

## CHAPITRE III

**ORDONNANCEMENT SUR UNE MACHINE AVEC  
PERIODES D'INDISPONIBILITE****1. Introduction :**

La majeure partie de la littérature dédiée aux problèmes d'ordonnancement d'ateliers se place dans le contexte où les machines nécessaires à l'exécution des tâches sont disponibles en permanence, cette hypothèse n'est pourtant pas fidèle à la réalité des entreprises, dans la mesure où pannes peuvent survenir à tout instant sur des machines les rendent indisponibles durant des périodes non connues a priori. En outre, des créneaux temporels fixes peuvent être réservés à des opérations de maintenance, de nettoyage ou de contrôle.

Dans ce travail, nous établissons la complexité du problème d'ordonnancement sur une machine à  $n$  tâches en présence des périodes d'indisponibilité de la machine pour la minimisation de la somme pondérée des retards de tâches, nous considérons le contexte déterministe et statique d'indisponibilité, Suivant la notation de *Blazwics* et al, le problème considéré est noté par :  $1,NC//\sum W_i T_i$  que nous désignons par la suite par **P1**.

L'ordonnancement sous contraintes de disponibilité des machines a largement été traité par plusieurs auteurs, citons le cas  $1,NC//\sum C_i$  et  $1,NC//\sum W_i C_i$  par *Belouadah* [BEL88] et  $m,NC//C_{max}$  par *Lee* [LEE96], [LEE97] et [LEE99], et plus récemment par *Schmidt* [SCH00].

Et puisqu'il est NP-dur au sens fort et comme les méthodes exactes requièrent un effort calculatoire qui croît exponentiellement avec la taille du problème. Pour cela, des méthodes approchées permettent de résoudre ce type de problèmes en un temps raisonnable et avec des instances de grandes tailles.

Dans ce chapitre on présente méthode métaheuristique : la méthode de la recherche tabou pour trouver une solution approchée au problème considéré.

La section 2 présente une description de ce problème, La section 3 présente quelques techniques de génération de voisinage qui seront exploités par la méthode métaheuristique

choisis, La section 4 présente la méthode de recherche tabou en détaillant les paramètres et les grandes étapes de la méthode ainsi les différentes techniques permettant son amélioration.

la section 5 est réservée à l'étude expérimentale, afin de fixer la version adéquate de l'algorithme pour la résolution du problème sujet d'étude.

La section 6 illustre l'application de la méthode au problème considéré ainsi les résultats obtenus.

La section 7 est réservée à une étude comparative entre les résultats obtenus avec différents techniques de la méthode tabou à travers une série d'expérimentations commentées.

## 2. Description du problème :

Le problème d'ordonnancement des tâches avec prise en compte des contraintes de disponibilité de machine se définit de la manière suivante :

- Une machine qui n'est pas disponible pendant  $m$  périodes  $[s_i, t_i]$  tel que  $s_i < t_i$  et  $t_i < s_{i+1}$  pour  $i=1, 2, 3, \dots, m-1$ , dont les dates et les durées sont fixées et connues à priori.
- La machine ne peut réaliser qu'une tâche à la fois et chaque tâche nécessite une seule machine.
- Une collection  $I = \{1, 2, 3, \dots, n\}$  de  $n$  tâches indépendantes où la tâche ne s'exécute qu'une seule fois.
- Les tâches sont disponibles au début de la période de l'ordonnancement ( $r_i=0$ ).
- Les tâches sont caractérisées par leurs temps de traitement  $p_i$ , leurs poids  $w_i$  et ainsi leurs dates d'échéances  $d_i$  (date à partir duquel le non exécution de la tâche  $i$  résulte un retard  $T_i = \max(0, C_i - d_i)$  ou  $C_i$  est la date de fin d'exécution de la tâche  $i$ ).
- Les tâches à ordonnancer sont strictement non préemptive, ce que signifie que l'exécution de toute opération ne peut être interrompue ni par une période d'indisponibilité, ni par la réalisation d'une autre tâche.
- L'objectif est de déterminer la séquence d'entrée des tâches sur la machine de telle sorte que la somme pondérée des retards de tâches soit minimale.

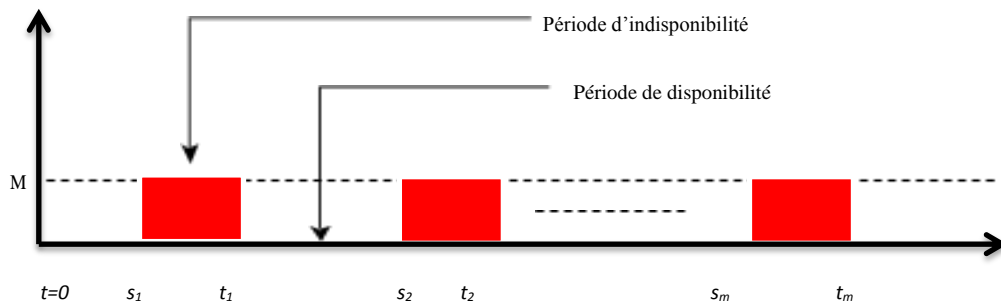


Figure 5 : Représentation des périodes d'indisponibilité

### 3. Techniques de génération de voisinage :

Les techniques de génération de voisinage sont des mécanismes destinés à générer des nouvelles séquences à partir d'une séquence admissible donnée dite séquence originale.

Ces mécanismes sont réalisés par l'application de certaines perturbations ou mouvements sur la séquence originale, notons que les techniques de voisinage sont fréquemment utilisées et exploitées par les heuristiques amélioratrices.

Parmi les techniques les plus connues on peut citer :

#### 3.1. Le voisinage par permutation de deux tâches :

Ce type de voisinage est l'ensemble de séquences obtenues par la permutation entre chaque paire de tâches dans la séquence originale, si on considère que la séquence originale est  $(J_1, J_2, J_3, \dots, J_n)$ , cette technique est basée sur les deux stratégies suivantes:

**3.1.1 la permutation simple de deux tâches adjacentes :** ce type de voisinage consiste à générer  $(n-1)$  séquences obtenues par la permutation de l'ordre de traitement de chaque paire de tâches adjacentes dans la séquence originale. L'illustration de cette stratégie est décrite dans la figure 6 (a).

**3.1.2 la permutation simple de deux tâches :** cette stratégie consiste à générer  $n(n-1)/2$  séquences obtenues par la permutation entre chaque paire de tâches (non nécessairement adjacentes) dans la séquence originale, une telle stratégie est illustrée dans la figure 6 (b).

