Departamento de Engenharia Informática e de Sistemas

Metodologias de Otimização e Apoio à Decisão

Resolução de problemas de PL em Python

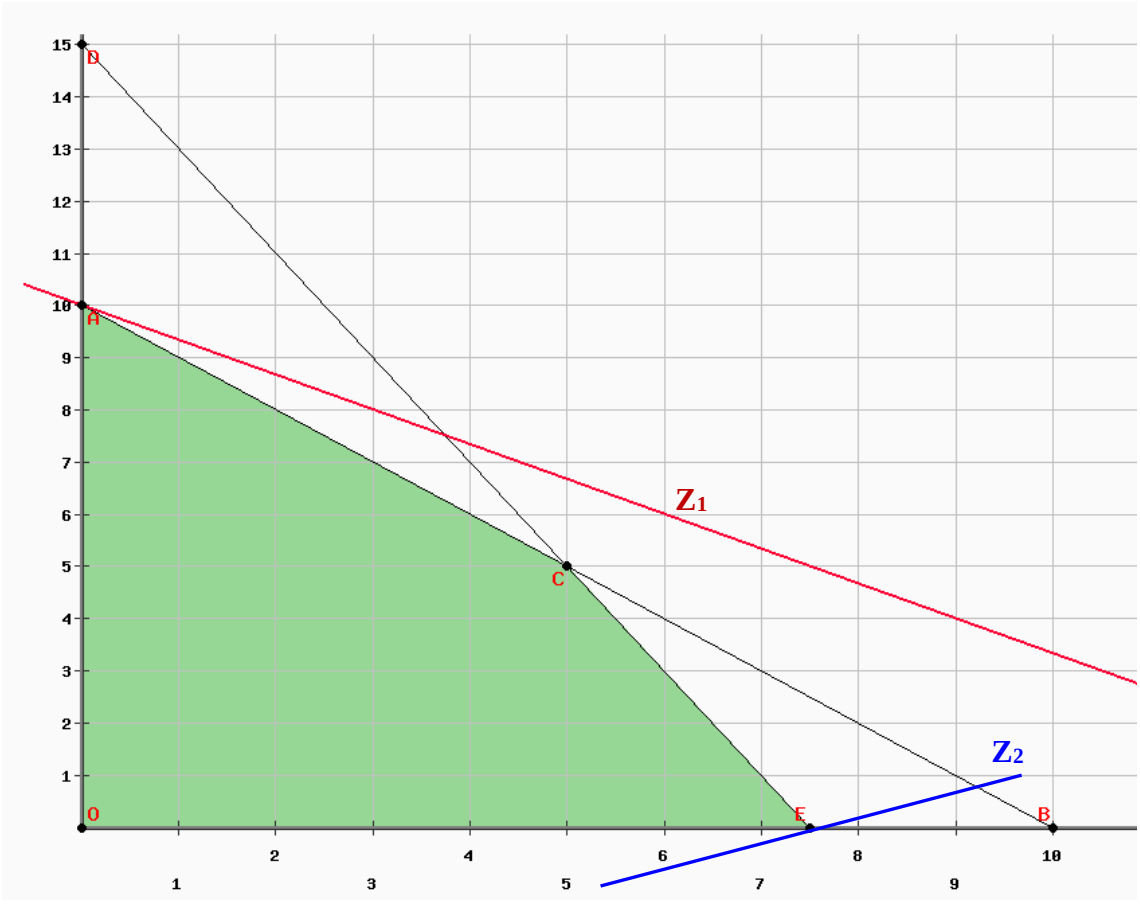
- Parte VI -

Na continuidade do documento anterior, foi desenvolvida uma nova abordagem que usa a soma ponderada das funções objetivo proposta pela 2ª abordagem, para detetar os pontos extremos da região eficiente. A partir daí, calcula as imagens desses pontos no espaço das funções objetivo, ou seja, os valores correspondentes de z_1 e z_2, obtendo, desse modo, os pontos extremos da região não dominada. Além de mostrar as soluções ideal e anti-ideal, no final apresenta dois gráficos que representam as duas regiões mencionadas: eficiente e não dominada. Desta forma, a resolução dos problemas torna-se mais próxima da que é efetuada nas aulas teóricas e teórico-práticas.

Seguidamente, mostra-se a implementação desta nova abordagem na resolução do **EXEMPLO 6**, cujo modelo matemático é o seguinte:

$$\begin{aligned}
&\text{Max } z_1 = 2\,x_1 + 3\,x_2 \\
&\text{Max } z_2 = 4\,x_1 - 2\,x_2 \\
&\text{sujeito a} \\
&\quad x_1 + x_2 \leq 10 \\
&\quad 2\,x_1 + x_2 \leq 15 \\
&\quad x_1 \geq 0,\ x_2 \geq 0
\end{aligned}$$

A representação gráfica deste modelo é:



De acordo com o gráfico anterior, as funções objetivo z_1 e z_2 atingem o seu valor ótimo nos pontos A e E, respetivamente. Recorrendo à técnica dos cones de dominância, seria possível concluir que a região eficiente, ou conjunto das soluções eficientes, é:

$$K_{EF} = [A,C] \cup [C,E]$$

O código seguinte resolve este problema com a nova abordagem:

