Algorithmique avancée
Introduction à la complexité
Une note culturelle

Denis TRYSTRAM
ENSIMAG Alternants 2A

Oct. 2021

Modèles de Calcul

Les informaticiens, dont un des objectifs est de s'occuper de *calcul*, ont eu besoin de formaliser cette notion, pour classifier les problèmes qui peuvent être résolus à l'aide d'un ordinateur.

La vision de mathématiciens n'est pas la même que celle des programmeurs.

- Une fonction f est un ensemble de couples (a, b) tels que si $(a, b) \in f$ avec $a \in A$ et $b \in B$ et $(a, c) \in f$ alors $b = c$.
- Un programmeur/informaticien préfère plutôt définir les fonctions par des *algorithmes*, c'est-à-dire des méthodes/procédures qui les calculent. Ainsi, étant donné a on cherche une méthode qui calcule $b = f(a)$.

La question importante est

déterminer quelles fonctions sont calculables par une machine donnée ?

pour cela, il est nécessaire de définir ce qu'est un *modèle de calcul* et si possible, un qui soit *universel*.

La machine de Turing (TM)

- C'est le modèle de calcul fondamental le plus *simple*.

C'est à partir d'exécutions sur une TM que l'on définit la notion de complexité (en temps, par le nombre de transitions de l'automate) et en espace (nombre de cases élémentaire utilisées lors de l'exécution).

Notion de problème

On considère les problèmes sous forme de décision (c'est-à-dire dont la réponse est OUI ou NON).

Considérons par exemple le problème du test de primalité d'un nombre :

- PRIME
- **Instance :** un entier N
- **question** N est-il premier ?

Codage de l'entrée

Attention, soulignons que la complexité d'un problème est définie par rapport à un codage naturel de ses instances.

Codage de l'entrée

Attention, soulignons que la complexité d'un problème est définie par rapport à un codage naturel de ses instances.

- Il existe un algorithme connu pour répondre à la question de PRIME (test de primalité d'un nombre) en temps $O(\sqrt{N})$ (le crible d'Erathostène).
- Mais un codage *naturel* de PRIME consiste à représenter N en binaire.
  Le mot d'entrée étant ainsi codé sur $\log_2(N)$ bits, la complexité de l'algorithme est en fait exponentielle !

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

La notion de complexité a été introduite pour qualifier la nature plus ou moins ardue des problèmes qui admettent une solution sous forme de *procédure effective* (i.e. calculable sur un ordinateur universel).

La classe de complexité $\mathcal{P}$ correspond aux problèmes pouvant être résolus en temps polynomial.
Malheureusement une très large proportion de problèmes "intéressants" ont peu d'espoir d'admettre un algorithme de résolution polynomiale.
La plupart de ces problèmes appartiennent à une classe de complexité plus vaste, $\mathcal{NP}$.

Différences

- Pour un problème de $\mathcal{P}$, il est possible de *trouver* en temps polynomial la réponse à **toutes** les instances
- Pour un problème de $\mathcal{NP}$ il est possible de *vérifier* en temps polynomial qu'une réponse est correcte.

Caractérisation de $\mathcal{NP}$

La classe $\mathcal{NP}$ est définie par rapport aux machines de Turing non déterministes (NTM). Rappelons qu'une NTM accepte un mot simplement si il existe une exécution acceptant ce mot. Ainsi, une NTM résout le problème en temps $f(n)$ si :

- Pour tout mot w d'entrée, il existe une exécution acceptant w en au plus $f(|w|)$ transitions.
- Pour tout mot w qui n'appartient pas aux entrées, aucune exécution (quelle que soit sa longueur) ne conduit à un état accepteur de la machine de Turing.

Caractérisation pratique

$\mathcal{NP}$ il est l'ensemble des problèmes dont est possible de *vérifier* en temps polynomial qu'une réponse est correcte.

Par exemple HamiltonianCircuit (HC) qui décide de l'existence d'un cycle Hamiltonien dans un graphe :

- **HC**
- **Instance:** Un graphe $G = (V, E)$
- **question** Existe-t-il un cycle Hamiltonien ? (c'est-à-dire un cycle passant une fois et une seule par chaque sommet).

HC appartient à $\mathcal{NP}$.
Considérons comme codage naturel la matrice d'adjacence du graphe. Un certificat de positivité consiste à donner une permutation des sommets.

Ce certificat est de taille $O(n \log n)$ qui correspond au codage des nombres successifs de cette permutation.
On peut vérifier en temps $O(n^2)$ que la permutation correspond à un cycle du graphe.
Ce certificat est bien polynomial en la taille de l'instance $\Theta(n^2)$.

