The CNN problem and its variants: A Survey

Chopra, Ishan
choprai@myumanitoba.ca

Khokhar, Arsh
khokharm@myumanitoba.ca

28th December 2020

Abstract

In this Survey Paper on the CNN problem and its variants, we discuss the origins of the CNN problem and some of it's interesting variants, namely the *continuous* CNN problem and the *orthogonal* CNN problem. These problems are related to the k-server and generalized k-server problems, and are studied because they provide useful insight into how researchers should or shouldn't approach the k-server and generalized k-server problems. The CNN problem is named after the setting it models. Imagine a Certain New Network crew in a grid-like city such as Manhattan. This crew is shooting scenes which happen in an online manner at some intersections. This news crew with a powerful camera must be on a matching street or avenue to shoot a scene. This is the CNN problem. The Orthogonal CNN problem restricts the movement of the scenes such that each new scene happens on the same street or avenue as the previous scene. We provide a summary of the existing works for these problems in the normal setting and the setting with advice and we discuss some of the well known existing algorithms and bounds. We also discuss some interesting and approachable technical results from the literature on these problems: A lower bound of 3 for the orthogonal and continuous CNN problems, and a 9-competitive algorithm for the orthogonal CNN problem.

1 Introduction

1.1 The k-server problem

The k-server problem, first introduced by Manasse et. al [1], is one of the more popular problems currently studied in online algorithms. In the same paper, Manasse et. al provide the famous *k-server conjecture* which states that there is a deterministic k-competitive algorithm for the k-server problem for every metric space. So far, this conjecture has only been proven to be true for some metrics like trees [2] but it is conjectured that the Work Function Algorithm [3] is k-competitive for any metric. The k-server problem is a generalization of many online problems such as paging and caching [4].

Definition 1 (The k-server problem)**.** *In the k-server problem, given a metric space M consisting m nodes and k servers such that $m > k$, the input is an online sequence of requests to the nodes of the metric space, which need to be served by moving any one of the k servers. An online algorithm is given the current location of the servers and the sequence of requests on M, and the objective is to minimize the total distance travelled by the servers while serving the requests.*

1.2 The Generalized k-server problem

The generalized k-server problem, which was introduced by introduced by Koutsoupias and Taylor [5], extends the k-server problem even further by allowing each of the k servers to have its own metric space. Formally,

Definition 2 (The generalized k-server problem)**.** *In the generalized k-server problem, there are k servers, and each server moves on a metric space independent of the other servers. Given k metric spaces $M_1, M_2, ...M_k$, each request is a set of points $\{x_1, x_2, ...x_k\}$ such that $x_i \in M_i$. To serve a request an algorithm must move at least one server to the requested point in that server's metric space. An online algorithm is given the current location of the servers and a series of requests appear in an online manner. The objective is to minimize the total distance travelled by the servers while serving all the requests.*

The k-server problem is a special case of the generalized k-server problem where all the metric spaces are the same ($M = M_1 = M_2 = ... = M_k$) and each request is to one point ($x = x_1 = x_2 = ... = x_k$). The general lower bound known for the generalized k-server problem is $2^k - 1$ [5].

2 Problem Description

2.1 The CNN problem

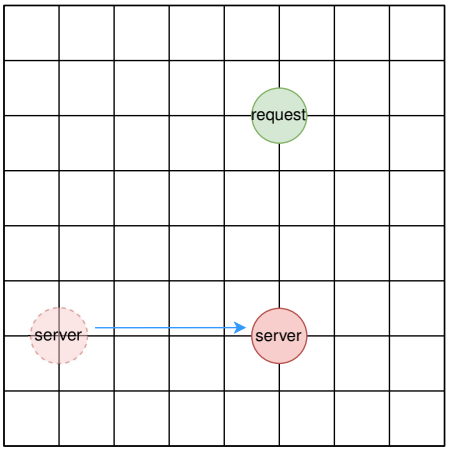
Assume that we have a grid-like city such as Manhattan, where events are occurring in an online sequence. A camera crew has to shoot these events, which occur at street intersections. The crew can shoot an event by aligning itself to either the horizontal or vertical axis of the event location. This problem is known as the CNN problem and has the following constraints:

- The location of an event is described as (i, j), where i and j are integers representing the row and column number on the grid, respectively.
  - To shoot an event, the crew must be at grid position (i, j') or (i', j), where i' and j' can be any integers.
- The goal is to minimize the distance travelled by the news crew while shooting these events serially.

