Practica4_Colonias

December 4, 2023

1 Práctica 4: Metaheurísticas basadas en poblaciones - Sistemas de Colonias de Hormigas

David Pacios Vázquez

1.1 Instrucciones

Esto es **Jupyter Notebook**, un documento que integra código Python en un archivo Markdown. Esto nos permite ir ejecutando celdas de código poco a poco, así como generar automáticamente un informe bien formateado de la práctica.

Puedes añadir una celda con el botón **"Insert"** de la barra de herramentas, y cambiar su tipo con **"Cell > Cell Type"**

Para ejecutar una celda de código, la seleccionaremos y pulsaremos el botón **" Run"** de la barra de herramentas. Para pasar el documento a HTML, seleccionaremos **"File > Download as > HTML (.html)"**

Sigue este guión hasta el final. Ejecuta el código proporcionado paso a paso comprendiendo lo que estás haciendo y reflexionando sobre los resultados. Habrá preguntas intercaladas a lo largo del guión, responde a todas ellas en la sección reservada para ese fin: **"Respuestas a los cuestionarios"**. Por favor, no modifiques ninguna linea de código excepto cuando se te pida explícitamente.

No olvides insertar tu **nombre y apellidos** en la celda superior.

IMPORTANTE: Escribe el código de tu o tus soluciones/respuestas en las celdas que se indican para ello. Además, a lo largo de la práctica se plantearán varias preguntas que debéis responder en la parte inferior del documento, incluyendo las celdas que veáis necesarias (si hacéis referencia a partes concretas de vuestro código, etc) para reponder a ellas.

1.2 Entrega de la práctica

La fecha límite de entrega será la indicada en el Campus Virtual. La entrega consistirá de un único archivo comprimido con nombre `APELIDOS_NOME_Colonias.zip` que contenga los seguientes ficheros:

- `APELIDOS_NOME_Colonias.html`: Archivo HTML fruto de la exportación del presente Notebook, con las preguntas respondidas al final del documento.
- `APELIDOS_NOME_Colonias.ipynb`: Archivo fuente Jupyter Notebook.
- Archivo de datos de los problema utilizados en la resolución.

1.3 El Problema del Viajante de Comercio (VC) con Sistemas de Colonias de Hormigas

El objetivo de esta práctica es modelar e implementar un agente inteligente que sea capaz de resolver el problema del VC mediante la metaheurística (MH) de algoritmos para sistemas de colonias de hormigas (ACS, *Ant Colony Systems*, del inglés).

Para ello, realizarás una implementación del algoritmo básico visto en la clase expositiva y valorarás algunos de los parámetros de diseño del algoritmo en relación a la calidad de las soluciones alcanzadas.

Recuerda que debes importar el módulo Python que acompaña esta práctica, que como ocurría en prácticas anteriores implementa la carga de datos y el cálculo de distancias utilizando una matriz de adyacencia de acuerdo a la definición del problema ya vista.

[1]:
```
from helpers_mod_aco import *

g1=Localizaciones(filename='./data/grafo8cidades.txt')

# distancia entre ciudad 0 y 1 son unos 55 km
print (g1.distancia(0,1))
```

```
55.88273580792048
```

1.4 P4.1: Implementación de Sistemas de Colonias de Hormigas (SCH)

Implementa el algoritmo de sistemas de colonia de hormigas que materializa la MH vista en la sesión expositiva y revisa las notas particulares para el problema de VC. Para facilitar vuestra labor, se os proporciona aquí una descripción algorítmica de referencia.

1. Establecer parámetros: $G, m, q_0, \tau_0, \rho, \xi, iter_{min}, iter_{max}, iter_{sinmejoras}, ratio_{mejora}$
2. Inicializar pistas de feromonas: $\forall (i,j) \tau_{ij} = \tau_0$
3. HACER
   3.1. PARA cada hormiga k en colonia m HACER
      3.1.1.construir una solución para k siguiendo la regla de decisión a cada paso:

$$j = \begin{cases} argmax_{l \in N_i^k}\{[\tau_{il}]^\alpha [\eta_{il}]^\beta\}, \text{if } q \leq q_0; \\ J, \text{en otro caso;} \end{cases}$$

, donde J es la exploración dirigida por:

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta}, \text{if } j \in N_i^k$$

      3.1.2. (opcional) Aplicar actualización de feromona local/online con la regla:

$$\tau_{ij} \leftarrow (1-\xi)\tau_{ij} + \xi\tau_0$$

3.2. FIN PARA

3.3. Evalúa soluciones en la colonia m y registrar la mejor solución hasta el momento T^{bs}

3.4. Aplicar actualización de feromona global/offline con la regla:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij}^{bs}, \forall(i,j) \in T^{bs}$$

4. HASTA cumplir la condición de parada

La implementación debe permitir la inicialización de los parámetros y estructuras de información asociadas como sigue: - G es la estructura de grafo que permite calcular distancias entre ciudades sobre una instancia del problema dada (e.g., fichero de localizaciones). - Probabilidad de explotación $q_0 = 0.5$. - Inicialización de la información heurística: $\eta_{ij} = 1/d_{ij}, \forall(i,j)$, donde d_{ij} es la distancia directa entre las ciudades i y j. - Inicialización de las pistas de feromonas (depósito inicial): $\tau_{ij} = \tau_0 = 1/(nC^{NN})\forall(i,j)$ donde n es el número de ciudades y C^{NN} es el coste del tour generado siguiendo una estrategia voraz del vecino más próximo (NN, *Nearest Neighbour*). - Parámetros de influencia de las pistas de feromonas (α) e información heurística (β) en la regla de decisión establecidos a 1 y 3 respectivamente - Ratio de evaporación de feromona: deberás probar con diferentes magnitudes (p.ej., $\rho = 0.01$, $\rho = 0.1$ y $\rho = 0.5$) - Actualización de feromona global con depósito elitista basado en la mejor solución (bs, *best solution*): $\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij}^{bs}, \forall(i,j) \in T^{bs}, donde \Delta\tau_{ij}^{bs} = 1/C^{bs}$ - Actualización de feromona local siguiendo la regla: $\tau_{ij} \leftarrow (1-\xi)\tau_{ij} + \xi\tau_0, con \xi = 0.1$ - Condición de parada que de manera combinada tenga en cuenta $iter_{min}$, $iter_{max}$, $iter_{sinmejoras}$, $ratio_{mejora}$ - La implementación debe ser completamente parametrizable, de forma que todos los componentes susceptibles de ser ajustados mediante parámetros puedan ser establecidos en cada ejecución/problema, y que puedas modificar la inicialización heurística y de pistas de feromona, puedas fácilmente reemplazar las reglas de actualización de feromonas, activar/desactivar la actualización local de actualización de feromona, etc.

