Informatique fondamentale – Rapport projet
Calcul du nombre d'Alcuin d'un graphe

Célestin Pemmers, 000562451 Ashe Vazquez Nunez, 000577005

9 décembre 2024

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 $\text{Alcuin}(G) \leq k$.

Par théorème 1, on sait qu'il existe une séquence correcte s de longueur au plus $2n+1$ et telle que $\text{Alcuin}(G) = \text{Alcuin}(s)$. Ainsi, $\text{Alcuin}(G) \leq k \iff \text{Alcuin}(s) \leq k$.

Soit $p \leq k$. Remarquons que s'il existe une séquence correcte s dont $\text{Alcuin}(s) = p$, alors il existe nécessairement une séquence correccte s' telle que $\text{Alcuin}(s') = k$. Par la contraposée, s'il n'existe aucune séquence correcte de nombre d'Alcuin k, alors il n'existe aucune séquence correcte de nombre d'Alcuin p, pour tout $p \leq k$. Il suit donc que pour déterminer si $\text{Alcuin}(G) \leq k$, il suffit de vérfier s'il existe une séquence correcte de nombre d'Alcuin exactement égale à k. Notre formule est satisfaisable si et seulement si $\text{Alcuin}(G) \leq k$, si et seulement si il existe une sequence correcte s telle que $\text{Alcuin}(s) = k$.

Considérons l'ensemble de propositions $P = \{x_{i,a} \mid i \in \{0, \ldots, 2n+1\}, a \in S\}$, et donnons à la proposition $x_{i,a}$ la sémantique suivante : $x_{i,a}$ est évaluée à vrai si et seulement si, lors de la i-ème configuration, le sommet a se trouve sur la rive 1. Nous conservons ici la dénomination des rives utilisées dans l'énoncé du projet, où la rive 0 est la rive de départ et la rive 1 celle d'arrivée. Également, nous considérons comme i-ème configuration l'état après le i-ème trajet, la configuration 0 étant l'état initial avant tout déplacement.

En guise d'exemple pour expliciter notre notation, nous invitons le lecteur à se référer à la Figure 1 pour voir comment elle s'applique dans le cadre du problème du loup, de la chèvre et du chou.

Nous cherchons à présent à traduire le fait que s soit une solution de longueur au plus $2n+1$ et de nombre d'Alcuin k en une formule logique $\phi_{G,k}$. Précisons à présent les conditions qui caractérisent s :

1. Initialement, tous les sommets sont sur la rive 0.

$$\bigwedge_{a \in S} \neg x_{0,a}$$

2. À la fin ($2n+1$-ème configuration), tous les sommets sont sur la rive 1.

