Algorithm to Turn One Oriented Triangular Mesh Connectivity into Another

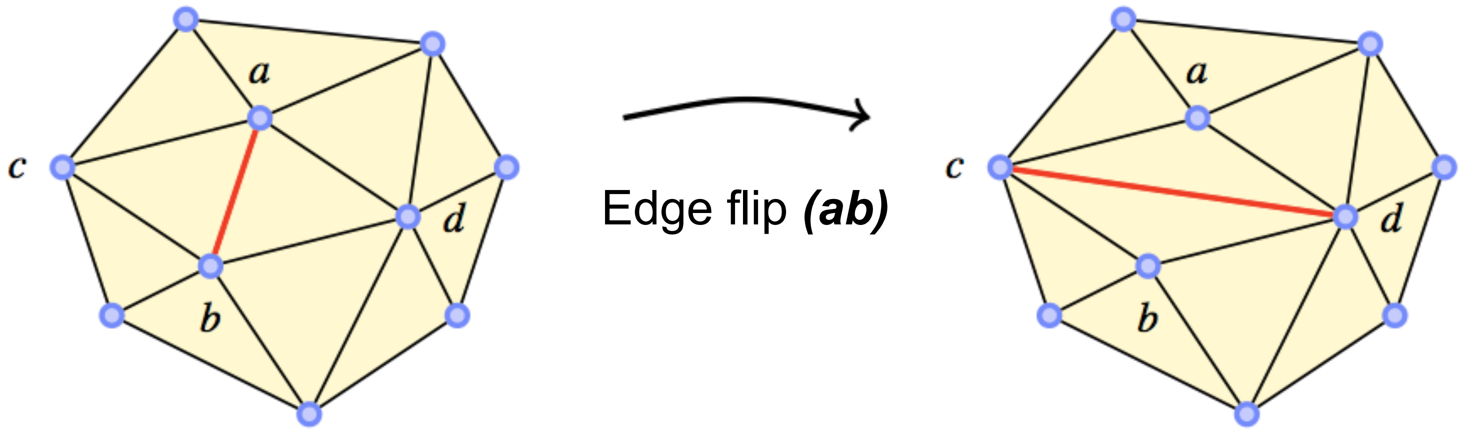
Jérémy Espinas Raphaëlle Chaine Pierre-Marie Gandoin

Lyon 1

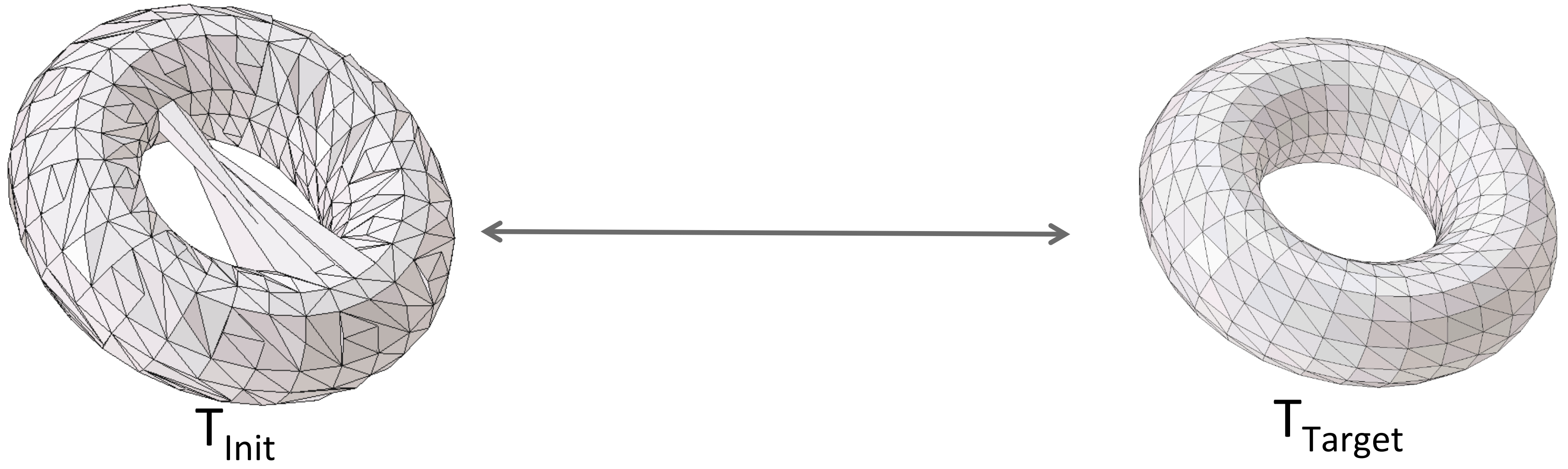
Journées de Géométrie Algorithmique 2012

DU 2 AU 6 AVRIL 2012

Introduction



Goal : find a sequence of edge flips between two triangulations.



Outline

I. Construct one edge in a triangulation that may contain constrained unflippable edges

II. Construct the edges of T_{target} on the evolving mesh using a strategy that converges towards the connectivity of T_{target}

Outline

I. Construct one edge in a triangulation that may contain constrained unflippable edges

II. Construct the edges of T_{target} on the evolving mesh using a strategy that converges towards the connectivity of T_{target}

State of the art about algorithm for determining a sequence of edge flips

Combinatorial Setting

• determining on edge flips sequence between two triangulations

- on the plane [Wagner 1936]
- on the torus [Dewdney 1976]
- on the Klein bottle [Negami Watanabe 1990]
- in the general case [Negami 1999]

Geometrical Setting

Flip conditioned by some geometric criteria

• determining on edge flips sequence between two triangulations

- on the plane [Lawson 1972]

A complete state of the art [Bose Hurtado 2009]

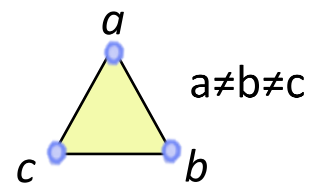
The class of triangulations

We work within oriented triangulations :

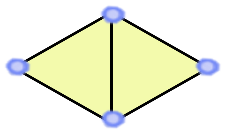
- composed of a single connected component,

- may contain boundaries,

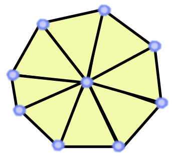
- every facet has three distinct vertices,



- every edge is incident to at most two facets oriented consistently,

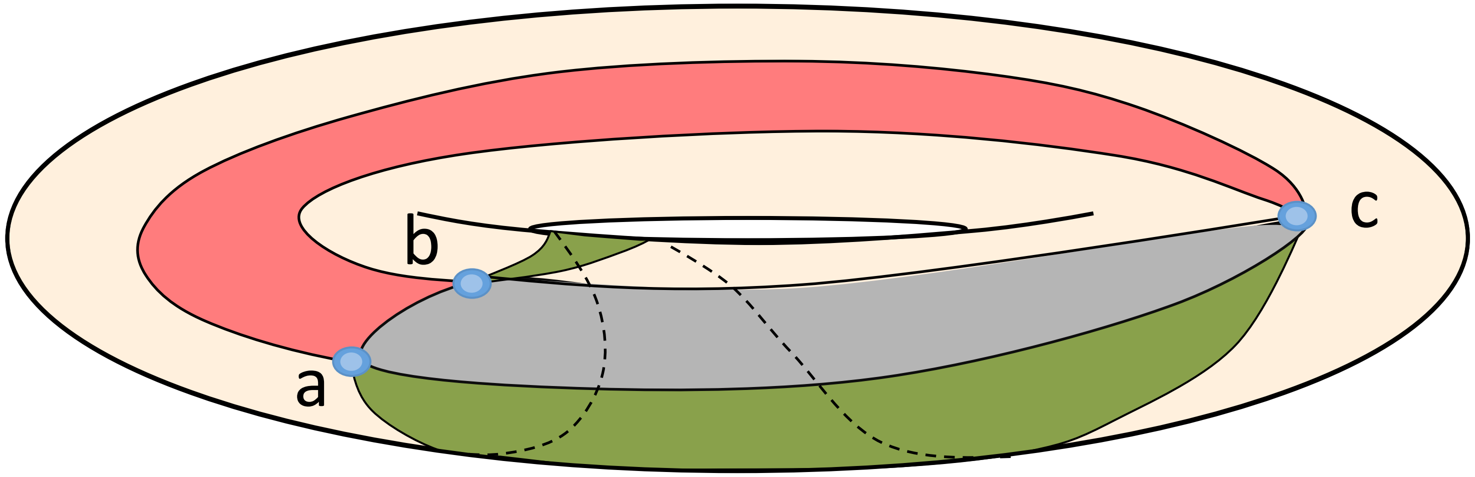


- the set of facets incident to one vertex must form a topological or half topological disk.



The class of triangulations

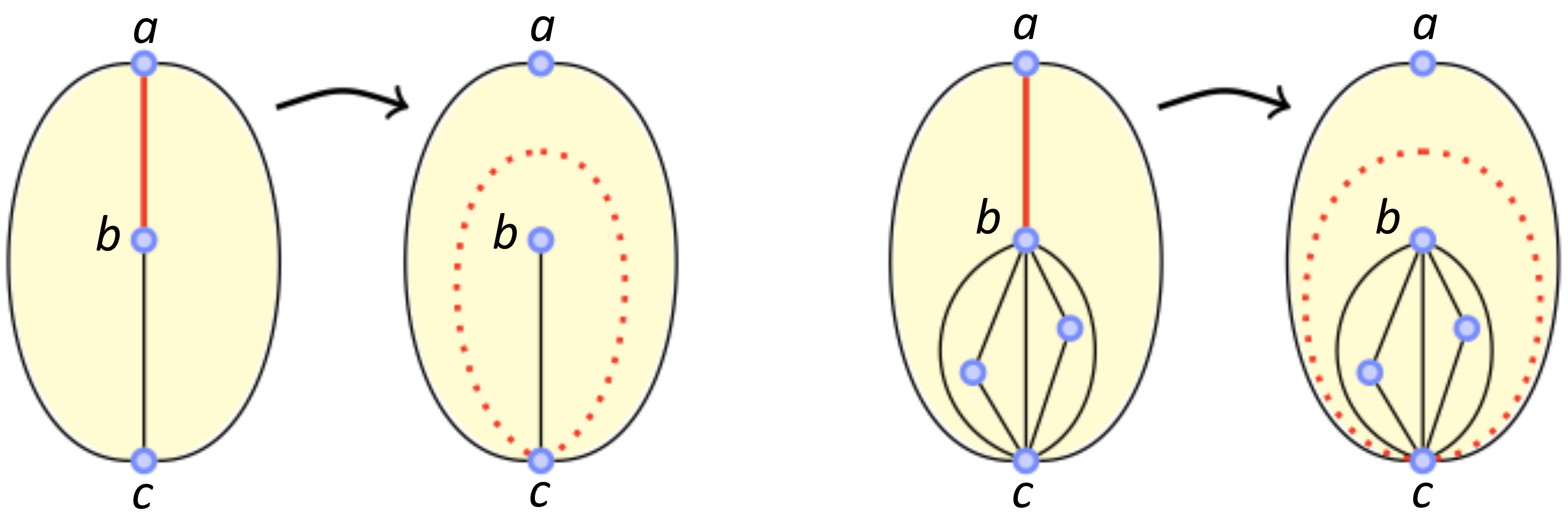
Consequence : it is possible to have multiple facets that share the same three vertices



The class of triangulations

Unflippable edge : the flip of the edge *(ab)* will be forbidden whenever it would result in the creation of an edge connecting a vertex *c* to itself

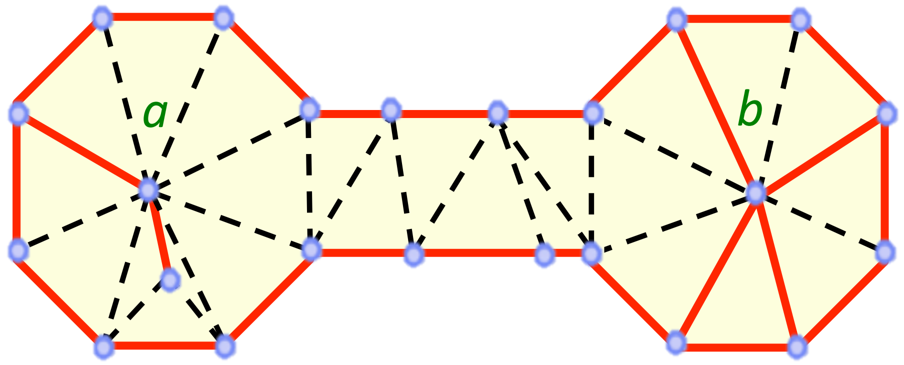
(i.e. when the two facets incident to *(ab)* are based on the same three vertices)

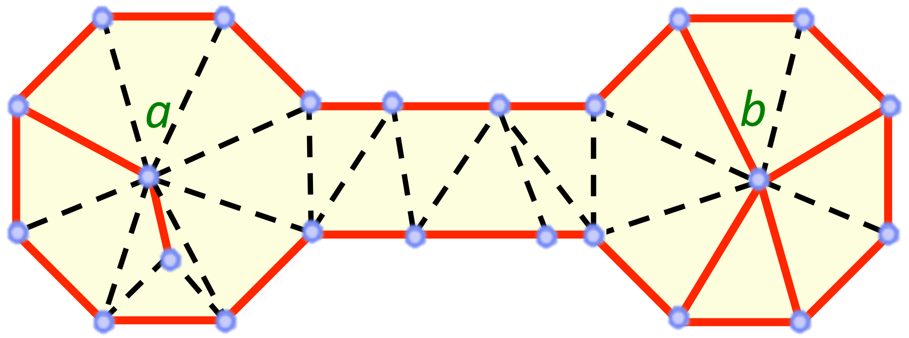


Constrained edges will also considered as unflippable

Construction of an edge (ab) into a constrained triangulation

- several edges *(ab)* depending on their position with regards to the order of constrained edges around *a* and *b*

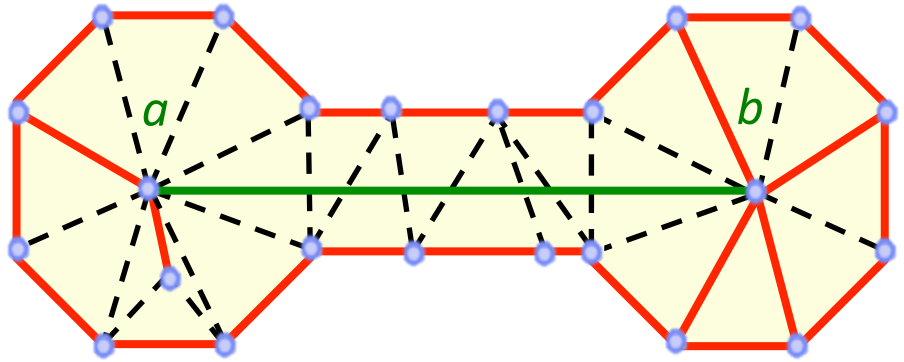


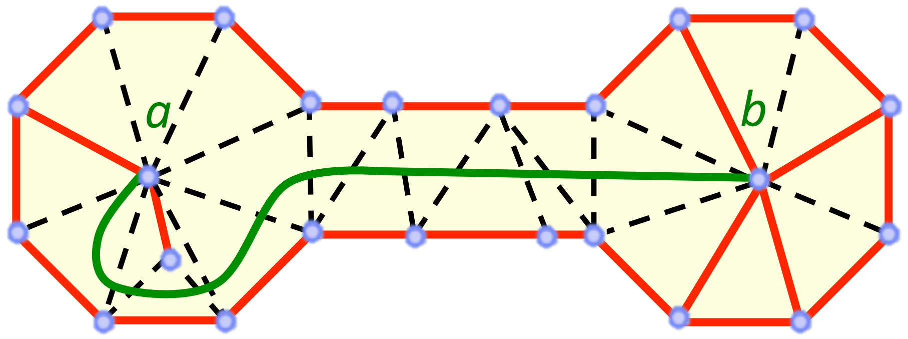


red lines : constrained edges

Construction of an edge (ab) into a constrained triangulation

• several edges *(ab)* depending on their position with regards to the order of constrained edges around *a* and *b*

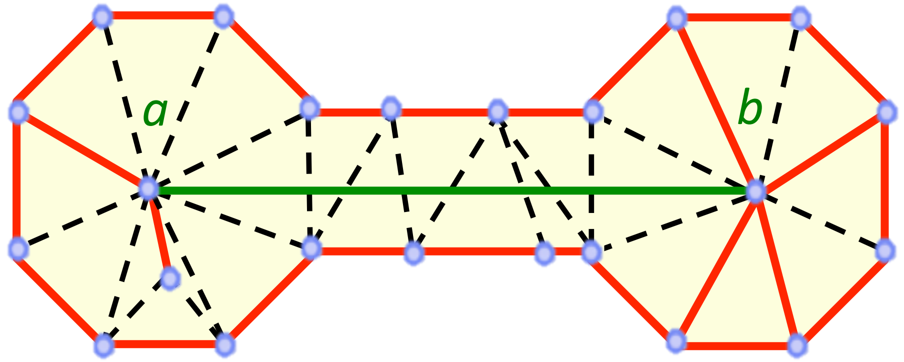


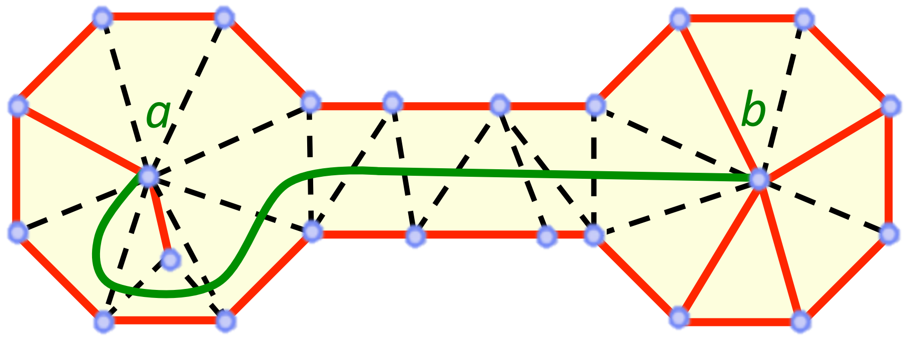


red lines : constrained edges

Construction of an edge (ab) into a constrained triangulation

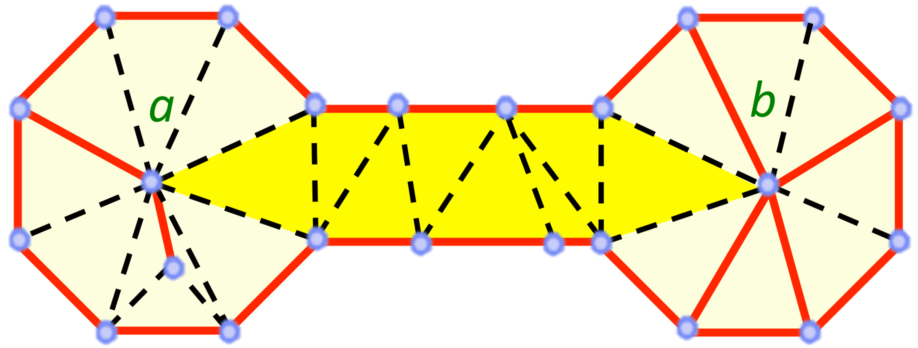
• several edges *(ab)* depending on their position with regards to the order of constrained edges around *a* and *b*

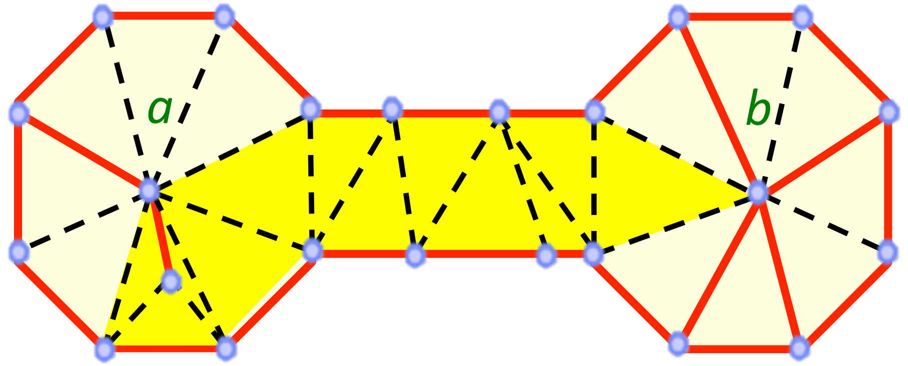




red lines : constrained edges

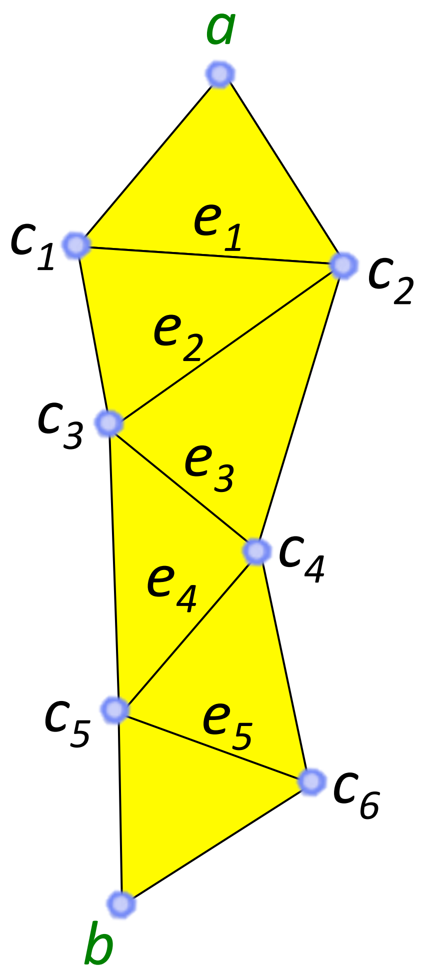
To construct the desired edge *(ab)*, we first determine a « good » simple path of facets between *a* and *b* :





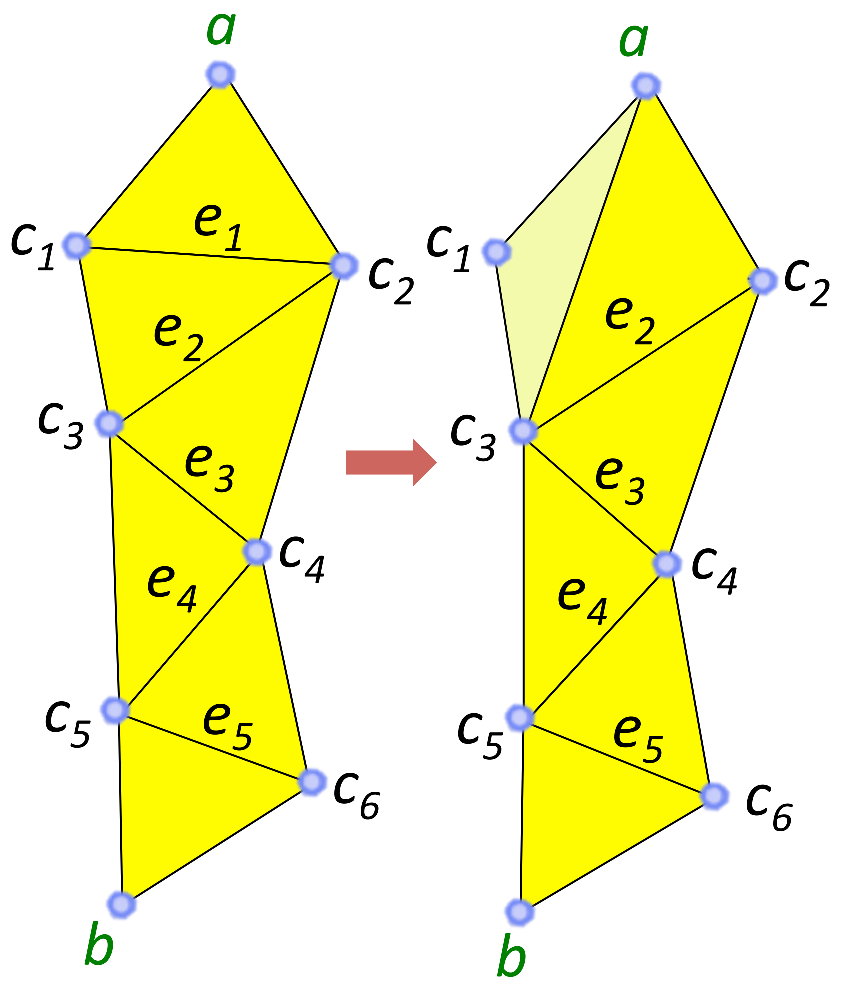
Reduce a path of facets

Case 1 : if e_1 is flippable
(ie $a \neq c_3$).



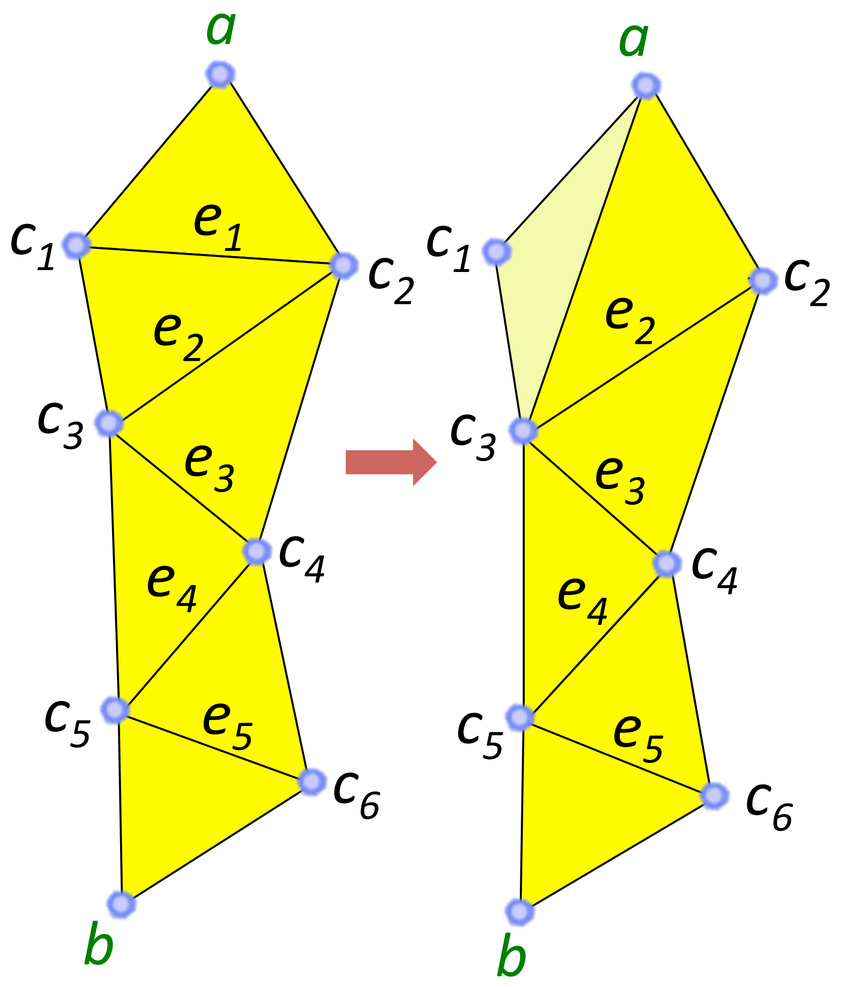
Reduce a path of facets

Case 1 : if e_1 is flippable (ie $a \neq c_3$).



Reduce a path of facets

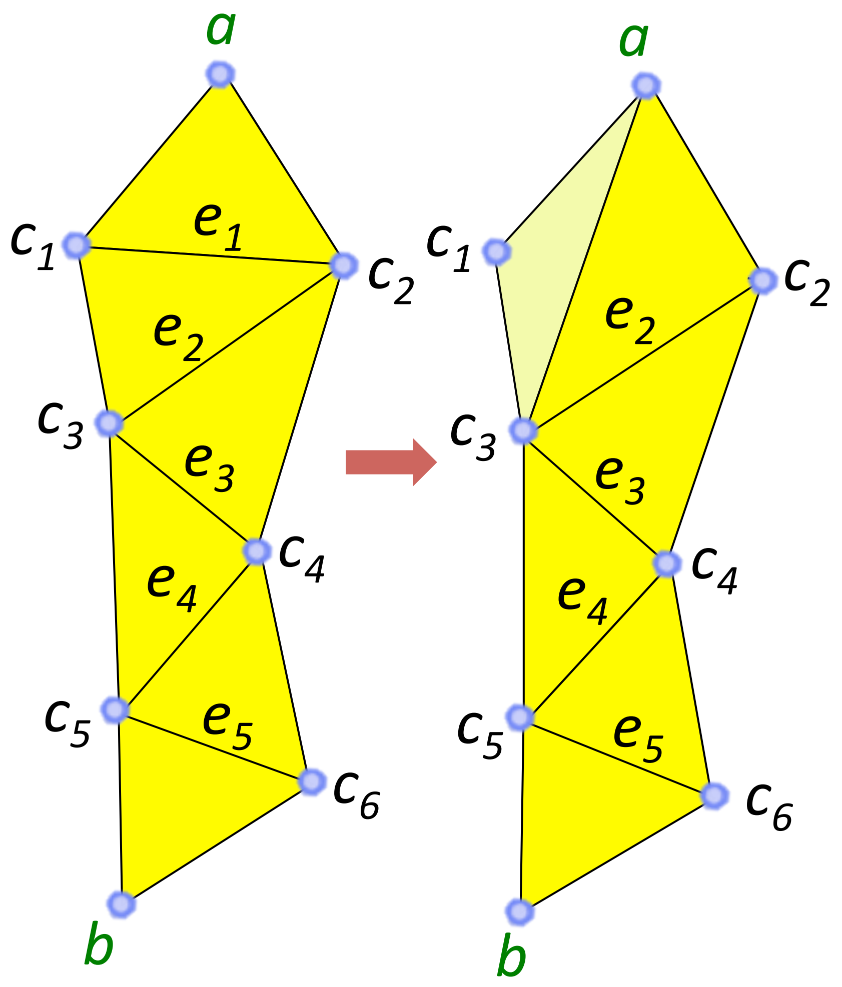
Case 1 : if e_1 is flippable
(ie $a \neq c_3$).



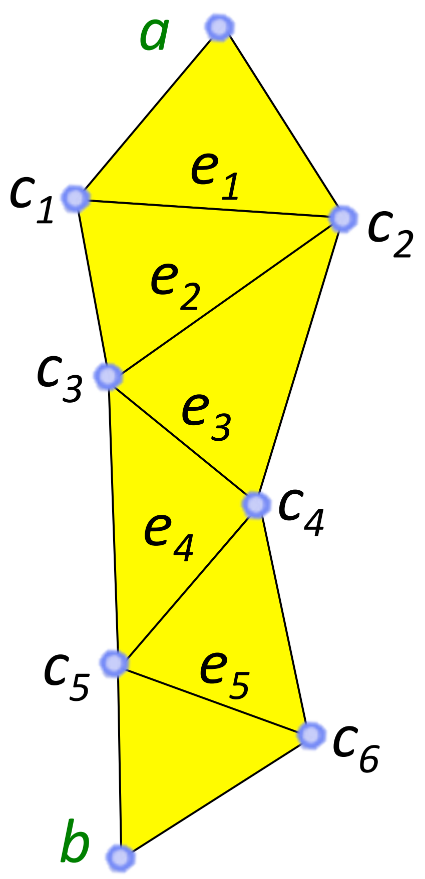
-> The length of this new path is reduced by one with respect to the initial path.

