Análisis y Diseño de Algoritmos II

Jesús Alexander Aranda Ph.D Robinson Duque, Ph.D
Juan Francisco Díaz, Ph. D

Universidad del Valle

jesus.aranda@correounivalle.edu.co
robinson.duque@correounivalle.edu.co
juanfco.diaz@correounivalle.edu.co
Programa de Ingeniería de Sistemas
Escuela de Ingeniería de Sistemas y Computación

Overview

Ejercicio: Problema de la Dieta

En una clase anterior resolvimos este problema...
Los datos de contenido nutricional de un grupo de alimentos y la necesidad semanal de un adulto se presentan en la tabla que se muestra a continuación. **Determine el costo semanal más bajo para cumplir con los requerimientos mínimos semanales (i.e., 550g de proteina, 600g de grasa, 2000g de carbohidratos).**

	Food	Proteins	Fats	Carbohydrates	Cost $ per 100g
1	Bread	8%	1%	55%	0.25
2	Butter	–	90%	–	0.5
3	Cheese	25%	36%	–	1.2
4	Cereal	12%	3%	75%	0.6
5	Diet Bar	8%	–	50%	1.5
	Weekly requirement (g)	550	600	2000	

Ejercicio: Problema de la Dieta

Modelo final:

$$
\begin{aligned}
\texttt{minimize} \quad & f = 0{,}25x_1 + 0{,}5x_2 + 1{,}2x_3 + 0{,}6x_4 + 1{,}5x_5 \\
\texttt{subject to} \quad & 0{,}08x_1 + 0{,}25x_3 + 0{,}12x_4 + 0{,}08x_5 \geq 550 \\
& 0{,}01x_1 + 0{,}9x_2 + 0{,}36x_3 + 0{,}03x_4 \geq 600 \\
& 0{,}55x_1 + 0{,}75x_4 + 0{,}5x_5 \geq 2000 \\
& x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0, x_5 \geq 0
\end{aligned}
$$

Actividad en Clase

Proponga un modelo genérico para el problema de la dieta. Utilice constantes, arreglos o matrices para representar los datos de entrada, variables, restricciones y función objetivo (Proteinas, grasa, carbohidratos, etc). Escriba su modelo y luego implementelo en MiniZinc.

Ejercicio: Problema de agricultura

En una clase anterior resolvimos este problema...
Un agricultor de vegetales tiene la opción de producir tomates, pimientos verdes o pepinos en su granja de 200 acres. Un total de 500 días-hombre de trabajo están disponibles. En la tabla se muestran los rendimientos y los días-hombre de trabajo por acre:

	Yield $/ acre	Labor man-days/acre
Tomatoes	450	6
Green Peppers	360	7
Cucumbers	400	5

Suponiendo que los costos de los fertilizantes son los mismos para cada producto, determine la combinación óptima de cultivos.

Ejercicio: Problema de agricultura

Sean x_1, x_2 y x_3 los acres de tierra para tomates, pimientos verdes y pepinos respectivamente. El problema del LP puede ser declarado como:

$$
\begin{array}{ll}
\texttt{maximize} & f = 450x_1 + 360x_2 + 400x_3 \\
\texttt{subject to} & x1 + x2 + x3 \leq 200 \\
 & 6x1 + 7x2 + 5x3 \leq 500 \\
 & x1 \geq 0, x2 \geq 0, x3 \geq 0
\end{array}
$$

Actividad en clase

Proponga un modelo genérico para el problema de agricultura. Utilice constantes, arreglos o matrices para representar los datos de entrada, variables, restricciones y función objetivo. Asuma primero un modelo que soporte solamente tomates, pimientos verdes y pepinos. Luego extiéndalo para soportar cualquier número de productos.

Listas y Conjuntos por Comprensión

MiniZinc provee **listas por comprensión** similar a las que ofrecen algunos lenguajes como Python. Por ejemplo, la lista por comprensión:

$$[i + j | i, j \text{ in } 1..3 \text{ where } j < i]$$

se evalúa como $[2 + 1, 3 + 1, 3 + 2]$ que es $[3, 4, 5]$. Por supuesto $[3, 4, 5]$ es simplemente un arreglo con índice establecido $1..3$.

