ELSEVIER

Theoretical Computer Science 261 (2001) 213–226

Theoretical Computer Science

www.elsevier.com/locate/tcs

Dynamic location problems with limited look-ahead

Fan Chung[a,*,1], Ronald Graham[a,b]

[a] *Department of Mathematics & Computer Science, University of California, San Diego, CA 92093, USA*

[b] *AT&T Labs, Florham Park, NJ, USA*

Accepted 14 March 2000

1. Introduction

The following problem arose in connection with studies of Internet web page caching. The general setting is as follows:

In some fixed metric space M, k "servers" $S_1,\ldots,S_k$ are given with some arbitrary initial locations in M. Requests for service at certain points $\sigma_1,\sigma_2,\sigma_3,\ldots,\sigma_N$, in M arrive over time. Immediately, after request σ_t is received, exactly one of several mutually exclusive actions must be taken:

(i) Some server is moved to σ_t, with a resulting cost of $c(\sigma_t)$, the "cost" of the point σ_t.

(ii) No server moves. In this case, the cost for "no service" is defined to be $\min_k d(S_k, \sigma_t)$, where $d(x,y)$ denotes the distance between x and y in M.

A further feature of our model is that two parameters $u,w\geqslant 0$ are specified, which are used as follows. Before having to decide how to service request σ_t, the servers have at their disposal the knowledge of the $u+w$ requests σ_i with $t-u\leqslant i\leqslant t+w-1$. Thus, the servers can only "remember" or store the past u requests $\sigma_{t-u},\sigma_{t-u+1},\ldots,\sigma_{t-1}$ but are allowed to know the w future requests $\sigma_t,\sigma_{t+1},\ldots,\sigma_{t+w-1}$ before having to service σ_t.

* Corresponding author.
E-mail address: fan@euclid.ucsd.edu (F. Chung).

[1] Supported in part by NSF Grant No. DMS 98-01446 and Bell Communications Research, Morristown, NJ, USA.

PII: S0304-3975(00)00140-7

The rules which govern the choices made for servicing all the σ_t define some algorithm A. In this model, A is deterministic and can only depend on the values of the σ_i which it currently knows, and nothing else. In particular, A is not allowed to make probabilistic choices based on some source of randomness. We denote by $A(\sigma)$, the cost of servicing the request sequence $\sigma = (\sigma_1, \ldots, \sigma_N)$.

Of course, if we are allowed to know *all* the σ_t before having to act, it is very likely the cost of servicing σ can be decreased. Let us denote by $\mathrm{OFF}(\sigma)$ the minimum possible cost of servicing σ in this case. Thus, $\mathrm{OFF}(\sigma)$ is the cost for an optimal *off-line* algorithm for σ.

Our main interest in this paper is to investigate the effect of being able to look ahead (as well as utilizing the past history) on the performance of such algorithms.

For a given request sequence, let $\mathrm{OPT}^{(u,w)}(\sigma)$ denote the minimum cost of any (u,w)-window algorithm servicing σ, and we define

$$\rho(u,w) := \inf_{\sigma} \frac{\mathrm{OPT}^{(u,w)}(\sigma)}{\mathrm{OFF}(\sigma)}$$

over all σ with $\mathrm{OFF}(\sigma) \to \infty$ to avoid certain inessential degeneracies. (In our model, we are assuming our "adversary", i.e., the source generating the request sequence σ, is all-powerful. That is, the adversary knows everything about the servers' strategy and can predict everything they will do.) The basic problem then is to understand how $\rho(u,w)$ is affected by the various parameters in our model, i.e., the metric space (M,d), and the choice of the cost function c.

This model can be further extended by allowing "excursions", i.e., where servers are allowed to go to arbitrary points in M in response to a request σ_t. In other words, we can replace (i) by

(i′) Some server, say S_i, is moved to some vertex v, with a resulting cost of $c(v) + d(S_k, v) + d(v, \sigma_t)$.

We point out that a fair amount is known when option (ii) is not allowed, $w = 1$, and the cost function C depends on $d(S_i, \sigma_t)$ and not just σ_t. In other words, some server must always be moved to the current request, but the cost only depends on the distance moved (see [2] or [6] for a survey). In fact, we show here that for this model, finite look-ahead does not help asymptotically. However, for the model of allowing excursions, finite values of w do make a difference. (We discuss this in Sections 4 and 5.)

In this paper, we will examine several special cases of our general problem. At the end, we will formulate a number of open questions which we feel are fundamental and interesting for these studies, and which (unfortunately) illustrate how incomplete our knowledge is here at this point in time.

