INFO-F302 — Projet
Calcul du nombre d'Alcuin d'un graphe

Année académique 2024-2025

1 Consignes (important)

Premièrement, nous vous encourageons à d'abord formaliser votre solution sur papier, avant toute implémentation (ce qui vous est de toute façon demander dans le rapport), la recherche de bug dans votre modélisation SAT étant peu aisée.

Ce projet est à réaliser en *binômes* et doit être rendu sur l'Université Virtuelle pour le **lundi 2 décembre 2024 à 23h59**. Vous devez remettre un unique fichier au format ZIP. En particulier, pas de RAR ou TAR.GZ !

Cette archive doit s'appeler : `MATRICULE1_MATRICULE2.zip`.

Elle doit contenir :

1. votre code *clair et commenté* en Python 3 (qui sera testé avec Python >=3.11) ;
2. votre rapport au format PDF avec le fichier source. Ce dernier peut être écrit en LaTeX ou avec `Typst`, à votre préférence.

Les fichiers nécessaires sont à votre disposition sur l'UV. Pour le code, vous ne devez remettre que votre fichier `project.py`, pas les fichiers `tests.py` et `utils.py` (en particulier, vous ne pouvez pas les modifier).

Les remises sont acceptées jusqu'au 6 décembre à 23h59 avec 2 points de pénalité par jour (en particulier rendre le projet le 6 décembre vous donnera au plus 12/20).

2 Calcul du nombre d'Alcuin d'un graphe

Considérons l'énigme suivante : une chèvre, un loup, un chou et un berger doivent traverser une rivière, à bord d'une barque qui contient une unique place passager et une place pour le berger (qui est le seul à pouvoir conduire la barque). Il sera nécessaire de faire plusieurs allers et retours. Seulement voilà, la chèvre et le chou ne peuvent se retrouver sans le berger, sinon la chèvre mangerait le chou. Même problème pour le loup et la chèvre : s'ils sont sans le berger, le loup mange la chèvre. Est-il possible de faire traverser tout le monde ?

On peut modéliser ce problème par la donnée d'un graphe non orienté à trois sommets $\{G, W, C\}$ (*goat*, *wolf*, *cabbage*), et une relation de "conflit" entre sujets donnée par les arêtes $\{G, W\}$ et $\{G, C\}$.

Le problème est donc représenté par le graphe $G_0 = (S_0, E_0) = (\{G, W, C\}, \{\{G, W\}, \{G, C\}\})$ (avec S_0 l'ensemble des sommets du graphe et E_0 l'ensemble des arêtes représentant les conflits). Il y a deux types de mouvements : les mouvements de la rive 0 vers la rive 1, qui alternent avec les mouvements de la rive 1 vers la rive 0. Chaque mouvement est donné par un sous-ensemble X de taille ≤ 1 de l'ensemble de sommets (les sujets/objets qui sont transportés dans la barque).

Un mouvement X est valide si tout $x \in X$ est bien présent sur la rive dont le mouvement part. On cherche donc une suite de mouvements valides tels qu'à la fin, tous les sujets sont sur la rive 1, et à aucun moment il n'y avait une rive sans berger qui contenait deux sujets/objets en conflit.

Formalisation du problème. On peut généraliser ce problème à un nombre quelconque de sujets et une relation de conflit arbitraire. Formellement, on se donne en entrée un graphe non-orienté $G = (S, E)$. Chaque sommet représente un sujet, et chaque arête un conflit entre sujets. On peut représenter toute configuration possible du problème par la donnée d'un élément $b \in \{0, 1\}$ qui indique si le berger est du côté de la rive 0 ou du côté de la rive 1, et la donnée d'une partition des sommets entre $S_0 \subseteq S$ ceux qui sont sur la rive 0 et $S_1 \subseteq S$ ceux qui sont sur la rive 1, avec $S_0 \cap S_1 = \emptyset$ et $S_0 \cup S_1 = S$. Une configuration (b, S_0, S_1) est valide si S_{1-b} ne contient pas deux sommets i, j tel que $\{i, j\} \in E$.

