CHAPITRE 2: LES MÉTAHEURISTIQUES À BASE DE SOLUTION UNIQUE, MÉTHODES DE TRAJECTOIRE

COURS : MÉTAHEURISTIQUES ET ALGORITHMES ÉVOLUTIONNISTES

RÉALISÉ DR. S. SLATNIA

UNIVERSITÉ DE BISKRA
Master d'informatique
2020-2021

PLAN DU CHAPITRE

1. **INTRODUCTION**
2. **ALGORITHME DE RECHERCHE LOCALE (Local Search)**
3. **ALGORITHME DE RECHERCHE AVEC TABOU (Tabu Search)**
4. **ALGORITHME DE RECHERCHE RECUIT SIMULÉ (Simulated Annealing)**

INTRODUCTION (1/3) [OC]

Métaheuristiques simples

I. Méthodes constructives

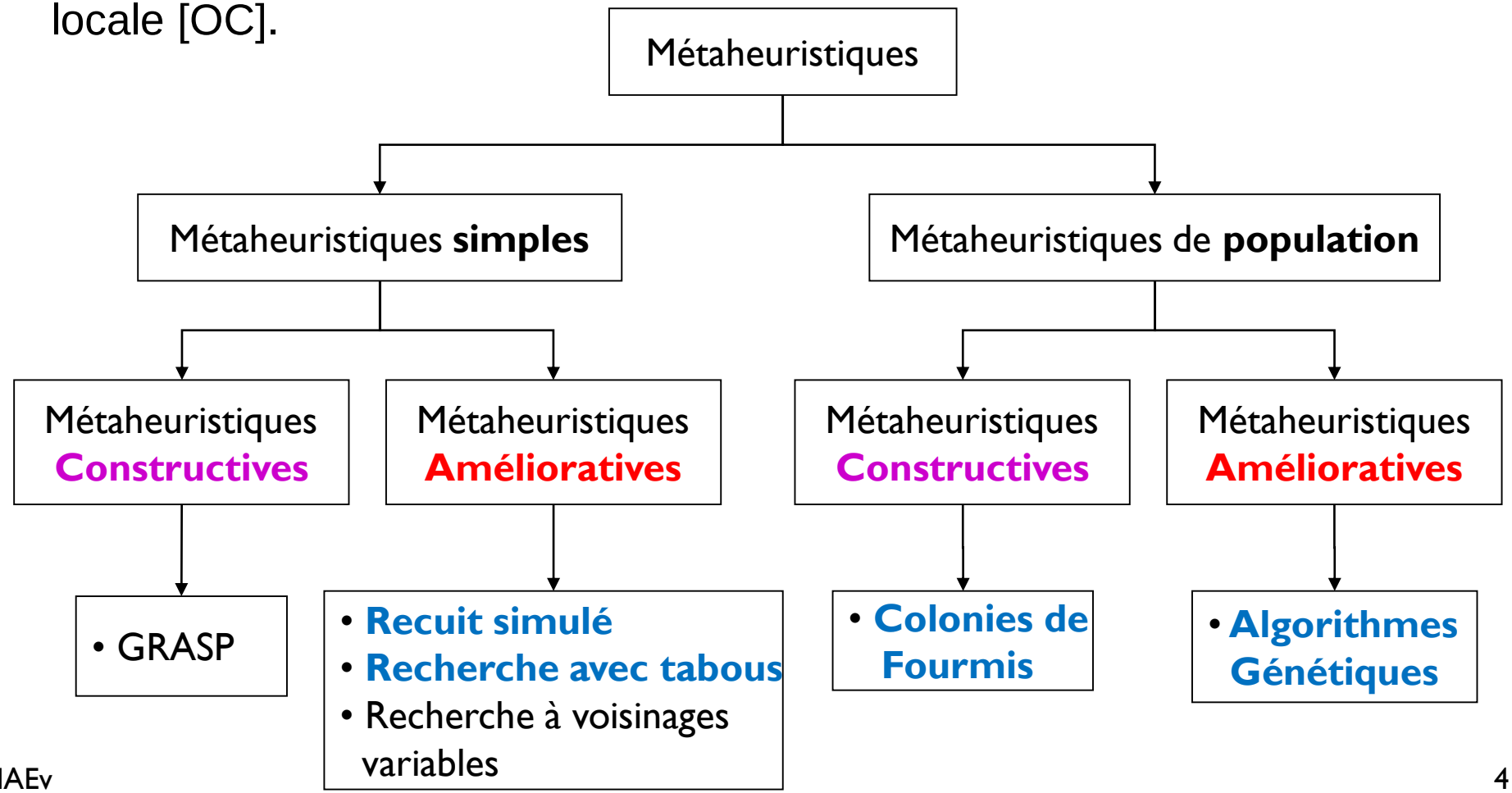
- exemple : méthode du plus proche voisin pour le PVC
  - ➢ très rapide
  - ➢ qualité des solutions laisse à désirer

II. Méthodes amélioratives

- Recherche locale
- Recherche tabou
- Recuit simulé

INTRODUCTION (2/3) [OC]

- Heuristiques qui permettent de surmonter l'obstacle de l'optimalité locale [OC].



I. ALGORITHME DE RECHERCHE LOCALE (RL)

DÉFINITIONS [Phill]

- **une solution** est une affectation de toutes les variables du problème.
- **une solution optimale** est une solution de coût minimal
- **un mouvement** est une opération élémentaire permettant de passer d'une solution à une solution voisine.
- **le voisinage** d'une solution est l'ensemble des solutions voisines, c'est à dire l'ensemble des solutions accessibles par un mouvement
  - ✓ un seul.
- **un essai** est une succession de mouvements.
- **une recherche locale** est une succession d'essais.

PRINCIPE DE LA RL [Phill]

- Partir d'une solution sinon approchée du moins potentiellement bonne et d'essayer de l'améliorer itérativement.
  - Pour **améliorer une solution** on ne fait que de **légers changements**
    - ➡ On parle de **changement local** (solution voisine).

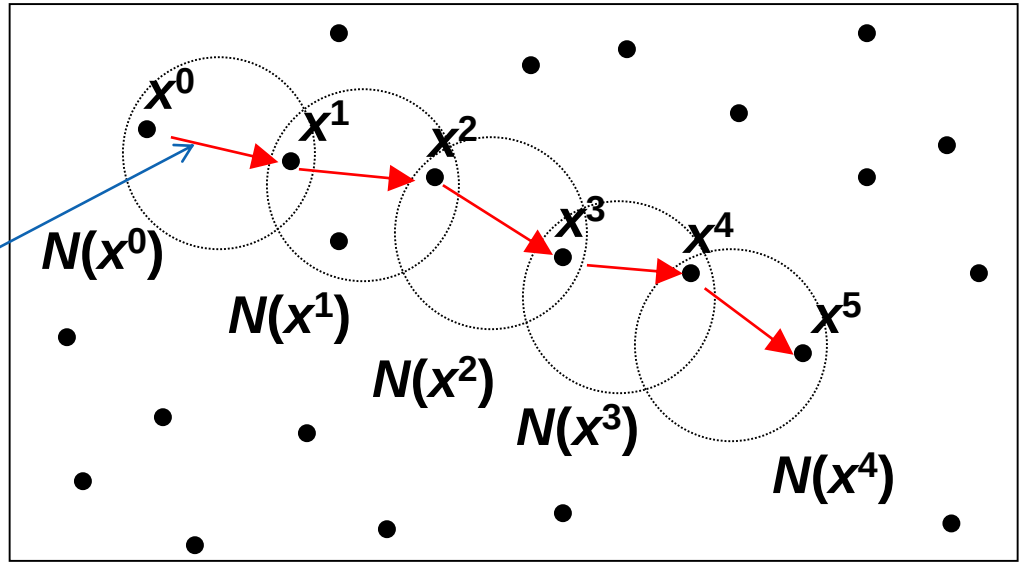
- **Relancer** la méthode **plusieurs** fois en changeant le point de départ pour avoir plus de couverture.

