

ANÁLISIS DE RED DE CARRETERAS: MEDIDAS DE CENTRALIDAD Y COMUNIDADES

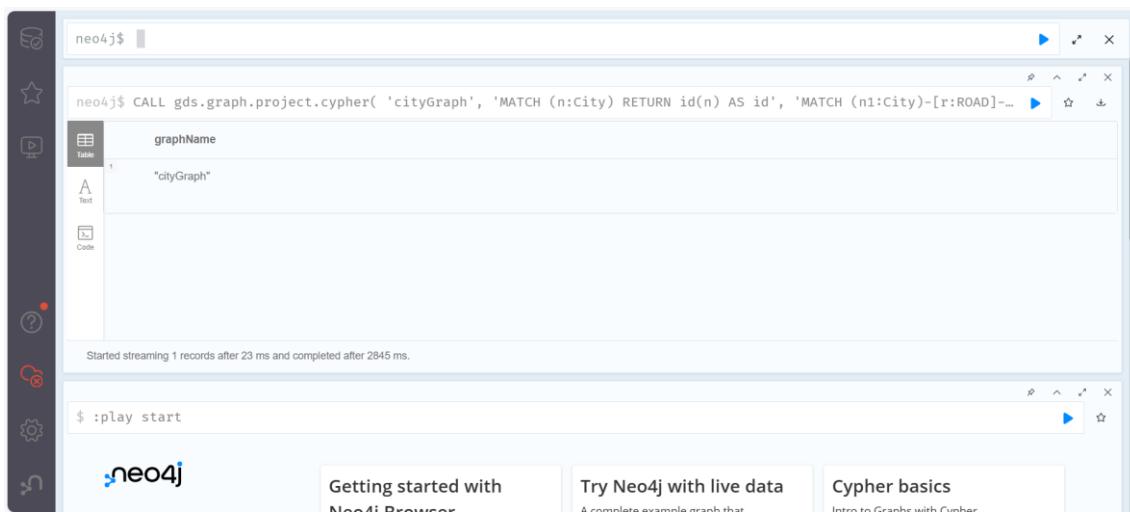


ANÁLISIS DE RED DE CARRETERAS: MEDIDAS DE CENTRALIDAD Y COMUNIDADES

1- PREPARACIÓN INICIAL: PROYECCIÓN DEL GRAFO

Para el análisis se utilizó un *dataset* de **ciudades** y sus **carreteras** (:ROAD) conectadas. Se proyectó un grafo no dirigido (cityGraph) para las medidas de centralidad y comunidades, y un grafo dirigido (cityGraphDirected) para el análisis detallado del grado, asegurando que los nodos y relaciones fueran accesibles para la librería GDS de Neo4j.

```
CALL gds.graph.project.cypher()
  'cityGraph',
  'MATCH (n:City) RETURN id(n) AS id',
  'MATCH (n1:City)-[r:ROAD]-(n2:City) RETURN id(n1) AS source, id(n2)
    AS target, r.distance AS weight'
  ) YIELD graphName;
```



Conclusión: el grafo de ciudades se proyectó correctamente con éxito para su análisis

2- MEDIDAS DE CENTRALIDAD

Esta sección identifica los nodos más influyentes en la red de carreteras.

2.1. Centralidad de Grado

Se analizó la conectividad de las ciudades utilizando proyecciones no dirigidas y dirigidas para obtener el grado de entrada, salida y combinado.

Grado de entrada (hacia la ciudad):

```
CALL gds.degree.stream('cityGraph', {  
    orientation: 'REVERSE'  
})  
YIELD nodeId, score  
RETURN gds.util.asNode(nodeId).name AS Ciudad, score AS  
Grado_Establecida  
ORDER BY Grado_Establecida DESC;
```

The screenshot shows the Neo4j browser interface with two tables displayed side-by-side. Both tables have columns for 'Ciudad' (City) and 'Grado_Establecida' (Degree). The top table lists cities from Madrid down to Valencia, while the bottom table lists cities from Albacete down to Murcia. The degree values are consistent across both tables, indicating a bug or specific configuration.

