

# Chapitre 1

## Introduction à la Recherche Opérationnelle

Dr. E. B. Med Mahmoud

LGTR

February 21, 2024

# Objectifs généraux

## Vocabulaire

- Modélisation des systèmes complexes;
- Analyse et interprétation des données opérationnelles;
- Méthodes de résolution mathématique et algorithmique;
- Optimisation des processus décisionnels;
- Développement de solutions opérationnelles;
- TPs:Utilisation de solveurs et de frameworks d'optimisation.

## EL BENANY Med Mahmoud



Professeur à la Faculté des Sciences  
et Techniques de l'Université de  
Nouakchott

Spécialité: Data science et  
Intelligence artificielle.

Support: Cours + TD/TP  
[benanymedmhd@fst.e-una.mr](mailto:benanymedmhd@fst.e-una.mr)

## Page du cours

- MAJ
- Notes de cours (mises à jour régulièrement)  
MAJ
- Séances via Teams :
  - 2h00 cours (Dr. EL Benany);
  - 2h00 TD/TP (Dr. EL Benany).
- Tous les supports sur la page web du cours.

# À propos du cours (1/2)

## Un tour d'horizon

- Concepts de base en optimisation... en RO
- .... L'importance de la Recherche Opérationnelle pour la prise de décision..

## Objectifs du cours

- Reconnaître des classes de problèmes spécifiques, notamment ceux rencontrés en recherche opérationnelle.
- Formaliser un problème d'optimisation;
- Acquérir une boîte à outils algorithmiques pour résoudre des problèmes d'optimisation.

La Recherche Opérationnelle (RO) se concentre sur **la prise de décision** en utilisant des techniques d'**optimisation**, ce qui en fait un élément essentiel dans la résolution de **problèmes complexes**.

## Un tour d'horizon

- Introduction aux principes de base de RO.
- Application de la science des données à la modélisation mathématique.
- Utilisation de la programmation linéaire et non linéaire

## Objectifs du cours

- Compréhension des concepts fondamentaux de RO.
- Capacité à formuler mathématiquement un problèmes.
- Maîtrise des méthodes exactes et approchées de résolution.
- Utilisation d'heuristiques pour résoudre des problèmes complexes.
- Application de solutions informatiques (solveurs et de frameworks).

# Relations entre RO, IA et ML/DL

## Recherche Opérationnelle (RO)

La RO fournit des méthodes d'optimisation pour résoudre des problèmes complexes de prise de décision.

## Intelligence Artificielle (IA)

L'IA utilise des techniques comme le ML/DL pour créer des systèmes capables d'apprendre à partir de données et de prendre des décisions.

## Optimisation en science des données

L'optimisation en science des données utilise des techniques d'optimisation pour résoudre des problèmes liés à l'analyse de données.

## Optimisation en RO

L'optimisation en RO fournit des outils pour modéliser et résoudre des problèmes d'optimisation complexes.

# Plan (provisoire) du cours

Comprendre et utiliser les techniques d'optimisation nécessaires à la prise de décision en Recherche Opérationnelle.

## Optimisation dans la Recherche Opérationnelle

- Optimisation combinatoire; - Méthodes exactes (branch and bound, programmation dynamique); - Méthodes approchées (métaheuristiques, algorithmes évolutionnaires).

## Optimisation avec contraintes dans la Recherche Opérationnelle

- Programmation linéaire; - Programmation en nombres entiers; - Programmation non linéaire; - Programmation dynamique; - Méthodes des contraintes pénalisées.

## Optimisation stochastique dans la Recherche Opérationnelle

- Processus stochastiques; - Programmation stochastique; - Simulation.



- ① Chapitre 1 : Généralité
  - Définition
  - historique
  - Méthodologie
- ② Chapitre 2 : Modèles utilisés
  - Programmation linéaire
  - Programmation non linéaire
  - Théorie des graphes
  - Programmation dynamique
  - Heuristiques
  - Programmation par contraintes
- ③ Chapitre 3 : Résolution informatique
  - Utilisation des solveurs
  - Utilisation des frameworks
- ④ Conclusion

- 1 Chapitre 1 : Généralité
  - Définition
  - historique
  - Méthodologie
- 2 Chapitre 2 : Modèles utilisés
  - Programmation linéaire
  - Programmation non linéaire
  - Théorie des graphes
  - Programmation dynamique
  - Heuristiques
  - Programmation par contraintes
- 3 Chapitre 3 : Résolution informatique
  - Utilisation des solveurs
  - Utilisation des frameworks
- 4 Conclusion

# Définition

## Champ d'application

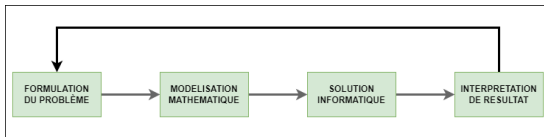
La recherche opérationnelle (RO) est au confluent de l'informatique, des mathématiques appliquées, de la gestion et du génie industriel. L'objet de cette discipline est de fournir des bases rationnelles à la prise de décisions, habituellement dans un but de contrôle ou d'optimisation (améliorer l'efficacité, diminuer les coûts, etc.). Les techniques de RO sont utilisées pour :

- Gérer les soins de santé dans les hôpitaux
- Organiser les services policiers ou ambulanciers
- Planifier l'utilisation et gérer la production d'énergie
- Planifier des systèmes de livraison ou de transport en commun
- Gérer la production, les stocks et la distribution de produits usinés
- Concevoir des systèmes de communication et des systèmes informatiques
- Établir des horaires de travail, de cours ou des calendriers sportifs
- Choisir des politiques économiques et financières

# Définition de la recherche opérationnelle

## La recherche opérationnelle :

- traite un problème pratique;
- a un objectif limité (cette application);
- nécessite une boîte à outils (algorithmes et structures des données, optimisation combinatoire, graphes, complexité, programmations linéaire et mathématique, processus stochastiques, probabilités et statistiques, méthodes multicritères....);
- est pluridisciplinaire (Mathématiques, Informatique, Economie);
- est banalisée (Programmation linéaire, Solveurs, Frameworks...);
- aide à la décision.



# Histoire de la Recherche Opérationnelle

Avant la Seconde Guerre mondiale

- IIIe siècle av. J.-C : Archimède propose des solutions pour la défense de Syracuse pendant les Guerres puniques.
- 1503 : Léonard de Vinci participe à la guerre contre Pise en utilisant ses connaissances pour développer des machines de guerre.
- XVIIe et XVIIIe siècles : Newton, Leibniz, Bernoulli et Lagrange travaillent sur l'optimisation mathématique.
- Fin du XIXe siècle : Frederik Winslow Taylor réalise des études pour maximiser le rendement des mineurs.

# Histoire de la Recherche Opérationnelle

Après la Seconde Guerre mondiale

- 1928 : Janos Von Neumann publie la "Théorie des jeux", fondement mathématique de l'optimisation linéaire.
- 1939 : Leonid Vitálievich Kantoróvich et Tjalling Charles Koopmans créent la théorie de l'optimisation linéaire.
- 1945 : George Joseph Stigler pose le problème du régime, résolu par la méthode du Simplexe.
- 1947 : Formation du projet SCOOP (Scientific Computation Of Optimum Programs) aux États-Unis pour améliorer les processus de planification à grande échelle.

# Histoire de la Recherche Opérationnelle

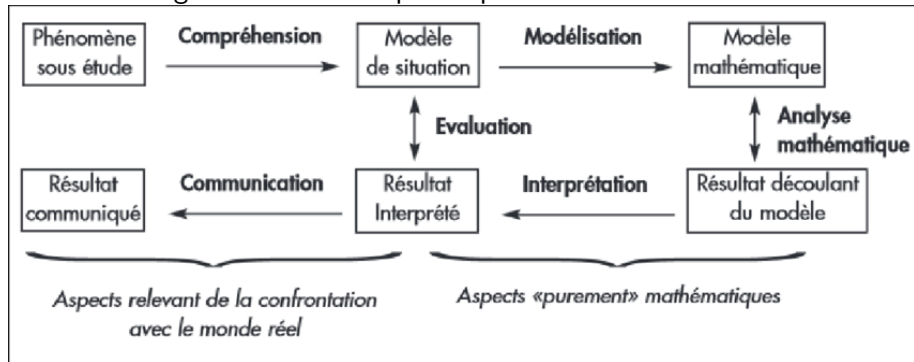
## Période moderne

- 1952 : Premier résultat obtenu avec l'ordinateur SEAC pour résoudre un problème, augmentant considérablement l'efficacité de la force aérienne.
- Années 50 et 60 : Expansion de l'intérêt et du développement de la Recherche Opérationnelle dans le commerce et l'industrie.
- Exemple : Calcul du projet optimal de transport de sable de construction à Moscou, réduisant les coûts de 11
- Applications : Agriculture, transport, localisation, distribution du personnel, réseaux, files d'attente, graphes, etc.

# Modélisation Mathématique

La recherche opérationnelle (RO) est une discipline qui vise à améliorer la prise de décision en s'appuyant sur des méthodes scientifiques et mathématiques.

Diagramme schématique du processus de modélisation





## Problème

Un fabricant de produits alimentaires doit déterminer la quantité de chaque produit à fabriquer chaque mois pour maximiser son profit total. Il dispose de plusieurs usines avec des capacités de production différentes et il doit respecter certaines contraintes, telles que la disponibilité des matières premières, la main-d'œuvre et le stockage.

- ❶ Définition du problème
  - Identifier le problème à résoudre et ses objectifs.
  - Déterminer les variables de décision
  - Définir la fonction objectif à optimiser.
  - Définir les contraintes.

# Étapes de la méthodologie de la RO (suite)

## Définition du problème:

- ① Variables de décision: Quantité de chaque produit à fabriquer
- ② Variables de décision: Quantité de chaque produit à fabriquer chaque mois.
- ③ Fonction objectif: Maximiser le profit total.
- ④ Contraintes:
  - Capacité de production de chaque usine.
  - Disponibilité des matières premières.
  - Main-d'œuvre disponible.
  - Capacité de stockage.

## ② Construction du modèle

- Choisir le type de modèle le plus adapté (linéaire, non linéaire, etc.).
- Formuler le modèle mathématique du problème.
- Valider la cohérence et la précision du modèle.

## ③ Résolution du modèle

- Choisir la méthode de résolution la plus appropriée.
- Utiliser un logiciel de RO pour résoudre le modèle.
- Analyser les résultats de la résolution.

## 4 Analyse des résultats

- Interpréter les solutions obtenues.
- Identifier les points forts et les points faibles des solutions.
- Evaluer la sensibilité des solutions aux changements de paramètres.

Pour modéliser ce problème d'optimisation, vous pouvez définir les éléments suivants :

Variables de décision :  $x_i$  représente la quantité du produit  $i$  à fabriquer chaque mois.

# Modélisation (suite)

Fonction objectif : Maximiser le profit total, qui peut être défini comme la somme des profits de chaque produit fabriqué. Si  $p_i$  est le profit unitaire du produit  $i$ , alors la fonction objectif peut être exprimée comme :

$$\text{Maximiser } \sum_i p_i \cdot x_i$$

Contraintes :

- Capacité de production de chaque usine : La somme des quantités de chaque produit fabriqué dans chaque usine ne doit pas dépasser la capacité de production de l'usine.
- Disponibilité des matières premières : La quantité de matières premières nécessaires pour fabriquer chaque produit ne doit pas dépasser la disponibilité des matières premières.
- Main-d'œuvre disponible : La quantité de main-d'œuvre nécessaire pour fabriquer chaque produit ne doit pas dépasser la main-d'œuvre disponible.

# Modélisation (suite)

Contraintes (suite) :

- Capacité de stockage : La quantité totale des produits stockés ne doit pas dépasser la capacité de stockage.

Mathématiquement, ces contraintes peuvent être représentées comme suit :

Capacité de production : 
$$\sum_i a_{ij} \cdot x_i \leq C_j, \quad \forall j$$

Disponibilité des matières premières : 
$$\sum_i b_i \cdot x_i \leq B$$

Main-d'œuvre disponible : 
$$\sum_i c_i \cdot x_i \leq D$$

Capacité de stockage : 
$$\sum_i x_i \leq S$$

Positivité : 
$$x_i \geq 0, \quad \forall i$$

## 5 Implémentation de la solution

- Définir un plan d'implémentation de la solution.
- Mettre en place la solution et suivre son évolution.
- Mesurer l'impact de la solution sur les performances du système.



## Exemple : Maximisation du profit (Partie 1)

Une entreprise fabrique deux types de produits alimentaires, notés  $P_1$  et  $P_2$ . Elle dispose de trois usines avec des capacités de production respectives de 10, 20 et 15 unités. Les produits nécessitent respectivement 7 et 9 unités de matières premières par unité produite, ainsi que 5 et 8 unités de main-d'œuvre. La capacité de stockage de l'entreprise est de 50 unités. Le profit unitaire pour chaque unité vendue est de 10 pour  $P_1$  et 12 pour  $P_2$ .

