Universidad de Concepción
Facultad de Ingeniería
Departamento de Ingeniería Industrial

Programación Aplicada a la Ingeniería Industrial

Tarea 2

Profesor: Carlos Contreras Bolton **Fecha:** 16 de Noviembre 2020

1. Entrega

Lunes 30 de Noviembre del 2020 hasta las 23:59 hrs en Canvas.

2. Objetivo de la tarea

a) Uso de estructuras de datos no lineales como los árboles binarios.

b) Aplicación del paradigma de orientación a objetos.

c) Uso de una biblioteca/módulo externo como Cplex.

3. Enunciado

Implementar el algoritmo de *branch and bound* o en español conocido como "ramificación y poda" o "ramificación y acotación", la versión particular que resuelve problemas de programación lineal entera binaria. La implementación se debe realizar mediante el uso de un árbol binario, cualquiera de sus tipos de implementación. Además, se deben resolver los sub-problemas de programación lineal con Simplex, por ende se debe resolver en cada nodo un problema relajado de programación lineala entera binaria. Se recomienda el uso de Cplex para resolver cada sub-problema relajado. Algunas consideraciones:

- Se debe implementar la versión más simple del algoritmo. Es decir, para la "ramificación" puede escoger el recorrido de árbol que estime conveniente. Mientras, que para la "poda", solo se podará si el nodo a ramificar no tiene un costo prometedor, es decir, es mayor a la cota superior si es un problema de minimización, o es menor a la cota inferior si es de maximización.
- Cualquier adición para mejorar la eficiencia del algoritmo será bonificada.
- La implementación debe estar completamente hecha en el paradigma de orientación a objetos.
- En cada nodo se debe resolver un subproblema de programación lineal y este se debe realizar mediante el algoritmo de Simplex, se recomienda utilizar Cplex.

4. Entrada y salida

La entrada es un archivo de texto plano que contiene es el siguiente formato: La primera línea indica cuántas variables de decisión tiene el problema, luego un espacio, y posteriomente, el tiempo límite para la ejecición del algoritmo de *branch and bound*. La segunda línea indica si es un problema de minimización o maximización. La tercera línea comienza con "Fobjetivo:" y luego la expresión de la función objetivo considerando el siguiente formato: espacio, costo númerico (real o entero), identificador de la variable de

decisión, operador (suma o resta), esto se repite cuantas términos tenga la función objetivo, obviamente el último término no finaliza con el operador. La cuarta línea contiene "Sujeto a". Luego, las siguientes últimas líneas corresponden a las restricciones, cada línea contiene en el inicio a Restricción_1, ..., Restricción_n, donde n es el número de restricciones del problema. Cada restricción tiene el formato: espacio, costo númerico (real o entero), identificador de la variable de decisión, operador (suma o resta), esto se repite cuantas términos tenga la función objetivo, obviamente el último término no finaliza con el operador, sino que con un operador lógico (<= para $\leq$, >= para $\geq$ y == para $=$) y luego un número que representa el lado derecho de una restricción. Considerar que si el valor númerico es un 1, entonces se omite el número y solo va la variable de decisión.

Para la salida, la primera línea es el número de nodos explorados para encontrar la solución óptima, usando "Nodos: " y luego el número de nodos explorados. La segunda línea es el tiempo computacional usado por el *branch and bound* en segundos, usando "Tiempo: y luego el tiempo en segundos. La tercera línea es el valor de la función objetivo, el valor óptimo si se encontró, sino mejor costo en el caso de terminar por el tiempo límite, se usa "Fobjetivo" y luego el valor encontrado. Las siguientes últimas líneas corresponden a las variables de decisión con solo valores de unos. Con el formato de "x_i = 1", con i como el índice de las variables de decisión. Dado el problema de programación entera binaria de las Ecuaciones 1 y 2 se puede observar su entrada en `entrada.txt` y su salida en `salida.txt`.

$$\text{máx } z = 24x_0 + 2x_1 + 20x_2 + 4x_3 \tag{1}$$

s. a:

$$8x_0 + x_1 + 5x_2 + 4x_3 \leq 9 \tag{2}$$

entrada.txt

```
1 4 5
2 Maximizar
3 Fobjetivo: 24 x_0 + 2 x_1 + 20 x_2 + 4 x_3
4 Sujeto a
5 Restricion_1: 8 x_0 + x_1 + 5 x_2 + 4 x_3 <= 9
```

salida.txt

```
1 Nodos: 6
2 Tiempo: 1.2
3 Fobjetivo: 26
4 x_0 = 1
5 x_1 = 1
```

5. Evaluación

La tarea es evaluada de acuerdo a los siguientes criterios:

- Implementación simple para problemas con variables de un índice (4 ptos).
- Implementación considerando problemas con más de un índice (1,5 ptos).
- Visualizaciones del árbol de búsqueda (0,5 ptos).
- Bonificación por estrategias para mejorar la eficiencia del algoritmo de *branch and bound*.

6. Condiciones de la tarea

- La tarea es individual.
- Las dudas respecto al trabajo, se realizan de manera personal (vía Teams) o vía correo electrónico.
- Está estrictamente prohibido resolver el problema sin relajarlo.
- El lenguaje a utilizar es Python 3.x de manera estándar.
- El enunciado podría sufrir modificaciones que serán publicadas en Canvas.
- Formato de entrega:
  - El sistema debe ser robusto, se penalizarán las caídas de cualquier tipo.
  - Debe estar bien documentado.

- La entrega se hace mediante la plataforma Canvas, en el link disponible para subir la tarea.
- En caso de detectarse copia, los estudiantes involucrados tendrán la nota mínima sin apelación.
- El archivo final subido a la plataforma debe ser un archivo comprimido con el siguiente formato:

  `letraInicialNombre_ApellidoPaterno_letraInicialApellidoMaterno.extensión`
  Ejemplo: Juan Pérez Valdivia `j_Pérez_V.zip`
- El código debe estar en un archivo de extensión ***.py*** y debe ser nombrado igual que el archivo comprimido.
- Si considera necesario, se puede agregar un archivo de texto plano, llamado `LEEME.txt`, donde se pueden agregar instrucciones para que el profesor pueda ejecutar de manera correcta su tarea. Sin embargo, no se permiten instrucciones en la cuales se deba intervenir el código para el buen funcionamiento.
- En caso de nos seguir alguna de las condiciones o instrucciones será penalizado con 1.0 punto por cada infracción.