Les Systèmes Temps Réel

Chapitre 5:
Ordonnancement des tâches dépendantes

Cours Master 1 GLSD

2019/2020

Pr. Salim BITAM

Département d'Informatique, Université de Biskra

07000 Biskra, Algérie

C’est quoi une dépendance ?

- Contraintes de précédence

- Contraintes par partage de ressources

Contraintes de précédence

- Une contrainte sur l'ordre d'exécution des tâches
- Représenter par un graphe orienté G:

- $J_a < J_b$ indique que la tâche J_a est un prédécesseur de J_b
- $J_a \rightarrow J_b$ indique que la tâche J_a est un prédécesseur immédiat de J_b

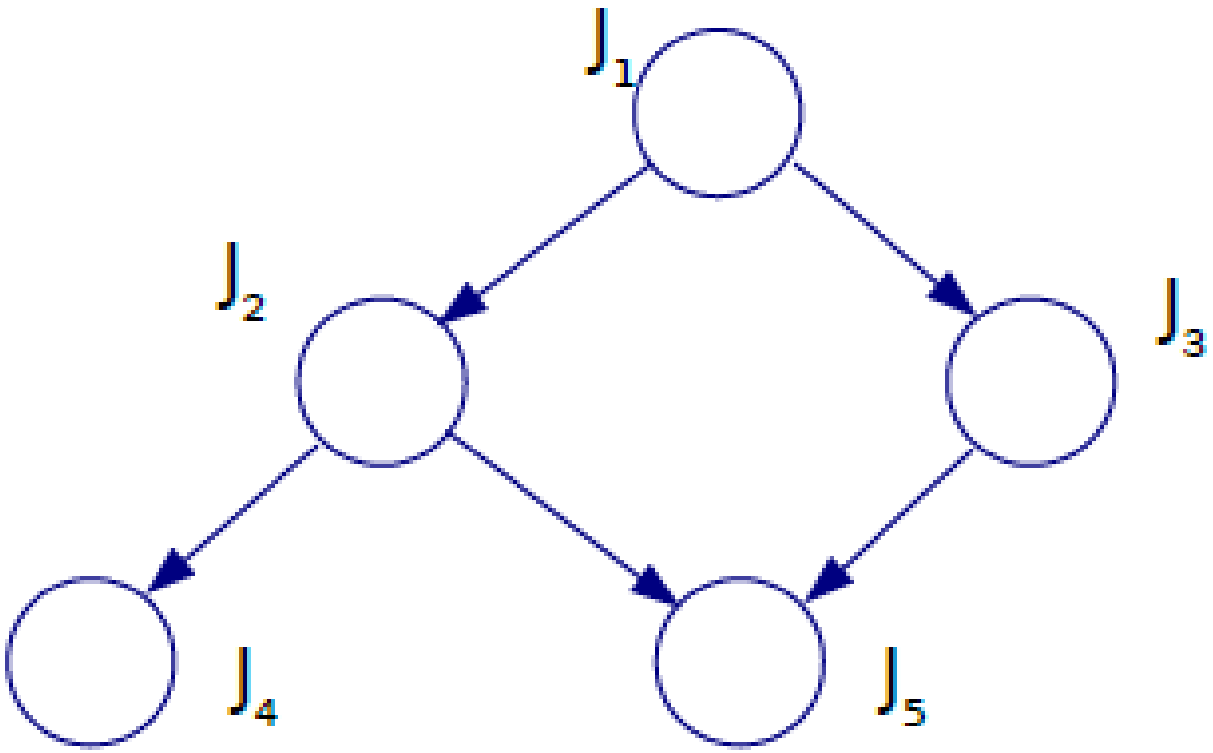
Exemple

$J_2 < J_4$

$J_2 \rightarrow J_4$

$J_1 < J_4$

$J_1 \not\rightarrow J_4$



Ordonnancement de tâches dépendantes avec des contraintes de précédences

Algorithme : Contraintes de précédence et EDF

Principe:

1. modification de la date de réveil et de des l'échéances des tâches :
2. de façon à ce qu'une tâche ait toujours une r_i supérieure à celle de ses prédécesseurs
3. de façon à ce qu'une tâche ait toujours un d_i supérieur à celui de ses prédécesseurs

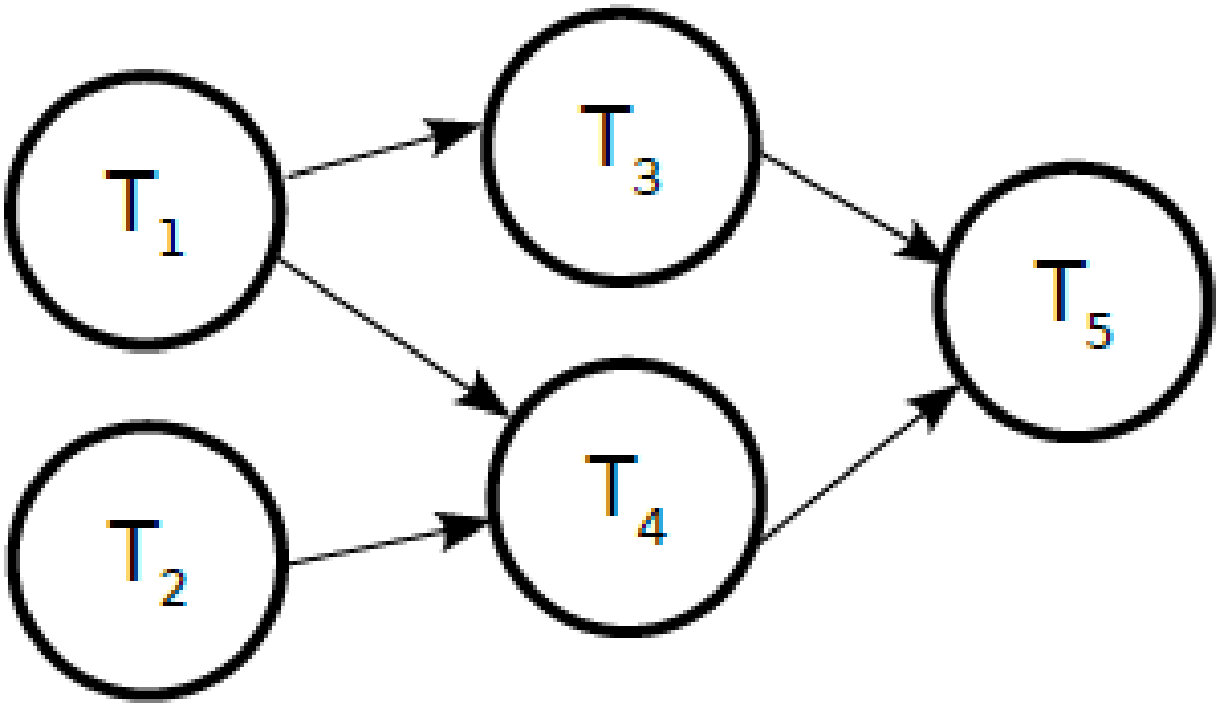
(algorithme de Chetto et al.)

(une tâche ne doit être activable que si tous ses prédécesseurs ont terminé leur exécution)

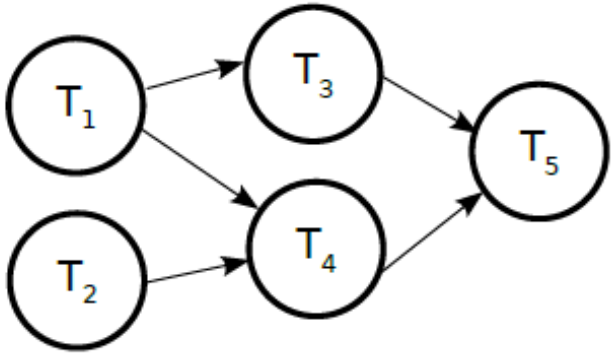
$$r^*_i = \text{Max}\{r_i, \text{Max}\{r^*_j + C_j\}\} \text{ pour tous les j tels que } T_j \rightarrow T_i$$

