Algorithmes Branch & Bound

Module IAD/RP/RO

Master d ’informatique Paris 6

Philippe Chrétienne

Algorithmes Branch & Bound

- Cadre général d'application.
- Arborescence valide.
- Opérations sur les arborescences valides: (troncature, séparation, évaluation, ajustement)
- Règles de dominance
- Un problème d'ordonnancement sur une machine: Minimiser la somme des retards

Problème d'optimisation

Définition :
Nom du problème : Π ;
Paramètres génériques (nombres, ensembles, fonctions, graphes, ...)
Un énoncé I de Π est une instanciation des paramètres génériques.
A chaque énoncé I correspond :

- un ensemble S(I) de solutions "réalisables";
- une fonction "objectif " $f_I : S(I) \rightarrow N^+$;

Résolution (cas d'une minimisation) :
Déterminer un algorithme A qui pour chaque énoncé I détermine une solution s*(I) de S(I) telle que :
$\forall\ s \in S(I) : f_I(s^*(I)) \leq f_I(s)$.

Exemple.

Nom: TSP (voyageur de commerce)
Paramètres génériques :

- un ensemble de n villes {1,2,...,n} ;
- une matrice $D_{n,n}$ des distances entre villes ;

Solutions réalisables d'un énoncé :
les (n-1)! permutations circulaires $(i_1,i_2,\ldots,i_n,i_1)$ des n villes ;
Fonction objectif :
$f_I(i_1,i_2,\ldots,i_n,i_1)=D(i_1,i_2)+D(i_2,i_3)+\ldots+D(i_{n-1},i_n)+d(i_n,i_1)$

Résolution :
Déterminer un algorithme qui, pour chaque matrice D, calcule un tour de longueur minimale.

Algorithme Branch and Bound

Algorithme d'énumération (intelligemment guidée) résolvant
un problème d'optimisation.

Ce type d'algorithme concerne les problèmes d'optimisation tels que
le nombre de solutions réalisables (i.e: Card(S(I))
est exponentiel en la taille de I.

Exemple: TSP
Donnée: D=[D(i,j)] est la matrice des distances entre les n villes
S(I) est l'ensemble des (n-1)! tours $(i_1,i_2,\ldots,i_n,i_1)$

Simplification des notations:
Dans la suite, on omettra dans les notations la référence à l'énoncé I.
X sera l'ensemble des solutions réalisables d'un énoncé ;
$f : X \rightarrow N^+$ sera la fonction objectif à minimiser

Principe général:
Faire évoluer un recouvrement de l'espace des solutions,
(représenté par les feuilles d'une arborescence)
jusqu'à ce que tous les feuilles de l'arborescence soient
déclarées « éliminées ».

Le calcul de cette suite d'arborescences valides utilise:

1) des fonctions d'évaluation par excès et par défaut
de la valeur optimale de sous-ensembles des solutions,

2) des règles de séparation de sous-ensembles de solutions,

3) des règles de dominance entre sous-ensembles de solutions.

Arborescence valide

Notations:
$f^*=\text{Min}\{f(x) \mid x \in X\}$ est la valeur d'une solution optimale.
f^+ est une évaluation par excès de f^* (en général la valeur de la meilleure solution connue).

Une arborescence valide est une arborescence dont les feuilles représentent le recouvrement courant de X .

Remarque :
A partir de:
- la valeur courante de f^+;
- l'évaluation par défaut des feuilles;

certaines feuilles sont marquées « éliminées » car le sous-ensemble de solutions associé ne peut contenir de solution optimale.

A est une arborescence valide pour f^+ si:

1) les sommets de A correspondent à des parties de X;
(on note X(s) la partie de X représentée par le sommet s de A),

2) la racine r de A correspond à X (soit X(r)=X);

3) si le sommet s a pour successeurs $s_1, s_2, \ldots, s_j$, alors:
$X(s)=X(s_1)\cup X(s_2) \cup \ldots \cup X(s_j)$;

4) Certains sommets de A peuvent être marqués « éliminés »;

5) Si $f^* < f^+$, alors A possède une feuille non éliminée contenant une solution optimale;

6) A toute feuille s de A est associée un nombre g(s) qui est une évaluation par défaut de la valeur de la meilleure solution de X(s):
c'est-à-dire : $\forall x \in X(s), g(s) \leq f(x)$

Opérations sur les arborescences valides

Troncature: Eliminer une feuille.

