Polygonisation d'aires optimales

Année 2023-2024

Augustin Boonekamp, Antoine Teyssier, Azhar Gana, Charlotte Macaire, Maxence Hilbert,
Maxime Dalle, Tom Schmitt, Youta Kikuchi (3e, 2nd et 1er)

Établissement(s) : Lycée Français Vincent van Gogh, La Haye, Pays-Bas

Enseignant·e(s) : Florence Decool, Stéphane Beringue

Chercheur·Chercheuse(s) : Pierre Albert, université d'Utrecht aux Pays-Bas

Table des matières

1. Introduction

1.1. Sujet

Au début de l'année scolaire, nous avons reçu un objectif plutôt vague : trouver un moyen de déterminer de manière systématique les polygones avec les aires maximale et minimale d'un ensemble S de points dans un plan.

1.2. Définitions

Rappelons d'abord ce qu'est un polygones et les types de polygones éventuels.
— Polygone : une figure plane fermée dont chaque sommet est relié à deux autres sommets.
— Polygone concave : un polygone dont tous les angles sont inférieurs à 180 degrés.
— Polygone convexe : un polygone dont au moins un angle est strictement supérieur à 180 degrés.
— Polygone croisé : un polygone dont au moins deux segments se croisent.

2. Cas spécifiques

Dans cette section, nous proposons des solutions pour des polygones particuliers d'un certain nombre de points.

2.1. 3 points

Pour les polygones à 3 points, nous avons trouvé que l'aire maximale est égale à l'aire minimale , car on ne peut former qu'un seul polygone. On a alors que $A_{\text{min}} = A_{\text{max}}$

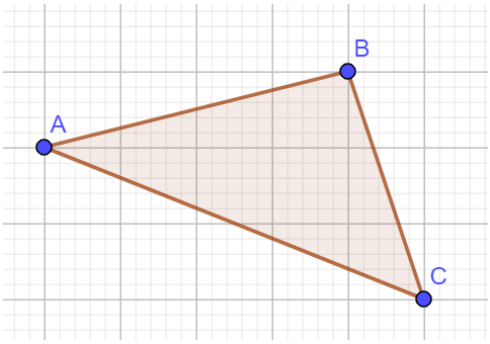

FIGURE 1 – Aire minimale et maximale pour 3 points

2.2. 4 points

2.2.1 Conjecture

Pour les polygones à 4 points, nous avons conjecturée une méthode qui permet de trouver l'aire minimale. La conjecture se base sur le polygone croisé construit à partir des diagonales de l'aire maximales. Voici la conjecture :

Proposition. *Si* $a+c>b+d$, *alors* $A_{\text{min}} = A(ACBD)$
Sinon $A_{\text{min}} = A(ABDC)$

Voici un exemple pour illustrer la conjecture :

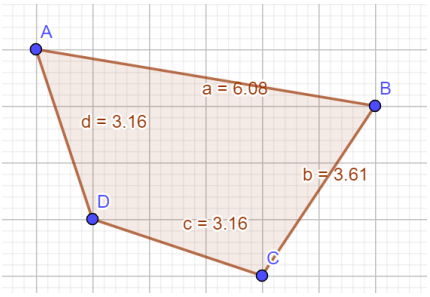

FIGURE 2 – Aire maximale

Ici, $a+c = 6.08+3.16 = 9.24$ et $b+d = 3.16+3.61 = 6.77$
Donc, $a+c > b+d$, alors $A_{\text{min}} = A(ACBD)$

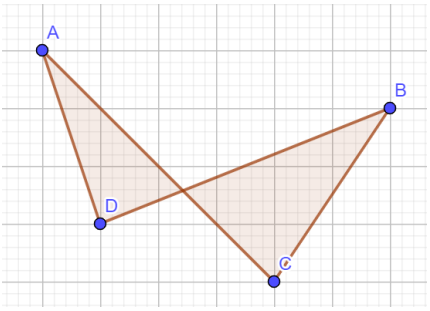

FIGURE 3 – Aire minimale

2.2.2 Exceptions

Cette conjecture ne fonctionne pas avec les points dont l'aire maximale est un polygone concave comme ci-dessous :

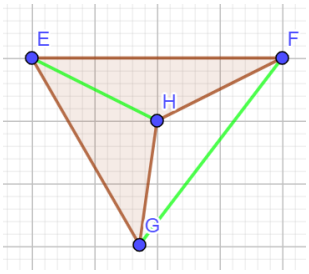

FIGURE 4 – Exception des polygones concaves

On peut voir que les diagonales (en vert) ne se coupent pas ce qui ne permet pas de trouver le polygone croisé qui correspond à l'aire minimale.

De plus, la conjecture ne fonctionne pas lorsque 3 points sont alignés : lorsque trois points sont alignés, le même problème apparaît. La conjoncture nous indique que l'aire maximale est A_{NVWO} alors que l'aire minimale est A_{NVOW}.

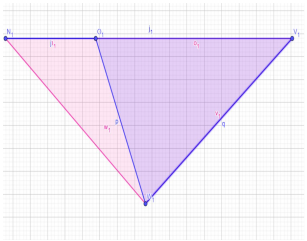

FIGURE 5 – Exception des polygones avec 3 points alignés

2.3. Trouver l'aire maximale

2.3.1 Méthode 1

Afin de trouver l'aire maximale, nous avons émis une première hypothèse. Nous avons placé un point aléatoirement au milieu des autres points puis nous avons créé une demi-droite qui tourne autour de ce point. Au début, cette demi-droite est horizontale et orientée vers la gauche. Voici un exemple avec 6 points et avec G (en rouge) le point autour de laquelle tourne la demi-droite.

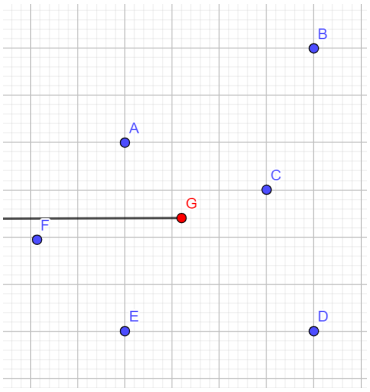

FIGURE 6 – Première étape

Nous allons ensuite commencer à faire tourner la demi-droite dans le sens des aiguilles d'une montre jusqu'à ce qu'elle rentre en contact avec un point. Dans l'exemple, la demi-droite arrive au point A, en vert.