2. The value of temporal information

In this section we consider the problem of determining $\rho(u,w)$ for $k = 1$ server, for given u and w allowing actions (i) and (ii). We first consider several small cases for a

Table 1
Values of $\rho(u,w)$ for G_0 with $k=1$ server

$u\backslash w$	0	1	2	3	4
0	*	4	2	2	$\frac{3}{2}$
1	6	3	2		
2	4	$\frac{8}{3}$			
3	$\frac{10}{3}$				
4	3				

rather simple graph G_0 on two vertices, 0 and 1, with $d(0,1)=1$ and $c(0)=c(1)=2$. Indeed, even for this very special case, the problem of determining $\rho(u,w)$ for small values of u and w is still quite intriguing. This hints at the difficulty in treating the problem of determining $\rho(u,w)$ for arbitrary u and w (Table 1.

The proofs for these cases are not too difficult and involve case analysis. We will give the proofs for two cases. The remaining ones can be proved in similar ways and proofs will be omitted.

Theorem 1. *For the graph G_0 with two vertices 0 and 1 with $d(0,1)=1$ and $c(0)=c(1)=2$, we have $\rho(0,2)=2$.*

Proof. Let $S(t)$ denote the location of the server after moving at time t. We assume that $S(0)=0$. First, we consider the following algorithm A. The server at x is moved to $\bar{x}=1-x$ if and only if the window shows $\bar{x}\bar{x}$. (Otherwise, it does not change.) We claim that for any request sequence σ, we have $A(\sigma)\leqslant 2\cdot \mathrm{OFF}(\sigma)$. To see this, we dynamically partition σ into blocks of the form $\sigma_{t+1}\sigma_{t+2}\ldots\sigma_{t+j}=\bar{x}\overbrace{x\ldots x}^{j-1}$, for $j\geqslant 2$, where $S(t)=x$. There are two possibly exceptional blocks: the initial block, which may have the form $\overbrace{0\ldots 0}^{j'}\overbrace{11\ldots 1}^{j-1}$, $j'\neq 1$ is possible, and the last block, which may have the form $\bar{x}x\bar{x}$. Then, for each block OFF pays at least 1 while A pays at most 2. Thus, we have $A(\sigma)\leqslant 2\cdot\mathrm{OFF}(\sigma)$, as claimed. In the other direction, for any algorithm B, we consider the the sequence ϕ satisfying $\phi_{t+2}=\bar{x}$ where $S(t)=x$ (of course, S depends on B). It is not difficult to show that $B(\phi)\geqslant 2\cdot\mathrm{OFF}(\phi)$, and so, $\rho(0,2)=2$. □

Theorem 2. *For the graph G_0 above, we have $\rho(2,0)=4$.*

Proof. We will use the same algorithm A as above, namely, the server at x moves to $\bar{x}$ if and only if the window (now into the past) shows $\bar{x}\bar{x}$. We first claim that for any request sequence σ, we have $A(\sigma)\leqslant 4\,\mathrm{OFF}(\sigma)$. To see this, we partition σ into blocks of the form $x_1x_2\ldots x_r11y_1y_2\ldots y_s00$ where the 11 is the first occurrence of 00 in the block of x's, and the terminal 00 is the first occurrence of 00 in the y's. Thus, every $x_i=1$ has $x_{i+1}=0$, and every $y_j=0$ has $y_{j+1}=1$. Suppose there are m such $x_i=1$ and m' such $y_j=0$. Then it is not hard to see that A pays at most a cost of $m+m'+8$ for

this block while OFF must pay at least $m+m'+2$. This then implies $A(\sigma)\leqslant 4\,\mathrm{OFF}(\sigma)$. In the other direction, for any algorithm B, we will use the request sequence λ defined by taking $\lambda_t=\bar{x}$ where $S(t-1)=x$. (This is possible because of the strength of the adversary.) An easy argument shows $B(\lambda)\geqslant 4\,\mathrm{OFF}(\phi)$ and the proof is complete. □

The above theorem implies that a window into the past *does* matter. In the remainder of this section, we will give upper and lower bounds for $\rho(0,w)$ in terms of w.

Theorem 3. *In a graph G, suppose that the cost of moving to v is $c(v)=\alpha\geqslant 2$ for all v. For given w with $w\geqslant 2\alpha$, there is a request sequence σ such that any $(0,w)$ algorithm A pays at least $1+2/(w+2)$ times the cost OFF(σ).*

Proof. Suppose the graph contains one edge with two endpoints, 0 and 1. Without loss of generality, we assume that the server S is at 0. To simplify the argument, we assume $w-2\alpha=2a$, for an integer $a\geqslant 0$. (The other cases can be done in a similar way.) Furthermore, we assume that the request sequence shown in the window of length w has the form $\overbrace{11\ldots1}^{2\alpha-2}\overbrace{0101\ldots01}^{2a+2}$. To proceed, there are two possibilities.

Case 1: The server stays at 0. In this case, the request sequence will then have a block of 1's of length w at the end of the window. We note that for these $2w$ requests, the ratio of the cost of the algorithm to that of the optimum is off by a factor $1+2/(w+2)$.

