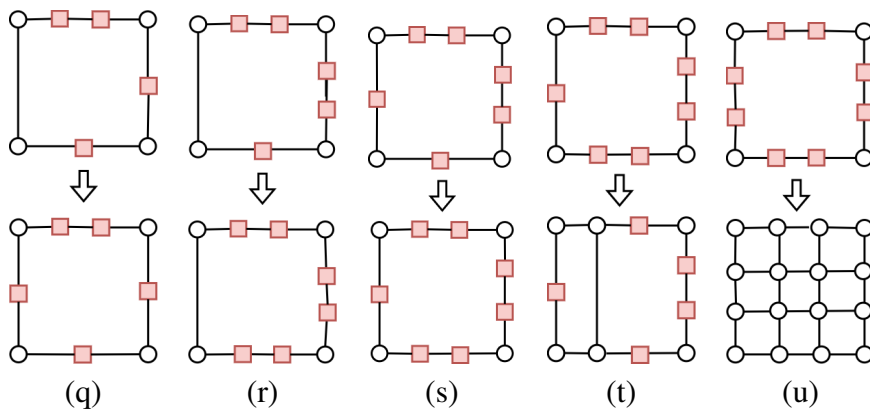


FIGURE 4.11 – Actions de propagations simples (4/4).

3. Les **actions de propagation critique** ont pour objectif de traiter des blocs disposant à la fois de sommets insatisfaits et de sommets non-conformes à propager. La figure 4.12 fournit les configurations que nous traitons.

Notons que nos actions n'introduisent jamais plus de deux sommets non-conformes le long d'une arête d'un bloc. En conséquence, les actions de propagation simple et de propagation critique considèrent qu'il n'est pas possible d'avoir plus de deux sommets non-conformes le long d'une arête d'un bloc observé.

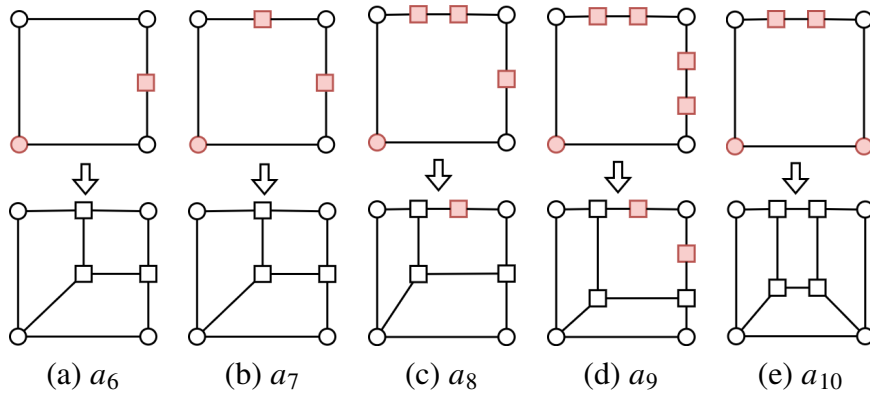


FIGURE 4.12 – Actions de propagations critiques. Les blocs sur lesquels sont appliquées ces actions ont des problèmes à la fois de criticité et de non-conformité.

### Cas particuliers des actions $a_4$ et $a_5$

L'action critique  $a_4$  a un comportement particulier. Si elle est appliquée, elle marque les faces voisines d'un sommet non conforme introduit. On utilise pour cela un carré vert (■) sur nos schémas. On peut voir que sur la figure 4.7.e, deux carrés verts sont présents. Ceci indique que le sommet associé est non conforme pour chacune des faces de part et d'autre. Ces faces marquées deviennent prioritaires à traiter. L'action  $a_4$  peut être appliquée plusieurs fois sur une même face si elle contient plusieurs sommets critiques de type  $c^-$ . Une exécution montrant les étapes de l'application de l'action  $a_4$  est illustrée sur la figure 4.13. La face  $f_2$  dispose de deux sommets de type  $c^-$ . Ces sommets sont les sommets étiquetés 0 et 1 en bleu. Un de ces sommets de type  $c^-$  est traité par appel à l'action  $a_4$ . Ici il s'agit du sommet étiqueté 1. La première étape (b) est d'insérer le sommet étiqueté 4 sur l'arête (1, 0) et le sommet étiqueté 5 sur l'arête (1, 2). Ensuite, en (c), nous créons un nouveau sommet, étiqueté 6, qui prend les connectivités des sommets 4 et 5, en (d). Le sommet 2 n'est pas de type  $c^-$ , il nécessite alors un traitement supplémentaire. Il s'agit simplement d'ajouter un nouveau sommet, étiqueté 7, et l'arête (6, 7) puis de supprimer l'arête (6, 2). A ce moment, le sommet étiqueté 6 est marqué comme étant non conforme et porte la prio-marque ■ pour les faces  $f_3$  et  $f_4$ . Enfin, le sommet étiqueté 2 est mis à jour comme étant non-conforme.

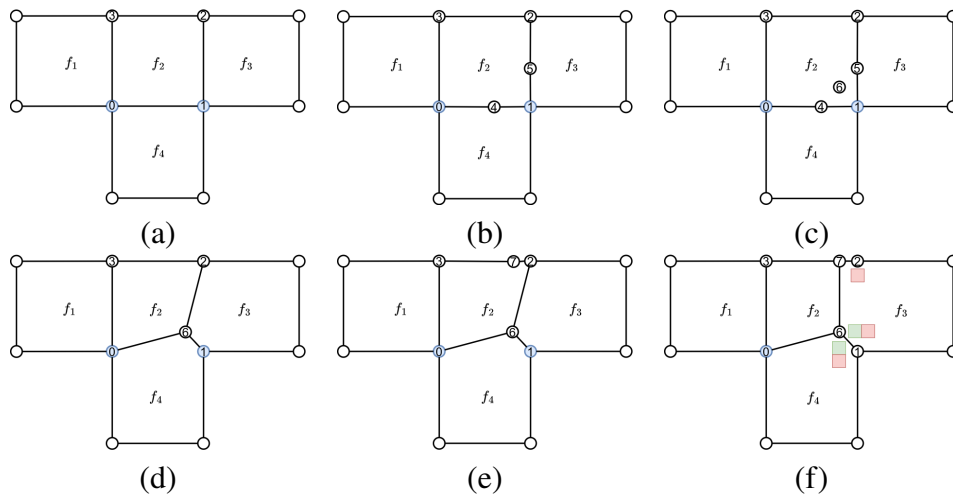


FIGURE 4.13 – Illustration pas à pas de l'exécution de l'action  $a_4$  sur la face  $f_2$ .

L'action critique  $a_5$  pour sa part supprime totalement le bloc sur lequel elle s'applique pour résoudre deux problèmes de satisfaction sur des sommets opposés dans le bloc. La figure 4.14 illustre son application en montrant l'évolution des blocs voisins aussi.

### Quelques éléments d'implémentation des actions

Nous avons globalement implémenté les actions de trois façons. Les actions telles que les actions  $a_0$ ,  $a_2$  et  $a_3$  sont basées sur l'insertion de couches telles que présentées au chapitre 3. En sélectionnant le chemin d'arêtes adéquat, on insère les faces quadrilatérales qui permettent de résoudre les problèmes de criticité rencontrés. Pour certaines actions, comme les actions  $a_1$ ,  $a_4$  et  $a_5$ , on remplace le bloc par un motif pour produire la topologie escomptée. Enfin, d'autres actions utilisent des fonctions de modification de plus haut niveau sans considérer la structure de carte généralisée sous-jacente. Les fonctions de modification en question sont l'insertion d'un sommet sur une arête et l'insertion d'une arête dans une face à partir de deux sommets de cette face.

#### 4.4.2. Définition des observations

L'observation est la perception que se fait un agent réflexe  $AL_i$  de l'environnement qui l'entoure. Dans notre modèle, nous considérons qu'un agent perçoit uniquement son bloc  $B_i$  : il observe la configuration de sommets critiques et non-conformes de  $B_i$ . Durant tout le processus, un bloc reste quadrangulaire. Chacun de ses 4 sommets peut être satisfait ou critique (● ou ●) et chaque côté du bloc peut être composé d'une ou plusieurs arêtes si ce côté a été subdivisé par la découpe d'un bloc adjacent lors d'une itération précédente. On dispose alors d'un sommet non-conforme (◻). Sur la figure 4.15, un sommet est critique et deux sommets non-conformes découpent deux

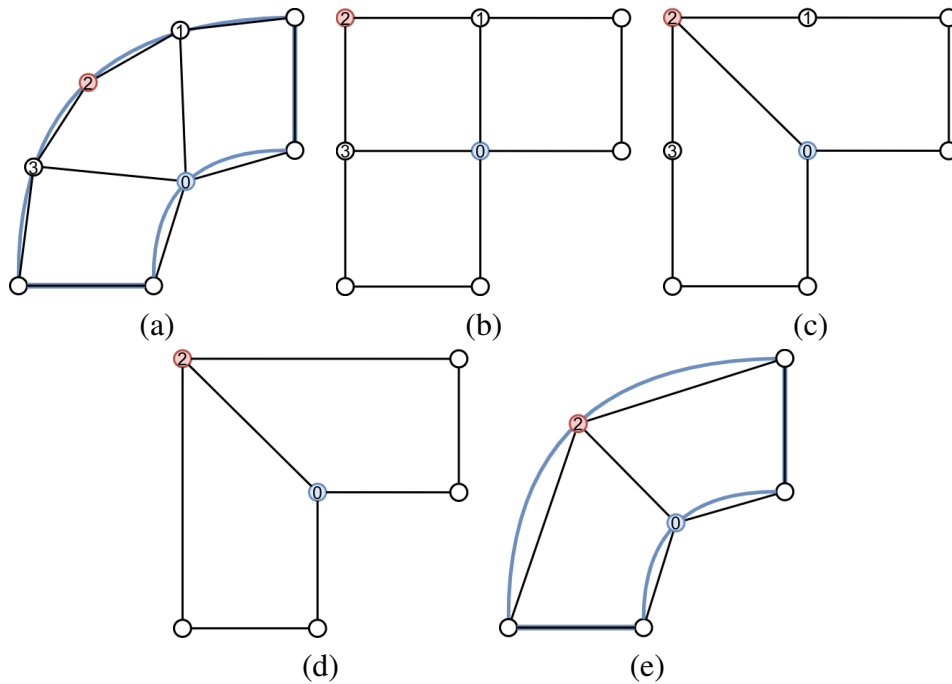


FIGURE 4.14 – Détails des étapes de l'action  $a_5$ . (a) Structure de blocs associée à la géométrie (b) Structure de bloc en représentation cartésienne (c) Suppression des deux arêtes (0, 1) et (0, 3) puis insertion de l'arête (0, 2). (d) Suppression des sommets 3 et 1. Le sommet 2 a bien augmenté sa valence de 1 vers 2 et le sommet 0 a diminué la sienne de 3 vers 2.

côtés du blocs. Un sommet peut aussi être conforme pour un bloc mais non-conforme pour un autre.

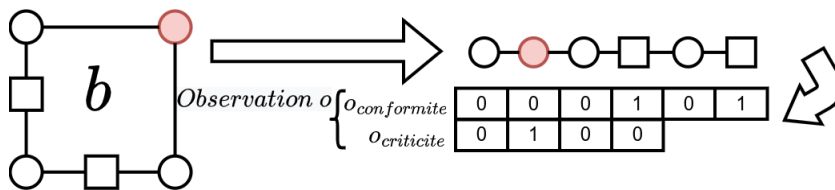


FIGURE 4.15 – Observation d'un bloc en considérant les caractéristiques de criticités et de conformité de ses sommets.

Pour modéliser l'observation d'un agent, nous encodons cette observation comme étant composé de deux parties une binaire et une ternaire :

- La partie  $o_{conformite}$  est la première partie, elle a pour longueur le nombre total de sommets locaux au bloc. Un bit vaut 0 si le sommet correspondant est conforme et 1 sinon ;
- La partie  $o_{criticite}$  est la seconde partie, elle est de taille 4 et indique pour chaque coin du bloc s'il est régulier (0) ou critique (-1 ou +1 selon le type de

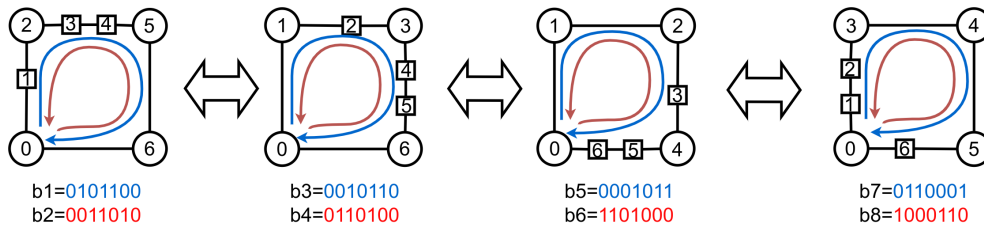


FIGURE 4.16 – Représentations possibles pour une même face. En partant d’un sommet initial, le parcours des sommets de la face encode le mot binaire. On ne peut pas présager du sens de parcours de la face, qui peut être effectué dans le sens horaire (bleu) ou anti-horaire (rouge).

criticité).

Une criticité  $c^-$  sur un sommet  $s$  est observée dans une face par un agent local seulement si deux arêtes incidentes à  $s$  ne sont pas au bord. Ceci est représenté sur la figure 4.17.a où le sommet 0 est critique  $c^-$  et à bien deux arêtes incidentes en rouges pour la face  $f_1$ . Ce qui n’est pas le cas pour les faces  $f_2$  et  $f_3$ .

Les observations de conformité sont obtenues en choisissant un sommet et en parcourant les sommets d’une face. Un sommet conforme est encodé en binaire par un 0 alors qu’un sommet non-conforme est encodé par un 1. Sur la figure 4.16 pour une même face, nous obtenons huit représentations binaires avec les rotations et symétries. Nous avons choisi de considérer des classes d’équivalence à rotation et symétrie près. Ainsi nous définissons un représentant par le minimum des valeurs binaires. Par exemple en restant sur la figure b5 est le représentant de b1, b2, b3, b4, b6, b7 et b8 car sa valeur en binaire est la plus petite. En considérant qu’il y a au plus deux sommets non-conformes sur chaque arête d’un bloc, nous avons manuellement listé les 21 représentants de conformités possibles. Le tableau 4.1 liste tous les représentants en binaire. Ces représentants nous ont permis de générer les autres représentants mais aussi la manière de s’y rapporter par des décalages et inversions. Cela nous permet en plus d’obtenir une indexation identique pour effectuer nos actions.

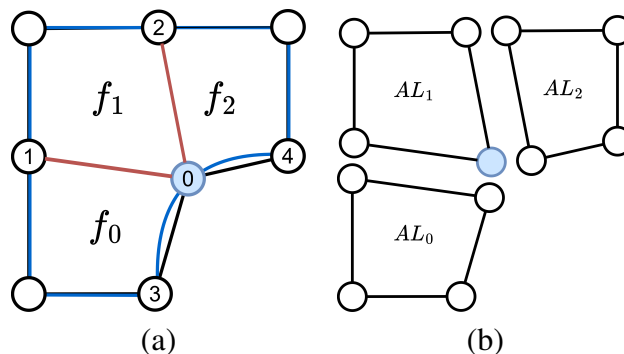


FIGURE 4.17 – L’agent placé sur la face  $f_1$  observe la criticité  $c^-$  en bleu sur la face  $f_1$  mais ce n’est pas le cas pour les autres agents  $AL_2$  et  $AL_0$ .



Identifiant	observation criticité	Identifiant	observation criticité
1	0000	12	00011011
2	00001	13	00101101
3	000011	14	001101011
4	000101	15	010101011
5	001001	16	001011011
6	0001011	17	0101101011
7	0010011	18	0011011011
8	0010101	19	0101011011
9	01010101	20	01011011011
10	00110011	21	011011011011
11	00101011		

TABLE 4.1 – Tableau récapitulatif de la représentation unique de la conformité d’une face.

#### 4.4.3 . Politique des agents réflexes locaux

Les agents locaux aux blocs proposent de manière déterministe un motif basé sur l’observation d’une face. Chaque agent observe le bloc qui lui est assigné et détermine, comme on a pu le voir à la section précédente, à quelle classe d’équivalence son bloc appartient. En fonction de cela, il sait si une face contient des sommets critiques et/ou non-conformes et applique l’action correspondant à la face : soit une action critique, soit une action de propagation simple, soit une action de propagation critique. Aucune action n’est proposée et la priorité est nulle si l’observation ne comporte aucune non-conformité ou criticité. Le diagramme de la figure 4.18 résume ces choix.

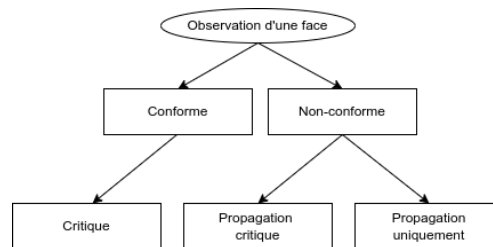


FIGURE 4.18 – Schéma de la politique d’un agent local réflexe AL.

#### 4.4.4 . Agent coordinateur

A chaque itération, tous les agents réflexes locaux choisissent l’action qu’ils veulent appliquer, puis l’agent coordinateur AC sélectionnera une action parmi celles proposées par les agents réflexes. Ce choix est fondé sur un indice de priorité calculé pour chaque action.

Le choix est réalisé de manière déterministe. L’agent coordinateur AC sélectionne

l'action de priorité la plus haute et ordonne à l'agent correspondant de l'exécuter. Si deux actions ont la même priorité, alors on sélectionne toujours l'agent agissant sur le bloc d'indice le plus faible <sup>5</sup> Le calcul de la priorité  $p_i$  d'une action tient compte à la fois de l'évolution de la criticité et de la conformité. On a

$$p_i = \alpha x^- + \beta m + \gamma x^+ + \delta nc,$$

avec  $x^-$  et  $x^+$  les nombres de sommets critiques négatifs et positifs respectivement,  $nc$  le nombre de sommets non-conformes et  $m$  le nombre de sommets marqués par l'action spécifique  $a_3$  (voir Fig. 4.18.a). On pose les coefficients  $\alpha = 100, \beta = 50, \gamma = 10, \delta = 1$  afin d'éliminer en premier lieu les sommets critiques négatifs ( $c^-$ ) puis de propager certaines non-conformités induites ( $m$ ), avant d'éliminer les sommets critiques positifs ( $c^+$ ) et enfin l'ensemble des non-conformités restantes ( $nc$ ).

#### 4.5 . Algorithme global et exécution pas à pas

Le processus général de l'exécution de notre système est décrit par l'algorithme 4. Si ce processus termine sur l'ensemble des expérimentations que nous avons menées (plus d'une centaine), nous n'avons pas réalisé de preuve de terminaison associée.