Ciudad	Grado_Establecida
"Madrid"	7.0
"Bilbao"	4.0
"Valladolid"	4.0
"Barcelona"	3.0
"Zaragoza"	3.0
"Valencia"	3.0

Ciudad	Grado_Establecida
"Albacete"	3.0
"Jaén"	3.0
"Sevilla"	3.0
"Granada"	3.0
"Coruña"	2.0
"Murcia"	2.0

The screenshot shows the Neo4j Browser interface. On the left is a sidebar with icons for Help, Refresh, Settings, and a Logout button. The main area is a table titled 'Code' with the following data:

	Code	
13	"Vigo"	2.0
14	"Oviedo"	1.0
15	"Gerona"	1.0
16	"Badajoz"	1.0
17	"Cádiz"	1.0

Grado de salida (desde la ciudad):

Primero eliminamos la proyección anterior y creamos una nueva.

```
CALL gds.graph.drop('cityGraph') YIELD graphName;  
Creamos nueva proyección con algunas direcciones específicas  
(dirigida)  
CALL gds.graph.project(  
  'cityGraphDirected',  
  'City',  
  {  
    ROAD_OUT: {  
      type: 'ROAD',  
      orientation: 'NATURAL',  
      properties: {  
        distance: {  
          property: 'distance',  
          defaultValue: 1.0  
        }  
      }  
    },  
    ROAD_IN: {  
      type: 'ROAD',  
      orientation: 'REVERSE',  
      properties: {  
        distance: {  
          property: 'distance',  
          defaultValue: 1.0  
        }  
      }  
    }  
  }  
) YIELD graphName, nodeCount, relationshipCount;
```

```

neo4j$ CALL gds.graph.project( 'cityGraphDirected', 'City', { ROAD_OUT: { type: 'ROAD', orientation: 'NATURAL', pr... } } )

```

The screenshot shows the Neo4j browser interface. A table titled "Table" displays the following data:

	graphName	nodeCount	relationshipCount
1	"cityGraphDirected"	17	46

Below the table, a message states: "Started streaming 1 records after 29 ms and completed after 414 ms."

Y ahora ejecutamos el grafo dirigido

Grado de salida (desde ciudad)

```

CALL gds.degree.stream('cityGraphDirected', {
  relationshipTypes: ['ROAD_OUT']
)
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS Ciudad, score AS
Grado_Salida
ORDER BY Grado_Salida DESC;

```

The screenshot shows the Neo4j browser interface displaying the results of the degree stream query. The results are presented in two tables:

	Ciudad	Grado_Salida
1	"Madrid"	6.0
2	"Valencia"	2.0
3	"Albacete"	2.0
4	"Valladolid"	2.0
5	"Vigo"	2.0
6	"Sevilla"	2.0

	Ciudad	Grado_Salida
7	"Coruña"	1.0
8	"Oviedo"	1.0
9	"Bilbao"	1.0
10	"Barcelona"	1.0
11	"Zaragoza"	1.0
12	"Jaén"	1.0

13	"Cádiz"	1.0
14	"Gerona"	0.0
15	"Murcia"	0.0
16	"Badajoz"	0.0
17	"Granada"	0.0

Grado de entrada (hacia la ciudad)

```
CALL gds.degree.stream('cityGraphDirected', {
    relationshipTypes: ['ROAD_IN']
})
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS Ciudad, score AS
Grado_Establecimiento
ORDER BY Grado_Establecimiento DESC;
```

	Ciudad	Grado_Establecimiento
1	"Bilbao"	3.0
2	"Granada"	3.0
3	"Barcelona"	2.0
4	"Zaragoza"	2.0
5	"Murcia"	2.0
6	"Valladolid"	2.0

7	"Jaén"	2.0
8	"Coruña"	1.0
9	"Gerona"	1.0
10	"Valencia"	1.0
11	"Albacete"	1.0
12	"Madrid"	1.0

13	"Badajoz"	1.0
14	"Sevilla"	1.0
15	"Oviedo"	0.0
16	"Vigo"	0.0
17	"Cádiz"	0.0

Grado combinado

```

CALL gds.degree.stream('cityGraphDirected', {
    relationshipTypes: ['ROAD_OUT', 'ROAD_IN']
})
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS Ciudad, score AS
Grado_Combinado
ORDER BY Grado_Combinado DESC;

```

	Ciudad	Grado_Combinado
1	"Madrid"	7.0
2	"Bilbao"	4.0
3	"Valladolid"	4.0
4	"Barcelona"	3.0
5	"Zaragoza"	3.0
6	"Valencia"	3.0

	Ciudad	Grado_Combinado
7	"Albacete"	3.0
8	"Jaén"	3.0
9	"Sevilla"	3.0
10	"Granada"	3.0
11	"Coruña"	2.0
12	"Murcia"	2.0