Séparation: Définir des successeurs $s_1, \ldots, s_p$ pour une feuille s de A tels que $X(s)=\cup_{i \in \{1..p\}} X(s_i)$;
on pose alors $g(s_i)=g(s)$.

Evaluation: Calculer pour une feuille s une nouvelle valeur de l'évaluation par défaut g(s) strictement plus grande.

Ajustement: Calculer une solution x^+ telle que $f(x^+) < f^+$ et faire $f^+ := f(x^+)$

Algorithme Branch & Bound:

Suite finie d'opérations de troncature, séparation, évaluation ou ajustement

- à partir de l'arborescence réduite à sa racine (arborescence valide pour $f^+ = +\infty$ et $g(X) = -\infty$ (ou mieux si possible!)
- telle que toutes les feuilles de la dernière arborescence valide sont éliminées.

Remarque:

Pour l'arborescence valide finale, on a nécessairement $f^+ = f^*$ (propriété (5) des arborescences valides).

Troncatures induites par les évaluations et les ajustements.

Soit A une arborescence valide et soit s une feuille de A.

1) Si $g(s) > f^+ = f(x^+)$, alors on peut éliminer s.

2) Si $\forall x \in X(s), f(x) \geq f^+$, alors on peut éliminer s.

Règles de dominance

Les règles de dominance fournissent des conditions suffisantes pour éliminer des sommets de l'arborescence valide en cours. Elles sont de 2 types:

Soit s une feuille d 'une arborescence valide A.
S'il existe $X' \subset X(s)$ tel que $\forall x \in X(s), \exists\, x' \in X' : f(x') \leq f(x)$,
alors on peut

1) séparer s en u et v tels que $X(u)=X'$ et $X(v)=X(s) \backslash X'$,
2) éliminer v.

Soient u et v deux feuilles d'une arborescence valide A.
Si $\forall x \in X(u), \exists\, y \in X(v) : f(y) \leq f(x)$, alors on peut éliminer u.

Principe de séparation

Soit s une feuille d'une arborescence valide A.

Les fils de s sont alors $s_1, s_2, \ldots, s_k$ où:
$\forall j \in \{1..k\}, X(s_j)=\{x \in X(s)/p_j(x) \text{ est vrai}\}$

Une séparation de s se fait à partir des propriétés
$p_1, p_2,..,p_k$ de X(s) telles que:
$\forall x \in X(s)$, x satisfait p_1 ou p_2 ou … ou p_k.

Remarques:
1) Souvent les propriétés $p_1, p_2,..,p_k$ induisent des parties 2 à 2 disjointes de X(s).
2) Souvent k=2.

Le choix du prochain sommet à séparer peut être réalisé
de diverses façons, les 2 suivantes sont les plus utilisées:

Profondeur:
On choisit la feuille la plus récemment créée de l'arborescence valide en cours
Remarques:
1) Une pile permet d 'implémenter efficacement cette règle de choix
2) Cette règle permet d'obtenir rapidement des solutions réalisables.

