ESCUELA NACIONAL DE ESTUDIOS SUPERIORES
UNIDAD MORELIA

Universidad Nacional Autónoma de México

Escuela Nacional de Estudios Superiores
Unidad Morelia

Tecnologías para la Información en Ciencias

Inteligencia Artificial

Búsquedas con restricciones

Situación problema

Este trabajo se realizó con el apoyo del Programa UNAM-DGAPA-PAPIME dentro del Proyecto para la innovación y mejoramiento para la enseñanza, con clave PE110324, Diseño didáctico multi-seriado para potenciar habilidades de modelación matemática y reflexión crítica en el aprendizaje de la Inteligencia Artificial.

Presentación

El presente diseño didáctico lleva por título “El problema del cubo” y es una situación-problema que admite la representación con esquemas lógicos, apoyándose en lenguaje algebraico o combinatorio, para modelar un problema de satisfacción de restricciones. Dentro de la asignatura de Inteligencia Artificial, corresponde al contenido temático de *representación y búsqueda de soluciones*; en él, permite motivar la solución de problemas con búsqueda en restricciones.

Su elaboración incluye un documento y un recurso interactivo. El documento tiene, a su vez, dos secciones. La primera es un manual para estudiantes que incorpora un documento de presentación de la situación, una hoja de trabajo y un manual donde el cuaderno interactivo se ha editado para impresión; la segunda es un manual para docentes con una guía para el uso de material concreto, orientaciones didácticas para su implementación y una edición del cuaderno interactivo que permite anticipar las respuestas que dará el alumnado.

El diseño se desarrolló en un proceso metodológico iterativo de investigación basada en diseño, que permitió observar la incidencia del modelo en el aprendizaje de habilidades de modelación. En particular, este diseño motiva que el alumnado (1) genere modelos de una situación concreta, considerando distintas representaciones, que (2) sean apropiados para que un meta-modelo pueda encontrar la solución a partir del modelo propuesto; en este caso, el meta-modelo es de satisfacción de restricciones con consistencia de aristas.

ESCUELA NACIONAL DE ESTUDIOS SUPERIORES
UNIDAD MORELIA

Universidad Nacional Autónoma de México

Escuela Nacional de Estudios Superiores
Unidad Morelia

Tecnologías para la Información en Ciencias

Inteligencia Artificial

Búsquedas con restricciones

Manual para estudiantes

Este trabajo se realizó con el apoyo del Programa UNAM-DGAPA-PAPIME dentro del Proyecto para la innovación y mejoramiento para la enseñanza, con clave PE110324, Diseño didáctico multi-seriado para potenciar habilidades de modelación matemática y reflexión crítica en el aprendizaje de la Inteligencia Artificial.

El problema del cubo

Los vértices de un cubo se numeran del 1 al 8 de manera que el resultado de sumar los cuatro números de cada cara es el mismo para todas las caras. Se han colocado ya los números 1, 4 y 6 como se muestra en la figura 1.**¿Qué número va en el vértice marcado con x?**

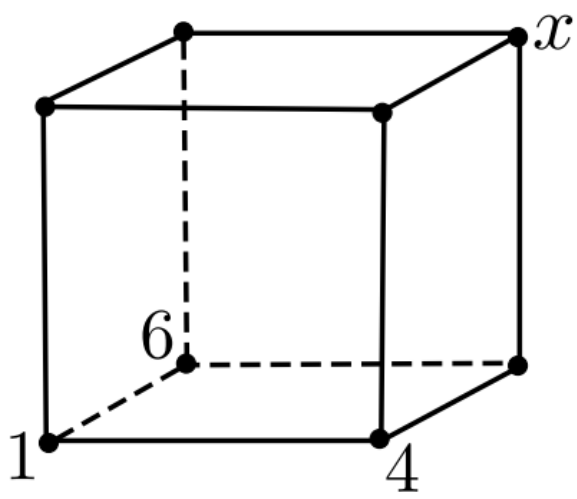

Figura 1: Cubo con vértices numerados.

