MA2015: Diseño de algoritmos matemáticos bioinspirados

Reto: Problema del Agente Viajero

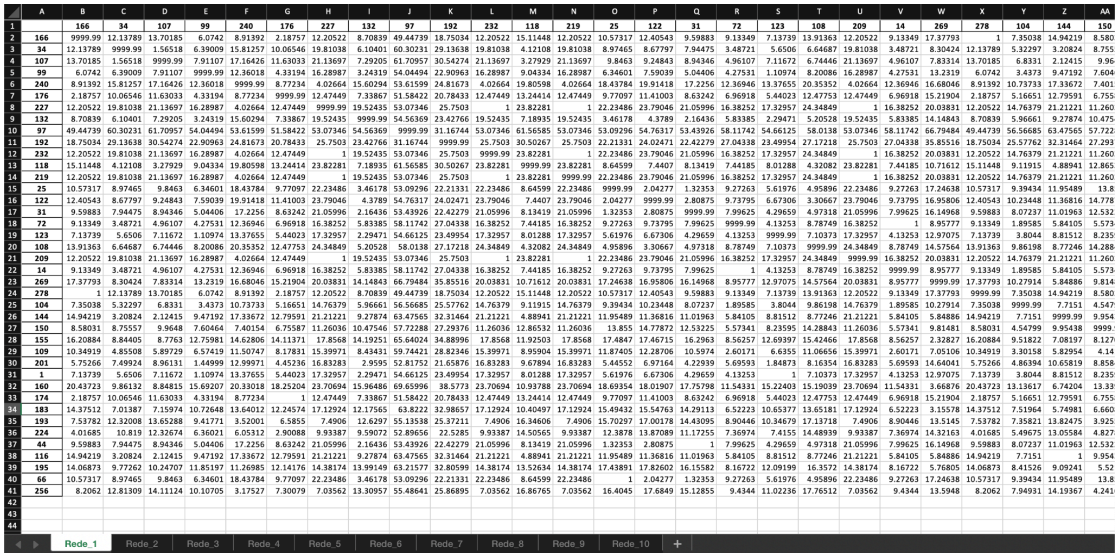
Nombre: Sebastián Miramontes Soto A01285296
Nombre: Mateo Zepeda A01722398
Nombre: Diego Armando Mijares A01722421

Instrucciones: *Resuelve completa y correctamente cada uno de los siguientes puntos. Se evalúa el procedimiento. Resultado sin procedimiento no tiene puntaje alguno.*

1. Diseña 10 redes de 40 nodos (direcciones) de forma aleatoria a partir de la siguiente lista de direcciones. Recuerda que vamos a minimizar la distancia recorrida.

Para este paso, se llevó a cabo un script que generó dichas 10 redes, por cantidad de nodos, subsecuentemente guardándolo en 4 diferentes excels: donde en cada uno hay 10 redes de diferentes cantidad de nodos por excel. El primer excel tiene 40 nodos por red, el segundo excel tiene 100 nodos por red, el tercer excel tiene 150 nodos por red, y el cuarto excel tiene 200 nodos por red.

