Aplicación de Algoritmos de Caminos Mínimos: Dijkstra y Bellman-Ford en Python

J. C. Barrera Guevara , D. A. Machado Tovar, J. G. Delgado
Facultad de Ciencias Básicas e Ingeniería, Universidad de los Llanos
Villavicencio, Colombia
jc.bguevara@unillanos.edu.co
damachado@unillanos.edu.co
jg.delgado@unillanos.edu.co

I. INTRODUCCIÓN

La teoría de grafos constituye un fundamento esencial para la modelación matemática de sistemas complejos, especialmente en áreas como optimización, análisis de redes, ingeniería de sistemas y ciencias de la computación. Su capacidad para representar relaciones de conectividad y flujos entre entidades permite abordar problemas de ruteo, transporte, comunicaciones y asignación de recursos.

El objetivo de esta práctica fue desarrollar una aplicación en Python que permitiera la construcción, validación y representación visual de grafos ponderados, así como la implementación de dos algoritmos fundamentales para el cálculo de rutas de costo mínimo: **Dijkstra** y **Bellman-Ford**. Con ellos se buscó analizar diferencias de funcionamiento, restricciones teóricas y desempeño computacional, integrando conceptos de teoría de grafos con un enfoque práctico de ingeniería.

II. REFERENTE TEÓRICO

A. Definición general

Un grafo se define como una estructura compuesta por un conjunto de vértices o nodos y un conjunto de aristas que representan las relaciones entre ellos. Cuando las aristas carecen de orientación se habla de un grafo no dirigido, mientras que en los grafos dirigidos cada arista posee una dirección que determina el sentido de la relación.

B. Representación

La matriz de adyacencia y la lista de adyacencia constituyen las representaciones más utilizadas. La primera es conveniente para operaciones matriciales y validación estructural; la segunda es más eficiente para grafos dispersos debido a su menor consumo de memoria.

C. Conceptos estructurales relevantes

Entre las propiedades necesarias para el análisis de grafos se encuentran el concepto de camino, ciclo, conectividad y completitud. Estas propiedades permiten evaluar la estructura y estabilidad del sistema modelado.

D. Algoritmo de Dijkstra

El algoritmo de Dijkstra encuentra el camino de costo mínimo desde un nodo origen hacia todos los demás nodos en grafos ponderados con **pesos no negativos**. Su funcionamiento se basa en:

1. Inicialización de distancias (0 al nodo origen, infinito a los demás).
2. Selección del vértice no visitado con menor distancia provisional.
3. Relajación de las distancias de sus vecinos.
4. Repetición del proceso hasta visitar todos los nodos alcanzables.

Dijkstra es eficiente y garantiza optimalidad bajo la restricción de no negatividad en los pesos.

E. Algoritmo de Bellman-Ford

Bellman-Ford extiende la resolución del problema de caminos mínimos permitiendo **pesos negativos**, con la limitación de que no existan ciclos negativos alcanzables desde el origen. Su operación se basa en la relajación sucesiva de todas las aristas del grafo durante |V|-1 iteraciones, donde V es el número de vértices.

El algoritmo:

1. Inicializa las distancias como en Dijkstra.
2. Recorre todas las aristas, actualizando las distancias si un camino más corto es encontrado.
3. Tras las iteraciones, ejecuta una fase adicional para detectar ciclos de peso negativo.

Bellman-Ford es menos eficiente que Dijkstra en términos de complejidad temporal, pero más general y robusto para grafos con pesos negativos.

III. METODOLOGÍA

El proyecto se implementó en Python 3.11 mediante un enfoque modular. La estructura final del sistema incluye:

graph_model.py: validación de la matriz de adyacencia, construcción de listas de adyacencia y ejecución de los algoritmos de Dijkstra y Bellman-Ford. Dijkstra utiliza un heap de prioridad, mientras que Bellman-Ford opera mediante relajaciones iterativas.

graph_drawer.py: cálculo geométrico de posiciones y representación gráfica de aristas mediante trayectorias y curvas Bézier.

graphgui.py: interfaz gráfica en Tkinter para ingreso de datos, construcción del grafo, ejecución de los algoritmos y visualización de resultados.

main.py: archivo de coordinación general entre módulos.

A. Etapas experimentales

- Definición del número de vértices, etiquetas y tipo de grafo.
- Construcción de la matriz de adyacencia y asignación de pesos.
- Validación de coherencia estructural.
- Visualización del grafo mediante distribución radial.
- Ejecución del algoritmo de Dijkstra.
- Ejecución del algoritmo de Bellman-Ford, incluyendo detección de ciclos negativos.
- Resaltado visual del camino mínimo encontrado por cada algoritmo.

