Algorithmique des graphes

David Pichardie

27 Avril 2018

Bilan du CM9

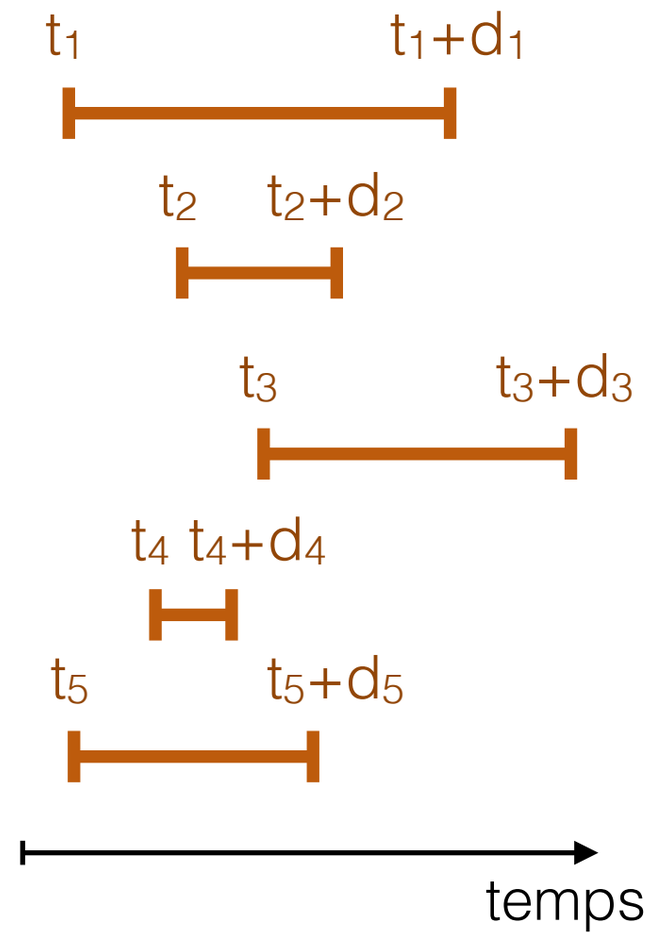
- Comparaison de Dijkstra et Prim
- Implémentation des files de priorité
- Algorithme de Bellman-Ford

Problèmes d’ordonnancement

Nature des problèmes

On considère un projet décomposé en *n* tâches $A_1, A_2, \ldots, A_n$ telles que

- chaque tâche est indivisible
- chaque tâche A_i a une date de début *à déterminer :* t_i
- chaque tâche A_i a une durée *connue* : d_i
- les dates $t_1, t_2, \ldots, t_n$ sont soumises à des *contraintes temporelles*



Cas d’étude

Cas d'étude

On considère un projet décomposé en 6 tâches

tâche	A_1	A_2	A_3	A_4	A_5	A_6
durée en unités de temps (ut)	10	12	6	20	12	7

Cas d'étude

Ces tâches sont soumises à des contraintes temporelles

- A_1 peut commencer dès le début du projet
- A_2 commence **au plus tôt** 7 ut après le début de A_1
- A_3 enchaîne sans délais avec A_2
- A_4 commence **au plus tôt** 3 ut après l'achèvement de A_5, et la seconde moitié de A_4 commence **au plus tôt** 13 ut après l'achèvement de A_3
- A_5 commence **au plus tôt** lorsque les 3/4 de A_2 sont achevés, **au plus tard** 6 ut après l'achèvement de A_2, et **au plus tôt** 3 ut après l'achèvement de A_6
- A_6 commence **au plus tôt** 20 ut après le début des travaux

Problèmes à résoudre

Pour ce type de projet, on souhaite pouvoir répondre aux questions suivantes

- Est-il possible de finir le projet en, par exemple ici, 50 ut ?
- Quelle est la durée minimale du projet ?
- Quelles tâches peuvent prendre du retard sans compromettre la durée du projet ?

Modélisation

On traduits les contraintes par des égalités/ inégalités sur les variables $t_{1,...,}t_6$

- A_1 peut commencer dès le début du projet
- A_2 commence **au plus tôt** 7 ut après le début de A_1
- A_3 enchaîne sans délais avec A_2
- A_4 commence **au plus tôt** 3 ut après l'achèvement de A_5,
  et la seconde moitié de A_4 commence **au plus tôt** 13 ut après l'achèvement de A_3
- A_5 commence **au plus tôt** lorsque les 3/4 de A_2 sont achevés,
  au plus tard 6 ut après l'achèvement de A_2,
  et **au plus tôt** 3 ut après l'achèvement de A_6
- A_6 commence **au plus tôt** 20 ut après le début des travaux

$$t_1 \geq 0$$

$$t_2 \geq t_1 + 7$$

$$t_3 = t_2 + 12$$

$$t_4 \geq t_5 + 12 + 3$$

$$t_4 + 10 \geq t_3 + 13 + 6$$

$$t_5 \geq t_2 + \frac{3}{4} \cdot 12$$

$$t_5 \leq t_2 + 12 + 6$$

$$t_5 \geq t_6 + 7 + 3$$

$$t_6 \geq 20$$

Conventions (cas général)

- on ajoute une date $t_0=0$ de démarrage du projet (associée à une tâche fictive A_0)
- on ajoute une date t_{n+1} de fin de projet (associée à une tâche fictive A_{n+1})

  $t_{n+1} \geq t_i + d_i$, pour i=1,…,n (contraintes implicites)

- autre contraintes implicites :

