Etude des preuves parallèles des SAT solveurs sous forme de grands graphes

Rohan Fossé Laurent Simon
rohan.fosse@labri.fr lsimon@labri.fr

LaBRI - Université de Bordeaux - UMR 5800

Le problème SAT

Définition

Soit $\Phi(a, b, ..)$ une formule logique. Existe-t-il une **affectation** des variables $(a, b, ..)$ rendant Φ vraie ?

En théorie

Théorème de **Cook-Levin** : SAT est **NP-Complet**.

$\Rightarrow$ Au cœur de la Hiérarchie Polynomiale

$\Rightarrow$ Au centre de la **la théorie de la complexité** (problème P = NP ?)

En pratique

- Nombreuses applications critiques
- Problèmes de grande taille

Notations

Littéraux

Un littéral $(a, b, \ldots)$ représente la variable booléenne x ou son complément $\neg x$

Clauses

Une clause est une disjonction de littéraux *i.e* :

$C = a \vee b \vee .. \vee z$

Formule

Une formule est une conjonction de clauses *i.e* :

$\Phi = C_1 \wedge C_2 \wedge ... \wedge C_m$

Exemple

$\Phi = (a \vee \neg b) \wedge b \wedge (\neg a \vee \neg b)$

La règle de résolution

Règle de résolution (Robinson 65)

Soit C_1 et C_2 deux clauses t.q :

$$C_1 = a \vee b \vee c \vee \mathrm{d}$$
$$C_2 = \neg d \vee e \vee f$$

La règle de résolution sur d donne :

$$\overbrace{(a \vee b \vee c \vee d)}^{C_1} \wedge \overbrace{(\neg d \vee e \vee f)}^{C_2}$$
$$\vdash a \vee b \vee c \vee e \vee f$$

Règle de résolution (Robinson 65)

Formellement,

Soit C_1 et C_2 deux clauses, la règle de résolution nous donne :

$$(C_1 \vee x) \wedge (C_2 \vee \neg x) \vdash C_1 \vee C_2$$

On appelle $C_1 \vee C_2$ le résolvant de $C_1 \vee x$ et $C_2 \vee \neg x$.

Règle de résolution (Robinson 65)

Formellement,

Soit C_1 et C_2 deux clauses, la règle de résolution nous donne :

$$(C_1 \vee x) \wedge (C_2 \vee \neg x) \vdash C_1 \vee C_2$$

On appelle $C_1 \vee C_2$ le résolvant de $C_1 \vee x$ et $C_2 \vee \neg x$.

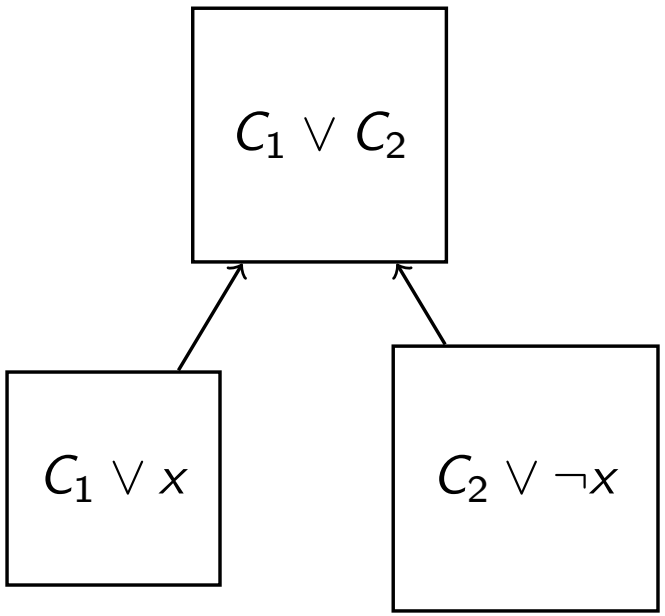


Figure 1 – Représentation graphique de la résolution

Inconsistance

Correction et complétude de la résolution

Une formule F inconsistante : Clause vide ($\square$) peut être dérivée en partant des clauses de F.

Derivation : séquence de résolutions

Une preuve d'inconsistance représente la séquence de résolutions.

Dans cet exposé

Nous parlerons seulement des preuves d'inconsistance (les preuves de satisfiabilité correspondant à un **assignement** des variables).

Inconsistance

Correction et complétude de la résolution

Une formule Φ inconsistante : Clause vide ($\square$) peut être dérivée en partant des clauses de F.

Derivation : séquence de résolutions

Une preuve d'inconsistance représente la séquence de résolutions.

Formellement

La séquence de résolutions Γ_r de F est une séquence de clauses $C_1, .., C_m$ tel que :

- $C_m = \square$
- $\forall i \in [1, m-1], C_i \in F$

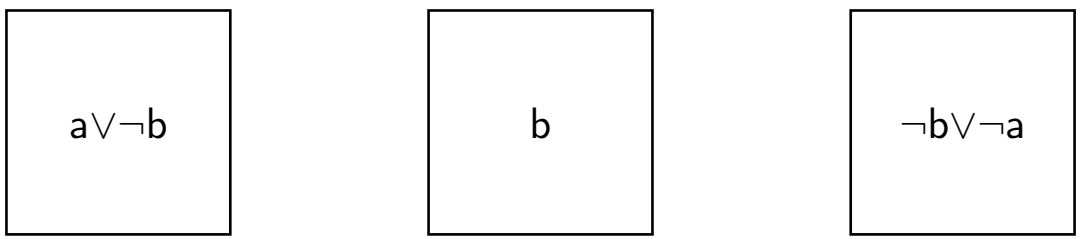
  ou C_i est le résolvant de deux clauses $C_j, C_k, j, k \in [1, i-1]$

Exemple simple

Formule

$$\Phi = (a \vee \neg b) \wedge b \wedge (\neg b \vee \neg a)$$

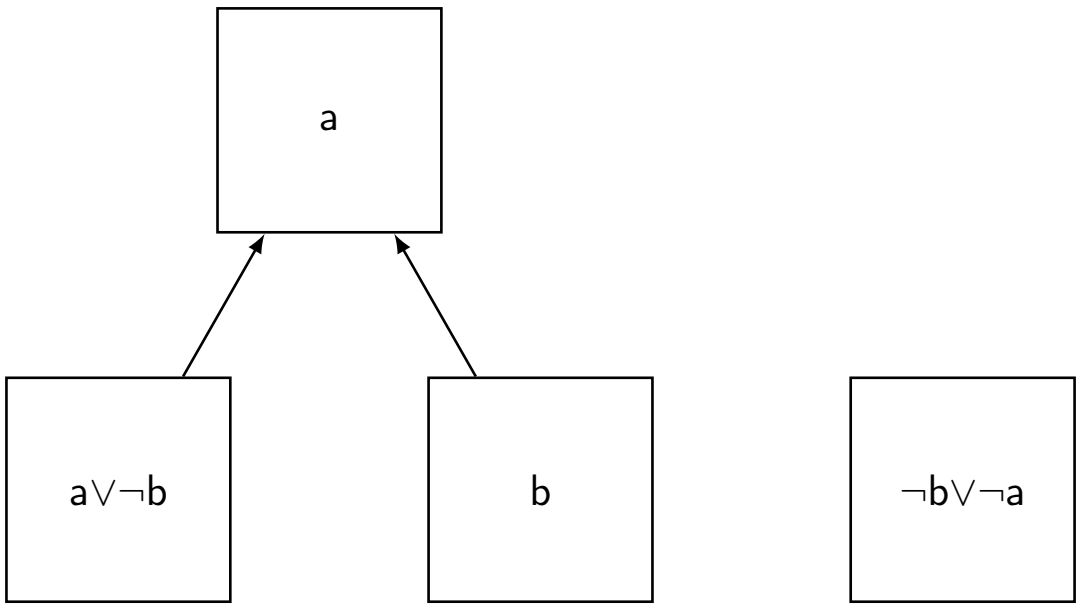
On peut représenter cette formule de la façon suivante :



Exemple simple

Formule

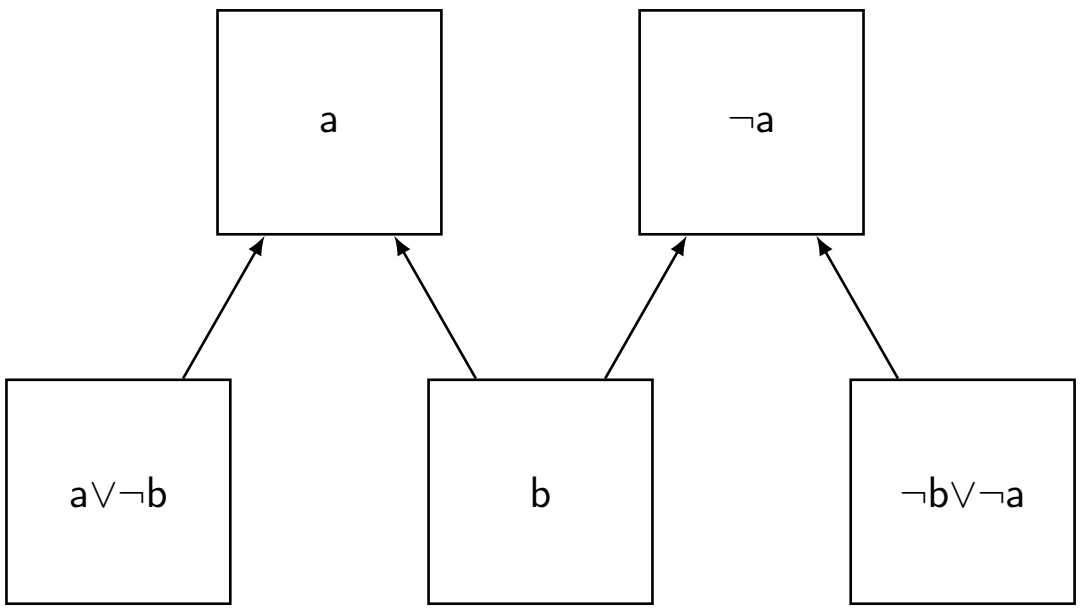
$$\Phi = (a \vee \neg b) \wedge b \wedge (\neg b \vee \neg a)$$



Exemple simple

Formule

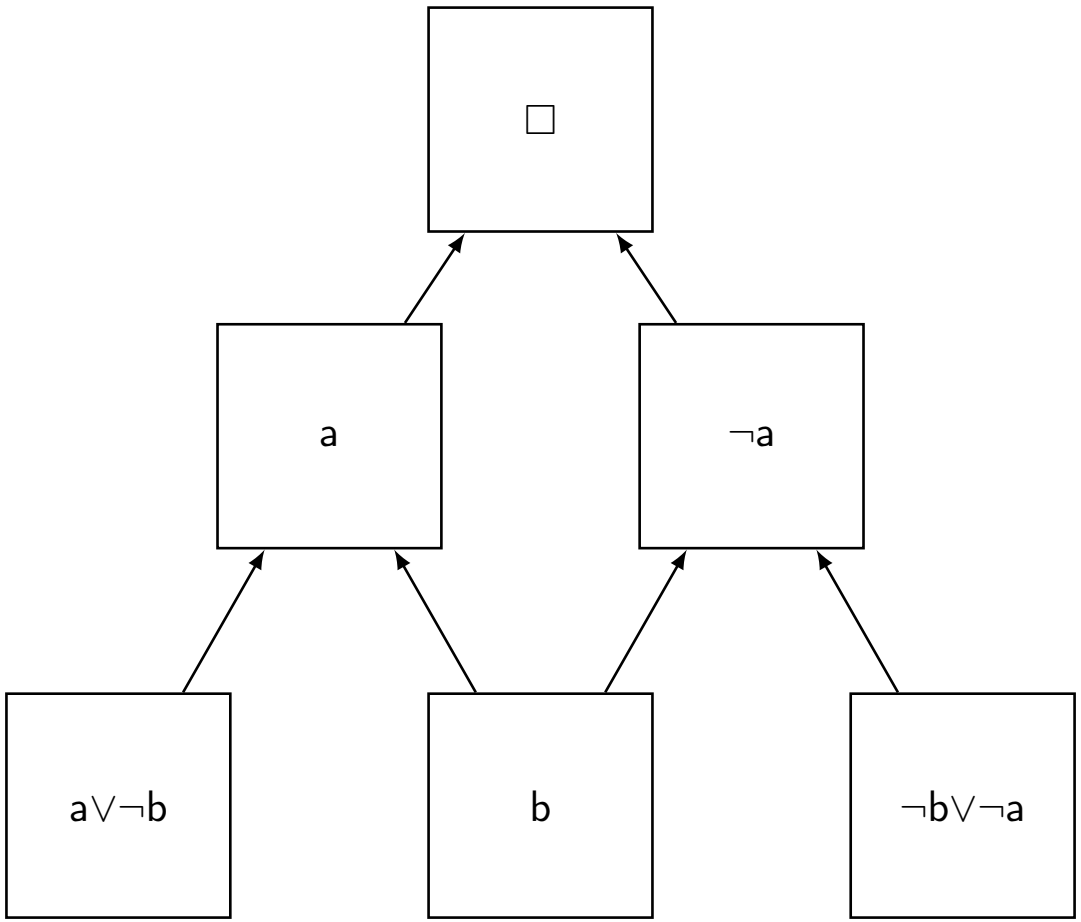
$$\Phi = (a \vee \neg b) \wedge b \wedge (\neg b \vee \neg a)$$



Exemple simple

Formule

$$\Phi = (a \vee \neg b) \wedge b \wedge (\neg b \vee \neg a)$$



Les solveurs SAT modernes

Φ : l'ensemble des clauses initiales

Σ : l'ensemble des clauses apprises.

