# Cours de Recherche Opérationnelle 2

**Enseignant : Afef Bouzaiene**
Ecole Nationale des Sciences et Technologies Avancées Borj Cedria – 2ème année Option S.I.C.

2020-2021

# Bibliographie


- Précis de Recherche Opérationnelle, 6ème édition, Robert Faure, Bernard Lemaire, Christophe Picouleau, DUNOD, Paris, 2009.
- Graphes et algorithmes , Michel Gondran et Michel Minoux, 4ème Edition, Lavoisier, 2009.
- Exercices et Problèmes résolus de recherche opérationnelle Tome 1, Roseaux, DUNOD, 2005.
- http://www.lamsade.dauphine.fr/~gabrel/enseignement.php ; Notes de cours « Algorithmique et applications », Virginie Gabrel, CPES 2ème année, 2016-2017.
- http://www.lgi.ecp.fr/~mousseau/Cours/S4/pmwiki/uploads/Main/GraphesAlgoBase.pdf; «Théorie des Graphes, algorithmes de bases », Vincent Mousseau, Ecole Centrale Paris, 2009.

# Problème d'ordonnancement en Gestion de Projets

- Problème de Gestion de projets sans contraintes de ressources :

- Ordonnancer en une durée minimale n tâches soumises à des contraintes de succession => contraintes de potentiels

**ordonnancer :** attribuer une date de début à chaque tâche, (ajout éventuel d'une tâche fictive début (0) et d'une tâche fictive fin (n+1))

**contrainte de succession :** la tâche j ne peut commencer avant la fin de la tâche i ($t_j$-$t_i$ $\geq a_{ij}$) ; où $t_i$ resp. $t_j$ est la date de début de la tâche i resp. j, $t_0$=0 et $a_{ij}$ étant la durée de la tâche i.

➔ l'exemple 5 illustre ce problème : cas d'ordonnancement de projet avec ressources illimitées (encore appelé problème central de l'ordonnancement)

# Problème d'ordonnancement en Gestion de Projets

- Formulation mathématique :

- Variables de décision : $(t_0, t_1, \ldots, t_n, t_{n+1})$
- Min $(t_{n+1}-t_0)$

s.c.

- Les contraintes de potentiels : $t_j-t_i \geq a_{ij}$
- Les contraintes de non-négativité : $t_i \geq 0$ ; $0 \leq i \leq (n+1)$

➔ C'est un P.L.

➔ Existence de méthodes de résolution plus simples pour ce problème en se basant sur une modélisation par les graphes :

La Méthode des Potentiels et la Méthode PERT

# Problème d'ordonnancement en Gestion de Projets

- La Méthode des Potentiels :

- Modélisation par un graphe orienté potentiels-tâches G(X,U) ; où les nœuds sont les tâches ; et tout arc (i,j), valué par $a_{ij}$, représente la contrainte de potentiels ($t_j$-$t_i$ $\geq a_{ij}$) si cette contrainte existe entre i et j.

- Détermination de la durée minimale du projet => correspond à la recherche du chemin le plus long entre les sommets 0 et (n+1) du graphe

Pour la durée minimale trouvée, on distingue 2 cas particuliers d'ordonnancements réalisables :

- l'ordonnancement au plus tôt , et

- l'ordonnancement au plus tard

# Problème d'ordonnancement en Gestion de Projets

- La Méthode des Potentiels :

- **Condition nécessaire et suffisante pour la détermination** du **chemin le plus long** entre les sommets 0 et (n+1) : absence de circuit de valeur strictement positive dans le graphe pouvant être atteint à partir du sommet 0.

