ASIGNATURA: TECNICAS DE OPTIMIZACIÓN

PRÁCTICA: OPTIMIZACIÓN CON GOOGLE OR-TOOLS

SESIONES: 1

SOFTWARE: OR-TOOLS (+ PYTHON)

Google OR-Tools[1] es un paquete de software portable de **código abierto** para resolución de problemas de **optimización**[2]**.** Así mismo, cuenta con metaheurísticas que buscan encontrar la mejor solución a un problema entre un conjunto de posibles soluciones. OR-Tools es una herramienta potente, diseñada para abordar los problemas más difíciles en el enrutamiento de vehículos, problemas de redes, la programación matemática con variables continuas y discretas.

Una de las mayores ventajas que presenta Google OR-Tools es que se trata de un software de código abierto, de manera que su modelo de desarrollo se basa en la colaboración. Esto representa un beneficio potencial, ya que cientos de personas están diariamente formulando modelos de optimización en diversos lenguajes de programación.

Otra de las ventajas potenciales de Google OR-Tools consiste en que, a pesar de estar desarrollado en lenguaje C++, permite la formulación de modelos matemáticos de problemas reales en múltiples lenguajes de programación, y del mismo modo, cuenta con múltiples solucionadores específicos para resolverlos, algunos comerciales y algunos de código abierto.

	Lenguajes de programación			
OR-Tools	Python	C++	Java	C#
Compatibilidad	✓	✓	✓	✓

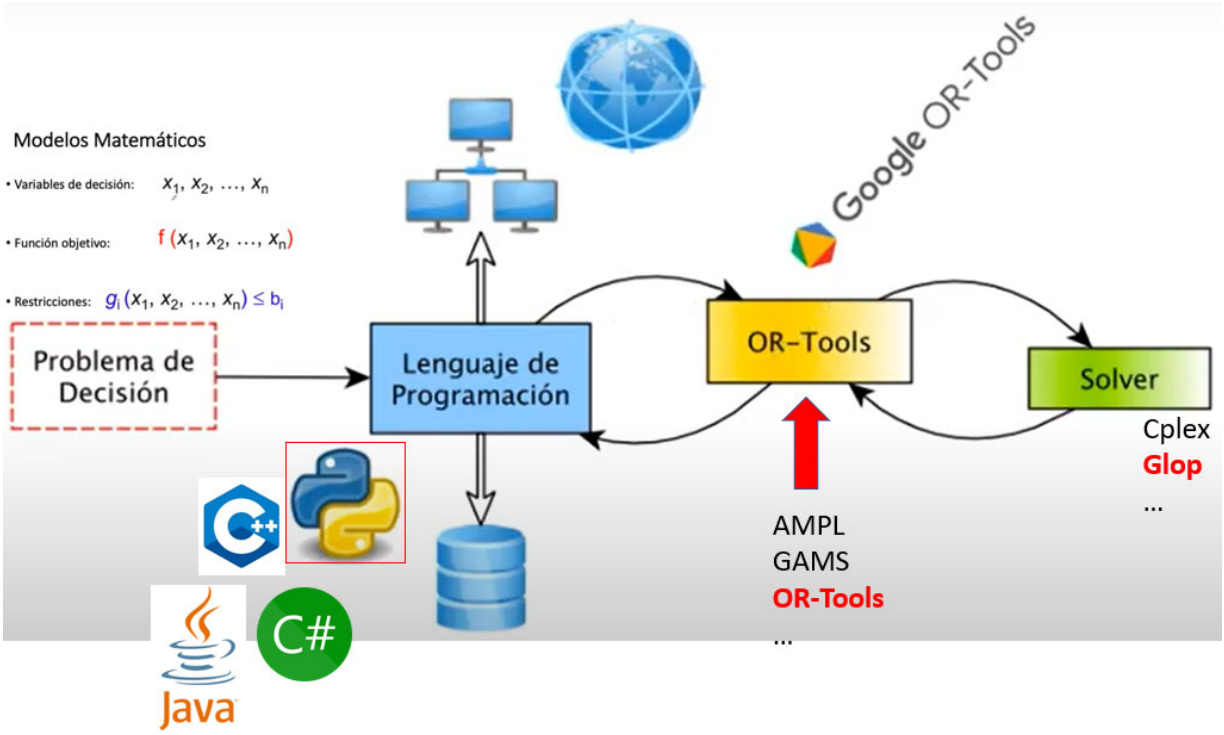
La obtención, el procesamiento y la disposición de un volumen considerable de datos, su convergencia en un modelo de optimización y su eventual integración hacia un solucionador, son procesos complejos, que implican a menudo el uso de herramientas poco compatibles.

Google OR-Tools, al permitir el modelamiento en diversos lenguajes de programación, ofrece un sinfín de posibilidades de integración entre herramientas de obtención y tratamiento de información; así mismo, ofrece posibilidades ilimitadas de desarrollo de soluciones, como por ejemplo interfaces de usuario amigables, o integración con sistemas de integración de recursos, app's, plataformas webs, dispositivos con un sistema de conexión a internet de las cosas (IoT), y mucho más.

OR-Tools es compatible con sistemas Linux, Mac y Windows; sin embargo, su instalación varía de acuerdo al lenguaje de programación que se desea utilizar (https://developers.google.com/optimization/install).

1 https://developers.google.com/optimization
2 https://acortar.link/HSa09

Así pues, utilizando OR-Tools es posible desarrollar sistemas de optimización cuya arquitectura se muestra en la siguiente figura:



En esta práctica utilizaremos **Python** como lenguaje de programación que llamará a **OR-Tools** para obtener la solución óptima de un modelo matemático utilizando el optimizador **SCIP** para programación lineal entera o **GLOP** para modelos de programación lineal continua.

Antes de empezar

Lo primero que debemos hacer es instalar Python. Para ello, si trabajamos en Windows, debemos ir a **Microsoft Store** y descargar e instalar la última versión de **Python**.

También se puede descargar e instalar desde https://www.python.org/downloads/

Todo habrá ido bien si al utilizar la consola del sistema (símbolo del sistema) o CMD, y ejecutar Python, aparece el siguiente mensaje:

```
C:\>python
Python 3.9.5 (tags/v3.9.5:0a7dcbd, May  3 2021, 17:27:52) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> _
```

A continuación, desde la consola del sistema (símbolo del sistema) o CMD debemos instalar el gestor de paquetes Phyton-pip[3]:

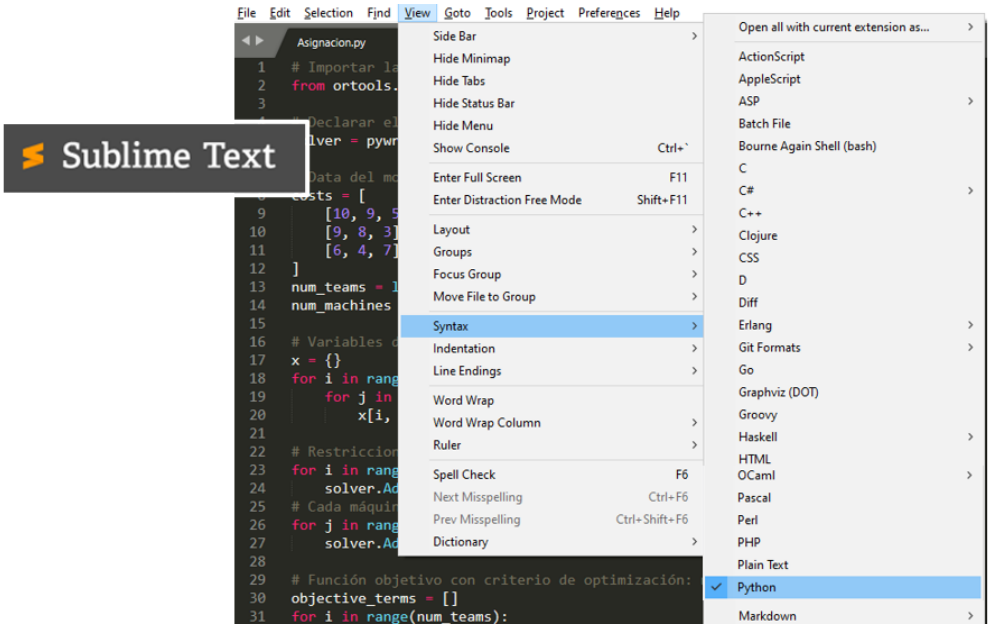
```
C:\python -m pip install --upgrade --user ortools
Collecting ortools
  Downloading ortools-9.0.9048-cp39-cp39-win_amd64.whl (52.2 MB)
     |████████████████████████████████| 52.2 MB 3.3 MB/s
Collecting absl-py>=0.11
  Downloading absl_py-0.12.0-py3-none-any.whl (129 kB)
     |████████████████████████████████| 129 kB 6.4 MB/s
Collecting protobuf>=3.15.8
  Downloading protobuf-3.15.8-py2.py3-none-any.whl (173 kB)
     |████████████████████████████████| 173 kB ...
Collecting six
  Downloading six-1.15.0-py2.py3-none-any.whl (10 kB)
Installing collected packages: six, absl-py, protobuf, ortools
Successfully installed absl-py-0.12.0 ortools-9.0.9048 protobuf-3.1
```

Y actualizamos **Phyton-pip**:

```
C:\Python -m pip install --upgrade pip
Collecting pip
  Downloading pip-21.1.1-py3-none-any.whl (1.5 MB)
     |████████████████████████████████| 1.5 MB 6.8 MB/s
Installing collected packages: pip
  WARNING: The scripts pip.exe, pip3.9.exe and pip3.exe ar
ftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local
  Consider adding this directory to PATH or, if you prefer
Successfully installed pip-21.1.1
```

Por último, es conveniente disponer de un editor de código de lenguaje de programación para poder verificar la sintaxis del código Python que se introduzca. Una posibilidad es usar **Sublime Text** (https://www.sublimetext.com/) y configurarlo para que detecte la sintaxis de Python:

3 En caso de **Mac** o **Linux**, consultar https://developers.google.com/optimization/install/python



¿Cómo se hace?

Los pasos que vamos a seguir para obtener la solución óptima de un modelo matemático con OR-Tools son los siguientes:

1. Plantear **MODELO MATEMÁTICO** del problema
2. Importar las **LIBRERÍAS** necesarias
3. Declarar el **SOLVER**
4. Crear las **VARIABLES**
5. Definir las **RESTRICCIONES**
6. Definir la **FUNCIÓN OBJETIVO**
7. Llamar al **SOLVER**
8. Mostrar los **RESULTADOS**

Plantear MODELO MATEMÁTICO del problema

Vamos a utilizar como ejemplo el problema de producción de placas base tipo 1 y placas base tipo 2 con el que trabajamos en clase.

El problema consiste en determinar los valores de las variables:

$X1 \geq 0$ y $X2 \geq 0$

que optimicen (maximicen en este caso) la función objetivo:

Max $3\,X1 + 5\,X2$

y verifiquen las restricciones:

$X1 \leq 4$

$2\,X2 \leq 12$

$3\,X1 + 2\,X2 \leq 18$

Para los siguientes apartados, crearemos un fichero con extensión **.PY** (extensión que utilizan los ficheros de Python) en el que incluiremos las sentencias que se muestran.

Importar las LIBRERÍAS necesarias

```
from ortools.linear_solver import pywraplp
```

Declarar el SOLVER[4]

Como consideraremos que el modelo es de programación lineal continua, utilizaremos el Solver GLOP (Google Linear Optimizer)

```
# Create the linear solver with the GLOP backend.
solver = pywraplp.Solver.CreateSolver('GLOP')
```

Crear las VARIABLES[5]

```
x1 = solver.NumVar(0, solver.infinity(), 'x1')
x2 = solver.NumVar(0, solver.infinity(), 'x2')

print('Número de variables =', solver.NumVariables())
```

Definir las RESTRICCIONES

```
# Restricción 1: x1<=1.
solver.Add(1 * x1 + 0 * x2 <= 4)

# Restricción 2: 2 x2 <= 12.
solver.Add(0 * x1 + 2 * x2 <= 12)

# Restricción 3: 3 * x1 + 2 * x2 <= 18.
solver.Add(3 * x1 + 2 * x2 <= 18)

print('Número de restricciones =', solver.NumConstraints())
```

Definir la FUNCIÓN OBJETIVO

```
# Función objetivo (max): 3x+y
solver.Maximize(3 * x1 + 5 * x2)
```

---

4 En caso de que el modelo sea de **programación lineal entera** el Solver es **SCIP** (Solving Constraint Integer Programs)

```
# Create the linear solver with the SCIP backend.
solver = pywraplp.Solver.CreateSolver('SCIP')
```

