ANNEXES

Compléments

6.1 Compléments au Chapitre 3 : détails de l'énumération

Consulter le livre.

6.2 Compléments au Chapitre 4 : quelques résultats théoriques supplémentaires

6.2.1 Notion de overload

Une description est donnée, avec un exemple, dans (Petr Vilım, 2007) plus précisément page 23 du chapitre 2.

Soit un ensemble Ω de tâches avec $C_\Omega = \min_{i\in\Omega}(C_i)$ et $D_\Omega = \max_{i\in\Omega}(D_i)$.

Si la somme des durées des tâches de Ω dépasse la longueur de Ω, le nœud actuel de l'arborescence est inconsistant. En effet, cela revient à essayer d'ordonnancer une tâche de durée égale à la somme des durées des tâches de Ω dans un intervalle plus petit que cette somme, ce qui est bien évidemment impossible. Sur la Figure 6-, la tâche i est ordonnancée au plus tôt car $C_\Omega = C_i$, et la tâche k est ordonnancée au plus tard car $D_\Omega = D_k$. Entre ces deux tâches, il faut donc ordonnancer la tâche j, mais celle-ci ne peut pas rentrer dans l'intervalle, provoquant ainsi une zone d'overload.

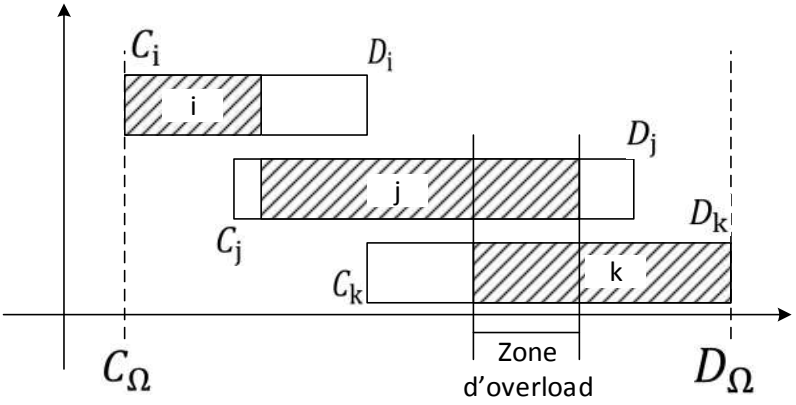

Figure 6-41. Situation d'overload

R4 : Soit Ω un ensemble de tâches ($\Omega \subset T$) avec un domaine $[C_i ; D_i]$ utilisant la ressource k.
Soit p_i la durée de la tâche i.
Soient $C_\Omega = \min_{i\in\Omega}(C_i)$ et $D_\Omega = \max_{i\in\Omega}(D_i)$.
Soit une ressource k présente en un seul exemplaire (une machine par exemple)

Si $\sum_i(p_i) > (D_\Omega - C_\Omega)$, alors le nœud est inconsistant.

C'est la définition classique de la règle overload mais, lorsqu'il s'agit de l'implémenter, une autre règle, plus générale et équivalente, est souvent privilégiée. Cette règle équivalente fait intervenir la notion de coupe "gauche" aussi appelée **LeftCut**.

Définition d'une coupe "gauche"

Soit LeftCut de j l'ensemble T des tâches i dont la date de fin au plus tard est inférieure à la date de fin au plus tard de j.

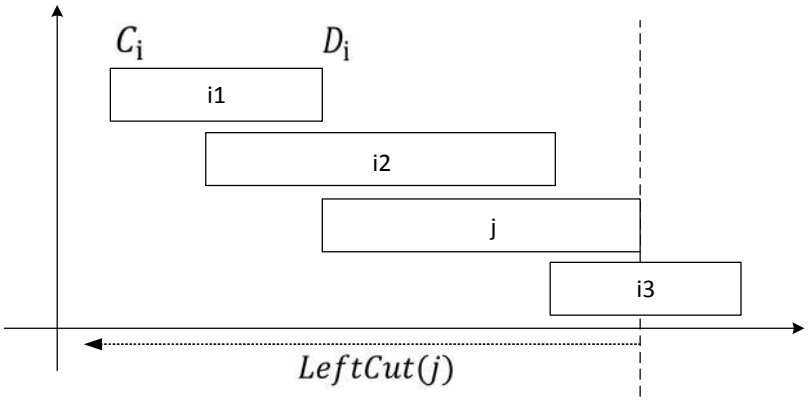

Figure 6-42. Représentation d'une coupe "gauche"

