Universidad Nacional del Altiplano
Facultad de Ingeniería Estadística e Informática
Docente: Fred Torres Cruz
Alumno: Clyde Neil Paricahua Pari

Trabajo Encargado - Restricciones y Sistemas de Ecuaciones

Ejercicio 1: Restricciones de Desarrollo de Software

Enunciado del Problema

Un desarrollador tiene 15 horas semanales para dedicar al desarrollo de software de front-end (x) y back-end (y). Además:

- Debe dedicar al menos 5 horas al desarrollo de front-end para cumplir con los entregables del cliente.
- El tiempo total no puede exceder 15 horas por restricciones de tiempo del sprint.

Se solicita formular las restricciones, representarlas gráficamente e identificar las combinaciones posibles de tiempo a invertir en cada actividad.

Formulación Matemática

Variables de decisión:

$$x = \text{horas dedicadas al desarrollo front-end} \tag{1}$$
$$y = \text{horas dedicadas al desarrollo back-end} \tag{2}$$

Restricciones:

$$x \geq 5 \quad \text{(mínimo 5 horas front-end)} \tag{3}$$
$$x + y \leq 15 \quad \text{(tiempo total máximo)} \tag{4}$$
$$x \geq 0, \quad y \geq 0 \quad \text{(restricciones de no negatividad)} \tag{5}$$

Ecuaciones de frontera:
Para la representación gráfica, las ecuaciones de frontera son:

$$\text{Ecuación 1: } x = 5 \tag{6}$$
$$\text{Ecuación 2: } y = 15 - x \tag{7}$$

Implementación del Software

```python
class GraficadoraTexto:
    def __init__(self, xmin=-1, xmax=20, ymin=-1, ymax=15):
        self.xmin = xmin
        self.xmax = xmax
        self.ymin = ymin
        self.ymax = ymax
        self.funciones = []

    def preparar_expresion(self, expr: str) -> str:
        expr = expr.replace(" ", "")
        expr = expr.replace("^", "**")
        if expr.startswith("x"):
            expr = "1*" + expr
        expr = expr.replace("-x", "-1*x")
```

```
        expr = expr.replace("+x", "+1*x")
        expr = expr.replace("x", "*x")
        expr = expr.replace("**x", "*x")
        return expr

    def agregar_funcion(self, expresion: str, simbolo: str):
        expr_preparada = self.preparar_expresion(expresion)
        self.funciones.append((expr_preparada, simbolo))

    def graficar(self):
        print("\nGr fico en Plano Cartesiano (ASCII)\n")
        for y in range(self.ymax, self.ymin - 1, -1):
            linea = ""
            for x in range(self.xmin, self.xmax + 1):
                simbolo = " "
                intersecciones = []
                for expr, simb in self.funciones:
                    try:
                        y_eval = eval(expr, {"x": x, "y": y})
                        if abs(y - y_eval) < 0.5:
                            intersecciones.append(simb)
                    except:
                        pass
                if len(intersecciones) > 1:
                    simbolo = "#"
                elif len(intersecciones) == 1:
                    simbolo = intersecciones[0]
                elif x == 0 and y == 0:
                    simbolo = "+"
                elif x == 0:
                    simbolo = "|"
                elif y == 0:
                    simbolo = "-"
                linea += simbolo
            print(linea)



if __name__ == "__main__":
    print("Ecuaci n 1: x = 5 (restricci n m nima front-end)")
    print("Ecuaci n 2: y = 15 - x (tiempo total)")
    print()

    graf = GraficadoraTexto()
    graf.agregar_funcion("(x==5)*y", "*")
    graf.graficar()
```

Resultado de la Ejecución

Al ejecutar el programa, se obtiene la siguiente representación gráfica:

Interpretación de la Solución Gráfica

Análisis de las Restricciones

- **Línea vertical (*):** Representa la restricción $x = 5$, que establece el mínimo de 5 horas que debe dedicarse al desarrollo front-end.
- **Línea diagonal (@):** Representa la ecuación $y = 15 - x$, que corresponde a la frontera de la restricción $x + y \leq 15$ (tiempo total máximo).

