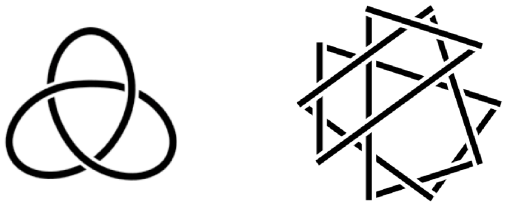
Nœuds et matroïdes orientés



Stage Master 2 Mathématiques Fondamentales

Université Montpellier 2

Mathieu VOLAND[1]

Dans le cadre du projet ANR : TEOMATRO (ANR-10-BLAN 0207)
Sous la supervision de **J.L. Ramírez-Alfonsín**[2]

Septembre 2013

1. mathieu.vol@gmail.com
2. jramirez@math.univ-montp2.fr

Table des matières

Remerciements

Je remercie M. Ramírez, qui m'a offert la possibilité d'effectuer ce stage dans le cadre du projet ANR : TEOMATRO (ANR-10-BLAN 0207), correspondant parfaitement à mes motivations dans le cadre de mon cursus personnalisé entre les mathématiques fondamentales et l'informatique.

Ce stage m'a permi de développer mon goût pour la recherche, et m'a permi de confirmer l'orientation de mon projet professionnel vers la recherche appliquée.

Je remercie particulièrement M. Ramírez pour sa grande disponibilité durant toute la durée du stage.

Je remercie également M. Giroudeau qui m'a orienté vers M. Ramírez en me suggérant d'étudier la théorie des matroïdes.

Première partie

Aspects théoriques

Chapitre 1

Présentation de la théorie des nœuds

Ce chapitre présente la théorie des nœuds. Il s'agit de définir les concepts principaux, rappeler certains résultats, et préciser le vocabulaire qui est utilisé dans le présent rapport.

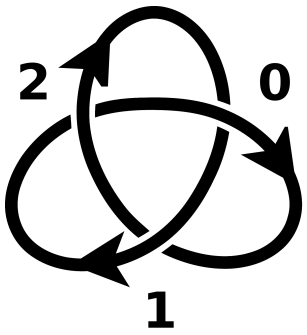


FIGURE 1.1 – Un diagramme du trèfle gauche

On définit dans un premier temps les nœuds et les diagrammes qui en constituent une modélisation pratique. On expose ensuite le principal résultat en lien avec ces diagrammes et les mouvements de Reidemeister. On donne enfin un bref historique de l'étude de la théorie des nœuds, en particulier concernant les motivations récentes issues de l'étude de l'ADN.

1.1 Nœuds et modélisation

Définition 1.1.1 (Nœud)**.**
Un nœud est un plongement du cercle $\mathcal{S}^1$ dans l'espace euclidien $\mathbb{R}^3$.

On appelle nœud le plongement $h : \mathcal{S}^1 \to \mathbb{R}^3$ lui même, ou simplement l'image $h(\mathcal{S}^1) \subset \mathbb{R}^3$ de celui-ci. On dit que deux nœuds sont *isotopes* s'il existe une isotopie de l'espace ambiant sur lui même telle que le second est l'image du premier.

Définition 1.1.2 (Isotopie - nœuds isotopes)**.**
Une isotopie de l'espace ambiant est une application continue $F : \mathbb{R}^3 \times [0,1] \to \mathbb{R}^3$ telle que :
– *$F(\cdot, 0) : \mathbb{R}^3 \to \mathbb{R}^3 = \mathrm{Id}_{R^3}$*
– *pour tout $t \in [0,1]$ l'application $F(\cdot, t) : \mathbb{R}^3 \to \mathbb{R}^3$ est un homéomorphisme.*
Si $K_1 = h_1(\mathcal{S}^1)$ et $K_2 = h_2(\mathcal{S}^1)$ sont deux nœuds, on dit qu'ils sont isotpes s'il existe une isotopie $F : \mathbb{R}^3 \times [0,1] \to \mathbb{R}^3$ vérifiant : $F(\cdot, 1) \circ h_1 = h_2$

Cette définition coïncide avec la représentation intuitive que l'on peut se faire d'un nœud sur une ficelle : une isotopie est une déformation de la ficelle réalisable sans "couper-recoller" la ficelle.

Définition 1.1.3 (Nœuds mirroirs)**.**
Deux nœuds sont dits mirroirs si le premier est isotope au symétrique (par rapport à un plan) du second.

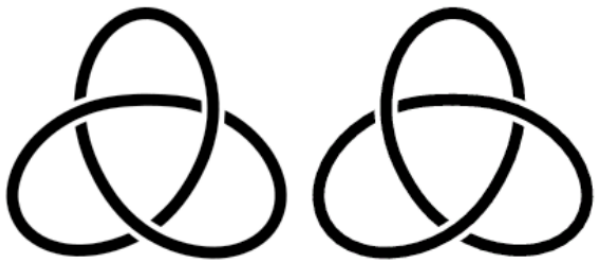


FIGURE 1.2 – Les trèfles droit et gauche (mirroirs)

Dans le cas général, deux nœuds *mirroirs* ne sont pas isotopes. On peut par exemple montrer que les trèfles droit et gauche (figure 1.2) qui sont le mirroir l'un de l'autre, ne sont pas isotopes.

Afin de représenter un nœud il est plus simple de travailler avec un *diagramme*. Il s'agit d'une projection du nœud sur l'espace euclidien $\mathbb{R}^2$ qui soit injective sauf en un nombre fini de points pour lequels il existe exactement deux antécédants. En ces points, appelés *croisements* on conserve l'information des altitudes relatives des points projetés.

Définition 1.1.4 (Diagramme)**.**
Un diagramme d'un nœud est une projection régulière d'un nœud sur un plan, c'est à dire une projection injective à l'exception d'un nombre fini de points. On conserve en ces points la position relative des deux arcs qui se croisent.

En fixant un sens de parcours du nœud on peut définir une notion d'*orientation* à chaque croisement comme dans la figure 1.3. On remarque que la notion d'orientation ne dépend que de la nature du croisement et non de l'orientation globale du nœud qui peut donc être définie de façon arbitraire.

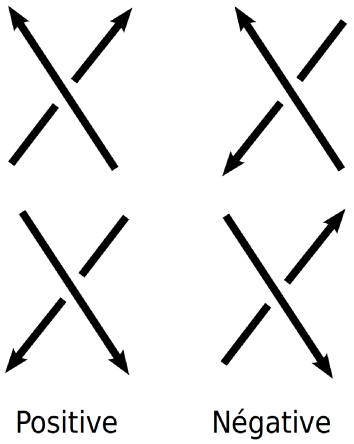


FIGURE 1.3 – Orientation des croisements

On appelle ici *diagramme d'un nœud* la donnée d'un graphe planaire 4-régulier, orienté, et dont les croisements sont orientés.

On peut montrer que tout plongement admet une projection régulière qui induit donc un diagramme de nœud, et réciproquement il est clair que tout diagramme induit un plongement d'un nœud dans $\mathbb{R}^3$. On dit également que deux diagrammes sont *équivalents* si les nœuds induits dans $\mathbb{R}^3$ sont isotopes.

Un nœud a plusieurs plongements différents dans $\mathbb{R}^3$, et de même chaque plongement induit plusieurs projections régulières. Ainsi un nœud donné peut être représenté par une classe de diagrammes qui peuvent paraître sans aucun lien (par exemple avec des nombres de croisements différents).

Bien que la question de savoir si deux diagrammes quelconques sont équivalents reste ouverte, nous verrons que les mouvements de Reidemeister permettent de restreindre cette étude à une suite de mouvements élémentaires.

1.2 Code de Gauss

On considère un diagramme d'un nœud avec n croisements. On va définir le code de Gauss de ce nœud comme une suite de n couples (signe, numéro).

- On fixe un croisement initial et un sens de parcours.
- On suit le nœud dans le sens fixé.
- On rencontre tous les croisements exactement deux fois. A chaque fois on associe un couple (signe, numéro).
- La première fois on donne le premier entier non-encore utilisé, et on donne un signe positif si l'arc passe au dessus, et négatif s'il passe en dessous.
- La deuxième fois le numéro du croisement à déjà été fixé et la position sera toujours l'opposée celle observée au premier passage. On donne donc un signe représentant l'orientation du nœud.

Le fait d'encoder l'orientation de chaque croisement est importante, en effet une simple suite de croisement et altitude relative ne permet pas de fixer un unique nœud.

La donnée d'un code de Gauss détermine un nœud. Cependant le diagramme exact n'est pas défini (choix de la face infinie). Par ailleurs toute suite de couple (signe, numéro) ou chaque

numéro apparait deux foix ne correspond pas forcément au code de Gauss d'un nœud : il peut y avoir des croisements virtuels [1].

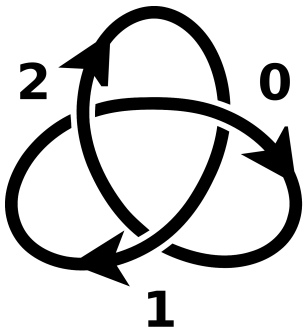


FIGURE 1.4 – Code de Gauss du trèfle gauche : +0, -1, +2, +0, +1, +2

1.3 Mouvements de Reidemeister

On définit trois types de mouvements, dits de Reidemeister, sur les diagrammes.

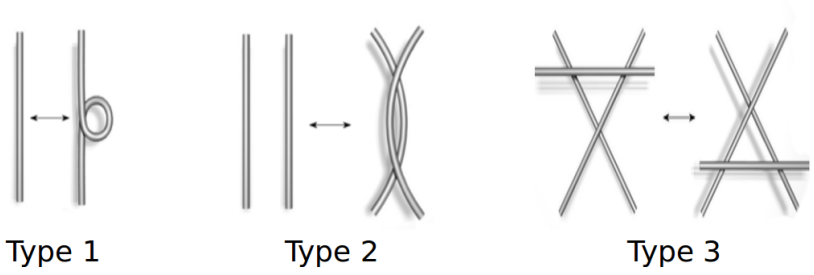


FIGURE 1.5 – Mouvements de Reidemeister

En réalité chacun des trois mouvements fait référence à 2 mouvements mirroirs. Dans la suite, toute mention du mouvement de type N pourra faire référence au mouvement présenté au dessus, ou bien à son mirroir.

Théorème 1.3.1 (Reidemeister [2])**.**
Deux diagrammes sont équivalents si et seulement si on peut obtenir l'un d'entre eux en modifiant l'autre part une série de mouvements de Reidemeister.

Il est assez clair qu'en appliquant un mouvement de Reidemeister, le diagramme obtenu est équivalent, l'intérêt de ce théorème réside donc dans sa réciproque.

Bien que ce théorème permette de réduire l'étude de l'équivalence de deux diagrammes à trois mouvements simples, il n'existe à ce jour aucun algorithme efficace (polynomial en le nombre de croisements) qui permette de tester si deux diagrammes sont équivalent. De même, étant donné un diagramme, on ne sait pas s'il existe un algorithme efficace permettant de décider s'il est trivial ou non.

1. L. H. Kauffman a développé une théorie des nœuds virtuels, avec des croisements virtuels et de mouvements de Reidemeister étendus.
2. K. Reidemeister, Homotopieringe und Linsenräume (1936) : ce théorème s'applique au cadre plus général des entrelacs (ensemble fini de nœuds).

1.4 Histoire et motivations

Les premiers articles mathématiques traitant de la théorie des nœuds datent du début du XIX^{e} siècle. On peut noter en particulier les travaux de C. F. Gauss sur les entrelacs (*links*) et leur enlacement (*linking number*). Il s'agit d'étudier un invariant sur la manière dont plusieurs courbes fermés peuvent être liées entre elles.

FIGURE 1.6 – Un entrelac : Hopf link

Au cours du XIX^{e}siècle, la notion de nœud est apparue dans certaines théories physiques traitant de la nature des atomes et des molécules. Ces théories visaient à expliquer notamment la nature discontinue du spectre d'émissions des différents éléments chimique. [3]

Par exemple, W. Thompson (Lord Kelvin) explique les deux bandes du spectre d'émission du sodium par l'entrelac "Hopf link" donc l'enlacement vaut 2.

Au cours du XX^{e} siècle les nœuds ont été étudiés du point de vue topologique, en particulier en lien avec les variétés de dimension 3 (M. Dehn, J. W. Alexander, K. Reidemeister).

Plus récemment, l'étude des nœuds est de nouveau motivée par une application dans un autre domaine scientifique. Les brins d'ADN peuvent en effet former des nœuds, lesquels sont modifiés par certains enzymes. Il est apparu que la nature du nœud formé a des conséquences sur les propriétés chimiques de l'ADN. Il est donc devenu important de comprendre les nœuds dans l'objectif des fournir des outils pratiques (des algorithmes efficaces) pour classifier les nœuds.

Ces algorithmes n'existent pas encore, ni la preuve de la NP-complétude ou non du problème. Le présent stage a pour objectif d'étudier un nouveau point de vu sur les nœuds, issus de la théorie des matroïdes orientés, afin d'étudier le comportement de certains algorithmes et d'aider à conjecturer de nouvelles solutions.

3. P. G. Tait, Scientific papers, Cambridge University Press (1898-1900)

Chapitre 2

Présentation de la théorie des matroïdes orientés

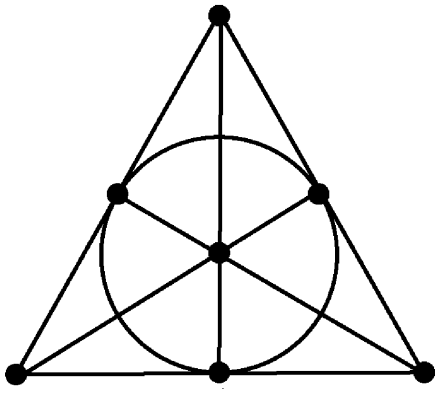


FIGURE 2.1 – Une représentation du matroïde de Fano

Ce chapitre introduit la théorie des matroïdes orientés. Les définitions et résultats utiles pour la suite sont présentés, sans les preuves.

L'ouvrage ayant servi de référence au cours de ce stage est : "Oriented Matroids" par A. Björner, M. Las Vergnas, B. Sturmfels, N. White, G. M. Ziegler (Cambridge University Press, second edition, 1999).

On donne tout d'abord une définition succinte des matroïdes non-orientés, avant de définir les matroïdes orientés. On étudie plus particulièrement les matroïdes orientés des dépendances affines qui apportent des résultats utiles pour les polytopes cycliques qui seront également définis.

2.1 Matroïdes non-orientés

2.1.1 Définitions

Définition 2.1.1 (Matroïde)**.**

Un matroïde $M = (E, \mathcal{I})$ est la donnée d'un ensemble fini E et d'une famille $\mathcal{I} \subset \mathcal{P}(E)$ de parties de E vérifiant :

1. $\emptyset \in \mathcal{I}$
2. $\forall J \in \mathcal{I},\ \forall I \subset J,\ I \in \mathcal{I}$
3. $\forall I, J \in \mathcal{I}$ *tels que* $|I| < |J|,\ \exists e \in J,\ I \cup \{e\} \in \mathcal{I}$

On dit que $\mathcal{I}$ est l'ensemble des indépendants du matroïde.

On peut alors définir les *bases* du matroïde comme les indépendants maximaux, et on montre facilement qu'elles ont toutes le même cardinal appelé *rang* du matroïde. On dit que les parties de E qui ne sont pas des indépendants sont des *dépendants* et on définit l'ensemble des *circuits* du matroïde comme les dépendants minimaux (on note que les circuits n'ont généralement pas tous le même cardinal).

Proposition 2.1.1 (Circuits)**.**

L'ensemble $\mathcal{C}$ des circuits d'un matroïde vérifie :

1. $\emptyset \notin \mathcal{C}$
2. $\forall C_1, C_2 \in \mathcal{C},\ C_1 \subset C_2 \Rightarrow C_1 = C_2$
3. *Si* $C_1 \neq C_2 \in \mathcal{C},\ e \in C_1 \cap \mathcal{C}_2$ *alors* $\exists C_3 \in \mathcal{C}$ *tel que* $C_3 \subset (C_1 \cup C_2) \setminus \{e\}$

Proposition 2.1.2 (Bases)**.**

L'ensemble $\mathcal{B}$ des bases d'un matroïde vérifie :

1. $\mathcal{B} \neq \emptyset$
2. $\forall B_1, B_2 \in \mathcal{B},\ \forall x \in B_1 \setminus B_2,\ \exists y \in B_2 \setminus B_1,\ (B_1 \setminus \{x\}) \cup \{y\} \in \mathcal{B}$

De plus si $\mathcal{C} \subset \mathcal{P}(E)$ ou $\mathcal{B} \subset \mathcal{P}(E)$ vérifient les axiomes des propriétés précédantes, alors les ensembles suivants constituent les indépendants d'un matroïde sur E :

- $\mathcal{I} = \{I \subset B : B \in \mathcal{B}\}$
- $\mathcal{I} = \{I \subset E : \forall X \subset I,\ X \notin \mathcal{C}\}$

Cela fournit ainsi trois définitions équivalentes d'un matroïde sur un ensemble fini E.

2.1.2 Exemples

Matroïde graphique

On considère $G = (V, E)$ un graphe non-orienté. On peut définir le matroïde associé sur E, l'ensemble des arêtes, de deux manières équivalentes :

- les indépendants sont les forêts
- les circuits sont les cycles élémentaires (minimaux pour l'inclusion)

Matroïde des dépendances linéaires

On considère V un espace vectoriel réel, et $E = \{v_1, v_2, \ldots, v_n\}$ un ensemble fini de vecteurs. Alors $\mathcal{I}$ défini comme l'ensemble des familles de vecteurs de E indépendantes dans V constitue les indépendants d'un matroïde sur E. Par ailleurs les bases de ce matroïde sont exactement les familles de vecteurs de E qui constituent une base de $\mathrm{Vect}(E)$.

Matroïde des dépendances affines

On considère encore V un espace vectoriel réel, et $E = \{v_1, v_2, \ldots, v_n\}$ un ensemble fini de vecteurs de V. On définit alors les familles *affinement* dépendantes, on dit que $\{v_{i_1}, v_{i_2}, \ldots, v_{i_k}\}$ sont affinement dépendants si :

$$\text{il existe } \lambda_1, \lambda_2, \ldots, \lambda_k \in \mathbb{R} \text{ non tous nuls tels que } \sum_{j=0}^{k} \lambda_j v_{i_j} = 0 \text{ et } \sum_{j=0}^{k} \lambda_j = 0$$

Cela revient à dire que les vecteurs $\{v_{i_1} \times \{1\}, v_{i_2} \times \{1\}, \ldots, v_{i_n} \times \{1\}\}$ sont linéairement dépendants dans $V \times \mathbb{R}$.

On définit alors $\mathcal{I}$ l'ensemble des parties de E dont les vecteurs ne sont pas affinement dépendant et on peut alors montrer que $(E, \mathcal{I})$ est un matroïde.

2.2 Orientation, cas du matroïdes des dépendances affines

2.2.1 Définitions

Définition 2.2.1 (Partie signée)**.**

On considère un ensemble fini E. Une partie signée X de E est la donnée d'une partie $\underline{X}$ de E et d'une partition (X^+, X^-) de X. On dit que $\underline{X}$ est le support de X, et X^+ (resp. X^-) est la partie positive (resp. négative) de X.

On définit également $-X$ par $\underline{-X} = \underline{X}$ et $((-X)^+, (-X^-)) = (X^-, X^+)$.
Pour $\mathcal{E} \subset \mathcal{P}(E)$ on note $-\mathcal{E} = \{-X : X \in \mathcal{E}\}$.
Et on note $X = \{x_1, x_2, \ldots, x_k, \overline{y_1}, \overline{y_2}, \ldots, \overline{y_l}\}$ si $\{x_i\}_{i=1}^k = X^+$ et $\{y_j\}_{j=1}^l = X^-$.

Définition 2.2.2 (Matroïde orienté)**.**

Une famille $\mathcal{C}$ de parties signées d'ensemble fini E consistue les circuits d'un matroïde orienté si :

1. $\emptyset \notin \mathcal{C}$
2. $\mathcal{C} = -\mathcal{C}$
3. $\forall X, Y \in \mathcal{C}$, *si* $\underline{X} \subset \underline{Y}$, *alors* $X = Y$ *ou* $X = -Y$
4. $\forall X \neq Y \in \mathcal{C}, \forall e \in X^+ \cap Y^-$, *il existe* $Z \in \mathcal{C}$ *vérifiant :* $Z^+ \subset (X^+ \cup Y^+) \setminus \{e\}$ *et* $Z^- \subset (X^- \cup Y^-) \setminus \{e\}$

On remarque qu'en "oubliant" les signes, c'est à dire en ne considérant que les supports, on obtient les axiomes d'un matroïde non orienté. En particulier tout matroïde orienté induit un matroïde non orienté.

