COMPLEXITÉ ET APPROXIMATION EN PRÉSENCE DE COMMUNICATIONS HIÉRARCHIQUES

Rodolphe Giroudeau

rgirou@lirmm.fr

(LIRMM, Université de Montpellier II)

Plan de l'exposé

- Motivations, modélisation et formulation

Plan de l'exposé

- Motivations, modélisation et formulation
- Résultats en présence d'une infinité de modules

Plan de l'exposé

- Motivations, modélisation et formulation
- Résultats en présence d'une infinité de modules
  - Résultat de non-approximabilité

Plan de l'exposé

- Motivations, modélisation et formulation
- Résultats en présence d'une infinité de modules
  - Résultat de non-approximabilité
  - Algorithmes approchés

Plan de l'exposé

- Motivations, modélisation et formulation
- Résultats en présence d'une infinité de modules
  - Résultat de non-approximabilité
  - Algorithmes approchés
- Résultats en présence d'un nombre fini de modules

Plan de l'exposé

- Motivations, modélisation et formulation
- Résultats en présence d'une infinité de modules
  - Résultat de non-approximabilité
  - Algorithmes approchés
- Résultats en présence d'un nombre fini de modules
  - Résultat de non-approximabilité

Plan de l'exposé

- Motivations, modélisation et formulation
- Résultats en présence d'une infinité de modules
  - Résultat de non-approximabilité
  - Algorithmes approchés
- Résultats en présence d'un nombre fini de modules
  - Résultat de non-approximabilité
  - Algorithmes approchés

Plan de l'exposé

- Motivations, modélisation et formulation
- Résultats en présence d'une infinité de modules
  - Résultat de non-approximabilité
  - Algorithmes approchés
- Résultats en présence d'un nombre fini de modules
  - Résultat de non-approximabilité
  - Algorithmes approchés
- Algorithme polynomial en présence de la duplication de tâches

Plan de l'exposé

- Motivations, modélisation et formulation
- Résultats en présence d'une infinité de modules
  - Résultat de non-approximabilité
  - Algorithmes approchés
- Résultats en présence d'un nombre fini de modules
  - Résultat de non-approximabilité
  - Algorithmes approchés
- Algorithme polynomial en présence de la duplication de tâches
- Conclusions et perspectives

Introduction

- L'application est modélisée par un graphe de précédence

Introduction

- L'application est modélisée par un graphe de précédence
- PRAM, granularité grossière, temps de communications négligées

Introduction

- L'application est modélisée par un graphe de précédence
- PRAM, granularité grossière, temps de communications négligées
- Prise en compte des délais de communications

Introduction

- L'application est modélisée par un graphe de précédence
- PRAM, granularité grossière, temps de communications négligées
- Prise en compte des délais de communications
- Exécution en parallèle sur des grappes de stations de travail

Modélisation

Existence de plusieurs modèles qui prennent en compte les communications

Le plus populaire : le modèle à communications homogènes

Modélisation

Existence de plusieurs modèles qui prennent en compte les communications

Le plus populaire : le modèle à communications homogènes

- Avantages :

Modélisation

Existence de plusieurs modèles qui prennent en compte les communications

Le plus populaire : le modèle à communications homogènes

- Avantages :
- Abstraction de la topologie de la machine parallèle

Modélisation

Existence de plusieurs modèles qui prennent en compte les communications

Le plus populaire : le modèle à communications homogènes

- Avantages :
- Abstraction de la topologie de la machine parallèle
- Simplicité du modèle

Modélisation

Existence de plusieurs modèles qui prennent en compte les communications

Le plus populaire : le modèle à communications homogènes

- Avantages :
- Abstraction de la topologie de la machine parallèle
- Simplicité du modèle
- Non dédié à une machine particulière

Formulation

Modèle homogène

$G = (V, E)$

Formulation

Modèle homogène

$G = (V, E)$

$t_i =$ date de début d'exécution de la tâche i

Formulation

Modèle homogène

$G = (V, E)$

$t_i =$ date de début d'exécution de la tâche i

$\pi_i =$ processeur sur lequel s'exécute i

Formulation

Modèle homogène

$G = (V, E)$
$t_i =$ date de début d'exécution de la tâche i
$\pi_i =$ processeur sur lequel s'exécute i
$p_i =$ durée d'exécution de i

Formulation

Modèle homogène

$G = (V, E)$

$t_i =$ date de début d'exécution de la tâche i

$\pi_i =$ processeur sur lequel s'exécute i

$p_i =$ durée d'exécution de i

Un ordonnancement réalisable :

Formulation

Modèle homogène

$G = (V, E)$

$t_i =$ date de début d'exécution de la tâche i

$\pi_i =$ processeur sur lequel s'exécute i

$p_i =$ durée d'exécution de i

Un ordonnancement réalisable :

$\forall (i, j) \in E$ nous associons une valeur c_{ij}

Formulation

Modèle homogène

$G = (V, E)$

$t_i =$ date de début d'exécution de la tâche i

$\pi_i =$ processeur sur lequel s'exécute i

$p_i =$ durée d'exécution de i

Un ordonnancement réalisable :

$\forall (i, j) \in E$ nous associons une valeur c_{ij}

- si $\pi_i = \pi_j$ alors $t_i + p_i \leq t_j$

Formulation

Modèle homogène

$G = (V, E)$
$t_i =$ date de début d'exécution de la tâche i
$\pi_i =$ processeur sur lequel s'exécute i
$p_i =$ durée d'exécution de i
Un ordonnancement réalisable :
$\forall (i, j) \in E$ nous associons une valeur c_{ij}

- si $\pi_i = \pi_j$ alors $t_i + p_i \leq t_j$
- sinon $t_i + p_i + c_{ij} \leq t_j$

Formulation

Modèle homogène

$G = (V, E)$
$t_i =$ date de début d'exécution de la tâche i
$\pi_i =$ processeur sur lequel s'exécute i
$p_i =$ durée d'exécution de i
Un ordonnancement réalisable :
$\forall (i, j) \in E$ nous associons une valeur c_{ij}

- si $\pi_i = \pi_j$ alors $t_i + p_i \leq t_j$
- sinon $t_i + p_i + c_{ij} \leq t_j$

Objectif : minimiser la date de fin d'exécution des tâches notée

C_m^{ax}

Notre approche

Étudier les problèmes d'ordonnancement des tâches d'une application dans une machine parallèle formée d'un ensemble de grappes de processeurs

Notre approche

Étudier les problèmes d'ordonnancement des tâches d'une application dans une machine parallèle formée d'un ensemble de grappes de processeurs

Nécessité de prendre en compte la notion de communications hiérarchiques

Notre approche

Étudier les problèmes d'ordonnancement des tâches d'une application dans une machine parallèle formée d'un ensemble de grappes de processeurs

Nécessité de prendre en compte la notion de communications hiérarchiques

Communications :

Notre approche

Étudier les problèmes d'ordonnancement des tâches d'une application dans une machine parallèle formée d'un ensemble de grappes de processeurs

Nécessité de prendre en compte la notion de communications hiérarchiques

Communications :

- intra-modulaires (notées ϵ_{ij})

Notre approche

Étudier les problèmes d'ordonnancement des tâches d'une application dans une machine parallèle formée d'un ensemble de grappes de processeurs

Nécessité de prendre en compte la notion de communications hiérarchiques

Communications :

- intra-modulaires (notées ϵ_{ij})
- extra-modulaires (notées c_{ij})

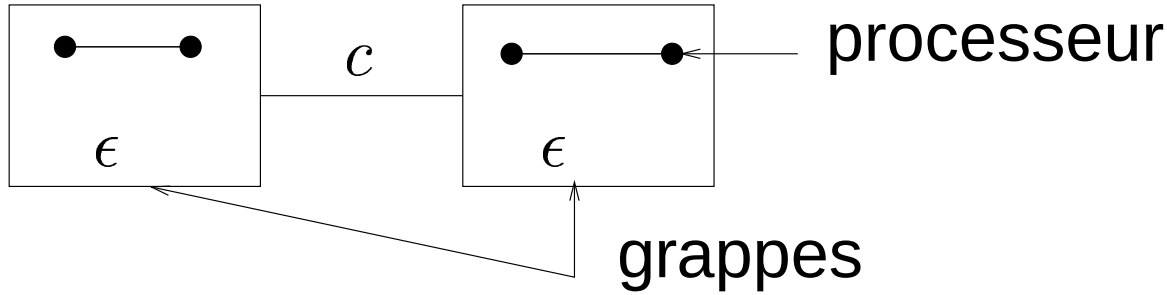
Notre approche

Étudier les problèmes d'ordonnancement des tâches d'une application dans une machine parallèle formée d'un ensemble de grappes de processeurs

Nécessité de prendre en compte la notion de communications hiérarchiques

Communications :

- intra-modulaires (notées ϵ_{ij})
- extra-modulaires (notées c_{ij})



Formulation

Modèle hiérarchique

$G = (V, E)$

Formulation

Modèle hiérarchique

$G = (V, E)$

$t_i =$ date de début d'exécution de la tâche i

Formulation

Modèle hiérarchique

$G = (V, E)$

$t_i =$ date de début d'exécution de la tâche i

$(\Pi^i, \pi^i_k) =$ module et processeur sur lequel s'exécute i

Formulation

Modèle hiérarchique

$G = (V, E)$
$t_i =$ date de début d'exécution de la tâche i
$(\Pi^i, \pi^i_k) =$ module et processeur sur lequel s'exécute i
$p_i =$ durée d'exécution de i