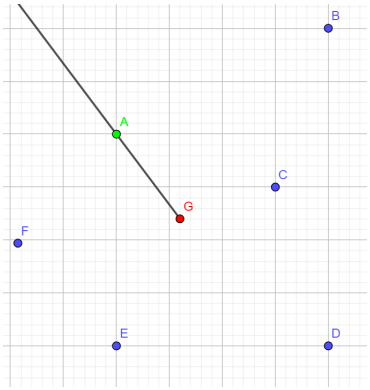

FIGURE 7 – Deuxième étape

Ensuite, nous avons continué à faire tourner la demi-droite vers un deuxième point, B et nous avons relié les deux premiers points. Sur la figure, le segment créé est en vert.

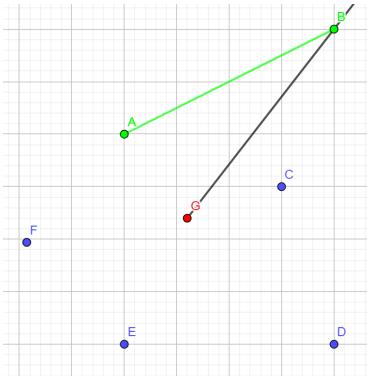

FIGURE 8 – Troisième étape

Nous allons répéter l'étape précédente afin de relier tous les points les uns aux autres et retrouver la position initiale de la demi-droite. On obtient alors l'aire maximale.

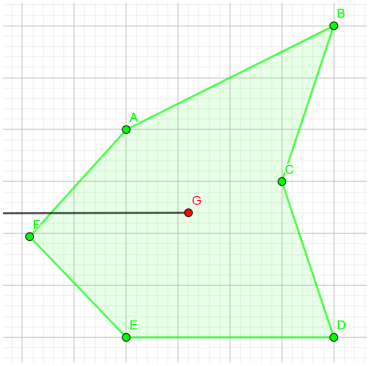

FIGURE 9 – Aire maximale

2.3.2 Méthode 2

Pour trouver l'aire maximale avec un nombre indéfini de points, on a émis une deuxième hypothèse qui consiste à relier tous les points par des segments. On ne garde ensuite que les segments qui font le contour extérieur pour former l'aire maximale.

Voici un exemple avec 5 points :

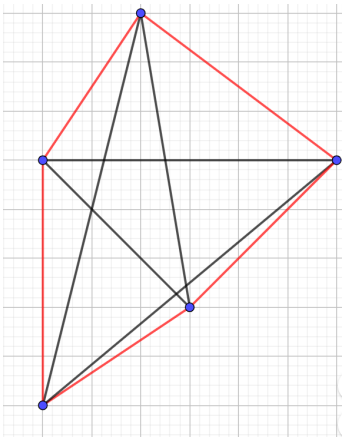

FIGURE 10 – Première étape

Après avoir tracé tous les segments, nous avons mis les segments extérieurs en rouge. Nous avons alors trouvé l'aire maximale.

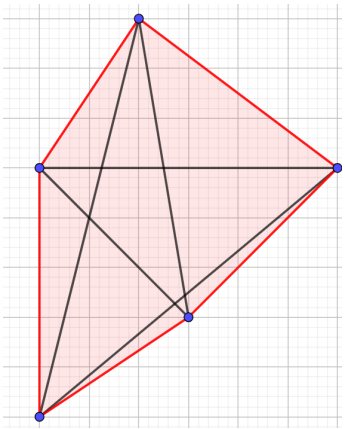

FIGURE 11 – Aire maximale

Cette méthode ne fonctionne pas avec les polygones concaves.

2.4. 5 points

Pour les 5 points, on utilise la même méthode pour trouver l'aire maximale.
Nous n'avons pas trouvé de formule générale pour déterminer l'aire minimale.

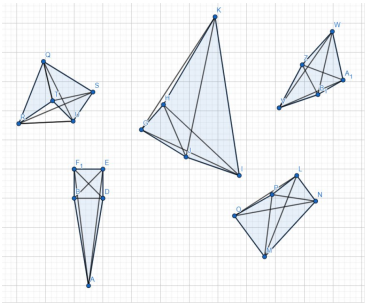

FIGURE 12 – 5 Points

3. Cas généraux

3.1. Relier le point le plus proche

3.1.1 Explication de la conjecture

Dans un ensemble de points S, on détermine les distances entre deux points et on relie chaque point aux deux points les plus proches. Ceci nous donne pour de nombreux ensembles, des polygones possédant l'aire maximale, mais l'on rencontre souvent des segments en trop, ce qui sera développé dans les prochains exemples.

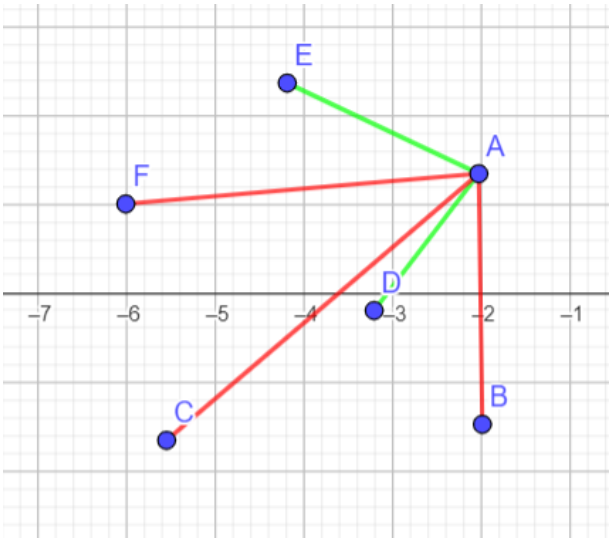

FIGURE 13 – Explication de la conjecture par image

Sur l'image ci-dessus, soient A, B, C, D, E, F six points du plan. Notre conjecture relie d'abord le point A à tous les points. Ensuite, il garde les segments les moins longs : $[AD]$ et $[AE]$(montrés par les segments verts) et supprime les autres : $[AB]$,$[AC]$,$[AF]$(montrés par les segments rouges).

Cette méthode est ensuite répétée sur tous les points afin d'obtenir la figure 14 ci-dessous.

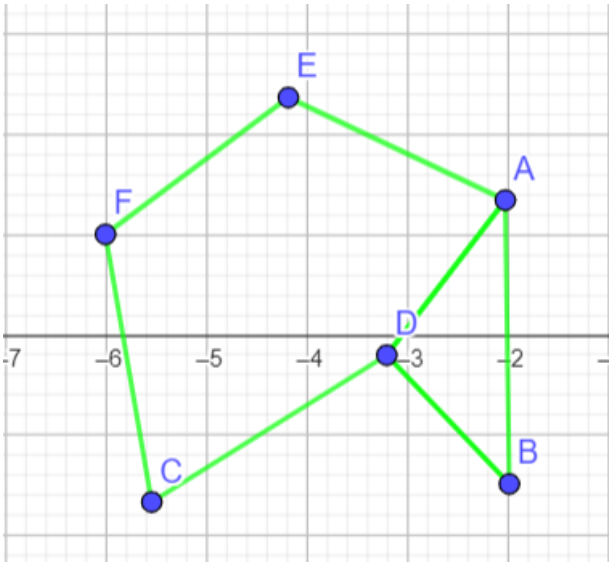

FIGURE 14 – Résultat

3.1.2 Exemples

Soient A, B, C, D, quatre points du plan, la conjecture donne la figure ci-dessous. Les points reliés par les segments rouges sont les plus proches entre eux, et D et B sont les points les plus proche de C. Un polygone à n sommets devrait avoir n segments. La figure formée par les sommets A, B, C et D a cinq segments, donc un de trop. On choisit de retirer le segment $[BD]$ car il relie deux sommets déjà reliés à trois autres sommets, et l'on obtient bien le polygone possédant l'aire maximale.

