😎 Análisis de sensibilidad gráfica

En investigación de operaciones, existen a grandes rasgos, dos tipos de análisis complementarios en PL:

- **Análisis de sensibilidad**: El cual determina las condiciones que mantendrán la solución actual sin cambios.
- **Análisis postóptimo**: El cual determina la nueva solución óptima cuando cambian los datos del modelo.

El análisis de sensibilidad en programación lineal corresponde al examen detallado de los límites dentro de los cuales los parámetros del modelo (recursos, utilidad o costo), pueden cambiar sin que esto afecte a la solución óptima y a la capacidad de calcular el impacto que tienen dichos cambios sobre la misma.

Como tal, constituye una herramienta importante en el análisis de los resultados obtenidos, sobre todo, en dos casos en particular:

1. La sensibilidad de la solución óptima ante los cambios de la disponibilidad de los recursos (lado derecho de las restricciones). Es decir, cuáles serían los límites dentro de los cuáles los resultados del modelo me permiten calcular el impacto que tiene sobre la solución óptima, el aumento o la disminución de la disponibilidad de un recurso.
2. La sensibilidad de la solución óptima ante los cambios en la utilidad unitaria o el costo unitario (coeficientes de las variables de la función objetivo). Es decir, cuáles serían los límites dentro de los cuáles los resultados del modelo me permiten calcular el impacto que tiene sobre la solución óptima, el aumento o la disminución de la utilidad unitaria o el costo unitario asociada a las variables que forman parte de la función objetivo.

Análisis de sensibilidad gráfica: Cambios en el lado derecho (Disponibilidad de recursos)

Una fábrica construye dos productos en dos máquinas. Una unidad del producto 1 requiere 2 horas en la máquina 1, y 1 hora en la máquina 2. Una unidad del producto 2 requiere 1 hora en la máquina 1, y 3 horas en la máquina 2. Los ingresos por unidad de los productos 1 y 2 son de 30 y 20 pesos, respectivamente. El tiempo de procesamiento diario total disponible en cada máquina es de 8 horas.

Si x1 y x2 son las cantidades diarias de productos 1 y 2, respectivamente, el modelo se da como:

Zmax = 30x1 + 20x2

Sujeto a:

2x1 + x2 <= 8 (Máquina 1)

x1 + 3x2 <= 8 (Máquina 2)

x1, x2 >= 0 (No negatividad)

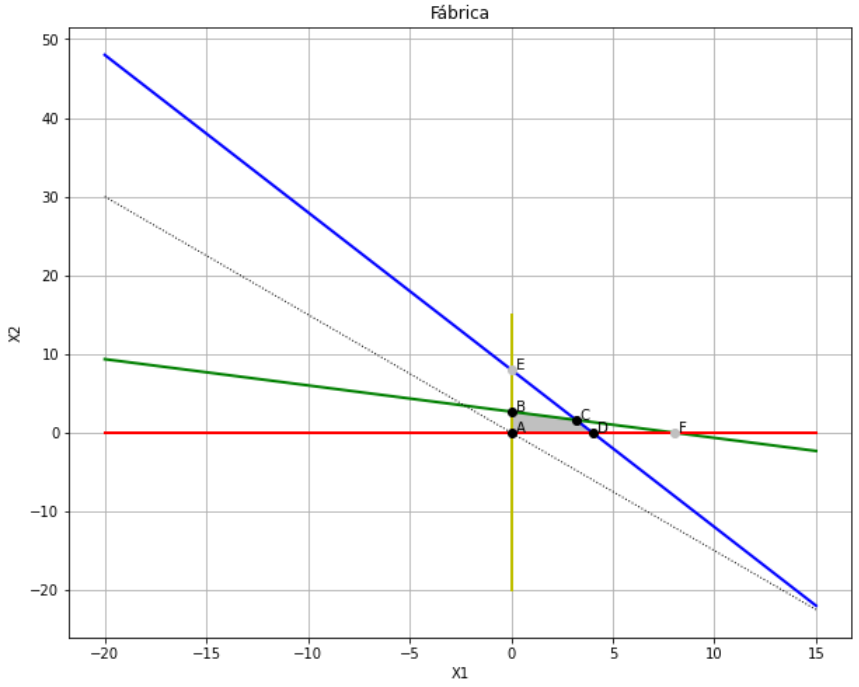
De acuerdo a lo visto en el apunte 13.5:

In [1]:

```
import matplotlib.pyplot as plt
import numpy as np
from shapely.geometry import LineString

plt.figure(figsize=(10,8))
x = np.arange(-20, 20, 5)
y = np.arange(-20, 20, 5)

y1 = 8 - (2 * x)
y2 = (8 - x) / 3
y3 = 0 * x
x1 = 0 * y
z = (-30 * x) / 20

primera_linea = LineString(np.column_stack((x, y1)))
segunda_linea = LineString(np.column_stack((x, y2)))
tercera_linea = LineString(np.column_stack((x, y3)))
cuarta_linea = LineString(np.column_stack((x1, y)))
quinta_linea = LineString(np.column_stack((x, z)))

plt.plot(x, y1, '-', linewidth=2, color='b')
plt.plot(x, y2, '-', linewidth=2, color='g')
plt.plot(x, y3, '-', linewidth=2, color='r')
plt.plot(x1, y, '-', linewidth=2, color='y')
plt.plot(x, z, ':', linewidth=1, color='k')

primera_interseccion = cuarta_linea.intersection(segunda_linea)
segunda_interseccion = primera_linea.intersection(segunda_linea)
tercera_interseccion = primera_linea.intersection(tercera_linea)
cuarta_interseccion = tercera_linea.intersection(cuarta_linea)
quinta_interseccion = cuarta_linea.intersection(primera_linea)
sexta_interseccion = segunda_linea.intersection(tercera_linea)

plt.plot(*primera_interseccion.xy, 'o', color='k')
plt.plot(*segunda_interseccion.xy, 'o', color='k')
plt.plot(*tercera_interseccion.xy, 'o', color='k')
plt.plot(*cuarta_interseccion.xy, 'o', color='k')
plt.plot(*quinta_interseccion.xy, 'o', color='silver')
plt.plot(*sexta_interseccion.xy, 'o', color='silver')

xi1m, yi1m = primera_interseccion.xy
xi2m, yi2m = segunda_interseccion.xy
xi3m, yi3m = tercera_interseccion.xy
xi4m, yi4m = cuarta_interseccion.xy
xi5m, yi5m = quinta_interseccion.xy
xi6m, yi6m = sexta_interseccion.xy

xi1 = np.float64(np.array(xi1m))
xi2 = np.float64(np.array(xi2m))
xi3 = np.float64(np.array(xi3m))
xi4 = np.float64(np.array(xi4m))
xi5 = np.float64(np.array(xi5m))
xi6 = np.float64(np.array(xi6m))
yi1 = np.float64(np.array(yi1m))
yi2 = np.float64(np.array(yi2m))
yi3 = np.float64(np.array(yi3m))
yi4 = np.float64(np.array(yi4m))
yi5 = np.float64(np.array(yi5m))
yi6 = np.float64(np.array(yi6m))

plt.annotate(' A', (xi4, yi4))
plt.annotate(' B', (xi1, yi1))
plt.annotate(' C', (xi2, yi2))
plt.annotate(' D', (xi3, yi3))
plt.annotate(' E', (xi5, yi5))
plt.annotate(' F', (xi6, yi6))

FOi1 = (xi1 * 30) + (yi1 * 20)
FOi2 = (xi2 * 30) + (yi2 * 20)
FOi3 = (xi3 * 30) + (yi3 * 20)
FOi4 = (xi4 * 30) + (yi4 * 20)

ZMAX = max(FOi1, FOi2, FOi3, FOi4)

print('\n SOLUCIÓN ÓPTIMA')
print('Solución óptima: {} '.format(ZMAX))

m = [xi1, xi2, xi3, xi4]
n = [yi1, yi2, yi3, yi4]

plt.fill(m, n, color='silver')

dict1 = {0:FOi1, 1:FOi2, 2:FOi3, 3:FOi4}
posicion = max(dict1, key=dict1.get)

XMAX = m[posicion]
YMAX = n[posicion]
```

```
print('\n VARIABLES DE DECISIÓN')
print('Cantidad de producto X1 a producir: {} unidades'.format(XMAX))
print('Cantidad de producto X2 a producir: {} unidades'.format(YMAX))

plt.grid()
plt.xlabel('X1')
plt.ylabel('X2')
plt.title('Fábrica')

plt.show()
```

```
 SOLUCIÓN ÓPTIMA
Solución óptima: 128.0

 VARIABLES DE DECISIÓN
Cantidad de producto X1 a producir: 3.2 unidades
Cantidad de producto X2 a producir: 1.5999999999999999 unidades
```



¿Qué pasaría con los ingresos totales si la máquina 1 cambia su capacidad?

Recordemos que la restricción base es la siguiente:

2x1 + x2 <= 8 (Máquina 1)

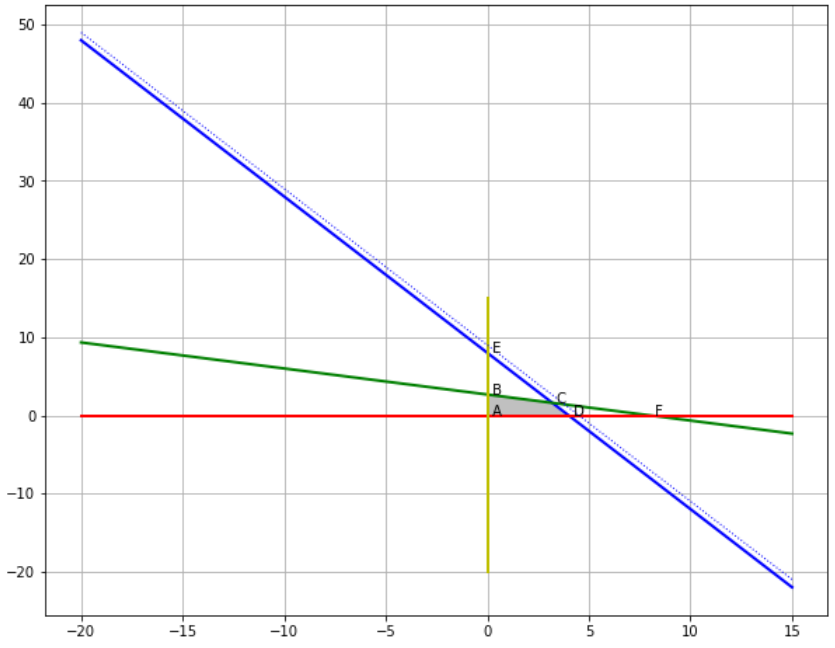
Un incremento en la capacidad (variación), se representaría mediante la siguiente inecuación:

2x1 + x2 <= 8 + 1

Esto quiere decir que, la nueva disponibilidad de la máquina 1 pasará de 8 horas a 9 horas. Y nuestro razonamiento nos indica que esta función, tendrá una representación distinta a la anterior. Graficando:

In [2]:

```
x = np.arange(-20, 20, 5)
y = np.arange(-20, 20, 5)

y1 = 8 - (2 * x) # ecuación con la cual se representa la restricción de la máquina 1
y2 = (8 - x) / 3
y3 = 0 * x
x1 = 0 * y
z = (-30 * x) / 20
y1v = (8 + 1) - (2 * x) # aumentar la capacidad de la máquina 1 (8) a una nueva capacidad (8 + 1 = 9)

primera_linea = LineString(np.column_stack((x, y1)))
segunda_linea = LineString(np.column_stack((x, y2)))
tercera_linea = LineString(np.column_stack((x, y3)))
cuarta_linea = LineString(np.column_stack((x1, y)))
quinta_linea = LineString(np.column_stack((x, z)))
sexta_linea = LineString(np.column_stack((x, y1v)))

plt.figure(figsize=(10,8))
plt.grid()

plt.plot(x, y1, '-', linewidth=2, color='b')
plt.plot(x, y2, '-', linewidth=2, color='g')
plt.plot(x, y3, '-', linewidth=2, color='r')
plt.plot(x1, y, '-', linewidth=2, color='y')
# plt.plot(x, z, ':', linewidth=1, color='k')  #Podemos ocultar esta línea (FO)
plt.plot(x, y1v, ':', linewidth=1, color='b')

plt.annotate(' A', (xi4, yi4))
plt.annotate(' B', (xi1, yi1))
plt.annotate(' C', (xi2, yi2))
plt.annotate(' D', (xi3, yi3))
plt.annotate(' E', (xi5, yi5))
plt.annotate(' F', (xi6, yi6))

plt.fill(m, n, color='silver')
plt.show()
```



La variación en la disponibilidad del recurso 1 (máquina 1), se representa con una línea recta (línea azul punteada) que conserva la misma pendiente de la restricción 1 (línea azul continua), que se mueve paralela a ella, y que en consecuencia, genera una nueva intersección solución (intersección con la restricción 2 – línea verde).

En este nuevo vértice solución se encuentra un nuevo punto óptimo, es decir, una coordenada diferente en la cual al evaluar la función objetivo, tendremos la respuesta al interrogante planteado:

¿Qué pasaría con los ingresos totales si la máquina 1 cambia su capacidad?

En consecuencia, debemos:

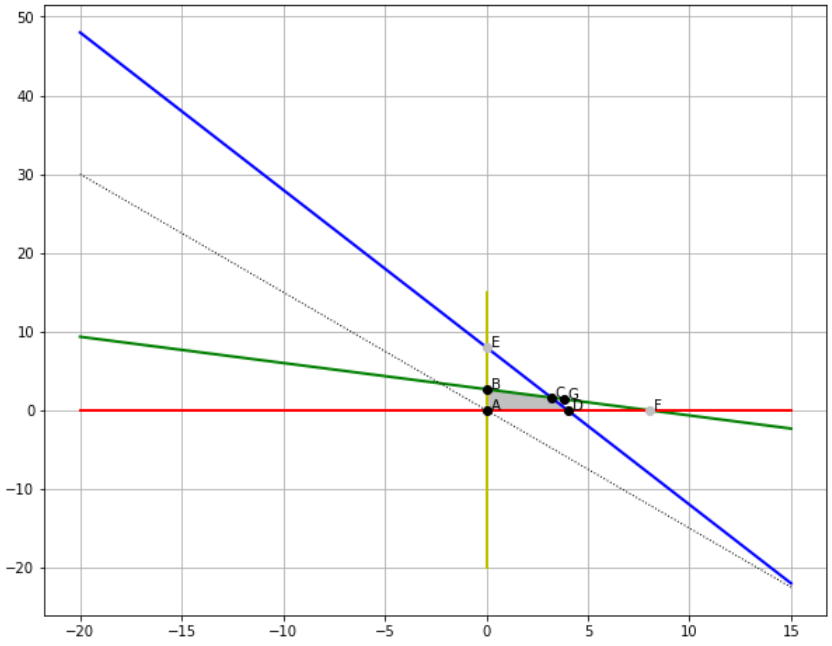
- Establecer gráficamente el nuevo vértice
- Nombrarlo gráficamente (Vértice G)
- Determinar las coordenadas de este vértice (G)
- Evaluar la función objetivo en este nuevo vértice (G)

