Análisis de ocupación y empleo en México a través de gráficas (Documentación técnica)

Callejas Hernández Edgar

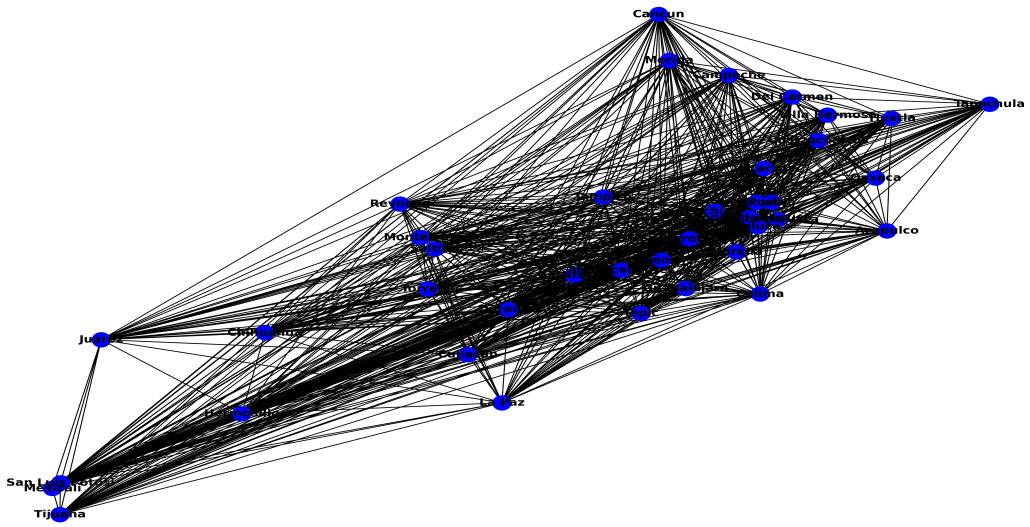
03/Diciembre/2021

1 Planteamiento

México fue uno de entre muchos países que resultan afectados económicamente por la pandemia de 2019. Con el uso de datos estadísticos sobre la ocupación y empleo en México se requiere un análisis sobre cómo ha ido evolucionando la economía, que ciudades son las más afectadas en algún sector económico, identificar en que lugares se pierden más empleos de mujeres o de hombres. Posterior al análisis seguiremos con la propuesta de alguna solución como planear la recuperación económica en alguna región que este más afectada ya sea en un sector especifico o particular, implementar un programa social para recuperar empleos de mujeres o salvar el sector económico primario que durante la pandemia resulto sumamente afectado.