Reduce a path of facets

Case 1 : if e_1 is flippable (ie $a \neq c_3$).

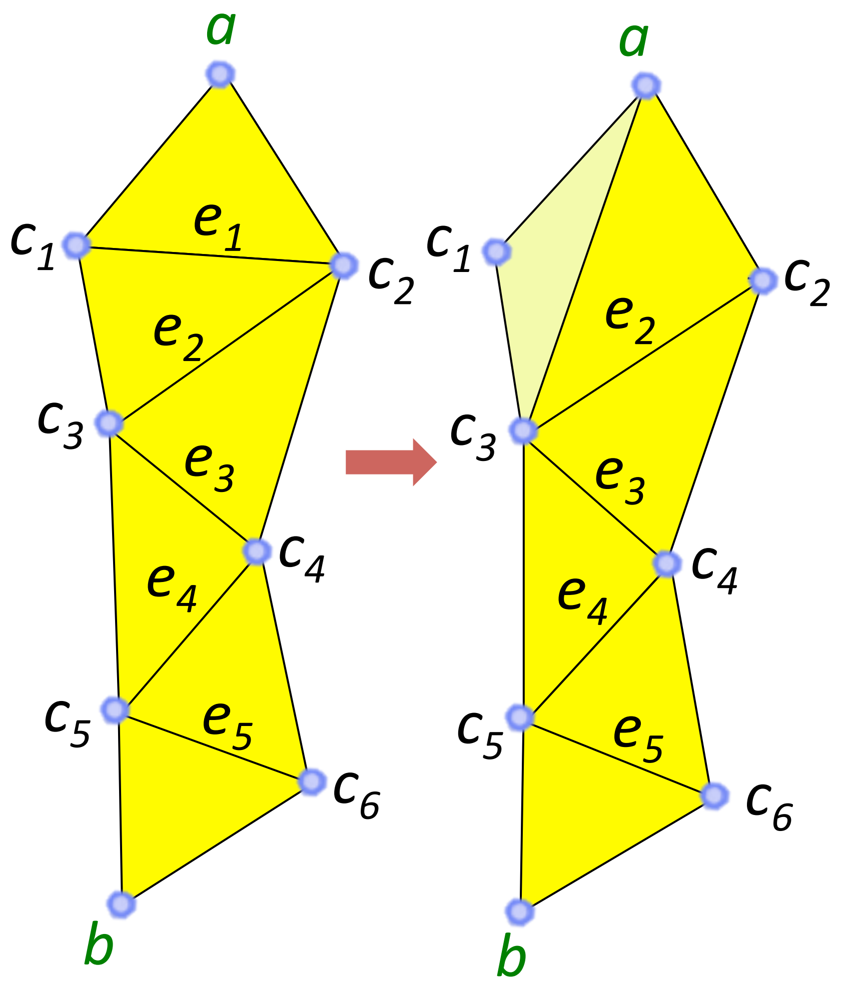


Note : if a is a vertex different than $c_1, \ldots, c_6$:

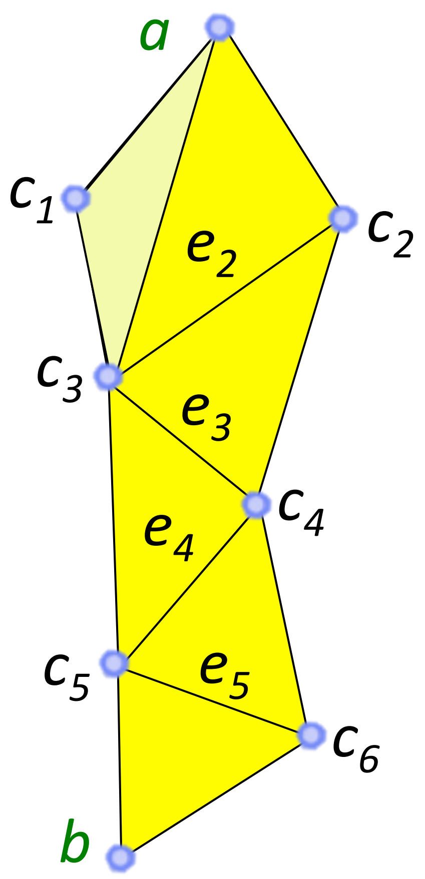


Reduce a path of facets

Case 1 : if e_1 is flippable
(ie $a \neq c_3$).

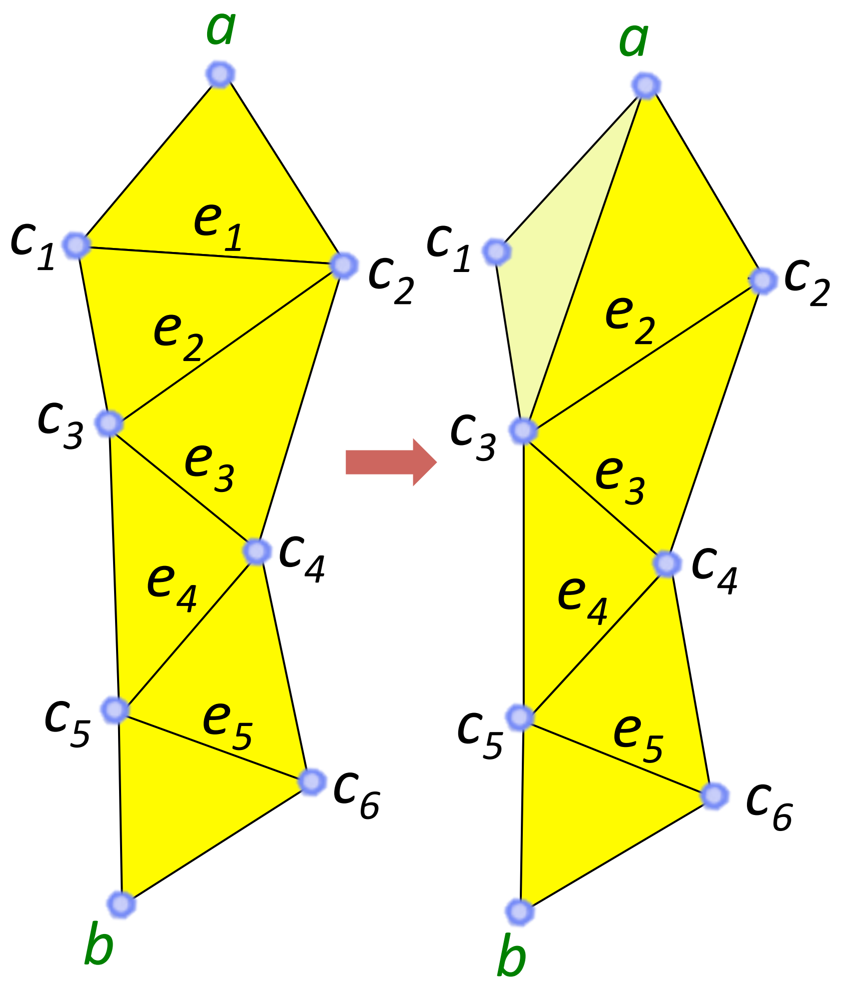


Note : if a is a vertex different than c_1, …, c_6 :

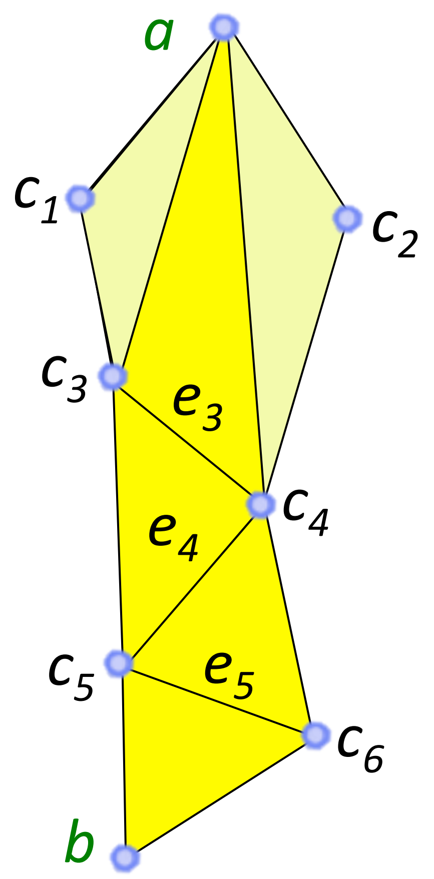


Reduce a path of facets

Case 1 : if e_1 is flippable
(ie $a \neq c_3$).

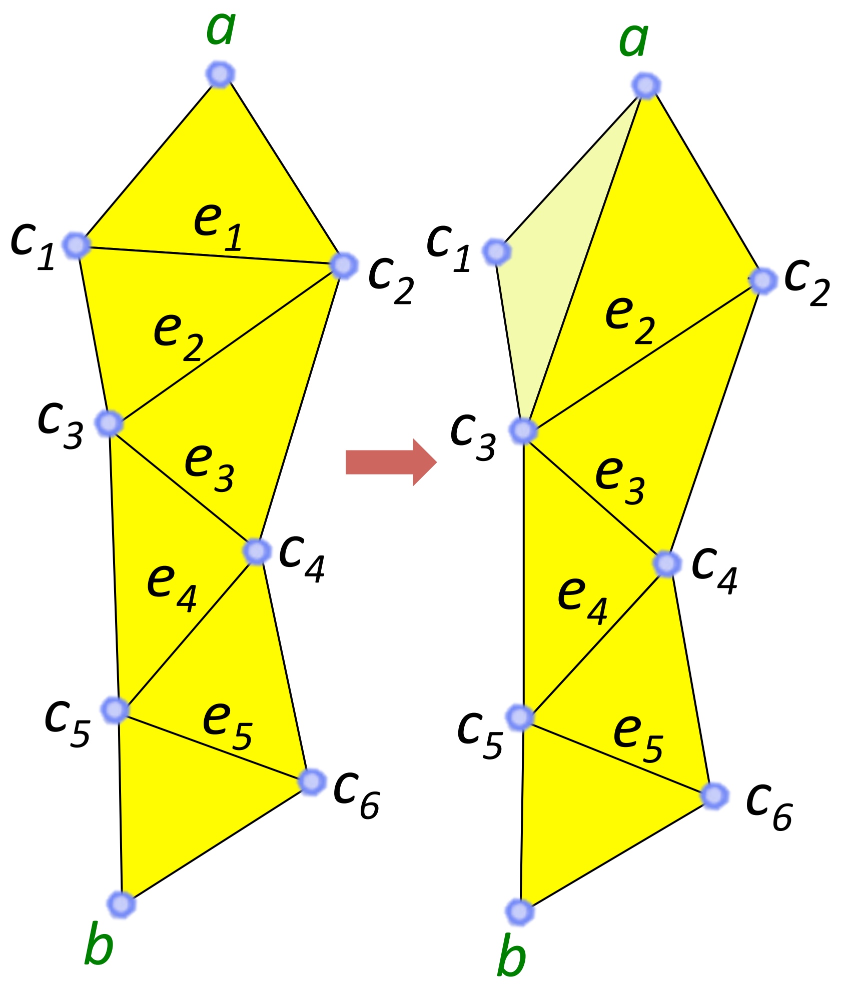


Note : if a is a vertex different than $c_1, \ldots, c_6$:

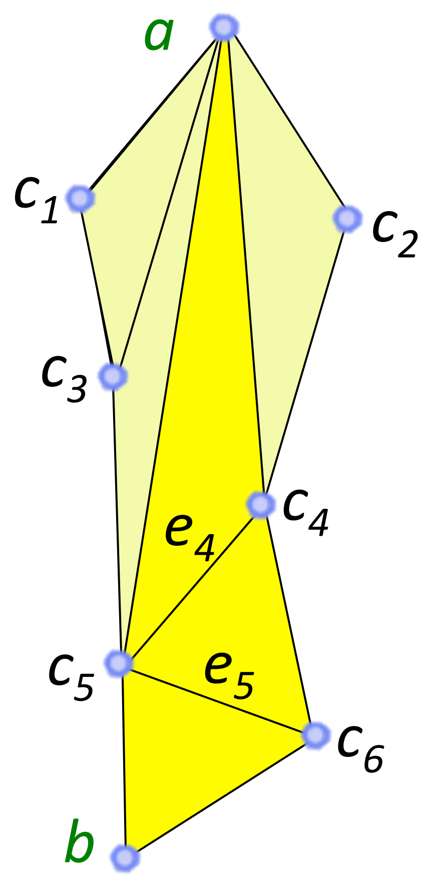


Reduce a path of facets

Case 1 : if e_1 is flippable (ie $a \neq c_3$).

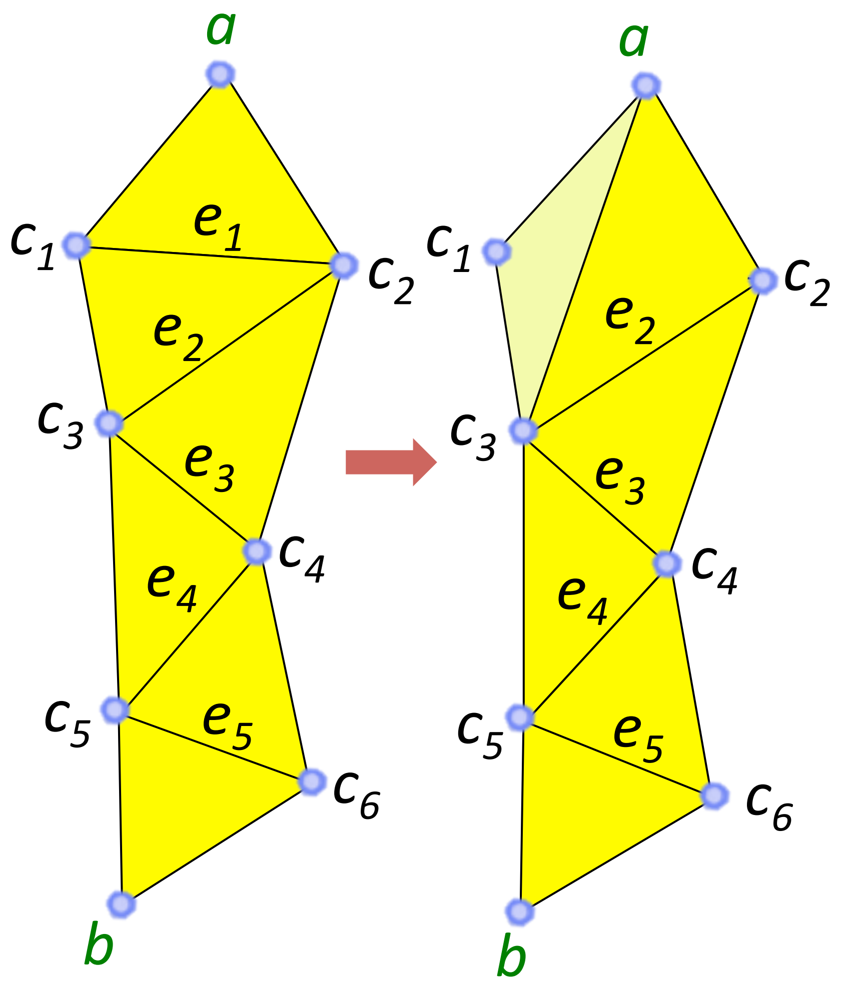


Note : if a is a vertex different than c_1, …, c_6 :

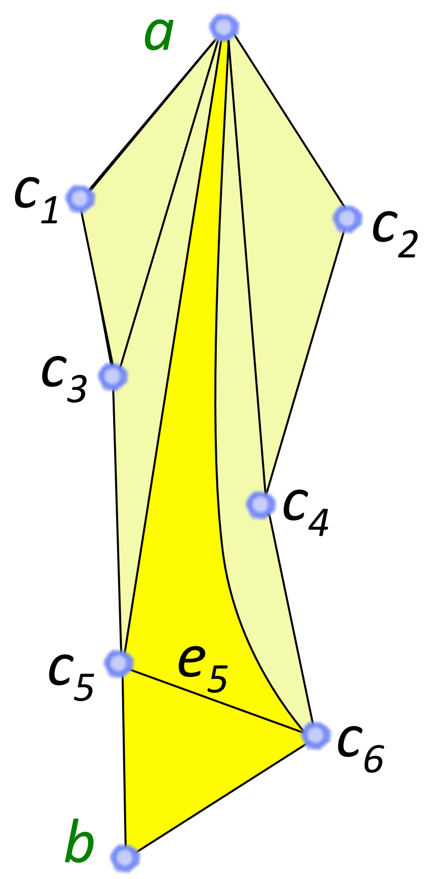


Reduce a path of facets

Case 1 : if e_1 is flippable (ie $a \neq c_3$).

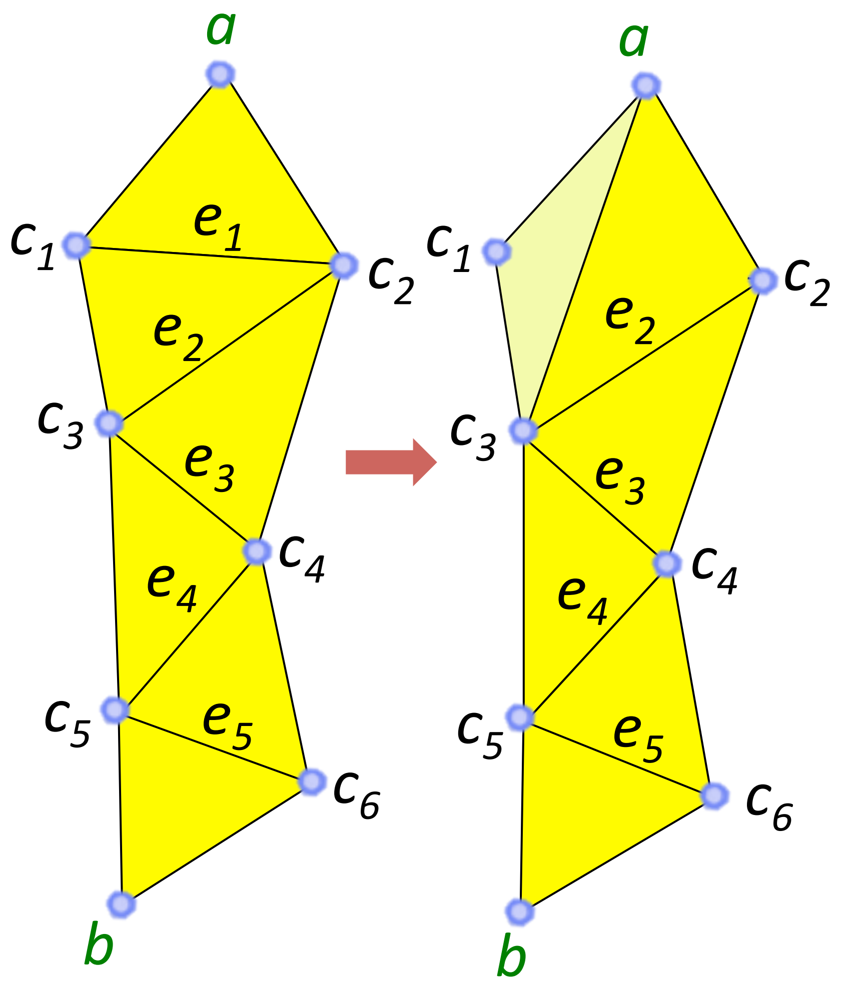


Note : if a is a vertex different than c_1, …, c_6 :

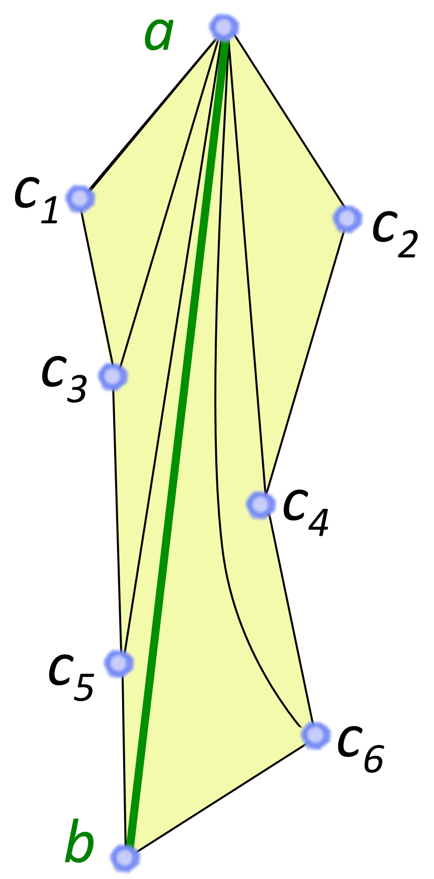


Reduce a path of facets

Case 1 : if e_1 is flippable (ie $a \neq c_3$).

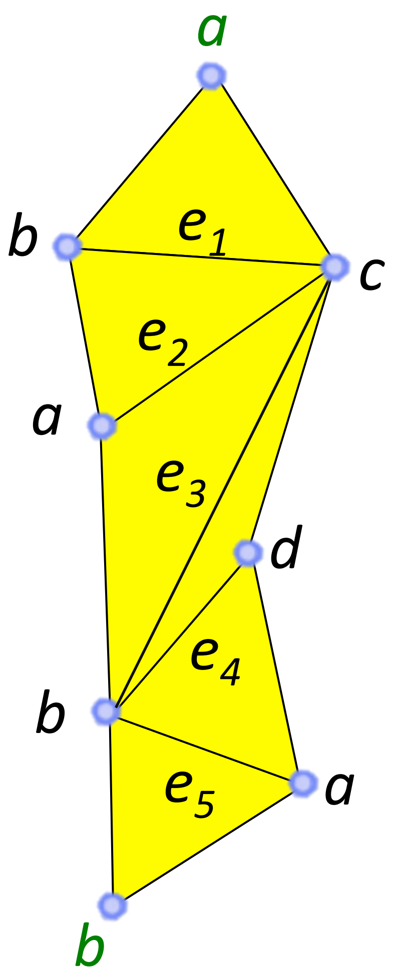


Note : if a is a vertex different than c_1, …, c_6 :



