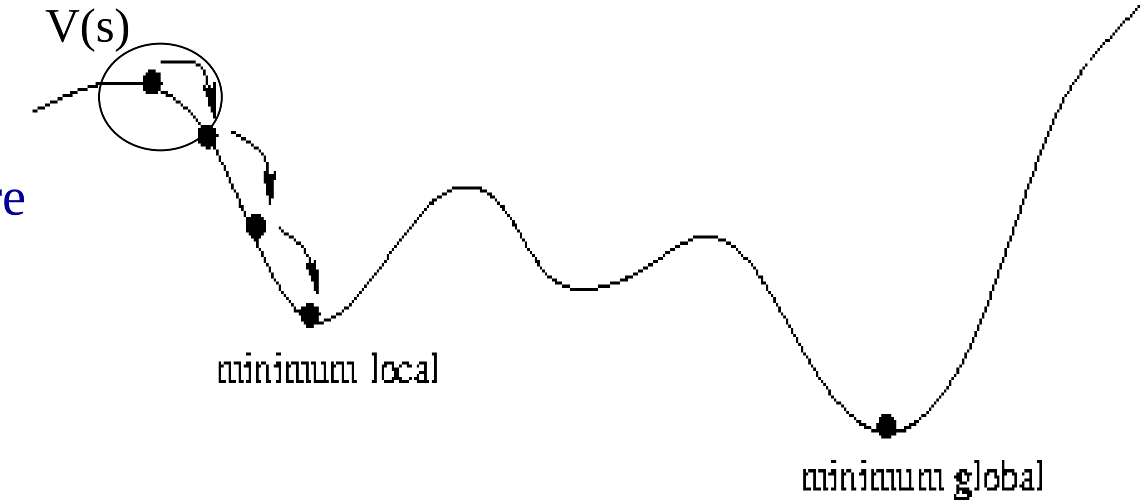
Les algorithmes de recherche locale
La recherche Tabou

Par

Maria ZRIKEM



A chaque itération, la meilleure solution s ' de V(s) est sélectionnée

Carences et solutions

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

⇓

Recherche Tabou

La méthode de recherche Tabou (RT)

Glover (1986), Hansen (1986)

Acceptation de solutions moins bonnes que la solution courante

⇓

Risque de « cycler »

⇓

Liste de tabou : solutions interdites pour un nombre d 'itérations

⇓

Critères d 'aspiration : conditions permettant de lever le statut tabou d 'une solution

La méthode de recherche Tabou : idee de base

Développée dans un cadre particulier par Glover en 1986 (et indépendamment par Hansen en 1986) puis généralisée en 1989-90.

Comme la descente, se déplace d'1 solution courante s vers une solution voisine s' telle que min$_{s'' \in N(s)}$ F(s'').

Si optimum local rencontré, déplacement vers le "moins mauvais" des voisins : dégradation de la fonction objectif
Mais risque de cycler autour de l'optimum local en faisant l'opération inverse à l'itération suivante!!!

➡ **Idée : interdire la dernière opération**

Mais la « vallée » de l'optimum local est « profonde » : plusieurs itérations pour s'en extraire

➡ **Idée : garder en mémoire les dernières solutions visitées et interdire le retour vers celles-ci pour un nombre fixé d'itérations**

La méthode de recherche Tabou : mise en œuvre

Dans la pratique : conservation à chaque étape d'une ou plusieurs liste T de solutions « Taboues »

2 alternatives :

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

  OU

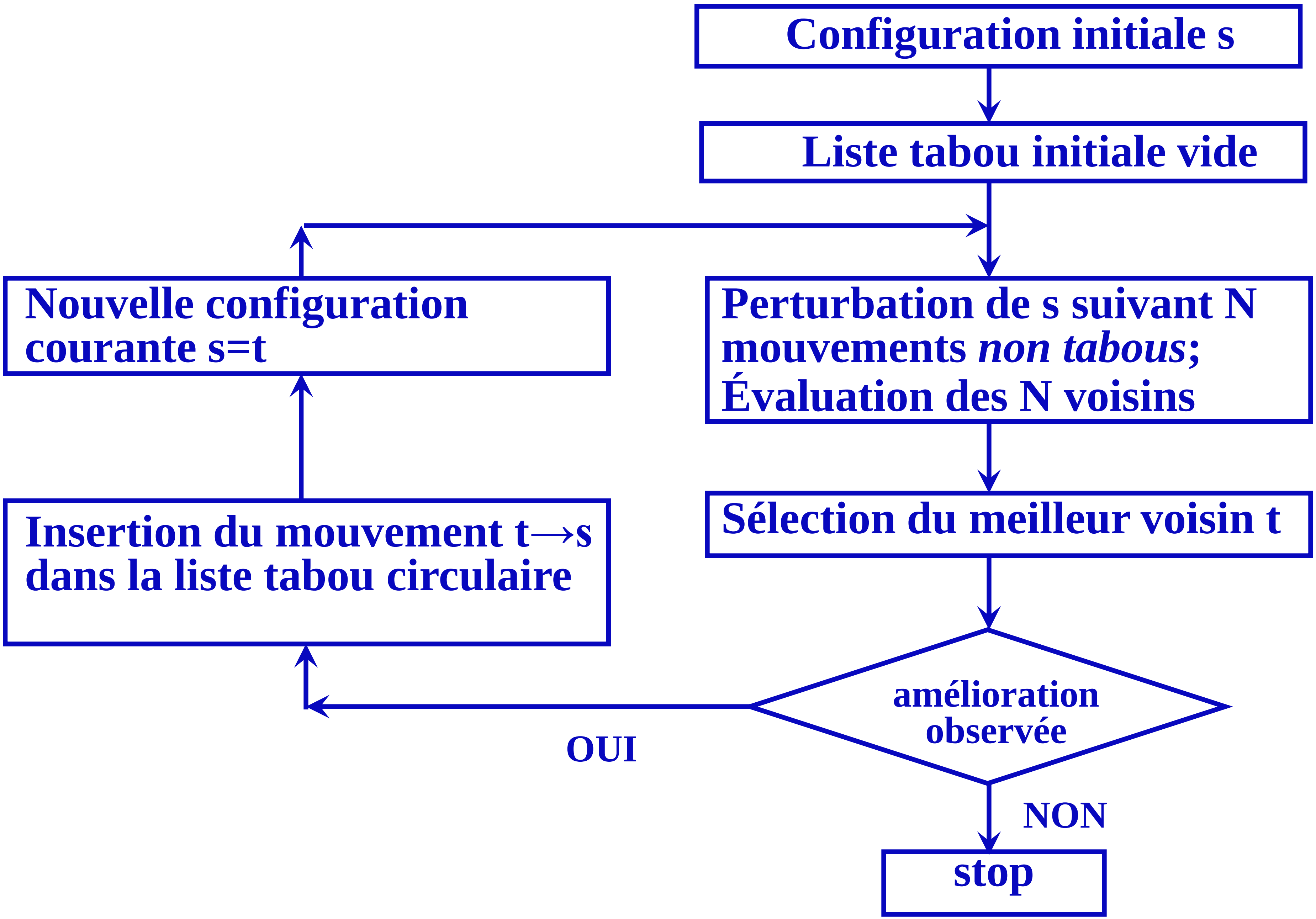
- **Une liste des mouvements interdits (qui ramènent vers ces solutions déjà visitées**

  Avantages : prend moins de place mémoire

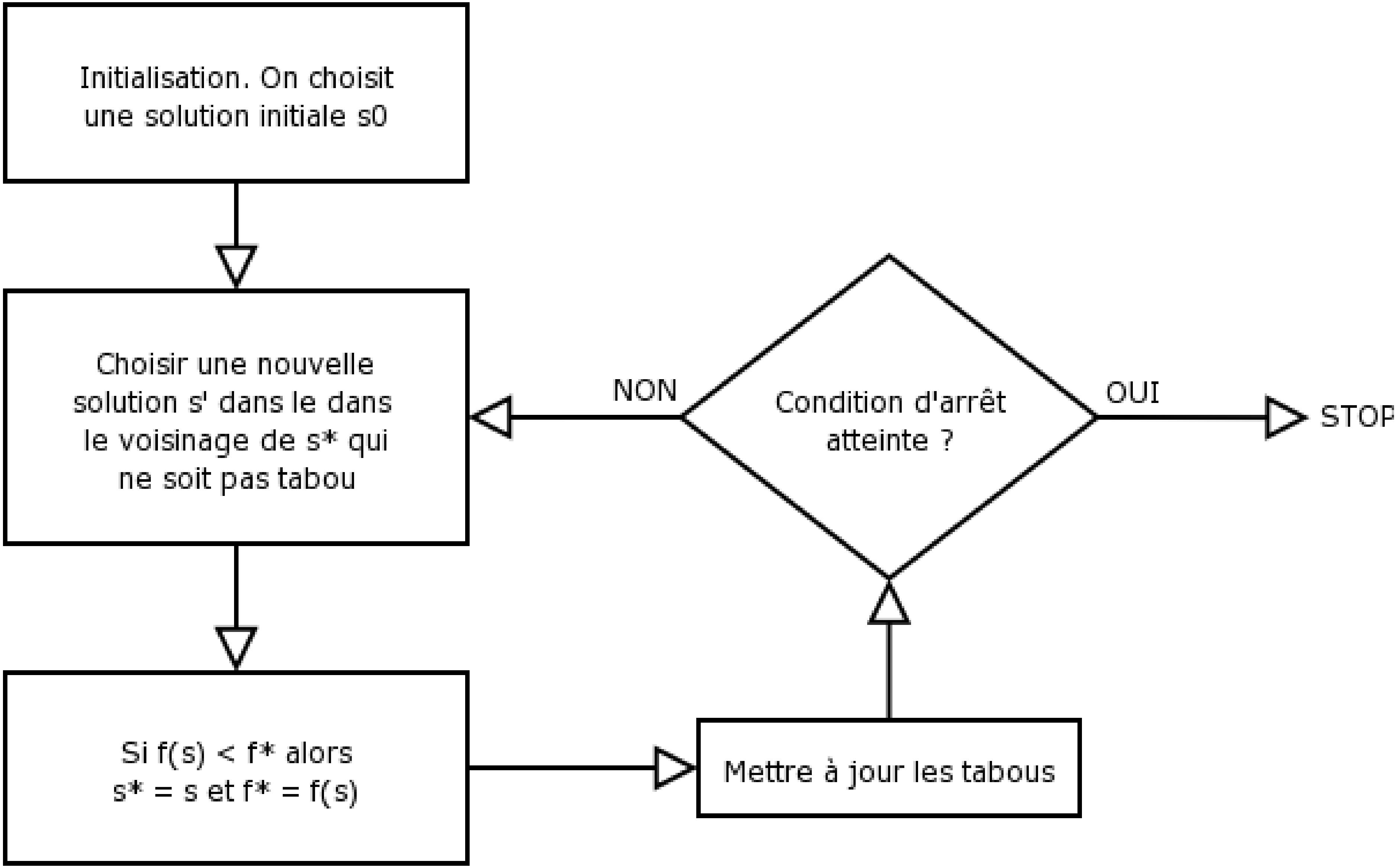
  élimine plus de solutions que celles visitées effectivement

