Algorítmica I - EPD Evaluable III

- 2º GIISI -

Realizado por:

- Víctor Jesús Reina López
- Jaime Baquerizo Delgado

Fecha de entrega:
07/01/2022

INTRODUCCION

Esta epd evaluable consiste en la continuación del problema del TSP hecho anteriormente pero con otros algoritmos. Este problema consiste en que dada una lista de ciudades y las distancias entre cada par de ellas, calcular la ruta más corta posible que visita cada ciudad exactamente una vez y regresa a la ciudad origen. Se hablará de las conclusiones y resultados obtenidos al final de este trabajo

ALGORITMOS

Backtracking (Vuelta atrás)

El método de Backtracking, o también llamado “vuelta atrás”, es un método utilizado para este tipo de problemas de optimización, ya que en el problema del TSP queremos llegar al camino con un coste lo más pequeño posible dentro de nuestras capacidades.
Este método consiste en calcular muchas de las combinaciones posibles como solución, y una vez obtenidas, elegir entre ellas la más óptima. Con este algoritmo lo que estamos intentando es maximizar un beneficio, o minimizar un coste.

PASOS:
-Identificar la solución (x1, x2, . . . , xN).
-Obtener las restricciones.
-En cada etapa k (máximo N), habrá un bucle que probará todos los posibles valores de xk :
    -Escoger un valor xk y comprobar que cumple las restricciones
    -Si la solución parcial no es válida, elegir otro valor y volver al paso anterior.
    -Si la solución parcial si es válida:
        1 Y hemos llegado a la solución final, el problema finalizará.
        2 Y no hemos llegado a la solución final, pasamos a la siguiente etapa k + 1.

En nuestro código, obtenemos todos los archivos “.tsp” de la carpeta data, y calculamos 1 por 1 sus recorridos. Calculamos los caminos, comprobando si las ciudades a estudiar están ya o no en el camino hasta completarlo, y una vez terminado lo introducimos en un vector donde después sacaremos la mejor solución hasta el momento.
Hemos establecido un criterio de parada de 10000000 iteraciones, por lo que una vez que ya ha superado ese límite devolvemos el recorrido con menos mejor resultado hasta el momento.

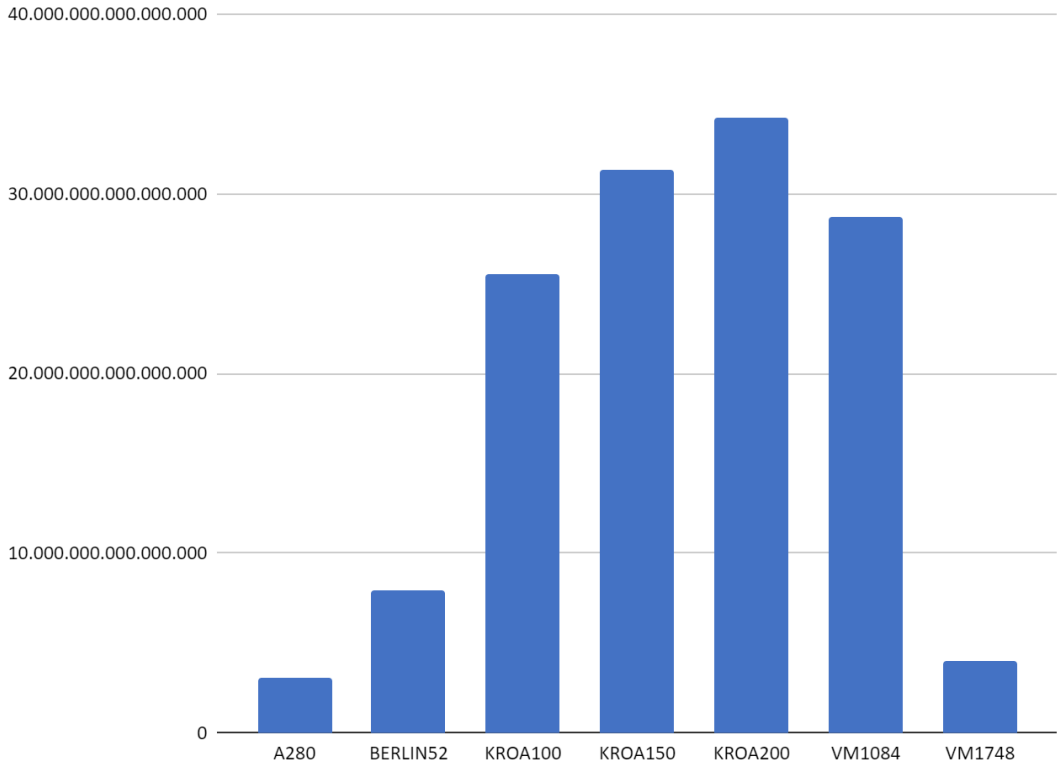
Pseudocódigo

```
public double caminoDeDistMinima(cities[][], camino[],ciudad, visitas, Smejor, Sparcial) {
	Cities opcion[] = cities[ciudadAct];
	pasosR++;
	int opt = 1;
	if (pasosR % 1000000 == 0) {
		System.out.println("current recursion steps: " + pasosR);
	}
	while (opt <= opcion.length - 1 && pasosR < 10000000) {
		camino[visitas] = opcion[opt].city;
		Sparcial += opcion[opt].distance;
		if (!esSolParcial(Smejor, Sparcial)) {
			camino[visitas] = 0;
			Sparcial -= opcion[opt].distance;
			break;
		}
		if (opcionValida(opcion[opt], Smejor, Sparcial, camino)) {
			visitas++;
			if (visitas == camino.length) {
				Smejor = Sparcial + cities[opcion[opt].city][camino[0]].distance;
				System.out.println("Nueva distancia mejorada: " + Smejor);
				pasosR = 0;
			} else {
				Smejor = caminoDeDistMinima(cities, camino, opcion[opt].city,
visitas, Smejor, Sparcial);
			}
			visitas--;
		}
		camino[visitas] = 0;
		Sparcial -= opcion[opt].distance;
		opt++;
	}
	return Smejor;
}
```

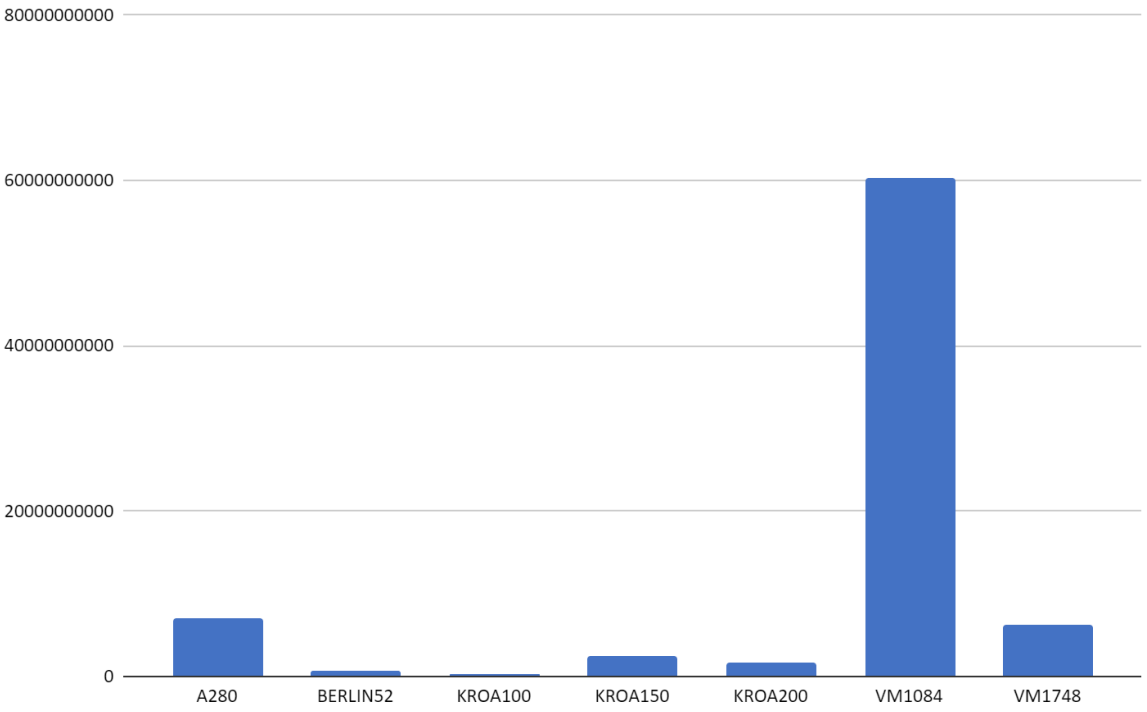
En este algoritmo nos entra como parámetros, la matriz ciudades, el vector de los caminos, la ciudad por la que nos encontramos, el número de ciudades que llevamos visitadas, la mejor solución y el recorrido hasta el momento.
Primero inicializamos las variables necesarias para el cálculo y mostramos por pantalla las repeticiones que llevamos cada vez que el número de iteraciones (pasosR) es divisible entre 10 millones. Luego comienza nuestro bucle “while” donde comenzaremos con recursividad a buscar el camino, siempre y cuando no hayamos superado el tamaño del vector ciudades o hayamos superado el número límite de iteraciones.
Lo primero que hacemos al entrar en el bucle es aumentar por la distancia el recorrido que llevamos hasta el momento y situarnos en la nueva ciudad.
Hay que tener en cuenta que una opción de camino es válida si la solución parcial generada por dicho camino no empeora la actual mejor solución o si el camino aún no está completo, por lo tanto si no se cumplen estos requisitos salimos del bucle. En caso de que sea una opción de camino válida, aumentamos las visitas y comprobamos si estamos en el final del recorrido, si es así establecemos la nueva mejor solución y actualizamos de nuevo el número de pasos a 0, ya que hemos terminado el camino. Y si estamos todavía en el cálculo del camino volvemos a llamar al método recursivamente. Por último devolvemos la solución calculada

EXPERIMENTACIÓN

BACKTRACKING RESULTADOS



BACKTRACKING TIEMPOS



	Backtracking
A280	3.070.949.3
BERLIN52	7.893.144.7
KROA100	25.560.583.
KROA150	31.354.035.
KROA200	34.239.149.
VM1084	28.705.038.
VM1748	4.031.653.1

RESULTADOS

	Backtracking
A280	6985364721
BERLIN52	597992311
KROA100	224932987
KROA150	2394897547
KROA200	1712241361
VM1084	6022981101
VM1748	6171510771

TIEMPOS

¿Solución más óptima?

Esté algoritmo generalmente está bastante bien planteado para la solución al problema del TSP. Ya que estamos calculando numeroso caminos posibles y escogiendo el mejor de ellos, por lo que es probable que encontremos una solución bastante óptima.
Aun así este problema plantea una complejidad exponencial, por lo que cuanto más ciudades haya menos posible es encontrar la solución más óptima, o necesitaríamos un número de intentos bastante grande y conllevaría un gran coste computacional.

CONCLUSIONES

Las conclusiones son que el algoritmo backtracking si existe una solución, la obtiene. Es un esquema sencillo de implementar y adaptable a las características de cada problema.

Tiene un coste exponencial en la mayoría de los casos ademas de que si el espacio de búsqueda es infinito, puede que no se encuentre la solución (el archivo usa es exageradamente costoso). También consume mucha memoria debido a la recursividad.

BIBLIOGRAFÍA

---

https://www.youtube.com/watch?v=vdVpRjO7g84
https://www.youtube.com/watch?v=LUuSLdBSejQ
https://www.youtube.com/watch?v=etQN4EfYN7k
http://dis.um.es/~nmarin/transparencias-backtracking-AED-II.pdf
EB BACKTRACKING
EPD BACKTRACKING