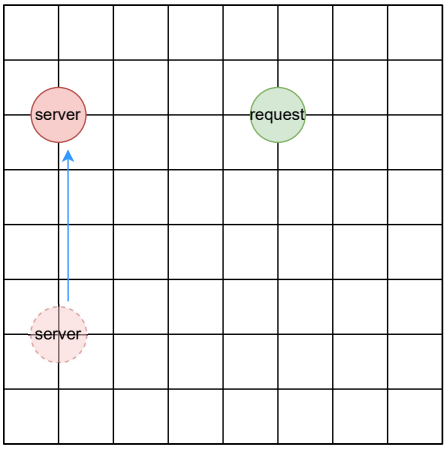
The CNN problem is a special variant of the generalized k-server problem where $k = 2$ and both metrics are paths. This is equivalent to having 1 server moving on a grid, where the server can make vertical or horizontal moves to either horizontally or vertically align itself with the current request, as shown in the figures 1a and 1b. As such, this problem is also referred to as a sum of two 1-server problems where both metric spaces are paths [5].

2.2 Orthogonal CNN problem

For our paper, we consider a special variant of the CNN problem known as the *Orthogonal* CNN problem. This variant places an additional constraint that every event must be either x-aligned or y-aligned with the previous event.



(a) Server moving to vertically align itself with the current request



(b) Server moving to horizontally align itself with the current request

Figure 1: Server serving the request in the CNN problem

Definition 3. *The CNN problem is* ***orthogonal*** *if* $x_i = x_{i+1}$ *or* $y_i = y_{i+1}$ *for all pairs of consecutive requests* (x_i, y_i) *and* (x_{i+1}, y_{i+1}).

2.3 Continuous CNN problem

The *Continuous* CNN problem, introduced by Augustine and Garvin [6], generalizes the orthogonal CNN problem. In this version, the requests are not restricted to be appearing on the grid as discrete points, and they can move continuously in $\mathbb{R}^2$, and the server must serve the moving request continuously by aligning itself with it either horizontally or vertically.

3 Summary of Existing Works

Sitters, Stougie, and de Paepe [7] presented a 10^5-competitive algorithm for the general 2-server problem in 2003. Sitters and Stougie [8] later improved this by presenting a 879-competitive algorithm for the general 2-server problem in 2006. This is the best known deterministic online algorithm for the general 2-server problem. Both these algorithms are for the general 2-server problem and therefore, also apply to the CNN problem since the CNN problem is just a special case of the general 2 server problem when both metric spaces are paths. This 879-competitive algorithm is the best existing algorithm for both the general 2-server problem and the CNN problem. The best known lower bound for the CNN problem is $6+\sqrt{17}$ [5], so there is a big gap between the best known upper and lower bound for this problem.

3.1 Orthogonal CNN problem

Iwama and Yonezawa [9] introduced the orthogonal CNN problem as well as a 9-competitive algorithm presented above. Based on their work, Augustine and Gravin [6] introduced the Continuous CNN problem which is a generalization of the orthogonal CNN problem where the request can move along any line, not just the grid. They presented a 6.46-competitive algorithm for the continuous CNN problem called the Bishop-Rook algorithm. This algorithm is naturally 6.46-competitive for the orthogonal CNN problem as well. They also presented a lower bound of 3 for the orthogonal CNN problem, which naturally applies to the Continuous CNN problem as well.

Until recently, the Bishop-Rook algorithm was the best known online algorithm for the orthogonal CNN problem. But recently, Naib [10] presented a 5-competitive algorithm for the orthogonal CNN problem called the Walk-Jump Algorithm. It is conjectured to have a competitive ratio of 3, which would make the lower bound of 3 tight.

