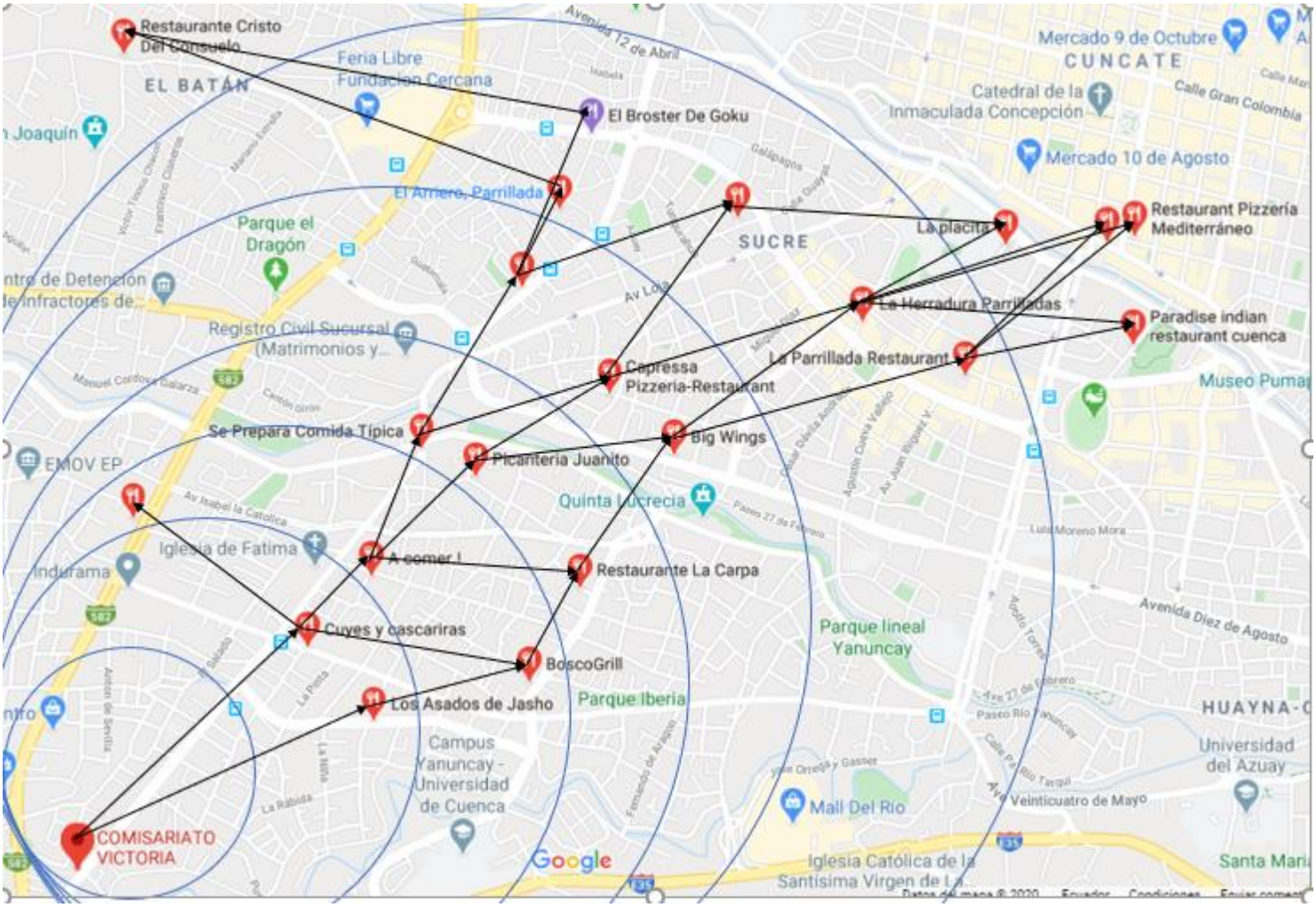
INFORME INTELIGENCIA ARTIFICIAL

ALGORITMO DE BUSQUEDA HEURISTICA –

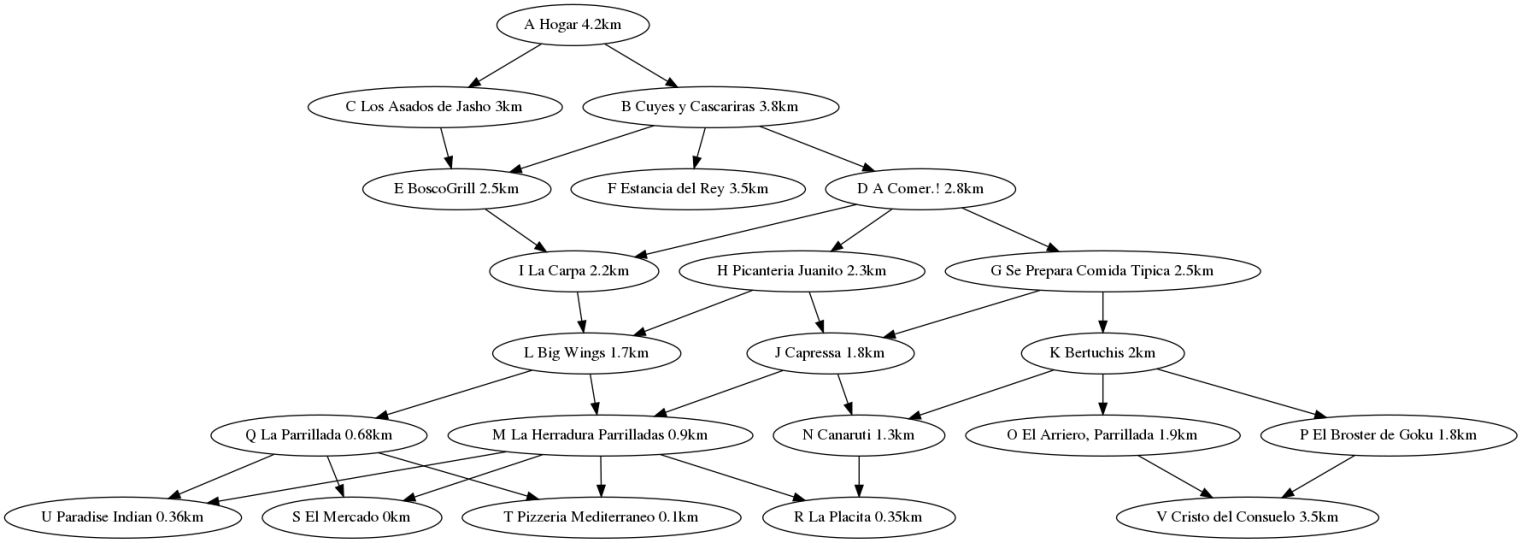
