

# Guided local search

## Recherche locale guidée

1

PRÉSENTATION BASÉE SUR L'ARTICLE  
*GUIDED LOCAL SEARCH* DE  
CHRITOS VOUDOURIS, EDWARD P.K TSANG,  
ABDULLAH ALSHEDDY

# Guided local search et Fast Local Search

## Vue d'ensemble

2

- Guided Local Search (GLS) : métaheuristique qui s'appuie sur un jeu de pénalisation, qui combinée avec une recherche locale, vise à améliorer l'efficacité et la robustesse de la résolution de problème de grande taille
- Fast Local Search (FLS) : une recherche locale qui réduit la taille du voisinage en attribuant une étiquette à des voisinages selon leur capacité à améliorer l'objectif.

# Introduction

3

- Recherche locale (rappel)
  1.  $N(s)$  := voisinage de la solution  $s$
  2. Déterminer la meilleure solution  $s'$  dans  $N(s)$
  3. Si  $f(s') < f(s)$  alors  $s' \leftarrow s$  Retourner en 1  
sinon STOP
- Techniques pour échapper aux minima locaux (Tabou, Simulated Annealing(SA), GLS)
- Particularité de GLS
  - Répartit l'effort de recherche dans l'espace de solutions, de façon à favoriser les zones prometteuses.

# Plan

4

1. Description de la recherche locale guidée (GLS)
  - Définitions et notations
  - Pseudo-code
2. Guided fast local search (GFLS)
  - Fast local search
  - GFLS pseudo-code
3. GLS et autres métaheuristiques
4. Quelques applications de GLS et performances
5. Gros plan sur l'application de la GLS au PVC