3.2 Advice

Naib [10] showed that an advice of m bits is sufficient to achieve an optimal solution for the CNN problem, where $m \leq n$. Here, m is the number of times OPT moves to serve all the requests and n is the number of requests. A simple proof of this is presented below.

Proof. Similar to Lemma 2 and Corollary 2, we know that there is an optimal algorithm that only moves horizontally xor vertically when it needs to move. So each of its moves (horizontal or vertical) can be encoded with 1 bit. For a sequence of n requests, an optimal algorithm that only moves orthogonally would move m times, where $m \leq n$. An oracle can encode these moves with m bits. □

This upper bound holds for the Orthogonal and Continuous CNN problems as well, since they are just special cases of the CNN problem. Naib used a reduction from Binary guessing to show that in some cases, $\Theta(n)$ bits of advice is required for an optimal solution. Therefore, $\Theta(n)$ bits of advice are required and sufficient to achieve an optimal solution for the CNN problem and its variants.

4 Some Technical results

4.1 The lower bound for the orthogonal CNN problem

Lemma 1. *[6] Any c-competitive algorithm for the continuous CNN problem is also c-competitive for the orthogonal CNN problem.*

Proof. From the definition of the orthogonal CNN problem, we know that any two consecutive requests are either horizontally or vertically aligned with each other. i.e. either the x or the y coordinate is the same. Let $\sigma_{ortho} = \langle p_1, p_2, ... \rangle$ be the adversarial sequence for an instance of the orthogonal CNN problem. Now, for any incoming request p_i for the given sequence, we can construct a request (p_{i-1}, d_{i-1}), where $d_{i-1} = \frac{p_i - p_{i-1}}{|p_i - p_{i-1}|}$ [6]. Clearly, for any sequence $\sigma_{ortho} = \langle p_1, p_2, .. \rangle$, we can construct a corresponding sequence $\sigma_{continuous} = \langle (p_1, d_1), (p_2, d_2), ... \rangle$, which is the adversarial

sequence for the continuous CNN problem. Since the request trajectories are the same in both cases, it can be concluded that any c-competitive algorithm for the continuous CNN problem is also c-competitive for the orthogonal CNN problem. □

Definition 4 (The orthogonal unit CNN problem)**.** *[6] In the orthogonal unit CNN problem, the sequence of requests $\sigma_{unit} = \langle r_1, r_2, ... \rangle$ appear in an online manner as vertices of the unit square, where each request r_i can be served by aligning the server either horizontally or vertically to the requested vertex. Each horizontal or vertical move of a server incurs a cost of 1, and a move consisting of both the horizontal and vertical components incurs a cost of 2.*

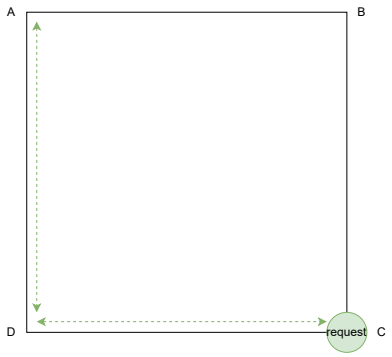
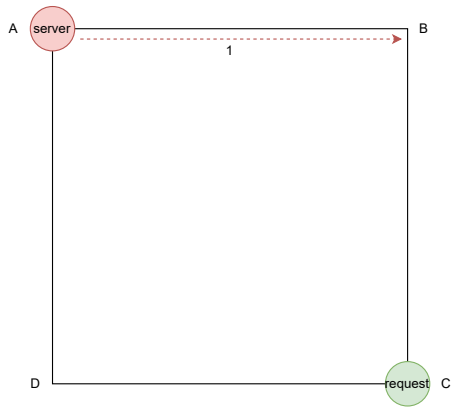


