Dissertation

DESIGN OF EFFICIENT ONLINE ALGORITHMS FOR SERVER PROBLEMS ON NETWORKS

Graduate School of
Natural Science and Technology
Kanazawa University

Division of Electrical Engineering and Computer Science

1524042006

Name: Amanj Khorramian

Chief advisor: Akira Matsubayashi

January 5, 2018

DESIGN OF EFFICIENT ONLINE ALGORITHMS FOR SERVER PROBLEMS ON NETWORKS

A dissertation submitted

to Kanazawa Univ. in partial

fulfillment of the requirements for the

degree of Doctor of Philosophy

by

Amanj Khorramian

2018, March

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.

TABLE OF CONTENTS

DEDICATION

To my dear parents ...

ACKNOWLEDGEMENTS

In the name of God, the Most Gracious, the Most Merciful, who may have had mercy upon me to find ideas in mind for the accomplishment of my doctoral research projects soon.

First of all, I must thank my dear teachers of elementary school, middle school, and high school in the lovely city of Saghez, as well as my dear professors of BSc and MSc programs in Iran. They generously taught me much lessons of life and science through valuable lectures and useful classes. It was not possible to obtain necessary skills for conducting my research without such serious and fundamental learnings. I would like to thank also my former colleagues at the department of Computer Engineering and Information Technology of University of Kurdistan. They kindly agreed with my proposal to study Online Algorithms. I am thankful also to the other colleagues as well as the decision makers of the University of Kurdistan for accepting my application to study abroad and helping me to receive a financial support from my home country.

I especially thank the assigned supervisor for my doctoral research, Prof. Akira Matsubayashi, for his guidance during my PhD projects. I thank him, also because after my achievements in two intensive research projects on Online Algorithms in the first half of the 3-year doctoral program, he agreed that I visit Japan Advanced Institute of Science and Technology (JAIST) for the second half. I would like to thank also the anonymous decision makers who approved that I visit JAIST, which is a well-equipped, nonracist-organized, scientifically-ethical, and highly-qualified university to study and

research in Informatics, through real lectures, fair rules, stable regulations, honest admission policy, and carefully-affiliated full-professors.

I am thankful to Prof. Ryuhei Uehara who accepted that I visit his informative lab at JAIST. I discovered various topics of Information Science from his laboratory. During my visit, I was happy to finish some research projects and perform novel experiments in different themes.

Amanj Khorramian

Jan 5, 2018

CHAPTER 1

Introduction

Daily-life is greatly influenced by electronic devices (e.g. smartphones, laptops, personal computers, workstations, supercomputers, etc.). The desired functionality of these appliances depends on their processing units. By developing the applications, the needs for faster processors are increased. In recent decade, there is no big improvement to the clock speed of processors, due to the limitations of electrons. One resolution is to interconnect multiple processors in a network which shares resources for gaining more performance. The running of such and other kinds of networks requires to deal with several challenging problems.

In informatics, the study of networks has been drawing more attention, continually. The structure of connected components appears in various areas and different platforms. For example, social networks, biological networks, computer networks, telecommunication networks, semantic networks, distributed systems, and graph theory, all are about investigating the connection or interaction among separate elements. The power of computation and computing resources is essential for coping with so many challenges which are arising to obtain solutions in the investigation.

Depending on the properties and models of the network, some problems are taken into consideration and need to be resolved, towards utilizing the network in a desirable way. “How to find a set of specific elements in the network?”, “What is the fastest way to reach a node?”, “Which representation is the best for storing the network, altogether or partially?”, “Are there any parts of network subject to reconfiguration?”, “Which elements are the optimum choice to carry a set of resources shared in the

network?", "How to serve the requests emerging from the nodes?", "We need to know about some unknown properties of network; then, what strategy is proper enough for processing the entire network?", and many other questions are real problems to study the networks in the contemporary world. These problems need computation for attaining resolution. Efficient solutions for big networks are demanding since they are coming into view of human life.

In practical point of view, agile electrons are helpful and widely available via processing units to accelerate the computation. Personal computers have great ability, and can be equipped well by programming tools and learning approaches, in a reasonable cost for tackling the problems with a scale which can fit the resources. It is obvious that PCs are restricted to handle the problems for a size that is small enough, even with a parallel processing attitude. More power of computing for larger problems can be found through suitable but hardly-available approaches like high-performance computing, cloud computing, grid computing, distributed processing, etc. To design a method for the problem, it is needed to take both the limitations of platform and the model of network into account. Usually, the art of planning a method to fully utilize the resources need specific expertise and deep thinking. Recent efforts on quantum and biological computation seem likely to reach a stage that could provide even more power for a significant amount of larger problems. A broad range of network problems has been studied practically.

In theoretical point of view, the network problem is formally defined, first of all. After that, depending on the attributes of problem, an efficient technique is employed to design an algorithm which guarantees a level of optimality through a comprehensive analysis. There exist some basic techniques (e.g. randomization,

dynamic programming, backtracking, approximation, divide and conquer, etc.[1]) which have been widely applied to the design of algorithms in real problems, including network problems. It is quite usual to represent the network as a graph, in which there is a bijection between the vertices of graph and the separate components of network, such that each connection between two components is represented as an edge between corresponding vertices of the graph. In this way, the problem in the network can be regarded as a graph problem. Graph problems, in various aspects, are dramatically studied under the area of graph theory[2], so that the achieved methods might be considered as the principal or subsidiary solution for designing of algorithms in real network problems.

Theoretically, depending on the aspect of view, 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, suppose considering the availability of input data to the network for making decision about the design. 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*. As another example, an algorithm is *deterministic* or *randomized,* depending on the behavior of algorithm to be either functional or based on random distribution, respectively. One more example; if the amount of computation to solve a problem increases linearly regarding the size of network, then the algorithm is categorized as linear. Other categories like constant-time,

---

[1] To study more about these techniques, there are textbooks like [8] and [49].

[2] Some fundamental algorithms in graph theory are discussed in [50] and [51].

polynomial-time, exponential-time, etc., exist to show the complexity of algorithm. Since the exponential-time algorithms are less practical by the enlargement of network (e.g. take tens of years to obtain optimal solution), there are helpful approaches like approximation algorithms, which give solution with a guarantee of maximum gap from the optimal solution, and applicable in a reasonable time. The similar thing 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.

1.1 Structure

This thesis is a collection of published and unpublished results and articles during a doctoral program of philosophy, and is organized as follows:

CHAPTER 1 firstly introduces the foundation and motivation of the program from two viewpoints, surrounded by some information and prospection about related topics, narrowed into a more specific target of subject, at which the main study has been accomplished. After this structure, there is a philosophical discussion about the quiddity of this study, followed by specialized sections. Section 1.3 specifies the area of problems in which our problem can be defined. The subsections discuss about the theoretical framework and techniques available to cope with the problems, as well as analytical methods and tools to measure proposed computations. Section 1.4 has further explanation about networks and section 1.5 narrows the range of problems to a more specific case of this study. Section 0 precisely defines some concentrated problem through proposing mathematical notations, and reviews the relevant study of problem in literature, followed by a sketch of our novel contributions in this study.

CHAPTER 2 focuses on a specific research work about a special kind of network in this topic. After a preamble, an unprecedented online algorithm with its detailed analysis and conclusion is demonstrated. This is a successful achievement of our work for the problem, and has been already published. [1]

CHAPTER 3 investigates another particular network as a separate project, and suggests a new online algorithm with a tight analysis for the network. The algorithm and analysis are illustrated and followed by remarks. In the proposed analysis, there is a slight extension to a standard analytical method. Although this work has been submitted for publication from long time ago, there is still no response from decision makers despite several contacts with editorial office. An earlier version of work is preprinted for reference. [2]

CHAPTER 4 is proposing an initial idea to develop some network problems with the goal of efficiently reduce the time (rather than the load), inspired by the original cases of study, which seem essential and interesting for modern applications. Finally, CHAPTER 5 concludes the dissertation by an essay on its main themes with some remarks and summaries.

1.2 Philosophy

In theoretical aspect of problem solving, online algorithms are considered as a framework for *interactive computation* that is a paradigm with modeling interactive agents, which can do computation concurrently [3]. Interaction is suggested as an extended model of Turing machines for understanding the principles of computing. This shift in the modes of thinking accords with some bases, including the assumption that interaction provides an empirical foundation for computer problem solving [4], as well as a theory saying that computing is the origin of informatics [5]. A formal

definition of interactive computational problems can be found in [6]. Anyway, after accepting this paradigm as a generalized computing approach towards finding mathematical truth, this potential question comes into mind whether the current framework for studying online problems (see section 1.3.2) also gets a capacity for a kind of generalization?