# 1. Description de GLS

5

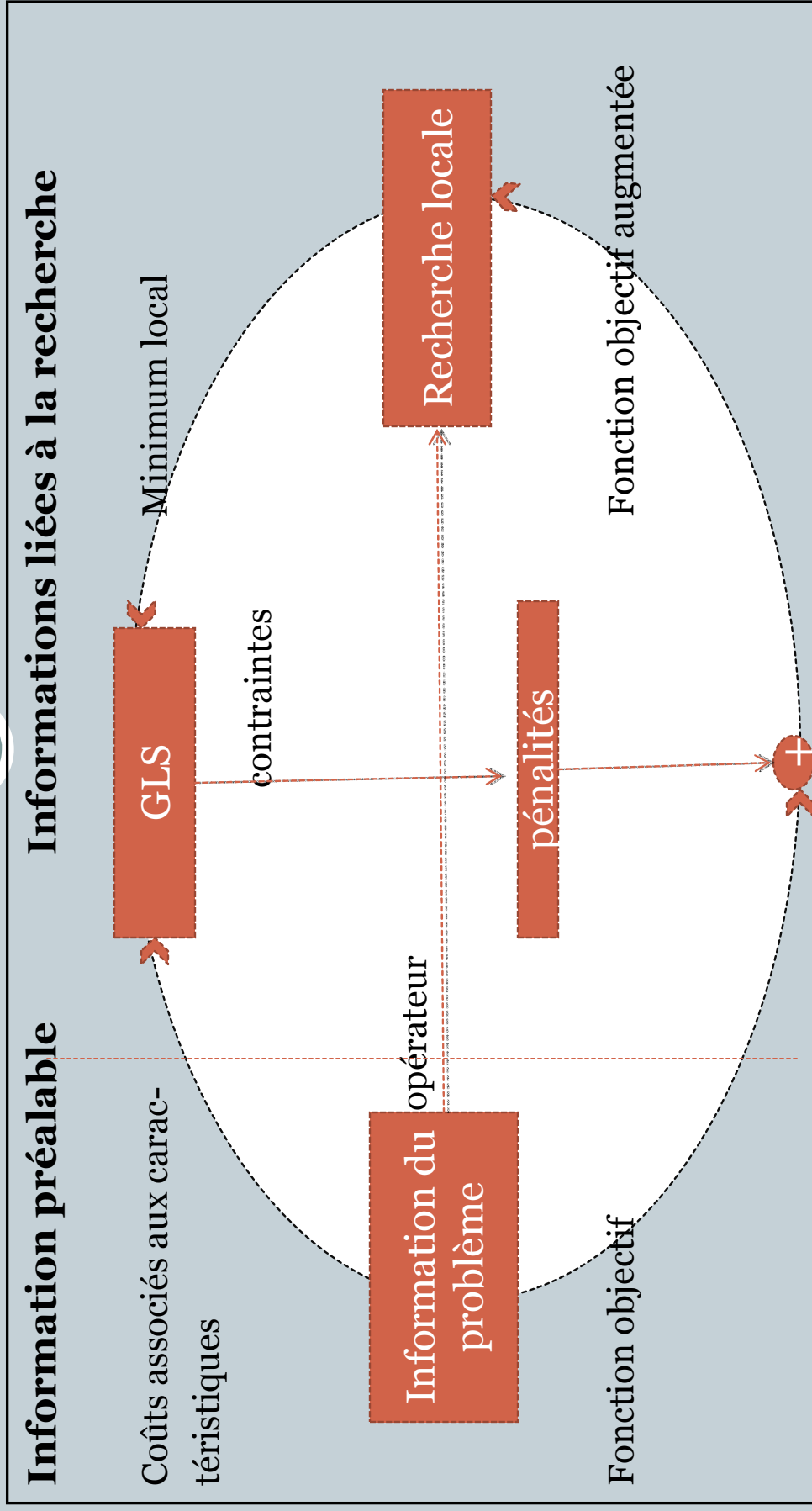
## Principes de GLS

- Définition d'un ensemble de caractéristiques pour les solutions candidates.
- A chaque minimum local, certaines caractéristiques sont sélectionnées et pénalisées
- Utilisation d'une fonction objectif augmentée par l'accumulation des pénalités.

➔ Nouveauté de GLS : la méthode de sélection des caractéristiques à pénaliser.

# Vue d'ensemble de l'utilisation de la GLS (Voudouris, Tsang 1995)

6



# GLS : qu'est ce qu'une caractéristique?

7

- Une caractéristique doit être définie de façon à ce qu'il soit possible de déterminer si une solution quelconque possède cette caractéristique ou non.
- Une caractéristique est souvent choisie pour son impact direct sur la fonction objectif.
- Toute propriété d'une solution peut être alors interprétée et utilisée pour guider la LS.

# GLS : Exemples de caractéristiques

8

- Problèmes de tournées et de planification  
**Arêtes**
- Problèmes d'affectation  
**Affectation d'un objet à l'autre**
- Optimisation de problèmes sous contraintes  
**Contraintes**
- GLS identifie et pénalise facilement les arêtes couteuses , menant LS vers une « bonne » solution contenant le plus de courtes arêtes possible.
- GLS ne mène pas nécessairement à de bonnes solutions pour des certaines variantes du problème, mais fortifie la diversification de la LS tenant compte de la « penalty memory ». Résultats comparables à d'autres méthodes de résolution.



# Cahier des charges pour appliquer la GLS

9

- Les caractéristiques
- Une fonction objectif  $g$
- Les pénalités  
Initialement à 0. Elles augmentent quand la LS atteint un minimum local.
- Fonction objectif augmentée:

# GLS : Comment sortir d'un min local?

10

- Augmentation de la fonction de coûts par ajout des pénalités sur les caractéristiques choisies
- Sont pénalisées les caractéristiques défavorables ou celles qui coûtent le plus.
  - Autre facteur pris en compte: la pénalité courante d'une caractéristique

# Comment choisir la ou les caractéristiques à pénaliser?

11

Coût associé à la caractéristique  $i$

Pénalité associée à la caractéristique  $i$

$$util_i(s^*) = I_i(s^*) * \frac{c_i}{1 + p_i}$$

Variation de util:

$c_i \uparrow$  alors  $util_i \uparrow$   $p_i \uparrow$  alors  $util_i \downarrow$