Se anexa una imagen de ejemplo de estos excels:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA
1		166	34	107	99	240	176	227	132	97	192	232	118	219	25	122	31	72	123	108	209	14	269	278	104	144	150
2	166	9999.99	12.13789	13.70185	6.0742	8.91392	2.18757	12.20522	8.70839	49.44739	18.75034	12.20522	15.11448	12.20522	10.57317	12.40543	9.59883	9.13349	7.13739	13.91363	12.20522	9.13349	17.37793	1	7.35038	14.94219	8.580
3	34	12.13789	9999.99	1.56518	6.39009	15.81257	10.06546	19.81038	6.10401	60.30231	29.13638	19.81038	4.12108	19.81038	8.97465	8.67797	7.94475	3.48721	5.6506	6.64687	19.81038	3.48721	8.30424	12.13789	5.32297	3.20824	8.755
4	107	13.70185	1.56518	9999.99	7.91107	17.16426	11.63033	21.13697	7.29205	61.70957	30.54274	21.13697	3.27929	21.13697	9.8463	9.24843	8.94346	4.96107	7.11672	6.74446	21.13697	4.96107	7.83314	13.70185	6.8331	2.12415	9.96
5	99	6.0742	6.39009	7.91107	9999.99	12.36018	4.33194	16.28987	3.24319	54.04494	22.90963	16.28987	9.04334	16.28987	6.34601	7.59039	5.04406	4.27531	1.10974	8.20086	16.28987	4.27531	13.2319	6.0742	3.4373	9.47192	7.604
6	240	8.91392	15.81257	17.16426	12.36018	9999.99	8.77234	4.02664	15.60294	53.61599	24.81673	4.02664	19.80598	4.02664	18.43784	19.91418	17.2256	12.36946	13.37655	20.35352	4.02664	12.36946	16.68046	8.91392	10.73733	17.33672	7.401
7	176	2.18757	10.06546	11.63033	4.33194	8.77234	9999.99	12.47449	7.33867	51.58422	20.78433	12.47449	13.24414	12.47449	9.77097	11.41003	8.63242	6.96918	5.44023	12.47753	12.47449	6.96918	15.21904	2.18757	5.16651	12.79591	6.755
8	227	12.20522	19.81038	21.13697	16.28987	4.02664	12.47449	9999.99	19.52435	53.07346	25.7503	1	23.82281	1	22.23486	23.79046	21.05996	16.38252	17.32957	24.34849	1	16.38252	20.03831	12.20522	14.76379	21.21221	11.260
9	132	8.70839	6.10401	7.29205	3.24319	15.60294	7.33867	19.52435	9999.99	54.56369	23.42766	19.52435	7.18935	19.52435	3.46178	4.3789	2.16436	5.83385	2.29471	5.20528	19.52435	5.83385	14.14843	8.70839	5.96661	9.27874	10.475
10	97	49.44739	60.30231	61.70957	54.04494	53.61599	51.58422	53.07346	54.56369	9999.99	31.16744	53.07346	61.56585	53.07346	53.09296	54.76317	53.43926	58.11742	54.66125	58.0138	53.07346	58.11742	66.79484	49.44739	56.56685	63.47565	57.722
11	192	18.75034	29.13638	30.54274	22.90963	24.81673	20.78433	25.7503	23.42766	31.16744	9999.99	25.7503	30.50267	25.7503	22.21331	24.02471	23.42766	27.04338	23.49954	27.17218	25.7503	27.04338	35.85516	18.75034	25.57762	32.31464	27.293
12	232	12.20522	19.81038	21.13697	16.28987	4.02664	12.47449	1	19.52435	53.07346	25.7503	9999.99	23.82281	1	22.23486	23.79046	21.05996	16.38252	17.32957	24.34849	1	16.38252	20.03831	12.20522	14.76379	21.21221	11.260
13	118	15.11448	4.12108	3.27929	9.04334	19.80598	13.24414	23.82281	7.18935	61.56585	30.50267	23.82281	9999.99	23.82281	8.64599	7.4407	8.13419	7.44185	8.01288	4.32082	23.82281	7.44185	10.71612	15.11448	9.11915	4.88941	12.865
14	219	12.20522	19.81038	21.13697	16.28987	4.02664	12.47449	1	19.52435	53.07346	25.7503	1	23.82281	9999.99	22.23486	23.79046	21.05996	16.38252	17.32957	24.34849	1	16.38252	20.03831	12.20522	14.76379	21.21221	11.260
15	25	10.57317	8.97465	9.8463	6.34601	18.43784	9.77097	22.23486	3.46178	53.09296	22.21331	22.23486	8.64599	22.23486	9999.99	2.04277	1.32353	9.27263	5.61976	4.95896	22.23486	9.27263	17.24638	10.57317	9.39434	11.95489	13.8
16	122	12.40543	8.67797	9.24843	7.59039	19.91418	11.41003	23.79046	4.3789	54.76317	24.02471	23.79046	7.4407	23.79046	2.04277	9999.99	2.80875	9.73795	6.67306	3.30667	23.79046	9.73795	16.95806	12.40543	10.23448	11.36816	14.778
17	31	9.59883	7.94475	8.94346	5.04406	17.2256	8.63242	21.05996	2.16436	53.43926	22.42279	21.05996	8.13419	21.05996	1.32353	2.80875	9999.99	7.99625	4.29659	4.97318	21.05996	7.99625	16.14968	9.59883	8.07237	11.01963	12.532
18	72	9.13349	3.48721	4.96107	4.27531	12.36946	6.96918	16.38252	5.83385	58.11742	27.04338	16.38252	7.44185	16.38252	9.27263	9.73795	7.99625	9999.99	4.13253	8.78749	16.38252	1	8.95777	9.13349	1.89585	5.84105	5.573
19	123	7.13739	5.6506	7.11672	1.10974	13.37655	5.44023	17.32957	2.29471	54.66125	23.49954	17.32957	8.01288	17.32957	5.61976	6.67306	4.29659	4.13253	9999.99	7.10373	17.32957	4.13253	12.97075	7.13739	3.8044	8.81512	8.235
20	108	13.91363	6.64687	6.74446	8.20086	20.35352	12.47753	24.34849	5.20528	58.0138	27.17218	24.34849	4.32082	24.34849	4.95896	3.30667	4.97318	8.78749	7.10373	9999.99	24.34849	8.78749	14.57564	13.91363	9.86198	8.77246	14.288
21	209	12.20522	19.81038	21.13697	16.28987	4.02664	12.47449	1	19.52435	53.07346	25.7503	1	23.82281	1	22.23486	23.79046	21.05996	16.38252	17.32957	24.34849	9999.99	16.38252	20.03831	12.20522	14.76379	21.21221	11.260
22	14	9.13349	3.48721	4.96107	4.27531	12.36946	6.96918	16.38252	5.83385	58.11742	27.04338	16.38252	7.44185	16.38252	9.27263	9.73795	7.99625	1	4.13253	8.78749	16.38252	9999.99	8.95777	9.13349	1.89585	5.84105	5.573
23	269	17.37793	8.30424	7.83314	13.2319	16.68046	15.21904	20.03831	14.14843	66.79484	35.85516	20.03831	10.71612	20.03831	17.24638	16.95806	16.14968	8.95777	12.97075	14.57564	20.03831	8.95777	9999.99	17.37793	10.27914	5.84886	9.814
24	278	1	12.13789	13.70185	6.0742	8.91392	2.18757	12.20522	8.70839	49.44739	18.75034	12.20522	15.11448	12.20522	10.57317	12.40543	9.59883	9.13349	7.13739	13.91363	12.20522	9.13349	17.37793	9999.99	7.35038	14.94219	8.580
25	104	7.35038	5.32297	6.8331	3.4373	10.73733	5.16651	14.76379	5.96661	56.56685	25.57762	14.76379	9.11915	14.76379	9.39434	10.23448	8.07237	1.89585	3.8044	9.86198	14.76379	1.89585	10.27914	7.35038	9999.99	7.7151	4.547
26	144	14.94219	3.20824	2.12415	9.47192	17.33672	12.79591	21.21221	9.27874	63.47565	32.31464	21.21221	4.88941	21.21221	11.95489	11.36816	11.01963	5.84105	8.81512	8.77246	21.21221	5.84105	5.84886	14.94219	7.7151	9999.99	9.954
27	150	8.58031	8.75557	9.9648	7.60464	7.40154	6.75587	11.26036	10.47546	57.72288	27.29376	11.26036	12.86532	11.26036	13.855	14.77872	12.53225	5.57341	8.23595	14.28843	11.26036	5.57341	9.81481	8.58031	4.54799	9.95438	9999
28	155	16.20884	8.84405	8.7763	12.75981	14.62806	14.11371	17.8568	14.19251	65.64024	34.88996	17.8568	11.92503	17.8568	17.4847	17.46715	16.2963	8.56257	12.69397	15.42466	17.8568	8.56257	2.32827	16.20884	9.51822	7.08197	8.127
29	109	10.34919	4.85508	5.89729	6.57419	11.50747	8.17831	15.39971	8.43431	59.74421	28.82346	15.39971	8.95904	15.39971	11.87405	12.28706	10.5974	2.60171	6.6355	11.06656	15.39971	2.60171	7.05106	10.34919	3.30158	5.82954	4.14
30	201	5.75266	7.49924	8.96131	1.44999	12.99971	4.45236	16.83283	2.9595	52.81752	21.65876	16.83283	9.67894	16.83283	5.44552	6.97164	4.22939	5.69593	1.84873	8.16354	16.83283	5.69593	14.64041	5.75266	4.86394	10.65819	8.858
31	1	7.13739	5.6506	7.11672	1.10974	13.37655	5.44023	17.32957	2.29471	54.66125	23.49954	17.32957	8.01288	17.32957	5.61976	6.67306	4.29659	4.13253	1	7.10373	17.32957	4.13253	12.97075	7.13739	3.8044	8.81512	8.235
32	160	20.43723	9.86132	8.84815	15.69207	20.33018	18.25204	23.70694	15.96486	69.65996	38.5773	23.70694	10.93788	23.70694	18.69354	18.01907	17.75798	11.54331	15.22403	15.19039	23.70694	11.54331	3.66876	20.43723	13.13617	6.74204	13.33
33	174	2.18757	10.06546	11.63033	4.33194	8.77234	1	12.47449	7.33867	51.58422	20.78433	12.47449	13.24414	12.47449	9.77097	11.41003	8.63242	6.96918	5.44023	12.47753	12.47449	6.96918	15.21904	2.18757	5.16651	12.79591	6.755
34	183	14.37512	7.01387	7.15974	10.72648	13.64012	12.24574	17.12924	12.17565	63.8222	32.98657	17.12924	10.40497	17.12924	15.49432	15.54763	14.29113	6.52223	10.65377	13.65181	17.12924	6.52223	3.15578	14.37512	7.51964	5.74981	6.660
35	193	7.53782	12.32008	13.65288	9.41771	3.52001	6.5855	7.4906	12.6297	55.13538	25.37211	7.4906	16.34606	7.4906	15.70297	17.00178	14.43095	8.90446	10.34679	17.13718	7.4906	8.90446	13.5145	7.53782	7.35821	13.82475	3.925
36	224	4.01685	10.819	12.32674	6.36021	6.05312	2.90088	9.93387	9.59072	52.89656	22.5285	9.93387	14.50565	9.93387	12.3878	13.87089	11.17255	7.36974	7.4155	14.48939	9.93387	7.36974	14.32163	4.01685	5.49675	13.05584	4.827
37	44	9.59883	7.94475	8.94346	5.04406	17.2256	8.63242	21.05996	2.16436	53.43926	22.42279	21.05996	8.13419	21.05996	1.32353	2.80875	1	7.99625	4.29659	4.97318	21.05996	7.99625	16.14968	9.59883	8.07237	11.01963	12.532
38	116	14.94219	3.20824	2.12415	9.47192	17.33672	12.79591	21.21221	9.27874	63.47565	32.31464	21.21221	4.88941	21.21221	11.95489	11.36816	11.01963	5.84105	8.81512	8.77246	21.21221	5.84105	5.84886	14.94219	7.7151	1	9.954
39	195	14.06873	9.77262	10.24707	11.85197	11.26985	12.14176	14.38174	13.99149	63.21577	32.80599	14.38174	13.52634	14.38174	17.43891	17.82602	16.15582	8.16722	12.09199	16.3572	14.38174	8.16722	5.76805	14.06873	8.41526	9.09241	5.52
40	66	10.57317	8.97465	9.8463	6.34601	18.43784	9.77097	22.23486	3.46178	53.09296	22.21331	22.23486	8.64599	22.23486	1	2.04277	1.32353	9.27263	5.61976	4.95896	22.23486	9.27263	17.24638	10.57317	9.39434	11.95489	13.8
41	256	8.2062	12.81309	14.11124	10.10705	3.17527	7.30079	7.03562	13.30957	55.48641	25.86895	7.03562	16.86765	7.03562	16.4045	17.6849	15.12855	9.4344	11.02236	17.76512	7.03562	9.4344	13.5948	8.2062	7.94931	14.19367	4.241
42																											
43																											
44																											

Rede_1 | Rede_2 | Rede_3 | Rede_4 | Rede_5 | Rede_6 | Rede_7 | Rede_8 | Rede_9 | Rede_10 | +

Imagen de ejemplo

Esto simplemente es un fragmento de la primera red para 40 nodos. Subsecuentemente se tienen 9 redes más, y tres Excels más para las otras cantidades de nodos. Los índices son las matrículas por ubicación del Excel, y las distancias son kilómetros, que se consiguieron encontrando la latitud y longitud de cada ubicación con un *scrapper* que entra a Google Maps y da la longitud y latitud. En algunos casos específicos, se utilizó el 1 kilómetro, como una diferencia significativa para distancias que el algoritmo marcó como 0, esto debido a que el algoritmo que generaba estas coordenadas, hizo que fueran muy similares o incluso iguales.

