Práctica 2 de Programación con restricciones

1. Restricciones básicas

Importante: para que el programa descrito funcione correctamente, los aceites vegetales siempre deben aparecer en los arrays antes que los no vegetales.

En primer lugar, se explicará el sistema empleado para comprar y refinar el aceite. Después, se desarrollarán el resto de restricciones.

Para gestionar la compra-venta de aceite se han empleado tres arrays para mostrar el estado de cada operación en cada mes.
Se tiene el array **iniMes**, que indica las toneladas de aceite que se tienen en el almacén cuando se inicia el mes y aún no se ha realizado la compra de dicho mes. En el caso de enero, se pide que empiece con unas toneladas determinadas de aceite en el almacén (datos proporcionados en el array **excedente**). Esta misma cantidad es con la que se debe acabar el año (expresado en **final**, que tiene meramente carácter informativo).
El siguiente array que permite expresar la solución es **comprado**, que indica cuánto aceite se compra al inicio de cada mes.
Por último, se tiene **refinado**, que muestra cuántas toneladas de cada tipo de aceite se han refinado cada mes.

El precio de compra de cada tipo de aceite en cada mes viene dado en el array **coste** y el precio de venta en euros por cada tonelada del producto viene especificado en **valor**.

Las líneas de producción pueden refinar un máximo de aceite cada mes establecido en **maxV** (para aceite vegetal) y **maxN** (para el caso contrario). Para ello se tienen dos restricciones que comprueban que la suma de aceite refinado de cada categoría en un mes no exceda los valores máximos.

La capacidad máxima de almacenamiento de la fábrica es **mCap**, que simboliza el número de toneladas máximo para cada aceite que se puede almacenar en la fábrica. Para no superar el almacenamiento, todos los meses se comprueba que la suma del aceite no consumido en el mes anterior (**iniMes**) y el aceite comprado en el mes actual (**comprado**) no sobrepasan el límite de capacidad.

Almacenar el aceite tiene un coste **CA** por cada tonelada almacenada en cada mes. Para calcular este coste vamos a realizar una serie de suposiciones. El aceite que será almacenado en la fábrica será el que no se vaya a refinar y se acumule para el mes siguiente (cumple con haber estado un mes almacenado y con que el aceite refinado no puede ser almacenado).
Se sabe que se debe pagar por el aceite que se almacena en la fábrica, es decir, el que no se refina. Por tanto, la cantidad de toneladas que se almacenan en un mes coincide con el valor de **iniMes** del mes siguiente. En el caso de diciembre, sabemos que no se va a utilizar **excedente** cantidad de aceite, que en la especificación inicial del problema, coincide con el valor de **iniMes** del primer mes. A continuación, se muestra una figura que ayudará en la explicación de la restricción que permite calcular el coste de almacenamiento.

Sumar todos los valores de **iniMes** es una opción pero en previsión de que la fábrica empiece el año con una cantidad de excedente determinada y finalice con otra, se sumarán los valores de **iniMes** a partir del segundo mes (**costeAlmacM**) y se añadirá la cantidad de **excedente** del último mes (**costeAlmacFin**). Debido a que va en contra del enunciado, no se establecerán valores distintos para el excedente inicial y el final aunque podría ser una posible extensión. Para hacer la extensión, bastaría con añadir un nuevo array de entrada con los datos respectivos y modificar la restricción (cambiar **excedente** por el nombre del nuevo array) que aparece debajo del comentario "%diciembre acaba con excedente".
Por tanto tenemos definido el coste de almacenamiento del aceite no refinado como **costeAlmacM** + **costeAlmacFin**.

El siguiente gasto al que se enfrenta la fábrica es al coste del aceite que compra para refinar. Este es el número de toneladas de aceite que ha comprado por el precio de cada una. Su valor es **costeAceiteCompra**.

Antes de calcular el beneficio obtenido por la empresa, se deben calcular sus ingresos por generar aceite refinado. Como el producto es común a todos los aceites, su valor será constante (entrada **valor**). Las **ganancias** son por tanto el número de toneladas de aceite refinado por el **valor** que genera cada tonelada.

Por tanto tenemos que el **beneficio** son las **ganancias** menos los distintos tipos de coste (**costeAlmacM** + **costeAlmacFin** + **costeAceiteCompra**).
El **beneficio** debe cumplir ser mayor o igual a **minB**. El **beneficio** es el valor que se va a maximizar.

Para calcular la dureza del producto, se guarda en una variable auxiliar **totalRefinado** el número de toneladas de aceite que se han refinado en cada mes. Como la dureza final debe estar entre un mínimo **minD** y un máximo **maxD**, se han codificado dos restricciones que comprueban que al hacer la media ponderada de la dureza del producto final, este valor no se salga del intervalo [**minD**, **maxD**]. La suma de la cantidad de cada tipo de aceite multiplicada por su dureza debe compararse con el producto de la cantidad total con cada extremo del intervalo.

