

Y el protocolo propuesto es una solución de la tarea pues respeta a Δ y es un mapeo simplicial.

- (b) Demuestra que con un protocolo de una ronda no se puede resolver el problema.

Observe que Δ manda a $B0$ en $B2$ y a $A1$ en $A1$, y que en el complejo dado el camino entre esos dos vértices que está en el mapeo de la arista $\{B0, A1\}$ es de longitud 5, es decir, en un paso tenemos que expandir una arista en un camino de longitud 5, pero como vimos en clase, cada paso cuando más trisecta la arista, por lo que no es posible resolver la tarea en un paso ■.

Problema 2 Sea δ un mapeo simplicial, que nos lleva de una gráfica G a una gráfica H , δ mapea los vértices de G sobre todos los vértices de H .

- Demuestra que si G es conexo entonces H es conexo.

Sean κ_1, κ_2 vértices en H , como δ es suprayectiva sean μ_1, μ_2 vértices en G tales que $\delta(\kappa_1) = \mu_1$ y $\delta(\kappa_2) = \mu_2$, si G es conexo, existe un camino en G que va de μ_1, μ_2 , considere la aplicación de δ sobre ese camino, cada arista va bien a una arista o a un vértice, pero su mapeo es también un camino, por lo que H es conexa también ■.

- Demuestra que si δ preserva colores entonces es rígido, es decir, mapea vértices diferentes a vértices diferentes.

Aquí hay hacer dos observaciones: Primero estamos suponiendo gráficas bicromáticas, pues si tenemos una gráfica monocromática, podríamos enviar toda la gráfica a un vértice y preservaría colores sin ser rígida. Segundo, solo funciona para una gráfica conexa, si no fuera conexa, digamos que tiene $n \geq 2$ componentes idénticos y los envía a una gráfica idéntica a uno de esos componentes, preserva colores, pero no es rígida.

Si δ preserva colores en una gráfica bicromática conexa, supongamos que envía dos vértices a uno mismo, ambos vértices deben de ser del mismo color pues preserva colores, y por ser conexa hay un camino que une a esos dos vértices y por ser bicromática cada arista une a vértices de distintos colores, por lo que hay por lo menos un vértice del color distinto en el camino, pero como envío ambos extremos del camino a un solo vértice, debí enviar todo el camino a esos vértices, pero eso no es posible preservando colores pues hay en el camino un vértice de otro color ■.

Problema 3 Considera protocolos de una sola iteración, de memoria compartida libre de espera, y un complejo de entrada I de N vértices y M aristas, define una tarea I, O, Δ sobre este complejo que se pueda resolver, suponer que $\Delta(v)$ siempre es un sólo vértice, tal que

- El número de soluciones posibles (protocolos) sea lo menor posible. Obviamente si la tarea se puede resolver hay al menos una solución, luego considere el complejo de salida igual a una arista, este complejo admite claramente solo una solución pues es aquella que manda los vértices A al vértice A y los vértices B al vértice B y es la única solución sobre esta salida, por lo que la cota inferior es 1.
- Que sea lo mayor posible. En ambos casos, explica cual es ese número. Ahora considere una gráfica que cada vértice lo manda a un vértice diferente y la gráfica bipartita completa que los unen, y dejemos que Δ de cada arista sea la gráfica completa, entonces, para cada arista m en I podemos elegir cualquiera de las $a \times b$ aristas de la gráfica de salida, donde a es el número de vértices del primer color y b el número de vértices del segundo, por lo que habrá $a! \times b!$ soluciones y no podría haber más, puesto que aquí cada orden de los vértices da una solución diferente, si hubiera una gráfica que diera más soluciones, tendría que dar para algún orden más de una solución ■

Problema 4 Presenta una caracterización de las tareas que tienen solución mediante este tipo de protocolos, protocolos del ejercicio 3, y un algoritmo (secuencial) que tome como entrada una tarea, y como salida diga si tiene o no solución. Cuál es la complejidad de tu algoritmo?

Respuesta Como hemos visto, todos los protocolos hacen lo siguiente: trisectan en cada paso las aristas y luego en el mapeo, pueden “doblarlas” y compactarlas. Así que se puede resolver una tarea si hay una forma de trisectar las aristas y ponerlas en el complejo de salida de tal forma que respeten Δ .

El algoritmo sería el siguiente:

```
def tiene_solucion (Delta, aristas_entrada):
    foreach (v,w) in aristas_entrada:
        for v_i in \Delta(v):
            for w_i in \Delta(w):
                if BFS(v_i, w_i) union Delta( (v,w) ) is empty:
                    return false
            else:
                continue
    return true
```

Esencialmente, se ve si por cada arista, el simplejo al que se envía en Δ es conexo. La complejidad es del orden del número de aristas de entrada por el número de aristas de su imagen en Δ