2.2.2 Matroïde orienté des dépendances affines

On considère de nouveau un ensemble fini E de vecteurs d'un espace vectoriel réel V. On considère uniquement les relations de dépendance affines minimales, en particulier tous les λ_i doivent être non nuls et sont fixés de manière unique à une constante multiplicative près. Ainsi à chaque relation de dépendance minimale on peut associer deux ensembles signés opposés : il s'agit des vecteurs intervenant dans la relation de dépendance partitionnés suivant le signe du coefficiant. On donne une définition plus précise :

On considère $\underline{\mathcal{C}}$ les circuits du matroïde non-orienté défini plus haut. On pose $\mathcal{C}$ l'ensemble des parties signées X de E telles que $\underline{X} \in \underline{\mathcal{C}}$ et telles qu'il existe une relation :

$$\sum_{x^+ \in X^+} \lambda_{x^+} \cdot x^+ + \sum_{x^- \in X^-} \lambda_{x^-} \cdot x^- = 0 \text{ avec } \sum_{x \in \underline{X}} \lambda_x = 0$$
$$\text{et } \forall x^+ \in X^+, \lambda_{x^+} > 0 \text{ et } \forall x^- \in X^-, \lambda_{x^-} < 0$$

On peut alors montrer que $\mathcal{C}$ ainsi défini forme les circuits d'un matroïde orienté sur E.

2.3 Matroïdes, partition de Radon, et polytopes cycliques

2.3.1 Dépendances affines sur le polytope cyclique

On se place dans l'espace euclidien $\mathbb{R}^d$ et on définit la courbe du moment $\gamma_d : \mathbb{R}_+ \to \mathbb{R}^d$ par $\gamma_d(t) = (t, t^2, \ldots, t^d)$.

Définition 2.3.1 (Polytope cyclique)**.**
On définit le polytope cyclique $C(d, n)$ comme l'enveloppe convexe de n points distincts sur la courbe du moment γ_d dans $\mathbb{R}^d$.

Dans la suite, on considère le cas où $d = 3$ et $\gamma = \gamma_3 : \mathbb{R}_+ \to \mathbb{R}^3$. On note alors $C(n) = \text{conv}\{\gamma(t_1), \gamma(t_2), \ldots, \gamma(t_n) : t_1 < t_2 < \cdots < t_n\}$. Les propriétés combinatoires du polytope ne dépendent en effet que des position relatives des points $\gamma(t_i)$ que l'on note x_i.

On considère alors le matroïde orienté des dépendances affines sur les sommets x_i du polytope cyclique $C(n)$. On peut montrer que les circuits du matroïde sont exactement les ensembles suivants (et leurs opposés) :

$$\{x_{i_1}, \overline{x_{i_2}}, x_{i_3}, \overline{x_{i_4}}, x_{i_5}\} \text{ avec } i_1 < i_2 < i_3 < i_4 < i_5$$

2.3.2 Partition de Radon

Définition 2.3.2 (Partition de Radon)**.**
Une partition de Radon d'un ensemble P de points d'un espace convexe est un partition $P = P_1 \sqcup P_2$ telle que $\text{conv}(P_1) \cap \text{conv}(P_2) \neq \emptyset$.

Dans le cas du polytope cyclique $C(n)$ les points se trouvant en position générale, les ensemble de 4 points ne contiennent pas de partition de Radon. En revanche tous les ensembles de 5 points contiennent une partition de Radon, en effet tous les ensembles de cardinal supérieur à $d + 2$ (où d est la dimension de l'espace, ici 3) contiennent une partition de Radon.

Il se trouve que les partitions de Radon des circuits du matroïde sont exactement les partitions formées par les signes des éléments. C'est à dire que si on considère 5 points distintcs $\{x_1, x_2, x_3, x_4, x_5\}$ on a l'équivalence suivante :

$$\{x_1, \overline{x_2}, x_3, \overline{x_4}, x_5\} \in \mathcal{C} \iff \text{conv}(\{x_1, x_3, x_5\}) \cap \text{conv}(\{x_2, x_4\}) \neq \emptyset$$

2.3.3 Projection du polytope cyclique et propriété du croisement

On projette le polytope cyclique sur le plan (x, y). La courbe du moment γ devient la courbe paramétrée $\{(t^2, t^3) : t \geq 0\}$ et les arêtes du polytope cyclique sont des segments entre points de la courbe. On identife dans la suite $x \in \{0, 1, \ldots, n-1\}$ et le point $\gamma(t_x)$.

Proposition 2.3.1 (Propriété du croisement). [1]

On se donne deux arêtes $\{x, y\}$ *et* $\{t, u\}$*. Alors* $\{t, u\}$ *passe au dessus de* $\{x, y\}$ *si et seulement si* $0 \leq x < t < y < u \leq (n-1)$.

Ce résultat peut se montrer en ajoutant un point z sur la gamma, et en vérifiant que l'arête $\{t, u\}$ intersecte le triangle $\{x, y, z\}$ (voir figure 2.2).

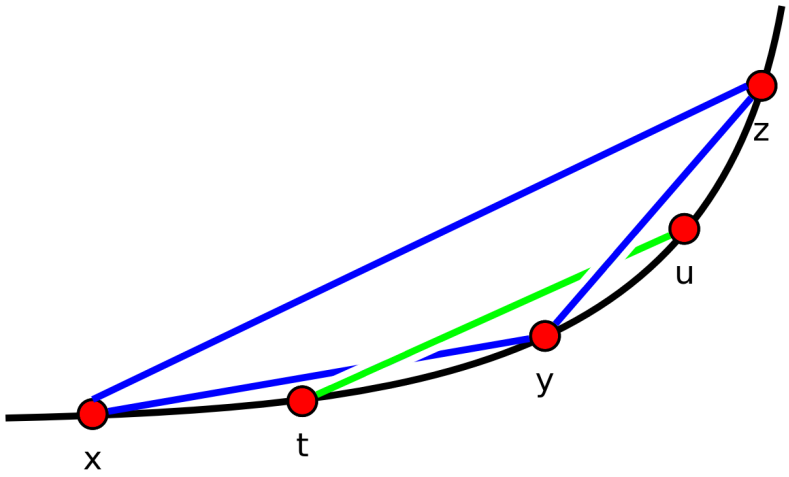


FIGURE 2.2 – La propriété du croisement

On peut alors représenter un polytope cyclique, ou une partie de ses arêtes de la façon suivante :

- On déforme la projection de la courbe du moment pour qu'elle soit sur l'axe horizontal.
- On représente les arêtes par des arcs de cercle entre les abscisses correspondantes.
- On vérifie que les croisements respectent la propriété du croisement.

On obtient ainsi un diagramme équivalent à la structure du polytope cyclique en trois dimension (voir figure 2.3).

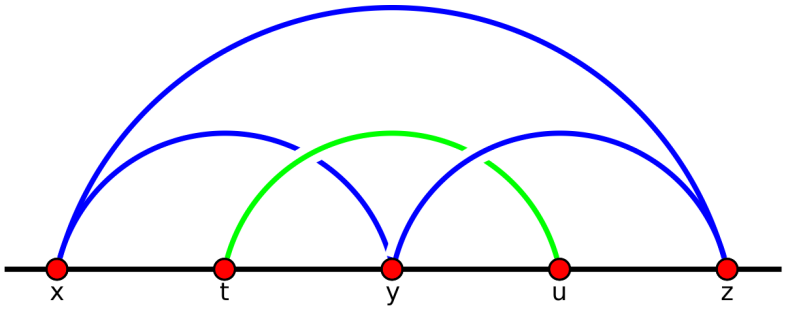


FIGURE 2.3 – Diagramme du polytope cyclique

1. J. L. Ramírez-Alfonsín, *Spatial Graphs, Knots and the Cyclic Polytope*

Chapitre 3

Le nœud sur le polytope cyclique

Les deux chapitres précédants avaient pour but d'énoncer les pré-requis, et de fixer le vocabulaire pour la suite du rapport. Ce chapitre concerne plus directement les travaux en lien avec le stage. Il s'agit d'étudier un nœud que l'on place "sur le polytope cyclique".

L'algorithme qui permet de passer d'un diagramme de nœud classique à une représentation sur le polytope cyclique sera décrit en détails dans la partie relative à son implémentation (partie 5.2).

On étudie en revanche dans ce chapitre les différentes propriétés et transformations définies sur le nœud lorsqu'il est placé sur le polytope cyclique.

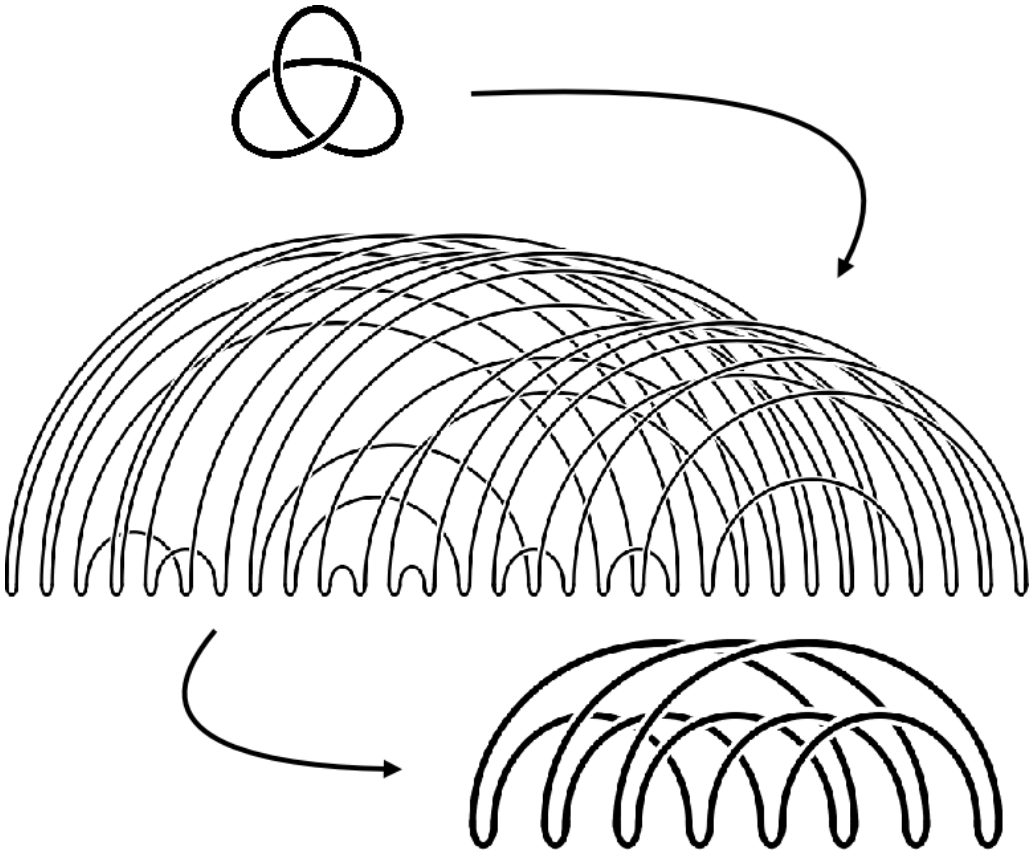


FIGURE 3.1 – Trèfle gauche : du diagramme classique au polytope cyclique réduit

3.1 Du nœud au polytope cyclique

On considère un nœud donné un diagramme avec n croisements. On peut montrer qu'il est possible de fournir un diagramme contenant uniquement des arêtes du polytope cyclique ($C(11n)$ [1]) qui est équivalent à ce nœud. Cet algorithme est décrit dans l'article *Spatails Graphs, Knots and the Cyclic Polytope* (J. L. Ramírez-Alfonsín), et une description précise est fournie avec les explications relatives à l'implémentation de celui-ci (voir la partie 5.2).

Une transformation du diagramme sur le polytope cyclique consiste à effectuer une opération sur le diagramme dont le résultat est encore un diagramme sur le polytope cyclique (respectant la propriété du croisement) du même nœud. L'intérêt est en général d'obtenir un nouveau digramme avec moins de sommets (sur gamma), ou moins de croisements (entre les arcs).

3.2 Rotation

La rotation droite consiste à prendre le dernier sommet pour le place en premier. Les arcs partant du dernier sommet passaient au dessus de tous les autres, ils passent ensuite en dessous de tous les autres. Cette opération conserve assez clairement le type du nœud (voir figure 3.2).

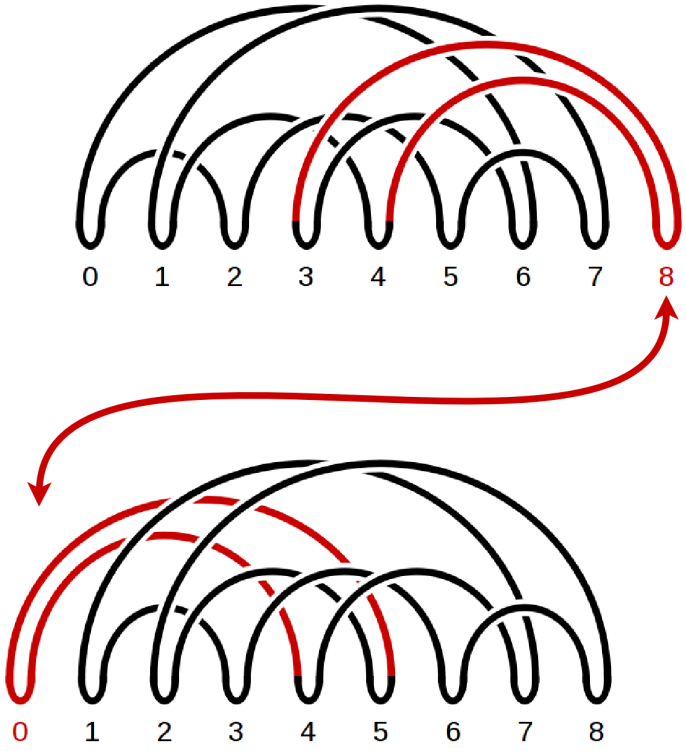


FIGURE 3.2 – Rotation

1. On peut montrer que $9n$ sommets suffisent, mais l'algorithme qui sera utilisé ajoute quelques sommets pour faciliter la représentation dans une structure informatique

3.3 Zones

On se donne un sommet x du diagramme sur le polytope cyclique. Il est toujours relié à deux autres sommets distincts y, z. On définit alors une partition des autres sommets en trois zones.

Plus précisément, en numérotant les sommets de 0 à $n-1$ on peut définir le sommet à droite de x comme $x+1 \mod n$ et le sommet à gauche de x comme $x-1 \mod n$. On peut alors définir z le voisin droit de x, c'est à dire le premier des deux autres sommets auquels il est relié que l'on rencontre en se déplaçant vers la droite. On définit de même y le voisin gauche de x.

Définition 3.3.1 (Sommet)**.**
Avec les notations précédantes on dit que x est un sommet du nœud sur le polytope cyclique, que l'on note (y, x, z).

Définition 3.3.2 (Intervalle)**.**
Pour a, b des sommets, on définit l'intervalle $[a, b]$ (resp. $]a, b[$) comme étant l'ensemble des sommets à droite (resp. strictement à droite) de a et à gauche (resp. strictement à gauche) de b.
On remarque en particulier que $]a, b[\cap]b, a[= \emptyset$.

Définition 3.3.3 (Zones)**.**
Soit (y, x, z) un sommet. On définit :
- *La zone 1 relativement à x comme l'intervalle $]y, x[$.*
- *La zone 2 relativement à x comme l'intervalle $]x, z[$.*
- *La zone 0 relativement à x comme l'intervalle $]z, y[$.*

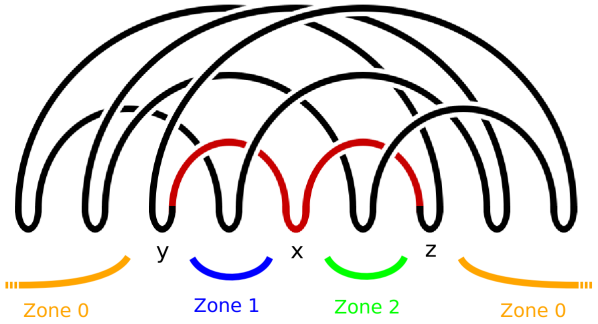


FIGURE 3.3 – Les trois zones relatives à un sommet

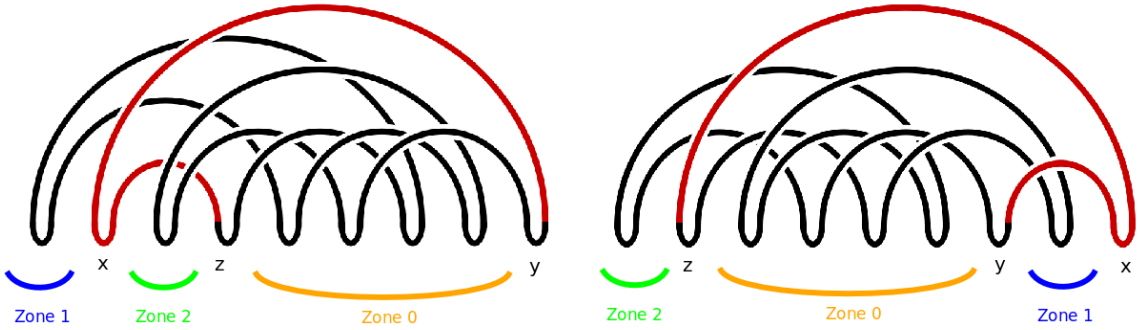


FIGURE 3.4 – Les trois zones après différentes rotations

3.4 Réduction

On considère un sommet (y, x, z). L'opération de réduction du sommet x consiste à supprimer ce sommet et remplacer les arc $\{y, x\}$ et $\{x, z\}$ par un arc $\{y, z\}$ (voir figure 3.5).

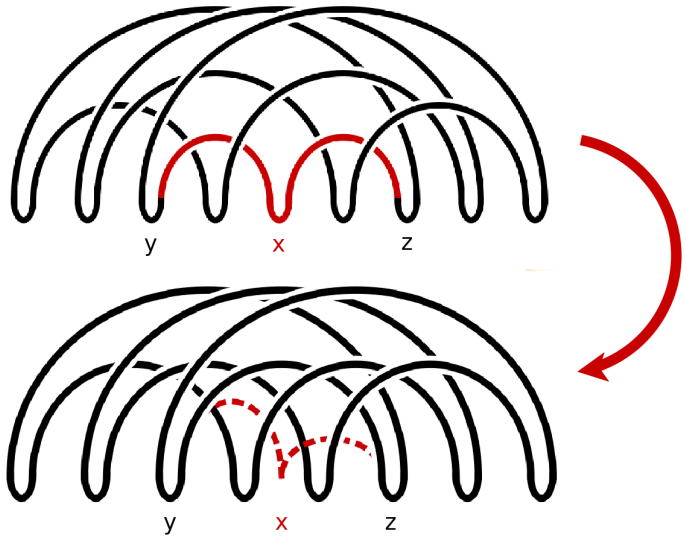


FIGURE 3.5 – Exemple de réduction

Proposition 3.4.1 (Sommet réductible)**.**

Avec les notations précédantes, si un sommet x vérifie l'une des trois conditions suivantes alors il est possible de le réduire sans changer le type du nœud représenté :

1. *Il n'y a aucun arcs entre la zone 0 et la zone 1.*
2. *Il n'y a aucun arcs entre la zone 1 et la zone 2.*
3. *Il n'y a aucun arcs entre la zone 2 et la zone 0.*

On dit alors que le sommet x est réductible*.*

Démonstration. Quitte à effectuer des rotations, on peut toujours se ramener au cas décrit dans la figure 3.6 : si aucun arc ne traverse le triangle, il est possible de déformer celui-ci en supprimant l'un des sommets. □

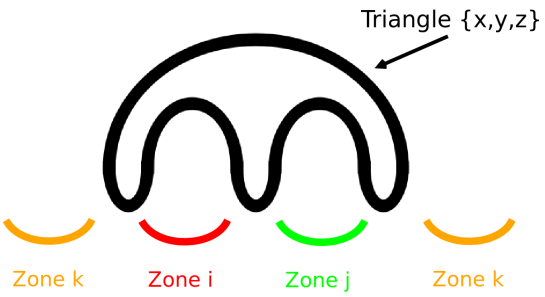


FIGURE 3.6 – Zones et triangle

On peut remarquer qu'il ne s'agit pas d'une équivalence, il est donc peut être possible d'affaiblir la condition pour détecter plus de sommets réductibles.

