

CHAPITRE II

La méthode tabou

2.1.1. Introduction

Les méthodes d'optimisation combinatoire est les moyens d'aider à trouver une solution à certains des problèmes que nous voulons les résoudre ou vous avez besoin certaines institutions.

peuvent être réparties en deux grandes classes de méthodes pour la résolution des problèmes :

- Les méthodes exactes.
- Les méthodes approchées.

2.1.2. Les méthodes exactes

L'intérêt des méthodes exactes réside dans le fait qu'elles assurent l'obtention de la solution optimale du problème traité. En fait, elles permettent de parcourir la totalité de l'ensemble de l'espace de recherche de manière à assurer l'obtention de toutes les solutions ayant le potentiel d'être meilleures que la solution optimale trouvée au cours de la recherche. Cependant, les méthodes exactes sont très connues par le fait qu'elles nécessitent un coût de recherche souvent prohibitif en termes de ressources requises. En effet, le temps de recherche et/ou l'espace mémoire nécessaire pour l'obtention de la solution optimale par une méthode exacte sont souvent trop grands, notamment avec des problèmes de grandes tailles. De ce fait, la complexité de ce type d'algorithme croît exponentiellement avec la taille de l'instance à traiter, elle devient très importante face à des problèmes comprenant plusieurs variables, fonctions «objectif»s et/ou critères.

Il existe de nombreux algorithmes exacts y compris l'algorithme du simplexe, les algorithmes de séparation et évaluation (Branch and Bound, Branch and Cut, Branch and Price et Branch and Cut and Price, la programmation linéaire, la programmation dynamique).

2.1.2.1. La méthode de branch and bound

La méthode de branch and bound (procédure par évaluation et séparation progressive) consiste à énumérer ces solutions d'une manière intelligente en ce sens que, en utilisant certaines propriétés du problème en question, cette technique arrive à éliminer des solutions partielles qui ne mènent pas à la solution que l'on recherche. De ce fait, on arrive souvent à obtenir la solution recherchée en des temps raisonnables. Bien entendu, dans le pire cas, on retombe toujours sur l'élimination explicite de toutes les solutions du problème.

Pour ce faire, cette méthode se dote d'une fonction qui permet de mettre une borne sur certaines solutions pour soit les exclure soit les maintenir comme des solutions potentielles. Bien entendu, La performance d'une méthode de branch and bound dépend, entre autres, de la qualité de cette fonction (de sa capacité d'exclure des solutions partielles tôt).

➤ Algorithme général

Début

Placer le noeud début de longueur 0 dans une liste.

Répéter

Si la première branche contient le noeud recherché **alors**

Fin avec succès.

Sinon

- Supprimer la branche de la liste et former des branches nouvelles en étendant la branche supprimée d'une étape.
- Calculer les coûts cumulés des branches et les ajouter dans la liste de telle sorte que la liste soit triée en ordre croissant.

Jusqu'à (liste vide ou noeud recherché trouvé)

Fin

2.1.2.2. Programmation dynamique

La programmation dynamique a été appelée comme cela depuis 1940 par Richard Bellman et permet d'appréhender un problème de façon différente de celle que l'on pourrait imaginer au premier abord. Le concept de base est simple : une solution optimale est la somme de sous-problèmes résolus de façon optimale. Il faut donc diviser un problème donné en sous-problèmes et les résoudre un par un.

➤ Algorithme général de programmation dynamique

La conception d'un algorithme de programmation dynamique peut être planifiée dans une séquence de quatre étapes.

1. Caractériser la structure d'une solution optimale.
2. Définir récursivement la valeur d'une solution optimale.
3. Calculer la valeur d'une solution optimale en remontant progressivement jusqu'à l'énoncé du problème initial.
4. Construire une solution optimale pour les informations calculées.

2.1.3. Les méthodes approchées ou heuristique

Une méthode heuristique ou approchée est une méthode d'optimisation qui a pour but de trouver une solution réalisable de la fonction objective en un temps raisonnable, mais sans garantie d'optimalité. L'avantage principale de ces méthodes est qu'elles peuvent s'appliquer à n'importe quelle classe de problèmes, faciles ou très difficiles. D'un autre côté les algorithmes d'optimisation tels que les algorithmes de recuit simulé, les algorithmes tabous et les algorithmes génétiques ont démontré leurs robustesses et efficacités face à plusieurs problèmes d'optimisation combinatoires [REE95].

