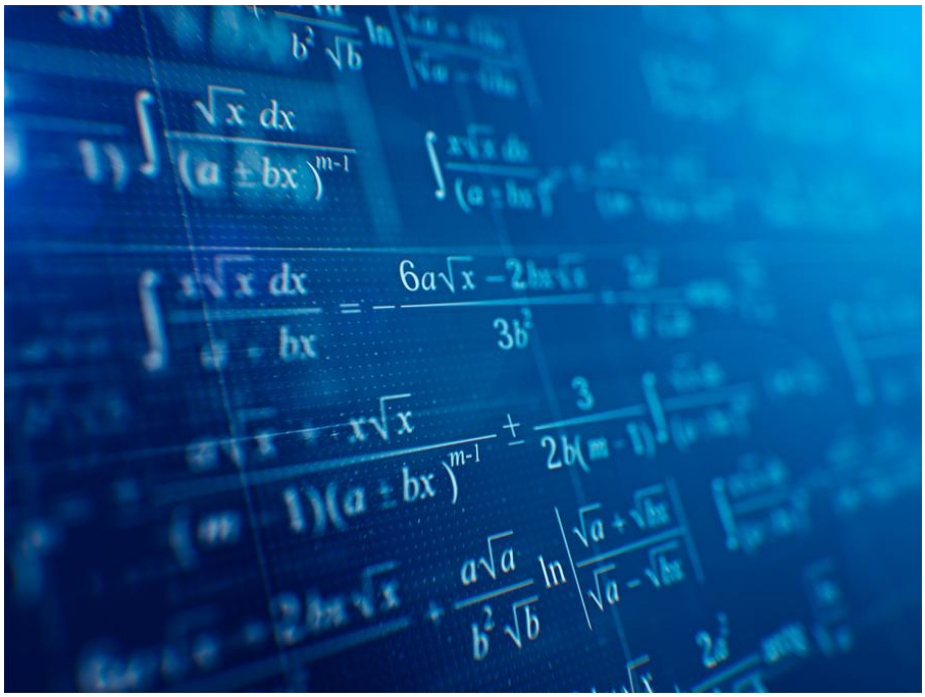
INP ENSEEIHT

Recherche Opérationnelle : TP5 et 6



Hamza MOUDDENE / Anass TYOUBI

15 janvier 2021

1 Prise en main : ordonnancement avec contraintes de précédence

Problème ($P1$) : Soit un ensemble de tâches T, l'objectif est de déterminer les dates d'exécution de chaque tâche de manière à minimiser la durée totale d'exécution tout en respectant les durées et les contraintes de précédence entre les tâches.

1.1 Comparaison de la solution obtenue avec celle du programme linéaire

En appliquant l'algorithme de **Bellman Ford** sur le graphe potentiel-tâche correspondant au Tableau1, on retrouve que le plus long chemin
est : debut -> A -> B -> D -> fin avec une durée totale d'exécution $t_f = 9$ tout en respectant les durées et les contraintes de précédence entre les tâches. Ce résultat est identique à la solution obtenue avec celle du programme linéaire. Or la solution avec le plus long chemin retourne un variable t plus exacte.

1.2 Résultat

```
Min tfin
Subject to
 -t[1] + t[2] ≥ 2.0
 -t[1] + t[3] ≥ 2.0
 -t[2] + t[4] ≥ 3.0
 -t[3] + t[4] ≥ 1.0
 -t[3] + t[5] ≥ 1.0
 -t[4] + tfin ≥ 4.0
 -t[5] + tfin ≥ 1.0
 t[1] ≥ 0.0
 t[2] ≥ 0.0
 t[3] ≥ 0.0
 t[4] ≥ 0.0
 t[5] ≥ 0.0

start solve ... ... end solve

Solution PL:
        t=[0.0, 2.0, 2.0, 5.0, 8.0]     tfin=9.0
```

FIGURE 1 – Solution obtenue avec la méthode de la programmation linéaire

```
t = [0.0, 2.0, 2.0, 5.0, 3.0]
tfin = 9.0
```

FIGURE 2 – Solution obtenue avec la méthode Bellman-Ford

2 Job-shop : ordonnancement avec contraintes de précédence et contraintes de ressources

2.1 Relaxation des contraintes de ressources

Le problème du job-shop consiste à planifier un ensemble de travaux pour minimiser la durée totale d'exécution tout en respectant des contraintes de précédence (chaque travail est décomposé en opérations à réaliser dans l'ordre) et de ressources (chaque opération utilise une machine et chaque machine ne peut traiter qu'une opération à la fois).
Soit une relaxation (R) qui consiste à ignorer les contraintes de ressources du job-shop. Et puisque le Tableau1 dépend juste de la durée d'une tâche, donc (R) est équivalent au problème $(P1)$.

```
Min tfin
Subject to
 -t[1,1] + t[1,2] ≥ 6.0
 -t[1,2] + t[1,3] ≥ 7.0
 -t[2,1] + t[2,2] ≥ 3.0
 -t[2,2] + t[2,3] ≥ 5.0
 -t[2,3] + tfin ≥ 1.0
 -t[1,3] + tfin ≥ 0.0
 t[1,1] ≥ 0.0
 t[2,1] ≥ 0.0
 t[1,2] ≥ 0.0
 t[2,2] ≥ 0.0
 t[1,3] ≥ 0.0
 t[2,3] ≥ 0.0

start solve ... ... end solve

Solution PL:
        t=[0.0 6.0 13.0; 0.0 3.0 8.0]   tfin=13.0
```

FIGURE 3 – Solution obtenue avec la méthode de la programmation linéaire

```
t = [6.0, 9.223372036854776e18, 0.0, 3.0, 8.0]
tfin = 13.0
```

FIGURE 4 – Solution obtenue avec la méthode Bellman-Ford