In [3]:

```
import matplotlib.pyplot as plt
import numpy as np
from shapely.geometry import LineString

plt.figure(figsize=(10,8))
x = np.arange(-20, 20, 5)
y = np.arange(-20, 20, 5)
y1 = 8 - (2 * x)
y2 = (8 - x) / 3
y3 = 0 * x
x1 = 0 * y
z = (-30 * x) / 20

primera_linea = LineString(np.column_stack((x, y1)))
segunda_linea = LineString(np.column_stack((x, y2)))
tercera_linea = LineString(np.column_stack((x, y3)))
cuarta_linea = LineString(np.column_stack((x1, y)))
quinta_linea = LineString(np.column_stack((x, z)))

plt.figure(figsize=(10,8))
plt.plot(x, y1, '-', linewidth=2, color='b')
plt.plot(x, y2, '-', linewidth=2, color='g')
plt.plot(x, y3, '-', linewidth=2, color='r')
plt.plot(x1, y, '-', linewidth=2, color='y')
plt.plot(x, z, ':', linewidth=1, color='k')

primera_interseccion = cuarta_linea.intersection(segunda_linea)
segunda_interseccion = primera_linea.intersection(segunda_linea)
tercera_interseccion = primera_linea.intersection(tercera_linea)
cuarta_interseccion = tercera_linea.intersection(cuarta_linea)
quinta_interseccion = cuarta_linea.intersection(primera_linea)
sexta_interseccion = segunda_linea.intersection(tercera_linea)
septima_interseccion = sexta_linea.intersection(segunda_linea) #Nuevo vértice

plt.plot(*primera_interseccion.xy, 'o', color='k')
plt.plot(*segunda_interseccion.xy, 'o', color='k')
plt.plot(*tercera_interseccion.xy, 'o', color='k')
plt.plot(*cuarta_interseccion.xy, 'o', color='k')
plt.plot(*quinta_interseccion.xy, 'o', color='silver')
plt.plot(*sexta_interseccion.xy, 'o', color='silver')
plt.plot(*septima_interseccion.xy, 'o', color='k') #Graficar nuevo vértice

xi1m, yi1m = primera_interseccion.xy
xi2m, yi2m = segunda_interseccion.xy
xi3m, yi3m = tercera_interseccion.xy
xi4m, yi4m = cuarta_interseccion.xy
xi5m, yi5m = quinta_interseccion.xy
xi6m, yi6m = sexta_interseccion.xy
xi7m, yi7m = septima_interseccion.xy #Coordenadas del nuevo vértice

xi1 = np.float64(np.array(xi1m))
xi2 = np.float64(np.array(xi2m))
xi3 = np.float64(np.array(xi3m))
xi4 = np.float64(np.array(xi4m))
xi5 = np.float64(np.array(xi5m))
xi6 = np.float64(np.array(xi6m))
xi7 = np.float64(np.array(xi7m)) #Nueva coordenada en x

yi1 = np.float64(np.array(yi1m))
yi2 = np.float64(np.array(yi2m))
yi3 = np.float64(np.array(yi3m))
yi4 = np.float64(np.array(yi4m))
yi5 = np.float64(np.array(yi5m))
yi6 = np.float64(np.array(yi6m))
yi7 = np.float64(np.array(yi7m)) #Nueva coordenada en y

plt.annotate(' A', (xi4, yi4))
plt.annotate(' B', (xi1, yi1))
plt.annotate(' C', (xi2, yi2))
plt.annotate(' D', (xi3, yi3))
plt.annotate(' E', (xi5, yi5))
plt.annotate(' F', (xi6, yi6))
plt.annotate(' G', (xi7, yi7)) #Nombrar el nuevo vértice como G

FOi1 = (xi1 * 30) + (yi1 * 20)
FOi2 = (xi2 * 30) + (yi2 * 20)
FOi3 = (xi3 * 30) + (yi3 * 20)
FOi4 = (xi4 * 30) + (yi4 * 20)
FOi5 = (xi5 * 30) + (yi5 * 20)
FOi6 = (xi6 * 30) + (yi6 * 20)
FOi7 = (xi7 * 30) + (yi7 * 20) #Calcular la FO en el nuevo vértice G

print('\n SOLUCIÓN ÓPTIMA')
print('Solución óptima: {} '.format(ZMAX))
print('Función objetivo en punto G: {} '.format(FOi7)) #Mostrar la FO en el punto G

plt.grid()
```

```
plt.fill(m, n, color='silver')
plt.show()
```

```
 SOLUCIÓN ÓPTIMA
Solución óptima: 128.0
Función objetivo en punto G: 142.0

<Figure size 720x576 with 0 Axes>
```



La figura ilustra el cambio de la solución óptima cuando se cambia la capacidad de la máquina 1. Si la capacidad diaria se incrementa de 8 a 9 horas, el nuevo óptimo se moverá al punto G y el valor de la función objetivo evaluada en el punto G (z en G) equivale a 142.0. Es decir, un valor superior a 128.0 (z en C).