Réduction

Nous aimerions introduire une relation d'ordre sur les problèmes, signifiant qu'un problème est plus facile qu'un autre.
Une telle relation d'ordre est définie par une *réduction polynomiale*.

La réduction τ transforme toutes les instances positives Π en instances positives de Π', et toutes instances négatives de Π en instances négatives de Π'.
L'existence d'une réduction polynomiale de Π vers Π' montre que $\Pi' = \tau(\Pi)$ est au moins aussi difficile que Π.

On notera $\Pi \alpha \Pi'$ si il existe une réduction polynomiale de Π vers Π'. Nous dirons que Π se réduit à Π'.

Exemples

Pour montrer que nous avons la réduction HP α HC, considérons une instance de HP, c'est-à-dire un graphe G.

A partir de G nous construisons une instance particulière $\tau(G)$ de HC en ajoutant G un nouveau sommet x relié à tous les autres : $\tau(G) = (V \cup \{x\}, E \cup (x, y) \forall y \in V)$.

- **HP**
- **Instance:** Un graphe $G = (V, E)$
- **question** Existe-t-il un chaine Hamiltonien ?
  (c'est-à-dire une chaine passant une fois et une seule par chaque sommet).

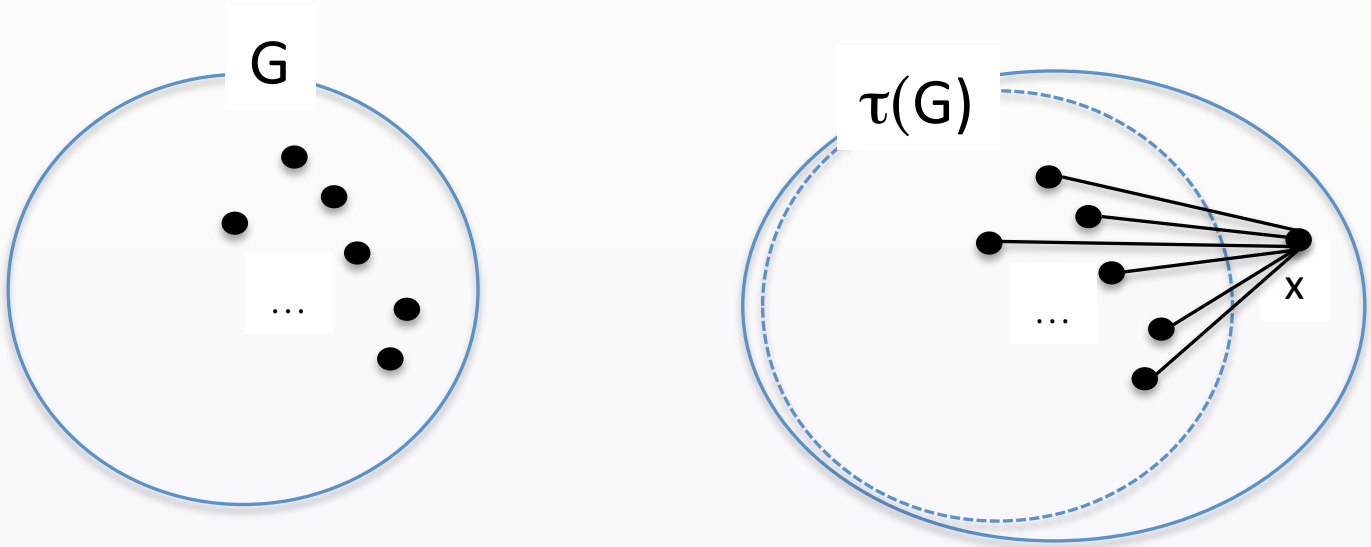


Figure: Réduction de *HP* vers *HC*.

