

# MULTICAST ROUTING PARA REDES 6G

Agustín Rebechi



# INTRODUCTION

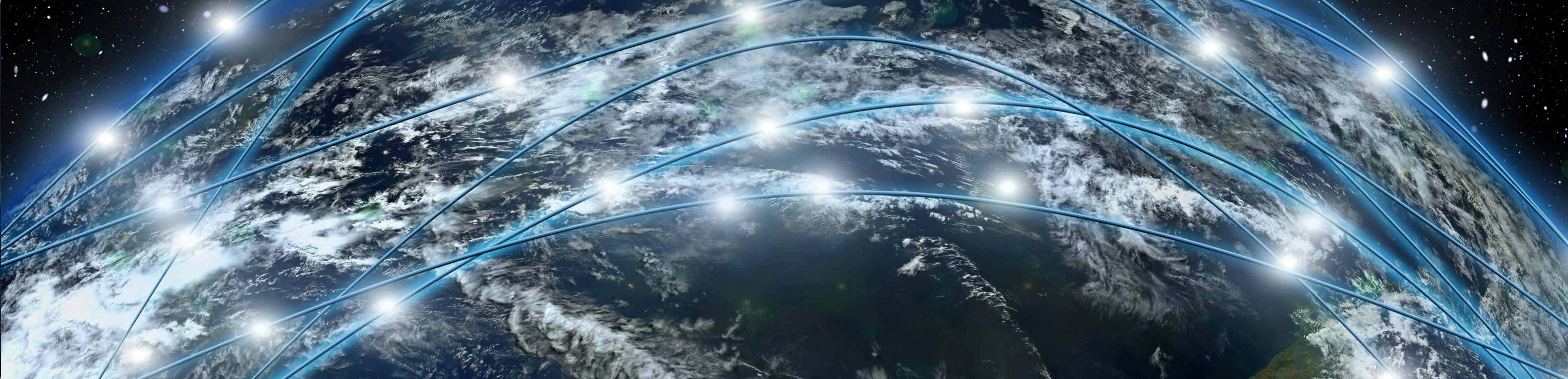
## Redes 6G

La aparición de las redes 6G está impulsando un cambio de paradigma en los servicios multimedia, una tecnología que permite velocidades de hasta 1 Tbps. De la mano de un crecimiento explosivo del tráfico multimedia global, y se preveé que solo el mercado del streaming crezca de \$87.3 Billon a \$3.78 Trillion



## Principal desafío

Estas transformaciones de servicios imponen retos sin precedentes a la infraestructura de red. Soportar demandas heterogéneas de QoS minimizando consumo de energía en topologías cada vez más complejas. Entregar datos frescos y sincronizados a múltiples destinos simultáneamente es un requisito fundamental.



## LIMITACIONES

01

Algoritmos clasicos como Dijkstra, Bellman-ford aunque eficientes para punto a punto, en multicast generan flujo redundante y asumen demandas homogeneas. Esto deriva en un uso suboptimo de recursos

02

Si se formula un Arbol de Steiner, la complejidad del mismo se convierte NP-hard lo cual lo hace computacionalmente inviable, inadecuados para entornos con limitaciones de latencia, no considera heterogeneidad de demandas

# On-Demand Steiner Tree

## Método exacto con Programación Dinámica

En este paper se aborda el Minimum Flow Problem (MFP) con demandas de salidas heterogéneas, para superar las limitaciones propone un On-Demand Steiner Tree mejorado con programación dinámica. La solución divide la red subgrafos homogéneos por demanda QoS, y luego los conecta a través de uniones de Steiner

Output-Constrained Minimum Flow Problem (OCMFP) generaliza el clásico árbol de Steiner al considerar demandas heterogéneas. Para resolver el OCMFP eficientemente, el OST emplea programación dinámica en dos etapas.

Divide el conjunto de destinos en subconjuntos y calcula recursivamente el costo mínimo de transmisión, considerando reutilización de rutas



# IMPLEMENTACIÓN CON 2 ALGORITMOS



1

## Algoritmo 1: Shortest Video Transmission Path

- Encuentra rutas desde nodo u → conjunto de destinos D
- Explora vecinos calculando el costo mínimo por subconjunto
- Usa recursión sobre subconjuntos de destinos

2

## Algoritmo 2: Optimize Video Transmission Paths

- Programación dinámica con enumeración exhaustiva
- Divide D en particiones óptimas
- Construye arbol multicast global

## Explicando OST: I

**Output-Constrained Minimum Flow Problem (OCMFP) se formula de la siguiente forma:**

**Problem 1.**

$$\min_{\mathbf{f}} \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} e_{(i,j)} f_{(i,j)}, \quad (1)$$

$$s.t. \max_{j \in \mathcal{V}} f_{(i,j)} \leq \max_{k \in \mathcal{V}} f_{(i,k)}, \quad \forall i \in \mathcal{V} / \mathcal{V}_s, \quad (1a)$$

$$\max_{j \in \mathcal{V}} f_{(i,j)} \geq \max_k x^k, \quad \forall k \in \mathcal{V}_d \wedge i \in \mathcal{V}_s, \quad (1b)$$

$$x^j \leq \max_{i \in \mathcal{V} \setminus \mathcal{V}_d} f_{(i,j)}, \quad \forall j \in \mathcal{V}_d, \quad (1c)$$

$$f_{(i,j)} \geq 0, \quad \forall i, j \in \mathcal{V}, \quad (1d)$$

$$f_{(i,j)} = 0, \quad \forall e_{(i,j)} \notin \mathcal{E}, \quad (1e)$$

El objetivo del problema 1 es minimizar la suma de pesos del flujo dentro de la red.