Figure 2: L-shaped trajectory of the requests

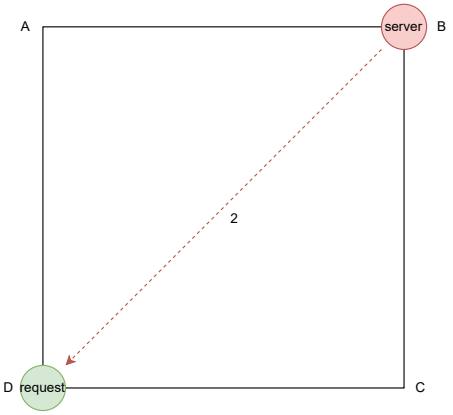
Theorem 1. *[6] For any c-competitive deterministic online algorithm for the orthogonal CNN problem, $c \geq 3$.*

Proof. Consider the unit square in the figure 2, where initially both the OPT and the ALG have placed their servers at the node A before serving any request. Now, since the requests at B and D can be served without moving any server, a cruel adversary presents the first request at vertex C. For serving the first request, any algorithm can move the server to either vertex D or B. Without loss of generality, assume that ALG moves its server to B by paying a cost of 1, as shown in the figure 3a. A cruel adversary sends further requests on the L-shaped trajectory, and with an additional constraint of discrete orthogonality, the requests appear on the diagonally opposite vertices to the server, as shown in the figure 2. So, the request sequence is $\sigma = \langle C, D, A, D \rangle^n$. For ALG to serve the sequence with a constant cost, ALG must move its server to D by paying an additional cost of 2 as shown in the figure 3b. After placing the server at D, ALG can serve any subsequent request of the sequence without paying any cost. Meanwhile, OPT moves the server to D by paying a cost of 1 for serving the initial request to C (as shown in figure 4), after which it pays no additional cost, which [6] refers to as "sweet spot". Since $cost(ALG) = 3$ and $cost(OPT) = 1$, it follows that the competitive ratio of any deterministic online algorithm is at least 3. □

Corollary 1. *[6] The lower bound for any deterministic online algorithm for the continuous CNN problem is 3.*

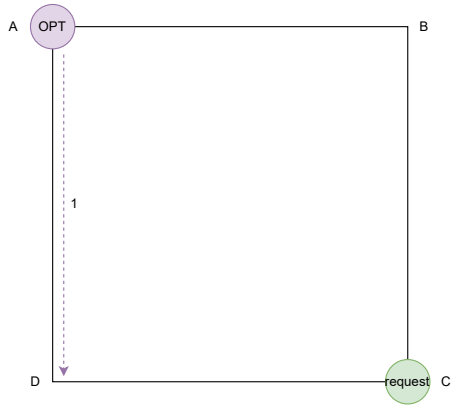


(a) ALG moving the server to B to serve the initial request

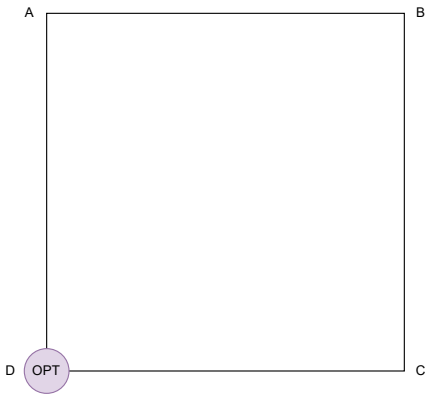


(b) ALG paying additional cost to reach the "sweet spot"

Figure 3



(a) OPT moving its server to D for serving the initial request



(b) OPT can serve the remaining requests without moving

Figure 4

4.2 A 9-competitive Algorithm for the Orthogonal CNN problem

Lemma 2. *Any optimal algorithm can be converted into an "orthogonal" optimal algorithm (that moves either vertically xor horizontally to serve a request) without increasing its cost.*

Proof. For any move of the algorithm, the non-orthogonal part of the move can be delayed. This is the same as how any non-lazy k-server algorithm can be converted into a lazy algorithm [4]. That applies here if we consider the case of 2 servers moving independently on paths. When only one of those 2 servers on paths is moved for each request, the single server on the grid moves in an "orthogonal" manner. □

Corollary 2. *There is an optimal algorithm that moves the server in an orthogonal manner to serve requests. That is, for each time it needs to move to serve a request, it moves the server either horizontally or vertically, but not both.*

Definition 5 (Knight Algorithm, 9-competitive for Orthogonal CNN)**.** *Iwama and Yonezawa [9] presented the following 9-competitive algorithm (Knight Algorithm) for the CNN problem:*

If the request is initially served by being x-aligned and the next request is horizontally aligned with the previous request (we wouldn't need to move if the request was vertically aligned with previous request, since we would remain x-aligned with the new request), move the shortest distance possible in a knight pattern (for every unit along x direction, move 2 units along y direction) to serve it.

