Débuter avec CPLEX en mode OPL

Kenza Oufaska et OUDANI Mustapha
Université Internationale de Rabat

Plan

Introduction

OPL Studio

Introduction

Historique et Applications

- CPLEX est créé par Robert E. Bixby en utilisant le langage C en 1987
- Racheté par ILOG en 1997
- Mentionné dans 95% des articles qui citent un solveur
- Utilisé comme solveur standard dans les applications de la chaîne logistique
- Employé par plusieurs compagnies aériennes (Delta, Continental, etc.)
- Lauréat du *INFORMS Impact Award* en 2004

Introduction

Problèmes couverts par CPLEX

- Programmation linéaire
- Programmation linéaire mixte
- Programmation quadratique
- Programmation mixte quadratique
- Programmation à contraintes quadratiques
- Programmation mixte à contraintes quadratiques

Introduction

Algorithmes

- **Programmation linéaire :**
  - Simplex Primal
  - Simplex Dual
  - Point intérieur (barrier)
- **Programmation quadratique :**
  - Simplex Primal
  - Simplex Dual
  - Point intérieur (barrier)
- **Programmation à contraintes quadratiques :**
  - Point intérieur (barrier)

Introduction à OPL

OPL : Optimization Programming Language

- OPL est un langage de haut niveau pour la description des programmes mathématiques
- Développé par ILOG, utilisé avec CPLEX pour modéliser et résoudre des problèmes d'optimisation
- OPL Studio est un Environnement de Développement Intégré (EDI) pour OPL

Eléments de syntaxe

Définitions des intervalles

- Pour créer un intervalle, on utilise le mot-clé range
  range I=1..10 ;
  range float R=5.0..50.0 ;

Définitions des variables de décisions

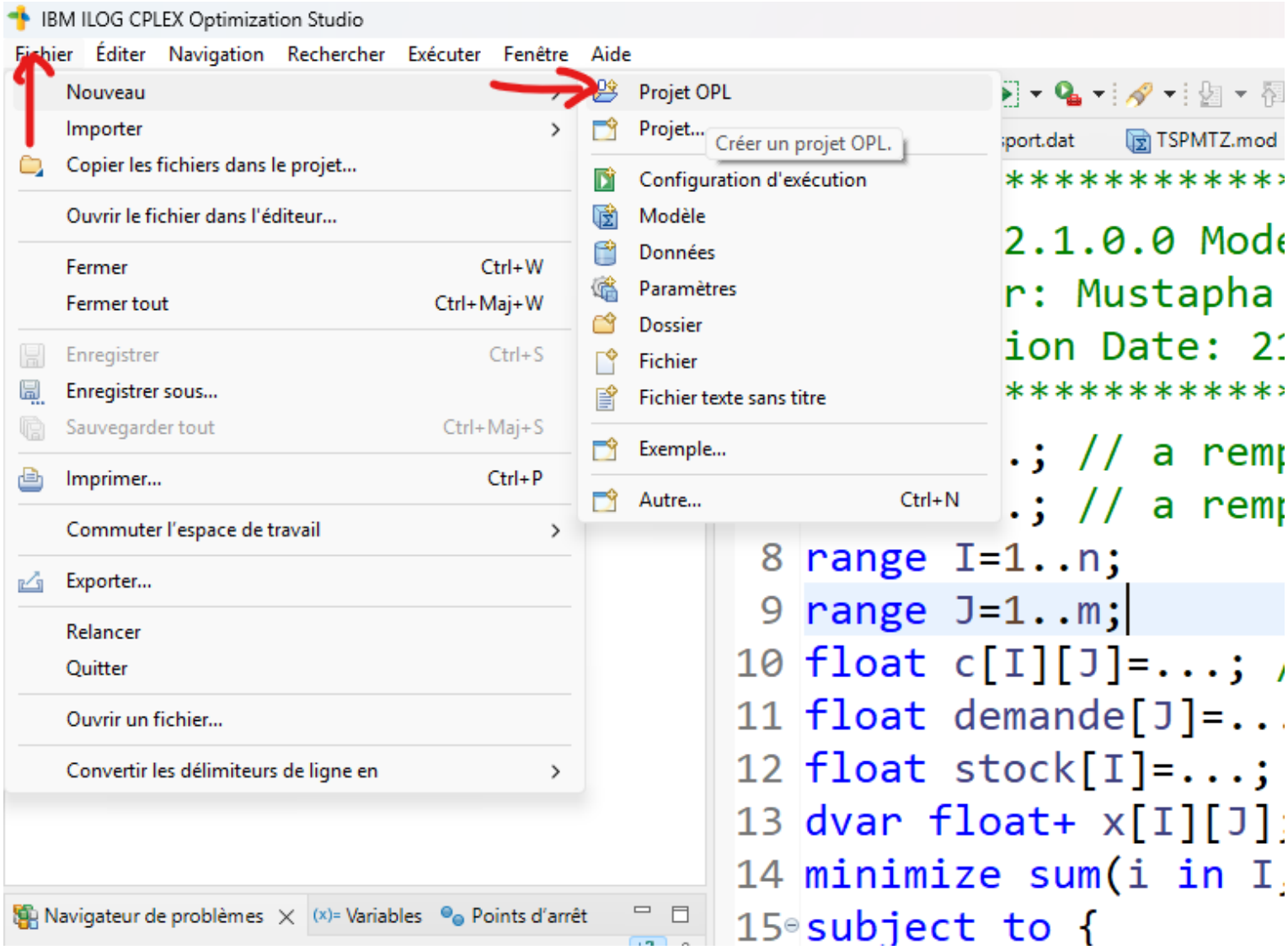
- Pour déclarer des variables de décisions, on utilise les mots-clés suivants le type de la variable boolean, int, float, int+, float+
  dvar boolean x ;
  dvar float y in R ;

Manipulation de CPLEX Studio IDE

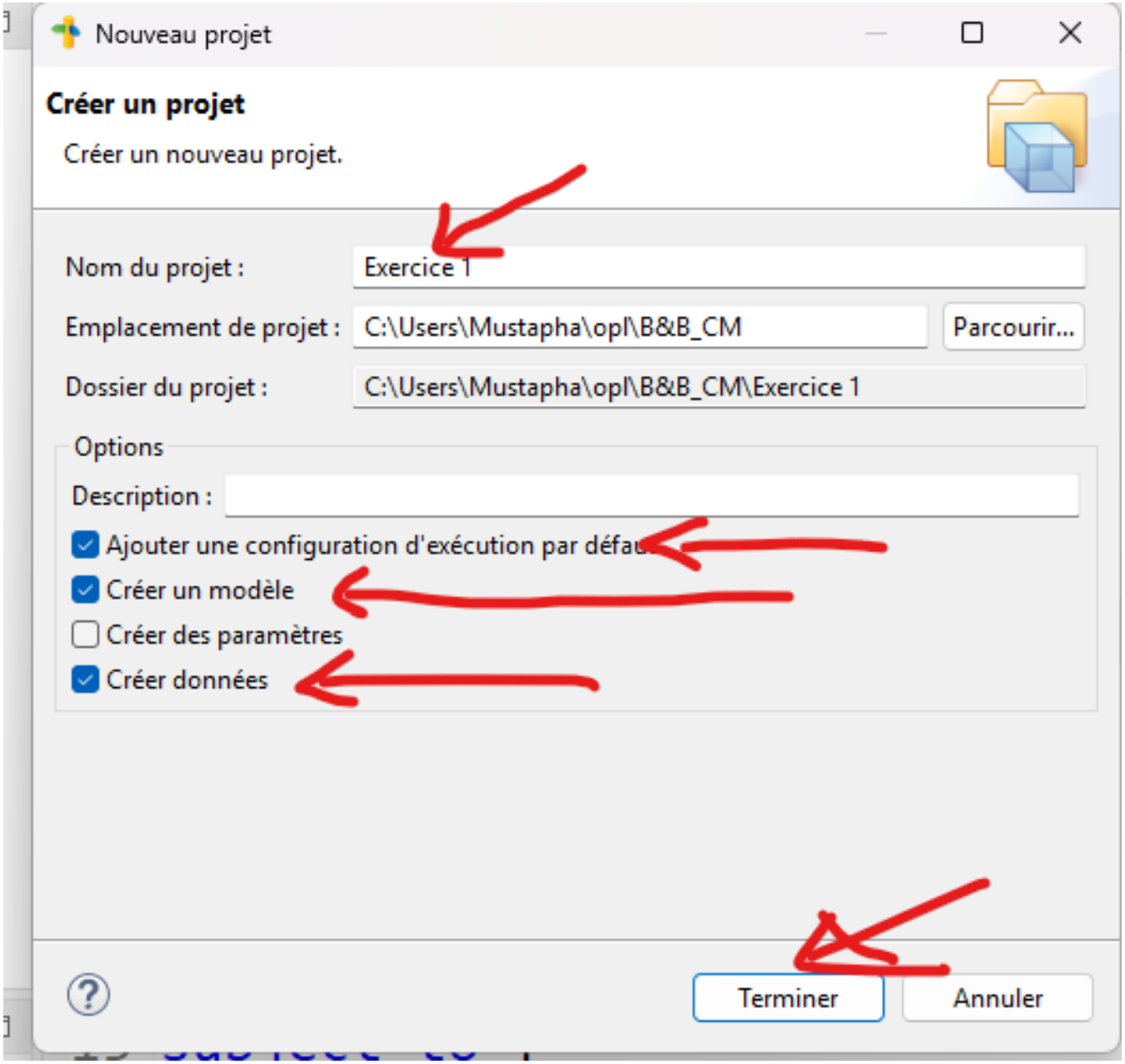
Manipulation de CPLEX Studio IDE

- Pour lancer CPLEX Studio IDE, utilisez le menu démarrer
- Créez un nouveau projet : `Fichier → Nouveau → Projet OPL`
- Pour insérer un modèle : `Nouveau → Modèle (.mod)`
- Pour insérer des données : `Nouveau → Données (.dat)`
- Pour exécuter un modèle : créer une configuration d'exécution

Manipulation de CPLEX Studio IDE



Manipulation de CPLEX Studio IDE



Manipulation de CPLEX Studio IDE

Le fichier .mod

```
/***** affectation.mod *****/
/* Ensembles */
{string} Ouvriers = ...;
{string} Taches   = ...;
int Eff[Ouvriers][Taches] = ...;
int Qual[Ouvriers][Taches] = ...; // valeurs 0/1
/* Variables de décision : x[i][j] = 1 si l'ouvrier i est affecté à la tâche j */
dvar boolean x[Ouvriers][Taches];
/* Objectif : maximiser l'efficacité totale */
maximize
  sum (i in Ouvriers, j in Taches) Eff[i][j] * x[i][j];
/* Contraintes */
subject to {
/* (1) Chaque ouvrier est affecté à exactement une tâche */
  forall (i in Ouvriers)
    sum (j in Taches) x[i][j] == 1;
/* (2) Chaque tâche reçoit exactement un ouvrier */
  forall (j in Taches)
    sum (i in Ouvriers) x[i][j] == 1;
/* (3) Interdictions (barres "--") : on force x[i][j] = 0 si non qualifié */
  forall (i in Ouvriers, j in Taches)
    x[i][j] <= Qual[i][j];}
```

Manipulation de CPLEX Studio IDE

Le fichier .dat

```
/***** affectation.dat *****/

Ouvriers = {"O1", "O2", "O3", "O4"};
Taches   = {"T1", "T2", "T3", "T4"};

/* Matrice d'efficacité (Eff[i][j])
   Une valeur numérique représente l'efficacité,