Case 2: The server moves to 1. In this case, the request sequence will then have a block of 0's, say of length w, at the end of the window. Then for these $2w$ requests, the ratio of the cost to the algorithm and to the optimum is again off by a factor of a least $1+2/(w+2)$. □

Theorem 4. *In a graph G with vertex set $\{0,1\}$, suppose that the cost of moving to v is $c(v)=\alpha\geqslant 2$ for all v. For given w with $w\geqslant 2\alpha$, there is a $(0,w)$-window algorithm A such that for any request sequence σ algorithm A pays no more than $1+4\alpha^2/w$ times the cost OFF(σ).*

Proof. We consider the following algorithm A. For a window $\sigma_{i+1}\ldots\sigma_{i+w}$ and $S(i)=x$, let j denote the least integer (if it exists) $j\leqslant w$ such that the number p of $\bar{x}$'s among $\sigma_{i+1}\ldots\sigma_{i+j}$ and the number q of x's among $\sigma_{i+1}\ldots\sigma_{i+j}$ satisfies

$$|p-q|\geqslant 2\alpha+2.$$

If such a j exists, the server is moved to the majority symbol y. Otherwise, the server does not move. It is not hard to show that Algorithm A will stay at y throughout the next j requests. We will call this a block of type I (of length j). If such a j does not exist, algorithm A will use the locally optimal strategy for the next w moves and we call this a block of type II (of length w). To upper bound the cost of Algorithm A, we partition the request sequence into blocks of types I and II. So we

have some consecutive blocks of type I followed by a block of type II, and so on. For consecutive blocks of type I, the cost of algorithm A is at most the optimum plus α (which corresponds to the possible cost of initialization). For each block of type II, the cost for algorithm A is at most the optimum plus 2α (which corresponds to both initialization and ending). For a block of type II, the optimum has cost at least $w/(2\alpha)$ since there is at least one alternation among 2α consecutive requests. So the ratio of the cost for algorithm A and the optimum is at most

$$1+\frac{2\alpha}{w/(2\alpha)}. \qquad \square$$

3. More on limited look-ahead

For a given request sequence σ, let $\overline{\mathrm{OPT}}_k^{(w)}(\sigma)$ denote the minimum possible cost for servicing σ with k servers over all $(0,w)$-window algorithms for which possibility (ii) is prohibited, i.e., every request must be visited by some server. Clearly, we have

$$\rho_k=\rho_k^{[1]}\geqslant\rho_k^{[2]}\geqslant\cdots\geqslant\rho_k^{[w]}\geqslant\cdots\ . \tag{1}$$

The first question which occurs is whether having limited look-ahead actually makes any difference. That is, are any of the inequalities in (1) strict? Unfortunately, the answer is no.

Fact 1. *For all* $w<\infty$,

$$\rho_k^{[w]}=\rho_k.$$

Proof. We need to show that $\rho_k^{[w]}=\rho_k$. Let $A^{[w]}$ be any on-line algorithm with look-ahead w, and let $\sigma=(\sigma_1,\sigma_2,\ldots)$ be any request sequence. Define

$$\sigma^{[w]}=\left(\overbrace{\sigma_1,\ldots,\sigma_1}^{w},\overbrace{\sigma_2,\ldots,\sigma_2}^{w},\overbrace{\sigma_3,\ldots,\sigma_3}^{w},\ldots\right),$$

where each request in σ is repeated w times in $\sigma^{[w]}$. What is $A^{[w]}(\sigma^{[w]})$?

Using the observation that in any of these problems, we can always restrict our attention to algorithms which do nothing if a server is already located at the next request, it is easy to see that the algorithm $A^{[w]}$ acting on $\sigma^{[w]}$ induces a (normal) on-line algorithm A acting on σ, namely if $A^{[w]}$ selects s_i to move to the first σ_r in the sequence $\overbrace{\sigma_r\sigma_r\ldots\sigma_r}^{w}$ in $\sigma^{[w]}$, then A selects s_i to move to σ_r in σ (see Fig. 1).

In this figure, the (s_i,σ_r) entry of the table indicates that server s_i is selected to serve request σ_r, etc.

A simple computation now shows that

$$A^{[w]}(\sigma^{[w]})=A(\sigma),\quad \mathrm{OFF}(\sigma^{[w]})=\mathrm{OFF}(\sigma).$$