El problema se modela como un grafo no dirigido ponderado.  $G = (\mathcal{V}, \mathcal{E})$ ,  $\mathcal{V}$  representa el conjunto de todos los nodos, hay 3 tipos:  
 $\mathcal{V}_s$  = Source node  $\rightarrow |\mathcal{V}_s| = 1$ ,  
 $\mathcal{V}_d$  = Destination node  $\rightarrow |\mathcal{V}_d| = K$   
 $\mathcal{V}_r$  = General Routing node

$f(i,j)$  representa el flujo entre nodo  $i$  a nodo  $j$ , y el peso de la arista  $e(i,j)$  representa el costo unitario de transmisión desde nodo  $i$  hasta nodo  $j$

Tiene las siguientes restricciones:

- El flujo de salida no puede ser superior al de entrada (excepto  $\mathcal{V}_s$ )
- garantiza que el flujo de salida del nodo de origen supere los requisitos máximos de flujo de entrada de los nodos de destino
- garantiza que el flujo de entrada en cada nodo de destino sea al menos igual a su requisito de flujo de entrada especificado
- garantizan un flujo no negativo dentro de la red y restringen el flujo a los enlaces que existen dentro del gráfico



## Explicando OST: II

**On-Demand Steiner Tree (OST) sigue el siguiente estado de transicion de ecuaciones**

$$\mathcal{H}_{(d,\{d\})} = 0 \quad d \in \mathcal{V}_d \quad (2)$$

$$\mathcal{H}_{(v,\emptyset)} = +\infty \quad v \in \mathcal{V} \quad (3)$$

$$\mathcal{H}_{(v,S)} = \min_{\mathcal{F} \subseteq S} \left\{ \max \left( \mathcal{H}_{(v,\mathcal{F})}, \mathcal{H}_{(v,S \setminus \mathcal{F})} \right) \right\} \quad v \in \mathcal{V}, S \subseteq \mathcal{V}_d \quad (4)$$

$$\mathcal{H}_{(v,S)} = \min_{(v,u) \in \mathcal{E}} \left\{ \mathcal{H}_{(u,S)} + \max_{j \in S} \{x^j\} e_{(v,u)} \right\} \quad v \in \mathcal{V}, S \subseteq \mathcal{V}_d \quad (5)$$

Para resolver OCMFP eficientemente, OST está basado en programación dinámica.

$\mathcal{H}_{\{v,S\}}$  es el flujo total desde el nodo  $v$  al conjunto nodos de destino  $S$

La ecuación 2 y 3 definen límites específicos para el proceso de selección de aristas. En concreto el 2) indica que el flujo de cada nodo fuente hacia si mismo es cero. Ecuación 3 representa que el flujo se establece en el infinito para un nodo  $v$  cuando no hay ninguna fuente que transmita a él

La ecuación 4 divide el conjunto  $S$  en dos subconjuntos  $\mathcal{F}$  y  $S \setminus \mathcal{F}$  y cubre ambos desde  $v$ ; el coste sería el máximo de los costes porque cuando reutilizas rutas, la carga mayor domina.

La ecuación 5 actualiza el flujo mínimo desde un nodo  $v$  a un conjunto de destinos  $S$ , explorando todos los nodos  $u$  vecinos y considerando la suma de los costos de enviar flujo a través de  $u$  más el costo de conectar  $v$  a  $u$ , seleccionando la opción de menor costo. Usa formula estandar de mínimos caminos.

## Explicando OST: III

**On-Demand Steiner Tree (OST) sigue el siguiente estado de transición de ecuaciones**

$$\delta_{(v, \mathcal{V}_d)} \leq \mathcal{H}_{(v, \mathcal{V}_d)} \leq \min_{\mathcal{F} \subseteq \mathcal{V}_d} \left\{ \max \left( \delta_{(v, \mathcal{F})}, \delta_{(v, \mathcal{V}_d \setminus \mathcal{F})} \right) \right\} \quad (6)$$

$$\delta_{(v, \mathcal{V}_d)} \leq \mathcal{H}_{(v, \mathcal{V}_d)} \leq \min_{(v', v) \in \mathcal{E}} \left\{ \delta_{(v, \mathcal{V}_d)} + \max_{j \in \mathcal{V}_d} \{r_j\} e_{(v', v)} \right\} \quad (7)$$

Conjunto  $\delta_{(v, S)}$  el mínimo flujo teórico desde nodo  $v$  hasta el conjunto de nodos destino  $S$ . A través de los 4 funciones de transición, se obtiene que  $H_{(v, S)} = \delta_{(v, S)}$