Définitions : $LeftCut(T,j) = \{i \epsilon T / LFT(i) < LFT(j)\}$.

Sur le schéma ci-dessus, $LeftCut(j) = \{i_1; i_2\}$.

R5 : Soit :
Ω un ensemble de tâches avec un domaine $[C_i; D_i]$ utilisant la ressource k.
p_i la durée de la tâche i.
Une ressource k présente en un seul exemplaire (une machine par exemple).
$C_\Omega = \min_{i\in\Omega}(C_i)$ et $D_\Omega = \max_{i\in\Omega}(D_i)$.
$LeftCut(T,j) = \{i \epsilon T / LFT(i) < LFT(j)\}$.
$ECT_\Omega = \max\{C_{\Omega'} + \sum_{i\in\Omega'}(p_i), \Omega' \backslash in \subseteq \Omega\}$.

Si $\exists i \epsilon \Omega$ tel que $ECT_{LeftCut(T,i)} > LFT_{LeftCut(T,i)}$ alors le nœud est inconsistant.

Les règles R4 et R5 sont équivalentes. La règle R5 est une généralisation directe de la règle R4 : si la date de début de la tâche commençant au plus tôt, plus la somme des durées des tâches devant être ordonnancées, est plus grande que la date de fin au plus tard de j, alors le nœud est inconsistant. La règle R5 permet de considérer des ensembles de tâches.

6.2.2 Notion de Edge Finding

L'Edge Finding est une des techniques de filtrage parmi les plus répandues. Elle consiste à identifier les situations où la position relative d'une opération i par rapport à un ensemble Ω peut être déterminée.

R6 : Soit Ω un ensemble de tâches avec un domaine $[C_i; D_i]$ utilisant la ressource k.
Soit p_i la durée de la tâche i.
Soient $C_\Omega = \min_{i\in\Omega}(C_i)$ et $D_\Omega = \max_{i\in\Omega}(D_i)$.
Soit une ressource k présente en un seul exemplaire (une machine par exemple).

Si $EST_{\Omega\cup i} + p_{\Omega\cup i} > LFT_\Omega$, alors
$\Omega \ll i$, c'est-à-dire que i est ordonnancée après la fin de toutes les opérations de Ω.
$EST_i = Max(EFT_\Omega;\ EST_i)$.

La situation à analyser est représentée ci-dessous.

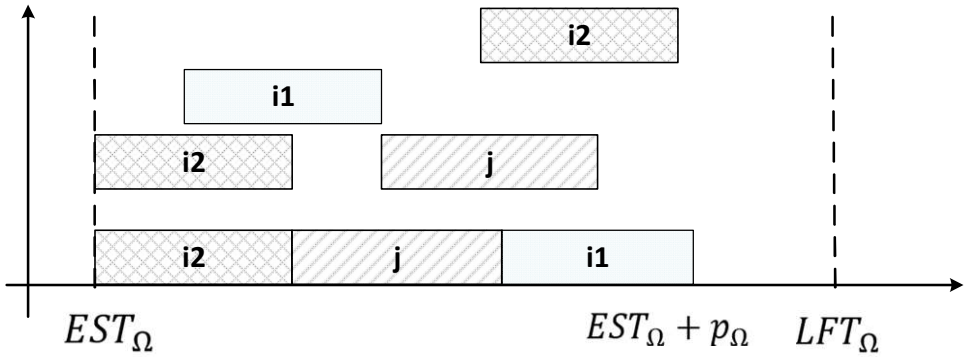

Figure 6-43. Visualisation de l'exemple

Ω est l'ensemble des activités qui respectent EST_Ω et LFT_Ω.

$$EST_\Omega = \min_{i\in\Omega}(EST_i)$$

$$LFT_\Omega = \max_{i\in\Omega}(LFT_i)$$

$$p_\Omega = \sum_{i\in\Omega} p_i$$

Il faut garder à l'esprit que les EST_i et LFT_i ne sont que des time-lags min et des time-lags max exprimés par rapport au sommet de départ du graphe.

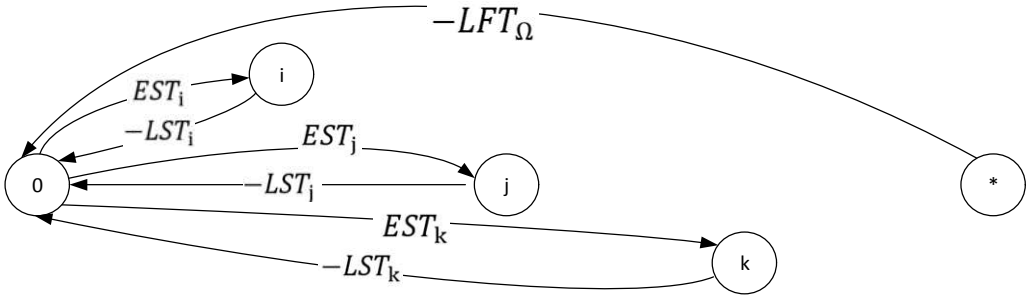

Figure 6-44. Graphe disjonctif avec les time-lags

L'objectif est d'avoir une estimation si l'ensemble des opérations i, j, k peuvent être inserés dans un interval Ω en **relaxant les contraintes** de time-lags min et max : si $EST_\Omega + p_\Omega \leq LFT_\Omega$, alors les opérations i, j et k peuvent figurer ensemble dans Ω. Traduction en français courant : "si la date de début de l'opération commençant le plus tôt (EST_Ω) plus le temps de processus de toutes les opérations est supérieur à la date de fin au plus tard de l'opération finissant au plus tard (LFT_Ω), alors cela revient à dire que la somme des temps de processus des opérations est supérieure à la largeur de la fenêtre de temps $[EST_\Omega, LFT_\Omega]$ donc c'est impossible".

Il est évident que si Ω est tel que $EST_\Omega + p_\Omega \leq LFT_\Omega$, mais que $EST_{\Omega \cup i} + p_{\Omega \cup i} > LFT_\Omega$, le time-lag max de valeur $-LFT_\Omega$ doit être mis à jour en time lag max de valeur $LFT_{\Omega \cup i}$. Cette situation est représentée sur la Figure 6-45.

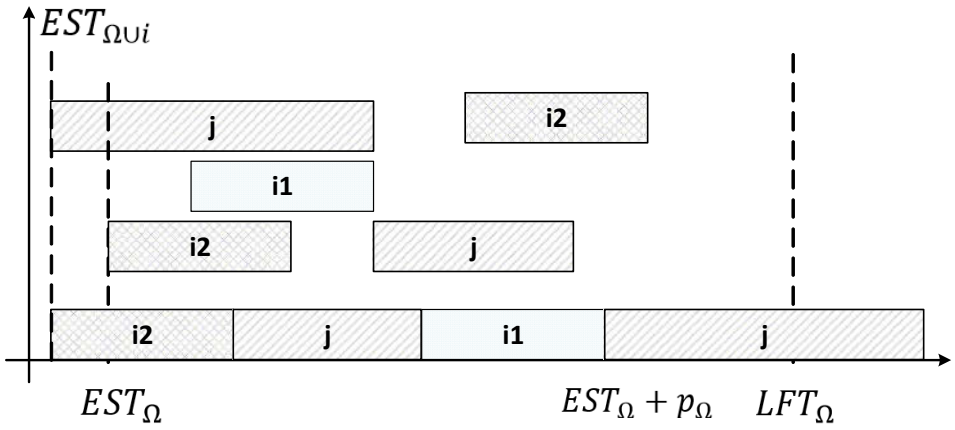

Figure 6-45. Visualisation d'une règle Edge Finding

L'exemple numérique proposé par (Petr Vilım, 2007) est retranscrit ci-dessous (Figure 6-46).