IV. RESULTADOS Y ANÁLISIS

El sistema representó adecuadamente grafos dirigidos y no dirigidos, validando la coherencia de la matriz de adyacencia y proporcionando una representación gráfica clara y estable.

El algoritmo de **Dijkstra** mostró comportamiento óptimo en grafos con pesos no negativos, obteniendo rutas mínimas consistentes con la teoría y visualizándolas adecuadamente en la interfaz.

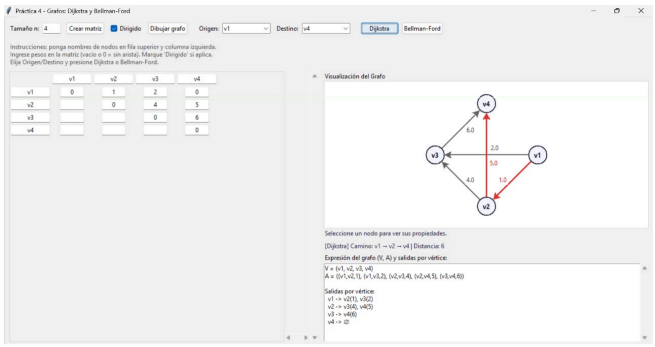

Figura 1. Implementación del algoritmo de Dijkstra.

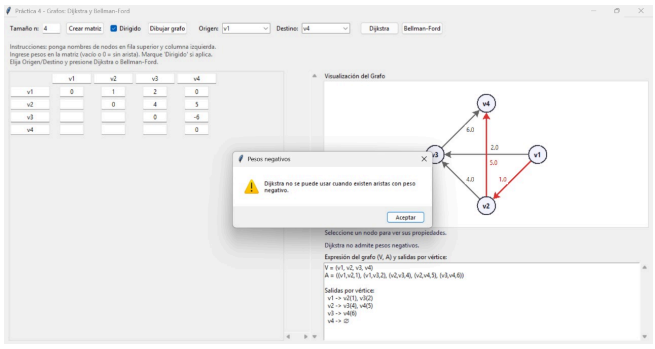

Figura 2. Validaciones para pesos no positivos.

El algoritmo de **Bellman-Ford** permitió procesar exitosamente grafos con pesos negativos y detectar escenarios donde se presentaban ciclos de peso negativo. En tales casos, el sistema emitió el correspondiente mensaje informando la imposibilidad de definir caminos mínimos.

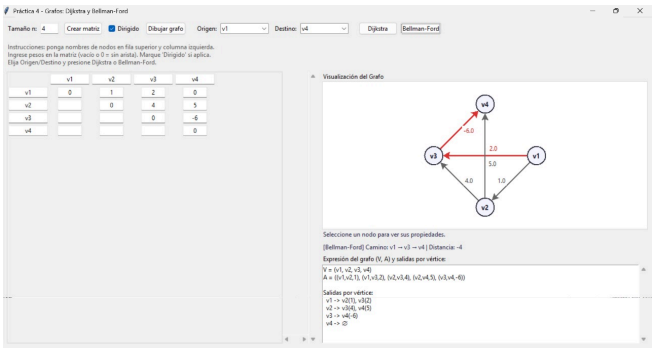

Figura 3. Implementación del algoritmo Bellman-Ford.

Entre las observaciones más relevantes se destacan:

- Coherencia en la validación del grafo.
- Precisión geométrica en la representación visual.
- Capacidad de manejo de pesos negativos sin comprometer la estabilidad del análisis.
- Comportamiento diferenciado de ambos algoritmos, verificando sus restricciones teóricas.

Estas evidencias confirman la adecuación del sistema para el análisis de rutas óptimas mediante algoritmos clásicos de teoría de grafos.

V. CONCLUSIONES

La práctica permitió integrar el formalismo teórico de la teoría de grafos con su aplicación computacional en Python. La incorporación conjunta de **Dijkstra y Bellman-Ford** permitió comparar sus capacidades, restricciones y eficiencia, reforzando la comprensión del problema de caminos mínimos en grafos ponderados.

El sistema desarrollado demostró ser modular, correcto y funcional, y la interfaz gráfica proporcionó un apoyo visual adecuado para la verificación de resultados. Se concluye que los grafos son herramientas fundamentales en la modelación de sistemas reales y que Python constituye una plataforma idónea para el prototipado y experimentación en optimización y análisis de redes.

REFERENCIAS

[1] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[2] R. Bellman, *Dynamic Programming*, Princeton University Press, 1957.

[3] L. R. Ford Jr., “Network flow theory,” *RAND Corporation*, Paper P-923, 1956.