Evaluation minimale:
On choisit la feuille s dont l'évaluation g(s) est minimale
Remarques:
1) Une file de priorité (tas) permet d 'implémenter efficacement cette règle de choix
2) Cette règle favorise l'obtention de bonnes solutions

Un exemple: minimiser la somme des retards d 'un ensemble de tâches éxécutées sur une machine

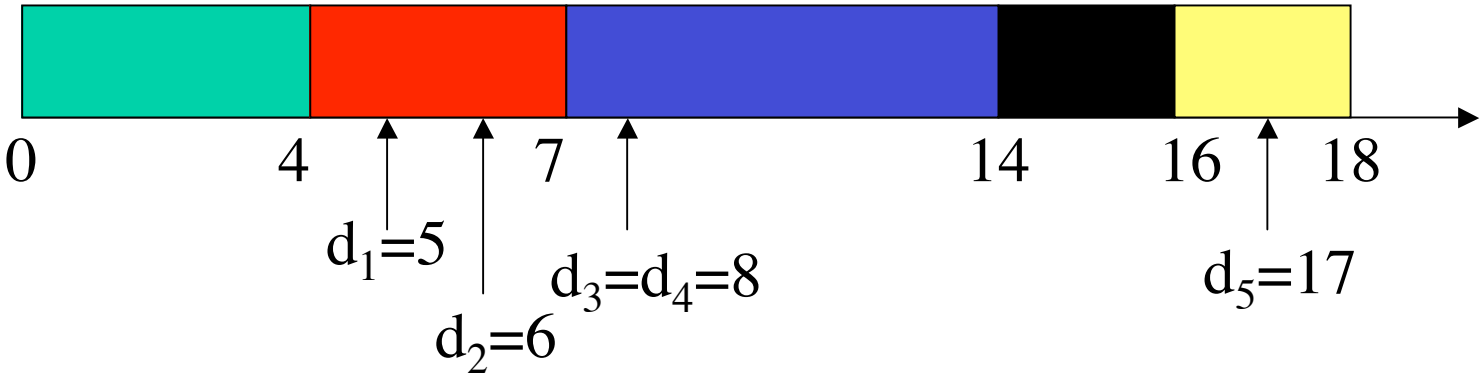
Données: un ensemble E =$\{T_1, T_2, \ldots, T_n\}$de n tâches
une machine M
$\forall i$, p_i:durée de l'exécution de T_i sur M
d_i: deadline de la tâche T_i.

Problème: Déterminer un séquencement de l'exécution des n tâches sur M tel que la somme des retards des tâches soit minimale.

Un énoncé: 5 tâches

tâche	T_1	T_2	T_3	T_4	T_5
durée	4	3	7	2	2
deadline	5	6	8	8	17

Un séquencement: (T_1,T_2,T_3,T_4,T_5)



$F((T_1,T_2,T_3,T_4,T_5))= 0 + 1 + 6 + 8 + 1$

Un sommet de l'arborescence correspond à une liste fixée $D=(T(i_1), \ldots T(i_r))$ des r dernières tâches du séquencement :

la tâche $T(i_r)$ est exécutée à θ-$p(T(i_r))$,

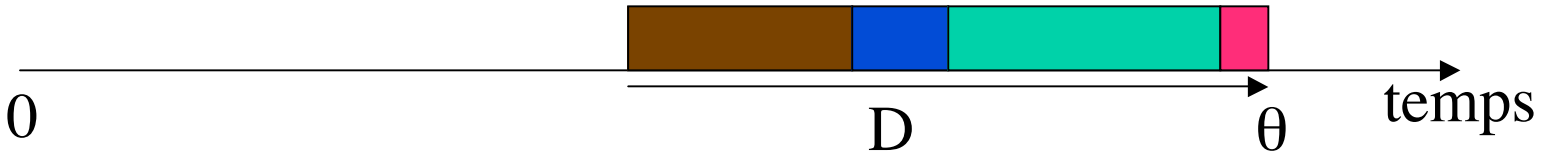
la tâche $T(i_{r-1})$ est exécutée à θ-$[p(T(i_r))+p(T(i_{r-1}))]$,

……..

la tâche $T(i_1)$ est exécutée à θ-$[p(T(i_r))+\ldots+p(T(i_1))]$,

où $\theta = \sum_{i=1}^{n} p(T_i)$.

Les (n-r) premières tâches ne sont pas arbitrées



La racine correspond à la liste vide : D = () .