The same applies when the request is initially served by being y-aligned. In this case, x/y and vertical/horizontal in the statement above are swapped.

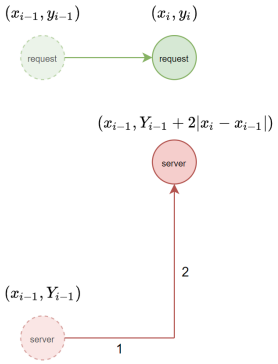
Figures 5a and 5b illustrate how this algorithm works when request moves horizontally. If the server was previously y-aligned, it wouldn't need to move to serve it, so the figures only show the case when it was previously x-aligned. It moves in a manner such that it travels twice the distance along the y-direction than what it travels along the x-direction. This is similar to how a knight moves on a chess board. It ends up with either the case in Figure 6 or Figure 7, whichever is shorter. *(The case when the request moves vertically is similar but with x/y and horizontal/vertical swapped.)*

Here is the terminology that helps in understanding the Figures above and the figures that will be presented below.

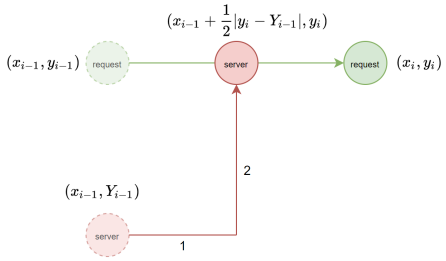
The previous request is at (x_{i-1}, y_{i-1}) and the current request is at (x_i, y_i). For this request, let the move of the optimal offline crew be from (X^*_{i-1}, Y^*_{i-1}) to (X^*_i, Y^*_i). By Corollary 2, we know that there is an optimal offline algorithm that moves the server orthogonally only. So for this optimal algorithm, we know that $X^*_{i-1} = X^*_i$ or $Y^*_{i-1} = Y^*_i$.

For the same request, let the online crew move from (X_{i-1}, Y_{i-1}) to (X_i, Y_i) to serve it.

Theorem 2. *[9] The Knight algorithm has a competitive ratio of at most 9.0*



(a) Server is sufficiently far from the request in the y-direction



(b) Server is close enough in the y-direction that it can serve the request by being y-aligned

Figure 5

Proof. The Potential function Φ_i is defined as 3 times the the Manhattan Distance (L_1 norm) between the online crew and the optimal offline crew.

$$\Phi_i = 3(|X_i - X_i^*| + |Y_i - Y_i^*|)$$

Further, we define the following:

$$OPT_i = |X_i^* - X_{i-1}^*| + |Y_i^* - Y_{i-1}^*|$$

$$ALG_i = |X_i - X_{i-1}| + |Y_i - Y_{i-1}|$$

$$\Delta\Phi_i = \Phi_i - \Phi_{i-1}$$

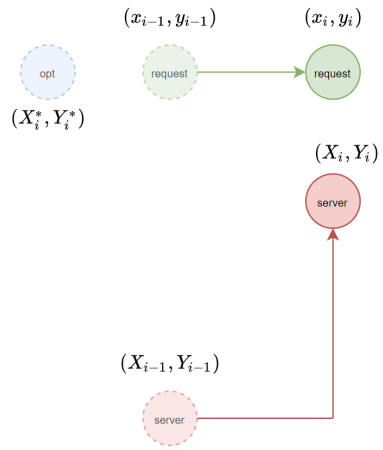
The amortized cost a_i is simply

$$a_i = ALG_i + \Delta\Phi_i$$

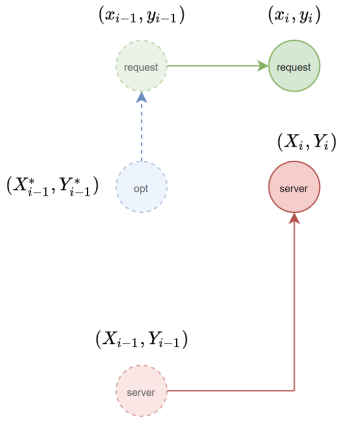
It's obvious that $\Phi_i \geq 0$ for any $i \geq 1$. To prove that the CR is at most 9, it remains to show that $a_i \leq 9 * OPT_i$. This is because summing this inequality for all i implies that $\sum ALG_i \leq 9 * \sum OPT_i$.

In the case when the online crew doesn't need to move to serve a request, $ALG_i = 0$, and all the change in potential is due to the move by OPT. So $a_i \leq 3 * OPT_i \leq 9 * OPT_i$. Let's consider the other 2 cases when the online algorithm does have to move. It can either serve the next request by being x-aligned or being y-aligned. We only show the cases when Online crew is initially x-aligned with the request ($x_{i-1} = X_{i-1}$). The cases for when Online Crew is y-aligned are the same. OPT can serve a request by being x-aligned or y-aligned. This gives us 2 cases:

Case 1: OPT serves request by being y-aligned. $Y_i^* = y_i$.
In this case, the move by OPT increases distance between its crew and the online crew by at most OPT_i. The y-direction move of ALG decreases the distance by $\frac{2}{3}ALG_i$ (since for every 1 unit along x direction, ALG moves the server 2 units along y-direction). While the x-direction move



(a) Case 1a: OPT's server already y-aligned with the request



(b) Case 1b: OPT moves the server to y-align it with the request

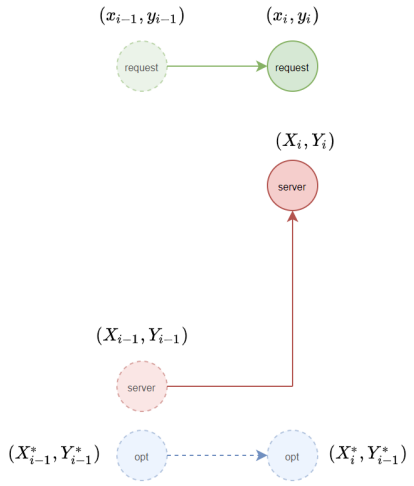
Figure 6

increases distance by at most $\frac{1}{3}ALG_i$. The 2 scenarios for this case are shown in Figure 6. So over the distance is changed by at most $OPT_i - \frac{2}{3}ALG_i + \frac{1}{3}ALG_i$. The change in potential is $\Delta\Phi_i \leq 3 * (OPT_i - \frac{2}{3}ALG_i + \frac{1}{3}ALG_i) = 3 * OPT_i - ALG_i$. This gives us an amortized cost of

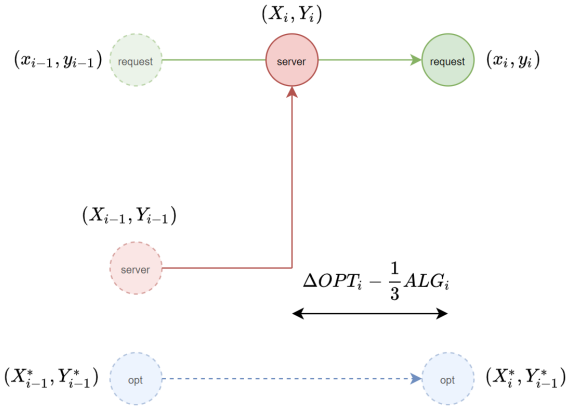
$$\begin{aligned} a_i &= ALG_i + \Delta\Phi_i \\ &\leq ALG_i + (3 * OPT_i - ALG_i) \\ &= 3 * OPT_i \\ &\leq 9 * OPT_i \end{aligned} \tag{1}$$

