

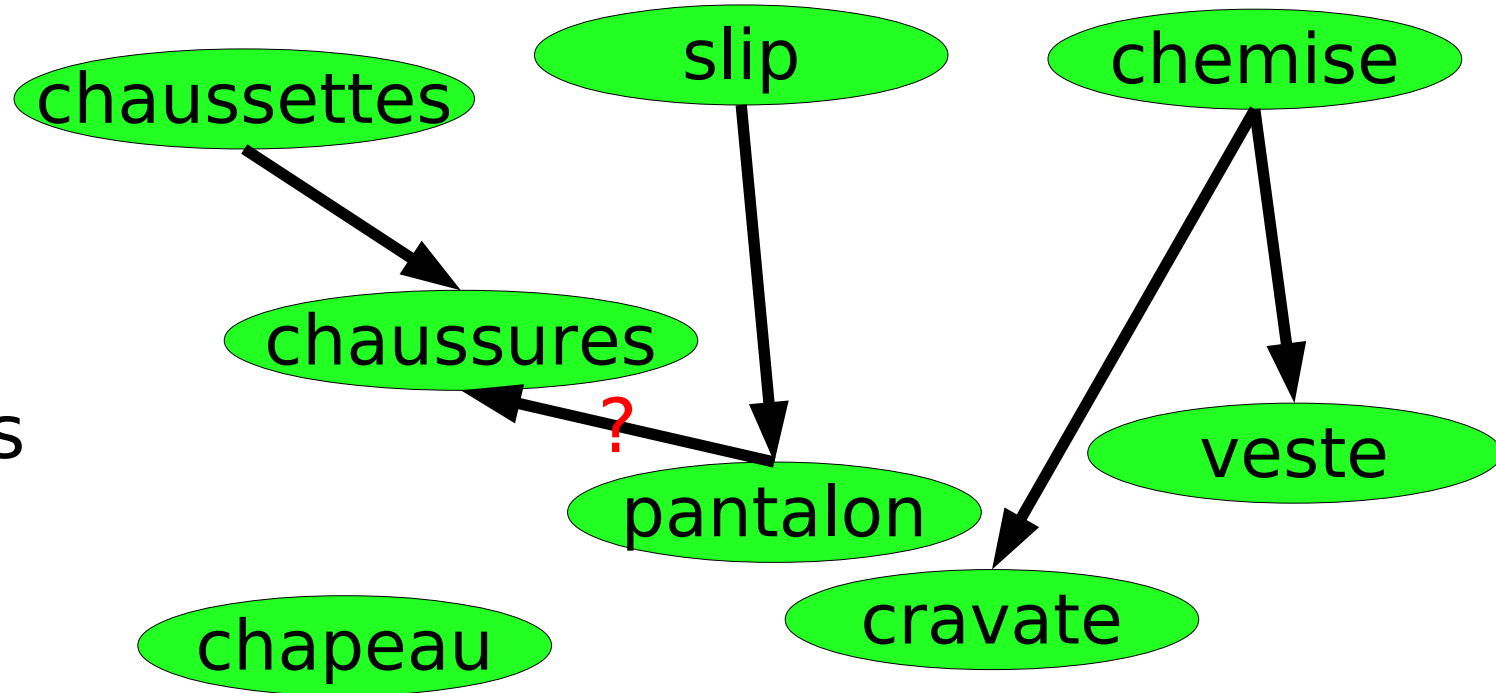
Ordonnancement

- Gestion et planification d'un projet
 - décomposable en travaux et tâches
 - avec des ressources
 - avec des contraintes horaires (durées)
 - avec des contraintes de précédence/simultanéité entre tâches
- Ordonnancement :
 - calendrier de débuts de tâches
 - allocation de ressources

Contraintes d'antériorité - s'habiller

- pantalon
- veste
- chaussures
- chaussettes
- slip
- cravate
- chemise
- bonnet

- Antériorité sur certaines tâches
 - graphe



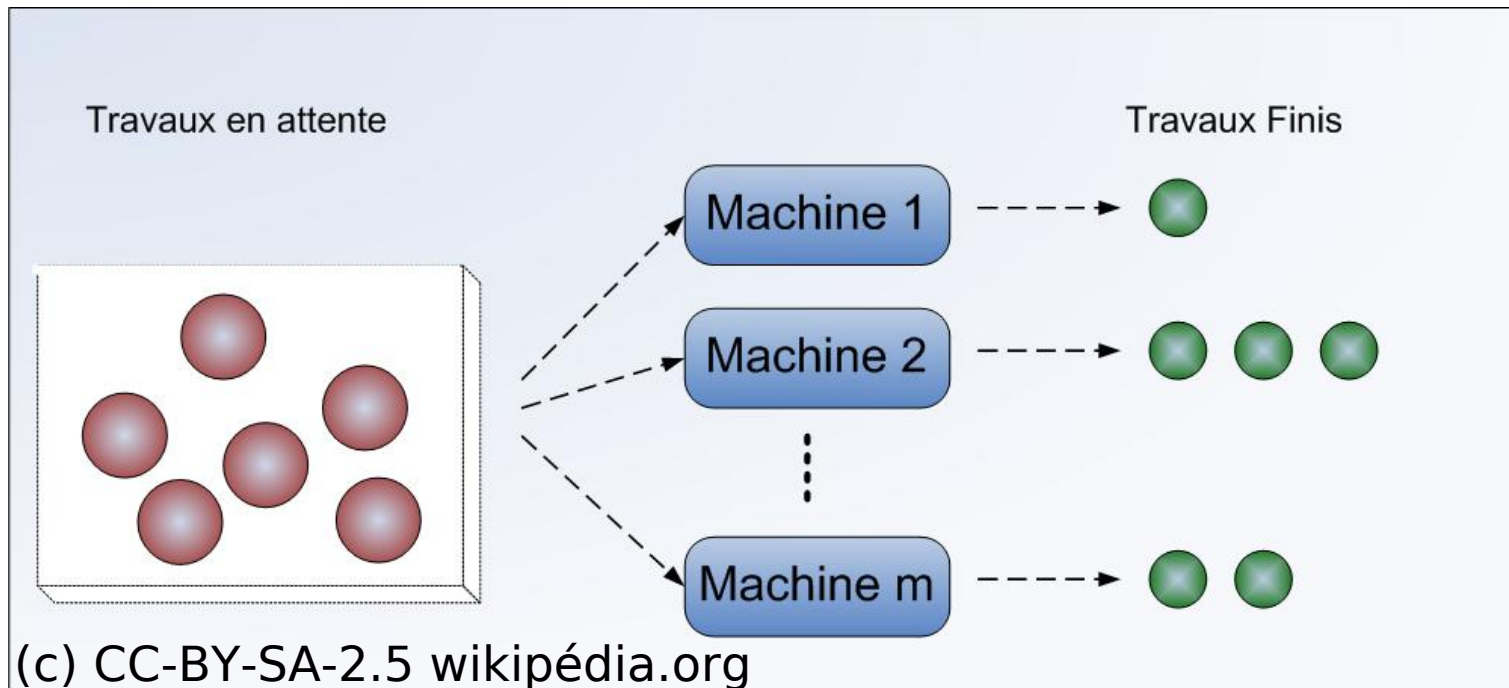
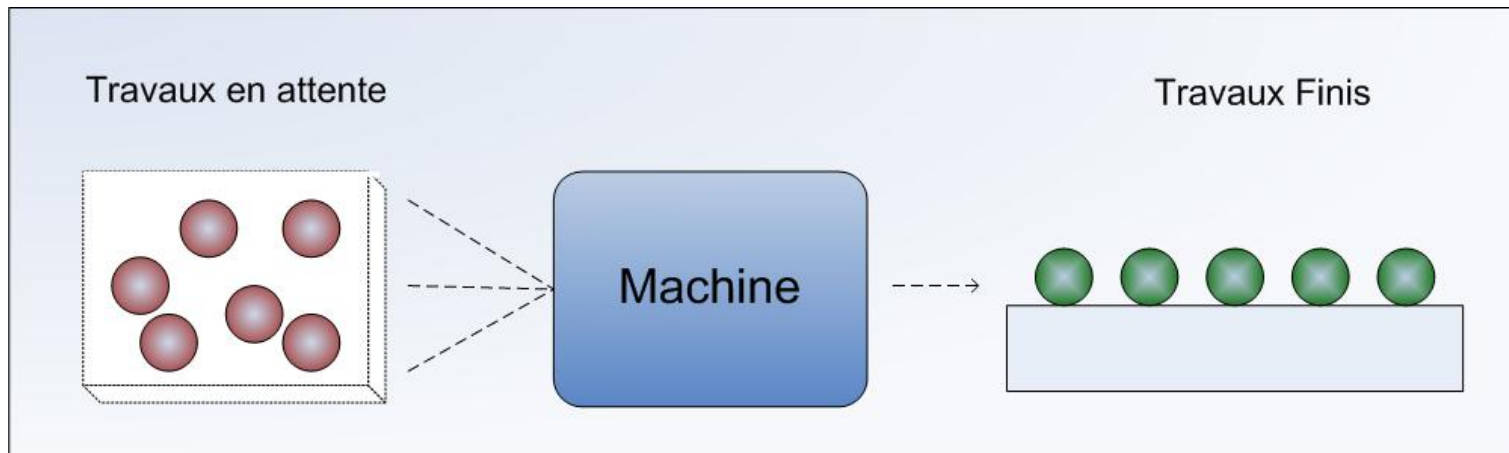
Ordonnancement d'ateliers

- Un ensemble de jobs à réaliser. Chacun d'entre eux est composé d'un ensemble de tâches
- Classification des ordonnancements d'atelier suivant les contraintes de précédence entre les tâches qui composent un job

Pour chaque job :	
tâches indépendantes	openshop
ordre identique sur les tâches	flowshop
ordre différents sur les tâches	jobshop

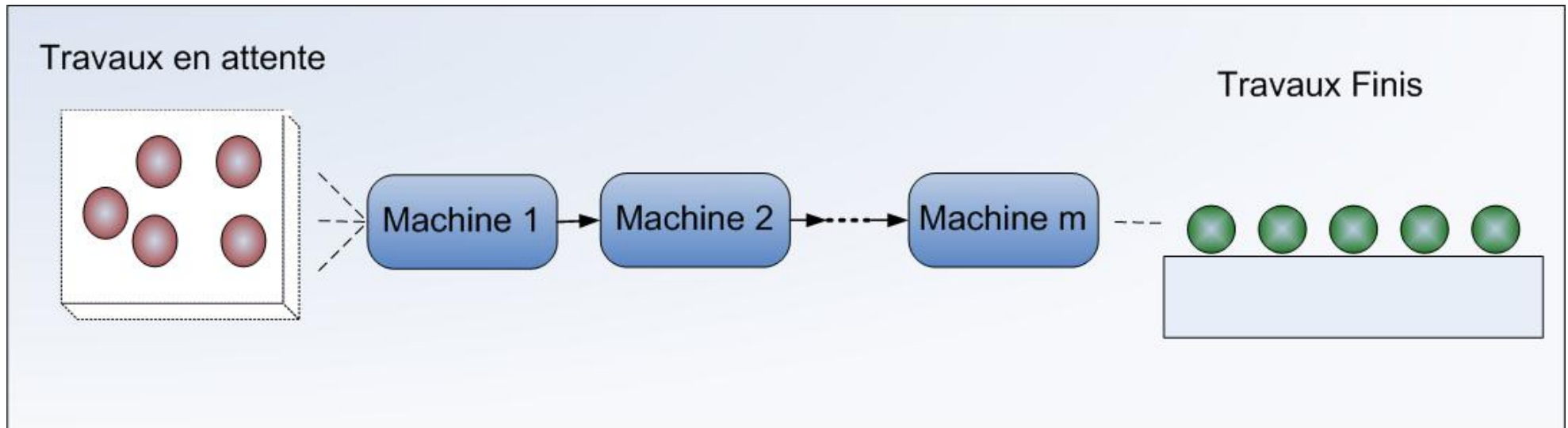
Contraintes de précédence (1)

- **Openshop**: tâches du job indépendantes



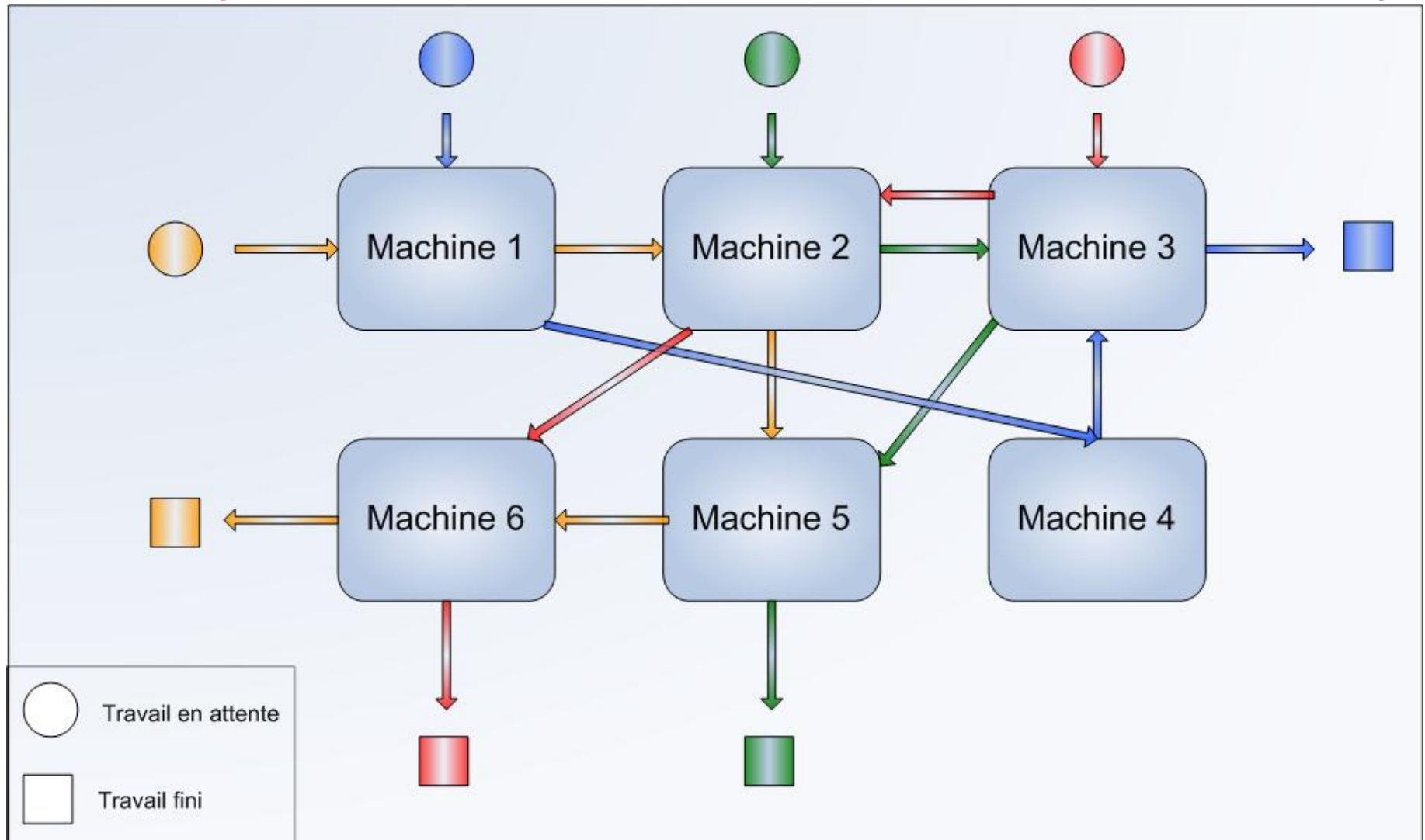
Contraintes de précédence (2)

- **Flowshop** : même ordre sur toutes les tâches des différents jobs



Contraintes de précédence (3)

- **Jobshop** : ordre des tâches différent suivant le job



Contraintes de temps

- Chaque tâche a
 - une durée
 - éventuellement des contraintes d'antériorité
 - éventuellement une plage temporelle de réalisation
- L'ordonnancement cherche à optimiser
 - une durée totale (*makespan*)
 - un respect de dates au plus tard
 - une minimisation de coût
 - ...