Séparation:

On sépare le sommet associé à D sur la dernière tâche de la partie non encore arbitrée :

si $D= (T(i_1), \ldots T(i_r))$, alors le sommet est séparé en les n-r sommets associés à $(T_j)D$, $j \notin \{i_1, \ldots i_r\}$.

Evaluation par défaut.

Soit D la liste des dernières tâches associée à une feuille s.

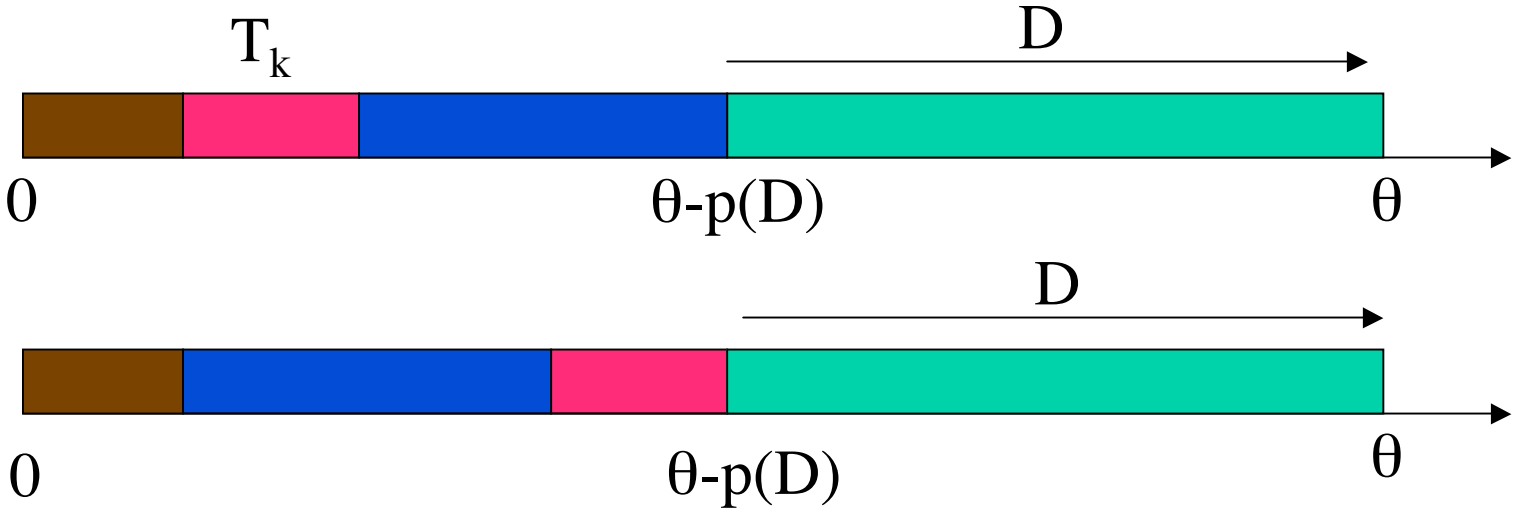
L'évaluation g(s) est la somme des retards des tâches de D.

Première règle de dominance:

Soit s une feuille caractérisée par la liste D.

Soit T_k une tâche n'appartenant pas à D.

Si $d_k \geq \theta - p(D)$, alors les séquencements caractérisés par la liste $(T_k).D$ sont dominants dans X(s).



