数理解析研究所講究録 1241 巻 2001 年 48-56

Axis-bound CNN Problem

京大・情報学研究科　米澤 弘毅 (Kouki Yonezawa), 岩間 一雄 (Kazuo Iwama)
School of Informatics
Kyoto University

Abstract: The CNN Problem is a kind of a one-server problem in which a request moves on the two-dimensional plane. If the request appears at position (i,j), the server does not have to move to the place (i,j) itself but has only to move somewhere given by (i,l) or (k,j). Namely it is enough to move either to the same row or to the same column. The goal is to minimize the total moving distance of the server. This problem has been quite popular for the last several years but it is still open whether or not there is a competitive algorithm, i.e., an algorithm whose competitive ratio is bounded by a constant. In this paper we consider a natural restriction of the problem, namely, the server can move only on the X-axis and Y-axis, which can be viewed as a one-dimensional version of the original problem. It is shown that there is an online algorithm for this "axis-bound CNN" whose competitive ratio is at most 9.0.

1 Introduction

Imagine a mesh-like city like Manhattan, where "scenes" occur at different places one after another, or we can also think that a scene moves from place to place in the city. A camera crew (of CNN) wishes to shoot the scene by following its move. Suppose that the scene is now at the crossroad of i-th street and j-th avenue. Then the crew does not have to go to the exact crossroad but only has to go somewhere on the i-th street or somewhere on the j-th avenue. The goal is to minimize the total moving distance of the crew. For example, one possible strategy for the crew is to move just on the first street back and forth. However, it is easy to see that this strategy is not competitive: If the scene moves back and forth on the second street, then the crew also has to move similarily back and forth on the first street. One can easily see, however, that if the crew moves to the second street at the beginning, then it does not have to move at all after that.

Due to [KT00], this problem was first proposed by Saks and Burley and its name was suggested by Woeginger. The problem was mentioned several times including at a Dagstuhl meeting in 1996 by Koutsoupias and in the open-problem session at ESA99 by Woeginger. Thus the problem has been quite popular in the community but it is still open whether or not competitive algorithms, i.e., algorithms of a constant competitive ratio, exist. The only known result is due to Koutsoupias and Taylor [KT00] showing that any algorithm must have a competitive ratio at least $6+\sqrt{17}$, where the authors look at this problem as a variant of the k-server problem [see, e.g., CKPV91, FRR90, KP94, MMS90]. Their lower bound of $6+\sqrt{17}$ comes from a lower bound for the weighted 2-server problem [FR94]. [KT00] claims that it is hard to apply positive results for the k-server problem to the CNN problem, but their conjecture was still positive, i.e., that there would be a competitive algorithm.

In this paper, we consider a natural restriction of this problem, namely, the crew can move only on the X-axis and Y-axis. For this restricted version, called the *axis-bounded* CNN, the above conjecture is true, i.e., we show that there is an online algorithm whose competitive ratio is at most 9.0. (Note that $9.0 < 6+\sqrt{17}$, but this is of course not a contradiction.) Our strategy is to follow after the "currently optimal position" *with some delay*. The currently optimal position (or the currently optimal solution in general) means the position such that the optimal offline algorithm should place the crew there if the input finishes at that moment. (Note that the currently optimal solution, or an optimal offline solution in general, can be computed easily from the previous input in the case of this problem. Some problems, such as the list-accessing problem, does not have this property [AHD83, ST85].) It appears that several existing online algorithms follow this basic scheme. For example, the algorithm for the famous ski-rental problem [Kar92, BE98] follows the currently optimal solution *without* delay. The Double-Coverage algorithm for the k-server problem is another example [CL91] (although the relation might not be so clear). As shown later,

the key issue in this paper is how to choose an appropriate value for the delay.

2 CNN Problem and Bind Restriction

The CNN Problem (or CNN for short) consists of the *scene* and the *camera crew* (or the crew for short). The scene appears sequentially on the two-dimensional plane, i.e., at $s_1 = (x_1, y_1)$, $s_2 = (x_2, y_2)$, $\cdots$, and $s_n = (x_n, y_n)$, where x_i and y_i are any real numbers. The sequence $S = s_1 s_2 \cdots s_n$ is called a *scene-sequence* (or an S-seq for short). The camera crew is originally placed at $(0, 0)$ and for a given S-seq S it is to move sequentially to $p_1 = (u_1, v_1)$, $p_2 = (u_2, v_2)$, $\cdots$, $p_n = (u_n, v_n)$, where the condition that $u_i = x_i$ or $v_i = y_i$ has to be met for all $1 \leq i \leq n$. Such a sequence $P = p_1 p_2 \cdots p_n$ is called a *(proper) crew-sequence* (or C-seq for short) for the S-seq S. The *cost* $\sigma(P)$ of the C-seq P is defined as

$$\sigma(P) = \|p_1 - p_0\| + \|p_2 - p_1\| + \cdots + \|p_n - p_{n-1}\|,$$

where $p_0 = (0, 0)$ and $\|p_{i+1} - p_i\| = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}$.

CNN is said to be *axis-bound* if the crew always has to stay on the X-axis or Y-axis. Namely, if $S = (x_1, y_1), \cdots (x_n, y_n)$ and $P = p_1 p_2 \cdots p_n$, then $p_i = (x_i, 0)$ or $(0, y_i)$. Such a C-seq is called an *axis-bound C-seq*. In this paper we mainly discuss axis-bound CNN, and therefore axis-bound C-seqs are simply denoted by C-seqs without otherwise stated. For an S-seq S, $\Gamma(S)$ denotes a set of all such (proper) C-seqs. For axis-bound CNN, we use the l_1-norm for calculating the cost, i.e., $\|p_{i+1} - p_i\|$ is defined to be $|x_{i+1} - x_i| + |y_{i+1} - y_i|$. $\mathrm{OPT}(S)$ denotes the optimal cost of C-seqs for the S-seq S, i.e., $\mathrm{OPT}(S) = \min_{P \in \Gamma(S)} \sigma(P)$. We also define $\mathrm{OPT}^X(S) = \min_{P \in \Gamma^X(S)} \sigma(P)$ where $\Gamma^X(S)$ is a set of C-seqs P^X whose last position is on the X-axis, i.e., if $P^X = (u_1, v_1) \cdots (u_n, v_n)$ then v_n must be 0. $\mathrm{OPT}^Y(S)$ is defined similarly. Obviously $\mathrm{OPT}(S) = \min(\mathrm{OPT}^X(S), \mathrm{OPT}^Y(S))$.

