# Improving branch-and-cut performance by random sampling

Matteo Fischetti[1], Andrea Lodi[2], Michele Monaci[1],
Domenico Salvagnin[1], Andrea Tramontani[3]

Présentation de l'article : Gauthier Gidel[4]

[1]DEI, University of Padova, Italy [2]DEI, University of Bologna, Italy

[3]CPLEX Optimization, Bologna, Italy [4]DIRO, Université de Montréal.

11 Avril 2017

Slides disponibles sur `www.di.ens.fr/~gidel`.

# Overview

1. Motivations
2. Contributions de l'article.
3. Variabilités des différents solvers.
4. L'algorithme `ksample`.
5. Son implémentation dans `Cplex`.
6. Conclusion.

# Motivations

Expérience avec `Cplex` sur un même dataset[1].
Dataset : `10teams` Version de `Cplex` : 11

[1] Danna Emilie. "Performance variability in mixed integer programing". In: *MIP 2008 workshop in New Work*. 2008.

# Motivations

Expérience avec `Cplex` sur un même dataset[1].
Dataset : `10teams` Version de `Cplex` : 11

- Première résolution de l'instance.
- Plateforme : `Linux`
- Temps **0.41 secs**. Itérations : **2731**. Nœuds : **0**

[1] Danna Emilie. "Performance variability in mixed integer programing". In: *MIP 2008 workshop in New Work*. 2008.

# Motivations

Expérience avec `Cplex` sur un même dataset[1].
Dataset : `10teams` Version de `Cplex` : 11

- Première résolution de l'instance.
- Plateforme : `Linux`
- Temps **0.41 secs**. Itérations : **2731**. Nœuds : **0**

- Seconde résolution de l'instance.
- Plateforme : `AIX`
- Temps **41.39 secs**. Itérations : **122948**. Nœuds : **1426**

[1] Danna Emilie. "Performance variability in mixed integer programing". In: *MIP 2008 workshop in New Work*. 2008.

# Motivations

**Même** algorithme. **Même** instance.
Facteur **100** pour le temps de résolution.
Facteur **50** pour le nombre d'itération.

# Motivations

**Même** algorithme. **Même** instance.
Facteur **100** pour le temps de résolution.
Facteur **50** pour le nombre d'itération.

Pourquoi le même algorithme sur les même données n'effectue pas le même nombre d'itération selon la plateforme ?

# Motivations

**Même** algorithme. **Même** instance.
Facteur **100** pour le temps de résolution.
Facteur **50** pour le nombre d'itération.

Pourquoi le même algorithme sur les même données n'effectue pas le même nombre d'itération selon la plateforme ?

- Choix aléatoires **différents** sur chaque plateforme.

# Causes de l'aléas dans le B & C

Des cas d'égalité apparaissent :

1. Égalité de scores : choix du branchement, choix du nœud après un retour en arrière,...

   Règle pour choisir parmi les égalités :

   - Choix "aléatoires".
   - Erreurs d'arrondis influencent ce choix.

# Causes de l'aléas dans le B & C

Des cas d'égalité apparaissent :

1. Égalité de scores : choix du branchement, choix du nœud après un retour en arrière,...

   Règle pour choisir parmi les égalités :

   - Choix "aléatoires".
   - Erreurs d'arrondis influencent ce choix.

2. Coupes : Choix de la variable pour la coupes de Gomory,...

# Causes de l'aléas dans le B & C

Des cas d'égalité apparaissent :

1. Égalité de scores : choix du branchement, choix du nœud après un retour en arrière,...

   Règle pour choisir parmi les égalités :
   - Choix "aléatoires".
   - Erreurs d'arrondis influencent ce choix.
2. Coupes : Choix de la variable pour la coupes de Gomory,...
3. Base optimale dégénérée des relaxations LP.

# Causes de l'aléas dans le B & C

Des cas d'égalité apparaissent :

1. Égalité de scores : choix du branchement, choix du nœud après un retour en arrière,...

   Règle pour choisir parmi les égalités :
   - Choix "aléatoires".
   - Erreurs d'arrondis influencent ce choix.
2. Coupes : Choix de la variable pour la coupes de Gomory,...
3. Base optimale dégénérée des relaxations LP.
4. **Première** (à la racine) base optimale **dégénérée**.

# Causes de l'aléas dans le B & C

Des cas d'égalité apparaissent :

1. Égalité de scores : choix du branchement, choix du nœud après un retour en arrière,...

   Règle pour choisir parmi les égalités :
   - Choix "aléatoires".
   - Erreurs d'arrondis influencent ce choix.
2. Coupes : Choix de la variable pour la coupes de Gomory,...
3. Base optimale dégénérée des relaxations LP.
4. **Première** (à la racine) base optimale **dégénérée**.

Paramètre "Random seed" non disponible à cette époque :

**Changement de plateforme** : moyen pour mettre en exergue cette variabilité !

# Première cause de l'aléas dans le B & C

**Dégénérescence :**

- Coin **dégénéré** : solution d'au moins deux ensembles **distincts** de contraintes d'inégalités.
- Base **dégénérée** : continent un coin dégénéré.

# Première cause de l'aléas dans le B & C

**Dégénérescence :**

- Coin **dégénéré** : solution d'au moins deux ensembles **distincts** de contraintes d'inégalités.
- Base **dégénérée** : continent un coin dégénéré.

De manière équivalente :

- Base **dégénérée** : une des variable de base est fixée à 0 dans la solution de base.

Plupart des programmes linéaires rencontrés en pratique sont dégénérés (dont celui de la **racine**).

# Cause de l'aléas dans le B & C

Illustration de la dégénérescence : $(x_3, x_4)$ et $(x_2, x_3)$ sont deux base optimales possibles.

| $z$ | $x_1$ | $x_2$ | $\boldsymbol{x_3}$ | $\boldsymbol{x_4}$ | |
|---|---|---|---|---|---|
| 1 | 3 | -2 | **0** | **0** | -2 |
| 0 | -3 | -3 | **1** | **0** | 2 |
| 0 | -1 | 2 | **0** | **1** | 0 |

$\rightarrow$

# Cause de l'aléas dans le B & C

Illustration de la dégénérescence : $(x_3, x_4)$ et $(x_2, x_3)$ sont deux base optimales possibles.

| $z$ | $x_1$ | $x_2$ | $\boldsymbol{x_3}$ | $\boldsymbol{x_4}$ | |
|---|---|---|---|---|---|
| 1 | 3 | -2 | **0** | **0** | -2 |
| 0 | -3 | -3 | **1** | **0** | 2 |
| 0 | -1 | 2 | **0** | **1** | 0 |

$\rightarrow$

| $z$ | $x_1$ | $\boldsymbol{x_2}$ | $\boldsymbol{x_3}$ | $x_4$ | |
|---|---|---|---|---|---|
| 1 | 3 | **0** | **0** | 1 | -2 |
| 0 | -3 | **0** | **1** | 2 | 2 |
| 0 | -1 | **1** | **0** | 1/2 | 0 |

**Conclusion.** Dégénérescence : grande cause de **variabilité**.

# Contribution

Contributions de l'article :

- Étude de la variabilité de performance des différents solvers. (`Cplex`, `Gurobi` et `XPRESS`)

# Contribution

Contributions de l'article :

- Étude de la variabilité de performance des différents solvers. (`Cplex`, `Gurobi` et `XPRESS`)
- Création d'un Algorithme, exploitant cette variabilité :

# Contribution

Contributions de l'article :

- Étude de la variabilité de performance des différents solvers. (`Cplex`, `Gurobi` et `XPRESS`)
- Création d'un Algorithme, exploitant cette variabilité :
  - Optimise $K$ fois la racine en parallèle.

# Contribution

Contributions de l'article :

- Étude de la variabilité de performance des différents solvers. (`Cplex`, `Gurobi` et `XPRESS`)
- Création d'un Algorithme, exploitant cette variabilité :
  - Optimise $K$ fois la racine en parallèle.
  - Utilise une base optimale initiale différente.

# Contribution

Contributions de l'article :

- Étude de la variabilité de performance des différents solvers. (`Cplex`, `Gurobi` et `XPRESS`)
- Création d'un Algorithme, exploitant cette variabilité :
  - Optimise $K$ fois la racine en parallèle.
  - Utilise une base optimale initiale différente.
  - Rassemble les informations obtenues par chaque résolution.

# Contribution

Contributions de l'article :

- Étude de la variabilité de performance des différents solvers. (`Cplex`, `Gurobi` et `XPRESS`)
- Création d'un Algorithme, exploitant cette variabilité :
  - Optimise $K$ fois la racine en parallèle.
  - Utilise une base optimale initiale différente.
  - Rassemble les informations obtenues par chaque résolution.
  - Cet algorithme est appelé `ksample`.

# Étude la variabilité de performance des solvers

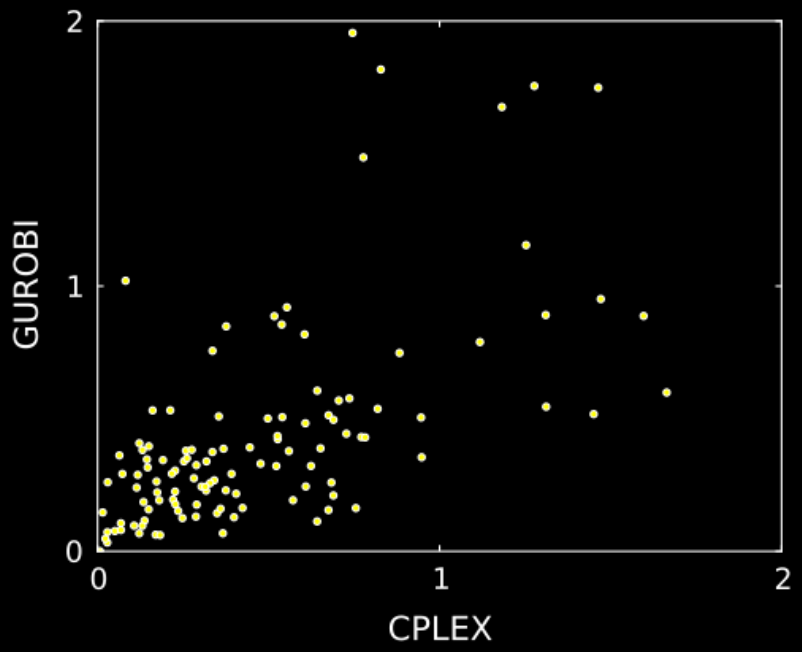
**Génération aléatoire :** Permutation aléatoire des lignes et des colonnes du modèle original.

# Étude la variabilité de performance des solvers

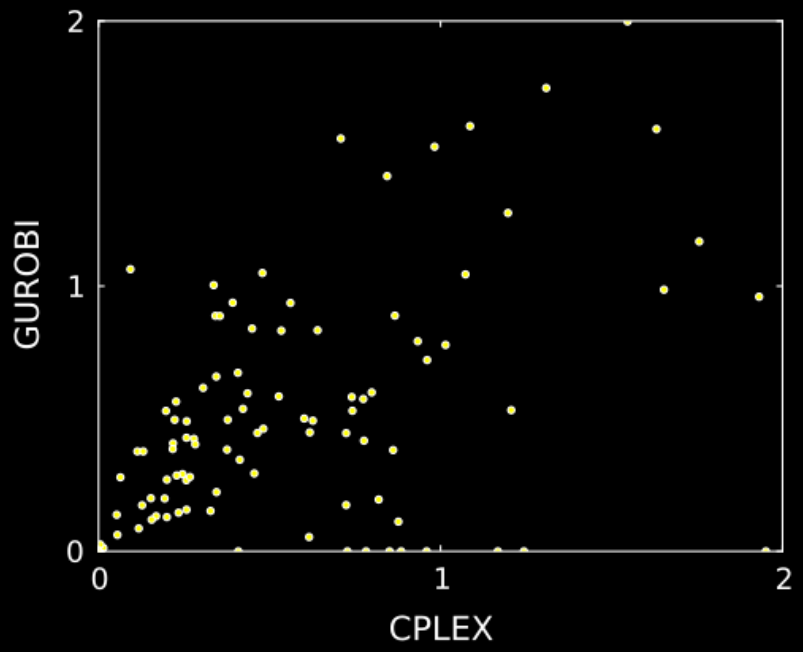
**Génération aléatoire :** Permutation aléatoire des lignes et des colonnes du modèle original.

Rapport déviation standard sur moyenne arithmétique :

**temps**




**nb de nœuds**

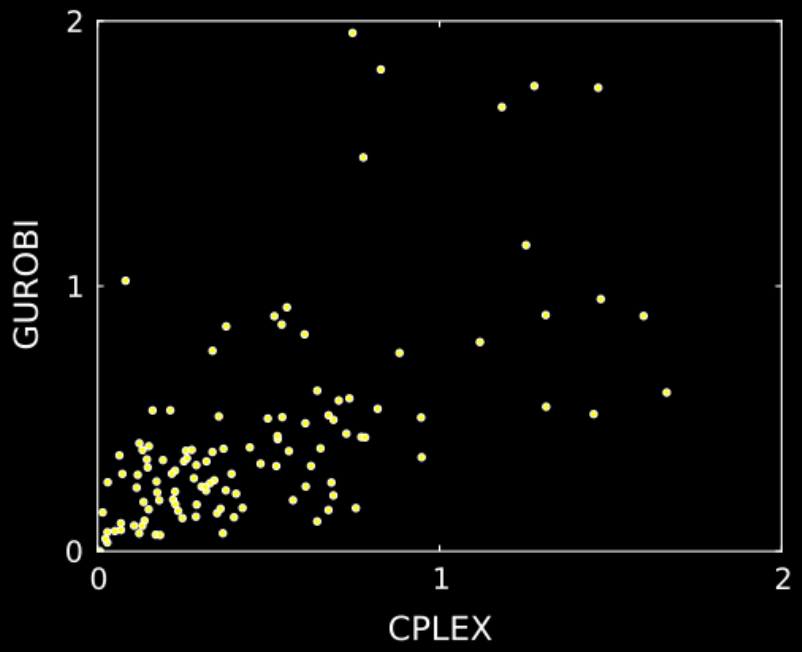


# Étude la variabilité de performance des solvers

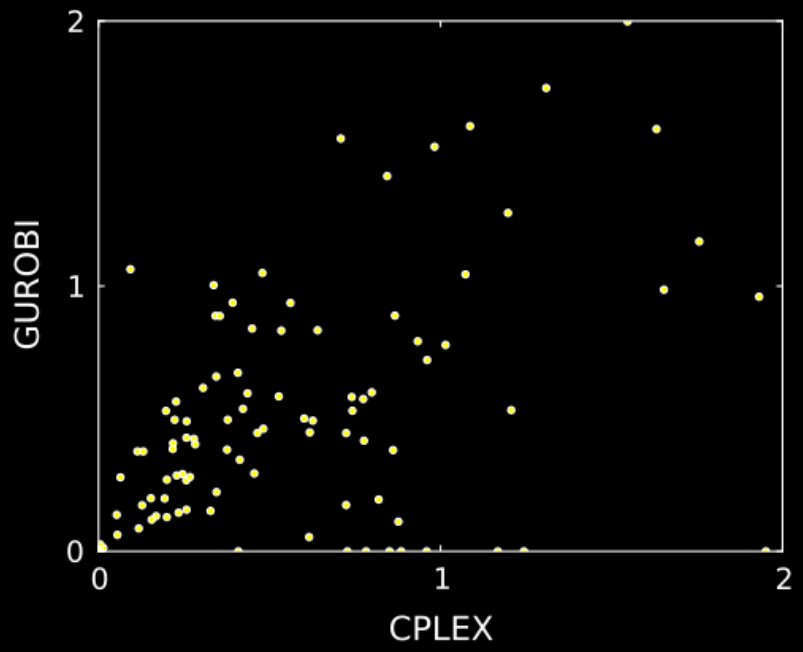
**Génération aléatoire :** Permutation aléatoire des lignes et des colonnes du modèle original.

Rapport déviation standard sur moyenne arithmétique :

**temps**




**nb de nœuds**




Variabilité fortement corrélée avec :

- Difficulté du problème.
- Prétraitement.

# Technique d'échantillonnage pratique

- Simuler la permutation des lignes et des colonnes.

[2]Danna Emilie. "Performance variability in mixed integer programing". In: *MIP 2008 workshop in New Work*. 2008.

# Technique d'échantillonnage pratique

- Simuler la permutation des lignes et des colonnes.
- Paramètre "random seed" rendu disponible après cet article.

[2]Danna Emilie. "Performance variability in mixed integer programing". In: *MIP 2008 workshop in New Work*. 2008.

# Technique d'échantillonnage pratique

- Simuler la permutation des lignes et des colonnes.
- Paramètre "random seed" rendu disponible après cet article.
- Introduit un aléa "neutre" n'influençant pas le processus d'optimisation.[2]

[2] Danna Emilie. "Performance variability in mixed integer programing". In: *MIP 2008 workshop in New Work.* 2008.

# L'Algorithme `ksample`

**Entrée :** une instance `I` de MIP.

# L'Algorithme `ksample`

**Entrée :** une instance `I` de MIP.

1. Prétraite l'instance `I`.

# L'Algorithme `ksample`

**Entrée :** une instance `I` de MIP.

1. Prétraite l'instance `I`.
2. Calcule les bases optimales de la relaxation `LP` initiale.

# L'Algorithme `ksample`

**Entrée :** une instance `I` de MIP.

1. Prétraite l'instance `I`.
2. Calcule les bases optimales de la relaxation `LP` initiale.
3. **for** $i = 1 \dots K - 1$:
   - Choisit aléatoirement une base optimale $B_i$ de `LP`.

# L'Algorithme `ksample`

**Entrée :** une instance `I` de MIP.

1. Prétraite l'instance `I`.
2. Calcule les bases optimales de la relaxation `LP` initiale.
3. **for** $i = 1 \dots K - 1$:
   - Choisit aléatoirement une base optimale $B_i$ de `LP`.
   - Exécute les méthodes de coupes à la racine.

# L'Algorithme `ksample`

**Entrée :** une instance `I` de MIP.

1. Prétraite l'instance `I`.
2. Calcule les bases optimales de la relaxation `LP` initiale.
3. **for** $i = 1 \dots K - 1$:
   - Choisit aléatoirement une base optimale $B_i$ de `LP`.
   - Exécute les méthodes de coupes à la racine.
   - Enregistre les coupes $P_i$ et les solutions réalisables $S_i$.

# L'Algorithme `ksample`

**Entrée :** une instance `I` de MIP.

1. Prétraite l'instance `I`.
2. Calcule les bases optimales de la relaxation `LP` initiale.
3. **for** $i = 1 \dots K-1$:
   - Choisit aléatoirement une base optimale $B_i$ de `LP`.
   - Exécute les méthodes de coupes à la racine.
   - Enregistre les coupes $P_i$ et les solutions réalisables $S_i$.
4. Résout `I` utilisant les ensembles :

$$S := \cup_{i=1}^{K-1} S_i \qquad \text{et} \qquad P := \cup_{i=1}^{K-1} P_i.$$

# L'Algorithme `ksample`

**Entrée :** une instance `I` de MIP.

1. Prétraite l'instance `I`.
2. Calcule les bases optimales de la relaxation `LP` initiale.
3. **for** $i = 1 \dots K-1$:
   - Choisit aléatoirement une base optimale $B_i$ de `LP`.
   - Exécute les méthodes de coupes à la racine.
   - Enregistre les coupes $P_i$ et les solutions réalisables $S_i$.
4. Résout `I` utilisant les ensembles :

$$S := \cup_{i=1}^{K-1} S_i \qquad \text{et} \qquad P := \cup_{i=1}^{K-1} P_i.$$

**Sortie :** Une solution optimale de `I`.

# Réduire la variabilité en restant compétitif

Avec une grande valeur pour $K$ :

- Variabilité des performances plus faible.

# Réduire la variabilité en restant compétitif

Avec une grande valeur pour $K$ :

- Variabilité des performances plus faible.
- Certaines instances sont résolues à la racines.

# Réduire la variabilité en restant compétitif

Avec une grande valeur pour $K$ :

- Variabilité des performances plus faible.
- Certaines instances sont résolues à la racines.
- Nombre de nœuds produits largement inférieur.

# Réduire la variabilité en restant compétitif

Avec une grande valeur pour $K$ :

- Variabilité des performances plus faible.
- Certaines instances sont résolues à la racines.
- Nombre de nœuds produits largement inférieur.
- Temps de calculs plus long dû à la synchronisation nécessaire de toutes les optimisation parallèles.

# Réduire la variabilité en restant compétitif

Avec une grande valeur pour $K$ :

- Variabilité des performances plus faible.
- Certaines instances sont résolues à la racines.
- Nombre de nœuds produits largement inférieur.
- Temps de calculs plus long dû à la synchronisation nécessaire de toutes les optimisation parallèles.
- Impossible pour `ksample` de paralléliser certain calculs d'une même optimisation.

# Réduire la variabilité en restant compétitif

Avec une grande valeur pour $K$ :

- Variabilité des performances plus faible.
- Certaines instances sont résolues à la racines.
- Nombre de nœuds produits largement inférieur.
- Temps de calculs plus long dû à la synchronisation nécessaire de toutes les optimisation parallèles.
- Impossible pour `ksample` de paralléliser certain calculs d'une même optimisation.

Un compromis peut être trouvé avec une petite valeur pour $K$.

# Résultats concernant la réduction de variabilité

Comparaison à la racine. (`cpxdef` : Cplex standard)

| Testbed | Instance | igap | | pgap | | dgap | |
|---|---|---|---|---|---|---|---|
| | | Avg (%) | St.dev (%) | Avg (%) | St.dev (%) | Avg (%) | St.dev (%) |
| Benchmark | `cpxdef` | 53.80 | 5.66 | 45.20 | 6.35 | 22.32 | 0.89 |
| | `ksample` | 37.74 | 4.63 | 25.88 | 6.49 | 21.61 | 0.84 |
| Primal | `cpxdef` | 63.37 | 10.05 | 63.37 | 10.05 | 0.00 | 0.00 |
| | `ksample` | 31.79 | 3.28 | 31.79 | 3.28 | 0.00 | 0.00 |
| All | `cpxdef` | 56.91 | 7.08 | 51.10 | 7.55 | 15.07 | 0.60 |
| | `ksample` | 35.81 | 4.19 | 27.80 | 5.45 | 14.59 | 0.57 |

- 10 répétions de `cpxdef` et `ksample` avec $K = 10$.
- Nette amélioration des performances.

# Résultats concernant le gain d'efficacité

Comparaison pour le B&C complet. (`cpxdef` : Cplex standard)

| Seed | Method | All | | Opt | | |
|---|---|---|---|---|---|---|
| | | Solved | Det.time | Solved | Det.time | Nodes |
| 1 | `cpxdef` | 107 | 172,584 | 104 | 109,569 | 2175 |
| | `ksample` | 108 | 151,004 | 104 | 96,370 | 1765 |
| 2 | `cpxdef` | 105 | 172,022 | 102 | 102,103 | 1871 |
| | `ksample` | 104 | 153,985 | 102 | 85,571 | 1296 |
| 3 | `cpxdef` | 107 | 170,467 | 104 | 104,152 | 2171 |
| | `ksample` | 106 | 142,778 | 104 | 84,352 | 1581 |
| 4 | `cpxdef` | 106 | 182,415 | 106 | 118,001 | 2314 |
| | `ksample` | 109 | 154,826 | 106 | 104,730 | 1816 |
| 5 | `cpxdef` | 105 | 199,470 | 104 | 119,466 | 2415 |
| | `ksample` | 111 | 148,777 | 104 | 100,566 | 1780 |

- Résultats plus mitigés.
- Meilleures performance mais beaucoup de variabilité.

# Implementation pour `Cplex`

- Le choix a été fait de $K = 2$.

# Implementation pour `Cplex`

- Le choix a été fait de $K = 2$.
- Permet de limiter le coût général de l'opération.

# Implementation pour `Cplex`

- Le choix a été fait de $K = 2$.
- Permet de limiter le coût général de l'opération.
- Permet néanmoins une meilleure exploration des coupes et des solutions réalisables.
- $K > 2$ donne un moins bon rapport coût/exploration.

# Implementation pour `Cplex`

- Le choix a été fait de $K = 2$.
- Permet de limiter le coût général de l'opération.
- Permet néanmoins une meilleure exploration des coupes et des solutions réalisables.
- $K > 2$ donne un moins bon rapport coût/exploration.
- Permet de synchroniser facilement les coupes et les solutions réalisables **au cours** de l'exploration.

# Résultats

Performances sur `IBM ILOG Cplex 12.5.1`.

- Tests sur $\sim 3200$ problemes.
- Limite de temps $10,000$s. Limite de taille pour l'arbre 6GB.

[3] Tobias Achterberg et al. "Constraint Integer Programming: A New Approach to Integrate CP and MIP". . In: *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Prblems*. 2008.

# Résultats

Performances sur `IBM ILOG Cplex 12.5.1`.

- Tests sur $\sim 3200$ problemes.
- Limite de temps $10,000$s. Limite de taille pour l'arbre 6GB.
- *Time* : ratio des moyennes géométriques baisées :

$$\bar{t}_n := \left( \prod_{k=1}^{n} (t_i + s) \right)^{1/n} - s$$

- *Nodes* : idem mais pour les nombre de nœuds.
- Moyenne géométrique biaisée réduit l'influence des outliers.[3].

[3] Tobias Achterberg et al. "Constraint Integer Programming: A New Approach to Integrate CP and MIP". . In: *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Prblems*. 2008.

# Résultats

Performances sur `IBM ILOG Cplex 12.5.1`.

| Class | All models | | | | Affected | | |
|---|---|---|---|---|---|---|---|
| | #models | #tilim | Time | Nodes | #models | Time | Nodes |
| All | 3157 | 84/82 | 0.99 | 0.96 | 1312 | 0.98 | 0.92 |
| [0,10k] | 3086 | 13/11 | 0.99 | 0.96 | 1312 | 0.98 | 0.92 |
| [1,10k] | 1870 | 13/11 | 0.98 | 0.97 | 1090 | 0.97 | 0.94 |
| [10,10k] | 1092 | 13/11 | 0.97 | 0.96 | 678 | 0.96 | 0.94 |
| [100,10k] | 559 | 13/11 | 0.95 | 0.92 | 367 | 0.92 | 0.88 |
| [1k,10k] | 219 | 13/11 | 0.90 | 0.86 | 154 | 0.86 | 0.81 |

- L'ensemble $[n, 10k]$ contient les instances qui ont pris au moins $n$ secs.
- *Affected* : modèles ayant menés à des calculs différents.

# Conclusion

- Haute sensibilité aux conditions initiale pour les solvers comme `Cplex` $\rightarrow$ grande variabilité dans les résultats.
- La première source de variabilité : dégénérescence de la relaxation continue de la racine.

# Conclusion

- Haute sensibilité aux conditions initiale pour les solvers comme `Cplex` $\rightarrow$ grande variabilité dans les résultats.
- La première source de variabilité : dégénérescence de la relaxation continue de la racine.
- Implémentation dans `Cplex` basée sur deux calculs effectués en parallèle sur la racine avec des choix aléatoires différents.
- Réduction du temps de calcul et du nombre de nœuds générés $\rightarrow$ par défaut depuis `Cplex 12.5.1`.

# Merci !

Slides disponibles sur `www.di.ens.fr/~gidel`.