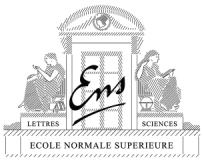


École Normale Supérieure

Rapport de stage

Eternity II et variantes

Auteur:
Ludovic Patey

Maître de stage:
Sylvain Gravier

8 septembre 2010

Avant-propos

Au sujet de cette étude

Ce document est une étude thématique du jeu Eternity II et de problèmes liés, réalisé au cours d'un stage de 3^e année de license à l'Ecole Normale Supérieure. En raison de la courte période du stage (2 mois), de nombreux points n'ont pu être approfondis et les démonstrations sont pour la plupart peu détaillées. La vérification minutieuse de leur validité est laissée aux bons soins du lecteur.

Ce que le lecteur peut espérer en attendre

À travers cette étude, le lecteur trouvera la preuve de NP-complétude d'un grand nombre de problèmes gravitant autour de la généralisation du jeu de combinatoire Eternity II, notamment des relaxations de contraintes et certaines spécialisations (ajouts de contraintes sur les pièces, notamment le nombre de motifs par pièces).

Le but de cette étude est de donner au lecteur une compréhension plus profonde des raisons intrinsèques de la complexité de ce jeu, et de le guider dans le choix d'une stratégie de résolution en fonction des résultats présentés dans ce document.

Organisation du document

Le présent document a été divisé en deux parties : la première est le compte-rendu de stage à destination de l'Ecole Normale Supérieure et ne présente pas d'intérêt particulier pour un lecteur extérieur. La seconde est une étude thématique du problème et de ses variantes. Enfin, des annexes résument les résultats obtenus et introduisent les problèmes utilisés dans le document et dont la complexité est supposée connue.

Version

Ce document a été réalisé entre Juin 2010 et Août 2010.

Remarques

Pour toute demande de précision, remarque ou rapport d'erreur, n'hésitez pas à me contacter à l'adresse électronique suivante :

`<ludovic.patey@ens.fr>`

Table des matières

1 Introduction

1.1 Le jeu Eternity II

Eternity II est un puzzle créé par Christopher Monckton et commercialisé par la société TOMY. Ce puzzle est composé de 256 pièces comportant un motif à chaque bord, et devant paver un plateau de taille 16×16 de manière à ce que les côtés adjacents des pièces correspondent. Un des motifs joue un rôle particulier : il doit former le contour du plateau.

1.2 Du jeu au problème scientifique

Outre l'aspect ludique du jeu, Eternity II est un véritable problème de combinatoire extrêmement difficile à résoudre et dont la récompense promise son auteur est un des éléments révélateurs.

Ce problème peut être vu comme une instance finie du problème du domino introduit par Berger dans les années 60 [1] et prouvé comme indécidable. Eternity II étant une instance particulière finie, son étude de complexité ne présente pas d'intérêt particulier sachant que l'algorithme le résolvant est en $\mathcal{O}(1)$. Nous allons donc nous intéresser à sa généralisation en un jeu de taille $m \times n$.

1.3 État de l'art

Depuis sa création, Eternity II a suscité l'intérêt de la communauté scientifique, certains étudiant le problème dans sa globalité [2] d'autres essayant différents paradigmes de résolution [3], d'autres enfin innovant par des hybridations des précédents paradigmes [4]. Des projets de calcul distribués ont vu le jour, notamment un projet BOINC eternity2.net, lequel a finalement annoncé sa dissolution en 2007.

À ce jour, aucune solution n'a été trouvée pour l'instance particulière, de même qu'aucune solution efficace n'a été apportée pour le problème général. L'auteur du jeu a cependant stipulé dans les règles du jeu que le détenteur d'une solution ne devrait pas révéler sa réussite sous peine de ne plus pouvoir revendiquer le prix, afin de pouvoir continuer à vendre son jeu avant le dépouillement des résultats qui a lieu chaque année en décembre. Il est donc possible qu'à l'heure actuelle, une solution soit connue.

Première partie

Rapport de stage

Cette partie apporte des méta-considérations, dans le sens où l'on n'aborde pas les résultats du stage, mais ses apports et les difficultés rencontréees.

2 Problèmes rencontrés lors du stage

Les problèmes suivants peuvent sembler triviaux tels qu'énoncés, car je les ai épurés pour retirer l'essence de la problématique. Ils ont été découverts dans des démonstrations erronées de ma part. À chaque démonstration invalidée a été trouvé une nouvelle démonstration corrigeant le problème, mais nécessitant parfois plusieurs jours de travail.

2.1 Réductions non polynomiales

La démonstration de la NP-complétude de **ETER2** provient de [5] et utilise une réduction de **3-PARTITION** avec un encodage unaire, opération permise car si l'on impose que le domaine des entiers du multi-ensemble aie une valeur majorée par un polynôme en la taille du multi-ensemble, le problème reste NP-complet.

Je n'avais pas pris la peine de vérifier la validité de l'encodage unaire pour **SUBSET-SUM** et **PARTITION**, j'avais également utilisé un encodage unaire, or il se trouve que pour ces problèmes, il existe un algorithme pseudo-polynomial permettant de les résoudre. Mes réductions n'étaient donc pas valides.

La solution adoptée a été d'essayer, au lieu de faire une correspondance par le nombre d'occurrence de 1, de faire une correspondance digit par digit, en s'assurant que l'on n'aie pas de problème de retenue, et c'est ce qui a motivé les rafinements **binary-digit SUBSET-SUM** et **binary-digit PARTITION**.

2.2 Nécessité de la connexité dans un graphe eulérien

Pour qu'un graphe soit eulérien, il est nécessaire qu'il soit connexe. Présentée ainsi, cette propriété est évidente. Mon erreur venait du fait que je considérais que dans un graphe eulérien de n arêtes, s'il existe un cycle de taille m, alors il existe un cycle de taille $n - m$, prétextant que le graphe obtenu en retirant les arêtes du cycle était encore eulérien. Il manquait la vérification de la connexité, invalidant parfois certains résultats.

Une autre question posée était "étant donné un graphe de n arêtes, s'il existe une partition en 2 cycles de $\frac{n}{2}$ arêtes, est-ce que pour tous les cycles $\mathcal{C}$ de $\frac{n}{2}$ arêtes, $G \setminus \mathcal{C}$ est connexe, et donc eulérien ?". La réponse est probablement pas, car cela impliquerait qu'une approche gloutonne sur les graphes cubiques permettrait de tester si le graphe est hamiltonien, donnant ainsi un algorithme polynomial à un problème NP-complet et montrerait que $P = NP$. Outre le fait que cela soit un problème ouvert, des contre-exemples aux algorithmes gloutons ont probablement été trouvés avant de chercher à démontrer que le problème était NP-complet.

2.3 Terminologie différente

Par défaut, j'ai considéré qu'un chemin et un cycle n'étaient pas nécessairement élémentaires. La plupart des articles sur le sujet sous-entendent l'élémentarité. J'ai ainsi cru que le problème **directed EVEN-WALK** était NP-complet en le nommant **directed EVEN-CYCLE** alors que dans le cas non élémentaire, le problème est dans P [6], rendant inutile une réduction.

2.4 Réductions de Turing vs réductions de Karp

Pendant les cours, seules les réductions de Karp ont été présentées, je n'avais pas envisagé l'existence d'autres types de réduction. P étant trivialement close par réduction de Turing, j'ai été naturellement amené à assimiler

les différents types de réduction. Plus tard, en apprenant l'existence d'un certain nombre de réductions et les conjectures portant sur leurs éventuelles distinctions concernant les ensembles C-complets, j'ai repris un certain nombre de réductions pour les transformer en réductions de Karp.

Actuellement, il y a 2 réductions de Turing dans l'étude, mais des réductions de Karp alternatives ont également été fournies. Les problèmes démontrés comme NP-complet dans ce document le sont donc tous sous réduction de Karp.

3 Apport du stage

Ce stage m'a permis d'approfondir mes connaissances dans la théorie de la complexité : J'ai été confronté à des problèmes obtenus à partir de problèmes NP-complets pour lesquels on retirait toutes les instances d'un ensemble de longueurs données. Ces problèmes étant de bons candidats à être dans NPI sous l'hypothèse que $P \neq NP$, cela m'a amené à regarder des preuves du théorème de Ladner, dont une notamment se basait sur un langage proche de celui étudié. Les études en complexité ont également entraîné la nécessité d'apprendre la distinction entre les différents types de réduction.

En outre, le grand nombre de problèmes à réduire m'a obligé à me pencher sur les problèmes existants et m'a fait ainsi acquerir une culture générale de la complexité des problèmes de la théorie des graphes.

4 Remerciements

Je tiens à remercier Sylvain Gravier pour avoir accepté de me prendre en stage sur un sujet comme celui-ci, et pour avoir prêté une oreille attentive lors de mes démonstrations qui étaient pour la plupart formulées de manière confuses.

Deuxième partie

Étude thématique d'Eternity II

5 Conventions, terminologie et formalisations

5.1 Terminologie de la théorie des graphes

Définition 5.1 (Graphe, multigraphe et pseudo-graphe)**.**

- *Un **graphe** est la donnée d'un couple (V, E) où V est un ensemble fini de sommets et E un ensembles de paires de sommets, appelés arêtes, sans boucles (ie. $\forall v \in V$, $\{v, v\} \notin E$)*
- *Un **multigraphe** est la donnée d'un couple (V, E) où V est un ensemble fini de sommets et E un multiensemble de paires de sommets sans boucles.*
- *Un **pseudographe** est la donnée d'un couple (V, E) où V est un ensemble fini de sommets et E un multiensemble de paires de sommets.*

Le dual orienté (ie. **graphe orienté**, **multigraphe orienté** et **pseudographe orienté**) s'obtient à partir de la définition précédente en remplaçant *paire* par *couple* et *arête* par *arc*.

Définition 5.2 (Chemin, cycle)**.**

- *Un **chemin** (walk) est la donnée d'une séquence finie $v_1, e_1, v_2, e_2 \cdots, v_n$ où les v_i sont des sommets et les e_i des arcs incidents à v_i et v_{i+1}. Un chemin est **simple** (trail) si de plus les arêtes sont deux à deux distinctes. Un chemin est dit **élémentaire** (path) si de plus les sommets sont deux à deux distincts. La **longueur** d'un chemin est son nombre d'arêtes.*
- *Un **cycle** (closed walk) est un chemin tel que $v_1 = v_n$, à permutation circulaire près. Le cycle est **simple** (closed trail) si le chemin est simple. Le cycle est dit **élémentaire** (cycle) si de plus le chemin est élémentaire. La **longueur** d'un cycle est la longueur d'un des chemins correspondants.*

Dans le cas orienté, on emploiera le terme *circuit* à la place de *cycle*.

Remarque : *Par défaut, les chemins seront considérés comme simples, tout comme les cycles. La nomenclature des problèmes étant en anglais, il gardent leur sens anglo-saxon. Ex : Dans **EVEN-CYCLE**, "CYCLE" est à interpréter comme "cycle élémentaire".*

5.2 Nomenclature des problèmes

Étant donnés deux problèmes **P1** et **P2**, nous appelerons problème **P2-FROM-P1** le problème consistant à décider s'il existe une configuration acceptante dans **P1** $\cap$ **P2** à partir d'une de **P1**.

Étant donné un problème **P** de la théorie des graphes consistant à tester l'existence d'une structure dans le graphe, nous appellerons **LOCAL P** le problème **P** augmenté de la contrainte de contenir une arête préalablement distinguée.

Par exemple, si **EVEN-CYCLE** consiste à tester l'existence d'un cycle de longueur paire dans un graphe, **LOCAL EVEN-CYCLE** teste l'existence d'un cycle de longueur paire passant par une arête distinguée.

Étant donné un problème **P** de la théorie des graphes, **directed P** est la version du problème pour les graphes orientés et **undirected P** celle pour les graphes non orientés.

Enfin, étant donné un problème **k P** où k est un paramètre, **k P,** $\forall k$ est la classe de problèmes où k est fixé.

5.3 Formalisation en problèmes

Afin d'étudier formellement le problème d'Eternity, nous allons introduire un certain nombre de problèmes, liés à celui d'Eternity par relachement de contraintes. Ils seront définis comme des problèmes de décision, mais nous évoquerons parfois leurs duals de construction en le précisant le cas échant.

FIGURE 5.1 – Représentations d'une même pièce

Les problèmes suivants vont requérir la notion de **jeu de pièces** à motifs dans un ensemble E, qui peut être définit formellement comme un multi-ensemble à valeurs dans E^4 quotienté par les permutations circulaires. Graphiquement, une pièce peut être représentée comme suit :

Nous représenterons en pratique une pièces possédant les motifs A,B,C et D dans cet ordre par (A, B, C, D). Lorsque les motifs sont créés pour les réductions, nous les noterons en minuscules, et ils seront à considérer comme uniques. Si par hasard, il faut considérer les pièces comme orientées, la convention est (haut,droite,bas,gauche).

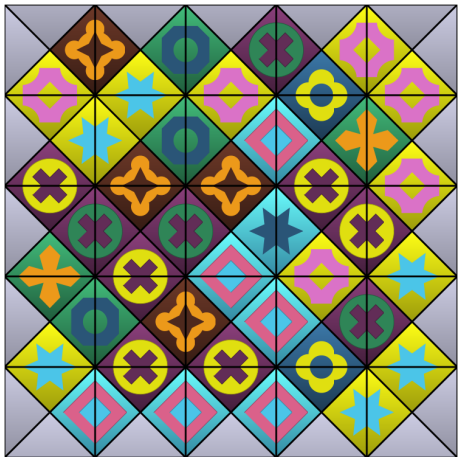
L'étude de ces problèmes est réalisée avec l'approche suivante :
- détermination de la complexité du problème et de ses variantes,
- réduction (de Turing) du problème de construction au problème de décision
- études éventuelles de problèmes liés présentant un intérêt intrinsèque ou relatif à la recherche d'une solution au jeu.

5.3.1 Le problème ETER2

Le problème de décision **ETER2** accepte un jeu de pièces et un couple (m, n) s'il existe un pavage de dimensions $m \times n$ tel que pour chaque couple de pièces adjacentes, les motifs sur le côté adjacent soient identiques. De plus, il existe 1 motif distingué dit "de bordure", caractérisant les côtés non-adjacents des pièces, c'est à dire le contour du rectangle $m \times n$.

Une **configuration** dans ce problème est la donnée d'un ensemble de triplets (pièce $\times$ position $\times$ rotations). On appelle **signature** d'une configuration le motif général obtenu en plaçant les pièces suivant le plateau. La fonction **signature** n'est pas injective.

FIGURE 5.2 – Représentation d'une configuration acceptante pour le problème **ETER2**



5.3.2 Le problème n-ROT

Le problème de décision **n-ROT** accepte un ensemble de pièces possédant $2n$ côtés s'il existe une configuration de rotation de chaque pièce, de telle sorte que pour chacun des n axes, la somme des motifs dans une direction soit égalem à la somme des motifs dans la direction opposée.

Un cas particulièrement intéressant est celui **2-ROT** car il utilise le même jeu de pièces que **ETER2**. Ce problème est obtenu à partir d'**ETER2** un relachement des contraintes de position. Toute solution d'**ETER2** est donc solution de **2-ROT**.

Une configuration est alors la donnée d'un ensemble de couples (pièce × rotations).

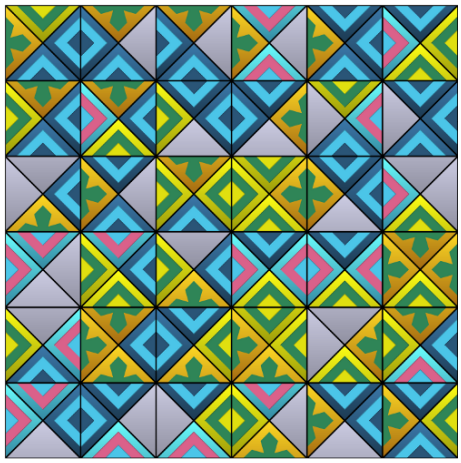
FIGURE 5.3 – Représentation d'une configuration acceptante pour le problème **2-ROT**

5.3.3 Le problème ETER2SET

Le problème de décision **ETER2SET** accepte un ensemble de pièces et deux entiers naturels m et n s'il existe une grille de dimensions $m \times n$ de telle sorte que chaque pièce adjacente partage un motif. Ce problème est donc un autre relachement de contraintes par rapport au problème **ETER2**.

Une configuration est la donnée d'un ensemble de couples (pièce × position).

