Universidad Politécnica de Cartagena

PLANIFICACIÓN Y GESTIÓN DE REDES

Grado en Ingeniería Telemática
Curso 2013-2014

Práctica 6. Algoritmos tipo gradiente para problema de asignación de ancho de banda

Autor:

Pablo Pavón Mariño

1. Objetivos

Los objetivos de esta práctica son:

1. Desarrollar en `net2plan` distintas variantes del algoritmo del gradiente para un problema de asignación de ancho de banda, comparando sus prestaciones de convergencia.
2. Aprender a valorar la posible implementación distibuida de un algoritmo.

2. Duración

Esta práctica tiene una duración de 1 sesión, cumpliendo un total de 3 horas de laboratorio.

3. Evaluación

Los alumnos no tienen que entregar ningún material al finalizar esta práctica. Este boletín es para el estudio del alumno. En él, el alumno deberá resolver los problemas planteados y anotar las aclaraciones que estime oportunas para su posterior repaso en casa.

4. Documentación empleada

La información necesaria para resolver esta práctica se encuentra en:

- Ayuda de la herramienta `net2plan` (`http://www.net2plan.com/`).
- Instrucciones básicas presentes en este enunciado.
- Apuntes de la asignatura.

5. Problema de asignación de ancho de banda

Sea un conjunto de nodos N, un conjunto de enlaces E y un conjunto de demandas con tráfico ofrecido D. Para cada demanda $d \in D$, el tráfico se encamina por un camino p_d conocido, que será el camino con la menor distancia en km entre el nodo origen y destino de la demanda. El objetivo es encontrar para cada demanda d, la cantidad de tráfico h_d a inyectar en la red, resolviendo el problema de optimización:

$$\min_{h_d} \sum_e F_e(y_e) - \sum_d \log h_d \tag{1a}$$

$$h^{min} \leq h_d \quad \forall d \in D \tag{1b}$$

La función objetivo computa un coste para cada enlace, dado por la función F_e que depende del tráfico y_e que circula por e. Por tanto, depende de las variables h_d, para todas aquellas demandas d que

atraviesen ese enlace. Nos centraremos en la función F_e dada por un valor proporcional a la estimación M/M/1 del retardo cuando la utilización del enlace es hasta el 99 %, y su proyección lineal cuando la utilización excede este valor. La función F_e buscada es:

$$F_e(y_e) = \begin{cases} \frac{M_e}{u_e - y_e} \text{ si } y_e \leq 0{,}99u_e \\ \frac{100M_e}{u_e} + \frac{10^4 M}{u_e^2}(y_e - 0{,}99u_e) \text{ si } y_e > 0{,}99u_e \end{cases}$$

Esta función tiene como derivada respecto a y_e:

$$F'_e(y_e) = \begin{cases} \frac{M_e}{(u_e - y_e)^2} \text{ si } y_e \leq 0{,}99u_e \\ \frac{10^4 M_e}{u_e^2} \text{ si } y_e > 0{,}99u_e \end{cases}$$

Y como derivada segunda respecto a y_e:

$$F''_e(y_e) = \begin{cases} \frac{2M_e}{(u_e - y_e)^3} \text{ si } y_e \leq 0{,}99u_e \\ 0 \text{ si } y_e > 0{,}99u_e \end{cases}$$

La función objetivo f es diferenciable. La coordenada del gradiente asociada a la variable de decisión h_d es:

$$\frac{\partial f}{\partial h_d} = \sum_{e \in p_d} F'_e(y_e) - \frac{1}{h_d}$$

Los valores de la diagonal de la matriz hessiana asociados a cada variable h_d son:

$$H_{dd} = \frac{\partial^2 f}{\partial h_d^2} = \sum_{e \in p_d} F''_e(y_e) + \frac{1}{h_d^2}$$

Los factores M_e se calculan de tal manera que si una demanda d atraviesa un enlace e que está al 99 % de utilización, no tenderá a incrementar su tasa h_d en ningún caso. Esto se cumplirá cuando se garantice que:

$$\frac{\partial f}{\partial h_d} \geq 0 \quad \forall h_d \geq h^{min} \Rightarrow$$
$$\Rightarrow \frac{10^4 M_e}{u_e^2} - \frac{1}{h^{min}} \geq 0 \Rightarrow M_e \geq \frac{u_e^2}{10^4 h^{min}}$$

6. Implementación del algoritmo

Las características del algoritmo pedido son:

- El algoritmo deberá implementarse en una clase de nombre `FBA_projectedGradient.java`. El algoritmo recibirá como entrada una topología de nodos y enlaces. Creará una demanda para cada par de nodos, con una única ruta asignada a ella que será la dada por el camino más corto en km o número de saltos (en función del parámetro `shortestPathType`) entre los nodos origen y destino de la demanda. La cantidad de tráfico a cursar por cada demanda, será la dada por un algoritmo tipo gradiente proyectado según lo explicado en la sección anterior.

- El parámetro interno del algoritmo h^{min} debe hacerse igual a: $h^{min} = \frac{\text{mín}_e\, u_e}{|D|}$, donde $\text{mín}_e\, u_e$ es la menor capacidad entre los enlaces de la red, y $|D|$ es la cantidad de total de demandas. De esta manera se garantiza que la solución en la que todas las demandas inyectan la cantidad mínima de tráfico h^{min}, es una solución en la que no se satura ningún enlace.

- Los parámetros de entrada definidos por el usuario son:

  - `maxNumIterations`: Número máximo de iteraciones a ejecutar del algoritmo. El algoritmo debe finalizar tras este número de iteraciones, devolviendo la mejor solución encontrada. El valor por defecto es `maxNumIterations = 1000`.
  - `shortestPathType`: Forma de calcular el camino más corto para cada demanda: "km" (distancia en km del camino) o "hops", número de enlaces atravesados. El valor por defecto es `shortestPathType = km`.
  - `beta`: Factor constante que multiplica al vector gradiente en cada iteración del algoritmo. El valor por defecto es `beta = 1`.
  - `diminishingSteps`: "yes" si el salto en cada iteración debe multiplicarse por el factor $1/k$, donde k es el número de la iteración. "no" en caso contrario. El valor por defecto es `diminishingSteps = ''no''`[1].
  - `diagonalScaling`: "yes" si el salto en cada iteración debe multiplicarse por el factor $1/H_{dd}$. "no" en caso contrario. El valor por defecto es `diagonalScaling = ''no''`.

  La siguiente tabla muestra el esquema del algoritmo a implementar en función de los parámetros de entrada:

Esquemas $k = 1, 2, \ldots, maxNumIterations$

`diminishingSteps`	`diagonalScaling`	Esquema
no	no	$h_d^{k+1} = \left[h_d^k - \beta s(f)(h_d)\right]_{h^{min}}$
no	yes	$h_d^{k+1} = \left[h_d^k - \frac{\beta}{H_{dd}} s(f)(h_d)\right]_{h^{min}}$
yes	no	$h_d^{k+1} = \left[h_d^k - \frac{\beta}{k} s(f)(h_d)\right]_{h^{min}}$
yes	yes	$h_d^{k+1} = \left[h_d^k - \frac{\beta}{kH_{dd}} s(f)(h_d)\right]_{h^{min}}$

6.1. Ayudas a la implementación

Se proporciona al alumno la clase `FBA_projectedGradient_template.java` que servirá como plantilla. Esta clase ya implementa una buena parte del código que recoge los parámetros de entrada y graba en el objeto `netPlan` la salida del algoritmo.

7. Análisis

Responda a las siguientes preguntas:

- Indique cómo realizaría una implementación distribuida del algoritmo para el caso de paso constante (sin escalado, ni reducción de tamaño del paso). Describa un mecanismo por el cada demanda podría iterar de manera independiente a las demás, utilizando únicamente información local e información señalizada por los enlaces atravesados. ¿Se le ocurre alguna manera en la que el algoritmos podría trabajar sin señalización *explícita* de los enlaces? Es decir, un mecanismo

[1] Nótese que el esquema

por el cual cada demanda pudiese estimar por medios propios el efecto en el gradiente de las penalizaciones a la función de coste impuestas por los enlaces.

- Rellene la siguiente tabla indicando la solución alcanzada por el algoritmo después de 1000 y 10000 iteraciones, para la topología `NSFNET`, tomando el resto de valores por defecto:

Resultados para topología `NSFNet_N14_E42.n2p`

`diminishingSteps`	`diagonalScaling`	Coste 1000 it	Coste 10000 it
no	no		
no	yes		
yes	no		
yes	yes		

- Haga pruebas para analizar empíricamente la convergencia para otras redes, otros valores `beta` etc. (por ejemplo, valores `beta` más altos, cuando se utiliza una reducción del paso en cada iteración).

8. Posibles variaciones y análisis (opcional)

Se sugieren las siguientes variaciones y análisis:

- Modifique el problema de tal manera que las funciones F_e sean igual a:

$$F_e = \begin{cases} 0 \text{ si } y_e \leq u_e \\ (y_e - u_e)M \text{ en caso contrario} \end{cases}$$

Es decir, esta función no añade ningún coste si el tráfico en un enlace es inferior a su capacidad, y en caso contrario añade una penalización de M unidades de coste, por cada unidad de tráfico que excede la capacidad. Esta función tiene derivada primera igual a 0 en todos los puntos $y_e < u_e$, igual a M en los puntos $y_e > u_e$, y cuando $y_e = u_e$ la función no es diferenciable, y cualquier valor entre 0 y M que elijamos será un subgradiente. El factor $M_e = M$ se calculará de tal manera que si una demanda d atraviesa un enlace e que está saturado, no tenderá a incrementar su tasa h_d. Por ejemplo, esto se cumple con:

$$M = \frac{1}{h^{min}}$$

- Implemente variaciones del tipo "bola pesada" (*heavy ball*) donde la dirección de salto en cada instante se calcula a partir del (sub)gradiente actual, y el (sub)gradiente en la iteración anterior.