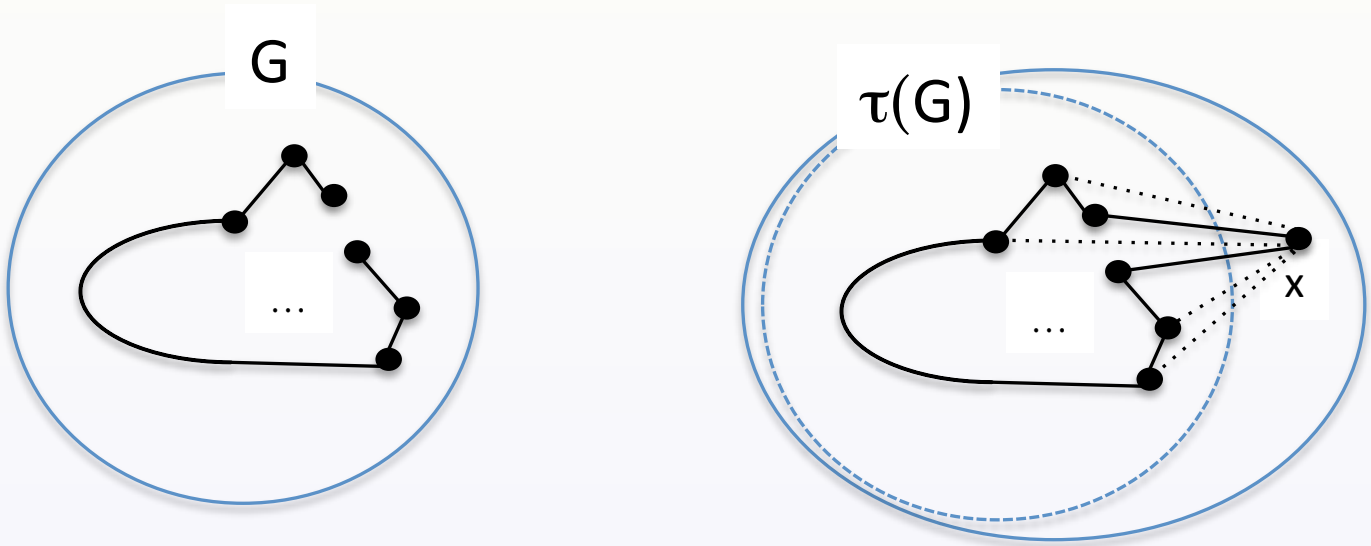


Figure: Un chemin hamiltonien de G donne un cycle de $\tau(G)$.

Preuve

La transformation τ est évidemment polynomiale.
Montrons que c'est une réduction, c'est-à-dire que G admet une chaine hamiltonienne si et seulement si $\tau(G)$ possède un cycle hamiltonien.

- Si G possède une chaine hamiltonienne φ, alors le cycle $x\varphi x$ est hamiltonien dans $\tau(G)$.
- Réciproquement, si $\tau(G)$ admet un cycle hamiltonien, son sous-graphe privé de x, G, possède une chaine hamiltonienne.

On peut également établir que nous avons la réduction du cycle hamiltonien au circuit hamiltonien, $HC \alpha HP$.

Ce résultat n'est pas aussi immédiat, même s'il semble facile de déduire une chaine hamiltonienne à partir d'un cycle hamiltonien, cela ne suffit pas à définir une réduction.

Considérons la transformation τ suivante d'une instance G de HC vers une instance $\tau(G)$ de HP :

Figure: Réduction de *HC* vers *HP*.

