# N05. Selección oferta

Con base en la proyección de precios de bolsa mensuales (Pbm) para tres escenarios, se realizará la selección de la mejor oferta.

(Para convertir a docx: pandoc N05\_SeleccionOferta.ipynb -s -o N04.docx).

## 5.1. Ofertas presentadas

Se presentaron 7 ofertas:

1. Cuatro con la modalidad M2, de precio atado al Precio de Bolsa Horario.
2. Una con la modalidad M1 de precio fijo.
3. Una modalidad M4, con la combinación por porcentajes de precio fijo (80%) y Precio de Bolsa Horario (20%).
4. Una modalidad M5, con la combinación por cantidades de energía de precio fijo (hasta 8 GWh) y Precio de Bolsa Horario (por encima de 8 GWh).

Las ofertas M2 y M5 de DICEL se integraron en una sola, ya que se complementan en los diferentes periodos.

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 5.1.1. Carga paquetes requeridos para realizar la selección.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
import numpy as np  
import pandas as pd  
import datetime as dt  
import matplotlib.pyplot as plt  
%matplotlib widget

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 5.1.2. Crea la estructura de las ofertas.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
ofertas = pd.DataFrame({'nombre\_oferta': ['ENEL\_M2', 'AIRE\_M2', 'DICEL\_M2\_M5', \  
 'EMSA\_M2', 'AIRE\_M1', 'AIRE\_M4'], 'periodos': ['P1:P7', \  
 'P1:P7', 'P1:P7', 'P1:P2', 'P1:P7', 'P1:P7'], 'cargoC\_pbh': [8.90, \  
 13.00, 16.00, 18.00, None, None], 'precioFijo': [None, \  
 None, 294.00, None, 280.20, 280.20], 'porcentajePFijo': [None, \  
 None, 'Hasta 8 GWh', None, '100%', 'Hasta 80%'], 'comentario': ['C se ajusta con IPP', \  
 'C se ajusta con IPP', 'Se integra M2 y M5', \  
 'C se ajusta con IPP', 'PFijo cambia por periodo', \  
 'PFijo cambia por periodo']}, index=range(1, 7))  
ofertas

nombre\_oferta periodos cargoC\_pbh precioFijo porcentajePFijo \  
1 ENEL\_M2 P1:P7 8.9 NaN None   
2 AIRE\_M2 P1:P7 13.0 NaN None   
3 DICEL\_M2\_M5 P1:P7 16.0 294.0 Hasta 8 GWh   
4 EMSA\_M2 P1:P2 18.0 NaN None   
5 AIRE\_M1 P1:P7 NaN 280.2 100%   
6 AIRE\_M4 P1:P7 NaN 280.2 Hasta 80%   
  
 comentario   
1 C se ajusta con IPP   
2 C se ajusta con IPP   
3 Se integra M2 y M5   
4 C se ajusta con IPP   
5 PFijo cambia por periodo   
6 PFijo cambia por periodo

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 5.1.3. Se crea estructura de vpn\_ofertas y vpn\_ofertas\_p1p2 (Em Millones de COP).  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
vpn\_ofertas = pd.DataFrame({'Escenario Alto': np.NaN, \  
 'Escenario Medio': np.NaN, 'Escenario Bajo': np.NaN}, index=['ENEL\_M2', 'AIRE\_M2', 'DICEL\_M2\_M5', \  
 'EMSA\_M2', 'AIRE\_M1', 'AIRE\_M4'])  
vpn\_ofertas\_p1p2 = pd.DataFrame({'Escenario Alto': np.NaN, \  
 'Escenario Medio': np.NaN, 'Escenario Bajo': np.NaN}, index=['ENEL\_M2', 'AIRE\_M2', 'DICEL\_M2\_M5', \  
 'EMSA\_M2', 'AIRE\_M1', 'AIRE\_M4'])  
vpn\_ofertas

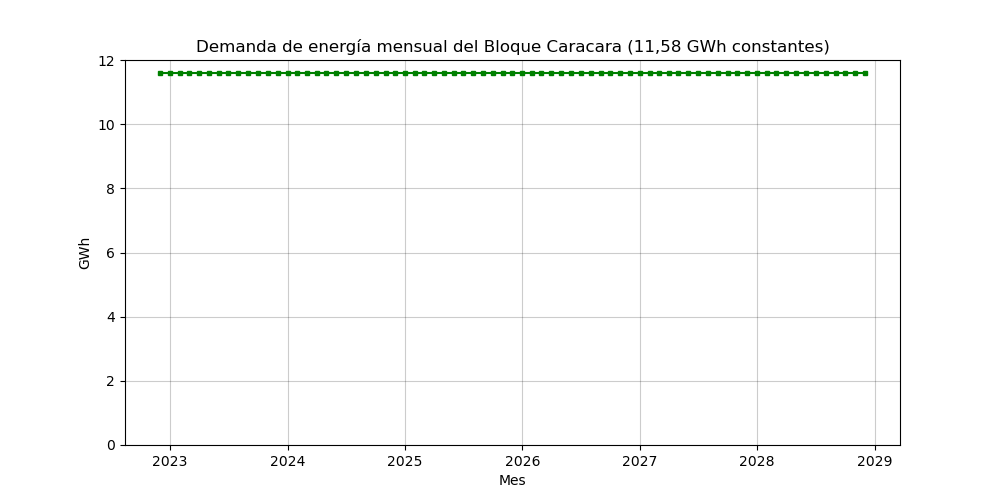
Escenario Alto Escenario Medio Escenario Bajo  
ENEL\_M2 NaN NaN NaN  
AIRE\_M2 NaN NaN NaN  
DICEL\_M2\_M5 NaN NaN NaN  
EMSA\_M2 NaN NaN NaN  
AIRE\_M1 NaN NaN NaN  
AIRE\_M4 NaN NaN NaN

## 5.2. Demanda de energía mensual

En el análisis se asumirá una demanda mensual constante de 11,58 GWh/mes. El primer mes es diciembre de 2022 y el último mes es diciembre de 2028.

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 5.2.1. Se crea la estructura de 'demanda'.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
demanda = pd.DataFrame({'demanda': 11.58}, \  
 index=pd.date\_range(start="2022-12-01",end="2028-12-01", freq='MS'))

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 5.2.2. Grafica de demanda.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig = plt.figure(figsize=(10,5))  
ax1 = fig.add\_subplot(1, 1, 1) # Crea una figura conteniendo un solo eje.  
plt.subplots\_adjust(left=None, bottom=None, right=None, top=None, \  
 wspace=None, hspace=None)  
ax1.set\_title ('Demanda de energía mensual del Bloque Caracara (11,58 GWh constantes)')  
ax1.set\_ylabel ('GWh')  
ax1.set\_xlabel ('Mes')  
ax1.plot(demanda.index, demanda.demanda, 'g-s', \  
 label='Demanda', markersize=3)  
# plt.legend(loc='best')  
# plt.xlim(pd.to\_datetime('2018-01-01'), pd.to\_datetime('2022-07-01'))  
plt.ylim(0, 12)  
plt.grid(axis='both', color='k', alpha=0.2)  
plt.show()



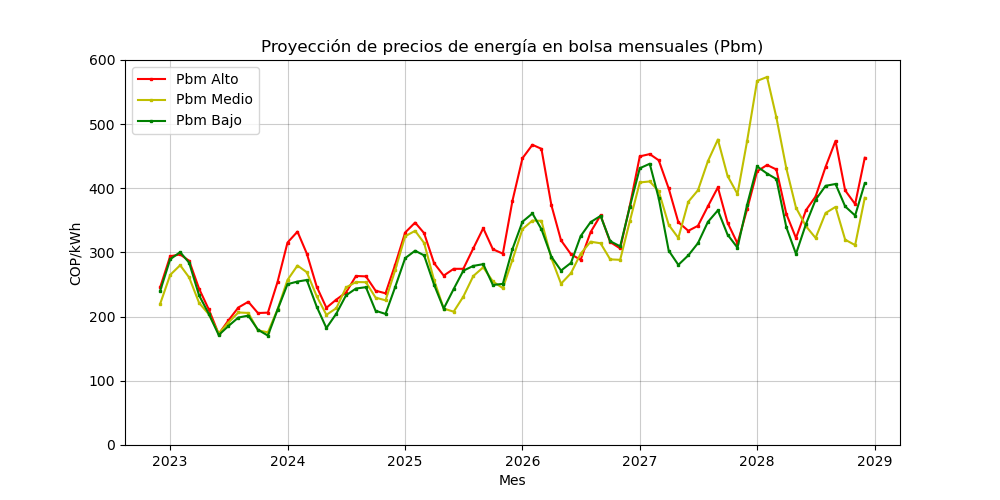
## 5.3. Proyección de precios de bolsa mensual

El análisis se realizará asumiendo tres escenarios de precios de energía de bolsa mensuales, que fueron elaborados previamente y cuya metodología de cálculo se muestra en un documento complementario.

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 5.3.1. Se leen las proyecciones de pbm, pbm\_py\_f. Calculadas previamente.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Lee pbm\_py  
pbm\_py\_f = pd.read\_parquet\  
 ('parquet/pbm\_py\_f.parquet.gzip', engine='fastparquet')

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 5.3.2. Se ajustan columnas y fechas.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
pbm\_py\_f = pbm\_py\_f.drop(columns=['pbm'], axis=1)  
pbm\_py\_f = pbm\_py\_f.loc[pd.to\_datetime('2022-12-01'):pd.to\_datetime('2028-12-01'), :]

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 5.3.3. Grafica de proyección de precios de energía en bolsa mensuales.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig = plt.figure(figsize=(10,5))  
ax1 = fig.add\_subplot(1, 1, 1) # Crea una figura conteniendo un solo eje.  
plt.subplots\_adjust(left=None, bottom=None, right=None, top=None, \  
 wspace=None, hspace=None)  
ax1.set\_title ('Proyección de precios de energía en bolsa mensuales (Pbm)')  
ax1.set\_ylabel ('COP/kWh')  
ax1.set\_xlabel ('Mes')  
ax1.plot(pbm\_py\_f.index, pbm\_py\_f.pbm\_alto, 'r-s', \  
 label='Pbm Alto', markersize=2)  
ax1.plot(pbm\_py\_f.index, pbm\_py\_f.pbm\_medio, 'y-s', \  
 label='Pbm Medio', markersize=2)  
ax1.plot(pbm\_py\_f.index, pbm\_py\_f.pbm\_bajo, 'g-s', \  
 label='Pbm Bajo', markersize=2)  
plt.legend(loc='best')  
# plt.xlim(pd.to\_datetime('2018-01-01'), pd.to\_datetime('2022-07-01'))  
plt.ylim(0, 600)  
plt.grid(axis='both', color='k', alpha=0.2)  
plt.show()

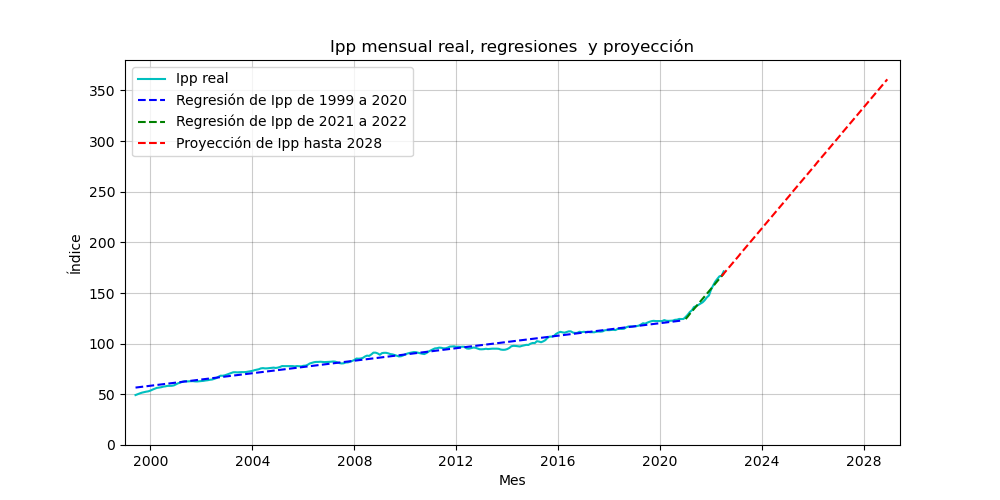


## 5.4. Proyección del IPP

Para poder calcular los precios futuros, que permiten estimar los egresos de energía, se debe realizar una proyección del IPP, ya que los componentes de los precios de las ofertas (cargo C de comercialización y el precio fijo) están expresados en pesos de abril de 2022. Estos precios se deben ajustar mensualmente.