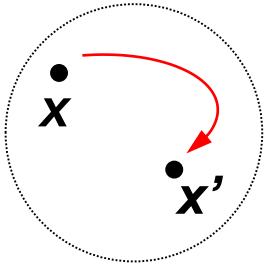
MÉTHODE DE RL (1/3) [OC]

X : ensemble des solutions

N(x) : ensemble de tous les voisins de x
x' : solution voisine de x

Trajectoire de recherche



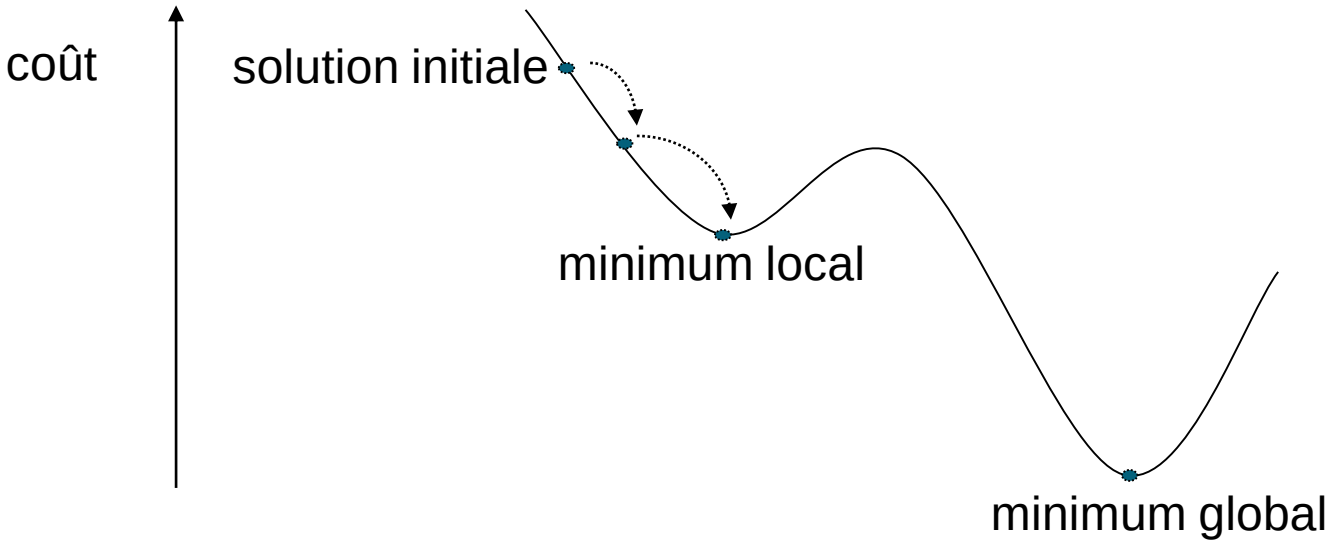


- passage d'une solution x ∈ X à une solution x' ∈ N(x)

  F(x') = min F(x''), x'' ∈ N(x)

MÉTHODE DE RL (2/3) [OC]

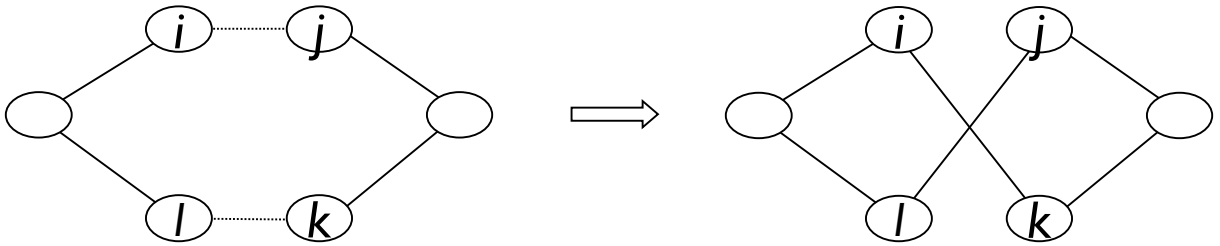
- L'insuffisance des méthodes de recherche locale
  - incapables de progresser au-delà du premier optimum local rencontré



Problème de minimisation

MÉTHODE DE RL (3/3) [OC]

- **Exemple** : méthode d'échange 2-opt pour le PVC
  - retirer 2 arêtes (*i*, *j*) et (*k*, *l*) et les remplacer par les arêtes (*i*, *k*) et (*j*, *l*)



- **AV** : plus puissantes que les méthodes constructives
- **INC** : plus coûteuses en termes de ressources informatiques

ALGORITHME DE RL[Phill]

A* ← creer une solution()
for all t=1 a max essais **do**
    A ← nouvelle solution()
    for all m=1 a max mouvements **do**
        A' ← choisir voisin(A)
        d ← (f(A')-f(A))
        if acceptable(d) **then** A ← A'
    end for
    if f(A*)>f(A) **then** A* ← A
end for
retourner A*,f(A*)
f : fonction à optimiser

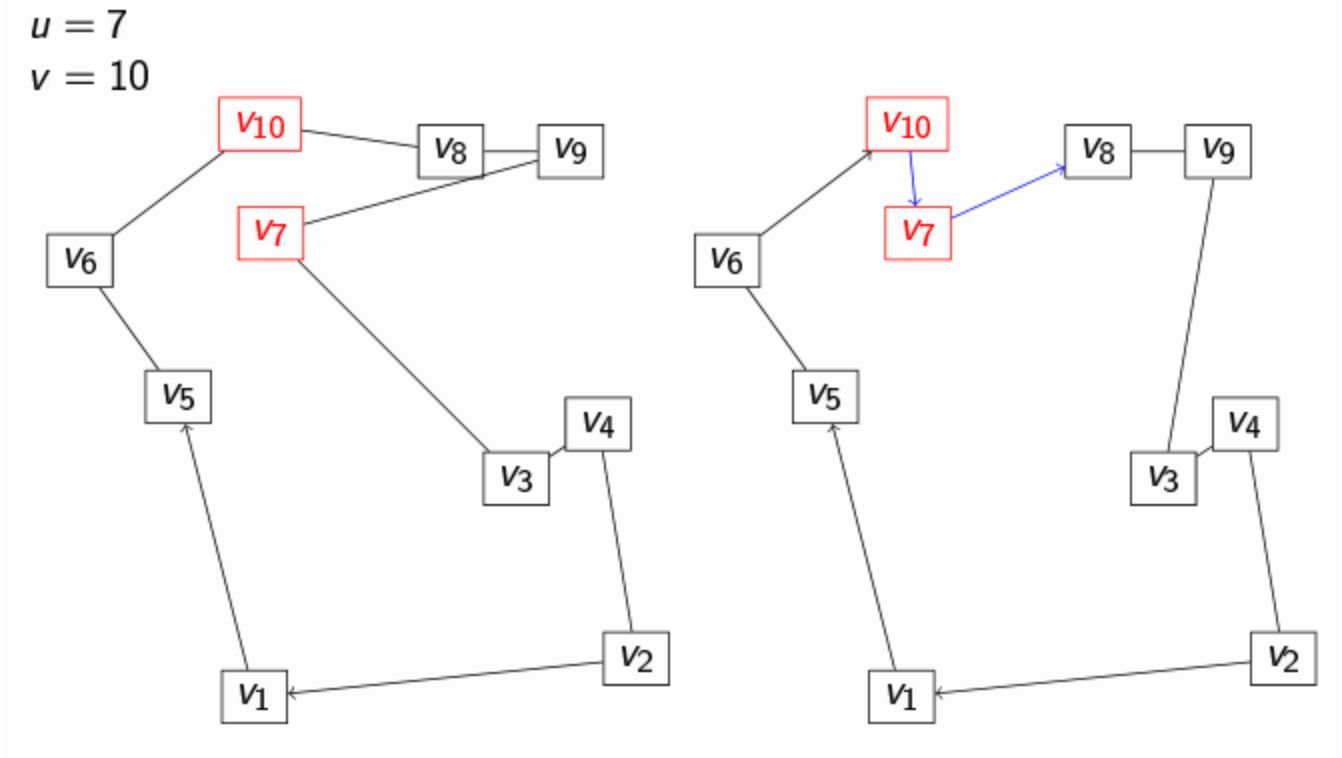
LES PARAMETRES À RÉGLER[Phill]