1.3 Online Problems

The 'main problem' of *online problems* is that there is 'partial knowledge' about the whole data. Finding a solution depends on processing the input data of problem. The problem is that the entire data are not available at once, but the input data are coming one by one. The computation must provide a solution upon each input, and the solution is subject to get change by the next input arrival. In this way, each input can be interpreted as a request of task that may change the state of solution. The final goal is to reduce the total cost[3] needed to process the inputs and change the states, as far as possible. As mentioned at the preface of CHAPTER 1, each problem needs to be formally defined. In the case of online problems, *metrical task systems* [7] have been widely utilized as a formal framework for defining online problems. This model is established based on *metric space* as discussed in the following section. Note that each section may borrow the notations of its preceding sections. For the notations which are not mentioned, the readers are referred to [8].

[3] This cost depends on the definition of the problem and its specific platform.

1.3.1 Metric Space

For a set S and a function $\gamma: S \times S \to \mathbb{R}$, the pair $M = (S, \gamma)$ is a metric space if γ satisfies the conditions of identity of indiscernibles (i.e. $\gamma(s_i, s_j) = 0 \leftrightarrow s_i = s_j$), symmetry (i.e. $\gamma(s_i, s_j) = \gamma(s_j, s_i)$), and triangle inequality (i.e. $\gamma(s_i, s_k) \leq \gamma(s_i, s_j) + \gamma(s_j, s_k)$), which imply non-negativity (i.e. $\gamma(s_i, s_j) \geq 0$), where $s_i, s_j, s_k \in S$. [9]

In online problems, we regard S as the set of all computable states of solution, and γ as the cost of computation to change from a state to another state. In the literature, γ is intuitively denoted as a function of *cost* or *distance* between two states [7], [10]. Besides, $|S|$ denotes the finite number of states.

1.3.2 Metrical Task System

Task is an abstraction of the request, and a pair $\mathcal{S} = (M, T)$ is a metrical task system (MTS) if $M = (S, \gamma)$ is a metric space (see section 1.3.1), and T is the set of tasks. 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. Based on this system, a huge number of online problems are defined and studied well. 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* [7] if $\gamma(s_i, s_j)$ is identical for every $s_i, s_j \in S$ where $s_i \neq s_j$.

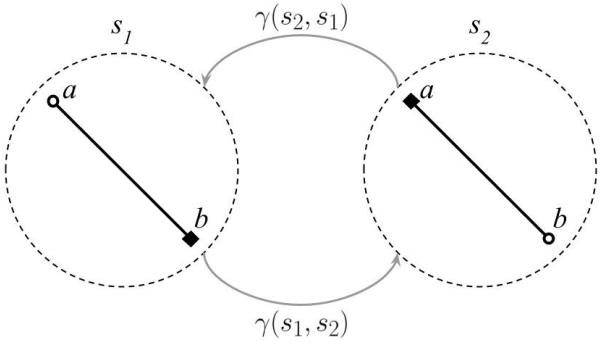


Figure 1. Both states s_1 and s_2 of an MTS are shown as a simple network with two nodes and one resource

The network branch of online problems is often suitable to be represented using graphs. For example, suppose a network with only two nodes and one shared resource (showed as filled square in Figure 1). Figure 1 represents the set of all possible states (i.e. s_1 and s_2) of the corresponding metrical task system for this network. This example is very simple, because its network has only two configurations which can be reconfigured to each other.

We think that, for studying the details of more complicated examples, MTS can be adopted as a practical application under a recently proposed, and actively studied framework called *reconfiguration graphs* [11]–[13], where each node of a corresponding graph maps to one state of the MTS. A question here arises, whether or not, some way exits to classify online problems from a hardness point of view, since reconfiguration directly works on the framework of online problem definition.

Machine learning appears everywhere and so are its utilizations. As for online problems, there exist research efforts recently proposed, to adopt some learning methods for coping with similar problems to MTS using randomized approaches [14], [15]. But it is worth paying attention that such works lose the important property of MTS that considers changing the states.

1.3.3 Online Algorithms

Having the initial state of solution at $s_0 \in S$, 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. In other words, according to the framework [7], upon each request r_i, the algorithm first changes the state of solution from A_{i-1} to A_i, and after that, processes request r_i. Thus

and so, A_i may be interpreted as the response of A to 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 $\gamma(A_{i-1}, A_i) = 0$ and no change occurs to the state upon request r_i.

In our example of Figure 1, the resource at state s_1 is on node b and far from node a, therefore it makes sense if we let $r_i = (5,1)$ for every request r_i on node a to access the resource, and similarly for $r_i = (1,5)$ on node b. Additionally, let the cost of changing the state $\gamma(s_1, s_2) = \gamma(s_2, s_1) = 10$. Let $s_0 = s_1$ for any algorithm and the sequence σ of requests sequentially arrive on $a, b, a, a, a,$ and a. If we assume an online algorithm U keeps the initial state unchanged (i.e. the resource is always maintained on b). Then, the total cost equals to $U(\sigma) = 26$, incurred only by request processing. Let another example V change the state once, e.g. upon the third request, then $V(\sigma) = 20$ caused by the costs of request processing and state changing. Although V took the heavy cost of changing a state $\gamma(s_1, s_2)$, it finally yielded a better gain than U for the example sequence $\sigma = a, b, a, a, a, a$.

"Does V work better than U?

For this sequence of example, yes!

"How about other sequences?

This is the actual question that challenges the design of online algorithms, because in online problems, the algorithm does not possess the whole sequence of requests in advance. The requests come one by one. For instance, if the sequence was $\sigma = b, b, b, b, b, b$, then U worked much better than V of course.

"Which one wins in this competition; U or V?

The quality of an online algorithm is measured relative to the other algorithms by taking all possibilities into account. 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.

In this fashion, a framework is invented [16] to analyse the efficiency of an online algorithm in a breathtaking comparison with an optimal algorithm which already possesses the request sequence. Next section describes the goal of this invention that is broadly used as a method of investigating online algorithms and quantifying their qualities.

1.3.4 Competitiveness

Competitive analysis provides a method to measure the quality of online algorithms. This measurement yields a value c for an online algorithm A that gives solution for a problem P. In a nutshell, this value shows a maximum ratio, calculated by dividing the cost incurred by A over the cost of any other offline algorithm, for any sequence of requests $\sigma = r_1, r_2, \dots, r_m$. The maximum ratio can be obtained using a comparison between the online algorithm A and an optimal algorithm X.

For example, suppose the set of all possible request sequences $\{\sigma 1, \dots, \sigma 10\}$ are determined by an algorithm A' and evaluated as shown in Figure 2 beside the squares above each σi. The costs for an optimal algorithm X' are shown near circles. $\sigma 4$ obtains the maximum ratio of $16/6$. This example is proposed to provide a kind of intuition, though it loses some minor condition of actual definition that is discussed as follows.

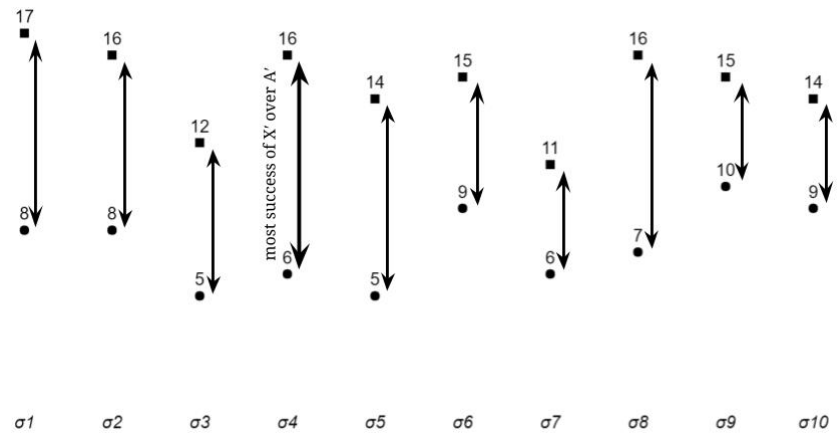


Figure 2. Representing a competition between an online algorithm A' and an optimal algorithm X'. The total cost of responding and processing each request sequence σi is shown above it, near a square for A' and near a circle for X'.

Formally [7], an online algorithm A is c-competitive, if there exists a constant α, such that $A(\sigma) \leq cX(\sigma) + \alpha$, for any finite sequence σ. If $\alpha = 0$ then A is *strictly* c-competitive. If A is a randomized algorithm, then $A(\sigma)$ denotes the expected value of cost, rather than denoting the determinable cost in the case of deterministic algorithm. Against a deterministic algorithm A, the optimal algorithm X cannot be online because X must know the entire sequence σ of requests in advance. A is *competitive* if it hits a constant *competitive ratio* c.