---

**Algorithme 4 :** Algorithme général du système multi-agent 2D co-ordonné.

---

**Données :**

```

1 Répéter
2   Pour chaque agent  $a \in [1; AL]$  proposer un motif tel que faire
3      $a_i \leftarrow \text{reflexe}(o_i)$  ;
4   fin
5   Pour chaque agent  $a \in [1; AL]$  faire
6      $\text{calculerPriorite}$  ;
7   fin
8    $p_i \leftarrow \text{obtenirPriorite}(AL_i)$  ;
9    $\text{trier\_selon\_priorite\_max}(p_i, a_i)$  ;
10   $i, a_x \leftarrow AC.\text{selectionner}(p_i)$  ;
11   $AL_i.\text{executer}(a_x)$  ;
12 jusqu'à Tant que toutes les contraintes ne sont pas satisfaites;

```

---

Afin d'illustrer l'application de cet algorithme, considérons la figure 4.19 sur laquelle une exécution pas à pas est illustrée. La géométrie initiale est celle la figure 4.5.b. Chaque représentation de la structure est un instantané pris après le tri de la ligne 9 de l'algorithme 4. Pour chaque itération, nous indiquons l'ordre et la priorité associées à chaque face de la structure. Comme indiqué précédemment, à chaque

---

5. Toutes les cellules (sommets, arêtes, faces) de la structure disposent d'un identifiant unique qui les caractérise.

itération, les agents réflexes se voient assigner une face à observer et proposent une action. Ensuite chaque agent calcule sa priorité et l'agent coordinateur effectue un tri, sélectionne l'agent ayant la priorité la plus haute et lui demande d'exécuter son action. Ce processus itératif se poursuit tant qu'il reste des contraintes de criticité ou de conformité à résoudre. Regardons maintenant le déroulement de ce processus plus précisément :

- A l'étape 0, la face  $f_2$  dispose de deux sommets  $x^-$ , ce qui la place en première position devant  $f_0$ , qui a deux sommets  $x^+$ , puis  $f_3$  qui n'en a qu'un. Les faces  $f_1$  et  $f_4$  n'ont pas de sommet critique ni de sommet non-conforme, leur priorité est donc de zéro. L'agent coordinateur va par conséquent effectuer l'action  $a_4$  sur la face  $f_2$ . Cette action va augmenter les priorités des faces incidentes  $f_1$  et  $f_4$  de  $\beta$ .
- L'agent coordinateur sélectionne de nouveau  $f_2$  avec la même action que précédemment.
- La face  $f_3$  augmente aussi de  $\beta$  et dispose aussi d'un sommet  $x^+$ . Cette face sera donc sélectionnée par le coordinateur avec une action de propagation sur le sommet marqué en vert.
- Les faces  $f_1$  et  $f_4$  ayant la même priorité, nous choisissons arbitrairement  $f_1$ .
- La face  $f_4$  a la priorité la plus élevée et se voit appliquer l'action de propagation. Il ne reste à ce moment plus aucun sommet  $x^-$  ni de marque  $m$ .
- Les faces  $f_0$  et  $f_3$  qui comportent des sommets  $x^+$  sont ensuite traitées durant les étapes 5 puis 6.
- La face  $f_2$  ayant le plus de sommets  $nc$ , elle est sélectionnée en priorité (étape 7). Puis, la face  $f_1$  sera traitée (étape 8).
- Finalement à l'étape 9, il ne reste plus qu'une seule face  $f_0$  qui sera séparée en deux par l'ajout d'une arête.
- A l'étape 10, il ne reste plus de contrainte à résoudre, l'algorithme s'achève.

#### 4.6 . Validation expérimentale du système multi-agent

Le système multi-agents mis en place dans ce chapitre a été testé sur de nombreuses structures de blocs de type polycubes. Nous illustrons ici quelques-uns des résultats obtenus. Sur chacune des figures, le code couleur des sommets sur la représentation cartésienne des blocs est le suivant :

- Un sommet vert est un sommet satisfait. Aucune modification topologique n'est nécessaire dans son voisinage ;
- Les sommets rouges sont des sommets critiques nécessitant l'augmentation de leur valence ;
- Les sommets bleus sont des sommets critiques nécessitant la diminution de leur valence.

Dans tous les cas, notre système multi-agent produit des structures de blocs qui répondent aux contraintes de criticité exprimées.

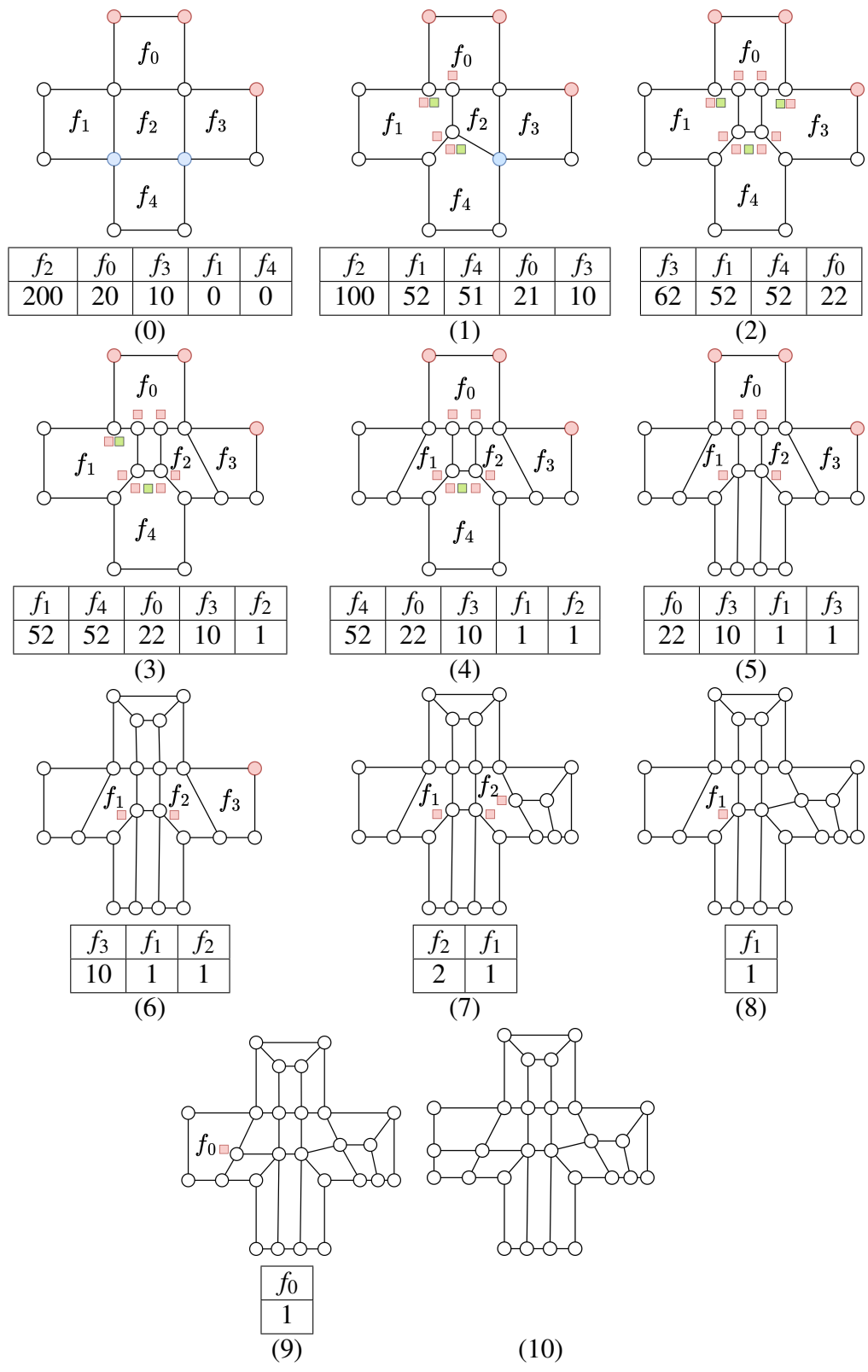


FIGURE 4.19 – Exemple de déroulement pas à pas de l’algorithme 4. Au fur et à mesure les contraintes de criticité sont résolues, puis l’entièreté des contraintes de conformité.

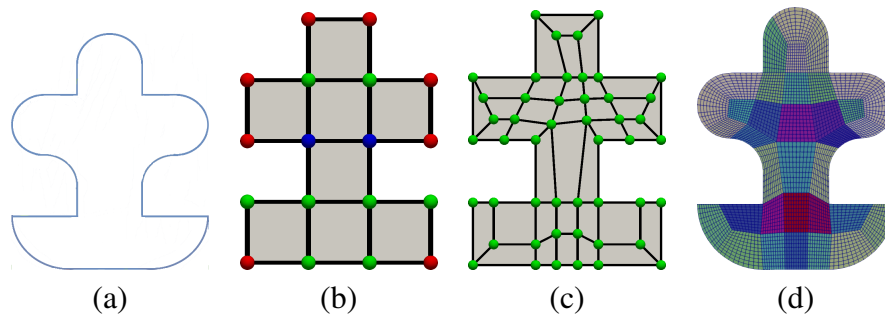


FIGURE 4.20 – Utilisation de notre système multi-agent pour optimiser une structure de blocs et obtenir le maillage final. En (a), la géométrie initiale ; en (b), la représentation cartésienne du polycube associé ; en (c), le polycube optimisé par notre approche ; en (d), le maillage final obtenu.

La figure 4.20 illustre le pipeline global pour mener nos expérimentations. En 4.20.a, le modèle géométrique que l'on veut discrétiser possède des formes courbes, des segments et des sommets. En 4.20.b, une structure polycube est construite à partir de cette géométrie. Pour chaque sommet au bord, on calcule sa criticité. Ensuite l'application du système multi-agent décrit précédemment nous permet de modifier la structure pour obtenir le résultat présenté en 4.20.c. Enfin sur la figure 4.20.d, chacun des blocs est discrétisé par une grille 10x10 pour obtenir le maillage final après projection et lissage. Dans la suite, nos résultats fournissent uniquement la structure améliorée en représentation cartésienne.

Les figures 4.21 et 4.22 fournissent respectivement des résultats sur des figures simples et des figures plus complexes. Dans le premier cas, le même polycube obtenu pour des géométries différentes est optimisé différemment car les contraintes de criticité (liées à la géométrie) impose des modifications différentes. Dans le second cas, sur la figure 4.22, on constate que notre approche fournit des résultats cohérents pour des géométries plus complexes disposant entre autres de trous.

## 4.7 . Conclusion et perspectives

L'approche multi-agent que nous avons proposée dans ce chapitre reposait sur deux concepts principaux :

1. Tout d'abord, nous partions du postulat qu'un ingénieur savait modifier localement une structure de blocs pour satisfaire des contraintes que nous avons modélisées sous la notion de sommets critiques.
2. Ensuite, nous considérons qu'un ensemble d'agents pouvaient conjointement appliquer un ensemble de règles de transformations locales, i.e. nos actions, qui aboutirait à la découverte d'une solution globale au problème.

En pratique, les résultats, que nous avons obtenus en considérant un ensemble d'agents réflexes coordonnés par un unique agent au comportement déterministe, ont validé

ces deux concepts. Les résultats sont cohérents avec ce qu'un ingénieur ferait, et ceci de manière automatique.

Même si nous ne l'avons pas abordé dans ce chapitre, la détermination du système multi-agent proposé dans ce chapitre n'a pas été directe. Nous avons évalué l'utilisation de l'éco-résolution [46], modélisé le problème comme un ensemble d'agents s'auto-organisant, évalué la possibilité de faire apprendre chaque agent local en considérant l'apprentissage par renforcement, pour finalement aboutir au système présenté précédemment. Nous pensons aujourd'hui que l'option la plus intéressante à étudier pour la suite, et en particulier le passage au 3D, est de considérer que l'agent coordinateur que nous avons proposé doit devenir un agent apprenant au moyen de l'apprentissage par renforcement. Rétrospectivement, nous pensions aller dans cette direction en 2D, lors de l'introduction de la notion d'agent coordinateur. Il s'est ensuite avéré que notre agent au comportement déterministe que nous avons introduit à des fins de comparaison fournit des résultats satisfaisants.

Nous verrons au chapitre suivant que nous n'avons pas poursuivi dans cette direction en dimension 3 malgré tout. Les raisons sont multiples. Tout d'abord le nombre d'actions à implémenter en 3D est bien plus important et implique de gérer des non-conformités de natures variées à l'interface entre blocs voisins. Nous pensons que le travail d'implémentation en 3D de ces actions était bien trop chronophage et complexe durant la thèse. De même que le niveau auquel nous plaçons nos agents est discutable en 2D. En effet, des actions à un niveau inférieur qui générerait nos actions par blocs pourraient être préférables afin de limiter leur implémentation en utilisant des opérateurs sur maillages hexaédriques. Une extension en dimension 3 nous paraît cependant légitime et possible à moyen terme.

Pour toutes ces raisons, nous nous sommes orientés vers un problème différent en dimension 3 : au lieu d'appliquer des séries d'actions locales entraînant l'apparition de non-conformités dans la structure de blocs, nous considérons l'application successive d'opérations globales, à savoir des insertions de feuillets. Nous avons vu au chapitre 3 que de telles insertions nécessitent de fournir une "nappe" de faces le long de laquelle la couche de mailles sera insérée. Pour définir de telles nappes, nous nous sommes tournés vers l'utilisation d'un autre système multi-agent : les colonies de fourmis. Nous levons de cette façon la difficulté liée à l'implémentation 3D de la méthode précédente basée sur des modifications locales de blocs et la propagation de non-conformités.

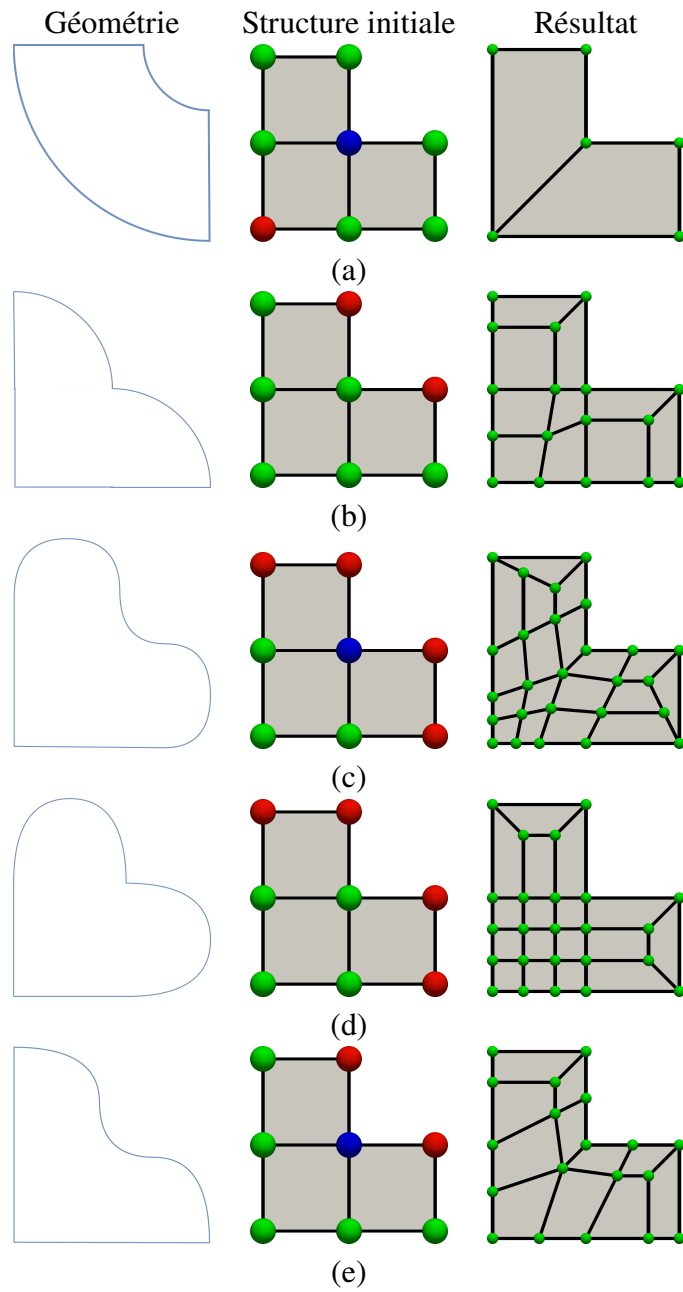


FIGURE 4.21 – Résultats du système multi-agent sur des géométries simples.

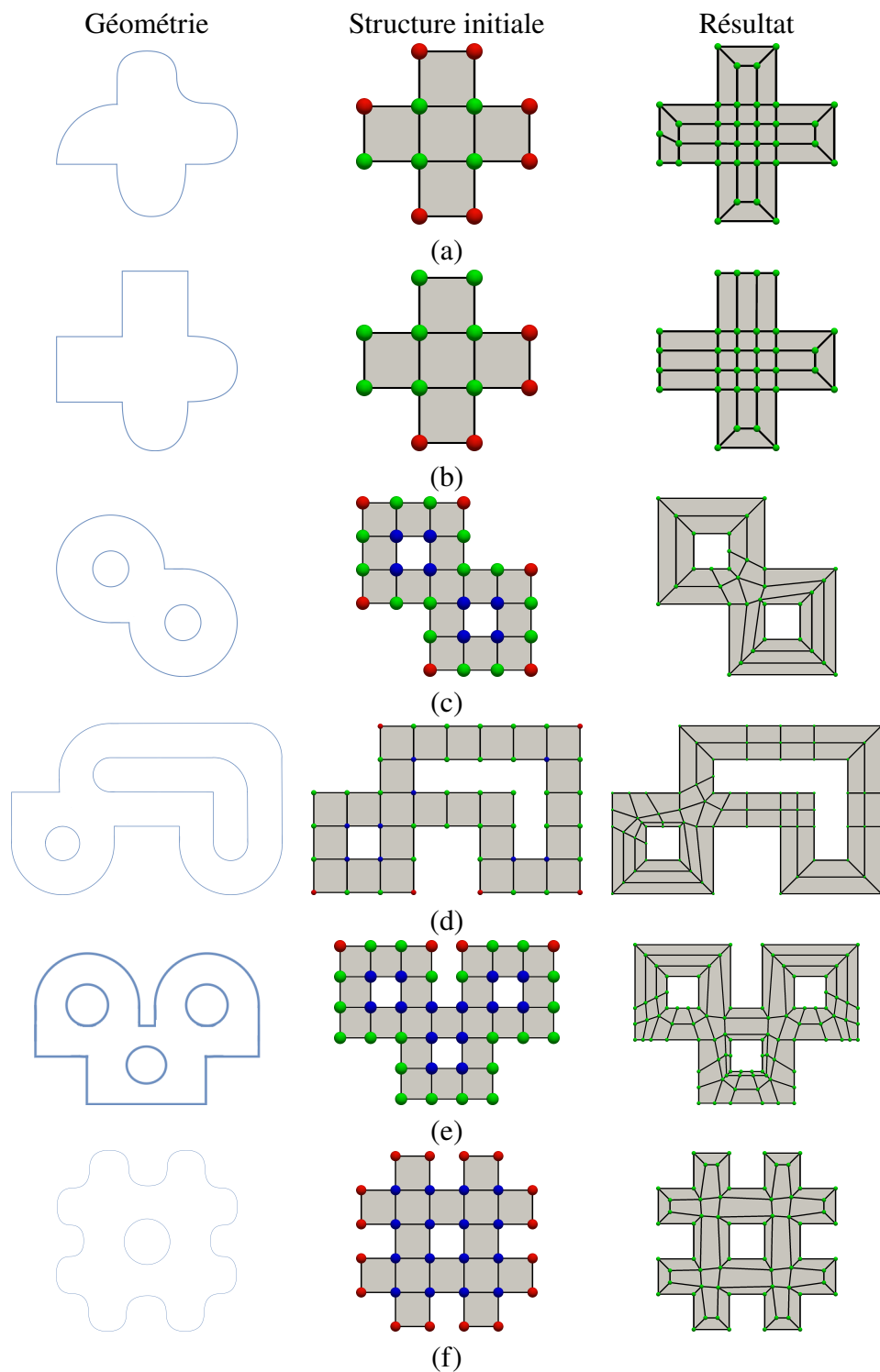


FIGURE 4.22 – Résultats du système multi-agent sur des géométries plus complexes.