Let an S-seq $S_{n-1} = (x_1, y_1) \cdots (x_{n-1}, y_{n-1})$ and $S_n = S_{n-1}(x_n, y_n)$. Several basic lemmas are as follows:

Lemma 1. $\mathrm{OPT}^X(S_n) \leq \mathrm{OPT}^Y(S_n) + |x_n| + |y_n|$. $\mathrm{OPT}^Y(S_n) \leq \mathrm{OPT}^X(S_n) + |x_n| + |y_n|$.

Proof. Otherwise, we can get a smaller $\mathrm{OPT}^X(S_n)$ by moving the crew from $(0, y_n)$ to $(x_n, 0)$. Similarly for the second statement. ∎

Lemma 2. $\mathrm{OPT}^X(S_n) = \min(\mathrm{OPT}^X(S_{n-1}) + |x_n - x_{n-1}|, \mathrm{OPT}^Y(S_{n-1}) + |y_{n-1}| + |x_n|)$. $\mathrm{OPT}^Y(S_n) = \min(\mathrm{OPT}^X(S_{n-1}) + |x_{n-1}| + |y_n|, \mathrm{OPT}^Y(S_{n-1}) + |y_n - y_{n-1}|)$.

Proof. Obvious by Lemma 1.

Lemma 3. If $x_n = x_{n-1}$, then $\mathrm{OPT}^X(S_n) = \mathrm{OPT}^X(S_{n-1})$. If $y_n = y_{n-1}$, then $\mathrm{OPT}^Y(S_n) = \mathrm{OPT}^Y(S_{n-1})$.

Proof. Since $x_n = x_{n-1}$, $\mathrm{OPT}^X(S_n) = \min(\mathrm{OPT}^X(S_{n-1}), \mathrm{OPT}^Y(S_{n-1}) + |y_{n-1}| + |x_{n-1}|)$ by Lemma 2. Furthermore, since $\mathrm{OPT}^X(S_{n-1}) \leq \mathrm{OPT}^Y(S_{n-1}) + |x_{n-1}| + |y_{n-1}|$ by Lemma 1, it follows that $\mathrm{OPT}^X(S_n) = \mathrm{OPT}^X(S_{n-1})$. Similarly for the second statement. ∎

Lemma 2 means that $\mathrm{OPT}^x(S_n)$, $\mathrm{OPT}^y(S_n)$ and $\mathrm{OPT}(S_n)$ can be calculated in polynomial time in n. Here are some examples for the general and axis-bound CNN.

Example 1. Consider the following algorithm $\mathcal{A}_1$ for the general CNN:

$$(u_i, v_i) = \begin{cases} (u_{i-1}, y_i) & \text{if } |y_i - v_{i-1}| \leq |x_i - v_{i-1}| \\ (x_i, v_{i-1}) & \text{otherwise.} \end{cases}$$

Namely, according to this algorithm, the camera crew always moves horizontally or vertically depending on which is shorter. Unfortunately this algorithm $\mathcal{A}_1$, which is a typical greedy algorithm, has a poor competitive ratio: Consider the input $S_m = (1, 2), (2, 2), (3, 2), \cdots, (m, 2)$ as shown in Fig. 1. For the first

position $(1,2)$ of the scene, the camera crew moves to the right since it is shorter than moving up. Then the crew continues to move to the right since it is always shorter than moving up. Thus the cost of this online algorithm for S-seq S_m is m. However one can see that the optimal camera crew should move to $(0,2)$ at the first step and to stay there after that. Thus the optimal cost is only two; in other words, the competitive ratio of $\mathcal{A}_1$ is at least $m/2$.

In the case of the axis-bound CNN, there are only two possibilities for the next position of the crew, i.e., staying on the current axis or moving to the other axis. Changing the axis, say from X-axis to Y-axis, usually costs more than simply moving horizontally or vertically; namely the greedy algorithm is not competitive for the axis-bound CNN, either.

Example 2. Note that in the previous example, $\text{OPT}(S_1) = 1$, $\text{OPT}(S_2) = 2$ at $(2,0)$ or $(0,2)$, $\text{OPT}(S_3) = 2$ at $(0,2)$ and so on. Hence, if we always follow the (currently) optimal position (recall that the optimal position can be calculated in polynomial time), then such an algorithm would move the crew to $(1,0)$, to $(2,0)$ and then to $(0,2)$. This algorithm is thus competitive for the input S_m. Unfortunately again, there is the following adversary: See Fig. 2. The scene first appears at (k,k). Then the camera crew is to move $(0,k)$ or $(k,0)$ both of which are currently-optimal. Suppose that it moved to $(0,k)$. Then the scene moves to $(k-1,k)$, $(k-1,k-2)$, $(k-3,k-2)$, $(k-3,k)$, $(k-1,k)$, and continues to move round on this square. It is not hard to see that the optimal position is $(0,k)$ or $(k,0)$ first and then changes between $(k-1,0)$ and $(0,k-2)$ alternately. Hence the algorithm which follows the optimal position has to move between those distant two positions, and costs approximately k times as much as the optimal cost which is achieved by staying on the Y-axis. (Ironically, the greedy algorithm is almost optimal in this case.)