In this area, the worst-case of efficiency is being measured in a comparison, and it is noteworthy that the computational complexity of algorithms (i.e. the amount of computation by algorithm) is not the case of studying online problems.

1.3.5 Adversary

Adversary models allow to analyse online algorithms in more detail. The analysis can be viewed as an unfair game, between the designed online algorithm and a malicious adversary against it. The adversary knows the algorithm well and produces a sequence of requests as difficult as possible, towards maximizing its competitive ratio.

Depending on the nature of an online algorithm A, three different models of adversary exist as follows:

Adaptive offline adversary knows everything about A, and is commonly utilized when A is deterministic. This adversary generates a sequence of requests σ, 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 for the produced request sequence σ and pays the optimal cost $X(\sigma)$. The other two models of adversary are defined for the randomized case of online algorithms.

Adaptive online adversary cannot determine the behavior of A, since A is a randomized algorithm. This adversary produces each request according to A's definition and against its random (not deterministic) responses so far. Hence, the behavior of adversary is non-deterministic and acts in an online fashion. This manner incurs an expected cost $X(\sigma)$ for its generated sequence of requests σ.

Oblivious adversary does not know anything about the random behavior of A and its responses, but still knows well the definition of A. In this model, the adversary produces the entire sequence of requests σ in advance, only based on the definition of algorithm. This adversary shares a common behavior with the adaptive offline adversary that pays optimal cost $X(\sigma)$ too. Oblivious adversary is even weaker than the adaptive online adversary, because it does not take the behavior of A into account.

Regarding the adversary models, an online algorithm A is c-competitive, if for an additive constant α we have $A(\sigma) \leq cX(\sigma) + \alpha$. At this point, we mention some useful relations among these models.

Theorem 1. [17]: *If A is a c-competitive randomized algorithm against any adaptive offline adversary, then there exists also a c-competitive deterministic algorithm.*

Theorem 2. [17]: *If A is a c-competitive randomized algorithm against any adaptive online adversary, and A' is a c'-competitive randomized algorithm against any oblivious adversary, then A is a (cc')-competitive randomized algorithm against any adaptive offline adversary.*

1.3.6 Amortized Analysis

Initially, a kind of amortization used to bound the computational complexity by aggregating a sequence of operations [18]. Later, this method was generalized as an approach to analyse algorithms [19] by amortizing the efficiency over a part of computation, in the form of a constant or function. The functional idea has been adapted as an analytical method of online computation [7].

In the case of online algorithms, the goal is 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_ζ be the set of all computable configurations of an algorithm ζ.

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 ζ to perform E_j, and $\Phi_j \geq \beta$ denotes the value of Φ right after finishing the accomplishment of E_j by both algorithms, for a constant β. Set Φ_0 as the value of Φ 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 ζ by two consequent partitions of actions. Each partition is called an event.

To analyse the algorithm of our second project in CHAPTER 3, we utilize this method, and also unearth the initial technique of aggregation [18], which leads to combine two consecutive subsets for following the analysis by somehow disturbing the common approach that is discussed above.

1.3.7 Work Functions

Work function algorithms provide a natural procedure for confronting adversaries. An adversary X knows the online algorithm and acts against it (see section 1.3.5), 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 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. In the

underlying abstraction, for any deterministic online problem, the work functions always provide an online algorithm which shows the power of this technique.

Theorem 3. [20]: *The work function algorithm is* $(2|S|-1)$*-competitive for any metrical task system* $((S,\gamma),T)$.

Later in CHAPTER 4, we will propose new online problems of telecommunicational servers and telecommunicational server, as time-efficient variants of some traditional problems in modern networking. For these problems, it would be very reasonable to think of designing online algorithms based on work functions.

1.4 Networks

The scope of networks has been broadened to study various topics. For example, a computer network allows its nodes to share resources (e.g. printers). Telecommunication networks and multiprocessor systems also share resources (e.g. packets, and data, respectively). The resources are located on the nodes of a network which are connected to other nodes for communicating and sharing. *Server* is a node of the network that is holding a resource. For example, in computer networks, the data are stored on powerful computers called servers [21]. Depending on the restrictions and construction of infrastructure, as well as the target and the scale of usage, each network is established with a specific topology. Some notable and basic topologies include complete graphs, paths, stars, rings, trees, and meshes [22]. Some examples of small networks with basic topologies are shown in Figure 4.

The concentration of our study considers the topology of networks in the metric space (see section 1.3.1). This space covers the basic topologies such as general and

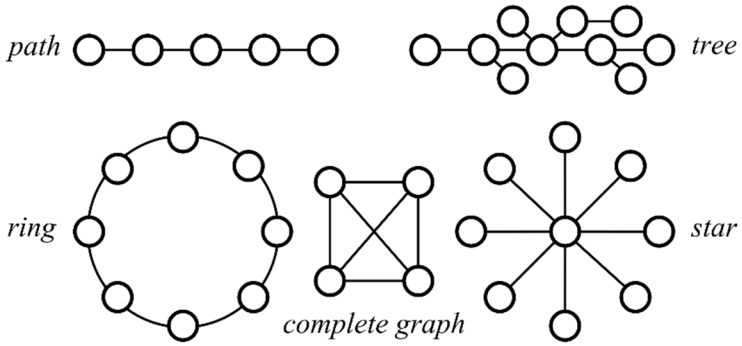


Figure 4. Examples of some basic network topologies

restricted graphs (Figure 4), and the inherent continuous systems such as Euclidean and taxicab geometries (Figure 3).

1.5 Server Problems

Perhaps, *list accessing problem* is one of the most extensively studied online problems. Minimizing the total cost for accessing a requested item in an ordered list and self-reordering of the list is the objective of this problem. Reordering strategies basically include moving the accessed item to the front of the list, transposing the item with its preceding one, and sorting according to the frequency of accesses.[4] 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

[4] 2-competitiveness of first strategy is known [16], and is privately noted to be the best possible competitiveness for deterministic algorithms [52].

requested page, then it achieves k-competitiveness, and no other deterministic online algorithm can achieve a better ratio [16]. The algorithm of evicting the least recently used page is recognized as a form of *marking*, a general strategy that works in phases[5] [23]–[25].

The study of server problems is roughly about managing resources in networks. *k-server problem* and *data management problem* are two of extensively studied online problems that come out in the management of data in networks.

1.5.1 k-Server Problem

k-server problem [26] is a general form of page replacing problem, in which the cost of fetching different pages may differ. More clearly, in a metric space (S, δ), 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. Figure 5 displays an example of k-server problem. Note that we would have $s_3 = s_2$ in this example.

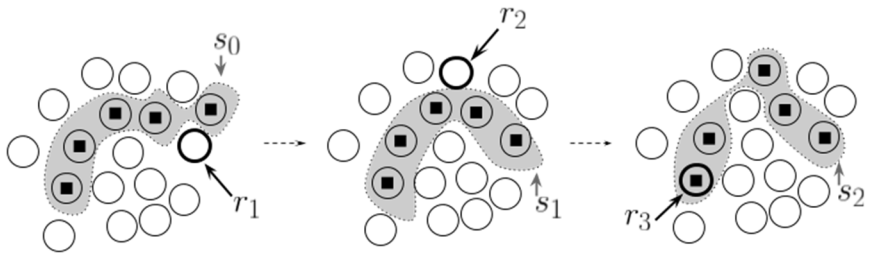


Figure 5. Example of an online algorithm to demonstrate k-server problem where $k = 5$, for the first triple of requests r_1, r_2, and r_3. Each circle represents a node in the network, and the copies of data on nodes are denoted by filled squares. Each grey area covers a set s_i of servers.

[5] It is interestingly worth mentioning there are 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).

There is no c-competitive deterministic [26], or randomized algorithm against adaptive online adversaries [10], 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 [27]. The lower bound of problem is shown [28] to be $\Omega(\log(k)/\log^2(\log(k))$ against oblivious adversaries, and is conjectured to be $\Theta(\log(k))$ [10]. If $|S| = k + c$, then there is a $O(c^6\log^6(k))$-competitive algorithm against oblivious adversary [29].