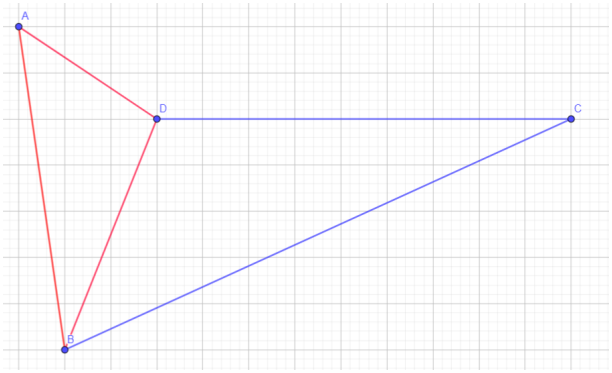

FIGURE 15 – Exemple à 4 points

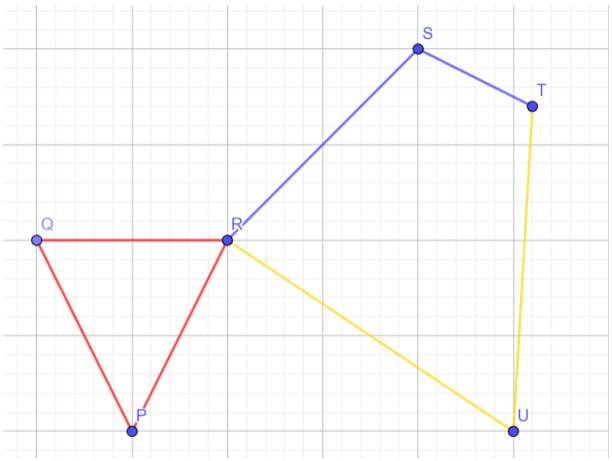

FIGURE 16 – Exemple à 6 points

Sur ce deuxième exemple, soient P, Q, R, S, T et U, six points du plan. On obtient la figure ci-dessus en appliquant la méthode précédente. Les points Q, R et P sont les plus proches entre eux, R et T sont les plus proches de S et de U. Sur cette figure, le sommet R est relié à quatre sommets, soient deux de trop. On souhaite supprimer deux segments d'extrémité R. On remarque que supprimer les couples de segments $([RQ],[RP])$ ou $([RS],[RU])$ ou $([RQ],[RU])$ ou $([RP],[RS])$ ne permettent pas de reconstruire une figure une fois supprimés. En comparant les aires des polygones $PQSTUR$ et $PQRSTUP$ formés en supprimant les couples de segments $([QR],[RS])$ et $([RP],[RU])$ respectivement et en reliant Q à S et P à U respectivement, on trouve que le polygone $PQRSTUP$ est le polygone d'aire maximale. Cependant, nous n'avons pas trouvé le raisonnement qui permet de décider de la suppression nécessaire.

3.1.3 Contre-exemple

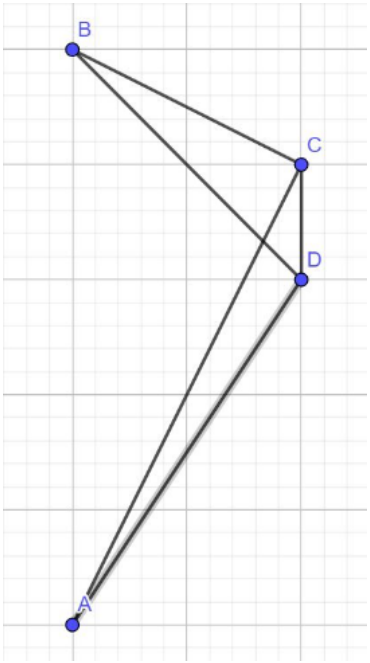

FIGURE 17 – Contre-exemple

En continuant d'expérimenter avec des exemples, un contre-exemple à la conjecture a été trouvé. Soient A, B, C, D, quatre points du plan de même disposition que sur la figure ci-contre, la conjecture nous donne le polygone $ACDB$ qui est un polygone croisé et n'est évidement pas le polygone d'aire maximale qui est $ABCD$. En conclusion, la conjecture ne fonctionne pas pour toutes les dispositions de points. Comme il n'est pas possible de déterminer lesquelles, cette conjecture est rejetée.

3.2. Triangulation

3.2.1 Triangulation algébrique

Afin de calculer l'aire de manière optimale, on découpe le polygone en triangles. Pour former les triangles au sein du polygone, on crée des segments dont les sommets sont des sommets de polygones et qui sont strictement inclus dans le polygone. Ensuite, on utilise les formules de trigonométrie sur chacun des triangles formés.

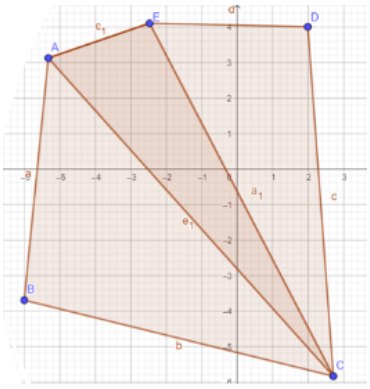

FIGURE 18 – Polygone $ABCDE$ découper en trois triangles

Dans la figure 18 ci-dessus, nous disposons de cinq points A, B, C, D, E. Ils forment un polygone convexe $ABCDE$. On peut donc découper ce polygone en triangles. On forme les segments $[AC]$ et $[EC]$ car ils sont inclus dans le polygone. On découpe le polygone en trois triangles ACE, BCA et DCE; le troisième triangle ACE est ainsi automatiquement formé. On utilise ensuite la trigonométrie pour calculer et déterminer l'aire du triangle.