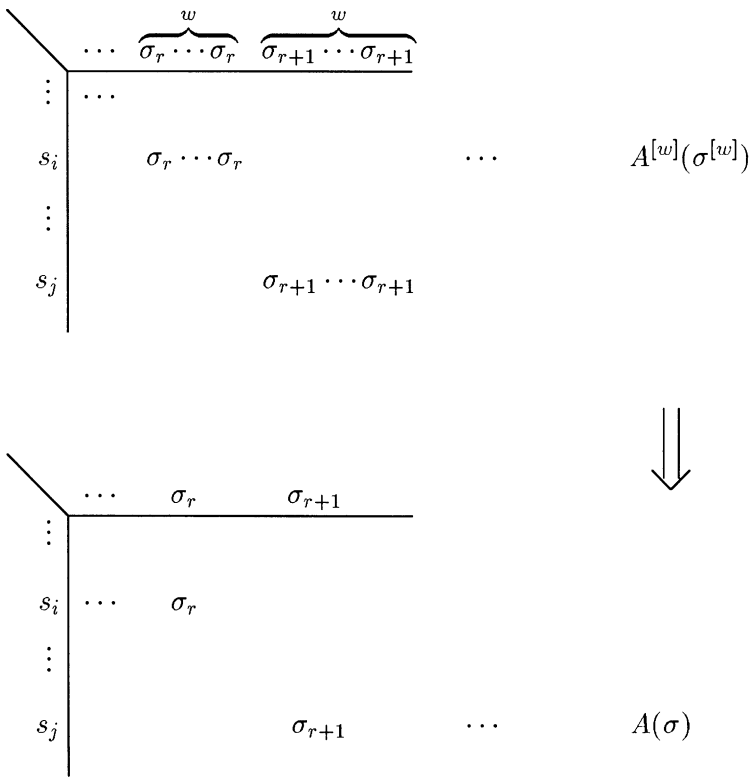


Fig. 1.

Therefore,

$$\begin{aligned}\rho_k^{[w]} &= \inf_{A^{[w]}} \limsup_{\mathrm{OFF}(\sigma)\to\infty} \frac{A^{[w]}(\sigma)}{\mathrm{OFF}(\sigma)} \\ &\geqslant \inf_{A^{[w]}} \limsup_{\mathrm{OFF}(\sigma^{[w]})\to\infty} \frac{A^{[w]}(\sigma^{[w]})}{\mathrm{OFF}(\sigma^{[w]})} \\ &= \inf_{A} \limsup_{\mathrm{OFF}(\sigma)\to\infty} \frac{A(\sigma)}{\mathrm{OFF}(\sigma)} = \rho_k, \end{aligned} \tag{2}$$

where the inequality in (5) comes from the fact that the sequences of the form $\sigma^{[w]}$ are a subset of the set of all possible request sequences which $A^{[w]}$ must deal with.

□

This would be a natural (though disappointing) place to conclude the paper if this were the end of the story. In fact, however, the story is just beginning.

4. Service with excursions

A variation of the k-server problem which is also studied in Manasse et al. [8] is what they term "server problems with excursions". In this variant, after each request σ_t, it is not necessary to move a server s_i all the way to σ_t. Instead, one can move a server s_i to some intermediate location x, with an associated cost of

$$d(s_i,x)+\alpha\ \min\left\{\min_{j\neq i}\{d(s_j,\sigma_t),d(x,\sigma_t)\}\right\},$$

where $\alpha>0$ is some fixed constant. This models a situation in which it may be relatively expensive to move a server all the way to σ_t, and instead "assistants" can be moved from the intermediate location x (e.g., suppose requests are fires, servers are firehouses and assistants are fire trucks).

How much do excursions help? For simplicity, let us restrict ourselves to the case $\alpha=1$. Define $\overline{\mathrm{OFF}}(\sigma)$ to be the minimum possible off-line cost for serving σ with excursions.

Fact 2. *For all* σ,

$$\mathrm{OFF}(\sigma)\leqslant 2\,\overline{\mathrm{OFF}}(\sigma). \tag{3}$$

Proof (**Sketch**). We restrict our attention to $k=2$. The general case follows by similar considerations. In Fig. 2 we show the behavior of some excursion algorithm achieving $\overline{\mathrm{OFF}}(\sigma)$, and an associated off-line algorithm B acting on the same sequence σ. The "graph" in Fig. 2(c) is given to help understand the analysis. We abuse notation slightly by assuming that servers s_i are initially located at positions s_i.

Now, observe that

$$\begin{aligned}
\mathrm{OFF}(\sigma)\leqslant B(\sigma) &\leqslant d(s_1,\sigma_1)+d(s_2,\sigma_2)+d(\sigma_1,\sigma_3)+d(\sigma_3,\sigma_4)+\cdots\\
&\leqslant (d(s_1,x_1)+d(x_1,\sigma_1))+(d(s_2,x_2)+d(x_2,\sigma_2))\\
&\quad +(d(\sigma_1,x_1)+d(x_1,x_3)+d(x_3,\sigma_3))\\
&\quad +(d(\sigma_3,x_3)+d(x_3,x_4)+d(x_4,\sigma_4))+\cdots\\
&\leqslant 2\{d(s_1,x_1)+d(x_1,\sigma_1)+d(x_1,x_3)+d(x_3,\sigma_3)+\cdots\\
&\quad +d(s_2,x_2)+d(x_2,\sigma_2)+d(x_2,x_5)+\cdots\}\\
&= 2\,\overline{\mathrm{OFF}}(\sigma).
\end{aligned}$$

Here, we have used the triangle inequality for d with a vengeance. □

In fact, the same inequality (3) also applies to on-line algorithms with look-ahead w, namely by allowing excursions, such algorithms can save at most a factor of 2 in