3 Competitive Algorithms

3.1 Orthogonal Sequences

Before presenting our algorithm, we need an important preparation, i.e., orthogonalization of S-seqs. Suppose that $S_n = (x_1,y_1)\cdots(x_n,y_n)$ is an S-seq. We first define an *axis-decomposition* of S_n, denoted by $F(S_n)$, as follows: Suppose that a line segment $(x_i,y_i)(x_{i+1},y_{i+1})$ intersects with the X-axis or the Y-axis. Then we denote such intersections by (x'_i,y'_i) (i.e., $x'_i = 0$ or $y'_i = 0$, see Fig. 3). $F(S_n)$ is an S-seq such that for each interval $(x_i,y_i)(x_{i+1},y_{i+1})$ in S_n, the intersection (x'_i,y'_i) is inserted (if any). Note that a line segment can intersect with both X-axis and Y-axis. Two intersections are inserted in this case.

Lemma 4. $\text{OPT}^X(S_n) = \text{OPT}^X(F(S_n))$ and $\text{OPT}^Y(S_n) = \text{OPT}^Y(F(S_n))$.

Proof. (By Induction.) If $n = 1$, i.e., $S_1 = (x_1,y_1)$ is given then $F(S_1)$ is also (x_1,y_1) by definition and therefore the lemma obviously holds. Suppose that the lemma holds for $S_i = (x_1,y_1)\cdots(x_i,y_i)$, and now consider $S_{i+1} = (x_1,y_1)\cdots(x_i,y_i)(x_{i+1},y_{i+1})$. If there is no intersection with the axis between (x_i,y_i) and (x_{i+1},y_{i+1}), then the lemma obviously holds again. So, suppose that there are intersections. We only prove the case that the intersection is with the Y-axis only and for OPT^X (see Fig. 3 again). The other cases are very similar and omitted.

Let $X_1 = (x_i,0)$, $X_2 = (x_{i+1},0)$, $X_3 = (x'_i,0) = (0,0)$, $Y_1 = (0,y_i)$, $Y_2 = (0,y_{i+1})$ and $Y_3 = (0,y'_i)$. Then using Lemma 2 twice we can claim that $\text{OPT}^X(F(S_{i+1}))$ is the minimum of the following four values, where $d(z_1,z_2)$ denotes the l_1-distance between point z_1 and point z_2.

$$\begin{aligned}
&\text{(i)} \quad \text{OPT}^X(S_i) + d(X_1,X_3) + d(X_3,X_2),\\
&\text{(ii)} \quad \text{OPT}^X(S_i) + d(X_1,Y_3) + d(Y_3,X_2),\\
&\text{(iii)} \quad \text{OPT}^Y(S_i) + d(Y_1,X_3) + d(X_3,X_2),\\
&\text{(iv)} \quad \text{OPT}^Y(S_i) + d(Y_1,Y_3) + d(Y_3,X_2).
\end{aligned}$$

It is not hard to see (i) $\leq$ (ii) and (iii) $\leq$ (iv) always hold. Also obviously,

$$d(X_1, X_3) + d(X_3, X_2) = d(X_1, X_2),$$
$$d(Y_1, X_3) + d(X_3, X_2) = d(Y_1, X_2).$$

Thus we can write that

$$\text{OPT}^X(F(S_{i+1})) = \min(\text{OPT}^X(S_i) + d(X_1, X_2), \text{OPT}^Y(S_i) + d(Y_1, X_2)).$$

The righthand side is equal to $\text{OPT}^X(S_{i+1})$ by definition and the lemma holds. ∎

We then define an *orthogonal-decomposition* of $F(S_n)$, denoted by $\delta(S_n)$, as follows: Let $F(S_n) = (x_1, y_1) \cdots (x_m, y_m)$. Then $\delta(S_n)$ is written as

$$\delta(S_n) = (x_1, y_1)(x'_1, y'_1)(x_2, y_2) \cdots (x_i, y_i)(x'_i, y'_i)(x_{i+1}, y_{i+1}) \cdots (x_{m-1}, y_{m-1})(x'_{m-1}, y'_{m-1})(x_m, y_m),$$

where x'_i and y'_i are determined as follows (recall that (x_i, y_i) and (x_{i+1}, y_{i+1}) are in the same quadrant including on the border):

(i) $x'_i = x_{i+1}$ and $y'_i = y_i$ if $|x_{i+1}| < |x_i|$.

(ii) $x'_i = x_i$ and $y'_i = y_{i+1}$ otherwise.

Namely, if point (x_{i+1}, y_{i+1}) is closer to the Y-axis than (x_i, y_i), then the movement of the scene from (x_i, y_i) to (x_{i+1}, y_{i+1}) is decomposed into the horizontal movement followed by the vertical movement (see Fig. 4). Otherwise, the vertical movement comes first.

Lemma 5. $\text{OPT}^X(S_n) = \text{OPT}^X(\delta(S_n))$ and $\text{OPT}^Y(S_n) = \text{OPT}^Y(\delta(S_n))$.

Proof. The proof is by induction and is similar to the previous lemma. The base case for the induction is omitted and suppose that the lemma is true for S_i. Again we only consider the case that $|x_{i+1}| < |x_i|$ and $|y_{i+1}| > |y_i|$ as shown in Fig. 4 and only for OPT^X. $\text{OPT}^X(\delta(S_{i+1}))$ is the minimum among the following four values:

(i) $\text{OPT}^X(S_i) + d(X_1, X_2) + d(X_2, X_2)$,
(ii) $\text{OPT}^X(S_i) + d(X_1, Y_1) + d(Y_1, X_2)$,
(iii) $\text{OPT}^Y(S_i) + d(Y_1, X_2) + d(X_2, X_2)$,
(iv) $\text{OPT}^Y(S_i) + d(Y_1, Y_2) + d(Y_2, X_2)$.