Généralement les listes sont gérées en FIFO (first in first out)

La méthode de recherche Tabou : schéma général



La méthode de recherche Tabou : schéma général



La méthode de recherche Tabou : aspiration

Liste tabou des mouvements interdits élimine plus de solutions que celles visitées effectivement :

+ efficace que la liste des solutions taboues mais élimine éventuellement de très bonnes solutions !!!

Idée : définition d'une fonction d'aspiration

Possibilité de lever l'interdiction pour une solution si elle parait « prometteuse »

permet d'explorer une nouvelle région voisine de *s*
si $s' \in N(s)$, $s' \in T$, telle que $F(s') \leq A\ F(s)$ alors s' redevient candidate

La méthode de recherche Tabou : aspiration

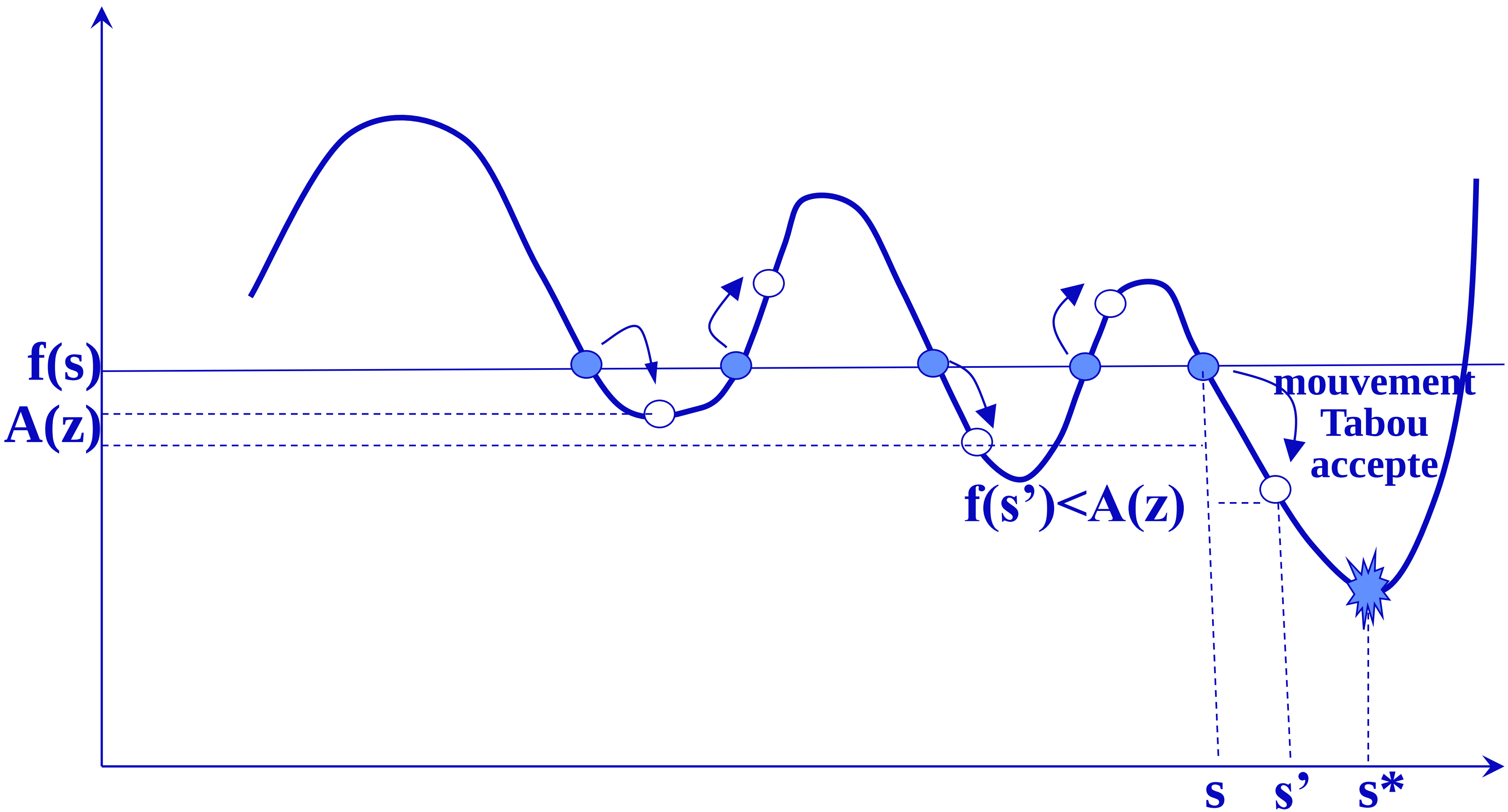


Illustration du concept de fonction d'aspiration

Définition : méthode heuristique de recherche locale utilisée pour résoudre des problèmes complexes et/ou de très grande taille (souvent NP-durs). La RT a plusieurs applications en programmation non linéaire (PNL).

Principe de base : poursuivre la recherche de solutions même lorsqu'un optimum local est rencontré et ce,

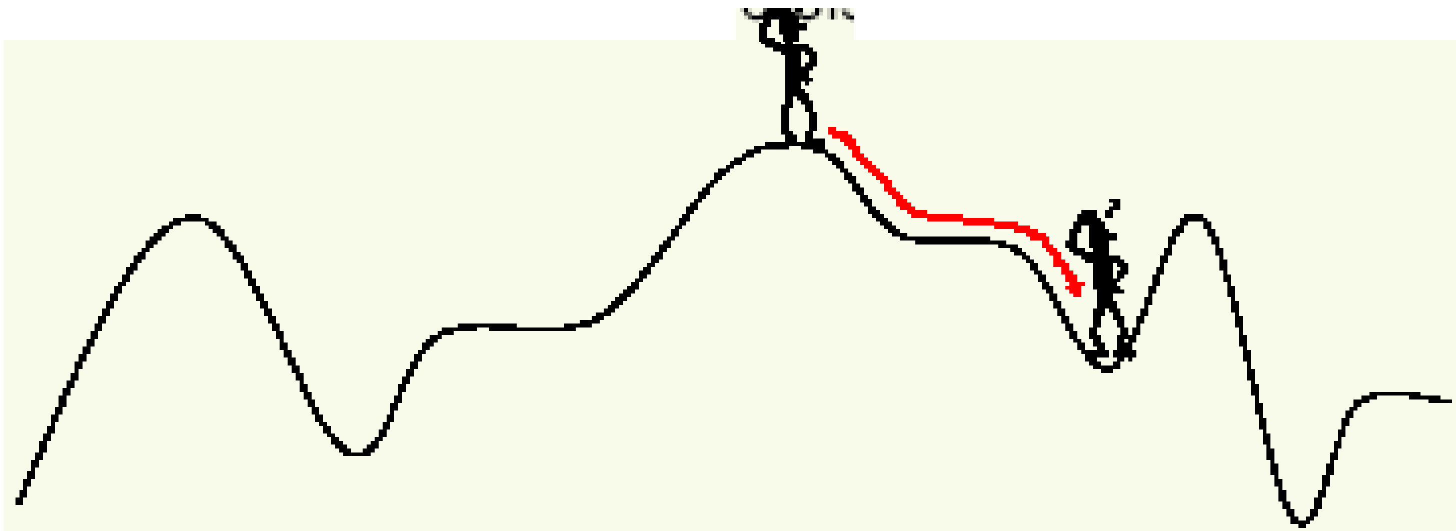
- ✓ en permettant des déplacements qui n'améliorent pas la solution
- ✓ en utilisant le principe de mémoire pour éviter les retours en arrière (mouvements cycliques)

- **Mémoire** :
  - ✓elle est représentée par une liste taboue qui contient les mouvements qui sont temporairement interdits
  - ✓mouvements interdits ou solutions interdites
  - ✓son rôle évolue au cours de la résolution: diversification (exploration de l'espace des solutions) vers intensification
- **Exception aux interdictions** : il est possible de violer une interdiction lorsqu'un mouvement interdit permet d'obtenir la meilleure solution enregistrée jusqu'à maintenant

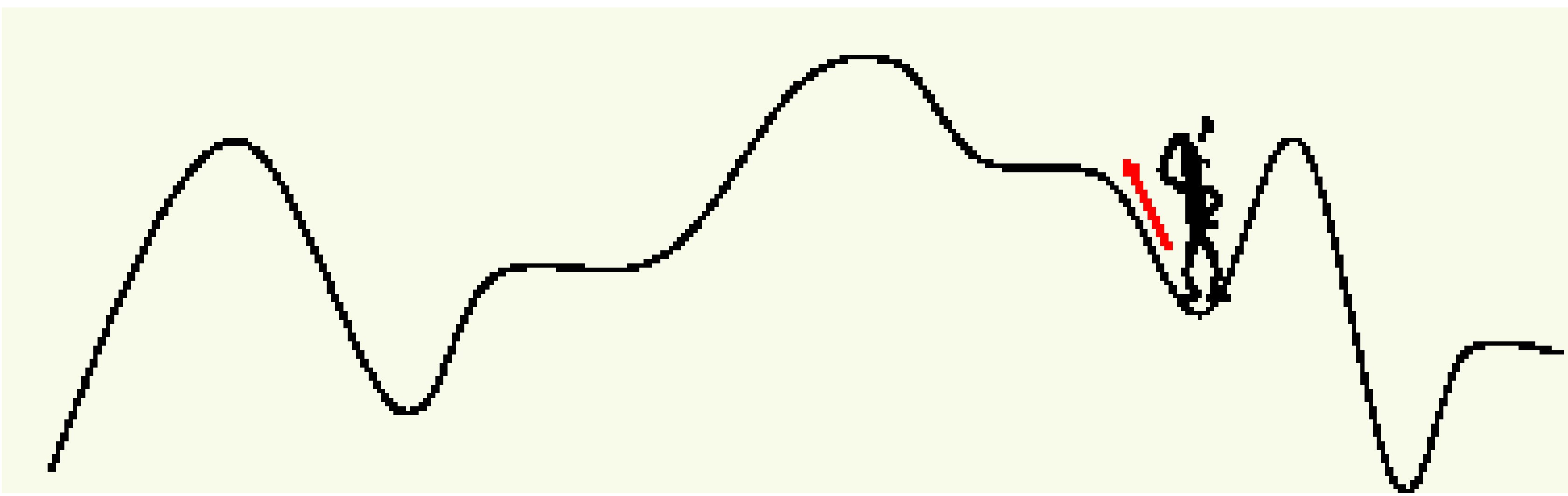
Mise en contexte : « Fable des randonneurs »

Un randonneur malchanceux , T. A. Bhoulx, est perdu dans une région montagneuse. Toutefois, il sait qu'une équipe de secours passe régulièrement par le point situé à la plus basse altitude dans la région. Ainsi, il doit se rendre à ce point pour attendre les secours. Comment s'y prendra-t-il ? Il ne connaît pas l'altitude de ce point et, à cause du brouillard, il ne voit pas autour de lui. Donc, arrivé à un croisement, il doit s'engager dans une direction pour voir si le chemin monte ou descend.

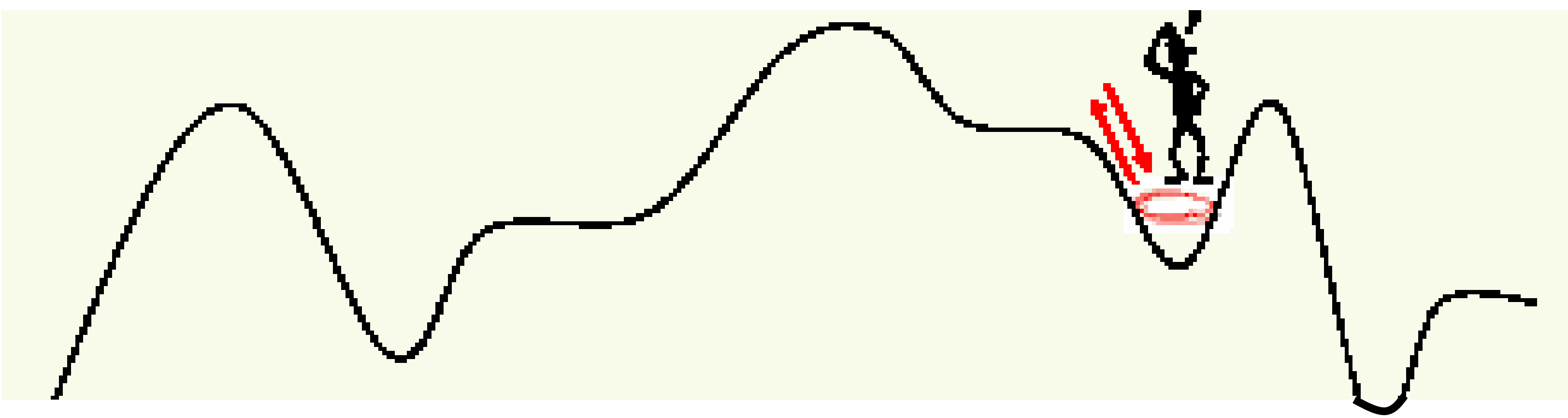
Tout d'abord, il commence par descendre tant qu'il peut, en choisissant le chemin de plus grande pente à chaque croisement.



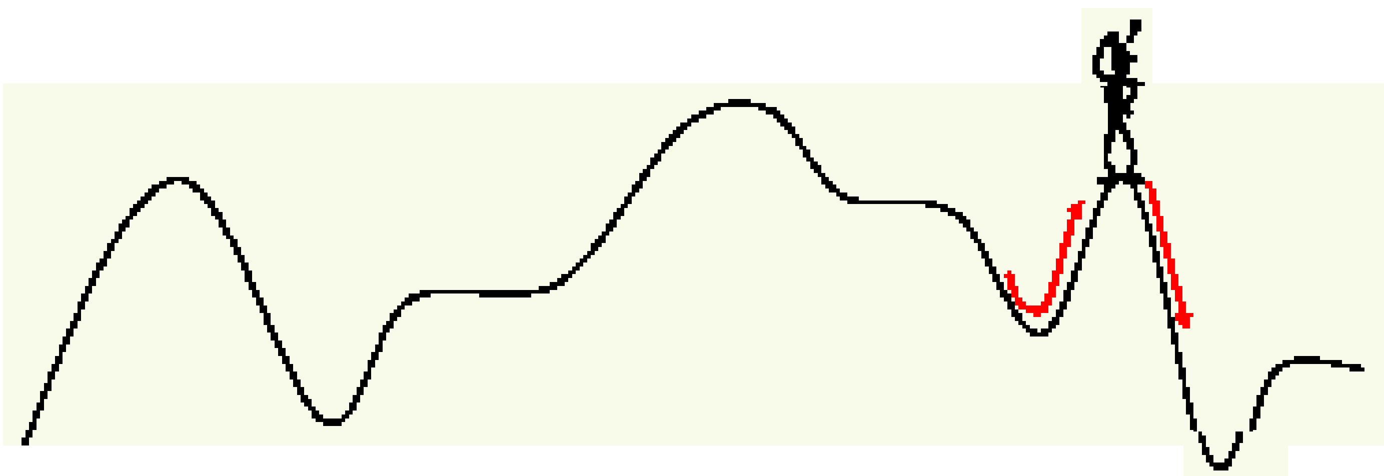
Puis, lorsqu'il n'y a plus de sentier menant vers le bas, il décide de suivre le chemin qui remonte avec la plus faible pente car il est conscient qu'il peut se trouver à un minimum local.



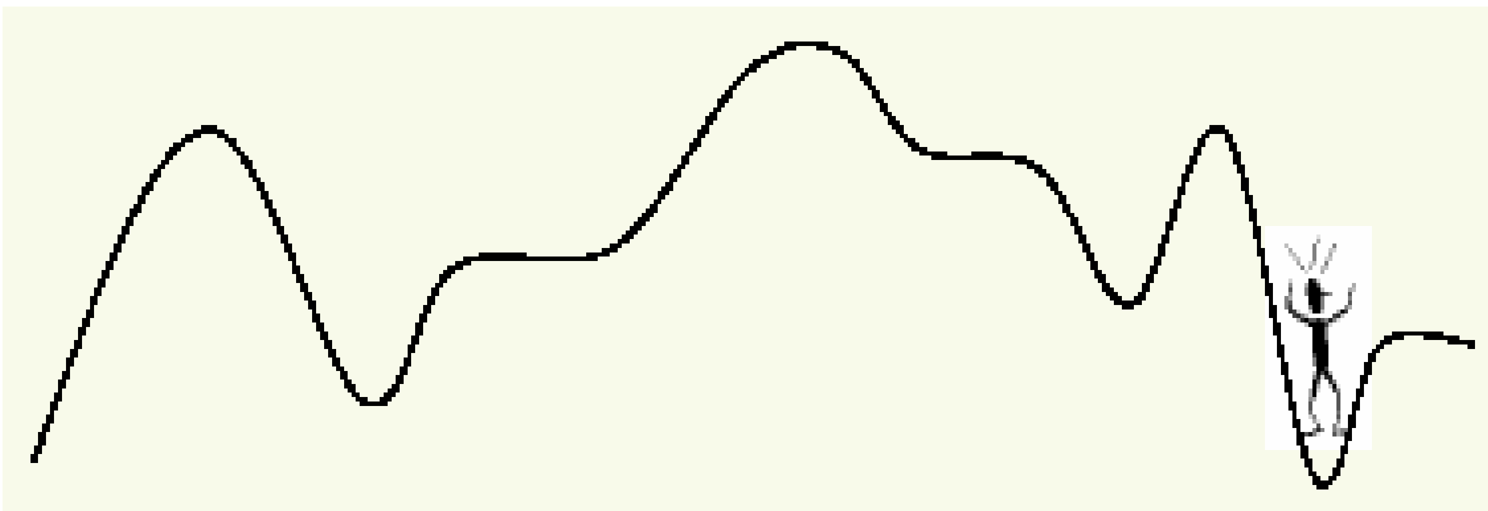
Toutefois, dès qu'il remonte, il redescend vers le point où il était. Cette stratégie ne fonctionne pas. Par conséquent, il décide de s'interdire de faire marche arrière en mémorisant la direction d'où il vient. Il est à noter que sa mémoire ne lui permet de mémoriser que les deux dernières directions prohibées.



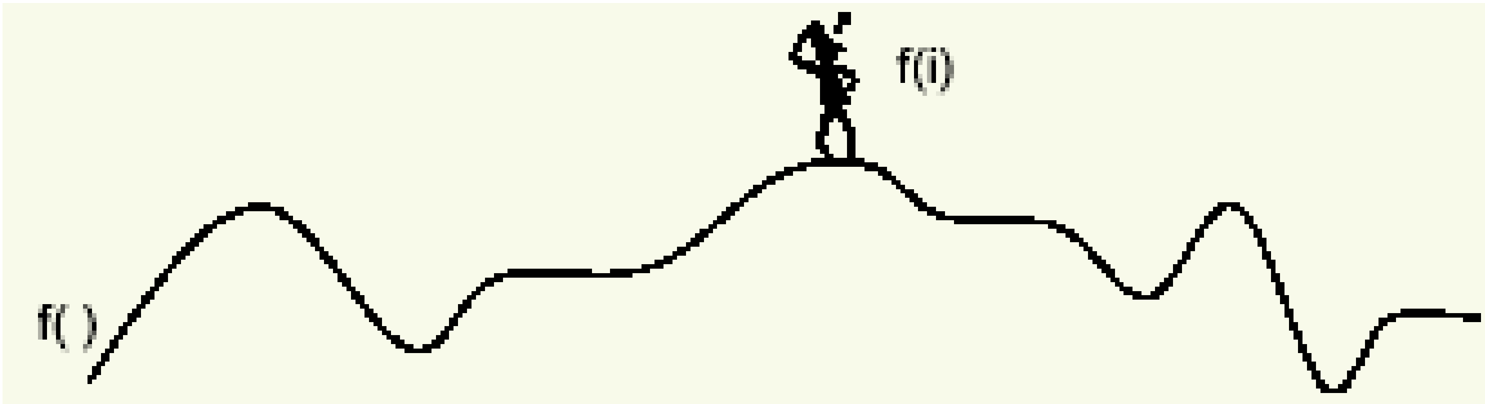
Cette nouvelle stratégie lui permet d'explorer des minimum locaux et d'en ressortir. À un moment donné, il arrive à un point où il décèle une forte pente descendante vers le sud. Toutefois, les directions mémorisées lui interdisent d'aller vers le sud car cette direction est prohibée. Il décide d'ignorer cette interdiction et emprunte ce chemin.



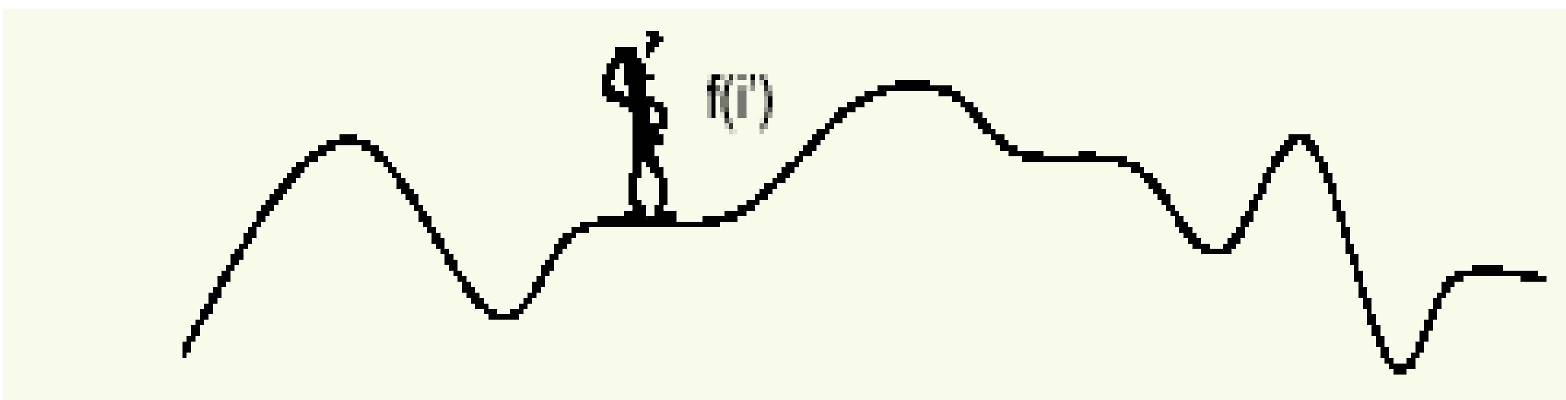
Cette décision fut bénéfique: il arriva au point de plus basse altitude et attendit les secours qui ne tardèrent à arriver.



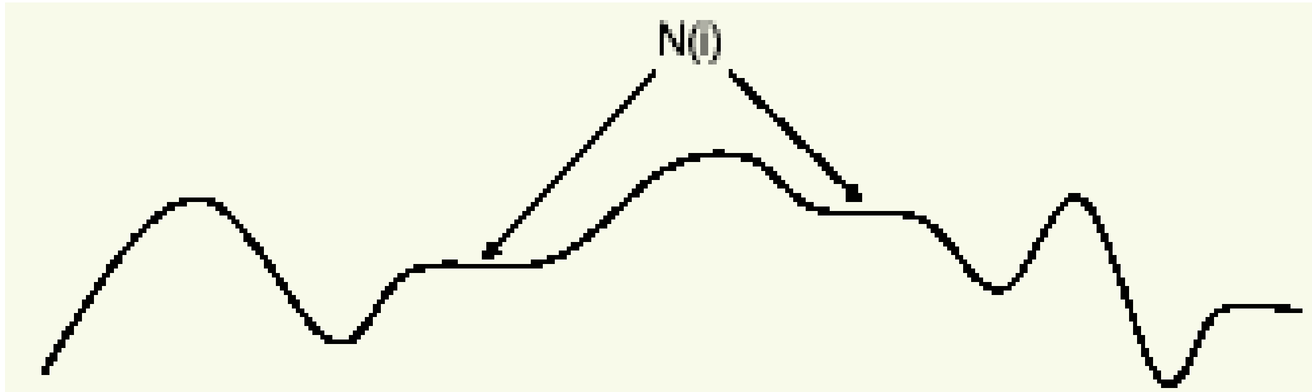
- i : la solution actuelle



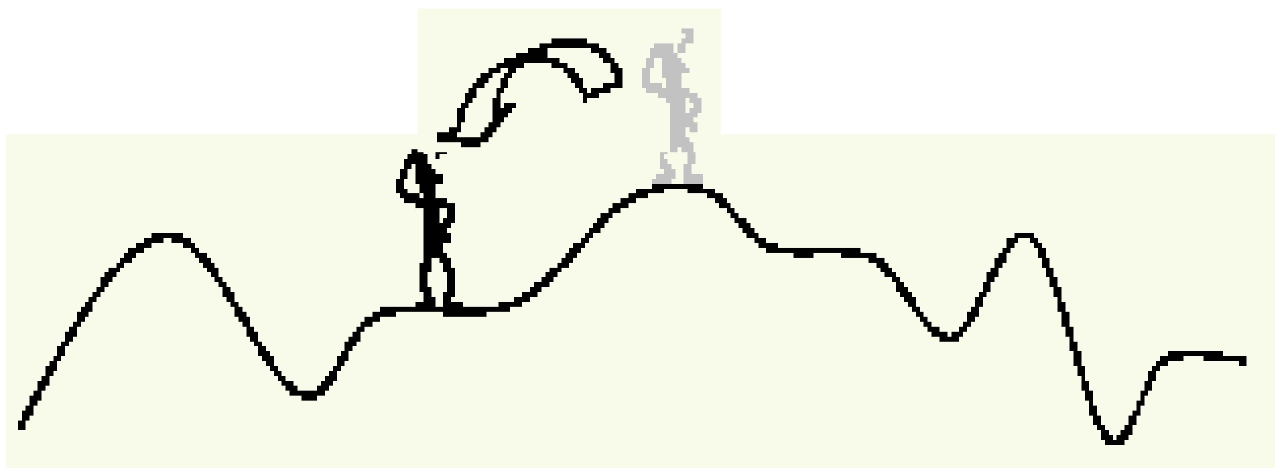
- i' : la prochaine solution atteinte (solution voisine)



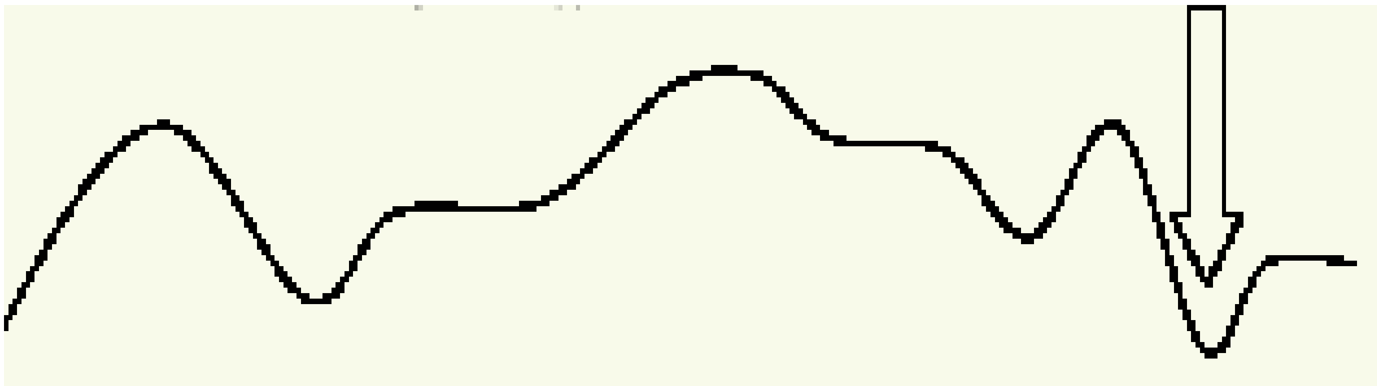
- N(i): l’espace de solutions voisines à i (l’ensemble des i’)



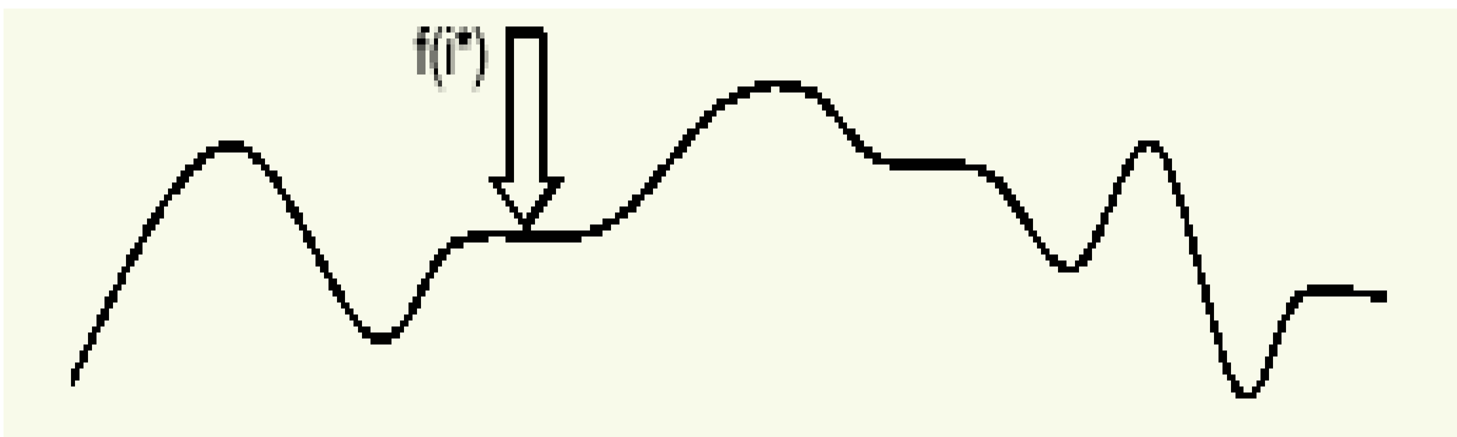
- m : mouvement de i à i’



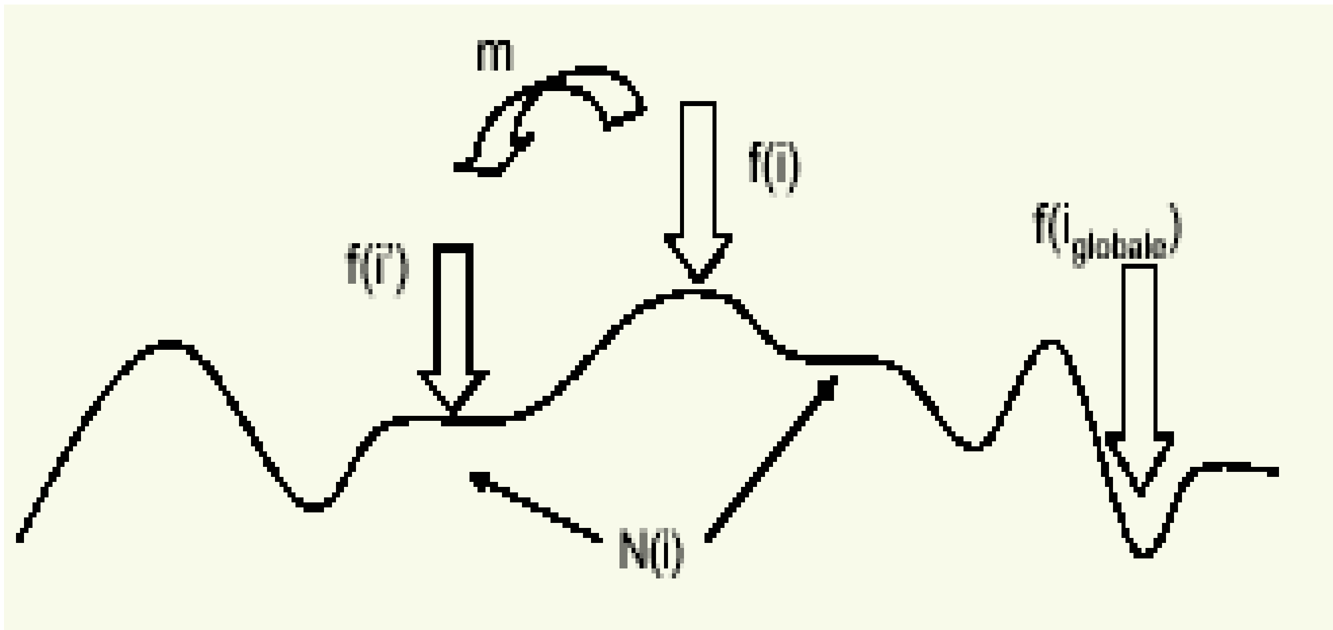
- $i_{globale}$: la solution optimale globale qui minimise la fonction objectif f( ).



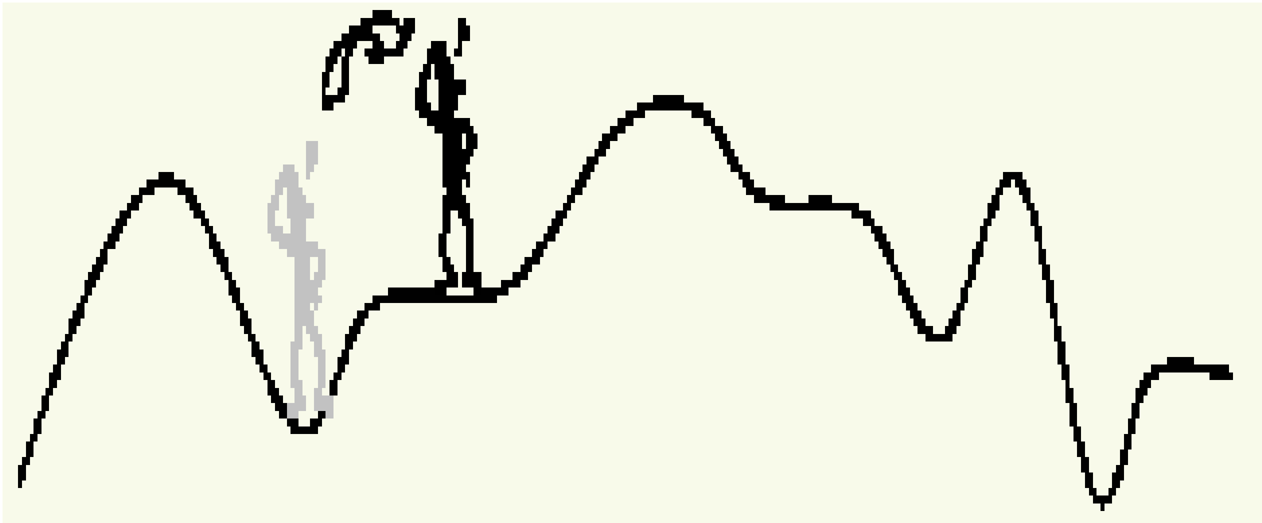
- i* : la solution optimale actuelle



Et donc, jusqu'à présent, nous avons:

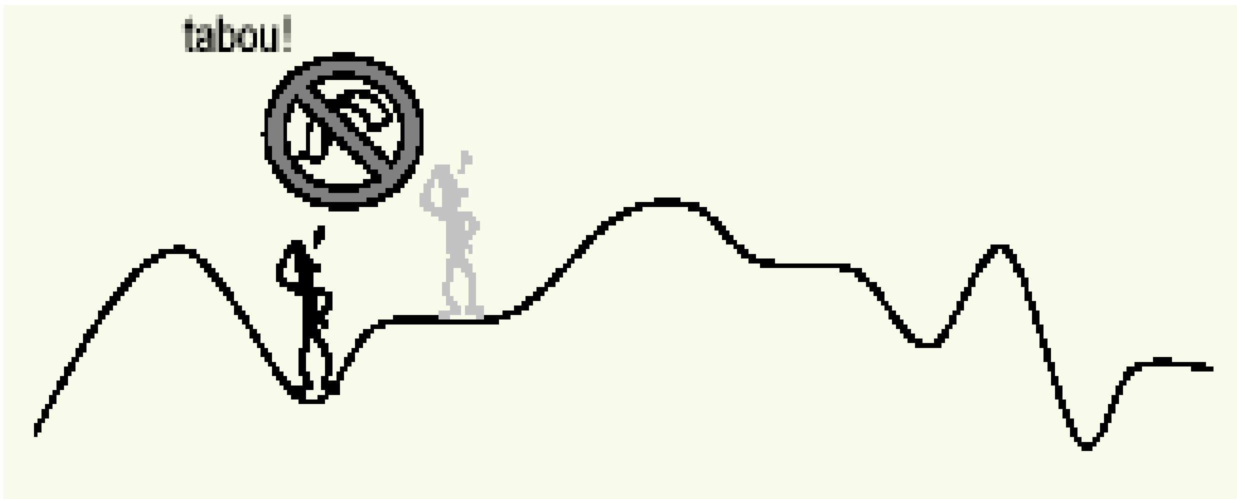


• Mouvement non améliorateur : un mouvement qui nous sortirait d’un minimum local i* en nous amenant à une solution voisine i’ *pire* que l’actuelle.



La méthode tabou permet un mouvement non améliorateur.

• Mouvement tabou: un mouvement non souhaitable, comme si on redescendait à un minimum local d'où on vient juste de s'échapper.



Le mouvement est considéré tabou pour un nombre prédéterminé d'itérations. k représente l'index des itérations (l'itération actuelle).

- T : liste des mouvements tabous. Il peut exister plusieurs listes simultanément. Les éléments de la liste sont t(i,m).

  Une liste T avec trop d'éléments peut devenir très restrictive. Il a été observé que trop de contraintes (tabous) forcent le programme à visiter des solutions voisines peu alléchantes à la prochaine itération.

  Une liste T contenant trop peu d'éléments peu s'avérer inutile et mener à des mouvements cycliques.

- a(i,m) : critères d'aspiration. Déterminent quand il est avantageux d'entreprendre m, malgré son statut tabou.

Étape 1: choisir une solution initiale i dans S (l’ensemble des solutions)

Appliquer i* = i et k = 0

Étape 2: appliquer k = k+1 et générer un sous-ensemble de solutions en N(i,k) pour que:

- les mouvements tabous ne soient pas choisis
- un des critères d’aspiration a(i,m) soit applicable

Étape 3: choisir la meilleure solution i’ parmi l’ensemble de solutions voisines N(i,k)

Appliquer i = meilleur i’

L’algorithme (suite)

Étape 4: si f(i) <= f(i*), alors nous avons trouvé une meilleure solution

Appliquer i* = i

Étape 5: mettre à jour la liste T et les critères d’aspiration

Étape 6: si une condition d’arrêt est atteinte, stop.
Sinon, retour à Étape 2.

Condition d’arrêt: condition qui régira l’arrêt de l’algorithme.
ex: arrêt après 22 itérations (k = 22).

Améliorations

La recherche de la solution optimale peut être améliorée. Voici quelques options:

– choix stratégique de la solution initiale i. Ceci donnera une «bonne » valeur de f(i*)

– Intensifier la recherche dans les voisinages de solutions qui semblent propices à mener à des solutions proches ou égales à l'optimum

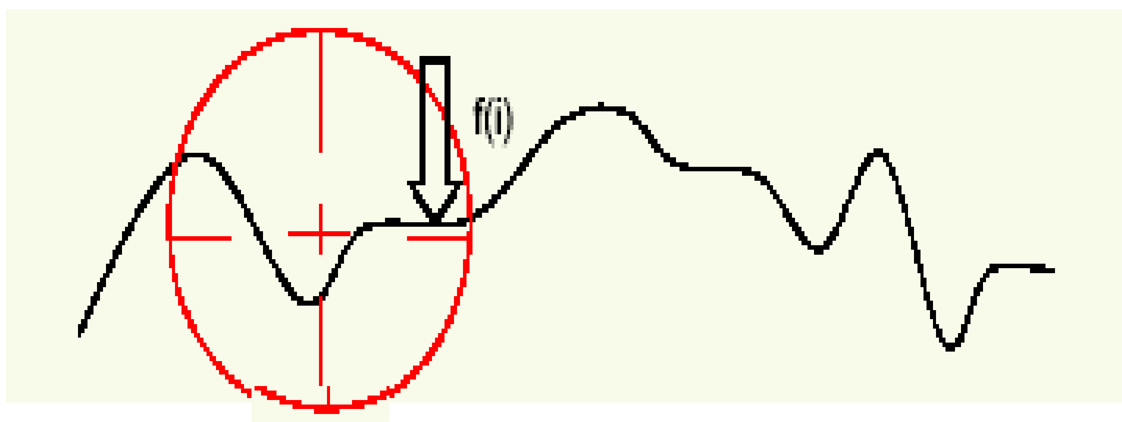
– Diversifier la recherche en éloignant celle-ci de voisinages peu propices à produire de bonnes solutions

Concepts d'intensification et de diversification

La recherche est menée dans un voisinage N(i) de S, l'ensemble des solutions

Une haute priorité est donnée aux solutions f(i') qui ressemblent à la solution actuelle f(i)

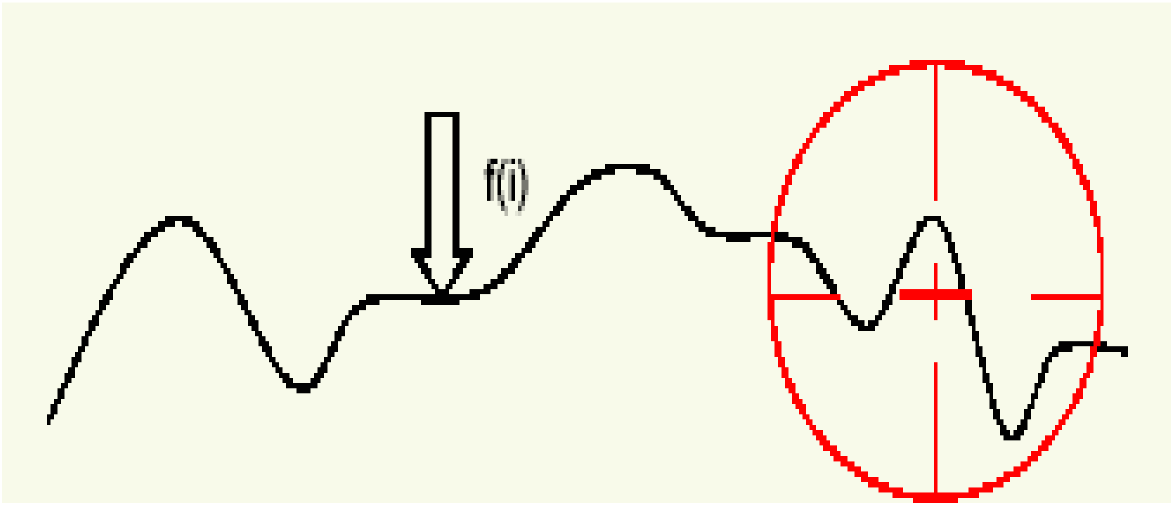
Le résultat est donc une intensification de la recherche dans un certain secteur, dans un voisinage choisi:



La recherche est éloignée du voisinage N(i) actuelle de l'ensemble des solutions

Une haute priorité est donnée aux solutions f(i') d'une autre région que celle actuellement sous exploration

Le résultat : chercher ailleurs



Modifications à f( )

L'intensification et la diversification sont présentées comme des modifications à la fonction objectif.

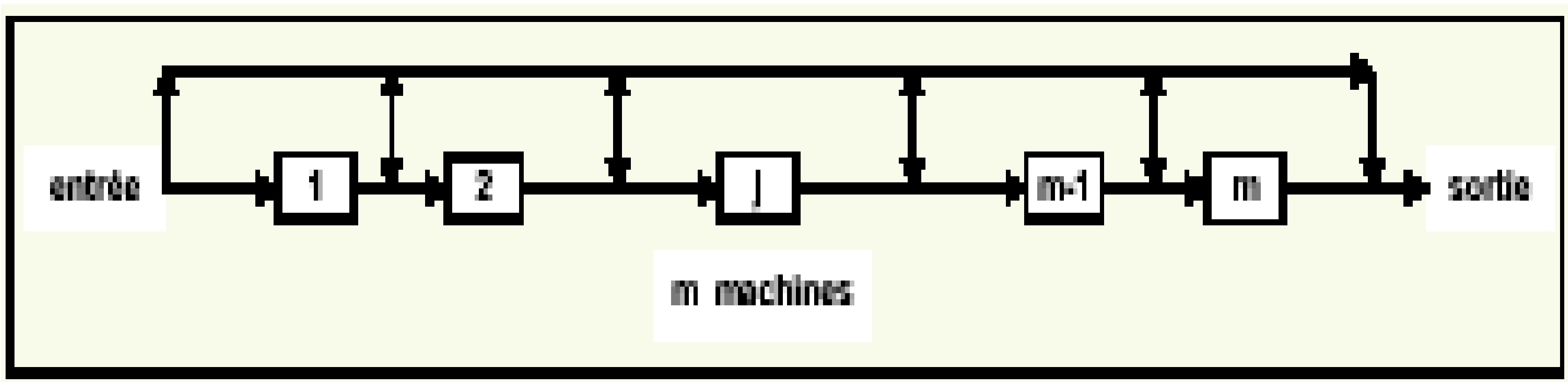
Pour l'intensification, une « pénalité » est attribuée à des solutions éloignées de l'actuelle. Ceci cause un gonflement de la fonction objectif : les solutions semblables seront donc privilégiées. Pour la diversification, l'effet est le contraire. Les solutions proches de l'actuelle sont pénalisées.

Donc: f ' = f + intensification + diversification

Il est à noter que l'intensification et diversification se manifestent pendant quelques itérations k seulement, ensuite il y a alternance.

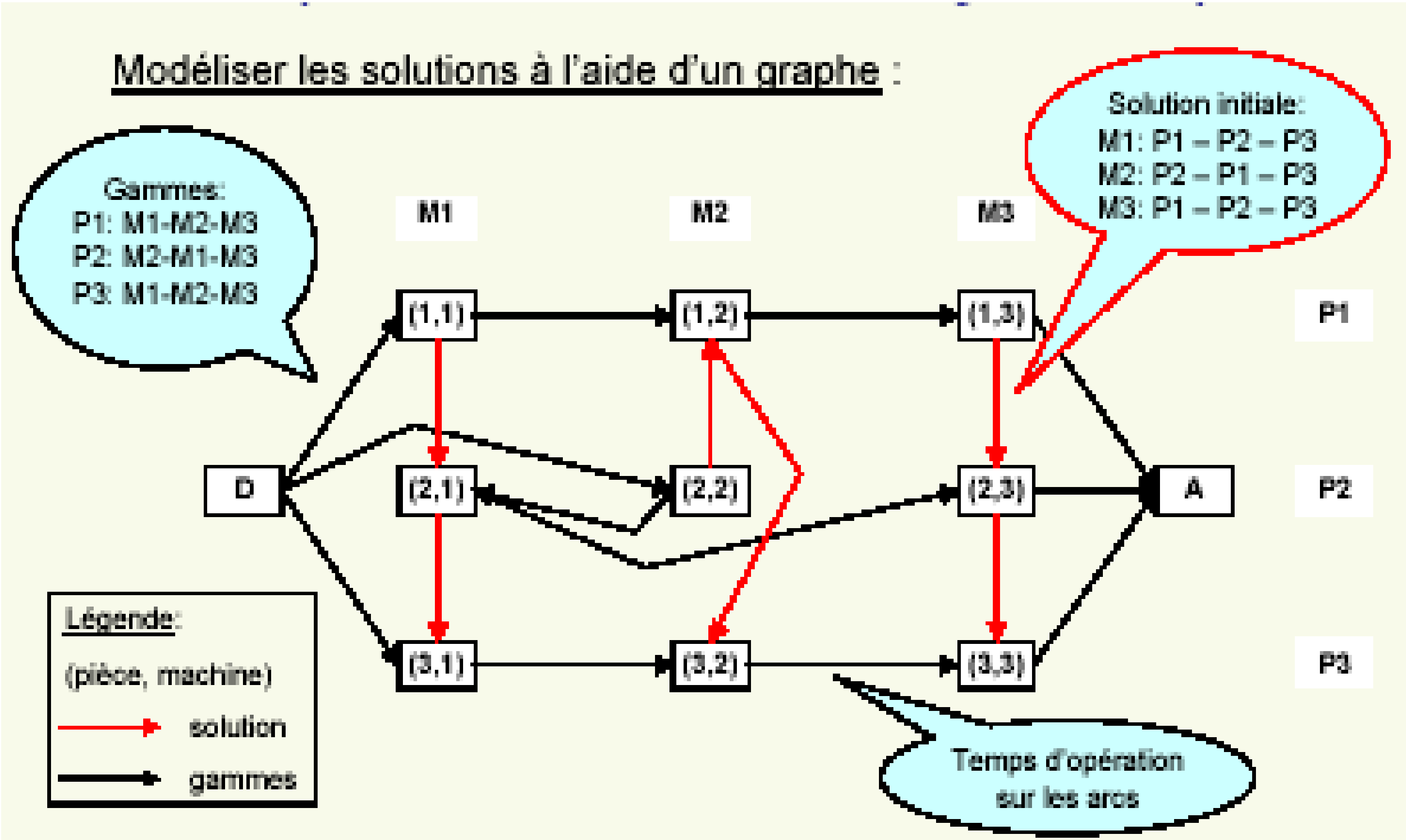
Les problèmes d’ordonnancement sur plusieurs machines (NP complets) sont divisés en deux grandes classes:

- **Problème de flow shop : Exemple : ligne d’assemblage**



- **problème de *job shop* :**
  - **- les pièces n’ont pas la même progression sur les machines**
  - **- n pièces, m machines → (n!)m cédules possibles**
  - **- 10 pièces, 8 machines = 3 * 1052 cédules possibles**
  - **- solution : utilisation d’heuristiques comme la RT**

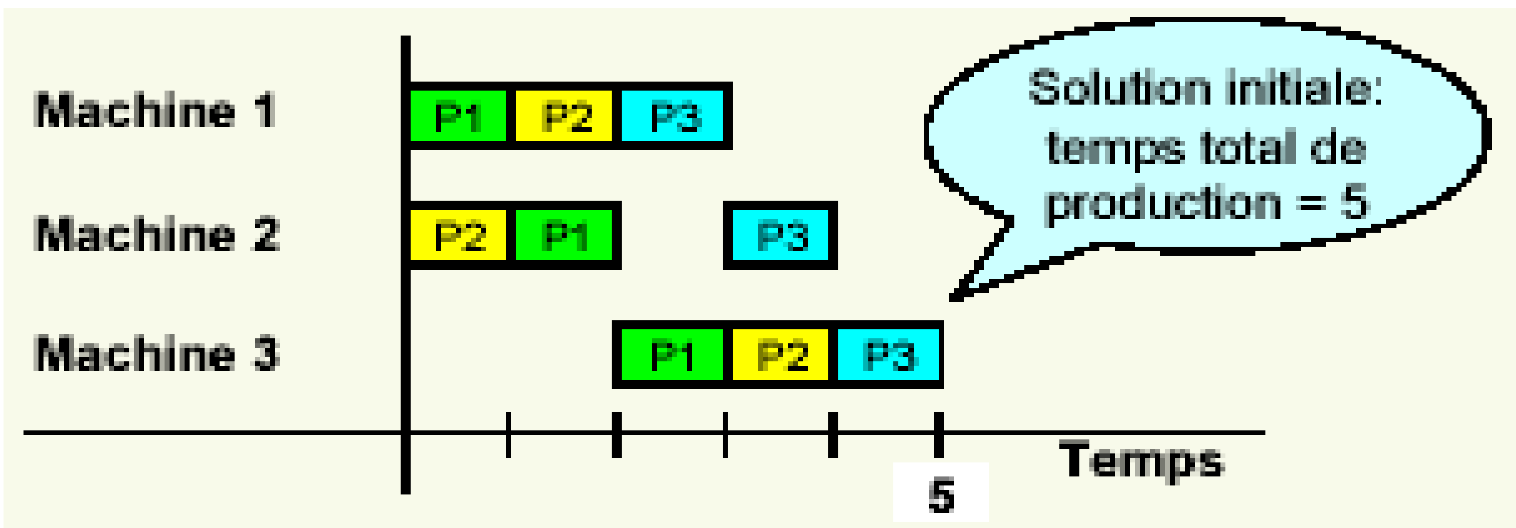
Exemple : Problème du job shop



Déterminer la valeur de la fonction objectif :

- **Fonction objectif = minimiser le temps total de production :**
  → Minimiser le plus long chemin dans le graphe

- **Diagramme de Gantt (supposons que toutes les opérations ont une durée unitaire) :**

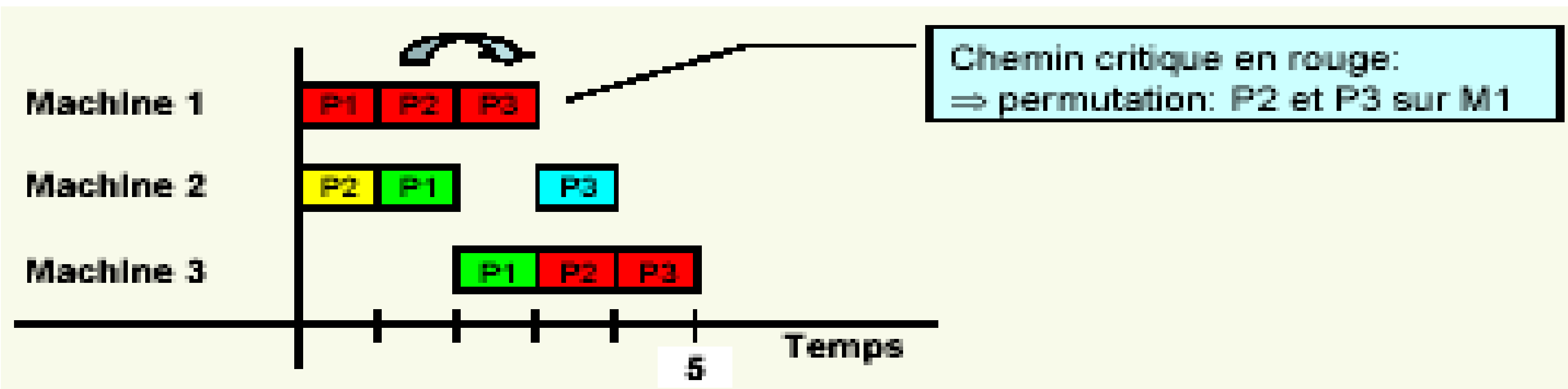


Déterminer le voisinage :

• **Utilisation de la RT pour trouver une bonne solution puisqu'il est impossible de tester le temps total de production pour toutes les cellules applicables**

• **Comment déterminer le voisinage d'une solution ?**

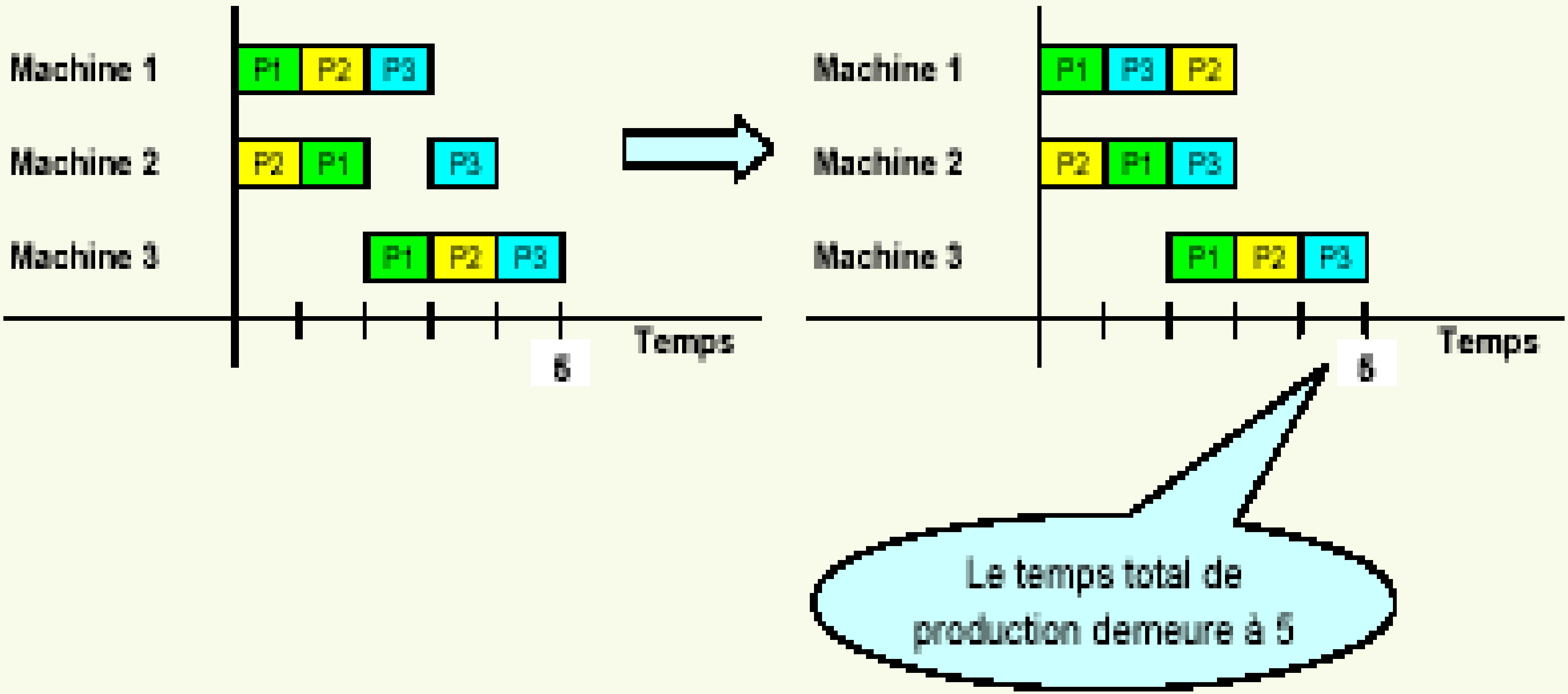
→ À partir d'une solution, on peut obtenir une solution voisine réalisable en permutant deux tâches consécutives sur le chemin critique (et sur une même machine)



Exemple : Problème du job shop

Déterminer le voisinage (suite) :

- Résultat de cette permutation :



Définition du problème :

- **Ordonnancement pour un problème de job shop**
- **n = 10 pièces, m = 10 machines**
- **Hypothèse : chaque pièce doit passer 1 fois sur chaque machine**
- **Données :**

Gammes opératoires

Pièces	Séquence sur les machines									
P1	1	2	3	4	5	6	7	8	9	10
P2	1	3	5	10	4	2	7	6	8	9
P3	2	1	4	3	9	6	8	7	10	4
P4	2	3	1	5	7	9				
P5	3	1	2	6	4	5				
P6	3	2	6	4	9	10				
P7	2	1	4	3	7	6				
P8	3	1	2	6	5	7				
P9	1	2	4	6	3	10				
P10	2	1	3	7	9	10				

Temps d'opérations

	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
P1	29	78	9	36	49	11	62	56	44	21
P2	43	28	90	69	75	46	46	72	30	11
P3	85	91	74	39	33	10	89	12	90	45
P4	71	81	95	98	99	43	9	85	52	22
P5	6	22	14	26	69	61	53	49	21	72
P6	47	2	84	95	6	52	65	25	48	72
P7	37	46	13	61	55	21	32	30	89	32
P8	86	46	31	79	32	74	88	36	19	48
P9	76	69	85	76	26	51	40	89	74	11
P10	13	85	61	52	90	47	7	45	64	76

La résolution comporte :

1. **Codage du graphe de base**

2. **Codage du calcul du plus long chemin pour la solution courante**

3. **Codage de l'algorithme permettant de déterminer les tâches critiques**

4. **Codage de l'algorithme permettant de déterminer les permutations possibles (voisinage)**

5. **Codage de l'algorithme permettant de modifier le graphe en fonction des permutations effectuées**

6. **Codage de l'algorithme de RT**

Codage de l'algorithme de RT :

1- Créer une solution initiale, qui devient la solution courante (i) :

Solution	Séquence des pièces sur les machines									
M1	1	2	3	4	5	6	7	8	9	10
M2	1	2	3	4	5	6	7	8	9	10
M3	1	2	3	4	5	6	7	8	9	10
M4	1	2	3	4	5	6	7	8	9	10
M5	1	2	3	4	5	6	7	8	9	10
M6	1	2	3	4	5	6	7	8	9	10
M7	1	2	3	4	5	6	7	8	9	10
M8	1	2	3	4	5	6	7	8	9	10
M9	1	2	3	4	5	6	7	8	9	10
M10	1	2	3	4	5	6	7	8	9	10

2- Poser : i* (solution optimale) = i, f(i*) = f(i) = 3425

Codage de l'algorithme de RT (suite) :

3- Initialiser la matrice Tabou à 0

(la matrice Tabou contient toutes les permutations qui sont taboues: une permutation est taboue pendant 10 itérations)

4- Faire :

4.1- Pour la solution courante (i), déterminer toutes les permutations possibles (voisinage): ensemble N(i)

4.2- Initialiser la variable meilleure solution voisine à 100000

4.3- Déterminer la meilleure solution voisine à i. Pour chaque permutation dans N(i), faire :

4.3.1- Permuter les tâches correspondantes dans le graphe courant i, pour obtenir i', et calculer le temps de production f(i')

Codage de l’algorithme de RT (suite) :

4.3.2- Si f(i’) est inférieur à la valeur de la variable meilleure solution voisine et que la permutation n’est pas tabou ou que f(i’) ≤ f(i*), on retient la permutation en cours et on ajuste la variable meilleure solution voisine

4.4- Modifier la solution courante (i) avec la meilleure permutation trouvée (meilleure solution voisine)

4.5- Ajouter cette permutation dans la matrice Tabou (elle sera taboue pour les 10 prochaines itérations)

4.6- Si f(i) ≤ f(i*), ajuster i* et f(i*) en fonction de i

4.7- Tant qu’il ne se passe pas 100 itérations consécutives sans amélioration de i*, revenir à 4.1

Exemple : Problème du job shop

<u>Exemple de matrice Tabou (mémoire à court terme)</u> :

Voici les 5 premières permutation effectuées:

P1	P2	M	Itération
8	9	4	11
9	8	4	11
9	10	9	12
10	9	9	12
1	2	10	13
2	1	10	13
3	4	5	14
4	3	5	14
2	3	2	15
3	2	2	15

La première permutation effectuée a été de permuter les pièces 8 et 9 sur la machine 4. Donc, les permutations 8 et 9 ainsi que 9 et 8 seront taboues jusqu'à l'itération 11.

Résultats de la RT :

- Temps de production initial: f(i*) = 3425
- Temps de production après 74 itérations: f(i*) = 1298
- Temps de résolution : court si l'algorithme est bien codé*
- Solution :

Machines	Séquences sur les machines									
M1	1	2	3	5	7	4	8	9	10	6
M2	3	1	4	7	2	5	6	8	10	9
M3	2	1	5	4	3	8	6	7	10	9
M4	1	3	2	7	5	6	9	4	10	8
M5	1	2	4	5	3	8	6	7	10	9
M6	1	2	3	5	6	8	7	9	10	4
M7	1	2	3	4	7	8	10	6	5	9
M8	1	3	2	4	5	6	7	9	8	10
M9	1	3	2	4	5	6	10	7	8	9
M10	2	1	3	7	6	5	10	4	9	8

Etant donné l'ensemble {A, B, C, D, E} des cinq premiers lettres de l'alphabet, on s'intéresse aux configurations formées par les paires non ordonnées de deux lettres différentes.
Les mouvements assurant le passage d'une configuration à une voisine sont les changements d'une seule lettre de la configuration. On souhaite gérer une liste des tabous de longueur 7.

1. Compléter le tableau ci-dessous :

Configuration courante	Mouvement effectué	Liste les mouvements tabous
AB	B remplacé par C	
	A remplacé par D	
	C remplacé par E	

2. A l'issue de ces trois mouvements, peut-on atteindre la configuration AE.

La méthode de recherche à voisinage variable (RVV)

Initialisation : Sélectionner les voisinages
V_i ,($i=1, \ldots ,I$)
Trouver une solution initiale s ;
Fin $\leftarrow$ *Faux*

Tantque non Fin **Faire**
$i \leftarrow 1$;
Tantque $i \leq I$ **Faire**
Soit $s' \in V_i(s)$ générée aléatoirement ;
$s'' \leftarrow$ *Méthode-Descente*(s') ;
Si $f(s'') \leq f(s)$ **Alors** $s \leftarrow s''$; $i \leftarrow 1$;
Sinon $i \leftarrow i+1$;
Finsi
FinFaire
Si *condition d'arrêt est rencontrée* **Alors**
Fin $\leftarrow$ *Vrai* ;
Finsi
FinFaire

N. Mladenoviç et P. Hansen (1988)