## 5 - Colonie de fourmis pour piloter l'insertion de feuillets 3D

Modifier une structure de blocs hexaédriques ne peut pas se faire localement si on veut assurer la conformité entre blocs de la structure. Si au chapitre précédent, nous avons fait le choix d'introduire des non-conformités puis de les résoudre par propagation, nous suivons une approche différente dans ce chapitre. Nous considérons l'utilisation de l'insertion et de la suppression complète de couches de mailles, ou feuillets, pour justement garantir la conservation de blocs hexaédriques uniquement<sup>1</sup>. Si la suppression de couches est une opération assez simple à manipuler pour un ingénieur dans un logiciel interactif - on sélectionne une arête sur l'écran et on visualise directement la couche de mailles associée-, l'insertion d'une couche est plus complexe : il faut sélectionner une nappe valide de faces dans la structure de blocs.

Pour aider l'utilisateur/ingénieur, nous avons conçu une nouvelle approche basée sur un algorithme d'optimisation par colonies de fourmis (OCF). Un tel algorithme fait partie des techniques probabilistes permettant de résoudre des problèmes d'optimisation qui peuvent se réduire à la recherche de chemins dans des graphes, ou bien à des problèmes de sélection d'un sous-ensemble d'objets. Ainsi à la différence de l'algorithme de *Selective Padding*, présenté à la section 2.6.4, nous cherchons à proposer plusieurs solutions aux ingénieurs en un temps réduit. Nous considérons pour cela la structure combinatoire des blocs hexaédriques comme un graphe et nous essayons de fournir une surface discrète, qui soit une quasi-variété, améliorant la qualité de la structure des blocs. Cette qualité est définie sur des critères discrets, qui sont : (1) des contraintes de bord ; (2) la valence des arêtes participant à la définition de la surface discrète insérée.

Dans ce chapitre nous proposons une méthode de sélection de feuillet sur une entrée de type Polycube. Une sélection initiale de faces est donnée par l'utilisateur, ou bien déduite de la classification entre la structure de blocs et la géométrie. Les critères de qualité sont la satisfaction des contraintes d'alignement au bord par augmentation ou diminution de valence via l'insertion de couches de mailles.

### Publications et communications relatives à ce chapitre :

- Communication orale au Meshing workshop à LANL, *Ant Colony Optimization for Selective Padding in Hexahedral Block Structures*, en juillet 2023.
- Communication orale au *Symposium on Trends in Unstructured Mesh Generation, Ant Colony Optimization for Selective Padding in Hexahedral Block Structures*, à Albuquerque, USA, en juillet 2023.

---

1. Voir le chapitre 3 pour les notions relatives à l'insertion et la suppression de feuillets.

## 5.1 . Introduction à la méta-heuristique OCF pour des problèmes combinatoires

L'optimisation par colonie de fourmis OCF (ou « ACO » en anglais) simule le comportement des vraies fourmis [47]. Les fourmis réelles ont la capacité de trouver le chemin le plus court entre une source de nourriture identifiée par quelques fourmis au départ, et leur nid, et ce sans aucun repère visuel permettant de voir la source de nourriture en question. Pour cela, les fourmis communiquent entre elles par le biais de substances chimiques, appelées *phéromones* dans leur environnement immédiat. Elles sont également capables de s'adapter aux changements de l'environnement, par exemple en trouvant un nouveau chemin plus court lorsque l'ancien n'est plus praticable en raison d'un nouvel obstacle.

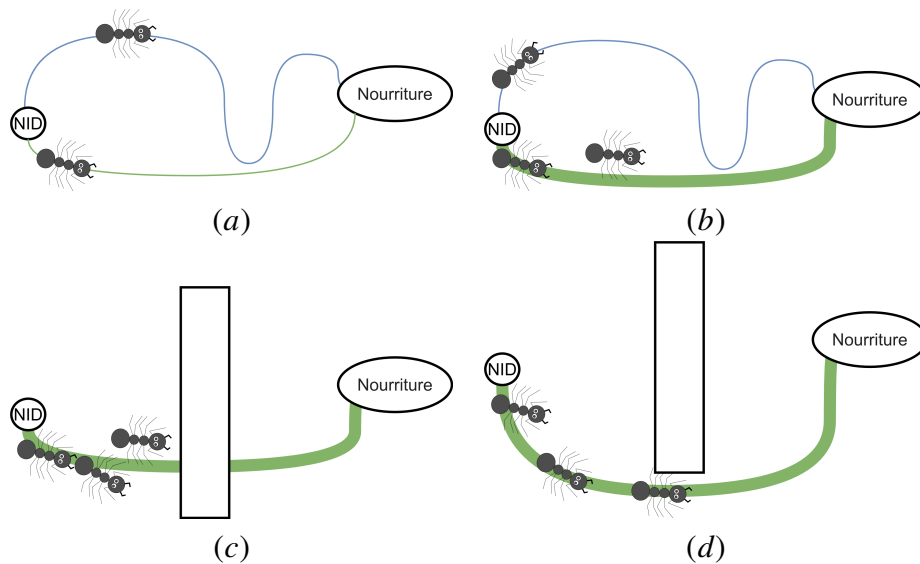


FIGURE 5.1 – (a) Des fourmis cherchent de la nourriture et tracent un chemin de phéromones (un bleu et un vert); (b) Le trajet le plus court marqué de plus de phéromones influence les autres fourmis à l'utiliser; (c) Un obstacle est placé sur le chemin précédemment découvert, le séparant en deux; (d) Les fourmis trouvent progressivement un autre chemin contournant l'obstacle.

En pratique, les fourmis déposent une certaine quantité de phéromones, notée  $\tau$ , sur le sol lors de leur déplacement depuis leur nid jusqu'à la source de nourriture (voir la figure 5.1.a avec deux traces de phéromones en bleu et en vert). Les phéromones forment le facteur trivial pour trouver la meilleure solution. Une fourmi sent les phéromones déposées par les fourmis précédentes et a tendance à imiter le chemin qui contient la quantité de phéromones le plus élevé. Ainsi un chemin ayant la plus grande quantité de phéromones devient le plus probable (voir la figure 5.1.b), et les fourmis auront tendance à choisir le chemin le plus court. Une autre propriété intéressante des colonies de fourmis est qu'une trace de phéromone précédente peut aider à trouver

un chemin même si un obstacle a été placé sur le précédent (voir la figure 5.1.c). Une nouvelle phase exploratoire aura alors lieu afin de trouver un nouveau chemin menant à la nourriture (voir la figure 5.1.d). A titre d'exemples applicatifs, nous pouvons citer l'utilisation d'OCF pour la mise en place d'un algorithme de routage d'un réseaux AntNet [48], pour des problèmes de sélection de sous-ensembles comme le problème du sac à dos [49] ou au problème du voyageur de commerce [50].

OCF est une méta-heuristique qui permet de mettre en place des algorithmes distribués, profitant en particulier des machines parallèles, pour un grand nombre de problèmes combinatoires. Un problème combinatoire classique est le problème du voyageur de commerce. Nous utilisons le formalisme du premier algorithme *Ant system* à avoir adapté OCF pour le voyageur de commerce. Les données considérées dans le problème du voyageur de commerce sont :

- Un ensemble fini  $X$  de nœuds représentant des villes ;
- Un ensemble fini  $U = (i, j) | i, j \in X$  d'arcs reliant les nœuds de  $X$  ;
- Et un ensemble de constantes  $d_{i,j}$  représentant la longueur de chaque arc  $(i, j) \in U$ , c'est-à-dire la distance séparant la ville  $i$  de la ville  $j$  tel que  $i, j \in X$ .

Nous imaginons alors un voyageur de commerce qui doit réaliser sa tournée en visitant une et une seule fois l'ensemble  $X$  des villes. Son objectif est de déterminer la suite de villes qui minimisera la distance à parcourir. À titre d'exemple, considérons la figure 5.2 où cinq villes, notées  $\{1, 2, 3, 4, 5\}$ , sont reliées par des arcs de différents poids  $d_{i,j}$  (écrits en bleu sur les arcs des graphes). Dans le problème du voyageur de commerce, on cherche à trouver un circuit qui soit un cycle hamiltonien, c'est-à-dire qui passe exactement une fois par tous les sommets du graphe. La longueur d'un circuit  $\mu$  est la somme des longueurs des arcs qui le composent, c'est-à-dire que l'on a :

$$L(\mu) = d_{q,1} + \sum_{i=1}^{q-1} d_{u_i, u_{i+1}}.$$

Par exemple sur la figure 5.2.a, le circuit hamiltonien en vert composé des arcs  $(1, 2)$ ,  $(2, 3)$ ,  $(3, 5)$ ,  $(5, 4)$  et  $(4, 1)$  a pour longueur 26. Même si le problème semble simple, nous sommes en fait confrontés à une explosion combinatoire du nombre de solutions à construire. Il n'existe pas de méthode de résolution permettant d'avoir des solutions exactes en un temps raisonnable pour des instances de graphes  $G$  ayant un grand nombre de sommets  $n$ . Il existe  $\frac{(n-1)!}{2}$  chemins considérant l'utilisation d'un algorithme naïf, qui testerait toutes les solutions possibles et aurait une complexité de  $O(n!)$ , ce qui en pratique ne permet pas de trouver de solution en un temps acceptable.

## 5.2 . Méta-heuristique OCF pour l'optimisation de structures de blocs en dimension 2

Afin de mettre en place notre algorithme OCF appliqué au problème de sélection d'un sous-ensemble de faces dans une structure de blocs, nous nous sommes inspirés

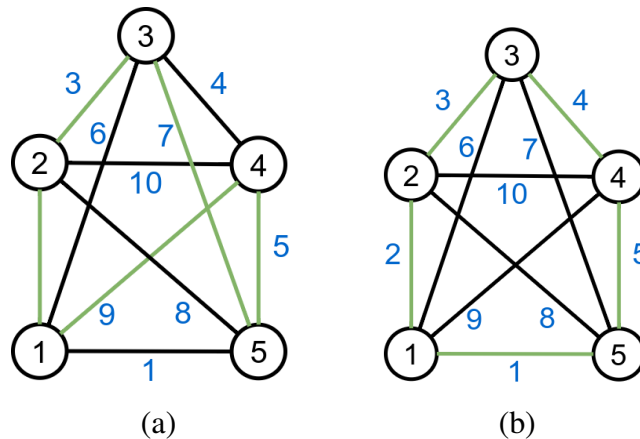


FIGURE 5.2 – Cycle hamiltonien en vert de valeur 26 (a) et 15 pour (b).

de son application sur le voyageur de commerce. Nous devons, avant de pouvoir l’appliquer, expliciter les relations spécifiques du maillage qui seront utiles ici. Nous illustrons l’approche en 2D dans un premier temps, mais notons dès à présent que celle-ci se généralise en dimension 3.

### 5.2.1 . Définition du problème 2D à résoudre

Étant donné une structure de blocs quadrilatérales de type polycube, l’objectif est d’optimiser cette structure par insertion de couches de mailles. Le critère à optimiser est similaire à celui du chapitre 4, à savoir éliminer les sommets critiques présents potentiellement sur le bord du domaine. L’élimination d’un sommet critique qui nécessiterait l’augmentation ou la diminution de sa valence se fera par insertion de couches. Localement, cela revient à sélectionner les deux arêtes au bord incidentes à ce sommet et à insérer une couche. A titre d’exemple, nous reprenons à la figure 5.3 les configurations présentées sur la figure 2.11, page 25. En (a), le sommet en noir requiert une augmentation de valence. L’insertion d’une couche le long de ses 2 arêtes incidentes au bord produira le résultat présenté en (b), où le sommet noir a été poussé à l’intérieur du domaine - sommet entouré en bleu. De même, le sommet en noir visible en (c), qui nécessite une diminution de sa valence, sera poussé à l’intérieur du domaine - sommet entouré en rouge en (d) - via l’insertion d’une couche le long de ses 2 arêtes incidentes au bord.

Pour effectuer l’insertion d’une couche, il nous faut déterminer un ou plusieurs chemins d’arêtes qui contiendront les arêtes au bord incidentes aux sommets critiques. Nous exprimons deux contraintes pour la construction de tels chemins :

1. Un chemin  $c$  ne s’auto-intersecte pas, c’est-à-dire que l’on n’a pas plus de deux arêtes dans la sélection autour d’un nœud ;
2. Un chemin  $c$  peut être un cycle mais peut aussi s’arrêter en un nœud  $n$  du bord sous certaines conditions :

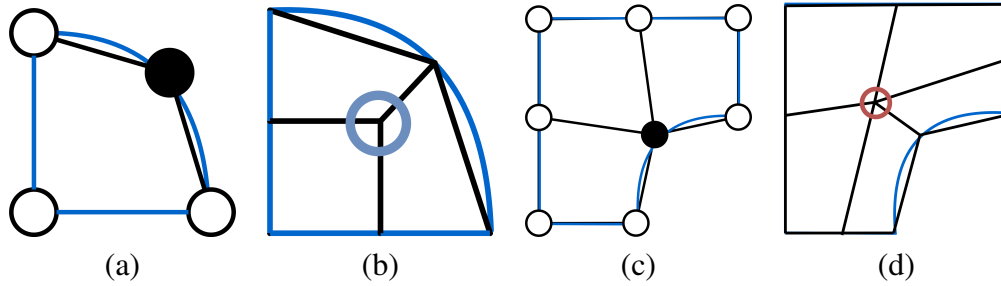


FIGURE 5.3 – Résolution d'un problème de criticité au bord par insertion de couches pour un cas convexe en (a) et (b), et un cas concave en (c) et (d).

- Si  $n$  est classifié sur un sommet géométrique  $S$ , alors l'arête de  $c$  qui est incidente à  $n$  est classifiée sur une courbe incidente à  $S$ ;
- Si  $n$  est classifié sur une courbe géométrique  $C$ , alors l'arête de  $c$  qui est incidente à  $n$  est classifiée dans la surface.

### 5.2.2. Qualité d'un chemin d'arêtes en 2D

Pour déterminer quel chemin sélectionner, nous devons être capable d'évaluer la qualité d'un chemin. Pour cela, nous utilisons la représentation d'une structure de blocs notée  $B = (N, A, F)$  composée d'ensembles de nœuds  $N$ , d'arêtes  $A$  et de faces  $F$ , de tailles respectives  $\#N, \#A, \#F$ . Nous définissons  $S$  l'ensemble des arêtes d'un chemin et nous notons  $Q(S)$  la fonction qui associe une qualité à  $S$ . Notre objectif est de trouver le chemin qui maximise  $Q(S)$ , sachant que  $Q(S)$  est définie comme suit :

$$N_{part} = \frac{\#n \in S - \#nt \in S}{\#n \in S} \quad (5.1)$$

$$El_{part} = \frac{\#a \in A - \#a \in S}{\#a \in A} \quad (5.2)$$

$$Q(S) = Z * (Ar_{part} + \lambda * El_{part}) \quad (5.3)$$

Cette fonction de qualité a deux termes principaux :

1. Le premier est le terme  $N_{part}$ , qui correspond au nombre de "tournants" de nœud présents dans le chemin d'arêtes  $S$  sélectionnées. Étant donné un nœud  $n$  incident à deux arêtes de  $S$ , un tournant de nœud est calculé en comparant les normales des arêtes de  $S$  incidentes à  $n$  : si leurs directions sont différentes alors on a un tournant de nœud noté  $nt$ . Sur la figure 5.4, le nœud  $n_0$  dispose d'un tournant mais pas les autres nœuds appartenant à l'ensemble des arêtes sélectionnées (en vert). Cette métrique est inspirée de [34], où les auteurs minimisent le nombre d'éléments insérés, les tournants d'arêtes et aussi des tournants de sommets. Ce dernier est de notre point de vue, corrélé à la partie tournant d'arête.
2. Le second terme  $El_{part}$  est la différence entre le nombre d'arêtes total dans la structure de blocs  $\#a \in A$  et le nombre d'arêtes dans la sélection  $\#a \in S$  divisé par le nombre d'arêtes total.

Le facteur  $\lambda$  est une pondération de l'importance du nombre d'éléments dans la qualité de la solution. Ainsi  $\lambda = 0$ , implique que les tournants de nœuds n'ont aucune importance alors que si il augmente, l'influence du nombre  $nt$  diminue. Pour sa part, le facteur  $Z$  est un facteur d'amplitude utile pour augmenter la différence entre deux sélections.

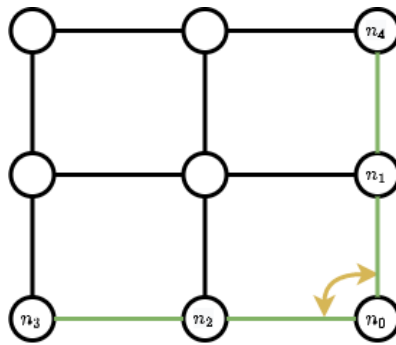


FIGURE 5.4 – Illustration d'un tournant de nœud  $nt$  où deux faces incidentes à  $n_0$  font partie de la sélection et la direction des arêtes est différente.

### 5.2.3 . Construction d'un chemin solution par une fourmi

Afin d'illustrer le processus de construction de solutions à l'aide de colonies de fourmis, commençons par illustrer la construction d'un chemin par une unique fourmi. Pour cela considérons l'exemple de la figure 5.5, où les arêtes en rouge sur la figure 5.5.a sont incidentes à un nœud contraint, et doivent donc appartenir au chemin d'arêtes à construire.

Notre problème est d'insérer un chemin d'arêtes. À l'instant initial, en (a), La fourmi dispose de deux arêtes déjà sélectionnées en rouge. Les nœuds en vert forment un front, ou une bordure de notre sélection d'arêtes, que nous noterons *bnd*. Ce front est mémorisé et modifié par la fourmi durant l'exécution. Partant des nœuds en vert constituant le front, les arêtes incidentes en bleu sont proposées pour étendre le chemin. Notre fourmi sélectionne de manière probabiliste une arête parmi les quatre. Une fois une arête sélectionnée, on obtient la situation illustrée à la figure 5.5(b). La fourmi marque le nouveau sommet inséré comme étant terminal puisque le sommet en question est classifié sur un coin géométrique et que l'arête incidente qui vient d'être sélectionnée est classifiée sur une courbe incidente à ce coin géométrique. Notons que le chemin aurait aussi pu continuer à se poursuivre.