Con los datos obtenidos podemos establecer en este caso la tasa de cambio en la función objetivo a consecuencia del cambio de la capacidad de la máquina 1 de 8 a 9 horas. Esta tasa de cambio se denomina de diversas formas: valor unitario de un recurso, precio dual (múltiples solucionadores), shadow price (precio sombra). La fórmula del cáculo de la tasa sería:

$$\frac{\text{nuevo vértice óptimo} - \text{vértice óptimo base}}{\text{nueva capacidad del recurso} - \text{capacidad del recurso base}}$$

$$\frac{142.0 - 128.0}{9 \text{ horas} - 8 \text{ horas}}$$

$$\text{Tasa de cambio} = 14 \text{ pesos}$$

Esto indica que: Un incremento unitario en la capacidad de la máquina 1, aumentará el ingreso en 14 pesos. O lo que es igual una reducción unitaria en la capacidad de la máquina 1, reducirá el ingreso en 14 pesos.

Será cuestión de crear una nueva variable que represente el precio dual y la operación correspondiente:

In [4]:
```
Dual1 = (FOi7 - ZMAX) #Calculamos el precio dual para la restricción 1

print('\n ANÁLISIS DE SENSIBILIDAD')
print('Precio dual de la máquina 1: ${}/h'.format(Dual1)) #Imprimimos el precio dual 1
```

```
 ANÁLISIS DE SENSIBILIDAD
Precio dual de la máquina 1: $14.0/h
```

Recordemos que la función objetivo del punto G (z en G) se representa por la variable FOi7 (función objetivo en la intersección 7). La operación efectúa la diferencia entre este valor y la función objetivo en el vértice óptimo base (ZMAX).

Los cambios en el lado derecho de las restricciones – capacidad de los recursos -, precisa de unos límites dentro de los cuáles los resultados del modelo permiten calcular el impacto que tiene sobre la solución óptima.

Es decir los límites dentro de los cuales el precio dual para la máquina 1 es $14/h. Estamos en condiciones de calcular el impacto de una variación en la máquina 1 en los ingresos totales, pero, ¿Hasta cuándo esa proyección es válida? Y sí, existen unos límites dentro de los cuales nuestra proyección, tasa o precio dual es válido, por fuera de ellos, sería necesario recalcular.

Revisemos nuevamente el último gráfico. La línea que representa los cambios en la capacidad de la máquina 1 (recurso) es paralela sobre el segmento de la línea que representa la capacidad de la máquina 2 (línea verde, o desde el punto B hasta el punto F). Esa es la clave para determinar los límites de validez del precio dual, mejor conocidos como **intervalos de factibilidad**.

La ecuación que representa la capacidad de la máquina 1 está dada por:

2x1 + x2 <= 8 horas (máquina 1)

En consecuencia, si eliminamos el valor de la capacidad del recurso (8 horas), se obtiene la ecuación de la utilización o el uso del recurso:

2x1 + x2 = Utilización o uso de la máquina 1

Teniendo la ecuación de la restricción 1 (máquina 1), podemos evaluar cuál sería su utilización en los vértices B y F. Así se encontrarían los límites de validez para esta restricción (intervalos de factibilidad para nuestro precio dual = 14 pesos/h).

O sea que encontraríamos la capacidad mínima y máxima de la máquina 1, para que una variación unitaria en la disponibilidad de la misma (máquina 1), represente una variación de 14 pesos en la función objetivo. Por fuera de ese intervalo, no podemos asegurar de ninguna manera que el precio dual es de 14 pesos/h y sería necesario recalcular.

Evaluaremos la ecuación de la restricción en los puntos B y F. Para ello es preciso conocer las coordenadas de cada uno de estos puntos:

In [5]:
```
IFmin1 = (2 * xi1) + yi1
IFmax1 = (2 * xi6) + yi6

print('Límite mínimo de factibilidad (PD máquina 1): {} h'.format(IFmin1))
print('Límite máximo de factibilidad (PD máquina 1): {} h'.format(IFmax1))
```

```
Límite mínimo de factibilidad (PD máquina 1): 2.6666666666666665 h
Límite máximo de factibilidad (PD máquina 1): 16.0 h
```

El punto B corresponde a la segunda intersección, por ende sus coordenadas están definidas por las variables xi1 y yi1. Así entonces, evaluamos la capacidad de la máquina 1 en estas coordenadas y tenemos el límite mínimo de factibilidad para el precio dual de la máquina 1 (IFmin1).

El punto F corresponde a la sexta intersección, por ende sus coordenadas están definidas por las variables xi6 y yi6. Así entonces, evaluamos la capacidad de la máquina 1 en estas coordenadas y tenemos el límite mínimo de factibilidad para el precio dual de la máquina 1 (IFmax1).