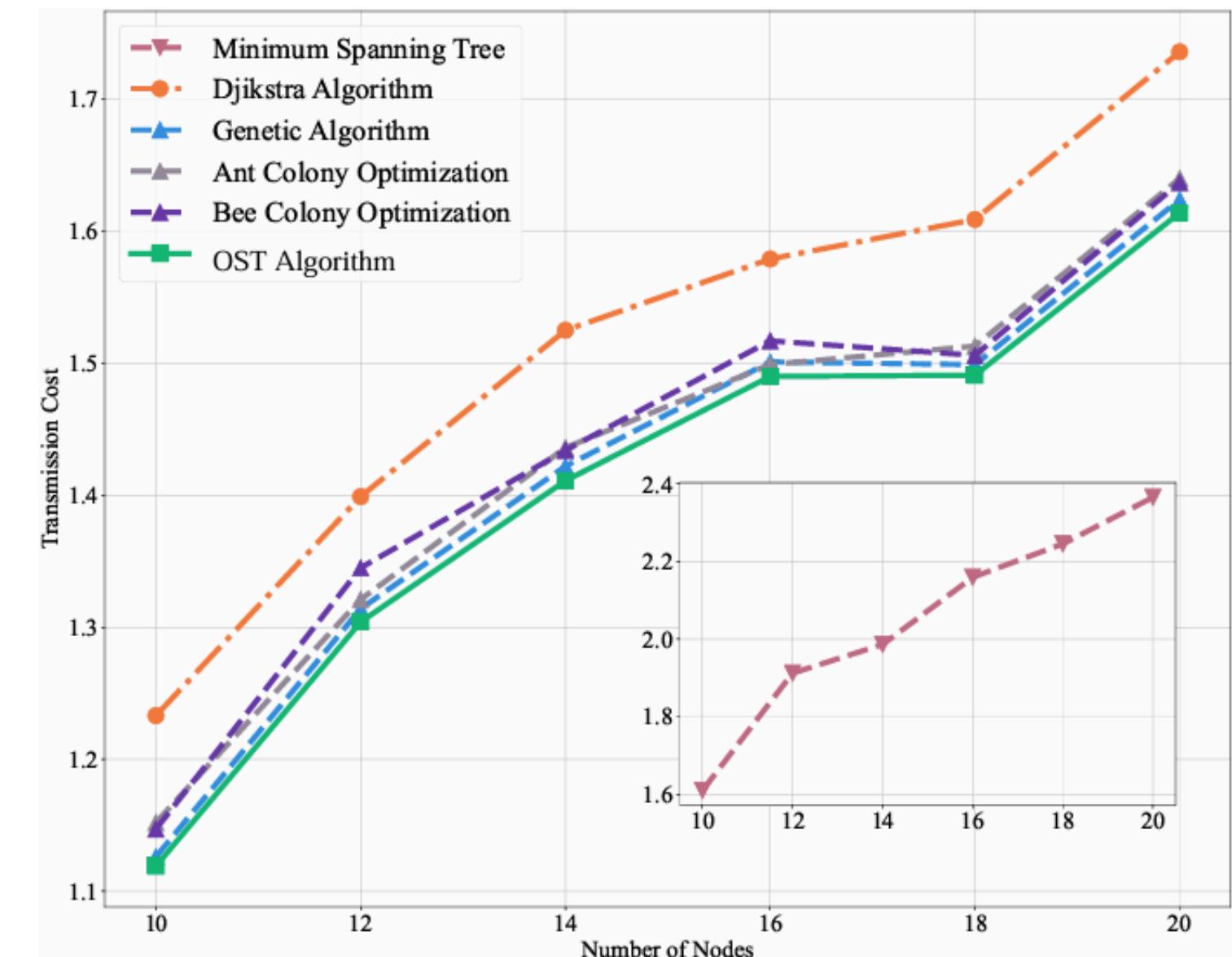
De La ecuación 4 tenemos la inecuación. Asegura que el flujo total desde  $v$  hacia  $S$  es igual al máximo flujo entre los subconjuntos, garantizando que la solución sea consistente y optimizada

De la ecuación 5 tenemos la inecuación 6, existe un nodo  $v$  para ambas inecuaciones que mantiene la igualdad. Combinando las soluciones recursivas para obtener el flujo más eficiente desde cada nodo  $v$  hacia el conjunto de destinos  $S$ , integrando toda la información para la construcción óptima del camino (ruta y flujo) mediante programación dinámica.

# Resultados

El paper señala que la complejidad del algoritmo escala exponencialmente en base a la cantidad de nodos destinatarios:  $O(2^{|V_d|})$

Resultados experimentales demuestran que OST obtiene 10%+ reducción en el flujo de la red en comparación a métodos existentes. El paper menciona que va a trabajar en la integración de arquitectura SDN



# Graph Neural Network-based (GNN) Multicast Routing

Con la evolución de las redes programables, las redes definidas por software SDN dan arquitectura centralizada para gestionar multicast más flexiblemente.

Enfoques basados en redes neuronales aunque ofrecen mayor velocidad de inferencia, suelen carecer de escalabilidad y generalización topológica

Para abordar estas limitaciones, este paper presenta un multicast routing basado en GNN que logra minimizar costo de TT y admite requisitos de calidad QoS.

El problema al igual que el anterior paper plantea desde MFP, pero este algoritmo desarrolla un algoritmo de aprendizaje por refuerzo para construir secuencialmente árboles multicast eficientes reutilizando rutas y adaptándose a la dinámica de la red.



# COMPONENTES PRINCIPALES

Para aprender eficientemente decisiones de de enrutamiento, el paper propone un Graph Policy Network (GPN) basado en los siguientes componentes:

Dado que aplica aprendizaje por refuerzo para guiar las decisiones de enrutamiento por nodos, el modelo no puede recibir retroalimentación inmediata después de cada acción. El costo total de transmisión solo está disponible después ya creada la ruta completa



## **Encoder: Graph Attention Network (GAT)**

Responsable de aprender node embedding que reflejen topología local y contexto de demanda QoS



## **Agregador: Long short-term Memory (LSTM)**

Se utiliza para capturar la dependencia secuencial de los nodos de enrutamiento seleccionados, un LSTM para agregar características de nodos elegidos previamente.



# Explicando GPN: I

Siendo  $\{x_k\}$  son las demandas de usuario, los usuarios son procesados de forma decscendiente, i.e.,  $(u_1, \dots, u_k)$ .

