M2 MPRO

Complexité paramétrée & Approximation Polynomiale (CAP)

Partie 2 : Complexité Paramétrée
& Algorithmes FPT

Dimitri Watel (ENSIIE)
& Cédric Bentz (CNAM)

A quoi ressemble « vraiment » NPO ?

- Du point de vue de la complexité « classique », un problème de NPO est, en général, soit NP-difficile (au sens fort ou non), soit dans PO
- Deux problèmes NP-difficiles sont-ils donc « aussi » difficiles l'un que l'autre ?
- L'Approximation Polynomiale (AP) offre ***un*** moyen d'affiner la hiérarchie entre les différents problèmes NP-difficiles de NPO
- La complexité paramétrée en offre un autre, qui s'applique également (et, en réalité, essentiellement) à NP

Brefs rappels sur l'AP

- STABLE et TRANSVERSAL sont tous 2 (fortement) NP-difficiles en général
- STABLE ⇔ TRANSVERSAL du point de vue de la ***résolution exacte***
- Mais TRANSVERSAL admet plusieurs algorithmes 2-approchés
- STABLE, lui, n'admet aucun algorithme O(1)-approché, sauf si P=NP !

La classe FPT (1/3)

- On considère un problème Π de NP
- On suppose en plus la donnée d'un paramètre k (indépendant de l'instance) pour Π, qui devient alors « **paramétré** »
- Par exemple, soit DCLIQUE (qui est dans NP) la version décision de CLIQUE :
  - Instance : un graphe G=(V,E), un entier k
  - Question : existe-t-il dans G une clique d'ordre au moins k ?

La classe FPT (2/3)

- k-CLIQUE, la version « paramétrée » de DCLIQUE, est alors la suivante :
  - Instance : un graphe G=(V,E)
  - Paramètre : un entier k
  - Question : existe-t-il dans G une clique d'ordre au moins k ?
- Pourquoi cette distinction ?
  - On considère que le paramètre k prendra de « petites » valeurs en pratique

La classe FPT (3/3)

- Pour un problème Π paramétré (par k), on dit qu'un algorithme qui résout Π est **FPT** (vis-à-vis de k) si son temps d'exécution est de la forme $O(f(k)|I|^c)$, où :
  - f est une fonction « raisonnable » de k (donnée explicitement ou calculable en temps polynomial) : si f est polynomiale, alors l'algorithme l'est aussi !
  - $|I|$ = taille du codage de l'instance I de Π
  - c = constante (indépendante de k)
- Si Π admet un tel algorithme, alors on dit que Π est FPT (ou dans FPT) vis-à-vis de k
- Classe FPT = **Fixed-Parameter Tractable**

FPT vs non-FPT : quel intérêt ?

- Un algorithme s'exécutant en $O(f(k)|I|^c)$ est FPT, et donc, en théorie, efficace
- En pratique, on espère que $f(k)$ n'est pas de la forme $\exp(\exp(k))$ (ou pire)
- Dans certains domaines (ex. bioinfo.), les hypothèses « f pas trop grande » et « k petit » sont raisonnables et vérifiées, mais :
  - $O(2^k|I|^{10})$ est FPT, alors que $O(|I|^{\log k})$ ne l'est pas
  - D'un autre côté, $2^k|I|$ est effectivement beaucoup plus petit que $|I|^k$ pour $|I|>>2$!

Paramétrisation de problèmes

- En réalité, k peut aussi être un vecteur de paramètres (en général, pas plus de 2 !)
- Un problème peut être FPT vis-à-vis d'un (ensemble de) paramètre(s), mais ne pas l'être vis-à-vis d'un autre
- Beaucoup de problèmes NP-difficiles deviennent « faciles » si un paramètre k est fixé, mais pas nécessairement FPT !
  - Par exemple, k-CLIQUE peut être résolu par « force brute » en temps $O(|V|^{k+2})$
  - **On dit que k-CLIQUE est dans la classe XP**

Hypothèse de base (similaire à P≠NP)

