Séance 5 et 6 : Kernels

Alix Munier-Kordon et Claire Hanen

Décembre 2024

Ce cours est basé sur les deux livres [1, 2] Nous traiterons tout d'abord deux exemples avant d'introduire plus rigoureusement la notion de Kernel. Nous verrons ensuite un exemple de résultat général qui peut être utilisé pour construire les Kernels de plusieurs problèmes.

1 Deux exemples

Dans cette partie nous allons aborder deux problèmes :

k EDGE CLIQUE COVER

Input : Un graphe non orienté G, un entier k ;

Question : Existe-t-il k cliques sous-graphes de G, $H_1, \ldots, H_k$ dont la réunion des arêtes est l'ensemble des arêtes de G ? - toutes les arêtes de G sont donc couvertes par les cliques.

k VERTEX COVER

Input : un graphe non orienté G un entier $k > 0$;

Question : Existe-t-il un sous ensemble X de cardinal au plus k tel que toute arête a au moins une extrémité dans X ?

Nous allons voir que des règles de prétraitement polynômiales permettent soit de conclure sur la réponse à la question, soit de réduire la taille de l'instance.

1.1 k EDGE CLIQUE COVER

Notons tout d'abord que si G comporte des sommets isolés, la réponse à la question sera la même si l'on considère G' obtenu en enlevant de G les sommets isolés.

reduction ECC1 : retirer de G les sommets isolés.

Considérons maintenant la propriété suivante :

Propriété 1. *Si l'une des composantes connexes de G est réduite à une seule arête $e = \{i, j\}$, alors dans toute couverture des arêtes de G par k cliques, cette arête et ses deux extrémités fait partie des sous-graphes $H_1, \ldots, H_k$*

Démonstration. l'arête e forme à elle même une clique. Et elle ne peut faire partie d'une clique plus large. Par conséquent l'un des sous-graphes H_u correspond à cette arête □

reduction ECC2 On déduit de cette propriété que la réponse à la question EDGE CLIQUE COVER(k, G) est non si $k = 0$, et que si une composante connexe de G est réduite à une arête e, alors en considérant le graphe G' obtenu en enlevant e et ses extrémités de G, la réponse à la question est exactement EDGE CLIQUE COVER$(k-1, G')$

On peut donc maintenant supposer que dans notre instance, il n'y a pas de sommets isolés ni de composantes connexe réduite à une arête.

Propriété 2. *S'il existe une arête de G $\{i,j\}$ telle que i et j ont exactement les mêmes voisins, c'est à dire $\Gamma(i)-\{j\}=\Gamma(j)-\{i\}\neq\emptyset$. soit $H_1,\dots H_k$ une couverture des arêtes de $G-\{j\}$ par une clique. Alors on peut ajouter j à la composante connexe de i et cela forme une couverture des arêtes de G par une clique. Réciproquement, si l'on a une couverture des arêtes de G, alors en enlevant $\{j\}$ on forme une couverture des arêtes de $G-\{j\}$*

reduction ECC3 On déduit de cette propriété que si deux extrémités d'une arête ont les mêmes voisins, on peut supprimer l'une des deux extrémités. Notons que cette réduction peut réduire le nombre d'arêtes dans les composantes connexes et qu'en conséquence la règle ECC2 peut être de nouveau appliquée. Mais comme soit on enlève 1 ou deux sommets à chaque fois il ne peut y avoir qu'au plus $\mathcal{O}(n)$ réductions.

Propriété 3. *En appliquant itérativement les règles $ECC1, ECC2, ECC3$ sur un graphe G, on obtient en temps polynomial soit la réponse à la question, soit un graphe G' d'au plus 2^k sommets.*

Démonstration. Montrons que lorsqu'on a une instance pour laquelle la réponse est oui, alors le graphe G' a au plus 2^k sommets. Donnons nous donc une couverture par k Cliques $H_1,\dots,H_k$ des arêtes de G.

Supposons par contradiction que G' a plus de 2^k sommets. Associons à chaque sommet i un vecteur binaire de k bits indiquant dans quelle(s) clique $H_1,\dots,H_k$ il se trouve. Considérons deus sommets i et j qui ont le même vecteur binaire. Ils sont donc contenus dans les mêmes cliques et $\{i,j\}$ est une arête. Mais alors alors i et j ont exactement les mêmes voisins (dans chaque clique). La règle $ECC2$ ou $ECC3$ devrait donc s'appliquer, contradiction. □

On voit donc ici un premier exemple de règles de réduction qui conduisent en temps polynomial soit à une réponse, soit à une instance de taille $f(k)$, pour laquelle il est ensuite possible d'opérer une énumération (la plus intelligente possible) mais qui conduit naturellement à un algorithme FPT

1.2 k VERTEX COVER

Considérons maintenant le problème k VERTEX COVER. On définit les règles de réduction suivantes :

Réduction VC1 Si G contient un sommet isolé, le retirer.

Réduction VC2 Si G a un sommet i de degré au moins $k+1$ alors , si $k=1$ la réponse est fausse, sinon résoudre le problème pour le graphe $G'=G-\{i\}$ avec pour valeur de paramètre $k-1$.

Propriété 4. *L'application itérée des deux règles de réduction conduit à un graphe G' qui comporte au plus $k(k+1)$ sommets et k^2 arêtes ou à une réponse à la question posée*

Démonstration. L'application itérée des règles de réduction permet d'obtenir un graphe G' dont tous les sommets sont de degrés inférieurs ou égal à k et qui ne comporte pas de sommets isolés. Supposons que la réponse à la quesiton soit oui. Soit X un sous-ensemble d'au plus k sommets qui couvre toutes les arêtes. Si $x\notin X$ est un sommet du graphe, tous ses voisins sont dans X. On a donc au plus $k\times|X|$ tels sommets (en supposant que tous les sommets de k ont k voisins dans $V'-X$). il en résulte que si la réponse est oui le nombre total de sommets est inférieur ou égal à $k(k+1)$. Maintenant, chaque sommet de X peut couvrir au plus k arêtes. Il ne peut donc y avoir, dans une instance où la réponse est oui, plus de k^2 arêtes dans G'. □

Cet exemple montre ici qu'en temps polynomial par rapport au nombre de sommets de graphe, on peut obtenir soit la réponse à la question, soit une instance dont la taille est **polynômiale** en k.

On appelle l'instance réduite le kernel (noyau) du problème paramétré, et on dit qu'il s'agit d'un kernel polynômial lorsque comme pour vertex cover, la taille de l'instance réduite est polynômiale en k.

Cela ouvre sur des questions générales pour les problèmes paramétrés : Quels sont les problèmes pour lesquels il existe des kernels ? Y a-t-il un rapport avec FPT ? Quels sont les problèmes pour lesquels le kernel est polynômial.

2 Kernels : définitions

Donnons maintenant les principales définitions relatives aux Kernels.

Définition 1. *Un algorithme de kernelization ou un kernel (noyau) pour un problème Q paramétré est un algorithme $\mathcal{A}$, qui à partir d'une instance (I, k) du problème Q, en temps polynômial par rapport à $|I|$, soit résoud le problème soit produit une instance équivalente (au sens où la réponse au problème est la même) (I', k') de Q avec $|I'| + k' \leq g(k)$ pour une fonction calculable g.*

Autrement dit, la taille de l'instance résultante ne dépend que du paramètre.

Définition 2. *Un algorithme de kernelization est dit polynômial (ou le kernel est dit polynômial) lorsque la fonction g est un polynôme*

Dans les exemples précédents, nous avons bien deux algorithmes de kernelization, mais seul celui pour VERTEX COVER est polynomial. L'une des hypothèses (raisonnable) de la théorie de la complexité paramétrée est que k EDGE CLIQUE COVER n'admet pas de kernelization polynômiale.
On peut faire le lien entre l'existence d'un algorithme FPT pour résoudre un problème Q et l'existence d'un algorithme de kernelization.

Théorème 1. *un problème paramétré décidable Q admet un algorithme de kernelization si et seulement si il peut être résolu par un algorithme FPT*

Démonstration. $\implies$ Supposons tout d'abord que Q admet un algorithme de kernelization. Alors soit on a réussi à résoudre Q en temps polynômial, soit il reste à décider pour l'instance (I', k') dont la taille dépend uniquement de k et plus de la taille de l'instance (I, k). Ainsi, il sera calculé en temps $f(k)$ calculable si Q est un problème calculable.

$\impliedby$ Supposons que Q possède un algorithme FPT $\mathcal{A}$. soit (I, k) une instance de Q. La complexité de $\mathcal{A}$ sur l'instance est donc bornée par $f(k)|I + k|^c$ où c est une constante. Appliquons l'algorithme $\mathcal{A}$ à l'instance (I, k) pendant $|I|^{c+1}$ étapes. S'il se termine avec une réponse c'est que le problème est résolu en temps polynômial. Sinon, c'est que $f(k)|I|^c \geq |I|^{c+1}$ et donc que $|I| \leq f(k)$. On a donc un kernel de taille $f(k) + k$. □

La portée de ce résultat n'est pas d'ordre pratique. Il indique juste que la kernelization est un autre moyen de montrer qu'un problème est FPT. Nous avons vu autour des deux exemples que les kernels se construisent avec des règles de réduction.

Définition 3. *Une règle de réduction Φ d'un problème paramétré Q est une application calculable en temps polynomial (en $|I|$) qui à toute instance (I, k) soit associe une instance $\Phi(I, k) = (I', k')$*

Habituellement les règles vérifient que $k' \leq k$ et ou $|I'| \leq |I|$ mais ce n'est pas une obligation.

Définition 4. *Une règle de réduction Φ est dite* **valide** *si la réponse à (I, k) est oui si et seulement si la réponse à $\Phi(I', k')$ est oui.*

3 Décomposition en couronne (Crown decomposition)

Nous étudions maintenant l'application d'un outil générique (il en existe de nombreux dans les graphes) qui permet d'obtenir des algorithmes de kernelization. Nous allons étudier avec cet outil de problème suivant :

k MAX-SAT

Input : n variables binaires, m clauses disjonctives, un entier $k > 0$;

Question : Existe-t-il une instanciation des variables de sorte qu'au moins k clauses soient vraies ?

Introduisons tout d'abord les résultats de théorie des graphes dont nous aurons besoin. On vous demande d'admettre les deux théorèmes suivants :

Théorème 2. *(Hall's theorem). Soit G est un graphe biparti basé sur les sous-ensembles de sommets $V_1, V2$. Il existe un couplage qui sature V_1 si et seulement si pour tout sous ensemble $X \subseteq V_1$ de sommets, Le nombre des voisins de X ($\Gamma(X)$) est au moins égal au nombre de sommets de X ($|\Gamma(X)| \geq |X|$)*

Il en découle un algorithme (Hopcroft et Karp) :

Théorème 3. *Soit G est un graphe biparti basé sur les sous-ensembles de sommets $V_1, V2$. L'algorithme de Hopcroft et Karp calcule soit un couplage qui sature V_1 ou exhibe un sous-ensemble $X \subseteq V_1$ minimum par inclusion qui vérifie $|\Gamma(X)| < |X|$. Sa complexité est $\mathcal{O}(m\sqrt{n})$*

Définition 5. *Une* **décomposition en couronne** *d'un graphe non orienté G est une partition de ses sommets en trois sous-ensembles C, H, R de sorte que :*

- *C est non vide.*
- *C est un stable*
- *il n'y a pas d'arêtes entre C et R.*
- *il existe un couplage de taille $|H|$ dans le sous graphe induit par H et par C*

Propriété 5. *Si l'on a un graphe biparti G avec les deux ensembles de sommets V_1, V_2 et qu'en appliquant l'algorithme de Hopcroft et Karp on obtienne un sous ensemble $C \subseteq V_1$ tel que $|\Gamma(C)| < |X|$. Alors G possède une décomposition en couronne.*

Démonstration. Posons $H = \Gamma(C)$. et R l'ensemble des autres sommets. Observons que C est non vide. Comme $C \subseteq V_1$ et que G est biparti, aucune arête ne relie deux sommets de C. Il ne peut pas non plus y avoir d'arête entre C et R car aucun voisin de C ne fait partie de R. Considérons maintenant la notion de couplage. Prenons un sommet x de C. Puisque l'ensemble C est minimal pour l'inclusion on a $|\Gamma(C - \{x\})| \geq |C - \{x\}|$ et c'est aussi le cas pour tous ses sous-ensembles. Ainsi, d'après le théorème de Hall, il existe bien un couplage qui sature $C_{\{}x\}$. Maintenant $|\Gamma(C)| < |C|$ et $|\Gamma(C - \{x\})| \geq |C| - 1$. il en résulte que $|\Gamma(C - \{x\})| = |\Gamma(C)| = |C| - 1 = |H|$. Ainsi ce couplage est de taille $|H|$ et l'on a bien une décomposition en couronne de G □

Revenons maintenant à notre problème κ MAX-SAT. Tout d'abord on peut montrer (à faire en TD) le résultat suivant :

Propriété 6. *Si $k < \frac{m}{2}$ alors le problème est toujours satisfiable.*

Réduction MS1 si $k < \frac{m}{2}$ l'instance est satisfiable, étudier uniquement les cas $k \geq \frac{m}{2}$

Théorème 4. *κ MAX-SAT a un kernel avec au plus k variables et $2k$ clauses.*

Démonstration. Considérons le graphe biparti construit de la manière suivante : le premier ensemble de sommets V_1 est associé aux variables et le second V_2 aux clauses. On ajoute un arc dans G entre une variable et une clause si elle y apparait (ou sa négation).

Supposons qu'on trouve dans G un couplage qui sature V_1 (donc de taille n). Alors on peut toujours trouver une instanciation des variables qui satisfait n clauses. Si $k \leq n$ l'instance est donc satisfiable dans ce cas (il suffit de prendre k arêtes dans le couplage). Si $k > n$ le nombre de variables est borné par k et si le nombre de clauses est par hypothèse $\leq 2k$ et l'on a donc une instance kernel dont la taille ne dépend que du paramètre.

Réduction MS2 S'il existe un couplage qui sature V_1 alors si $n \geq k$ alors l'instance est satisfiable sinon l'instance est bien de taille bornée $n \leq k, m \leq 2k$.

Si aucun couplage ne permet de saturer V_1, alors on a vu dans la proposition 5 que l'on pouvait construire en temps polynomial une décomposition en couronne (C, H, R) de G, où C est un ensemble de variables et H un ensemble de clauses, de sorte q'une clause en dehors de H ne concerne aucune des variables de C. Observons que si $k \leq |H|$ alors l'instance est satsifiable.

Dans le cas contraire, supposons qu'il existe une instanciation des variables telle qu'au moins k clauses soient satisfaites. Prenons l'instanciation qui satisfait le maximum de clauses et montrons qu'elle satisfait toutes les clauses de H. En effet si ce n'est pas le cas, il suffit d'utiliser le couplage de C, H de taille $|H|$ de la décomposition en couronne, pour simplement augmenter le nombre de clauses satisfaites (en mettant à vrai le littéral correspondant dans la clause couplée à la variable). Par conséquent, la réponse à la question se trouve par dans l'existence ou non d'une instanciation des variables de $V_1 - C$ qui satisfait au moins $k - |H|$ clauses parmi $V_2 - H$.

Réduction MS3 S'il n'existe pas de couplage qui sature V_1 alors calculer la décomposition en couronne (C, H, R) et si $|H| \geq k$ s'instance est satisfiable. Sinon Constituer l'instance des clauses de $V_2 - H$ et les variables de $V_1 - C$, pour la valeur de paramètre $k - |H|$

On peut alors recommencer à appliquer les trois réductions itérativement jusqu'à un point fixe. Le nombre d'itération est au plus n puisqu'à chaque réduction d'instance on réduit le nombre de variables de 1. Ainsi la complexité du processus de réduction est $\mathcal{O}(mn\sqrt{n})$. □

Références

[1] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Daniel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer Publishing Company, Incorporated, 1st edition, 2015.

[2] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization : Theory of Parameterized Preprocessing*. Cambridge University Press, 2019.