Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI5652 - Diseño de Algoritmos II
Trimestre Abril - Julio 2012

INFORME III
Resolviendo el Capacited Vehicle Routing Problem (CVRP)

Integrante:
Bishma Stornelli 08-11094
Vicente Santacoloma 08-11044

Sartenejas, 12 de julio de 2012

INTRODUCCIÓN

1. Motivación del proyecto

El presente informe pretende realizar una descripción y análisis de la metaheurística **Ant Colony Optimization** para resolver el problema del **Capacited Vehicle Routing Problem (CVRP)** y además comparar los resultados obtenidos con aquellos obtenidos al usar la metaheurística **Tabu Search** cuyos resultados se presenta en el informe anterior.

Para realizar el análisis de la metaheurística se empleará diversos resultados experimentales para las corridas de 14 instancias bien conocidas del CVRP que son las propuesta por Christofides, Mingozzi y Toth [CHR01]. Estos resultados permitirán discutir aspectos como: calidad de las soluciones obtenidas, esfuerzo computacional, robustez y fiabilidad.

Además se prevé discutir otros aspectos acerca de la metaheurística en cuestión, que permitirá tener un criterio acerca de su utilización.

2. Breve descripción del problema

El **Vehicle Routing Problem (VRP)**, se define como un grafo no dirigido $G=(V,E)$ donde $V=\{v_{0,}v_{1,}...,v_n\}$ es el conjunto de todos los vértices y $E=\{(v_i,v_j):v_i,v_j\in V,i<j\}$ es el conjunto de los arcos. El vértice v_0 es el depósito de donde podrán salir m vehículos idénticos con capacidad Q, mientras que los vértices restantes representan clientes. La matriz $C=(c_{ij})$ definida sobre E representa los costos no negativos, distancia o tiempo de viaje. Cada cliente tiene una demanda no negativa q_i y un tiempo de servicio no negativo s_i . El **VRP** consiste en diseñar un conjunto de m rutas **i)** del menor costo total, **ii)** cada una empezando y terminando en el depósito, tal que **iii)** cada cliente sea visitado exactamente una vez por el vehículo, **iv)** la demanda total de cada ruta no exceda Q y **v)** que la duración total de cada ruta no exceda una cota D.

El **Capacited Vehicle Routing Problem (CVRP)** es la versión más básica del **VRP**. En el **CVRP** a todos los clientes se les entregarán bienes y sus demandas son conocidas de anticipado. Los vehículos son idénticos y partirán de un único depósito central y solo las restricciones de capacidad de los vehículos son impuestas. El objetivo es minimizar el costo total en despachar a todos los clientes.

3. Descripción del contenido del informe

El presente informe consta de:

- **Portada.**
- **Introducción.** Donde se describe brevemente el problema.
- **Algoritmo para el CVRP.** Se presenta una descripción de la metaheurística **Ant Colony Optimization**, donde se detalla su funcionamiento, estructuras empleadas y otra información adicional.
- **Detalles de software y hardware empleados.**
- **Instrucciones de operación.** Descripción detallada de como compilar y correr el software, así como el estado actual del misma.
- **Resultados experimentales y discusión.** Estudio experimental para caracterizar el rendimiento de la metaheurística propuesta en base a tres aspectos.
- **Conclusiones y recomendaciones.**
- **Referencias bibliográficas.**

ALGORITMO PARA EL CVRP

1. Estructuras de datos utilizados

- **Librería TSP-TEST.V09.9 [THST01]:** Librería usada para optimizar cada ruta encontrada como si fuese una instancia del Traveling Salesman Problem. Las variables más importantes son las siguientes:
  - **distMat:** Matriz de distancias de las ciudades para una instancia dada.
  - **nnMat:** Matriz que contiene en el arreglo $nnMat[i]$ la lista de ciudades más cercanas a la ciudad i .
  - **two_opt_best:** Procedimiento que realiza búsqueda local usando movimientos 2-opt y que usa las listas en nnMat para optimizar la búsqueda
- **Archivos cvrp_instance.h y cvrp_instance.c:** Contienen la información necesaria para resolver una instancia del problema. Las variables y procedimientos que incluye son los descritas a continuación:
  - **cvrp_distMat:** Matriz de distancia redondeadas a enteros entre las ciudades y el depósito. Su tamaño es de $(cvrp\_num\_cities+1)*(cvrp\_num\_cities+1)$. $cvrp\_distMat[i][j]$ contiene la distancia entre la ciudad i y j con $0 \leq i, j \leq cvrp\_num\_cities$. Note que $cvrp\_distMat[i][j]=cvrp\_distMat[j][i]$.
  - **cvrp_num_cities:** Número de ciudades en el problema sin incluir al depósito.
  - **cvrp_demand:** Vector de tamaño $cvrp\_num\_cities+1$ de demandas de los clientes. El +1 en el tamaño permite facilidades de computación.
  - **cvrp_max_route_duration:** Duración máxima permitida para cada ruta.
  - **cvrp_truck_capacity:** Capacidad máxima de cada camión.
  - **cvrp_drop_time:** Tiempo que tarda cada camión en atender a un cliente.
  - **cvrp_file_name(file_name):** Procedimiento que recibe como entrada el nombre de un archivo conteniendo la instancia del problema y carga todas las variables nombradas anteriormente y las necesarias por la librería [THST01].
- **Archivos aoc.h y aoc.c:** Contiene las variables específicas usadas por la metaheurística AOC y los procedimientos necesarios para resolver el problema. Supone que la información de la instancia ya fue cargada. Las variables y procedimientos que incluye se describen a continuación:

- **aoc_evaporation_rate:** Número real en el intervalo [0,1). Es usado para determinar que porcentaje de las feromonas se van a evaporar en cada iteración.
- **aoc_total_ants:** Número que indica la cantidad de soluciones que se van a construir por iteración, es decir, el tamaño de la población.
- **finish_param:** Número que determina el número de iteraciones que se van a realizar, el número de iteraciones sin mejorar que se va a permitir o el número de segundos que se va a correr la metaheurística, dependiendo del criterio de terminación indicado.
- **finish_function:** Función que regresa 0 si todavía no se debe terminar la ejecución de la metaheurística o 1 en caso contrario. Puede ser una de las siguientes:
  - **finish_not_improvement:** Terminar luego de finish_param iteraciones sin mejorar la solución.
  - **finish_fixed_iterations:** Terminar luego de finish_param iteraciones.
  - **finish_time:** Terminar luego de finish_param segundos transcurridos.
- **aoc_pheromone_amplification:** Número real positivo que determina la influencia de las feromonas para la selección de los componentes.
- **aoc_heuristic_amplification:** Número real positivo que determina la influencia de los costos de los arcos entre dos ciudades para la selección de los componentes.
- **aoc_time_best_found:** Al terminar la ejecución de la metaheurística, contiene el segundo, luego de haber iniciado, en el que se encontró la mejor solución.
- **aoc_total_time:** Al terminar la ejecución de la metaheurística, contiene el número de segundos transcurridos desde que se inició hasta que terminó.
- **aoc_iteration_best_found:** Al terminar la ejecución de la metaheurística, contiene el número de la iteración en la que fue encontrada la mejor solución.
- **aoc_total_iterations:** Al terminar la ejecución de la metaheurística, contiene el número de iteraciones totales que se realizaron.
- **aoc_best:** Arreglo que contiene la mejor solución encontrada por la ejecución de la metaheurística. La representación usada es un vector de la forma $\langle 0, c_{1,} c_{2,} \ldots, c_{x_1}, 0, c_{x_1+1}, c_{x_1+2}, \ldots, c_{x_1+x_2}, 0, \ldots, c_{x_1+x_2+\ldots+x_k}, 0 \rangle$ donde cada c_i representa una ciudad, k es el número de rutas en la solución y x_i es la duración de la $i-ésima$ ruta. Cada ruta se encuentra delimitada por dos ceros.
- **aoc_best_size:** Número de rutas en la mejor solución (el k en el punto anterior).
- **aoc_best_duration:** Duración de la mejor solución incluyendo los costos de atender a los clientes.

- **initialize_aoc(argv, argc):** Procedimiento que inicializa todas las variables descritas anteriormente. Argv es un arreglo de Strings que contiene los parámetros de entrada del programa y argc el número de argumentos recibidos (el tamaño de argv). Los posibles valores recibidos se describen en el numeral 3 de esta sección.
- **run_aoc_metaheuristic:** Procedimiento que ejecuta la metaheurística.
- **print_results:** Imprime los resultados obtenidos luego de la ejecución de la metaheurística.

2. Descripción de los principales algoritmos para resolver el problema

Para resolver el **CVRP**, como ya se mencionó anteriormente se empleará la metaheurística **Ant Colony Optimization**. Esta metaheurística consiste en construir un conjunto de soluciones al agregar iterativamente componentes factibles a soluciones parciales tomando en cuenta **(1)** información del problema que se resuelve y **(2)** rastros de feromonas que se actualizan dinámicamente para reflejar la experiencia adquirida durante la búsqueda [HND01].

En el caso del **CVRP** la información usada son los costos de ir de una ciudad a otra y los componentes factibles son el conjunto de arcos que parten de la ciudad actual y llega a ciudades no visitadas por la solución parcial o al depósito y que al agregarlos no rompen las restricciones específicas para el **CVRP** y que permiten que en iteraciones posteriores se pueda seguir construyendo una solución factible hasta obtener la solución completa.

Para entender este procedimiento mejor, supongamos que en una iteración dada, la solución parcial construida es $\langle 0,2,3,1,0,4,5 \rangle$ y que la instancia tiene 7 ciudades. Para elegir los componentes factibles, se evalúa que las rutas $\langle 0,4,5,0 \rangle$, $\langle 0,4,5,6,0 \rangle$ y $\langle 0,4,5,7,0 \rangle$ cumplan las restricciones del problema. En caso de que las 3 sean factibles, los componentes factibles serían $\langle 0,6,7 \rangle$. En caso de que se pueda ir a la ciudad 7 pero no regresar al depósito 0, el componente 7 no sería factible.

La función usada para seleccionar el componente a insertar en la solución parcial es la presentada en [HND01].

Las feromonas se mantienen en una matriz $(cvrp\_num\_cities+1)\,x\,(cvrp\_num\_cities+1)$ donde en la posición i, j se encuentra las feromonas presentes en el arco entre la ciudad i y j. Estas se inicializan en cero y se actualizan al finalizar la construcción de las soluciones en cada iteración.

La actualización consta de un proceso de evaporación y otro de incremento de los componentes usados por las soluciones construidas:

- **Evaporación:** Consiste en asignarle a cada feromona un porcentaje de lo que tenían. Esto es, para $0 \leq i, j \leq cvrp\_num\_cities$ se realiza la siguiente asignación: $pheromones[i][j] = (1 - aoc\_evaporation\_rate) * pheromones[i][j]$.
- **Incremento:** Para todos los componentes usados por las soluciones construidas en la iteración actual, se realiza la siguiente asignación: $pheromones[i][j] = pheromones[i][j] + \frac{1}{duration[k]}$ donde $0 \leq k \leq aoc\_total\_ants$ y el componente i, j se encuentra en la $k-ésima$ solución obtenida. Adicionalmente se actualizan los componentes usados por la mejor solución pero asignándole un factor de amplificación de 3 para remarcar la importancia de estos componentes en una buena solución. Esta estrategia se conoce como estrategia elitista y se encuentra descrita en [HDN02].

Adicional a la implementación inicialmente propuesta, la cual no incluye un proceso de optimización de cada solución construida, en nuestro trabajo aplicamos un optimización usando el procedimiento two_opt_best implementando en [THST01] para cada ruta en las soluciones construidas lo cual permite obtener mejores resultados que sin la optimización.

Finalmente, la implementación descrita produce resultados que varían bastante dependiendo de los parámetros de entrada, los cuales se describen en el siguiente numeral. Para seleccionar los valores por defecto se realizó un análisis estadístico para estudiar como los parámetros de entrada se relacionaban con el costo de la solución obtenida. Se realizó una matriz de correlación y se observó que los costos obtenidos mejoran (se reducen) cuando se le da un factor de amplificación alto para la información del problema y que el costo aumenta cuando se aumenta el factor de amplificación de la información feromonas. El número de hormigas usadas y el factor de evaporación no tienen una influencia notable en el resultado obtenido pero sí en el tiempo en el que tarda en encontrar la mejor solución.

3. Parámetros que utiliza el algoritmo

Este software requiere de un archivo:

- **Instancia del problema:** Este archivo contiene en la primera línea el número de clientes, la capacidad de los vehículos, el tiempo máximo de las rutas y el tiempo de descarga. En la segunda línea contiene las coordenadas del depósito. En las siguientes líneas se encuentran las coordenadas de cada cliente seguido por su demanda.

Adicionalmente, recibe cinco parámetros por consola que son los siguientes:

-a <total_ants>

Especifica el numero total de soluciones que se construyen por iteración.
El numero por defecto es 20.

-e <evaporation_rate>

Especifica el factor de evaporación de las feromonas.
Debe estar entre [0.0, 1.0]. Por defecto: 0.3

-h <heuristic_amplification>

Especifica el factor de amplificación de la información especifica del CVRP usada para seleccionar los componentes.
Debe ser >= 0. Por defecto: 5.0

-p <pheromone_amplification>

Especifica el factor de amplificación de las feromonas usadas para seleccionar los componentes.
Debe ser >= 0. Por defecto: 0.5

-t <finish_function> <finish_param>

Especifica la función para terminar la metaheurísticas. Los posibles valores son:

improvement en cuyo caso finish_param es el numero de iteraciones que pasan sin mejorar la solución

fixed en cuyo caso finish_param especifica el numero de iteraciones máximas

time en cuyo caso finish_param es el numero de segundos máximo que debe correr la solución

Cuando no se especifica, la función usada por defecto es por tiempo con 60 segundos.

DETALLES DE SOFTWARE Y HARDWARE EMPLEADO

Hardware Empleado

Procesador: Intel(R) Core(TM) i3 CPU M 370 @ 2.40GHz
Memoria RAM: 4GiB de 1067MHz

Software Empleado

Sistema Operativo: Debian GNU/Linux Squeezy.
Kernel: 2.6.32-5-amd64
Lenguaje de Programación: C
Compilador: gcc version 4.4.5 (Debian 4.4.5-8)

Estado Actual

El software se encuentra totalmente funcional.

INSTRUCCIONES DE OPERACIÓN

Se provee una carpeta CVRP-AOC que contiene el código fuente del software. Para compilarlo se provee un Makefile y para ejecutarlo se usa el comando:

```
$ ./cvrp <file_name> [-a <total_ants>] [-e <evaporation_rate>] [-h <heuristic_amplification>] [-p <pheromone_amplification>] [-t <finish_function> <finish_param>]
```

Los parámetros son los descritos en la sección de algoritmos usados y los que se encuentran entre corchetes son opcionales. El orden tampoco importa.

RESULTADOS EXPERIMENTALES Y DISCUSIÓN

Los resultados que se presentan a continuación fueron obtenidos usando los parámetros por defectos en su mayoría pero como función de terminación se usó improvement con 60 como parámetro de terminación, es decir, 60 iteraciones sin mejorar.

TABLA DE RESULTADOS 1

A	B	C	D	E	F
vrpnc1	578.4	10.25	8.65	568	1
vrpnc2	925.6	10.82	14.59	907	1
vrpnc3	938.8	13.64	11.03	928	1
vrpnc4	1233.2	19.91	21.19	1204	1
vrpnc5	1625.4	25.87	20.22	1602	1
vrpnc6	622.2	12.02	22.54	602	1
vrpnc7	1043.2	14.68	18.31	1013	1
vrpnc8	1005.6	16.13	12.50	989	1
vrpnc9	1388.4	19.43	8.56	1377	1
vrpnc10	1733.2	24.17	19.93	1711	1
vrpnc11	1146.6	10.03	24.70	1122	1
vrpnc12	851.4	3.89	7.23	843	1
vrpnc13	1636.8	6.21	11.67	1621	1
vrpnc14	919.6	6.14	5.27	913	1

Reseña:

A. Nombre de la instancia.
B. Distancia promedio de 5 corridas de la heurística.
C. Porcentaje de desviación de la distancia promedio de la heurística, con respecto a la solución óptima.
D. Desviación estándar del valor promedio de la heurística.
E. Distancia de la mejor solución obtenida en las 5 corridas de la heurística.
F. Número de ocurrencias de la mejor solución en las 5 corridas de la heurística.

TABLA DE RESULTADOS 2

A	B	C	D	E	F
vrpnc1	524,61	6,36	8,27	0	1,80
vrpnc2	835,26	6,43	8,59	0	3,13
vrpnc3	826,14	4,95	12,33	1	5,94
vrpnc4	1028,42	7,54	17,07	2	13,52
vrpnc5	1291,29	5,17	24,06	11,4	30,62
vrpnc6	555,43	8,20	8,38	0	1,30
vrpnc7	909,68	6,74	11,36	0,6	2,31
vrpnc8	865,94	3,82	14,21	3,4	5,52
vrpnc9	1162,55	10,96	18,45	4,8	9,05
vrpnc10	1395,85	7,10	22,58	8	21,21
vrpnc11	1042,11	0,09	7,67	1,2	9,49
vrpnc12	819,56	1,88	2,86	0	5,35
vrpnc13	1541,14	2,46	5,18	2,2	5,45
vrpnc14	866,37	0,53	5,38	0,8	5,04

Reseña:

A. Nombre de la instancia.
B. Distancia de la solución óptima.
C. Porcentaje de desviación de la mejor solución de la heurística Tabu Search con respecto a la solución óptima.
D. Porcentaje de desviación de la mejor solución de la heurística AOC con respecto a la solución óptima.
E. Tiempo promedio de las 5 corridas de la heurística TS en segundos.
F. Tiempo promedio de las 5 corridas de la heurística AOC en segundos.

ANÁLISIS DE RESULTADOS

Tomando en cuenta los tres aspectos:

1. Calidad de las soluciones que son obtenidas.

Tal como se puede ver en las tablas de resultados, se logró obtener buenos soluciones, a excepción de las instancias 5, 9, 10 ya que su desviación estándar fue un poco alta. Para los demás valores podemos decir que la desviación estuvo en un rango aceptable. Para ninguna de las instancias de las 5 corridas se logro el valor óptimo.

Sin comparamos los resultados obtenidos mediante Ant Colony Optimization con los de Tabu Search, se obtuvo para todos las instancias peores resultados. Solo para las instancias 1, 2, 6, 12, 13 se obtuve bastantes cercanos, lo cual lo vemos en la comparación de las columnas B y C de la Tabla 2. Cambien cabe destacar que que para la metaheurística Tabu Search se utilizo la heurística Clarke-Wright [CLWR01] para obtener la soluciones iniciales, la cual es una de las mejores para resolver este problema, y esta no se utilizo en la metaheurística ACO, ya que esta construye las soluciones desde cero.

Los resultados obtenidos concuerdan con los resultados obtenidos por otros autores, ya que la metaheurística Tabu Search ha presentado mejor soluciones en comparación con las demás metaheurísticas.

2. Esfuerzo computacional

El numero de hormigas a utilizar para el ACO influye notablemente en el tiempo de ejecución del mismo, sin degradar los resultados obtenidos. Esto se debe a que al aumentar el numero de hormigas, aumenta el numero de soluciones construidas en cada solución, por lo que se tardara mas tiempo construyendo soluciones sin actualizar las feromonas, el cual es uno de los factores característicos de esta metaheurística.

Si comparamos el tiempo computacional requerido para la metaheurística ACO con el requerido por Tabu Search bajo el criterio de terminación de cierto numero de iteraciones sin mejora; para el ACO es mayor puesto que no una heurística especializada para la construcción de la solución, mientras que en Tabu Search, como se menciono anteriormente, se utilizo la heurística Clarke-Wright, por lo que Tabu Search tarda poco en mejorarla dada la calidad de la misma.

3. Robustez y fiabilidad

La metaheurística Ant Colony Optimization por tener una gran cantidad de parámetros configurables para ejecutar el algoritmo; los resultados que produce se ven muy influenciados por ellos. Sin embargo tal como se puede ver en la Tabla 1 para la columna D, aun con unos parámetros fijos se obtienen soluciones algo distintas, pudiendo observarse para la instancia 12 una desviación de 24.70.

CONCLUSIONES Y RECOMENDACIONES

En base al análisis y discusión de resultados, realizado a lo largo del informe, podemos en primer lugar concluir que esta heurística representa una buena opción para resolver el problema CVRP y que tiene muchas variables ajustables para obtener los resultados.

Podemos decir que las ventajas de usar la metaheurística de ACO para resolver el problema CVRP, es que dado la complejidad del mismo, nos permite jugar bastante con la configuración que podemos usar para resolverlo y obtener así buenos resultados en un tiempo razonable.

Entre las desventajas podemos decir, que este algoritmo no garantiza obtener soluciones óptimas y la solución que se obtenga depende de un factor probabilístico.

Evaluando las ventajas y desventajas se puede señalar que esta metaheurística es una buena opción para resolver el problema del CVRP. Sin embargo, la metaheurística de Tabu Search ha mostrado ser mucho mejor en la práctica que cualquier otra metaheurística para este problema en específico. Para mejorar las soluciones que da la solución ACO se puede implementar mejoras en el algoritmo de búsqueda local usado para mejorar cada individuo de la población o seleccionar mejores funciones para evaporación y actualización de las feromonas así como diferentes criterios para seleccionar el componente a añadir en la solución parcial en cada paso.

REFERENCIAS BIBLIOGRÁFICAS

CHR01: Christofides, N. Combinatorial Optimization, 1979

THST01: Thomas Stuetzle, TSP-TEST, Version 0.9, 2004, http://www.sls-book.net

HND01: Marco Dorigo and Thomas Stützleu, Ant Colony Optimization: Overview and Recent Advances,

HDN02: Marco Dorigo and Thomas Stützleu, Ant Colony Optimization: Overview and Recent Advances,

CLWR01: Clarke and Wright, , , http://es.scribd.com/doc/35387231/Clarke-Wright