$\overline{\mathrm{OFF}}(\sigma)$:

	σ_1	σ_2	σ_3	σ_4	σ_5	$\cdots$
s_1	x_1	x_1	x_3	x_4	x_4	$\cdots$
s_2	s_2	x_2	x_2	x_2	x_5	$\cdots$

Typical behavior of $\overline{\mathrm{OFF}}(\sigma)$

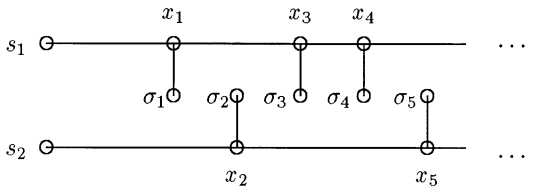
(a)

$B(G)$:

	σ_1	σ_2	σ_3	σ_4	σ_5	$\cdots$
s_1	σ_1	σ_1	σ_3	σ_4	σ_4	$\cdots$
s_2	s_2	σ_2	σ_2	σ_2	σ_5	$\cdots$

Behavior of $B(\sigma)$

(b)



Distances for $\overline{\mathrm{OFF}}(\sigma)$ and $B(\sigma)$

(c)

Fig. 2.

cost. Now, let us define

$$\bar{\rho}_k^{[w]} := \inf_{\bar{A}^{[w]}} \limsup_{\overline{\mathrm{OFF}}(\sigma)\to\infty} \frac{\bar{A}^{[w]}(\sigma)}{\overline{\mathrm{OFF}}(\sigma)}$$

where $\bar{A}^{[w]}$ ranges over all on-line excursion algorithms with look-ahead w.

Fact 3.

$$\bar{\rho}_k^{[w]} \geqslant \bar{\rho}_k^{[w+1]}.$$

Proof. This is clear, since more information about the future can only help. □

This leads us to define

$$\bar{\rho}_k^{[\infty]} := \lim_{w\to\infty} \bar{\rho}_k^{[w]},$$

the limiting behavior of on-line excursion algorithms as the window size tends to infinity. The main result relating $\bar{\rho}_k^{[\infty]}$ to our (nonexcursion) ratios ρ_k is the following.

Lemma 1. *For all k,*

$$\bar{\rho}_k^{[\infty]} \geqslant \rho_k. \tag{4}$$

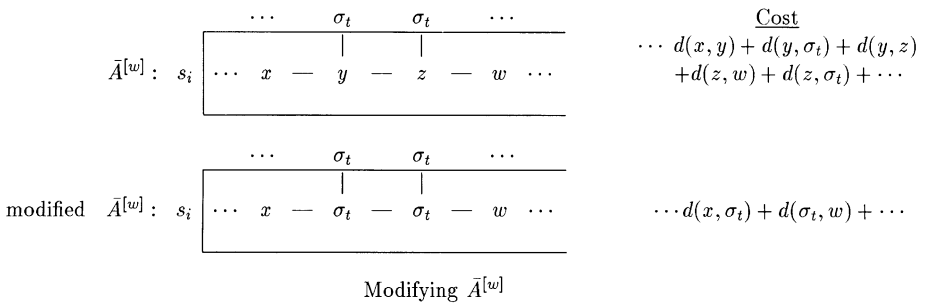


Fig. 3.

In other words, even though we allow excursions, no matter how far we can see into the future (a bounded amount), we cannot improve in general over what an off-line algorithm can achieve for which excursions are not allowed.

Proof. We first observe that if $\sigma_t = \sigma_{t+1}$ then any "sensible" on-line excursion algorithm $\bar{A}^{[w]}$ with look-ahead $w \geqslant 2$, will in fact send a server to σ_t. This follows from computations given in Fig. 3. The cost of the modified algorithm has not increased since by the triangle inequality,

$$d(x,\sigma_t) \leqslant d(x,y) + d(y,\sigma_t)$$
$$d(\sigma_t,w) \leqslant d(z,w) + d(z,\sigma_t).$$

Thus, with $\sigma^{[w]}$ denoting $(\overbrace{\sigma_1,\ldots,\sigma_1}^{w}, \overbrace{\sigma_2,\ldots,\sigma_2}^{w},\ldots)$ for $\sigma=(\sigma_1,\sigma_2,\ldots)$ as before, we have

$$\begin{aligned}
\bar{\rho}_k^{[w]} &= \inf_{\bar{A}^{[w]}} \limsup_{\overline{\mathrm{OFF}}(\sigma)\to\infty} \frac{\bar{A}^{[w]}(\sigma)}{\overline{\mathrm{OFF}}(\sigma)} \\
&\geqslant \inf_{\bar{A}^{[w]}} \limsup_{\overline{\mathrm{OFF}}(\sigma^{[w]})\to\infty} \frac{\bar{A}^{[w]}(\sigma^{[w]})}{\overline{\mathrm{OFF}}(\sigma^{[w]})} \\
&= \inf_{B} \limsup_{\mathrm{OFF}(\sigma)\to\infty} \frac{B(\sigma)}{\mathrm{OFF}(\sigma)} \\
&= \rho_k,
\end{aligned}$$

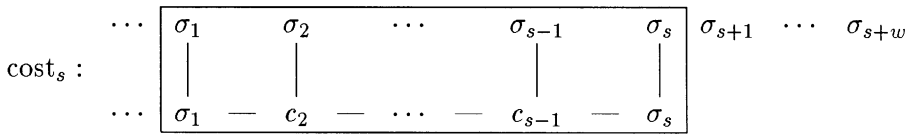
where B ranges over all on-line (nonexcursion) algorithms.