El punto B tiene la siguientes coordenadas (x = 0, y = 2,67)

Evaluando estas coordenadas en la ecuación de la máquina 1, tenemos:

2(0)+ (2,67) = Límite mínimo de factibilidad (máquina 1)

2(0)+ (2,67) = 2,67 horas

El punto F tiene la siguientes coordenadas (x = 8, y = 0)

Evaluando estas coordenadas en la ecuación de la máquina 1, tenemos:

2(8)+ (0) = Límite máximo de factibilidad (máquina 1)

2(8)+ (0) = 16 horas

La conclusión es que el precio dual de $14/h permanece válido en el intervalo:

2,67 h <= Capacidad de la máquina 1 <= 16 h

Los cambios que se encuentren fuera de esos intervalos de factibilidad producen un precio dual diferente, y por lo tanto, precisan nuevos cálculos.

Incluiremos las líneas que permitan llegar al precio dual y a los intervalos de factibilidad de la máquina 2.

Programa completo

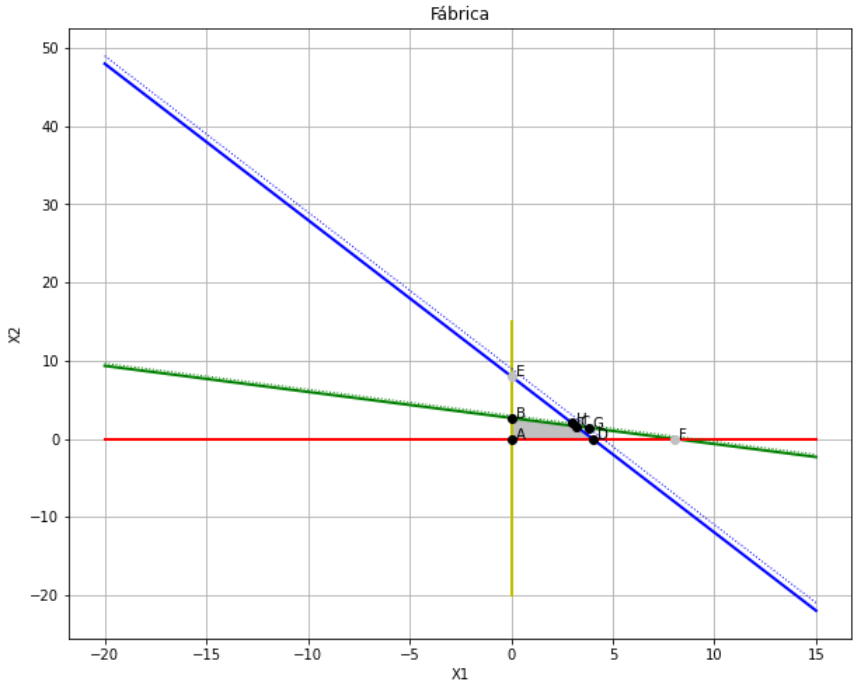
In [6]:

```
import matplotlib.pyplot as plt
import numpy as np
from shapely.geometry import LineString

#Ecuaciones e intervalos (Para tabular)
x = np.arange(-20, 20, 5)
y = np.arange(-20, 20, 5)
y1 = 8 - (2 * x)
y2 = (8 - x) / 3
y3 = 0 * x
x1 = 0 * y
z = (-30 * x) / 20
y1v = (8 + 1) - (2 * x) #Variación en la máquina 1 (8 + 1)
y2v = ((8 + 1) - x) / 3 #Variación en la máquina 2 (8 + 1)

#Identificadores para las líneas
primera_linea = LineString(np.column_stack((x, y1)))
segunda_linea = LineString(np.column_stack((x, y2)))
tercera_linea = LineString(np.column_stack((x, y3)))
cuarta_linea = LineString(np.column_stack((x1, y)))
quinta_linea = LineString(np.column_stack((x, z)))
sexta_linea = LineString(np.column_stack((x, y1v))) #Nueva línea Máquina 1 (M1)
septima_linea = LineString(np.column_stack((x, y2v))) #Nueva línea Máquina 2 (M2)

#Graficando las líneas

plt.figure(figsize=(10,8))
plt.plot(x, y1, '-', linewidth=2, color='b')
plt.plot(x, y2, '-', linewidth=2, color='g')
plt.plot(x, y3, '-', linewidth=2, color='r')
plt.plot(x1, y, '-', linewidth=2, color='y')
#plt.plot(x, z, ':', linewidth=1, color='k')
plt.plot(x, y1v, ':', linewidth=1, color='b') #Nueva línea M1
plt.plot(x, y2v, ':', linewidth=1, color='g') #Nueva línea M2


#Generando las intersecciones (vértices)
primera_interseccion = cuarta_linea.intersection(segunda_linea)
segunda_interseccion = primera_linea.intersection(segunda_linea)
tercera_interseccion = primera_linea.intersection(tercera_linea)
cuarta_interseccion = tercera_linea.intersection(cuarta_linea)
quinta_interseccion = cuarta_linea.intersection(primera_linea)
sexta_interseccion = segunda_linea.intersection(tercera_linea)
septima_interseccion = sexta_linea.intersection(segunda_linea) #Nuevo vértice M1
octava_interseccion = septima_linea.intersection(primera_linea) #Nuevo vértice M2

#Graficando los vértices
plt.plot(*primera_interseccion.xy, 'o', color='k')
plt.plot(*segunda_interseccion.xy, 'o', color='k')
plt.plot(*tercera_interseccion.xy, 'o', color='k')
plt.plot(*cuarta_interseccion.xy, 'o', color='k')
plt.plot(*quinta_interseccion.xy, 'o', color='silver')
plt.plot(*sexta_interseccion.xy, 'o', color='silver')
plt.plot(*septima_interseccion.xy, 'o', color='k') #Graficar nuevo vértice M1
plt.plot(*octava_interseccion.xy, 'o', color='k') #Graficar nuevo vértice M2

#Identificando los valores de las coordenadas x y y de cada vértice
xi1m, yi1m = primera_interseccion.xy
xi2m, yi2m = segunda_interseccion.xy
xi3m, yi3m = tercera_interseccion.xy
xi4m, yi4m = cuarta_interseccion.xy
xi5m, yi5m = quinta_interseccion.xy
xi6m, yi6m = sexta_interseccion.xy
xi7m, yi7m = septima_interseccion.xy #Coordenadas del nuevo vértice M1
xi8m, yi8m = octava_interseccion.xy #Coordenadas del nuevo vértice M2

#Cambiamos el formato de matriz a float
xi1 = np.float64(np.array(xi1m))
xi2 = np.float64(np.array(xi2m))
xi3 = np.float64(np.array(xi3m))
xi4 = np.float64(np.array(xi4m))
xi5 = np.float64(np.array(xi5m))
xi6 = np.float64(np.array(xi6m))
xi7 = np.float64(np.array(xi7m)) #Nueva coordenada en x (Máquina 1)
