# CC4102 - Diseño y Análisis de Algoritmos
# Auxiliar 7

Prof. Gonzalo Navarro; Aux. Mauricio Quezada

26 de Diciembre, 2011

## 1 El problema de los $k$ servidores

Considere el escenario donde tiene $k$ puntos (*servidores*) en un *espacio métrico* (donde está definida una función de distancia $d$: simétrica, no-negativa y que cumple la desigualdad triangular) y una secuencia de puntos (*peticiones*) que debe atender. Cada vez que llega una petición, un servidor debe moverse hacia esa posición.

El problema *online* consiste en minimizar la distancia recorrida por todos los servidores luego de $n$ peticiones, sin saber la secuencia de puntos a atender.

Recuerde que un algoritmo *online* $ALG$ es $r$-competitivo si existe una constante $a$ tal que para cualquier instancia $I$ y el algoritmo óptimo $OPT$,

$$\text{cost}_{ALG}(I) \leq r \cdot \text{cost}_{OPT}(I) + a$$

Donde $r$ es el *radio competitivo*.

1. Sea $ALG$ un algoritmo online para el problema de los $k$ servidores bajo un espacio métrico arbitrario con al menos $k+1$ puntos. Pruebe que el radio competitivo de $ALG$ es al menos $k$.

2. Para el siguiente análisis competitivo, necesitamos usar una conocida herramienta, la función potencial. *Una función $\Phi$ es una función de potencial que demuestra un radio competitivo $r$ de un algoritmo $ALG$ si satisface las siguientes condiciones:*

   - $\Phi$ es nonegativa
   - Cada respuesta a una petición a $OPT$ incrementa $\Phi$ no más de $r$ veces el costo cargado a $OPT$ por esa respuesta.
   - Cada respuesta a una petición a $ALG$ disminuye el potencial por al menos el costo cargado a $ALG$ por esa respuesta.

Por lo que, un algoritmo es $r$-competitivo, si existe una función de potencial $\Phi$ para $r > 0$ que cumple las propiedades anteriores.

Considere el problema de $k$ servidores en una linea (un espacio de dimensión 1) y el siguiente algoritmo:

- Si todos los servidores están a un lado de la petición, entonces envía el servidor más cercano a ella.

- Si una petición se encuentra entre dos servidores, envía los dos servidores a velocidad constante, y se detienen cuando uno de ellos llega a su objetivo.

Finalmente, de una función de potencial $\Phi$ que demuestre un radio competitivo de $k$ para este algoritmo.

## 1.1 Solución

### 1.1.1 Parte I

Suponga, sin pérdida de generalidad, que $ALG$ mueve sólo un servidor tras cada petición y que cada uno de los $k$ servidores comienzan en lugares distintos. Escoja un subespacio del espacio métrico con las $k$ posiciones más una arbitraria. Sea $d_{ij}$ la distancia entre los puntos $i$ y $j$.

Mostraremos cómo un adversario puede escoger una secuencia de peticiones $\sigma_n$ tal que el costo de $ALG$ sea al menos $k$ veces el costo de un algoritmo óptimo. La elección es simple, el adversario escogerá el siguiente punto como uno no cubierto por $ALG$. El costo total de $ALG$ es entonces:

$$\text{cost}_{ALG}(\sigma_n) = \sum_{i=1}^{n} d(\sigma(i+1), \sigma(i))$$

Donde $\sigma(i)$ es la petición $i$-ésima. Como sabemos que en la petición $(i+1)$-ésima el adversario escogerá la ubicación vaciada por $ALG$ en la petición $i$-ésima, por lo que el movimiento será de $\sigma(i+1)$ a $\sigma(i)$.

Para mostrar que $ALG$ es $k$-competitivo, mostraremos que existe un algoritmo que puede servir la secuencia por $1/k$ del costo.

Para esto, considere $k$ algoritmos que se comportan de la misma forma y que sólo difieren en dónde comienzan los servidores de cada uno. Existen $\binom{k+1}{k} = k+1$ formas de escoger las $k$ posiciones iniciales. De ellas, $k$ tendrán un servidor en $\sigma(1)$. Esas $k$ posibilidades serán las posiciones iniciales de los $k$ algoritmos.

Si el adversario ya tiene un servidor en la ubicación pedida, entonces no hará nada. Si el request $\sigma(n)$ no está ocupado por alguno de los servidores del adversario, lo moverá desde la ubicación $\sigma(n-1)$. Note que tras cada petición, cada uno de los $k$ algoritmos tendrá distintas *configuraciones* de servidores (el conjunto de las ubicaciones de éstos).

Al comienzo está claro que las configuraciones son distintas. Asuma que es cierto hasta la petición $\sigma(n-1)$ y comparemos dos algoritmos. Antes de la petición $\sigma(n)$ tendrán configuraciones distintas.

- Si ambos tienen un servidor en esa posición, no harán nada y tendrán configuraciones distintas.
- Si ninguno tiene un servidor ahí, ambos moverán el servidor en $\sigma(n-1)$ hacia $\sigma(n)$ y seguirán teniendo configuraciones distintas.
- Si uno de los dos tiene un servidor ahí, pero el otro no, entonces éste moverá su servidor de $\sigma(n-1)$ a $\sigma(n)$. El otro dejará su servidor donde está, por lo que tendrán configuraciones distintas.

Con esto, la idea es ver el costo total de los $k$ algoritmos si los ejecutáramos simultáneamente. No hay costo en la primera petición pues todos ellos tendrán ya un servidor en $\sigma(1)$. Luego de $n-1$ peticiones, cada uno de los $k$ algoritmos tendrá un servidor en $\sigma(n-1)$. Como cada algoritmo

tiene sus servidores en distintas configuraciones, y como ningún algoritmo tiene dos servidores en la misma posición al mismo tiempo, tenemso que todos menos un algoritmo tendrá un servidor en $\sigma(n)$. El costo en ese caso será de $d(\sigma(n-1), \sigma(n))$. Por lo tanto, el costo total de correr los $k$ algoritmos sobre la secuencia $\sigma_n$ es

$$\sum_{i=2}^{n} d(\sigma(i-1), \sigma(i))$$

Como el costo de los $k$ algoritmos no es mayor que el de $ALG$, entonces al menos uno de ellos no tendrá costo mayor que $\text{cost}_{ALG}(\sigma_n)/k$, por lo que el radio competitivo es $k$.

### 1.1.2 Parte II

Sean $a_1, \ldots, a_k$ los servidores del adversario y $s_1, \ldots, s_k$ los de nuestro algoritmo. Usaremos tanto $a_j$ como $s_j$ para denotar la posición de cada servidor en la recta real (por lo que habrá que renombrarlos cada vez que se sobrepasen).

Definamos $\Phi$ como

$$\Phi = \Psi + \Theta$$

donde

$$\Psi = k \sum_{i} |a_i - s_i|$$

y

$$\Theta = \sum_{i<j} |s_i - s_j|$$

Asumiremos, sin pérdida de generalidad, que el adversario no mueve más de un servidor por petición. Consideremos los distintos casos que pueden ocurrir:

- El adversario puede mover un servidor $a_i$ una distancia $d$ incurriendo un costo $d$. Si se acerca al servidor $s_i$, entonces $\Psi$ disminuirá en $kd$. Si se aleja de $s_i$, $\Psi$ aumentará $kd$. En cualquier caso, $\Delta\Psi \leq kd$ y por lo tanto $\Delta\Phi \leq kd$.

- Cuando nuestro algoritmo mueve un servidor, tenemos dos casos:

  - Cuando mueve uno solo, puede ser $s_1$ o $s_k$. Asumamos que mueve $s_1$ a la izquierda una distancia $d$. Como el adversario ya respondió a esa solicitud, tendrá a $a_1$ a la izquierda de $s_1$, por lo que $\Phi$ disminuye exactamente $d$, el costo nuestro algoritmo, ya que $\Psi$ disminuirá en $kd$ y $\Theta$ aumentará en $(k-1)d$.
  - En el caso en el que mueve dos servidores hacia una petición, supongamos que mueve a $s_i$ y a $s_{i+1}$ y que $a_j$ es un servidor del adversario que ya está en la posición pedida. Si $j \leq i$, entonces $s_i$ que se mueve hacia $a_j$ también se mueve hacia $a_i$, en este caso, $\Psi$ no aumenta debido a que cada incremento causado por el movimiento de $s_{i+1}$ es cancelado por el movimiento de $s_i$. Si $j \geq i+1$, entonces los roles de $s_i$ y $s_{i+1}$ se intercambian y tenemos que $\Psi$ no aumenta.

Falta mostrar que $\Theta$ disminuirá lo suficiente para pagar el movimiento de los dos servidores. Dado que cada servidor $s_j$, $j \neq i, i+1$ está o a la izquierda de ambos o a la derecha de ambos, tenemos que uno de los servidores se mueve hacia $s_j$ y el otro se aleja. Luego la suma $|s_j - s_i| + |s_j - s_{i+1}|$ se mantiene constante durante la fase. El único término en $\Theta$ que no hemos contado es $|s_i - s_{i+1}|$. Este término disminuirá exactamente el costo del movimiento durante esta fase.

# 2 Buscando la casa

De repente, usted despierta en medio de una carretera desierta y no sabe dónde está. Además tiene sed y dolor de cabeza, y no recuerda cómo llegó ahí. Lo único que sabe es que su casa está en algún lugar de la carretera, pero no sabe en cuál dirección ir. Además, debido al sol y al dolor de cabeza no puede ver bien y sólo se daría cuenta de que está en el lugar correcto una vez estando ahí.

Para facilitarle las cosas, suponga que la carretera es una recta infinita y que usted está en $x = 0$ y que su casa está en $x^*$, con $|x^*| > 1$.

1. De un algoritmo determinista para encontrar el punto $x^*$ que sea 9-competitivo.

2. De un algoritmo aleatorizado basado en su estrategia anterior, y calcule el radio competitivo.

## 2.1 Solucion

### 2.1.1 Parte I

Considere la estrategia $A_m$ de de buscar la casa en fases. Las fases comienzan con $i = 0$, comenzando en el origen, y caminar una distancia $(-m)^i$ y luego volver al origen, deteniéndose antes si se llega al objetivo. En cada fase en la cual no se encuentre el objetivo, se habrá recorrido una distancia $2m^i$.

Dividamos las posibles ubicaciones en regiones $1, 2, \ldots, k$ donde la región $k$ satisface $m^{k-1} < |x^*| \leq m^k$. Suponga que $|x^*|$ cae en la región $k$. Entonces no habrá encontrado el punto sino hasta la fase $k$. Si tenemos suerte, entonces $-(-m)^{k-1} < x^* \leq (-m)^k$ y encuentra la casa en la fase $k$, por lo que la distancia recorrida es

$$A_m(x^*) = \sum_{i=0}^{k-1} 2m^i + |x^*| = 2\frac{m^k - 1}{m - 1} + |x^*|$$

Si no tenemos suerte, entonces no encontraremos la casa sino hasta la fase $k + 1$, por lo que

$$A_m(x^*) = \sum_{i=0}^{k} 2m^i + |x^*| = 2\frac{m^{k+1} - 1}{m - 1} + |x^*|$$

Como el adversario tratará de maximizar el costo, pondrá la casa en el lado desafortunado, por lo que el radio será

$$r = \frac{2}{|x^*|}\left(\frac{m^{k+1} - 1}{m - 1}\right) + 1$$

Como asumimos que $|x^*|$ cae en la región $k$, el peor radio para la región $k$ ocurre si el adversario pone la casa a distancia $m^k + \epsilon$ del origen, por lo tanto,

$$r < 2\Big(\frac{m^2 - \frac{1}{m^{k-1}}}{m-1}\Big) + 1$$

Cuando $k$ tiende a infinito, el radio entonces está dado por

$$r = 2\Big(\frac{m^2}{m-1}\Big) + 1$$

Derivando e igualando a 0,

$$\frac{dr}{dm} = 0 = 2\frac{x(x-2)}{(x-1)^2}$$

Tenemos que cuando $m = 2$, $r = 9$, por lo que la estrategia de duplicar la distancia recorrida es 9-competitiva.

### 2.1.2 Parte II

Como el adversario puede escoger a priori la dirección en la cual estará la casa para aumentar nuestro costo, podemos evitarlo escogiendo al azar al comienzo hacia qué dirección ir[1].

Al comienzo, lanzamos una moneda equilibrada: si sale cara, en cada fase $i$ caminamos desde 0 hasta $(-m)^i$ y volvemos. Si sale sello, entonces en cada fase $i$ caminamos desde 0 hasta $-(-m)^i$ y volvemos. Usando el análisis anterior y tomando la esperanza[2], tenemos:

$$\begin{aligned} r &= \frac{1}{2}\Big(\frac{2}{|x^*|}\Big(\frac{m^k - 1}{m-1}\Big) + 1\Big) + \frac{1}{2}\Big(\frac{2}{|x^*|}\Big(\frac{m^{k+1} - 1}{m-1}\Big) + 1\Big) \\ &= 1 + \frac{1}{|x^*|}\Big(\frac{m^k + 1 + m^{k-1} - 2}{m-1}\Big) \end{aligned}$$

Escogiendo $|x^*| = m^k + \epsilon$ y haciendo tender $k$ a infinito, tenemos

$$r < 1 + \frac{m^2 + m}{m-1}$$

Si escogemos $m = 2$, el radio competitivo es entonces $r = 7$. Si derivamos e igualamos a 0, tenemos que cuando $m = 1 + \sqrt{2} \approx 2.414$, el radio competitivo es $r = 4 + 2\sqrt{2} \approx 6.8284$.

# 3 Footnotes

[1] En ciertos modelos, el adversario **sí** puede saber las elecciones aleatorias del algoritmo. Para este caso, consideraremos un *oblivious adversary* que no conoce las elecciones aleatorias y sólo puede fijar el input.

[2] En este caso, un algoritmo aleatorizado es $r$-competitivo si el valor esperado del costo del algoritmo es menor o igual que $r$ veces el costo del óptimo más una constante.