Formulation

Modèle hiérarchique

$G = (V, E)$

$t_i =$ date de début d'exécution de la tâche i

$(\Pi^i, \pi^i_k) =$ module et processeur sur lequel s'exécute i

$p_i =$ durée d'exécution de i

Un ordonnancement réalisable :

Formulation

Modèle hiérarchique

$G = (V, E)$

$t_i =$ date de début d'exécution de la tâche i

$(\Pi^i, \pi^i_k) =$ module et processeur sur lequel s'exécute i

$p_i =$ durée d'exécution de i

Un ordonnancement réalisable :

$\forall (i, j) \in E$ nous associons un couple de valeurs (ϵ_{ij}, c_{ij}) avec $\epsilon_{ij} \leq c_{ij}$

Formulation

Modèle hiérarchique

$G = (V, E)$
$t_i =$ date de début d'exécution de la tâche i
$(\Pi^i, \pi^i_k) =$ module et processeur sur lequel s'exécute i
$p_i =$ durée d'exécution de i
Un ordonnancement réalisable :
$\forall (i,j) \in E$ nous associons un couple de valeurs (ϵ_{ij}, c_{ij}) avec $\epsilon_{ij} \leq c_{ij}$

- $\Pi^i = \Pi^j$ et si $\pi^i_k = \pi^j_k$ alors $t_i + p_i \leq t_j$

Formulation

Modèle hiérarchique

$G = (V, E)$
$t_i =$ date de début d'exécution de la tâche i
$(\Pi^i, \pi^i_k) =$ module et processeur sur lequel s'exécute i
$p_i =$ durée d'exécution de i
Un ordonnancement réalisable :
$\forall (i,j) \in E$ nous associons un couple de valeurs (ϵ_{ij}, c_{ij}) avec $\epsilon_{ij} \leq c_{ij}$

- $\Pi^i = \Pi^j$ et si $\pi^i_k = \pi^j_k$ alors $t_i + p_i \leq t_j$
- sinon si $\Pi^i = \Pi^j$ et si $\pi^i_k = \pi^j_{k'}$ avec $k \neq k'$ alors $t_i + p_i + \epsilon_{ij} \leq t_j$

Formulation

Modèle hiérarchique

$G = (V, E)$
$t_i =$ date de début d'exécution de la tâche i
$(\Pi^i, \pi^i_k) =$ module et processeur sur lequel s'exécute i
$p_i =$ durée d'exécution de i
Un ordonnancement réalisable :
$\forall (i,j) \in E$ nous associons un couple de valeurs (ϵ_{ij}, c_{ij}) avec $\epsilon_{ij} \leq c_{ij}$

- $\Pi^i = \Pi^j$ et si $\pi^i_k = \pi^j_k$ alors $t_i + p_i \leq t_j$
- sinon si $\Pi^i = \Pi^j$ et si $\pi^i_k = \pi^j_{k'}$ avec $k \neq k'$ alors $t_i + p_i + \epsilon_{ij} \leq t_j$
- sinon $\Pi^i \neq \Pi^j$ alors $t_i + p_i + c_{ij} \leq t_j$

Configurations possibles

- Problèmes hiérarchiques

Configurations possibles

- Problèmes hiérarchiques
- nombre de machine par clusters $2, m$

Configurations possibles

- Problèmes hiérarchiques
- nombre de machine par clusters $2, m$
- nombre de clusters ∞, M

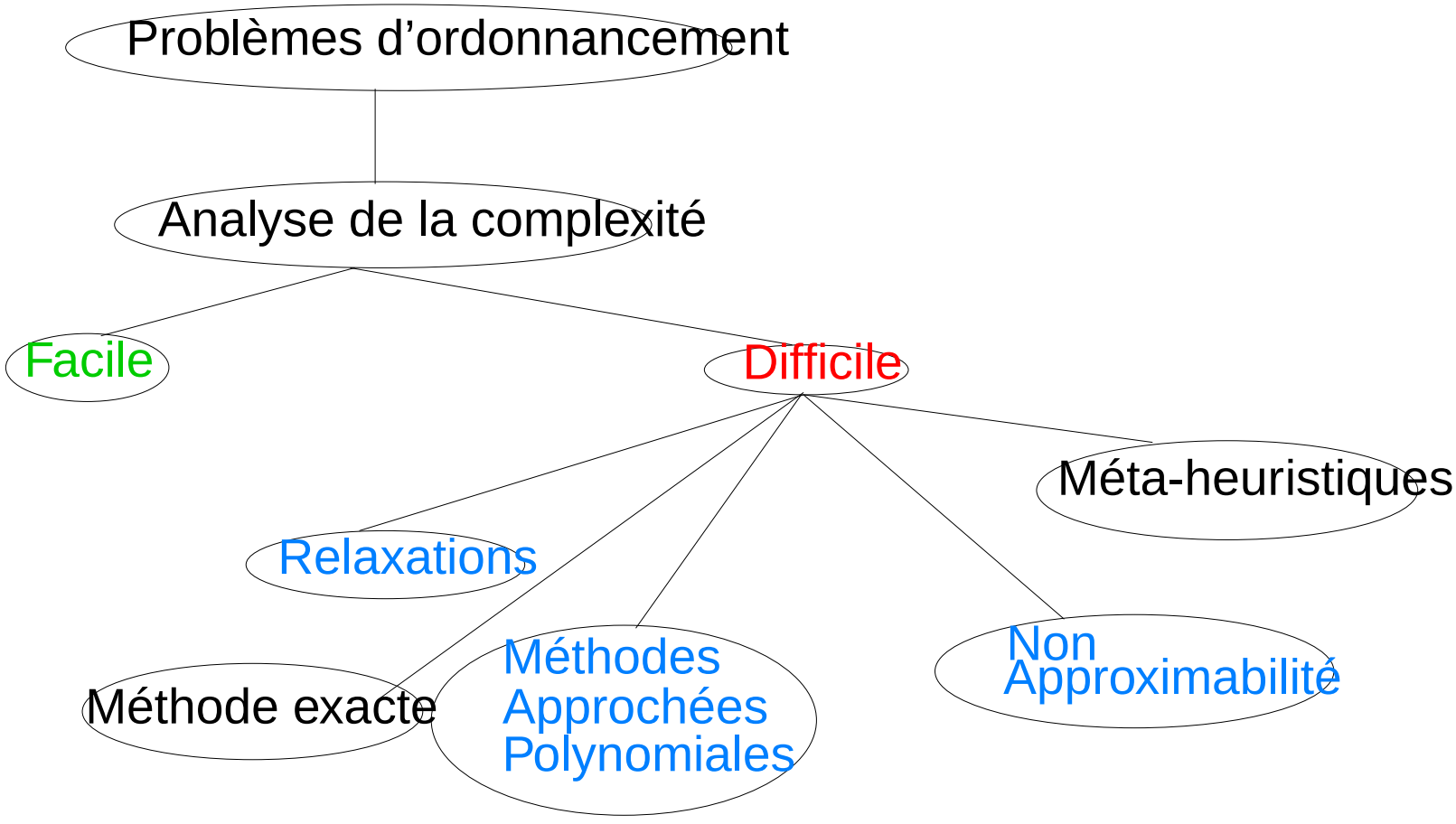
Configurations possibles

- Problèmes hiérarchiques
- nombre de machine par clusters $2, m$
- nombre de clusters ∞, M
- valeurs de $\epsilon,\ c \in \{1, 2, \ldots, \}$

Configurations possibles

- Problèmes hiérarchiques
  - nombre de machine par clusters $2, m$
  - nombre de clusters ∞, M
  - valeurs de $\epsilon,\ c \in \{1, 2, \ldots, \}$
  - processeurs homogènes et hétérogènes

Méthodologie



Mesure de la qualité

Utilisation de la notion de performance relative pour une heuristique h

$$\rho^h = \max_I \frac{C^h_{max}(I)}{C^{opt}_{max}(I)}.$$

Notre objectif :

Déterminer

Notre objectif :

Déterminer

- des bornes supérieures (résultats positifs)

Notre objectif :

Déterminer

- des bornes supérieures (résultats positifs)
- des bornes inférieures (résultats négatifs)

Notre objectif :

Déterminer

- des bornes supérieures (résultats positifs)
- des bornes inférieures (résultats négatifs)
- Regarder les techniques qui résistent au passage homogène $\Longleftrightarrow$ hiérarchique

Techniques employées :

- Programmation dynamique

Techniques employées :

- Programmation dynamique
- Programmation linéaire (en nombres entiers)

Techniques employées :

- Programmation dynamique
- Programmation linéaire (en nombres entiers)
- Réduction polynomiale

Techniques employées :

- Programmation dynamique
- Programmation linéaire (en nombres entiers)
- Réduction polynomiale
- Algorithmes polynomiaux

Techniques employées :

- Programmation dynamique
- Programmation linéaire (en nombres entiers)
- Réduction polynomiale
- Algorithmes polynomiaux
- Techniques de pliage pour le passage d'une infinité de machines à un nombre limité

Techniques employées :

- Programmation dynamique
- Programmation linéaire (en nombres entiers)
- Réduction polynomiale
- Algorithmes polynomiaux
- Techniques de pliage pour le passage d'une infinité de machines à un nombre limité
- . . .