Algorithm 1 solveurs SAT modernes

Tant que $\square \notin \Phi \cup \Sigma$ **faire**
  $C \leftarrow apprendreClause()$
  $\Sigma = \Sigma \cup C$
  Si $tropPlein(\Sigma)$ **Alors**
    $\Delta = clausesAEffacer(\Sigma)$
    $\Sigma = \Sigma \backslash \Delta$
  Fin Si
Fin Tant que

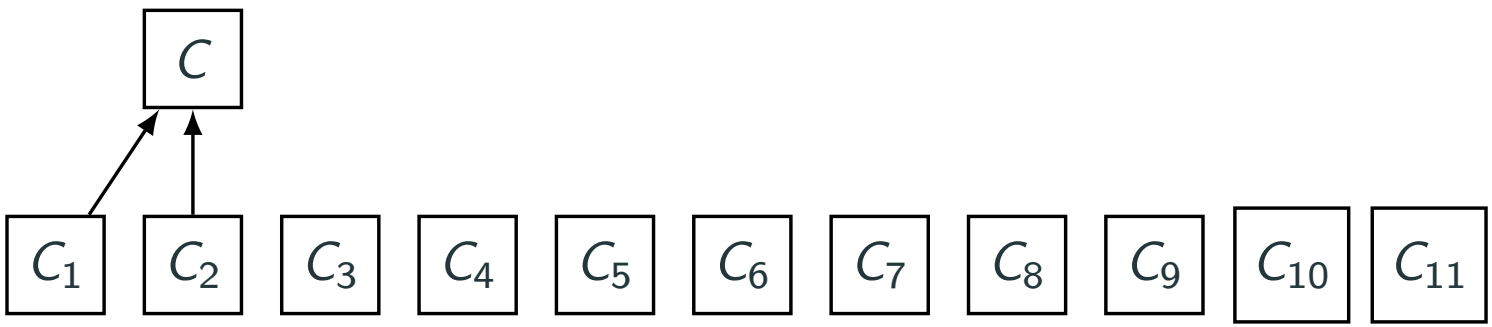
Séquence de résolutions sous forme de graphe

Formule

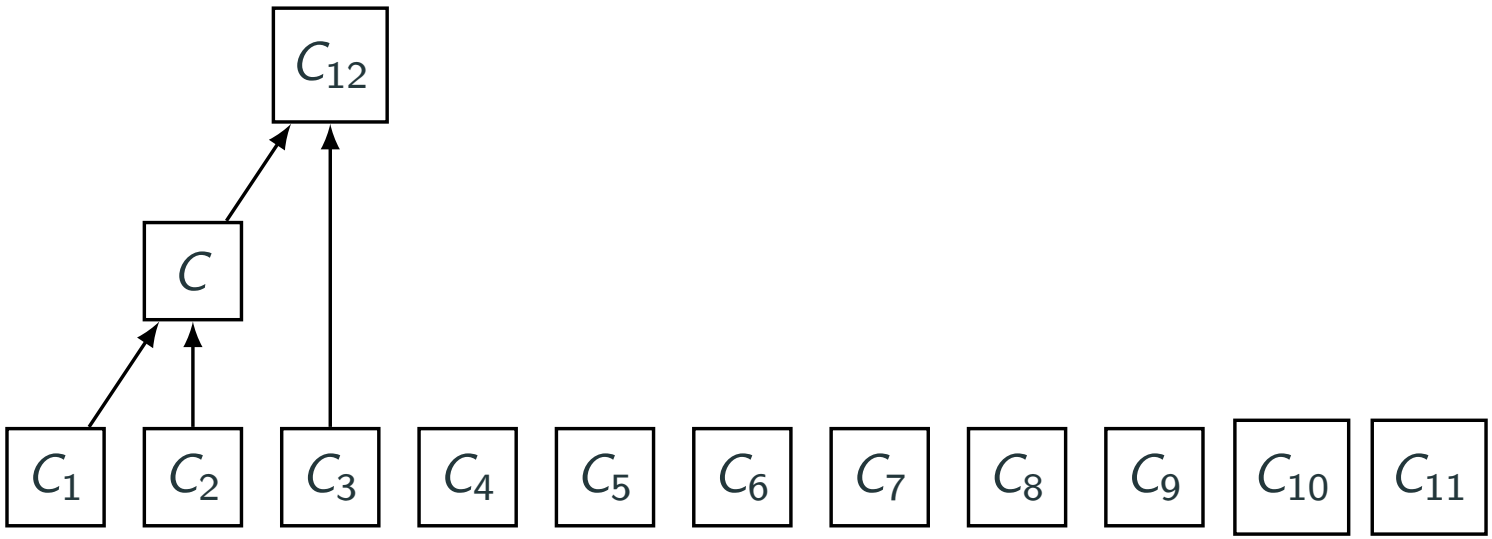
Soit $\Phi = C_1 \wedge C_2 \wedge .. \wedge C_{11}$, tq. $\forall i \in [1..11], C_i$ une clause quelconque.

On peut représenter cette formule de la façon suivante :

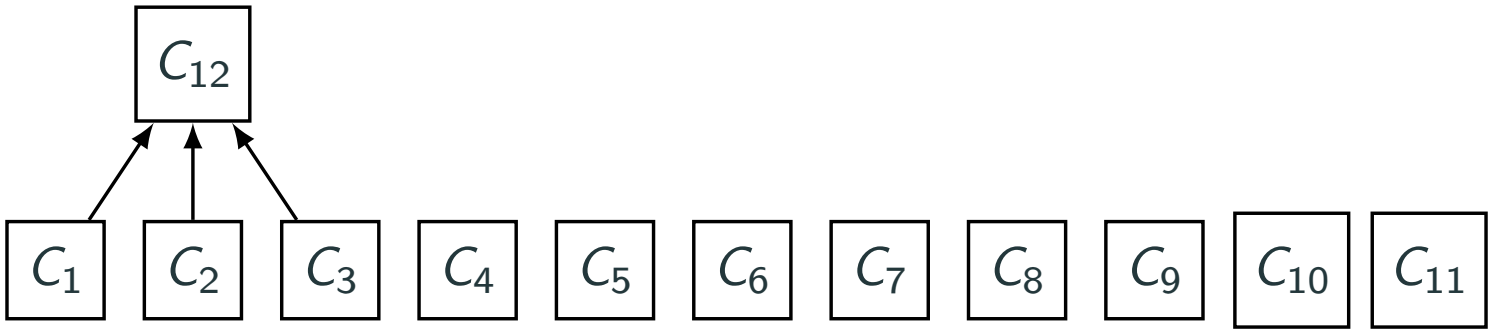
Séquence de résolutions sous forme de graphe



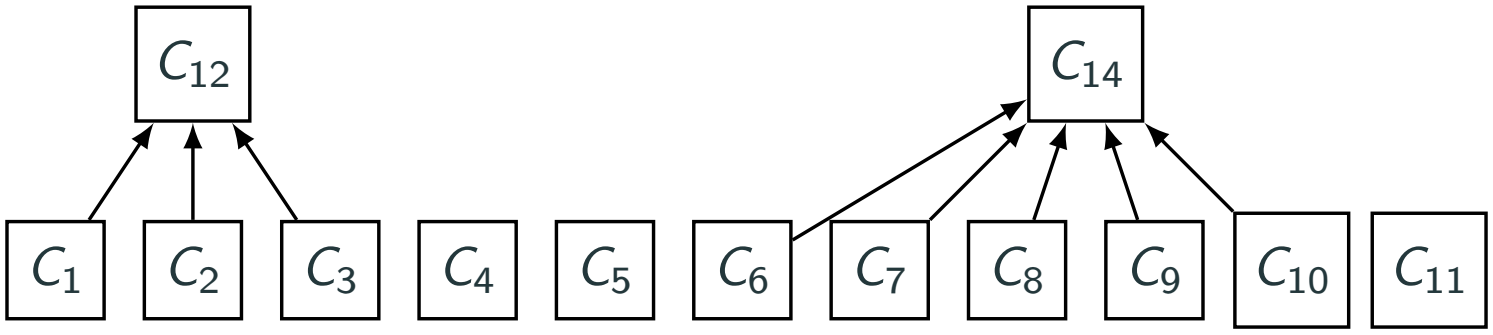
Séquence de résolutions sous forme de graphe



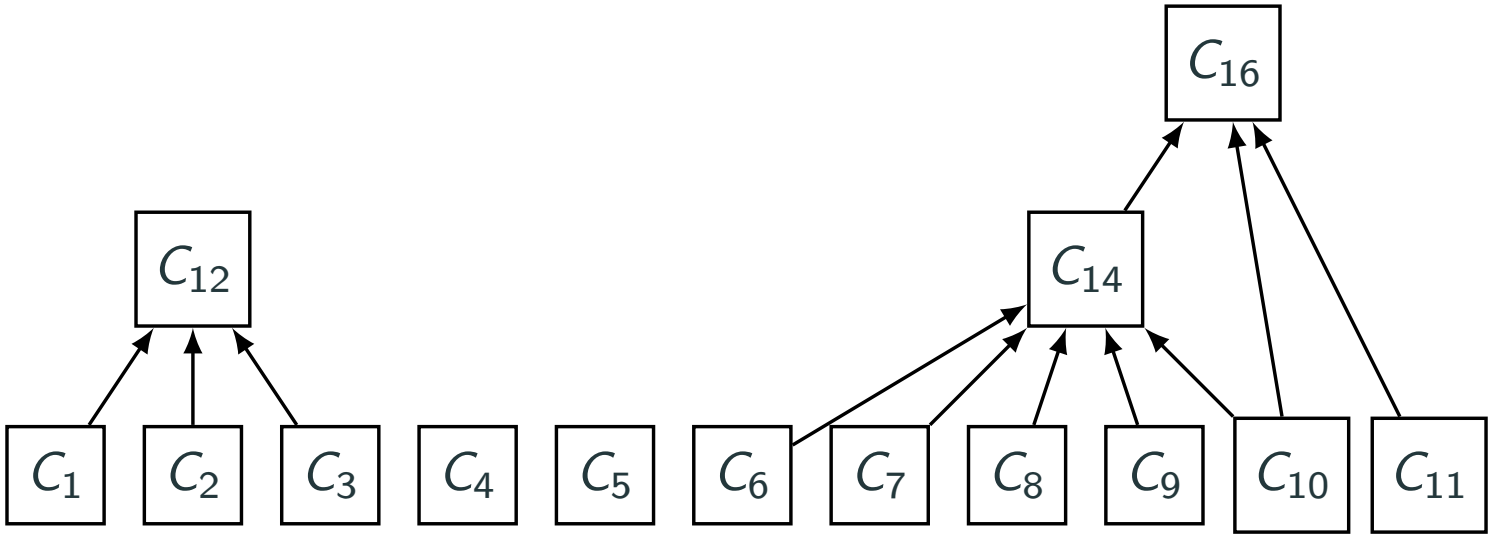
Séquence de résolutions sous forme de graphe



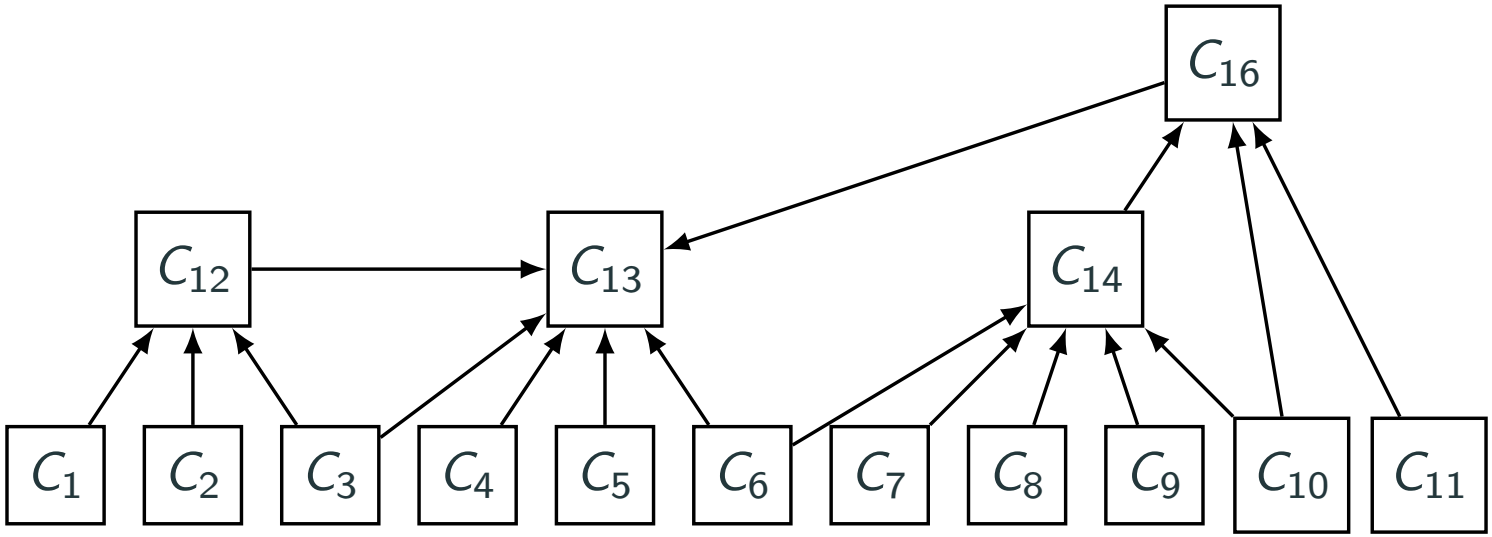
Séquence de résolutions sous forme de graphe



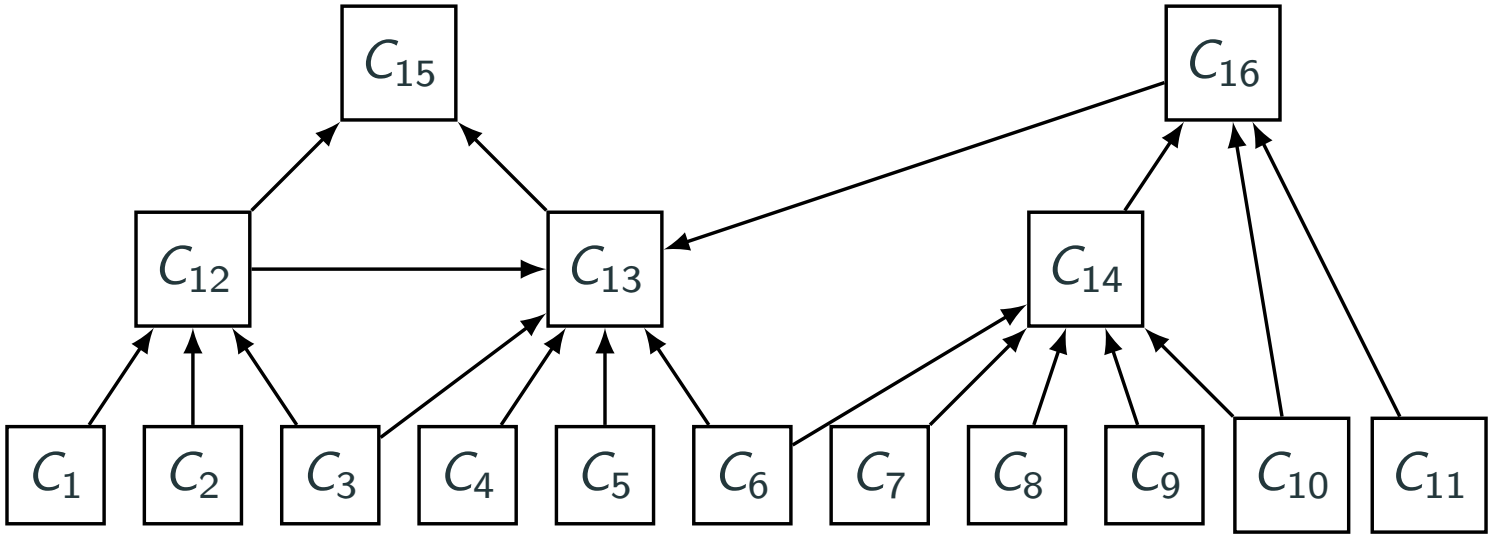
Séquence de résolutions sous forme de graphe



