Departamento de Ingeniería Industrial
Facultad de Ciencias Físicas y Matemáticas
Universidad de Chile
CC3101-2 Matemáticas Discretas

MODELADO Y RESOLUCIÓN DE PROBLEMAS USANDO LÓGICA

Trabajo Grupal

Integrantes: Mario Benavente
Felipe Gómez
Leonardo Rikhardsson
Profesor: Alejandro Hevia
Federico Olmedo
Nelson Baloian T.
Profesor Auxiliar: Lucas Torrealba A.

Fecha de entrega: 30 de Noviembre de 2023
Santiago, Chile

Índice de Contenidos

1. P1

Modele la situación como un problema de satisfacibilidad. Esto es, defina un conjunto de variables proposicionales (las que necesite) y entregue una fórmula de la lógica proposicional ϕ sobre dichas variables que se haga verdadera cuando se cumplan las reglas para que cada computador tenga asociado un mega-servidor, respetando las restricciones.

Como grupo decidimos definir el problema de la siguiente manera:

[Restricción 1: Cada computador debe estar asociado a un único mega-servidor]

La forma en que expresamos la restricción se basa en la idea de «qué pasaría si, para algún computador i arbitrario en nuestro conjunto C, tenemos que está conectado a algún mega-servidor j en S?».
Nuestro razonar fue el siguiente:

1) Estando conectado a un servidor j en S, i no podría estar conectado a ningún otro servidor.

2) Representar que no puede estar conectado a otro servidor en S dado que ya se encuentra conectado a j puede hacerse mediante un «implica».

3) Notemos que, por lo anterior, tendríamos la siguiente idea: para todo computador i en C, y todo mega-servidor j en S, si i está conectado a j, entonces i no puede estar conectado a ningún otro mega-servidor. De esta forma, realizamos una serie de conjunciones para i entre 1 y 100 (equivalente al dominio de C), donde dentro de la misma realizamos una segunda serie de conjunciones para dos mega-servidores j_1, j_2 distintos entre sí, y entre 1 y K (dominio de S), tal que si i se conecta al mega-servidor j_1, no puede estar conectado también al mega-servidor j_2.

[Restricción 2: Cada mega-servidor tiene al menos un computador conectado a éste]

Acá el razonamiento es más simple, pues buscamos únicamente definir qué es un mega-servidor, y la definición deriva directamente del enunciado, diciéndonos que no debemos preocuparnos por los recursos. De esto deducimos que pueden estar conectados varios computadores. Luego, necesariamente tiene que tener al menos un computador conectado, pues añadir mega-servidores que no tengan computadores es ineficiente.

De esta forma, nos interesa que quede planteado como una serie de conjunciones para cada servidor j en S y, para cada servidor j, evaluar una serie de disyunciones con respecto a todos los computadores i en C, con respecto a la condición de que un computador arbitrario esté conectado a S: así, basta con que al menos un computador esté conectado a cada mega-servidor para cumplir con esta restricción

[Restricción 3: Una lista negra es válida si y sólo si posee únicamente 3 computadores]

En este caso, nos importa tomar 4 elementos arbitrarios de C, los cuales llamaremos i, r_1, r_2 y r_3. De esta forma, supondremos que r_1, r_2 y r_3 son los elementos que corresponden a la lista negra de i. Para que esto funcione, deberemos establecer ciertas condiciones:

1) r_1, r_2 y r_3 son distintos de i, y distintos entre sí.

2) Como i, r_1, r_2 y r_3 son arbitrarios en C, necesitamos que puedan tomar valores entre 1 y 100 o, en otras palabras, cualquier valor del dominio de C.

3) Para que cuente como una lista negra válida, necesitamos que 3 y solo 3 computadores existan en esta lista.

Con esto, podríamos modelar esta restricción como una serie de conjunciones para i, r_1, r_2 y r_3 entre 1 y 100, distintos entre sí, donde se cumple que r_1, r_2 y r_3 pertenecen a la lista negra de i.

[Restricción 4: Un computador no puede compartir mega-servidor con ninguno de los 3 computadores en su lista]

Siguiendo un razonamiento parecido a la Restricción 3, comenzamos indicando una serie de conjunciones para i, r_1, r_2 y r_3 entre 1 y 100, distintos entre sí, donde nuevamente r_1, r_2 y r_3 pertenecen a la lista negra de i, además de que, con una serie de conjunciones para todo servidor j en S, el computador i esté conectado a éste.

Lo importante es identificar qué implica que i esté conectado a j, con r_1, r_2 y r_3 en su lista negra: es directo que ninguno de estos 3 podrá conectar con j. Así, la restricción queda como dos series de conjunciones, una que determina i, r_1, r_2 y r_3 entre 1 y 100, distintos entre sí, y la otra determina j entre 1 y K, para la cual se nos cumple que r_1, r_2 y r_3 forman parte de la lista negra de i, además de que i esté conectado al mega-servidor j. Esto implica que ni r_1, r_2 o r_3 pueden compartir servidor con i. Notamos que ver que negar que algo se cumpla para al menos un elemento, equivale a decir que nunca se cumple, por lo que el lado derecho de la implicancia nos quedaría como la negación de que uno de los 3 elementos de la lista comparte servidor con i.

Definiciones:

- Conjunto de los computadores: $C = \{1, \ldots, 100\}$
- Conjunto de los mega-servidores : $S = \{1, \ldots, k\}$
- $N = \{X_{i,j} \mid i \in C, j \in S\}$
- $\sigma(X_{i,j}) : N \to \{0, 1\}$
- $\sigma(X_{i,j}) = \begin{cases} 1 & \text{si } i \text{ está conectado a } j \\ 0 & \text{si no} \end{cases}$
- Lista de incompatibilidad : $L = \{Y_{i,r} \mid i, r \in C\}$
- $\sigma(Y_{i,r}) : L \to \{0, 1\}$
- $\sigma(Y_{i,r}) = \begin{cases} 1 & \text{si } r \text{ está en la lista negra de } i \\ 0 & \text{si no} \end{cases}$

Restricciones:

1. **Cada computador debe estar conectado a un único mega-servidor:**

$$\bigwedge_{i=1}^{100} \left(\bigwedge_{1 \leq j_1, j_2 \leq K} \left(X_{i,j_1} \Rightarrow \neg \left(X_{i,j_2} \right) \right) \right)$$

2. **Cada mega-servidor tiene al menos un computador conectado a éste:**

$$\bigwedge_{j=1}^{K} \left(\bigvee_{i=1}^{100} X_{i,j} \right)$$

3. **Una "lista negra" es válida si y sólo si posee únicamente 3 computadores:**

$$\bigwedge_{1 \leq i, r_1, r_2, r_3 \leq 100} \left(Y_{i,r_1} \wedge Y_{i,r_2} \wedge Y_{i,r_3} \right)$$

4. **Un computador no puede compartir mega-servidor con ninguno de los 3 computadores en su lista:**

$$\bigwedge_{1 \leq i, r_1, r_2, r_3 \leq 100} \left(\bigwedge_{j=1}^{K} \left(\left(Y_{i,r_1} \wedge Y_{i,r_2} \wedge Y_{i,r_3} \wedge X_{i,j} \right) \Rightarrow \neg \left(X_{r_1,j} \vee X_{r_2,j} \vee X_{r_3,j} \right) \right) \right)$$

$$\phi = \left(\bigwedge_{i=1}^{100} \left(\bigwedge_{1 \leq j_1, j_2 \leq K} \left(X_{i,j_1} \Rightarrow \neg \left(X_{i,j_2} \right) \right) \right) \right) \wedge$$
$$\left(\bigwedge_{j=1}^{K} \left(\bigvee_{i=1}^{100} X_{i,j} \right) \right) \wedge$$
$$\left(\bigwedge_{1 \leq i, r_1, r_2, r_3 \leq 100} \left(Y_{i,r_1} \wedge Y_{i,r_2} \wedge Y_{i,r_3} \right) \right) \wedge$$
$$\left(\bigwedge_{1 \leq i, r_1, r_2, r_3 \leq 100} \left(\bigwedge_{j=1}^{K} \left(\left(Y_{i,r_1} \wedge Y_{i,r_2} \wedge Y_{i,r_3} \wedge X_{i,j} \right) \Rightarrow \neg \left(X_{r_1,j} \vee X_{r_2,j} \vee X_{r_3,j} \right) \right) \right) \right)$$

2. P2

Eliminación de implicaciones:

$$\begin{aligned}\phi = &\left(\bigwedge_{i=1}^{100}\left(\bigwedge_{1\leq j_1,j_2\leq K}(\neg X_{i,j_1}\vee\neg X_{i,j_2})\right)\right)\wedge\\ &\left(\bigwedge_{j=1}^{K}\left(\bigvee_{i=1}^{100}X_{i,j}\right)\right)\wedge\\ &\left(\bigwedge_{1\leq i,r_1,r_2,r_3\leq 100}(Y_{i,r_1}\wedge Y_{i,r_2}\wedge Y_{i,r_3})\right)\wedge\\ &\left(\bigwedge_{1\leq i,r_1,r_2,r_3\leq 100}\left(\bigwedge_{j=1}^{K}\left(\neg\left(Y_{i,r_1}\wedge Y_{i,r_2}\wedge Y_{i,r_3}\wedge X_{i,j}\right)\vee\neg\left(X_{r_1,j}\vee X_{r_2,j}\vee X_{r_3,j}\right)\right)\right)\right)\end{aligned}$$

Ley de De Morgan para conjunción:

$$\begin{aligned}\phi = &\left(\bigwedge_{i=1}^{100}\left(\bigwedge_{1\leq j_1,j_2\leq K}(\neg X_{i,j_1}\vee\neg X_{i,j_2})\right)\right)\wedge\\ &\left(\bigwedge_{j=1}^{K}\left(\bigvee_{i=1}^{100}X_{i,j}\right)\right)\wedge\\ &\left(\bigwedge_{1\leq i,r_1,r_2,r_3\leq 100}(Y_{i,r_1}\wedge Y_{i,r_2}\wedge Y_{i,r_3})\right)\wedge\\ &\left(\bigwedge_{1\leq i,r_1,r_2,r_3\leq 100}\left(\bigwedge_{j=1}^{K}\left((\neg Y_{i,r_1}\vee\neg Y_{i,r_2}\vee\neg Y_{i,r_3}\vee\neg X_{i,j})\vee\neg\left(X_{r_1,j}\vee X_{r_2,j}\vee X_{r_3,j}\right)\right)\right)\right)\end{aligned}$$

Ley de De Morgan para disyunción:

$$\begin{aligned}\phi = &\left(\bigwedge_{i=1}^{100}\left(\bigwedge_{1\leq j_1,j_2\leq K}(\neg X_{i,j_1}\vee\neg X_{i,j_2})\right)\right)\wedge\\ &\left(\bigwedge_{j=1}^{K}\left(\bigvee_{i=1}^{100}X_{i,j}\right)\right)\wedge\\ &\left(\bigwedge_{1\leq i,r_1,r_2,r_3\leq 100}(Y_{i,r_1}\wedge Y_{i,r_2}\wedge Y_{i,r_3})\right)\wedge\\ &\left(\bigwedge_{1\leq i,r_1,r_2,r_3\leq 100}\left(\bigwedge_{j=1}^{K}\left((\neg Y_{i,r_1}\vee\neg Y_{i,r_2}\vee\neg Y_{i,r_3}\vee\neg X_{i,j})\vee(\neg X_{r_1,j}\wedge\neg X_{r_2,j}\wedge\neg X_{r_3,j})\right)\right)\right)\end{aligned}$$

3. P3

```
from pysat.solvers import Glucose3
from itertools import combinations
import glob
import os

def leer_archivo(archivo):
    with open(archivo, 'r') as file:
        lineas = file.readlines()

    servidores = [list(map(int, linea.strip().split(','))) for linea in lineas[1:4]]
    computadoras = [list(map(int, linea.strip().split(','))) for linea in lineas[4:14]]
   incompatibilidades = {i + 1: list(map(int, linea.strip().split(',')[1:])) for i, linea in enumerate(lineas[14:])}

    return servidores, computadoras, incompatibilidades

def agregar_restriccion_1(solver, computadoras):
    for comp in computadoras:
        solver.add_clause(comp[1:])

def agregar_restriccion_2(solver, mega_servidores):
    for mega_servidor in mega_servidores:
        solver.add_clause(mega_servidor)

def agregar_restriccion_3(solver, incompatibilidades):
    for comp, incompatibles in incompatibilidades.items():
        for servidor in incompatibles:
            solver.add_clause([-comp, -servidor])

def resolver_problema(servidores, computadoras, incompatibilidades):
    solver = Glucose3()

    # Restricción 1
    agregar_restriccion_1(solver, computadoras)

    # Restricción 2
    agregar_restriccion_2(solver, servidores)

    # Restricción 3
    agregar_restriccion_3(solver, incompatibilidades)

    resultado = solver.solve()

    if resultado:
      modelo = [i for i in range(1, len(computadoras) * len(servidores) + 1) if solver.get_model()[i -
```

```
1] > 0]
        print("Solución encontrada:")
        print(modelo)
        return modelo
    else:
        print("No se encontró solución.")
        return None


# Ejemplo de uso
# La idea es colocar los archivos en una carpeta y te lee todo
source = r"C:\Users\..."
type_of_file = "*.txt"
filenames = glob.glob(os.path.join(source, type_of_file))

for filename in filenames:
    servidores, computadoras, incompatibilidades = leer_archivo(filename)
    result = resolver_problema(servidores, computadoras, incompatibilidades)

    if result:
        print(f"Para el archivo: {os.path.split(filename)[1]} - Asignación de servidores: {result}")
    else:
     print(f"Para el archivo: {os.path.split(filename)[1]} - No se encontró una asignación válida.")
```