On peut passer d'une configuration $c = (b, S_0, S_1)$ à une configuration $c' = (b', S'_0, S'_1)$ (en une étape) si :

1. $b = 1 - b'$;
2. et il existe un sous-ensemble $X \subseteq S_b$ tel que (a) $S'_b = S_b \setminus X$ et (b) $S'_{1-b} = S_{1-b} \cup X$.

On notera $c \to_X c'$ lorsqu'un tel mouvement est possible. La configuration initiale est $c_0 = (0, S, \emptyset)$, et on cherche à atteindre la configuration $c_F = (1, \emptyset, S)$ en ne passant que par des configurations valides. Autrement dit, on cherche une suite :

$$s : c_0 \to_{X_1} c_1 \to_{X_2} c_2 \cdots c_{n-1} \to_{X_n} c_n = c_F$$

telle que tous les c_i sont des configurations valides. On dira dans ce cas que s est correcte.

Si nous reprenons l'énigme avec le chou, la chèvre et le loup, on a par exemple les séquences suivantes :

— $s_1 : (0, \{G, W, C\}, \emptyset) \to_{\{G,W,C\}} (1, \emptyset, \{G, W, C\})$ qui est correcte
— $s_2 : (0, \{G, W, C\}, \emptyset) \to_{\{G\}} (1, \{W, C\}, \{G\}) \to_{\emptyset} (0, \{W, C\}, \{G\}) \to_{\{W,C\}} (1, \emptyset, \{W, C, G\})$ qui est correcte
— $s_3 : (0, \{G, W, C\}, \emptyset) \to_{\{W\}} (1, \{G, C\}, \{W\}) \to_{\emptyset} (0, \{G, C\}, \{W\}) \to_{\{G,C\}} (1, \emptyset, \{G, C, W\})$ qui n'est pas correcte car la configuration $(1, \{G, C\}, \{W\})$ n'est pas valide (la chèvre mange le chou).

Pour une séquence correcte s, on notera :

$$\text{Alcuin}(s) \coloneqq \max_{1 \le i \le n} |X_i|,$$

donc Alcuin(s) correspond au nombre maximal de sujets ayant été transportés simultanément sur la barque durant les étapes de la séquence s. Par exemple, on a Alcuin$(s_1) = 3$ et Alcuin$(s_2) = 2$. Le *nombre d'Alcuin* de G est :

$$\text{Alcuin}(G) = \min \{\text{Alcuin}(s) \mid s \text{ est une séquence correcte}\}.$$

Pour l'exemple ci-dessus, on a Alcuin$(G) = 1$. Résoudre l'énigme revient à trouver une séquence correcte qui prouve que Alcuin$(G) = 1$.

Exemple. Prenons un autre exemple que l'énigme. Soit $n \ge 2$ un entier. Prenons un graphe G_n à $n + 1$ sommets $S = \{0, 1, \ldots, n\}$ et pour ensemble d'arêtes $E = \{\{0, i\} \mid 1 \le i \le n\}$. Autrement dit, le sommet 0 est relié à tous les autres sommets, qui par contre ne sont pas reliés entre eux.

On va démontrer que $\text{Alcuin}(G_n) = 2$. En effet, il suffit de toujours garder le sommet 0 dans le bateau, et faire des allers et retours pour transporter un par un les sommets restants. Après le dernier aller, on dépose également le sommet 0 sur la rive 1. Cela donne la suite :

$$\begin{array}{llll}
(0, \{0, 1, \ldots, n\}, \emptyset) & \to_{\{0,1\}} & (1, \{2, \ldots, n\}, \{0, 1\}) & \to_{\{0\}} \\
(0, \{0, 2, \ldots, n\}, \{1\}) & \to_{\{0,2\}} & (1, \{3, \ldots, n\}, \{0, 1, 2\}) & \to_{\{0\}} \\
(0, \{0, 3, \ldots, n\}, \{1, 2\}) & \to_{\{0,3\}} & (1, \{4, \ldots, n\}, \{0, 1, 2, 3\}) & \to_{\{0\}} \\
 & \vdots & & \\
(0, \{0, n\}, \{1, \ldots, n-1\}) & \to_{\{0,n\}} & (1, \emptyset, \{0, 1, \ldots, n\}) &
\end{array}$$

