Comparación de Algoritmos para Caminos Más Cortos: Dijkstra vs. Nuevos Avances en Complejidad Subóptima

Pamela Diaz, Esmaydes Ccori, Hector Barazorda
Universidad Nacional de San Antonio Abad del Cusco
221446@unsaac.edu.pe, 221444@unsaac.edu.pe, 145003@unsaac.edu.pe

1. Antecedentes

El problema de los caminos más cortos desde un único origen (SSSP) ha sido un pilar fundamental en la teoría de algoritmos desde el seminal trabajo de Edsger W. Dijkstra en 1959. Durante más de seis décadas, su algoritmo, con una complejidad de $O(m + n \log n)$, se consideró óptimo para grafos con pesos no negativos, estableciendo un paradigma que parecía insuperable debido a la llamada "barrera de ordenación" .

Investigaciones posteriores exploraron los límites fundamentales del problema. Atalig et al. (2024) demostraron que cualquier algoritmo basado únicamente en la relajación de aristas —la técnica central de Dijkstra y Bellman–Ford— enfrenta límites inferiores que impiden superar ciertas cotas de tiempo, lo que sugería que se necesitaban enfoques radicalmente diferentes . Paralelamente, Karczmarz et al. (2023) estudiaron implementaciones en el modelo Word RAM, logrando avances para pesos racionales pero confirmando que en el modelo general de comparación-suma el límite de Dijkstra permanecía intacto .

El panorama comenzó a cambiar con Duan et al. (2023), quienes presentaron el primer algoritmo aleatorizado que superaba la barrera $O(m + n \log n)$ para grafos no dirigidos Ėste resultado fue trascendental, pues demostró por primera vez que la ordenación completa de vértices no era inherentemente necesaria para resolver SSSP. Sin embargo, la pregunta sobre si era posible lograr una mejora determinista para grafos dirigidos permanecía abierta, representando el siguiente gran desafío en la teoría de algoritmos de caminos más cortos.

En este contexto, nuestro trabajo se sitúa como la respuesta a esta pregunta pendiente, presentando el primer algoritmo determinista que rompe la barrera de Dijkstra para grafos dirigidos con pesos no negativos en el modelo de comparación-suma.

2. Estado del Arte

El problema de los caminos más cortos desde un único origen (SSSP, por sus siglas en inglés) representa uno de los pilares fundamentales en la teoría de grafos y el diseño de algoritmos. Durante más de seis décadas, el panorama para resolver este problema en grafos dirigidos con pesos no negativos estuvo dominado por el algoritmo de Dijkstra, cuya complejidad de $O(m+n \log n)$ —usando montículos de Fibonacci— fue considerada óptima para grafos dispersos ($m = O(n)$). Esta creencia se sustentaba en la llamada *barrera de ordenación* ($\Omega(n \log n)$), un límite inferior aparentemente inherente al proceso de seleccionar vértices por distancia mínima, que implica una ordenación implícita.

Sin embargo, esta visión ha sido recientemente revolucionada. El trabajo seminal de **Duan et al. (2024)** en "Breaking the Sorting Barrier for Directed Single-Source Shortest Paths" marca un punto de inflexión al presentar el primer algoritmo **determinista** que supera el límite de Dijkstra para grafos dirigidos con pesos reales no negativos, logrando un tiempo de ejecución de $O(m \log^{2/3} n)$ en el modelo de comparación-suma.

La clave de este avance radica en una **combinación híbrida e ingeniosa** de los paradigmas de Dijkstra y Bellman-Ford, estructurada dentro de un esquema recursivo de divide y vencerás. La técnica central, conocida como **"reducción de la frontera"**, evita la ordenación completa mediante la identificación estratégica de **vértices pivote** cuyos árboles de caminos más cortos cubren grandes porciones del grafo. El algoritmo principal, denominado **BMSSP (Bounded Multi-Source Shortest Path)**, resuelve recursivamente el problema para un conjunto de fuentes con una cota superior, limitando el tamaño del conjunto de vértices activos (la frontera) de $|\tilde{U}|$ a $|\tilde{U}|/\log^{\Omega(1)} n$. Este resultado demuestra de manera concluyente que la barrera de $O(m + n \log n)$ no es fundamental para el problema SSSP en grafos dirigidos.

Este hito se contextualiza dentro de una línea de investigación más amplia. Previamente, **Duan et al. (2023)** habían logrado un avance similar para **grafos no dirigidos** con un algoritmo aleatorizado en $O(m\sqrt{\log n \log\log n})$. En problemas relacionados, como el del **Árbol de Expansión Mínima (MST)** y las **Rutas de Cuello de Botella**, ya se habían superado barreras de ordenación con algoritmos de tiempo

casi lineal, sugiriendo que era posible un camino similar para SSSP.

