Caractéristique d'Euler-Poincaré

Généralisation en nD de la caractéristique d'Euler

Caractéristique d'Euler-Poincaré

Caractéristique d'Euler χ de C un complexe cellulaire nD sans bord :

$$\chi(C) = \sum_{i=0}^{n} (-1)^i \times k_i(C)$$

avec $k_i(C)$ le nombre de cellules de dimension i de C.

Classification des surfaces

Théorème de classification des surfaces

Toute surface fermée est homéomorphe à une des trois surfaces suivantes :

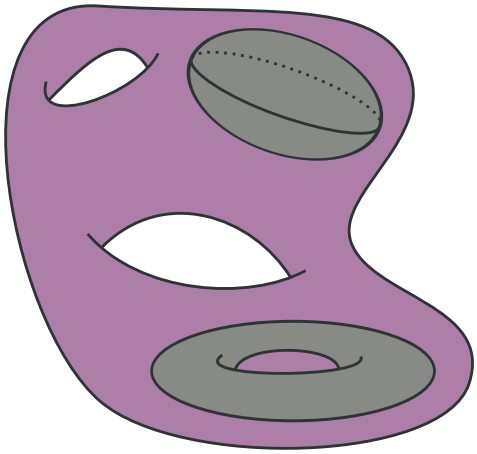
- S^2, la sphère de dimension 2 ;
- la somme connexe de g tores (ou tore à g trous) ;
- la somme connexe de k plans projectifs.

Classification complète des surfaces fermées par caractéristique d'Euler + orientabilité

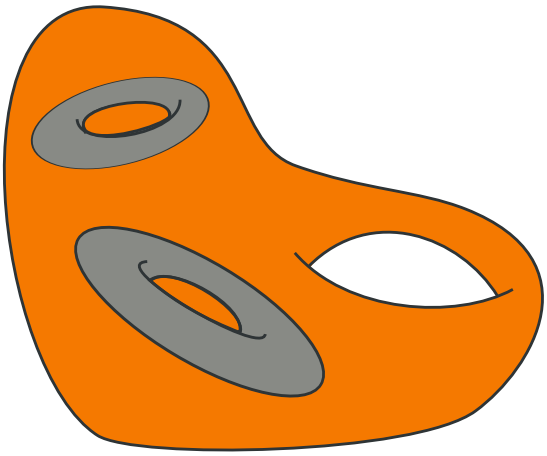
Extension directe aux surfaces à bord/non connexe :
ajouter nombre de bords et nombre de composantes connexes.

Pas de classification pour $n > 2$

2 objets non homéomorphe avec mêmes caractéristiques :
Euler, nb composantes connexes, nb bords



$\chi = 0, \#cc = 1, \#b = 3$



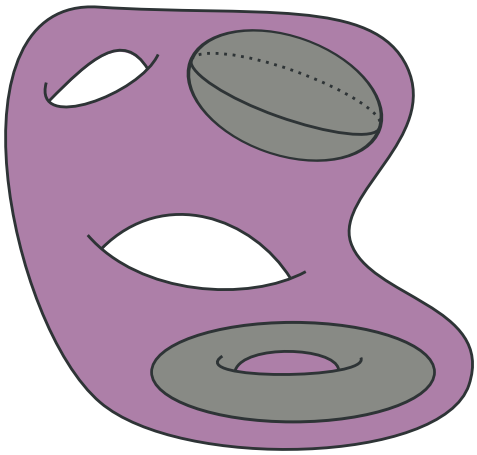
$\chi = 0, \#cc = 1, \#b = 3$

Nombres de Betti

Nombres de Betti

$\mathfrak{b}_0, \ldots, \mathfrak{b}_n$ (après tous nul)
Chaque $\mathfrak{b}_i$ est le rang du $i^{ème}$ groupe d'homologie.

Intuitivement : $\mathfrak{b}_i$ nombre de composantes connexes de i-surfaces



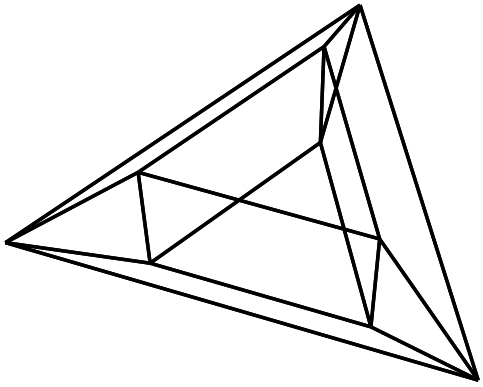
$\mathfrak{b}_0 = 1$
$\mathfrak{b}_1 = 3$
$\mathfrak{b}_2 = 2$

Nombres de Betti

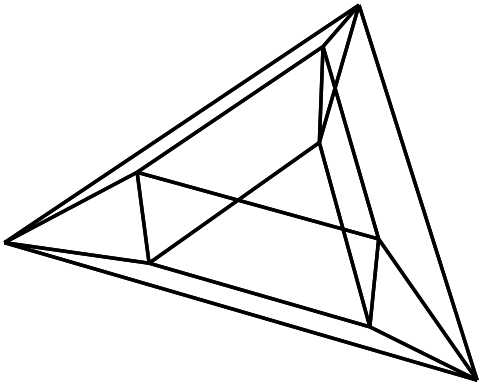
Lien avec caractéristique d'Euler-Poincaré

$$\chi(C) = \sum_{i=0}(-1)^i \times \mathfrak{b}_i(C)$$

Attention à la dimension :



2D : $\mathfrak{b}_0 = 1$ $\mathfrak{b}_1 = 2$ $\mathfrak{b}_2 = 1 \Rightarrow$ $\chi(C) = 0$.



3D : $\mathfrak{b}_0 = 1$ $\mathfrak{b}_1 = 1$ $\mathfrak{b}_2 = 0$ $\mathfrak{b}_3 = 0 \Rightarrow \chi(C) = 0$.

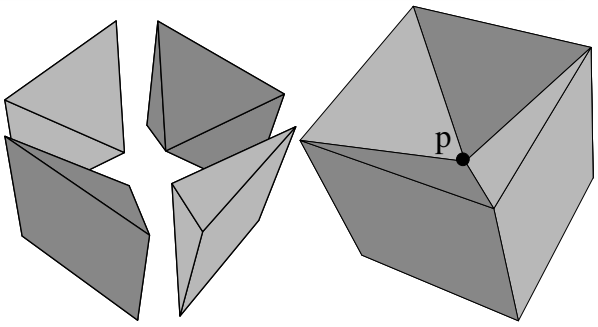
Liens avec cartes

- n-Carte : quasi variété nD avec ou sans bord orientable ;
- nG-Carte : quasi variété nD avec ou sans bord orientable ou non-orientable.

quasi variété nD

Des quasi variété $(n-1)$D collées entre elles le long de $(n-1)$-cellules.

En 2D, quasi-variété = variété ;
Pas en dimension supérieure ;
PB : $\nexists$ définition combinatoire de variété.



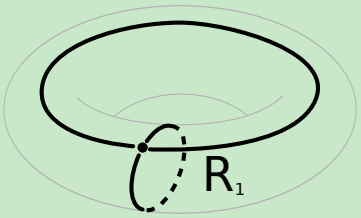
Liens avec cartes

Problème

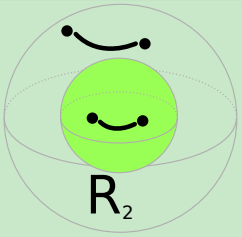
Les cellules ne sont pas forcément homéomorphe à des boules

Alors que c'est une condition des complexe cellulaires
$\Rightarrow$ pas possible d'utiliser directement les définitions précédentes

Exemples



$k_0 = 1, k_1 = 2, k_2 = 1, k_3 = 1$
1-2+1-1=-1 mais $\chi(C) = 0$



$k_0 = 4, k_1 = 2, k_2 = 2, k_3 = 1$
4-2+2-1=3 mais $\chi(C) = 2$

Heureusement il y a des solutions

Calcul de la caractéristique d'Euler-Poincaré

1. We use the alternating sum of number of cells in the border of R

Definition

Euler-Poincaré characteristic χ' of the border of a region R in dimension n :

$$\chi'(R) = \sum_{i=0}^{n-1}(-1)^i \times k_i(R)$$

2. We add implicit cells :

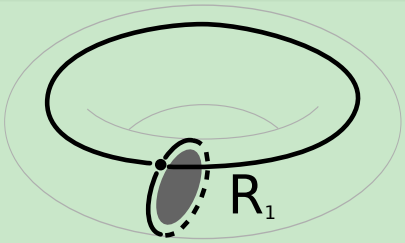
Implicit cells

cells that must be added in R so that each cell of R become homeomorphic to a ball

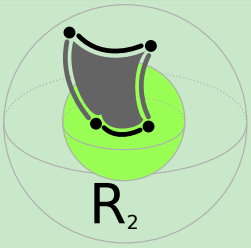
Implicit cells

In 3D, only two types of implicit cells (for tunnels and cavities)

With Implicit Cells



$k_0 = 1, k_1 = 2, k_2 = 1 + 1, k_3 = 1$
$\Rightarrow \chi'(R_1) = 0$ and $\chi(R_1) = 0$



$k_0 = 4, k_1 = 2 + 2, k_2 = 2 + 1, k_3 = 1$
$\Rightarrow \chi'(R_2) = 4$ and $\chi(R_2) = 2$

$\Rightarrow$ By counting implicit cells, we obtain the correct Euler-Poincaré Characteristic

Even better

we do not need to count the implicit cells, because we have :

$$\chi(R) = \frac{\chi'(R)}{2}$$

How to Compute Betti Numbers

Link Between Betti Numbers and Euler-Poincaré Characteristic

$$\chi(C) = \sum_{i=0}(-1)^i \times \mathfrak{b}_i(C)$$

Betti Numbers of a 3D objects [DUPAS DAMIAND 09]

- $\mathfrak{b}_0(R)$: number of connected components ;
- $\mathfrak{b}_2(R) = |\text{surfaces}(R)| - 1$.
- $\mathfrak{b}_1(R) = \mathfrak{b}_0(R) + \mathfrak{b}_2(R) - \chi'(R)/2$;

PS : Si chaque cellule homéomorphe à une boule $\Rightarrow$
$\mathfrak{b}_1(R) = \mathfrak{b}_0(R) + \mathfrak{b}_2(R) - \chi(R)$.

4. Algorithmes

Parcours

Très important : outils pour retrouver les orbites
Exemple : parcours d'une face en 3D

Algorithme 1 : Parcours de la face incidente à b ; Version basic

F une file;
push(b);
tant que F *non vide* **faire**
    $b' \leftarrow pop(F)$;
    si b' *n'est pas marqué* **alors**
        traitement b';
        marquer b';
        push($\alpha_0(b')$) ; push($\alpha_1(b')$) ; push($\alpha_3(b')$);
Démarquer les brins marqué;

Parcours optimisés

Parcours d'une face en 3D
on peux faire mieux en étudiant les propriétés des G-cartes
($\alpha_0 \circ \alpha_3$ et $\alpha_1 \circ \alpha_3$ sont des involutions)

Algorithme 2 : Parcours de la face incidente à b ; Version optimisée

$b' \leftarrow b$;
$nb_3 \leftarrow true$;
$nb_0 \leftarrow true$;
répéter
  traitement b';
  si nb_3 ***et*** $\alpha_3(b') \neq b'$ **alors** $b' \leftarrow \alpha_3(b')$;
  sinon
    si nb_0 **alors** $b' \leftarrow \alpha_0(b')$;
    sinon $b' \leftarrow \alpha_1(b')$;
    $nb_0 \leftarrow !nb_0$;
  $nb_3 \leftarrow !nb_3$;
jusqu'à $b' = b$;

Attention : marche pas si des brins tel que $\alpha_0(b) = b$ ou $\alpha_1(b) = b$

Algorithme local

Principe

- traiter un ensemble de brins, dans n'importe quel ordre
- faire un traitement local, c'est à dire concernant souvent ce brin et ses voisins

Avantages

- Algorithme souvent très simple à écrire
- souvent pas de cas particulier (brins sans voisin, nD)

Exemple : triangulation 1D, 2D, 3D

Triangulation 1D

Algorithme 3 : Triangulation 1d de l'arête incidente au brin d

pour chaque *brin* b *de l'arête incidente à* d **faire**
    $b_1 \leftarrow$ un nouveau brin ;
    si $\alpha_0(b)$ *déjà traité* **alors**
        $\alpha_1 - sew(b_1, \alpha_{00}(b))$;
    $\alpha_0 - sew(b, b_1)$
Démarquer les brins marqué;