1.5.2 Data Management Problem

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 $r_1, r_2, \ldots, r_n$ in an online manner. The corresponding metric space of a task system (S, γ) 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 $w \in$

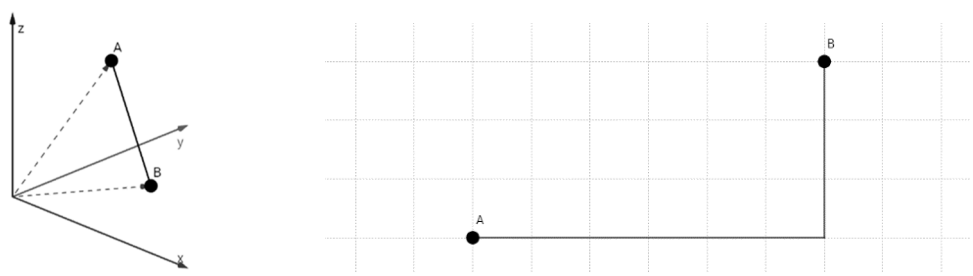


Figure 3. Examples of Euclidean distance in 3D space (left) and Manhattan distance in 2D space (right), between two nodes *A* and *B*

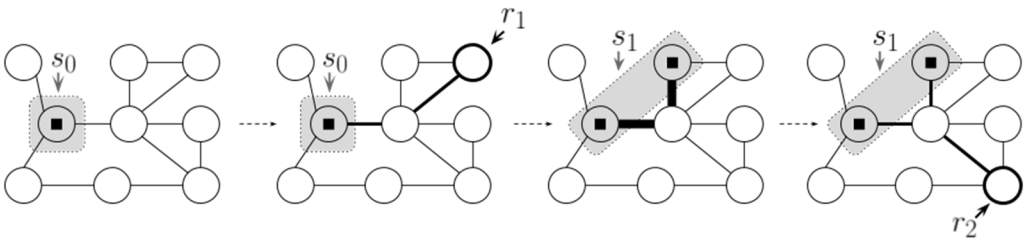


Figure 6. Representation of a sample algorithm for data management problem in a graph. The 'read' request on r_1 is served through a minimal path (slightly shaded) from s_0. After that, the algorithm moves a copy of data through a minimal path (strongly shaded) to another node. A 'write' request on r_2 then sets both copies of data through a minimum Steiner tree (slightly shaded).

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 $s_{i-1} \cup \{r_i\}$. Here we open an interesting question: "how to define $r_i(s_{i-1})$ for the metric spaces other than graphs"? One answer might be "the maximum distance from s_{i-1} to r_i" for the case of Euclidean space. Anyway, 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 data management algorithm against adaptive online adversary [30], and the results of data management problem apply to online Steiner tree problem for every graph.

Theorem 4. [30]: *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.*

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 [31]. Some existing results in the literature of data management problem include the followings.

Theorem 5. [31]: *There exists an* $O(\min\{\log(|V|),\log(\max\{\delta(v_i, v_j)\})\}$*-competitive algorithm on general graphs, for all* $v_i, v_j \in V$.

Theorem 6. [30], [32]: *There exists a graph* $G(V, E)$ *such that the competitive ratio of any randomized algorithm against oblivious adversary is in* $\Omega(\log(|V|))$.

Theorem 7. [33]: *There exist an optimal* 3*-competitive deterministic algorithm, as well as a* $(2 + 1/D)$*-competitive randomized algorithm against oblivious adversary, for trees.*

Theorem 8. [34]: *There is no deterministic algorithm and no randomized algorithm against adaptive online adversary, with* c*-competitiveness on uniform ring networks, where* $c < 4.25$ *and* $c < 3.833$*, respectively.*

Other available results include the lower bound of 3 for both deterministic [35] and randomized algorithms against adaptive online adversary [30], as well as $2 + 1/D$ for randomized algorithms against oblivious adversaries [33], in a network with only two points. As for ring networks, randomized algorithms against adaptive online adversary [30] and against oblivious adversary [33] combined, [36] 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 [33] combined, [37], and against adaptive online adversary [38], respectively.

As for the uniform ring networks, there is no randomized c-competitive algorithm against adaptive online adversary for $c < 3.833$ [34], and in addition, if there is no 'write' requests in the network, then a lower bound of 2.311 [39] and an upper bound of 3 [40] 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 page migration (also known as data migration, and file migration) problem, so that we will contemplate it with more details in the rest of these sections.

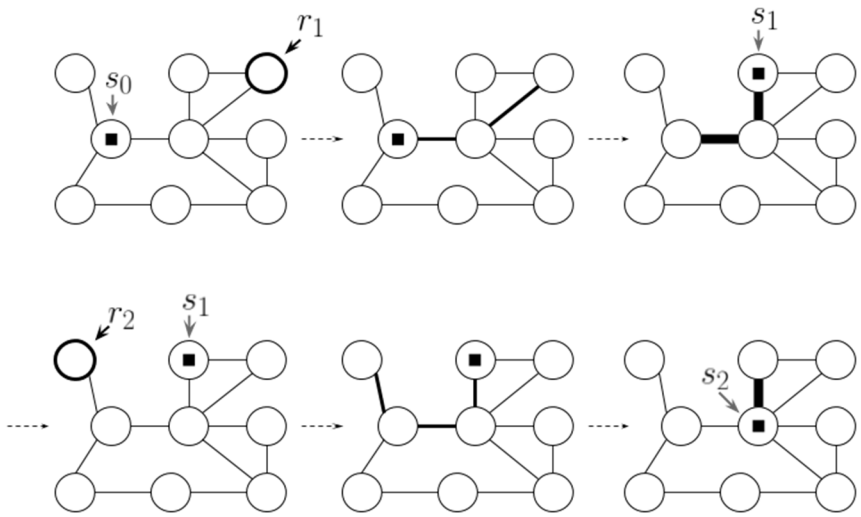


Figure 7. Representing an example of an online algorithm computing s_1 and s_2, upon r_1 and r_2, for the page migration problem. The slightly shaded edges show the paths of servicing requests, and the strongly shaded edges show the paths of migrating the page.

1.6 Page Migration Problem

In the page migration problem, there is only one single data (i.e. page or server), located on one of the nodes of a network. Online requests arrive on nodes to access the page. Before the first request, the page is located on an initial server. Upon each request, the request must be served with a cost equal to the distance between the requested point and the server location. After this service, the online algorithm decides if the server shall move (i.e. page shall migrate) to a new location of network or not. The cost of the page migration equals to the page size, multiplied by the distance between former and new locations of the page. The problem is to design an online algorithm that efficiently migrates the page, so that the total cost of services and migrations is as small as possible. Figure 7 shows an example of a scenario for this problem for a couple of requests r_1 and r_2, as well as the responses of a sample online algorithm shown as s_1 and s_2.

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.

1.6.1 Notations

To provide a formal definition for the problem, here we first summarize the notations, which will be used in the rest of article, and are subject to get slight changes according to each context.

Table 1. Notations

S	set of nodes in the network
(S, δ)	metric space of network with distance function δ
$s_0 \in S$	initial page location (i.e. initial server before the first request)
$r_i \in S$	location of ith request to access the page on s_{i-1}, where $i \geq 1$
$s_i \in S$	location of server after serving request r_i, where $i \geq 1$
$\sigma = r_1, r_2, \dots, r_n$	sequence of the first n requests
$A = s_1, s_2, \dots, s_n$	designed algorithm (i.e. response of algorithm on first n requests)
$\delta(a, b)$	distance between two nodes $a, b \in S$
D	page size (i.e. data size)
$\|S\|$	number of nodes in network; $\|S\| \geq 2$
$\Delta\|A\|$	amount of change in cost of algorithm A before and after an event
$cost_A(s_0, \sigma)$	cost of A to respond σ for the initial server location s_0
X	algorithm of an adversary; (See section 1.3.5)
c	competitiveness of X against A; (See section 1.3.4)

Here, we have mapped the metrical task system (section 1.3.2) to the page migration problem, such that s_i, r_i, and δ denote the ith state, ith task, and the function γ in the corresponding metrical task system respectively, noting that changing the state is done after processing the request.

1.6.2 Definition

For a given initial page location $s_0 \in S$, a sequence of requests $\sigma = r_1, \dots, r_n \in S$, and a page size $D \geq 1$, 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.

If the requests come one by one, then offline algorithms do not adapt, so that an online algorithm A is required to reduce the cost of producing online responses as far as possible. A does not have any information about the future requests. An adversary X generates the requests against the algorithm, for keeping the algorithm far from the minimum cost as possible. This competition is evaluated as a ratio of A's cost over X's cost and quantifies the quality of algorithm in some sense. Smaller is better for this competitiveness. Oblivious, adaptive online, and adaptive offline adversaries have different powers against an online algorithm, depending on its design as an either deterministic or randomized algorithm.

1.6.3 Uniform Model

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 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 results of the current thesis are achieved (and discussed in the next two chapters) for this *uniform model* of page migration, in which the page has a unit size. In other words, 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 for a ‘read’ or ‘write’ task.