One can see that (i) $\leq$ (ii) and (iii) $\leq$ (iv) always hold and $d(X_2, X_2) = 0$. So, $\text{OPT}^X(\delta(S_{i+1})) = \min(\text{OPT}^X(S_i) + d(X_1, X_2), \text{OPT}^Y(S_i) + d(Y_1, X_2))$, which is equal to $\text{OPT}^X(S_{i+1})$ by definition. ∎

3.2 Algorithm OptFollow and Its Competitiveness

The two examples given in Sec. 2 suggest the following: (i) If the camera crew is now on the X-axis and OPT^X is getting larger than OPT^Y, then the camera crew must move to the position on the Y-axis ultimately. (ii) However, the camera crew should not make such an axis-change move too often. Namely, it should postpone the axis-change move until $\text{OPT}^X - \text{OPT}^Y$ (or vice versa) becomes greater than some value d. Note that the cost for an axis-change move is usually higher than an axis-keep move, i.e., the former needs some extra cost than the latter.

The problem is of course how we select this value d. In the following algorithm, called OptFollow, this value d is related to the next position of the scene. Since this position is related to the (extra) moving cost the camera crew has to pay when changing the axis, it is also true that d is related to this extra cost. We say that an S-seq $S_n = (x_1, y_1) \cdots (x_n, y_n)$ is *orthogonal* if $x_i = x_{i+1}$ or $y_i = y_{i+1}$ holds for all $1 \leq i \leq n-1$. We first define the operation of OptFollow for an orthogonal S-seq S_n:

Algorithm OptFollow (for orthogonal S-seqs)

(i) For the first position (x_1, y_1) of the scene, the crew moves to $(x_1, 0)$ if $|x_1| \leq |y_1|$ and to $(0, y_1)$ otherwise.

(ii) Note that the crew must be at $(x_i, 0)$ or at $(0, y_i)$ after the scene has appeared at (x_i, y_i). Suppose that the crew is at $(x_i, 0)$. Then, for the next position (x_{i+1}, y_{i+1}) of the scene, (a) the crew stays at $(x_i, 0)$ if $x_{i+1} = x_i$. Otherwise (i.e., if $y_{i+1} = y_i$), (b) it moves to $(x_{i+1}, 0)$ if $\mathrm{OPT}^X(S_{i+1}) - \mathrm{OPT}^Y(S_{i+1}) \leq |y_{i+1}|$ and (c) it moves to $(0, y_{i+1})$ otherwise. (see Fig. 5 for (b) and (c))

(iii) Suppose that the crew is at $(0, y_i)$. Then for (x_{i+1}, y_{i+1}), (a) it stays at $(0, y_i)$ if $y_{i+1} = y_i$ and otherwise, (b) it moves to $(0, y_{i+1})$ if $\mathrm{OPT}^y(S_{i+1}) - \mathrm{OPT}^x(S_{i+1}) \leq |x_{i+1}|$ and (c) to $(x_{i+1}, 0)$ otherwise.

Now we define OptFollow for a general S-seq $S_n = (x_1, y_1) \cdots (x_n, y_n)$. Roughly speaking, the crew moves exactly the same as the crew would move for $\delta(S_n)$. In more detail:

Algorithm OptFollow (for general S-seqs)

(i) For the first (x_1, y_1), the crew moves the same as before.

(ii) Suppose that the crew is at $(x_i, 0)$ for (x_i, y_i). Then for the next scene-position (x_{i+1}, y_{i+1}), it translates the path from (x_i, y_i) to (x_{i+1}, y_{i+1}) into the orthogonal path $\delta(F((x_i, y_i)(x_{i+1}, y_{i+1})))$ as illustrated in Fig. 6. Then the crew simulates its movement for $\delta(F((x_i, y_i)(x_{i+1}, y_{i+1})))$ completely.

(iii) Similarly for the case that the crew is at $(0, y_i)$ and omitted.

It should be noted that the crew may make a strange movement as illustrated in Fig. 7. This kind of movement can be avoided by slightly changing the algorithm (i.e., computing the shortest path in advance), but we do not do so in the following analysis.

Lemma 6. If OptFollow achieves a competitive ratio of r for orthogonal S-seqs, then it achieves r for general inputs.

Proof. For a general S-seqs S_n, let $\mathrm{OPT}(S_n)$ and $\mathcal{A}(S_n)$ be the optimal cost and the cost of OptFollow, respectively. One can see that $\mathrm{OPT}(S_n) = \mathrm{OPT}(\delta(S_n))$ by Lemma 5 and that $\mathcal{A}(S_n) = \mathcal{A}(\delta(S_n))$ by the definition of OptFollow. ∎

By this lemma, we can concentrate upon only orthogonal inputs. From now on, we assume that all S-seqs are orthogonal. Also for convenience of the following argument, we divide a given S-seq $S = (x_1, y_1) \cdots (x_n, y_n)$ into *phases*. Phase i (≥ 1) consists of a subsequence $(x_{p_i}, y_{p_i})(x_{p_i+1}, y_{p_i+1}) \cdots (x_{p_i+q_i}, y_{p_i+q_i})$ of S, such that: (i) OptFollow has made the i-th axis-change when the scene appeared at (x_{p_i}, y_{p_i}). (The first movement from $(0,0)$ to (x_1, y_1) is defined to be the 0th axis-change.) (ii) No axis-change is made at (x_{p_i+1}, y_{p_i+1}), $\cdots$, or at $(x_{p_i+q_i}, y_{p_i+q_i})$. (iii) OptFollow again makes (the $(i+1)$st) axis-change for $(x_{p_i+q_i+1}, y_{p_i+q_i+1})$ $(= (x_{p_{i+1}}, y_{p_{i+1}}))$ or $(x_{p_i+q_i}, y_{p_i+q_i})$ is the last position of S. Thus the crew stays on the same axis during a phase, which is called an X-phase (Y-phase, resp.) if the camera crew stays on X-axis (Y-axis, resp.). Note that phases are well-defined since OptFollow is a deterministic algorithm. Now we prove that OPT must increase at least by $|x_{p_i}|$ or $|y_{p_i}|$, depending on whether it is an X-phase or a Y-phase, during this phase, which plays a key role in our analysis.