3.5 Echange

On considère deux sommets (x_1, x, x_2) et (y_1, y, y_2) tels que y soit le sommet immédiatement à droite de x (c'est à dire $y = x + 1 \mod n$). On dit qu'on échange x et y si on remplace les arcs $\{x, x_i\}$ par $\{y, x_i\}$ et les arc $\{y, y_i\}$ par $\{x, y_i\}$.

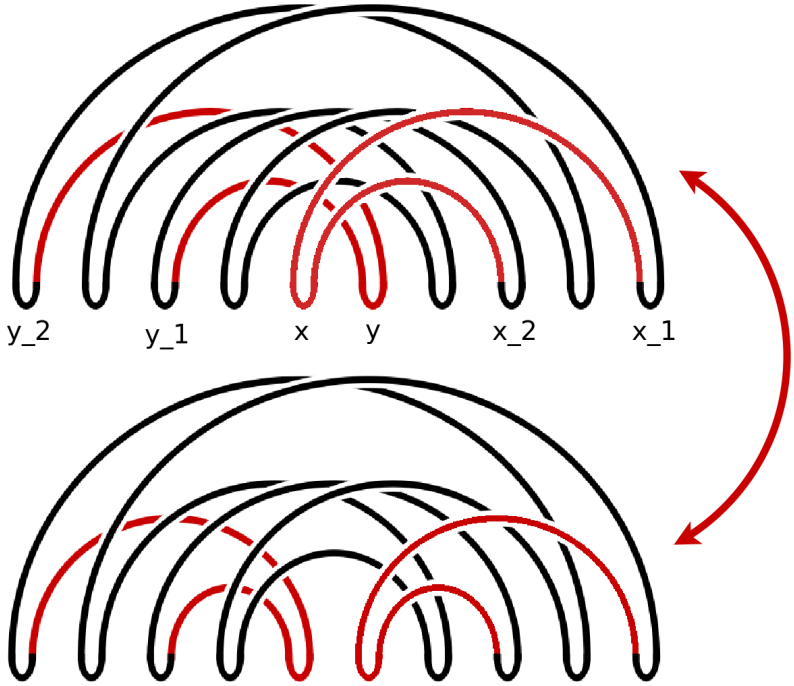


FIGURE 3.7 – Exemple d'échange

Proposition 3.5.1 (Sommets échangeables)**.**

Avec les notations précédantes, les sommets x et y vérifient l'une des trois conditions suivantes alors il est possible de les échanger sans changer le type du nœud représenté :

1. *Il y a un arc entre x et y (c'est à dire $x_2 = y$ et $y_1 = x$).*
2. *Ni y_1, ni y_2 ne situe dans la zone 0 relativement à x.*
3. *y_1 et y_2 se situent tous les deux dans la zone 0.*

On dit alors que les sommets x et y sont échangeables.

Démonstration. Il suffit de vérifier dans chacun des trois cas que les déplacements des quatres arcs en jeu peuvent être décomposés en mouvements de Reidemeister. □

Un échange ne réduit pas le nombre de croisements, cependant il peut faire apparaitre une réduction ou bien réduire strictement le nombre de croisements. Dans ces cas on dit qu'il s'agit d'un "bon échange" (comme dans la figure 3.7 où le nombre de croisements est réduit de 4).

3.6 Insertion

On considère un arc $\{y, z\}$, en supposant $y < z$. Insérer x entre y et z signifie ajouter un sommet entre x et le suivant, et remplacer l'arc $\{y, z\}$ par deux arcs $\{y, x\}$ et $\{x, z\}$. Attention, les numéros de tous les sommets après x sont modifiés.

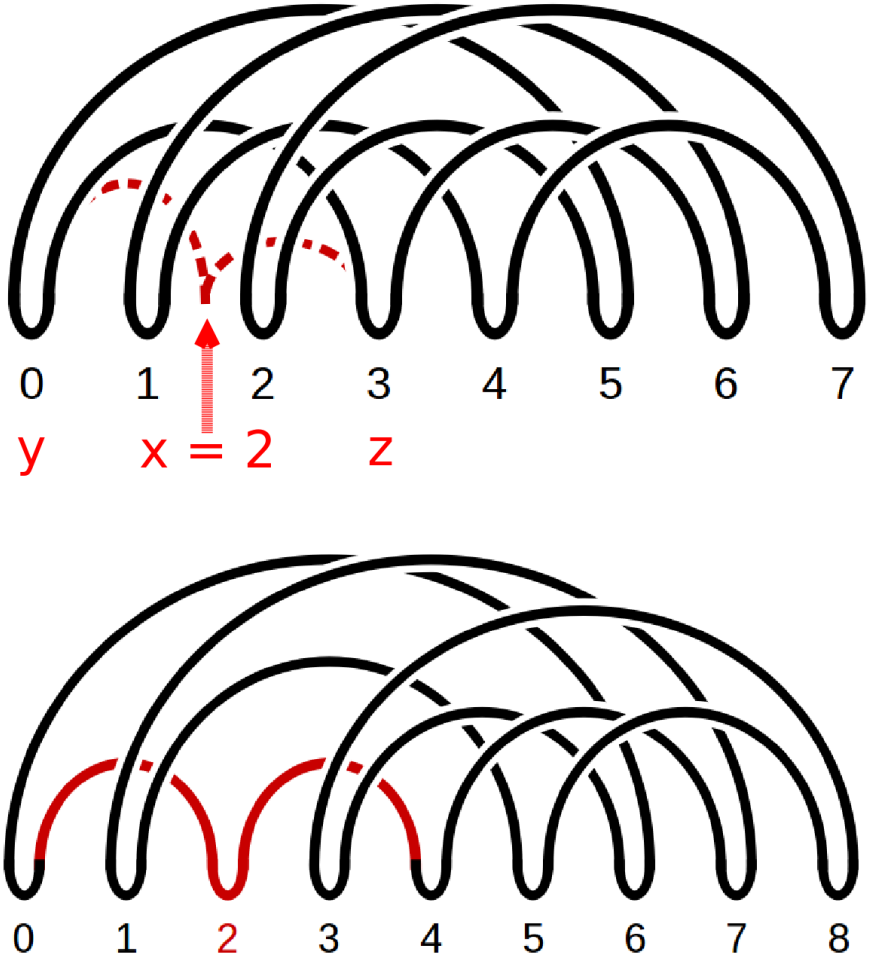


FIGURE 3.8 – Exemple d'insertion

Après une insertion le nombre de sommets augmente, mais le nombre de croisements peut éventuellement rester stable (comme dans l'exemple de la figure 3.8). Par ailleurs, le fait d'insérer x entre y et z ne change pas le type du nœud représenté si le sommet ainsi ajouté est réductible. On dit alors que x est *insérable* entre y et z.

3.7 Inversion

L'inversion est une opération plus complexe, dont l'intérêt reste encore à démontrer. On remarque qu'après une première inversion, on a toujours "l'inversion de l'inversion" reste identique.

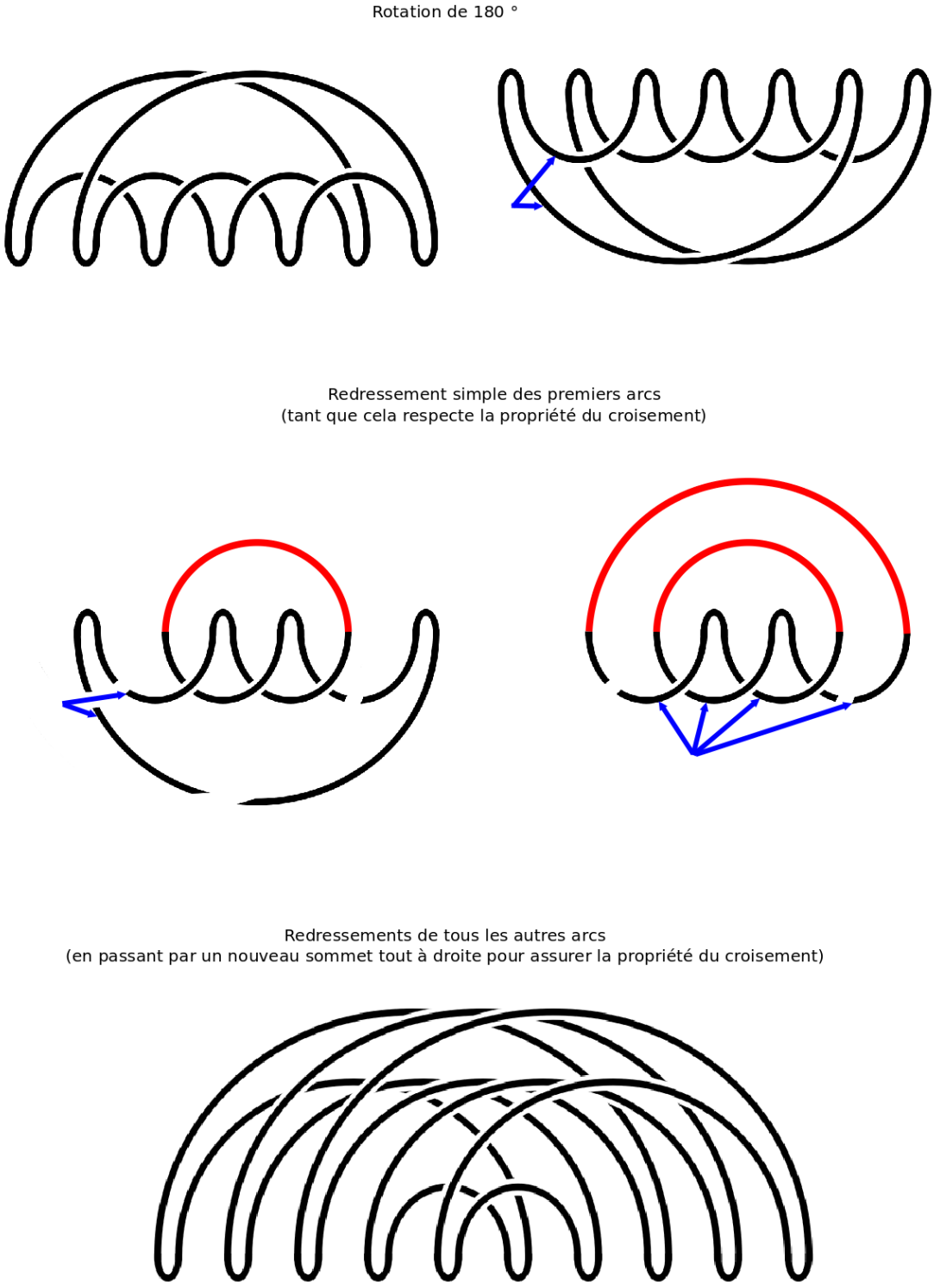


FIGURE 3.9 – Exemple d'inversion

3.8 Mirroir

Ce n'est pas une transformation du diagramme, dans le sens où le type du nœud représenté peut changer. Cependant c'est une involution (pour le type du nœud et non pour le diagramme), et une transformation classique qu'il parait donc important d'étudier.

L'algorithme actuel est cependant très naïf : il s'agit de prendre chaque arête de droite à gauche pour la faire passer par un nouveau sommet ajouté tout à droite. Ainsi à chaque étape l'arc considéré passait en dessous des arcs précédemment traités et au dessus des suivants, et se retrouve au dessus des arcs précédemment traités et sera en dessous des suivants. Les croisements entre tous les couples d'arcs se retrouvent donc inversés.

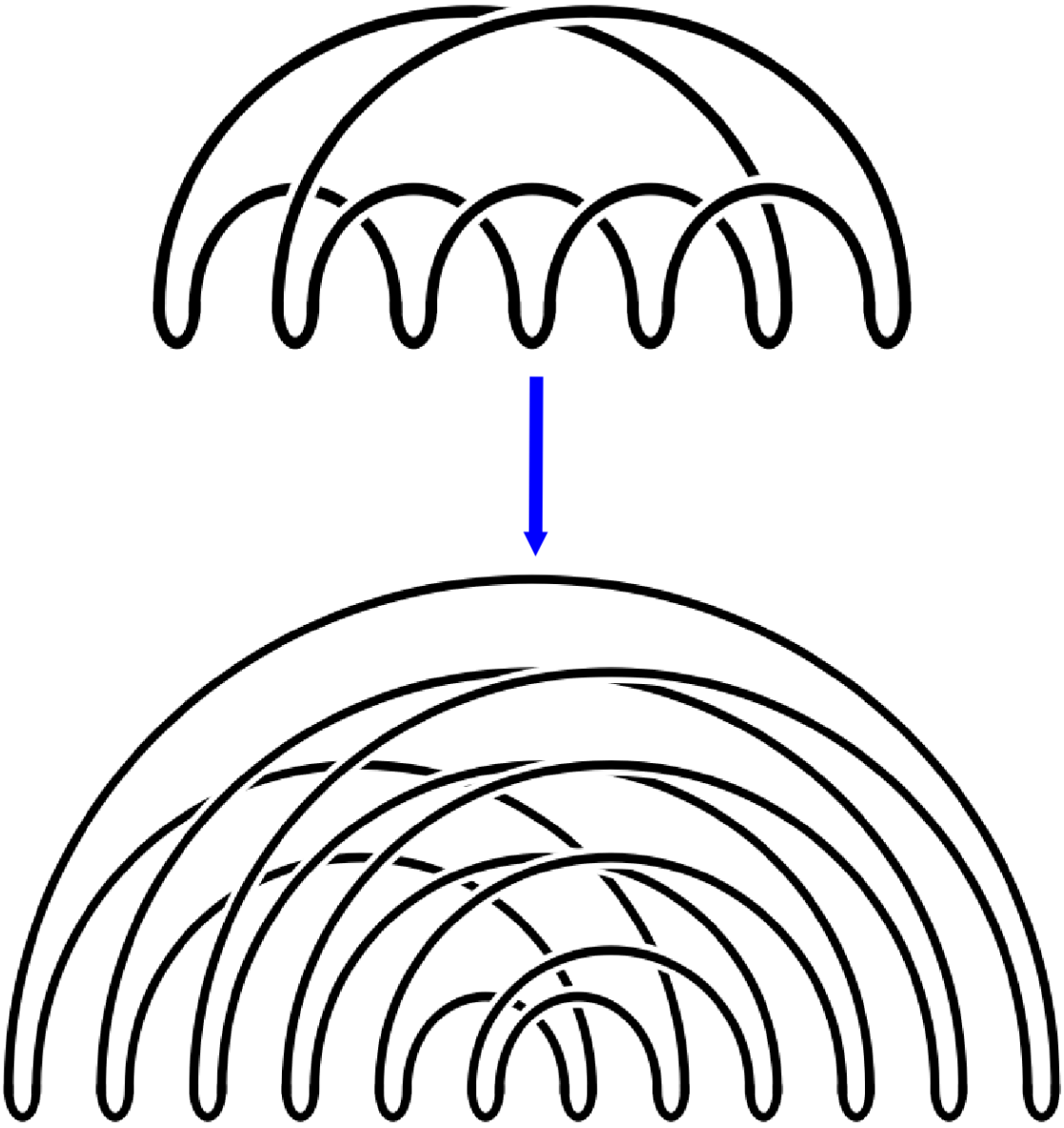


FIGURE 3.10 – Exemple de mirroir

Deuxième partie

Le programme ANR_TEOMATRO

Chapitre 4

Cahier des charges du programme

4.1 Contexte

4.1.1 Cadre général

Il s'agit de réaliser un programme dans le cadre de mon stage de Master 2 de mathématiques et du projet ANR "TEOMATRO"[1]. L'objectif du stage est la compréhension d'algorithmes sur les nœuds, issus de la théorie des matroïdes orientés, ainsi que leur mise en œuvre dans une application graphique. Ce programme devra permettre une visualisation graphique de certains algorithmes afin d'aider à leur compréhension et à l'évaluation de nouvelles conjectures.

4.1.2 Moyens techniques et ressources humaines

La conception des algorithmes est basée sur différentes ressources. Il s'agit en premier lieu d'une publication de M. Ramirez, maître du stage, "Spatial Graphs, Knots and the Cyclic Polytope", ainsi que différents algorithmes qu'il m'a expliqués directement car non encore publiés. D'autres algorithmes constituent des expérimentations de différentes conjectures formulées lors de nos entretiens.

La modélisation des idées mathématiques abstraites en concepts informatiques, et l'implémentation sont à ma seule charge. Par ailleurs les moyens techniques mis à ma dispostion sont constitués de mon ordinateur portable (GNU/Linux) pour la conception, ainsi que de l'ordinateur de bureau (Apple/MacOS) de mon maître de stage pour l'expérimentation. Le choix des outils logiciels devra en particulier permettre un déploiement multi-plateformes.

4.1.3 Calendrier

La période du stage s'étend du 1er mars 2013 au 31 juillet 2013. Cette période comprend l'étude théorique du sujet, l'implémentation des premiers algorithmes, et plusieurs autres cycles de la forme :

1. Conception de conjectures
2. Elaboration d'algorithmes permettant de les tester
3. Implémentation
4. Evaluation de la conjecture
5. Choix de garder ou non la fonctionnalité

1. http://www.math.univ-montp2.fr/~ramirez/Teomatro/teomatro.html

4.1.4 Vocabulaire

Le vocabulaire relatif aux concepts matématiques a été défini dans la partie théorique du présent rapport. Les termes spécifiques aux outils informatiques et à la conception des algorithmes sont précisés dans la partie relative à l'implémentation qui suit.

4.1.5 License

Les outils utilisés, et les choix d'implémentation devront permettre d'appliquer une licence convenable au programme réalisé. C'est à dire qu'il doit être possible de réutiliser, de modifier, et d'étendre ce programme dans le cadre de la recherche ANR. Il doit en revanche être possible de protéger le code du programme s'il venait à être diffusé ou vendu sous forme compilé à un public plus large. Le choix d'une license spécifique n'étant pas précisé au début de la conception, les choix d'implémentation évitent, dans la mesure du possible, de poser des contraintes sur la license finale.

4.2 Fonctionnalités

4.2.1 Modélisation d'un nœud

L'utilisateur doit pouvoir, à l'aide d'une interface, fournir un nœud au programme. Le format d'entré doit être le plus naturel possible afin qu'il soit aisé de fournir au programme un nœud dont on connait le diagramme classique.

Le nœud doit également pouvoir être fourni par un fichier que l'utilisateur peut ouvrir dans une interface. Le format de fichier doit être normalisé et lisible par l'humain afin de permettre d'une part de le modifier directement, et d'autre part qu'il puisse être généré par un autre programme éventuel.

De manière interne, le nœud doit être modélisé de façon à permettre l'application de différents algorithmes, et en particulier sa transformation en diagramme sur le polytope cyclique.

4.2.2 Modélisation d'un diagramme sur le polytope cyclique

De la même manière, un diagramme sur un polytope cyclique doit pouvoir être directement fourni par l'utilisateur, directement ou via un fichier, en utilisant un format naturel pour la lecture par l'homme.

Un diagramme sur le polytope cyclique peut également être le résultat d'un algorithme à partir d'un diagramme classique, et doit donc pouvoir être sauvegardé dans un format compatible à sa lecture future, et normalisé pour une utilisation (ou création) éventuelle par un autre programme.

De manière interne, le diagramme sur le polytope cyclique doit être modélisé pour permettre l'application de différents algorithmes de transformation, dont certains seront conçus au fur et à mesure. Le choix de la modélisation doit donc avoir une certaine souplesse de manipulation et rester efficace en terme de complexité des opérations envisagées.

4.2.3 Affichage graphique d'un diagramme sur le polytope cyclique

Le programme doit permettre, au minimum, un affichage graphique d'un diagramme sur le polytope cyclique à chaque étape des algorithmes qui lui seront appliqués. Une convention de représentation devra être choisie, dans le cadre d'une modélisation en trois dimension ou d'une projection en deux dimension. Les choix d'échelles et les déformations éventuelles par rapport à

une stricte représentation du polytope cyclique devront permettre un visualisation plus agréable et adapté pour y lire les informations pertinentes.

Des fonctionnalités supplémentaires pouvent être ajoutés, comme la gestion d'un historique des transformations, ou une présentation graphique des actions réalisables pour aider le choix de l'utilisateur.

4.2.4 Implémentation des algorithmes

Le premier algorithme qui sera implémenté doit permettre, à partir d'une représentation d'un diagramme classique, de fournir une représentation du diagramme sur le polytope cyclique du même nœud. Autant que possible, les différentes étapes feront l'objet d'un affichage graphique.

Ensuite, les autres algorithmes s'appliquent sur le diagramme sur le polytope cyclique. En premier lieu, le concept de réduction simple devra être implémenté. Par la suite différentes transformations seront ajoutées. Ces opérations doivent être disponibles pour l'utilisateur à l'aide d'éléments graphiques, il doit être possible de choisir parmi toutes les opérations en controlant le résultat immédiatement via un retour graphique.