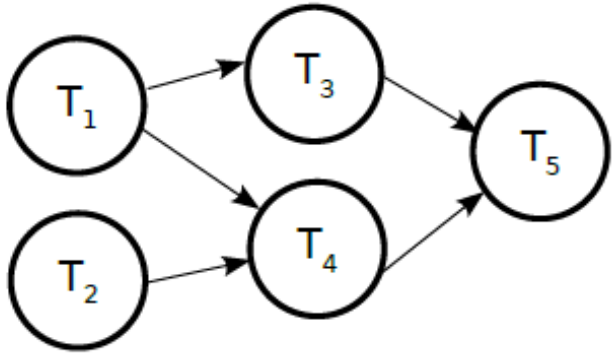
$$d^*_i = \text{Min}\{d_i, \text{Min}\{d^*_j - Cj\}\} \text{ pour tous les j tels que } T_i \rightarrow T_j$$

Exemple:



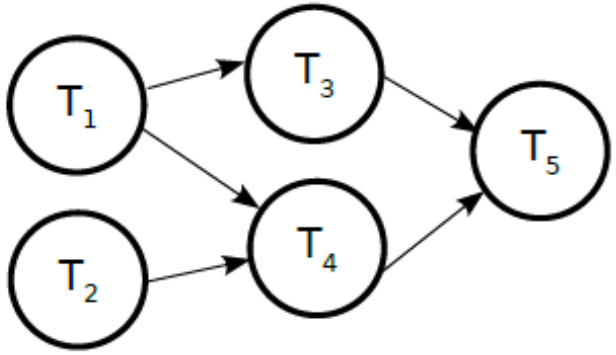
	Paramètres des tâches		
Tâche	r_i	C_i	d_i
T_1	0	1	5
T_2	5	2	7
T_3	0	2	5
T_4	0	1	10
T_5	0	3	12





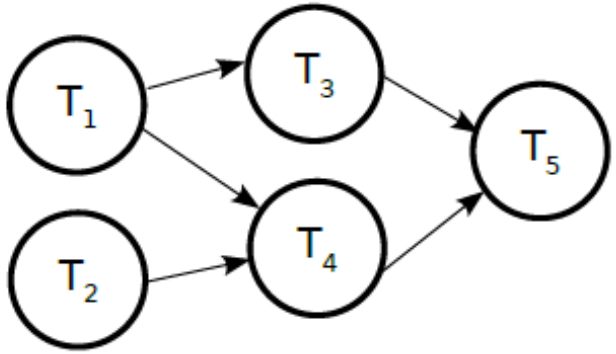
	Paramètres des tâches			Nouvelles valeurs	
Tâche	r_i	C_i	d_i	r^*_i	d^*_i
T_1	0	1	5		
T_2	5	2	7		
T_3	0	2	5		
T_4	0	1	10		
T_5	0	3	12		

T1 et T2 n'ont pas de prédécesseurs alors r1* = 0 et r2* =5



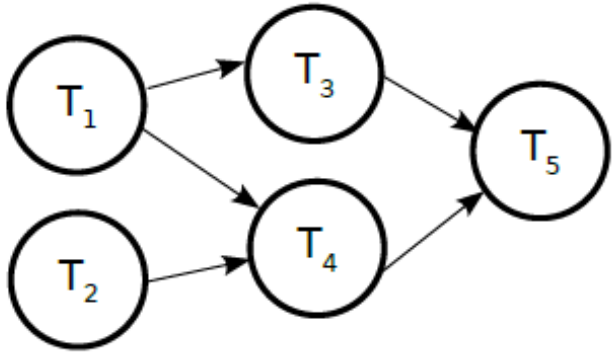
	Paramètres des tâches			Nouvelles valeurs	
Tâche	r_i	C_i	d_i	r^*_i	d^*_i
T_1	0	1	5	0	
T_2	5	2	7	5	
T_3	0	2	5		
T_4	0	1	10		
T_5	0	3	12		