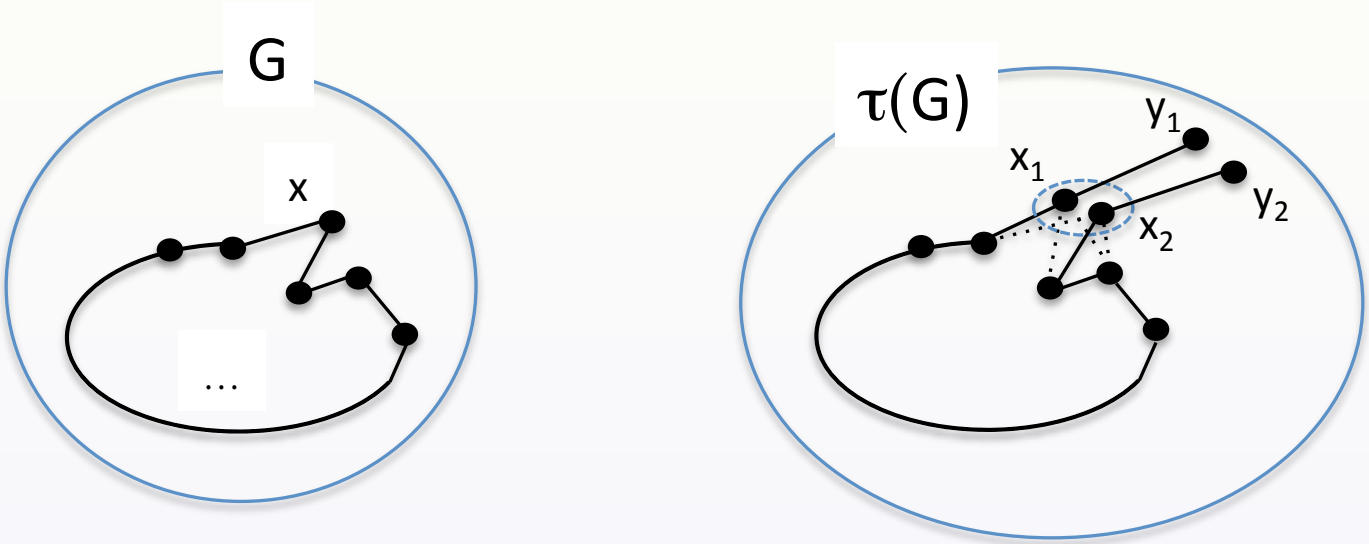


Figure: Un cycle hamiltonien de G donne un chemin de $\tau(G)$.

On duplique un sommet quelconque x du graphe G en (x_1, x_2), puis on relie x_1 et x_2 respectivement à deux nouveaux sommets y_1 et y_2 comme c'est indiqué dans la figure.

Cette transformation est bien polynomiale.

C'est une réduction : G admet un cycle hamiltonien si et seulement si $\tau(G)$ possède une chaine hamiltonienne.

- En effet, si G possède un cycle hamiltonien, la chaine formée de l'arrête y_1 à x_1, de la partie du cycle jusqu'à un voisin de x_1, puis des arrêtes de ce voisin à x_2 et celle de x_2 à y_2 est une chaine hamiltonienne de $\tau(G)$.
- Réciproquement, s'il existe une chaine hamiltonienne φ dans $\tau(G)$, elle est nécessairement de la forme $y_1 x_1 \psi x_2 y_2$.
  Alors $x\psi x$ est un cycle hamiltonien sur G.

Un autre exemple

Considérons la version de décision du problème du voyageur de commerce (D-TSP).

- D-TSP
- **Instance.** un ensemble V de villes, la matrice des distances inter-villes $(d_{i,j})$ et une constante k.
- **Question.** Déterminer s'il existe un parcours fermé, passant une fois et une seule par toutes les villes, de longueur inférieure à k.

Principe

La réduction est la suivante :

L'ensemble des villes correspond aux sommets du graphe G, les distances sont données par $d_{i,j} = 1$ si i et j sont reliés, 2 sinon. La constante k est égale à n (nombre de villes).

Cette transformation est polynomiale de manière évidente.

- Si G possède un cycle hamiltonien alors ce cycle correspond à une tourne (qui passe par toutes villes une fois et une seule) et sa longueur est n, donc $\tau(G)$ est positive.
- Réciproquement, une instance positive de D-TSP dans $\tau(G)$ est transformée en une instance positive de HC.
  En effet, la solution de D-TSP possède par définition des arêtes de coût unitaire puisque la longueur totale est n, ce qui correspond bien à un cycle hamiltonien de G.

Problèmes NP-complets

Nous venons de voir qu'il existe des problèmes dans $\mathcal{P}$ et $\mathcal{NP}$, dans un soucis de classification, muni de l'outil des rductions polynomiale, il est naturel de s'intéresser à la question de la structuration de $\mathcal{NP}$.
En particulier, il existe un plus grand élément au sens de ces réductions, c'est-à-dire une classe qui contient les problèmes plus difficiles que tous les autres (c'est la classe NP-complet).

Cook a exhibé un tel problème en 1971 avec SAT (satisfability) en montrant que la résolution de tout problème pouvait se coder sous forme d'expression logique en forme normale conjonctive. Résoudre le problème revient ainsi à résoudre SAT.

SAT est un problème de logique propositionnelle[1]. Rappelons qu'un prédicat est défini sur un ensemble de variables logiques, à l'aide des 3 opérations élémentaires de négation NON ($\neg x$), la conjonction ET ($x \wedge y$) et la disjonction OU ($x \vee y$).

Un littéral est un prédicat formé d'une seule variable (x) ou de sa négation $\bar{x}$.
Une clause est un prédicat particulier, formée uniquement de disjonctions de littéraux, par exemple $C = x \vee \bar{y} \vee z$.
Une formule est sous forme normale conjonctive si elle s'écrit comme la conjonction de clauses.
Le problème SAT consiste à décider si une formule en forme normale conjonctive est satisfiable, c'est-à-dire si il existe une assignation τ aux variables telles que toutes les clauses sont *TRUE*.

[1] appelée logique d'ordre 0 ou logique des prédicats

- SAT (Satisfiability)
- **Instance.** m clauses C_i formées à l'aide de n variables logiques (littéraux)
- **Question.** la formule $\Phi = C_1 \wedge C_2 \wedge ... \wedge C_m$ est-elle satisfiable ?[2]

[2] existe-t-il une fonction d'interprétation qui rende la formule vraie ?