Résultats négatifs (1)

Résultats négatifs (1)

Modèle classique

Résultats négatifs (1)

Modèle classique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$

Résultats négatifs (1)

Modèle classique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$
- Nombre de processeurs suffisant

Résultats négatifs (1)

Modèle classique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$
- Nombre de processeurs suffisant

$\nexists$ ρ-approximation avec $\rho < \frac{7}{6}$, sauf si $\mathcal{P} = \mathcal{NP}$

[Hoogeveen,Lenstra,Veltman 94]

Résultats négatifs (1)

Modèle classique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$
- Nombre de processeurs suffisant

$\nexists$ ρ-approximation avec $\rho < \frac{7}{6}$, sauf si $\mathcal{P} = \mathcal{NP}$

[Hoogeveen,Lenstra,Veltman 94]

Modèle hiérarchique

Résultats négatifs (1)

Modèle classique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$
- Nombre de processeurs suffisant

$\nexists$ ρ-approximation avec $\rho < \frac{7}{6}$, sauf si $\mathcal{P} = \mathcal{NP}$

[Hoogeveen,Lenstra,Veltman 94]

Modèle hiérarchique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$ et $\epsilon_{ij} = 0$

Résultats négatifs (1)

Modèle classique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$
- Nombre de processeurs suffisant

$\nexists$ ρ-approximation avec $\rho < \frac{7}{6}$, sauf si $\mathcal{P} = \mathcal{NP}$

[Hoogeveen,Lenstra,Veltman 94]

Modèle hiérarchique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$ et $\epsilon_{ij} = 0$
- Nombre de modules suffisant, 2 processeurs identiques par module

Résultats négatifs (1)

Modèle classique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$
- Nombre de processeurs suffisant

$\nexists$ ρ-approximation avec $\rho < \frac{7}{6}$, sauf si $\mathcal{P} = \mathcal{NP}$

[Hoogeveen,Lenstra,Veltman 94]

Modèle hiérarchique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$ et $\epsilon_{ij} = 0$
- Nombre de modules suffisant, 2 processeurs identiques par module

$\nexists$ ρ-approximation avec $\rho < \frac{5}{4}$, sauf si $\mathcal{P} = \mathcal{NP}$

[Bampis,Giroudeau,König 99]

Résultats positif (1)

Résultats positif (1)

Modèle classique

Résultats positif (1)

Modèle classique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$

Résultats positif (1)

Modèle classique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$
- Nombre de processeurs suffisant

Résultats positif (1)

Modèle classique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$
- Nombre de processeurs suffisant

algorithme $\frac{4}{3}$-approché [Munier,König 97]

Résultats positif (1)

Modèle classique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$
- Nombre de processeurs suffisant

algorithme $\frac{4}{3}$-approché [Munier,König 97]

Modèle hiérarchique

Résultats positif (1)

Modèle classique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$
- Nombre de processeurs suffisant

algorithme $\frac{4}{3}$-approché [Munier,König 97]

Modèle hiérarchique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$ et $\epsilon_{ij} = 0$

Résultats positif (1)

Modèle classique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$
- Nombre de processeurs suffisant

algorithme $\frac{4}{3}$-approché [Munier,König 97]

Modèle hiérarchique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$ et $\epsilon_{ij} = 0$
- Nombre de modules suffisant, 2 processeurs identiques par module

Résultats positif (1)

Modèle classique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$
- Nombre de processeurs suffisant

algorithme $\frac{4}{3}$-approché [Munier,König 97]

Modèle hiérarchique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$ et $\epsilon_{ij} = 0$
- Nombre de modules suffisant, 2 processeurs identiques par module

algorithme $\frac{8}{5}$-approché [Bampis,Giroudeau,König 99]

Algorithme

- Formulation du problème par un programme linéaire en nombre entiers

Algorithme

- Formulation du problème par un programme linéaire en nombre entiers
- Relaxation des contraintes $\rightarrow$ programme linéaire

Algorithme

- Formulation du problème par un programme linéaire en nombre entiers
- Relaxation des contraintes $\rightarrow$ programme linéaire
- Résolution du programme linéaire $\rightarrow B_{inf}$ pour C_{max}

Algorithme

- Formulation du problème par un programme linéaire en nombre entiers
- Relaxation des contraintes $\rightarrow$ programme linéaire
- Résolution du programme linéaire $\rightarrow B_{inf}$ pour C_{max}
- *Étape 1:* Étape d'arrondi sur les variables

Algorithme

- Formulation du problème par un programme linéaire en nombre entiers
- Relaxation des contraintes $\rightarrow$ programme linéaire
- Résolution du programme linéaire $\rightarrow B_{inf}$ pour C_{max}
- *Étape 1:* Étape d'arrondi sur les variables
- *Étape 2:* Étape d'arrondi réalisable pour obtenir un ordonnancement réalisable

Formulation

- $\min C_m^{ax}$

Formulation

- $\min C_m{}^{ax}$
- $\forall (i,j) \in E, x_{ij} \in \{0,1\}$

Formulation

- $\min C_m{}^{ax}$
- $\forall (i,j) \in E, x_{ij} \in \{0,1\}$
- $\forall i \in V, t_i \geq 0$

Formulation

- $\min {C_m}^{ax}$
- $\forall (i,j) \in E, x_{ij} \in \{0,1\}$
- $\forall i \in V, t_i \geq 0$
- $\forall (i,j) \in E, t_i + 1 + x_{ij} \leq t_j$

Formulation

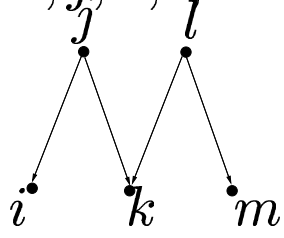
- $\min C_m{}^{ax}$
- $\forall (i,j) \in E, x_{ij} \in \{0,1\}$
- $\forall i \in V, t_i \geq 0$
- $\forall (i,j) \in E, t_i + 1 + x_{ij} \leq t_j$
- $\forall (i,j) \in E - U, \sum_{j \in \Gamma^+(i)} x_{ij} \geq |\Gamma^+(i)| - 2$

Formulation

- $\min {C_m}^{ax}$
- $\forall (i,j) \in E, x_{ij} \in \{0,1\}$
- $\forall i \in V, t_i \geq 0$
- $\forall (i,j) \in E, t_i + 1 + x_{ij} \leq t_j$
- $\forall (i,j) \in E - U, \sum_{j \in \Gamma^+(i)} x_{ij} \geq |\Gamma^+(i)| - 2$
- $\forall (i,j) \in E - Z, \sum_{j \in \Gamma^-(i)} x_{ji} \geq |\Gamma^-(i)| - 2$

Formulation

- $\min {C_m}^{ax}$
- $\forall (i,j) \in E, x_{ij} \in \{0,1\}$
- $\forall i \in V, t_i \geq 0$
- $\forall (i,j) \in E, t_i + 1 + x_{ij} \leq t_j$
- $\forall (i,j) \in E - U, \sum_{j \in \Gamma^+(i)} x_{ij} \geq |\Gamma^+(i)| - 2$
- $\forall (i,j) \in E - Z, \sum_{j \in \Gamma^-(i)} x_{ji} \geq |\Gamma^-(i)| - 2$
- $\forall i,j,k,l,m \in V, x_{ji} + x_{jk} + x_{lk} + x_{lm} \geq 1$ $(j,i),(j,k),(l,k),(l,m) \in E,$



Formulation

- $\min C_m^{ax}$
- $\forall (i,j) \in E, x_{ij} \in \{0,1\}$
- $\forall i \in V, t_i \geq 0$
- $\forall (i,j) \in E, t_i + 1 + x_{ij} \leq t_j$
- $\forall (i,j) \in E - U, \sum_{j \in \Gamma^+(i)} x_{ij} \geq |\Gamma^+(i)| - 2$
- $\forall (i,j) \in E - Z, \sum_{j \in \Gamma^-(i)} x_{ji} \geq |\Gamma^-(i)| - 2$
- $\forall i,j,k,l,m \in V, x_{ji} + x_{jk} + x_{lk} + x_{lm} \geq 1 \ (j,i),(j,k),(l,k),(l,m) \in E,$
- $\forall i,j,k,l,m \in V, x_{ij} + x_{kj} + x_{kl} + x_{ml} \geq 1 \ (i,j),(k,j),(k,l),(m,l) \in E,$

Formulation

- $\min C_m{}^{ax}$
- $\forall (i,j) \in E, x_{ij} \in \{0,1\}$
- $\forall i \in V, t_i \geq 0$
- $\forall (i,j) \in E, t_i + 1 + x_{ij} \leq t_j$
- $\forall (i,j) \in E - U, \sum_{j \in \Gamma^+(i)} x_{ij} \geq |\Gamma^+(i)| - 2$
- $\forall (i,j) \in E - Z, \sum_{j \in \Gamma^-(i)} x_{ji} \geq |\Gamma^-(i)| - 2$
- $\forall i,j,k,l,m \in V, x_{ji} + x_{jk} + x_{lk} + x_{lm} \geq 1 \ (j,i),(j,k),(l,k),(l,m) \in E,$
- $\forall i,j,k,l,m \in V, x_{ij} + x_{kj} + x_{kl} + x_{ml} \geq 1 \ (i,j),(k,j),(k,l),(m,l) \in E,$
- $\forall i \in V, t_i + 1 \leq C_{max}$

Formulation

- $\min {C_m}^{ax}$

$$\forall (i,j) \in E, x_{ij} \in [0,1]$$

- $\forall i \in V, t_i \geq 0$
- $\forall (i,j) \in E, t_i + 1 + x_{ij} \leq t_j$
- $\forall (i,j) \in E - U, \sum_{j \in \Gamma^+(i)} x_{ij} \geq |\Gamma^+(i)| - 2$
- $\forall (i,j) \in E - Z, \sum_{j \in \Gamma^-(i)} x_{ji} \geq |\Gamma^-(i)| - 2$
- $\forall i,j,k,l,m \in V, x_{ji} + x_{jk} + x_{lk} + x_{lm} \geq 1 \ (j,i),(j,k),(l,k),(l,m) \in E,$
- $\forall i,j,k,l,m \in V, x_{ij} + x_{kj} + x_{kl} + x_{ml} \geq 1 \ (i,j),(k,j),(k,l),(m,l) \in E,$
- $\forall i \in V, t_i + 1 \leq C_{max}$