Therefore, $\bar{\rho}_k^{[\infty]} \geqslant \rho_k$ and the lemma is proved. □

Our main conjecture concerning $\bar{\rho}_k^{[\infty]}$ is the following.

Conjecture 2. *For all k,*

$$\bar{\rho}_k^{[\infty]} = \rho_k. \tag{5}$$



Behavior of $\bar{C}^{[w]}$ on a block of σ

Fig. 4.

Note that if both Conjectures 1 and 2 hold then we would have $\bar{\rho}_k^{[\infty]} = k$ for all k.

At present, Conjecture 2 is only known to hold for $k = 1$. This is a consequence of the following result.

Theorem 5. *There is an on-line excursion algorithm $\bar{C}^{[w]}$ with look-ahead $w \geqslant 2$ such that for all σ,*

$$\bar{C}^{[w]}(\sigma) \leqslant \left(1 + \frac{2}{\lfloor w/2 \rfloor}\right) \overline{OFF}(\sigma). \tag{6}$$

Remark. For $w = 1$ it is easy to see that the best possible inequality for any on-line excursion algorithm $\bar{C}^{[1]}$ is $\bar{C}^{[1]}(\sigma) \leqslant 2\,\overline{\mathrm{OFF}}(\sigma)$.

Proof. Let us consider a block of w consecutive requests:

$$\sigma_1, \sigma_2, \ldots, \sigma_{\lfloor w/2 \rfloor}, |\sigma_{\lfloor w/2 \rfloor + 1}, \ldots, \sigma_s, \sigma_{s+1}, \ldots, \sigma_w.$$

Let $d(\sigma_s, \sigma_{s+1})$ be the minimum distance among all $d(\sigma_i, \sigma_{i+1})$ where $\lfloor w/2 \rfloor \leqslant i \leqslant w-1$. The algorithm $\bar{C}^{[w]}$ constructs the *optimal* solution for the request sequence $\sigma_1, \ldots, \sigma_s$ (see Fig. 4). Now, repeat this operation for the block $\sigma_{s+1}, \sigma_{s+2}, \ldots, \sigma_{s+w}$, etc. Let S denote the set of "special" indices s. Then,

$$\left\lfloor \frac{w}{2} \right\rfloor \sum_{s \in S} d(\sigma_s, \sigma_{s+1}) \leqslant \sum_{i \geqslant 1} d(\sigma_i, \sigma_{i+1}) \leqslant 2\,\overline{\mathrm{OFF}}(\sigma).$$

Therefore,

$$\begin{aligned} \bar{C}^{[w]}(\sigma) &= \sum_{s \in S} \mathrm{cost}_s + \sum_{s \in S} d(\sigma_s, \sigma_{s+1}) \\ &\leqslant \left(1 + \frac{2}{\lfloor w/2 \rfloor}\right) \overline{\mathrm{OFF}}(\sigma), \end{aligned}$$

where cost_s denotes the sum of the distances shown in the box in Fig. 4, i.e., $d(\sigma_1, c_2) + d(c_2, \sigma_2) + \cdots + d(c_{s-1}, \sigma_s)$. □

In particular, it follows that

$$\bar{\rho}_1^{[\infty]} = 1 = \rho_1.$$

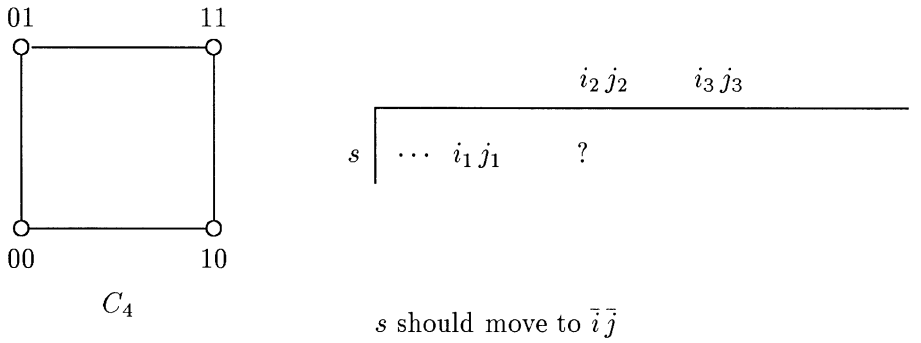


Fig. 5.