## GPN: Estado

$$s_t = (\mathcal{G}, \mathcal{V}_{\text{inflow}}^{(k)}, \mathcal{P}_t), \quad (5)$$

$$\mathcal{N}_t \triangleq \{ v \in \mathcal{V} : (u_t, v) \in \mathcal{E} \}. \quad (6)$$

1) Estado: Para el usuario  $k$  en el paso de desición  $t$ , el estado se define como en (5). Este estado incluye:

- Estructura grafo  $\mathcal{G}$
- El conjunto  $V$  inflow  $k$ -ésimo que contiene los nodos visitados o habilitados para ingreso por las desiciones previas de los usuarios 1 a  $k-1$
- La ruta parcial  $P_t$  para el usuario  $k$ , cuyo ultimo nodo es  $u_t$

El conjunto de nodos candidatos para avanzar en la ruta,  $N_t$ , estpa definido en 6)



# Explicando GPN: II

## GPN: Acción y recompensa

$$\mathcal{V}_{\text{inflow}}^{(k+1)} = \mathcal{V}_{\text{inflow}}^{(k)} \cup \mathcal{P}_k, \quad (7)$$

$$\mathcal{P}_{t+1} = \emptyset. \quad (8)$$

$$r_t = -x_{(k)} e(u_t, v_t), \quad (9)$$

$$R = \sum_t \gamma^t r_t$$

2) Acción: En cada paso t, se elige un nodo  $v_t$  perteneciente a  $N_t \cup V_{\text{inflow}}$  k-ésimo

- Si  $v_t$  pertenece a  $V_{\text{inflow}}$  k-ésimo, esto indica que la ruta para el usuario k ha terminado; en ese caso el episodio finaliza y el camino completo  $P_k$  se integra en  $V_{\text{inflow}}$  segun las ecuaciones (7) y (8)
- Si no, entonces el conjunto de las siguientes rutas parciales se actualiza como  $P_{t+1} = P_t \cup \{v_t\}$

3) Recompensa y objetivo: La recompensa en cada paso t refleja el costo de transmitir por la arista elegida (9) con un descuento gamma perteneciente a (0,1)

- El retorno total de una trayectoria es la suma ponderada de recompensas
- La meta es maximizar la expectativa de R, lo cual se traduce en optimizar la política para reducir el costo total de transmisión mientras se preservan oportunidades de reutilización futura del flujo.





## Explicando GPN: III

### GAT

$$e_{ij} = \text{LeakyReLU}(\mathbf{a}^\top [\mathbf{W}\mathbf{x}_i, |, \mathbf{W}\mathbf{x}_j]), \quad (10)$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}(i)} \exp(e_{ik})}, \quad (11)$$

$$\mathbf{h}_i = \sigma \left( \sum j \in \mathcal{N}(i) \alpha_{ij} \mathbf{W}\mathbf{x}_j \right), \quad (12)$$

GAT se utiliza para aprender las representaciones de los nodos que capturan tanto la topología local como el contexto de la demanda.

Para cada nodo  $i$  y su vecino  $j$  perteneciente a  $\mathcal{N}(i)$  se calcula el coeficiente de atención  $\alpha_{ij}$  mediante las ecuaciones (10) y (11).

Estos coeficientes determinan la importancia relativa de cada vecino para la actualización de la representación del nodo  $i$ .

La salida del encoder GAT son las embeddings de los nodos, que integran la información relevante para la decisión de ruteo en cada paso.





## Explicando GPN: IV

### LSTM

$$\mathbf{h}^t, \mathbf{c}^t = \text{LSTM}(\mathbf{x}^t; \mathbf{h}^{t-1}, \mathbf{c}^{t-1}), \quad (13)$$

$$p_v^t = \begin{cases} (\mathbf{h}^{t-1})^\top \tanh(W_2 \mathbf{x}_v + W_3 \mathbf{h}^{t-1}) & \text{if } v \notin \mathcal{P}_t, \\ -\infty & \text{otherwise,} \end{cases} \quad (14)$$

$$\pi\theta(s_t, a_t) = \text{Softmax}(p_v^t). \quad (15)$$

Para capturar la dependencia secuencial en la construcción de la ruta, se emplea un LSTM que agrega las características de los nodos seleccionados en pasos previos.

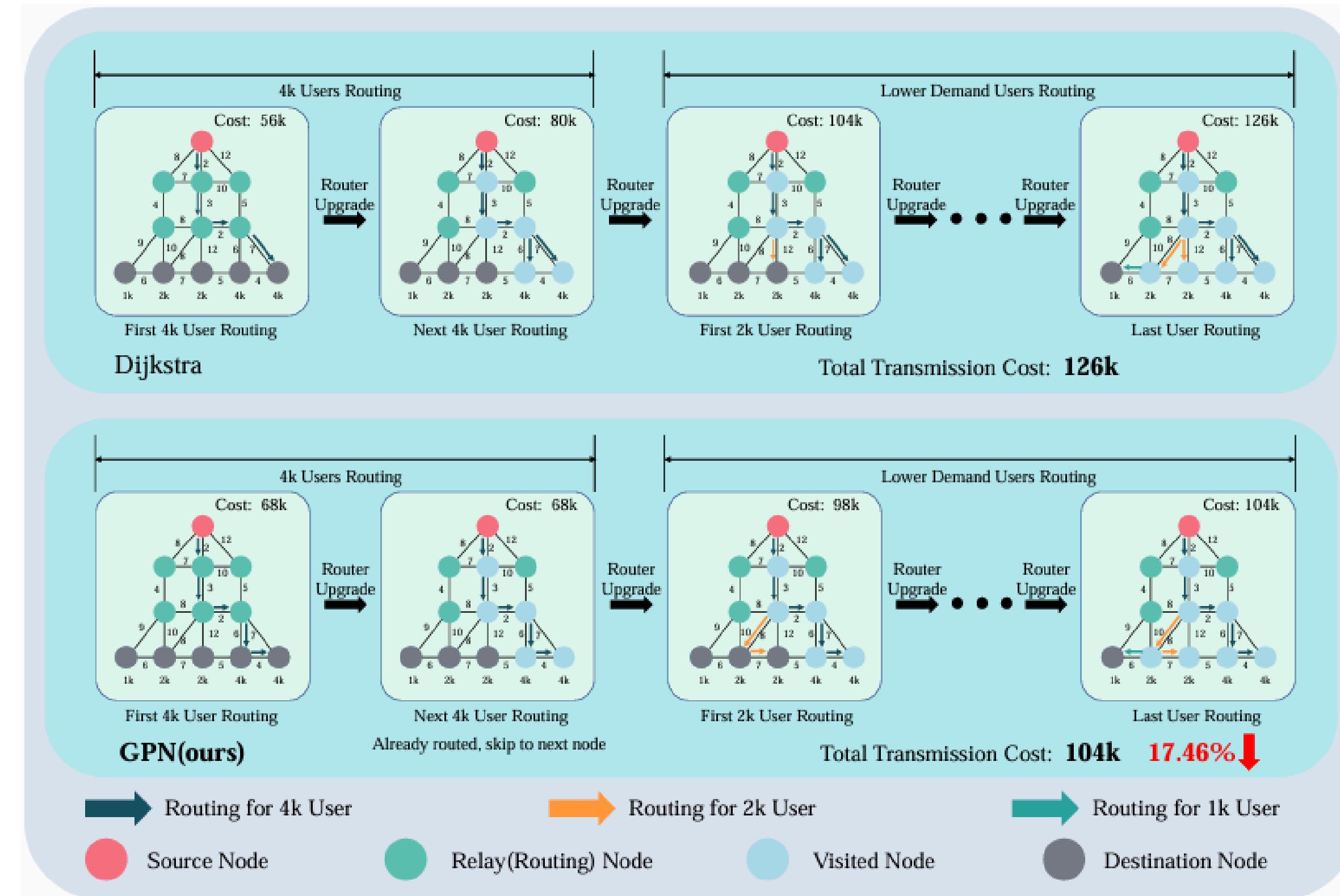
a actualización del estado oculto  $\mathbf{h}_t$  y la celda  $\mathbf{c}_t$  en el paso  $t$  se realiza mediante la ecuación (13), que recibe como entrada el embedding del nodo seleccionado en ese paso y los estados anteriores.

La salida del LSTM en cada paso,  $\mathbf{h}_t$ , representa la historia combinada de la ruta hasta ese momento, permitiendo decisiones informadas en pasos futuros.

El decodificador con atención usa el estado del LSTM ( $\mathbf{h}_{t-1}$ ) y las embeddings de los nodos candidatos para calcular un puntaje de atención  $p_{t,v}$  usando la ecuación (14). Estos puntajes indican la relevancia de cada nodo candidato para continuar la ruta, considerando tanto la historia acumulada como las características del nodo.



# VPN VS DIJKSTRA



# Conclusion GPN

GPN presenta un marco de enrutamiento multicast robusto y escalable que aprovecha las redes neuronales gráficas y el aprendizaje por refuerzo para abordar los retos críticos de las demandas heterogéneas de calidad de servicio, el dinamismo de la red y las limitaciones de escalabilidad inherentes a los métodos tradicionales de enrutamiento multicast

El algoritmo logra una complejidad de  $O(n)$ , logra costos solo 7% superiores al óptimo teórico y comparado con Dijkstra reduce el costo de transmisión total 24% manteniendo tiempo de respuesta subsegundo

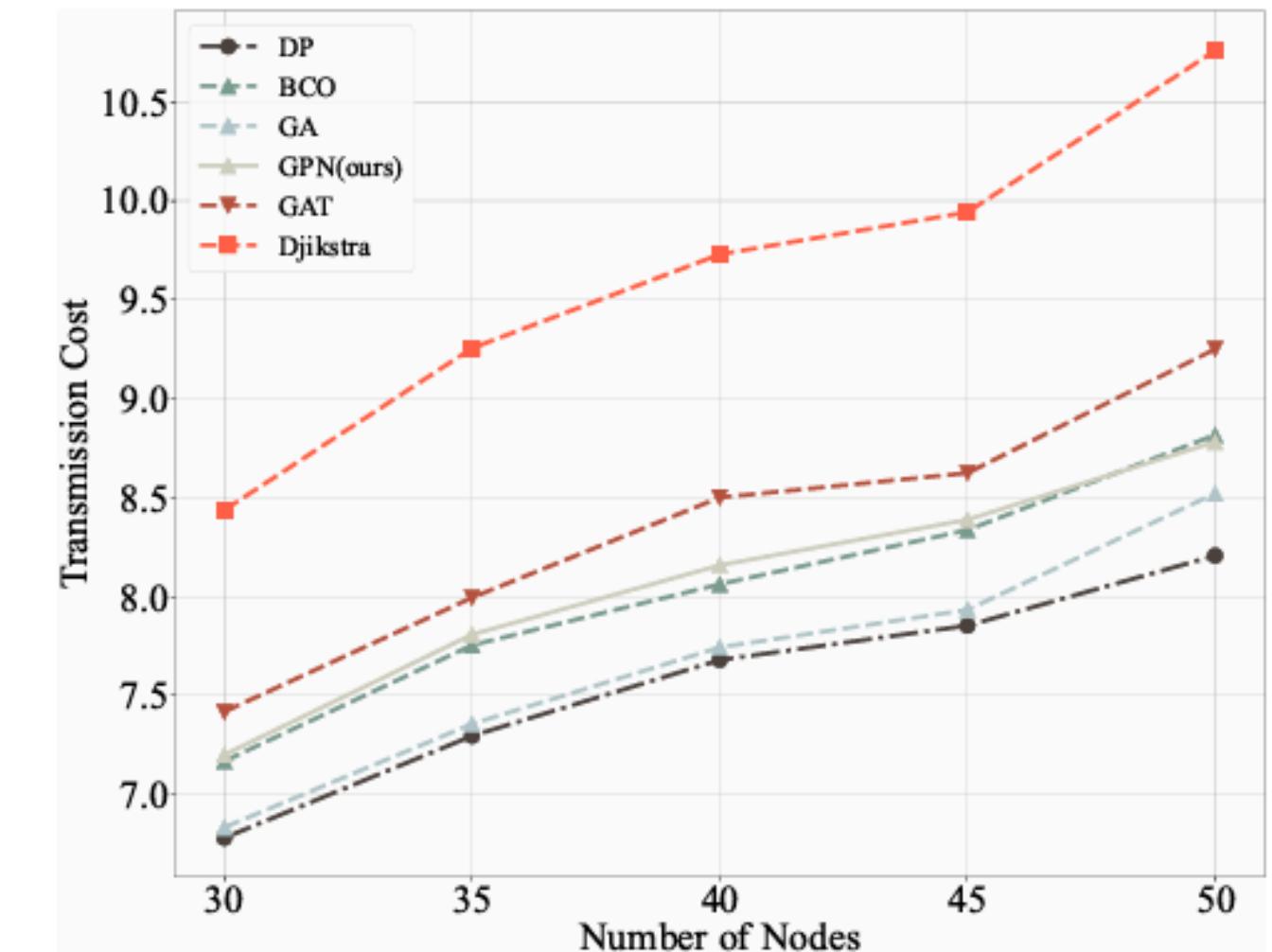
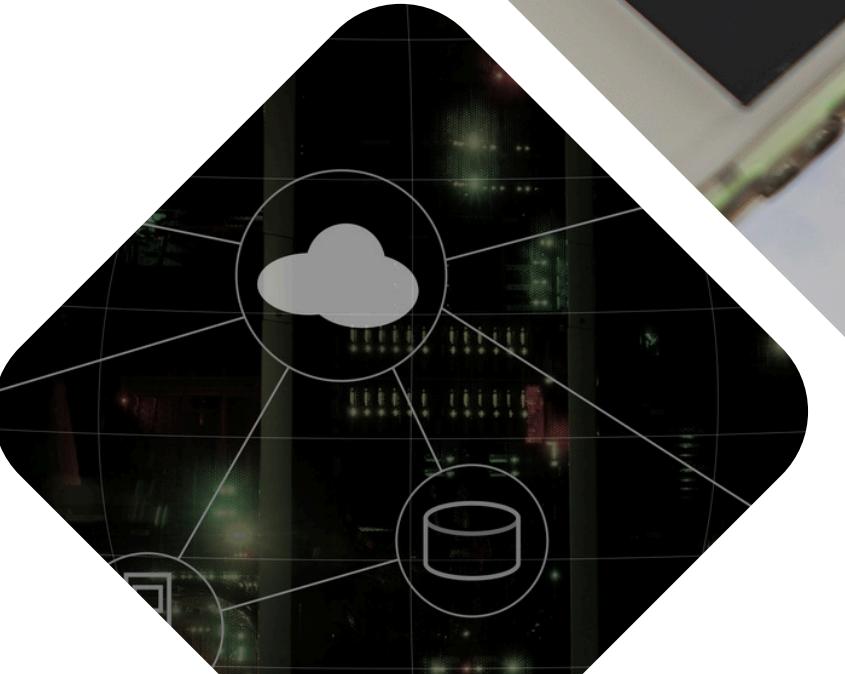


Fig. 3: Total routing cost vs. number of nodes (users = 12).

# Tree Generator-based Multicast Scheduling

Mientras la latencia ha sido tradicionalmente una métrica clave en redes de comunicación, es ampliamente reconocido que minimizar por sí solo el delay no garantiza actualización puntual de información. El concepto de Age of Information (AoI) a la frescura de la información disponible sobre un monitor, lo que hace una métrica de rendimiento adecuada para aplicaciones en tiempo real, como por ejemplo: sincronización entre transportes autónomos

Este paper propone un diseño de capa cruzada para optimizar conjuntamente las decisiones de routing y scheduling



# Resumen

Aunque algunos estudios han abordado programación multicast para optimización de AOL, a menudo pasan por alto el problema de enrutamiento multicast. Esta limitación surge de la naturaleza de las black box de las estructuras TCP/IP → Dificulta compartir info entre capas

Se ha propuesto diseño de capa cruzada (Cross-layers) para abordar estas limitaciones al optimizar conjuntamente las decisiones de enrutamiento y scheduling. Esta descomposición reduce la complejidad del problema original y facilita un diseño por RL Cross-layer



## **Sub problema 1: Scheduling**

El subproblema de programación es un problema primario que se centra en seleccionar los destinos que deben actualizarse en cada intervalo de tiempo, normalmente en una capa superior, como la capa de transporte.



## **Sub problema 2: Generación de árboles**

El subproblema de generación de árboles tiene como objetivo obtener un árbol de multidifusión óptimo para los destinos elegidos, que normalmente opera en una capa inferior, como la capa de red



# Explicando TGMS: I

## Métrica AOL

$$\hat{A}_p(t) \triangleq t - t_p, \quad \forall t \geq t_p, \quad (3)$$

$$A_u(t) \triangleq \begin{cases} \hat{A}_p(t), & \text{if } d_{u,p}(t) = 1, \\ A_u(t-1) + 1, & \text{otherwise,} \end{cases} \quad (4)$$

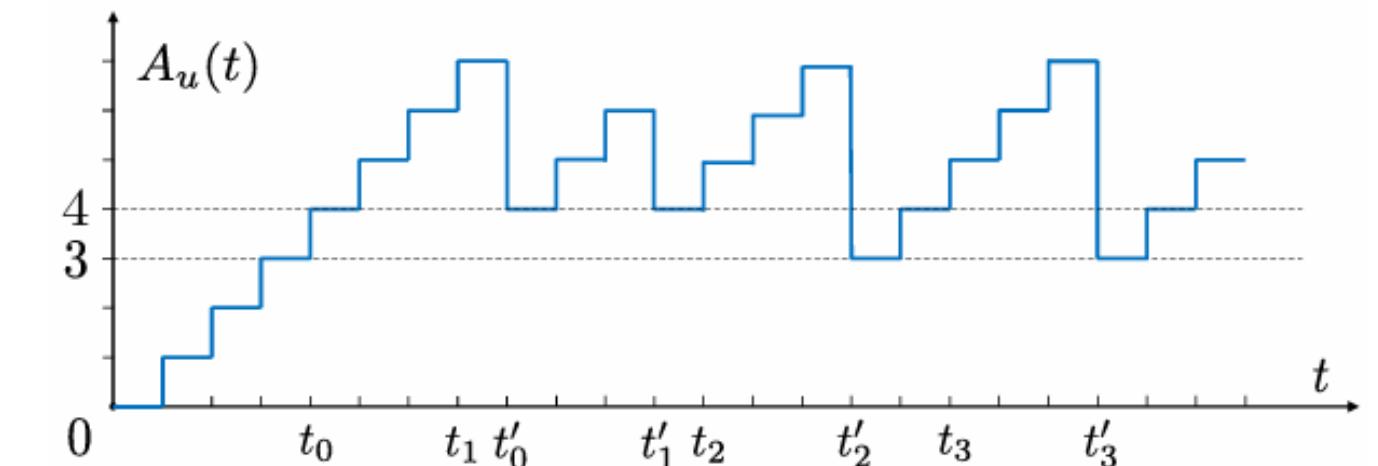
$$\bar{A} \triangleq \limsup_{T \rightarrow \infty} \frac{1}{T} \sum_{t=0}^T \sum_{u \in \mathcal{U}_t} \omega_u A_u(t), \quad (5)$$

Evolución del AOL a través del tiempo →

Para analizar AOL de cada destino, primero es necesario saber que es el age of packet(3). Donde  $t_p$  representa el momento en el que el packet p fue generado y  $A_p(t)$  incrementa linealmente en el tiempo.

El AOL de destino u perteneciente a  $\mathcal{U}_t$  es definido como en (4). Donde  $D_{u,p}(t)$  es el indicador que representa cuál packet p llega a destino u al momento t, este puede ser 1 si cumple 0 si no.

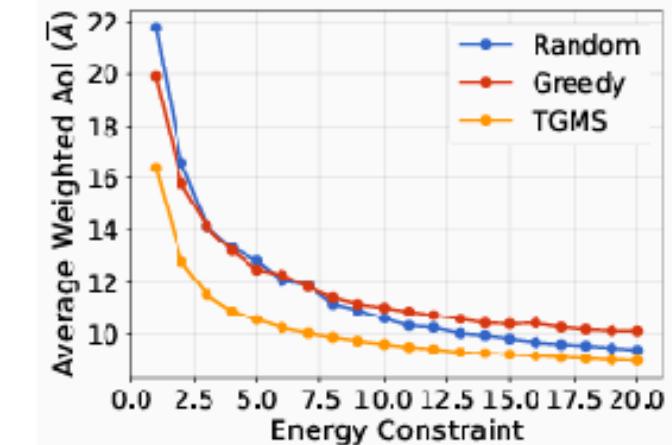
El promedio ponderado de AOL de la red se define como en (5)



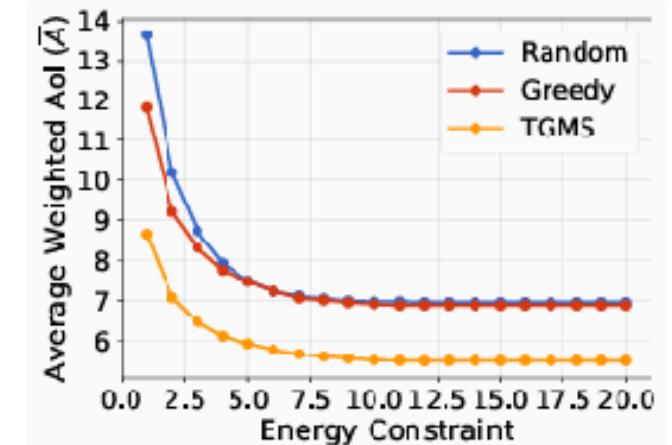
# Conclusion TGMS

El scheduler propuesto tiene un rendimiento superior a otros baselines para todos los datasets. reduce el peso promedio de AoI por 21,6% en escenarios de baja energía. También reduce el pico de antigüedad ponderada por 29,9%. s hasta 9.85x veces más eficiente computacionalmente que los algoritmos de enrutamiento multicast tradicionales, al tiempo que logra un rendimiento comparable a métodos SOTA.