Le front est désormais réduit à un sommet situé à l'opposé. On poursuit le chemin en sélectionnant une arête parmi les deux restantes. En l'occurrence, ce sera l'arête verte menant la fourmi sur un nœud interne à la surface sur la figure 5.5(c). À cet instant, l'arête en jaune ne fait pas partie des candidats possibles pour respecter la contrainte 1 (un nœud du chemin serait sinon incident à 3 arêtes du chemin). La fourmi sélectionne alors l'une des deux arêtes bleues. En (d), l'arête sélectionnée est celle

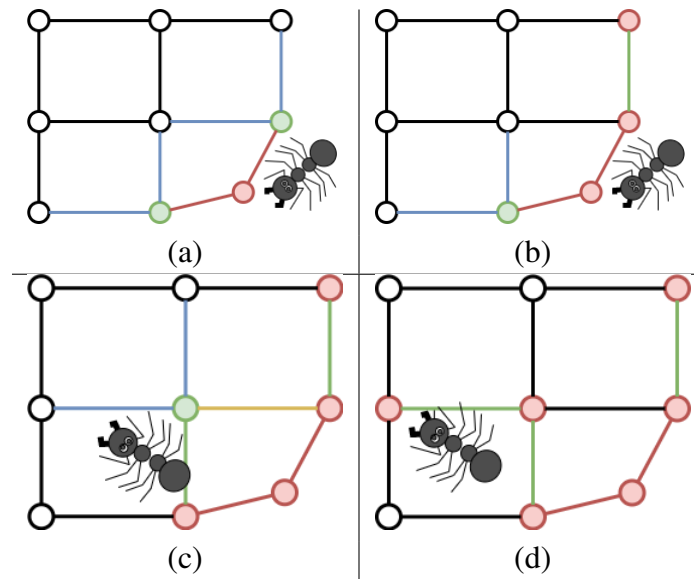


FIGURE 5.5 – Construction d’une solution par une fourmi. En (a), le problème initial avec des arêtes contraintes en rouge ; En (b), sélection de l’arête en haut à droite ; En (c), sélection d’une arête interne à la structure de blocs ; Enfin, en (d), sélection de l’arête à gauche et fin de l’algorithme de construction. Le résultat de l’exécution contient 3 faces et 2 tournants de sommet.

en vert et son sommet extrémité est défini comme terminal par la fourmi. Ceci est possible car le sommet est classifié sur une courbe et que son unique arête incidente dans le chemin est classifiée sur la surface. Le chemin est alors complet et une couche de mailles pourrait être insérée.

L’étape suivante pour améliorer la qualité du maillage est d’insérer un feuillet avec l’opération de *pillowing* illustré sur la figure 5.6.a. Une couche de blocs en bleu est insérée en utilisant le chemin construit par notre fourmi. Des sommets de singularité 3 (entourés en bleu) et 5 (entourés en rouge) sont insérés lors de l’opération (figure 5.6.b).

#### 5.2.4 . Algorithme général

Notre algorithme, nommé OCFSF pour *Optimisation par Colonie de Fourmis pour la Sélection de Feuillet*, peut se découper en trois parties. La première est l’initialisation, la seconde est la construction de la sélection et la troisième est la mise à jour de la trace de phéromones  $\tau$  sur les arêtes conformément à l’algorithme général OCF (illustré par l’algorithme 5).

#### Initialisation

Nous initialisons les phéromones avec une petite valeur notée  $\tau_{min} = 0.1$ . Cette valeur est donc attribuée à toutes les arêtes de la structure de blocs. Ensuite nous



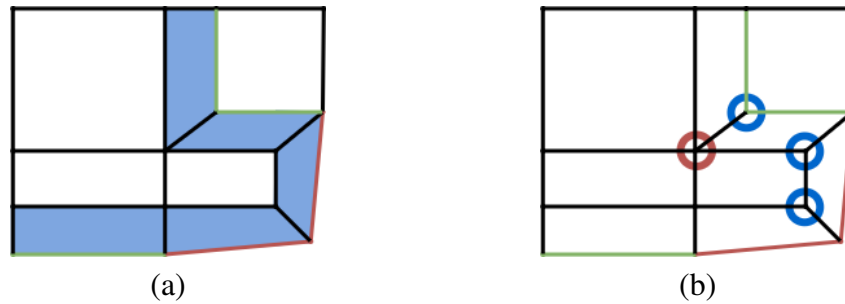


FIGURE 5.6 – Illustration après insertion de feuillet. En (a), les quadrilatères en bleu sont insérés à partir de la sélection initiale d’arêtes en rouge et les arêtes sélectionnées par notre algorithme en vert. En (b), on voit que ce processus a créé des nœuds singuliers dans le domaine : une singularité 5 entourée en rouge et plusieurs singularités 3 en bleu.

---

**Algorithme 5 :** Squelette de l’algorithme OCF.

---

- 1 Initialiser les paramètres, initialiser les phéromones ;
  - 2 **Tant que** la condition d’arrêt n’est pas satisfaite **faire**
  - 3     Les fourmis construisent des solutions ;
  - 4     Effectuer une recherche locale // Éventuellement
  - 5     Mise à jour des phéromones ;
  - 6 **fin**
-

initialisons chaque solution  $S$  avec les arêtes contraintes déduites précédemment.

### Construction de la sélection

La construction d'une sélection se fait en deux temps : (1) la construction de l'ensemble des arêtes candidates selon nos contraintes, et (2) l'établissement des probabilités de transition pour chacune de ces arêtes candidates. L'algorithme 8 permet de construire l'ensemble des candidats  $C$ . De la ligne 1 à 7 il s'agit de l'actualisation de notre front  $bnd$ . Pour cela nous parcourons toutes les arêtes de notre solution, nous récupérons les nœuds et comptons le nombre d'arêtes incidentes à ces nœuds et présentes dans la solution grâce à l'algorithme 6. Ensuite selon la classification de l'arête  $a$  et du nœud  $n$ , conformément à l'algorithme 7, au tableau 5.8 et à la figure 5.7. Si le nœud  $n$  dispose d'une arête dans la solution et que la condition  $ajouterFront$  est vérifiée alors  $bnd(n) = ns(e)$ , et le nœud  $n$  est inséré dans le front.

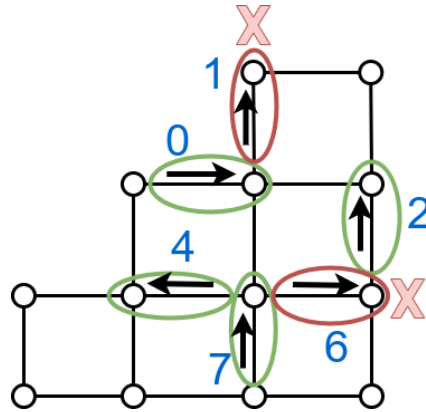


FIGURE 5.7 – Illustration des cas d'arrêt pour l'avancement du front de nœuds. La classification d'une paire (arête,nœud) entourée en vert indique que le nœud sera ajouté au front alors que cela ne sera pas le cas pour les couples entourés en rouge. Le chiffre noté sur le côté correspond au numéro du cas correspondant dans le tableau de classification. 5.8

Ensuite selon les lignes 9 à 19 de l'algorithme 8, nous construisons l'ensemble des candidats. Pour chaque nœud étant dans le front ( $bnd(n) = 1$ ), les arêtes de ces nœuds sont ajoutées dans l'ensemble des candidats si et seulement si elles ne sont pas incidentes à un autre nœud de la solution (il s'agit du cas que nous avons rencontré avec l'arête jaune lors de l'exécution 5.5.c).

Les probabilités de transition entre les arêtes candidates sont calculées à l'aide de la formule suivante :

$$\forall a_i \in C, p(c) = \frac{[\tau(a_i)]^\alpha \cdot [\eta(a_i)]^\beta}{\sum_{a_j \in C} [\tau(a_j)]^\alpha \cdot [\eta(a_j)]^\beta}$$

n°	$cf(a)$	$cf(n)$	Classification a	Classification n	Ajout dans front	Existence
0	1	0	surface	concavité	Oui	Oui
1	1	1	surface	convexité	Non	Oui
2	1	2	surface	externe	Oui	Oui
3	1	3	surface	interne	Non	Non
4	2	0	volume	concavité	Oui	Oui
5	2	1	volume	convexité	Non	Non
6	2	2	volume	externe	Non	Oui
7	2	3	volume	interne	Oui	Oui

FIGURE 5.8 – Tableau de classification pour un couple composé d’une arête  $a$  et d’un nœud  $n$ .  $cf(x)$  est la valeur de classification d’une cellule  $x$ .

---

**Algorithme 6 :  $ns(n)$**

---

**Data :** Nœud  $n$

```

1 début
2    $cpt \leftarrow 0$  ;
3   Pour  $a \in Noeuds(n)$  faire
4     Si  $a \in S$  alors
5        $cpt \leftarrow cpt + 1$  ;
6     fin
7   fin
8   retourner  $cpt$  ;
9 fin

```

---

---

**Algorithme 7 : ajouterFront(Nœud n, Arête a)**

---

```
1 début
2   Si  $cf(n) = 1 \wedge cf(a) = 0$  alors
3     retourner Vrai ;
4   sinon si  $cf(n) = 1 \wedge cf(a) = 1$  alors
5     retourner Faux ;
6   sinon si  $cf(n) = 1 \wedge cf(a) = 2$  alors
7     retourner Vrai ;
8   sinon si  $cf(n) = 1 \wedge cf(a) = 3$  alors
9     retourner Faux ;
10  sinon si  $cf(n) = 2 \wedge cf(a) = 0$  alors
11    retourner Vrai ;
12  sinon si  $cf(n) = 2 \wedge cf(a) = 1$  alors
13    retourner Faux ;
14  sinon si  $cf(n) = 2 \wedge cf(a) = 2$  alors
15    retourner Faux ;
16  sinon si  $cf(n) = 2 \wedge cf(a) = 3$  alors
17    retourner Vrai ;
18 fin
```

---

où  $\eta$  est une valeur heuristique que nous calculons dans le but de privilégier les arêtes qui ne forment pas de tournant de sommet. Pour cela nous attribuons un poids élevé pour une arête candidate ne formant pas un tournant de nœud avec une arête déjà dans la sélection et un poids moins élevé pour une arête formant un tournant de nœud. Le poids que nous attribuons à une arête  $a$  est notée  $h(a)$ . Considérons l'exemple de la figure 5.9. Les arêtes en jaune ont par exemple une probabilité plus faible d'être sélectionnées que celles en bleu. Les paramètres  $\alpha$  et  $\beta$  déterminent l'influence des phéromones sur l'attrait de la trace et l'information de l'heuristique. Lorsque  $\alpha$  est fixé à 0, les arêtes proches ne formant pas de tournant sont sélectionnées de manière gloutonne c'est à dire qu'une arête  $a$  en bleu sur la figure 5.9 aura une valeur  $p(a) = 0$ . Les phéromones n'auront plus d'importance dans le choix de la fourmi. Si  $\beta = 0$  alors, de la même manière l'heuristique n'a plus d'importance et les phéromones sont alors le seul facteur pour la prise de décision.

Dans l'algorithme général 9, une fois que les phéromones, les paramètres et la sélection sont initialisés, les  $m$  fourmis construisent une sélection d'arêtes. Chaque fourmi construit sa sélection de manière itérative de ligne 6 à ligne 12.

### Mise à jour des phéromones

Ces sélections sont ensuite triées en fonction de la qualité  $Q(S)$  et en mémorisant la meilleure sélection depuis le début de l'algorithme  $S_{Best}$ . Les phéromones sont

---

**Algorithme 8 : *candidats()***

---

**Données :**  $S$  : ensemble d'arêtes dans la solution

**Résultat :** Ensemble de candidats  $C$

```
1 début
2   Pour  $a \in S$  faire
3     Pour  $n \in \text{Noeuds}(a)$  faire
4       Si  $ns(n) = 1 \wedge \text{ajouterFront}(n, a)$  alors
5         |  $bnd(n) \leftarrow ns(e)$ ;
6         | fin
7     fin
8   fin
9    $C \leftarrow \emptyset$ ;
10  Pour  $n \in N$  faire
11    Si  $bnd(n) = 1$  alors
12      Pour  $a \in \text{Arete}(n)$  faire
13        Si  $\forall n_a \in \text{Noeuds}(a), n_a \notin S$  alors
14          |  $C \leftarrow C \cup a$ ;
15          | fin
16        fin
17      fin
18    fin
19  retourner  $C$ ;
20 fin
```

---

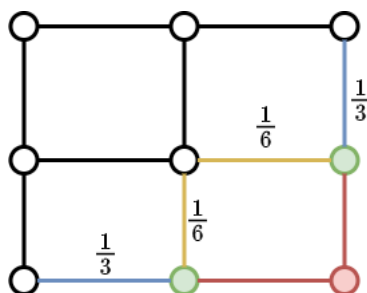


FIGURE 5.9 – Les arêtes en jaune sont des tournant de nœuds, ce qui implique qu'elles ont une probabilité plus faible d'être sélectionnées que les arêtes en bleu.

actualisées sur chaque arête réduisant la trace précédente avec le facteur d'évaporation  $\rho \in [0, 1]$ . La quantité de phéromones déposée est calculée en fonction de la distance entre la meilleure solution  $Q(S_{Best})$  et les solutions des meilleures fourmis  $k \in [1; e]$ . Le nombre  $e$  est le nombre de fourmis qui seront prises en compte pour la modification de la trace. L'algorithme se termine quand un nombre d'itérations déterminé par l'utilisateur est dépassé, qu'une qualité suffisante est atteinte, ou qu'une limite de temps est dépassée.

---

**Algorithme 9 : Algorithme général en dimension 2.**

---

**Données :**

1 **début**

2     Initialiser phéromones et paramètres ;

3     Initialiser  $S_k$  avec  $S_{ini}$  ;

4     **Répéter**

5         **Pour** chaque fourmi  $k \in [1; m]$  **construire une sélection faire**

6              $C \leftarrow \text{candidats}()$  ;

7             **Tant que**  $C \neq \emptyset$  **faire**

8                 Choisir une arête  $a \in C$  avec la probabilité ;

9                 
$$p(a_i, S_k) = \frac{[\tau_{factor}(a_i, S_k)]^\alpha \cdot [\eta_{factor}(a_i, S_k)]^\beta}{\sum_{a_j \in \text{candidats}} [\tau_{factor}(a_j, S_k)]^\alpha \cdot [\eta_{factor}(a_j, S_k)]^\beta}$$
 ;

10                  $C \leftarrow \text{candidats}()$  ;

11             **fin**

12         **fin**

13         Trier les solutions par ordre décroissant en fonction de ;

14          $Q(S) = Z * (Ar_{part} + \lambda * El_{part})$  ;

15         **Pour** chaque fourmi  $k \in [1; e]$  **faire**

16             **Pour** chaque arête  $a \in S$  **faire**

17                  $\tau(a_i) \leftarrow \tau(a_i) * \rho + \frac{1}{1 + Q(S_{Best}) - Q(S_k)}$  ;

18             **fin**

19         **fin**

20     **jusqu'à**  $itermax$  atteint ou une qualité  $Q(S)$  convient;

21 **fin**

---

### 5.3 . Méta-heuristique OCF pour l'optimisation de structures de blocs en dimension 3

Nous étendons maintenant l'approche proposée en dimension 2 à la dimension 3. Cette extension est directe. Les différences principales résident dans le fait que l'on raisonne au niveau des arêtes et non plus des nœuds, et que la sélection d'un chemin d'arêtes est remplacé par la sélection d'une nappe de faces.

### 5.3.1 . Définition du problème à résoudre

Nous cherchons à créer une nappe de faces qui va permettre l'introduction d'un feuillet dans la structure de blocs. Une nappe est un ensemble connexe de faces  $F$  tel que pour chaque arête  $a$  incidente à une face de  $F$ , nous avons :

- Si  $a$  est classifiée dans un volume, alors  $a$  est incidente à exactement deux faces de  $F$  ;
- Si  $a$  est classifiée sur une surface, alors  $a$  est incidente à deux faces ou à une face classifié dans le volume ;
- Si  $a$  est classifiée sur une courbe  $c$ , alors  $a$  est incidente a deux faces ou à une face classifiée sur une surface incidente à la courbe  $c$ .

L'objectif est de trouver une ou plusieurs nappes de faces qui modifient, lorsque nous effectuons l'opération d'insertion de couche, les valences de arêtes critiques en 3D. Pour cela, il est nécessaire, comme en 2D, de démarrer des arêtes critiques et de faire apparaître leurs faces incidentes au bord dans la nappe à insérer. À titre d'exemple, considérons le modèle présenté sur la figure 5.10, où l'arête rouge en (b) est critique et nécessite de voir son nombre de blocs incidents augmenter de un à deux. Dans [34], cette détection est faite en utilisant la métrique *Scaled Jacobian (SJ)* et un seuil fixé par l'utilisateur avec parfois des complications sur l'ensemble d'arêtes à traiter comme des trous au milieu de leur ensemble d'arêtes. De notre côté, les arêtes critiques sont définies en considérant uniquement la classification et la valence des arêtes au bord. Cela nous permet d'avoir toujours des ensembles exploitables de faces, au contraire de [34]. Cet ensemble de faces forme notre sélection initiale, notée  $S_{ini}$ . Sur la figure 5.10.(c) l'arête en rouge dispose de deux faces incidentes colorisées en rouge. Ce sont ces faces qui composent notre ensemble  $S_{ini}$ . Tout l'enjeu de notre méthode est d'ajouter des faces à  $S_{ini}$  afin que cet ensemble constitue une nappe valide pour effectuer une opération d'insertion de feuillet tout en minimisant le nombre de blocs et de singularités insérées et en améliorant la qualité de la structure de blocs.

Les contraintes définies plus haut sont toutes transposées en 3D de cette manière.

1. Pas plus de deux faces dans la sélection autour d'une arête
2. Nos fourmis doivent s'arrêter dans certaines conditions. Lorsqu'une fourmi arrive sur le bord elle devra s'arrêter en considérant la classification de l'arête et de la face choisie.

Celles-ci ne sont pas suffisantes pour générer une sélection valide pour l'opération d'insertion de feuillet. En effet nous n'assurons pas lors de l'étape de construction que la sélection soit valide pour l'opération de *pillowing*. Les meilleures sélections seront validées une fois l'algorithme terminé en utilisant la définition d'une nappe.