# Correction : Les Données du problème (Partie 2)

L'entreprise cherche à déterminer la quantité de chaque produit à fabriquer chaque mois pour maximiser son profit total, tout en respectant les contraintes de capacité de production, disponibilité des matières premières, main-d'œuvre disponible et capacité de stockage.

## Données:

- - Usines :  $usines = \{1, 2, 3\}$
- - Produits :  $produits = \{1, 2\}$
- - Capacité des usines :  $capacite\_usine = [10, 20, 15]$
- - Matières premières :  $matieres\_premieres = [7, 9]$
- - Main-d'œuvre :  $main\_oeuvre = [5, 8]$
- - Capacité de stockage :  $capacite\_stockage = 50$
- - Profit unitaire :  $profit\_unitaire = [10, 12]$

# Corection :Modélisation (Partie 3)

Variables de décision :  $x_i$ , la quantité du produit  $i$  à fabriquer chaque mois.

Fonction objectif : Maximiser le profit total :

$$\text{Maximiser } \sum_{i \in \text{produits}} \text{profit\_unitaire}[i] \cdot x_i$$

Contraintes :

1. Capacité de production de chaque usine :

$$\sum_{i \in \text{produits}} a_{ij} \cdot x_i \leq \text{capacite\_usine}[j], \quad \forall j \in \text{usines}$$

2. Disponibilité des matières premières :

$$\sum_{i \in \text{produits}} \text{matieres\_premieres}[i] \cdot x_i \leq B$$

3. Main-d'œuvre disponible :

$$\sum_{i \in \text{produits}} \text{main\_oeuvre}[i] \cdot x_i \leq D$$

4. Capacité de stockage :

$$\sum_{i \in \text{produits}} x_i \leq \text{capacite\_stockage}$$

avec  $x_i \geq 0, \forall i \in \text{produits}$ .

# Correction : Résolution et la prise de décision (Partie 4)

1. Quelle quantité de chaque produit l'entreprise devrait-elle fabriquer chaque mois pour maximiser son profit total ?
2. Quel est le profit total réalisé par l'entreprise dans ce cas optimal ?

## Processus de prise de décision

Pour répondre à ces questions, on va utiliser deux implémentations différentes:

- Le solveur Gurobi pour résoudre le modèle Pyomo et obtenir la solution optimale.
- Le framework Pyomo pour modéliser le problème et déterminer les quantités de chaque produit à fabriquer pour maximiser le profit total.
- Le solveur glpk pour résoudre le modèle Pyomo et obtenir la solution optimale.
- Calculer le profit total réalisé par l'entreprise dans ce cas optimal.

# Implémentation: Solveur Gurobi en Python (Partie 1)

```
import gurobipy as gp
from gurobipy import GRB

# Problem data
usines = range(3)
produits = range(2)
capacite_usine = [10, 20, 15]
matieres_premieres = [7, 9]
main_oeuvre = [5, 8]
capacite_stockage = 50
profit_unitaire = [10, 12]
```

# Implémentation avec Gurobi en Python (Part 2)

*# Create the model*

```
m = gp.Model(" Profit - Maximization")
```

*# Decision variables*

```
x = m.addVars(produits , name=" x" ,  
              vtype=GRB.CONTINUOUS)
```

*# Objective function*

```
m.setObjective(gp.quicksum(  
    profit_unitaire[i] * x[i] for i in produits),  
               GRB.MAXIMIZE)
```

# Implémentation avec Gurobi en Python (Part 3)

```
# Constraints
m.addConstrs(
    (gp.quicksum(a[i, j] * x[i] for i in produits)
    <= capacite_usine[j] for j in usines),
    name="capacite_usine")
m.addConstr(
    gp.quicksum(matieres_premieres[i] * x[i]
    for i in produits)
    <= B, name="matieres_premieres")
m.addConstr(
    gp.quicksum(main_oeuvre[i] * x[i]
    for i in produits) <= D, name="main_oeuvre")
m.addConstr(
    gp.quicksum(x[i] for i in produits)
    <= capacite_stockage, name="capacite_stockage")
```



# Implémentation avec Gurobi en Python (Part 4)

```
# Solve the model
m.optimize()

# Display the solution
if m.status == GRB.OPTIMAL:
    for i in produits:
        print(f"Quantity of product {i}+-1}
        -----to manufacture: {x[i].x}")
        print(f"Total profit: {m.objVal}")
else:
    print("No optimal solution found.")
```

# Implémentation: Framework Pyomo (Part 1)

*#First, make sure you have Pyomo installed*  
`pip install pyomo`

*#Then, you can create and solve the optimization model*  
`from pyomo.environ import *`

*# Problem data*  
`usines = range(3)`  
`produits = range(2)`  
`capacite_usine = [10, 20, 15]`  
`matieres_premieres = [7, 9]`  
`main_oeuvre = [5, 8]`  
`capacite_stockage = 50`  
`profit_unitaire = [10, 12]`

# Implémentation: Framework Pyomo (Part 2)

```
# Create a Pyomo model  
model = ConcreteModel()
```

```
# Decision variables  
model.x = Var(produits, domain=NonNegativeReals)
```

```
# Objective function  
model.obj = Objective(expr=sum  
    (profit_unitaire[i] * model.x[i] for i in produits),  
    sense=maximize)
```

## Implémentation: Framework Pyomo (Part 3)

```
# Constraints
model.capacity_constraints = ConstraintList()
for j in usines:
    model.capacity_constraints.add(
        sum(a[i, j] * model.x[i] for i in produits)
        <= capacite_usine[j])
model.matières_premières_constraint =
    Constraint(expr=sum(matières_premières[i] *
        model.x[i] for i in produits) <= B)
model.main_oeuvre_constraint =
    Constraint(expr=sum(main_oeuvre[i] *
        model.x[i] for i in produits) <= D)
model.capacity_stockage_constraint =
    Constraint(expr=sum(model.x[i]
        for i in produits) <= capacite_stockage)
```

## Implémentation: Framework Pyomo (Part 4)

```
# Solve the model
solver = SolverFactory('glpk')
solver.solve(model)

# Display the solution
if 'ok' == str(solver.status):
    for i in produits:
        print(f"Quantity-of-product-{i}-1}
-----to-manufacture:{model.x[i].value}")
        print(f"Total-profit:{model.obj()}")
else:
    print("No-optimal-solution-found.")
```

# Aperçu des prochains chapitres

Dans les prochains chapitres, nous explorerons en détail les concepts suivants :

- La **programmation linéaire** et son application à la résolution de problèmes d'optimisation.
- Les méthodes de **programmation non linéaire** pour résoudre des problèmes plus complexes.
- L'utilisation de la **théorie des graphes** pour modéliser et résoudre divers problèmes pratiques.
- La **programmation dynamique** comme méthode puissante pour résoudre des problèmes récurrents.
- Les **heuristiques** et leur rôle dans la recherche de solutions efficaces pour des problèmes difficiles.
- La **programmation par contraintes** pour résoudre des problèmes où les contraintes sont spécifiées avant les variables.
- L'utilisation des **solveurs** et des **frameworks** pour la résolution informatique des problèmes d'optimisation.

- ① Chapitre 1 : Généralité
  - Définition
  - historique
  - Méthodologie
- ② Chapitre 2 : Modèles utilisés
  - Programmation linéaire
  - Programmation non linéaire
  - Théorie des graphes
  - Programmation dynamique
  - Heuristiques
  - Programmation par contraintes
- ③ Chapitre 3 : Résolution informatique
  - Utilisation des solveurs
  - Utilisation des frameworks
- ④ Conclusion

- 1 Chapitre 1 : Généralité
  - Définition
  - historique
  - Méthodologie
- 2 Chapitre 2 : Modèles utilisés
  - Programmation linéaire
  - Programmation non linéaire
  - Théorie des graphes
  - Programmation dynamique
  - Heuristiques
  - Programmation par contraintes
- 3 Chapitre 3 : Résolution informatique
  - Utilisation des solveurs
  - Utilisation des frameworks
- 4 Conclusion



- 1 Chapitre 1 : Généralité
  - Définition
  - historique
  - Méthodologie
- 2 Chapitre 2 : Modèles utilisés
  - Programmation linéaire
  - Programmation non linéaire
  - Théorie des graphes
  - Programmation dynamique
  - Heuristiques
  - Programmation par contraintes
- 3 Chapitre 3 : Résolution informatique
  - Utilisation des solveurs
  - Utilisation des frameworks
- 4 Conclusion