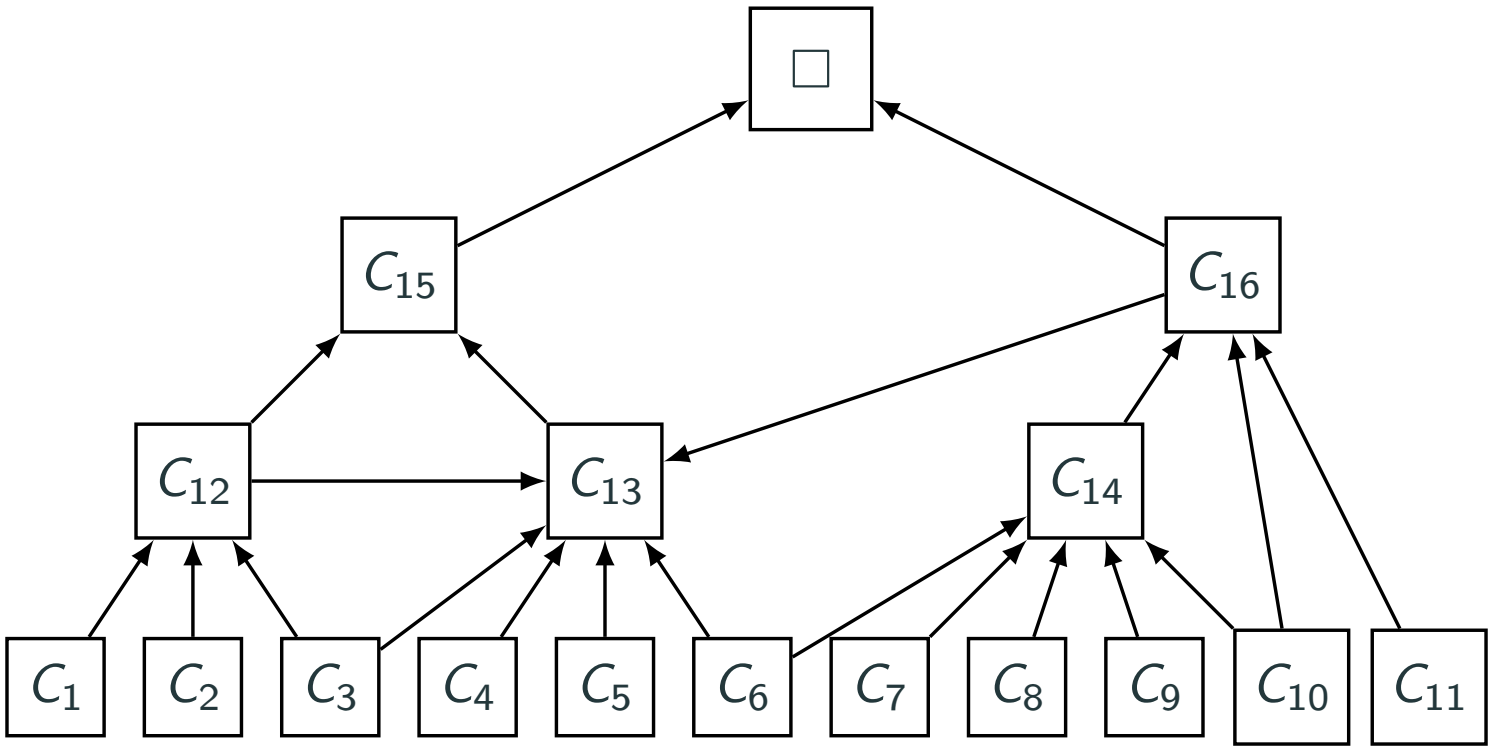
Séquence de résolutions sous forme de graphe



Séquence de résolutions sous forme de graphe



Séquence de résolutions sous forme de graphe

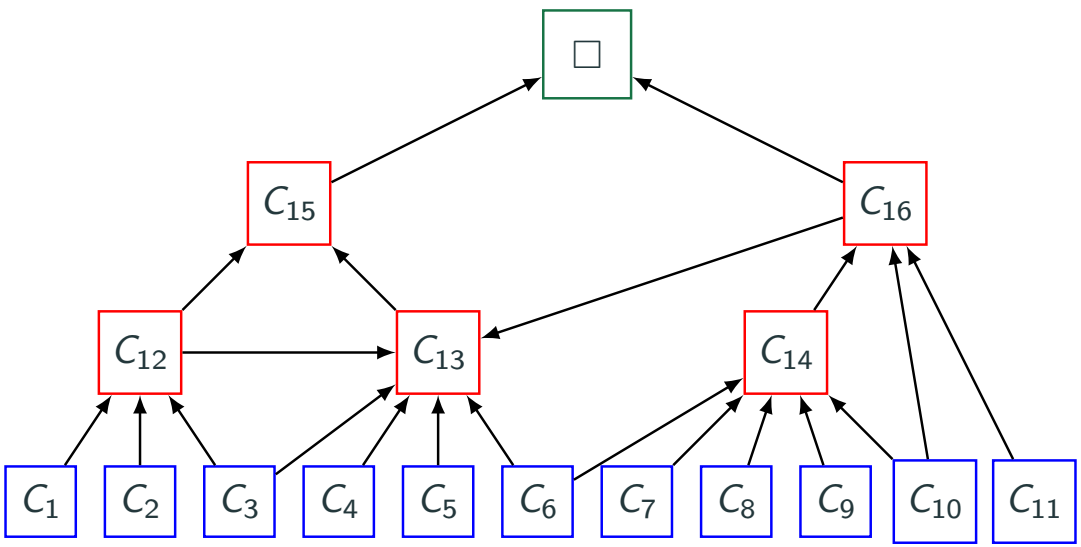


Graphe de résolution

Définition

Le graphe de résolution est un graphe orienté acyclique (ou **DAG**) tel que :

- Les feuilles sont les clauses initiales
- les noeuds internes sont les clauses apprises
- La racine est la clause **vide**



Preuve en séquentiel

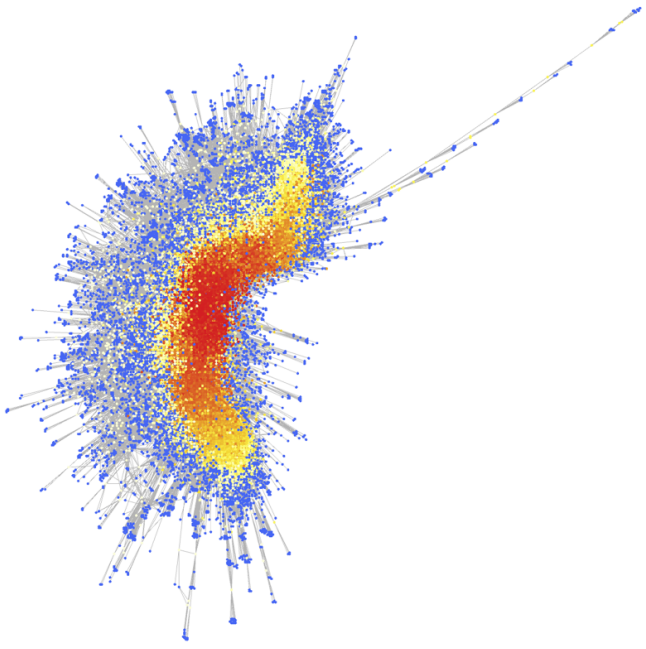


Figure 2 – Preuve en séquentiel

Informations

Formule

clauses : 55585

variables : 50076

Conflits

conflits : 59792

Temps calcul : 6s

Graphe

sommets : 51274

arêtes : 960620

Les solveurs SAT modernes parallèles

Φ : l'ensemble des clauses initiales

Σ : l'ensemble des clauses apprises.

Algorithm 2 Les solveurs SAT modernes

Tant que $\square \notin \Phi \cup \Sigma$ **faire**
  $C \leftarrow apprendreClause()$
  $\Sigma = \Sigma \cup C$
  Si $tropPlein(\Sigma)$ **Alors**
    $\Delta = clausesAEffacer(\Sigma)$
    $\Sigma = \Sigma \backslash \Delta$
  Fin Si
Fin Tant que

Les solveurs SAT modernes parallèles

Φ : l'ensemble des clauses initiales

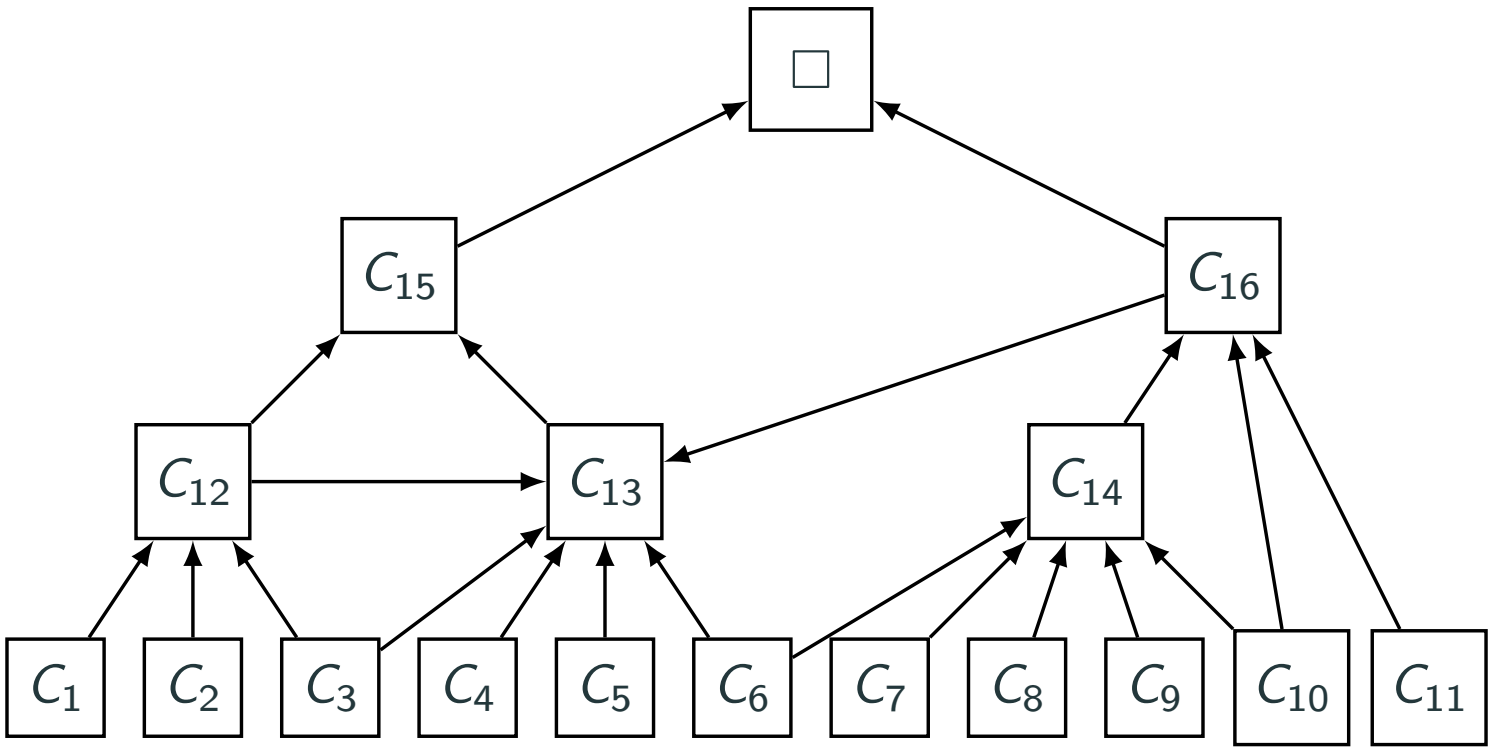
Σ : l'ensemble des clauses apprises.

Algorithm 3 Les solveurs SAT modernes parallèles

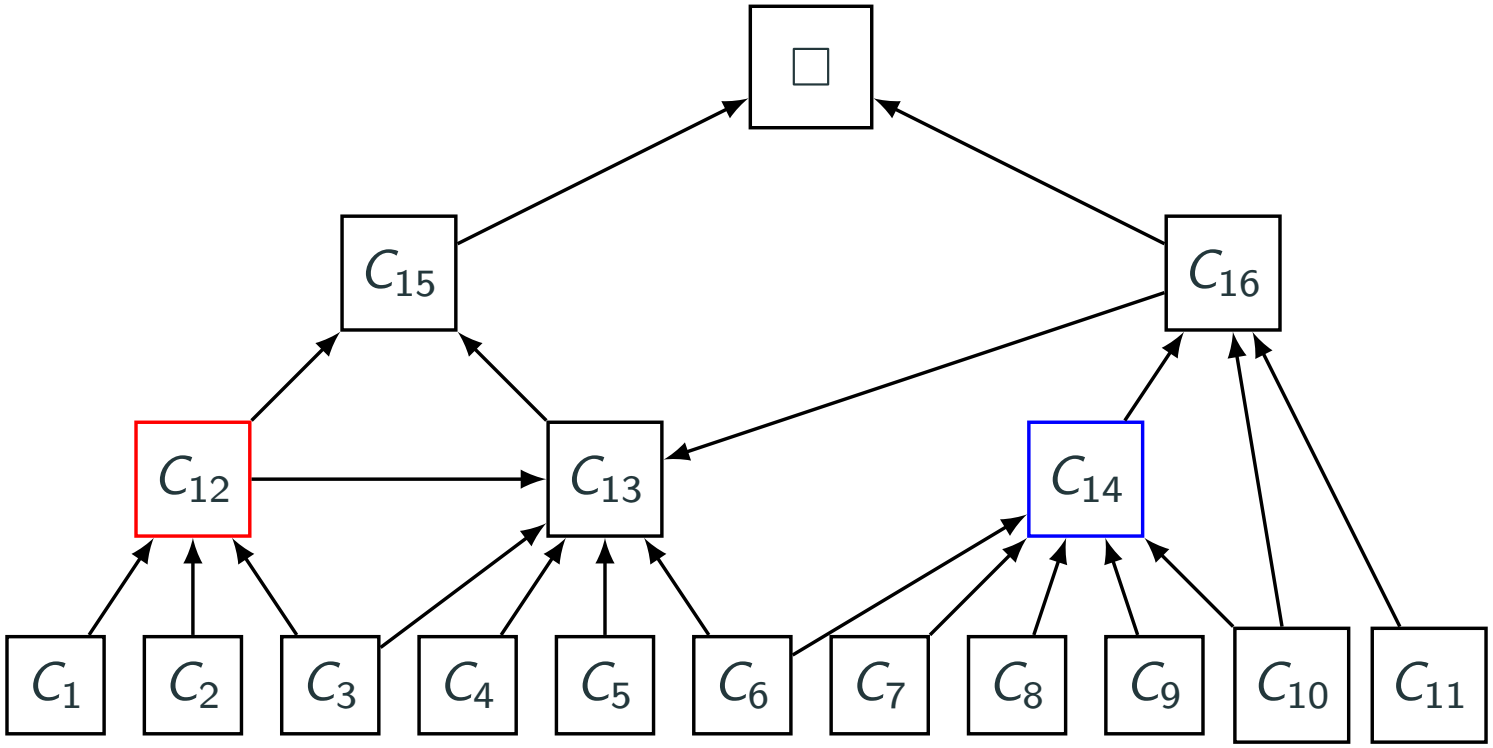
Tant que $\square \notin \Phi \cup \Sigma$ **faire**
  $C = apprendreClause()$
  $\Sigma = \Sigma \cup C$
  $Exporter(C)$
  $\Sigma = \Sigma \cup importerClause()$
  Si $tropPlein(\Sigma)$ **Alors**
    $\Delta = clauseAEffacer(\Sigma)$
    $\Sigma = \Sigma \backslash \Delta$
  Fin Si
Fin Tant que

Problématique

Reprenons la même résolution que précédemment, mais observons le comportement lorsque l'on utilise deux solveurs.