Formulation du problème (1)

- Un ensemble $T = \{T_1, T_2, \dots, T_n\}$ de tâches à réaliser
 - p_i : durée de T_i
 - r_i : date de début au plus tôt d'exécution pour T_i
 - d_i : date de fin au plus tard d'exécution pour T_i
 - t_i : date de début effectif d'exécution pour T_i
 - c_i : date de fin effective d'exécution pour T_i
 - W_{ik} : coût associé à T_i sur machine k
 - P_{ik} : temps d'exécution associé à T_i sur machine k

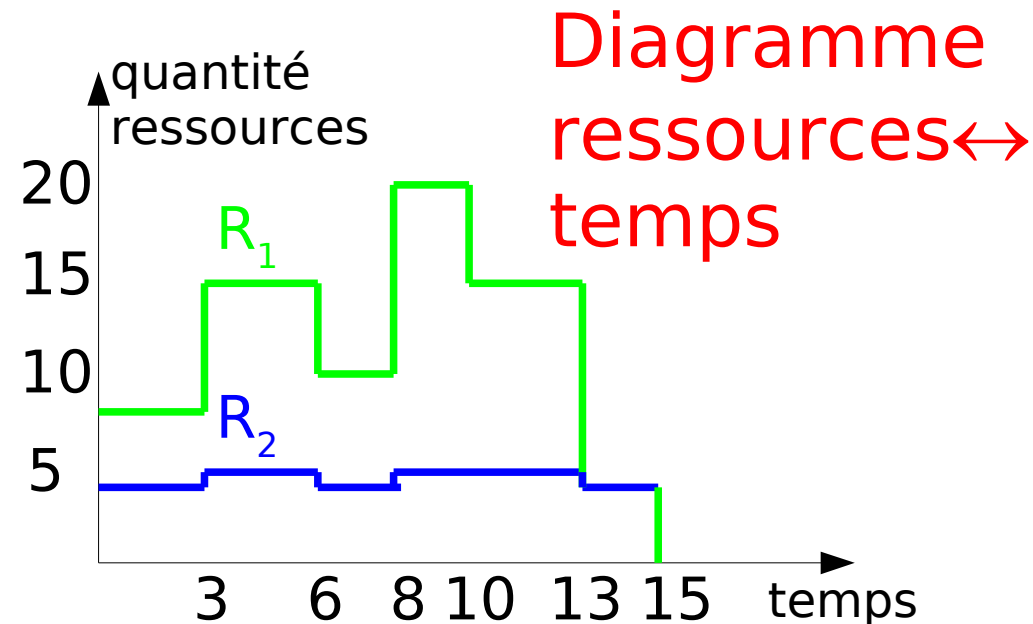
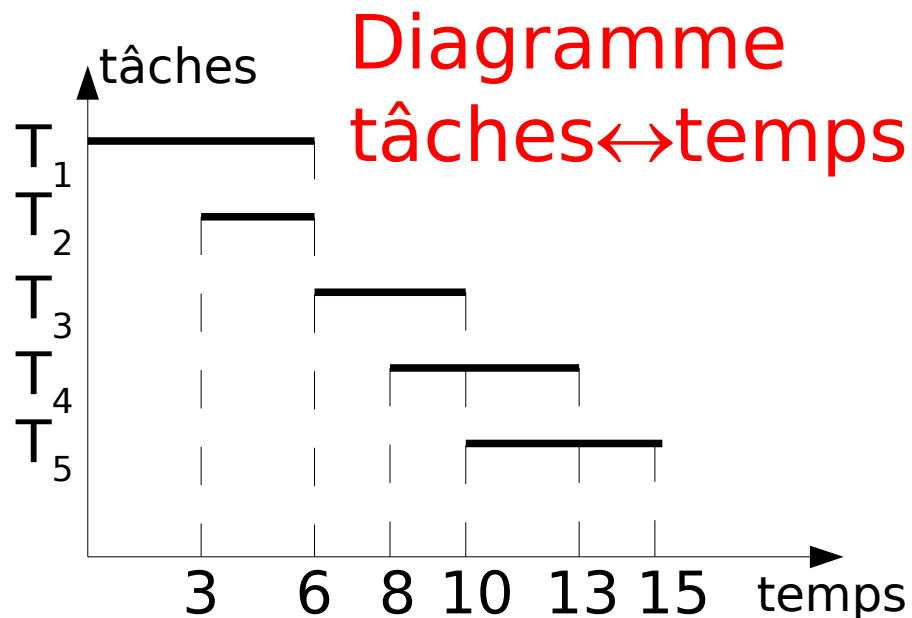
Formulation du problème (2)

- Pour chaque tâche T_i il faut que
 - $p_i = c_i - t_i$
 - $r_i \leq t_i \leq c_i \leq d_i$
- Si des contraintes de ressources sur des machines existent : $P_{ik} = c_i - t_i$ (∞ si T_i non exécutable sur k)
- Si des contraintes d'antériorité (ou de terminaison partielle a_{ij}) existent entre T_i et T_j : $t_i + a_{ij} \cdot p_i \leq t_j$
- Optimiser le coût total (suivant $\sum W_{ik}$) ou la durée totale en fonction des contraintes

Représentation du problème

Diagrammes de Gantt

- Visualisation d'une solution
 - $A = \{T_1, T_2, T_3, T_4, T_5\}$
 - Avec les durées $p_1=6, p_2=3, p_3=4, p_4=5, p_5=5$,
 - Et utilisant respectivement 8, 7, 10, 10, 4 et 4, 1, 3, 2, 3 unités de ressources 1 et 2



Gantt pour Jobshop

- Pour visualiser une solution

- $J_1 = \{(M_1, 2), (M_2, 1), (M_4, 2)\}$

- $J_2 = \{(M_2, 2), (M_3, 1), (M_4, 1)\}$

- $J_3 = \{(M_3, 1), (M_4, 2)\}$

- $J_4 = \{(M_1, 1), (M_2, 1), (M_3, 2)\}$

Données du
problème
jobshop

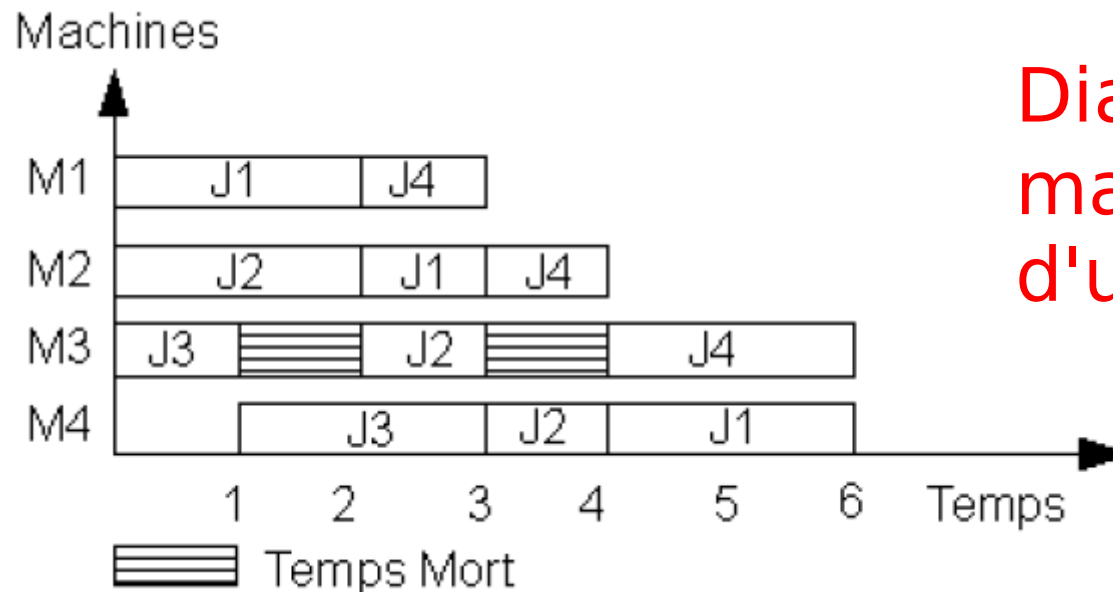


Diagramme
machines↔temps
d'une solution

Résolution dans les cas simples

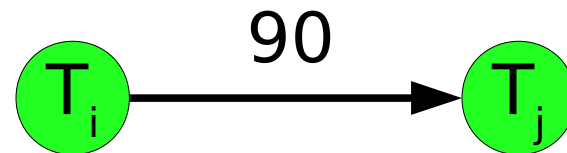
- Pour chaque tâche
 - durée
 - contraintes d'antériorité
- Trouver un ordonnancement des tâches avec une date d'exécution pour chacune d'entre elles
- Tâches **critiques** : ne peut être retardées sans reporter la fin du projet

Représentation par graphes

- 2 représentations
 - **Potentiel \leftrightarrow tâches**
MPM : Méthode Potentiels Metra
tâches sur les sommets, potentiels sur les arcs
(B Roy – superstructures du France)
 - **Potentiel \leftrightarrow étapes**
PERT : Program Evaluation and Review Technique
tâches sur les arcs, potentiels sur les sommets
(US Navy pour les 6000 constructeurs missiles
Polaris - 1950)

Méthode MPM

- Méthode Potentiels Metra
 - tâches sur les sommets (T_i)
 - contraintes temporelles sur les arcs (t_i : durée de T_i)



- précédences entre tâches
- deux tâches fictives de début et fin
- tout sommet sans prédecesseur est relié à début (0)
- tout sommet sans successeur est relié à fin

Représentation MPM

Exemple - repas

Tâches

- T_1 : menu (30 min)
- T_2 : achats (90 min)
- T_3 : apéritif (30 min)
- T_4 : ménage (10 min)
- T_5 : mettre la table (10 min)
- T_6 : épluchage (30 min)
- T_7 : cuisiner (30 min)
- T_8 : servir (10 min)

Contraintes

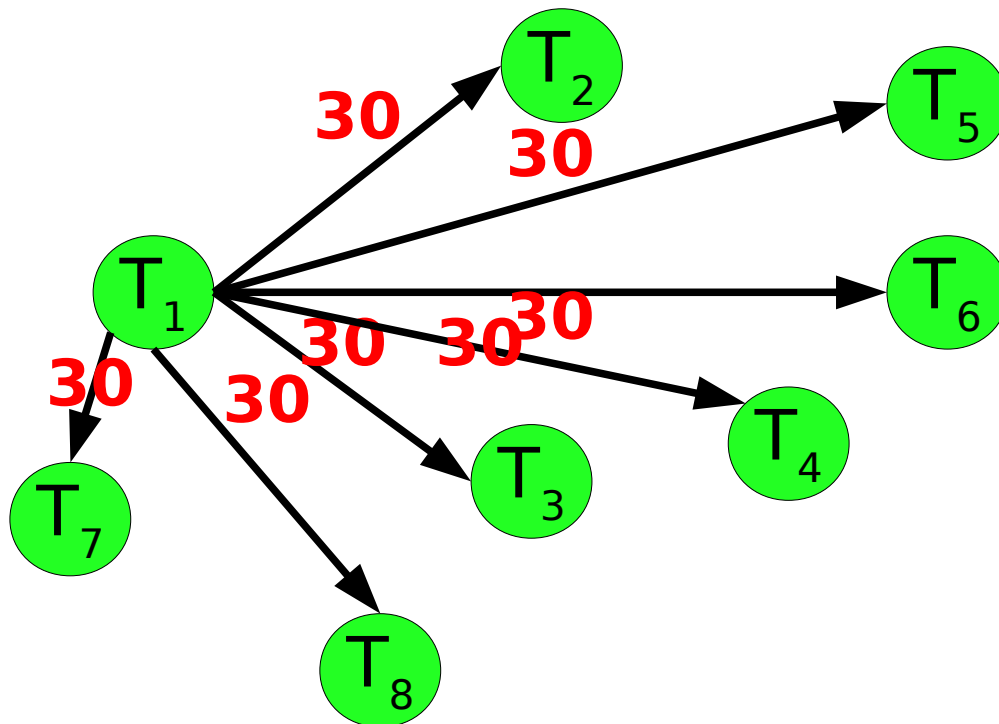
- Menu en premier
- Servir après cuisiner
- Ménage avant table
- Eplucher avant cuisiner
- Achats avant épluchage
- Apéritif avant servir