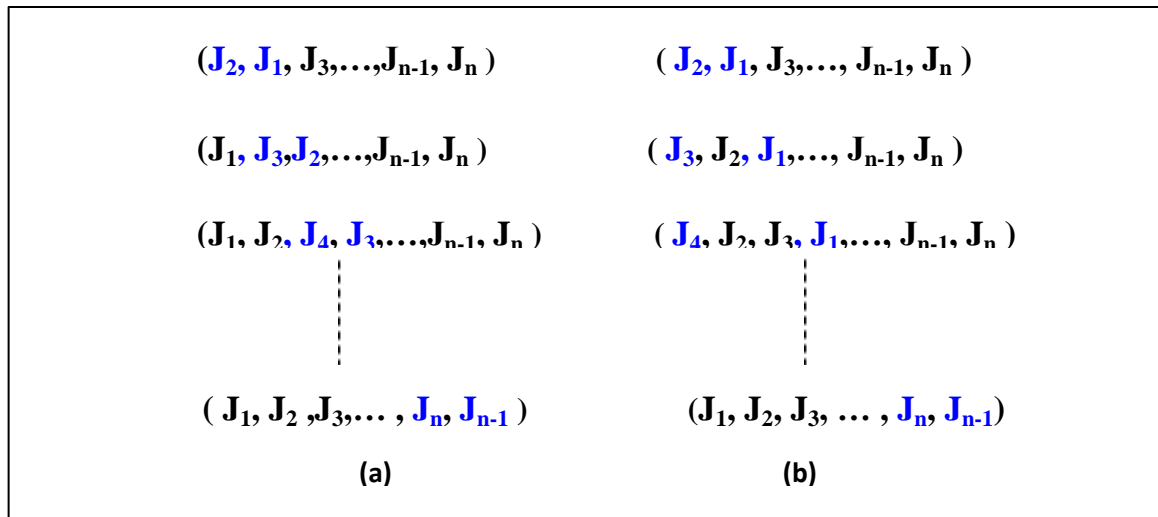


Figure 6 : Voisinage par permutation de deux tâches

**Exemple**

Soit une séquence  $\sigma = (1, 2, 3, 4)$ , le voisinage  $N(\sigma)$  est:

$$N(\sigma) = \{ (2, 1, 3, 4), (3, 2, 1, 4), (4, 2, 3, 1), (1, 3, 2, 4), (1, 4, 3, 2), (1, 2, 4, 3) \}$$

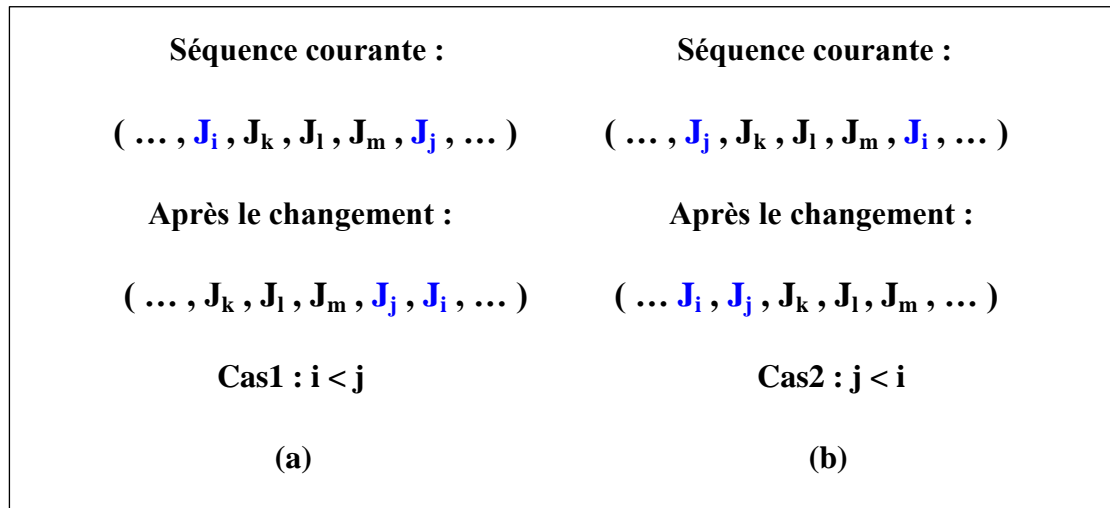
Tâche i	Tâche j	Séquence obtenue
1	2	(2, 1, 3, 4)
	3	(3, 2, 1, 4)
	4	(4, 2, 3, 1)
2	3	(1, 3, 2, 4)
	4	(1, 4, 3, 2)
3	4	(1, 2, 4, 3)

**Tableau (1)**

**3.2. Le voisinage par insertion de tâches :**

Ce type de voisinage consiste à choisir deux positions  $i$  et  $j$  tel que  $i \neq j$  à partir de la séquence courante et déplacer la tâche de la position  $i$  et l'insérer dans la position  $j$ , dans ce constat on peut distinguer deux cas possibles  $i > j$  ou  $i < j$ . Ces deux cas sont illustrés dans la figure (a) et (b) respectivement.

Cette stratégie consiste à générer  $(n-1)^2$  séquences.



**Figure 7 : Voisinage par insertion de tâches**

**Exemple**

Soit une séquence  $\sigma = (1, 2, 3, 4)$

$$N(\sigma) = \{ (2, 1, 3, 4), (2, 3, 1, 4), (2, 3, 4, 1), (1, 3, 2, 4), (1, 3, 4, 2), (3, 1, 2, 4), (1, 2, 4, 3), (4, 1, 2, 3), (1, 4, 2, 3) \}.$$

Position de tâches	1	2	3	4
1	-	(2, 1, 3, 4)	(2, 3, 1, 4)	(2, 3, 4, 1)
2	(2, 1, 3, 4)	-	(1, 3, 2, 4)	(1, 3, 4, 2)
3	(3, 1, 2, 4)	(1, 3, 2, 4)	-	(1, 2, 4, 3)
4	(4, 1, 2, 3)	(1, 4, 2, 3)	(1, 2, 4, 3)	-

Tableau (2)

### 3.3. Voisinage par insertion en block :

Ce type de voisinage consiste à choisir deux positions  $i$  et  $j$  tel que  $i \neq j$  à partir de la séquence courante de même manière de voisinage par insertion mais déplacer un block de tâches à partir la position  $i$  et l'insérer dans la position  $j$ , dans ce constat on peut distinguer deux cas possibles  $i > j$  ou  $i < j$  mêmes manières dans voisinage par insertion.

### 3.4. Hybride1 et Hybride2 : le principe de ce type de voisinage est de faire une

hybridation entre les types de voisinages précédents. En effet, pour le voisinage

hybride1, on peut appliquer la stratégie suivante :

- Chercher le meilleur voisin on appliquant la stratégie de voisinage simple par permutation de deux tâches adjacentes pour le meilleur voisin.
- Appliquer la technique de voisinage par permutation de deux tâches non-adjacentes pour améliorer cette solution.

Enfin, pour le voisinage Hybride2 on peut appliquer la stratégie suivante :

- lancer le voisinage simple par permutation de deux tâches adjacentes pour trouver le meilleur voisin.
- Appliquer la technique de voisinage par insertion de tâches pour améliorer cette solution.
- Appliquer la technique de voisinage par insertion en bloc de tâches pour une deuxième amélioration.

### Comment choisir séquence initiale ?

Le paramètre de la séquence initiale  $\sigma_0$  est un élément de l'ensemble  $\{\mathbf{R}, \mathbf{EDD}, \mathbf{SWPT}\}$  tel que :

**R** : indique la génération aléatoire de la séquence initiale.

**EDD** : indique que la séquence initiale est générée à l'aide de la règle EDD (*Earliest Due Date*), dans ce cas les tâches sont ordonnancées suivant l'ordre croissant de leurs dates

d'échues, la complexité de cette heuristique est  $O(n \log n)$ .

**SWPT** : indique que la séquence initiale est générée à l'aide de la règle SWPT (*Shortest Weighted Processing Time*), les tâches sont ordonnancées suivant l'ordre décroissant des rapports  $w_i/p_i$ , la complexité de cette heuristique est  $O(n \log n)$ .

Ces éléments permettent de montrer l'avantage de démarrer avec une bonne séquence initiale qui influe sur la qualité des solutions trouvées.