Triangulation 2D

Algorithme 4 : Triangulation 2d de la face incidente au brin d

pour chaque *brin* b *de la face incidente à* d **faire**
    $b_1 \leftarrow$ un nouveau brin ; $b_2 \leftarrow$ un nouveau brin;
    $\alpha_0 - sew(b_1, b_2)$;
    si $\alpha_0(b)$ *déjà traité* **alors**
        $\alpha_1 - sew(b_2, \alpha_{010}(b))$;
    si $\alpha_1(b)$ *déjà traité* **alors**
        $\alpha_2 - sew(b_1, \alpha_{11}(b))$;
        $\alpha_2 - sew(b_2, \alpha_{110}(b))$;
    $\alpha_1 - sew(b, b_1)$
Démarquer les brins marqué;

Triangulation 3D

Algorithme 5 : Triangulation 3d du volume incident au brin d

pour chaque *brin* b *du volume incident à* d **faire**
- $b_1 \leftarrow$ un nouveau brin ; $b_2 \leftarrow$ un nouveau brin ; $b_3 \leftarrow$ un nouveau brin;
- $\alpha_1 - sew(b_1, b_2)$; $\alpha_0 - sew(b_2, b_3)$;
- **si** $\alpha_0(b)$ *déjà traité* **alors**
  - $\alpha_0 - sew(b_1, \alpha_{02}(b))$;
  - $\alpha_1 - sew(b_3, \alpha_{0210}(b))$;
- **si** $\alpha_1(b)$ *déjà traité* **alors**
  - $\alpha_2 - sew(b_2, \alpha_{1221}(b))$;
  - $\alpha_2 - sew(b_3, \alpha_{12210}(b))$;
- **si** $\alpha_2(b)$ *déjà traité* **alors**
  - $\alpha_3 - sew(b_1, \alpha_{22}(b))$;
  - $\alpha_3 - sew(b_2, \alpha_{221}(b))$;
  - $\alpha_3 - sew(b_3, \alpha_{2210}(b))$;
- $\alpha_2 - sew(b, b_1)$

Démarquer les brins marqué;

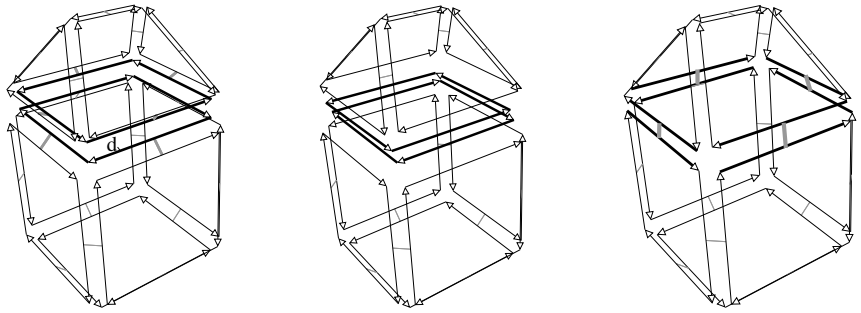
5. Opérations de base

Opérations de suppression

Suppressions

- Définies en dimension quelconque
- Contrainte : cellules localement de degré 2
- Suppression d'une i cellule $\Rightarrow$ fusion des (i+1) cellules

2-Suppression



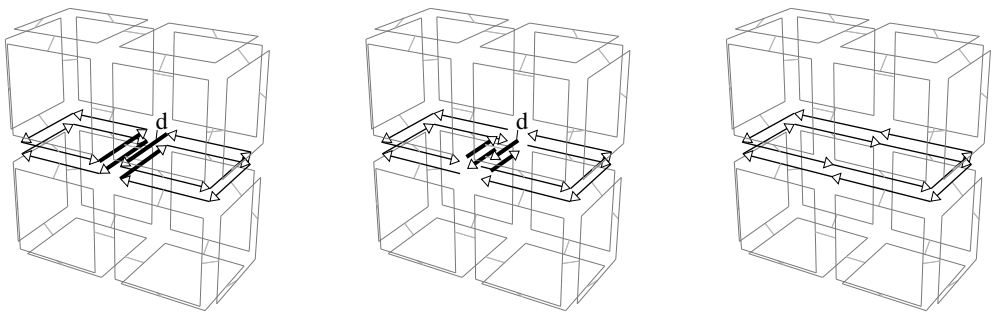
Quelques modèles Cartes combinatoires/généralisées Invariants Topologiques Algorithmes Opérations de base Cartes combinatoires dans CGAL Conclu

Opérations de suppression

Suppressions

- Définies en dimension quelconque
- Contrainte : cellules localement de degré 2
- Suppression d'une i cellule $\Rightarrow$ fusion des (i+1) cellules

1-Suppression

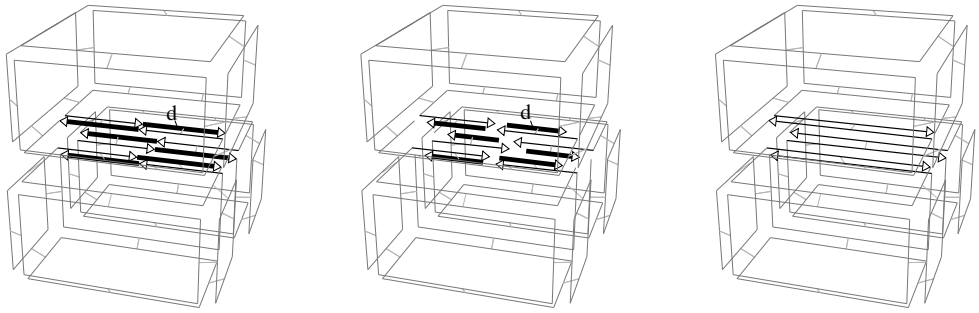


Opérations de suppression

Suppressions

- Définies en dimension quelconque
- Contrainte : cellules localement de degré 2
- Suppression d'une i cellule $\Rightarrow$ fusion des (i+1) cellules

0-Suppression



Autres opérations de base : contraction, insertion, éclatement

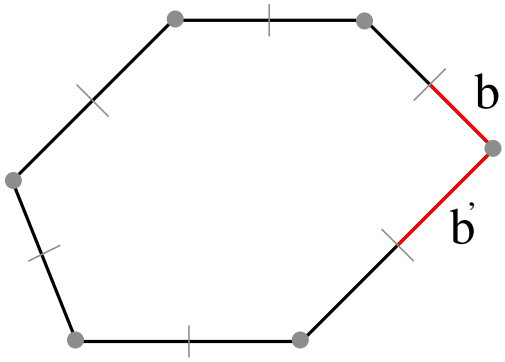
Removal / Contraction Operations

We define the i-removal (or contraction) in a n-dimensional space

- Removal
  - 1D
  - 2D
  - nD
- Contraction : dual of removal
- Generalization for simultaneous operations

Dimension 1 : 0-removal

removing a vertex $C = <>_{N-\{0\}}$ $(N = \{0, 1\})$

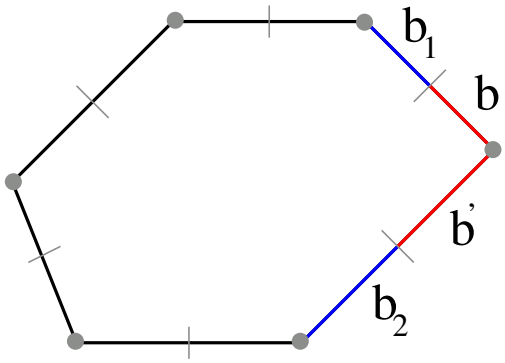


Initial 1-G-map ; $C = \{b, b'\}$

Quelques modèles Cartes combinatoires/généralisées Invariants Topologiques Algorithmes Opérations de base Cartes combinatoires dans CGAL Conclu
Removal

Dimension 1 : 0-removal

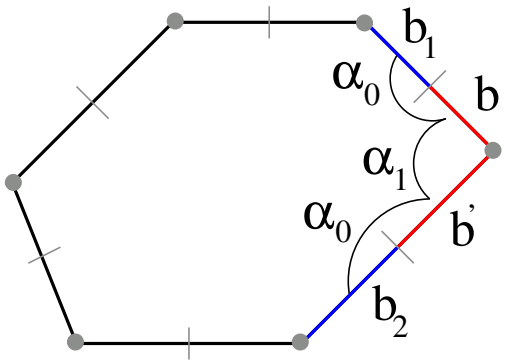
removing a vertex $C = <>_{N-\{0\}}$ $(N = \{0, 1\})$



$B^S = C\alpha_0 - C$ $(= \{b_1, b_2\})$

Dimension 1 : 0-removal

removing a vertex $C = <>_{N-\{0\}}$ $(N = \{0, 1\})$

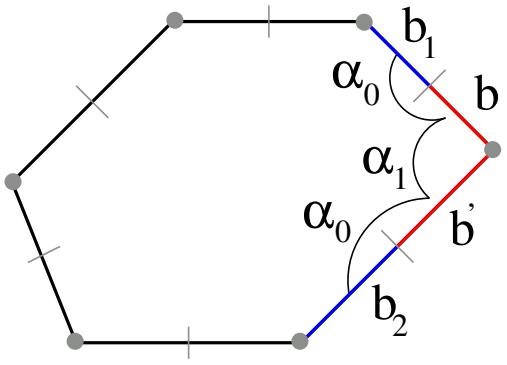


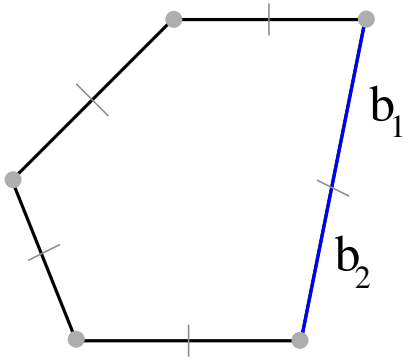
$\forall b' \in B^S,\ b'\alpha'_0 = b'(\alpha_0\alpha_1)^k\alpha_0$ k smallest integer

Quelques modèles Cartes combinatoires/généralisées Invariants Topologiques Algorithmes Opérations de base Cartes combinatoires dans CGAL Conclu
Removal

Dimension 1 : 0-removal

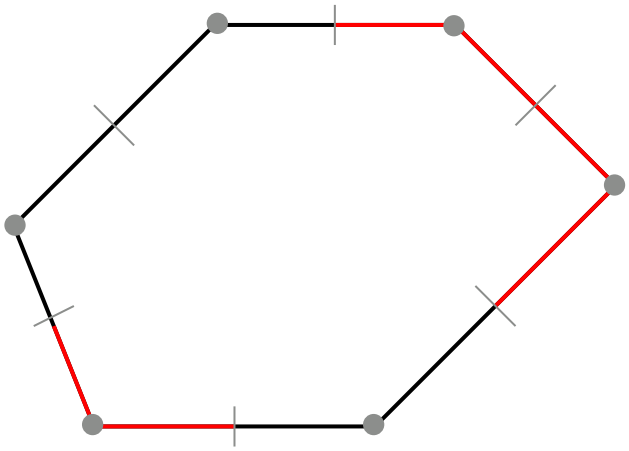
removing a vertex $C = <>_{N-\{0\}}$ $(N = \{0, 1\})$





Resulting 1-G-map

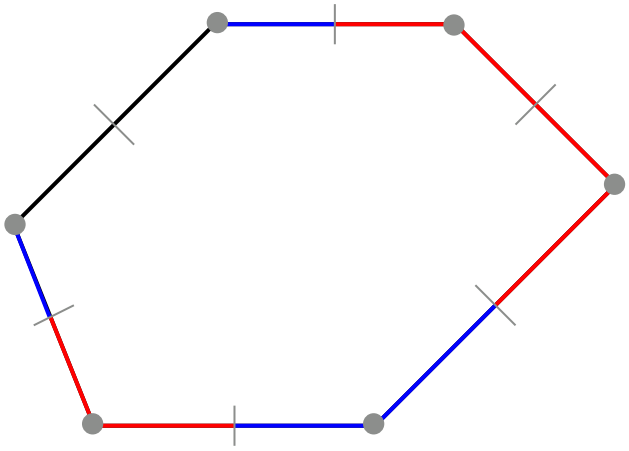
Simultaneous 0-removals



Initial 1-G-map ; vertices to remove

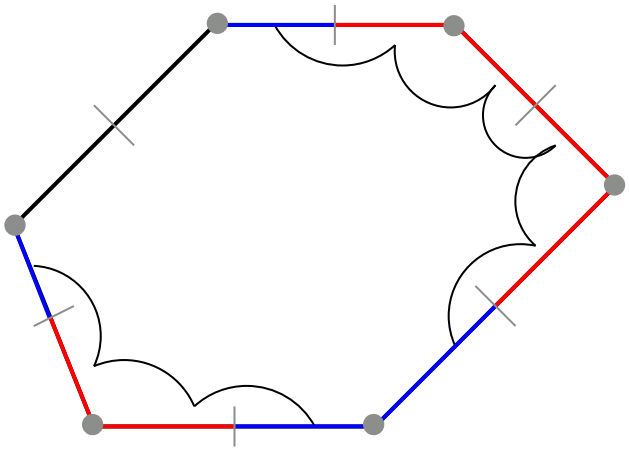
Quelques modèles Cartes combinatoires/généralisées Invariants Topologiques Algorithmes Opérations de base Cartes combinatoires dans CGAL Conclu
Removal