1. Durante los próximos minutos, intenta resolverlo. Registra en la hoja anexa todas tus ideas para la solución. No borres ni taches los razonamientos que vas considerando, los retomaremos después.
2. Reúnete en equipo para conversar sobre las estrategias que se les han ocurrido. Intenten completar tantas soluciones como sea posible. ¡Recuerden registrar todo!
3. Desarrolla un programa que encuentre la solución del problema utilizando un algoritmo de búsqueda con esquemas lógicos.

Hoja de trabajo

Nombre:

Información útil para resolver el problema.

Registra aquí las estrategias que se te ocurren para resolverlo, desde el borrador hacia las ideas más acabadas.

Compara tus estrategias con las de tu equipo, ¿qué aprendiste?

¿Hubo alguna información del problema que no consideraste y era importante? ¿Cuál fue?

Escribe tu solución del problema.

El problema del cubo

Los vértices de un cubo se numeran del 1 al 8 de manera que el resultado de sumar los cuatro números de cada cara es el mismo para todas las caras. Se han colocado ya los números 1, 4 y 6 como se muestra en la figura 1. **¿Qué número va en el vértice marcado con x?**

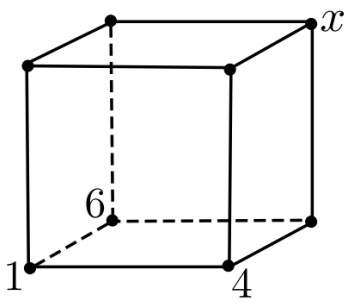

Figura 1: Cubo con vértices numerados.

Definición de variables

```python
from itertools import product
from collections import deque
```

```python
class Variable:
    def __init__(self, id_, domain):
        self.id = id_
        self.domain = domain


    def __str__(self):
        return f'{self.id} -> {self.domain}'
```

En este caso, nuestras variables serán los vértices del cubo, por lo tanto, primero debemos asignar nombre a cada uno de los vértices. Esto se hace para poder definirlos y diferenciarlos durante el algoritmo.

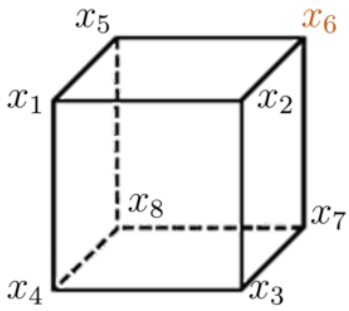

Figura 2: Cubo con vértices nombrados.

Definimos los valores para cada variable, considerando que algunos vértices ya tienen un valor asignado.

```python
values = [
    ['X1', [1,2,3,4,5,6,7,8]],
```

```
    ['X2', [1,2,3,4,5,6,7,8]],
    ['X3', [4]],
    ['X4', [1]],
    ['X5', [1,2,3,4,5,6,7,8]],
    ['X6', [1,2,3,4,5,6,7,8]],
    ['X7', [1,2,3,4,5,6,7,8]],
    ['X8', [6]]
]
```

```
variables = {}

for item in values:
    variables[item[0]] = Variable(*item)
```

```
for v in variables:
    print(variables[v])
```

```
>>> X1 -> [1, 2, 3, 4, 5, 6, 7, 8]
>>> X2 -> [1, 2, 3, 4, 5, 6, 7, 8]
>>> X3 -> [4]
>>> X4 -> [1]
>>> X5 -> [1, 2, 3, 4, 5, 6, 7, 8]
>>> X6 -> [1, 2, 3, 4, 5, 6, 7, 8]
>>> X7 -> [1, 2, 3, 4, 5, 6, 7, 8]
>>> X8 -> [6]
```

Definición de restricciones

Para este problema tenemos dos restricciones claramente establecidas en las instrucciones.