$$\bigwedge_{a \in S} x_{2n+1,a}$$

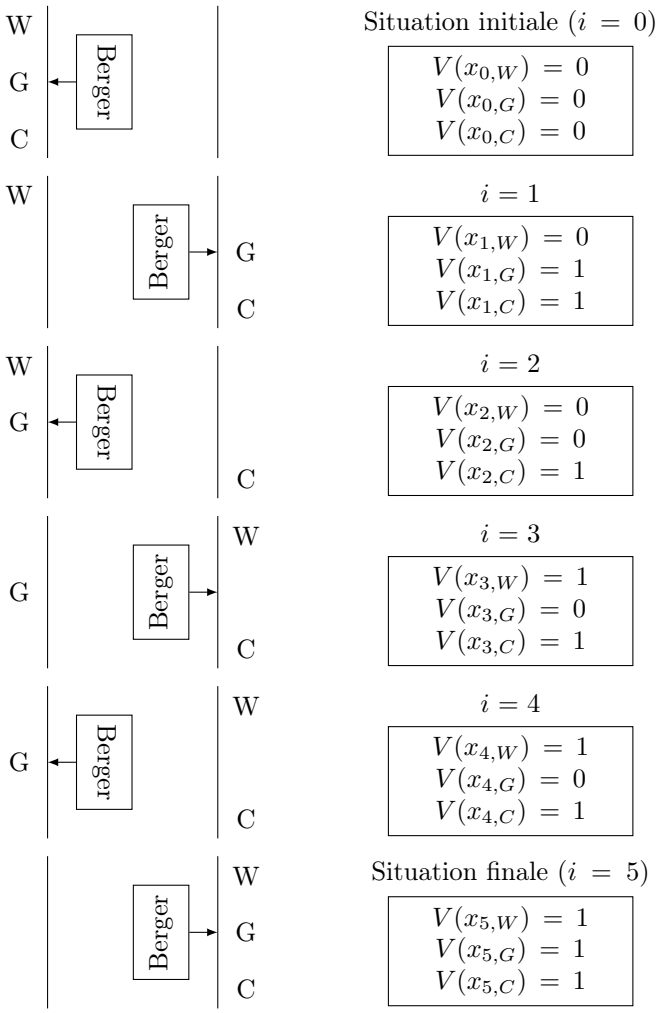

FIGURE 1 – Application de la notation avec l'exemple du loup, de la chèvre et du chou, où V est la fonction de valuation associée à la solution

3. À chaque étape, il n'y a pas deux sommets reliés (en conflits) sur la rive opposée au berger.
   Cette condition est équivalente à dire que, pour toute configuration i et pour deux sommets reliés a et b, au moins l'un des deux est avec le berger. Sachant qu'aux configurations i impaires, le berger se trouve sur la rive 1, nous avons que a est sur la rive 1, ou b est sur la rive 1, donc

$$\bigwedge_{\substack{i=0 \\ i \bmod 2=1}}^{2n+1} \bigwedge_{(a,b)\in E} (x_{i,a} \vee x_{i,b})$$

   Également, comme pour i paire le berger se trouve sur la rive 0, nous avons que a est sur la rive 0, ou b est sur la rive 0, donc

$$\bigwedge_{\substack{i=0 \\ i \bmod 2=0}}^{2n+1} \bigwedge_{(a,b)\in E} (\neg x_{i,a} \vee \neg x_{i,b})$$

4. Les sommets ne passent d'une rive à l'autre qu'avec le berger.
   Lors d'une configuration paire, le berger se trouve sur la rive 0. Ainsi, les sommets peuvent rester sur leur rive respective, ou passer de la rive 0 à la rive 1. En revanche, les sommets sur la rive 1 ne peuvent pas passer sur la rive 0, car le berger ne peut pas les transporter, étant du mauvais côté.

$$\bigwedge_{\substack{i=0 \\ i \bmod 2=0}}^{2n} \bigwedge_{a\in S} \underbrace{(\neg x_{i,a} \vee x_{i+1,a})}_{\neg(x_{i,a}\wedge\neg x_{i+1,a})}$$

   Inversément, lors d'une configuration impaire, le berger se trouve sur la rive 1, et ne peut donc pas transporter les sommets de la rive 0 à la rive 1.

$$\bigwedge_{\substack{i=0 \\ i \bmod 2=1}}^{2n} \bigwedge_{a\in S} \underbrace{(x_{i,a} \vee \neg x_{i+1,a})}_{\neg(\neg x_{i,a}\wedge x_{i+1,a})}$$

5. À chaque configuration, il y a au plus k sommets sur le bateau (car $\text{Alcuin}(s) = k$).
   Comme au plus k sommets sont transportés, cela signifie que, étant donné un ensemble $\{a_1, \ldots, a_{k+1}\}$ de $k+1$ sommets, au moins un de ces sommets n'est pas transporté, i.e. il existe $j \in \{1, \ldots, k+1\}$ tel que a_j n'est pas simultanément avec le berger à la configuration i et sur l'autre rive à la configuration $i+1$. Plus formellement, cela s'exprime comme

$$\begin{cases} \neg(\neg x_{i,a_j} \wedge x_{i+1,a_j}) & \text{si } i \bmod 2 = 0 \\ \neg(x_{i,a_j} \wedge \neg x_{i+1,a_j}) & \text{si } i \bmod 2 = 1 \end{cases}$$

