UNIVERSIDAD
ANDRES BELLO

Universidad Andrés Bello
Facultad de Ingeniería

Ingeniería Civil Informática

TAREA5

PROBLEM SOLVING

SISTEMAS INTELIGENTES

DAVID ANDRÉS MOLINA GARRIDO

Profesor Alejandro Figueroa
Ayudante Jean Contreras

Santiago - Chile.

Noviembre, 2017.

Introducción

El presente informe tiene por objetivo resolver el problema del vendedor viajero (TSP por sus siglas en inglés), este problema NP-Completo consiste en que dado un número "n" de ciudades, un hombre debe visitarlas todas regresando a la ciudad de partida. Bajo las condiciones de visitar cada ciudad al menos una vez y hacerlo de la manera que genere el menor coste posible. Además, este problema será resuelto con el algoritmo Ant Colony Systems (ACS).

En ACS, un conjunto de agentes cooperativos llamados hormigas cooperan para encontrar buenas soluciones para TSP. Las hormigas cooperan utilizando una forma indirecta de comunicación mediante feromonas que son depositadas en los bordes del gráfico TSP mientras crean soluciones.

Utilizando el algoritmo ACS se encontrará el mejor tour, por otra parte, se analizará cómo influye la variación de los valores de las variables utilizadas.

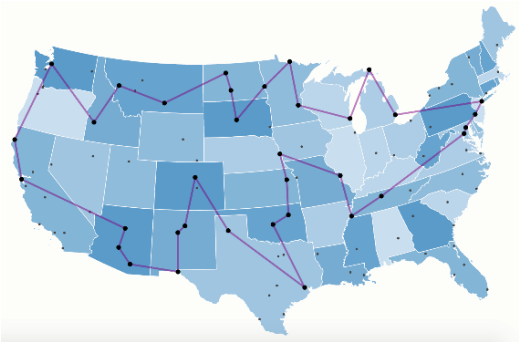

Imagen1: Problema del Vendedor Viajero (TSP).

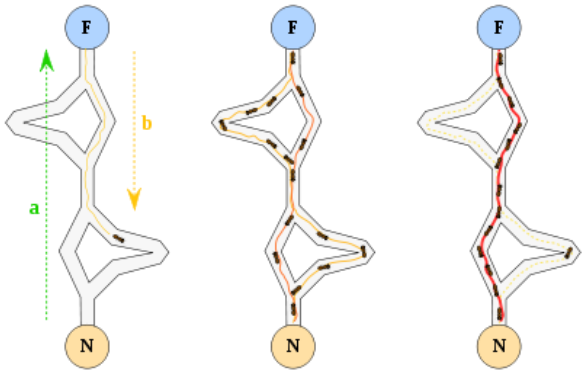

Imagen2: Metaheurística ACS.

Descripción del Problema

Para el presente informe se espera mediante el diseño de un algoritmo ACS obtener el mejor tour para un numero de "n" ciudades, las cuales se tienen que recorrer como se indica en el problema del vendedor viajero.

Para resolver el problema en el algoritmo ACS se utilizaron las librerías públicas con los archivos en formato de entrada .tsp y de salida en .tour como se explica en el siguiente link: https://acrogenesis.com/or-tools/documentation/user_manual/manual/tsp/tsp.html

Lo que se probará en este problema del vendedor viajero con ACS será lo siguiente:

a) Probar varias instancias de diferentes tamaños y analizar los resultados en términos de calidad de la solución versus tiempo de ejecución. También entre número de iteraciones y tiempo de ejecución.

b) Probar diferentes parámetros de la estrategia seleccionada de manera de optimizar el desempeño.

c) Analizar las mejores soluciones encontradas a la luz de las mejores provistas por la biblioteca.

d) Proponer mejoras que se podrían hacer para mejorar el desempeño.

Análisis del Resultado

Para el algortimo ACS en el problema del vendedor viajero se utilizaron diferentes entradas, con diferentes números de ciudades en cada uno para posteriormente realizar un análisis de las respectivas salidas del algoritmo.

El algortimo ACS utilizado cuenta con variables las cuales proporcionan los parámetros para el funcionamiento del algoritmo en el problema del vendedor viajero. Primeramente se utilizaron los parámetros que "a priori" (y teóricamente) son los mejores para resolver este tipo de problema.

Los parámetros utilizados inicialmente fueron:

-Número de Ciudades: 48 (Depende del valor de entrada).
-Iteraciones: 10.
-Hormigas: 10.
-Alpha: 0.1.
-Beta: 2.
-Q: 0.1.
-Ro: 0.1.
-Feromonas máximas: 2.

Tiempo: 1.10959
Distancia: 42111

Para analizar primeramente las soluciones que nos da el algoritmo al enfrentarse a este problema hay que mencionar que con esa "configuración" de parámetros se comenzó, o sea, se tomó como base.

Se fue variando en primer caso el parámetro Beta, éste determina la importancia relativa de la feromona versus la distancia. Para este caso se hicieron pruebas modificando a partir del caso base, sólo este parámetro y estos fueron los resultados:

Con parámetro = 5 :
-Tiempo: 0.8335
-Distancia: 90434

Con parámetro = 10 :
-Tiempo: 1.11244
-Distancia: 39403,9

Con parámetro = 50 :
-Tiempo: 0.11959
-Distancia: 42111

Como muestran los resultados, vemos que la importancia de la feromona a la hora de elegir el camino es relevante, pero sigue una curva la cual tiene un punto de inflexión, bajo diferentes pruebas se determina que con el parámetro en 10 es la mejor prueba ya que luego el tiempo y la distancia va en aumento. Por otro lado, se denota que el tiempo no es directamente proporcional a la distancia del tour.

Para el parámetro Q, que representa la importancia de la explotación vs la exploración determinó que al cambiarlo desde nuestro caso base, no provocó mayores cambios en el resultado del tiempo ni de la distancia.

Para el parámetro Alpha, el cual es un parámetro de decaimiento, con los parámetros base:

Con parámetro = 0.1:
-Tiempo: 1.10562
-Distancia: 42111

Con parámetro = 0.5:
-Tiempo: 1.11455
-Distancia: 45326

Con parámetro = 0.9:
-Tiempo: 0.11329
-Distancia: 45396

Como vemos, dejando el parámetro como estaba en el caso base (el más optimo teóricamente) fue efectivamente el que nos entrega mejores resultados.

En el caso de la cantidad de hormigas, se aumentaron al doble (20) pero solo vemos que aumento el tiempo a 2.20471 (casi el doble) pero mantuvo la misma distancia en la solución, por lo que vemos que lo teóricamente optimo nuevamente fue efectivo.

Para el valor que Ro, el cual representa la evaporación de las feromonas (0<Ro<=1) en los caminos, se determinó que el mantenerla o aumentarla no provocó absolutamente ningún cambio, lo que si provocó con otra cantidad de ciudades.

Por otra parte, se realizó otra se realizó otra prueba, en este caso con 280 ciudades, las cuales determinaron lo siguiente:

En esta prueba se realizaron absolutamente las mismas variaciones, partiendo de la base teóricamente optima, y los resultados fueron un tanto diferentes, si bien es cierto los datos con 48 ciudades fueron muy parecidos a lo teóricamente optimo, en el caso de más ciudades (280) hubo variaciones:

Con respecto a Beta se encontró que lo ideal fue 8. Y en el caso de la evaporación de las feromonas (Ro) sí importó el subir el valor, pero tan sólo a 0.3, los otros parámetros siguieron la misma

Entonces, como podemos ver en estas dos entradas (con diferentes matices), nos indican que en el caso de las iteraciones y la cantidad de hormigas no son uno de los parámetros donde debemos de poner mayor énfasis, esto ya que como vimos en las pruebas no provocan cambios sustantivos y manteniendo los teóricamente óptimos es lo mejor.

Por otra parte, la calidad de la solución vs el tiempo de ejecución de el algoritmo si se hace notar mucho y sobre todo en el parámetro Beta, el cual determina la importancia relativa de la feromona versus la distancia, o sea le asigna mayor relevancia a la cantidad previamente colocada en el parámetro Ro (la evaporación de las feromonas).

Por último, las soluciones presentadas en:
http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/STSP.html ,nos indican que nuestras soluciones de los tour no son muy acorde a ellas, ya que en algunas casos la diferencia no fue menor, esto se debe principalmente al algoritmo utilizado.

Conclusión

En conclusión, podemos ver que el parámetro Beta es sumamente relevante a la hora de decidir para las hormigas porque camino ir, esto ya que realza y enfatiza la importancia de las feromonas vs la distancia, haciendo que ellas sigan esto en la mayoría de las ocasiones, pero en contraste al usar una muy alta las mediciones arrojan resultados negativos. Además, vemos que como en las pruebas expuestas la cantidad de hormigas no es muy relevante cuando se tiene un "mapa" de tamaño medio como los que usamos en las pruebas descritas, por lo que a tamaños más grandes si sería necesario.

Por otra parte, vemos que el tiempo no es directamente proporcional a la mejor solución, es decir, un tiempo menor no nos da peor o mejor resultado necesariamente.

Una propuesta que haría mejor el algoritmo y el problema en sí, sería que el algoritmo vaya guardando ciertos parámetros que mejoran el tour y que en cada prueba estos converjan hasta un óptimo para una entrada determinada.

Anexo-Código

Para hacer funcionar el código, se debe correr un makefile, en éste se incluyen los argumentos como la entrada y la salida y los archivos a compilar:

```
EXE = main
CFLAGS = -Wall -O2 -std=c++11
C = g++
all: obj/main.o obj/ACO.o obj/lectura.o
    $(C) $(CFLAGS) -o $(EXE) obj/main.o obj/ACO.o obj/lectura.o

obj/main.o: main.cpp glob.h
    $(C) -c -o obj/main.o main.cpp

obj/ACO.o: ACO.cpp ACO.h glob.h
    $(C) -c -o obj/ACO.o ACO.cpp

obj/lectura.o: lectura.cpp lectura.h glob.h
    $(C) -c -o obj/lectura.o lectura.cpp

clean:
    rm -fr $(EXE)
    rm -fr obj/*

run:
    ./$(EXE) Entrada/att48.tsp Salida/BestRute_att48.tsp
```

Ejecutar la limpieza de los archivos compilados:
$ make clean

Compilar:
$ make

Ejecutar:
$ make run

Los archivos.cpp se encuentran comentados para entender su funcionamiento.