T3 a comme prédécesseur T1 alors r3* =Max (r3 = 0, r1*+c1=0+1) = 1



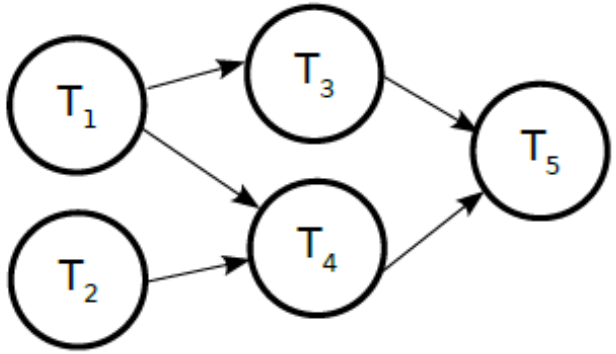
	Paramètres des tâches			Nouvelles valeurs	
Tâche	r_i	C_i	d_i	r^*_i	d^*_i
T_1	0	1	5	0	
T_2	5	2	7	5	
T_3	0	2	5	1	
T_4	0	1	10		
T_5	0	3	12		

T4 a comme prédécesseur T1 et T2 alors r4* =
Max (r4 = 0, r1*+c1=0+1, r2*+c2=5+2) = 7



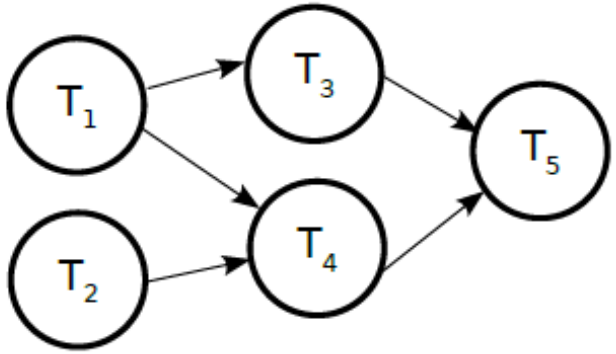
	Paramètres des tâches			Nouvelles valeurs	
Tâche	r_i	C_i	d_i	r^*_i	d^*_i
T_1	0	1	5	0	
T_2	5	2	7	5	
T_3	0	2	5	1	
T_4	0	1	10	7	
T_5	0	3	12		

T5 a comme prédécesseur T3 et T4 alors r5* =
Max (r5 = 0, r3*+c3=1+2, r4*+c4=7+1) = 8



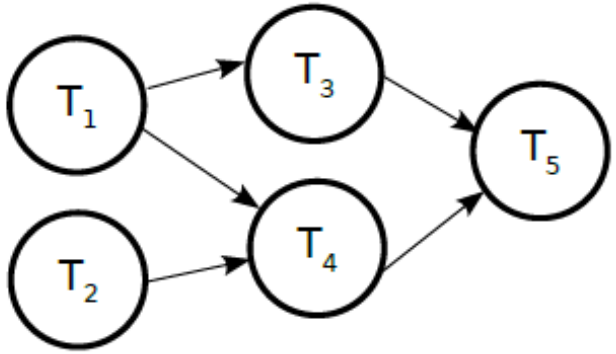
	Paramètres des tâches			Nouvelles valeurs	
Tâche	r_i	C_i	d_i	r^*_i	d^*_i
T_1	0	1	5	0	
T_2	5	2	7	5	
T_3	0	2	5	1	
T_4	0	1	10	7	
T_5	0	3	12	8	

T5 n’a pas de successeurs alors d5* = 12



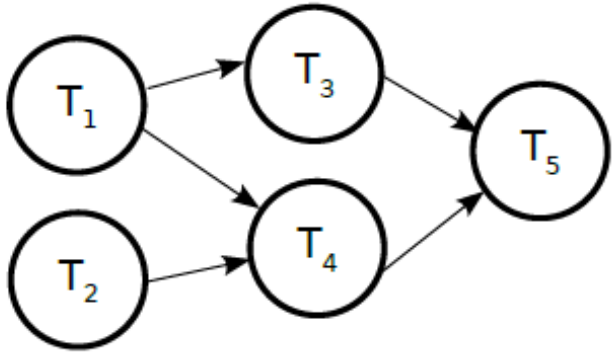
	Paramètres des tâches			Nouvelles valeurs	
Tâche	r_i	C_i	d_i	r^*_i	d^*_i
T_1	0	1	5	0	
T_2	5	2	7	5	
T_3	0	2	5	1	
T_4	0	1	10	7	
T_5	0	3	12	8	12

T4 a comme successeur T5 alors d4* =Min (d4 = 10, d5*- c5= Min(10, 12-3) = 9



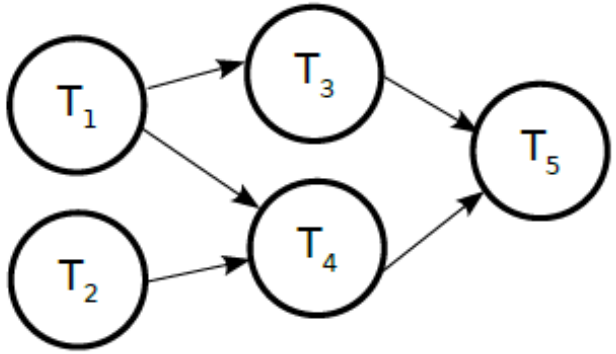
	Paramètres des tâches			Nouvelles valeurs	
Tâche	r_i	C_i	d_i	r^*_i	d^*_i
T_1	0	1	5	0	
T_2	5	2	7	5	
T_3	0	2	5	1	
T_4	0	1	10	7	9
T_5	0	3	12	8	12

T3 a comme successeur T5 alors d3* =Min (d3 = 5, d5*-c5= Min(5, 12-3) = 5



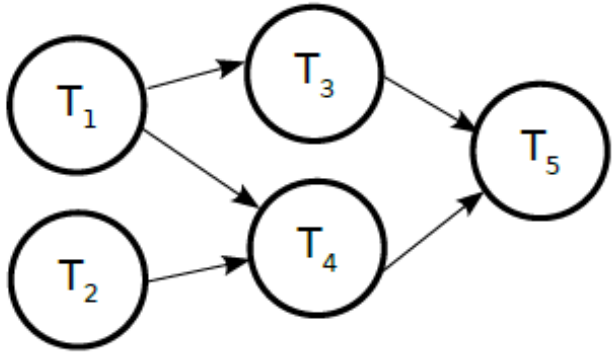
	Paramètres des tâches			Nouvelles valeurs	
Tâche	r_i	C_i	d_i	r^*_i	d^*_i
T_1	0	1	5	0	
T_2	5	2	7	5	
T_3	0	2	5	1	5
T_4	0	1	10	7	9
T_5	0	3	12	8	12

T2 a comme successeur T4 alors d2* =Min (d2 = 7, d4*- c4= Min(7, 9-1) = 7

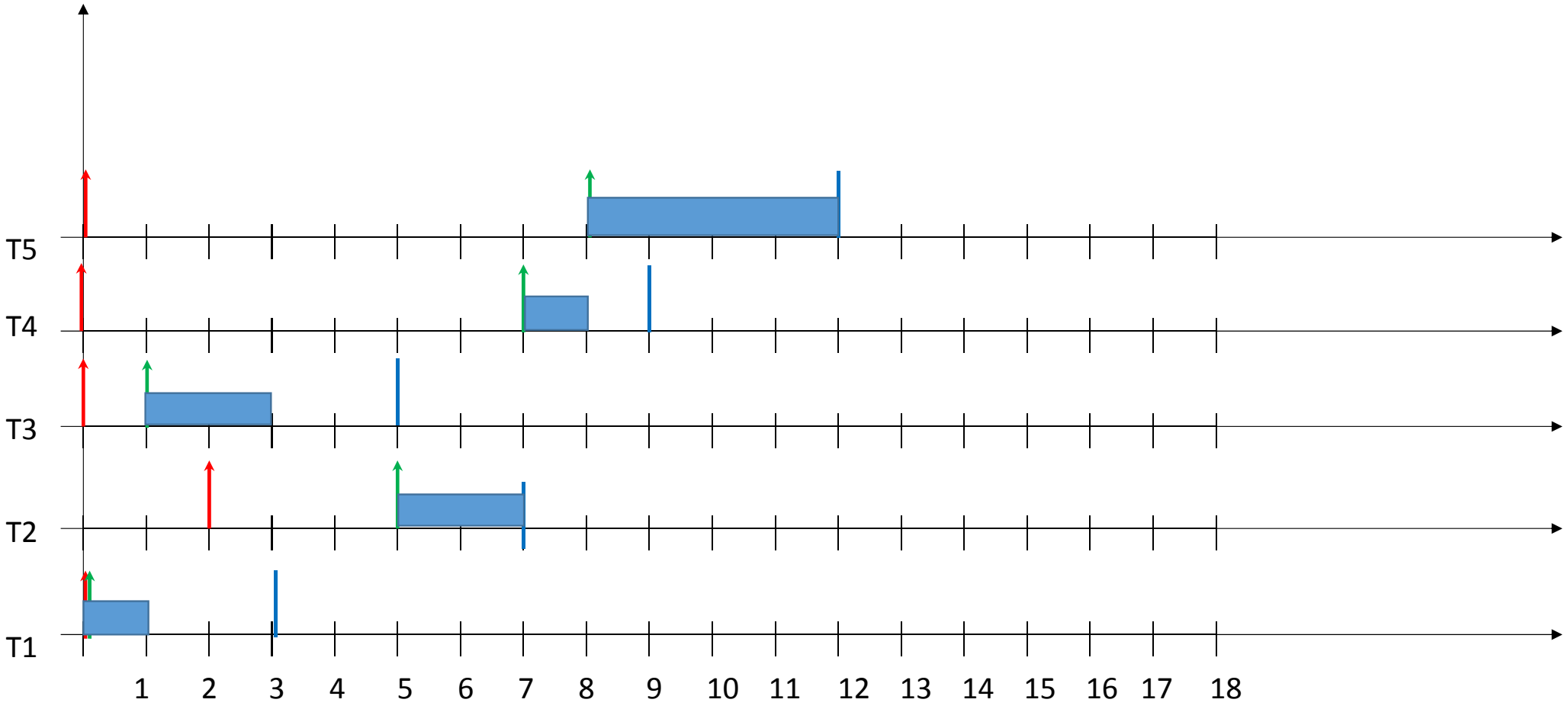


	Paramètres des tâches			Nouvelles valeurs	
Tâche	r_i	C_i	d_i	r^*_i	d^*_i
T_1	0	1	5	0	
T_2	5	2	7	5	7
T_3	0	2	5	1	5
T_4	0	1	10	7	9
T_5	0	3	12	8	12

T1 a comme successeurs T3 et T4 alors d1* =Min (d1 = 5, d3*- c3, d4* - c4) = Min(5, 5-2, 9-1) = 3



	Paramètres des tâches			Nouvelles valeurs	
Tâche	r_i	C_i	d_i	r^*_i	d^*_i
T_1	0	1	5	0	3
T_2	5	2	7	5	7
T_3	0	2	5	1	5
T_4	0	1	10	7	9
T_5	0	3	12	8	12

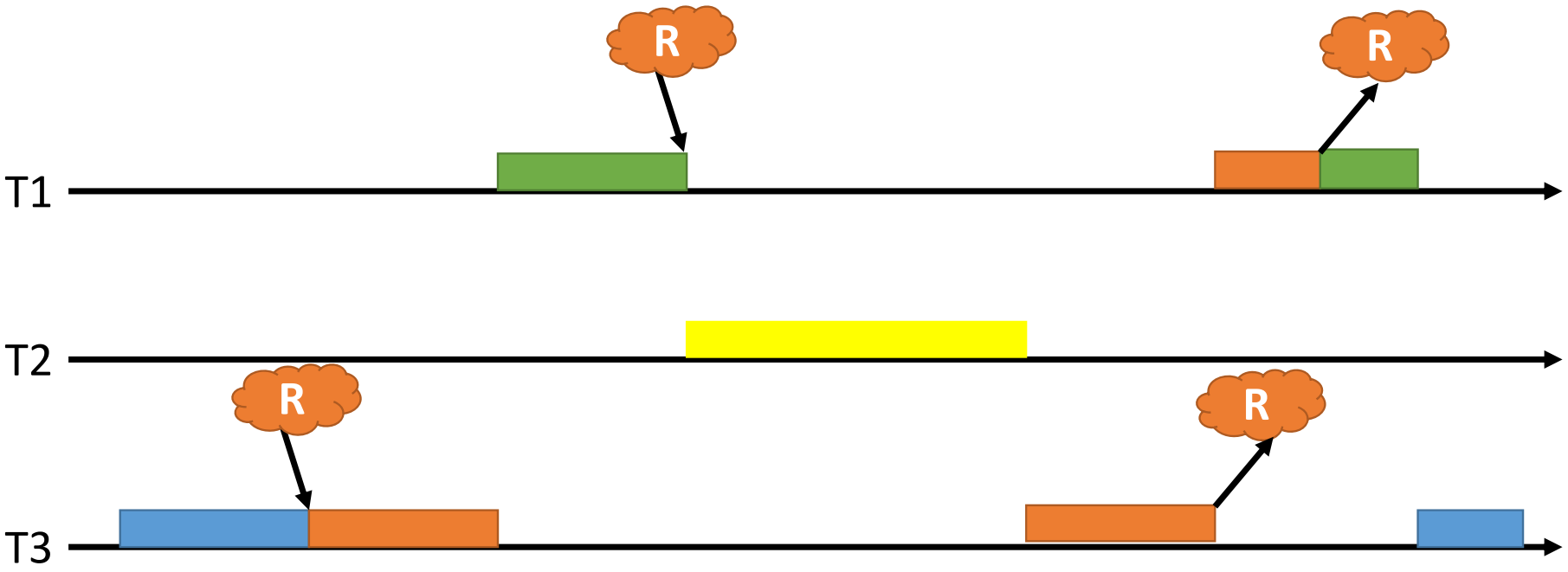

T5
T4
T3
T2
T1
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18

Contraintes par partage de ressources

- Rappel:
- une ressource critique : pas de possibilité d'une utilisation simultanée par plusieurs tâches à la fois,
- une section critique : une séquence d'instructions pendant lesquelles une ressource critique est utilisée,
- Problématique:
- En plus de l'ordonnancement ordinaire (partage de processeur entre tâches) on doit assurer que la cohérence de l'utilisation de la ressource critique qui mène parfois à une impossibilité d'ordonnancement (donc d'exécution de tâches)

Exemple illustratif de la problématique

Soit quatre (04) tâches dont:
Priorité(T1) > Priorité(T2) > Priorité(T3)



Ordonnancement de tâches partageant des ressources critiques:

Protocole d'héritage de priorité

Protocole d'héritage de priorité

- Hypothèses:
- Soit n tâches périodiques T1, T2, ... , Tn (période Pi, capacité Ci), qui partagent m ressources R1, R2, ... , Rm
- Chaque ressource Rj est gardée par un sémaphore binaire Sj

limitant une section critique $SC_{i,j}$ (pour Ti)

- On distingue: Pi : priorité nominale et pi : priorité active (pour Ti)

