

**Universidad Simón Bolívar**

**Departamento de Computación y Tecnología de la Información**

**CI5652 - Diseño de Algoritmos II**

**Trimestre Abril - Julio 2012**

**INFORME II**

**Resolviendo el Capacited Vehicle Routing Problem (CVRP)**

**Integrante:**

**Bishma Stornelli 08-11094**

**Vicente Santacoloma 08-11044**

**Sartenejas, 25 de junio de 2012**

**INTRODUCCIÓN**

1. **Motivación del proyecto**

El presente informe pretende realizar una descripción y análisis de la metaheurística **Tabu Search** para resolver el problema del **Capacited Vehicle Routing Problem (CVRP).**

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 [1]. 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.

1. **Breve descripción del problema**

El **Vehicle Routing Problem (VRP)**, se define como un grafo no dirigido donde es el conjunto de todos los vértices y es el conjunto de los arcos. El vértice es el depósito de donde podrán salir m vehículos idénticos con capacidad Q, mientras que los vértices restantes representan clientes. La matriz definida sobre E representa los costos no negativos, distancia o tiempo de viaje. Cada cliente tiene una demanda no negativa y un tiempo de servicio no negativo . 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.

1. **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 **Tabu Search**, donde se detalla su funcionamiento, estructuras empleadas y otra información adicional.
* **Detalles de software y hardware empleados.**
* I**nstrucciones 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**

**Reseña:**

* **Paquete**
* **Clase**
* **CVRP:** Algoritmo para la solución del CVRP mediante la metaheurística Tabu Search.
* **abstracts**
* **TerminationCriteria:** Clase abstracta del criterio de terminación.
* **classes**
* **ClarkeAndWrightAlgorithm:** Heurística Clarke and Wright [2] para obtener la solución inicial del CVRP.
* **CustomerTabu:** Tabu que prohíbe mover a un cliente.
* **CustomerRouteTabu:** Tabu que prohíbe mover a un cliente en una ruta específica.
* **CustomerRouteTabuPosition:** Tabu que prohíbe mover un cliente a una ruta en una posición específica.
* **Instance:** Instancia del problema en cuestión que además contiene la configuración a realizar del problema.
* **MoveSingle:** Movimiento de un cliente a una ruta en una posición.
* **MoveSwap:** Movimiento de intercambiar dos clientes.
* **MoveTwoOpt:** Movimiento de 2-opt.
* **Neighbor:** Un vecino generado del CVRP para la solución en actual. Por cuestiones de almacenamiento en lugar de ser esta la solución vecina, se tendrá la referencia a la solución actual más en lo movimiento.
* **NeighborSelectorBest:** Selecciona el mejor vecino de la vecindad.
* **NeighborSelectorFirst:** Selecciona el primer mejor vecino de la vecindad.
* **NeighborhoodStructureClassic:** Estructura para generar una vecindad a partir de una solución, mediante el movimiento de un cliente de una ruta a otra.
* **NeighborhoodStructureSwap:** Estructura para generar una vecindad a partir de una solución, mediante el intercambio de dos clientes.
* **NeighborhoodStructureTwoOpt:** Estructura para generar una vecindad a partir de una solución, mediante el 2-opt.
* **NeighborhoodStructureLambda:** Estructura para generar una vecindad a partir de una solución, mediante la combinación de cualquiera de las tres estructuras generadoras anteriores.
* **PrintableSolution:** Guarda el estado de una solución, almacenando solo los datos requeridos para imprimirla.
* **Route:** Representa una ruta de una solución.
* **Solution:** Representa una solución para el CVRP.
* **TerminationCriteriaImproving:** Criterio de terminación por un cierto número de iteraciones, basado en la cantidad de clientes, sin mejorar la solución actual encontrada.
* **TerminationCriteriaIteration:** Criterio de terminación por un número de iteraciones basado en la cantidad de clientes.
* **interfaces**
* **Move:** Interfaz de cada las estructuras de movimientos.
* **NeighborSelector:** Interfaz de las estructuras para la selección de un vecino de la vecindad.
* **NeighborhoodStructure:** Interfaz de las estructuras para la generación de una vecindad.
* **Tabu:** Interfaz de las estructuras del Tabu.
* **exceptions**
* **MaxCapacityExceededException:** Excepción por violación de la capacidad de carga permitida por un vehículo.
* **MaxDurationExceededException:** Excepción por violación de la duración permitida por una ruta.
* **TabuListFullException:** Excepción por haberse llenado la lista tabu en base a un valor dado. En dicho caso deberá ser parcialmente vaciada.
* **TerminationCriteriaNotStartedExcepcion:** Excepción por no inicializar el criterio de terminación antes de que este inicie.
* **UnexpectedAmountOfCustomersException:** Excepción por cantidad no permitida de clientes en la generación de un vecino.

1. **Descripción de los principales algoritmos para resolver el problema**

Para resolver el **CVRP**, como ya se mencionó anteriormente se empleará la metaheurística **Tabu Search**. Esta metaheurística es similar a hill-climbing pero al realizar movimientos, guarda una lista de movimientos tabus para evitar volver a soluciones ya evaluadas. De esta lista se eliminan los elementos más viejos cada cierto tiempo para poder obtener nuevas soluciones.

En el caso del **CVRP** se almacenarán los diversos movimientos realizados para prohibir que estos vuelvan a ser realizados por un periodo dado. Para este caso volverán a permitir movimientos prohibidos cada vez que se llene la lista Tabu, siendo estos movimientos los realizados con mayor anterioridad.

Para obtener la solución inicial del problema se utiliza el algoritmo de **Clarke & Wrights** [2], el cual consistes en recortar la duración de cada ruta mediante la mezcla de dos rutas para así reducir la distancia entre un extremo de cada ruta con el depósito.

Las vecindades se generan usando la estructura NeighborhoodStructureLambda, la cual realiza movimientos de tipo SingleMove, SwapMove y TwoOptMove.

De la vecindad generada, se escoge la mejor, usando NeighborSelectorBest.

Cabe destacar que lo descrito hasta ahora fue lo usado para obtener los datos que se presentan más adelante en este documento. Sin embargo, la solución fue diseñada para usar diferentes configuraciones mediante el uso del archivo settings. De esta forma, por ejemplo, se puede usar la estructura simple para generar los vecinos y seleccionar el primero mejor con un criterio de terminación basado en un número fijo de iteraciones.

Después de varias pruebas, se encontró que las mejores soluciones se hallaban usando la configuración descrita en la primera parte de esta sección.

1. **Parámetros que utiliza el algoritmo**

Este software requiere de dos archivos:

* **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.
* **settings:** archivo de configuración para especificar las modalidades con las que se resolverá el problema. Más detalles de este archivo en la sección de Instrucciones de Operación.

**DETALLES DE SOFTWARE Y HARDWARE EMPLEADO**

**Hardware Empleado**

**Procesador:** Intel(R) Core(TM) 2 Duo CPU    P8600    @ 2.40GHz

**Memoria RAM:** 4GiB

**Software Empleado**

**Sistema Operativo:** Debian GNU/Linux Wheezy.

**Kernel:** Linux version 3.1.0-1-amd64

**Lenguaje de Programación:** JAVA

**java version "1.7.0\_03"**

**OpenJDK Runtime Environment** (IcedTea7 2.1.1) (7~u3-2.1.1-1)

**OpenJDK 64-Bit Server VM** (build 22.0-b10, mixed mode)

**NetBeans IDE 7.1**

**Estado Actual**

Algunos métodos diseñados no fueron implementados por falta de tiempo y por la complejidad de los mismos. Específicamente el generador Granular no fue implementado. Sin embargo debido a los buenos resultados obtenidos por la heurística **Clarke & Wrights**, y por los generadores Full y Random implementados es poco probable que se vaya a mejor mucho la solución obtenida del **CVRP**.

**INSTRUCCIONES DE OPERACIÓN**

El software viene dado en la carpeta generada por NetBeans. Para utilizarlo se puede abrir el proyecto con esta IDE, o compilar el código fuente que se encuentra en la ruta relativa del proyecto CVRP/src/cvrp/ mediante:

***$ javac \*.java***

Sin embargo es recomendable la primera opción.

Para ejecutar el proyecto con una instancia del problema, usando NetBeans, se debe agregar la instancia manualmente en el método main de la clase CVRP.java, tal como se señala:

.

.

Instance instance **=** **new** Instance**(**"instanciasCVRP/vrpncX.txt"**,** "settings"**);**

**.**

**.**

En cuyo caso el archivo de configuración ya se encuentra por defecto.

Si el software se esta usando sin NetBeans, se debe descomentar la línea:

*// Instance instance = new Instance(args[0],args[1]);*

del método main para la clase CVRP.java, y pasarle como primer argumento la instancia del problema y como segundo el archivo de configuración. Y comentar la línea antes mencionada para el uso con NetBeans. Para ejecutarlo se debe escribir en la consola:

***$ java CVRP instancia archivo\_de\_configuración***

Todas las instancias del problema utilizadas se encuentran en la ruta relativa [CVRP](https://github.com/tamerdark/CVRP)/**instanciasCVRP/**

**El archivo de configuración por defecto se encuentra en la ruta relativa CVRP/settings. A continuación se describe este archivo:**

TERMINATION\_CRITERIA [I|B]

NEIGHBORHOOD\_STRUCTURE [C|S|L|2OPT]

NEIGHBORHOOD\_GENERATOR [F|R|G]

NEIGHBOR\_SELECTOR [F|B]

TERMINATION\_CRITERIA

Criterio de terminación a utilizar

I Hasta cumplirse un número de iteración dado por el software

B Hasta cumplirse un número de iteración dado sin mejora por el software

NEIGHBORHOOD\_STRUCTURE

C Generación clásica de vecindario

S Generación del vecindario mediante swap. No funciona.

L Generación lambda del vecindario

2OPT Generación del vecindario mediante 2-opt

NEIGHBORHOOD\_GENERATOR

F Generación completa de una gran cantidad de vecinos

R Generación aleatoria de un vecino

G Generación granular de un vecino. No fue implementado

NEIGHBOR\_SELECTOR

F Selecciona el primer mejor vecino de la vecindad

B Selecciona el mejor vecino de la vecindad

**RESULTADOS EXPERIMENTALES Y DISCUSIÓN**

**TABLA DE RESULTADOS 1**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **E** | **F** | **G** |
| **vrpnc1** | 565,2 | 7,73717619 | 4,086563 | 558 | 1 | 50 |
| **vrpnc2** | 889 | 6,43392477 | 0 | 889 | 5 | 75 |
| **vrpnc3** | 869,4 | 5,23640061 | 1,67332 | 867 | 1 | 100 |
| **vrpnc4** | 1106 | 7,54361059 | 0 | 1106 | 5 | 150 |
| **vrpnc5** | 1362,8 | 5,53787298 | 4,549725 | 1358 | 1 | 199 |
| **vrpnc6** | 610,4 | 9,89683669 | 6,542171 | 601 | 1 | 50 |
| **vrpnc7** | 975,6 | 7,24650427 | 3,507136 | 971 | 1 | 75 |
| **vrpnc8** | 936 | 8,09062984 | 24,77902 | 899 | 1 | 100 |
| **vrpnc9** | 1301,4 | 11,9435723 | 7,700649 | 1290 | 1 | 150 |
| **vrpnc10** | 1500,6 | 7,50438801 | 4,560702 | 1495 | 1 | 199 |
| **vrpnc11** | 1043 | 0,08540365 | 3,082207 | 1041 | 3 | 120 |
| **vrpnc12** | 835 | 1,88393772 | 0 | 835 | 5 | 100 |
| **vrpnc13** | 1582,8 | 2,70319374 | 4,38178 | 1579 | 2 | 120 |
| **vrpnc14** | 873 | 0,76526196 | 1,81659 | 871 | 1 | 100 |

**Reseña:**

1. Nombre de la instancia.
2. Distancia promedio de 5 corridas de la heurística.
3. Porcentaje de desviación de la distancia promedio de la heurística, con respecto a la solución óptima.
4. Desviación estándar del valor promedio de la heurística.
5. Distancia de la mejor solución obtenida en las 5 corridas de la heurística.
6. Número de ocurrencias de la mejor solución en las 5 corridas de la heurística.
7. Número de clientes.

**TABLA DE RESULTADOS 2**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **E** | **F** | **G** | **A** | **B** |
| **vrpnc1** | 524,61 | 558 | 6,364728084 | 0 | 8008,6 | 3007,6 | 0 | 0 |
| **vrpnc2** | 835,26 | 889 | 6,433924766 | 0 | 7607,8 | 106,8 | 0 | 0 |
| **vrpnc3** | 826,14 | 867 | 4,945892948 | 0 | 11200,6 | 1199,6 | 1 | 0 |
| **vrpnc4** | 1028,42 | 1106 | 7,543610587 | 0 | 15001 | 0 | 2 | 0 |
| **vrpnc5** | 1291,29 | 1358 | 5,166151678 | 7 | 50895,8 | 30994,8 | 11,4 | 6,8 |
| **vrpnc6** | 555,43 | 601 | 8,204454207 | 0 | 8463,6 | 3462,6 | 0 | 0 |
| **vrpnc7** | 909,68 | 971 | 6,740831941 | 0 | 10834,8 | 3333,8 | 0,6 | 0 |
| **vrpnc8** | 865,94 | 899 | 3,817816477 | 2 | 30260,2 | 20259,2 | 3,4 | 2,2 |
| **vrpnc9** | 1162,55 | 1290 | 10,96296933 | 6 | 26466,8 | 5062,2 | 4,8 | 1,8 |
| **vrpnc10** | 1395,85 | 1495 | 7,103198768 | 6 | 32727,6 | 12826,6 | 8 | 3 |
| **vrpnc11** | 1042,11 | 1043 | 0,106514667 | 0 | 15157,6 | 3156,6 | 1,2 | 0,2 |
| **vrpnc12** | 819,56 | 835 | 1,883937723 | 0 | 10001 | 0 | 0 | 0 |
| **vrpnc13** | 1541,14 | 1579 | 2,456623019 | 0 | 18486 | 6485 | 2,2 | 0,4 |
| **vrpnc14** | 866,37 | 871 | 0,534413703 | 0 | 12228 | 2227 | 0,8 | 0 |

**Reseña:**

1. Nombre de la instancia.
2. Distancia de la solución óptima.
3. Distancia de la mejor solución obtenida en las 5 corridas de la heurística.
4. Porcentaje de desviación de la mejor solución de la heurística con respecto a la solución óptima.
5. Tiempo en encontrar la mejor solución.
6. Número promedio total de iteraciones.
7. Número promedio de iteraciones para encontrar la mejor solución.
8. Tiempo promedio de las 5 corridas de la heurística en segundos.
9. Tiempo promedio para encontrar la mejor solución.

**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 muy buenas soluciones para todas las instancias. Esto se puede apreciar en que la desviación fue bastante pequeña. Sin embargo para ninguna de las instancias se logro el óptimo.

Otro aspecto muy importante a señalar es que la distancia promedio de las cinco corridas no difiere mucho de la mejor solución obtenida.

1. **Esfuerzo computacional**

El esfuerzo computacional requerido para todas las instancias fue bastante pequeño. En muchos casos fue de cero segundos. Vale la pena señalar que el tiempo promedio de las 5 corridas de la heurística difiere mucho del tiempo promedio en encontrar la solución. También difiere significativamente el número promedio de iteraciones de las 5 corridas de la heurística y el número promedio de iteraciones en encontrar la mejor solución.

Lo dicho anteriormente resalta la gran ventaja del uso de esta metaheurística en combinación con otras heurísticas para resolver el problema, por el poco tiempo invertido para obtener buenos resultados.

1. **Robustez y fiabilidad**

La metaheurística **Tabu Search** implementada permite obtener buenos resultado, sin verse afectada por los distintos números de clientes. Parte de este éxito se debe al uso de la heurística Clarke & Wrigth para conseguir la solución inicial, ya que esta misma en la modalidad de parallel que va construyendo las rutas en paralelo, permite minimizar la duración de la ruta, utilizando un costo computacional bastante bajo. Si no se utilizara esta heurística aun se conseguirían buenos resultados, pero un poco mayores a los presentados en la tabla de resultados.

**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 tabu-search 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; la solución que se obtenga depende de varios factores que se pueden considerar aleatorios y por lo tanto no es determinístico; y finalmente, no es posible proporcionar una cota superior del tiempo computacional, especialmente para problemas muy grandes.

Evaluando las ventajas y desventajas se puede señalar que esta metaheurística es una de las mejores para resolver el problema del CVRP. Sin embargo, es necesario implementar mejores formas de generar las vecindades, agregando por ejemplo, criterios de aspiración, diversificadores e intensificadores que permitan alejarnos un poco más del espacio de búsqueda que se explora y optimizar ciertas partes de la solución para lograr conseguir mejores óptimos locales que puedan llegar a ser óptimos globales.

**REFERENCIAS BIBLIOGRÁFICAS**

1. Christofides, N. Combinatorial optimization. A Wiley-Interscience Publication. Wiley, 1979.
2. http://www.scribd.com/doc/35387231/Clarke-Wright
3. Cordeau, J., Gendreau, M., Hertz, A., Laporte, G., and Sormany, J. New heuristics for the vehicle routing problem. Logistics systems: design and optimization (2005), 279-297.
4. Gendreau, M., Laporte, G., and Potvin, J. Metaheuristics for the capacitated vrp. The vehicle routing problem 9 (2002), 129-154.
5. Laporte, G., Gendreau, M., Potvin, J., and Semet, F. Classical and modern heuristics for the vehicle routing problem. International transactions in operational research 7, 4-5 (2000), 285-300.
6. Golden, B., Raghavan, S., and Wasil, E. The vehicle routing problem: latest advances and new challenges, vol. 43. Springer Verlag, 2008.
7. Toth, P., and Vigo, D. The Vehicle Routing Problem. Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, 2002.