HILL CLIMBING

Alumno: Jordan Murillo

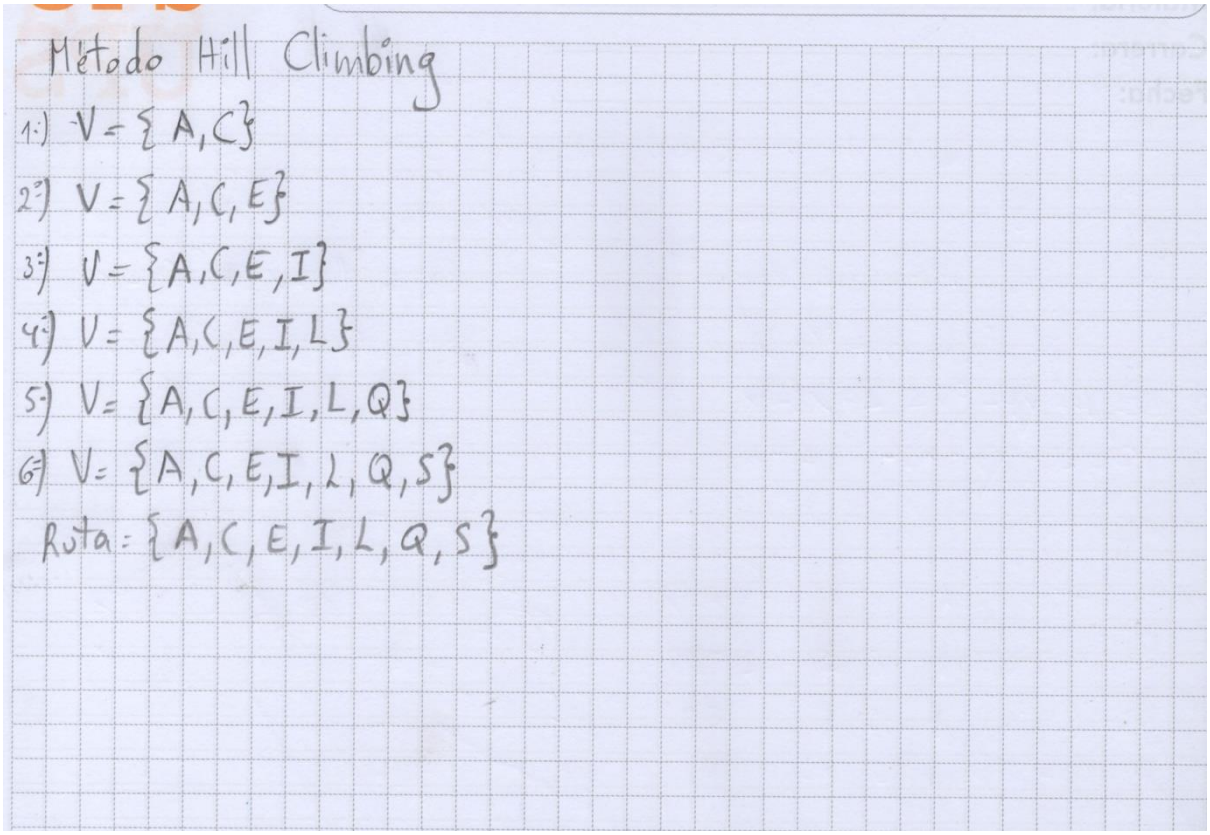
Imagen de los nodos a utilizar – Restaurantes



Dibujo de los nodos – Uso de Python



Proceso a mano de A*



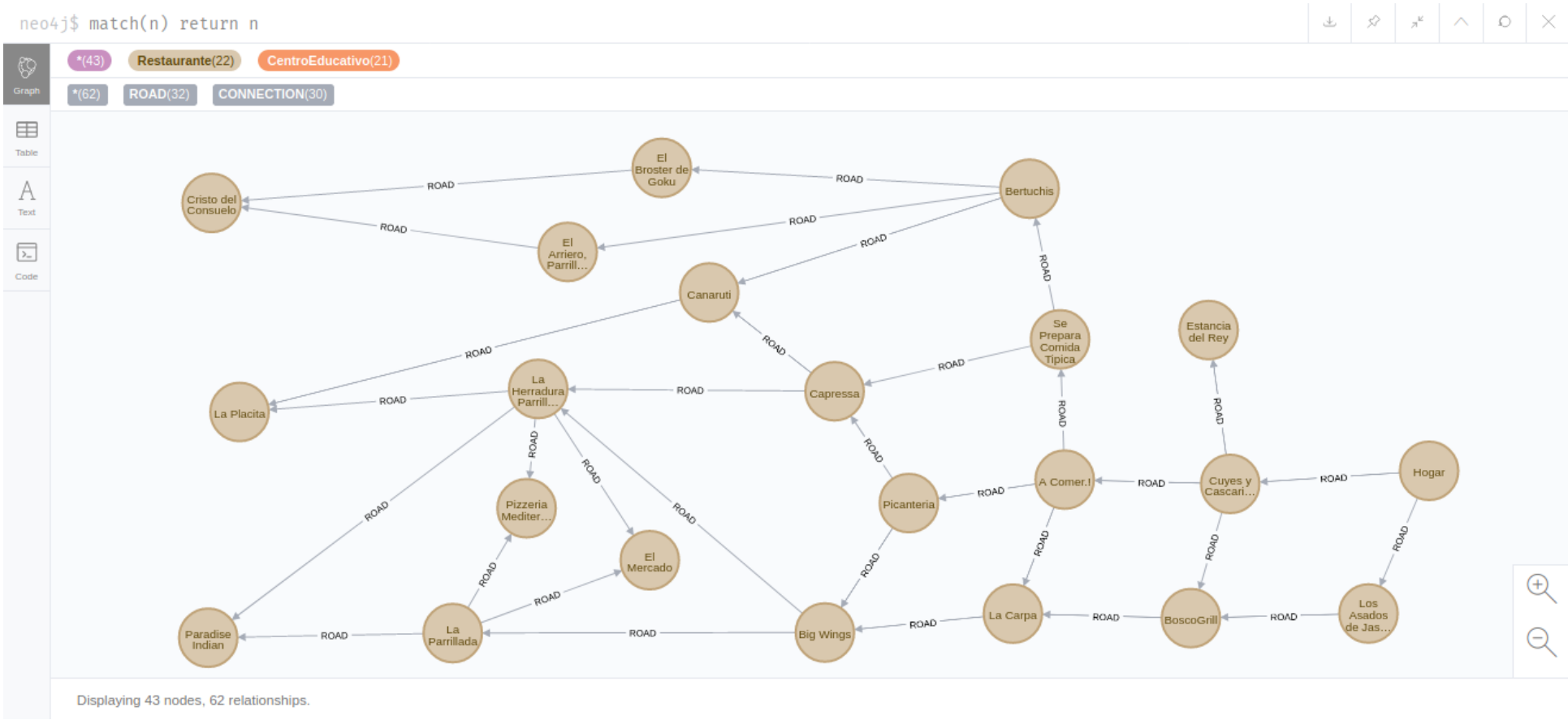
Desarrollo en Neo4j – Shortest Path

a. Código de Neo4j para la creación de los grafos con sus relaciones.

```
CREATE (a:Restaurante {name: 'Hogar'}),
       (b:Restaurante {name: 'Cuyes y Cascarinas'}),
       (c:Restaurante {name: 'Los Asados de Jasho'}),
       (d:Restaurante {name: 'A Comer.!'}),
       (e:Restaurante {name: 'BoscoGrill'}),
       (f:Restaurante {name: 'Estancia del Rey'}),
       (g:Restaurante {name: 'Se Prepara Comida Tipica'}),
       (h:Restaurante {name: 'Picanteria Juanito'}),
       (i:Restaurante {name: 'La Carpa'}),
       (j:Restaurante {name: 'Capressa'}),
       (k:Restaurante {name: 'Bertuchis'}),
       (l:Restaurante {name: 'Big Wings'}),
       (m:Restaurante {name: 'La Herradura Parrilladas'}),
       (n:Restaurante {name: 'Canaruti'}),
       (o:Restaurante {name: 'El Arriero, Parrillada'}),
       (p:Restaurante {name: 'El Broster de Goku'}),
       (q:Restaurante {name: 'La Parrillada'}),
       (r:Restaurante {name: 'La Placita'}),
       (s:Restaurante {name: 'El Mercado'}),
       (t:Restaurante {name: 'Pizzeria Mediterraneo'}),
       (u:Restaurante {name: 'Paradise Indian'}),
       (v:Restaurante {name: 'Cristo del Consuelo'}),
       (a)-[:ROAD {cost: 1.2}]->(b),
       (a)-[:ROAD {cost: 1.4}]->(c),
       (b)-[:ROAD {cost: 0.6}]->(d),
       (b)-[:ROAD {cost: 1}]->(e),
       (b)-[:ROAD {cost: 0.9}]->(f),
       (c)-[:ROAD {cost: 0.85}]->(e),
       (d)-[:ROAD {cost: 0.6}]->(g),
       (d)-[:ROAD {cost: 0.75}]->(h),
       (d)-[:ROAD {cost: 0.85}]->(i),
       (e)-[:ROAD {cost: 0.9}]->(i),
       (g)-[:ROAD {cost: 1}]->(j),
       (g)-[:ROAD {cost: 0.75}]->(k),
       (h)-[:ROAD {cost: 1.3}]->(j),
       (h)-[:ROAD {cost: 1.3}]->(l),
       (i)-[:ROAD {cost: 1.1}]->(l),
       (j)-[:ROAD {cost: 1.1}]->(m),
       (j)-[:ROAD {cost: 1}]->(n),
       (k)-[:ROAD {cost: 0.4}]->(o),
```

```
(k)-[:ROAD {cost: 0.75}]->(p),
(k)-[:ROAD {cost: 0.0.9}]->(n),
(l)-[:ROAD {cost: 1.1}]->(m),
(l)-[:ROAD {cost: 1.7}]->(q),
(m)-[:ROAD {cost: 0.75}]->(r),
(m)-[:ROAD {cost: 1.3}]->(s),
(m)-[:ROAD {cost: 1.4}]->(t),
(m)-[:ROAD {cost: 1.2}]->(u),
(n)-[:ROAD {cost: 1.2}]->(r),
(o)-[:ROAD {cost: 1.9}]->(v),
(p)-[:ROAD {cost: 1.8}]->(v),
(q)-[:ROAD {cost: 0.9}]->(s),
(q)-[:ROAD {cost: 1}]->(t),
(q)-[:ROAD {cost: 0.7}]->(u);
```