   La formule sous FNC finale est donc

$$\bigwedge_{\substack{i=0 \\ i \bmod 2=0}}^{2n} \bigwedge_{\{a_1,\ldots,a_{k+1}\}\subseteq S} \bigvee_{j=1}^{k+1} (x_{i,a_j} \vee \neg x_{i+1,a_j})$$

   quand i est pair (le berger est sur la rive 0), et

$$\bigwedge_{\substack{i=0 \\ i \bmod 2=1}}^{2n} \bigwedge_{\{a_1,\ldots,a_{k+1}\}\subseteq S} \bigvee_{j=1}^{k+1} (\neg x_{i,a_j} \vee x_{i+1,a_j})$$

   quand i est impair (le berger est sur la rive 1).

Question 2. Implémentation de notre construction de la formule de la question 1.

Les cinq contraintes logiques décrites dans la question 1 sont implémentées dans le même ordre dans la fonction `generate_Q2_cnf`. Pour l'implémentation en Python, nous avons combiné les formules qui distinguent des cas sur base de la parité de i. Notons qu'à l'état i, le berger se trouve sur la rive $i \bmod 2$. Alors, en définissant $-x := \neg x$, dire « le sommet a est sur la même rive que le berger à l'état i » est équivalent à

$$(-1)^{(i \bmod 2)+1} x_{i,a}$$

On peut également décrire qu'un sommet a est sur la rive opposée au berger par

$$(-1)^{i \bmod 2} x_{i,a}$$

Avec cette idée, on peut par exemple regrouper les deux formules pour la contrainte 3. en une seule :

$$\bigwedge_{i=0}^{2n+1} \bigwedge_{(a,b)\in E} \left((-1)^{(i \bmod 2)+1} x_{i,a} \vee (-1)^{(i \bmod 2)+1} x_{i,b} \right)$$

Les formules pour les contraintes 3, 4 et 5 sont ainsi simplifiées dans notre code.

Question 3. Algorithme pour trouver le nombre d'Alcuin du graphe $G = (S, E)$.

Comme il existe toujours une solution pour $k = n \overset{\text{not}}{=} |S|$, nous savons que 1 est une borne inférieure pour $\text{Alcuin}(G)$ et n est une borne supérieure. De plus, l'implémentation de la question précédente nous donne un algorithme pour déterminer, à $k \leq n$ fixé, si $\text{Alcuin}(G) \leq k$. Si $\text{Alcuin}(G) \leq k$, alors k devient notre borne supérieure, et si $\text{Alcuin}(G) > k$, $k+1$ devient notre borne inférieure. Ainsi, nous pouvons implémenter une recherche binaire qui trouve $\text{Alcuin}(G)$ en temps $\Theta(\log(n))$ (ici, nous ne considérons pas la complexité de l'algorithme implémenté dans la question 2). L'algorithme de recherche binaire est optimal, car il revient à chercher dans une liste triée $[0, 0, \ldots, 0, 1, \ldots, 1]$ le premier élément non-nul, qui est d'indice $\text{Alcuin}(G)$.

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 $\text{Alcuin}_c(G) \leq k$.

Pour modéliser ce problème, gardons l'ensemble $P = \{x_{i,a} \mid i \in \{0, \ldots, 2n+1\}, a \in S\}$ et sa sémantique associée. De plus, ajoutons un ensemble de propositions $P_1 = \{p_{i,a,j} \mid i \in \{1, \ldots, 2n+1\}, a \in S, j \in \{0, \ldots, c\}\}$ muni de la sémantique suivante : $p_{i,a,j}$ est vrai si et seulement si lors du i-ème trajet, le sommet a a été transporté dans le compartiment j, et $p_{i,a,0}$ est vrai si et seulement si le sommet a n'a pas été transporté lors du i-ème trajet. L'objectif de cet ensemble est de founir une manière directe de dire en logique que deux sommets en conflit ne peuvent pas être placés dans le même compartiment.