Représentation MPM

Exemple - repas

Contraintes

- Menu (T1) en premier
- Servir après cuisiner
- Ménage avant table
- Eplucher avant cuisiner
- Achats avant épluchage
- Apéritif avant servir



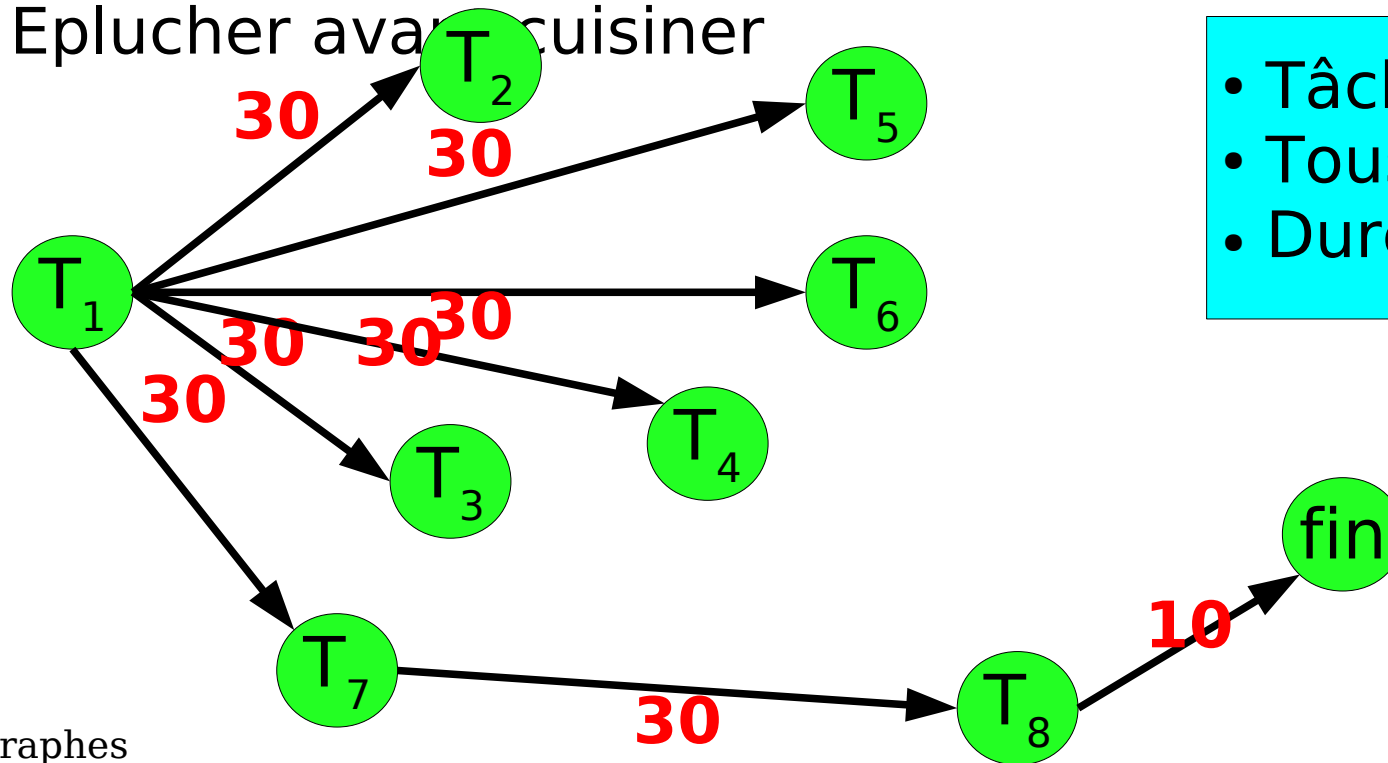
- T_1 sert de tâche initiale
- Pas de tâche finale

Représentation MPM

Exemple - repas

Contraintes

- Menu en premier
- Servir (T_8) après cuisiner (T_7)
- Ménage avant table
- Eplucher avant cuisiner
- Achats avant épluchage
- Apéritif avant servir



- Tâche finale
- Tous reliés à fin
- Durée de T_8

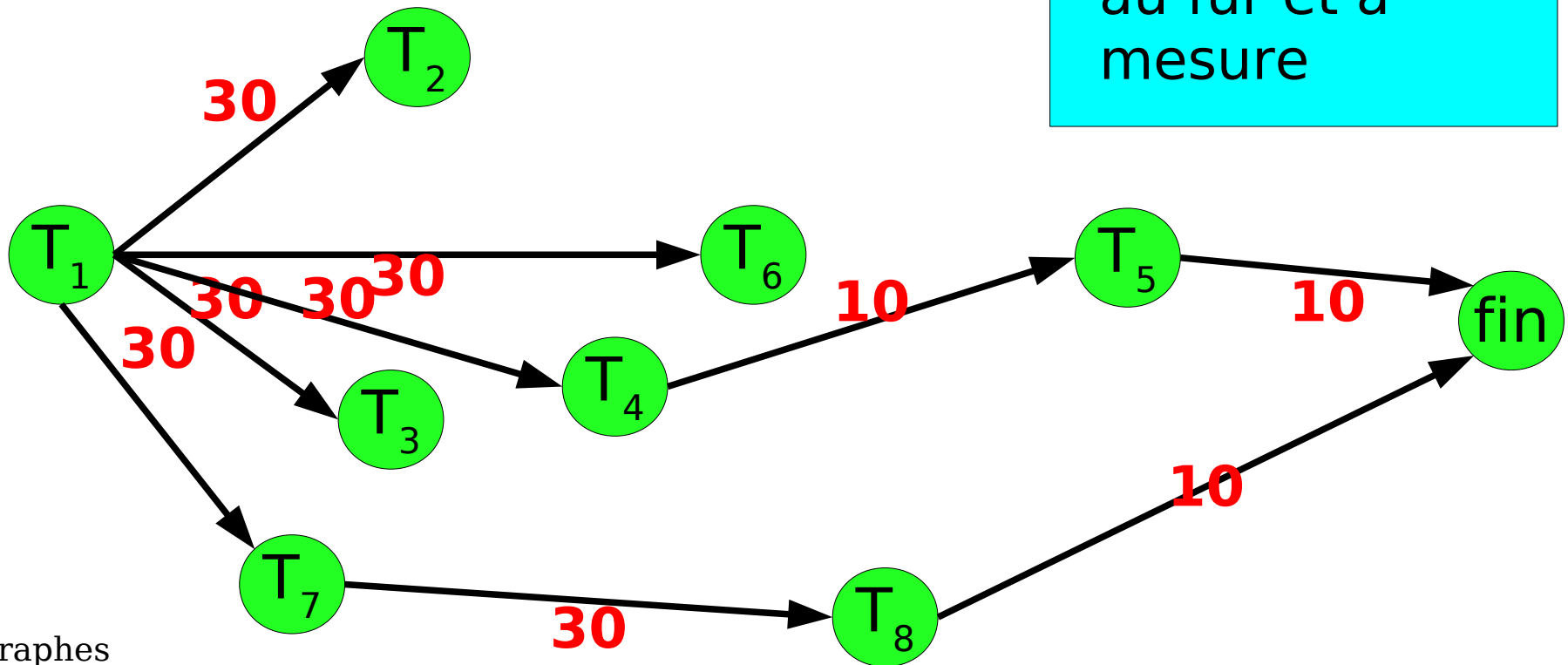
Représentation MPM

Exemple - repas

Contraintes

- Menu en premier
- Servir après cuisiner
- Ménage (T_4) avant table (T_5)
- Eplucher avant cuisiner
- Achats avant épluchage
- Apéritif avant servir

• Tous reliés à fin au fur et à mesure

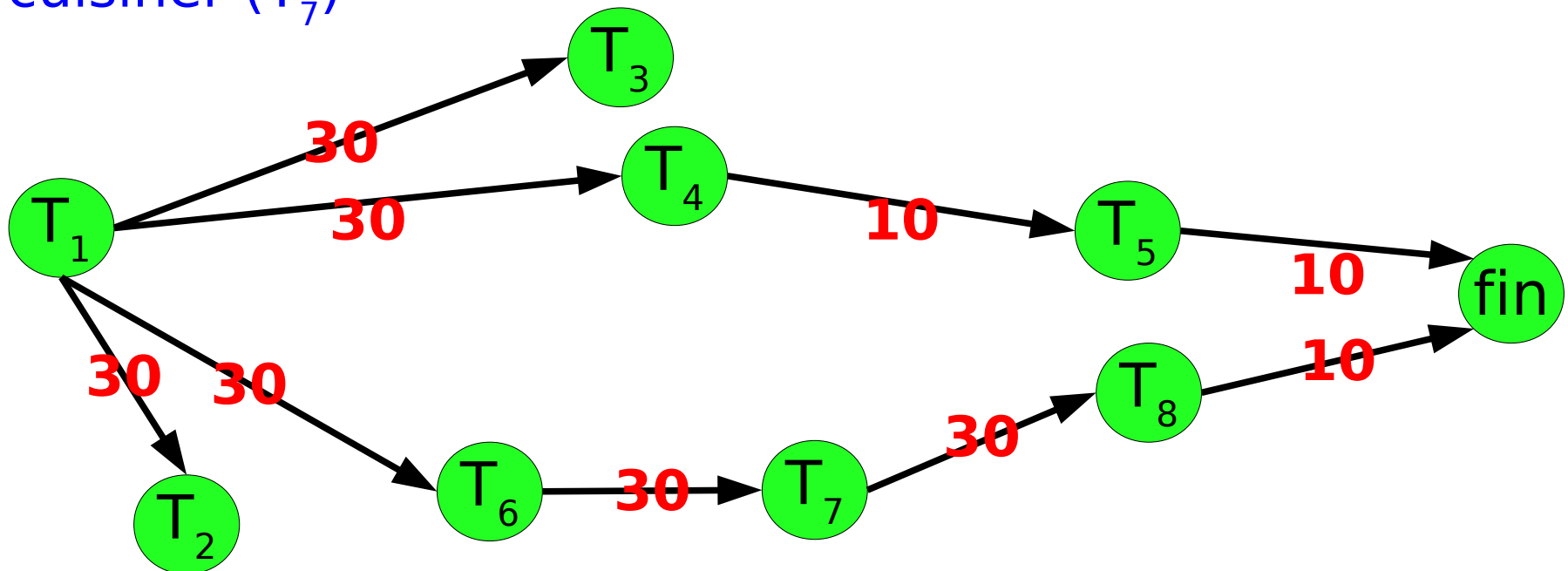


Représentation MPM

Exemple - repas

Contraintes

- Menu en premier
- Servir après cuisiner
- Ménage avant table
- Eplucher (T_6) avant cuisiner (T_7)
- Achats avant épluchage
- Apéritif avant servir

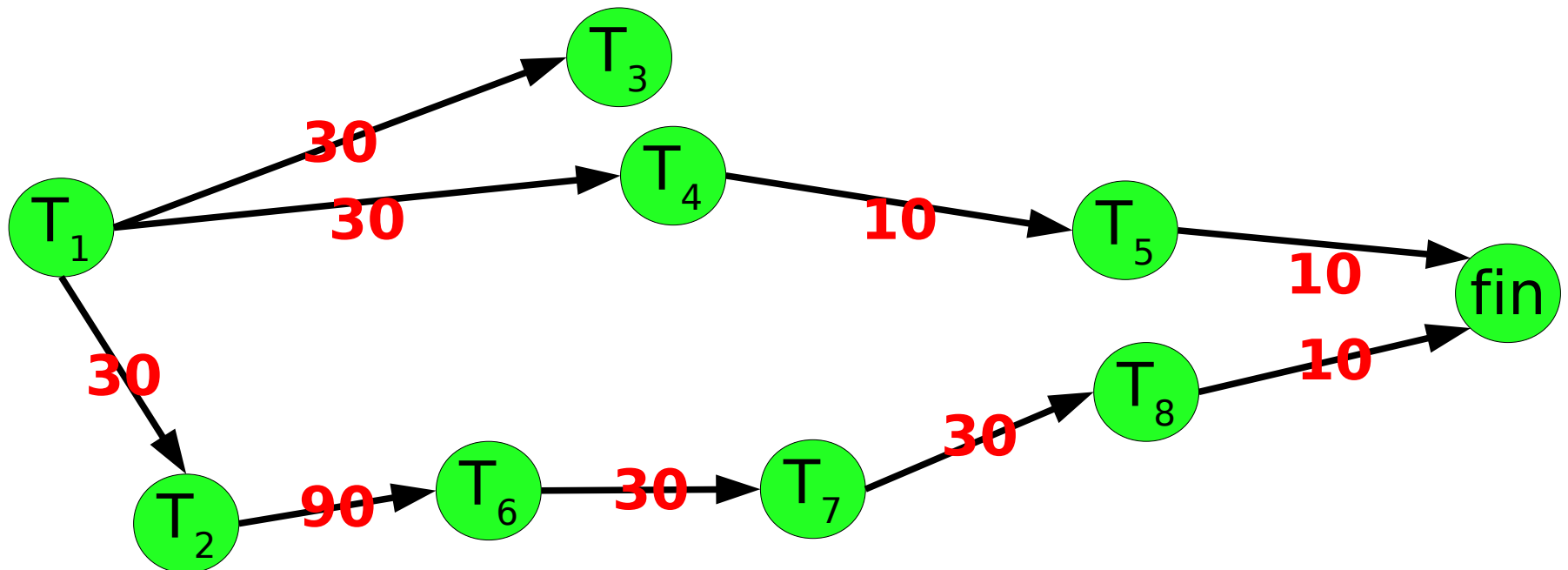


Représentation MPM

Exemple - repas

Contraintes

- Menu en premier
- Servir après cuisiner
- Ménage avant table
- Eplucher avant cuisiner
- Achats (T_2) avant épluchage (T_6)
- Apéritif avant servir



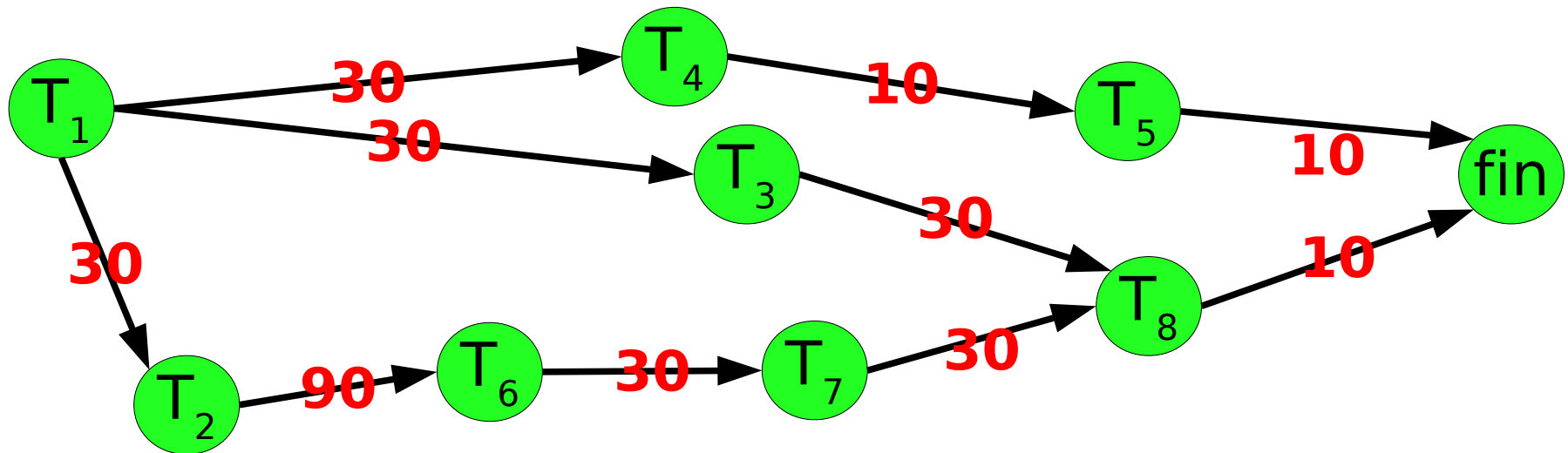
Représentation MPM

Exemple - repas

Contraintes

- Menu en premier
- Servir après cuisiner
- Ménage avant table
- Eplucher avant cuisiner
- Achats avant épluchage
- Apéritif (T_3) avant servir (T_8)

• Graphe final sans redondances

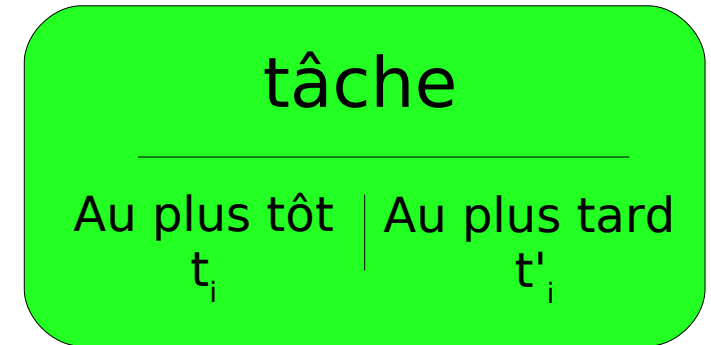


Représentation MPM

Calcul des dates

- Sommets : tâches T_i avec :

- des dates au plus tôt t_i
- et dates au plus tard t'_i



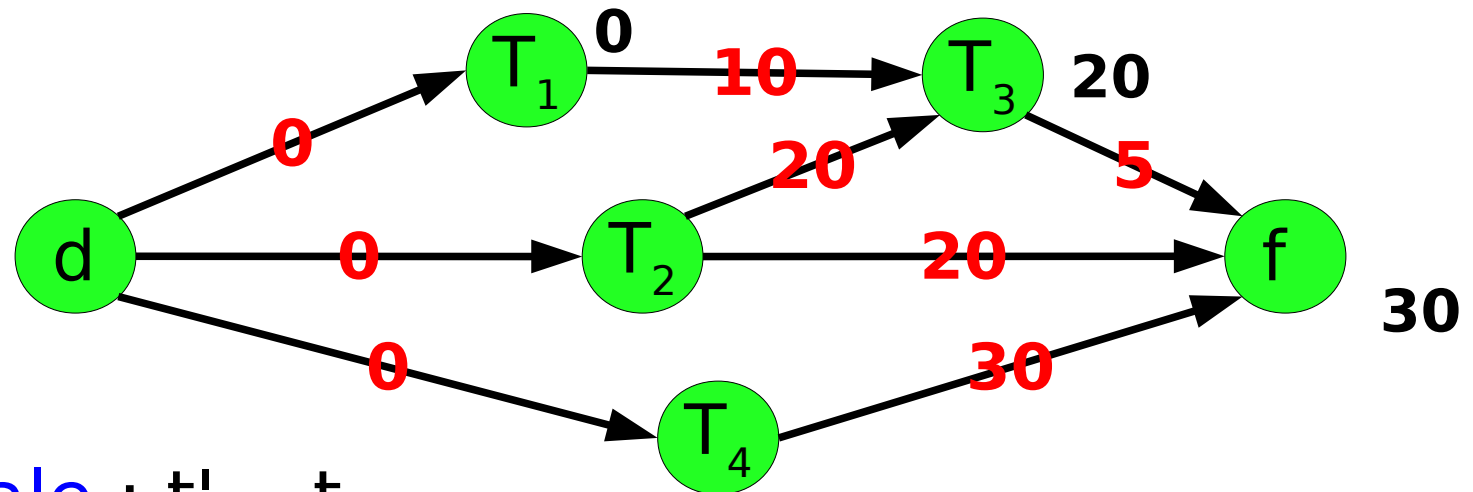
- Une étape **début**, une étape **fin** de projet
- Le but de l'algorithme calcDates est de calculer les dates au plus tôt/au plus tard pour chaque tâche :
 - Marge totale : $t'_i - t_i$
 - Marge libre : $\min_{T_j \in \Gamma(T_i)} t_j - (t_i + d_i)$
 - Identifier les **chemins critiques** : marge totale nulle

Représentation MPM

Calcul des dates

- **Marge libre** : $\min_{T_j \in \Gamma(T_i)} t_j - (t_i + d_i)$

Temps maximal dont la tâche peut être retardée ou allongée sans retarder la fin du projet, indépendamment des autres tâches (10 pour T_1)
(sans impact possible sur les tâches)



- **Marge totale** : $t'_i - t_i$

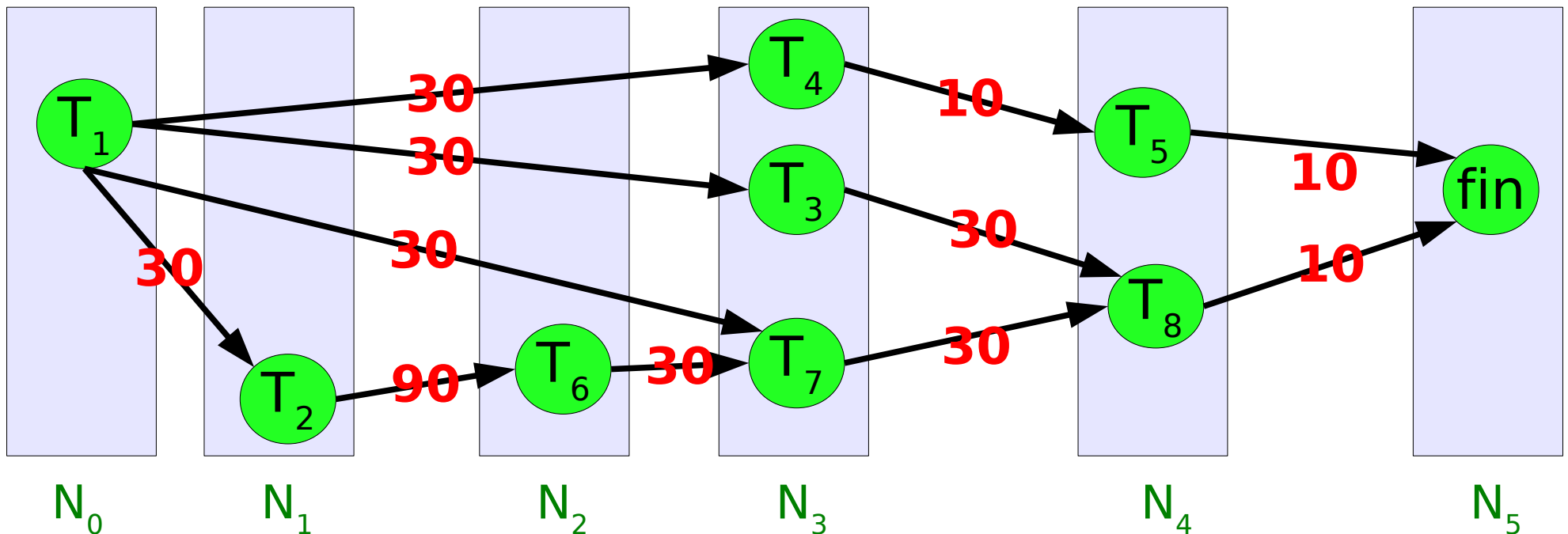
Temps maximal dont la tâche peut être retardée ou allongée sans retarder la fin du projet (15 pour T_1)

(impact possible sur le début des autres tâches)

Représentation MPM

Calcul des dates

- 2 passes :
 - Calcul des dates au plus tôt t_i
 - Puis des dates au plus tard t'_i
- Tri topologique des sommets

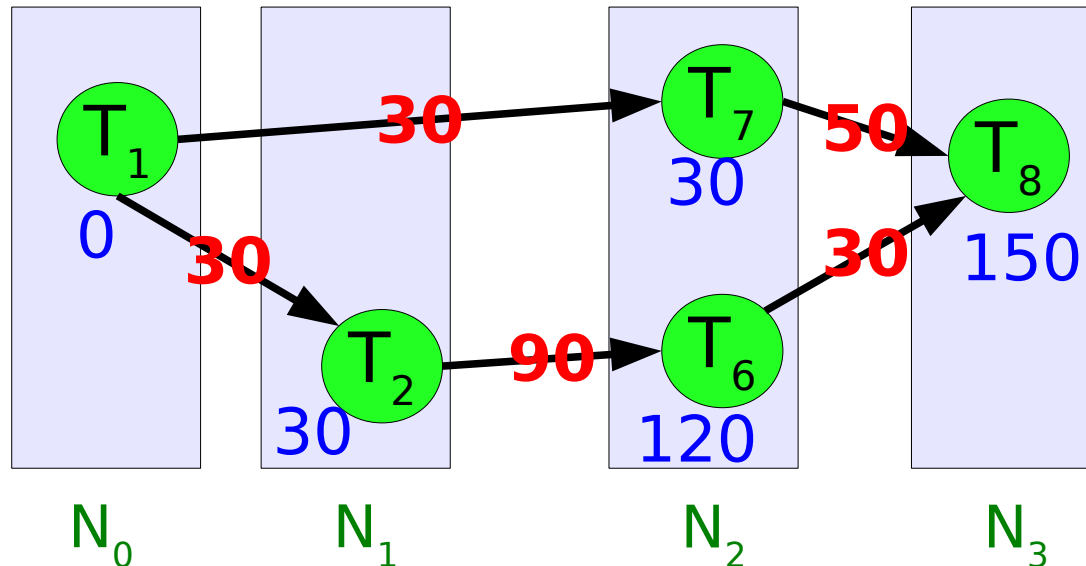


Représentation MPM

Calcul des dates au plus tôt

- Date t_i à laquelle peut commencer l'exécution d'une tâche
 - Tri topologique, et $t_i=0$ au niveau N_0
 - Pour chaque niveau, à partir du niveau N_1
 - Pour tous les sommets à ce niveau

$$t_i = \max_{T_j \in \text{préds}(T_i)} t_j + d_j$$

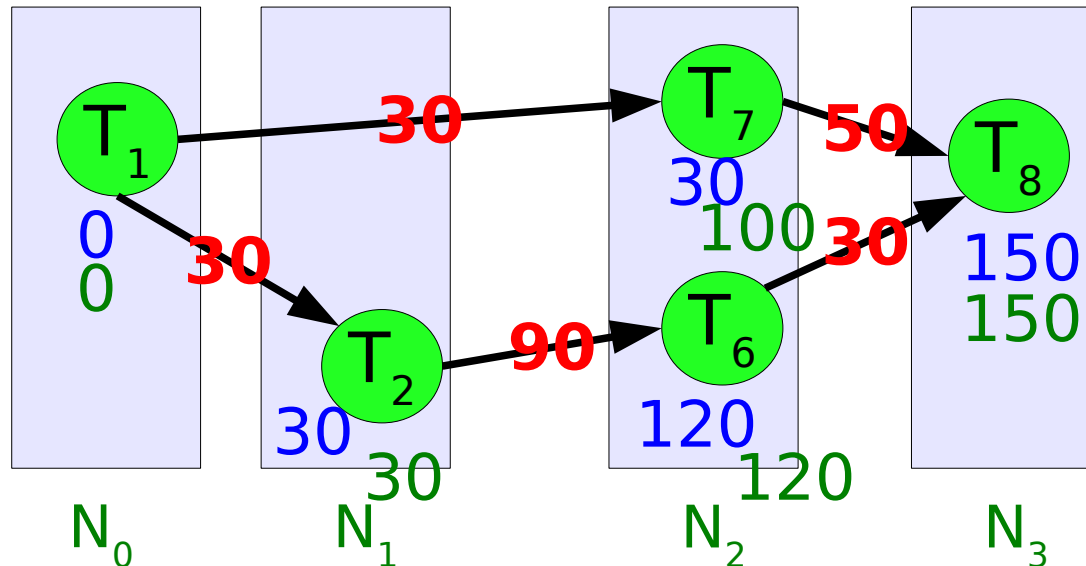


Représentation MPM

Calcul des dates au plus tard

- Date t_i' à laquelle peut commencer l'exécution d'une tâche **sans retarder les tâches suivantes**
 - Pour chaque niveau, par ordre décroissant à partir du dernier
 - Pour tous les sommets à ce niveau