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 5.4.1. Lee el ipp\_py (proyectado).  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Lee ipp\_py  
ipp\_py = pd.read\_parquet\  
 ('parquet/ipp\_py.parquet.gzip', engine='fastparquet')  
ipp\_28 = ipp\_py.loc[pd.to\_datetime('2022-12-01'): pd.to\_datetime('2028-12-01'), 'proyeccion28']  
ipp\_28.name = 'ipp'

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 5.4.2. Grafica el IPP.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig = plt.figure(figsize=(10,5))  
ax1 = fig.add\_subplot(1, 1, 1) # Crea una figura conteniendo un solo eje.  
plt.subplots\_adjust(left=None, bottom=None, right=None, top=None, \  
 wspace=None, hspace=None)  
ax1.set\_title ('Ipp mensual real, regresiones y proyección')  
ax1.set\_ylabel ('Índice')  
ax1.set\_xlabel ('Mes')  
ax1.plot(ipp\_py.index, ipp\_py.ipp, 'c-', \  
 label='Ipp real', markersize=2)  
ax1.plot(ipp\_py.index, ipp\_py.regresion99, 'b--', \  
 label='Regresión de Ipp de 1999 a 2020', markersize=2)  
ax1.plot(ipp\_py.index, ipp\_py.regresion21, 'g--', \  
 label='Regresión de Ipp de 2021 a 2022', markersize=2)  
ax1.plot(ipp\_py.index, ipp\_py.proyeccion28, 'r--', \  
 label='Proyección de Ipp hasta 2028', markersize=2)  
plt.legend(loc='best')  
plt.xlim(pd.to\_datetime('1999-01-01'), pd.to\_datetime('2029-06-01'))  
plt.ylim(0, 380)  
plt.grid(axis='both', color='k', alpha=0.2)  
plt.show()

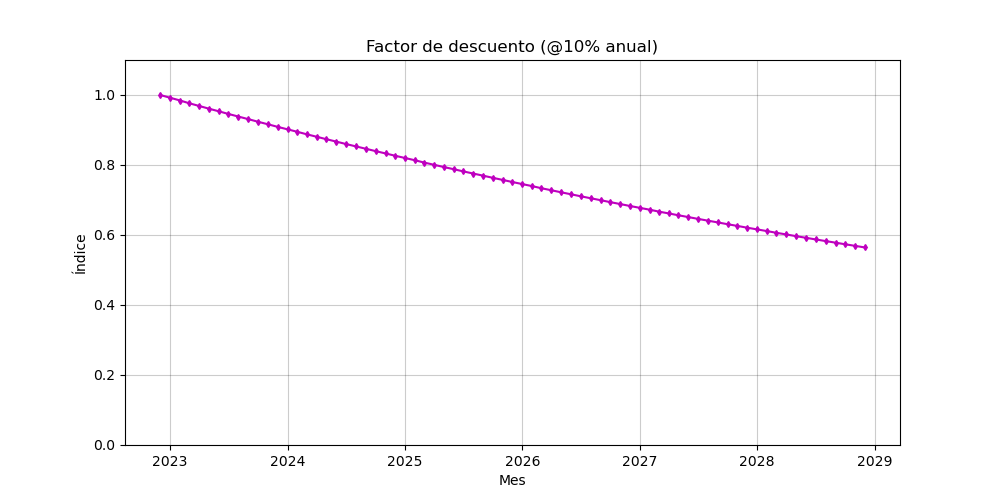


## 5.5. Factor de descuento

Para poder calcular el VPN de los egresos por compra de energía se requiere calcular el factor de descuento mensual. Inicialmente se asumirá una tasa de descuento anual del 10%, equivalente al 0,79% mensual.

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 5.5.1. Factor de descuento de los egresos. Tasa 10% anual.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
tasa\_descuento\_anual = 0.1 #10% anual.  
tasa\_descuento\_mensual = ((1 + tasa\_descuento\_anual) \*\* (1/12)) - 1  
tasa\_descuento\_mensual #tasa\_descuento\_mensual = 0,79%.  
factor\_descuento = pd.DataFrame({'n\_mes':range(0, 73)}, \  
 index=pd.date\_range(start="2022-12-01",end="2028-12-01", freq='MS'))  
factor\_descuento.insert(loc=len(factor\_descuento.columns), \  
 column='factor\_descuento', value=(1 / (1 + tasa\_descuento\_mensual) \*\* \  
 factor\_descuento.n\_mes), allow\_duplicates=True)  
factor\_descuento = factor\_descuento.drop(columns='n\_mes', axis=1)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 5.5.2. Grafica de factor de descuento.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig = plt.figure(figsize=(10,5))  
ax1 = fig.add\_subplot(1, 1, 1) # Crea una figura conteniendo un solo eje.  
plt.subplots\_adjust(left=None, bottom=None, right=None, top=None, \  
 wspace=None, hspace=None)  
ax1.set\_title ('Factor de descuento (@10% anual)')  
ax1.set\_ylabel ('Índice')  
ax1.set\_xlabel ('Mes')  
ax1.plot(factor\_descuento.index, factor\_descuento.factor\_descuento, 'm-d', \  
 label='Factor de descuento', markersize=3)  
# plt.legend(loc='best')  
# plt.xlim(pd.to\_datetime('2018-01-01'), pd.to\_datetime('2022-07-01'))  
plt.ylim(0, 1.1)  
plt.grid(axis='both', color='k', alpha=0.2)  
plt.show()



## 5.5. Egresos de las ofertas

La comparación de las ofertas se realizará con base en el VPN de los egresos de los cargos No Regulados, es decir los componentes de Generación (G) más Comercialización (C). Una vez se seleccione la mejor oferta y con el fin de estimar el presupuesto de los egresos por compra de energía, a la oferta seleccionada se le adicionarán los componentes de los cargos Regulados. Los componentes Regulados son iguales para todas las ofertas.

### 5.5.1. Egreso de la oferta ENEL\_M2

Con esta propuesta ENEL ofertó para todos los periodos, bajo la modalidad de Precio de Bolsa Horario más un cargo de Comercialización de 8,90 COP/kWh. Este precio está expresado en pesos de abril de 2022 y se debe indexar mensualmente con el IPP.

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 5.5.1.1. Se calcula egreso.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
egreso\_ENEL\_M2 = pd.concat([demanda, pbm\_py\_f, ipp\_28, factor\_descuento], \  
 axis=1)  
ipp\_202204 = ipp\_py.loc[pd.to\_datetime('2022-04-01'), 'ipp']  
comer0 = 8.9 #Oferta de ENEL\_M2 para abril de 2022.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Ajusta el cargo de comercialización con el IPP.  
egreso\_ENEL\_M2.insert(loc=len(egreso\_ENEL\_M2.columns), column='comer', \  
 value=egreso\_ENEL\_M2.ipp \* comer0/ipp\_202204, allow\_duplicates=True )  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Calcula el precio (COP/kWh) de energía en los tres escenarios (G+C).  
egreso\_ENEL\_M2.insert(loc=len(egreso\_ENEL\_M2.columns), column='precio\_alto', \  
 value=egreso\_ENEL\_M2.pbm\_alto + egreso\_ENEL\_M2.comer, allow\_duplicates=True )  
egreso\_ENEL\_M2.insert(loc=len(egreso\_ENEL\_M2.columns), column='precio\_medio', \  
 value=egreso\_ENEL\_M2.pbm\_medio + egreso\_ENEL\_M2.comer, allow\_duplicates=True )  
egreso\_ENEL\_M2.insert(loc=len(egreso\_ENEL\_M2.columns), column='precio\_bajo', \  
 value=egreso\_ENEL\_M2.pbm\_bajo + egreso\_ENEL\_M2.comer, allow\_duplicates=True )  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Calcula el egreso de energía (G+C) en los tres escenarios (Millones de COP).  
egreso\_ENEL\_M2.insert(loc=len(egreso\_ENEL\_M2.columns), column='egreso\_alto', \  
 value=egreso\_ENEL\_M2.precio\_alto \* egreso\_ENEL\_M2.demanda, allow\_duplicates=True )  
egreso\_ENEL\_M2.insert(loc=len(egreso\_ENEL\_M2.columns), column='egreso\_medio', \  
 value=egreso\_ENEL\_M2.precio\_medio \* egreso\_ENEL\_M2.demanda, allow\_duplicates=True )  
egreso\_ENEL\_M2.insert(loc=len(egreso\_ENEL\_M2.columns), column='egreso\_bajo', \  
 value=egreso\_ENEL\_M2.precio\_bajo \* egreso\_ENEL\_M2.demanda, allow\_duplicates=True )  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Calcula el egreso de energía descontado en los tres escenarios (Millones de COP).  
egreso\_ENEL\_M2.insert(loc=len(egreso\_ENEL\_M2.columns), column='egreso\_alto\_d', \  
 value=egreso\_ENEL\_M2.egreso\_alto \* egreso\_ENEL\_M2.factor\_descuento, \  
 allow\_duplicates=True )  
egreso\_ENEL\_M2.insert(loc=len(egreso\_ENEL\_M2.columns), column='egreso\_medio\_d', \  
 value=egreso\_ENEL\_M2.egreso\_medio \* egreso\_ENEL\_M2.factor\_descuento, \  
 allow\_duplicates=True )  
egreso\_ENEL\_M2.insert(loc=len(egreso\_ENEL\_M2.columns), column='egreso\_bajo\_d', \  
 value=egreso\_ENEL\_M2.egreso\_bajo \* egreso\_ENEL\_M2.factor\_descuento, \  
 allow\_duplicates=True )  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Carga el VPN de egresos en los tres escenarios en la estructura vpn\_ofertas.  
vpn\_ofertas.loc['ENEL\_M2', 'Escenario Alto'] = \  
 egreso\_ENEL\_M2.egreso\_alto\_d.sum()  
vpn\_ofertas.loc['ENEL\_M2', 'Escenario Medio'] = \  
 egreso\_ENEL\_M2.egreso\_medio\_d.sum()  
vpn\_ofertas.loc['ENEL\_M2', 'Escenario Bajo'] = \  
 egreso\_ENEL\_M2.egreso\_bajo\_d.sum()  
# #\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Carga el VPN de egresos en los tres escenarios en vpn\_ofertas\_p1p2.  
vpn\_ofertas\_p1p2.loc['ENEL\_M2', 'Escenario Alto'] = \  
 egreso\_ENEL\_M2.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2023-12-01'), 'egreso\_alto\_d'].sum()  
vpn\_ofertas\_p1p2.loc['ENEL\_M2', 'Escenario Medio'] = \  
 egreso\_ENEL\_M2.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2023-12-01'), 'egreso\_medio\_d'].sum()  
vpn\_ofertas\_p1p2.loc['ENEL\_M2', 'Escenario Bajo'] = \  
 egreso\_ENEL\_M2.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2023-12-01'), 'egreso\_bajo\_d'].sum()

### 5.5.2. Egreso de la oferta AIRE\_M2

Con esta propuesta AIRE ofertó para todos los periodos, bajo la modalidad de Precio de Bolsa Horario más un cargo de Comercialización de 13,00 COP/kWh. Este precio está expresado en pesos de abril de 2022 y se debe indexar mensualmente con el IPP.