- **max essais** : le nombre d'essai à réaliser
- **max mouvements** : ...
- **créer une solution** : génère une solution (aléatoirement ou non)
- **nouvelle solution :** même chose
- **choisir un voisin** : choisit dans le voisinage de la solution courante (c'est généralement ce qui caractérise principalement une méthode de recherche locale).
- **acceptable** : accepte ou pas la nouvelle solution générée.

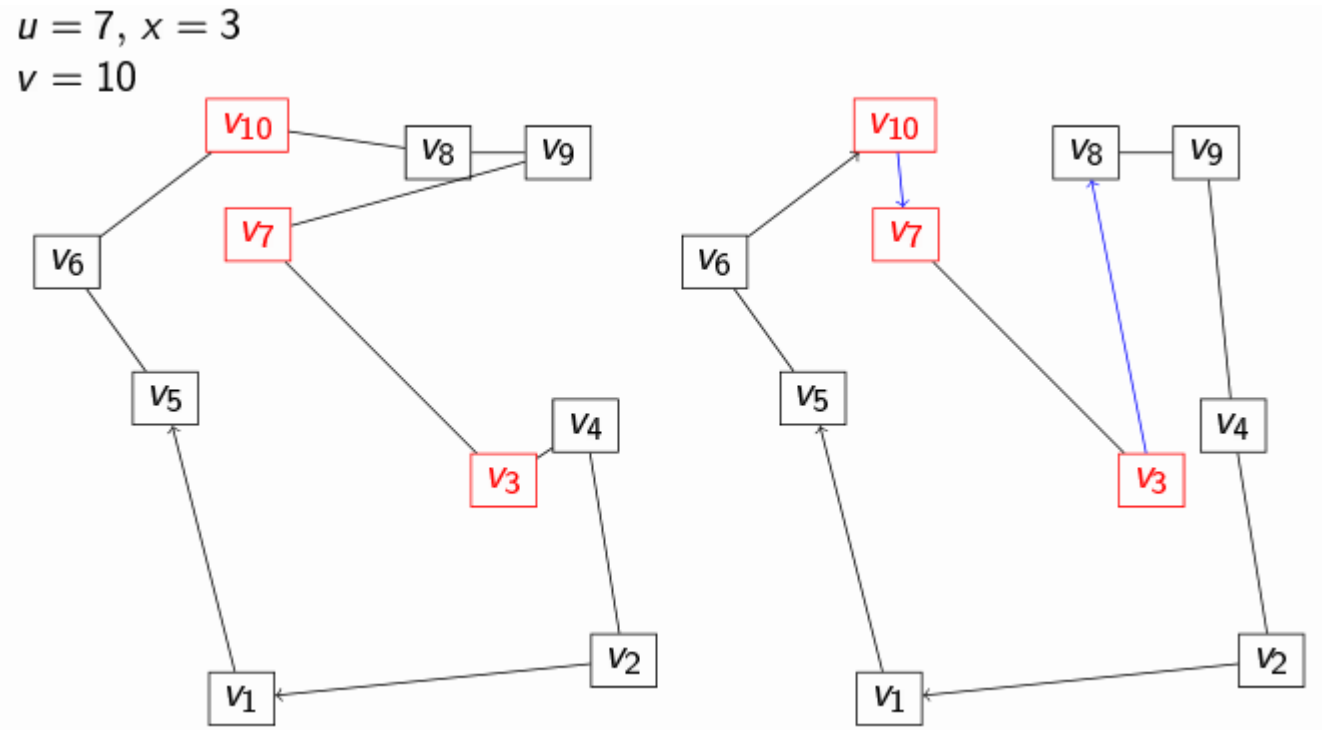
Exemple de voisinages pour le TSP[Ben]

- Soit u et v deux villes, x et y leurs villes consécutives respectives dans la solution d'un TSP.
- **Opérateurs de voisinage:**
  - **Insertion**
    - ✓ supprimer u et l'insérer après v
    - ✓ supprimer u et x, et insérer (u,x) après v
    - ✓ supprimer u et x, et insérer (x,u) après v
  - **Échange** (Swap)
    - ✓ échange u et v
    - ✓ échange(u,x) et v
    - ✓ échange(u,x) et (v,y)
  - **2-Opt**
    - ✓ remplacer (u,x) et (v,y) par (u,v) et (x,y)

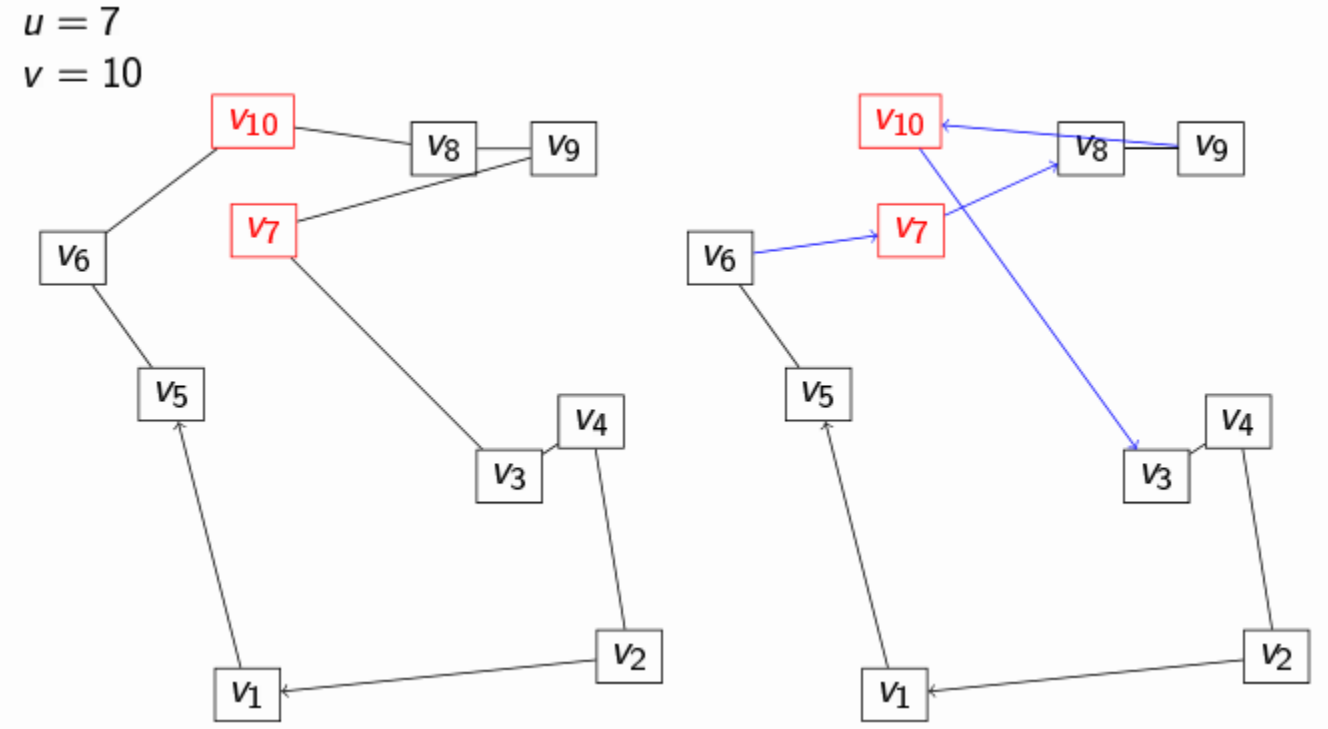
Insertion: illustration 1 [Ben]