#### 4. La méthode de recherche tabou :

##### 4.1. Description et principe de la méthode :

Cette méthode a été présentée pour la première fois par F. Glover, l'idée de base consiste à introduire la notion d'histoire dans la politique d'exploration des solutions. Le nom de baptême de recherche tabou donné par Glover exprime l'interdiction de reprendre des solutions récemment visitées. Un mouvement permet de passer d'une solution valide à une autre. On dit que la nouvelle solution est un voisin de la précédente. L'ensemble des solutions que l'on peut atteindre à partir d'une solution s'appelle le voisinage.

A chaque itération tous les mouvements possibles sont examinés et le meilleur, ou plus exactement le moins mauvais est sélectionné, comme cette manière peut cycler c'est-à-dire répéter indéfiniment la même suite de mouvement, les  $t$  derniers mouvements effectués sont considérés comme interdits « tabous »,  $t$  représente la taille de la liste tabou et dépend du problème traité. Le mouvement effectivement choisi à chaque itération, est donc le meilleur mouvement non tabou.

##### 4.2. La liste tabou :

L'un des aspects critiques d'une recherche tabou est la nécessité d'ajuster la longueur de la liste tabou pour parcourir efficacement l'espace des solutions.

Si la liste est trop courte, la recherche finit par explorer un optimum local de rayon légèrement supérieur. Les différentes solutions explorées forment un cycle qui va se répéter indéfiniment.

A l'inverse, si la liste est trop longue, tous les mouvements peuvent devenir tabous et sans nouveaux voisins la recherche s'arrête, c'est un blocage.

La liste tabou est gérée selon la stratégie FIFO (**F**irst **I**n **F**irst **O**ut), on élimine le plus vieux tabou et on insère la nouvelle solution. En générale la liste tabou doit être maintenue à une longueur minimale permettant d'éviter un cycle.

La complexité d'une approche de résolution basée sur la recherche tabou dépend essentiellement de :

- La taille du voisinage de la solution courante.

- La méthode d'évaluation de chacun de ces voisins afin de déterminer celui qui minimise la fonction coût.

Comme le temps de cette évaluation est considérable pour un voisinage plus large, il intéressant d'utiliser des voisinages aussi contraints que possible afin de diminuer au maximum possible la complexité de résolution en travaillant généralement sur un sous-ensemble de voisinage. Par exemple on prend comme voisinage d'une solution toutes les solutions admissibles obtenues à partir de cette solution.

#### **4.3. Sélection du voisin :**

La sélection du meilleur voisin est souvent selon la politique du Best Fit qui permet de choisir le meilleur voisin non tabou, ou la politique du First Fit dans ce cas on sélectionne le premier voisin qui satisfait les contraintes tabous. Cette politique est souvent utilisée lorsque la taille du voisinage ne permet pas d'effectuer une évaluation complète.

#### **4.4. Critère d'aspiration :**

Les interdictions engendrées par la liste tabou peuvent s'avérer trop fortes et restreindre l'ensemble des solutions admises à chaque itération d'une manière jugée trop brutale. Le mécanisme d'aspiration, est mis en place afin de pallier cet inconvénient. Ce mécanisme permet de lever le statut tabou d'une configuration, sans pour autant introduire un risque de cycles dans le processus de recherche.

La fonction d'aspiration la plus simple consiste à révoquer le statut tabou d'un mouvement si ce dernier permet d'atteindre une solution de coût inférieur à celui de la meilleure solution trouvée jusqu'à présent.

#### **4.5. Techniques d'amélioration :**

Parmi les stratégies qui ont été proposées récemment pour améliorer l'efficacité de la méthode tabou la stratégie de l'intensification et celle de la diversification.

##### **4.5.1. L'intensification :**

On mémorise les meilleures solutions rencontrées et l'on essaie d'en dégager quelques propriétés communes pour définir des régions intéressantes vers lesquelles on oriente la recherche, par exemple en rendant tabou tous les mouvements qui font sortir de cette région.

Donc l'intensification permet de stopper périodiquement le processus normal d'exploration et intensifier l'effort de recherche dans une région qui paraît prometteuse,

Une autre manière d'appliquer l'intensification consiste à mémoriser une liste de solutions de bonne qualité et à retourner vers une de ces solutions.

##### **4.5.2. La diversification :**

A un objectif inverse de l'intensification, elle cherche à diriger la recherche vers des zones inexplorées. Sa mise en œuvre consiste souvent à mémoriser les solutions les plus

fréquemment visitées et imposer un système de pénalités, afin de favoriser les mouvements les moins souvent utilisés.

Notons que les mécanismes de l'intensification et de la diversification toutes les deux jouent un rôle complémentaire et se basant sur l'utilisation d'une mémoire à long terme mais se différencient selon la façon d'exploiter les informations de cette mémoire.

Les techniques précédentes sont très intéressantes pour améliorer la puissance de la méthode tabou, généralement elles peuvent guider le processus de recherche pour :

- Insister sur les configurations payantes.
- Quitter des configurations peu payantes.
- Rechercher des régions non encore explorées.

#### 4.5.3. Algorithme générale de la méthode tabou :

Soit  $f$  une fonction à minimiser, l'algorithme générale de la méthode de recherche tabou peut être résumé comme suit :

**Algorithme TS :**

**Debut**

**Etape1 :**

Choisir une solution initiale  $x_0$ .

Poser  $x^* \leftarrow x_0$ ,

$f^* \leftarrow f(x_0)$ ,

$k \leftarrow 0$ ,

Liste tabou  $\leftarrow \{ \}$ .

**Etape 2 :**

**Répéter tant qu'un critère de fin n'est pas vérifié**

**Etape 2.1 :** Choisir parmi le voisinage  $N(x_0)$  de  $x_0$ , le meilleur voisin non tabou, meilleur( $x_k$ ).

**Etape 2.2 :** Poser  $x_{k+1} \leftarrow \text{meilleur}(x_k)$ .

**Etape 2.3 :** Si ( $f(x_{k+1}) < f^*$ ) alors

$f^* \leftarrow f(x_{k+1})$

$x^* \leftarrow x_{k+1}$

**Etape 3.4 :** Mise à jour de la liste tabou.

$k \leftarrow k+1$ .

**Fin .**

### 5. Application de la méthode de recherche tabou au problème P1 :

Dans cette section on veut appliquer la méthode de la recherche tabou pour résoudre notre problème P1.



### 5.1. Choix des paramètres de la méthode :

Le voisinage retenu pour la recherche tabou est les six types de voisinage précédents : génération par permutation de deux tâches adjacents et non-adjacents avec un mouvement systématique et le voisinage par insertion du tâches et par insertion en bloc du tâches avec un mouvement aléatoire pour passer d'un voisin vers un autre et deux voisinage de hybridation.

La stratégie de sélection du voisin utilisée est la sélection du premier voisin non tabou dans le voisinage de la solution (séquence) courante.

La taille de la liste tabou est fixée à 10 pour tout nombre du tâches ( 50 , 100, 200,300 et 400 tâches ) ces valeurs pour la taille de la liste tabou sont déterminées empiriquement d'une façon qu'elles évitent le cyclage et le blocage du processus de recherche.

Le critère d'arrêt est choisi comme un nombre d'itérations à effectuer, ce nombre est fixé à 5000 itérations.