5. Window index for a graph

Suppose our metric space M is given by a connected graph G with the *shortest path* metric. That is, the distance between two vertices of G is defined to be minimum number of edges contained in any path joining the vertices. In this case there is a fairly complete theory available for on-line excursion algorithms on G with $k=1$ server and limited look-ahead (again for the choice of "excursion factor" $\alpha=1$). In this section we give a brief survey of what is known for this case.

Theorem 6 (Chung et al. [4]).

For each connected graph G, there is a least value $WX(G)\in\{1,2,3,\ldots\}\cup\{\infty\}$, called the *window index* (or "windex") of G, so that if $w\geqslant WX(G)$ then

$$\bar{\rho}_1^{[w]}=1.$$

In fact, there is an on-line excursion algorithm $\bar{A}^{[w]}$ with look-ahead w so that for all σ,

$$\bar{A}^{[w]}(\sigma)\leqslant\overline{\mathrm{OFF}}(\sigma)+\mathrm{O}(1).$$

In what follows, we give a brief summary of the basic results for this case.

First, note that when $G=C_4$, the cycle on four vertices (see Fig. 5), we have $WX(C_4)=2$. It is easy to see that $WX(C_4)>1$. On the other hand, it is not hard to show that a "majority rule" excursion algorithm with look-ahead 2 is optimal for C_4. That is, if the server is currently at i_1j_1 and the next two requests are i_2j_2 and i_3j_3 then the server should move to $\bar{i}\,\bar{j}$ where $\bar{i}$ is the most frequently occurring value in i_1, i_2, i_3, and $\bar{j}$ is the most frequently occurring value in j_1, j_2, j_3. Other graphs having $WX=2$ are trees, grids and n-cubes, for example (see [3]).

On the other hand, for the complete graph K_3 on 3 vertices, $WX(K_3)=3$. Which graphs G have $WX(G)=2$? (Note that only the trivial graph has windex 1.)

Let us say that G has the *unique Steiner property* if for any three vertices x, y, z, there is a unique vertex s which minimizes

$$d(s,x) + d(s,y) + d(s,z).$$

Also, we call an induced subgraph H of G a *retract* of G if there is a function $\beta : V(G) \to V(G)$ (vertices of G into vertices of H) such that:

(i) $v \in V(H) \Rightarrow \beta(v) = v$,

(ii) $\{u,v\} \in E(G)$ (the edges of G) $\Rightarrow$ either $\{\beta(u), \beta(v)\} \in E(H)$ or $\beta(u) = \beta(v)$.

Next, let us say that a subgraph H is an *isometric* subgraph of G if for all $x, y \in V(H)$,

$$d_H(x,y) = d_G(x,y).$$

Finally, we consider the n-cube Q^n. This graph has as its set of vertices all binary n-tuples $(x_1,\ldots,x_n)$, $x_i = 0$ or 1, and edges $\{(x_1,\ldots,x_n), (y_1,\ldots,y_n)\}$ precisely when the two n-tuples differ in just *one* coordinate (this metric is also known as the Hamming metric in coding theory). We call a subset $X \subset V(Q^n)$ *majority closed* if the "majority" n-tuple formed from any three vertices in X is also in X.

Theorem 7 (Chung et al. [3]). *The following conditions are equivalent for a connected graph G:*

(i) *G has the unique Steiner property;*

(ii) *G is a retract of Q^n for some n;*

(iii) *G is a majority closed isometric subgraph of Q^n for some n;*

(iv) *$WX(G) = 2$.*

To describe the corresponding result for general values of $WX(G)$, we need the following definition. Given graphs $G_i = (V_i, E_i)$, $1 \leqslant i \leqslant r$, define the *Cartesian product* $G_1 \Box G_2 \Box \cdots \Box G_r$ to be the graph with vertex set $V_1 \times V_2 \times \cdots \times V_r$ and edges $\{(v_1, v_2, \ldots, v_r), (v_1', v_2', \ldots, v_r')\}$ where for each i, either $\{v_i, v_i'\} \in E_i$, or $v_i = v_i'$.

For example, if K_2 denotes the complete graph on two vertices (with one edge), then

$$\overbrace{K_2 \Box K_2 \Box \cdots \Box K_2}^{n} \cong Q^n.$$

Theorem 8 (Chung et al. [4]). *$WX(G)$ is finite if and only if G is a retract of $K_{m_1} \Box K_{m_2} \Box \cdots \Box K_{m_r}$ for some choice of $m_1, m_2, \ldots, m_r$. For such G,*

$$WX(G) = \max\{m_i : 1 \leqslant i \leqslant r\}.$$

Corollary. *Almost all graphs G have $WX(G) = \infty$.*

It is not hard to show that if the look-ahead length for G is smaller than $WX(G)$ then you must pay a definite penalty.