# Méthode des Potentiels

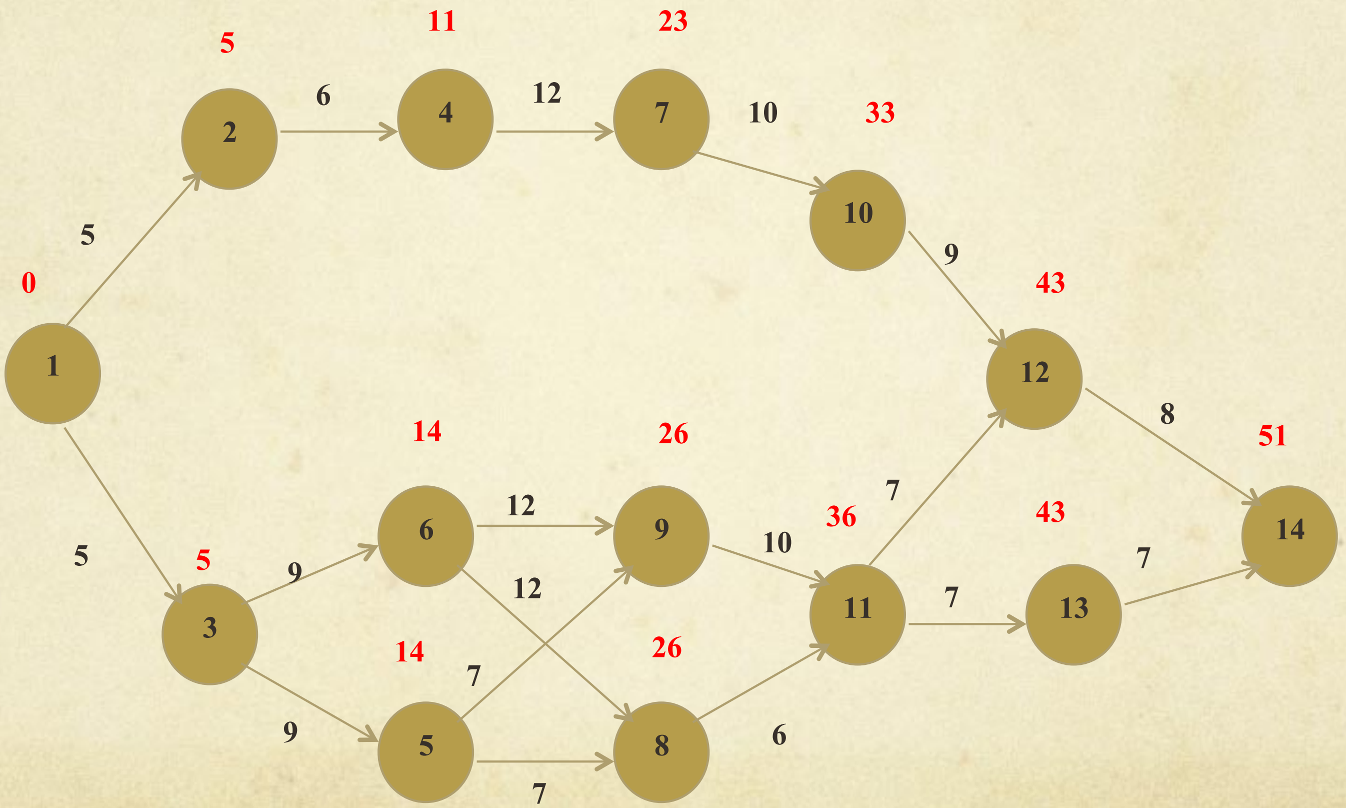
- **Dates de début au plus tôt :** La date de début au plus tôt $\underline{t}_i$ d'une tâche i est égale à la longueur du plus long chemin de la tâche 0 à i

- Dans le cas d'un graphe sans circuit, on a :

$$\underline{t}_0 = 0;$$
$$\underline{t}_i = \max_{j \in \Gamma^{-1}(i)} (\underline{t}_j + v_{j,i}),$$

Où $v_{j,i}$ est la valuation de l'arc (j,i) (ici durée de j)