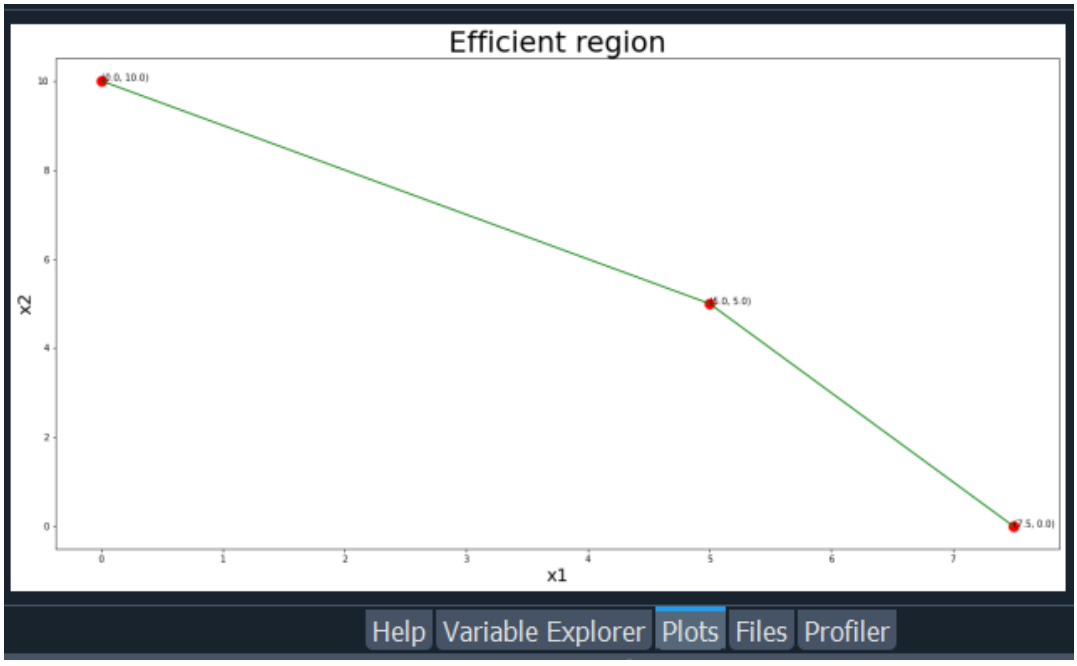
```
"""
1o Problema de PLMO / 1st MOLP problem
"""
# Importar biblioteca PuLP / Import PuLP library
from pulp import *

# Importar sub-modulo Pyplot da biblioteca Matplotlib
# Import sub-module Pyplot from libray Matplotlib
import matplotlib.pyplot as plt

# Inicializar duas listas para guardar pontos extremos (PE) da região eficiente
# e respetivos valores de z1 e z2 / Initialize two lists to save extreme points
# (EP) of efficient region and respective values of z1 and z2
X=[]
Z=[]
# Criar variáveis de decisão / Create decision variables
x1=LpVariable("x1",lowBound=0)
x2=LpVariable("x2",lowBound=0)

# Para determinar PE usa abordagem da soma ponderada das duas funções objetivo
# To find EP use approach of the weighted sum of two objective functions
# Definir incremento de  alpha / Define step-size for alpha
step = 0.1
# Numero total de amostras / Total number of samples
n = int(1/step)
# Iterar para valores de alpha (peso) entre 0 e 1
# Iterate through alpha (weight) values from 0 to 1
for i in range(0,n+1):
        # Calcula valor de alpha / Calculate alpha value
        alpha=i/n
        # Criar novo modelo / Create new model
        model=LpProblem("PLMO_1/MOLP_1",LpMaximize)
        # Adicionar funcao objetivo z = alpha*z1 + (1-alpha)*z2
        # Add objective function z = alpha*z1 + (1-alpha)*z2
        model += alpha*(2*x1+3*x2)+(1-alpha)*(4*x1-2*x2)
        # Adicionar restricoes / Add constraints
        model += x1 + x2 <= 10
        model += 2*x1 + x2 <= 15
        # Resolver modelo / Solve model
        model.solve()

        # Guardar cada nova solução determinada pois é PE da região eficiente
        # Save each new solution found since it is an EP of efficient region
        if X.count([value(x1),value(x2)]) == 0:
                X=X+[[value(x1),value(x2)]]

# Mostra PE da região eficiente / Show EP of efficient region
print("Extreme points of the efficient region:\n",X)
# Calcula valores correspondentes de z1 e z2
# Find corresponding values of z1 and z2
for i in range(0, len(X)):
    Z=Z+[[2*X[i][0]+3*X[i][1],4*X[i][0]-2*X[i][1]]]
```

```
# Mostra imagem dos PE no espaço das funcoes objetivo
# Show image of EP in objective function space
print("Images of these EP in objective function space:\n",Z)

# Constroi vetores para os graficos / Set vectors for the graphics
XX1=[] # Valores do eixo de x1 / Values of x1 axis
XX2=[] # Valores do eixo de x2 / Values of x2 axis
ZZ1=[] # Valores do eixo de z1 / Values of z1 axis
ZZ2=[] # Valores do eixo de z2 / Values of z2 axis
for i in range(0, len(X)):
    XX1=XX1+[X[i][0]]
    XX2=XX2+[X[i][1]]
    ZZ1=ZZ1+[Z[i][0]]
    ZZ2=ZZ2+[Z[i][1]]

# Soluções ideal e anti-ideal / Ideal and anti-ideal solutions
ideal=[max(ZZ1),max(ZZ2)]
anti_ideal=[min(ZZ1),min(ZZ2)]
print("---> Ideal solution =",ideal)
print("---> Anti-ideal solution =",anti_ideal)

# Grafico no espaço das variaveis de decisao
# Graphic in the space of decision variables
# -- Inicializar tamanho da figura / Set figure size
plt.figure(figsize=(20,10))
# -- Criar grafico de linhas / Create line plot
plt.plot(XX1,XX2,color="green")
plt.scatter(XX1,XX2,color="red",linewidth=6.0)
# -- Adicionar legendas dos eixos / Add axis labels
plt.xlabel("x1",size=20)
plt.ylabel("x2",size=20)
# -- Adicionar titulo ao grafico / Add plot title
plt.title("Efficient region",size=32)
# -- Mostrar coordenadas dos PE / Show coordinates of EP
for i, j in zip(XX1,XX2):
   plt.text(i,j,'({}, {})'.format(i, j))
# -- Mostrar grafico / Show plot
plt.show()
input("Press ENTER to continue...")
# Grafico no espaço das funcoes objetivo
# Graphic in the objective space
# -- Inicializar tamanho da figura / Set figure size
plt.figure(figsize=(20,10))
# -- Criar grafico de linhas / Create line plot
plt.plot(ZZ1,ZZ2,color="blue")
plt.scatter(ZZ1,ZZ2,color="red",linewidth=6.0)
# -- Adicionar legendas dos eixos / Add axis labels
plt.xlabel("z1",size=20)
plt.ylabel("z2",size=20)
# -- Adicionar titulo ao grafico / Add plot title
plt.title("Non-dominated region",size=32)
# -- Mostrar coordenadas dos PE / Show coordinates of EP
for i, j in zip(ZZ1,ZZ2):
   plt.text(i,j,'({}, {})'.format(round(i,2), round(j,2)))
# -- Mostrar grafico / Show plot
plt.show()
```

