Moderando el conflicto interno de opiniones en una red social

Calderón Prieto Brandon (2125974) Cely Archila Juleipssy Daianne ()
Juan Fonseca ()

31 de marzo de 2025

Índice general

0.1. Programación dinámica

0.1.1. Caracterización de la subestructura óptima

Definimos la notación $\mathrm{IC}(i, r)$ como el problema de minimizar el conflicto interno utilizando los primeros i grupos de agentes con un esfuerzo disponible r. La notación $\mathrm{modIC}(i, r)$ denota la secuencia de decisiones óptimas (es decir, el número de agentes a moderar en cada grupo) para los grupos i hasta n utilizando r de esfuerzo, donde la secuencia vacía $\langle\rangle$ indica que no se realizan más moderaciones.

$$\mathrm{modIC}(i, r) = \begin{cases} \langle\rangle & i = n + 1 \\ k_i :: \mathrm{modIC}(i + 1, r - \mathrm{e\_effort}(i, k_i)) & i \neq n \end{cases} \tag{1}$$

$$\mathrm{IC}(i, R_{\mathrm{máx}}) = \mathrm{modIC}(1, R_{\mathrm{máx}}) \tag{2}$$

0.1.2. Valor recursivo de la solución óptima

- $\mathrm{e\_effort}(i, k)$: es el esfuerzo requerido para **moderar** k agentes del grupo i ($\lceil |o_{i,1} - o_{i,2}| \cdot r_i \cdot k \rceil$).
- $\mathrm{conflict}(i, k)$: es la contribución al conflicto interno del grupo i luego de **moderar** k agentes ($(n_i - k) * (o_{i,1} - o_{i,2})^2$).

Normalización en cada paso

$$\mathrm{recIC}_1(i, r) = \begin{cases} 0 & i = 0 \\ \mathrm{mín}_{0 \leq k_i \leq n_i} \left\{ \frac{(i-1) \cdot \mathrm{recIC}_1(i-1, r - \mathrm{e\_effort}(i,k_i)) + \mathrm{conflict}(i,k_i)}{i} \right\} & i > 0 \end{cases} \tag{3}$$

$$\mathrm{valueIC}(n, R_{\mathrm{máx}}) = \mathrm{recIC}_1(n, R_{\mathrm{máx}}) \tag{4}$$

Acumulación total y normalización final

$$\mathrm{recIC}_2(i, r) = \begin{cases} 0 & i = 0 \\ \mathrm{mín}_{0 \leq k_i \leq n_i} \{\mathrm{recIC}_2(i - 1, r - \mathrm{e\_effort}(i, k_i)) + \mathrm{conflict}(i, k_i)\} & i > 0 \end{cases} \tag{5}$$

$$\mathrm{valueIC}(n, R_{\mathrm{máx}}) = \frac{\mathrm{recIC}_2(n, R_{\mathrm{máx}})}{n} \tag{6}$$

Justificación equivalencia

Sea $F(i, r)$ la suma total del conflicto acumulado en los i primeros grupos.
Ambas formulaciones calculan el mismo valor $\mathrm{valueIC}(n, R_{\mathrm{máx}}) = \frac{1}{n} \sum_{i=0}^{n-1} \mathrm{conflict}(i, k_i)$.

- Caso base: para $i = 0$, ambas definiciones retornan 0.
- Paso inductivo: supongamos que para $i - 1$, $\mathrm{recIC}_1(i, r') = \frac{F(i-1,r')}{i-1}$ y $\mathrm{recIC}_2(i, r') = F(i - 1, r')$.

- recIC_1.

$$\text{recIC}_1(i, r) = \frac{(i-1) \cdot \text{recIC}_1(i-1, r - \text{e\_effort}(i, k_i)) + \text{conflict}(i, k_i)}{i} \tag{7}$$

$$= \frac{F(i-1, r') + \text{conflict}(i, k_i)}{i} \tag{8}$$

- recIC_2.

$$\text{recIC}_2(i, r) = \text{recIC}_2(i-1, r - \text{e\_effort}(i, k_i)) + \text{conflict}(i, k_i) \tag{9}$$

$$= F(i-1, r') + \text{conflict}(i, k_i) \tag{10}$$

Al final $\text{valueIC}(n, R_{\text{máx}}) = \frac{\text{recIC}_2(n, R_{\text{máx}})}{n} = \frac{F(n, R_{\text{máx}})}{n}$, que coincide con $\text{recIC}_1(n, R_{\text{máx}})$.

Lo cual tiene sentido, minimizar para $\frac{F(i-1,r') + \text{conflict}(i,k_i)}{i}$ es equivalente a minimizar $F(i-1, r') + \text{conflict}(i, k_i)$ porque dividir por una constante positiva (i) no cambia la elección del k_i que da el mínimo valor.

0.1.3. Algoritmo para calcular el costo de una solución óptima

Para calcular el costo óptimo mediante programación dinámica, utilizamos un bottom-up que construye sistemáticamente la solución a partir de los casos base.