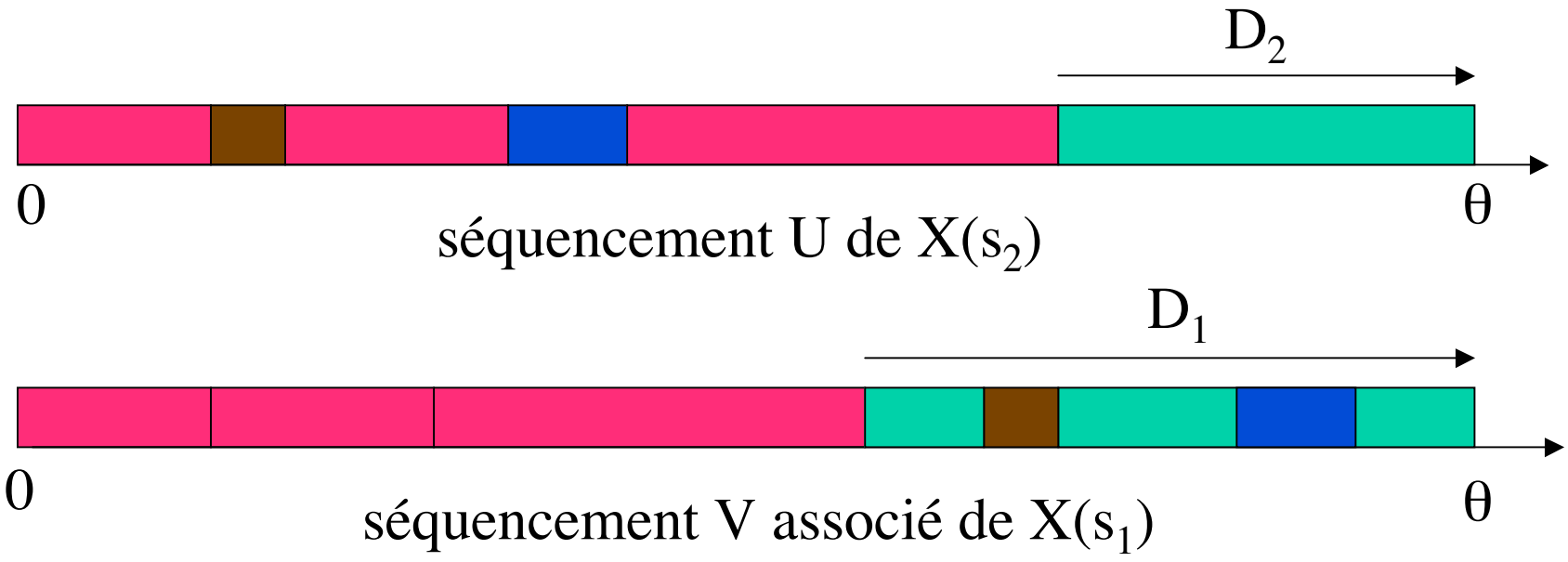
Si une feuille s est caractérisée par la liste D, on note
T(D) l 'ensemble des tâches de D et
R(D) le retard lié aux tâches de D.

Deuxième règle de dominance

Considérons deux feuilles s_1 et s_2 caractérisées par les listes respectives D_1 et D_2.
Si $R(D_2) \geq R(D_1)$ et $T(D_2) \subset T(D_1)$ alors s_1 domine s_2.

On part d 'un séquencement quelconque U de $X(s_2)$;
On construit un meilleur séquencement V de $X(s_1)$.

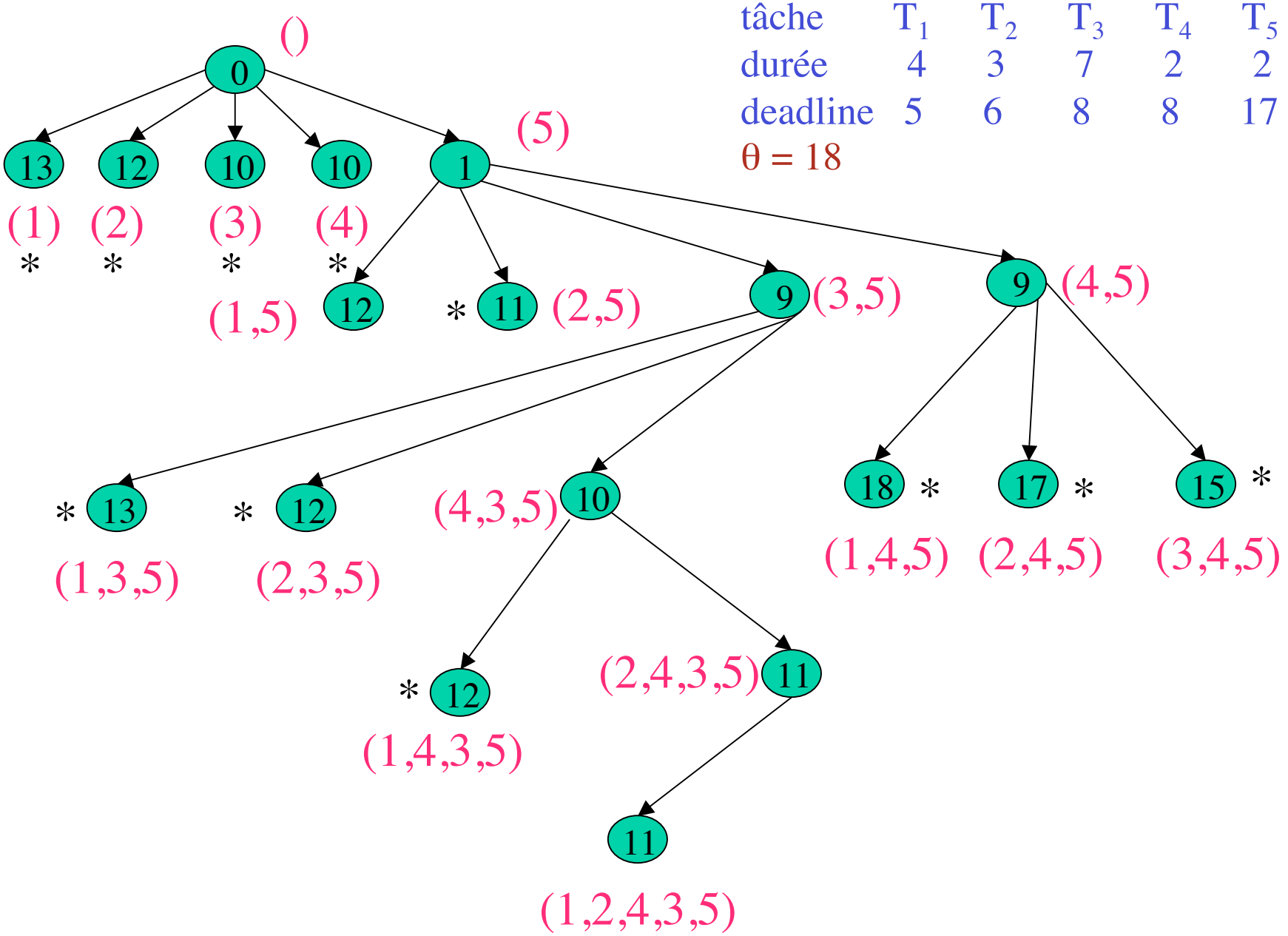
Si F est une sous-ensemble de tâches et si x est un séquencement , on note $R_x(F)$ la somme des retards des tâches de F dans le séquencement x.



Les tâches rouges (de E \ T(D1)) sont
- soit ordonnancées à la même date ;
- soit avancées;
donc $R_V(E \setminus T(D1)) \leq R_U(E \setminus T(D1))$.

$R_U(E) = R(D_2) + R_U(T(D_1) \setminus T(D_2)) + R_U(E \setminus T(D1))$.
$R_V(E) = R(D1) + R_V(E \setminus T(D1))$.

Il en résulte que V est meilleur que U.


tâche T1 T2 T3 T4 T5
durée 4 3 7 2 2
deadline 5 6 8 8 17
θ = 18
()
0
13
12
10
10
1
(5)
(1)
(2)
(3)
(4)
*
*
*
*
(1,5)
12
*
11
(2,5)
9
(3,5)
9
(4,5)
*
13
(1,3,5)
*
12
(2,3,5)
(4,3,5)
10
18
*
(1,4,5)
17
*
(2,4,5)
15
*
(3,4,5)
*
12
(1,4,3,5)
(2,4,3,5)
11
11
(1,2,4,3,5)