   même si la tâche n'est pas autorisée (elle sera bloquée par Qual). */
Eff = [
  [45,  0,  0, 30],
  [50, 55, 15,  0],
  [ 0, 60, 25, 75],
  [45,  0,  0, 35]
];

/* Matrice de qualification (Qual[i][j])
   1 = l'ouvrier est qualifié, 0 = non qualifié  */
Qual = [
  [1, 0, 0, 1],
  [1, 1, 1, 0],
  [0, 1, 1, 1],
  [1, 0, 0, 1]
];
```

Exemple de Modèle OPL : Problème de Transport

Modèle (.mod) pour le Problème de Transport

```
{string} usines = ... ;
{string} clients = ... ;
float cout[usines][clients] = ... ;
dvar float+ x[usines][clients] ;
minimize sum (i in usines, j in clients)
cout[i][j] * x[i][j] ;
subject to {
forall (i in usines) sum(j in clients)
x[i][j] <= offre[i];
forall (j in clients) sum(i in usines)
x[i][j] >= demande[j];
    }
```

Exemple de Modèle OPL : Problème de Transport

Données (.dat) pour le Problème de Transport

```
nbr_usines = 3 ;
nbr_clients = 4 ;
usines = {"1", "2", "3"} ;
clients = {"1", "2", "3", "4"} ;
offre = [35, 50, 40] ;
demande = [45, 20, 30, 30] ;
cout = [[8, 6, 10, 9],
[9, 12, 13, 7],[14, 9, 16, 5]] ;
```

Exemple de Modèle OPL : Problème de Sac à Dos

Modèle (.mod) pour le Problème de Sac à Dos

```
int n = ... ;
int profit[1..n] = ... ;
int poids[1..n] = ... ;
int poidmax = ... ;
dvar int x[1..n] ;
maximize
sum(i in 1..n) profit[i] * x[i] ;
subject to {
sum(i in 1..n) poids[i] * x[i]
<= poidmax ;
    }
```

Exemple de Modèle OPL : Données pour le Problème de Sac à Dos

Données (.dat) pour le Problème de Sac à Dos

```
n = 3 ; poidmax = 59 ; profit = [10, 11, 13] ; poids =
[10, 12, 13] ;
```