### 5.3.2 . Qualité d'une sélection de face en 3D

La fonction qui associe une qualité à une sélection de face  $Q(S)$  est définie par :

$$Q(S) = Z * (Ar_{part} + \lambda * El_{part})$$

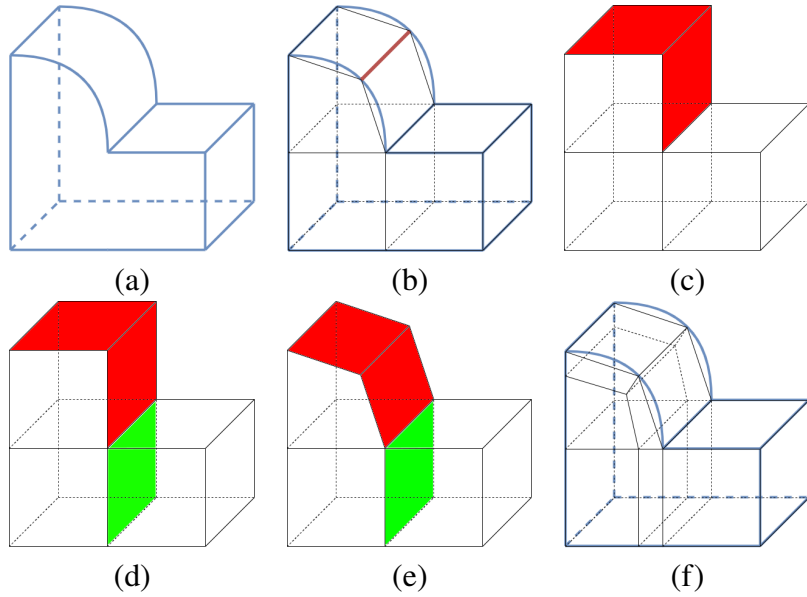


FIGURE 5.10 – (a)  $\Omega$  (b)  $\Omega + B_H$  et une arête  $c$  en rouge (c)  $C(B_H)$ ,  $S_{ini}$  en rouge (d)  $S_{ini}$  augmentée en vert (e)  $S$  sur  $B_H$  (f)  $\Omega + B_H$  après *pillowing*

avec

$$Ar_{part} = \frac{\#a \in S - \#a_t \in S}{\#a \in S}$$

et

$$El_{part} = \frac{\#f \in F - \#f \in S}{\#f \in F}.$$

Comme en dimension 2, le terme  $Ar_{part}$  correspond à une proportion de nombre de tournant d'arêtes  $\#a_t$  par rapport au nombre total d'arêtes. Un tournant d'arête est une extension directe de la notion de tournant de nœud. Un tel tournant est illustré sur la figure 5.11, où trois faces  $f_0, f_1, f_2$  sont sélectionnées. Le second terme,  $El_{part}$ , porte sur le nombre de faces dans la sélection  $\#f \in S$  par rapport au nombre total de faces  $\#f \in F$ . Comme en dimension 2,  $\lambda$  est le facteur de pondération entre les deux termes  $Ar_{part}$  et  $El_{part}$  et  $Z$  est un facteur d'amplitude.

### 5.3.3 . Construction d'une sélection de faces par une fourmi

Nos fourmis vont construire une sélection de faces de la même manière qu'en dimension 2\*, c'est-à-dire en utilisant un algorithme d'avancée de front avec certaines conditions d'arrêt. Nous illustrons le processus en dimension 3 à l'aide de l'exemple de la figure 5.12. Sur la figure 5.12.a, les faces en rouge font partie de la sélection initiale, et les arêtes en vert sont les arêtes du front, ou par analogie avec le cas 2D, ces arêtes forment le bord de la sélection courante. L'ensemble des faces candidates est construit en utilisant ce front d'arêtes. Ce sont les faces bleues visibles sur la figure 5.12.b. Une fois qu'une face est sélectionnée parmi ces faces bleues, le front d'arêtes est mis à jour et de nouvelles faces candidates sont déduites pour une éventuelle sélection à venir



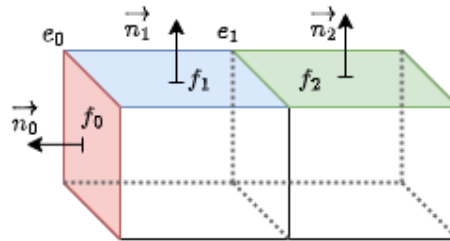


FIGURE 5.11 – Structure de blocs comportant deux blocs. Les faces  $f_0$  en rouge et  $f_1$  en bleu forment un tournant d'arête sur l'arête  $e_0$  qu'ils ont en commun. Leur vecteurs normaux  $\vec{n}_0$  et  $\vec{n}_1$  ne sont pas égaux. Au contraire  $\vec{n}_1$  et  $\vec{n}_2$  sont égaux, et donc  $e_1$  n'est pas un tournant d'arête.

(voir la figure 5.12.c). La construction est terminée lorsqu'il ne reste plus aucune arête à traiter dans le front. C'est le cas sur la figure 5.12.d pour notre exemple. Notons que comme en 2D, une arête au bord du domaine et incidente à une seule face de la sélection peut être retirée du front de propagation si elle vérifie les bonnes conditions de classification et que la fourmi "décide" de ne pas continuer dans cette direction.

### 5.3.4 . Algorithme général

Les paramètres de l'algorithme général sont :  $nbItermax, m, \alpha, \beta, \rho, \tau_{initial}, \tau_{min}, \tau_{max}$ , et  $e$ . La signification de chacun de ces paramètres est donné dans le tableau 5.1. L'algorithme général complet est lui décrit par l'algorithme 13, qui reprend la structure de l'algorithme 2D.

$nbItermax$	Nombre d'itérations maximum
$m$	Nombre de fourmis
$\alpha$	Coefficient d'augmentation du facteur pour les phéromones
$\beta$	Coefficient d'augmentation du facteur pour l'heuristique géométrique
$\rho$	Coefficient d'évaporation
$\tau_{initial}$	Valeur initiale de phéromone sur une face
$\tau_{min}$	Valeur minimale de phéromone sur une face
$\tau_{max}$	Valeur maximale de phéromone sur une face
$e$	Nombre de fourmis augmentant les phéromones
$\tau$	Montant de phéromone présent sur une face

TABLE 5.1 – Paramètres de l'algorithme général en dimension 3.

### Initialisation

Les phéromones sur les faces sont initialisées à la valeur  $\tau_{ini}$  et les sélections  $S_k$  à la valeur  $S_{ini}$ .

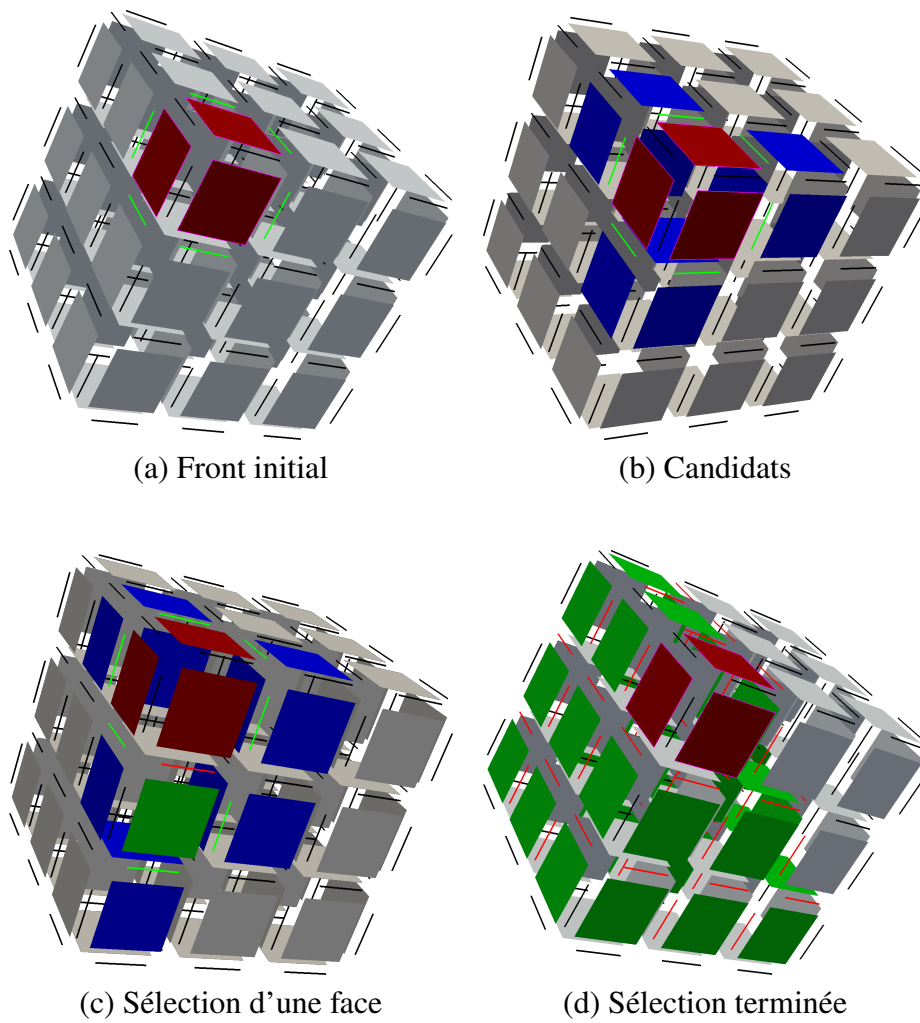


FIGURE 5.12 – Processus de création d'une nappe par une fourmi. En (a), le front d'arêtes initial est composé des arêtes vertes, et les faces en rouge sont les faces de la sélection initiale ; En (b), les faces candidates sont en bleu ; En (c), la sélection d'une face candidate verte ; En (d) la construction de la sélection est terminée.

## Construction de la sélection

La sélection des faces s'effectue de manière probabiliste comme en dimension 2. De même, l'heuristique est similaire à celle proposée en 2D en remplaçant les tournants de nœuds par des tournants d'arêtes. Afin de privilégier la recherche de surfaces planes, nous attribuons un poids élevé pour une face candidate ne formant pas un tournant d'arête avec une face déjà dans la sélection. Le poids d'une face  $f$  est noté  $h(f)$ . Considérons l'exemple de la figure 5.13. Les faces en rouge  $f_0$  et  $f_1$  font partie de l'ensemble  $S_{initial}$  alors que  $f_2$  et  $f_3$  sont des faces candidates. Les faces  $f_1$  et  $f_3$  forment un tournant d'arête nous attribuons alors  $h(f_3) = 1/2$  alors que ce n'est pas le cas de  $f_2$  et  $f_1$ . Nous calculons la probabilité de sélectionner une face  $f$  candidate de la manière suivante :

$$p(f) = \frac{h(f)}{\sum_{j \in C} h(f_j)}$$

En remplaçant, nous obtenons bien que  $f_2$  à une plus grande probabilité d'être sélectionnée.

$$p(f_3) = \frac{\frac{1}{2}}{1 + \frac{1}{2}} = \frac{1}{3}$$

$$p(f_2) = \frac{1}{1 + \frac{1}{2}} = \frac{2}{3}$$

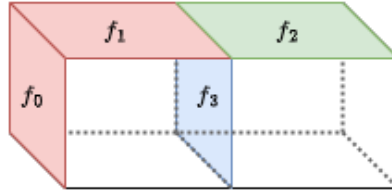


FIGURE 5.13 – Configuration pour la propagation de nappes en 3D. La nappe composée de  $f_0$  et  $f_1$  peut, entre autres, se propager en ajoutant la face  $f_2$  ou la face  $f_3$ .

Les probabilités de transition pour les faces candidates sont calculées en appliquant la formule suivante :

$$\forall f_i \in C, p(f_i, S_k) = \frac{[\tau_{factor}(f_i, S_k)]^\alpha \cdot [\eta_{factor}(f_i, S_k)]^\beta}{\sum_{f_j \in candidats} [\tau_{factor}(f_j, S_k)]^\alpha \cdot [\eta_{factor}(f_j, S_k)]^\beta}$$

Le front d'arêtes peut cesser sa propagation lorsqu'une fourmi atteint le bord du domaine. Une arête peut être classifiée sur une courbe, une surface ou un volume. Dans le cas d'une structure de blocs de type Polycube, cette classification peut être déduite à partir du nombre d'hexaèdres incidents à l'arête. Les arêtes sont alors répertoriées dans quatre catégories : E0, E1, E2 ou E3 (voir la figure 5.15). Une face quant à elle peut être classifiée uniquement sur une surface  $F2$  avec un seul hexaèdre ou un volume

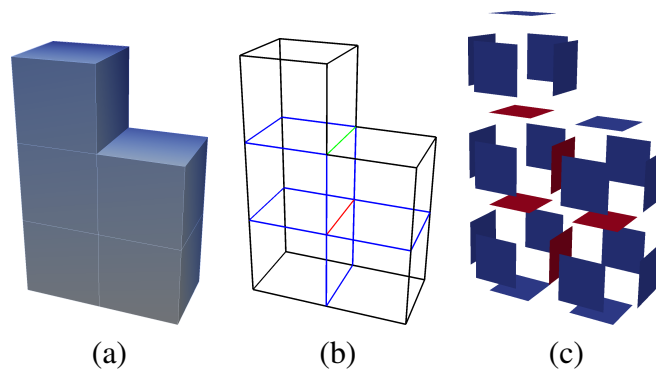


FIGURE 5.14 – En (a), structure de blocs en représentation cartésienne; En (b), les arêtes sont coloriées selon leur nombre d’hexaèdres incidents : noir=1, bleu=2, vert=3 et rouge=4; En (c), les faces sont coloriées selon leur nombre d’hexaèdres incidents : bleu=1, rouge=2.

$F3$  avec deux hexaèdres incidents. Le tableau 5.16 est la combinaison de toutes les classifications (face, arête) possibles. On y voit que si un couple (face, arête) rentre dans les cas n°0,2,3,4,5 ou 7 alors l’arête sera ajoutée au front. Sinon elle ne sera pas ajoutée. Les cas n°3 et 4 sont quant à eux impossibles. Le front d’arêtes est mis à jour de manière naïve (nous traitons plusieurs fois une même arête) après chaque ajout d’une face à la sélection. De ces tableaux, nous déduisons l’algorithme 11 utilisé dans la construction de l’ensemble des candidats 12. Pour sa part, l’algorithme 10 permet de récupérer le nombre de faces faisant partie d’une solution autour d’une arête.

Ensemble	Classification	nombre de bloc incident
E0	concavité	3
E1	convexité	1
E2	surface	2
E3	volume	4

FIGURE 5.15 – Tableau de classification pour une arête.

### Mise à jour des phéromones

La mise à jour des phéromones placées sur les faces s’effectue de la même manière qu’en dimension 2. Les solutions sont triées selon leur qualité  $Q(S)$  et les  $k$  meilleures sont utilisées pour déposer des phéromones supplémentaires sur les faces qui les composent.

## 5.4 . Résultats expérimentaux en dimension 3

n°	$cf(f)$	$cf(a)$	Classification $f$	Classification $a$	Ajouter dans front	Existence
0	2	0	surface	concavité	Oui	Oui
1	2	1	surface	convexité	Non	Oui
2	2	2	surface	surface	Oui	Oui
3	2	3	surface	volume	Non	Non
4	3	0	volume	concavité	Oui	Oui
5	3	1	volume	convexité	Non	Non
6	3	2	volume	surface	Non	Oui
7	3	3	volume	volume	Oui	Oui

FIGURE 5.16 – Tableau de classification pour un couple face F et arête A.

---

**Algorithme 10 :** ns(Arête a)

---

**Data :** Arête  $a$

```

1 début
2    $cpt \leftarrow 0$  ;
3   Pour  $f \in Faces(a)$  faire
4     Si  $f \in S$  alors
5        $cpt \leftarrow cpt + 1$  ;
6     fin
7   fin
8   retourner  $cpt$  ;
9 fin

```

---

---

**Algorithme 11** : ajouterFront(Arête a,Face f)

---

```
1 début
2   Si  $cf(a) = 1 \wedge cf(f) = 0$  alors
3     | retourner Vrai ;
4   sinon si  $cf(a) = 1 \wedge cf(f) = 1$  alors
5     | retourner Faux ;
6   sinon si  $cf(a) = 1 \wedge cf(f) = 2$  alors
7     | retourner Vrai ;
8   sinon si  $cf(a) = 1 \wedge cf(f) = 3$  alors
9     | retourner Faux ;
10  sinon si  $cf(a) = 2 \wedge cf(f) = 0$  alors
11    | retourner Vrai ;
12  sinon si  $cf(a) = 2 \wedge cf(f) = 1$  alors
13    | retourner Faux ;
14  sinon si  $cf(a) = 2 \wedge cf(f) = 2$  alors
15    | retourner Faux ;
16  sinon si  $cf(a) = 2 \wedge cf(f) = 3$  alors
17    | retourner Vrai ;
18 fin
```

---

Cet algorithme dispose de nombreux paramètres, nous avons décidé d'effectuer une recherche sur certains d'entre eux afin d'identifier leur impact sur la convergence de l'algorithme ainsi que sur la vitesse d'exécution. De plus nous avons lancé l'algorithme sur plusieurs formes afin d'observer le comportement de celui-ci dans différentes situations pour lesquels nous avons une intuition du résultat à obtenir.

#### 5.4.1 . Recherche de paramètres "optimaux"

Une partie importante du travail d'expérimentation a consisté à évaluer quelles sont les "bonnes" valeurs à attribuer aux paramètres de notre méta-heuristique. Nous essayons ici d'expliquer le choix de certaines de ces valeurs.

Nous nous intéressons dans cette section à l'impact du nombre de fourmis  $m$ , du facteur de l'heuristique  $\beta$  et du nombre de fourmis élitistes  $e$ . Nous considérons pour cela le modèle cube5x5x5 de la figure 5.17.a. Nous fixons dans ce cas  $\lambda = 0.1$  et  $A = 30$  car durant les exécutions, la qualité des sélections obtenues étaient trop proches. La solution optimale obtenue est illustrée sur la figure 5.17.b. Elle correspond bien à la solution souhaitée par un utilisateur.