Case 2: OPT serves the request by being x-aligned. $X_i^* = x_i$.
We also get that $X_{i-1}^* = x_{i-1}$ since otherwise the optimal offline crew would have been y-aligned with the initial request ($Y_{i-1}^* = y_{i-1}$), which means it can serve the new request by being y-aligned as well ($y_i = y_{i-1}$), which puts us in Case 1 (Figure 6a). Since we are not in case 1, we know that $Y_i^* = Y_{i-1}^* \neq y_i$, ie. it serves the request by being x-aligned and not y-aligned. Both the $(i-1)$st and ith request are served by OPT by being x-aligned. So $OPT_i = |x_i - x_{i-1}|$ since OPT moves serves the request by being x-aligned and moves from (x_{i-1}, Y^*) to (x_i, Y^*), where Y^* is some value.

Now the algorithm moves the server in a knight pattern to serve the request (1 unit along x-direction, 2 units along y-direction). So the cost of the algorithm for this step is
$ALG_i \leq 3|x_i - x_{i-1}| = 3 * OPT$. Online crew's move along the y-direction increases the distance between the crews by $\frac{2}{3}ALG_i$. Due to OPT's move and Online Algorithm's move along the x-direction, the distance between them increases by at most $OPT_i - \frac{1}{3}ALG_i$. The 2 scenarios for this case are shown in figure 7. The change in potential is at most $\Delta\Phi_i = 3 * ((OPT_i - \frac{1}{3}ALG_i) + \frac{2}{3}ALG_i)$. This gives us an amortized cost of



(a) Case 2a: OPT and ALG both move their servers to x-align with the request



(b) Case 2b: OPT moves the server to x-align with the request while ALG moves to y-align with the request

Figure 7

$$
\begin{aligned}
a_i &= ALG_i + \Delta\Phi_i \\
&= ALG_i + 3 * ((OPT_i - \frac{1}{3} ALG_i) + \frac{2}{3} ALG_i) \\
&= ALG_i + 3 * OPT_i + ALG_i \\
&= 2 * ALG_i + 3 * OPT_i \\
&\leq 6 * OPT_i + 3 * OPT_i \\
&= 9 * OPT_i
\end{aligned}
\tag{2}
$$

In both cases, the amortized cost at time i is bounded by 9 times the cost of the optimal algorithm at time i. This proves that the Competitive Ratio of the Knight algorithm for the Orthogonal CNN problem is at most 9. This bound is known to be tight.

□

5 Concluding remarks

We reviewed the existing works related to the CNN problem and some of its variants in the normal deterministic setting as well as the setting with advice. We also presented some interesting technical results that we found, including a lower bound of 3 for the orthogonal and continuous CNN problems as well as a 9-competitive algorithm for the orthogonal CNN problem.

References

[1] Mark S Manasse, Lyle A McGeoch, and Daniel D Sleator. Competitive algorithms for server problems. *Journal of algorithms*, 11(2):208–230, 1990.

[2] Marek Chrobak and Lawrence L. Larmore. An optimal on-line algorithm for k servers on trees. In *SIAM Journal on Computing*, volume 20, pages 144–148. SIAM, 1991.

[3] Elias Koutsoupias and Christos Papadimitriou. On the k-server conjecture. In *Journal of the ACM (JACM)*, volume 42, pages 971–983. ACM, 1995.

[4] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 1998.

[5] Elias Koutsoupias and David Scot Taylor. The cnn problem and other k-server variants. *Theoretical computer science*, 324(2):347–359, 2004.

[6] John Augustine and Nick Gravin. On the continuous cnn problem. In *Algorithms and Computation*, Lecture Notes in Computer Science, pages 254–265. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[7] René Sitters, Leen Stougie, and Willem de Paepe. A competitive algorithm for the general 2-server problem. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 624–636, 2003.

[8] René Sitters and Leen Stougie. The generalized two-server problem. In *Journal of the ACM(JACM)*, pages 437–458, 2006.

[9] Kazuo Iwama and Kouki Yonezawa. The orthogonal cnn problem. *Information Processing Letters*, 90:115–120, 2004.

[10] Sameer Naib. Alternative settings and models of the certain news network problem. Master's thesis, University of Manitoba, 2020.