Avec un bateau à une place seulement, il n'existe pas de séquence correcte. En effet, comme $n \geq 2$, il y a au moins 2 sommets en plus du sommet 0, et au moins un de ces deux sommets devra se retrouver seul à un moment avec le sommet 0.

Pour calculer le nombre d'Alcuin d'un graphe, on pourra exploiter le résultat suivant qui borne linéairement la longueur des séquences correctes suffisantes :

Théorème 1 ([1])**.** *Pour tout graphe G à n sommets, il existe une séquence correcte s de longueur au plus $2n + 1$ telle que* $\text{Alcuin}(G) = \text{Alcuin}(s)$.

Question 1 Étant donné un graphe non-orienté G à n sommets et un entier k, construire une formule en FNC $\phi_{G,k}$ qui est satisfaisable si et seulement si $\text{Alcuin}(G) \leq k$. La donner et l'expliquer dans le rapport.

Question 2 Implémenter en Python, avec la librairie Python-SAT, la construction de la formule précédente et sa résolution. Utilisez la méthode de test fournie pour tester votre implémentation. Commentez votre code. Votre fonction devra retourner `None` si aucune solution n'a été trouvée, ou une liste de triplets $(b_i, S_{0,i}, S_{1,i})$ représentant la solution trouvée.

Question 3 En se basant sur l'implémentation de la question 2, implémenter un algorithme qui, étant donné un graphe G non-orienté, calcul son nombre d'Alcuin. Expliquer comment vous faites dans le rapport et commentez votre code. Expliquer également dans le rapport en quoi votre implémentation est efficace.

3 Généralisation avec compartiments

Dans l'énigme précédente, on suppose que les sujets en conflit qui sont sur le bateau sont sous surveillance. Dans cette généralisation, ce n'est plus le cas, et il sera nécessaire de compartimenter le bateau pour mettre à part les sujets en conflit. Le problème précédent est le cas particulier où il y a exactement $\text{Alcuin}(G)$ compartiments de capacité 1 chacun. Dans cette généralisation, pour un nombre de compartiments $c \geq 1$, on dira qu'un mouvement $(b, S_0, S_1) \to_X (1 - b, S'_0, S'_1)$ est c-valide si en plus des conditions de la section précédente, X peut être partitionné en c ensembles $X_1, \ldots, X_c$ tels que chaque X_i est un stable de G, c'est-à-dire que chaque X_i ne possède pas deux sommets i, j tels que $\{i, j\}$ est une arête du graphe. Une séquence sera c-valide si elle est correcte et tous les mouvements qui la composent sont c-valides. On définit :

$$\text{Alcuin}_c(G) = \min \{\text{Alcuin}(s) \mid s \text{ est une séquence } c\text{-valide}\},$$

en prenant comme convention que le minimum d'un ensemble vide est $+\infty$.

Remarquez que plus on a de compartiments, plus on a de possibilités de transporter des sujets en conflit, et plus le nombre d'Alcuin sera petit :

$$\text{Alcuin}_1(G) \geq \text{Alcuin}_2(G) \geq \cdots \geq \text{Alcuin}_{\text{Alcuin}(G)}(G) = \text{Alcuin}(G)$$

Exemple. Reprenons l'exemple du graphe G_n de la section précédente. Nous avions $\text{Alcuin}(G_n) = 2$. Remarquez que $\text{Alcuin}_2(G_n) = 2$. En effet, lorsqu'on a deux compartiments, on peut s'en sortir en ne gardant qu'un seul sommet par compartiment. La séquence correcte donnée dans l'exemple de la section précédente est également 2-valide. En effet, les ensembles de sommets qui voyagent sont soit $\{0\}$, soit de la forme $\{0, i\}$ pour tout $1 \leq i \leq n$. On peut bien les partionner en deux sous-ensembles sans conflits.

Par contre, si nous n'avons qu'un seul compartiment ($c = 1$), alors il faudra beaucoup plus de places dans le bateau. Comme 0 est en conflit avec tout le monde et que nous n'avons qu'un seul compartiment, alors 0 doit voyager tout seul. De plus, 0 ne peut rester sur une rive avec d'autres sommets. Le mieux qu'on puisse faire est la stratégie suivante : lors du premier aller, on doit prendre le sommet 0 tout seul, et le déposer sur la rive 1. On revient sur la rive 0, et on peut prendre la moitié des sommets, les déposer sur la rive 1, reprendre le sommet 0, le déposer sur la rive 0 et repartir avec l'autre moitié des sommets. Cela donne la séquence 1-valide suivante (où $m = \lceil n/2 \rceil$) :

$(0, \{0, \ldots, n\}, \emptyset)$	$\rightarrow_{\{0\}}$	$(1, \{1, \ldots, n\}, \{0\})$	$\rightarrow_{\emptyset}$
$(0, \{1, \ldots, n\}, \{0\})$	$\rightarrow_{\{1,\ldots,m\}}$	$(1, \{m+1, \ldots, n\}, \{0, 1, \ldots, m\})$	$\rightarrow_{\{0\}}$
$(0, \{0, m+1, \ldots, n\}, \{1, \ldots, m\})$	$\rightarrow_{\{m+1,\ldots,n\}}$	$(1, \{0\}, \{1, \ldots, n\})$	$\rightarrow_{\emptyset}$
$(0, \{0\}, \{1, \ldots, n\})$	$\rightarrow_{\{0\}}$	$(1, \emptyset, \{0, \ldots, n\})$	

Cela montre que $\text{Alcuin}_1(G_n) \leq \lceil n/2 \rceil$. On peut également montrer que $\text{Alcuin}_1(G_n) \geq \lceil n/2 \rceil$.

Question 4 Étant donnés un graphe non-orienté G à n sommets et deux entiers k et c, construire une formule en FNC $\phi_{G,k,c}$ qui est satisfaisable si et seulement si $\text{Alcuin}_c(G) \leq k$. Donner la formule et l'expliquer dans le rapport.

Question 5 Implémenter la question 4 avec la librairie Python-SAT et tester avec la méthode fournie. Commentez votre code.

Question 6 Implémenter un algorithmique qui utilise le code de la question précédente, et calcule $\text{Alcuin}_c(G)$, étant donnés G et c. Tester avec la méthode fournie. Expliquer votre méthode dans le rapport et commentez votre code. Votre fonction devra retourner `None` si aucune solution n'est trouvée ou une liste de 4-uplets $(b_i, S_{0,i}, S_{1,i}, (X_{1,i}, X_{2,i}, \ldots, X_{c,i}))$ représentant la solution trouvée.

4 Bonus sur 2 points : une variante sans conflits mais avec des durées de traversée

Cette partie est en bonus et compte pour 2 points. Elle est substantiellement plus difficile que les deux parties précédentes réunies ! Nous ne vous recommendons de la commencer que si vous avez déjà *entièrement* fini les questions 1 à 6.