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 5.5.2.1. Se calcula egreso.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
egreso\_AIRE\_M2 = pd.concat([demanda, pbm\_py\_f, ipp\_28, factor\_descuento], \  
 axis=1)  
ipp\_202204 = ipp\_py.loc[pd.to\_datetime('2022-04-01'), 'ipp']  
comer0 = 13.00 #Oferta de AIRE\_M2 para abril de 2022.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Ajusta el cargo de comercialización con el IPP.  
egreso\_AIRE\_M2.insert(loc=len(egreso\_AIRE\_M2.columns), column='comer', \  
 value=egreso\_AIRE\_M2.ipp \* comer0/ipp\_202204, allow\_duplicates=True )  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Calcula el precio (COP/kWh) de energía en los tres escenarios (G+C).  
egreso\_AIRE\_M2.insert(loc=len(egreso\_AIRE\_M2.columns), column='precio\_alto', \  
 value=egreso\_AIRE\_M2.pbm\_alto + egreso\_AIRE\_M2.comer, allow\_duplicates=True )  
egreso\_AIRE\_M2.insert(loc=len(egreso\_AIRE\_M2.columns), column='precio\_medio', \  
 value=egreso\_AIRE\_M2.pbm\_medio + egreso\_AIRE\_M2.comer, allow\_duplicates=True )  
egreso\_AIRE\_M2.insert(loc=len(egreso\_AIRE\_M2.columns), column='precio\_bajo', \  
 value=egreso\_AIRE\_M2.pbm\_bajo + egreso\_AIRE\_M2.comer, allow\_duplicates=True )  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Calcula el egreso de energía (G+C) en los tres escenarios (Millones de COP).  
egreso\_AIRE\_M2.insert(loc=len(egreso\_AIRE\_M2.columns), column='egreso\_alto', \  
 value=egreso\_AIRE\_M2.precio\_alto \* egreso\_AIRE\_M2.demanda, allow\_duplicates=True )  
egreso\_AIRE\_M2.insert(loc=len(egreso\_AIRE\_M2.columns), column='egreso\_medio', \  
 value=egreso\_AIRE\_M2.precio\_medio \* egreso\_AIRE\_M2.demanda, allow\_duplicates=True )  
egreso\_AIRE\_M2.insert(loc=len(egreso\_AIRE\_M2.columns), column='egreso\_bajo', \  
 value=egreso\_AIRE\_M2.precio\_bajo \* egreso\_AIRE\_M2.demanda, allow\_duplicates=True )  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Calcula el egreso de energía descontado en los tres escenarios (Millones de COP).  
egreso\_AIRE\_M2.insert(loc=len(egreso\_AIRE\_M2.columns), column='egreso\_alto\_d', \  
 value=egreso\_AIRE\_M2.egreso\_alto \* egreso\_AIRE\_M2.factor\_descuento, \  
 allow\_duplicates=True )  
egreso\_AIRE\_M2.insert(loc=len(egreso\_AIRE\_M2.columns), column='egreso\_medio\_d', \  
 value=egreso\_AIRE\_M2.egreso\_medio \* egreso\_AIRE\_M2.factor\_descuento, \  
 allow\_duplicates=True )  
egreso\_AIRE\_M2.insert(loc=len(egreso\_AIRE\_M2.columns), column='egreso\_bajo\_d', \  
 value=egreso\_AIRE\_M2.egreso\_bajo \* egreso\_AIRE\_M2.factor\_descuento, \  
 allow\_duplicates=True )  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Carga el VPN de egresos en los tres escenarios en la estructura vpn\_ofertas.  
vpn\_ofertas.loc['AIRE\_M2', 'Escenario Alto'] = \  
 egreso\_AIRE\_M2.egreso\_alto\_d.sum()  
vpn\_ofertas.loc['AIRE\_M2', 'Escenario Medio'] = \  
 egreso\_AIRE\_M2.egreso\_medio\_d.sum()  
vpn\_ofertas.loc['AIRE\_M2', 'Escenario Bajo'] = \  
 egreso\_AIRE\_M2.egreso\_bajo\_d.sum()  
# #\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Carga el VPN de egresos en los tres escenarios en vpn\_ofertas\_p1p2.  
vpn\_ofertas\_p1p2.loc['AIRE\_M2', 'Escenario Alto'] = \  
 egreso\_AIRE\_M2.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2023-12-01'), 'egreso\_alto\_d'].sum()  
vpn\_ofertas\_p1p2.loc['AIRE\_M2', 'Escenario Medio'] = \  
 egreso\_AIRE\_M2.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2023-12-01'), 'egreso\_medio\_d'].sum()  
vpn\_ofertas\_p1p2.loc['AIRE\_M2', 'Escenario Bajo'] = \  
 egreso\_AIRE\_M2.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2023-12-01'), 'egreso\_bajo\_d'].sum()

### 5.5.3. Egreso de la oferta DICEL\_M2 integrada con DICEL\_M5

DICEL ofertó para el periodo 2022 y 2023 con la modalidad M2 de Precio de Bolsa Horario más un cargo de Comercialización de 16,00 COP/kWh. Para los periodos 2024 a 2028 ofertó con la modalidad M5 de precio fijo hasta una demanda de 8 GWh y para la demanda por encima de ese valor con un precio de bolsa más una comercialización. Estas dos ofertas son complementarias. Por ello, la oferta DICEL\_M2 se analizará como una sola oferta complementada con la oferta DICEL\_M5.

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 5.5.3.1. Se calcula egreso.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
demanda\_precio\_fijo = 8.00  
egreso\_DICEL\_M2\_M5 = pd.concat([demanda, pbm\_py\_f, ipp\_28, factor\_descuento], \  
 axis=1)  
ipp\_202204 = ipp\_py.loc[pd.to\_datetime('2022-04-01'), 'ipp']  
comer0 = 16.00 #Oferta de DICEL\_M2 para abril de 2022.  
# # #\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Ajusta el cargo de comercialización con el IPP para entre 2022-12 y 2023-12.  
#Crea columna 'comer'.  
egreso\_DICEL\_M2\_M5.insert(loc=len(egreso\_DICEL\_M2\_M5.columns), column='comer', \  
 value=np.NaN, allow\_duplicates=True)  
#  
egreso\_DICEL\_M2\_M5.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2028-12-01'), 'comer'] = \  
 egreso\_DICEL\_M2\_M5.ipp \* comer0/ipp\_202204  
# #\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Carga los precios fijos por periodo.  
egreso\_DICEL\_M2\_M5.insert(loc=len(egreso\_DICEL\_M2\_M5.columns), \  
 column='precio\_fijo', value=np.NaN, allow\_duplicates=True)  
#Carga precio\_fijo de 2024.  
egreso\_DICEL\_M2\_M5.loc[pd.to\_datetime('2024-01-01') : \  
 pd.to\_datetime('2024-12-01'), 'precio\_fijo'] = 294.00  
#Carga precio\_fijo de 2025.  
egreso\_DICEL\_M2\_M5.loc[pd.to\_datetime('2025-01-01') : \  
 pd.to\_datetime('2025-12-01'), 'precio\_fijo'] = 286.00  
#Carga precio\_fijo de 2026.  
egreso\_DICEL\_M2\_M5.loc[pd.to\_datetime('2026-01-01') : \  
 pd.to\_datetime('2026-12-01'), 'precio\_fijo'] = 280.00  
#Carga precio\_fijo de 2027.  
egreso\_DICEL\_M2\_M5.loc[pd.to\_datetime('2027-01-01') : \  
 pd.to\_datetime('2027-12-01'), 'precio\_fijo'] = 267.00  
#Carga precio\_fijo de 2028.  
egreso\_DICEL\_M2\_M5.loc[pd.to\_datetime('2028-01-01') : \  
 pd.to\_datetime('2028-12-01'), 'precio\_fijo'] = 263.00  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Ajusta el precio fijo con el IPP.  
#Crea columna 'precio\_fijo\_ipp'.  
egreso\_DICEL\_M2\_M5.insert(loc=len(egreso\_DICEL\_M2\_M5.columns), \  
 column='precio\_fijo\_ipp', value=np.NaN, allow\_duplicates=True)  
egreso\_DICEL\_M2\_M5.loc[:, 'precio\_fijo\_ipp'] = egreso\_DICEL\_M2\_M5.ipp \* \  
 egreso\_DICEL\_M2\_M5.precio\_fijo/ipp\_202204  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# # #Calcula el precio (COP/kWh) de energía en los tres escenarios (G+C).  
egreso\_DICEL\_M2\_M5.insert(loc=len(egreso\_DICEL\_M2\_M5.columns), \  
 column='precio\_bolsa\_alto', value=np.NaN, allow\_duplicates=True )  
egreso\_DICEL\_M2\_M5.insert(loc=len(egreso\_DICEL\_M2\_M5.columns), \  
 column='precio\_bolsa\_medio', value=np.NaN, allow\_duplicates=True )  
egreso\_DICEL\_M2\_M5.insert(loc=len(egreso\_DICEL\_M2\_M5.columns), \  
 column='precio\_bolsa\_bajo', value=np.NaN, allow\_duplicates=True )  
  
egreso\_DICEL\_M2\_M5.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2028-12-01'), 'precio\_bolsa\_alto'] = \  
 egreso\_DICEL\_M2\_M5.pbm\_alto + egreso\_DICEL\_M2\_M5.comer  
egreso\_DICEL\_M2\_M5.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2028-12-01'), 'precio\_bolsa\_medio'] = \  
 egreso\_DICEL\_M2\_M5.pbm\_medio + egreso\_DICEL\_M2\_M5.comer  
egreso\_DICEL\_M2\_M5.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2028-12-01'), 'precio\_bolsa\_bajo'] = \  
 egreso\_DICEL\_M2\_M5.pbm\_bajo + egreso\_DICEL\_M2\_M5.comer  
  