  $t_i \geq 0$, pour i=1,…,n

- on peut ainsi exprimer toutes les contraintes comme une conjonction d'inégalités de la forme $t_j - t_i \geq$ constante

Cas d'étude

A_1	$t_1 - t_0 \geq 0$
A_2	$t_2 - t_1 \geq 7$
A_3	$t_3 - t_2 \geq 12 \quad , \quad t_2 - t_3 \geq -12$
A_4	$t_4 - t_5 \geq 15 \quad , \quad t_4 - t_3 \geq 9$
A_5	$t_5 - t_2 \geq 9 \quad , \quad t_2 - t_5 \geq -18 \quad , \quad t_5 - t_6 \geq 10$
A_6	$t_6 - t_0 \geq 20$

+ les contraintes implicites

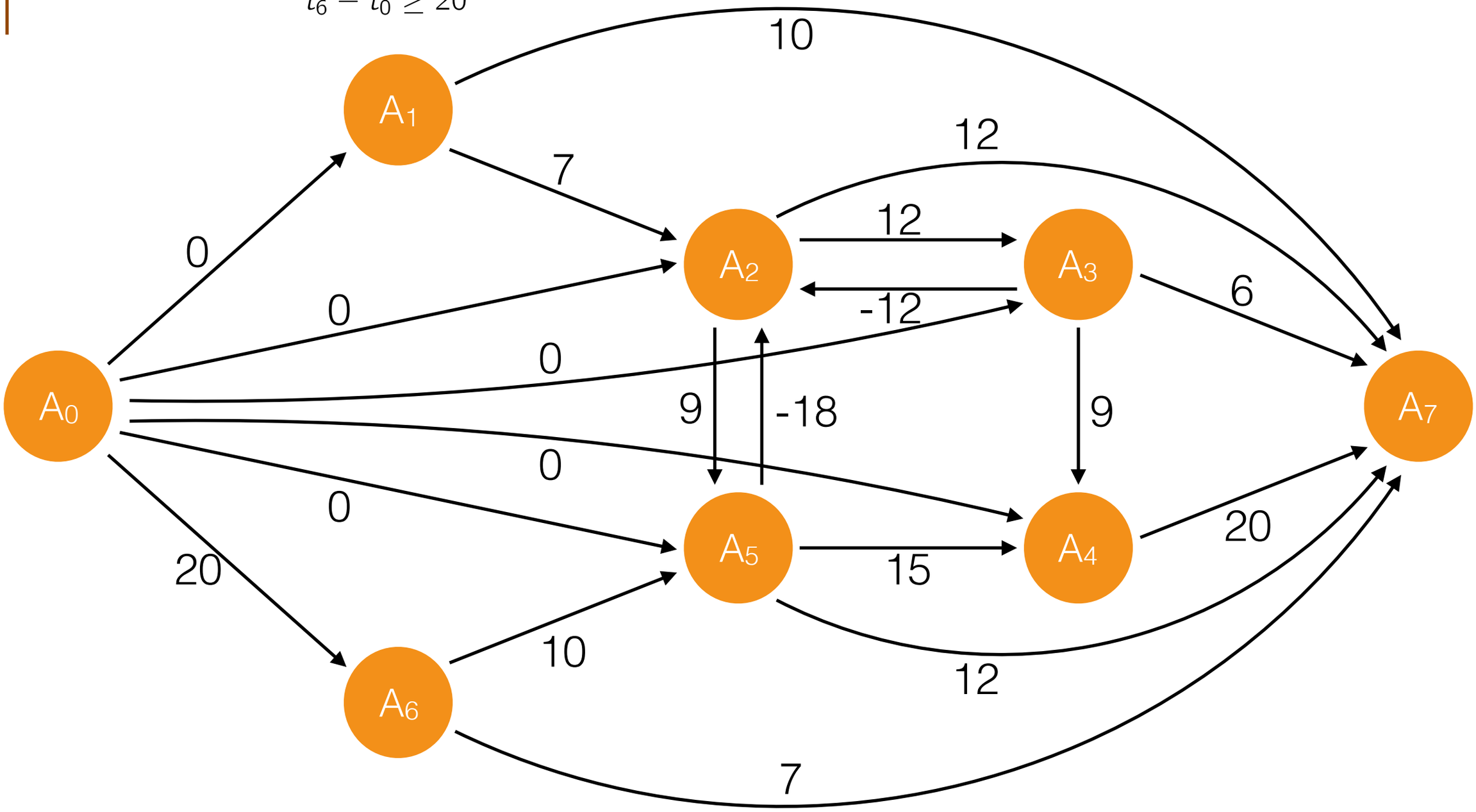
Représentation par un graphe potentiel-tâche

On synthétise cet ensemble de contraintes sur un graphe orienté pondéré appelée *potentiel-tâche*

- sommets : les actions $A_0, A_1, A_2, \ldots, A_n, A_{n+1}$
- arcs : un arc $A_i \xrightarrow{a_{ij}} A_j$ pour chaque contrainte $t_j - t_i \geq a_{ij}$