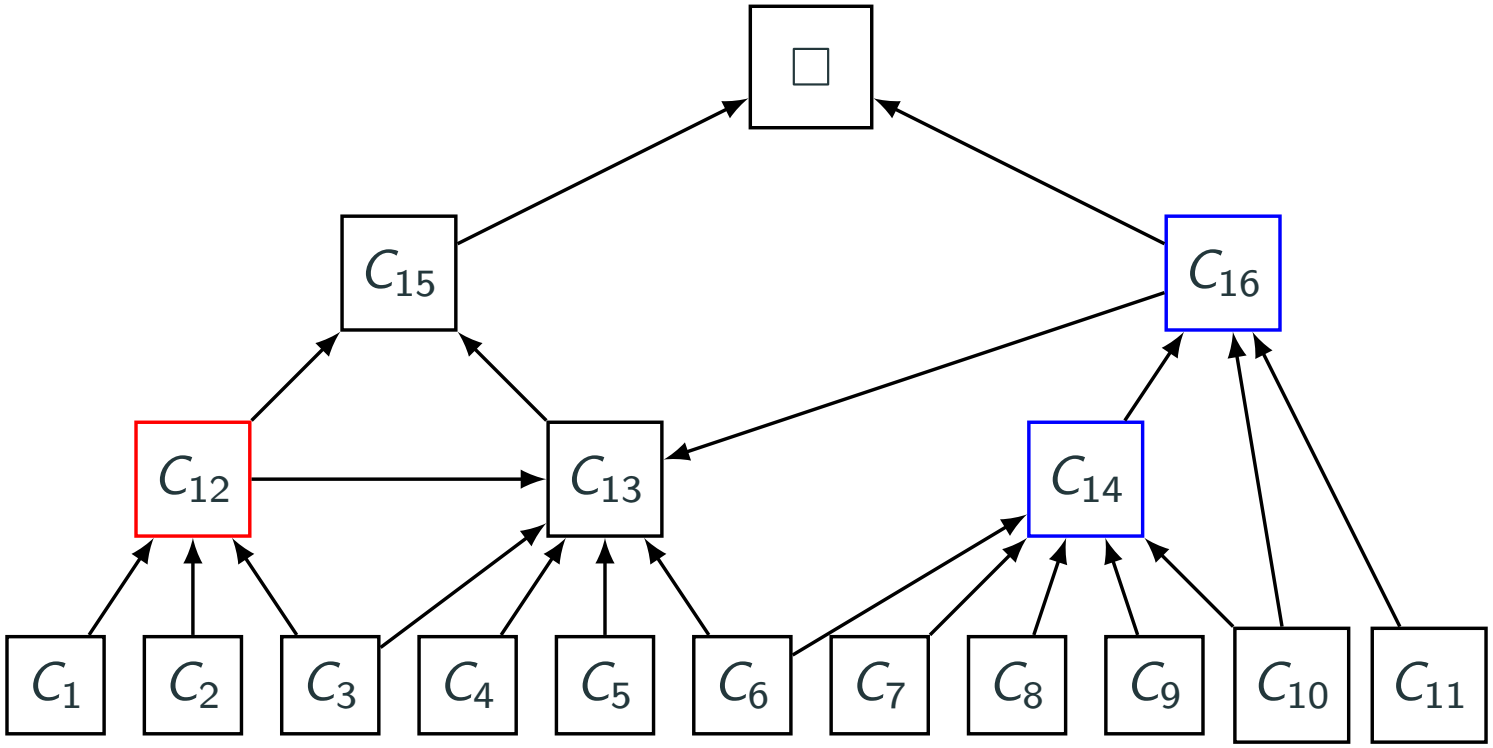
La parallélisation



Légende

- 1^{er} solveur
- 2^{eme} solveur

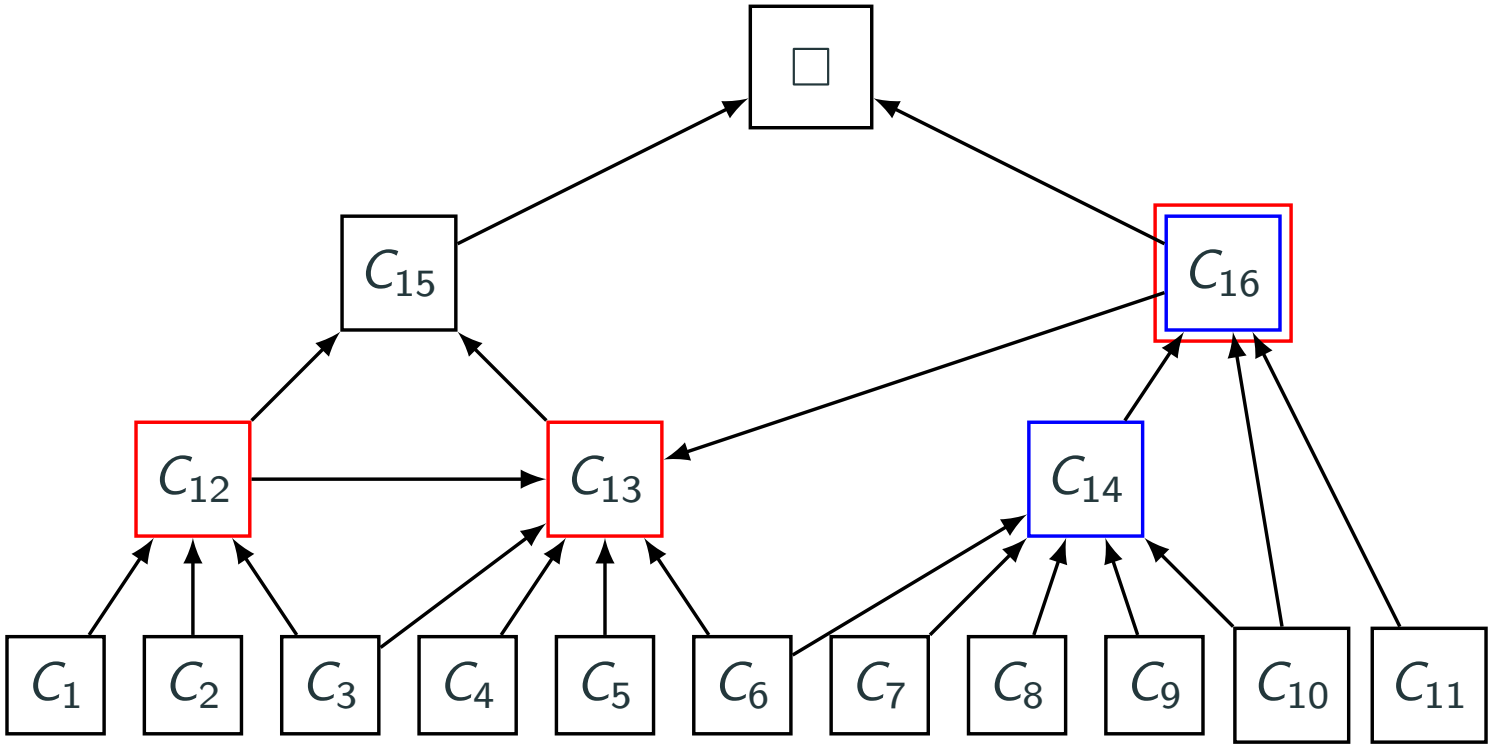
La parallélisation



Légende

- 1^{er} solveur
- 2^{eme} solveur

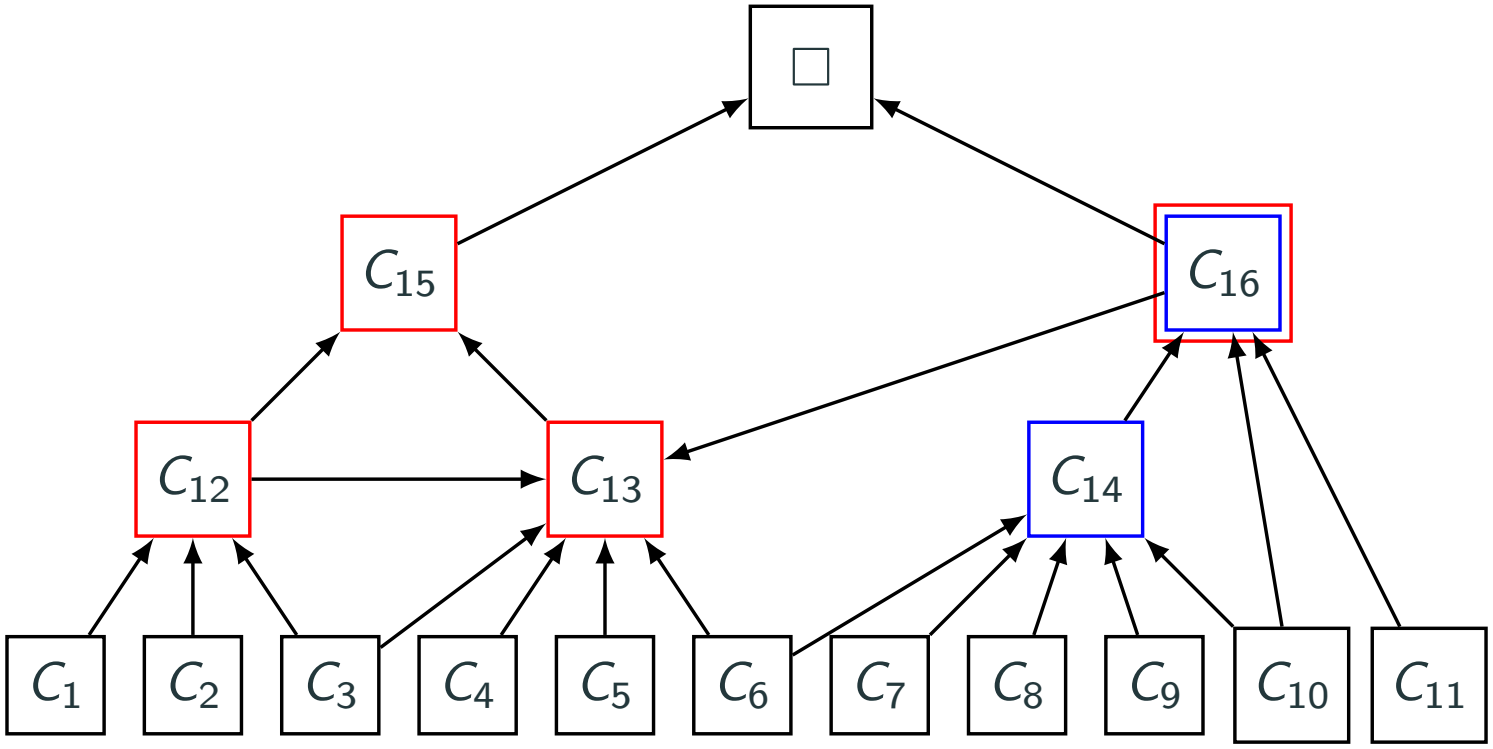
La parallélisation



Légende

- 1^{er} solveur
- 2^{eme} solveur

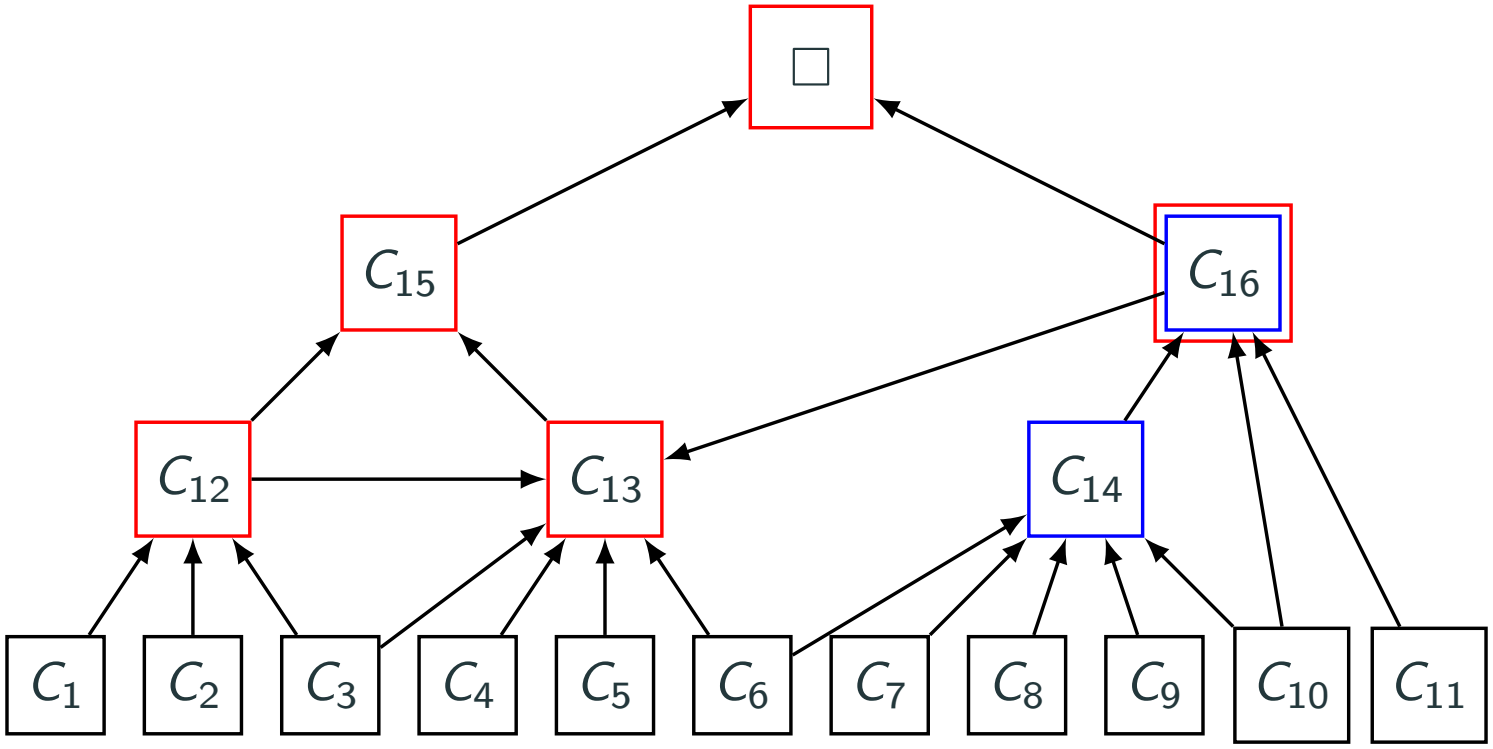
La parallélisation



Légende

- 1^{er} solveur
- 2^{eme} solveur

La parallélisation



Légende

- 1^{er} solveur
- 2^{eme} solveur

Parallélisation idéale

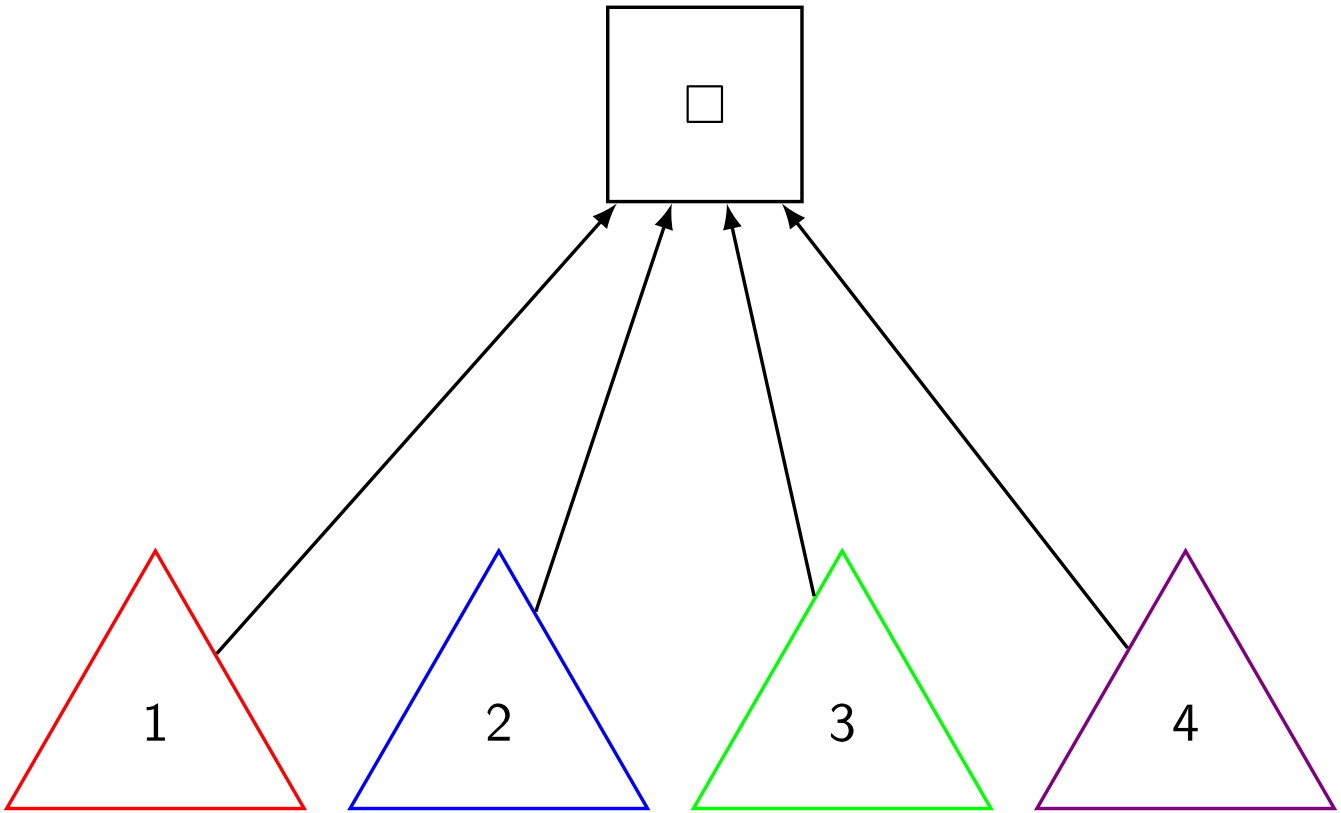


Figure 3 – Représentation d'une parallélisation idéale.

Parallélisation en pratique

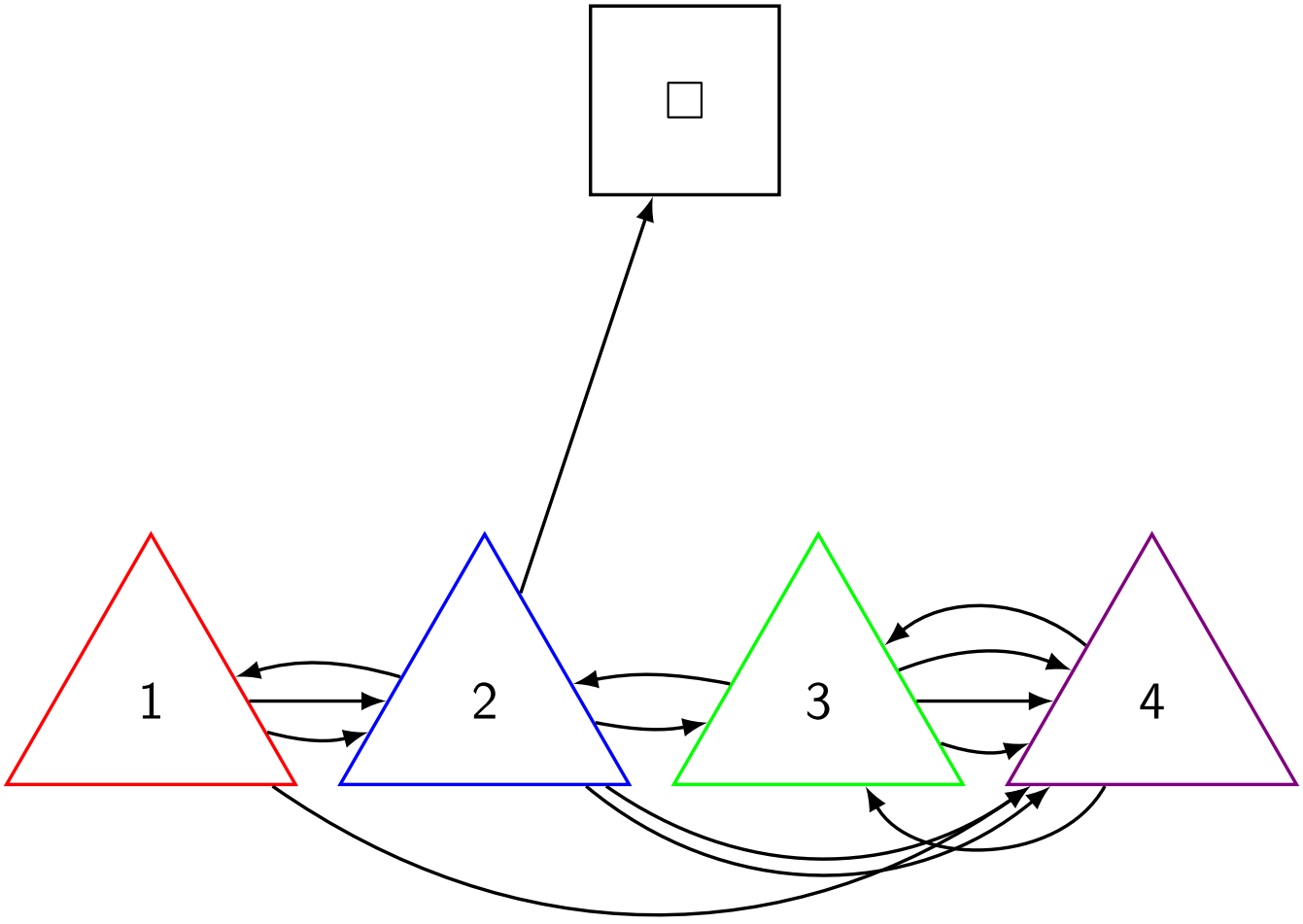


Figure 4 – Représentation d'une parallélisation en pratique.

Les questions

- Comprendre le type de graphe construit
- Existe-t-il des limites aux méthodes de parallélisation actuelle ?
- Avons-nous une **clusterisation** lors de la recherche ?