Refer again to the example shown in Figure 7 on Page 22. Suppose the weight of all edges are equal to one, as an example. Then, the total cost incurred by the first two requests equals to 8, i.e. the number of shaded edges.

1.6.4 Review and Contribution

The starting work to efficiently cope with the page migration problem was initially reported more than 28 years ago 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 [35], and conjecturing the optimality of the lower bound.[6]

Note that *counter-based algorithms* play a significant role in designing efficient online algorithms for server problems. This kind of algorithms usually work 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. Each counter-based algorithm follows a specific strategy of decrementing and incrementing the counters for the management of resources while maintaining its competitiveness. In a private communication, a simplified version of the 3-competitive counter-based algorithm for trees fixes the bound for the total number of counters, and modifies the original algorithm [41].

---

[6] The conjecture was disproved by showing a higher lower bound of $85/27$ [45], which later improved to 3.1639 for uniform model [47], as well as $3 + \Omega(1/D)$ with respect to D [46], and disproved even in an asymptotic sense with a lower bound of $3 + 7.4 \times 10^{-6}$ [53].

For general graphs, a 4.086-competitive algorithm is known [42] that works in consecutive phases. Essentially, in this algorithm, for parameters α and β, the sequence of requests is separated into subsequences $\sigma_i = r_{i1}, r_{i2}, \dots, r_{ik}$ of a fixed length $k = \alpha \times D$. The algorithm is shown to be $\max(3 + 2/\alpha,\ 1.5\alpha + \beta/2 + 1)$-competitive, if at the end of each phase, the page migrates from its current location s_{i-1} to a location x (called local minimum) that minimizes $\sum_{j=1}^{\alpha \times D} \delta(x, r_{ij}) + \beta \times D \times \delta(s_{i-1}, x)$. The best competitiveness is achieved by the setting of $\alpha \approx 1.841$ and $\beta \approx 0.648$, which yields a 4.086-competitive ratio. This algorithm was known as the most efficient online algorithm from 17 years ago. But just recently, this ratio has been improved to 4, by a new algorithm that considers to dynamically change the length of the subsequence in each phase [43], rather than the approach of fixing a length throughout the algorithm. Moreover, in the study of randomized algorithms against adaptive online adversaries, a 3-competitive algorithm for general metric spaces was proposed [44] that matches the lower bound on two points [30].

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. [45]

Theorem 9. [45]: *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.*

For general metric spaces, a randomized $c(D)$-competitive algorithm is available [44] 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, by Theorem 9. For this reason, a deterministic 2.8-competitive algorithm exists on $\mathbb{R}^n$ in Euclidean space for $D = 1$ from 24 years ago, and our first online algorithm in this thesis is to propose a more efficient and deterministic 2.75-competitive algorithm. The algorithm is quite simple. It is discussed with details in CHAPTER 2 alongside its analysis. Roughly speaking, the algorithm maintains the server at the center of two assumptive points, and each new request grabs the farthest point. 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 [45]. 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 kth 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 in CHAPTER 2.

For the graphs restricted with only three points (a.k.a. three-node ring networks, three-node cycles), optimal 3-competitive deterministic algorithms with $D \in \{1,2\}$ [45], [46], and asymptotically optimal $(3 + 1/D)$-competitive deterministic algorithms with $D \geq 3$ [46] 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 [43] 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 [47]. In our second study of uniform model (CHAPTER 3), we propose a quite complicated deterministic algorithm with the competitiveness of 3.326 on ring networks, provided by a tight analysis.

We find it appropriate to mention about two observations on the page migration problem. Unlike the case of continues spaces under L^1 norm, we showed on Euclidean metric that the optimum cost is not always obtained by migrating the page only to the requesting points, due to the counterexample of Fermat point. As for the path network (a tree with only two leaves), the states of the art of work functions and counter-based algorithms do not behave the same. These observations played a useful and important role in conducting the research project of CHAPTER 2 in a proper way.[7]

---

[7] These observations are done as disprovements of two wrong statements, given to us to be proved. The basic motivation of considering these statements was to design work function algorithms under the norms larger than 1.

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 [48].

CHAPTER 2

Euclidean Space

The paradigm of cloud computing authorizes the requests to access the resources which are present and found everywhere. The classical problem of uniform page migration is revisited, covering such recently hot topics, where a server is able to take the natural Euclidean distances for the purpose of reducing the overhead of management.

In this chapter, 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 δ is defined by norm L^2 (i.e. Euclidean metric), which yields the length of a straight-line segment between two points as their distance. Initially, the page is located at a point s_0 of the space. 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.

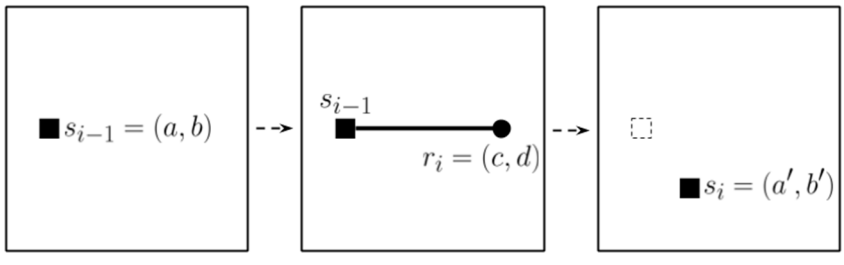


Figure 8. An example to show the behavior of an online algorithm for the problem of uniform page migration in a two-dimensional network of Euclidean space after a sample request r_i at (c, d). The server s_{i-1} is located at (a, b) upon the request, so that firstly serves the request with the service cost of $\delta(s_{i-1}, r_i) = \sqrt{(c-a)^2 + (d-b)^2}$, and then migrates to a new server location s_i at (a', b') with the migration cost of $\delta(s_{i-1}, s_i) = \sqrt{(a'-a)^2 + (b'-b)^2}$.

Figure 8 shows an example of online algorithm for this problem that incurs a cost of $\delta(s_{i-1}, r_i) + \delta(s_{i-1}, s_i)$ for a request r_i. The total cost for a sequence $\sigma = r_1, \ldots, r_n$ of n requests $\sum_{i=1}^{n}\big(\delta(s_{i-1}, r_i) + \delta(s_{i-1}, s_i)\big)$ is the aim of this problem for minimization, and we design an efficient deterministic online algorithm PQ [1], which improves the former online algorithm for this problem.

Note that 25 years ago, the former algorithm was originally proposed as a randomized algorithm [44], but later (24 years ago) derandomized to a deterministic algorithm [45], and there is no improvement known before our study. The algorithm PQ maintains two auxiliary points in the network to control its configuration in such a way that it moves the farthest point to the requesting point of network, and always migrates the page to the center of the two points.

Intuitively, the algorithm determines the location of the page using two locations from previous requests. In the following sections of this chapter, the algorithm PQ is formally defined, and analysed to bound its competitiveness.

2.1 Algorithm PQ

This algorithm 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

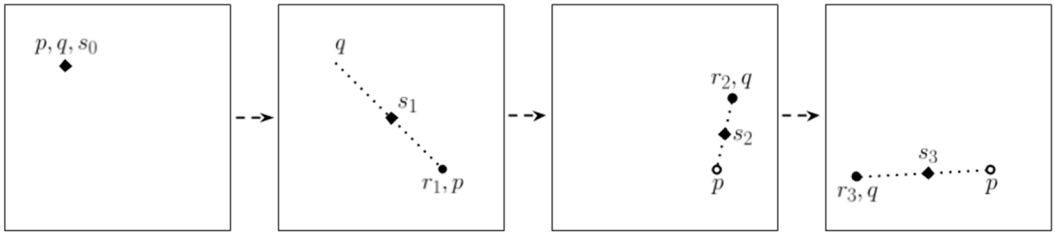


Figure 9. Showing the responses of PQ to the first three examples of requests in two dimensions

migrates its server to $s = \frac{p+q}{2}$ after p or q moves. Figure 9 shows the behavior of PQ for a sequence example of first three requests r_1, r_2, and r_3 in two-dimensional space.

2.2 Analysis of PQ

We prove the competitiveness of 2.75, claimed in Theorem 10, using a potential function Φ. As mentioned in Section 1.3.6, defining suitable events and showing that $\Delta|PQ| + \Delta\Phi \leq 2.75\Delta|X|$ for any defined partition of actions is sufficient. For the proof of the theorem, we need to show an inequality different from a triangle inequality. We separately provide a technical part for that inequality as Lemma 1. The lemma is about points computed by the algorithm.

Lemma 1. *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 $\delta(p,r) = \delta(p,q) + \delta(q,r)$.*

Proof. We may assume without loss of generality that $p = (-1,0), s = (0,0), q = (1,0)$, and $r = (\ell \cos\theta, \ell \sin\theta)$ with $0 \leq \theta \leq \pi/2$. Moreover, we fix $\delta(s,r) = \ell$ and regard g as a function of θ. The aim is to prove that g is maximized at $\theta = 0$ or $\theta = \pi/2$.

It follows that:

$$\delta(p,r) = \sqrt{(\ell \cos\theta + 1)^2 + \ell^2 \sin^2\theta} = \sqrt{\ell^2 + 2\ell \cos\theta + 1}$$

$$\delta(q,r) = \sqrt{(\ell \cos\theta - 1)^2 + \ell^2 \sin^2\theta} = \sqrt{\ell^2 - 2\ell \cos\theta + 1}$$

$$\frac{d\delta(p,r)}{d\theta} = \frac{-\ell \sin\theta}{\sqrt{\ell^2 + 2\ell \cos\theta + 1}} = -\frac{\ell \sin\theta}{\delta(p,r)}$$

and

$$\frac{d\delta(q,r)}{d\theta} = \frac{\ell \sin\theta}{\sqrt{\ell^2 - 2\ell\cos\theta + 1}} = \frac{\ell \sin\theta}{\delta(q,r)}$$

Therefore, we have

$$\frac{dg}{d\theta} = \ell \sin\theta \left\{ \frac{\rho - 1}{2} \cdot \frac{1}{\delta(p,r)} - \left(\frac{\rho}{2} - 1\right) \cdot \frac{1}{\delta(q,r)} \right\}$$

$$= \frac{(\rho - 1) \cdot \ell \cdot \sin\theta}{2\delta(q,r)} \left\{ \frac{\delta(q,r)}{\delta(p,r)} - \frac{\rho - 2}{\rho - 1} \right\}$$

$\frac{\delta(q,r)}{\delta(p,r)} = \sqrt{\frac{\ell^2 - 2\ell\cos\theta + 1}{\ell^2 + 2\ell\cos\theta + 1}}$ monotonically increases from $\frac{|\ell - 1|}{\ell + 1}$ to 1 as θ changes from 0 to $\pi/2$. If $\frac{|\ell - 1|}{\ell + 1} \geq \frac{\rho - 2}{\rho - 1}$, then $\frac{dg}{d\theta} \geq 0$ for any θ. Therefore, g is maximized at $\theta = \pi/2$. Otherwise, since $0 < \frac{\rho - 2}{\rho - 1} < 1$, there exists $0 < t < \pi/2$ such that $\sqrt{\frac{\ell^2 - 2\ell\cos t + 1}{\ell^2 + 2\ell\cos t + 1}} = \frac{\rho - 2}{\rho - 1}$. Since $\frac{dg}{d\theta} \geq 0$ for $\theta \geq t$ and $\frac{dg}{d\theta} \leq 0$ for $\theta < t$, g is maximized at $\theta = 0$ or $\theta = \pi/2$. ■

Theorem 10. *PQ is ρ-competitive for* $\rho = \frac{11}{4}$.

Proof. We use the following potential function for X's server location t, PQ's server s, and point locations p and q:

$$\Phi = \frac{\rho}{2} \cdot \big(\delta(p,t) + \delta(q,t)\big) - \frac{\rho - 2}{2} \cdot \delta(p,q)$$

We separate the online events into two parts. The first is to consider only the migration costs incurred by X's server, and the second is to consider the service costs incurred by X together with the migration and service costs incurred by PQ. It is sufficient to show that the inequality $\Delta|PQ| + \Delta\Phi \leq \rho \cdot \Delta|\mathrm{X}|$ follows in both parts, upon each request r.

Part 1. The migration of X's server from t to t' induces a change of $\delta(t,t')$ to the total cost of the optimal algorithm X but no change to the cost incurred by PQ. The

total change of Φ is then $\frac{\rho}{2} \cdot \big(\delta(p,t') - \delta(p,t) + \delta(q,t') - \delta(q,t)\big)$. Therefore, it is sufficient to show the following inequality:

$$\frac{\rho}{2}\delta\big(p,t^{'}\big) - \frac{\rho}{2}\delta(p,t) + \frac{\rho}{2}\delta\big(q,t^{'}\big) - \frac{\rho}{2}\delta(q,t) \leq \frac{\rho}{2}\delta\big(t,t^{'}\big) + \frac{\rho}{2}\delta\big(t,t^{'}\big)$$

This follows by the symmetry of the distance function ($\delta(a,b) = \delta(b,a)$) and the triangle inequalities $\delta(p,t') \leq \delta(t,t') + \delta(p,t)$ and $\delta(q,t') \leq \delta(t,t') + \delta(q,t)$.

Part 2. We may assume, without loss of generality, that $\delta(p,r) \geq \delta(q,r)$. By this assumption, PQ moves p to r. Since PQ maintains its server at the center of p and q, the migration cost incurred by PQ is then $\frac{1}{2}\delta(p,r)$. For this part, we have the following equalities:

$$\Delta|PQ| = \delta(s,r) + \frac{1}{2}\delta(p,r)$$

$$\Delta\Phi = \frac{\rho}{2} \cdot \big(\delta(r,t) - \delta(p,t)\big) + \Big(\frac{\rho}{2} - 1\Big) \cdot \big(\delta(p,q) - \delta(q,r)\big)$$

$$\Delta|X| = \delta(t,r)$$

Therefore, we shall show the following inequality:

$$\delta(s,r) + \frac{1}{2}\delta(p,r) + \frac{\rho}{2} \cdot \big(\delta(r,t) - \delta(p,t)\big) + \Big(\frac{\rho}{2} - 1\Big) \cdot \big(\delta(p,q) - \delta(q,r)\big) - \rho\delta(t,r) \leq 0$$

Since $\delta(r,t) + \delta(p,t) \geq \delta(p,r)$, it is sufficient to show:

$$\delta(s,r) - \Big(\frac{\rho}{2} - \frac{1}{2}\Big) \cdot \delta(p,r) - \Big(\frac{\rho}{2} - 1\Big) \cdot \big(\delta(q,r) - \delta(p,q)\big) \leq 0. \quad (1)$$

This follows for the cases $p = q$ and $q = r$, because $s = p = q$ and $\delta(s,r) = \frac{1}{2}\,\delta(p,r)$, respectively. We assume $p \neq q$ and $q \neq r$. It is sufficient to show that the maximum value of the left-hand side of Equation (1) is less than or equal to zero. If we regard p, q, r, and s as vectors in $\mathbb{R}^n$, then at most three vectors of them, say p, q, and

r, are independent. Therefore, the points p, q, r, and s are on a plane in $\mathbb{R}^n$. Applying Lemma 1 on this plane, the left-hand side of Equation (1) is maximized in one of three situations. Situation 1: $\delta(p,r) = \delta(q,r)$, Situation 2: $\delta(p,q) = \delta(p,r) + \delta(r,q)$, or Situation 3: $\delta(p,r) = \delta(p,q) + \delta(q,r)$. We proceed to show inequality (1) for each of these three situations.

Situation 1: Substitution of $\delta(p,q)$ by $2 \cdot \delta(s,q)$ in Equation (1) reveals the following inequality.

$$\delta(s,r) + (\rho - 2) \cdot \delta(s,q) \leq \left(\rho - \frac{3}{2}\right) \cdot \delta(q,r)$$

By applying $\rho = \frac{11}{4}$ and dividing both sides by $\frac{5}{4}$, it is sufficient to show

$$\frac{4}{5}\,\delta(s,r) + \frac{3}{5}\,\delta(s,q) \leq \delta(q,r) \tag{2}$$

Since $\big(\delta(q,r)\big)^2 = \big(\delta(s,r)\big)^2 + \big(\delta(s,q)\big)^2$, Equation (2) can be written as:

$$\frac{4}{5}\,\delta(s,r) + \frac{3}{5}\sqrt{\big(\delta(q,r)\big)^2 - \big(\delta(s,r)\big)^2} \leq \delta(q,r).$$

By taking the derivative with respect to $\delta(s,r)$ on the plane containing q, r, and s, the left-hand side of Equation (2) is maximized at $\delta(s,r) = \frac{4}{5}\,\delta(q,r)$ and $\delta(s,q) = \frac{3}{5}\delta(q,r)$. Therefore, Equation (2) follows.

Situation 2: The inequality in Equation (1) can be rewritten as:

$$\delta(s,r) - \left(\frac{\rho}{2} - 1\right)\big(\delta(p,r) + \delta(q,r) - \delta(p,q)\big) \leq \frac{1}{2}\delta(p,r) \tag{3}$$

In this situation, we recall that p, s, r, and q are all located on the same line segment. Since $\delta(p,q) \geq \delta(p,r)$, and s is located at the center of p and q, it follows that $\frac{1}{2}\delta(p,r) = \delta(p,s) = \delta(s,q) = \delta(s,r) + \delta(r,q) \geq \delta(s,r)$. Since $\delta(p,r) + \delta(q,r) - \delta(p,q) = 0$, the inequality in Equation (3) follows.

Situation 3: Since $\rho = \frac{11}{4}$, we rewrite Equation (1) as:

$$\delta(s,r) - \frac{7}{8}\delta(p,r) - \frac{3}{8}\delta(q,r) + \frac{3}{8}\delta(p,q) \leq 0$$

The points $p, s, r,$ and q are also located on the same line segment in this situation. By $\delta(p,r) = \delta(p,q) + \delta(q,r)$, we have $\delta(s,r) - \frac{1}{2}\delta(p,q) - \frac{5}{4}\delta(q,r) \leq 0$. By $\delta(p,q) = 2\,\delta(s,q)$ we have $\delta(s,r) - \delta(s,q) - \frac{5}{4}\delta(q,r) \leq 0$. In addition, by $\delta(s,r) = \delta(s,q) + \delta(q,r)$ we have $\delta(q,r) - \frac{5}{4}\delta(q,r) \leq 0$.

Therefore, the proof completes. ■

2.3 Bound of Analysis

In this section, we show that the exact competitiveness of the algorithm is greater than 2.732. We introduce an adversary through Theorem 11 and prove the existence of such a lower bound. The adversary makes a special sequence of requests on $\mathbb{R}^2$ against the proposed algorithm. The requests are given at vertices of a triangle that tends to be equilateral. After the second request, the server repeatedly migrates between the center points of two sides of the triangle.

Theorem 11. For a sufficiently large integer n, there exist a request sequence $\sigma = r_1, \ldots, r_{2n+1}$, three request locations $a, b,$ and s on a plane, and the initial server location s, such that $cost_{PQ}(s,\sigma) \geq \rho \cdot cost_{OPT}(s,\sigma)$ for $\rho = 1 + \sqrt{3} \approx 2.732$.

Proof. We describe our approach before more technical discussion in the subsequent paragraphs. In Figure 10, we illustrate the behavior of PQ against requests by our adversary. We assume that $\delta(s,b)$ is slightly larger than $\delta(s,a) = \delta(b,a)$. Points u and v are the centers of a and s, and a and b, respectively. For requests $a, b, s, b, s, b, s, \ldots$, PQ migrates its server along the dashed line, because p does not

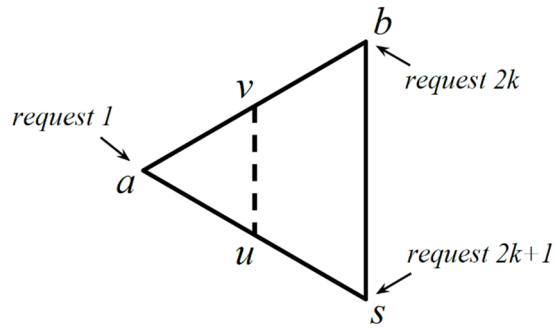


Figure 10. The request sequence $a, b, s, b, s, b, s, \ldots$ to access the page at the server; $k \geq 1$.

move from a. A request occurs at b when the server is at u, and a request occurs at s when the server is at v. Note that a, b, and s tend to be the vertices of an equilateral triangle as $\delta(s, b)$ approaches $\delta(s, a) = \delta(b, a)$.

For $r_1 = a$, $r_{2k} = b$, and $r_{2k+1} = s$, where $1 \leq k \leq n$ and a, b, and s are the vertices of an equilateral triangle with a unit side length, the cost of the optimal algorithm $cost_{OPT}(s, \sigma)$ is at most $n + 1$ by keeping the server at the initial location s.

The adversary infinitesimally perturbs the distances by slightly increasing the distance between s and b. Upon the first request at a, PQ serves the request by the cost of $\delta(s, a)$ and migrates the server to u, which is the center of s and a, by the cost of $\delta(s, u)$. This is because point p at s moves to a. The $2k$th request at b is served with the cost of $\delta(u, b)$, causing point q at s to move to b, and the server migrates to v, which is at the center of a and b, with the cost of $\delta(u, v)$. The $(2k + 1)$st request at s is served with the cost of $\delta(v, s)$. The point q at b moves to s; hence, the server migrates to u with the cost of $\delta(u, v)$.

We compute the total cost of PQ by using the distances of the unperturbed triangle, since the actual distances (and hence the actual costs of PQ and OPT) may differ by an infinitesimally small amount from these computations. Therefore, we have

$$cost_{PQ}(s,\sigma) = \delta(s,a) + \delta(s,u) + n \cdot \big(\delta(u,b) + \delta(u,v) + \delta(v,s) + \delta(v,u)\big)$$

$$\approx 1.5 + \big(\sqrt{3}+1\big)n$$

Since

$$\lim_{n\to\infty} \frac{1.5 + \big(\sqrt{3}+1\big)n}{n+1} = \sqrt{3}+1$$

then $\frac{cost_{PQ}(s,\sigma)}{cost_{OPT}(s,\sigma)}$ is at least 2.732. This completes the proof. ■

2.4 Remarks

In this chapter, we proposed a deterministic algorithm for the uniform page migration problem in Euclidean space. In this problem, the server is able to migrate in any direction and choose any destination of the space. The 2.75-competitiveness of the algorithm is an improvement on the former 2.8 ratio. An adversary was found to express a lower bound of 2.732 for the algorithm. If possible, one could seek to find an algorithm to cover the page migration problem of general page size 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 norms other than the Euclidean and Manhattan ones remains an open problem in this research area.

CHAPTER 3

Ring Networks

This chapter explores the problem of uniform page migration in ring networks. A ring network is a connected graph, in which each node is connected with exactly two other nodes. In this problem, one of the nodes in a given network holds the page. This node is called the server and the page is a non-duplicable data in the network. Requests are issued on nodes to access the page one after another. Every time a new request is issued, the server must serve the request and may migrate to another node before the next request arrives. A service costs the distance between the server and the requesting node, and the migration costs the distance of the migration. Figure 11 shows an example to deal with the requests on a ring network, in which shaded parts represent the service and migration costs.

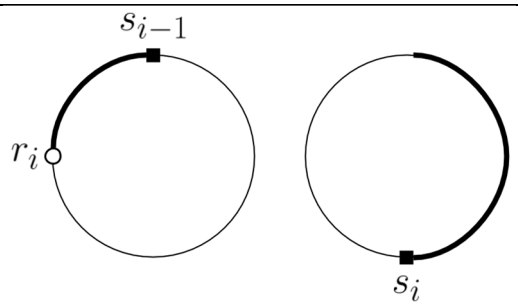


Figure 11. An example of online algorithm to serve a request r_i and migrate to s_i in a ring network

The problem is to minimize the total costs of services and migrations. A deterministic 3.326-competitive algorithm TriAct, improving the current best upper bound is designed, and provided that this ratio is tight for our complicated algorithm.

3.1 Algorithm TriAct

We propose a deterministic algorithm, called TriAct, on a ring network as defined in Figure 12. We set $r_0 = s_0$, the initial location of the server in the ring network. For $i \geq 1$, upon the request r_i at any node, there are three choices for the server to act according to the algorithm. The server keeps its current location at node s_{i-1} or migrates to either r_i or r_{i-1}. The decision is based on the distances among s_{i-1}, r_i, and r_{i-1}. There are six different cases to determine which action must be done.

$L \leftarrow$ network length
$x \leftarrow \delta(s_{i-1}, r_{i-1})$
$y \leftarrow \delta(s_{i-1}, r_i)$
$z \leftarrow \delta(r_{i-1}, r_i)$

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}

server migrates from s_{i-1} to s_i

Figure 12. Definition of Algorithm TriAct upon a request r_i to access the page at the server s_{i-1} on a ring network

In the ring topology, there are exactly two paths between each pair of nodes a, b. Let $\pi(a, b)$ denote a shortest path between a, b. The length of $\pi(a, b)$ equals to the distance $\delta(a, b)$. Let L denote the length of the ring, then it is obvious that $0 \leq \delta(a, b) \leq L/2$. In our calculations, we set $x = \delta(s_{i-1}, r_{i-1})$, $y = \delta(s_{i-1}, r_i)$, and $z = \delta(r_{i-1}, r_i)$.