A_1	$t_1 - t_0 \geq 0$
A_2	$t_2 - t_1 \geq 7$
A_3	$t_3 - t_2 \geq 12 \quad , \quad t_2 - t_3 \geq -12$
A_4	$t_4 - t_5 \geq 15 \quad , \quad t_4 - t_3 \geq 9$
A_5	$t_5 - t_2 \geq 9 \quad , \quad t_2 - t_5 \geq -18 \quad , \quad t_5 - t_6 \geq 10$
A_6	$t_6 - t_0 \geq 20$

tâche	A_1	A_2	A_3	A_4	A_5	A_6
durée	10	12	6	20	12	7



Contraintes redondantes

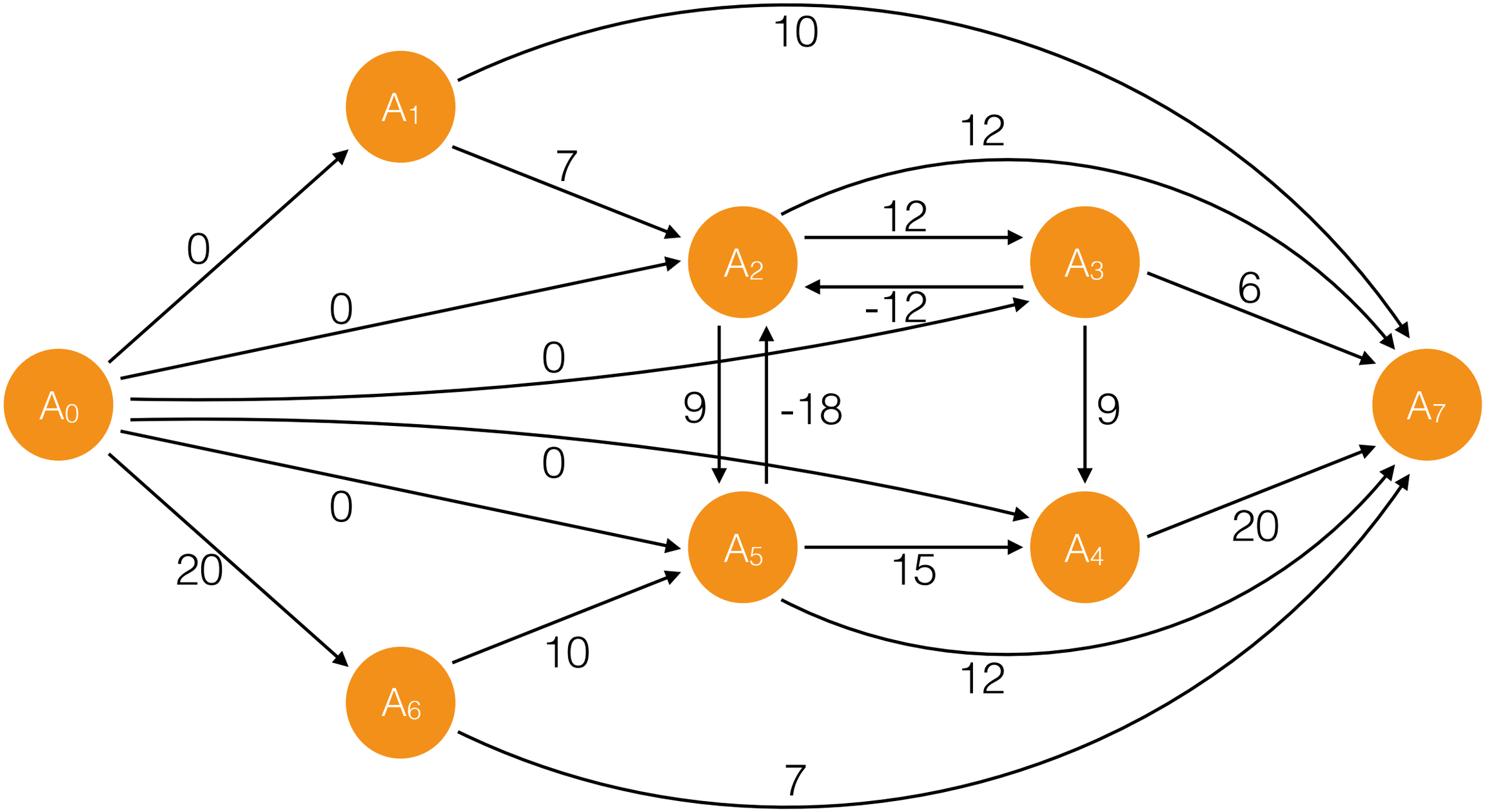
Une contrainte est *redondante* si elle est conséquence d'autres contraintes du système.