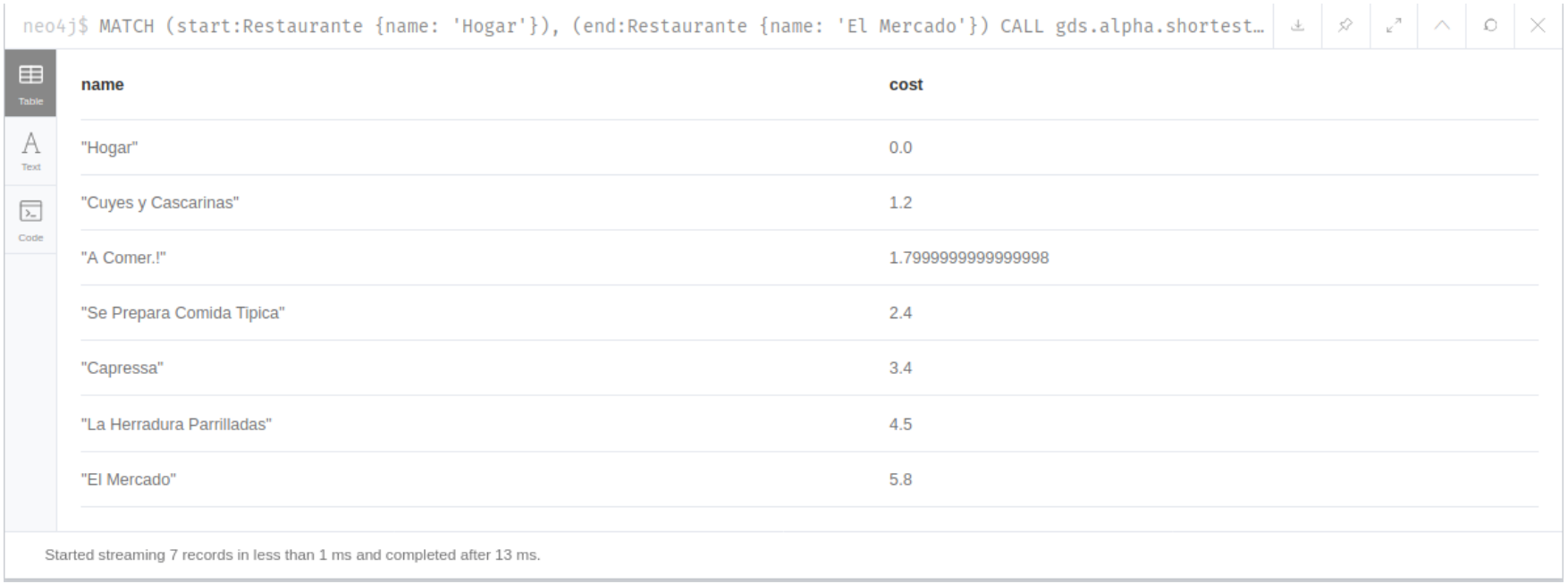
b. Dibujo de Neo4j



c. Código para búsqueda por Shortest Path en Neo4j

```
MATCH (start:Restaurante {name: 'Hogar'}), (end:Restaurante {name: 'El Mercado'})
CALL gds.alpha.shortestPath.stream({
  nodeProjection: 'Restaurante',
  relationshipProjection: {
    ROAD: {
      type: 'ROAD',
      properties: 'cost',
      orientation: 'UNDIRECTED'
    }
  },
  startNode: start,
  endNode: end,
  relationshipWeightProperty: 'cost'
})
YIELD nodeId, cost
RETURN gds.util.asNode(nodeId).name AS name, cost
```

d. Resultado en Neo4j



Conclusiones

Como se puede observar haciendo el proceso a mano determinamos la solución mediante el proceso del algoritmo Hill Climbing o Ascenso de Colinas. Podemos comprobar que es de gran ayuda ya que nos ayuda a encontrar rápido la solución al problema y con unos pasos muy sencillos. Además, al aplicarlo estamos aprendiendo métodos de búsqueda que nos ayudaran a resolver problemas.

Referencias

Neo4j.com. 2019. *A* - Path Finding Algorithms*. [online] Available at: <https://neo4j.com/docs/graph-data-science/current/alpha-algorithms/> [Accessed 19 May 2020].