Description

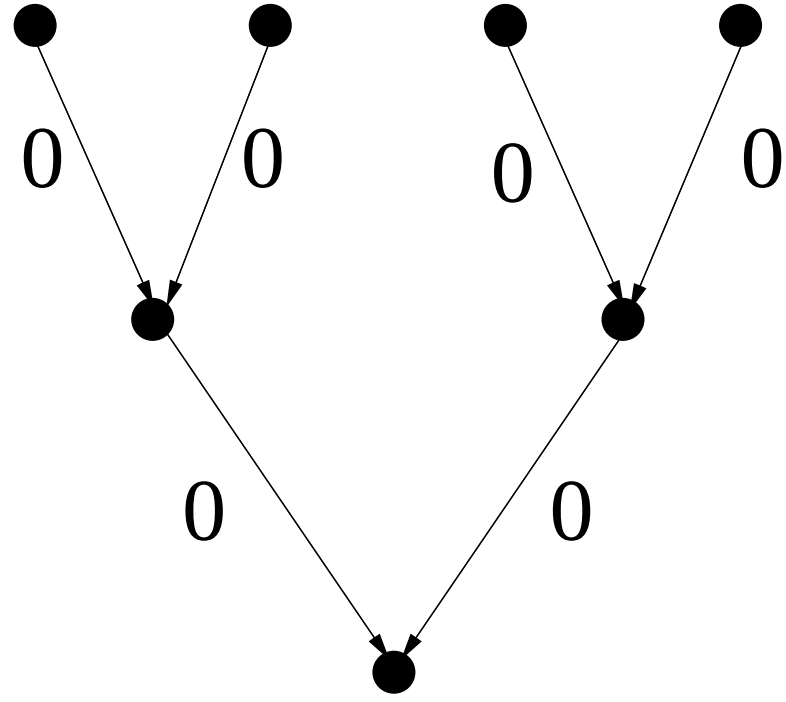
Tout ordonnancement doit satisfaire les contraintes du programme linéaire en nombre entiers

Un ordonnancement optimal issu du programme linéaire en nombre entiers n'implique pas forcément un ordonnancement réalisable

Description

Tout ordonnancement doit satisfaire les contraintes du programme linéaire en nombre entiers

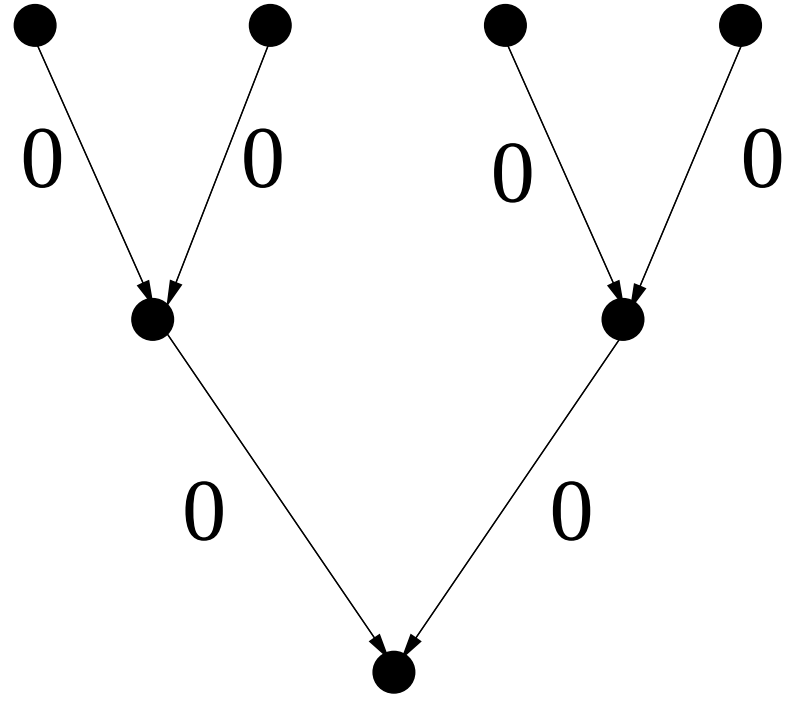
Un ordonnancement optimal issu du programme linéaire en nombre entiers n'implique pas forcément un ordonnancement réalisable



Description

Tout ordonnancement doit satisfaire les contraintes du programme linéaire en nombre entiers

Un ordonnancement optimal issu du programme linéaire en nombre entiers n'implique pas forcément un ordonnancement réalisable



Description

Lemme : Chaque tâche $i \in V$ possède au plus deux tâches successeurs (resp. prédécesseurs) tel que $e_{ij} < 0.25$ (resp. $e_{ji} < 0.25$).

Description

Lemme : Chaque tâche $i \in V$ possède au plus deux tâches successeurs (resp. prédécesseurs) tel que $e_{ij} < 0.25$ (resp. $e_{ji} < 0.25$).

Étape 1 [Arrondi]:
Si $e_{ij} < 0.25 \Rightarrow x_{ij} = 0$ (noté $0 - arc$) sinon $x_{ij} = 1$, $(1 - arc)$

Description

Lemme : Chaque tâche $i \in V$ possède au plus deux tâches successeurs (resp. prédécesseurs) tel que $e_{ij} < 0.25$ (resp. $e_{ji} < 0.25$).

Étape 1 [Arrondi]:
Si $e_{ij} < 0.25 \Rightarrow x_{ij} = 0$ (noté $0 - arc$) sinon $x_{ij} = 1$, $(1 - arc)$
Nécessité d'introduire une autre étape pour garantir un ordonnancement réalisable

Description

Arrondi réalisable : Soit A_i, l'ensemble des arcs qui sont éléments d'un chemin critique ayant pour sommet terminal la tâche i.

Description

Arrondi réalisable : Soit A_i, l'ensemble des arcs qui sont éléments d'un chemin critique ayant pour sommet terminal la tâche i.

- Si $\Gamma^-(i) = \varnothing$, nous gardons inchangé les valeurs des variables x_{ij} obtenues dans l'*Étape 1*

Description

Arrondi réalisable : Soit A_i, l'ensemble des arcs qui sont éléments d'un chemin critique ayant pour sommet terminal la tâche i.

- Si $\Gamma^-(i) = \varnothing$, nous gardons inchangé les valeurs des variables x_{ij} obtenues dans l'*Étape 1*

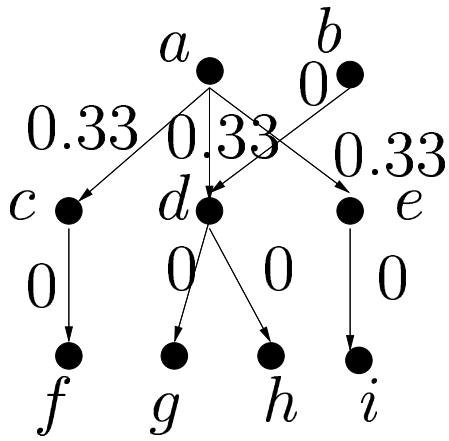
- Si $\exists$ *0-arc* $\in A_i$, $\forall k, (i,k) \in E, x_{ik} = 1$

Description

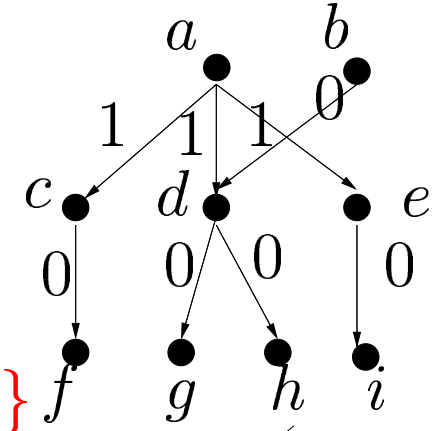
Arrondi réalisable : Soit A_i, l'ensemble des arcs qui sont éléments d'un chemin critique ayant pour sommet terminal la tâche i.

- Si $\Gamma^-(i) = \varnothing$, nous gardons inchangé les valeurs des variables x_{ij} obtenues dans l'*Étape 1*
- Si $\exists$ *0-arc* $\in A_i$, $\forall k, (i,k) \in E, x_{ik} = 1$
- Si $\exists$ *1-arc* $\in A_i$, alors la valeur des arcs sortants x_{ij} reste inchangée comme dans l'*Étape 1* et tous les *0-arcs* entrants sont transformés en *1-arcs*

Exemple

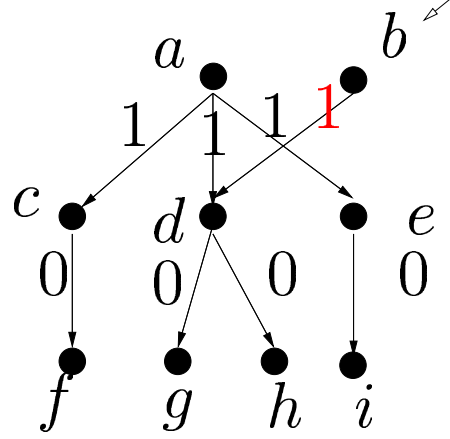


Arrondi



$A_d = \{(a, d)\}$

Arrondi réalisable



Performance relative

Performance relative

Soit t_i^h le début d'exécution de la tâche i donné par l'heuristique h
Soit t_i^* le début d'exécution de la tâche i donné par le programme linéaire

Performance relative

Soit t_i^h le début d'exécution de la tâche i donné par l'heuristique h
Soit t_i^* le début d'exécution de la tâche i donné par le programme linéaire

Lemme : Pour chaque tâche $i \in V, t_i^h \leq \frac{8}{5} t_i^*$.

