UNIVERSIDAD DE GRANADA

Algorítmica

Curso 2023

Practica 2:

Divide y Vencerás: problema del frente de Pareto

Amador Carmona Méndez

Miguel Ángel López Sánchez

Índice:

1. Estudio del problema mediante un método básico.

Método básico o Fuerza bruta:

Explicación:

Recorremos todos los puntos pi comparándolos con todos los demás puntos pj y comprobamos si son dominados por los demás puntos o si son dominantes, si es dominante lo metemos en la lista solución si no es dominante no.
Para verificar si el punto i no es dominado por otro punto j en el conjunto C, debes comparar las coordenadas de ambos puntos en cada dimensión del espacio K-dimensional. Si para todas las coordenadas de i y j se cumple que vi[a] >= vj[a], entonces el punto i no es dominado por el punto j y no es necesario continuar comparando con otros puntos en el conjunto. Por lo tanto, si al menos para una coordenada a se cumple que vi[a] > vj[a], entonces el punto i es dominado por el punto j y se puede detener la comparación. En este caso, se procede a comparar el siguiente punto en el conjunto C.

Pseudocódigo:

Funcion Dominantes(L: lista de puntos con características, K: número de características):

puntosFrontera: lista de puntos con características.
Para cada punto pi de la lista **hacer**:
Para cada otro punto pj de la lista **hacer**:
si ***EsDominante(pi,pj,K)*** **entonces:**
esNoDominado = verdadero
si (EsNoDominado)
añadir punto a puntos frontera
Devolvemos puntos frontera.

Función esDominante (puntoi, puntoj, número de caracteristicas K):

para cada caracteristica c dentro de las K-caracteristicas **hacer**:
si pj[c]>pi[c] **entonces:**
alMenosIgual = Verdadero
si pj[c]<pi[c] **entonces:**
alMenosMayor = Falso
si alMenosMayor = alMenosMenor = true
devolvemos Verdadero;
sino
devolvemos falso;

Eficiencia:

Siendo N el tamaño de la lista de puntos y K el número de características:
N y K son las variables de las que depende el tamaño del problema.
La eficiencia en el peor caso es N para recorrer todos los puntos por N para recorrer el punto a comparar por K características para compararlas entre los puntos, es decir, su eficiencia en el peor caso es de (N*N*K) o lo que es lo mismo $\rightarrow O(N^2 * K)$

2. Estudio del problema por la técnica Divide y Vencerás mediante dos estrategias.

(Estudiar si el problema puede ser abordado mediante la técnica Divide y Vencerás. Si es posible, piense en **al menos dos estrategias** para dividir el problema en subproblemas y fusionar sus soluciones. Seleccione una de ellas como la mejor, de forma justificada, y realice el diseño completo Divide y Vencerás. Analice su eficiencia teórica.)

Divide y Vencerás estrategia 1 (Aplicamos DyV al tamaño de la lista de puntos: afectará en la eficiencia a N^2):

Explicación:

En este caso aplicamos la técnica de Divide y Vencerás, lo aplicamos para dividir la lista de puntos en dos y así que sea más pequeño el caso en cuanto a número de puntos.
El caso base o más pequeño es que la lista de puntos sea de tamaño 2 y por lo tanto devuelve cuál es dominante.

Pseudocódigo:

Funcion Dominantes(L: lista de puntos con características, K: número de características):

puntosFrontera: lista de puntos con características.
Si es el tamaño de L es==2
si *EsDominante(p[0],p[1])*
añadir punto pi a puntosFrontera
Devolvemos puntos frontera.
En otro caso si es el tamaño de L es==3
si *EsDominante(p[0],p[1])*
añadir punto pi a puntosFrontera
si *EsDominante(p[1],p[2])*
añadir punto pi a puntosFrontera
si *EsDominante(p[0],p[2])*
añadir punto pi a puntosFrontera
Devolvemos puntos frontera.
En otro caso:
Dividimos la lista en dos sublistas L1 y L2
añadimos a puntosFrontera ***Dominantes(L1,K)***
añadimos a puntosFrontera ***Dominantes(L2,K)***
Devolvemos puntos frontera.

Función esDominante (puntoi, puntoj, número de caracteristicas K):

para cada caracteristica c dentro de las K-caracteristicas **hacer**:
si pj[c]<pi[c] **entonces**:
devolvemos Falso

devolvemos Verdadero

Eficiencia:

O(N*log(N)*K)

Divide y Vencerás estrategia 2 (Aplicamos DyV al tamaño de la lista de características: afectará en la eficiencia a K):

Explicación:

En este caso utilizamos la técnica de Divide y Vencerás para hacer más pequeños las características de ambos puntos y que así compararlos no sea tan costoso.

Pseudocódigo:

Funcion Dominantes(L: lista de puntos con características, K: número de características):

puntosFrontera: lista de puntos con características.
Para cada punto pi de la lista **hacer**:
Para cada otro punto pj de la lista **hacer**:
si ***EsDominante(pi,pj)***
añadir punto pi a puntosFrontera
Devolvemos puntos frontera.

Función esDominante (puntoi, puntoj, inicio: caracteristica primera de las características, fin: ultima caracteristica de las caracteristicas):

mitad: entero que nos dice la mitad de las caracteristicas.
si inicio==fin
si puntoj[inicio]<puntoi[inicio]
devolvemos Falso
En otro caso:
devolvemos Verdadero
En otro caso:
mitad=(incio+fin)/2
dom1=***esDominante (puntoi, puntoj, inicio, mitad)***
dom2=***esDominante (puntoi, puntoj,mitad+1, fin)***
devolvemos dom1 **Y** dom2

Eficiencia:

O(n2*K*log(K))

Selección de la mejor Estrategia:

La mejor estrategia a nivel de eficiencia es la primera, ya que reducimos los tiempos cuadráticos a tiempos N*log(N), con la segunda mantenemos los tiempos cuadráticos y aumentamos el coste de Kl por lo que preferimos reducir los tiempos cuadráticos.

3. Implementación y resolución del problema del Umbral.

(Implemente los algoritmos básico y Divide y Vencerás. Resuelva el problema del umbral de forma experimental. Aunque los algoritmos deben ser generales y permitir cualquier valor de K, para la práctica asume siempre un valor de K=10.)

Implementación del Algoritmo Fuerza Bruta:

```
struct Punto {
    vector<int> coordenadas;
};

bool esDominante(Punto pi, Punto pj, int K) {
    bool alMenosIgual = true;
    bool alMenosMayor = false;

    for (int a = 0; a < K; a++) {
        if (pj.coordenadas[a] > pi.coordenadas[a]) {
            alMenosIgual = false;
        }
        if (pj.coordenadas[a] < pi.coordenadas[a]) {
            alMenosMayor = true;
        }
    }
    if(alMenosIgual == alMenosMayor == true){
        return true;
    }else{
        return false;
    }

}

vector<Punto> paretoBasico(vector<Punto> L) {
    vector<Punto> puntosFrontera;

    for (int i = 0; i < L.size(); i++) {
        bool esNoDominado = true;
        for (int j = 0; j < L.size(); j++) {
            if (i != j && esDominante(L[j], L[i], K)) {
                esNoDominado = false;
                break;
            }
        }
        if (esNoDominado) {
            puntosFrontera.push_back(L[i]);
        }
    }

    return puntosFrontera;
}
```