2. Modelo Matemático

Conjuntos:

$\text{Sea } I = \{1, 2, \ldots, 40\}$ (Índice para filas y columnas)

$x_{ij} \in \{0, 1\} \quad \text{para todo } i, j \in I$ (Variable binaria de decisión)

$u_i \geq 0 \quad \text{para todo } i \in I$ (Variable positiva)

Parámetros:

c_{ij} (Costo o distancia para cada par (*i,j*))

Función Objetivo:

Minimizar *z*, que es el costo total:

$$\text{Minimizar } z = \sum_{i \in I} \sum_{j \in I} c_{ij} \cdot x_{ij}$$

Restricciones:

- Restricción de filas: Cada fila tiene exactamente una asignación saliente.

$$\sum_{j \in I} x_{ij} = 1 \quad \text{para todo } i \in I$$

- Restricción de columnas: Cada columna tiene exactamente una asignación entrante.

$$\sum_{i \in I} x_{ij} = 1 \quad \text{para todo } j \in I$$

- Restricción de eliminación de subcircuitos: Evita subcircuitos asegurando que no se

formen ciclos, excepto el recorrido completo.

$$u_i - u_j + |I| \cdot x_{ij} \leq |I| - 1 \quad \forall i, j \in I \quad (i \neq j \text{ y } i > 1 \text{ y } j > 1)$$

3. Obtén la solución óptima usando GAMS.

La distancia recorrida óptima ponderada de 10 redes de 40 nodos, da un total de: 151.34 km

El fitness ponderada da 0.00684388

El mejor recorrido sugiere este orden: 111.739082 - Caso 8

[1, 15, 17, 6, 37, 11, 25, 35, 14, 13, 2, 26, 40, 32, 23, 33, 5, 3, 38, 39, 19, 10, 7, 21, 28, 30, 29, 24, 18, 4, 9, 34, 22, 20, 16, 27, 12, 8, 31, 36, 1]

El tiempo de cómputo ponderado fue de 1.58 segundos.

Se anexa imágenes de un ejemplo del código, los códigos completos se encuentran en anexos finales:

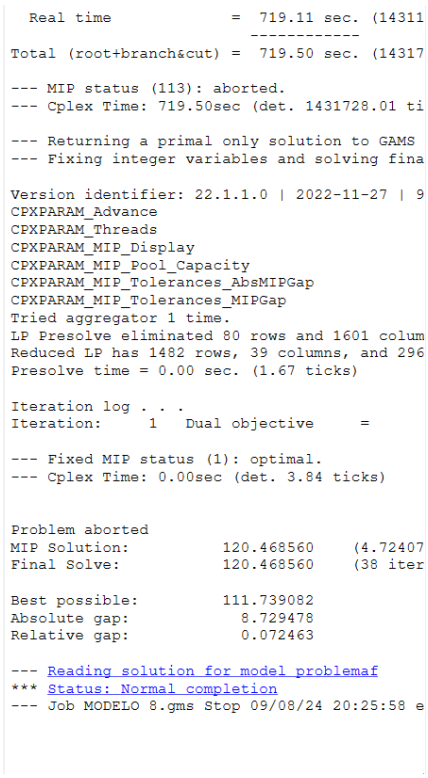
```
option optcr = 0.00001;

Set
 i /1*40/
 alias (i,j);
 variable z;
Binary Variable x(i, j);
positive variable u(i);;

Table
c(i,J)
```

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	1
1	9999.99	8.91392	17.2256	17.33672	17.2256	4.02664	19.41942	3.17527	19.96409	18.43784	4.02664	3.17527	8.77234	4.9183	4.02664	16.
2	8.91392	9999.99	9.59883	14.94219	9.59883	12.20522	11.75484	8.2062	16.34276	10.57317	12.20522	8.2062	2.18757	5.17897	12.20522	17.
3	17.2256	9.59883	9999.99	11.01963	1	21.05996	2.20791	15.12855	10.37664	1.32353	21.05996	15.12855	8.63242	12.3511	21.05996	16.
4	17.33672	14.94219	11.01963	9999.99	11.01963	21.21221	11.65039	14.19367	3.24564	11.95489	21.21221	14.19367	12.79591	13.34664	21.21221	5.
5	17.2256	9.59883	1	11.01963	9999.99	21.05996	2.20791	15.12855	10.37664	1.32353	21.05996	15.12855	8.63242	12.3511	21.05996	16.
6	4.02664	12.20522	21.05996	21.21221	21.05996	9999.99	23.26479	7.03562	23.92228	22.23486	1	7.03562	12.47449	8.88583	1	20.
7	19.41942	11.75484	2.20791	11.65039	2.20791	23.26479	9999.99	17.26762	10.38298	1.28004	23.26479	17.26762	10.83979	14.53357	23.26479	17.
8	3.17527	8.2062	15.12855	14.19367	15.12855	7.03562	17.26762	9999.99	16.88915	16.4045	7.03562	1	7.30079	3.1132	7.03562	13
9	19.96409	16.34276	10.37664	3.24564	10.37664	23.92228	10.38298	16.88915	9999.99	11.01078	23.92228	16.88915	14.32287	15.63786	23.92228	8
10	18.43784	10.57317	1.32353	11.95489	1.32353	22.23486	1.28004	16.4045	11.01078	9999.99	22.23486	16.4045	9.77097	13.58762	22.23486	17.
11	4.02664	12.20522	21.05996	21.21221	21.05996	1	23.26479	7.03562	23.92228	22.23486	9999.99	7.03562	12.47449	8.88583	1	20.
12	3.17527	8.2062	15.12855	14.19367	15.12855	7.03562	17.26762	1	16.88915	16.4045	7.03562	9999.99	7.30079	3.1132	7.03562	13
13	8.77234	2.18757	8.63242	12.79591	8.63242	12.47449	10.83979	7.30079	14.32287	9.77097	12.47449	7.30079	9999.99	4.20156	12.47449	15.
14	4.9183	5.17897	12.3511	13.34664	12.3511	8.88583	14.53357	3.1132	15.63786	13.58762	8.88583	3.1132	4.20156	9999.99	8.88583	14.
15	4.02664	12.20522	21.05996	21.21221	21.05996	1	23.26479	7.03562	23.92228	22.23486	1	7.03562	12.47449	8.88583	9999.99	20.
16	16.68046	17.37793	16.14968	5.84886	16.14968	20.03831	17.14188	13.5948	8.7127	17.24638	20.03831	13.5948	15.21904	14.06468	20.03831	99
17	4.02664	12.20522	21.05996	21.21221	21.05996	1	23.26479	7.03562	23.92228	22.23486	1	7.03562	12.47449	8.88583	1	20.
18	15.66162	13.2518	10.102	1.75409	10.102	19.5691	11.03011	12.53744	4.48126	11.15775	19.5691	12.53744	11.0923	11.60196	19.5691	6.
19	18.43784	10.57317	1.32353	11.95489	1.32353	22.23486	1.28004	16.4045	11.01078	1	22.23486	16.4045	9.77097	13.58762	22.23486	17.
20	18.36934	20.2415	19.51468	9.06161	19.51468	21.33387	20.49581	15.46698	11.65568	20.61187	21.33387	15.46698	18.1276	16.46468	21.33387	3.
21	18.76996	11.86484	2.74728	9.42854	2.74728	22.71819	2.24885	16.35865	8.15947	2.92279	22.71819	16.35865	10.5729	13.85227	22.71819	14.
22	18.36934	20.2415	19.51468	9.06161	19.51468	21.33387	20.49581	15.46698	11.65568	20.61187	21.33387	15.46698	18.1276	16.46468	21.33387	3.
23	13.05183	4.94198	4.66299	12.31511	4.66299	16.7228	6.8133	11.40171	12.8619	5.64481	16.7228	11.40171	4.27967	8.37862	16.7228	16.
24	15.66162	13.2518	10.102	1.75409	10.102	19.5691	11.03011	12.53744	4.48126	11.15775	19.5691	12.53744	11.0923	11.60196	19.5691	6.
25	1	8.91392	17.2256	17.33672	17.2256	4.02664	19.41942	3.17527	19.96409	18.43784	4.02664	3.17527	8.77234	4.9183	4.02664	16.
26	9.87489	2.11944	7.74878	12.9935	7.74878	13.49978	9.94415	8.4821	14.266	8.82528	13.49978	8.4821	1.18209	5.37972	13.49978	15.
27	12.59507	14.26441	15.18231	7.2915	15.18231	15.92482	16.63089	9.54898	10.53706	16.42734	15.92482	9.54898	12.20945	10.34837	15.92482	4.
28	19.54091	12.8732	3.84846	9.04467	3.84846	23.51567	3.0021	17.03598	7.44812	3.90298	23.51567	17.03598	11.50326	14.63247	23.51567	14.
29	12.98289	7.37287	5.10114	8.09993	5.10114	16.97287	7.01594	10.4903	9.02166	6.42193	16.97287	10.4903	5.49327	8.08783	16.97287	12.
30	20.35352	13.91363	4.97318	8.77246	4.97318	24.34849	3.94799	17.76512	6.82466	4.95896	24.34849	17.76512	12.47753	15.46376	24.34849	14.
31	3.17527	8.2062	15.12855	14.19367	15.12855	7.03562	17.26762	1	16.88915	16.4045	7.03562	1	7.30079	3.1132	7.03562	13
32	13.05183	4.94198	4.66299	12.31511	4.66299	16.7228	6.8133	11.40171	12.8619	5.64481	16.7228	11.40171	4.27967	8.37862	16.7228	16.
33	14.75716	6.9321	2.66675	11.44791	2.66675	18.52297	4.84059	12.8603	11.54612	3.72192	18.52297	12.8603	6.05215	9.95251	18.52297	15