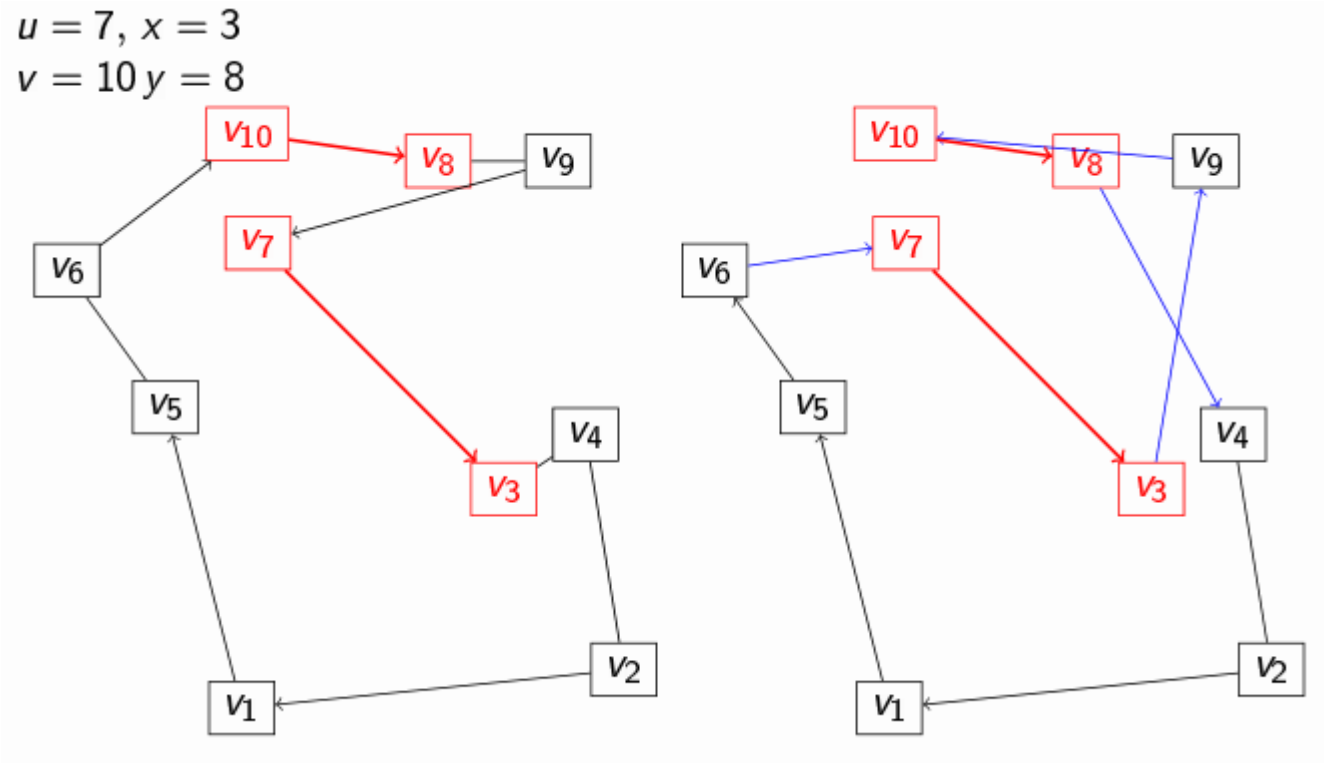
Insertion: illustration 2 [Ben]



Échange : illustration[Ben]



2-Opt : illustration[Ben]

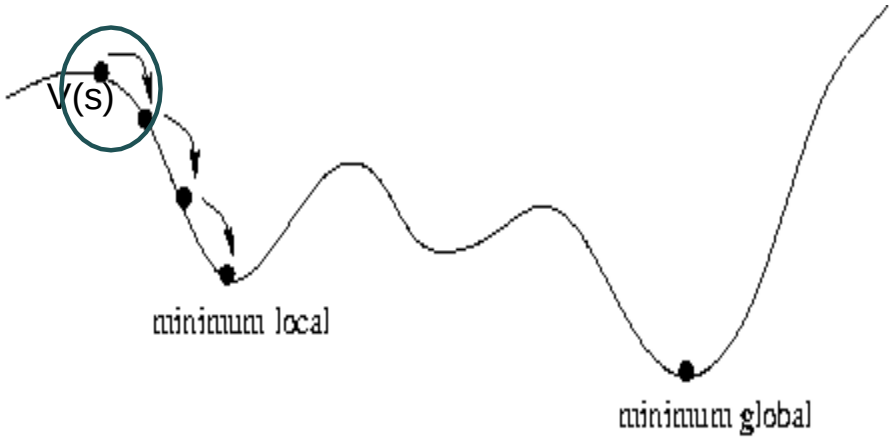


II. ALGORITHME DE RECHERCHE AVEC TABOU (TABU SEARCH)

INTRODUCTION : LA MÉTHODE DEDESCENTE

[HAM ET ALL]

A chaque itération, la meilleure solution s ' du Voisinage de s (telle que f(s')<f(s))est sélectionnée.



insuffisance et solutions

- **Taille du voisinage** → **exploration partielle**
- **Blocage en optima locaux** → **autorisation de l'acceptation de mouvements de dégradation**

La recherche tabou est une correction de la méthode de descente mise en place pour sortir des minima locaux

DÉFINITION DE BASE DE LA **RT**

- Développée par **Glover** et Hansen en **1986.**

Définition

La recherche tabou (TS) est une méthode de recherche locale combinée avec un ensemble de techniques permettant **d'éviter** d'être **piégé** dans un **minimum local** ou la **répétition d'un cycle** **[Sidi et all].**

- **H**euristique généralisation de recherche locale traditionnelles

But : échapper aux optima locaux
Principe : Introduction d'une notion de mémoire dans la stratégie d'exploration

- ✓ **Interdiction de reprendre des solutions déjà (ou récemment) rencontrées[sebastien]**

PRINCIPE DE BASE DE LA **RT**

Principe de base [JoMa] :

Poursuivre la recherche de solutions même lorsqu'un optimum local est rencontré et ce,

1. En permettant des déplacements qui n'améliorent pas la solution
   (Acceptation de solutions moins bonnes que la solution courante)

   ➡ **Problème: Retours en arrière (**mouvements cycliques**)**

2. En utilisant l'histoire de la recherche [Mich]
   (mémoriser le cheminement récent de la recherche)

   ➡ **Liste de tabou : solutions interdites pour un nombre d'itérations, interdire le retour vers des solutions déjà visitées**

3. Critères d'aspiration
   Conditions permettant de lever le statut tabou d'une solution
   - Lorsque la solution est la meilleure jamais rencontrée.