.

Performance relative

Soit t_i^h le début d'exécution de la tâche i donné par l'heuristique h
Soit t_i^* le début d'exécution de la tâche i donné par le programme linéaire

Lemme : Pour chaque tâche $i \in V, t_i^h \leq \frac{8}{5} t_i^*$.

Preuve : La preuve se fait par induction sur les tâches i .

Performance relative

Soit t_i^h le début d'exécution de la tâche i donné par l'heuristique h
Soit t_i^* le début d'exécution de la tâche i donné par le programme linéaire

Lemme : Pour chaque tâche $i \in V, t_i^h \leq \frac{8}{5} t_i^*$.

Preuve : La preuve se fait par induction sur les tâches i .

- A_i contient un $0 - arc$ noté (j, i) tel que $e_{ji} < 0.25$.

Performance relative

Soit t_i^h le début d'exécution de la tâche i donné par l'heuristique h
Soit t_i^* le début d'exécution de la tâche i donné par le programme linéaire

Lemme : Pour chaque tâche $i \in V, t_i^h \leq \frac{8}{5} t_i^*$.

Preuve : La preuve se fait par induction sur les tâches i .

- A_i contient un $0 - arc$ noté (j, i) tel que $e_{ji} < 0.25$.
- A_i contient un $1 - arc$ noté (j, i) tel que $e_{ji} \geq 0.25$.

Performance relative

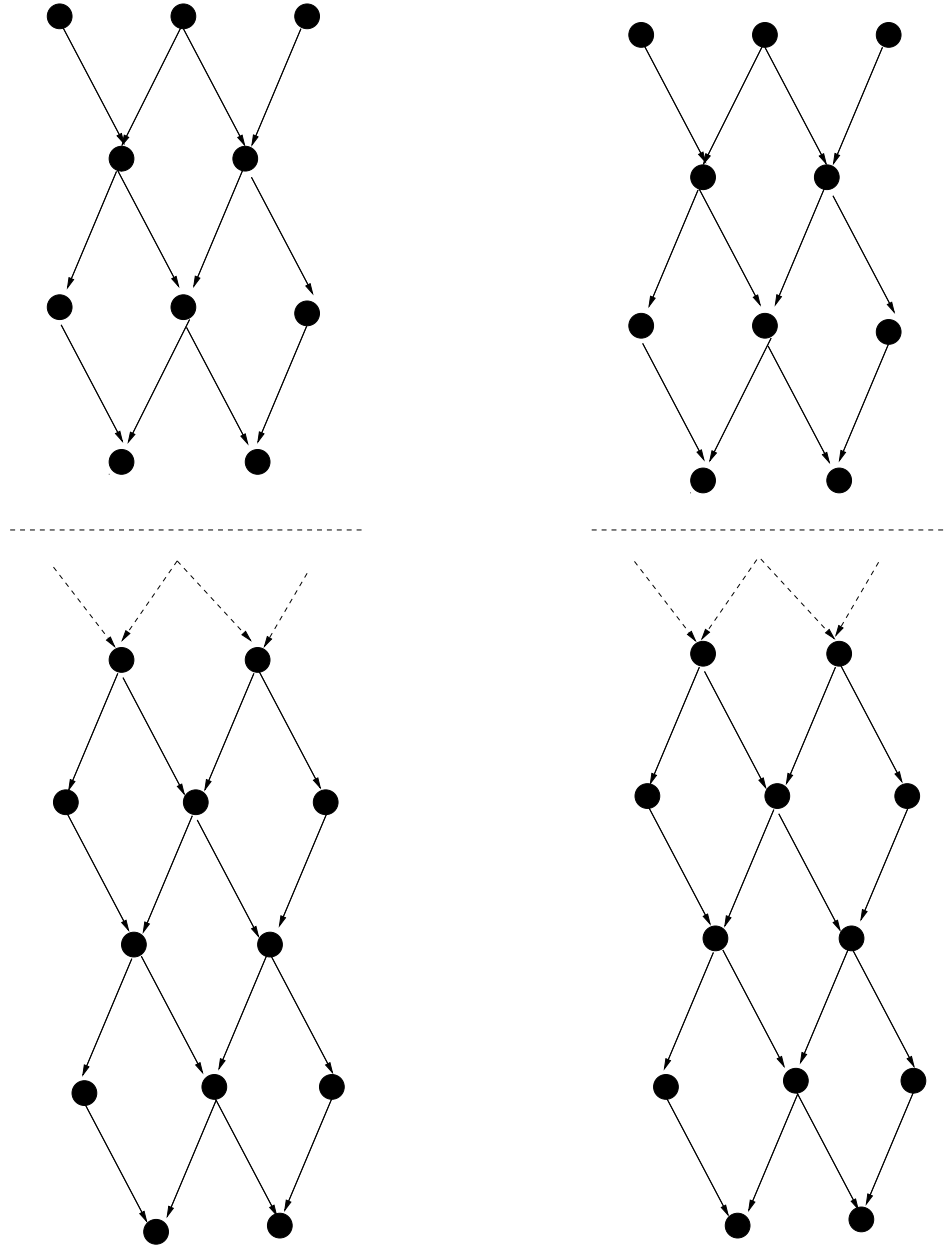
Soit t_i^h le début d'exécution de la tâche i donné par l'heuristique h
Soit t_i^* le début d'exécution de la tâche i donné par le programme linéaire

Lemme : Pour chaque tâche $i \in V, t_i^h \leq \frac{8}{5} t_i^*$.

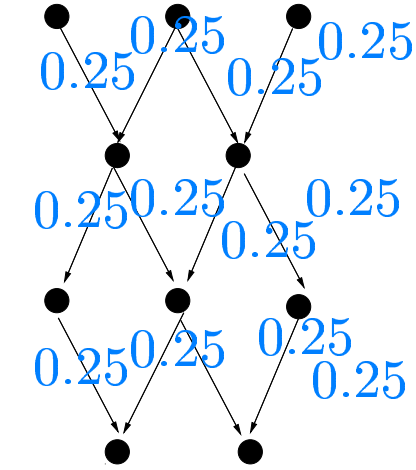
Preuve : La preuve se fait par induction sur les tâches i .

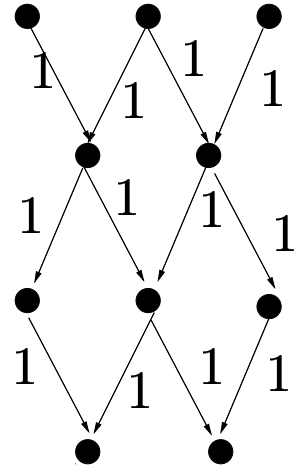
- A_i contient un $0 - arc$ noté (j, i) tel que $e_{ji} < 0.25$.
- A_i contient un $1 - arc$ noté (j, i) tel que $e_{ji} \geq 0.25$.
- A_i contient un $1 - arc$ noté (j, i) tel que $e_{ji} < 0.25$.

