Comparaison des partitions par la distance de transfert pour la classification de solutions en optimisation —applications à la coloration de graphes—

Daniel Porumbel*

Université d'Artois, LGI2A, Béthune (PRES Lille Nord de France)

*en collaboration avec Jin-Kao Hao (LERIA, Angers) et Pascale Kuntz (LINA, Nantes)

Le couplage entre Classification et Optimisation

- L'optimisation pour la classification: technique souvent utilisée
  - De nombreux problèmes de classification peuvent se modéliser comme des problèmes d'optimisation
  - Recherche d'une isométrie entre une dissimilarité initiale et une métrique associée à une structure classificatoire

→ Notre objectif: la classification pour l'optimisation combinatoire

- Un grand corpus de données: l'espace de recherche – les solutions candidates pour un problème d'optimisation
- Un vrai défi: la classification des solutions candidates de bonne qualité

Le couplage entre Classification et Optimisation

- L'optimisation pour la classification: technique souvent utilisée
  - De nombreux problèmes de classification peuvent se modéliser comme des problèmes d'optimisation
  - Recherche d'une isométrie entre une dissimilarité initiale et une métrique associée à une structure classificatoire

→ Notre objectif: la classification pour l'optimisation combinatoire

  - Un grand corpus de données: l'espace de recherche – les solutions candidates pour un problème d'optimisation
  - Un vrai défi: la classification des solutions candidates de bonne qualité

Apport de classification en optimisation

La classification et l'apprentissage offrent plusieurs outils en optimisation

1. meilleure compréhension de l'espace de recherche pour: identifier des clusters de solutions de qualité, comprendre leur distribution spatiale, leur densité, etc.
2. construction automatiques de nouveaux opérateurs (e.g., croisement)
3. apprendre de nouvelles fonctions d'évaluation – mieux approximer la distance vers un optimum global
4. sélection automatique d'heuristiques et des meilleurs paramètres
   - paramétrage automatique via apprentissage par renforcement ou processus de décision markoviens

Le cas des heuristiques

Algorithmes heuristiques : utilisées souvent pour les problèmes NP-durs

- l'espace de recherche : un nombre exponentiel de solutions
  - Le nombre de colorations possibles d'un graphe,
  - Le nombre de tours possibles pour le problème du voyageur de commerce
- Les approches exactes sont souvent limitées par la taille de l'espace de recherche

$$\Downarrow$$

Une heuristique cherche uniquement certaines zones de l'espace

- Exemple: une recherche locale passe d'une solution candidate à une solution voisine en maximisant un critère de qualité (fonction objectif)
- **Objectif heuristique**: cibler la recherche vers les zones les plus "riches" en solutions de qualité

Le cas des heuristiques

Algorithmes heuristiques : utilisées souvent pour les problèmes NP-durs

- l'espace de recherche : un nombre exponentiel de solutions
  - Le nombre de colorations possibles d'un graphe,
  - Le nombre de tours possibles pour le problème du voyageur de commerce
- Les approches exactes sont souvent limitées par la taille de l'espace de recherche

$$\Downarrow$$

Une heuristique cherche uniquement certaines zones de l'espace

- Exemple: une recherche locale passe d'une solution candidate à une solution voisine en maximisant un critère de qualité (fonction objectif)
- **Objectif heuristique**: cibler la recherche vers les zones les plus "riches" en solutions de qualité

Classification pour l'optimisation

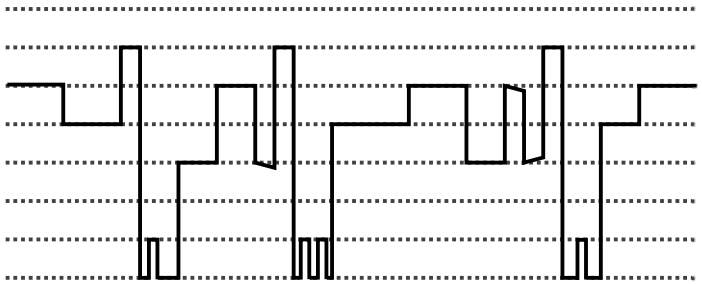
Les heuristiques sont efficaces mais leurs performances dépendent de routines et paramètres souvent déterminés de façon ad hoc.

- Une recherche locale a été comparée avec *"recherche du mont Everest dans un brouillard épais tout en souffrant de l'amnésie"*

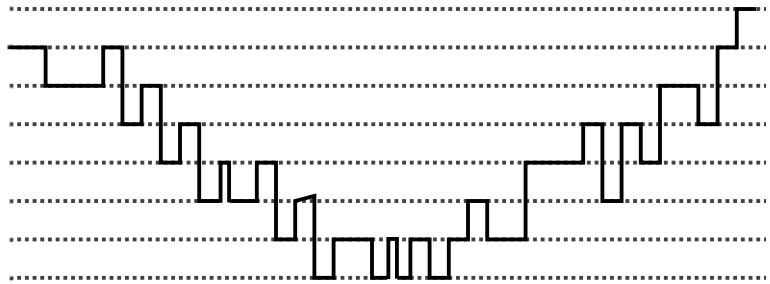
  [Russell et Norvig, 2002, Artificial Intelligence: A Modern Approach]

- L'espace de recherche peut être différent d'un problème à l'autre, ou même d'une instance à l'autre:

Plusieurs clusters avec solutions de qualité



Une "grande vallée" de solutions de qualité



- La classification et l'apprentissage seront utilisés pour découvrir ce type de structures afin de rendre leur exploration plus efficace

Contraintes de la classification en optimisation

Les heuristiques sont habituellement des algorithmes très rapides

- Pour la coloration de graphe, il est possible de visiter quelques millions de configurations par minute
- Volumes de données à traiter très importants

L'intégration de toute information apprise ne doit pas ralentir significativement le processus de recherche

Plan de la présentation

- Analyse de l'espace de recherche du problème de coloration de graphe
  - Classification de solutions de bonne qualité
  - On obtient l'hypothèse de clusterisation en utilisant une métrique:
    la distance de transfert

- Application à la conception d'algorithmes heuristiques
  - L'hypothèse de clustérisation nous a conduit à de meilleures méthodes d'exploration de l'espace

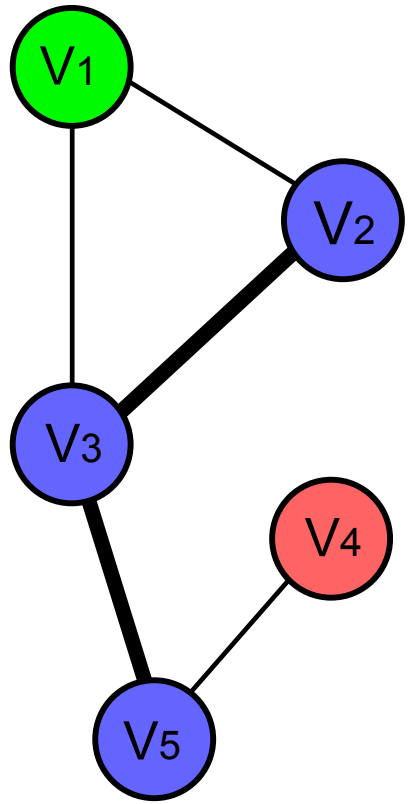
---

- Calcul linéaire de la distance de transfert entre partitions
  - La distance de transfert est utile non-seulement pour la coloration; elle apparaît assez souvent en classification.

Espace de recherche pour la k-coloration

Le problème de la k-coloration de graphe (pour $G(V, E)$ et k):

- Trouver une coloration (k couleurs) avec le minimum d'arêtes avec les extrémités de la même couleur
  - L'espace de recherche Ω: les $|V|^k$ colorations possibles (solutions candidates)
  - Fonction objectif f: le nombre de conflits (d'arêtes avec les deux extrémités de même couleur)
  - Fonction de voisinage $N : \Omega \longrightarrow 2^{\Omega}$
    $C' \in N(C)$ s'il est possible d'obtenir C' avec un seul changement de couleur sur C
  - Une coloration: une partition de V en classes/couleurs
  - Distance de transfert entre deux partitions: le nombre minimal de changements de classes nécessaires pour arriver d'une partition à l'autre

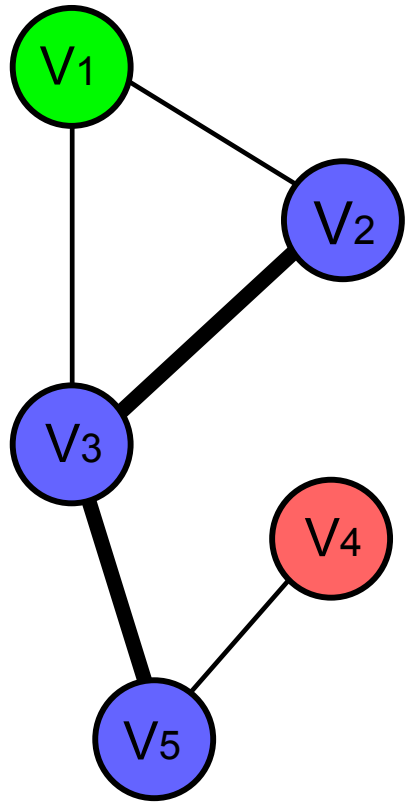


|V1|V2,V3,V5 |V4|

Espace de recherche pour la k-coloration

Le problème de la k-coloration de graphe (pour $G(V, E)$ et k):

- Trouver une coloration (k couleurs) avec le minimum d'arêtes avec les extrémités de la même couleur
  - L'espace de recherche Ω: les $|V|^k$ colorations possibles (solutions candidates)
  - Fonction objectif f: le nombre de conflits (d'arêtes avec les deux extrémités de même couleur)
  - Fonction de voisinage $N : \Omega \longrightarrow 2^{\Omega}$
    $C' \in N(C)$ s'il est possible d'obtenir C' avec un seul changement de couleur sur C
  - Une coloration: une partition de V en classes/couleurs
  - Distance de transfert entre deux partitions: le nombre minimal de changements de classes nécessaires pour arriver d'une partition à l'autre



|V1|V2,V3,V5 |V4|

Étude de l'espace de recherche

Questions intéressantes pour la classification

- Quelle est la distribution des solutions de bonne qualité ?
- Les recherches locales, sont-elles toujours attirées vers certaines zones de l'espace?
- Quelle pourrait être la trajectoire d'un processus de recherche?

Ces questions peuvent être approchées en utilisant des techniques de classification basées par exemple sur la distance de transfert.

Étude de l'espace de recherche

Questions intéressantes pour la classification

- Quelle est la distribution des solutions de bonne qualité ?
- Les recherches locales, sont-elles toujours attirées vers certaines zones de l'espace?
- Quelle pourrait être la trajectoire d'un processus de recherche?

Ces questions peuvent être approchées en utilisant des techniques de classification basées par exemple sur la distance de transfert.

Étude de l'espace de recherche

Questions intéressantes pour la classification

- Quelle est la distribution des solutions de bonne qualité ?
- Les recherches locales, sont-elles toujours attirées vers certaines zones de l'espace?
- Quelle pourrait être la trajectoire d'un processus de recherche?

Ces questions peuvent être approchées en utilisant des techniques de classification basées par exemple sur la distance de transfert.