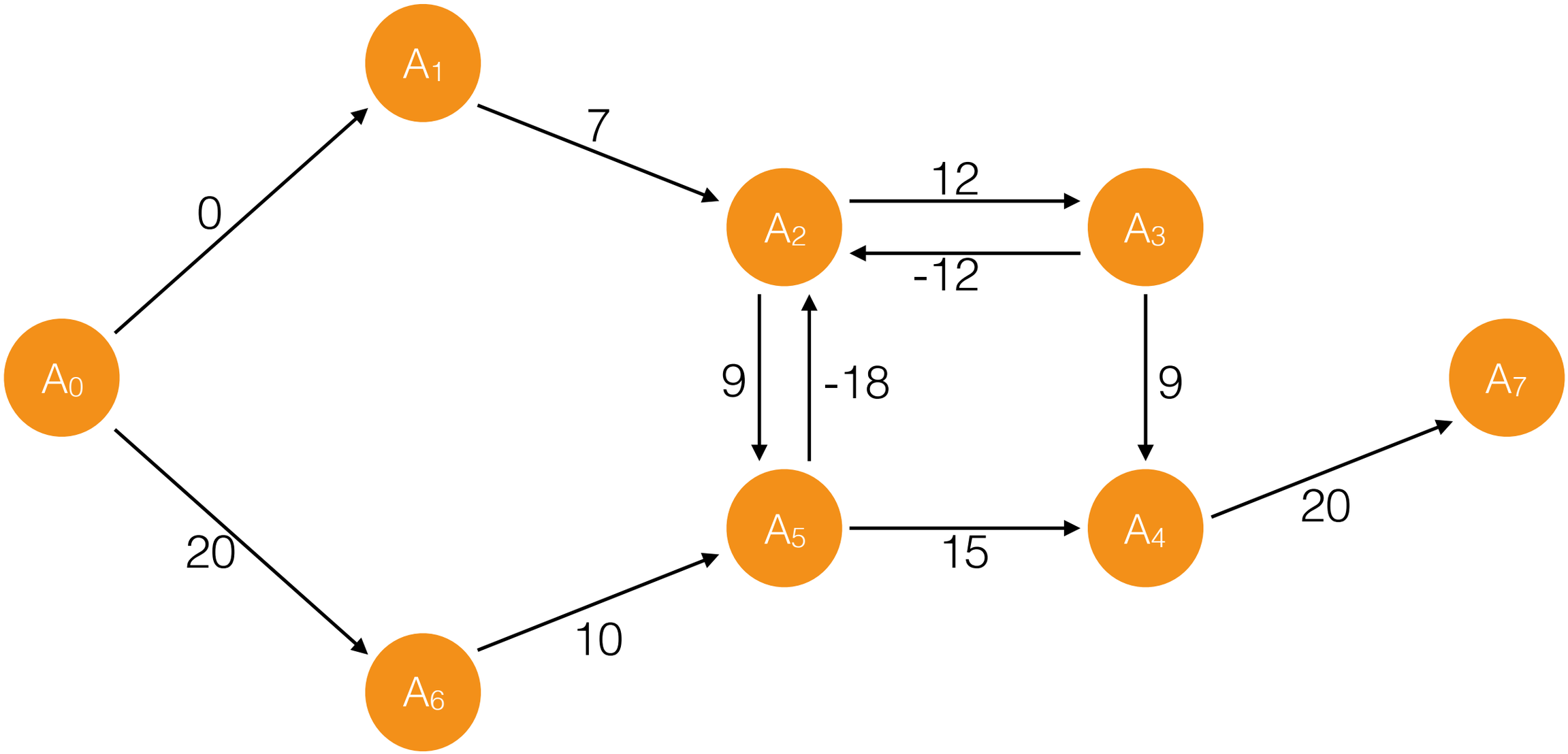
Exemple 1 : $t_6 \geq 20 \Longrightarrow t_6 \geq 0$

Exemple 2 : $\left\{ \begin{array}{l} t_5 - t_6 \geq 10 \\ t_6 \geq 20 \end{array} \right. \Longrightarrow t_5 \geq 0$

Exemple 3 : $\left\{ \begin{array}{l} t_7 - t_4 \geq 20 \\ t_4 - t_3 \geq 9 \end{array} \right. \Longrightarrow t_7 - t_3 \geq 6$

Il est inutile de représenter ces contraintes sur le graphe potentiel-tâche


A0
A1
A2
A3
A4
A5
A6
A7
10
12
7
12
0
0
-12
0
9
-18
9
0
0
20
20
15
10
12
7


A1
A0
A2
A3
A5
A4
A6
A7
0
7
12
-12
9
-18
9
20
15
20
10

Ordonnancement au plut tôt

Un ordonnancement (t_0, t_1, …, t_n, t_{n+1}) est dit *compatible* s'il satisfait les contraintes du problème.

Un *ordonnancement au plus tôt* est un ordonnancement pour lequel, parmi tous les ordonnancement, t_{n+1} est minimal.

Théorèmes

Il existe un ordonnancement **au plus tôt** si et seulement si il existe un plus grand chemin de A_0 à A_{n+1}.

Cet ordonnancement est donné par :

$$t_i = \Delta(0, i)$$

$\Delta(j, i)$: distance maximale du sommet A_j au sommet A_i.

Pour tout autre ordonnancement (t_0, t_1, …, t_n, t_{n+1}), on a :

$$\Delta(0, i) \leq t_i$$

un plus grand chemin minimise aussi les temps intermédiaires

Calcul de plus grands chemins

Algorithme	Hypothèse pour pouvoir calculer des plus grands chemins	Modification à apporter par rapport au calcul de plus courts chemins
Ordinal	pas de cycle	remplacer MIN par MAX
Dijkstra	pas de poids **positifs**	remplacer MIN par MAX
Bellman-Ford	pas de cycles de poids **positifs**	remplacer MIN par MAX

critère *optimal*

Propriété

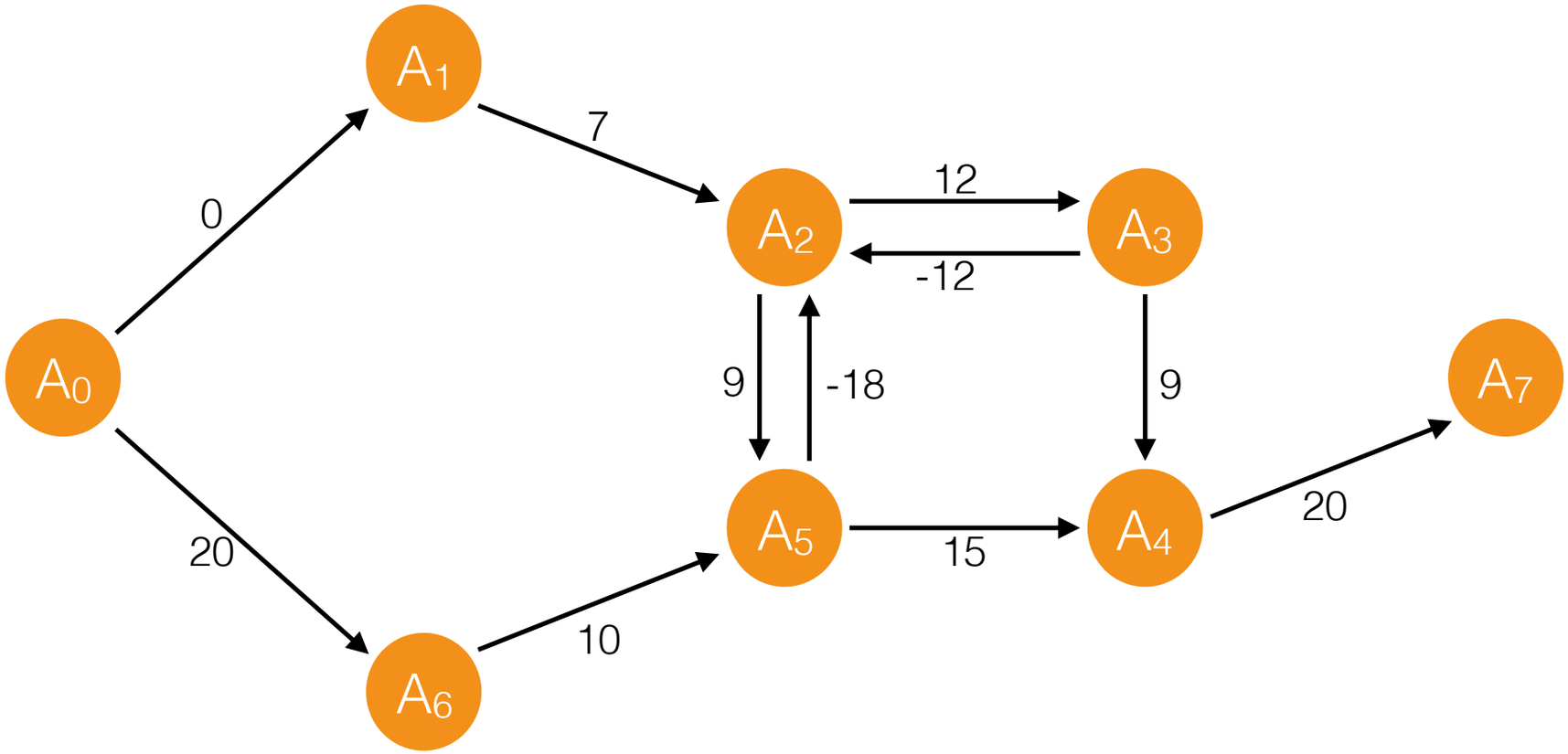
Dans un graphe potentiel-tâche, les cycles sont de poids négatifs ou nuls si et seulement si il existe au moins un ordonnancement compatible

Preuve $\Rightarrow$: si les cycles sont de poids négatifs ou nuls, l'algorithme de Bellman-Ford va permettre de calculer les plus grandes distance et fournir un ordonnancement.

Preuve $\Leftarrow$: soit $A_{i_1}, \ldots, A_{i_p}, A_{i_1}$ un cycle de poids

$$a_{i_1 i_2} + \cdots + a_{i_p i_1} : \quad \left.\begin{array}{l} t_{i_2} - t_{i_1} \geq a_{i_1 i_2} \\ \cdots \\ t_{i_1} - t_{i_p} \geq a_{i_p i_1} \end{array}\right\} \Rightarrow 0 \geq a_{i_1 i_2} + \cdots + a_{i_p i_1}$$

Utilisation de Bellman-Ford



	A_0	A_1	A_2	A_3	A_4	A_5	A_6	A_7
k=0	0	-oo	-oo	-oo	-oo	-oo	-oo	-oo
k=1	0	0	7	19	45	30	20	65
k=2	0	0	12	24	45	30	20	65

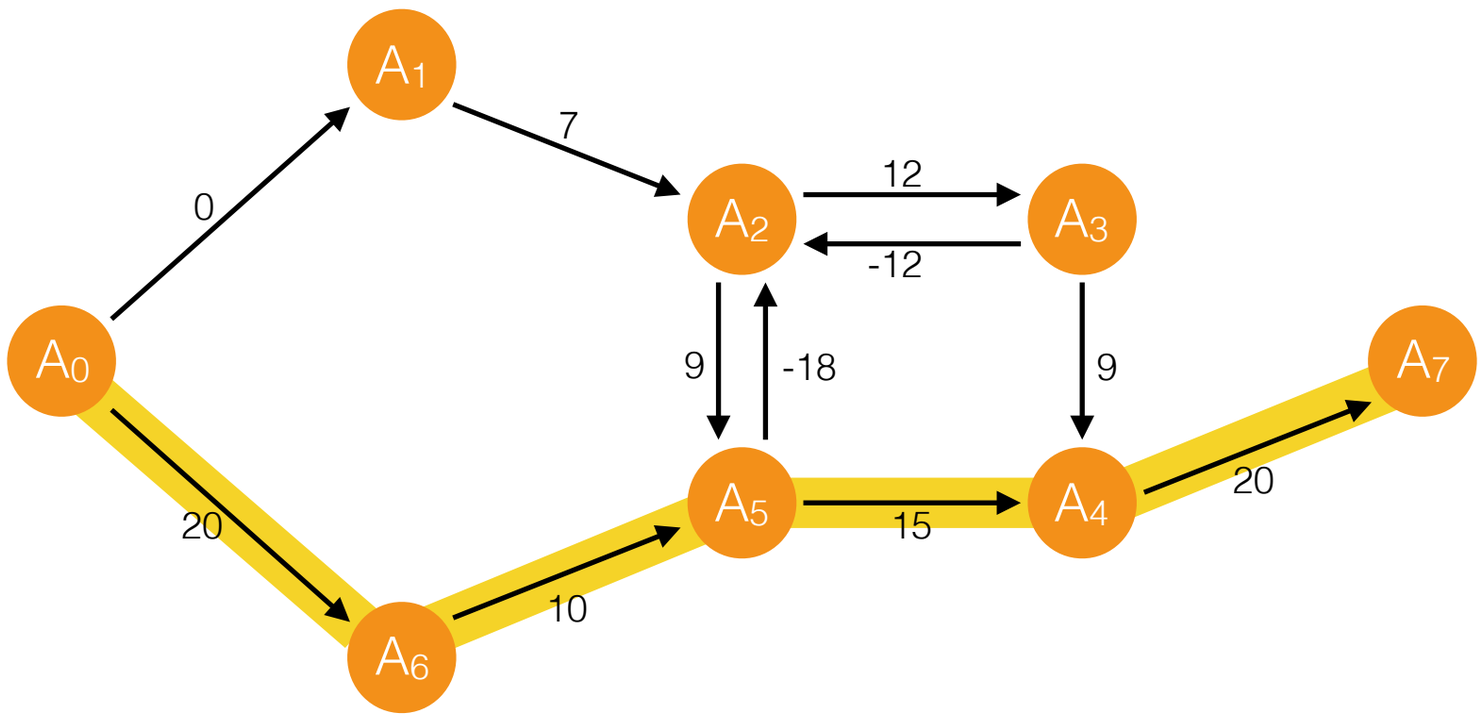
(stable)

durée minimal du projet

Chemin critique

On appelle *chemin critique* d'un graphe potentiel-tâche un chemin tel que si une tâche de ce chemin est retardée, alors la durée totale du projet est retardée aussi.

Théorème : pour l'ordonnancement au plus tôt, les chemins critiques sont les plus grands chemins entre A_0 et A_{n+1}.



Ordonnancement au plus tard

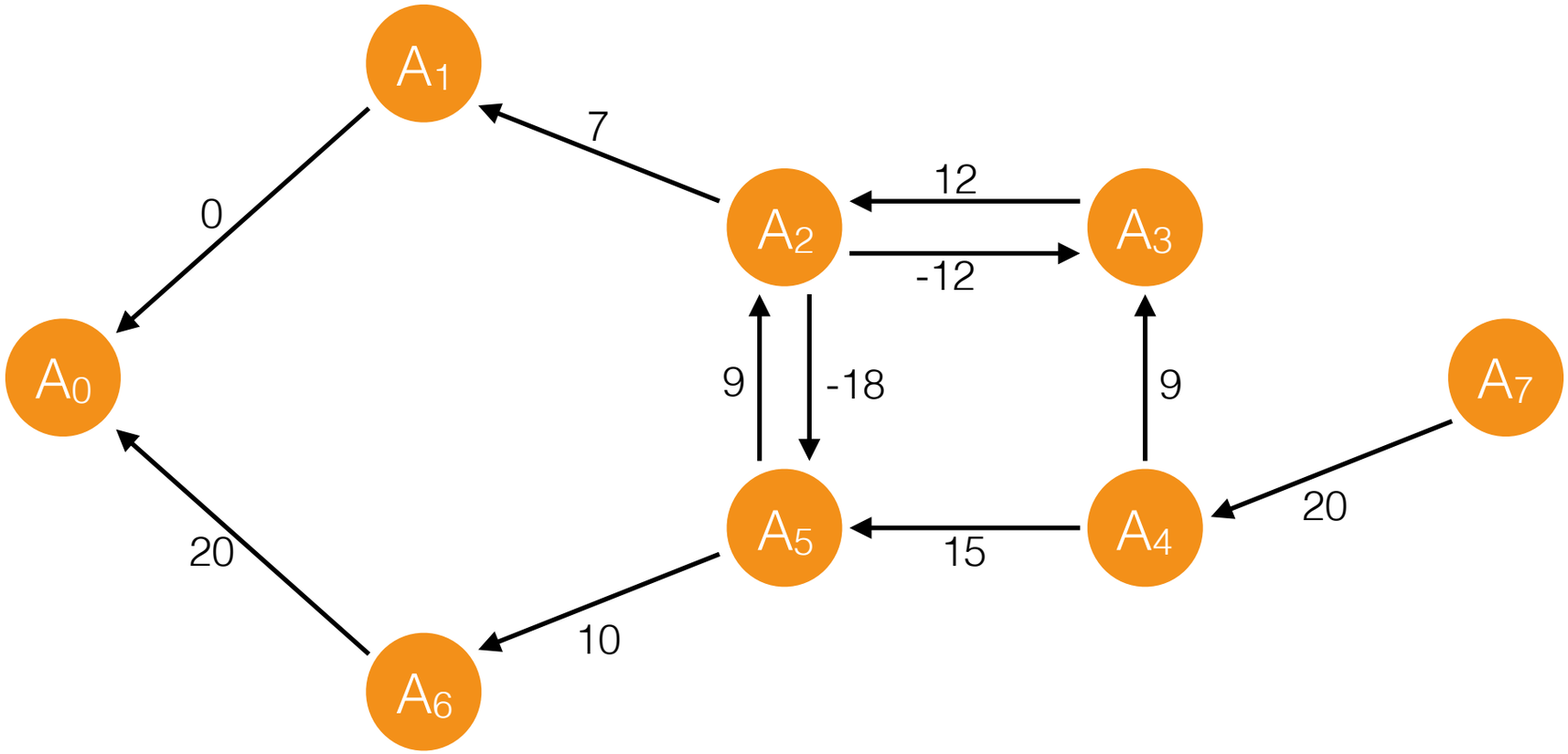
$\Delta^{-1}(0, n+1)$

Pour une date de fin projet F donnée, un *ordonnancement au plus tard* est un ordonnancement pour lequel, parmi tous les ordonnancement, t_{n+1} =F et t_0 est maximal.

