Problèmes des k-cliques

2ème présentation

GROS Loric, SAMBET Mathys, MUNOZ Matéo

Rappel du problème

Dans un graphe non orienté G=(V,E)

V est l'ensemble des sommets

E est l'ensemble des arêtes

Une clique est un sous-ensemble de sommets tels que chaque paire de sommets dans ce sous-ensemble est connectée par une arête

Trouver toute clique de taille k

Solver utilisé: OR-Tools

Avantage:

- Simple
- Efficace
- Bien documenté

Inconvénient:

- Gestion de ressources sur grand graphe

Solver utilisé: Choco

Avantage:

- Facilement paramétrable
- Flexible
- Bien documenté

Inconvénient:

- Gestion de ressources sur grand graphe

Modèle

Paramètres

- n: Le nombre de sommets dans le graphe.
- k: La taille de la clique (nombre de sommets dans la clique).
- $E \in \{0,1\}^{n \times n}$: La matrice d'adjacence du graphe, où $E[i,j] = 1$ si les sommets i et j sont reliés, et $E[i,j] = 0$ sinon.

Modèle

Variables

- $\mathbf{C} = (c_1, c_2, \ldots, c_k)$: Les sommets de la clique, où chaque c_i est un entier tel que $c_i \in \{1, 2, \ldots, n\}$ pour $i \in \{1, 2, \ldots, k\}$.

Modèle

Contraintes

- **Contrainte 1 : Distinction des sommets de la clique**

$$c_i \neq c_j \quad \forall i, j \in \{1, 2, \ldots, k\}, i \neq j$$

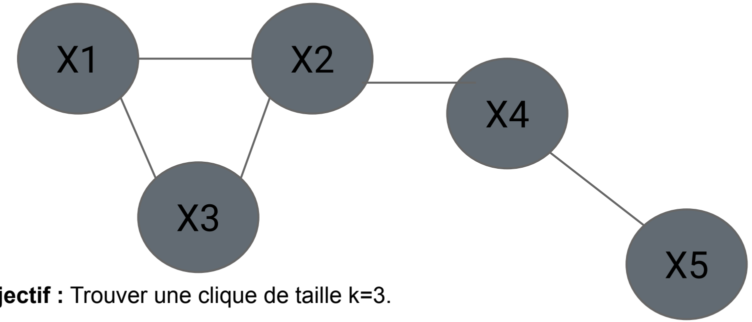
- **Contrainte 2 : Connexité entre tous les sommets de la clique**

$$E[c_i, c_j] = 1 \quad \forall i, j \in \{1, 2, \ldots, k\}, i \neq j$$

Modèle : exemple

V={1,2,3,4,5}

E={(1,2),(2,3),(1,3),(4,5)}



Objectif : Trouver une clique de taille k=3.

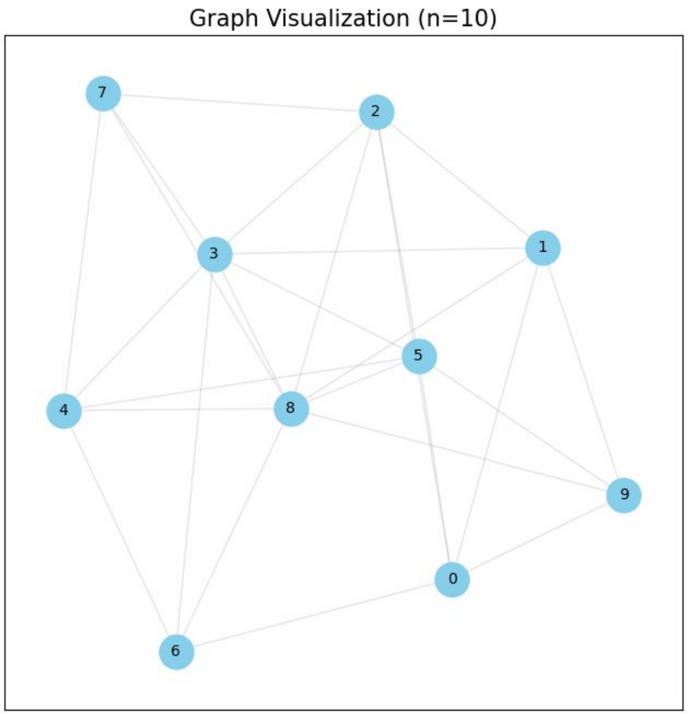
- **Contrainte de taille** : On cherche au moins 3 sommets qui forment une clique.
- $X1+X2+X3 \geq 3$

Les sommets 1,2,3 satisfont la contrainte de taille et la contrainte d'arête car ils sont connectés entre eux.