Exemple

la formule $(x \vee \bar{y} \vee z) \wedge (\bar{x} \vee \bar{y}) \wedge (x \vee \bar{z})$ est satisfaite avec l'assignation $\gamma(x) = \mathit{TRUE}$ ou 1 et $\gamma(y) = \mathit{FALSE}$.

- Il est facile de se convaincre que le problème $\textsc{Sat}$ est dans $\mathcal{NP}$.
  Un certificat de positivité consiste à donner une assignation des variables, codable sur un vecteur de n bits.
- La vérification que toutes les clauses sont satisfaites est alors clairement polynomiale (plus précisément, dans $\Theta(nm)$).

variante : kSAT

Ici, toutes les clauses ont exactement k littéraux ($k \geq 2$).
La variante obtenue pour $k = 3$ est particulièrement utile, on peut établir le résultat simple suivant.

Proposition. 3SAT est NP-complet.

Tout d'abord, on vérifie facilement que 3SAT $\in \mathcal{NP}$ (c'est un cas particulier de SAT qui est dans $\mathcal{NP}$).

Preuve.

Lidée de la réduction à partir de SAT est simplement de réécrire chaque clause comme une clause de cardinalité 3.

- Pour les clauses de deux littéraux, on rajoute une variable et on introduit deux nouvelles clauses qui ne changent pas lévaluation de la formule.
  Par exemple pour le cas dune clause de cardinalité 2, $(x \vee y)$, on rajoute la variable z, et on réécrit la clause comme $(x \vee y \vee z) \wedge (x \vee y \vee \bar{z})$.

  Pour les clauses d'une seule variable, on rajoutera de même deux variables...

Preuve.

- Pour les clauses de cardinalité p strictement supérieure 3, on procède avec la même idée de rajouter des variables qui conservent la valeur logique à l'expression.
  Ainsi, on rajoute $p-3$ variables et autant de clauses.
  Par exemple pour une clause formée de 5 littéraux $(x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5)$,
  on introduit 2 nouvelles variables y_1 et y_2 :

  $$(x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5) = (x_1 \vee x_2 \vee y_1) \wedge (\bar{y_1} \vee x_3 \vee y_2) \wedge (\bar{y_2} \vee x_4 \vee x_5)$$

  La réduction proprement dite est alors simple à vérifier.

Prouver la NP-complétude d'un problème

Comment prouve-t-on qu'un problème est dans la classe NP-complet ?

- Montrer qu'il est dans $\mathcal{NP}$
- Choisir un problème NP-complet (il en existe aujourd'hui un très grand nombre) et construire une réduction polynomiale adéquate

Quelques problèmes

Les problèmes NP-complets que nous allons utilisés dans ce cours sont :

- 2-Partition
- Knapsack
- 3SAT
- SubSetSum
- Ordonnancement sur 2 machines
- HC et HP
- TSP

Un autre exemple détaillé (Vertex Cover)

La couverture des sommets d'un graphe par ses arêtes.

- **VC**
- **Instance.** Un graphe $G = (V, E)$ donné par exemple par sa matrice d'adjacence et un entier Q
- **Question.** Existe-t-il un ensemble V' d'au plus Q sommets qui couvrent G ?
  C'est-à-dire, tel que chaque arête de G a au moins une de ces extrémités dans V'.

Réduction

On transforme une instance (positive) de 3SAT en instance de VC.

$C_1 \wedge C_2 \wedge \cdots \wedge C_m$

$C_i = x_{i,1} \vee x_{i,2} \vee x_{i,3}$
où $x_{i,j}$ est un littéral sur les variables propositionnelles
$\{p_1, p_2, \cdots, p_n\}$

Construction du graphe associé (les sommets)

- Une paire de sommets pour chaque variable propositionnelle étiquetés p_i et $\neg p_i$
- Un triplet de sommets pour chaque clause C_i

Le nombre de sommets est donc égal à $2n + 3m$

Construction du graphe associé (les arêtes)

- Une arête entre chaque p_i et $\neg p_i$
- Une arête entre chacun des trois sommets des triangles C_i
- Une arête entre chaque $x_{i,j}$ et le sommet p ou $\neg p$ représentant le littéral correspondant

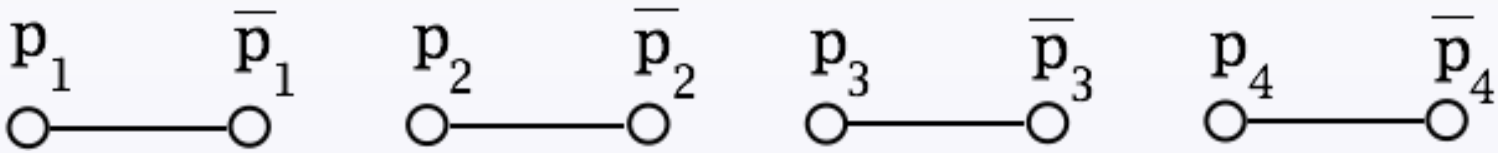
Le nombre d'arêtes est donc égal à $3m + 3m + n$

Construction du graphe associé (la constante Q)

Q est égal à $2m + n$

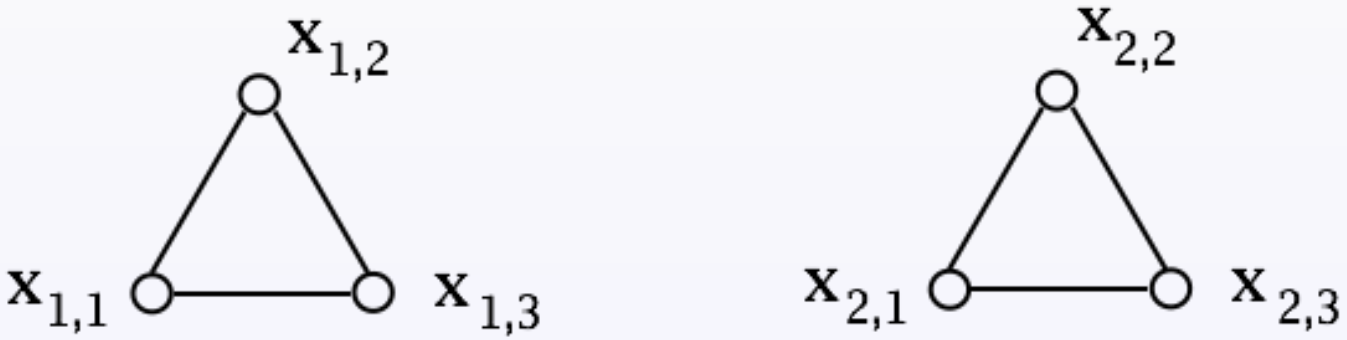
Exemple (4 variables propositionnelles)

$$(p_2 \vee \neg p_1 \vee p_4) \wedge (\neg p_3 \vee \neg p_2 \vee \neg p_4)$$



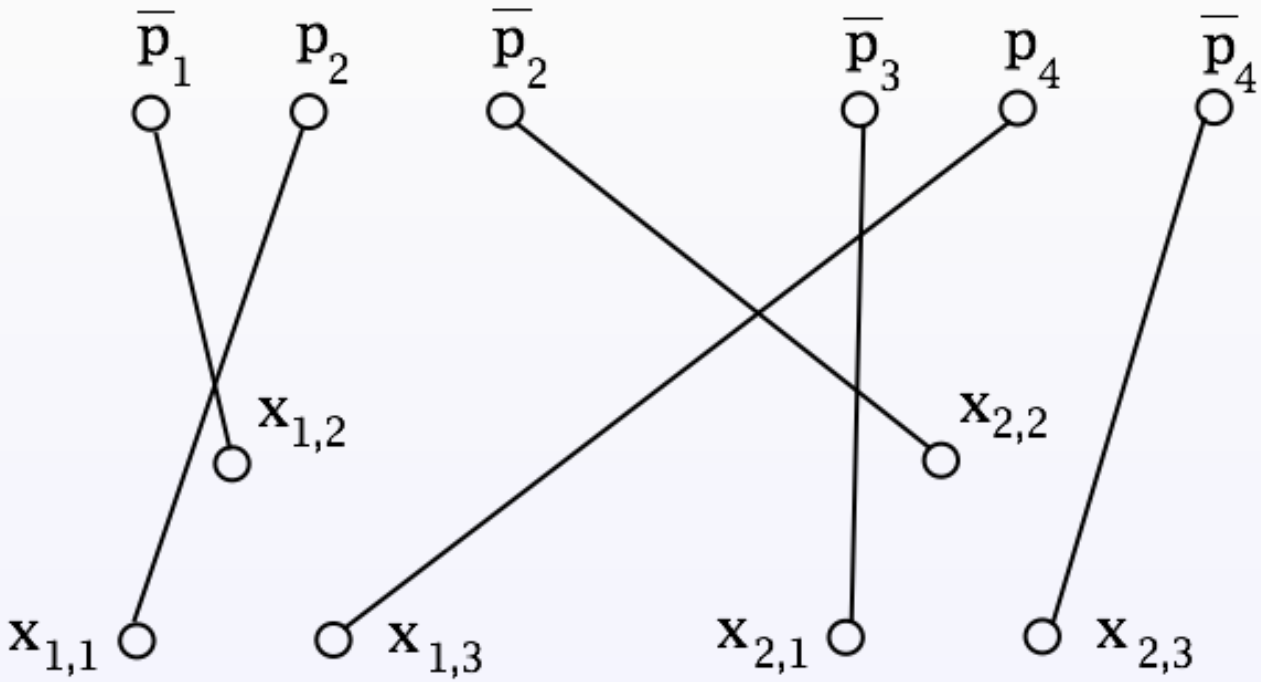
Exemple (2 triangles)

$$(p_2 \vee \neg p_1 \vee p_4) \wedge (\neg p_3 \vee \neg p_2 \vee \neg p_4)$$



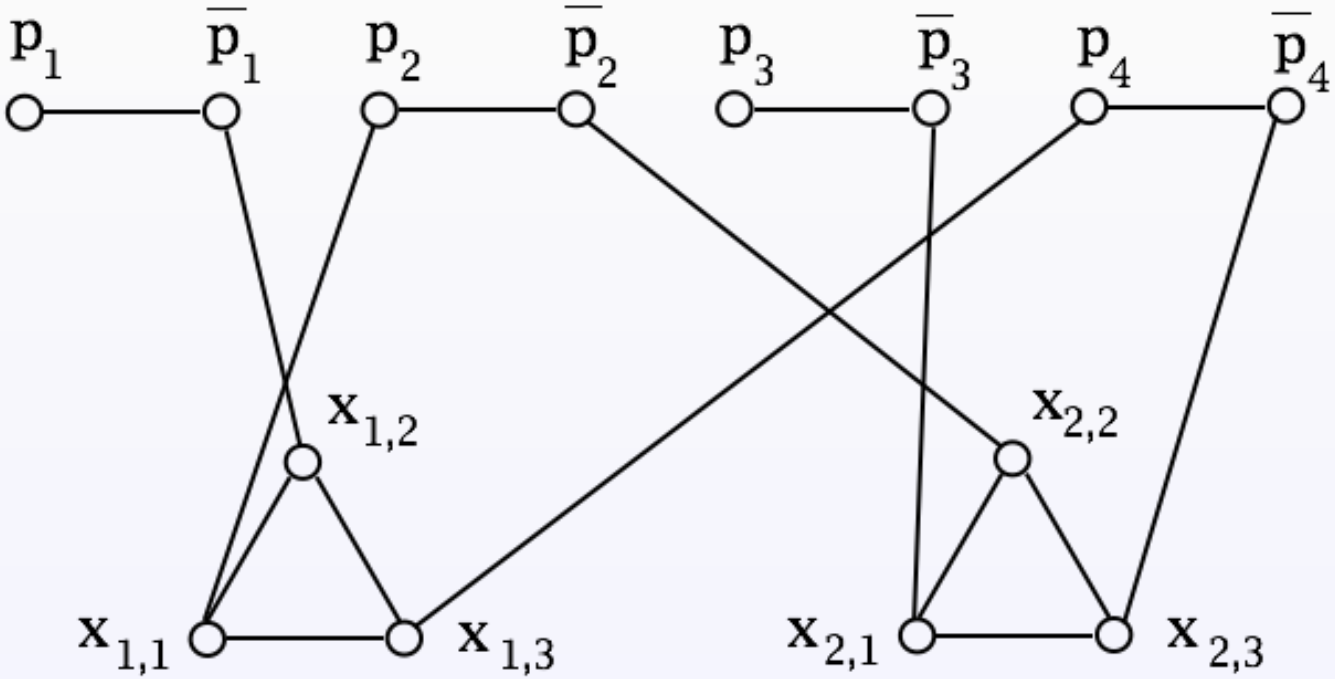
Exemple (Les associations x vers p)

$$(p_2 \vee \neg p_1 \vee p_4) \wedge (\neg p_3 \vee \neg p_2 \vee \neg p_4)$$



Le graphe final

$(p_2 \vee \neg p_1 \vee p_4) \wedge (\neg p_3 \vee \neg p_2 \vee \neg p_4)$



Preuve : ce qu'il faut démontrer

La transformation décrite est polynomiale.

Il faut montrer maintenant que l'instance de 3SAT est satisfiable si et seulement si le graphe ainsi généré a une couverture au plus $2m + n$.