Simultaneous 0-removals



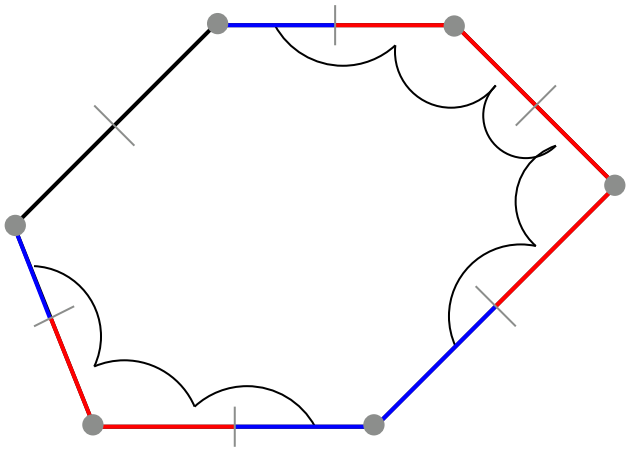
$B^S = C\alpha_0 - C$

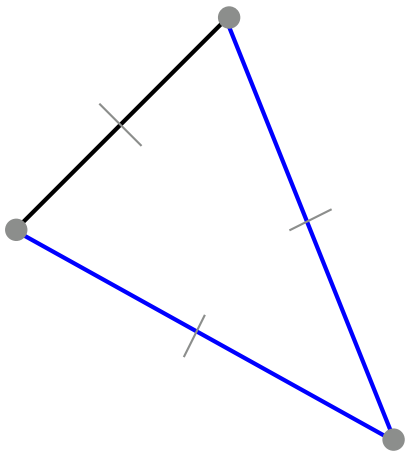
Simultaneous 0-removals



$\forall b' \in B^S, b'\alpha'_0 = b'(\alpha_0\alpha_1)^k\alpha_0$ k smallest integer

Quelques modèles Cartes combinatoires/généralisées Invariants Topologiques Algorithmes Opérations de base Cartes combinatoires dans CGAL Conclu
Removal
Simultaneous 0-removals

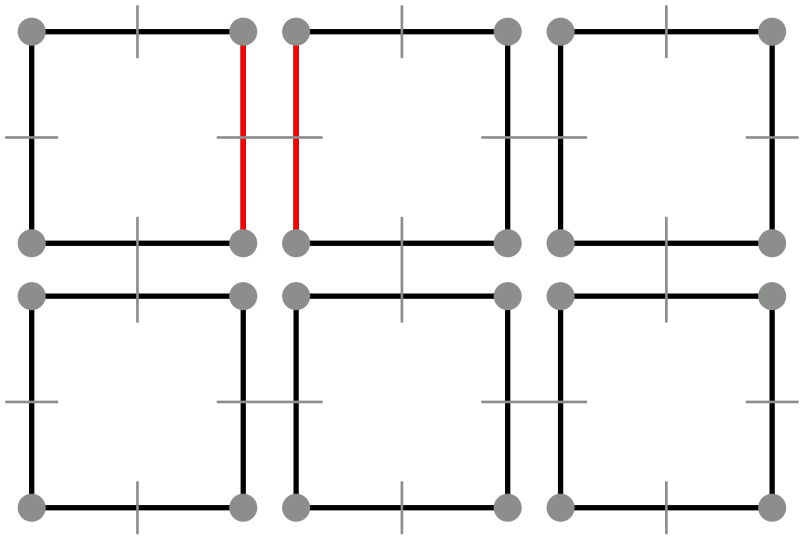




Resulting 1-G-map

Dimension 2 : 1-removal

removing an edge $C = <>_{N-\{1\}}$ $(N = \{0, 1, 2\})$

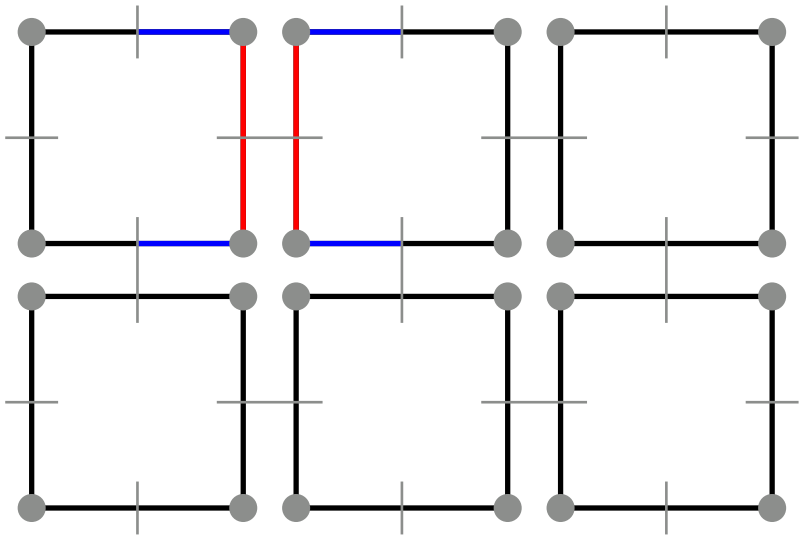


Initial 2-G-map ; edge to remove

Quelques modèles Cartes combinatoires/généralisées Invariants Topologiques Algorithmes Opérations de base Cartes combinatoires dans CGAL Conclu
Removal

Dimension 2 : 1-removal

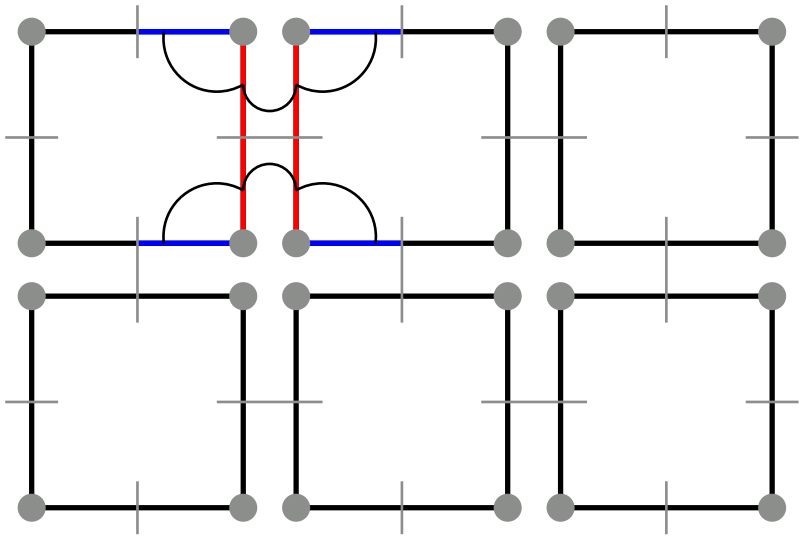
removing an edge $C = <>_{N-\{1\}}$ $(N = \{0, 1, 2\})$



$B^S = C\alpha_1 - C$

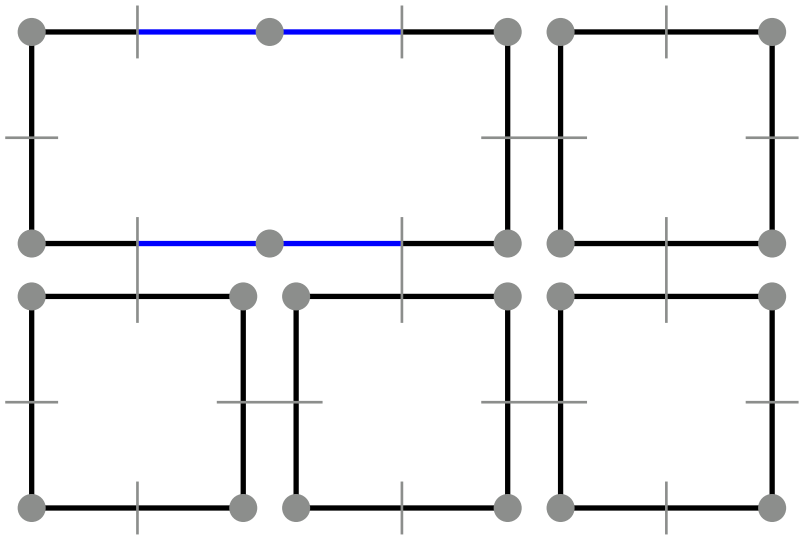
Dimension 2 : 1-removal

removing an edge $C = <>_{N-\{1\}}$ $(N = \{0,1,2\})$



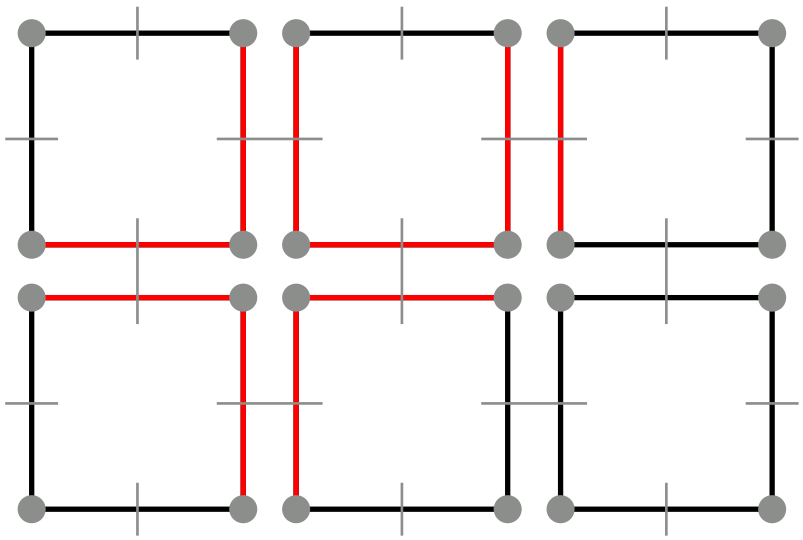
$\forall b' \in B^S,\ b'\alpha'_1 = b'(\alpha_1\alpha_2)^k\alpha_1$ k smallest integer

Quelques modèles Cartes combinatoires/généralisées Invariants Topologiques Algorithmes Opérations de base Cartes combinatoires dans CGAL Conclu
Removal
Dimension 2 : 1-removal
removing an edge $C = <>_{N-\{1\}}$ $(N = \{0,1,2\})$