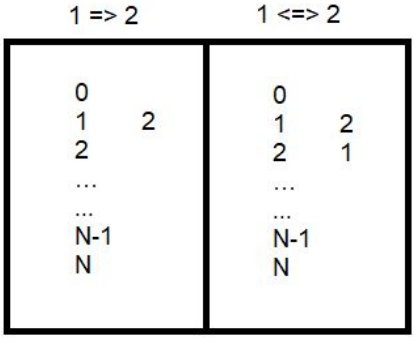
2. Extensiones

a. Propuestas en el enunciado

- Extensión 1: “El producto no debe hacerse con más de K aceites”
  Si en cualquiera de los meses para un tipo de aceite el valor de **refinado** es mayor a 0 toneladas, implica que este ha sido usado. Por tanto se debe contar el número del tipo de aceites que se refinan en un mes y que este valor sea igual o inferior a **K**.

- Extensión 2: “Si un mes usamos un cierto aceite, entonces debemos usar como mínimo T toneladas”
  Para todos los meses y todos los aceites si uno de ellos se refina (**refinado** > 0), entonces se tienen que usar para refinar ese aceite **T** toneladas o más.

- Extensión 3 generalizada: “Si usamos un aceite en un cierto mes, entonces otro aceite también debe ser usado ese mes”
  La restricción dice que si se usa un aceite (**refinado** > 0), el otro también debe ser usado.
  MiniZinc: Para expresar que el uso de un aceite implica tener que refinar otro, se ha empleado como dato de entrada el array **obligacionMezcla**. En este array se permiten implicaciones simples o dobles. Si se quiere indicar que usar el aceite de la posición 1 implica usar el aceite de la posición 2 se tendría obligacionMezcla = [{2},{},{},...,{},{}]. Si se quiere indicar usar uno de los aceites implica el uso del otro se expresaría como obligacionMezcla = [{2},{1},{},...,{},{}].
  Python/SMT2: en esta versión el nombre de la entrada se llama **dependencias** y es un diccionario que asocia clave (número del aceite que genera la dependencia) y valor (número del o de los aceites que se deben usar en caso de utilizar el de la clave). Si un aceite no genera dependencias, se introducirá ese aceite como clave y no se le asociará ningún valor. Para el ejemplo anterior se tendría:



  Si usar el aceite 1 implicase usar el 2, 3, 7 y 9 se tendría: “1 2 3 7 9”
  Si usar el aceite 1 no obliga a usar otro aceite, se tendría: “1”

b. Nuevas

- Mínimo aceite de cada tipo que se debe haber comprado al finalizar el año
  La fábrica puede negociar con los proveedores del aceite que si se comprometen a comprar un mínimo de toneladas a final de año les harán un rappel en la factura. Para ello, a final de año la cantidad comprada de aceite debe ser mayor o igual al mínimo de toneladas acordadas (**minComprar**).
  Como puede ser que cada tipo de aceite lo adquieran de distintos proveedores o aunque sea el mismo, la cantidad acordada a comprar varíe, se tiene el array **minComprar** donde se especifica las toneladas mínimas que se deben haber comprado de cada aceite al finalizar el año.

- Cambiar el periodo de tiempo en el que se gestiona la fábrica
  Puede darse el caso de que la fábrica quiera aumentar o disminuir el periodo de tiempo en el que se gestiona esta. En vez de ser anual, pueden aumentar el período a 5 años para hacer un cálculo aproximado de cómo evolucionarán los beneficios de la empresa o qué volumen de compras y venta deben cumplir para alcanzar un objetivo de beneficio a largo plazo.
  Si la empresa se encuentra en una mala situación económica y dispone de un breve periodo de tiempo para afrontar las facturas, también pueden reducir el periodo de tiempo a 6 meses (por ejemplo). Así podrán establecer el beneficio al que deben llegar en ese periodo para poder salvar la situación y también, conocer cuánto aceite se debe comprar y refinar para alcanzar el objetivo.
  Es la generalización del periodo de tiempo en el que se gestiona la fábrica. Para ello se tiene el entero **numMeses**, que indica cuántos meses tiene el periodo a gestionar.

- Variar la cantidad de tipos de aceite
  Se trata de otra generalización del problema. En los .dzn se pueden introducir todos los tipos de aceite que se vayan a usar especificando cuántos de ellos son vegetales (**numVeg**) y cuáles no lo son (**numNoVeg**).
  Es muy importante que aparezcan al inicio todos los aceites vegetales y después, todos los no vegetales.