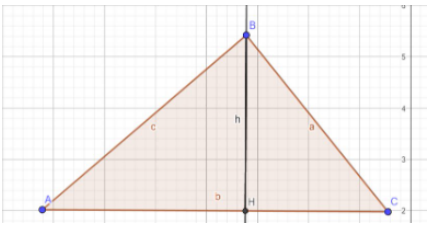

FIGURE 19 – Triangle ABC avec la hauteur h

On utilise la formule générale pour calculer l'aire d'un triangle :

$$A = \frac{a \times b \times \sin \widehat{ACB}}{2}$$

On refait de même pour tous les triangles qui composent un polygone.

3.2.2 Algorithme triangulation

Nous proposons un algorithme pour trouver l'aire optimale d'un polygone à n côtés. On va l'appeler l'algorithme de triangulation par oreille.

1. **L'identification des oreilles** : commencer par identifier tous les sommets du polygone qui forment ce que l'on va appeler des "oreilles". Une oreille est une diagonale du polygone qui ne croise pas d'autres arêtes du polygone et qui se situe à l'intérieur du polygone. Plus simplement une oreille est un triangle formé par trois sommets du polygone, dont deux sont consécutifs sur le bord du polygone et le troisième est un sommet intérieur du polygone.

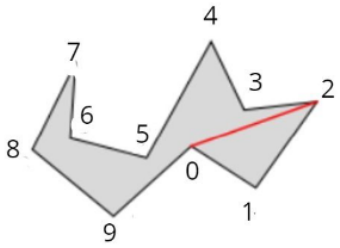

FIGURE 20 – Identification de l'oreille

2. **Surpression de l'oreille** : Choisir l'une des oreilles identifiées et retirer-la du polygone en connectant ses voisins avec une diagonale. Cela forme un triangle et réduit le nombre de côtés du polygone.

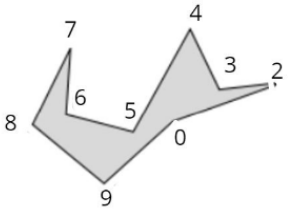

FIGURE 21 – Surpression de l'oreille

3. **Répétition de l'opération** : Répéter l'opération d'identification et de surpression des oreilles jusqu'à ce qu'il ne reste plus que 3 sommets dans le polygone

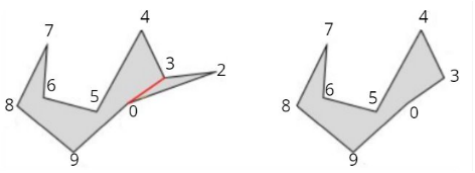

FIGURE 22 – Répétition

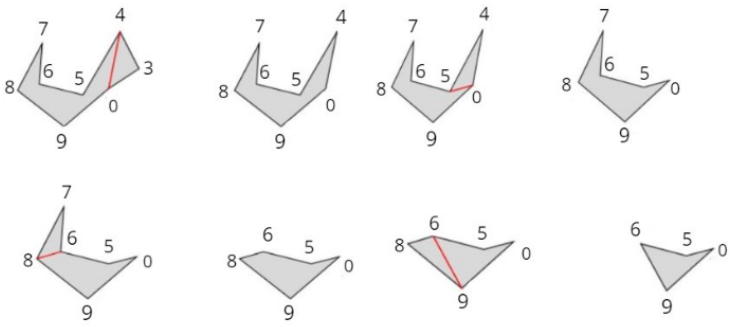

FIGURE 23 – Répétition jusqu'aux quatre sommets restants

4. **Connexion des trois derniers sommets** : lorsqu'il reste seulement trois sommets, relier-les entre eux pour former le dernier triangle et ainsi terminer la triangulation du polygone.

FIGURE 24 – Relier les 3 derniers sommets

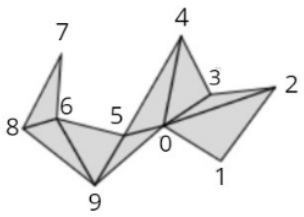

FIGURE 25 – Triangulation du polygone complète

5. **Trouvez l'aire maximale et l'aire minimale du polygone** : permuter les 10 points, jusqu'à trouver le polygone avec l'aire maximale et minimale grâce à la formule de l'aire d'un triangle.

   Ainsi on aura : $10! = 10 \times 9 \times 8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1 = 3628800$ polygones.

   On arrive donc à cet algorithme :

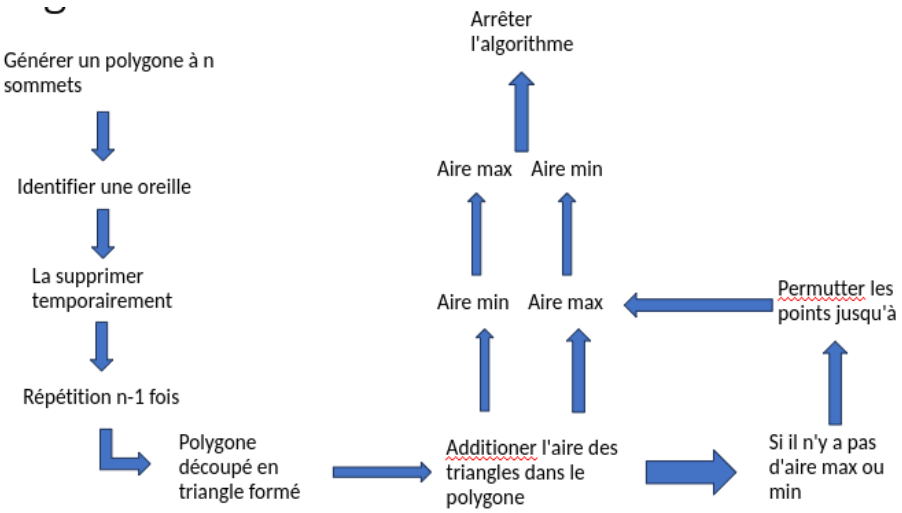

FIGURE 26 – L'algorithme de la triangulation d'aire optimale

4. Polygones à trous

En générant des polygones croisés, nous remarquons que certains possèdent une ou plusieurs sections entièrement côtoyées par d'autres, qui se trouvent ainsi isolées de l'extérieur du polygone. Nous distinguons alors deux conventions définissant l'aire de telles figures (voir Figure 27) :

— La règle *NonZero* (Non-nul) : Toutes les sections du polygone contribuent à son aire.
— La règle *EvenOdd* (Parité) : Certaines sections du polygone, définies comme étant des trous d'après une règle de parité (voir section 4.1), ne contribuent pas à son aire.