Établissons à présent des règles de compatibilité entre l'ensemble P et l'ensemble P_1.

1. Tout sommet voyage dans au plus un compartiment

$$\bigwedge_{i=1}^{2n+1} \bigwedge_{a\in S} \bigwedge_{j=0}^{c} \bigwedge_{\substack{j'=0 \\ j'\neq j}}^{c} \neg p_{i,a,j} \vee \neg p_{i,a,j'}$$

2. Tout sommet voyage dans au moins un compartiment

$$\bigwedge_{i=1}^{2n+1} \bigwedge_{a\in S} \bigvee_{j=0}^{c} p_{i,a,j}$$

3. Un sommet a n'était dans aucun des compartiments non nuls si et seulement si il n'a pas été bougé lors du i-ème trajet.
Commençons avec l'implication directe. Supposons qu'un sommet a ne soit dans aucun des compartiments non nuls. Comme a est dans un unique compartiment, il est nécessairement dans le compartiment 0. Ainsi, nous voulons que si a est dans ce compartiment, alors il ne voyage pas. Si la configuration i est paire, cela nous donne

$$p_{i+1,a,0} \implies \neg(\neg x_{i,a} \wedge x_{i+1,a}) \equiv \neg p_{i+1,a,0} \vee x_{i,a} \vee \neg x_{i+1,a}$$

Si la configuration i est impaire, nous obtenons

$$p_{i+1,a,0} \implies \neg(x_{i,a} \wedge \neg x_{i+1,a}) \equiv \neg p_{i+1,a,0} \vee \neg x_{i,a} \vee x_{i+1,a}$$

Les formules ainsi obtenues sont

$$\bigwedge_{\substack{i=0 \\ i \bmod 2=0}}^{2n} \bigwedge_{a \in S} \neg p_{i+1,a,0} \vee x_{i,a} \vee \neg x_{i+1,a}$$

pour i paire et

$$\bigwedge_{\substack{i=0 \\ i \bmod 2=1}}^{2n} \bigwedge_{a \in S} \neg p_{i+1,a,0} \vee \neg x_{i,a} \vee x_{i+1,a}$$

pour i impaire.

Passons à présent à l'implication inverse. Supposons ainsi qu'un sommet a n'a pas bougé lors du $i+1$ trajet. Si la configuration i est paire, cela veut dire que le sommet n'était pas sur la rive 0 avec le berger lors de la i-ème configuration, puis sur la rive 1 lors de la $i+1$-ème configuration (soit, $\neg(\neg x_{i,a} \wedge x_{i+1,a})$). Nous obtenons alors que

$$\begin{aligned} \neg(\neg x_{i,a} \wedge x_{i+1,a}) \implies p_{i+1,a,0} &\equiv \neg(\neg(\neg x_{i,a} \wedge x_{i+1,a})) \vee p_{i+1,a,0} \\ &\equiv (\neg x_{i,a} \wedge x_{i+1,a}) \vee p_{i+1,a,0} \\ &\equiv (\neg x_{i,a} \vee p_{i+1,a,0}) \wedge (x_{i+1,a} \vee p_{i+1,a,0}) \end{aligned}$$

Si i est impaire, cela veut dire que le sommet n'est pas sur la rive 1 avec le berger lors de la i-ème configuration, puis sur la rive 0 lors de la $i+1$-ème configuration (soit, $\neg(x_{i,a} \wedge \neg x_{i+1,a})$). Nous obtenons donc que

$$\begin{aligned} \neg(x_{i,a} \wedge \neg x_{i+1,a}) \implies p_{i+1,a,0} &\equiv \neg(\neg(x_{i,a} \wedge \neg x_{i+1,a})) \vee p_{i+1,a,0} \\ &\equiv (x_{i,a} \wedge \neg x_{i+1,a}) \vee p_{i+1,a,0} \\ &\equiv (x_{i,a} \vee p_{i+1,a,0}) \wedge (\neg x_{i+1,a} \vee p_{i+1,a,0}) \end{aligned}$$