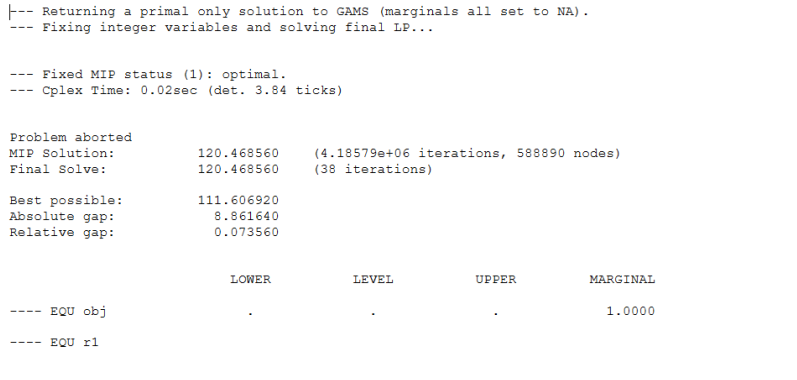
	33	34	35	36	37	38	39	40
1	14.75716	31.20614	4.9183	1	4.02664	17.2256	18.43784	13.05183
2	6.9321	30.41809	5.17897	8.91392	12.20522	9.59883	10.57317	4.94198
3	2.66675	25.37734	12.3511	17.2256	21.05996	1	1.32353	4.66299
4	11.44791	15.51093	13.34664	17.33672	21.21221	11.01963	11.95489	12.31511
5	2.66675	25.37734	12.3511	17.2256	21.05996	9999.99	1.32353	4.66299
6	18.52297	34.59502	8.88583	4.02664	1	21.05996	22.23486	16.7228
7	4.84059	25.125	14.53357	19.41942	23.26479	2.20791	1.28004	6.8133
8	12.8603	28.08224	3.1132	3.17527	7.03562	15.12855	16.4045	11.40171
9	11.54612	15.02642	15.63786	19.96409	23.92228	10.37664	11.01078	12.8619
10	3.72192	25.90983	13.58762	18.43784	22.23486	1.32353	1	5.64481
11	18.52297	34.59502	8.88583	4.02664	1	21.05996	22.23486	16.7228
12	12.8603	28.08224	3.1132	3.17527	7.03562	15.12855	16.4045	11.40171
13	6.05215	28.25072	4.20156	8.77234	12.47449	8.63242	9.77097	4.27967
14	9.95251	28.14504	1	4.9183	8.88583	12.3511	13.58762	8.37862
15	18.52297	34.59502	8.88583	4.02664	1	21.05996	22.23486	16.7228
16	15	14.	14.	16.	20.	16.	17.	16.
17	18.52297	34.59502	8.88583	4.02664	1	21.05996	22.23486	16.7228
18	6.05215	28.25072	4.20156	8.77234	12.47449	8.63242	9.77097	4.27967
19	9.95251	28.14504	1	4.9183	8.88583	12.3511	13.58762	8.37862
20	18.52297	34.59502	8.88583	4.02664	1	21.05996	22.23486	16.7228

```
Equations
obj, r1, r2, r3;

obj.. z=e=sum((i,j),c(i,j)*x(i,j));
r1(i).. sum(j, x(i,j))=e=1;
r2(j).. sum(i, x(i,j))=e=1;
r3(i,j)$((ord(i)>1 and ord(j)>1) and (ord(i)<>ord(j))).. u(i)-u(j)+card(i)*x(i,j)=L=card(i)-1;

model problemaf /all/;
solve problemaf using MIP min z;
```

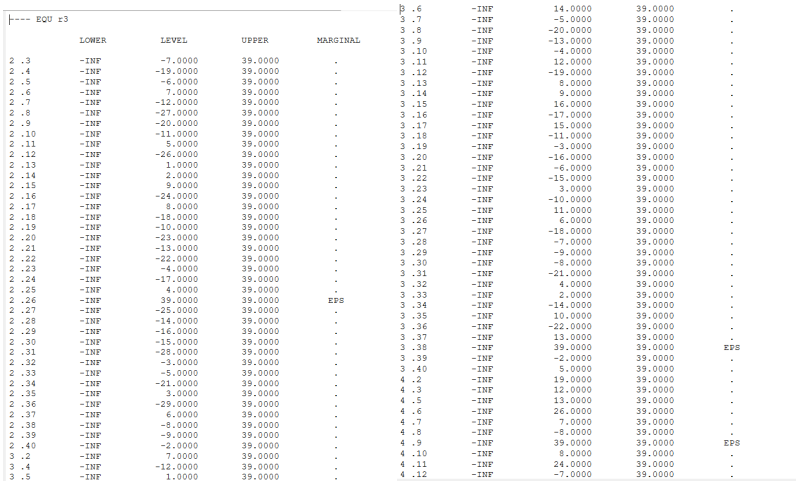
Imagenes de código en GAMS: ejemplo de 1 de las 10 redes



```
  Real time             =  719.11 sec. (14311
                           ------------
Total (root+branch&cut) =  719.50 sec. (14317

--- MIP status (113): aborted.
--- Cplex Time: 719.50sec (det. 1431728.01 ti

--- Returning a primal only solution to GAMS
--- Fixing integer variables and solving fina

Version identifier: 22.1.1.0 | 2022-11-27 | 9
CPXPARAM_Advance
CPXPARAM_Threads
CPXPARAM_MIP_Display
CPXPARAM_MIP_Pool_Capacity
CPXPARAM_MIP_Tolerances_AbsMIPGap
CPXPARAM_MIP_Tolerances_MIPGap
Tried aggregator 1 time.
LP Presolve eliminated 80 rows and 1601 colum
Reduced LP has 1482 rows, 39 columns, and 296
Presolve time = 0.00 sec. (1.67 ticks)

Iteration log . . .
Iteration:     1   Dual objective     =

--- Fixed MIP status (1): optimal.
--- Cplex Time: 0.00sec (det. 3.84 ticks)


Problem aborted
MIP Solution:          120.468560    (4.72407
Final Solve:           120.468560    (38 iter

Best possible:         111.739082
Absolute gap:            8.729478
Relative gap:            0.072463

--- Reading solution for model problemaf
*** Status: Normal completion
--- Job MODELO 8.gms Stop 09/08/24 20:25:58 e
```



```
--- Returning a primal only solution to GAMS (marginals all set to NA).
--- Fixing integer variables and solving final LP...


--- Fixed MIP status (1): optimal.
--- Cplex Time: 0.02sec (det. 3.84 ticks)


Problem aborted
MIP Solution:          120.468560    (4.18579e+06 iterations, 588890 nodes)
Final Solve:           120.468560    (38 iterations)

Best possible:         111.606920
Absolute gap:            8.861640
Relative gap:            0.073560


                           LOWER          LEVEL          UPPER         MARGINAL

---- EQU obj                 .              .              .             1.0000

---- EQU r1
```



