Université Libre de Bruxelles
Faculté des Sciences
Département de Mathématiques

Approximabilité de problèmes de couverture

Christophe DUMEUNIER

Mémoire présenté sous la direction de **Mr. Jean CARDINAL**
en vue de l'obtention du grade de Licencié en Sciences Mathématiques
Année académique 2006-2007

Je tiens à exprimer mes plus vifs remerciements à mon directeur de mémoire Monsieur Jean Cardinal pour la confiance et l'aide qu'il m'a apportées durant la réalisation de ce travail qui n'aurait jamais pu voir le jour sans ses précieux conseils.
J'aimerais aussi remercier mes parents pour les nombreuses relectures orthographiques.

Introduction

Un écureuil a l'habitude de se promener dans une petite parcelle de forêt, sautant d'arbre en arbre lorsque les branches basses le permettent. Malheureusement, le joli petit animal est fort vulnérable, et les prédateurs sont sans pitié. Chaque fois qu'il est dangereusement menacé, il n'a que très peu de temps pour trouver un abri et s'y dissimuler. Il décide donc de construire des cachettes un peu partout dans la forêt, pour toujours en avoir une à proximité et pouvoir s'y rendre en un temps raisonnable, quel que soit l'endroit où il se trouve. Bien entendu notre ami n'est pas un grand travailleur, et il préfère rester calmement à réfléchir sous une confortable feuille de chêne avec une noisette chaude à grignoter plutôt que de travailler à l'aménagement de cachettes. C'est ainsi qu'il décide de trouver une stratégie à la fois rapide et efficace pour construire un minimum de cachettes réparties de façon intelligente dans les arbres.

Stephen Arthur Cook, né en 1939 à New York, est un éminant chercheur en informatique. En 1971, il formalise la notion de $\mathcal{NP}$-complétude dans un célèbre article *The Complexity of Theorem Proving Procedures* contenant une preuve de $\mathcal{NP}$-complétude du problème de satisfaisabilité booléen (appelé aussi SAT). C'est à ce moment que se soulève la grande question d'équivalence entre les classes $\mathcal{P}$ et $\mathcal{NP}$, c'est-à-dire savoir si tout problème vérifiable en temps polynomial est aussi résoluble en temps polynomial. Cook recevra en 1982 un Turing Award pour cette découverte. En 1972, Richard Karp, 37 ans, professeur à l'Université de Californie, reprend cette idée et fournit, dans son article *Reductibility Among Combinatorial Problems*, la preuve de la $\mathcal{NP}$-complétude de 21 problèmes divers de combinatoire et théorie des graphes, parmi eux le problème de Couverture d'ensemble.

Le problème de Couverture d'ensemble (*Set Covering Problem* dans la littérature originale) est une question classique en théorie de la complexité et en optimisation combinatoire. Une collection d'ensembles est considérée, certains ayant des éléments en commun ; on cherche alors à sélectionner un minimum de ces ensembles de façon à ce que leur union contienne tous les éléments donnés. À partir de là, de nombreux autres problèmes apparaissent, ne visant plus à minimiser le nombre d'ensembles nécessaires, mais utilisant d'autres critères poursuivant la même idée. Nous présenterons un certain nombre des résultats de Cook et de Karp, que nous puisons dans l'ouvrage de Jon Kleinberg et Éva Tardos *Algorithm Design* [5], menant à l'étude des problèmes plus précis que nous pourrons alors traiter.

Nous nous intéressons à une résolution intelligente de ces problèmes. Il est en effet dangereux de chercher à résoudre de façon optimale un problème n'étant pas dans $\mathcal{P}$, le temps et les ressources nécessaires pouvant réellement exploser. Le choix étant fait de ne travailler qu'avec des algorithmes polynomiaux, nous portons notre attention sur l'*algorithme glouton* pour résoudre ce type de problème. Il s'avèrera, parmi d'autres résultats, que l'algorithme glouton est le meilleur algorithme polynomial pour chacun des problèmes que nous aborderons, à moins que $\mathcal{P}$=$\mathcal{NP}$. Nous nous attachons à représenter la capacité de l'algorithme glouton à s'approcher de la solution optimale, et en particulier nous allons regarder les pires situations qu'il pourrait rencontrer.

Se basant sur les résultats présentés par Jon Kleinberg et Éva Tardos dans *Algorithm Design* [5], nous présentons l'approximation de l'algorithme glouton pour le Problème de Couverture d'ensemble, et nous donnons la preuve que l'algorithme glouton approxime ce problème avec un rapport maximum de l'ordre du logarithme de la taille du plus grand des ensembles donnés par le problème.

À partir de l'article de Uriel Feige, László Lovász et Prasad Tetali *Approximating Min-Sum Set Cover* [6], nous présentons, avec des notations différentes, le problème de Couverture d'ensemble de somme minimum, visant cette fois à minimiser le temps moyen de couverture d'un élément quelconque de ces ensembles donnés, par sélection successive des ensembles. Nous transcrivons, dans de nouvelles notations, la démonstration de l'approximation de l'algorithme glouton fournissant une solution au pire cas 4 fois plus grande que la solution optimale. Nous présentons également le fait que l'algorithme glouton est le meilleur algorithme polynomial pour résoudre ce problème, à moins que $\mathcal{P}$=$\mathcal{NP}$.

Partant de l'article de Jean Cardinal, Samuel Fiorini et Gwenaël Joret *Tight Results on Minimum Entropy Set Cover* [9], nous présentons le problème de couverture d'ensemble d'entropie minimum, visant à minimiser l'entropie de la distribution que l'on obtient en considérant les probabilités de recouvrement de chacun des éléments des ensembles. En s'appliquant à conserver les notations des problèmes précédents, nous présentons des résultats équivalents à ceux fournis pour le problème de Couverture d'ensemble de somme minimum, à savoir la borne maximum d'approximation de l'algorithme glouton fixée à la constante additive $\log_2(e)$. Nous présentons la démonstration de l'optimalité de glouton dans la recherche du meilleur algorithme polynomial en supposant que $\mathcal{P}$ n'est pas égal à $\mathcal{NP}$, pour laquelle nous ajoutons les détails et les formalismes de nombreux calculs.

Poursuivant dans cette lignée, nous présentons un nouveau problème, et apportons de nouveaux résultats similaires à ceux des problèmes ci-dessus. Il s'agit du problème de Couverture d'ensemble de somme des carrés maximum, qui tente cette fois de maximiser la somme quadratique de la distribution, utilisée dans le problème précédent, que l'on obtient en considérant les probabilités de recouvrement de chacun des éléments des ensembles. Nous fournissons une preuve de l'approximabilité de l'algorithme glouton avec un rapport égal à 2, et nous démontrons que l'algorithme glouton est encore le meilleur algorithme polynomial pour ce problème, à moins que $\mathcal{P}$=$\mathcal{NP}$.

Le but de l'étude de ces problèmes particuliers n'est pas de connaître et d'accumuler de nombreux résultats indépendants, mais bien d'entrevoir une généralisation possible. Nous donnerons ainsi une forme généralisée de ces problèmes (mis à part le problème de Couverture d'ensemble de somme minimum dont les caractéristiques le font appartenir à une classe différente de problèmes), à savoir un problème de couverture d'ensemble visant à maximiser la moyenne généralisée $\mathcal{M}_r$ de la distribution déjà présentée ci-dessus ; chacun des r réels fournissant un problème différent.

Chapitre 1

Notions élémentaires

Nous allons introduire ici quelques notions élémentaires, afin de familiariser le lecteur avec les notations et les outils qui seront utilisés tout au long de ce travail. Après avoir fixé ces bases, nous considérerons cela comme acquis et n'y ferons plus référence.

1.1 Graphes et Hypergraphes

Les problèmes que nous traiterons nécessitent la manipulation d'ensembles et de collections de sous-ensembles dotés de certaines propriétés. Nous choisissons de prendre la notion habituelle d'hypergraphe comme support et de présenter quelques résultats qui nous seront utiles pour leur manipulation. La notion d'hypergraphe dans ces problèmes est une structure simple permettant de faciliter les notations et la visualisation des problèmes, cependant elle n'intervient pas fondamentalement dans les problèmes que nous traiterons. Toutes ces notions sont présentées plus amplement dans le cours de Francis Buekenhout et Jean Doyen *Ensembles structurés et groupes de symétrie* [4] et dans le cours de Geir Dahl *An introduction to convexity, polyhedral theory and combinatorial optimisation* [3].

On appelle *graphe* et on note $G(V, E)$ un ensemble V d'éléments appelés *sommets* dont la structure est déterminée par la donnée d'un ensemble E de paires de sommets nommées *arêtes*.
Deux sommets u et v d'un graphe $G(V, E)$ sont dits *adjacents* si l'arête $\{u, v\}$ appartient à E. La *matrice d'adjacence* est une $|V|\times|V|$-matrice A définie par $A_{ij} = 1$ si $\{v_i, v_j\} \in E$ et $A_{ij} = 0$ sinon. Cette matrice est symétrique.
Soit $v \in V$, on définit $inc(v)$ l'ensemble des arêtes incidentes à v, c'est-à-dire $inc(v) = \{e \in E | v \in e\}$. On appelle *degré* de v et on note d_v le nombre d'arêtes incidentes à v, c'est-à-dire $d_v = |inc(v)|$.

Proposition 1.1 *Étant donné un graphe $G(V, E)$,*

1. $\sum_{v\in V} d_v = 2|E|$
2. *Le nombre de sommets de degré impair est pair*

Démonstration 1.1

1. En comptant toutes les arêtes aboutissant à chacun des sommets, chaque arête est comptée exactement deux fois.
2. Découle immédiatement du point précédent. En effet, dans le cas contraire, la somme devrait être impaire.

□

Étant donné un graphe $G(V, E)$, on appelle *promenade* dans G une suite $\pi =< v_0, v_1, \ldots, v_n >$ telle que $\{v_i, v_{i+1}\} \in E$ pour tout i. La valeur n est appelée *longueur* de la promenade.
On appelle *chemin* une promenade $\gamma =< v_0, v_1, \ldots, v_n >$ telle que $\{v_i, v_{i+1}\} \neq \{v_j, v_{j+1}\}$ pour tout $i \neq j$, c'est-à-dire une promenade ne passant pas deux fois par la même arête.
On note $\mathcal{C}(u, v)$ l'ensemble des chemins allant de u à v, et on note $\mathcal{C}(G)$ l'ensemble $\bigcup_{u,v \in V} \mathcal{C}(u, v)$ des chemins dans G.
Un graphe $G(V, E)$ est dit *connexe* si $\mathcal{C}(u, v) \neq \emptyset$ pour tout couple $(u, v) \in V^2$, c'est-à-dire s'il existe au moins un chemin reliant chaque paire de sommets du graphe.
On appelle *circuit* un chemin $c =< v_0, v_1, \ldots, v_n >$ tel que $v_0 = v_n$.
On dit qu'un graphe $G'(V', E')$ est un *sous-graphe* du graphe $G(V, E)$ si $V' \subseteq V$ et $E' \subseteq E$.
On appelle *cycle* un graphe constitué d'un seul circuit et dont tous les sommets sont de degré deux. Un graphe $G(V, E)$ est appelé *forêt* s'il n'existe pas de sous-graphe G' de G qui soit un cycle. On appelle *arbre* une forêt connexe.
Un graphe $G(V, E)$ est dit *complet* si toute paire de sommets de V est une arête de E. On définit le *graphe complémentaire* $G^c(V, E^c)$ du graphe $G(V, E)$ par la donnée de $E^c = \{e \notin E\}$. On remarque que $G^{cc} = G$.
On appelle *graphe biparti* un graphe $G(V_1 \cup V_2, E)$ tel que V_1 et V_2 soient non-vides, d'intersection vide et tel que toute arête de E prenne l'une de ses extrémités dans V_1 et l'autre extrémité dans V_2.

Proposition 1.2 *Un graphe est biparti si et seulement s'il ne possède pas de cycle impair.*

Démonstration 1.2 Soit un graphe biparti $G(V_1 \cup V_2, E)$. Pour suivre un chemin dans G, on est obligé de sauter de V_1 à V_2, puis de V_2 à V_1. Pour suivre un cycle, on est donc obligé de faire un nombre pair de pas pour retrouver le premier point du cycle.
Soit un graphe $G(V, E)$ ne contenant pas de cycle impair. On choisit un sommet de départ, que l'on colorie en rouge, on saute ensuite à un sommet adjacent que l'on colorie en vert, puis un autre en rouge, et ainsi de suite jusqu'à avoir colorié tous les sommets du graphe. Le graphe ne contenant pas de cycle impair, aucun des sommets n'a pu être colorié en rouge et en vert. On peut donc partitionner notre ensemble en $Vert \cup Rouge$ et ainsi le graphe est biparti. □
On appelle *graphe dirigé* (ou *digraphe*) et on note $D(V, A)$ un ensemble V de *sommets* muni d'un ensemble A de couples de sommets de V nommés *arcs*, à la différence du *graphe* pour lequel les arêtes sont des paires de sommets.

On appelle *graphe pondéré* (ou *multigraphe*) et on note $P(V, W)$ un ensemble V de *sommets* muni d'un ensemble W d'arêtes auxquelles on associe un nombre naturel appelé *poids*.
On étend naturellement aux digraphes et graphes pondérés les notions d'*adjacence*, *promenade*, *chemin*, *circuit*, etc. Dans le cas des graphes pondérés, on appelle *degré* d'un sommet v, et on note d_v, la somme des poids des arêtes incidentes à v.

On appelle *hypergraphe* et on note $H(V, E)$ un ensemble V de *sommets* muni d'une collection E de sous-ensembles de V de cardinal au moins deux, et qu'on appellera *hyperarêtes*.
Étant donné un hypergraphe $H(V, E)$, on dit d'un ensemble $K \subseteq V$ de sommets qu'ils sont *adjacents* s'il existe une hyperarête $e \in E$ contenant tous les sommets de K. On dit d'un ensemble L d'hyperarêtes qu'elles sont *concourantes* s'il existe un sommet $v \in V$ contenu dans toute hyperarête de L.
On appelle *degré* d'un sommet v le nombre d'hyperarêtes de E qui contiennent v, et on le note d_v. On appelle *degré* d'une hyperarête e le nombre de sommets qu'elle contient, et on note r_e.
On appelle *hypergraphe d-régulier* un hypergraphe $H(V, E)$ tel que chaque sommet v de V est de degré d et on appelle *hypergraphe r-uniforme* un hypergraphe $H(V, E)$ tel que chaque hyperarête e de E est de degré r.
Soit un hypergraphe $H(V, E)$ avec $|V| = n$ et $|E| = m$. On appelle *hypergraphe dual* de

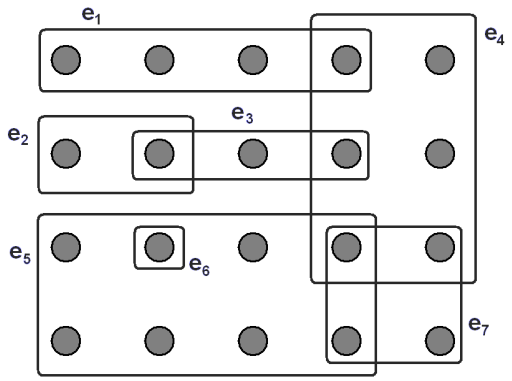

FIG. 1.1 – Exemple d'hypergraphe constitué de 20 sommets et 7 hyperarêtes

H l'hypergraphe $H^*(E, V^*)$ où $V^* = \{v_1^*, v_2^*, \ldots, v_n^*\}$ avec $v_i^* = \{e_j | v_i \in e_j \in E\}$. Plus intuitivement la dualité est une transformation du graphe qui à chaque sommet associe une hyperarête, et à chaque hyperarête associe un sommet ; de plus des hyperarêtes concourantes deviennent des sommets adjacents en conservant leur degré, et vice-versa.

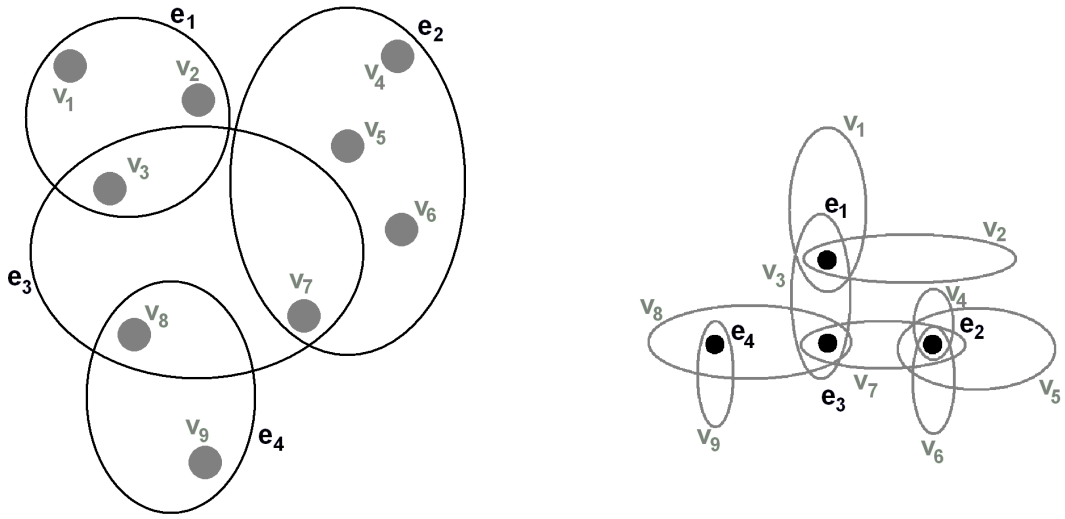

Fig. 1.2 – Exemple d'hypergraphe et son dual

1.2 Modèles de calcul et complexité

Pour cette section et la suivante, le lecteur retrouvera ces informations dans le cours de Thierry Massart *Programmation* [2] et dans l'ouvrage de Jon Kleinberg et Eva Tardos *Algorithm Design* [5].

L'*algorithmique* ne peut exister que dans un cadre de *traitement de l'information*, une *information* étant un ensemble d'éléments qui ont une signification dans le contexte observé. Les *données* d'un problème sont l'ensemble des informations dont on dispose pour résoudre ce problème, et qui nous mèneront, au travers de l'algorithme, vers le *résultat*.

Un *algorithme* est une méthode permettant d'obtenir le résultat correspondant aux données introduites, données qui seront toutes d'un type semblable pour un même problème. On peut voir un algorithme comme une application $f : D \to R : d \mapsto f(d)$ pour lequel l'image de toute donnée (parmi l'ensemble D des données possibles) est un résultat (parmi l'ensemble R des résultats possibles). On considérera qu'un algorithme est la méthode calculant $f(d)$ en un temps fini, lorsque d est une donnée valide.

Cela nous amène inévitablement à parler de machines théoriques, permettant de concevoir une définition formelle d'algorithme et d'exécution de celui-ci. Regardons en particulier la *Machine de Turing*. Il s'agit d'un modèle abstrait de fonctionnement des appareils mécaniques de calcul (comme les ordinateurs) conçu par Alan Turing. La thèse de Church-Turing affirme en effet que tout traitement d'information réalisable mécaniquement peut être accompli par une machine de Turing. Elle est généralement considérée comme vraie, mais n'est pas un énoncé mathématique : chercher

à la démontrer n'a pas de sens. En l'occurrence, tout programme informatique peut être traduit en machine de Turing. Concrètement, on représente la machine de Turing de la manière suivante :

1. Un ruban de longueur infinie (on entend par là un ruban aussi grand qu'il sera nécessaire) est divisé en cases consécutives. Chaque case contient un symbole appartenant à un alphabet fini, préalablement établi, dont l'un des éléments est un symbole clef appelé *blanc* et contenant un ou plusieurs autres symboles.
2. Une tête de lecture/écriture peut lire et écrire un symbole de l'alphabet dans la case sur laquelle elle se trouve, et peut se déplacer vers la gauche ou vers la droite sur le ruban.
3. Un registre d'état mémorise l'état courant de la machine parmi un nombre fini d'états préétablis. L'un d'eux est un état particulier appelé *état initial*, qui, comme son nom l'indique, représente l'état de la machine au début du traitement.
4. Une table d'actions indique à la machine l'action suivante à exécuter (c'est-à-dire l'écriture d'un symbole dans la case courante, le déplacement d'une case vers la droite ou la gauche, la détermination du nouvel état de la machine) en fonction du symbole lu et de l'état courant.

De façon plus théorique, nous avons un alphabet A qui est un ensemble fini de symboles $\{a_0, a_1, a_2, \ldots, a_n\}$ et un ensemble fini $E = \{e_0, e_1, e_2, \ldots, e_m\}$ d'états. On décide que a_0 est l'élément particulier de A appelé blanc, et e_0 est l'élément particulier de E appelé état initial. Notre ruban initial peut être représenté par une suite $< R^i >_{i\in\mathbb{Z}}$ d'éléments de A indicée sur les entiers positifs et négatifs ; on note $\mathcal{R}$ l'ensemble des tels rubans. La table d'action est donnée par une fonction $f : A \times E \to A \times \{-1, 1\} \times E$ que l'on décompose en trois fonctions $f_1 : A \times E \to A$ fournissant le symbole à écrire, $f_2 : A \times E \to \{-1, 1\}$ indiquant le déplacement de la tête, et $f_3 : A \times E \to E$ fournissant le nouvel état de la machine. Le traitement est alors une suite $< R^i_t, k_t, E_t >_{t\in\mathbb{N}}$ dans $\mathcal{R} \times \mathbb{Z} \times E$ indicée par t et définie par la relation de récurrence suivante :

$$R^i_0 = R^i \qquad \forall i \in \mathbb{Z} \tag{1.1}$$

$$k_0 = 0 \tag{1.2}$$

$$E_0 = e_0 \tag{1.3}$$

$$R^i_{t+1} = \begin{cases} f_1(R^{k_t}_t, E_t) & \text{si } i = k_t \\ R^i_t & \text{sinon} \end{cases} \quad \forall i \in \mathbb{Z} \tag{1.4}$$

$$k_{t+1} = k_t + f_2(R^{k_t}_t, E_t) \tag{1.5}$$

$$E_{t+1} = f_3(R^{k_t}_t, E_t) \tag{1.6}$$

Les machines de Turing sont des représentations théoriques très simples d'un algorithme et de son exécution. D'autres machines plus puissantes peuvent être regardées, par exemple des langages simples comme l'assembleur ou le C++, ou encore des langages très évolués avec des actions complexes comme l'openGL ou les fonctions mathlab.

Dans notre représentation d'un algorithme, la notion fondamentale est celle d'*action*. Chaque action aura une durée finie et bornée par des constantes dépendant de la configuration de la machine ; et elle aura un effet précis. Un *algorithme* sera une succession finie de telles actions. Bien évidemment, il n'existe pas d'algorithme pour tout problème, il suffirait de citer les problèmes pour lesquels l'impossiblité de trouver une solution par une quelconque méthode est démontrée. Il est donc important de se poser des questions d'existence d'un algorithme lorsque l'on est face à un problème. En particulier, nous verrons qu'il est bien des cas où l'on ne trouve pas d'algorithme lorsque l'on soumet celui-ci à certaines conditions de complexité. En général, on pourra trouver différentes méthodes pour résoudre un même problème ; il faudra alors choisir celle qui sera la "meilleure", la plus optimale, mais selon quel critère ? Les critères prédominants sont la simplicité et l'efficacité ; cependant ces deux valeurs sont bien souvent contradictoires, et ce sera au concepteur de choisir à quelle optimalité se confier.

On parlera de *complexité d'algorithme* lorsque l'on quantifiera la quantité de ressources utilisées par cet algorithme. Cela se rapporte à différents critères tels que le temps d'exécution, le nombre d'appels à une méthode précisée, l'utilisation mémoire, etc. Nous nous bornerons ici à observer le temps d'exécution, ou plus exactement, le nombre d'actions élémentaires effectuées par l'algorithme.

La *complexité* peut s'exprimer de multiples façons selon le but. Nous pouvons aussi bien nous intéresser à la complexité minimale, maximale, moyenne, ou d'autres encore. Pour formaliser, nous noterons la complexité choisie $T(n)$, et elle représentera le nombre d'instructions élémentaires effectuées par l'ensemble de l'algorithme, le tout dépendant d'un paramètre n de *taille du problème* (directement liée à la taille des données).

Une complexité $T(n)$ est dite *en grand O de* $f(n)$ ou en $O(f(n))$ si on peut trouver $n_0 \in \mathbb{N}$ et $c > 0$ tels que $T(n) \leq cf(n)$ pour tout $n \geq n_0$. Ainsi par exemple, si nous avons un algorithme dont la complexité maximale est $T(n) = 2n^3 + 7n^2 + 20n + 11$ nous parlerons d'algorithme en $O(n^3)$.

Ces complexités n'ont de sens que dans un modèle de calcul précis qu'il faut déterminer. Par exemple, compter, au cours d'un programme, le nombre de comparaisons, le nombre d'additions, le nombre de multiplications, ou plus largement le nombre d'appels à une fonction particulière ou le nombre de lectures de certaines données (on peut encore imaginer nombre d'autres critères). Pour un modèle déterminé, on est amené à définir différentes classes de complexités particulièrement fréquentes (il en existe cependant bien d'autres) présentées dans le tableau 1.1.

Remarque 1.3 *Dans la notion d'algorithme logarithmique, la base n'importe pas si on calcule en termes de* $O(.)$*. En effet, on a* $\log_b(n) = \log_a(b) \log_a(n) = C^{te} \log_a(n)$*, et donc* $O(\log_b(n)) = O(\log_a(n))$*.*

Grand O	Classe d'algorithmes
$O(1)$	Constant
$O(\log(n))$	logarithmique
$O(n)$	Linéaire
$O(n \cdot \log(n))$	$n \cdot \log(n)$
$O(n^2)$	Quadratique
$O(n^3)$	Cubique
$O(a^n)$	Exponentiel
$O(n!)$	Factoriel

Tab. 1.1 – Classes de complexité les plus courantes.

1.3 Classifications de complexité

Nous avons vu que l'on pouvait classer une grande partie des problèmes dans un tableau avec différents types de complexités, mais ce n'est qu'un classement représentatif peu intéressant.

Notre but, ici, n'est pas de regarder des problèmes que l'on pourrait qualifier de 'simples' d'un point de vue algorithmique, ni trop compliqués. Nous allons nous intéresser à la frontière séparant les problèmes que l'on peut résoudre en temps polynomial, et ceux que l'on ne peut pas résoudre en temps polynomial. Et nous porterons notre attention sur une classe toute particulière de problèmes, pour lesquels nous ne connaissons ni algorithme en temps polynomial, ni preuve qu'il n'en existe pas. Une grande partie de ces problèmes ont été caractérisés, et on sait qu'ils sont équivalents au sens suivant : s'il existe un algorithme en temps polynomial pour l'un d'entre eux, alors il en existe un pour chacun d'entre eux. Ce sont les problèmes dits $\mathcal{NP}$-complets, et nous verrons qu'ils peuvent apparaître dans des contextes très différents (mais pas si différents que ça d'un point de vue algorithmique).

1.3.1 Réduction d'un problème

Pour formaliser la notion de *réduction*, nous dirons qu'un problème X est au moins aussi difficile qu'un problème Y si Y peut être résolu au moyen de X. C'est-à-dire que si nous possédons une boîte noire capable de résoudre X, alors l'utilisation de cette boîte noire permet de résoudre Y. Seulement, cette utilisation de la boîte noire ne peut pas se faire n'importe comment, sans quoi tout problème pourrait être résolu à partir de n'importe quel autre.

Il faudra se poser la question suivante : « Des instances arbitraires d'un problème Y peuvent-elles être résolues par un nombre polynomial d'étapes standards plus un nombre polynomial d'appels à la boîte noire du problème X ? ». Si la réponse est « oui », on dira que le problème Y se réduit au problème X et on note $Y \leqslant_{\mathcal{P}} X$.

Notons bien qu'il ne s'agit pas d'une relation d'ordre. En effet, bien qu'elle soit réflexive ($X \leqslant_{\mathcal{P}} X$) et transitive ($X \leqslant_{\mathcal{P}} Y$ et $Y \leqslant_{\mathcal{P}} Z \Rightarrow X \leqslant_{\mathcal{P}} Z$), nous verrons qu'elle

n'est pas antisymétrique, c'est-à-dire qu'on pourra trouver des problèmes distincts X et Y tels que $X \leqslant_{\mathcal{P}} Y$ et $Y \leqslant_{\mathcal{P}} X$; c'est en fait le cas pour tous les problèmes $\mathcal{NP}$-complets.

Proposition 1.4 *Supposons que $Y \leqslant_{\mathcal{P}} X$.*
Si X peut être résolu en temps polynomial, alors Y peut également être résolu en temps polynomial. Inversément, si Y ne peut être résolu en temps polynomial, alors X ne le peut pas non plus.

1.3.2 Les ensembles $\mathcal{P}$ et $\mathcal{NP}$

Portons notre attention sur le fait qu'il existe une distinction cruciale entre trouver une solution d'un problème donné et tester si une solution proposée est valide. Par exemple, considérons un graphe $G(V, E)$. On appelle *couverture* un ensemble S de sommets de V tel que toute arête de E possède une extrémité dans S. Il est extrêmement simple de vérifier si une donnée C est une couverture de taille donnée, mais il est difficile de trouver une couverture de taille donnée dans G.