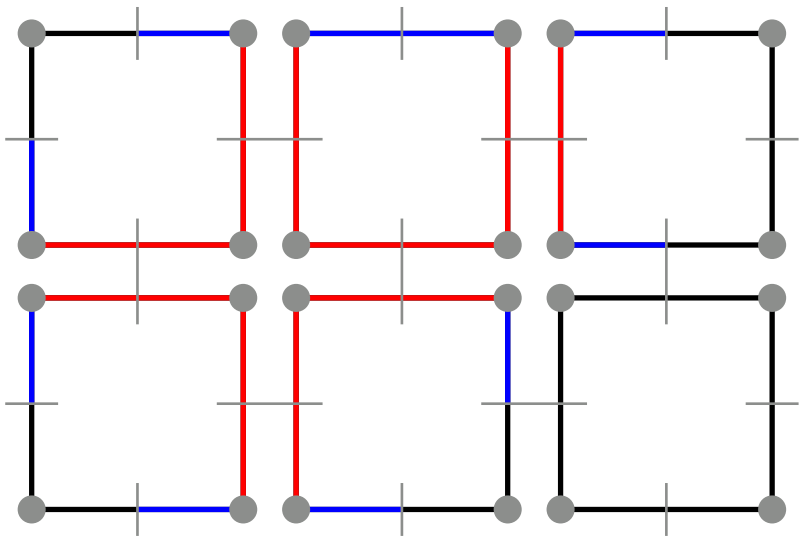
Resulting 2-G-map

Simultaneous 1-removals



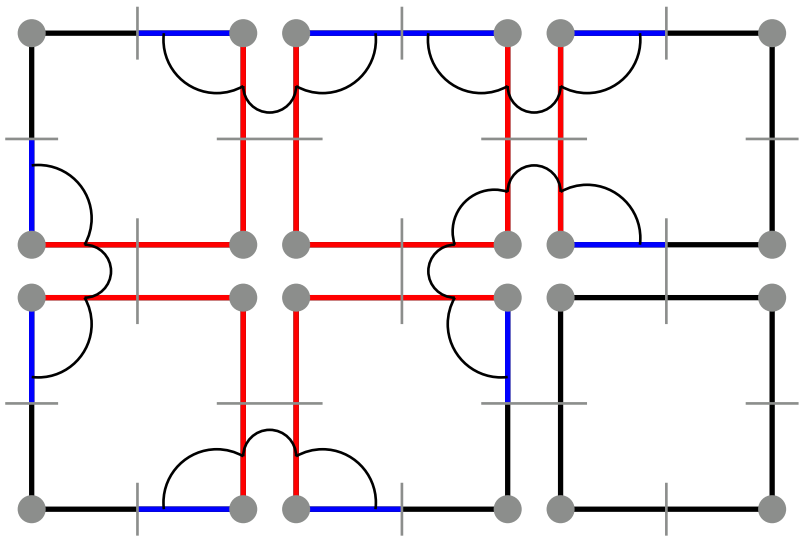
Initial 2-G-map ; edges to remove

Quelques modèles Cartes combinatoires/généralisées Invariants Topologiques Algorithmes Opérations de base Cartes combinatoires dans CGAL Conclu
Removal
Simultaneous 1-removals



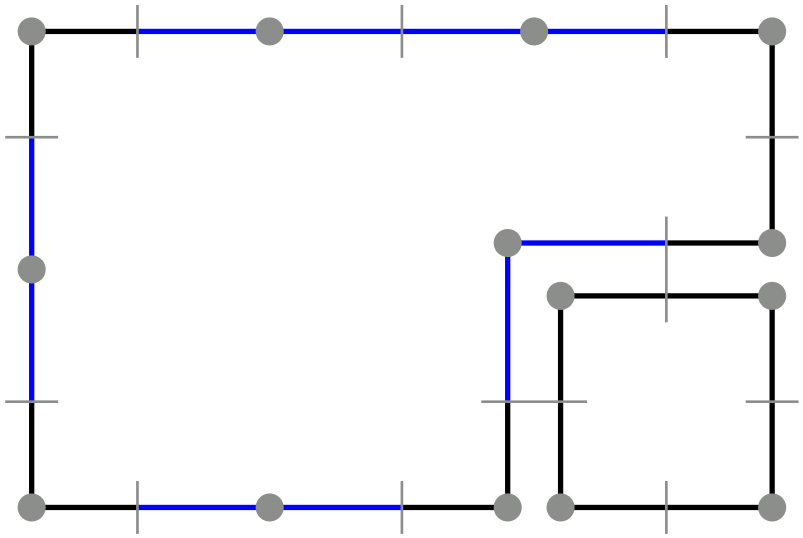
$B^S = C\alpha_1 - C$

Simultaneous 1-removals



$\forall b' \in B^S, b'\alpha'_1 = b'(\alpha_1\alpha_2)^k\alpha_1$ k smallest integer

Quelques modèles Cartes combinatoires/généralisées Invariants Topologiques Algorithmes Opérations de base Cartes combinatoires dans CGAL Conclu
Removal
Simultaneous 1-removals



Resulting 2-G-map

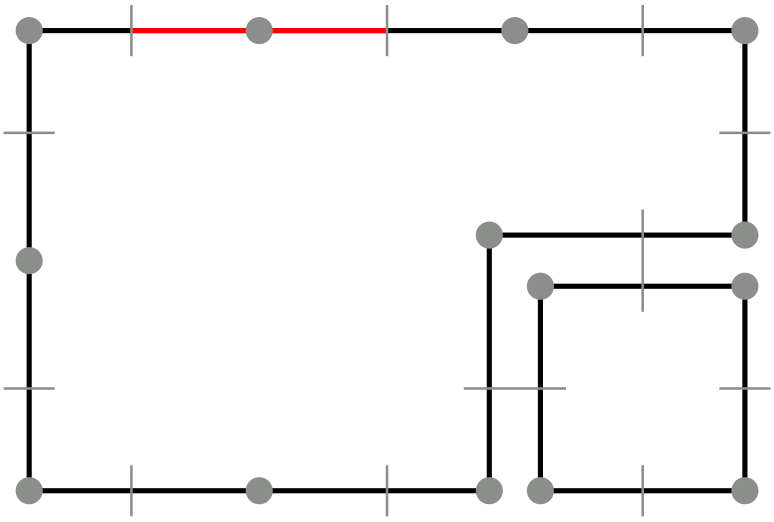
Dimension 2 : 0-removal

removing a vertex $C = <>_{N-\{0\}}$ $(N = \{0, 1, 2\})$

Precondition : $\forall b' \in C,\ b'\alpha_1\alpha_2 = b'\alpha_2\alpha_1$

Initial 2-G-map

vertex to remove



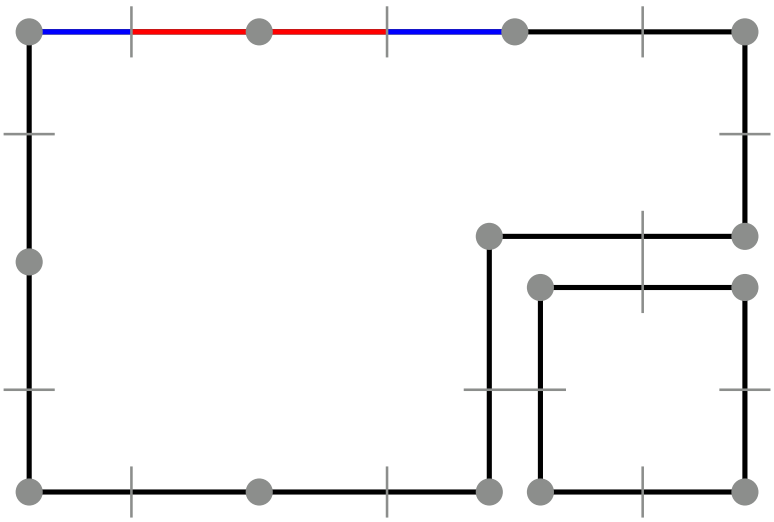
Quelques modèles Cartes combinatoires/généralisées Invariants Topologiques Algorithmes Opérations de base Cartes combinatoires dans CGAL Conclu
Removal

Dimension 2 : 0-removal

removing a vertex $C = <>_{N-\{0\}}$ $(N = \{0, 1, 2\})$

Precondition : $\forall b' \in C,\ b'\alpha_1\alpha_2 = b'\alpha_2\alpha_1$

$B^S = C\alpha_0 - C$



Utilisation des Cartes pour le Traitement d'Images et la Modélisation Géométrique Guillaume Damiand 57 / 88

Dimension 2 : 0-removal

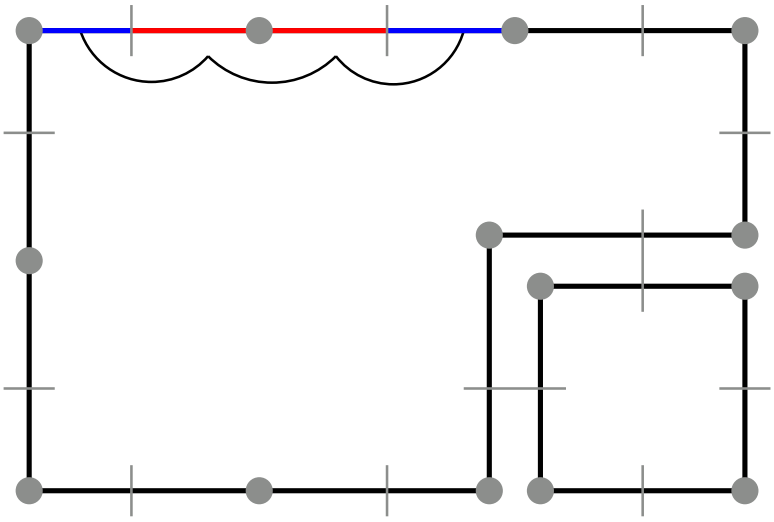
removing a vertex $C = <>_{N-\{0\}}$ $(N = \{0,1,2\})$

Precondition : $\forall b' \in C,\ b'\alpha_1\alpha_2 = b'\alpha_2\alpha_1$

$\forall b' \in B^S$

$b'\alpha'_0 = b'(\alpha_0\alpha_1)^k\alpha_0$

k smallest integer



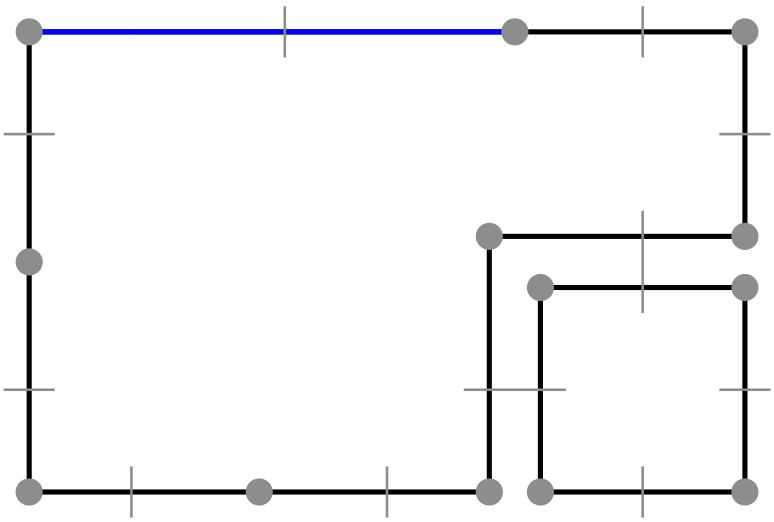
Quelques modèles Cartes combinatoires/généralisées Invariants Topologiques Algorithmes Opérations de base Cartes combinatoires dans CGAL Conclu
Removal

Dimension 2 : 0-removal

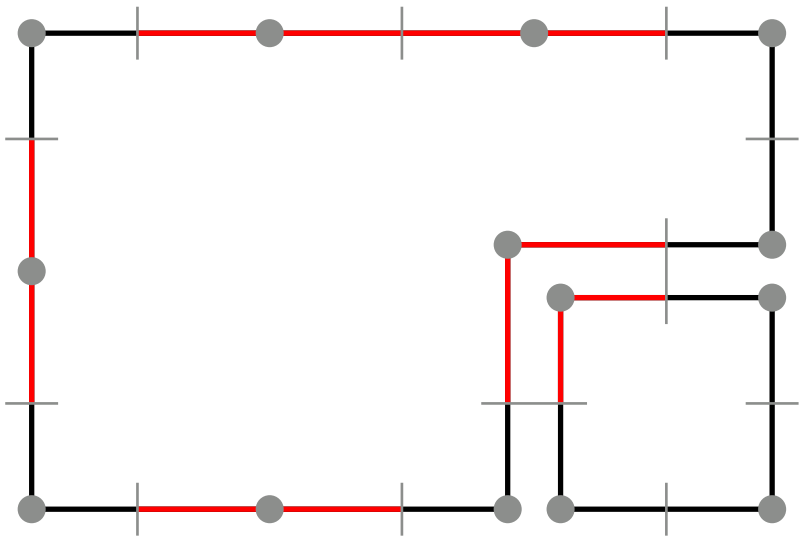
removing a vertex $C = <>_{N-\{0\}}$ $(N = \{0,1,2\})$