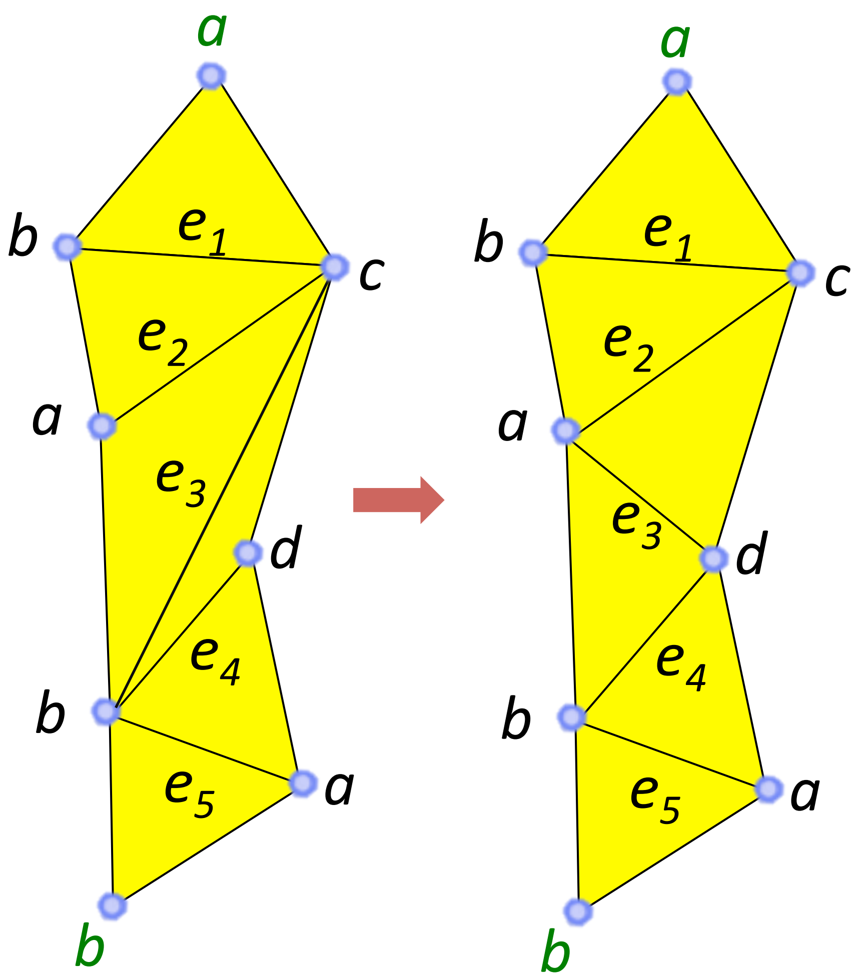
Reduce a path of facets

Case 2 : if e_1, …, e_{i-1} are unflippable, but not e_i



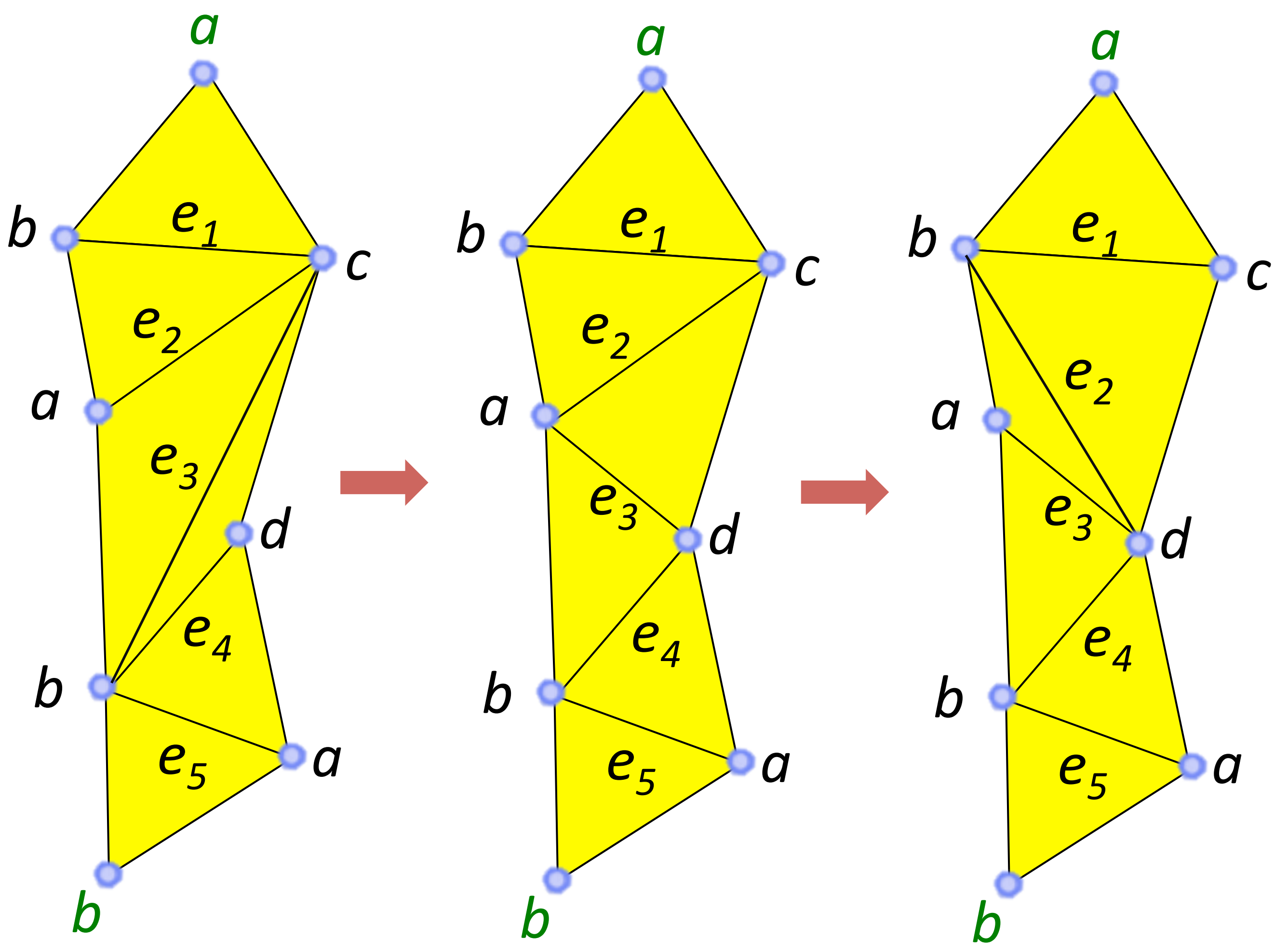
Reduce a path of facets

Case 2 : if e_1, …, e_{i-1} are unflippable, but not e_i



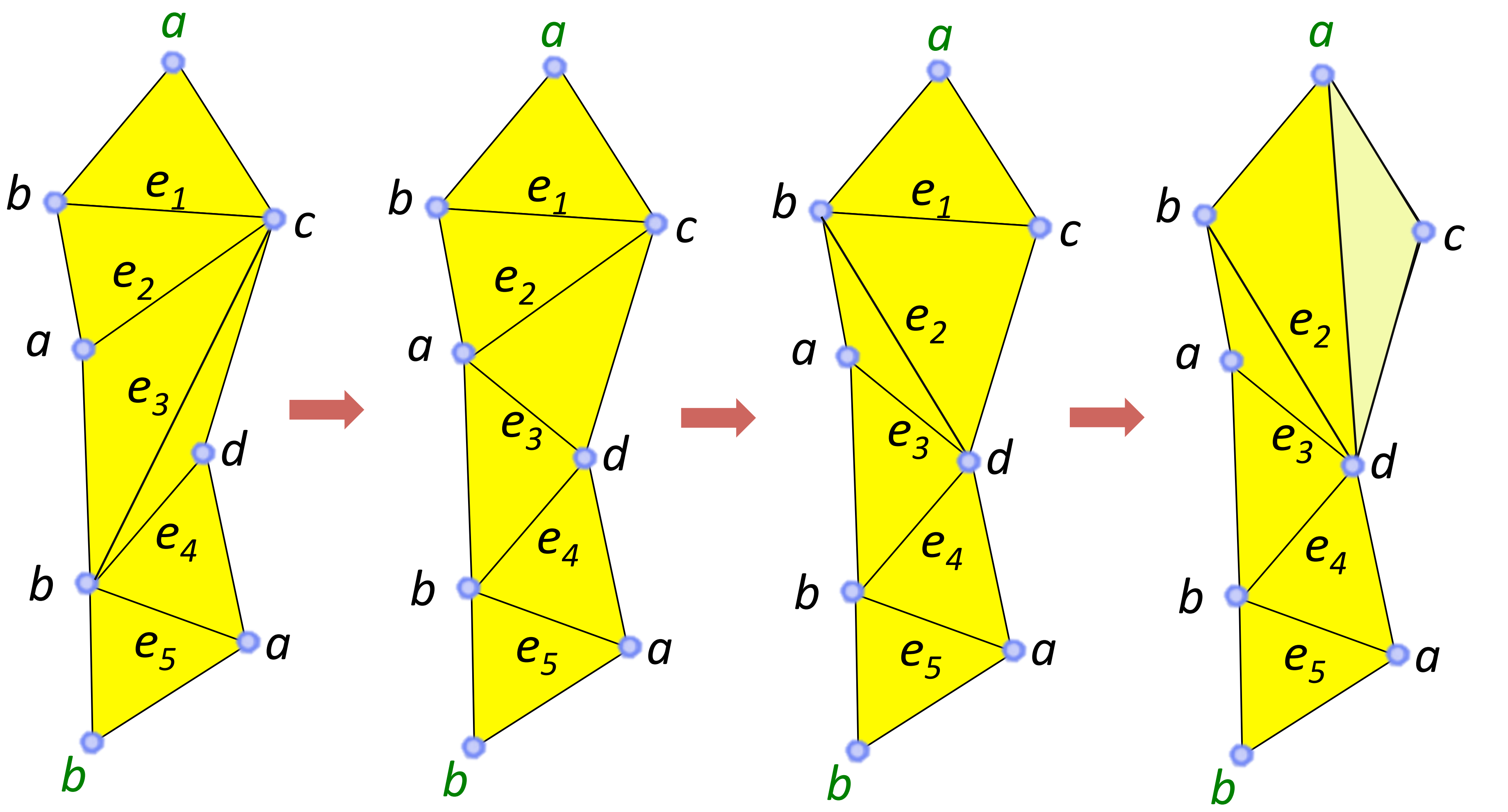
Reduce a path of facets

Case 2 : if e_1, …, e_{i-1} are unflippable, but not e_i



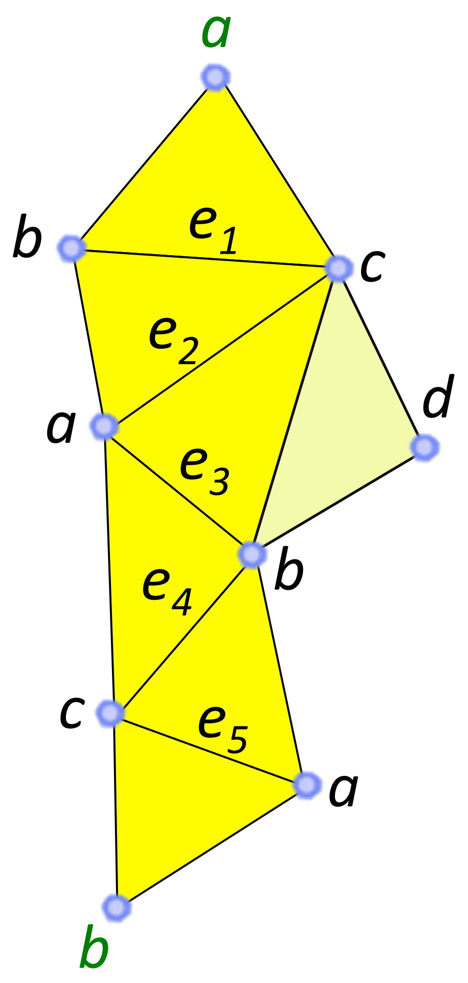
Reduce a path of facets

Case 2 : if $e_1, \ldots, e_{i-1}$ are unflippable, but not e_i



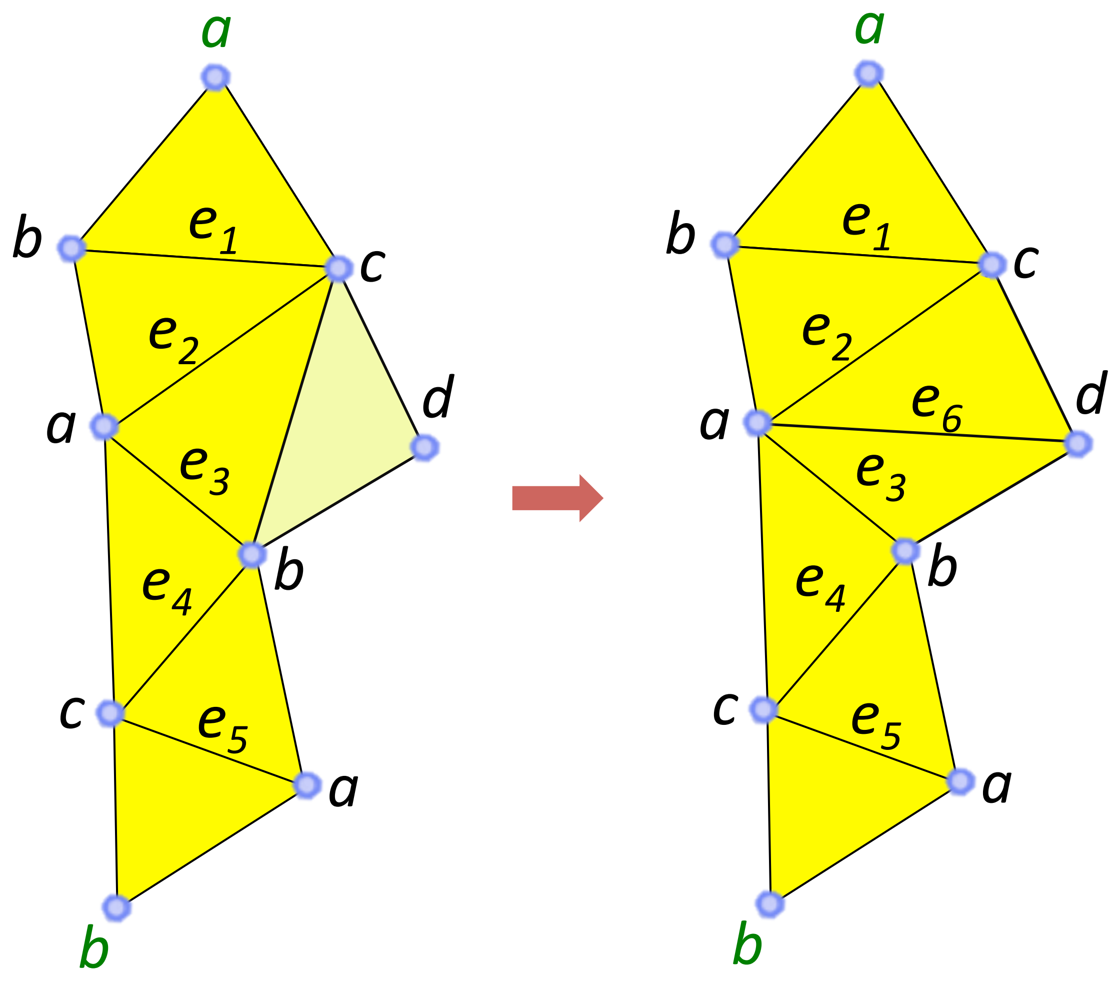
Reduce a path of facets

Case 3 : if all the edges are unflippable



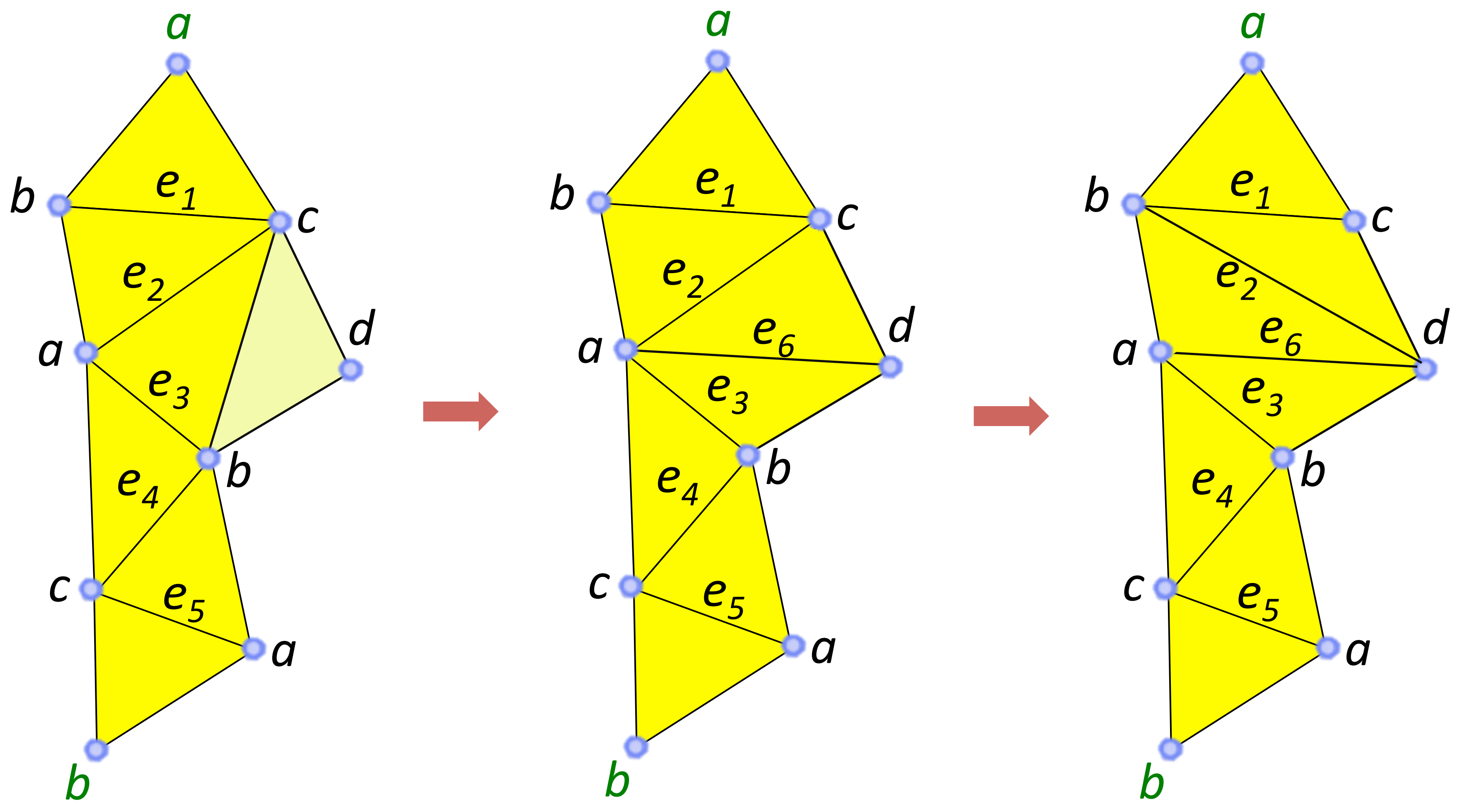
Reduce a path of facets

Case 3 : if all the edges are unflippable



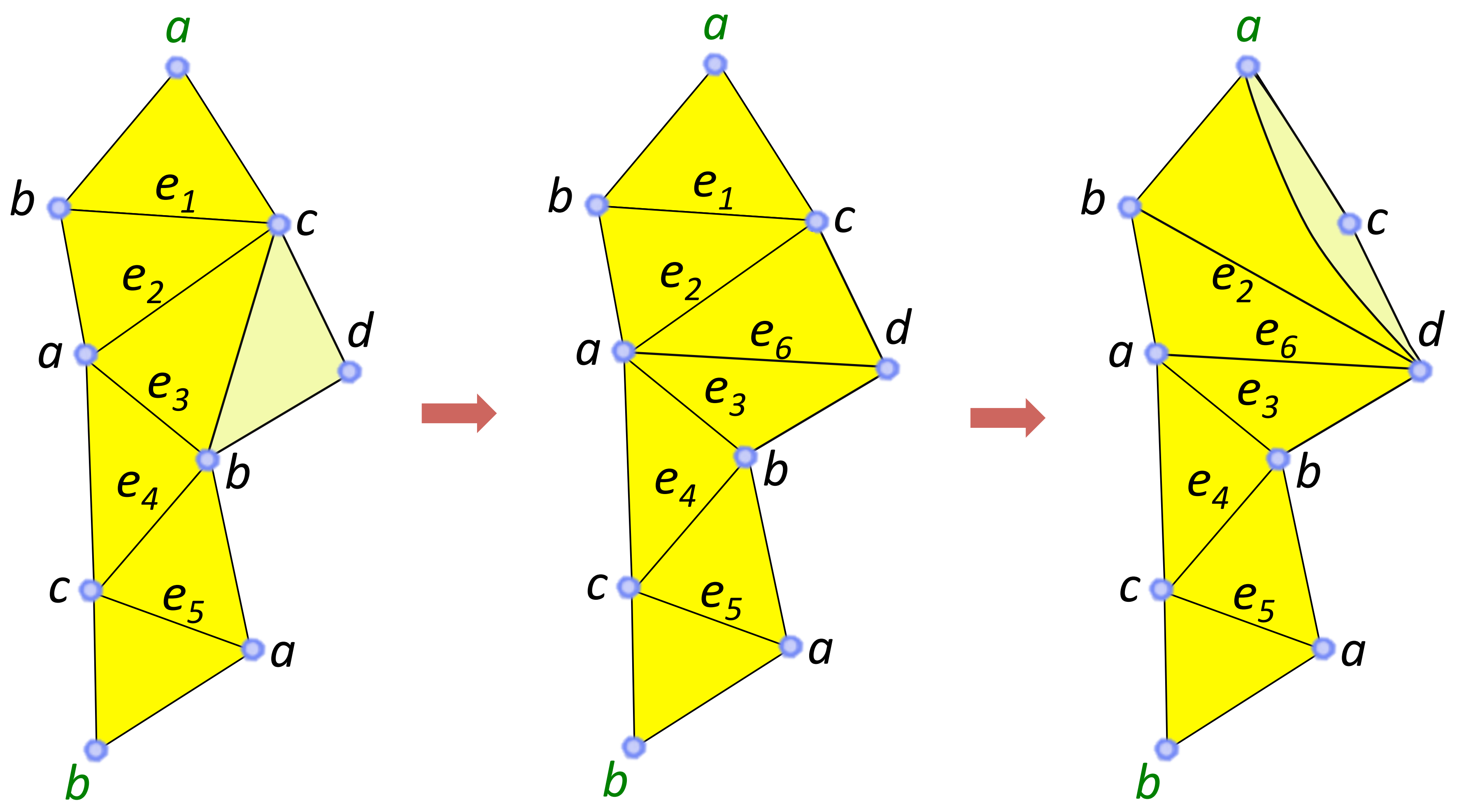
Reduce a path of facets

Case 3 : if all the edges are unflippable



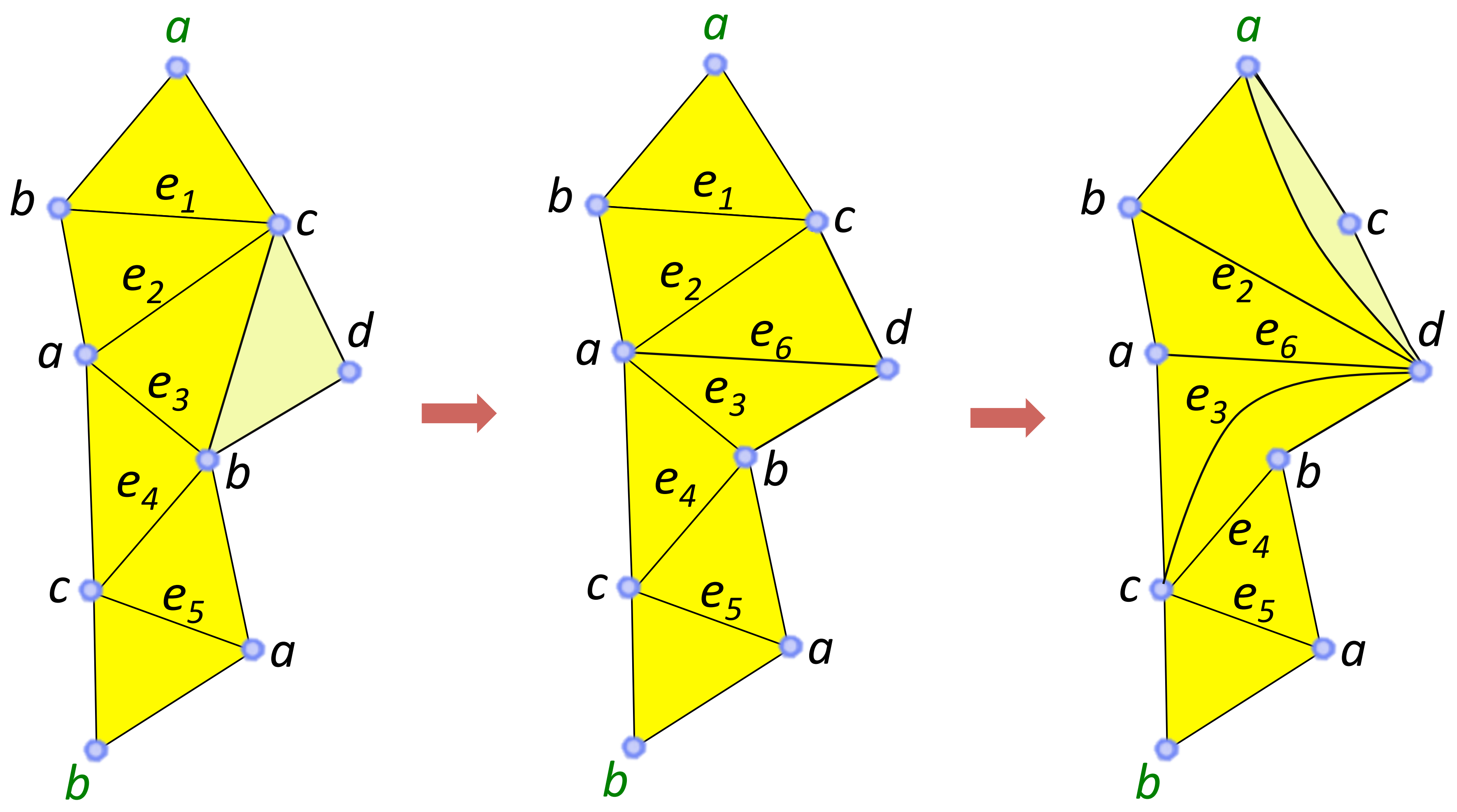
Reduce a path of facets

Case 3 : if all the edges are unflippable



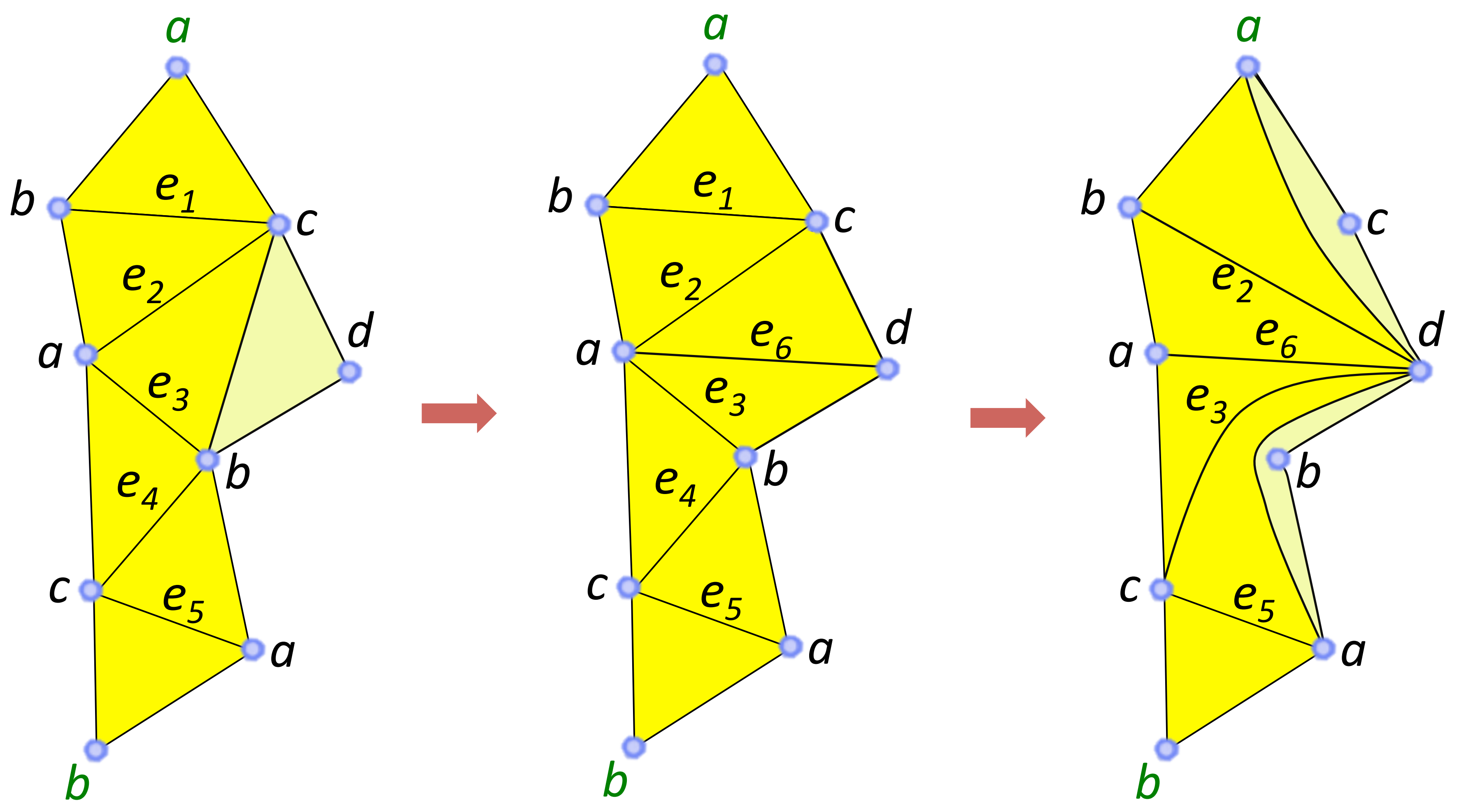
Reduce a path of facets

Case 3 : if all the edges are unflippable



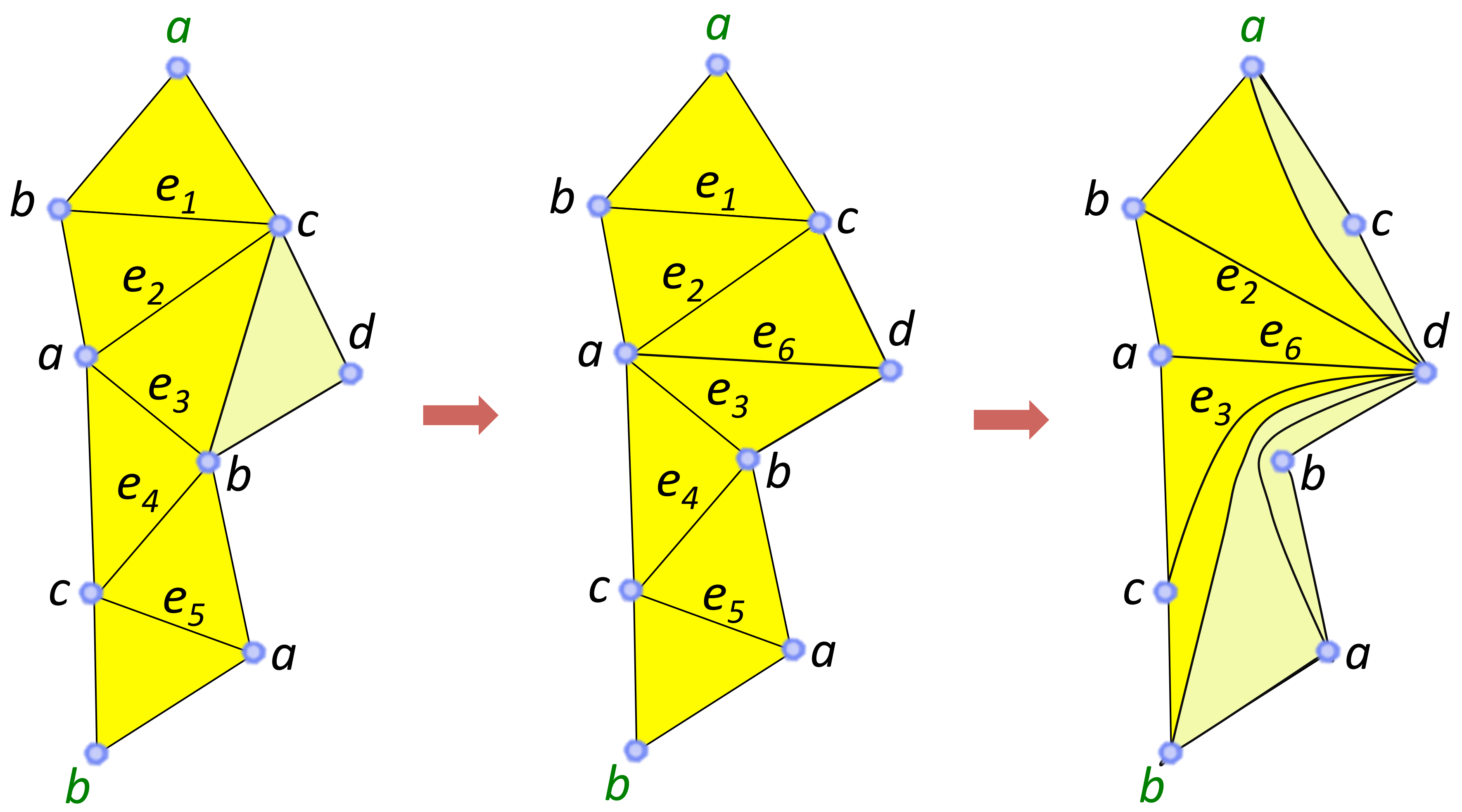
Reduce a path of facets

Case 3 : if all the edges are unflippable



Reduce a path of facets

Case 3 : if all the edges are unflippable



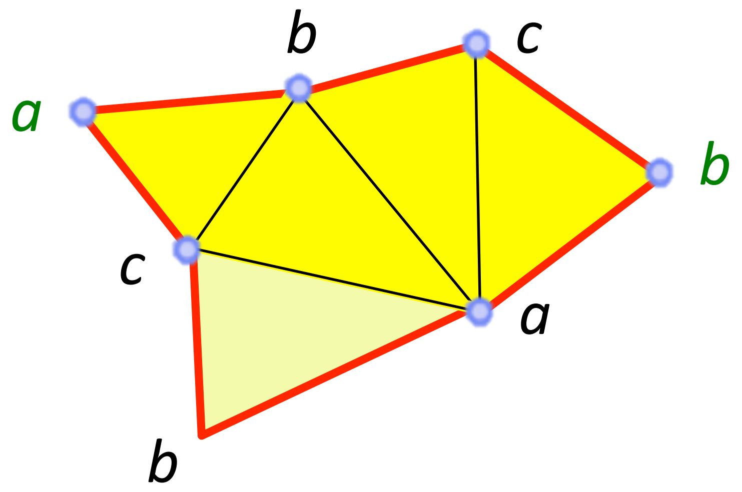
Reduce a path of facets

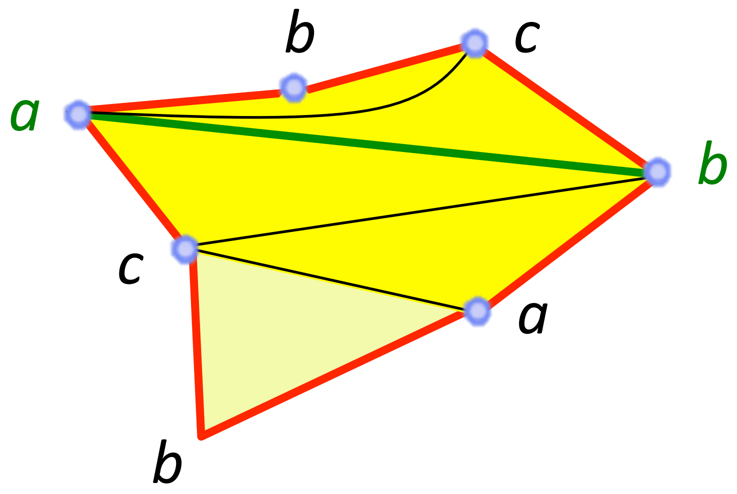
But, it is not always possible to reduce the path…

Reduce a path of facets

But, it is not always possible to reduce the path…

If the enclosing region(*) contains only three different vertices :



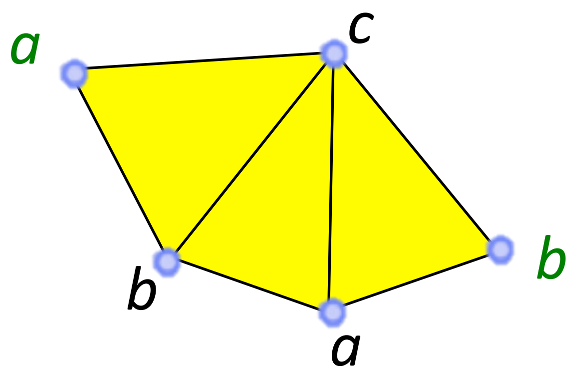


(*) **Enclosing region** : connected component with regards to the adjacency of facets across unconstrained edges

Reduce a path of facets

But, it is not always possible to reduce the path…

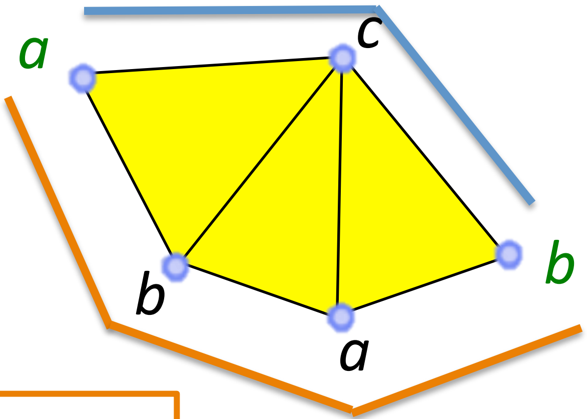
If the enclosing region contains more than four different vertices :



Reduce a path of facets

But, it is not always possible to reduce the path…

If the enclosing region contains more than four different vertices :