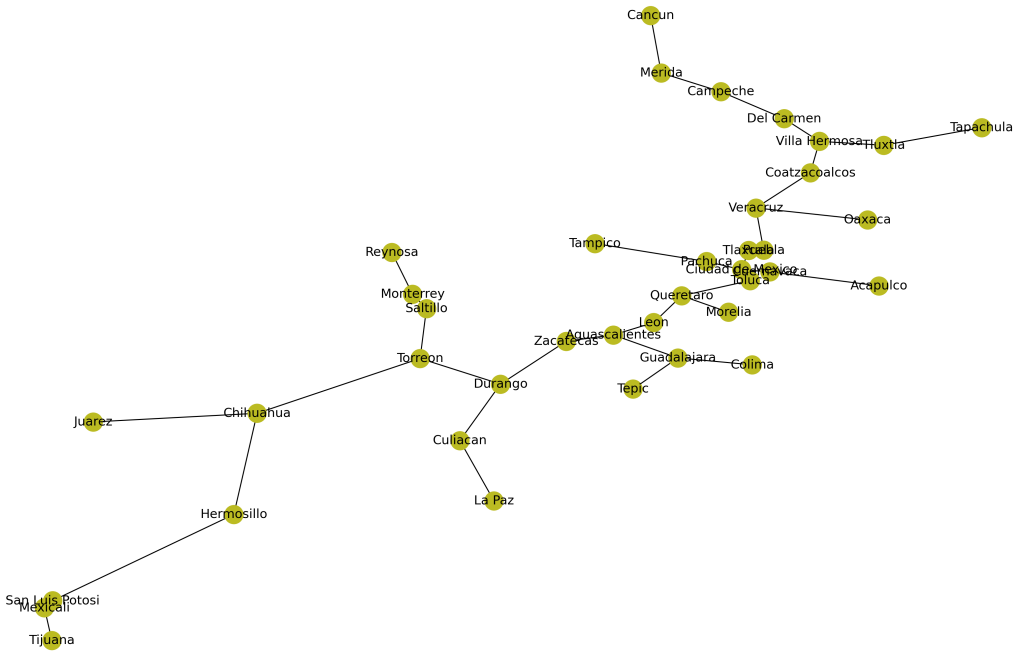
2 Forma de modelarlo matemáticamente

Primero debemos hacer una modelo general de las ciudades más cercanas para esto lo que hacemos es construir una gráfica completa de grado 38 porque en total son 39 ciudades, cada nodo le asignamos la diferencia entre los empleos del tercer trimestre y segundo trimestre, asignamos pesos a las aristas como la distancia en línea recta en kilómetros.



Teniendo el modelo general ahora aplicaremos una modelo más particular el cual se aproxime a encontrar las conexiones entre ciudades del país, para lograr esto decidimos construir el árbol generado aplicando el algoritmo de Prime ya que este se basa en tomar siempre las aristas más cortas.

Con este nuevo modelo ahora podemos aplicar otros algoritmos para poner restricciones ya sea de distancia o de algún atributo de interés como por ejemplo la pérdida de empleos en el sector terciario.



3 Enunciado del problema algorítmicos formal general a resolver

Para solucionar el problema de encontrar las ciudades más afectadas en algún sector económico con una distancia máxima entre ciudades lo que debemos de hacer es crear subconjuntos de ciudades dentro de una distancia máxima de 300 km (las distancia se puede modificar como quiera el usuario) y evaluar que subconjunto es el que estamos buscando como propuesta de una región para implementar un rescate económico. Para esto los debemos de asignar atributos a cada nodo o ciudad que serán la diferencia entre los empleos del tercer y segundo trimestre para ver la evolución de la ocupación por sector económico. también para separar entre hombres y mujeres dividimos cada sector quedando los atributos de los nodos de la siguiente manera:

1. Pri = Sector Primario
2. PriH = Sector Primario Hombres
3. PriM = Sector Primario Mujeres
4. Sec = Sector Secundario
5. SecH = Sector Secundario Hombres
6. SecM = Sector Secundario Mujeres
7. Ter = Sector Terciario

8. TerH = Sector Terciario Hombres
9. TerM = Sector Terciario Mujeres

4 Propuesta de solución mediante algoritmos combinatorios

Se realizan combinaciones para crear los subconjuntos y vamos eligiendo los subconjunto que cumplen con los requisitos

```
afectadosTer = {}
    empleosAfec = 10000
    nodcen = ''
    for c in G.nodes:
        I4 = ciudades_cercanas(G,c,distanciaMax)
        NTer = nx.get_node_attributes(I4,sector)
        N_Ter_sorted = sorted(NTer.items(), key=lambda x: x[1])
        cconMin = dict(N_Ter_sorted[0:numCiu])
        sumadeCEA = sum(cconMin.values())
        if sumadeCEA < empleosAfec:
            empleosAfec = sumadeCEA
            afectadosTer.clear()
            afectadosTer = cconMin.copy()
            I5 = I4.copy()
            nodcen = c
        I4.clear()
```

```
def CiuMasAfectDistMin(G,sector,distanciaMax=300,numCiu=3,plotGraph=False):
    afectadosTer = {}
    empleosAfec = 10000
    nodcen = ''
    for c in G.nodes:
        I4 = ciudades_cercanas(G,c,distanciaMax)
        NTer = nx.get_node_attributes(I4,sector)
        N_Ter_sorted = sorted(NTer.items(), key=lambda x: x[1])
        cconMin = dict(N_Ter_sorted[0:numCiu])
        sumadeCEA = sum(cconMin.values())
        if sumadeCEA < empleosAfec:
            empleosAfec = sumadeCEA
```

```
            afectadosTer.clear()
            afectadosTer = cconMin.copy()
            I5 = I4.copy()
            nodcen = c
        I4.clear()
    if plotGraph:
        nx.draw(I5,pos=Dpos,with_labels=True, node_color='#bbbb22')
        plt.title(nodcen)
    return afectadosTer
```

5 Análisis de correctitud y análisis asintótico de tiempo y espacio

Algoritmo de Prime corre en complejidad $O(|V| \cdot |E|)$

```
Prim (Grafo G)
       /* Inicializamos todos los nodos del grafo.
       La distancia la ponemos a infinito y el padre de cada nodo a NULL
        Encolamos, en una cola de prioridad
                  donde la prioridad es la distancia,
               todas las parejas <nodo, distancia> del grafo*/
       por cada u en V[G] hacer
           distancia[u] = INFINITO
           padre[u] = NULL
           Añadir(cola,<u, distancia[u]>)
       distancia[u]=0
       Actualizar(cola,<u, distancia[u]>)
       mientras !esta_vacia(cola) hacer
           // OJO: Se entiende por mayor prioridad aquel nodo cuya distancia[u] es menor.
           u = extraer_minimo(cola) //devuelve el mínimo y lo elimina de la cola.
           por cada v adyacente a 'u' hacer
               si ((v  cola) && (distancia[v] > peso(u, v)) entonces
                   padre[v] = u
                   distancia[v] = peso(u, v)
                   Actualizar(cola,<v, distancia[v]>)
```

Algoritmo de Dijkstra corre en $O(|V|^2)$

```
DIJKSTRA (Grafo G, nodo_fuente s)
      para u  V[G] hacer
          distancia[u] = INFINITO
          padre[u] = NULL
          visto[u] = false
      distancia[s] = 0
      adicionar (cola, (s, distancia[s]))
      mientras que cola no es vacía hacer
          u = extraer_mínimo(cola)
          visto[u] = true
          para todos v  adyacencia[u] hacer
              si ¬ visto[v]
                  si distancia[v] > distancia[u] + peso (u, v) hacer
                      distancia[v] = distancia[u] + peso (u, v)
                      padre[v] = u
                      adicionar(cola,(v, distancia[v]))
```

6 Estrategias de diseño de algoritmos o estructuras de datos utilizadas

Como estructura principal implementamos graficas de la libreria de networkx, para la implemetacion de los algoritmos implementamos diccionarios y para el guardar las distancias creamos una matriz de adyacencia por medio de un diccionario de doble entrada.

Uso de diccionarios para crear matriz de adyacencia para guardar los pesos de las aristas

```
disOp = {}
for i, e in enumerate(coor.values):
    nom1 = e[0]
    c1 = e[1:3]
    for i2, e2 in enumerate(coor.values):
        if i >= i2:
            continue
        nom2 = e2[0]
        c2 = e2[1:3]
        distancia = geopy.distance.distance(c1, c2).km
        disOp[nom1,nom2] = distancia
```

Uso de graficas para guardar datos

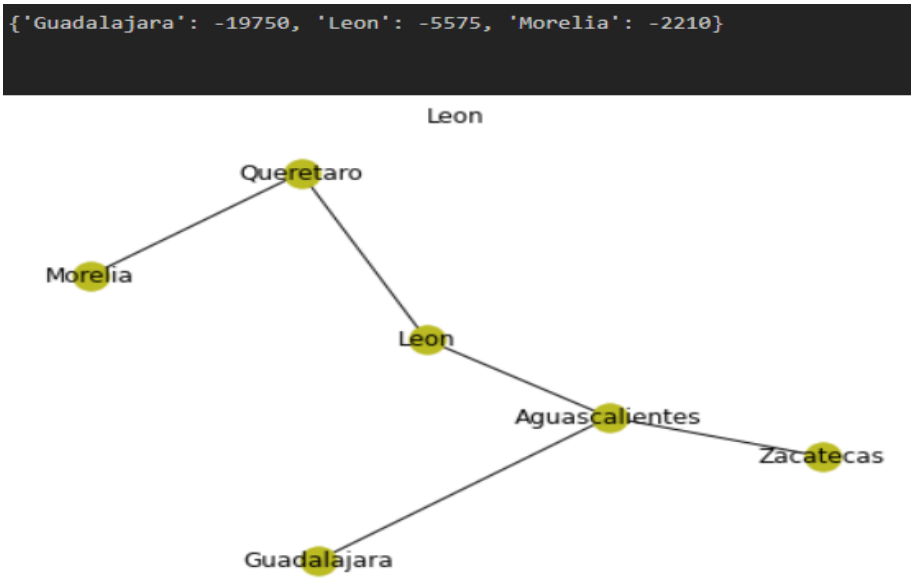
```
G = nx.Graph()
G.add_weighted_edges_from(dataFG)
H=nx.Graph()
H.add_weighted_edges_from(prim(G))
nx.set_node_attributes(H, node_attr)
```

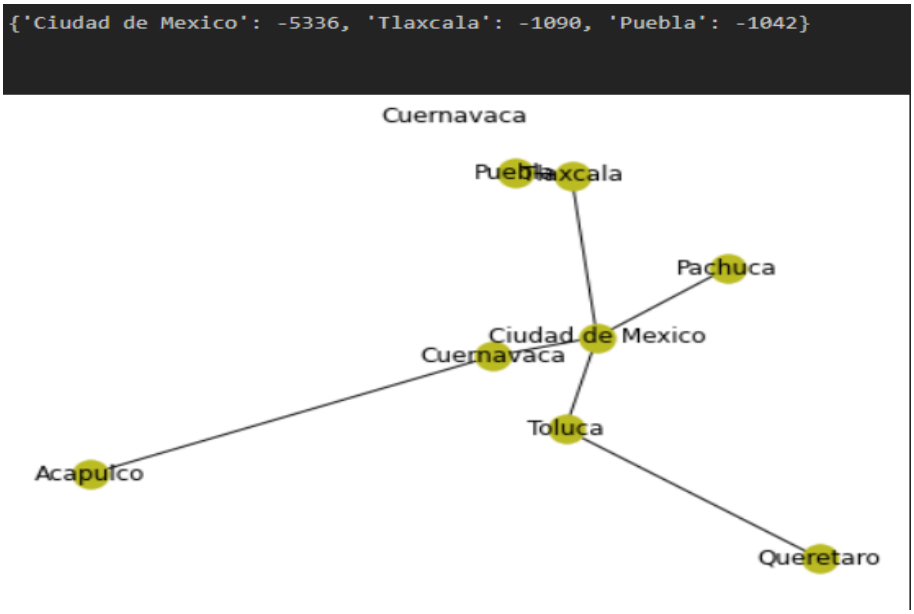
7 Aplicación a los datos concretos

La función requiere de una gráfica, atributo de nodo, distancia máxima, cantidad de ciudades, si queremos graficar el subconjunto de la gráfica)

Retorna el subconjunto de tres ciudades dentro de una distancia máxima con la mayor afectación en un sector económico especifico

```
CiuMasAfectDistMin(H,'TerM',300,3,True)
CiuMasAfectDistMin(H,'Pri',300,3,True)
```





8 Conclusiones

Las conclusiones son que las ciudades cercanas (distancia no mayor a 300 km) donde se pude implementar algún programa para la recuperación de empleos de mujeres en el sector terciario son León, Guadalajara y Morelia en las cuales dejaron de trabajar alrededor de 27,000 mujeres. Mientras que para los hombres las ciudades más cercanas con mayor afectación en el sector Terciario es Tijuana, Mexicali y San Luis Potosí con una baja de 27,000 aproximadamente. En general, las ciudades más cercanas donde el sector primario resulto más afectado fueron Ciudad de México, Tlaxcala y Puebla. Ahora se pude decidir en qué regiones invertir para ayudar a la reactivación económica de las ciudades más afectadas por la pandemia.

Para un posible trabajo futuro se pude mejorar la exactitud de nuestro modelo de conexiones en el país tomando en cuenta vía terrestre, aérea o marítima, además de ir actualizando los datos de encuestas posteriores, también esta la posibilidad de probar mejores algoritmos o plantear problemas distintos incorporando bases de datos complementarias.

9 Bibliografía

Algoritmo de Prime Liga: `https://es.wikipedia.org/wiki/Algoritmo_de_Prim`

Algoritmo Dijkstra Liga: `https://es.wikipedia.org/wiki/Algoritmo_de_Dijkstra`

Fuente ENOE INEGI Liga: `https://www.inegi.org.mx/app/descarga/?p=8&ag=00`

Fuente Coordenadas Geográficas Liga: `https://www.latlong.net/category/cities-142-15.html`