Reconstruction de communauté

Maxime Darrin

21 mars 2020

Table des matières

Notations

Notations	Descriptions
$G \sim \mathbb{G}(N, p_A, p_B, p)$	G est un graphe aléatoire construit selon le modèle 1
$A : B$	Désigne une coupe ou une bissection d'un graphe dont A et B forment une partition des sommets
$\|A : B\|$	est le flot entre A et B, c'est à dire le nombre d'arêtes entre A et B si le graphe n'est pas pondéré et la somme des capacités de arêtes si le graphe est pondéré.
$\preceq$	désigne la relation d'ordre sur les coupes ou les bissections telle que une coupe est plus petite qu'une autre si son flot est plus petit.
$\leqslant_P$	désigne la relation d'ordre partielle entre les problèmes par réduction de Karp[7].
$\mathcal{NP}, \mathcal{P}$	désignent respectivement les classes de complexité polynomiale déterministe et polynomiale non déterministe.
$\mathcal{NP}$-difficile	désigne la classe des problèmes au moins aussi durs que tout problème de $\mathcal{NP}$
$\mathcal{NP}$-complet	désigne la classe des problèmes de $\mathcal{NP}$ et au moins aussi durs que tous les problèmes de $\mathcal{NP}$, ie $\mathcal{NP}$-difficile.

1 Introduction

Une importante part des problèmes modernes se ramène à l'étude de graphes. En particulier, la grande majorité des données et traces liées aux activités humaines peuvent être modélisés par des réseaux, des relations ou des circuits. Que ce soit les réseaux informatiques – centralisés ou pair à pairs, ou plus simplement les interactions ou liens sociaux, toutes ces données peuvent l'objet de recherche de communautés. La recherche de communautés à bien évidemment de très importantes applications pour les études sociologiques et les analyses socio-économiques. Un exemple original est celui de l'identification d'utilisateurs du *Bitcoin*. Si a priori c'est un système anonyme, il n'en reste pas moins que par principe de la *blockchain* toutes les transactions et tous les échanges sont publics, ainsi il est possible pour n'importe qui de les étudier. L'anonymat provient du fait que chacun peut se créer autant de *comptes* qu'il le souhaite, et ensuite effectuer des transactions entre ses propres comptes pour rappatrier l'argent là où il le souhaite. Il est ainsi possible en étudiant le graphe, dont les sommets sont les comptes et les arêtes les transactions, de reconstruire les communautés d'échanges et identifier des acteurs, des groupes de comptes appartenant à une unique entité etc...[5].

Dans ce cours, nous allons plus particulièrement étudié le problème de la reconstruction exacte de communauté[1], c'est à dire, en supposant l'existence de 2 communautés sous-jacentes, de tailles égales conditionnant la construction du graphe de relations entre les individus, retrouver ces deux communautés exactement en étudiant uniquement le graphe induit. Nous nous ramèneront à un problème de partitionnement de graphe courant – mais $\mathcal{NP}$-difficile – dont les applications ne se limitent pas à la reconstruction de communautés. En effet, le problème de la bissection minimale a des applications importantes en ordonnancement de tâches ou en calculs scientifiques[3].

Pour étudier le problème de reconstruction exacte de communautés dans un graphe nous définirons tout d'abord un modèle de graphe aléatoire modélisant les graphes induits par des communautés pré-établies puis nous donnerons une méthode reconstruction exacte avec grande probabilité de ces communautés avant d'en étudier en détail la classe de complexité algorithmique. Plus précisément, nous montrerons qu'une bissection minimale du graphe est solution du problème de reconstruction exacte avec grande probabilité mais que la recherche d'une telle bissection est un problème $\mathcal{NP}$-complet.

1.1 Modèle

1.1.1 Construction de graphe

Définition 1 (Modèle 1)**.** L'objectif est de construire un graphe aléatoire $G = (V, E)$. On se donne $V = \{1, \cdots, 2N\}$ un ensemble, $p, p_A, p_B \in [0, 1]$ des probabilités et on suppose qu'on dispose de $A \subset V$, tel que $|A| = N$ construit aléatoirement, on note alors $B = V \backslash A$. On alors : $V = A \sqcup B$ $|B| = |A| = N$.

On définit $\sigma : V \longrightarrow \{-1, 1\}$ la fonction d'étiquetage des sommets telle que :

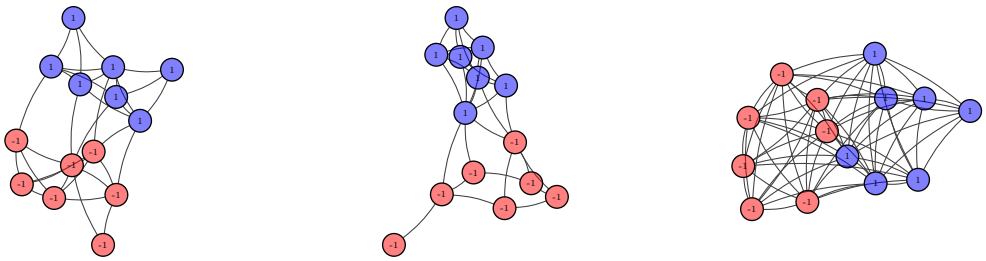
$$\forall v \in V, \sigma(v) = \begin{cases} 1 & \text{si } v \in A \\ -1 & \text{si } v \in B \end{cases}$$

On construit alors les arêtes de E de la façon suivante pour tout $(u, v) \in V^2$.

$$\mathbb{P}[(u,v) \in E] = \begin{cases} p & \text{si } \sigma(u) \neq \sigma(v) \\ p_A & \text{si } u, v \in A \\ p_B & \text{si } u, v \in B \end{cases}$$

Dans la suite on notera $G \sim \mathbb{G}(N, p_A, p_B, p)$, G est un graphe aléatoire construit selon ce modèle.

1.1.2 Exemples



(a) $p_A = p_B = 0.8, p = 0.1$ (b) $p_A = 0.9, p_B = 0.3, p = 0.05$ (c) $p_A = p_B = 0.9, p = 0.5$

FIGURE 1 – Exemples de graphes générés par le modèle avec différents paramètres

1.2 Problème de reconstruction exacte

On suppose maintenant que l'on dispose d'un graphe $G \sim \mathbb{G}(N, p_A, p_B, p)$ et que A et B étaient les deux communautés originales et σ la fonction d'étiquetage associée.