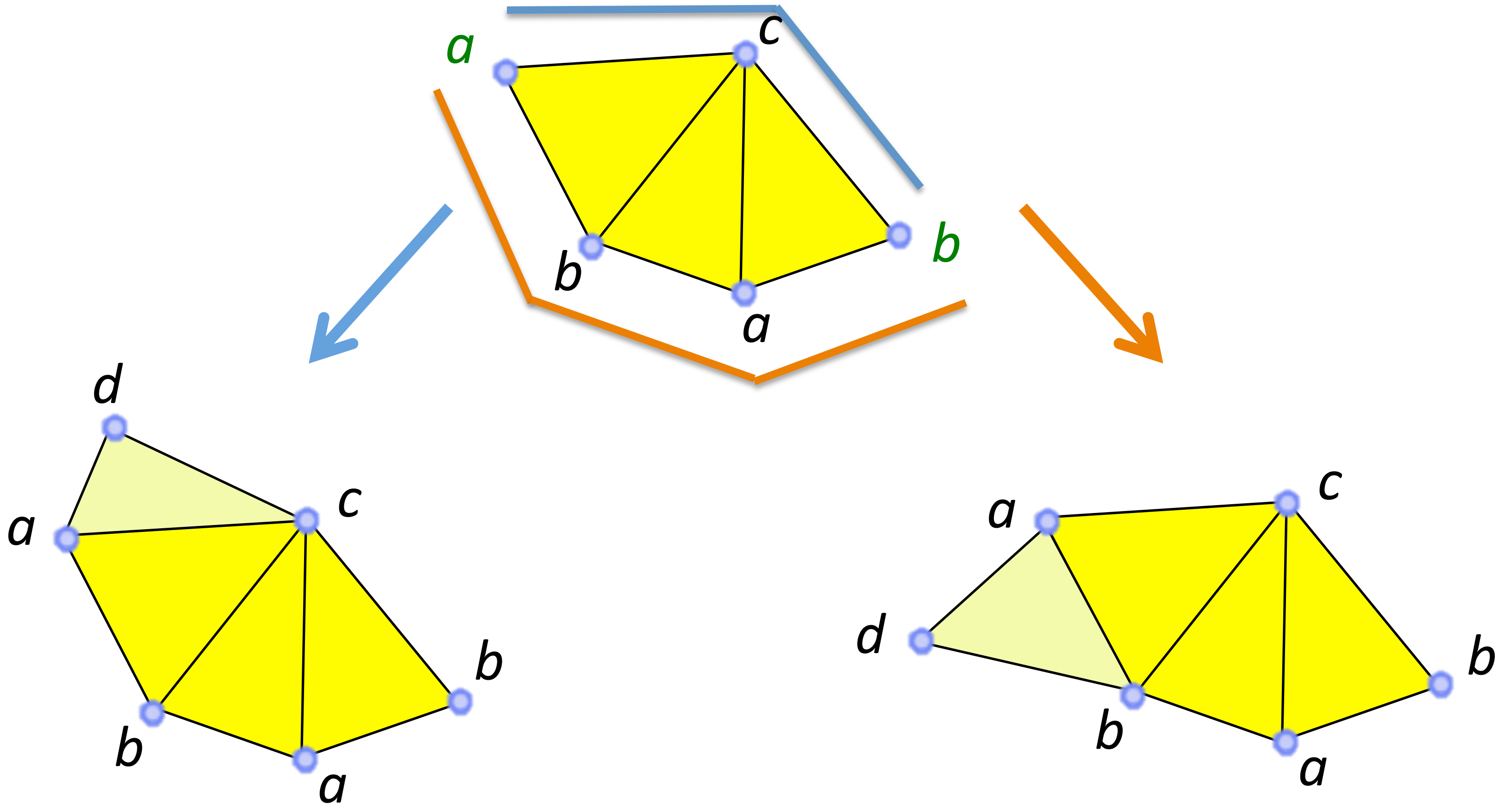


Half-envelope which contains only two different vertices

Reduce a path of facets

But, it is not always possible to reduce the path…

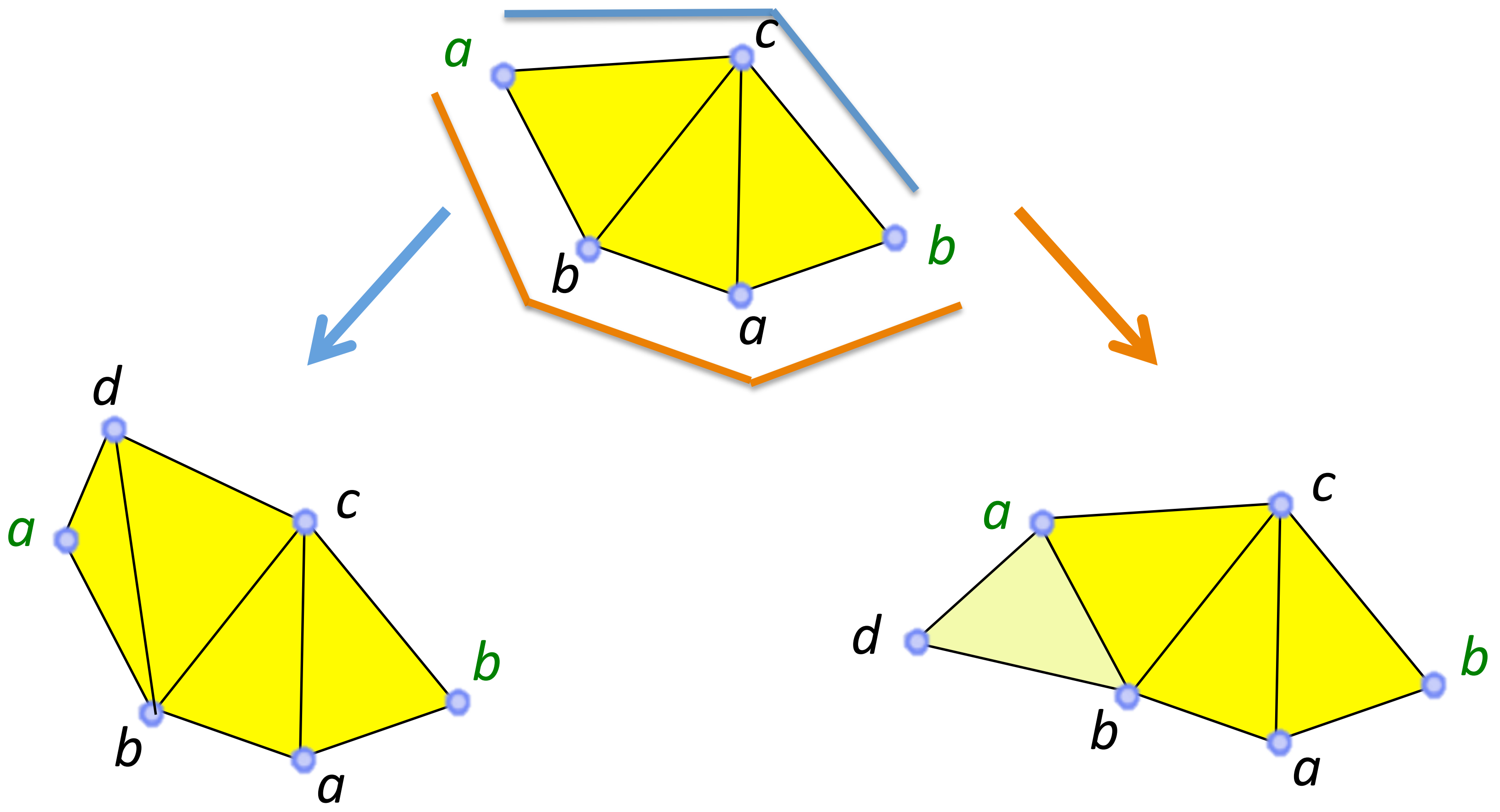
If the enclosing region contains more than four different vertices :



Reduce a path of facets

But, it is not always possible to reduce the path…

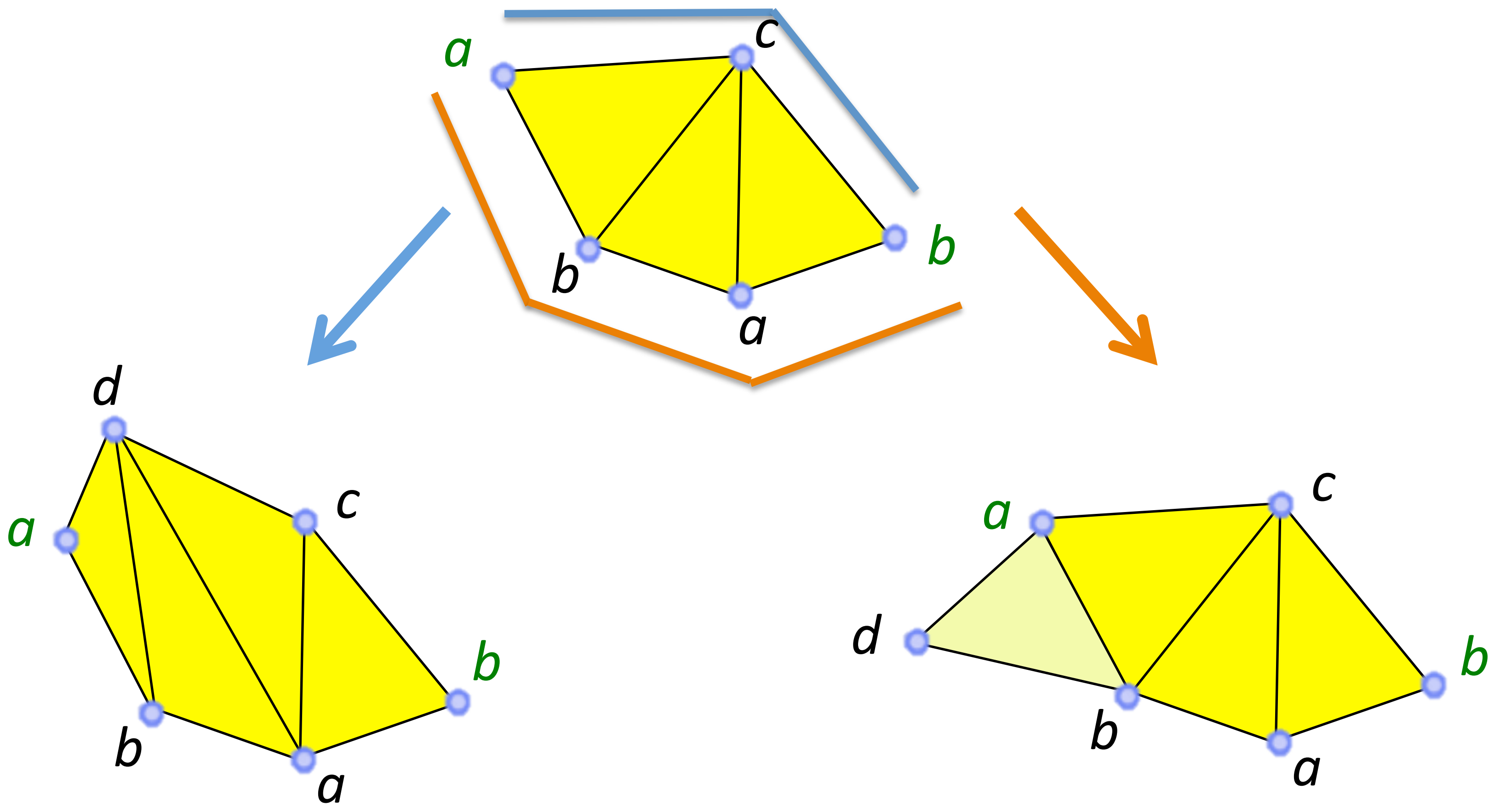
If the enclosing region contains more than four different vertices :



Reduce a path of facets

But, it is not always possible to reduce the path…

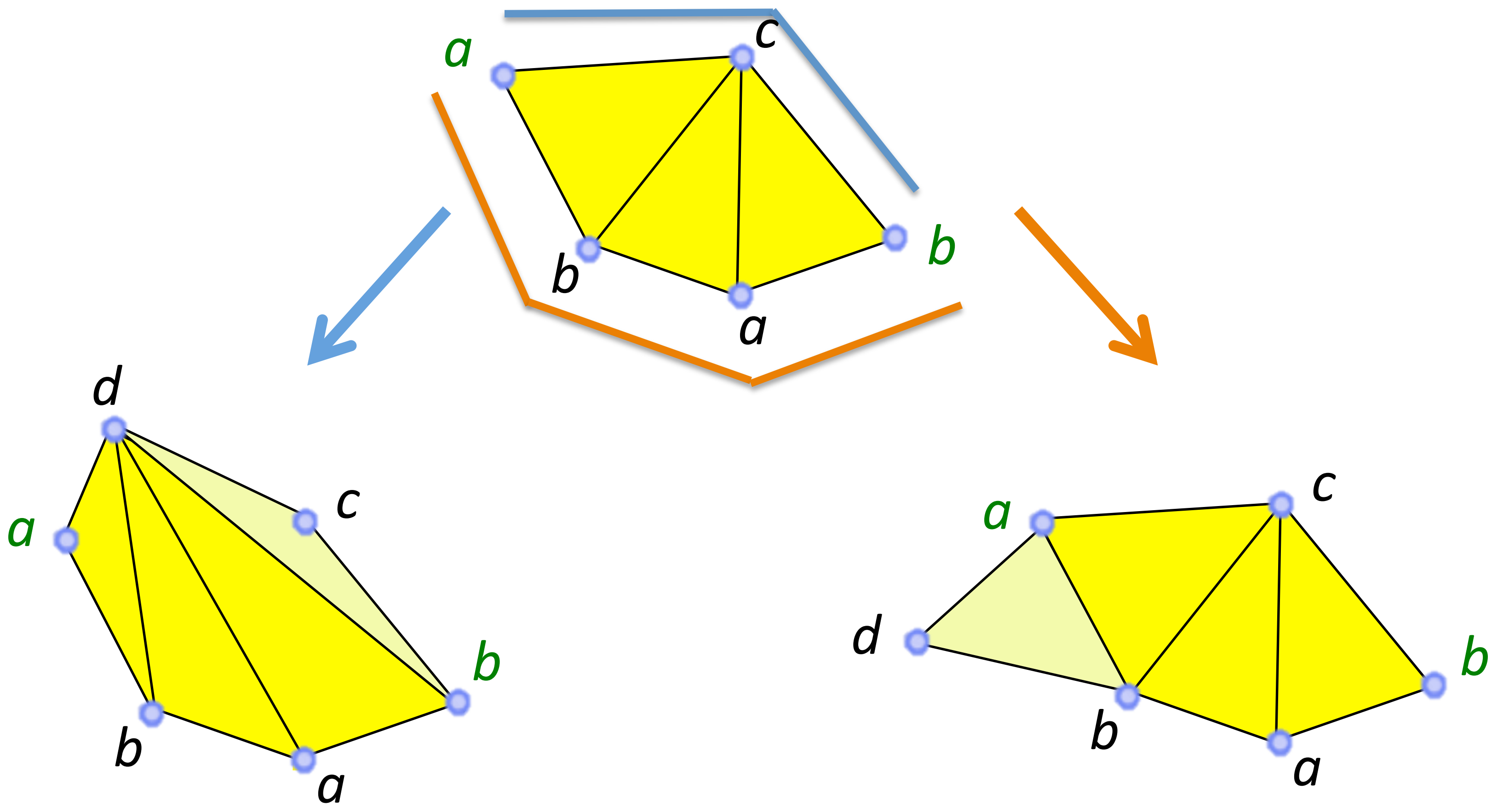
If the enclosing region contains more than four different vertices :



Reduce a path of facets

But, it is not always possible to reduce the path…

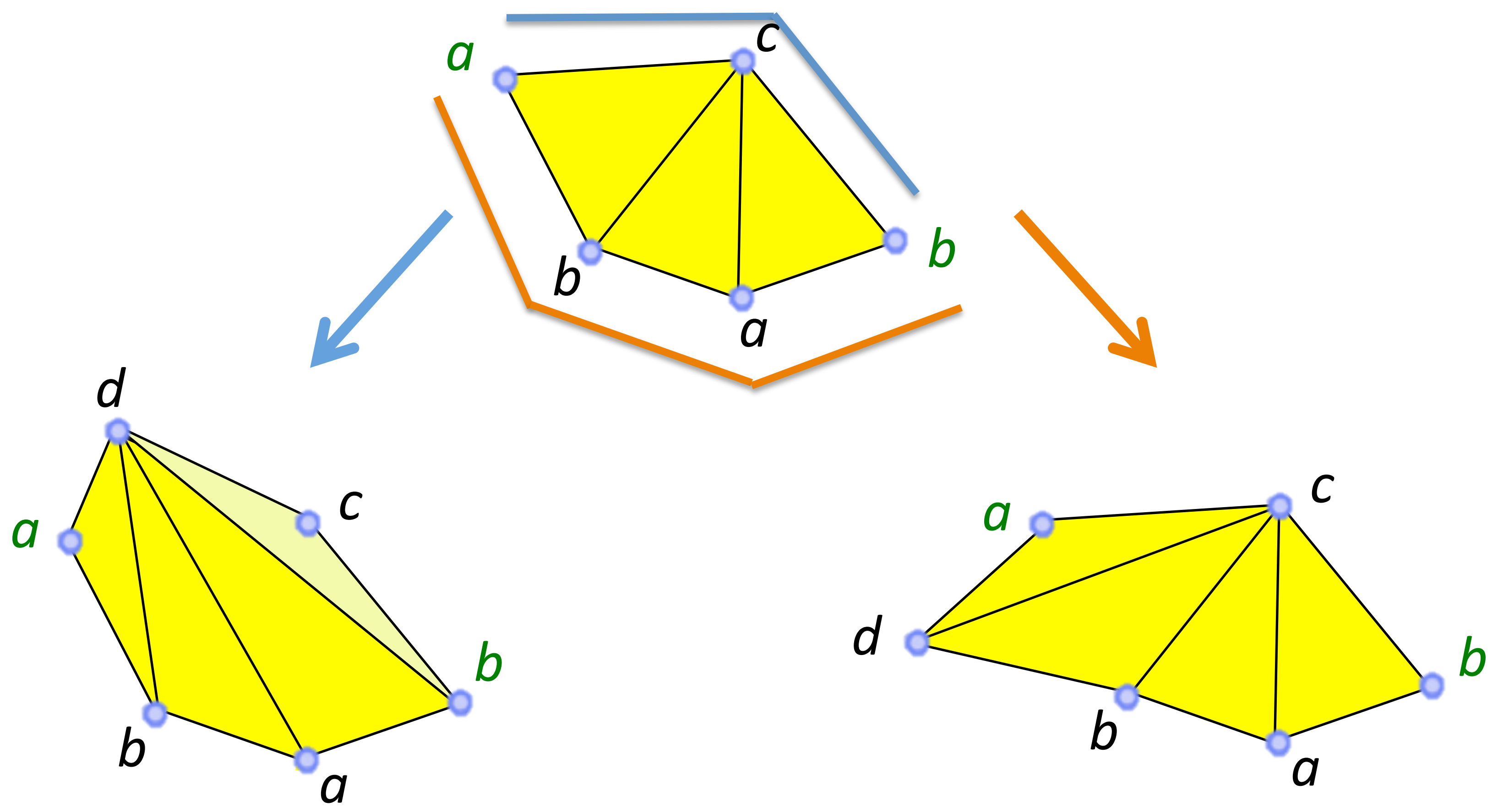
If the enclosing region contains more than four different vertices :



Reduce a path of facets

But, it is not always possible to reduce the path…

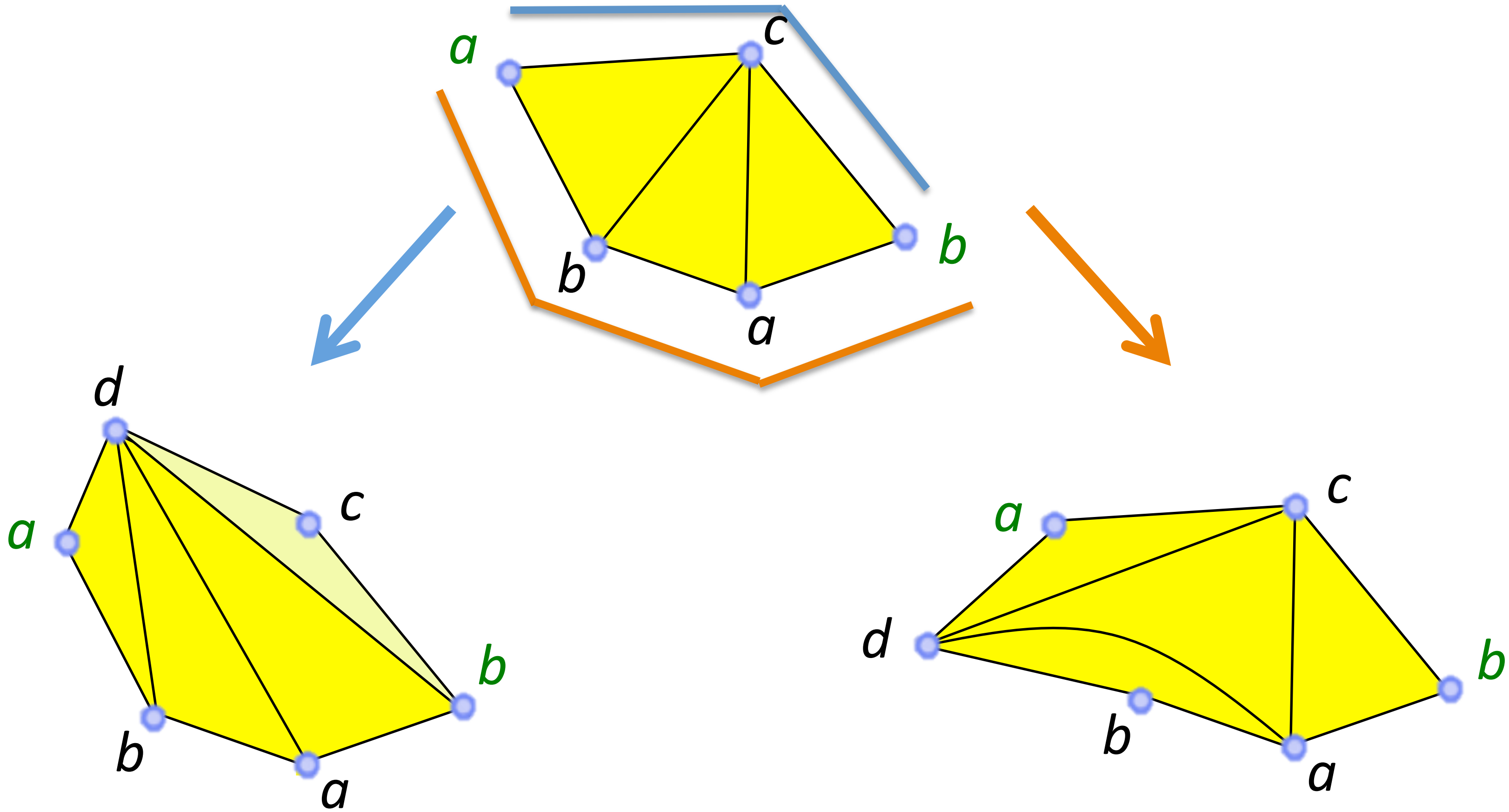
If the enclosing region contains more than four different vertices :



Reduce a path of facets

But, it is not always possible to reduce the path…

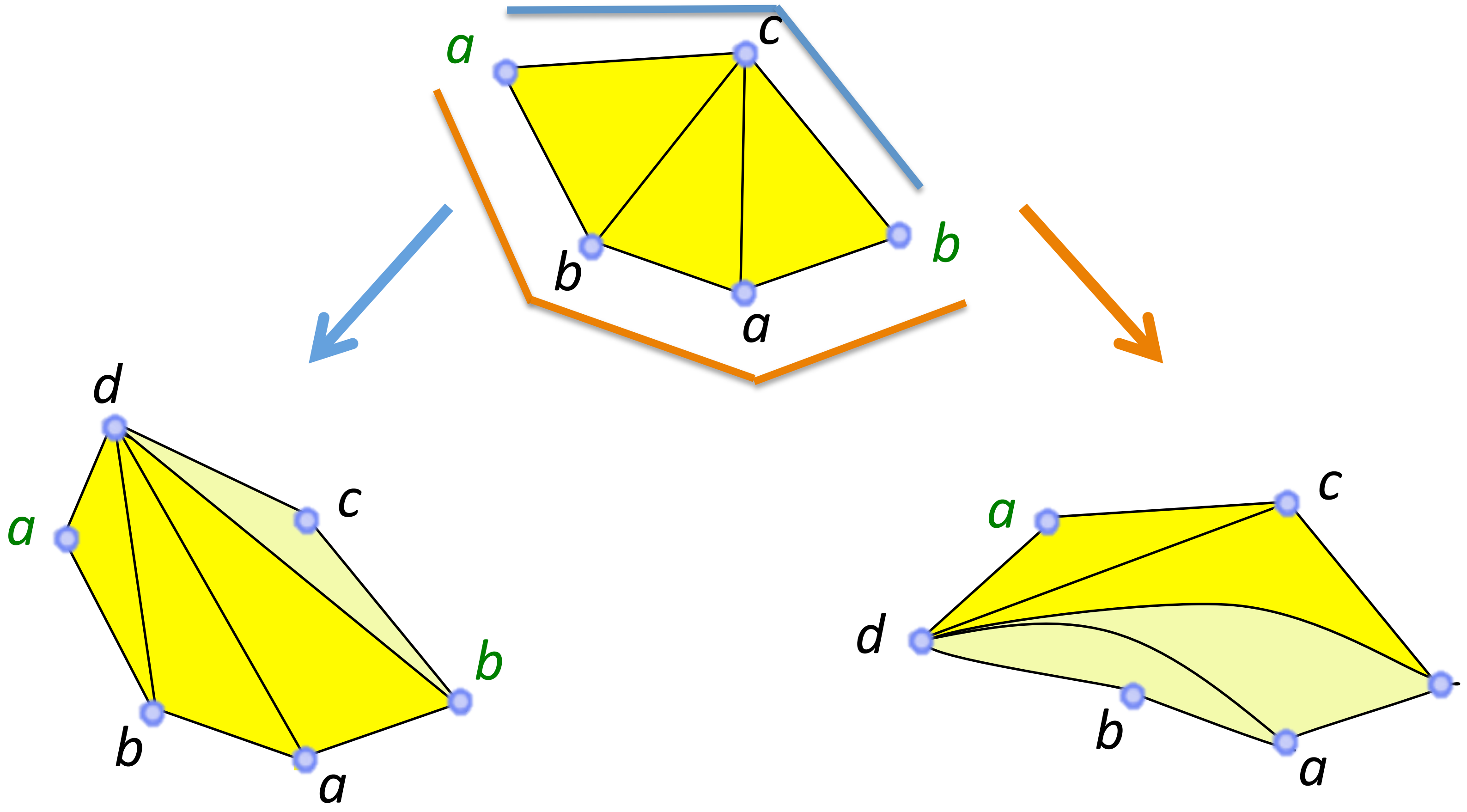
If the enclosing region contains more than four different vertices :



Reduce a path of facets

But, it is not always possible to reduce the path…

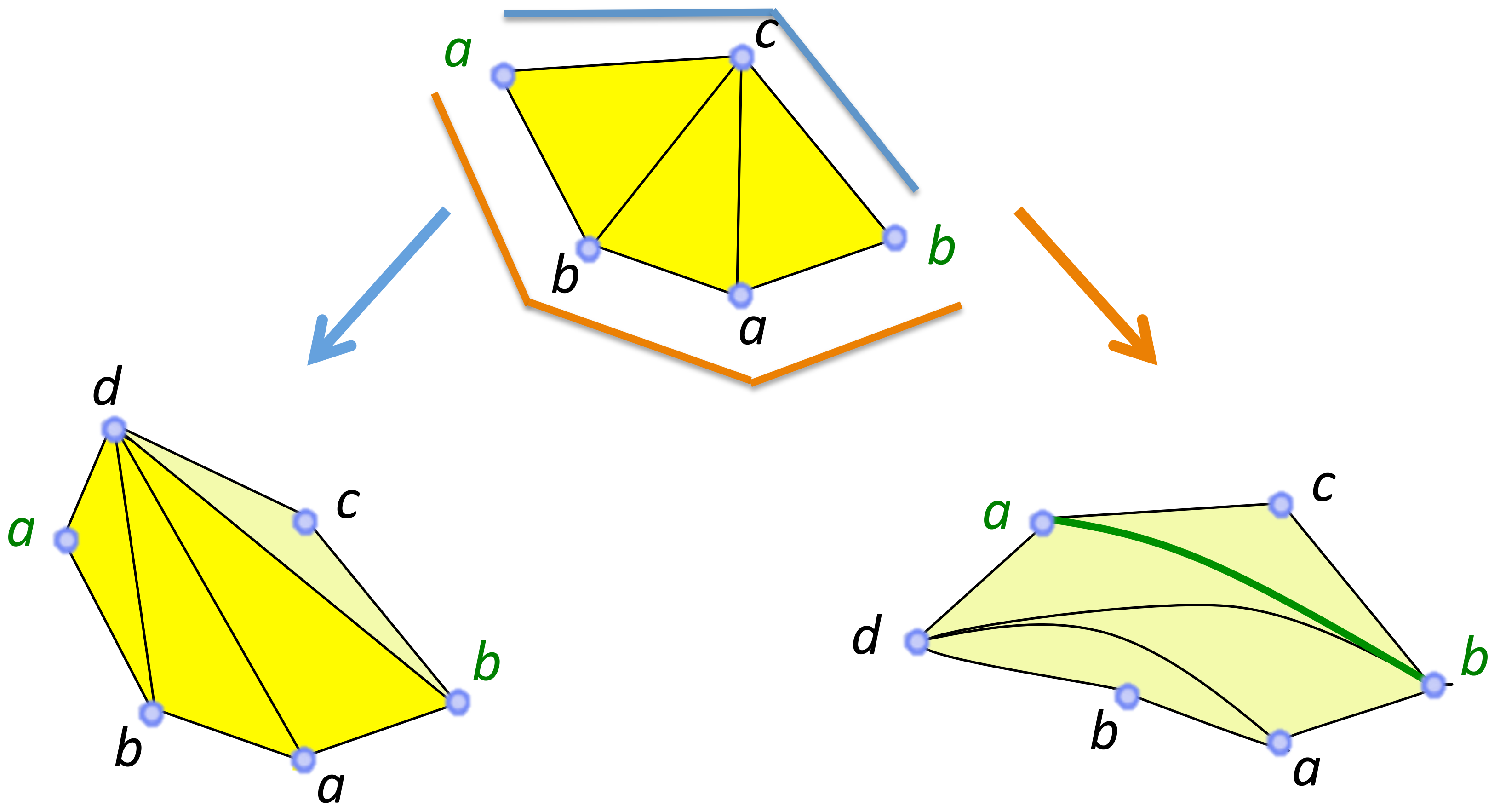
If the enclosing region contains more than four different vertices :



Reduce a path of facets

But, it is not always possible to reduce the path…

If the enclosing region contains more than four different vertices :



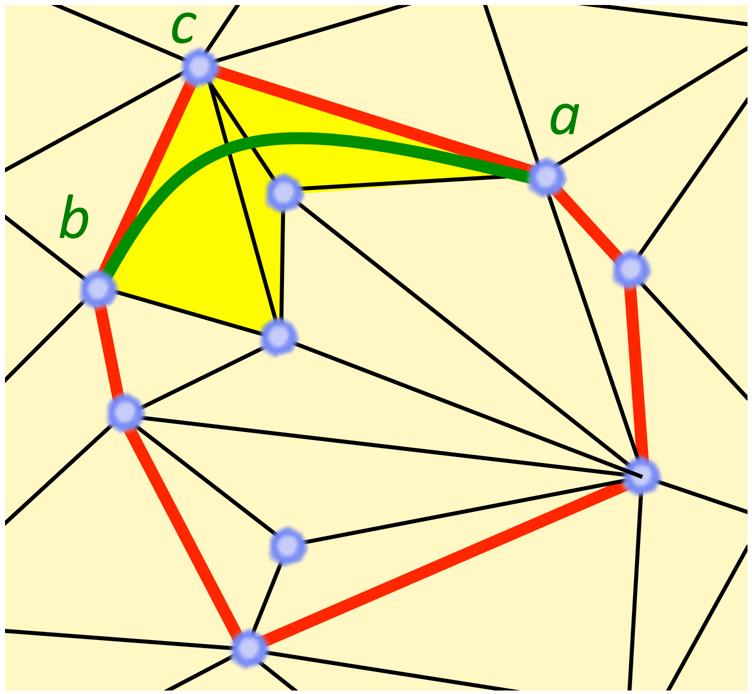
Reduce a path of facets

If the enclosing region contains more than four different vertices , we can construct these edges :

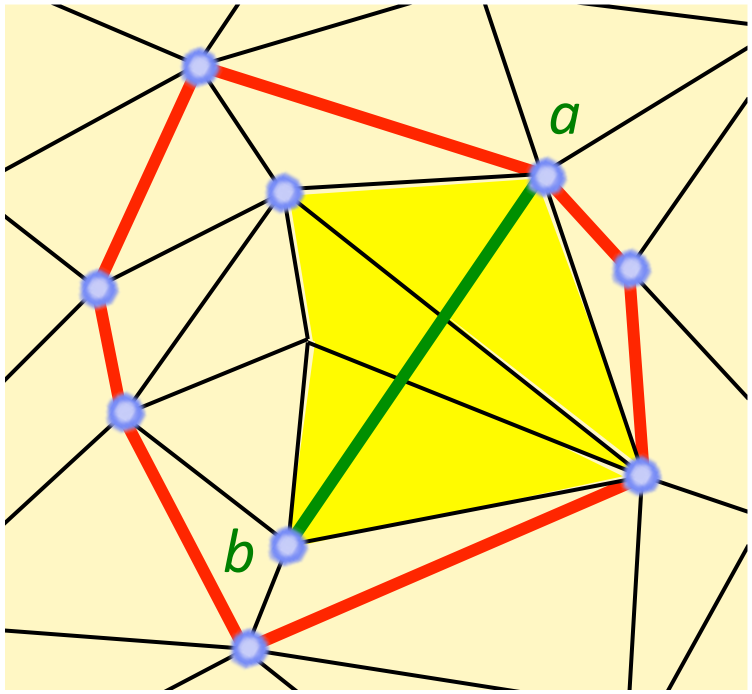
Reduce a path of facets

If the enclosing region contains more than four different vertices , we can construct these edges :

If the edge will create a face :



If the edge will not split the region :



Outline

I. Construct one edge in a triangulation that may contain constrained unflippable edges

II. Construct the edges of T_{target} on the evolving mesh using a strategy that converges towards the connectivity of T_{target}

Construction of all the target edges

Whenever an edge is constructed, it is constrained in the two meshes, meaning it is unusable in any path involved in a future construction!