Definimos una matriz *storage* donde $storage[i][r]$ representa el mínimo conflicto interno posible considerando los primeros i grupos con un esfuerzo disponible r. El algoritmo completa esta matriz iterativamente:

Algorithm 1 cálculo del costo óptimo mediante programación dinámica (bottom-up)

$n \leftarrow$ número de grupos
$r_{max} \leftarrow$ esfuerzo máximo disponible
inicializar $storage[0][r] = 0$ para todo $r \in [0, r_{\text{máx}}]$
inicializar $storage[i][r] = \infty$ para todo $i \in [1, n]$, $r \in [0, r_{\text{máx}}]$
for $i = 1$ hasta n **do**
    obtener grupo actual: $a_{i-1} = \langle n_i, o_{i,1}, o_{i,2}, r_i \rangle$
    $conflict\_per\_agent \leftarrow (o_{i,1} - o_{i,2})^2$
    $effort\_per\_agent \leftarrow |o_{i,1} - o_{i,2}| \cdot r_i$
    for $r = 0$ hasta $r_{\text{máx}}$ **do**
        for $k = 0$ hasta n_i **do**
            $required\_effort \leftarrow \lceil effort\_per\_agent \cdot k \rceil$
            if $required\_effort \leq r$ **then**
                $remaining\_conflict \leftarrow (n_i - k) \cdot conflict\_per\_agent$
                $total\_conflict \leftarrow storage[i-1][r - required\_effort] + remaining\_conflict$
                if $total\_conflict < storage[i][r]$ **then**
                $storage[i][r] \leftarrow total\_conflict$
                end if
            end if
        end for
    end for
end for
return $storage[n][r_{\text{máx}}]$

El valor $storage[n][r_{\text{máx}}]$ representa la suma total del conflicto mínimo posible. Para obtener el conflicto interno normalizado según la definición del problema, debemos dividir este valor por n:

$$\text{valueIC}(n, R_{\text{máx}}) = \frac{storage[n][r_{\text{máx}}]}{n} \tag{11}$$

0.1.4. Algoritmo para calcular una solución óptima

Mientras que la sección anterior nos permite calcular el valor mínimo de conflicto interno, también necesitamos determinar la estrategia óptima que produce dicho valor. Para reconstruir esta estrategia, utilizamos una matriz adicional durante el proceso que registra las decisiones óptimas.

Definimos una matriz *decisions* donde $decisions[i][r]$ representa el número óptimo de agentes a moderar del grupo i cuando se dispone de un esfuerzo r. Esta matriz se completa durante el mismo proceso iterativo en que calculamos el costo mínimo.

Algorithm 2 registro de decisiones (extensión del algoritmo anterior)

inicializar $decisions[i][r] = 0$ para todo $i \in [0, n]$, $r \in [0, r_{\text{máx}}]$
...
if $total\_conflict < storage[i][r]$ **then**
    $storage[i][r] \leftarrow total\_conflict$
    $decisions[i][r] \leftarrow k$ ▷ registramos la decisión óptima
end if
...

Una vez completadas las matrices *storage* y *decisions*, podemos reconstruir la estrategia óptima mediante un proceso de retroceso:

Algorithm 3 reconstrucción de la estrategia óptima

1: $n \leftarrow$ número de grupos
2: $r_{\text{máx}} \leftarrow$ esfuerzo máximo disponible
3: inicializar *optimal_strategy* como un arreglo de n ceros
4: $remaining\_effort \leftarrow r_{\text{máx}}$
5: **for** $i = n$ hasta 1 (en orden descendente) **do**
6: $k \leftarrow decisions[i][remaining\_effort]$ ▷ número óptimo de agentes a moderar
7: $optimal\_strategy[i-1] \leftarrow k$
8: obtener grupo actual: $a_{i-1} = \langle n_i, o_{i,1}, o_{i,2}, r_i \rangle$
9: $effort\_per\_agent \leftarrow |o_{i,1} - o_{i,2}| \cdot r_i$
10: $required\_effort \leftarrow \lceil effort\_per\_agent \cdot k \rceil$
11: $remaining\_effort \leftarrow remaining\_effort - required\_effort$
12: **end for**
13: **return** *optimal_strategy*

El resultado *optimal_strategy* es un arreglo donde la posición i indica cuántos agentes deben ser moderados del grupo i para lograr el mínimo conflicto interno posible. Esta estrategia satisface la restricción de que su esfuerzo total no excede $R_{\text{máx}}$.

La estrategia resultante $E = \langle e_0, e_1, \ldots, e_{n-1} \rangle$ representa nuestra solución final al problema, donde cada e_i indica exactamente cuántos agentes del grupo i deben ser moderados para minimizar el conflicto interno de la red social.

0.1.5. Complejidad

Rellenando la matriz *storage* **y** *decisions*

Algorithm 4 estructura del cálculo del costo óptimo mediante programación dinámica (bottom-up)

```
...
for i = 1 hasta n do
    ...
    for r = 0 hasta r_máx do
        for k = 0 hasta n_i do
            ...
        end for
    end for
end for
return storage[n][r_máx]
```

- Temporal: $O(n \cdot R_{\text{máx}} \cdot \text{máx}(n_i))$, donde n es el número de grupos, $R_{\text{máx}}$ es el esfuerzo máximo disponible y $\text{máx}(n_i)$ es el número máximo de agentes en cualquier grupo.
- Espacial: $O(n \cdot R_{\text{máx}})$, donde n es el número de grupos, $R_{\text{máx}}$ es el esfuerzo máximo disponible.

  Esto es válido porque en el ciclo interior, las operaciones que se hacen son asignaciones, es decir, tienen complejidad $O(1)$.

Recuperando la solución de la matriz *decisions*

Algorithm 5 estructura de la reconstrucción de la estrategia óptima

```
...
inicializar optimal_strategy como un arreglo de n ceros
...
for i = n hasta 1 (en orden descendente) do
    ...
end for
return optimal_strategy
```

- Temporal: $O(n)$, donde n es el número de grupos.
- Espacial: $O(n)$, donde n es el número de grupos.

  Esto es válido porque en el ciclo interior, las operaciones que se hacen son asignaciones, es decir, tienen complejidad $O(1)$.

Por lo tanto, la complejidad temporal del algoritmo es $O(n \cdot R_{\text{máx}} \cdot \text{máx}(n_i) + n)$ y la espacial es $O(n \cdot R_{\text{máx}} + n)$.