Une caractéristique  $j$  est pénalisée si :

$$j = \arg \max_i (util_i(s^*))$$

$$p_j \leftarrow p_{j+1}$$

# GLS : Pseudo-code

Caractéristiques

12

GuidedLocalSearch(p, g,  $\lambda$ , [ $I_1, \dots, I_M$ ], [ $c_1, \dots, c_M$ ]) {

$k \leftarrow 0$ ,  $s_0$  solution initiale

$p_i \leftarrow 0 \quad i=1, \dots, M$   $h \leftarrow g + \lambda \sum_i p_i * I_i$

Génération aléatoire,  
heuristique de construction

Tant que le critère d'arrêt n'est pas vérifié

$s_{k+1} \leftarrow$  MethodeAmelioration ( $s_k, h$ )

$util_i(s_{k+1}) = I_i(s_{k+1}) * \frac{c_i}{1 + p_i} \quad i=1, \dots, M$

LS, VNS, VDS,

$p_j \leftarrow p_{j+1}$  pour tout  $j = \arg \max_i (util_i(s^*))$   
 $k \leftarrow k+1$

retourner  $s^* \leftarrow$  meilleure solution obtenue pour g

}

# Description de GLS

13

- Le paramètre  $\lambda$ :

$$\lambda = \frac{\alpha * g(x_*)}{\#\{\text{caractéristiques dans } x_*\}}$$

Minimum local initial

- Évaluation d'une modification:

$$\Delta g + \sum_{j \in JJ} p_j - \sum_{i \in JJ} p_i$$

Où

JJ est l'ensemble des caractéristiques présentes dans la solution courante qui ne sont pas la solution candidate.

JJ contient les caractéristiques qui sont dans la solution candidate, non commune à la solution courante.

# GLS : Paramètre $\lambda$

14

Plus une caractéristique est couteuse, plus GLS travaillera à la supprimer de la solution.

L'effort consacré dépend du paramètre  $\lambda$ :

❑  $\lambda$  grand  $\longleftrightarrow$  Forte diversification/faible intensification

❑  $\lambda$  faible  $\longleftrightarrow$  Le processus de diversification est ralenti. Grande difficulté à échapper à un minimum local .

➔ Recherche d'un intervalle d'équilibre auquel  $\lambda$  pourrait appartenir

# GLS : Critère d'arrêt

15

- Critère d'arrêt (CritereStop)
  - Longueur du processus de la recherche (ex. nombre de modifications effectuées)
  - Durée CPU de l'algorithme
  - Si une borne inférieure est connue, le critère d'arrêt est le gap min à atteindre entre la meilleure solution obtenue et cette borne inférieure.

## 2. Guided Fast Local Search

# Principes de la Fast Local Search

16

Motivation: Réduire le nombre de voisinages considérés pour accélérer la recherche, sans perdre en qualité de solution.

→ Utilisation d'heuristiques pour augmenter l'efficacité de la recherche:

Il s'agit d'identifier et ignorer des voisinages qui ne mèneront pas à des mouvements améliorant .



# FLS : Voisinage et sous-voisinages

17

Division du voisinage en sous-voisinages.

- Sous-voisinage  $\Leftrightarrow$  Bit d'activation
- Un sous-voisinage est dit actif si le bit d'activation associé est égal à 1.
- Si le bit d'activation est nul, le sous-voisinage associé est dit inactif. Aucune recherche n'a alors lieu sur cet ensemble.

## FLS: Activation des sous-voisinages au cours d'une recherche locale

18

- Initialement, tous les sous-voisinages sont actifs
- Si un sous-voisinage est examiné, et qu'aucun mouvement améliorant l'objectif n'y est trouvé: il devient inactif.
- Plus le processus évolue, moins il y a de sous-voisinages actifs.
- Le processus s'arrête quand tous les sous-voisinages sont inactifs.

# GFLS = GLS + FLS Idée principale

19

- A une caractéristique est associée un sous-voisinage.
- Les associations sont faites de telle façon que pour chaque caractéristique on connaît quels sous-voisinages contiennent des mouvements qui ont effet immédiat sur l'état d'une caractéristique (ie mouvement qui supprime cette caractéristique de la solution courante)

# FLS Pseudo-code (1/2)

Objectif augmenté

Bits d'activation

Nombre de sous-voisinages

```

procedure FastLocalSeach( $s, h, [bit_1, \dots, bit_L], L$ )
begin
  while  $\exists bit, bit = 1$  do
    /* i.e. while active sub-neighbourhood exists */
    for  $i \leftarrow 1$  until  $L$  do
      begin
        if  $bit_i = 1$  then
          /* search sub-neighbourhood  $i$  */
          begin
            Moves  $\leftarrow$  MovesForSubneighbourhood( $i$ );
            for each move  $m$  in Moves do
              begin
                /* minimization case is assumed here */
                move  $m$ 
              end
            end
          end
        /* spread activation */
      end
    end
  end

```

Retourne l'ensemble des mouvements  
contenus dans le sous-voisinage

## FLS Pseudo code (2/2)

21

Ensemble des sous-voisinages à activer, m fait

```

ActivateSet ←
  SubneighbourhoodsForMove(m);
for each sub-neighbourhood  $j$  in
  ActivateSet do
     $bit_j \leftarrow 1$ ;
     $s \leftarrow s'$ ;
    goto ImprovingMoveFound
  end
end
 $bit_i \leftarrow 0$ ; /* no improving move found */
end
ImprovingMoveFound:
  continue;
end
return  $s$ ;
end
```

# GFLS Pseudo-code

22

```
procedure GuidedFastLocalSearch( $p, g, \lambda, [I_1, \dots, I_M], [c_1, \dots, c_M], M, L$ )  
begin  
   $k \leftarrow 0$ ;  
   $s_0 \leftarrow \text{ConstructionMethod}(p)$ ;  
  /* set all penalties to 0 */  
  for  $i \leftarrow 1$  until  $M$  do  
     $p_i \leftarrow 0$ ;  
    /* set all sub-neighbourhoods to the active state */  
    for  $i \leftarrow 1$  until  $L$  do  
       $bit_i \leftarrow 1$ ;  
    /* define the augmented objective function */  
     $h \leftarrow g + \lambda * \sum p_i * I_i$ ;
```

Initialisation

# GFLS Pseudo-code(2/2)

23

```
while StoppingCriterion do  
  begin
```

```
     $s_{k+1} \leftarrow \text{FastLocalSearch}(s_k, h, [bit_1, \dots, bit_L], L);$ 
```

détermine les sous-voisins à activer en cherchant les mouvements qui suppriment des caractéristiques de la solution courante qui sont pénalisées

```
  begin
```

```
     $p_i \leftarrow p_i + 1;$ 
```

```
    /* activate sub-neighbourhoods related  
    to penalized feature  $i^*$  */
```

```
     $\text{ActivateSet} \leftarrow \text{SubneighbourhoodsForFeature}(i);$ 
```

```
    for each sub-neighbourhood  $j$  in  $\text{ActivateSet}$  do
```

```
       $bit_j \leftarrow 1;$ 
```

```
    end
```

```
     $k \leftarrow k + 1;$ 
```

```
  end
```

```
   $s^* \leftarrow$  best solution found with respect to objective function  $g;$ 
```

```
  return  $s^*;$ 
```

nouveauté

### 3. GLS et autres métaheuristiques

24

- a) GLS et Recherche tabou (TS)
- b) GLS ET algorithmes génétiques (GA)
- c) GLS Hybrides



# GLS et Recherche tabou(TS) (1/2)

25

Beaucoup de points communs entre les deux méthodes

➔ « GLS variante de TS »

- Pénalités <-> « soft » tabou
- Les 2 méthodes imposent des conditions sur ces caractéristiques(attributs) pour guider la recherche.

# GLS et Recherche tabou(TS) (2/2)

26

- (TS) tous les mécanismes associés sont enclenchés à chaque itération, et le recherche locale ne peut rester à un minimum local.  
  
(GLS) la procédure s'enclenche quand la recherche locale est coincée en un minimum local, et s'achève quand celle-ci en sort.
- (TS) Mémoire à court ou long terme  
  
(GLS) La mise à jour des pénalités visent à satisfaire ces 2 stratégies.

- GLS et GA peuvent interagir comme GLS et une LS  
→ Guided Genetic Algorithm (GGA)
- Objectifs principaux visés par cette association:  
Étendre les applications des 2 méta-heuristiques  
Améliorer la robustesse de GLS

# Guided Genetic Algorithm (GGA)

28

- GGA ~ GA utilisant GLS pour sortir d'un minimum local
  - Si aucune amélioration après un nombre prédéterminé d'itérations, GLS modifie la mesure d'évaluation par une mise à jour des pénalités via

$$util_i(s^*) = I_i(s^*) * \frac{c_i}{1 + p_i}$$

- Cette nouvelle mesure de performance est utilisée par GA.
- GLS intervient aussi par les pénalités sur la reproduction et la mutation

# GLS Hybrides

29

- GLS + Evolutionnary Strategies (ES)
- GLS + Variable Neighbourhood Search (VNS) = GVNS
- GTS = Guided Tabu Search [Tarantilis et al]  
-> VRP avec flotte hétérogène
- GLS+ Ant Colony Optimization (ACO) [Hani et al] -> Quadratic Assignment Problem

## 4. Quelques applications de GLS et performances

30

- Workforce Scheduling Problem(WSP)  
GLS + FLS → a fourni de meilleurs résultats que les travaux publiés avant 1997
- TSP  
GLS + FLS + 2-opt vs Lin-Kernighan (LK) algorithm  
En moyenne, GLS trouve de meilleurs résultats sur une même durée d'exécution que LK.

 Gros travail d'implantation

Avantage de GLS: Ne s'applique pas qu'au TSP

## 4. Quelques applications de GLS et performances

31

- **VRP**

- Kilby, Prosser, Shaw : VRPTW

A donné de très bons résultats.

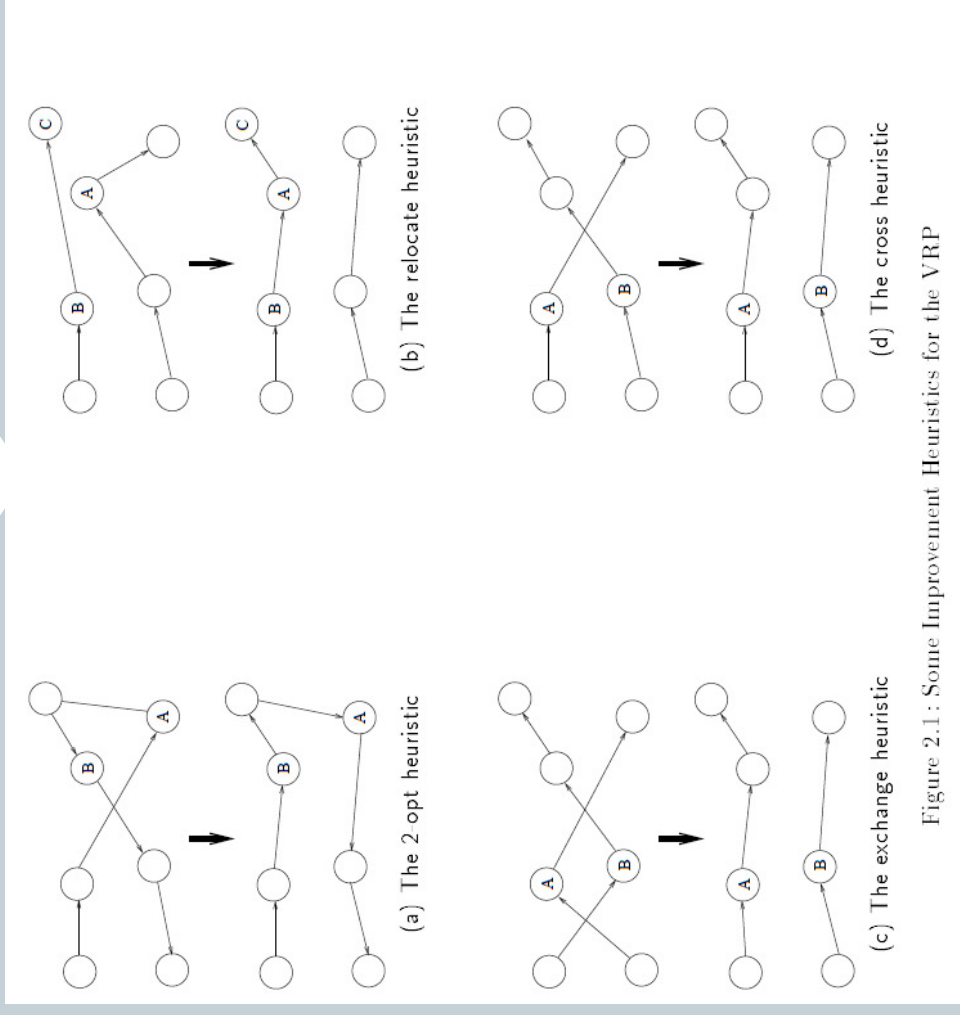
A été incorporé au package Dispatcher développé par ILOG