xi8 = np.float64(np.array(xi8m)) #Nueva coordenada en x (Máquina 2)
yi1 = np.float64(np.array(yi1m))
yi2 = np.float64(np.array(yi2m))
yi3 = np.float64(np.array(yi3m))
yi4 = np.float64(np.array(yi4m))
yi5 = np.float64(np.array(yi5m))
yi6 = np.float64(np.array(yi6m))
yi7 = np.float64(np.array(yi7m)) #Nueva coordenada en y (Máquina 1)
yi8 = np.float64(np.array(yi8m)) #Nueva coordenada en y (Máquina 2)

#literales de las intersecciones (nombres en el gráfico)
plt.annotate(' A', (xi4, yi4))
plt.annotate(' B', (xi1, yi1))
```

```
plt.annotate(' C', (xi2, yi2))
plt.annotate(' D', (xi3, yi3))
plt.annotate(' E', (xi5, yi5))
plt.annotate(' F', (xi6, yi6))
plt.annotate(' G', (xi7, yi7)) #Nombrar el nuevo vértice como G
plt.annotate(' H', (xi8, yi8)) #Nombrar el nuevo vértice como H

#Evaluando la función objetivo en cada vértice
FOi1 = (xi1 * 30) + (yi1 * 20)
FOi2 = (xi2 * 30) + (yi2 * 20)
FOi3 = (xi3 * 30) + (yi3 * 20)
FOi4 = (xi4 * 30) + (yi4 * 20)
FOi7 = (xi7 * 30) + (yi7 * 20) #Calcular la FO en el nuevo vértice G
FOi8 = (xi8 * 30) + (yi8 * 20) #Calcular la FO en el nuevo vértice H

#Calculando el mejor resultado (Maximizar)
ZMAX = max(FOi1, FOi2, FOi3, FOi4)

#Imprimiendo la solución óptima en la consola
print('\n SOLUCIÓN ÓPTIMA')
print('Solución óptima: {} '.format(ZMAX))
print('Función objetivo en punto G: {} '.format(FOi7))
print('Función objetivo en punto H: {} '.format(FOi8))

#Ordenando las coordenadas de los vértices (Las coordenadas x en m y las coordenadas y en n)
m = [xi1, xi2, xi3, xi4]
n = [yi1, yi2, yi3, yi4]

#Graficando el polígono solución a partir de las coordenadas de los vértices (importante el orden según las maneci
plt.fill(m, n, color='silver')

#Identificando el índice del vértice de la mejor solución
dict1 = {0:FOi1, 1:FOi2, 2:FOi3, 3:FOi4}
posicion = max(dict1, key=dict1.get)

#Obteniendo las coordenadas del vértice de la mejor solución de acuerdo al índice del paso anterior
XMAX = m[posicion]
YMAX = n[posicion]

#Imprimiendo las coordenadas del vértice de la mejor solución (variables de decisión)
print('\n VARIABLES DE DECISIÓN')
print('Cantidad de producto X1 a producir: {} unidades'.format(XMAX))
print('Cantidad de producto X2 a producir: {} unidades'.format(YMAX))

#Precio dual (Restricción 1)
Dual1 = (FOi7 - ZMAX) #Calculamos el precio dual para la restricción 1
Dual2 = (FOi8 - ZMAX) #Calculamos el precio dual para la restricción 1

#Imprimir los precios duales
print('\n ANÁLISIS DE SENSIBILIDAD')
print('Precio dual de la máquina 1: {} $/h'.format(Dual1)) #Imprimimos el precio dual 1
print('Precio dual de la máquina 2: {} $/h'.format(Dual2)) #Imprimimos el precio dual 2

#Intervalos de factibilidad (Máquina 1)
IFmin1 = (2 * xi1) + yi1
IFmax1 = (2 * xi6) + yi6

#Intervalos de factibilidad (Máquina 2)
IFmin2 = xi3 + (3 * yi3)
IFmax2 = xi5 + (3 * yi5)

#Imprimir los intervalos de factibilidad
print('\n INTERVALOS DE FACTIBILIDAD')
print('Límite mínimo de factibilidad (PD máquina 1): {} h'.format(IFmin1))
print('Límite máximo de factibilidad (PD máquina 1): {} h'.format(IFmax1))
print('Límite mínimo de factibilidad (PD máquina 2): {} h'.format(IFmin2))
print('Límite máximo de factibilidad (PD máquina 2): {} h'.format(IFmax2))


#Configuraciones adicionales del gráfico
plt.grid()
plt.xlabel('X1')
plt.ylabel('X2')
plt.title('Fábrica')

plt.show()
```

```
 SOLUCIÓN ÓPTIMA
Solución óptima: 128.0
Función objetivo en punto G: 142.0
Función objetivo en punto H: 130.0

 VARIABLES DE DECISIÓN
Cantidad de producto X1 a producir: 3.2 unidades
Cantidad de producto X2 a producir: 1.5999999999999999 unidades
```

```
 ANÁLISIS DE SENSIBILIDAD
Precio dual de la máquina 1: 14.0 $/h
Precio dual de la máquina 2: 2.0 $/h

 INTERVALOS DE FACTIBILIDAD
Límite mínimo de factibilidad (PD máquina 1): 2.6666666666666665 h
Límite máximo de factibilidad (PD máquina 1): 16.0 h
Límite mínimo de factibilidad (PD máquina 2): 4.0 h
Límite máximo de factibilidad (PD máquina 2): 24.0 h
```