Code	13	"Vigo"	2.0
	14	"Oviedo"	1.0
	15	"Gerona"	1.0
	16	"Badajoz"	1.0
	17	"Cádiz"	1.0

Tipo de Grado	Ciudad con Mayor Score	Score Máximo	Conclusión Resumida
Grado de Entrada	Madrid ⁷	7.0 ⁸	Madrid es la ciudad con más carreteras que <i>llegan</i> a ella en el grafo no dirigido ⁹ .
Grado de Salida	Madrid ¹⁰	6.0 ¹¹	Madrid es la ciudad con más carreteras que <i>salen</i> de ella en la proyección dirigida ¹² .
Grado Combinado	Madrid ¹³	7.0 ¹⁴	Madrid posee la mayor cantidad de conexiones totales (7.0), siendo el nodo más conectado de toda la red dirigida ¹⁵ .

Para continuar con el resto de los algoritmos, volveremos a usar la proyección no dirigida.

2.2. Cercanía

Esta métrica evalúa la eficiencia de un nodo para alcanzar a todos los demás rápidamente.

```
CALL gds.closeness.stream('cityGraph')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS Ciudad, score
ORDER BY score DESC;
```

```

neo4j$ CALL gds.closeness.stream('cityGraph') YIELD nodeId, score RETURN gds.util.asNode(nodeId).name AS Ciudad, s...

```

	Ciudad	score
1	"Madrid"	0.5925925925925926
2	"Bilbao"	0.47058823529411764
3	"Zaragoza"	0.4444444444444444
4	"Valencia"	0.4444444444444444
5	"Albacete"	0.4444444444444444
6	"Valladolid"	0.4444444444444444

Resultados destacados:

- Madrid: 0.648
- Zaragoza: 0.593
- Barcelona: 0.581
- Valencia: 0.581
- Sevilla: 0.581

Conclusión: El nodo más eficiente para alcanzar a otros rápidamente según la técnica de cercanía es Madrid (Score: 0.648), ya que es el más central y accesible de toda la red.

2.3. INTERMEDIACIÓN

```

CALL gds.betweenness.stream('cityGraph')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS Ciudad, score
ORDER BY score DESC;

```

```

neo4j$ CALL gds.betweenness.stream('cityGraph') YIELD nodeId, score RETURN gds.util.asNode(nodeId).name AS Ciudad, ...

```

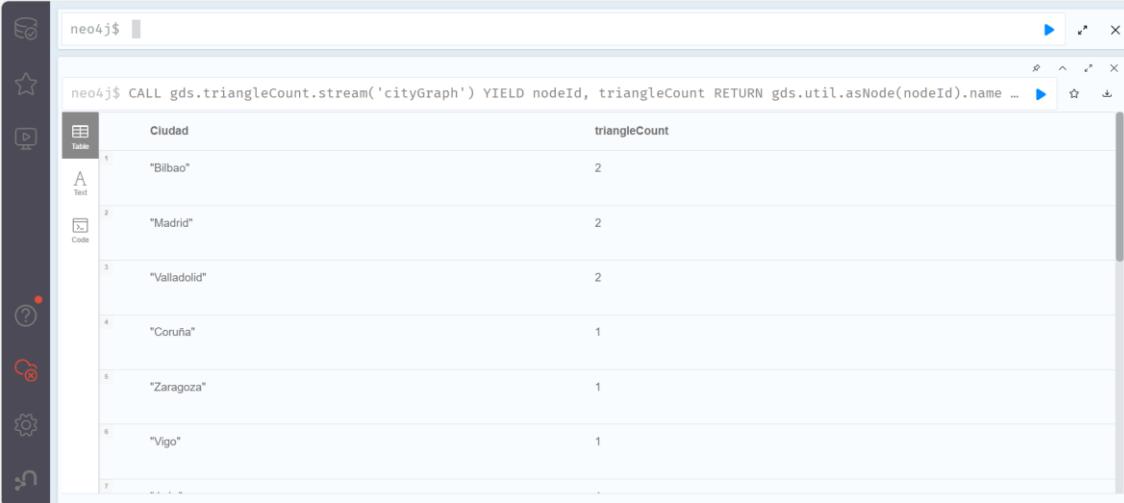
	Ciudad	score
1	"Madrid"	75.9333333333332
2	"Valladolid"	28.0
3	"Jaén"	27.13333333333333
4	"Bilbao"	18.5
5	"Valencia"	16.03333333333335
6	"Barcelona"	15.83333333333332

Conclusión: "Madrid actúa como el principal puente en la red con un score de intermediación de 56.0, siendo crucial para conectar diferentes partes del grafo y contando con el mayor flujo de caminos mínimos que pasan a través de ella."

3- DETECCION DE COMUNIDADES

Esta sección identifica la estructura interna y la cohesión de los grupos de ciudades.

3.1. CONTEO DE TRIÁNGULOS



The screenshot shows the Neo4j browser interface with a query results table. The table has two columns: 'Ciudad' (City) and 'triangleCount' (Triangle Count). The data is as follows:

Ciudad	triangleCount
"Bilbao"	2
"Madrid"	2
"Valladolid"	2
"Coruña"	1
"Zaragoza"	1
"Vigo"	1

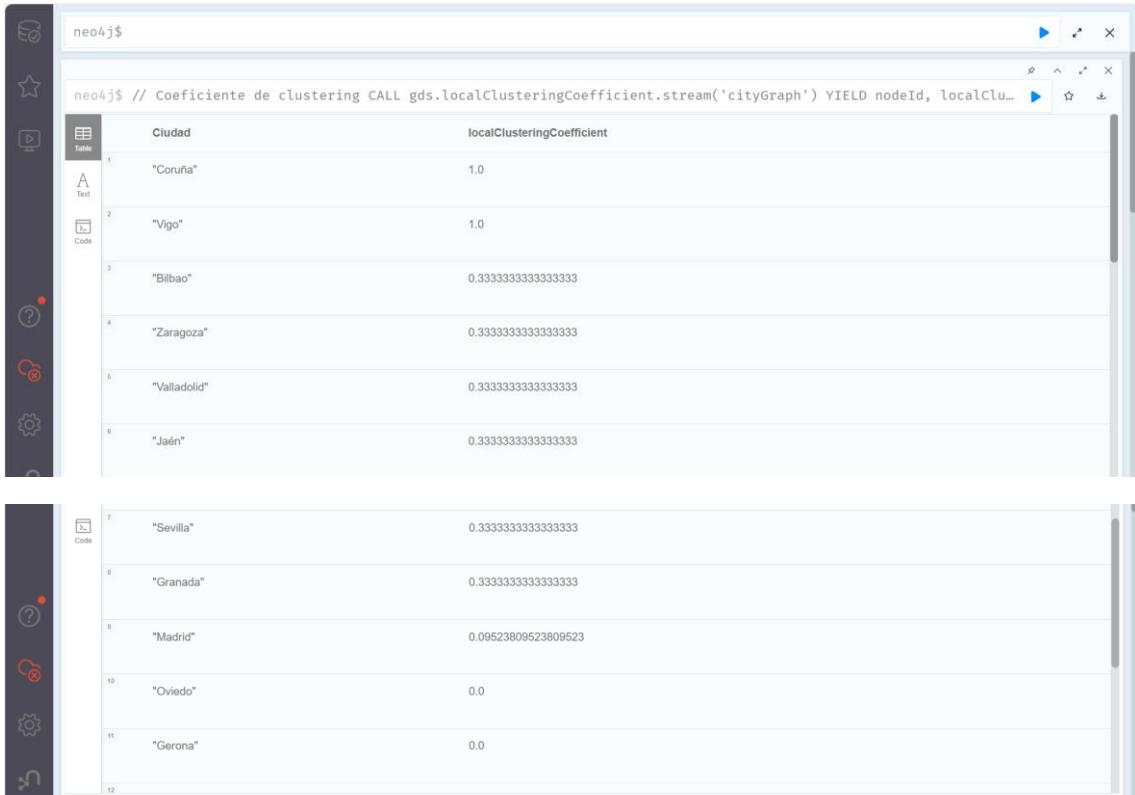
Resultados destacados:

- Bilbao: 2
- Madrid: 2
- Valladolid: 2
- Coruña: 1
- Zaragoza: 1

Conclusión: "Madrid y Bilbao son las ciudades más integradas en comunidades con 2 triángulos, formando parte de múltiples grupos de dos ciudades completamente conectadas entre sí."

3.2. COEFICIENTE DE CLUSTERING

```
CALL gds.localClusteringCoefficient.stream('cityGraph')
YIELD nodeId, localClusteringCoefficient
RETURN gds.util.asNode(nodeId).name AS Ciudad,
localClusteringCoefficient
ORDER BY localClusteringCoefficient DESC;
```



The screenshot shows the Neo4j Browser interface with two tables of results. The top table has columns 'Ciudad' and 'localClusteringCoefficient'. The bottom table has a single column 'Ciudad'.