Preuve

Si l'instance de 3SAT est satisfiable, alors il existe une fonction d'interprétation γ qui rend chaque clause C_i vraie.
La couverture comprend les sommets tels que

- γ affecte la valeur 1 à p_i et 0 à $\neg p_i$.
- 2 sommets pour chaque triangle, choisis tels que γ affecte 1 au littéral correspondant au sommet non choisi[3].

La taille de cette couverture est exactement $2m + n$.

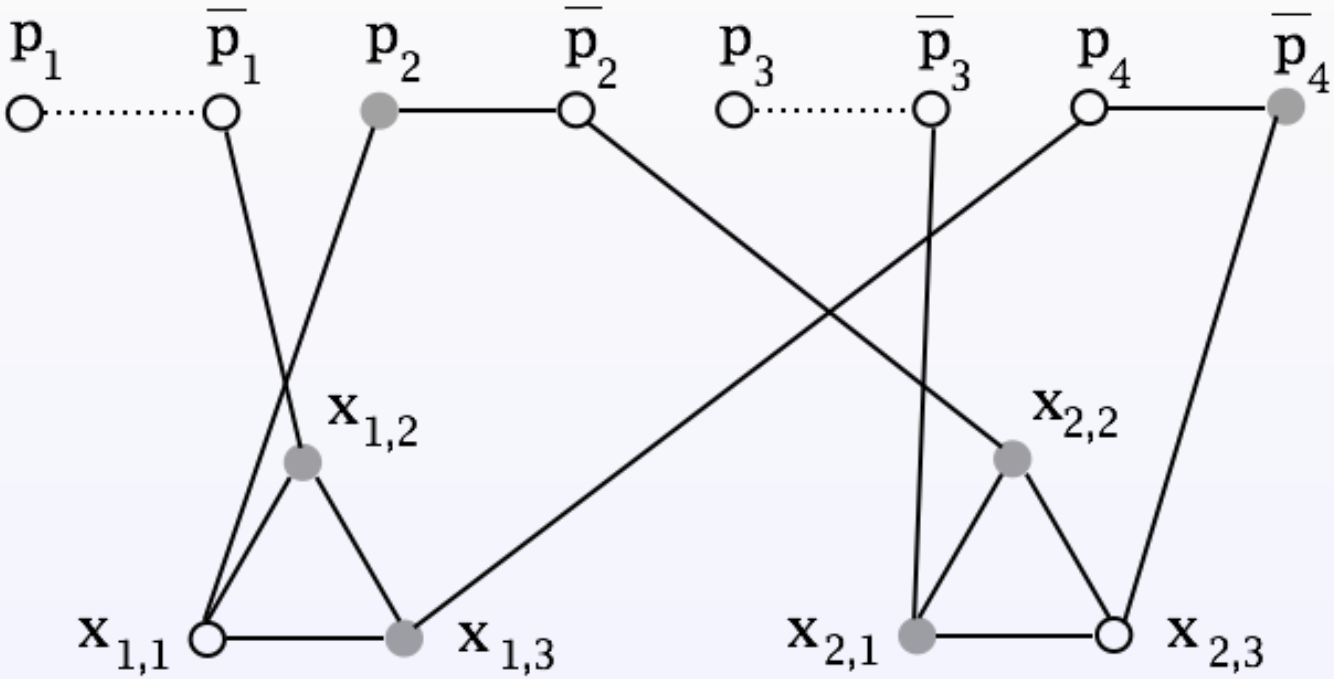
Il faut vérifier que toutes les arêtes sont bien convertes.

[3] ce qui est toujours possible

Une solution de VC

$(p_2 \vee \neg p_1 \vee p_4) \wedge (\neg p_3 \vee \neg p_2 \vee \neg p_4)$

$(p_2 = 1, \neg p_4 = 1)$ est une solution (quels que soient p_1 et p_3).



Réciproque

Si le graphe a une couverture de taille $2m + n$, alors celle-ci définit une fonction d'interprétation qui rend la formule vraie.
En effet :

- Une couverture de taille $2m + n$ a forcément un sommet dans chaque paire $(p_i, \neg p_i)$ et deux sommets pour chaque triangle. Elle ne peut en avoir plus sinon, la couverture serait plus que $2m + n$.
- Considérons la fonction d'interprétation qui affecte 1 à p_i si il est dans la couverture et 0 si $\neg p_i$ est dans la couverture. L'existence de cette couverture implique que la formule de 3SAT est satisfaite car la fonction d'interprétation affecte la valeur 1 à au moins un littéral de chaque clause.