---- EQU r3

	LOWER	LEVEL	UPPER	MARGINAL
2 .3	-INF	-7.0000	39.0000	.
2 .4	-INF	-19.0000	39.0000	.
2 .5	-INF	-6.0000	39.0000	.
2 .6	-INF	7.0000	39.0000	.
2 .7	-INF	-12.0000	39.0000	.
2 .8	-INF	-27.0000	39.0000	.
2 .9	-INF	-20.0000	39.0000	.
2 .10	-INF	-11.0000	39.0000	.
2 .11	-INF	5.0000	39.0000	.
2 .12	-INF	-26.0000	39.0000	.
2 .13	-INF	1.0000	39.0000	.
2 .14	-INF	2.0000	39.0000	.
2 .15	-INF	9.0000	39.0000	.
2 .16	-INF	-24.0000	39.0000	.
2 .17	-INF	8.0000	39.0000	.
2 .18	-INF	-18.0000	39.0000	.
2 .19	-INF	-10.0000	39.0000	.
2 .20	-INF	-23.0000	39.0000	.
2 .21	-INF	-13.0000	39.0000	.
2 .22	-INF	-22.0000	39.0000	.
2 .23	-INF	-4.0000	39.0000	.
2 .24	-INF	-17.0000	39.0000	.
2 .25	-INF	4.0000	39.0000	.
2 .26	-INF	39.0000	39.0000	EPS
2 .27	-INF	-25.0000	39.0000	.
2 .28	-INF	-14.0000	39.0000	.
2 .29	-INF	-16.0000	39.0000	.
2 .30	-INF	-15.0000	39.0000	.
2 .31	-INF	-28.0000	39.0000	.
2 .32	-INF	-3.0000	39.0000	.
2 .33	-INF	-5.0000	39.0000	.
2 .34	-INF	-21.0000	39.0000	.
2 .35	-INF	3.0000	39.0000	.
2 .36	-INF	-29.0000	39.0000	.
2 .37	-INF	6.0000	39.0000	.
2 .38	-INF	-8.0000	39.0000	.
2 .39	-INF	-9.0000	39.0000	.
2 .40	-INF	-2.0000	39.0000	.
3 .2	-INF	7.0000	39.0000	.
3 .4	-INF	-12.0000	39.0000	.
3 .5	-INF	1.0000	39.0000	.
3 .6	-INF	14.0000	39.0000	.
3 .7	-INF	-5.0000	39.0000	.
3 .8	-INF	-20.0000	39.0000	.
3 .9	-INF	-13.0000	39.0000	.
3 .10	-INF	-4.0000	39.0000	.
3 .11	-INF	12.0000	39.0000	.
3 .12	-INF	-19.0000	39.0000	.
3 .13	-INF	8.0000	39.0000	.
3 .14	-INF	9.0000	39.0000	.
3 .15	-INF	16.0000	39.0000	.
3 .16	-INF	-17.0000	39.0000	.
3 .17	-INF	15.0000	39.0000	.
3 .18	-INF	-11.0000	39.0000	.
3 .19	-INF	-3.0000	39.0000	.
3 .20	-INF	-16.0000	39.0000	.
3 .21	-INF	-6.0000	39.0000	.
3 .22	-INF	-15.0000	39.0000	.
3 .23	-INF	3.0000	39.0000	.
3 .24	-INF	-10.0000	39.0000	.
3 .25	-INF	11.0000	39.0000	.
3 .26	-INF	6.0000	39.0000	.
3 .27	-INF	-18.0000	39.0000	.
3 .28	-INF	-7.0000	39.0000	.
3 .29	-INF	-9.0000	39.0000	.
3 .30	-INF	-8.0000	39.0000	.
3 .31	-INF	-21.0000	39.0000	.
3 .32	-INF	4.0000	39.0000	.
3 .33	-INF	2.0000	39.0000	.
3 .34	-INF	-14.0000	39.0000	.
3 .35	-INF	10.0000	39.0000	.
3 .36	-INF	-22.0000	39.0000	.
3 .37	-INF	13.0000	39.0000	.
3 .38	-INF	39.0000	39.0000	EPS
3 .39	-INF	-2.0000	39.0000	.
3 .40	-INF	5.0000	39.0000	.
4 .2	-INF	19.0000	39.0000	.
4 .3	-INF	12.0000	39.0000	.
4 .5	-INF	13.0000	39.0000	.
4 .6	-INF	26.0000	39.0000	.
4 .7	-INF	7.0000	39.0000	.
4 .8	-INF	-8.0000	39.0000	.
4 .9	-INF	39.0000	39.0000	EPS
4 .10	-INF	8.0000	39.0000	.
4 .11	-INF	24.0000	39.0000	.
4 .12	-INF	-7.0000	39.0000	.
4 .13	-INF	20.0000	39.0000	.
4 .14	-INF	21.0000	39.0000	.
4 .15	-INF	28.0000	39.0000	.
4 .16	-INF	-5.0000	39.0000	.
4 .17	-INF	27.0000	39.0000	.
4 .18	-INF	1.0000	39.0000	.
4 .19	-INF	9.0000	39.0000	.
4 .20	-INF	-4.0000	39.0000	.
4 .21	-INF	6.0000	39.0000	.
4 .22	-INF	-3.0000	39.0000	.
4 .23	-INF	15.0000	39.0000	.
4 .24	-INF	2.0000	39.0000	.
4 .25	-INF	23.0000	39.0000	.
4 .26	-INF	18.0000	39.0000	.
4 .27	-INF	-6.0000	39.0000	.
4 .28	-INF	5.0000	39.0000	.
4 .29	-INF	3.0000	39.0000	.
4 .30	-INF	4.0000	39.0000	.
4 .31	-INF	-9.0000	39.0000	.
4 .32	-INF	16.0000	39.0000	.
4 .33	-INF	14.0000	39.0000	.
4 .34	-INF	-2.0000	39.0000	.
4 .35	-INF	22.0000	39.0000	.
4 .36	-INF	-10.0000	39.0000	.
4 .37	-INF	25.0000	39.0000	.
4 .38	-INF	11.0000	39.0000	.
4 .39	-INF	10.0000	39.0000	.
4 .40	-INF	17.0000	39.0000	.
5 .2	-INF	6.0000	39.0000	.
5 .3	-INF	39.0000	39.0000	EPS
5 .4	-INF	-13.0000	39.0000	.
5 .6	-INF	13.0000	39.0000	.
5 .7	-INF	-6.0000	39.0000	.
5 .8	-INF	-21.0000	39.0000	.
5 .9	-INF	-14.0000	39.0000	.
5 .10	-INF	-5.0000	39.0000	.
5 .11	-INF	11.0000	39.0000	.
5 .12	-INF	-20.0000	39.0000	.
5 .13	-INF	7.0000	39.0000	.
5 .14	-INF	8.0000	39.0000	.
5 .15	-INF	15.0000	39.0000	.
5 .16	-INF	-18.0000	39.0000	.
5 .17	-INF	14.0000	39.0000	.
5 .18	-INF	-12.0000	39.0000	.
5 .19	-INF	-4.0000	39.0000	.
5 .20	-INF	-17.0000	39.0000	.
5 .21	-INF	-7.0000	39.0000	.
5 .22	-INF	-16.0000	39.0000	.
5 .23	-INF	2.0000	39.0000	.
5 .24	-INF	-11.0000	39.0000	.
5 .25	-INF	10.0000	39.0000	.
5 .26	-INF	5.0000	39.0000	.
5 .27	-INF	-19.0000	39.0000	.
5 .28	-INF	-8.0000	39.0000	.
5 .29	-INF	-10.0000	39.0000	.
5 .30	-INF	-9.0000	39.0000	.
5 .31	-INF	-22.0000	39.0000	.
5 .32	-INF	3.0000	39.0000	.
5 .33	-INF	1.0000	39.0000	.
5 .34	-INF	-15.0000	39.0000	.