MÉCANISME DE **RT**

❖ **Mémoire** :

Elle est représentée par une **liste taboue** qui contient des **mouvements** ou des **solutions** qui sont temporairement **interdits** [JoMa].

❖ **2 alternatives**

- Une liste contient les solutions interdites (coûteux en place mémoire).

→ **Liste tabou des mouvements interdits:** qui ramènent vers ces solutions déjà visitées.

- **Avantages** :
  - prend moins de place mémoire.
  - Plus efficace que la liste des solutions taboues mais

- **Problème :**
  - élimine plus de solutions que celles visitées effectivement
  - élimine éventuellement de très bonnes solutions.

MÉCANISME DE **RT**

- **Exception aux interdictions**

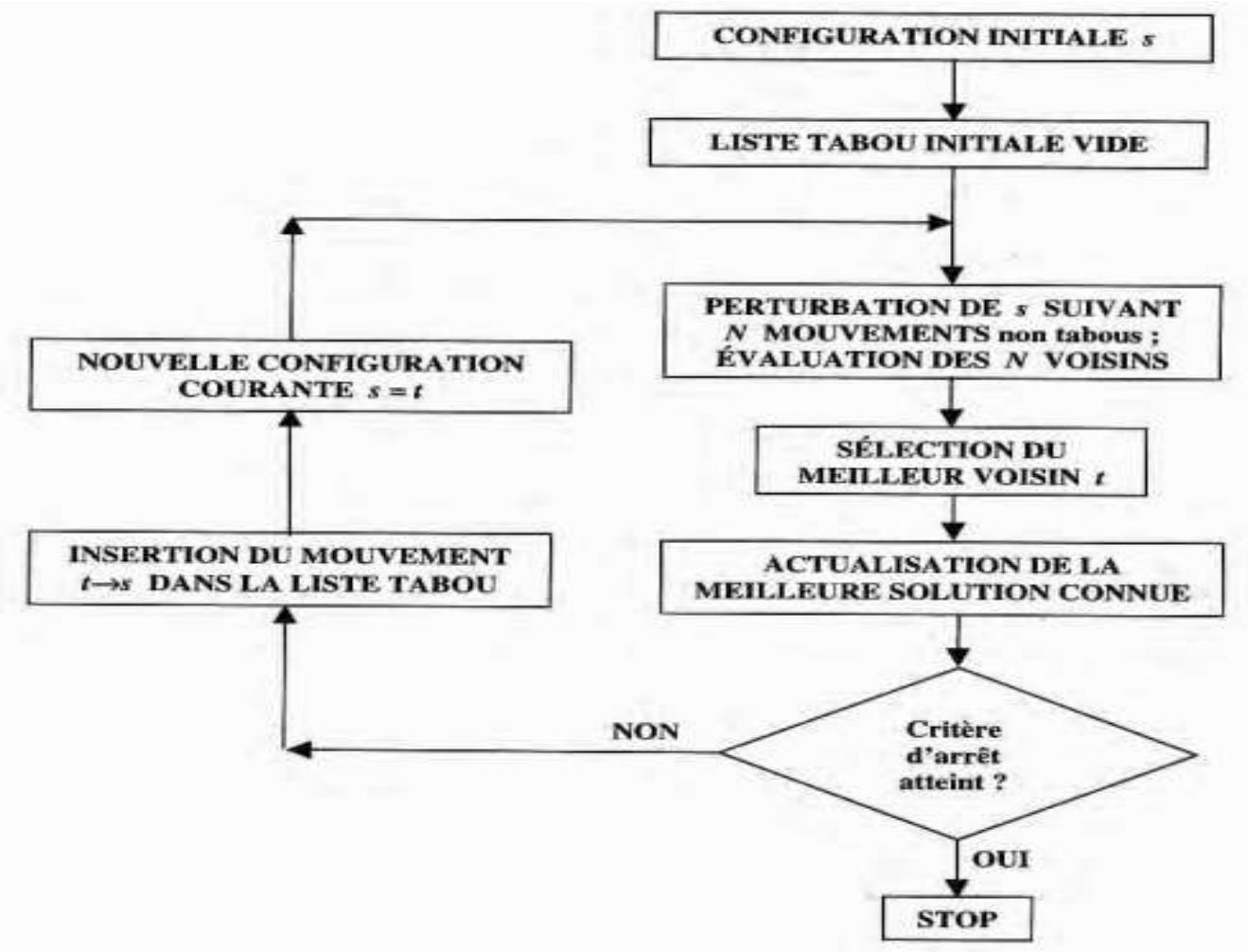
  il est possible de **violer** une interdiction lorsqu'un mouvement interdit permet d'obtenir la meilleure solution enregistrée jusqu'à maintenant [JoMa].

Une fonction d'aspiration qui permet de lever le status tabou.
(l'execpetion au interdiction).

NB.

- Généralement les listes sont gérées en **FIFO** (first in first out) [Ham et all].
- La méthode Tabou a de bonnes performances utilisée pour résoudre des problèmes complexes et/ou de très grande taille (**NP-durs**).
- Il a montré qu'une liste Tabou de taille #T=7,...,20 suffit pour obtenir de bons résultats **[Jean].**

PROCESSUS D'ALGORITHME **RT** [JEAN]



ALGORITHME **RT** [DER]

❑ ALGORITHME

- Construire une solution initiale "s" et calculer son coût Z = f(s)
- Z* : = Z { meilleur coût obtenu }
- S* : = s { meilleure solution connue }
- Initialiser MaxIter { nombre maximum d'itérations }
- T : = ∅ { la liste tabou T est vide }
- NIter : = 0

Répéter

NIter : = NIter + I {exploration du voisinage V(s) de s }

Z" : = + ∞

Pour toute solution s' de V(s)

Si s' ∉ T **alors** { si s' n'est pas taboue }

si f(s') < Z" **alors** { mise à jour du voisin " le moins pire " }

s" : = s'

Z" : = f(s')

Stocker l'inverse de la transformation dans u

FS

FS

FP

Si Z" < + ∞ **alors** { si un voisin non tabou a été trouvé }

S : = s"

Z : == Z"

Enlever la transformation en tête de T (la plus ancienne)

Si Z < Z" **alors** { mise à jour de la meilleure solution }

S* : = s

Z* : = Z

FS

FS

Jusqu'à (NIter = MaxIter) ou (Z" = + ∞)

CRITÈRE D'ARRÊT DE **RT**[REF]

- Les critères d'arrêt possibles sont :
  1. Si une **solution prouvée optimale** a été trouvée.
  2. Si une **limite a été atteinte** en ce qui concerne
     → Le nombre d'itérations.
     → Le temps de calcul.
  3. Si la **recherche semble stagner** : nombre d'itérations sans amélioration de la meilleure configuration trouvée.

NB. On peut arrêter la recherche à tout moment.
→ Contrairement au recuit simulé.