Precondition : $\forall b' \in C,\ b'\alpha_1\alpha_2 = b'\alpha_2\alpha_1$

Resulting 2-G-map

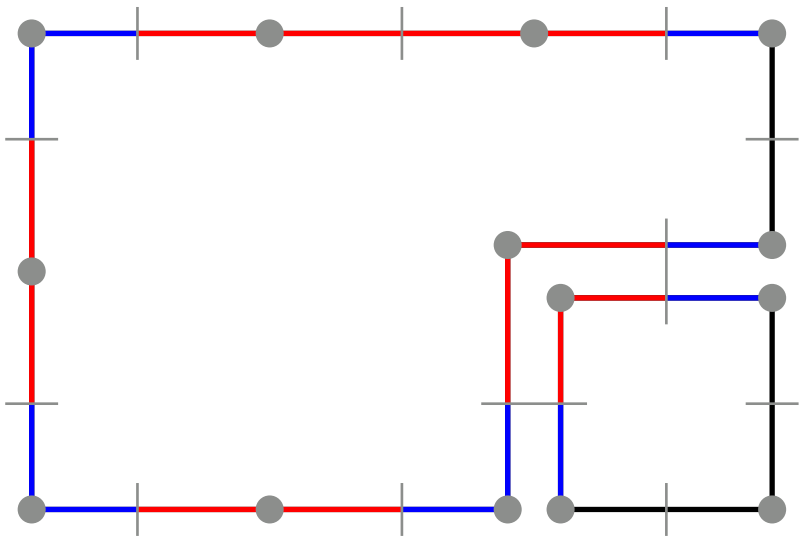


Simultaneous 0-removals



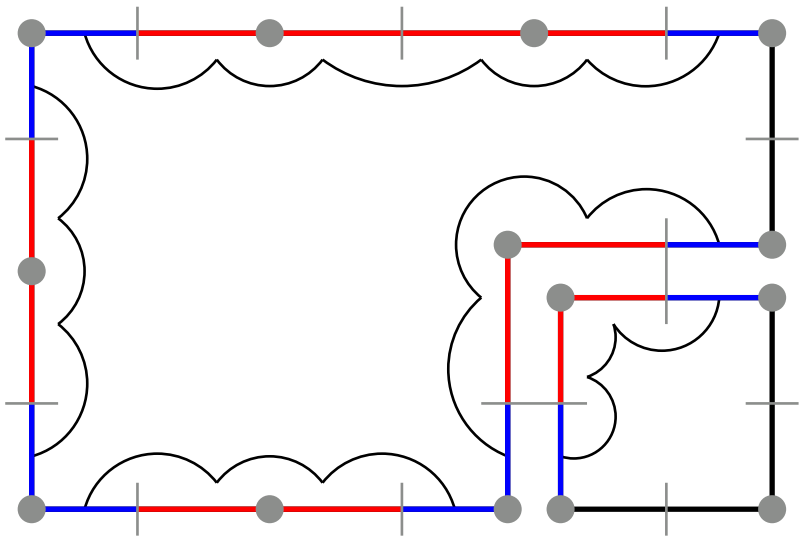
Initial 2-G-map ; vertices to remove

Quelques modèles Cartes combinatoires/généralisées Invariants Topologiques Algorithmes Opérations de base Cartes combinatoires dans CGAL Conclu
Removal
Simultaneous 0-removals



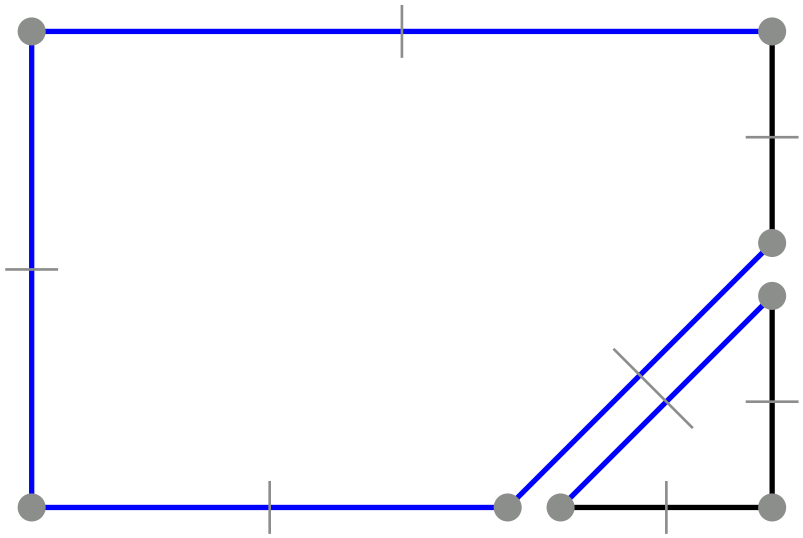
$B^S = C\alpha_0 - C$

Simultaneous 0-removals



$\forall b' \in B^S, b'\alpha'_0 = b'(\alpha_0\alpha_1)^k\alpha_0$ k smallest integer

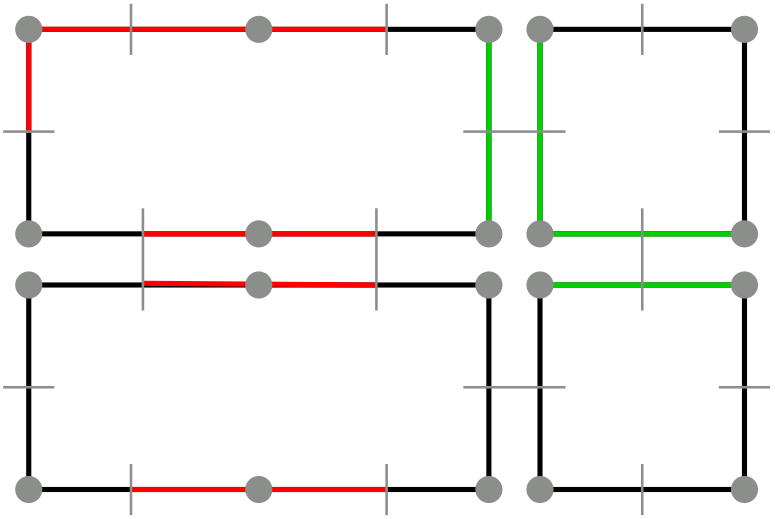
Quelques modèles Cartes combinatoires/généralisées Invariants Topologiques Algorithmes Opérations de base Cartes combinatoires dans CGAL Conclu
Removal
Simultaneous 0-removals



Resulting 2-G-map

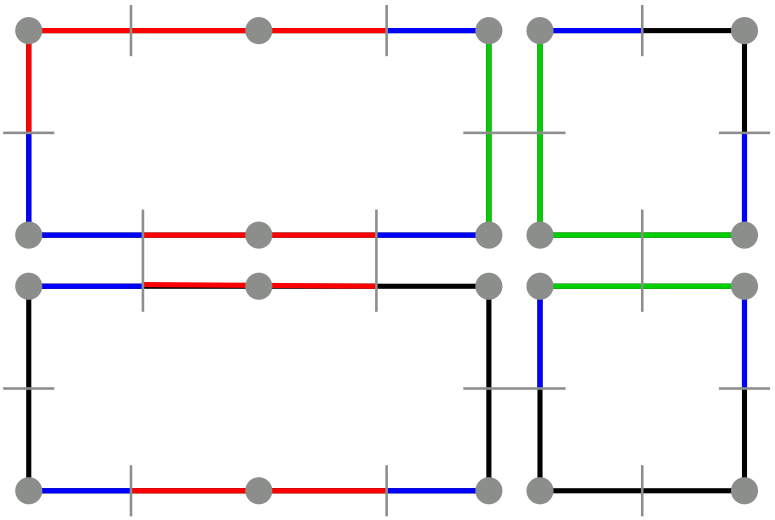
Simultaneous 0 and 1-removals

Precondition : disjoined cells



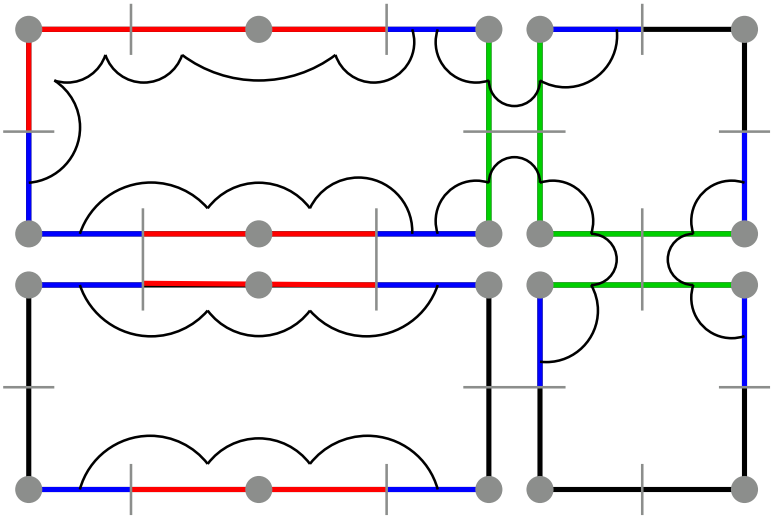
Initial 2-G-map ;
vertices to remove ; edges to remove

Quelques modèles Cartes combinatoires/généralisées Invariants Topologiques Algorithmes Opérations de base Cartes combinatoires dans CGAL Conclu
Removal
Simultaneous 0 and 1-removals



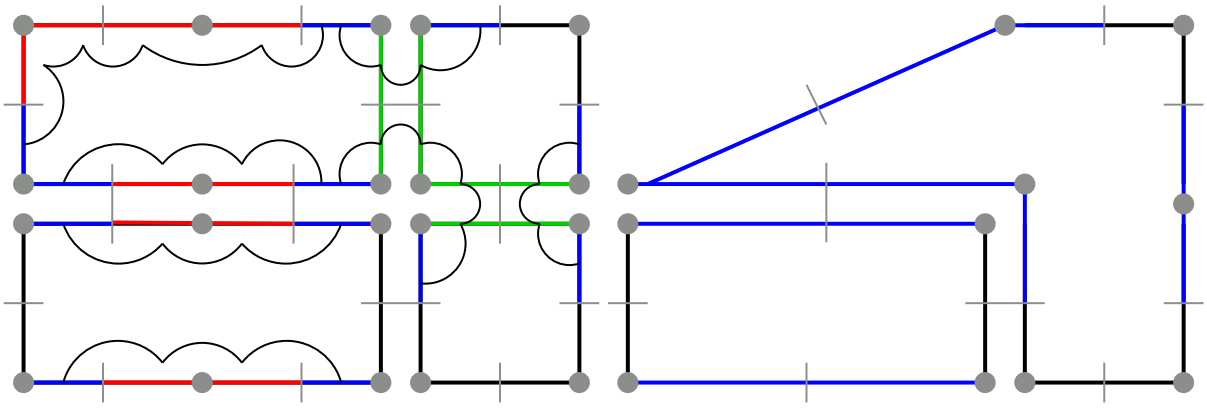
$B^{S0} = C\alpha_0 - C$ and $B^{S1} = C\alpha_1 - C$

Simultaneous 0 and 1-removals



Quelques modèles Cartes combinatoires/généralisées Invariants Topologiques Algorithmes Opérations de base Cartes combinatoires dans CGAL Conclu
Removal

Simultaneous 0 and 1-removals



Resulting 2-G-map

Dimension n : simultaneous i-removals

$C = <>_{N-\{i\}}$ $\quad N = \{0, \ldots, n\}$

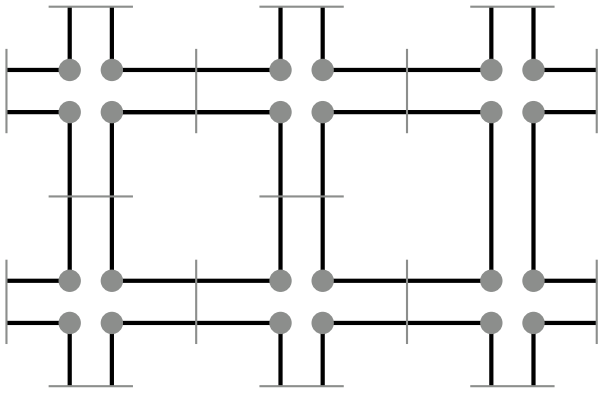
Preconditions :

- $i \in \{0, \ldots, n-1\}$
- $\forall b' \in C, b'\alpha_{(i+1)}\alpha_{(i+2)} = b'\alpha_{(i+2)}\ \alpha_{(i+1)}$
  (except for $i = n-1$)

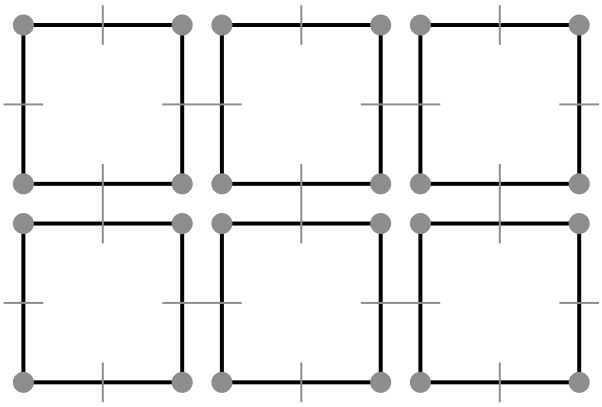
$B^S = C\alpha_i - C$
$\forall b' \in B^S, b'\alpha'_i = b'(\alpha_i\alpha_{(i+1)})^k\alpha_i$ $\quad$ k smallest integer

Dimension 2 : 1-contraction

$G = (B, \alpha_0, \ldots, \alpha_n) \Rightarrow$ dual $G' = (B, \alpha_n, \ldots, \alpha_0)$



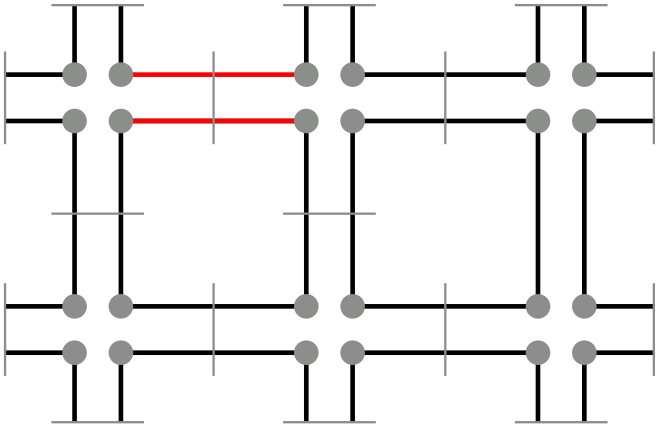
Initial 2-G-map



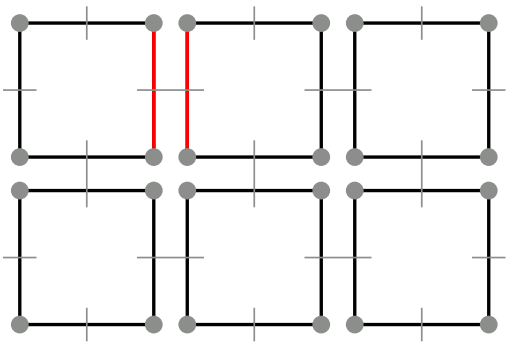
Dual 2-G-map

Dimension 2 : 1-contraction

contracting an edge $C = <>_{N-\{1\}}$ $(N = \{0, 1, 2\})$



Initial 2-G-map
edge to contract

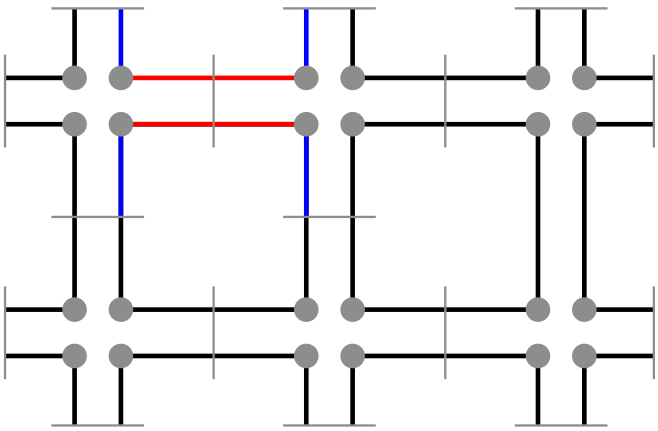


Dual 2-G-map
edge to remove

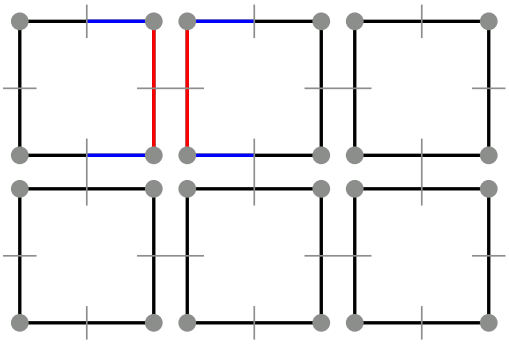
Quelques modèles Cartes combinatoires/généralisées Invariants Topologiques Algorithmes Opérations de base Cartes combinatoires dans CGAL Conclu
Contraction

Dimension 2 : 1-contraction

contracting an edge $C = <>_{N-\{1\}}$ $(N = \{0, 1, 2\})$



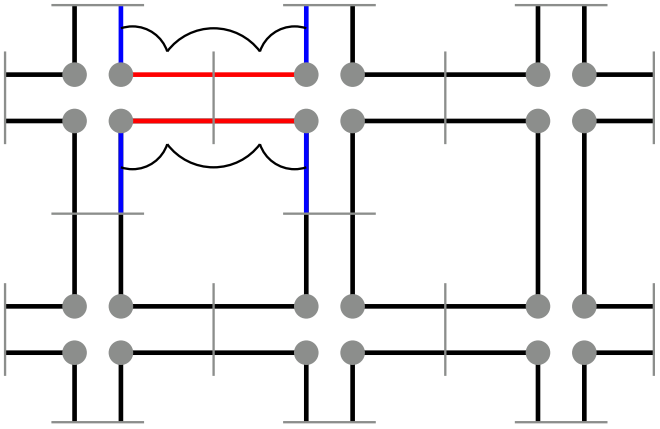
$B^S = C\alpha_1 - C$



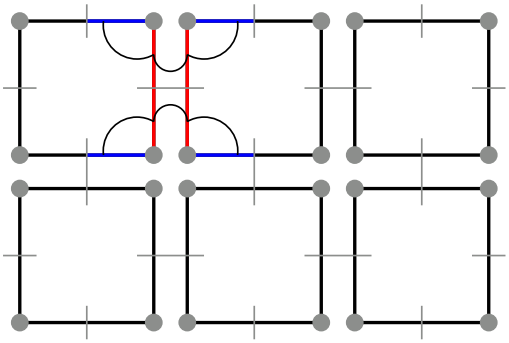
$B^S = C\alpha_1 - C$

Dimension 2 : 1-contraction

contracting an edge $C = <>_{N-\{1\}}$ $(N = \{0, 1, 2\})$

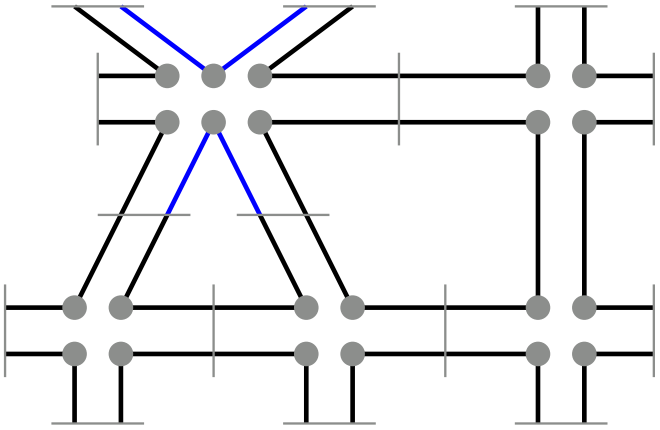


$\forall b' \in B^S, b'\alpha'_1 = b'(\alpha_1\alpha_0)^k\alpha_1$
k smallest integer

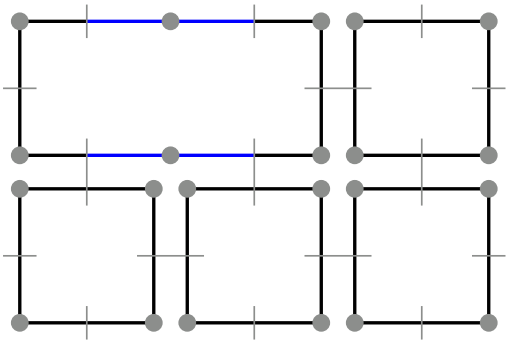


$b'\alpha'_1 = b'(\alpha_1\alpha_2)^k\alpha_1$
k smallest integer

Quelques modèles Cartes combinatoires/généralisées Invariants Topologiques Algorithmes Opérations de base Cartes combinatoires dans CGAL Conclu
Contraction
Dimension 2 : 1-contraction
contracting an edge $C = <>_{N-\{1\}}$ $(N = \{0, 1, 2\})$



Resulting 2-G-map



Resulting 2-G-map

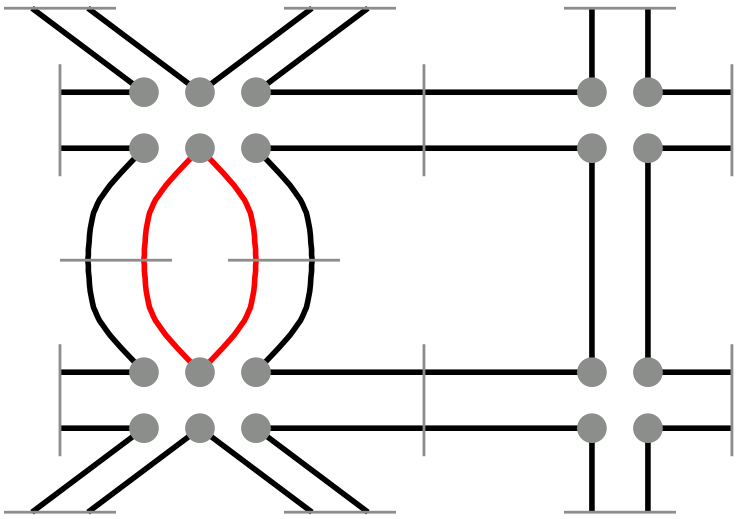
Dimension 2 : 2-contraction

contracting a face $C = <>_{N-\{2\}}$ $(N = \{0, 1, 2\})$

Precondition : $\forall b' \in C,\ b'\alpha_0\alpha_1 = b'\alpha_1\alpha_0$

Initial 2-G-map

face to contract



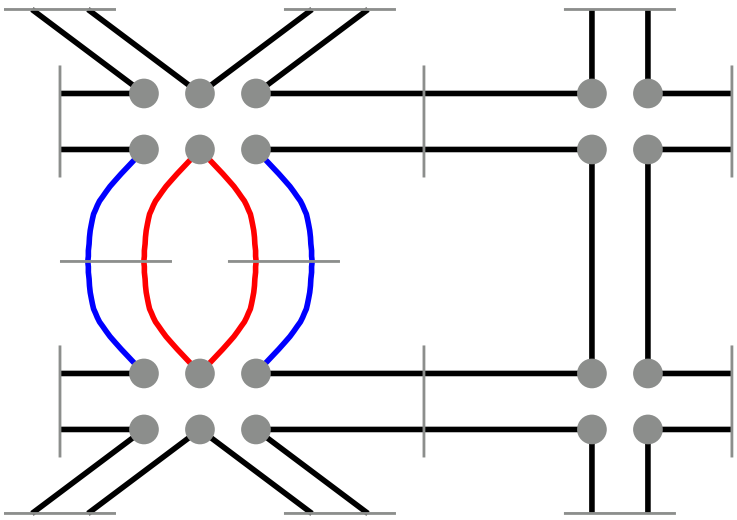
Quelques modèles Cartes combinatoires/généralisées Invariants Topologiques Algorithmes Opérations de base Cartes combinatoires dans CGAL Conclu
Contraction

Dimension 2 : 2-contraction

contracting a face $C = <>_{N-\{2\}}$ $(N = \{0, 1, 2\})$

Precondition : $\forall b' \in C,\ b'\alpha_0\alpha_1 = b'\alpha_1\alpha_0$

$B^S = C\alpha_2 - C$



Utilisation des Cartes pour le Traitement d'Images et la Modélisation Géométrique Guillaume Damiand 62 / 88

Dimension 2 : 2-contraction

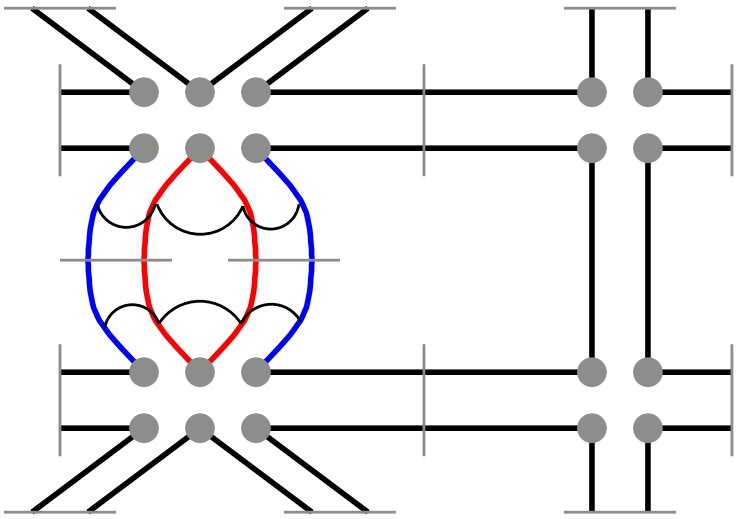
contracting a face $C = <>_{N-\{2\}}$ $(N = \{0,1,2\})$

Precondition : $\forall b' \in C,\ b'\alpha_0\alpha_1 = b'\alpha_1\alpha_0$

$\forall b' \in B^S$

$b'\alpha'_2 = b'(\alpha_2\alpha_1)^k\alpha_2$

k smallest integer



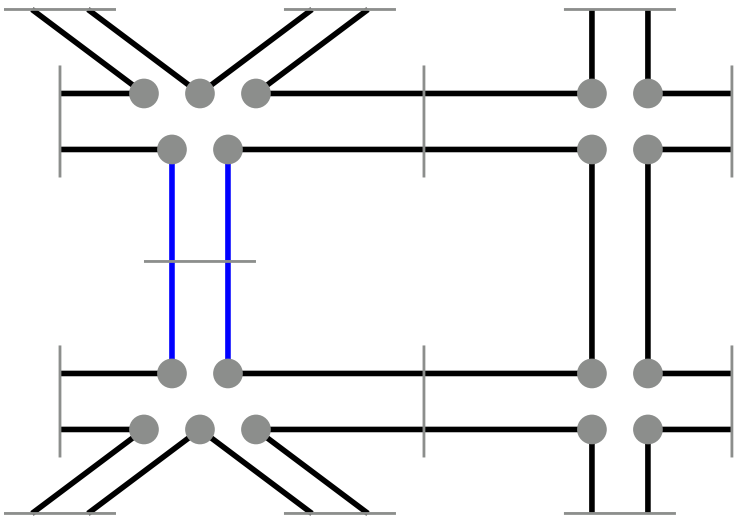
Quelques modèles Cartes combinatoires/généralisées Invariants Topologiques Algorithmes Opérations de base Cartes combinatoires dans CGAL Conclu
Contraction

Dimension 2 : 2-contraction

contracting a face $C = <>_{N-\{2\}}$ $(N = \{0,1,2\})$

Precondition : $\forall b' \in C,\ b'\alpha_0\alpha_1 = b'\alpha_1\alpha_0$

Resulting 2-G-map



Dimension n : simultaneous i-contractions

$C = <>_{N-\{i\}}$ $N = \{0, \ldots, n\}$

Preconditions :

- $i \in \{1, \ldots, n\}$
- $\forall b' \in C, b'\alpha_{(i-1)}\alpha_{(i-2)} = b'\alpha_{(i-2)}\ \alpha_{(i-1)}$ (except for $i = 1$)

$B^S = C\alpha_i - C$

$\forall b' \in B^S, b'\alpha'_i = b'(\alpha_i\alpha_{(i-1)})^k\alpha_i$ k smallest integer

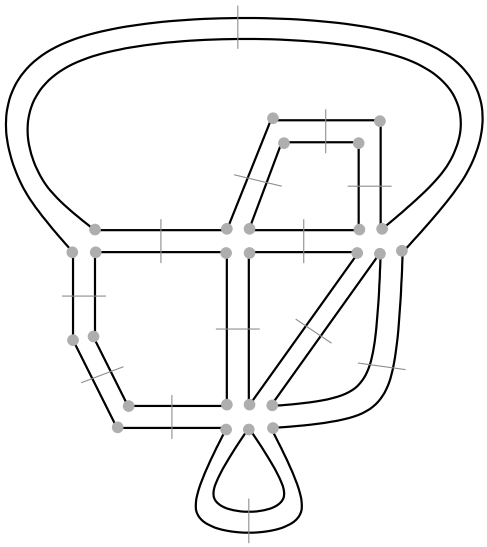
Suppressions et contractions simultanées

Definition

$G = (B, \alpha_0, \ldots, \alpha_n)$ une nG-carte
S les cellules à supprimer
C les cellules à contracter

Préconditions :

- $\forall c, c' \in C \cup S : c \cap c' = \emptyset$;
- les cellules de S sont supprimables ;
- les cellules de C sont contractibles.



vertices to remove ; edges to remove ;
edges to contract ; face to contract ;

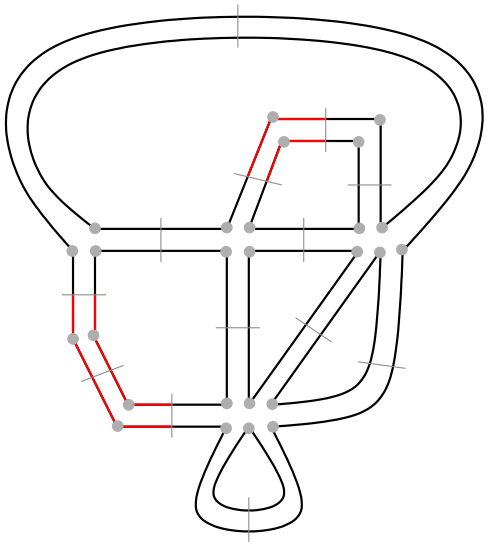
Suppressions et contractions simultanées

Definition

$G = (B, \alpha_0, \ldots, \alpha_n)$ une nG-carte
S les cellules à supprimer
C les cellules à contracter

Préconditions :

- $\forall c, c' \in C \cup S : c \cap c' = \emptyset$;
- les cellules de S sont supprimables ;
- les cellules de C sont contractibles.



vertices to remove ; edges to remove ;
edges to contract ; face to contract ;

Quelques modèles Cartes combinatoires/généralisées Invariants Topologiques Algorithmes Opérations de base Cartes combinatoires dans CGAL Conclu
Generalization

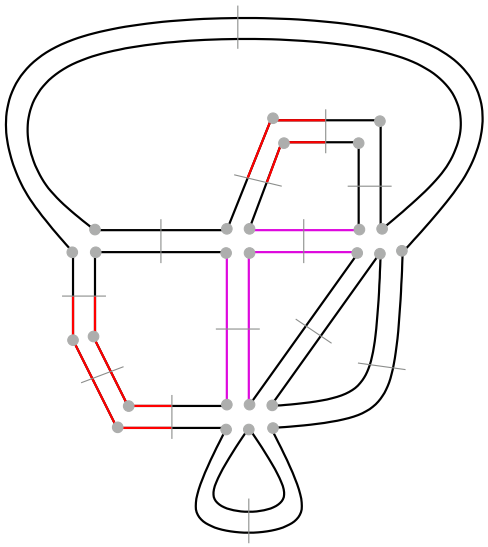
Suppressions et contractions simultanées

Definition

$G = (B, \alpha_0, \ldots, \alpha_n)$ une nG-carte
S les cellules à supprimer
C les cellules à contracter

Préconditions :

- $\forall c, c' \in C \cup S : c \cap c' = \emptyset$;
- les cellules de S sont supprimables ;
- les cellules de C sont contractibles.



vertices to remove ; edges to remove ;
edges to contract ; face to contract ;

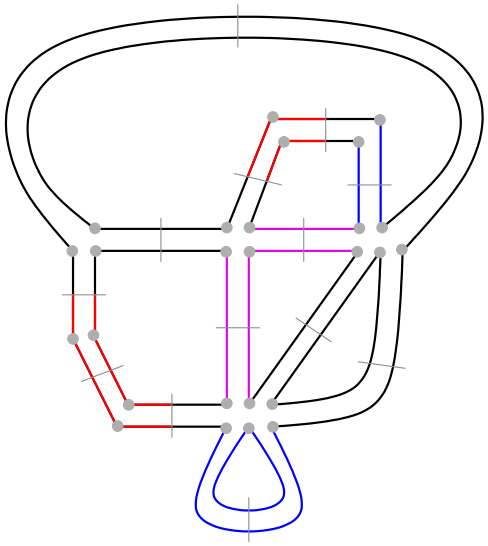
Suppressions et contractions simultanées

Definition

$G = (B, \alpha_0, \ldots, \alpha_n)$ une nG-carte
S les cellules à supprimer
C les cellules à contracter

Préconditions :

- $\forall c, c' \in C \cup S : c \cap c' = \emptyset$;
- les cellules de S sont supprimables ;
- les cellules de C sont contractibles.



vertices to remove ; edges to remove ;
edges to contract ; face to contract ;

Quelques modèles Cartes combinatoires/généralisées Invariants Topologiques Algorithmes Opérations de base Cartes combinatoires dans CGAL Conclu
Generalization

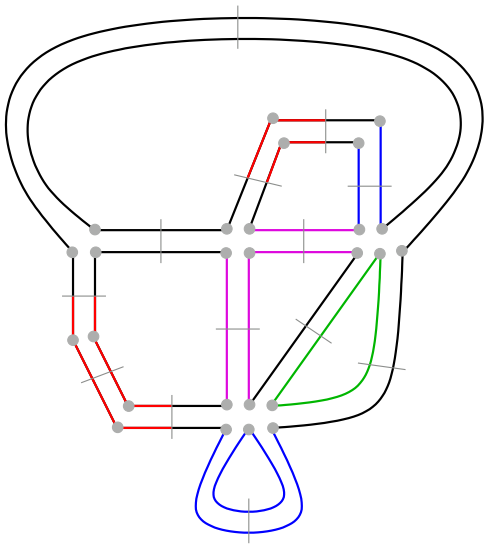
Suppressions et contractions simultanées

Definition

$G = (B, \alpha_0, \ldots, \alpha_n)$ une nG-carte
S les cellules à supprimer
C les cellules à contracter

Préconditions :

- $\forall c, c' \in C \cup S : c \cap c' = \emptyset$;
- les cellules de S sont supprimables ;
- les cellules de C sont contractibles.



vertices to remove ; edges to remove ;
edges to contract ; face to contract ;

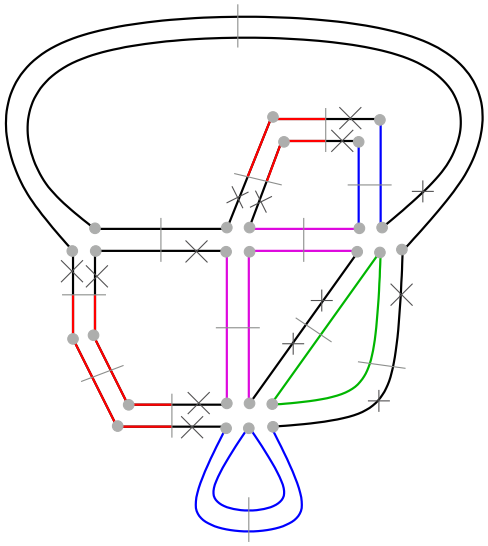
Suppressions et contractions simultanées

Definition

$G = (B, \alpha_0, \ldots, \alpha_n)$ une nG-carte
S les cellules à supprimer
C les cellules à contracter

Préconditions :

- $\forall c, c' \in C \cup S : c \cap c' = \emptyset$;
- les cellules de S sont supprimables ;
- les cellules de C sont contractibles.



vertices to remove ; edges to remove ;
edges to contract ; face to contract ;

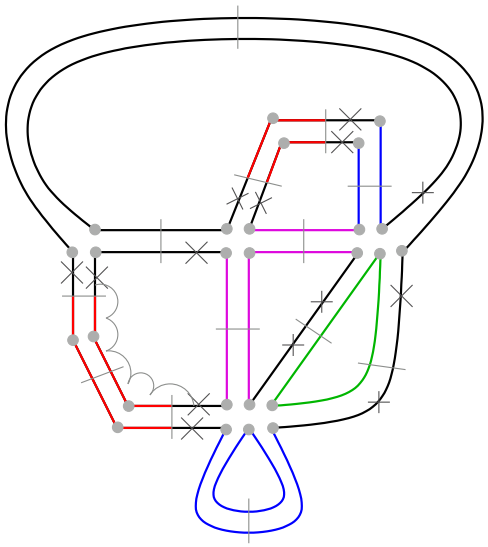
Quelques modèles Cartes combinatoires/généralisées Invariants Topologiques Algorithmes Opérations de base Cartes combinatoires dans CGAL Conclu
Generalization

Suppressions et contractions simultanées

Definition

$G' = (B', \alpha'_0, \ldots, \alpha'_n)$:

- $\forall i : 0 \leq i \leq n, \forall b \in BV_i$:
  $b\alpha'_i = b' = b(\alpha_i\alpha_{l_1}) \ldots (\alpha_i\alpha_{l_r})\alpha_i$, où :
  - r est le plus petit entier tel que $b' \in BV_i$;
  - $\forall j : 1 \leq j \leq r, l_j =$
    - $i+1$ si $b_j = b(\alpha_i\alpha_{l_1}) \ldots (\alpha_i\alpha_{l_{j-1}})\alpha_i \in S_i$
    - $i-1$ sinon $(b_j \in C_i)$



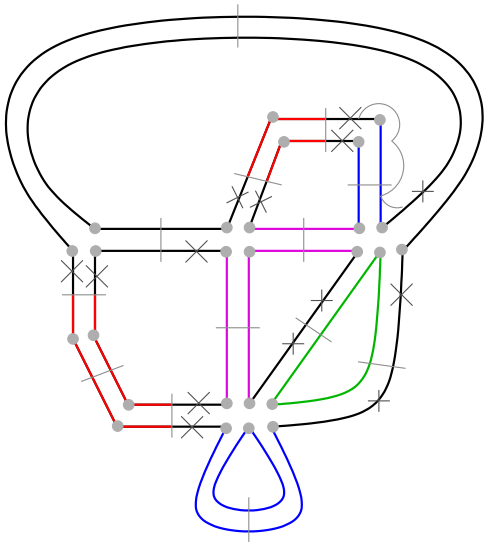
vertices to remove ; edges to remove ;
edges to contract ; face to contract ;

Suppressions et contractions simultanées

Definition

$G' = (B', \alpha'_0, \ldots, \alpha'_n)$:

- $\forall i : 0 \leq i \leq n, \forall b \in BV_i$:
  $b\alpha'_i = b' = b(\alpha_i\alpha_{l_1}) \ldots (\alpha_i\alpha_{l_r})\alpha_i$, où :
  - r est le plus petit entier tel que $b' \in BV_i$;
  - $\forall j : 1 \leq j \leq r, l_j =$
    - $i+1$ si $b_j = b(\alpha_i\alpha_{l_1}) \ldots (\alpha_i\alpha_{l_{j-1}})\alpha_i \in S_i$
    - $i-1$ sinon $(b_j \in C_i)$



vertices to remove ; edges to remove ;
edges to contract ; face to contract ;

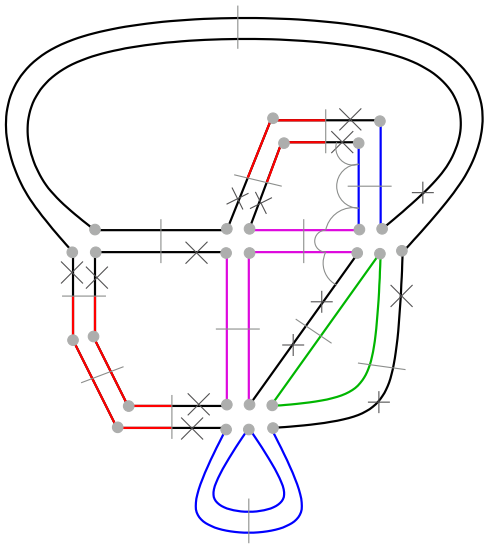
Quelques modèles Cartes combinatoires/généralisées Invariants Topologiques Algorithmes Opérations de base Cartes combinatoires dans CGAL Conclu
Generalization

Suppressions et contractions simultanées

Definition

$G' = (B', \alpha'_0, \ldots, \alpha'_n)$:

- $\forall i : 0 \leq i \leq n, \forall b \in BV_i$:
  $b\alpha'_i = b' = b(\alpha_i\alpha_{l_1}) \ldots (\alpha_i\alpha_{l_r})\alpha_i$, où :
  - r est le plus petit entier tel que $b' \in BV_i$;
  - $\forall j : 1 \leq j \leq r, l_j =$
    - $i+1$ si $b_j = b(\alpha_i\alpha_{l_1}) \ldots (\alpha_i\alpha_{l_{j-1}})\alpha_i \in S_i$
    - $i-1$ sinon $(b_j \in C_i)$



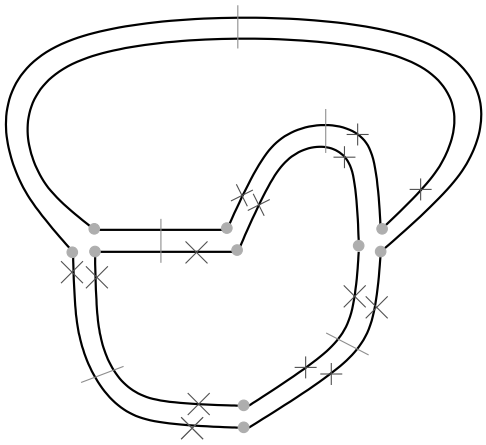
vertices to remove ; edges to remove ;
edges to contract ; face to contract ;

Suppressions et contractions simultanées

Definition

$G' = (B', \alpha'_0, \ldots, \alpha'_n)$:

- $\forall i : 0 \leq i \leq n, \forall b \in BV_i$:
  $b\alpha'_i = b' = b(\alpha_i\alpha_{l_1}) \ldots (\alpha_i\alpha_{l_r})\alpha_i$, où :
  - r est le plus petit entier tel que $b' \in BV_i$;
  - $\forall j : 1 \leq j \leq r, l_j =$
    - $i+1$ si $b_j = b(\alpha_i\alpha_{l_1}) \ldots (\alpha_i\alpha_{l_{j-1}})\alpha_i \in S_i$
    - $i-1$ sinon $(b_j \in C_i)$



vertices to remove ; edges to remove ;
edges to contract ; face to contract ;

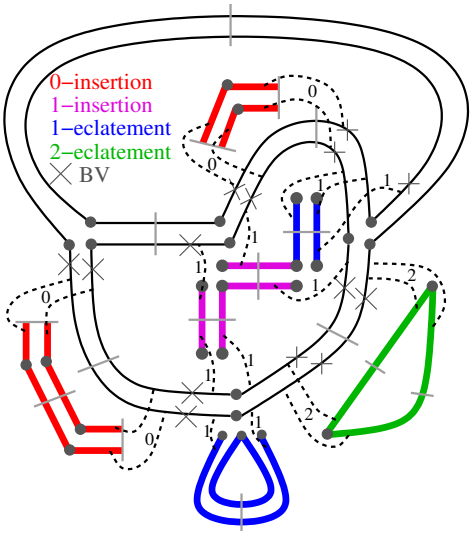
Insertions et éclatements simultanées

Definition

$G = (B, \alpha_0, \ldots, \alpha_n)$ une nG-carte
$G' = (B', \alpha_0, \ldots, \alpha_n)$ une seconde nG-carte
I les cellules à insérer
E les cellules à éclater
γ_i une involution sur $BV_i \cup BV'_i$ avec
$b \in BV_i \Leftrightarrow b\gamma_i \in BV'_i$

Préconditions :

- $\forall c, c' \in I \cup E : c \cap c' = \emptyset$
- les cellules de I sont supprimables ;
- les cellules de E sont contractibles ;
- toutes les cellules de G' sont à insérer ou à éclater : $I \cup E = B'$

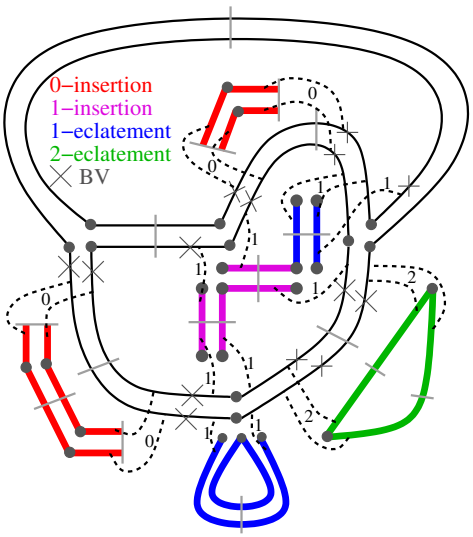


Insertions et éclatements simultanées

Definition

Préconditions pour l'opération : $\forall i$:

- $\forall b \in BV'_i$: b est i-libre ;
- $\forall b \in BV_i \cup BV'_i$: $\forall j : 0 \leq j \leq n$ tel que $|i-j| \geq 2$ et $b\alpha_j \in BV_i \cup BV'_i$: $b\alpha_j\gamma_i = b\gamma_i\alpha_j$;
- $\forall b \in BV_i$: $b\alpha_i = b' = b\gamma_i\alpha_{l_1}(\alpha_i\alpha_{l_2})\ldots(\alpha_i\alpha_{l_k})\gamma_i$
  - k le plus petit entier tel que $b' \in BV_i$;
  - $l_1 = i+1$ si $b\gamma_i \in I$; $l_1 = i-1$ sinon
  - $\forall j : 2 \leq j \leq k$, $l_j =$
    - $i+1$ *si* $b_j = b\gamma_i\alpha_{l_1}(\alpha_i\alpha_{l_2})\ldots(\alpha_i\alpha_{l_{j-1}}) \in I$
    - $i-1$ *sinon* (c-à-d $b_j \in E$)

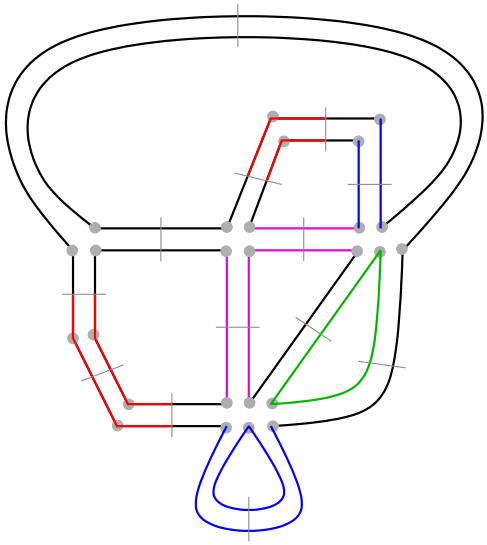


Quelques modèles Cartes combinatoires/généralisées Invariants Topologiques Algorithmes Opérations de base Cartes combinatoires dans CGAL Conclu
Insertions et éclatements
Insertions et éclatements simultanées

Definition

$G'' = (B'', \alpha''_0, \ldots, \alpha''_n)$ est définie par :

1. $B'' = B \cup B'$;
2. $\forall i : 0 \leq i \leq n$, $\forall b \in B'' - (BV_i \cup BV'_i)$: $b\alpha''_i = b\alpha_i$;
3. $\forall i : 0 \leq i \leq n$, $\forall b \in BV_i \cup BV'_i$: $b\alpha''_i = b\gamma_i$.



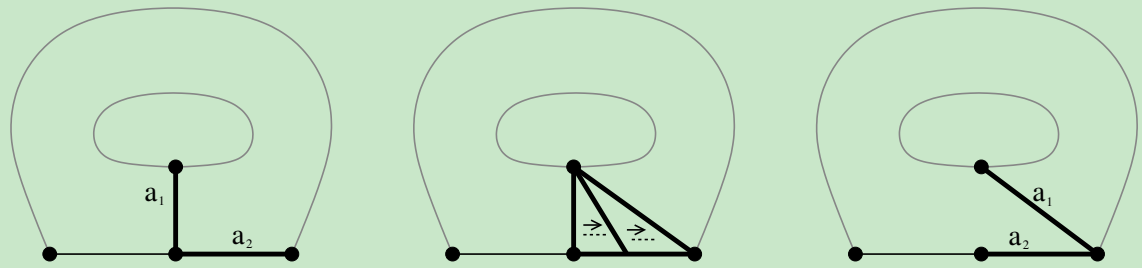
Utilisation des Cartes pour le Traitement d'Images et la Modélisation Géométrique Guillaume Damiand 65 / 88

Décalage d'arêtes

Principe

Pousser une arête le long d'une de ses voisines

Example



Intuitivement : revient à supprimer localement l'arête puis à l'insérer

Suppressions et Contractions pour les Cartes

- Toute G-carte orientable $\Leftrightarrow$ une carte combinatoire
- les opérations suppressions et contractions préservent l'orientation

$\Rightarrow$ les opérations sont valides pour les cartes.
Mais plus complexe car à chaque fois cas particulier pour β_1

(Exemple Moka)