FIGURE 5.4 – Représentation d'une configuration acceptante pour le problème **ETER2SET**



6 Quelques problèmes préalables

6.1 Raffinement de SUBSET-SUM et PARTITION

Une première étape consiste à essayer de classifier les problèmes soulevés pour déterminer vers lequel s'orienter plus précisément. Nous allons avoir besoin de quelques raffinements des problèmes connus **SUBSET-SUM** et **PARTITION**.

Définition 6.1. *Étant donné* ***P****, un problème de décision sur un multi-ensemble d'entiers,* ***binary-digit P*** *est une variante de* ***P*** *où l'on impose que les nombres aient une représentation en base* $\Omega(n)$ *(*n *est le cardinal du multi-ensemble) telle que chaque digit soit 0 ou 1.*

Théorème 6.1. ***binary-digit SUBSET-SUM*** *est NP-complet.*

Démonstration. Nous allons réduire **monotone One-in-three 3SAT** à **binary-digit SUBSET-SUM**.

Soit $I = (C_1, \cdots, C_n)$ une instance de **monotone One-in-three 3SAT**, possédant m variables $(x_1, \cdots, x_m)$. Soit A une matrice de taille $m \times n$ définie par

$$a_{i,j} = \begin{cases} 1 \text{ si } x_i \in C_j \\ 0 \text{ sinon} \end{cases}$$

Chaque ligne de la matrice A peut être vue comme un nombre en base $k = \Omega(m)$. Soit r le nombre dont la représentation en base k est $\overbrace{11\cdots11}^{n \text{ fois}}{}_m$.

S'il existe une valuation telle que l'instance soit satisfiable, la somme des nombres associée aux variables assignées à 1 sera égale à r. S'il existe un sous-ensemble des nombres tels que la somme soit égale à r, alors nous pouvons assigner les variables associée à 1 et les autres à 0, et nous aurons une valuation satisfiant l'instance car chaque clause sera vérifiée.

Cette réduction est une logspace-réduction car il suffit de stocker sur la bande de travail la position en binaire de la variable courante dont on écrit le nombre sur la bande de sortie, et la position en binaire de la clause courante pour laquelle on teste la présence de la variable courante, prenant ainsi un espace logarithmique sur la bande de travail. □

Théorème 6.2. ***binary-digit PARTITION*** *est NP-complet.*

Démonstration. Nous allons réduire **monotone One-in-three 3SAT** à **binary-digit PARTITION**, en créant la même matrice A que dans la démonstration de NP-complétude de **binary-digit SUBSET-SUM**, mais en ajoutant r comme colonne à la fin.

La somme des nombres est égale au nombre dont la représentation en base k est $\overbrace{44\cdots44}^{n \text{ fois}}{}_m$. S'il existe une partition de S en deux ensembles de même somme, alors soit S' le sous-ensemble de la partition contenant r. $S' \setminus \{r\}$ a pour somme r, et nous obtenons une valuation satisfiant l'instance **monotone One-in-three 3SAT** en assignant à vrai les variables associées aux nombres de $S' \setminus \{r\}$.

Réciproquement, soit une valuation satisfiant l'instance, alors l'ensemble des nombres associés aux variables assignéees à vrai a pour somme r, et si l'on ajoute r, on obtient $2r$, soit un sous-ensemble de somme égal à la moitié de la somme totale. Il existe donc une partition en deux sous-ensembles de même somme.

Pour les mêmes raisons que précédemment, la réduction est une logspace-réduction.

Remarque : Cette démonstration repose sur la réduction canonique de **SUBSET-SUM** à **PARTITION** consistant à ajouter l'élément $S - 2m$ où S est la somme totale et m la somme du sous-ensemble que l'on cherche à obtenir. En l'occurrence, ici, $S = 3r$, $m = r$ et $S - 2m = r$, donc nous avons ajouté r. □

6.2 Cycles dans un graphe

Beaucoup de problèmes concernant Eternity II gravitent autour de l'existence de cycles, de partitions des arêtes en cycles et autres variantes de la théorie des graphes. Nous allons démontrer ici quelques résultats utiles qui nous serviront pour la plupart dans les réductions des problèmes liés à Eternity, notamment la complexité de **4-ETER2-BORDERS**.

Le problème de décision général qui, étant donné un graphe non orienté $G = (V, E)$ et un entier n, décide s'il existe un cycle de longueur n est NP-complet, car pour $n = \#V$, les cycles de longueur n pour les graphes cubiques sont les cycles hamiltoniens, problème prouvé NP-complet pour ce type de graphes. [7].

Nous allons nous intéresser plus précisément au cas de l'existence de cycles de longueurs paire ou impaire, et de cycles de longueur $\frac{m}{2}$ où $m = \#E$.

6.2.1 Cycles de longueur paire dans un graphe non orienté

Problème (**undirected EVEN-CYCLE**)
Étant donné un graphe non orienté, existe-t-il un cycle élémentaire de longueur paire ?

Problème (**undirected LOCAL EVEN-CYCLE**)
Étant donné un graphe non orienté et une arête, existe-t-il un cycle élémentaire de longueur paire passant par cette arête ?

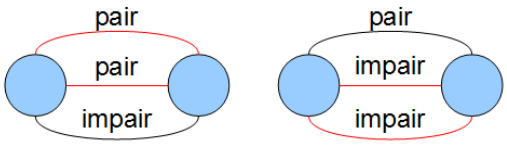
Ce problème appartient à P comme montré par [8]. Le problème consistant à savoir s'il existe un cycle élémentaire passant par une arête distinguée est encore dans P, comme le montre [6].

Proposition 6.1. *Étant donné un graphe non orienté, si deux cycles élémentaires distincts partagent au moins une arête, alors il existe un cycle élémentaire de longueur paire.*

Démonstration. Chaque cycle peut se décomposer en 2 chemins : le chemin commun aux cycles et le chemin distinct. Notons I le chemin commun, et AI et BI les cycles. AB forme donc un cycle. Alors
- Si AI ou BI est pair, alors nous avons un cycle pair.
- Si A et B sont pairs, alors AB est un cycle pair.
- Si A est impair, alors I est pair, et donc B est impair, donc AB est impair.

Nous pouvons le voir graphiquement figure 6.1.

FIGURE 6.1 – Cycles élémentaires partageant une arête. (1 ligne = 1 chemin)



□

Ainsi, si un graphe ne contient pas de cycle élémentaire de longueur paire, alors tous les cycles élémentaires du graphe sont de longueur impaire et disjoints (ie. ne partagent aucune arête). Nous pouvons donc en déduire un algorithme polynomial :

Etape 1 : On construit un ensemble de cycles élémentaires disjoints de telle sorte que le graphe oté de ces cycles soit acyclique. Pour cela, il suffit de construire itérativement l'ensemble en trouvant un cycle (problème dans P) et en retirant ses arêtes du graphe. Cet ensemble contient au plus $\frac{m}{2}$ cycles. Si l'un des cycles est de longueur paire, alors nous avons terminé. Sinon, passer à l'étape 2.

Etape 2 : Pour chaque cycle, pour chaque couple d'arête, tester s'il existe dans le graphe un cycle passant par la première arête et ne passant pas par la seconde (problème dans P : il suffit de supprimer les arêtes, et tester l'existence d'un chemin d'un des sommets incidents vers l'autre sommet incident). Si un tel graphe existe, alors il existe un cycle élémentaire de longueur paire d'après la propriété. Sinon, il n'en n'existe pas.

Problème (**undirected EVEN-TRAIL**)
Étant donné un graphe non orienté, existe-t-il un cycle de longueur paire ?

Problème (**undirected LOCAL EVEN-TRAIL**)
Étant donné un graphe non orienté et une arête, existe-t-il un cycle de longueur paire passant par cette arête ?

Proposition 6.2. *Étant donné un graphe non orienté, si deux cycles distincts partagent au moins un sommet, alors il existe un cycle de longueur paire.*

La preuve est la même que pour la proposition précédente. L'algorithme de test est sensiblement le même : lors de la première étape, nous ne retirons pas que les arêtes du cycle, mais également les sommets. Pour la seconde étape, nous distinguons un sommet et non une arête.

Le problème de l'existence de cycle élémentaire induit de longueur paire dans un graphe non-orienté a été prouvé dans P par Conforti et Al. [9]

6.2.2 Cycles de longueur paire dans un graphe orienté

Problème (**directed EVEN-TRAIL**)
Étant donné un graphe orienté, existe-t-il un cycle de longueur paire ?

Problème (**directed LOCAL EVEN-TRAIL**)
Étant donné un graphe orienté et un arc, existe-t-il un cycle de longueur paire passant par cet arc ?

Nous allons commencer par prouver la NP-complétude du problème **directed LOCAL EVEN-CYCLE**, et pour cela, nous allons introduire le problème suivant :

Problème (**directed LOCAL ODD-CYCLE**)
Étant donné un graphe orienté et un arc, existe-t-il un cycle élémentaire de longueur impaire passant par cet arc ?

Ce problème est équivalent à **directed LOCAL EVEN-CYCLE** car pour tester l'existence d'un cycle pair passant par un arc, il suffit de scinder l'arc en 2 et tester l'existence d'un cycle de longueur impaire. La réduction de **directed LOCAL ODD-CYCLE** à **directed LOCAL EVEN-CYCLE** est strictement la même.

Théorème 6.3. ***directed LOCAL ODD-CYCLE*** *est NP-complet.*

Démonstration. Le problème est dans NP car la donnée d'un cycle de longueur impaire passant par l'arc suffit pour prouver son existence. Nous allons réduire **directed 2 DISJOINT PATH** à **directed LOCAL ODD-CYCLE**.

Soit un graphe orienté $G = (V, E)$ et 2 couples de sommets distingués (v_1, v_2) et (v'_1, v'_2). Nous allons définir un graphe G' en scindant chaque arc en 2, en ajoutant un sommet v' et les arcs (v_1, v'_1), (v'_1, v_1), (v_2, v'), (v', v_2), (v'_2, v'), (v', v'_2).

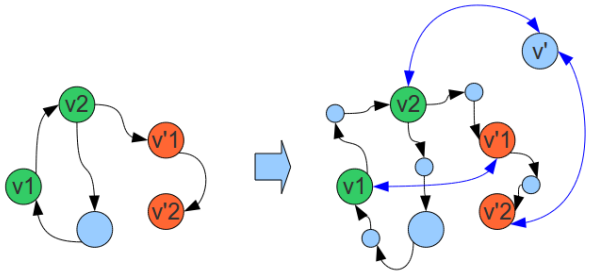
Ainsi, les cycles élémentaires de longueur impaire dans le graphe G seront les cycles élémentaires de G' passant par (v_1, v'_1) ou (v'_1, v_1).

Il existe dans G un chemin entre v_1 et v_2 et un chemin entre v'_1 et v'_2, disjoints au niveau des sommets, si et seulement s'il existe un cycle élémentaire de longueur impaire dans G' passant par (v_2, v') ou par (v', v_2).

□

Théorème 6.4. ***directed LOCAL EVEN-TRAIL*** *est NP-complet.*

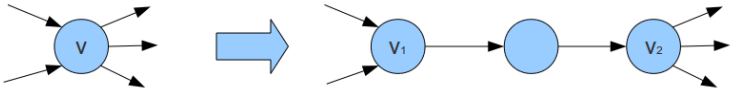
FIGURE 6.2 – Transformation d'un graphe G en G'



Démonstration. Ce problème est trivialement dans NP car la donnée d'un cycle de longueur paire suffit à prouver l'existence d'une solution. Nous allons réduire **directed LOCAL EVEN-CYCLE** à **directed LOCAL EVEN-TRAIL**.

Soit un graphe orienté $G = (V, E)$. A chaque sommet v tel que $\deg^-(v) \geq 2$ et $\deg^+(v) \geq 2$, nous allons le scinder en 2 sommets v_1 et v_2, tel que les arcs entrants aillent vers v_1, les arcs sortants partent de v_2 et les sommets v_1 et v_2 soient reliés par un chemin de longueur 2. (figure 6.3).

FIGURE 6.3 – Scission du sommet v



Les cycles passant plusieurs fois par le même sommet sont donc supprimés car plus aucun sommet n'a de degré entrant et de degré sortant > 2 en même temps. Les cycles élémentaires ne sont pas supprimés, car pour tout cycle élémentaire, pour chaque sommet de ce cycle, si celui-ci a été scindé, il existe encore un chemin entre l'arc entrant et l'arc sortant. Certains cycle ont une longueur augmentée par un nombre pair, ce qui ne change pas l'existence de cycle élémentaires de longueur paire. Le graphe obtenu aura donc un cycle de longueur paire si et seulement s'il a un cycle élémentaire de longueur paire. □

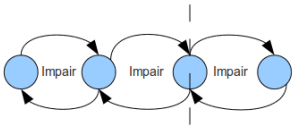
Remarque : *Le même raisonnement permet de réduire* ***directed EVEN-CYCLE*** *à* ***directed EVEN-TRAIL****.*

Nous allons maintenant voir la réduction opposée, permettant ainsi de montrer l'équivalence des deux problèmes.

Proposition 6.3. *Soit un graphe orienté contenant un cycle de longueur paire. Alors il existe un cycle de longueur paire ayant au plus 1 sommet redondant.*

Démonstration. En effet, soit il existe un cycle élémentaire de longueur paire, soit tous les cycles sont de longueur impaire, et s'il existe un cycle non élémentaire de longueur paire, il suffit de 2 "boucles" pour obtenir un cycle de longueur paire. (figure 6.4)

FIGURE 6.4 – Obtention d'un cycle de longueur paire ayant au plus un sommet redondant



□

Théorème 6.5. ***directed EVEN-TRAIL*** $\leq_m^P$ ***directed EVEN-CYCLE***

Démonstration. Soit un graphe orienté $G = (V, E)$.

Pour chaque sommet v de degré entrant au moins 2 et de degré sortant au moins 2, pour chaque paire de couples (a_1, a_2), (a_3, a_4) d'arcs entrants,sortants, nous allons définir le graphe $G_{(a_1,a_2),(a_3,a_4)}$ à partir de G en scindant le sommet intermédiaire v en 2 sommets v_1 et v_2, et en redirigeant a_1 vers v_1, a_2 depuis v_1, a_3 vers v_2 et a_4 depuis v_2.

Nous allons enfin définir le graphe G' en regroupant G et les $G_{(a_i,a_j),(a_k,a_l)}$.

Il existe un cycle élémentaire pair dans G' si et seulement s'il existe un cycle pair dans G.

La réduction est polynomiale car le nombre de composantes $G_{(a_i,a_j),(a_k,a_l)}$ est $\sum_{v\in V} 2\binom{\deg^-(v)}{2}\binom{\deg^+(v)}{2}$. La validité de la réduction repose sur la proposition précédente. En scindant un sommet, nous "élémentarisons" les cycles ayant ce sommet redondant. □

La complexité de **directed EVEN-TRAIL** et **directed EVEN-CYCLE** est donc la même.

Remarque : *La réduction de* ***directed EVEN-TRAIL*** *à* ***directed EVEN-CYCLE*** *était à l'origine une réduction de Turing, retransformée en réduction de Karp en rassemblant tous graphes testés en un graphe non connexe. Une logspace-réduction de Karp est donnée dans [10].*

6.2.3 Partition en cycles de longueur $\frac{m}{2}$ pour des graphes non-orientés

Problème (**undirected EULER-HALF-TRAIL-PART**)
Étant donné un graphe eulérien $G = (V, E)$, existe-t-il une partition des arêtes en 2 cycles de longueur $\frac{\#E}{2}$?

Le problème est plus contraint que celui consistant à trouver un cycle de longueur $\frac{\#E}{2}$, car il faut s'assurer que le graphe obtenu en retirant les arêtes du premier cycle est encore connexe. Si c'est le cas, sachant qu'il est eulérien et vérifie $\#E' = \frac{\#E}{2}$, nous voyons que la condition est suffisante.

Théorème 6.6. ***undirected EULER-HALF-TRAIL-PART*** *est NP-complet.*

Démonstration. Ce problème est dans NP car la donnée des cycles permet de vérifier la condition.

La réduction va se faire à partir du problème **3-HAMILT**. Nous allons travailler sur des graphes valués dans des entiers naturels majorés par un polynôme. Une arête de valeur i peut être remplacée par un chemin de longueur i, et le problème de savoir s'il existe un cycle de valuation $\frac{m}{2}$ dans un graphe eulérien de valuation totale m est donc équivalent à savoir s'il existe un cycle de longueur $\frac{\#E}{2}$ dans un graphe eulérien non valué. Il est important de noter que le passage d'un graphe valué à un graphe non valué ne change pas le fait d'être eulérien ou non.