RT: MÉMOIRE DES TABOUS

- **Mémoire à court terme** du processus de recherche
  - Liste des ***t*** **dernières solutions** visitées
    - peu utilisé
    - assez coûteux
  - Liste des ***t*** **dernières transformations** effectuées et on interdit la transformation inverse
    - type de tabou le **plus** fréquent
  - Liste des **caractéristiques** des *t* dernières **solutions** ou **transformations**
    - **moins** fréquent
    - peut être très **efficace**

RT: MÉMOIRE DES TABOUS[Sebastien]

- Lorsqu'un mouvement est effectuée : interdiction pendant n itérations
  - Si n trop faible, tabou peu éfficace
  - Si n trop grand, les solutions sont "à flanc de coteau".

    → Stratégie de **diversification**

- **Mémoire a long terme**
- Statistique sur les mouvements : Repérer les mouvements trop utilisés (difficulté de recherche, optimum local...)
  - Fréquence freq(m) d'utilisation d'un mouvement m :
  - Pénalisation du mouvement m par ajout d'interdiction en fonction de freq(m).

RT: MÉMOIRE DES TABOUS

ADVANCED MECHANISMS (Diverses améliorations de la RT)

❖ **Intensification** (medium-term memory): The medium-term memory stores the elite (e.g., best) solutions found during the search.

→ The idea is to give priority to attributes of the set of elite solutions, usually in weighted probability manner.

✓ The search is biased by these attributes.

❖ **Diversification** (long-term memory): The long-term memory stores information on the visited solutions along the search.

→ It explores the unvisited areas of the solution space. For instance, it will discourage the attributes of elite solutions in the generated solutions to diversify the search to other areas of the search space [Talbi].

COMPORTEMENT DE L'ALGORITHME TABOU

- **Si la liste taboue est courte**
  - Il y a **moins d'interdictions** (mouvements tabous).
  - La recherche épouse mieux **les optima locaux** rencontrés.
  - L'algorithme tend à parcourir de **moins** grandes distances dans l'espace de recherche.

    → Il explore moins l'espace de recherche.
  - Le **risque de cycles** est plus grand [ref].

COMPORTEMENT DE L'ALGORITHME TABOU

- **Si la liste taboue est longue**

  -Il y a avantage d'interdictions (mouvements tabous).

  → La recherche **risque de manquer de nombreux optima locaux** sur son chemin.

  → L'algorithme tend à **parcourir de plus grandes distances** dans l'espace de recherche. Il explore davantage l'espace de recherche.

  -Le **risque de cycles est réduit** [ref].

COMPORTEMENT DE L'ALGORITHME TABOU

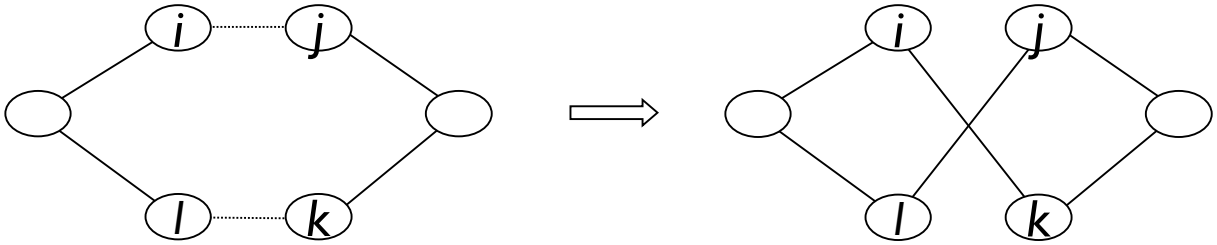
- **Le comportement de l'algorithme dépend**
  - La **longueur** de la liste taboue.
  - La **taille** de la liste de candidats.

Search Memory	Role	Popular Representation
Tabu list	Prevent cycling	Visited solutions, moves attributes Solution attributes
Medium-term memory	Intensification	Recency memory
Long-term memory	Diversification	Frequency memory

Table. The Different Search Memories of Tabu Search [Talbi]

RT : EXEMPLES DE TABOUS [OC]

- **PVC avec voisinage de type 2-opt**
  - 2-opt constitue l'heuristique interne de l'algorithme de RT
  - transformations : retirer les arêtes (*i*, *j*) et (*k*, *l*) et les remplacer par (*i*, *k*) et (*j*, *l*)



  - tabous possibles :
    - interdire les tournées elles-mêmes
    - interdire la transformation inverse

      $[(i, k), (j, l)] \rightarrow [(i, j), (k, l)]$ **interdit**
    - interdire toute transformation impliquant (*i*, *k*) ou (*j*, *l*)

AVANTAGES ET INCONVÉNIENTS DE LA **RT**

Avantage

- Grande efficacité
- Fonctionnement simple à comprendre.

Inconvénients

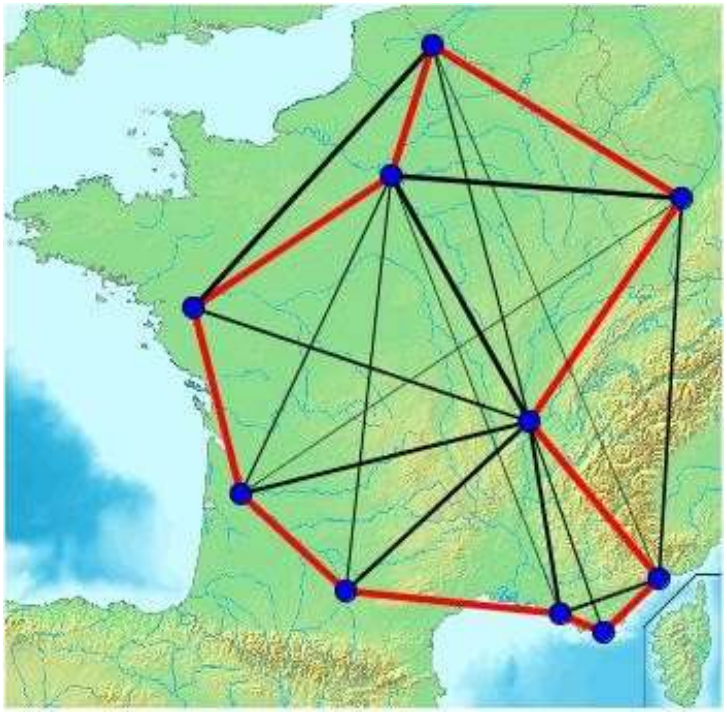
- Paramètres peu intuitifs.
- Demande en ressources importantes si la liste des tabous est trop imposante.
- Aucune démonstration de la convergence.

DOMAINES CONCERNÉS PAR LA RT

- Problèmes de transport.
- Planification et ordonnancement.
- Optimisation de graphes.
- Télécommunications.
- Logique et intelligence artificielle.
- Création d'horaires.
- Optimisation de structures.
- Techniques spécialisées.

RT pour le Problème de Voyageur de Commerce [HAM ET ALL]

- Un voyageur de commerce doit visiter un certain nombre de villes, et chaque ville une et une seule fois.
- Etant donne des distances entre chaque paire de villes,
  - il doit minimiser la distance totale parcourue.



RT pour le Problème de Voyageur de Commerce [HAM ET ALL]

- On peut représenter ce problème par un graphe : chaque ville correspond a un sommet et chaque arête a une paire de villes pouvant être visitées l'une a la suite de l'autre.

- Le problème correspond a trouver un tour complet (circuit Hamiltonien) dans ce graphe qui minimise la somme des distances **f**.

- Le nombre de solutions pour **n** villes est de **(n-1)!/2.**
  - Si **n = 4**, il y a trois solutions possibles.
  - Si **n = 30**, il y en a **4 420 880 996 869 850 977 271 808 000 000** !