- Privilégier une méthode d'**exploration** ou d'**intensification** ?

Représentation abstraite de la preuve

Représentation

Représentons une preuve **non-parallèle** par le schéma suivant.

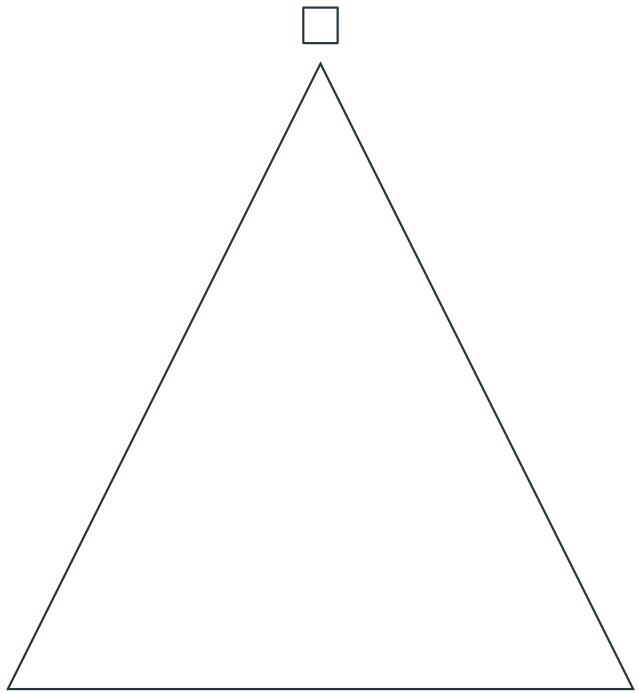


Figure 5 – Représentation abstraite de la preuve

Représentation abstraite de la preuve

Représentation

La preuve **parallèle** est-elle plus large mais moins profonde ?

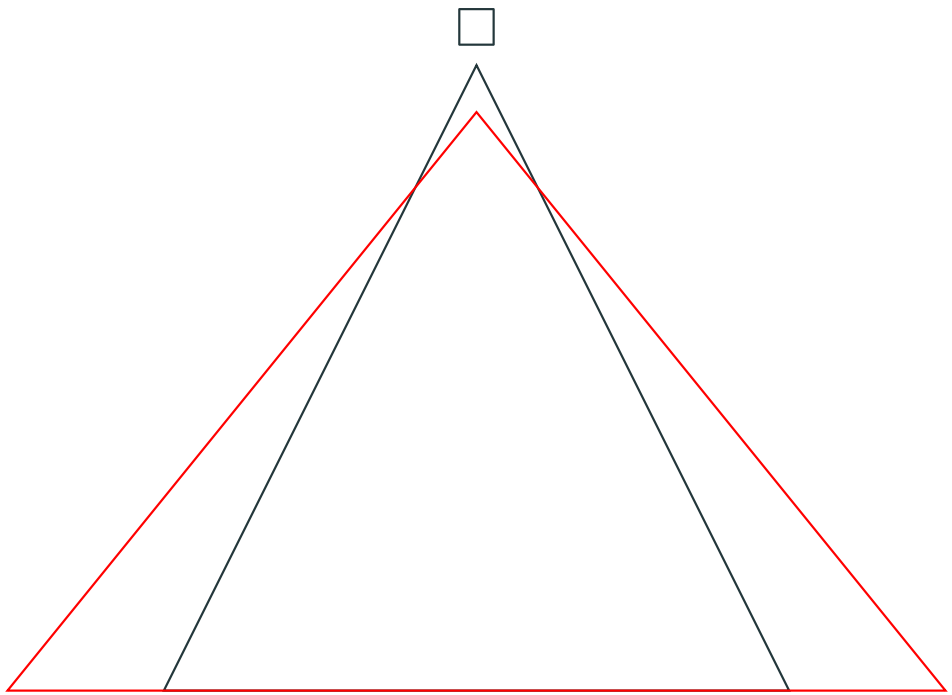


Figure 6 – Représentation abstraite de la preuve

$\Rightarrow$ Peut-être privilégier une méthode d'exploration.

Forme de la preuve

Représentation

La preuve **parallèle** est-elle moins large ?

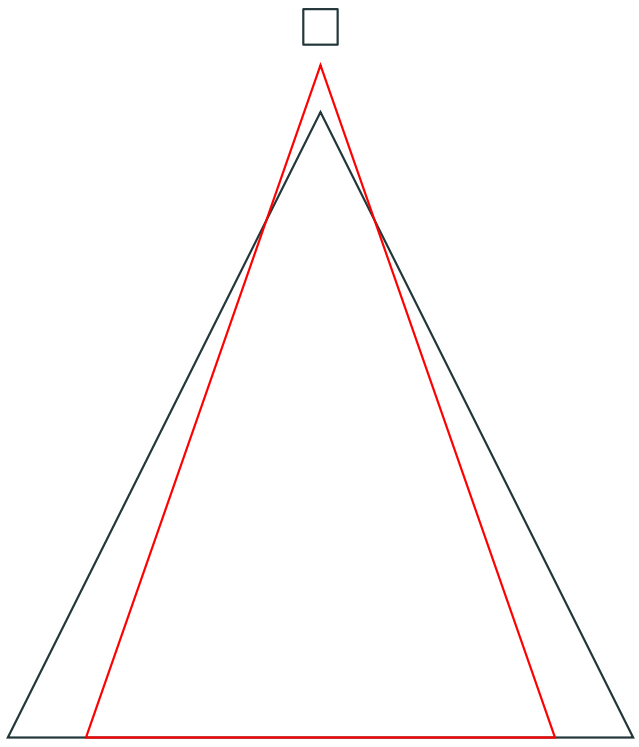


Figure 7 – Représentation abstraite de la preuve

$\Rightarrow$ Peut-être privilégier une méthode d'intensification.

Les expériences

Conditions d'expériences

Le solveur SAT

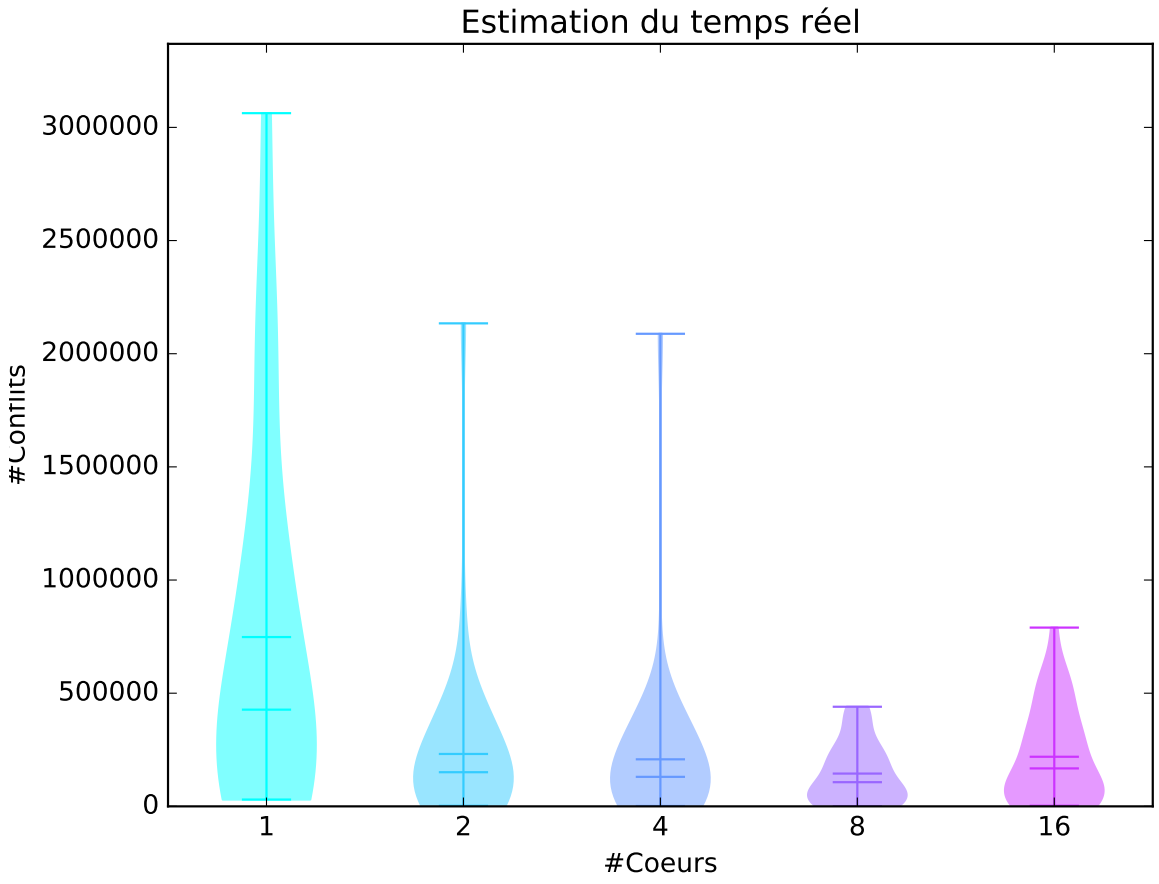
Nous avons utilisé une version modifiée de **Glucose**, un solveur SAT co-developpé au **LaBRI**.

Problèmes utilisés

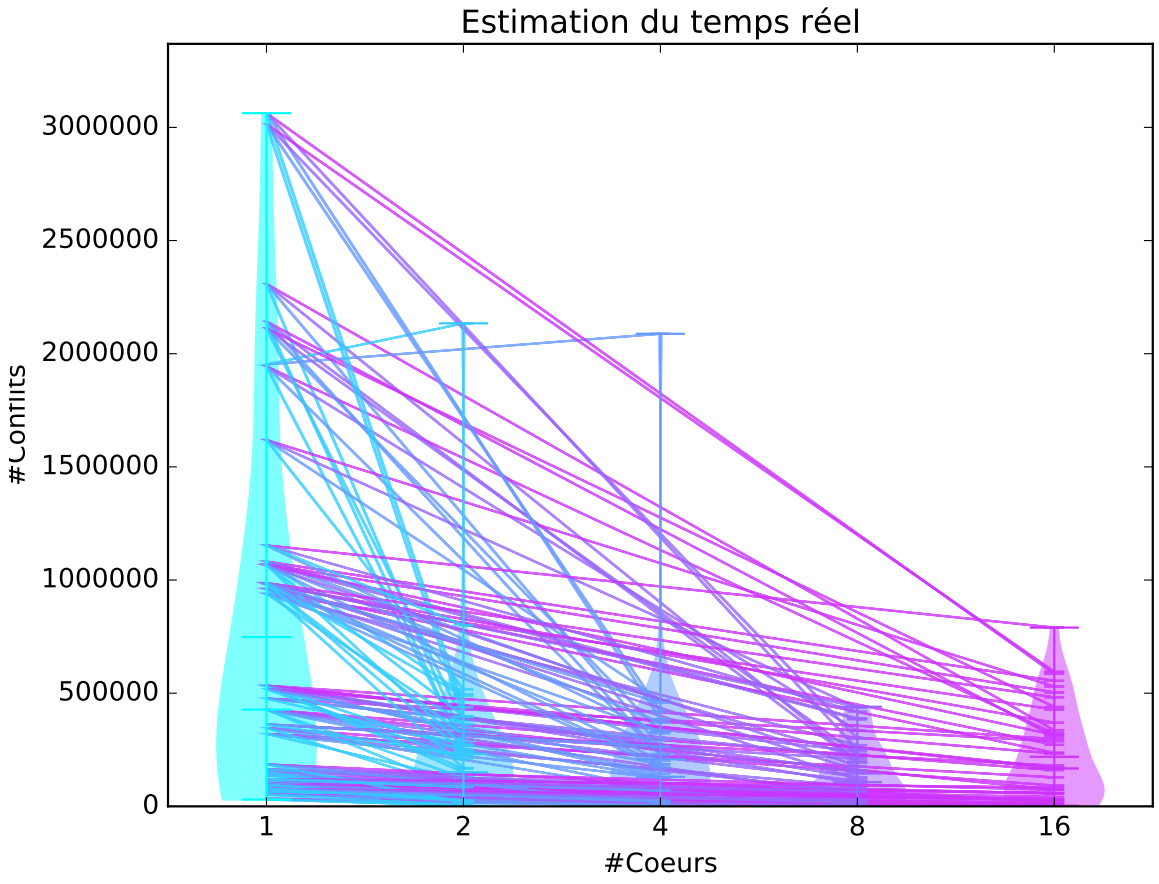
Les calculs ont été lancés sur un ensemble de **problèmes** qui sont :

- représentatifs des différents problèmes SAT,
- de taille raisonnable,
- des problèmes industriels.

Etude expérimentale



Etude expérimentale



Taille de la preuve

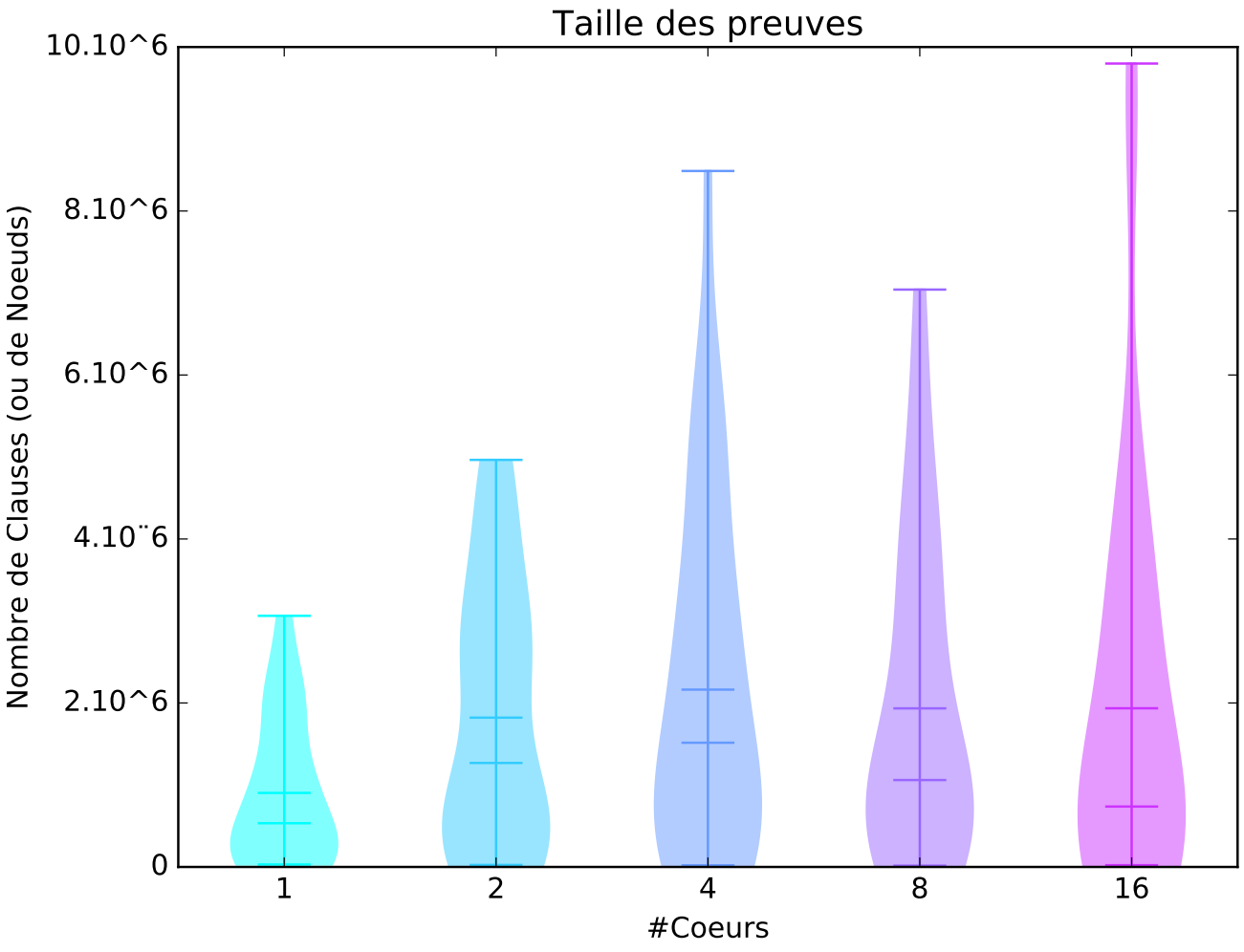
Etat de l'art

On ne sait pas si une augmentation du nombre de solveurs implique une augmentation de la taille de la preuve.

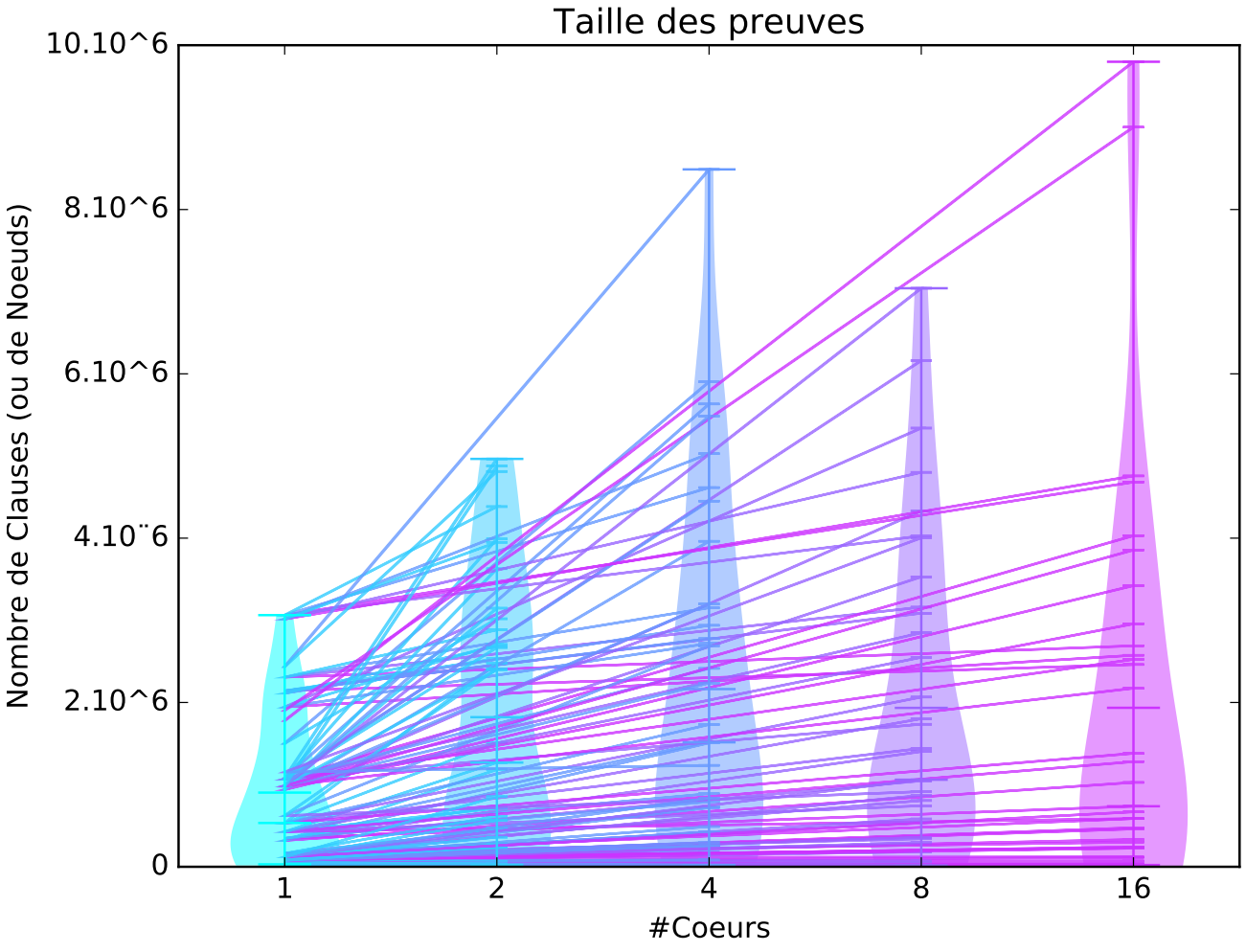
Intêret

Les solveurs SAT peuvent être certifiés grâce à leurs preuves, donc une petite preuve peut être intéressante.

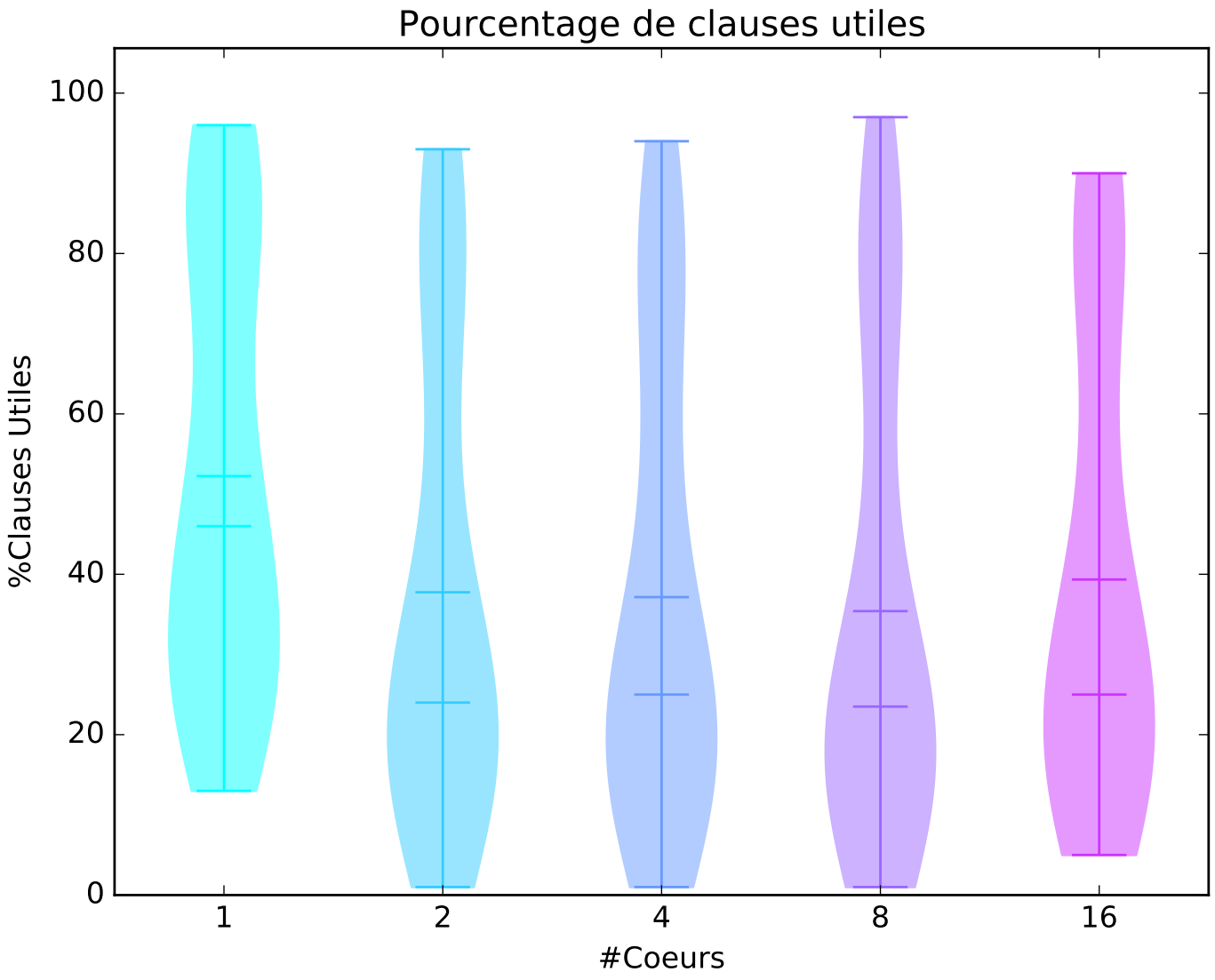
Etude expérimentale



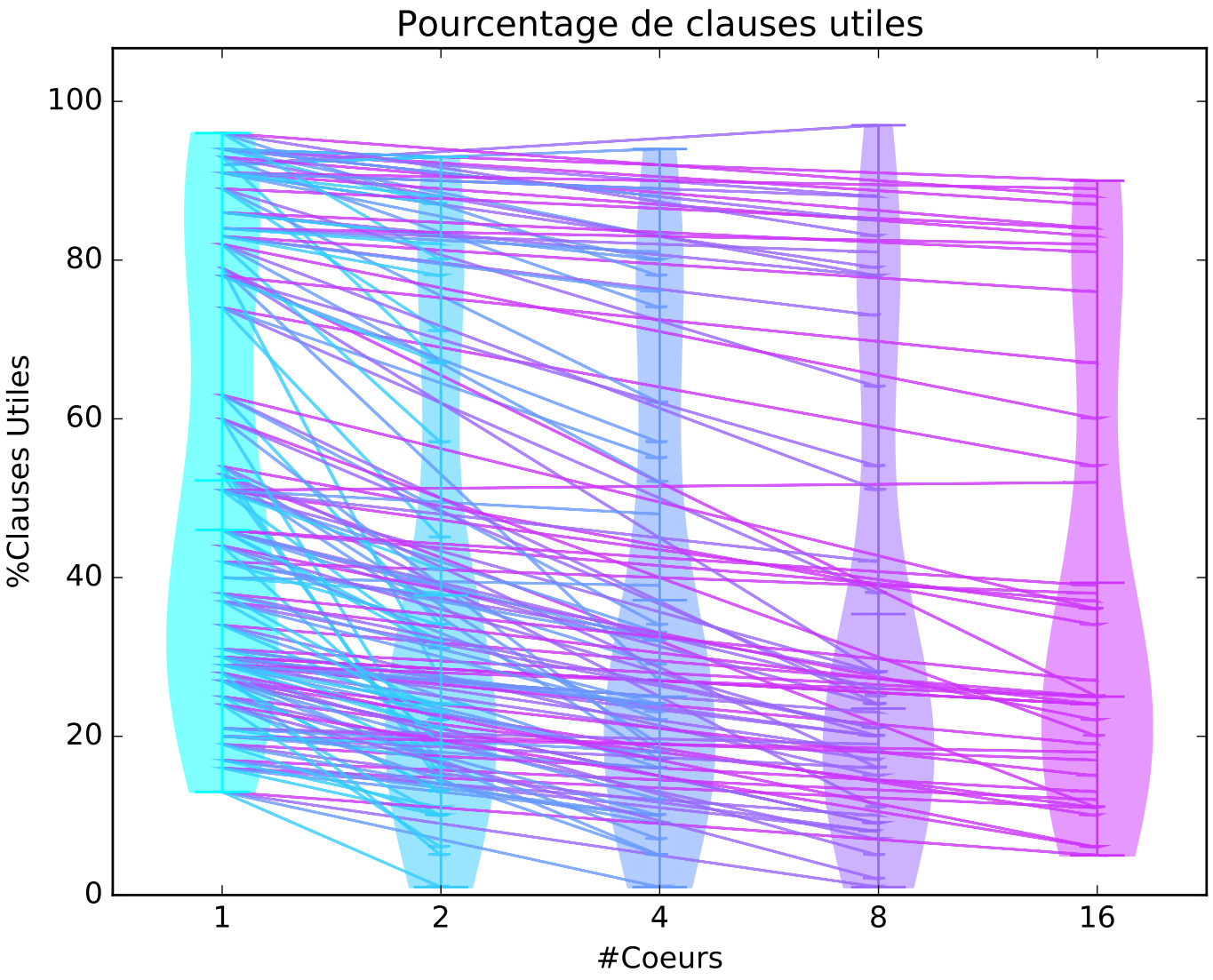
Etude expérimentale



Etude expérimentale



Etude expérimentale



Représentation des preuves

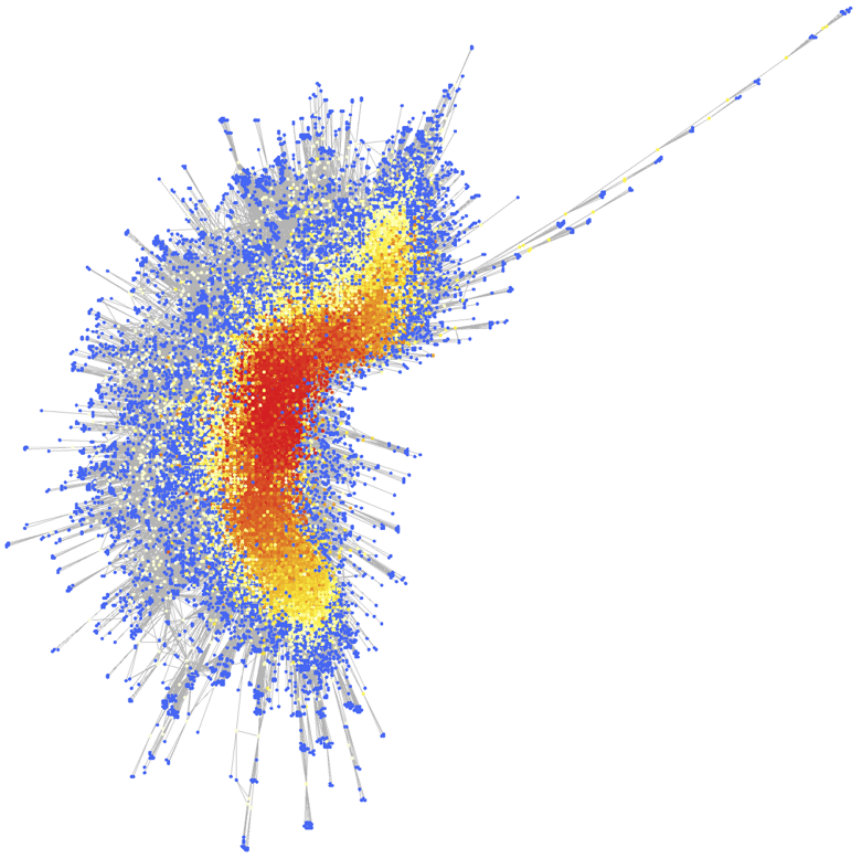


Figure 8 – Preuve en séquentiel

Informations

Formule
clauses :
55585
variables :
50076

Conflits
conflits :
59792
Temps calcul :
6s

Graphe
sommets :
51274
arêtes :
960620

Représentation des preuves

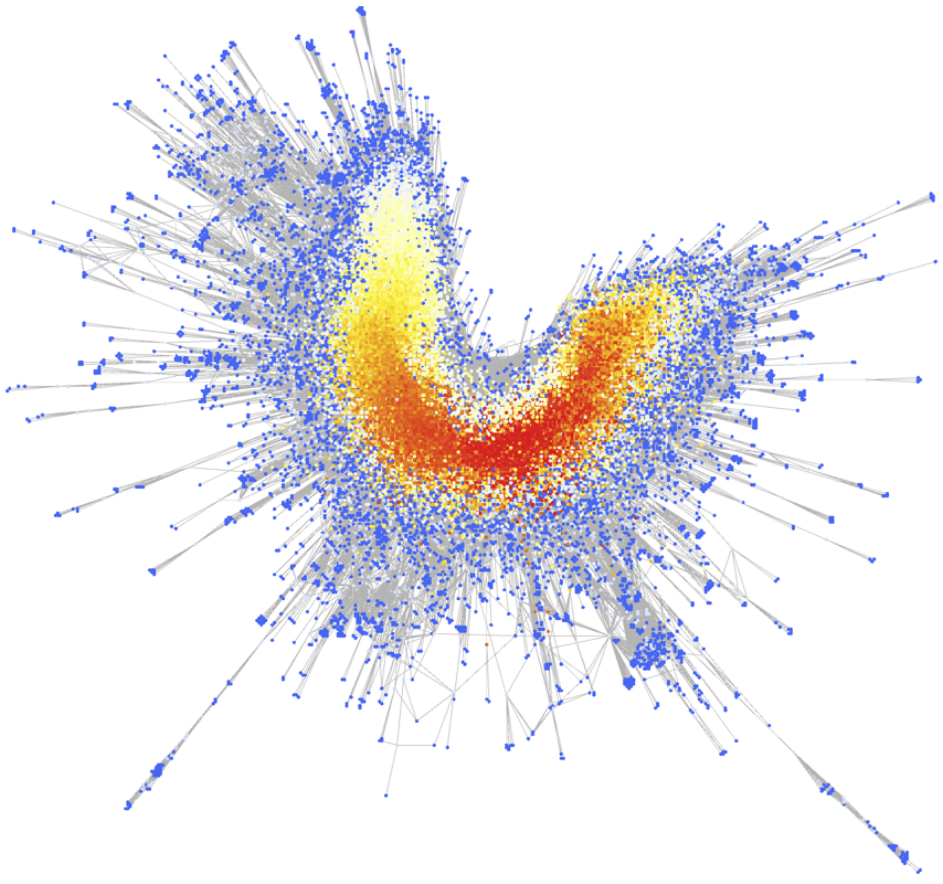


Figure 9 – Preuve avec **16** coeurs

Informations

Formule

clauses :
55585

variables :
50076

Conflits

conflits/solveur :
8339

conflits totaux :
133424

Graphe

sommets :
63301

arêtes :
1213688

La *coreness*

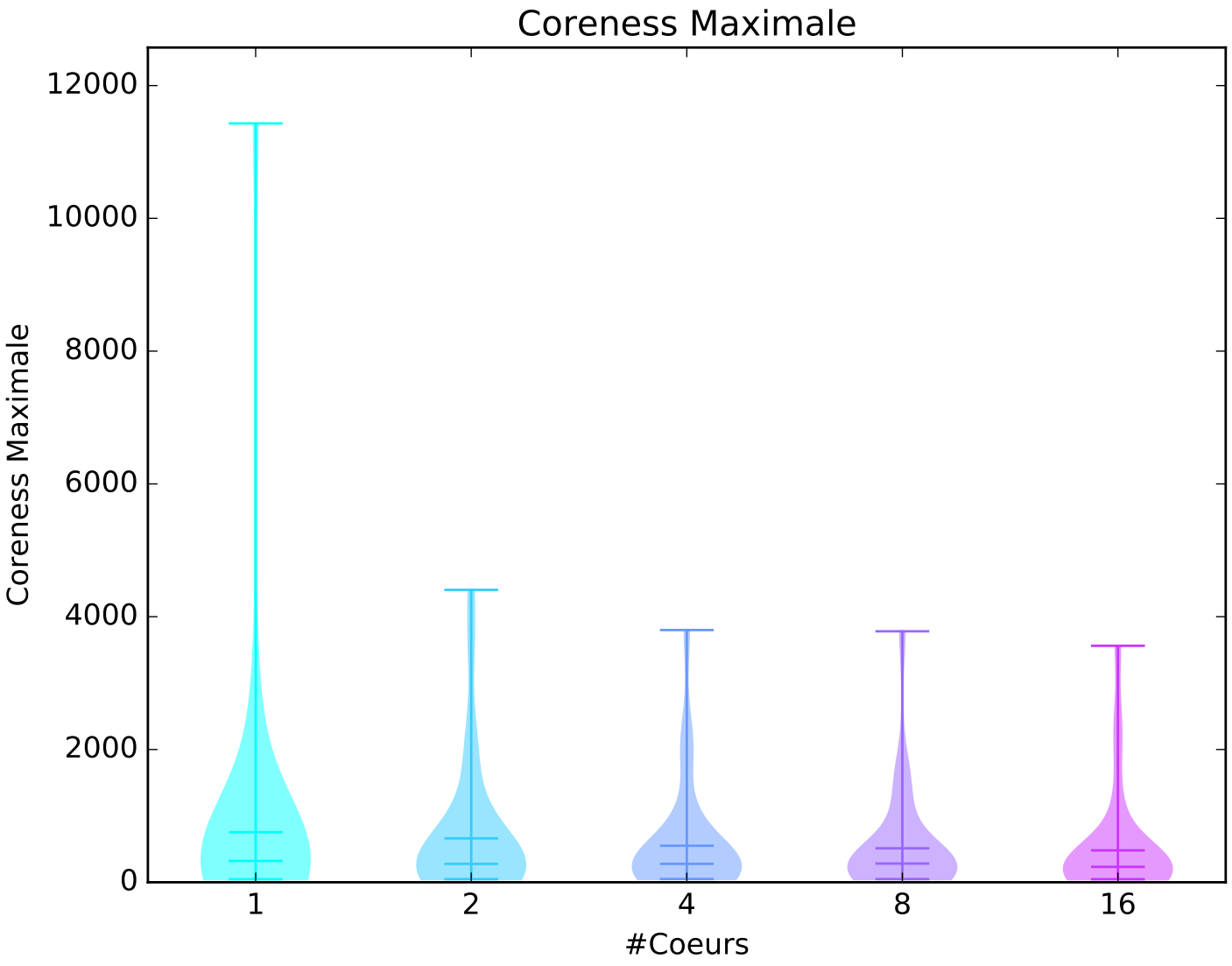
Le k-core

Le ***k-core*** d'un graphe orienté est un sous-graphe orienté maximal dans lequel chaque sommet à un degré sortant d'au moins k.
La *coreness* d'un sommet est k si il appartient au ***k-core*** mais pas au ***k+1-core***.

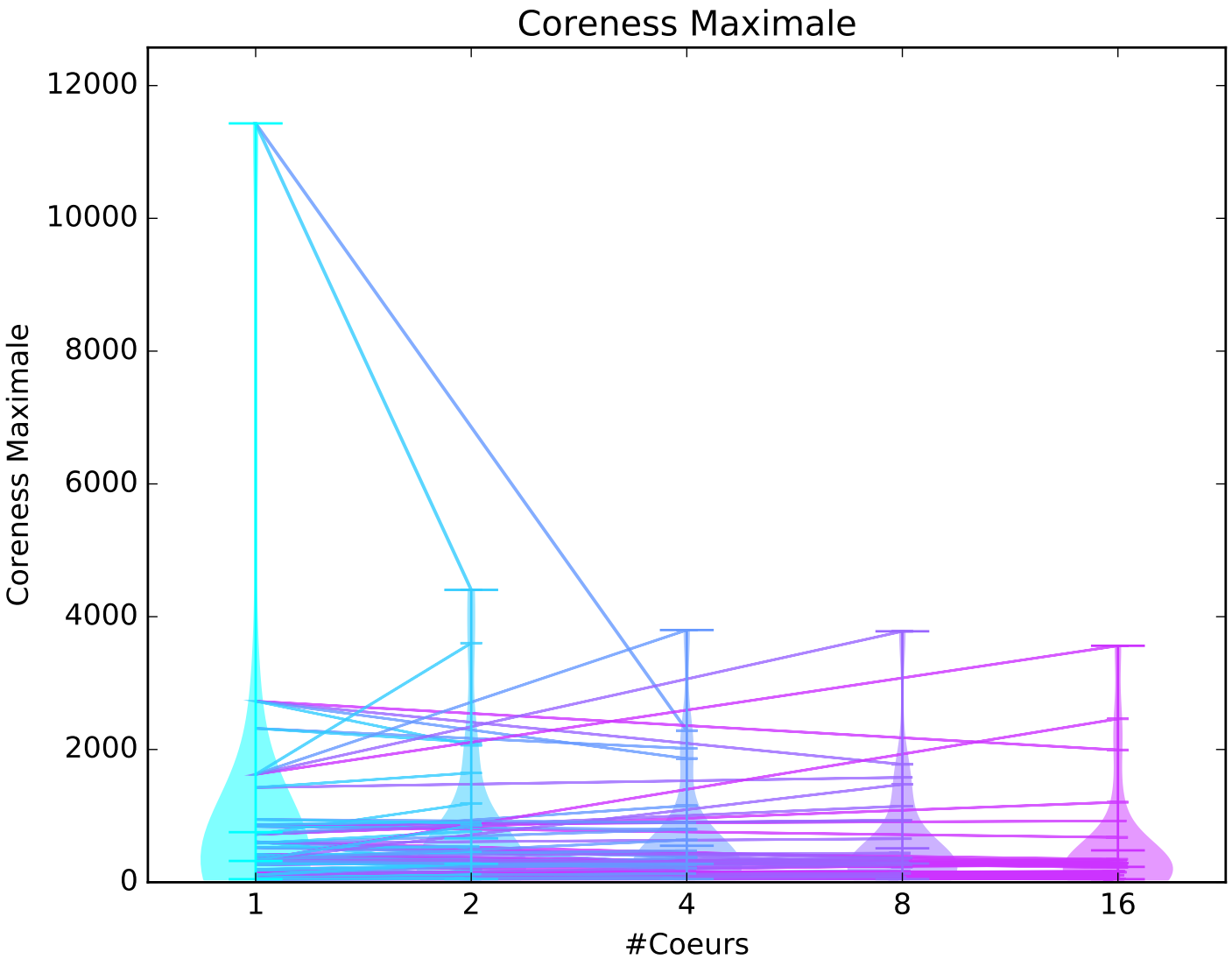
Utilité de la mesure

La *coreness* permet d'étudier la densité d'un graphe dans des zones locales.

Etude expérimentale



Etude expérimentale



Conclusion

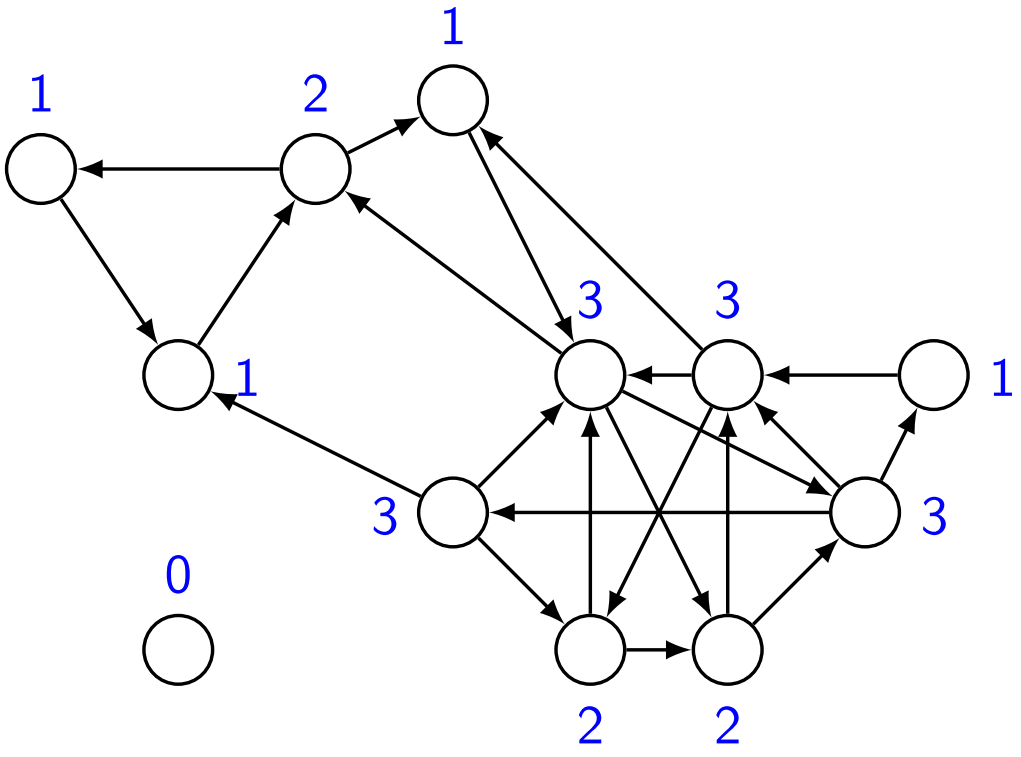
Pistes d'études

- Comprendre et analyser le ***k*-*core*** de la preuve,
- Prédire le coreness de la preuve,
- Prédiction des clauses importantes,

Merci de votre attention.

Exemple

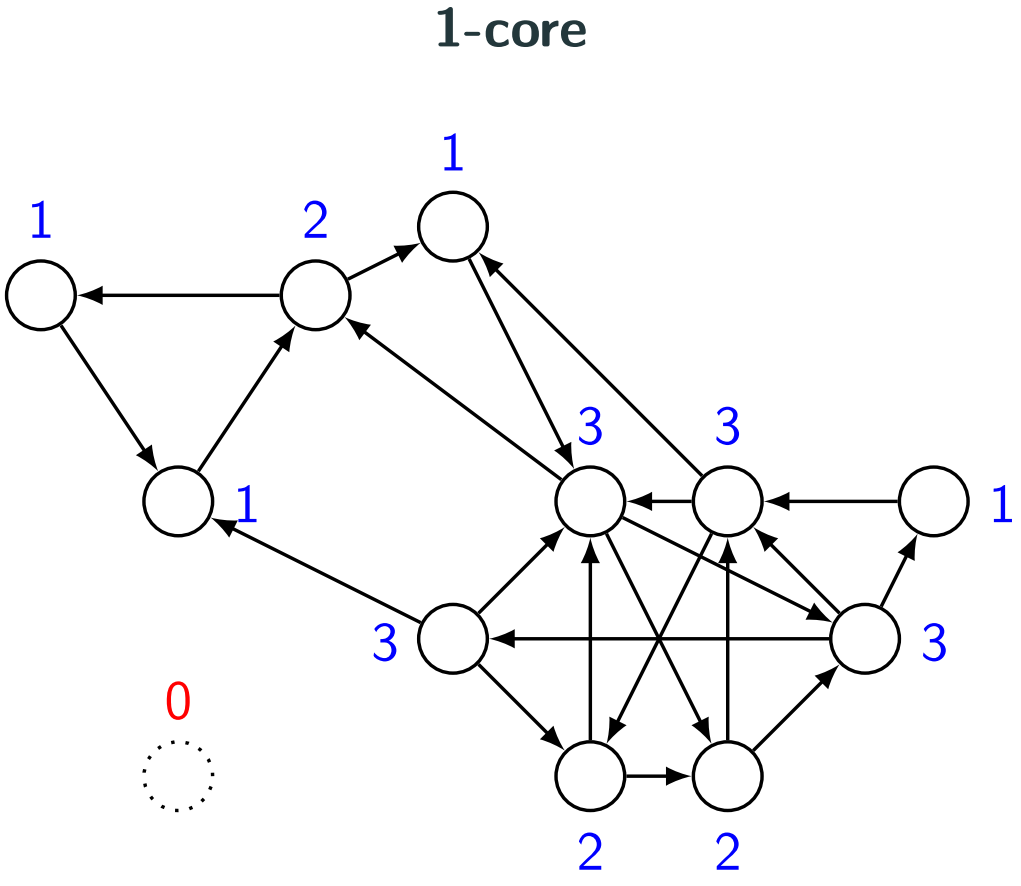
Graphe de départ : 0-core



Légende

- Degré sortant dans le k-core

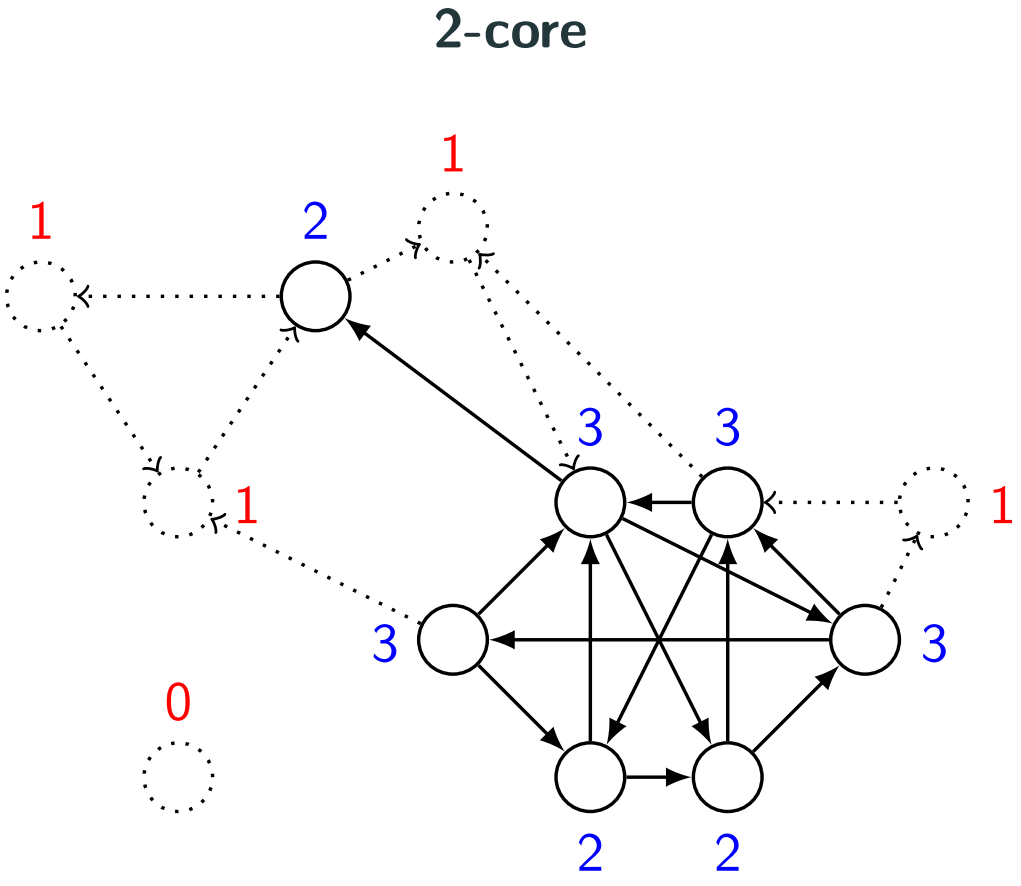
Exemple



Légende

- Degré sortant
- Coreness du sommet

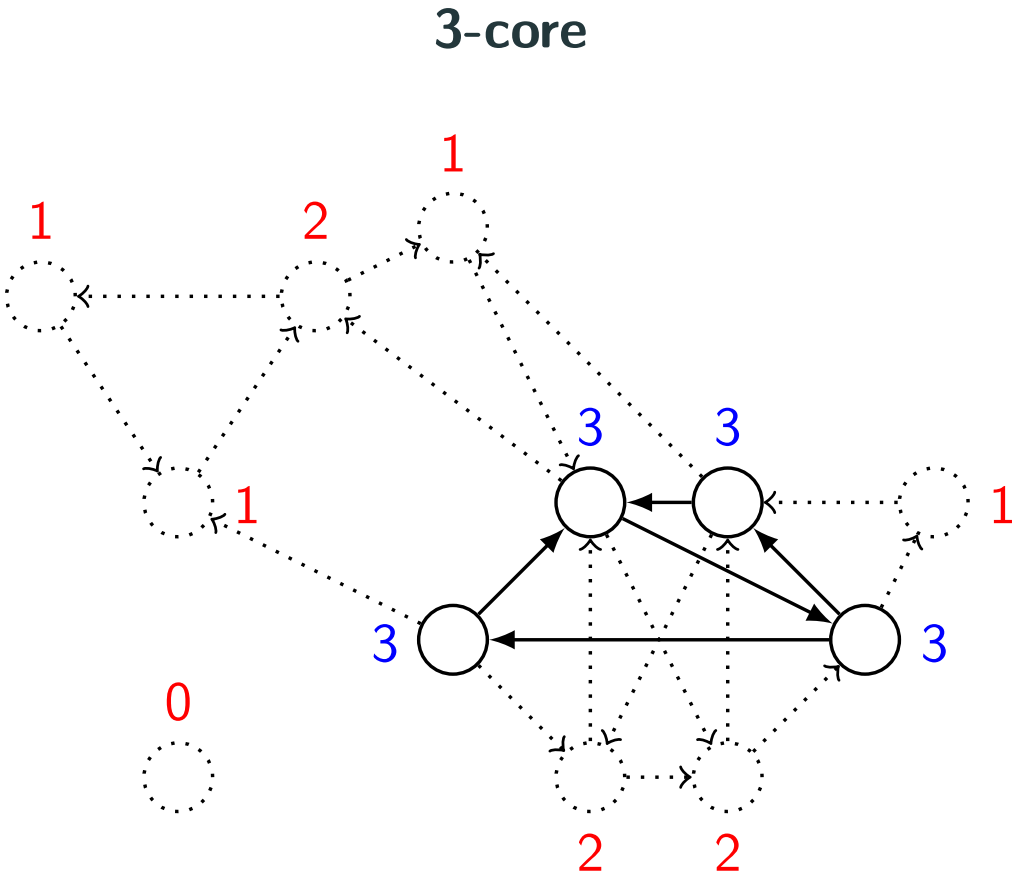
Exemple



Légende

- Degré sortant
- Coreness du sommet

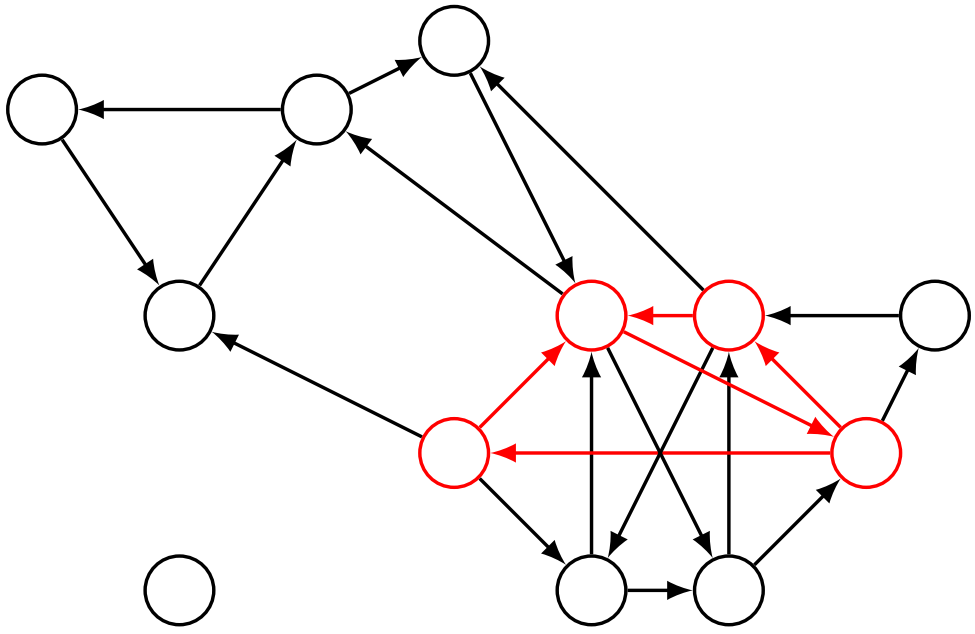
Exemple



Légende

- Degré sortant
- Coreness du sommet

Exemple



La *coreness* maximale de ce graphe est de **3**.