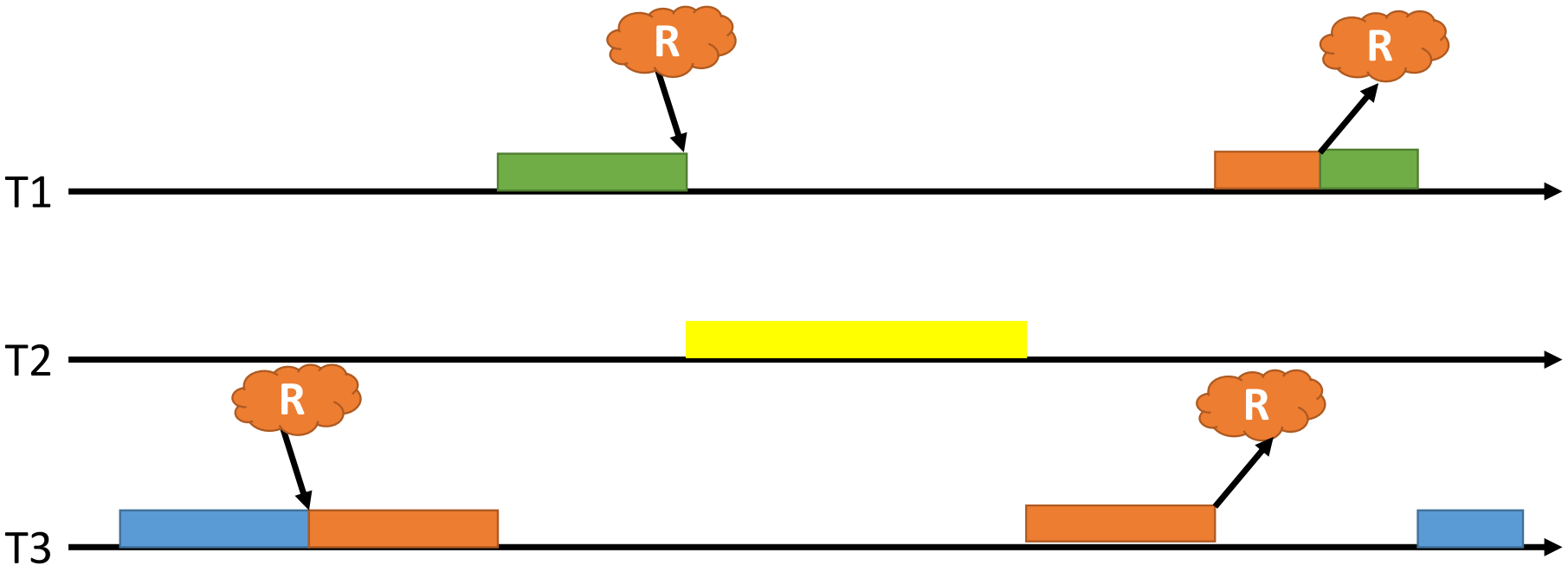
- Principe :
- donner à la tâche qui possède la ressource Rj la priorité de la tâche de plus haute priorité qui attend cette Ressource Rj
- Détails :
- les tâches sont ordonnancées suivant leur priorité active
- lorsqu'une tâche Ti cherche à entrer dans une section critique

$SC_{i,j}$ et que la ressource Rj est déjà possédée par une tâche Tj,

de priorité plus faible, alors Ti se bloque (sinon Ti entre dans $SC_{i,j}$)

- la tâche Ti, bloquée, transmet sa priorité à Tj (la priorité active pj de Tj = la priorité nominale Pi de Ti)
- Ceci aide Tj à reprendre et à terminer son exécution de la section critique $SC_{i,j}$
- PS. Tj a déjà lancée un sémaphore Sj en entrant à $SC_{i,j}$
- Lorsque Tj sort de la $SC_{i,j}$, elle relâche le sémaphore Sj et donc la tâche de plus haute priorité (Ti) est réveillée
- La priorité active pj de Tj est ramenée à la priorité nominale Pj (sa priorité initiale)

Exemple (sans héritage de priorité)



Exemple (avec héritage de priorité)