Étude de l'espace de recherche

Questions intéressantes pour la classification

- Quelle est la distribution des solutions de bonne qualité ?
- Les recherches locales, sont-elles toujours attirées vers certaines zones de l'espace?
- Quelle pourrait être la trajectoire d'un processus de recherche?

Ces questions peuvent être approchées en utilisant des techniques de classification basées par exemple sur la distance de transfert.

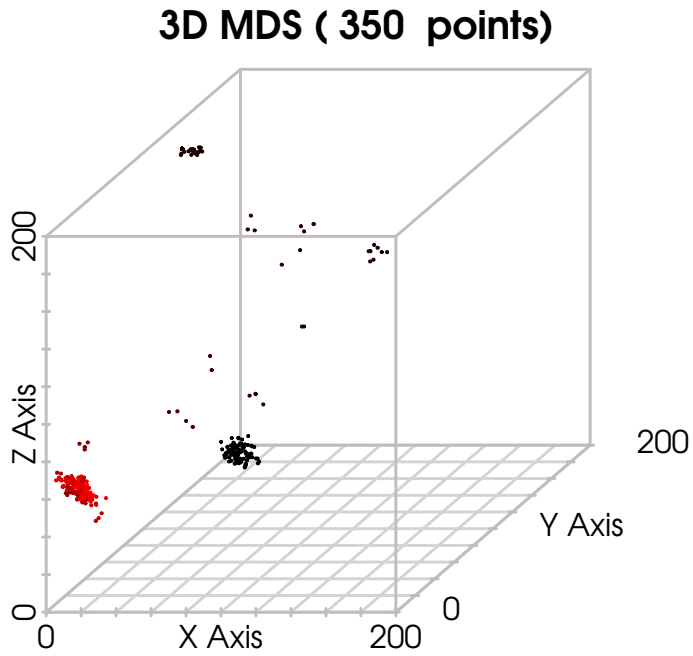
Distribution spatiale des solutions de bonne qualité

Représentation Multidimensional Scaling(MDS)

- Projection: *espace n-dimensionnel* $\longrightarrow$ *Espace 2D/3D*
- Distance Euclidienne entre les points 3D = approximation de la distance réelle entre les colorations associées
- Objectif MDS : minimiser la distorsion de la représentation (stress)
  - De nombreuses approches existent dans la communauté de classification
  - Nous évaluons nos représentations avec le stress de Kruskal

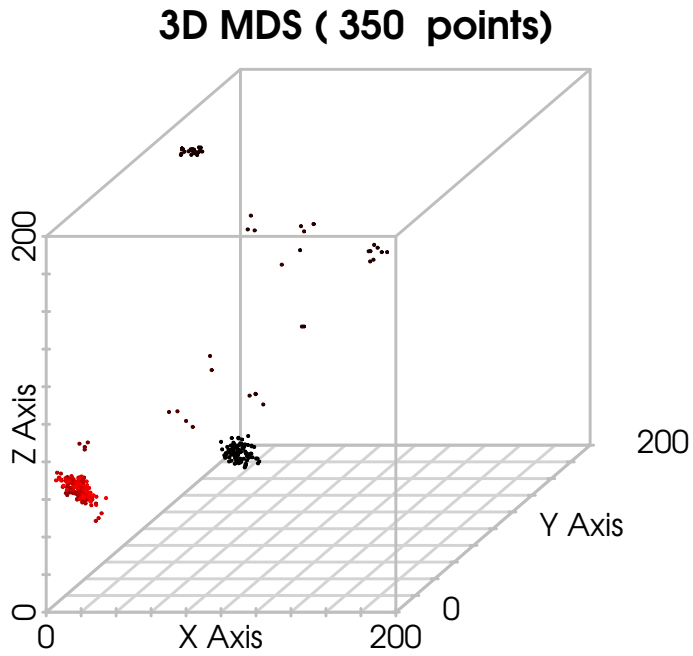
Représentation intuitive des solutions de qualité

- Représentation MDS de 350 solutions de bonne qualité trouvées par des recherches indépendantes
  - $G = dsjc250.5$, $k = 27$
- Ces points forment des clusters qui peuvent être couverts par des sphères de petit rayon
- Les points présentés sont en fait des optima locaux difficiles à atteindre
  - chaque point : plusieurs journées avec l'algorithme Tabucol (une recherche locale Tabou très populaire pour la coloration)



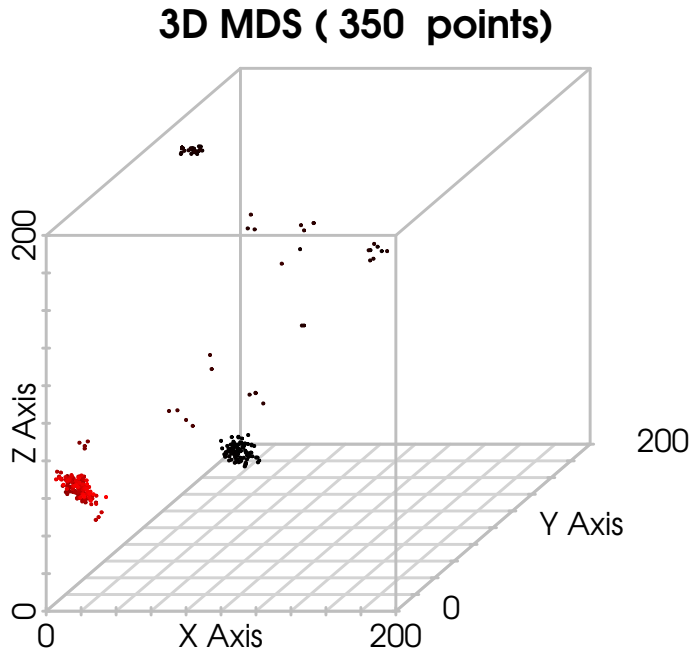
Représentation intuitive des solutions de qualité

- Représentation MDS de 350 solutions de bonne qualité trouvées par des recherches indépendantes
  - $G = dsjc250.5$, $k = 27$
- Ces points forment des clusters qui peuvent être couverts par des sphères de petit rayon
- Les points présentés sont en fait des optima locaux difficiles à atteindre
  - chaque point : plusieurs journées avec l'algorithme Tabucol (une recherche locale Tabou très populaire pour la coloration)



Représentation intuitive des solutions de qualité

- Représentation MDS de 350 solutions de bonne qualité trouvées par des recherches indépendantes
  - $G = dsjc250.5$, $k = 27$
- Ces points forment des clusters qui peuvent être couverts par des sphères de petit rayon
- Les points présentés sont en fait des optima locaux difficiles à atteindre
  - chaque point : plusieurs journées avec l'algorithme Tabucol (une recherche locale Tabou très populaire pour la coloration)



La trajectoire de la Recherche Tabou: représentation MDS

Soit un processus de recherche lancé à partir d'un optimum local

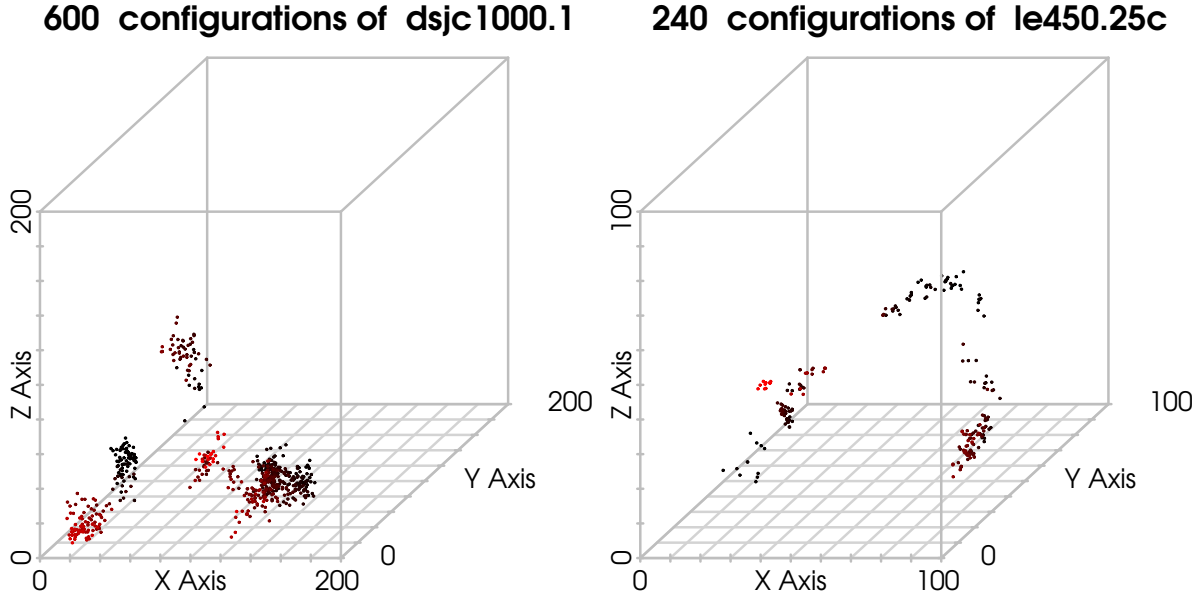
- on étudie la distribution des solutions de qualité

- Intuition: ces solutions de qualité sont regroupées dans des clusters
- Trajectoire étudiée sur une période courte, < 1000 solutions

La trajectoire de la Recherche Tabou: représentation MDS

Soit un processus de recherche lancé à partir d'un optimum local

- on étudie la distribution des solutions de qualité



- Intuition: ces solutions de qualité sont regroupées dans des clusters
- Trajectoire étudiée sur une période courte, < 1000 solutions

Étude de la trajectoire sur une longue période

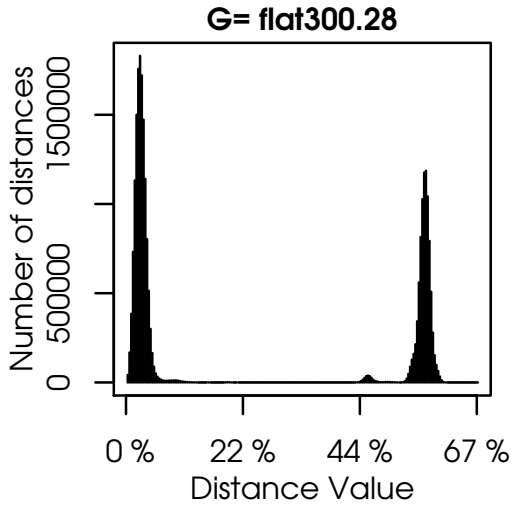
Soit une recherche Tabou qui explore l'espace:

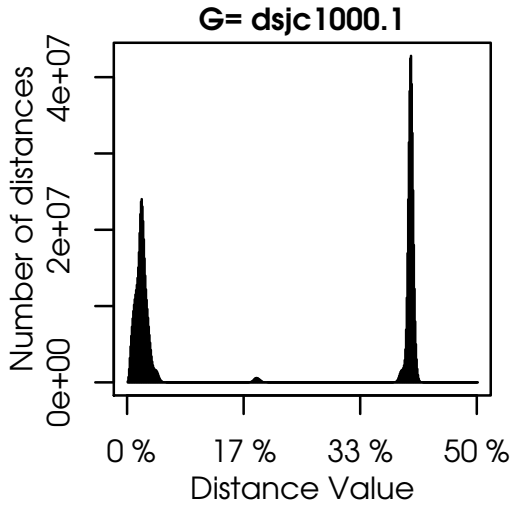
- Mémoriser les 40.000 premières solutions de qualité
- L'histogramme des distances montre le nombre de paires de solutions distancées par chaque valeur de distance
- Distances petites : distances intra-classes
- Distances longues : distance inter-classes
- Les distances petites sont toujours plus petites que $10\% \cdot |V|$
- La sphère de la coloration C : ensemble de colorations situées à moins de $10\% \cdot |V|$ de C

Étude de la trajectoire sur une longue période

Soit une recherche Tabou qui explore l'espace:

- Mémoriser les 40.000 premières solutions de qualité
- L'histogramme des distances montre le nombre de paires de solutions distancées par chaque valeur de distance
- Distances petites : distances intra-classes
- Distances longues : distance inter-classes
- Les distances petites sont toujours plus petites que $10\% \cdot |V|$
- La sphère de la coloration C : ensemble de colorations situées à moins de $10\% \cdot |V|$ de C

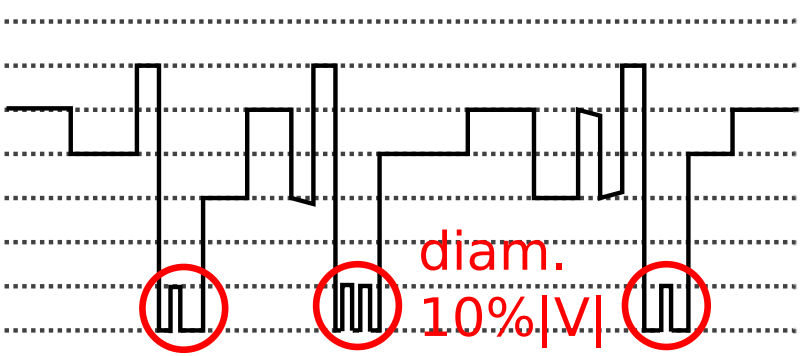


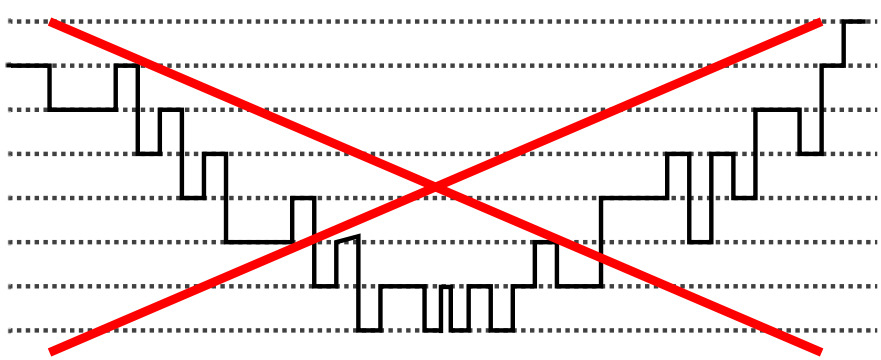


Conclusions sur l'espace de recherche

Cette analyse nous a conduit à l'hypothèse de "clusterisation"

- Les meilleures solutions candidates ne sont pas dispersées aléatoirement: elle sont regroupées dans des clusters
- Ces clusters peuvent être couverts par des sphères de rayon $R = 10\% \cdot |V|$





- Le profil de l'espace de recherche ressemble à la figure gauche
- La figure de droite (grande vallée) est plutôt applicable au problème du voyageur de commerce, selon [Booese et. al. 1993, On the big valley and adaptive multi-start for discrete global optimizations])

Résumé

Cibler la recherche en utilisant l'hypothèse de clusterisation

- Hypothèse de clusterisation: les meilleures solutions sont groupées en clusters qui peuvent être couverts par des sphères
- La probabilité d'avoir un optimum global isolé est faible

$$\Longrightarrow$$

Stratégie de recherche en deux étapes (deux algorithmes)

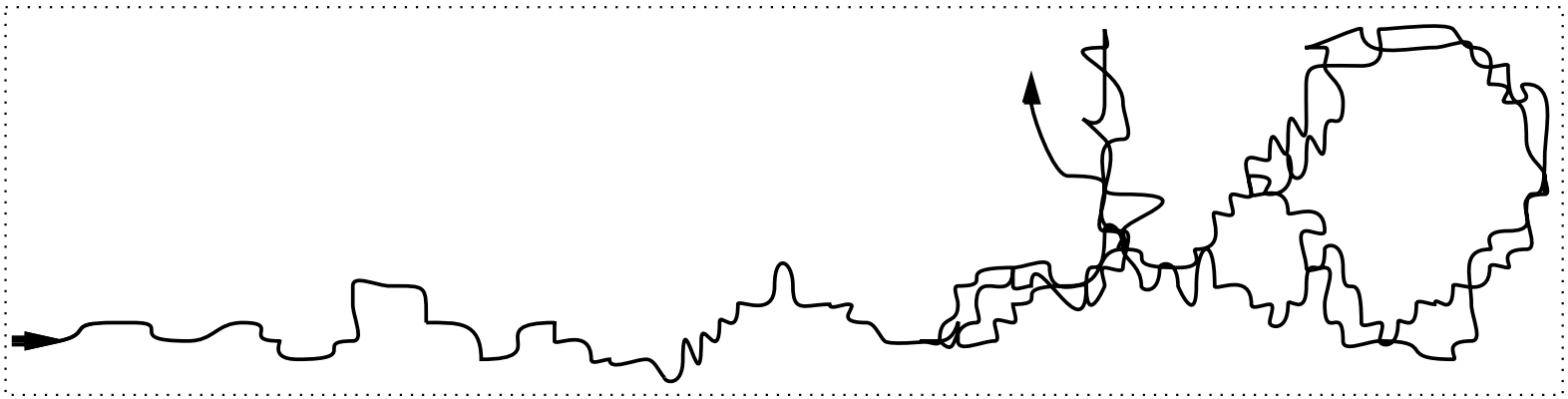
1. Localiser le plus de clusters avec des solutions de qualité
2. Chercher l'optimum global à l'intérieur d'un tel cluster

- Étape 1 : l'objectif est de trouver le plus de clusters différents
  - Mémoriser la trajectoire et éviter les sphères déjà visitées

Etape 1: Recherche de clusters différents

Mémoriser la trajectoire et éviter les sphères déjà visitées

- Il est impossible de mémoriser toute la trajectoire pour chercher ensuite les clusters
- Il est possible de faire une mémorisation à gros grain, sphère par sphère

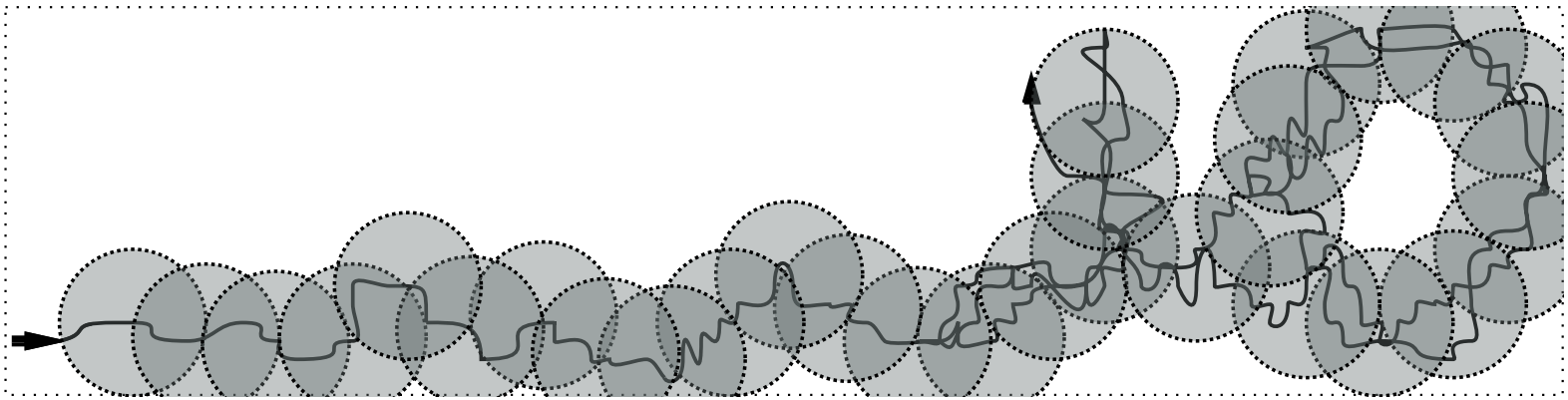


1. Objectifs: éviter des sphères déjà visitées et chercher continuellement de nouvelles sphères (clusters)

Etape 1: Recherche de clusters différents

Mémoriser la trajectoire et éviter les sphères déjà visitées

- Il est impossible de mémoriser toute la trajectoire pour chercher ensuite les clusters
- Il est possible de faire une mémorisation à gros grain, sphère par sphère

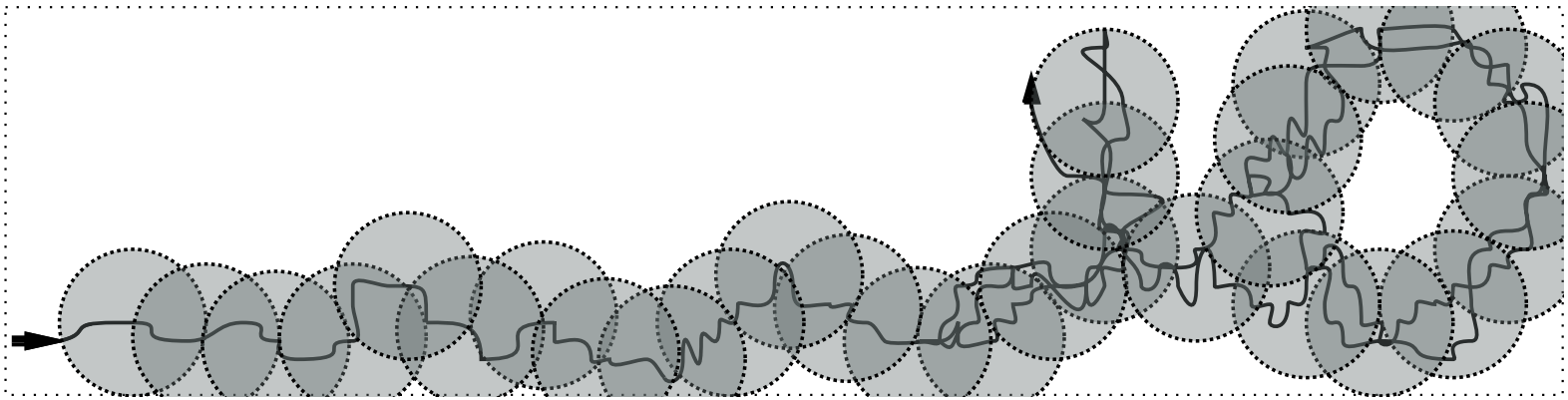


1. Objectifs: éviter des sphères déjà visitées et chercher continuellement de nouvelles sphères (clusters)

Etape 1: Recherche de clusters différents

Mémoriser la trajectoire et éviter les sphères déjà visitées

- Il est impossible de mémoriser toute la trajectoire pour chercher ensuite les clusters
- Il est possible de faire une mémorisation à gros grain, sphère par sphère