# Ordonnancement au plus tôt



# Méthode des Potentiels

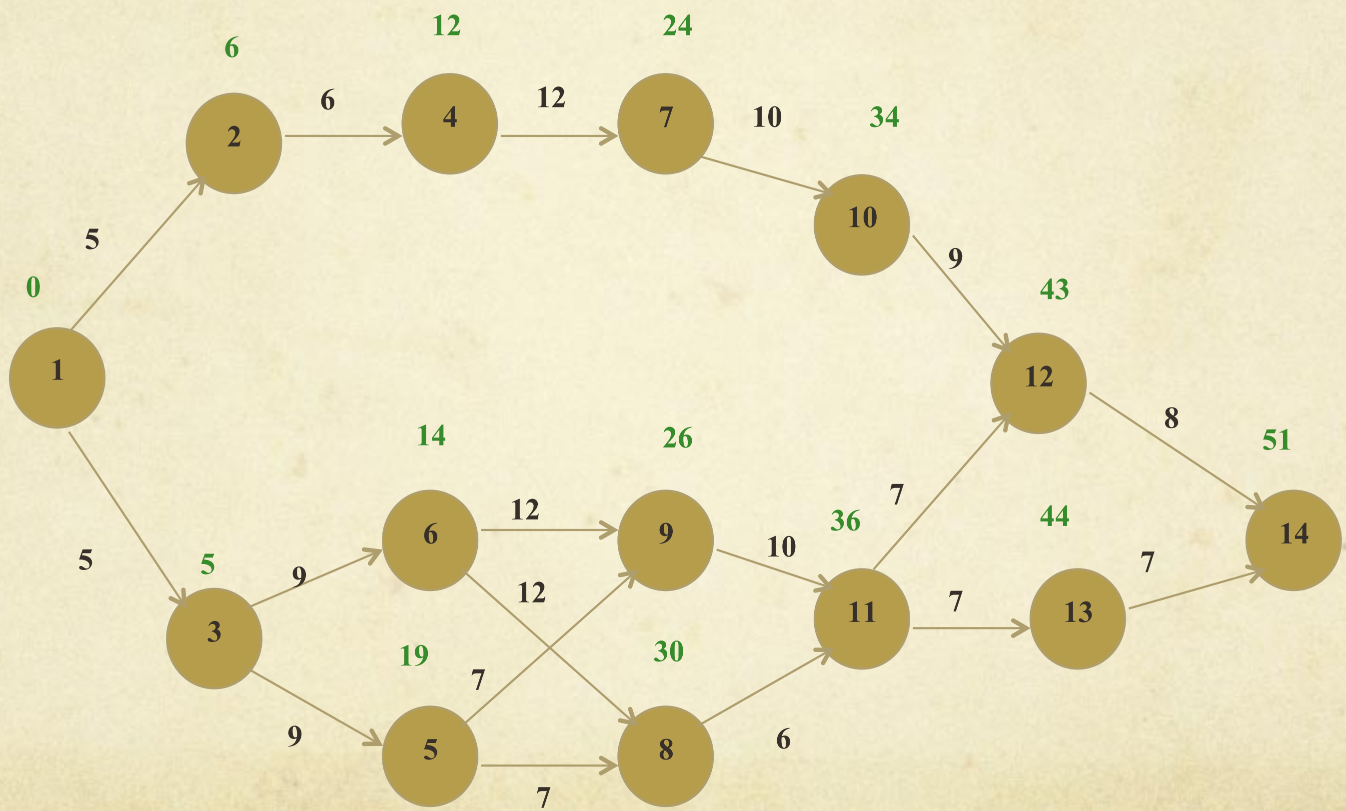
- Soit **D** la durée minimale du projet
- **Dates de début au plus tard :** La date de début au plus tard $\overline{t}_i$ de i est la date maximum à laquelle on peut exécuter i sans retarder la date de fin du projet **D**.
- Dans le cas d'un graphe sans circuit, on a :

$$\overline{t}_{n+1} = D;$$

$$\overline{t}_i = \min_{j \in \Gamma^{+1}(i)} (\overline{t}_j - v_{i,j}),$$

Où $v_{i,j}$ est la valuation de l'arc (i,j) (ici durée de i)