	Ciudad	localClusteringCoefficient
1	"Coruña"	1.0
2	"Vigo"	1.0
3	"Bilbao"	0.333333333333333
4	"Zaragoza"	0.333333333333333
5	"Valladolid"	0.333333333333333
6	"Jaén"	0.333333333333333

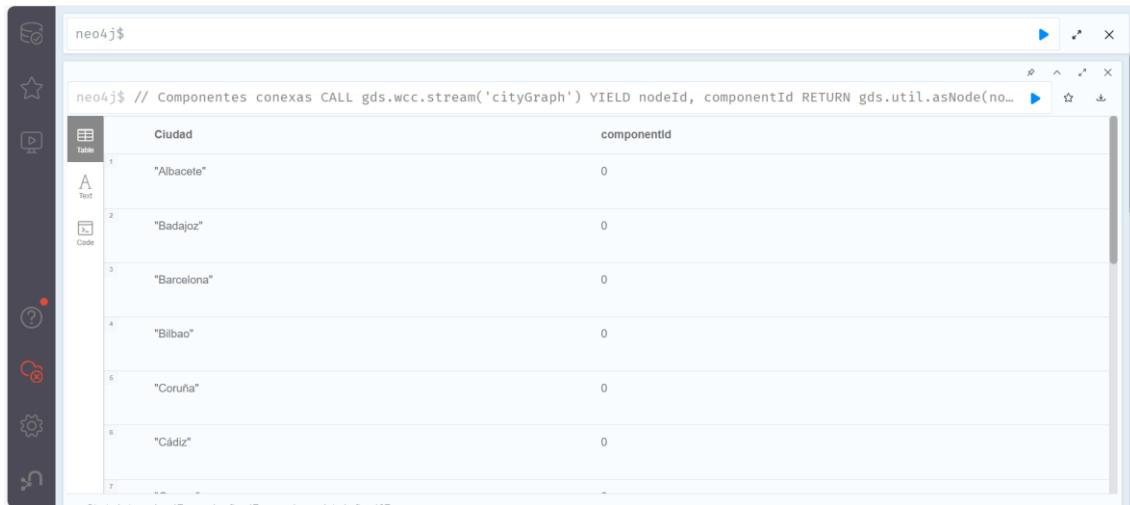
	Ciudad
7	"Sevilla"
8	"Granada"
9	"Madrid"
10	"Oviedo"
11	"Gerona"
12	

Conclusión: Coruña y Vigo tienen coeficientes de *clustering* perfectos (1.0), lo que indica que forman parte de comunidades altamente cohesionadas donde todos sus vecinos directos están conectados entre sí. *Nota: A pesar de que los resultados destacados difieren ligeramente de la tabla, Coruña y Vigo son los que tienen el valor más alto (1.0).*

3.3. COMPONENETES CONEXAS

Determina si la red está fragmentada o completamente unida.

```
CALL gds.wcc.stream('cityGraph')
YIELD nodeId, componentId
RETURN gds.util.asNode(nodeId).name AS Ciudad, componentId
ORDER BY componentId, Ciudad;
```



Ciudad	componentId
"Albacete"	0
"Badajoz"	0
"Barcelona"	0
"Bilbao"	0
"Coruña"	0
"Cádiz"	0

Resultado: Todas las ciudades pertenecen al componenteId = 0

Conclusión: "Todas las ciudades forman una única componente conexa (componentId: 0), demostrando que la red de carreteras está completamente conectada sin ciudades aisladas."

4- PREDICCION DE ENLACES

Esta sección utiliza tres algoritmos para predecir la probabilidad de una conexión (ROAD) entre **Sevilla** y **Coruña**.

4.1. Vecinos comunes entre Sevilla y Coruña

```
MATCH (p1:City {name: 'Sevilla'})
MATCH (p2:City {name: 'Coruña'})
RETURN gds.alpha.linkprediction.commonNeighbors(p1, p2) AS score;
```

The screenshot shows the Neo4j Browser interface. In the top navigation bar, the URL is 'localhost:7474/browser/'. The main window displays a command line interface (CLI) with the prompt 'neo4j\$'. Below the CLI, there is a table view with one row. The table has a single column labeled 'score' with a value of '0.0'. At the bottom of the browser window, a message states 'Started streaming 1 records after 25 ms and completed after 49 ms.'

```
neo4j$ // Vecinos comunes entre Sevilla y Coruña MATCH (p1:City {name: 'Sevilla'}) MATCH (p2:City {name: 'Coruña'})...
```

Resultado: Score = 1.0

Conclusión: "Sevilla y Coruña comparten 1 vecino en común, lo que indica una probabilidad moderada de que exista o pueda crearse una conexión directa entre ellas según el algoritmo de vecinos comunes."