De forma similar, se pueden generar **conjuntos por comprensión**:

$$\{i + j | i, j \text{ in } 1..3 \text{ where } j < i\}$$

se evalúa como $\{3, 4, 5\}$.

Listas y Conjuntos por Comprensión- Listas y Conjuntos por Comprensión

```
[i*2 | i in 1..8] evaluates-to
    [2,4,6,8,10,12,14,16]

[i*j | i,j in 1..3 where i<j]
evaluates-to [2,3,6]

[i + 2*j | i in 1..3, j in 1..4]
evaluates-to [3,5,7,9,4,6,8,10,5,7,9,11]

{i + 2*j | i in 1..3, j in 1..4}
evaluates-to {3,4,5,6,7,8,9,10,11}
```

Listas y Conjuntos por Comprensión

La forma genérica de una **lista por comprensión** es:

```
[ <expr> | <generator-exp> ]
```

La expresión $< expr >$ especifica cómo construir elementos en la lista de salida a partir de los elementos generados por $< generator - exp >$. El generador $< generator - exp >$ consiste en una secuencia separada por comas de expresiones generadoras, seguida opcionalmente por una expresión booleana. Las dos formas son:

```
<generator>
<generator> where <bool-exp>
```

Listas y Conjuntos por Comprensión

El $< bool - exp >$ opcional en la segunda forma actúa como un filtro en la expresión del generador: solo los elementos que satisfacen la expresión booleana se usan para construir elementos en la lista de salida. Un generador $< generator >$ tiene la forma:

```
<identifier>, ..., <identifier> in <array-exp>
```

Cada identificador es un iterador que toma los valores de la expresión de $< array - exp >$ a su vez, y el último identificador varía más rápidamente.

Funciones de agregación

- MiniZinc proporciona una serie de funciones integradas que toman una matriz unidimensional y agregan los elementos. Probablemente el más conocido es: *forall*.
- *forall* toma una matriz de expresiones booleanas (es decir, restricciones) y devuelve una expresión booleana única que es la conjunción lógica de las expresiones booleanas en la matriz.

Por ejemplo, considere la expresión:

```
forall( [a[i] != a[j] | i,j in 1..3 where i < j])
```

donde *a* es una matriz aritmética con índice establecido 1..3. Esto restringe los elementos a ser diferentes. La lista de comprensión se evalúa como `[a[1] != a[2], a[1] != a[3], a[2] != a[3]]` y la función *forall* devuelve la conjunción lógica a[1] != a[2] /\a[1] != a[3] /\a[2] != a[3].

Funciones de agregación

La expresión:

```
forall( [a[i] != a[j] | i,j in 1..3 where i < j])
```

También puede ser escrita como:

```
forall (i,j in 1..3 where i < j) (a[i] != a[j])
```

```
forall (i in 1..3, j in 1..3 where i < j)
          (a[i] != a[j])
```

```
forall (i in 1..3)(
  forall(j in 1..3 where i<j) (a[i] != a[j])
)
```

Funciones de agregación

Las funciones de agregación para las matrices aritméticas son:

- *sum* que suma los elementos
- *product* que multiplica los elementos
- *min* y *max* que devuelven respectivamente el elemento menor y mayor en la matriz

Cuando se aplican funciones de agregación a una matriz vacía, *min* y *max* dan un error de tiempo de ejecución, *sum* devuelve 0 y *product* devuelve 1. Ejemplos de uso:

```
sum(j in 1..n)(w[j] * x[j])

product(i,j in 1..n)(a[i] + b[j])
```

Funciones de agregación

MiniZinc proporciona cuatro funciones de agregación para matrices que contienen expresiones booleanas:

- *forall* devuelve una única restricción que es la conjunción lógica de las restricciones. Impone que todas las restricciones en la matriz se satisfagan
- *exists* devuelve la disyunción lógica de las restricciones. Por lo tanto, asegura que al menos una de las restricciones se satisfaga
- *xorall* garantiza que se satisfaga un número impar de restricciones
- *iffall* asegura que se satisfaga un número par de restricciones

Funciones de agregación

Una función de agregación tiene la forma:

```
<agg-func> ( <generator-exp> ) ( <exp> )
```

Los parentesis que rodean la expresión del generador $< generator - exp >$ y la expresión del constructor $< exp >$ no son opcionales: deben estar allí. Esto es equivalente a escribir:

```
<agg-func> ( [ <exp> | <generator-exp> ] )
```

La función de agregación $< agg - func >$ es cualquier función de MiniZinc que espera una única matriz como argumento.