# Ordonnancement au plus tard



# Marges totales & Chemin Critique

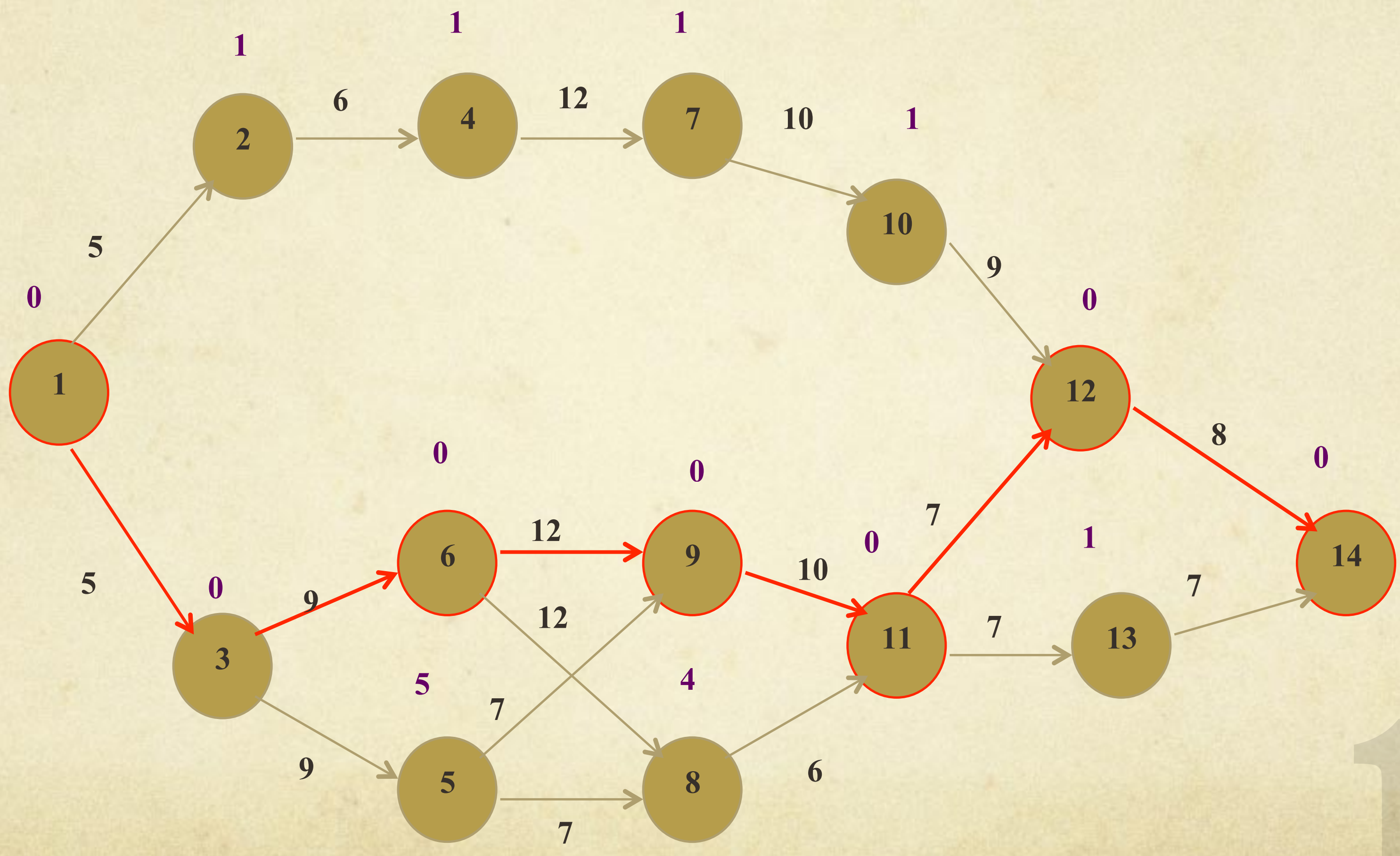
- **Marge totale :** La marge totale d'une tâche i est le retard total qu'on peut se permettre sur i sans remettre en cause la date de fin du projet.

$$MT_i = \overline{t}_i - \underline{t}_i$$

- Les **tâches critiques** ont une marge nulle
  => tout retard sur leur exécution entraîne le même retard sur la date de fin du projet.

- Un **chemin est critique** s'il relie le sommet 0 au sommet (n+1) et s'il ne contient que des tâches critiques.

# Marges totales & Chemin critique



# Marges Libres

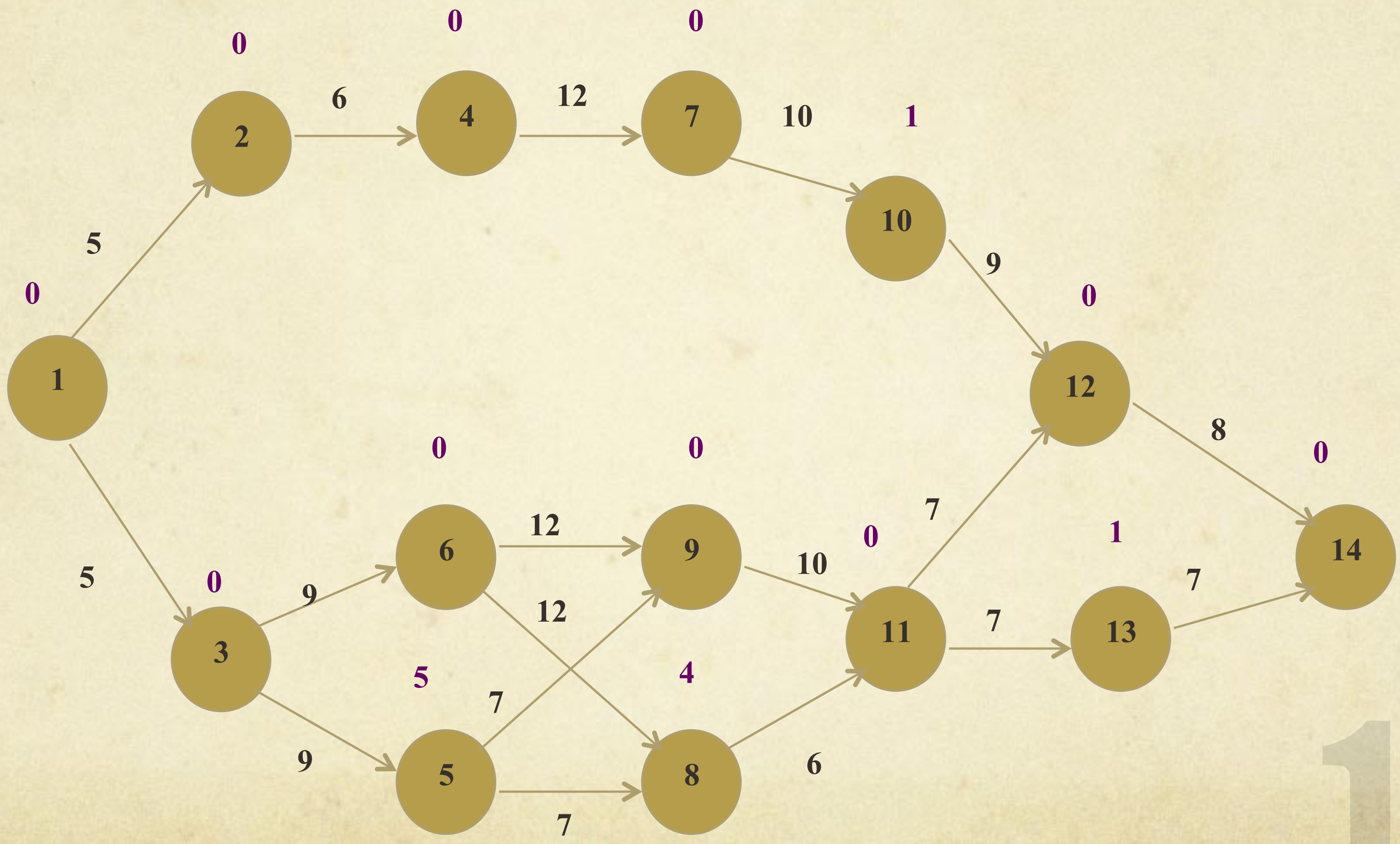
- La **marge libre** d'une tâche i est le retard total qu'on peut se permettre sur i sans retarder l'exécution d'une autre tâche (par rapport aux dates de début au plus tôt).

- $$ML_i = \min_{j \in \Gamma^{+1}(i)} \left\{ \underline{t}_j - \underline{t}_i - v_{i,j} \right\}$$

où $v_{i,j}$ est la valuation de l'arc (i,j) (ici la durée de i)

**Ex :** Calculer les marges libres dans l'exemple 5

# Marges Libres



# Autres variantes des problèmes d'ordonnancement de projets

- Les problèmes de type RCPSP :

Resource-Constrained Project Scheduling Problem

- qui traitent des problèmes d'ordonnancement de projet avec des limitations sur les disponibilités des ressources,  ne font pas l'objet de ce cours

# Notions de programmation dynamique

Sous des conditions peu restrictives, pour un graphe donné :

- Propriété : Toute partie (sous-chemin) d'un chemin optimal est, elle-même, optimale.