En primer lugar, se menciona que la suma de los vértices de las caras debe ser la misma. Esto se puede expresar de la siguiente forma:

$$\begin{aligned} x_1 + x_2 + x_3 + x_4 &= x_5 + x_6 + x_7 + x_8 \\ &= x_2 + x_3 + x_6 + x_7 \\ &= x_3 + x_4 + x_7 + x_8 \\ &= x_1 + x_4 + x_5 + x_8 \\ &= x_4 + x_3 + x_7 + x_8 \end{aligned}$$

Por otro lado, ningún vértice puede repetir la asignación de número, por lo tanto:

$$x_1 \neq x_2 \neq x_3 \neq x_4 \neq x_5 \neq x_6 \neq x_7 \neq x_8$$

```
def equal_sum(x1, x2, x3, x4, x5, x6, x7, x8):
    sums = [
        x1 + x2 + x3 + x4,
        x5 + x6 + x7 + x8,
        x2 + x3 + x6 + x7,
        x3 + x4 + x7 + x8,
        x1 + x4 + x5 + x8,
        x4 + x3 + x7 + x8
    ]
```

```
    return len(set(sums)) == 1


def all_different(x1, x2, x3, x4, x5, x6, x7, x8):
    return len(set([x1, x2, x3, x4, x5, x6, x7, x8])) == 8
```

```
class Constraint:
    def __init__(self, id_, rel_variables, func):
        self.id = id_
        self.related_variables = rel_variables
        self.function = func
```

```
constraints_values = [
    ['R01', ['X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X7', 'X8'], equal_sum],
    ['R02', ['X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X7', 'X8'], all_different]
]
```

```
constraints = {}

for item in constraints_values:
    constraints[item[0]] = Constraint(*item)
```

Reducción de dominio

Creamos todas las aristas (X, C) para todas las variables y restricciones.

```
edges = []

for key in constraints:
    for var in constraints[key].related_variables:
        if (var, key) not in edges:
            edges.append((var, key))
```

Realizamos el algoritmo $AC-3$ para la consistencia de arco.

```
edges_deque = deque(edges)

while(edges_deque):
    variable_name, constraint_name = edges_deque.popleft()

    for value in list(variables[variable_name].domain):
        valid = False

        for combination in product(*[variables[v].domain if v != variable_name else [value] for v in constraints[constraint_name].related_variables]):
            if constraints[constraint_name].function(*combination):
                valid = True
                break

        if not valid:
            variables[variable_name].domain.remove(value)

            related_cons = [constraint for variable, constraint in edges if variable == variable_name]
            affected_edges = [(v, c) for v, c in edges if c in related_cons and v != variable_name]
            edges_deque += deque([edge for edge in affected_edges if edge not in edges_deque])
```

Imprimimos los valores restantes después de la reducción de dominio.

```python
for v in variables:
    print(variables[v])
```

```
>>> X1 -> [8]
>>> X2 -> [5]
>>> X3 -> [4]
>>> X4 -> [1]
>>> X5 -> [3]
>>> X6 -> [2]
>>> X7 -> [7]
>>> X8 -> [6]
```

Búsqueda

En este caso, como cada variable solamente tiene un valor restante, únicamente tenemos una combinación por probar. Si esta combinación cumple con las restricciones, entonces la solución es la combinación. En caso contrario, el problema no tendría solución.

```python
equal_sum(8,5,4,1,3,2,7,6) and all_different(8,5,4,1,3,2,7,6)
```

```
>>> True
```

Por lo tanto, la solución es esa única asignación posible.

```python
for v in variables:
    print(f'{v}: {variables[v].domain[0]}')
```

```
>>> X1: 8
>>> X2: 5
>>> X3: 4
>>> X4: 1
>>> X5: 3
>>> X6: 2
>>> X7: 7
>>> X8: 6
```

Y respondiendo a la pregunta del problema, el vértice marcado con una x (que corresponde a nuestra variable x_6) debe tomar el valor 2 para cumplir con las restricciones del problema.

Ejercicio 1

Ahora consideremos números en las aristas en vez de los vértices, si tomamos números del 1 al 12 ¿Podemos asignar un número diferente a cada arista de manera que la suma de las aristas de todas las caras sumen lo mismo considerando que $H = 2$, $G = 11$ y $F = 12$? Encuentra todas las soluciones posibles que hay.

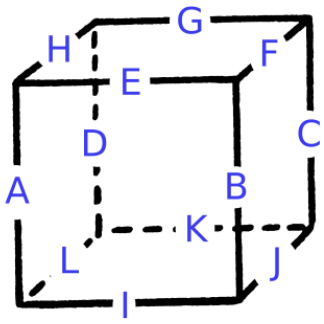

Figura 3: Cubo con aristas nombradas.

¡Coloca aquí tus respuestas!