# # #\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# #Calcula el egreso de energía (G+C) en los tres escenarios (Millones de COP).  
#\*\*\*Crea columnas de egresos\*\*\*  
egreso\_DICEL\_M2\_M5.insert(loc=len(egreso\_DICEL\_M2\_M5.columns), \  
 column='egreso\_alto', value=np.NaN, allow\_duplicates=True )  
egreso\_DICEL\_M2\_M5.insert(loc=len(egreso\_DICEL\_M2\_M5.columns), \  
 column='egreso\_medio', value=np.NaN, allow\_duplicates=True )  
egreso\_DICEL\_M2\_M5.insert(loc=len(egreso\_DICEL\_M2\_M5.columns), \  
 column='egreso\_bajo', value=np.NaN, allow\_duplicates=True )  
#\*\*\*Egreso para periodo 2022-12 a 2023-12\*\*\*  
egreso\_DICEL\_M2\_M5.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2023-12-01'), 'egreso\_alto'] = \  
 egreso\_DICEL\_M2\_M5.precio\_bolsa\_alto \* egreso\_DICEL\_M2\_M5.demanda  
egreso\_DICEL\_M2\_M5.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2023-12-01'), 'egreso\_medio'] = \  
 egreso\_DICEL\_M2\_M5.precio\_bolsa\_medio \* egreso\_DICEL\_M2\_M5.demanda  
egreso\_DICEL\_M2\_M5.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2023-12-01'), 'egreso\_bajo'] = \  
 egreso\_DICEL\_M2\_M5.precio\_bolsa\_bajo \* egreso\_DICEL\_M2\_M5.demanda  
#\*\*\*Egreso para periodo 2024-01 a 2028-12\*\*\*  
egreso\_DICEL\_M2\_M5.loc[pd.to\_datetime('2024-01-01') : \  
 pd.to\_datetime('2028-12-01'), 'egreso\_alto'] = \  
 (egreso\_DICEL\_M2\_M5.precio\_fijo\_ipp \* demanda\_precio\_fijo + \  
 egreso\_DICEL\_M2\_M5.precio\_bolsa\_alto \* (egreso\_DICEL\_M2\_M5.demanda \  
 - demanda\_precio\_fijo))  
egreso\_DICEL\_M2\_M5.loc[pd.to\_datetime('2024-01-01') : \  
 pd.to\_datetime('2028-12-01'), 'egreso\_medio'] = \  
 (egreso\_DICEL\_M2\_M5.precio\_fijo\_ipp \* demanda\_precio\_fijo + \  
 egreso\_DICEL\_M2\_M5.precio\_bolsa\_medio \* (egreso\_DICEL\_M2\_M5.demanda \  
 - demanda\_precio\_fijo))  
egreso\_DICEL\_M2\_M5.loc[pd.to\_datetime('2024-01-01') : \  
 pd.to\_datetime('2028-12-01'), 'egreso\_bajo'] = \  
 (egreso\_DICEL\_M2\_M5.precio\_fijo\_ipp \* demanda\_precio\_fijo + \  
 egreso\_DICEL\_M2\_M5.precio\_bolsa\_bajo \* (egreso\_DICEL\_M2\_M5.demanda \  
 - demanda\_precio\_fijo))  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Calcula el egreso de energía descontado en los tres escenarios (Millones de COP).  
egreso\_DICEL\_M2\_M5.insert(loc=len(egreso\_DICEL\_M2\_M5.columns), \  
 column='egreso\_alto\_d', value=egreso\_DICEL\_M2\_M5.egreso\_alto \* \  
 egreso\_DICEL\_M2\_M5.factor\_descuento, allow\_duplicates=True )  
egreso\_DICEL\_M2\_M5.insert(loc=len(egreso\_DICEL\_M2\_M5.columns), \  
 column='egreso\_medio\_d', value=egreso\_DICEL\_M2\_M5.egreso\_medio \* \  
 egreso\_DICEL\_M2\_M5.factor\_descuento, allow\_duplicates=True )  
egreso\_DICEL\_M2\_M5.insert(loc=len(egreso\_DICEL\_M2\_M5.columns), \  
 column='egreso\_bajo\_d', value=egreso\_DICEL\_M2\_M5.egreso\_bajo \* \  
 egreso\_DICEL\_M2\_M5.factor\_descuento, allow\_duplicates=True )  
# #\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# #Carga el VPN de egresos en los tres escenarios en la estructura vpn\_ofertas.  
vpn\_ofertas.loc['DICEL\_M2\_M5', 'Escenario Alto'] = \  
 egreso\_DICEL\_M2\_M5.egreso\_alto\_d.sum()  
vpn\_ofertas.loc['DICEL\_M2\_M5', 'Escenario Medio'] = \  
 egreso\_DICEL\_M2\_M5.egreso\_medio\_d.sum()  
vpn\_ofertas.loc['DICEL\_M2\_M5', 'Escenario Bajo'] = \  
 egreso\_DICEL\_M2\_M5.egreso\_bajo\_d.sum()  
# #\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Carga el VPN de egresos en los tres escenarios en vpn\_ofertas\_p1p2.  
vpn\_ofertas\_p1p2.loc['DICEL\_M2\_M5', 'Escenario Alto'] = \  
 egreso\_DICEL\_M2\_M5.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2023-12-01'), 'egreso\_alto\_d'].sum()  
vpn\_ofertas\_p1p2.loc['DICEL\_M2\_M5', 'Escenario Medio'] = \  
 egreso\_DICEL\_M2\_M5.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2023-12-01'), 'egreso\_medio\_d'].sum()  
vpn\_ofertas\_p1p2.loc['DICEL\_M2\_M5', 'Escenario Bajo'] = \  
 egreso\_DICEL\_M2\_M5.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2023-12-01'), 'egreso\_bajo\_d'].sum()

### 5.5.4. Egreso de la oferta AIRE\_M1

AIRE ofertó para los periodos del 2022 al 2028 con la modalidad M1 de Precio Fijo. Los Precios Fijos son diferentes para cada periodo.

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 5.5.4.1. Se calcula egreso.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
egreso\_AIRE\_M1 = pd.concat([demanda, pbm\_py\_f, ipp\_28, factor\_descuento], \  
 axis=1)  
ipp\_202204 = ipp\_py.loc[pd.to\_datetime('2022-04-01'), 'ipp']  
# #\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Carga los precios fijos por periodo.  
egreso\_AIRE\_M1.insert(loc=len(egreso\_AIRE\_M1.columns), column='precio\_fijo', \  
 value=np.NaN, allow\_duplicates=True)  
#Carga precio\_fijo de 2022.  
egreso\_AIRE\_M1.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2022-12-01'), 'precio\_fijo'] = 280.00  
#Carga precio\_fijo de 2023.  
egreso\_AIRE\_M1.loc[pd.to\_datetime('2023-01-01') : \  
 pd.to\_datetime('2023-12-01'), 'precio\_fijo'] = 305.59  
#Carga precio\_fijo de 2024.  
egreso\_AIRE\_M1.loc[pd.to\_datetime('2024-01-01') : \  
 pd.to\_datetime('2024-12-01'), 'precio\_fijo'] = 298.21  
#Carga precio\_fijo de 2025.  
egreso\_AIRE\_M1.loc[pd.to\_datetime('2025-01-01') : \  
 pd.to\_datetime('2025-12-01'), 'precio\_fijo'] = 278.26  
#Carga precio\_fijo de 2026.  
egreso\_AIRE\_M1.loc[pd.to\_datetime('2026-01-01') : \  
 pd.to\_datetime('2026-12-01'), 'precio\_fijo'] = 266.03  
#Carga precio\_fijo de 2027.  
egreso\_AIRE\_M1.loc[pd.to\_datetime('2027-01-01') : \  
 pd.to\_datetime('2027-12-01'), 'precio\_fijo'] = 251.84  
#Carga precio\_fijo de 2028.  
egreso\_AIRE\_M1.loc[pd.to\_datetime('2028-01-01') : \  
 pd.to\_datetime('2028-12-01'), 'precio\_fijo'] = 250.18  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# Ajusta el precio fijo con el IPP.  
# Crea columna 'precio\_fijo\_ipp'.  
egreso\_AIRE\_M1.insert(loc=len(egreso\_AIRE\_M1.columns), \  
 column='precio\_fijo\_ipp', value=np.NaN, allow\_duplicates=True)  
egreso\_AIRE\_M1.loc[:, 'precio\_fijo\_ipp'] = egreso\_AIRE\_M1.ipp \* \  
 egreso\_AIRE\_M1.precio\_fijo/ipp\_202204  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Calcula el precio (COP/kWh) de energía en los tres escenarios (G+C).  
egreso\_AIRE\_M1.insert(loc=len(egreso\_AIRE\_M1.columns), column='precio\_alto', \  
 value=np.NaN, allow\_duplicates=True )  
egreso\_AIRE\_M1.insert(loc=len(egreso\_AIRE\_M1.columns), column='precio\_medio', \  
 value=np.NaN, allow\_duplicates=True )  
egreso\_AIRE\_M1.insert(loc=len(egreso\_AIRE\_M1.columns), column='precio\_bajo', \  
 value=np.NaN, allow\_duplicates=True )  
  
egreso\_AIRE\_M1.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2028-12-01'), 'precio\_alto'] = egreso\_AIRE\_M1.precio\_fijo\_ipp  
egreso\_AIRE\_M1.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2028-12-01'), 'precio\_medio'] = egreso\_AIRE\_M1.precio\_fijo\_ipp  
egreso\_AIRE\_M1.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2028-12-01'), 'precio\_bajo'] = egreso\_AIRE\_M1.precio\_fijo\_ipp  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Calcula el egreso de energía (G+C) en los tres escenarios (Millones de COP).  
egreso\_AIRE\_M1.insert(loc=len(egreso\_AIRE\_M1.columns), column='egreso\_alto', \  
 value=egreso\_AIRE\_M1.precio\_alto \* egreso\_AIRE\_M1.demanda, \  
 allow\_duplicates=True )  
egreso\_AIRE\_M1.insert(loc=len(egreso\_AIRE\_M1.columns), column='egreso\_medio', \  
 value=egreso\_AIRE\_M1.precio\_medio \* egreso\_AIRE\_M1.demanda, \  
 allow\_duplicates=True )  
egreso\_AIRE\_M1.insert(loc=len(egreso\_AIRE\_M1.columns), column='egreso\_bajo', \  
 value=egreso\_AIRE\_M1.precio\_bajo \* egreso\_AIRE\_M1.demanda, \  
 allow\_duplicates=True )  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Calcula el egreso de energía descontado en los tres escenarios (Millones de COP).  
egreso\_AIRE\_M1.insert(loc=len(egreso\_AIRE\_M1.columns), column='egreso\_alto\_d', \  
 value=egreso\_AIRE\_M1.egreso\_alto \* egreso\_AIRE\_M1.factor\_descuento, \  
 allow\_duplicates=True )  
egreso\_AIRE\_M1.insert(loc=len(egreso\_AIRE\_M1.columns), column='egreso\_medio\_d', \  
 value=egreso\_AIRE\_M1.egreso\_medio \* egreso\_AIRE\_M1.factor\_descuento, \  
 allow\_duplicates=True )  
egreso\_AIRE\_M1.insert(loc=len(egreso\_AIRE\_M1.columns), column='egreso\_bajo\_d', \  
 value=egreso\_AIRE\_M1.egreso\_bajo \* egreso\_AIRE\_M1.factor\_descuento, \  
 allow\_duplicates=True )  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Carga el VPN de egresos en los tres escenarios en la estructura vpn\_ofertas.  
vpn\_ofertas.loc['AIRE\_M1', 'Escenario Alto'] = \  
 egreso\_AIRE\_M1.egreso\_alto\_d.sum()  
vpn\_ofertas.loc['AIRE\_M1', 'Escenario Medio'] = \  
 egreso\_AIRE\_M1.egreso\_medio\_d.sum()  
vpn\_ofertas.loc['AIRE\_M1', 'Escenario Bajo'] = \  
 egreso\_AIRE\_M1.egreso\_bajo\_d.sum()  
# #\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Carga el VPN de egresos en los tres escenarios en vpn\_ofertas\_p1p2.  
vpn\_ofertas\_p1p2.loc['AIRE\_M1', 'Escenario Alto'] = \  
 egreso\_AIRE\_M1.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2023-12-01'), 'egreso\_alto\_d'].sum()  
vpn\_ofertas\_p1p2.loc['AIRE\_M1', 'Escenario Medio'] = \  
 egreso\_AIRE\_M1.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2023-12-01'), 'egreso\_medio\_d'].sum()  
vpn\_ofertas\_p1p2.loc['AIRE\_M1', 'Escenario Bajo'] = \  
 egreso\_AIRE\_M1.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2023-12-01'), 'egreso\_bajo\_d'].sum()

### 5.5.5. Egreso de la oferta AIRE\_M4

AIRE ofertó para los periodos del 2022 al 2028 con la modalidad M4 de 80% Precio Fijo y 20% de Precio de bolsa más Comercialización. Los Precios Fijos son diferentes para cada periodo.