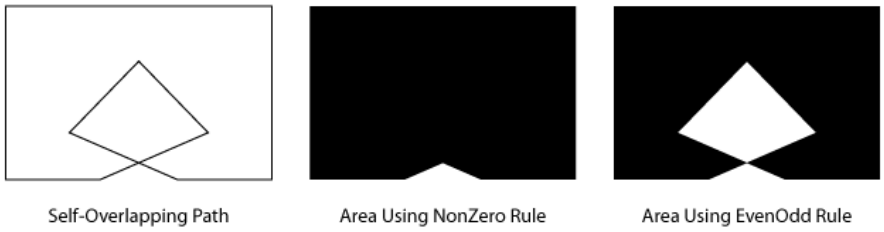

FIGURE 27 – Conventions d'aire

Dans cette section, nous résoudrons le problème le plus généralisé qui s'applique aux polygones à trous. Nous adoptons alors la règle de parité comme convention.

4.1. Formalisation d'un trou

La règle de parité définit si un point contribue ou non à l'aire d'un polygone, en fonction du nombre de fois que l'on croise les segments du polygone en se déplaçant du point vers l'infini dans une direction donnée.

Sous forme d'étapes, elle donne :

1. Choisir un point à l'intérieur du polygone.
2. Tracer une demi-droite partant de ce point dans une direction quelconque, ne passant par aucun sommet et aucune auto-intersection du polygone.
3. Compter combien de fois cette demi-droite croise les segments du polygone.
4. Si le nombre de croisements est pair, alors le point se trouve dans un trou.
   Si le nombre de croisements est impair, alors le point contribue à son aire.

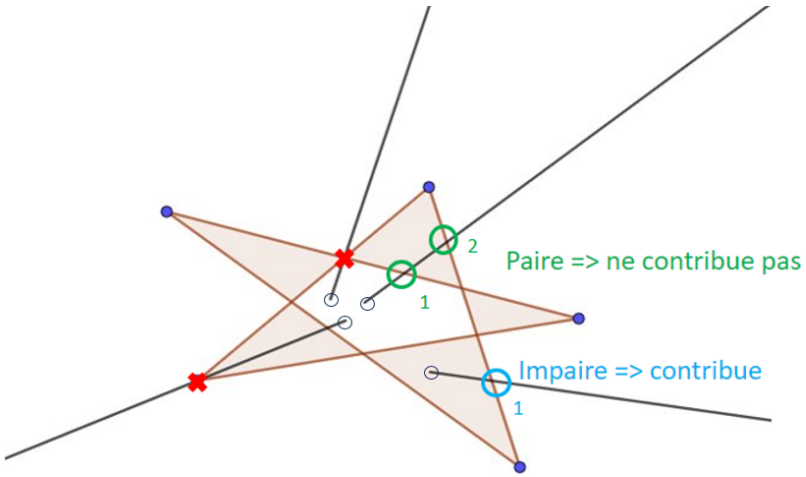

FIGURE 28 – Description graphique d'un trou

4.2. Calcul d'aires à trous

Nous présentons ci-dessous la description d'une solution algorithmique pour calculer l'aire d'un polygone à trous, que nous nommons "algorithme à tranches verticales" [1] :

- Pour chaque auto-intersection du polygone ainsi que ses sommets, une droite verticale passant par ce point est tracée.
- Le corps du polygone entre deux droites verticales consécutives est alors composé de trapèzes ou de triangles (cas particulier). La somme des aires de tous ces trapèzes représente l'aire totale du polygone.
- Les segments se trouvant entièrement entre deux droites consécutives peuvent être ordonnés de haut en bas. Selon la règle du parité, traverser le premier segment mène à être à l'intérieur du polygone, traverser le deuxième segment mène à l'extérieur, le troisième, à l'intérieur de nouveau etc... Ainsi, si l'on regroupe les segments par paires consécutives, chaque paire représente les côtés d'un trapèze, dont l'aire est donnée par $A = \frac{a+b}{2}h$.

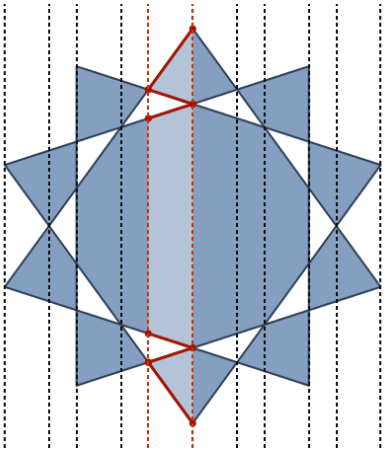

FIGURE 29 – Illustration du calcul d'aire par "tranches verticales"

La quantité de ressources nécessaires pour traiter des entrées est une fonction de n, taille de l'entrée, est nommée sa complexité. Dans l'ensemble, cet algorithme a une complexité de $\mathcal{O}(n^3)$.

Nous notons que la première étape qui consiste à trouver toutes les intersections dans le polygone peut s'effectuer grâce à un algorithme tel que celui de Bentley et de Ottmann [2] .

4.3. Algorithme "force brute" optimisé

Afin de trouver les aires extrêmes, nous avons supposé que le problème étendu aux polygones à trous est trop complexe pour une solution algorithmique autre qu'un algorithme de type "force brute" [1]. La visée est alors d'optimiser celui-ci en réduisant à un maximum l'ensemble des possibilités.

L'algorithme de base que nous optimiserons par la suite consiste à naïvement itérer toutes les permutations [2] de l'ensemble des points d'entrée, calculant pour chacune l'aire du polygone correspondant grâce à l'algorithme décrit précédemment et la comparant aux aires minimale et maximale obtenues jusqu'ici afin de les remplacer si celle-ci est plus extrême qu'une de deux aires respective.
Pour n points d'entrées, cet algorithme itère $1 \times 2 \times ... \times n = n!$ permutations, menant à une complexité temporelle de $\mathcal{O}(n!)$

Remarque. Pour souligner les optimisations de notre algorithme, nous considérons par la suite qu'une itération est effectuée en temps constant $\mathcal{O}(1)$ alors qu'elle est d'ordre $\mathcal{O}(n^3)$ par le calcul d'aire.

1. Algorithme qui itère sur tous les éléments d'un ensemble pour résoudre un problème, souvent inefficace
2. Les permutations d'un ensemble de trois éléments A, B et C sont : "ABC", "ACB", "BCA", "BAC", "CAB" et "CBA".

Nous proposons alors deux optimisations à effectuer : une première confondant les cyclisassions de ces permutations, et une deuxième confondant les miroirs de celles-ci.

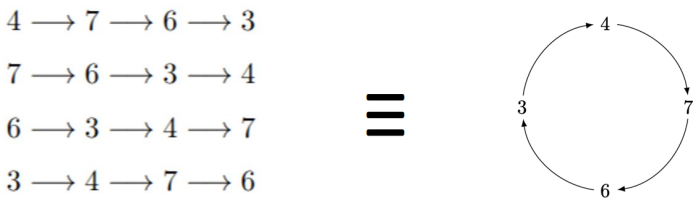

FIGURE 30 – Confusion des cyclisassions

Premièrement, nous remarquons que certaines de ces permutations désignent en réalité un même polygone par le fait qu'elles sont des cyclisassions de l'une l'autre. En effet, les permutations "$ABCD$", "$BCDA$", "$CDAB$" et "$DABC$" sont des cyclisassions de l'une l'autre et décrivent le même polygone. Nous cherchons alors à éliminer tout dupliqué en ignorant les cyclisassions d'une permutation donnée. Pour cela, nous évoquons l'ismorphisme [3] démontrée en annexe (voir Théorème 1) :

> *Considérer toutes les permutations commençant par un même élément fixe revient à considérer toutes le cyclisassions de celles-ci si on les considère équivalentes.*

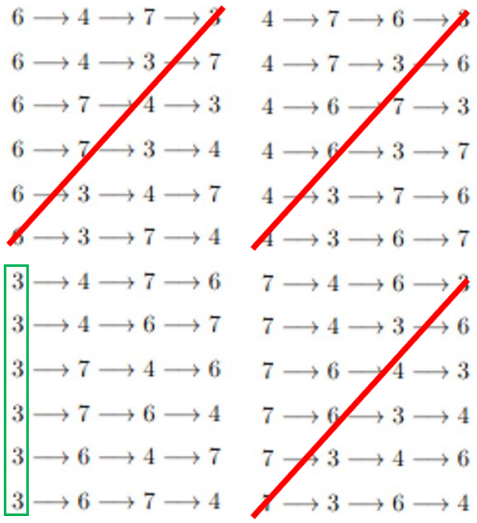

FIGURE 31 – Conséquence de l'isomorphisme

Ainsi, nous itérerons dorénavant seulement les permutations ayant un même point fixe pour premier élément, choisi arbitrairement. Ceci est effectué en construisant chaque permutation en joignant cet élément fixe à une "proto"-permutation du sous-ensemble où l'élément fixe est exclus. La complexité temporelle est alors de $\mathcal{O}((n-1)!) = \mathcal{O}(\frac{n!}{n})$.

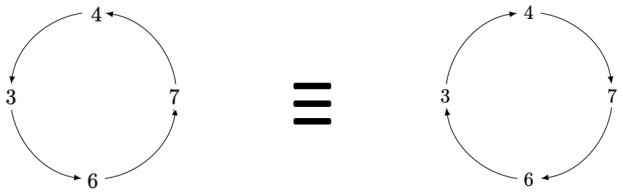

FIGURE 32 – Confusion des miroirs

3. Forme d'équivalence par association un-à-un

Finalement, nous remarquons également qu'une permutation et son miroir désignent le même polygone : "$ABCD$" et "$ADCB$" (équivaut à "$DCBA$" par cyclisassion) décrivent le même polygone. Afin d'éliminer ces dupliqués dans l'algorithme, nous ignorons les miroirs en considérant un ensemble des miroirs des permutations que nous mettons à jour à chaque itération, que nous utiliserons pour "sauter" une permutation si elle est dans cet ensemble en passant à l'itération suivante. La complexité, divisée en deux, est alors de $\mathcal{O}(\frac{n!}{2n})$.

Algorithm 1 Algorithme "force brute" avec cyclisassions et miroirs confondus $\mathcal{O}(\frac{n!}{2n})$

```
p ← {p_i}                                   ▷ Définition de l'ensemble des points d'entrées

P_min ← ∅                                   ▷ Initialisation des polygones et des aires extrêmes
P_max ← ∅
A_min ← +∞
A_max ← −∞

p_1 ← x : x ∈ p                             ▷ Définition de l'élément fixe, arbitraire
q ← p \ {p_1}                               ▷ Définition du sous-ensemble par exclusion

r ← {}                                      ▷ Initialisation de l'ensemble des miroirs, vide

for Q_i ∈ Sym(q) do [4]                     ▷ Itération sur toutes les proto-permutations
    P_i ← (p_1, Q_i^1, ..., Q_i^n)          ▷ Construction de la permutation par jonction
    if P_i ∈ r then
        continue                            ▷ "Sauter" si elle est le miroir d'une permutation déjà rencontrée
    end if
    A_i ← A(P_i)                            ▷ Calcul d'aire
    if A_i < A_min then                     ▷ Comparaison et substitution avec l'aire minimale
        A_min ← A_i
        P_min ← P_i
    end if
    if A_i > A_max then                     ▷ Comparaison et substitution avec l'aire maximale
        A_max ← A_i
        P_max ← P_i
    end if
    R_i ← (p_1, Q_i^n, ..., Q_i^1)          ▷ Construction du miroir
    r ← r ∪ {R_i}                           ▷ Ajout du miroir à l'ensemble r
end for
return (P_min, P_max)
```

4.4. Implémentation en Python

Afin d'évaluer les effets de chaque optimisation, nous avons implémenté l'algorithme décrit dans la section précédente en langage Python [3]. Nous avons utilisé une solution proposée en ligne [1] pour le calcul d'aires à trous par tranches verticales. Cette dernière est cependant optimisée pour la taille du script, et non pas pour sa complexité temporelle ou en mémoire. En conséquence, l'implémentation est considérablement ralentit, la complexité du calcul d'aires n'étant pas de l'ordre de celle proposé en section 4.2.

En outre, le calcul d'aires est limité aux polygones dont les points ont tous des abscisses différentes. En effet, nous avons trouvé que le script de calcul d'aire été basé sur l'algorithme Bentley-Ottman sous sa forme spécifique, qui est contraint à des points aux abscisses différents. Ceci peut être remédié en ré-implémentant le script en utilisant l'algorithme Bentley-Ottman généralisé par de Berg [4].

Néanmoins, le langage Python étant loin d'être le langage optimal pour les calculs rapides, nous rappelons que le but de cette implémentation est uniquement de mettre en évidence les rapports

4. Le groupe symétrique d'un ensemble S, noté $\mathrm{Sym}(S)$, est le groupe de toutes ses permutations.

entre les temps d'exécution de chaque génération d'optimisation. Les résultats que nous décrivons n'illustre donc pas l'efficacité d'une implémentation réellement optimisée.

Génération	Régulier		Quelconques	
d'optimisation	5 points	8 points	5 points	8 points
Base	53 ms	78.0 s	55 ms	85.9 s
Cyclisassions	14 ms	8.8 s	17 ms	9.7 s
Cyclisassions + Miroirs	6 ms	4.4 s	7 ms	4.4 s

TABLE 1 – Temps d'exécution

Dans le Tableau 1, nous présentons les temps d'exécution sur une machine locale générique, pris en moyenne sur 3 exécutions. Nous considérons les temps pour l'algorithme "force-brute" de base ainsi que les deux générations d'optimisation présenté en section 4.3, pour des pentagones et octogones réguliers et quelconques. L'ensemble des points d'entrée pour les polygones quelconques est construit en prélevant uniformément des points sur un espace carré.

5. Conclusion

La polygonisation d'aire optimale peut être utilisée pour générer des modèles numériques de terrain, ou pour diviser des zones géographiques.

Dans les industries telles que le bois, le métal ou le textile, où des matériaux sont découpés en formes spécifiques, la subdivision d'une feuille de matériau en polygones d'aire optimale peut aider à minimiser les déchets et à maximiser l'utilisation du matériau. Cela peut conduire à des économies de coûts significatives lors de la production en masse.

Références

[1] Anonymous. (2015) Area of a self-intersecting polygon. Stack Exchange (code golf). [Online]. Available : https://codegolf.stackexchange.com/questions/47638/area-of-a-self-intersecting-polygon/47640#47640

[2] Bentley and Ottmann, "Algorithms for reporting and counting geometric intersections," *IEEE Transactions on Computers*, vol. C-28, no. 9, pp. 643–647, 1979.

[3] A. Gana. (2024) Optimal area polygonisation : Brute-force in python. MEJ LVVG. [Online]. Available : https://replit.com/@EventualDesk/Polygones#main.py

[4] M. O. M. de Berg, M. van Kreveld and O. Schwarzkopf, *Computational Geometry*. Springer-Verlag, 2000, ch. Chapter 2 : Line segment intersection, pp. 19–44.

A. Annexe - Isomorphisme pour cyclisassions

A.1. Définitions et théorèmes suivants

Definition 1 (Permutation à élément fixe)**.** Une permutation à élément fixe d'un ensemble fini arbitrairement ordonné $S = \{s_0, s_1, \ldots, s_n\}$ est un élément du groupe symétrique de S fixant le premier élément. Nous notons l'ensemble de toutes les permutations à élément fixe de S comme $\mathfrak{F}(S)$.

$$\mathfrak{F}(S) = \{\varphi : s_0 \mapsto s_0\} \subseteq \mathrm{Sym}(S)$$

Théorème. $(\mathfrak{F}, \cdot)$ *forme un groupe sous la composition de fonctions.*

Démonstration. Vérifions que $(\mathfrak{F}, \cdot)$ satisfasse les axiomes définissant un groupe :

1. Fermeture. La composition de deux permutations quelconques qui associent s_0 à lui-même mène à une permutation qui fait de même :

   $$\forall \varphi, \varphi' \in \mathfrak{F} \quad \varphi \cdot \varphi' : s_0 \mapsto s_0$$

   Nous en déduisons alors que $\mathfrak{F}$ est fermé.
2. Associativité. Les permutations à élément fixe sont des éléments du groupe symétrique et sont alors, par extension, associatives sous la composition de fonctions.
3. Élément neutre. L'élément neutre de Sym associe par définition chaque élément à lui-même, ce qui inclut s_0. Nous avons alors que $e \in \mathfrak{F}$.
4. Élément inverse. L'inverse d'une permutation qui associe s_0 à lui-même est une permutation qui fait de même. Cet inverse est alors inclus dans $\mathfrak{F}$.

Ayant satisfait les axiomes d'un groupe, nous concluons que $(\mathfrak{F}, \cdot)$ forme un groupe sous la composition de fonctions. □

Théorème. $\mathfrak{F}$ *est un sous-groupe de* Sym.

Démonstration. L'ensemble $\mathfrak{F}$ est un sous-ensemble de Sym. De plus, son groupe correspondant partage avec Sym la composition de fonctions en tant que opération. Nous en concluons que $\mathfrak{F}$ est un sous-groupe de Sym. □

Corollaire. $\mathfrak{F}$ *et* Sym *partagent le même élément neutre, e.*

Definition 2 (Permutation cyclique directe)**.** La i-ème permutation cyclique directe d'un ensemble fini arbitrairement ordonné $S = \{s_0, s_1, \ldots, s_n\}$, notée σ^i, est un élément du groupe symétrique de S qui associe le n-ème élément au $n+i$-ème élément, recyclant grâce à l'indice congru modulo $|S|$. Nous notons l'ensemble de toutes les permutations cycliques directes de S comme $\mathfrak{C}(S)$.

$$\mathfrak{C}(S) = \{\sigma^i : s_n \mapsto s_{n+1 \bmod |S|}\} \subseteq \mathrm{Sym}(S)$$

Théorème. $(\mathfrak{C}, \cdot)$ *forme un groupe sous la composition de fonctions.*

Démonstration. Vérifions que $(\mathfrak{C}, \cdot)$ satisfasse les axiomes définissant un groupe :

1. Fermeture. Pour tous $\sigma^i, \sigma^j \in \mathfrak{C}$, leur composition de fonctions donne

   $$(\sigma^j \cdot \sigma^i) : s_n \mapsto s_{n+i} \mapsto s_{n+i+j}$$

   La composition $\sigma^j \cdot \sigma^i$ donne $\sigma^{i+j} \in \mathfrak{C}$, rendant ainsi $\mathfrak{C}$ fermé.
2. Associativité. Les permutations à élément fixe sont des éléments du groupe symétrique et sont alors, par extension, associatives sous la composition de fonctions.

3. Élément neutre. Pour tout $\sigma^i \in \mathfrak{C}$, nous avons

$$\sigma^i \cdot \sigma^0 = \sigma^{0+i} = \sigma^i = \sigma^{i+0} = \sigma^0 \cdot \sigma^i$$

   Ainsi, $\mathfrak{C}$ a σ^0 comme élément neutre.

4. Élément inverse. Pour tout $\sigma^i \in \mathfrak{C}$, nous avons

$$\begin{cases} \sigma^{-i} \cdot \sigma^i = \sigma^{i-i} = \sigma^0 \\ \sigma^i \cdot \sigma^{-i} = \sigma^{-i+i} = \sigma^0 \end{cases}$$

   Donc, l'inverse de tout σ^i dans $\mathfrak{C}$ est donné par σ^{-i}.

Ayant satisfait les axiomes d'un groupe, nous concluons que $(\mathfrak{C}, \cdot)$ forme un groupe sous la composition de fonctions. □

Théorème. $\mathfrak{C}$ *est un sous-groupe de* Sym.

Démonstration. L'ensemble $\mathfrak{C}$ est un sous-ensemble de Sym. De plus, son groupe correspondant partage avec Sym la composition de fonctions en tant que opération. Nous en concluons que $\mathfrak{C}$ est un sous-groupe de Sym. □

Corollaire. $\mathfrak{C}$ *et* Sym *partagent le même élément neutre. Nous notons que* $\sigma^0 = e$.

Théorème. $\mathfrak{C}$ *est un sous-groupe normal de* Sym.

Démonstration. Par définition, $\mathfrak{C}$ est un sous-groupe normal de Sym si et seulement si il est invariant sous la conjugaison par des membres de Sym. Autrement dit, $\mathfrak{C}$ est normal dans Sym si et seulement si $\pi^{-1} \cdot \sigma^i \cdot \pi \in \mathfrak{C}$, pour tout $\pi \in \mathfrak{C}$ et $\sigma^i \in \mathfrak{C}$.

Explicitions ces permutations en utilisant la notation de Cauchy :

$$\pi = \begin{pmatrix} s_0 & s_1 & \dots & s_n \\ \pi_0 & \pi_1 & \dots & \pi_n \end{pmatrix}, \sigma^i = \begin{pmatrix} s_0 & s_1 & \dots & s_n \\ s_i & s_{i+1} & \dots & s_{i+n} \end{pmatrix}$$

où les indices sont tous implicitement congrus modulo $|S|$. L'inverse de π est donné par

$$\pi^{-1} = \begin{pmatrix} \pi_0 & \pi_1 & \dots & \pi_n \\ s_0 & s_1 & \dots & s_n \end{pmatrix}$$

En opérant successivement $\pi^{-1} \cdot \sigma^i \cdot \pi = \pi^{-1}(\sigma^i(\pi))$, nous obtenons :

$$\begin{aligned} &\begin{pmatrix} \pi_0 & \pi_1 & \dots & \pi_n \\ s_0 & s_1 & \dots & s_n \end{pmatrix} \left[\begin{pmatrix} s_0 & s_1 & \dots & s_n \\ s_i & s_{i+1} & \dots & s_{i+n} \end{pmatrix} \begin{pmatrix} s_0 & s_1 & \dots & s_n \\ \pi_0 & \pi_1 & \dots & \pi_n \end{pmatrix} \right] \\ =& \begin{pmatrix} \pi_0 & \pi_1 & \dots & \pi_n \\ s_0 & s_1 & \dots & s_n \end{pmatrix} \begin{pmatrix} \pi_0 & \pi_1 & \dots & \pi_n \\ \pi_i & \pi_{i+1} & \dots & \pi_{i+n} \end{pmatrix} \\ =& \begin{pmatrix} s_0 & s_1 & \dots & s_n \\ s_i & s_{i+1} & \dots & s_{i+n} \end{pmatrix} = \sigma^i \end{aligned}$$

Ainsi, nous avons $\pi^{-1} \cdot \sigma^i \cdot \pi = \sigma^i \in \mathfrak{C}$, pour tout $\pi \in \mathfrak{C}$ et $\sigma^i \in \mathfrak{C}$, satisfaisant l'invariance sous la conjugaison. Nous concluons que $\mathfrak{C}$ est un sous-groupe normal de Sym. □

Corollaire. *Étant donné que* $\mathfrak{C}$ *est normal dans* Sym, *il existe le groupe quotient de* Sym $/\mathfrak{C}$, *soit le groupe de tous les cosets* $\pi \cdot \mathfrak{C} = \mathfrak{C} \cdot \pi$ *pour* $\pi \in$ Sym *sous la même opération.*

A.2. Démonstration de l'isomorphisme

Théorème 1. *Le groupe quotient* $\mathrm{Sym}/\mathfrak{C}$ *est isomorphe à* $\mathfrak{F}$*, noté* $\mathrm{Sym}/\mathfrak{C} \cong \mathfrak{F}$.

Démonstration. Soit $f : \mathfrak{F} \to \mathrm{Sym}/\mathfrak{C}$ une fonction qui associe chaque permutation à élément fixe à son coset par rapport à $\mathfrak{C}$, telle que $f : \varphi \mapsto \mathfrak{C} \cdot \varphi$ où $\mathfrak{C} \cdot \varphi$ est le coset de φ, parmi d'autres permutations. Pour démontrer l'isomorphisme, la fonction f doit être (1) bijective et (2) satisfaire la relation

$$\forall \varphi, \varphi' \in \mathfrak{F} \quad f(\varphi \cdot \varphi') = f(\varphi) \cdot f(\varphi')$$

Commençons par la bijectivité de la fonction. Nous devons montrer que chaque élément du codomaine $\mathrm{Sym}/\mathfrak{C}$ est l'image d'un seul élément du domaine $\mathfrak{F}$. Cela revient à montrer qu'il existe exactement un tel élément $\varphi \in \mathfrak{F}$ pour chaque $\mathfrak{C} \cdot \pi \in \mathrm{Sym}/\mathfrak{C}$ de sorte que $f(\varphi) = \mathfrak{C} \cdot \pi$. En substituant et en utilisant la définition du coset, pour chaque π, un φ unique doit satisfaire

$$f(\varphi) = \mathfrak{C} \cdot \varphi = \mathfrak{C} \cdot \pi \iff \{\sigma^i \cdot \varphi : \sigma^i \in \mathfrak{C}\} = \{\sigma^i \cdot \pi : \sigma^i \in \mathfrak{C}\}$$

En utilisant l'élément neutre e dans $\mathfrak{C}$, nous obtenons qu'il faut montrer qu'il existe, pour tout π, un φ unique tel que $\varphi \cdot e = \varphi \in \{\sigma^i \cdot \pi : \sigma^i \in \mathfrak{C}\}$, ou de manière équivalente

$$\forall \pi \, \exists! \varphi \, \exists \sigma^i \quad \varphi = \sigma^i \cdot \pi$$

Or, φ est seulement contraint à associer s_0 à lui-même. Alors, pour n'importe quel π, nous mettons σ^i égal à la permutation trivialement unique tel que $\sigma^i(\pi(s_0)) = s_0$, et définissons φ comme la solution conséquemment unique $\sigma^i \cdot \pi$, satisfaisant ainsi la bijectivité.

Pour montrer que f satisfait la deuxième relation, nous devons simplement évaluer $f(\varphi \cdot \varphi')$ pour obtenir $f(\varphi) \cdot f(\varphi')$ en utilisant la propriété des cosets sous la composition de groupe :

$$f(\varphi \cdot \varphi') = (\varphi \cdot \varphi') \cdot \mathfrak{C} = (\varphi \cdot \mathfrak{C}) \cdot (\varphi' \cdot \mathfrak{C}) = f(\varphi) \cdot f(\varphi')$$

Ayant satisfait les deux conditions, nous concluons que le groupe quotient $\mathrm{Sym}/\mathfrak{C}$ est isomorphe à $\mathfrak{F}$. □

Par ce dernier théorème, considérer toutes les permutations commençant par un même élément fixe revient à considérer toutes le cyclisassions de celles-ci si on les considère équivalentes.