En faisant varier le nombre de fourmis par itération  $m \in \{10, 25, 50, 75\}$ , le pourcentage de fourmis élitistes  $e \in \{0.01, 0.25, 0.5, 0.75, 1.0\}$  avec  $nbe = 1$  si  $m * e < 0$  et le facteur heuristique  $\beta \in \{0, 1\}$ . Nous fixons les autres paramètres en nous inspirant de l'utilisation de l'optimisation par colonies de fourmis pour d'autres problèmes combinatoires :  $\rho = 0.5$ ,  $\tau_{min} = 0.1$ ,  $\tau_{ini} = 1.0$ . Nous assignons  $\tau_{max} = \infty$

---

**Algorithme 12 : *candidats()***

---

**Données :**  $S$  : ensemble de faces dans la solution

**Résultat :** Ensemble de candidats  $C$

```
1 début
2   Pour  $f \in S$  faire
3     Pour  $a \in \text{Aretes}(f)$  faire
4       Si  $ns(a) = 1 \wedge \text{ajouterFront}(a, f)$  alors
5          $bnd(a) \leftarrow ns(a)$  ;
6       fin
7     fin
8   fin
9    $C \leftarrow \emptyset$  ;
10  Pour  $a \in A$  faire
11    Si  $bnd(a) = 1$  alors
12      Pour  $f \in \text{Face}(a)$  faire
13        Si  $\forall a_f \in \text{Arete}(f), a_f \notin S$  alors
14           $C \leftarrow C \cup a$  ;
15        fin
16      fin
17    fin
18  fin
19  retourner  $C$  ;
20 fin
```

---

---

**Algorithme 13** : Algorithme général en dimension 3.

---

**Données :**

```
1 début
2   Initialiser phéromones et paramètres ;
3   Initialiser  $S_k$  avec  $S_{ini}$  ;
4   Répéter
5     Pour chaque fourmis  $k \in [1; m]$  construire une sélection faire
6        $C \leftarrow \text{candidats}()$  ;
7       Tant que  $C \neq \emptyset$  faire
8         Choisir une face  $f \in C$  avec la probabilité ;
9         
$$p(f_i, S_k) = \frac{[\tau_{factor}(f_i, S_k)]^\alpha \cdot [\eta_{factor}(f_i, S_k)]^\beta}{\sum_{a_j \in \text{candidats}} [\tau_{factor}(f_j, S_k)]^\alpha \cdot [\eta_{factor}(f_j, S_k)]^\beta}$$
 ;
10         $C \leftarrow \text{candidats}()$  ;
11      fin
12    fin
13    Trier les solutions par ordre décroissant en fonction de ;
14     $Q(S) = Z * (Ar_{part} + \lambda * El_{part})$  ;
15    Pour chaque fourmi  $k \in [1; e]$  faire
16      Pour chaque face  $f \in S$  faire
17         $\tau(f_i) \leftarrow \tau(f_i) * \rho + \frac{1}{1 + Q(S_{Best}) - Q(S_k)}$  ;
18      fin
19    fin
20  jusqu'à  $itermax$  atteint ou une qualité  $Q(S)$  convient;
21 fin
```

---



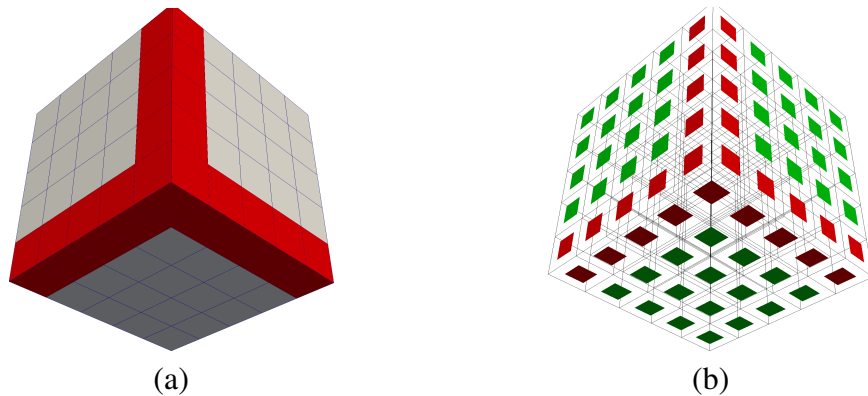


FIGURE 5.17 – (a) Cube5x5x5 avec des contraintes au bord (b) Meilleure solution pour  $\lambda = 0.1$  et  $A = 30$ . La qualité de cette solution est de  $Q(S) = 32.68$ .

et  $\alpha \in \{0, 1\}$ . Afin de suivre l'évolution des solutions construites par nos fourmis nous avons utilisé plusieurs mesures : la meilleure qualité  $Q(S)$  noté *Best* à chaque itération toutes exécutions confondues, les quartiles  $Q1$  et  $Q3$ , la médiane *Med*, la moyenne *Avg* avec  $N$  le nombre de solutions. Les valeurs de  $Q1$ ,  $Q3$  et *Med* sont calculées comme suit :

$$Q1 = \frac{N + 3}{4}; Q3 = \frac{3N + 1}{4}, \text{ et } Med = \frac{N + 1}{2}.$$

Le tableau 5.2 récapitule les résultats des expérimentations effectuées. Chaque ligne correspond à 10 exécutions du programme avec les mêmes paramètres. La colonne « index » est la colonne utilisée pour identifier chaque ligne. Certaines de ces lignes sont en gras car ce sont celles dont nous avons tracé les courbes de convergence. La colonne « temps » correspond à la durée totale pour les dix exécutions. Les durées en gras sont celles où  $\beta = 0$ , nous remarquons que le temps est supérieur à chaque fois en comparaison avec  $\beta = 1$ . La colonne « optimum » indique le nombre de fois que la meilleure solution à été trouvée. Ces expérimentations ont été effectuées avec un ordinateur portable disposant d'un processeur Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz. Le programme s'exécute à ce jour séquentiellement.

### Modification du paramètre $\beta$

Nous observons maintenant plus particulièrement l'allure des courbes lors des exécutions des lignes 3-4, 13-14, 23-24, et 33-34 du tableau 5.2 . Nous avons ici fait varier le facteur  $\beta$  afin de déterminer si il avait un impact sur les exécutions. Sur la figure 5.18.a est tracée la courbe de convergence de la population de  $m$  fourmis durant 10 exécutions. La courbe montre que des individus proches de la valeur optimum de  $Q(S_{Best}) = 32.68$  ont été découverts. Il s'agit des solutions proches de la meilleure qualité présentes sur la figure 5.18.a. L'algorithme n'a trouvé l'optimum pour aucune de ces exécutions. Pourtant quelques unes s'en rapprochent. Sur la figure 5.19 sont

Index	$m$	nbe	$\beta$	temps (h :m :s)	Optimum
1	10	0.01	0	<b>00 :10 :54</b>	0
2	10	0.01	1	00 :08 :12	7
<b>3</b>	10	0.25	0	<b>00 :10 :12</b>	0
<b>4</b>	10	0.25	1	00 :05 :39	10
5	10	0.5	0	<b>00 :09 :43</b>	1
6	10	0.5	1	00 :05 :14	10
7	10	0.75	0	<b>00 :09 :41</b>	1
8	10	0.75	1	00 :05 :16	10
9	10	1.0	0	<b>00 :10 :18</b>	0
10	10	1.0	1	00 :05 :16	10
11	25	0.01	0	<b>00 :26 :38</b>	0
12	25	0.01	1	00 :15 :10	10
<b>13</b>	25	0.25	0	<b>00 :21 :25</b>	6
<b>14</b>	25	0.25	1	00 :11 :27	10
15	25	0.5	0	<b>00 :19 :12</b>	10
16	25	0.5	1	00 :11 :07	10
17	25	0.75	0	<b>00 :22 :20</b>	9
18	25	0.75	1	00 :11 :33	10
19	25	1.0	0	<b>00 :25 :32</b>	0
20	25	1.0	1	00 :11 :41	10
21	50	0.01	0	<b>00 :50 :29</b>	1
22	50	0.01	1	00 :29 :09	10
<b>23</b>	50	0.25	0	<b>00 :31 :30</b>	10
<b>24</b>	50	0.25	1	00 :20 :08	10
25	50	0.5	0	<b>00 :35 :34</b>	10
26	50	0.5	1	00 :21 :13	10
27	50	0.75	0	<b>00 :41 :58</b>	10
28	50	0.75	1	00 :22 :05	10
29	50	1.0	0	<b>00 :50 :38</b>	0
30	50	1.0	1	00 :23 :22	10
31	75	0.01	0	<b>01 :14 :46</b>	4
32	75	0.01	1	00 :43 :39	10
<b>33</b>	75	0.25	0	<b>00 :45 :52</b>	10
<b>34</b>	75	0.25	1	00 :30 :02	10
35	75	0.5	0	<b>00 :53 :03</b>	10
36	75	0.5	1	00 :32 :52	10
37	75	0.75	0	<b>01 :08 :01</b>	10
38	75	0.75	1	00 :37 :36	10
39	75	1.0	0	<b>01 :20 :27</b>	0
40	75	1.0	1	00 :33 :45	10

TABLE 5.2 – Tableau récapitulatif des résultats

listées les meilleures solutions pour chacune des exécutions de l'algorithme. On notera en particulier que les solutions des figures 5.19.a.g sont très proches de l'optimum.

Considérons maintenant d'autres points singuliers de l'analyse de l'impact du paramètre  $\beta$  :

- Tout d'abord, à la ligne 4 du tableau 5.2, nous obtenons avec l'aide de l'heuristique 10 sélections identiques à l'optimum. Ceci est normal selon nous étant donnée la forme considérée ici.
- Ensuite pour les lignes 13-14 du tableau 5.2 (voir la figure 5.20), nous avons augmenté le nombre de fourmis  $m$  à 50. Lorsque  $\beta = 0$  nous obtenons 7 fois sur 10 l'optimum mais on découvre l'optimum pour une exécution à l'itération 80 alors que avec  $\beta = 1$  l'optimum est découvert pour la première fois à la troisième itération. On peut en déduire que nos colonies de fourmis convergent vers une solution à l'aide uniquement des phéromones, mais que l'heuristique aide grandement à la convergence de la méthode.
- Sur les lignes 23-24 du tableau 5.2 (voir la figure 5.21), les sélections optimales sont découvertes à chaque exécution. La population converge lentement vers des meilleures qualités et avec  $\beta = 0$  nous obtenons l'optimum lors de l'itération 17 pour une exécution contre la quatrième pour  $\beta = 1$ .
- Sur les lignes 33-34 du tableau 5.2 (voir la figure 5.22), nous constatons les mêmes tendances que lorsque le nombre de fourmis est moins élevé.

### **Modification du nombre $m$ de fourmis**

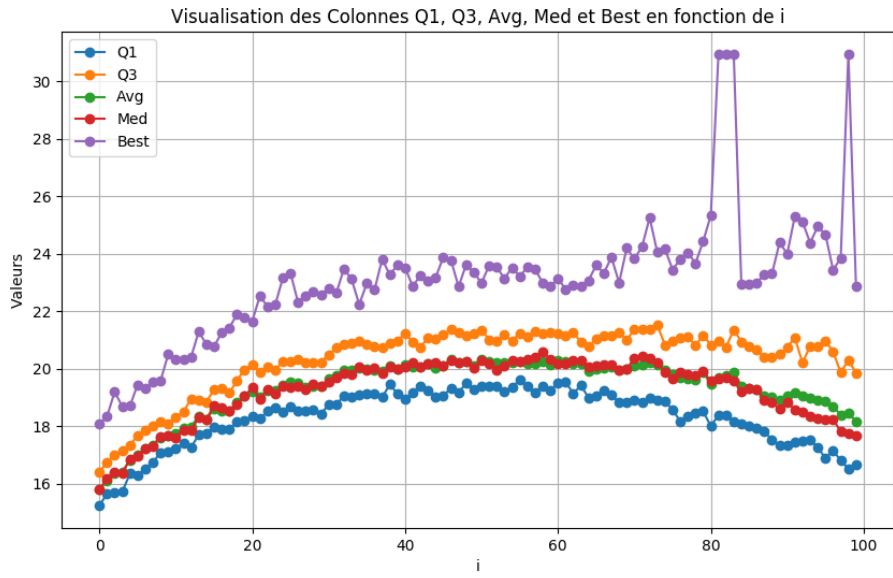
Nous souhaitons observer l'impact du nombre de fourmis sur la convergence de l'algorithme. Pour cela concentrons nous sur les colonne où  $\beta = 0$ , c'est-à-dire en conservant uniquement le facteur phéromones et avec un  $nbe \in [0.25, 0.75]$  (lignes 3, 5, et 7). Nous remarquons que pour  $m = 10$  nous ne convergions pas vers la meilleure solution à chaque exécution. Alors que si nous augmentons le nombre de fourmis à  $m = 25$ , on a une amélioration significative (voir les lignes 13, 15, et 17). Ces résultats sont encore meilleurs pour  $m = 50$  et  $m = 75$  où l'algorithme converge vers la meilleure solution.

### **Modification du nombre $e$ de fourmis élitistes**

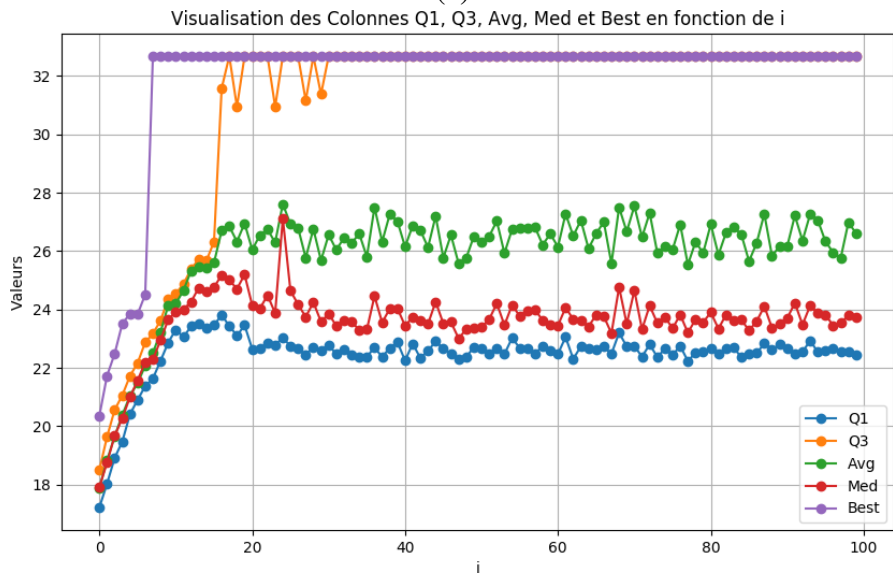
Observons maintenant l'impact du nombre de fourmis élitiste  $e$ . En fixant de nouveau  $\beta = 0$ , nous pouvons constater que l'algorithme converge moins souvent pour la valeur  $nbe = 0.01$ , c'est à dire quand le nombre de fourmis élitistes  $e = 1$  (voir les lignes 1, 11, 21 et 31). Nous remarquons aussi que pour  $nbe = 1$ , nous ne convergions pas vers une solution (voir les lignes 9, 19 et 29).

#### **5.4.2 . Expérimentations sur plusieurs géométries et structures de blocs**

Nous avons exécuté notre algorithme d'optimisation par colonies de fourmis pour la recherche de nappes 3D sur plusieurs structures de blocs dont celles illustrées sur



(a)



(b)

FIGURE 5.18 – Courbes de convergence des populations de fourmis de la ligne 3 du tableau 5.2 en (a) et de la ligne 4 du tableau 5.2 en (b).

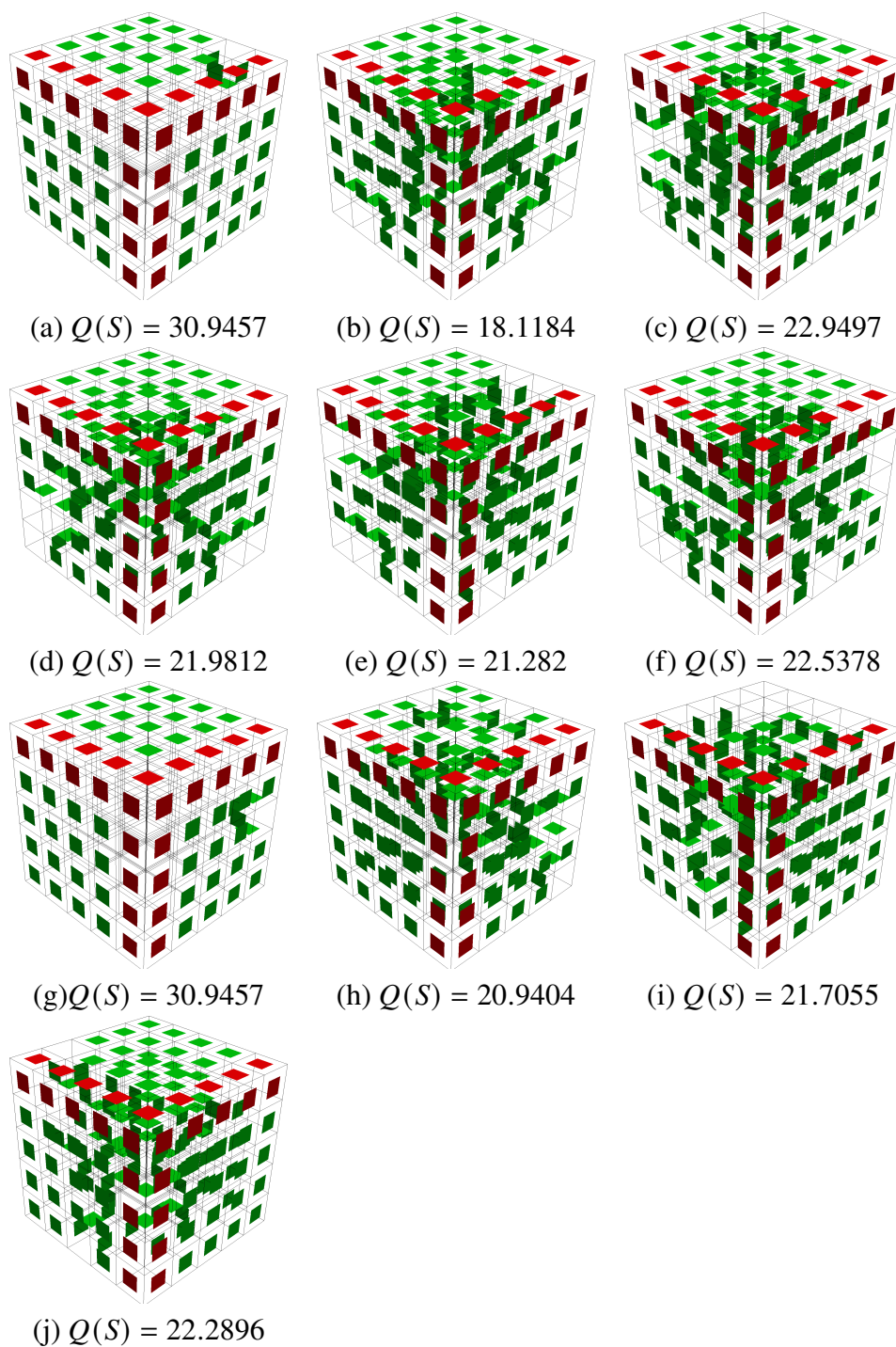
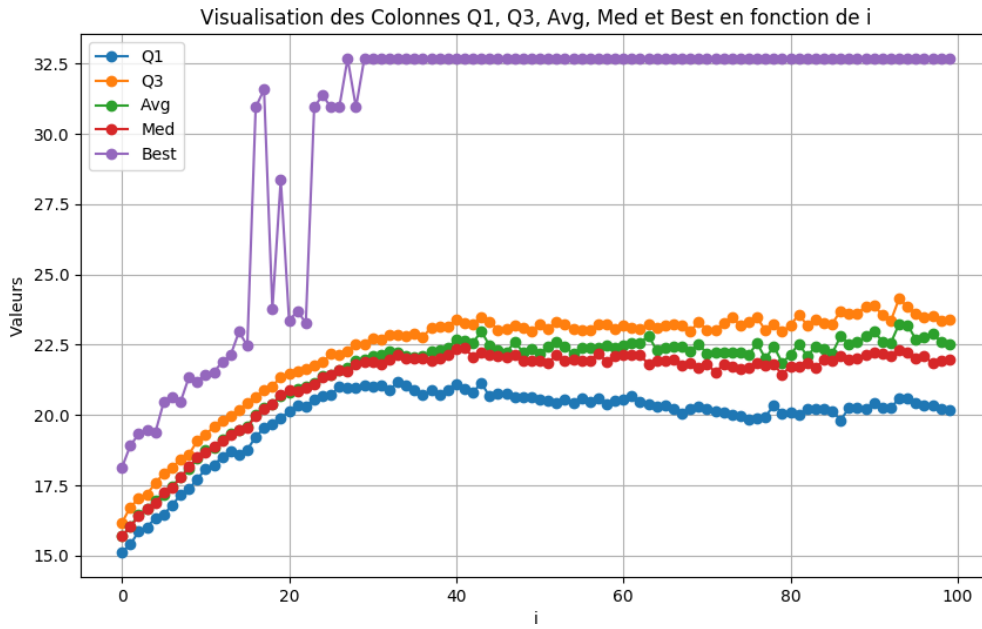
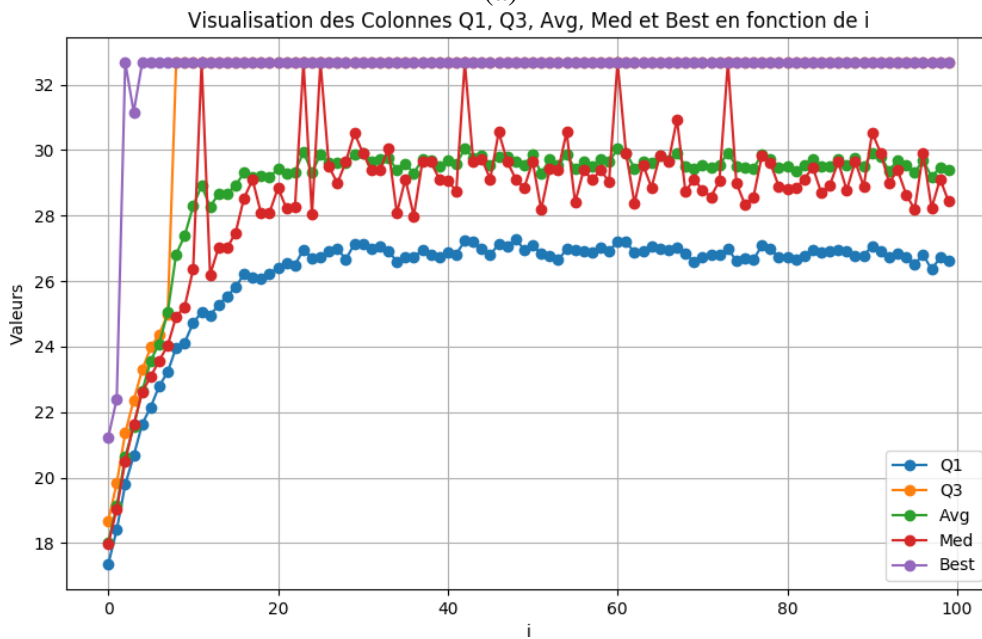


FIGURE 5.19 – Meilleure sélection découverte pour chacune des dix exécutions. A l'indice 3 du tableau récapitulatif.

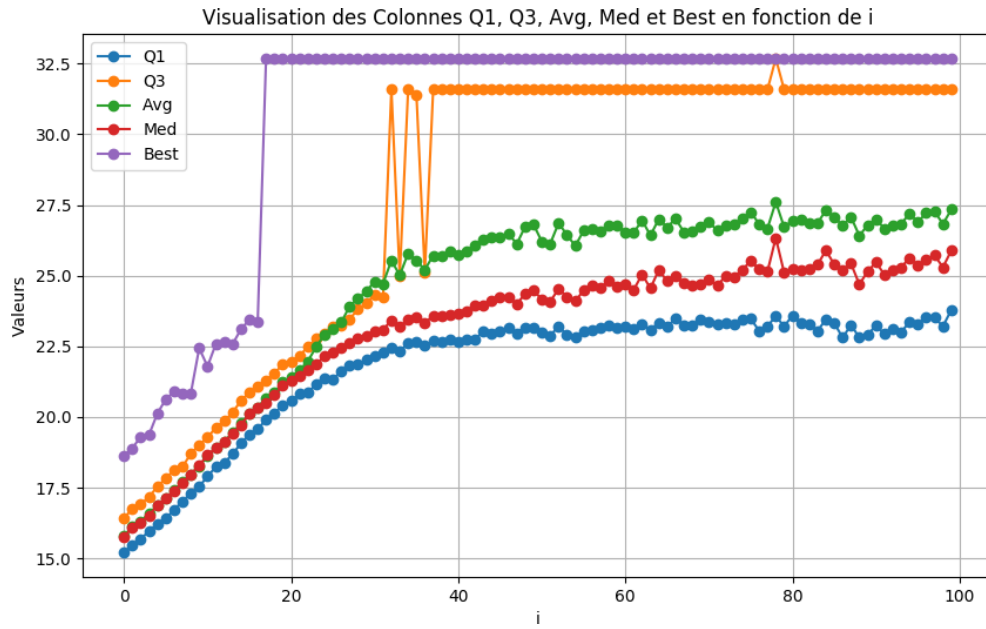


(a)

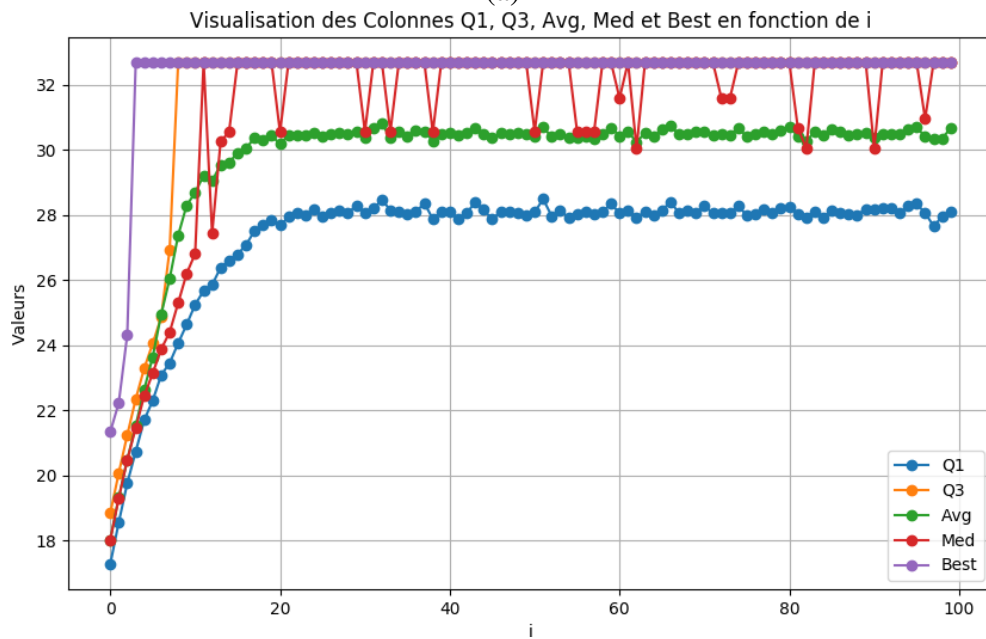


(b)

FIGURE 5.20 – Courbes de convergence des populations de fourmis de la ligne 13 du tableau 5.2 en (a) et de la ligne 14 du tableau 5.2 en (b).

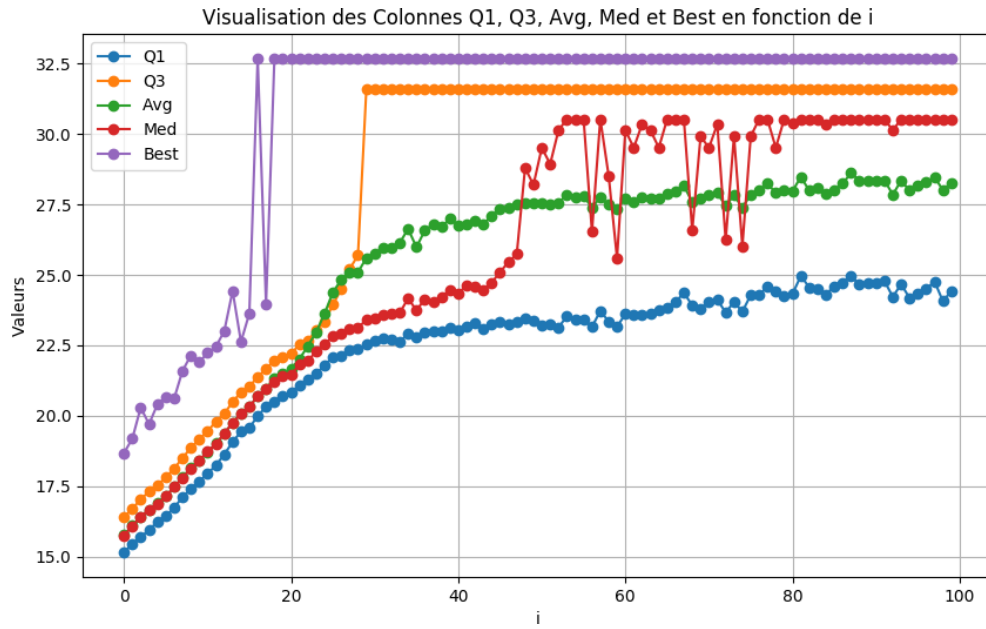


(a)

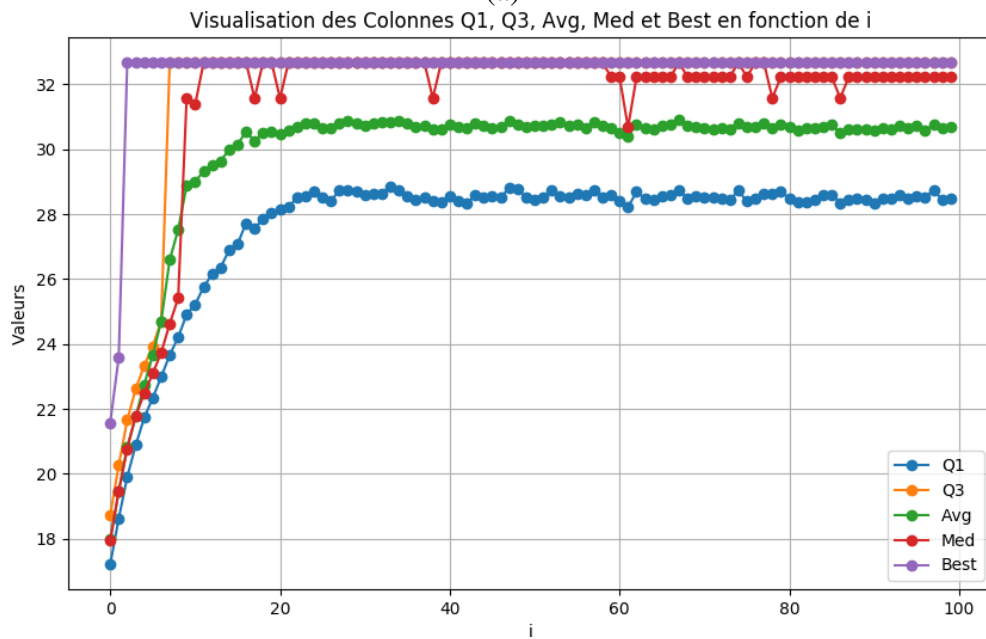


(b)

FIGURE 5.21 – Courbes de convergence des populations de fourmis de la ligne 23 du tableau 5.2 en (a) et de la ligne 24 du tableau 5.2 en (b).



(a)



(b)

FIGURE 5.22 – Courbes de convergence des populations de fourmis de la ligne 33 du tableau 5.2 en (a) et de la ligne 34 du tableau 5.2 en (b).



les figures 5.23 et 5.24. Notons que les structures de la figure 5.23 (de *a* à *d*) ont peu de faces tandis que celles des figures 5.23.*e* et *f* sont proches de configurations 2D. Les figures 5.24.*i* – *j* sont plus complexes en termes de nombre de blocs hexaédriques.

Les paramètres utilisés pour les expérimentations menées sur ces modèles sont  $itermax = 100$ ,  $m = 30$ ,  $\tau_{min} = 0.1$ ,  $\tau_{max} = \infty$ ,  $\rho = 0.5$ ,  $\alpha = 1$ ,  $\beta = 1$ ,  $e = 6$ , et  $\tau_{ini} = 1$  (valeurs de paramètres déduites des expérimentations précédentes). Le tableau 5.3 fournit des caractéristiques de ces expérimentations, en particulier la dernière colonne du tableau indique le temps d'exécution total de l'algorithme sur les 100 itérations. Rappelons que nous pourrions arrêter l'exécution, lorsque l'algorithme converge, lorsqu'une sélection acceptable est trouvée, si nous détectons une stagnation au niveau de la qualité des sélections ou tout simplement après une durée (passée) trop importante.

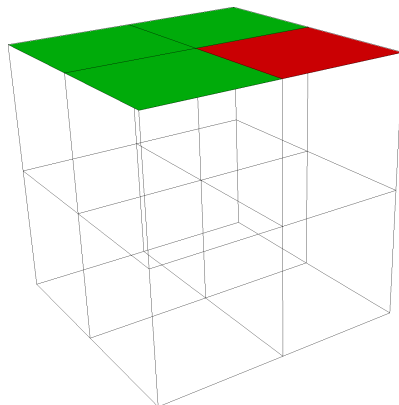
Fig.	Modèle	#H	#F	#E	#N	temps (h :m :s)
5.23.(a)	cube2x2x2surface1	8	36	54	27	00 :00 :01
5.23.(b)	cube2x2x2surface3	8	36	54	27	00 :00 :02
5.23.(c)	colonesimple	20	88	126	125	00 :00 :05
5.23.(d)	siege3D	24	100	138	125	00 :00 :07
5.23.(e)	1exterieur1interieur	48	208	289	729	00 :00 :07
5.23.(f)	2exterieurinterieur	56	240	329	1331	00 :00 :09
5.24.(g)	cube3x3x3	27	108	144	64	00 :00 :10
5.24.(h)	colonne2D	80	340	461	1331	00 :00 :14
5.24.(i)	cube5x5x5	125	450	540	216	00 :01 :23
5.24.(j)	hc2	462	1655	1960	4096	00 :18 :48

TABLE 5.3 – Récapitulatif des modèles.

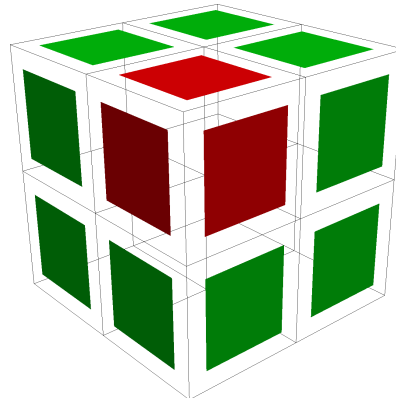
### 5.4.3 . Limites observées expérimentalement

Nous avons observé plusieurs limitations de notre approche. Nous utilisons une heuristique afin d'influencer les choix effectués lors de la construction des sélections par les fourmis. Ceci permet d'accélérer la convergence mais cela pourrait ne pas être le cas pour toutes les valeurs de  $\lambda$ . Nous ne garantissons pas que la meilleure sélection obtenue sera valide à la fin de l'exécution de l'algorithme. Nous ne garantissons pas non plus la convergence sur une exécution de l'algorithme même si en pratique nous avons eu des cas de non convergence que dans les modèles ayant un grand nombre de faces (>1500) en prenant les mêmes paramètres que ceux de la section 5.4.2.

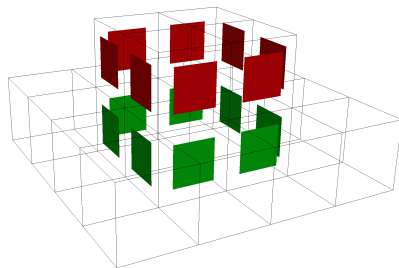
Nous avons fait le choix que les feuillets sélectionnés par l'algorithme ne puissent pas s'auto-intersecter ni s'auto-toucher alors que l'insertion de feuillets décrite au chapitre 3 le permet. C'est une option à considérer à l'avenir. Enfin, notons que le temps de calcul augmente selon la complexité de la structure de blocs. Ceci est normal mais peut limiter aujourd'hui l'utilisation de cette approche.



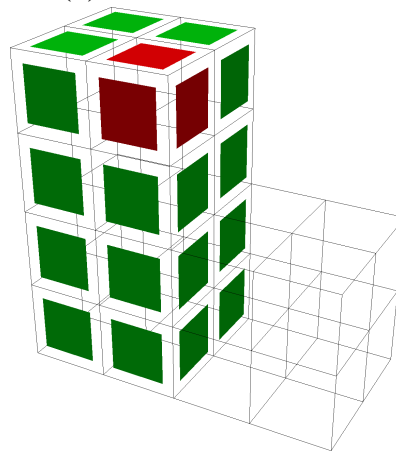
(a) cube2x2x2surface1



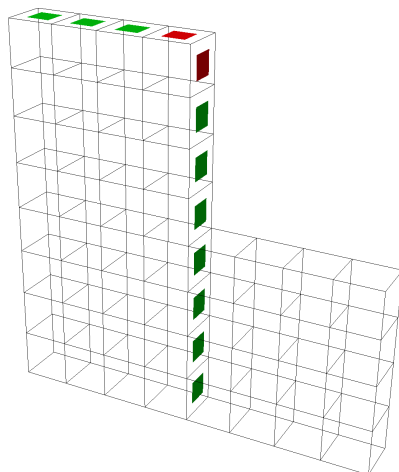
(b) cube2x2x2surface3



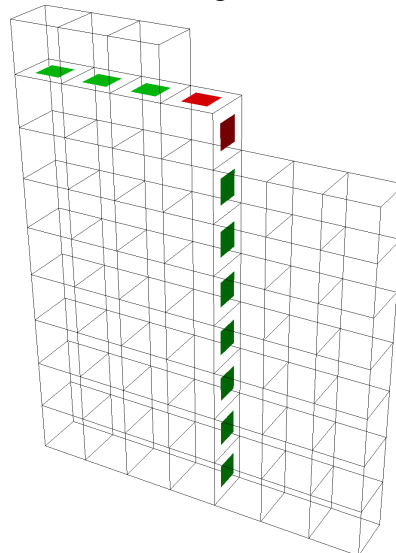
(c) colonnesimple



(d) siege3D

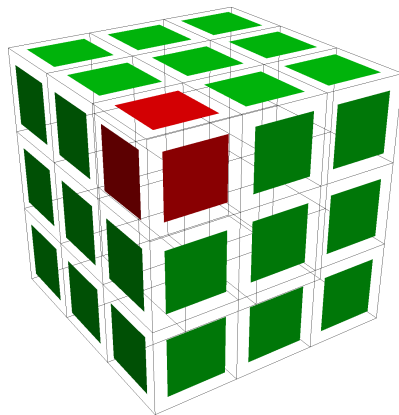


(e) 1exterieur1interieur

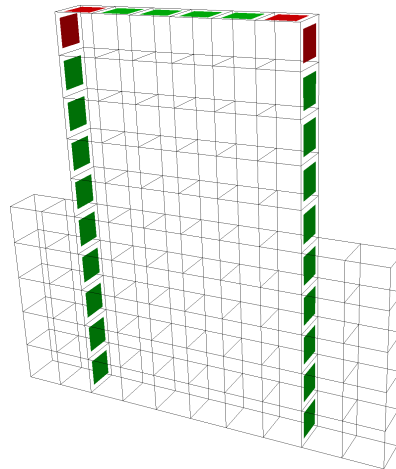


(f) 2exterieurinterieur

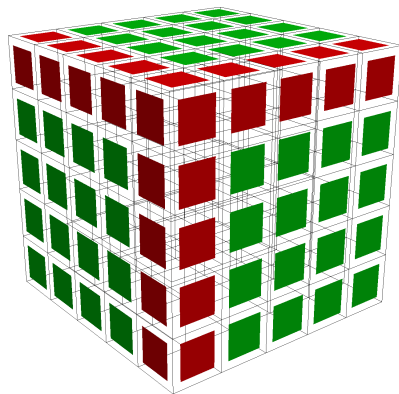
FIGURE 5.23 – Différents exemples de sélections "optimales" de faces obtenues pour résoudre les contraintes de criticité sur les arêtes incidentes aux faces rouges formant la sélection initiale (1/2).



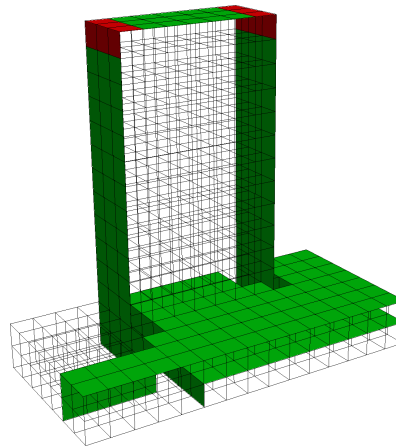
(g) cube3x3x3



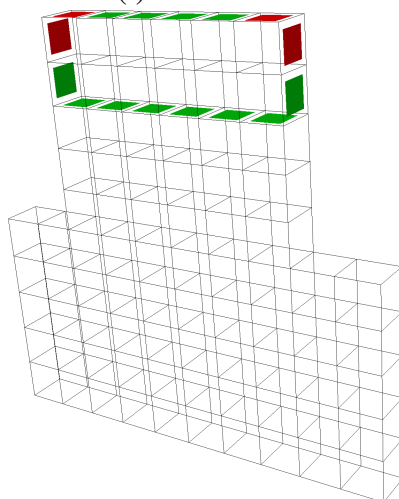
(h) colonne2D



(i) cube5x5x5



(j) hc2



(k) colonne2D avec  $\lambda = 2$

FIGURE 5.24 – Différents exemples de sélections "optimales" de faces obtenues pour résoudre les contraintes de criticité sur les arêtes incidentes aux faces rouges formant la sélection initiale (2/2).

## 5.5 . Conclusion et perspectives

Dans ce chapitre, nous avons mis en place un algorithme utilisant une méthode d'optimisation par colonie de fourmis pour résoudre un problème de sélection d'un sous-ensemble de faces permettant l'insertion d'un feuillet dans une structure de blocs hexaédriques. Nous avons décrit l'algorithme et montré par des expérimentations que notre méthode convergeait sans avoir besoin d'une heuristique particulière bien que celle-ci augmente considérablement la vitesse de calcul et accroît la convergence pour un paramètre  $\lambda$  fixé. Nous avons ensuite donné en entrée à notre algorithme plusieurs structures de blocs de type polycube qui pourraient être aussi complexes que celles devant être optimisées par les ingénieurs.

Dans les axes d'améliorations que nous pensons utiles d'approfondir, il serait possible de considérer une approche topologique pour détecter les tournant d'arêtes, ce qui nous permettrait d'étendre l'approche à d'autres structures de blocs que celles issues de polycubes, afin d'éventuellement faire des insertions successives sur une même structure. L'amélioration de l'implémentation pourrait passer par l'ajout d'une exécution parallèle de la construction des solutions.

Dans le problème du voyageur de commerce [51], au début de chaque itération une fourmi est placée sur chaque ville. Par contre, ce n'est pas le cas des problèmes de sélection d'un sous-ensemble [49] comme celui du sac à dos. En effet, dans ce cas, au début de la construction de chacune des sous-sélections, une face est aléatoirement choisie. Pour notre part les fourmis démarrent toutes du front d'arêtes de la sélection initiale. Nous nous demandons si l'algorithme pourrait converger plus rapidement en changeant cette politique comme dans [49] ou [51].



## 6 - Conclusion

Durant ce travail de thèse, nous nous sommes intéressés à la représentation et la manipulation de structures de blocs hexaédriques afin de les optimiser, au sens d'améliorer leur qualité. Ces structures de blocs sont aujourd'hui particulièrement importantes car elles sont nécessaires pour de nombreux codes de simulation numérique, et impossible à générer automatiquement dans la majorité des cas.

La première partie de nos travaux, présentée au chapitre 3, a consisté à définir et représenter formellement les structures de blocs hexaédriques ainsi que les opérations associées à ces blocs. Pour cela, nous avons défini de manière formelle la notion de maillage structuré par blocs en utilisant le modèle des cartes généralisées. Ce modèle nous a permis de représenter la topologie et la géométrie de ces structures en conservant une relation de classification forte vers les modèles géométriques de type CAO représentés par leur bords. D'un point de vue opérations, nous avons défini et implémenté des opérations agissant sur les feuillets des structures de blocs. En particulier, nous avons tenu compte des configurations complexes des feuillets auto-touchants et auto-intersectants, et de la relation de classification géométrique.

Afin d'optimiser des structures de blocs en dimension 2, nous avons proposé au chapitre 4 un algorithme multi-agents pour résoudre un problème d'alignement au bord en appliquant des motifs locaux de modification topologique et géométrique. Nous avons maintenu la cohérence de la structure globale grâce à un agent coordinateur et nous avons exécuté l'algorithme sur de nombreux cas tests, améliorant à chaque fois la structure de blocs manipulée.

Enfin, au chapitre 5, nous avons adapté la métaheuristique d'optimisation par colonie de fourmis au problème de sélection d'un sous-ensemble de faces pour effectuer une insertion globale d'un feuillet en dimension deux et trois. Nous avons effectué des expérimentations afin de trouver un ensemble de paramètres maximisant l'efficacité de l'algorithme en terme d'obtention de l'optimum et de temps d'exécution. Puis nous avons utilisés ces paramètres sur un ensemble de structures de blocs.

### 6.1 . Perspectives de travail

Les perspectives sont relatives à deux aspects : d'une part la représentation des structures de blocs à l'aide des cartes généralisées, et d'autre part l'optimisation de structures de blocs.

#### 6.1.1 . Opérations robustes sur structures de blocs hexaédriques

Nous avons pu voir au chapitre 3 que les structures de blocs hexaédriques ainsi que certaines opérations sur ces structures pouvaient être représentées à l'aide de cartes

généralisées. Dans la continuité de ces travaux, deux axes nous paraissent importants à étudier.

Tout d'abord définir formellement de nouvelles opérations sur les structures de blocs, et prouver le maintien de propriétés attendues de ces opérations. On pense en particulier à la capacité de garantir que tous les blocs insérés sont bien hexaédriques. Actuellement, nous avons seulement pu le valider *a posteriori*.

L'ensemble des travaux présentés dans ce manuscrit a été implémenté en s'appuyant sur l'implémentation des cartes généralisées disponible dans la bibliothèque CGAL [42, 43]. L'une des forces de cette implémentation est de bien mettre en œuvre la séparation des aspects géométriques et topologiques. Ainsi, l'implémentation des opérations de couture et découture de brins met à jour automatiquement les attributs associés aux cellules de la structure si des opérateurs de *fusion (fuse)* et *séparation (splitting)* sont bien fournis pour chaque type de cellule. Dans l'optique de considérer des blocs hexaédriques courbes et non plus linéaires, nous envisageons de définir spécifiquement des opérateurs pour les arêtes, faces, et blocs, en considérant des plongements d'ordre supérieur, où nous utiliserons des représentations de type Bézier ou Spline.

### 6.1.2 . Vers des systèmes multi-agents plus intelligents

Le système d'agents réflexes supervisés par un agent coordinateur proposé au chapitre 4 nous a permis d'obtenir des résultats satisfaisants pour des structures de blocs 2D. Le passage en 3D nous semble possible mais nécessite au moins deux ingrédients majeurs :

1. Tout d'abord, il faut définir la famille de motifs de transformation de blocs qui pourrait exister pour gérer les configurations d'arêtes critiques autour d'un bloc, et les propagations de non-conformité. Nous pensons nécessaire pour cela de définir un système de règles nous permettant de générer l'ensemble des configurations possibles.
2. Ensuite, il sera question de munir l'agent coordinateur d'une intelligence lui permettant de sélectionner les priorités entre actions. Contrairement au cas 2D, il ne nous semble pas possible de définir un comportement déterministe pour cet agent. Nous pensons par contre qu'il est possible d'utiliser l'apprentissage par renforcement pour cela.

Un autre aspect à étudier selon nous est la capacité à imposer des solutions sous-optimales à certains agents locaux pour assurer une meilleure qualité globale. Ceci induit que les agents locaux ne fournissent plus une solution unique à l'agent coordinateur. Cela pose aussi la question de la notion de qualité de structure de blocs, au sens global versus local. Dans pareil cas, la question d'une autre forme de coordination entre agents, telle que l'éco-résolution, serait alors envisageable.

Dans le chapitre 5, nous avons proposé un algorithme de colonies de fourmis pour définir une nappe "optimale" de faces qui permettrait d'insérer un feuillet dans

la structure de blocs pour résoudre des problèmes de criticité au bord. Si les résultats obtenus ont été pour nous satisfaisant en matière de solutions obtenues, plusieurs aspects peuvent être améliorés selon nous :

1. Tout d'abord, comme toute méta-heuristique appliqué à un problème spécifique (ici la sélection d'une nappe de face "optimale"), il existe de nombreux hyperparamètres dont la valeur doit être fixée. Nous avons trouvé des valeurs qui nous semblent adéquates, mais cela uniquement de manière expérimentales et sur un nombre restreint de cas. Une étude plus complète devrait être menée pour valider ou changer ces valeurs. On peut aussi imaginer que tout ou partie de ces paramètres soit des inconnues du problème et soit découverte au cours de simulations successives.
2. Un objectif initial que nous souhaitions atteindre était de proposer à un utilisateur d'un outil interactif permettant de choisir entre plusieurs nappes solutions de qualités proches. Nous n'avons pas pu le faire mais nous pensons toujours que ces expérimentations seraient intéressantes dans le futur.
3. Pour permettre de poursuivre le travail dans les deux directions précédemment énoncées, un travail d'implémentation doit être réalisé pour améliorer les performances de l'algorithme. En effet, l'implémentation est pour le moment assez naïve et non parallélisée. La parallélisation est ici naturelle (fourmis cherchant des solutions indépendantes).





## Bibliographie

- [1] Cubit, “Sandia national laboratories : CUBIT geometry and mesh generation toolkit.”
- [2] M. Smith, *ABAQUS/Standard User’s Manual, Version 6.9*. United States : Dassault Systèmes Simulia Corp, 2009.
- [3] ANSYS, “Ansys fluent - cfd software | ansys,” 2016.
- [4] N. Pietroni, M. Campen, A. Sheffer, G. Cherchi, D. Bommers, X. Gao, R. Scateni, F. Ledoux, J.-F. Remacle, and M. Livesu, “Hex-mesh generation and processing : A survey,” *ACM Trans. Graph.*, jul 2022. Just Accepted.
- [5] M. Mäntylä, *An Introduction to Solid Modeling*. USA : Computer Science Press, Inc., 1987.
- [6] T. D. Blacker, S. J. Owen, M. L. Staten, W. R. Quadros, B. Hanks, B. W. Clark, R. J. Meyers, C. Ernst, K. Merkley, R. Morris, C. McBride, C. J. Stimpson, M. Plooster, and S. Showman, “Cubit geometry and mesh generation toolkit 15.1 user documentation,” 2 2016.
- [7] P. Lienhardt, “Subdivisions of  $n$ -dimensional spaces and  $n$ -dimensional generalized maps,” in *Annual ACM Symposium on Computational Geometry*, pp. 228–236, 1989.
- [8] J.-F. Remacle and M. Shephard, “An algorithm oriented mesh database,” *International Journal for Numerical Methods in Engineering*, vol. 58, no. 2, 2003.
- [9] E. S. Seol, *FMDB : Flexible Distributed Mesh Database for Parallel Automated Adaptive Analysis*. PhD thesis, Rensselaer Polytechnic Institute, 2005.
- [10] R. V. Garimella, “Mesh data structure selection for mesh generation and fea applications,” in *International Journal for Numerical Methods in Engineering*, vol. 55, pp. 451–478, 2002.
- [11] R. V. Garimella, *MSTK : MeSh ToolKit, v 1.3 User’s manual*. Los Alamos National Laboratory, 2012. LA-UR-04-0878.
- [12] T. Tautges, C. Ernst, K. Merkley, R. Meyers, and C. Stimpson, “Mesh oriented database (moab),” 2005. <http://cubit.sandia.gov/cubit>.
- [13] F. Ledoux, Y. Bertrand, and J.-C. Weill, *Generic Mesh Data Structure in HPC Context*, vol. 26 of *Computation Technologies and Innovation Series*, ch. 3, pp. 49–80. Stirlingshire : Saxe-Coburg Publications, 2010.
- [14] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*. New-York : Springer-Verlag, 1987.
- [15] B. G. Baumgart, “A polyhedron representation for computer vision,” in *Proceedings of the May 19-22, 1975, National Computer Conference and Exposition*,

- AFIPS '75, (New York, NY, USA), p. 589–596, Association for Computing Machinery, 1975.
- [16] D. E. Muller and F. P. Preparata, “Finding the intersection of two convex polyhedra,” *Theor. Comput. Sci.*, vol. 7, pp. 217–236, 1978.
- [17] K. Weiler, “Edge-based data structures for solid modeling in curved-surface environments,” *IEEE Computer Graphics and Applications*, vol. 5, no. 1, pp. 21–40, 1985.
- [18] D. Sieger and M. Botsch, “Design, implementation, and evaluation of the surface\_mesh data structure,” in *IMR*, 2011.
- [19] P. Lienhardt, “N-dimensional generalized combinatorial maps and cellular quasi-manifolds,” *Int. J. Comput. Geom. Appl.*, vol. 4, pp. 275–324, 1994.
- [20] G. Damiani and P. Lienhardt, *Combinatorial Maps : Efficient Data Structures for Computer Graphics and Image Processing*. A K Peters/CRC Press, September 2014.
- [21] F. Protais, M. Reberol, N. Ray, E. Corman, F. Ledoux, and D. Sokolov, “Robust Quantization for Polycube Maps,” *Computer-Aided Design*, vol. Volume 150, Sept. 2022.
- [22] J. Gregson, A. Sheffer, and E. Zhang, “All-hex mesh generation via volumetric polycube deformation,” *Computer Graphics Forum (Special Issue of Symposium on Geometry Processing 2011)*, vol. 30, no. 5, 2011.
- [23] M. Livesu, N. Vining, A. Sheffer, J. Gregson, and R. Scateni *Transactions on Graphics (Proc. SIGGRAPH ASIA 2013)*, vol. 32, no. 6, 2013.
- [24] N. Ray, D. Sokolov, and B. Lévy, “Practical 3d frame field generation,” *ACM Trans. Graph.*, vol. 35, Nov. 2016.
- [25] H. Liu, P. Zhang, E. Chien, J. Solomon, and D. Bommes, “Singularity-constrained octahedral fields for hexahedral meshing,” *ACM Trans. Graph.*, vol. 37, July 2018.
- [26] N. Ray, D. Sokolov, M. Reberol, F. Ledoux, and B. Lévy, “Hex-dominant meshing : mind the gap !,” in *SPM 2018 - International Conference on Solid and Physical Modeling*, (Bilbao, Spain), June 2018.
- [27] R. Schneiders, “A grid-based algorithm for the generation of hexahedral element meshes,” *Engineering with Computers*, vol. 12, pp. 168–177, Sept. 1996.
- [28] L. Maréchal, “Advances in Octree-Based All-Hexahedral Mesh Generation : Handling Sharp Features,” in *Proceedings of the 18th International Meshing Roundtable* (B. W. Clark, ed.), pp. 65–84, Springer Berlin Heidelberg, 2009.
- [29] L. Maréchal, “All Hexahedral Boundary Layers Generation,” *Procedia Engineering*, vol. 163, pp. 5–19, Jan. 2016.
- [30] X. Gao, H. Shen, and D. Panozzo, “Feature Preserving Octree-Based Hexahedral Meshing,” *Computer Graphics Forum*, vol. 38, no. 5, pp. 135–149, 2019.

- [31] S. J. Owen, M. L. Staten, and M. C. Sorensen, "Parallel Hex Meshing from Volume Fractions," in *Proceedings of the 20th International Meshing Roundtable* (W. R. Quadros, ed.), pp. 161–178, Springer Berlin Heidelberg, 2012.
- [32] Y. J. Zhang, *Geometric Modeling and Mesh Generation from Scanned Images*. Apr. 2016.
- [33] N. Le Goff, F. Ledoux, and S. J. Owen, "Hexahedral mesh modification to preserve volume," *Computer-Aided Design*, vol. 105, pp. 42–54, Dec. 2018.
- [34] G. Cherchi, P. Alliez, R. Scateni, M. Lyon, and D. Bommès, "Selective Padding for Polycube-Based Hexahedral Meshing," *Computer Graphics Forum*, Jan. 2019.
- [35] J. H.-C. Lu, I. Song, W. R. Quadros, and K. Shimada, "Geometric reasoning in sketch-based volumetric decomposition framework for hexahedral meshing," *Engineering with Computers*, vol. 30, pp. 237–252, Apr. 2014.
- [36] D. R. White, S. Saigal, and S. J. Owen, "CCSweep : automatic decomposition of multi-sweep volumes," *Engineering with Computers*, vol. 20, pp. 222–236, Sept. 2004.
- [37] H. Wu and S. Gao, "Automatic Swept Volume Decomposition based on Sweep Directions Extraction for Hexahedral Meshing," *Procedia Engineering*, vol. 82, pp. 136–148, Jan. 2014.
- [38] K. Takayama, "Dual Sheet Meshing : An Interactive Approach to Robust Hexahedralization," *Computer Graphics Forum*, vol. 38, no. 2, pp. 37–48, 2019.   
\_eprint : <https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13617>.
- [39] S. Calderan, G. Hutzler, and F. Ledoux, "Dual-Based User-Guided Hexahedral Block Generation Using Frame Fields," Zenodo, 2019.
- [40] E. Brisson, "Representing geometric structures in d dimensions : Topology and order," in *Symposium on Computational Geometry*, pp. 218–227, 1989.
- [41] M. L. Staten, J. F. Shepherd, F. Ledoux, and K. Shimada, "Hexahedral mesh matching : Converting non-conforming hexahedral-to-hexahedral interfaces into conforming interfaces," *International Journal for Numerical Methods in Engineering*, vol. 82, no. 12, pp. 1475–1509, 2010.
- [42] The CGAL Project, *CGAL User and Reference Manual*. CGAL Editorial Board, 5.6 ed., 2023.
- [43] G. Damiand, "Generalized maps," in *CGAL User and Reference Manual*, CGAL Editorial Board, 5.6 ed., 2023.
- [44] H. Belhaouari, A. Arnould, P. Le Gall, and T. Bellet, "JERBOA : A Graph Transformation Library for Topology-Based Geometric Modeling," in *7th International Conference on Graph Transformation (ICGT 2014)* (H. Giese and B. König, eds.), vol. 8571, (York, United Kingdom), Springer, July 2014.
- [45] J. Ferber, *Les systèmes multi-agents : vers une intelligence collective*. Paris : InterEditions, 1995.

- [46] C. Delaye, J. Ferber, and E. Jacopin, “Approche inter-active de la résolution de problèmes : l’éco-résolution,” 01 1991.
- [47] A. Coloni, M. Dorigo, and V. Maniezzo, “Distributed optimization by ant colonies,” 01 1991.
- [48] G. D. Caro and M. Dorigo, “Antnet : Distributed stigmergetic control for communications networks,” *CoRR*, vol. abs/1105.5449, 2011.
- [49] C. Solnon and D. Bridge, “An ant colony optimization meta-heuristic for subset selection problems,” *Systems Engineering using Particle Swarm Optimization*, 01 2007.
- [50] M. Dorigo, V. Maniezzo, and A. Coloni, “Ant system : optimization by a colony of cooperating agents,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 29–41, 1996.
- [51] A. Bajpai and R. Yadav, “Ant colony optimization (aco) for the traveling salesman problem (tsp) using partitioning,” *International Journal of Scientific & Technology Research*, vol. 4, pp. 376–381, 2015.