Les méthodes approchées englobent deux classes :

- Les méthodes constructives.
- Les métaheuristiques.

2.1.3.1. Les méthodes constructives

Ce sont des méthodes itératives qui construisent pas à pas une solution. Partant d'une solution partielle initialement vide, ils cherchent à étendre à chaque étape la solution partielle de l'étape précédente, et ce processus se répète jusqu'à ce que l'on obtienne une solution complète.

Utilisation : Les méthodes constructives sont généralement utilisables quand la qualité de solution n'est pas un facteur primordial ou la taille de l'instance est raisonnable, en l'occurrence pour générer une solution initiale dans une métaheuristique, ces méthodes rapides et faciles d'implémentation.

2.1.3.1.1. L'Algorithme glouton

Construction d'une solution réalisable en ramenant à une suite de décisions qu'on prend à chaque fois au mieux en fonction d'un critère local sans remettre en question les décisions déjà prises. Généralement, la solution obtenue est approchée.

- **Avantage :** algorithmes simples à implémenter.
- **Inconvénient :** solutions approchées obtenues plus ou moins bonnes, critère local.

2.1.3.2. Les métaheuristiques

Le mot métaheuristique est dérivé de la composition de deux mots grecs :

- heuristique qui vient du verbe heuriskein et qui signifie 'trouver'.
- méta qui est un suffixe signifiant 'au -delà', 'dans un niveau supérieur'. Les métaheuristiques se sont des méthodes inspirées de la nature, ce sont des heuristiques modernes dédiées à la résolution des problèmes et plus particulièrement aux problèmes d'optimisation, qui visent d'atteindre un optimum global généralement enfoui au milieu de nombreux optima locaux.

Les métaheuristiques qui se subdivisent en deux sous-classes :

- Les méthodes évolutives.
- Les méthodes de voisinage.

2.1.3.2.1. Les méthodes évolutives

Ces méthodes se distinguent de celles déjà étudiées par le fait qu'elles opèrent sur une population de solutions, pour cela, elles sont souvent appelées des méthodes à base de population. Certaines d'entre elles ont des principes inspirés de la génétique et du comportement des insectes. La complexité de ces deux phénomènes biologiques a servi de modèle pour des algorithmes toujours plus sophistiqués ces vingt dernières années [AIS09]. Nous avons retenu seulement par les algorithmes génétiques.

2.1.3.2.2. Les méthodes de voisinage

Ces méthodes partent d'une solution initiale (obtenue de façon exacte, ou par tirage aléatoire) et s'en éloignent progressivement, pour réaliser une trajectoire, un parcours progressif dans l'espace des solutions. Dans cette catégorie, se rangent :

- Le recuit simulé.
- La méthode Tabou le terme de *recherche locale* est de plus en plus utilisée pour qualifier ces méthodes.

2.1.3.2.2.1. Le recuit simulé

La méthode du recuit simulé est une généralisation de la méthode Monte-Carlo ; son but est de trouver une solution optimale pour un problème donné. Elle a été mise au point par trois chercheurs de la société IBM : S. Kirkpatrick, C.D. Gelatt et M.P. Vecchi en 1983, et indépendamment par V. Cerny en 1985 à partir de l'algorithme de Metropolis; qui permet de décrire l'évolution d'un système thermodynamique [HAJ03].

L'idée principale du recuit simulé tel qu'il a été proposé par Metropolis en 1953 est de simuler le comportement de la matière dans le processus du recuit très largement utilisé dans la métallurgie. Le but est d'atteindre un état d'équilibre thermodynamique, cet état d'équilibre (où l'énergie est minimale) représente - dans la méthode du recuit simulé - la solution optimale d'un problème ; L'énergie du système sera calculé par une fonction coût (ou fonction objectif). La méthode va donc essayer de trouver la solution optimale en optimisant une fonction objectif, pour cela, un paramètre fictif de température a été ajouté par Kirkpatrick, Gelatt et Vecchi. En gros le principe consiste à générer successivement des configurations à partir d'une solution initiale S_0 et d'une température initiale T_0 qui diminuera tout au long du processus jusqu'à atteindre une température finale ou un état d'équilibre (optimum global).

Algorithme de recuit simulé

Début

Construire une solution initiale s ;

Calculer la fitness $f(s)$ de s ;

Initialiser une valeur de la température T ;

$s_{best} = s$;

Tant que la condition d'arrêt n'est pas satisfaite **faire**

 Générer une solution s' voisine de s ;

 Calculer $f(s')$;

 Calculer $\Delta(f) = f(s') - f(s)$;

Si $\Delta(f) \geq 0$ **alors** // cas de maximisation

$s_{best} = s'$

$s = s'$;

Sinon Si $r < \exp(\frac{\Delta(f)}{T})$ **alors**

$s = s'$;

Fin Si

 Décroître la température T ;

Fin Tant que

Retourner s_{best} ;

Fin

2.1.3.2.2.2. La recherche tabou

La recherche tabou est une méthode d'optimisation mathématique de la famille des techniques de recherche locale présentée pour la première fois par Fred Glover en 1986 [GLO92], et elle est devenue très classique en optimisation combinatoire.

Elle se distingue des méthodes de recherche locale simples par l'introduction de la notion d'historique dans la politique d'exploration des solutions afin de diriger au mieux la recherche dans l'espace. Cette méthode s'est révélée particulièrement efficace et a été appliquée avec succès à de nombreux problèmes difficiles [GLO98].

2.2.1. Définition de La recherche tabou

La méthode de recherche tabou, ou simplement méthode tabou, a été formalisée en 1986 par F. Glover [LAM11]. Sa principale particularité tient dans la mise en œuvre de mécanismes inspirés de la mémoire humaine. L'idée consiste à garder la trace du cheminement passé dans une mémoire et de s'y référer pour guider la recherche.

- **Notion de base**

À chaque itération, l'algorithme tabou choisit le meilleur voisin non tabou, même si celui-ci dégrade la fonction de coût. Pour cette raison, on dit de la recherche avec tabou qu'elle est une méthode agressive.

- **Principe de base :**

Poursuivre la recherche de solutions même lorsqu'un optimum local est rencontré,
⇒ En permettant des déplacements qui n'améliorent pas la solution.
⇒ En utilisant le principe de mémoire pour éviter les retours en arrière (mouvements cycliques).

- **La liste tabou (mémoire)**

L'idée de base de la liste tabou consiste à mémoriser les configurations ou régions visitées et à introduire des mécanismes permettant d'interdire à la recherche de retourner trop rapidement vers ces configurations.

- **Mémoire**

⇒ Elle est représentée par une liste tabou 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.

2.2.2. Principe de la méthode

2.2.2.1. L'idée de la méthode

L'idée de la recherche tabou consiste, à partir d'une position donnée, à en explorer le voisinage et à choisir la position dans ce voisinage qui minimise la fonction objectif. Il est essentiel de noter que cette opération peut conduire à augmenter la valeur de la fonction, c'est le cas lorsque tous les points du voisinage ont une valeur plus élevée.

C'est à partir de ce mécanisme que l'on échappe aux minima locaux. Le risque cependant est qu'à l'étape suivante, on retombe dans le minimum local auquel on vient d'échapper. C'est pourquoi il faut que l'heuristique ait de la mémoire, le mécanisme consiste à interdire (d'où le nom de tabou) certains mouvements ou certaines composantes de ce mouvement (l'exemple le plus simple est d'interdire les derniers mouvements).

Les positions déjà explorées sont conservées dans ce qu'on appelle la Liste Tabou d'une taille donnée, qui est un paramètre ajustable de l'heuristique.

2.2.2.2. Algorithme tabou

- (1) *Générer* une solution initiale S de manière aléatoire
- (2) $S^* \leftarrow S$; $C^* \leftarrow F(S)$ / S^* est la meilleure solution rencontrée, C^* est son cout et F la fonction objectif
- (3) *Ajouter* S a la liste tabou ; $K \leftarrow 0$
- (4) **Répéter tant qu'**un critère de fin n'est pas vérifié
- (5) *Choisir* parmi le voisinage de S_K , $V(S_K)$, le mouvement qui minimise F et qui n'appartient pas à la liste tabou, meilleur(S_K)
- (6) $S_{K+1} \leftarrow \text{meilleur}(S_K)$
- (7) **Si** la liste tabou est pleine **alors**
- (8) *Remplacer* le dernier élément de la liste tabou par S_{K+1}
- (9) **Si non**
- (10) *Ajouter* S_{K+1} a la liste tabou
- (11) **Fin si**
- (12) **Si** ($C(S_{K+1}) < C^*$) **alors**
- (13) $S^* \leftarrow S_{K+1}$, $C^* \leftarrow C(S_{K+1})$
- (14) **Fin si**
- (15) **Fin d'algorithme**

Lorsque la mémoire est pleine, elle est gérée comme une liste circulaire en FIFO (*First In First Out*) : on élimine le plus vieux point tabou et on insère la nouvelle solution. La taille de la mémoire permet de ne pas saturer rapidement les ressources disponibles pour la

recherche et permet de surcroît d'adapter facilement la méthode à un espace de recherche dynamique.

Le critère d'arrêt :

Le critère d'arrêt sert à déterminer le moment où l'on considère que la solution trouvée est d'assez bonne qualité pour être recevable. On peut par exemple :

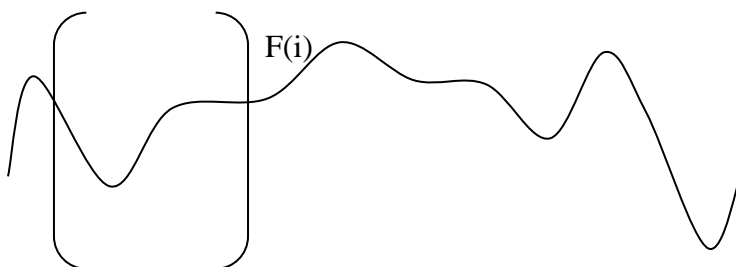
- fixer un nombre maximum d'itérations .
- après un nombre fixe d'étapes n'ayant pas amélioré la solution s^* .
- fixer un temps limite après lequel la recherche doit s'arrêter.

A partir de cet algorithme initial, certaines adaptations ont été élaborées. Ces améliorations ont été introduites afin de pallier à des problèmes constatés dans l'analyse de l'exploration de l'espace de recherche. [GLO98] recense toutes ces techniques.

2.2.2.3. Diverses améliorations

- La stratégie d'intensification

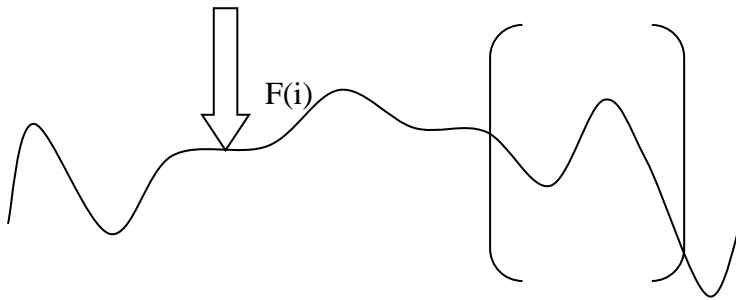
Il s'agit de repérer les éléments faisant partie des meilleures solutions trouvées, qui seront utilisées pour générer de nouvelles solutions, devant être proches de l'optimum. Par exemple, [CHA93] utilise cette technique en repartant de la meilleure solution avec une liste tabou vide. Cet examen approfondi peut permettre de dégager quelques propriétés communes définissant les régions intéressantes de l'espace de recherche. Il est alors aisé d'orienter la recherche vers ces zones « prometteuses » en rendant tabou tous les points menant à sortir de ces régions, ou bien on peut également ajouter une pénalité dans la fonction objectif pour les solutions appartenant à d'autre région. Elle est dite aussi : mémoire à moyen terme (quand une région semble contenir de bonne solution, une procédure intelligente est d'intensifier la recherche dans cette région).



- La stratégie de diversification

D'une manière symétrique, lorsque le processus de recherche parcourt une branche sur une longue période, il est possible de le stopper et de diversifier la recherche sur une autre

zone de l'espace. L'algorithme reprend alors généralement sur une autre solution générée aléatoirement. Mais il est possible d'utiliser une stratégie plus fine en mémorisant les solutions les plus fréquemment visitées et en imposant un système de pénalités, afin de favoriser les mouvements les moins souvent utilisés en forçant ainsi l'exploration de nouvelles régions [FAI92] [ROC95]. On la trouve aussi sous le nom : mémoire à long terme (car il n'y a pas la notion d'apprentissage du passé).



Donc la diversification permet de bien couvrir l'espace des solutions, et de déterminer les zones « prometteuses », tant dit que l'intensification permet d'approfondir la recherche, à l'intérieur de chacune des zones prometteuses localisées. [CHE00]

La mémoire à court terme se présente sous la forme de la liste tabou ou sont stockés les dernières solutions visitées.

2.2.3. Quelques exemples

Exemple 1

Application pour le Sudoku

Le jeu Sudoku a récemment atteint une popularité internationale, le succès de ce jeu vient probablement de la simplicité de ses règles : placer les chiffres de 1 à 9 dans une grille de 9 par 9 de telle sorte que chaque chiffre n'apparaît qu'une seule fois par ligne, par colonne et par région de 3 par 3. Ce problème peut évidemment être considéré comme un problème de satisfaction de contraintes et ainsi résolu par différentes méthodes de recherche locale dont la recherche tabou. Le Sudoku, généralisé à des grilles de $n^2 \times n^2$ à remplir avec les nombres de 1 à n^2 , est NP-complet, [TAK02].

	7		6		3		1	4
8				9				
	6	3	2	1		8		
				3			5	9
6			9			4		1
9	2		4		1			8
	5		8					3
4		1	3	6	2	7	8	
3		6	1			9		2

Un chemin de recherche locale $p = (s_1, s_2, \dots, s_n)$ avec s_i une affectation (instanciation de toutes les cellules de la grille), i.e, une grille remplie entièrement. Chaque affectation appartient à l'espace de recherche S , l'ensemble des grilles possibles.

V le voisinage du dernier élément de p tel que $V \subseteq N(s_n)$.

le couple (p, V) est la configuration de recherche locale avec $p = (s_1, s_2, \dots, s_n)$ et $V \subseteq N(s_n)$. Les fonctions de voisinage correspondent à des fonctions d'une configuration à une autre

$$(p, V) \rightarrow (p, V \cup V').$$

La méthode tabou interdit les mouvements vers des affectations déjà visitées lors des l derniers pas de recherche.

Nous utilisons une liste l de taille 10 et les fonctions Voisinage Tabou et Mouvement Meilleur, Voisinage Tabou : $p = (s_1, s_2, \dots, s_n)$ et $V' = \{s \in N(s_n) \mid \nexists k, n - l \leq k \leq n, s_k = s\}$.

Mouvement Meilleur : $s_{n+1} = s' \text{ t.q. } eval(s') = \min_{s'' \in V} eval(s'')$.

De même la recherche tabou est utilisé pour résoudre les problèmes de transport tel que le problème du voyageur de commerce, le design de réseaux, dans des problèmes d'affectation et d'allocation, dans l'optimisation de graphe (coloration de graphes), dans la télécommunication (conception de réseaux, routage d'appels), dans la logique et l'intelligence artificielle (reconnaissance et classification de formes, réseaux de neurones).

Exemple 2

Pour faciliter la compréhension des divers composants de la recherche Tabou, nous présentons son application sur un problème très connu en intelligence artificielle : le problème des reines.

Ce problème consiste à placer n reines sur un échiquier $n \times n$, de telle manière qu'aucune reine n'en capture une autre. Pour cela, il ne faut pas que deux reines soient placées sur la même ligne, la même colonne ou la même diagonale.

Si ce principe n'est pas vérifié, on parle alors de collision.

Le but de l'optimisation du problème est de minimiser le nombre de collisions.

Formulation d'une collision :

$X = \{X(1), X(2), X(3), X(4), \dots, X(n)\}$; $X(i)$ est l'index de la colonne avec :

- $X(i) \neq X(j)$ (pour que deux reines ne soient pas placées dans la même ligne ou la même colonne).

Et pour identifier les diagonales, nous remarquons que :

La somme des index des colonnes et lignes est constant pour chaque diagonale négative.

Exemple : si R1 est diagonale négative avec R3 on a : $1 + X(1) = 3 + X(3)$

La différence des index des colonnes et lignes est constant pour chaque diagonale positive.

Exemple : si R1 est diagonale positive avec R3 on a : $1 - X(1) = 3 - X(3)$

Pour représenter ce problème, nous utilisons :

- Liste Tabou : contient les mouvements interdits.
- Mouvement (permettant d'aller d'une solution à une autre) : correspond à permuter les positions des deux reines en collision.
- Critère d'aspiration : entreprendre le mouvement en respectant les contraintes de collision.

La fonction f à minimiser : minimiser le nombre de collision.

Pour illustrer la méthode et ses composants, nous utiliserons un échiquier 7×7 .

Nous supposons au début qu'une solution initiale peut être construite (d'une manière aléatoire ou intelligente), cette solution est représentée comme suit :

Itération 0 :

4	5	3	6	7	1	2
---	---	---	---	---	---	---

Cette solution indique qu'il y'a :

- Une reine ($R1$) dans la ligne1, colonne4 ;
- Une reine ($R2$) dans la ligne2, colonne5 ;
- Une reine ($R3$) dans la ligne3, colonne3 ;
-

			R1			
				R2		
		R3				
					R4	
						R5
R6						
	R7					

Cette configuration a un total de 4 collisions :

- (R1, R2)
- (R4, R5)
- (R6, R7)
- (R2, R6)

La fonction $f = \text{somme (collisions)} = 4$

Itération 1 :

Initiale

4	5	3	6	7	1	2
---	---	---	---	---	---	---

Appliquer le critère d'aspiration : l'échange de position des reines (R1, R7) est le meilleur mouvement.

2	5	3	6	7	1	4
---	---	---	---	---	---	---

	R1					
				R2		
		R3				
					R4	
						R5
R6						
			R7			

Cette configuration a un total de 2 collisions :

- (R4, R5)
- (R2, R6)

La liste Tabou contient : { (R1, R7) }

La fonction $f = \text{somme (collisions)} = 2$

Itération 2 :

Appliquer le critère d'aspiration : l'échange de position des reines (R2, R4) est le meilleur mouvement.

2	6	3	5	7	1	4
---	---	---	---	---	---	---

	R1					
					R2	
		R3				
				R4		
						R5
R6						
			R7			

Cette configuration a un total de 1 collision :

- (R1, R4)

La liste Tabou contient : { (R1, R7), (R2, R4) }

La fonction $f = \text{somme (collisions)} = 1$

Itération 3 :

Appliquer le critère d'aspiration : l'échange de position des reines (R1, R3) est le meilleur mouvement.

3	6	2	5	7	1	4
---	---	---	---	---	---	---

		R1				
					R2	
	R3					
				R4		
						R5
R6						
			R7			

Cette configuration a un total de 1 collision :

- (R1, R5)

La liste Tabou contient : { (R1, R7), (R2, R4), (R1, R3) }

La fonction $f = \text{somme}(\text{collisions}) = 1$

Itération 4 :

Appliquer le critère d'aspiration : l'échange de position des reines (R5, R7) est le meilleur mouvement.

3	6	2	5	4	1	7
---	---	---	---	---	---	---

		R1				
					R2	
	R3					
				R4		
			R5			
R6						
						R7

Cette configuration a un total de 2 collisions :

- (R3, R5)
- (R4, R5)

La liste Tabou contient : { (R1, R7), (R2, R4), (R1, R3) (R5, R7) }

La fonction $f = \text{somme (collisions)} = 2$

Itération 5 :

Appliquer le critère d'aspiration : l'échange de position des reines (R4, R7) est le meilleur mouvement.

3	6	2	7	4	1	5
---	---	---	---	---	---	---

		R1				
					R2	
	R3					
						R4
			R5			
R6						
				R7		

Cette configuration a un total de 1 collision :

- (R3, R5)

La liste Tabou contient : { (R1, R7), (R2, R4), (R1, R3), (R5, R7), (R4, R7) }

La fonction $f = \text{somme (collisions)} = 1$

Itération 6 :

Appliquer le critère d'aspiration : l'échange de position des reines (R1, R3) est le meilleur mouvement.

2	6	3	7	4	1	5
---	---	---	---	---	---	---

	R1					
					R2	
		R3				
						R4
			R5			
R6						
				R7		

Cette configuration a un total de 0 collision.

$f = \min.$

2.2.4. Avantages et inconvénients

2.2.4.1. Avantages

→ La recherche tabou à l'avantage d'être facilement paramétrable. Il existe en effet au maximum deux paramètres qui peuvent réellement influencer la recherche. Le premier consiste à bien choisir la taille de la liste tabou, Elle est généralement déterminée empiriquement et varie avec les problèmes, mais c'est une donnée primordiale : une taille trop petite peut amener l'algorithme à boucler sur une zone locale de l'espace de recherche tandis qu'une trop grande taille saturera rapidement les ressources disponibles. Le deuxième est le choix du critère d'arrêt. Ce dernier peut s'avérer difficile à déterminer pour éviter de ne prolonger trop longtemps la recherche ou de rater l'optimum global recherché [PIC04].

→ L'efficacité de la méthode Tabou fait qu'elle est largement employée dans les problèmes d'optimisation combinatoire : elle a été testée avec succès sur les grands problèmes classiques (voyageur de commerce, ordonnancement d'ateliers) et elle est fréquemment appliquée sur les problèmes de constitution de planning, de routage, d'exploration géologique, etc [AUT06].

→ Possibilité de compromis entre le temps de calcul et la qualité de solution.

→ Possibilité d'intégrer des connaissances spécifiques du problème.

2.2.4.2. Inconvénients

→ Toutes les méthodes dites méta-heuristiques [HAO99], ont une caractéristique commune: l'absence de garantie de résultat. En effet dans la majorité des cas, elles ne disposent en retour aucune information sur la qualité des solutions obtenues.

→ Difficulté de prévoir la performance (qualité et temps).

→ La méthode Tabou exige une gestion de la mémoire de plus en plus lourde à mesure que l'on voudra raffiner le procédé en mettant en place des stratégies de mémorisation complexe.

2.2.5. Comparaison avec d'autres techniques :

Si on raisonne par rapport à l'usage que font les métaheuristiques de la *fonction objectif*, on trouve que certaines la laissent « telle quelle » d'un bout à l'autre du processus de calcul, tandis que d'autres la modifient en fonction des informations collectées au cours de l'exploration – l'idée étant toujours de « s'échapper » d'un minimum local, pour avoir davantage de chance de trouver l'optimal. La *recherche tabou* est un exemple de métaheuristique qui modifie la fonction objectif.

Il y a des métaheuristiques qui ont la faculté de **mémoriser** des informations à mesure que leur recherche avance, et celles qui fonctionnent sans mémoire, en aveugle, et qui peuvent revenir sur des solutions qu'elles ont déjà examinées. Le meilleur représentant des métaheuristiques avec mémoire reste la recherche Tabou.

A l'inverse du recuit simulé qui génère de manière aléatoire une seule solution voisine $s' \in V(s)$ à chaque itération, Tabou examine un échantillonnage de solutions de $V(s)$ et retient la meilleure s' même si $f(s') > f(s)$. La recherche Tabou ne s'arrête donc pas au premier optimum trouvé.

La méthode d'acceptation d'un candidat dans la méthode du Recuit simulé est probabiliste tant dit que dans la recherche tabou elle est déterministe.

	A. du Recuit Simulé	Méthode Tabou	A. génétique	A. à colonies de fourmis hybridé
Facilité d'adaptation		-	-	-
Connaissance		+	+	+
Qualité	+	++	+	+++
Rapidité	-	-	--	--

Table (2.1) : Comparaison générale des principales métaheuristiques.

Ce tableau compare les principales métaheuristiques que nous avons vues dans les séances précédentes.

Les critères de comparaisons retenus sont les suivants :

- facilité d'adaptation au problème.

- possibilité d'intégrer des connaissances spécifiques au problème.
- qualité des meilleures solutions trouvées.
- rapidité, c'est-à-dire temps de calcul nécessaire pour trouver une telle solution.

2.2.6. Conclusion :

Dans le prochain chapitre on considère une étude de cas c'est l'application de la méthode tabou pour déterminer une solution approchée au problème d'ordonnancement de n tâches sur une seule machine afin de minimiser la somme pondérée des dates de fin.