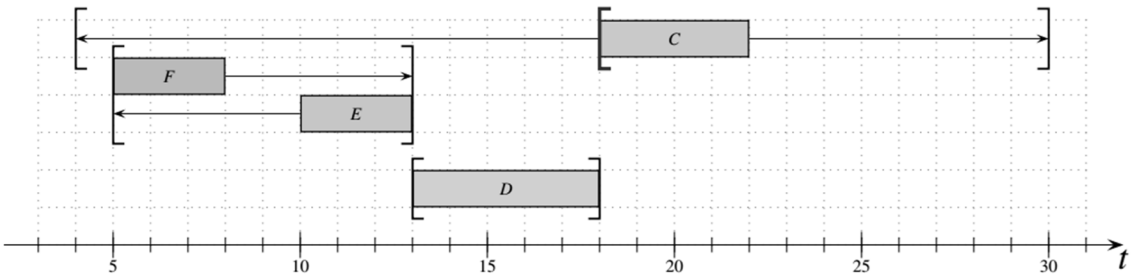

Figure 6-46. Exemple d'application d'Edge Finding

La formule se justifie comme suit :

Soit $EST_{\Omega \cup i} + p_{\Omega \cup i} > LFT_{\Omega \cup i}$

Alors $EST_{\Omega \cup i} + p_{\Omega \cup i} > LFT_{\Omega \cup i} = Max\ (LFT_\Omega; LFT_\Omega + p_i)$

$$EST_{\Omega \cup i} + p_{\Omega \cup i} > LFT_{\Omega \cup i} > LFT_\Omega$$

$$EST_{\Omega \cup i} + p_{\Omega \cup i} > LFT_\Omega$$

Pour des raisons pratiques, la règle 6 n'est jamais implémentée sous cette forme. Elle est remplacée par la règle 6b. Les règles R6 et R6b sont équivalentes.

Au lieu de considérer l'ensemble Ω, une seule activité $i \epsilon \Omega$ est considérée et la règle s'intéresse à $EFT_{LeftCut(T,j)}$ avec $LeftCut(T,j) = \{i \epsilon T / LFT(i) < LFT(j)\}$.

R6b : Soit $LeftCut(T,j)$.
Soit p_i la durée de la tâche i.
Soient $C_\Omega = \min_{i \in \Omega} (C_i)$ et $D_\Omega = \max_{i \in \Omega} (D_i)$.
Soit une ressource k présente en un seul exemplaire (une machine par exemple)

Si $EFT_{LeftCut(T,j) \cup i} > LFT_i$ alors :
$LeftCut(T,j) \ll i$.
$EST_i = Max(EFT_{LeftCut(T,j)}; EST_i)$.

Par analogie, le même raisonnement peut être reproduit en partant de $LFT_{\Omega \cup i}$. Ceci donne la règle R7.

R7 : Soit Ω un ensemble de tâches avec un domaine $[C_i ; D_i]$ utilisant la ressource k.
Soit p_i la durée de la tâche i.
Soient $C_\Omega = \min_{i \in \Omega} (C_i)$ et $D_\Omega = \max_{i \in \Omega} (D_i)$.
Soit une ressource k présente en un seul exemplaire (une machine par exemple).

Si $LFT_{\Omega \cup i} - p_{\Omega \cup i} < EST_\Omega$, alors :
$i \ll \Omega$: i est ordonnancée avant le début de toutes les opérations de Ω.
$LFT_i = Min(LST_\Omega ; LFT_i)$.

6.2.3 Notions de Not-First et Not-Last

Not-First et Not-Last sont deux conditions de propagation qui s'appliquent dans le cas de ressources présentes en un seul exemplaire.

La règle R8 considère un ensemble Ω ayant une date de fin au plus tôt $EST_\Omega + p_\Omega$. Si $EST_\Omega + p_\Omega > LST_i$ avec $LST_i = LFT_i - p_i$, alors le nœud considéré est inconsistant et un élément $j \in \Omega$ doit être supprimé. Pour rappel : $LST_j = LFT_j - p_j$.

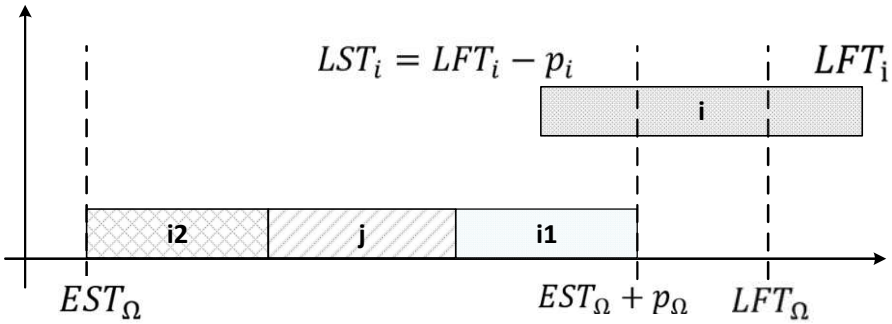

Figure 6-47. Nœud inconsistant avec une règle de type Not-Last

En considérant $Max_j(LST_j)$, c'est-à-dire la situation la plus défavorable.

R8 : Soit Ω un ensemble de tâches.
Soit p_i la durée de la tâche i.

Soient $C_\Omega = \min_{i\in\Omega}(C_i)$ et $D_\Omega = \max_{i\in\Omega}(D_i)$.
Soit une ressource k présente en un seul exemplaire (une machine par exemple)

Si $EST_\Omega + p_\Omega > LST_i$ alors $LST_i = Min(LST_i;\ Max_j LST_j)$.

6.2.4 Contraintes de précédence détectables

Les contraintes de précédence détectables permettent de réduire le domaine $[C_i; D_i]$ d'une tâche i en prenant en compte le fait que i est ordonnancée avant j et j avant k, cette situation peut être illustrée avec trois tâches comme dans l'exemple de (Petr Vilm, 2007), et présenté Figure 6-48.

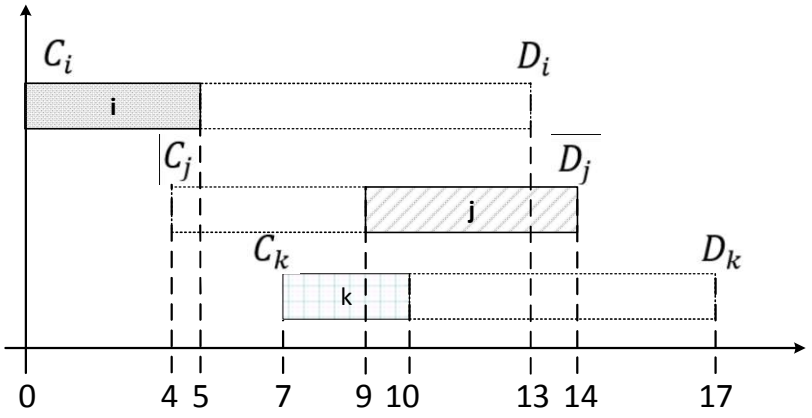

Figure 6-48. Exemple d'application de Detectable Precedence

Il est trivial de constater que le domaine de k peut être modifié pour devenir [10 ; 17], car i est ordonnancé au plus tôt, puis j, et, en respectant les contraintes de précédences, la tâche k ne pourra jamais débuter avant la date 10 (Figure 6-49).

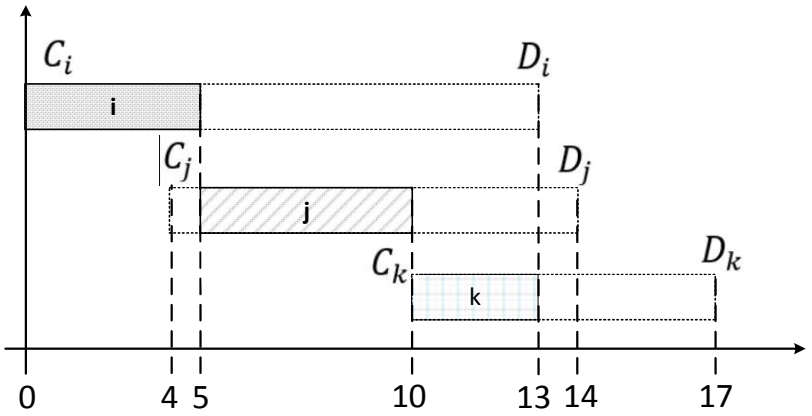

Figure 6-49. Exemple d'application de Detectable Precedence

Ce type de situation se modélise facilement sous la forme d'un graphe, le graphe initial est défini sur la Figure 6-50.

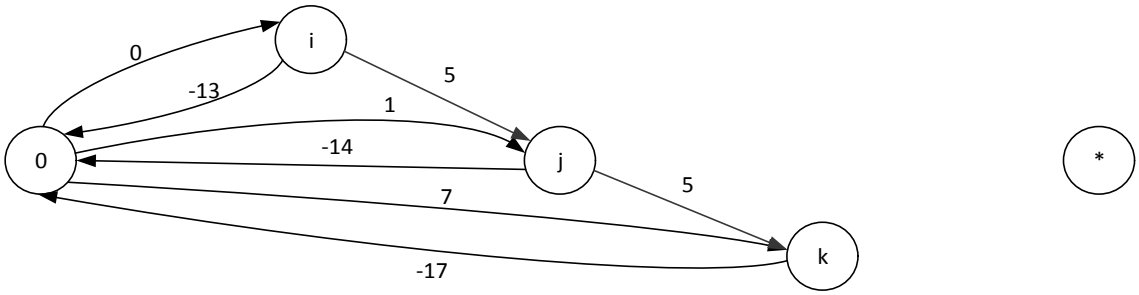

Figure 6-50. Graphe disjonctif modélisant deux contraintes de précédence

Un simple calcul de plus long chemin permet d'obtenir les dates minimales de début de i, j et k, respectivement C_i, C_j et C_k.

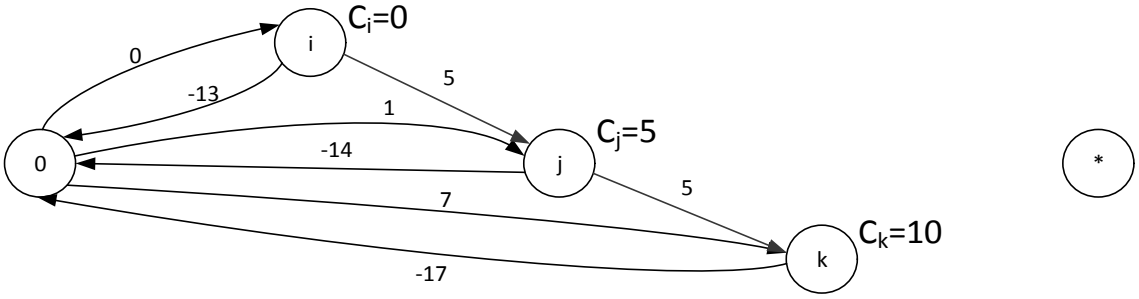

Figure 6-51. Graphe disjonctif modélisant deux contraintes de précédence après calcul

Cela veut dire que le problème de calcul des dates EST_i, EST_j et EST_k peut se traiter avec un nouveau graphe "simplifié" (Figure 6-52).

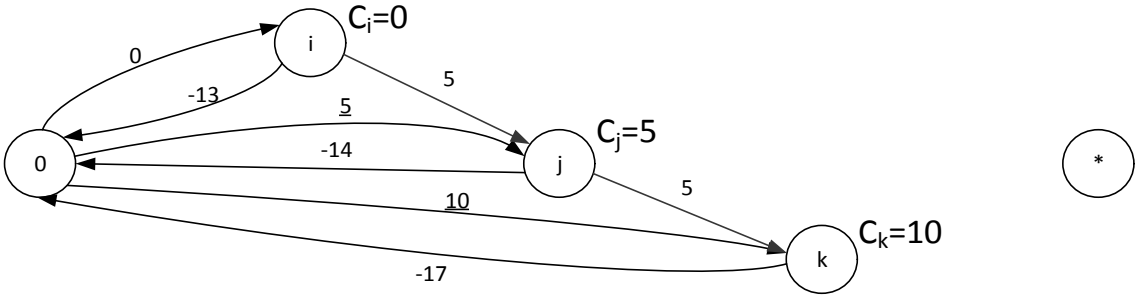

Figure 6-52. Graphe disjonctif avec les times-lags min mis à jour

Règle permettant de détecter que j est avant i

R9 : Soit i et j deux tâches utilisant la même ressource. Soit une ressource k présente en un seul exemplaire (une machine par exemple) Si $EST_i + p_i > LST_j$ alors : i ne peut pas être avant j. j est avant i.

Les situations recherchées sont celles de la Figure 6-53.

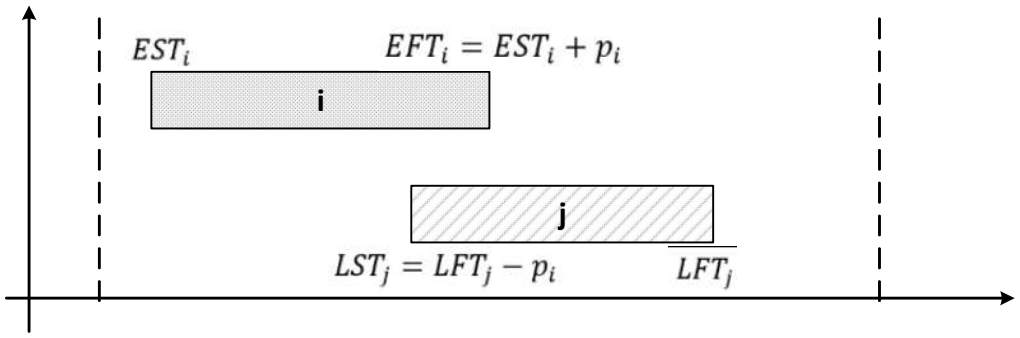

Figure 6-53. Détection de contraintes de précédence

Ceci correspond à la détection d'un cycle dans le graphe puisque la Latest Starting Time de j est un time lag max dans le graphe. Si le chemin a une longueur qui dépasse LST_j, alors un cycle est présent dans le graphe. En conclusion, i ne peut pas être avant j.

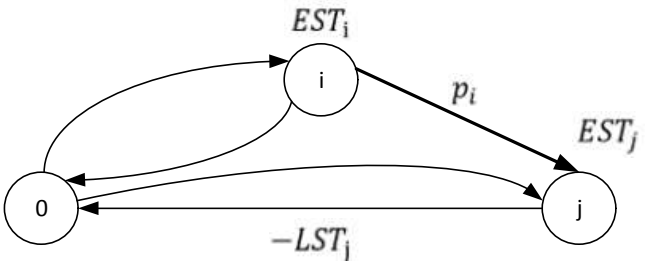

Figure 6-54. Un graphe avec un cycle potentiel

L'ensemble de tous les sommets j qui peuvent être avant i peut être caractérisé par $Dprec(T,i) = \{j \in T / EST_i + p_i \geq LST_j\}$.

En résumé, $Dprec(T,i)$ constitue l'ensemble des prédécesseurs de i.

R9 : Soit i une tâche

$Dprec(T,i) = \{j \in T / EST_i + p_i \geq LST_j\}$.
$EST_i = Max_{\ i \in Dprec(T,i)}(EST_i;\ EST_j + p_j)$.