5 .35	-INF	9.0000	39.0000	.
5 .36	-INF	-23.0000	39.0000	.
5 .37	-INF	12.0000	39.0000	.
5 .38	-INF	-2.0000	39.0000	.
5 .39	-INF	-3.0000	39.0000	.
5 .40	-INF	4.0000	39.0000	.
6 .2	-INF	-7.0000	39.0000	.
6 .3	-INF	-14.0000	39.0000	.
6 .4	-INF	-26.0000	39.0000	.
6 .5	-INF	-13.0000	39.0000	.
6 .7	-INF	-19.0000	39.0000	.
6 .8	-INF	-34.0000	39.0000	.
6 .9	-INF	-27.0000	39.0000	.
6 .10	-INF	-18.0000	39.0000	.
6 .11	-INF	-2.0000	39.0000	.
6 .12	-INF	-33.0000	39.0000	.
6 .13	-INF	-6.0000	39.0000	.
6 .14	-INF	-5.0000	39.0000	.
6 .15	-INF	2.0000	39.0000	.
6 .16	-INF	-31.0000	39.0000	.
6 .17	-INF	1.0000	39.0000	.
6 .18	-INF	-25.0000	39.0000	.
6 .19	-INF	-17.0000	39.0000	.
6 .20	-INF	-30.0000	39.0000	.
6 .21	-INF	-20.0000	39.0000	.
6 .22	-INF	-29.0000	39.0000	.
6 .23	-INF	-11.0000	39.0000	.
6 .24	-INF	-24.0000	39.0000	.
6 .25	-INF	-3.0000	39.0000	.
6 .26	-INF	-8.0000	39.0000	.
6 .27	-INF	-32.0000	39.0000	.
6 .28	-INF	-21.0000	39.0000	.
6 .29	-INF	-23.0000	39.0000	.
6 .30	-INF	-22.0000	39.0000	.
6 .31	-INF	-35.0000	39.0000	.
6 .32	-INF	-10.0000	39.0000	.
6 .33	-INF	-12.0000	39.0000	.
6 .34	-INF	-28.0000	39.0000	.
6 .35	-INF	-4.0000	39.0000	.
6 .36	-INF	-36.0000	39.0000	.
6 .37	-INF	39.0000	39.0000	EPS
6 .38	-INF	-15.0000	39.0000	.
6 .39	-INF	-16.0000	39.0000	.
6 .40	-INF	-9.0000	39.0000	.
7 .2	-INF	12.0000	39.0000	.
7 .3	-INF	5.0000	39.0000	.
7 .4	-INF	-7.0000	39.0000	.
7 .5	-INF	6.0000	39.0000	.
7 .6	-INF	19.0000	39.0000	.
7 .8	-INF	-15.0000	39.0000	.
7 .9	-INF	-8.0000	39.0000	.
7 .10	-INF	1.0000	39.0000	.
7 .11	-INF	17.0000	39.0000	.
7 .12	-INF	-14.0000	39.0000	.
7 .13	-INF	13.0000	39.0000	.
7 .14	-INF	14.0000	39.0000	.
7 .15	-INF	21.0000	39.0000	.
7 .16	-INF	-12.0000	39.0000	.
7 .17	-INF	20.0000	39.0000	.
7 .18	-INF	-6.0000	39.0000	.
7 .19	-INF	2.0000	39.0000	.
7 .20	-INF	-11.0000	39.0000	.
7 .21	-INF	39.0000	39.0000	EPS
7 .22	-INF	-10.0000	39.0000	.
7 .23	-INF	8.0000	39.0000	.
7 .24	-INF	-5.0000	39.0000	.
7 .25	-INF	16.0000	39.0000	.
7 .26	-INF	11.0000	39.0000	.
7 .27	-INF	-13.0000	39.0000	.
7 .28	-INF	-2.0000	39.0000	.
7 .29	-INF	-4.0000	39.0000	.
7 .30	-INF	-3.0000	39.0000	.
7 .31	-INF	-16.0000	39.0000	.
7 .32	-INF	9.0000	39.0000	.
7 .33	-INF	7.0000	39.0000	.
7 .34	-INF	-9.0000	39.0000	.
7 .35	-INF	15.0000	39.0000	.
7 .36	-INF	-17.0000	39.0000	.
7 .37	-INF	18.0000	39.0000	.
7 .38	-INF	4.0000	39.0000	.
7 .39	-INF	3.0000	39.0000	.
7 .40	-INF	10.0000	39.0000	.
8 .2	-INF	27.0000	39.0000	.
8 .3	-INF	20.0000	39.0000	.
8 .4	-INF	8.0000	39.0000	.
8 .5	-INF	21.0000	39.0000	.
8 .6	-INF	34.0000	39.0000	.
8 .7	-INF	15.0000	39.0000	.
8 .9	-INF	7.0000	39.0000	.
8 .10	-INF	16.0000	39.0000	.
8 .11	-INF	32.0000	39.0000	.
8 .12	-INF	1.0000	39.0000	.
8 .13	-INF	28.0000	39.0000	.
8 .14	-INF	29.0000	39.0000	.
8 .15	-INF	36.0000	39.0000	.
8 .16	-INF	3.0000	39.0000	.
8 .17	-INF	35.0000	39.0000	.
8 .18	-INF	9.0000	39.0000	.
8 .19	-INF	17.0000	39.0000	.
8 .20	-INF	4.0000	39.0000	.
8 .21	-INF	14.0000	39.0000	.
8 .22	-INF	5.0000	39.0000	.
8 .23	-INF	23.0000	39.0000	.
8 .24	-INF	10.0000	39.0000	.
8 .25	-INF	31.0000	39.0000	.
8 .26	-INF	26.0000	39.0000	.
8 .27	-INF	2.0000	39.0000	.
8 .28	-INF	13.0000	39.0000	.
8 .29	-INF	11.0000	39.0000	.
8 .30	-INF	12.0000	39.0000	.
8 .31	-INF	39.0000	39.0000	EPS
8 .32	-INF	24.0000	39.0000	.
8 .33	-INF	22.0000	39.0000	.
8 .34	-INF	6.0000	39.0000	.
8 .35	-INF	30.0000	39.0000	.
8 .36	-INF	-2.0000	39.0000	.
8 .37	-INF	33.0000	39.0000	.
8 .38	-INF	19.0000	39.0000	.
8 .39	-INF	18.0000	39.0000	.
8 .40	-INF	25.0000	39.0000	.
20.2	-INF	23.0000	39.0000	.
20.3	-INF	16.0000	39.0000	.
20.4	-INF	4.0000	39.0000	.
20.5	-INF	17.0000	39.0000	.
20.6	-INF	30.0000	39.0000	.
20.7	-INF	11.0000	39.0000	.
20.8	-INF	-4.0000	39.0000	.
20.9	-INF	3.0000	39.0000	.
20.10	-INF	12.0000	39.0000	.
20.11	-INF	28.0000	39.0000	.
20.12	-INF	-3.0000	39.0000	.
20.13	-INF	24.0000	39.0000	.
20.14	-INF	25.0000	39.0000	.
20.15	-INF	32.0000	39.0000	.
20.16	-INF	39.0000	39.0000	EPS
20.17	-INF	31.0000	39.0000	.
20.18	-INF	5.0000	39.0000	.
20.19	-INF	13.0000	39.0000	.
20.21	-INF	10.0000	39.0000	.
20.22	-INF	1.0000	39.0000	.
20.23	-INF	19.0000	39.0000	.
20.24	-INF	6.0000	39.0000	.
20.25	-INF	27.0000	39.0000	.
20.26	-INF	22.0000	39.0000	.
20.27	-INF	-2.0000	39.0000	.
20.28	-INF	9.0000	39.0000	.
20.29	-INF	7.0000	39.0000	.
20.30	-INF	8.0000	39.0000	.