Ici c'est la seul 3-clique, donc notre résultat est 1.

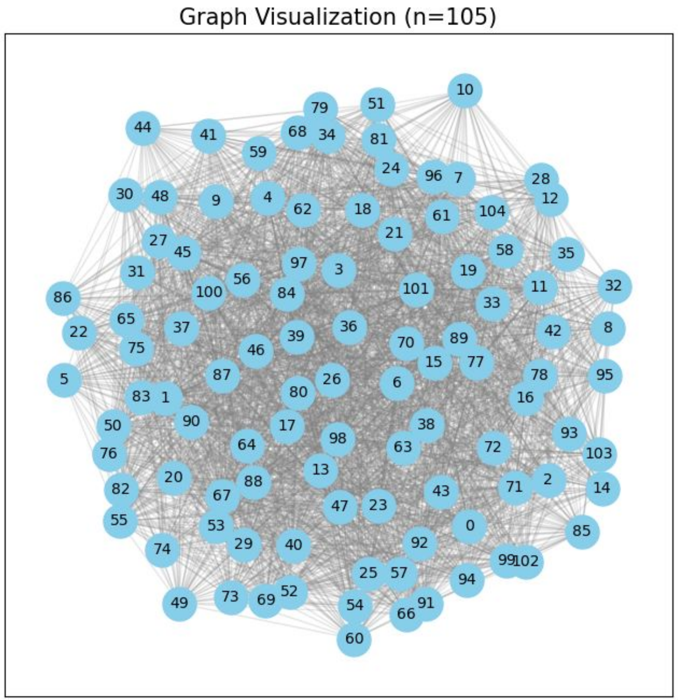
Benchmark

Instance générées



Benchmark

Instance générées

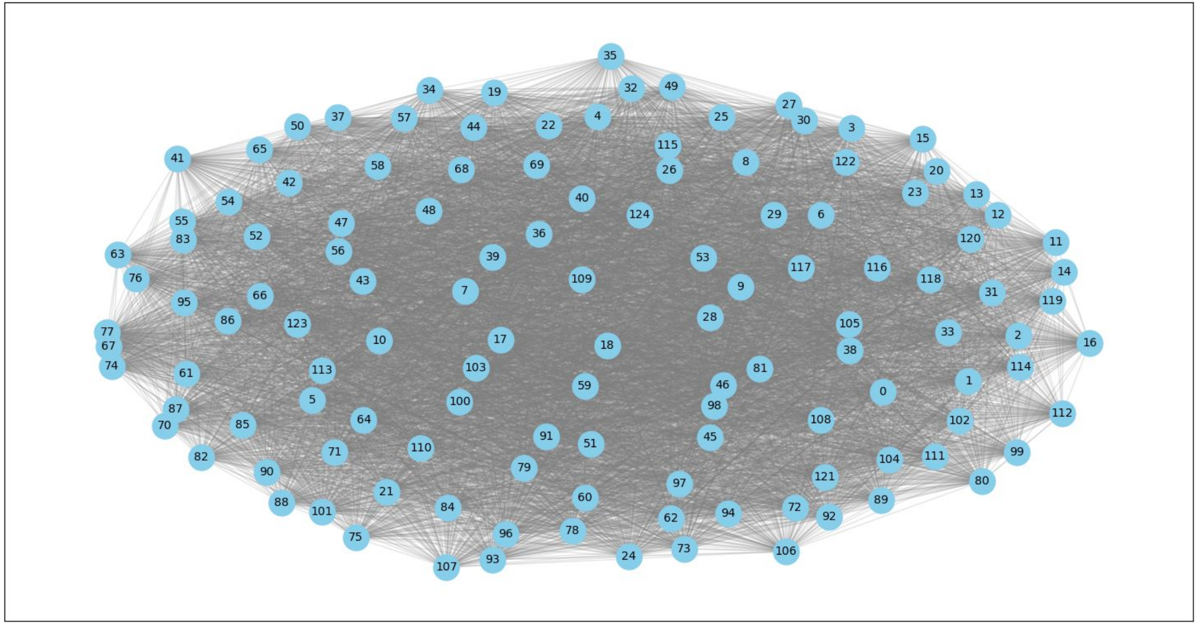


Benchmark - Instance DIMACS

- 2 graphes (125 et 250 noeuds).
- À la base pour la recherche de clique maximum.
- Plus grande clique possible :
    - 34 pour le graphe de taille 125.
    - 44 pour le graphe de taille 250.