Soit un graphe non orienté connexe sans boucle 3-régulier $G = (V, E)$, possédant $2m$ sommets et $3m$ arêtes. Ce graphe est hamiltonien si et seulement s'il existe un cycle de longueur $2m$.

Ajoutons un sommet v_d. Nous allons compléter ce graphe pour le rendre eulérien, en ajoutant reliant chaque sommet à v_d par une arête d'étiquette $2m$. Nous obtenons donc un graphe G' de valeur totale $4m^2 + 3m$. Le graphe G' admet un cycle de valeur $2m$ si et seulement si G est hamiltonien.

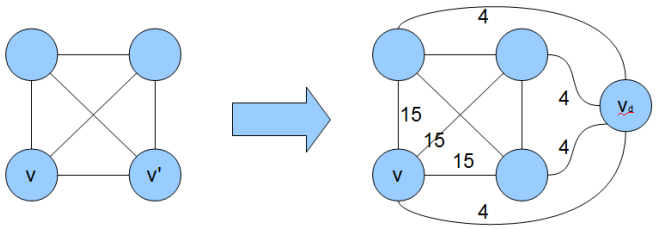
Soit un sommet v de G, et e_1, e_2 et e_3 ses 3 arêtes, remplaçons dans G' la valeur de e_1, e_2 et e_3 par $4m^2 - m + 1$. Nous obtenons donc un graphe eulérien G'' possédant une valeur totale de $16m^2$. (Voir figure 6.5)

Si G est hamiltonien, il existe dans G'' un cycle valeur $8m^2$, ne passant que par des arêtes de G. Sachant que tous les sommets sont reliés à v_d, le graphe obtenu en retirant les arêtes du premier cycle est encore eulérien, et il existe donc un second cycle de valeur $8m^2$.

Réciproquement, si G'' peut être partitionné en deux cycles de valeur $8m^2$, alors l'un des deux cycles passera par 2 arêtes de valeur $4m^2 - m + 1$, or si un cycle passe par deux des arêtes de valeur $4m^2 - m + 1$, il faut une valeur de $2m - 2$ pour le compléter, or les sommets sont séparés par des arêtes de valeur 1 ou $2m$, il y a $2m + 1$ sommets, et si l'on passe 2 fois par le même sommet, on est obligé de passer par une arête de valeur $2m$. Le cycle ne passe donc qu'une fois au plus par chaque sommet, et ne passe pas par v_d. Le cycle est bien hamiltonien dans G.

□

FIGURE 6.5 – Exemple : transformation du graphe K_4 en G''



6.2.4 Partition en cycles de longueur $\frac{m}{2}$ pour des graphes orientés

Nous pouvons définir le dual du problème **undirected EULER-HALF-TRAIL-PART** pour les graphes orientés.

Problème (**directed EULER-HALF-TRAIL-PART**)
Étant donné un graphe orienté eulérien $G = (V, E)$, existe-t-il une partition des arcs en 2 circuits de longueur $\frac{\#E}{2}$?

En outre, nous devons définir deux problèmes intermédiaires :

Problème (**directed HALF-TRAIL**)
Étant donné un graphe orienté $G = (V, E)$, existe-t-il un circuit de longueur $\frac{\#E}{2}$?

Problème (**directed LOCAL HALF-TRAIL**)
Étant donné un graphe orienté $G = (V, E)$ et un arc, existe-t-il un circuit de longueur $\frac{\#E}{2}$ passant par cet arc ?

Théorème 6.7. ***directed HALF-TRAIL*** *est NP-complet.*

Démonstration. Ce problème est dans NP, car la donnée d'un tel cycle suffit. Nous allons réduire **directed EVEN-TRAIL** à **directed HALF-TRAIL**. Nous utiliserons toujours le même principe d'équivalence entre graphes et graphes étiquetés par des entiers.

Soit un graphe $G = (V, E)$. Nous allons nous arranger pour que tout cycle aie une longueur $< \frac{m}{2}$ en ajoutant un puits, et faisant pointer $m+2$ arcs vers ce puits. S'il le faut, nous ajoutons 2 arcs vers ce puit pour que le graphe $G' = (V', E')$ obtenu aie $4m'$ arcs.

Soit e, une arête distinguée. Nous allons augmenter son étiquette de 4 itérativement, jusqu'à atteindre m itérations, ie. scinder l'arête itérativement en un chemin de 5 arêtes.

S'il existe un cycle de longueur $2p$ passant par e, alors il existe un nombre k d'itérations tel que $2p+4k = \frac{4m'+4k}{2}$ ($k = m' - p$), et donc au cours d'une des itérations, le graphe possédera un cycle de valeur $\frac{\#E'}{2}$. Réciproquement, si au cours d'une des itérations, le graphe possède un cycle de valeur $\frac{\#E'}{2}$, sachant qu'initialement, le graphe n'en possède pas et seuls les cycles passant par e on augmenté de valeur, le cycle de valeur $\frac{\#E'}{2}$ passe par e, et comme on n'a ajouté des valeures paires, on n'a pas changé la parité des cycles passant par e, et il existe donc bien un cycle de longueur paire. □

Théorème 6.8. ***directed LOCAL HALF-TRAIL*** *est NP-complet*

Démonstration. La réduction est celle canonique d'un problème **P** vers **LOCAL P** : on applique le problème **directed LOCAL HALF-TRAIL** sur chaque arête. □

Les deux réductions précédentes étant des réductions de Turing, nous allons maintenant proposer des réductions de Karp pour prouver les NP-complétudes de **directed HALF-TRAIL** et **directed LOCAL HALF-TRAIL**.

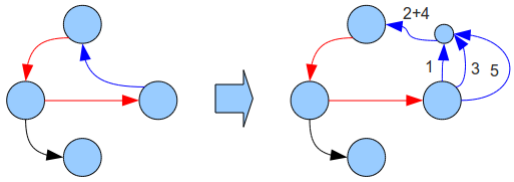
Théorème 6.9. ***directed LOCAL HALF-TRAIL*** *est NP-complet*

Démonstration. Nous allons réduire **directed LOCAL ODD-CYCLE** à **directed LOCAL HALF-TRAIL**. Pour simplifier la réduction, nous allons considérer des graphes étiquetés par des entiers naturels non nuls, où l'étiquette est à remplacer par un chemin de la même longueur.

Soit $G = (V, E)$ un graphe orienté, $\#E = 8m+4$ (on peut se ramener à ce cas canonique sans perte de généralité en ajoutant une source, un puits et autant d'arcs nécessaire pour satisfaire la congruence) et a un arc distingué. Nous allons définir $G' = (V', E')$ à partir de G en scindant a en a_1, a_2, avec a_1 partant du sommet existant v_1 et rejoignant le sommet créé v_2. Nous allons ajouter à a_2 une étiquette de valeur $\sum_{i=1}^{4m+2} 2i = (4m+2)(4m+3)$.(Voir figure 6.6) Nous allons enfin ajouter des arcs $b_1, \cdots, b_{4m+2}$ d'étiquette $2i+1$ pour l'arc i, entre v_1 et v_2. Le graphe G' possède donc $8m + 4 + (4m + 2)(4m + 3) + (4m + 2)(4m + 4) = 32m^2 + 52m + 18$ arcs. (Voir figure 6.6)

Le graphe G possède un circuit de longueur impaire passant par a si et seulement si le graphe G' possède un circuit de longueur $16m^2 + 26m + 9$ passant par a_2.

FIGURE 6.6 – Exemple : transformation du graphe G en G'



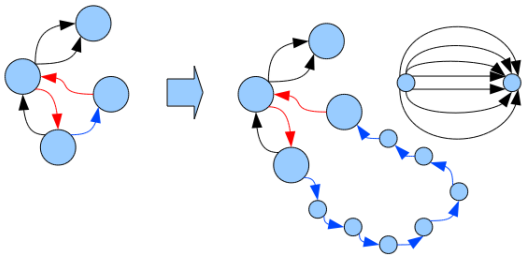
□

Théorème 6.10. ***directed HALF-TRAIL*** *est NP-complet*

Démonstration. Nous allons réduire **directed LOCAL HALF-TRAIL** à **directed HALF-TRAIL**.

Soit $G = (V, E)$ un graphe orienté comportant $2m$ arcs et e un arc distingué. Nous allons construire le graphe $G' = (V', E')$ à partir de G en ajoutant un sommet source, un sommet puits, $2m+1$ arcs entre la source et le puits, et scindant l'arc distingué e en un chemin de longueur $2m + 2$. (Voir figure 6.7)

FIGURE 6.7 – Exemple : transformation du graphe G en G'



S'il existe dans G un chemin cycle de longueur $m = \frac{\#E}{2}$ passant par e, alors il existe un dans G' un cycle de longueur $3m + 1 = \frac{\#E'}{2}$.

Réciproquement, s'il existe dans G' un cycle de longueur $3m + 1$, il passera nécessairement par le chemin de longueur $2m+2$ car les arcs ajoutés entre la source et le puits ne peuvent appartenir à un cycle sachant que le puits a un degré sortant nul, et il est impossible que le chemin ne passe que par des arcs de G car il n'y en n'a pas assez. En fusionnant le chemin de longueur $2m + 2$ en l'arc distingué e, on obtient dans G un cycle de longueur $2m$ □

Nous allons maintenant démontrer la NP-complétude de **directed EULER-HALF-TRAIL-PART**.

Théorème 6.11. ***directed EULER-HALF-TRAIL-PART*** *est NP-complet*

Démonstration. Le problème est dans NP, car la donnée des circuits en question suffit. Nous allons réduire **directed LOCAL HALF-TRAIL** à **directed EULER-HALF-TRAIL-PART**.

Soit $G = (V, E)$ un graphe orienté possédant $2m$ arcs. On va lui ajouter un sommet v_d, et choisir un arc $e \in E$. On relie chaque sommet à v_d dans les deux sens avec des arcs d'étiquette $2m$. On complète ensuite chaque sommet en le reliant par des arcs à v_d afin d'obtenir une égalité entre les degrés entrants et les degrés sortants. A chaque fois que l'on ajoute un arc, aussi bien à la première étape qu'à la seconde, on ajoute $2m$ à l'étiquette de e. Le graphe obtenu $G' = (V', E')$ a pour valeur totale $2m + 4km$, avec $k \geq 2m$. e a pour étiquette $2mk + 1$.

S'il existe un circuit de longueur m passant par e dans G, il existe un circuit de longueur $2km + m$ passant par e dans G'. De plus, sachant que tous les sommets sont doublement reliés à v_d, le graphe obtenu en retirant les arcs du cycle est encore fortement connexe. Il est donc eulérien, et admet ainsi un second circuit de longueur $2km + m$.

Réciproquement, si l'ensemble des arêtes de G' peut être partitionné en 2 circuits de longueur $2km + m$, alors un des deux circuits passe par e. Ce circuit doit alors être complété avec une valeur de $m - 1$, et ne peut donc passer que par des arcs de G. Il existe alors un circuit de longueur m dans G passant par e. □

7 Le problème ETER2

7.1 Présentation

Le problème **ETER2** est la généralisation direct du jeu Eternity II. L'étude de ses variantes présentent donc un intérêt majeur pour la compréhension de sa complexité globale.

7.2 Études de complexités

Nous allons prouver la NP-complétude du problème **ETER2**. Cette preuve de [5]

Théorème 7.1. ***ETER2*** *est NP-complet*

Démonstration. Il est facile de voir qu'**ETER2** est dans NP sachant qu'étant donné une solution, il suffit de tester chaque contrainte, ce qui se fait en $\mathcal{O}\left(n^2\right)$

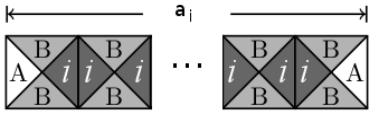
Nous allons réduire le problème de **3-PARTITION** à **ETER2**

Nous allons maintenant encoder ce problème sous forme d'un jeu de pièces pour **ETER2**.

Le principe de la réduction consiste à créer un plateau de $m \times S$, et encoder chaque entier naturel i avec i pièces de manière à ce qu'on ne puisse les positionner que les uns à côté des autres horizontalement. D'après la contrainte supplémentaire, une ligne du plateau ne peut être remplie qu'avec 3 blocs de pièces. Si le plateau est complet, cela signifie donc que l'on aura réussi à remplir toutes les lignes, donc regrouper tous les nombres en triplets de même somme.

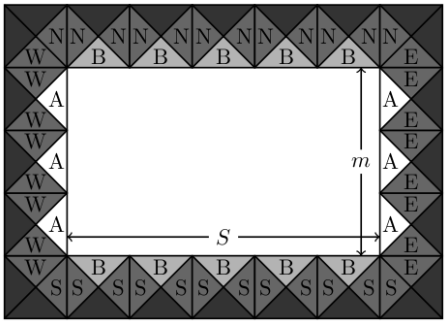
Soit E le multi-ensemble. Pour tout élément $a_i \in E$, nous allons l'encoder en a_i pièces de la forme suivante (figure 7.1) :

FIGURE 7.1 – Encodage du nombre a_i



Et afin de respecter les contraintes de bordure, nous allons entourer le jeu d'un cadre comme suit :

FIGURE 7.2 – Encodage du nombre a_i



Les motifs A et B forcent l'orientation des blocs de pièce. L'unicité de la couleur i oblige les pièces représentant le nombre a_i à rester sous forme de bloc. Toute solution de 3-partition est également solution de ce jeu, et inversement. □

Variante 1 (ETER2 carré).
Le problème reste NP-complet si l'on impose que le plateau soit carré : il suffit de combler les lignes vides par des carrés en s'arrangeant pour utiliser des motifs forçant l'existence d'une solution unique parmi ceux-ci. Voir [5] pour plus de détails.

Variante 2 (ETER2 pièces uniques).
On peut s'arranger pour que chaque pièce soit différente en n'utilisant pas le même motif i pour toutes les pièces encodant a_i. Le problème reste donc NP-complet si l'on impose que les pièces soient toutes différentes.

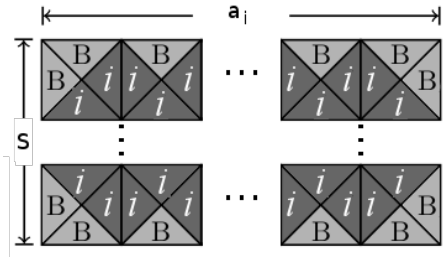
Variante 3 (ETER2 sans bordures).
On peut retirer les contraintes de bordure tout en conservant la NP-complétude du problème.

Variante 4 (ETER2 3 motifs).
On peut imposer qu'il n'y aie que des pièces ayant exactement 3 motifs sans perdre en NP-complétude en remarquant que la démonstration n'en n'utilise pas qui en ont 4, et que l'on peut en ajouter un 3e pour ceux qui n'en n'ont que 2.

Variante 5 (ETER2 4 motifs).
On peut également imposer qu'il n'y aie que des pièces ayant exactement 4 motifs (à part les coin) sans perdre en NP-complétude. Il suffit de remplacer les B par des I (impair) en haut et des P (pair) en bas, et suivant la ligne où l'on place la barre, la faire tourner de 180°, de remplacer les i par des motifs uniques en fixant ainsi l'ordre des pièces dans la barre.

Variante 6 (ETER2 2 motifs).
On peut également imposer qu'il n'y aie que des pièces ayant exactement 2 motifs tout en restant NP-complet. Il faut prendre $A = B = N = E = S = W$ et pour empêcher les blocs d'effectuer des rotations, il ne faut plus les encoder sur en une ligne, mais en $S + 1$ lignes collées entre elles toujours avec le motif i. Voir figure 7.3.

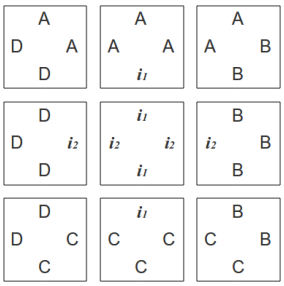
FIGURE 7.3 – Encodage du nombre a_i



Bien entendu, si l'on impose que les pièces aient exactement 1 motif, on peut décider en un temps linéaire s'il existe une solution : il existe une solution si et seulement si toutes les pièces sont identiques.

Une autre démonstration de la variante 6 consiste à réduire le problème **ETER2** à sa variante 5 en effectuant l'encodage suivant (voir figure 7.4).