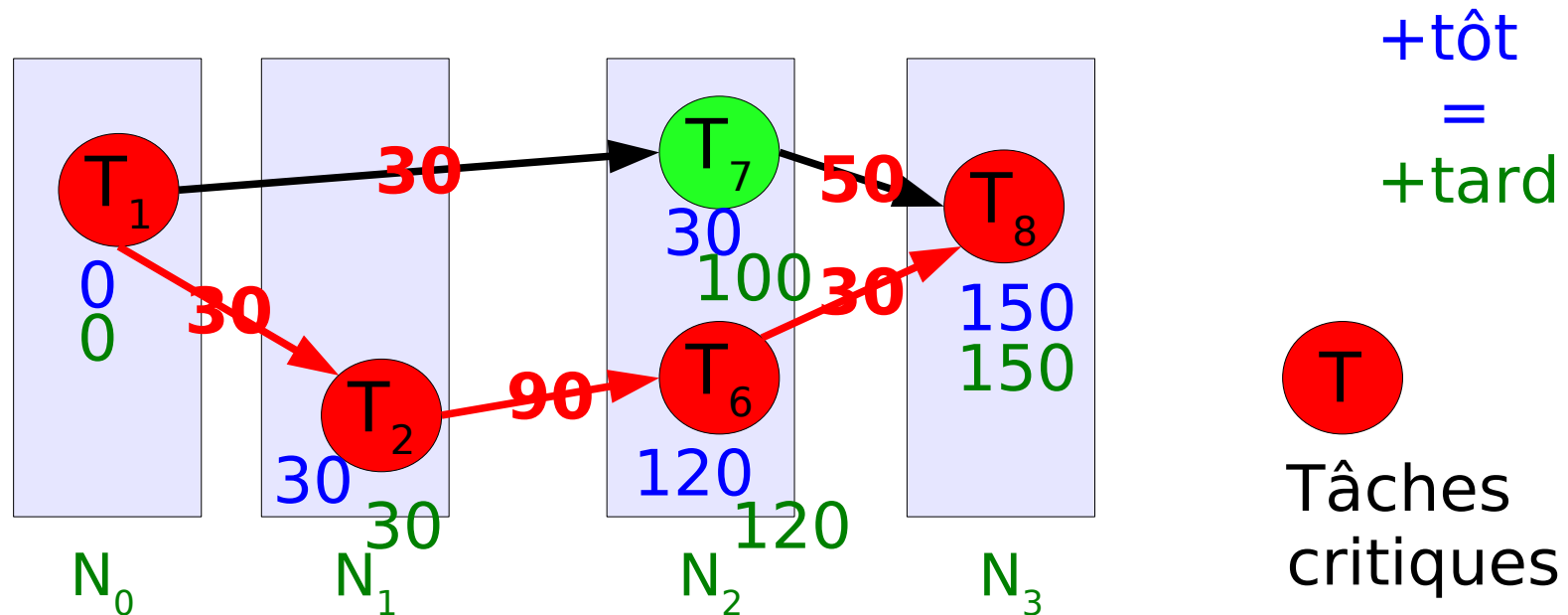
$$t_i' = \min_{T_j \in \text{succs}(T_i)} t_j' - d_i$$



Représentation MPM

Chemin critique

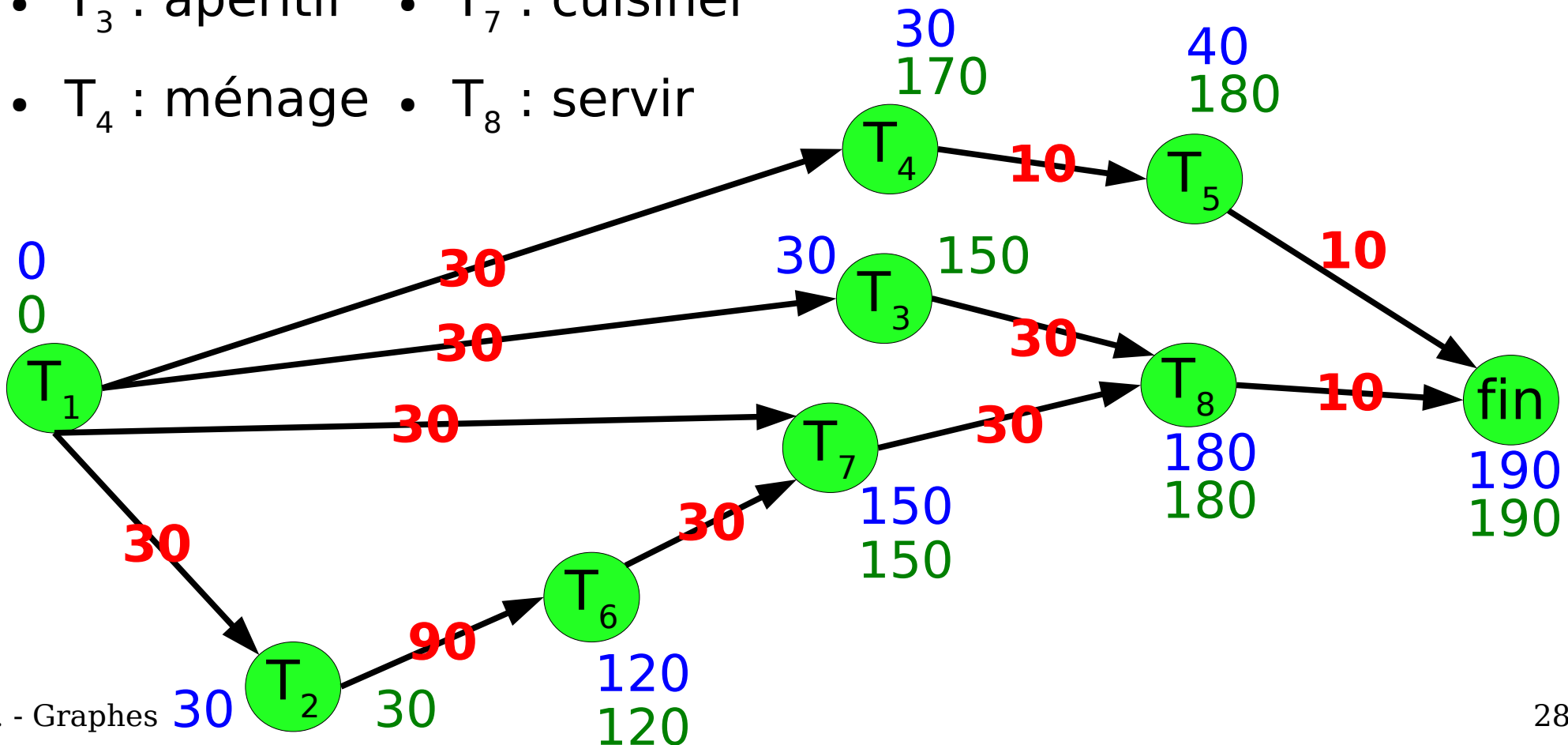
- Dates t_i et t_i' identiques : tâches ne pouvant être retarder sans **sans retarder l'ensemble du projet**



Représentation MPM

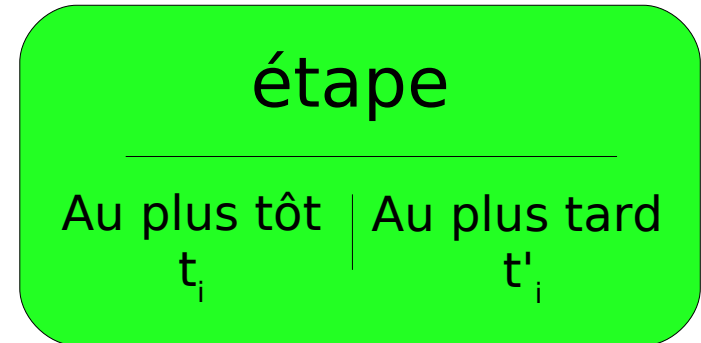
Calcul des dates

- T_1 : menu
- T_5 : mettre la table
- T_2 : achats
- T_6 : épluchage
- T_3 : apéritif
- T_7 : cuisiner
- T_4 : ménage
- T_8 : servir



Représentation PERT potentiels ↔ étapes

- Sommets : étapes
 - Les arcs représentent les tâches



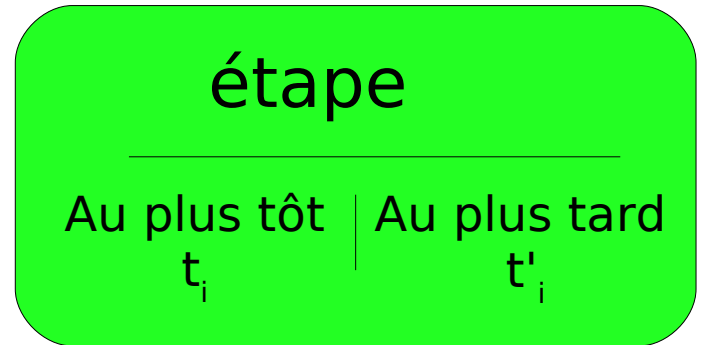
- Une étape **début**, une étape **fin** de projet
- Même but de calcul du chemin critique et de détermination de la durée totale du projet que avec MPM (date de l'étape de fin)