Lemma 7. Suppose that $(x_{p_i}, y_{p_i}) \cdots (x_{p_i+q_i}, y_{p_i+q_i})$ is an X-phase where (x_{p_i}, y_{p_i}) is not the first appearance (i.e., $p_i > 1$) and $(x_{p_i+q_i}, y_{p_i+q_i})$ is not the last one. Also let $\mathrm{OPT}_i = \mathrm{OPT}(S_{p_i})$ and $\mathrm{OPT}_{i+1} = \mathrm{OPT}(S_{p_{i+1}})$ $(= \mathrm{OPT}(S_{p_i+q_i+1}))$. Then $\mathrm{OPT}_{i+1} - \mathrm{OPT}_i \geq |x_{p_i}|$. As a dual case, if $(x_{p_i}, y_{p_i}) \cdots (x_{p_i+q_i}, y_{p_i+q_i})$ is a Y-phase, then $\mathrm{OPT}_{i+1} - \mathrm{OPT}_i \geq |y_{p_i}|$.

Proof. Fig. 8 illustrates how the values of $\mathrm{OPT}^X(S_{p_i+j})$ and $\mathrm{OPT}^Y(S_{p_i+j})$ changes for $j = 0, 1, \cdots$. We can make the following observations:

(1) At the beginning of this phase, $\mathrm{OPT}^Y(S_{p_i}) - \mathrm{OPT}^X(S_{p_i}) \geq |x_{p_i}|$ since OptFollow has just moved the crew from the Y-axis to the X-axis.

(2) While $\mathrm{OPT}^Y(S_{p_i+j}) \geq \mathrm{OPT}^X(S_{p_i+j})$, the value of OPT^X increases monotonically and eventually

catches up the value of OPT^Y within this phase. We assume that this happens at point W in the Fig. 8. Also, without loss of generality we can assume that W exactly corresponds to some appearance of the scene. (Otherwise, i.e., if this happens between (x_{p_i+j}, y_{p_i+j}) and $(x_{p_i+j+1}, y_{p_i+j+1})$, then we can insert a new appearance without changing the cost of OPT or the cost of OptFollow.) Since $\text{OPT}^X = \text{OPT}^Y = \text{OPT}$ at W, OPT_{i+1} is at least as large as this value (note that OPT does not decrease).

From these facts, (1) and (2), we are done if we can show that for any j the value of $\text{OPT}^Y(S_{p_i+j})$ is at least as large as $\text{OPT}^X(S_{p_i}) + |x_{p_i}|$. Suppose contrarily that $\text{OPT}^Y(S_{p_i+j}) < \text{OPT}^X(S_{p_i}) + |x_{p_i}|$ for some $j \geq 1$. Such situation could only happen when the value of OPT^Y decreases, since $\text{OPT}^Y(S_{p_i}) \geq \text{OPT}^X(S_{p_i}) + |x_{p_i}|$ where $j = 0$.

OPT^Y can decrease by the movement of the scene as shown in Fig. 9. Namely, $\text{OPT}^Y(S_{p_i+j-1}) + d_1 > \text{OPT}^X(S_{p_i+j}) + d_2$, where d_1 and d_2 are the distances of the crew's movements from $(0, y_{p_i+j-1})$ to $(0, y_{p_i+j})$ and from $(x_{p_i+j}, 0)$ to $(0, y_{p_i+j})$, respectively. Thus one can see that

$$\text{OPT}^Y(S_{p_i+j}) \geq \text{OPT}^X(S_{p_i+j}) + d_2 \geq \text{OPT}^X(S_{p_i+j}) + |x_{p_i+j}|.$$

Since we have assumed that $\text{OPT}^Y(S_{p_i+j}) < \text{OPT}^X(S_{p_i}) + |x_{p_i}|$, it follows that $\text{OPT}^X(S_{p_i}) + |x_{p_i}| > \text{OPT}^X(S_{p_i+j}) + |x_{p_i+j}|$ or equivalently

$$\text{OPT}^X(S_{p_i+j}) - \text{OPT}^X(S_{p_i}) < |x_{p_i}| - |x_{p_i+j}|.$$

However, this is a contradiction since the crew must move at least the distance of $|x_{p_i}| - |x_{p_i+j}|$ on the X-axis and OPT^X must increase at least by this value. ∎

Lemma 8. Under the same condition as Lemma 7 except that (x_{p_i}, y_{p_i}) may be (x_1, y_1), let $\mathcal{A}_i = \mathcal{A}(S_{p_i})$ and $\mathcal{A}_{i+1} = \mathcal{A}(S_{p_{i+1}})$. Then $\mathcal{A}_{i+1} \leq \mathcal{A}_i + 2(\text{OPT}_{i+1} - \text{OPT}_i) + |x_{p_i}| + 3|y_{p_{i+1}}|$. As a dual case, if the phase $(x_{p_i}, y_{p_i}) \cdots (x_{p_i+q_i}, y_{p_i+q_i})$ is a Y-phase, then $\mathcal{A}_{i+1} \leq \mathcal{A}_i + 2(\text{OPT}_{i+1} - \text{OPT}_i) + |y_{p_i}| + 3|x_{p_{i+1}}|$.