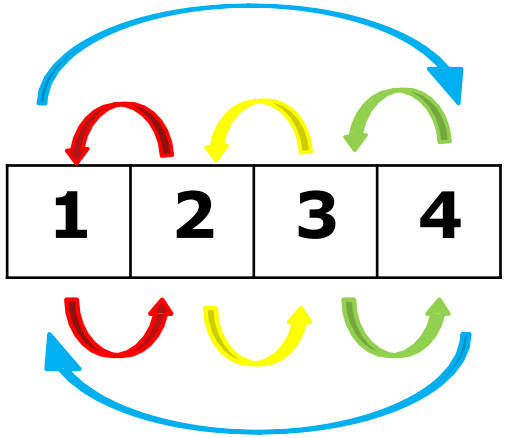
INSTANCE DU PROBLÈME [HAM ET ALL]

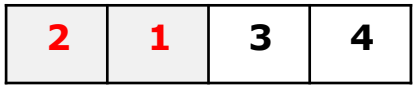
- Une instance du **PVC** est un graphe complet de **n** sommets dont les arêtes sont pondérées par un coût strictement positif.
- L'instance sera alors implantée comme une matrice **M n×n** dont les coefficients sont strictement positifs sauf sur la diagonale où ils sont tous nuls.
- **M** est appelé matrice de coût. Ainsi la distance entre le sommet **i** et le sommet **j** est **Mij**.

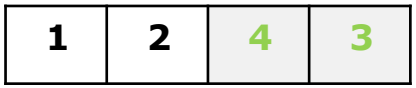
ESPACE DE RECHERCHE ET VOISINAGE

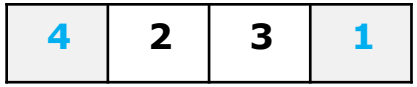
- L'espace de recherche **Sn** est l'ensemble des permutations de **{1,2, ⋯,n}.** Un point de l'espace de recherche est une permutation

- Voisinage **N(x)** : pour une solution **X**, **N(x)** est l'ensemble des permutation par paire de sommets [Ham et all],









FONCTION D'ÉVALUATION

- Pour une permutation x :

$$f(x) = \sum_{i=0}^{n-2} Mx_{(i)}x_{(i+1)} + Mx_{(n)}x_{(0)}$$

  - C'est cette fonction que nous cherchons à minimiser [Ham et all]

EXEMPLE D'INSTANCE DUPVC

	0	1	2	3	4	5	6	7	8	9
0	0	12	26	11	47	23	46	44	52	35
1	12	0	44	20	49	43	90	92	27	91
2	26	44	0	50	35	21	11	82	1	92
3	11	20	50	0	5	57	69	62	74	91
4	47	49	35	5	0	61	89	26	28	80
5	23	43	21	57	61	0	11	8	73	36
6	46	90	11	69	89	11	0	16	55	61
7	44	92	82	62	26	8	16	0	83	71
8	52	27	1	74	28	73	55	83	0	74
9	35	91	92	91	80	36	61	71	74	0

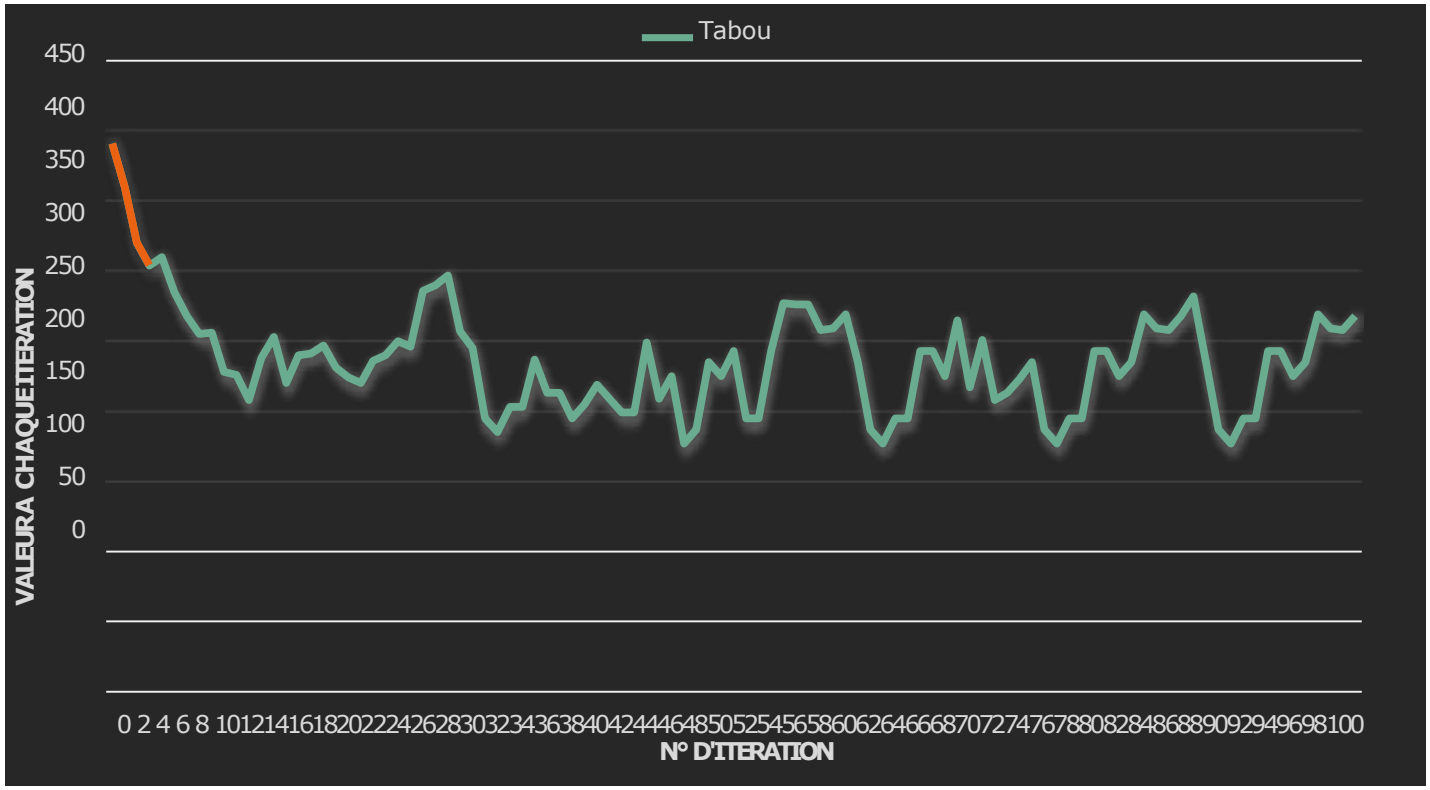
- Exemple de 10 villes avec une distance entre 0 et 100 [Ham et all]

RÉSULTAT DE LA RECHERCHE TABOU

- Configuration [Ham et all] :
  - Liste tabou de taille 10
  - Critère d'arrêt : nombre d'itération (100)

Solution initial :	s = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9],	valeur = 391
Solution Descente :	s = [0, 2, 1, 3, 4, 7, 5, 6, 8, 9],	valeur = 304
Solution Tabou :	s = [9, 0, 3, 1, 8, 4, 7, 5, 6, 2],	valeur = 177

RÉSULTAT DE LA RECHERCHE TABOU



1. ALGORITHME DE RECUIT SIMULÉ
(Simulated Annealing Algorithm)

RECUIT SIMULÉ [M.-J. HUGUET]

- Idée:
  - Analogie métallurgie: en chauffant un métal puis en le refroidissant doucement on peut obtenir des structures cristallines résistantes.
  - S. Kirpatrick 1983 / V. Cerny 1985.
  - Pour l'optimisation :
    - Diversifier la recherche en acceptant des voisins qui dégradent la fonction objectf en fonction d'une probabilité d'acceptation qui décroit dans le temps.