Imagenes de resultados completos de GAMS (Reading Solution): ejemplo de 1 de las 10 redes

4. Implementa un algoritmo genético para el TSP.

La distancia recorrida óptima ponderada de 10 redes de 40 nodos, da un total de: 407.26592 km

El mejor recorrido sugiere este orden: [73, 67, 34, 59, 74, 266, 219, 165, 19, 149, 53, 139, 171, 203, 185, 155, 94, 95, 85, 157, 49, 117, 90, 29, 150, 31, 15, 129, 106, 33, 78, 54, 215, 224, 228, 5, 17, 114, 41, 216, 73] , con distancia de 407.26592km

El tiempo de cómputo ponderado fue de 4.84 segundos.

Se anexa imagenes del código, los códigos completos se encuentran en anexos finales:

In [43]:
```
import pandas as pd
import random as rand
import numpy as np
import time
from geopy.distance import geodesic
```

Generar Cromosona

In [18]:
```
def generar_cromosona(casas):
    return rand.sample(casas, len(casas))
```

Crear Poblacion Inicial

In [19]:
```
def generar_poblacion_inicial(n, casas):

    poblacion = []

    for i in range(n):
        poblacion.append(generar_cromosona(casas))

    return poblacion
```

Funcion Fitness

In [65]:
```
def calcular_fitness(cromosona, comienzo, red):

    cromosona_con_fin = [comienzo] + cromosona + [comienzo]

    distancias = []

    for idx in range(len(cromosona_con_fin) - 1):
        actual = cromosona_con_fin[idx]
        siguiente = cromosona_con_fin[idx + 1]

        distancia = red.loc[red[red.columns[0]] == actual, siguiente].values[0]

        distancias.append(distancia)

    distancia_total = sum(distancias)
    fitness = 1 / distancia_total if distancia_total != 0 else float('inf')

    return fitness, distancia_total
```

In [21]:
```
def fitness_poblacional(poblacion, comienzo, red):

    lista_fitness = []
    lista_distancia = []

    for cromosona in poblacion:
        fitness_crom, distancia_crom = calcular_fitness(cromosona, comienzo, red)
        lista_fitness.append(fitness_crom)
        lista_distancia.append(distancia_crom)

    total_fit = np.sum(lista_fitness)  # Use NumPy for summation

    return lista_fitness, lista_distancia, total_fit
```

Seleccion

In [22]:
```
def seleccion_rouleta(poblacion, lista_fitness, fitness_total):
    lista_probabilidad = np.array(lista_fitness) / fitness_total
    padres = rand.choices(poblacion, weights=lista_probabilidad, k=2)
    return padres[0], padres[1]
```

In [92]:
```
def cruce(padre1, padre2, probabilidad_cruce):

    if rand.random() < probabilidad_cruce:

        punto = rand.randint(0,len(padre1))

        hijo1 = padre1[:punto] + padre2[punto:]
        hijo2 = padre2[:punto] + padre1[punto:]

        return hijo1, hijo2

    else:
        return padre1, padre2
```

Mutacion

In [24]:
```
def mutacion(hijos, probabilidad_mutacion):

    for hijo in hijos:
        if rand.random() < probabilidad_mutacion:

            idx1, idx2 = rand.sample(range(len(hijo)), 2)
            hijo[idx1], hijo[idx2] = hijo[idx2], hijo[idx1]

    return hijos
```

Forzar factibilidad de hijos

In [25]:
```
def forzar_factibilidad(hijos, lista_ciudades):

    for hijo in hijos:

        ciudades_faltantes = set(lista_ciudades) - set(hijo)
        ciudad_counts = {}

        for ciudad in hijo:
            if ciudad in ciudad_counts:
                ciudad_counts[ciudad] += 1
            else:
                ciudad_counts[ciudad] = 1

        for idx, ciudad in enumerate(hijo):
            if ciudad_counts[ciudad] > 1:

                if ciudades_faltantes:
                    new_city = ciudades_faltantes.pop()
                    hijo[idx] = new_city
                    ciudad_counts[ciudad] -= 1
                else:
                    break

    return hijos
```

Checar Fitness Max

In [105]:
```
def checar_fitness_max(poblacion, max_fitness, min_distancia, generacion, comienzo, red, crom_min):

    lista_fit, lista_dist, fit_pob = fitness_poblacional(poblacion, comienzo, red)

    max_fit_pob = max(lista_fit)

    if max_fit_pob > max_fitness:

        max_fitness = max_fit_pob

        max_fit_index = lista_fit.index(max_fit_pob)

        min_distancia = lista_dist[max_fit_index]

        crom_min_sin_comienzo = poblacion[max_fit_index]

        crom_min = [comienzo] + crom_min_sin_comienzo + [comienzo]

        #print(f"Generacion: {generacion}\nFitness: {max_fitness}\nDistancia: {min_distancia}\nCromosona:{crom_min}\n\n")

    return max_fitness, min_distancia, crom_min
```

Parametros

In [27]:
```
max_generaciones = 100
max_fitness = 0
min_distancia = 999999
tamanio_poblacional = 4

probabildad_cruce = 0.9
probabilidad_mutacion = 0.05
```

Disena 10 redes de 40 nodos (direcciones) de forma aleatoria a partir de la siguiente lista de direcciones. Recuerda que vamos a minimizar la distancia recorrida.

In [144]:
```
red1_40 = pd.read_excel("matrices_40.xlsx", sheet_name = "Rede_1")
red2_40 = pd.read_excel("matrices_40.xlsx", sheet_name = "Rede_2")
red3_40 = pd.read_excel("matrices_40.xlsx", sheet_name = "Rede_3")
red4_40 = pd.read_excel("matrices_40.xlsx", sheet_name = "Rede_4")
red5_40 = pd.read_excel("matrices_40.xlsx", sheet_name = "Rede_5")
red6_40 = pd.read_excel("matrices_40.xlsx", sheet_name = "Rede_6")
red7_40 = pd.read_excel("matrices_40.xlsx", sheet_name = "Rede_7")
red8_40 = pd.read_excel("matrices_40.xlsx", sheet_name = "Rede_8")
red9_40 = pd.read_excel("matrices_40.xlsx", sheet_name = "Rede_9")
red10_40 = pd.read_excel("matrices_40.xlsx", sheet_name = "Rede_10")

redes_40 = [red1_40,red2_40,red3_40,red4_40,red5_40,red6_40,red7_40,red8_40,red9_40,red10_40]
```

```python
def algoritmo_genetico(redes):

    for count, red in enumerate(redes):

        inicia_tiempo = time.time()

        comienzo_ruta = int(red.columns[1])
        print(f"Red #{count+1}\nComenzando en {comienzo_ruta}\n\n")

        lista_casas = list(red.columns[2:])

        max_generaciones = 100
        max_fitness = 0
        min_distancia = 99999999
        tamanio_poblacional = 4
        probabilidad_mutacion = 0.1
        probabilidad_cruce = 0.9
        crom_min = []

        poblacion_inicial = generar_poblacion_inicial(tamanio_poblacional, lista_casas)

        for generacion in range(max_generaciones):

            max_fitness, min_distancia, crom_min = checar_fitness_max(
                poblacion_inicial, max_fitness, min_distancia, generacion, comienzo_ruta, red, crom_min
            )

            hijos_finales = []

            while len(hijos_finales) < tamanio_poblacional:
                lista_fitness, lista_distancias, fitness_total = fitness_poblacional(poblacion_inicial, comienzo_ruta, red)
                padre1, padre2 = seleccion_rouleta(poblacion_inicial, lista_fitness, fitness_total)
                hijo1, hijo2 = cruce(padre1, padre2, probabilidad_cruce)
                hijos = [hijo1, hijo2]
```

```python
                hijos_mutados = mutacion(hijos, probabilidad_mutacion)
                hijos_factibles = forzar_factibilidad(hijos_mutados, lista_casas)
                hijos_finales.extend(hijos_factibles)

            poblacion_inicial = hijos_finales

        acaba_tiempo = time.time()
        tiempo = acaba_tiempo - inicia_tiempo

        tcp_200_df.loc[0, f"Red {count+1}"] = max_fitness
        tcp_200_df.loc[1, f"Red {count+1}"] = min_distancia
        tcp_200_df.loc[2, f"Red {count+1}"] = tiempo
        tcp_200_df.loc[3, f"Red {count+1}"] = str(crom_min)

        tcp_200_df.to_excel("TCP200.xlsx", index=False)
```

```
In [151]: algoritmo_genetico(redes_200)

Red #1
Comenzando en 0


Red #2
Comenzando en 0
```