Proof. Fig. 10 illustrates how the scene and the crew move during this x-phase. It should be noted that the scene's last move in this phase from $(x_{p_i+q_i}, y_{p_i+q_i})$ to $(x_{p_i+q_i+1}, y_{p_i+q_i+1})$ $(= (x_{p_{i+1}}, y_{p_{i+1}}))$ must be horizontal. (Reason: If this move is vertical, then OPT^X does not change by Lemma 1. So, if $\text{OPT}^Y(S_{p_i+q_i}) \leq \text{OPT}^X(S_{p_i+q_i})$, then OPT^Y increases for this scene's move. Otherwise if $\text{OPT}^Y(S_{p_i+q_i}) > \text{OPT}^X(S_{p_i+q_i})$, then OPT^Y may decrease but $\text{OPT}^Y(S_{p_i+q_i+1})$ must not be less than $\text{OPT}^X(S_{p_i+q_i+1})$. In either case, OptFollow does not make an axis-change.)

Another fact we should know is that the distance OptFollow moves the crew in this phase until the scene has appeared at $(x_{p_i+q_i}, y_{p_i+q_i})$ (i.e., $= \mathcal{A}(S_{p_i+q_i}) - \mathcal{A}(S_{p_i})$) is bounded above by $\text{OPT}^X(S_{p_i+q_i}) - \text{OPT}^X(S_{p_i})$. The reason is as follows: While $\text{OPT}^X \leq \text{OPT}^Y$, the distance OptFollow moves the crew in each step is exactly the same as the increase of OPT^X. After OPT^X becomes greater than OPT^Y, the amount of increase (or even decrease) of OPT^X may be different from the moving distance of the crew (= the horizontal moving distance of the scene) at that step. Namely, OPT^X may be equal to the previous OPT^Y + the distance of an axis-change. However, this means that OptFollow would also make an axis-change, which does not happen until $(x_{p_{i+1}}, y_{p_{i+1}})$ by the assumption.

Now look at Fig. 10 again. $\mathcal{A}_{i+1} - \mathcal{A}_i$ is the sum of (i) the moving distance of the crew until $(x_{p_i+q_i}, y_{p_i+q_i})$ and (ii) the cost of an axis-change for $(x_{p_{i+1}}, y_{p_{i+1}})$. As shown above, the cost for (i) is at most $\text{OPT}^X(S_{p_i+q_i}) - \text{OPT}^X(S_{p_i})$. One can also see that the cost for (ii) is at most $\text{OPT}^X(S_{p_i+q_i}) - \text{OPT}^X(S_{p_i}) + |x_{p_i}| + |y_{p_{i+1}}|$. (This is for the case that $|x_{p_i+q_i}| \geq |x_{p_i}|$. If $|x_{p_i+q_i}| < |x_{p_i}|$, then (ii) is bounded by a much smaller value.) By summing up (i) and (ii), we can get

$$\mathcal{A}_{i+1} - \mathcal{A}_i \leq 2(\text{OPT}^X(S_{p_i+q_i}) - \text{OPT}^X(S_{p_i})) + |x_{p_i}| + |y_{p_{i+1}}|. \qquad (*)$$

Now notice that $\text{OPT}^X(S_{p_i+q_i}) - \text{OPT}^Y(S_{p_i+q_i}) \leq |y_{p_i+q_i}| = |y_{p_{i+1}}|$, since the phase-change did not happen when the scene has appeared at $(x_{p_i+q_i}, y_{p_i+q_i})$. Also, one can see that $\text{OPT}^X(S_{p_i}) = \text{OPT}_i$ and $\text{OPT}^Y(S_{p_i+q_i}) = \text{OPT}^Y(S_{p_{i+1}}) = \text{OPT}_{i+1}$. Substituting these three inequalities to $(*)$, we can get

$$\begin{aligned} \mathcal{A}_{i+1} - \mathcal{A}_i &\leq 2(\text{OPT}^Y(S_{p_i+q_i}) + |y_{p_{i+1}}| - \text{OPT}^X(S_{p_i})) + |x_{p_i}| + |y_{p_{i+1}}| \\ &= 2(\text{OPT}_{i+1} - \text{OPT}_i) + |x_{p_i}| + 3|y_{p_{i+1}}|, \end{aligned}$$

which is what we wanted to prove.

It should be noted that $(x_{p_i+q_i}, y_{p_i+q_i})$ might not exist, i.e., the scene migl to $(x_{p_{i+1}}, y_{p_{i+1}})$ directly. In that case, as we did in the previous proof, we can in corresponding to $(x_{p_i+q_i}, y_{p_i+q_i})$ without giving any effect to all the costs involved su and $\mathcal{A}$.

Theorem 1. $\mathcal{A}(S_n)/\text{OPT}(S_n) \leq 9$.

Proof. We only consider the following case: (i) For (x_1, y_1), the crew moves to (ii) Then OptFollow makes a sequence of axis-changes between $(x_{p_1+q_1}, y_{p_1+q_1})$ and axis to the Y-axis, between $(x_{p_2+q_2}, y_{p_2+q_2})$ and (x_{p_3}, y_{p_3}) from the Y-axis to th $(x_{p_{k-1}+q_{k-1}}, y_{p_{k-1}+q_{k-1}})$ and (x_{p_k}, y_{p_k}) from the Y-axis to the X-axis. (iii) Finally (may be zero) moves on the X-axis. Other cases are similar and omited.