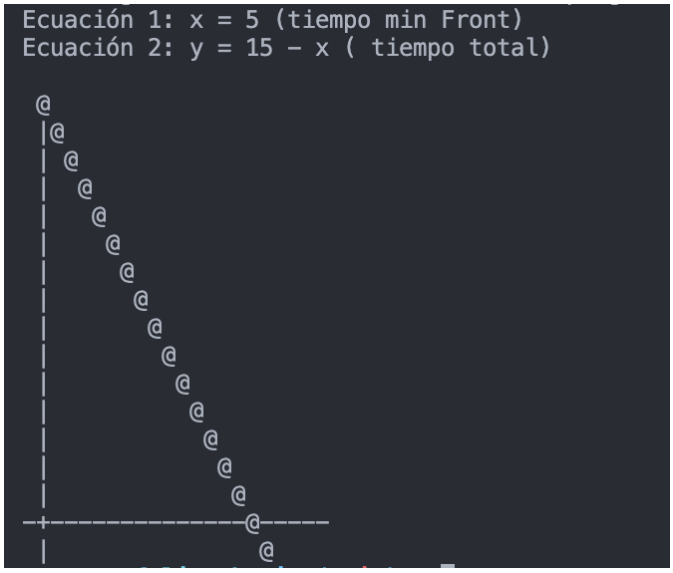

Figura 1: Compilacion Ejercicio 1

- **Intersección (#):** Indica el punto donde se cruzan ambas restricciones, ubicado en el punto $(5, 10)$.

Región Factible

La región factible está definida por el área que satisface simultáneamente todas las restricciones:

- A la derecha de la línea $x = 5$ (cumple $x \geq 5$)
- Debajo de la línea $y = 15 - x$ (cumple $x + y \leq 15$)
- En el primer cuadrante (cumple $x \geq 0, y \geq 0$)

Vértices de la Región Factible

Los vértices que delimitan la región factible son:

$$A = (5, 0) \quad \text{5 horas front-end, 0 horas back-end} \tag{8}$$
$$B = (15, 0) \quad \text{15 horas front-end, 0 horas back-end} \tag{9}$$
$$C = (5, 10) \quad \text{5 horas front-end, 10 horas back-end} \tag{10}$$

Combinaciones Posibles

Cualquier punto (x, y) dentro de la región factible representa una asignación válida de tiempo. Algunos ejemplos de combinaciones factibles son:

- $(8, 7)$: 8 horas front-end + 7 horas back-end = 15 horas totales
- $(10, 3)$: 10 horas front-end + 3 horas back-end = 13 horas totales
- $(5, 5)$: 5 horas front-end + 5 horas back-end = 10 horas totales
- $(12, 2)$: 12 horas front-end + 2 horas back-end = 14 horas totales

Ejercicio 2: Restricciones de Presupuesto para Servidores en la Nube

Enunciado del Problema

Un ingeniero de datos administra dos tipos de servidores en la nube: Servidores A y Servidores B.

- El costo por hora de Servidor A es S/3. - El costo por hora de Servidor B es S/5. - El presupuesto máximo semanal para mantener los servidores es S/20.

Se solicita determinar cuántas horas puede mantener activos cada tipo de servidor, formular el sistema de ecuaciones y representarlo gráficamente.

Formulación Matemática

Variables de decisión:

$$x = \text{horas de Servidor A} \tag{11}$$

$$y = \text{horas de Servidor B} \tag{12}$$

Restricción de presupuesto:

$$3x + 5y \leq 20 \tag{13}$$

Restricciones de no negatividad:

$$x \geq 0, \quad y \geq 0 \tag{14}$$

Ecuación de frontera para graficar:

$$y = 4 - 0{,}6x \tag{15}$$

Implementación del Software

```python
class GraficadoraTexto:
    def __init__(self, xmin=0, xmax=10, ymin=0, ymax=5):
        self.xmin = xmin
        self.xmax = xmax
        self.ymin = ymin
        self.ymax = ymax
        self.funciones = []

    def preparar_expresion(self, expr: str) -> str:
        expr = expr.replace(" ", "")
        expr = expr.replace("^", "**")
        if expr.startswith("x"):
            expr = "1*" + expr
        expr = expr.replace("-x", "-1*x")
        expr = expr.replace("+x", "+1*x")
        expr = expr.replace("x", "*x")
        expr = expr.replace("**x", "*x")
        return expr

    def agregar_funcion(self, expresion: str, simbolo: str):
        expr_preparada = self.preparar_expresion(expresion)
        self.funciones.append((expr_preparada, simbolo))

    def graficar(self, sombrear=False):
        for y in range(self.ymax, self.ymin - 1, -1):
            linea = ""
            for x in range(self.xmin, self.xmax + 1):
                simbolo = " "
                intersecciones = []

```

Figura 2: Ejecucion de Ejercicio 02:

```
                if sombrear:
                    if 3*x + 5*y <= 20 and x >= 0 and y >= 0:
                        simbolo = "."

                for expr, simb in self.funciones:
                    try:
                        y_eval = eval(expr, {"x": x})
                        if abs(y - y_eval) < 0.5:
                            intersecciones.append(simb)
                    except:
                        pass

                if len(intersecciones) > 1:
                    simbolo = "#"
                elif len(intersecciones) == 1:
                    simbolo = intersecciones[0]
                elif simbolo == " ":
                    if x == 0 and y == 0:
                        simbolo = "+"
                    elif x == 0:
                        simbolo = "|"
                    elif y == 0:
                        simbolo = "-"

                linea += simbolo
            print(linea)


if __name__ == "__main__":
    graf = GraficadoraTexto(xmin=0, xmax=10, ymin=0, ymax=5)

    f1 = "4 - 0.6*x"

    graf.agregar_funcion(f1, "*")
    graf.graficar(sombrear=True)
```

Resultado de la Ejecución

Al ejecutar el programa, se obtiene una representación ASCII del plano cartesiano donde:
- **Símbolo *:** Representa la frontera de presupuesto $y = 4 - 0{,}6x$. - **Símbolo .:** Representa la región factible donde se cumplen $3x + 5y \leq 20$ y $x, y \geq 0$. - **Ejes:** Se representan con | y − y el origen con +.

Interpretación de la Solución Gráfica

- El punto donde la recta corta el eje Y: $x = 0 \implies y = 4$ horas.
- El punto donde la recta corta el eje X: $y = 0 \implies x \approx 6{,}67$ horas.
- Cualquier punto dentro de la región sombreada representa una combinación de horas factible para los servidores A y B que no excede el presupuesto.

Ejercicio 3: Organización del Tiempo del Administrador de Proyectos

Enunciado del Problema

Un administrador de proyectos tecnológicos organiza su tiempo entre:

- Reuniones con stakeholders (x)
- Trabajo en la documentación técnica (y)

Condiciones:

- Reuniones: al menos 4 horas semanales ($x \geq 4$)
- Documentación: al menos 6 horas semanales ($y \geq 6$)
- Tiempo total disponible: 12 horas ($x + y \leq 12$)

Se solicita determinar la región factible y analizar las combinaciones posibles de tiempo.

Formulación Matemática

Variables de decisión:

$$x = \text{horas semanales en reuniones} \tag{16}$$
$$y = \text{horas semanales en documentación} \tag{17}$$

Restricciones:

$$x \geq 4 \tag{18}$$
$$y \geq 6 \tag{19}$$
$$x + y \leq 12 \tag{20}$$
$$x \geq 0, \quad y \geq 0 \tag{21}$$

Ecuaciones de frontera:

$$\text{Línea vertical: } x = 4 \tag{22}$$
$$\text{Línea horizontal: } y = 6 \tag{23}$$
$$\text{Línea diagonal: } y = 12 - x \tag{24}$$

Implementación del Software

```
class GraficadoraTexto:
    def __init__(self, xmin=0, xmax=12, ymin=0, ymax=12):
        self.xmin = xmin
        self.xmax = xmax
        self.ymin = ymin
        self.ymax = ymax
        self.funciones = []

    def preparar_expresion(self, expr: str) -> str:
```

```
        expr = expr.replace(" ", "")
        expr = expr.replace("^", "**")
        if expr.startswith("x"):
            expr = "1*" + expr
        expr = expr.replace("-x", "-1*x")
        expr = expr.replace("+x", "+1*x")
        expr = expr.replace("x", "*x")
        expr = expr.replace("**x", "*x")
        return expr

    def agregar_funcion(self, expresion: str, simbolo: str):
        expr_preparada = self.preparar_expresion(expr)
        self.funciones.append((expr_preparada, simbolo))

        for y in range(self.ymax, self.ymin - 1, -1):
            linea = ""
            for x in range(self.xmin, self.xmax + 1):
                simbolo = " "
                intersecciones = []

                if sombrear:
                    if x >= 4 and y >= 6 and x + y <= 12:
                        simbolo = "."

                for expr, simb in self.funciones:
                    try:
                        y_eval = eval(expr, {"x": x})
                        if abs(y - y_eval) < 0.5:
                            intersecciones.append(simb)
                    except:
                        pass

                if len(intersecciones) > 1:
                    simbolo = "#"
                elif len(intersecciones) == 1:
                    simbolo = intersecciones[0]
                elif simbolo == " ":
                    if x == 0 and y == 0:
                        simbolo = "+"
                    elif x == 0:
                        simbolo = "|"
                    elif y == 0:
                        simbolo = "-"

                linea += simbolo
            print(linea)


if __name__ == "__main__":
    graf = GraficadoraTexto(xmin=0, xmax=12, ymin=0, ymax=12)

    graf.agregar_funcion("4", "*")       # x >= 4 (l nea vertical)
    graf.agregar_funcion("6", "@")       # y >= 6 (l nea horizontal)
    graf.agregar_funcion("12 - x", "#")  # x + y <= 12 (l nea diagonal)

    graf.graficar(sombrear=True)
```

Interpretación de la Solución Gráfica

- La región sombreada (.) representa todas las combinaciones posibles de horas que cumplen simultáneamente las tres restricciones.

Figura 3: Ejecucion de Ejercicio 03

- La línea vertical (`*`) indica $x = 4$ (mínimo de reuniones).
- La línea horizontal (`@`) indica $y = 6$ (mínimo de documentación).
- La línea diagonal (`#`) indica $x + y = 12$ (tiempo total máximo).
- Cualquier punto dentro de la región sombreada es una asignación válida de tiempo para reuniones y documentación.

Vértices de la Región Factible

$$A = (4, 6) \tag{25}$$
$$B = (4, 8) \tag{26}$$
$$C = (6, 6) \tag{27}$$

Ejemplos de Combinaciones Posibles

- $(5, 6)$: 5 horas reuniones + 6 horas documentación = 11 horas
- $(4, 7)$: 4 horas reuniones + 7 horas documentación = 11 horas
- $(6, 6)$: 6 horas reuniones + 6 horas documentación = 12 horas

Ejercicio 4: Producción de Assets de Videojuegos

Enunciado del Problema

Una empresa de desarrollo de videojuegos produce dos tipos de *assets*: Modelos 3D (x) y Texturas (y). Cada modelo 3D requiere 2 horas de trabajo y cada textura requiere 3 horas. El equipo de arte dispone de un total de 18 horas semanales.

Se solicita formular las restricciones, representarlas gráficamente y determinar cuántos *assets* de cada tipo pueden producirse en función del tiempo disponible.

Formulación Matemática

Variables de decisión:

$$x = \text{número de Modelos 3D (P1)} \tag{28}$$
$$y = \text{número de Texturas (P2)} \tag{29}$$

Restricciones:

$$2x + 3y \leq 18 \tag{30}$$

$$x \geq 0, \quad y \geq 0 \tag{31}$$

Ecuación de frontera:

$$y = 6 - \frac{2}{3}x \tag{32}$$

Implementación del Software

```python
class GraficadoraTexto:
    def __init__(self, xmin=0, xmax=10, ymin=0, ymax=7):
        self.xmin = xmin
        self.xmax = xmax
        self.ymin = ymin
        self.ymax = ymax
        self.funciones = []

    def preparar_expresion(self, expr: str) -> str:
        expr = expr.replace(" ", "")
        expr = expr.replace("^", "**")
        if expr.startswith("x"):
            expr = "1*" + expr
        expr = expr.replace("-x", "-1*x")
        expr = expr.replace("+x", "+1*x")
        expr = expr.replace("x", "*x")
        expr = expr.replace("**x", "*x")
        return expr

    def agregar_funcion(self, expresion: str, simbolo: str):
        expr_preparada = self.preparar_expresion(expresion)
        self.funciones.append((expr_preparada, simbolo))

    def graficar(self, sombrear=False):
        print("\nGr fico en Plano Cartesiano\n")
        for y in range(self.ymax, self.ymin - 1, -1):
            linea = ""
            for x in range(self.xmin, self.xmax + 1):
                simbolo = " "
                intersecciones = []

                if sombrear:
                    if 2*x + 3*y <= 18 and x >= 0 and y >= 0:
                        simbolo = "."

                for expr, simb in self.funciones:
                    try:
                        y_eval = eval(expr, {"x": x})
                        if abs(y - y_eval) < 0.5:
                            intersecciones.append(simb)
                    except:
                        pass

                if len(intersecciones) > 1:
                    simbolo = "#"
                elif len(intersecciones) == 1:
                    simbolo = intersecciones[0]
                elif simbolo == " ":
                    if x == 0 and y == 0:
                        simbolo = "+"
```

```
                    elif x == 0:
                        simbolo = "|"
                    elif y == 0:
                        simbolo = "-"

                linea += simbolo
            print(linea)


if __name__ == "__main__":
    graf = GraficadoraTexto(xmin=0, xmax=10, ymin=0, ymax=7)
    graf.agregar_funcion("(18 - 2*x)/3", "*")
    graf.graficar(sombrear=True)
```

Figura 4:

Interpretación de la Solución Gráfica

- **Frontera (*):** Representa la ecuación $2x + 3y = 18$.
- **Región sombreada (.):** Conjunto de soluciones factibles que satisfacen $2x + 3y \leq 18$.
- **Vértices:** $(0, 0)$, $(9, 0)$, $(0, 6)$.

Cada punto dentro de la región sombreada indica una combinación factible de producción. Ejemplos:

- (3,4): $2(3) + 3(4) = 18$
- (6,2): $2(6) + 3(2) = 18$
- (2,5): $2(2) + 3(5) = 19 > 18$ (no factible)

Ejercicio 5: Ensamblaje de Dispositivos con Componentes Electrónicos

Enunciado del Problema

Una startup de hardware dispone de un máximo de 50 unidades de componentes electrónicos. Cada dispositivo tipo A requiere 5 unidades y cada dispositivo tipo B requiere 10 unidades.

Se solicita determinar cuántos dispositivos de cada tipo pueden ensamblarse sin exceder las 50 unidades de componentes, formular las restricciones y representarlas gráficamente.

Formulación Matemática

Variables de decisión:

$$x = \text{número de dispositivos tipo A} \tag{33}$$

$$y = \text{número de dispositivos tipo B} \tag{34}$$

Restricciones:

$$5x + 10y \leq 50 \tag{35}$$

$$x \geq 0, \quad y \geq 0 \tag{36}$$

Ecuación de frontera:

$$y = 5 - 0{,}5x \tag{37}$$

Implementación del Software

```python
class GraficadoraTexto:
    def __init__(self, xmin=0, xmax=12, ymin=0, ymax=6):
        self.xmin = xmin
        self.xmax = xmax
        self.ymin = ymin
        self.ymax = ymax
        self.funciones = []

    def preparar_expresion(self, expr: str) -> str:
        expr = expr.replace(" ", "")
        expr = expr.replace("^", "**")
        if expr.startswith("x"):
            expr = "1*" + expr
        expr = expr.replace("-x", "-1*x")
        expr = expr.replace("+x", "+1*x")
        expr = expr.replace("x", "*x")
        expr = expr.replace("**x", "*x")
        return expr

    def agregar_funcion(self, expresion: str, simbolo: str):
        expr_preparada = self.preparar_expresion(expresion)
        self.funciones.append((expr_preparada, simbolo))

    def graficar(self, sombrear=False):
        print("\nGr fico en Plano Cartesiano\n")
        for y in range(self.ymax, self.ymin - 1, -1):
            linea = ""
            for x in range(self.xmin, self.xmax + 1):
                simbolo = " "
                intersecciones = []

                if sombrear:
                    if 5*x + 10*y <= 50 and x >= 0 and y >= 0:
```

```
                        simbolo = ","

                for expr, simb in self.funciones:
                    try:
                        y_eval = eval(expr, {"x": x})
                        if abs(y - y_eval) < 0.5:
                            intersecciones.append(simb)
                    except:
                        pass

                if len(intersecciones) > 1:
                    simbolo = "#"
                elif len(intersecciones) == 1:
                    simbolo = intersecciones[0]
                elif simbolo == " ":
                    if x == 0 and y == 0:
                        simbolo = "+"
                    elif x == 0:
                        simbolo = "|"
                    elif y == 0:
                        simbolo = "-"

                linea += simbolo
            print(linea)


if __name__ == "__main__":
    graf = GraficadoraTexto(xmin=0, xmax=12, ymin=0, ymax=6)
    graf.agregar_funcion("5 - 0.5*x", "*")
    graf.graficar(sombrear=True)
```

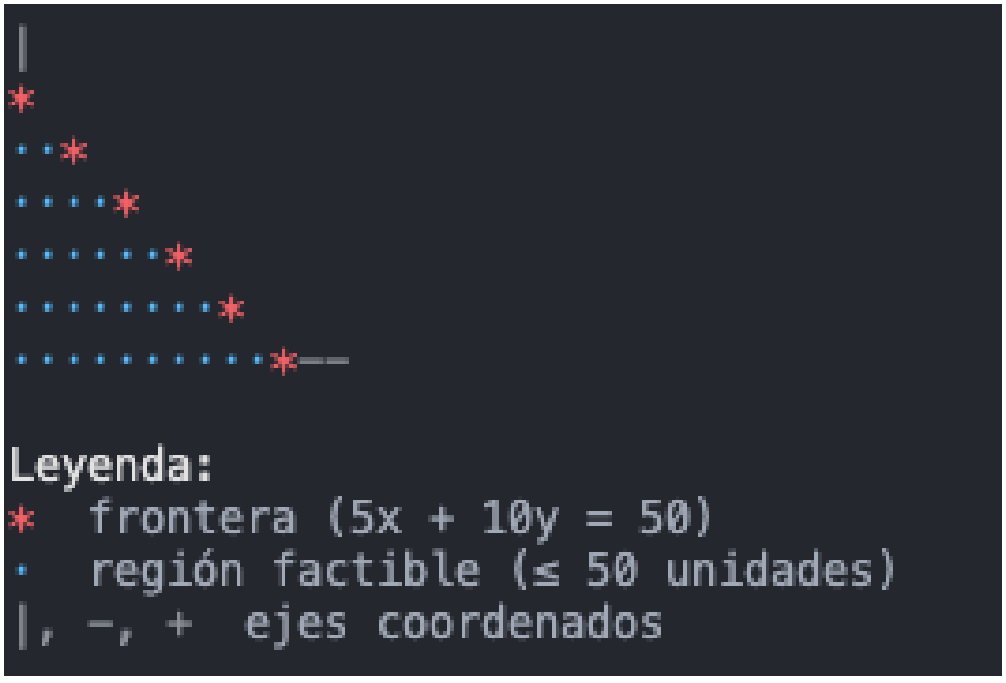

Figura 5:

Interpretación de la Solución Gráfica

- **Frontera (*):** Representa la ecuación $5x + 10y = 50$.

- **Región sombreada (,):** Indica la zona donde $5x + 10y \leq 50$.
- **Vértices:** $(0, 0)$, $(10, 0)$, $(0, 5)$.

Cada punto en la región sombreada representa una combinación de producción posible. Ejemplos:

- (4,3): $5(4) + 10(3) = 50$
- (6,2): $5(6) + 10(2) = 50$
- (8,1): $5(8) + 10(1) = 50$