La conclusión es que el precio dual de $2/h permanece válido en el intervalo:

4 h <= Capacidad de la máquina 2 <= 24 h

Pregunta 1: Si la fábrica puede incrementar la capacidad de ambas máquinas,¿Cuál máquina tendrá la prioridad?

Respuesta: De acuerdo a los precios duales para las máquinas 1 y 2, cada hora adicional de la máquina 1 incrementa el ingreso total en $14; mientrastanto, cadahoraadicionaldelamáquina2incrementaelingresototalen2$. Por lo tanto, la máquina 1 debe tener la prioridad.

Pregunta 2: Se sugiere incrementar las capacidades de las máquinas 1 y 2 al costo adicional de $10/h para cada máquina. ¿Es esto aconsejable?

Respuesta: De acuerdo a los precios duales para las máquinas 1 y 2. Los ingresos netos adicionales por hora serían de la siguiente manera:

Máquina 1: $14/h(Preciodualmáquina1) – 10$/h (Costo para aumentar capacidad) = 4 $/h (Ingreso neto)

Máquina 2: $2/h(Preciodualmáquina2) – 10$/h (Costo para aumentar capacidad) = – 8 $/h (Ingreso neto)

Por consiguiente, solo la máquina 1 debe considerarse para el incremento de capacidad.

Pregunta 3: Si la capacidad de la máquina 1 se incrementa de 8 a 13 horas, ¿Cómo impactará este incremento al ingreso óptimo?

Respuesta: Lo primero que debe evaluarse es que la nueva capacidad de la máquina 1 se encuentre dentro del intervalo de factibilidad (2,67 h – 16 h). Como se encuentra en dicho intervalo (13 h), el siguiente paso consiste en calcular el cambio en la disponibilidad:

Cambio en la capacidad = 13 (Nueva capacidad) – 8 (Capacidad inicial)

Cambio en la capacidad = 5 horas

En consecuencia multiplicamos dicho cambio por la tasa de cambio del ingreso (precio dual):

Cambio en el ingreso = Precio dual * Cambio en la capacidad

Cambio en el ingreso = 14 $/h * (5 horas)

Cambio en el ingreso = 70 $

Nuevo ingreso = Ingreso base (Solución óptima) + Cambio en el ingreso

Nuevo ingreso = 128 $+70$

Nuevo ingreso = 198 $

En el caso de que la nueva capacidad de la máquina se encuentre por fuera del intervalo de factibilidad para dicho recurso, no se dispondría de información suficiente para llegar a una conclusión válida.

In [ ]: