# Design of Efficient Online Algorithms for Server Problems on Networks



*Dissertation Abstract*

# Design of Efficient Online Algorithms for Server Problems on Networks

**Amanj Khorramian**

khorramian@gmail.com
+98-918-974-3123 (IRAN)

31 December 2017

**Abstract:** Shared resources reside on server nodes in a network. Requests arrive separately in succession to access the resources. Several problems are unavoidable in managing the server locations in order to reduce the load of network. There is no way to apply optimal algorithms, because the future data are unknown. The design of online algorithms to efficiently solve these problems is the topic of our study. We introduce the bases of online problems as well as the systematic approaches to finding solution. After that, we have a survey about the server problems and design techniques with their algorithmic qualities in terms of competitiveness in basic networks. Two specific networks of Euclidean space and rings are focused, independently. The server problem of migrating a unit data upon access requests is enquired in both networks towards designing deterministic algorithms. For the case of Euclidean space, an efficient 2.75-competitive online algorithm is designed and equipped by a lower bound of 2.732. It improves the former 2.8-competitive and 24-year-old algorithm. As for general ring networks, a 3.326-competitive algorithm is achieved with a tight analysis. Then, we propose time-efficient definitions of server problems which arise in modern networks.

**Keywords:** server problems; uniform page migration problem; competitive analysis; online algorithms; efficient design; adversary model; euclidean space; ring networks

---

## 1. Introduction

Daily-life is greatly influenced by electronic devices. The desired functionality of these appliances depends on their processing units, and the needs for faster processors has been increased. Since there is no recent big improvement to the clock speed of processors, one resolution is to interconnect multiple processors in a network which shares resources which cause several problems. The study of networks has been drawing more attention, continually. The power of computation and computing resources is essential for coping with so many challenges which arise. Efficient solutions for big networks are demanding since they are enlarging and coming into view of human life. In practical point of view, PCs have ability with a reasonable cost for tackling the problems with some scale. More power of computing for larger problems can be found through high-performance computing, cloud computing, grid computing, distributed processing, etc. In theoretical point of view, the network problem is defined formally, and efficient techniques are employed to design an algorithm which guarantees a level of optimality through comprehensive analysis. The problems are subject to be divided into two or more categories. For each category, some specific frameworks are known to be utilized for dealing with the problem. For example, if the whole data of the problem are given in advance, then the scope of design would be *offline algorithms*. Otherwise, i.e., the input data are given one by one, then the problem needs to be coped with through *online algorithms*. An algorithm might be *deterministic* or *randomized*, depending on the behavior of algorithm to be either functional or based on random distribution, respectively. For complexity, the amount of computation to solve a problem could be linear, constant, polynomial, exponential, etc. Since the exponential-time algorithms are less practical, there are helpful approaches like approximation algorithms, which give solution with a guarantee of maximum gap from the optimal solution, but applicable in a reasonable time. Similar situation happens also for online algorithms, which are the theme of our study. A deterministic online algorithm guarantees a maximum gap with the optimal solution. Moreover, we focus on the *server problems* of networks, for which sequential requests to access shared resources appear one by one, and must be served per se.

The main problem of *online problems* is that there is partial knowledge about the whole data. The entire data are not available at once, but coming one by one. The solution must adapt with a new input arrival. Each input may be interpreted as a request of task and is likely to change the state of solution. The final goal is to reduce the total cost of processing the inputs and changing the states, as far as possible. In the case of online problems, *metrical task systems* [1] have been widely utilized as a formal framework for defining online problems. This model is established based on a network of

metric space $M = (S, \gamma)$ for $\gamma: S \times S \to \mathbb{R}$, where $|S|$ denotes the finite number of network states. $(M, T)$ is a metrical task system (MTS), where $T$ is the set of tasks (abstractions of requests). Each task is denoted by $r_i = (r_i(s_1), r_i(s_2), \dots, r_i(s_{|S|}))$, where $i \geq 1$, and $r_i(s_j) \geq 0$ is the cost of processing task $r_i$ at state $s_j$. In addition, $s_0 \in S$ is set as a starting state of the problem for dealing with. A metrical task system $((S, \gamma), T)$ is called *uniform* [1] if $\gamma(s_i, s_j)$ is identical for every $s_i, s_j \in S$ where $s_i \neq s_j$. An online algorithm $A$ sets $A_0 = s_0$, and processes a request sequence $\sigma = r_1, r_2, \dots, r_m$ one by one, while determining a state sequence $A_1, A_2, \dots, A_m$, where $A_i \in S$ denotes the state at which $r_i$ is processed. Upon each request $r_i$, the algorithm first changes the state of solution from $A_{i-1}$ to $A_i$, and then processes request $r_i$. The objective of an online algorithm is to lessen the total cost for changing the states and processing $m$ requests $A(r_1, r_2, \dots, r_m) = \sum_{i=1}^{m} (\gamma(A_{i-1}, A_i) + r_i(A_i))$ as far as possible. In this way, we denote $A(r_1, r_2, \dots, r_m)_{s_j}$ as the total cost if $A_m = s_j$. Note that if $A_i = A_{i-1}$, then we have $\gamma(A_{i-1}, A_i) = 0$, i.e., no change occurs to the state upon request $r_i$. The quality of an online algorithm is measured relative to the all other possible algorithms. The algorithms $U$ and $V$ are *deterministic* since they uniquely decide the state upon each request. On the other hand, *randomized algorithms* distribute probability on the states for random choices, upon each request. The framework of *competitive analysis* is available to evaluate the efficiency of an online algorithm in a breathtaking comparison with an optimal algorithm which already possesses the request sequence, in advance. This measurement yields a value $c$ for an online algorithm $A$. This value is calculated by dividing the cost incurred by $A$ over the cost of any other offline algorithm, for any sequence of requests. The maximum ratio can be obtained using a comparison between the online algorithm $A$ and an optimal algorithm $X$. An online algorithm $A$ is *$c$-competitive*, if there exists a constant $\alpha$, such that $A(\sigma) \leq cX(\sigma) + \alpha$, for any finite sequence $\sigma$. If $\alpha = 0$ then $A$ is *strictly* $c$-competitive. If $A$ is a randomized algorithm, then $A(\sigma)$ denotes the expected value of cost. $A$ is *competitive* if it hits a constant *competitive ratio* of $c$.

## 2. Efficient Design

*Adversary models* allow to analyse online algorithms as an unfair game, between the designed online algorithm and a malicious adversary against it. The adversary knows the algorithm and produces a sequence of requests as difficult as possible (to maximize its competitive ratio). *Adaptive offline adversary* knows everything about $A$. This adversary generates a sequence of requests $\sigma$, based on the complete information about the algorithm's response for any request. Hence, at each step, it chooses the next request according to the responses of the algorithm so far. The algorithm $X$ of the adversary behaves optimally and pays the optimal cost $X(\sigma)$. *Adaptive online adversary* cannot determine the behavior of $A$, since $A$ is a randomized algorithm for this model. This adversary produces each request according to $A$'s definition and against its random responses so far, and incurs an expected cost $X(\sigma)$. *Oblivious adversary* does not know anything about the random behavior of $A$ and its responses. This adversary produces the entire sequence in advance, only based on the definition of algorithm. Regarding the adversary models, an online algorithm $A$ is $c$-competitive, if for an additive constant $\alpha$ we have $A(\sigma) \leq cX(\sigma) + \alpha$. At this point, we mention some useful relations among these models.

*Amortized Analysis* is a useful way to bound the competitiveness of an algorithm upon each request. After receiving a request $r_i$, the algorithm $A$ processes the request by some actions and responses to the request by some other actions. Similarly, the adversary $X$ also processes and responses to the request by some actions. These actions are specified by the definitions of $A$ and $X$. Let $\langle a_{i,1}, a_{i,2}, \dots, a_{i,n} \rangle$ be any order of all the mentioned actions performed upon the request $r_i$, and $C_\zeta$ be the set of all computable configurations of an algorithm $\zeta$. Moreover, let $E = \langle a_{1,1}, \dots, a_{1,n}, a_{2,1}, \dots, a_{2,n}, \dots, a_{m,1}, \dots, a_{m,n} \rangle$ be an ordered set of all actions for a sequence of $m$ requests, and $E$ is partitioned into $p$ subsets $E_j$ for $1 \leq j \leq p$, such that the actions in $E_k$ happen before the actions in $E_l$ for $k < l$. In the analysis of an online algorithm $A$ for showing its $c$-competitiveness, it would be enough to define a function $\Phi: C_A \times C_X \to \mathbb{R}$ and show that $|A|_j + \Phi_j \leq c|X|_j + \Phi_{j-1}$ for every $E_j$, where $|\zeta|_j$ denotes the total cost incurred by $\zeta$ to perform $E_j$, and $\Phi_j \geq \beta$ denotes the value of $\Phi$ right after finishing the accomplishment of $E_j$ by both algorithms, for a constant $\beta$. Set $\Phi_0$ as the

value of $\Phi$ just before performing any actions. In this analysis, $\Delta\Phi = \Phi_j - \Phi_{j-1}$ prepares an upper-bound $c$ for the algorithm, so that the art of choosing proper subsets to minimize the amortized value as well as mapping the configuration of algorithms in an appropriate way are the basic ideas of this investigation. We denote by $\Delta|\zeta| = |\zeta|_j - |\zeta|_{j-1}$ the change of value in the cost of $\zeta$ by two consequent partitions of actions. Each partition is called an event. To analyse the algorithm for second network, we utilize this method, and also unearth the initial technique of aggregation [2], which leads to combine two consecutive subsets to follow the proof by somehow disturbing the discussed approach.

*Work function algorithms* provide a standard procedure for confronting adversaries. An adversary $X$ knows the online algorithm and acts against it, thus a work function algorithm $W$ attempts to know the adversary as soon as possible and reacts against it. This defense is done through designing $W$, such that upon request $r_i$, $W_i$ recursively minimizes $W(r_0, r_1, \dots, r_{i-1})_{s_j} + \gamma(s_j, W_i) + r_i(W_i)$ for all $s_j \in S$ and $W(r_0)_{s_j} = \gamma(s_0, s_j)$. Here, $W(r_0, r_1, \dots, r_{i-1})_{s_j} = X(r_0, r_1, \dots, r_{i-1})_{s_j}$ is a *work function* which can be optimally calculated through dynamic programming. This technique of efficient design always provides an online algorithm which shows the power of this technique.

## 3. Server Problems

The *list accessing problem* is extensively studied. The objective is to minimize the total cost for accessing a requested item in an ordered list and self-reordering of the list. Reordering strategies basically include moving the accessed item to the front of the list, transposing the item with its preceding one, and sorting based on the frequency of accesses. The *page replacing problem* (a.k.a. *paging problem*) is a variant of the list accessing problem, which arises in the management of a virtual memory by an operating system. Assume the fast memory is limited to hold $k$ pages of the virtual memory. Faster response to access the pages is desirable for online requests as usual, but fetching a page from the virtual memory into the fast memory is costly. If an online algorithm drops the least recently used or the oldest fetched page from fast memory for fetching a newly requested page, then it achieves $k$-competitiveness, and no other deterministic online algorithm can achieve a better ratio [3]. The algorithm of evicting the least recently used page is recognized as a form of *marking*, a general strategy that works in phases [4]–[6]. a significant number of efficient online algorithms which work in a phase-based style for famous online problems. Hastily speaking, in each phase, a part of request sequence is being investigated with a length that is a function of some property of data (e.g. data size). The study of server problems is roughly about managing resources in networks. *$k$-server problem* and *data management problem* are extensively studied online problems that come out in the management of data in networks.

$k$-server problem [7] is a general form of page replacing problem, in which the cost of fetching different pages may differ. In a metric space $(S, \delta)$, there are $k$ copies of data on each node of $s_{i-1} \subset S$, and upon each request at $r_i \in S$, if $r_i \notin s_{i-1}$ then a copy of data on $x \in s_{i-1}$ moves to $r_i$ with the cost of $\delta(x, r_i)$, where $s_0$ is the initial set of servers. The objective is to minimize the total cost of movements. There is no $c$-competitive deterministic [7], or randomized algorithm against adaptive online adversaries [8], where $c \leq k$. It is an important open question if the work function algorithm is $k$-competitive, though its $(2k-1)$-competitiveness is known [9]. The lower bound of problem is [10] to be $\Omega(\log(k)/\log^2(\log(k))$ against oblivious adversaries, and is conjectured to be $\Theta(\log(k))$ [8]. If $|S| = k + c$, then there is a $O(c^6 \log^6(k))$-competitive algorithm against oblivious adversary [11].

Suppose there is a unique data in the network and the sequence of requests arrive either to obtain or to update the data. This data may be in the form of a file or database, and may be replicated among servers for handling read or write tasks arrive at the nodes $r_1, r_2, \dots, r_n$ in an online manner. The corresponding metric space of a task system $(S, \gamma)$ is specified in such a way that $s_i \in S$ is the set of nodes which hold the copies of data, right after serving the request at $r_i$ and before receiving a next request. $\gamma(s_{i-1}, s_i)$ is the minimum cost to move copies of data from $s_{i-1}$ to $s_i$ such that moving each copy of data from $u \in s_{i-1}$ to $v \in s_i$ costs the data size $D$ multiplied by $\delta(u, v)$, the distance between $u$ and $v$. The cost of processing $r_i$ at state $s_{i-1}$ is denoted by $r_i(s_{i-1})$. If $r_i$ is a read request, then $r_i(s_{i-1})$ equals to the distance of the closest $w \in s_{i-1}$ to $r_i$ in the network. If the network is represented by a graph $G = (V, E)$ and $r_i$ is a write request, then $r_i(s_{i-1})$ is the weight of a minimum weight tree that

spans $s_{i-1} \cup \{r_i\}$. The problem is known as *data management problem* (a.k.a. *file allocation problem*), targeted to minimize $\sum_{i=1}^{m}(r_i(s_{i-1}) + \gamma(s_{i-1}, s_i))$. If the metric space is a graph and $D = 1$, as well as if there is no 'write' request and 'read' requests always cause replication, then the problem is called *online Steiner tree problem.* The study of this restricted case of the problem has close connection with the original problem in such a way that any $c$-competitive online Steiner tree algorithm against adaptive online adversary can derive an online $(2 + \sqrt{3})c$-competitive algorithm against adaptive online adversary [12], and for every graph, if there exists a $c$-competitive algorithm on data management problem, then there exists a strictly $c$-competitive algorithm for online Steiner tree problem [12].

Data management problem has been received a substantial amount of interests. For arbitrary graphs, there exists a phase-based algorithm that considers only one part of the request sequence at each phase. Each part contains exactly $D$ write requests if there are enough, and the algorithm copes only with read requests at each phase [13]. There exists an $O(\min\{\log(|V|),\log(\max\{\delta(v_i, v_j)\})\}$-competitive algorithm on general graphs, for all $v_i, v_j \in V$ [13], and there exists a graph $G(V, E)$ such that the competitive ratio of any randomized algorithm against oblivious adversary is in $\Omega(\log(|V|))$ [12], [14]. Some other results for this problem include the lower bound of 3 for both deterministic [15] and randomized algorithms against adaptive online adversary [12], as well as $2 + 1/D$ for randomized algorithms against oblivious adversaries [16], in a network with only two points. As for ring networks, randomized algorithms against adaptive online adversary [12] and against oblivious adversary [16] combined, [17] exist, each with the competitiveness of $2(2 + \sqrt{3})$ and $2(2 + 1/D)$ respectively. In the study of outerplanar graphs, randomized algorithms with the competitiveness of $8(2 + 1/D)$ and $(3 + 2\sqrt{2})(2 + \sqrt{3})$ are designed, against oblivious adversary [16] combined, [18], and against adaptive online adversary [19], respectively. As for the uniform ring networks, there is no randomized $c$-competitive algorithm against adaptive online adversary for $c < 3.833$ [20], and in addition, if there is no 'write' requests in the network, then a lower bound of 2.311 [21] and an upper bound of 3 [22] for deterministic algorithms are proposed. Another well-studied server problem, called *page migration problem*, is a restricted variant of the data management problem, in which $|s_i| = 1$. In other words, data management problem is the same as page migration problem, if there is no 'write' request. Since here is no 'write', there is no matter about defining the problem in Euclidean and Manhattan spaces which are different from the metric space of finite graphs. Our new algorithms are concentrated on the problem of page migration (a.k.a. data migration, file migration), so that we will contemplate it with more details later.

To provide a formal definition for the page migration problem, we first summarize the notations here, which are subject to get slight changes according to each context of the article. $\sigma = r_1, r_2, \dots, r_n$, $A = s_1, s_2, \dots, s_n$, $\delta(a, b)$, $D$, $|S|$, $\Delta|A|$, and $cost_A(s_0, \sigma)$, respectively denote the sequence of first $n$ requests, the designed algorithm, the distance between two nodes $a, b \in S$, the page size, the number of nodes in network, the amount of change in cost of algorithm $A$ before and after an event, and the cost of $A$ to respond $\sigma$ for the initial server location $s_0$. Therefore, the page migration problem is to compute the servers, i.e., the sequence of page locations $s_1, \dots, s_n$, such that the objective of cost function $\sum_{i=1}^{n}\big(\delta(s_{i-1}, r_i) + D \cdot \delta(s_{i-1}, s_i)\big)$ is minimized. This problem is a formulation for the efficient management of memory shared among a network of processors, such as multiprocessor units, multicore processors, and graphical processing units. The problem can also be viewed as a formulation for the efficient handling of shared objects in the network of a distributed system, such as computer and telecommunication networks. A restricted form of the page migration problem with $D = 1$ is regarded as *uniform page migration problem* and received interests in the area. For this *uniform model* of problem, the objective is to minimize $\sum_{i=1}^{n}\big(\delta(s_{i-1}, r_i) + \delta(s_{i-1}, s_i)\big)$. The uniform page migration problem is identical to the page migration problem if the cost incurred by servicing a request on $b$ from the server $a$, equals to the cost incurred by the migration of server from $a$ to $b$ for every $a, b \in S$. The problem arises when the requests are always issued to access the entire page, or resource.

The page migration problem was reported by proposing 3-competitive algorithms which work based on counters, in uniform graphs and trees, as well as showing a lower bound of 3 for any metric

space [15], and (disproved) conjecturing the optimality of the lower bound. Note that *counter-based algorithms* play a significant role in designing efficient online algorithms for server problems. This kind of algorithms usually work with a strategy by considering counters on the nodes of network, such that the total amount of counts is bounded by a function of the page size from above. For general graphs, a 4.086-competitive algorithm is known [23] that for parameters $\alpha \approx 1.841$ and $\beta \approx 0.648$, after $j$th subsequence of fixed $\alpha \times D$ requests $r_i$, the page migrates from $s_{j-1}$ to $s_j$ that minimizes $\sum_{i=1}^{\alpha \times D} \delta(s_j, r_i) + \beta \times D \times \delta(s_{j-1}, s_j)$. Just recently, the ratio of 4.086 has been improved to 4, by a new algorithm that considers to dynamically change the length of subsequences [24]. Moreover, in the study of randomized algorithms against adaptive online adversaries, a 3-competitive algorithm for general metric spaces was proposed [25] that matches the lower bound on two points [12].

About continuous metric spaces, a randomized $(2 + 1/2D)$-competitive algorithm against oblivious adversary is proposed using work functions, and showed to be optimal, for a segment between two points. This algorithm is utilized as a module for designing a new algorithm for the network of continuous tree (a concatenation of two-point segments), while preserving its competitiveness. The new randomized algorithm migrates its server to a distribution $s_i = \sum_{j=1}^{2D} \frac{1}{2D} s_{ij}$ upon request $s_{i,1} = r_i$. The rest of $s_{i,j}$ are determined using an initial subtree $T = [s_{i,1}, s_{i-1,1}]$ that is developed as $s_{ij}$ gets the nearest point in $T$ to $s_{i-1,j}$ and $T$ grows to $T \cup [s_{i,j}, s_{i-1,j}]$; while $j$ increases from 2 to $2D$. This algorithm works even on finite products of tree such as continuous hypercubes and meshes. The algorithm is derandomized by migrating to $\overline{s_i} = \sum_x x s_i(x)$, the *barycenter* of $s_i$ while keeping the same competitive ratio of $2 + 1/2D$, as the best ratio on continuous trees. The competitiveness is admitted even in $\mathbb{R}^n$ but still under $L^1$ norm [26]. If there exists a $c$-competitive algorithm with finite distribution against oblivious adversary on $\mathbb{R}^n$, then a deterministic $c$-competitive algorithm also exists [26].

For general metric spaces, a randomized $c(D)$-competitive algorithm is available [25] against oblivious adversary, where $c(1) = 2.8$ and $c(D)$ gets smaller to approach 2.618 as $D$ enlarges. This algorithm is derandomized in $\mathbb{R}^n$ under $L^p$ norm for any $n$ and $p$. This algorithm from 24 years ago is beated by one of our contributions in this thesis, by proposing a more efficient and deterministic 2.75-competitive algorithm. We bound the ratio for the algorithm with 2.732 from below. Note that for the interval $[0, 1]$, we have $2 + 1/2D$ as a lower bound for any randomized or deterministic algorithm [26]. This lower bound is also admitted on $\mathbb{R}^n$ under any norm, because for the interval $I = [0, 1]$ on a dimension $k$ in $\mathbb{R}^n$, any online algorithm $A$ locating its server in $\mathbb{R}^n \setminus I$ for requests only in $I$ has a cost of at least that of a certain algorithm locating its server only in $I$, i.e., projection on the $k$th coordinate of $A$'s server location if the projection is in $I$, and the closer endpoint of $I$ otherwise. It was a longstanding question how the gap of $c(D)$ and $2 + 1/2D$ can be tighter under $L^p$ norm with $p \geq 2$. The question is partially answered by our algorithm. For the graphs restricted with only three points (a.k.a. three-node ring networks, three-node cycles), optimal 3-competitive deterministic algorithms with $D \in \{1,2\}$ [26], [27], and asymptotically optimal $(3 + 1/D)$-competitive deterministic algorithms with $D \geq 3$ [27] are proposed. Specifically for ring networks with more nodes (i.e. $|S| > 3$), there is no general study and the current 4-competitive upper bound [24] is still the best known for $D > 1$.

In the uniform model, this upper bound is reduced to $2 + \sqrt{2} \approx 3.4142$ which works on general graphs including the rings, together with showing a lower bound of 3.1213 for a ring with five nodes [28]. In another study, we propose a quite complicated deterministic algorithm with the tight competitiveness of 3.326 on ring networks. *Mobile server problem* is a variant of page migration problem that restricts the movement distance of the page in Euclidean space. The problem is introduced and provided by a deterministic and near-optimal algorithm that migrates the server towards the center of some requesting points [29]. Additionally, we point out that the current problems of managing resources are defined to efficiently reduce the load of networks, and we propose new definitions for reducing the time in the networks.

## 4. Euclidean Space

The uniform page migration problem in a network of Euclidean system is considered. Any point in a space with one, two, or even more dimensions, is likely to be a source for the request or the location of server. The distance function $\delta$ is defined by norm $L^2$. Each request at a point $r_i$ is served by the cost of ordinary distance between $s_{i-1}$ and $r_i$, and after that, the page may migrate by the cost of distance between $s_{i-1}$ and $s_i$, the new server location. Cloud computing authorizes the requests to access the resources which are present and found everywhere. The problem included in such recently hot topics, where a server can take the natural Euclidean distances for the purpose of reducing the overhead of management.

25 years ago, a 2.8-competitive randomized algorithm was proposed [25], and later, 24 years ago, that algorithm was derandomized to a deterministic algorithm [26] with the same ratio for this problem. No improvement is known before our study. Our algorithm PQ, maintains the server at the center of two points $p$ and $q$, both of which are initially located at the initial server location. Upon each request at location $r$, if $\delta(p,r) \geq \delta(q,r)$, then $p$ moves to $r$; otherwise, $q$ moves to $r$. The algorithm migrates its server to $s = \frac{p+q}{2}$ after $p$ or $q$ moves. The competitiveness of 2.75 for the algorithm is proved [30]. For the proof of the competitiveness, the correctness of an inequality, different from a triangle inequality, is needed. The inequality is about the points computed by the algorithm that is shown by a lemma [30], saying that for any $\rho > 2$ and $p, q, r, s \in \mathbb{R}^2$ such that $p \neq q$, $\delta(p,r) \geq \delta(q,r) > 0$, and $s$ is the center of $p$ and $q$, $g = \delta(s,r) - \left(\frac{\rho}{2} - \frac{1}{2}\right) \cdot \delta(p,r) - \left(\frac{\rho}{2} - 1\right) \cdot \big(\delta(q,r) - \delta(p,q)\big)$ is maximized if $\delta(p,r) = \delta(q,r)$, or $\delta(p,q) = \delta(p,r) + \delta(q,r)$, or $\delta(p,r) = \delta(p,q) + \delta(q,r)$.

For bounding the competitive ratio of the analysis, an adaptive offline adversary is found. For $r_1 = a$, $r_{2k} = b$, and $r_{2k+1} = s_0$ on a plane, where $1 \leq k \leq n$ and $a$, $b$, and $s$ are the vertices of an equilateral triangle with a unit side length, the adversary infinitesimally perturbs the distances by slightly increasing the distance between $s$ and $b$. In this sequence, for sufficiently large $n$, then $cost_{PQ}(s,\sigma)/cost_{OPT}(s,\sigma)$ is at least $1 + \sqrt{3} \approx 2.732$, which bounds the competitiveness of PQ from below. [30]

As for future works on this problem, one could seek to find an online algorithm to cover the non-uniform page migration problem with better competitiveness than 2.618. Another area of improvement is to narrow the upper and lower bounds of the algorithm, though we conjecture that this gap can be closed towards the lower bound. Moreover, the generalization of the algorithm under other norms remains an open problem in this research area.

## 5. Ring Network

Ring is a connected graph with exactly two paths between every couple of nodes. In the problem of uniform page migration, a server, i.e., one of the nodes, holds a non-duplicable page. Through shortest path $\pi(,)$ (with the length of less than or equal to half of network length $L$), the server must serve every request, and may migrate to another node before the next request arrives. We propose a tightly deterministic 3.326-competitive algorithm TriAct, improving 3.414, the best upper bound. We set $r_0 \leftarrow s_0$, $x \leftarrow \delta(s_{i-1}, r_{i-1})$, $y \leftarrow \delta(s_{i-1}, r_i)$, and $z \leftarrow \delta(r_{i-1}, r_i)$. Our algorithm TriAct migrates the server from $s_{i-1}$ to $s_i$ through the following 6-case algorithm:

**if** $z = x - y$ **then** $s_i \leftarrow r_i$ {Case A}
**else if** $z = y - x$ **then** $s_i \leftarrow r_{i-1}$ {Case B}
**else if** $z = x + y$ **then** $s_i \leftarrow s_{i-1}$ {Case C}
**else if** $y \geq -\frac{\rho-3}{\rho-2}x + \frac{1}{2}L$ **and** $y \geq \frac{2}{\rho}x + \frac{\rho-2}{2\rho}L$ **then** $s_i \leftarrow r_{i-1}$ {Case D}
**else if** $y \leq \frac{\rho-1}{2}x$ **and** $y \geq -\frac{\rho}{\rho-2}x + \frac{\rho}{2\rho-4}L$ **then** $s_i \leftarrow r_i$ {Case E}
**else** $s_i \leftarrow s_{i-1}$ {Case F}

If $\pi(s_{i-1}, r_{i-1})$ and $\pi(s_{i-1}, r_i)$ share any edge of the network, then either Case A or Case B follows by the algorithm. If $\pi(r_{i-1}, r_i) = \pi(s_{i-1}, r_{i-1}) \cup \pi(s_{i-1}, r_i)$ then Case C follows (see Figure 1, left side). For the rest of cases, we have $L = \delta(r_{i-1}, r_i) + \delta(s_{i-1}, r_{i-1}) + \delta(s_{i-1}, r_i) = x + y + z$, and the algorithm

separates all possible conditions among distances into three Cases D, E, and F, to decide the action of server for migration. Since $L$ is a constant value, we calculate $z$ as a function of $x$ and $y$. To show TriAct is $\rho$-competitive with $\rho \approx 3.325722333398888$, we utilize the function of amortized analysis $\Phi(s_i, r_i, t_i) = (\rho/2) \cdot \big(\delta(s_i, t_i) + \delta(r_i, t_i)\big) + (\rho/2 - 1)\delta(s_i, r_i)$, use the triangular inequality frequently, and separate the online events into two parts to show that $\Delta|\text{TriAct}| + \Delta\Phi - 3.326\Delta|\text{X}| \leq 0$ follows in every case. The first part includes the migration costs incurred by X, and the second part covers the service costs incurred by X together with the migration and service costs incurred by TriAct. Let $\Delta_1 = \Phi(s_i, r_i, t_i) - \Phi(s_i, r_i, t_{i-1}) - \rho \cdot \delta(t_{i-1}, t_i)$, and $\Delta_2 = \delta(s_{i-1}, r_i) + \delta(s_{i-1}, s_i) + \Phi(s_i, r_i, t_{i-1}) - \Phi(s_{i-1}, r_{i-1}, t_{i-1}) - \rho \cdot \delta(t_{i-1}, r_i)$. It would be sufficient to show that $\Delta_1 \leq 0$ and $\Delta_2 \leq 0$. For the first part, $\Delta_1 = (\rho/2) \cdot \big(\delta(s_i, t_i) - \delta(s_i, t_{i-1}) + \delta(r_i, t_i) - \delta(r_i, t_{i-1})\big) - \rho \cdot \delta(t_{i-1}, t_i) \leq (\rho/2) \cdot \big(\delta(t_{i-1}, t_i) + \delta(t_{i-1}, t_i)\big) - \rho \cdot \delta(t_{i-1}, t_i) = 0$.

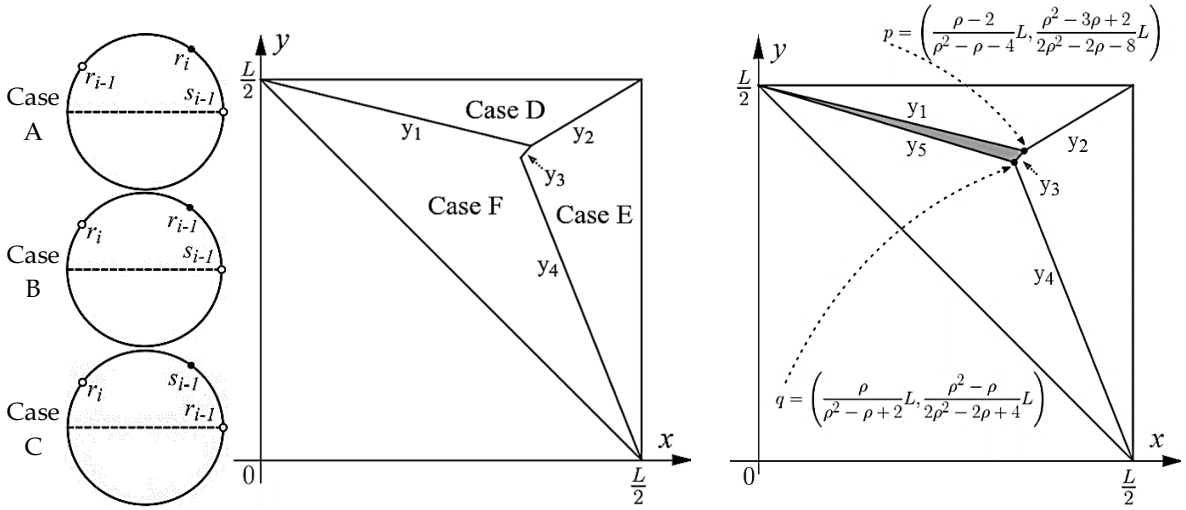


**Figure 1**. Representing the conditions of Cases A-F, as well as the positive region of $\Delta_2$ in Case F (grey region).
$\rho = 1 - \left(9\sqrt[3]{\lambda}^{3/2} + 42\sqrt[3]{\lambda} - 71\right)^{1/2}/6\sqrt[6]{\lambda} + \left(-\sqrt[3]{\lambda} + 48\sqrt[6]{\lambda}/\left(9\sqrt[3]{\lambda}^{3/2} + 42\sqrt[3]{\lambda} - 71\right)^{1/2} + 71/9\sqrt[3]{\lambda} + 28/3\right)^{1/2}/2$ where $\lambda = 2\sqrt{13438}/3 - 1999/27$.

For the second part, since TriAct has three choices of $s_i$ for migration as $r_i$ (in Cases A and E), $r_{i-1}$ (in Cases B and D), and $s_{i-1}$ (in Cases C and F), $\Delta_2$ is bounded from above as $(1-\rho)x + 2y$, $(2-\rho/2)x + (1-\rho/2)y + (\rho/2-1)z$, and $(1-\rho/2)x + (\rho/2)y - (\rho/2)z$, respectively. Therefore, for Cases A, B, and C, we clearly have $\Delta_2 \leq (3-\rho)x < 0$, $\Delta_2 \leq (3-\rho)x < 0$, and $\Delta_2 \leq (1-\rho)x < 0$, in the same order already mentioned. For Cases D, E, and F, we have $z = L - x - y$ and the conditions of these cases are defined using four functions $y_1 = -((\rho/3)/(\rho/2))x + L/2$, $y_2 = (2/\rho)x + ((\rho-2)/\rho) \cdot L/2$, $y_3 = ((\rho-1)/2)\,x$, and $y_4 = (\rho/(\rho-2)) \cdot (L/2 - x)$, which separate the area of a plane with dimensions of $x$ and $y$ (see Figure 1). In Cases D and E, since $y \geq y_1$ it follows that $\Delta_2 \leq (3-\rho)x + (2-\rho)(-((\rho-3)/(\rho-2))x + L/2) + (\rho/2-1)L = 0$, and since $y \leq y_3$ it follows that $\Delta_2 \leq (1-\rho)x + 2(((\rho-1)/2)x) = 0$, respectively. In Case F, for $y_5 = L/2 - x/\rho$, if $y \leq y_5$, then $\Delta_2 \leq \rho y + x - (\rho/2)L \leq 0$. However, if $y > y_5$, i.e., if $x$ and $y$ are in the grey region in Figure 1, then $\Delta_2 > 0$. For this situation, instead of bounding $\Delta_2$, we bound $\Delta_2 + \Delta_2'$, the aggregation of $\Delta_2$ and $\Delta_2' = \delta(s_i, r_{i+1}) + \delta(s_i, s_{i+1}) + \Phi(s_{i+1}, r_{i+1}, t_i) - \Phi(s_i, r_i, t_i) - \rho \cdot \delta(t_i, r_{i+1})$, which is defined as the value of $\Delta_2$ for the next request $r_{i+1}$. We note that $s_i = s_{i-1}$ and we show that $\Delta_2 + \Delta_2' \leq 0$ for all six cases of $r_{i+1}$, as follows.

For $r_{i+1}$, in Cases A, B, and C, it follows that $\Delta_2' \leq (3-\rho)\delta(s_i, r_i) = (3-\rho)\delta(s_{i-1}, r_i) = (3-\rho)y$. Therefore, $\Delta_2 + \Delta_2' \leq -((\rho^2 - 3\rho - 1)/(\rho - 2))\,L/2$ which is negative since $\rho > (3+\sqrt{13})/2$. In Case D, it follows that $\Delta_2' \leq -(\rho-2)\delta(s_i, r_{i+1}) - (\rho-3)y + (\rho/2-1)L$ and $\delta(s_i, r_{i+1}) \geq (2/\rho)y + ((\rho-2)/\rho)\,L/2$, hence $\Delta_2 + \Delta_2' \leq ((\rho+4)/\rho)y + x - ((\rho^2 - 2\rho + 4)/\rho)\,L/2$. The value of $((\rho+4)/\rho)y + x$ is maximized at $p$, at which $y_1$ and $y_3$ intersect. In Case E, we have $\Delta_2' \leq 2\delta(s_i, r_{i+1}) - (\rho-1)$ and $\delta(s_i, r_{i+1}) \leq (2/\rho)y + ((\rho-2)/\rho)\,L/2$. Therefore, we also have $\Delta_2 + \Delta_2' \leq ((\rho+4)/\rho)y + x - ((\rho^2 - 2\rho + 4)/\rho)\,L/2$. In Case F, it follows $\Delta_2' \leq \rho\delta(s_i, r_{i+1}) + y - (\rho/2)L$. For $x$ and $y$ in the grey region in the figure, it follows that $y \geq ((\rho^2 - \rho)/(\rho^2 - \rho + 2))\,L/2$, which is the $y$-coordinate of $q$ at which $y_5$ and $y_3$ intersect. This implies that $\delta(s_i, r_i)$ is larger than the $x$-coordinate

$(\ (\rho-2)/(\rho^2-\rho-4))L$ of $p$. Thus, $\delta(s_i, r_{i+1}) \leq (\rho/(\rho-2))(L/2-y)$. Therefore, $\Delta_2+\Delta_2' \leq -((\rho+2)/(\rho-2))y+x-(\rho(\rho-4)/(\rho-2))\,L/2$. The value of $-((\rho+2)/(\rho-2))y+x$ is maximized at $q$, at which $y_5$ and $y_3$ intersect. Therefore, $\Delta_2' \leq -(\rho(\rho-3)/(\rho-2))\,L/2 < 0$. Now, the competitiveness of $\rho$ is clarified (detailed proof is preprinted [31]).

To show $\rho$ is indeed tight, the behavior of an adaptive offline adversary is found as follows. For a sufficiently large $n$, a request sequence $\sigma = r_1, \dots, r_{4n}$, and four request nodes $s_0$, $a$, $b$, and $c$ on a ring network such that neither $\pi(s_0, a)$ nor $\pi(b, c)$ has an edge of $\pi(s_0, b)$. The adversary sets $\delta(s_0, b) = ((\rho^2-3\rho+2)/(2\rho^2-2\rho-8))L$ and $\delta(s_0, a) = \delta(b, c) = ((\rho-2)/(\rho^2-\rho-4))L$. The requests $r_{4k+1}$, $r_{4k+2}$, $r_{4k+3}$, and $r_{4k+4}$ are at $a$, $b$, $c$, and $s_0$, respectively, where $0 \leq k < n$. Moreover, the algorithm of adversary $X$ migrates the server from $s$ to $a$ before any request occurs, with the cost of $\delta(s, a)$. The request $r_{4k+1}$ is served with no cost and the server does not migrate from $a$. The request $r_{4k+2}$ is served with the cost of $\delta(a, b)$ and the server migrates from $a$ to $c$ with a cost of $\delta(a, c)$. The request $r_{4k+3}$ is served with no cost and the server does not migrate from $c$. The request $r_{4k+4}$ is served with the cost of $\delta(c, s)$ and the server migrates from $c$ to $a$ with a cost of $\delta(c, a)$. Therefore, $cost_{TriAct}(s, \sigma) = n\big(2\delta(s, a) + 4\delta(s, b)\big)$ and $cost_X(s, \sigma) = \delta(s, a) + n\big(2\delta(s, a)\big)$. Thus, we have $cost_{TriAct}(s, \sigma)/cost_{ALG}(s, \sigma) = \rho$, while $n$ approaches infinity.

Concerning future works, we think similar technique can improve our solution on restricted rings such as 4-node and 5-node networks and we conjecture the optimality of proposed algorithm for general rings. One might design an algorithm for $D \geq 2$ with a competitiveness of $c < 4$.

## 6. Telecommunicational Servers

The traditional problems of managing data arise for optimizing the *load* of network, in which the cost is a *summation* of service and movement upon each request. We rather propose to optimize the *time* which leads considering a *maximization* of service and movement. This assumption is due to the free interconnections in the network when there is no overlap, as well as assuming enough capacity in modern networking like the telecommunications in the case of overlapping of service and movement. Therefore, we propose to establish definition of *telecommunicational servers problem*, with the objective of $\sum_{i=1}^{m} \max\{r_i(s_{i-1}), \gamma(s_{i-1}, s_i)\}$, and similarly, *telecommunicational server problem*, with the objective of $\sum_{i=1}^{m} \max\{\delta(s_{i-1}, r_i), D \cdot \delta(s_{i-1}, s_i)\}$ for the distance metric $\delta$ in the network with a single page size $D$, respectively as time-efficient variants of data management and page migration problems. We have considered to apply work functions towards designing deterministic algorithms to solve these problems. For this purpose, we first need to define the problems in recursive form, then to use dynamic programming to find work functions of the problems by terminating at all possible nodes, and follow the rest of art, mentioned in Section 2 (Efficient Design).

## 7. Conclusion

We demonstrate the designative tactics and the analytical frameworks which have been founded to study online computation. Popular server problems of networks are surveyed, involving their general algorithmic ideas and qualitive states of the art. The concentration of illustration falls towards the uniform page migration problem, in which the problem is to locate the server to reduce the overall load of network. A couple of deterministic algorithms are efficiently designed and theoretically analysed, separately for the pair of Euclidean space and ring networks, for this online problem. Furthermore, we propose time-efficient definitions of server problems in modern networks, and then decide on and arrange in advance, to study it later by grasping the powerful tool of work functions. Since some suggestions indicate online algorithms as a framework of interactive computation, we propose to generalize the frameworks of online computation to cover problems beyond online algorithms. Additionally, since there is no clear classification for the hardness of online algorithms, we propose the study of reconfiguration graphs to attain a kind of sense on this matter, since the framework of online problems is a subject of reconfiguration.

## References

[1] A. Borodin and R. El-Yaniv, *Online Computation and Competitive Analysis*, vol. 2, no. 1471–4914 LA–eng PT–Journal

Article PT–Review PT–Review, Tutorial. cambridge university press, 1998.
[2] A. V Aho and J. E. Hopcroft, *The design and analysis of computer algorithms*. Pearson Education India, 1974.
[3] D. D. Sleator and R. E. Tarjan, "Amortized efficiency of list update and paging rules," *Commun. ACM*, vol. 28, no. 2, pp. 202–208, Feb. 1985.
[4] E. Torng, "A unified analysis of paging and caching," *Algorithmica*, vol. 20, no. 2, pp. 175–200, 1998.
[5] A. Fiat, R. M. Karp, M. Luby, L. A. McGeoch, D. D. Sleator, and N. E. Young, "Competitive paging algorithms," *J. Algorithms*, vol. 12, no. 4, pp. 685–699, 1991.
[6] A. Borodin, S. Irani, P. Raghavan, and B. Schieber, "Competitive paging with locality of reference," *J. Comput. Syst. Sci.*, vol. 50, no. 2, pp. 244–258, 1995.
[7] M. Manasse, L. McGeoch, and D. Sleator, "Competitive algorithms for on-line problems," in *Proceedings of the twentieth annual ACM symposium on Theory of computing*, 1988, pp. 322–333.
[8] S. Albers, "Online Algorithms," in *Interactive Computation*, Springer Berlin Heidelberg, 2006, pp. 143–164.
[9] E. Koutsoupias and C. H. Papadimitriou, "On the k-server conjecture," *J. ACM*, vol. 42, no. 5, pp. 971–983, 1995.
[10] Y. Bartal, B. Bollobás, and M. Mendel, "A Ramsey-type theorem for metric spaces and its applications for metrical task systems and related problems," in *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, 2001, pp. 396–405.
[11] Y. Bartal, A. Blum, C. Burch, and A. Tomkins, "A polylog (n)-competitive algorithm for metrical task systems," in *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, 1997, pp. 711–719.
[12] Y. Bartal, A. Fiat, and Y. Rabani, "Competitive algorithms for distributed data management," *J. Comput. Syst. Sci.*, pp. 341–358, 1995.
[13] B. Awerbuch, Y. Bartal, and A. Fiat, "Competitive distributed file allocation," *Inf. Comput.*, vol. 185, no. 1, pp. 1–40, 2003.
[14] M. Imase and B. M. Waxman, "Dynamic Steiner tree problem," *SIAM J. Discret. Math.*, vol. 4, no. 3, pp. 369–384, 1991.
[15] D. L. Black and D. D. Sleator, "Competitive Algorithms for Replication and Migration Problems," Pittsburgh, PA, USA, 1989.
[16] C. Lund, N. Reingold, J. Westbrook, and D. Yan, "Competitive On-Line Algorithms for Distributed Data Management," *SIAM J. Comput.*, vol. 28, no. 3, pp. 1086–1111, 1999.
[17] R. M. Karp, "A 2k-competitive algorithm for the circle," *Manuscript, August*, vol. 5, 1989.
[18] A. Gupta, I. Newman, Y. Rabinovich, and A. Sinclair, "Cuts, trees and ℓ1-embeddings of graphs," *Combinatorica*, vol. 24, no. 2, pp. 233–269, 2004.
[19] A. Matsubayashi, "Non-greedy Online Steiner Trees on Outerplanar Graphs," Springer, Cham, 2017, pp. 129–141.
[20] Y. Kawamura and A. Matsubayashi, "Randomized Online File Allocation on Uniform Cactus Graphs," *IEICE Trans. Inf. Syst.*, vol. 92, no. 12, pp. 2416–2421, 2009.
[21] W. Glazek, "Lower and Upper Bounds for the Problem of Page Replication in Ring Networks," in *Mathematical Foundations of Computer Science*, 1999, pp. 273–283.
[22] W. Glazek, "Online algorithms for page replication in rings," *Theor. Comput. Sci.*, vol. 268, no. 1, pp. 107–117, 2001.
[23] Y. Bartal, M. Charikar, and P. Indyk, "On page migration and other relaxed task systems," *Theor. Comput. Sci.*, vol. 268, no. 1, pp. 43–66, 2001.
[24] M. Bienkowski, J. Byrka, and M. Mucha, "Dynamic beats fixed: On phase-based algorithms for file migration," in *The 44th International Colloquium on Automata, Languages, and Programming*, 2017, no. 13, pp. 1–13.
[25] J. Westbrook, "Randomized algorithms for multiprocessor page migration," *On-line Algorithms*, vol. 7, no. 5, pp. 135–150, 1992.
[26] M. Chrobak, L. L. Larmore, N. Reingold, and J. Westbrook, "Page migration algorithms using work functions," *Proc. 4thInternational Symp. Algorithms Comput.*, vol. 762, pp. 406–415, 1993.
[27] A. Matsubayashi, "Asymptotically Optimal Online Page Migration on Three Points," *Algorithmica*, vol. 71, no. 4, pp. 1035–1064, 2015.
[28] A. Matsubayashi, "UNIFORM PAGE MIGRATION ON GENERAL NETWORKS," *Int. J. Pure Appl. Math.*, vol. 42, no. 2, pp. 161–168, 2008.
[29] B. Feldkord and F. Meyer auf der Heide, "The Mobile Server Problem," in *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures - SPAA '17*, 2017, pp. 313–319.
[30] A. Khorramian and A. Matsubayashi, "Uniform Page Migration Problem in Euclidean Space," *Algorithms*, vol. 9, no. 3, p. 57, 2016.
[31] A. Khorramian and A. Matsubayashi, "Online Page Migration on Ring Networks in Uniform Model," Dec. 2016.

平成３０年　１月３０日

# 学位論文審査報告書（甲）

１．学位論文題目（外国語の場合は和訳を付けること。）

Design of Efficient Online Algorithms for Server Problems on Networks

（ネットワーク上のサーバ問題のための効率的なオンラインアルゴリズムの設計）

２．論文提出者　(1) 所　　属　電子情報科学　　専攻

(2) 氏（ふり）　名（がな）　Amanj Khorramian（あまんじ　ほっらみやん）

３．審査結果の要旨（600～650 字）

平成 30 年 1 月 30 日に第 1 回学位論文審査委員会を開催，同日に口頭発表，その後に第 2 回審査委員会を開催し，慎重審議の結果，以下の通り判定した．なお，口頭発表における質疑を最終試験に代えるものとした．

ネットワーク上で通信負荷を抑制するサーバ配置を求める問題やマルチプロセッサシステムにおいて効率的なキャッシュ配置を求める問題は，サーバ問題と総称されるいくつかのオンライン問題に定式化できることが知られ，盛んに研究されている．本論文ではそうした問題の一つであるページ移動問題に対して効率的なオンラインアルゴリズムを提案している．具体的には，任意の次元のユークリッド空間上で実行できる極めて単純なアルゴリズムを設計し，このアルゴリズムがある条件下で既存のアルゴリズムを上回る性能を持つことを証明している．また，リングネットワーク上で実行できる別のアルゴリズムを設計し，このアルゴリズムがある条件下で既存のアルゴリズムを上回る性能を持つことを証明している．

以上の研究成果は，長年未解決であった問題を部分的に解決しており，オンラインアルゴリズムの研究分野において重要な知見を与えている．従って，本論文は博士(学術)に値すると判定する．

４．審査結果　(1) 判　　定（いずれかに○印）○合　格　・　不合格

(2) 授与学位　　博　士（　学術　）