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 5.5.5.1. Se calcula egreso.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
porcentaje\_precio\_fijo = 0.80  
porcentaje\_precio\_bolsa = 0.20  
egreso\_AIRE\_M4 = pd.concat([demanda, pbm\_py\_f, ipp\_28, factor\_descuento], \  
 axis=1)  
ipp\_202204 = ipp\_py.loc[pd.to\_datetime('2022-04-01'), 'ipp']  
comer0 = 2.00 #Oferta de AIRE\_M4 para abril de 2022. Este es K. No hay C?.  
# # #\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Ajusta el cargo de comercialización con el IPP para entre 2022-12 y 2023-12.  
#Crea columna 'comer'.  
egreso\_AIRE\_M4.insert(loc=len(egreso\_AIRE\_M4.columns), column='comer', \  
 value=np.NaN, allow\_duplicates=True)  
#  
egreso\_AIRE\_M4.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2028-12-01'), 'comer'] = \  
 egreso\_AIRE\_M4.ipp \* comer0/ipp\_202204  
# #\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Carga los precios fijos por periodo.  
egreso\_AIRE\_M4.insert(loc=len(egreso\_AIRE\_M4.columns), \  
 column='precio\_fijo', value=np.NaN, allow\_duplicates=True)  
#Carga precio\_fijo de 2022.  
egreso\_AIRE\_M4.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2022-12-01'), 'precio\_fijo'] = 280.20  
#Carga precio\_fijo de 2023.  
egreso\_AIRE\_M4.loc[pd.to\_datetime('2023-01-01') : \  
 pd.to\_datetime('2023-12-01'), 'precio\_fijo'] = 305.59  
#Carga precio\_fijo de 2024.  
egreso\_AIRE\_M4.loc[pd.to\_datetime('2024-01-01') : \  
 pd.to\_datetime('2024-12-01'), 'precio\_fijo'] = 298.21  
#Carga precio\_fijo de 2025.  
egreso\_AIRE\_M4.loc[pd.to\_datetime('2025-01-01') : \  
 pd.to\_datetime('2025-12-01'), 'precio\_fijo'] = 278.26  
#Carga precio\_fijo de 2026.  
egreso\_AIRE\_M4.loc[pd.to\_datetime('2026-01-01') : \  
 pd.to\_datetime('2026-12-01'), 'precio\_fijo'] = 266.03  
#Carga precio\_fijo de 2027.  
egreso\_AIRE\_M4.loc[pd.to\_datetime('2027-01-01') : \  
 pd.to\_datetime('2027-12-01'), 'precio\_fijo'] = 251.84  
#Carga precio\_fijo de 2028.  
egreso\_AIRE\_M4.loc[pd.to\_datetime('2028-01-01') : \  
 pd.to\_datetime('2028-12-01'), 'precio\_fijo'] = 250.18  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Ajusta el precio fijo con el IPP.  
#Crea columna 'precio\_fijo\_ipp'.  
egreso\_AIRE\_M4.insert(loc=len(egreso\_AIRE\_M4.columns), \  
 column='precio\_fijo\_ipp', value=np.NaN, allow\_duplicates=True)  
egreso\_AIRE\_M4.loc[:, 'precio\_fijo\_ipp'] = egreso\_AIRE\_M4.ipp \* \  
 egreso\_AIRE\_M4.precio\_fijo/ipp\_202204  
# #\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# # #Calcula el precio (COP/kWh) de energía en los tres escenarios (G+C).  
egreso\_AIRE\_M4.insert(loc=len(egreso\_AIRE\_M4.columns), \  
 column='precio\_bolsa\_alto', value=np.NaN, allow\_duplicates=True )  
egreso\_AIRE\_M4.insert(loc=len(egreso\_AIRE\_M4.columns), \  
 column='precio\_bolsa\_medio', value=np.NaN, allow\_duplicates=True )  
egreso\_AIRE\_M4.insert(loc=len(egreso\_AIRE\_M4.columns), \  
 column='precio\_bolsa\_bajo', value=np.NaN, allow\_duplicates=True )  
  
egreso\_AIRE\_M4.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2028-12-01'), 'precio\_bolsa\_alto'] = \  
 egreso\_AIRE\_M4.pbm\_alto + egreso\_AIRE\_M4.comer  
egreso\_AIRE\_M4.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2028-12-01'), 'precio\_bolsa\_medio'] = \  
 egreso\_AIRE\_M4.pbm\_medio + egreso\_AIRE\_M4.comer  
egreso\_AIRE\_M4.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2028-12-01'), 'precio\_bolsa\_bajo'] = \  
 egreso\_AIRE\_M4.pbm\_bajo + egreso\_AIRE\_M4.comer  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# #Calcula el egreso de energía (G+C) en los tres escenarios (Millones de COP).  
#\*\*\*Crea columnas de egresos\*\*\*  
egreso\_AIRE\_M4.insert(loc=len(egreso\_AIRE\_M4.columns), \  
 column='egreso\_alto', value=np.NaN, allow\_duplicates=True )  
egreso\_AIRE\_M4.insert(loc=len(egreso\_AIRE\_M4.columns), \  
 column='egreso\_medio', value=np.NaN, allow\_duplicates=True )  
egreso\_AIRE\_M4.insert(loc=len(egreso\_AIRE\_M4.columns), \  
 column='egreso\_bajo', value=np.NaN, allow\_duplicates=True )  
#\*\*\*Egreso para periodo 2022-12 a 2028-12\*\*\*  
egreso\_AIRE\_M4.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2028-12-01'), 'egreso\_alto'] = \  
 (egreso\_AIRE\_M4.precio\_fijo\_ipp \* porcentaje\_precio\_fijo \* egreso\_AIRE\_M4.demanda + \  
 egreso\_AIRE\_M4.precio\_bolsa\_alto \* porcentaje\_precio\_bolsa \* egreso\_AIRE\_M4.demanda)  
egreso\_AIRE\_M4.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2028-12-01'), 'egreso\_medio'] = \  
 (egreso\_AIRE\_M4.precio\_fijo\_ipp \* porcentaje\_precio\_fijo \* egreso\_AIRE\_M4.demanda + \  
 egreso\_AIRE\_M4.precio\_bolsa\_medio \* porcentaje\_precio\_bolsa \* egreso\_AIRE\_M4.demanda)  
egreso\_AIRE\_M4.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2028-12-01'), 'egreso\_bajo'] = \  
 (egreso\_AIRE\_M4.precio\_fijo\_ipp \* porcentaje\_precio\_fijo \* egreso\_AIRE\_M4.demanda + \  
 egreso\_AIRE\_M4.precio\_bolsa\_medio \* porcentaje\_precio\_bolsa \* egreso\_AIRE\_M4.demanda)  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Calcula el egreso de energía descontado en los tres escenarios (Millones de COP).  
egreso\_AIRE\_M4.insert(loc=len(egreso\_AIRE\_M4.columns), \  
 column='egreso\_alto\_d', value=egreso\_AIRE\_M4.egreso\_alto \* \  
 egreso\_AIRE\_M4.factor\_descuento, allow\_duplicates=True )  
egreso\_AIRE\_M4.insert(loc=len(egreso\_AIRE\_M4.columns), \  
 column='egreso\_medio\_d', value=egreso\_AIRE\_M4.egreso\_medio \* \  
 egreso\_AIRE\_M4.factor\_descuento, allow\_duplicates=True )  
egreso\_AIRE\_M4.insert(loc=len(egreso\_AIRE\_M4.columns), \  
 column='egreso\_bajo\_d', value=egreso\_AIRE\_M4.egreso\_bajo \* \  
 egreso\_AIRE\_M4.factor\_descuento, allow\_duplicates=True )  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# #Carga el VPN de egresos en los tres escenarios en la estructura vpn\_ofertas.  
vpn\_ofertas.loc['AIRE\_M4', 'Escenario Alto'] = \  
 egreso\_AIRE\_M4.egreso\_alto\_d.sum()  
vpn\_ofertas.loc['AIRE\_M4', 'Escenario Medio'] = \  
 egreso\_AIRE\_M4.egreso\_medio\_d.sum()  
vpn\_ofertas.loc['AIRE\_M4', 'Escenario Bajo'] = \  
 egreso\_AIRE\_M4.egreso\_bajo\_d.sum()  
# #\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Carga el VPN de egresos en los tres escenarios en vpn\_ofertas\_p1p2.  
vpn\_ofertas\_p1p2.loc['AIRE\_M4', 'Escenario Alto'] = \  
 egreso\_AIRE\_M4.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2023-12-01'), 'egreso\_alto\_d'].sum()  
vpn\_ofertas\_p1p2.loc['AIRE\_M4', 'Escenario Medio'] = \  
 egreso\_AIRE\_M4.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2023-12-01'), 'egreso\_medio\_d'].sum()  
vpn\_ofertas\_p1p2.loc['AIRE\_M4', 'Escenario Bajo'] = \  
 egreso\_AIRE\_M4.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2023-12-01'), 'egreso\_bajo\_d'].sum()

### 5.5.6. Egreso de la oferta EMSA\_M2

EMSA solo ofertó para los periodos 2022 y 2023, con la modalidad M2 de Precio de Bolsa Horario más un cargo de Comercialización de 18.00 COP/kWh.

