Recodido simulado con aceptación por umbrales: problema del agente viajero

Angie Daniela Velásquez Garzón

1. Introducción

El recocido simulado es método de optimización basado en el proceso de recocido del acero, es un algoritmo de búsqueda local cuyo objetivo es encontrar una buena solución en un espacio de búsqueda grande. Existen diferentes variantes del recocido simulado, una de ellos es aceptación por umbral el cual plantea que la aceptación de un movimiento se hace únicamente cuando se rebase un umbral. El objetivo del presente documento es describir una implementación propuesta a una variante del problema del agente viajero (también conocido como TSP por sus siglas en inglés) utilizando el algoritmo de recocido simulado con aceptación por umbrales, se describirá el modelo propuesto, su implementación y los resultados obtenidos.

2. Terminología

Sea G una gráfica ponderada:

- $V(G)$ vertices de G
- $E(G)$ aristas de G
- Si $e \in E(G)$, entonces $w(e)$ es el peso de la arista e
- Si $v, u \in V(G)$ y $(v, u) \in E(G)$ entonces $w(v, u)$ es el peso de la arista (v, u)

3. Descripción problema

3.1. Descripción general

El TSP es un problema combinatorio, NP-duro donde a partir de un conjunto de ciudades y la distancia que existe entre ellas (en caso de que las ciudades estén conectadas) el objetivo consiste en encontrar la ruta más óptima que permite visitar cada ciudad únicamente una vez y regresar a la ciudad de origen.

La variante del problema TSP que se va a tratar, pretende encontrar una ruta optima en un conjunto de ciudades (escogidas de forma aleatoria por el sistema) donde el agente viajero no retorna a la ciudad de origen.

3.2. Descripción implementación

Sea C una gráfica conexa ponderada donde $V(C)$ representan ciudades distribuidas a nivel mundial y $E(C)$ determinan cómo están conectadas dichas ciudades, el peso de cada arista determina la distancia que existe entre ellas; dicha información está almacenadas en un base de datos SQL y se trabajará con un total de $n = 277$ ciudades.
Las simulaciones contendrán un conjunto de ciudades $c \subset V(C)$, donde las ciudades de c se escogerán de forma aleatoria; c estará compuesta de k ciudades, valor que será determinado por el usuario.
Se espera que los resultados de una simulación sean reproducibles, de modo que se trabaja con un valor s que es una semilla utilizada por el generador de números aleatorios (**RNG** por sus siglas en inglés), el cual es el encargado de escoger aleatoriamente las ciudades que componen a c, el valor de la semilla s será determinado por el usuario.
A partir de s el RNG escogerá las k ciudades de c; la primera ciudad c_0 se escogerá de forma aleatoria a partir de las ciudades en $V(C)$, por cada nueva ciudad c_i que se añade a c se busca en sus ciudades adyacentes la ciudad más lejana c_x, si c_x no se encuentra en c entonces c_x se añade a c, si c_x pertenece a c, se escoge una ciudad aleatoria en C que no esté en c; este proceso se lleva a cabo hasta alcanzar el número k de ciudades requeridas; de esta forma se escogen las ciudades de c, apreciar que el orden en que fueron escogidas las ciudades es relevante, este es la base de la solución inicial s_0 compuesta por las ciudades de c, de la cual parte el algoritmo de recocido simulado con aceptación por umbrales; nótese que si nunca se da el caso de que c_x pertenezca a c entonces se garantiza que en s_0 todas sus ciudades tienen al menos un camino entre sí.
Todas las soluciones generadas s_i, contendrá a las ciudades de c y estas estarán organizadas de forma que se puede determinar el orden en que deben ser recorridas, es decir:

$$s_i = \{c_1, c_2, \ldots..c_i, c_j, \ldots.c_k\}$$

Sean $i = 1, 2, 3, \ldots, k-1$ y $j = 2, 3, 4, \ldots k$ donde $i < j$, i y j representa la posición de una ciudad en la solución.
Supongase que $j = i+1$, lo cual significa que se debe recorrer primero la ciudad c_i y seguido la ciudad c_j
Una solución s_i para el problema que se desea abordar también será conocida como *Tour*.
Cada solución s_i tiene una función de costo f asociada, la cual permite determinar su calidad, si $f(s_i) \geq 1$, significa que al menos entre un par de ciudades de s_i no hay un camino que las una. Por el contrario si $f(si) < 1$, entonces entre todas sus ciudades existe al menos un camino; el valor de $f(s_i)$ el cual se encuentra normalizado determina qué tan largo es el recorrido total entre las