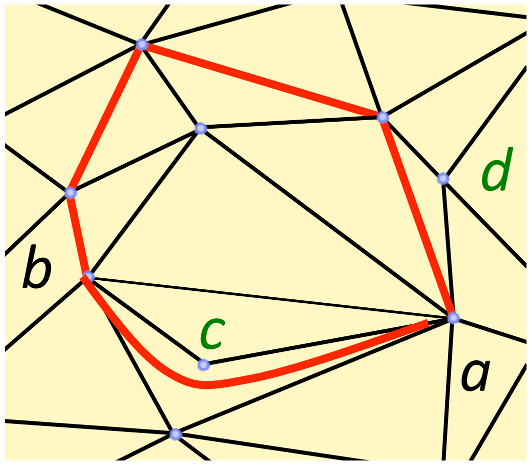
Any constructed edge must meet the following criterion:

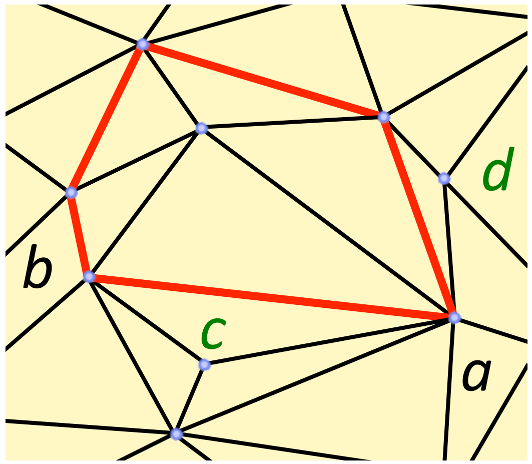
Construction of all the target edges

Whenever an edge is constructed, it is constrained in the two meshes, meaning it is unusable in any path involved in a future construction!

Any constructed edge must meet the following criterion:

- In the evolving triangulation, there always exists a triangle path between two vertices that are adjacent in T_{target}



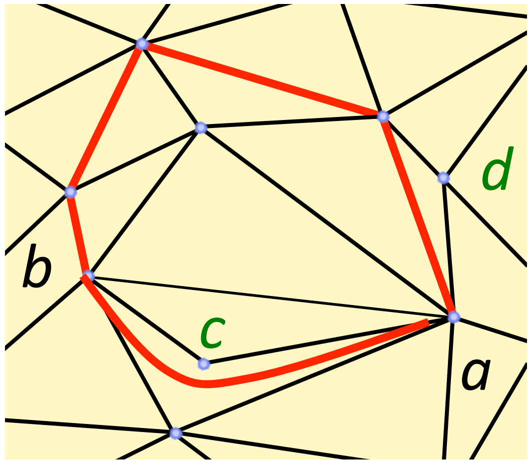


Construction of all the target edges

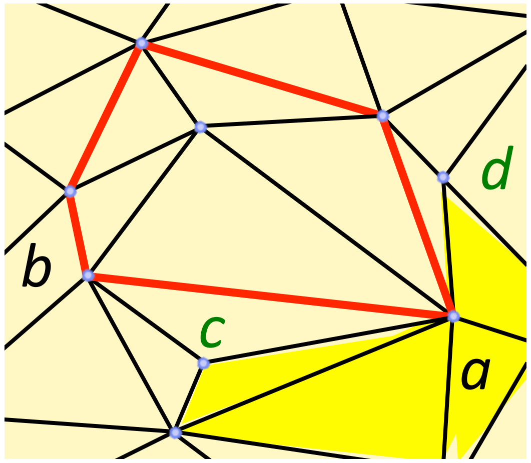
Whenever an edge is constructed, it is constrained in the two meshes, meaning it is unusable in any path involved in a future construction!

Any constructed edge must meet the following criterion:

- In the evolving triangulation, there always exists a triangle path between two vertices that are adjacent in T_{target}



(cd) is not constructible



(cd) is constructible

Construction of all edges

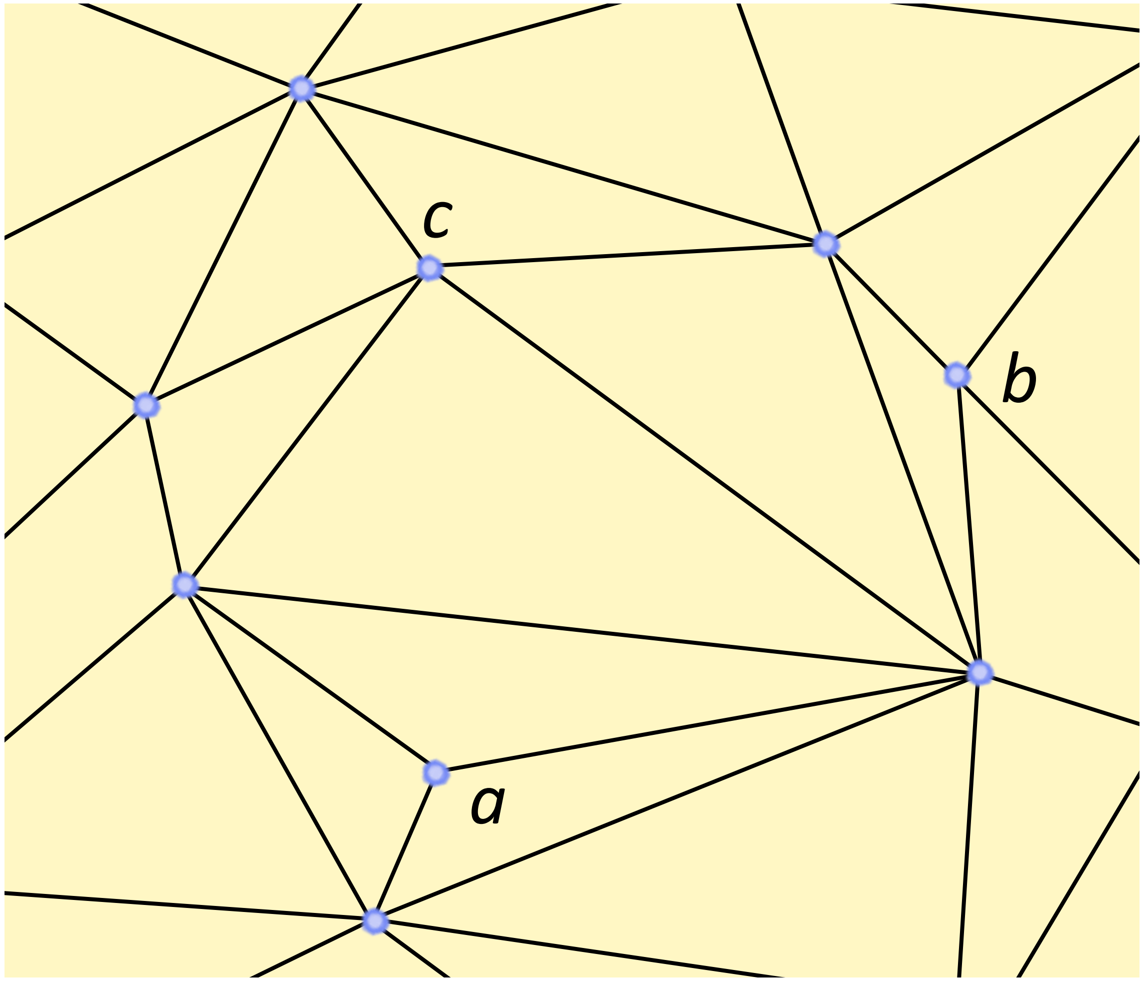
Our solution : construct all the facets of T_{target} by <u>region growing</u>!

Constructing a facet *(abc)* consists in constructing the edges *(ab)*, *(bc)* and *(ca)* such that the enclosing region *(abc)* contains no vertices.

1) The algorithm is initiated by the construction of one facet in T_{Target}

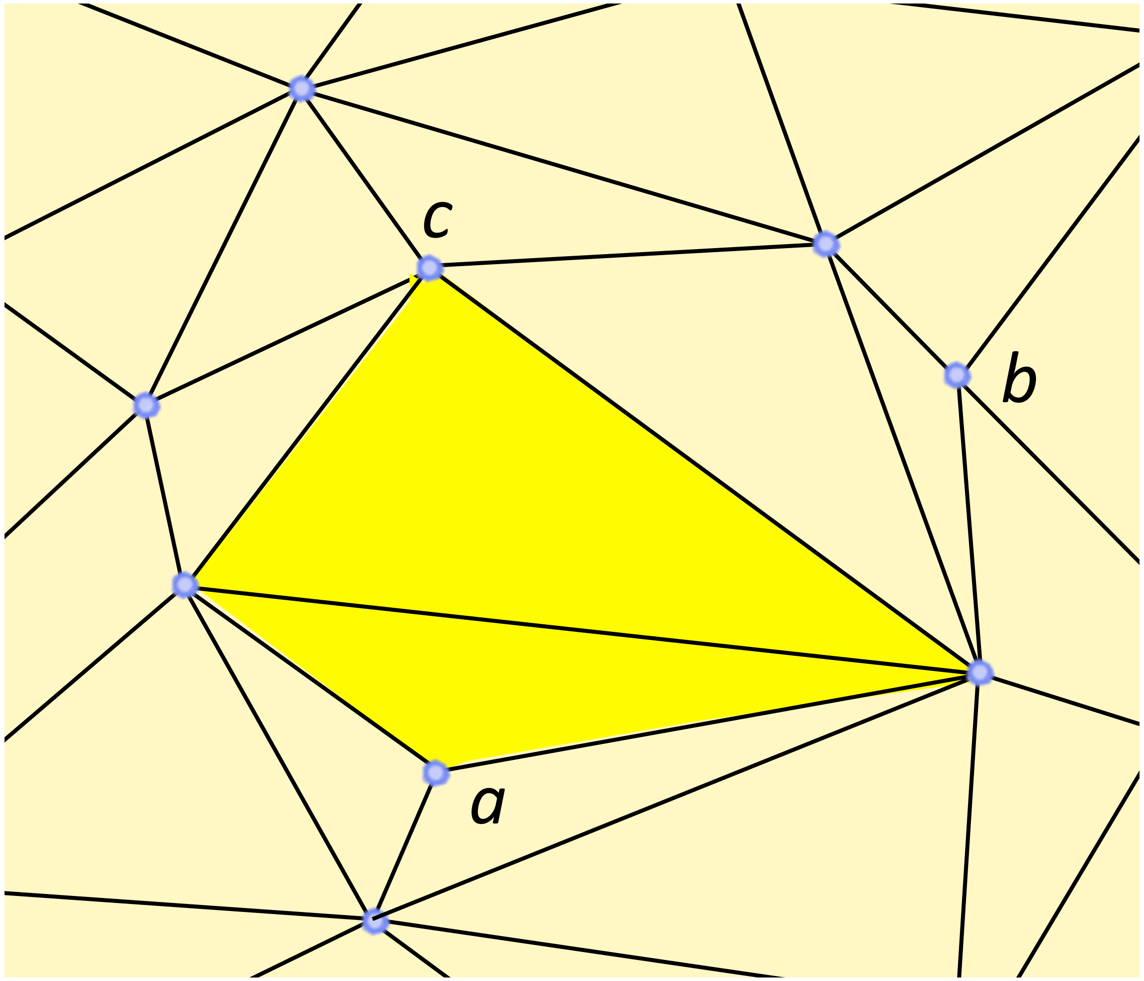
2) The current triangulation evolves by incrementally constructing a facet of T_{Target} adjacent to a facet already blocked.

Region Growing Strategy



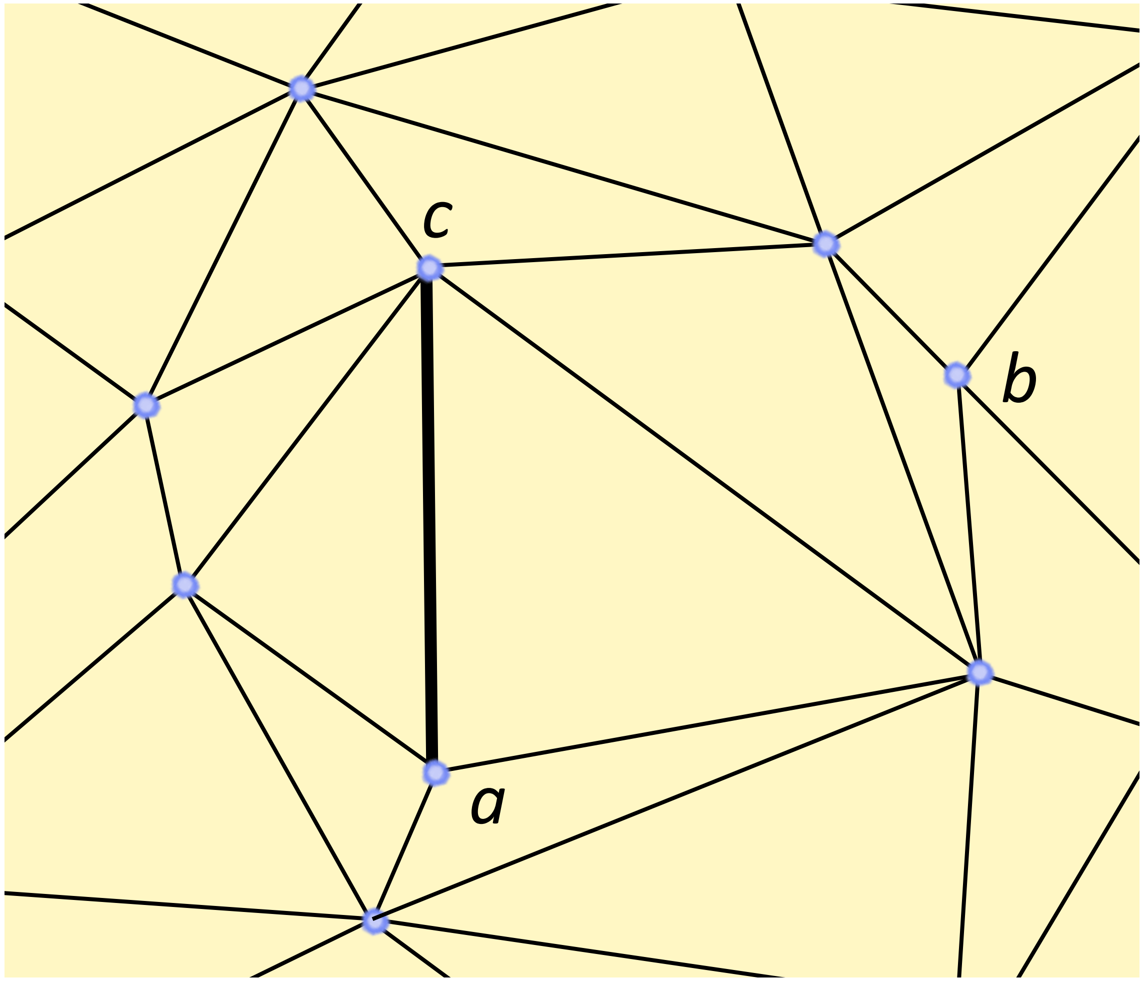
Constructing the first well-oriented facet *(abc)*

Region Growing Strategy



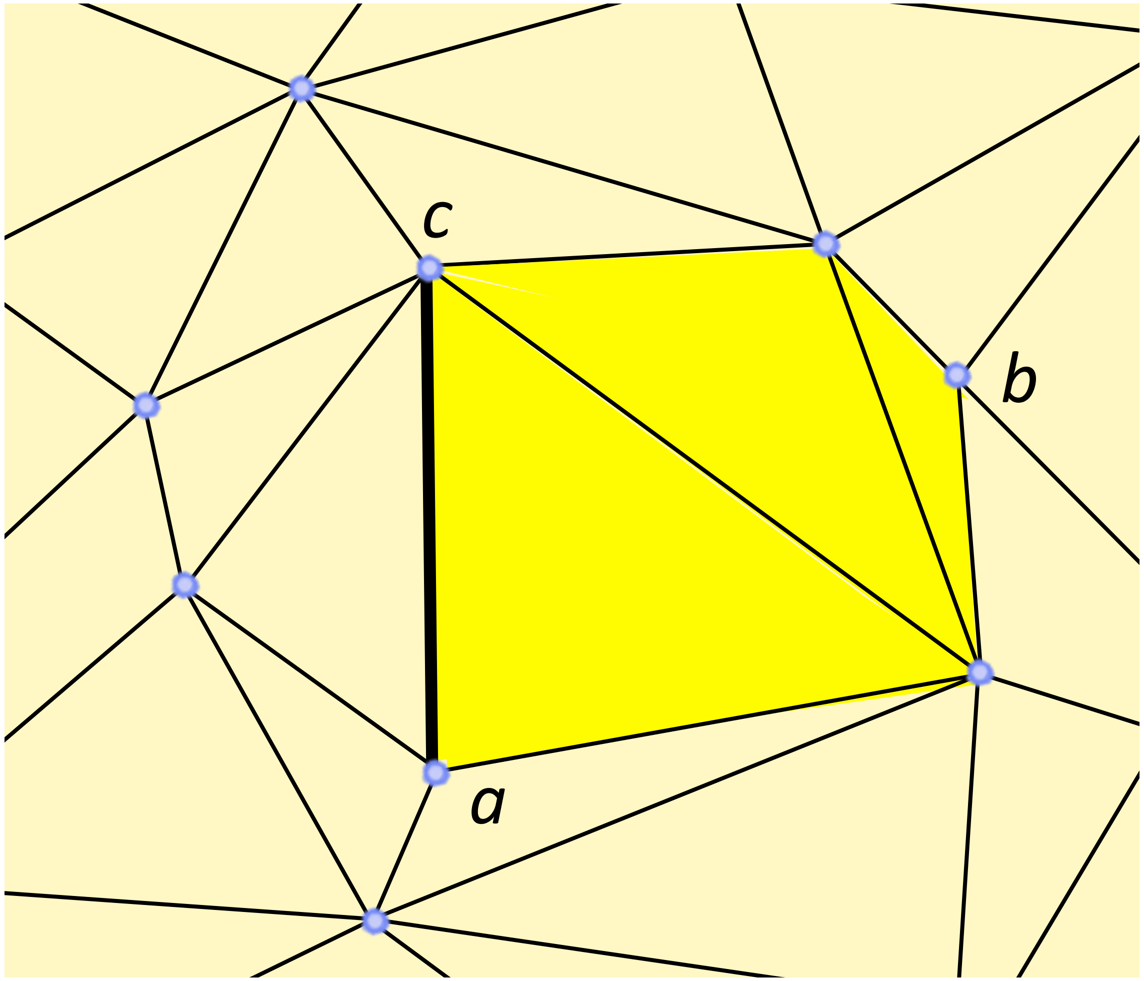
Constructing the first well-oriented facet *(abc)*

Region Growing Strategy



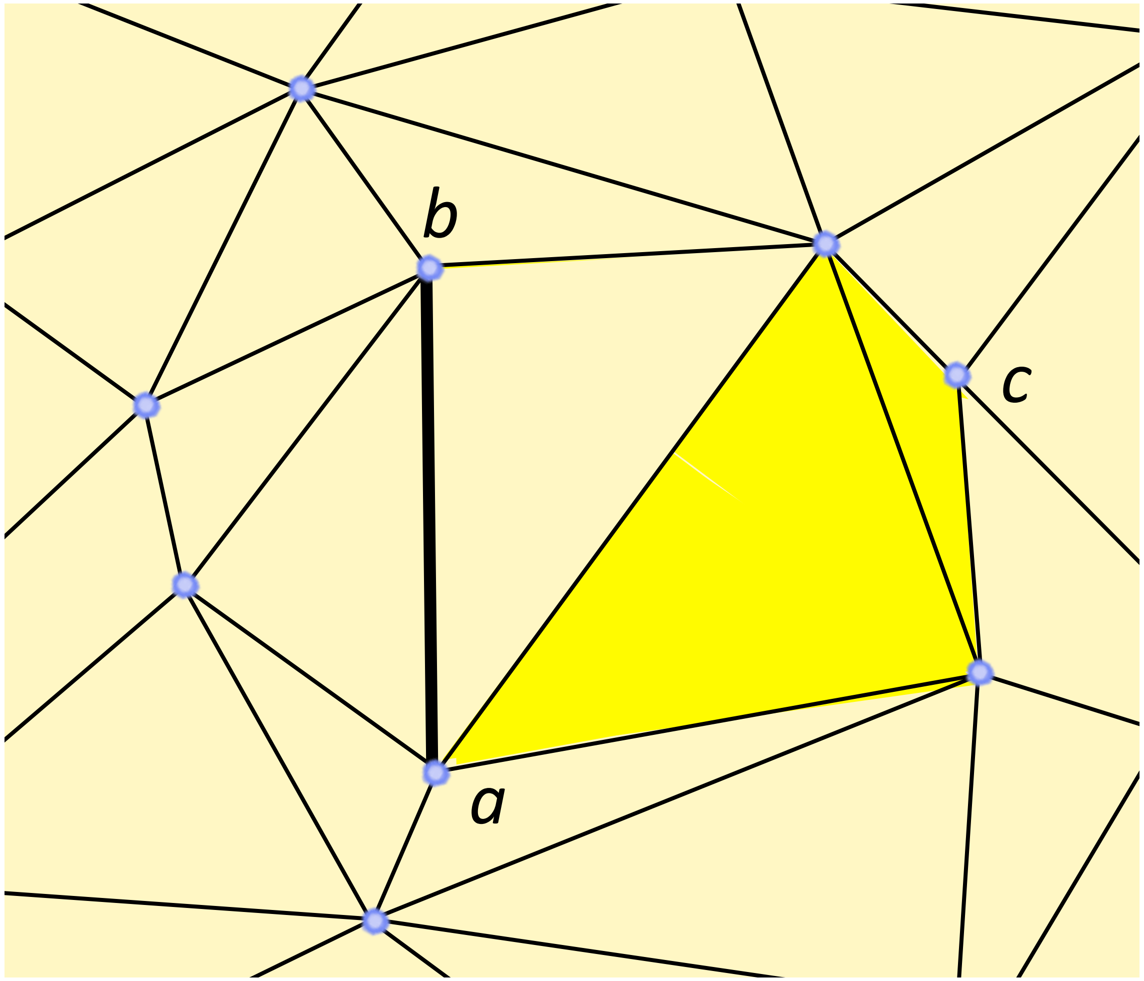
Constructing the first well-oriented facet *(abc)*

Region Growing Strategy



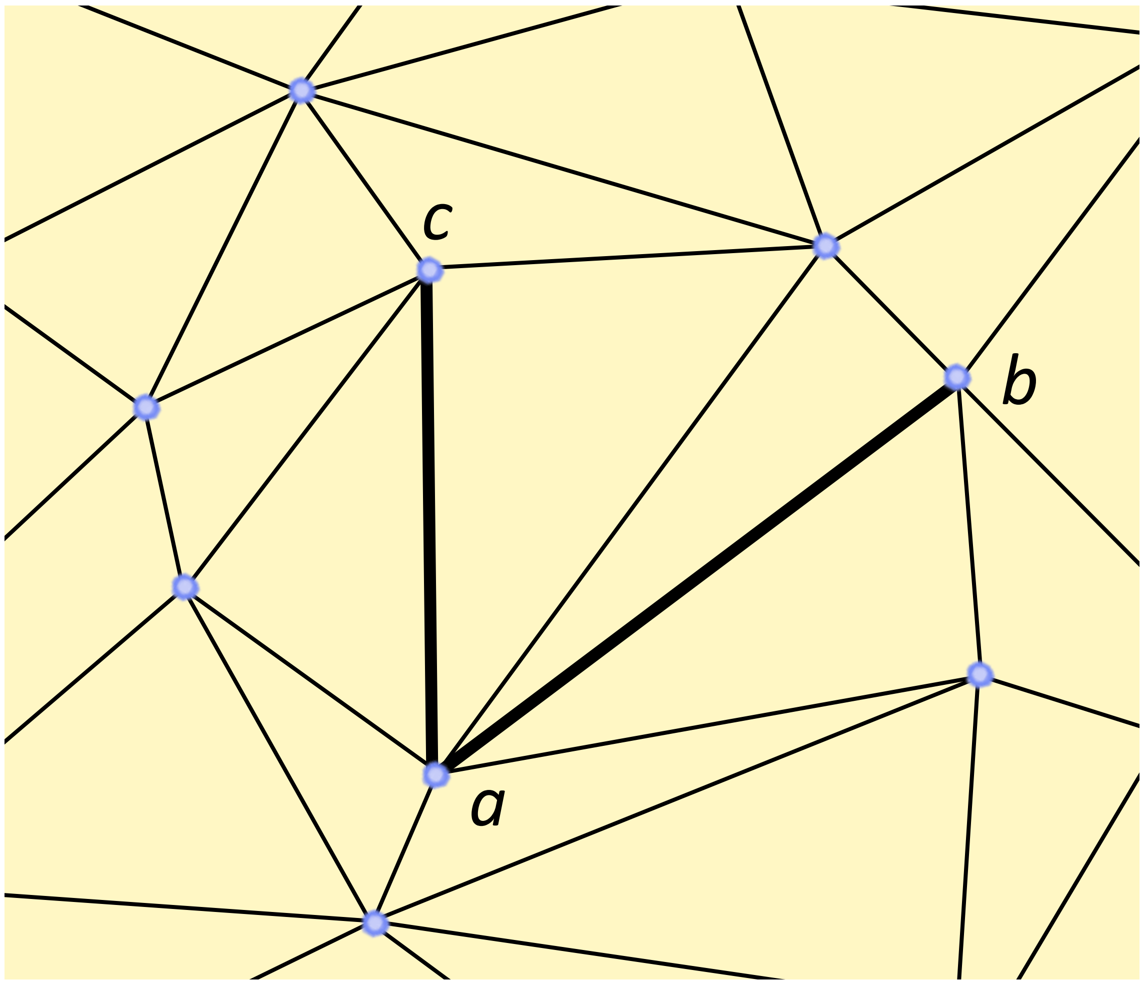
Constructing the first well-oriented facet *(abc)*

Region Growing Strategy



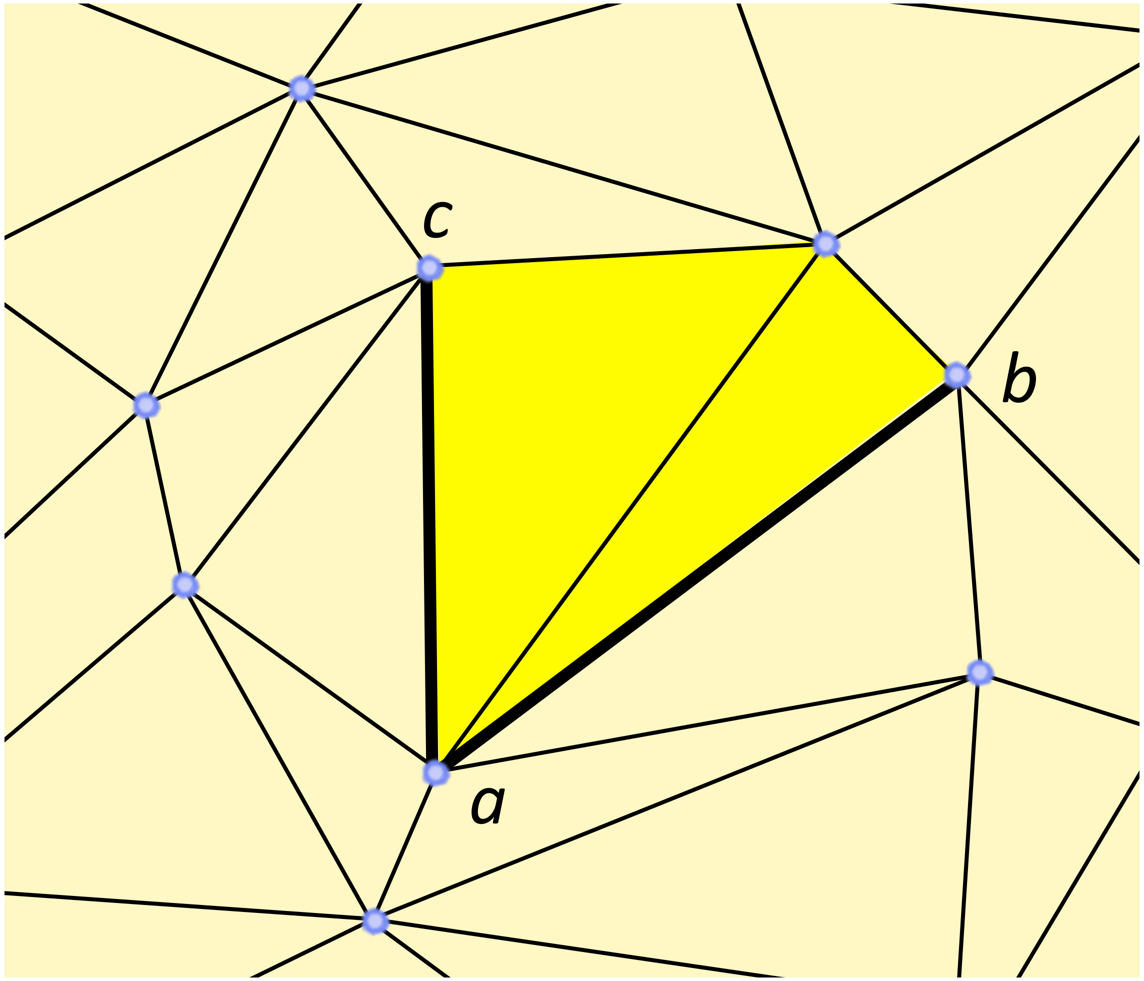
Constructing the first well-oriented facet (abc)