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 5.5.6.1. Se calcula egreso.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
egreso\_EMSA\_M2 = pd.concat([demanda, pbm\_py\_f, ipp\_28, factor\_descuento], \  
 axis=1)  
ipp\_202204 = ipp\_py.loc[pd.to\_datetime('2022-04-01'), 'ipp']  
comer0 = 18.00 #Oferta de EMSA\_M2 para abril de 2022. Sólo 2022 y 2023.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Ajusta el cargo de comercialización con el IPP para entre 2022-12 y 2023-12.  
#Crea columna 'comer'.  
egreso\_EMSA\_M2.insert(loc=len(egreso\_EMSA\_M2.columns), column='comer', \  
 value=np.NaN, allow\_duplicates=True)  
egreso\_EMSA\_M2.loc[pd.to\_datetime('2022-12-01') : pd.to\_datetime('2023-12-01'), 'comer'] = egreso\_EMSA\_M2.ipp \* comer0/ipp\_202204  
# #Calcula el precio (COP/kWh) de energía en los tres escenarios (G+C).  
egreso\_EMSA\_M2.insert(loc=len(egreso\_EMSA\_M2.columns), column='precio\_alto', \  
 value=egreso\_EMSA\_M2.pbm\_alto + egreso\_EMSA\_M2.comer, allow\_duplicates=True )  
egreso\_EMSA\_M2.insert(loc=len(egreso\_EMSA\_M2.columns), column='precio\_medio', \  
 value=egreso\_EMSA\_M2.pbm\_medio + egreso\_EMSA\_M2.comer, allow\_duplicates=True )  
egreso\_EMSA\_M2.insert(loc=len(egreso\_EMSA\_M2.columns), column='precio\_bajo', \  
 value=egreso\_EMSA\_M2.pbm\_bajo + egreso\_EMSA\_M2.comer, allow\_duplicates=True )  
# #\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# #Calcula el egreso de energía (G+C) en los tres escenarios (Millones de COP).  
egreso\_EMSA\_M2.insert(loc=len(egreso\_EMSA\_M2.columns), column='egreso\_alto', \  
 value=egreso\_EMSA\_M2.precio\_alto \* egreso\_EMSA\_M2.demanda, allow\_duplicates=True )  
egreso\_EMSA\_M2.insert(loc=len(egreso\_EMSA\_M2.columns), column='egreso\_medio', \  
 value=egreso\_EMSA\_M2.precio\_medio \* egreso\_EMSA\_M2.demanda, allow\_duplicates=True )  
egreso\_EMSA\_M2.insert(loc=len(egreso\_EMSA\_M2.columns), column='egreso\_bajo', \  
 value=egreso\_EMSA\_M2.precio\_bajo \* egreso\_EMSA\_M2.demanda, allow\_duplicates=True )  
# #\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# #Calcula el egreso de energía descontado en los tres escenarios (Millones de COP).  
egreso\_EMSA\_M2.insert(loc=len(egreso\_EMSA\_M2.columns), column='egreso\_alto\_d', \  
 value=egreso\_EMSA\_M2.egreso\_alto \* egreso\_EMSA\_M2.factor\_descuento, \  
 allow\_duplicates=True )  
egreso\_EMSA\_M2.insert(loc=len(egreso\_EMSA\_M2.columns), column='egreso\_medio\_d', \  
 value=egreso\_EMSA\_M2.egreso\_medio \* egreso\_EMSA\_M2.factor\_descuento, \  
 allow\_duplicates=True )  
egreso\_EMSA\_M2.insert(loc=len(egreso\_EMSA\_M2.columns), column='egreso\_bajo\_d', \  
 value=egreso\_EMSA\_M2.egreso\_bajo \* egreso\_EMSA\_M2.factor\_descuento, \  
 allow\_duplicates=True )  
# #\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Carga el VPN de egresos en los tres escenarios en vpn\_ofertas\_p1p2.  
vpn\_ofertas\_p1p2.loc['EMSA\_M2', 'Escenario Alto'] = \  
 egreso\_EMSA\_M2.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2023-12-01'), 'egreso\_alto\_d'].sum()  
vpn\_ofertas\_p1p2.loc['EMSA\_M2', 'Escenario Medio'] = \  
 egreso\_EMSA\_M2.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2023-12-01'), 'egreso\_medio\_d'].sum()  
vpn\_ofertas\_p1p2.loc['EMSA\_M2', 'Escenario Bajo'] = \  
 egreso\_EMSA\_M2.loc[pd.to\_datetime('2022-12-01') : \  
 pd.to\_datetime('2023-12-01'), 'egreso\_bajo\_d'].sum()

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 5.5.6.2. Salva vpn\_ofertas y vpn\_ofertas\_p1p2.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# #Salva vpn\_ofertas y vpn\_ofertas\_p1p2  
vpn\_ofertas.to\_parquet('parquet/vpn\_ofertas.parquet.gzip', \  
 compression='gzip', engine='fastparquet')  
vpn\_ofertas\_p1p2.to\_parquet('parquet/vpn\_ofertas\_p1p2.parquet.gzip', \  
 compression='gzip', engine='fastparquet')  
#Lee vpn\_ofertas y vpn\_ofertas\_p1p2  
vpn\_ofertas = pd.read\_parquet('parquet/vpn\_ofertas.parquet.gzip', \  
 engine='fastparquet')  
vpn\_ofertas\_p1p2 = pd.read\_parquet('parquet/vpn\_ofertas\_p1p2.parquet.gzip', \  
 engine='fastparquet')

## 5.6. Minimización de Máximo Arrepentimiento

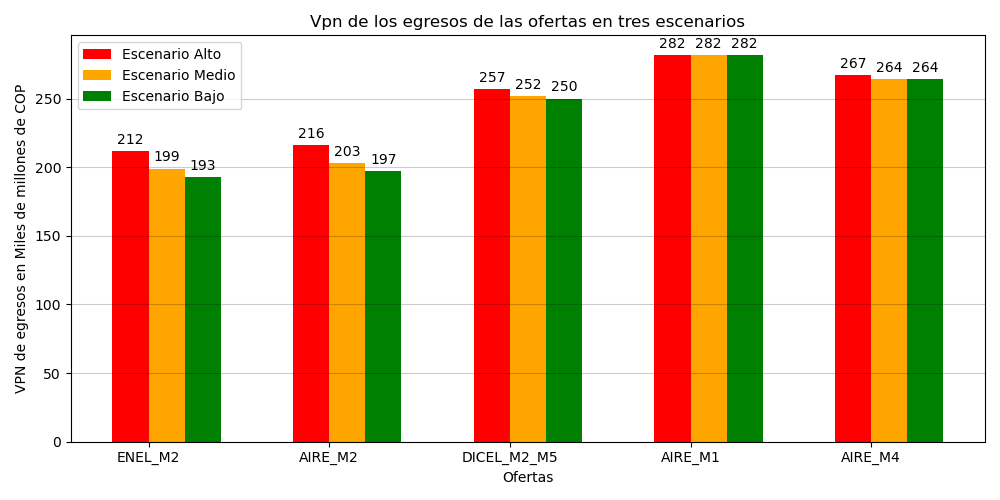
Se aplicará la metodología de la minimización de máximo arrepentimiento a los Vpn de los egresos de las ofertas.

La oferta de EMSA solo se presentó para los periodos P1 y P2. Por ello, se retira de la comparación inicial.

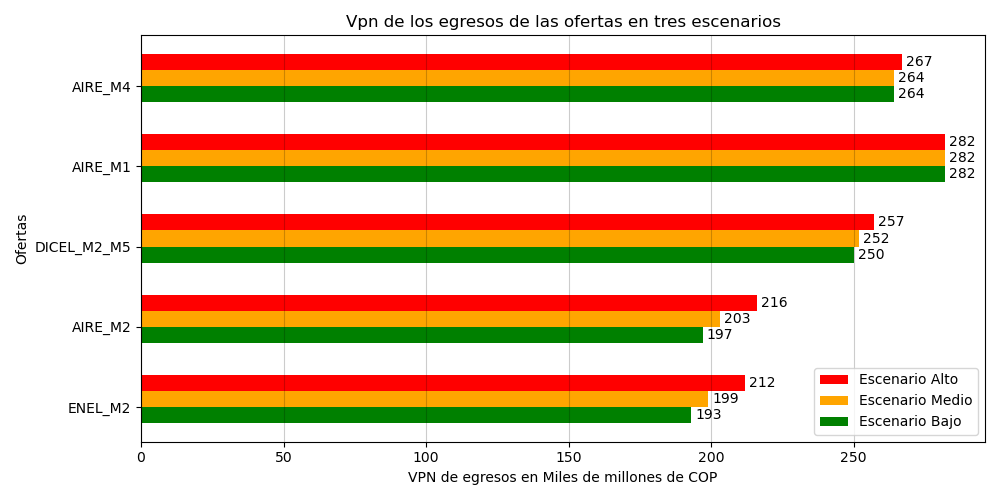
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 5.6.1. Lee vpn\_ofertas.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# # #Salva vpn\_ofertas  
# vpn\_ofertas.to\_parquet('parquet/vpn\_ofertas.parquet.gzip', \  
# compression='gzip', engine='fastparquet')  
#Lee vpn\_ofertas  
vpn\_ofertas = pd.read\_parquet('parquet/vpn\_ofertas.parquet.gzip', engine='fastparquet')

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 5.6.2. Elimina la oferta EMSA\_M2.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
vpn\_ofertas = vpn\_ofertas.drop(index='EMSA\_M2', axis=0)  
vpn\_ofertas = vpn\_ofertas/1e3 #Miles de millones de pesos.

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# # 5.6.3. Grafica de Vpn de las ofertas en tres escenarios (vertical).  
# #\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig, ax = plt.subplots(figsize=(10,5))  
  
ofertas = vpn\_ofertas.index  
escenario\_alto = vpn\_ofertas['Escenario Alto'].astype(int)  
escenario\_medio = vpn\_ofertas['Escenario Medio'].astype(int)  
escenario\_bajo = vpn\_ofertas['Escenario Bajo'].astype(int)  
   
x = np.arange(len(escenario\_alto)) # the label locations  
width = 0.20 # the width of the bars  
  
rects1 = ax.bar(x - width/2, escenario\_alto, width, label='Escenario Alto', \  
 color='r')  
rects2 = ax.bar(x + width/2, escenario\_medio, width, label='Escenario Medio', \  
 color='orange')  
rects3 = ax.bar(x + 3 \* width/2, escenario\_bajo, width, label='Escenario Bajo', \  
 color='g')  
  
# Add some text for labels, title and custom x-axis tick labels, etc.  
ax.set\_title('Vpn de los egresos de las ofertas en tres escenarios')  
ax.set\_xlabel('Ofertas')  
ax.set\_ylabel('VPN de egresos en Miles de millones de COP')  
ax.set\_xticks(x, ofertas)  
ax.legend()  
  
ax.bar\_label(rects1, padding=3)  
ax.bar\_label(rects2, padding=3)  
ax.bar\_label(rects3, padding=3)  
  
fig.tight\_layout()  
  
plt.grid(axis='y', color='k', alpha=0.2)  
plt.show()



#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 5.6.4. Grafica de Vpn de las ofertas en tres escenarios (horizontal).  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig, ax = plt.subplots(figsize=(10,5))  
  
ofertas = vpn\_ofertas.index  
escenario\_alto = vpn\_ofertas['Escenario Alto'].astype(int)  
escenario\_medio = vpn\_ofertas['Escenario Medio'].astype(int)  
escenario\_bajo = vpn\_ofertas['Escenario Bajo'].astype(int)  
   
x = np.arange(len(escenario\_alto)) # the label locations  
width = 0.20 # the width of the bars  
  
rects1 = ax.barh(x - width/2, escenario\_bajo, width, label='Escenario Bajo', \  
 color='g')  
rects2 = ax.barh(x + width/2, escenario\_medio, width, label='Escenario Medio', \  
 color='orange')  
rects3 = ax.barh(x + 3 \* width/2, escenario\_alto, width, label='Escenario Alto', \  
 color='r')  
  
# Add some text for labels, title and custom x-axis tick labels, etc.  
ax.set\_title('Vpn de los egresos de las ofertas en tres escenarios')  
ax.set\_xlabel('VPN de egresos en Miles de millones de COP')  
ax.set\_ylabel('Ofertas')  
ax.set\_yticks(x, ofertas)  
  
ax.bar\_label(rects1, padding=3)  
ax.bar\_label(rects2, padding=3)  
ax.bar\_label(rects3, padding=3)  
  
#obtiene los manejadores y etiquetas  
handles, labels = plt.gca().get\_legend\_handles\_labels()  
  
#especific wl orden de los elementos en la leyenda  
order = [2,1,0]  
  
#agrega la leyenda al gráfico  
ax.legend([handles[idx] for idx in order],[labels[idx] for idx in order])   
  
fig.tight\_layout()  
  
plt.grid(axis='x', color='k', alpha=0.2)  
plt.show()