Ejemplo: Haciendo Pasteles

Suponga que sabemos cómo hacer dos tipos de pasteles (ADVERTENCIA: no use estas recetas en casa):

- Un pastel de banano que toma 250g de harina, 2 bananos, 75g de azúcar y 100g de mantequilla
- Un pastel de chocolate que lleva 200g de harina, 75g de cacao, 150g de azúcar y 150g de mantequilla

Podemos vender un pastel de chocolate por $4.50 y un pastel de plátano por $4.00. Y tenemos 4kg de harina, 6 bananas, 2kg de azúcar, 500g de mantequilla y 500g de cacao.
¿cuántos de cada tipo de torta debemos preparar para la fiesta para maximizar el beneficio?

Ejemplo: Haciendo Pasteles

```
% Variables
var 0..100: b; % no. de pasteles de banano
var 0..100: c; % no. de pasteles de chocolate
% Harina
constraint 250*b + 200*c <= 4000;
% Bananos
constraint 2*b  <= 6;
% Azucar
constraint 75*b + 150*c <= 2000;
% Mantequilla
constraint 100*b + 150*c <= 500;
% Cacao
constraint 75*c <= 500;
% Maximizar el beneficio
solve maximize 400*b + 450*c;
output ["no. de pasteles de banano = \(b)\n",
        "no. de pasteles de chocolate = \(c)\n"];
```

Ejemplo: Haciendo Pasteles

(Ingredients/Resources) per cake

Product/Consume	Flour	Banana	Sugar	Butter	Cacao
Banana	250	2	75	100	0
Chocolate	200	0	150	150	75

Available Resources

Max. Capacity	Flour	Banana	Sugar	Butter	Cacao
Value	4000	6	2000	500	500

(Profit) Cost per Product

Product	**Cost**
Banana	400
Chocolate	450

**Es muy útil dibujar una representación tabular del problema.

Ejemplo: Haciendo Pasteles

¿Cómo lo generalizamos?

Ejemplo: Haciendo Pasteles

¿Cómo lo generalizamos?
Parámetros:

Ejemplo: Haciendo Pasteles

¿Cómo lo generalizamos?

Parámetros: Sea P el conjunto de productos y R el conjunto de recursos disponibles tal que: $i \in P, j \in R$

- $Costo_i$: el costo de venta al producir el pastel (o producto) i
- $Capacidad_j$: capacidad máxima del recurso j
- $Consumo_{ij}$: cantidad que consume producir cada pastel i sobre el recurso j

Ejemplo: Haciendo Pasteles

```
% Archivo de datos
Productos = {PastelBanano, PastelChocolate};
costo = [400, 450]; % en centavos

Recursos={Harina,Banano,Azucar,Mantequilla, Cacao};
capacidad = [4000, 6, 2000, 500, 500];

Consumo= [| 250, 2, 75, 100, 0,
         | 200, 0, 150, 150, 75 |];
```

La nueva característica de este modelo es el uso de tipos enumerados. Esto nos permitirá tratar la elección de Recursos y Productos como parámetros del modelo.

```
array[Productos] of int: costo;
array[Recursos] of int: capacidad;
array[Productos, Recursos] of int: consumo;
```

Ejemplo: Haciendo Pasteles

Sea P el conjunto de productos y R el conjunto de recursos disponibles tal que: $i \in P, j \in R$.

Variables:

Ejemplo: Haciendo Pasteles

Sea P el conjunto de productos y R el conjunto de recursos disponibles tal que: $i \in P, j \in R$.

Variables:

- $produccion_i$: cantidad a producir del pastel (o producto) i
- uso_j: cantidad utilizada del recurso j

Restricciones:

Ejemplo: Haciendo Pasteles

Sea P el conjunto de productos y R el conjunto de recursos disponibles tal que: $i \in P, j \in R$.