Region Growing Strategy



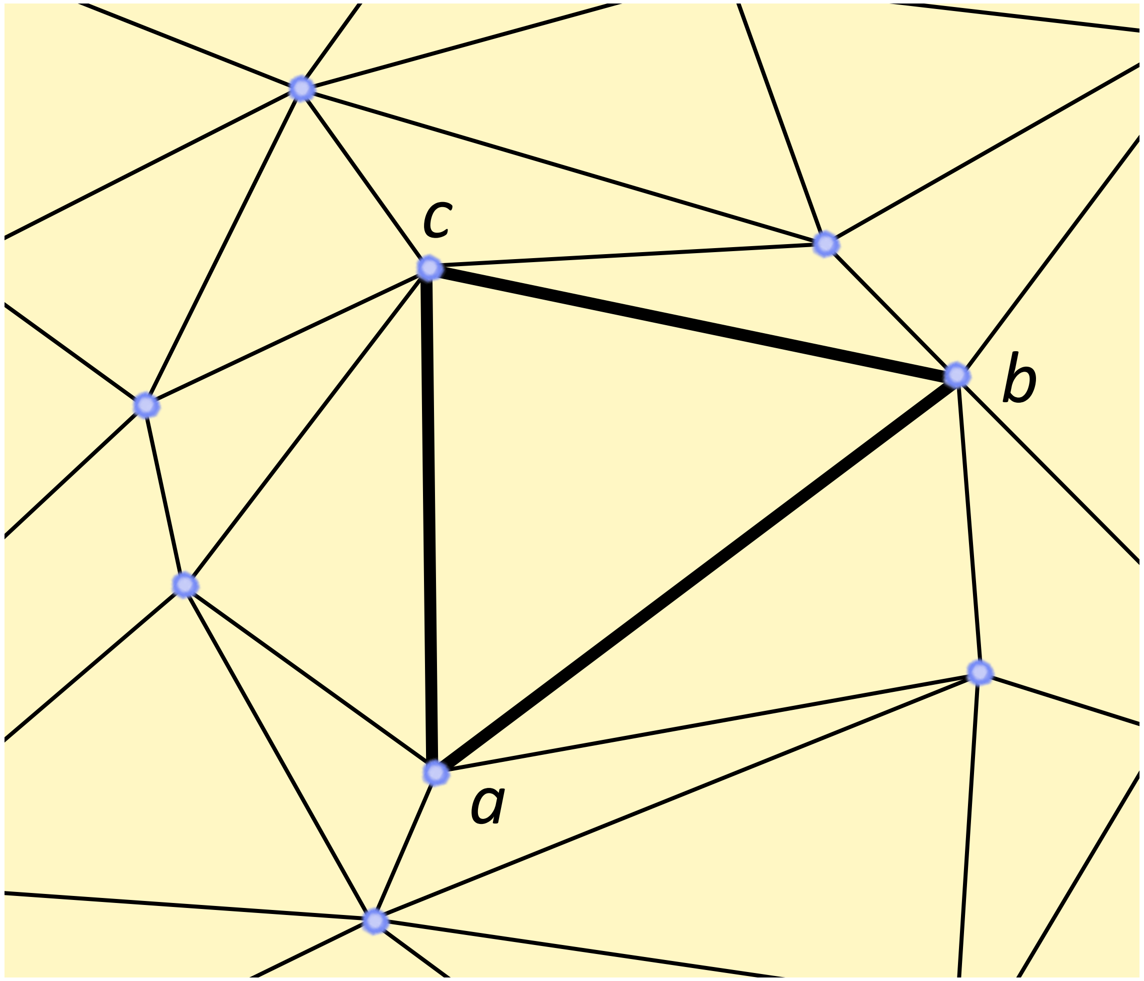
Constructing the first well-oriented facet *(abc)*

Region Growing Strategy



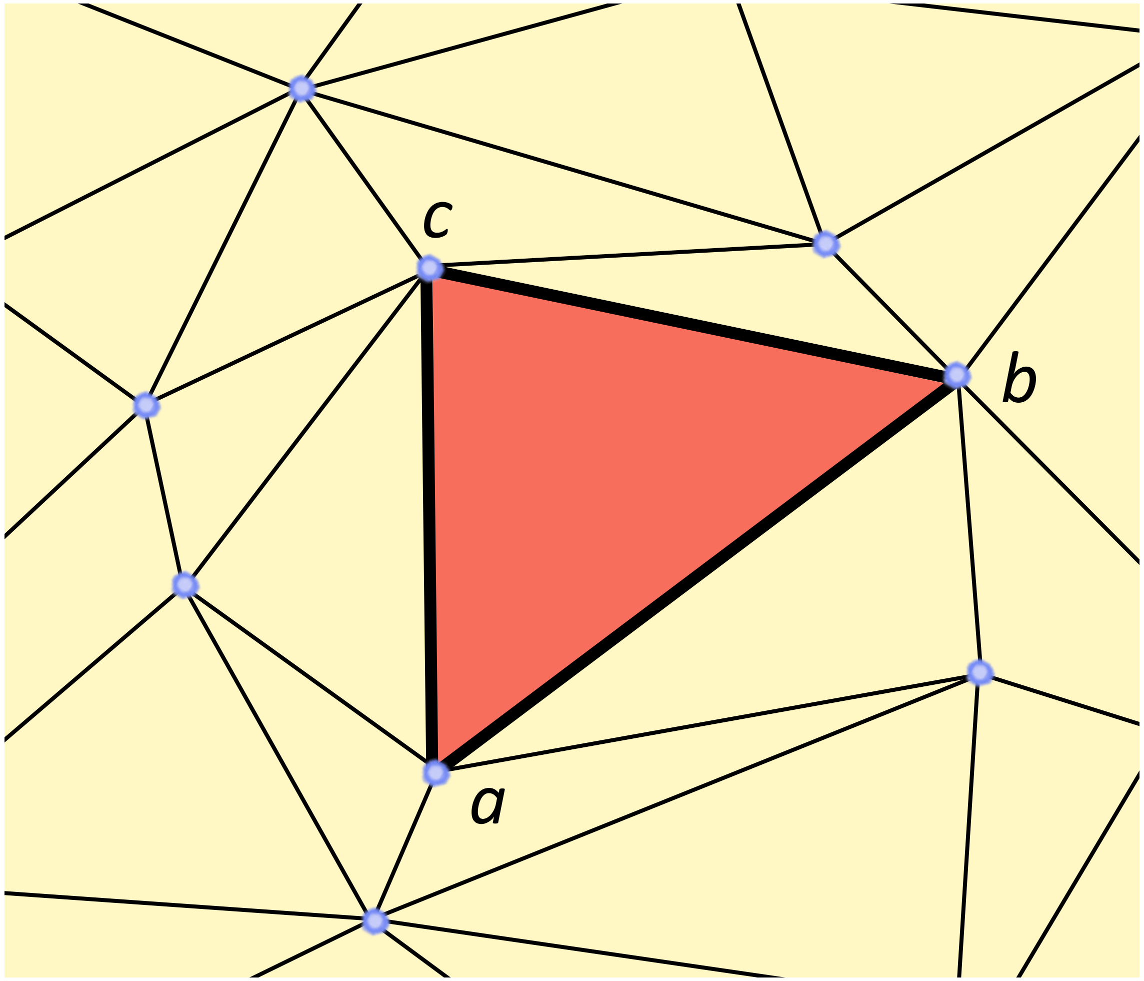
Constructing the first well-oriented facet *(abc)*

Region Growing Strategy



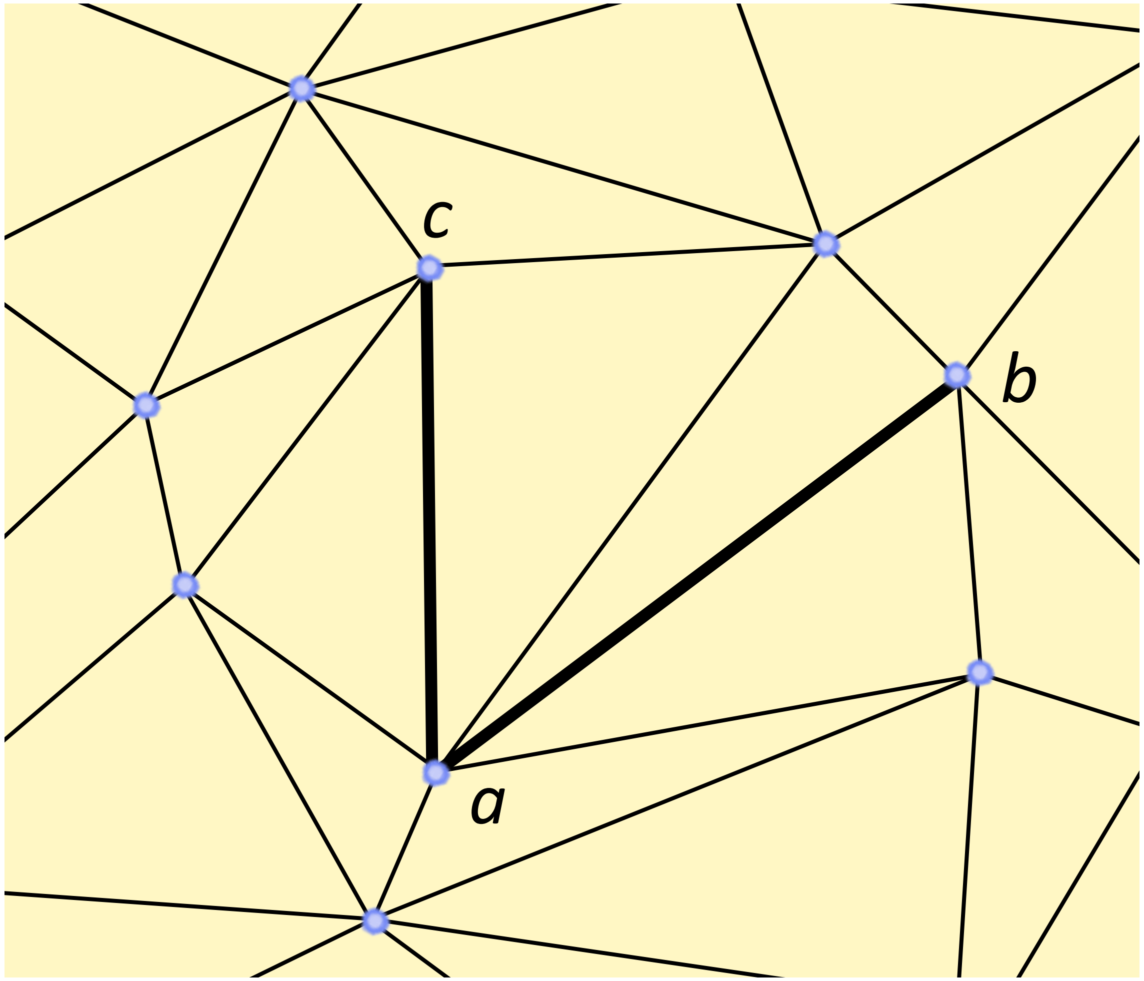
Constructing the first well-oriented facet *(abc)*

Region Growing Strategy



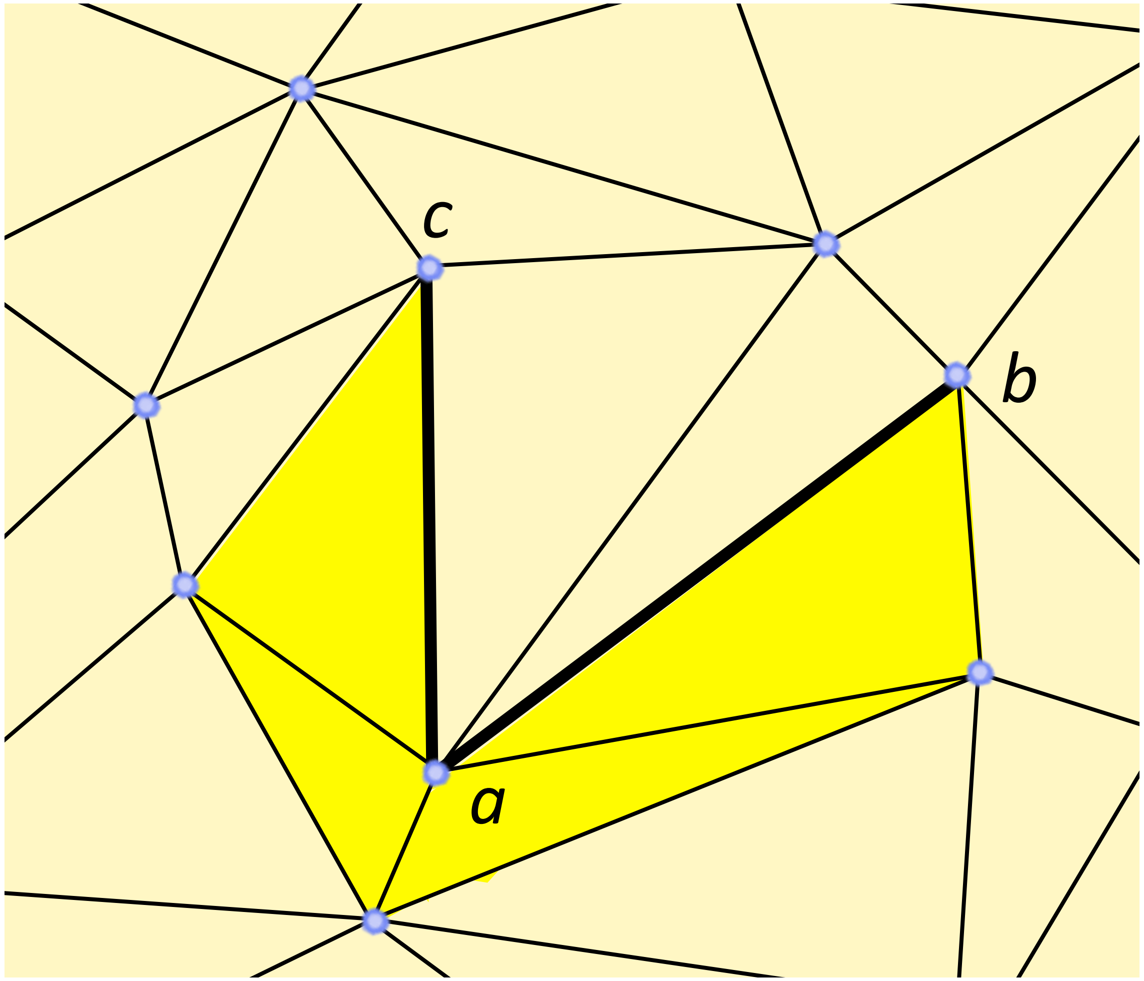
Constructing the first well-oriented facet *(abc)*

Region Growing Strategy



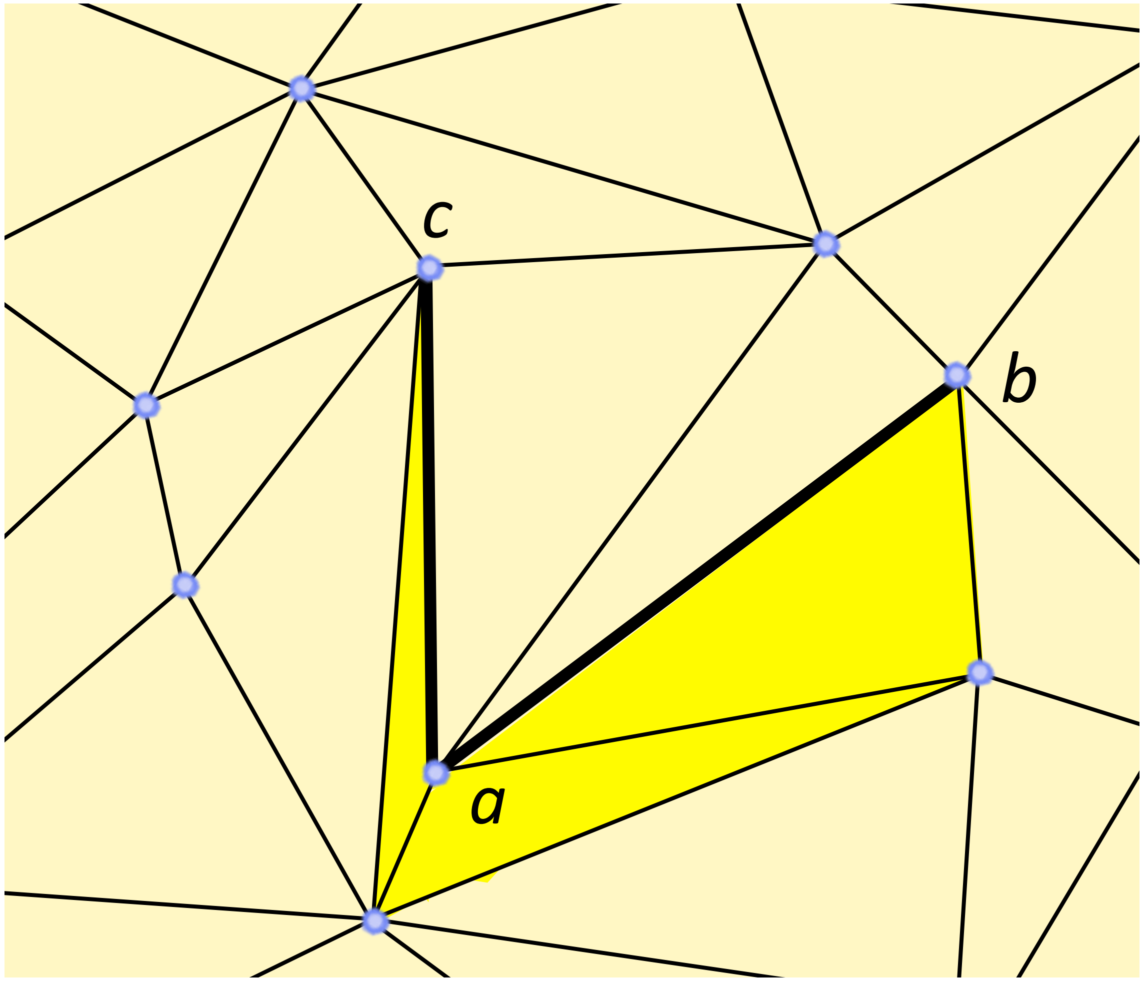
Constructing the first well-oriented facet *(acb)*

Region Growing Strategy



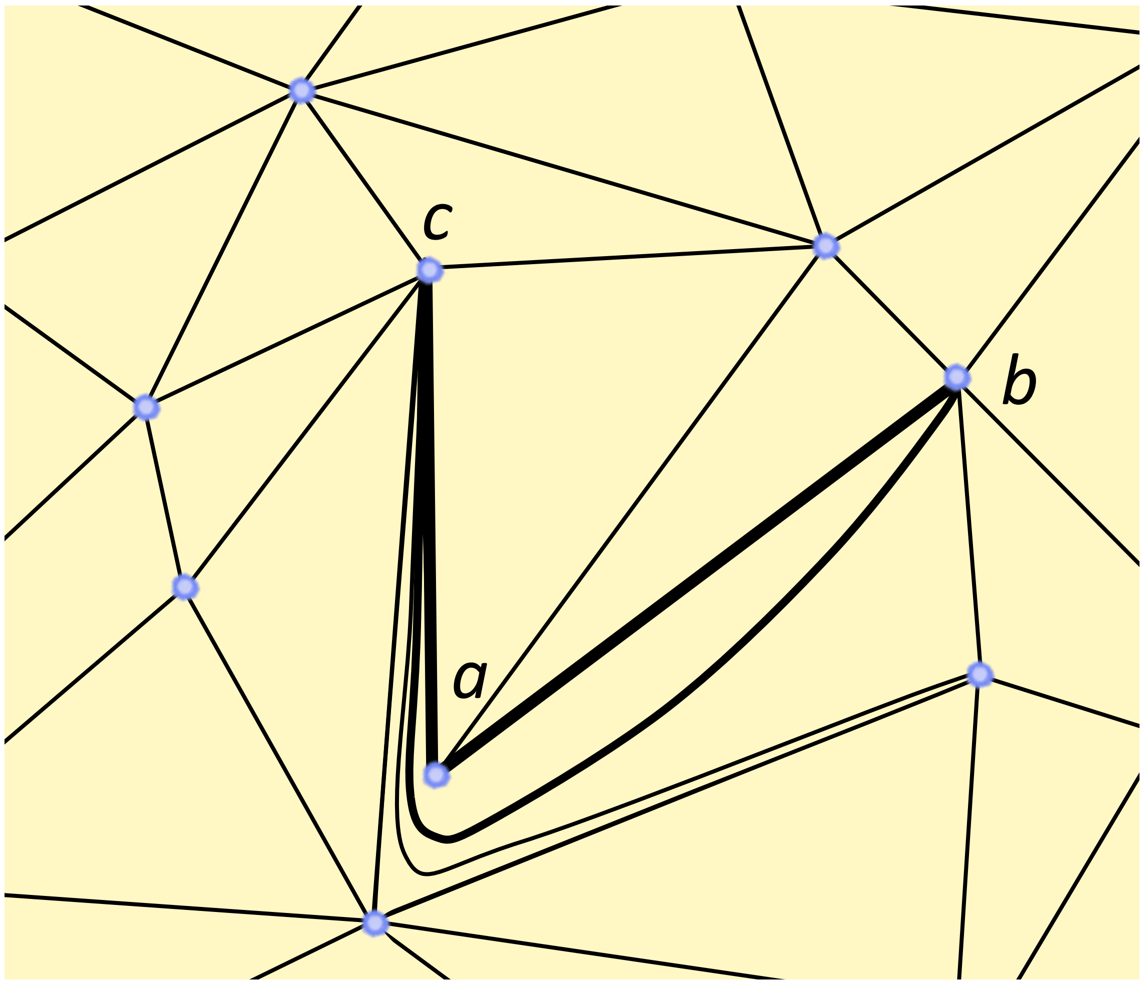
Constructing the first well-oriented facet *(acb)*

Region Growing Strategy



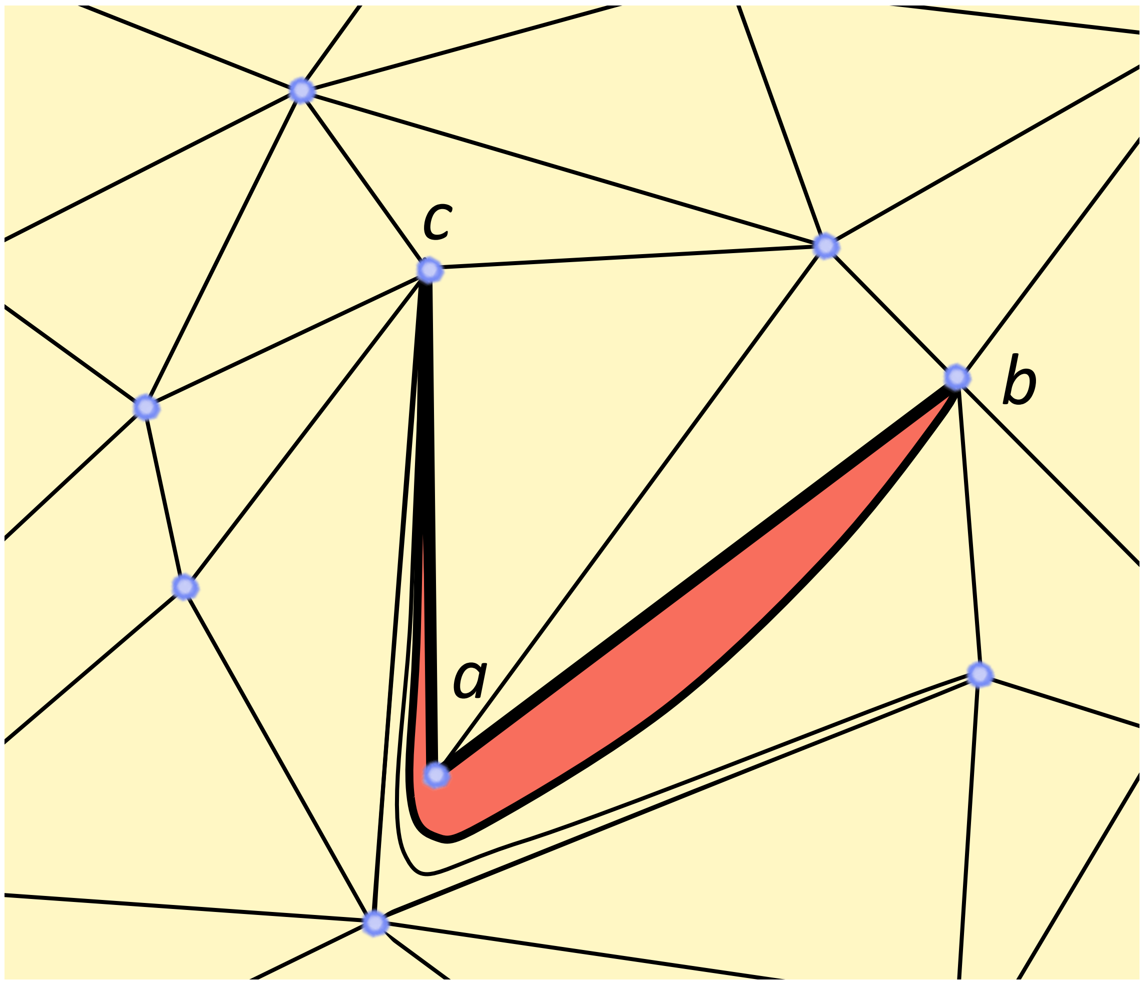
Constructing the first well-oriented facet *(acb)*

Region Growing Strategy



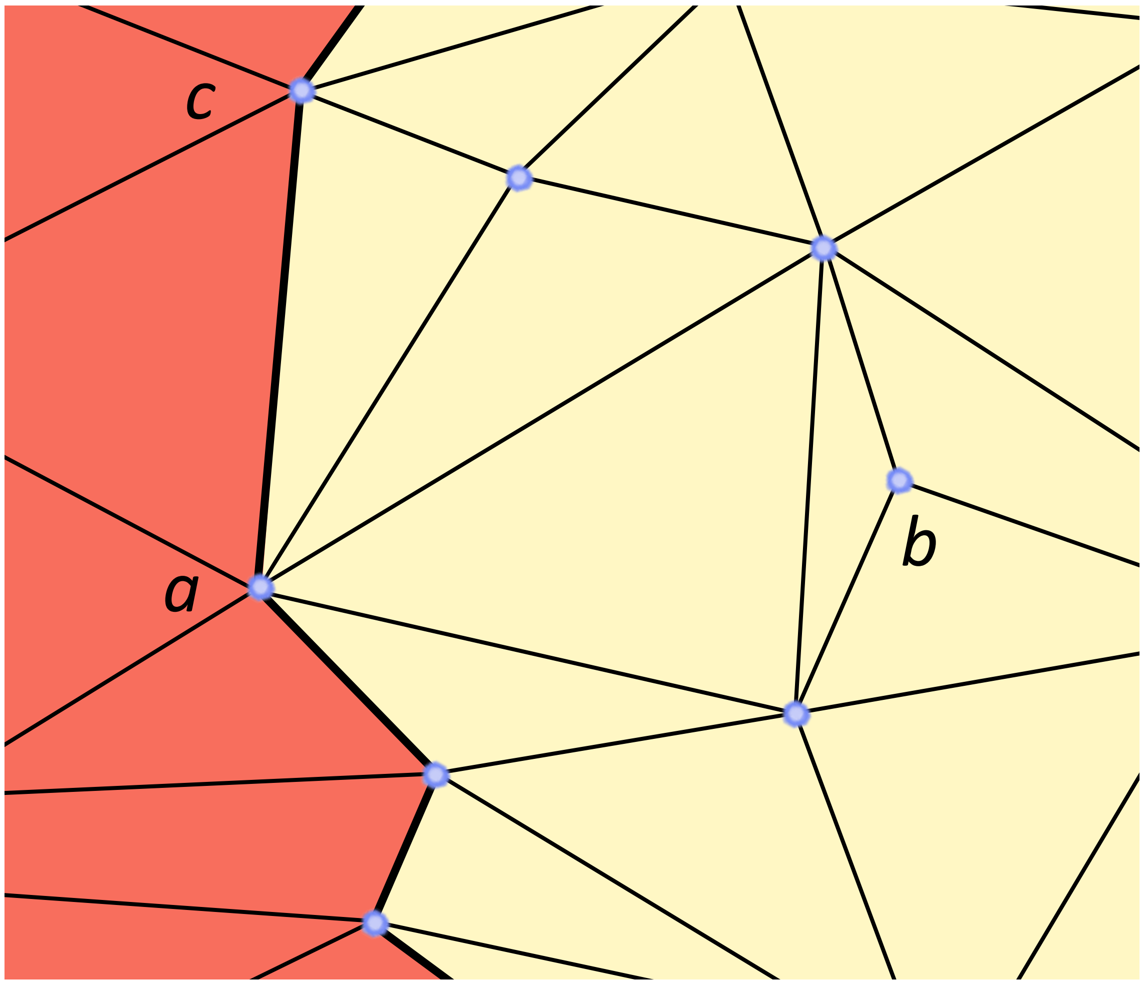
Constructing the first well-oriented facet *(acb)*