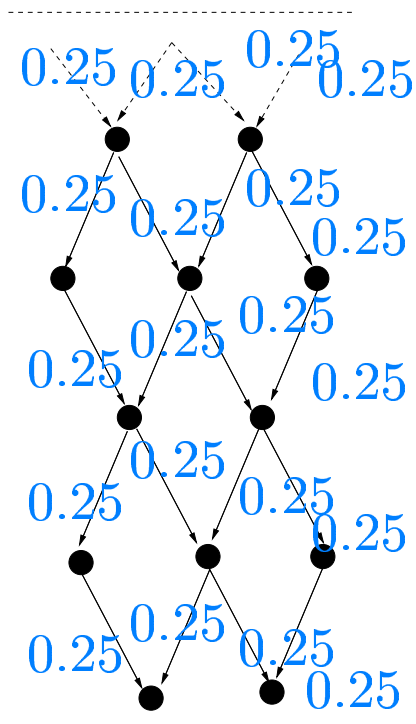
Borne supérieure

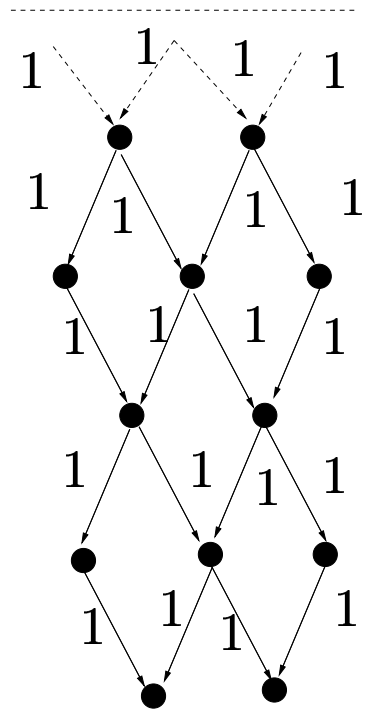


Borne supérieure

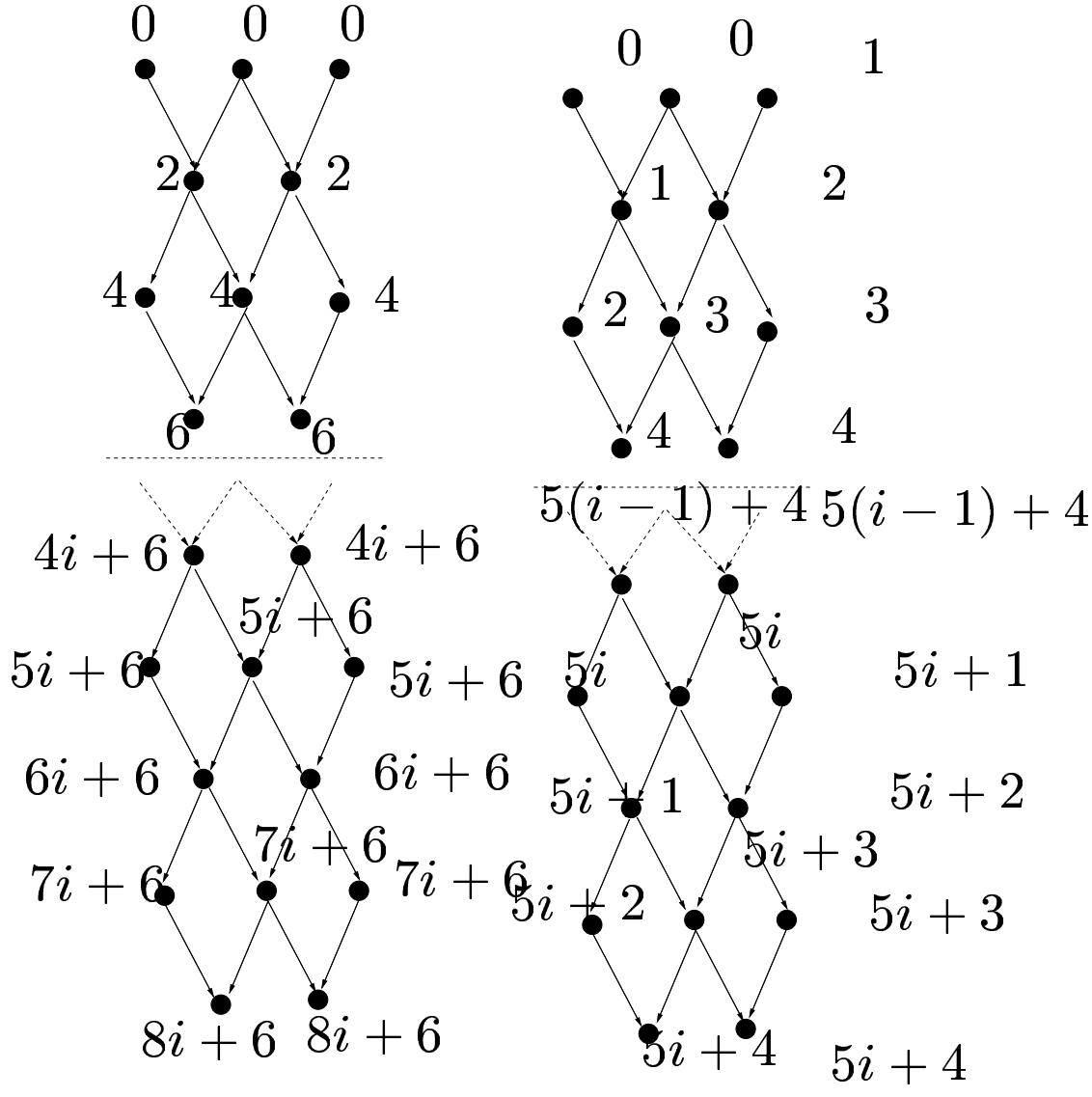








Borne supérieure



Extensions (1)

Extensions (1)

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$ et $\epsilon_{ij} = 0$

Extensions (1)

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$ et $\epsilon_{ij} = 0$
- Nombre de modules suffisant, m processeurs identiques par module

Extensions (1)

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$ et $\epsilon_{ij} = 0$
- Nombre de modules suffisant, m processeurs identiques par module

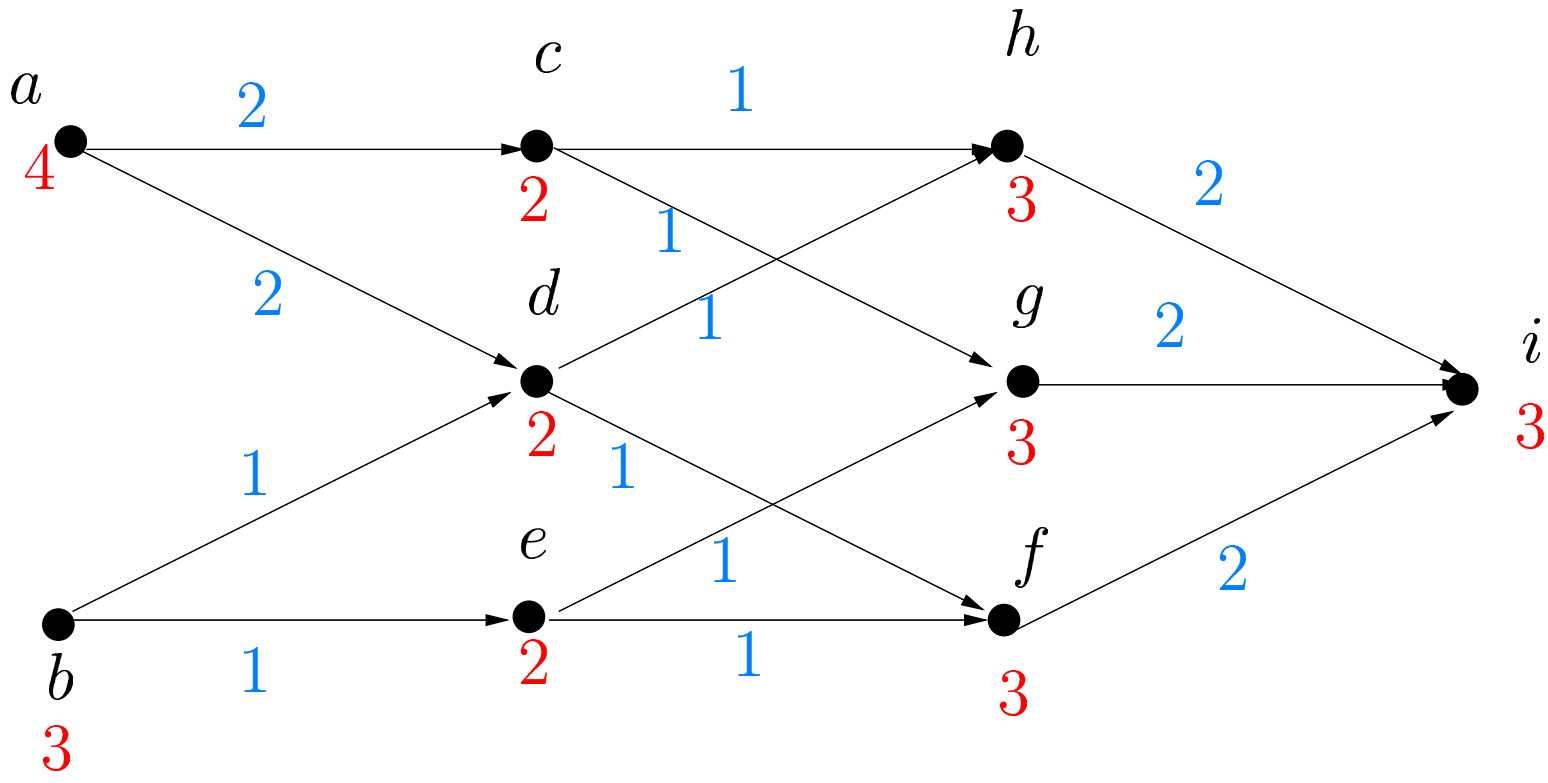
algorithme $(2 - \frac{2}{2m+1})$-approché [Bampis,Giroudeau,König 99]

Petits délais de communications

Définition : nous dirons que nous sommes dans l'hypothèse des petits délais de communications si $\Phi \geq 1$ où $\Phi = \dfrac{\min\limits_{i \in V} p_i}{\max\limits_{(k,j) \in E} c_{kj}}$

Petits délais de communications

Définition : nous dirons que nous sommes dans l'hypothèse des petits délais de communications si $\Phi \geq 1$ où $\Phi = \dfrac{\min\limits_{i \in V} p_i}{\max\limits_{(k,j) \in E} c_{kj}}$



Extensions (2)

Modèle classique

Extensions (2)

Modèle classique

- $\forall i \in V, p_i, \forall (i,j) \in E, c_{ij}$ et $\Phi \geq 1$

Extensions (2)

Modèle classique

- $\forall i \in V, p_i, \forall (i,j) \in E, c_{ij}$ et $\Phi \geq 1$
- Nombre de processeurs suffisant

Extensions (2)

Modèle classique

- $\forall i \in V, p_i, \forall (i,j) \in E, c_{ij}$ et $\Phi \geq 1$
- Nombre de processeurs suffisant

algorithme $\frac{2(1+\Phi)}{2\Phi+1}$-approché [Munier,Hanen 95]

Extensions (2)

Modèle classique

- $\forall i \in V, p_i, \forall (i,j) \in E, c_{ij}$ et $\Phi \geq 1$
- Nombre de processeurs suffisant

algorithme $\frac{2(1+\Phi)}{2\Phi+1}$-approché [Munier,Hanen 95]

Modèle hiérarchique

Extensions (2)