- Stratégie d'exploration :
  - Paramètre T (température) qui décroit au cours des itérations.
  - Choix d'un voisin S' tel que :
    - Si $\Delta = f(s') - f(s) < 0$ alors $s \leftarrow s'$ -- intensification
    - Si $x < e^{\frac{-\Delta}{T}}$ (x choisi aléatoirement dans [0,1]) alors $s \leftarrow s'$ -- diversification

RECUIT SIMULÉ [M.-J. HUGUET]

- **Température** : probabilité d'accepter une solution non améliorante.
- **Condition de Métropolis** :
  - Accepter la nouvelle solution avec une probabilité: $e^{\frac{-\Delta}{T}}$
  - $si\ \Delta\uparrow\ alors\ e^{\frac{-\Delta}{T}}\downarrow; si\ T\downarrow\ alors\ e^{\frac{-\Delta}{T}}\downarrow$

- **Algorithme**

```
1.  s0 : = solution initiale
2.  T0 : = Température initiale
3.  S : = S0 ; T : = T0 ;
4.  while (Condition) loop
5.      S' : = Random (N(S));
6.      Delta : = f(s') - f(s)
7.      if Delta<0 ou random< exp(-Delta/T) then
8.          s : = s';
9.      end if
10.     T : = k.T;
11. end while
12. Return S
```

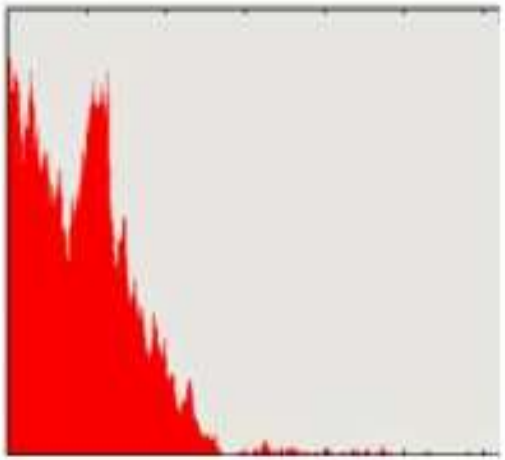
RECUIT SIMULÉ [M.-J. HUGUET]

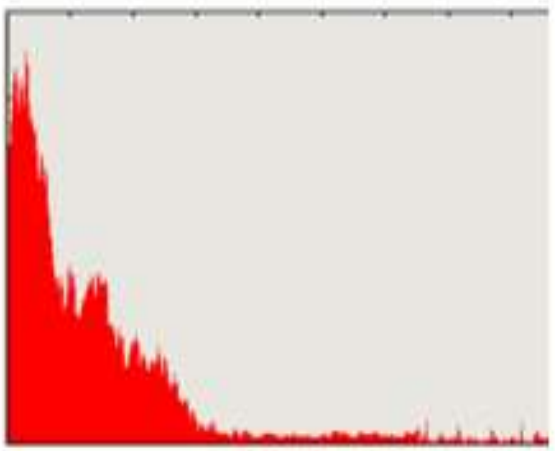
- **Au début des itérations :**
  - T élevé : Acceptation fréquente de solutions non améliorantes
- **En fin des itérations :**
  - T faible : acceptation rare de solution non améliorante
- **Réglage des paramètres**
  - Température initiale
  - Variation de température : à chaque étape / par palier / adaptative
  - Conditions d'arrêt (température, fonction objectif, …)
  - Trouver le bon paramétrage ….

RECUIT SIMULÉ [M.-J. HUGUET]

- **Etat initial**
  - ❖ Solution initiale
  - ❖ Température : doit permettre d'accepter « suffisamment » de solutions voisines

- **Schéma de refroidissement**
  - ❖ Si trop rapide : convergence prématurée : on reste dans un optimum local
  - ❖ Si trop lente : exploration trop importante





RÉSUMÉ : VUE D'ENSEMBLE DE RS, AG ET RT [OC]

	RS	AG	RT
Construction	solution initiale x	population initiale	solution initiale x
Recombinaison	–	Reproduction	–
Modification aléatoire	x' tiré au hasard dans N(x) et accepté avec probabilité p(x')	sélection et mutation	RT probabiliste
Amélioration	–	recherche locale pour optimiser les enfants	recherche locale pour améliorer x
Mise-à-jour de la mémoire	–	remplacement d'une génération	liste tabou
Mise-à-jour des paramètres	paramètre T, longueur des paliers	–	liste des tabous, longueur des listes, poids λ

- Plus sur les tabous [Mich]
  - Il est souvent utile d'utiliser plusieurs listes de tabous simultanément.
  - Les tabous « simples » peuvent implantés au moyen de listes circulaires de longueur fixe.
  - Les tabous de durée fixe ne peuvent pas toujours prévenir le cyclage: plusieurs auteurs ont proposé diverses techniques pour varier la longueur de la liste des tabous en cours d'exécution (Skorin-Kapov, Taillard).
  - Autre solution: les étiquettes tabous aléatoires, qui imposent des tabous dont la durée est générée aléatoirement à leur création.
  - Autre solution encore: les tabous activés aléatoirement (à chaque itération, un nombre aléatoire indique jusqu'où regarder dans la liste des tabous).

- Critères d'aspiration [Mich]
- Les tabous sont parfois trop puissants:
  - ils interdisent des mouvements intéressants, même s'il n'y a aucun risque de cyclage;
  - ceci peut provoquer une stagnation de la recherche.
- Les critères d'aspiration sont des éléments algorithmiques qui permettent d'annuler des tabous dans certaines circonstances.
- Le plus simple des critères d'aspiration consiste à toujours permettre un mouvement qui permet d'atteindre une solution supérieure à la meilleure solution connue.
- Des critères plus compliqués ont été proposés et parfois appliqués.
- RÈGLE D'OR: S'il n'y a pas de danger de cyclage, on peut ignorer les tabous!

RÉFÉRENCES DU CHAPITRE

- **[OC]** Optimisation Combinatoire, Support de cours, Ecole polytechnique MOTREALE.
- **[Ham et all]** Hamza Deroui, Cherkaoui Soufiane et Oussama Zaki, « Algorithme de recherche avec tabou, coptimisation combinatoire », support de cours, INSA Toulouse
- **[Mich]** Michel Gendreau , Introduction à la recherche avec tabous, CIRRELT et MAGI, École Polytechnique de Montréal, 2015.
- **[Jean]** Méthode Tabou, Jean-Philippe Préaux, http://www.i2m.univ-amu.fr/~preaux
- **[sidi et all]** Sidi Mohamed Douiri, Souad Elbernoussi, Halima Lakhbab, « cours des méthodes de résolution exactes heuristiques et métaheuristiques », support de cours, Université Mohammed V, Rabat.
- **[JoMa]** Joseph Ayas et Marc André Viau , « La Recherche Tabou », support de cours 2004.
- **[DER]** DERBALA Ali, Ordonnancement dans les ateliers. Support de Cours, université saad dahlab , blida.
- **[Ben]** S. Ben Ismail, « introduction à l'optimisation combinatoire », support de cours
- **[Phill]** Philippe Muller, Optimisation combinatoire : méthodes approchées Master 1, IUP SI, 2011-2012.
- [M.-J. Huguet] M.-J. Huguet, « Méta-Heuristiques », support de cours, LAAS,CNRS, 2018.