Region Growing Strategy



Constructing the first well-oriented facet *(acb)*

Region Growing Strategy

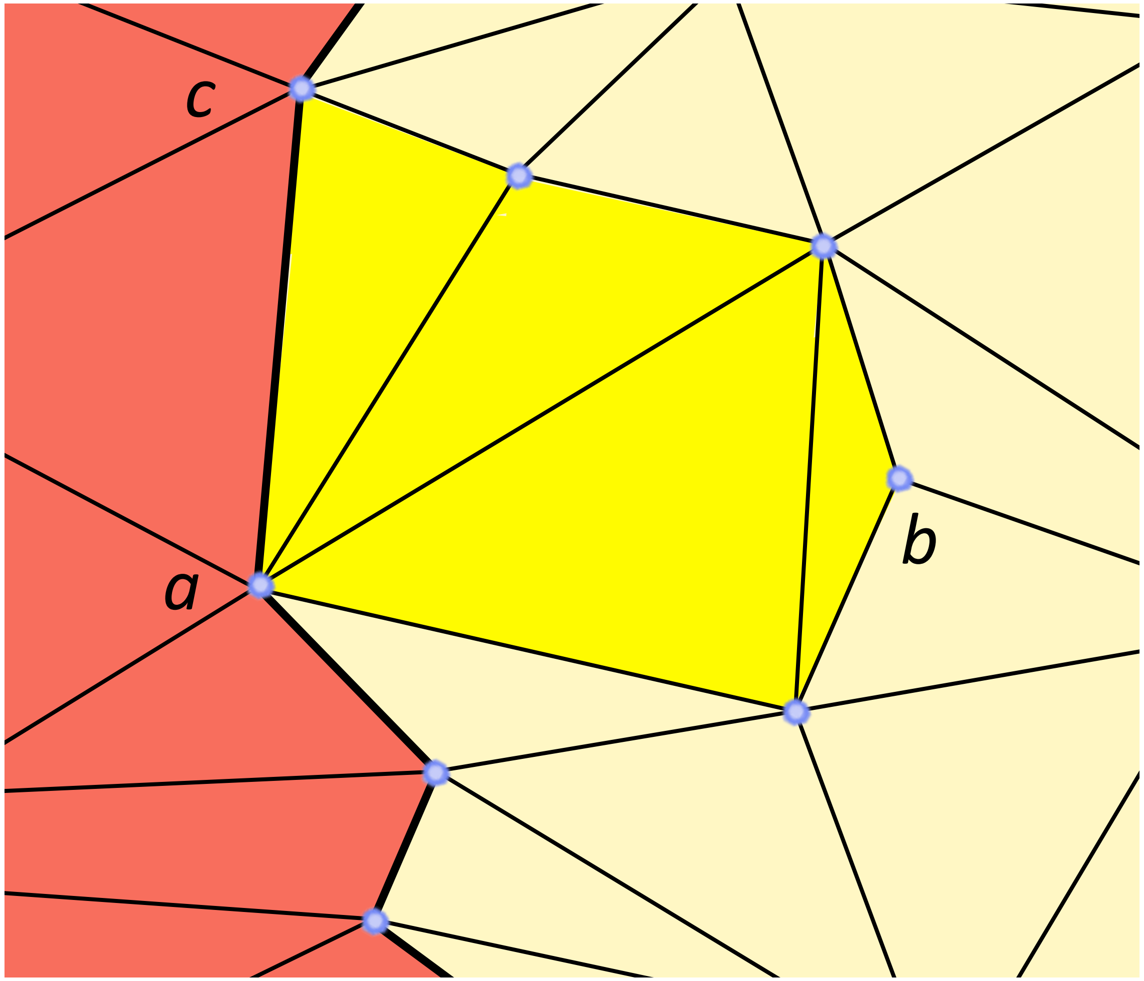


<u>First case of expansion</u> :

Creation of a facet *(abc)* towards a new vertex

(1)

Region Growing Strategy

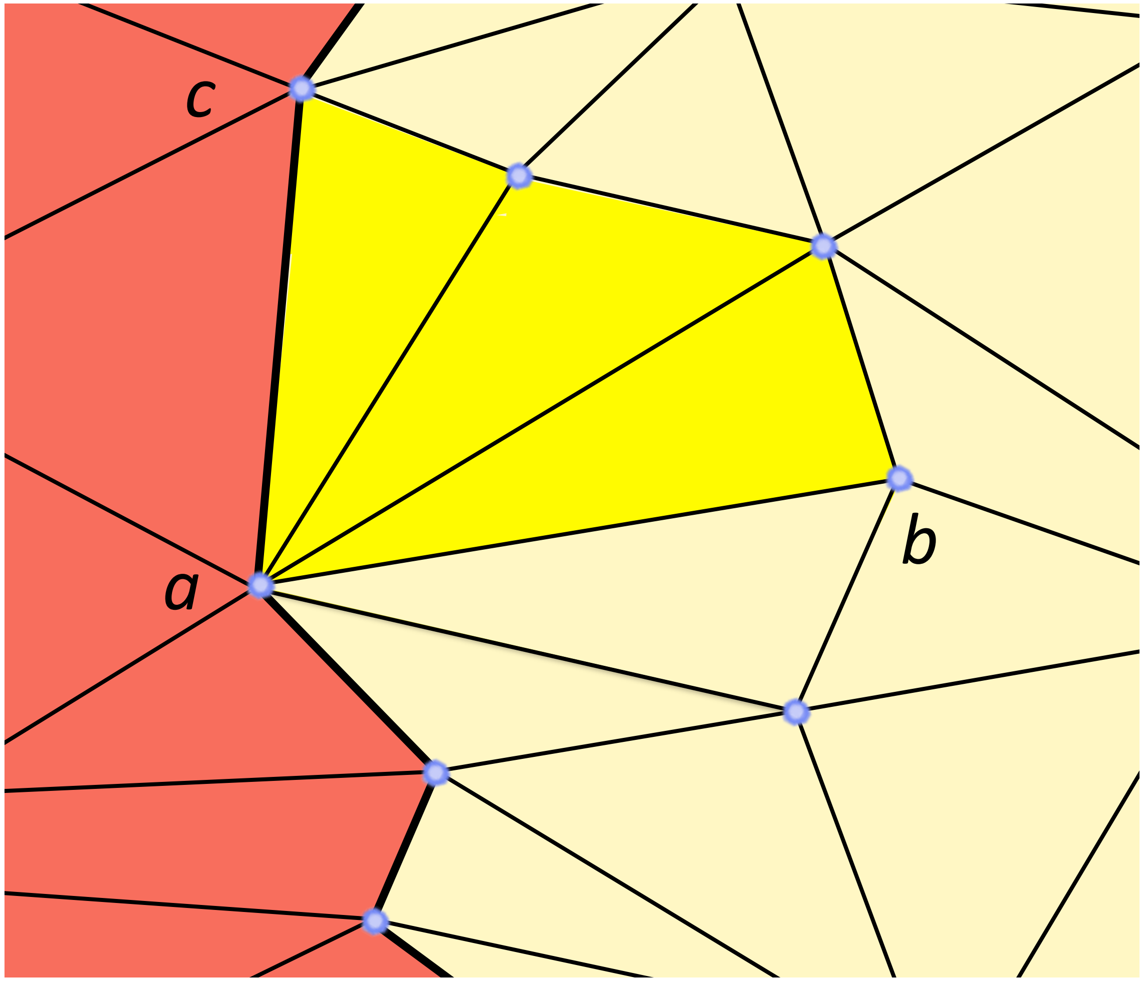


<u>First case of expansion</u> :

Creation of a facet *(abc)* towards a new vertex

(1)

Region Growing Strategy

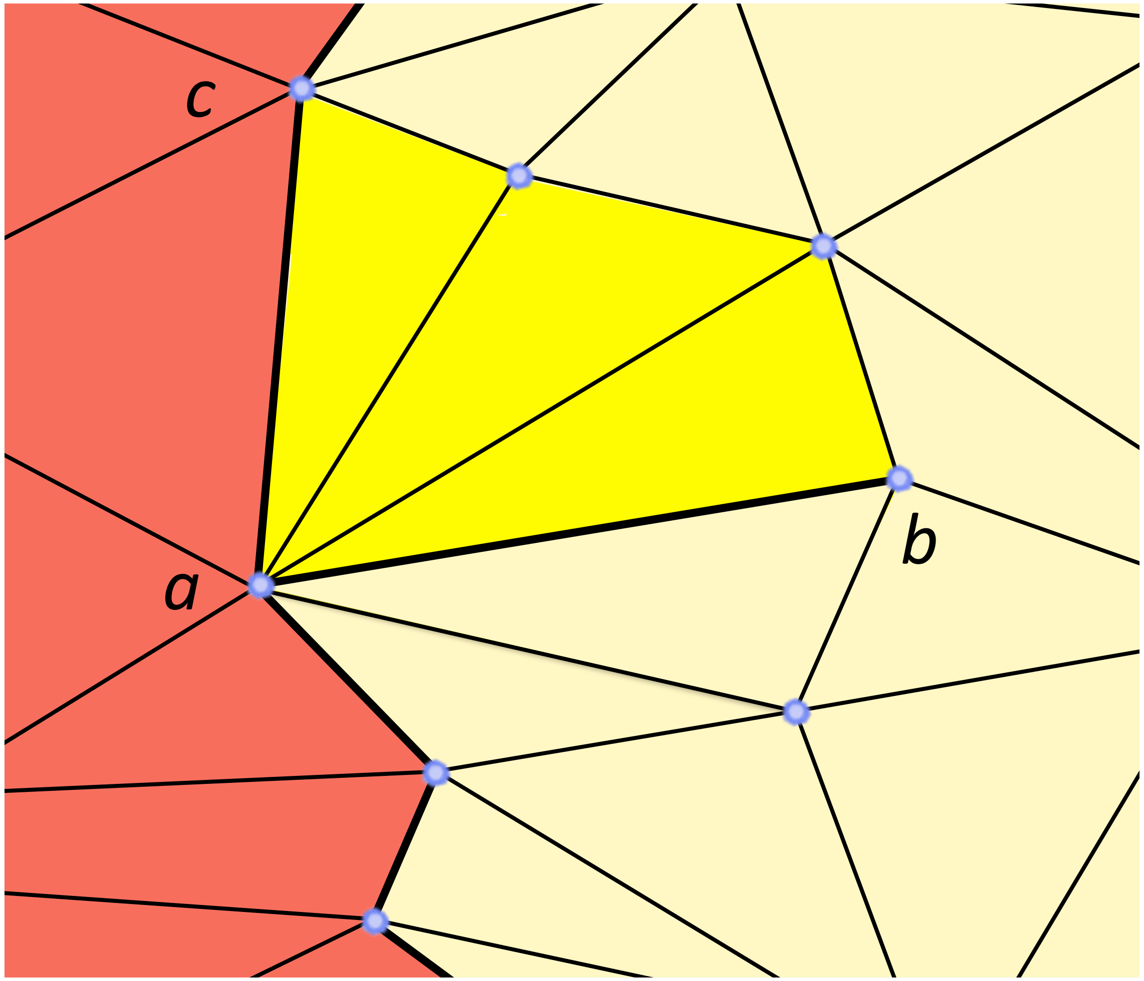


<u>First case of expansion</u> :

Creation of a facet *(abc)* towards a new vertex

(1)

Region Growing Strategy

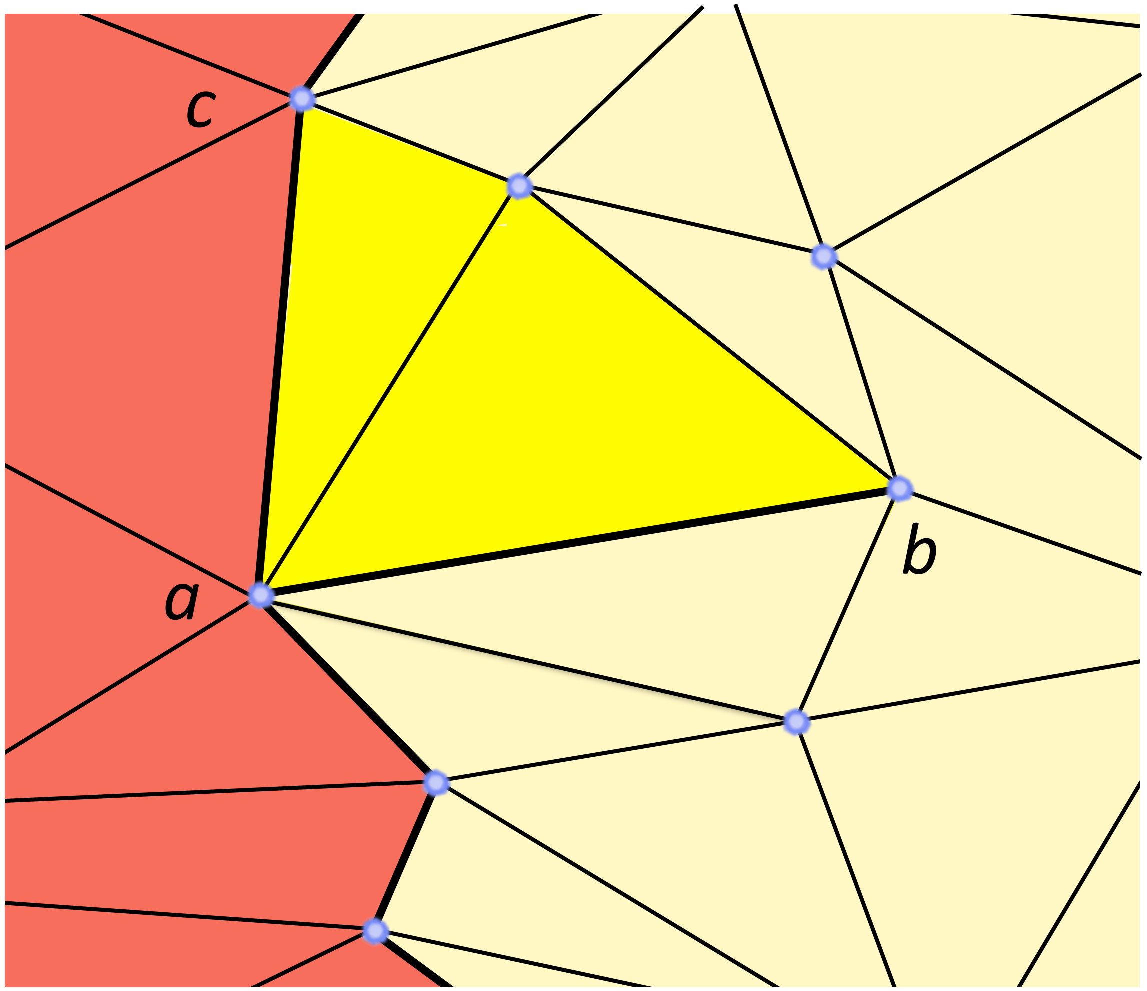


First case of expansion :

Creation of a facet *(abc)* towards a new vertex

(1)

Region Growing Strategy

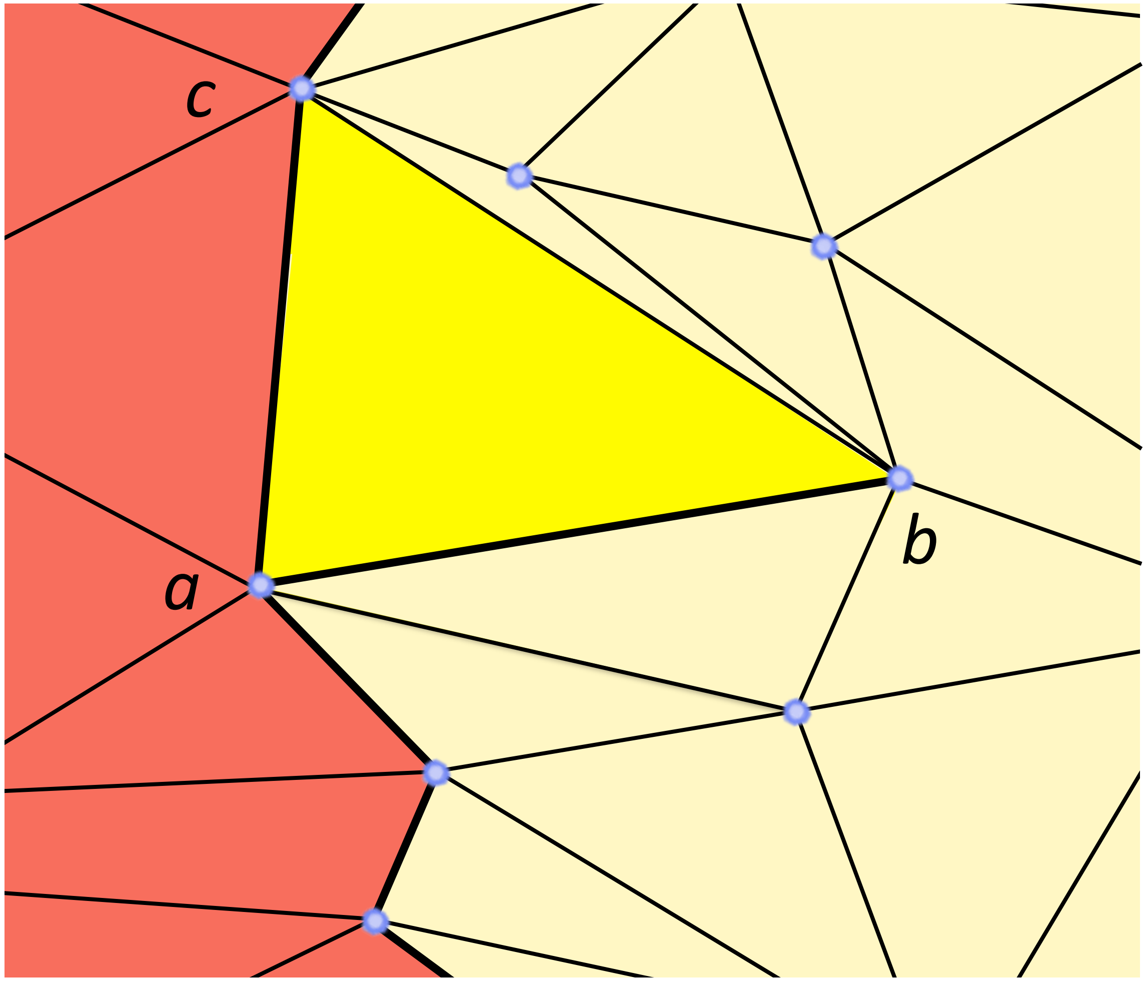


<u>First case of expansion</u> :

Creation of a facet *(abc)* towards a new vertex

(1)

Region Growing Strategy

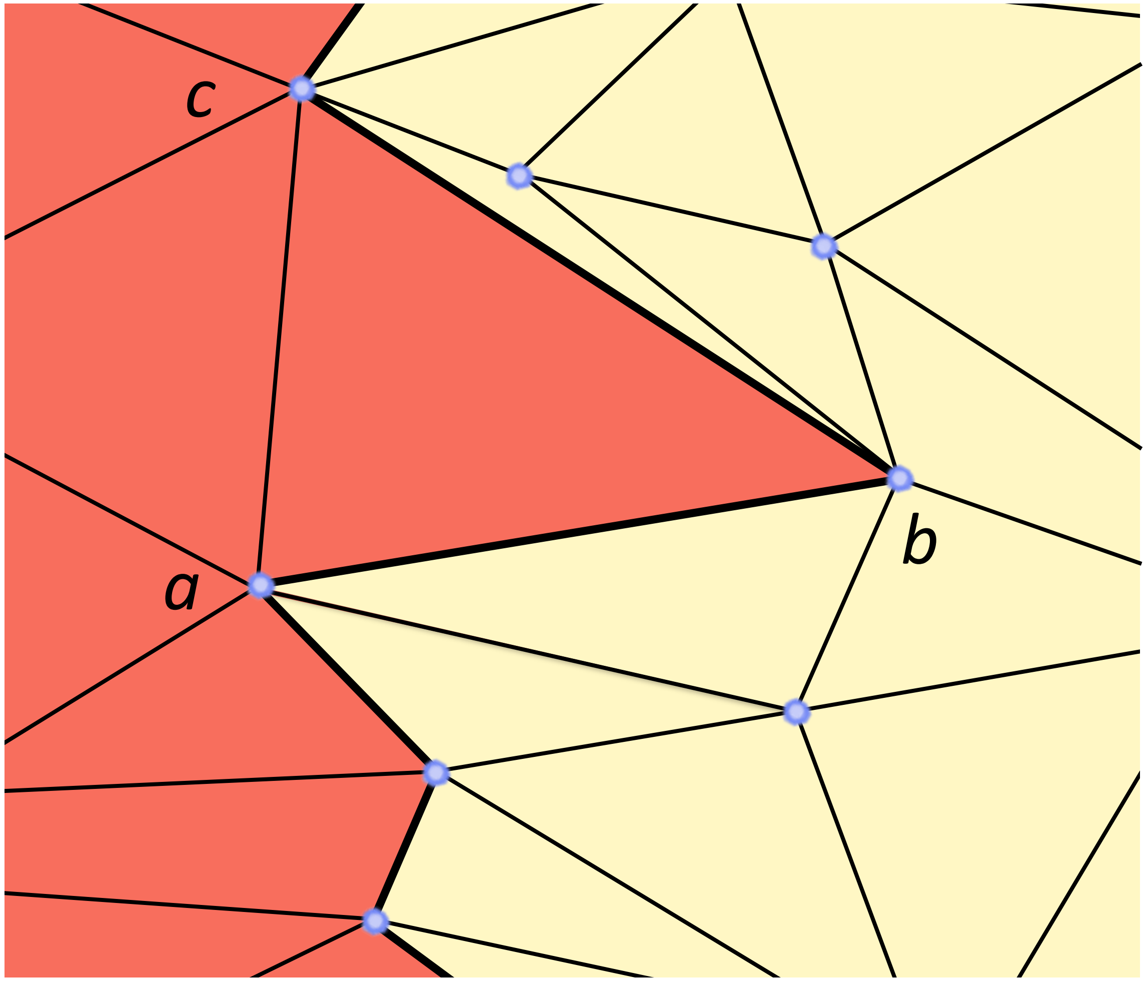


First case of expansion :

Creation of a facet *(abc)* towards a new vertex

(1)

Region Growing Strategy

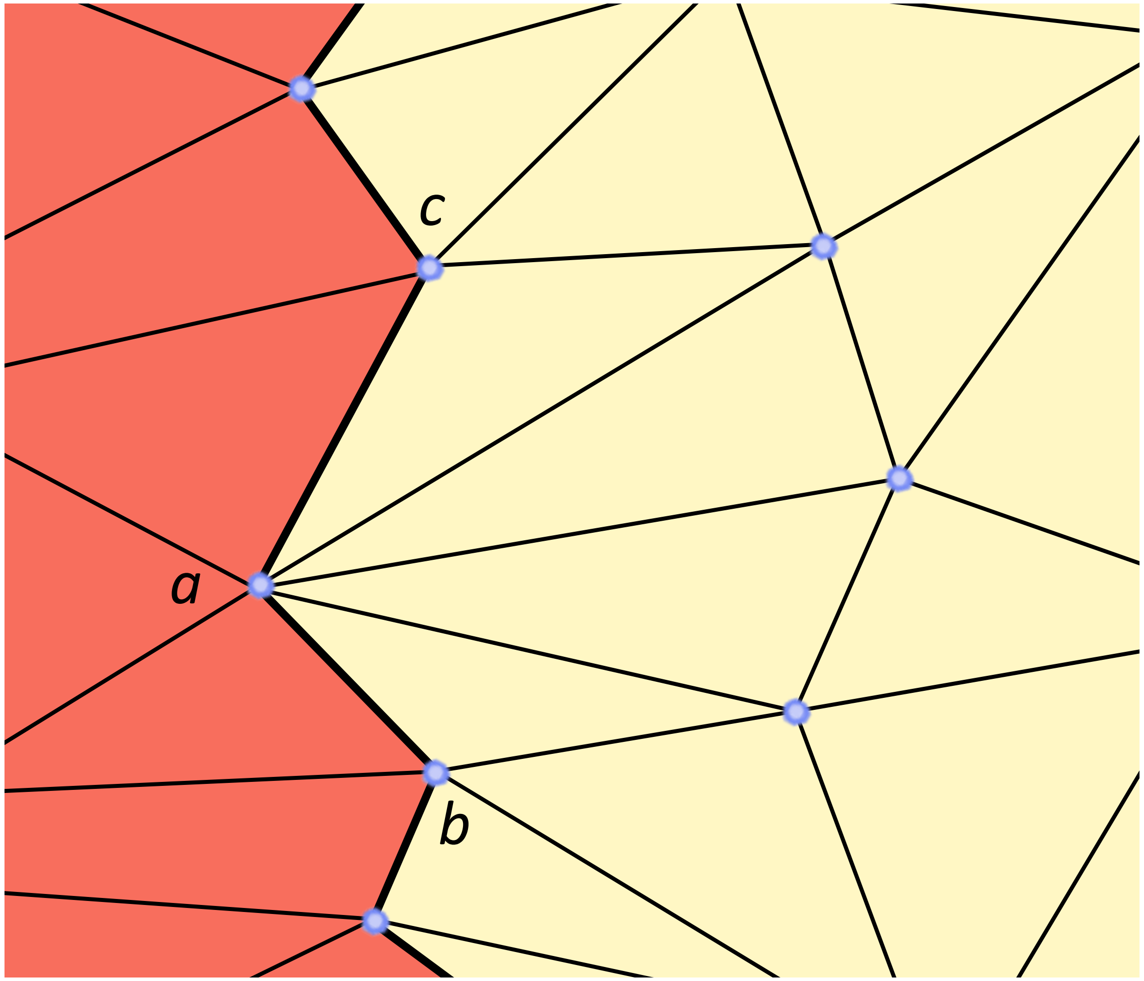


First case of expansion :

Creation of a facet *(abc)* towards a new vertex

(1)

Region Growing Strategy

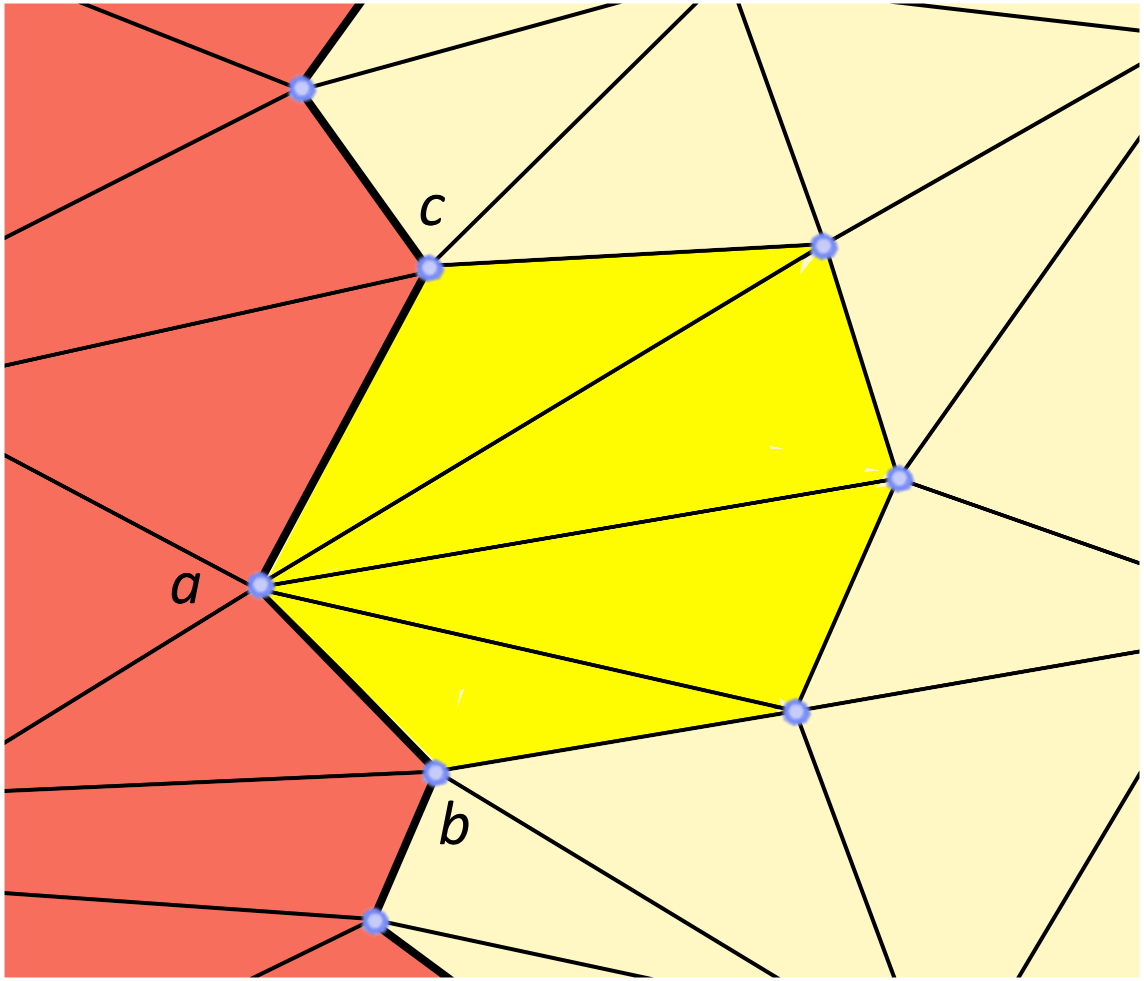


<u>Second case of expansion</u> :

Facet closure to create *(abc)*

(2)

Region Growing Strategy

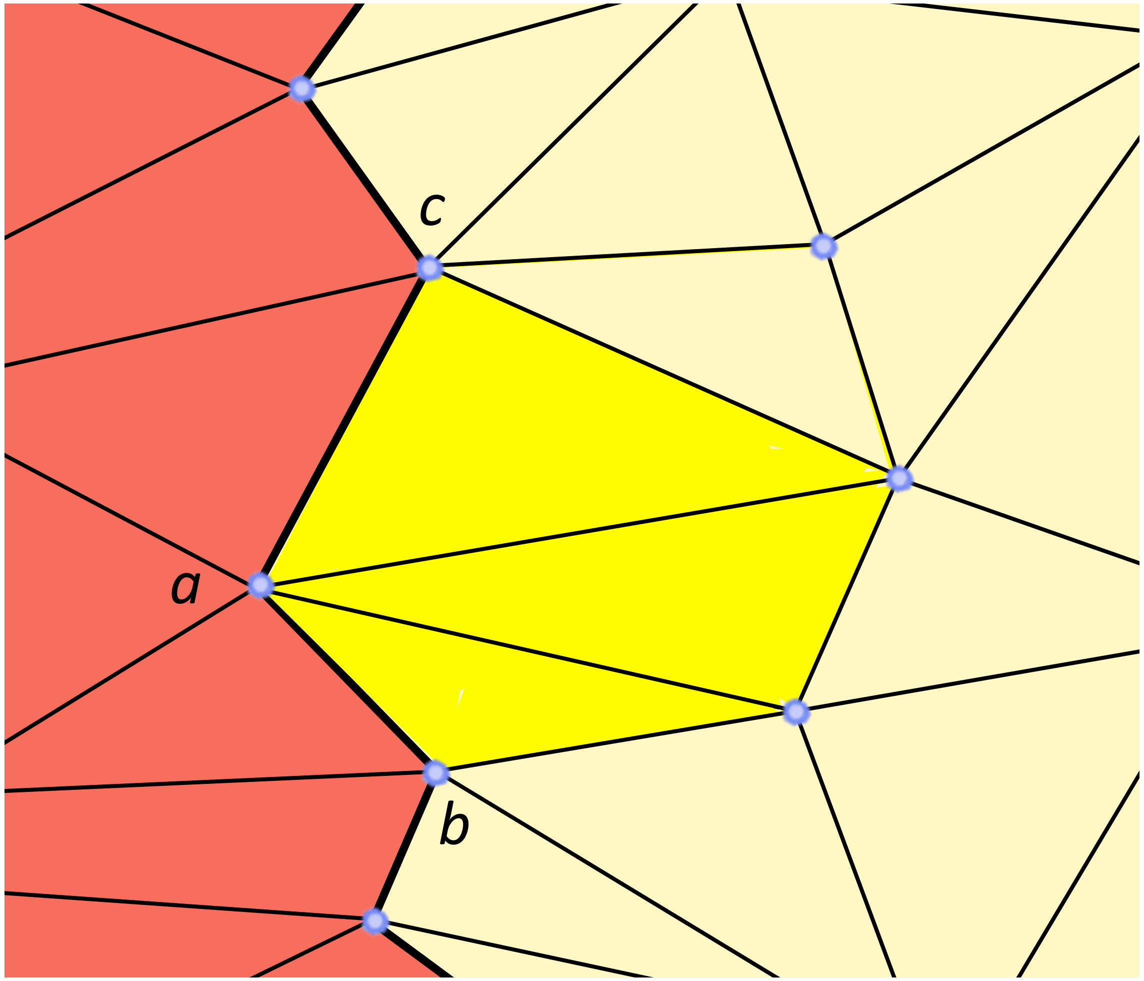


<u>Second case of expansion</u> :