Les formules ainsi obtenues sont

$$\bigwedge_{\substack{i=0 \\ i \bmod 2=0}}^{2n} \bigwedge_{a \in S} (\neg x_{i,a} \vee p_{i+1,a,0}) \wedge (x_{i+1,a} \vee p_{i+1,a,0})$$

pour i paire et

$$\bigwedge_{\substack{i=0 \\ i \bmod 2=1}}^{2n} \bigwedge_{a \in S} (x_{i,a} \vee p_{i+1,a,0}) \wedge (\neg x_{i+1,a} \vee p_{i+1,a,0})$$

4. Deux sommets en conflit ne sont jamais dans un même compartiment non nul

$$\bigwedge_{i=1}^{2n+1} \bigwedge_{(a,b)\in E} \bigwedge_{j=1}^{c} \neg p_{i,a,j} \vee \neg p_{i,b,j}$$

Remarquons qu'il est ici inutile d'inclure le compartiment 0. En effet, soient a et b, deux sommets en conflits. Si, lors du $i+1$-ème trajet, a et b sont dans le compartiment 0, alors aucun d'eux n'est dans un compartiment non nul. Donc, par le point précédent, aucun n'a été transporté. Ainsi, a et b sont chacun situés sur la même rive lors des configurations i et $i+1$. Si ces deux sommets sont sur des rives opposées, il n'y a pas de conflit. S'ils sont tout deux sur la même rive, alors ils sont avec le berger lors d'une configuration, et sur la rive opposée au berger lors de l'autre. Or, par point 3 de la question 1, ce n'est pas possible. Autrement dit, deux sommets en conflits peuvent être dans le compartiment 0, à condition qu'ils soient, avant et après le trajet, sur des rives différentes.

Question 5. Implémentation en Python de notre construction de la formule de la question 4.

Les contraintes sont encodées dans la fonction `generate_Q5_cnf`, dans le même ordre qu'elles sont décrites dans ce rapport. Nous avons utilisé la même technique que dans la question 2 pour simplifier les expressions dans notre code.

Question 6. Algorithme pour trouver le nombre d'Alcuin à c compartiments pour un graphe $G = (S, E)$.

S'il existe une séquence c-valide, alors $\text{Alcuin}_c(G) \in \{1, \ldots, n\}$, où $n = |S|$. De plus, comme pour le cas sans compartiments, l'existence d'une séquence correcte pour un certain $k \leq n$ implique l'existence d'une séquence correcte pour tout $j \in \{k, \ldots, n\}$, et la non-existence d'une séquence c-valide pour $k \leq n$ implique la non-existence pour tout $j \in \{1, \ldots, k\}$. Par contre, dans le cas précédent, nous étions garantis d'avoir une séquence valide pour $k = n$. Dans le cas des compartiments, il n'existe pas toujours une solution (e.g., le graphe complet à trois sommets K_3, avec $c = 1$). Nous définissons alors $\text{Alcuin}_c(G) = +\infty$, ou `INFINITY` en Python. Notre algorithme vérifie d'abord l'existence d'une séquence c-valide de nombre d'alcuin $k = n$, et retourne `INFINITY` si une telle séquence n'existe pas. Sinon, notre algorithme effectue une recherche binaire comme pour la question 3. Ainsi, nous trouvons $\text{Alcuin}_c(G)$ en temps $\Theta(\log(n))$ dans le pire dans cas (encore une fois, sans considérer la complexité de l'algorithme implémenté à la question 5).