- Cambiar que el excedente de inicio del periodo tenga que ser igual al de fin
  Se podría tener un valor con el que comenzar el año y exigir que el aceite no refinado a final de año esté entre un mínimo y un máximo que puede variar en función de la producción anual. No implementado por entrar en contradicción con el enunciado.

- Exigir un mínimo de producción mensual
  En una fábrica se puede exigir cumplir con un mínimo de producción mensual para poder afrontar los gastos (a parte de los calculados en el problema se deben pagar salarios, mantenimiento, limpieza...). Por ello para todos los meses se debe cumplir que el producto final (suma de todos los aceites **refinado**s) sea mayor que **minProduccion**, que es la cantidad mínima de toneladas de aceite que se deben refinar cada mes.

- Existencia de aceites que no se pueden combinar
  Pueden existir aceites que por sus propiedades no se pueden combinar cuando se crea el producto final.
  MiniZinc: Se proporcionará como entrada un array con los aceites **incombinables** y ambos no podrán estar simultáneamente en la mezcla.
  SMT2: La entrada es un diccionario llamado incompatibilidad que determina que aceites no se pueden mezclar en el producto final.

3. Tiempos MiniZinc

Para todos los casos de prueba se tarda en conseguir la primera solución menos de un segundo. Para encontrar la mejor solución:

- data0: más de 30 minutos y no encuentra el óptimo
- data1: más de 5 horas 43 minutos y no encuentra el óptimo
- data2: más de 4 horas 23 minutos y no encuentra el óptimo
- data3: más de 1 horas 8 minutos y no encuentra el óptimo (beneficio = 97210)
- data4: más de 13 horas 21 minutos y no encuentra el óptimo (beneficio = 103360)
- data5: más de 7 horas y no encuentra el óptimo (beneficio = 96695)
- data6: más de 4 horas 40 minutos y no encuentra el óptimo (beneficio = 96892)
- data7: más de 2 horas 32 minutos y no encuentra el óptimo (beneficio = 126156)
- data8: más de 6 horas 52 minutos y no encuentra el óptimo (beneficio = 119570)
- data9: más de 2 horas 49 minutos y no encuentra el óptimo (beneficio = 162100)

4. Tiempos SMT2

A partir de input3_aceite (incluido), los datos introducidos en MiniZinc y en SMT son iguales para el mismo número de input.

Maximize():

- input0_aceite : inmediato, óptimo (beneficio = 125310)
- input1_aceite : inmediato, óptimo (beneficio = 126010)
- input2_aceite : inmediato, óptimo (beneficio = 80340)
- input3_aceite : inmediato, óptimo (beneficio = 125800)
- input4_aceite : inmediato, óptimo (beneficio = 125286)
- input5_aceite : inmediato, óptimo (beneficio = 96890)
- input6_aceite : inmediato, óptimo (beneficio = 125120)
- input7_aceite : inmediato, óptimo (beneficio = 137570)
- input8_aceite : inmediato, óptimo (beneficio = 251100)
- input9_aceite : 1 segundo, óptimo (beneficio = 363250)
- input12_aceite: 6 segundos, óptimo (beneficio = 758000)

Max-SMT:

Para la parte de optimización de Max-SMT se han asignado distintos pesos al posible valor final del beneficio y se ha dejado como hard el mínimo beneficio pedido ya que es una restricción que siempre se debe cumplir. Es decir, se han establecido varias condiciones soft a distintos intervalos del posible valor que pueda tomar el beneficio; por ejemplo, (beneficio, beneficio+1000] peso 10, (beneficio+1000, beneficio+2000] peso 20 ... Y así para varios intervalos hasta (beneficio+x, valor máximo que acepte smt) siendo beneficio + x = 11 * minB.

- input0_aceite : inmediato, no mejor solución (beneficio = 125000, diferencia = 310)
- input1_aceite : inmediato, no mejor solución (beneficio = 125002, diferencia = 1008)
- input2_aceite : inmediato, no mejor solución (beneficio = 80001, diferencia = 339)
- input3_aceite : inmediato, no mejor solución (beneficio = 117021, diferencia = 8779)
- input4_aceite : inmediato, no mejor solución (beneficio = 117000, diferencia = 8286)
- input5_aceite : inmediato, no mejor solución (beneficio = 96552, diferencia = 338)
- input6_aceite : inmediato, no mejor solución (beneficio = 123513, diferencia = 1607)
- input7_aceite : inmediato, no mejor solución (beneficio = 132000, diferencia = 5570)
- input8_aceite : 1s, no mejor solución (beneficio = 249900, diferencia = 1200)
- input9_aceite : 1.5s, no mejor solución (beneficio = 357020, diferencia = 6230)
- input12_aceite: 5s, no mejor solución (beneficio = 748025, diferencia = 10075)

Los tiempos no son reales, pero indican si la solución se ha mostrado de forma inmediata o ha tardado un poco en aparecer.