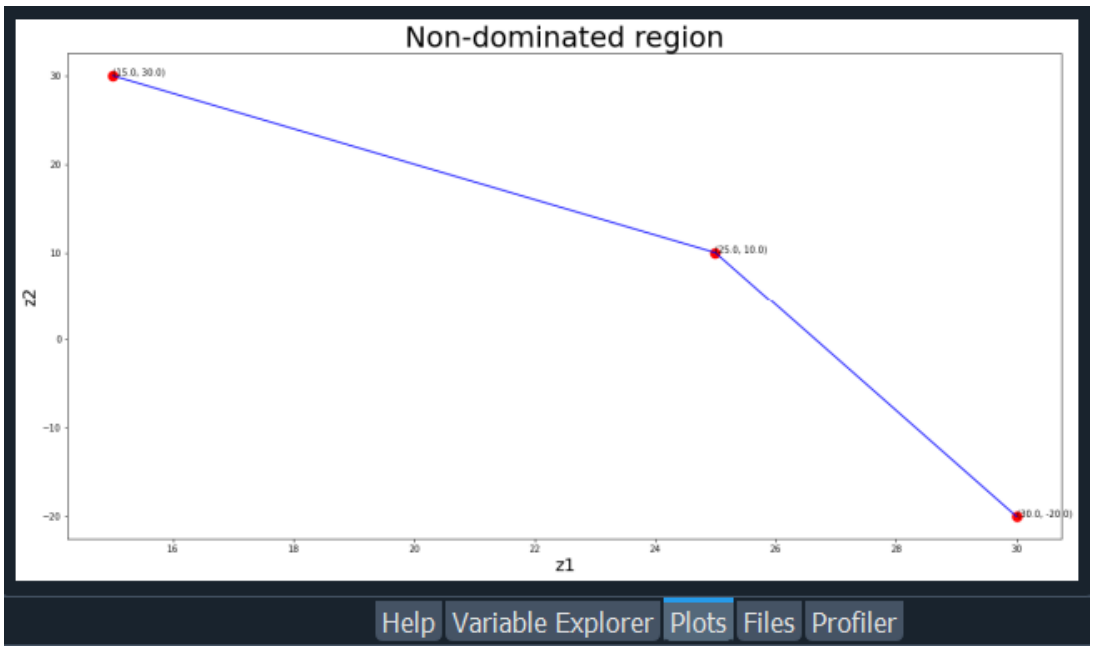
O resultado que surge na consola é:

```
In [35]: runfile('C:/Users/Teresa/Arquivos/MOAD/
Example 6-v3.py', wdir='C:/Users/Teresa/Arquivos
Projetos')
Extreme points of the efficient region:
 [[7.5, 0.0], [5.0, 5.0], [0.0, 10.0]]
Images of these EP in objective function space:
 [[15.0, 30.0], [25.0, 10.0], [30.0, -20.0]]
---> Ideal solution = [30.0, 30.0]
---> Anti-ideal solution = [15.0, -20.0]

Press ENTER to continue...
```

E em modo gráfico:



Após pressionar a tecla ENTER surge novo gráfico:



Interpretação dos resultados

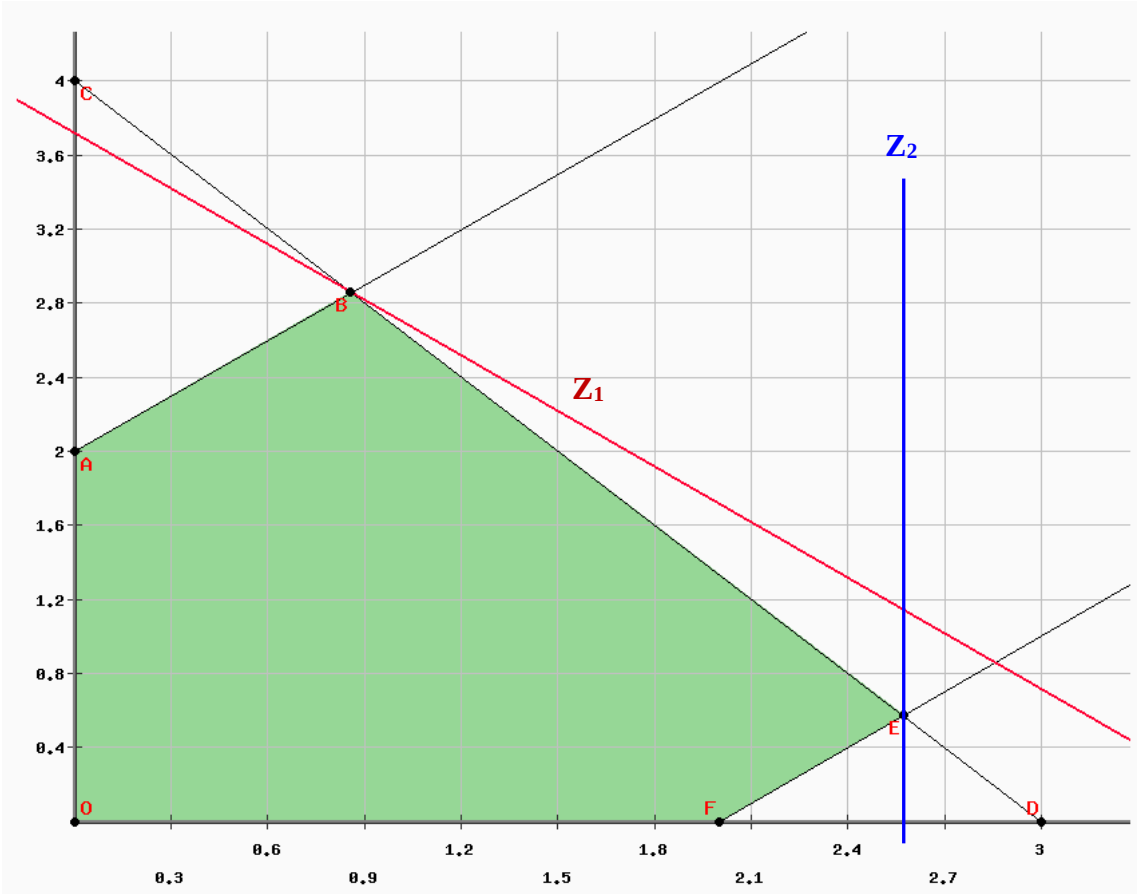
Os resultados anteriores traduzem as conclusões que foram apresentadas inicialmente, ou seja, que $K_{EF} = [A,C] \cup [C,E]$. Para além disso, mostram a região não dominada, como sendo as arestas que ligam as imagens dos pontos A, C e E, no espaço z_1z_2. As soluções ideal e anti-ideal são, respetivamente, $z_{ideal}=(30,30)$ e $z_{anti\text{-}ideal}=(15,-20)$.

EXEMPLO 7

Considere o seguinte problema de PLMO (exercício nº 4 da Folha Prática nº 4):

$$\begin{aligned}
&\text{Max } z_1 = x_1 + x_2\\
&\text{Max } z_2 = 2\,x_1\\
&\text{sujeito a}\\
&\quad -3\,x_1 + 3\,x_2 \le 6\\
&\quad 4\,x_1 + 3\,x_2 \le 12\\
&\quad x_1 - x_2 \le 2\\
&\quad x_1 \ge 0,\ x_2 \ge 0
\end{aligned}$$

A representação gráfica é a seguinte:



Verifica-se que as funções objetivo z_1 e z_2 atingem o seu valor ótimo nos pontos B e E, respetivamente. Recorrendo à técnica dos cones de dominância, seria possível concluir que a região eficiente, ou conjunto das soluções eficientes, é:

$$K_{EF} = [B,E]$$

O conjunto das soluções não dominadas (K_{ND}) é, pois, a aresta delimitada pelas imagens dos pontos B e E no espaço das funções objetivo.

Segue-se a resolução deste problema pelas três abordagens conhecidas.

1ª Abordagem

Segundo esta abordagem, o problema a resolver primeiramente é:

$$\begin{aligned} &\text{Max } z_1 = x_1 + x_2 \\ &\text{sujeito a} \\ &\quad -3x_1 + 3x_2 \leq 6 \\ &\quad 4x_1 + 3x_2 \leq 12 \\ &\quad x_1 - x_2 \leq 2 \\ &\quad x_1 \geq 0,\ x_2 \geq 0 \end{aligned}$$

A implementação em Python e os respetivos resultados de otimização, são os que se mostram em seguida.

```
"""
2o Problema de PLMO / 2nd MOLP problem
"""

# Importar biblioteca / Import library
from pulp import *

# Criar modelo / Create model
model=LpProblem("PLMO_2/MOLP_2",LpMaximize)

# Criar variaveis de decisao / Create decision variables
x1=LpVariable("x1",lowBound=0)
x2=LpVariable("x2",lowBound=0)

# Adicionar funcao objetivo ao modelo / Add objective function to model
model+= x1 + x2

# Adicionar restricoes ao modelo / Add constraints to model
model+= -3*x1 + 3*x2 <= 6
model+= 4*x1 + 3*x2 <= 12
model+= x1 - x2 <= 2

print(model)

# Resolver modelo / Solve model
model.solve()

# Visualizar resultados / Visualize results
print("-------- Resultados / Results ---------")
print(f"Status = {model.status} <=> {LpStatus[model.status]}")
print(f"z* = {value(model.objective)}")
for var in model.variables():
    print(f"{var.name}* = {var.value()}")
for name,constraint in model.constraints.items():
    print(f"{name}: {constraint.value()}")
```

O resultado que surge na consola é.

```
-------- Resultados / Results ---------
Status = 1 <=> Optimal
z* = 3.714283
x1* = 0.857143
x2* = 2.85714
_C1: -9.000000000369823e-06
_C2: -8.000000001118224e-06
_C3: -3.9999969999999996
```

De acordo com a resolução gráfica apresentada anteriormente, estes resultados correspondem ao ponto B: x=(6/7,20/7), com $z_1 = 26/7$.