[2]:
```
Loc1 = Localizaciones(filename='./data/grafo8cidades.txt')
Loc2 = Localizaciones(filename='./data/US120.txt')
```

Prueba tu implementación tomando instancias de problemas que has utilizado en prácticas anteriores. Lanza varias ejecuciones para verificar que puede resolver el problema con los problemas de las 8 ciudades gallegas (`data/grafo8cidades.txt`) y de las 120 ciudades estadounidenses (`data/US120.txt`).

[3]:
```
def coste(Loc, solucion):
    cost=0
    aux = solucion
    for i in range(0,len(aux)-1):
        cost+= Loc.distancia(aux[i], aux[i+1])
    return cost;
```

[4]:
```
def solucion_voraz(Loc):
        solucion = [0]
        for i in range(1, Loc.nciudades):
            mejor_punto, mejor_coste = None, None
```

```
            for nueva_ciudad in range(1, Loc.nciudades):
                if nueva_ciudad in solucion:
                    continue

                coste_nueva_ciudad = Loc.distancia(solucion[len(solucion) - 1],␣
 ↪nueva_ciudad)

                if not mejor_punto or coste_nueva_ciudad < mejor_coste:
                    mejor_punto, mejor_coste = nueva_ciudad, coste_nueva_ciudad

            solucion.append(mejor_punto)

        solucion = solucion  + [0]
        return solucion
```

[5]:
```
def inicializar_Nij(Loc):
    n = [[0.0 for _ in range(Loc.nciudades)] for _ in range(Loc.nciudades)]
    for i in range(Loc.nciudades):
        for j in range(i + 1, Loc.nciudades):
            n[i][j] = 1 / Loc.distancia(i, j)
            n[j][i] = 1 / Loc.distancia(j, i)
    return n
```

[6]:
```
def inicializar_Tij(Loc):
    cost = coste(Loc, solucion_voraz(Loc))
    aux = 1 / (cost * Loc.nciudades)
    return [[aux for _ in range(Loc.nciudades)] for _ in range(Loc.nciudades)]
```

[21]:
```
def explotacion(ciudades_sin_visitar, ultima_ciudad_visitada, t , n, alpha,␣
 ↪beta):
    ciudad_feromona_x_distancia = []

    for ciudad in ciudades_sin_visitar:
        ciudad_feromona_x_distancia.append((ciudad,␣
 ↪(t[ultima_ciudad_visitada][ciudad] ** alpha) *␣
 ↪(n[ultima_ciudad_visitada][ciudad] ** beta)))

    #Maximo
    #La variable feromona_x_distancia representa la influencia combinada de la␣
 ↪feromona y la distancia. Si alpha es mayor que beta, se da más peso a la␣
 ↪feromona, y si beta es mayor que alpha, se da más peso a la distancia.
    ciudad_feromona_x_distancia = sorted(ciudad_feromona_x_distancia,␣
 ↪key=lambda ciudad_feromona_x_distancia: ciudad_feromona_x_distancia[1],␣
 ↪reverse=True)

    print(ciudad_feromona_x_distancia)
```

```
    return ciudad_feromona_x_distancia[0][0]
```

[46]:
```
def exploracion_dirigida(ciudades_sin_visitar, ultima_ciudad_visitada, t, n,␣
↪alpha, beta):
    sumatorio = 0
    for ciudad in ciudades_sin_visitar:
        sumatorio += (t[ultima_ciudad_visitada][ciudad] ** alpha) *␣
↪(n[ultima_ciudad_visitada][ciudad] ** beta)

    ciudad_probabilidad = []

    for ciudad in ciudades_sin_visitar:
        ciudad_probabilidad.append((ciudad, (t[ultima_ciudad_visitada][ciudad]␣
↪** alpha) * (n[ultima_ciudad_visitada][ciudad] ** beta) / sumatorio))␣
↪#Normaliza las probabilidades


    #Ordenar de menor a mayor el inverso de la probabilidad == probabilidad de␣
↪mayor a menor ciudad_probabilidad = sorted(ciudad_probabilidad,  key=lambda␣
↪ciudad_probabilidad: ciudad_probabilidad[1], reverse=True)
    ciudad_probabilidad = sorted(ciudad_probabilidad, key=lambda␣
↪ciudad_probabilidad: 1 / ciudad_probabilidad[1])
    corte = random.uniform(0, 1) #Entre 0 y 1 por estar normalizadas las␣
↪probabilidades
    siguiente_ciudad = None
    probabilidad_agregada= 0

    for ciudad in ciudad_probabilidad:
        probabilidad_agregada += ciudad[1]
        siguiente_ciudad = ciudad[0]

        if probabilidad_agregada >= corte:
            print(ciudad)
            break

    return siguiente_ciudad
```

[9]:
```
def calcular_soluciones_hormigas(Loc, m, q0, t, n, alpha, beta):
    soluciones = []

    for i in range(m):
        solucion = [0]

        ciudades_sin_visitar = list(range(1, Loc.nciudades))
```

```
        while len(ciudades_sin_visitar) > 0:

            ultima_ciudad_visitada = solucion[len(solucion) - 1]
            q = random.uniform(0, 1)

            if q < q0:
                siguiente_ciudad = explotacion(ciudades_sin_visitar,␣
 ↪ultima_ciudad_visitada, t, n, alpha, beta)
            else:
                siguiente_ciudad = exploracion_dirigida(ciudades_sin_visitar,␣
 ↪ultima_ciudad_visitada, t, n, alpha, beta)

            solucion.append(siguiente_ciudad)

            ciudades_sin_visitar.remove(siguiente_ciudad)

        solucion = solucion + [0]

        soluciones.append((solucion, coste(Loc, solucion)))

    return soluciones
```

[10]:
```
def mejor_solucion(soluciones, mejor_solucion_global):
    mejor_solucion = None

    if mejor_solucion_global is None:
        return soluciones[0]

    if mejor_solucion_global[1] > soluciones[0][1]:
        return soluciones[0]

    return mejor_solucion_global
```

[11]:
```
def actualizar_feromonas_hormigas(mejor_solucion_global, t, p):
    for k in range(1, len(mejor_solucion_global[0])):
                i, j = mejor_solucion_global[0][k - 1],␣
 ↪mejor_solucion_global[0][k]
                t[i][j] = (1 - p) * t[i][j] + p * (1 / mejor_solucion_global[1])
    return t
```

[28]:
```
import random
from tqdm.notebook import trange, tqdm

def colonias_hormigas(Loc, m, q0, alpha, beta, p, iter_min, iter_max,␣
 ↪iter_sin_mejora_limite, ratio_mejora_limite):
    n = inicializar_Nij(Loc); #Matriz de distancia inversa
    t = inicializar_Tij(Loc); #Matriz de feromonas
```

```
    mejor_solucion_global = None

    iter_sin_mejora = 0

    for i in tqdm(range(iter_max), desc=""):
        soluciones = calcular_soluciones_hormigas(Loc, m, q0, t, n, alpha, beta)
        mejor_solucion_global_aux = mejor_solucion(soluciones,␣
↪mejor_solucion_global)

        if mejor_solucion_global is None:
            ratio_mejora = 1
        else:
            ratio_mejora = (mejor_solucion_global[1] /␣
↪mejor_solucion_global_aux[1])

        mejor_solucion_global  = mejor_solucion_global_aux

        t = actualizar_feromonas_hormigas(mejor_solucion_global, t, p)

        if iter_sin_mejora_limite is not None and ratio_mejora >␣
↪ratio_mejora_limite:
            iter_sin_mejora = 0

        elif iter_sin_mejora_limite is not None:
            iter_sin_mejora += 1

        if(i < iter_min):
            continue

        if iter_sin_mejora_limite is not None and iter_sin_mejora >=␣
↪iter_sin_mejora_limite:
                break

    return mejor_solucion_global
```

Pregunta 1. Explica brevemente los detalles relevantes de tu código para entender tu implementación (p.ej., estructura de tu código, funciones, implementación de operadores, etc.). Explica también cómo has realizado la verificación de tu implementación.

1.5 P4.2: Laboratorio

Realiza los laboratorios planteados a continuación para estudiar los efectos de los parámetros de la metaheurística. Para esta práctica, reporta también el coste temporal promedio de resolución.

Pregunta 2. Realiza el estudio de la calidad de la solución variando el tamaño de la colonia: 1, 2, 4, 8, 16, 32, 64, 128,... ¿Qué valor recomendarías para el problema de las 120 ciudades? ¿Por qué?

Pregunta 3. Ahora realiza un estudio similar variando la probabilidad de explotación q0 en el rango 0, …, 0.95 en pasos de 0.05~0.10. ¿Qué valor recomendarías para este parámetro? ¿Por qué?

Pregunta 4. Estudia la influencia de pistas de feromonas e información heurística: =1; =2, …, 5. ¿Qué valor de y recomendarías? ¿Por qué?

Pregunta 5. Finalmente, argumenta de manera crítica las diferencias y similitudes con la metaheurísticas vistas hasta el momento en las interactivas/prácticas anteriores. Discute críticamente acerca de las bondades y limitaciones de las técnicas. Por ejemplo: ¿Eres capaz de resolver instancias de diferente tamaño? ¿Con cuál estás obteniendo mejores soluciones? ¿Cuál parece tener más limitaciones en términos de tiempo de ejecución necesario? ¿Te quedarías con una de las Metaheurísticas en vista de los resultados obtenidos? ¿Cuál/es y por qué?

Apoya todas tus respuestas en datos-gráficos resultantes de tus estudios. Para la pregunta 5 no es necesario volver a tomar datos de otras prácticas, pues puedes tomar los resultados y evidencias obtenidas en prácticas anteriores. No olvides indicar qué valores has establecido para los parámetros relacionados a la condición de parada.

Importante: además de la calidad de la soluciones obtenidas, en esta práctica debes medir tiempos para tomar medidas operativas sobre el número de repeticiones que permitan realizar promedios y prescindir de manera razonada de valores en las series de ejecución que no sean computacionalmente rentables/viables con tu implementación/ordenador (p.ej., tamaños de colonia elevados pueden tomar mucho tiempo para resolver).

[194]:
```
# puedes añadir aquí la implementación o cualquier código que necesites para␣
 ↪responder las preguntas
# utiliza tantas celdas como necesites, pero recuerda que el informe final que␣
 ↪resuma las cuestiones
# no debe superar las 1200 palabras.
```

Respuestas y evaluación Recordatorio: No olvides escribir tu nombre y apellidos en la segunda celda de este documento. La respuestas a las preguntas deben venir acompañadas de las implementaciones necesarias para su respuesta.

P4.1: Implementación básica (5 puntos)

La implementación básica se evaluará mediante un cuestionario automático de evaluación. Este lo realizarás en la primera sesión tras la entrega, y se centrará en la resolución por tu parte de diversas cuestiones prácticas relacionadas con la implementación realizada, pudiendo ser necesaria la ejecución, adaptación y modificación de la misma.

Pregunta 1.

Independientemente del cuestionario automático de evaluación, considera siempre que las preguntas planteadas en el notebook deben ser respondidas también. Esas preguntas generales están diseñadas para formarte, y te servián para razonar y reflexionar sobre el tema, así como también para fomentar una discusión constructiva con los docentes en caso de dudas.

P4.2: Laboratorio (5 puntos)

El informe a elaborar no debe exceder la longitud máxima de 1200 palabras.

Aclaraciones: La evaluación de esta parte se llevará a cabo en términos de la completitud y correctitud del laboratorio implementado, así como de la calidad del propio informe, que debe ser conciso y preciso, pudiendo acompañarse de gráficas y tablas que faciliten y fundamenten la explicación e argumentación. Es muy importante explicar de manera clara, precisa y fundamentada. Se reservará hasta un punto que se asignará en términos de la calidad de la mejor solución obtenida entre el conjunto de las prácticas entregadas (es por ello que no debes olvidar marcar en tu informe muy claramente cuál ha sido tu mejor solución y con qué configuración/versión).

Pregunta 2

Pregunta 3

Pregunta 4

Pregunta 5

[18]:
```
import statistics
import time
from tqdm.notebook import trange, tqdm
def pruebas(nombre_prueba, algoritmo, Loc, m, q0, alpha, beta, p, iter_min,␣
 ↪iter_max, iter_sin_mejora_limite, ratio_mejora_limite):
    tiempos = []
    resultados = []
    print(nombre_prueba)

    for i in tqdm(range(5), desc=nombre_prueba):
        inicio = time.time()
        a = algoritmo(Loc, m, q0, alpha, beta, p, iter_min, iter_max,␣
 ↪iter_sin_mejora_limite, ratio_mejora_limite)
        delta_time = time.time() - inicio
        tiempos.append(delta_time)
        resultados.append(a[1])

    mediaT = statistics.mean(tiempos)
    print("La media de los tiempos es: ", mediaT)
    desvtipT = statistics.stdev(tiempos)
    print("La desviacion de los tiempos es: ", desvtipT)

    mediaR = statistics.mean(resultados)
    print("La media de los resultados es: ", mediaR)
    desvtipR = statistics.stdev(resultados)
    print("La desviacion de los resultados es: ", desvtipR)
    print("-------------\n")

    return (resultados, tiempos, mediaR, mediaT)
```

[19]:
```
import matplotlib.pyplot as plt
import numpy as np
```

```
def graficar(resultados, tiempos, mediasR, mediasT, test, titulo):
    # Crear una figura y una matriz de subgráficas con 1 fila y 2 columnas
    fig, axs = plt.subplots(1, 2, figsize=(10, 4))

    # Agregar título a la figura
    fig.suptitle(titulo, fontsize=16)

    # Grafica 1
    axs[0].set_title("Coste")
    axs[0].set_xticks(test)

    for pob in tqdm(test, leave=None):
        axs[0].plot(test, resultados, "bo")

    # Grafica 2
    axs[1].set_title("Tiempo")
    axs[1].set_xticks(test)

    for pob in tqdm(test, leave=None):
        axs[1].plot(test, tiempos, "bo")

    # Crear una línea que une las medias de a2 y a3
    axs[0].plot(test, mediasR, 'r-', label='Media Resultados')
    axs[1].plot(test, mediasT, 'r-', label='Media Tiempo')

    # Ajusta el diseño para evitar solapamientos
    plt.tight_layout()

    # Muestra las gráficas
    plt.show()
```

2 Informe de Laboratorio

2.1 Pregunta 2. Tamaño de la colonia

2.1.1 Parte Experimental

[27]:
```
test = [1, 2, 4, 8, 16, 32, 64, 128]
a0 = []
a1 = []
a2 = []
a3 = []
for i in tqdm(test, desc="Algoritmo de Sistemas de Colonias de Hormigas"):
    per = pruebas("Algoritmo de Sistemas de Colonias de Hormigas con n=" +␣
 ↪str(i), colonias_hormigas, Loc2, i, 0.5, 1, 3, 0.01, 100, 500, 30, 0.75 )
    a0.append(per[0])
    a1.append(per[1])
```

```
    a2.append(per[2])
    a3.append(per[3])
```

```
Algoritmo de Sistemas de Colonias de Hormigas:   0%|          | 0/8 [00:00<?, ?
 ↪it/s]

Algoritmo de Sistemas de Colonias de Hormigas con n=1

Algoritmo de Sistemas de Colonias de Hormigas con n=1:   0%|          | 0/5 [00:
 ↪00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

La media de los tiempos es:  1.4532214641571044
La desviacion de los tiempos es:  0.0821554725786651
La media de los resultados es:  25636.696231274622
La desviacion de los resultados es:  742.0262524225857
-------------

Algoritmo de Sistemas de Colonias de Hormigas con n=2

Algoritmo de Sistemas de Colonias de Hormigas con n=2:   0%|          | 0/5 [00:
 ↪00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

La media de los tiempos es:  3.137051010131836
La desviacion de los tiempos es:  0.06370491102484904
La media de los resultados es:  25984.863143789673
La desviacion de los resultados es:  1110.9328320380037
-------------

Algoritmo de Sistemas de Colonias de Hormigas con n=4

Algoritmo de Sistemas de Colonias de Hormigas con n=4:   0%|          | 0/5 [00:
 ↪00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]
```

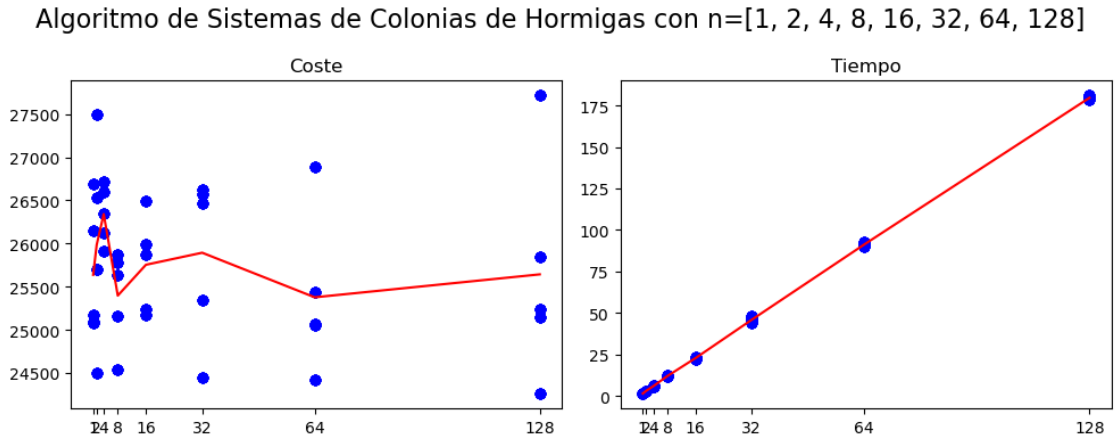
```
  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

La media de los tiempos es:  5.915947723388672
La desviacion de los tiempos es:  0.20944146646158085
La media de los resultados es:  26341.31355726929
La desviacion de los resultados es:  331.8010746169509
-------------

Algoritmo de Sistemas de Colonias de Hormigas con n=8

Algoritmo de Sistemas de Colonias de Hormigas con n=8:   0%|          | 0/5 [00:
 ↪00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

La media de los tiempos es:  11.995638132095337
La desviacion de los tiempos es:  0.3096939991946907
La media de los resultados es:  25396.048462932595
La desviacion de los resultados es:  553.3345797523756
-------------

Algoritmo de Sistemas de Colonias de Hormigas con n=16

Algoritmo de Sistemas de Colonias de Hormigas con n=16:   0%|          | 0/5 [00:
 ↪00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

La media de los tiempos es:  22.804132843017577
La desviacion de los tiempos es:  0.5661996404262818
La media de los resultados es:  25753.554901555657
La desviacion de los resultados es:  549.3724908802919
-------------

Algoritmo de Sistemas de Colonias de Hormigas con n=32
```

```
Algoritmo de Sistemas de Colonias de Hormigas con n=32:   0%|          | 0/5 [00:
↪00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

La media de los tiempos es:  45.9218147277832
La desviacion de los tiempos es:  1.7690732433596772
La media de los resultados es:  25893.348888732122
La desviacion de los resultados es:  967.1034567517481
-------------

Algoritmo de Sistemas de Colonias de Hormigas con n=64

Algoritmo de Sistemas de Colonias de Hormigas con n=64:   0%|          | 0/5 [00:
↪00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

La media de los tiempos es:  91.28020253181458
La desviacion de los tiempos es:  1.0354654935567984
La media de los resultados es:  25376.483994788385
La desviacion de los resultados es:  921.1017874199181
-------------

Algoritmo de Sistemas de Colonias de Hormigas con n=128

Algoritmo de Sistemas de Colonias de Hormigas con n=128:   0%|          | 0/5␣
↪[00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

La media de los tiempos es:  179.56567454338074
La desviacion de los tiempos es:  0.8619659848761547
```

```
La media de los resultados es:  25642.4077122344
La desviacion de los resultados es:  1290.160754662639
-------------
```

[28]:
```
test = [1, 2, 4, 8, 16, 32, 64, 128]

graficar(a0,a1,a2,a3,test,"Algoritmo de Sistemas de Colonias de Hormigas con␣
 ↪n=[1, 2, 4, 8, 16, 32, 64, 128]")
```

```
  0%|          | 0/8 [00:00<?, ?it/s]
  0%|          | 0/8 [00:00<?, ?it/s]
```



2.1.2 Analisis de Resultados

Para analizar la influencia del número de hormigas sobre el coste de la solución, se han realizado diferentes ejecuciones con los siguientes tamaños de la colonia de hormigas: 1, 2, 4, 8, 16, 32, 64, 128. De esta forma se ha decidido generar dos gráficas para analizar el coste temporal para cada uno de los tamaños y su correspondiente coste:

Con todo esto, se puede deducir que el coste de la solución va a estar en el intervalor de [25.500, 26.500] km y por lo tanto, no es un valor con excesiva importancia dentro del problema. Sin embargo, su coste temporal aumenta cuanto mayor es el número de hormigas, por lo tanto viendo que el coste de la solución va estar en el intervalo previo, se tomará un valor pequeño para reducir el tiempo de ejecución. Se toma como mejor m = 16

2.2 Pregunta 3. Probabilidad de explotación

2.2.1 Parte Experimental

[30]:
```
test = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]
a00 = []
a11 = []
a22 = []
a33 = []
for i in tqdm(test, desc="Algoritmo de Sistemas de Colonias de Hormigas"):
    per = pruebas("Algoritmo de Sistemas de Colonias de Hormigas con q0=" +␣
 ↪str(i), colonias_hormigas, Loc2, 16, i, 1, 3, 0.01, 100, 500, 30, 0.75 )
    a00.append(per[0])
    a11.append(per[1])
    a22.append(per[2])
    a33.append(per[3])
```

```
Algoritmo de Sistemas de Colonias de Hormigas:   0%|          | 0/11 [00:00<?, ?
 ↪it/s]

Algoritmo de Sistemas de Colonias de Hormigas con q0=0

Algoritmo de Sistemas de Colonias de Hormigas con q0=0:   0%|          | 0/5 [00:
 ↪00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

La media de los tiempos es:  27.379718828201295
La desviacion de los tiempos es:  1.944547448469145
La media de los resultados es:  27961.705601952413
La desviacion de los resultados es:  957.6249593748521
-------------


Algoritmo de Sistemas de Colonias de Hormigas con q0=0.1

Algoritmo de Sistemas de Colonias de Hormigas con q0=0.1:   0%|          | 0/5␣
 ↪[00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]
```

```
La media de los tiempos es:  25.972693014144898
La desviacion de los tiempos es:  0.4702639050480725
La media de los resultados es:  26460.914929860104
La desviacion de los resultados es:  357.98905160508207
-------------

Algoritmo de Sistemas de Colonias de Hormigas con q0=0.2

Algoritmo de Sistemas de Colonias de Hormigas con q0=0.2:   0%|          | 0/5␣
 ↪[00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

La media de los tiempos es:  24.430014657974244
La desviacion de los tiempos es:  0.2695123234299044
La media de los resultados es:  26466.51011632863
La desviacion de los resultados es:  494.79896819475624
-------------

Algoritmo de Sistemas de Colonias de Hormigas con q0=0.3

Algoritmo de Sistemas de Colonias de Hormigas con q0=0.3:   0%|          | 0/5␣
 ↪[00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

La media de los tiempos es:  23.21363353729248
La desviacion de los tiempos es:  0.3020148423281723
La media de los resultados es:  26169.016540191184
La desviacion de los resultados es:  762.7401231046165
-------------

Algoritmo de Sistemas de Colonias de Hormigas con q0=0.4

Algoritmo de Sistemas de Colonias de Hormigas con q0=0.4:   0%|          | 0/5␣
 ↪[00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]
```

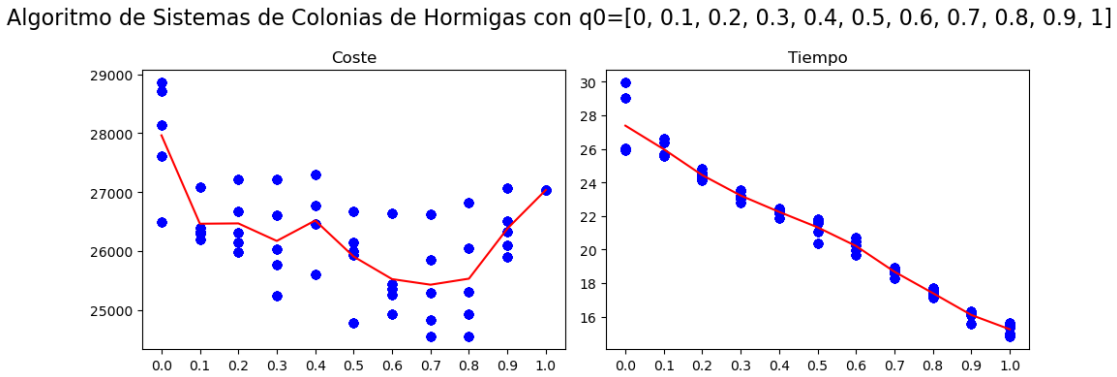
```
  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

La media de los tiempos es:  22.24108386039734
La desviacion de los tiempos es:  0.22987009992965587
La media de los resultados es:  26515.5566929516
La desviacion de los resultados es:  610.6426622402969
-------------

Algoritmo de Sistemas de Colonias de Hormigas con q0=0.5

Algoritmo de Sistemas de Colonias de Hormigas con q0=0.5:   0%|          | 0/5␣
 ↪[00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

La media de los tiempos es:  21.321234607696532
La desviacion de los tiempos es:  0.5979697432397071
La media de los resultados es:  25903.670682295346
La desviacion de los resultados es:  694.8071870232817
-------------

Algoritmo de Sistemas de Colonias de Hormigas con q0=0.6

Algoritmo de Sistemas de Colonias de Hormigas con q0=0.6:   0%|          | 0/5␣
 ↪[00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

La media de los tiempos es:  20.20814256668091
La desviacion de los tiempos es:  0.4173501767628777
La media de los resultados es:  25520.683180484062
La desviacion de los resultados es:  657.8841034690042
-------------

Algoritmo de Sistemas de Colonias de Hormigas con q0=0.7
```

```
Algoritmo de Sistemas de Colonias de Hormigas con q0=0.7:   0%|          | 0/5␣
 ↪[00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

La media de los tiempos es:  18.673994922637938
La desviacion de los tiempos es:  0.25155830946646673
La media de los resultados es:  25425.9901913084
La desviacion de los resultados es:  826.7799234271619
-------------

Algoritmo de Sistemas de Colonias de Hormigas con q0=0.8

Algoritmo de Sistemas de Colonias de Hormigas con q0=0.8:   0%|          | 0/5␣
 ↪[00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

La media de los tiempos es:  17.398238134384155
La desviacion de los tiempos es:  0.22097995899084907
La media de los resultados es:  25528.766642018272
La desviacion de los resultados es:  909.0433863988044
-------------

Algoritmo de Sistemas de Colonias de Hormigas con q0=0.9

Algoritmo de Sistemas de Colonias de Hormigas con q0=0.9:   0%|          | 0/5␣
 ↪[00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

La media de los tiempos es:  16.09661955833435
La desviacion de los tiempos es:  0.3038915541892269
```

```
La media de los resultados es:  26378.62685285071
La desviacion de los resultados es:  447.2312790845889
-------------

Algoritmo de Sistemas de Colonias de Hormigas con q0=1

Algoritmo de Sistemas de Colonias de Hormigas con q0=1:   0%|          | 0/5 [00:
 ↪00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

La media de los tiempos es:  15.242986011505128
La desviacion de los tiempos es:  0.34045161624869635
La media de los resultados es:  27033.84988200304
La desviacion de los resultados es:  0.0
-------------
```

[33]:
```
test = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]
graficar(a00,a11,a22,a33,test,"Algoritmo de Sistemas de Colonias de Hormigas␣
 ↪con q0=[0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]")
```

```
  0%|          | 0/11 [00:00<?, ?it/s]

  0%|          | 0/11 [00:00<?, ?it/s]
```



2.2.2 Analisis de Resultados

Para analizar la influencia de q0, probabilidad de explotacion, se han realizado una serie de experiementos similares a los realizados en la pregunta anterior. Para ello se han realizado diferentes

ejecuciones con los siguientes valores para q0: 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1 y se ha decidido graficar los resultados para una mejor comprensión de los resultados:

Gracias a la gráfica del coste temporal se observa que cuanto mayor es q0, cuanto más se explote, menos tiempo de ejecución se necesita. Esto es obvio ya que la función heurística de exploración dirigida tiene más coste computacional. Utilizando un q0 comprendido entre [0.6, 0.8] se obtienen los mejores resultados, por lo tanto se tomará un valor alto dentro de ese intervalo, con el fin de obtener un buen ratio coste temporal y coste de la solución. Por ende se tomará el valor de 0.75 para q0.

2.3 Pregunta 4. Pistas de feromonas e información heurística

2.3.1 Parte Experimental

[34]:
```
test = [1,2,3,4,5]
a000 = []
a111 = []
a222 = []
a333 = []
for i in tqdm(test, desc="Algoritmo de Sistemas de Colonias de Hormigas"):
    per = pruebas("Algoritmo de Sistemas de Colonias de Hormigas con alfa=1 y␣
 ↪beta=" + str(i), colonias_hormigas, Loc2, 16, 0.75, 1, i, 0.01, 100, 500,␣
 ↪30, 0.75 )
    a000.append(per[0])
    a111.append(per[1])
    a222.append(per[2])
    a333.append(per[3])
```

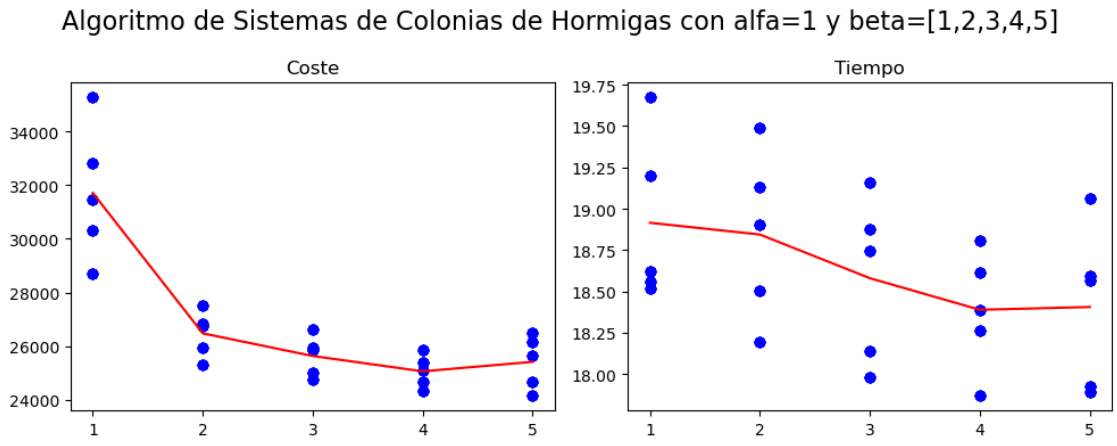
```
Algoritmo de Sistemas de Colonias de Hormigas:   0%|          | 0/5 [00:00<?, ?
 ↪it/s]

Algoritmo de Sistemas de Colonias de Hormigas con alfa=1 y beta=1

Algoritmo de Sistemas de Colonias de Hormigas con alfa=1 y beta=1:   0%|       ␣
 ↪ | 0/5 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

La media de los tiempos es:  18.914903450012208
La desviacion de los tiempos es:  0.5060868445650201
La media de los resultados es:  31708.288190657404
La desviacion de los resultados es:  2500.6906281474767
-------------
```

```
Algoritmo de Sistemas de Colonias de Hormigas con alfa=1 y beta=2

Algoritmo de Sistemas de Colonias de Hormigas con alfa=1 y beta=2:   0%|          ␣
 ↪ | 0/5 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

La media de los tiempos es:  18.843775701522826
La desviacion de los tiempos es:  0.5100938596974176
La media de los resultados es:  26479.489344723475
La desviacion de los resultados es:  856.7484472429635
-------------

Algoritmo de Sistemas de Colonias de Hormigas con alfa=1 y beta=3

Algoritmo de Sistemas de Colonias de Hormigas con alfa=1 y beta=3:   0%|          ␣
 ↪ | 0/5 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

La media de los tiempos es:  18.5797278881073
La desviacion de los tiempos es:  0.500913660349271
La media de los resultados es:  25636.660834938604
La desviacion de los resultados es:  755.2051045477599
-------------

Algoritmo de Sistemas de Colonias de Hormigas con alfa=1 y beta=4

Algoritmo de Sistemas de Colonias de Hormigas con alfa=1 y beta=4:   0%|          ␣
 ↪ | 0/5 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]
```

```
La media de los tiempos es:  18.388767147064208
La desviacion de los tiempos es:  0.35774949810015205
La media de los resultados es:  25064.752033418525
La desviacion de los resultados es:  598.475327510329
-------------

Algoritmo de Sistemas de Colonias de Hormigas con alfa=1 y beta=5

Algoritmo de Sistemas de Colonias de Hormigas con alfa=1 y beta=5:   0%|
 ↪ | 0/5 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

La media de los tiempos es:  18.405608081817626
La desviacion de los tiempos es:  0.496646634342197
La media de los resultados es:  25418.61574010119
La desviacion de los resultados es:  989.4118152763693
-------------
```

[37]:
```
test = [1,2,3,4,5]
graficar(a000,a111,a222,a333,test,"Algoritmo de Sistemas de Colonias de
 ↪Hormigas con alfa=1 y beta=[1,2,3,4,5]")
```

```
  0%|          | 0/5 [00:00<?, ?it/s]

  0%|          | 0/5 [00:00<?, ?it/s]
```



[40]:
```
test = [0.2,0.4,0.6,0.8,1,2]
a0000 = []
a1111 = []
a2222 = []
a3333 = []
for i in tqdm(test, desc="Algoritmo de Sistemas de Colonias de Hormigas"):
    per = pruebas("Algoritmo de Sistemas de Colonias de Hormigas con alfa="+␣
 ↪str(i) +" y beta=1" , colonias_hormigas, Loc2, 16, 0.75, i, 1, 0.01, 100,␣
 ↪500, 30, 0.75 )
    a0000.append(per[0])
    a1111.append(per[1])
    a2222.append(per[2])
    a3333.append(per[3])
```

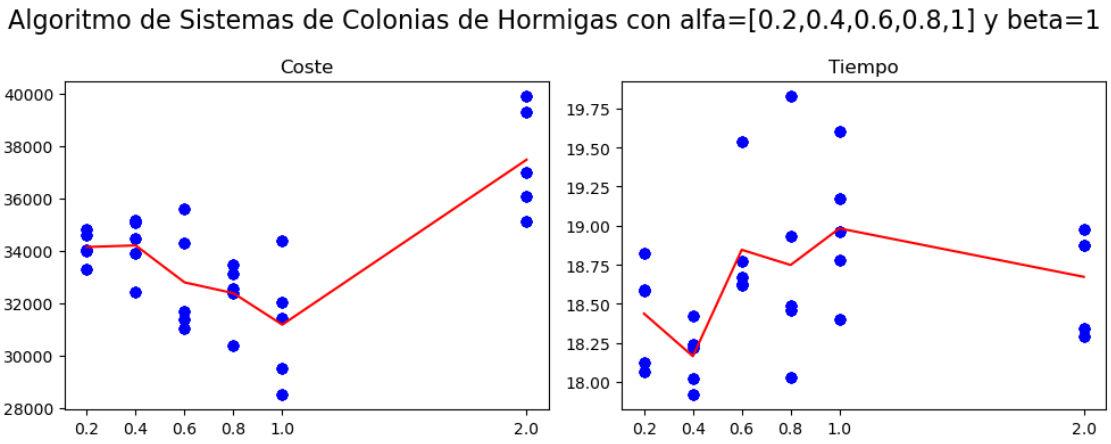
```
Algoritmo de Sistemas de Colonias de Hormigas:   0%|          | 0/6 [00:00<?, ?
 ↪it/s]

Algoritmo de Sistemas de Colonias de Hormigas con alfa=0.2 y beta=1

Algoritmo de Sistemas de Colonias de Hormigas con alfa=0.2 y beta=1:   0%|      ␣
 ↪   | 0/5 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

La media de los tiempos es:  18.4371120929718
La desviacion de los tiempos es:  0.32790529745216934
La media de los resultados es:  34134.001570385866
La desviacion de los resultados es:  585.8852799840303
-------------

Algoritmo de Sistemas de Colonias de Hormigas con alfa=0.4 y beta=1

Algoritmo de Sistemas de Colonias de Hormigas con alfa=0.4 y beta=1:   0%|      ␣
 ↪   | 0/5 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

La media de los tiempos es:  18.1641508102417
La desviacion de los tiempos es:  0.19813897641422995
```

```
La media de los resultados es:  34195.010655154976
La desviacion de los resultados es:  1126.8393606945526
-------------

Algoritmo de Sistemas de Colonias de Hormigas con alfa=0.6 y beta=1

Algoritmo de Sistemas de Colonias de Hormigas con alfa=0.6 y beta=1:   0%|
 ↪    | 0/5 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

La media de los tiempos es:  18.845052766799927
La desviacion de los tiempos es:  0.3924523862428541
La media de los resultados es:  32785.057644702945
La desviacion de los resultados es:  2030.5711505910785
-------------

Algoritmo de Sistemas de Colonias de Hormigas con alfa=0.8 y beta=1

Algoritmo de Sistemas de Colonias de Hormigas con alfa=0.8 y beta=1:   0%|
 ↪    | 0/5 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

La media de los tiempos es:  18.747728013992308
La desviacion de los tiempos es:  0.6812621818616708
La media de los resultados es:  32372.625044325985
La desviacion de los resultados es:  1208.828417916904
-------------

Algoritmo de Sistemas de Colonias de Hormigas con alfa=1 y beta=1

Algoritmo de Sistemas de Colonias de Hormigas con alfa=1 y beta=1:   0%|
 ↪ | 0/5 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]
```

```
  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

La media de los tiempos es:  18.98203573226929
La desviacion de los tiempos es:  0.4462686750619333
La media de los resultados es:  31171.657338793186
La desviacion de los resultados es:  2290.4565376049054
-------------

Algoritmo de Sistemas de Colonias de Hormigas con alfa=2 y beta=1

Algoritmo de Sistemas de Colonias de Hormigas con alfa=2 y beta=1:   0%|
 ↪ | 0/5 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

  0%|          | 0/500 [00:00<?, ?it/s]

La media de los tiempos es:  18.672167205810545
La desviacion de los tiempos es:  0.3259292580362998
La media de los resultados es:  37466.5426776699
La desviacion de los resultados es:  2047.1100545807467
-------------
```

[41]:
```
test = [0.2,0.4,0.6,0.8,1,2]
graficar(a0000,a1111,a2222,a3333,test,"Algoritmo de Sistemas de Colonias de
 ↪Hormigas con alfa=[0.2,0.4,0.6,0.8,1] y beta=1")
```

```
  0%|          | 0/6 [00:00<?, ?it/s]

  0%|          | 0/6 [00:00<?, ?it/s]
```



2.3.2 Analisis de Resultados

Para analizar la influencia de pistas de feromonas se han realizado experimentos con alpha y beta, dejando uno constante en 1 y dandole valores al otro. Tras una primeras pruebas se observa que se obtienen mejores soluciones cuando el valor de alpha es cercano a 1 y beta es supeior a 1, pero ambos cambios implican un mayor coste en el tiempo de ejecución, coste temporal. Para poder analizar mejor ambos experimentos (alpha cercano a 1 y aumento del valor de beta) se han generado las siguientes gráficas:

Con la ayuda de estas gráficas se puede deducir que beta tiene un mayor impacto en el valor del coste de la solución y que a partir de beta = 4 el coste ya se estabiliza y no es necesario aumentar dicho valor porque aumentaría también el coste temporal. Por otro lado, el valor de alpha tiene que ser <= 1 y la mejor opción en cuanto al ratio de coste temporal y coste de la solución es alha = 1. Por lo tanto, alpha = 1 y beta = 4.

Con todos los parámetros obtenidos a lo largo de la experiemtación se han obtenido soluciones entre 24.000km y 25.000km de media, aunque existen valores atípicos por debajo de 24.000km

2.4 Pregunta 5. Diferencias y similitudes de todas las metaheurísticas

Tabla de tiempos y resultados de todas las metaheurísticas:

Metaheurística	Mejores resultados (kilómetros)	Mejores tiempos (segundos)
Temple Simulado	[35.000, 40.000]	[0.5, 1]
Búsqueda Tabú	[26.000, 30.000]	[200, 220]
Algoritmo Genético	[31.000, 35.000]	[50, 100]
Algoritmo de Sistemas de Colonias de Hormigas	[24.000, 27.000]	[18, 19]

Ordenado a través de mejores resultados:

Metaheurística	Mejores resultados (kilómetros)	Mejores tiempos (segundos)
Algoritmo de Sistemas de Colonias de Hormigas	[24.000, 27.000]	[18, 19]
Búsqueda Tabú	[26.000, 30.000]	[200, 220]
Algoritmo Genético	[31.000, 35.000]	[50, 100]
Temple Simulado	[35.000, 40.000]	[0.5, 1]

Ordenado a través de los mejores tiempos:

Metaheurística	Mejores resultados (kilómetros)	Mejores tiempos (segundos)
Temple Simulado	[35.000, 40.000]	[0.5, 1]
Algoritmo de Sistemas de Colonias de Hormigas	[24.000, 27.000]	[18, 19]
Algoritmo Genético	[31.000, 35.000]	[50, 100]
Búsqueda Tabú	[26.000, 30.000]	[200, 220]

Analizando las tablas realizadas previamente, se puede afirmar:

- A pesar de que el **Temple Simulado** tiene un tiempo de ejecución muy bajo, tiene un problema muy notorio y es que obtiene resultados entre un 40 y 50% peor que los del Algoritmo de Sistemas de Colonias de Hormigas y que además su desviación típica es muy alta, por lo que no siempre te asegura un buen resultado.
- La **Búsqueda Tabú** obtiene buenos resultados, cercanos al Algoritmo de Sitemas de Colonias de Hormigas, pero necesita diez veces más de tiempo de ejecución.
- El **Algoritmo Genético**, obtiene peores resultados que la mayoría y requiere de un alto coste temporal.
- La **mejor metaheurística para el problema de Viajante Comercio y las experiementaciones realizadas es el Algoritmo de Sistemas de Colonias de Hormigas**, ya que obtiene el mejor resultado y en cuanto a coste temporal es la segunda mejor, por debajo de Temple Simulado.