# Partition d'un graphe en sous-ensembles connexes et équilibrés

## Problème BCPk
## Formulations et Algorithmes

Bedis BACCAR Rayen ZARGUI

École Nationale Supérieure de Techniques Avancées
5OD21 - Optimisation Discrète

2025/2026

# Introduction : Le Problème BCPk

## Objectif

Diviser un graphe en $k$ sous-ensembles (classes) en respectant deux contraintes majeures :

1. **Connexité :** Chaque classe doit former un sous-graphe d'un seul tenant.
2. **Équilibre :** Les classes doivent avoir des poids (somme des poids des nœuds) similaires.

## Applications

- Traitement d'images (segmentation)
- Logistique (flottes de robots)
- Démarcation de zones (ex: patrouilles de police)

# Définition Formelle et Complexité

## Instance

- Un graphe connexe $G = (V, E)$.
- Une fonction de poids $w : V \to \mathbb{Q}_{\geq 0}$.
- Un entier $k \geq 2$.

## Objectif : Max-Min

Trouver une $k$-partition connexe $\{V_i\}_{i \in [k]}$ de $V$ qui maximise le poids de la classe la plus "légère".

$$\max \left( \min_{i \in [k]} \{w(V_i)\} \right)$$

## Complexité

Le problème BCPk est NP-difficile au sens fort.

# Approche 1 : Formulation Compacte par Flux ($F_k$)

## Principe

Forcer la connexité en modélisant $k$ arborescences de flux disjointes.

- On ajoute $k$ "sources" virtuelles $s_1, \dots, s_k$.
- Chaque source $s_i$ "alimente" sa classe $V_i$.
- Chaque nœud $v \in V$ "consomme" un flux égal à son poids $w(v)$.

## Contrainte Clé (Conservation du flux)

Pour chaque nœud $v \in V$ :

$$f(\text{entrant}) - f(\text{sortant}) = w(v)$$

## Caractéristiques

- **Taille :** Compacte (polynomiale).
- **Inconvénient :** Relaxation linéaire faible, très lente en pratique.

# Approche 2 : Formulation par Coupes ($C_k$)

## Principe : Branch-and-Cut

Utiliser des "Lazy Constraints" (contraintes paresseuses).

- Le modèle de base assigne juste les nœuds ($x_{v,i} \in \{0, 1\}$).
- On ne garantit pas la connexité au départ.
- **Séparation :** Si une solution n'est pas connexe, on ajoute une "coupe" pour l'interdire.

## Contraintes de Connexité (Ajoutées dynamiquement)

$$x_{u,i} + x_{v,i} - \sum_{z \in S} x_{z,i} \leq 1$$

(Si $u, v$ sont dans $i$, au moins un nœud $z$ de tout séparateur $S$ doit aussi y être.)

## Caractéristiques

- **Taille :** Contraintes à taille Exponentielle (en théorie).
- **Avantage :** Relaxation linéaire forte, très rapide en pratique.

# Approche 3 : Améliorations de l'Approche par Coupes

Objectif : Accélérer la convergence du Branch-and-Cut (Approche 2).

## 1. Inégalités de Couverture (Cover / Lifted Cover)

- **Idée :** Renforcer la relaxation linéaire en ajoutant des contraintes de type "sac à dos" sur les poids.
- Exploitent les contraintes d'équilibrage pour élaguer l'arbre.

## 2. Propagation de Domaine

- **Idée :** Analyser les solutions fractionnaires pendant la recherche.
- Si un nœud $u$ ne peut plus être connecté à une classe $i$, on force $x_{u,i} = 0$.
- Permet d'élaguer l'arbre plus tôt.

# Protocole Expérimental

## Environnement

- **Langage :** Julia (v1.9+)
- **Modélisation :** JuMP.jl
- **Solveur :** Gurobi
- **Limite de temps :** 120 secondes

## Instances Testées

- **Grille :** $5 \times 5$ $(n = 25, m = 40, k = 2)$
- **Aléatoire :** $(n = 20, m = 50, k = 4)$
- **Aléatoire Dense :** $(n = 20, m = 100, k = 5)$

# Résultats 1 : Grille $5 \times 5$ ($n = 25, k = 2$)

width=center

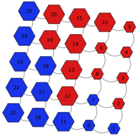
| Méthode | Type / Amélioration | Objectif | Temps (s) | Coupes |
|---|---|---|---|---|
| Q1-Flow (Flux) | Formulation compacte | 515.0 | **79.65** | 0 |
| Q2-Cut (Baseline) | Branch-and-Cut | 515.0 | 1.11 | 1042 |
| Q2 + Cover | Coupe de couverture | 515.0 | 0.15 | 292 |
| **Q2 + Lifted Cover** | **Coupe rehaussée** | **515.0** | **0.13** | **292** |
| Q2 + Propag | Propagation de domaine | 515.0 | 2.25 | 1985 |

## Analyse (Grille)

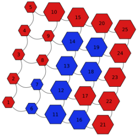
- L'approche Coupes (Q2) est 70x plus rapide que l'approche Flux (Q1).
- Les **Lifted Covers (Q3)** sont très efficaces sur cette grille : temps réduit de 88%.

# Visualisation des résultats (Grille 2x3)

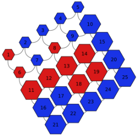
Q1-Flow Obj=515.0



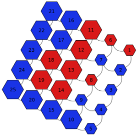
Q2-Cut+Connexe - Obj=515.0



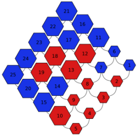
Q2-Cut+Cover+Connexe - Obj=515.0



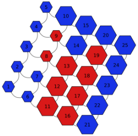
Q2-Cut+Lifted+Connexe - Obj=515.0



Q2-Cut+Propag+Connexe - Obj=515.0



Q2-Cut+Lifted+Propag+Connexe - Obj=515.0



# Résultats 2 : Aléatoire Dense ($n = 20, k = 5$)

width=center

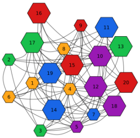
| Méthode | Type / Amélioration | Objectif | Temps (s) | Coupes |
|---|---|---|---|---|
| Q1-Flow (Flux) | Formulation compacte | 203.0 | 79.21 | 0 |
| **Q2-Cut (Baseline)** | **Branch-and-Cut** | **203.0** | **3.56** | **1448** |
| Q2 + Cover | Coupe de couverture | 203.0 | 9.42 | 2994 |
| Q2 + Lifted Cover | Coupe rehaussée | 203.0 | 8.47 | 2994 |
| Q2 + Propag | Propagation de domaine | 203.0 | 52.55 | 3970 |

## Analyse (Aléatoire Dense)

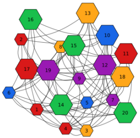
- L'approche Coupes (Q2) reste 20x plus rapide que Flux (Q1).
- **Effet inverse :** Toutes les améliorations (Q3) sont **contre-productives** et ralentissent la convergence.

# Visualisation (Aléatoire Dense, $k = 5$)

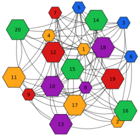
Q1-Flow Obj=200.0



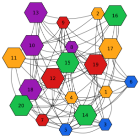
Q2-Cut+Connexe - Obj=203.0



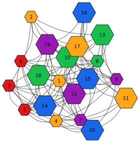
Q2-Cut+Cover+Connexe - Obj=203.0



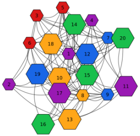
Q2-Cut+Lifted+Connexe - Obj=203.0



Q2-Cut+Propag+Connexe - Obj=203.0



Q2-Cut+Lifted+Propag+Connexe - Obj=203.0



# Conclusion Générale

## Comparaison des Formulations (Q1 vs Q2)

- **Q1 (Flux) :** Compacte mais **très coûteuse** (lente) en pratique.
- **Q2 (Coupes)** : Exponentielle (en théorie) mais **nettement plus performante** (B&C).

## Impact des Améliorations (Q3)

L'efficacité dépend fortement de la structure du graphe :

- **Efficace (+)** sur les **grilles** structurées.
- **Contre-productif (-)** sur les graphes **aléatoires denses**.

## Bilan

La performance dépend de la synergie entre la formulation et le solveur. Une approche par séparation dynamique (B&C) bien menée surpasse une formulation compacte.