FIGURE 7.4 – Encodage de la pièce $p_i = (A, B, C, D)$



Variante 7 (ETER2 (A,A,B,B) et (A,A,A,B)).
On peut enfin imposer qu'il n'y aie que des pièces étant de la forme (A, A, B, B) et (A, A, A, B) tout en restant NP-complet, car la réduction de la variante 6 n'utilise que des pièces de ces deux formes.

Variante 8 (ETER2 (A,B,A,C)).
*La démonstration de la NP-complétude d'**ETER2** n'ayant fait intervenir que des pièces de la forme (A, B, A, C) où A, B et C ne sont pas nécessairement distincts, nous pouvons nous restreindre à ces pièces tout en gardant la NP-complétude.*

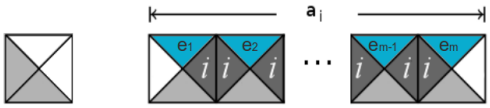
Variante 9 (ETER2 (A,B,A,C) sans bordures).
*Nous pouvons appliquer en même temps les deux remarques faites pour **ETER2 (A,B,A,C)** et **ETER2 sans bordures**. Ce problème est donc encore NP-complet.*

Variante 10 (ETER2 $2 \times n$).
Si l'on restreint les dimensions de la grille à $2 \times n$, le problème reste NP-complet.

Nous allons réduire **binary-digit PARTITION** à cette variante.

Étant donné un multi-ensemble d'entiers naturels de taille n, de somme totale $2s$ et tels qu'encodés en base n, les digits soient binaires. Nous pouvons associer à chaque élément a_i l'ensemble $S_{a_i} = \{e_1, \cdots, e_m\}$ de ses puissances de n (nous avons bien une bijection entre S_{a_i} et a_i car par hypothèse, chaque digit vaut soit 0, soit 1. Nous allons coder chaque a_i avec $\#S_{a_i}$ pièces comme suit :

FIGURE 7.5 – Coin et encodage de a_i



Considérons que nous obtenons s pièces. Alors, il existe un contour de taille $s \times 2$ si et seulement s'il existe une partition du multi-ensemble en 2 ensembles de même taille.

Concernant le problème **ETER2-FROM-ETER2SET**, il suffit de remarquer que toutes les pièces partagent le motif B, donc n'importe quelle répartition en grille sera une configuration valide pour **ETER2SET**.

Le problème **ETER2-FROM-2-ROT** est équivalent à trouver une configuration acceptante sans autoriser la rotation des pièces. La démonstration de NP-complétude s'obtient à partir de la précédente, en conservant le même encodage et en remarquant que l'on connait déjà la rotation des pièces.

7.3 Construction vs décision

Soit P un prédicat à valeurs dans la classe des jeux de pièces, qui indique si un jeu de pièces est compatible avec **ETER2**. Nous allons construire un algorithme permettant de construire une solution à partir d'un jeu de pièces compatibles.

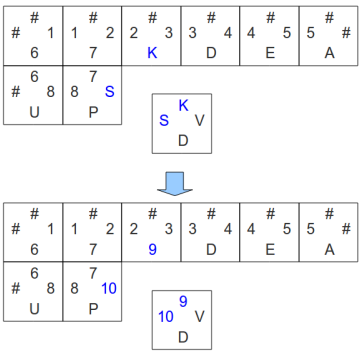
Il est important de noter que dans l'algorithme, lorsque l'on place une pièce, on ne restaure pas l'ancienne couleur. Ainsi, toutes les pièces déjà placées sont liées entre elles par des motifs uniques qui forcent leur position et leur rotation.

On peut placer n'importe quel coin en position (1,1) sans perte de généralité sachant que les pièces peuvent être tournées et donc le plateau aussi. Sachant que l'on effectue un chemin de remplissage sous forme de lignes, il y aura toujours au moins 2 contraintes de bordure à respecter, et parfois 3 ou même 4 pour la dernière ligne. Si le prédicat vaut vrai pour le jeu obtenu en remplaçant les bordures contraintes d'une pièce par des motifs uniques ainsi que celles de ses pièces environnantes responsables de la contrainte, alors en raison de l'unicité des motifs, il existe une solution avec les pièces déjà placées comme telles. A chaque étape, on se trouve avec un jeu compatible, car on ne place jamais une pièce sans s'assurer qu'il existe une solution avec ce pièce posée, et que l'on part d'un

```
Positions[1,1] ← any corner
while has-free-positions() do
  t ← next-free-position()
  for all free piece as p do
    for all compatible rotation of p do
      replace filled colors by unique color
      if P( set with new colors ) then
        t ← p
        Go to while
      else
        restore previous colors
      end if
    end for
  end for
end while
```

jeu qui a nécessairement une solution. Ainsi, il existera nécessairement une pièce pour laquelle le prédicat sera vrai, et donc le nombre de pièces libres est strictement décroissant, ce qui prouve que le programme va s'arrêter.

FIGURE 7.6 – Étape de l'algorithme de construction d'une solution pour **ETER2**



Le nombre d'appels au prédicat P est en $\mathcal{O}\left((mn)^2\right)$. (On peut majorer le nombre d'appels par $mn*(mn+1)$).

8 Le problème ETER2SET

8.1 Présentation

Le problème **ETER2SET** est la relaxation de contraintes proposée dans le sujet de stage. Nous verrons dans cette étude que de manière expérimentale, le nombre de solutions augmente de manière considérable entre **ETER2** et **ETER2SET**. En outre, n'ayant pas trouvé de propriété facilitant la recherche de solution, cette piste ne sera pas retenue.

8.2 Études de complexités

Théorème 8.1. ***ETER2SET*** *est NP-complet*

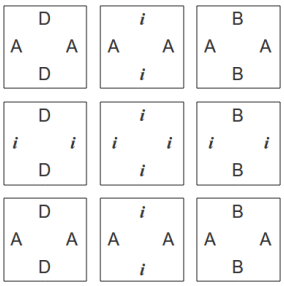
Démonstration. Sachant qu'il existe un algorithme polynomial qui, étant donnée une solution, va vérifier sa validité, le problème est dans NP. Pour ce problème, sachant que les rotations n'importe pas, chaque pièce peut être représentée par l'ensemble de ses motifs, de cadinalité au plus 4.

Nous allons réduire **ETER2 (A,B,A,C)** à **ETER2SET** pour prouver sa NP-complétude. L'intérêt de cette variante est que l'on peut échanger deux motifs opposés sans changer la pièce, car cela correspondra à une rotation. ie $(A, B, A, C) = (A, C, A, B)$ alors que $(A, B, C, D) \neq (A, D, C, B)$.

Pour cela, nous allons encoder chaque pièce p_i (A, B, C, D) d'**ETER2** par un groupe de 9 pièces :

$$\{\{i\}, \{i, A\}, \{i, B\}, \{i, C\}, \{i, D\}, \{A, B\}, \{B, C\}, \{C, D\}, \{D, A\}\}$$

FIGURE 8.1 – Encodage de la pièce $p_i = (A, B, C, D)$



Nous devons également ajouter un contour pour forcer le motif de bordure à se mettre autour, et éviter les déformations des pièces précédentes car il n'y a pas de contrainte particulière au niveau des extrémités dans **ETER2SET**.

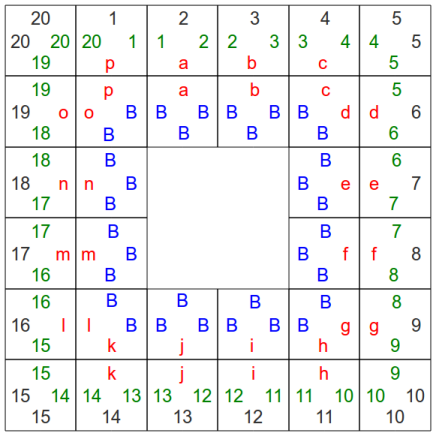
Soit B le motif de bordure. Nous allons encoder la contrainte de bordure comme sur la figure. Les motifs sont uniques à part B qui est partagée avec certaines autres pièces.

Si l'on a une solution au problème **ETER2**, nous aurons une solution au problème **ETER2SET** en positionnant les pièces comme sur les figures.

Maintenant, soit une configuration acceptante pour **ETER2SET**. Prouvons tout d'abord que la bordure est nécessairement comme indiquée dans la figure 8.2.

Considérons tout d'abord les pièces positionnées sur les extrémités coins. Les pièces partageant le même motif sont au nombre de 2, ce qui obligent ces pièces à être dans des coins. Par le même raisonnement, les pièces de bordures partageant les motifs avec 3 autres pièces, elle sont nécessairement en bordure. La rangée extérieure est donc conservée. La seconde rangée est déterminée par la première rangée à cause de l'unicité des lettres. Il s'ensuit

FIGURE 8.2 – Encodage de la contrainte de bordure



que la bordure est nécessairement comme la représentation graphique, moyennant des rotations et sans tenir compte du sens horaire ou antihoraire.

Sachant que seuls les motifs B sont partagés avec le reste des pièces, ce sont nécessairement eux qui vont entrer en compte dans la correspondance avec les pièces intérieures. Par des raisonnements similaires, on montre que les blocs encodant les pièces sont conservés, et que ce sont les motifs de la pièce encodée qui vont entrer en compte dans la correspondance avec les autres blocs de pièce. On aura donc bien une solution à **ETER2**. □

Variante 1 (**ETER2SET pièces uniques**).
On peut ajouter la contrainte d'unicité des pièces sans perdre la NP-complétude. En effet, les pièces de bordure sont toutes uniques, et chaque pièce du centre n'ayant au plus que 2 motifs différents, on peut ajouter à chacune un motif unique, rendant ainsi le jeu sans doublon.

Variante 2 (**ETER2SET-FROM-2-ROT**).
*Sachant que **ETER2** sans les contraintes de bordure est encore NP-complet, et que chaque pièce est encodée en un ensemble clos par **2-ROT**, alors **ETER2SET-FROM-2-ROT** est NP-complet.*

Variante 3 (**ETER2SET 4 motifs**).
On peut imposer que chaque pièce aie exactement 4 motifs distincts, car il suffit de compléter les pièces avec des motifs uniques et le résultat ne sera pas changé.

Variante 4 (**ETER2SET 3 motifs**).
On peut également imposer que chaque pièce aie exactement 3 motifs distincts, car la réduction n'a utilisé que des pièces ayant au plus 3 motifs et il suffit de rajouter des motifs uniques pour celles qui en ont moins. Le problème reste donc NP-complet.

Variante 5 (**ETER2SET 2 motifs**).
*On peut enfin également imposer que chaque pièce aie exactement 2 motifs distincts, car le problème **ETER2** sans les contraintes de bordure et restreint aux pièces (A, B, A, C) est encore NP-complet, or chaque pièce a été encodée avec des pièces ayant au plus 2 motifs distincts. Le problème reste donc NP-complet.*

8.3 Construction vs réduction

L'algorithme de réduction pour **ETER2SET** va être un peu plus compliqué que celui de **ETER2**, même s'il se base toujours sur un remplissage ligne par ligne en se basant sur un remplacement de motifs.

Nous allons tout d'abord essayer d'identifier les bords et les coins. Pour cela, il faut remarquer qu'un coin n'est contigü qu'à 2 pièces, ce qui fait que parmi ses 4 motifs, seuls 2 sont utilisés. De même, pour les bordures, un

motif n'est pas utilisé et peut donc être remplacé tout en conservant au moins une solution et sans en ajouter de nouvelles.

La première étape consiste à trouver un coin. Pour cela, nous allons prendre tous les triplets (p_1, p_2, p_3) de pièces de telle sorte que la pièce p_1 partage au moins 1 motif avec p_2 et p_3 (pas nécessairement le même), et remplacer tous les motifs de p_1 par un motif unique 1, puis remplacer un motif de p_2 et de p_3 par le même motif 1. D'après les considérations précédentes, il existera un triplet pour lequel le remplacement de motif gardera l'existence d'une solution. De plus, comme la pièce p_1 ne contient qu'un seul motif, et qui n'est partagé qu'avec 2 motifs, alors nécessairement, c'est un coin.

Accessoirement, on peut placer la pièce p_2 à droite de la pièce p_1 sans perte de généralité, mais elle pourra également être retrouvée par la seconde étape.

Pour chaque nouvel emplacement libre, pour chaque pièce partageant un motif avec chacune des pièces l'environnant (nous savons que cet ensemble est non vide, car à chaque étape, on s'assure qu'il existe une solution), nous allons créer un nouveau motif i, remplacer un des motifs de la pièce et de celles avoisinantes par i. L'existence d'une solution implique l'existence d'une telle pièce et d'un tel remplacement de motif permettant de garder une solution.

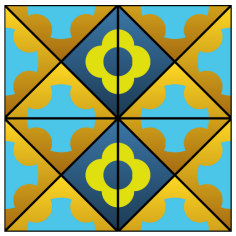
Le nombre de pièces étant strictement décroissant, le programme va terminer. La conservation de l'existence d'une solution et l'unicité des motifs nous guarantit que l'on a une solution.

8.4 Questions annexes

Question 8.1. *Etant donnée une configuration acceptée par* ***ETER2SET****, existe-t-il des solutions d'****ETER2*** *de signature différente ?*

Nous pouvons déjà facilement construire un exemple avec plusieurs pièces identiques :

FIGURE 8.3 – Exemple de configuration **ETER2SET** donnant plusieurs signatures compatibles avec **ETER2**

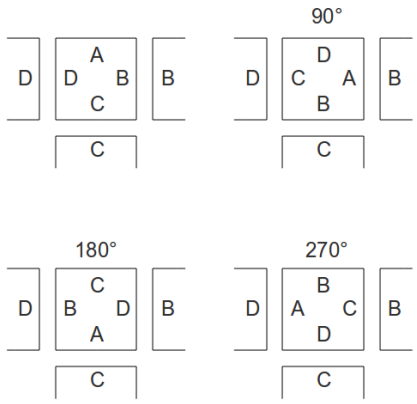


En revanche, il n'est pas facile de construire de solution avec des pièces uniques. Essayons donc de mettre en évidence des conditions nécessaires à l'obtention d'un tel résultat. S'il existe 2 signatures acceptées par **ETER2**, alors soient C_1 et C_2 deux configurations acceptées, de signature différente. Nous pouvons passer de C_1 à C_2 en effectuant une rotation d'au plus toutes les pièces et laissant invariant les motifs extérieurs. Le nombre de pièces tournées laissant invariant les motifs extérieurs est donc une partie non vide de $\mathbb{N}$ et admet un plus petit élément. Soit n ce plus petit élément, et soit E un ensemble de n pièces tournéees laissant invariant les motifs extérieurs et permettant de passer de C_1 à C_2.

Chaque pièce de E a au moins deux pièces adjacentes qui sont dans E. En effet, dans le cas contraire, nous avons une pièce qui a 3 côtés invariants pour sa rotation et alors elle est nécessairement de la forme (A, B, A, B) car si elle est tournée de 90 ou de 270°, elle est de la forme (A, A, A, A) et si elle est tournée de 180°, elle est de la forme (A, B, A, B), avec A et B non nécessairement distincts. Mais dans ce cas, la signature ne change pas, ce qui est en contradiction avec l'hypothèse.

Il existe donc nécessairement 4 pièces ayant exactement 2 côtés adjacents à des pièces dans E. De plus, nous pouvons imposer que ces côtés ne soient pas en face, car sinon, si toutes les pièces étaient ainsi, on aurait un cycle

FIGURE 8.4 – Les rotations imposent $A = B = C = D$ ou $A = C$ et $B = D$



infini, ce qui est impossible. Pour chaque pièce (A, B, C, D) vérifiant ces propriétés, si celle-ci est tournée de 180°, alors $A = C$ et $B = D$ et la signature est conservée. Nécessairement, il faut qu'elle soit tournée de 90° ou 270°. Mais alors $B = C = D$ ou $A = D = C$ et donc la pièce est de la forme (A, A, A, B).

Faute de temps, la question n'a pas été étudiée en profondeur. Il est probable que la création d'algorithme de résolution de contraintes permettrait par une étude systématique de mettre en relief un exemple de solution ayant plusieurs signatures, mais une telle solution ne serait pas de petite taille.

9 Le problème 2-ROT

9.1 Présentation

Dans cette partie, nous allons nous pencher plus en profondeur sur le problème **2-ROT**. Il s'agit d'un problème obtenu à partir d'une relaxation des contraintes d'**ETER2**. L'étude de ce problème en particulier sera justifié par les propriétés intéressantes ce cette relaxation de contraintes, notamment par la possibilité d'obtenir un grand nombre de solutions à partir de partitions en formes closes, et son nombre de solution relativement proche de ceux du problème **ETER2**.