- ✦ Quelques remarques sur la méthode:

- Pas d'heuristiques de construction pour générer la solution initiale. Aucun client livré est la solution initiale (non livraison étant pénalisée)
- Heuristiques d'amélioration utilisées par la LS
- Caractéristiques = arcs (à cause des fenêtres de temps)

# GLS appliquée au VRPTW: Heuristiques d'amélioration utilisées par la LS

32





# GLS et VRP

33

- Guided Tabu Search appliquée au VRP avec flotte hétérogène (VRPHF) [Tarantilis, Zachariadis, Kiranoudis (2007)]
- GTS appliquée au VRP avec cueillette et livraison (VRPSDP) [Tarantilis, Zachariadis, Kiranoudis (2008)]
- GLS + ES utilisée pour résoudre des VRPTW de grande taille [Mester, Brysy. (2005)]

## 5. Gros plan sur GLS appliquée au PVC

34

N villes

$D = [d_{ij}]$  matrice des distances entre villes

$P = [p_{ij}]$  matrice des pénalités

$\Pi$  permutation cyclique sur les N villes

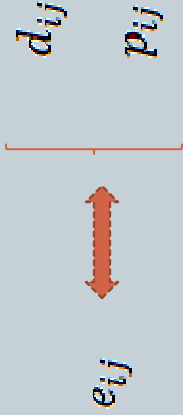
$\Pi(i)$  = la ville visitée après i

Fonction objectif:

$$g(\pi) = \sum_{i=1}^N d_{i\pi(i)}$$

# GLS appliquée au PVC

35

- caractéristique = arête d'un tour  $e_{ij}$
- Arête  

- Matrice auxiliaire D' :  
 $D' = D + \lambda * P$  sert à évaluer les modifications effectuées à la solution courante.

GLS modifie P dès qu'un minimum local est atteint

# GLS appliquée au PVC

36

- Méthode d'amélioration :

Analogie à la méthode 2-Opt utilisée lors d'une recherche locale appliquée au PVC.

Notons  $e_1, e_2$  les arêtes supprimées du tour,  $e_3$  et  $e_4$  les nouvelles arêtes introduites dans le tour.

$d_1, d_2, d_3$  et  $d_4$ , les coûts (distances) associées.