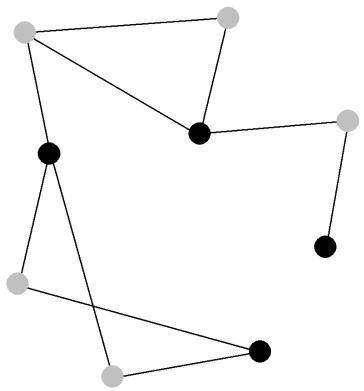

FIG. 1.3 – Exemple de couverture de graphe par les sommets

Considérons un problème P quelconque. Bien souvent, la réponse de ce problème ne sera pas « oui » ou « non ». Nous rencontrerons principalement des problèmes qui s'énonceront comme « Combien peut-on [...] au minimum ? » ou « Quel est le maximum de [...] ? ».

Prenons comme exemple concret le problème d'*Ensemble Indépendant* (EI) qui s'énonce dans sa forme initiale :
« Étant donné un graphe $G(V, E)$, on dit que l'ensemble de sommets $S \subseteq V$ est *indépendant* si deux sommets quelconques de S ne sont jamais joints par une arête de E. Il s'agit alors de trouver un ensemble indépendant de taille maximale. »
On dit que le problème est sous la forme de *problème d'optimisation*. Il n'est pas facile de formaliser une réponse de ce type, c'est pourquoi nous transformerons notre problème

de la manière suivante :
« Étant donné un graphe $G(V, E)$ et un entier k, peut-on trouver dans G un ensemble $S \subseteq V$ de sommets indépendants et de dimension au moins k ? »
On dira alors que le problème est sous la forme de *problème de décision*. Il n'y a évidemment pas de différence significative entre ces deux formulations, seulement la seconde version rend l'utilisation de notre boîte noire beaucoup plus aisée.

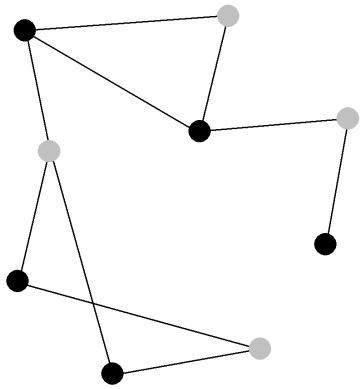

FIG. 1.4 – Exemple d'ensemble indépendant dans un graphe

Formalisons la notion de *problème*. Étant donné un problème Q, on suppose qu'il possède une *entrée* (les données) et une *sortie* (le résultat) dont les structures sont fixées par le problème et dont seules les valeurs varient selon les différentes instances du problème. On représentera l'entrée par une chaîne s finie de caractères binaires. On note $|s|$ la longueur de cette chaîne. On identifie à cette chaîne un problème de décision pour lequel la réponse sera « oui » ou « non ». Ainsi un algorithme A établi pour ce problème (un algorithme est bien établi pour un problème et non pour une instance d'un problème) reçoit la chaîne s en entrée et renvoie $A(s) \in \{0, 1\}$ en sortie.

Nous parlerons ici de problèmes *polynomiaux* et *non-polynomiaux*. On dira que l'algorithme A est en *temps polynomial* s'il existe un polynôme P tel que pour toute entrée s l'algorithme se termine en tout au plus $O(P(|s|))$ étapes. On note $\mathcal{P}$ l'ensemble des problèmes de décision qui se résolvent en temps polynomial.

Nous avons notre entrée s, on présente ensuite une chaîne t de *certification* qui contient la preuve que s est une instance de notre problème fournissant « oui » en sortie. Un algorithme B recevant en entrée deux chaînes s et t sera appelé une *certification efficace* pour un problème donné si B est en temps polynomial et s'il existe un polynôme P tel que pour toute instance s du problème dont la réponse est « oui », il existe une certification t telle que $|t| \leq P(|s|)$ et telle que $B(s, t) = 1$. On note $\mathcal{NP}$ l'ensemble des problèmes de décision pour lesquels il existe une certification efficace. En d'autres termes, il s'agit de l'ensemble des problèmes dont la vérification d'une solution proposée

se fait en temps polynomial. Notons bien qu'on ne prend pas en considération le temps de recherche d'un tel élément t.

Exemple 1 *Considérons le problème qui, pour un graphe $G(V, E)$ donné où l'on pose $n = |V|$, pour deux sommets distincts u, v choisis dans V et pour un entier $k > 0$ donné, consiste à trouver un chemin de longueur k entre u et v. Connaissant la structure du problème, les données peuvent se résumer à une suite de nombres entiers, constituée des $\frac{1}{2}n(n-1)$ valeurs booléennes significatives d'une matrice symétrique de taille n (dont les éléments de la diagonale n'importent pas) représentant les sommets du graphe joints ou non par une arête, suivis des indices des deux sommets u et v donnés. La chaîne de vérification t serait alors une suite d'indices de sommets, constituée de $k+1$ nombres naturels. La certification efficace se contenterait alors de vérifier que pour chaque paire d'indices succesifs a, b l'élément (a, b) dans la matrice soit la valeur booléenne 1. Cela se fait en temps linéaire, notre problème est donc dans $\mathcal{NP}$.*

Proposition 1.5 $\mathcal{P} \subseteq \mathcal{NP}$

Démonstration 1.5 Considérons un problème $X \in \mathcal{P}$. Cela signifie qu'il existe un algorithme A, en temps polynomial, qui résoud le problème X pour toute entrée s. Montrons que l'on peut fabriquer une certification efficace du problème :
Posons $B(s, t) = A(s)$ pour tout s, t. Il est évident que B est en temps polynomial, puisque A l'est. Considérons maintenant une entrée s telle que $A(s) = 1$, alors pour toute chaîne t dont la longueur est au plus $P(|s|)$, on a $B(s, t) = A(s) = 1$. B est donc bien une certification efficace du problème X. □

1.3.3 Les problèmes $\mathcal{NP}$-complets

Il est naturel de se demander quels peuvent être les problèmes les plus difficiles au sein de l'ensemble $\mathcal{NP}$. On dira qu'un problème X est *complet* dans $\mathcal{NP}$ si :

1. $X \in \mathcal{NP}$;
2. pour tout problème Y dans $\mathcal{NP}$, on a $Y \leqslant_{\mathcal{P}} X$.

En d'autres termes, un problème X est complet dans $\mathcal{NP}$ si tout problème de $\mathcal{NP}$ peut se ramener à X. On parlera de *problèmes $\mathcal{NP}$-complets.*

Si on impose à un problème X uniquement la condition :

1. pour tout problème Y dans $\mathcal{NP}$, on a $Y \leqslant_{\mathcal{P}} X$

on parlera de *problème $\mathcal{NP}$-difficile.*

Théorème 1.6 *Soit X un problème $\mathcal{NP}$-complet,*
alors X peut être résolu en temps polynomial si et seulement si $\mathcal{P} = \mathcal{NP}$.

Démonstration 1.6 Si X peut être résolu en temps polynomial, alors tout problème Y de $\mathcal{NP}$ peut se ramener à X et peut donc être résolu en temps polynomial, ce qui signifie qu'il est dans $\mathcal{P}$.
S'il existe X qui ne peut être résolu en temps polynomial, alors il n'est pas dans $\mathcal{P}$ tout en étant dans $\mathcal{NP}$, et donc $\mathcal{P} \neq \mathcal{NP}$. □

Corollaire 1.7 *S'il existe ne serait-ce qu'un seul problème de* $\mathcal{NP}$ *qui ne peut être résolu en temps polynomial, alors* $\mathcal{P} \neq \mathcal{NP}$.

1.3.4 Un problème $\mathcal{NP}$-complet

Nous avons là une belle définition, mais rien ne prouve que de tels problèmes existent effectivement. Nous allons construire un premier problème $\mathcal{NP}$-complet qui servira de base pour démontrer la $\mathcal{NP}$-complétude de nombreux autres problèmes. En effet, si on a un problème X que l'on sait $\mathcal{NP}$-complet, il suffit de montrer que X peut se ramener à un problème choisi Y de $\mathcal{NP}$ pour savoir que Y est également $\mathcal{NP}$-complet.

Le problème que nous allons construire porte le nom de *satisfaisabilité de circuit* (SC).
Considérons un graphe G dirigé acyclique et connexe. Chaque noeud N de G possède $e_N \in \{0, 1, 2\}$ arcs *entrant* (c'est-à-dire des couples (N', N)) et $s_N \in \mathbb{N}$ arcs *sortant* (c'est-à-dire des couples (N, N'')), avec les propriétés suivantes :

1. Les noeuds qui n'ont pas d'arcs entrant sont appelés les *données* du graphe ;
2. Les noeuds qui n'ont qu'un arc entrant sont appelés les *opérateurs unaires* du graphe ;
3. Les noeuds qui ont deux arcs entrant sont appelés les *opérateurs binaires* du graphe ;
4. Le noeud qui n'a pas d'arc sortant existe et est unique dans le graphe, on l'appelle la *réponse* du graphe.

On associe à chaque donnée du graphe une valeur booléenne (0 ou 1) constante ou une variable booléenne (qu'on appelle les *input*). On associe à chaque opérateur unaire l'opérateur d'identité ou l'opérateur de négation. On associe à chaque opérateur binaire l'opérateur 'ou' logique ou l'opérateur 'et' logique. Le graphe dirigé sert alors de transmetteur d'information entre les valeurs sortant des noeuds, pour aller aux noeuds suivants via les arcs. Le problème est alors le suivant :
« Étant donné un tel graphe, existe-t-il une séquence booléenne à fournir en input au graphe telle que la réponse du graphe soit la valeur booléenne '1' ? »

Proposition 1.8 *Le problème de satisfaisabilité de circuit est un problème* $\mathcal{NP}$*-complet.*

Démonstration 1.8 Nous nous contenterons d'une esquisse de la démonstration.
Pour démontrer la proposition, il suffirait de considérer un problème quelconque X dans $\mathcal{NP}$ et de montrer que X peut se ramener au problème de satisfaisabilité de

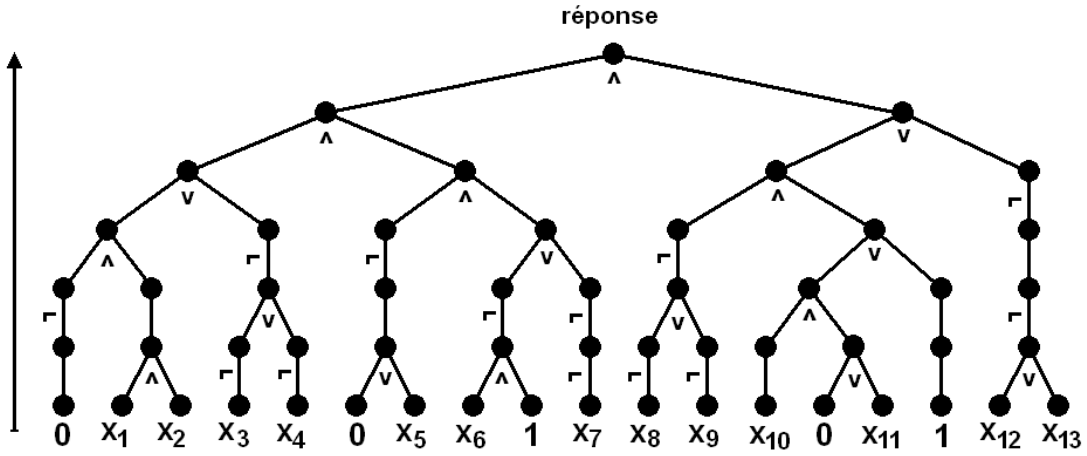

FIG. 1.5 – Exemple d'instance du problème de Satisfaisabilité de circuit

circuit. On utilise le fait que tout algorithme dont l'entrée est une chaîne de caratères de type booléen de longueur finie et produisant une réponse dans {« oui »,« non »} peut s'exprimer comme un graphe tel que nous l'avons défini ci-dessus, et dont la réponse est la même que l'algorithme pour une même donnée en input. La preuve de cette affirmation devrait se faire par voie d'expressions logiques, sachant que tout algorithme implémenté sur une machine[1] est codé uniquement à l'aide de valeurs booléennes et est exécuté avec les opérations « ou », « et » et « non ». La démonstration se base alors sur la notion de machine, en particulier de machine de Turing. □

1.3.5 Évolution

Maintenant que nous avons en main un problème $\mathcal{NP}$-complet, non seulement nous savons qu'il en existe, mais nous avons là un magnifique outil auquel nous pourrons nous ramener lorsque nous souhaiterons montrer la $\mathcal{NP}$-complétude d'autres problèmes. Regardons en particulier le problème SAT et son cas particulier 3-SAT :

Supposons que nous ayons un ensemble X composé de n variables booléennes $x_1, x_2, \ldots, x_n$. On appelle *terme* une variable x_i ou sa négation $\overline{x_i}$. On appelle *clause* une disjonction de termes distincts $t_1 \vee t_2 \vee \ldots \vee t_k$, on dit dans ce cas que la clause est de longueur k. Une *attribution de vérité* sera une attribution de valeurs booléennes pour chaque $x_i \in X$; en d'autres termes, c'est une application $\nu : X \to \{0, 1\}$.

On dira que l'attribution de vérité ν *satisfait* une clause c si cette clause est évaluée à 1 avec cette attribution de vérité. Il est équivalent d'exiger qu'au moins l'un des t_j de la clause c se voit attribuer la valeur 1. On dira que l'attribution de vérité satisfait une collection de clauses $c_1, c_2, \ldots, c_q$ si l'expression $c_1 \wedge c_2 \wedge \cdots \wedge c_q$ est évaluée à 1 avec cette attribution de vérité.

[1] Se rapporter à la section 1.2 pour la notion de machine.

Le problème de satisfaisabilité ou problème SAT se formule de la manière suivante : « Étant donné une collection de clauses $c_1, c_2, \ldots, c_q$ sur un ensemble de variables $X = \{x_1, x_2, \ldots, x_n\}$, existe-t-il une attribution de vérité satisfaisante ? »

Le problème de 3-satisfaisabilité ou problème 3-SAT, cas particulier du cas précédent SAT, se formule de la manière suivante : « Étant donné une collection de clauses $c_1, c_2, \ldots, c_q$, toutes de longueur 3, sur un ensemble de variables $X = \{x_1, x_2, \ldots, x_n\}$, existe-t-il une attribution de vérité satisfaisante ? »

Proposition 1.9 *Le problème 3-SAT est $\mathcal{NP}$-complet.*

Démonstration 1.9 Il est évident que ce problème est dans $\mathcal{NP}$, en effet vérifier une solution donnée se fait en temps linéaire, donc polynomial. Il nous suffit donc de montrer la réduction $SC \leqslant_{\mathcal{P}} 3\text{-}SAT$. Considérons donc une instance arbitraire du problème de satisfaisabilité de circuit. Nous allons tout d'abord construire une instance équivalente du problème SAT dont les clauses sont au plus de longueur 3. Nous convertirons alors ce problème en une instance du problème 3-SAT, ce qui complétera la démonstration.

Considérons un graphe arbitraire K défini comme au début de la section 1.3.4. On associe une variable x_v à chaque noeud v du graphe K, qui contiendra la valeur booléenne de celui-ci. Il nous faut maintenant définir un ensemble de clauses sur les variables x_i. Nous devons tout d'abord traduire le fait que certains noeuds adjacents ont leurs valeurs soumises à certaines conditions, il y a quatre cas possibles :

1. Si le noeud v se voit attribuer l'opérateur d'identité, et si le noeud qui le précède dans K est le noeud u, alors nous avons la condition $x_v = x_u$ que l'on traduit en termes de clauses par $(\overline{x_v} \vee x_u) \wedge (x_v \vee \overline{x_u})$;
2. Si le noeud v se voit attribuer l'opérateur de négation, et si le noeud qui le précède dans K est le noeud u, alors nous avons la condition $x_v = \overline{x_u}$, que l'on traduit en termes de clauses par $(x_v \vee x_u) \wedge (\overline{x_v} \vee \overline{x_u})$;
3. Si le noeud v se voit attribuer l'opérateur logique « ou », et si les deux noeuds qui le précèdent dans K sont les noeuds u et w, alors nous avons la condition $x_v = x_u \vee x_w$, que l'on traduit en termes de clauses par $(x_v \vee \overline{x_u}) \wedge (x_v \vee \overline{x_w}) \wedge (\overline{x_v} \vee x_u \vee x_w)$;
4. Si le noeud v se voit attribuer l'opérateur logique « et », et si les deux noeuds qui le précèdent dans K sont les noeuds u et w, alors nous avons la condition $x_v = x_u \wedge x_w$, qui se traduit en termes de clauses par $(\overline{x_v} \vee x_u) \wedge (\overline{x_v} \vee x_w) \wedge (x_v \vee \overline{x_u} \vee \overline{x_w})$.

Enfin nous devons garantir que les constantes en entrée ont bien leur valeur spécifiée et que le résultat est évalué à 1. Ainsi, pour une valeur constante v de K, on impose la clause (x_v) ou la clause $(\overline{x_v})$ selon la valeur booléenne que doit prendre x_v, et pour le noeud r du résultat on ajoute la clause (x_r). Ceci termine de construire notre instance du problème SAT, il n'est pas difficile de voir qu'il est bien équivalent à notre problème initial de satisfaisabilité de circuit.

Il nous faut maintenant modifier nos clauses pour les rendre de longueur 3. Pour cela, on introduit quatre nouvelles variables booléennes fictives z_1,z_2,z_3 et z_4. L'idée est de s'assurer que nous aurons toujours $z_1 = z_2 = 0$. Pour ce faire, nous créons huit premières

clauses à notre instance de 3-SAT qui sont les suivantes : $(\overline{z_1} \vee z_3 \vee z_4) \wedge (\overline{z_1} \vee \overline{z_2} \vee z_4) \wedge (\overline{z_1} \vee z_3 \vee \overline{z_4}) \wedge (\overline{z_1} \vee \overline{z_3} \vee \overline{z_4}) \wedge (\overline{z_2} \vee z_3 \vee z_4) \wedge (\overline{z_2} \vee \overline{z_2} \vee z_4) \wedge (\overline{z_2} \vee z_3 \vee \overline{z_4}) \wedge (\overline{z_2} \vee \overline{z_3} \vee \overline{z_4})$. Ensuite, nous conservons toutes les clauses de longueur 3 de l'instance de SAT créée ci-dessus, puis nous transformons les clauses $(t_1 \vee t_2)$ de longueur 2 en clauses de longueur 3 $(t_1 \vee t_2 \vee z_1)$, et enfin les clauses (t) de longueur unitaire sont remplacées par $(t \vee z_1 \vee z_2)$. Nous conservons ainsi les mêmes conditions, mais sur des clauses de longueur 3.

Nous avons donc bien ramené le problème de satisfaisabilité de circuit au problème 3-SAT. □

1.4 Algorithmes d'approximation

Nous connaissons déjà un certain nombre de problèmes de décision $\mathcal{NP}$-complets, et nous en rencontrerons beaucoup d'autres. Il est établi que si $\mathcal{P} \neq \mathcal{NP}$, il n'existe pas d'algorithme polynomial qui puisse résoudre ces problèmes. Cependant ces problèmes de décision sont bien souvent liés à des problèmes d'optimisation (en fait, dans les problèmes que nous étudierons ici, tous les problèmes de décision sont directement issus de problèmes d'optimisation, qui forment le véritable objectif). Nous retrouverons ces notions dans *An introduction to convexity, polyhedral theory and combinatorial optimisation* de Geir Dahl [3] et *Algorithm Design* de Jon Kleinberg et Eva Tardos [5].

Considérons par exemple le problème de *Couverture par les sommets*(*Vertex Cover* dans la littérature en anglais) : Dans un graphe $G(V, E)$, on dit que l'ensemble de sommets $S \subseteq V$ est une *couverture par les sommets* si toute arête de E a au moins une de ses extrémités dans S. On cherche alors un ensemble S de taille minimum qui soit une couverture. Le problème de décision associé s'exprime de la façon suivante : « Étant donné un graphe $G(V, E)$ et un entier k, existe-t-il dans V une couverture par les sommets de taille au plus k ? ». Nous verrons dans le chapitre suivant que le problème de décision de Couverture par les sommets est un problème $\mathcal{NP}$-complet.

En supposant que $\mathcal{P} \neq \mathcal{NP}$, on voit facilement que ce problème d'optimisation ne possède pas d'algorithme polynomial. En effet, si un tel algorithme existait nous pourrions calculer la dimension de l'ensemble S renvoyé par l'algorithme et le comparer à un k choisi pour une instance quelconque du problème de décision qui est $\mathcal{NP}$-complet. Cela fournirait un algorithme polynomial pour le problème de décision, ce qui aurait pour conséquence que $\mathcal{P} = \mathcal{NP}$.

On est souvent amené à résoudre des problèmes cherchant un minimum ou un maximum d'une valeur en respectant un ensemble de contraintes données. On appelle *problème d'optimisation* un problème P du type

$$(P) \equiv maximiser\{f(x)|x \in A\} \tag{1.7}$$

où $f : A \to \mathbb{R}$ est une fonction définie appelée *fonction objectif*, où A est un ensemble donné appelé *région réalisable*, et chaque élément de A est appelé *solution réalisable*. En ce qui nous concerne, f sera généralement une fonctionnelle sur un ensemble de bijections.

Un élément x^* est appelé *solution optimale* sur A du problème P si $f(x^*) \geq f(x)$ pour tout x dans A. On appelle *valeur optimale* du problème P le réel $v(P) = \sup\{f(x)|x \in A\}$. Notons que $f(x^*) = v(P)$. On dit qu'une solution x_1 est *meilleure* qu'une solution x_2 lorsque $f(x_1) > f(x_2)$. Remarquons qu'un problème du type

$$(P) \equiv minimiser\{f(x)|x \in A\} \tag{1.8}$$

est un problème d'optimisation en posant $f'(x) = -f(x)$ et en regardant le problème

$$(P) \equiv maximiser\{f'(x)|x \in A\} \tag{1.9}$$

Dans notre exemple du problème de Couverture par les sommets, la région réalisable serait l'ensemble des fonctions qui ordonnent les sommets d'un graphe donné $G(V, E)$, et la fonction objectif serait la fonction qui compte le nombre de sommets nécessaires (dans un ordre donné par la solution réalisable) pour obtenir une couverture du graphe.

Nous appelons *algorithme de ρ-approximation* un algorithme polynomial renvoyant une solution approchée au pire cas à un facteur ρ de la solution optimale. Autrement dit, si OPT est la solution optimale de l'instance du problème et $ALGO$ la solution renvoyée par l'algorithme de ρ-approximation, alors nous avons la certitude que

$$OPT \leq ALGO \leq \rho \cdot OPT \tag{1.10}$$

Considérons l'algorithme polynomial ci-dessous, qui fournit une solution approchée du problème de Couverture par les sommets pour un graphe donnée $G(V, E)$.

1. un *ensemble de sommets* $S \leftarrow \emptyset$
2. un *ensemble d'arêtes* $T \leftarrow E$
3. tant que $T \neq \emptyset$ faire :
   (a) sélectionner une arête $e = \{x, y\}$ dans T
   (b) ajouter $\{x, y\}$ à S
   (c) pour chaque arête e' de T adjacente à x : supprimer e' de T
   (d) pour chaque arête e' de T adjacente à y : supprimer e' de T
4. retourner S

Proposition 1.10 *Cet algorithme est un algorithme de 2-approximation pour le problème d'optimisation de Couverture par les sommets.*

Démonstration 1.10 Soit A l'ensemble des arêtes sélectionnées par l'algorithme. On sait que $|S| = 2 \cdot |A|$ car chaque arête sélectionnée ajoute deux sommets à S, et deux arêtes distinctes dans A n'ont pas de sommet commun. Or $OPT \geq |A|$ car il faut au moins couvrir les arêtes de A. Ainsi $ALGO = |S| \leq 2 \cdot OPT$. □

On appellera *facteur d'inapproximabilité* pour un problème d'approximation donné le plus petit ρ tel qu'il soit possible de trouver un algorithme de ρ-approximation pour ce problème.

De façon plus générale, on appellera *algorithme de $\rho(n)$-approximation* un algorithme polynomial pour un problème de taille n renvoyant une solution approchée au pire cas à un facteur $\rho(n)$ de l'optimal. Nous verrons au chapitre suivant que c'est par exemple le cas du problème de Couverture d'ensemble, qui est $\alpha \log(n)$-approximable, où α est une constante dépendant de l'instance du problème. La notion de facteur d'inapproximabilité s'étend naturellement aux algorithmes de $\rho(n)$-approximations comme *fonction d'inapproximabilité*.

Chapitre 2

Problème de Couverture d'ensemble

Une école compte 223 enfants habitant tous à des endroits différents. 25 lignes de bus desservent cette école, couvrant l'ensemble des habitations des enfants. Chaque enfant ou groupe d'enfants ne peut circuler sans être accompagné d'un moniteur. Dans quels bus mettre quels enfants pour minimiser le nombre de moniteurs nécessaires ?

Voilà un problème tout à fait anodin qui créera bien des surprises à qui voudrait en trouver une solution optimale. En effet, le temps de calcul nécessaire pour trouver une solution optimale de ce genre de problème devrait être absolument gigantesque, à moins bien entendu que $\mathcal{P} = \mathcal{NP}$, et il est probablement impossible d'entreprendre de tels calculs.

Par contre nous allons voir que nous pouvons facilement utiliser un algorithme en temps polynomial qui, dans le cas présent, nous fournirait une proposition de répartition utilisant dans le pire des cas moins de six fois la quantité de moniteurs optimale, ce qui peut devenir assez intéressant si par hasard la quantité optimale était assez petite.

Le problème de *Couverture d'ensemble* (*Set Cover* dans la littérature en anglais) est un problème très général, qui pourra bien souvent apparaître dans la résolution d'autres problèmes apparemment sans lien.

2.1 Présentation du problème

Considérons un hypergraphe $H(V, E)$ où $|V| = n$ et $|E| = m$. On dira que $\mathcal{C} \subseteq E$ est une *couverture* si pour tout sommet $v \in V$ on peut trouver $e \in \mathcal{C}$ tel que $v \in e$. On pose C_H l'ensemble des couvertures sur H. Considérons une fonction $w : E \to \mathbb{R}^+ : e \mapsto w(e)$, on appelle $w(e)$ le *poids* de l'hyperarête e. Le but du problème est de trouver une

couverture qui minimise la fonction objectif

$$F_{obj} : C_H \to \mathbb{R}^+ : \mathcal{C} \mapsto \sum_{e \in \mathcal{C}} w(e) \tag{2.1}$$

Remarque 2.1 *Le problème initial présenté dans* Algorithm Design *[5] envisage que E puisse contenir des singletons, mais cela ne modifie pas fondamentalement le problème. En effet, si on a dans E un singleton $\{v\} = e$ tel qu'aucun autre élément de E ne contienne v, alors on devra toujours sélectionner e dans notre couverture ; mais s'il existe un $e' \in E$ contenant v qui ne soit pas un singleton, alors il sera toujours plus intéressant de choisir e' plutôt que e dans notre couverture, et on peut donc oublier e. Cette opération sur l'ensemble des singletons n'influence pas sensiblement la résolution de notre problème.*

Remarque 2.2 *En passant à l'hypergraphe dual $H^*(E, V^*)$, on obtient une nouvelle forme du problème. On appellera $\mathcal{C} \subseteq E$ une* couverture duale *si pour toute arête $v^* \in V^*$ on peut trouver un sommet $e \in E$ tel que $e \in v^*$. Nous pouvons alors associer un poids à chacun des sommets du graphe et chercher une couverture duale qui minimise le poids total des sommets choisis. C'est une autre façon de voir le problème de Couverture d'ensemble.*

Remarque 2.3 *Un cas particulier habituel pour le problème de Couverture d'ensemble est d'associer à chaque hyperarête e un poids $w(e) = 1$, c'est-à-dire un cas particulier du problème qui cherche à minimiser le nombre d'hyperarêtes nécessaires pour couvrir l'ensemble des sommets. Nous verrons que ce cas particulier sera généralisable, contrairement au cas pondéré.*

2.2 $\mathcal{NP}$-complétude

Nous allons montrer, présenté comme dans *Algorithm Design* [5], que le problème de Couverture d'ensemble est $\mathcal{NP}$-complet. Nous avions déjà rencontré le problème de Couverture par les sommets dans la section 1.4, voici un autre problème qui nous sera bien utile pour montrer la $\mathcal{NP}$-complétude du problème de Couverture d'ensemble.

Dans un graphe $G(V, E)$, on dit que l'ensemble de sommets $S \subseteq V$ est *indépendant* si deux sommets quelconques de S ne sont jamais adjacents. Le problème de décision associé au problème d'*Ensemble indépendant* (*Independent Set* dans la littérature en anglais) s'exprime comme suit :
« Étant donné un graphe $G(V, E)$ et un entier k, peut-on trouver dans V un ensemble de sommets indépendants et de taille au moins k ? ».