Algorithme : on inverse le graphe et on calcule les plus grandes distances $\Delta^{-1}(n+1, i)$ à partir de A_{n+1}. Le résultats attendu est alors :

$$t_i = F - \Delta^{-1}(n+1, i)$$

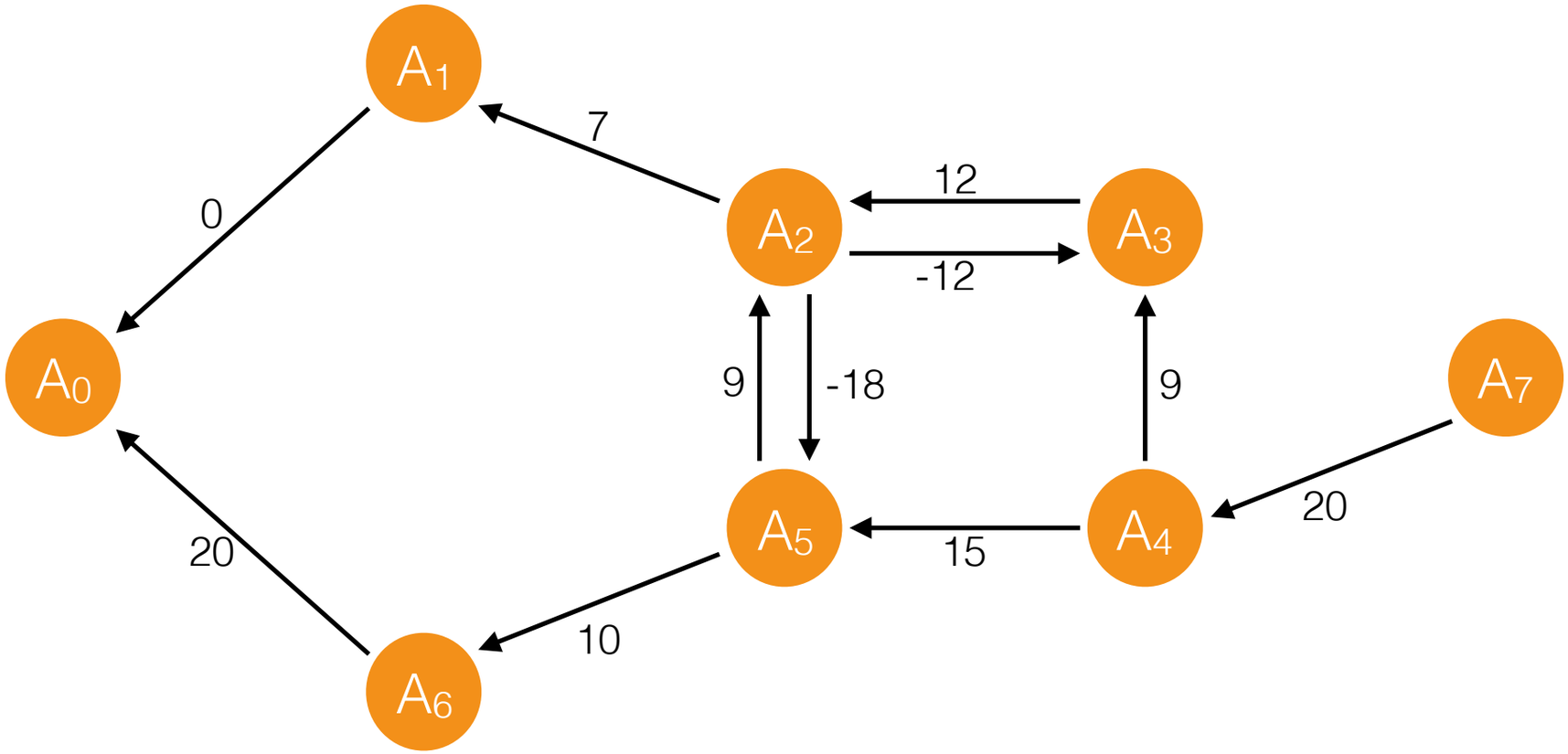
Ordonnancement au plus tard



	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
k=0	0	-oo	-oo	-oo	-oo	-oo	-oo	-oo
k=1	0	45	35	20	29	41	48	65
k=2	0	45	35	20	32	44	51	65

(stable)

Ordonnancement au plus tard



	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
$\Delta^{-1}(7,i)$	0	45	35	20	32	44	51	65
$t_i = 65 - \Delta^{-1}(7,i)$	65	20	30	45	33	21	14	0

Marges

On fixe une date de fin projet F (par exemple $\Delta(0, n+1)$)

On note λ_i les dates de début de l'ordonnancement **au plus tôt**

On note λ'_i les dates de début de l'ordonnancement **au plus tard**

Marge totale de la tâche A_i : $m_i = \lambda'_i - \lambda_i$

Marge d'un chemin : $m(u) = \max_{i \in u} m_i$ (u un chemin)

Marge libre pour une tâche A_i d'un ordonnancement (t_i) donné :

$$\mu_i = \min_{i \xrightarrow{a_{ij}} j} (t_j - t_i - a_{ij})$$

Marge totale

C’est le retard maximum que peut prend la tâche A_i sans compromettre la durée totale prévue F

