MIS 7 : Sujet d'étude n°2

Merwan Achibet
Université du Havre

1 Problème 1

1.1 Dates au plus tôt

$\underline{b_1}$
1 n'a pas de prédécesseur donc $b_1 = 0$.

$\underline{b_2}$
2 a un seul prédécesseur (1), donc $b_2 = b_1 + p_1 = 0 + 3 = 3$

$\underline{b_3}$
3 a un seul prédécesseur (1), donc $b_3 = b_1 + p_1 = 0 + 3 = 3$

$\underline{b_4}$
4 a un seul prédécesseur (1), donc $b_4 = b_1 + p_1 = 0 + 3 = 3$

$\underline{b_5}$
5 a trois prédécesseurs (2, 3 et 4), on choisit parmi eux s tel que $b_s + p_s + c_{s5}$ soit maximum.
pour 2 : $b_2 + p_2 + c_{25} = 3 + 3 + 2 = 8$
pour 3 : $b_3 + p_3 + c_{35} = 3 + 5 + 3 = 11$
pour 4 : $b_4 + p_4 + c_{45} = 3 + 3 + 1 = 7$
on prend donc $s = 3$.

$$\begin{aligned} b_5 &= \max(b_S + p_s, \max_{i \in \mathrm{PRED}(5) - \{s\}}(b_i + p_i + c_{i5})) \\ &= \max(3 + 5, \max(3 + 3 + 2, 3 + 3 + 1)) \\ &= 8 \end{aligned}$$

$\underline{b_6}$
6 a deux prédécesseurs (3 et 4), on choisit parmi eux s tel que $b_s + p_s + c_{s6}$ soit maximum.
pour 3 : $b_3 + p_3 + c_{36} = 3 + 5 + 2 = 10$
pour 4 : $b_4 + p_4 + c_{46} = 3 + 3 + 1 = 7$
on prend donc $s = 3$.

$$\begin{aligned} b_6 &= \max(b_S + p_s, \max_{i \in \text{PRED}(6)-\{s\}}(b_i + p_i + c_{i6})) \\ &= \max(3 + 5, 3 + 3 + 1) \\ &= 8 \end{aligned}$$

$\underline{b_7}$
7 a deux prédécesseurs (5 et 6), on choisit parmi eux s tel que $b_s + p_s + c_{s7}$ soit maximum.
pour 5 : $b_5 + p_5 + c_{57} = 8 + 4 + 1 = 13$
pour 6 : $b_6 + p_6 + c_{67} = 8 + 3 + 1 = 12$
on prend donc $s = 5$.

$$\begin{aligned} b_7 &= \max(b_S + p_s, \max_{i \in \text{PRED}(7)-\{s\}}(b_i + p_i + c_{i7})) \\ &= \max(8 + 4, 8 + 3 + 1) \\ &= 12 \end{aligned}$$

Au final, on obtient les bornes au plus tôt suivantes :

X	1	2	3	4	5	6	7
PRED(**X**)	0	3	3	3	8	8	12

1.2 Diagramme de Gantt

On construit un graphe critique contenant les arcs (i, j) tels que $b_i + p_i + c_{ij} > b_j$.

<u>arc (1,2)</u>
$b_1 + p_1 + c_{12} = 0 + 3 + 1 = 4 > b_2 = 3$ Donc on le conserve

<u>arc (1,3)</u>
$b_1 + p_1 + c_{13} = 0 + 3 + 2 = 5 > b_3 = 3$ Donc on le conserve

arc (1,4)
$b_1 + p_1 + c_{14} = 0 + 3 + 1 > b_4 = 3$ Donc on le conserve

arc (2,5)
$b_2 + p_2 + c_{25} = 3 + 3 + 2 = b_5 = 8$ Donc on le conserve

arc (3,5)
$b_3 + p_3 + c_{35} = 3 + 5 + 3 > b_5 = 8$ Donc on le conserve

arc (3,6)
$b_3 + p_3 + c_{36} = 3 + 5 + 2 > b_5 = 8$ Donc on le conserve

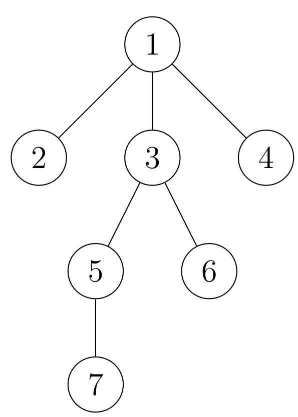
arc (4,5)
$b_4 + p_4 + c_{45} = 3 + 3 + 1 < b_5 = 8$ Donc on ne le conserve pas

arc (4,6) arc(4, 6)
$b_4 + p_4 + c_{46} = 3 + 3 + 1 < b_6 = 8$ Donc on ne le conserve pas