Modèle classique

- $\forall i \in V, p_i, \forall (i,j) \in E, c_{ij}$ et $\Phi \geq 1$
- Nombre de processeurs suffisant

algorithme $\frac{2(1+\Phi)}{2\Phi+1}$-approché [Munier,Hanen 95]

Modèle hiérarchique

- $\forall i \in V, p_i, \forall (i,j) \in E, c_{ij},\ \epsilon_{ij} = 0$ et $\Phi \geq 1$

Extensions (2)

Modèle classique

- $\forall i \in V, p_i, \forall (i,j) \in E, c_{ij}$ et $\Phi \geq 1$
- Nombre de processeurs suffisant

algorithme $\frac{2(1+\Phi)}{2\Phi+1}$-approché [Munier,Hanen 95]

Modèle hiérarchique

- $\forall i \in V, p_i, \forall (i,j) \in E, c_{ij},\ \epsilon_{ij} = 0$ et $\Phi \geq 1$
- Nombre de modules suffisant, 2 processeurs identiques par module

Extensions (2)

Modèle classique

- $\forall i \in V, p_i, \forall (i,j) \in E, c_{ij}$ et $\Phi \geq 1$
- Nombre de processeurs suffisant

algorithme $\frac{2(1+\Phi)}{2\Phi+1}$-approché [Munier,Hanen 95]

Modèle hiérarchique

- $\forall i \in V, p_i, \forall (i,j) \in E, c_{ij},\ \epsilon_{ij} = 0$ et $\Phi \geq 1$
- Nombre de modules suffisant, 2 processeurs identiques par module

algorithme $\frac{12(\Phi+1)}{12\Phi+1}$-approché [Bampis,Giroudeau,Kononov 00]

Nombre limité de modules

Modèle classique

Nombre limité de modules

Modèle classique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$

Nombre limité de modules

Modèle classique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$
- Nombre de processeurs borné

Nombre limité de modules

Modèle classique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$
- Nombre de processeurs borné

algorithme $(1 + \rho^{\infty} - \frac{\rho^{\infty}}{m})$-approché [Munier,Hanen 95]

Nombre limité de modules

Modèle classique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$
- Nombre de processeurs borné

algorithme $(1 + \rho^{\infty} - \frac{\rho^{\infty}}{m})$-approché [Munier,Hanen 95]

Modèle hiérarchique

Nombre limité de modules

Modèle classique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$
- Nombre de processeurs borné

algorithme $(1 + \rho^{\infty} - \frac{\rho^{\infty}}{m})$-approché [Munier,Hanen 95]

Modèle hiérarchique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$ et $\epsilon_{ij} = 0$

Nombre limité de modules

Modèle classique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$
- Nombre de processeurs borné

algorithme $(1 + \rho^{\infty} - \frac{\rho^{\infty}}{m})$-approché [Munier,Hanen 95]

Modèle hiérarchique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$ et $\epsilon_{ij} = 0$
- Nombre de modules borné, m processeurs identiques par module

Nombre limité de modules

Modèle classique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$
- Nombre de processeurs borné

algorithme $(1 + \rho^{\infty} - \frac{\rho^{\infty}}{m})$-approché [Munier,Hanen 95]

Modèle hiérarchique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$ et $\epsilon_{ij} = 0$
- Nombre de modules borné, m processeurs identiques par module

algorithme $(1 + (2 - \frac{1}{m})\rho^{\infty})$-approché

Nombre limité de modules

Modèle classique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$
- Nombre de processeurs borné

algorithme $(1 + \rho^{\infty} - \frac{\rho^{\infty}}{m})$-approché [Munier,Hanen 95]

Modèle hiérarchique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$ et $\epsilon_{ij} = 0$
- Nombre de modules borné, m processeurs identiques par module

algorithme $(1 + (2 - \frac{1}{m})\rho^{\infty})$-approché

Résultats négatifs (2)

Modèle classique

Résultats négatifs (2)

Modèle classique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$

Résultats négatifs (2)

Modèle classique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$
- m processeurs (m entrée du problème) $\&$ graphe biparti

Résultats négatifs (2)

Modèle classique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$
- m processeurs (m entrée du problème) $\&$ graphe biparti

$\nexists$ ρ-approximation avec $\rho < \frac{5}{4}$, sauf si $\mathcal{P} = \mathcal{NP}$

[Hoogeveen,Lenstra,Veltman 92]

Résultats négatifs (2)

Modèle classique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$
- m processeurs (m entrée du problème) $\&$ graphe biparti

$\nexists$ ρ-approximation avec $\rho < \frac{5}{4}$, sauf si $\mathcal{P} = \mathcal{NP}$

[Hoogeveen,Lenstra,Veltman 92]

Modèle hiérarchique

Résultats négatifs (2)

Modèle classique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$
- m processeurs (m entrée du problème) $\&$ graphe biparti

$\nexists$ ρ-approximation avec $\rho < \frac{5}{4}$, sauf si $\mathcal{P} = \mathcal{NP}$

[Hoogeveen,Lenstra,Veltman 92]

Modèle hiérarchique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$ et $\epsilon_{ij} = 0$

Résultats négatifs (2)

Modèle classique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$
- m processeurs (m entrée du problème) $\&$ graphe biparti

$\nexists$ ρ-approximation avec $\rho < \frac{5}{4}$, sauf si $\mathcal{P} = \mathcal{NP}$

[Hoogeveen,Lenstra,Veltman 92]

Modèle hiérarchique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$ et $\epsilon_{ij} = 0$
- 2 modules, m processeurs identiques par module (m entrée) $\&$ graphe biparti

Résultats négatifs (2)

Modèle classique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$
- m processeurs (m entrée du problème) $\&$ graphe biparti

$\nexists$ ρ-approximation avec $\rho < \frac{5}{4}$, sauf si $\mathcal{P} = \mathcal{NP}$

[Hoogeveen,Lenstra,Veltman 92]

Modèle hiérarchique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$ et $\epsilon_{ij} = 0$
- 2 modules, m processeurs identiques par module (m entrée) $\&$ graphe biparti

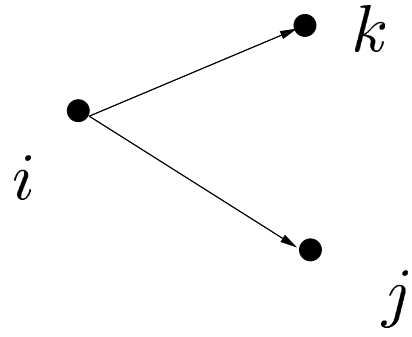
$\nexists$ ρ-approximation avec $\rho < \frac{4}{3}$, sauf si $\mathcal{P} = \mathcal{NP}$

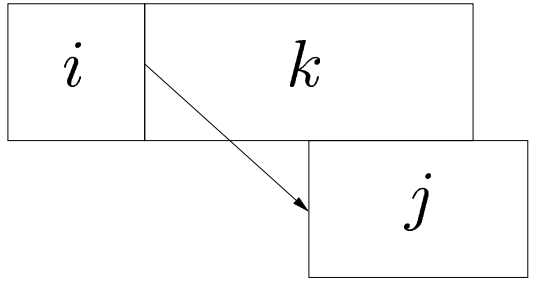
[Bampis,Giroudeau,König 01]

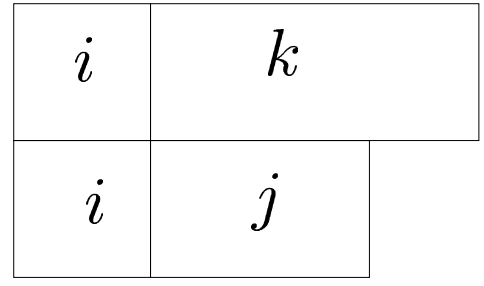
Introduction de la duplication

Modèle classique

Autorisation de la duplication







Sans duplication

Avec duplication

Résultats positifs (3) en présence de la duplication

Modèle classique

Résultats positifs (3) en présence de la duplication

Modèle classique

- $\forall j \in V, \min_{i \in \Gamma^-(j)} \{p_i\} \geq \max_{i \in \Gamma^-(j)} \{c_{ij}\}$

Résultats positifs (3) en présence de la duplication

Modèle classique

- $\forall j \in V, \min_{i \in \Gamma^-(j)} \{p_i\} \geq \max_{i \in \Gamma^-(j)} \{c_{ij}\}$
- Nombre de processeurs suffisant, duplication

Résultats positifs (3) en présence de la duplication

Modèle classique

- $\forall j \in V, \min_{i \in \Gamma^-(j)} \{p_i\} \geq \max_{i \in \Gamma^-(j)} \{c_{ij}\}$
- Nombre de processeurs suffisant, duplication

algorithme polynomial [Colin,Chrétienne 91]

Résultats positifs (3) en présence de la duplication

Modèle classique

- $\forall j \in V, \min_{i \in \Gamma^-(j)} \{p_i\} \geq \max_{i \in \Gamma^-(j)} \{c_{ij}\}$
- Nombre de processeurs suffisant, duplication

algorithme polynomial [Colin,Chrétienne 91]

Modèle hiérarchique

Résultats positifs (3) en présence de la duplication

Modèle classique

- $\forall j \in V, \min_{i \in \Gamma^-(j)} \{p_i\} \geq \max_{i \in \Gamma^-(j)} \{c_{ij}\}$
- Nombre de processeurs suffisant, duplication

algorithme polynomial [Colin,Chrétienne 91]

Modèle hiérarchique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$ et $\epsilon_{ij} = 0$

Résultats positifs (3) en présence de la duplication

Modèle classique

- $\forall j \in V, \min_{i \in \Gamma^-(j)} \{p_i\} \geq \max_{i \in \Gamma^-(j)} \{c_{ij}\}$
- Nombre de processeurs suffisant, duplication

algorithme polynomial [Colin,Chrétienne 91]

Modèle hiérarchique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$ et $\epsilon_{ij} = 0$
- Nombre de modules suffisant, 2 processeurs identiques par module, duplication

Résultats positifs (3) en présence de la duplication

Modèle classique

- $\forall j \in V, \min_{i \in \Gamma^-(j)} \{p_i\} \geq \max_{i \in \Gamma^-(j)} \{c_{ij}\}$
- Nombre de processeurs suffisant, duplication

algorithme polynomial [Colin,Chrétienne 91]

Modèle hiérarchique

- $\forall i \in V, p_i = 1, \forall (i,j) \in E, c_{ij} = 1$ et $\epsilon_{ij} = 0$
- Nombre de modules suffisant, 2 processeurs identiques par module, duplication

algorithme polynomial [Bampis,Giroudeau,König 99]

Idée de l'algorithme

Idée de l'algorithme

1. La première étape de l'algorithme calcule pour chaque tâche (et ses copies) sa date de début d'exécution au plus tôt et construit un graphe critique.

Idée de l'algorithme

1. La première étape de l'algorithme calcule pour chaque tâche (et ses copies) sa date de début d'exécution au plus tôt et construit un graphe critique.

2. La seconde étape utilise le graphe critique et la duplication des tâches pour construire un ordonnancement optimal.

Idée de l'algorithme

1. La première étape de l'algorithme calcule pour chaque tâche (et ses copies) sa date de début d'exécution au plus tôt et construit un graphe critique.

2. La seconde étape utilise le graphe critique et la duplication des tâches pour construire un ordonnancement optimal.

Notation : $b_i =$ date de début d'exécution au plus tôt de la tâche i

Idée de l'algorithme

1. La première étape de l'algorithme calcule pour chaque tâche (et ses copies) sa date de début d'exécution au plus tôt et construit un graphe critique.

2. La seconde étape utilise le graphe critique et la duplication des tâches pour construire un ordonnancement optimal.

Notation : $b_i =$ date de début d'exécution au plus tôt de la tâche i

Définition: Une tâche i est un prédécesseur critique de j si l'arc $(i, j) \in E$ et $b_j = b_i + 1$.

Algorithme

- Calculer la borne inférieure des dates de début d'exécution :

Algorithme

- Calculer la borne inférieure des dates de début d'exécution :
  - Pour chaque source i de G, $b_i = 0$.

Algorithme

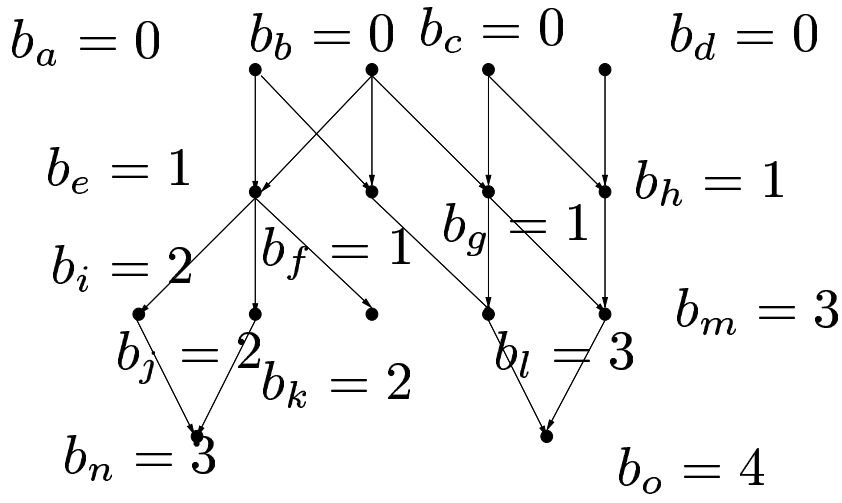
- Calculer la borne inférieure des dates de début d'exécution :
  - Pour chaque source i de G, $b_i = 0$.
  - Si dans le graphe induit par les prédécesseurs critiques de i, il existe au moins 3 prédécesseurs qui sont à la même distance l et ayant la date de début d'exécution au plus tôt maximale alors une communication est introduite entre i et ses prédécesseurs critiques.

Algorithme

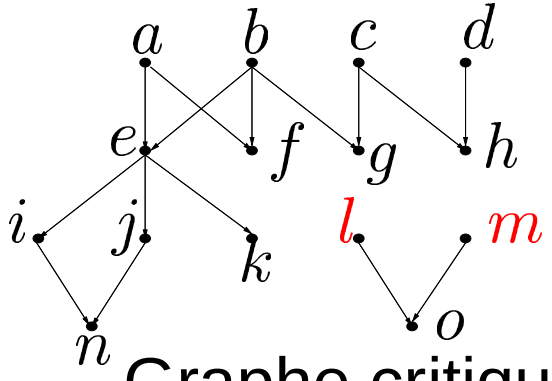
- Calculer la borne inférieure des dates de début d'exécution :
  - Pour chaque source i de G, $b_i = 0$.
  - Si dans le graphe induit par les prédécesseurs critiques de i, il existe au moins 3 prédécesseurs qui sont à la même distance l et ayant la date de début d'exécution au plus tôt maximale alors une communication est introduite entre i et ses prédécesseurs critiques.
- Affecter chaque sous-graphe critique défini à partir d'une tâche sans successeur à un module différent et exécuter chaque copie des tâches à sa date au plus tôt.

Exemple

Graphe UET-UCT



Date de début d'exécution



Graphe critique

0	1	2	3	4	5
a	e	i	n		
b		j			
a	e	k			
b					
a	f				
b					
c	g				
b					
d	h				
c					
			l	o	
			m		

Ordonnancement associé

Tableau récapitulatif

Modèle	Modèle homogène		Modèle hiérarchiqu	
	Type de machine	Borne(s)	Type de machine	Borne(s
UET-UCT	$\bar{P}$	$\frac{7}{6} \leq \rho \leq \frac{4}{3}$,	$\bar{P}(P2)$	$\frac{5}{4} \leq \rho \leq$
SCT	$\bar{P}$	$\rho \leq \frac{2(1+\Phi)}{2\Phi+1}$,	$\bar{P}(P2)$	$\rho \leq \frac{12(1+}{12\Phi+}$
UET-UCT,biparti	P	$\frac{5}{4} \leq \rho$	$P2(P)$	$\frac{4}{3} \leq \rho$
UET-UCT	P	$\rho \leq \frac{7}{3} - \frac{4}{3m}$	$P(Pm)$	$\rho \leq (3 - \frac{}{M}$
UET-UCT,dup	$\bar{P}$	$\rho = 1$	$\bar{P}(P2)$	$\rho = 1$

Tableau de nouveaux résultats (1)

Nouveaux résultats avec processeurs homogènes

Type de machine	(c_{ij}, ϵ_{ij})	$C_m{}^{ax}$	Complexité
$\bar{P}(P2)$	$(2,1)$	5	$\mathcal{NP}$-complet
$\bar{P}(P2)$	$(2,1)$	3	Polynomiale
$\bar{P}(P2)$	$(2,1)$	4	?
$\bar{P}(Pk),\ k \geq 3$	$(2,1)$	4	$\mathcal{NP}$-complet
$\bar{P}(Pk),\ k \geq 3$	$(c,1)$	$c+2$	$\mathcal{NP}$-complet
$\bar{P}(Pk),\ k \geq 3$	(c,c')	$c+2$	$\mathcal{NP}$-complet

Tableau de nouveaux résultats (2)

Nouveaux résultats avec processeurs hétérogènes

Type de machine	(c_{ij}, ϵ_{ij})	$(p_i^{\Pi^1}, p_i^{\Pi^2})$	$C_m{}^{ax}$	Complexité
$P2(Pk \bigcup P2k)$	$(1,0)$	$(2,1)$	4	$\mathcal{NP}$-complet
$P2(Pk \bigcup P2k)$	(c,c')	$(2,1)$	$3+c$	$\mathcal{NP}$-complet
$P2(Pk \bigcup P2k)$	$(1,0)$	$(2,1)$	3	$\mathcal{NP}$-complet
$P2(Pk \bigcup Pk')$	(c,c')	$(2,1)$	$3+c'$	$\mathcal{NP}$-complet
$P2(Pk \bigcup Pk)$	$(1,0)$	$(2,1)$	5	$\mathcal{NP}$-complet
$P2(Pk \bigcup Pk)$	(c,c')	$(2,1)$	$4+c$	$\mathcal{NP}$-complet

Conclusion

- les techniques résistent aux passages du modèle homogène à hiérarchique,

Conclusion

- les techniques résistent aux passages du modèle homogène à hiérarchique,
- “Perte de la qualité” des solutions approchées,

Conclusion

- les techniques résistent aux passages du modèle homogène à hiérarchique,
- “Perte de la qualité” des solutions approchées,
- Seuil d’approximation relevé

Perspectives

- Prendre en considération plusieurs critères

Perspectives

- Prendre en considération plusieurs critères
- Developper des algorithmes approchées avec $c_{ij} \geq 2$ et $\epsilon_{ij} \neq 0$

Perspectives

- Prendre en considération plusieurs critères
- Developper des algorithmes approchées avec $c_{ij} \geq 2$ et $\epsilon_{ij} \neq 0$

Ne peut-on pas appliquées les techniques utilisées dans le modèle hiérarchique pour le modèle homogène