9.2 Études de complexités

Théorème 9.1. *2-ROT est NP-complet*

2-ROT $\in$ NP car étant donné une solution, il suffit d'effectuer la somme de chaque côté pour vérifier sa validité, et cela se fait en $4m$ opérations pour un ensemble de m motifs.

Nous allons maintenant réduire **binary-digit PARTITION** à **2-ROT**. Soit $S = \{a_1, \cdots, a_n\}$ un multi-ensemble d'entiers naturels.

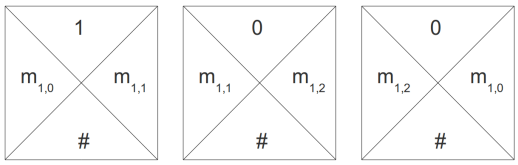
A chaque nombre a_i, nous allons associer le multi-ensemble S_{a_i} de ses puissances de n tel que la somme de ses éléments soit égal à a_i. Par exemple, pour un ensemble de 3 élements contenant le nombre 5, $S_5 = \{1, 0, 0\}$. Il existe bien une bijection entre a_i et S_{a_i} pour un n donné.

Le principe de la réduction consiste à encoder chaque nombre a_i en $\#S_{a_i}$ pièces dont les motifs définissent une orientation, et forcer ces pièces à être nécessairement dans la même direction. Ensuite, associer la gauche et le haut à la première partition, la droite et le bas à la seconde. S'il existe une solution à **2-ROT**, on obtiendra une orientation du même nombre de pièce vers chaque partition, et aura ainsi une solution à **binary-digit PARTITION**.

Nous allons donc encoder $S_{a_i} = \{v_{i,0}, \cdots, v_{i,j-1}\}$ en j pièces $\{p_0, \cdots p_j\}$ où $p_k = (m_{i,k}, \#, v_{i,k}, m_{i,k+1 \mod j})$.

Voir l'exemple de l'encodage de 5 figure 9.1

FIGURE 9.1 – Encodage de 5 pour $c = 3$.



Du fait de l'unicité des motifs, la contrainte d'égalité du nombre d'occurrences du même motif d'un côté et de l'autre permet de forcer tous $p_{a,b}$ à avoir le motif 1 dans la même direction. Le fait de coder les nombres en base n permet de s'assurer de ne pas avoir de retenue, car il n'y aura jamais n

Si l'on a une solution au problème des partitions, on peut facilement construire une solution pour son équivalent encodé dans **2-ROT** en faisant faisant pointer le motif 1 vers la gauche pour les pièces qui représentent un nombre appartenant au premier sous-ensemble, et vers la droite pour les pièces qui représentent un nombre appartenant au second sous-ensemble de la partition.

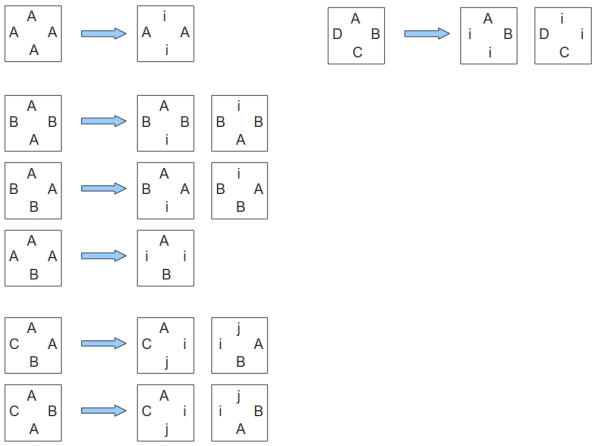
Maintenant, étant donné une solution de l'instance encodée dans **2-ROT**, il suffit d'assigner les éléments de S dont les pièces ont le motif 1 qui pointe vers le haut et la gauche au premier sous-ensemble, et ceux dont les pièces pointent vers le bas et la droite au second sous-ensemble. Du fait des contraintes d'égalité entre parties opposées, nous savons que c'est une solution à **binary-digit PARTITION**. **2-ROT** est donc NP-complet.

Nous pouvons faire la même remarque sur le problème **2-ROT-FROM-ETER2SET** que sur **ETER2-FROM-ETER2SET**. En effet, toutes les pièces partagent le motif #, ce qui fait que l'instance est compatible **ETER2SET**.

Variante 1 (2-ROT 4 motifs).
Nous pouvons imposer que chaque pièce aie exactement 4 motifs distincts en augmentant le nombre de motifs suivant la table 9.2.

Variante 2 (2-ROT 3 motifs).
Nous pouvons imposer que chaque pièce aie exactement 3 motifs distincts en augmentant le nombre de motifs pour ceux en ayant moins, et réduisant les nombre de motifs pour ceux en ayant 4, toujours suivant la table 9.2.

FIGURE 9.2 – Table des encodages de pièces pour augmenter/réduire le nombre de motifs



Variante 3 (2-ROT 2 motifs).
Si l'on impose que chaque pièce aie exactement 2 motifs distincts, nous ne savons pas si le problème reste NP-complet.

Ce problème est cependant équivalent au problème suivant :

Problème (BIPARTITE-CYCLE-PART)
Étant donné un graphe eulérien non orienté, où certains sommets sont reliés par des couples d'arêtes d'une même couleur, existe-t-il une partition des arêtes en cycles élémentaires n'ayant au plus qu'une occurrence de chaque couleur par cycle et telle que le graphe obtenu en créant un sommet par cycle et une arête entre chaque couple de cycle partageant une couleur soit biparti ?

Variante 4 (2-ROT (A,A,A,B)).
*Si l'on impose que chaque pièce soit de la forme (A, A, A, B), le problème est dans P. En effet, le problème est équivalent à **1-ROT**.*

Variante 5 (2-ROT (A,A,B,B)).
*Si l'on impose que chaque pièce soit de la forme (A, A, B, B), le problème est dans P. Le problème est également équivalent à **1-ROT**.*

En effet, soit le graphe composé des couleurs commet sommets, et des arêtes (A, B) pour chaque pièce (A, A, B, B). Alors si chaque composante connexe du graphe est eulérienne, nous obtenons une solution en prenant chaque arête (A, B) d'un cycle eulérien de chaque composante connexe, et en tournant la pièce de façon à obtenir (A, A, B, B).

S'il existe une composante connexe du graphe qui n'est pas eulérien, cela signifie qu'il existe un motif pour lequel le nombre de pièces le comprenant est impair, et il n'existe clairement pas de solution au problème **2-ROT**.

2-n-ROT
Le problème **2-n-ROT** est un cas particulier du problème **2-ROT** où le nombre n de motifs est fixé.

Théorème 9.2. ***2-2-ROT*** *est dans P*

Pour $n = 2$, il n'y a que 4 configurations de motifs pour former les cases : (a, a, a, a), (a, a, a, b), (a, a, b, b) et (a, b, a, b). Les cases de type (a, a, a, a) et (a, b, a, b) sont déjà solutions elles-même et n'ont aucune interaction avec les autres et peuvent donc ne pas être prises en compte dans le problème de décision. De plus, si les cases de la forme (a, a, a, b) sont en nombre impair, il n'y a pas de solution car les motifs de chaque couleur sont en nombre impair et il est donc facile de conclure. Il s'agit donc de regarder la parité du nombre de cases de la forme (a, a, b, b). Si ce nombre est pair, alors l'ensemble est compatible et il est facile de fournir une configuration valide de rotations. Si ce nombre est impair, alors soit le nombre de cases de la forme (a, a, a, b) est non nul, et comme il est pair, alors on peut former un sous-ensemble compatible avec 2 cases (a, a, a, b) et une case (a, a, b, b), soit il n'y en a pas, et alors l'ensemble n'est pas compatible.

Pour résumer, voici un algorithme testant l'appartenance d'un ensemble à **2-2-ROT**.

```
1: aaabEven ← true
2: aabbEven ← true
3: aaabExists ← false
4: for i = 1 to size(S) do
5:   if S[i] matches (a, a, a, b) then
6:     aaabEven ←!aaabEven
7:     aaabExists ← true
8:   end if
9:   if S[i] matches (a, a, b, b) then
10:    aabbEven ←!aabbEven
11:  end if
12: end for
13: return aaabEven and (aabbEven or aaabExists)
```

Complexité de l'algorithme : $\mathcal{O}(n)$

Question 9.1. *Le problème* ***2-ROT*** *est-il FPT (fixed parameter tractable) ?*

Le problème **SUBSET-2-ROT** consiste, étant donnée une configuration valide de **2-ROT**, à trouver un sous-ensemble strict étant également une configuration valide de **2-ROT**. Autrement dit, ce problème consiste à partitionner l'ensemble en 2 sous-ensembles compatibles.

Théorème 9.3. ***SUBSET-2-ROT*** *est NP-complet.*

Démonstration. Etant donné un résultat, il est facile de vérifier en temps polynomial qu'il est compatible avec les contraintes fixées. Le problème est donc dans NP.

Nous allons réduire le problème **binary-digit SUBSET-SUM** à **SUBSET-2-ROT**. Soit $S = \{a_1, \cdots, a_n\}$ une instance de **binary-digit SUBSET-SUM** où les nombres encodés en base $n+1$ n'ont que des digits binaires.

Nous allons encoder chaque élément $a_i \in S$ de la même manière que pour le problème **2-ROT**, mais la puissance de $n+1$ sera positionné en haut si $a_i > 0$ et en bas si $a_i < 0$. (La notion de haut et de bas a un sens car les pièces sont déjà tournées et restent désormais fixes). Soit donc $S_i = \{e_1, \cdots, e_m\}$ le multi-ensemble des puissances de $n+1$ apparaissant dans a_i autant de fois que la valeur du digit. En l'occurrence, d'après l'hypothèse de **binary-digit SUBSET-SUM**, le multi-ensemble sera un ensemble pour les a_i car chaque digit vaudra 0 ou 1.

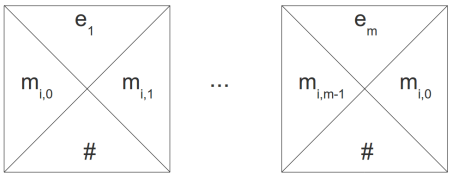
Enfin, soit s la somme des éléments de e. Nous allons encoder $-s$ de la même manière. Il est à noter que cette fois-ci, le multi-ensemble aura probablement des doublons.

Comme nous sommes en base $n+1$, même si tous les a_i sont de même signe, leur somme n'aura jamais de retenue. Nous pouvons donc baser notre égalité de sous-somme sur une comparaison digit à digit.

La configuration de pièces obtenue en conservant les rotations initiales est bien une configuration compatible.

De plus, d'après l'encodage des nombres, si l'on retire une pièce, on est obligé de retirer toutes les pièces liées au même nombre pour conserver la compatibilité.

FIGURE 9.3 – Encodage de a_i



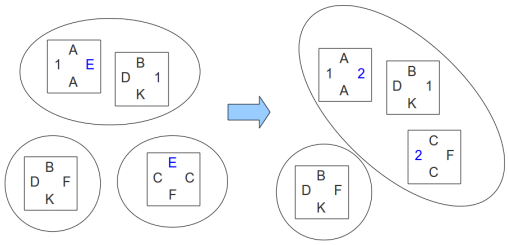
S'il existe une partition de cette configuration en 2 sous-ensembles, un des deux sous-ensembles ne contiendra aucune pièce de $-s$, et les nombres associés à ses pièces seront donc bien un sous-ensemble de S de somme nulle. Réciproquement, s'il existe un sous-ensemble de somme nulle, alors on pourra partitionner la configuration en mettant les pièces correspondantes mettant dans un ensemble, et en mettant les autres pièces dans l'autre ensemble. □

9.3 Construction vs décision

Le principe de réduction pour le problème **2-ROT** reste le même. Nous partons de l'état initial qui est l'ensemble des singletons du jeu de pièces, et nous allons itérativement regrouper des ensembles de pièces qui ont une rotation unique les unes par rapport aux autres, jusqu'à n'obtenir qu'un seul ensemble.

Etant donnée une étape de l'algorithme, soit tous les ensembles sont déjà clos par la propriété, et nous avons une solution par n'importe quelle combinaison de rotations entre les ensembles, soit il existe au moins deux ensembles ayant chacun une pièce partageant la même couleur et tel que si on oriente les pièces pour qu'elles soient dans des direction opposées, alors le jeu admette une solution. Pour chaque couple bordures de même couleur dans dans 2 groupes différents, remplacer ces deux bordures par un nouveau motif unique, et tester l'existence d'une solution. Si c'est le cas, unir les deux groupes, sinon, restaurer l'ancienne couleur.

FIGURE 9.4 – Étape de l'algorithme de construction d'une solution pour **2-ROT**



De même que pour **ETER2**, on voit qu'à chaque étape, l'existence d'une solution est conservée, et que le nombre d'ensembles va strictement diminuer, montrant que la programme va s'arrêter en laissant un seul ensemble possédant une solution, et dont l'unicité de chaque motif va assurer que l'on détient la solution.

9.4 Algorithme de backtracking

Un premier algorithme consisterait à effectuer un algorithme de backtracking en testant la validité pour chaque couleur l'une après l'autre, en revenant à la couleur précédente si la suivante ne peut compléter la solution partielle en une solution partielle valide avec celle-ci.

Le problème **2-ROT** semble moins bien se prêter au backtracking que **ETER2** en ce qu'il ne peut tester la validité d'une couleur qu'une fois l'intégralité des pièces comprenant cette couleur tournées. On pourrait également

effectuer le même backtracking que pour **ETER2**, mais cela signifierait prendre le même temps de calcul pour vérifier des conditions plus faibles alors que l'on pourrait vérifier les conditions fortes. De plus, on perdrait des solution de **2-ROT**.

L'espoir réside donc dans d'autres propriétés du problème **2-ROT** que nous allons essayer de mettre en lumière.

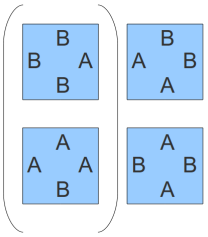
9.5 Sous-ensembles solution

L'idée générale est la suivante : si nous arrivons à obtenir une partition de l'ensemble de pièces en sous-ensembles solutions, leur combinaison serait encore une solution. Par conséquent, une partition en n sous-ensembles nous donneraient 4^n solutions à **1-ROT**. Par un algorithme naïf de recherche empirique de sous-ensembles solutions de taille minimale, nous obtenons au bout d'une semaine une quarantaine d'ensembles, mais cela ne signifie pas pour autant qu'il existe une partition avec au moins 40 sous-ensembles comme nous allons le voir :

Question 9.2. *Etant donné E, un ensemble de pièces. Est-ce que l'ensemble $\mathcal{F}$ de ses sous-ensembles solution est stable par complémentaire ?*

Il est possible de trouver des contre-exemples pour la stabilité par complémentaire. La réponse à la question est donc non :

FIGURE 9.5 – Sous-ensemble solution, de complémentaire non solution



Malgré l'absence de guarantie de trouver une solution en effectuant un algorithme glouton à cause de l'absence de stabilité par complémentarité, nous pouvons nous baser sur une amélioration de l'algorithme glouton approximant la problème **CYCLE-PACKING** [11] en essayant de l'adapter à nos besoins en y ajoutant des contraintes.

Pour cela, nous allons associer au jeu de pièces le graphe $G = (V, E)$ où chaque sommet est un motif, et associant à chaque pièce 2 arêtes étiquetées par le numéro de la pièce, chacune correspondant à deux côtés opposés de la pièce.

L'algorithme glouton naïf consiste à trouver un plus court cycle dans le graphe associé, puis retirer les arêtes utilisées et réitérer jusqu'à ce que le graphe soit acyclique. Cet algorithme est une $\mathcal{O}(\sqrt{n})$-approximation. L'amélioration de [11] consiste à retirer les sommets de degré 1 et l'arête qui va avec, puis retirer les sommets de degré 2 en fusionnant les arêtes incidentes, de manière à choisir un "meilleur" cycle de degré minimal. Cette amélioration est une $\mathcal{O}(\log n)$-approximation, et fonctionne particulièrement bien avec les graphes denses ($(\frac{2t}{3} + \epsilon)$-approximation où t est défini par $\#E = \Omega\left((\#V)^{1+\frac{1}{t}+\delta}\right)$, ce qui est le cas du graphe associé au jeu d'Eternity II.

Faute de temps, la piste n'a pas été explorée plus en profondeur.

10 Le problème 1-ROT

10.1 Présentation

Le problème **1-ROT** est un relachement des contraintes de **2-ROT** et dont l'étude a pour but de permettre d'améliorer l'algorithme de recherche d'une partition en blocs compatibles **2-ROT**. L'avantage de ce problème est qu'il est résoluble en temps polynomial de manière efficace.

10.2 Études de complexités

Ce problème peut être vu comme la recherche d'une partition des arêtes d'un graphe en cycles. En effet, étant donné un multi-ensemble de pièces, nous pouvons créer un multi-graphe ayant pour sommets les motifs, tel que deux sommets aient autant d'arêtes en commun qu'il n'y a de pièces ayant ces deux motifs.

Étant donné un ensemble solution, l'ensemble des sous-ensembles solutions est clairement stable par complémentaire car si l'on retire autant d'occurrences d'un même motif dans chaque direction, l'égalité du nombres d'occurrences de chaque motif reste inchangé.

Théorème 10.1. *Un jeu de pièces est solution de **1-ROT** si chaque composante connexe du graphe associé est eulérienne.*

Si le graphe associé est eulérien, alors nous pouvons créer une partition regroupant les arêtes par composantes connexes. Si le graphe associé n'est pas eulérien, il existe un sommet d'arité impaire, ce qui implique que le nombre d'occurrences d'un motif est impair, rendant impossible la satisfaction des contraintes.

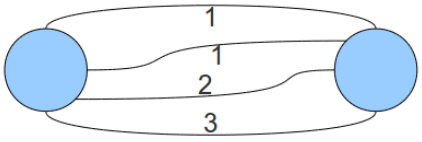
Du théorème 10.1, nous pouvons déduire un algorithme polynomial pour tester l'existence d'une solution.

L'algorithme de recherche de composantes connexes se fait en temps linéaire, de même pour vérifier qu'un graphe est eulérien. Nous pouvons même combiner les deux algorithmes pour effectuer les deux tests dans le même parcours. **1-ROT** est donc dans P.

Variante 1 (**labelled 1-ROT**).
Si l'on associe à chaque arête du graphe associé un entier de 1 à m et que l'on impose qu'il existe une partition en cycles n'ayant que des étiquettes uniques, l'ensemble des sous-ensembles solutions cesse d'être stable par complémentaire.

En effet, soit le graphe de la figure 10.1. L'ensemble des arêtes est solution, et le sous-ensemble $\{2, 3\}$ l'est, mais son complémentaire ne l'est pas.

FIGURE 10.1 – Contre exemple à la stabilité par complémentaire



Nous pouvons considérer que les arêtes peuvent être étiquetées par des ensembles, sachant qu'il suffira d'éclater l'arête en un chemin en rajoutant des sommets entre. L'ensemble vide est également admis car il suffit de le remplacer par un singleton contenant une étiquette unique dans tout le graphe.

Démontrons que ce problème est NP-complet. Pour cela, nous allons réduire **3-SAT** à la variante 1.

Nous pouvons impos(n est le cardinal du multi-ensemble) telle que chaque digit soit 0 ou 1.er qu'une variable et sa négation ne se trouve pas dans la même clause, car dans le cas contraire, il suffit de retirer la clause.

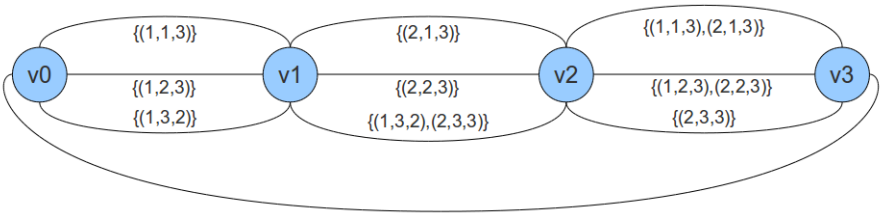
Le problème **3-SAT** est NP-complet. Réduisons donc **3-SAT** à **labelled 1-ROT**. Pour cela, nous allons associer à chaque instance de **3-SAT** un graphe tel que l'instance soit satisfiable si et seulement si le graphe admet un cycle hamiltonien, puis puis rajouter des arêtes de façon à ce que l'existence de cycle hamiltonien soit équivalente à trouver une partition des arêtes en cycles dont toutes les étiquettes sont uniques.

Soit $(C_1, \cdots, C_n)$ une instance de **3-SAT**, avec $C_i = (l_{i,1}, l_{i,2}, l_{i,3})$. Définissons $S := \{(i,j,k), \overline{l_{i,j}} \in C_k\}$. Etant donné un litéral $l_{i,j}$, nous allons associer l'ensemble $U_{i,j} := \{(i,j,k) \in S\} \cup \{(a,b,i) \in S\}$. Nous allons encoder l'instance par un graphe en associant un sommet v_i à chaque clause C_i et ajoutant un sommet v_0. A chaque litéral $l_{i,j}$, nous allons associer une arête entre les sommets v_{i-1} et v_i, étiquetée par $U_{i,j}$. Enfin, une arête relie v_n et v_0. L'ensemble $U^+_{i,j}$ permet de s'assurer que lorsque l'on passe par l'arête encodant $l_{i,j}$, l'assignant ainsi à 1, que l'on ne puisse pas passer par les arêtes encodant les litéraux opposés. $U^-_{i,j}$ contient l'assurance que l'on n'est pas déjà passé par le litéral opposé, car sinon, une des étiquettes serait déjà utilisée bloquant ainsi le passage.

Par exemple, pour $n = 3$ et
- $C_1 = (x_1, \bar{x}_2, x_3)$
- $C_2 = (x_1, \bar{x}_2, \bar{x}_3)$
- $C_3 = (\bar{x}_1, x_2, x_3)$

Le graphe associé est présenté figure 10.2

FIGURE 10.2 – Graphe associé à l'instance **3-SAT**

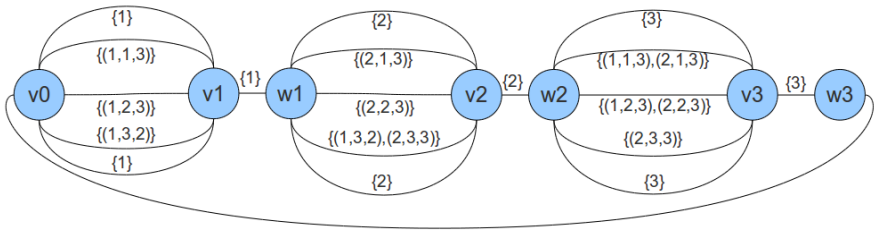


L'instance est satisfiable si et seulement si il existe un cycle hamiltonien ne rencontrant pas 2 fois la même étiquette. En effet, si l'instance est satistifiable, il existe un chemin de v_0 à v_n et ce chemin peut être bouclé par le cycle. Et s'il existe un cycle passant par chacun des v_i, alors au moins 1 litéral de chaque clause est vrai.

Nous allons maintenant ajouter 2 arêtes entre chaque couple de sommets $(v_i, v_{i+1}$ afin que les arêtes qui n'ont pas été utilisées dans le cycle puissent être dans une boucle. Et enfin, pour s'assurer que cela ne va pas ajouter la possibilité d'obtenir un cycle hamiltonien en passant par une des arêtes ajoutées, nous allons scinder chaque sommet en 2, les reliant par la même étiquette que les arêtes ajoutées précédentes.

Toujours avec l'exemple précédent, voici (figure 10.3) le graphe résultant.

FIGURE 10.3 – Graphe final associé à l'instance **3-SAT**



Si l'instance 3-SAT est satisfiable, alors il existe un chemin hamiltonien d'étiquettes distinctes sur le graphe précédent, et une partition des arêtes en cycles d'étiquettes distinctes sur ce graphe. Réciproquement, s'il existe une partition des arêtes en cycles d'étiquettes distinctes, parmi ces cycles, l'un va passer par chaque sommet v_i et il suffit d'assigner 1 à chaque litéral encodant les arêtes par lesquelles on passe pour obtenir une valuation satisfiant l'instance. La variante 1 est donc NP-complète.

Remarque : *On peut imposer que chaque étiquette apparaisse exactement 2 fois, en associant une étiquette différente à chaque litéral apparaissant dans la clause et non une commune à toutes les occurrences de ce litéral dans la clause. Cela revient à créer des quadruplets et non des triplets. Le problème reste alors NP-complet.*

Variante 2 (**1-labelled 1-ROT**).
Au lieu d'autoriser un nombre arbitraire d'étiquettes, on restreint l'ensemble des étiquettes à un singleton $\{e\}$. Ainsi, chaque arête est soit étiquetée par e, soit n'est pas étiquetée.

Nous allons démontrer que ce problème est FPT.

Le problème est NP-complet car il est équivalent à **undirected K EDGE-DISJOINT PATH** quand $G+H$ est eulérien, où H est le graphe de demande. La version orientée est également NP-complete. [12].

Théorème 10.2. *Si l'on fixe le nombre d'arêtes étiquetées,* ***1-labelled 1-ROT*** *est dans P*

Démonstration. Nous allons réduire **1-labelled 1-ROT** à **undirected k DISJOINT PATH** $\forall k$. Pour cela, nous allons utiliser la version "edge-disjoint".

Soit un graphe eulérien non orienté G et (a_i, b_i) des arêtes étiquetées. Il existe une partition des arêtes de G en cycles ne contenant qu'au plus une arête étiquetée si et seulement s'il existe un ensemble de cycles disjoints au niveau des arêtes, contenant exactement une arête étiquetée dans chaque cycle. En effet, en otant un cycle dans un graphe eulérien, on obtient une forêt d'arbres eulériens.

Nous pouvons donc définir G' à partir de G en supprimant les arêtes (a_i, b_i), et tester l'existence de chemins entre les sommets a_i, b_i, disjoints par les arêtes. □

Le problème est également NP-complet si l'on étiquette des sommets au lieu d'arêtes :

1-vertex-labelled 1-ROT : Soit un graphe eulérien et un ensemble de sommets distingués. Existe-t-il une partition des arêtes en cycles ne passant par au plus qu'un sommet distingué ?

Théorème 10.3. ***1-vertex-labelled 1-ROT*** *est NP-complet.*

Démonstration. Le problème est dans NP car la donnée des chemins permet de vérifier en temps polynomial la validité de l'entrée.

Nous allons réduire **undirected k EDGE-DISJOINT PATH** à **1-vertex-labelled 1-ROT**.

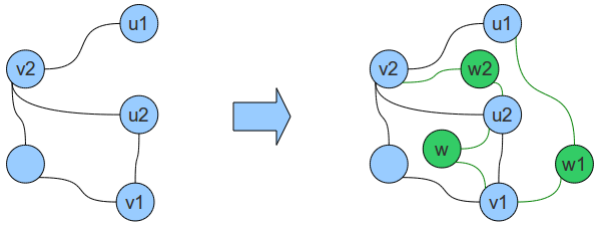
Soit $G = (V, E)$ un graphe non orienté et $(u_i, v_i)_{i \in I}$ un ensemble de couples de sommets. Nous allons construire un graphe G' à partir de G en ajoutant pour chaque $i \in I$ un sommet w_i étiquetée et des arêtes (u_i, w_i) et (w_i, v_i). Enfin, nous allons ajouter un sommet w étiqueté et pour chaque sommet v de degré impair, nous allons ajouter une arête (v, w). (Voir 10.4).

S'il existe un ensemble de chemins disjoints par les arêtes dans G reliant les u_i aux v_i alors il existe un ensemble de cycles disjoints par les arêtes dans G' passant par les sommets u_i, v_i et w_i et n'ayant exactement qu'un sommet étiqueté. Soit C cet ensemble de cycles. Alors $G' - C$ a chacune de ses composantes connexes eulériennes et ne comporte plus qu'un sommet distingué. Il existe donc dans G' une paritition des arêtes en cycles ayant au plus un sommet distingué .

Réciproquement, s'il existe dans G' une partition des arêtes en cycles ayant au plus un sommet distingué, alors soit un cycle passant par un des w_i. Il passe nécessairement par u_i et v_i. En retirant les arêtes (u_i, w_i) et (w_i, v_i) du cycle, nous obtenons un chemin entre u_i et v_i. Si ce chemin passe par une des arêtes ajoutées dans G', sachant que chacune des arêtes ajoutées est incidente à un sommet distingué, le cycle contient au moins deux sommets distingués, ce qui est contraire à l'hypothèse. Le chemin ne contient que des arêtes de G. Les cycles étant distincts par les arêtes, il existe bien dans G un ensemble de chemins disjoints par les arêtes dans G reliant les u_i aux v_i.

□

FIGURE 10.4 – Exemple de transformation en instance **1-vertex-labelled 1-ROT**



10.3 Sous-ensembles solution

Question 10.1. *Etant donné E, un ensemble de pièces. Est-ce que l'ensemble $\mathcal{F}$ de ses sous-ensembles solution est une tribu ?*

$E \in \mathcal{F}$, $\mathcal{F}$ est stable par complémentaire, mais non pas par union dénombrable. En effet, si

$$E = \{(A,B),(B,C),(C,A),(B,A),(A,E),(E,B)\}$$

prenons $I = \{(A,B),(B,C),(C,A)\}$ et $J = \{(A,B),(B,A)\}$. $I \cup J = \{(A,B),(B,C),(C,A),(B,A)\}$ n'est pas un sous-ensemble solution.

11 Stratégie de résolution

Une fois les problèmes précédents étudiés, il convient d'en tirer des conclusions concernant l'intérêt de l'études de certains problèmes par rapport à d'autres. Nous allons donc maintenant essayer de déduire de nos observations précédentes une stratégie de résolution du jeu Eternity II.

11.1 Évaluation de la qualité d'un relachement de contrainte

Un principe intuitif d'évaluation de la qualité d'un problème avec des contraintes relachés est le rapport

$$\frac{\text{nombre de solutions du problème général}}{\text{temps de calcul} * \text{nombre de solutions du problème relaché}}$$

En réalité, ce modèle présente des lacunes, car il ne prend pas en compte le degré de facilité d'énumération des solutions : il se peut qu'il soit dûr de trouver une première solution, mais qu'il soit facile de les énumérer, or lorsque l'on cherche un problème plus contraint, il est courant de couper des branches avec un problème relaché, et d'énumérer les solutions pour les tester ensuite avec le problème contraint.

Nous allons cependant en premier lieu effectuer une approximation le degré de discrimination des problèmes **2-ROT** et **ETER2SET** par rapport à **ETER2**, c'est-à-dire le rapport entre le nombre de solutions du problème général et celui du problème relaché.

FIGURE 11.1 – Nombre moyen de solutions sur 100 instances de taille 3×3

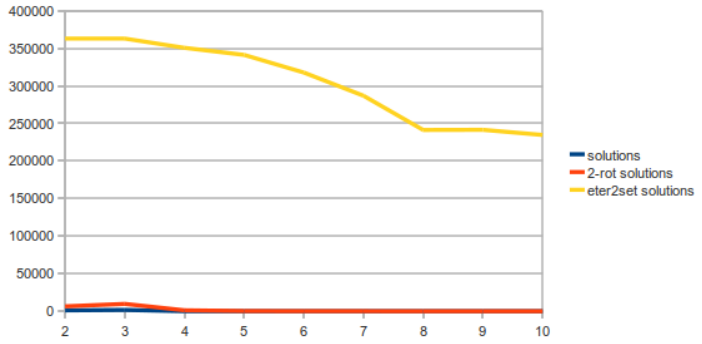


TABLE 11.1 – Résultats d'un test sur 100 instances 3×3 avec 7 couleurs

Problème	# avg solutions	# max solutions	# min solutions	max ratio	min ratio
ETER2	12	84	7	$\frac{7}{7}$	$\frac{7}{7}$
2-ROT	70	714	7	$\frac{14}{14}$	$\frac{7}{714}$
ETER2SET	286790	362880	89280	$\frac{84}{89280}$	$\frac{7}{362880}$

La figure 11.1 et la table 11.1 contiennent les résultats d'un test sur des petites instances générées pseudo-aléatoirement. Faute de capacités de calculs, les échantillons sont extrêmement petits et le comportement peut changer avec l'accroissement de la taille des instances. Il apparaît cependant que le problème **2-ROT** soit beaucoup plus contraint que **ETER2SET**.

11.2 Vers l'ébauche d'un algorithme

Pour résumer ce que nous savons actuellement :

- ✓ Le problème **ETER2** connaît un problème relaché **2-ROT** dont le nombre de solutions est probablement du même ordre
- ✓ Il est possible d'obtenir un grand nombre de solutions à **2-ROT** en trouvant une partition des pièces en formes closes.
- ✓ Il est difficile de trouver une partition des pièces en formes closes.

Partons tout d'abord du principe que nous disposons d'une partition des pièces en n, formes closes et qu'il existe une solution d'Eternity parmi les solutions de **2-ROT** générées. Nous pouvons alors en déduire un algorithme de backtracking qui, lorsqu'il fixe une pièce appartennant à une forme close, fixe en même temps la rotation de chacune des pièces. Alors le nombre de combinaisons de rotations passe de $4^2 56$ à 4^n, avec $n << 256$, ce qui est une amélioration considérable de l'algorithme.

Cependant, en raison de la difficulté de générer une partition en formes closes - un algorithme naïf trouve relativement rapidement (quelques heures) des formes closes de telle sorte qu'il ne reste plus que 60 pièces -, nous pouvons imaginer un raffinement de l'algorithme précédent, lequel commencera par générer une solution partielle de partition en formes closes, et dès lors qu'il aura atteint un seuil de temps de calcul, passera à la phase de backtracking en considérant que les pièces non affectées à une forme close sont des singletons forme close. La question qui se pose est de savoir si l'algorithme pourra effectuer en un temps correct (toujours en heures) une exploration exhaustive de l'arbre afin de pouvoir ensuite revenir à la première phase et explorer ainsi petit à petit toutes solutions partielles.

Faute de temps, un tel algorithme n'a pas encore été réalisé et nous ne disposons donc pas de résultats concernant son apport relativement aux autres algorithmes de résolution.

12 Problèmes liés à Eternity

12.1 Cycles à partir de pièces d'Eternity

Problème (**SQUARE-TRAIL**)
Étant donné un ensemble de pièces comportant chacune 4 motifs, existe-t-il un cycle de telle sorte que les motifs des pièces voisines correspondent ?

Théorème 12.1. ***SQUARE-TRAIL*** *est NP-complet*

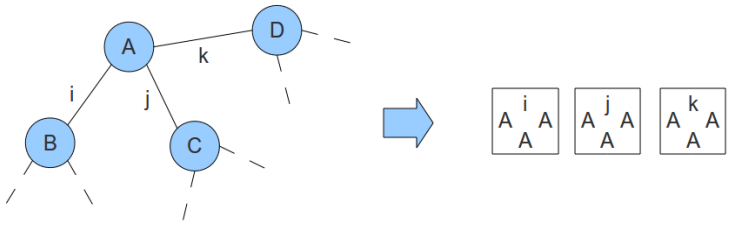
Démonstration. Le problème **SQUARE-TRAIL** est dans NP car étant donné la suite des pièces dans l'ordre du cycle avec la rotation de chaque pièce, il est facile de vérifier en temps polynomial que le tout forme bien un cycle.

Nous allons maintenant réduire **3-HAMILT** à **SQUARE-TRAIL**.

Soit $G = (V, E)$ un graphe 3-régulier.

À chaque sommet, nous allons associer une couleur et 3 pièces. Soit A le sommet, ayant pour arêtes incidentes $\{i = \{A, B\}, j = \{A, C\}, k = \{A, D\}\}$. Nous allons créer les pièces $\{(A, A, A, i), (A, A, A, j), (A, A, A, k)\}$ (figure 12.1).

FIGURE 12.1 – Encodage du sommet A



S'il existe un cycle hamiltonien, alors pour chaque sommet A, le cycle va passer par un couple d'arête, mais pas par les 3 arêtes. Sachant que le motif A n'est présent que dans ces 3 pièces, il faut que l'orientation soit telle que l'on commence et l'on termine par un i,j ou k. Pour chaque couple d'arête, il existe une orientation compatible. Par exemple, pour (i, j), un chemin compatible est $\{(A, A, A, i), (A, A, A, k), (A, j, A, A)\}$. Les arêtes jouant un rôle permutatif, il n'est pas besoin de tester les autres combinaisons et il existe une solution à **SQUARE-TRAIL**. La réciproque est vraie en raison de l'obligation d'assembler les pièces par groupes de 3 à cause de l'unicité des motifs, de la nécessité de passer par tous les sommets et de par l'impossibilité de passer par 3 arêtes en même temps.

La réduction est une logspace réduction car il suffit de stocker la position du sommet courant en binaire dans la bande de travail (sous réserve d'avoir défini une relation d'ordre totale entre les sommets). □

12.2 Placement des pièces de bordure

Problème (**undirected n-ETER2-BORDERS**)
Étant donné un ensemble de pièces de dimension 1 (bordures) et n pièces distinguées (coins), existe-t-il un cycle passant par toutes les pièces tel que les motifs des pièces incidentes correspondent et que les pièces distinguées soient équidistantes ?

Problème (**directed n-ETER2-BORDERS**)
Étant donné un ensemble de pièces de dimension 1 (bordures) et n pièces distinguées (coins), existe-t-il un cycle passant par toutes les pièces tel que les motifs des pièces incidentes correspondent et que les pièces distinguées soient équidistantes ? Les pièces sont orientées. Ainsi $(A, B) \neq (B, A)$.

Le problème **directed 4-ETER2-BORDERS** correspond à la pose des pièces de bordure d'eternity.

Théorème 12.2. ***undirected 2-ETER2-BORDERS*** *est NP-complet*

Démonstration. Le problème est dans NP car étant donné une suite de couples (pièce, rotation), il est facile de vérifier en temps polynomial que cela forme un cycle et que les pièces distinguées sont à la bonne distance les unes des autres.

Nous allons réduire **undirected EULER-HALF-TRAIL-PART** à **undirected 2-ETER2-BORDERS**. **undirected n-ETER2-BORDERS**, $n \geq 2$ s'en déduit ensuite en rajoutant des pièces à motif unique formant des blocs permettant de compléter le polygone.

Soit un graphe eulérien $G = (V, E)$, nous allons encoder chaque sommet par une couleur, et chaque arête entre A et B par une pièce (A, B). Si le graphe admet une partition en 2 cycles de taille $\frac{\#E}{2}$, alors le graphe étant connexe, les cycles partagent au moins un sommet, donc il existe un sommet A tel que si l'on ajoute deux pièces distinguées (A, A), nous obtenons une solution à la variante de **undirected 2-ETER2-BORDERS**. Réciproquement, s'il existe un sommet A tel quel si l'on ajoute les pièces distinguées (A, A), on obtient une solution, alors on peut former 2 cycles de longueur $\frac{\#E}{2}$. □

undirected n-ETER2-BORDERS $\forall n \geq 2$

Théorème 12.3. $\forall n \geq 2$, ***undirected n-ETER2-BORDERS*** *est NP-complet*

Démonstration. Le problème est dans NP car satisfaction des contraintes est vérifiable en temps polynomial. Nous allons réduire **undirected 2-ETER2-BORDERS** à **undirected n-ETER2-BORDERS**.

Étant donnée une instance de **undirected 2-ETER2-BORDERS** contenant 2 pièces distinguées (A, B) et (C, D) et $2m$ pièces non distinguées, nous allons remplacer (C, D) par 2 pièces distinguées $(C, \#)$ et $(\#, D)$, $n - 3$ pièces distinguées $(\#, \#)$ et $(n - 3)m$ pièces non distinguées $(\#, \#)$. Il est clair que s'il existe une solution à **undirected 2-ETER2-BORDERS**, on pourra former des barres de m pièces $(\#, \#)$ séparant les autres pièces distinguées. Réciproquement, du fait de l'unicité du motif $\#$, les pièces de l'instance et celles ajoutées ne pourront pas se regrouper dans une même barre, et donc en reprenant les barres obtenues à partir des pièces de l'instance, nous pouvons former une solution pour **undirected 2-ETER2-BORDERS**. □

directed n-ETER2-BORDERS La démonstration de la NP-complétude de **directed 2-ETER2-BORDERS** est strictement la même. Idem pour **directed n-ETER2-BORDERS**.

12.3 Majoration du nombre de bordures d'Eternity II

12.3.1 Règles du jeu

Le jeu est de taille 16×16, et possède dont 60 pièces de bordure, dont 4 coins et 56 bords stricts. A des fins de simplification, nous allons considérer les coins comme des bords stricts, et rechercher le nombre de cycles de pièces de telle sorte que les modifs des pièces contigües soient les mêmes.

Les motifs sont au nombre de 5.

Du fait de leur motif de bordure gris fixé, les pièces sont orientées.

12.3.2 Modélisation

A chaque couleur est attribué un numéro entre 1 et 5. Du fait de son orientation, chaque pièce de bordure peut être modélisée par un couple $(i, j) \in \{1 \dots 5\}^2$. Nous pouvons donc définir un graphe $G = (E, V)$ avec $E = \{1 \dots 5\}$ et $V = \{(i, j)\}$. C'est donc un pseudographe orienté.

12.3.3 Majoration

Compter le nombre de bordures d'Eternity II revient donc à calculer le nombre de circuits eulériens dans G. Nous allons pour cela utiliser le théorème BEST (de Bruijn, van Aardenne-Ehrenfest, Smith et Tutte). Ce théorème a été prouvé dans un article de van Aardenne-Ehrenfest et de de Bruijn [13], lui même généralisant un cas étudier par Simth et Tutte [14]. Le cas des graphes non-orientés est #P-complet. Voir [15].

Théorème 12.4 (BEST)**.** *Soit G un multigraphe eulérien orienté,* $\deg(v)$ *le degré entrant (= degré sortant) et* $\Delta_w(G)$ *son nombre d'arbres couvrants de racine w et dirigés vers sa racine, alors son nombre de circuits eulériens est*

$$\mathrm{ec}(G) = \Delta_w(G) \prod_{v \in V} \big(\deg(v) - 1\big)!$$

Démonstration. Cette preuve utilise l'interprétation de [15].

Soit $G = (V, E)$ un pseudographe orienté eulérien. Fixons x un sommet de G.

Etant donné un circuit eulérien C et e une arête sortante de x, nous pouvons définir $T(C, e) := \{\mathrm{exit}(v),\ v \neq x\}$ où $\mathrm{exit}(v)$ est la dernière arête sortante de v avant que le circuit ne passe à nouveau par e. $T(C, e)$ est un arbre couvrant.

Nous pouvons donc associer exactement $\deg x$ arbres couvrants à C suivant cette propriété.

Etant donné un arbre couvrant T, il existe $(\deg x)! \prod_{v \in V} (\deg v - 1)!$ couples distincts (C, e) qui lui sont associés. En effet, on passe par chaque arête en s'arrangeant pour que l'arête de l'arbre couvrant soit utilisée en dernier, ce qui nous donne $(\deg x)!$ numérotations pour les arêtes sortantes de x, et $(\deg v - 1)!$ numérotations pour les arêtes sortantes de v.

De plus, les associations des couples (C, e) aux arbres couvrants forment une partition de l'ensemble des couples de circuits eulériens et d'arête sortante de x. Il existe donc

$$\Delta_x \deg(x)! \prod_{v \in V \setminus x} \big(\deg(v) - 1\big)!$$

couples distincts, donc $\Delta_x \deg(x - 1)! \prod_{v \in V \setminus x} \big(\deg(v) - 1\big)!$ circuits eulériens,

$$\mathrm{ec}(G) = \Delta_x \prod_{v \in V} \big(\deg(v) - 1\big)!$$

□

Il est possible de calculer facilement $\Delta_w(G)$ grâce au théorème de Kirchhoff.

Théorème 12.5 (Kirchhoff)**.** *Soit G un pseudographe orienté et M sa matrice de Laplace, Le nombre d'arbres couvrant de G est égal à la valeur absolue de n'importe lequel des cofacteurs de M.*

12.3.4 Applications numériques

Le graphe $G = (\{1, 2, 3, 4, 5\}, \{(1,1), (1,1), (1,1), (1,2), (1,2), (1,2), (1,2), (1,3), (1,3), (1,4), (1,4), (1,5), (2,1), (2,1), (2,1), (2,2), (2,2), (2,4), (2,4), (2,4), (2,5), (2,5), (2,5), (2,5), (3,1), (3,1), (3,2), (3,2), (3,3), (3,3), (3,3), (3,3), (3,4), (3,4), (3,5), (3,5), (4,1), (4,1), (4,2), (4,2), (4,3), (4,3), (4,4), (4,4), (4,4), (4,5), (4,5), (4,5), (5,1), (5,1), (5,2), (5,2), (5,3), (5,3), (5,3), (5,3), (5,4), (5,4), (5,5), (5,5)\})$

La matrice d'ajacence est donc

$$\begin{pmatrix} 3 & 4 & 2 & 2 & 1 \\ 3 & 2 & 0 & 3 & 4 \\ 2 & 2 & 4 & 2 & 2 \\ 2 & 2 & 2 & 3 & 3 \\ 2 & 2 & 4 & 2 & 2 \end{pmatrix}$$

Sa matrice de Laplace est

$$\begin{pmatrix} 9 & -4 & -2 & -2 & -1 \\ -3 & 10 & -0 & -3 & -4 \\ -2 & -2 & 8 & -2 & -2 \\ -2 & -2 & -2 & 9 & -3 \\ -2 & -2 & -4 & -2 & 10 \end{pmatrix}$$

Nous obtenons donc $\Delta_w(G) = 3432$.

En outre, $\deg(w) = 12$ pour tout sommet w. Nous obtenons donc une majoration de

$$3432 * (11!)^5 = 34779703024910365783503327461376000000000$$

13 Problèmes hors contexte

13.1 Démonstration du théorème BEST

Ce théorème a fait de notre part plusieurs tentatives de démonstrations ayant échouées. Nous nous sommes basés sur la démonstration de C. Berge dans son ouvrage [16]. Citons-le :

Considérons une arborescence H de racine x_1 qui soit un graphe partiel de G et montrons qu'il y a exactement

$$\prod_{v \in V} \big(\deg(v) - 1\big)!$$

circuits eulériens pour lesquels les arcs nous amenant à un sommet pour le première fois sont ceux de H.

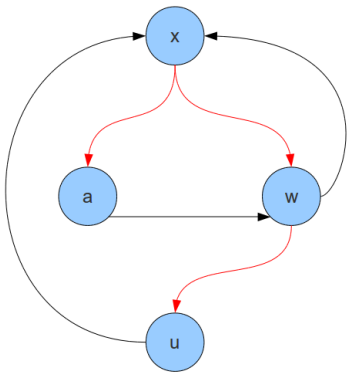
13.1.1 Première interprétation

Soit Δ_x le nombre d'arbres couvrant partant de la racine. Nous avons voulu démontrer que le nombre de circuits eulériens tels que pour chaque sommet, on arrive la première fois à celui-ci par une arête de l'arbre est

$$\prod_{v \in V} \big(\deg(v) - 1\big)!$$

En se basant sur l'idée suivante : pour tout sommet et pour toute permutation d'ordre d'arête entrante de chaque sommet (on fixe l'arête de l'arbre comme étant la dernière), il existe un graphe passant par les arêtes de chaque sommet dans cet ordre, soit $\prod_{v \in V} \big(\deg(v) - 1\big)!$ combinaisons possibles. Cependant, il s'avère qu'étant donnée une numérotation, plus d'un cycle eulérien soit associé :

FIGURE 13.1 – Graphe et arbre couvrant associé



Dans l'exemple précédent, sachant que l'on a fixé à chaque sommet l'arête de l'arbre comme étant la dernière, il n'existe qu'une seule numérotation, tandis qu'il existe plusieurs cycles eulériens :

x -> w -> x -> a -> w -> u et x -> w -> u -> x -> a -> w

13.1.2 Seconde interprétation

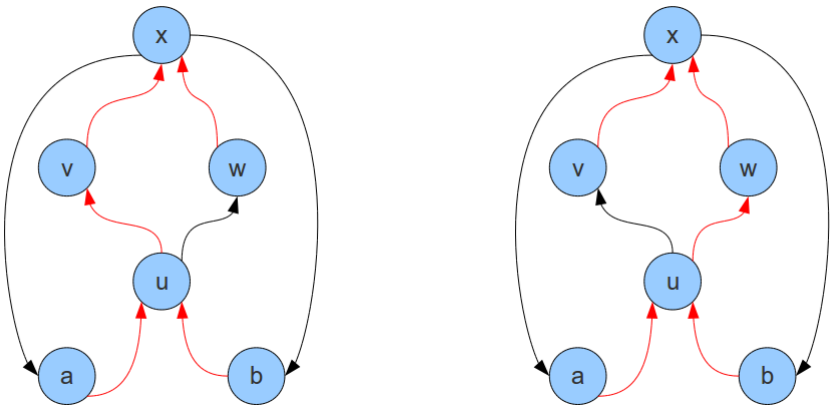
Cette seconde interprétation s'inspire de la preuve de [15].

Soit Δ_x le nombre d'arbres couvrant pointant vers la racine. Nous avons voulu démontrer que le nombre de circuits eulériens tels que pour chaque sommet, lorsque celui-ci passe par une arête de l'arbre, celui-ci ne réapparaît plus dans le circuit est

$$\prod_{v \in V} (\deg(v) - 1)!$$

Cependant, il s'est avéré que cette propriété associait un même circuit eulérien à plusieurs arbres couvrants :

FIGURE 13.2 – Arbres couvrants d'un graphe de contre-exemple



Voici un cycle eulérien associé aux deux arbres couvrants : x -> b -> u -> v -> x -> a -> u -> w = x -> a -> u -> w -> x -> b -> u -> v

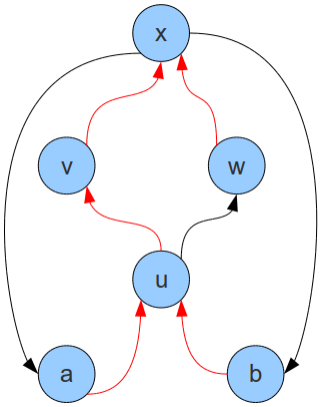
13.1.3 Troisième interprétation

Soit Δ_x le nombre d'arbres couvrant pointant vers la racine. Essayons de prouver que le nombre de circuits eulériens tels que pour chaque sommet, on arrive la première fois à celui-ci par une arête de l'arbre est

$$\prod_{v \in V} (\deg(v) - 1)!$$

Cette propriété n'est pas bien-définie comme le montre le graphe suivant :

FIGURE 13.3 – Graphe et arbre couvrant associé



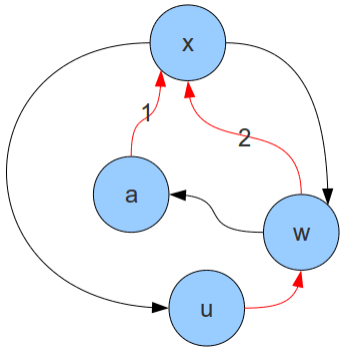
Dans ce cas-ci, w n'a pas d'arête entrante provenant de l'arbre.

13.1.4 Quatrième interprétation

Soit Δ_x le nombre d'arbres couvrant pointant vers la racine. Essayons de prouver que le nombre de circuits eulériens tels que pour chaque sommet, lorsqu'on entre la première fois chez celui-ci, on sorte par une arête de l'arbre (propriété bien-définie car l'arbre est couvrant).

Si on base la numérotation sur les arêtes entrantes, on n'aura pas un graphe associé à chaque numérotation comme il suffit de le voir dans le graphe suivant :

FIGURE 13.4 – Graphe, arbre couvrant associé et numérotation des arêtes entrantes



Si on se base sur la numérotation des arêtes sortantes, à chaque numérotation est associé un unique circuit eulérien. En effet, sachant que le graphe est eulérien, on ne se retrouvera jamais bloqué, donc il existe un circuit eulérien pour chaque numérotation. Et son unicité vient de l'aspect déterministe dû à la numérotation. Il reste à démontrer que chaque circuit eulérien n'est associé qu'à un unique arbre couvrant dirigé vers sa racine x. Mais là encore, le contre exemple de la seconde interprétation s'applique et un même circuit eulérien est associé à plusieurs arbres.

13.1.5 Interprétation des échecs

Les preuves de [16] et [15] utilisent implicitement l'explosion des circuits eulériens en $\deg x$ chemins afin d'établir une partition de l'ensemble de ces chemins, puis diviser le tout par $\deg x$. Cette transformation implicite a porté à confusion.

13.2 Mesure dynamic alternative de complexité

Une démonstration erronée conduisant à une contradiction permet parfois d'obtenir d'autres résultats en s'apercevant notamment que l'un des prérequis n'est pas satisfait. Cette démonstration peut alors être retransformée en une preuve valide de l'insatisfaction d'un des prérequis. C'est ce qui m'est arrivé lorsque je me suis posé la question de la complexité d'une machine de Turing universelle restreinte à la classe P. La preuve de l'erreur de mon raisonnement était qu'il impliquait que $P \neq NP$ or la preuve supportait la relativisation et donc on obtenait une contradiction si l'on prenait un oracle tel que $P^A = NP^A$ comme prouvé dans [17].

Par manque de temps et sachant que ce n'était pas le sujet du stage, seules les notions nécessaires au théorème sont introduite. Le théorème lui-même n'est pas un résultat très intéressant, mais présente l'avantage de mettre en relief l'essence même de l'erreur de mon raisonnement.

Définition 13.1. *Un langage $L \subseteq \Sigma^*$ est en*

- *$DCERTTIME(f)$ si son certificat est vérifié par une machine de Turing déterministe f-bornée dans le temps.*

- $NCERTTIME(f)$ *si son certificat est vérifié par une machine de Turing non déterministe f-bornée dans le temps.*

Définition 13.2.
- $PCERT := \bigcup_{k\in\mathbb{N}} DCERTTIME(n^k)$
- $NPCERT := \bigcup_{k\in\mathbb{N}} NCERTTIME(n^k)$

Proposition 13.1. $PCERT = NP$

Démonstration. Immédiat d'après les définitions. □

Proposition 13.2. $PCERT = NPCERT$

Démonstration. Il suffit de remarquer que le certificat de la machine de Turing vérifiant la validité du certificat est un certificat valide pour la première machine. □

Théorème 13.1. *Si* $P = NP$, *alors* $\forall k \in \mathbb{N}$, $P - DCERTTIME(n^k) \neq \emptyset$

Démonstration. Soit U, une machine de Turing universelle restreinte à P, ie. prenant en entrée l'encodage d'un couple (machine de Turing M de langage d'acceptation dans P, entrée x de M). Une telle machine existe du fait de l'existence de machine de Turing universelle. S'il existe un $k \in \mathbb{N}$ tel que $P - DCERTTIME(n^k) = \emptyset$, alors $U \in NP$ car toute entrée admet un certificat vérifiable en temps $\mathcal{O}\left(n^k\right)$, or $P = NP$. Il existe donc un k tel que U s'exécute en temps $\mathcal{O}\left(k\right)$. or d'après le théorème de hiérarchie [18], il existe une machine de Turing M, avec $L = \mathcal{L}(M)$ tel que $L \in PTIME(n^{k+1}) - PTIME(n^k)$, or U', la machine de Turing obtenue à partir de U en fixant la machine de Turing d'entrée à M. Alors $\mathcal{L}(U') = L$ et U' s'exécuterait en $\mathcal{O}\left(n^k\right)$ ce qui est contradictoire. □

14 Conclusion

L'approche de cette étude n'a pas été la tentative de résolution directe du problème **ETER2** à l'aide d'un paradigme de résolution connue, mais d'étudier diverses variantes dont la résolution permettrait d'obtenir une solution d'**ETER2**, déplaçant ainsi le sujet d'étude sur la résolution de la variante.

Cette étude a permis de mettre en relief le problème **2-ROT** présentant les qualités requises. Le manque de temps a empêché d'explorer plus profondément ce problème. La prochaine étape consiste donc à créer des heuristiques permettant de trouver rapidement des partitions en formes closes au sens de **2-ROT** conduisant ensuite à une exploration exhaustive de l'arbre de résolution de **ETER2** par un algorithme de backtracking adapté.

D'après l'hypothèse $P \neq NP$, il n'existe probablement pas d'algorithme polynomial permettant de résoudre l'instance généralisée d'Eternity, mais d'après l'auteur du document, la taille de l'instance est suffisament petite pour que l'on trouve dans les années qui viennent un algorithme permettant de résoudre efficacement les instances de même taille.

Bibliographie

[1] R. Berger. *The undecidability of the domino problem.* Amer Mathematical Society, 1966.

[2] T. Jolivet. Eternity II : puzzles, algorithmique et complexité.

[3] T. Benoist and E. Bourreau. La programmation par contraintes à l'attaque d'Eternity II. 2008.

[4] P. Schauss and Y. Deville. Hybridation de la programmation par contraintes et d'un voisinage à très grande taille pour Eternity II. 2008.

[5] Erik D. Demaine and Martin L. Demaine. Jigsaw puzzles, edge matching, and polyomino packing : Connections and complexity. *Graph. Comb.*, 23 :195–208, February 2007.

[6] V. Klee, R. Ladner, and R. Manber. Signsolvability revisited. *Linear algebra and its applications*, 59 :131–157, 1984.

[7] M.R. Garey and D.S. Johnson. *Computers and intractability. A guide to the theory of NP-completeness. A Series of Books in the Mathematical Sciences.* WH Freeman and Company, San Francisco, Calif, 1979.

[8] A.S. Lapaugh and C.H. Papadimitriou. The even-path problem for graphs and digraphs. *Networks*, 14(4) :507–513, 1984.

[9] G.C. Michele Conforti and K.V. Ajai Kapoor. Even-hole-free graphs part I : Decomposition theorem. *Journal of Graph Theory*, 39(1) :6–49, 2002.

[10] M. Brundage. *From the Even Cycle Mystery to the L-Matrix Problem and Beyond.* PhD thesis, University of Washington, 1996.

[11] A. Caprara, A. Panconesi, and R. Rizzi. Packing cycles in undirected graphs. *Journal of Algorithms*, 48(1) :239–256, 2003.

[12] J. Vygen. NP-completeness of some edge-disjoint paths problems. *Discrete Applied Mathematics*, 61(1) :83–90, 1995.

[13] T. van Aardenne-Ehrenfest and NG BRUIJN. Circuits and trees in oriented linear graphs. *Classic papers in combinatorics*, page 149, 1987.

[14] CAB Smith and W.T. Tutte. On unicursal paths in a network of degree 4. *Amer. Math. Monthly*, 48(233-237), 1941.

[15] G.R. Brightwell and P. Winkler. Note on counting eulerian circuits. *lanl. arXiv. org*, 2008.

[16] Berge C. *Théorie des graphes et ses applications.* Collection Universitaire de Mathématiques, 1958.

[17] T. Baker, J. Gill, R. Solovay, S.A. Kurtz, S.R. Mahaney, and J.S. Royer. Relativizations of the P=? NP question. *J. ACM*, 42 :401–420, 1975.

[18] J. Hartmanis and R.E. Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117(5) :285–306, 1965.

[19] R.M. Karp. Reducibility among combinatorial problems. *50 Years of Integer Programming 1958-2008*, pages 219–241, 2010.

[20] T.J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, page 226. ACM, 1978.

[21] C. Picouleau. Complexity of the hamiltonian cycle in regular graph problem. *Theoretical computer science*, 131(2) :463–473, 1994.

[22] C. Thomassen. Even cycles in directed graphs. *European. J. Combin*, 1985.

[23] N. Robertson, P.D. Seymour, and R. Thomas. Permanents, Pfaffian orientations, and even directed circuits. *Annals of Mathematics*, 150(3) :929–975, 1999.

[24] R. M. Karp. *On the computational complexity of combinatorial problems.* Networks 5, 1975.

[25] J.F. Lynch. The equivalence of theorem proving and the interconnection problem. *ACM SIGDA Newsletter*, 5(3) :31–36, 1975.

[26] N. Robertson and P.D. Seymour. Graph minors. XIII. The disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63(1) :65–110, 1995.

[27] S. Fortune, J. Hopcroft, and J. Wyllie. The directed subgraph homeomorphism problem* 1. *Theoretical Computer Science*, 10(2) :111–121, 1980.

[28] MR Kramer and J. van Leeuwen. The complexity of wire-routing and finding minimum area layouts for arbitrary VLSI circuits, FP Preparata (ed.),\ Advances in Computing Research", Bd. 2 : VLSI theory, 1984.

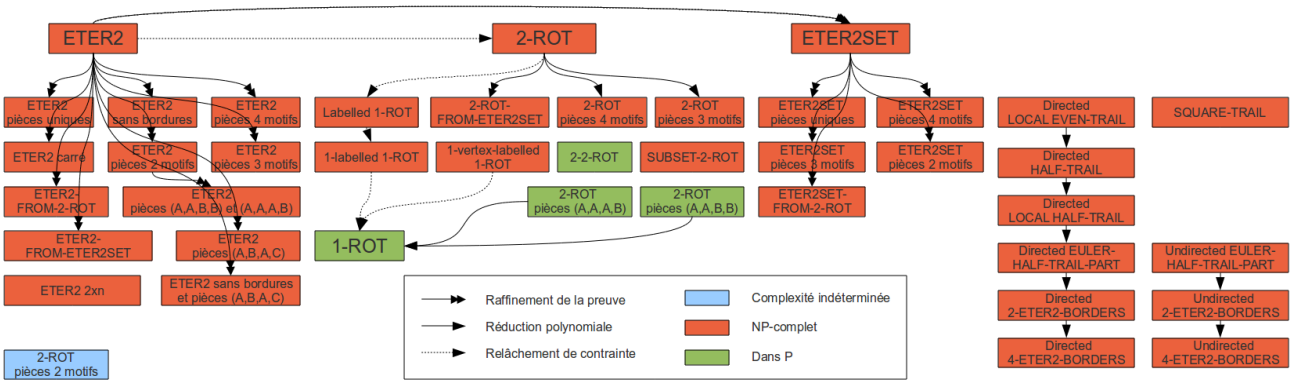
[29] P.D. Seymour. Directed circuits on a torus. *Combinatorica*, 11(3) :261–273, 1991.

Index des problèmes

Index des réductions

Index des complexités

Annexe 1 : Arbre des démonstrations de complexités



Annexe 2 : Définition des problèmes utilisés pour les réductions

Problème (**SUBSET-SUM**)
Étant donné un multi-ensemble d'entiers relatifs non nuls, existe-t-il un sous-ensemble de somme nulle ? De manière équivalente, pour un multi-ensemble d'entiers naturels, existe-il un sous-ensemble de somme égale à un nombre donné ? [19]

Problème (**PARTITION**)
Étant donné un multi-ensemble d'entiers, existe-t-il une partition en 2 sous-ensembles de même somme ? [19]

Problème (**3-PARTITION**)
Étant donné un multi-ensemble de $3m$ *entiers naturels et de somme totale* Sm*, existe-t-il une partition en* m *triplets de somme* S *? [7]*

Nous pouvons imposer que le domaine des entiers naturels soit polynomial en fonction de m. Cela permet donc l'encodage en notation unaire. Nous pouvons également ajouter la contrainte selon laquelle les seuls sous-ensembles de somme S sont les triplets.

Problème (**3-SAT**)
Soit une formule logique en CNF ayant exactement 3 litéraux par clause, admet-elle une valuation sur les variables propositionnelles telle que la formule soit vraie ? [19]

Problème (**monotone One-in-three 3SAT**)
Une formule 3SAT sans négation admet-elle une valuation telle que chaque clause vérifie exactement 1 litéral ? [20]

Problème (**3-HAMILT**)
Étant donné un graphe cubique non-orienté, existe-t-il un cycle hamiltonien ?

Ce problème est prouvé NP-complet dans [7]. La NP-complétude des problèmes **n-HAMILT**, $n \geq 3$ est établie dans [21].

Problème (**directed LOCAL EVEN (resp. ODD)-CYCLE**)
Étant donné un graphe orienté et un arc, existe-t-il un cycle élémentaire de longueur paire (resp. impaire) qui passe par cet arc ? [6], [22].

La classe de complexité du problème **directed EVEN-CYCLE** consistant à savoir si un graphe orienté admet un cycle élémentaire de longueur paire a longtemps été est dans P [23]. Le problème consistant à savoir si un graphe orienté admet un cycle élémentaire de longueur impaire est dans P car les graphes n'en possédant pas sont les graphes bipartis.

Problème (**k DISJOINT PATH**)
Étant donné un graphe et k *couples de sommets* (a_i, b_i)*, existe-t-il* k *chemins disjoints (vertex-disjoint) entre les* a_i *et* b_i*.*

Le problème est NP-complet dans les cas orientés et non orientés [24],[25]. Dans le cas non orienté, pour k fixé, il est dans P [26]. Il est NP-complet pour $k = 2$ dans le cas orienté [27] et pour k non fixé pour la classe des graphes planaires [28],[25]. En revanche, il est dans P pour $k = 2$ avec des graphes planaires. [29].