We first calculate $\text{OPT}(S_n)$. By applying Lemma 7 to the y-phase $(x_{p_2}, y_{p_2}) \cdots$ claim that

$$\text{OPT}(S_{p_3}) - \text{OPT}(S_{p_2}) \geq |y_{p_2}|.$$

Similarly

$$\text{OPT}(S_{p_4}) - \text{OPT}(S_{p_3}) \geq |x_{p_3}|, \cdots, \text{OPT}(S_{p_k}) - \text{OPT}(S_{p_{k-1}}) \geq |y_{p_k}$$

By summing up these inequalities, we obtain

$$\text{OPT}(S_{p_k}) - \text{OPT}(S_{p_2}) \geq |y_{p_2}| + |x_{p_3}| + \cdots + |y_{p_{k-1}}|.$$

Since $\text{OPT}(S_{p_2})$ is obviously at least $|x_1|$, we can get the key inequality

$$\text{OPT}(S_n) \geq \text{OPT}(S_{p_k}) \geq |x_1| + |y_{p_2}| + \cdots + |y_{p_{k-1}}|. \qquad (*$$

We next calculate $\mathcal{A}(S_n)$. By applying Lemma 8 to the x-phase $(x_1, y_1) \cdots (x_1$ phase 1) and to the y-phase $(x_{p_2}, y_{p_2}) \cdots (x_{p_2+q_2}, y_{p_2+q_2})$ (for phase 2), we obtain

$$\begin{aligned} \mathcal{A}(S_{p_2}) - \mathcal{A}(S_1) &\leq 2(\text{OPT}(S_{p_2}) - \text{OPT}(S_1)) + |x_1| + 3|y_{p_2}|, \\ \mathcal{A}(S_{p_3}) - \mathcal{A}(S_{p_2}) &\leq 2(\text{OPT}(S_{p_3}) - \text{OPT}(S_{p_2})) + |y_{p_2}| + 3|x_{p_3}|. \end{aligned}$$

Similarly

$$\begin{aligned} &\mathcal{A}(S_{p_4}) - \mathcal{A}(S_{p_3}) \leq 2(\text{OPT}(S_{p_4}) - \text{OPT}(S_{p_3})) + |x_{p_3}| + 3|y_{p_4}|, \\ &\cdots\cdots \\ &\mathcal{A}(S_{p_k}) - \mathcal{A}(S_{p_{k-1}}) \leq 2(\text{OPT}(S_{p_k}) - \text{OPT}(S_{p_{k-1}})) + |y_{p_{k-1}}| + 3 \end{aligned}$$

Since $\mathcal{A}(S_1) = \text{OPT}(S_1) = |x_1|$, again by summing up these inequalities we obtain

$$\mathcal{A}(S_{p_k}) \leq 2\text{OPT}(S_{p_k}) + 4(|y_{p_2}| + |x_{p_3}| + \cdots + |y_{p_{k-1}}|) + 3|x_{p_k}|.$$

If $S_n = S_{p_k}$, namely, if S_n is finished with an axis-change move of the crew, th can be written as

$$\begin{aligned} \mathcal{A}(S_n) &\leq 2\text{OPT}(S_n) + 4(|y_{p_2}| + |x_{p_3}| + \cdots + |y_{p_{k-1}}|) + 3|x_{p_k}| \\ &\leq 2\text{OPT}(S_n) + 4\text{OPT}(S_n) + 3|x_{p_k}|, \end{aligned}$$

by using the inequality (**). Since the final axis-change is from the Y-axis to the X which implies

$$\mathcal{A}(S_n) \leq 9\text{OPT}(S_n).$$

If there are some moves on the X-axis after the final axis-change, then we must con

(1) If $\text{OPT}^X \leq \text{OPT}^Y$ at the end, then

$$\begin{aligned}\mathcal{A}(S_n) &\leq \mathcal{A}(S_{p_k}) + \text{OPT}(S_n) - \text{OPT}(S_{p_k}) \\ &\leq 2\text{OPT}(S_{p_k}) + 4(|y_{p_2}| + |x_{p_3}| + \cdots + |y_{p_{k-1}}|) + 3|x_{p_k}| + \text{OPT}(S_n) - \text{OPT}(S_{p_k}) \\ &\leq \text{OPT}(S_{p_k}) + 4\text{OPT}(S_n) + 3|x_{p_k}| + \text{OPT}(S_n) \\ &\leq 9\text{OPT}(S_n),\end{aligned}$$

by using (†), (**), $\text{OPT}(S_{p_k}) \leq \text{OPT}(S_n)$ and $|x_{p_k}| \leq \text{OPT}(S_n)$. (The last inequality holds since $\text{OPT}^X \leq \text{OPT}^Y$ at this moment.)

(2) Otherwise, i.e., if $\text{OPT}^X > \text{OPT}^Y$ at the end, then we can obtain the following inequality instead of (**):

$$\text{OPT}(S_n) \geq |x_1| + |y_{p_2}| + \cdots + |y_{p_{k-1}}| + |x_{p_k}|, \qquad (‡)$$

since $\text{OPT}(S_n) - \text{OPT}(S_{p_k}) \geq |x_{p_k}|$ (proof is omitted but is very similar to the proof of Lemma 7). Also, since $\text{OPT}^X(S_n) \leq \text{OPT}^Y(S_n) + |y_n|$ we can show that

$$\mathcal{A}(S_n) \leq \mathcal{A}(S_{p_k}) + \text{OPT}(S_n) - \text{OPT}(S_{p_k}) + |y_n|,$$

which implies that

$$\begin{aligned}\mathcal{A}(S_n) &\leq 2\text{OPT}(S_{p_k}) + 4(|y_{p_2}| + |x_{p_3}| + \cdots + |y_{p_{k-1}}|) + 3|x_{p_k}| + \text{OPT}(S_n) - \text{OPT}(S_{p_k}) + |y_n| \\ &\leq 2\text{OPT}(S_{p_k}) + 4(|y_{p_2}| + |x_{p_3}| + \cdots + |y_{p_{k-1}}|) + 4|x_{p_k}| + \text{OPT}(S_n) - \text{OPT}(S_{p_k}) + |y_n| \\ &\leq 2\text{OPT}(S_{p_k}) + 4\text{OPT}(S_n) + \text{OPT}(S_n) - \text{OPT}(S_{p_k}) + |y_n| \\ &\leq 2\text{OPT}(S_{p_k}) + 4\text{OPT}(S_n) + \text{OPT}(S_n) + |y_n| \leq 8\text{OPT}(S_n),\end{aligned}$$

by using (‡), (**) and $|y_n| \leq \text{OPT}(S_n)$. (The last inequality holds since $\text{OPT}^X > \text{OPT}^Y$ at the end.) ∎

References

[AHD83] A. V. Aho, J. E. Hopcroft amd J. D. Ullman. "Data Structures and Algorithms", Reading, Mass.: Addison-Wesley, 1983.