Chapitre 5

Implémentation du programme

Ce chapitre présente d'une part les différents choix relatifs à la conduite du projet, et d'autre part les explications relatives à l'implémentation des algorithmes.

On présente ici les structures de données, et les algorithmes ; le code lui même est documenté séparément (documentation *Doxygen* disponible avec le code source).

En particulier, l'algorithme permettant de déformer un diagramme classique pour le placer sur le polytope cyclique est décrit en détails dans la partie 5.2.

5.1 La conduite du projet

5.1.1 Le choix langage de programmation

Le choix du langage de programmation est influencé par plusieurs critères :
- Rapidité du programme final
- Possibilité de déploiement sur plusieurs plateforme (au moins Apple/MacOS et GNU/Linux)
- Existence d'une bibliothèque graphique performante
- Expérience avec le langage

Les langages interprétés ont été écartés car ils produisent des programmes plus lents. Les langages dans lesquels j'avais plus d'expérience ont été étudiés :
- Le *C* est probablement le langage permettant une rapidité optimale pour le programme. Cependant la difficulté d'utiliser une conception par objet, et la tendance à nécessiter plus de lignes de code pour effectuer des opérations simples et une interface graphique ont été des obstacles pour ce projet.
- Le langage *Java* convient très bien à une conception orienté objet, avec des performances légèrement inférieures. Il existe cependant peu de bibliothèques graphiques réellement multiplateforme, performante, et ditribués avec une license permissive.
- Le *C++* a été choisi, il est presque aussi efficace que le *C*, permet une conception par objet et l'utilisation d'une bibliothèque standard (*STL*) pour simplifier le code source. De plus il existe plusieurs bibliothèques graphiques libres et performantes.

Parmi les bibliothèques graphiques, après avoir considéré *GTK* (et *GTKmm* pour le *C++*), mon choix s'est porté sur la bibliothèque *Qt* car j'en connaissais les bases, et qu'elle permet un développement relativement rapide pour des applications de petite taille. Par ailleurs elle est

distribuée sous une license libre de type GNU GPL ou LGPL [1].

5.1.2 Choix de l'environnement de développement et déploiement

L'IDE *QtCreator* est apparu comme un choix naturel en permettant une gestion de projet simple et portable via l'outil libre *qmake* intégré. Le projet pourra donc être diffusé, en tout cas dans un premier temps transmis à mon directeur de stage, en fournissant l'arborescence du code source et un fichier de configuration de projet *qmake*.

Cet IDE intègre un analyseur de performance, et un outil d'internationalisation *QtLinguist*. L'outil de documentation du code source *Doxygen* sera utilisé en complément.

Afin de dépendre le moins possible de la bibliothèque *Qt*, tous les algorithmes et structures de données indépendants de l'affichage graphique (y compris les opérations d'entrée/sortie dans les fichiers) seront codés de manière à ne pas faire appel à la librairies *Qt*. Il s'agit du dossier *core* des fichiers sources.

5.1.3 Cycles de développement

Un premier cycle de développement consiste à implémenter et tester l'algorithme de passage d'un diagramme classique à un diagramme sur le polytope cyclique, ainsi que l'affichage de ce dernier.

Un second cycle consiste à implémenter le concept de réduction simple, et à tester son efficacité.

Par la suite différentes conjectures sont proposées, et font l'objet d'une implémentation afin de les tester par l'expérimentation.

Les détails de l'implémentation sont disponibles dans la documentation du code source. On présente ici uniquement les principaux concepts directement en rapport avec l'implémentation des algorithmes.

5.2 Implémentation de l'algorithme de déformation du nœud sur le polytope

5.2.1 Description de l'algorithme et des structures de données

La donnée initiale est un code de Gauss. Il induit un diagramme classique du nœud qui est modélisé par un objet `Knot`, lui même contenant des croisements modélisés par des objets `Crossing`.

L'idée est ensuite de faire passer une courbe (gamma) dans le diagramme vérifiant :
- les extrémités de gamma se situent à l'extérieur du diagramme
- gamma ne s'intersecte pas avec elle même
- gamma est localement confondue avec le nœud à chaque croisement au niveau de l'arc passant au dessus (les segments)
- gamma traverse transversalement le diagramme partout ailleurs

Ensuite, on place gamma sur l'axe horizontal, en déformant le nœud. Le diagramme obtenu a donc tous les croisements sur l'axe horizontal. De plus à chaque croisement c'est l'arc horizontal (le segment) qui passe au dessus de l'autre. Cette structure est représenté par un objet `Gamma`.

1. http://qt.digia.com/licensing

Enfin on supprime tous les arcs inférieurs et les segments que l'on remplace uniquement par des arcs supérieur. On obtient une représentation du nœud sur des arêtes du polytope cyclique. Cette structure est représenté par un objet `Cypol`.

L'algorithme est schématisé dans la figure 5.1.

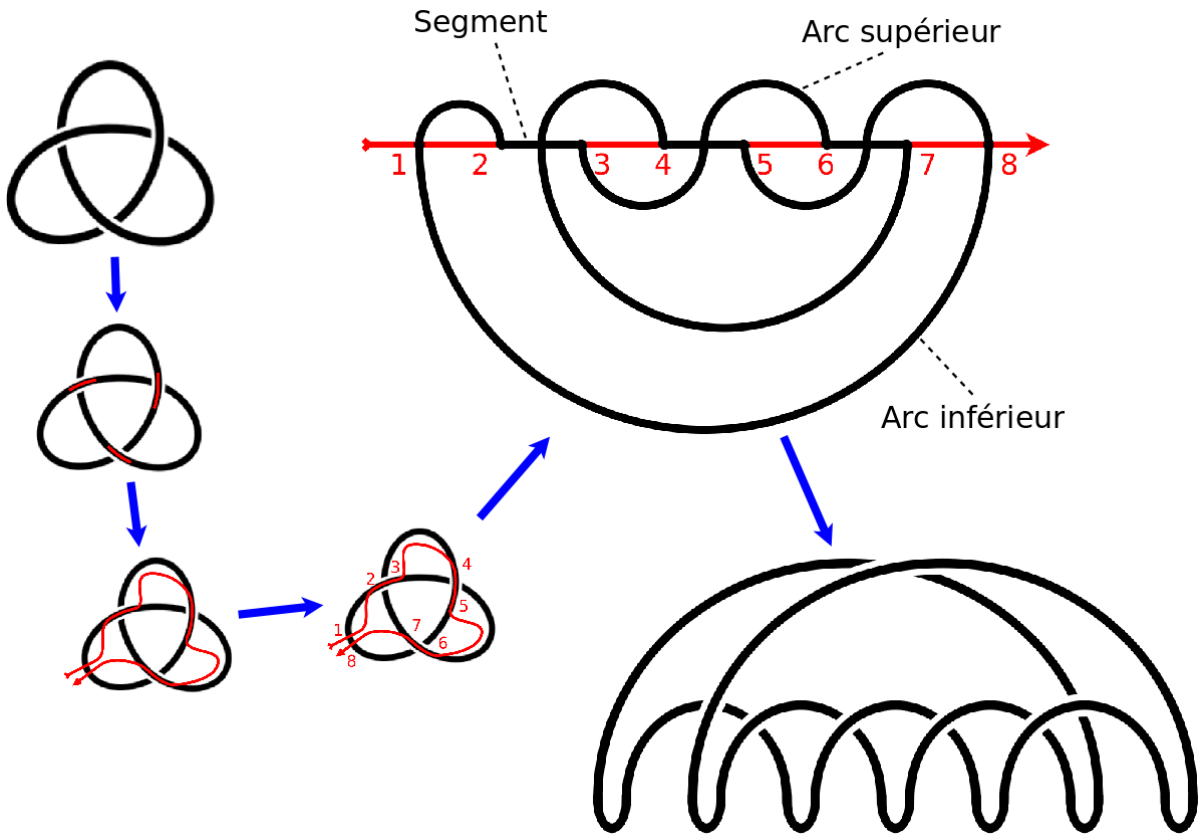


FIGURE 5.1 – Construction de la courbe gamma et du polytope cyclique

5.2.2 Du diagramme `Knot`, à la structure `Gamma`

Un nœud est représenté par un objet `Knot`, initialisé par un code de Gauss. Cette structure représente le diagramme classique du nœud, et contient les croisements représentés par des objets de type `Crossing` qui forment les sommets du graphes 4-réguliers.

Chaque `Crossing` est lié à 4 autres `Crossing` pour former le graphe. Les quatres voisins sont dans quatres directions représentées par un `Cardinal`. C'est l'orientation qui détermine le sens de l'arc Est-Ouest, qui est en dessous. Les différentes localisations susceptibles d'être traversées par gamma sont représentées sur les `Crossing` (voir figure 5.2).

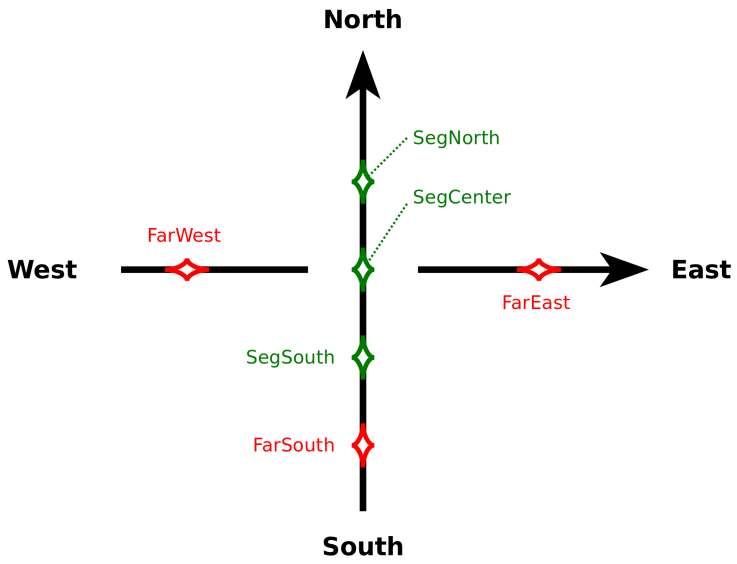


FIGURE 5.2 – La structure `Crossing` dans `Knot`.

L'idée générale de l'algorithme est la suivante :

1. Les faces du graphes sont colorés avec 2 couleurs (blanc/noir) de façon à ce que deux faces adjacentes soient de couleurs différentes.
2. On fixe un sommet de départ.
3. On parcours les faces de façon à obtenir un arbre couvrant des faces noires.
4. A chaque croisement, selon si les deux faces noires sont liées ou non dans l'arbre on peut dessiner la courbe gamma localement (voir figure 5.3).
5. A partir du croisement de départ, on parcours gamma pour former la courbe définitive qui consiste en une suite de croisements et positions sur ces croisements.
6. On parcourt le nœud pour le placer relativement à gamma.

La courbe gamma est alors représentée par une droite horizontale avec plusieurs points numérotés de 0 à $n-1$. A chaque point on associe un objet de type `GVertex`. En suivant le nœud, on avance de `GVertex` en `GVertex` en passant soit au-dessus, en-dessous, ou à l'intérieur de la courbe gamma (voir figure 5.4).

On obtient un nœud sur la courbe gamma horizontale (voir figure 5.5). Ce ne nœud est représenté par la structure `Gamma` contenant des `GVertex`. Les seuls croisements correspondent aux segments Nord-Sud où le nœud et gamma sont confondus. C'est donc la partie horizontale qui passe au-dessus (car l'arc Nord-Sud est toujours au dessus de l'arc Est-Ouest).

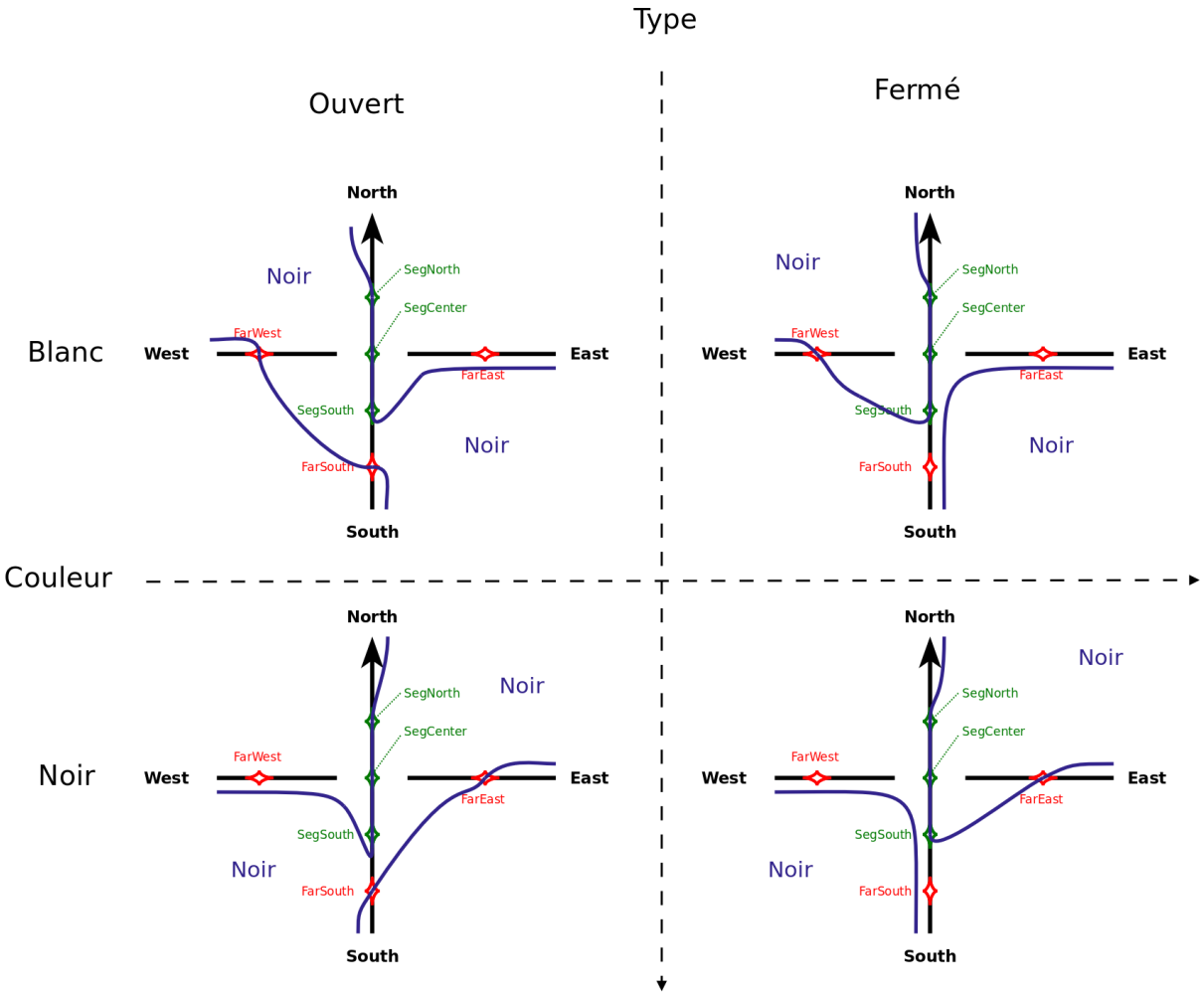


FIGURE 5.3 – Gamma au voisinage d'un croisement.

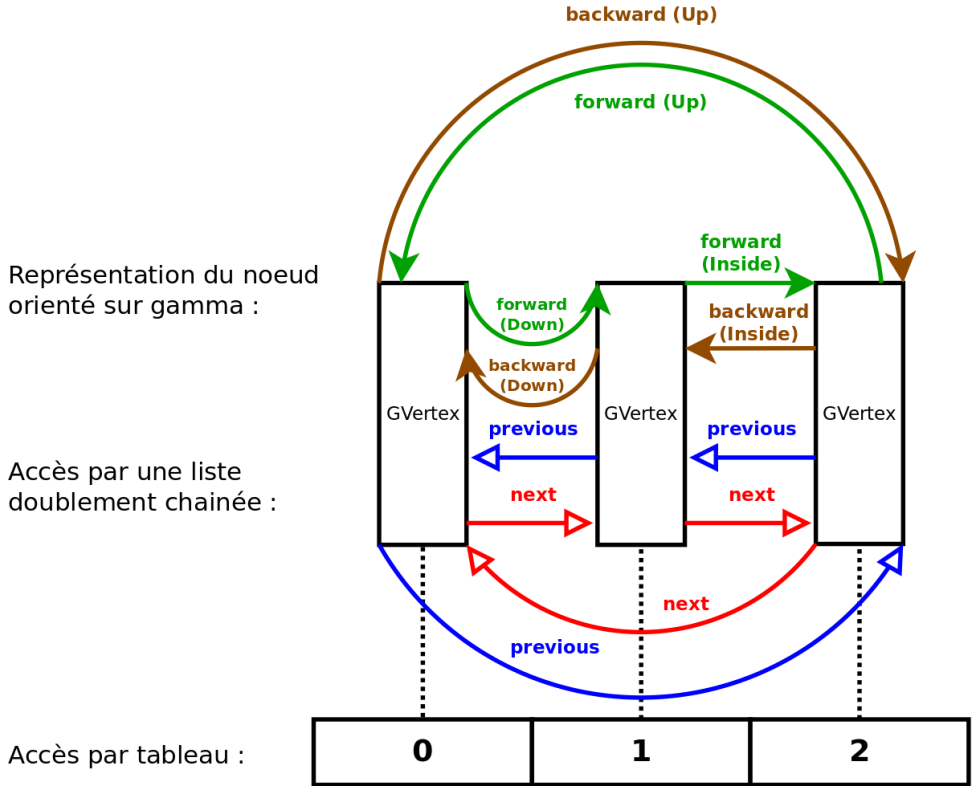


FIGURE 5.4 – La structure `GVertex` dans `Gamma`.

5.2.3 De la structure `Gamma`, à la structure `Cypol`

L'étape suivante consiste à supprimer tous les segments, que l'on va remplacer par un ou plusieurs arcs passant strictement au dessus de gamma. Deux cas sont possibles (voir figure 5.6) :

1. Soit il suffit de remplacer le segment par un arc avec les mêmes extrémités.
2. Soit il est nécessaire d'ajouter plusieurs arcs pour conserver la propriété du croisement.

On obtient une représentation avec des arcs au dessus, qui se croisent éventuellement en respectant la propriété du croisement, et des arcs en dessous qui ne se croisent pas. Il reste à supprimer ces arcs inférieurs. Pour cela on les remplace un à un (de gauche à droite) par deux arcs, en passant par un nouveau point de gamma à droite de tous les autres (voir figure 5.7).

On obtient enfin une représentation du nœud sur le poytope cyclique. En effet tous les arcs sont maintenant au dessus de gamma et respectent la propriété du croisement. Cette structure est représenté par un objet de type `Cypol`, contenant des `CVertex`.

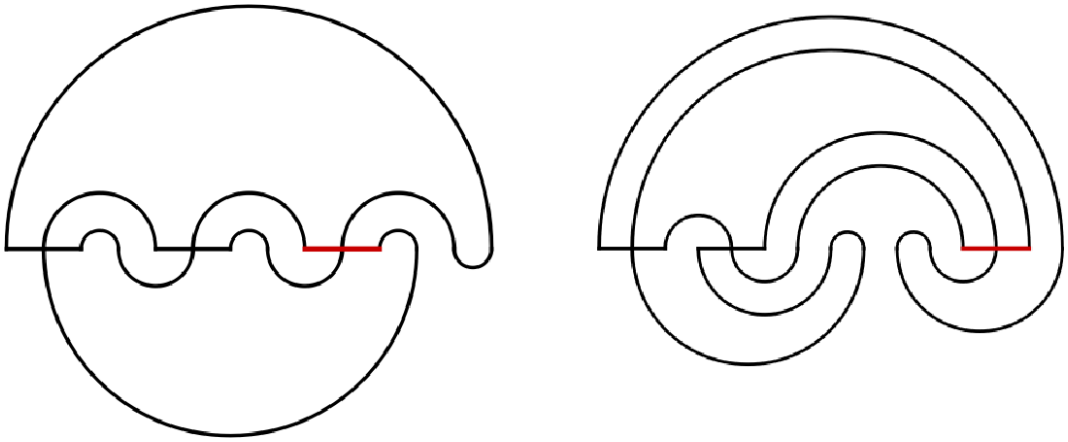


FIGURE 5.5 – La structure `Gamma`.