La variation de l'objectif (2-opt) est:

$$d_3 + d_4 - d_1 - d_2 + \lambda^*(p_3 + p_4 - p_1 - p_2)$$

# GLS appliquée au PVC

37

- Définition du paramètre :

Pour obtenir de bonnes solutions:  $1/8 \leq \alpha \leq 1/2$

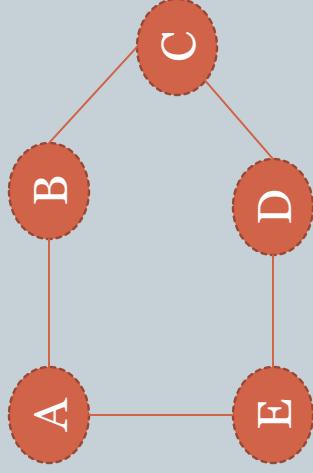
$$\lambda = \frac{\alpha * g(x_*)}{\#\{\textit{caractéristiques dans } x_*\}}$$

**Rappel :** *Le voisinage d'un tour  $t$  est l'ensemble des tours  $t'$  obtenus de  $t$  après suppression de 2 arêtes et introduction de 2 autres pour créer un nouveau tour.*

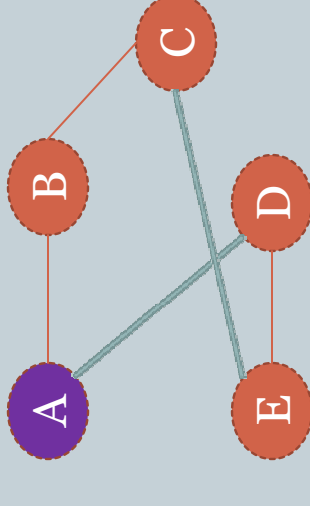
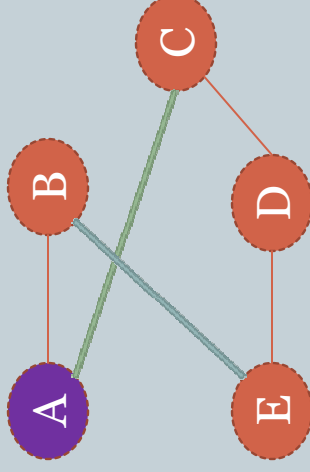
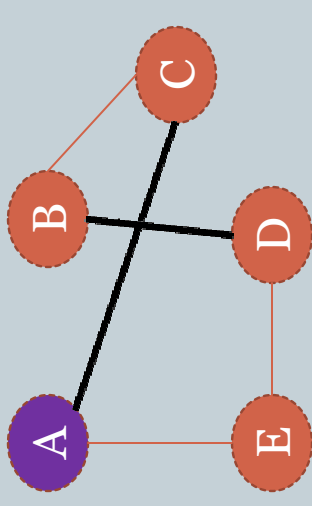
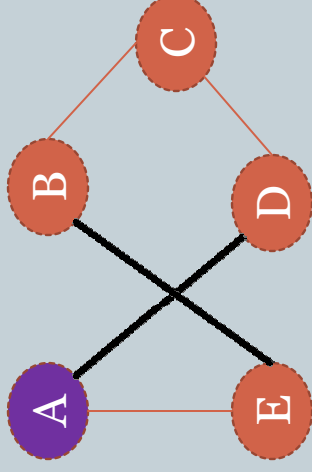
- Partition du voisinage en  $N$  sous-voisinages.  
A chaque ville est associé un sous-voisinage, qui contient tout tour issu d'un 2-opt sur le tour courant, et dont au moins une des arêtes supprimées est adjacente à la ville.
- Activation d'un sous-voisinage si
  - la ville à laquelle il est associé, est aux extrémités des arêtes supprimées et ajoutées.

# Exemple de sous-voisinage

39



Solution courante



Sous-voisinage associé à la ville A  
Est activé si AE ou AB est pénalisée.

# Références supplémentaires

40

- Kilby, P., Prosser, P., and Shaw, P. (1999) Guided local search for the vehicle routing problem with time windows, in Voss, S., Martello, S., Osman, I.H., and Roucairol, C. (eds.), *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Kluwer Academic Publishers, 473–486.
- Lau, T.L. and Tsang, E.P.K. (1998) The guided genetic algorithm and its application to the general assignment problem. *IEEE 10th International Conference on Tools with Artificial Intelligence (ICTAI'98)*, Taiwan, 336–343.
- Voudouris, C. and Tsang, E.P.K. (1999) Guided Local Search and its application to the Travelling Salesman Problem. *European Journal of Operational Research* 113:2, 469–499.