[BE98] A. Borodin and R. El-Yaniv. "Online Computation and Competitive Analysis", Cambridge University Press, 1998.

[CL91] M. Chrobak and L. Larmore. "An optimal on-line algorithm for k-servers on trees", In SIAM Journal on Computing, 20(1):144-148, February 1991.

[CLR90] T. H. Cormen, C. E. Leiserson and R. L. Rivest. "Introduction to Algorithms", Cambridge, Mass.: MIT Press, 1990.

[FR94] A. Fiat and M. Ricklin. "Competitive algorithms for the weighted server problem", In Theoretical Computer Science, 130(1) pages 85-99, 1994.

[FRR90] A. Fiat, Y. Rabili and Y. Ravid. "Competitive k-server algorithm", In 31st IEEE Annual Symposium on Foundations of Computer Science, (FOCS '90), pages 454-463, October 1990.

[Kar92] Richard M. Karp. "On-Line Algorithms Versus Off-Line Algorithms: How Much is it Worth to Know the Future?", Proc. IFIP 12th World Computer Congress, Vol.1, pp.416-429, 1992.

[KT00] E. Koutsoupias, D. S. Taylor. "The CNN Problem and Other k-Server Variants", Proc. 17th Annual Symposium on Theoretical Aspects of Computer Science, 2000.

[KP94] E. Koutsoupias and C. Papadimitriou. "On the k-server conjecture", In Proc. 26th Symposium on Theory of Computing, (STOC '94), pages 507-511, 1994.

[MMS90] M. S. Manasse, L. A. McGeoch and D. D. Sleator. "Competitive algorithms for server problems", Journal of Algorithms, 11:208-230, 1990.

[ST85] D. D. Sleator and R. E. Tarjan. "Amortized efficiency of list update and paging rules", Communications of the ACM, 28(2):202-208, 1985.

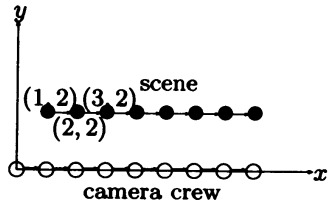

Fig. 1

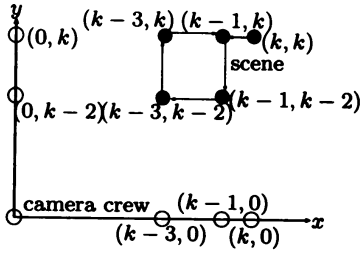

Fig. 2

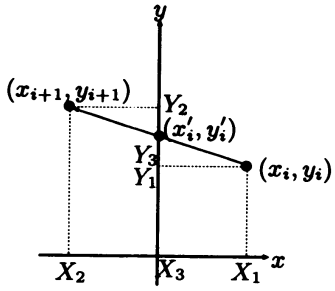

Fig. 3

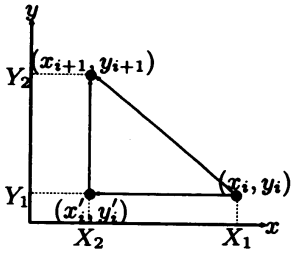

Fig. 4

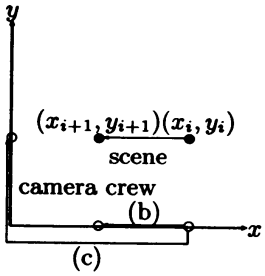

Fig. 5

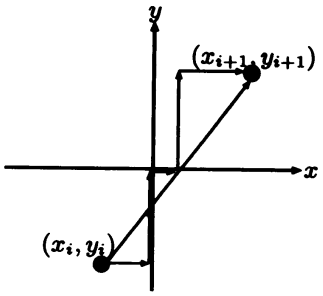

Fig. 6

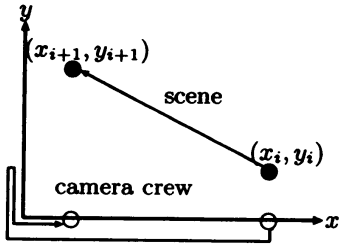

Fig. 7

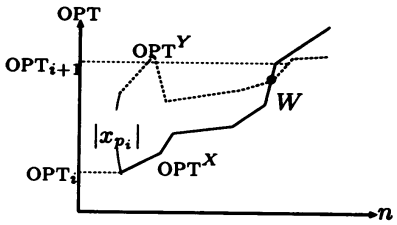

Fig. 8

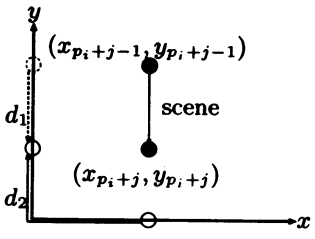

Fig. 9

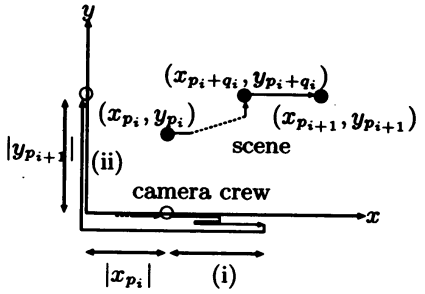

Fig. 10