Agora, reformulamos o problema original de modo a que a segunda função objetivo seja maximizada sujeita a uma restrição adicional. Essa restrição adicional exige que o primeiro objetivo tenha pelo menos o valor 26/7 (≈3.7142).

O modelo reformulado que agora teremos que resolver é o seguinte:

$$\text{Max } z_2 = 2\,x_1$$

sujeito a

$$\begin{aligned} -3\,x_1 + 3\,x_2 &\le 6 \\ 4\,x_1 + 3\,x_2 &\le 12 \\ x_1 - x_2 &\le 2 \\ x_1 + x_2 &\ge 26/7 \\ x_1 \ge 0,\ x_2 &\ge 0 \end{aligned}$$

O código em Python, depois das alterações, é o seguinte:

```
"""
2o Problema de PLMO / 2nd MOLP problem
"""

# Importar biblioteca / Import library
from pulp import *

# Criar modelo / Create model
model=LpProblem("PLMO_2/MOLP_2",LpMaximize)

# Criar variaveis de decisao / Create decision variables
x1=LpVariable("x1",lowBound=0)
x2=LpVariable("x2",lowBound=0)

# Adicionar funcao objetivo ao modelo / Add objective function to model
model+= 2*x1

# Adicionar restricoes ao modelo / Add constraints to model
model+= -3*x1 + 3*x2 <= 6
model+= 4*x1 + 3*x2 <= 12
model+= x1 - x2 <= 2
model+= x1 + x2 >= 3.7142

print(model)

# Resolver modelo / Solve model
model.solve()

# Visualizar resultados / Visualize results
print("-------- Resultados / Results ---------")
print(f"Status = {model.status} <=> {LpStatus[model.status]}")
print(f"z* = {value(model.objective)}")
for var in model.variables():
    print(f"{var.name}* = {var.value()}")
for name,constraint in model.constraints.items():
    print(f"{name}: {constraint.value()}")
```

O resultado final que surge na consola é.

```
-------- Resultados / Results ---------
Status = 1 <=> Optimal
z* = 1.7148
x1* = 0.8574
x2* = 2.8568
_C1: -0.001800000000001134
_C2: 0.0
_C3: -3.9993999999999996
_C4: 0.0
```

Interpretação dos resultados

A abordagem usada sugere que a melhor solução para o problema é a que otimiza z_1, ou seja, o ponto B. Salvo pequenos erros de arredondamento (na 4ª casa decimal), os valores são $x_1 = 6/7$ e $x_2 = 20/7$, com os valores das funções objetivo $z_1 = 26/7$ (confirmado por _C4=0) e $z_2 = 12/7$. Se tivéssemos otimizado z_2 em primeiro lugar e a considerássemos como restrição para maximizar z_1, a melhor solução seria o ponto E, com $x_1 = 18/7$ e $x_2 = 4/7$, correspondendo a $z_1 = 22/7$ e $z_2 = 36/7$. Ambas as soluções referidas são soluções eficientes / não dominadas, pois correspondem aos ótimos individuais de z_1 e z_2.

2ª Abordagem

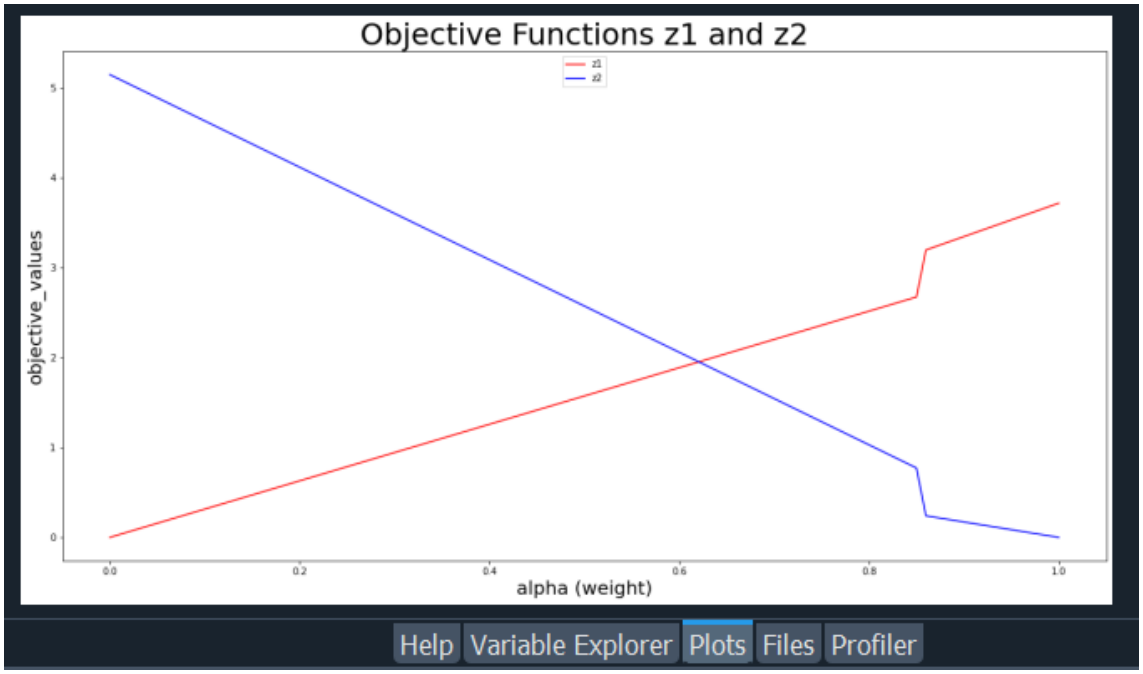
A implementação que se segue, resolve o problema enunciado anteriormente usando o método da “soma ponderada”.

```
"""
# Importar biblioteca PuLP / Import PuLP library
from pulp import *

# Importar sub-modulo Pyplot da biblioteca Matplotlib / Import sub-module Pyplot from libray Matplotlib
import matplotlib.pyplot as plt

# Importar biblioteca Pandas para poder guardar dados em formato Dataframe
# Import Pandas library for being able to store data in DataFrame format
#import pandas as pd
from pandas import DataFrame

# Inicializar um DataFrame vazio para guardar resultados da otimização
# Initialize empty DataFrame for storing optimization results
results = DataFrame(columns=["alpha","x1_opt","x2_opt","z1","z2"])

# Criar variáveis de decisão / Create decision variables
x1=LpVariable("x1",lowBound=0)
x2=LpVariable("x2",lowBound=0)

# Definir tamanho do passo (incremento) / Define step-size
step = 0.01

# Iterar para valores de alpha (peso) entre 0 e 1 com incremento step, e guardar resultados no DataFrame
# Iterate through alpha (weight) values from 0 to 1 with step, and save results into DataFrame
for i in range(0,101):
        # Criar novo modelo / Create new model
        model=LpProblem("PLMO_2/MOLP_2",LpMaximize)
        # Adicionar funcao objetivo como soma ponderada de z1 e z2: z = alpha*z1 + (1-alpha)*z2
        # Add objective function as a weighted sum of z1 and z2: z = alpha*z1 + (1-alpha)*z2
        alpha=i/100
        model += alpha*(x1+x2)+(1-alpha)*(2*x1)
        # Adicionar restricoes / Add constraints
        model += -3*x1 + 3*x2 <= 6
        model += 4*x1 + 3*x2 <= 12
        model += x1 - x2 <= 2
        # Resolver problema / Solve problem
        model.solve()
        # Guardar resultados no DataFrame / Save results into DataFrame
        results.loc[i] = [alpha,
                          value(x1),
                          value(x2),
                          value(alpha*(x1+x2)),
                          value((1-alpha)*(2*x1))]

# Visualizar resultados de otimizacao numa tabela / Visualize optimization results in a table
for i in range(0,100,15):
       print(results[i:i+15])
       input("Press ENTER key to continue...")

# Visualizar resultados de otimizacao num grafico
# Visualize optimization results in a plot
# -- Inicializar tamanho da figura / Set figure size
plt.figure(figsize=(20,10))
# -- Criar grafico de linhas / Create line plot
plt.plot(results["alpha"],results["z1"],color="red",label="z1")
plt.plot(results["alpha"],results["z2"],color="blue",label="z2")
plt.legend(loc="upper center")
# -- Adicionar legendas dos eixos / Add axis labels
plt.xlabel("alpha (weight)",size=20)
plt.ylabel("objective_values",size=20)
# -- Adicionar título ao gráfico / Add plot title
plt.title("Objective Functions z1 and z2",size=32)
# -- Mostrar grafico / Show plot
plt.show()
```

Os resultados que surgirão na consola em formato de tabela serão:

```
In [18]: runfile('C:/Users/Teresa/Arquivos/MOAD/AULAS PRÁTICAS/
Users/Teresa/Arquivos/MOAD/AULAS PRÁTICAS/PYTHON/Projetos')
    alpha   x1_opt    x2_opt        z1        z2
0    0.00  2.57143  0.571429  0.000000  5.142860
1    0.01  2.57143  0.571429  0.031429  5.091431
2    0.02  2.57143  0.571429  0.062857  5.040003
3    0.03  2.57143  0.571429  0.094286  4.988574
4    0.04  2.57143  0.571429  0.125714  4.937146
5    0.05  2.57143  0.571429  0.157143  4.885717
6    0.06  2.57143  0.571429  0.188572  4.834288
7    0.07  2.57143  0.571429  0.220000  4.782860
8    0.08  2.57143  0.571429  0.251429  4.731431
9    0.09  2.57143  0.571429  0.282857  4.680003
10   0.10  2.57143  0.571429  0.314286  4.628574
11   0.11  2.57143  0.571429  0.345714  4.577145
12   0.12  2.57143  0.571429  0.377143  4.525717
13   0.13  2.57143  0.571429  0.408572  4.474288
14   0.14  2.57143  0.571429  0.440000  4.422860

Press ENTER key to continue...
```

E em formato de gráfico:



Interpretação dos resultados

Esta abordagem serve principalmente para determinar os pontos extremos da região eficiente que, neste caso, correspondem simplemente aos ótimos individuais de z_1 e z_2. Estes pontos são facilmente obtidos pela análise da tabela.