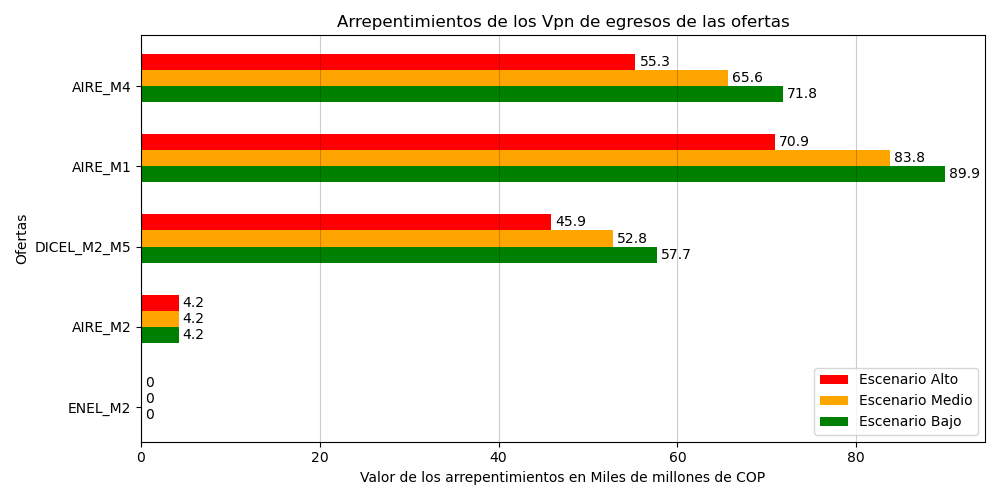
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 5.6.5. Cálculo del Mínimo del Máximo arrepentimiento.  
# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
minimo = pd.DataFrame({'Escenario Alto': vpn\_ofertas['Escenario Alto'].min(), \  
 'Escenario Medio': vpn\_ofertas['Escenario Medio'].min(), \  
 'Escenario Bajo': vpn\_ofertas['Escenario Bajo'].min()}, index=['Mínimo'])  
minimo  
arrepentimientos = pd.concat([vpn\_ofertas, minimo])  
arrepentimientos.insert(loc=len(arrepentimientos.columns), \  
 column='arrepentimiento\_alto', value=arrepentimientos['Escenario Alto'] - \  
 arrepentimientos.loc['Mínimo', 'Escenario Alto'])  
arrepentimientos.insert(loc=len(arrepentimientos.columns), \  
 column='arrepentimiento\_medio', value=arrepentimientos['Escenario Medio'] - \  
 arrepentimientos.loc['Mínimo', 'Escenario Medio'])  
arrepentimientos.insert(loc=len(arrepentimientos.columns), \  
 column='arrepentimiento\_bajo', value=arrepentimientos['Escenario Bajo'] - \  
 arrepentimientos.loc['Mínimo', 'Escenario Bajo'])  
arrepentimientos.insert(loc=len(arrepentimientos.columns), \  
 column='maximo\_arrepentimiento',   
 value=arrepentimientos.loc[:, ['arrepentimiento\_alto', 'arrepentimiento\_medio', 'arrepentimiento\_bajo']].max(axis=1))  
arrepentimientos

Escenario Alto Escenario Medio Escenario Bajo \  
ENEL\_M2 212.059584 199.227905 193.053734   
AIRE\_M2 216.286930 203.455251 197.281080   
DICEL\_M2\_M5 257.970889 252.062473 250.759887   
AIRE\_M1 282.991216 282.991216 282.991216   
AIRE\_M4 267.384079 264.817743 264.817743   
Mínimo 212.059584 199.227905 193.053734   
  
 arrepentimiento\_alto arrepentimiento\_medio \  
ENEL\_M2 0.000000 0.000000   
AIRE\_M2 4.227346 4.227346   
DICEL\_M2\_M5 45.911306 52.834568   
AIRE\_M1 70.931633 83.763311   
AIRE\_M4 55.324495 65.589838   
Mínimo 0.000000 0.000000   
  
 arrepentimiento\_bajo maximo\_arrepentimiento   
ENEL\_M2 0.000000 0.000000   
AIRE\_M2 4.227346 4.227346   
DICEL\_M2\_M5 57.706153 57.706153   
AIRE\_M1 89.937482 89.937482   
AIRE\_M4 71.764009 71.764009   
Mínimo 0.000000 0.000000

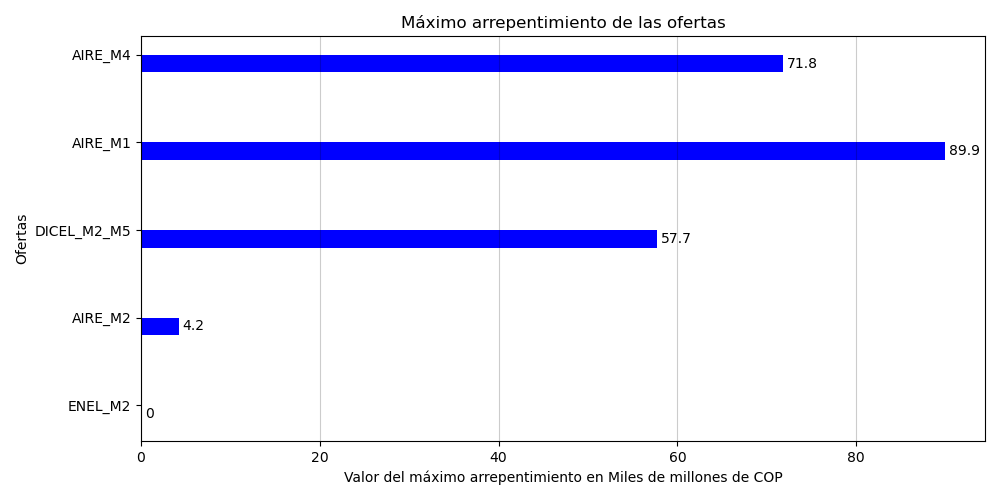
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 5.6.6. Guarda y lee arrepentimientos.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# # #Salva arrepentimientos  
# arrepentimientos.to\_parquet('parquet/arrepentimientos.parquet.gzip', \  
# compression='gzip', engine='fastparquet')  
#Lee arrepentimientos  
arrepentimientos = pd.read\_parquet('parquet/arrepentimientos.parquet.gzip', engine='fastparquet')

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 5.6.8. Quita la fila de Minimo.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
arrepentimientos = arrepentimientos.drop(index='Mínimo', axis=0)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 5.6.9. Grafica los arrepentimientos.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig, ax = plt.subplots(figsize=(10,5))  
  
ofertas = arrepentimientos.index  
escenario\_alto = arrepentimientos['arrepentimiento\_alto'].round(1)  
escenario\_medio = arrepentimientos['arrepentimiento\_medio'].round(1)  
escenario\_bajo = arrepentimientos['arrepentimiento\_bajo'].round(1)  
   
x = np.arange(len(escenario\_alto)) # the label locations  
width = 0.20 # the width of the bars  
  
rects1 = ax.barh(x - width/2, escenario\_bajo, width, label='Escenario Bajo', \  
 color='g')  
rects2 = ax.barh(x + width/2, escenario\_medio, width, label='Escenario Medio', \  
 color='orange')  
rects3 = ax.barh(x + 3 \* width/2, escenario\_alto, width, label='Escenario Alto', \  
 color='r')  
  
# Add some text for labels, title and custom x-axis tick labels, etc.  
ax.set\_title('Arrepentimientos de los Vpn de egresos de las ofertas')  
ax.set\_xlabel('Valor de los arrepentimientos en Miles de millones de COP')  
ax.set\_ylabel('Ofertas')  
ax.set\_yticks(x, ofertas)  
  
ax.bar\_label(rects1, padding=3)  
ax.bar\_label(rects2, padding=3)  
ax.bar\_label(rects3, padding=3)  
  
#obtiene los manejadores y etiquetas  
handles, labels = plt.gca().get\_legend\_handles\_labels()  
  
#especific wl orden de los elementos en la leyenda  
order = [2,1,0]  
  
#agrega la leyenda al gráfico  
ax.legend([handles[idx] for idx in order],[labels[idx] for idx in order], loc=4)   
  
fig.tight\_layout()  
  
plt.grid(axis='x', color='k', alpha=0.2)  
plt.show()



#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 5.6.10. Grafica del máximo arrepentimiento de cada oferta.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig, ax = plt.subplots(figsize=(10,5))  
  
ofertas = arrepentimientos.index  
escenario\_alto = arrepentimientos['maximo\_arrepentimiento'].round(1)  
   
x = np.arange(len(escenario\_alto)) # the label locations  
width = 0.20 # the width of the bars  
  
rects1 = ax.barh(x - width/2, escenario\_bajo, width, label='Escenario Bajo', \  
 color='b')  
  
# Add some text for labels, title and custom x-axis tick labels, etc.  
ax.set\_title('Máximo arrepentimiento de las ofertas')  
ax.set\_xlabel('Valor del máximo arrepentimiento en Miles de millones de COP')  
ax.set\_ylabel('Ofertas')  
ax.set\_yticks(x, ofertas)  
  
ax.bar\_label(rects1, padding=3)  
# ax.bar\_label(rects2, padding=3)  
# ax.bar\_label(rects3, padding=3)  
  
# #obtiene los manejadores y etiquetas  
# handles, labels = plt.gca().get\_legend\_handles\_labels()  
  
# #especific wl orden de los elementos en la leyenda  
# order = [2,1,0]  
  
# #agrega la leyenda al gráfico  
# ax.legend([handles[idx] for idx in order],[labels[idx] for idx in order])   
  
fig.tight\_layout()  
  
plt.grid(axis='x', color='k', alpha=0.2)  
plt.show()



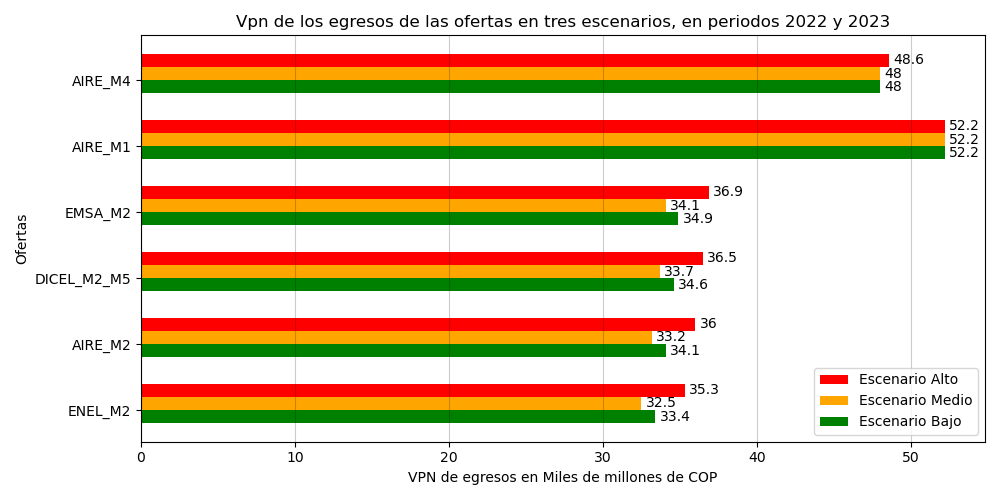
## 5.7. Análisis de la oferta de EMSA\_M2 en periodos P1 y P2

Se analizará la conveniencia de la oferta de EMSA en los periodos P1 y P2, al compararla con las otras ofertas en esos dos periodos.