5 Si quisiéramos indicar que la variable x1 es una variable de naturaleza discreta, deberemos indicar:

x1 = solver.**Int**Var(0, solver.infinity(), 'x1')

Y si se desea indicar que la variable x1 es de naturaleza binaria, deberemos indicar:

x1 = solver.**BoolVar**('x1')

Llamar al SOLVER

```
status=solver.Solve()
```

Mostrar los RESULTADOS

```
if status == solver.ABNORMAL :
    print("Se ha producido un error mientras se ejecutaba el solver")
elif status == solver.FEASIBLE :
    print("Se ha encontrado una SOLUCIÓN FACTIBLE")
    print('Valor de la Función Objetivo =', objetivo.Value())
    print('Valor óptimo de [x1] =', x1.solution_value())
    print('Valor óptimo de [x2] =', x2.solution_value())
elif status == solver.INFEASIBLE :
    print("El problema NO tiene SOLUCIÓN POSIBLE")
elif status == solver.NOT_SOLVED :
    print("No se ha podido encontrar NINGUNA SOLUCIÓN en el TIEMPO proporcionado")
elif status == solver.OPTIMAL :
    print("Se ha encontrado la SOLUCIÓN ÓPTIMA")
    print('Valor de la Función Objetivo =', solver.Objective().Value())
    print('Valor óptimo de [x1] =', x1.solution_value())
    print('Valor óptimo de [x2] =', x2.solution_value())
elif status == solver.UNBOUNDED :
    print("Solución no acotada por las restricciones")
else :
    print("Código de error desconocido")
```

Una vez indicadas todas las instrucciones en el fichero, el siguiente paso es ejecutar el código, y para eso podemos utilizar la consola del sistema (símbolo del sistema) o CMD. En ella, debemos dirigirnos hacia el directorio en el cual se encuentre nuestro archivo **.py** y ejecutarlo de la siguiente manera:

python Nombre del archivo.py

Obteniéndose la siguiente salida:

```
Número de variables = 2
Número de restricciones = 3
Se ha encontrado la SOLUCIÓN ÓPTIMA
Valor de la Función Objetivo = 36.0
Valor óptimo de [x1] = 2.0
Valor óptimo de [x2] = 6.0

C:\>_
```

Hazlo tú mismo

Sigue los 8 pasos descritos en el apartado anterior para obtener la solución óptima del siguiente problema:

En una explotación agropecuaria se está modernizando el sistema de regadío de un riego de superficie a un moderno sistema de riego por aspersión de tipo estacionario fijo enterrado.

La parcela es muy irregular y es necesario instalar un número elevado de aspersores. La parcela se ha dividido en 6 sectores diferentes y existen 8 posibles puntos donde se puede instalar un aspersor. Existen dos tipos de aspersores, el de tipo medio, con un coste de 300 €/unidad y el de tipo grande con un coste de 490 €/unidad. Dependiendo del tipo de aspersor que se instale, más o menos sectores de la parcela quedarán cubiertos.

La tabla siguiente nos indica los sectores de la parcela que quedan cubiertos según se instale un aspersor medio o grande en cada uno de los 8 posibles puntos:

Posibles puntos para los aspersores	Sectores cubiertos con aspersor medio		Sectores cubiertos con aspersor grande		
Punto 1	1		1	2	
Punto 2	4		2	4	5
Punto 3	2	5	2	5	
Punto 4	4	5	2	4	5
Punto 5	2		2	3	5
Punto 6	5	6	3	5	6
Punto 7	3	6	3	6	
Punto 8	6		3	6	

En un punto dado sólo se puede instalar un aspersor, sea de tipo mediano o de tipo grande.

Adicionalmente, debido a restricciones del caudal máximo de agua se deben tener en cuenta las siguientes condiciones:

- Tan sólo se pueden instalar como mucho 2 aspersores de tipo grande.
- Si se instala un aspersor de tipo medio en el punto 3 también se debe instalar un aspersor del mismo tipo en el punto 6.
- La empresa proveedora de los aspersores aplicaría un descuento de 300 euros en la factura total en el caso de que se instalen al menos 3 aspersores de tipo medio.

Plantea un **modelo de programación lineal**, (explicando con claridad variables, función objetivo y restricciones), que permita determinar los puntos de aspersión a ocupar y con qué tipos de aspersores, de manera que se minimice el coste total de instalación y se tengan en cuenta todas las condiciones indicadas.