Théorème 2.4 *En attribuant naturellement les abréviations EI pour le problème d'Ensemble indépendant, CS pour le problème de Couverture par les sommets et CE pour le problème de Couverture d'ensemble, on obtient les relations suivantes :*

$$3\text{-}SAT \leqslant_{\mathcal{P}} EI \leqslant_{\mathcal{P}} CS \leqslant_{\mathcal{P}} CE \tag{2.2}$$

Démonstration 2.4

Montrons la première réduction 3-$SAT \leqslant_{\mathcal{P}} EI$.
On suppose que l'on possède une boîte noire pour le problème d'Ensemble indépendant et nous souhaitons résoudre une instance du problème 3-SAT construit sur les clauses $c_1, \ldots, c_k$ et les variables $x_1, \ldots, x_n$. Pour résoudre cette instance de 3-SAT, il suffit de choisir un terme de chaque clause et de fournir une attribution de vérité pour laquelle chacun de ses termes est évalué à 1. On dira que deux termes sont en *conflit* si l'on veut que l'un soit x_i et l'autre $\overline{x_i}$. Ainsi, il suffit de piocher un terme de chaque clause de façon à ce qu'il n'y ait jamais deux termes en conflit. Voyons cela de façon graphique : on construit un graphe $G(V, E)$ constitué de $3k$ sommets groupés en k triangles. Pour chaque triangle i, on a les sommets v_{i1}, v_{i2}, v_{i3} reliés par trois arêtes. On attribue v_{ij} au j-ème terme de la i-ème clause. On va devoir piocher un sommet de chaque triangle (jamais deux points d'un même triangle (c'est-à-dire de la même clause), c'est pourquoi on les relie par une arête) de façon à ce que deux quelconques de ces points ne soient pas en conflit. Pour cela, on relie par une arête toute paire de points $(v_{i_1 j_1}, v_{i_2 j_2})$ qui sont en conflit. Notre problème sera résolu seulement si le graphe G possède un ensemble indépendant de sommets de taille au moins k. On ne fait qu'un seul appel à la boîte noire du problème d'Ensemble indépendant, et la construction se fait en $O(n^2)$, on a donc bien notre réduction.

Montrons la deuxième réduction $EI \leqslant_{\mathcal{P}} CS$.
Cette fois nous possédons une boîte noire pour résoudre le problème de Couverture par les sommets, et nous souhaiterions résoudre une instance du problème d'Ensemble indépendant donnée par un graphe $G(V, E)$ et un entier k. Il suffit de voir que $S \subseteq V$ sera un ensemble indépendant de sommets si et seulement si $V \setminus S$ est une couverture par les sommets. En effet, supposons S indépendant, et considérons une arête $\{u, v\} = e \in E$. Alors il est impossible que u et v soient tous les deux dans S, et l'un des deux est donc dans $V \setminus S$. Toute arête possède donc une extrémité dans $V \setminus S$, et $V \setminus S$ est bien une couverture par les sommets. Supposons maintenant que $V \setminus S$ est une couverture par les sommets. Alors s'il existait u et v dans S reliés par une arête, cette arête n'aurait aucune extrémité dans $V \setminus S$, et $V \setminus S$ ne serait pas une couverture par les sommets. Il n'existe donc pas deux sommets reliés par une arête et S est donc bien un ensemble indépendant.

Il nous reste à montrer la troisième réduction $CS \leqslant_{\mathcal{P}} CE$.
Formulons le problème de Couverture d'ensemble non pondéré de la façon suivante (problème de décision) : « Soit un ensemble U de n éléments, et soit une collection $S_1, \ldots, S_m$ de sous-ensembles de U, et soit k un entier. Existe-t-il une collection d'au plus k des sous-ensembles $S_{j_1}, \ldots, S_{j_k}$ dont l'union soit égale à U ? ». Supposons que l'on possède une boîte noire pour résoudre ce problème de Couverture d'ensemble non pondéré, et considérons une instance du problème de Couverture par les sommets donnée par un graphe $G(V, E)$ et un entier k. Si on pose $U = E$ et si pour tout i de 1 à m on pose S_i l'ensemble des arêtes incidentes au sommet v_i, alors en gardant le même entier k on est tout de suite ramené au problème de Couverture d'ensemble non

pondéré. Il est de plus évident que si l'on possède une boîte noire pour le problème de Couverture d'ensemble pondéré, alors on résoud immédiatement le cas particulier non pondéré. On obtient ainsi la troisième réduction. □

Corollaire 2.5 *Le problème de Couverture d'ensemble est $\mathcal{NP}$-complet.*

Démonstration 2.5 Nous avons 3-$SAT \leqslant_{\mathcal{P}} CE$, et nous avons vu au théorème 1.9 que 3-SAT est $\mathcal{NP}$-complet. D'autre part, le problème de Couverture d'ensemble est dans $\mathcal{NP}$ car tester qu'un ensemble d'hyperarêtes est bien une couverture se fait en temps polynomial. Cela suffit à démontrer le corollaire. □

2.3 Algorithme glouton

On appelle habituellement *algorithme glouton* un algorithme cherchant, étape par étape, à maximiser une valeur locale, avec l'espoir de maximiser un résultat global.

Exemple 2 *Le problème de rendu de monnaie donne un exemple simple. Il s'agit de pouvoir rendre une certaine somme d'argent avec un nombre minimum de pièces. Dans la résolution de ce problème, l'algorithme glouton serait celui consistant à choisir, à chaque étape, la plus grande pièce plus petite que la somme encore à rendre. Selon le système de pièces, l'algorithme glouton est optimal ou non. Par exemple, avec le système de pièces européen (1,2,5,10,20,50,100,200 centimes), on peut montrer que l'algorithme glouton sera optimal. Pour une somme de 37 centimes à rendre, il choisira successivement des pièces de 20,10,5,2 centimes. Par contre dans un système dont les pièces sont 1,3,4 centimes, l'algorithme glouton rendra 6 centimes sous la forme 4+1+1, alors que la solution 3+3 est optimale.*

Nous allons développer et analyser un algorithme glouton pour le problème de Couverture d'ensemble. Cet algorithme se veut à la fois simple, efficace en terme de solution rapportée, et rapide. Nous décrirons cet algorithme pour le problème de Couverture d'ensemble sous sa forme approximative « Étant donné un hypergraphe $H(V, E)$ pondéré par une application $w : E \to \mathbb{R}^+$, on cherche une séquence $e_1, \ldots, e_k$ dans E telle que $\bigcup_{i=1}^{k} e_i = V$ et qui minimise la somme $\sum_{i=1}^{k} w(e_i)$ »où k n'est pas déterminé à l'avance. Pratiquement, on l'implémente comme ceci :

1. un *ensemble de sommets* $R \leftarrow V$
2. un *ensemble d'hyperarêtes* $D \leftarrow \emptyset$
3. tant que $R \neq \emptyset$ faire :
   (a) sélectionner e dans $E \backslash D$ qui minimise $\frac{w(e)}{|R \cap e|}$
   (b) pour chaque $v \in e$, définir $c_v \equiv \frac{w(e)}{|R \cap e|}$ si ce n'est pas encore défini
   (c) ajouter e dans D

(d) supprimer de R les sommets de e qui s'y trouvent

4. retourner D

En d'autres termes, l'algorithme sélectionne à chaque étape l'hyperarête qui minimise le rapport entre son poids et le nombre de sommets qu'elle couvrira, et ce tant qu'il y a des sommets non couverts. L'algorithme définit R comme l'ensemble évolutif des sommets à couvrir et D comme l'ensemble des hyperarêtes sélectionnées. On voit facilement que notre algorithme glouton est polynomial en $O(n^2)$.

Remarque 2.6 *La valeur de c_v introduite ne perturbe en rien la résolution de l'algorithme, mais elle nous servira pour connaître l'approximation de notre résultat. Intuitivement, il s'agit du prix à payer pour la couverture du sommet v avec l'hyperarête e ; et nous voulons minimiser le prix total de recouvrement de tous les éléments de V.*

Proposition 2.7 *Si $\mathcal{D}$ est la couverture obtenue par l'algorithme glouton pour un hypergraphe $H(V, E)$, alors on a l'égalité suivante :*

$$\sum_{e \in \mathcal{D}} w(e) = \sum_{v \in V} c_v \tag{2.3}$$

Démonstration 2.7 On note R_e la valeur de R au début de l'étape où l'algorithme glouton sélectionnera e comme nouvelle hyperarête. On voit facilement que l'on a les propriétés suivantes :

1. Si $e_1 \neq e_2$ dans $\mathcal{D}$, alors

$$(R_{e_1} \cap e_1) \cap (R_{e_2} \cap e_2) = \emptyset \tag{2.4}$$

2. L'union des sommets couverts à chaque étape est V tout entier, c'est-à-dire

$$\bigcup_{e \in \mathcal{D}} (R_e \cap e) = V \tag{2.5}$$

Ainsi les $(R_e \cap e)$ avec e dans $\mathcal{D}$ forment une partition de V, ce qui fournit, pour toute fonction $f : V \to \mathbb{R} : v \mapsto f(v)$ l'égalité

$$\sum_{e \in \mathcal{D}} \sum_{v \in (R_e \cap e)} f(v) = \sum_{v \in V} f(v) \tag{2.6}$$

On se rappelle que $c_v = \frac{w(e)}{|R_e \cap e|}$, donc

$$\sum_{v \in (R_e \cap e)} c_v = w(e) \tag{2.7}$$

ce qui fournit

$$\sum_{e \in \mathcal{D}} w(e) = \sum_{e \in \mathcal{D}} \left(\sum_{v \in (R_e \cap e)} c_v \right) = \sum_{v \in V} c_v \tag{2.8}$$

□

2.4 Facteur d'approximation

Rappelons que le problème de Couverture d'ensemble est $\mathcal{NP}$-complet, cela signifie qu'il n'existe pas d'algorithme polynomial capable de le résoudre, à moins toujours que $\mathcal{P}$ soit égal à $\mathcal{NP}$. Notre algorithme glouton n'est donc certainement pas optimal, et ne fournira la solution optimale que par simple coïncidence. Cependant nous pouvons regarder en quelle mesure la solution qu'il nous fournit est proche de la solution optimale. En particulier, nous aimerions fixer une borne supérieure sur la valeur renvoyée par la fonction objectif lors de l'exécution de l'algorithme glouton, par rapport à la valeur associée à une couverture optimale. Les résultats suivants sont présentés et démontrés dans *Algorithme Design* [5].

On rappelle la définition de la fonction *harmonique*. Il s'agit de la fonction

$$H : \mathbb{N} \to \mathbb{R} : n \mapsto \sum_{k=1}^{n} \frac{1}{k} \tag{2.9}$$

La fonction harmonique H est en $O(\log(n))$. En effet, il suffit de remarquer que

$$H(n) \leq 1 + \int_1^n \frac{dx}{x} = 1 + \ln(n) \tag{2.10}$$

Lemme 2.8 *Pour toute hyperarête $e \in \mathcal{D}$, on a l'inégalité*

$$\sum_{v \in e} c_v \geq H(|e|) \cdot w(e) \tag{2.11}$$

Démonstration 2.8 Soit $e \in \mathcal{D}$. On pose $d = |e|$. On indice les éléments de e dans l'ordre dans lequel ils seront recouverts par l'algorithme glouton : on a donc $e = \{v_1, \ldots, v_d\}$. On a c_{v_j} le poids associé à l'élément v_j. Considérons l'étape précise pendant laquelle l'algorithme glouton recouvrira le sommet v_j à l'aide de l'hyperarête e_i (i dépendant de j et de la structure du graphe). Au début de cette étape, $v_j, v_{j+1}, \ldots, v_d$ sont dans R_{e_i}. Donc $|R_{e_i} \cap e|$ vaut au moins $d - j + 1$, et ainsi on a

$$\frac{w(e)}{|R_{e_i} \cap e|} \leq \frac{w(e)}{d - j + 1} \tag{2.12}$$

L'algorithme choisit e_i (avant ou en même temps que e) de manière à minimiser le rapport $\frac{w(e_i)}{|R_{e_i} \cap e_i|}$, donc forcément

$$\frac{w(e_i)}{|R_{e_i} \cap e_i|} \leq \frac{w(e)}{|R_{e_i} \cap e|} \tag{2.13}$$

Cela nous mène à la suite d'inégalités suivante :

$$c_{v_j} = \frac{w(e_i)}{|R_{e_i} \cap e_i|} \leq \frac{w(e)}{|R_{e_i} \cap e|} \leq \frac{w(e)}{d - j + 1} \tag{2.14}$$

On peut alors conclure

$$\sum_{v \in e} c_v = \sum_{j=1}^{d} c_{v_j} \tag{2.15}$$

$$\leq \sum_{j=1}^{d} \frac{w(e)}{d-j+1} \tag{2.16}$$

$$= \frac{w(e)}{d} + \frac{w(e)}{d-1} + \cdots + \frac{w(e)}{1} \tag{2.17}$$

$$= H(d) \cdot w(e) \tag{2.18}$$

□

Théorème 2.9 *Posons $d^* = max_{e \in E}|e|$ la taille de la plus grande hyperarête de notre hypergraphe, et posons w^* le poids d'une couverture optimale. Alors la couverture $\mathcal{D}$ sélectionnée par l'algorithme glouton a un poids de tout au plus $H(d^*) \cdot w^*$.*

Démonstration 2.9 Par le lemme, nous avons

$$w(e) \geq \frac{1}{H(d^*)} \sum_{v \in e} c_v \quad \text{pour tout } e \in \mathcal{D}^* \quad \text{où } \mathcal{D}^* \text{ est une couverture optimale} \tag{2.19}$$

De plus, comme $\mathcal{D}^*$ est une couverture de V, on a

$$\sum_{e \in \mathcal{D}^*} \sum_{v \in e} c_v \geq \sum_{v \in V} c_v \tag{2.20}$$

On peut donc conclure avec la série d'inégalités suivante :

$$w^* = \sum_{e \in \mathcal{D}^*} w(e) \tag{2.21}$$

$$\geq \sum_{e \in \mathcal{D}^*} \frac{1}{H(d^*)} \sum_{v \in e} c_v \tag{2.22}$$

$$= \frac{1}{H(d^*)} \sum_{e \in \mathcal{D}^*} \sum_{v \in e} c_v \tag{2.23}$$

$$\geq \frac{1}{H(d^*)} \sum_{v \in V} c_v \tag{2.24}$$

$$= \frac{1}{H(d^*)} \sum_{e \in \mathcal{D}} w(e) \tag{2.25}$$

□

Corollaire 2.10 *On dira que notre algorithme trouve une solution avec un facteur d'approximation en* $O(\ln(d^*))$.
De façon plus générale, comme $n \geq d^*$, *on peut dire que notre algorithme trouve une solution avec un facteur d'approximation en* $O(\ln(n))$.

2.5 Inapproximabilité

Lund et Yannakakis prouvent en 1994 qu'il est impossible de trouver un algorithme en temps polynomial avec un rapport $(\frac{1}{2} - \epsilon)\log_2(n)$ pour le problème de Couverture d'ensemble, quel que soit $\epsilon > 0$. Remarquons que $\frac{1}{2}\log_2(n) \approx 0,72$, ainsi le facteur d'approximation est fort en dessous de $\ln(n)$. Ce résultat ne nous garantit donc absolument pas l'optimalité de l'algorithme glouton parmi les algorithmes polynomiaux (en supposant toujours que $\mathcal{P}\neq\mathcal{NP}$). Pourtant nous aimerions bien que l'algorithme glouton soit le plus efficace. La motivation vient de la construction que fait Uriel Feige dans *A threshold of ln n for approximating set cover* [7]. Il construit en effet un système de partitions de l'ensemble des sommets où chacune d'elles consiste en k hyperarêtes aléatoires de taille $\frac{n}{k}$ où k est une constante. Une bonne couverture consiste alors en la sélection des k hyperarêtes d'une même partition, alors qu'une mauvaise couverture par l'algorithme glouton consiste en la sélection de d hyperarêtes de différentes partitions avec une forte probabilité que chacune de ces d hyperarêtes couvre une proportion $\frac{1}{k}$ des sommets non encore sélectionnés. En d'autres termes, on cherche le plus petit d tel que le nombre de sommets non encore couverts $n(1 - \frac{1}{k})^d$ est plus petit que 1, c'est-à-dire d tel que $d\ln(\frac{k}{k-1}) \approx \ln(n)$. Nous avons de plus $\frac{1}{k} < \ln(\frac{k}{k-1}) = \ln(k) - \ln(k-1) < \frac{1}{k-1}$ (présenté plus en détail en (4.61)), cela signifie que $d \approx k\ln(n)$ pour k assez grand. Ainsi, le rapport entre une mauvaise et une bonne solution est à peu près $\ln(n)$. Il est alors naturel de croire qu'il n'existe pas d'algorithme polynomial approximant le problème de Couverture d'ensemble avec un rapport meilleur que $\ln(n)$, sauf si $\mathcal{P}=\mathcal{NP}$.

Dans *The Hardness of Approximating Set Cover* [8], Alexander Wolff présente le problème de Couverture d'ensemble et une preuve d'inapproximabilité par un algorithme polynomial meilleur que l'algorithme glouton, à moins que $\mathcal{P}=\mathcal{NP}$. Il utilise le problème ϵ-Robe3Sat-5. Il s'agit d'un problème semblable au problème SAT : on considère une formule booléenne ϕ où chaque clause contient exactement 3 termes, où chaque variable apparaît exactement dans 5 clauses et où chaque variable ne peut pas apparaître plus d'une fois dans une clause. De plus, soit ϕ est satisfaisable, soit au plus une fraction $1-\epsilon$ de ses clauses sont simultanément satisfaisables. La question qui se pose alors est de savoir si ϕ est satisfaisable. Alexander Wolff affirme qu'il est prouvé qu'il existe $\epsilon \in]0,1[$ pour lequel ϵ-Robe3Sat-5 est $\mathcal{NP}$-difficile à résoudre. La démonstration de l'inapproximabilité du problème de Couverture d'ensemble consiste à montrer une réduction de chaque instance de ϵ-Robe3Sat-5 en une instance du problème de Couverture d'ensemble dont l'approximation est meilleure que $\ln(n)$. Ainsi, tout algorithme polynomial permettant de résoudre le problème de Couverture d'ensemble avec un rapport meilleur que $\ln(n)$ pourra également résoudre ϵ-Robe3Sat-5 ; ce qui prouve l'inexistence d'un tel algorithme, à moins que $\mathcal{P}=\mathcal{NP}$.

Chapitre 3

Problème de Couverture d'ensemble de somme minimum

Un supercalculateur est à disposition de diverses entreprises. Chaque entreprise a besoin d'exécuter un certain nombre de tâches sur le calculateur. Mais bien souvent, des problèmes de partage de ressources ne permettent pas à certaines tâches de s'exécuter en même temps. On aimerait exécuter les tâches de manière groupée et dans un certain ordre, de façon à minimiser le temps moyen nécessaire pour qu'une tâche quelconque soit terminée.

3.1 Présentation du problème

Regardant le travail de Uriel Feige, László Lovász et Prasad Tetali *Approximating Min-Sum Set Cover* [6], nous décidons de retranscrire leurs résultats dans une notation différente, semblable à celle que nous venons d'utiliser pour le problème de Couverture d'ensemble. Tout au long de ce chapitre et selon les résultats, nous ajouterons les détails de certains calculs, ou au contraire, omettrons certaines démonstrations.

Dans le problème du chapitre précédent de Couverture d'ensemble, nous avions un hypergraphe $H(V, E)$, où V est un ensemble de sommets qui pourrait être par exemple un ensemble de tâches quelconques à effectuer, et E un ensemble d'hyperarêtes qui pourraient être les différents groupements possibles de tâches effectuables en même temps, ces hyperarêtes étant pondérées d'un certain poids représentant le coût d'exécution de l'ensemble de ces tâches. Nous souhaitions alors réduire le coup total d'exécution de l'ensemble des tâches, en définissant notre fonction objectif

$$F_{obj} : C_H \to \mathbb{R}^+ : \mathcal{C} \mapsto \sum_{e \in \mathcal{C}} w(e) \tag{3.1}$$

où C_H était l'ensemble des couvertures sur H. Dans le cas particulier où chaque hyperarête se voyait attribuer un poids unitaire, nous demandions en fait à minimiser le temps total d'exécution des tâches.

Dans le problème de *Couverture d'ensemble de somme minimum* (*Min Sum Set Cover* dans la littérature en anglais), nous allons modifier notre fonction objectif. Notre but est toujours de couvrir tout l'ensemble V à l'aide des hyperarêtes de E, mais nous baserons notre choix sur l'idée que plus un sommet est couvert tard, plus sa sélection coûtera cher. Cela signifie que cette fois, l'ordre de sélection des hyperarêtes a de l'importance (contrairement au cas précédent de Couverture d'ensemble). En l'occurrence, la sélection d'un sommet à la n-ème étape par une hyperarête de m sommets coûtera n/m. Ainsi pour reprendre notre exemple des tâches, plus longtemps une tâche reste en attente, plus elle coûtera. Formalisons cela :

Considérons un hypergraphe $H(V, E)$. On appelle *organisation linéaire* une bijection $f : E \to \{1, 2, \ldots, |E|\} : e \mapsto f(e)$ représentant un ordre dans lequel nous choisirons les hyperarêtes pour couvrir les sommets. Soit v un sommet quelconque, on pose $E_v = \{e \in E | v \in e\}$ l'ensemble des hyperarêtes de E contenant le sommet v et on définit $f(v) = \min_{e \in E_v} f(e)$, c'est-à-dire l'indice de la première hyperarête couvrant le sommet v. Notre fonction objectif est définie par

$$F_{obj} : \mathcal{F}_E \to \mathbb{R}^+ : f \mapsto \sum_{v \in V} f(v) \tag{3.2}$$

où $\mathcal{F}_E$ est l'ensemble des organisations linéaires d'hyperarêtes de E. Nous tenterons alors de déterminer une organisation linéaire f qui minimise la fonction objectif donnée ci-dessus.

Remarque 3.1 *Dans* Approximating Min-Sum Set Cover *[6], le problème identique est traité dans sa version duale. C'est-à-dire que pour un hypergraphe $H(V, E)$ donné, ils cherchent à sélectionner des sommets $\{v_1, \ldots, v_k\}$ de V pour que chaque hyperarête de E ait au moins un sommet sélectionné, et qui minimise $\sum_{e \in E} t_e$ où t_e est l'indice de la première étape durant laquelle l'hyperarête e voit l'un de ses sommets sélectionné par l'algorithme.*

Considérons notre problème de décision « Étant donné un hypergraphe $H(V, E)$ et un entier k, existe-t-il une organisation linéaire f telle que $\sum_{v \in V} f(v) \leq k$? ». Tester une solution se fait bien en temps polynomial, le problème de Couverture d'ensemble de somme minimum est donc un problème de $\mathcal{NP}$. Nous verrons plus tard qu'il est également $\mathcal{NP}$-complet.

3.2 Algorithme Glouton

L'algorithme glouton tentera, de façon similaire à la situation précédente, d'apporter une solution approchée du problème d'optimisation suivant :
« Étant donné un hypergraphe $H(V, E)$, on cherche une séquence $e_1, e_2, \ldots, e_k$ dans E (c'est-à-dire une organisation linéaire) telle que $\bigcup_{i=1}^{k} e_i = V$ qui minimise notre fonction objectif $\sum_{i=1}^{k} f(e_i)$ ».
L'algorithme glouton s'implémente toujours de la même façon :

1. un *entier* $j \leftarrow 1$
2. un *entier* $s \leftarrow 0$
3. un *ensemble de sommets* $R \leftarrow V$
4. un *ensemble indicé d'hyperarêtes* $D \leftarrow \emptyset$
5. tant que $R \neq \emptyset$ faire :
   (a) sélectionner $e \in E \backslash D$ qui maximise $|R \cap e|$
   (b) ajouter $j \cdot |R \cap e|$ à s
   (c) ajouter e dans D
   (d) supprimer de R les éléments de e qui s'y trouvent
   (e) ajouter 1 à j
6. retourner D

En d'autres termes, à chaque étape l'algorithme choisit l'hyperarête contenant le plus de sommets non sélectionnés, et ce jusqu'à ce que tous les sommets soient sélectionnés.

Remarque 3.2 *La variable s n'est là qu'à titre indicatif, et sa valeur à la fin de l'algorithme sera la somme* $\sum_{i=1}^{k} f(e_i)$.

Notre algorithme glouton est essentiellement le même que celui approximant le problème de Couverture d'ensemble, et sa complexité est toujours polynomiale en $O(n^2)$.

3.3 Facteur d'approximation

Nous allons montrer que cette fois, l'algorithme glouton est une 4-approximation. Notre fonction objectif rend dans ce cas-ci l'algorithme glouton beaucoup plus intéressant que pour le problème précédent. En effet, dans le problème de *Couverture d'ensemble*, nous avions montré que la valeur d'une solution trouvée par l'algorithme glouton était bornée par une constante dépendant de la taille du problème. Ici nous montrerons que notre constante vaut au plus 4, quelle que soit la taille du problème.

Appelons OPT la plus petite valeur que peut prendre la fonction objectif, et appelons $ALGO$ la valeur retournée par l'algorithme glouton pour une instance particulière du problème. On pose C_e l'ensemble des sommets qui seront couverts lors de l'étape où l'algorithme glouton sélectionne l'hyperarête e, et on pose R_e l'ensemble des sommets non encore couverts au début de cette étape. Par définition de $f(v)$, l'algorithme glouton nous renvoie

$$ALGO = \sum_{v \in V} f(v) = \sum_{e \in E} f(e)|C_e| \tag{3.3}$$

Lemme 3.3 *Nous affirmons que*

$$ALGO = \sum_{e \in E} |R_e| \tag{3.4}$$

Démonstration 3.3 On pose $\mathcal{D}_e$ l'ensemble des hyperarêtes de E déjà sélectionnées au début de l'étape où e sera sélectionnée, c'est-à-dire $\mathcal{D}_e = \{e' \in E : f(e') < f(e)\}$. On a

$$R_e = V \setminus \bigcup_{e^* \in \mathcal{D}_e} C_{e^*} \tag{3.5}$$

où les C_{e^*} sont disjoints dans V, et donc

$$|R_e| = \left| V \setminus \bigcup_{e^* \in \mathcal{D}_e} C_{e^*} \right| = |V| - \sum_{e^* \in \mathcal{D}_e} |C_{e^*}| \tag{3.6}$$

Remarquons aussi que dans une somme

$$\sum_{e \in E} \sum_{e^* \in \mathcal{D}_e} k(e^*) \tag{3.7}$$

où k est une fonction arbitraire, l'indice e^* est utilisé très exactement $|E| - f(e^*)$ fois, et par conséquent cette somme peut aussi s'écrire

$$\sum_{e \in E} (|E| - f(e))k(e) \tag{3.8}$$

Cela se voit facilement en posant $E = \{e_1, e_2, \ldots, e_{|E|}\}$ où les e_k sont tels que $k = f(e_k)$, c'est-à-dire dans l'ordre de sélection par l'algorithme, et en regardant la figure 3.1 symbolisant les deux sommations possibles.

Ces deux arguments conduisent à la suite d'égalités suivante :

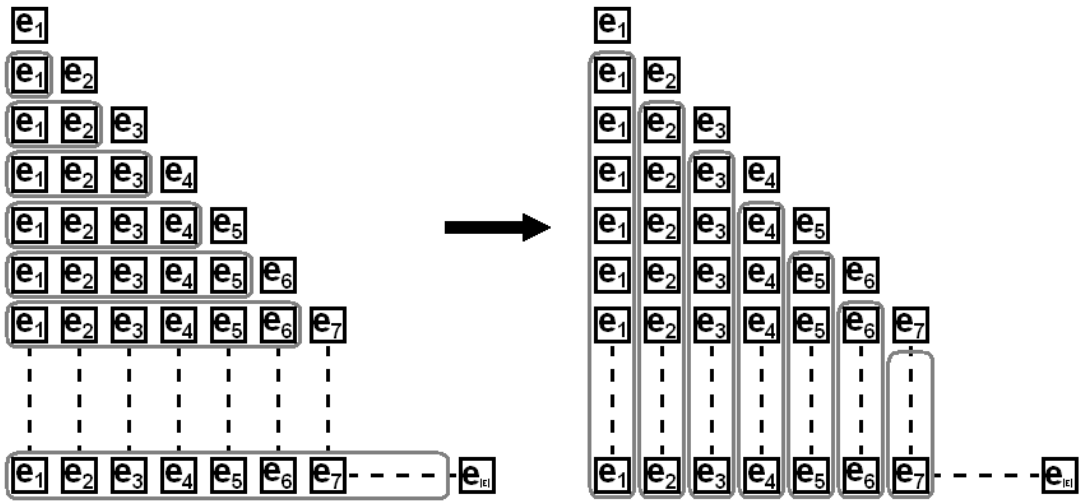

FIG. 3.1 – Représentation de la transformation de sommation

$$\sum_{e\in E}|R_e| = \sum_{e\in E}|V| - \sum_{e\in E}\sum_{e^*\in\mathcal{D}_e}|C_{e^*}| \tag{3.9}$$
$$= \sum_{e\in E}|V| - \sum_{e\in E}(|E| - f(e))|C_e| \tag{3.10}$$
$$= \sum_{e\in E}|V| - \sum_{e\in E}|C_e|\cdot|E| + \sum_{e\in E}f(e)|C_e| \tag{3.11}$$
$$= |E|\cdot|V| - |V|\cdot|E| + \sum_{e\in E}f(e)|C_e| \tag{3.12}$$
$$= \sum_{e\in E}f(e)|C_e| \tag{3.13}$$
$$= ALGO \tag{3.14}$$

□

On définit $P_e = \frac{|R_e|}{|C_e|}$, et pour chaque sommet $v \in R_e \cap e$, on définit son *prix* $p_v = P_e$, et on pose $PR = \sum_{v\in V} p_v$. On a alors

$$PR = \sum_{v\in V} p_v \tag{3.15}$$
$$= \sum_{e\in E} P_e|C_e| \tag{3.16}$$
$$= \sum_{e\in E}|R_e| \tag{3.17}$$
$$= ALGO \tag{3.18}$$

Théorème 3.4 *La valeur renvoyée par l'algorithme glouton est majorée selon la relation*

$$OPT \geq \frac{PR}{4} \tag{3.19}$$

Démonstration 3.4 Considérons l'histogramme H_1 présenté dans la figure 3.2 correspondant à la solution optimale. On y trouve $|V|$ colonnes, une pour chaque $v \in V$, chacune de largeur 1 et ordonnées de gauche à droite dans l'ordre de recouvrement des sommets de V par un algorithme optimal. La hauteur de la colonne associée au sommet v représente l'instant auquel ce sommet est couvert. L'aire de notre histogramme H_1 est définie par $A(H_1)$ et vaut exactement OPT.
Considérons à présent l'histogramme H_2 dans la figure 3.2 correspondant cette fois à la solution fournie par l'algorithme glouton. Nous lui attribuons également $|V|$ colonnes de largeur 1, une pour chaque sommet, ordonnées de gauche à droite selon l'ordre de recouvrement par l'algorithme glouton. Mais cette fois la hauteur de la colonne sera le prix p_v de recouvrement du sommet associé. Notre second histogramme H_2 a une aire $A(H_2)$ valant exactement PR.

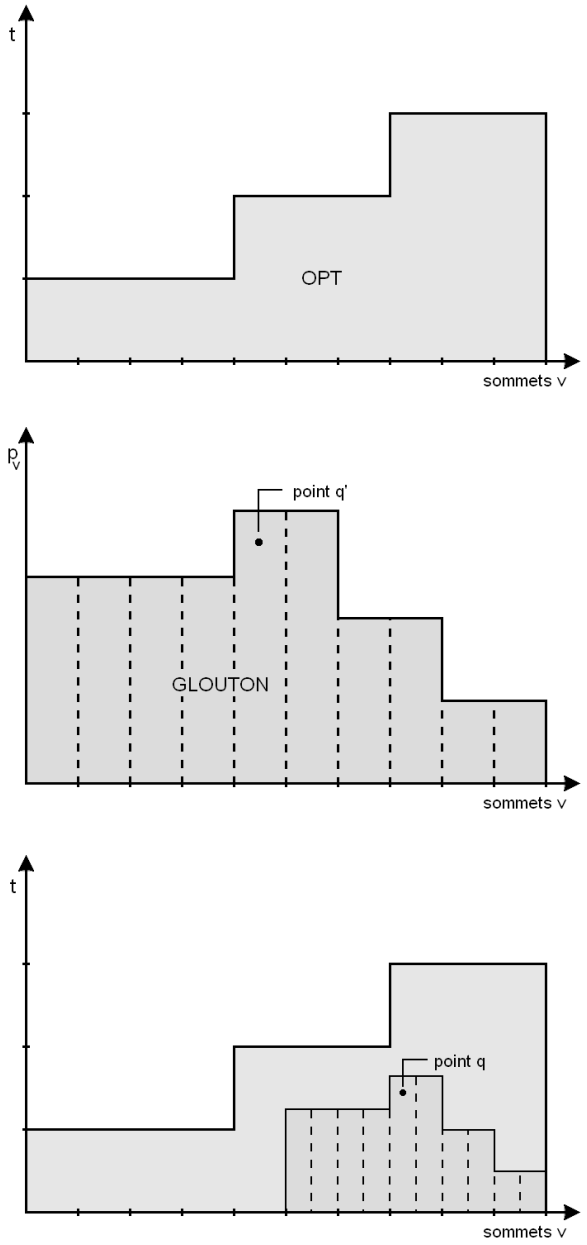

FIG. 3.2 – Histogrammes H_1, H_2 et rapport des deux.

On voudrait bien montrer que $A(H_1) \geq \frac{A(H_2)}{4}$. On crée pour ce faire un troisième histogramme H_2^*, qui est une réduction à l'échelle 1/2 de l'histogramme H_2. C'est donc un histogramme de largeur totale $\frac{|V|}{2}$ et dont chaque colonne se voit attribuer une hauteur $\frac{p_v}{2}$. Son aire vaut donc $A(H_2^*) = \frac{A(H_2)}{4}$.
On aligne notre nouvel histogramme H_2^* sur l'histogramme H_1, en le plaçant à l'extrême droite, comme présenté dans la figure 3.2. Il occupe donc les colonnes $\frac{|V|}{2}+1$ à $|V|$. Pour simplifier les notations, on suppose que $|V|$ est pair. Nous allons montrer que l'aire de notre histogramme H_2^* ne dépasse pas celle de H_1, ce qui prouvera le lemme. Pour ce faire, il suffit de montrer que tout point de H_2^* est également dans H_1.
Considérons un point arbitraire q' dans H_2, dont la colonne est c_v, associée au sommet v. Soit e l'hyperarête sélectionnée par l'algorithme glouton pour recouvrir v. Alors la hauteur de q' est tout au plus $p_v = \frac{|R_e|}{|C_e|}$, et la distance séparant q' de la droite du diagramme est inférieure ou égale à $|R_e|$, puisque chaque étape couvre au moins un sommet. L'application naturelle qui envoie H_2 sur H_2^* nous fournit un point q dans H_2^*. Montrons que ce point est dans H_1.
La hauteur de q, que l'on appellera h est inférieure ou égale à $\frac{|R_e|}{2|C_e|}$ et la distance r séparant q de la droite de l'histogramme vaut au plus $\frac{|R_e|}{2}$. Pour montrer que q est dans H_1, il suffit de prouver qu'au moment h (que l'on arrondit à l'entier inférieur $\lfloor h \rfloor$, car il ne se passe rien entre $\lfloor h \rfloor$ et h), au moins r sommets (que l'on arrondit à l'entier supérieur $\lceil r \rceil$, car il s'agit d'une valeur entière) sont encore non couverts par l'algorithme optimal.
Considérons un instant uniquement l'ensemble R_e. Aucune hyperarête ne peut couvrir plus de $|C_e|$ sommets de R_e. Ainsi, après $\lfloor h \rfloor$ étapes, l'algorithme optimal a pu couvrir au plus $\lfloor h \rfloor \cdot |C_e|$ sommets dans R_e (avec $\lfloor h \rfloor |C_e| \leq \left\lfloor \frac{|R_e|}{2} \right\rfloor$, car $h \leq \frac{|R_e|}{2|C_e|}$, donc $h|C_e| \leq \frac{|R_e|}{2}$, et ainsi $\lfloor h \rfloor |C_e| \leq \lfloor h|C_e| \rfloor \leq \left\lfloor \frac{|R_e|}{2} \right\rfloor$), laissant au moins $\left\lceil \frac{|R_e|}{2} \right\rceil \geq \lceil r \rceil$ sommets de R_e non couverts. □

Corollaire 3.5 *On en déduit immédiatement le résultat d'approximation concernant notre algorithme glouton*

$$ALGO \leq 4 \cdot OPT \tag{3.20}$$

Nous allons maintenant montrer que cette borne peut être atteinte asymptotiquement lorsque la taille du problème augmente. Mais avant cela, quelques résultats seront nécessaires.

Lemme 3.6

$$\sum_{i=1}^{\infty} \frac{1}{i^2} = \frac{\pi^2}{6} \tag{3.21}$$

La démonstration de ce lemme dûe à Euler ne sera pas présentée ici.

Lemme 3.7 *Soit n un entier strictement positif. Alors*

$$\frac{1}{2n} + \sum_{i=1}^{n} \frac{1}{i^2} < \frac{\pi^2}{6} \tag{3.22}$$

Démonstration 3.7

$$\frac{\pi^2}{6} = \sum_{i=1}^{n} \frac{1}{i^2} + \sum_{i=n+1}^{\infty} \frac{1}{i^2} \tag{3.23}$$

$$> \sum_{i=1}^{n} \frac{1}{i^2} + \int_{n+1}^{\infty} \frac{dx}{x^2} \tag{3.24}$$

$$= \sum_{i=1}^{n} \frac{1}{i^2} + \frac{1}{n+1} \tag{3.25}$$

$$\geq \sum_{i=1}^{n} \frac{1}{i^2} + \frac{1}{2n} \tag{3.26}$$

□

Lemme 3.8 *Soient x et n des entiers strictement plus grands que 0, alors*

$$x + \sum_{i=1}^{n} \frac{x}{i^2} = \frac{x(n+1)}{n^2} + \sum_{i=1}^{n-1} \left(\frac{x}{i^2} + \frac{x}{i(i+1)} \right) \tag{3.27}$$

Démonstration 3.8 On obtient par récurrence que

$$\sum_{i=1}^{n} \frac{1}{i(i+1)} = \frac{n}{n+1} \tag{3.28}$$

En effet, c'est évident pour $n = 1$, et en supposant le résultat correct pour n, on obtient

$$\sum_{i=1}^{n+1} \frac{1}{i(i+1)} = \sum_{i=1}^{n} \frac{1}{i(i+1)} + \frac{1}{(n+1)(n+2)} \tag{3.29}$$

$$= \frac{n}{n+1} + \frac{1}{(n+1)(n+2)} \tag{3.30}$$

$$= \frac{n+1}{n+2} \tag{3.31}$$

Cela prouve la récurrence. On peut en déduire que

$$\sum_{i=1}^{n-1}\left(\frac{1}{i^2}+\frac{1}{i(i+1)}\right)=\frac{n-1}{n}+\sum_{i=1}^{n-1}\frac{1}{i^2} \tag{3.32}$$

$$\sum_{i=1}^{n-1}\left(\frac{1}{i^2}+\frac{1}{i(i+1)}\right)+\frac{1}{n}=1+\sum_{i=1}^{n-1}\frac{1}{i^2} \tag{3.33}$$

$$\sum_{i=1}^{n-1}\left(\frac{1}{i^2}+\frac{1}{i(i+1)}\right)+\frac{n}{n^2}+\frac{1}{n^2}=1+\sum_{i=1}^{n}\frac{1}{i^2} \tag{3.34}$$

On obtient le lemme en multipliant l'égalité par x. □

Nous avons maintenant les outils pour démontrer le théorème suivant :

Théorème 3.9 *Pour tout ϵ strictement positif, on peut trouver un hypergraphe pour lequel l'algorithme glouton fournit une solution dont la valeur s est strictement supérieure à $(4-\epsilon)$ fois la valeur s^* d'une solution optimale.*

Démonstration 3.9 Considérons un entier $n > 1$, et posons x un multiple de $1^2, 2^2, \ldots, n^2$. On peut alors définir deux séquences d'entiers

$$< S_i > =< \frac{x}{1\cdot 1}, \frac{x}{1\cdot 2}, \frac{x}{2\cdot 2}, \frac{x}{2\cdot 3}, \frac{x}{3\cdot 3}, \ldots, \frac{x}{(n-1)\cdot n}, \underbrace{\frac{x}{n^2}, \ldots, \frac{x}{n^2}}_{n+1 \text{ termes}} > \tag{3.35}$$

$$< T_i > =< x, \frac{x}{1^2}, \frac{x}{2^2}, \frac{x}{3^2}, \ldots, \frac{x}{n^2} > \tag{3.36}$$

dont les sommes de leurs termes respectifs (possédant respectivement $3n-1$ et $n+1$ termes) sont égales (par le lemme 3.8) et dont la valeur commune est notée $\phi(n)$.
Nous allons maintenant construire un hypergraphe $H(V,E)$ possédant la propriété souhaitée, à savoir que le problème de recouvrement associé à cet hypergraphe possède une solution au moins $(4-\epsilon)$ fois meilleure que celle fournie par l'algorithme glouton. À noter que cet hypergraphe est en fait l'hypergraphe dual d'un multigraphe biparti utilisé dans la démonstration originale présentée dans [6].
On impose que le nombre de sommets $|V|$ soit $\phi(n)$ et que le nombre d'hyperarêtes soit $|E| = (3n-1)+(n+1) = 4n$. Les sommets seront notés $v_1, v_2, \ldots, v_{\phi(n)}$ et les hyperarêtes, partitionnées en deux ensembles, seront notées $s_1, s_2, \ldots, s_{3n-1}, t_1, t_2, \ldots, t_{n+1}$. On impose de plus que pour tous i, j dans l'intervalle approprié, le degré de l'hyperarête s_i soit S_i, le degré de l'hyperarête t_j soit T_j, et le degré de chacun des sommets de V soit 2. On demande encore que chaque sommet soit contenu dans l'une des hyperarêtes s_i et dans l'une des hyperarêtes t_j, et que les s_i, ainsi que les t_j forment une partition de V. Nous pouvons construire un exemple $H(V,E)$ d'un tel hypergraphe de la façon suivante :
On dispose des $\phi(n)$ sommets $v_1, v_2, \ldots, v_{\phi(n)}$. On va ensuite former progressivement les hyperarêtes.

On définit facilement les hyperarêtes s_i par

$$s_i = \bigcup_{k=1}^{S_i} \{v_{\sum_{j=1}^{i-1} S_j + k}\} \tag{3.37}$$

c'est-à-dire

$$s_1 = \{v_1, \ldots, v_{S_1}\} \tag{3.38}$$
$$s_2 = \{v_{S_1+1}, \ldots, v_{S_1+S_2}\} \tag{3.39}$$
$$s_3 = \{v_{S_1+S_2+1}, \ldots, v_{S_1+S_2+S_3}\} \tag{3.40}$$
$$\vdots$$
$$s_{3n-1} = \{v_{\sum_{j=1}^{3n-2} S_j+1}, \ldots, v_{\sum_{j=1}^{3n-1} S_j}\} \tag{3.41}$$

Pour définir les t_j, on les initialise tous à $\emptyset$, et on pose les entiers $d_1, d_2, \ldots, d_{n+1}$ initialisés par $d_j = T_j$. On démarre le procédé avec le compteur $k = 1$, et à chaque étape, on ajoute le sommet v_k dans l'hyperarête t_j pour laquelle d_j est un maximum de l'ensemble des d_i. S'il y a plusieurs tels t_j, on sélectionne celui pour lequel l'indice est le plus petit. On décrémente ensuite d_j d'une unité, et on incrémente le compteur k d'une unité. On fait cela tant qu'il y a des v_k non encore sélectionnés. Plus précisément, les v_k sont sélectionnés par les t_j dans cet ordre :

$$\underbrace{t_1, t_2, t_1, t_2, \ldots, t_1, t_2}_{2(x-\frac{x}{4}) \text{ termes}}, \underbrace{t_1, t_2, t_3, t_1, t_2, t_3, \ldots, t_1, t_2, t_3}_{3(\frac{x}{4}-\frac{x}{9}) \text{ termes}}, \underbrace{t_1, t_2, t_3, t_4, \ldots, t_1, t_2, t_3, t_4}_{4(\frac{x}{9}-\frac{x}{16}) \text{ termes}},$$
$$\ldots, \underbrace{t_1, \ldots, t_k, \ldots, t_1, \ldots, t_k}_{k(\frac{x}{(k-1)^2}-\frac{x}{k^2}) \text{ termes}}, \ldots, \underbrace{t_1, \ldots, t_{n+1}, \ldots, t_1, \ldots, t_{n+1}}_{(n+1)(\frac{x}{n^2}) \text{ termes}} \tag{3.42}$$

Il nous faut trouver maintenant un algorithme mieux construit que l'algorithme glouton et assez pour réaliser le résultat attendu. Cet algorithme sera celui qui sélectionne les hyperarêtes $t_1, t_2, \ldots, t_{n+1}$ dans cet ordre. La valeur s' qu'il renverra sera

$$s' = \sum_{i=1}^{n+1} i \cdot T_i = x + \sum_{i=1}^{n} (i+1)\frac{x}{i^2} \tag{3.43}$$

Ainsi la valeur s^* renvoyée par un algorithme optimal sera plus petite que s'. Nous avons

$$s^* \leq s' = x\left(1 + \sum_{i=1}^{n} \frac{i+1}{i^2}\right) \tag{3.44}$$
$$= x\left(1 + \sum_{i=1}^{n} \frac{1}{i} + \sum_{i=1}^{n} \frac{1}{i^2}\right) \tag{3.45}$$
$$\leq x\left(H_n + 1 + \frac{\pi^2}{6}\right) \quad \text{par le lemme 3.6} \tag{3.46}$$
$$= x\left(H_n + \frac{\pi^2+6}{6}\right) \tag{3.47}$$

C'est à nous de jouer maintenant pour montrer que l'algorithme glouton peut être suffisamment mauvais. En particulier, nous allons montrer qu'il est possible qu'il sélectionne les hyperarêtes $s_1, s_2, \ldots, s_{3n-1}$ dans cet ordre. Pour que cela soit possible, il faut, à chaque fois que l'algorithme s'apprête à sélectionner s_j, que chacune des hyperarêtes $t_1, t_2, \ldots, t_{n+1}$ contienne au plus S_j sommets. Pour nous en assurer nous allons surveiller les choix de l'algorithme glouton étape par étape à l'aide d'un vecteur compteur $< d_1, d_2, \ldots, d_{n+1} >$ dont chaque élément d_j compte le nombre de sommets non encore couverts dans l'hyperarête t_j. Ce vecteur compteur est ainsi initialisé à $< x, \frac{x}{1^2}, \frac{x}{2^2}, \frac{x}{3^2}, \ldots, \frac{x}{n^2} >$. On crée de plus une variable Q comptant le nombre total de sommets couverts par l'algorithme glouton. Q est initialisé à 0.
À chaque fois que l'algorithme glouton sélectionne une hyperarête s_j, un certain nombre de sommets contenus dans les $t_1, t_2, \ldots, t_{n+1}$ sont couverts, et on obtient donc un nouveau vecteur compteur dont certaines valeurs ont diminué. Ainsi, lors de la sélection de la première hyperarête par l'algorithme glouton (qui est justifiée car $S_1 = x$ est supérieur ou égal à chacun des éléments du vecteur compteur), les x sommets de s_1 sont répartis équitablement entre les hyperarêtes t_1 et t_2. En effet, x est pair et observant (3.42) on s'en assure car $x \leq 2(x - \frac{x}{4})$. La première étape ôte donc $\frac{x}{2}$ à d_1 et d_2, ce qui transforme le vecteur compteur en

$$< \frac{x}{2}, \frac{x}{2}, \frac{x}{2^2}, \frac{x}{3^2}, \ldots, \frac{x}{n^2} > \tag{3.48}$$

et d'autre part Q augmente à x. La sélection de la première hyperarête par l'algorithme glouton est ainsi terminée. À ce stade, la sélection de la seconde hyperarête s_2 par l'algorithme glouton est toujours justifiée, car $S_2 = \frac{x}{2}$ est supérieur ou égal à tous les éléments du vecteur compteur. Les $\frac{x}{2}$ sommets de s_2 sont répartis équitablement entre les hyperarêtes t_1 et t_2. En effet, x est pair et observant (3.42) et se rappelant que l'étape précédente a couvert x sommets, on s'en assure car $x + \frac{x}{2} = 2(x - \frac{x}{4})$. La seconde étape ôte donc $\frac{x}{4}$ à d_1 et à d_2, ce qui transforme le vecteur compteur en

$$< \frac{x}{4}, \frac{x}{4}, \frac{x}{2^2}, \frac{x}{3^2}, \ldots, \frac{x}{n^2} > \tag{3.49}$$

et la valeur $\frac{x}{2}$ s'ajoute à Q, ce qui amène à $Q = 2(x - \frac{x}{4})$.
Nous allons regarder les étapes suivantes par récurrence et par paires, c'est-à-dire que nous allons observer les étapes de sélection de s_{2k-1} et s_{2k} par l'algorithme glouton. La récurrence suppose que le vecteur compteur vaut, au commencement de ces deux étapes, exactement

$$< \underbrace{\frac{x}{k^2}, \frac{x}{k^2}, \ldots, \frac{x}{k^2}}_{k+1 \text{ termes}}, \frac{x}{(k+1)^2}, \frac{x}{(k+2)^2}, \ldots, \frac{x}{n^2} > \tag{3.50}$$

Remarquons que c'était le cas au début de l'étape sélectionnant s_1 (en posant $k = 1$) et que c'est toujours valable à la fin de l'étape ayant sélectionné s_2 (en posant $k = 2$). La récurrence suppose également qu'au début de cette double sélection Q soit exactement égal à

$$Q = 2\left(\frac{x}{1^2} - \frac{x}{2^2}\right) + 3\left(\frac{x}{2^2} - \frac{x}{3^2}\right) + 4\left(\frac{x}{3^2} - \frac{x}{4^2}\right) + \cdots + k\left(\frac{x}{(k-1)^2} - \frac{x}{k^2}\right) \tag{3.51}$$

De la même façon, on remarque que c'était le cas au début de l'étape sélectionnant s_1 (en posant $k = 1$) et que c'est toujours valable à la fin de l'étape ayant sélectionné s_2 (en posant $k = 2$). Nous allons montrer que la sélection de s_{2k-1} et s_{2k} par l'algorithme glouton est justifiée, et que la récurrence s'étend à l'étape $k + 1$, c'est-à-dire qu'après sélection de s_{2k-1} et s_{2k}, la situation reste la même, mais en remplaçant k par $k + 1$.
La sélection de l'hyperarête s_{2k-1} par l'algorithme glouton est justifiée, car $S_{2k-1} = \frac{x}{k^2}$ est supérieur ou égal à tous les éléments du vecteur compteur. Les $\frac{x}{k^2}$ sommets de s_{2k-1} sont répartis équitablement entre les hyperarêtes $t_1, t_2, \ldots, t_{k+1}$. En effet, $\frac{x}{k^2}$ est un multiple de $k + 1$ et observant (3.42), on s'en assure car $\frac{x}{k^2} \leq (k + 1)(\frac{x}{k^2} - \frac{x}{(k+1)^2})$. Cette étape ôte donc $\frac{x}{k^2(k+1)}$ à chacun des $d_1, d_2, \ldots, d_{k+1}$, ce qui transforme le vecteur compteur en

$$< \underbrace{\frac{x}{k^2} - \frac{x}{k^2(k+1)}, \ldots, \frac{x}{k^2} - \frac{x}{k^2(k+1)}}_{k+1 \text{ termes}}, \frac{x}{(k+1)^2}, \frac{x}{(k+2)^2}, \ldots, \frac{x}{n^2} > \tag{3.52}$$

$$= < \underbrace{\frac{x}{k(k+1)}, \ldots, \frac{x}{k(k+1)}}_{k+1 \text{ termes}}, \frac{x}{(k+1)^2}, \frac{x}{(k+2)^2}, \ldots, \frac{x}{n^2} > \tag{3.53}$$

et Q est incrémenté de $\frac{x}{k^2}$. Ceci termine l'étape de sélection de s_{2k-1} par l'algorithme glouton. De nouveau, la sélection de s_{2k} par l'algorithme glouton est justifiée car $S_{2k} = \frac{x}{k(k+1)}$ est supérieur ou égal à tous les éléments du vecteur compteur. Les $\frac{x}{k(k+1)}$ sommets de s_{2k} sont répartis équitablement entre les hyperarêtes $t_1, t_2, \ldots, t_{k+1}$. En effet, $\frac{x}{k(k+1)}$ est un multiple de $k + 1$ et observant (3.42) et se rappelant que l'étape précédente a couvert $\frac{x}{k^2}$ sommets, on s'en assure car $\frac{x}{k^2} + \frac{x}{k(k+1)} = (k+1)(\frac{x}{k^2} - \frac{x}{(k+1)^2})$. Cette seconde étape de la récurrence ôte donc $\frac{x}{k(k+1)^2}$ à chacun des $d_1, d_2, \ldots, d_{k+1}$, ce qui transforme le vecteur compteur en

$$< \underbrace{\frac{x}{k(k+1)} - \frac{x}{k(k+1)^2}, \ldots, \frac{x}{k(k+1)} - \frac{x}{k(k+1)^2}}_{k+1 \text{ termes}}, \frac{x}{(k+1)^2}, \ldots, \frac{x}{n^2} > \tag{3.54}$$

$$= < \underbrace{\frac{x}{(k+1)^2}, \ldots, \frac{x}{(k+1)^2}}_{k+2 \text{ termes}}, \frac{x}{(k+2)^2}, \frac{x}{(k+3)^2}, \ldots, \frac{x}{n^2} > \tag{3.55}$$

et la valeur $\frac{x}{k(k+1)}$ s'ajoute à Q, ce qui amène à

$$Q = 2\left(\frac{x}{1^2} - \frac{x}{2^2}\right) + 3\left(\frac{x}{2^2} - \frac{x}{3^2}\right) + \cdots + k\left(\frac{x}{(k-1)^2} - \frac{x}{k^2}\right) + (k+1)\left(\frac{x}{k^2} - \frac{x}{(k+1)^2}\right) \tag{3.56}$$

La seconde étape de la récurrence est ainsi terminée, et nous avons bien la forme en $k + 1$ souhaitée, aussi bien pour le vecteur compteur que pour le nombre Q de sommets couverts. Notre récurrence est donc démontrée, c'est-à-dire que l'on s'est assuré qu'il est possible que l'algorithme glouton sélectionne les $s_1, s_2, \ldots, s_{3n-1}$ pour l'instance que l'on

s'est fixée.
De cette façon, la valeur de test s renvoyée par l'algorithme glouton vaut

$$s = \sum_{i=1}^{3n-1} i \cdot S_i \tag{3.57}$$

$$= x\left(\sum_{i=1}^{n-1} \frac{2i-1}{i^2} + \sum_{i=1}^{n-1} \frac{2i}{i(i+1)} + \sum_{i=1}^{n+1} \frac{2n-2+i}{n^2}\right) \tag{3.58}$$

$$= x\left(2\sum_{i=1}^{n-1} \frac{1}{i} - \sum_{i=1}^{n-1} \frac{1}{i^2} + 2\sum_{i=1}^{n-1} \frac{1}{i+1} + (n+1)\frac{2n-2}{n^2} + \frac{1}{n^2}\frac{(n+1)(n+2)}{2}\right) \tag{3.59}$$

$$= x\left(2\sum_{i=1}^{n} \frac{1}{i} - \frac{2}{n} - \sum_{i=1}^{n-1} \frac{1}{i^2} + 2\sum_{i=1}^{n} \frac{1}{i} - 2 + \frac{5}{2} + \frac{3}{2n} - \frac{1}{n^2}\right) \tag{3.60}$$

$$= x\left(4\sum_{i=1}^{n} \frac{1}{i} + \frac{1}{2} - \sum_{i=1}^{n} \frac{1}{i^2} - \frac{1}{2n}\right) \tag{3.61}$$

$$> x\left(4\sum_{i=1}^{n} \frac{1}{i} - \frac{\pi^2}{6} + \frac{1}{2}\right) \quad \text{par le lemme 3.7} \tag{3.62}$$

$$= x\left(4H_n - \frac{\pi^2-3}{6}\right) \tag{3.63}$$

Et ainsi, nous avons le rapport

$$\frac{s}{s^*} > \frac{x(4H_n - \frac{\pi^2-3}{6})}{x(H_n - \frac{\pi^2+6}{6})} \tag{3.64}$$

$$= \frac{4H_n - \frac{\pi^2-3}{6}}{H_n + \frac{\pi^2+6}{6}} \tag{3.65}$$

$$= \frac{4(H_n + \frac{\pi^2+6}{6}) - \frac{5\pi^2+21}{6}}{H_n + \frac{\pi^2+6}{6}} \tag{3.66}$$

$$= 4 - \frac{\frac{5\pi^2+21}{6}}{H_n + \frac{\pi^2+6}{6}} \tag{3.67}$$

H_n étant une fonction de n strictement croissante et non bornée, on peut toujours trouver un n assez grand pour que $\frac{\frac{5\pi^2+21}{6}}{H_n+\frac{\pi^2+6}{6}}$ soit plus petit que ϵ. Ceci termine de prouver le théorème. □

3.4 Hypergraphes uniformes réguliers

Rappelons qu'un hypergraphe régulier et un hypergraphe uniforme sont respectivement un hypergraphe dont tous les sommets sont de même degré et un hypergraphe dont toutes les hyperarêtes sont de même degré. Nous allons montrer que dans un hypergraphe uniforme régulier, l'algorithme glouton approxime la solution optimale avec un rapport inférieur ou égal à $\frac{2d}{d+1}$, où d est le degré de chaque sommet de l'hypergraphe.

Lemme 3.10 *Considérons le problème d'urnes suivant :*
On a à disposition m boules dont d sont blanches. On tire aléatoirement et de façon homogène des boules sans les replacer, jusqu'à tirer une boule blanche. On définit la variable aléatoire $X_{m,d}$ du nombre de tirages nécessaires à l'obtention d'une boule blanche. On affirme que l'espérance de $X_{m,d}$ vaut $E[X_{m,d}] = \frac{m+1}{d+1}$.

Démonstration 3.10 Pour que la première boule blanche soit tirée exactement au k-ème tirage, il faut tirer $k-1$ boules non blanches suivies d'une blanche. La probabilité d'un tel tirage est

$$P[X_{m,d} = k] = \frac{m-d}{m} \cdot \frac{m-d-1}{m-1} \cdot \dots \cdot \frac{m-d-(k-2)}{m-(k-2)} \cdot \frac{d}{m-(k-1)} \tag{3.68}$$

$$= \frac{(m-d)!}{(m-d+1-k)!} \cdot \frac{(m+1-k)!}{m!} \cdot \frac{d}{m+1-k} \tag{3.69}$$

$$= d \cdot \frac{(m-d)!}{m!} \cdot \frac{(m-k)!}{(m-d+1-k)!} \tag{3.70}$$

Remarquons que la variable $X_{m,d}$ peut prendre les valeurs 1 à $m-d+1$. Ainsi

$$E[X_{m,d}] = \sum_{k=1}^{m-d+1} k \cdot d \cdot \frac{(m-d)!}{m!} \cdot \frac{(m-k)!}{(m-d+1-k)!} \tag{3.71}$$

$$= d \cdot \frac{(m-d)!}{m!} \sum_{k=1}^{m-d+1} k \cdot \frac{(m-k)!}{(m-d+1-k)!} \tag{3.72}$$

$$= d \cdot \frac{(m-d)!}{m!} \sum_{k=1}^{m-d+1} k \cdot (d-1)! \cdot \binom{m-k}{d-1} \tag{3.73}$$

$$= \frac{d! \cdot (m-d)!}{m!} \sum_{k=1}^{m-d+1} k \cdot \binom{m-k}{d-1} \tag{3.74}$$

$$= \frac{1}{\binom{m}{d}} \sum_{k=1}^{m-d+1} k \cdot \binom{m-k}{d-1} \tag{3.75}$$

$$= \frac{1}{\binom{m}{d}} \sum_{k=0}^{m-d} (m-d+1-k) \cdot \binom{d-1+k}{d-1} \tag{3.76}$$

$$= \frac{1}{\binom{m}{d}} \sum_{k=0}^{m-d} \sum_{j=0}^{k} \binom{d-1+j}{d-1} \tag{3.77}$$

$$= \frac{1}{\binom{m}{d}} \sum_{k=0}^{m-d} \binom{d+k}{d} \tag{3.78}$$

$$= \frac{\binom{m+1}{d+1}}{\binom{m}{d}} \tag{3.79}$$

$$= \frac{m+1}{d+1} \tag{3.80}$$

□

Théorème 3.11 *Pour tout hypergraphe r-uniforme d-régulier, le rapport d'approximation de l'algorithme glouton est inférieur ou égal à* $\frac{2d}{d+1}$.

Démonstration 3.11 Soit $H(V,E)$ un tel hypergraphe. On pose $n = |V|$ et $m = |E|$. La relation $rm = dn$ est immédiate. Nous allons montrer que la valeur associée à une solution optimale est au moins $n\frac{m+d}{2d}$. Nous montrerons ensuite que l'algorithme glouton renvoie une solution dont la valeur est inférieure ou égale à $n\frac{m+1}{d+1}$. Ainsi le rapport d'approximation sera inférieur ou égal à

$$\frac{n\frac{m+1}{d+1}}{n\frac{m+d}{2d}} = \frac{2d}{d+1} - \frac{d-1}{m+d} \tag{3.81}$$

Dans le cas $d = 0$ et $d = 1$, l'algorithme glouton fournit une solution optimale, on ne s'en préoccupe donc pas et on suppose $d \geq 2$, ce qui nous fournit le résultat. Ceci termine de montrer que le rapport est inférieur ou égal à $\frac{2d}{d+1}$. Remarquons que pour $d \geq 2$ nous avons l'inégalité stricte.
Montrons donc que la valeur associée à une solution optimale est au moins $n\frac{m+d}{2d}$.
Quel que soit l'algorithme, à chaque étape au plus r sommets seront couverts ; et à ce rythme il faut au moins $\lceil \frac{n}{r} \rceil = \lceil \frac{m}{d} \rceil$ étapes pour couvrir l'ensemble des sommets de V. Ne pouvant couvrir plus de r sommets à chaque étape, et ayant au moins $\lceil \frac{m}{d} \rceil$ étapes, la valeur associée à une solution optimale sera

$$s^* \geq \sum_{k=1}^{\lceil \frac{m}{d} \rceil} kr \tag{3.82}$$

$$= r\frac{\lceil \frac{m}{d} \rceil (\lceil \frac{m}{d} \rceil + 1)}{2} \tag{3.83}$$

$$\geq r\frac{\frac{m}{d}(\frac{m}{d} + 1)}{2} \tag{3.84}$$

$$= n\frac{m+d}{2d} \tag{3.85}$$

Il nous reste à montrer que la valeur associée à la solution de l'algorithme glouton est inférieure ou égale à $n\frac{m+1}{d+1}$. Pour cela, considérons un sommet quelconque v de V, et tirons aléatoirement des hyperarêtes de E jusqu'à obtenir l'une de celles qui contiennent le sommet v. On est ramené au problème d'urnes traité dans le lemme 3.10 où chaque boule serait une hyperarête, et blanche si elle contient le sommet v. Le nombre moyen d'hyperarêtes à sélectionner aléatoirement et de façon homogène pour couvrir le sommet v est dès lors $\frac{m+1}{d+1}$. En cours de sélection des hyperarêtes, considérons un sommet v non encore sélectionné. L'algorithme glouton choisissant les hyperarêtes possédant le plus de sommets non encore couverts couvrira en moyenne v plus rapidement que l'algorithme aléatoire choisissant de façon homogène une hyperarête non encore sélectionnée. En effet, si $< G_1, G_2, \ldots, G_k >$ est la suite des hyperarêtes sélectionnées par l'algorithme glouton jusqu'à la fin de l'algorithme, alors $< g_1, g_2, \ldots, g_k >$ où $g_i = \frac{|G_i|}{|G_1|+\cdots+|G_k|}$ est une distribution de probabilité décroissante. De même, si $< A_1, A_2, \ldots, A_k >$ est la suite des hyperarêtes sélectionnées par l'algorithme aléatoire, alors $< a_1, a_2, \ldots, a_k >$ où $a_i = \frac{|A_i|}{|A_1|+\cdots+|A_k|}$ est aussi une distribution de probabilité non nécessairement décroissante. La première étant décroissante et l'autre non, on voit immédiatement que l'instant $t = \min\{j | v \in G_1 \cup \cdots \cup G_j\}$ est inférieur ou égal à l'instant $t' = \min\{j | v \in A_1 \cup \cdots \cup A_j\}$. En moyenne, l'algorithme glouton couvre donc plus de sommets à chaque étape que l'algorithme aléatoire. Le nombre moyen d'étapes nécessaires pour obtenir une couverture est donc plus petit pour l'algorithme glouton que pour l'algorithme aléatoire, la valeur pour ce dernier étant égale à $\frac{m+1}{d+1}$. En appliquant cela à tous les sommets de V, on sait que la valeur retournée par l'algorithme glouton sera inférieure ou égale à $n\frac{m+1}{d+1}$. □

3.5 Inapproximabilité

Nous avions vu le problème de Couverture d'ensemble, et en particulier le cas non pondéré, où chaque hyperarête est de poids unitaire, qui cherche à minimiser le nombre d'hyperarêtes nécessaires pour recouvrir l'ensemble des sommets de l'hypergraphe. On appellera *problème de k-Couverture d'ensemble* (k-CE) le fait de chercher à couvrir, pour un entier k fixé, le plus de sommets possibles avec au plus k hyperarêtes.

Théorème 3.12 *Pour tout ϵ strictement positif, il est $\mathcal{NP}$-difficile d'approximer le problème de Couverture d'ensemble de somme minimum sur les hypergraphes uniformes réguliers avec un rapport de $2 - \epsilon$.*

Une démonstration de ce théorème est esquissée dans sa notation duale dans *Approximating Min-Sum Set Cover* [6], similaire à la preuve, donnée dans *A threshold of ln n for approximating set cover* [7], qu'il est $\mathcal{NP}$-difficile d'approximer le problème k-CE avec un rapport de $\frac{e-1}{e} + \epsilon$.

Théorème 3.13 *Pour tout ϵ strictement positif, il est $\mathcal{NP}$-difficile d'approximer le problème de Couverture d'ensemble de somme minimum avec un rapport de $4 - \epsilon$.*

Une démonstration complète de cette affirmation est fournie dans *Approximating Min-Sum Set Cover* [6]. Elle se base sur le même principe de réduction que celui que nous utiliserons pour démontrer le théorème 4.16 similaire, dans le cadre du problème de Couverture d'ensemble d'entropie minimum présenté au chapitre 4. À savoir présenter un problème de décision P (appelé problème 3-SAT-6) connu pour être $\mathcal{NP}$-difficile, et montrer que si le problème de Couverture d'ensemble d'entropie minimum peut être résolu par un algorithme polynomial avec un rapport $4 - \epsilon$, alors on peut construire une réduction fournissant un algorithme polynomial pour le problème P ; ce qui fournit une contradiction.

Chapitre 4

Problème de Couverture d'ensemble d'entropie minimum

4.1 Détails sur l'entropie

L'informatique, et plus particulièrement la théorie de l'information, nous amènent à regarder la notion d'*entropie*, que nous allons brièvement présenter ici.

Considérons une distribution de probabilité uniforme sur N événements $e_1, e_2, \ldots, e_N$ disjoints et dont l'union est Ω. Chacun de ces événements se produit donc avec une probabilité $1/N$ pour une expérience donnée. Afin de déterminer lequel de ces événements s'est produit lors de l'expérience, on peut se poser un certain nombre de questions auxquelles on répondra par oui ou par non, et dont l'une d'elle déterminera précisément l'élément. Par la méthode de dichotomie sur les indices (méthode qui est optimale) on est amené à poser $\log_2(N)$ questions (à une erreur d'arrondi près). C'est ce qu'on appellera le *prix* à payer pour connaître l'information du résultat de l'expérience.

Considérons maintenant les naturels non nuls $n_1, n_2, \ldots, n_k$ dont la somme vaut N, et désignons $e_1, e_2, \ldots, e_N$ respectivement par $e_{11}, \ldots, e_{1n_1}, e_{21}, \ldots, e_{2n_2}, \ldots, e_{k1}, \ldots, e_{kn_k}$. Le prix à payer pour connaître l'information d'une expérience quelconque reste toujours $\log_2(N)$, et le prix à payer pour connaître l'information en sachant que c'est l'un des $e_{j.}$ vaut $\log_2(n_j)$. Le prix pour savoir que c'est un des $e_{j.}$ est donc $\log_2(N) - \log_2(n_j) = \log_2(\frac{N}{n_j})$.

Supposons maintenant que les événements e_{ji} de $e_{j.}$ pour un j fixé soient égaux pour n'importe quel i. Nous avons alors une distribution de probabilité sur k événements $e_j \equiv e_{j.}$ de probabilité respectivement $p_j \equiv \frac{n_j}{N}$. Le prix à payer pour connaître un tel élément est donc $\log_2(\frac{N}{n_j}) = -\log_2(p_j)$. La valeur N peut être aussi grande qu'on le souhaite, ainsi pour un p_j irrationnel, on peut trouver une suite $< \frac{n_{jt}}{N_t} >_{t\to\infty}$ de rationnels convergeant vers p_j. La mesure de probabilité étant complète (se rapporter à la théorie de la mesure en analyse pour plus de précisions), le résultat reste valable pour n'importe quelle distribution de probabilité dont les probabilités sont des réels ; ce qui se traduit par la proposition suivante :

Proposition 4.1 *Pour toute distribution de probabilité $p_1, \ldots, p_k$ sur les événements $e_1, \ldots, e_k$, le prix à payer pour découvrir le résultat e_j d'une expérience quelconque vaut*

$$-\log_2(p_j) \tag{4.1}$$

Corollaire 4.2 *Pour toute distribution de probabilité $p_1, \ldots, p_k$ sur les événements $e_1, \ldots, e_k$, le prix moyen à payer pour connaître le résultat d'une expérience quelconque vaut*

$$-\sum_{j=1}^{k} p_j \log_2(p_j) \tag{4.2}$$

On appelle cette valeur l'*entropie* de la distribution.

Exemple 3 *Regardons l'ensemble des lettres de l'alphabet simples (sans majuscule et non accentuées). On peut attribuer à chacune d'elles la probabilité d'apparition à un endroit donné d'un texte quelconque. On peut calculer ces occurrences de façon empirique, en considérant différents textes de grande taille et possédant un vocabulaire varié. Le tableau ci-dessous est le résultat de l'analyse (trouvée sur la page http ://www.apprendre-en-ligne.net/crypto/stat/francais.html du site de Didier Müller (Ingénieur informaticien EPFL)) de 100000 lettres réparties sur des textes de Gustave Flaubert (20600 lettres), de Jules Verne (19438 lettres) et de trois articles de l'Encyclopedia Universalis, le premier consacré à Bruges (8182 lettres), le deuxième à l'artillerie (25078 lettres) et le dernier à la population (26702 lettres). On obtient :*

Lettre	*Fréquence*	*Lettre*	*Fréquence*
A	*8.40%*	*N*	*7.13%*
B	*1.06%*	*O*	*5.26%*
C	*3.03%*	*P*	*3.01%*
D	*4.18%*	*Q*	*0.99%*
E	*17.26%*	*R*	*6.55%*
F	*1.12%*	*S*	*8.08%*
G	*1.27%*	*T*	*7.07%*
H	*0.92%*	*U*	*5.74%*
I	*7.34%*	*V*	*1.32%*
J	*0.31%*	*W*	*0.04%*
K	*0.05%*	*X*	*0.45%*
L	*6.01%*	*Y*	*0.30%*
M	*2.96%*	*Z*	*0.12%*

Après un rapide calcul, on obtient l'entropie $H = 3.998$ pour cette distribution. Regardons comment nous pouvons envoyer un message constitué de ces lettres de la façon la plus concise possible. Il est évident que les deux interlocuteurs doivent se mettre d'accord sur certains principes de lecture et d'écriture. En particulier, nous avons vu que

ces lettres peuvent être trouvées une à une par une série de questions dont les réponses sont oui ou non. Pour chacune des lectures d'une lettre la démarche des questions doit être la même et évoluer selon les réponses obtenues. Les deux interlocuteurs sont donc d'accord sur cette démarche et sur l'arbre des questions. De cette façon, l'algorithme de décriptage lira les réponses en attendant de trouver celle qui est décisive, il saura alors qu'il passe à une nouvelle lettre et recommence la démarche. On est ainsi sûr de ne jamais avoir le code d'une lettre qui puisse être le préfixe d'une autre. En transcrivant le texte en oui/non (c'est-à-dire en binaire) la longueur moyenne du code d'une lettre sera l'entropie $H = 3.998$. Cela signifie que pour coder en binaire un texte en français de n lettres, il est raisonnable de s'attendre à un peu moins de $4n$ bits.

Une distribution favorisant les événements apparaissant avec une forte probabilité et avec peu d'événements de faible probabilité aura une entropie faible, contrairement à une distribution étalant sa masse de probabilité sur de nombreux événements peu probables. C'est ce que nous dit le lemme suivant :

Lemme 4.3 *Soient $q =< q_i >$ et $r =< r_i >$ deux distributions de probabilité sur les entiers strictement positifs, dont le support est fini (c'est-à-dire qu'il existe un naturel M tel que pour tout $m \geq M$, les probabilités p_m et q_m sont nulles) et telles que*

$$\sum_{i=1}^{k} r_i \geq \sum_{i=1}^{k} q_i \quad \textit{pour tout naturel } k \tag{4.3}$$

On suppose de plus que $q_i \geq q_{i+1}$ pour tout i.
Alors il existe un naturel N tel que pour tout $n \geq N$ nous avons la relation

$$-\sum_{i=1}^{n} q_i \log_2(q_i) \geq -\sum_{i=1}^{n} r_i \log_2(r_i) \tag{4.4}$$

Démonstration 4.3 Considérons $p =< p_i >$ une distribution de probabilité sur les entiers strictement positifs, à support fini et décroissante. On choisit deux probabilités de cette distribution p_a et p_b avec $a < b$ (ainsi $p_a \geq p_b$). Soit δ dans $]0, p_b]$. Considérons la distribution $p' =< p'_i >$ où $p'_a = p_a + \delta$, $p'_b = p_b - \delta$ et les autres p'_i restent égaux à p_i. On affirme que l'entropie de la distribution p' est inférieure ou égale à l'entropie de la distribution p. En d'autres termes, une telle transformation sur une telle distribution diminue son entropie. Montrons-le :
La fonction $\mathbb{R}_o^+ \to \mathbb{R} : x \mapsto \ln(x) + 1$ est une fonction définie et strictement croissante sur l'intervalle $]0, 1]$. Ainsi

$$\int_{p_b-\delta}^{p_b} (\ln(x) + 1)dx \leq \int_{p_a}^{p_a+\delta} (\ln(x) + 1)dx \tag{4.5}$$

En calculant les intégrales, on trouve

$$p_b \ln(p_b) - (p_b - \delta)\ln(p_b - \delta) \leq (p_a + \delta)\ln(p_a + \delta) - p_a \ln(p_a) \tag{4.6}$$

C'est-à-dire

$$p_a \ln(p_a) + p_b \ln(p_b) \leq (p_a + \delta) \ln(p_a + \delta) + (p_b - \delta) \ln(p_b - \delta) \tag{4.7}$$

En divisant le tout par $-\ln(2)$, on trouve

$$-p_a \log_2(p_a) - p_b \log_2(p_b) \geq -(p_a + \delta) \log_2(p_a + \delta) - (p_b - \delta) \log_2(p_b - \delta) \tag{4.8}$$

En ajoutant les termes $-p_i \log_2(p_i)$ pour les autres indices différents de a et b, on trouve notre affirmation, à savoir qu'il existe un N tel que pour tout $n \geq N$ on a

$$-\sum_{i=1}^{n} p_i \log_2(p_i) \geq -\sum_{i=1}^{n} p'_i \log_2(p'_i) \tag{4.9}$$

Il suffit alors, pour montrer le lemme, de prouver que l'on peut transformer la distribution $q =< q_i >$ en $r =< r_i >$ avec une suite finie de telles transformations. Préalablement, on reclasse les r_i par ordre décroissant, ce qui ne modifie pas l'entropie de la distribution. Le principe est le suivant : à chaque étape i, on ajoutera à q_i ce qu'il lui faut pour égaler r_i. On puisera cette masse de probabilité dans q_{i+1}, et si q_{i+1} s'avère trop petit, on procédera à plusieurs transformations, puisant successivement dans $q_{i+1}, q_{i+2}, \ldots$ Il suffira donc de s'assurer qu'au début de chaque étape i, q_i est bien inférieur ou égal à r_i. Supposons un instant qu'au début de la première transformation visant à rendre q_k égal à r_k on remarque que $q_k > r_k$, alors on a forcément $q_k > 0$. Si q_k n'est pas nul, c'est qu'on n'a jamais eu à puiser dans $q_{k+1}, q_{k+2}, \ldots$, ce qui signifie que la masse de probabilité des $q_1, \ldots, q_k$ n'a pas augmenté, elle est donc restée égale à la somme initiale $\sum_{i=1}^{k} q_i$ qui était inférieure ou égale à la somme $\sum_{i=1}^{k} r_i$. Cela veut dire qu'à ce stade, on a toujours $\sum_{i=1}^{k} q_k \leq \sum_{i=1}^{k} r_i$, avec $q_i = r_i$ pour $i < k$. Il y a donc contradiction avec la supposition $q_k > r_k$. Ceci termine de démontrer le lemme. □

4.2 Présentation du problème

Nous nous intéressons cette fois à l'article de Jean Cardinal, Samuel Fiorini et Gwenaël Joret *Tight Results on Minimum Entropy Set Cover*. Comme nous l'avons fait pour l'article précédent, nous retranscrivons les résultats relatifs au problème de Couverture d'ensemble d'entropie minimum dans les mêmes notations que précédemment. Nous ajoutons à ces résultats un certains nombre de lemmes et de détails de calculs.

Toujours dans l'idée de couvrir l'ensemble V d'un hypergraphe $H(V, E)$ à l'aide des hyperarêtes de E, nous nous tournons cette fois vers la minimisation de l'entropie. Selon le même schéma que précédemment, on considère un hypergraphe $H(V, E)$. Soit $f : E \to \{1, 2, \ldots, |E|\} : e \mapsto f(e)$ une organisation linéaire. Si v est un sommet quelconque de V, on définit $E_v = \{e \in E | v \in e\}$ l'ensemble des hyperarêtes de E contenant v et on définit $f(v) = \min_{e \in E_v} f(e)$. Posons $C_e = \{v \in V : f(v) = f(e)\}$ l'ensemble des sommets

couverts par e, comme nous l'avions déjà remarqué dans la section précédente, nous avons $\sum_{e \in E} |C_e| = |V|$, et $\frac{|C_e|}{|V|}$ représente la probabilité qu'un sommet quelconque v soit couvert par l'hyperarête e, sous l'hypothèse d'une distribution uniforme des sommets. On note $p_e = \frac{|C_e|}{|V|}$. Soulignons que p_e dépend de l'organisation linéaire f. En prenant pour convention de poser $x \cdot \log(x) = 0$ lorsque $x = 0$, nous définissons la fonction objectif

$$F_{obj} : \mathcal{F}_E \to \mathbb{R}^+ : f \mapsto -\sum_{e \in E} p_e \log_2(p_e) \tag{4.10}$$

où $\mathcal{F}_E$ est l'ensemble des organisations linéaires sur E. Le logarithme est utilisé en base 2 pour des raisons évidentes d'utilisation du binaire en informatique. Notons néanmoins qu'il est possible de rencontrer des situations différentes, avec un logarithme en base différente de 2 (en base e par exemple) ; le problème reste cependant essentiellement le même. Nous tenterons alors de déterminer une organisation linéaire f qui minimise la fonction objectif.

Proposition 4.4 *Dans le cadre du problème présenté ici, il est équivalent de chercher à maximiser la fonction objectif*

$$F'_{obj} : \mathcal{F}_E \to \mathbb{R}^+ : f \mapsto \prod_{e \in E} |C_e|^{|C_e|} \tag{4.11}$$

où l'on convient de poser $|C_e|^{|C_e|} = 0$ *lorsque* $C_e = \emptyset$.

Démonstration 4.4 Selon les propriétés habituelles des logarithmes, nous obtenons la suite d'égalités :

$$\begin{aligned}
F_{obj}(f) &= -\sum_{e \in E} p_e \log_2(p_e) && (4.12)\\
&= -\sum_{e \in E} \frac{|C_e|}{|V|} \log_2 \left(\frac{|C_e|}{|V|} \right) && (4.13)\\
&= -\frac{1}{|V|} \sum_{e \in E} |C_e| (\log_2(|C_e|) - \log_2(|V|)) && (4.14)\\
&= -\frac{1}{|V|} \sum_{e \in E} \log_2 \left(|C_e|^{|C_e|} \right) + \frac{1}{|V|} \log_2(|V|) \sum_{e \in E} |C_e| && (4.15)\\
&= -\frac{1}{|V|} \log_2 \left(\prod_{e \in E} |C_e|^{|C_e|} \right) + \log_2(|V|) && (4.16)
\end{aligned}$$

Minimiser cette valeur revient à maximiser

$$\log_2 \left(\prod_{e \in E} |C_e|^{|C_e|} \right) \tag{4.17}$$

C'est-à-dire maximiser

$$\prod_{e \in E} |C_e|^{|C_e|} = F'_{obj}(f) \tag{4.18}$$

□

4.3 Algorithme glouton

L'algorithme glouton pour ce problème reste identique aux précédents. Il se contente, à chaque étape, de sélectionner l'hyperarête couvrant le plus de sommets non encore couverts. Nous rappelons son implémentation, pour une instance du problème d'optimisation « Étant donné un hypergraphe $H(V, E)$, on cherche une organisation linéaire $e_1, e_2, \ldots, e_k$ dans E telle que $\bigcup_{i=1}^{k} e_i = V$ et qui minimise notre fonction objectif ». Pratiquement :

1. un *ensemble de sommets* $R \leftarrow V$
2. un *ensemble indicé d'hyperarêtes* $D \leftarrow \emptyset$
3. tant que $R \neq \emptyset$ faire :
   (a) sélectionner $e \in E \backslash D$ qui maximise $|R \cap e|$
   (b) ajouter e dans D
   (c) supprimer de R les éléments de e qui s'y trouvent
4. retourner D

4.4 Facteur d'approximation

Nous allons montrer que cette fois-ci, l'algorithme glouton est encore plus puissant que dans les deux cas précédents. En effet, nous verrons qu'au pire cas, l'algorithme glouton fournit une solution qui est égale à la solution optimale à laquelle s'ajoute une constante indépendante de la taille du problème.

Soit f_G l'organisation linéaire sélectionnée par l'algorithme glouton, et soit f_{OPT} une organisation linéaire d'entropie minimum. Soit C_e l'ensemble des sommets qui seront couverts lors de l'étape où l'algorithme glouton sélectionne e, et soit X_e l'ensemble des sommets qui seront couverts lors de l'étape où l'algorithme optimal sélectionne e, où on pose $x_e = |X_e|$. On note e_v l'hyperarête telle que $v \in C_{e_v}$, c'est-à-dire l'hyperarête qui couvrira le sommet v, et soit $a_v = |C_{e_v}|$ le nombre de sommets couverts en même temps que v.

Lemme 4.5

$$\prod_{v \in X_e} a_v \geq x_e! \quad \textit{pour tout } e \in E \tag{4.19}$$

Démonstration 4.5 Notons $X_e = \{v_1, v_2, \ldots, v_{x_e}\}$ où les v_i sont dans l'ordre dans lequel ils seront couverts par l'algorithme glouton ; on les choisit de façon arbitraire lorsqu'ils sont dans la même hyperarête sélectionnée par l'algorithme (c'est-à-dire si $C_{e_{v_i}} = C_{e_{v_j}}$). Considérons un v_j, l'algorithme glouton le couvre avec l'hyperarête e_{v_j}. À cet instant, au plus $j-1$ sommets de X_e sont couverts, ce qui signifie qu'au moins $x_e - j + 1$ sommets ne sont pas encore couverts dans X_e. L'algorithme glouton étant celui qui, à chaque étape, sélectionne l'hyperarête possédant le plus de sommets non encore couverts, et cette hyperarête étant ici e_{v_j}, on sait qu'au moins $x_e - j + 1$ des sommets de e_{v_j} ne sont pas couverts, sans quoi l'algorithme glouton aurait sélectionné X_e. Cela signifie que

$$a_{v_j} = |C_{e_{v_j}}| \geq x_e - j + 1 \tag{4.20}$$

Et donc

$$\prod_{v \in X_e} a_v \geq \prod_{j=1}^{x_e} (x_e - j + 1) = x_e! \tag{4.21}$$

□

Rappelons la formule de Stirling :

Lemme 4.6 *Pour tout entier positif n, nous avons la relation*

$$\left(\frac{n}{e}\right)^n \leq n! \leq 2\sqrt{2\pi n}\left(\frac{n}{e}\right)^n \tag{4.22}$$

Théorème 4.7

$$F_{obj}(f_G) \leq F_{obj}(f_{OPT}) + \log_2(e) \tag{4.23}$$

Démonstration 4.7 En posant $|V| = n$, nous avons :

$$F_{obj}(f_G) = -\sum_{e\in E} \frac{|C_e|}{n} \log_2\left(\frac{|C_e|}{n}\right) \tag{4.24}$$

$$= -\frac{1}{n}\sum_{e\in E} |C_e| \log_2\left(\frac{|C_e|}{n}\right) \tag{4.25}$$

$$= -\frac{1}{n}\sum_{e\in E}\sum_{v\in C_e} \log_2\left(\frac{|C_{e_v}|}{n}\right) \tag{4.26}$$

$$= -\frac{1}{n}\sum_{v\in V} \log_2\left(\frac{|C_{e_v}|}{n}\right) \tag{4.27}$$

$$= -\frac{1}{n}\sum_{v\in V} \log_2\left(\frac{a_v}{n}\right) \tag{4.28}$$

$$= -\frac{1}{n}\sum_{e\in E}\sum_{v\in X_e} \log_2\left(\frac{a_v}{n}\right) \tag{4.29}$$

$$= -\frac{1}{n}\sum_{e\in E} \log_2\left(\prod_{v\in X_e} \frac{a_v}{n}\right) \tag{4.30}$$

$$\leq -\frac{1}{n}\sum_{e\in E} \log_2\left(\frac{x_e!}{n^{x_e}}\right) \quad \text{par le lemme 4.5} \tag{4.31}$$

$$\leq -\frac{1}{n}\sum_{e\in E} \log_2\left(\frac{{x_e}^{x_e}}{n^{x_e} e^{x_e}}\right) \quad \text{par la formule de Stirling} \tag{4.32}$$

$$= -\frac{1}{n}\sum_{e\in E} \left(\log_2\left(\frac{{x_e}^{x_e}}{n^{x_e}}\right) - \log_2(e^{x_e})\right) \tag{4.33}$$

$$= -\frac{1}{n}\sum_{e\in E} x_e \log_2\left(\frac{x_e}{n}\right) + \frac{1}{n}\sum_{e\in E} x_e \log_2(e) \tag{4.34}$$

$$= -\sum_{e\in E} \frac{|X_e|}{n} \log_2\left(\frac{|X_e|}{n}\right) + \log_2(e) \tag{4.35}$$

$$= F_{obj}(f_{OPT}) + \log_2(e) \tag{4.36}$$

□

Théorème 4.8 *Pour tout ϵ strictement positif il existe une instance P du problème de Couverture d'ensemble d'entropie minimum, où f_P est l'organisation linéaire obtenue en appliquant l'algorithme glouton à P, telle que*

$$F_{obj}(f_P) \geq F_{obj}(f_{OPT}) + \log_2(e) - \epsilon \tag{4.37}$$

Démonstration 4.8 Pour montrer cela, il suffit de donner un exemple d'un tel P. Considérons un ensemble V de $n \cdot n!$ sommets, que l'on représente comme une matrice

$$V = \{v_{ij} | i \in \{1, \ldots, n\}, j \in \{1, \ldots, n!\}\} \tag{4.38}$$

Posons $\mathcal{C}_{COL}$ et $\mathcal{C}_{LIG}$ deux collections d'hyperarêtes définies par

$$\mathcal{C}_{COL} = \{\{v_{ij} | i \in \{1, \ldots, n\}\} | j \in \{1, \ldots, n!\}\} \tag{4.39}$$

$$\mathcal{C}_{LIG} = \{\{v_{ij} | j = qi + r, r \in \{1, \ldots, i\}\} | q \in \{1, \ldots, \frac{n!}{i}\}, i \in \{1, \ldots, n\}\} \tag{4.40}$$

comme dans la figure 4.1. Parmi ces deux collections, dont l'union forme l'ensemble des

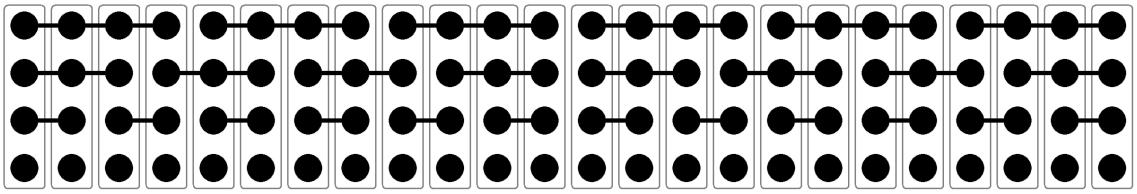

FIG. 4.1 – Représentation de $\mathcal{C}_{COL}$ et $\mathcal{C}_{LIG}$ en n lignes et $n!$ colonnes

hyperarêtes de notre instance, l'algorithme glouton sélectionnera la collection définie par f_{LIG}, et nous pouvons déterminer un algorithme sélectionnant la collection définie par f_{COL}. Calculons $F_{obj}(f_{COL})$ et $F_{obj}(f_{LIG})$.

$$F_{obj}(f_{COL}) = -\sum_{e \in \mathcal{C}_{COL}} \frac{|C_e|}{|V|} \log_2 \left(\frac{|C_e|}{|V|} \right) \tag{4.41}$$

$$= -\sum_{e \in \mathcal{C}_{COL}} \frac{1}{n!} \log_2 \left(\frac{1}{n!} \right) \tag{4.42}$$

$$= -n! \frac{1}{n!} \log_2 \left(\frac{1}{n!} \right) \tag{4.43}$$

$$= \log_2(n!) \tag{4.44}$$

$$F_{obj}(f_{LIG}) = -\sum_{e \in \mathcal{C}_{LIG}} \frac{|C_e|}{|V|} \log_2 \left(\frac{|C_e|}{|V|} \right) \tag{4.45}$$

$$= -\sum_{i=1}^{n} \sum_{j=1}^{\frac{n!}{i}} \frac{i}{n \cdot n!} \log_2 \left(\frac{i}{n \cdot n!} \right) \tag{4.46}$$

$$= -\sum_{i=1}^{n} \frac{n!}{i} \frac{i}{n \cdot n!} \log_2 \left(\frac{i}{n \cdot n!} \right) \tag{4.47}$$

$$= -\frac{1}{n} \sum_{i=1}^{n} \log_2 \left(\frac{i}{n \cdot n!} \right) \tag{4.48}$$

$$= -\frac{1}{n} \sum_{i=1}^{n} \log_2(i) + \frac{1}{n} \sum_{i=1}^{n} \log_2(n \cdot n!) \tag{4.49}$$

$$= -\frac{1}{n} \log_2(n!) + \log_2(n) + \log_2(n!) \tag{4.50}$$

Par le lemme de Stirling, nous obtenons alors

$$F_{obj}(f_{LIG}) = \log_2(n!) + \log_2(n) - \frac{1}{n} \log_2(n!) \tag{4.51}$$

$$\geq \log_2(n!) + \log_2(n) - \frac{1}{n} \log_2 \left(2\sqrt{2\pi n} \left(\frac{n}{e} \right)^n \right) \tag{4.52}$$

$$= \log_2(n!) + \log_2(n) - \frac{1}{n} \log_2(2\sqrt{2\pi n}) - \log_2 \left(\frac{n}{e} \right) \tag{4.53}$$

$$= \log_2(n!) + \log_2(e) - \frac{1}{n} \log_2(2\sqrt{2\pi n}) \tag{4.54}$$

$$= F_{obj}(f_{COL}) + \log_2(e) - \frac{1}{n} \log_2(2\sqrt{2\pi n}) \tag{4.55}$$

Par définition de f_{OPT}, nous savons que $F_{obj}(f_{OPT}) \leq F_{obj}(f_{COL})$, ce qui permet d'écrire

$$F_{obj}(f_{LIG}) \geq F_{obj}(f_{OPT}) + \log_2(e) - \frac{1}{n} \log_2(2\sqrt{2\pi n}) \tag{4.56}$$

Il suffit alors de choisir n suffisamment grand pour que

$$\frac{1}{n} \log_2(2\sqrt{2\pi n}) \leq \epsilon \tag{4.57}$$

□

4.5 Inapproximabilité

Nous allons montrer qu'il n'existe pas d'algorithme polynomial pour le problème de Couverture d'ensemble d'entropie minimum fournissant une meilleure approximation que l'algorithme glouton, à moins que $\mathcal{P}$=$\mathcal{NP}$.

Avant cela, il nous faut présenter le problème 3-SAT-6. Nous avions déjà vu les problèmes SAT et 3-SAT à la section 1.3.5. Nous avions un ensemble X composé de n variables booléennes $x_1, x_2, \ldots, x_n$. Avec des termes t_j de la forme x_i ou $\overline{x_i}$, nous considérions des clauses $t_1 \vee t_2 \vee \cdots \vee t_k$. Une attribution de vérité était une application $\nu : X \to \{0, 1\}$ qui à chaque terme associait une valeur booléenne. Une clause était satisfaite lorsqu'au moins l'un de ses termes était évalué à 1. Le problème SAT considérait alors une conjonction de clauses $c_1 \wedge c_2 \wedge \cdots \wedge c_q$ et consistait à chercher une attribution de vérité vérifiant chacune de ces clauses. Le problème 3-SAT était quant à lui un cas particulier de SAT pour lequel la longueur de chacune des clauses était fixée à 3 termes. Le problème 3-SAT-6 est un cas particulier de 3-SAT pour lequel chaque clause ne contient pas deux fois la même variable x_i, et chaque terme x_j ou $\overline{x_i}$ apparaît dans exactement 3 clauses. Pour exemple, l'expression suivante est une instance de 3-SAT-6 :

$$(x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee x_2 \vee \overline{x_3}) \quad (4.58)$$

Nous dirons qu'une instance de 3-SAT-6 est *δ-satisfaisable* si au moins une proportion δ des q clauses est satisfaite. Une instance est *satisfaisable* si elle est 1-satisfaisable, c'est-à-dire si toutes les clauses peuvent être satisfaites. Il est établi dans [6] que pour certains δ dans $]0, 1[$, il est $\mathcal{NP}$-difficile de distinguer une instance de 3-SAT-6 satisfaisable d'une instance δ-satisfaisable. Dans *Approximating Min-Sum Set Cover* [6] et *A threshold of ln n for approximating set cover* [7] est montrée une réduction qui à toute instance ϕ de 3-SAT-6 associe un hypergraphe $H(\phi)(V, E)$, qui possède la propriété suivante : Pour tous réels $c > 0$ et $\lambda > 0$ fixés, et pour tout naturel $t > 0$ fixé, il est possible de choisir les paramètres de la réduction de telle façon que :

- Les hyperarêtes de E sont toutes de taille commune $\frac{n}{t}$ où $n = |V|$, et où tous les sommets sont de même degré.
- Si ϕ est satisfaisable, alors V peut être couvert par t hyperarêtes disjointes de E.
- Si ϕ est δ-satisfaisable, chaque collection de ℓ hyperarêtes choisies dans E couvre au plus une proportion $1 - (1 - \frac{1}{t})^{\ell} + \lambda$ des sommets de V, pour tout ℓ avec $1 \leq \ell \leq ct$.

La démonstration du théorème nécessite un certain nombre de lemmes d'analyse que nous démontrons avant. La démonstration est présentée de façon similaire à celle publiée dans *Tight Results on Minimum Entropy Set Cover* [9]. Dans notre version, le canevas reste le même, cependant nous ajoutons les détails de nombreux calculs, et certaines affirmations sont démontrées de façon différente. Le squelette de la démonstration repose sur l'idée suivante : On sait initialement qu'il est $\mathcal{NP}$-difficile de différencier une instance satisfaisable d'une instance δ-satisfaisable pour le problème 3-SAT-6. On utilise alors une réduction du problème 3-SAT-6 vers le problème de Couverture d'ensemble d'entropie minimum sur un hypergraphe uniforme régulier, réduction qui permet de prouver que si le problème de couverture sur l'hypergraphe peut être résolu de façon plus intéressante que ce que fait l'algorithme glouton, alors ce même algorithme peut être utilisé pour différencier une instance satisfaisable d'une instance δ-satisfaisable. Sachant que cette différenciation est $\mathcal{NP}$-difficile, le problème de Couverture d'ensemble d'entropie minimum sera également $\mathcal{NP}$-difficile.

Lemme 4.9 *Soit $t > 1$, alors nous avons la relation*

$$\frac{-t}{t-1} < t \ln\left(1 - \frac{1}{t}\right) < -1 \tag{4.59}$$

Démonstration 4.9 On a trivialement la suite d'inégalités suivante

$$\frac{1}{t} < \int_{t-1}^{t} \frac{dx}{x} < \frac{1}{t-1} \tag{4.60}$$

$$\frac{1}{t} < \ln(t) - \ln(t-1) < \frac{1}{t-1} \tag{4.61}$$

$$\frac{-1}{t-1} < \ln\left(\frac{t-1}{t}\right) < \frac{-1}{t} \tag{4.62}$$

$$\frac{-t}{t-1} < t \ln\left(1 - \frac{1}{t}\right) < -1 \tag{4.63}$$

qui montrent le lemme. □

Corollaire 4.10 *Soit $t > 1$, alors nous avons la relation*

$$e^{\frac{-t}{t-1}} < \left(1 - \frac{1}{t}\right)^t < e^{-1} \tag{4.64}$$

Démonstration 4.10 La démonstration est immédiate si on remarque que

$$\left(1 - \frac{1}{t}\right)^t = e^{\ln\left(\left(1-\frac{1}{t}\right)^t\right)} = e^{t \ln\left(1-\frac{1}{t}\right)} \tag{4.65}$$

□

Lemme 4.11

$$\lim_{t \to +\infty} \left[te^{-1} - t\left(1 - \frac{1}{t}\right)^t \right] \leq e^{-1} \tag{4.66}$$

Démonstration 4.11 On a tout naturellement pour tout $t > 1$

$$
\begin{aligned}
-\ln(e^{-1}) &= 1 && (4.67)\\
&= (t-1)\frac{1}{t-1} && (4.68)\\
&> (t-1)\int_{t-1}^{t}\frac{dx}{x} && (4.69)\\
&= (t-1)(\ln(t)-\ln(t-1)) && (4.70)\\
&= -(t-1)\ln\left(\frac{t-1}{t}\right) && (4.71)\\
&= -\ln\left(\left(1-\frac{1}{t}\right)^{t-1}\right) && (4.72)
\end{aligned}
$$

On en déduit successivement les inégalités

$$
\begin{aligned}
\ln(e^{-1}) &< \ln\left(\left(1-\frac{1}{t}\right)^{t-1}\right) && (4.73)\\
e^{-1} &< \left(1-\frac{1}{t}\right)^{t-1} && (4.74)\\
e^{-1}-\frac{e^{-1}}{t} &< \left(1-\frac{1}{t}\right)^{t} && (4.75)\\
e^{-1}-\left(1-\frac{1}{t}\right)^{t} &< \frac{e^{-1}}{t} && (4.76)\\
te^{-1}-t\left(1-\frac{1}{t}\right)^{t} &< e^{-1} && (4.77)
\end{aligned}
$$

On en déduit immédiatement le lemme. □

Lemme 4.12 *Si on considère un réel $\alpha > 0$ assez petit, alors nous avons l'inégalité*

$$\ln(\alpha+e^{-\alpha}-1) > \ln\left(\frac{\alpha^2}{2}-\frac{\alpha^3}{6}\right) \tag{4.78}$$

Démonstration 4.12 Comme les logarithmes sont des fonctions strictement croissantes, il s'agit de montrer l'inégalité

$$\alpha+e^{-\alpha}-1 > \frac{\alpha^2}{2}-\frac{\alpha^3}{6} \tag{4.79}$$

ou encore, montrer que

$$\alpha^3-3\alpha^2+6\alpha-6+6e^{-\alpha} > 0 \tag{4.80}$$

On a égalité pour $\alpha = 0$, il suffit donc de prouver que la dérivée est strictement positive dans un voisinage positif de 0, c'est-à-dire montrer que

$$3\alpha^2 - 6\alpha + 6 - 6e^{-\alpha} > 0 \tag{4.81}$$

Selon la même remarque, il suffit de montrer que

$$6\alpha - 6 + 6e^{-\alpha} > 0 \tag{4.82}$$

ou encore que

$$6 - 6e^{-\alpha} > 0 \tag{4.83}$$

ce qui est évident puisque $\alpha \mapsto e^{-\alpha}$ est une fonction strictement décroissante et que $e^{-0} = 1$. □

Lemme 4.13 *Soient $\alpha : \mathbb{R}_o^+ \to \mathbb{R}_o^+ : t \mapsto \alpha(t)$ et $\beta : \mathbb{R}_o^+ \to \mathbb{R}_o^+ : t \mapsto \beta(t)$ des fonctions telles que $\lim_{t\to+\infty} \alpha(t) = a$ et $\lim_{t\to+\infty} \beta(t) = b$, avec a et b dans $\mathbb{R}_o^+$. Alors on a la relation*

$$\lim_{t\to+\infty} \alpha(t)^{\beta(t)} = a^b \tag{4.84}$$

Démonstration 4.13 Nous avons successivement

$$\alpha(t)^{\beta(t)} = e^{\ln(\alpha(t)^{\beta(t)})} = e^{\beta(t)\ln(\alpha(t))} \longrightarrow e^{b\ln(a)} = e^{\ln(a^b)} = a^b \tag{4.85}$$

□

Lemme 4.14 *Soit $0 < \alpha < y < x < 1$ des réels, alors $x^\alpha - y^\alpha < x - y$.*

Démonstration 4.14 Il faut montrer que $x^\alpha - x < y^\alpha - y$, c'est-à-dire montrer que $x \mapsto x^\alpha - x$ est décroissante pour $x > \alpha$. Pour cela, il suffit de montrer que sa dérivée $\alpha x^{\alpha-1} - 1$ est strictement négative. Or, avec les hypothèses posées, nous avons $\alpha x^\alpha < \alpha 1^\alpha = \alpha < x$, ce qui fournit $\alpha x^{\alpha-1} < 1$ qui confirme que la dérivée est négative. □

Lemme 4.15 *Pour tout entier positif n et pour tout réel $q > 0$ différent de 1, nous avons*

$$\sum_{k=0}^{n} q^k = \frac{q^{n+1} - 1}{q - 1} \tag{4.86}$$

Démonstration 4.15 Montrons cela par induction sur n. Pour $n = 0$ le résultat est évident. Nous avons, pour le pas récursif,

$$\sum_{k=0}^{n+1} q^k = q^{n+1} + \sum_{k=0}^{n} q^k \tag{4.87}$$

$$= q^{n+1} + \frac{q^{n+1} - 1}{q - 1} \tag{4.88}$$

$$= \frac{q^{n+2} - q^{n+1} + q^{n+1} - 1}{q - 1} \tag{4.89}$$

$$= \frac{q^{n+2} - 1}{q - 1} \tag{4.90}$$

ce qui montre l'induction et donc le lemme. □

À présent, nous pouvons formaliser et démontrer le théorème d'inapproximabilité par algorithme polynomial du problème de Couverture d'ensemble d'entropie minimum.

Théorème 4.16 *Pour tout $\epsilon > 0$, il est $\mathcal{NP}$-difficile d'approximer le problème de Couverture d'ensemble d'entropie minimum avec un terme additif $(1 - \epsilon)\log_2(e)$.*

Démonstration 4.16 Si nous montrons le théorème pour tous les hypergraphes uniformes réguliers, nous aurons le résultat. Considérons une instance ϕ de 3-SAT-6 qui soit satisfaisable ou δ-satisfaisable. On note f_{OPT} une organisation linéaire optimale pour l'instance du problème de Couverture d'ensemble d'entropie minimum définie par l'hypergraphe $H(\phi) = H(\phi)(V, E)$. Pour chaque $e \in E$ on définit $p_e = p_e(f_{OPT}) = \frac{|C_e|}{|V|}$ comme nous l'avons fait dans la section 4.2, où les C_e sont associés à l'organisation linéaire f_{OPT}. On peut indicer les éléments de E $e_1, e_2, \ldots, e_{|E|}$ de telle façon que $p_{e_1} \geq p_{e_2} \geq \cdots \geq p_{e_{|E|}}$. Nous savons que $< p_{e_i} >$ est une distribution de probabilité, ici décroissante.
Considérons le premier cas : ϕ est satisfaisable. Alors le lemme 4.3 nous informe qu'une solution optimale consiste à recouvrir V avec t hyperarêtes disjointes, car la somme $\sum_{i=1}^{k} p_{e_i}$ est maximale pour tout k, étant donné que l'hypergraphe est uniforme régulier. L'entropie d'une solution optimale est donc l'entropie de la distribution $< p_{e_i} >$, c'est-à-dire

$$-\sum_{i=1}^{t} p_{e_i} \log_2(p_{e_i}) = -\sum_{i=1}^{t} \frac{|C_e|}{|V|} \log_2\left(\frac{|C_e|}{|V|}\right) \tag{4.91}$$

$$= -\sum_{i=1}^{t} \frac{1}{t} \log_2\left(\frac{1}{t}\right) \tag{4.92}$$

$$= \log_2(t) \tag{4.93}$$

Considérons le second cas : ϕ est δ-satisfaisable. Posons $\alpha = \frac{\epsilon}{2}$, $\lambda = \frac{\alpha^2}{2} - \frac{\alpha^3}{6}$, et $c = -\ln(\lambda)$. Si on arrive à prouver que

$$ENT(< p_{e_i} >) \geq \log_2(t) + \left(1 - \frac{\epsilon}{2}\right)\log_2(e) + o(1) \tag{4.94}$$

où o(1) tend vers 0 lorsque t tend vers $+\infty$, alors tout algorithme A approximant le problème de Couverture d'ensemble d'entropie minimum avec un terme additif $(1-\epsilon)\log_2(e)$ pourra être utilisé pour décider si ϕ est satisfaisable ou δ-satisfaisable. En effet, remarquant que

$$\log_2(t) + \left(1 - \frac{\epsilon}{2}\right)\log_2(e) + o(1) > \log_2(t) + (1-\epsilon)\log_2(e) \tag{4.95}$$

si notre algorithme trouve une solution $x \leq \log_2(t) + (1-\epsilon)\log_2(e)$, il est évident qu'une solution optimale sera inférieure ou égale à x, c'est-à-dire strictement inférieure à $\log_2(t) + (1-\frac{\epsilon}{2})\log_2(e) + o(1)$. Ainsi l'instance ϕ ne peut pas être δ-satisfaisable, elle est donc satisfaisable. Si au contraire, notre algorithme trouve une solution $x > \log_2(t) + (1-\epsilon)\log_2(e)$, la solution optimale ne peut être $\log_2(t)$, puisque l'algorithme nous garantit un terme additif inférieur ou égal à $(1-\epsilon)\log_2(e)$. Ainsi ϕ n'est pas satisfaisable mais δ-satisfaisable. Ceci implique le théorème, sans quoi nous irions à l'encontre du fait établi qu'il est $\mathcal{NP}$-difficile de distinguer une instance de 3-SAT-6 satisfaisable d'une instance δ-satisfaisable.
Il nous faut maintenant montrer l'affirmation (4.94). Pour cela, on définit une suite $< r_i >$ comme suit :

$$r_i = \begin{cases} \frac{1}{t} & \text{si } 1 \leq i \leq \lceil \alpha t \rceil \\ \frac{1}{t}(1-\frac{1}{t})^{i-1} & \text{si } \lceil \alpha t \rceil + 1 \leq i \leq \lfloor \tilde{c} t \rfloor \\ 1 - \sum_{i=1}^{\lfloor \tilde{c} t \rfloor} r_i & \text{si } i = \lfloor \tilde{c} t \rfloor + 1 \\ 0 & \text{sinon} \end{cases} \tag{4.96}$$

où $\tilde{c}$ est un réel tel que

$$\frac{\lceil \alpha t \rceil}{t} + \left(1 - \frac{1}{t}\right)^{\lceil \alpha t \rceil} - \left(1 - \frac{1}{t}\right)^{\tilde{c} t} = 1 \tag{4.97}$$

Affirmation 4.17 *$\tilde{c}$ est bien défini.*

Démonstration 4.17 Remarquons tout d'abord que l'on peut toujours augmenter t sans modifier le théorème ; de même, le théorème ne change pas si on diminue α, c'est-à-dire si on dimine ϵ. Ainsi, dans cette démonstration nous modifierons α et t de manière à ce que αt reste supérieur ou égal à une constante que nous fixerons en temps voulu. On voit facilement que $\tilde{c}$ est défini si et seulement si on peut écrire

$$\tilde{c} = \frac{\ln\left(\frac{\lceil \alpha t \rceil}{t} + (1-\frac{1}{t})^{\lceil \alpha t \rceil} - 1\right)}{t \cdot \ln(1-\frac{1}{t})} \tag{4.98}$$

On voit immédiatement que des restrictions s'imposent sur α et t, en effet, pour $\alpha t < 1$ le logarithme n'est plus défini. On augmente donc t sans modifier α de telle façon que $t > \frac{1}{\alpha}$. On pose alors $a = \lceil \alpha t \rceil$ un entier strictement plus grand que 1. On peut alors se permettre de faire augmenter α et diminuer t de façon à garder a constant. On peut réécrire la définition de $\tilde{c}$:

$$\tilde{c} = \frac{\ln\left(\frac{a}{t} + (1 - \frac{1}{t})^a - 1\right)}{t \cdot \ln(1 - \frac{1}{t})} \tag{4.99}$$

Pour que $\tilde{c}$ soit bien défini, il suffit de s'assurer que pour tout t assez grand on a bien

$$\frac{a}{t} + \left(1 - \frac{1}{t}\right)^a - 1 > 0 \tag{4.100}$$

Afin de faciliter les calculs suivants, on suppose a pair ; si ce n'était pas le cas, il suffit de multiplier t par 2. En choisissant $t > \frac{a}{2}$, on trouve, pour tout k entier dans $[1, \frac{a}{2}]$, la suite d'inégalités suivante :

$$t > \frac{a}{2} \geq \frac{a - 2k}{2k + 1} \tag{4.101}$$

$$\frac{1}{a - 2k} > \frac{1}{2k + 1} \cdot \frac{1}{t} \tag{4.102}$$

$$\frac{a!}{(a - 2k)!(2k)!} \cdot \frac{1}{t^{2k}} > \frac{a!}{(a - 2k - 1)!(2k + 1)!} \cdot \frac{1}{t^{2k+1}} \tag{4.103}$$

$$\binom{a}{2k}\left(\frac{1}{t}\right)^{2k} - \binom{a}{2k+1}\left(\frac{1}{t}\right)^{2k+1} > 0 \tag{4.104}$$

$$\sum_{k=1}^{\frac{a}{2}} \left(\binom{a}{2k}\left(\frac{1}{t}\right)^{2k} - \binom{a}{2k+1}\left(\frac{1}{t}\right)^{2k+1} \right) > 0 \tag{4.105}$$

$$\sum_{j=2}^{a} \binom{a}{j}\left(-\frac{1}{t}\right)^{j} > 0 \tag{4.106}$$

$$\frac{a}{t} - 1 + \sum_{j=0}^{a} \binom{a}{j}\left(-\frac{1}{t}\right)^{j} > 0 \tag{4.107}$$

$$\frac{a}{t} - 1 + \left(1 - \frac{1}{t}\right)^a > 0 \tag{4.108}$$

Ainsi $\tilde{c}$ est bien défini. □

Affirmation 4.18 *La suite des $< r_i >$ est parfaitement définie.*

Démonstration 4.18 Ce sera le cas si on montre que

$$1 \leq \lceil \alpha t \rceil \quad \text{et} \quad \lceil \alpha t \rceil + 1 \leq \lfloor \tilde{c} t \rfloor \tag{4.109}$$

$1 \leq \lceil \alpha t \rceil$ est évident puisqu'on a supposé précédemment que $\lceil \alpha t \rceil = a \geq 2$. Pour montrer que $\lceil \alpha t \rceil + 1 \leq \lfloor \tilde{c}t \rfloor$ il suffit de montrer que $\tilde{c}t$ tend vers $+\infty$ lorsque t grandit vers $+\infty$ et que a reste constant. La fonction $\tilde{c}t$ qui est une fonction de t est définie par

$$\tilde{c}t = \frac{\ln(\frac{a}{t} + (1 - \frac{1}{t})^a - 1)}{\ln(1 - \frac{1}{t})} \tag{4.110}$$

On affirme que cette fonction de t est strictement croissante pour t assez grand. En effet, on a

$$\lim_{t \to +\infty} \left[\frac{a}{t} + \left(1 - \frac{1}{t}\right)^a - 1 \right] = 0 \quad \text{et} \quad \lim_{t \to +\infty} \left[1 - \frac{1}{t} \right] = 1 \tag{4.111}$$

ce qui implique immédiatement que

$$\lim_{t \to +\infty} \left[\ln \left(\frac{a}{t} + \left(1 - \frac{1}{t}\right)^a - 1 \right) \right] = -\infty \quad \text{et} \quad \lim_{t \to +\infty} \left[\ln \left(1 - \frac{1}{t} \right) \right] = 0 \tag{4.112}$$

avec $\ln(1 - \frac{1}{t}) < 0$. Ainsi $\lim_{t \to +\infty} \tilde{c}t = +\infty$. La suite $< r_i >$ est donc bien définie. □

Affirmation 4.19 $< r_i >$ *est une distribution de probabilité.*

Démonstration 4.19 Pour s'en assurer, il faut remarquer que la somme des r_i vaut 1, ce qui est évident par la définition de r_i lorsque $i = \lfloor \tilde{c}t \rfloor + 1$, et il faut démontrer que chacun des r_i définis est positif ou nul. C'est évident pour les termes $\frac{1}{t}$ et $\frac{1}{t}(1 - \frac{1}{t})^{i-1}$. Il suffit donc de montrer que

$$\sum_{i=1}^{\lfloor \tilde{c}t \rfloor} r_i \leq 1 \tag{4.113}$$

Le lemme 4.15 nous apprend que pour toute paire d'entiers m, n tels que $1 \leq m \leq n$, et pour tout réel $q > 0$ différent de 1 nous avons

$$\sum_{k=m}^{n} q^k = \frac{q^{n+1}}{q-1} - \frac{q^m}{q-1} \tag{4.114}$$

Nous pouvons établir les rélations suivantes :

$$\sum_{i=1}^{\lfloor \tilde{c}t \rfloor} r_i = \sum_{i=1}^{\lceil \alpha t \rceil} r_i + \sum_{i=\lceil \alpha t \rceil + 1}^{\lfloor \tilde{c}t \rfloor} r_i \tag{4.115}$$

$$= \sum_{i=1}^{\lceil \alpha t \rceil} \frac{1}{t} + \sum_{i=\lceil \alpha t \rceil + 1}^{\lfloor \tilde{c}t \rfloor} \frac{1}{t} \left(1 - \frac{1}{t}\right)^{i-1} \tag{4.116}$$

$$= \frac{\lceil \alpha t \rceil}{t} + \frac{1}{t} \sum_{i=\lceil \alpha t \rceil}^{\lfloor \tilde{c}t \rfloor - 1} \left(1 - \frac{1}{t}\right)^{i} \tag{4.117}$$

$$= \frac{\lceil \alpha t \rceil}{t} + \frac{1}{t} \left(\frac{(1 - \frac{1}{t})^{\lfloor \tilde{c}t \rfloor}}{(1 - \frac{1}{t}) - 1} - \frac{(1 - \frac{1}{t})^{\lceil \alpha t \rceil}}{(1 - \frac{1}{t}) - 1} \right) \tag{4.118}$$

$$= \frac{\lceil \alpha t \rceil}{t} + \left(1 - \frac{1}{t}\right)^{\lceil \alpha t \rceil} - \left(1 - \frac{1}{t}\right)^{\lfloor \tilde{c}t \rfloor} \tag{4.119}$$

$$\leq \frac{\lceil \alpha t \rceil}{t} + \left(1 - \frac{1}{t}\right)^{\lceil \alpha t \rceil} - \left(1 - \frac{1}{t}\right)^{\tilde{c}t} \tag{4.120}$$

$$= 1 \quad \text{par (4.97)} \tag{4.121}$$

Ainsi $< r_i >$ est bien une distribution de probabilité. □

Rappelons que nous sommes toujours dans le cas où ϕ est δ-satisfaisable. Par les propriétés de l'hypergraphe $H(\phi)$, on sait que pour $1 \leq \ell \leq \lfloor ct \rfloor$ nous avons

$$\sum_{i=1}^{\ell} p_{e_i} \leq \frac{\ell}{t} \quad \text{et} \quad \sum_{i=1}^{\ell} p_{e_i} \leq 1 - \left(1 - \frac{1}{t}\right)^{\ell} + \lambda \tag{4.122}$$

et de plus $\tilde{c} \leq c$ pour t assez grand. En effet, les lemmes 4.9, 4.10, et 4.12 nous permettent d'écrire la suite d'inégalités suivante :

$$\tilde{c} = \frac{\ln\left(\frac{a}{t} + (1-\frac{1}{t})^a - 1\right)}{t \cdot \ln(1-\frac{1}{t})} \tag{4.123}$$

$$\leq -\ln\left(\frac{a}{t} + \left(1-\frac{1}{t}\right)^a - 1\right) \tag{4.124}$$

$$\leq -\ln\left(\frac{\alpha t}{t} + \left(1-\frac{1}{t}\right)^{\alpha t+1} - 1\right) \tag{4.125}$$

$$= -\ln\left(\alpha + \left(1-\frac{1}{t}\right)\left(\left(1-\frac{1}{t}\right)^t\right)^\alpha - 1\right) \tag{4.126}$$

$$\longrightarrow_{t\to+\infty} -\ln(\alpha + e^{-\alpha} - 1) \tag{4.127}$$

$$< -\ln\left(\frac{\alpha^2}{2} - \frac{\alpha^3}{6}\right) \tag{4.128}$$

$$= -\ln(\lambda) \tag{4.129}$$

$$= c \tag{4.130}$$

Notons que nous avons de façon similaire :

$$\tilde{c} = \frac{\ln\left(\frac{a}{t} + (1-\frac{1}{t})^a - 1\right)}{t \cdot \ln(1-\frac{1}{t})} \tag{4.131}$$

$$\geq -\frac{t-1}{t}\ln\left(\frac{a}{t} + \left(1-\frac{1}{t}\right)^a - 1\right) \tag{4.132}$$

$$\geq -\frac{t-1}{t}\ln\left(\frac{\alpha t+1}{t} + \left(1-\frac{1}{t}\right)^{\alpha t} - 1\right) \tag{4.133}$$

$$\geq -\frac{t-1}{t}\ln\left(\alpha + \frac{1}{t} + e^{-\alpha} - 1\right) \tag{4.134}$$

$$\longrightarrow_{t\to+\infty} -\ln(\alpha + e^{-\alpha} - 1) \tag{4.135}$$

Avec (4.127) on peut en déduire que

$$\lim_{t\to+\infty} \tilde{c} = -\ln(\alpha + e^{-\alpha} - 1) \tag{4.136}$$

Affirmation 4.20 *La distribution $< r_i >$ domine la distribution $< p_{e_i} >$. C'est-à-dire que pour tout $1 \leq \ell \leq \lfloor ct \rfloor$ nous avons*

$$\sum_{i=1}^{\ell} p_{e_i} \leq \sum_{i=1}^{\ell} r_i \tag{4.137}$$

Démonstration 4.20 Pour $1 \leq \ell \leq \lceil \alpha t \rceil$ l'inégalité découle de la définition des $< r_i >$ et de (4.122). Notons qu'en prenant t suffisamment grand nous avons

$$1 - \left(1 - \frac{1}{t}\right)^{\lceil \alpha t \rceil} + \lambda \leq 1 - \left(1 - \alpha + \frac{\alpha^2}{2} - \frac{\alpha^3}{6}\right) + \lambda = \alpha \leq \frac{\lceil \alpha t \rceil}{t} \tag{4.138}$$

Pour s'en persuader, montrons que pour un α bien choisi et pour t assez grand nous avons

$$\left(1 - \frac{1}{t}\right)^{\lceil \alpha t \rceil} \geq 1 - \alpha + \frac{\alpha^2}{2} - \frac{\alpha^3}{6} \tag{4.139}$$

Cela s'établit de la façon suivante :

$$\left(1 - \frac{1}{t}\right)^{\lceil \alpha t \rceil} \geq \left(1 - \frac{1}{t}\right)^{\alpha t + 1} \tag{4.140}$$

$$\geq e^{-\frac{\alpha t}{t-1}} \left(1 - \frac{1}{t}\right) \quad \text{par le lemme 4.10} \tag{4.141}$$

$$\longrightarrow_{t \to +\infty} e^{-\alpha} \tag{4.142}$$

$$> 1 - \alpha + \frac{\alpha^2}{2} - \frac{\alpha^3}{6} \quad \text{déjà montré en (4.79)} \tag{4.143}$$

Ainsi pour le cas $\lceil \alpha t \rceil + 1 \leq \ell \leq \lfloor \tilde{c} t \rfloor$ les résultats (4.122) et (4.138) ainsi que la définition des r_i nous livrent

$$\sum_{i=1}^{\ell} p_{e_i} \leq 1 - \left(1 - \frac{1}{t}\right)^{\ell} + \lambda \tag{4.144}$$

$$= 1 - \left(1 - \frac{1}{t}\right)^{\lceil \alpha t \rceil} + \lambda + \sum_{i=\lceil \alpha t \rceil + 1}^{\ell} \frac{1}{t} \left(1 - \frac{1}{t}\right)^{i-1} \tag{4.145}$$

$$\leq \frac{\lceil \alpha t \rceil}{t} + \sum_{i=\lceil \alpha t \rceil + 1}^{\ell} \frac{1}{t} \left(1 - \frac{1}{t}\right)^{i-1} \tag{4.146}$$

$$= \sum_{i=1}^{\ell} r_i \tag{4.147}$$

La distribution $< p_{e_i} >$ est donc bien dominée par la distribution $< r_i >$. □

Le lemme 4.3 nous apprend alors que l'entropie de la distribution $< p_{e_i} >$ est supérieure ou égale à l'entropie de la distribution $< r_i >$. Pour terminer de montrer l'affirmation (4.94) il suffit alors de prouver que

$$ENT(< r_i >) \geq \log_2(t) + \left(1 - \frac{\epsilon}{2}\right) \log_2(e) + o(1) \tag{4.148}$$

Montrons cela. On a successivement :

$$ENT(< r_i >) \tag{4.149}$$

$$= - \sum_{i=1}^{\lfloor \tilde{c}t \rfloor + 1} r_i \log_2(r_i) \tag{4.150}$$

$$= - \sum_{i=1}^{\lfloor \tilde{c}t \rfloor} r_i \log_2(r_i) + o(1) \tag{4.151}$$

$$= - \frac{\lceil \alpha t \rceil}{t} \log_2 \left(\frac{1}{t} \right) - \sum_{i=\lceil \alpha t \rceil + 1}^{\lfloor \tilde{c}t \rfloor} \frac{1}{t} \left(1 - \frac{1}{t} \right)^{i-1} \log_2 \left(\frac{1}{t} \left(1 - \frac{1}{t} \right)^{i-1} \right) + o(1) \tag{4.152}$$

$$= \alpha \cdot \log_2(t) - \frac{1}{t} \sum_{i=\lceil \alpha t \rceil + 1}^{\lfloor \tilde{c}t \rfloor} \left(1 - \frac{1}{t} \right)^{i-1} \left((i-1) \log_2 \left(1 - \frac{1}{t} \right) - \log_2(t) \right) + o(1) \tag{4.153}$$

$$\begin{aligned} = & \alpha \cdot \log_2(t) - \frac{1}{t} \log_2 \left(1 - \frac{1}{t} \right) \sum_{i=\lceil \alpha t \rceil + 1}^{\lfloor \tilde{c}t \rfloor} (i-1) \left(1 - \frac{1}{t} \right)^{i-1} \\ & + \frac{1}{t} \log_2(t) \sum_{i=\lceil \alpha t \rceil + 1}^{\lfloor \tilde{c}t \rfloor} \left(1 - \frac{1}{t} \right)^{i-1} + o(1) \end{aligned} \tag{4.154}$$

$$\begin{aligned} = & \alpha \cdot \log_2(t) + \frac{1}{t} \log_2 \left(\frac{t}{t-1} \right) \sum_{i=\lceil \alpha t \rceil + 1}^{\lfloor \tilde{c}t \rfloor} (i-1) \left(1 - \frac{1}{t} \right)^{i-1} \\ & + \frac{1}{t} \log_2(t) \sum_{i=\lceil \alpha t \rceil + 1}^{\lfloor \tilde{c}t \rfloor} \left(1 - \frac{1}{t} \right)^{i-1} + o(1) \end{aligned} \tag{4.155}$$

En se rappelant du résultat obtenu en (4.136), on pose $\beta = \lim_{t \to +\infty} \tilde{c} = -\ln(\alpha + e^{-\alpha} - 1)$.

Affirmation 4.21

$$\frac{1}{t} \log_2 \left(\frac{t}{t-1} \right) \sum_{i=\lceil \alpha t \rceil + 1}^{\lfloor \tilde{c}t \rfloor} (i-1) \left(1 - \frac{1}{t} \right)^{i-1} = \log_2(e)((1+\alpha)e^{-\alpha} - (1+\beta)e^{-\beta}) + o(1) \tag{4.156}$$

Affirmation 4.22

$$\frac{1}{t} \log_2(t) \sum_{i=\lceil \alpha t \rceil + 1}^{\lfloor \tilde{c}t \rfloor} \left(1 - \frac{1}{t} \right)^{i-1} = \log_2(t)(e^{-\alpha} - e^{-\beta}) + o(1) \tag{4.157}$$

Démonstration 4.21 De façon générale, nous avons :

$$\sum_{i=1}^{\ell}(i-1)\left(1-\frac{1}{t}\right)^{i-1} \tag{4.158}$$

$$=\sum_{k=1}^{\ell-1}\sum_{i=k}^{\ell-1}\left(1-\frac{1}{t}\right)^{i} \tag{4.159}$$

$$=\sum_{k=1}^{\ell-1}\left(-t\left(1-\frac{1}{t}\right)^{\ell}+t\left(1-\frac{1}{t}\right)^{k}\right) \tag{4.160}$$

$$=-(\ell-1)t\left(1-\frac{1}{t}\right)^{\ell}+t\sum_{k=1}^{\ell-1}\left(1-\frac{1}{t}\right)^{k} \tag{4.161}$$

$$=-\ell t\left(1-\frac{1}{t}\right)^{\ell}+t\left(1-\frac{1}{t}\right)^{\ell}+t\left(-t\left(1-\frac{1}{t}\right)^{\ell}+t\left(1-\frac{1}{t}\right)\right) \tag{4.162}$$

$$=-t^2\ell\frac{\ell}{t}\left(1-\frac{1}{t}\right)^{\ell}+t\left(1-\frac{1}{t}\right)^{\ell}-t^2\left(1-\frac{1}{t}\right)^{\ell}+t^2-t \tag{4.163}$$

$$=t^2\left(1-\left(1-\frac{1}{t}\right)^{\ell}-\frac{\ell}{t}\left(1-\frac{1}{t}\right)^{\ell}\right)-t\left(1-\left(1-\frac{1}{t}\right)^{\ell}\right) \tag{4.164}$$

$$=t^2\left(1-\left(1+\frac{\ell}{t}\right)\left(1-\frac{1}{t}\right)^{\ell}\right)-t\left(1-\left(1-\frac{1}{t}\right)^{\ell}\right) \tag{4.165}$$

Ainsi, nous pouvons réécrire le membre gauche de notre égalité :

$$\frac{1}{t}\log_2\left(\frac{t}{t-1}\right)\sum_{i=\lceil\alpha t\rceil}^{\lfloor\tilde{c}t\rfloor}(i-1)\left(1-\frac{1}{t}\right)^{i-1} \tag{4.166}$$

$$=\frac{1}{t}\log_2\left(\frac{t}{t-1}\right)\left(\sum_{i=1}^{\lfloor\tilde{c}t\rfloor}(i-1)\left(1-\frac{1}{t}\right)^{i-1}-\sum_{i=1}^{\lceil\alpha t\rceil}(i-1)\left(1-\frac{1}{t}\right)^{i-1}\right) \tag{4.167}$$

$$\begin{aligned}=&\frac{1}{t}\log_2\left(\frac{t}{t-1}\right)\left[t^2\left(1-\left(1+\frac{\lfloor\tilde{c}t\rfloor}{t}\right)\left(1-\frac{1}{t}\right)^{\lfloor\tilde{c}t\rfloor}\right)-t\left(1-\left(1-\frac{1}{t}\right)^{\lfloor\tilde{c}t\rfloor}\right)\right.\\&\left.-t^2\left(1-\left(1+\frac{\lceil\alpha t\rceil}{t}\right)\left(1-\frac{1}{t}\right)^{\lceil\alpha t\rceil}\right)+t\left(1-\left(1-\frac{1}{t}\right)^{\lceil\alpha t\rceil}\right)\right]\end{aligned} \tag{4.168}$$

$$\begin{aligned}=&t\log_2\left(\frac{t}{t-1}\right)\left[\left(1+\frac{\lceil\alpha t\rceil}{t}\right)\left(1-\frac{1}{t}\right)^{\lceil\alpha t\rceil}-\left(1+\frac{\lfloor\tilde{c}t\rfloor}{t}\right)\left(1-\frac{1}{t}\right)^{\lfloor\tilde{c}t\rfloor}\right]\\&+\log_2\left(\frac{1}{t-1}\right)\left[\left(1-\frac{1}{t}\right)^{\lfloor\tilde{c}t\rfloor}-\left(1-\frac{1}{t}\right)^{\lfloor\tilde{c}t\rceil}\right]\end{aligned} \tag{4.169}$$

D'autre part, le lemme 4.10 et le lemme 4.13 nous apprennent que

$$\lim_{t\to+\infty}\left(1-\frac{1}{t}\right)^{\tilde{c}t}=\lim_{t\to+\infty}\left(1-\frac{1}{t}\right)^{\tilde{c}t-1}=e^{-\beta} \tag{4.170}$$

et comme $(1-\frac{1}{t})^{\tilde{c}t}\leq(1-\frac{1}{t})^{\lfloor\tilde{c}t\rfloor}\leq(1-\frac{1}{t})^{\tilde{c}t-1}$, nous avons les égalités

$$\left(1-\frac{1}{t}\right)^{\lfloor\tilde{c}t\rfloor}=e^{-\beta}+o(1)\quad\text{et}\quad\left(1-\frac{1}{t}\right)^{\lceil\alpha t\rceil}=e^{-\alpha}+o(1) \tag{4.171}$$

la seconde égalité étant obtenue de façon similaire à la première. Lorsque l'on fait tendre t vers $+\infty$, le premier terme de (4.169) tend vers $\log_2(e)((1+\alpha)e^{-\alpha}-(1+\beta)e^{-\beta})$, et le second terme tend vers $\log_2(1)(e^{-\beta}-e^{-\alpha})$ qui est nul. Ce qui montre l'affirmation (4.156). □

Démonstration 4.22 Selon le même développement que nous avons utilisé pour passer de (4.116) à (4.119), nous avons

$$\frac{1}{t}\log_2(t)\sum_{\lceil\alpha t\rceil}^{\lfloor\tilde{c}t\rfloor}\left(1-\frac{1}{t}\right)^{i-1}=\log_2(t)\left(\left(1-\frac{1}{t}\right)^{\lceil\alpha t\rceil}-\left(1-\frac{1}{t}\right)^{\lfloor\tilde{c}t\rfloor}\right) \tag{4.172}$$

En se rappelant du résultat obtenu en (4.171), on voit qu'il suffit de montrer que

$$\log_2(t)\left(e^{-\alpha}-\left(1-\frac{1}{t}\right)^{\lceil\alpha t\rceil}\right)=o(1)\quad\text{et}\quad\log_2(t)\left(e^{-\beta}-\left(1-\frac{1}{t}\right)^{\lfloor\tilde{c}t\rfloor}\right)=o(1) \tag{4.173}$$

De façon similaire à ce que nous avons déjà fait à plusieurs reprises, il suffit de montrer que

$$\log_2(t)\left(e^{-\alpha}-\left(1-\frac{1}{t}\right)^{\alpha t}\right)=o(1)\quad\text{et}\quad\log_2(t)\left(e^{-\beta}-\left(1-\frac{1}{t}\right)^{\tilde{c}t}\right)=o(1) \tag{4.174}$$

Les deux cas se prouvant de manière semblable, nous nous contenterons de prouver le premier. Nous obtenons le résultat en démontrant que

$$\lim_{t\to+\infty}\left[\log_2(t)\left(e^{-\alpha}-\left(1-\frac{1}{t}\right)^{\alpha t}\right)\right]=\lim_{t\to+\infty}\left[\log_2\left(t^{e^{-\alpha}-\left(1-\frac{1}{t}\right)^{\alpha t}}\right)\right]=0 \tag{4.175}$$

c'est-à-dire en montrant que

$$\lim_{t\to+\infty}\left[t^{e^{-\alpha}-\left(1-\frac{1}{t}\right)^{\alpha t}}\right]=\lim_{t\to+\infty}\left[\left(t^{\frac{1}{t}}\right)^{t\left(e^{-\alpha}-\left(1-\frac{1}{t}\right)^{\alpha t}\right)}\right] \tag{4.176}$$

$$=\left(\lim_{t\to+\infty}\left[t^{\frac{1}{t}}\right]\right)^{\left(\lim_{t\to+\infty}\left[t\left(e^{-\alpha}-\left(1-\frac{1}{t}\right)^{\alpha t}\right)\right]\right)} \tag{4.177}$$

$$=1 \tag{4.178}$$

Il est facile de montrer que $t^{\frac{1}{t}}$ tend vers 1 lorsque t tend vers $+\infty$. En effet, il suffit de voir que $\ln(t^{\frac{1}{t}}) = \frac{1}{t}\ln(t)$ tend vers 0. Les lemmes 4.14 et 4.11 nous apprennent que la limite en exposant dans l'expression (4.177) existe dans $\mathbb{R}^+$, ce qui prouve (4.178) et termine de montrer l'affirmation (4.157). □

En plus de ces deux affirmations , rappelons que $\beta = -\ln(\alpha + e^{-\alpha} - 1)$, c'est-à-dire que $\alpha + e^{-\alpha} - e^{-\beta} = 1$. Nous pouvons ainsi écrire notre entropie

$$ENT(< r_i >) \tag{4.179}$$
$$= \alpha \cdot \log_2(t) + \log_2(e)((1+\alpha)e^{-\alpha} - (1+\beta)e^{-\beta}) + \log_2(t)(e^{-\alpha} - e^{-\beta}) + o(1) \tag{4.180}$$
$$= (\alpha + e^{-\alpha} - e^{-\beta})\log_2(t) + ((1+\alpha)e^{-\alpha} - (1+\beta)e^{-\beta})\log_2(e) + o(1) \tag{4.181}$$
$$= log_2(t) + ((1+\alpha)e^{-\alpha} - (1+\beta)e^{-\beta})log_2(e) + o(1) \tag{4.182}$$

Affirmation 4.23 *Si on choisit ϵ assez petit, on a*

$$\alpha e^{-\alpha} - \beta e^{-\beta} \geq 0 \tag{4.183}$$

Démonstration 4.23 On a vu respectivement en (4.79) et (4.81) que

$$\frac{\alpha^2}{2} - \frac{\alpha^3}{6} < \alpha + e^{-\alpha} - 1 \quad \text{et} \quad \frac{\alpha^2}{2} > \alpha + e^{-\alpha} - 1 \tag{4.184}$$

ce qui implique aussi

$$-\ln\left(\frac{\alpha^2}{2} - \frac{\alpha^3}{6}\right)\frac{\alpha^2}{2} > -\ln(\alpha + e^{-\alpha} - 1)(\alpha + e^{-\alpha} - 1) \tag{4.185}$$

D'autre part, par intégrations successives sur ses deux membres et en observant les égalités des limites en 0, on peut montrer l'inégalité suivante, pour α assez petit :

$$\frac{6}{\alpha^2}e^{-\frac{1}{\alpha}} < 3 - \alpha \tag{4.186}$$

Il s'ensuit les inégalités successives :

$$e^{-\frac{1}{\alpha}} < \frac{\alpha^2}{2} - \frac{\alpha^3}{6} \tag{4.187}$$
$$-\frac{1}{\alpha} < \ln\left(\frac{\alpha^2}{2} - \frac{\alpha^3}{6}\right) \tag{4.188}$$
$$1 > -\ln\left(\frac{\alpha^2}{2} - \frac{\alpha^3}{6}\right)\alpha \tag{4.189}$$
$$\frac{\alpha}{2} > -\ln\left(\frac{\alpha^2}{2} - \frac{\alpha^3}{6}\right)\frac{\alpha^2}{2} \tag{4.190}$$

De plus, il est évident que pour α assez petit on a $e^{-\alpha} > \frac{1}{2}$ et donc $\alpha e^{-\alpha} > \frac{\alpha}{2}$. Les résultats (4.184), (4.185), (4.190) et ce dernier argument fournissent alors

$$\alpha e^{-\alpha} > \frac{\alpha}{2} > -\ln\left(\frac{\alpha^2}{2} - \frac{\alpha^3}{6}\right)\frac{\alpha^2}{2} > -\ln(\alpha + e^{-\alpha} - 1)(\alpha + e^{-\alpha} - 1) = \beta e^{-\beta} \quad (4.191)$$

ce qui prouve l'affirmation (4.183). □

On trouve alors

$$(1+\alpha)e^{-\alpha} - (1+\beta)e^{-\beta} = 1 - \alpha + \alpha e^{-\alpha} - \beta e^{-\beta} \quad (4.192)$$

$$\geq 1 - \alpha \quad (4.193)$$

$$= 1 - \frac{\epsilon}{2} \quad (4.194)$$

ce qui implique l'affirmation (4.148), et qui implique donc le théorème. □

Chapitre 5

Problème de Couverture d'ensemble de somme des carrés maximum

Un fermier possède différentes barrières dont certaines sont de types différents, certaines barrières étant néanmoins compatibles les unes avec les autres. Il souhaite, à l'aide de ces barrières, créer différents enclos (chaque enclos devant être constitué uniquement des barrières compatibles) de façon à ce que la surface totale des enclos soit maximale.

5.1 Présentation du problème

Comme pour chacun des problèmes précédents, on considère un hypergraphe $H(V, E)$. Soit $f : E \to \{1, 2, \ldots, |E|\} : e \mapsto f(e)$ une organisation linéaire des hyperarêtes de E. Si v est un sommet quelconque de V, on définit $E_v = \{e \in E | v \in e\}$ l'ensemble des hyperarêtes de E contenant v et on définit $f(v) = \min_{e \in E_v} f(e)$. Posons $C_e = \{v \in V : f(v) = f(e)\}$ l'ensemble des sommets couverts par e. Nous avons $\sum_{e \in E} |C_e| = |V|$, et $\frac{|C_e|}{|V|}$ représente la probabilité qu'un sommet quelconque v soit couvert par l'hyperarête e, sous l'hypothèse d'une distribution uniforme des sommets. On note $p_e = \frac{|C_e|}{|V|}$.

Nous pouvons définir notre fonction objectif

$$F_{obj} : \mathcal{F}_E \to \mathbb{R}^+ : f \mapsto \sum_{e \in E} p_e^2 \tag{5.1}$$

où $\mathcal{F}_E$ est l'ensemble des organisations linéaires sur E, et où les p_e dépendent de f. Cette fois nous tenterons de maximiser la fonction objectif.

Remarque 5.1 *Il est équivalent de vouloir maximiser la somme des carrés du nombre de sommets sélectionnés à chaque étape de l'algorithme. Cela se voit en multipliant les p_e par $|V|$.*

Remarque 5.2 *L'exemple donné en introduction est une instance du problème de Couverture d'ensemble de somme des carrés maximum. Pour nous en assurer, nous avons besoin du résultat suivant :*
Pour tout périmètre p donné, la surface maximale qui peut être obtenue est proportionnelle à ce périmètre.
En effet, pour un périmètre donné 1*, on suppose que l'aire maximale vaut A. Alors en multipliant l'échelle de la figure par p, on obtient un périmètre p et une aire $A \cdot p^2$. L'aire maximale est donc bien proportionnelle au carré du périmètre selon un facteur A. On trouvera plus de précisions sur ce résultat dans* The Dolciani Mathematical Expositions, number six, Maxima and minima without calculus *[12] pages 80 à 85.*

On définit notre hypergraphe de la façon suivante. Chaque sommet représente une barrière, et chaque hyperarête est un ensemble de barrières qui sont toutes compatibles. Lorsque l'algorithme sélectionne une hyperarête e, on forme un enclos avec l'ensemble des barrières de C_e. La surface maximale de l'enclos est alors proportionnelle au carré du périmètre de l'enclos, lui-même proportionnel au nombre de barrières. Par la remarque 5.1 ci-dessus, on obtient le résultat de la remarque.

5.2 Algorithme glouton

L'algorithme glouton pour ce problème reste identique aux précédents. Il se contente, à chaque étape, de sélectionner l'hyperarête couvrant le plus de sommets non encore couverts. Nous rappelons son implémentation, pour une instance du problème d'optimisation « Étant donné un hypergraphe $H(V, E)$, on cherche une organisation linéaire $e_1, e_2, \ldots, e_k$ dans E telle que $\bigcup_{i=1}^{k} e_i = V$ et qui maximise notre fonction objectif ».
Pratiquement :

1. un *ensemble de sommets* $R \leftarrow V$
2. un *ensemble indicé d'hyperarêtes* $D \leftarrow \emptyset$
3. tant que $R \neq \emptyset$ faire :
   (a) sélectionner $e \in E \backslash D$ qui maximise $|R \cap e|$
   (b) ajouter e à D
   (c) supprimer de R les sommets de e qui s'y trouvent
4. retourner D

5.3 Facteur d'approximation

Nous allons montrer que pour ce problème d'optimisation, l'algorithme glouton fournit une solution au pire cas deux fois inférieure à une solution optimale, et que cette borne est atteinte asymptotiquement par l'algorithme glouton. Comme dans les problèmes précédents, on définit f_G l'organisation linéaire sélectionnée par l'algorithme glouton, et

f_{OPT} une organisation linéaire optimale. On rappelle C_e l'ensemble des sommets qui seront couverts lors de l'étape où l'algorithme glouton sélectionne e, et soit X_e l'ensemble des sommets qui seront couverts lors de l'étape où l'algorithme optimal sélectionne e, où on pose $x_e = |X_e|$. On note e_v l'hyperarête telle que $v \in C_{e_v}$, c'est-à-dire l'hyperarête de E qui couvrira le sommet v.

Théorème 5.3 *Si OPT est la valeur de la somme des carrés renvoyée par une solution optimale et ALGO la valeur associée à la solution de l'algorithme glouton, alors*

$$ALGO \geq \frac{OPT}{2} \tag{5.2}$$

Démonstration 5.3 Notons $X_e = \{v_1, v_2, \ldots, v_{x_e}\}$ où les v_i sont dans l'ordre dans lequel ils seront couverts par l'algorithme glouton ; on les choisit de façon arbitraire lorsqu'ils sont dans la même hyperarête sélectionnée par l'algorithme (c'est-à-dire si $C_{e_{v_i}} = C_{e_{v_j}}$). Considérons un v_j, l'algorithme glouton le couvre avec l'hyperarête e_{v_j}. À cet instant, au plus $j-1$ sommets de X_e sont couverts, ce qui signifie qu'au moins $x_e - j + 1$ sommets ne sont pas encore couverts dans X_e. L'algorithme glouton étant celui qui, à chaque étape, sélectionne l'hyperarête possédant le plus de sommets non encore couverts, et cette hyperarête étant ici e_{v_j}, on sait qu'au moins $x_e - j + 1$ des sommets de e_{v_j} ne sont pas couverts, sans quoi l'algorithme glouton aurait sélectionné X_e. Cela signifie que

$$|C_{e_{v_j}}| \geq |X_e| - j + 1 \tag{5.3}$$

En posant $n = |V|$, on en déduit la suite d'égalités et d'inégalités suivante :

$$ALGO = \sum_{e \in E} \left(\frac{|C_e|}{n} \right)^2 \tag{5.4}$$

$$= \frac{1}{n^2} \sum_{v \in V} |C_{e_v}| \tag{5.5}$$

$$= \frac{1}{n^2} \sum_{e \in E} \sum_{v \in X_e} |C_{e_v}| \tag{5.6}$$

$$= \frac{1}{n^2} \sum_{e \in E} \sum_{j=1}^{|X_e|} |C_{e_{v_j}}| \quad \text{où les } v_j \text{ sont différents pour des } e \text{ différents} \tag{5.7}$$

$$\geq \frac{1}{n^2} \sum_{e \in E} \sum_{j=1}^{|X_e|} (|X_e| - j + 1) \tag{5.8}$$

$$= \frac{1}{n^2} \sum_{e \in E} \sum_{j=1}^{|X_e|} j \tag{5.9}$$

$$= \frac{1}{n^2} \sum_{e \in E} \frac{1}{2} |X_e|(|X_e| + 1) \tag{5.10}$$

$$= \frac{1}{2n^2} \sum_{e \in E} |X_e|^2 + \frac{1}{2} \sum_{e \in E} |X_e| \tag{5.11}$$

$$= \frac{1}{2n^2} \sum_{e \in E} |X_e|^2 + \frac{1}{2} |V| \tag{5.12}$$

$$\geq \frac{1}{2} \sum_{e \in E} \left(\frac{|X_e|}{n} \right)^2 \tag{5.13}$$

$$= \frac{OPT}{2} \tag{5.14}$$

□

Théorème 5.4 *Pour tout ϵ strictement positif il existe une instance P du problème de Couverture d'ensemble de somme des carrés maximum, où f_P est l'organisation linéaire obtenue en appliquant l'algorithme glouton à P, telle que*

$$F_{obj}(f_P) \leq \frac{F_{obj}(f_{OPT})}{2 - \epsilon} \tag{5.15}$$

Démonstration 5.4 Pour montrer cela, il suffit de donner un exemple d'un tel P, nous prendrons le même exemple que celui présenté dans la démonstration du théorème 4.8. Considérons donc un ensemble V de $n \cdot n!$ sommets, que l'on représente comme une

matrice

$$V = \{v_{ij} | i \in \{1, \dots, n\}, j \in \{1, \dots, n!\}\} \tag{5.16}$$

Posons $\mathcal{C}_{COL}$ et $\mathcal{C}_{LIG}$ deux collections d'hyperarêtes définies par

$$\mathcal{C}_{COL} = \{\{v_{ij} | i \in \{1, \dots, n\}\} | j \in \{1, \dots, n!\}\} \tag{5.17}$$

$$\mathcal{C}_{LIG} = \{\{v_{ij} | j = qi + r, r \in \{1, \dots, i\}\} | q \in \{1, \dots, \frac{n!}{i}\}, i \in \{1, \dots, n\}\} \tag{5.18}$$

comme dans la figure 5.1. Parmi ces deux collections, dont l'union forme l'ensemble des

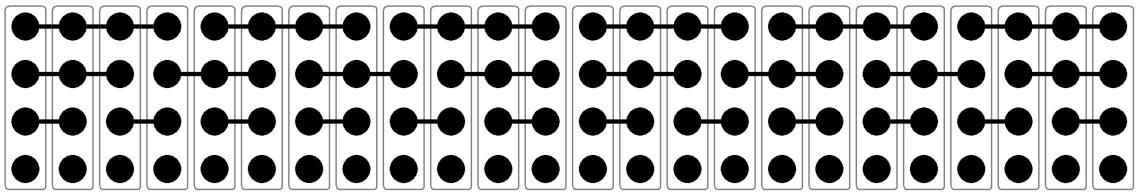

FIG. 5.1 – Représentation de $\mathcal{C}_{COL}$ et $\mathcal{C}_{LIG}$ en n lignes et $n!$ colonnes

hyperarêtes de notre instance, l'algorithme glouton sélectionnera la collection définie par f_{LIG}, et nous pouvons déterminer un algorithme sélectionnant la collection définie par f_{COL}. Calculons $F_{obj}(f_{COL})$ et $F_{obj}(f_{LIG})$.

$$F_{obj}(f_{COL}) = \sum_{k=1}^{n!} \left(\frac{n}{n \cdot n!}\right)^2 \tag{5.19}$$

$$= \frac{1}{n!} \tag{5.20}$$

$$F_{obj}(f_{LIG}) = \sum_{k=1}^{n} \sum_{j=1}^{\frac{n!}{k}} \left(\frac{k}{n \cdot n!}\right)^2 \tag{5.21}$$

$$= \sum_{k=1}^{n} \frac{k}{n^2 \cdot n!} \tag{5.22}$$

$$= \frac{1}{n^2 \cdot n!} \cdot \frac{1}{2} n(n+1) \tag{5.23}$$

$$= \frac{1}{n!} \frac{n+1}{2n} \tag{5.24}$$

Ainsi nous avons

$$\frac{F_{obj}(f_{COL})}{F_{obj}(f_{LIG})} = \frac{\frac{1}{n!}}{\frac{1}{n!}\frac{n+1}{2n}} = \frac{2n}{n+1} \tag{5.25}$$

Ce qui nous donne, pour n assez grand

$$F_{obj}(f_P) = F_{obj}(f_{LIG}) = \frac{n+1}{2n} F_{obj}(f_{COL}) \leq \frac{n+1}{2n} F_{obj}(f_{OPT}) \leq \frac{1}{2-\epsilon} F_{obj}(f_{OPT}) \tag{5.26}$$

□

5.4 Inapproximabilité

Comme dans les deux problèmes précédents, nous allons montrer qu'il n'existe pas d'algorithme polynomial pour le problème de Couverture d'ensemble de somme des carrés maximum fournissant une meilleure approximation que l'algorithme glouton, à moins que $\mathcal{P}$=$\mathcal{NP}$.

Lemme 5.5 *Soient $q =< q_i >$ et $r =< r_i >$ deux distributions de probabilité sur les entiers strictement positifs, dont le support est fini (c'est-à-dire qu'il existe un naturel M tel que pour tout $m \geq M$, les probabilités p_m et q_m sont nulles) et telles que*

$$\sum_{i=1}^{k} r_i \geq \sum_{i=1}^{k} q_i \quad \textit{pour tout naturel } k \tag{5.27}$$

On suppose de plus que $q_i \geq q_{i+1}$ pour tout i.
Alors il existe un naturel N tel que pour tout $n \geq N$ nous avons la relation

$$\sum_{i=1}^{n} q_i^2 \leq \sum_{i=1}^{n} r_i^2 \tag{5.28}$$

Démonstration 5.5 Considérons $p =< p_i >$ une distribution de probabilité sur les entiers strictement positifs, à support fini et décroissante. On choisit deux probabilités de cette distribution p_a et p_b avec $a < b$ (ainsi $p_a \geq p_b$). Soit δ dans $]0, p_b]$. Considérons la distribution $p' =< p'_i >$ où $p'_a = p_a + \delta$, $p'_b = p_b - \delta$ et les autres p'_i restent égaux à p_i. On affirme que

$$\sum_{i=0}^{n} p_i'^2 \geq \sum_{i=0}^{n} p_i^2 \tag{5.29}$$

En d'autres termes, une telle transformation sur une telle distribution augmente sa somme quadratique. Montrons-le :
Nous avons tout naturellement

$$\begin{aligned} p_a'^2 + p_b'^2 &= (p_a + \delta)^2 + (p_b - \delta)^2 && (5.30)\\ &= p_a^2 + 2\delta p_a + \delta^2 + p_b^2 - 2\delta p_b + \delta^2 && (5.31)\\ &= p_a^2 + p_b^2 + 2\delta(p_a - p_b) + 2\delta^2 && (5.32)\\ &\geq p_a^2 + p_b^2 && (5.33) \end{aligned}$$

En ajoutant les termes p_i pour les autres indices différents de a et b on prouve l'affirmation (5.29).
Selon les mêmes arguments que dans la preuve du lemme 4.3, on prouve que l'on peut transformer la distribution $q =< q_i >$ en $r =< r_i >$ avec une suite finie de telles transformations ; ce qui prouve le lemme. □

Lemme 5.6 *Si $t > 1$ est un entier, alors la série*

$$\sum_{k=0}^{\infty} \left(1 - \frac{1}{t}\right)^k \tag{5.34}$$

est uniformément convergente et converge vers t.

Démonstration 5.6 Les résultats classiques de combinatoire nous fournissent la suite d'égalités

$$\sum_{k=0}^{n} \left(1 - \frac{1}{t}\right)^k = \sum_{k=0}^{n} \sum_{j=0}^{k} \binom{k}{j} \left(-\frac{1}{t}\right)^j \tag{5.35}$$

$$= \sum_{j=0}^{n} \sum_{k=j}^{n} \binom{k}{j} \left(-\frac{1}{t}\right)^j \tag{5.36}$$

$$= \sum_{j=0}^{n} \left(-\frac{1}{t}\right)^j \sum_{k=j}^{n} \binom{k}{j} \tag{5.37}$$

$$= \sum_{j=0}^{n} \left(-\frac{1}{t}\right)^j \binom{n+1}{j+1} \tag{5.38}$$

$$= -t \sum_{j=1}^{n+1} \left(-\frac{1}{t}\right)^j \binom{n+1}{j} \tag{5.39}$$

$$= -t \left(\sum_{j=0}^{n+1} \left(-\frac{1}{t}\right)^j \binom{n+1}{j} - 1 \right) \tag{5.40}$$

$$= t \left(1 - \left(1 - \frac{1}{t}\right)^{n+1}\right) \tag{5.41}$$

On voit alors que cette valeur tend vers t lorsque n grandit. □

Lemme 5.7 *Si $t > 1$ est un entier, alors la série*

$$\sum_{k=0}^{\infty} \left(1 - \frac{1}{t}\right)^{2k} \tag{5.42}$$

est uniformément convergente et converge vers $\frac{t^2}{2t-1}$.

Démonstration 5.7 Il s'agit d'une sous-série de la série du lemme 5.6, qui était uniformément convergente, celle-ci l'est donc également. On peut alors poser

$$X = \sum_{k=0}^{\infty} \left(1 - \frac{1}{t}\right)^{2k} \tag{5.43}$$

On en déduit la suite d'égalités

$$X = 1 + \sum_{k=1}^{\infty} \left(1 - \frac{1}{t}\right)^{2k} \tag{5.44}$$

$$= 1 + \left(1 - \frac{1}{t}\right)^{2} \sum_{k=1}^{\infty} \left(1 - \frac{1}{t}\right)^{2k-2} \tag{5.45}$$

$$= 1 + \left(1 - \frac{1}{t}\right)^{2} \sum_{k=0}^{\infty} \left(1 - \frac{1}{t}\right)^{2k} \tag{5.46}$$

$$= 1 + \left(1 - \frac{1}{t}\right)^{2} X \tag{5.47}$$

Cela nous fournit l'équation en X

$$X\left(\frac{2}{t} - \frac{1}{t^2}\right) = 1 \tag{5.48}$$

dont l'unique solution est

$$X = \frac{t^2}{2t-1} \tag{5.49}$$

Ce qui prouve le lemme. □

Avant de présenter le théorème, il faut se rappeler les notions introduites en début de section concernant les instances de 3-SAT-6 satisfaisables et δ-satisfaisables. Il est établi dans *Approximating Min-Sum Set Cover* [6] que pour certains δ dans $]0,1[$, il est $\mathcal{NP}$-difficile de distinguer une instance de 3-SAT-6 satisfaisable d'une instance δ-satisfaisable. Dans *Approximating Min-Sum Set Cover* [6] et *A threshold of ln n for approximating set cover* [7] est montrée une réduction qui à toute instance ϕ de 3-SAT-6 associe un hypergraphe $H(\phi)(V, E)$, qui possède la propriété suivante : Pour tous réels $c > 0$ et $\lambda > 0$ fixés, et pour tout naturel $t > 0$ fixé, il est possible de choisir les paramètres de la réduction de telle façon que :

- Les hyperarêtes de E sont toutes de taille commune $\frac{n}{t}$ où $n = |V|$, et où tous les sommets sont de même degré.
- Si ϕ est satisfaisable, alors V peut être couvert par t hyperarêtes disjointes de E.
- Si ϕ est δ-satisfaisable, chaque collection de ℓ hyperarêtes choisies dans E couvre au plus une proportion $1 - (1 - \frac{1}{t})^{\ell} + \lambda$ des sommets de V, pour tout ℓ avec $1 \leq \ell \leq ct$.

Théorème 5.8 *Pour tout ϵ strictement positif, il est $\mathcal{NP}$-difficile de résoudre le problème de Couverture d'ensemble de somme des carrés minimum avec un rapport de $2 - \epsilon$.*

Démonstration 5.8 Si nous montrons le théorème pour tous les hypergraphes uniformes réguliers, nous aurons le résultat. Considérons une instance ϕ de 3-SAT-6 qui soit satisfaisable ou δ-satisfaisable. On note f_{OPT} une organisation linéaire optimale pour l'instance du problème de Couverture d'ensemble d'entropie minimum définie par l'hypergraphe $H(\phi) = H(\phi)(V, E)$. Pour chaque $e \in E$ on définit $p_e = p_e(f_{OPT}) = \frac{|C_e|}{|V|}$ comme nous l'avons fait dans la section 5.1, où les C_e sont associés à l'organisation linéaire f_{OPT}. On peut indicer les éléments de E $e_1, e_2, \ldots, e_{|E|}$ de telle façon que $p_{e_1} \geq p_{e_2} \geq \cdots \geq p_{e_{|E|}}$. Nous savons que $< p_{e_i} >$ est une distribution de probabilité, ici décroissante.
Considérons le premier cas : ϕ est satisfaisable. Une solution optimale consiste alors à couvrir V avec t hyperarêtes disjointes. Nous obtenons ainsi la valeur de la fonction objectif

$$F_{obj}(f_{OPT}) = \sum_{i=1}^{t} \left(\frac{1}{t}\right)^2 = \frac{1}{t} \tag{5.50}$$

Considérons le second cas : ϕ est δ-satisfaisable. Si on arrive à prouver que

$$\sum_{i=1}^{|E|} p_{e_i}^2 \leq \frac{1}{2 - \frac{\epsilon}{2}} \cdot \frac{1}{t} \tag{5.51}$$

alors tout algorithme A approximant le problème de Couverture d'ensemble de somme des carrés minimum avec un rapport de $2 - \epsilon$ pourra être utilisé pour décider si ϕ est satisfaisable ou δ-satisfaisable. En effet, remarquant que

$$\frac{1}{2 - \frac{\epsilon}{2}} \cdot \frac{1}{t} < \frac{1}{2 - \epsilon} \cdot \frac{1}{t} \tag{5.52}$$

si notre algorithme trouve une solution $x \geq \frac{1}{2-\epsilon}\frac{1}{t}$, il est évident qu'une solution optimale sera plus grande ou égale à x, c'est-à-dire strictement plus grande que $\frac{1}{2-\frac{\epsilon}{2}}\frac{1}{t}$, ainsi l'instance ϕ ne peut pas être δ-satisfaisable mais bien satisfaisable. Si au contraire notre algorithme trouve une solution $x < \frac{1}{2-\epsilon}\frac{1}{t}$ alors la solution optimale ne peut être $\frac{1}{t}$ et ϕ n'est donc pas satisfaisable mais bien δ-satisfaisable. Ceci implique le théorème, sans quoi nous irions à l'encontre du fait établi qu'il est $\mathcal{NP}$-difficile de distinguer une instance de 3-SAT-6 satisfaisable d'une instance δ-satisfaisable.
Il nous faut maintenant montrer l'affirmation (5.51). Le lemme 5.5 et la structure de notre hypergraphe nous apprennent qu'on ne pourra pas trouver meilleur algorithme qu'un hypothétique algorithme qui, après exactement k étapes, aura couvert une proportion $1 - (1 - \frac{1}{t})^k + \lambda$ de sommets. On pose ℓ le nombre d'étapes nécessaires à l'algorithme pour couvrir l'ensemble des sommets. Excepté à la première étape où il couvre $1-(1-\frac{1}{t})+\lambda = \frac{1}{t}+\lambda$ sommets, l'algorithme couvrira, lors de la k-ème étape, exactement $(1 - (1 - \frac{1}{t})^k + \lambda) - (1 - (1 - \frac{1}{t})^{k-1} + \lambda) = \frac{1}{t}(1 - \frac{1}{t})^{k-1}$ sommets. Ainsi, on obtient par le lemme 5.5

$$\sum_{i=1}^{|E|} p_{e_i}^2 \leq \left(\frac{1}{t} + \lambda\right)^2 + \sum_{k=2}^{\ell} \left(\frac{1}{t}\left(1 - \frac{1}{t}\right)^{k-1}\right)^2 \tag{5.53}$$

C'est-à-dire

$$\sum_{i=1}^{|E|} p_{e_i}^2 \leq \sum_{k=0}^{\infty} \left(\frac{1}{t} \left(1 - \frac{1}{t} \right)^k \right)^2 + \frac{2\lambda}{t} + \lambda^2 \tag{5.54}$$

$$= \frac{1}{t^2} \sum_{k=0}^{\infty} \left(1 - \frac{1}{t} \right)^{2k} + \frac{2\lambda}{t} + \lambda^2 \tag{5.55}$$

$$= \frac{1}{t^2} \frac{t^2}{2t-1} + \frac{2\lambda}{t} + \lambda^2 \quad \text{par le lemme 5.7} \tag{5.56}$$

$$= \frac{1}{2t-1} + \frac{2\lambda}{t} + \lambda^2 \tag{5.57}$$

Pour t assez grand et λ assez petit, on a alors le résultat attendu

$$\frac{1}{2t-1} + \frac{2\lambda}{t} + \lambda^2 \leq \frac{1}{2 - \frac{\epsilon}{2}} \cdot \frac{1}{t} \tag{5.58}$$

qui prouve l'affirmation (5.51), et prouve donc le théorème. □

Chapitre 6

Généralisation

6.1 Présentation

Nous allons regarder la formulation des problèmes que nous avons développé dans les chapitres précédents, à l'exception du problème de Couverture d'ensemble de somme minimum qui est différent par l'attention que l'on porte à l'ordre dans lequel l'algorithme sélectionne les hyperarêtes. Nous allons voir que ces trois problèmes, Couverture d'ensemble, Couverture d'ensemble d'entropie minimum et Couverture d'ensemble de somme des carrés maximum, sont en fait trois cas particuliers d'un problème plus général que nous appellerons problème de *Couverture d'ensemble de moyenne maximum*.

Rappelons-nous que pour chacun de ces problèmes nous utilisions les conventions suivantes. Nous considérons un hypergraphe $H(V, E)$ et $f : E \to \{1, 2, \ldots, |E|\} : e \mapsto f(e)$ une organisation linéaire. Si v est un sommet quelconque de V, on définit $E_v = \{e \in E | v \in e\}$ l'ensemble des hyperarêtes de E contenant v et on définit $f(v) = \min_{e \in E_v} f(e)$. On pose alors $C_e = \{v \in V : f(v) = f(e)\}$ l'ensemble des sommets couverts par e. Nous avons $\sum_{e \in E} |C_e| = |V|$, et $p_e = \frac{|C_e|}{|V|}$ représente la probabilité qu'un sommet quelconque v soit couvert par l'hyperarête e, sous l'hypothèse d'une distribution uniforme des sommets. Nous pouvons alors définir notre fonction objectif

$$F_{obj} : \mathcal{F}_E \to \mathbb{R}^+ : f \mapsto F_{obj}(f) \tag{6.1}$$

où $\mathcal{F}_E$ est l'ensemble des organisations linéaires sur E. Nous avons également définit e_v comme l'hyperarête de E telle que $v \in C_{e_v}$, c'est-à-dire l'hyperarête couvrant v. Nous allons nous intéresser aux valeurs $|C_{e_v}|$ que nous noterons ici α_v. Il s'agit intuitivement du nombre de sommets couverts en même temps que v par l'algorithme.

Nous avions défini le problème de Couverture d'ensemble par

$$\textit{minimiser} \quad \sum_{e \in \mathcal{C}} w(e) \quad \text{où } w(e) = 1 \text{ dans le cas non pondéré} \tag{6.2}$$

En décidant que $\frac{|C_e|}{|C_e|} = 0$ lorsque $C_e = \emptyset$, le problème de Couverture d'ensemble non

pondéré est équivalent à, successivement

$$minimiser \quad \sum_{e \in E} \frac{|C_e|}{|C_e|} \tag{6.3}$$

$$minimiser \quad \sum_{v \in V} \frac{1}{|C_{e_v}|} = \sum_{v \in V} \frac{1}{\alpha_v} \tag{6.4}$$

$$maximiser \quad \frac{|V|}{\sum_{v \in V} \frac{1}{\alpha_v}} \tag{6.5}$$

c'est-à-dire maximiser la moyenne harmonique des α_v.

Proposition 6.1 *Si ALGO est la moyenne harmonique de l'algorithme glouton et OPT la moyenne harmonique d'un algorithme optimal pour le problème de Couverture d'ensemble de moyenne harmonique maximum, alors nous avons*

$$\frac{ALGO}{OPT} \geq \frac{1}{1 + \ln(|V|)} \tag{6.6}$$

Démonstration 6.1 Nous savons que le problème initial

$$minimiser \quad \sum_{e \in \mathcal{C}} w(e) = \sum_{v \in V} \frac{1}{\alpha_v} \tag{6.7}$$

est approximé par l'algorithme glouton selon les relations

$$ALGO \leq H(d*)OPT \leq H(|V|)OPT \leq (1 + \ln(|V|))OPT \tag{6.8}$$

Ainsi le problème équivalent

$$maximiser \quad \frac{|V|}{\sum_{v \in V} \frac{1}{\alpha_v}} \tag{6.9}$$

bénéficie du rapport d'approximation

$$ALGO \geq \frac{OPT}{1 + \ln(|V|)} \tag{6.10}$$

ce qui prouve la proposition. □

Nous avions défini le problème de Couverture d'ensemble d'entropie minimum par

$$minimiser \quad \sum_{e \in E} |C_e| \log_2(|C_e|) \tag{6.11}$$

où $|C_e| \log_2(|C_e|)$ est défini à 0 lorsque $C_e = \emptyset$. Nous avions vu également que cette formulation est équivalente à

$$maximiser \quad \prod_{e \in E} |C_e|^{|C_e|} \tag{6.12}$$

où $|C_e|^{|C_e|}$ est défini à 1 lorsque $C_e = \emptyset$. Ainsi, nous avons équivalence avec les problèmes

$$maximiser \quad \prod_{v \in V} |C_{e_v}| = \prod_{v \in V} \alpha_v \tag{6.13}$$

$$maximiser \quad \left(\prod_{v \in V} \alpha_v \right)^{\frac{1}{|V|}} \tag{6.14}$$

c'est-à-dire maximiser la moyenne géométrique des α_v.

Proposition 6.2 *Si ALGO est la moyenne géométrique de l'algorithme glouton et OPT la moyenne géométrique d'un algorithme optimal pour le problème de Couverture d'ensemble de moyenne géométrique maximum, alors nous avons*

$$\frac{ALGO}{OPT} \geq \frac{1}{e} \tag{6.15}$$

Démonstration 6.2 Nous savons que le problème initial

$$minimiser \quad -\sum_{e \in E} \frac{|C_e|}{|V|} \log_2 \left(\frac{|C_e|}{|V|} \right) = -\frac{1}{|V|} \log_2 \left(\prod_{e \in E} |C_e|^{|C_e|} \right) + \log_2(|V|) \tag{6.16}$$

est approximé par l'algorithme glouton selon les relations

$$ALGO \leq OPT + \log_2(e) \tag{6.17}$$

Ainsi le problème équivalent

$$maximiser \quad \log_2 \left(\prod_{e \in E} |C_e|^{|C_e|} \right) \tag{6.18}$$

est résolu par l'algorithme glouton selon le rapport

$$ALGO \geq OPT - |V| \log_2(e) \tag{6.19}$$

Le problème suivant

$$maximiser \quad \prod_{e \in E} |C_e|^{|C_e|} \tag{6.20}$$

est donc résolu selon le rapport

$$ALGO \geq \frac{OPT}{e^{|V|}} \tag{6.21}$$

Cela fournit le résultat en exposant le tout par $\frac{1}{|V|}$. □

Et enfin, nous avions défini le problème de Couverture d'ensemble de somme des carrés maximum par

$$maximiser \quad \sum_{e \in E} |C_e|^2 \tag{6.22}$$

ce qui est équivalent à, successivement

$$maximiser \quad \sum_{v \in V} |C_{e_v}| = \sum_{v \in V} \alpha_v \tag{6.23}$$

$$maximiser \quad \frac{1}{|V|} \sum_{v \in V} \alpha_v \tag{6.24}$$

c'est-à-dire maximiser la moyenne arithmétique des α_v. Dans ce cas-ci les modifications de formulation ne modifient pas le rapport d'approximation, ainsi

Proposition 6.3 *Si ALGO est la moyenne arithmétique de l'algorithme glouton et OPT la moyenne arithmétique d'un algorithme optimal pour le problème de Couverture d'ensemble de moyenne arithmétique maximum, alors nous avons*

$$\frac{ALGO}{OPT} \geq \frac{1}{2} \tag{6.25}$$

6.2 Moyenne généralisée

Soient $a_1, a_2, \ldots, a_n$ des réels strictement positifs et soit r un réel non nul. On appelle moyenne $\mathcal{M}_r$ des a_i la valeur

$$\mathcal{M}_r(< a_i >_{i=1,\ldots,n}) = \left(\frac{1}{n} \sum_{k=1}^{n} a_k^r \right)^{\frac{1}{r}} \tag{6.26}$$

Pour $r = 0$, on définit la moyenne $\mathcal{M}_r$ des a_i par la valeur

$$\mathcal{M}_r(< a_i >_{i=1,\ldots,n}) = \left(\prod_{k=1}^{n} a_k \right)^{\frac{1}{n}} \tag{6.27}$$

Théorème 6.4 *Soient $a_1, a_2, \ldots, a_n$ des réels strictement positifs et soient $r < s$ des réels. Alors*

$$\mathcal{M}_r(< a_i >_{i=1,\ldots,n}) \leq \mathcal{M}_s(< a_i >_{i=1,\ldots,n}) \tag{6.28}$$

avec égalité si et seulement si tous les a_i sont égaux. De plus, nous avons les limites

$$\lim_{t \to -\infty} \mathcal{M}_t(< a_i >_{i=1,\ldots,n}) = \min(< a_i >_{i=1,\ldots,n}) \tag{6.29}$$

$$\lim_{t \to +\infty} \mathcal{M}_t(< a_i >_{i=1,\ldots,n}) = \max(< a_i >_{i=1,\ldots,n}) \tag{6.30}$$

On est naturellement tenté d'énoncer le problème sous sa forme généralisée, que nous appellerons *problème de Couverture d'ensemble de moyenne maximum*. Il s'agit, pour les mêmes notations que précédemment et pour r un réel non nul, de

$$maximiser \quad \left(\frac{1}{|V|} \sum_{v \in V} |C_{e_v}|^r \right)^{\frac{1}{r}} \tag{6.31}$$

Notons qu'il est équivalent de

$$maximiser \quad \sum_{v \in V} |C_{e_v}|^r \tag{6.32}$$

c'est-à-dire

$$maximiser \quad \sum_{e \in E} p_e^{r+1} \tag{6.33}$$

Nous avons un corollaire immédiat du théorème :

Corollaire 6.5 *Soient $r < s$ des réels, et soient $ALGO_r$ et $ALGO_s$ les valeurs renvoyées par l'algorithme glouton pour la maximisation de la moyenne $\mathcal{M}_r$ et de la moyenne $\mathcal{M}_r$ respectivement. Alors*

$$ALGO_r \leq ALGO_s \tag{6.34}$$

avec égalité si et seulement si l'algorithme glouton couvre le même nombre de sommets à toutes les étapes.

Conclusion

Tout au long de ces problèmes de couverture, nous avons pu trouver une borne minimale à l'efficacité de l'algorithme glouton, et nous avons pu montrer que dans la plupart des cas cette borne pouvait être atteinte asymptotiquement, c'est-à-dire qu'elle était la plus restrictive possible. Cela semble normal. Le fait d'avoir pu trouver certaines bornes constantes, quelle que soit la taille de l'instance, est un point supplémentaire poussant à persévérer dans cette voie de problèmes. Le plus interpellant reste le fait que l'algorithme glouton est à chaque fois le meilleur algorithme polynomial, en supposant que $\mathcal{P}\neq\mathcal{NP}$. Tout porte à croire que la structure même de ces problèmes possède un lien certain avec la façon dont l'algorithme glouton sélectionne ses hyperarêtes.

De nombreuses questions se posent sur cette généralisation par les moyennes $\mathcal{M}_r$. Comment se comporte la borne d'approximation de l'algorithme glouton en fonction de r ? Est-ce une fonction continue ? est-elle définie pour tout r réel ? Est-elle calculable ? Le théorème d'inégalité sur les moyennes présenté à la dernière section joue-t-il un rôle dans la détermination de cette borne ? Et concernant l'inapproximabilité, on est en droit d'espérer que l'algorithme glouton reste le meilleur algorithme polynomial pour tout r, sauf s'il s'avère que $\mathcal{P}=\mathcal{NP}$. Ou plus généralement, quel est l'ensemble des fonctions objectif pour lesquelles l'algorithme glouton est le meilleur algorithme polynomial sous l'hypothèse d'inégalité entre $\mathcal{P}$ et $\mathcal{NP}$?

Ce ne sont certes pas des questions simples. Et il est possible que certaines d'entre elles ne trouvent jamais de réponse. Cependant ce sont là quelques motivations pour une éventuelle généralisation du problème de Couverture d'ensemble.

Bibliographie

[1] G. H. Hardy, J. E. Littlewood et G. Pólya. *Inequalities*. Cambridge University Press. 1952.

[2] Thierry Massart. *Programmation, Info-F-101*. Cours de première candidature en informatique, ULB. Presses universitaires de Bruxelles. 2004.

[3] Geir Dahl. *An introduction to convexity, polyhedral theory and combinatorial optimization*. Cours de l'University of Oslo, Department of Informatics. 1997.

[4] Francis Buekenhout et Jean Doyen. *Ensembles structurés et groupes de symétrie*. Cours de première candidature en sciences mathématiques, ULB. 1988.

[5] Jon Kleinberg et Eva Tardos. *Algorithm Design*. [Chapitre 8 ($\mathcal{NP}$ *and Computational Intractability*) et chapitre 11 (*Approximation Algorithms*)]. Editions *Pearson International*. 2006.

[6] Uriel Feige, László Lovász et Prasad Tetali. *Approximating Min-Sum Set Cover. Algorithmica*, 40(4) :219-234, 2004.

[7] Uriel Feige. *A threshold of ln n for approximating set cover. J. ACM*, 45(4) :634-652, 1998.

[8] Alexander Wolff. *The Hardness of Approximating Set Cover. Lectures on Proof Verification and Approximation Algorithms*, volume 1367 of *Lecture Notes in Computer Science Tutorial*, chapitre 10, pages 249-262. Springer-Verlag, 1998.

[9] Jean Cardinal, Samuel Fiorini et Gwenaël Joret. *Tight Results on Minimum Entropy Set Cover. 9th. International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, volume 4110 of *Lecture Notes in Computer Science* pages 61-69. Springer-Verlag, 2006.

[10] Amotz Bar-Noy, Magnús M. Halldórsson, Guy Kortsarz. *A Matched Approximation Bound for the Sum of a Greedy Coloring. Information Processing Letters*, 71 (1999), 135-140.

[11] Eran Halperin et Richard M. Karp. *The minimum-entropy set cover problem. Theoret. Comput. Sci.*, 348(2-3) :240-250, 2005.

[12] Ivan Niven. *The Dolciani Mathematical Expositions, number six, Maxima and minima without calculus*. Publié et distribué par *The Mathematical Association Of America*. 1981.

Table des matières