Représentation PERT

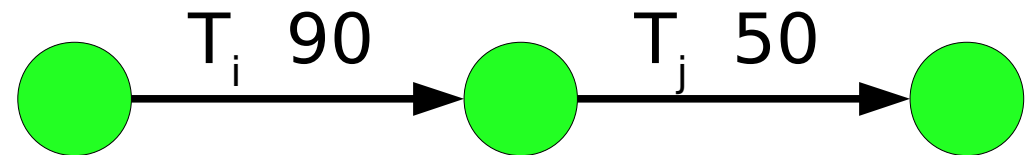
Tâches

- Arcs : tâches

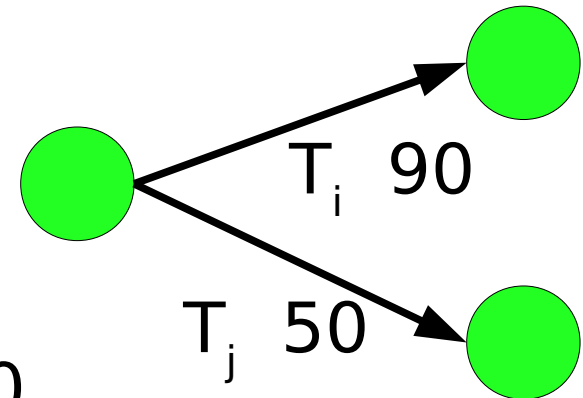
- nom
 - durée



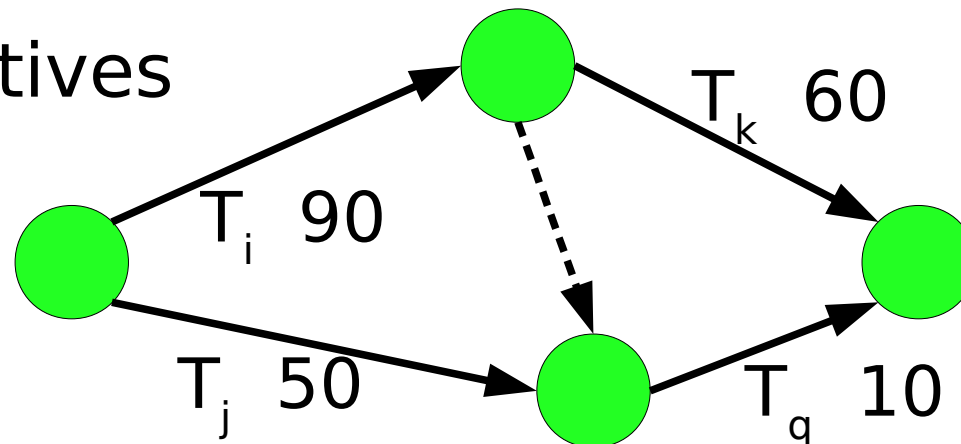
- Tâches successives



- Tâches simultanées



- Tâches fictives

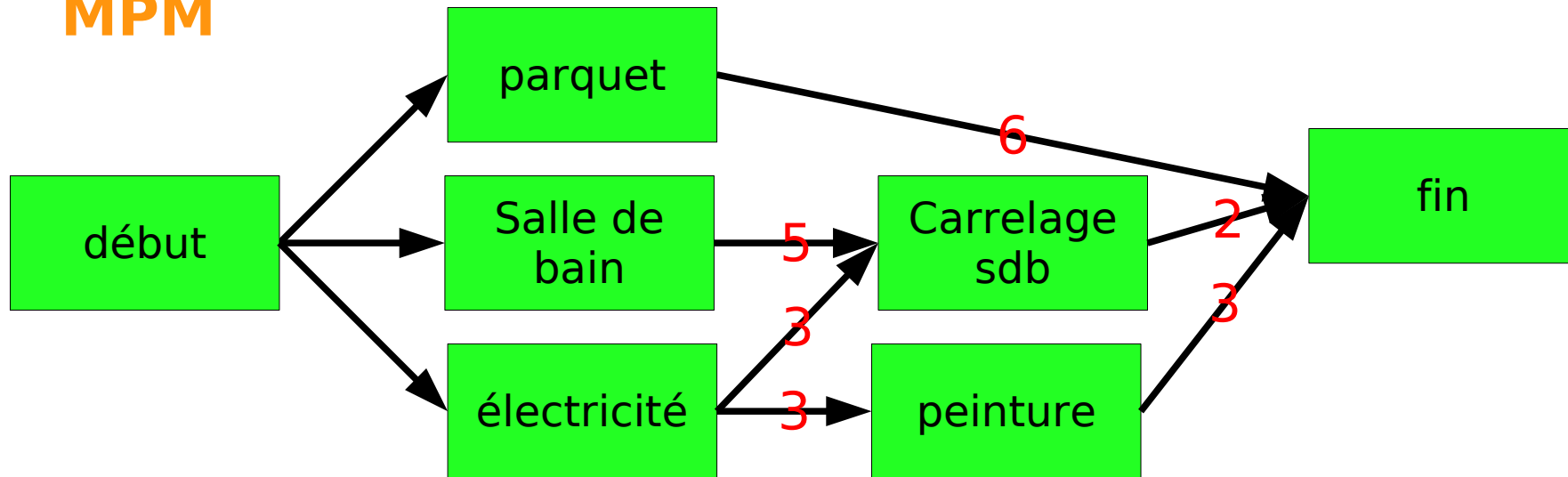


T_i précède T_q

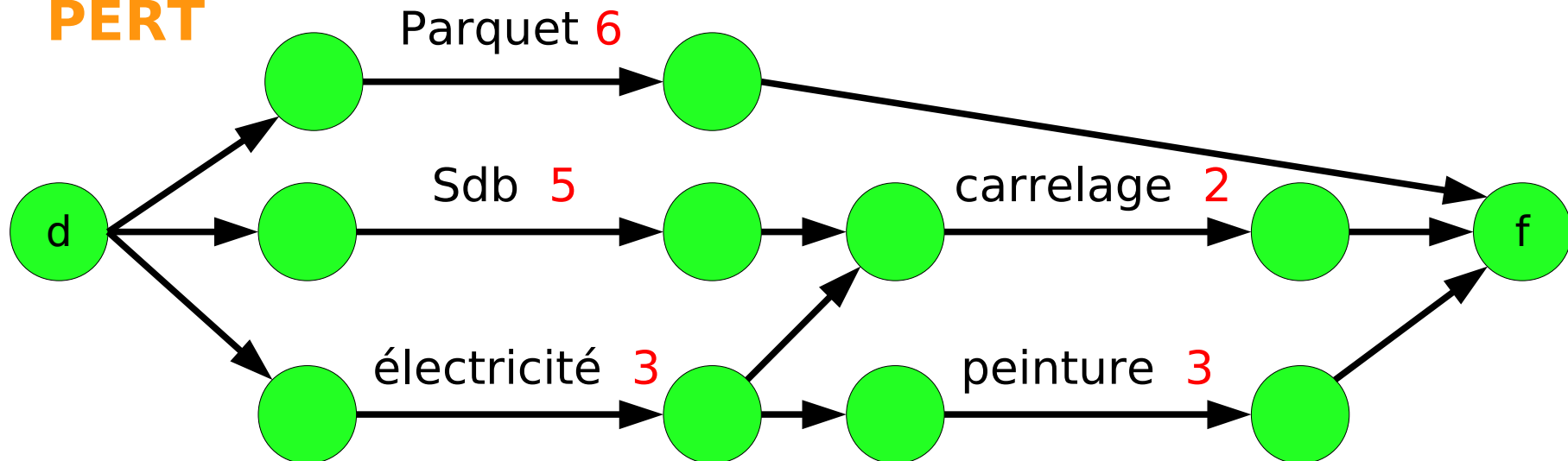
Représentations MPM et PERT

Exemple - travaux

MPM



PERT



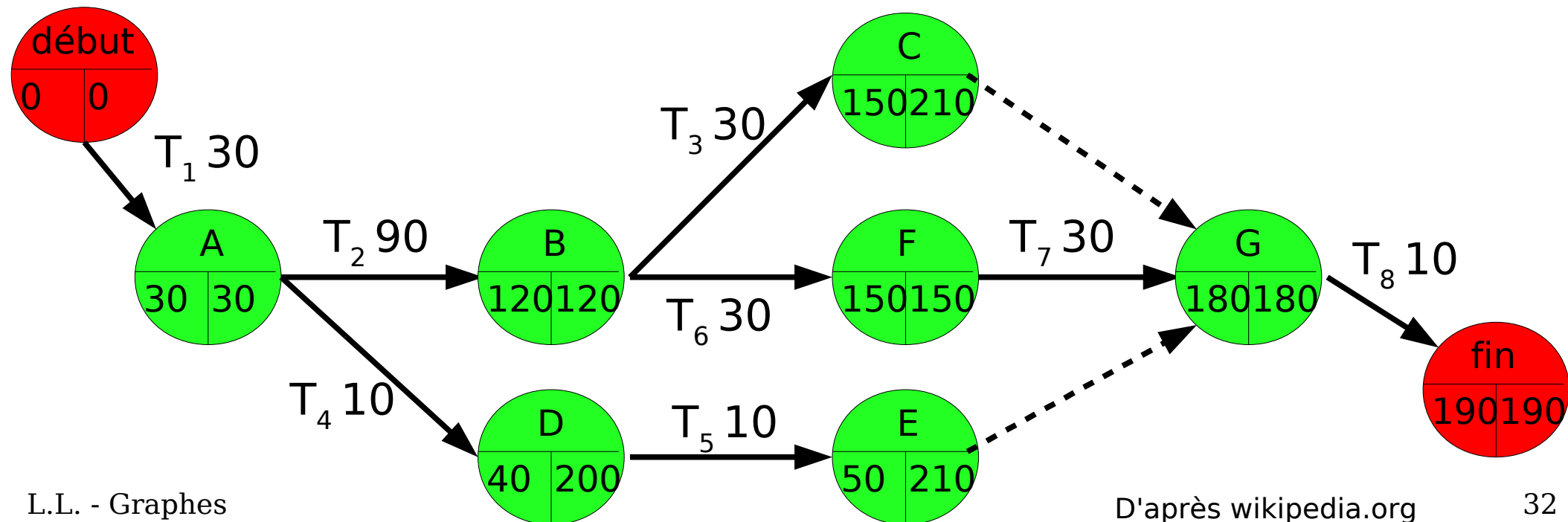
220

Représentation PERT

Exemple - repas

- T_1 : menu (30 min)
- T_2 : achats (90 min)
- T_3 : apéritif (30 min)
- T_4 : ménage (10 min)

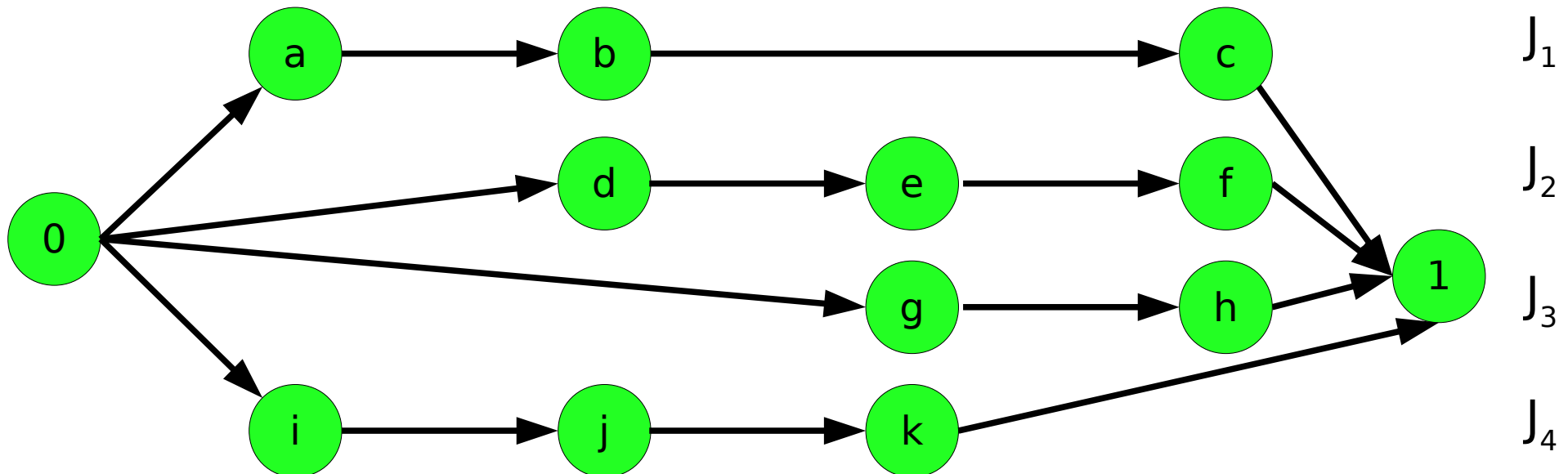
- T_5 : mettre la table (10 min)
- T_6 : épluchage (30 min)
- T_7 : cuisiner (60 min)
- T_8 : servir (10 min)



Représentation Jobshop graphe disjonctif

- Exemple :
- $J_1 = \{a=(M_1,2), b=(M_2, 1), c=(M_4, 2)\}$
- $J_2 = \{d=(M_2,2), e=(M_3, 1), f=(M_4, 1)\}$
- $J_3 = \{g=(M_3, 1), h=(M_4, 2)\}$
- $J_4 = \{i=(M_1,1), j=(M_2, 1), k=(M_3, 2)\}$

**Contraintes de
précédence des
taches**

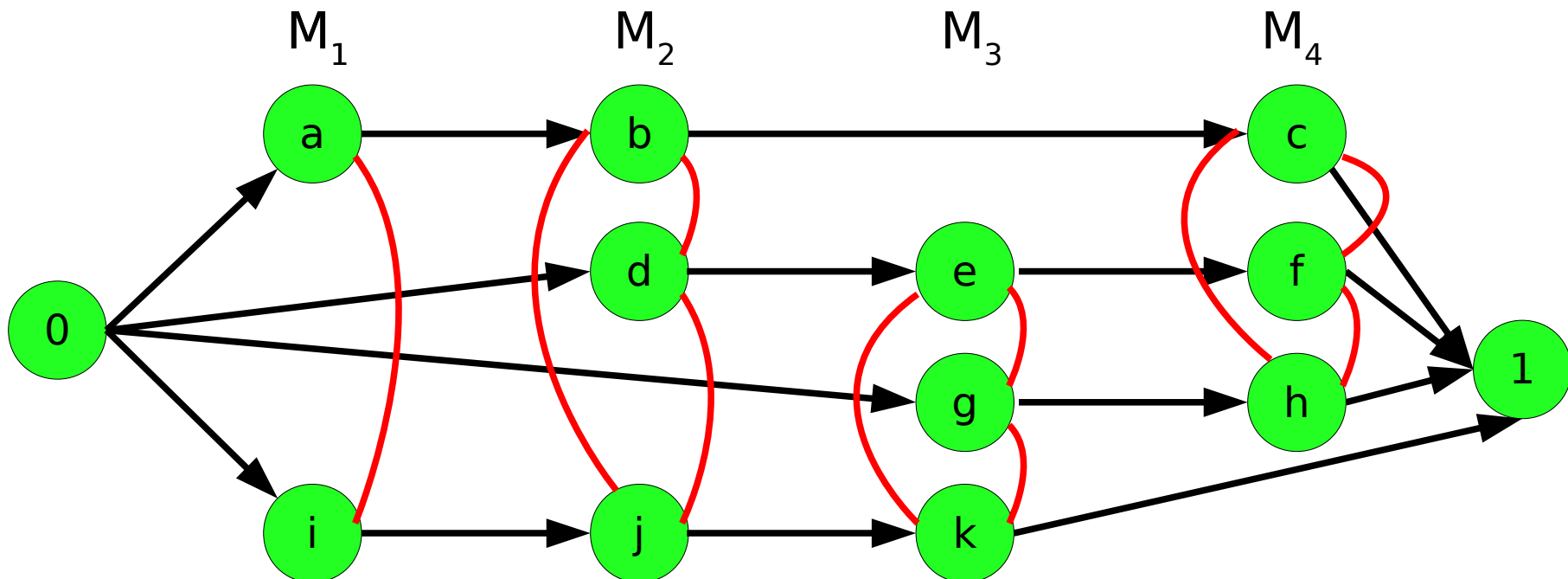


Représentation Jobshop graphe disjonctif

- $M_1 : J_1 J_4$
- $M_2 : J_2 J_1 J_4$
- $M_3 : J_3 J_2 J_4$
- $M_4 : J_3 J_2 J_1$

**Contraintes
d'exclusivité
Des machines**

 **cliques**



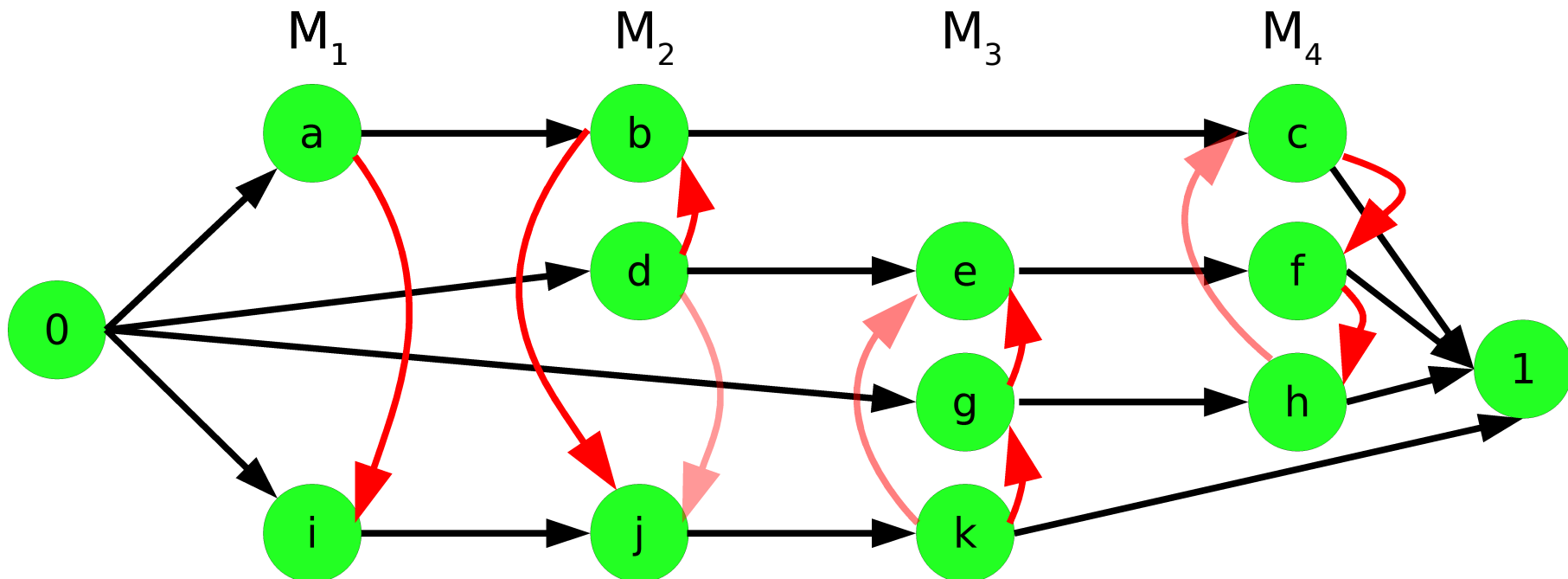
Représentation Jobshop graphe disjonctif

- Une solution:

- $M_1 : J_1 J_4$
- $M_2 : J_2 J_1 J_4$
- $M_3 : J_3 J_2 J_4$
- $M_4 : J_3 J_2 J_1$

**Choisir une direction
Pour les arêtes**

PAS DE CYCLE

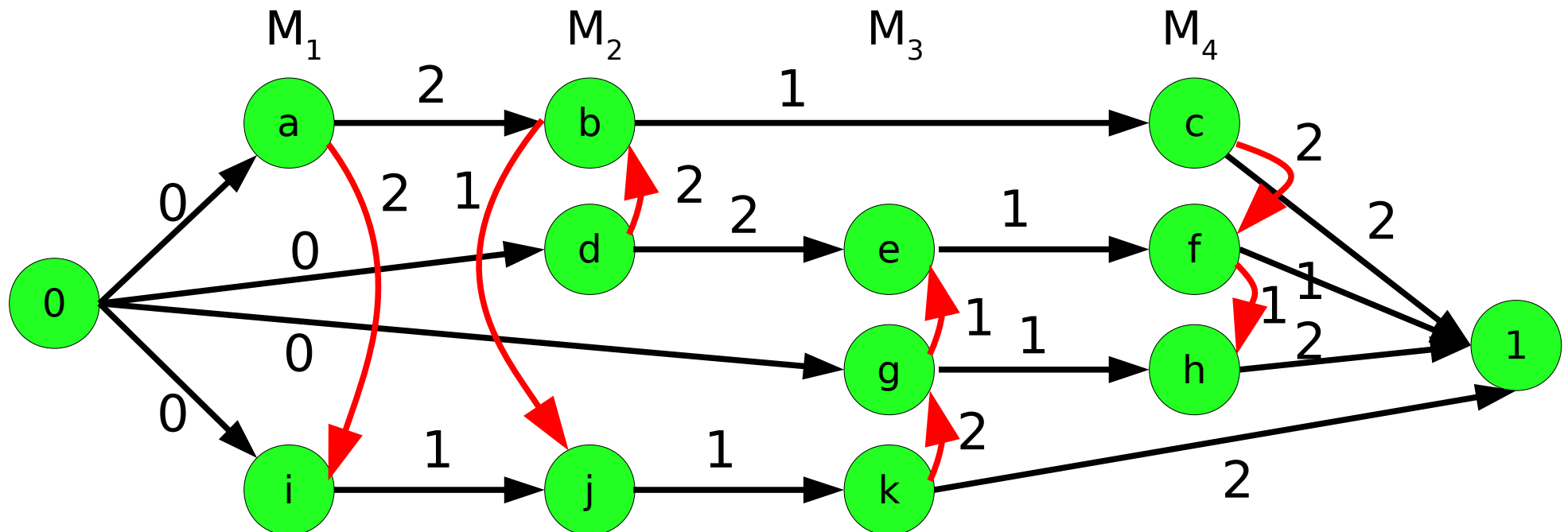


Représentation Jobshop graphe disjonctif

- $J_1 = \{a=(M_1,2), b=(M_2, 1), c=(M_4, 2)\}$
- $J_2 = \{d=(M_2,2), e=(M_3, 1), f=(M_4, 1)\}$
- $J_3 = \{g=(M_3, 1), h=(M_4, 2)\}$
- $J_4 = \{i=(M_1,1), j=(M_2, 1), k=(M_3, 2)\}$

**Appliquer les
algorithmes de
calcul des dates**

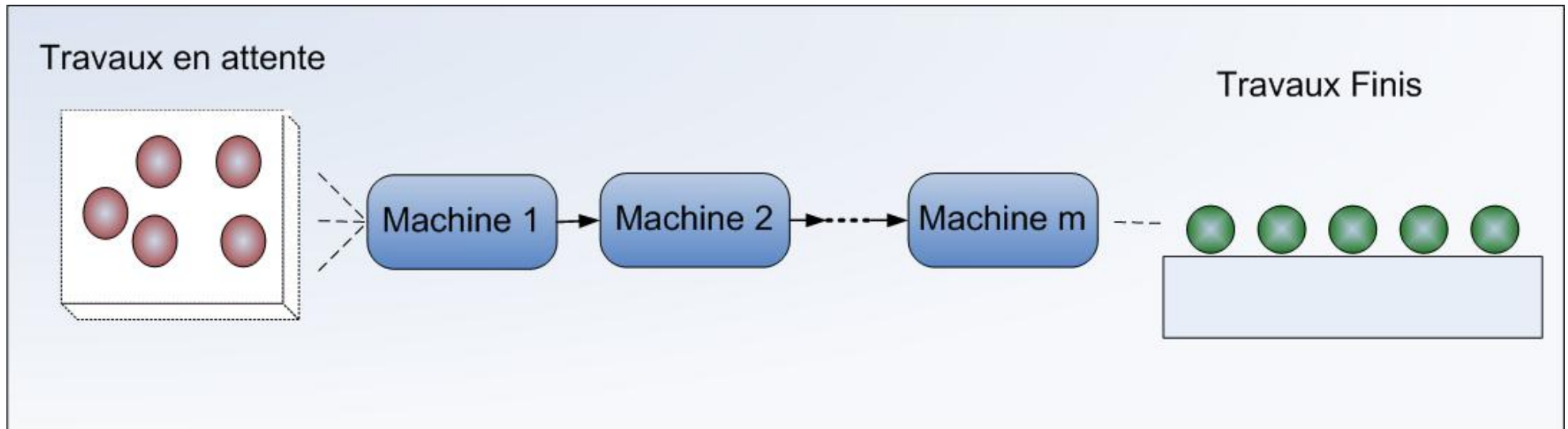
**Jobshop: choix des
arêtes qui minimise
la date finale**



Flowshop 2 machines $F_2 || C_{\max}$

algorithme de Johnson

- Algorithme Jobshop (*bottleneck shifting*) trop complexe
- Algorithme pour Flowshop à 2 machines



- Chaque Job J_i a un temps d'exécution $T_{i1} + T_{i2}$
- Trouver dans quel ordre les jobs entrent sur M_1

Résolution Flowshop algorithme de Johnson

- Intuitivement
 - Les plus courts (T_{i1}) sur la machine M_1 d'abord :
ils passeront plus vite sur M_2 en libérant M_1 pour les autres
 - Les plus courts (T_{i2}) sur la machine M_2 d'abord :
ils termineront plus vite sur M_2
- Remplir la file d'attente par les 2 cotés à la fois
- Algorithme : boucle dans que il y a des jobs à placer :
 - Calculer le min T_{ij}
 - Si $j = 1$, rajouter J_i à la file en partant du début
 - Si $j = 2$, rajouter J_i à la file en partant de la fin

Algorithme de Johnson exemple

- Algorithme : boucle dans que il y a des jobs à placer :
 - Calculer le min T_{ij}
 - Si $j = 1$, rajouter J_i à la file en partant du début
 - Si $j = 2$, rajouter J_i à la file en partant de la fin

J_i	T_{i1}	T_{i2}
1	4	3
2	1	2
3	5	4
4	2	3
5	5	6

étape	Job choisi	File
-------	---------------	------

Algorithme de Johnson exemple

- Algorithme : boucle dans que il y a des jobs à placer :
 - Calculer le min T_{ij}
 - Si $j = 1$, rajouter J_i à la file en partant du début
 - Si $j = 2$, rajouter J_i à la file en partant de la fin

J_i	T_{i1}	T_{i2}
1	4	3
2	1	2
3	5	4
4	2	3
5	5	6

étape	Job choisi	File
1	2	2

Algorithme de Johnson exemple

- Algorithme : boucle dans que il y a des jobs à placer :
 - Calculer le min T_{ij}
 - Si $j = 1$, rajouter J_i à la file en partant du début
 - Si $j = 2$, rajouter J_i à la file en partant de la fin

J_i	T_{i1}	T_{i2}
1	4	3
3	5	4
4	2	3
5	5	6

étape	Job choisi	File
1	2	2
2	4	2 4

Algorithme de Johnson exemple

- Algorithme : boucle dans que il y a des jobs à placer :
 - Calculer le min T_{ij}
 - Si $j = 1$, rajouter J_i à la file en partant du début
 - Si $j = 2$, rajouter J_i à la file en partant de la fin

J_i	T_{i1}	T_{i2}
1	4	3
3	5	4
5	5	6

étape	Job choisi	File
1	2	2
2	4	2 4
3	1	2 4 1

Algorithme de Johnson exemple

- Algorithme : boucle dans que il y a des jobs à placer :
 - Calculer le min T_{ij}
 - Si $j = 1$, rajouter J_i à la file en partant du début
 - Si $j = 2$, rajouter J_i à la file en partant de la fin

J_i	T_{i1}	T_{i2}
3	5	4
5	5	6

étape	Job choisi	File
1	2	2
2	4	2 4
3	1	2 4 1
4	3	2 4 3 1

Algorithme de Johnson exemple

- Algorithme : boucle dans que il y a des jobs à placer :
 - Calculer le min T_{ij}
 - Si $j = 1$, rajouter J_i à la file en partant du début
 - Si $j = 2$, rajouter J_i à la file en partant de la fin

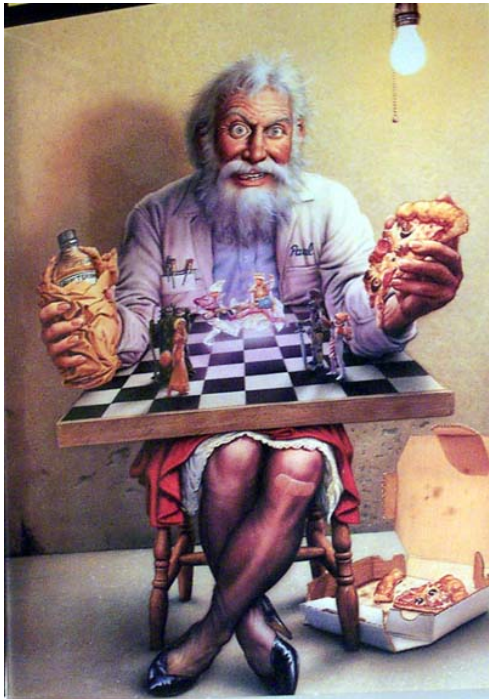
J_i	T_{i1}	T_{i2}
5	5	6

étape	Job choisi	File
1	2	2
2	4	2 4
3	1	2 4 1
4	3	2 4 3 1
5	5	2 4 5 3 1

Recherche dans les arbres de jeu

- Jeux visés :
 - Jeux à deux joueurs
 - Sans hasard (\neq dés)
 - Complètement informés (\neq cartes)

Jeux de stratégie (échecs, dames, othello...)



L.L. - Graphes

... à la main

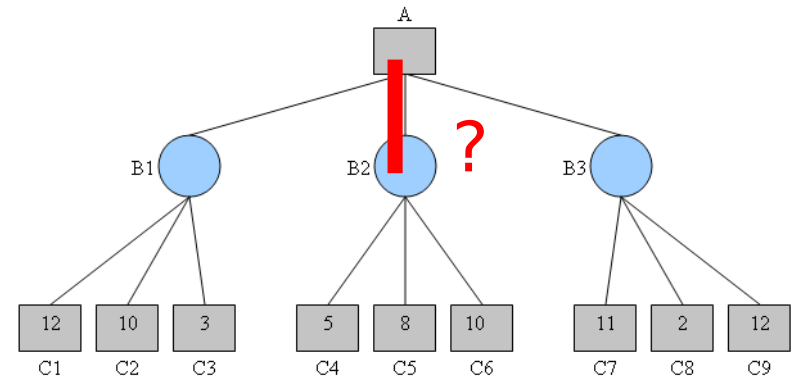
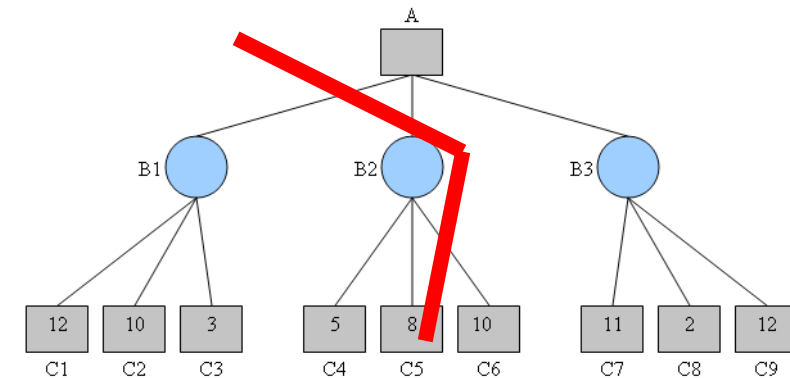
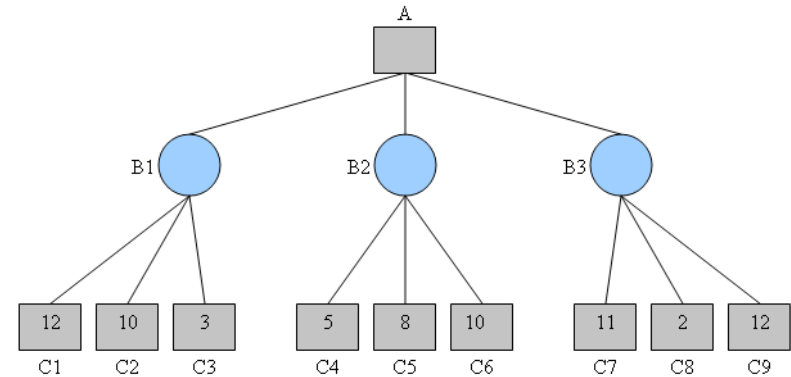
... par ordinateur

Deeper Blue bat Kasparov en
1997



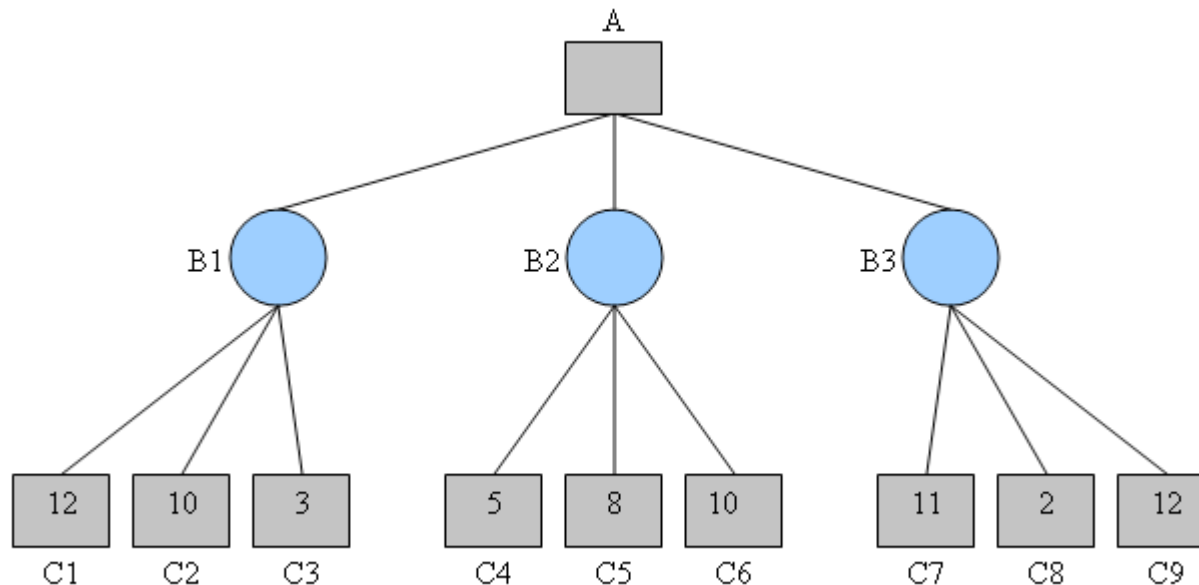
Recherche dans les arbres de jeu

- Représentation du problème
arbre de jeu
- Recherche d'une stratégie gagnante
 - recherche exhaustive
 - recherche heuristique
- Recherche du meilleur coup à jouer
 - algorithmes min-max
 - Alpha-beta
- **Profondeur de l'arbre ?**



Arbres de jeu

- Arbre tel que :
 - La racine = état initial du jeu
 - Un arc = un coup possible
 - Un nœud = un état possible du jeu
 - Une feuille = une fin de partie possible (victoire d'un des joueurs ou nul)



Exemple

Le jeu de Grundy

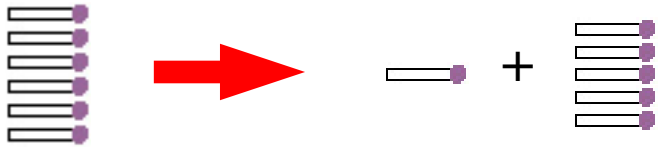
- Principe du jeu
On a une pile de n allumettes. Chaque joueur à son tour divise une pile en deux piles **non égales**. Le premier qui ne peut plus jouer a perdu.



Exemple

Le jeu de Grundy

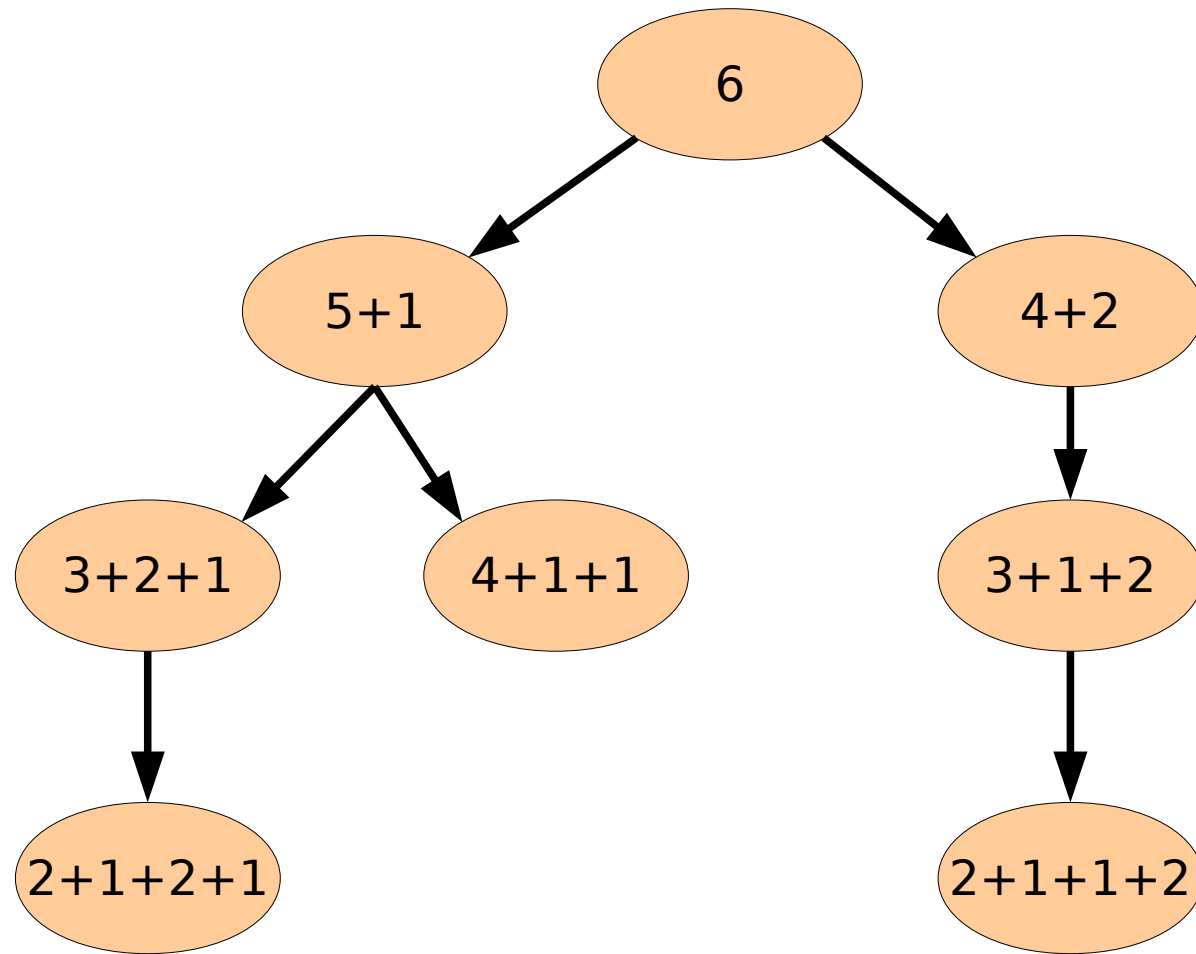
- Principe du jeu
On a une pile de n allumettes. Chaque joueur à son tour divise une pile en deux piles **non égales**.
Le premier qui ne peut plus jouer a perdu.



Construire l'arbre du jeu ...

Le jeu de Grundy

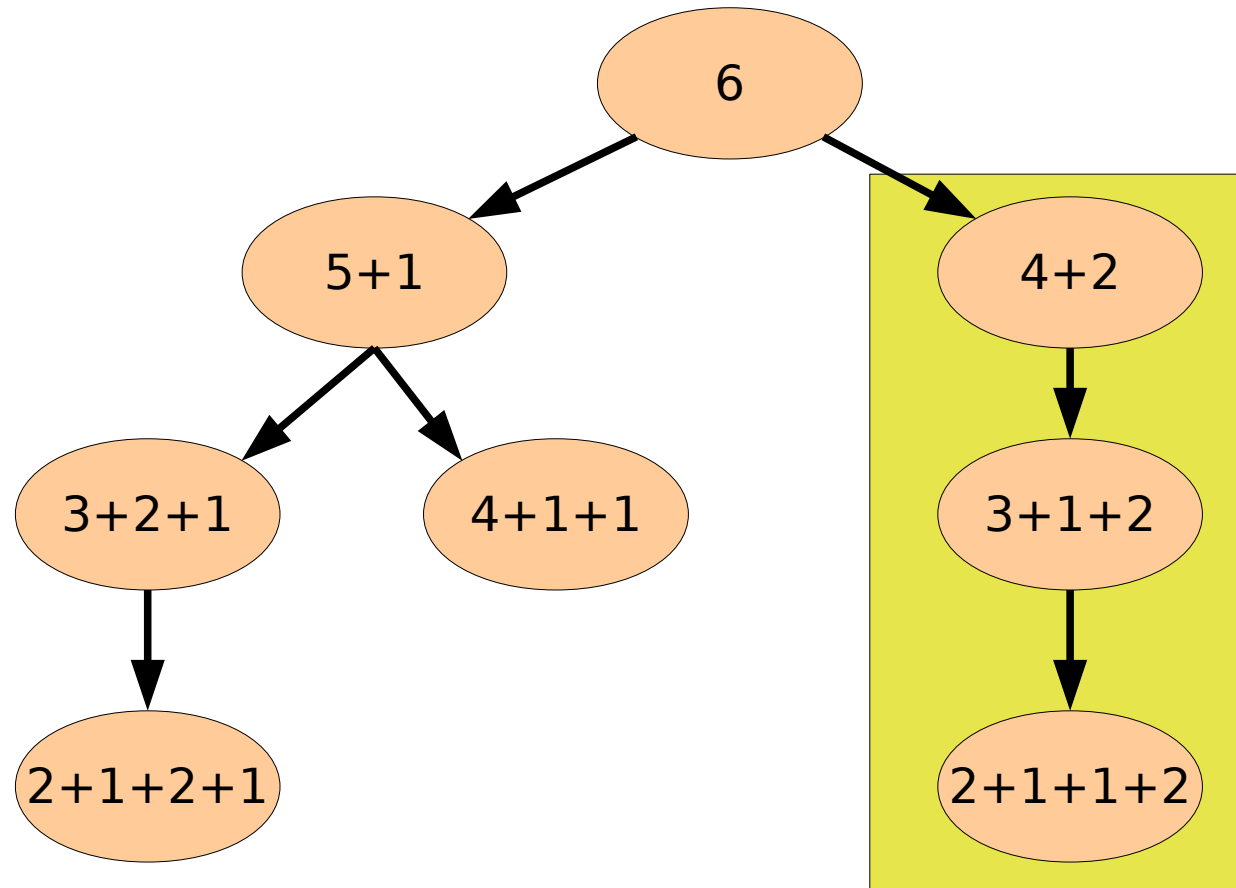
- Arbre du jeu pour 6 allumettes



Qui gagne quand ?

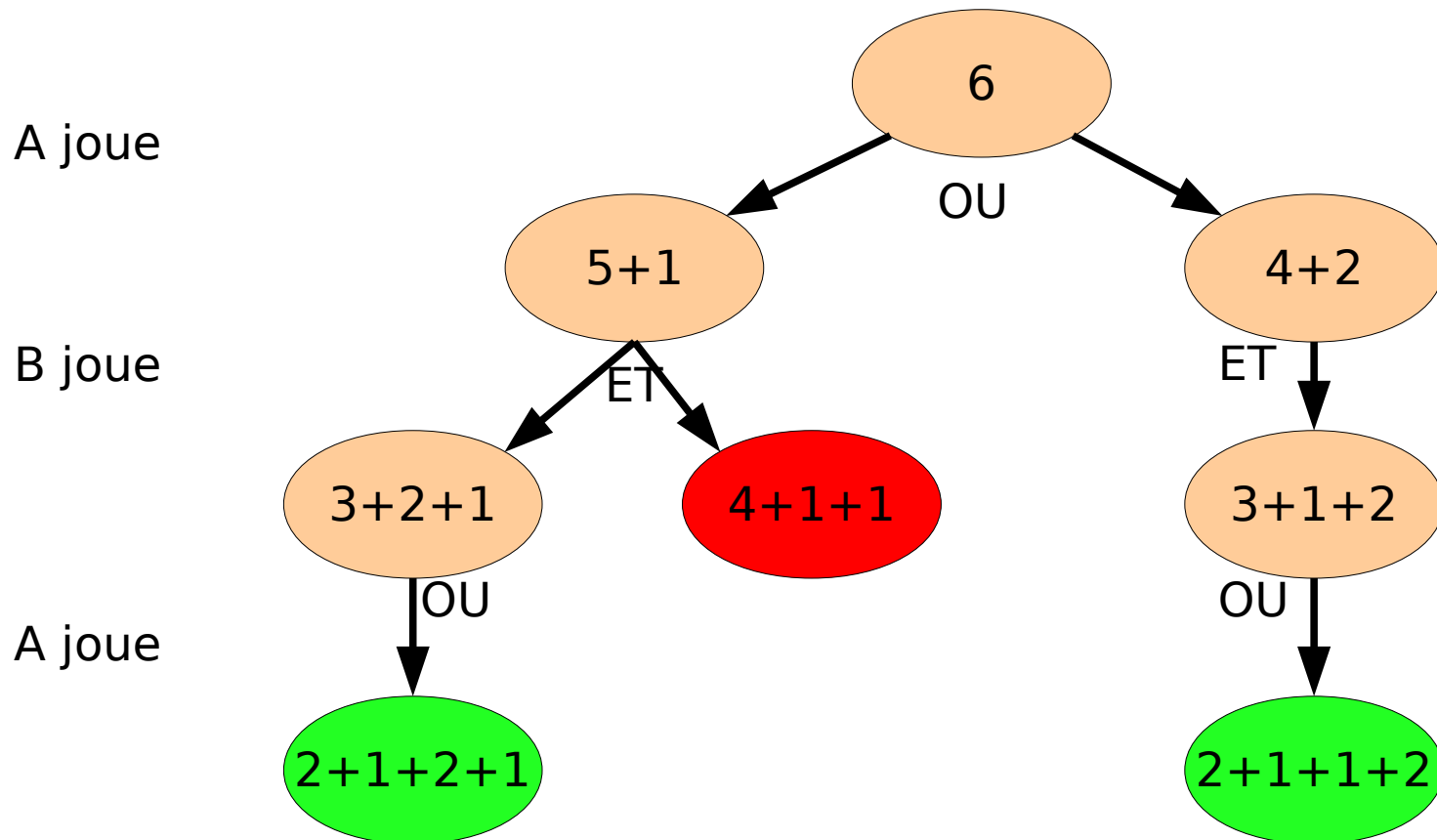
Stratégie de recherche

- Du point de vue d'un des 2 joueurs, trouver une stratégie qui permet de gagner à coup sûr.
 - Arbre ET/OU
 - Stratégie : Sous-graphe gagnant dans l'arbre ET/OU



Arbre ET/OU

- Du point de vue du joueur A, il suffit qu'un coup mène à la victoire si c'est à lui de jouer : il le choisira
- Noeud OU



Arbre ET/OU

- Du point de vue du joueur A, il faut que tous les coups mènent à la victoire quand c'est à B de jouer.
Noeud ET

