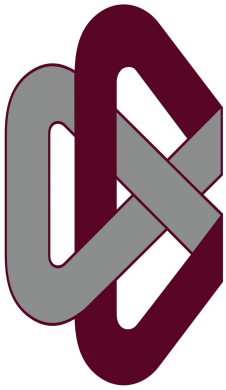
MAESTRÍA EN CIENCIAS CON ESPECIALIDAD EN COMPUTACIÓN Y MATEMÁTICAS INDUSTRIALES



CIMAT

PROYECTO

ANT COLONY

JOEL CHACÓN CASTILLO

Notas

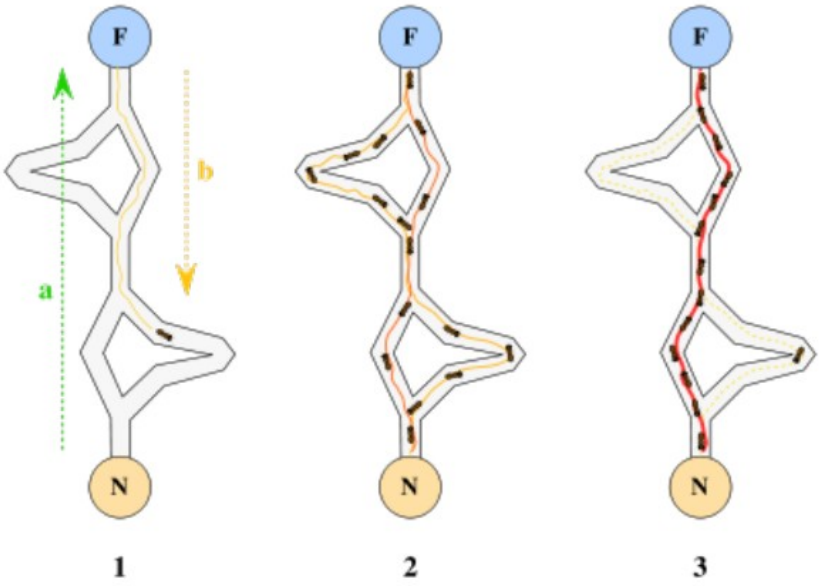
Se anexa la implementación en serial (paralelizado con memoria compartida OpenMP) y en paralelo (dos implementaciones) en sus correspondientes carpetas, cada programa cuenta con su respectivo archivo Makefile, los proceso asignados a los programas en paralelo deben ser mayor a uno dado que se utiliza el proceso cero como el nodo maestro, para cargar una instancia debe asignar el número de ciudades en el macro DIMENSION y cambiar el nombre de la instancia se encuentran fichero “main.c”.

Introducción

Los algoritmos basado en el comportamiento de hormigas fueron introducidos por Dorigo el cual formaliza una nueva meta-heurística llamada como “Colonización de Hormigas”, en este documento se expone brevemente la teoría sobre las variantes más conocidas del algoritmo “Ant Colony”, posteriormente se explica detalladamente una implementación en paralelo sobre este algoritmo, por último se presentan los resultados además de análisis de paralelizar el algoritmo.

Teoría

Este algoritmo está basado en el comportamiento de las hormigas, donde exploran desde el hormiguero en búsca de comida, pero dado que las hormigas son ciegas dejan feromonas en las rutas descubiertas, dodne cada hormiga se mueve de forma aleatoria, un mayor número de feromonas en el camino incrementan la probabilidad de utilizar esa ruta por las demás hormigas.



El algoritmo “Ants Colony” es una heurística la cual tiene como objetivo resolver problemas de optimización ya sea minimizar o maximizar, los problemas más conocidos en los cuales se utiliza este algoritmo son:

- **Travelling Salesman Problem (TSP)**.
- Quadrict Assignement Problem (QAP).
- Vehicle Routing Problem (VRP).
- Graph Coloring Problem (GCP).
- Sequential Ordering Problem (SOP).
- Job Scheduling Problem (JSP).
- Routing In Telecommunication Network.

El problema que se aborda es el TSP (Problema del Agente Viajero) el cual consiste en visitar todas las ciudades propuestas por medio del cálculo de la ruta que realice el menor recorrido.

El algoritmo “Ant Colony” se basa en al ajuste adaptativo de la feromonas que corresponden a las rutas de cada nodo (búsqueda tabu), la elección de cada nodo es guiado en un enfoque de probabilidad. Este

proceso se realiza basado en una lista Tabu, la cual almacena las feromonas correspondientes a cada ruta, se hace mención que la lista Tabu conforme pasa el tiempo (iteraciones) tiende a decrementarse el valor de las feromonas o descomponerse.

Las hormigas son dirigidas por medio de una regla de probabilidad escogiendo la trayectoria de su camino conocido como tour.

Los aspectos de mayor importancia en el algoritmo son dos:

- Descomposición de feromonas.
- Decisión probabilística y construcción de soluciones.

Descomposición o actualización de feromonas

La cantidad de feromonas es actualizado de acuerdo a las ecuación:

$$\tau_{ij}(t+n)=\rho\,\tau_{ij}+\sum_{k}\Delta\tau_{ij}^{k}$$

$\Delta\tau_{ij}^{k}=Q/L_k$ si la hormiga k utiliza el arista (i,j) en caso contrario es cero.

Donde

- $\Delta\tau_{ij}^{k}$ es la cantidad de feromona depositada en el arista (i,j) por la $k-esima$ hormiga en el intérvalo de tiempo $(t,t+n)$.
- Q es una constante.
- L_k Es la longitud o costo de la ruta construida por la $k-esima$ hormiga.
- ρ Esn la razón de evaporación de deromonas y debe ser menor que 1, de otra forma la feromona se acumulará de forma no limitada (se recomienda 0.5).

Decisión probabilística y contrucción de soluciones

Una hormifa se moverá desde el nodo i hasta el nodo j con probabilidad $P_{ij}^{k}(t)$, donde la probabilidad de agregar un arista (i,j) (donde $j\in\{N-tabu_k\}$ en la ruta

$$P_{ij}^{k}(t)=\frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l}[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}} \quad \text{si} \quad j\in\{N-tabu_k\} \quad \text{de otra forma} \quad P_{ij}^{k}(t)=0$$

Definición:

- $l\in\{N-tabu_k\}$
- α,β definen la importancia relevante de la feromona y la visibilidad.
- τ_{ij} es la cantidad de feromonas en el arista i, j .
- η_{ij} es la atracción a el arista i, j .

Entonces la probabilidad se encuentra muy relacionado entre la visibilidad y la intensidad de la feromona, donde la visibilidad se refiere a la preferencia de las ciudades más cercanas, y la intensidad indica un umbral de que aristas o caminos se utilizan más frecuentemente.

Variantes del algoritmo

En la literatura se han propuesto muchas variantes del algoritmo ACO, el algoritmo original es conocido como “Ant System”, las versiones más conocidas:

Ant System.

- Este fue el primer algortimo propuesto.

MAX-MIN

Este algoritmo tiene mejores sobre el algoritmo original ACO, las principales mejoras que posee son:

- Sólo la mejor hormiga agrega feromonas a las trayectorias.
- Las feromonas tienen valores mínimos y máximos.
- Las feromonas son inicializadas a su máximo valor.
- Los valores correspondientes a las feromonas son reinicializados cada cierto tiempo o número de iteraciones con el objetivo de manejar más diversidad.

Ant Colony System (ACS)

Las principales diferencias entre los dos algoritmos mencionados son tres principiales aspectos:

- Existen reglas de transición las cuales proveen una forma de proporcionar un balance entre exporación de nuevos aristas y explotación de la información conocida del problema.
- Existen reglas de actualización global que son aplicados a los aristas los cuales pertenecen a la mejor trayectoria obtenida por la hormigas.
- Mientras las hormigas construyen una solución una regla de la feromona local es aplicada.

Algoritmo en serial

El algoritmo programado en serial corresponde al Ant System, donde τ , η son matrices de dimensión [Numero Ciudades x Numero Ciudades], la lista Tabu está conformada por una matriz con un número de filas correspondiente al número de hormigas y un número de columnas correspondiente al número de ciudades.

En el pseudocódigo que se presenta en la línea 1 se inicializan los parámetros, posteriormente se fectúa la lectura de las coordenadas que corresponden a la ciudad, en este caso se implementa una representación donde se construye una matriz de distancia, cada entrada corresponde a una distancia entre la ciudad i a la ciudad j . En esta representación se obtene una matriz simétrica dado que el grafo es no dirigido, sin embargo existen representaciones donde la matriz es no simétrica dado que el grafo de distancias es dirigido, es decir no existe la misma distancia en el recorrido de la ciudad A a B que de B a A.

En la línea 4 se inicia el ciclo, donde el criterio de paro está establecido dado un número maximo de iteraciones, la otra condición se explicadá más adelante.

De la línea 5 a la 16 cada hormiga genera un recorrido completo de todas las ciudades, donde cada hormiga genera un vector de probabilidades en base a los valores de las feromonas (línea 8), en la sigueinte línea dadas las probabilidades se escoge una ciudad de las que no se encuentran en la lista Tabu (con la probabilidad P_{ij} calculada), entonces se agrega a la lista Tabu el índice (indice en relación a la matriz de distancias) que corresponde a la ciudad escogida.

De la línea 10 a la 13 se obtiene el costo de la trayectoria obtenida para la hormiga i , donde posteriormente de las lineas 14 a la 16, se efectúa elitismo, es decir se verifica que se tenga el mejor

individuo o trayectoria obtenida hasta el momento.

En la línea 17 y 18 se realiza la actualización de la matriz de feromonas, normalmente la variante de todos los algoritmo cambien en la forma en que se actualizan las feromonas. En la línea 19 se obtiene el escalar $N\tau$, la cual es la sumatoria de los componentes normaliación de la matriz de feromonas τ, este cálculo se efectúa como criterio de convergencia el cual se especifíca en la línea 20, el criterio de convergencia debe cumplirse cuatro veces, esto es para evitar convergencia prematura, lo cual ofrece una mayor diversiadad en el espacio de búsqueda.

Algorithm 1 Ant Colony

Require: Initial solution $Cont = 0$, $\alpha = 1$, $\beta = 5$, $\rho = 0.5$, $MaxIteraciones$

1: Inicializar τ , $\eta = \frac{1}{Distancia_{i,j}}$,
2: Realizar la lectura de la coordenadas o generar coordenadas de las ciudades
3: Construir la matriz de adyacencia
4: **while** $MaxIteraciones > 0$ AND $Cont < 4$ **do**
5: **for** i = 1 to Numero de hormigas **do**
6: **for** j = 1 to Numero de ciudades **do**
7: $p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum_l [\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}$ if $j \in \{N - tabu_k\}$
8: IndexCity = Sample(p_{ij}^k)
9: push IndexCiy to Tabu
10: **for** k = 0 to Numero de Ciudades - 1 **do**
11: IndexI = $Tabu_{i,k}$
12: IndexJ = $Tabu_{i,k+1}$
13: $L_i = Mapa_{IndexI,IndexJ}$
14: **if** fbest $> L_i$ **then**
15: fbest = L_i
16: $Xbest = Tabu_i$
17: $\Delta\tau = \sum_k \frac{Q}{L_k}$ $\forall i, j \in$ Numero de Ciudades
18: $\tau_{ij}(t+n) = \rho\tau_{ij}(t) + \Delta\tau$
19: $N\tau = \sum \frac{\tau}{||\tau||}$
20: **if** $\left(\frac{(NumeroCiudades)^2}{4} - N\tau\right) < ValorMinimo$ **then**
21: Contador++;
22: $MaxIteraciones = MaxIteraciones - 1$
23: Imprimir la solución

- d_{ij} es la distancia euclidianta desde la ciudad i a la ciudad j.
- m es el número de hormigas.
- $\tau_{ij}(t)$ es la intensidad de feromonas que tiene el arista (i, j) en el tiempo t .
- η_{ij} es la visibilidad expresada por $\frac{1}{d_{ij}}$.
- $(1 - \rho)$ es el factor de evaporación de las feromonas.
- $tabu_k$ es el vector dinámico de las ciudades que ya han sido visitadas por la $k - esima$ hormiga.
- $\Delta\tau_{ij}^{k}$ es la cantidad de feromonas depositadas en el arista (i, j) por la hormiga $k - esima$ en un intérvalo de tiempo $(t, t + \eta)$.
- Q es una constante.
- L_k es la longitud de la ruta construida por la $k - esima$ hormiga.
- ρ debe ser menor que 1 (recomendado 0.5)

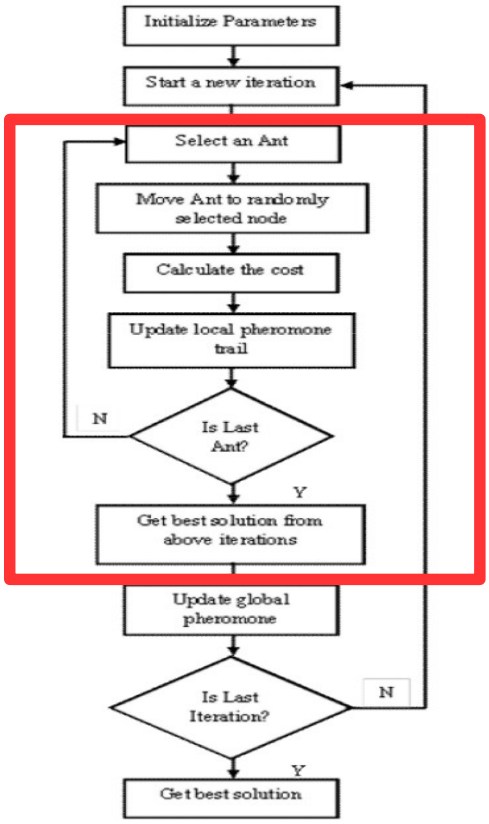
Selección de parámetros

En la referencia [i] se indica que la configuración de parámetros que ofrecen un mejor resultado están en función de la instancia o de las coordenadas de las ciudades que se resolverá, los parámetros lo cuales

pueden variar son α . β y Q . Se explica que el parámetro β regula la importancia de la visivilidad η_{ij} donde un valor elevado de β proporciona una elevada importancia a la heurística de distancia, un valor elevado de β cambia el algoritmo a algo más parecido a una búsqueda greedy. También se concluye que para tener una mejor estimación de los parámetros lo mas adecuado se realiza un algorito genético modificado par a la colonización de hormigas conocido como “Genetically Modified Ant Colony system (GMACS)”, el cual consiste en inicializar cada hormiga con una combinación de parámetros, donde los parámetros son escogidos de un rango establecido.

En el algoritmo presenta se utilizan los parámetros propuestos por el Dorian con $\alpha=1$, $\beta=5$, $\rho=0.5$.

Diagrama de flujo del algoritmo “Ant Colony”



Implementación en paralelo

Existen muchas variantes para efectuar la paralelización del algoritmo propuesto, en la referencia[ii] se

mencionan aspectos importantes en la paralelización del algoritmo, se estalece que la comunicación de la matriz entera de feromonas puede afectar a la calidad de la solución en función de un buen o un mal comportamiento del tiempo de ejecución , mientras que intercambiar los elites produce mejores resultados en la calidad de la solución, la estrategia de comunicación que se sugiere es intercambiar a la mejor solución y cada n iteraciones intercambiar las soluciones, esto proporciona una mejor diversidad del espacio de búsqueda y por lo tanto una solución de mayor calidad, la implementación efectuada es del algoritmo “MAX-MIN” Ant System donde se eliminan los reinicios ocacionales de las feromonas y utilizando sólo las actualizaciones de las mejores feromonas.

Las topologías en las cuales se implementó el algoritmo “Ant Colony” son:

- Distribución de evaluaciones, ene ste modelo el nodo maestro administra la población a procesar, en este caso se divide el número total de hormigas entre el número de procesos menos el proceso maestro el cual no realiza el mismo trabajo que los procesos esclavos.
- Parallel independent runs, en este modelo se realizan n copias y el algoritmo es ejecutado simultáneamente e independientemente, con distintas semillas, al final se obtiene la mejor solución de las n ejecuciones.

El primer algoritmo implementado se basa en distribuir la población de hormigas en la cantidad de nodos disponibles donde cada nodo contiene una subpoblación de hormigas y cada hormiga calcula una trayectoria, posteriormente se envía la mejor solución de la subpoblacion al nodo maestro, el cual obtiene la mejor trayectoria de todos los individuos, en el diagrama de flujo presentado se puede apreciar que la parte paralelizada se encuentra dentro del recuadro de color rojo, la desventaja de este método es que ante subpoblaciones grantes, puede existir un tiempo de retraso al momento de enviar toda la información, al nodo esclavo correspondiente. En la implementación efectuada se hace énfasis que es necesario tener más de un proceso (que es el proceso maestro), en su defecto la implementación no servirá.

La segunda implementación consiste en asignar una semilla distinta a cada proceso, en este caso se utiliza el generador de números pseudo aleatorios “Hybrid Taus”, el cual recibe como semilla una combinación de la función “time(0)” y del “Rank” del proceso, al igual que en la implementación anterior el programa debe tener disponible al menos dos procesadores, donde uno está destinado al

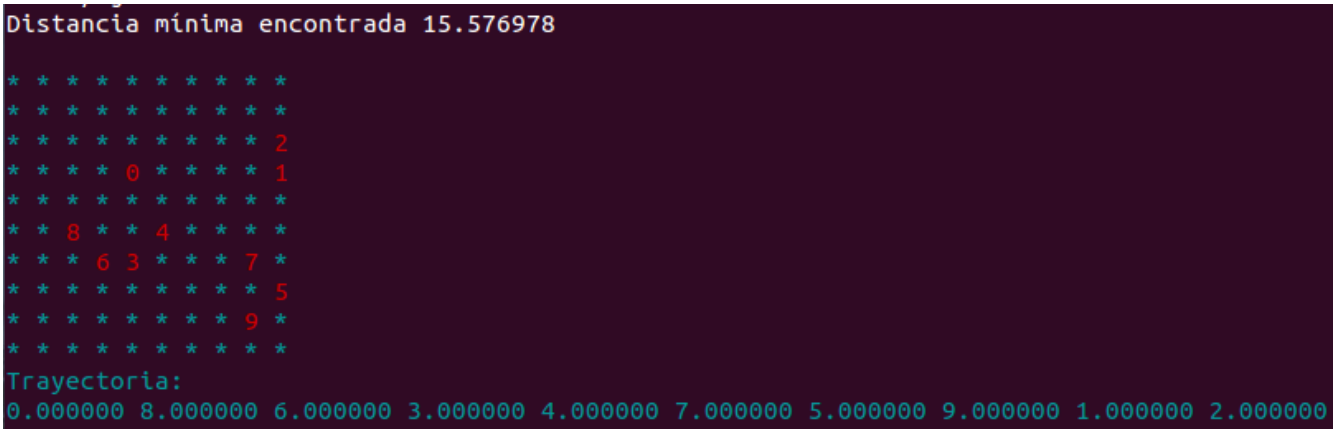
proceso maestro y los demás a los procesos esclavos.

Herramienta de MPI utilizadas para paralelizar

En esta implementación se utilizaron paquetes para enviar-recibir información, lo cual facilita notablemente la comunicación colectiva, en las primera implementaciones se utilizó la función MPI_Gather, pero posteriormente se verificó que para esta implementación es más factible enviar y recibir paquetes entre las comunicaciones de los procesos esclavos y el proceso maestro.

Impresión de las ciudades

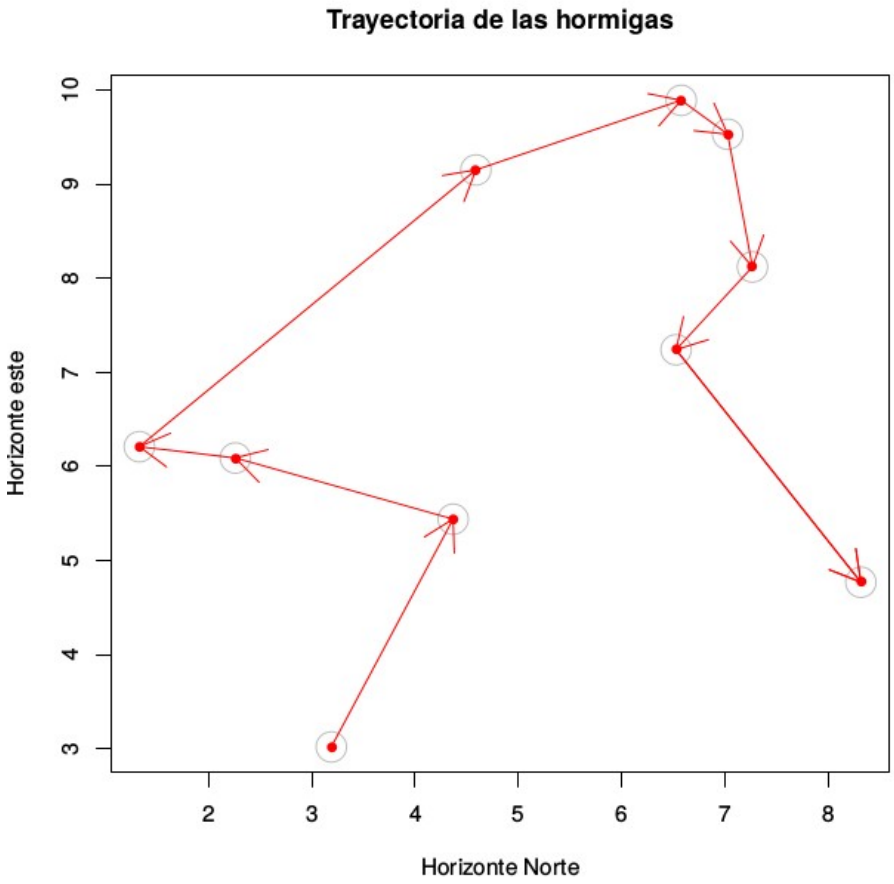
Se tienen dos mecanismos para imprimir las ciudades, la primer forma es por medio de terminal (tiene algunos problemas) donde las coordenadas de las ciudades deben ser enteros (no existe una validación en las coordenadas repetidas), en la impresión de pantalla se presenta la distancia mínima obtenida posteriormente la locación de las ciudades, cada ciudad es representado por un número de color rojo, al final se presenta la trayectoria calculada, por ejemplo en la imágen primero se visita la ciudad con identificador 0 luego la ciudad con identificador 8 entonces el recorrido es 0→ 8→ 6→ 3→ 4→ 7 → 5 → 9 → 1 → 2

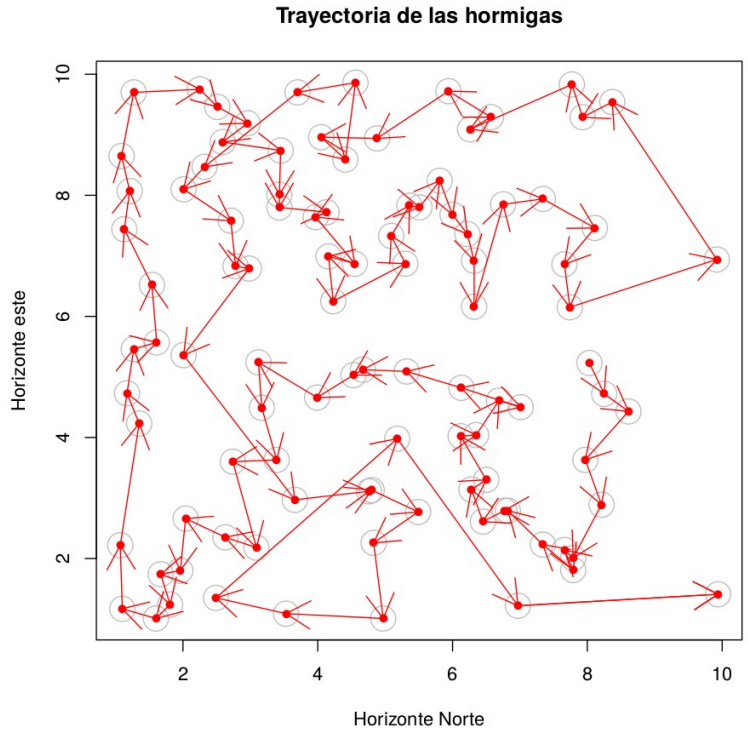


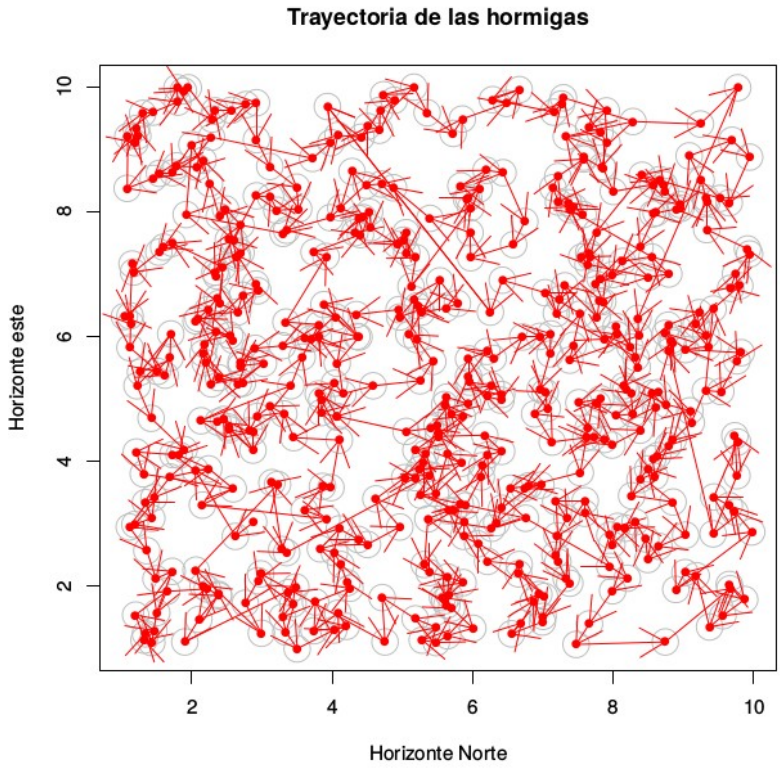
La segunda forma de presentar el resultado es por medio de la interacción con R, es decir que el programa construye el comando y porteriormente se lo envía a R, por último se imprime la solución en un archivo PDF.

El generador de coordenadas proporciona números de tipo double, en los resultados que se muestran a continuación el límite para generar numeros entre [0,10]. Las imágenes que se encuentra a continuación

son de 10, 80 y 300 ciudades generadas en forma aleatorio.




Trayectoria de las hormigas
Horizonte este
Horizonte Norte
2
4
6
8
10


Trayectoria de las hormigas
Horizonte este
Horizonte Norte
2
4
6
8
10

Resultados

Los resultados varían en las dos implementaciones, en la primera implementación se distribuyen los recorridos de las hormigas en cada proceso por lo que el tiempo de procesamiento disminuye pero no ofrece un resultado de alta calidad, en la segunda implementación cada proceso realiza una ejecución independiente, se obtiene una solución que se puede considerar de mejor calidad, dado que al proporcionar una semilla por proceso aumenta la diversidad en el espacio de solución, esto se puede diferenciar notablemente conforme se aumenta el número de ciudades.

Se utilizó como benchmark los problemas de la dirección:

http://www.math.uwaterloo.ca/tsp/world/countries.html

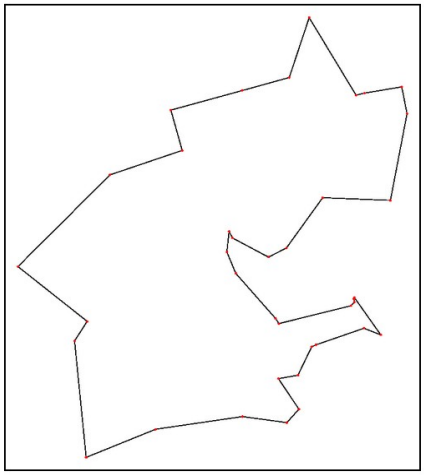
donde escogieron las instancias de la ciudad “Djibouti” y “Qatar”, lo cuales se comparan con los resultados obtenidos por el algoritmo descrito en este documento, en las imágenes de los resultados en la imagen de la izquierda indica el recorrido sugerido por el algoritmo y el de la derecha indica el recorrido optimo (del benchmark).

Djibouti

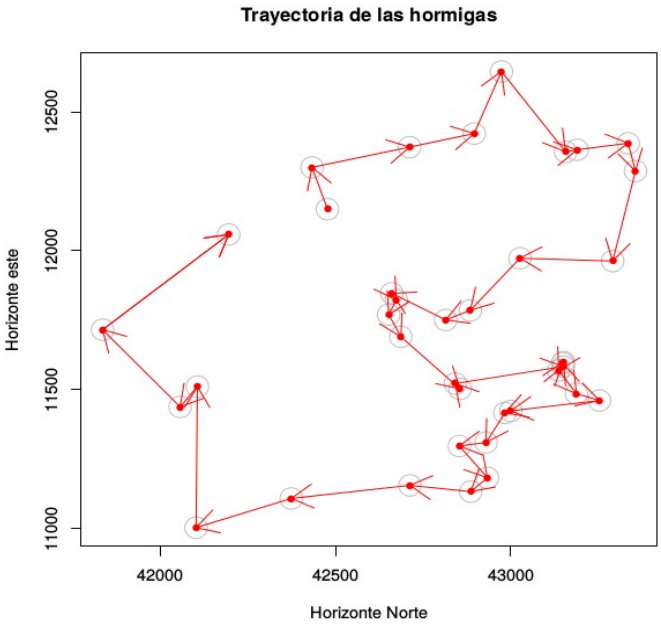
Esta instancia esta conformado de **38** ciudades y las distancias optimas son:

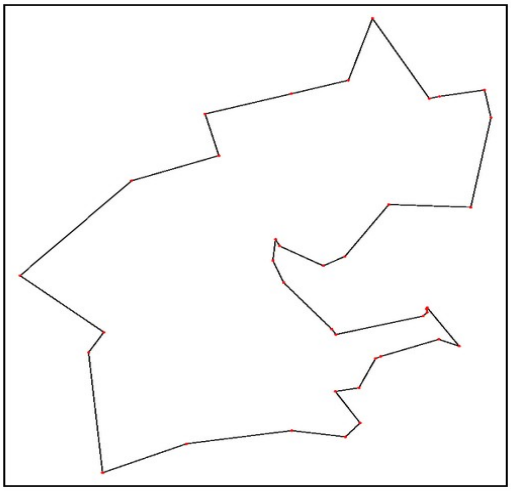
Método	Distancia óptima
Solución benchmark	6656
Paralelización de recorridos	6804 (4 procesos)
Peralelización de ejecuciones independientes	6659 (4 procesos)

Mapa con trayectoria optima del benchmark

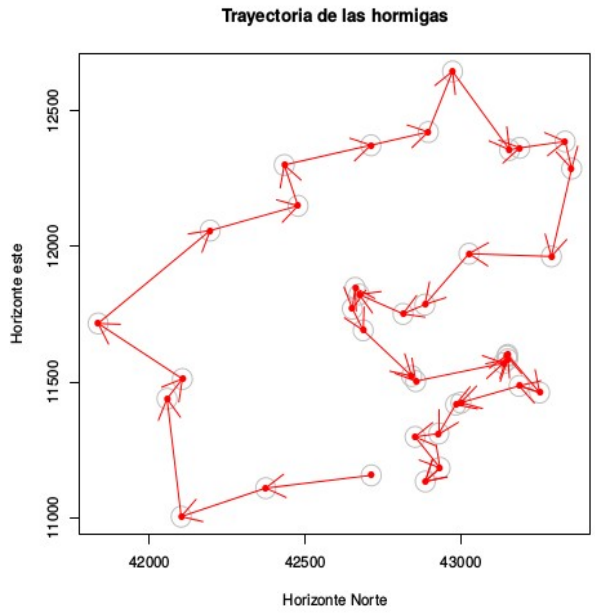


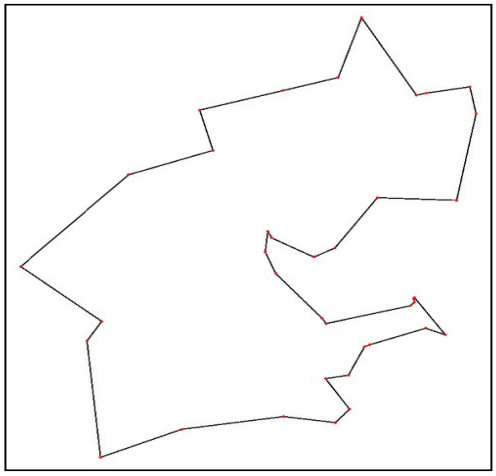
Mapa con método paralelizando recorridos





Mapa con ejecuciones independientes



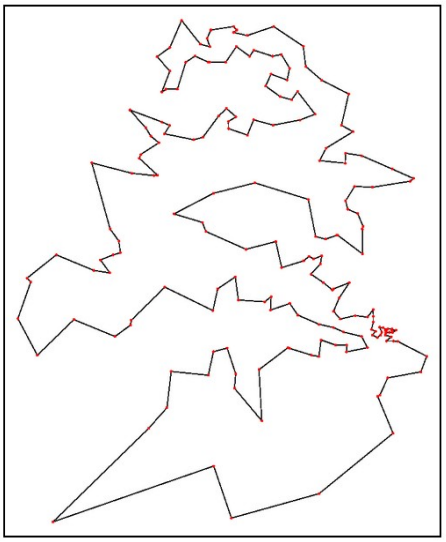


Qatar

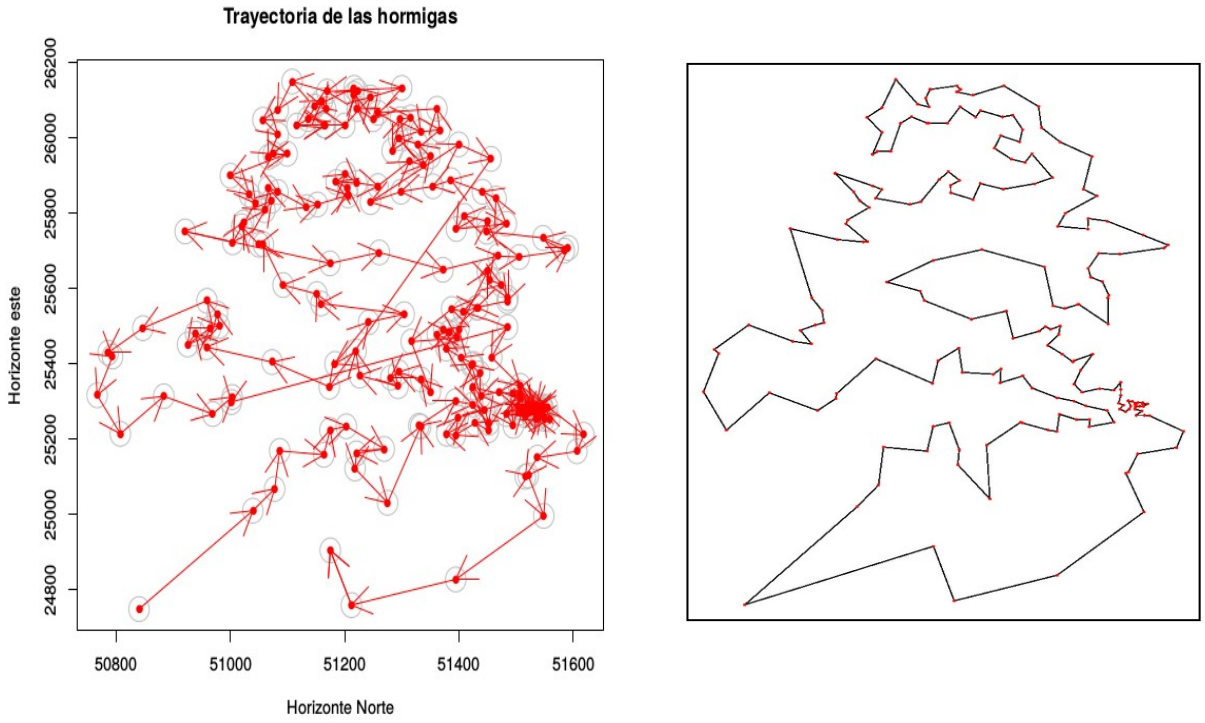
Esta instancia está conformada por **194** ciudades y las distancias optimas obtenidas son:

Método	Distancia óptima
Solución benchmark	9352
Paralelización de recorridos	12000 (4 procesos)
Peralelización de ejecuciones independientes	10394 (4 procesos)

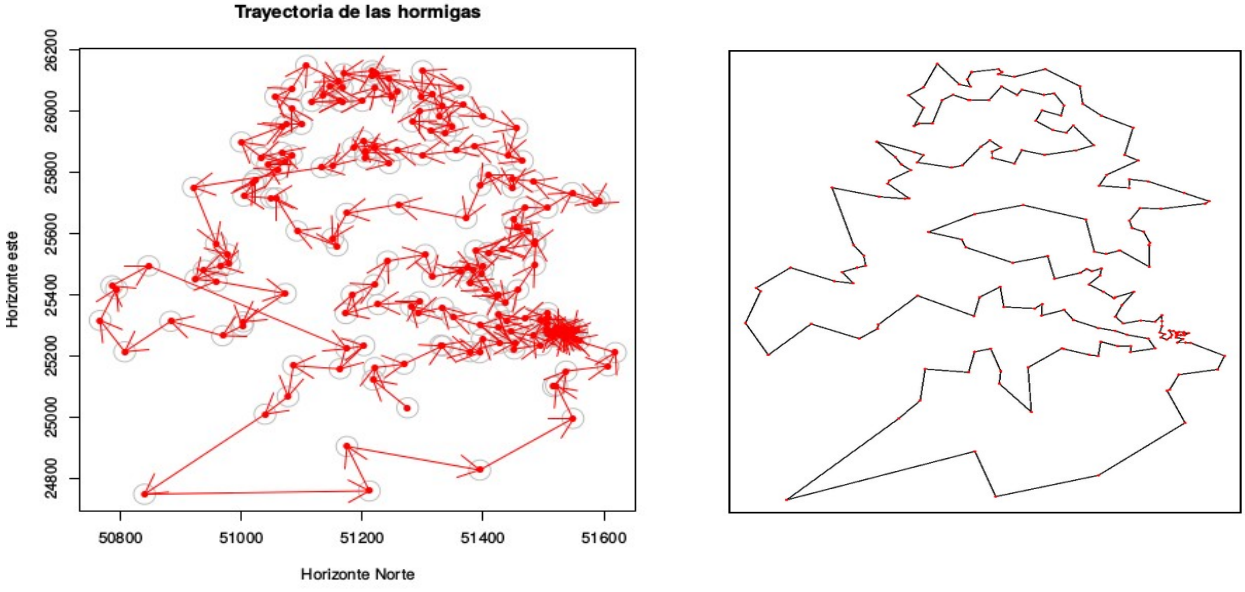
Mapa con trayectoria optima del benchmark



Mapa con método paralelizando recorridos

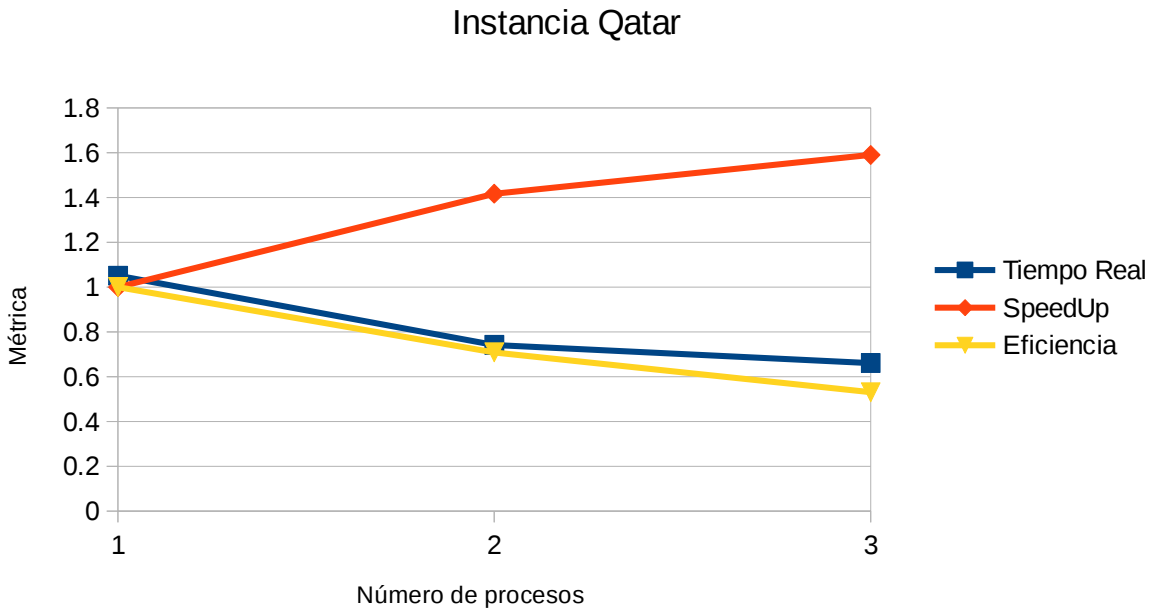


Mapa con ejecuciones independientes



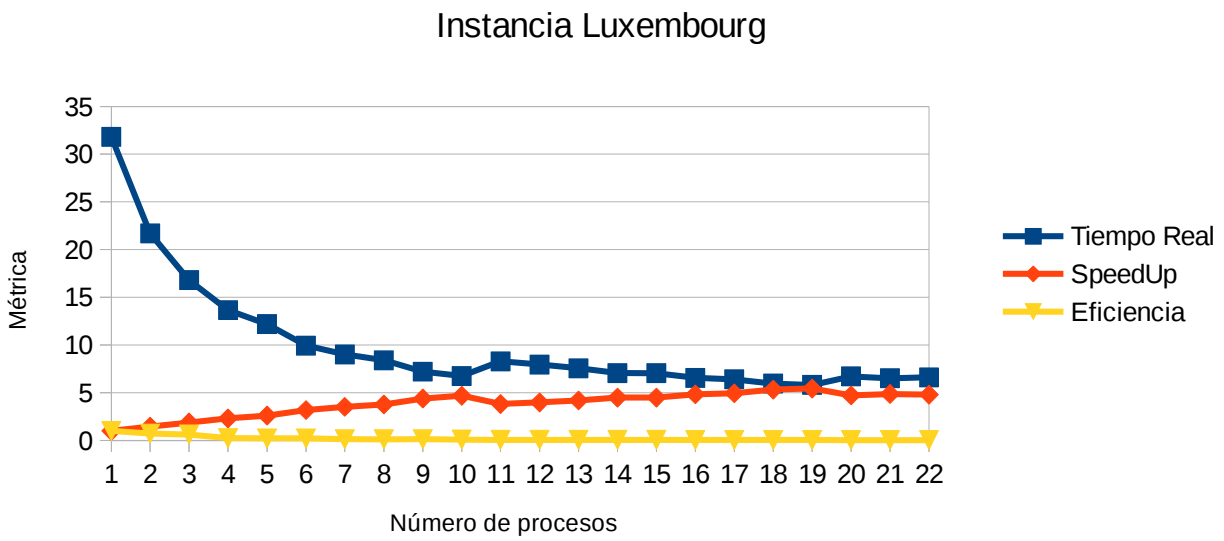
Analisis de la paralelización de recorridos Qatar

Numero Procesadores	Tiempo Real	SpeedUp	Eficiencia	Distancia
1	1.0511	1	1	12379.042418
2	0.742	1.4165768194	0.7082884097	12234.741086
3	0.661	1.5901664145	0.5300554715	12451.828534



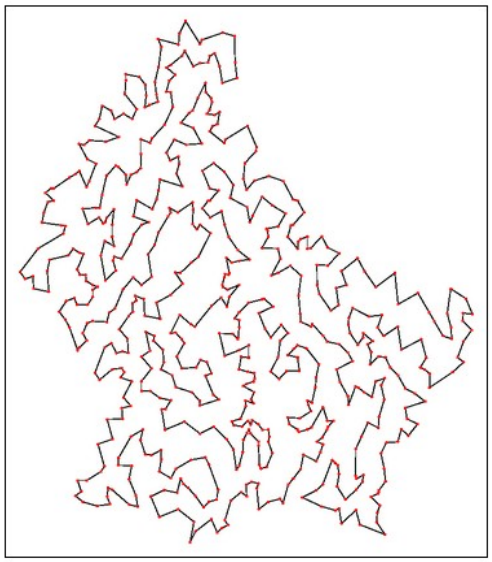
Ejecución en el cluster

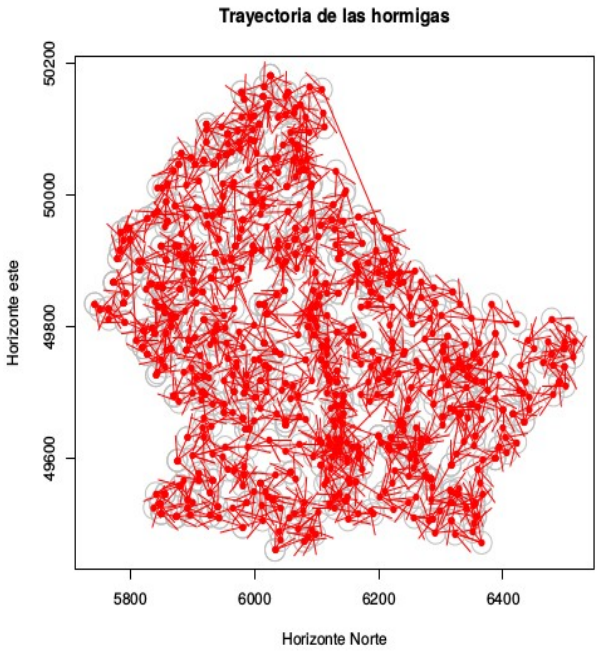
Se utilizó la instancia “Luxembourg” con un número de **980** ciudades, el recorrido óptimo que tiene registrado es es **11340**.



Se puede observar que el SpeedUp disminuye al momento de utilizar 11 procesadores, posiblemente esto se debe al “hyper threading”, se puede observar que las distancias óptimas en cada procesador ocsila entre 15000 y 16000.

Numero Procesadores	Tiempo Real	SpeedUp	Eficiencia	Distancia
1	31.802	1	1	15872.872553
2	21.709	1.4649223824	0.7324611912	15713.681738
3	16.808	1.8920752023	0.6306917341	16015.369677
4	13.671	2.3262380221	0.25	16093.164444
5	12.199	2.6069349947	0.2241331257	16140.070401
6	9.941	3.1990745398	0.2292022935	15969.522478
7	9.025	3.523767313	0.1428571429	15823.12578
8	8.414	3.7796529594	0.1340771333	15824.605087
9	7.221	4.4040991552	0.1388696549	16041.362985
10	6.764	4.701655825	0.1	15927.520099
11	8.305	3.8292594822	0.0740408297	16192.531647
12	7.966	3.9922169219	0.0707590593	16095.36587
13	7.569	4.2016118378	0.0769230769	16033.127111
14	7.067	4.5000707514	0.0765024561	16114.28796
15	7.062	4.5032568677	0.0714528462	15844.766219
16	6.569	4.8412239306	0.0625	15973.618784
17	6.421	4.9528110886	0.0601793747	16194.155432
18	5.965	5.3314333613	0.061180963	15854.799523
19	5.83	5.4548885077	0.0526315789	16119.881148
20	6.723	4.7303287223	0.0433586197	15882.145777
21	6.528	4.871629902	0.0425274276	15950.186379
22	6.626	4.7995774223	0.0454545455	16069.237655





Conclusión

Existen muchos algoritmos sobre todos heuristicos los cuales se pueden paralelizar en disntintos modelos, el tipo de paralelización implementar depende mucho de las necesidades del programador, por ejemplo si se desean realizar cáculos en poco tiempo y la solución puede tener un error relajado, se puede paralelizar las evaluaciones, en caso de que se busque una error estricto se puede implementar un modelo en islas ya sea intercambiando

elites o cierta información constantemente o ejecutar cada proceso con una distinta semilla, para que al final se tenga una mejor exploración y tener un solución más precisa.

i On Optimal Parameters for Ant Colony Optimization algorithms Dorian Gaertner and Keith Clark

ii Parallel Ant Colony Optimization for the Traveling Salesman Problem Max Manfrin, Mauro Birattari, Thomas St¨ utzle, and Marco Dorigo