Variables:

- $produccion_i$: cantidad a producir del pastel (o producto) i
- uso_j: cantidad utilizada del recurso j

Restricciones:

- No se pueden utilizar más de los recursos disponibles:

$$\forall j \in R, uso_j = \sum_{\forall i \in P} Consumo_{ij} * produccion_i$$

$$\forall j \in R, uso_j \leq Capacidad_j$$

Ejemplo: Haciendo Pasteles

Sea P el conjunto de productos y R el conjunto de recursos disponibles tal que: $i \in P, j \in R$.

Función objetivo:

Ejemplo: Haciendo Pasteles

Sea P el conjunto de productos y R el conjunto de recursos disponibles tal que: $i \in P, j \in R$.

Función objetivo:

- Maximizar:

$$\sum_{\forall i \in P} Costo_i * produccion_i$$

Ejemplo: Haciendo Pasteles

```
% Lectura de datos
enum Productos;
array[Productos] of int: costo;
enum Recursos;
array[Recursos] of int: capacidad;
array[Productos, Recursos] of int: consumo;

% Variables
array[Productos] of var int: produccion;
array[Recursos] of var 0..max(capacidad): uso;
```

Ejemplo: Haciendo Pasteles

```
%No se pueden utilizar más de los recursos
    disponibles:
constraint forall (r in Recursos) (
 uso[r] = sum (p in Productos)(consumo[p, r] *
     produccion[p])
);
constraint forall (r in Recursos) (
 uso[r] <= capacidad[r]
);

%Maximizar costo
solve maximize sum (p in Productos) (costo[p]*
    produccion[p]);

output [ "\(p) = \(produccion[p]);\n" | p in
    Productos ] ++
        [ "\(r) = \(uso[r]);\n" | r in Recursos ];
```

Ejercicio: El dueño del Barco

El propietario de un barco tiene un carguero con una capacidad de 700 toneladas. La empresa transporta contenedores de diferentes pesos para una ruta específica. En el viaje actual, el propietario del barco podría enviar algunos de los siguientes contenedores:

Container	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
Weight	100	155	50	112	70	80	60	118	110	55

Ejercicio: El dueño del Barco

Actividad en clase: Determine los contenedores que se deben transportar de tal manera que maximice la carga transportada.

- Proponga un modelo genérico para el problema.
- Generalice el problema. Utilice notación formal para proponer un modelo que soporte cualquier número de contenedores.
- Implemente los dos modelos en MiniZinc.

Ejercicio: El dueño del Barco

Actividad en clase: Determine los contenedores que se deben transportar de tal manera que maximice la carga transportada y el número de contenedores si le pagan $10 por cada contenedor que transporte.

- Proponga un modelo genérico para el problema.
- Generalice el problema. Utilice notación formal para proponer un modelo que soporte cualquier número de contenedores.
- Implemente los dos modelos en MiniZinc.

Ejercicio: El Periódico

El jefe de edición tiene que preparar el resumen de su periódico, que tiene 10 páginas. Tiene artículos organizados en varios temas: noticias internacionales, nacionales, locales, deportes y cultura. Estima que cada página dedicada a un tema dado puede interesar en promedio a cierto número de lectores.
El editor debe elegir los temas que se tratarán junto con el número de páginas para atraer el número máximo de lectores. Si decide incluir un tema determinado, debe tener en cuenta un número mínimo/máximo de páginas. La siguiente tabla muestra los rangos de páginas junto con el número promedio de lectores interesados en cada tema.

Topic	Min nb of pages	max nb of pages	potential readers (per page)
International	5	9	1500
National	4	7	2000
Local news	2	5	1000
Sport	2	4	1500
Culture	1	3	750

Ejercicio: El Periódico

Actividad en clase

- Proponga un modelo genérico para el problema.
- Generalice el problema. Utilice notación formal para proponer un modelo que soporte cualquier número de noticias.
- Implemente los dos modelos en MiniZinc.

Fin de la Presentación

Lecturas recomendadas:

Modelamiento básico en MiniZinc:
`https://www.minizinc.org/doc-2.2.3/en/modelling.html`

Modelos más complejos:
`https://www.minizinc.org/doc-2.2.3/en/modelling2.html`

Fin de la Presentación

¿Preguntas?