1. Objectifs: éviter des sphères déjà visitées et chercher continuellement de nouvelles sphères (clusters)

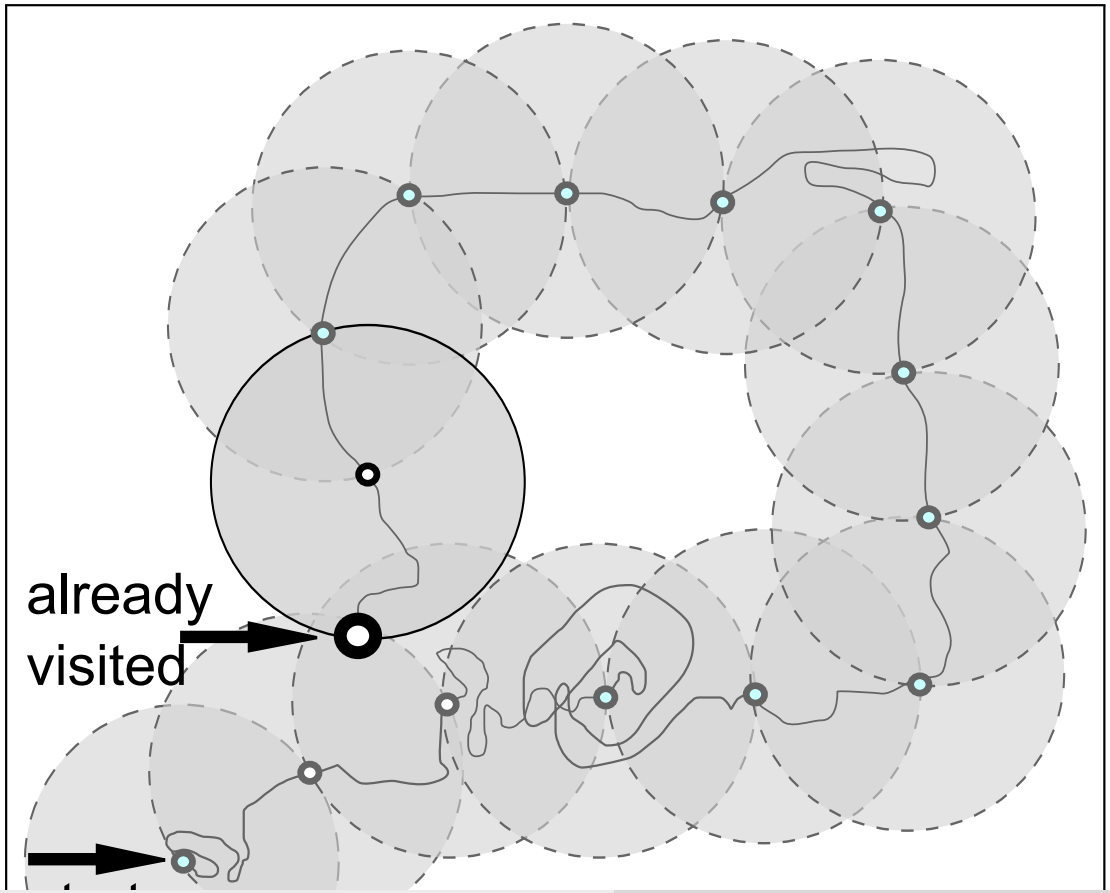
Etape 1: Recherche de clusters différents (diversification)

Diversification: Comment s'assurer que la recherche ne re-visite les mêmes sphères

- Soit une itération où la recherche visite une solution couverte par une sphère déjà visitée

  $\Longrightarrow$ phase de diversification:

  - perturber le processus de recherche (incrémenter la longueur Tabou)



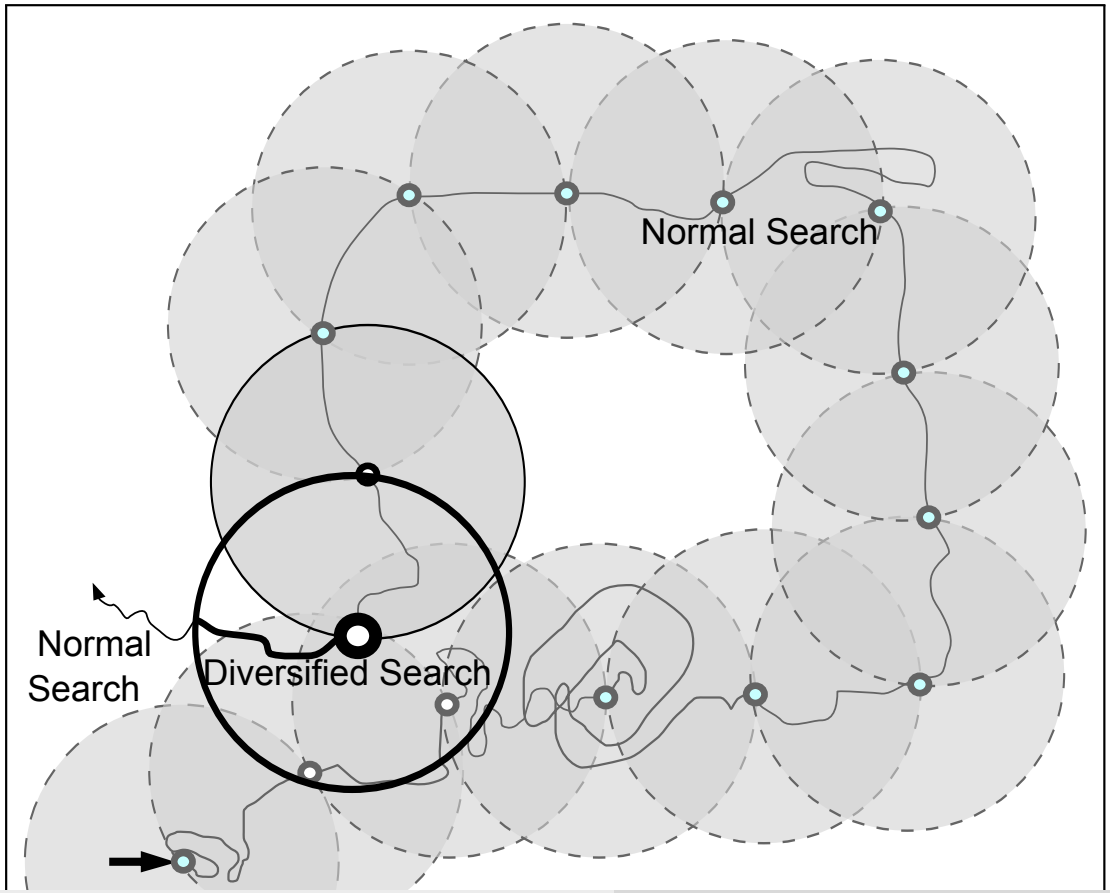
Etape 1: Recherche de clusters différents (diversification)

Diversification: Comment s'assurer que la recherche ne re-visite les mêmes sphères

- Soit une itération où la recherche visite une solution couverte par une sphère déjà visitée

  $\Longrightarrow$ phase de diversification:

  - perturber le processus de recherche (incrémenter la longueur Tabou)



Etape 2: recherche approfondie dans les sphères

Etape 2: Exploitation de meilleures sphères trouvées à l'étape 1

- D'après l'hypothèse de clusterisation, l'optimum global peut se retrouver dans une sphère avec d'autres solutions de qualité
- Les meilleures sphères sont re-explorées séparément sous tous les angles $\longrightarrow$ algorithme d'exploitation fine de la sphère

Graphe	k^*	Notre Alg.	[1]	[2]	[3]	[4]	[5]	[6]
		2010	2008	2008	2008	1993	1999	2008
*dsjc*1000.1	20	*20*	20	20	20	21	20	20
*dsjc*1000.5	83	85	87	88	84	88	83	83
*dsjc*1000.9	224	223	224	225	224	226	224	225
*flat*300.28	28	*28*	28	28	31	31	31	31
*flat*1000.76	82	85	86	87	84	89	83	82
*le*450.25*c*	25	*25*	26	25	26	25	26	25

[1] Hertz et. al. Variable space search for graph coloring, [2] Blöchliger et. al. A graph coloring heuristic using partial solutions and a reactive tabu scheme, [3] Galinier et. al. An adaptive memory algorithm for the k-coloring problem, [4] C. Morgenstern. Distributed coloration neighborhood search (DIMACS), [5] Galinier et. al.Hybrid evolutionary algorithms for graph coloring, [6] Malaguti et. al. A Metaheuristic Approach for the Vertex Coloring Problem

Etape 2: recherche approfondie dans les sphères

Etape 2: Exploitation de meilleures sphères trouvées à l'étape 1

- D'après l'hypothèse de clusterisation, l'optimum global peut se retrouver dans une sphère avec d'autres solutions de qualité
- Les meilleures sphères sont re-explorées séparément sous tous les angles $\longrightarrow$ algorithme d'exploitation fine de la sphère

Graphe	k^*	Notre Alg.	[1]	[2]	[3]	[4]	[5]	[6]
		2010	2008	2008	2008	1993	1999	2008
*dsjc*1000.1	20	*20*	20	20	20	21	20	20
*dsjc*1000.5	83	85	87	88	84	88	83	83
*dsjc*1000.9	224	**223**	224	225	224	226	224	225
*flat*300.28	28	*28*	28	28	31	31	31	31
*flat*1000.76	82	85	86	87	84	89	83	82
*le*450.25*c*	25	*25*	26	25	26	25	26	25

[1] Hertz et. al. Variable space search for graph coloring, [2] Blöchliger et. al. A graph coloring heuristic using partial solutions and a reactive tabu scheme, [3] Galinier et. al. An adaptive memory algorithm for the k-coloring problem, [4] C. Morgenstern. Distributed coloration neighborhood search (DIMACS), [5] Galinier et. al.Ḣybrid evolutionary algorithms for graph coloring, [6] Malaguti et. al. A Metaheuristic Approach for the Vertex Coloring Problem

Conclusion optimisation

- Grâce à la clusterisation, on a construit une méthode d'optimisation en deux étapes:
  - Trouver un grand nombre de sphères différentes
  - Exploration fine des meilleures sphères
- Cette approche de sphères n'aurait pas été très utile sur une structure d'espace de type grande vallée.
- Des tests empiriques montrent une grande compétitivité par rapport à une littérature assez grande
- Des extensions ont été réalisés pour d'autres types de méta-heuristiques et d'autres problèmes [Porumbel et al., Spacing Memetic Algorithms, GECCO 2011]

Résumé

1. Classification des meilleures solutions candidates
   - Représentation graphique des solutions de bonne qualité
   - Classification des solutions découvertes par la recherche Locale
2. Exploration guidée par distance
   - Etape 1: Mémorisation et localisation des clusters
   - Etape 2: Exploration fine dans les sphères
3. Algorithme linéaire pour calculer la distance de transfert entre partitions
   - Définitions Distance et Similarité
   - Méthode classique et méthode nouvelle
   - Théorèmes garantissant la complexité linéaire pour les cas définis
4. Conclusions et discussion

La Distance de transfert: définition

- La distance de transfert entre partitions est définie depuis 1965 par Règnier [Régnier, Sur quelques aspects mathématiques des problèmes de classification automatique]
  - = le nombre minimal d'éléments de S qui doivent être transférés entre les classes d'une partition pour obtenir l'autre
- Applications en classification, clustering ou segmentation d'image
  - Exemple: comparer la partition/segmentation générée par un algorithme avec la segmentation idéale (le "golden standard")

Illustration du calcul

Soient P_1 et P_2 deux k-partitions:

- La distance $d(P_1, P_2)$ est le nombre minimum d'éléments qui doivent être transférés entre les classes d'une partition pour passer d'une partition à l'autre

---

P_1: |1 2 3 4 | 5 6 7 | 8 9|

Combien de transferts pour égaliser les deux partitions?

P_2: |1 2 3 4 8| 5 9 | 6 7|

Illustration du calcul

Soient P_1 et P_2 deux k-partitions:

- La distance $d(P_1, P_2)$ est le nombre minimum d'éléments qui doivent être transférés entre les classes d'une partition pour passer d'une partition à l'autre

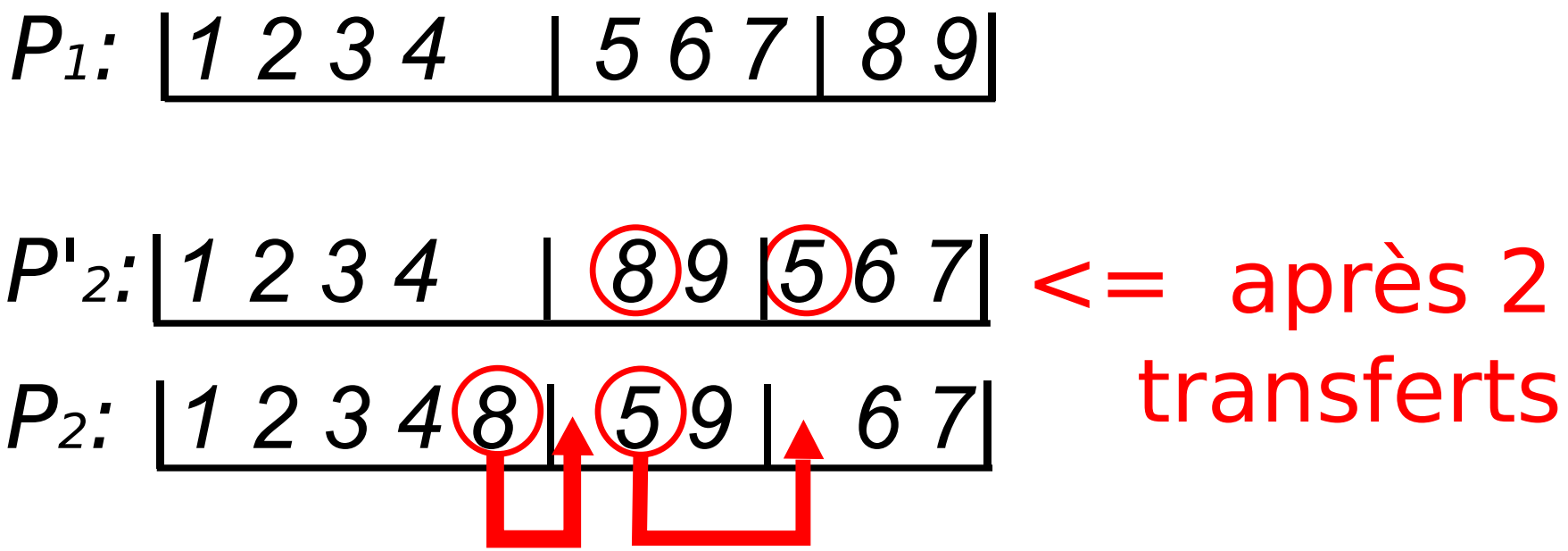
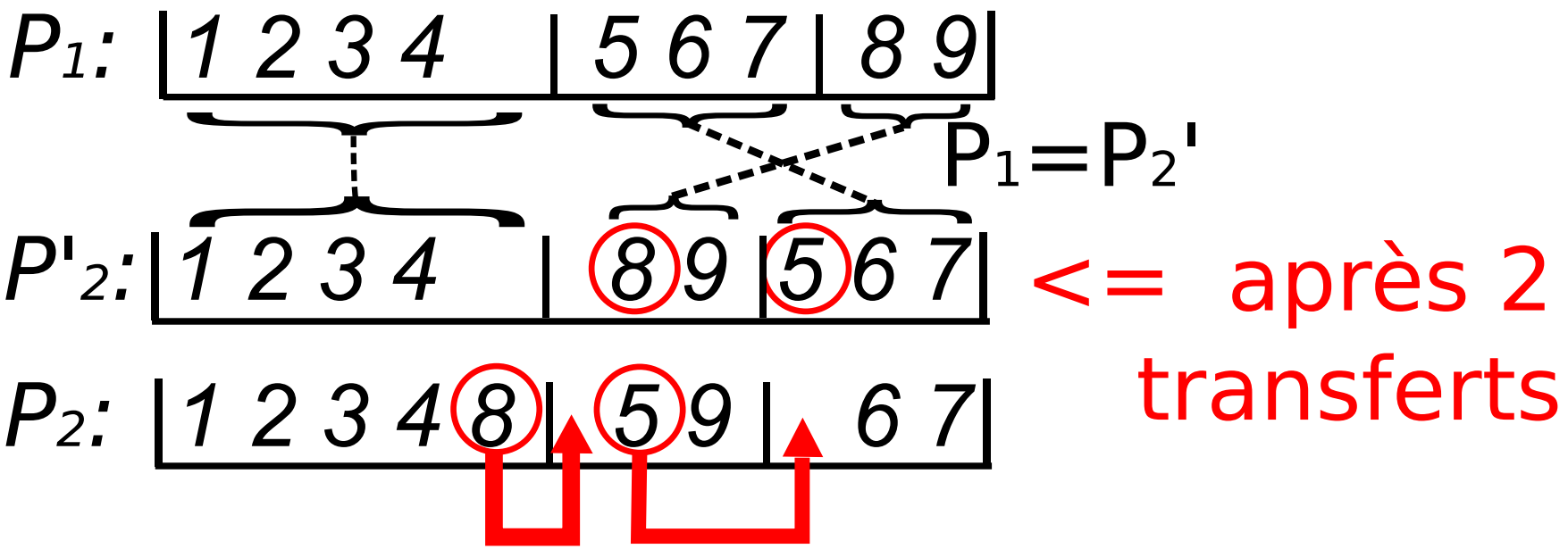


Illustration du calcul

Soient P_1 et P_2 deux k-partitions:

- La distance $d(P_1, P_2)$ est le nombre minimum d'éléments qui doivent être transférés entre les classes d'une partition pour passer d'une partition à l'autre

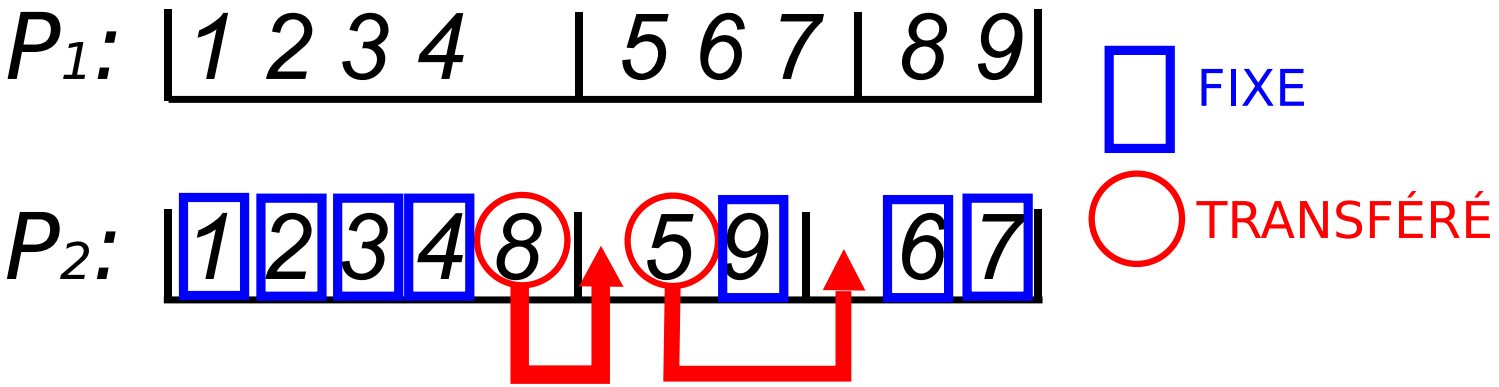


Similarité entre partitions

Soit P_1 et P_2 deux k-partitions de S:

- La similarité $s(P_1, P_2)$ est le nombre maximum d'éléments qui n'ont pas besoin d'être transférés pour passer d'une partition à l'autre
- La distance est calculée via:

$$\boxed{d(P_1, P_2) = n - s(P_1, P_2)}$$



7 éléments fixes, 2 éléments transférés

$\Rightarrow (P_1, P_2) + d(P_1, P_2) = 7 + 2 = 9$

Résumé

3. Algorithme linéaire pour calculer la distance de transfert entre partitions
   - Définitions Distance et Similarité
   - Méthode classique et méthode nouvelle
   - Théorèmes garantissant la complexité linéaire pour les cas définis

Calcul de la similarité : méthode classique en $O(n + k^2)$

Deux étapes:

P_1: |1 2 3 4 | 5 6 7 | 8 9|

P_2: |1 2 3 4 8 | 5 9 | 6 7|

Calcul de la similarité : méthode classique en $O(n + k^2)$

Étape 1 Construire la matrice de similarité $T = [T_{ij}]_{k \times k}$ où:

$$\boxed{T_{ij} = |P_1^i \cap P_2^j|}$$

$(P_1^i = \text{classe } i \text{ de } P_1,\ P_2^j = \text{classe } j \text{ de } P_2)$

Étape 2 Calculer une correspondance entre les classes de P_1 et de P_2 : une bijection $\sigma : \{1, 2, \ldots, k\} \to \{1, 2, \ldots, k\}$ qui maximise

$$\max_{\sigma} \sum_{1 \leq i \leq k} T_{i,\sigma(i)}$$

P_1: |1 2 3 4 | 5 6 7 | 8 9|

P_2: |1 2 3 4 8 | 5 9 | 6 7|

$$\Rightarrow \underset{\text{(Mat. de Similarité)}}{T} = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 1 & 2 \\ 1 & 1 & 0 \end{bmatrix}$$

Calcul de la similarité : méthode classique en $O(n + k^2)$

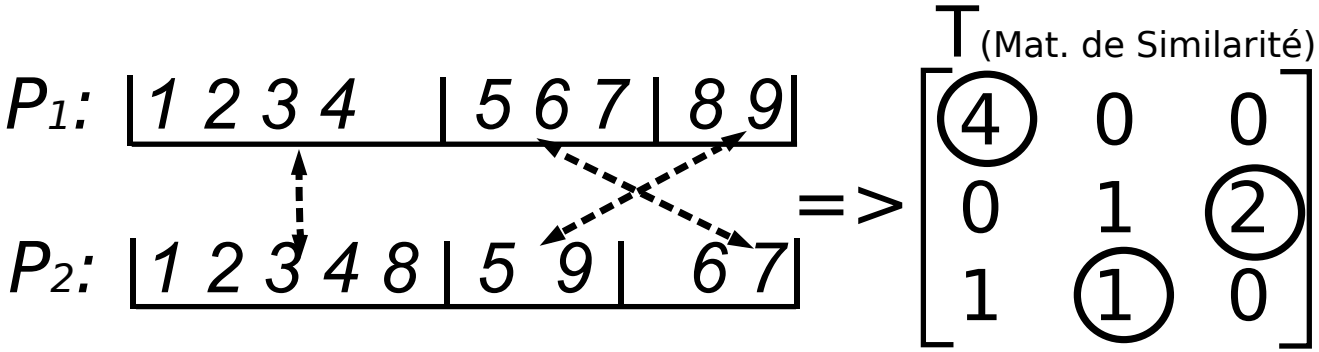
Étape 1 Construire la matrice de similarité $T = [T_{ij}]_{k \times k}$ où:

$$\boxed{T_{ij} = |P_1^i \cap P_2^j|}$$

$(P_1^i = \text{classe } i \text{ de } P_1,\ P_2^j = \text{classe } j \text{ de } P_2)$

Étape 2 Calculer une correspondance entre les classes de P_1 et de P_2 : une bijection $\sigma : \{1, 2, \ldots, k\} \to \{1, 2, \ldots, k\}$ qui maximise

$$\boxed{\max_{\sigma} \sum_{1 \le i \le k} T_{i,\sigma(i)}}$$



Calcul de la similarité : méthode classique en $O(n + k^2)$

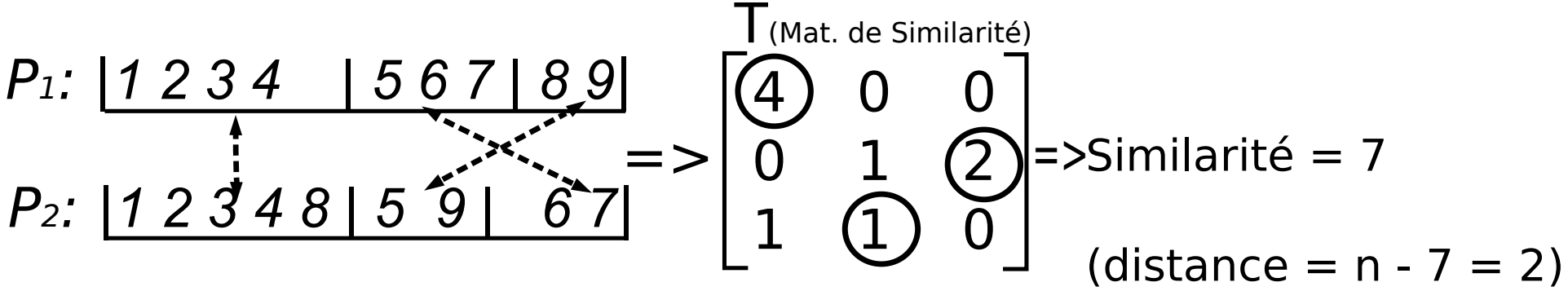
Étape 1 Construire la matrice de similarité $T = [T_{ij}]_{k \times k}$ où:

$$\boxed{T_{ij} = |P_1^i \cap P_2^j|}$$

$$(P_1^i = \text{classe } i \text{ de } P_1,\ P_2^j = \text{classe } j \text{ de } P_2)$$

Étape 2 Calculer une correspondance entre les classes de P_1 et de P_2 : une bijection $\sigma : \{1, 2, \ldots, k\} \rightarrow \{1, 2, \ldots, k\}$ qui maximise

$$\boxed{\max_{\sigma} \sum_{1 \leq i \leq k} T_{i,\sigma(i)}}$$



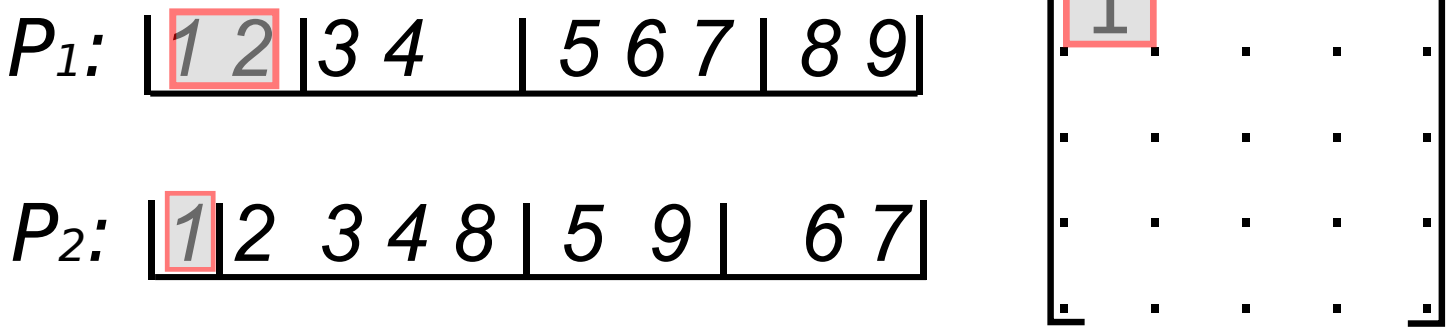
Calcul Mat. Similarité : Méthode Classique en $O(n + k^2)$

Par définition: La matrice de similarité T est calculée via:

$$\boxed{T_{ij} = |P_1^i \cap P_2^j|}$$

(P_1^i =i-ème classe de P_1, P_2^j =j-ème classe de P_2)

$$T_{11} = |P_1^1 \cap P_2^1| = |\{1,2\} \cap \{1\}| = 1$$



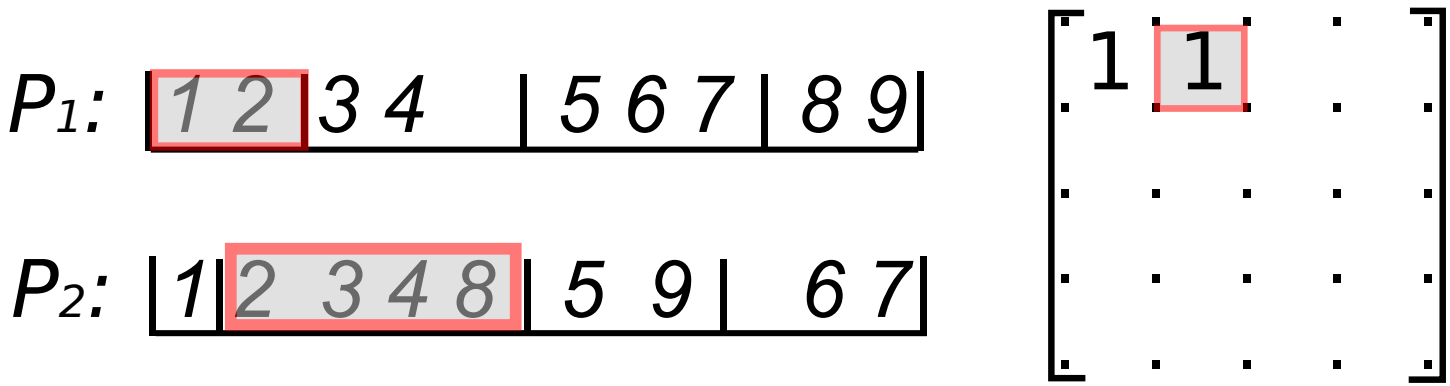
Calcul Mat. Similarité : Méthode Classique en $O(n + k^2)$

Par définition: La matrice de similarité T est calculée via:

$$\boxed{T_{ij} = |P_1^i \cap P_2^j|}$$

(P_1^i =i-ème classe de P_1, P_2^j =j-ème classe de P_2)

$$T_{12} = |P_1^1 \cap P_2^2| = |\{1,2\} \cap \{2,3,4,8\}| = 1$$



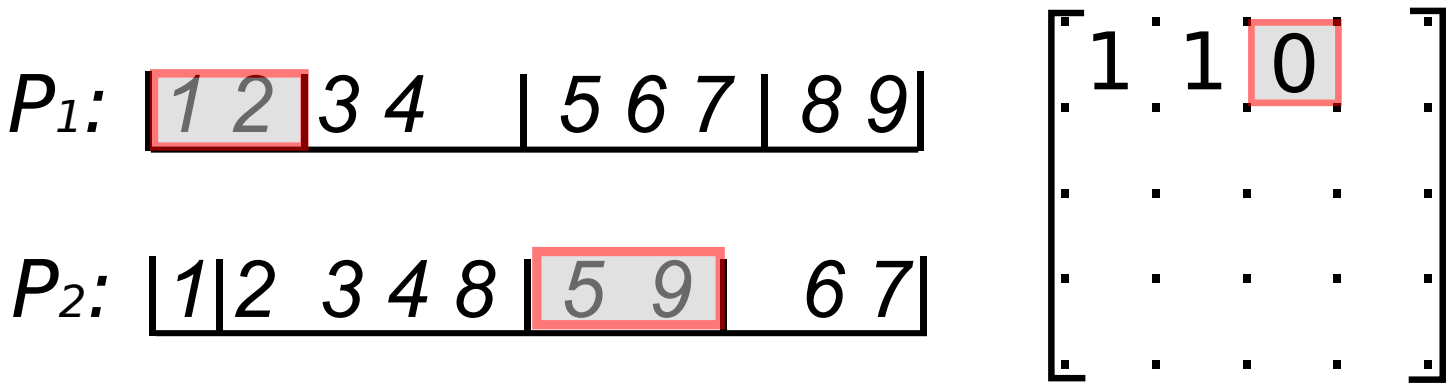
Calcul Mat. Similarité : Méthode Classique en $O(n + k^2)$

Par définition: La matrice de similarité T est calculée via:

$$\boxed{T_{ij} = |P_1^i \cap P_2^j|}$$

(P_1^i =i-ème classe de P_1, P_2^j =j-ème classe de P_2)

$$T_{13} = |P_1^1 \cap P_2^3| = |\{1,2\} \cap \{5,9\}| = 0$$



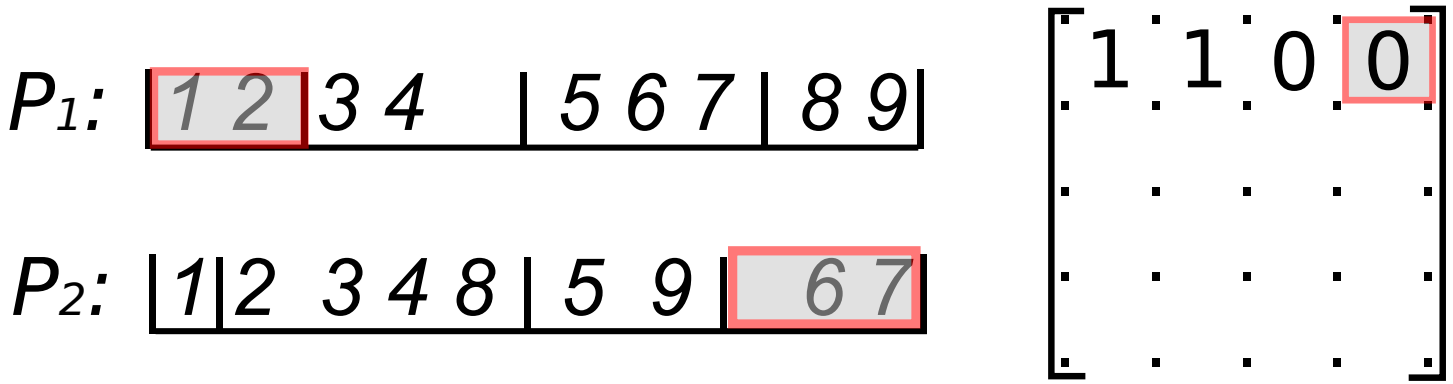
Calcul Mat. Similarité : Méthode Classique en $O(n + k^2)$

Par définition: La matrice de similarité T est calculée via:

$$\boxed{T_{ij} = |P_1^i \cap P_2^j|}$$

(P_1^i =i-ème classe de P_1, P_2^j =j-ème classe de P_2)

$$T_{14} = |P_1^1 \cap P_2^3| = |\{1,2\} \cap \{6,7\}| = 0$$



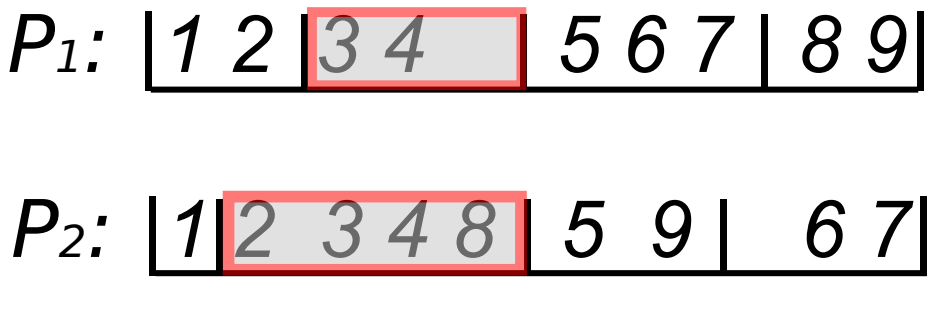
Calcul Mat. Similarité : Méthode Classique en $O(n + k^2)$

Par définition: La matrice de similarité T est calculée via:

$$\boxed{T_{ij} = |P_1^i \cap P_2^j|}$$

(P_1^i =i-ème classe de P_1, P_2^j =j-ème classe de P_2)

$$T_{22} = |P_1^2 \cap P_2^2| = |\{3,4\} \cap \{2,3,4,8\}| = 0$$



$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 2 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{bmatrix}$$

Calcul Mat. Similarité : Méthode Classique en $O(n + k^2)$

Par définition: La matrice de similarité T est calculée via:

$$\boxed{T_{ij} = |P_1^i \cap P_2^j|}$$

(P_1^i =i-ème classe de P_1, P_2^j =j-ème classe de P_2)

Après le calcul de $k^2 = 16$ intersections:

P_1: |1 2|3 4 |5 6 7|8 9|

P_2: |1|2 3 4 8|5 9|6 7|

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 2 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Calcul Mat. Similarité : Méthode Classique en $O(n + k^2)$

Par définition: La matrice de similarité T est calculée via:

$$\boxed{T_{ij} = |P_1^i \cap P_2^j|}$$

(P_1^i =i-ème classe de P_1, P_2^j =j-ème classe de P_2)

→ l'affectation maximale est construite via un algorithme hongrois modifié

P_1: |1 2|3 4 | 5 6 7|8 9|

P_2: |1|2 3 4 8|5 9|6 7|

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 2 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Calcul Matrice de Similarité : Nouvelle Méthode

1. Allocation matrice $k \times k$ sans initialisation: $O(1)$ temps
2. parcourir les éléments $x \in \{1, 2, \ldots, n\}$ et affecter
$$T_{P_1[x],P_2[x]} := 0$$
3. parcourir les éléments $x \in \{1, 2, \ldots, n\}$ et affecter
$$T_{P_1[x],P_2[x]} := T_{P_1[x],P_2[x]} + 1$$

P_1: |1 2|3 4 | 5 6 7| 8 9|

P_2: |1|2 3 4 8| 5 9| 6 7|

$$\begin{bmatrix} ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \end{bmatrix}$$

Calcul Matrice de Similarité : Nouvelle Méthode

1. Allocation matrice $k \times k$ sans initialisation: $O(1)$ temps
2. parcourir les éléments $x \in \{1, 2, \ldots, n\}$ et affecter
$$T_{P_1[x],P_2[x]} := 0$$
3. parcourir les éléments $x \in \{1, 2, \ldots, n\}$ et affecter
$$T_{P_1[x],P_2[x]} := T_{P_1[x],P_2[x]} + 1$$

$$x = 1 \Rightarrow (P_1[1], P_2[1]) = (1, 1) \Rightarrow T_{1,1} = 0$$

P_1: |1 2|3 4 | 5 6 7 | 8 9|

P_2: |1|2 3 4 8 | 5 9 | 6 7|

$$\begin{bmatrix} 0 & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \end{bmatrix}$$

Calcul Matrice de Similarité : Nouvelle Méthode

1. Allocation matrice $k \times k$ sans initialisation: $O(1)$ temps
2. parcourir les éléments $x \in \{1, 2, \ldots, n\}$ et affecter
$$T_{P_1[x],P_2[x]} := 0$$
3. parcourir les éléments $x \in \{1, 2, \ldots, n\}$ et affecter
$$T_{P_1[x],P_2[x]} := T_{P_1[x],P_2[x]} + 1$$

$$x = 2 \Rightarrow (P_1[2], P_2[2]) = (1, 2) \Rightarrow T_{1,2} = 0$$

P_1: |1 2|3 4 | 5 6 7 | 8 9|

P_2: |1|2 3 4 8 | 5 9 | 6 7|

$$\begin{bmatrix} 0 & 0 & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \end{bmatrix}$$

Calcul Matrice de Similarité : Nouvelle Méthode

1. Allocation matrice $k \times k$ sans initialisation: $O(1)$ temps
2. parcourir les éléments $x \in \{1, 2, \ldots, n\}$ et affecter
$$T_{P_1[x],P_2[x]} := 0$$
3. parcourir les éléments $x \in \{1, 2, \ldots, n\}$ et affecter
$$T_{P_1[x],P_2[x]} := T_{P_1[x],P_2[x]} + 1$$

$$x = 3 \Rightarrow (P_1[3], P_2[3]) = (2, 2) \Rightarrow T_{2,2} = 0$$

P_1: |1 2 | 3 4 | 5 6 7 | 8 9|

P_2: |1|2 | 3 4 8 | 5 9 | 6 7|

$$\begin{bmatrix} 0 & 0 & ? & ? \\ ? & 0 & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \end{bmatrix}$$

Calcul Matrice de Similarité : Nouvelle Méthode

1. Allocation matrice $k \times k$ sans initialisation: $O(1)$ temps
2. parcourir les éléments $x \in \{1, 2, \ldots, n\}$ et affecter
$$T_{P_1[x],P_2[x]} := 0$$
3. parcourir les éléments $x \in \{1, 2, \ldots, n\}$ et affecter
$$T_{P_1[x],P_2[x]} := T_{P_1[x],P_2[x]} + 1$$

$$x = 4 \Rightarrow (P_1[4], P_2[4]) = (2, 2) \Rightarrow T_{2,2} = 0$$

P_1: |1 2 |3 4 | 5 6 7 | 8 9|

P_2: |1|2 3 4 8 | 5 9 | 6 7|

$$\begin{bmatrix} 0 & 0 & ? & ? \\ ? & 0 & ? & ? \\ ? & ? & ? & ? \\ ? & ? & ? & ? \end{bmatrix}$$

Calcul Matrice de Similarité : Nouvelle Méthode

1. Allocation matrice $k \times k$ sans initialisation: $O(1)$ temps
2. parcourir les éléments $x \in \{1, 2, \ldots, n\}$ et affecter
$$T_{P_1[x],P_2[x]} := 0$$
3. parcourir les éléments $x \in \{1, 2, \ldots, n\}$ et affecter
$$T_{P_1[x],P_2[x]} := T_{P_1[x],P_2[x]} + 1$$

$$x = 5 \Rightarrow (P_1[5], P_2[5]) = (3, 3) \Rightarrow T_{3,3} = 0$$

P_1: |1 2 |3 4 | 5 6 7 | 8 9|

P_2: |1|2 3 4 8 | 5 9 | 6 7|

$$\begin{bmatrix} 0 & 0 & ? & ? \\ ? & 0 & ? & ? \\ ? & ? & 0 & ? \\ ? & ? & ? & ? \end{bmatrix}$$

Calcul Matrice de Similarité : Nouvelle Méthode

1. Allocation matrice $k \times k$ sans initialisation: $O(1)$ temps
2. parcourir les éléments $x \in \{1, 2, \ldots, n\}$ et affecter
$$T_{P_1[x],P_2[x]} := 0$$
3. parcourir les éléments $x \in \{1, 2, \ldots, n\}$ et affecter
$$T_{P_1[x],P_2[x]} := T_{P_1[x],P_2[x]} + 1$$

Le dernier est $x = 9$; après $n = 9$ étapes, l'algorithme initialise toutes les positions clés, celles qui ne sont pas "?"

P_1: |1 2 |3 4 | 5 6 7 | 8 9|

P_2: |1|2 3 4 8 | 5 9| 6 7|

$$\begin{bmatrix} 0 & 0 & ? & ? \\ ? & 0 & ? & ? \\ ? & ? & 0 & 0 \\ ? & 0 & 0 & ? \end{bmatrix}$$

Calcul Matrice de Similarité : Nouvelle Méthode

1. Allocation matrice $k \times k$ sans initialisation: $O(1)$ temps
2. parcourir les éléments $x \in \{1, 2, \ldots, n\}$ et affecter
$$T_{P_1[x],P_2[x]} := 0$$
3. parcourir les éléments $x \in \{1, 2, \ldots, n\}$ et affecter
$$T_{P_1[x],P_2[x]} := T_{P_1[x],P_2[x]} + 1$$

$$x = 1 \Rightarrow (P_1[1], P_2[1]) = (1, 1) \Rightarrow T_{1,1} = 0 + 1 = 1$$

P_1: |1 2|3 4 | 5 6 7 | 8 9|

P_2: |1|2 3 4 8 | 5 9 | 6 7|

$$\begin{bmatrix} 1 & 0 & ? & ? \\ ? & 0 & ? & ? \\ ? & ? & 0 & 0 \\ ? & 0 & 0 & ? \end{bmatrix}$$

Calcul Matrice de Similarité : Nouvelle Méthode

1. Allocation matrice $k \times k$ sans initialisation: $O(1)$ temps
2. parcourir les éléments $x \in \{1, 2, \ldots, n\}$ et affecter
$$T_{P_1[x],P_2[x]} := 0$$
3. parcourir les éléments $x \in \{1, 2, \ldots, n\}$ et affecter
$$T_{P_1[x],P_2[x]} := T_{P_1[x],P_2[x]} + 1$$

$$x = 2 \Rightarrow (P_1[2], P_2[2]) = (1, 2) \Rightarrow T_{1,2} = 0 + 1 = 1$$

P_1: |1 2|3 4 | 5 6 7 | 8 9|

P_2: |1|2 3 4 8 | 5 9 | 6 7|

$$\begin{bmatrix} 1 & 1 & ? & ? \\ ? & 0 & ? & ? \\ ? & ? & 0 & 0 \\ ? & 0 & 0 & ? \end{bmatrix}$$

Calcul Matrice de Similarité : Nouvelle Méthode

1. Allocation matrice $k \times k$ sans initialisation: $O(1)$ temps
2. parcourir les éléments $x \in \{1, 2, \ldots, n\}$ et affecter
$$T_{P_1[x],P_2[x]} := 0$$
3. parcourir les éléments $x \in \{1, 2, \ldots, n\}$ et affecter
$$T_{P_1[x],P_2[x]} := T_{P_1[x],P_2[x]} + 1$$

$$x = 3 \Rightarrow (P_1[3], P_2[3]) = (2, 2) \Rightarrow T_{2,2} = 0 + 1 = 1$$

P_1: |1 2 | 3 4 | 5 6 7 | 8 9|

P_2: |1|2 | 3 4 8 | 5 9 | 6 7|

$$\begin{bmatrix} 1 & 1 & ? & ? \\ ? & 1 & ? & ? \\ ? & ? & 0 & 0 \\ ? & 0 & 0 & ? \end{bmatrix}$$

Calcul Matrice de Similarité : Nouvelle Méthode

1. Allocation matrice $k \times k$ sans initialisation: $O(1)$ temps
2. parcourir les éléments $x \in \{1, 2, \ldots, n\}$ et affecter
$$T_{P_1[x],P_2[x]} := 0$$
3. parcourir les éléments $x \in \{1, 2, \ldots, n\}$ et affecter
$$T_{P_1[x],P_2[x]} := T_{P_1[x],P_2[x]} + 1$$

$$x = 4 \Rightarrow (P_1[4], P_2[4]) = (2, 2) \Rightarrow T_{2,2} = 1 + 1 = 2$$

P_1: |1 2|3 4 | 5 6 7| 8 9|

P_2: |1|2 3 4 8| 5 9| 6 7|

$$\begin{bmatrix} 1 & 1 & ? & ? \\ ? & 2 & ? & ? \\ ? & ? & 0 & 0 \\ ? & 0 & 0 & ? \end{bmatrix}$$

Calcul Matrice de Similarité : Nouvelle Méthode

1. Allocation matrice $k \times k$ sans initialisation: $O(1)$ temps
2. parcourir les éléments $x \in \{1, 2, \ldots, n\}$ et affecter
$$T_{P_1[x],P_2[x]} := 0$$
3. parcourir les éléments $x \in \{1, 2, \ldots, n\}$ et affecter
$$T_{P_1[x],P_2[x]} := T_{P_1[x],P_2[x]} + 1$$

Après $n = 9$ étapes, l'algorithme calcule les éléments clés; les non-nuls.
(les "?" sont nuls et on les ignore)

P_1: |1 2|3 4 |5 6 7|8 9|

P_2: |1|2 3 4 8|5 9|6 7|

$$\begin{bmatrix} 1 & 1 & ? & ? \\ ? & 2 & ? & ? \\ ? & ? & 1 & 2 \\ ? & 1 & 1 & ? \end{bmatrix}$$

Calcul affectation : méthode proposée, méthode classique

1. Calculer la matrice T incomplète, i.e. les éléments clés, en $O(n)$
2. Appliquer des théorèmes qui garantissent que ces éléments sont suffisants pour calculer l'affectation maximale. Exemple:

Théorème

Si pour toute ligne i du T, il existe une colonne j tel que $T_{ij} > T_{ij'} + T_{i'j}$ (où $i \neq i', j \neq j'$), alors la distance peut être calculée en $O(n)$.

$$\begin{matrix} \boxed{2} & 1 & ? & ? \\ ? & \boxed{2} & ? & ? \\ ? & ? & 1 & \boxed{2} \\ ? & 1 & \boxed{3} & 0 \end{matrix}$$

1. Calculer la matrice T complète en $O(n + k^2)$
2. Transformation $\boxed{T'_{ij} = n - T_{ij}}$
3. L'algorithme hongrois est appliqué sur T'

Calcul affectation : méthode proposée, méthode classique

1. Calculer la matrice T incomplète, i.e. les éléments clés, en $O(n)$
2. Appliquer des théorèmes qui garantissent que ces éléments sont suffisants pour calculer l'affectation maximale. Exemple:

Théorème

Si pour toute ligne i du T, il existe une colonne j tel que $\boxed{T_{ij} > T_{ij'} + T_{i'j}}$ *(où $i \neq i', j \neq j'$), alors la distance peut être calculée en $O(n)$.*

$$\begin{matrix} \boxed{2} & 1 & ? & ? \\ ? & \boxed{2} & ? & ? \\ ? & ? & 1 & \boxed{2} \\ ? & 1 & \boxed{3} & 0 \end{matrix}$$

1. Calculer la matrice T complète en $O(n + k^2)$
2. Transformation $\boxed{\boxed{T'_{ij} = n - T_{ij}}}$
3. L'algorithme hongrois est appliqué sur T'

Résumé

3. Algorithme linéaire pour calculer la distance de transfert entre partitions
   - Définitions Distance et Similarité
   - Méthode classique et méthode nouvelle
   - Théorèmes garantissant la complexité linéaire pour les cas définis

Affectation Maximale: Nouvelle méthode

1. Calculer la matrice T incomplète, i.e. les éléments clés, en $O(n)$
2. Utiliser des théorèmes qui garantissent que l'affectation maximale peut être calculée juste avec les éléments clés. Pour cela, parcourir les éléments clés plusieurs fois et déterminer:
   - la plus grande valeur Max1 sur chaque ligne/colonne, en $O(n)$
   - la deuxième plus grande valeur Max2 sur chaque ligne/colonne,en $O(n)$
   - la position des valeurs qui sont maximales sur leurs ligne **et** leurs colonne, – les valeurs marquées ci-dessous –, en $O(n)$

*Max*1/*Max*2	1/0	2/1	1/1	2/0
1/1	$\boxed{2}$	1	?	?
2/0	?	$\boxed{2}$	?	?
2/1	?	?	1	$\boxed{2}$
1/1	?	1	$\boxed{1}$	0

Premier théorème

Théorème

Si pour toute ligne i du T, il existe une colonne j tel que $T_{ij} > T_{ij'} + T_{i'j}$ *(où $i \neq i', j \neq j'$), alors la distance peut être calculée en $O(n)$.*

- Il faut juste trouver l'affectation maximale σ; on montre $\sigma(i) = j$
- T_{ij} doit être l'unique maximum de la ligne i, car l'inégalité est stricte
- Il suffit de parcourir les éléments clés pour trouver ce maximum de ligne T_{ij} et vérifier la condition
- Supposons qu'il y a une affectation maximale σ avec $\sigma(i) = j' \neq j$ et $\sigma(i') = j$:
  - Construisons une affectation γ à partir de σ via une inversion des valeurs en i et i', i.e. $\gamma(i) = j,\ \gamma(i') = j'$.
  - Comme $T_{i\gamma(i)} + T_{i'\gamma(i')} \geq T_{ij} > T_{i\sigma(i)} + T_{i'\sigma(i')}$, σ n'est pas maximale $\rightarrow$ contradiction, i.e. il n'est pas possible que $\sigma(i) \neq j$

Décompositions sans équivalent dans la méthode hongroise

- Le théorème précédent nécessite des conditions sur chaque ligne
  - Similaire aux cas de la méthode hongroise où on trouve k zéros mutuellement indépendants

Théorème

Si pour une ligne i, il y a une colonne j tel que $T_{ij} \geq T_{ij'} + T_{i'j}$
$\forall i \neq i', j \neq j'$, *il existe une affectation maximale telle que* $\sigma(i) = j$.
Si le nombre de lignes i qui ne satisfont pas cette condition est borné par $\sqrt[3]{n}$, *alors la distance peut être calculée en* $O(n)$.

- L'existence d'un σ maximal (pas forcement unique) avec $\sigma(i) = j$ est prouvée comme dans le théorème précédent
- Il suffit de parcourir les éléments clés pour marquer les lignes i et les colonnes j satisfaisant cette condition

Conclusions sur le calcul de distance

- La complexité de calcul peut être réduite de $O(n + k^2)$ à $O(n)$ dans certaines situations
  - Pour la coloration: si $n = 1000$ et $k \approx 250$, alors $1000 << 60000$
  - Ces cas sont souvent associées avec des distances petites (comme celles intra-cluster)
- Un algorithme linéaire pourrait servir dans plusieurs types d'applications utilisant des partitions

---

- Des bornes précises pour les valeurs de distance ont été calculées (ex. $n - \lceil \frac{n}{k} \rceil$) [Charon et al., Maximum Transfer Distance Between Partitions, 2006]
- Des comparaisons avec d'autres mesures de similarités (index de Rand) existent [Denoeud et. al, Comparison of distance indices between partitions, 2006]

Résumé

Conclusions et discussion

Deux directions de recherche complémentaires en rapport avec la classification:

1. l'apport des méthodes de classification en optimisation combinatoire

   [Porumbel et. al., Position Guided Tabu Search for Graph Coloring, Learning and Intelligent Optimisation, 2009]

   [Porumbel et. al., A Search Space Cartography for Guiding Graph Coloring Heuristics. Computers & Operations Research, 2010]

2. la comparaison de partitions via un calcul linéaire de la distance de transfert

   [Porumbel et. al., An Efficient Algorithm for Computing the Distance between Close Partitions, Discrete Applied Mathematics 2011]

Conclusion sur la classification en optimisation

- La classification nous a permis de découvrir une forme de structuration dans l'espace de recherche: l'hypothèse de clusterisation
- Cette hypothèse a été utilisée pour construire une stratégie d'optimisation mieux adaptée au problème.
- Des classifications plus complexes pourraient être appliquées mais on doit prendre en compte la contrainte du temps de calcul

Conclusion sur la Distance de Transfert entre Partitions

- La distance peut être calculée par un algorithme Las Vegas en $O(n)$ au lieu de $O(n + k^2)$
  - si certaines conditions sont satisfaites, il donne la réponse correcte; sinon, il prévient de l'échec
- La comparaison de partitions a donné lieu à de nombreux travaux dans la classification; elle a été introduite par Simon Régnier en 1965