4. P4

Por intuición, notamos que se necesitan al menos n=4 computadores y K=4 mega-servidores, puesto que cada computador posee una lista con 3 computadores no compatibles.
Segundo, para facilitar la búsqueda de esta cota, asumimos que las listas son de forma cíclica. Esto es, para un computador i, su lista consiste de los computadores i+1, i+2 e i+3, y da la vuelta a los índices si es que i+1, i+2 y/o i+3 es/son mayor(es) que 100.

Con esto, examinemos los casos $n = 4$, $n = 5$, $n = 6$ y $n = 7$.

Para $n = 4$, tendríamos la siguiente configuración:
1,2,3,4
2,3,4,1
3,4,1,2
4,1,2,3

Es directo notar que necesitamos un mega-servidor por computador, dándonos un total de 4 servidores.

Para $n = 5$:
1,2,3,4
2,3,4,5
3,4,5,1
4,5,1,2
5,1,2,3

En este caso, vemos que, dentro de la lista negra del computador 1 no se encuentra el computador 5, pero éste sí aparece en la lista del quinto computador, por lo que necesitaríamos 5 servidores.

Para $n = 6$, es similar al caso anterior, donde necesitaríamos 6 servidores por un razonamiento muy similar:
1,2,3,4
2,3,4,5
3,4,5,6
4,5,6,1
5,6,1,2
6,1,2,3

Lo mismo si $n = 7$, con la configuración:
1,2,3,4
2,3,4,5
3,4,5,6
4,5,6,7
5,6,7,1
6,7,1,2
7,1,2,3

Notemos que es relativamente trivial, en conjunto con el razonamiento presentado en la P1, que para 4, 5 y 6 computadores, podemos igualmente solucionar el problema usando 7 servidores. Lo interesante aparece para 8 computadores, donde la distribución quedaría como:
1,2,3,4
2,3,4,5
3,4,5,6
4,5,6,7
5,6,7,8
6,7,8,1
7,8,1,2
8,1,2,3

A diferencia de los casos anteriores, esta vez podemos hacer que el primer y quinto computador compartan servidor, y lo mismo para el segundo y el sexto, tercero y quinto, y cuarto con el octavo, por lo que solo necesitamos 4 servidores.

De esta forma, plantearemos un problema de inducción, donde queremos demostrar que para cualquier cantidad de computadores n mayor o igual que 4, podemos resolver el problema con 7 servidores. Para esto, tomaremos como Casos Base $n = 4$, $n = 5$, $n = 6$ y $n = 7$, los cuales acabamos de demostrar: así, podemos seguir con el procedimiento.

Hipótesis de inducción:
Supongamos que la afirmación es verdadera para n computadores, es decir, se puede asignar $K_0 = 7$ servidores para n computadoras sin incompatibilidades.

Paso inductivo:
Consideremos ahora $n + 1$. Queremos demostrar que también podemos asignar $K_0 = 7$ servidores para n + 1 computadores.

Por la naturaleza del problema, es posible visualizar la situación como añadir un i-ésimo computador a un problema ya satisfecho para n computadores.

Por Principio del Palomar, si removemos este i-ésimo computador, sabemos que el problema está satisfecho, donde tendríamos al menos un servidor que contiene al menos $\lceil n/7 \rceil$ computadores. Seguido de esto, tenemos dos casos:

i) $\lceil n/7 \rceil = \lceil i/7 \rceil$: si añadir el i-ésimo elemento al problema no altera el Principio del Palomar, entonces el problema sigue siendo satisfactible, pues tendríamos algún servidor donde ubicar el computador i sin romper restricciones.

ii) $\lceil n/7 \rceil = \lceil i/7 \rceil + 1$: contrario a lo anterior, añadir este i-ésimo elemento al problema SI altera el Principio del Palomar, diciéndonos que ahora hay al menos un mega-servidor donde están conectados $\lceil n/7 \rceil + 1$ servidores. Siendo ese el caso, ahora simplemente bastaría con alocar este i-ésimo elemento en cualquiera de los servidores que cumpla con nuestras restricciones, pues todos, antes de la adición del último elemento, tienen la misma cantidad $\lceil n/7 \rceil$ de elementos.

Por lo tanto, hemos llegado a que el mínimo valor de K para que el problema sea siempre satisfactible es 7.