ciudades de c según el orden en que se encuentran en s_i, es decir que entre más pequeño sea el valor de $f(s_i)$ menor es la distancia que se recorre y viceversa.
La función de costo f se definió como:

$$f(s_i) = \frac{\sum_{i=1}^{k-1} d(c_i, c_{i+1})}{3 * M * k/2}$$

Donde M es el diámetro de la tierra y $d(c_i, c_{i+1})$ es la distancia entre dos ciudades c_i y c_{i+1} de s_i, definida como:

$$d(c_i, c_{i+1}) = \begin{cases} w(c_i, c_{i+1}) & si \quad (c_i, c_{i+1}) \in E(C) \\ \\ 3 * M * k/2 & si \quad (c_i, c_{i+1}) \notin E(C) \end{cases}$$

Una solución s_i es capaz de generar una solución vecina a sí misma con ayuda del **RNG**, un vecino s_j de s_i se generará intercambiando la posición de dos ciudades en s_i ; el **RNG** elige dos ciudades c_i y c_j aleatoriamente en s_i donde c_i y c_j no son necesariamente consecutivas en s_i y ubica a la ciudad c_i en la posición j del tour y a la ciudad c_j en la posición i del tour.
Se dirá que el sistema acepta un vecino s_j de s_i si:

$$f(s_j) \leq f(s_i) + T$$

Donde T es la temperatura actual del sistema

El recocido simulado con aceptación por umbrales que se implementó cuenta con un conjunto de parámetros libres, valores que pertenecen a los reales positivos y determinan la calidad del sistema, en términos de tiempo de ejecución y cálidad de las soluciones encontrardas, dichos valores se determinan a través de la experimentación corriendo el sistema varias veces con diferentes valores; los parámetros libres son valores constantes para cada problema, es decir que cada vez que se desea aplicar el recocido simulado en un problema diferente estas deben ser ajustadas nuevamente.
La solución inicial s_0 creada inicialmente por el **RNG** se utiliza como base para iniciar la búsqueda; primero que todo se desea calcular cual sería la temperatura inicial apropiada para resolver la instancia del problema que contiene a las k ciudades de c, la temperatura inicial debe ser un valor apropiado que evite que la mejor solución encontrada sea un mínimo local o que el sistema tarde demasiado en terminar; de forma general lo que se hace es que a partir de una temperatura inicial denotada como $TEMPINIT$ que es un parámetro libre, se generan vecinos de s_0 y se determina la cantidad de soluciones aceptadas con dicho valor de la temperatura, a partir de $TEMPINIT$ se toma un valor más alto o más bajo como temperatura hasta encontrar un porcentaje mínimo de soluciones aceptadas por dicha temperatura
Una vez se ha determinado la temperatura inicial apropiada para la instancia del problema, se reinicia nuevamente el **RNG** con el valor de s, puesto que

durante el cálculo de la temperatura inicial se generaron vecinos a partir de s_0, lo cual implica que se usó el **RNG** y no reiniciarlo es exponerse a obtener resultados anormales. Finalmente inicia la simulación.

4. Descripción técnica implementación

La implementación del problema cuenta con las siguientes características:

- Lenguaje de desarrollo: C#
- Plataforma de desarrollo: Visual studio express 2013
- Biblioteca graficar mapas: GMAP.NET

5. Diseño

El diseño del sistema en general, se divide en dos partes, la primera que se encarga de modelar los elementos relacionados con el recocido simulado con aceptación por umbrales y el segundo que modela los elementos correspondientes al problema del agente viajero.

5.1. Recocido simulado con aceptación por umbrales

El modelo pretende sea capaz de generar solución sin importar cual sea el problema planteado, para lo cual se identificaron los siguientes elementos, con sus correspondientes interacciones:

1. **IManager** Administrador de problemas, encargado de contener toda la información necesaria para dar solución a un problema, dado que es un conocedor del problema será el encargado de generar la solución inicial aleatoria s_0 para dar inicio a la simulación.
2. **ISolution** Una solución s_i del problema, como se mencionó anteriormente cada solución s_i es capaz de generar una solución vecina s_j , con ayuda del **RNG**, el cual está representado en el sistema por un objeto Random, además cada s_i calcula cuál es su función de costo.

Los elementos descritos anteriormente se proponen como interfaces que definen el conjunto mínimo de operaciones necesarias de estos, así su implementación se puede realizar conforme lo demande el problema que se desea resolver.