On considère maintenant un problème légèrement différent. Cette fois il n'y a pas de conflits, mais chaque sujet prend un temps différent pour traverser la rivière. Pour illustrer ce nouveau problème, considérons l'énigme suivante. Quatre sujets A, B, C, D veulent traverser une rivière, de nuit, en utilisant un pont qui ne peut accueillir que deux sujets à la fois. Il fait nuit, et les sujets ont impérativement besoin d'une lampe pour traverser, mais il n'y a qu'une seule pour tout le monde. Si A et B traversent en premier, ils prennent la lampe, mais alors un des deux, A

ou B, doit rapporter la lampe à C et D. Si c'est A qui rapporte la lampe, alors il restera A, C, D d'un côté et B de l'autre. Chaque sujet s prend un temps $t(s) \in \mathbb{N}$ pour traverser, et lorsque deux sujets (s_1 et s_2) traversent en même temps, leur temps de parcours est le maximum des deux, soit $\max(t(s_1), t(s_2))$. Pour cette énigme, on a $t(A) = 1$, $t(B) = 2$, $t(C) = 5$ et $t(D) = 10$ (en minutes). Il est possible que tout le monde traverse endéans 17 minutes, mais comment ?

Cette partie du projet s'intéresse à une généralisation de cette énigme et à sa résolution avec un solveur SAT. On se donne en entrée un nombre de sujets n, un temps pour chaque sujet via une fonction $t : \{1, \ldots, n\} \to \mathbb{N}$, un entier $T \in \mathbb{N}$ et une capacité $C \in \mathbb{N}$. La question est de savoir s'il est possible que tous les sujets traversent endéans T unités de temps, avec un pont de capacité C.

Formalisation Comme précédemment, on cherchera une séquence correcte de configurations $s : c_0 \to_{X_1} c_1 \cdots c_{n_1} \to_{X_{n-1}} c_n = c_F$ allant de la configuration initiale $c_0 = (0, \{1, \ldots, n\}, \emptyset)$ à la configuration finale $c_F = (1, \emptyset, \{1, \ldots, n\})$. On notera $\text{cap}(s) = \max_{1 \leq i \leq n} |X_i|$ le nombre de sujets maximum qui traversent en même temps, et $\text{tps}(s) = \sum_{i=1}^{n} \max_{s \in X_i} t(s)$ le temps pour faire traverser tout le monde selon cette solution.

Question 7 Etant donnés n, t, T, C, construire une formule en FNC $\phi_{n,t,T,C}$ qui est satisfaisable si et seulement si il existe une séquence de configurations s telle que $\text{tps}(s) \leq T$ et $\text{cap}(s) \leq C$.

Question 8 Implémenter et tester la solution.

Question 9 Étant donnés n et t, une paire (T, C) est optimale s'il n'existe pas de paire (T', C') avec $T' \leq T$, $C' \leq C$ et, $T' < T$ ou $C' < C$ (on doit être strictement inférieur sur au moins une des coordonnées), et une séquence correcte s telle que $\text{tps}(s) = T'$ et $\text{cap}(s) = C'$. Implémenter un algorithme qui énumère toutes les paires optimales. Expliquer dans le rapport votre implémentation.

5 Une autre généralisation possible (non notée)

On voudrait maintenant généraliser les deux problèmes précédents, en ayant des temps et des conflits. Une entrée de ce nouveau problème général sera donc un graphe G à n sommets, la relation d'arête représentant les conflits, et une fonction de temps $t : \{1, \ldots, n\} \to \mathbb{N}$.

Question 10 Même question que Q7 mais pour ce nouveau problème.

Question 11 Même question que Q8 mais pour ce nouveau problème.

Question 12 Même question que Q9 mais pour ce nouveau problème.

Question 13 Créer une énigme intéressante [1] avec des conflits et des temps.

1. C'est une notion bien subjective, mais on pourrait dire d'un énigme qu'elle est intéressante si après avoir réfléchi un peu à sa résolution, on se dit que c'est impossible et qu'il y a un problème dans l'énoncé.

Références

[1] Péter Csorba, Cor A. J. Hurkens and Gerhard J. Woeginger (2012). *The Alcuin Number of a Graph and Its Connections to the Vertex Cover Number*, SIAM.