Attention : les marges totales ne sont pas indépendantes (ce retard pourrait modifier l’ordonnancement des tâches ultérieures)

Marge par chemin

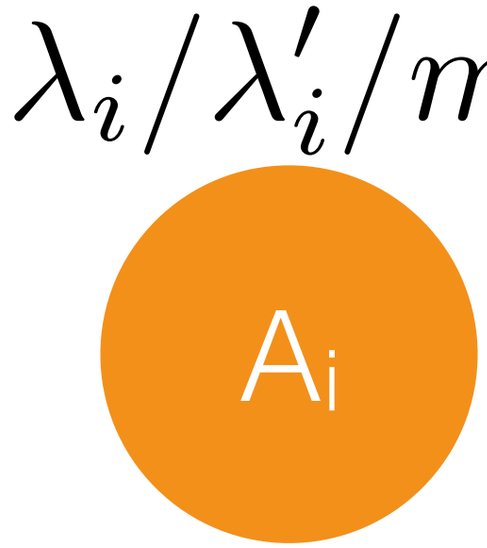
La somme des retards d’un chemin doit rester inférieure à la marge du chemin

Marge libre

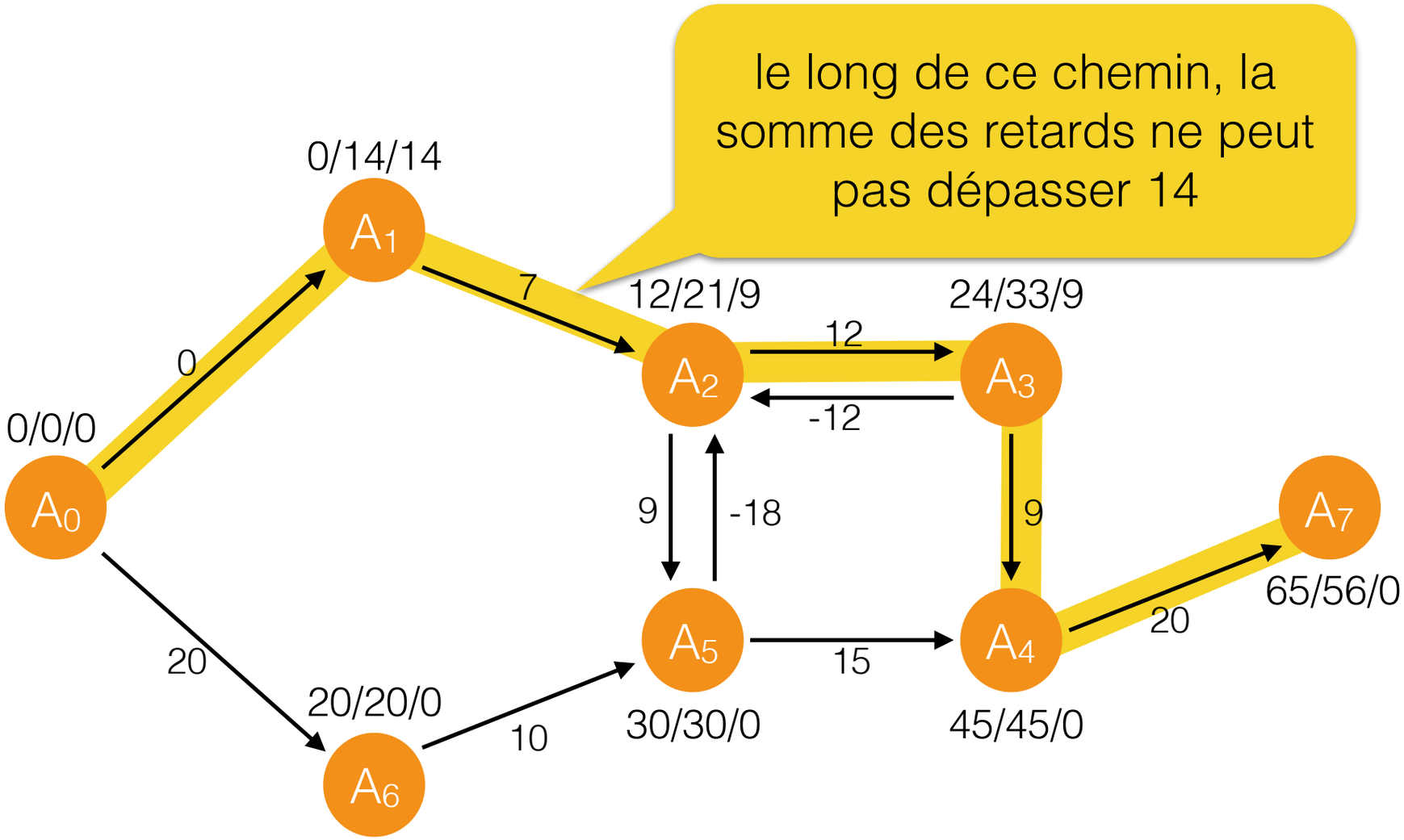
C’est le retard maximum que peut prendre l’achèvement de A_i sans compromettre la date de début prévue pour les autres tâches.

marge libre $\leq$ *marge totale*

Représentation graphique



Cas d'étude


le long de ce chemin, la somme des retards ne peut pas dépasser 14
0/14/14
A1
7
12/21/9
24/33/9
12
-12
A2
A3
0
0/0/0
A0
9
-18
9
A7
65/56/0
20
A5
15
A4
20
20/20/0
10
30/30/0
45/45/0
A6