El algoritmo TGMS propuesto tiene algunas limitaciones. En primer lugar, requiere mucha memoria para el entrenamiento de un grafo grande. En segundo lugar, es posible que el árbol multicast no pueda actualizarse en tiempo real debido a la complejidad del graf Tambien, tiene necesidad de reentrenarse si cambian restricciones

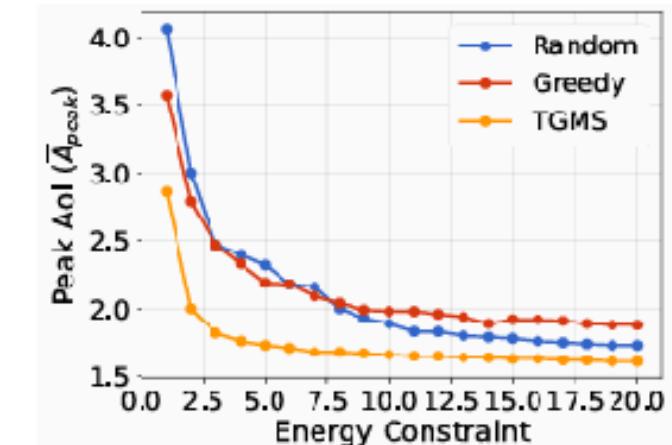


(a) Average Weighted AoI of I080.

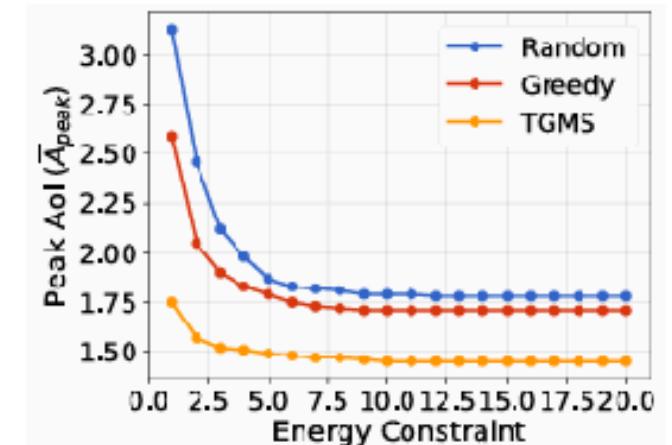


(b) Average Weighted AoI of AS-733.

Fig. 9. Average weighted AoI over datasets with  $|V| = 80$ .



(a) Peak Weighted Age of I080.

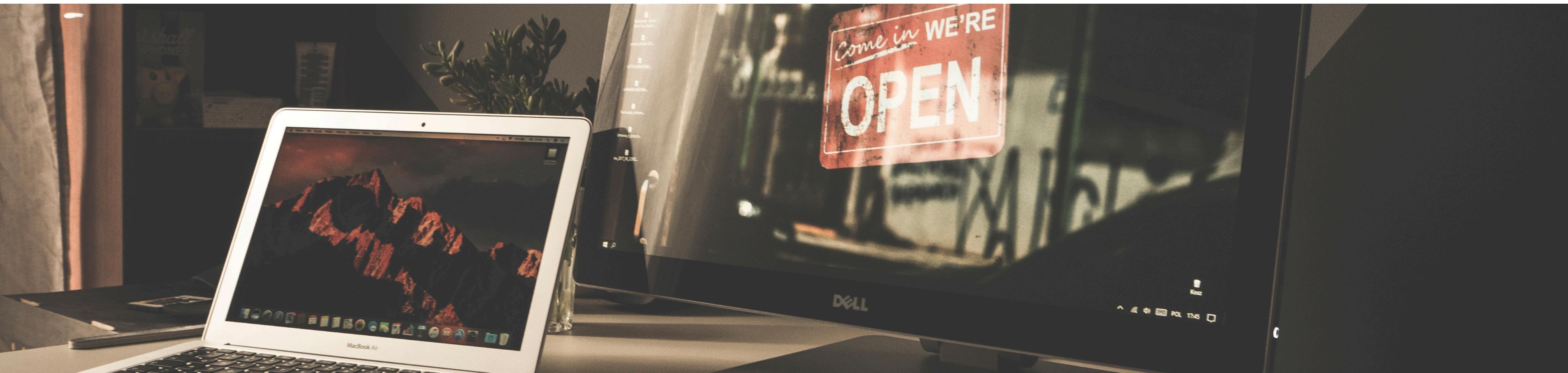


(b) Peak Weighted Age of AS-733.

# CONCLUSIONES FINALES



Este trabajo comenzó con la pregunta sobre cómo distribuir masivas cantidades de datos en las próximas redes 6G, estas deben poder reducir el costo de transmisión, adaptarlas a la diversidad de usuarios, dinámica topología, eficiencia computacional, y bajas latencias; como también AOL. Si bien, las investigaciones actuales todavía tienen algunos límites, el uso de redes neuronales, inteligencia artificial y aprendizaje automático parece ser el camino a seguir de este avance tecnológico que se construye día a día.





# MUCHAS GRACIAS