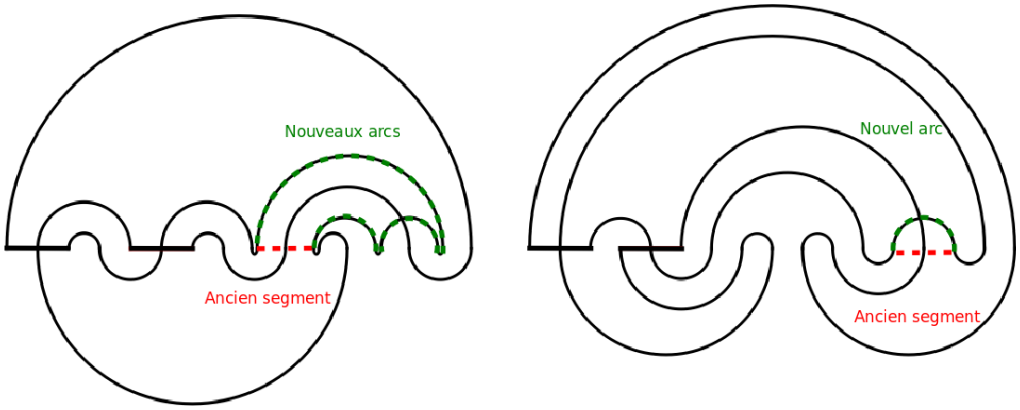


FIGURE 5.6 – Deux cas pour supprimer un segment

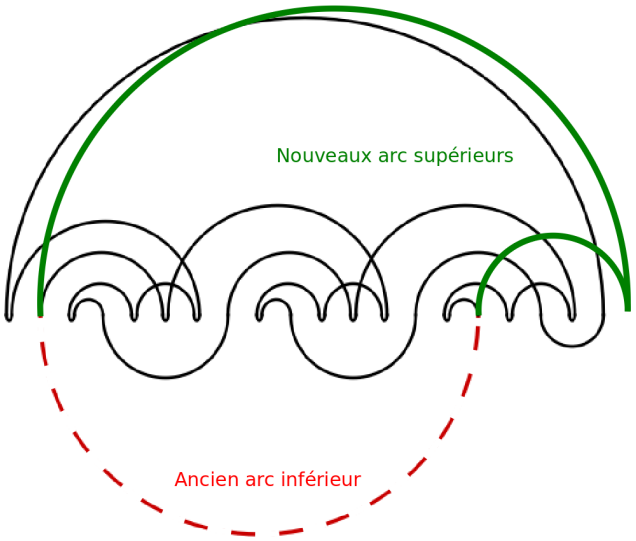


FIGURE 5.7 – Suppression d'un segment inférieur

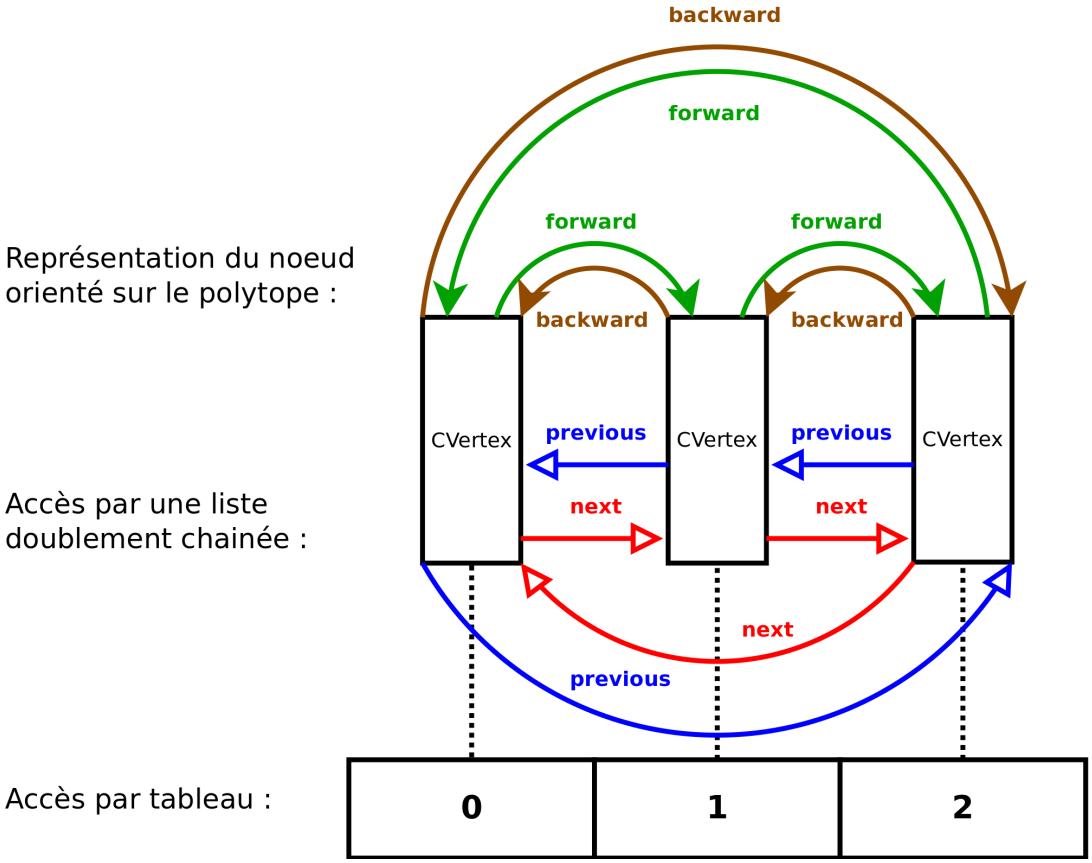


FIGURE 5.8 – La structure `CVertex` dans un `Cypol`

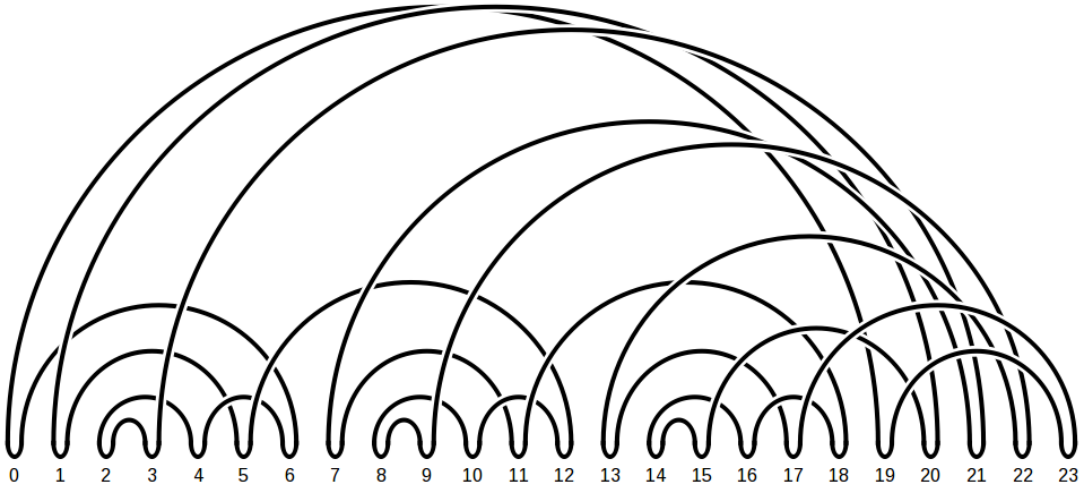


FIGURE 5.9 – Représentation d'un nœud sur le polytope cyclique

5.2.4 Complexité

La complexité de cette transformation du code de Gauss en une structure représentant le nœud sur le polytope cyclique est linéaire (temps et espace mémoire) en le nombre de croisements du diagramme.

1. Une structure GaussCode créée en temps linéaire permet d'obtenir en temps constant toutes les informations sur les voisins d'un croisement.
2. La strucutre `Knot` peut ainsi créer tous les `Crossing` en temps linéaire.
3. Le parcours du graphe pour la coloration visite chaque arête une fois, et chaque croisement exactement deux fois.
4. Le parcours du graphe pour décider du type de chaque `Crossing` visite chaque croisement au plus deux fois.
5. Les sommets de la courbe gamma sont placés localement sur chaque croisement en temps constant (au plus 5 sommets par croisements sont placés en connaissant uniquement les informations locales à un croisement).
6. La reconstruction de la structure `Gamma` qui place les sommets dans l'ordre avec la position relative du nœud par rapport à `Gamma` se fait en temps linéaire (par rapport au nombre de sommets de `Gamma`, au plus 5 fois le nombre de croisements).

Ensuite pour la suppression des segments sur `Gamma` : la liste des segments est fournie par un seul parcours des sommets et chaque segment est traité en temps constant. Il est en de même pour les arcs inférieurs.

Attention cependant, cela est vrai dans le cas où tous les segments, puis tous les arcs inférieurs sont supprimés d'un coup, la réindexation des sommets intervenant uniquement à la fin. Si on souhaite mettre à jour l'indexation à chaque étape, chaque étape a alors une complexité linéaire et la procèdure complète devient quadratique en le nombre de sommets (donc aussi en le nombre de croisements). C'est cette procédure qui est suivie dans les utilisations de l'algorithme par le programme afin de permettre un affichage étape par étape.

Il faut également noter que le nombre de sommets sur la structure `Gamma` va augmenter, mais ce nombre reste majoré par 11 fois le nombre de croisements :

- intialement : 5 fois le nombre de croisements
- après suppression des segments : + 2 fois le nombre de croisements
- après suppression des arcs inférieurs : + 1 fois le nombre d'arc inférieurs
- le nombre d'arcs inférieurs étant majoré par 3 fois le nombre de croisements

5.3 Les transformations du diagramme sur le polytope cyclique

Le sommets du polytope cyclique sont représentés par la structure `CVertex` qui permet un accés et une modification en temps constant :

- d'un sommet à partir de son numéro par un accés direct
- des deux voisins dans le sens de parcours du nœud (*forward/backward*) par un accés direct
- des deux sommets immédiatement à droite et à gauche (*next/previous*) par un accés direct
- des deux voisins à droite et à gauche en fonction des positions relatives des sommets *backward/forward*

Toutes les transformations locales (réduction, échange, insertion) sont donc effectuées en temps constant, et les transformations globales (rotation, inversion, mirroir) ont une complexité linéaire. Cependant la réindexation des sommets pour permettre un accés par tableau a une complexité linéaire. De plus le test pour savoir si un sommet est réductible, échangeable ou insérable a une complexité linéaire pour chaque sommet.

Ainsi, à l'exception du test pour l'insertion, toutes les opérations (transformations et tests des futures transformations possibles) ont une complexité globale quadratique.

Les tests pour l'insertion sont implémentés de la façon suivante :

1. On considère chaque arc $\{y, z\}$
2. On considère tous les sommets $0 < x < n - 1$
3. On teste si on peut insérer x entre y et z
4. Pour ce dernier test on ajoute effectivement le sommet, et on teste sa réductibilité

On obtient $O(n^2)$ étapes (3). Chacune avec une complexité $O(n^2)$. Finalement la complexité est $O(n^4)$. Pour cette raison, les tests d'insertion peuvent être activés de manière optionnelle (et sont désactivés par défaut).

5.4 Générateur

Le générateur permet de fournir une liste de diagrammes sur le polytope cyclique avec un nombre n de sommets fixé.

On commence par générer la liste $(c_0, c_1, \ldots, c_{(n-1)!-1})$ de tous les codes de polytopes. Cela correspond à la liste des permutations de $(0, 1, \ldots, n-1)$ commençants par 0.

On initialise une liste de polytopes résultats $(r)_k$. On ajoute un à un tous les polytopes c_i s'il ne s'agit pas d'un doublon de l'un des r_i précédemment ajoutés. Le concept de doublon peut être paramétré de différentes façons : à rotation près, à réduction près, ...

En particulier la complexité est factorielle, donc impossible à utiliser en pratique pour des valeurs de n supérieures à une dizaine de sommets (temps *et* utilisation mémoire).

5.5 Implémentation de l'interface graphique

Les détails de l'implémentation de l'interface graphique sont décrits dans la documentation du code source. Les composants graphiques de la librairie *Qt* sont utilisés (boutons, listes, fenêtre, onglets,...).

On présente ici uniquement les principes du dessin des diagrammes.

5.5.1 Abstraction des coordonnées : `RenderingArea`

L'objectif de l'objet `RenderingArea` est de permettre de dessiner les diagrammes dans des coordonnées abstraites adaptés à la nature de ce diagramme en servant d'interface avec les coordonnées "physiques" du support. Ce dernier support pouvant être une zone de l'écran, ou un fichier PDF.

Cet objet permet en particulier de gérer les fontions de zoom et déplacements du diagramme, et permet de choisir si le ratio *hauteur/largeur* doit rester identique à celui des coordonnées abstraites.

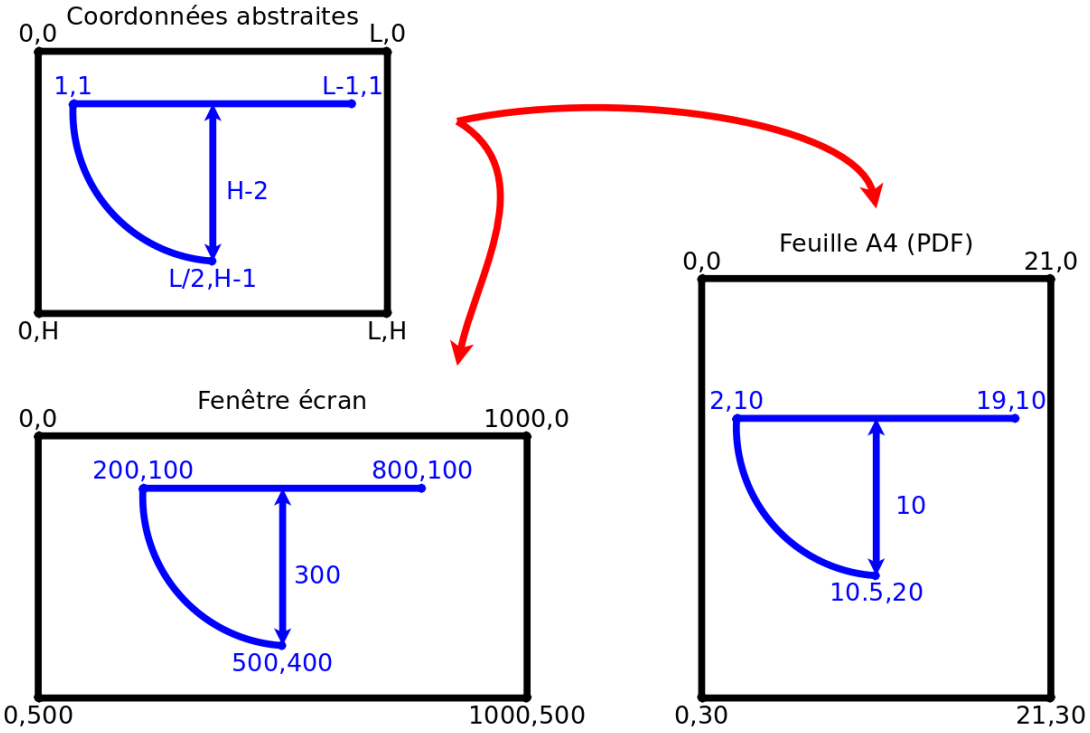


FIGURE 5.10 – `RenderingArea`

5.5.2 Une représentation du diagramme sur gamma

On considère un diagramme sur gamma avec n sommets (voir figure 5.11) :

- le cadre est un carré de coté $n+2$
- la courbe gamma est la droite $(1, n/2+1) - (n+1, n/2+1)$
- chaque sommet i se situe au point $(i+1, n/2+1)$
- un arc intérieur est un segment entre les sommets
- un arc supérieur (resp. inférieur) est un demi-cercle au dessus (res. en dessous) de la courbe gamma

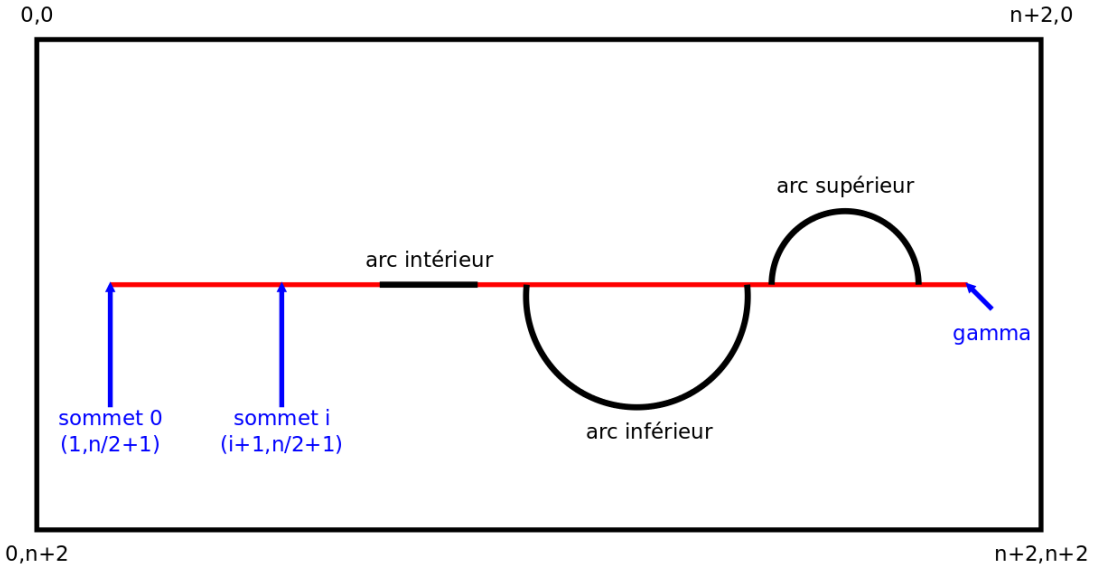


FIGURE 5.11 – Représentation (coordonnées abstraites)

5.5.3 Deux représentations du diagramme sur le polytope

Un première représentation est similaire à la représentation précédante du diagramme sur gamma, avec uniquement des arcs supérieurs et donc une hauteur du cadre réduite de moitié.

Une seconde représentation "circulaire" a été développée par la suite, afin de rendre le concept de rotation plus explicite. Il s'agit de déformer gamma pour la placer sur un cercle, et tous les arcs deviennent des cordes de ce cercle.

Dans les deux cas, les arcs sont dessinés de gauche à droite, de façon à être "empilés" pour respecter visuellement la propriété du croisement.

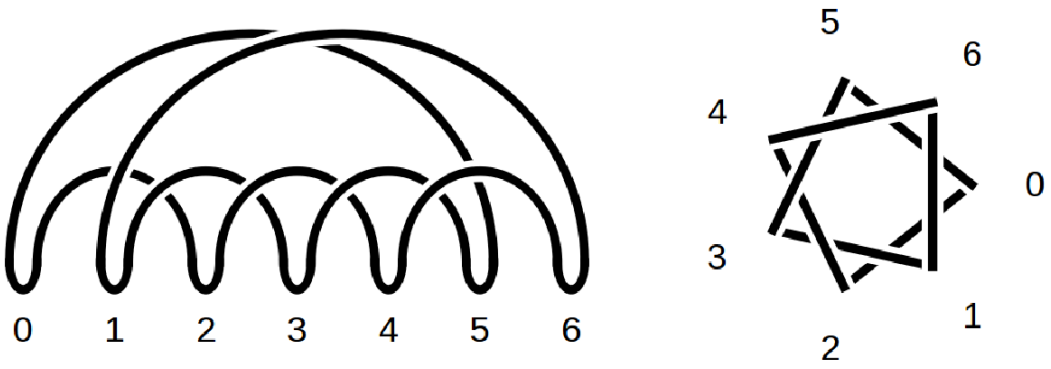


FIGURE 5.12 – A gauche : représentation classique, à droite : représentation circulaire

Chapitre 6

Quelques résultats établis à l'aide du programme, conjectures et ouvertures

6.1 Les trèfles

A l'aide du programme, une étude systématique de tous les diagrammes sur le polytope cyclique avec moins de 8 sommets a pu être effectuée. On a pu vérifier que les nœuds correspondants sont triviaux à l'exception des 3 suivants (à rotation près) :

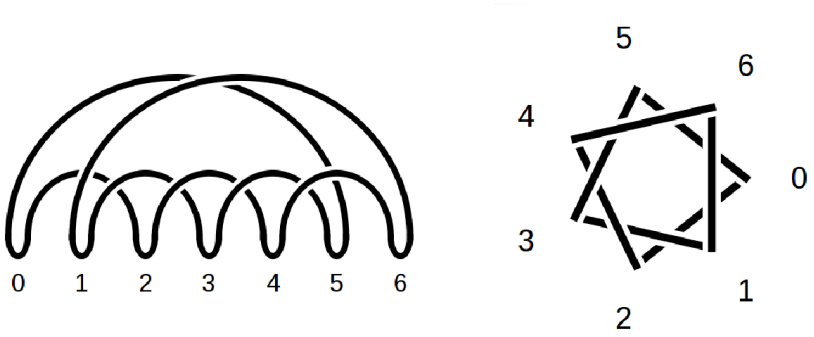


FIGURE 6.1 – Le trèfle gauche – 0 5 3 1 6 4 2 – (7 sommets, 7 croisements)

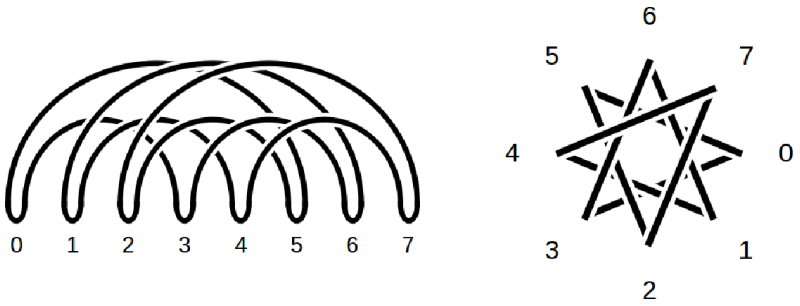


FIGURE 6.2 – Le trèfle gauche – 0 5 2 7 4 1 6 3 – (8 sommets, 16 croisements)

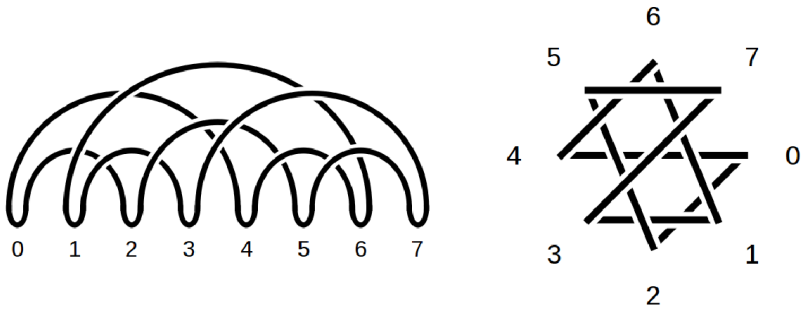


FIGURE 6.3 – Le trèfle droit – 0 2 5 7 3 1 6 4 – (8 sommets, 11 croisements)

De plus le diagramme du trèfle gauche avec 8 sommets n'a aucun sommet réductible. Cela confirme que les réductions ne suffisent pas à définir un "diagramme minimal" pour un nœud, que ce soit en observant le nombre de sommets ou de croisements.

Il serait pourtant intéressant de définir :

– un paramètre sur les diagrammes
– un ensemble d'opérations faisant décroitre strictement ce paramètre

tels qu'il existe pour chaque nœud un diagramme minimisant cet invariant et pouvant être obtenu à partir de tous les diagrammes du nœud en utilisant cet ensemble d'opérations. Il semblerait intuitivement logique que ce paramètre décroisse avec le nombre de croisements et/ou de sommets.

En considérant l'exemple du trèfle gauche, on peut chercher une suite d'opérations permettant à partir du diagramme à 8 sommets d'obtenir celui à 7 sommets. Une suite d'opérations (décrite dans la figure 6.4) a donné un résultat partiel puisque le nombre de croisements est décroissant, mais pas strictement, et le nombre de sommets augmente à une étape.

Dans cette transformation le paramètre "nombre de croisements" semble intéressant mais peut être pas assez fin pour justifier les opérations d'échanges et d'insertions qui semblent indispensables.

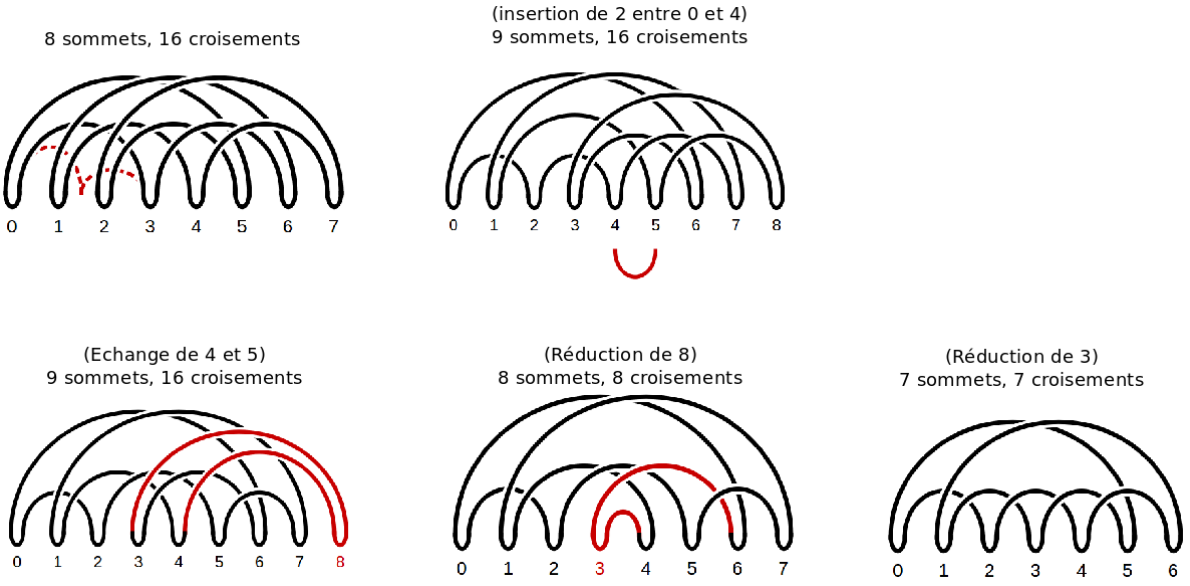


FIGURE 6.4 – Lien entre les deux trèfles gauches

6.2 Longueur de la ficelle

On considère une ficelle de longueur L et de section circulaire de diamètre D. On se demande alors quels sont les nœuds que l'on peut réaliser avec cette ficelle ?

Une autre formulation de la question est : considérant une ficelle de diamètre 1 et un diagramme classique d'un nœud avec n croisement, quelle longueur de courbe sera suffisante pour réaliser un plongement du nœud dans $\mathbb{R}^3$?

Plus précisément, on définit $L(n)$ comme la longueur minimale suffisante pour réaliser tous les plongements de tous les nœuds pour lesquels il existe un diagramme avec n croisements, en imposant que le plongement $h(\mathcal{S}^1)$ soit formé par des segments unitaires à extrémités dans $\mathbb{Z}^3$. Cela signifie que $L(n)$ est une longueur minimale suffisante pour réaliser tous les nœuds ayant un diagramme avec moins de n croisements avec une ficelle de diamètre 1.

Proposition 6.2.1.

$$L(n) \leq 22n^2$$

Démonstration. On considère un diagramme avec n croisements. En le plaçant sur la courbe gamma comme décrit dans la partie 5.2, on peut dessiner un diagramme équivalent sur la courbe gamma avec au plus $11n$ sommets. On remplace alors les demi-cercles des arcs supérieurs et inférieurs par des demi-carrés du plan (x, y) de $\mathbb{R}^3(x, y, z)$, et on remplace les segments par un demi-carré dans le plan orthogonal (x, z).

On obtient alors $11n$ arcs de longueur inférieure à $2n$. □

La constante 22 peut probablement être améliorée. Mais il faut surtout noter que la puissance 2 n'est pas optimale, on peut en effet montrer [1] que $L(n) = O(n^{3/2})$. La démonstration précédante a cependant l'avantage de fournir un algorithme très efficace.

6.3 Etude systématique

Dans l'objectif de poursuivre l'étude des propriétés des diagrammes de nœuds sur le polytope cyclique, il parait utile de considérer une étude systématique de *tous* les diagrammes de petite taille. C'est dans cet objectif que le générateur a été implémenté. Cependant l'ordre de grandeur factoriel du nombre de résultats rend l'expérimentation impossible même pour des diagrammes de tailles modestes. Il pourrait être utile de concevoir un nouvel algorithme de génération plus intelligent, qui n'énumère pas *toutes* les permutations avant de comparer les polytopes. Il faudrait pour cela trouver une méthode "naturelle" d'énumération des diagrammes qui ne passe pas par l'énumération des permutations de $[0, n-1]$.

6.4 Complétude de l'ensemble des transformations

Conjecture 6.4.1. *Etant donnés deux diagrammes sur le polytope d'un même nœud, il est possible de passer de l'un à l'autre uniquement en appliquant des réductions, échanges et insertions.*

Il y a certainement un lien qui reste à établir entre les mouvements de Reidemeister et les transformations sur un diagramme sur le polytope, qui permettrait une telle preuve. Dans le cas où cette conjecture s'avère fausse, il serait nécessaire de déterminer un ensemble d'opérations élémentaires plus large permettant de rendre une conjecture similaire vraie.

1. *Hamiltonian knot projections and lenth of thick knots*, Yuanan Diao, Claus Ernst, Xingxing Yu. L'algorithme est relativement complexe et utilise les cycles Hamiltoniens. Et on ne sait pas si la puissance 3/2 est optimale.

6.5 Echange et insertion, quand faut-il les appliquer ?

La concept de réduction parait être systématiquement une opération "positive" dans le sens où elle va dans la direction d'une représentation "plus simple" du nœud.

Cela est beaucoup moins clair pour l'échange, même si l'étude du nombre de croisements s'est avéré expérimentalement efficace pour décider s'il est "positif" d'effectuer un échange.

Enfin l'opération d'insertion est la moins "naturellement positive" car on aimerait *a priori* toujours réduire le nombre de sommets.

Ces considérations amènent à se poser les questions suivantes :

1. Existe-t-il un paramètre permettant de décider si un échange ou une insertion s'avère utile ?
2. Existe-t-il un paramètre plus pertinent que le nombre de sommets pour mesurer la "complexité" de la représentation et décider de la représentation minimale ?
3. Existe-t-il d'autres opérations naturelles sur les diagrammes ?

Il semble que l'on se confronte au même type de problématique qu'avec les mouvements de Reidemeister (où il n'y a que trois mouvements, mais on ne peut pas déterminer d'ordre naturel pour les effectuer). Cependant la structure du polytope cyclique, et éventuellement son lien avec la structure de matroïde et en particulier les partitions de Radon, permettraient peut-être de définir plus de paramètres pertinents sur un diagramme (et pas seulement le nombre de sommets ou de croisements).

Troisième partie

Annexes

Annexe A

Manuel du programme

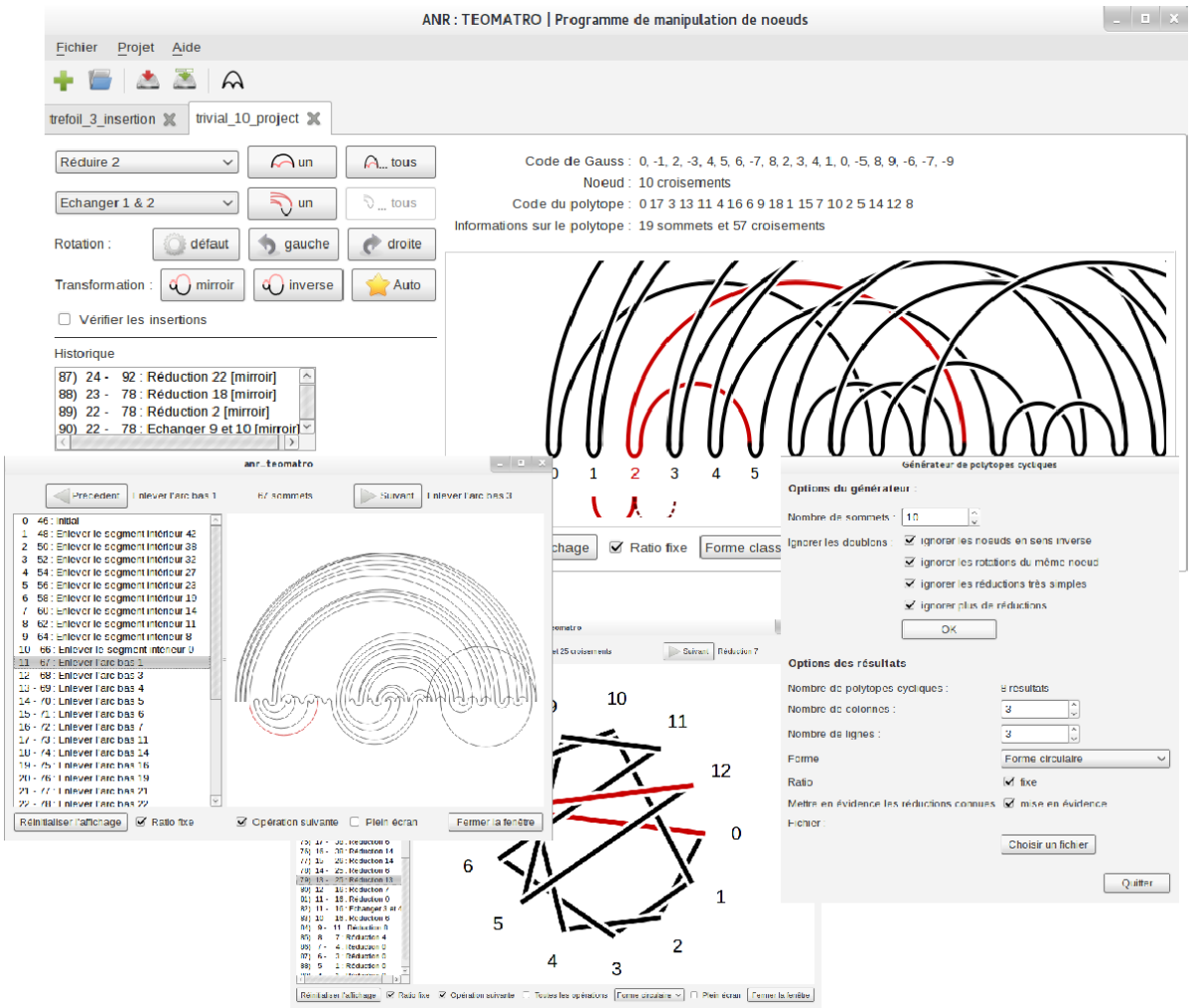
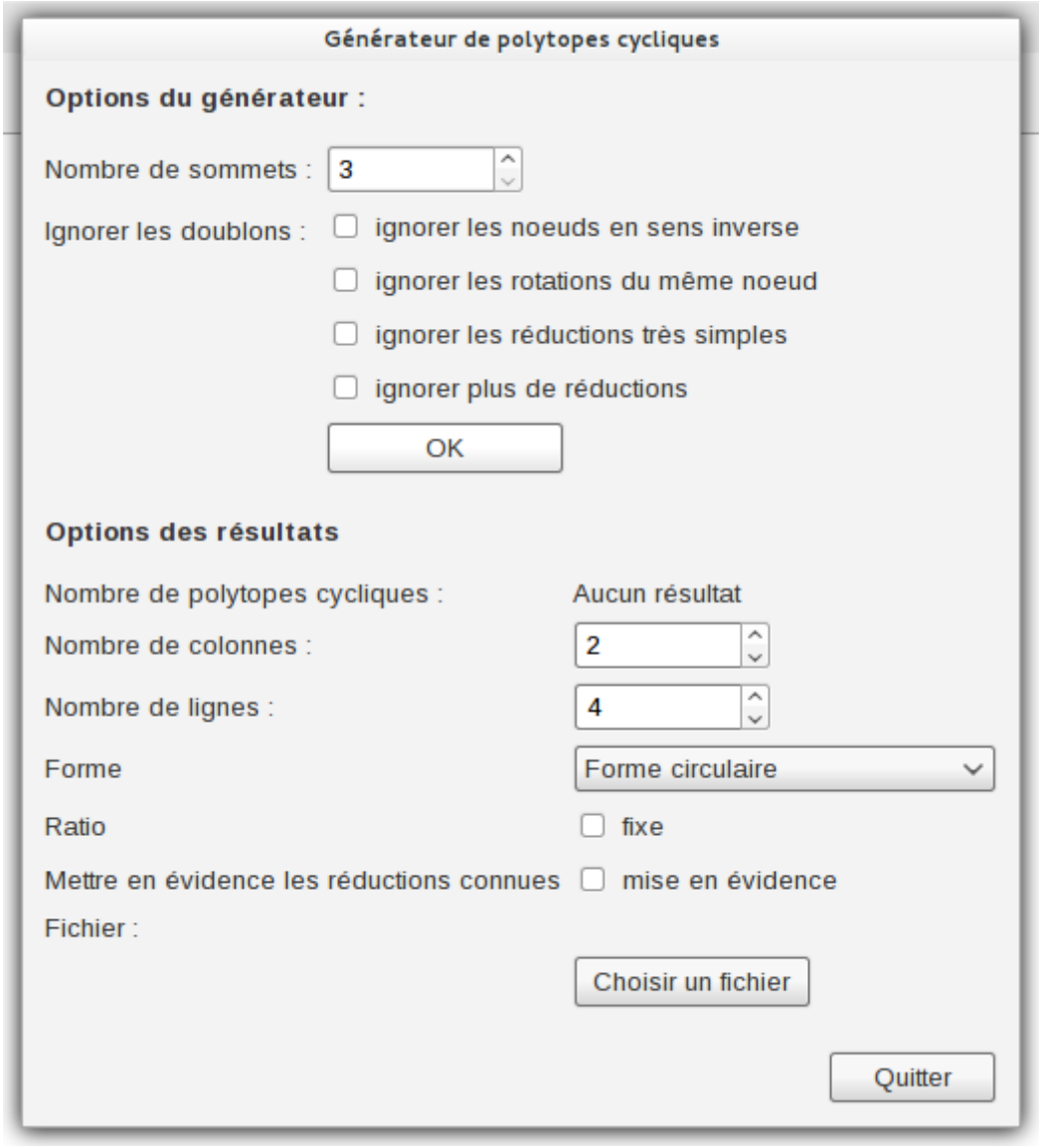


FIGURE A.1 – Le programme ANR_TEOMATRO

A.1 Générateur de nœuds

Un générateur de nœuds sur le polytope cyclique est accessible via le menu *Fichier -> Générateur de Polytopes Cycliques*. Il se présente sous la forme d'un dialogue :



A.1.1 Options du générateur

Le nombre de sommets est le nombre de sommets sur la courbe gamma à partir desquels les nœuds sur le polytope cyclique seront générés. Attention : Au delà d'une dizaine de sommets, la mémoire nécessaire et les temps de calculs peuvent être *très* importants.

Ignorer des doublons signifie que certains nœuds ne seront pas ajoutés à la liste des nœuds générés si (plusieurs choix possibles) :

- Il s'agit exactement du même nœud qu'un de ceux précédemment générés avec uniquement le sens de parcours différent.
- Il s'agit d'un nœud pouvant être obtenu à partir de l'un de ceux précédemment générés en appliquant uniquement des rotations.

– Le nœud contient une réduction très simple (il y a un arc entre un sommet et celui immédiatement à coté).
– Le nœud contient une réduction connue.

Lorsque ces critères sont choisis, cliquer sur OK lance la génération des nœuds.

A.1.2 Options des résultats

Lorsque les nœuds ont été générés, le nombre de résultats apparait.
On peut alors choisir le nombre de lignes et de colonnes par page de résultats. On choisit également la forme du dessin (classique ou circulaire) et si l'on souhaite que les réductions connues apparaissent en évidence.

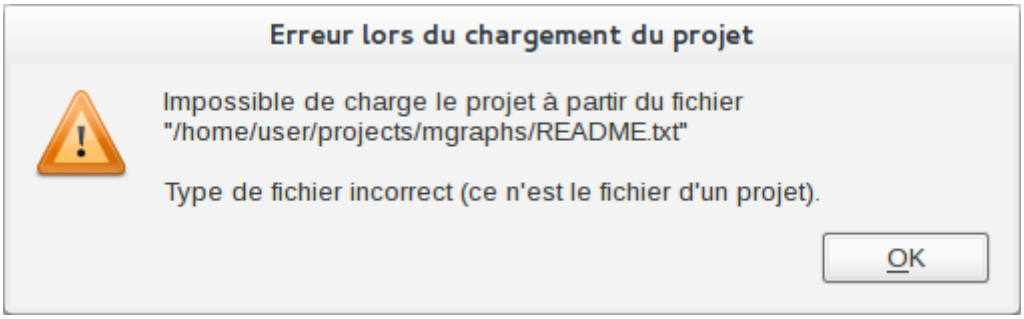
On choisit enfin le fichier où enregistrer les résultats (il doit s'agir d'un fichier avec une extension *pdf* et les droits d'écriture).

L'annexe C présente un exemple de résultats.