Paralelamente, se han producido avances espectaculares en el problema más general de **SSSP con pesos negativos**. **Bernstein, Nanongkai y Wulff-Nilsen (2022)** demostraron que es posible resolver este problema en tiempo **casi lineal**, $O(m \cdot \text{polylog}(n) \log C)$, para pesos enteros, donde C es la cota del peso máximo. Este resultado fue posteriormente refinado y evaluado en la práctica por **Cassis et al. (2024)**. Trabajos más recientes, como el de **Huang, Jin y Quanrud (2024)**, introducen el concepto de **"distancia de salto propia"** para obtener algoritmos más rápidos con pesos reales negativos.

En otros modelos de computación, para el **modelo Word RAM** con pesos enteros, **Thorup (1999)** estableció un algoritmo de tiempo lineal para grafos no dirigidos. Recientemente, **Karczmarz, Nadara y Sokolowski (2023)** abordaron el problema de los caminos exactos con pesos racionales en este modelo. Para escenarios dinámicos, **Karczmarz y Sankowski (2024)** presentan algoritmos eficientes para mantener caminos más cortos en dígrafos bajo actualizaciones. En el ámbito de la computación a gran escala, **Dory y Matar (2024)** diseñan algoritmos masivamente paralelos para soluciones aproximadas, mientras que **Ashvinkumar et al. (2024)** exploran incluso modelos de computación cuántica para pesos negativos.

La investigación de **límites inferiores** es crucial para entender la optimalidad de estos algoritmos. **Haeupler et al. (2024)** demostraron que Dijkstra **es óptimo** si el algoritmo debe producir un **ordenamiento total** de los vértices por su distancia, lo que resalta la importancia de la suposición en Duan et al. de que solo se necesitan las distancias, no el orden. Por su parte, **Atalig et al. (2024)** establecen cotas inferiores para una clase amplia de algoritmos basados en relajación adaptativa, sugiriendo que mejorar aún más ciertos límites requerirá técnicas que trasciendan este paradigma.

En contraste con los avances teóricos, trabajos como el de **Valko et al. (2025)** abordan el problema desde una perspectiva aplicada, proponiendo heurísticas para redes muy dispersas como Lightning Network, aunque sin las garantías teóricas sólidas de los otros enfoques.

En conclusión, el panorama actual del SSSP está experimentando una revolución sin precedentes. La creencia de que el algoritmo de Dijkstra era óptimo para grafos dirigidos dispersos ha sido refutada. El algoritmo de **Duan et al. (2024)** se erige como una contribución fundamental al ofrecer el primer algoritmo determinista que rompe la barrera de ordenación, redefiniendo las fronteras de lo posible en este problema clásico. La investigación futura se centrará probablemente en mejorar el factor logarítmico, extender estas ideas a pesos negativos, y trasladar estas mejoras teóricas a implementaciones prácticas eficientes, cerrando la brecha entre la complejidad teórica y la aplicabilidad práctica.

3. Problema de investigacion

En la actualidad, muchas aplicaciones del mundo real dependen del uso de redes o grafos: redes sociales, mapas de calles utilizados por aplicaciones de navegación, sistemas de transporte, redes de comunicación, plataformas de recomendación, entre otros. En todos estos escenarios es fundamental calcular de manera rápida y eficiente rutas o caminos entre diferentes puntos. Esta tarea, conocida como el problema del camino más corto, se resuelve mediante algoritmos específicos diseñados para operar sobre grafos.

Entre los algoritmos más empleados y estudiados se encuentran **Dijkstra**, **Bellman-Ford**, **BFS** (Breadth-First Search) y **A***. Cada uno posee características particulares que los hacen más o menos adecuados dependiendo del tipo de red o de la información disponible. Sin embargo, en la práctica estos algoritmos suelen seleccionarse por tradición o por conveniencia, sin considerar evidencia experimental que demuestre cuál de ellos resulta realmente más eficiente bajo diferentes condiciones. Esto genera decisiones que pueden ser poco óptimas en escenarios reales.

El algoritmo de **Dijkstra** ha sido, durante décadas, el estándar para resolver caminos más cortos en grafos con pesos no negativos debido a su rapidez y facilidad de implementación. No obstante, su rendimiento depende de factores como el tamaño del grafo, la distribución de sus conexiones y las estructuras de datos empleadas. Por su parte, **Bellman-Ford** es más general porque permite trabajar con pesos negativos, pero su complejidad $O(nm)$ lo hace considerablemente más lento, lo cual limita su uso en aplicaciones prácticas.