3. **Batch** Lote de soluciones generadas a partir de una temperatura y una solución si, cada lote intenta generar una cantidad $BATCH_SIZE$ (parámetro libre) de soluciones s_i, sin embargo debe hacerlo con en un número limitado $MAXITERATIONSIZE$ (parámetro libre) de iteraciones durante la búsqueda de soluciones por aceptar, por ende un lote debe conocer si completó la cantidad requerida de soluciones.

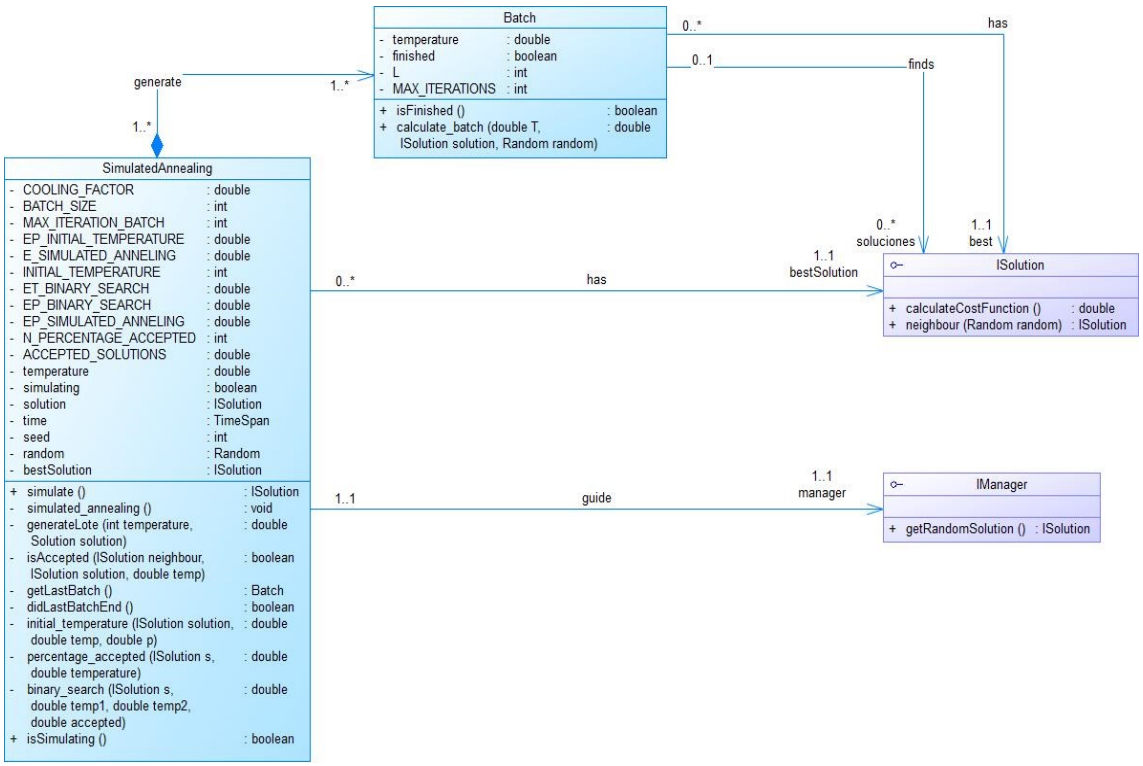

Figura 1: Modelo recocido simulado aceptación por umbrales

3. **SimulatedAnneling** Componente realiza la simulación del recocido simulado con aceptación por umbrales a partir del problema que se encuentra inmerso en el manager que lo guía, tiene como función calcular la temperatura inicial adecuada para la instancia del problema, realizar la simulación a partir de la solución inicial s_0 entregada por el manager, conocer los lotes de soluciones generados durante una simulación e identificar cual fue la mejor solución generada durante la simulación.

5.2. Problema agente viajero

El problema del agente viajero, involucra elementos específicos a la naturaleza del problema que resuelve, se identificaron los siguientes elementos:

1. **City** Representa una ciudad pertenece $V(C)$, una ciudad está compuesta por su información geográfica, nombre, nombre del país al que pertenece, la cantidad de personas que la habitan, un número identificador único y un conjunto de ciudades A que son subconjunto de $V(C)$ con las cuales tiene un camino y la distancia hacia cada una de estas.

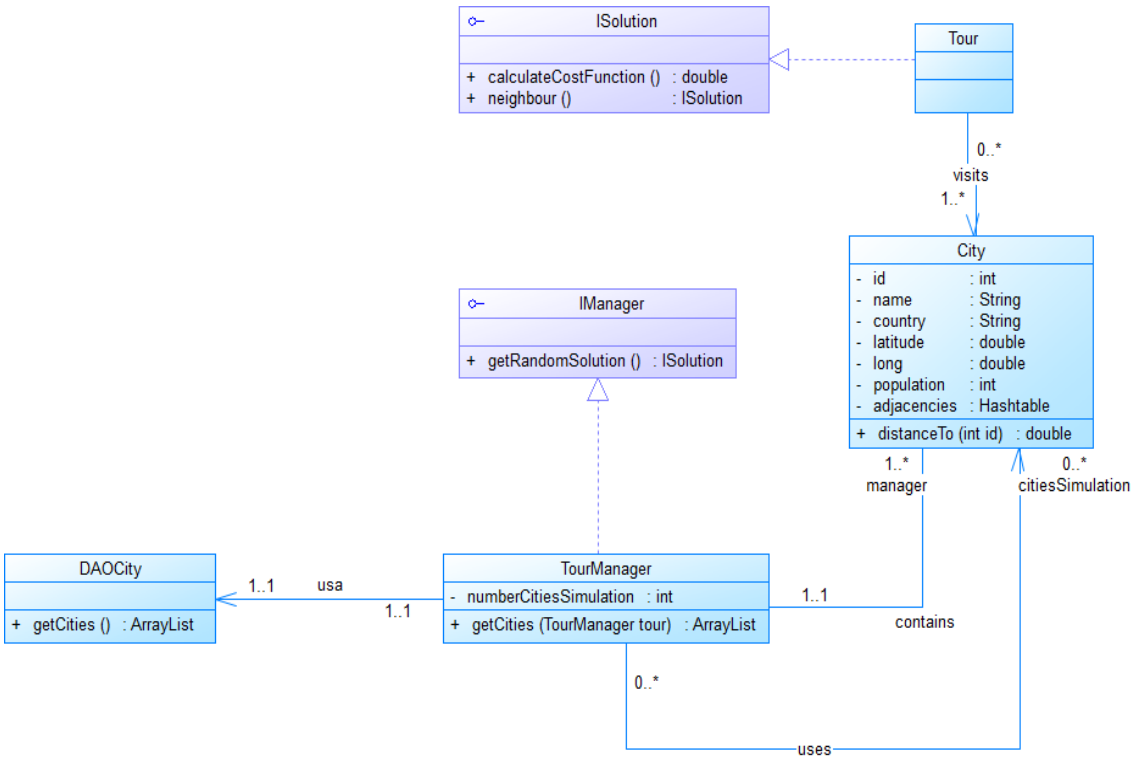

Figura 2: Modelo problema agente viajero

2. **Tour** Solución s_i del problema, contiene el conjunto de ciudades que debe visitar el agente viajero, es la implementación de **ISolution**, en la cual especifica cómo se debe generar un vecino y realizar el cálculo de su función de costo.

3. **TourManager** Administrador del problema, contiene el conjunto de ciudades $V(C)$ y conoce el número k de ciudades requeridas para la simulación; es la implementación de **IManager**, así define el proceso que se debe llevar a cabo para generar la solución inicial s_0.

4. **DAOCity** Componente realiza la lectura de las ciudades $V(C)$ a partir de una base de datos SQL

6. Ejecución implementación

El sistema implementado tiene dos modos de ejecución, a continuación se pretende explicar cómo es el funcionamiento de cada uno de estos.

6.1. Interfaz gráfica

El sistema inicia mostrando un mapa del mundo y la gráfica C, la cual permite ver cual es el conjunto de ciudades que la conforma, su distribución a nivel mundial, información relacionada a cada ciudad y cómo estan conectadas unas con otras.

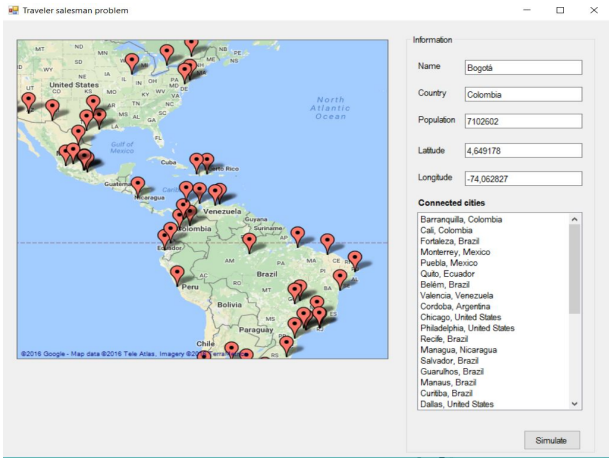

Figura 3: Inicio aplicación

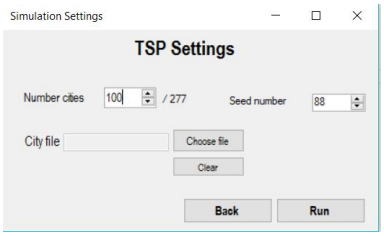

Figura 4: Configuración simulación

A continuación al seleccionar *Simulate*, dispone un formulario en el cual el usuario puede configurar una simulación, se debe determinar la cantidad k de ciudades necesarias para la simulación en el campo *Number cities*, indicar el valor de la semilla s con la que se desea trabajar en el **RNG** en el campo *Seed number*, también si el usuario lo desea puede cargar un archivo de texto con el formato:

$$id - city_1, id - city_2, ..., id - city_m$$

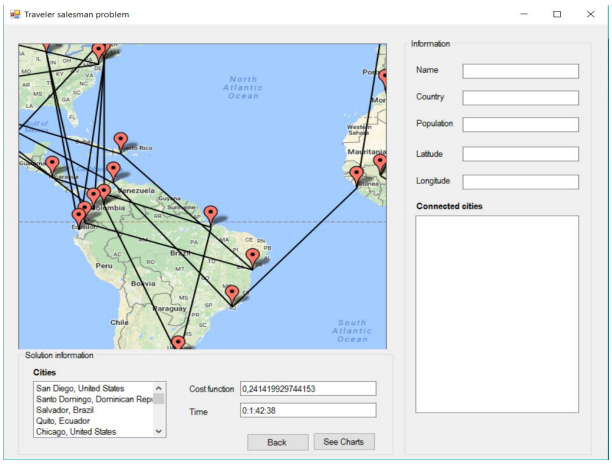

Figura 5: Resultado simulación

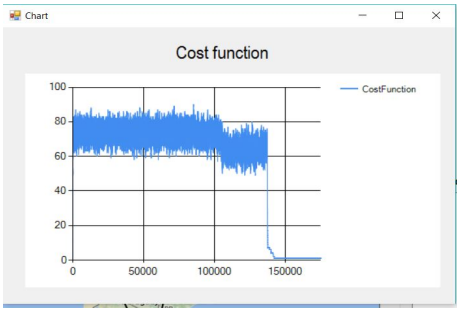

Figura 6: Análisis

Donde $m \leq k$ y $id-city_i$, es el identificador de una ciudad $\in V(C)$, $i = 1, 2, ...m$
Notar que si $m < k$ el sistema escoge las $k - m$ ciudades faltantes como se describió anteriormente, en caso que $m = k$ entonces el conjunto de ciudades c será el determinado por el usuario.
Seleccionar *Run* , si un archivo fue seleccionado se realiza su respectiva lectura, y seguido el sistema inicia la simulación, esta puede tarde varios minutos de acuerdo a la cantidad k de ciudades seleccionadas y la complejidad de la solución inicial; una vez se concluye la simulación, se presenta nuevamente el mapa del mundo con el conjunto de ciudades c dispuesto en él, su distribución y la mejor ruta encontrada para recorrerlas, se podrá ver el tiempo total de la simulación y cual es la función de costo asociada a la solución presentada:
La opcion *See chart* permite observar cómo fue la función de costo de todas las soluciones aceptadas a través del tiempo, en el eje x se ubica el número de la solución, es decir estan ubicadas conforme fueron generadas $1, 2, 3, ...$ y en el eje y se ubica el valor de costo de dicha solución generada en el $i-$esimo puesto.

Aquí se puede observar como al inicio de la simulación se aceptan soluciones que tiene un valor de costo excesivamente grande , con esto se evidencia que se trata de escapar de mínimos locales; conforme se acerca la finalización de la simulación se aceptan soluciones con funciones de costo cada vez más pequeñas

6.2. Consola de comandos

El programa podrá ejecutarse también desde la consola de comandos, para ello existirán 2 diferentes forma de configurar la simulación

- *csc.exe SimulatedAnneling cities seed* El sistema inicia la ejecución tomando a *cities* como las k ciudades necesarias para la simulación y a *seed* como el valor de la semilla s que se desea utilizar
- *csc.exe SimulatedAnneling cities seed* $id - city_1, id - city_2, ..., id - city_m$ El presente modo de ejecución pide que se ingrese de forma directa los identificadores de las ciudades en la linea de comando, en el mismo formato que tendrían si los valores estuvieran en un archivo de texto