Theorem 9 (Chung et al. [4]). *If* $w < WX(G)$ *then*

$$\bar{\rho}_1^{[w]}(G) \geqslant 1 + \frac{1}{w}. \tag{7}$$

Conjecture 3. *The bound in* (7) *is best possible.*

6. Concluding remarks

There are many unresolved questions remaining, some of which we now mention.

To begin with, to what extent do the preceding results extend to the case of $k > 1$ servers with excursions? In particular, does the "windex" phenomenon occur? In other words, is there always a number $WX_k(G)$ so that if $w \geqslant WX_k(G)$ then

$$\bar{\rho}_k^{[w]}(G) = \bar{\rho}_k^{[\infty]}(G)?$$

It is not hard to show that this does in fact happen for $G = K_3$, $k = 2$. In this case we have

$$\bar{\rho}_2^{[1]}(K_3) = \bar{\rho}_2^{[2]}(K_3) = 3,$$

$$\bar{\rho}_2^{[w]}(K_3) = \bar{\rho}_2^{[\infty]}(K_3) = 2 = \rho_2(K_3), \quad w \geqslant 3.$$

What happens for $G = C_5$ (where we know from the earlier result characterizing graphs with finite windex that $WX_1(C_5) = \infty$)?

Can graphs G with $WX_k(G) < \infty$ be characterized? What happens for general metric spaces? How do the previous results change if we allow excursion factors $\alpha \neq 1$ (even for $k = 1$)?

For our basic problem from Section 2 (i.e., the graph has two vertices 0 and 1 with $c(0) = c(1) = 2, k = 1$), we do see an example of the "windex phenomenon". That is, we have $\rho(0,4) = 3 = \rho(t,0)$ for all $t \geqslant 4$. In other words, if you cannot see into the future at all, then it does not help to see more than four steps into the past. Does this phenomenon occur for all $w < \infty$? Is there an efficient algorithm for determining $\rho(u,w)$ for given large values of u and w? What happens for varying costs $c(o)$ and $c(1)$? On more general graphs? With $k > 1$ servers?

For graphs G and $k = 1$, we have previously defined [3] the *asymptotic worst-case average excursion service cost*

$$\bar{\lambda}(G) := \limsup_{\sigma=(\sigma_1,\ldots,\sigma_N)} \frac{1}{N}\overline{\mathrm{OFF}}(\sigma).$$

For example, it is known (see [3]) that:

$$\bar{\lambda}(C_{2m}) = \frac{m}{2}, \quad \bar{\lambda}(C_{2m+1}) = \frac{m(m+1)}{2m+1}.$$

A result of Saks [9] shows that $\bar{\lambda}(G)$ is always a rational number. Is there a polynomial-time algorithm for computing $\bar{\lambda}$? What happens for $k>1$ servers, e.g., is

$$\bar{\lambda}_k(G) := \limsup_{\sigma=(\sigma_1,\ldots,\sigma_N)} \frac{1}{N}\overline{\mathrm{OFF}}_k(\sigma)$$

always rational? Is it computable in polynomial time?

Clearly, there is much work to be done before we have a full understanding of even this rather special topic in the field of on-line algorithms.

Acknowledgements

The authors wish to thank Phong Vo for suggesting this problem to us and pointing out its connection to Internet web page caching.

References

[1] S. Ben-David, A. Borodin, R. Karp, G. Tardos, A. Wigderson, On the power of randomization in on-line algorithms, Proc. 22nd Symp. on the Theory of Algorithms, ACM, 1990, pp. 379–386.
[2] M. Chrobak, L.L. Larmore, The server problem and on-line games, in: DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 7, American Mathematical Society, Providence, RI, 1992, pp. 74–94.
[3] F.R.K. Chung, R.L. Graham, M.E. Saks, Dynamic search in graphs, in: Discrete Algorithms and Complexity, Academic Press, New York, 1987, pp. 351–388.
[4] F.R.K. Chung, R.L. Graham, M.E. Saks, A dynamic location problem for graphs, Combinatorica 9 (1989) 111–131.
[5] A. Fiat, Y. Rabini, Y. Ravid, Competitive k-server algorithms, Proc. 31st Ann. Symp. on Foundation of Computer Science, J. Comput. System Sci. 48 (1994) 410–428.
[6] A. Fiat, G.J. Woeginger (Eds.), Online Algorithms: State of the Art, Springer, Berlin, 1998.
[7] E. Grove, The harmonic on-line k-server algorithm is competitive, in: On-line Algorithms, DIMACS Series, Discrete Mathematics and Theoretical Computer Science, vol. 7, American Mathematical Society, Providence, RI, 1992, pp. 65–75.
[8] M.S. Manasse, L.A. McGeoch, D.D. Sleator, Competitive algorithms for server problems, J. Algorithms 11 (1990) 208–230.
[9] M.E. Saks, A limit theorem for (min,+) matrix multiplication, Math. Oper. Res. 13 (1988) 606–618.
[10] D. Sleator, R.E. Tarjan, Amortized efficiency for list update and paging rules, Comm. ACM 28 (1985) 202–208.