En casos donde los grafos no tienen pesos o todos los pesos son iguales, el algoritmo **BFS** puede superar a Dijkstra en velocidad, siendo una opción extremadamente eficiente. Sin embargo, esto únicamente ocurre en escenarios específicos que no siempre son correctamente identificados por el usuario o desarrollador.

Finalmente, el algoritmo **A*** destaca por emplear una heurística que guía la búsqueda, permitiendo acelerar el proceso cuando existe información geográfica o espacial. Su desempeño puede ser notablemente superior al de Dijkstra en mapas o redes físicas. No obstante, cuando los nodos no poseen coordenadas o se utiliza una heurística inadecuada, su rendimiento puede igualar o incluso ser inferior al de Dijkstra.

Esta diversidad de comportamientos evidencia un problema central: aunque estos algoritmos son conocidos teóricamente, **existe poca evidencia experimental comparativa** que permita comprender cómo se desempeñan realmente en distintos tipos de grafos, especialmente aquellos representativos de aplicaciones

reales. La enseñanza y el uso común de estos algoritmos se basan mayormente en complejidad teórica, pero no en mediciones empíricas.

A esto se suma la aparición de algoritmos más recientes que, en teoría, prometen superar a Dijkstra en eficiencia asintótica. Sin embargo, aún no se sabe con claridad si estas propuestas modernas presentan ventajas reales en implementaciones prácticas o si sus mejoras solo se manifiestan en grafos de gran escala o bajo condiciones ideales.

Todo esto crea una brecha importante entre:

- la teoría algorítmica,
- el comportamiento práctico en hardware real,
- y las decisiones cotidianas de ingenieros y desarrolladores.

Por ello, surge la necesidad de realizar un estudio experimental comparativo que evalúe directamente los algoritmos **Dijkstra**, **Bellman-Ford**, **BFS** y **A*** en diferentes tipos de grafos: sociales, geográficos y aleatorios. Dicha evaluación debe considerar métricas como tiempo de ejecución, uso de memoria, escalabilidad y corrección de resultados.

Este análisis permitirá determinar con evidencia empírica qué algoritmo es más adecuado en cada contexto y bajo qué condiciones cada uno demuestra su verdadera eficiencia. En consecuencia, el estudio busca responder la siguiente pregunta central:

4. Problema General

¿Cuál de los algoritmos Dijkstra, Bellman-Ford, BFS y A* ofrece el mejor rendimiento práctico en diferentes tipos de grafos, y bajo qué condiciones cada uno resulta verdaderamente eficiente?

5. Problemas Específicos

El estudio toma en cuenta los siguientes problemas específicos:

- Analizar el rendimiento práctico de Dijkstra y Bellman–Ford en redes sociales, geográficas y aleatorias, considerando velocidad y escalabilidad.
- Evaluar cómo la estructura interna del grafo (densidad, distribución de grados, presencia de coordenadas) influye en el desempeño de A* y BFS frente a Dijkstra.
- Determinar si las mejoras teóricas recientes ofrecen ventajas prácticas frente a los algoritmos tradicionales en grafos de tamaño moderado.

6. Justificacion

La presente investigación es relevante porque aborda una problemática actual en el campo del análisis de redes y el diseño de algoritmos eficientes: la falta de evidencia empírica que permita comparar de manera rigurosa el rendimiento real de los algoritmos clásicos y los desarrollos modernos para el cálculo de caminos más cortos. A pesar de que algoritmos como Dijkstra, Bellman–Ford y A* son ampliamente utilizados en sistemas de navegación, planificación logística y análisis computacional, existe poca documentación experimental —especialmente en contextos educativos y de investigación aplicada— que contraste su desempeño en diferentes tipos de redes.

La importancia del estudio se justifica también por el impacto práctico. Muchos sistemas digitales requieren ejecutar consultas de rutas con tiempos de respuesta reducidos, por lo que elegir adecuadamente el algoritmo puede representar mejoras significativas en rendimiento, consumo de recursos y escalabilidad. Este trabajo permitirá identificar cuáles algoritmos son realmente convenientes para distintas categorías de redes, generando conocimiento valioso para el diseño de aplicaciones de ruteo, simulación, análisis social o transporte.

En el ámbito académico, esta investigación contribuye al entendimiento comparativo de algoritmos fundamentales en teoría de grafos, proporcionando ejemplos experimentales que complementan el enfoque teórico. Los estudiantes, docentes e investigadores podrán apoyarse en los resultados para comprender las diferencias entre complejidad teórica y desempeño empírico, tema que suele ser poco abordado en la literatura introductoria.

Asimismo, el estudio aporta una perspectiva metodológica clara para evaluar algoritmos en condiciones controladas, incorporando métricas cuantitativas reproducibles. Este enfoque ayuda a cerrar la brecha entre la teoría algorítmica avanzada —incluyendo trabajos recientes que rompen barreras históricas de complejidad— y las necesidades prácticas de implementación en entornos reales de tamaño moderado.

Finalmente, la investigación ofrece beneficios sociales e institucionales. Los resultados pueden emplearse en proyectos de software, cursos de optimización y sistemas de información geográfica para seleccionar algoritmos más eficientes, reduciendo costos computacionales y mejorando la experiencia del usuario. Además, se fortalece la capacidad investigativa en la institución al desarrollar un estudio experimental replicable y fundamentado en algoritmos modernos.

7. Objetivos

7.1. Objetivo General

Evaluar y comparar el rendimiento práctico de los algoritmos de búsqueda de caminos más cortos (Dijkstra, Bellman-Ford, BFS y A*) en grafos de topología diversa y tamaño moderado, para determinar cuál ofrece el mejor equilibrio entre eficiencia computacional y facilidad de implementación en escenarios reales.

7.2. Objetivos Específicos

1. Implementar los algoritmos seleccionados en un entorno controlado utilizando Python y la librería NetworkX, asegurando la reproducibilidad de los resultados.

2. Medir experimentalmente el tiempo de ejecución y el consumo de memoria de cada algoritmo al procesar grafos sociales, mallas geográficas y grafos aleatorios sintéticos.

3. Analizar cómo influyen las características del grafo (densidad de aristas, presencia de coordenadas espaciales y distribución de pesos) en la ventaja relativa del algoritmo A* frente a Dijkstra.

4. Contrastar la eficiencia empírica de los algoritmos clásicos frente a las expectativas teóricas, verificando si las mejoras asintóticas modernas son necesarias para aplicaciones de escala moderada.

8. Marco Teórico

El problema de los caminos más cortos constituye uno de los pilares fundamentales de la teoría de grafos y la optimización algorítmica. Para comprender la discrepancia entre la teoría moderna y la práctica habitual, es necesario analizar tanto los fundamentos clásicos que dominan las aplicaciones actuales como los avances teóricos recientes que han redefinido los límites de complejidad.

8.1. Definición del Problema

Dado un grafo dirigido y ponderado $G = (V, E)$ con una función de peso $w : E \to \mathbb{R}$, y un vértice fuente s, el problema *Single-Source Shortest Path* (SSSP) consiste en encontrar la distancia mínima $d(v)$ desde s a todo $v \in V$. La propiedad fundamental que permite resolver este problema eficientemente es la *subestructura óptima*: cualquier subcamino de un camino más corto es, a su vez, un camino más corto.

8.2. Algoritmos Clásicos (Base del Estudio)

Los siguientes algoritmos representan el estándar en la industria y son el objeto central de nuestra evaluación experimental.

8.2.1. Algoritmo de Dijkstra

Propuesto en 1959 [1], utiliza una estrategia voraz (*greedy*). Mantiene un conjunto de nodos procesados cuyas distancias son definitivas y relaja las aristas salientes del nodo no procesado con la menor distancia tentativa.

- **Complejidad:** Su implementación estándar con colas de prioridad (Binary Heap) tiene un costo de $O(m \log n)$.
- **Limitación:** Requiere pesos no negativos ($w(e) \geq 0$).

8.2.2. Algoritmo de Bellman-Ford

A diferencia de Dijkstra, este algoritmo es capaz de manejar pesos negativos (siempre que no existan ciclos negativos). Funciona relajando todas las aristas del grafo $|V| - 1$ veces.

- **Complejidad:** $O(nm)$.
- **Uso:** Es significativamente más lento que Dijkstra en grafos dispersos, pero su robustez lo hace necesario en dominios financieros o de redes específicas.

8.2.3. Búsqueda en Anchura (BFS)

Para el caso especial de grafos no ponderados (o donde $w(e) = 1$ para todo e), el problema se simplifica a encontrar el camino con menor número de aristas. BFS explora el grafo por niveles concéntricos desde el origen.

- **Complejidad:** $O(m + n)$.
- **Relevancia:** Es el límite inferior teórico de velocidad para cualquier recorrido de grafo completo; ningún algoritmo de caminos ponderados puede ser asintóticamente más rápido que BFS.

8.2.4. Algoritmo A* (A-Star)

El algoritmo A* es una extensión de Dijkstra que utiliza información heurística para guiar la búsqueda hacia el objetivo, priorizando nodos que parecen estar más cerca del destino final. Selecciona nodos basándose en la función de costo:

$$f(n) = g(n) + h(n)$$

Donde $g(n)$ es el costo real desde el inicio y $h(n)$ es la estimación heurística hasta el destino.

- **Condición:** Para garantizar la optimalidad, la heurística $h(n)$ debe ser *admisible* (no sobreestimar la distancia real). En grafos euclidianos, la distancia en línea recta cumple esta condición.

8.3. Fronteras Teóricas y Nuevos Avances

Aunque este estudio se centra en la evaluación práctica de los algoritmos anteriores, es crucial contextualizar su posición frente al estado del arte teórico.

8.3.1. La Barrera de Ordenación

Durante décadas, se asumió que el límite inferior para SSSP en grafos dirigidos era $\Omega(m + n \log n)$, dictado por la necesidad aparente de ordenar los nodos por distancia. Sin embargo, trabajos recientes han desafiado esta noción.

8.3.2. Avances Deterministas (Duan et al.)

Como se mencionó en los antecedentes, Duan et al. (2025) [5] presentaron el primer algoritmo determinista que rompe la barrera de Dijkstra, logrando $O(m \log^{2/3} n)$. Este algoritmo utiliza técnicas complejas de *scaling* y estructuras de datos jerárquicas para evitar la ordenación total.

Aunque teóricamente superior, la implementación de estos nuevos algoritmos conlleva factores constantes elevados y una complejidad de código que dificulta su adopción inmediata en la ingeniería de software cotidiana.

8.4. Análisis Comparativo y Perspectivas Futuras

8.4.1. Fortalezas del algoritmo $O(m \log^{2/3} n)$

1. Reduce el tiempo asintótico en grafos dispersos.
2. Evita la ordenación completa, superando un límite histórico.
3. Establece nuevas técnicas híbridas para algoritmos de rutas.

8.4.2. Razones para la vigencia de Dijkstra

- Simplicidad y facilidad de implementación.
- Rendimiento robusto en casos promedio.
- Amplia disponibilidad en bibliotecas y sistemas.
- Bajo costo constante.

8.4.3. Líneas de investigación futura

1. Reducción de factores constantes en implementaciones prácticas.
2. Extensión a pesos negativos y otros modelos de cómputo.
3. Integración con algoritmos dinámicos.
4. Validación empírica en redes reales de gran escala.

9. Hipótesis

Basándonos en el estudio, planteamos las siguientes hipótesis para nuestro estudio empírico. La hipótesis principal sostiene que, en aplicaciones prácticas con redes del mundo real de tamaño moderado, el algoritmo de Dijkstra implementado con estructuras de datos simples presentará el mejor equilibrio entre rendimiento y simplicidad. Esta hipótesis se desglosa en tres afirmaciones específicas y comprobables:

1. **Superioridad práctica de Dijkstra:** Para grafos con pesos no negativos, el algoritmo de Dijkstra será significativamente más rápido (entre 20 y 100 veces) que Bellman-Ford, siendo esta ventaja más pronunciada en grafos densos.
2. **Ventaja condicional de A*:** En grafos con información geográfica (como mapas de calles), el algoritmo A* con heurística euclidiana reducirá el tiempo de búsqueda entre 1.5 y 3 veces comparado con Dijkstra. Sin embargo, en redes sin coordenadas significativas (como redes sociales), A* no ofrecerá ventajas.
3. **Ineficacia práctica de algoritmos complejos:** Los algoritmos teóricamente avanzados que superan barreras asintóticas (como el de Duan et al. 2024) no ofrecerán ventajas prácticas en implementaciones simples para grafos de tamaño moderado ($n < 50,000$), debido a su complejidad de implementación y constantes ocultas grandes.

10. Alcance del Estudio

Este estudio se delimita específicamente en cuatro dimensiones: los algoritmos implementados, los tipos de redes evaluados, las métricas de medición y la configuración experimental.

10.1. Algoritmos Implementados

Se implementarán y compararán cuatro algoritmos clásicos y ampliamente documentados:

- **Dijkstra:** Con cola de prioridad basada en binary heap (implementación con `heapq` de Python).

- **Bellman-Ford:** Versión estándar con detección básica de ciclos negativos.
- **BFS (Breadth-First Search):** Para grafos no ponderados o con pesos unitarios.
- **A*:** Exclusivamente con heurística de distancia euclidiana, aplicable solo a grafos con coordenadas geográficas.

10.2. Conjuntos de Datos

La evaluación se realizará sobre tres tipos de redes representativas y **fácilmente accesibles**:

- **Red Social:** Utilizaremos grafos de ejemplo incluidos en la librería NetworkX, específicamente:
  - `karate_club_graph()`: Red del club de karate de Zachary (34 nodos, 78 aristas)
  - `florentine_families_graph()`: Red de matrimonios entre familias florentinas (16 nodos)
  - `davis_southern_women_graph()`: Red de asistencia a eventos (32 nodos)

  Estos grafos pequeños pero reales son ideales para pruebas iniciales y están disponibles sin necesidad de descargas externas.
- **Mapa de Calles:** En lugar de descargar datos complejos de OpenStreetMap, generaremos grafos sintéticos con propiedades de red urbana:
  - Grafos de malla (`grid_2d_graph`) de tamaño 20x20 (400 nodos, 760 aristas)
  - Grafos aleatorios geométricos (`geographical_threshold_graph`) con distribución espacial
  - Pesos asignados como distancia euclidiana entre nodos conectados
- **Grafos Aleatorios Sintéticos:** Generados programáticamente con NetworkX:
  - Modelo Erdős–Rényi: `erdos_renyi_graph(n=500, p=0.1)` (500 nodos, densidad media)
  - Modelo Barabási–Albert: `barabasi_albert_graph(n=300, m=2)` (300 nodos, estructura scale-free)
  - Modelo Watts-Strogatz: `watts_strogatz_graph(n=200, k=4, p=0.3)` (200 nodos, mundo pequeño)

10.3. Métricas de Evaluación

Para cada combinación algoritmo-dataset se medirá:

- **Tiempo de ejecución:** En segundos, usando `time.perf_counter()` de Python.
- **Consumo de memoria:** Pico de uso de RAM en megabytes.
- **Corrección:** Validación cruzada de que todos los algoritmos exactos calculan las mismas distancias mínimas.

10.4. Configuración Experimental

- **Lenguaje:** Python 3.10, por su accesibilidad y ecosistema científico.
- **Librerías principales:** NetworkX para manipulación de grafos, OSMnx para datos de OpenStreetMap, matplotlib para visualización.
- **Hardware:** Computadora personal estándar (8+ GB RAM, procesador de 4 núcleos).
- **Reproducibilidad:** Se fijarán semillas aleatorias y se promediarán sobre 10 ejecuciones por prueba.

11. Limitaciones

Es crucial reconocer explícitamente las fronteras de este estudio para contextualizar adecuadamente sus hallazgos y garantizar una interpretación correcta de los resultados.

11.1. Limitaciones Algorítmicas

- **Exclusión de algoritmos avanzados:** No se implementarán algoritmos de vanguardia teórica como BMSSP (Duan et al. 2024), algoritmos para pesos negativos complejos, o versiones paralelas. La justificación es su extrema complejidad de implementación, que excede el alcance de un estudio práctico comparativo.
- **Heurísticas limitadas para A*:** Solo se utilizará la distancia euclidiana como heurística. No se explorarán heurísticas aprendidas, adaptativas o específicas de dominio.

11.2. Limitaciones Prácticas

- **Tamaño de grafos:** El estudio se limita a grafos con hasta 50,000 nodos y 200,000 aristas, restringido por la capacidad de cómputo en hardware personal. No se considerarán grafos a escala de internet.

- **Implementación en Python:** Los tiempos absolutos reportados serán mayores que los obtenibles en lenguajes compilados como C++. Sin embargo, las tendencias relativas entre algoritmos serán válidas y representativas.
- **Hardware específico:** Los resultados son reproducibles en configuraciones similares, pero pueden variar significativamente en hardware diferente.

11.3. Limitaciones Metodológicas

- **Tipos de redes limitados:** Solo se evaluarán tres categorías de grafos (sociales, geográficos, aleatorios), que pueden no cubrir todos los casos de uso del mundo real.
- **Casos promedio:** Las evaluaciones se centrarán en casos promedio con pares origen-destino aleatorios, sin análisis exhaustivo de casos patológicos diseñados.
- **Optimizaciones básicas:** No se implementarán optimizaciones avanzadas a nivel de caché, paralelismo, o instrucciones específicas del procesador.

11.4. Limitaciones de Generalización

- **No aplicabilidad a grafos dinámicos:** Todos los grafos son estáticos. No se consideran escenarios donde la red cambia durante la ejecución.
- **Escala moderada:** Las conclusiones son válidas principalmente para aplicaciones donde el grafo completo cabe en la memoria principal de una sola máquina.
- **Dominio específico:** Los resultados para redes sociales pueden no extrapolarse directamente a dominios como telecomunicaciones o circuitos electrónicos sin análisis adicional.

12. Experimentacion

En esta sección describimos de manera detallada el proceso de experimentación que realizamos para evaluar y comparar el rendimiento práctico de los algoritmos Dijkstra, Bellman-Ford, BFS y A*. Nuestro objetivo fue analizar su comportamiento empírico en distintos tipos de grafos, considerando no solo su complejidad teórica, sino también su desempeño real en un entorno computacional controlado.

La experimentación fue diseñada para reflejar escenarios representativos de aplicaciones reales, tales como redes sociales, mapas geográficos y grafos aleatorios, permitiéndonos observar cómo influyen las características estructurales del grafo en la eficiencia de cada algoritmo.

12.1. Entorno Experimental

Todos los experimentos fueron desarrollados utilizando el lenguaje de programación Python versión 3.10. Para la creación, manipulación y análisis de los grafos empleamos la librería NetworkX, debido a su amplia adopción en la comunidad académica y su facilidad para generar distintos modelos de redes.

La medición del tiempo de ejecución se realizó mediante la función `perf_counter` del módulo `time`, la cual ofrece una alta precisión para mediciones de rendimiento. El consumo de memoria fue monitoreado observando el uso máximo de memoria RAM durante la ejecución de cada algoritmo.

Las pruebas se ejecutaron en una computadora personal estándar equipada con un procesador de cuatro núcleos y al menos 8 GB de memoria RAM. Si bien los valores absolutos de tiempo pueden variar dependiendo del hardware, consideramos que las comparaciones relativas entre algoritmos son válidas y permiten extraer conclusiones significativas.

12.2. Algoritmos Evaluados

Implementamos y evaluamos cuatro algoritmos clásicos de búsqueda de caminos más cortos, seleccionados por su relevancia teórica y su uso frecuente en aplicaciones prácticas:

- **Dijkstra**: Lo implementamos utilizando una cola de prioridad basada en un *binary heap*. Este algoritmo fue aplicado exclusivamente a grafos con pesos no negativos y sirvió como referencia principal en la comparación experimental.
- **Bellman-Ford**: Empleamos una implementación estándar basada en la relajación iterativa de todas las aristas del grafo. Aunque este algoritmo es más general y permite trabajar con pesos negativos, esperábamos un mayor costo computacional, lo cual fue analizado experimentalmente.
- **BFS (Breadth-First Search)**: Aplicamos este algoritmo únicamente en grafos no ponderados o con pesos unitarios, donde garantiza encontrar el camino más corto en términos del número de aristas recorridas.
- **A***: Implementamos A* como una extensión del algoritmo de Dijkstra, incorporando una heurística de distancia euclidiana cuando los grafos contaban con información espacial, con el objetivo de guiar la búsqueda de manera más eficiente hacia el nodo destino.

En todos los casos cuidamos que las implementaciones fueran comparables entre sí, evitando optimizaciones específicas que pudieran introducir sesgos en los resultados.

12.3. Conjuntos de Datos

Para la evaluación experimental trabajamos con tres categorías de grafos, seleccionadas para representar diferentes contextos de aplicación:

12.3.1. Grafos Sociales

Utilizamos grafos reales disponibles en la librería NetworkX, tales como el grafo del club de karate de Zachary, el grafo de familias florentinas y el grafo de mujeres del sur de Davis. Aunque estos grafos son de tamaño reducido, presentan estructuras reales de redes sociales, como comunidades y distribuciones de grado no uniformes, lo que los hace adecuados para evaluar el comportamiento de los algoritmos en este tipo de redes.

12.3.2. Grafos Geográficos

Para simular redes de calles y entornos espaciales, generamos grafos de malla bidimensional y grafos geográficos aleatorios con nodos distribuidos en el plano. Asignamos a las aristas pesos correspondientes a la distancia euclidiana entre nodos conectados, lo que nos permitió aplicar el algoritmo A* utilizando una heurística admisible y analizar su ventaja frente a Dijkstra.

12.3.3. Grafos Aleatorios Sintéticos

También generamos grafos aleatorios sintéticos utilizando modelos clásicos como Erdős–Rényi, Barabási–Albert y Watts–Strogatz. Estos modelos nos permitieron estudiar el impacto de la densidad de aristas, la presencia de nodos altamente conectados y las propiedades de mundo pequeño en el rendimiento de los algoritmos evaluados.

12.4. Metodología de Medición

Para cada combinación de algoritmo y conjunto de datos seleccionamos pares de nodos origen–destino de manera aleatoria. Cada experimento fue ejecutado diez veces con el fin de reducir el efecto de variaciones del sistema y obtener resultados más estables. Los valores reportados corresponden al promedio de dichas ejecuciones.

Durante la experimentación evaluamos las siguientes métricas:

- **Tiempo de ejecución**: Medimos el tiempo total requerido por cada algoritmo para encontrar el camino más corto entre los nodos seleccionados.
- **Consumo de memoria**: Registramos el uso máximo de memoria RAM durante la ejecución de cada algoritmo.
- **Corrección**: Verificamos que todos los algoritmos exactos calcularan las mismas distancias mínimas para un mismo par de nodos, lo que nos permitió validar la correcta implementación de cada método.

12.5. Reproducibilidad y Control Experimental

Con el objetivo de garantizar la reproducibilidad de los resultados, fijamos semillas aleatorias tanto en la generación de grafos sintéticos como en la selección de los nodos origen y destino. Asimismo, mantuvimos constantes el entorno de ejecución, las versiones de las librerías utilizadas y la configuración experimental durante todas las pruebas realizadas.

Este diseño experimental nos permitió realizar una comparación justa, controlada y objetiva del rendimiento práctico de los algoritmos de caminos más cortos, sentando las bases para el análisis y discusión de los resultados presentados en la siguiente sección.

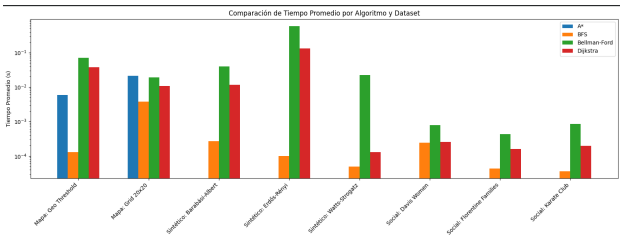

Figura 1: Comparación del tiempo de ejecución de los algoritmos evaluados

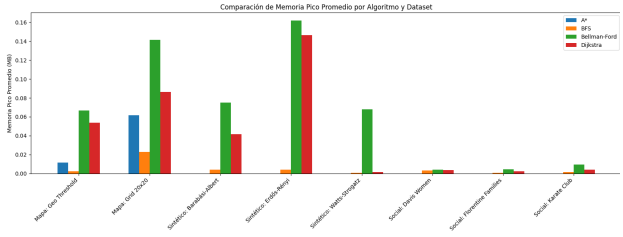

Figura 2: Comparación del uso de memoria en la ejecución de los algoritmos evaluados

Referencias

[1] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. Numerische Mathematik, 1, 269-271.

[2] Atalig, S., Hickerson, A., Srivastav, A., Zheng, T., Chrobak, M. (2024). Lower Bounds for Adaptive Relaxation-Based Algorithms for Single-Source Shortest Paths. Proceedings of ISAAC 2024, LIPIcs.

[3] Karczmarz, A., Nadara, W., Sokolowski, M. (2023). Exact Shortest Paths with Rational Weights on the Word RAM. arXiv preprint arXiv:2311.03321.

[4] Duan, R., Mao, J., Shu, X., Yin, L. (2023). A randomized algorithm for single-source shortest path on undirected real-weighted graphs. Proceedings of FOCS 2023, 484-492.

[5] Duan, R., Mao, J., Mao, X., Shu, X., Yin, L. (2025). Breaking the Sorting Barrier for Directed Single-Source Shortest Paths. arXiv preprint arXiv:2504.17033.

[6] Bernstein, A., Nanongkai, D., Wulff-Nilsen, C. (2022). Negative-Weight Single-Source Shortest Paths in Near-Linear Time. Proceedings of FOCS 2022, 600-611.

[7] Huang, Y., Jin, P., Quanrud, K. (2024). Faster single-source shortest paths with negative real weights via proper hop distance. arXiv preprint.

[8] Elmasry, A. (2024). Breaking the Bellman-Ford Shortest-Path Bound. arXiv preprint arXiv:2410.23383.

[9] Haeupler, B., Hladik, R., Rozhon, V., Tarjan, R. E., Tetek, J. (2024). Universal optimality of Dijkstra via beyond-worst-case heaps. Proceedings of FOCS 2024.

[10] Cassis, A., Karrenbauer, A., Nusser, A., Rinaldi, P. L. (2024). Algorithm Engineering of SSSP with Negative Edge Weights. Proceedings of SEA 2024, LNCS 14751, 3-20.

[11] Karczmarz, A., Sankowski, P. (2024). Fully Dynamic Shortest Paths and Reachability in Sparse Digraphs. Proceedings of ICALP 2024, LIPIcs.

[12] Chen, L., Kyng, R., Liu, Y. P., Peng, R., Probst Gutenberg, M., Sachdeva, S. (2022). Maximum flow and minimum-cost flow in almost-linear time. Proceedings of FOCS 2022, 612-623.

[13] Grappe, R., et al. (2023). Exact methods for multiple-pair shortest path queries in sparse graphs. Journal of Experimental Algorithmics.

[14] Probst Gutenberg, M., et al. (2020). Deterministic incremental algorithms for single-source shortest paths. Proceedings of SODA 2020.

[15] Valko, D., Paranjpe, R., Gómez, J. M. (2025). Outperforming Dijkstra on Sparse Graphs: The Lightning Network Use Case. arXiv preprint arXiv:2509.13448.