Las soluciones obtenidas con Max-SMT no distan mucho de las conseguidas con maximize y esto es debido a que se han introducido valores para minB que no son excesivamente pequeños en comparación con la solución. Es por esto que el resultado pertenece a alguno de los intervalos fijados que no sea el de (11 * minB, +infinito). Sin embargo, si el valor de minB es muy bajo, este sí pertenecerá dicho intervalo y no se conseguirá una buena optimización. En concreto se tiene:

➢ input0_minB_bajo: datos iguales a input0_aceite del apartado Max-SMT con beneficio mínimo de 1000. El beneficio obtenido es de 60350, que es aproximadamente la mitad del beneficio conseguido con maximize.
Se debe a que el assert-soft que contempla el intervalo con los valores más altos es *“(assert-soft (>= beneficio 11000) :weight 10000.0)”*, por lo que cualquier beneficio obtenido por encima de 11000 se interpretará como igual de bueno.

Una posible solución sería añadir más intervalos de forma que se tenga (x * minB, +infinito) siendo x un valor mayor a 11 (que es el valor empleado en la solución propuesta).
Pensando que se van a manejar datos reales y que el valor de minB será cercano al máximo beneficio posible a obtener, se puede suponer que el resultado no será 11 veces minB. También, para mejorar la precisión de los resultados obtenidos se podrían establecer intervalos más pequeños.

Max-SMT v2:
En esta versión, se han añadido 3 arrays para representar de forma mensual la ganancia de la empresa y los gastos de esta respectivamente. A los asserts del apartado anterior, se le han añadido condiciones para lograr una ganancia mayor y costes menores. Además, se han establecido id para el beneficio total y el mensual. El peso que se le ha dado a estos nuevos asserts es tres cuartos del máximo peso del beneficio total (el perteneciente al intervalo (minB*11,+infinito)). Los nuevos asserts soft son:

1. Lo que se gane en un mes, debe ser mayor o igual que el mínimo beneficio que queremos lograr en el periodo entre el número de meses en el que se realiza la actividad.
2. El coste de compra de aceite no debe ser mayor que un tercio de lo que se gane ese mes.
3. El coste de almacenamiento del aceite no debe ser mayor que un quinto de lo que se gane ese mes.

Como resultado se obtiene (comparando con maximize):

- input0_aceite : 1s, no mejor solución (beneficio = 125010, diferencia = 300)
- input1_aceite : 1s, no mejor solución (beneficio = 125002, diferencia = 1008)
- input2_aceite : 1s, no mejor solución (beneficio = 80070, diferencia = 270)
- input3_aceite : 1s, no mejor solución (beneficio = 117077, diferencia = 8723)
- input4_aceite : 1s, no mejor solución (beneficio = 117011, diferencia = 8275)
- input5_aceite : 1s, no mejor solución (beneficio = 96787, diferencia = 103)
- input6_aceite : 1s, no mejor solución (beneficio = 123504, diferencia = 28616)
- input7_aceite : 1s, no mejor solución (beneficio = 132057, diferencia = 5513)
- input8_aceite : más de 22min, no terminado (beneficio = 130905)
- input9_aceite : más de 10min, no terminado (beneficio = 172045)
- input12_aceite: más de 15min, no terminado (beneficio = 357060)

Si el fondo aparece verde, es que se ha conseguido una mejor solución que con la otra versión de Max-SMT. Si es rojo implica que la solución es peor. En caso de ser azul, se ha obtenido el mismo resultado. Si el fondo es amarillo, el programa no ha terminado pero llevaba mucho tiempo ejecutándose (no se comparará con otros resultados).
Se puede observar que el programa es mucho más lento al introducir las nuevas variables. Para mejorar la rapidez, se podría probar el código reduciendo el número de intervalos del beneficio total.

5. Explicación datos de entrada para Python

Para saber cómo se introducen los datos en Python, se ha adjuntado un archivo llamado plantilla_input.txt, que como su nombre indica se puede utilizar como plantilla para saber cuál es el formato de la entrada.
Para los inputs especificados en el enunciado, se ha indicado su nombre en el lugar en el que se deben sustituir. En el caso de los arrays y las matrices, se ha puesto un ejemplo general del formato que debe seguir la entrada.
Para mayor claridad al leer la plantilla, se han añadido líneas de separación entre los datos aunque estos deben ser eliminados cuando en la plantilla se introduzcan los valores reales.

Andrea Peña Calvin