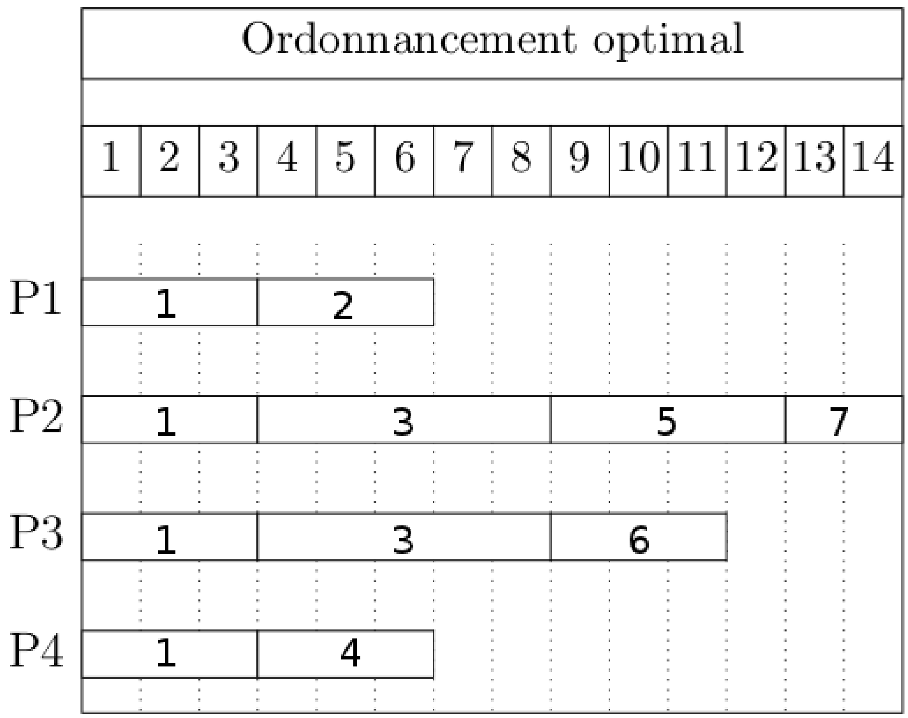
arc (5,7)
$b_5 + p_5 + c_{57} = 8 + 4 + 1 > b_7 = 12$ Donc on le conserve

arc (6,7)
$b_6 + p_6 + c_{67} = 8 + 3 + 1 = b_7 = 12$ Donc on ne le conserve pas

Le graphe critique est donc :



Á partir du graphe critique, on obtient le diagramme de Gantt suivant :


Ordonnancement optimal
1 2 3 4 5 6 7 8 9 10 11 12 13 14
P1
1
2
P2
1
3
5
7
P3
1
3
6
P4
1
4

2 Problème 2

```
entrée :
       matrice A (de taille n*n)
       vecteur b (de taille n)
       entier n

sortie :
       vecteur solution x

DEBUT

      // Initialisation du vecteur solution.
      x = [0,0,0,0,0,0,0,0]

      // Le processeur maître distribue les données.
      SI moi = 0 ALORS

         POUR i = 1 à P

              // On envoie uniquement les lignes utiles de A.
              // Chaque processeur en traîte deux.
              envoyer(moi, i, A[2*(i-1)], n)
              envoyer(moi, i, A[2*(i-1)+1], n)

              // On envoie b en entier à tout le monde.
              envoyer(moi, i, b, n)

        FIN POUR

        // Attend les 2 solutions de chaque esclave
        // (une fois qu'ils ont convergé).
        POUR i = 1 à P
              recevoir(moi, i, x[2*(i-1)], 1);
              recevoir(moi, i, x[2*(i-1)+1], 1);

        // Affiche la solution.
        affiche(x)
```

```
// Les processeurs esclaves éxécutent les calculs.
SINON

  convergence = faux

  a1 : première ligne de A à traîter
  a2 : seconde ligne de A à traîter
  b : vecteur b

  recevoir(moi, 0, a1, n)
  recevoir(moi, 0, a2, n)
  recevoir(moi, 0, b, n)

  TANT QUE convergence = faux

       // Calcule l'indice de la ligne en fonction
       // du numéro du processeur.
       i = 2*(moi-1)

       // Calcule les nouvelles valeurs de X
       x[i] = calculer(i, x, b)
       x[i+1] = calculer(i+1, x, b)

       // Diffuser les résultats et recevoir
       // ceux des autres processeurs.
       POUR i = 1 à P différent de moi
           envoyer(moi, i, x, n);
           recevoir(moi, i, x2, n);
           x[2*(i-1)] = x2[2*(i-1)]
           x[2*(i-1)+1] = x2[2*(i-1)+1]

       // Vérifie la convergence du calcul.
       convergence = t(x,y)

       // Enregistre x pour comparer à la prochaine itération.
       y = x;
```

```
        FIN TANT QUE

      FIN SINON

FIN

PROCEDURE calculer : i, x, b

         xi = 0

         SI i = 1 ALORS
            xi = (b[1] - x[2] - x[3]) / 2
         SINON SI i = 2 ALORS
            xi = (b[2] - x[1] - x[3] - x[4]) / 2
         SINON SI i = 7 ALORS
            xi = (b[7] - x[5] - x[6] - x[8]) / 2
         SINON SI i = 8 ALORS
            xi = (b[8] - x[6] - x[7]) / 2
         SINON
            xi = ([i] - x[i-2] - x[i-1] - x[i+1] - x[i+2]) / 2
         FIN SI

         retourner xi

FIN PROCEDURE
```

Dans cet algorithme, le processeur maître ne réalise pas de calcul mais répartit les données sur les quatres processeurs esclaves. Puisqu'il y a 4 processeurs et que la matrice est de dimension 8, on envoie deux lignes de A à chaque processeur. Par exemple, le processeur 1 va traîter les ligne 1 et 2 tandis que le processeur 2 va traîter les lignes 3 et 4.

Après chaque série de calcul, les processeurs esclaves envoient leurs deux valeurs de x aux autres. De cette façon, les solutions locales de chaque P sont continuellement mises à jour.

Le test de convergence est effectué de façon décentralisée dans chaque processeur esclave. Une fois qu'un processeur a convergé, il envoie son résultat au processeur maître qui affiche le résultat final lorsque tous ses esclave ont convergé.

Le graphe suivant illustre les premières itérations de l'algorithme.