#### Algorithme TS :

##### Debut

Liste tabou  $\leftarrow \{ \}$

$i \leftarrow 1$

sol\_cour  $\leftarrow$  solution initiale

meil\_solution  $\leftarrow$  sol\_cour

**Tant que** (  $i \leq nb\_iter\_max$  ) **faire**

##### Debut

voisinage  $\leftarrow$  determiner\_voisinage(sol\_cour)

meil\_voisin  $\leftarrow$  meilleur\_voisin(voisinage)

ajouter\_dans\_list\_tabou(meil\_voisin)

sol\_cour  $\leftarrow$  meil\_voisin

**Si** ( coût(sol\_cour) < coût(meil\_sol) ) **alors**

meil\_sol  $\leftarrow$  sol\_cour

##### Fsi

$i \leftarrow i + 1$

##### Fin tant que

##### Fin

- La fonction *determiner\_voisinage( )* retourne l'ensemble des voisins de la solution courante suivant le type de voisinage utilisé.
- La fonction *meilleur\_voisin( )* retourne le meilleur voisin non tabou parmi les éléments de voisinage généré par la fonction *determiner\_voisinage( )*.

- La procédure *ajouter\_dans\_list\_tabou* permet la mise à jour de la liste tabou suivant la stratégie FIFO et l'ajout du nouveau voisin choisi.
- La fonction *coût()* retourne la valeur de la fonction objective d'une solution.

### 5.2. Exemple illustrative :

Soit le problème suivant de 5 tâches, avec une seule période d'indisponibilité d'une durée égale à 2. La taille de la liste tabou est fixée à 3.

<i>N</i>	1	2	3	4	5
<i>Pi</i>	7	2	3	8	4
<i>Wi</i>	2	5	6	7	1
<i>Di</i>	9	4	8	6	3

La séquence initiale générée aléatoirement est  $\sigma_0 = (2,4,3,1,5)$  d'où  $f_0 = 247$ .

On adopte la stratégie de sélection du meilleur voisin parmi tous les voisins générés par la permutation de deux tâches adjacentes (i.e on génère à chaque étape 4 voisins).

#### Initialisation :

```

nb_iter_max = 6           // le nombre d'itérations.
 $\sigma^* \leftarrow (2,4,3,1,5)$  et  $f^* \leftarrow 247$ 
list_tabou  $\leftarrow \{ \}$            // initialement la liste tabou est vide.
i  $\leftarrow 1$ 

```

#### Itération 1 : le voisinage est :

$\sigma_1$ :	(4,2,3,1,5)	(2,3,4,1,5)	(2,4,1,3,5)	(2,4,3,5,1)
$f_1$ :	153	182	173	138

Meilleur voisin non tabou

```

 $\sigma_0 \leftarrow (2,4,3,5,1)$ 
list_tabou  $\leftarrow \{(2,4,3,5,1)\}$ 
 $f_1 < f^* \Rightarrow f^* \leftarrow 138$  et  $\sigma^* \leftarrow (2,4,3,5,1)$ 
i  $\leftarrow 2$ 

```

#### Itération 2 : le voisinage est :

$\sigma_1$ :	(4,2,3,5,1)	(2,3,4,5,1)	(2,4,5,3,1)	(2,4,3,1,5)
$f_1$ :	154	183	159	137

Meilleur voisin non tabou

```

 $\sigma_0 \leftarrow (2,4,3,1,5)$ 
list_tabou  $\leftarrow \{(2,4,3,1,5), (2,4,3,5,1)\}$ 
 $f_1 < f^* \Rightarrow f^* \leftarrow 137$  et  $\sigma^* \leftarrow (2,4,3,1,5)$ 
i  $\leftarrow 3$ 

```

#### Itération 3 : le voisinage est :

$\sigma_1$ :	(4,2,3,1,5)	(2,3,4,1,5)	(2,4,1,3,5)	(2,4,3,5,1)
$f_1$ :	153	182	173	138

Meilleur voisin non tabou

Meilleur voisin mais tabou... !

$\sigma_0 \leftarrow (4,2,3,1,5)$   
 $list\_tabou \leftarrow \{(4,2,3,1,5), (2,4,3,1,5), (2,4,3,5,1)\}$   
 $i \leftarrow 4$

**Itération 4 :** le voisinage est :

$\sigma_i :$	<b>(2,4,3,1,5)</b>	(4,3,2,1,5)	(4,2,1,3,5)	(4,2,3,5,1)
$f_i :$	<b>137</b>	<b>144</b>	189	154

Meilleur voisin mais tabou... !   Meilleur voisin non tabou

$\sigma_0 \leftarrow (4,3,2,1,5)$   
mise à jour de liste tabou  
 $list\_tabou \leftarrow \{(4,3,2,1,5), (4,2,3,1,5), (2,4,3,1,5)\}$   
 $i \leftarrow 5$

**Itération 5 :** le voisinage est :

$\sigma_i :$	<b>(3,4,2,1,5)</b>	(4,2,3,1,5)	(4,3,1,2,5)	(4,3,2,5,1)
$f_i :$	147	153	175	<b>145</b>

Meilleur voisin non tabou

$\sigma_0 \leftarrow (4,3,2,5,1)$   
mise à jour de liste tabou  
 $list\_tabou \leftarrow \{(4,3,2,5,1), (4,3,2,1,5), (4,2,3,1,5)\}$   
 $i \leftarrow 6$

**Itération 6 :** le voisinage est :

$\sigma_i :$	<b>(3,4,2,5,1)</b>	(4,2,3,5,1)	(4,3,5,2,1)	(4,3,2,1,5)
$f_i :$	<b>148</b>	154	163	<b>144</b>

Meilleur voisin non tabou   Meilleur voisin mais tabou... !

$\sigma_0 \leftarrow (3,4,2,5,1)$   
mise à jour de liste tabou  
 $list\_tabou \leftarrow \{(3,4,2,5,1), (4,3,2,5,1), (4,3,2,1,5)\}$   
 $i \leftarrow 7$  .....**Stop.**

Le processus de recherche s'arrête et la meilleure solution trouvée est :

$$\sigma^* = (2,4,3,1,5) \text{ avec } \sum_{i=1}^6 W_i T_i = 137.$$

## 6. Etude expérimentale :

Dans cette section on veut présenter un ensemble des résultats expérimentaux pour déterminer les meilleurs paramètres pour l'algorithme de recherche tabou à travers une série des tests sur différentes versions pour l'algorithme, et de fixer la version la plus adéquate pour résoudre notre problème.

Notons que les algorithmes sont implémenter en Delphi sous Windows 7 et tester sur un PC ( lap top ), du processeur AMD Turion(tm) 64x2 Mobile Technology TL-50 1.6 GHZ et avec une mémoire de 1 Go, les valeurs de  $p_i$  et  $w_i$  sont générées à partir d'une distribution

uniforme discrète sur  $[1,10]$ , les  $d_i$  sont générées à partir d'une distribution uniforme dans l'intervalle  $[P(1-TF-RDD/2), P(1-TF+RDD/2)]$  tel que :

- $P = \sum_{j=1}^{j=n} p_j$
- $TF$  (*average tardiness factor*) = 0.2, 0.4, 0.6, 0.8, 1.0
- $RDD$  (*relative range of due dates*) = 0.2, 0.4, 0.6, 0.8, 1.0

Les périodes d'indisponibilité de la machine sont déterminées par la somme des temps de traitement des tâches ( $T = \sum_{j=1}^{j=n} p_j$ ) et la durée  $L$  de chaque période comme suit :  $si = \mu[T/(m+1)]$ ,  $ti = si + L$  pour  $i, \mu = 1, 2, \dots, m$ . et  $l = 1$ , Le nombre de tâches est  $n = 50, 100, 200, 300$  et  $400$  tâches. Ainsi que le nombre de périodes est  $m = 1$  et  $3$  périodes d'indisponibilité. pour chaque combinaison ( $n, m, l, TF, RDD$ ) on génère un seul problème.