Imagenes de código completo de algoritmo genético

		Red 1	Red 2	Red 3	Red 4	Red 5	Red 6	Red 7	Red 8	Red 9	Red 10	Promedio	Mejor
2	Fitness	0.00196	0.00246	0.00224	0.00217	0.00219	0.00236	0.0021	0.00221	0.00235	0.00226	0.00223	0.00246
3	F.O. Min	509.122	407.266	446.8	460.443	457.25	423.341	476.33	451.801	425.288	443.121	450.076	407.266
4	Tiempo Computacional	4.78686	4.83532	4.9896	4.97506	4.91787	4.99541	4.86012	4.97543	5.03374	5.0256	4.9395	4.83532
5	Cromosona	[166, 150,	[73, 67, 34	[96, 263, 7	[167, 176,	[20, 93, 61	[90, 161, 6	[51, 64, 13	[156, 247,	[123, 30, 1	[92, 259, 2.		[73, 67, 34, 5

Imagen de resultados completos de algoritmo genético

5. Implementa un algoritmo de colonia de hormigas para obtener una solución al problema.

La distancia recorrida óptima ponderada de 10 redes de 40 nodos, da un total de: 162.94 km

El fitness ponderada da 0.0063

El mejor recorrido sugiere este orden:

[0, 35, 24, 11, 30, 7, 13, 34, 12, 25, 1, 22, 39, 31, 32, 2, 37, 4, 9, 38, 18, 6, 20, 27, 29, 28, 17, 23, 3, 8, 33, 15, 19, 21, 26, 10, 5, 16, 14, 36, 0] con distancia 125.50 km

El tiempo de cómputo ponderado fue de 2.72 segundos.

Se anexa imagenes del código, los códigos completos se encuentran en anexos finales:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import time  # Importar el módulo time

# Cargar los archivos de Excel
file_paths = {
    40: '/content/drive/MyDrive/Reto bioinspirados/matrices_40.xlsx',
    100: '/content/drive/MyDrive/Reto bioinspirados/matrices_100.xlsx',
    150: '/content/drive/MyDrive/Reto bioinspirados/matrices_150.xlsx',
    200: '/content/drive/MyDrive/Reto bioinspirados/matrices_200.xlsx'
}

# Definir los parámetros del algoritmo de hormigas
alpha = 1.0  # Importancia de la feromona
beta = 5.0   # Importancia de la distancia
rho = 0.5    # Tasa de evaporación de la feromona
Q = 100      # Constante de la cantidad de feromona depositada
num_ants = 10  # Número de hormigas
num_iterations = 100  # Número de iteraciones

# Función para inicializar la matriz de feromonas
def initialize_pheromone_matrix(num_nodes):
    return np.ones((num_nodes, num_nodes))

# Función para calcular la probabilidad de seleccionar el siguiente nodo
def calculate_transition_probabilities(current_node, unvisited_nodes, pheromone_matrix, distance_matrix):
    pheromones = np.power(pheromone_matrix[current_node, unvisited_nodes], alpha)
    heuristic = np.power(1.0 / distance_matrix[current_node, unvisited_nodes], beta)
    probabilities = pheromones * heuristic
```

```python
    return probabilities / probabilities.sum()

# Función para ejecutar el recorrido de una hormiga
def run_ant(num_nodes, pheromone_matrix, distance_matrix):
    current_node = 0  # Iniciar en el nodo de origen
    visited_nodes = [current_node]
    total_distance = 0

    for _ in range(num_nodes - 1):
        unvisited_nodes = list(set(range(num_nodes)) - set(visited_nodes))
        probabilities = calculate_transition_probabilities(current_node, unvisited_nodes, pheromone_matrix, distance_matrix)
        next_node = np.random.choice(unvisited_nodes, p=probabilities)
        total_distance += distance_matrix[current_node, next_node]
        visited_nodes.append(next_node)
        current_node = next_node

    # Regresar al nodo inicial
    total_distance += distance_matrix[current_node, 0]
    visited_nodes.append(0)

    return visited_nodes, total_distance

# Función para actualizar la matriz de feromonas
def update_pheromone_matrix(pheromone_matrix, ant_paths, ant_distances):
    num_nodes = len(pheromone_matrix)
    pheromone_matrix *= (1 - rho)  # Evaporación
    for path, distance in zip(ant_paths, ant_distances):
        for i in range(num_nodes):
            pheromone_matrix[path[i], path[i+1]] += Q / distance
            pheromone_matrix[path[i+1], path[i]] += Q / distance
    return pheromone_matrix
```

```python
# Función principal para resolver el problema TSP usando ACO
def solve_tsp_with_aco(distance_matrix, num_ants, num_iterations):
    num_nodes = len(distance_matrix)
    pheromone_matrix = initialize_pheromone_matrix(num_nodes)
    best_path = None
    best_distance = float('inf')

    for _ in range(num_iterations):
        ant_paths = []
        ant_distances = []

        for _ in range(num_ants):
            path, distance = run_ant(num_nodes, pheromone_matrix, distance_matrix)
            ant_paths.append(path)
            ant_distances.append(distance)
            if distance < best_distance:
                best_distance = distance
                best_path = path

        pheromone_matrix = update_pheromone_matrix(pheromone_matrix, ant_paths, ant_distances)

    return best_path, best_distance

# Procesar cada archivo de Excel
for num_nodes, file_path in file_paths.items():
    excel_data = pd.ExcelFile(file_path)
    fitness_scores = []
    total_distances = []
    best_global_path = None
    best_global_distance = float('inf')
```

```python
for sheet_name in excel_data.sheet_names:
    # Cargar la red (distancia) desde el Excel
    distance_matrix = pd.read_excel(excel_data, sheet_name=sheet_name, index_col=0).values

    # Medir el tiempo de resolución
    start_time = time.time()  # Iniciar el temporizador
    best_path, best_distance = solve_tsp_with_aco(distance_matrix, num_ants, num_iterations)
    end_time = time.time()  # Detener el temporizador

    # Calcular el tiempo de resolución
    elapsed_time = end_time - start_time

    # Calcular fitness
    fitness = 1 / best_distance
    fitness_scores.append(fitness)
    total_distances.append(best_distance)

    # Guardar la mejor ruta global
    if best_distance < best_global_distance:
        best_global_distance = best_distance
        best_global_path = best_path

# Calcular fitness ponderada
weighted_fitness = np.mean(fitness_scores) / 10
average_distance = np.mean(total_distances)
```

```python
# Imprimir resultados
print(f"Excel con {num_nodes} nodos:")
print(f"Fitness ponderada: {weighted_fitness:.4f}")
print(f"Distancia promedio: {average_distance:.2f}")
print(f"Mejor ruta: {best_global_path} con distancia {best_global_distance:.2f}")
print(f"Tiempo computacional: {elapsed_time:.2f} segundos")  # Imprimir el tiempo computacional
print("\n")
```

Imagenes de código completo de algoritmo genético

```
Excel con 40 nodos:
Fitness ponderada: 0.2528
Distancia promedio: 162.94
Mejor ruta: [0, 35, 24, 11, 30, 7, 13
Tiempo computacional: 2.72 segundos
```

Imagen de resultados completos de algoritmo genético

**La mejor ruta se despliega completa junto a su kilometraje, por la longitud de aquella imagen favor de recurrir al anexo del código al final en caso de querer ver el despliegue completo.*

6. Compara el valor exacto con los algoritmos bioinspirados que creaste.

El algoritmo genético, tiene un fitness de 35.88% de lo que es el fitness hecho con el valor exacto que se consiguió con GAMS. El tiempo de computación con genéticos, fue un 306.03% de lo que es para el modelo hecho en GAMS.

El algoritmo de hormigas, tiene un fitness de 92.05% de lo que es el fitness hecho con el valor exacto que se consiguió con GAMS. El tiempo de computación con hormigas, fue un 172.15% de lo que es para el modelo hecho en GAMS.

Para apreciar las diferencias para las 10 redes de 100, 150 y 200 nodos, favor de recurrir al infográfico.

[Anexos finales]:
https://drive.google.com/drive/folders/1BNb5nqIS0jgv2pDuI5PbLDSYJJ6deo3K?usp=sharing

Bibliografía:

Gutin, G., & Punnen, A. P. (2007). The Traveling Salesman Problem and Its Variations. Springer.

Dorigo, M., & Stützle, T. (2004). Ant Colony Optimization. MIT Press.

Mitchell, M. (1998). An Introduction to Genetic Algorithms. MIT Press.

Liga a infografía:
https://www.canva.com/design/DAGQO23mm40/djKTgPUuRE3P7W9SMJelDA/edit?utm_content=DAGQO23mm40&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton