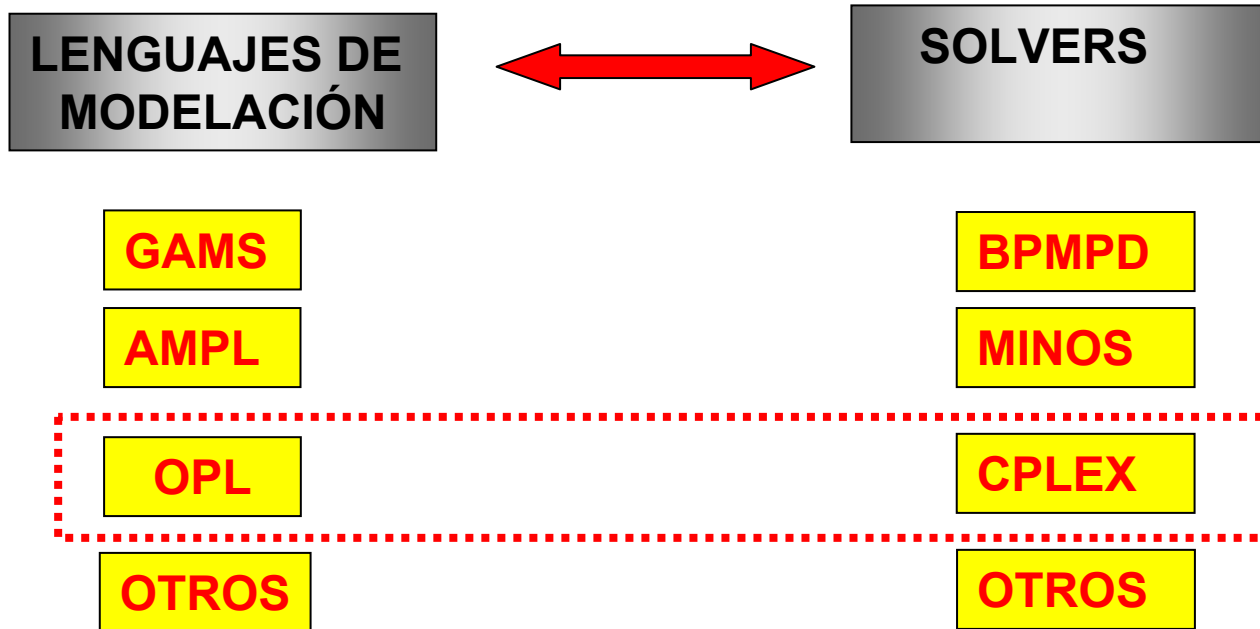


Introducción

MODELACIÓN Y RESOLUCIÓN DE MODELOS DE PROGRAMACIÓN MATEMÁTICA USANDO CPLEX Y OPL STUDIO

DESARROLLO DE UNA APLICACIÓN DE OPTIMIZACIÓN





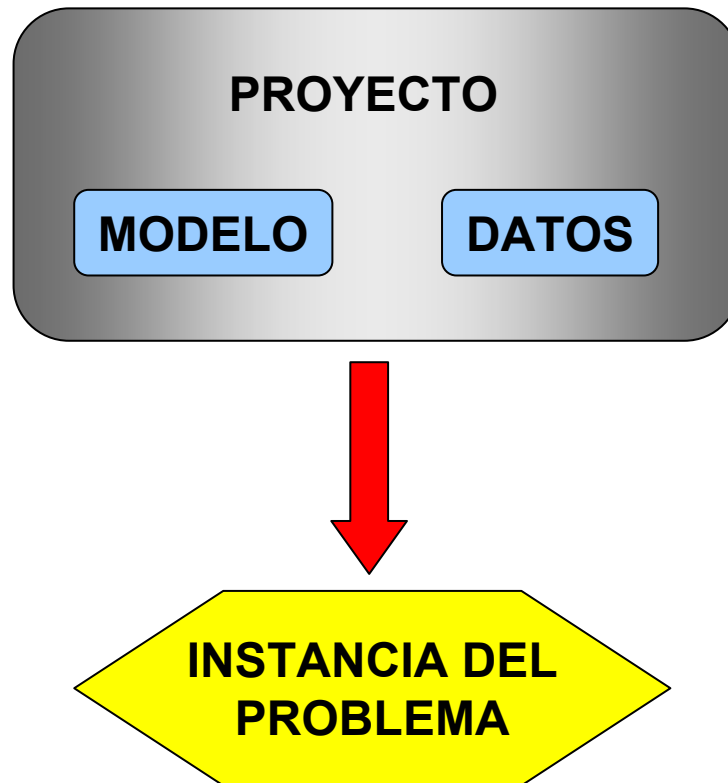
Lenguajes de programación
para representar
problemas de optimización

Programas que implementan
algoritmos de optimización lineal,
no lineal (Simplex, punto interior,
B&B, etc)

- **OPL Optimization Programming Language**
 - Lenguaje para representar problemas de optimización
 - Facilidades para la organización de los datos
 - Soporta programación lineal y entera, constraint programming, y constraint-based scheduling, modelos quadraticos.
 - Conección con Bases de Datos y Excel.
 - OPLScript para resolución iterativa, optimización híbrida y programación de heurísticas.
- **Interfaz Gráfica**
 - Amigable (ej: editor con keywords de colores)
 - Visualización de Datos y Solución
 - Menús/botones para controlar la optimización
 - Online help

- API's para conectar modelos OPL y OPL scripts
 - C++
 - Microsoft COM
 - **Visual Basic**
 - **Visual Basic for Applications (Excel, Access, etc.)**
 - **Microsoft .NET**
 - C#
 - VB.NET
 - Managed C++
 - Java
 - Web
 - **ASP, JSP**
- Links con ILOG CPLEX, Solver y/o Scheduler

- Un modelo es un abstracción del problema independiente de los datos
- OPL permite escribir la representación matemática de un modelo independiente de los datos



$$\min c^t x$$

← **Función Objetivo**

$$s.t \ Ax = b$$

← **Restricciones**

Cotas Inferiores →

$$l \leq x \leq u$$

← **Cotas Superiores**

↑
Variables

Ejemplo Modelo: Fábrica que produce un conjunto de productos

- Satisfacción de demanda por producto
- Solo costos variables por producto
- Utilización de recursos por productos
- Capacidad máxima por recurso
- Posibilidad de importar productos a costo unitario por producto

Conjuntos :

p : Productos
 r : Recursos

Función Objetivo : Minimizar Costos

$$\min \sum_p ic_p ix_p + \sum_p oc_p ox_p$$

Parametros

u_{pr} : utilización de producto p por recurso r
 c_r : capacidad de recurso r
 d_p : demanda por producto p
 ic_p : costo de fabricación propia de producto p
 oc_p : costo de importación de producto p

Variables

ix_p : fabricación propia de producto p
 ox_p : importación de producto p

Restricciones

1. Capacidad de producción

$$\sum_p u_{pr} ix_p \leq c_r \quad \forall r$$

2. Satisfacción de Demanda

$$ix_p + ox_p \geq d_p \quad \forall p$$

2. Naturaleza de las Variables

$$ix_p, ox_p \geq 0 \quad \forall p$$

Conjuntos :

p : Productos
 r : Recursos

```
enum Products ...;  
enum Resources ...;
```

Parametros

u_{pr} : utilización de producto p por recurso r
 c_r : capacidad de recurso r
 d_p : demanda por producto p
 ic_p : costo de fabricación propia de producto p
 oc_p : costo de importación de producto p

```
float+ consumption[Products, Resources] = ...;  
float+ capacity[Resources] = ...;  
float+ demand[Products] = ...;  
float+ insideCost[Products] = ...;  
float+ outsideCost[Products] = ...;
```

Variables

ix_p : fabricación propia de producto p
 ox_p : importación de producto p

```
var float+ inside[Products];  
var float+ outside[Products];
```

Función Objetivo : Minimizar Costos

$$\min \sum_p ic_p ix_p + \sum_p oc_p ox_p$$

```
minimize  
    sum(p in Products) (insideCost[p]*inside[p] + outsideCost[p]*outside[p])
```

Restricciones

1. Capacidad de producción

$$\sum_p u_{pr} ix_p \leq c_p \quad \forall r$$

2. Satisfacción de Demanda

$$ix_p + ox_p \leq c_p \quad \forall p$$

2. Naturaleza de las Variables

$$ix_p, ox_p \geq 0 \quad \forall p$$

```
var float+ inside[Products];  
var float+ outside[Products];
```

Datos en production.dat

```
subject to {  
    forall(r in Resources)  
        sum(p in Products) consumption[p, r] * inside[p] <= capacity[r];  
  
    forall(p in Products)  
        inside[p] + outside[p] >= demand[p];  
};
```

Trabajo con Software xsteel.mod