source : https://iridia.ulb.ac.be/~fmascia/maximum_clique/DIMACS-benchmark

Graph Visualization (n=125)



Méthodes complètes

OR-Tools

Deux paramétrages:

- Paramétrage par défaut
- Notre paramétrage
  - modification de la prise de décision : degrés des noeuds

Benchmark utilisés:

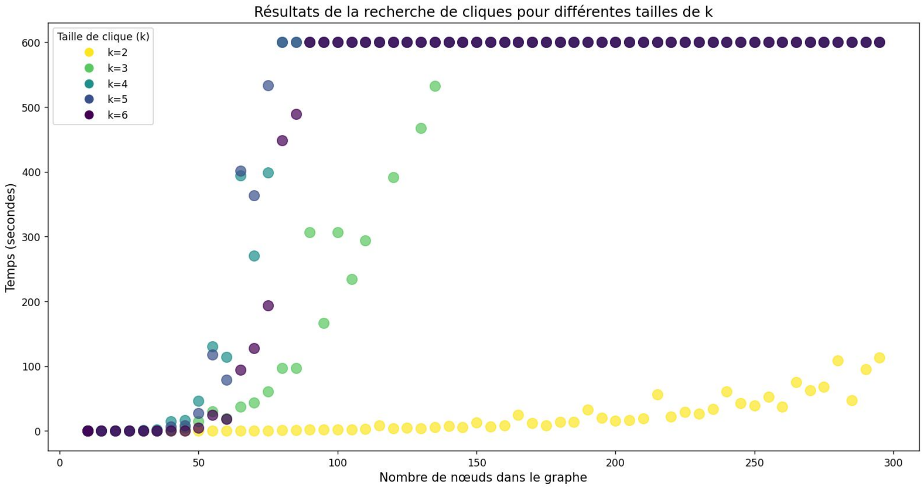
- Graphe générées aléatoirement (FlatZinc, de 10 à 295 noeuds) soit 58 graphes, une instance par valeur de k.
- k de 2 à 6.