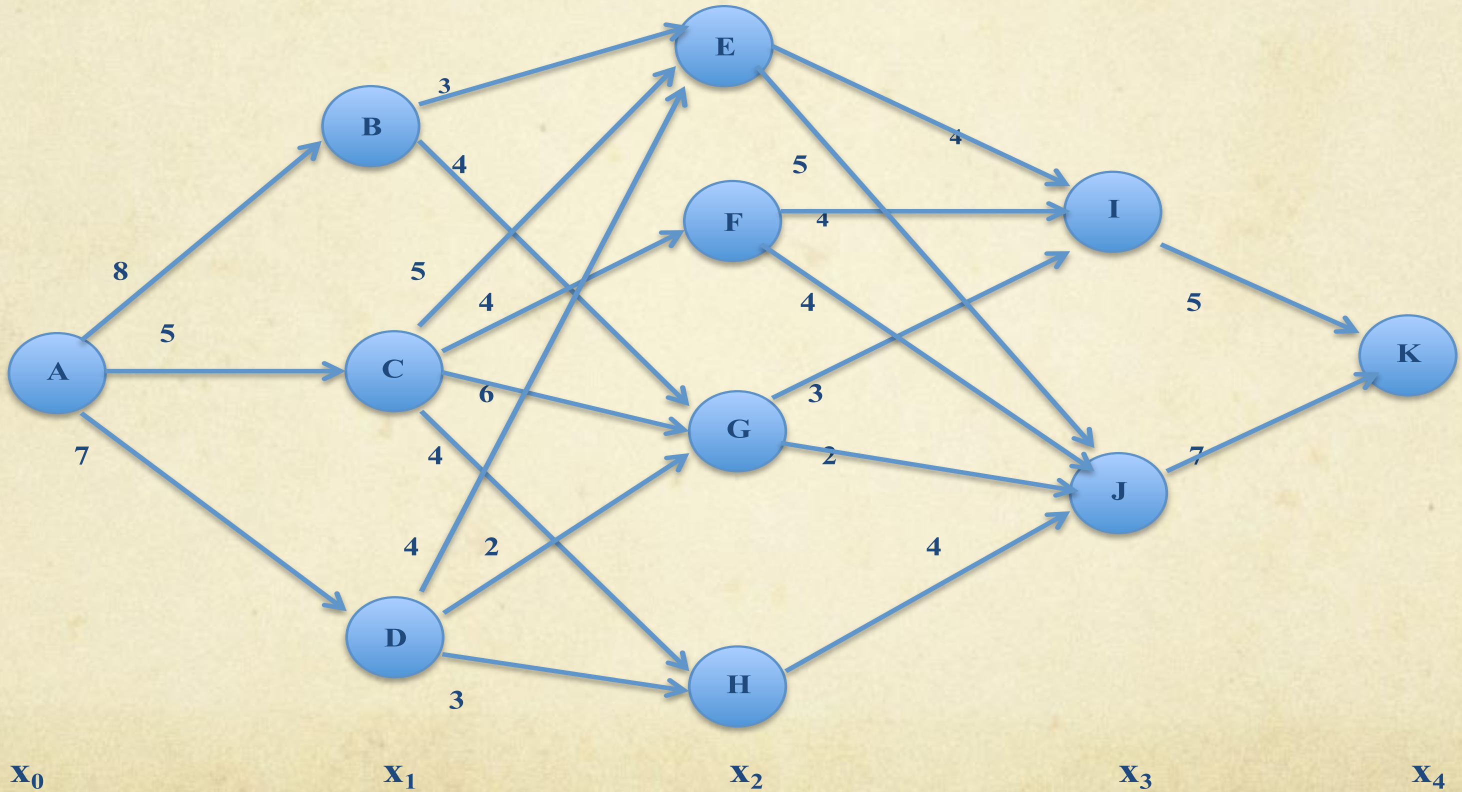
- Preuve : S'il n'en était pas ainsi, on pourrait lui substituer une partie meilleure ce qui améliorerait la valeur de l'ensemble du chemin, préalablement supposé optimal, d'où une contradiction.

⇒ Généralisation sous forme de principe :

Toute sous-politique d'une politique optimale est elle–même, optimale.

# Notions de programmation dynamique

- Exemple : Nous avons à construire une autoroute entre les villes A et K, les sommets intermédiaires pouvant être classés en phases et les arcs du graphe ayant été valués par l'indication des coûts totaux (coûts de réalisation= coûts des chaussées, ouvrages, etc.)



# Notions de programmation dynamique

- Soit $(x_0, x_1, x_2, x_3, x_4)$ : une politique de construction où $x_0$=A; $x_1 \in \{B, C, D\}$ ; $x_2 \in \{E, F, G, H\}$ ; $x_3 \in \{I, J\}$ et $x_4$=K ;
- Son coût est: $F(x_0, x_1, x_2, x_3, x_4) = v(x_0, x_1) + v(x_1, x_2) + v(x_2, x_3) + v(x_3, x_4)$; où v(i,j) désigne le coût de l'arc (i,j).
- Les inconnues du problème sont $x_1$, $x_2$ et $x_3$;
- La meilleure politique est celle qui minimise F.

# Notions de programmation dynamique

**Idée de la PRD :** Par exemple, déterminer pour la variable attachée à chaque phase, le chemin optimal depuis l'origine :

1. déterminer pour les deux premières phases, le sous-chemin optimal entre A et chacun des sommets de $X_2$={E,F,G,H};
2. en ne retenant que les sous-chemins optimaux de l'étape 1, calculer les sous-chemins optimaux entre A et chacun des sommets de $X_3$={I,J};
3. en ne retenant que les sous-chemins optimaux de l'étape 2, calculer les sous-chemins optimaux entre A et chacun des sommets de K.

=> Dessiner les tableaux de calculs correspondants

# Notions de programmation dynamique

| $x_1$ | Chemin opt | Coût |
|---|---|---|
| B | | |
| C | | |
| D | | |

| $x_2$ | Chemins possibles | Chemin opt. | Coût |
|---|---|---|---|
| E | ABE, ACE, ADE | | |
| F | ACF | | |
| G | ABG, ACG, ADG | | |
| H | ACH, ADH | | |

| $x_3$ | Chemins possibles | Chemin opt. | Coût |
|---|---|---|---|
| I | ACEI, ACFI, ADGI | | |
| J | ACEJ, ACFJ, ADGJ, ACHJ | | |

| $x_4$ | Chemins possibles | Chemin opt. | Coût |
|---|---|---|---|
| K | ADGIK, ADGJK | | |

# Notions de programmation dynamique

| $x_1$ | Chemin opt | Coût |
|---|---|---|
| B | AB | 8 |
| C | AC | 5 |
| D | AD | 7 |

| $x_2$ | Chemins possibles | Chemin opt. | Coût |
|---|---|---|---|
| E | ABE, ACE, ADE | ACE | 10 |
| F | ACF | ACF | 9 |
| G | ABG, ACG, ADG | ADG | 9 |
| H | ACH, ADH | ACH | 9 |

| $x_3$ | Chemins possibles | Chemin opt. | Coût |
|---|---|---|---|
| I | ACEI, ACFI, ADGI | ADGI | 12 |
| J | ACEJ, ACFJ, ADGJ, ACHJ | ADGJ | 11 |

| $x_4$ | Chemins possibles | Chemin opt. | Coût |
|---|---|---|---|
| K | ADGIK, ADGJK | ADGIK | 17 |

# Notions de programmation dynamique

Equations de récurrence :

$$f_{k+1}^{*}(x_{k+1}) = \underset{x_k \in X_k}{OPT}[f_k^{*}(x_k) + v_{k+1}(x_k, x_{k+1})]$$

Avec

$f_k^{*}(x_k)$ La valeur optimale des chemins entre A et chacun des sommets $x_k$ de l'ensemble $X_k$ ;

$v_{k+1}(x_k, x_{k+1})$ La valeur de l'arc $(x_k, x_{k+1})$

$x_k \in X_k; \; x_{k+1} \in X_{k+1}$

# Notions de programmation dynamique

Etape 1 : $f_1^*(B) = 8;\ f_1^*(C) = 5;\ f_1^*(D) = 7;$

Etape 2 : Calculer, pour chaque $x_2$ de $X_2$ les valeurs optimales $f_2^*(x_2)$

$$f_2^*(E) = \underset{x_1 \in X_1}{OPT}[f_1^*(x_1) + v_2(x_1, E)]$$

$$= \underset{x_1 \in X_1}{OPT}[f_1^*(B) + v_2(B,E); f_1^*(C) + v_2(C,E); f_1^*(D) + v_2(D,E)]$$

$$= OPT[11;10;11] = 10 \quad => \quad (A,C,E)\ est\ optimal$$

$$f_2^*(F) = \underset{x_1 \in X_1}{OPT}[f_1^*(x_1) + v_2(x_1, F)]$$

$$= \underset{x_1 \in X_1}{OPT}[f_1^*(C) + v_2(C,F)] = 9 \quad => \quad (A,C,F)\ est\ optimal$$

$$f_2^*(G) = \underset{x_1 \in X_1}{OPT}[f_1^*(x_1) + v_2(x_1, G)]$$

$$= \underset{x_1 \in X_1}{OPT}[f_1^*(B) + v_2(B,G); f_1^*(C) + v_2(C,G); f_1^*(D) + v_2(D,G)]$$

$$= OPT[12;11;9] = 9 \quad => (A,D,G)\ est\ optimal$$

# Notions de programmation dynamique

Etape 2 (suite) : $f_2^*(H) = \underset{x_1 \in X_1}{OPT}[f_1^*(x_1) + v_2(x_1, H)]$

$$= \underset{x_1 \in X_1}{OPT}[f_1^*(C) + v_2(C,H); f_1^*(D) + v_2(D,H)]$$

$$= OPT[9;10] = 9 \quad \Rightarrow \quad (A,C,H) \ est \ optimal$$

Etape 3 : Calculer, pour chaque $x_3$ de $X_3$ les valeurs optimales $f_3^*(x_3)$

$$f_3^*(I) = \underset{x_2 \in X_2}{OPT}[f_2^*(x_2) + v_3(x_2, I)]$$

$$= \underset{x_2 \in X_2}{OPT}[f_2^*(E) + v_3(E,I); f_2^*(F) + v_3(F,I); f_2^*(G) + v_3(G,I)]$$

$$= \underset{x_2 \in X_2}{OPT}[10+4; 9+4; 9+3] = 12 \Rightarrow (A,D,G,I) \ est \ optimal$$

$$f_3^*(J) = \underset{x_2 \in X_2}{OPT}[f_2^*(x_2) + v_3(x_2, J)]$$

$$= \underset{x_2 \in X_2}{OPT}[f_2^*(E) + v_3(E,J); f_2^*(F) + v_3(F,J); f_2^*(G) + v_3(G,J); f_2^*(H) + v_3(H,J)]$$

$$= \underset{x_2 \in X_2}{OPT}[10+5; 9+4; 9+2; 9+4] = 11 \Rightarrow (A,D,G,J) \ est \ optimal$$

# Notions de programmation dynamique

Etape 4 : Calculer $f_4^*(K)$

$$f_4^*(K) = \underset{x_3 \in X_3}{OPT}[f_3^*(x_3) + v_4(x_3, K)]$$

$$= \underset{x_3 \in X_3}{OPT}[f_3^*(I) + v_4(I, K); f_3^*(J) + v_4(J, K)]$$

$$= \underset{x_3 \in X_3}{OPT}[12 + 5; 11 + 7] = 17 \quad \Rightarrow \quad (A, D, G, I, K) \textit{ est optimal}$$

⇒ Economie de calcul par rapport à une énumération naïve des 64 chemins de 4 arcs : cela permet de réaliser 18 additions de 2 nombres au lieu de 192 (=3 × 64)

⇒ Dans ce type de problème, le calcul aurait pu commencer de K en remontant vers A (Backward dynamic programming) **au lieu du** Forward dynamic programming