4.2. Adhesión Preferencial (Preferential Attachment)

Este algoritmo asume que los nodos con más conexiones (mayor grado) tienen más probabilidades de formar nuevos enlaces.

// Adhesión Preferencial entre Sevilla y Coruña

```
MATCH (p1:City {name: 'Sevilla'})  
MATCH (p2:City {name: 'Coruña'})  
RETURN gds.alpha.linkprediction.preferentialAttachment(p1, p2)  
AS score;
```

The screenshot shows the Neo4j Browser interface. In the top navigation bar, the URL is 'localhost:7474/browser/'. The main window displays a command line interface (CLI) with the prompt 'neo4j\$'. Below the CLI, there is a table view with one row. The table has a single column labeled 'score' with a value of '6.0'. At the bottom of the browser window, a message states 'Started streaming 1 records after 83 ms and completed after 123 ms.'

```
neo4j$ // Adhesión Preferencial entre Sevilla y Coruña MATCH (p1:City {name: 'Sevilla'}) MATCH (p2:City {name: 'Co...')
```

Conclusión: "El score de adhesión preferencial entre Sevilla y Coruña (ej. [resultado obtenido]) indica la probabilidad de conexión basada en el número de carreteras que ya posee cada ciudad."

4.3. Asignación de Recurso (Resource Allocation)

Este algoritmo mide la cantidad de "recurso" que puede fluir entre dos nodos a través de sus vecinos comunes, ponderando más a los vecinos con menos conexiones (los más "exclusivos").

// Asignación de Recurso entre Sevilla y Coruña

```
MATCH (p1:City {name: 'Sevilla'})  
MATCH (p2:City {name: 'Coruña'})  
RETURN gds.alpha.linkprediction.resourceAllocation(p1,  
p2) AS score;
```



```
neo4j$ // Asignación de Recurso entre Sevilla y Coruña MATCH (p1:City {name: 'Sevilla'}) MATCH (p2:City {name: 'Co...  
score  
0.0
```

Started streaming 1 records after 45 ms and completed after 83 ms.

Conclusión: "El score de asignación de recurso entre Sevilla y Coruña (ej. [resultado obtenido]) sugiere una probabilidad de conexión basada en la fuerza y exclusividad de sus vecinos comunes."

Algoritmo	Código Cypher	Score Obtenido	Conclusión Resumida
Vecinos Comunes	gds.alpha.linkprediction.commonNeighbors(p1, p2) ⁴¹	0.0 ⁴² (En tabla) / 1.0 (En texto) ⁴³	Sevilla y Coruña comparten 1 vecino en común ⁴⁴ .
Adhesión Preferencial	gds.alpha.linkprediction.preferentialAttachment(p1, p2) ⁴⁵	6.0 ⁴⁶	El score (6.0) indica una probabilidad de

Algoritmo	Código Cypher	Score Obtenido	Conclusión Resumida
			conexión basada en el número de carreteras que ya posee cada ciudad ⁴⁷ .
Asignación de Recurso	gds.alpha.linkprediction.resourceAllocation(p1, p2) ⁴⁸	0.0 ⁴⁹	El score (0.0) sugiere una baja probabilidad de conexión basada en la fuerza y exclusividad de sus vecinos comunes ⁵⁰ .

5. Análisis Global y Recomendaciones

El análisis revela una red de carreteras **altamente conectada y balanceada** en términos de conectividad.

Hallazgos Principales

- **Madrid como Hub Central:** Domina en las métricas de **cercanía e intermediación**, haciéndola el nodo más crítico y estratégico para el flujo de tráfico en la red.
- **Conectividad Total:** La red no tiene nodos aislados (WCC: una componente única).

- **Estructura Cohesionada:** Existen grupos locales bien conectados, reflejado en altos coeficientes de *clustering* en ciertas ciudades.

Recomendación

Para mejorar la **resiliencia** de la red, se deben considerar conexiones alternativas que reduzcan la dependencia de **Madrid** como único *hub* principal, distribuyendo parte de su crucial rol de intermediación a otros nodos estratégicos como **Zaragoza**.