- L'hypothèse suivante est à la base de la « hiérarchie W[t] » : k-CLIQUE n'est ***pas*** FPT (vis-à-vis de k)
- Hiérarchie W[t] :
  - On a $(P\subseteq)FPT=W[0]\subseteq W[1]\subseteq W[2]\subseteq \ldots \subseteq XP$
  - On conjecture que $W[0]\subset W[1]\subset W[2]\subset \ldots \subset XP$
- k-CLIQUE est XP mais W[1]-difficile
  - Si k-CLIQUE s'avère en réalité FPT, alors W[1]=FPT !
  - On a donc P=NP ➔ FPT=W[1]

FPT-réductions et équivalence entre problèmes paramétrés

- Définition (**FPT-réduction**) :

  Un problème paramétré Π_k se FPT-réduit à un problème paramétré $\Pi'_{k'}$ s'il existe un algorithme ***A*** s'exécutant en temps $O(f(k)|I|^c)$ (où $|I|$=taille du codage de l'instance I de Π_k) tel que :

  - ***A*** transforme toute instance I de Π_k en une instance I' de $\Pi'_{k'}$, à laquelle la réponse est '*oui*' ssi la réponse pour I est '*oui*' (instances équivalentes),
  - $k' \leq g(k)$ pour une certaine fonction g raisonnable.

- Conséquences (pour de tels Π_k et $\Pi'_{k'}$) :
  - Si Π_k n'est pas FPT, alors $\Pi'_{k'}$ ne l'est pas non plus.
  - Si $\Pi'_{k'}$ est FPT, alors Π_k l'est également.
  - Si Π_k se FPT-réduit à $\Pi'_{k'}$ et si $\Pi'_{k'}$ se FPT-réduit à Π_k, alors Π_k et $\Pi'_{k'}$ sont FPT-équivalents.

Exemples de réductions simples

- k-STABLE et k-CLIQUE sont trivialement FPT-équivalents !
  - Par passage au graphe complémentaire.
- La réduction « classique » entre STABLE et TRANSVERSAL n'est ***pas*** une FPT-réduction entre k-STABLE et k-TRANSVERSAL !
  - En revanche, c'est une FPT-réduction (et même une FPT-équivalence) triviale entre k-STABLE et le problème paramétré suivant : étant donnés un graphe G=(V,E) et un paramètre k, existe-t-il dans G un transversal d'ordre au plus |V|-k ?

Un premier résultat : k-TRANSVERSAL est FPT (via arbres de recherche bornés)

- Résolution de k-TRANSVERSAL via un ARB=arbre de recherche borné (bounded search tree) :
  - Pour chaque arête uv de G=(V,E), soit u est dans le transversal, soit v y est (ou les deux)
  - On va donc « brancher » sur u, puis sur v :
    - A chaque étape, on inclut le sommet sélectionné dans le transversal, puis on enlève toutes les arêtes incidentes, et on recommence
    - La profondeur de l'arbre de recherche obtenu est bornée par k (sinon, on répond 'non'), et son nombre de feuilles est donc borné par 2^k
    - Le temps d'exécution est en $O(2^k|V|)$ (on enlève au plus $|V|-1$ arêtes à chaque étape) ➔ FPT
- k-TRANSVERSAL est FPT, k-CLIQUE non !
- Décider s'il n'existe ***pas*** de TRANSVERSAL de taille au plus k est aussi FPT !

Lien avec les schémas d'approximation

- Théorème : un problème paramétré (à valeurs entières) du type « existe-t-il une solution admissible de taille $\geq$ k ? » (resp. « de taille $\leq$ k ? ») qui est dans **EPTAS** (PTAS en temps $O(f(1/\varepsilon)|I|^c)$) est FPT.
  - Preuve (pour un problème en *MAX*) : utiliser le EPTAS avec $\varepsilon = 1/2k \leq 1/2$ (car k est un entier > 0), et noter APP la valeur de la solution obtenue.
    - On a, comme corollaire de la définition d'EPTAS (?) : $APP \leq OPT \leq (1+2\varepsilon)APP = (1+1/k)APP$.
    - Si $k \leq APP$, alors $k \leq OPT$ ➔ répondre 'oui'.
    - Si $k > APP$, alors :

      $APP \leq OPT \leq (1+1/k)APP < (1+1/APP)APP = APP+1$

      Donc $OPT = APP<k$ ➔ répondre 'non'.
    - Temps d'exécution = $O(f(1/\varepsilon)|I|^c) = O(f(2k)|I|^c)$
- Si problème en *MIN*, on choisit $\varepsilon = 1/(k+2)$.
- L'inverse est faux : k-TRANSVERSAL est FPT, mais n'admet pas de (E)PTAS, sauf si P=NP !

Pour faire mieux : la technique de « kernelization » (« noyautage »)

- Un « **kernel** » (ou « **noyau** ») est une instance « réduite » d'un problème paramétré, qui contient toutes les informations « essentielles »
- Plus formellement, étant donnée une instance I d'un problème paramétré Π_k, un noyau de I est une instance I' de $\Pi_{k'}$ qui vérifie les propriétés suivantes :
  - $k' \leq k$,
  - On peut calculer I' à partir de I en temps **polynomial**, et la réponse pour l'une est 'oui' ssi celle pour l'autre est 'oui' (instances équivalentes),
  - $|I'| \leq g(k)$, pour une certaine fonction g raisonnable.
- Liens entre noyaux et algorithmes FPT ?

Equivalence entre l'appartenance à FPT et l'existence d'un noyau

- En réalité, l'existence d'un noyau pour un problème de NP est une CNS pour son appartenance à FPT !
  - On suppose ici l'existence d'instances triviales 'oui' et 'non' pour tout k,
  - Vrai en pratique (cf k-TRANSVERSAL).
- Théorème : un problème paramétré de NP est FPT (en $f(k)|I|^c$) $\Leftrightarrow$ il admet un noyau.
  - $\leftarrow$: recherche exhaustive en $O(2^{poly(g(k))})$
  - $\rightarrow$: $|I| > f(k)$ (on résout) ou $|I| \leq f(k)$ (noyau)
- Cela ne garantit pas la taille du noyau !

Un exemple de noyau pour le problème paramétré k-TRANSVERSAL

- On part d'une instance I :
  - Règle 1 : on sélectionne tous les sommets de degré >k dans le transversal, et on enlève les arêtes qui y sont incidentes (on met aussi à jour les degrés et la valeur de k),
  - Règle 2 : on enlève les sommets isolés.
  - L'instance I' obtenue a au plus k^2 arêtes (sinon, on répond 'non') et $2k^2$ sommets.
  - C'est donc un noyau pour I !

Obtention d'un algorithme FPT pour k-TRANSVERSAL à l'aide du noyautage

- On résout I' par « force brute »
  - Temps d'exécution en $O(n+m+2^k k^{2(k+1)})$, où $n=|V|$ et $m=|E|$
- Comment faire mieux ?
  - C'est simple : combiner noyautage et arbre de recherche borné !
  - On résout I' par cette technique
  - Temps d'exécution en $O(n+m+2^k k^2)$

Existence d'un noyau linéaire pour k-TRANSVERSAL (1/3)

- En fait, non seulement il existe un noyau **polynomial** ($O(k^2)$) pour k-TRANSVERSAL, mais il existe même un noyau **linéaire** (3k)
- Utilise la notion de « crown decomposition » (décomposition couronnée), qui est une partition (C,H,R) des sommets du graphe où
  - C induit un ensemble stable,
  - Il n'y a pas d'arêtes entre C et R,
  - Il existe un couplage entre C et H qui sature H.

Existence d'un noyau linéaire pour k-TRANSVERSAL (2/3)

- Si une telle partition (C,H,R) existe pour G, alors G a un transversal de taille au plus k ssi le sous-graphe de G induit par V(G)-(C+H) en a un de taille au plus k-|H|
- On utilise alors l'algorithme suivant :
  - Tant que non STOP
    - Trouver un couplage maximum M de G
    - Si |M|>k alors répondre 'non'; STOP;
    - Sinon
      - Si |V(G\M)|≤k alors G est un 3k-noyau; STOP;
      - Sinon, trouver une décomposition couronnée (C,H,R) de G, et réduire l'instance (***crown reduction***) en retirant C et H (et tous les nouveaux sommets isolés)

Existence d'un noyau linéaire pour k-TRANSVERSAL (3/3)

- Il reste à montrer que, si $|M| \leq k$ et $|V(G \backslash M)| > k$, alors la partition (C,H,R) existe, et peut être trouvée en temps polynomial
  - Soit M' un couplage maximum entre $X = V(M)$ et $S = V(G \backslash M)$ (S est un ensemble stable)
  - On sait qu'il existe un transversal T de taille $|M'|$ dans ce graphe biparti (et comment le trouver)
  - Chaque arête de M' a donc exactement une extrémité dans T (soit dans X, soit dans S)
  - Si $|X \cap T| = 0$, alors $T = S$ (car S n'a aucun sommet isolé) ➔ $|M'| = |T| = |S| > k \geq |M|$: contradiction !
  - Sinon, soit $S' = \text{voisins}_{M'}(X \cap T)$: alors, la partition $(S', X \cap T, (X - X \cap T) + (S - S'))$ est une décomposition couronnée de G, car $|S' \cap T| = |(X - X \cap T) \cap T| = 0$

Borne inférieure sur la taille d'un noyau k-TRANSVERSAL : liens avec l'AP

- Avec un autre type d'approche, on peut réussir à déterminer un noyau de taille 2k pour k-TRANSVERSAL
- Il est peu probable qu'on puisse faire mieux
  - Car trouver un algorithme α-approché pour TRANSVERSAL avec $\alpha<2$ est un problème **ouvert** depuis longtemps (NP-difficile pour $\alpha<1.36$)
  - Car un αk-noyau impliquerait un tel algorithme :
    - Pour k de 1 à |V(G)| faire
      - Calculer un αk-noyau (*en fait, αk'-noyau*) et renvoyer comme solution potentielle tous ses sommets
    - Garder la meilleure solution (pour k=opt≤|V(G)|)

Noyau linéaire pour k-STABLE dans les graphes planaires

- Rappel : en général, k-STABLE et k-CLIQUE sont W[1]-difficiles
- Pourtant, dans les graphes planaires, k-CLIQUE est trivial, et **construire un noyau linéaire** pour k-STABLE est une tâche relativement aisée, sous réserve d'admettre le théorème des 4 couleurs :
  - *Observation* : le plus grand stable au sein d'une 4-coloration contient au moins ¼ des sommets.
  - *Conséquence immédiate* : 4k-noyau via le théorème des 4 couleurs.
  - *Remarque* : noyau en temps O(1) !

Noyau pour (k,Δ)-STABLE

- En général, **(k,Δ)-STABLE est en fait FPT** (où Δ=degré maximum du graphe)
- k(Δ+1)-noyau :
  - $\chi(G) \leq \Delta+1$ ➔ on répond 'oui' si $k \leq |V|/\Delta+1$, et on a un k(Δ+1)-noyau sinon
- Algorithme FPT via ARB :
  - Pour tout sommet v, un stable maximal contient v ou un de ses (< Δ+1) voisins
  - A chaque étape, on enlève au plus Δ sommets et on a au plus Δ+1 branchements possibles
  - L'algorithme s'exécute donc en temps $O(\Delta(\Delta+1)^k)$

Paramétrisations de (MAX) SAT ?

- Exercices :
  - 6k-noyau pour k-MAX 3SAT (« Peut-on satisfaire au moins k clauses d'une instance de 3SAT ? »)
    - Toute clause est satisfaite par un choix de valeurs de vérité ou par son opposé (*)
  - $O(k^2)$-noyau pour k-MAX SAT
    - Le nombre de variables ne dépend pas de k !
    - Retirer les clauses triviales (toujours satisfaites) et les clauses « longues » (de taille >k)
    - Enfin, utiliser (*) de nouveau

Bilan

- La complexité paramétrée offre un cadre pour affiner la « hiérarchie » entre les problèmes NP-difficiles :
  - k-TRANSVERSAL (FPT) $\leq$ k-CLIQUE (W[1]-difficile) (on a « < » sous l'hypothèse FPT$\neq$W[1])
- Différentes techniques de conception d'algorithmes FPT :
  - ARB
  - Noyaux (et les méthodes pour les obtenir)
- Liens avec l'AP


"That's all Folks!"