Soit 232 graphes générées

Méthodes complètes

OR-Tools

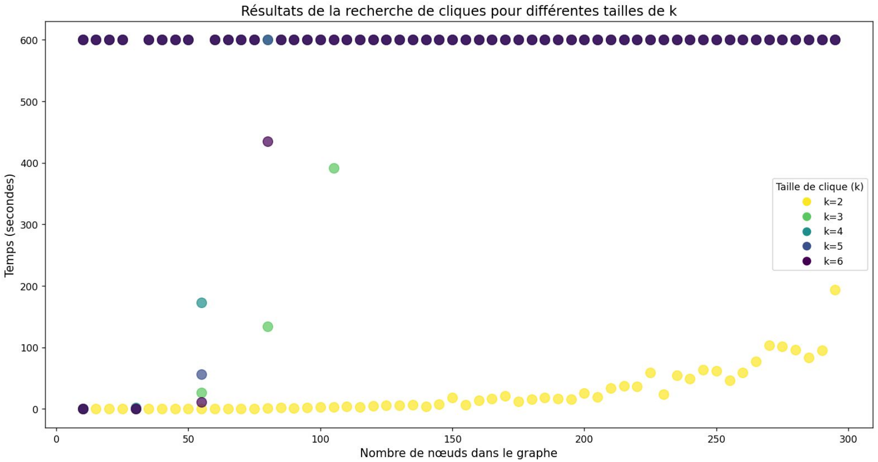
Paramétrage par défaut



Méthodes complètes

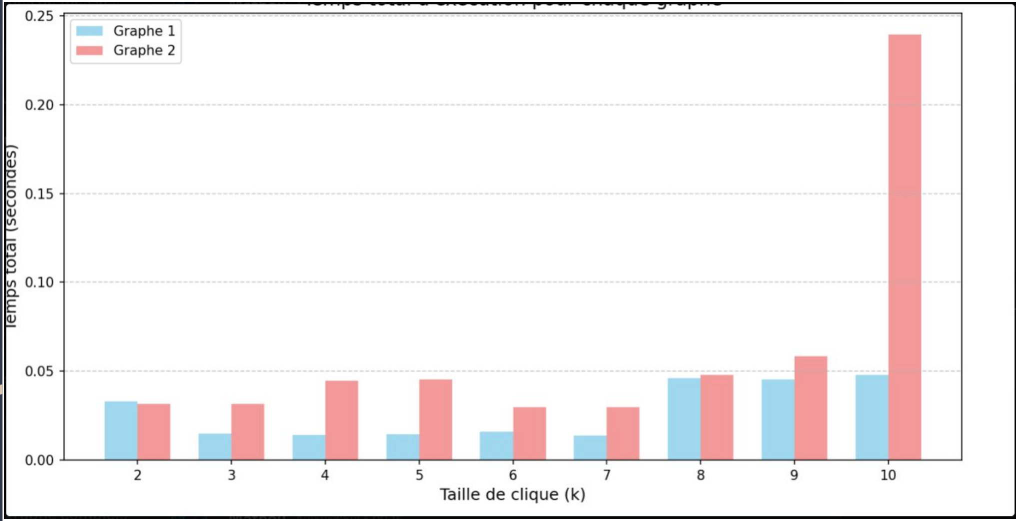
OR-Tools

Ordre des noeuds décroissants



Méthodes complètes

OR-Tools



Méthodes complètes

Choco-Solver

Deux paramétrages:

- Paramétrage par défaut de choco
- Notre paramétrage

Benchmark utilisés:

- Les graphes générées aléatoirement (FlatZinc, de 10 à 295 noeuds avec k de 2 à 6)
- Instance DIMACS (125 noeuds et 250 noeuds, k de 2 à 44)

Méthodes complètes Choco-Solver Paramétrage par défaut

Heuristique utilisée : **DomOverWDeg**

But : Prioriser les variables ayant un fort impact potentiel sur la propagation des contraintes.

- **Concepts :**
  1. **Weighted Degree (WDeg) :** Nombre de contraintes connectées à une variable. Ex. : Si x3 est lié à 4 autres variables, WDeg(x3)=4.
  2. **Domaine d'une variable :** Ensemble des valeurs possibles pour une variable (Domaine(xi) = {0,1}

$$Score = \frac{Taille\ du\ domaine}{Degré\ pondéré}$$

Méthodes complètes Choco-Solver
Notre paramétrage

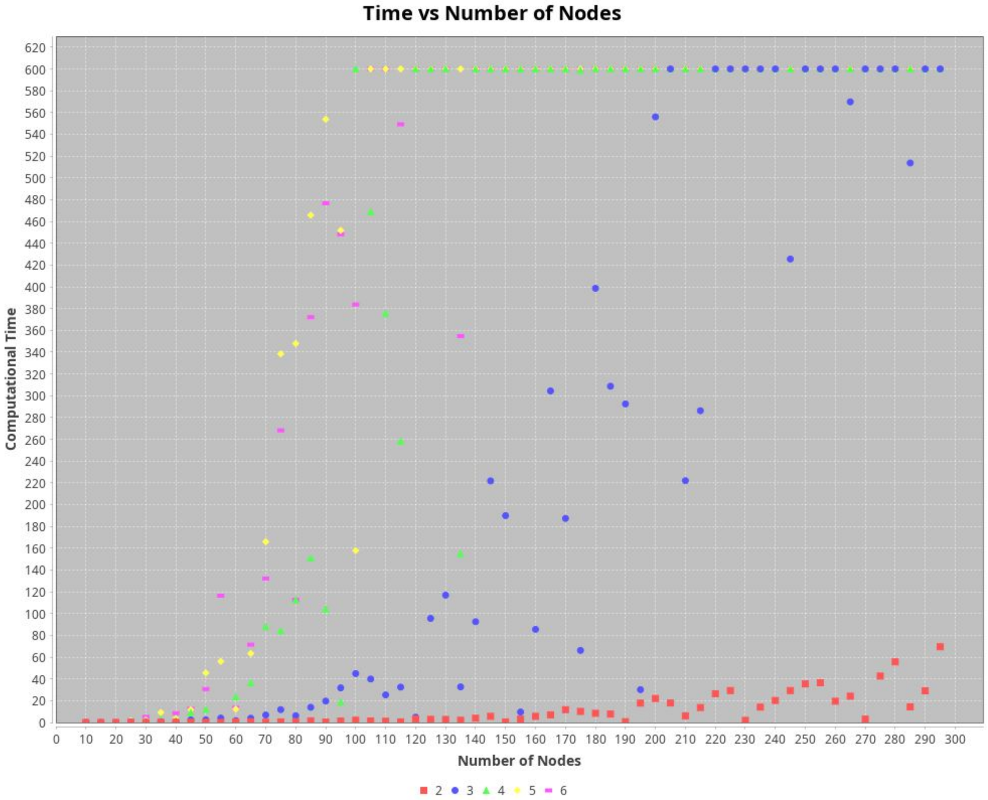
Stratégie utilisée **: Sélection des Variables par Priorité Dynamique**

But : Prioriser les variables ayant un impact maximal sur la propagation des contraintes pour améliorer l'efficacité de la recherche de solutions.

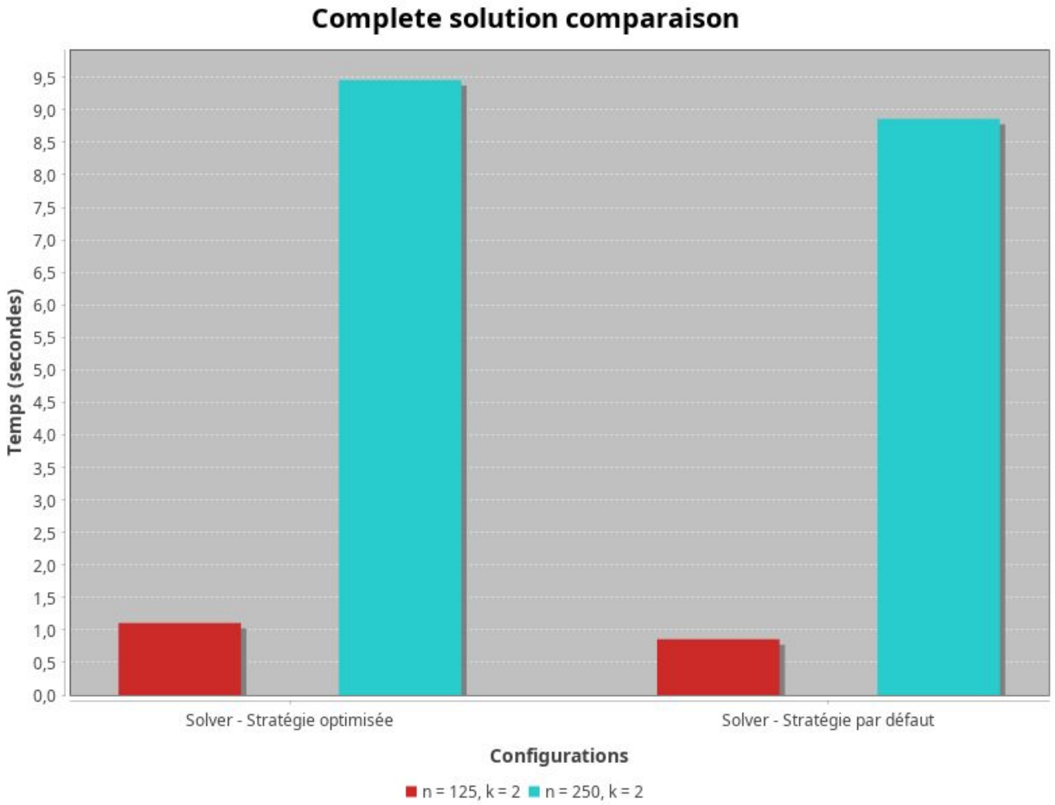
Fonctionnement :

- **Calcul des priorités** :
  - **Centralité dynamique** : Nombre de contraintes liées à la variable.
  - **Poids dynamique** :
    - **Degré résiduel** (contraintes non satisfaites),
    - **Taille du domaine** (plus le domaine est petit, plus la variable est prioritaire),
    - **Historique des échecs** (augmentant le poids en cas d'échec).
- **Sélection des variables** :
  Les variables sont triées par priorité. La variable avec la plus haute priorité est sélectionnée en premier.

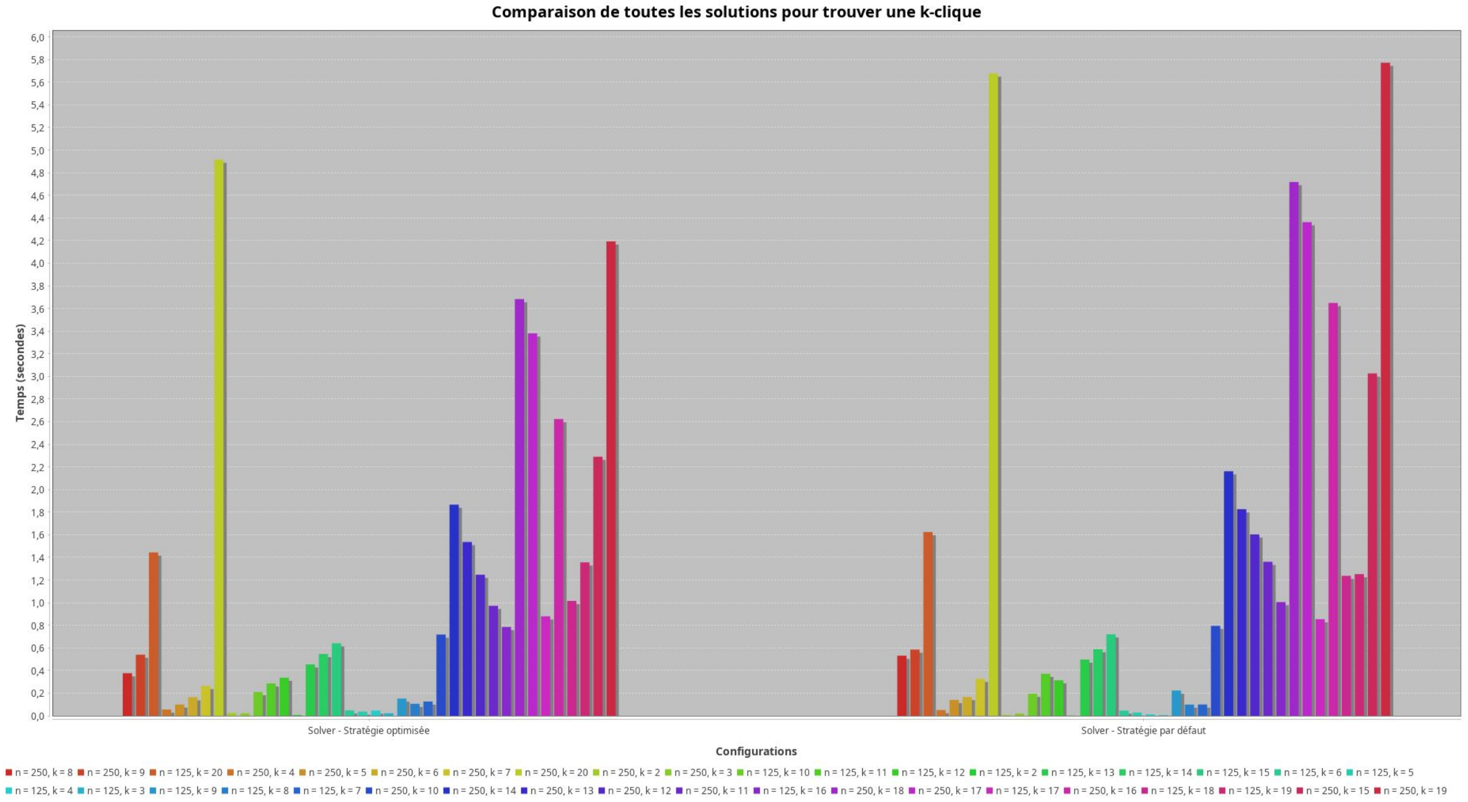
Méthodes complètes Choco-Solver
Complexité du problème



Méthodes complètes Choco-Solver Comparaison des méthodes



Méthodes complètes Choco-Solver
Comparaison des méthodes



Méthodes incomplètes

première méthode – trouver une k-clique

Algorithm 1 Recherche de clique par tri décroissant des degrés

Require: Graphe $G = (V, E)$ avec n sommets, taille de clique cible k

Ensure: Nombre d'essais pour trouver une clique de taille k

```
Initialisation : nbTry ← 0
for m = 1 to 10000 do
    clique ← ∅
    nodes ← Liste des sommets V
    Trier nodes par ordre décroissant de degrés
    if m > 1 then
        Mélanger aléatoirement nodes
    end if
    for i ∈ nodes do
        if taille de clique < k then
            canAdd ← true
            for node ∈ clique do
                if (node, i) ∉ E then
                    canAdd ← false
                    break
                end if
            end for
            if canAdd then
                Ajouter i à clique
            end if
        end if
    end for
    if taille de clique = k then
        Retourner m (nombre d'essais)
    end if
    nbTry ← m
end for
Retourner nbTry
```

Méthodes incomplètes

seconde méthode – trouver une k-clique

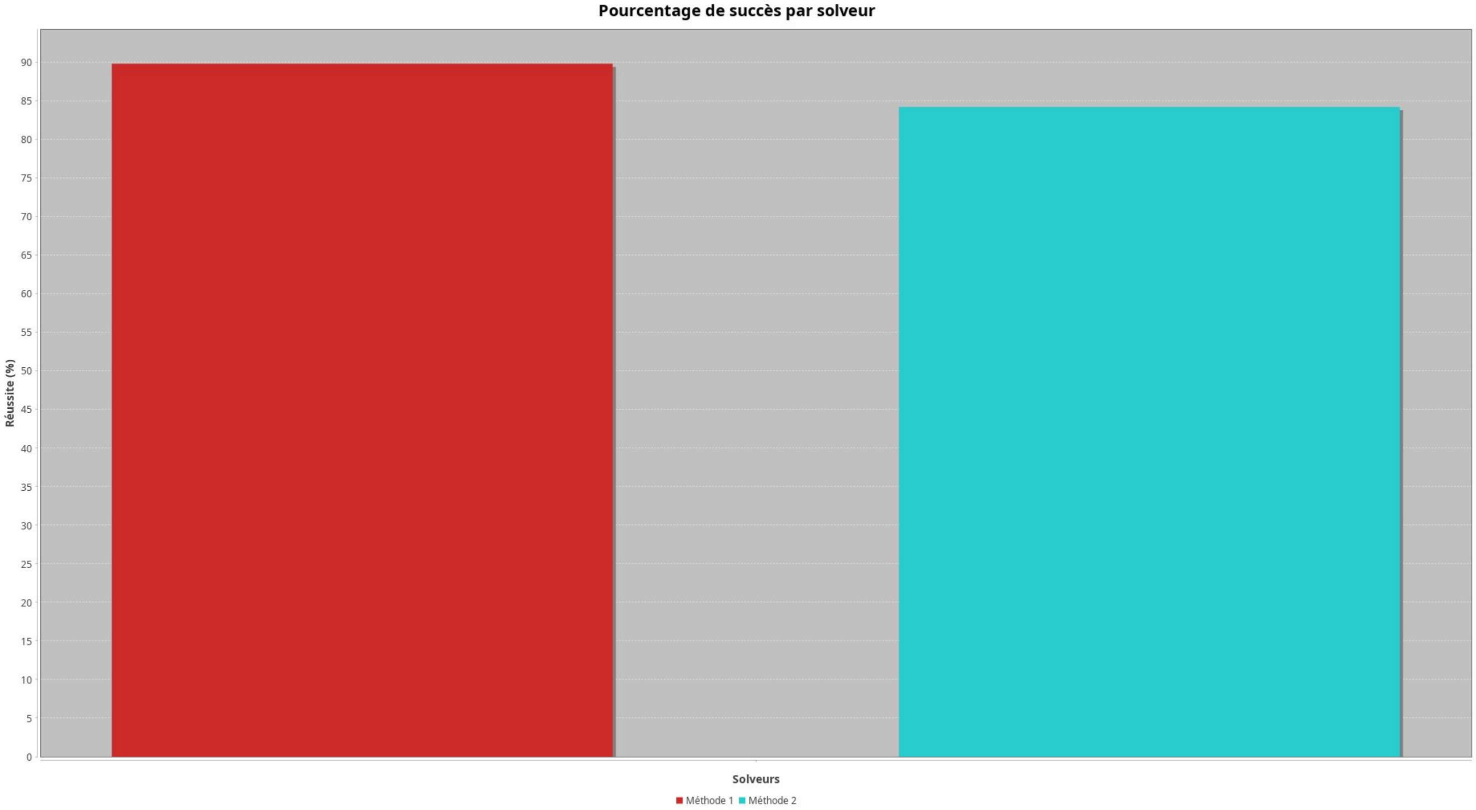
Algorithm 2 Recherche de clique par mélange aléatoire

Require: Graphe $G = (V, E)$ avec n sommets, taille de clique cible k

Ensure: Nombre d'essais pour trouver une clique de taille k

```
Initialisation : nbTry ← 0
for m = 1 to 10000 do
    clique ← ∅
    nodes ← Liste des sommets V
    Mélanger aléatoirement nodes
    for i ∈ nodes do
        canAdd ← true
        for node ∈ clique do
            if (node, i) ∉ E then
                canAdd ← false
                break
            end if
        end for
        if canAdd then
            Ajouter i à clique
            if taille de clique = k then
                Retourner m (nombre d'essais)
            end if
        end if
    end for
    nbTry ← m
end for
Retourner nbTry
```

Méthodes incomplètes - Comparaison pour une k-clique



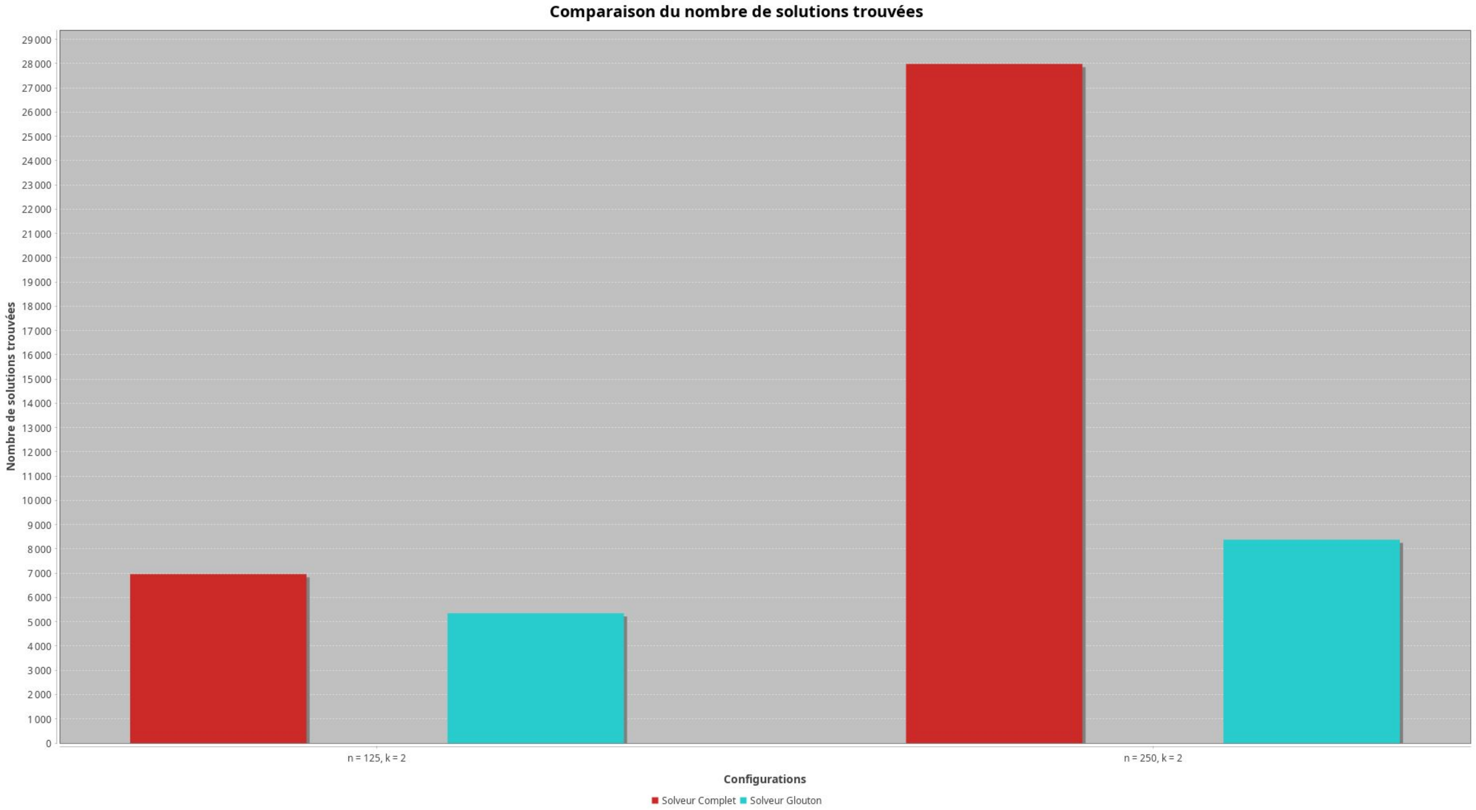
Méthodes incomplètes

Méthode pour trouver le plus de k-cliques (problème initial)

Algorithm 3 Recherche des k-cliques uniques (Version Modifiée)

Entrée : Graphe $G = (V, E)$, taille de clique k, nombre d'essais $M = 10000$
Sortie : Liste des k-cliques uniques
Initialiser la liste $allCliques \leftarrow \emptyset$
Trier les sommets V par degré décroissant
for $m \leftarrow 0$ à $M - 1$ **do**
    Initialiser $clique \leftarrow \emptyset$
    if $m > 0$ **then**
        Mélanger aléatoirement les sommets de V
    end if
    for chaque sommet i dans V **do**
        if i est connecté à tous les sommets dans $clique$ **then**
            Ajouter i à $clique$
            if $|clique| = k$ **then**
                if $clique$ n'est pas déjà dans $allCliques$ **then**
                    Ajouter $clique$ à $allCliques$
                end if
                break
            end if
        end if
    end for
end for
Retourner $allCliques$

Méthodes incomplètes - Comparaison sur le problème initial



Des questions ?