La taille de la liste tabou est choisie égale à  $10$  pour tous nombre du tâches ( $50, 100, 200, 300$  et  $400$  tâches), la stratégie de sélection du voisin est la stratégie du meilleur voisin non tabou.

Et comme l'espace de recherche est très élevé si on retient une tel stratégie, à cause des tailles des instances considérées si on prend tous les voisins pour la solution courante ce qui implique que le temps d'une évaluation complète croît remarquablement. Pour cette raison on se limite de générer un sous-ensemble de nombre voisins (la taille du voisinage est fixée à (taille de voisinage/20) voisins) pour une solution courante à chaque itération de l'algorithme.

### 6.1. Le nombre d'itérations et la séquence initiale :

L'objectif de cette section est de chercher le nombre d'itérations nécessaires pour trouver la meilleure solution. Ainsi la séquence initiale adéquate pour la résolution de notre problème par la méthode de recherche tabou.

Les tables 2, 3, 4, 5, 6 et 7 respectivement les résultats obtenus par l'algorithme de recherche tabou, pour problème de taille  $50, 100, 200, 300$  tâches et  $400$  tâches où le nombre d'itérations est  $5000$  itérations, le nombre de périodes est fixé à  $1$  et  $3$  périodes d'indisponibilité d'une durée unitaire.

Les résultats de comparaison des différentes versions de l'algorithme de recherche tabou basée sur les trois heuristiques ( $R, EDD, SWPT$ ) pour générer la séquence initiale et les six types de voisinage ( $V2tadj, V2tâch, Vins, Vinsbloc, VH1, VH2$ ).

La comparaison entre la solution initiale et la solution approchée est représentée par les figures 8, 9, 10, 11, 12, 13.

Pour chaque catégorie de problème généré on calcule :

- La moyenne des valeurs de la fonction objective ( $Ofv$ ).

- Le pourcentage moyen d'amélioration (**%Ime**) par rapport à la solution donnée par la séquence initiale. Le pourcentage d'amélioration est donnée par la relation suivante:  $Ime = [(F0 - Val\_Heur)/F0] \times 100$ .

Où **Val\_Heur**, **F0** représentent la valeur de la fonction objective trouvée par l'algorithme suivant l'heuristique utilisée pour générer la séquence initiale et le type de voisinage retenu, la valeur de la fonction objective donnée par la séquence initiale.

- **F0** : La solution initiale (la valeur de la fonction objective pour la séquence initiale).
- **V2tadj** : La solution finale (la valeur de la fonction objective pour appliqué méthode de voisinage permutation deux tâches adjacents).
- **V2t** : La solution finale (la valeur de la fonction objective pour appliqué méthode de voisinage permutation deux tâches non adjacents).
- **Vins** : La solution finale (la valeur de la fonction objective pour appliqué méthode de voisinage par insertion du tâches).
- **Vblock** : La solution finale (la valeur de la fonction objective pour appliqué méthode de voisinage par insertion en bloc du tâches).
- **VH1** : La solution finale (la valeur de la fonction objective pour appliqué méthode de voisinage par hybride1).
- **VH2** : La solution finale (la valeur de la fonction objective pour appliqué méthode de voisinage par hybride2).

Table 2 : Résultats pour l'algorithme de recherche tabou

Nombre de périodes = 1 , La durée de la période = 1

N	La solution initiale est générée aléatoirement							
		F <sub>0</sub>	V2tadj	V2t	Vins	Vblock	VH1	VH2
50	Ofv	23455,4	11404,4	11404,4	11404,4	11417	11378,2	11384
	Ame%		51,378	51,378	51,378	51,324	<b>51,490</b>	51,465
100	Ofv	89913,2	44957	44947,6	44994,2	45002,8	44975,6	44948,4
	Ame%		49,999	<b>50,010</b>	49,958	49,948	49,978	50,009
200	Ofv	352278,6	257410,8	256585	254239,6	257718,4	258955,6	256606
	Ame%		26,929	27,164	<b>27,829</b>	26,842	26,491	27,158
300	Ofv	782957,2	607120,2	605013,8	599836,6	602294,6	590732	588339,8
	Ame%		22,458	22,727	23,388	23,074	24,551	<b>24,856</b>
400	Ofv	1387148,4	935901,6	928702,4	923621,4	922666,6	908996,8	902685,4
	Ame%		32,530	33,049	33,415	33,484	34,470	<b>34,925</b>

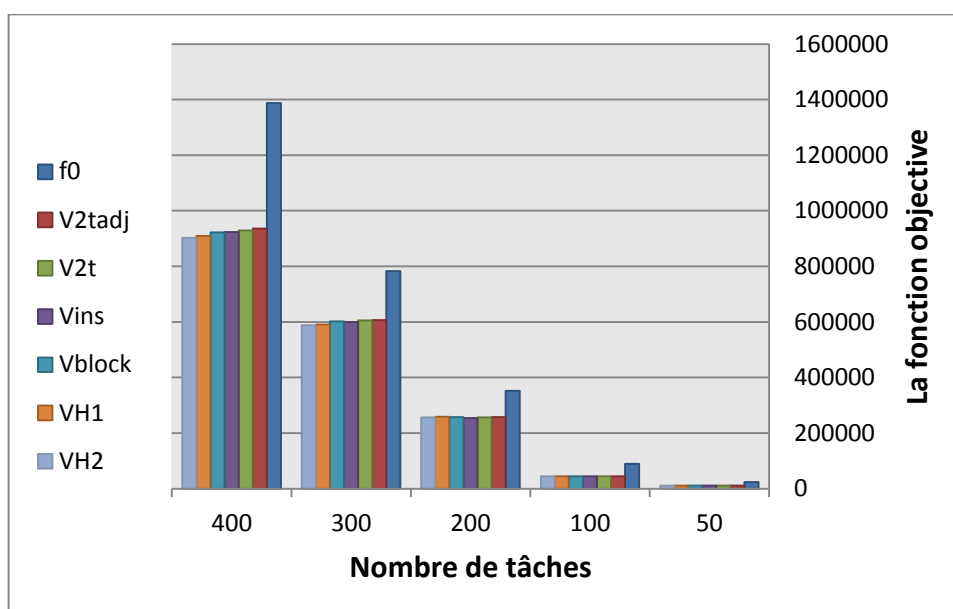


Figure 8 : Résultats sur la solution initiale et la solution approchée.  
( La solution initiale est générée aléatoirement , nombre de périodes = 1)

Table 3 : Résultats pour l'algorithme de recherche tabou

Nombre de périodes = 1 , La durée de la période = 1

N	La solution initiale est générée par la règle EDD							
		F <sub>0</sub>	V2tadj	V2t	Vins	Vblock	VH1	VH2
50	Ofv	19978	10104,4	10100,6	10100,6	10106,2	10100,6	10100,6
	Ame%		49,422	<b>49,441</b>	<b>49,441</b>	49,413	<b>49,441</b>	<b>49,441</b>
100	Ofv	79597,8	46141,8	46100,6	46135,4	46135,6	46267	46215,6
	Ame%		42,031	<b>42,083</b>	42,039	42,039	41,874	41,938
200	Ofv	305153,6	227988,2	227222,2	228898,2	228902,4	228911,2	226389,2
	Ame%		25,287	25,538	24,989	24,987	24,984	<b>25,811</b>
300	Ofv	663546,4	490467	485303,4	480881,6	479681,8	480558,2	475430,2
	Ame%		26,083	26,862	27,528	27,709	27,577	<b>28,350</b>
400	Ofv	1198486	934575	925760,8	917991,6	913872,2	898602	894776
	Ame%		22,020	22,755	23,404	23,747	25,021	<b>25,341</b>

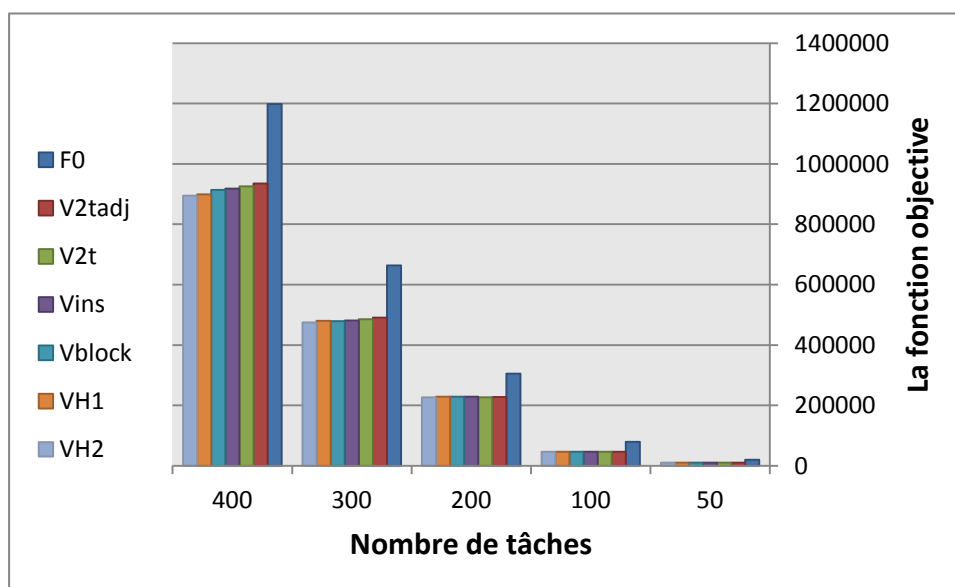


Figure 9 : Résultats sur la solution initiale et la solution approchée.

( La solution initiale est générée par la règle EDD , nombre de périodes = 1)

Table 4 : Résultats pour l'algorithme de recherche tabou

Nombre de périodes = 1 , La durée de la période = 1

N	La solution initiale est générée par la règle SWPT							
		F <sub>0</sub>	V2tadj	V2t	Vins	Vblock	VH1	VH2
50	Ofv	35843	8893,4	8889,6	8889,6	8905,4	8891,6	8900,4
	Ame%		75,187	<b>75,198</b>	<b>75,198</b>	75,154	75,192	75,168
100	Ofv	148045,6	32995,6	32995,6	32959,6	32999,6	32996,8	33004,8
	Ame%		77,712	77,712	<b>77,736</b>	77,709	77,711	77,706
200	Ofv	572750,8	147879,8	148237,6	149782,6	149103,8	148708	146612,6
	Ame%		74,180	74,118	73,848	73,967	74,036	<b>74,402</b>
300	Ofv	1296211,4	312795,8	311593,4	308222,8	308629,6	307090,4	303304,4
	Ame%		75,868	75,961	76,221	76,189	76,308	<b>76,600</b>
400	Ofv	2322081	962380,4	950336,2	947117,8	946562	937426,6	929553
	Ame%		58,555	59,073	59,212	59,236	59,629	<b>59,968</b>

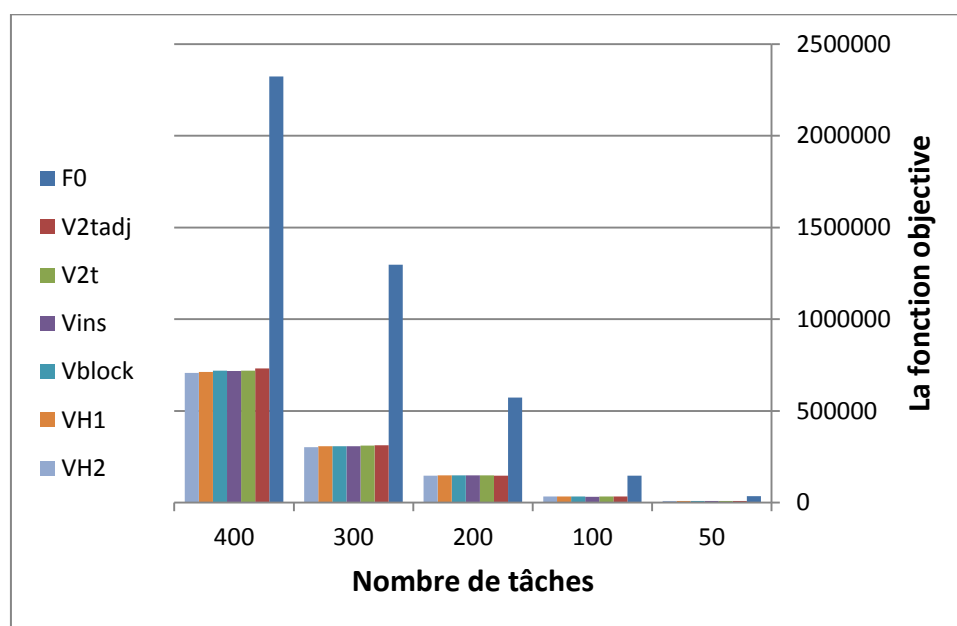


Figure 10 : Résultats sur la solution initiale et la solution approchée.

( La solution initiale est générée par la règle SWPT , nombre de périodes = 1)



Table 5 : Résultats pour l'algorithme de recherche tabou

Nombre de périodes = 3 , La durée de la période = 1

N	La solution initiale est générée aléatoirement							
		F <sub>0</sub>	V2tadj	V2t	Vins	Vblock	VH1	VH2
50	Ofv	26544,4	13730,2	13724,2	13724,2	13744,4	13732,4	13733,4
	Ame%		48,274	<b>48,297</b>	<b>48,297</b>	48,221	48,266	48,262
100	Ofv	92994,4	47479	47268,2	47479,4	47288	47056,4	47032,6
	Ame%		48,944	49,170	48,943	49,149	49,398	<b>49,424</b>
200	Ofv	361690,2	263626,4	262859,6	258076,8	258626,6	256593	256294,6
	Ame%		27,112	27,324	28,647	28,494	29,057	<b>29,139</b>
300	Ofv	783966	612468,4	609581,6	604910,4	607720,8	597278,2	595323,2
	Ame%		21,875	22,243	22,839	22,481	23,813	<b>24,062</b>
400	Ofv	1389099,6	970725,2	967544,6	962456,4	959156	940958,8	936077,6
	Ame%		30,118	30,347	30,713	30,951	32,261	<b>32,612</b>

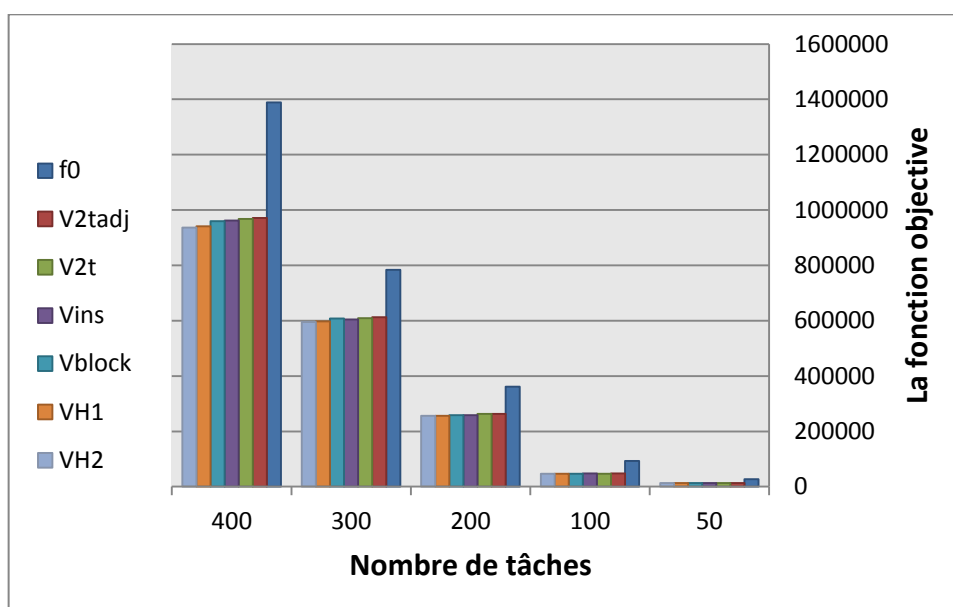


Figure 11 : Résultats sur la solution initiale et la solution approchée.  
 ( La solution initiale est générée aléatoirement , nombre de périodes = 3)

Table 6 : Résultats pour l'algorithme de recherche tabou

Nombre de périodes = 3, La durée de la période = 1

N	La solution initiale est générée par la règle EDD							
		F <sub>0</sub>	V2tadj	V2t	Vins	Vblock	VH1	VH2
50	Ofv	21715,2	12879,8	12876	12870,8	12877	12878,4	12875,8
	Ame%		40,687	40,705	<b>40,729</b>	40,700	40,694	40,706
100	Ofv	82547,2	48400,2	48566	48200,8	48201	47926,4	47854,4
	Ame%		41,366	41,165	41,608	41,607	41,940	<b>42,027</b>
200	Ofv	311932,2	225974	224606	225956,6	225322,4	224687,8	223129
	Ame%		27,556	27,995	27,562	27,765	27,969	<b>28,468</b>
300	Ofv	669761,2	494372,6	491278	486598,8	485199	481414,4	480609,2
	Ame%		26,186	26,648	27,347	27,556	28,121	<b>28,241</b>
400	Ofv	1204407	933889,4	925160	919035,4	915120,4	906609,4	903370,8
	Ame%		22,460	23,185	23,693	24,019	24,725	<b>24,994</b>

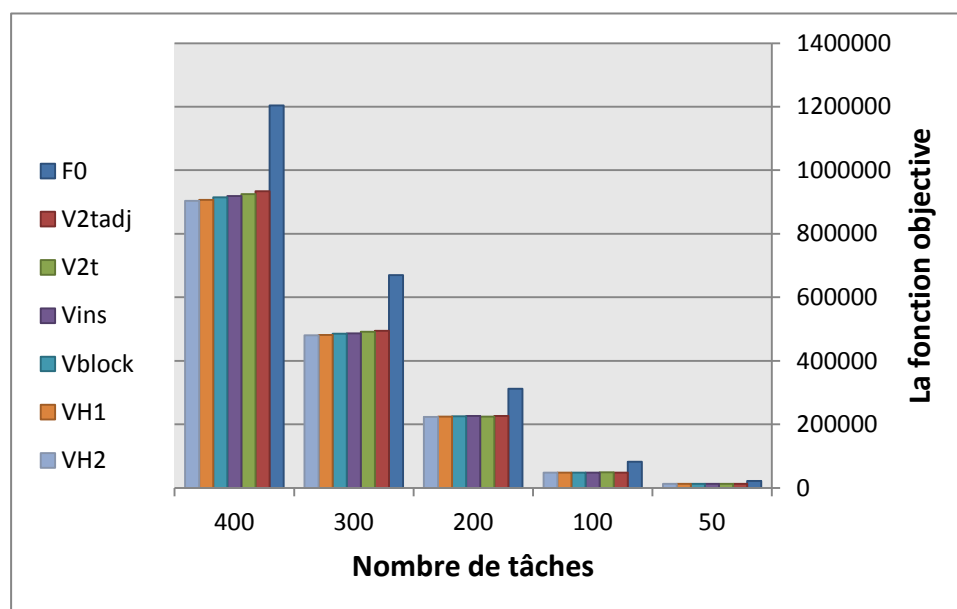


Figure 12 : Résultats sur la solution initiale et la solution approchée.

( La solution initiale est générée par la règle EDD , nombre de périodes = 3)

Table7 : Résultats pour l'algorithme de recherche tabou

Nombre de périodes = 3, La durée de la période = 1

N	La solution initiale est générée par la règle SWPT							
		F <sub>0</sub>	V2tadj	V2t	Vins	Vblock	VH1	VH2
50	Ofv	39570,4	10075,8	10075,8	10075,8	10079	10077,2	10086,4
	Ame%		<b>74,537</b>	<b>74,537</b>	<b>74,537</b>	74,528	74,533	74,510
100	Ofv	151302,4	34418	34218	34218	34419,2	34019,2	34023,6
	Ame%		77,252	77,384	77,384	77,251	<b>77,515</b>	77,512
200	Ofv	588378,4	144277,6	144911,8	146768,4	145272,6	145215,8	143361
	Ame%		75,478	75,370	75,055	75,309	75,319	<b>75,634</b>
300	Ofv	1312683	315165,4	313637,4	310682	311264,4	308880,6	291751,6
	Ame%		75,990	76,107	76,332	76,287	76,469	<b>77,774</b>
400	Ofv	2326812,6	735933,2	722430,2	720082,4	726420,8	719488,4	712266,4
	Ame%		68,371	68,951	69,052	68,780	69,078	<b>69,388</b>

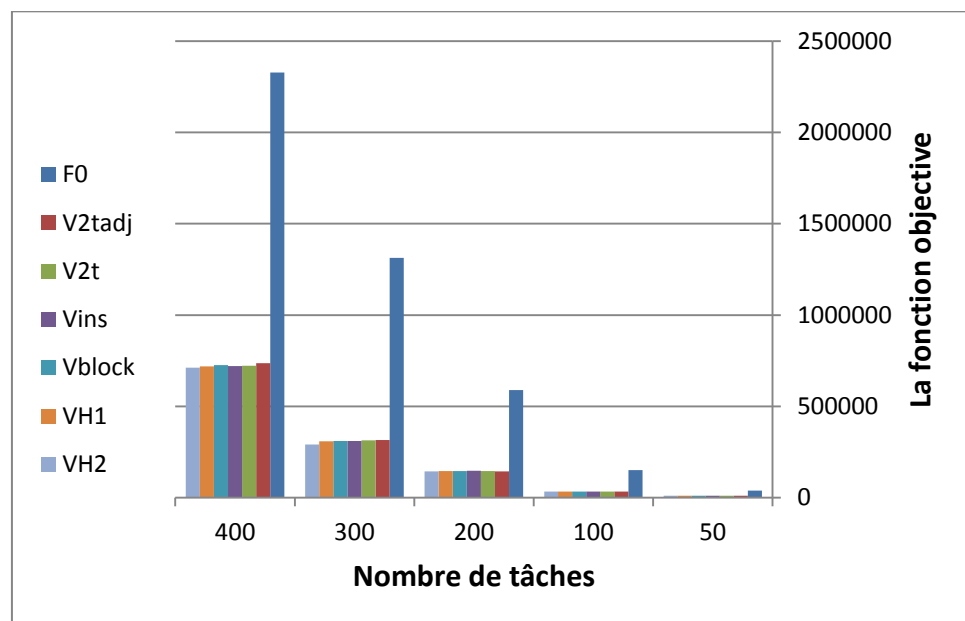


Figure 13 : Résultats sur la solution initiale et la solution améliorée.

(La solution initiale est générée par la règle SWPT, nombre de périodes = 3)

## 7. Analyse des résultats :

Une comparaison entre les valeurs de F0 obtenues par chaque heuristique de génération de la séquence initiale montre l'efficacité de l'heuristique SWPT par rapport aux valeurs obtenues par une génération aléatoire ou par l'heuristique EDD de la séquence initiale pour les différentes tailles et les différents nombres des périodes d'indisponibilité.

Les résultats de la figure 8 et 11 indiquent que :

- La solution obtenue par génération aléatoire de la séquence initiale ou voisinage hybride2 est la meilleur solution ( **R, VH2** ).

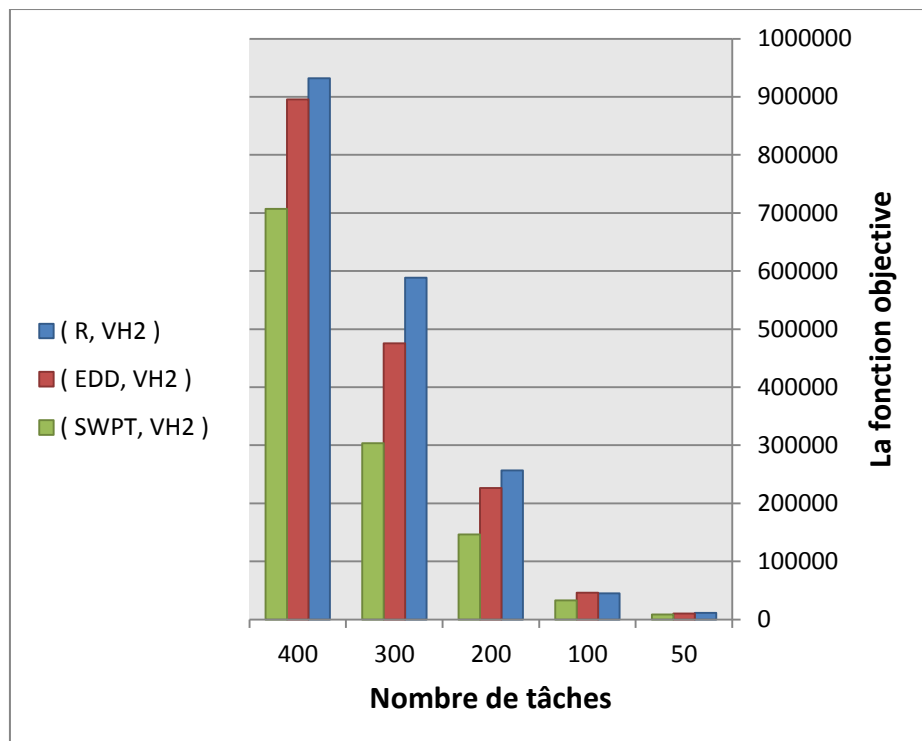
Les résultats de la figure 9 et 12 indiquent que :

- La solution obtenue par génération EDD de la séquence initiale ou voisinage hybride2 est la meilleur solution ( **EDD, VH2** ).

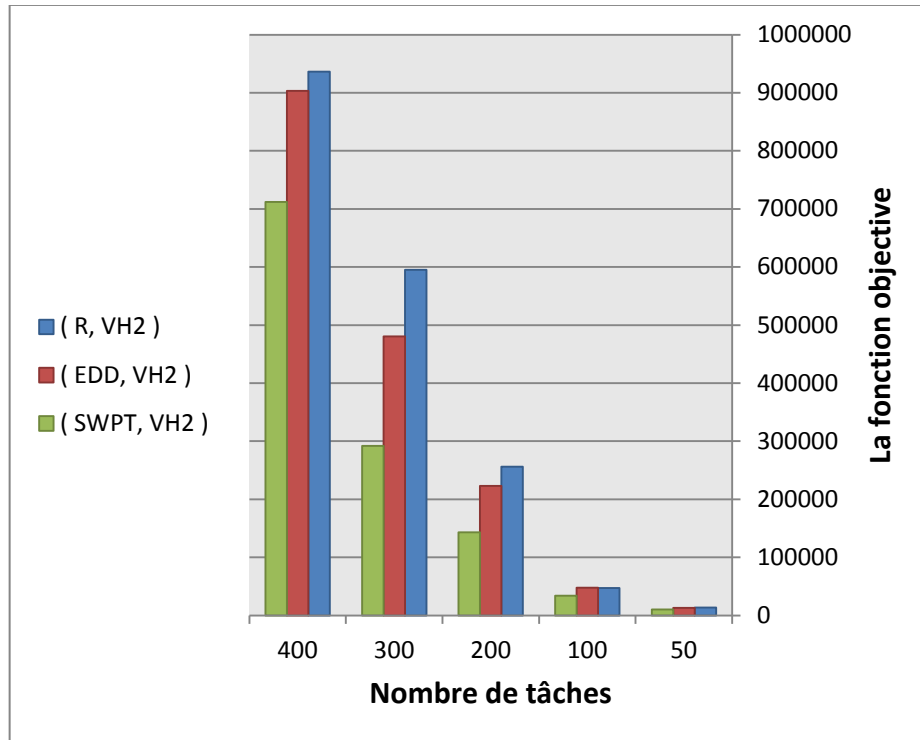
Les résultats de la figure 10 et 13 indiquent que :

- La solution obtenue par génération SWPT de la séquence initiale ou voisinage hybride2 est la meilleur solution ( **SWPT, VH2** ).

D'un point de vue du nombre de solutions meilleures, on constate que les meilleures solutions sont généralement obtenues par le voisinage généré par hybride2, ceci est pour les différentes versions de l'algorithme de recherche tabou. L'interprétation de cette remarque est que ce type de voisinage peut être considéré éventuellement riche et contient des minima de bonnes qualités.



**Figure 14 : comparaison entre les solutions approchée.**  
( Le voisinage utilisé est hybride2, nombre de périodes = 1)



**Figure 15 : comparaison entre les solutions approchée.**  
 ( Le voisinage utilisé est hybride2, nombre de périodes = 3)

Les résultats dans les figure 14 et 15 montrent que la méthode de recherche tabou si la séquence initiale générée par la règle SWPT et utilisé le voisinage hybride2 est plus puissante que les autres solutions.

## 8. Conclusion :

Avec la méthode de recherche tabou on obtient la solution la plus proche si on utilise SWPT pour choisir la séquence initiale et le voisinage hybride2 que les autres techniques de génération de séquences initiales et méthodes de voisinage, elle est moins sensible aux paramètres de réglage et permet de donner quand elle est bien appliquée de meilleurs résultats pour les problèmes d'ordonnancement. La taille de la liste tabou utilisée et la manière d'exploiter le voisinage sont des éléments très importants de cette méthode.