Définition 2 (Problème de reconstruction exacte)**.** On cherche à reconstruire σ en l'estimant à partir uniquement de l'observation du graphe. Le problème revient donc à construire $\hat{\sigma}$ tel que :

$$\mathbb{P}[\hat{\sigma} = \pm\sigma] \underset{N\to\infty}{\longrightarrow} 1$$

Cela revient à retrouver A et B à renommage près.

Nous allons examiner une solution particulière de ce problème.

1.3 Bissection minimale

Définition 3 (Bissection et coupe)**.** Soit $G = (V, E)$ un graphe. Une coupe de G est une partition des sommets en deux ensembles A, B.

On parle alors de la coupe $A : B$ et de la bissection $A : B$ si A et B sont de tailles égales. On note de plus $A : B$ l'ensemble des arêtes dont une extrémité est dans A et l'autre dans B.

Remarque 1. Par la suite on utilisera la notation $A : B$ pour introduire la coupe d'ensembles A, B : $V = A \sqcup B$. On parlera par la suite de la bissection ou de la coupe $A : B$ suivant si A et B sont de tailles égales ou non.

Définition 4 (Flot)**.** Soient $G = (V, E)$ et $A, B \subset V$ disjoints. On appelle flot entre A et B le nombre d'arêtes (ou la somme des capacités des arêtes)

On note cette valeur $|A : B|$. En effet, dans le cas d'un graphe non pondéré, le flot correspond au cardinal de $A : B$.

Définition 5 (Ordre sur les bissections)**.** Soit $G = (V, E)$, et $A : B$, $A' : B'$ deux bissections de G. On dit que $A : B \preceq A' :' B$ si et seulement $|A : B| \leqslant |A' : B'|$.

C'est à dire une bissection est plus petite qu'une autre si le flot, ie la capacité des arêtes allant d'un ensemble à l'autre de la bipartition, est plus petit que le flot de l'autre

Dans ces conditions la – ou les bissections – minimales sont les bissections minimales pour $\prec$, *ie.* toutes les autres bissections envisageables ont un flot au moins supérieur au leurs.

Il se trouve que le cadre du modèle 1 précédemment énoncé, la bissection minimale est presque sûrement solution du problème de reconstruction exacte[1]. Dans la section suivante nous allons voir la preuve de ce résultat avant d'analyser la complexité du problème de a bissection minimale d'un graphe[6].

2 Solution du problème de reconstruction exacte

On se place dans le cadre du modèle 1 et on suppose ainsi que l'on a $G \sim \mathbb{G}(N, p_A, p_B, p)$ avec A et B les communautés originelles et σ l'étiquetage associé.

On suppose sans perte de généralité que $p_A \leqslant p_B$. De plus on suppose que le bruit dans le graphe n'est pas trop grand, c'est à dire que la probabilité qu'il y ait des arêtes inter-communauté est plus petit que la probabilité d'arête intra-classe : $2p \leqslant p_A + p_B$.

Théorème 1. $\mathbb{P}[A : B \textit{ est l'unique bissection minimale}] \underset{N\to\infty}{=} 1 + o(1)$

Démonstration. Dans le cadre ci-dessus. Soient A', B' une bissection de G, on a donc $|A'| = |B'| = N$.

On va alors comparer le flot des deux bissections $A : B$ et $A' : B'$, on définit pour cela :

$$\Delta = |A' : B'| - |A : B|$$

On a donc $\Delta < 0$ si et seulement si $A' : B'$ est une bissection plus petite que $A : B$.

On commence par borner $\mathbb{P}[A' : B' \prec A : B] = \mathbb{P}[\Delta < 0]$, pour cela on va réécrire Δ.

Tout d'abord on définit $A_1 = A \cap B'$ $B_1 = B \cap B'$ $A_2 = A \cap A'$ $B_2 = B \cap A'$
et $k = |A_1| = |B_2|$. On peut d'ailleurs supposer sans perte de généralité que $k \leqslant \frac{1}{2}N$ quitte à permuter les notations [1]. En effet, on a $B' = A_1 \sqcup B_1$ donc $|B'| = k + |B_1|$ et donc $|B_1| = N - k$ or $B = B_1 \sqcup B_2$ donc $|B| = |B_1| + |B_2|$ et ainsi $|B_2| = N - (N - k) = k$.

1. Je pense que je mérite des points bonus rien que pour ces patates en Tikz.

De plus si $k \geqslant \frac{1}{2}N$, alors $|A_2| = N - k \leqslant \frac{1}{2}N$ et par le même raisonnement que précédemment B_1 aussi. Il suffit alors de permuter les notation et d'échanger A_1, A_2 et B_1, B_2.

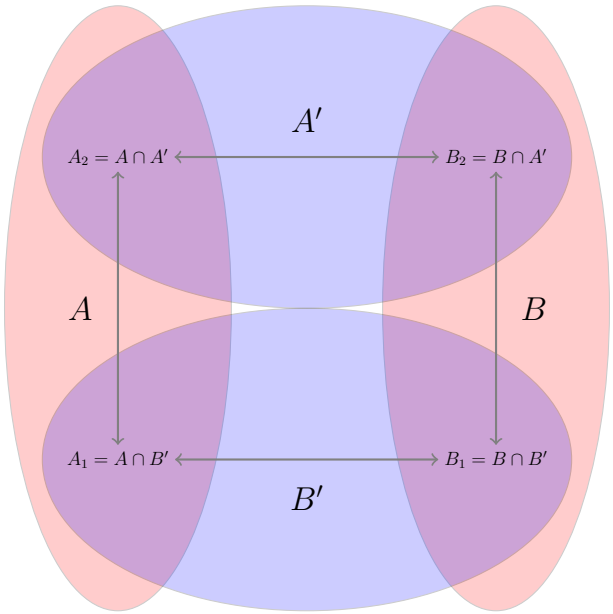


FIGURE 2 – Représentation des 2 coupes considérées

Et on remarque que :

$$A' : B' = (A \cap B') : (A \cap A') \sqcup (B \cap B' : B \cap A') = (A_1 : A_2) \sqcup (B_1 : B_2)$$
$$A : B = (A \cap B') : (B \cap B') \sqcup (A \cap A' : B \cap A') = (A_1 : B_1) \sqcup (A_2 : B_2)$$

Ainsi on a $\Delta = |A_1 : A_2| + |B_1 : B_2| - |A_1 : B_1| - |A_2 : B_2|$

De plus dans le cadre de ce modèle on a $|A_1 : B_1|, |A_2 : B_2| \sim \mathcal{B}(k(m-k), p)$.

En effet $|A_1 : B_1|$ est le nombre d'arêtes présentes entre $A_1 = A \cap B'$ et $B_1 = B \cap B'$, il y a $|A_1| \times |B_1| = k(N-k)$ arêtes possibles entre ces deux ensembles. Deplus chacune sont présentes avec probabilité p puisque $A_1 \subset A$ et $B_1 \subset B$.

Avec un raisonnement similaire on obtient que $|A1 : A2| \sim \mathcal{B}(k(m-k), p_A)$ et $|B1 : B2| \sim \mathcal{B}(k(m-k), p_B)$.

On peut alors écrire $\Delta = \sum_{i=1}^{k(m-k)} \chi_i$ où les χ_i sont des variables aléatoires entières à valeurs dans $[-2, 2]$ et d'espérance $\mathbb{E}[\chi_i] = p_a + p_b - 2p$.

On rappelle alors l'inégalité de Hoeffding :

Par application de l'inégalité d'Hoeffding on obtient alors :

$$\begin{aligned}\mathbb{P}[\Delta < 0] = \mathbb{P}[\Delta - \mathbb{E}\Delta < -\mathbb{E}\Delta] &= \mathbb{P}[\Delta - \mathbb{E}\Delta < -k(N-k)(p_a + p_b - 2p)] \\ &\leqslant \exp\left(-\frac{2(k(N-k)(p_a+p_b-2p))^2}{\sum_{i=1} k(n-k)(2+2)^2}\right) \\ &\leqslant \exp\left(-\frac{2k^2(N-k)^2(p_a+p_b-2p)^2}{k(n-k)16}\right) \\ &\leqslant \exp\left(-\frac{2k(N-k)(p_a+p_b-2p)^2}{16}\right)\end{aligned}$$

On peut maintenant contrôler $\mathbb{P}[A : B$ n'est pas la bissection minimale$]$

$$\begin{aligned}&\mathbb{P}[A : B \text{ n'est pas la bissection minimale}] \leqslant \sum_{A' : B' \text{une bissection}} \mathbb{P}[\Delta < 0] \\ &= \sum_{k=1}^{\lfloor \frac{N}{2} \rfloor} \binom{N}{k}^2 \mathbb{P}[\Delta < 0]\end{aligned}$$

Pour construire une nouvelle bissection on prend k éléments dans A et k éléments dans B pour former A', et cela fixe $B' = V \backslash A'$.

$$\begin{aligned}&\leqslant \sum_{k=1}^{\lfloor \frac{N}{2} \rfloor} \binom{N}{k}^2 \exp\left(-\frac{2k(N-k)(p_a+p_b-2p)^2}{16}\right) \\ &\leqslant \sum_{k=1}^{\lfloor \frac{N}{2} \rfloor} \binom{N}{k}^2 \exp\left(-2k(N-k)\lambda^2\right) \qquad \text{avec } \lambda = \frac{p_a+p_b-2p}{4} > 0\end{aligned}$$

Nous allons maintenant montrer que cette borne tend vers 0 lorsque N grandit, plus précisément nous montrons que c'est un $\mathcal{O}\left(N^2 \exp\left(-2N\lambda^2\right)\right)$.

On remarque tout d'abord que $\lambda^2 \in]0, \frac{1}{4}]$, posons alors $p = \left\lceil \frac{4}{\lambda^2} \right\rceil \geqslant 16$.

On va alors scinder la somme en deux en conservant d'un côté les $p-1$ premiers termes et les autres de l'autre.

Tout d'abord on a :

$$\begin{aligned}\sum_{k=1}^{p-1} \binom{N}{k}^2 \exp\left(-2k(N-k)\lambda^2\right) &\leqslant \sum_{k=1}^{p-1} N^{2k} \exp\left(-2k(N-p+1)\lambda^2\right) \\ &\leqslant N^2 \exp\left(-2(N-p+1)\lambda^2\right) \sum_{k=0}^{p-2} \left(N^2 \exp\left(-2k(N-p+1)\lambda^2\right)\right)^k \\ &= \mathcal{O}\left(N^2 \exp\left(-2N\lambda^2\right)\right)\end{aligned}$$

Maintenant on va montrer que le reste de la somme est négligeable.

$$\sum_{k=p}^{\lfloor \frac{N}{2} \rfloor} \binom{N}{k}^2 \exp\left(-2k(N-k)\lambda^2\right) \leqslant \frac{N}{2} 4^N \exp\left(-p\frac{N}{2}\lambda^2\right) \qquad \text{car } \binom{N}{k} \leqslant 2^N$$
$$= \frac{N}{2}\left(4\exp\left(-p\frac{N}{2}\lambda^2\right)\right)^N$$

Or puisque l'on a $p \geqslant 16$, on a $4\exp\left(-p\frac{2}{\lambda}^2\right) \leqslant 4\exp\left(-\frac{16}{2}\lambda^2\right) = 4\exp\left(-8\lambda^2\right) \leqslant \exp(-\lambda^2)$

On conclut donc que $\sum_{k=p}^{\lfloor \frac{N}{2} \rfloor} \binom{N}{k}^2 \exp\left(-2k(N-k)\lambda^2\right) \underset{N\to\infty}{=} o\left(N^2 \exp\left(-2N\lambda^2\right)\right)$.

Ainsi on obtient finalement $\mathbb{P}[A : B \text{ n'est pas la bissection minimale}] \underset{N\to\infty}{=} \mathcal{O}\left(N^2 \exp\left(-2N\lambda^2\right)\right)$

Ainsi on a bien

$$\mathbb{P}[A : B \text{ est la bissection minimale}] = 1 - \mathbb{P}[A : B \text{ n'est pas la bissection minimale}]$$
$$\underset{N\to\infty}{=} 1 - o(1)$$

□

Ainsi on peut ramener le problème de reconstruction exacte de communauté à la recherche d'une bissection minimale qui sera alors avec grande probabilité les communautés recherchées.

3 Problème de la bissection minimale

Jusqu'ici nous avons essentiellement parlé du problème d'optimisation associé à la bissection minimale, c'est à dire trouver effectivement la bissection minimale. Nous verrons plus tard que ce problème est $\mathcal{NP}$-difficile mais qu'il n'est a priori pas dans $\mathcal{NP}$, c'est pourquoi nous allons étudier le problème de décision dans la suite. Tout d'abord, nous ferons des rappels concernant les classes de complexité de problèmes et nous définirons proprement $\mathcal{P}$ et $\mathcal{NP}$, ensuite nous montrerons que le problème de décision – et d'optimisation – sont $\mathcal{NP}$-difficile et que le problème de décision appartient de plus à $\mathcal{NP}$ le rendant $\mathcal{NP}$-complet.

3.1 Complexité Algorithmique

On donne ici des définitions rapides des notions de complexité algorithmique, pour être absolument rigoureux et précis il serait nécessaire de donner des définitions formelles des machines de Turing, déterministes et non déterministes. Nous nous fonderons dans la suite sur la caractérisation par certificat de la classe $\mathcal{NP}$ et sur le théorème de *Cook-Levin*[4], et nous admettrons ces deux résultats.

3.1.1 Réductions de problèmes

Définition 6 (Réduction de Karp)**.** Soient A, B deux ensembles de problèmes. Une réduction de A à B est une fonction $f : \begin{cases} A \longrightarrow B \\ \mathcal{I}_A \mapsto f(\mathcal{I}_A) = \mathcal{I}_B \end{cases}$ Où $\mathcal{I}_A$ et $\mathcal{I}_B$ sont des instances de A et B.

Cette fonction vérifiant pour tout $\mathcal{I}_A$, $f(\mathcal{I}_A)$ a une solution équivaut à $\mathcal{I}_A$ a une solution.

Si de plus f est calculable en temps polynomial par un algorithme (ie par une machine de Turing déterministe) on dit alors que f est une réduction polynomiale.

Définition 7 (Relation d'ordre de complexité)**.** Si une telle réduction polynomiale existe on dit que B est plus difficile que A. Et on note $A \leqslant_P B$, signifiant : il existe une réduction polynomiale de A en B.

Définition 8 (Problèmes équivalents)**.** Si on a $A \leqslant_P B$ et $B \leqslant_P A$, alors on dit que ces deux problèmes sont équivalents. On peut montrer que c'est effectivement une relation d'équivalence. Dans la suite nous étudierons l'ensemble des problèmes quotientés par cette relation d'équivalence et nous verrons que les problèmes $\mathcal{NP}$-COMPLET forment une de ces classes d'équivalence.

Cela revient à dire que l'on peut transformer (rapidement) tout problème de type A en un problème de type B de sorte que cette transformation ait une solution si et seulement si le problème d'origine a une solution, ainsi il "suffit" de savoir résoudre les problèmes de B pour résoudre les problèmes de A.

3.1.2 $\mathcal{P}$ et $\mathcal{NP}$

Définition 9 ($\mathcal{P}$: Déterministe polynomial)**.** La classe de problèmes déterministe polynomiale, notée $\mathcal{P}$ correspond à l'ensemble des problèmes solubles par un algorithme (ie. une machine de Turing déterministe) en temps polynomial.

Définition 10 ($\mathcal{NP}$: Non Déterministe polynomial)**.** Il existe deux définitions de cette classe, nous nous contenterons de la caractérisation par certificat.

La classe de problèmes non déterministe polynomiale, notée $\mathcal{NP}$ correspond à l'ensemble des problèmes dont vérifier si une solution candidate – un certificat – est effectivement une solution ou non s'effectue en temps polynomial par un algorithme.

Cette caractérisation est équivalente à la définition par les machine de Turing non déterministes de la classe $\mathcal{NP}$: la classe $\mathcal{NP}$ est l'ensemble des problèmes solubles en temps polynomial sur une machine de Turing non-déterministe. Néanmoins, par simplicité nous nous contenterons de la première dans la suite et nous admettrons que ces deux définitions sont équivalentes.

Exemple 1 (Quelques problèmes dans $\mathcal{NP}$)**.** Soit $G = (V, E)$ un graphe, peut-on le colorier avec moins de k couleurs ? Trouver tel coloriage est difficile, mais si on en a un, il est facile de vérifier qu'il comporte moins de k couleurs et que 2 sommets adjacents n'ont jamais la même couleur.

Soient W, V, $v_1 \cdot v_n$, $w_1 \cdots w_n$ des entiers représentant un sac à dos pouvant contenir au maximum W kilogrammes, et des objets de valeurs v_i et de poids w_i. Puis-je remplir mon sac à dos de sorte

que la somme des valeurs des objets dedans soit plus grande que V et que la somme de leur poids plus petit que W. Il est difficile de remplir le sac, en revanche, si on dispose d'un sac rempli, on peut facilement vérifier s'il respecte les deux conditions.

Définition 11 (Problèmes $\mathcal{NP}$-difficile)**.** Un problème A est dit $\mathcal{NP}$-difficile si :

$$\forall B \in \mathcal{NP}, B \leqslant_P A$$

.

C'est à dire que A est au moins aussi difficile que tous les problèmes dans $\mathcal{NP}$: il existe une réduction polynomiale permettant de se ramener à un problème de A pour tous problèmes de $\mathcal{NP}$.

On appelle par ailleurs instance d'un problème un cas particulier du problème.

Définition 12 ($\mathcal{NP}$-complet)**.** On dit qu'un problème est $\mathcal{NP}$-complet s'il est à la fois $\mathcal{NP}$-difficile et dans NP.

Définition 13 (Problème SAT)**.** Ce problème s'intéresse à la satisfiabilité d'une formule logique propositionnelle (ie une combinaison de variables du premier ordre, de négations, de conjonction et de disjonction). C'est à dire répondre à la question : existe-il une assignation des variables qui la formule vraie ?

Soient $x = (x_1, \cdots, x_n) \in \mathcal{X}$ des variables et ϕ une formule de la logique propositionnelle.

Existe-il $\sigma : \mathcal{X} \longrightarrow \{\top, \bot\}$ telle que $\phi(\sigma(x_1) \cdots \sigma(x_n))$ s'évalue à $\top$?

Théorème 2 (Cook-Levin)**.** *SAT est* $\mathcal{NP}$-complet.

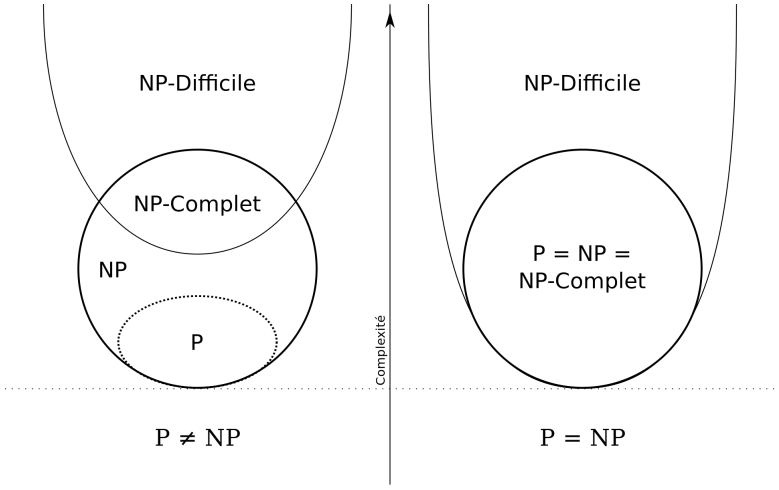


Figure 3 – Illustration des classes de problèmes $\mathcal{P}$ et $\mathcal{NP}$

De manière générale on s'appuie sur les réductions entre problème et la caractérisation par certificat de $\mathcal{NP}$ pour montrer qu'un problème donné est $\mathcal{NP}$-complet. En effet, pour démontrer qu'un problème A est $\mathcal{NP}$-complet, il suffit de vérifier qu'il est dans $\mathcal{NP}$ via la caractérisation par certificat et de construire une réduction polynomiale d'un problème B, que l'on sait être $\mathcal{NP}$-difficile, au problème A. On peut alors conclure de la façon suivante.

On sait que $\forall C \in \mathcal{NP}, C \leqslant_P B$ et on a montré $B \leqslant_P A$, donc $\forall C \in \mathcal{NP}, C \leqslant_P A$ donc A est $\mathcal{NP}$-DIFFICILE et si de plus il est dans $\mathcal{NP}$, il est $\mathcal{NP}$-COMPLET.

L'un des plus fameux articles sur la question est l'article publié en 1972 par Richard Karp[7], il propose des réductions pour 21 problèmes de $\mathcal{NP}$ à SAT, montrant ainsi que ces 21 problèmes sont non seulement tous dans $\mathcal{NP}$ mais aussi tous $\mathcal{NP}$-DIFFICILE et équivalents entre eux. On parle des 21 problèmes $\mathcal{NP}$-COMPLET de Karp. Plus tard nous verrons une réduction de *min bissection* au problème $max - cut$ qui fait partie des problèmes de Karp.

3.1.3 P versus NP

Digression problème du millénaire...

Une des grandes questions en informatique fondamentale est de savoir si $\mathcal{P} = \mathcal{NP}$, si $\mathcal{P} \neq \mathcal{NP}$ ou si ce résultat est indécidable.

Philosophiquement, en se basant sur les définitions de ces deux classes données plus tôt, cela revient à savoir si résoudre un problème en temps polynomial c'est aussi difficile que de vérifier si une solution-candidate est effectivement une solution en temps polynomial. Une analogie, très fausse pour beaucoup de raisons, mais qui donne une bonne idée du problème philosophique est celle avec une preuve de maths.

Dans ce cas la question revient à : est-il aussi facile de construire une preuve de maths que de la vérifier une fois construite ? A priori, on a envie de dire qu'il est infiniment plus compliqué de construire une preuve que de vérifier la cohérence et la justesse d'une preuve déjà construite. Néanmoins dans le cas des problèmes en question on n'a pu montrer ni que c'était strictement plus difficile ni que c'était aussi facile. De plus, pour montrer que $\mathcal{P} = \mathcal{NP}$, il suffirait de fournir un algorithme qui tourne en temps polynomial pour l'un des problèmes $\mathcal{NP}$-COMPLET, en effet puisque tous les problèmes de la classe peuvent se réduire par réduction successive à l'un de ces problèmes en temps polynomial, cet algorithme serait à même de résoudre tous les problèmes de la classe $\mathcal{NP}$ en temps polynomial.

3.2 Min bissection

Nous allons montrer que le problème de décision associé à *min bissection* est $\mathcal{NP}$-COMPLET car le problème d'optimisation n'est a priori pas dans $\mathcal{NP}$.

3.2.1 Appartenance à $\mathcal{NP}$

Définition 14 (Min bissection – Problème d'optimisation)**.** Soit $G = (V, E)$ un graphe, le problème consiste à trouver $A : B$ une bissection minimale pour $\preceq$ dans G.

Remarque 2. C'est en fait une version plus forte du problème de la coupe minimale, dans le cas de la coupe minimale on cherche uniquement une coupe et non une bissection, c'est à dire que les deux communautés n'ont pas à avoir la même taille. Il est important de noter que le problème de la

coupe minimale est significativement plus facile à résoudre en terme de complexité algorithmique car il est dans $\mathcal{P}$.

La preuve s'effectue en montrant qu'il y a équivalence entre le problème de la coupe minimale et celui du flot maximum[8] [2], et en vérifiant que ce dernier est soluble en temps polynomial par exemple avec l'algorithme de Ford-Fulkerson.

Il n'est a priori pas possible de vérifier que cette bissection est effectivement la plus petite sans la comparer à toutes les bissections possibles. Donc ce problème n'est a priori pas dans $\mathcal{NP}$. Nous allons l'affaiblir de sorte qu'il tombe dans NP.

Définition 15 (Min bissection – Problème de décision)**.** Soient $G = (V, E)$ un graphe et $K \in \mathbb{N}$, le problème consiste à trouver $A : B$ une bissection telle que $|A : B| < K$.

Dans ce cas, il est facile de vérifier si l'on a une bissection-candidate $A : B$ que $|A : B| \leqslant K$. Il suffit compter le nombre d'arêtes entre les deux ensembles (ou la somme de leur capacité), dans les deux cas, cela se fait en $\mathcal{O}(N^2)$ en pire cas. Ainsi le problème de décision de la bissection minimale est dans $\mathcal{NP}$.

3.2.2 $\mathcal{NP}$-difficile

La preuve permettant de montrer que *min-bissection* est $\mathcal{NP}$-difficile se fait grâce à 3 réductions remontant au problème *3-SAT* qui est $NP - complet$ car c'est l'un des 21 problèmes de Karp [7]. Nous allons montrer successivement :

$$\textit{3-SAT} \leqslant_P \textit{MAX-2SAT} \leqslant_P \textit{SIMPLE-MAX-CUT} \leqslant_P \textit{SIMPLE-MIN-BISSECTION}$$

Remarque 3. Ici nous montrons, en fait, que le problème *MIN-BISSECTION-SIMPLE*, c'est à dire restreint aux graphes non pondérés reste $\mathcal{NP}$-difficile. En effet, il est a priori moins général que *MIN-BISSECTION* ie le même problème avec des graphes pondérés. On peut montrer, qu'ils sont en fait équivalent puisque tous deux sont dans $\mathcal{NP}$-complet mais la réduction pour le problème générale se fait à partir du problème *MAX-CUT* dans le cas général.

Définition 16 (*3-SAT*[9])**.** Soient $C_1 \cdots C_p$ des clauses disjonctives contenant au plus 3 littéraux impliquant des variables $x_1 \cdots x_n$. Et $\Phi(x_1 \cdots x_n) = \bigwedge_{i=1}^{p} C_i$. On dit que Φ est une formule *3-SAT* et on cherche à décider s'il existe une affectation des variables faisant s'évaluer Φ à vrai.

Remarque 4. Si on restreint encore le problème *3-SAT* avec des clauses disjonctives contenant au plus deux littéraux on obtient le problème *2-SAT* qui est lui dans $\mathcal{P}$ [9]. Ainsi, on voit qu'une restriction en apparence minime change de manière fondamentale la classe de complexité du problème.

Définition 17 (*MAX-2SAT*)**.** Soient $C_1 \cdots C_p$ des clauses logiques disjonctives de 2 littéraux au plus et $K \in K$ impliquant des variables $x_1 \cdots x_n$. On cherche à décider s'il existe une affectation des variables $(x_i)_i$ telle que K clauses au moins s'évaluent à vrai.

Définition 18 (*SIMPLE-MAX-CUT*)**.** Soient $G = (V, E)$ et $W \in \mathbb{N}$ on cherche à décider s'il existe une coupe $S_1 : S_2$ telle que $|S_1 : S_2| \geqslant W$.

2. Excellent ouvrage de Christos Papadimitriou, personnage très important en théorie de la complexité et de la théorie des jeux.

Définition 19 (*SIMPLE-MIN-BISSECTION*)**.** Soient $G = (V, E)$ et $W \in N$, on cherche à décider s'il existe une bissection $|A : B| \leqslant W$.

Théorème 3. $3 - SAT \leqslant_P MAX - 2SAT$.

Démonstration. Soit $\mathcal{I}_1$ une instance de *3-SAT*. Soient $C_1, \cdots C_n$ des clauses disjonctives. On peut supposer sans perte de généralité que toutes ces clauses disjonctives de $\mathcal{I}_1$ possèdent exactement 3 littéraux : on peut en effet compléter celles qui en ont moins en ajoutant des variables pour obtenir une formule équivalente et de taille linéaire.

On se donne a_i, b_i, c_i pour $i \in [|1, n|]$ les littéraux de la clause i.

Remarque 5. Ce ne sont pas des variables distinctes seulement des littéraux c'est à dire une variable ou sa négation. On peut très bien avoir $a_i = b_k = a_j$.

On va construire une instance $\mathcal{I}_2$ de *MAX-2SAT* en formant pour chaque clause $C_i = a_i \vee b_i \vee c_i$ les clauses :

1. a_i, b_i, c_i, d_i
2. $\neg a_i \vee \neg b_i$, $\neg a_i \vee \neg c_i$, $\neg b_i \vee \neg c_i$
3. $a_i \vee \neg d_i$, $b_i \vee \neg d_i$, $c_i \vee \neg d_i$

et on pose $K = 7n$

Cette construction de $\mathcal{I}_2$ est linéaire en la taille de $\mathcal{I}_1$ et donc a fortiori polynomiale.

Supposons qu'il existe une assignation des variables évaluant la formule *3-SAT* à vrai, alors dans chaque clause, au moins un des littéraux est vrai. On va montrer que quel que soit le nombre de littéraux vrais dans une clause $C_i = a_i \vee b_i \vee c_i$ de $\mathcal{I}_1$, il existe une affectation de d_i permettant de valider exactement 7 clauses dans $\mathcal{I}_2$.

Soit $i \in [|1, n|]$,

- S'il y a un unique littéral à vrai, on suppose sans perte de généralité que c'est a_i, alors dans $\mathcal{I}_2$: $(a_i), (\neg a_i \vee \neg b_i), (\neg a_i \vee \neg c_i), (\neg b_i \vee \neg c_i)$ sont vraies, et quitte à prendre $d_i = \bot$ on a $(a_i \vee \neg d_i)$, $(b_i \vee \neg d_i)$, $(c_i \vee \neg d_i)$ vraies aussi. On a alors 7 clauses vraies.
- S'il y a exactement 2 littéraux à vrai, on suppose sans perte de généralité que c'est a_i, b_i alors dans $\mathcal{I}_2$: $(a_i), (b_i), (\neg a_i \vee \neg c_i), (\neg b_i \vee \neg c_i)$ sont vraies et quitte à prendre $d_i = \bot$ on a aussi $(a_i \vee \neg d_i)$, $(b_i \vee \neg d_i)$, $(c_i \vee \neg d_i)$ vraies. Et donc bien exactement 7 clauses vraies.
- Si les 3 littéraux sont vrais, alors dans ce cas dans $\mathcal{I}_2$: $(a_i), (b_i), (c_i), (a_i \vee \neg d_i)$, $(b_i \vee \neg d_i)$, $(c_i \vee \neg d_i)$ sont vraies et quitte à prendre $d_i = \top$ (d_i) aussi est vrai et on a bien nos 7 clauses.

On note que dans chacun des cas, l'autre affectation de d donne strictement moins de 7 clauses vraies.

Ainsi si $\mathcal{I}_1$ possède une solution alors $\mathcal{I}_2$ possède également une solution.

On montre maintenant que si $\mathcal{I}_2$ possède une solution alors $\mathcal{I}_1$ en possède une aussi. Par contraposée, on suppose qu'il n'existe pas d'affectation évaluant la formule *3-SAT* à vrai. Ainsi, il existe au moins une clause dont aucun des littéraux n'est vrai. Dans ce cas, on vérifie qu'au plus 6 clauses (pour $d_i = \bot$) peuvent être vérifiées, ainsi il y a strictement moins de $7m$ clauses vérifiées dans $\mathcal{I}_2$.

Ainsi $\mathcal{I}_1$ a une solution si et seulement si $\mathcal{I}_2$ a une solution. On a donc bien montré que *3-SAT* se réduit à *MAX-2SAT* □

Théorème 4. *MAX-2SAT* $\leqslant_{P}$ *SIMPLE-MAX-CUT*.

Démonstration. Soit $\mathcal{I}_1$ une instance de *MAX-2SAT* nous allons construire une instance $\mathcal{I}_2$ de *SIMPLE-MAX-CUT*.

Soient $C_1 \cdots C_m$ les clauses de $\mathcal{I}_1$ et K le nombre de clauses à valider, on peut supposer qu'elles contiennent toutes exactement 2 littéraux, quitte à répéter le littéral présent dans la clause, pour obtenir une formule équivalente.

Dans ce cas pour tout $i \in [|1, m|]$ on note $C_i = a_i \cup b_i$. On note $x_1 \cdots x_n$ les variables (et leur négation) impliquées dans les clauses de $\mathcal{I}_1$. On construit alors un graphe $G = (V, E)$ de *SIMPLE-MAX-CUT*.

On forme V les sommets du graphe de la façon suivante :

$$\begin{aligned} V = \{T_i : 0 \leqslant i \leqslant 3m\} \cup \{F_i : 0 \leqslant i \leqslant 3m\} \cup \{t_{ij} : 1 \leqslant i \leqslant n, 0 \leqslant j \leqslant 3m\} \\ \cup \{f_{ij} : 1 \leqslant i \leqslant n, 0 \leqslant j \leqslant 3m\} \\ \cup \{x_i : 1 \leqslant i \leqslant n\} \\ \cup \{\bar{x_i} : 1 \leqslant i \leqslant n\} \end{aligned}$$

On commence par construire un premier ensemble d'arêtes :

$$\begin{aligned} A_1 = &\{(T_i, Fj) : 0 \leqslant i \leqslant 3m, 0 \leqslant j \leqslant 3m\} \\ &\cup \{(t_{ij}, f_{ij}), 0 \leqslant i \leqslant n, 0 \leqslant j \leqslant 3m\} \\ &\cup \{(x_i, f_{ij}), 0 \leqslant i \leqslant n, 0 \leqslant j \leqslant 3m\} \\ &\cup \{(\bar{x_i}, t_{ij}), 0 \leqslant i \leqslant n, 0 \leqslant j \leqslant 3m\} \end{aligned}$$

$$\begin{aligned} A_2 = &\{(a_i, b_i) : 0 \leqslant i \leqslant m, a_i \neq b_i\} \cup \{(a_i, F_{2i-1}) : 0 \leqslant i \leqslant m\} \\ &\cup \{(b_i, F_{2i}) : 0 \leqslant i \leqslant m\} \end{aligned}$$

Dans la suite on dit, soit une partition S_1, S_2 des sommets que (u, v) est une bonne arête si et seulement si u et v appartiennent au même ensemble de la partition et que c'est une mauvaise arête sinon. On fait tout d'abord remarquer que toutes les arêtes de A_1 dès qu'une partition vérifie que tous les T_i sont dans un même ensemble et tous les F_i dans l'autre et si pour tout i, x_i et les f_{ij} sont dans un même ensemble et $\bar{x_i}$ et tous les t_{ij} dans l'autre.

De plus, si on a F_i, F_j, $i \neq j$ dans deux ensembles différents alors on a au moins $3m + 1$ arêtes de A_1 qui seront mauvaises (ie avec une extrémité dans un ensemble de la partition et l'autre dans l'autre) puisque ces deux sommets ont en commun $3p + 1$ voisins communs. De même, on a que si $x_i, \bar{x_i}$ sont dans le même ensemble, alors de la partition au moins $3m + 1$ arêtes de A_1 seront mauvaises car il y a $3m + 1$ chemin (disjoints) de longueur 3 de x_i à $\bar{x_i}$.

On construit ainsi $G = (V, E = A_1 \cup A_2)$ et $W = |A_1| + 2K$ une instance $\mathcal{I}_2$ de *SIMPLE-MAX-CUT*.

On suppose que $\mathcal{I}_1$ possède une solution, c'est à dire un assignement des variables $x_1 \cdots x_n$ rendant au moins K clauses vraies.

On construit alors une partition S_1, S_2 des sommets de G :

$$S_1 = \{(T_i, Fj) : 0 \leqslant i \leqslant 3m, 0 \leqslant j \leqslant 3m\} \cup \{x_i : x_i = \bot\}$$
$$\cup \{t_{ij} : x_i = \bot, 0 \leqslant j \leqslant 3m\}$$
$$\cup \{\bar{x_i}, x_i = \top\}$$
$$\cup \{f_{ij}, x_i = \top, 0 \leqslant j \leqslant 3m\}$$

$$S_2 = V \backslash S_1$$

Chaque clause possède alors un ou deux littéraux évalués à vrai et donc a_i ou b_i appartient à S_2 or dans A_2 il y a exactement 2 arêtes tombant sur cette clause qui sont bonnes. De plus toutes les arrêtes de A_1 sont donc bonnes. Et on a bien $W = |A_1| + 2k$.

Réciproquement, si on a S_1', S_2' une partition dont au moins W arêtes sont bonnes on a $|A_2| \leqslant 3m$ et $K \geqslant 0$. On en déduit qu'on a au plus $3m$ mauvaises arêtes.

Cela implique que tous les F_i sont dans le même ensemble de la partition, disons S_1', on a aussi pour chaque paire $x_i, \bar{x_i}$ l'un des deux appartient à S_1'.

Il suffit alors d'assigner la valeur $\top$ à x_i si et seulement si il appartient à S_2'.

Exercice 1. Le lecteur consciencieux pourra vérifier que cet assignement garantit qu'au moins K clauses de $\mathcal{I}_1$ sont vérifiées.

□

Théorème 5. *SIMPLE-MAX-CUT* $\leqslant_P$ *SIMPLE-MIN-BISSECTION*

Démonstration. Soit $\mathcal{I}_1$ une instance du problème *SIMPLE-MAX-CUT*, soit $G = (V, E)$ et $W \in \mathbb{N}^*$. On va construire une instance $\mathcal{I}_2$ de *MIN-BISSECTION*, on note $N = |V|$.

On construit $V' = V \cup \{U_1 \cdots U_N\}$ avec $\{U_1 \cdots U_N\} \cap V = \emptyset$. On double en fait le nombre de sommets par rapport au graphe initial pour garantir que l'on ait un nombre pair de sommets et pour ainsi pouvoir former la partition en ensembles de tailles égales.

On relie alors dans G' uniquement les sommets qui ne sont pas reliés dans G (les sommets ajoutés forment une clique) ie $E' = \{(u, v) : u, v \in V', (u, v) \notin E\}$. Et on fixe $W' = N^2 - W$.

Supposons alors que $\mathcal{I}_1$ possède une coupe solution : $S_1 : S_2$ telle que $|S_1 : S_2| \geqslant W \geqslant 1$, on a S_1 et S_2 non vides. On complète S_1 et S_2 avec les sommets U_i de sorte à former S_1' et S_2' de même taille.

On va vérifier que $S_1' : S_2'$ forme une bissection de poids plus petit que W'

$$|S_1' : S_2'| = N^2 - \left|\{(u, v) \notin E' : u \in S_1', v \in S_2'\}\right| = N^2 - |\{(u, v) \in E : u \in S_1, v \in S_2\}|$$
$$\leqslant N^2 - W = W'$$

Ainsi si $\mathcal{I}_1$ a une solution alors $\mathcal{I}_2$ a une solution. Montrons maintenant la réciproque.

Supposons que $\mathcal{I}_2$ ait une solution $|S'_1 : S'_2| \leqslant W' = N^2 - W$. On définit alors $S_1 = S'_1 \cap V$ et $S_2 = S'_2 \cap V$. On va voir que $S_1 : S_2$ est une solution de $\mathcal{I}_1$.

En effet :

$$\begin{aligned} |S_1 : S_2| = \left|\{(u,v) \notin E' : u \in S'_1, v \in S'_2\}\right| &= N^2 - \left|\{(u,v) \in E' : u \in S'_1, v \in S'_2\}\right| \\ &\geqslant N^2 - (N^2 - W) = W \end{aligned}$$

Ainsi si $\mathcal{I}_2$ a une solution alors $\mathcal{I}_1$ a une solution.

De plus la construction présentée est polynomiale en la taille du problème, donc c'est une réduction polynomiale. On conclut donc que tout problème de *SIMPLE-MAX-CUT* peut être transformé en un problème équivalent de *SIMPLE-MIN-BISSECTION* qui est donc $\mathcal{NP}$-difficile. □

On a alors montré que le problème de décision de la bissection minimale était bien $\mathcal{NP}$ et $\mathcal{NP}$-difficile, il est donc bien $\mathcal{NP}$-complet.

4 Pour aller plus loin

Nous avons vu jusqu'ici que si trouver la bissection minimale permettait de reconstruire de manière exacte les communautés, ce problème est intractable dès que les graphes considérés deviennent grands. Il existe d'autreS méthodes probabilistes permettant de résoudre le problème avec différentes garanties. L'article de Dyier et Frieze[1] propose d'abord un algorithme permettant d'obtenir la coupe minimale de manière sûre mais qui s'exécute en temps polynomial en espérance puis un algorithme polynomial (en $\mathcal{O}(N^3)$) – vu en cours – sous certaines conditions. D'autres solutions consistent à travailler sur des problèmes affaiblies de reconstruction non exacte comme vu en cours.

De manière plus générale, on cherche à contourner la complexité des problèmes $\mathcal{NP}$-difficile en utilisant des algorithmes probabilistes de type *Monte-Carlo* (termine en temps déterministe mais la sortie est probabiliste) ou *Atlantic city* (tourne en temps probabiliste, on précise l'espérance du temps de calcul mais donne toujours un résultat correct), ou encore en ayant recours à des algorithmes approximations dont la sortie est garantie être proche de l'optimum.

Dans le cas du problème *MIN-BISSECTION* général (c'est à dire avec des arêtes pondérées et on cherche la bissection qui minimise le flot entre les deux parties), on peut en fait montrer que ce problème n'est même pas approximable en temps raisonnable à moins que $\mathcal{P} = \mathcal{NP}$[2].

Plus précisément, on dit qu'un problème est approximable en temps polynomial, s'il existe un algorithme, tel que pour tout ϵ, cet algorithme retourne une solution optimale à un facteur $1 + \epsilon$ près en temps polynomial. Dans notre cas, si une bissection minimale à un coût de C, une ϵ-approximation de ce problème est un algorithme retournant un résultat R tel que $C \leqslant R \leqslant C(1+\epsilon)$. Et donc, pour cette définition, le problème *MIN-BISSECTION* n'est pas approximable – sauf si $\mathcal{P} = \mathcal{NP}$. On montre en fait que, s'il existe une telle approximation, on peut l'utiliser pour résoudre un problème connu pour être $\mathcal{NP}$-complet par réduction entre problèmes.

Par ailleurs, on peut simplifier le problème en relâchant les contraintes en cherchant non plus à reconstruire les communautés de manière exacte mais en autorisant des erreurs dans le cadre de la reconstruction faible vue en cours.

Références

[1] M. E. Dyier A. M. Frieze. The solution to some random np hard problems in polynomial expected time. *Complexity of Computer Computations, Plenum*, 1972.

[2] Konstantin Andreev and Harald Räcke. Balanced graph partitioning. In *Proceedings of the Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA 04, page 120124, New York, NY, USA, 2004. Association for Computing Machinery.

[3] Aydin Buluc, Henning Meyerhenke, Ilya Safro, Peter Sanders, and Christian Schulz. Recent advances in graph partitioning, 2013.

[4] Stephen Cook. The complexity of theorem proving procedures. *Proceedings of the third annual ACM symposium on Theory of computing*, 1971.

[5] Michael Fleder, Michael Kester, and Sudeep Pillai. Bitcoin transaction graph analysis. 02 2015.

[6] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified np-complete problems. In *Proceedings of the Sixth Annual ACM Symposium on Theory of Computing*, STOC 74, page 4763, New York, NY, USA, 1974. Association for Computing Machinery.

[7] Richard M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations, Plenum*, 1972.

[8] Christos Papadimitriou. *Combinatorial optimization : algorithms and complexity*. Prentice Hall, Englewood Cliffs, N.J, 1982.

[9] Denis Trystram. The solution to some random np hard problems in polynomial expected time.