3ª Abordagem

Segue-se a implementação usando a nova abordagem.

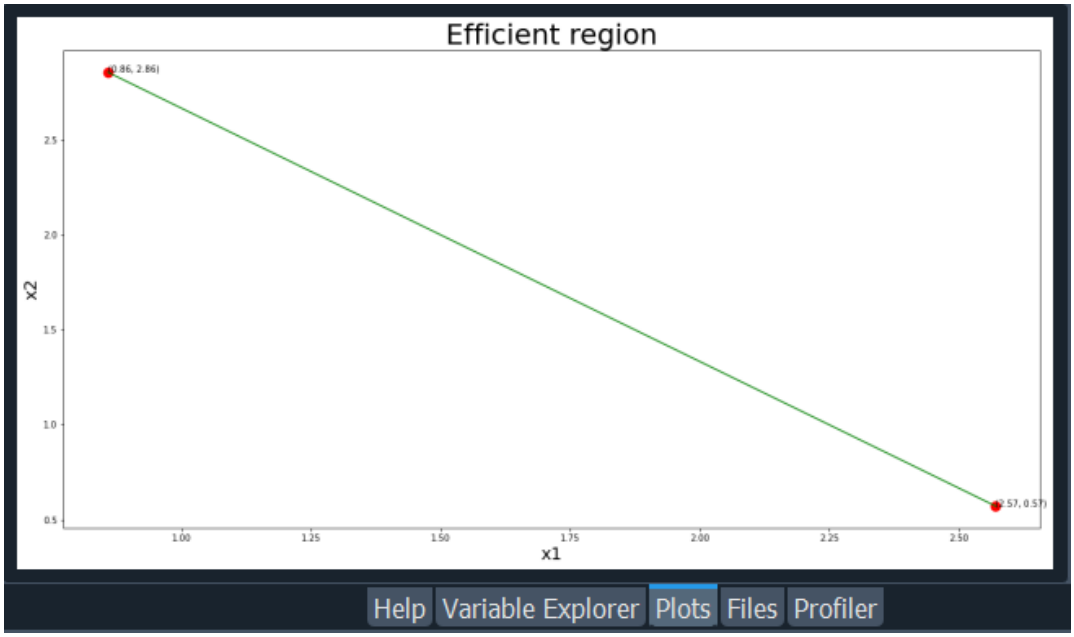
```
"""
2o Problema de PLMO / 2nd MOLP problem
"""
# Importar biblioteca PuLP / Import PuLP library
from pulp import *

# Importar sub-modulo Pyplot da biblioteca Matplotlib
# Import sub-module Pyplot from libray Matplotlib
import matplotlib.pyplot as plt

# Inicializar duas listas para guardar pontos extremos (PE) da região eficiente
# e respetivos valores de z1 e z2 / Initialize two lists to save extreme points
# (EP) of efficient region and respective values of z1 and z2
X=[]
Z=[]
# Criar variáveis de decisão / Create decision variables
x1=LpVariable("x1",lowBound=0)
x2=LpVariable("x2",lowBound=0)

# Para determinar PE usa abordagem da soma ponderada das duas funções objetivo
# To find EP use approach of the weighted sum of two objective functions
# Definir incremento de  alpha / Define step-size for alpha
step = 0.1
# Numero total de amostras / Total number of samples
n = int(1/step)
# Iterar para valores de alpha (peso) entre 0 e 1
# Iterate through alpha (weight) values from 0 to 1
for i in range(0,n+1):
        # Calcula valor de alpha / Calculate alpha value
        alpha=i/n
        # Criar novo modelo / Create new model
        model=LpProblem("PLMO_2/MOLP_2",LpMaximize)
        # Adicionar funcao objetivo z = alpha*z1 + (1-alpha)*z2
        # Add objective function z = alpha*z1 + (1-alpha)*z2
        model += alpha*(x1+x2)+(1-alpha)*(2*x1)
        # Adicionar restricoes / Add constraints
        model += -3*x1 + 3*x2 <= 6
        model += 4*x1 + 3*x2 <= 12
        model += x1 - x2 <= 2
        # Resolver modelo / Solve model
        model.solve()
        # Guardar cada nova solução determinada pois é PE da região eficiente
        # Save each new solution found since it is an EP of efficient region
        if X.count([value(x1),value(x2)]) == 0:
                X=X+[[value(x1),value(x2)]]

# Mostra PE da região eficiente / Show EP of efficient region
print("Extreme points of the efficient region:\n",X)
# Calcula valores correspondentes de z1 e z2
# Find corresponding values of z1 and z2
for i in range(0, len(X)):
    Z=Z+[[X[i][0]+X[i][1],2*X[i][0]]]
# Mostra imagem dos PE no espaço das funcoes objetivo
# Show image of EP in objective function space
print("Images of these EP in objective function space:\n",Z)
```

```
# Constroi vetores para os graficos / Set vectors for the graphics
XX1=[] # Valores do eixo de x1 / Values of x1 axis
XX2=[] # Valores do eixo de x2 / Values of x2 axis
ZZ1=[] # Valores do eixo de z1 / Values of z1 axis
ZZ2=[] # Valores do eixo de z2 / Values of z2 axis
for i in range(0, len(X)):
    XX1=XX1+[X[i][0]]
    XX2=XX2+[X[i][1]]
    ZZ1=ZZ1+[Z[i][0]]
    ZZ2=ZZ2+[Z[i][1]]

# Soluções ideal e anti-ideal / Ideal and anti-ideal solutions
ideal=[max(ZZ1),max(ZZ2)]
anti_ideal=[min(ZZ1),min(ZZ2)]
print("---> Ideal solution =",ideal)
print("---> Anti-ideal solution =",anti_ideal)

# Grafico no espaço das variaveis de decisao
# Graphic in the space of decision variables
# -- Inicializar tamanho da figura / Set figure size
plt.figure(figsize=(20,10))
# -- Criar grafico de linhas / Create line plot
plt.plot(XX1,XX2,color="green")
plt.scatter(XX1,XX2,color="red",linewidth=6.0)
# -- Adicionar legendas dos eixos / Add axis labels
plt.xlabel("x1",size=20)
plt.ylabel("x2",size=20)
# -- Adicionar titulo ao grafico / Add plot title
plt.title("Efficient region",size=32)
# -- Mostrar coordenadas dos PE / Show coordinates of EP
for i, j in zip(XX1,XX2):
   plt.text(i,j,'({}, {})'.format(round(i,2), round(j,2)))
# -- Mostrar grafico / Show plot
plt.show()
input("Press ENTER to continue...")
# Grafico no espaço das funcoes objetivo
# Graphic in the objective space
# -- Inicializar tamanho da figura / Set figure size
plt.figure(figsize=(20,10))
# -- Criar grafico de linhas / Create line plot
plt.plot(ZZ1,ZZ2,color="blue")
plt.scatter(ZZ1,ZZ2,color="red",linewidth=6.0)
# -- Adicionar legendas dos eixos / Add axis labels
plt.xlabel("z1",size=20)
plt.ylabel("z2",size=20)
# -- Adicionar titulo ao grafico / Add plot title
plt.title("Non-dominated region",size=32)
# -- Mostrar coordenadas dos PE / Show coordinates of EP
for i, j in zip(ZZ1,ZZ2):
    plt.text(i,j,'({}, {})'.format(round(i,2), round(j,2)))
# -- Mostrar grafico / Show plot
plt.show()
```

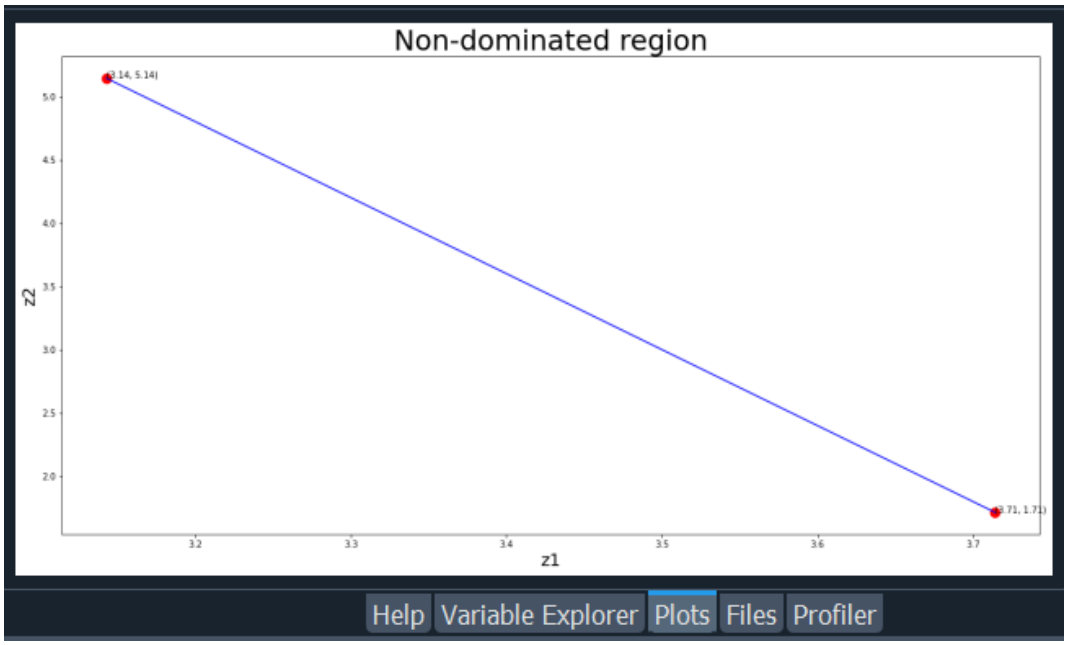
O resultado que surge na consola é:

```
In [39]: runfile('C:/Users/Teresa/Arquivos/MOAD/AULAS PRÁTICAS/PYTHON/Projetos/
Example 7-v3.py', wdir='C:/Users/Teresa/Arquivos/MOAD/AULAS PRÁTICAS/PYTHON/
Projetos')
Extreme points of the efficient region:
 [[2.57143, 0.571429], [0.857143, 2.85714]]
Images of these EP in objective function space:
 [[3.1428589999999996, 5.14286], [3.714283, 1.714286]]
---> Ideal solution = [3.714283, 5.14286]
---> Anti-ideal solution = [3.1428589999999996, 1.714286]

Press ENTER to continue...
```

E em modo gráfico:



Após pressionar a tecla ENTER surge novo gráfico:



Interpretação dos resultados

Os resultados anteriores traduzem as conclusões que foram obtidas no início (e na aula teórico-prática): o conjunto das soluções eficientes (ou região eficiente) corresponde à aresta formada pelos pontos B e E, sendo que o conjunto das soluções não dominadas (ou região não dominada) corresponde à aresta delimitada pelas imagens desses pontos no espaço z_1z_2.