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 5.7.1. Lee vpn\_ofertas\_p1p2.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#Lee vpn\_ofertas\_p1p2  
vpn\_ofertas\_p1p2 = pd.read\_parquet('parquet/vpn\_ofertas\_p1p2.parquet.gzip', engine='fastparquet')

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 5.7.2. Convierte el VPN a Miles de millones de pesos.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
vpn\_ofertas\_p1p2 = vpn\_ofertas\_p1p2/1e3 #Miles de millones de pesos.

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 5.7.3. Grafica de Vpn de las ofertas en 2022 y 2023.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig, ax = plt.subplots(figsize=(10,5))  
  
ofertas = vpn\_ofertas\_p1p2.index  
escenario\_alto = vpn\_ofertas\_p1p2['Escenario Alto'].round(1)  
escenario\_medio = vpn\_ofertas\_p1p2['Escenario Medio'].round(1)  
escenario\_bajo = vpn\_ofertas\_p1p2['Escenario Bajo'].round(1)  
   
x = np.arange(len(escenario\_alto)) # the label locations  
width = 0.20 # the width of the bars  
  
rects1 = ax.barh(x - width/2, escenario\_bajo, width, label='Escenario Bajo', \  
 color='g')  
rects2 = ax.barh(x + width/2, escenario\_medio, width, label='Escenario Medio', \  
 color='orange')  
rects3 = ax.barh(x + 3 \* width/2, escenario\_alto, width, label='Escenario Alto', \  
 color='r')  
  
# Add some text for labels, title and custom x-axis tick labels, etc.  
ax.set\_title('Vpn de los egresos de las ofertas en tres escenarios, en periodos 2022 y 2023')  
ax.set\_xlabel('VPN de egresos en Miles de millones de COP')  
ax.set\_ylabel('Ofertas')  
ax.set\_yticks(x, ofertas)  
  
ax.bar\_label(rects1, padding=3)  
ax.bar\_label(rects2, padding=3)  
ax.bar\_label(rects3, padding=3)  
  
#obtiene los manejadores y etiquetas  
handles, labels = plt.gca().get\_legend\_handles\_labels()  
  
#especific wl orden de los elementos en la leyenda  
order = [2,1,0]  
  
#agrega la leyenda al gráfico  
ax.legend([handles[idx] for idx in order],[labels[idx] for idx in order])   
  
fig.tight\_layout()  
  
plt.grid(axis='x', color='k', alpha=0.2)  
plt.show()



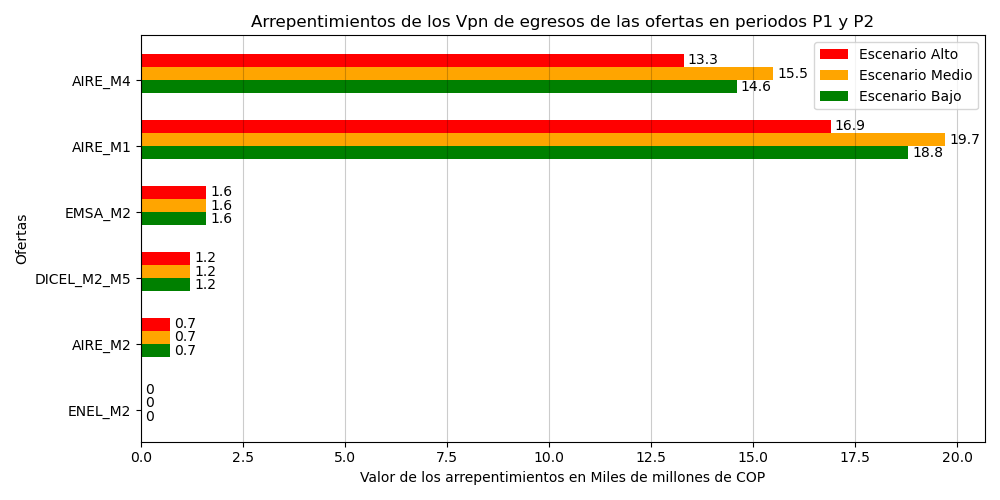
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 5.7.4. Cálculo del Mínimo del Máximo arrepentimiento en P1 y P2.  
# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
minimo = pd.DataFrame({'Escenario Alto': vpn\_ofertas\_p1p2['Escenario Alto'].min(), \  
 'Escenario Medio': vpn\_ofertas\_p1p2['Escenario Medio'].min(), \  
 'Escenario Bajo': vpn\_ofertas\_p1p2['Escenario Bajo'].min()}, index=['Mínimo'])  
  
arrepentimientos\_p1p2 = pd.concat([vpn\_ofertas\_p1p2, minimo])  
arrepentimientos\_p1p2.insert(loc=len(arrepentimientos\_p1p2.columns), \  
 column='arrepentimiento\_alto', value=arrepentimientos\_p1p2['Escenario Alto'] - \  
 arrepentimientos\_p1p2.loc['Mínimo', 'Escenario Alto'])  
arrepentimientos\_p1p2.insert(loc=len(arrepentimientos\_p1p2.columns), \  
 column='arrepentimiento\_medio', value=arrepentimientos\_p1p2['Escenario Medio'] - \  
 arrepentimientos\_p1p2.loc['Mínimo', 'Escenario Medio'])  
arrepentimientos\_p1p2.insert(loc=len(arrepentimientos\_p1p2.columns), \  
 column='arrepentimiento\_bajo', value=arrepentimientos\_p1p2['Escenario Bajo'] - \  
 arrepentimientos\_p1p2.loc['Mínimo', 'Escenario Bajo'])  
arrepentimientos\_p1p2.insert(loc=len(arrepentimientos\_p1p2.columns), \  
 column='maximo\_arrepentimiento',   
 value=arrepentimientos\_p1p2.loc[:, ['arrepentimiento\_alto', 'arrepentimiento\_medio', 'arrepentimiento\_bajo']].max(axis=1))  
arrepentimientos\_p1p2

Escenario Alto Escenario Medio Escenario Bajo \  
ENEL\_M2 35.304674 32.494418 33.371865   
AIRE\_M2 36.009346 33.199090 34.076538   
DICEL\_M2\_M5 36.524959 33.714703 34.592151   
EMSA\_M2 36.868702 34.058446 34.935894   
AIRE\_M1 52.193959 52.193959 52.193959   
AIRE\_M4 48.580971 48.018920 48.018920   
Mínimo 35.304674 32.494418 33.371865   
  
 arrepentimiento\_alto arrepentimiento\_medio \  
ENEL\_M2 0.000000 0.000000   
AIRE\_M2 0.704672 0.704672   
DICEL\_M2\_M5 1.220286 1.220286   
EMSA\_M2 1.564028 1.564028   
AIRE\_M1 16.889285 19.699541   
AIRE\_M4 13.276298 15.524502   
Mínimo 0.000000 0.000000   
  
 arrepentimiento\_bajo maximo\_arrepentimiento   
ENEL\_M2 0.000000 0.000000   
AIRE\_M2 0.704672 0.704672   
DICEL\_M2\_M5 1.220286 1.220286   
EMSA\_M2 1.564028 1.564028   
AIRE\_M1 18.822093 19.699541   
AIRE\_M4 14.647054 15.524502   
Mínimo 0.000000 0.000000

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 5.7.5. Guarda y lee arrepentimientos\_p1p2.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# # #Salva arrepentimientos  
# arrepentimientos\_p1p2.to\_parquet('parquet/arrepentimientos\_p1p2.parquet.gzip', \  
# compression='gzip', engine='fastparquet')  
#Lee arrepentimientos  
arrepentimientos\_p1p2 = \  
 pd.read\_parquet('parquet/arrepentimientos\_p1p2.parquet.gzip', \  
 engine='fastparquet')

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
# 5.7.6. Quita la fila de Minimo.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
arrepentimientos\_p1p2 = arrepentimientos\_p1p2.drop(index='Mínimo', axis=0)

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 5.7.7. Grafica los arrepentimientos de los periodos P1 y P2.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig, ax = plt.subplots(figsize=(10,5))  
  
ofertas = arrepentimientos\_p1p2.index  
escenario\_alto = arrepentimientos\_p1p2['arrepentimiento\_alto'].round(1)  
escenario\_medio = arrepentimientos\_p1p2['arrepentimiento\_medio'].round(1)  
escenario\_bajo = arrepentimientos\_p1p2['arrepentimiento\_bajo'].round(1)  
   
x = np.arange(len(escenario\_alto)) # the label locations  
width = 0.20 # the width of the bars  
  
rects1 = ax.barh(x - width/2, escenario\_bajo, width, label='Escenario Bajo', \  
 color='g')  
rects2 = ax.barh(x + width/2, escenario\_medio, width, label='Escenario Medio', \  
 color='orange')  
rects3 = ax.barh(x + 3 \* width/2, escenario\_alto, width, label='Escenario Alto', \  
 color='r')  
  
# Add some text for labels, title and custom x-axis tick labels, etc.  
ax.set\_title('Arrepentimientos de los Vpn de egresos de las ofertas en periodos P1 y P2')  
ax.set\_xlabel('Valor de los arrepentimientos en Miles de millones de COP')  
ax.set\_ylabel('Ofertas')  
ax.set\_yticks(x, ofertas)  
  
ax.bar\_label(rects1, padding=3)  
ax.bar\_label(rects2, padding=3)  
ax.bar\_label(rects3, padding=3)  
  
#obtiene los manejadores y etiquetas  
handles, labels = plt.gca().get\_legend\_handles\_labels()  
  
#especific wl orden de los elementos en la leyenda  
order = [2,1,0]  
  
#agrega la leyenda al gráfico  
ax.legend([handles[idx] for idx in order],[labels[idx] for idx in order])   
  
fig.tight\_layout()  
  
plt.grid(axis='x', color='k', alpha=0.2)  
plt.show()



#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
# 5.7.8. Grafica del máximo arrepentimiento de cada oferta.  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
fig, ax = plt.subplots(figsize=(10,5))  
  
ofertas = arrepentimientos\_p1p2.index  
maximo\_arrepentimiento = arrepentimientos\_p1p2['maximo\_arrepentimiento'].round(1)  
   
x = np.arange(len(escenario\_alto)) # the label locations  
width = 0.20 # the width of the bars  
  
rects1 = ax.barh(x - width/2, maximo\_arrepentimiento, width, label='Máximo arrepentimiento', \  
 color='b')  
  
# Add some text for labels, title and custom x-axis tick labels, etc.  
ax.set\_title('Máximo arrepentimiento de las ofertas en periodos P1 y P2')  
ax.set\_xlabel('Valor del máximo arrepentimiento en Miles de millones de COP')  
ax.set\_ylabel('Ofertas')  
ax.set\_yticks(x, ofertas)  
  
ax.bar\_label(rects1, padding=3)  
# ax.bar\_label(rects2, padding=3)  
# ax.bar\_label(rects3, padding=3)  
  
# #obtiene los manejadores y etiquetas  
# handles, labels = plt.gca().get\_legend\_handles\_labels()  
  
# #especific wl orden de los elementos en la leyenda  
# order = [2,1,0]  
  
# #agrega la leyenda al gráfico  
# ax.legend([handles[idx] for idx in order],[labels[idx] for idx in order])   
  
fig.tight\_layout()  
  
plt.grid(axis='x', color='k', alpha=0.2)  
plt.show()

