3. A continuacion explicaremos como funciona la eliminacion de subtours de cada modelo utilizado, para esto primero explicaremos que es un subtours

Un subtour es un recorrido de ciudades que NO incluye la ciudad de partida o NO incluye todas las ciudades.

Modelo TSP_MTZ:

En el modelo MTZ se usa un metodo para eliminar los subtours muy similar al visto en clase. Para esto se agrega la siguiente restriccion:

para todo i, j perteneciente a ciudades sin contar el inicio y i != j:

u[i] - u[j] + (n - 1) * x[< i, j >] <= n - 2;

siendo:

u[i] : la posicion de la ciudad i en el recorrido
x[i,j] : variable vivalente que se encuentra en 1 si la ciudad j es la siguiente a la ciudad i es decir u[i] + 1 = u[j]
n : numero de ciudades

Para demostrar que tanto u[i] como x[i,j] toman los valores segun lo sugerido imaginemos que y[i,j] = 1 entonces:

u[i] - u[j] <= (n-2) - (n-1)

u[i] - u[j] <= -1

u[j] >= u[i] + 1

Por lo tanto u[j] debe ser mayor a u[i] en almenos 1 unidad y a continuacion demostraremos que sea exactamente en 1 unidad:

Supongamos que k es la primera ciudad visitada luego del punto de partida y que l es la ultima ciudad visitada, por lo tanto y[k,l] = y[l,k] = 0

u[l] - u[k] <= n - 2

si k es la primera ciudad visitada luego del punto de partida, u[k] = 2 (por como estan modelizadas nuestras ciudades) y por lo tanto:

u[l] - 2 <= n - 2

por lo tanto u[l] <= n pero u[l] podria ser menor a n?

si asi fuera alguna de las variable u deberia repetir su numero y no se cumpliria lo visto anteriormente que u[j] >= u[i] + 1 si y[i,j] = 1

De esta forma demostramos que y[i,j] es uno solo si i es la ciudad anterior a j y que u[j] toma la posición de la ciudad j en el recorrido. y esto a la vez sirve para eliminar la posibilidad de subtoures.

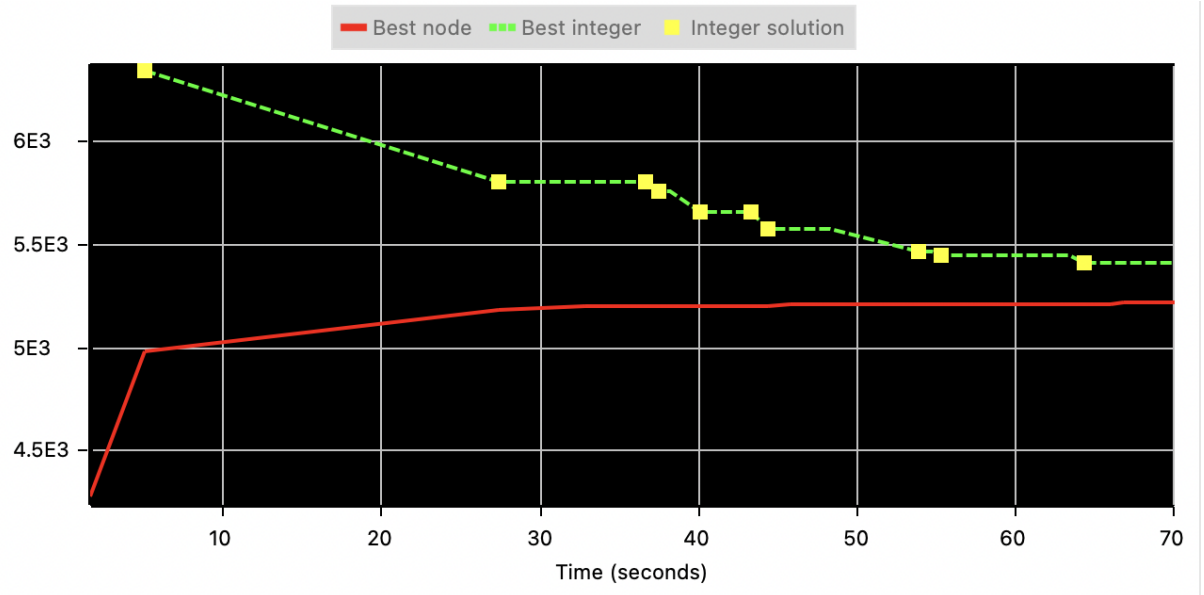
Modelo TSP:

En este modelo, la forma de eliminar subtours es usando el siguiente algoritmo

```
forall ( s in subtours )
  sum ( i in Cities : s.subtour[i] != 0 ) x[< minl ( i, s.subtour[i] ),
    maxl ( i, s.subtour[i] ) >] <= s.size - 1;
```

De esta forma a medida que se generan los subtours el programa no tiene permitido agregar ciudades que ya se encuentren en el subtour por otro lado el algoritmo en si no corta hasta que el largo del subtour sea igual a la cantidad de ciudades brindadas por lo que el subtour final no tendra ciclos y habra recorrido todas las ciudades y vuelto a la inicial.

Como dice el enunciado el algoritmo usa un ordered al momento de crear las variables X[i,j] esto es posible ya que el problema puede ser resuelto con un grafo dirigido de no ser asi no podría resolverse de esta forma. Esto nos permite tener la mitad de variables X.



En el grafico del método MTZ se puede ver como el programa llega a una solucion usando numero no eteros la cual es la linea roja y luego se va acercando a este resultado con la linea verde de manera asintotica, cada cuadrado amarillo es una solucion entera posible.

5. Al agregar la solución inicial al programa, el mismo corre en un tiempo menor, esto se debe a que CPLEX usa esta solución para eliminar parte del espacio de búsqueda a la hora de iniciar el proceso de ramificación y corte, sin embargo la diferencia es casi imperceptible y esto se puede deber al calidad de la solución utilizada. En el gráfico si se puede ver como al agregar la solución inicial esta figura desde el momento 0 y se parte desde esta para encontrar las siguientes soluciones