A.2 Création et sauvegarde d'un projet

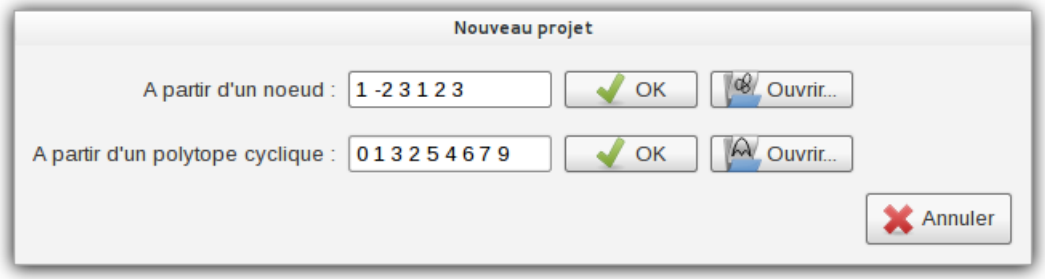
A.2.1 Ouvrir un projet

Il est possible d'ouvrir un projet précédemment enregistré par le menu *Fichier -> Ouvrir un projet*. Il suffit alors de sélectionner le fichier désiré via un dialogue.
En cas d'erreur, un message explicatif apparait :



A.2.2 Créer un projet

Pour créer un projet, il faut utiliser le menu *Fichier -> Nouveau projet*. Le dialogue suivant apparait :



A partir d'un code de Gauss

Saisir dans la première ligne le code de Gauss du nœud puis cliquer sur OK. Ou bien cliquer sur Ouvrir pour choisir un fichier contenant un nœud enregistré.

Le code de Gauss doit être une série de nombres signés. Il n'est pas nécessaire d'utiliser des nombres dans un ordre particulier, et le signe positif peut être omis. Les nombres sont séparés par un espace et/ou une virgule.

Ainsi le trèfle peut s'écrire au choix (et sera représenté de manière strictement identique dans le programme) :

- +0 -1 +2 +0 +1 +2
- 0,-1,2,0,1,2
- 123 -124 +155, 123 124 155

A partir du code d'un polytope cyclique

Saisir dans la seconde ligne le code du nœud sur le polytope puis cliquer sur OK, ou bien bien cliquer sur Ouvrir pour choisir un fichier contenant un polytope enregistré.

Le code du polytope doit être écrit comme une suite de nombres dans l'intervalle $[0, 1, \ldots, n-1]$ séparés par une virgule et/ou un espace. Par exemple :

- 0 3 2 1 (OK)
- 0,3, 2 1 (OK)
- 1 4 3 2 (Ne fonctionne pas)

A.2.3 Sauver un projet

Pour enregistrer le projet ouvert dans l'onglet courant, on ouvre un dialogue de sélection de nom de fichier par *Projet -> Sauver Sous*.

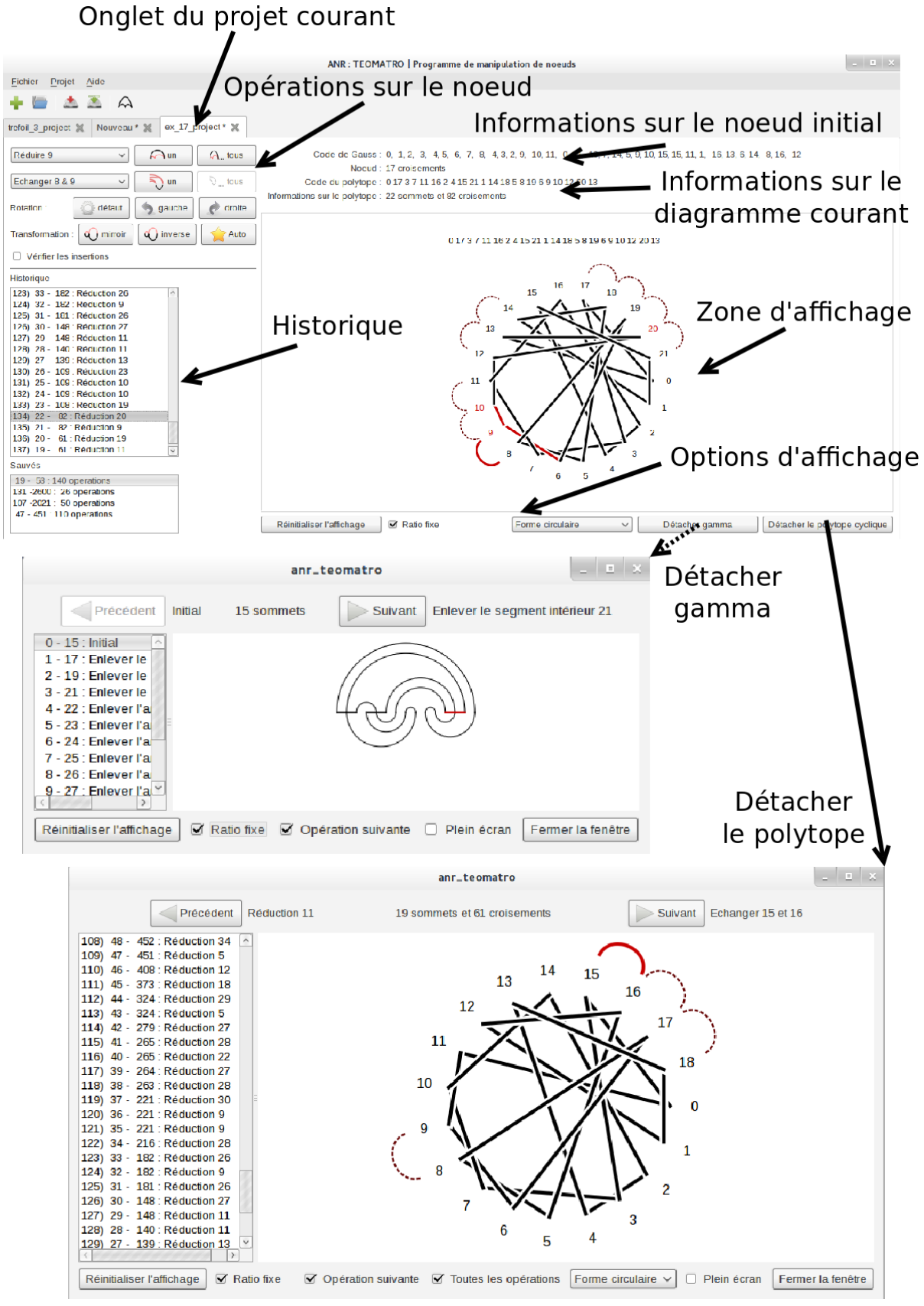
Si le projet a déjà été sauvé, il est possible de le sauver directement dans le même fichier par *Fichier -> Sauver*.

A.2.4 Fermer un projet

Un projet peut être fermé par le menu *Projet -> Fermer* ou bien en cliquant sur la croix de l'onglet correspondant.

Attention : actuellement le programme ne demande pas confirmation si le projet n'a pas encore été sauvé.

A.3 Fonctionnement d'un projet



A.3.1 La zone et les options d'affichage

Dans la zone d'affichage, la souris permet de zoomer/déplacer. Plusieurs options sont disponibles en dessous.

- Réinitialiser l'affichage : retour au zoom par défault, et placement du diagramme au centre.
- Ratio fixe : conserver les proportions hauteur/largeur, ou utiliser tout l'espace disponible quitte à déformer le diagramme.
- Forme : basculer entre la forme circulaire et classique.
- Détacher gamma : si le projet a été créé directement avec un polytope cette option n'est pas disponible. Sinon elle permet d'ouvrir une fenêtre ou l'on peut visualiser la transformation du nœud au polytope.
- Détacher le polytope cyclique : ouvrir une fenêtre avec uniquement l'historique en cours, permettant plusieurs options d'affichage.

A.3.2 Les informations sur le nœud et le polytope

Si le projet a été créé avec un code de Gauss du nœud initial, ce code est présenté. En dessous le code du polytope est affiché, avec le nombre de sommets et de croisements de celui-ci.

A.3.3 Les opérations

La première ligne concerne les réductions. Le premier menu déroulant permet de choisir parmi les sommets réductibles, le bouton suivant permet de réduire le sommet affiché dans ce menu, et le dernier bouton permet de réduire tous les sommets tant qu'il en reste un réductible.

La seconde ligne concerne les échanges de sommets. Le premier menu déroulant permet de choisir parmi les sommets échangeables. Le bouton suivant permet d'échanger les deux sommets précédemment sélectionnés, et le dernier permet d'effectuer tous les échanges dits "bons" (i.e. : réduisant le nombre de croisements ou faisant apparaitre une réduction).

La troisième ligne concerne les rotations. Le premier bouton permet de revenir à une position par défaut, le second permet d'effectuer une rotation vers la gauche, et le dernier permet d'effectuer une rotation vers la droite.

La quatrième ligne concerne plusieurs autres opérations :

mirroir : Transforme le nœud en son mirroir (et donc est susceptible de changer le type du nœud)

inversion : Une transformation assez complexe, qui permet de faire apparaitre certaines réductions (laissée dans un but expérimental)

auto : Essaye de réduire *au mieux* : en faisant des réductions et des "bons" échanges autant que possible.

La dernière ligne, concernant les insertions, apparait si on coche le bouton correspondant. On peut alors choisir où insérer un sommet dans un menu déroulant.

A.3.4 Historique

La partie supérieure de l'historique conserve toutes les opérations effectuées depuis le polytope initial.

La partie inférieure permet de stocker tous les historiques différents. Un double-clic sur l'un d'eux permet de le supprimer du projet.

A.4 Format des fichiers

A.4.1 Fichier nœud

Un fichier contenant le code de Gauss du trèfle gauche :

```
1: 123456 1
2: 6 0, -1, 2, 0, 1, 2
```

La première ligne contient des informations sur le type de fichier (`123456` signifie “fichier nœud” et `1` signifie “version 1”).

Le seconde ligne commence par deux fois le nombre de croisements (`6` = 2 fois 3 croisements) puis le code de Gauss est indiqué (avec virgule et espace comme séparateur, le signe positif étant omis).

A.4.2 Fichier polytope

Un fichier contenant le polytope à 7 sommets du trèfle gauche :

```
1: 123457 1
2: 7 0 2 4 6 1 3 5
```

La première ligne contient des informations sur le type de fichier (`123457` signifie “fichier polytope” et `1` signifie “version 1”).

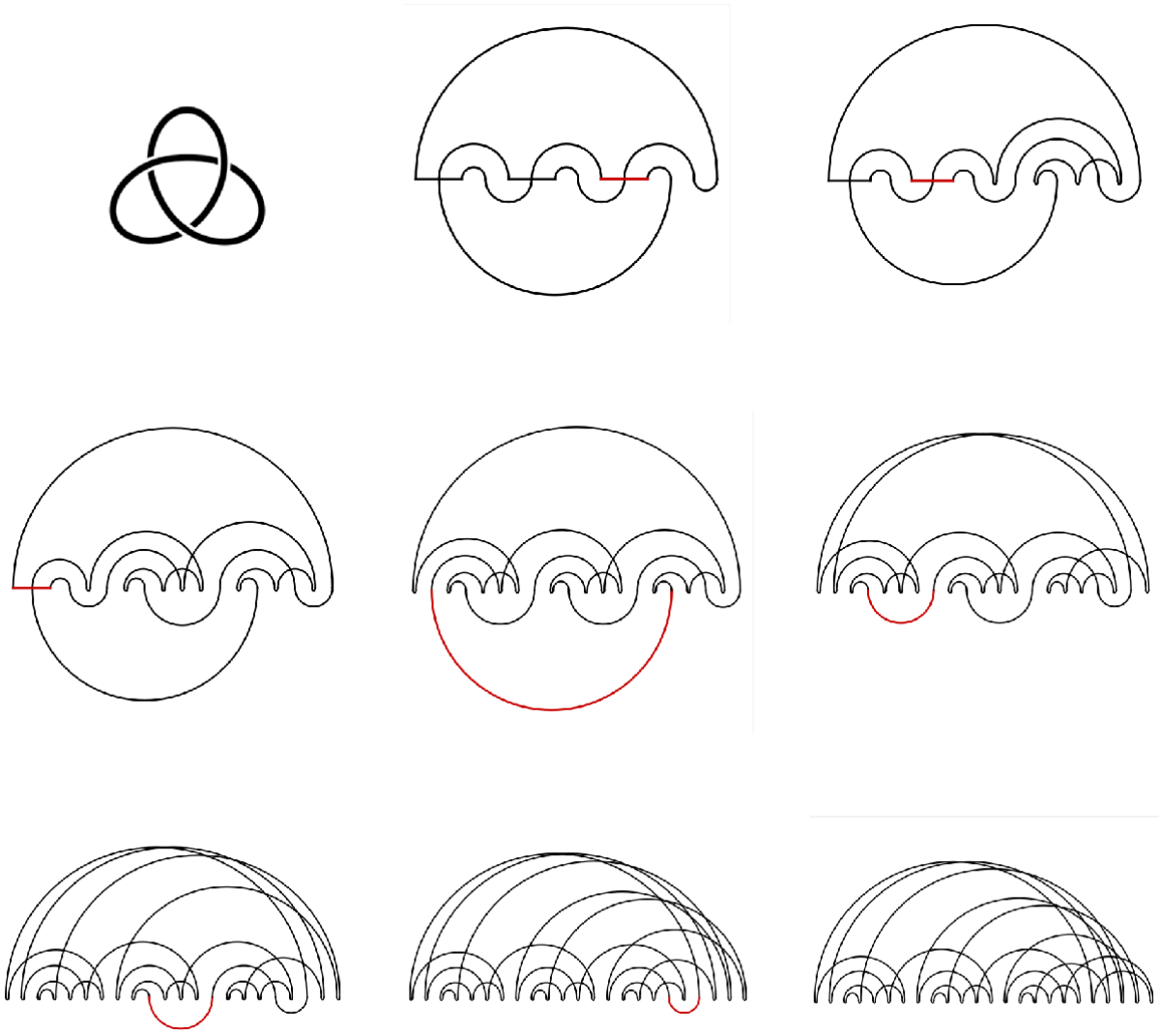
La seconde ligne commence par le nombre de sommets, puis le code du diagramme est indiqué (avec un espace comme séparateur).

A.4.3 Fichier projet

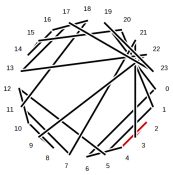
Un fichier projet n’est pas initialement destiné à être lu par l’homme. Il contient principalement le nœud initial, le polytope initial, la liste des historiques et opérations effectuées.

Annexe B

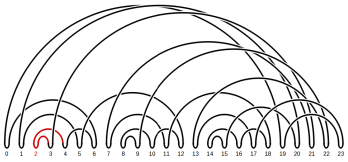
Exemple de déroulement de l'algorithme

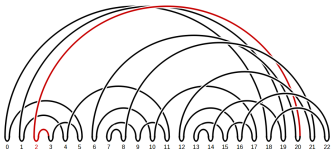


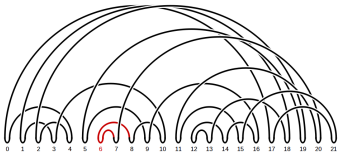
0) 24 -	48 : Initial
1) 23 -	47 : Réduction 2
2) 22 -	47 : Réduction 2
3) 21 -	46 : Réduction 6
4) 20 -	46 : Réduction 6
5) 19 -	45 : Réduction 9
6) 18 -	44 : Réduction 9
7) 17 -	44 : Réduction 9
8) 16 -	39 : Réduction 9
9) 15 -	27 : Réduction 9
10) 14 -	27 : Réduction 9
11) 13 -	17 : Réduction 9
12) 12 -	17 : Réduction 8
13) 11 -	17 : Réduction 8
14) 10 -	12 : Réduction 5
15) 9 -	11 : Réduction 4
16) 8 -	11 : Réduction 7

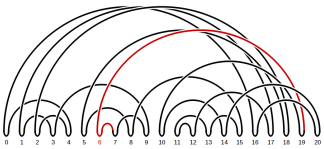


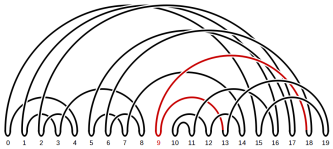
=

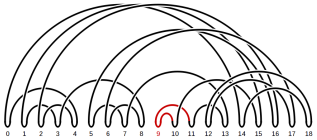


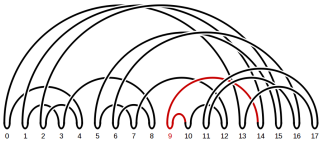


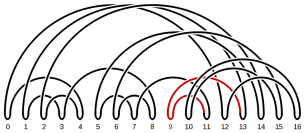


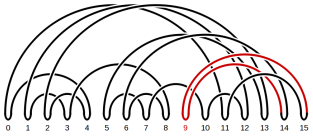


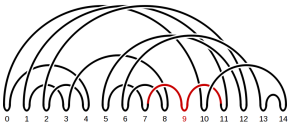


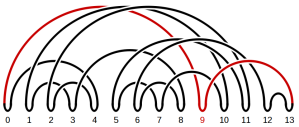


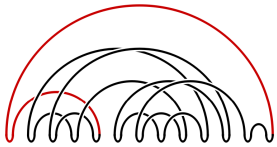


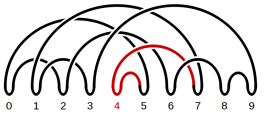


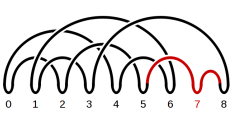


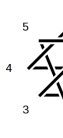




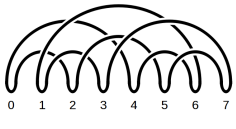








=



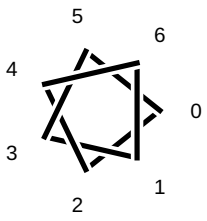
Annexe C

Exemple de résultats du générateur

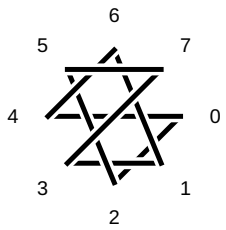
Les pages suivantes présentent certains résultats du générateur. Il s'agit de tous les nœuds sur le polytope cyclique avec 7 à 11 sommets, pour lesquels il n'existe aucune réduction connues, et aucun doublon à rotation près.

Les échanges possibles sont mis en évidence, et il peut éventuellement y avoir des diagrammes équivalents à un échange près dans la liste.

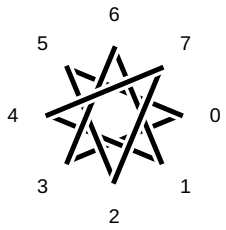
0 2 4 6 1 3 5



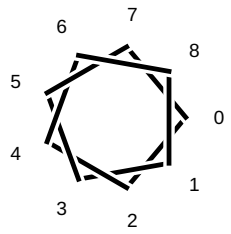
0 2 5 7 3 1 6 4



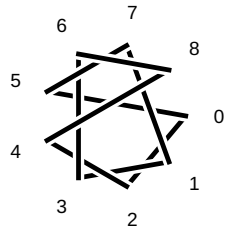
0 3 6 1 4 7 2 5



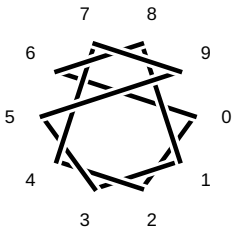
0 2 4 6 8 1 3 5 7



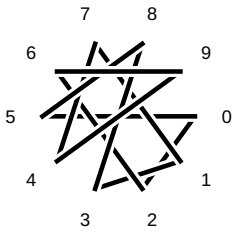
0 2 4 8 6 3 1 7 5



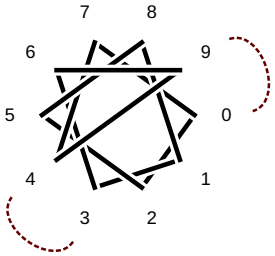
0247953186



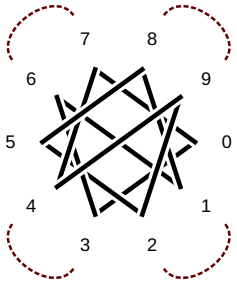
0269471385



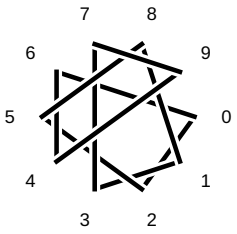
0258136947



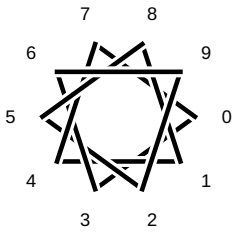
0361852947



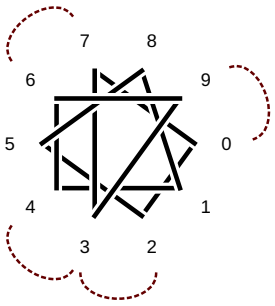
0258137946



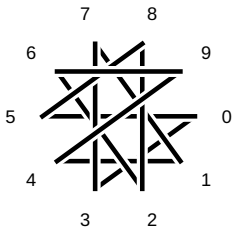
0369258147



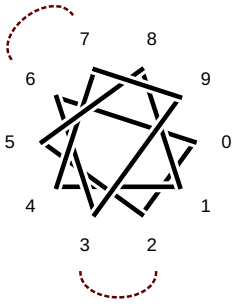
0258146937



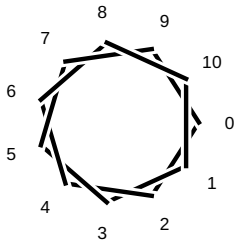
0371496285



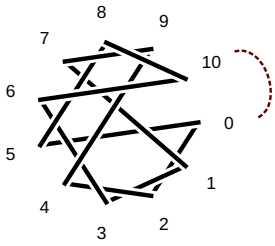
0258147936



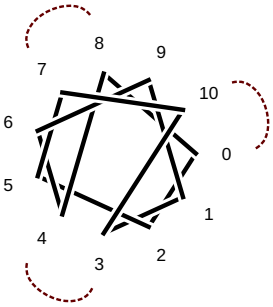
0 2 4 6 8 10 1 3 5 7 9



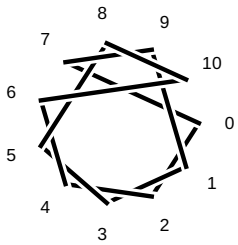
0 2 4 9 7 1 3 6 10 8 5



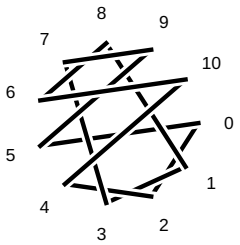
0 2 5 7 10 3 1 9 6 4 8



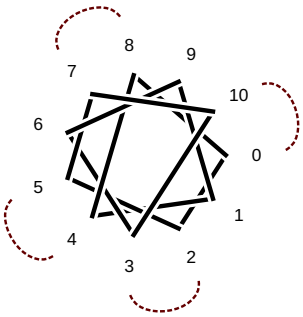
0 2 4 6 10 8 5 3 1 9 7



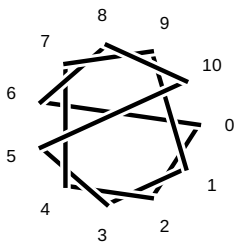
0 2 4 10 6 8 1 3 7 9 5



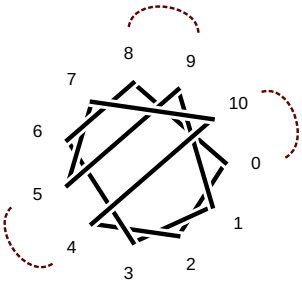
0 2 5 7 10 3 6 9 1 4 8



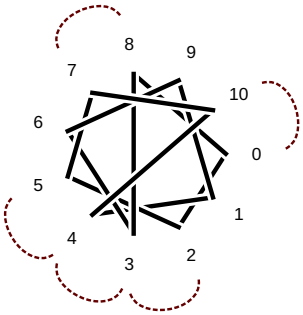
0 2 4 7 9 1 3 5 10 8 6



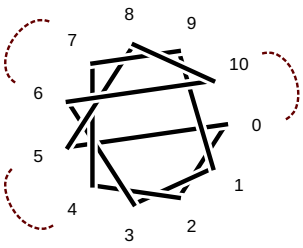
0 2 4 10 7 5 9 1 3 6 8



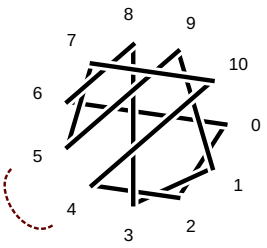
0 2 5 7 10 4 1 9 6 3 8



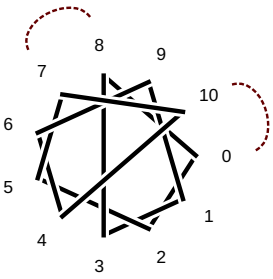
0 2 4 7 9 1 3 6 10 8 5



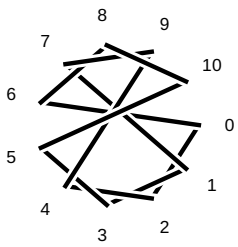
0 2 4 10 7 5 9 1 3 8 6



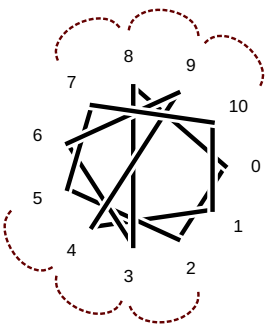
0 2 5 7 10 4 6 9 1 3 8



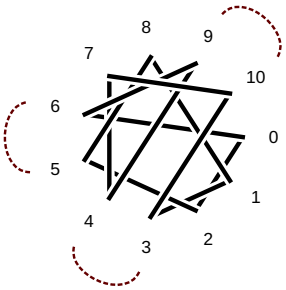
0 2 4 9 7 1 3 5 10 8 6



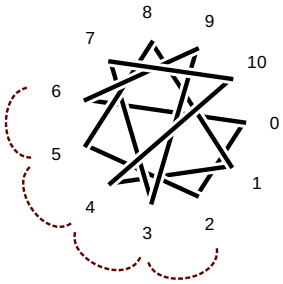
0 2 5 7 10 1 4 9 6 3 8



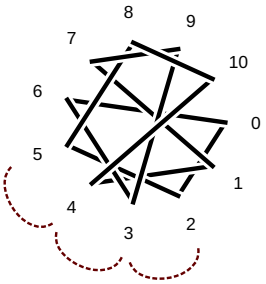
0 2 5 8 1 3 10 7 4 9 6



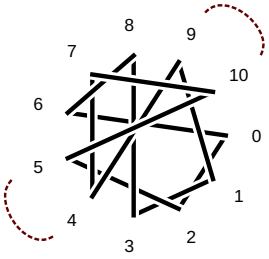
0 2 5 8 1 4 10 7 3 9 6



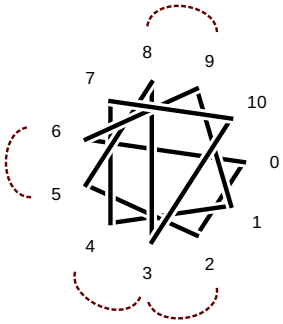
0 2 5 8 10 4 1 7 9 3 6



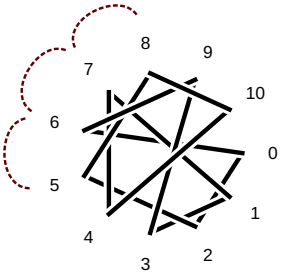
0 2 5 10 7 4 9 1 3 8 6



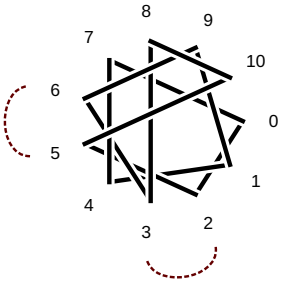
0 2 5 8 3 10 7 4 1 9 6



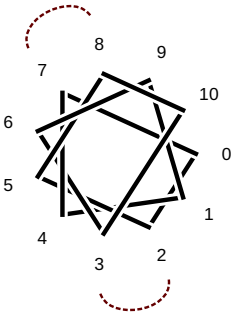
0 2 5 8 10 4 7 1 3 9 6



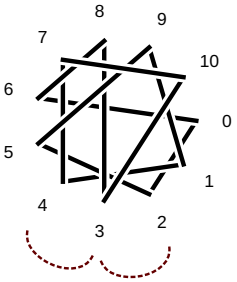
0 2 5 10 8 3 6 9 1 4 7



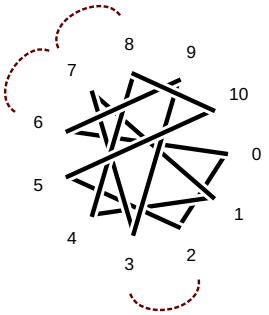
0 2 5 8 10 3 6 9 1 4 7



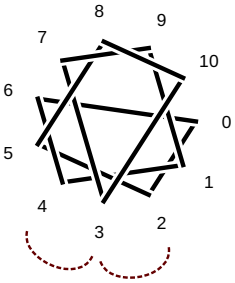
0 2 5 9 1 4 7 10 3 8 6



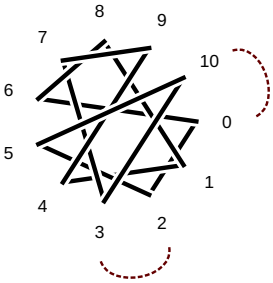
0 2 5 10 8 4 1 7 3 9 6



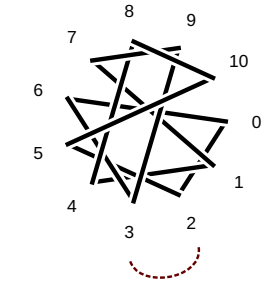
0 2 5 8 10 3 7 9 1 4 6



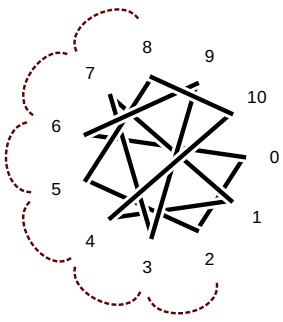
0 2 5 10 3 7 9 4 1 8 6



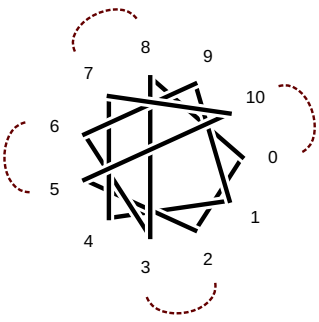
0 2 5 10 8 4 1 7 9 3 6



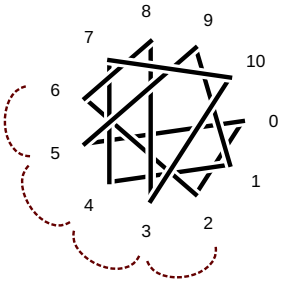
0 2 5 8 10 4 1 7 3 9 6



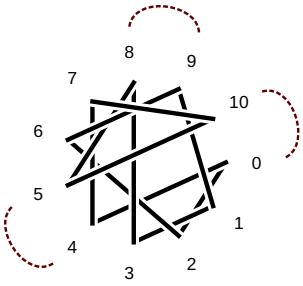
0 2 5 10 7 4 1 9 6 3 8



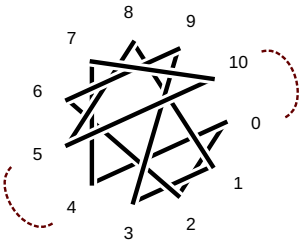
0 2 6 8 3 10 7 4 1 9 5



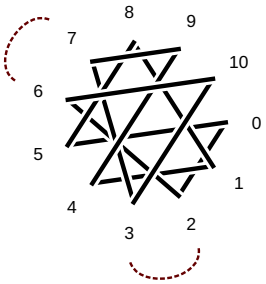
0 2 6 9 1 3 8 5 10 7 4



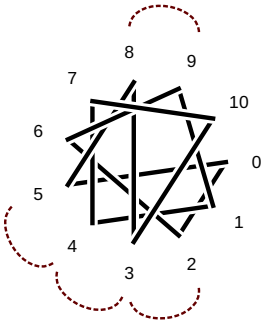
0 2 6 9 3 1 8 5 10 7 4



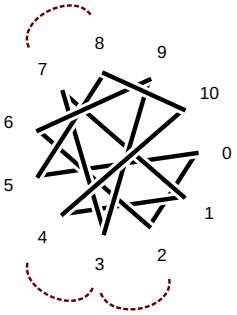
0 2 6 10 3 7 9 4 1 8 5



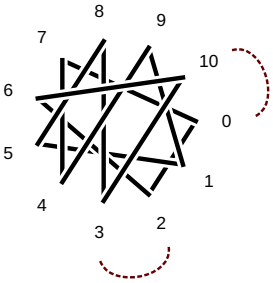
0 2 6 9 1 4 7 10 3 8 5



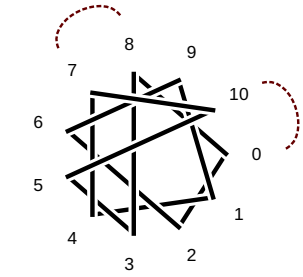
0 2 6 9 3 7 1 4 10 8 5



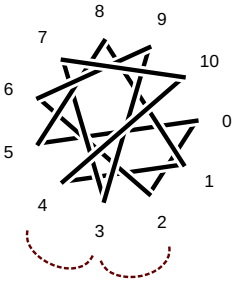
0 2 6 10 3 8 5 1 9 4 7



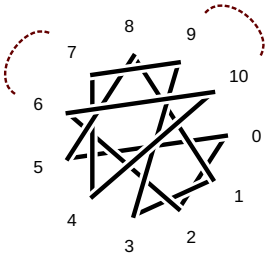
0 2 6 9 1 4 7 10 5 3 8



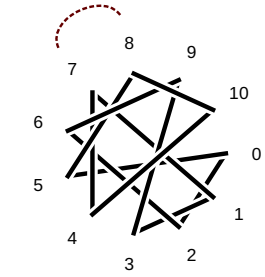
0 2 6 9 3 7 10 4 1 8 5



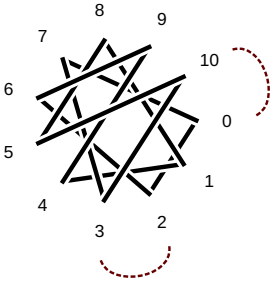
0 2 6 10 4 7 9 3 1 8 5



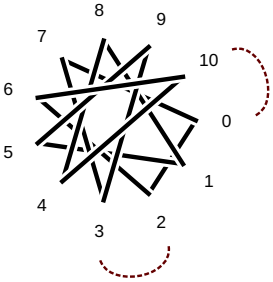
0 2 6 9 3 1 7 4 10 8 5



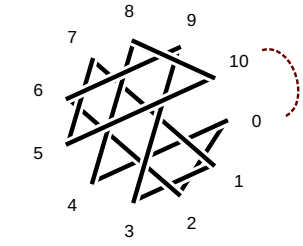
0 2 6 9 4 1 8 5 10 3 7



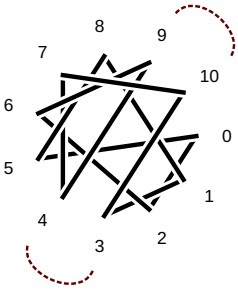
0 2 6 10 4 8 1 5 9 3 7



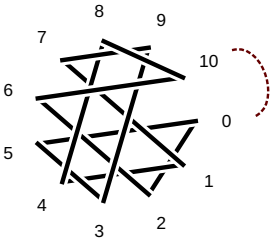
0 2 6 9 3 1 7 5 10 8 4



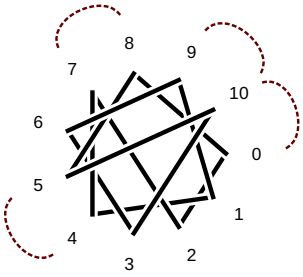
0 2 6 9 4 7 10 3 1 8 5



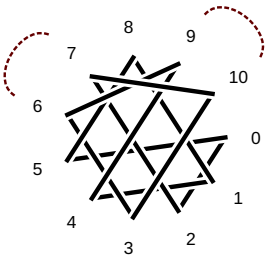
0 2 6 10 8 4 1 7 9 3 5



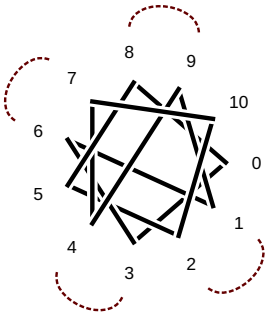
0 2 7 4 1 9 6 3 10 5 8



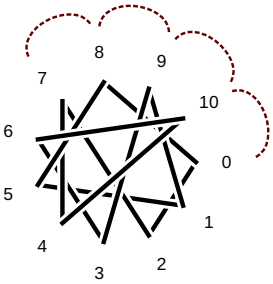
0 2 7 10 3 6 9 4 1 8 5



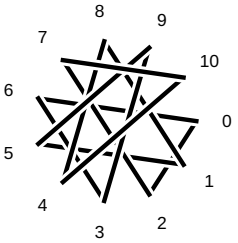
0 3 6 1 9 4 7 10 2 5 8



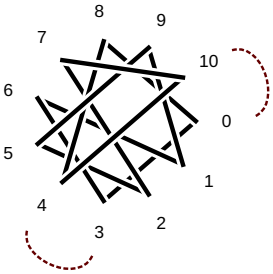
0 2 7 4 10 6 3 9 1 5 8



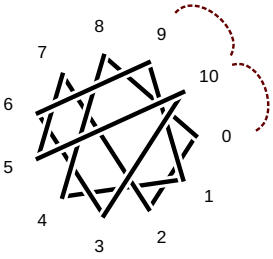
0 2 7 10 4 8 1 5 9 3 6



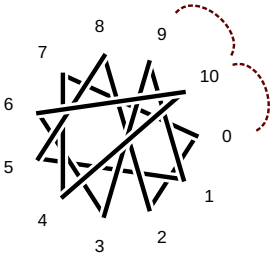
0 3 6 1 9 5 2 7 10 4 8



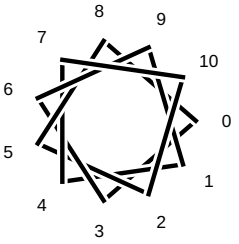
0 2 7 5 10 3 6 9 1 4 8



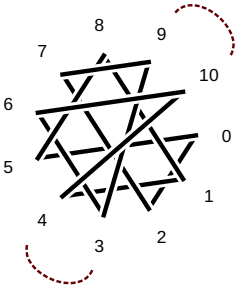
0 2 8 5 1 9 3 6 10 4 7



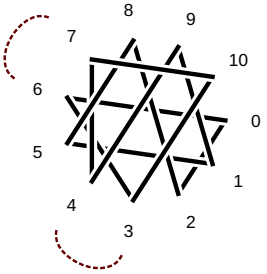
0 3 6 9 1 4 7 10 2 5 8



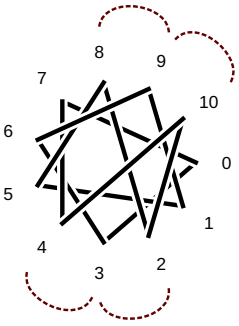
0 2 7 9 3 6 10 4 1 8 5



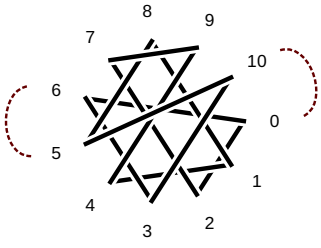
0 2 8 5 1 9 4 7 10 3 6



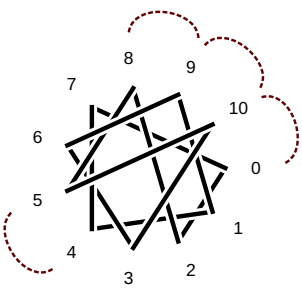
0 3 6 9 1 5 8 2 10 4 7



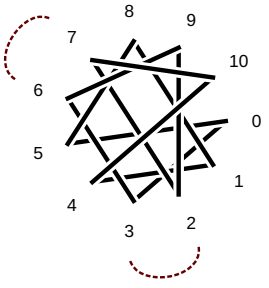
0 2 7 9 4 1 8 5 10 3 6



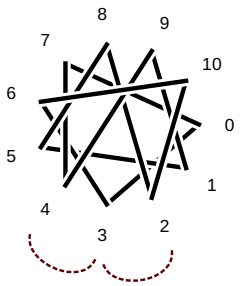
0 2 8 5 10 3 6 9 1 4 7



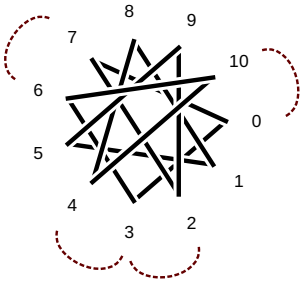
0 3 6 9 2 7 10 4 1 8 5



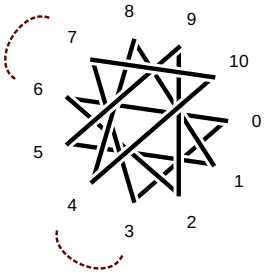
0 3 6 10 2 8 5 1 9 4 7



0 3 6 10 4 8 1 5 9 2 7



0 3 7 10 4 8 1 5 9 2 6



0 4 8 1 5 9 2 6 10 3 7