Facet closure to create *(abc)*

(2)

Region Growing Strategy

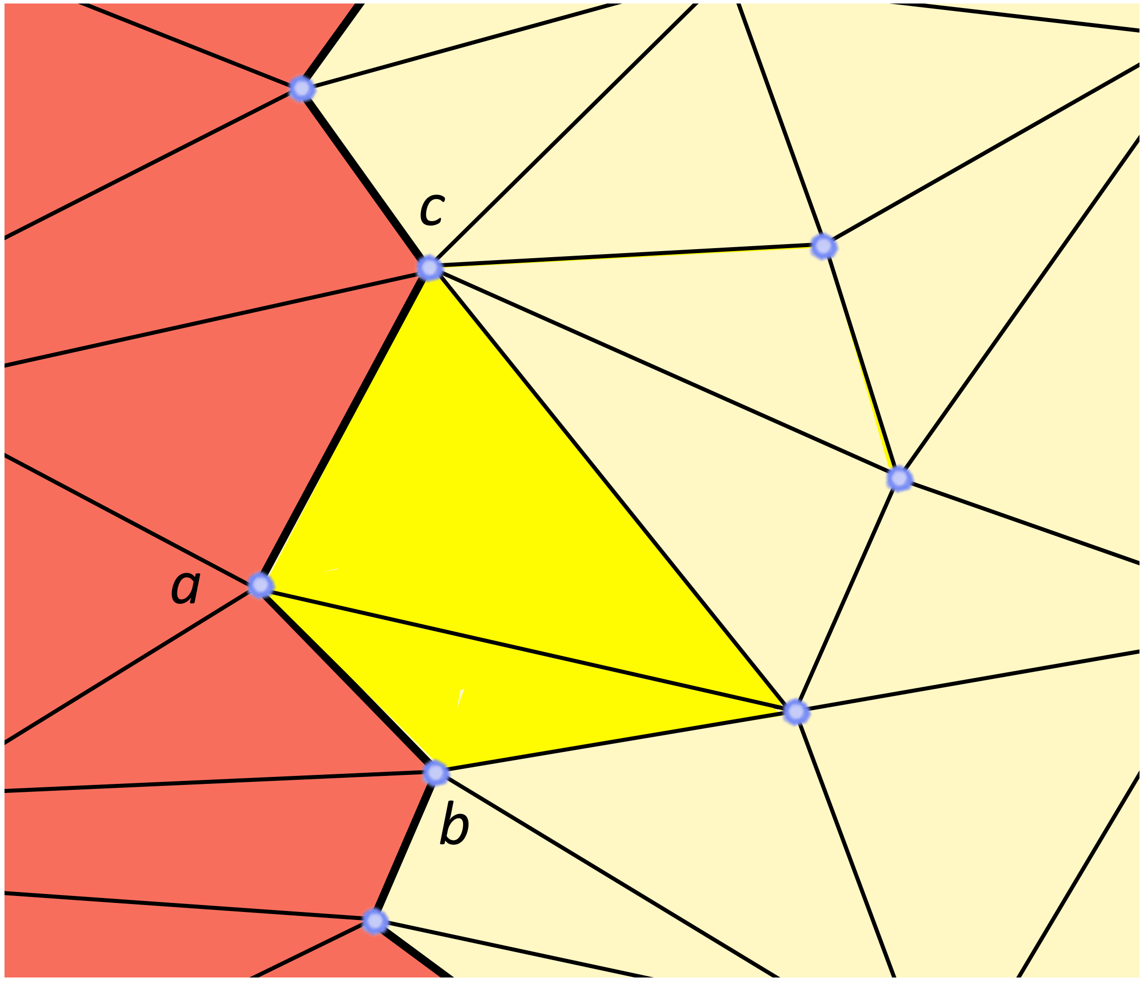


Second case of expansion :

Facet closure to create *(abc)*

(2)

Region Growing Strategy

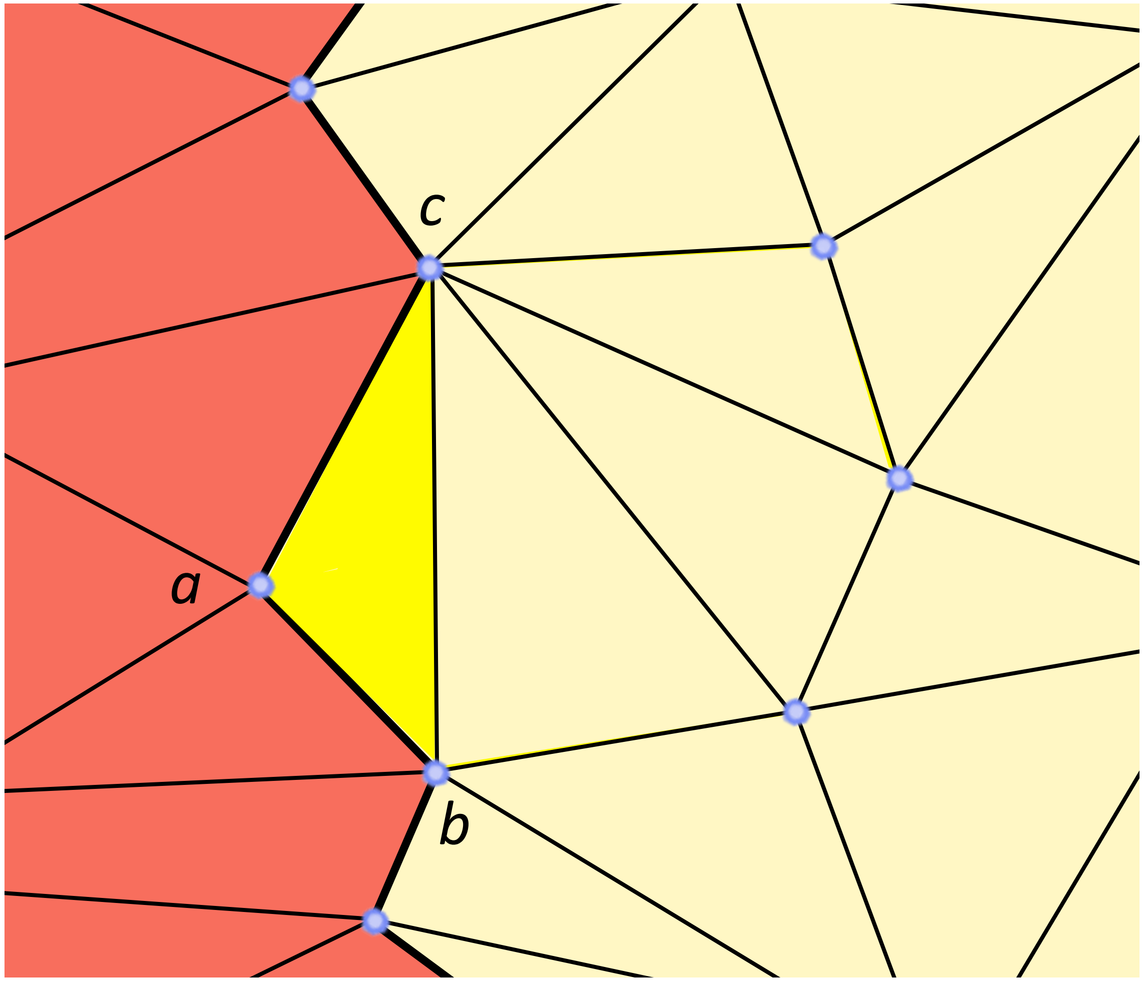


<u>Second case of expansion</u> :

Facet closure to create *(abc)*

(2)

Region Growing Strategy

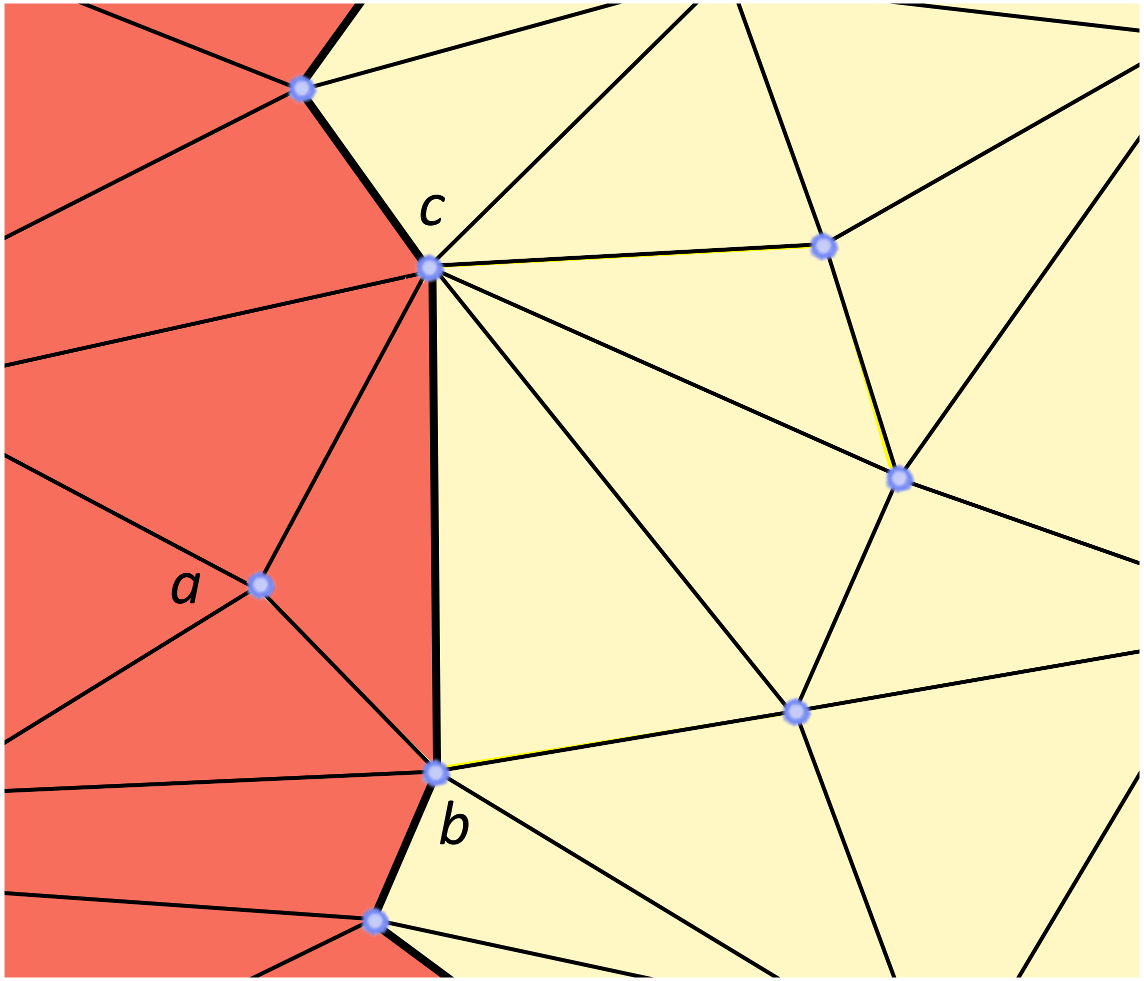


<u>Second case of expansion</u> :

Facet closure to create *(abc)*

(2)

Region Growing Strategy

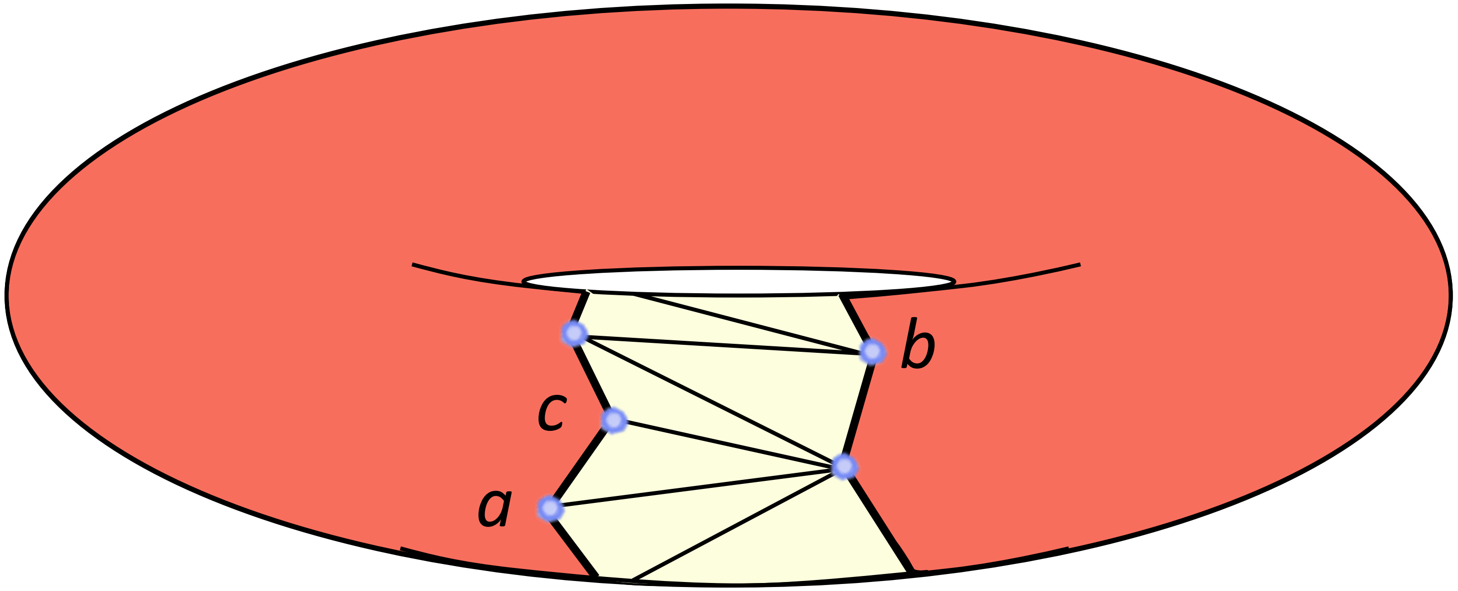


Second case of expansion :

Facet closure to create *(abc)*

(2)

Region Growing Strategy

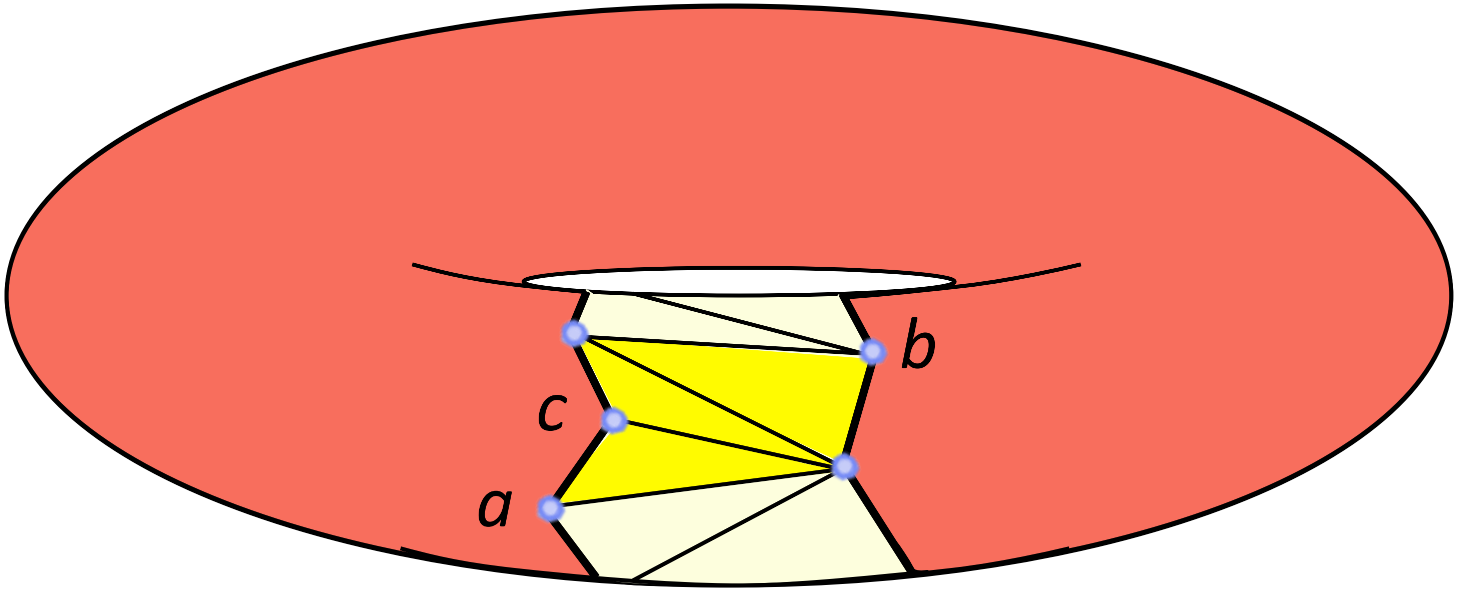


<u>Third case of expansion</u> :

Creation of a facet *(abc)* towards a previous vertex

(3)

Region Growing Strategy

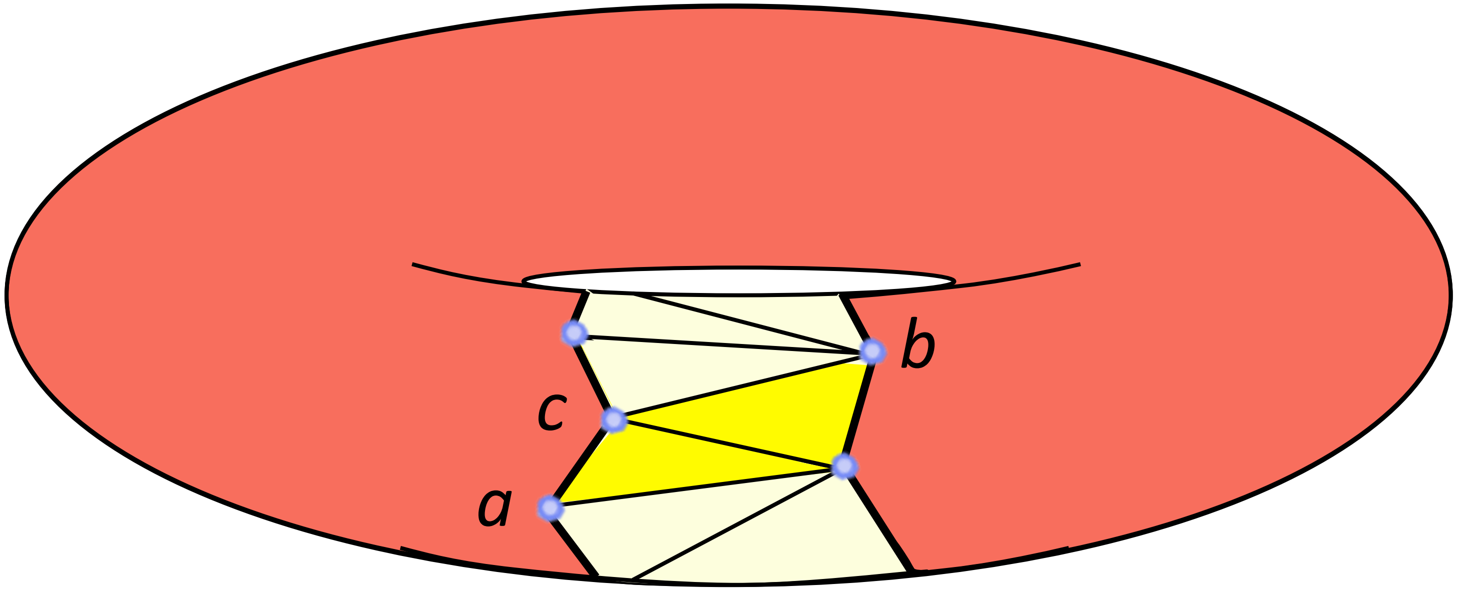


<u>Third case of expansion</u> :

Creation of a facet *(abc)* towards a previous vertex

(3)

Region Growing Strategy

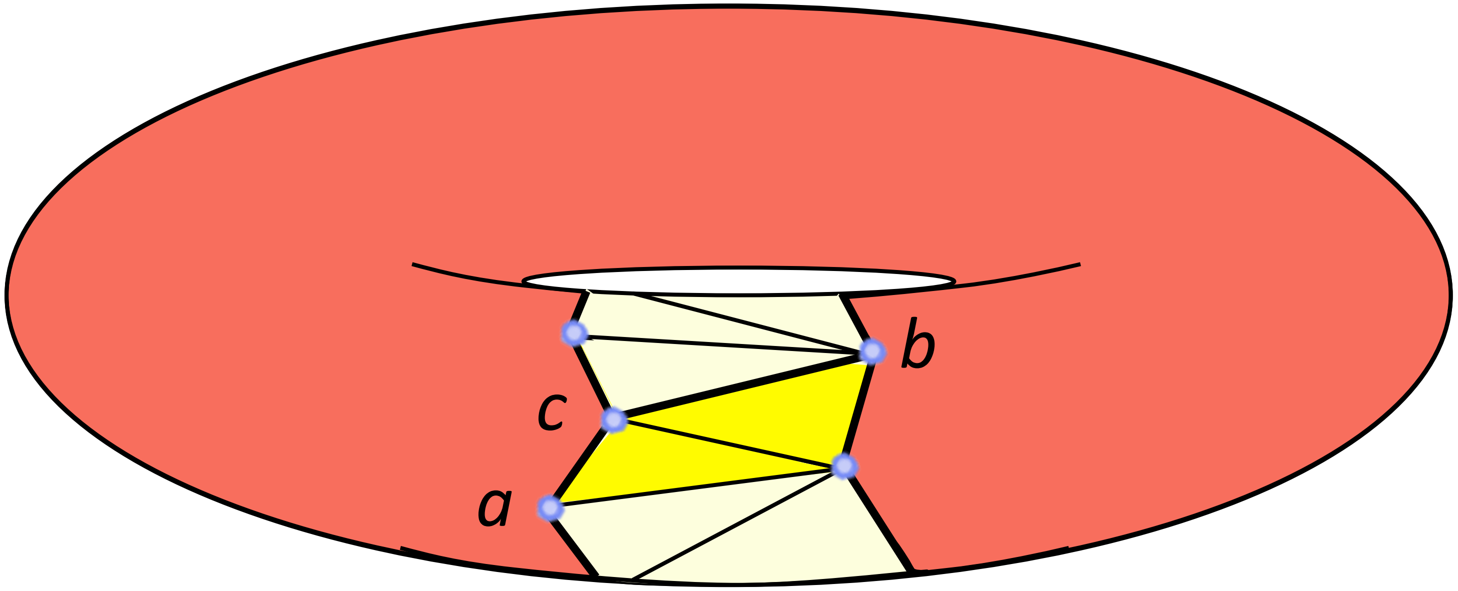


<u>Third case of expansion</u> :

Creation of a facet *(abc)* towards a previous vertex

(3)

Region Growing Strategy

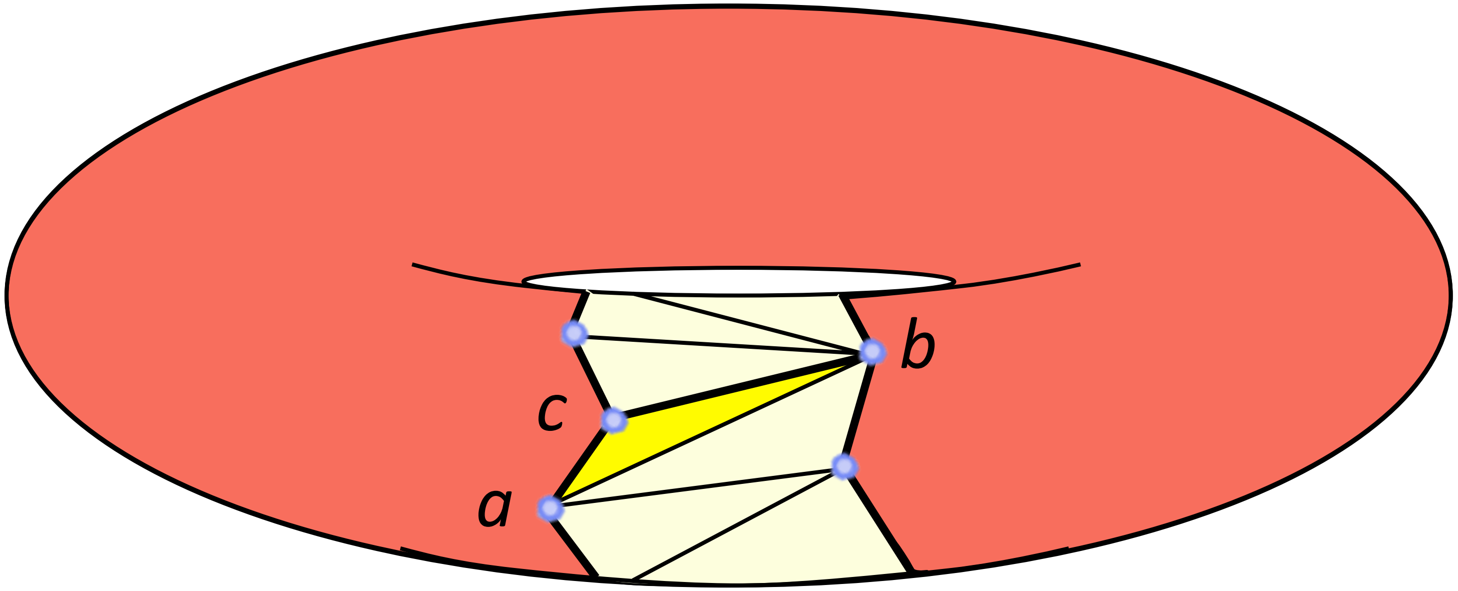


<u>Third case of expansion</u> :

Creation of a facet *(abc)* towards a previous vertex

(3)

Region Growing Strategy

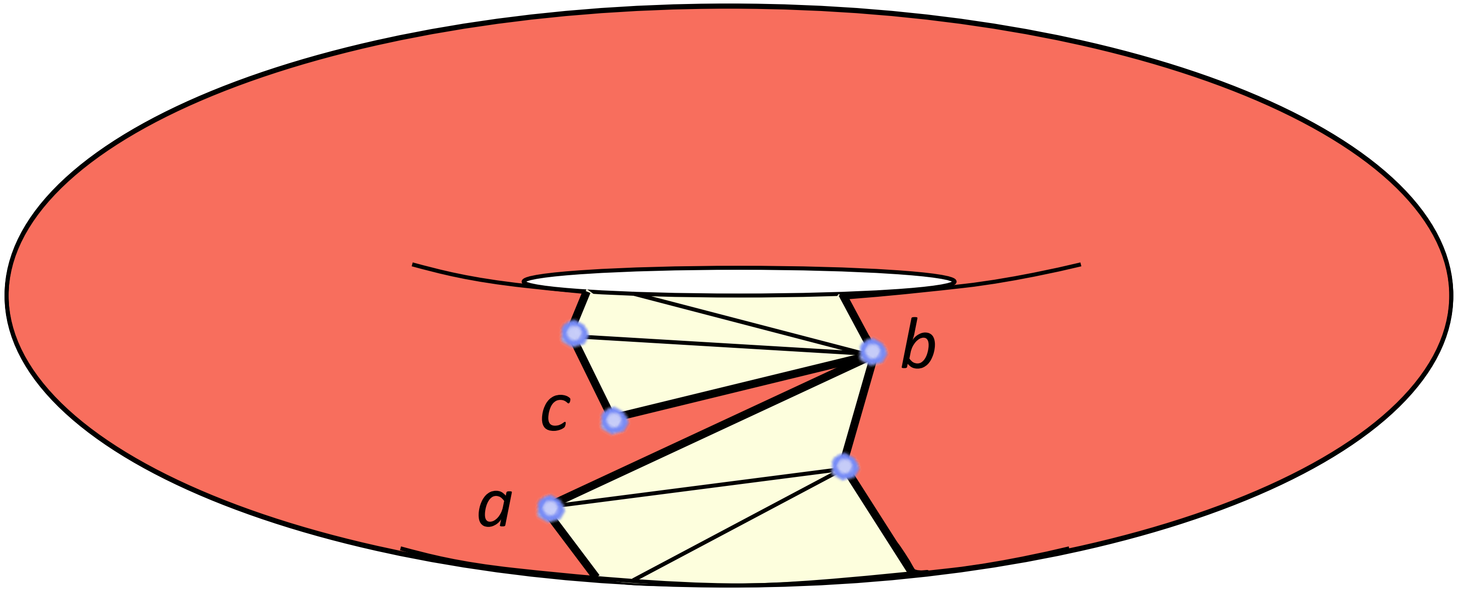


<u>Third case of expansion</u> :

Creation of a facet *(abc)* towards a previous vertex

(3)

Region Growing Strategy

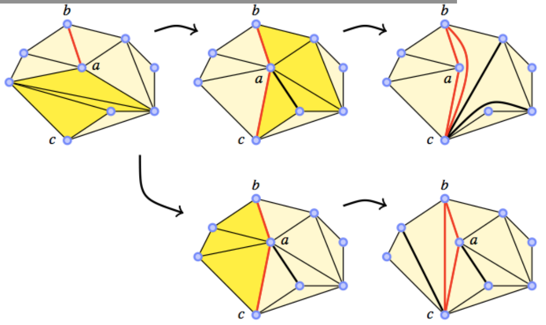


<u>Third case of expansion</u> :

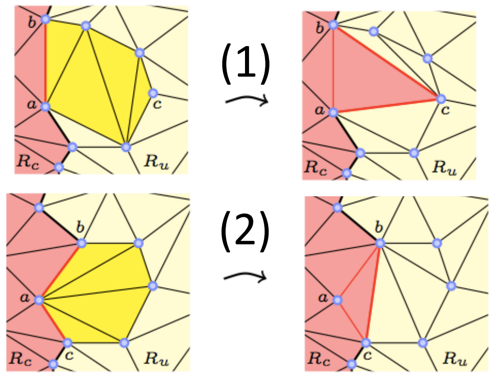
Creation of a facet *(abc)* towards a previous vertex

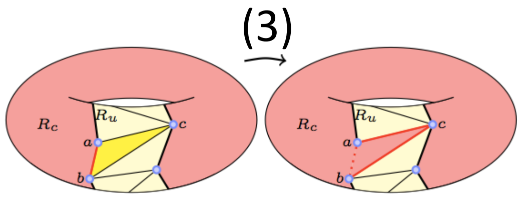
(3)

Algorithm



```
- Build the first facet (with the correct orientation)

While at least two facets remain to be built do

        While there is a facet corresponding to the cases (1) or (2) do
                - Build the facet
                - Block its edges
        end while

        If facets remain to be built then
                - Build a facet verifying the case (3)
                - Block its edges
        end if

End while
```





Specificities

- We can start from more seed facets,
- only the case (3) modifies the Euler characteristic of the region that contains constructed edges,
- the geometry plays no role in the algorithm,
- no use of a canonical connectivity configuration between the initial and target triangulations.

Conclusion and perspectives

Algorithm to determine a sequence of edge flips between two oriented, triangulated surfaces, with the same number of vertices and topological genus

Current works :

• generalize the algorithm to free it from a one to one correspondence between the vertices,

• generalize the algorithm so that the meshes do not have the same number of vertices and topological genus,

• simplify edge flips sequence,

• Embed the developped algorithm into compression strategies.

Thank you!