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. Figure 13 shows an example for each of these three cases.

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. The conditions of Cases D, E, and F are shown in Figure 14.

3.2 Analysis of TriAct

We show that TriAct is ρ-competitive with $\rho \approx 3.3258$ in Theorem 12 below. The exact value of ρ is provided in Appendix A. We use a potential function Φ to prove the theorem. We 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 proof for Cases A-E are straightforward. Our analysis for Case F uses an aggregation technique, because we need to consider two consecutive requests in that case to complete the proof.

Theorem 12. *TriAct is ρ-competitive for $D = 1$, where $\rho \approx 3.326$* is the positive solution of $-\rho^4 + 4\rho^3 + \rho^2 - 18\rho + 24 = 0$.

Proof. We use the potential function Φ, for X's server locations $t_1, \ldots, t_k$, TriAct's server locations $s_1, \ldots, s_k$, and request locations $r_1, \ldots, r_k$. We define

$$\Phi(s_i, r_i, t_i) = \frac{\rho}{2} \cdot \big(\delta(s_i, t_i) + \delta(r_i, t_i)\big) + \Big(\frac{\rho}{2} - 1\Big)\,\delta(s_i, r_i)$$

We separately consider the events in two parts. The first includes the migration costs incurred by X, and the second 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

$$\Delta_2 = \delta(s_{i-1}, r_i) + \delta(s_{i-1}, s_i) + \Phi(s_i, r_i, t_{i-1})$$
$$-\Phi(s_{i-1}, r_{i-1}, t_{i-1}) - \rho \cdot \delta(t_{i-1}, r_i).$$

In order to prove that $\Delta|\mathrm{TriAct}| + \Delta\Phi - \rho \cdot \Delta|\mathrm{X}| \leq 0$ follows in both parts, it suffices to show that $\Delta_1 \leq 0$ and $\Delta_2 \leq 0$.

Analysis of part 1:

For the first part,

$$\Delta_1 = \tfrac{\rho}{2} \cdot \big(\delta(s_i, t_i) - \delta(s_i, t_{i-1}) + \delta(r_i, t_i) - \delta(r_i, t_{i-1})\big) - \rho \cdot \delta(t_{i-1}, t_i).$$

By triangle inequality, we have

$$\Delta_1 \leq \frac{\rho}{2} \cdot \big(\delta(t_{i-1}, t_i) + \delta(t_{i-1}, t_i)\big) - \rho \cdot \delta(t_{i-1}, t_i) = 0.$$

Analysis of part 2:

For the second part, we have

$$\begin{aligned}
\Delta_2 = {} & \delta(s_{i-1}, r_i) + \delta(s_{i-1}, s_i) + \frac{\rho}{2}\delta(s_i, t_{i-1}) + \frac{\rho}{2}\delta(r_i, t_{i-1}) \\
& + \Big(\frac{\rho}{2} - 1\Big)\delta(s_i, r_i) - \frac{\rho}{2}\delta(s_{i-1}, t_{i-1}) - \frac{\rho}{2}\delta(r_{i-1}, t_{i-1}) \\
& - \Big(\frac{\rho}{2} - 1\Big)\delta(s_{i-1}, r_{i-1}) - \rho \cdot \delta(t_{i-1}, r_i) \\
= {} & \delta(s_{i-1}, r_i) + \delta(s_{i-1}, s_i) + \frac{\rho}{2}\delta(s_i, t_{i-1}) - \frac{\rho}{2}\delta(r_i, t_{i-1}) \\
& + \Big(\frac{\rho}{2} - 1\Big)\delta(s_i, r_i) - \frac{\rho}{2}\delta(s_{i-1}, t_{i-1}) - \frac{\rho}{2}\delta(r_{i-1}, t_{i-1}) \\
& - \Big(\frac{\rho}{2} - 1\Big)\delta(s_{i-1}, r_{i-1}).
\end{aligned} \tag{4}$$

TriAct has three choices of s_i, i.e., r_i, or r_{i-1}, or s_{i-1}. For these choices, we separately derive upper bounds of Δ_2.

Upper bound of Δ_2 for the action $s_i \leftarrow r_i$:

For the action of migrating the server to the current request location, it follows from (4):

$$\begin{aligned}\Delta_2 = {} & \delta(s_{i-1}, r_i) + \delta(s_{i-1}, r_i) + \frac{\rho}{2}\delta(r_i, t_{i-1}) - \frac{\rho}{2}\delta(r_i, t_{i-1}) \\ & + \left(\frac{\rho}{2} - 1\right)\delta(r_i, r_i) - \frac{\rho}{2}\delta(s_{i-1}, t_{i-1}) - \frac{\rho}{2}\delta(r_{i-1}, t_{i-1}) \\ & - \left(\frac{\rho}{2} - 1\right)\delta(s_{i-1}, r_{i-1}) \\ = {} & 2\delta(s_{i-1}, r_i) - \frac{\rho}{2}\delta(s_{i-1}, t_{i-1}) - \frac{\rho}{2}\delta(r_{i-1}, t_{i-1}) \\ & - \left(\frac{\rho}{2} - 1\right)\delta(s_{i-1}, r_{i-1}) \\ \leq {} & 2\delta(s_{i-1}, r_i) - \frac{\rho}{2}\delta(s_{i-1}, r_{i-1}) - \left(\frac{\rho}{2} - 1\right)\delta(s_{i-1}, r_{i-1}) \\ = {} & (1-\rho)\delta(s_{i-1}, r_{i-1}) + 2\delta(s_{i-1}, r_i) = (1-\rho)x + 2y. \end{aligned} \quad (5)$$

Here, we used the triangle inequality $\delta(s_{i-1}, t_{i-1}) + \delta(r_{i-1}, t_{i-1}) \geq \delta(s_{i-1}, r_{i-1})$. We note that this upper bound of Δ_2 is used for Cases A and E.

Upper bound of Δ_2 for the action $s_i \leftarrow r_{i-1}$:

For the action of migrating the server to the previous request location, it follows from (4) that:

$$\begin{aligned}\Delta_2 = {} & \delta(s_{i-1}, r_i) + \delta(s_{i-1}, r_{i-1}) + \frac{\rho}{2}\delta(r_{i-1}, t_{i-1}) - \frac{\rho}{2}\delta(r_i, t_{i-1}) \\ & + \left(\frac{\rho}{2} - 1\right)\delta(r_{i-1}, r_i) - \frac{\rho}{2}\delta(s_{i-1}, t_{i-1}) - \frac{\rho}{2}\delta(r_{i-1}, t_{i-1}) \\ & - \left(\frac{\rho}{2} - 1\right)\delta(s_{i-1}, r_{i-1}) \\ = {} & \delta(s_{i-1}, r_i) + \left(2 - \frac{\rho}{2}\right)\delta(s_{i-1}, r_{i-1}) - \frac{\rho}{2}\delta(t_{i-1}, r_i) \\ & + \left(\frac{\rho}{2} - 1\right)\delta(r_{i-1}, r_i) - \frac{\rho}{2}\delta(s_{i-1}, t_{i-1}) \end{aligned}$$

$$\leq \left(2-\frac{\rho}{2}\right)\delta(s_{i-1},r_{i-1})+\left(1-\frac{\rho}{2}\right)\delta(s_{i-1},r_i)+\left(\frac{\rho}{2}-1\right)\delta(r_{i-1},r_i)$$

$$= \left(2-\frac{\rho}{2}\right)x+\left(1-\frac{\rho}{2}\right)y+\left(\frac{\rho}{2}-1\right)z. \quad (6)$$

Here, we used the triangle inequality $\delta(t_{i-1},r_i)+\delta(s_{i-1},t_{i-1})\geq\delta(s_{i-1},r_i)$. We note that this upper bound of Δ_2 is used for Cases B and D.

Upper bound of Δ_2 for the action $s_i \leftarrow s_{i-1}$:

For the action of no migration, it follows from (4) that

$$\begin{aligned}\Delta_2 = {} & \delta(s_{i-1},r_i)+\delta(s_{i-1},s_{i-1})+\frac{\rho}{2}\delta(s_{i-1},t_{i-1})-\frac{\rho}{2}\delta(r_i,t_{i-1})\\ & +\left(\frac{\rho}{2}-1\right)\delta(s_{i-1},r_i)-\frac{\rho}{2}\delta(s_{i-1},t_{i-1})-\frac{\rho}{2}\delta(r_{i-1},t_{i-1})\\ & -\left(\frac{\rho}{2}-1\right)\delta(s_{i-1},r_{i-1})\\ = {} & \frac{\rho}{2}\delta(s_{i-1},r_i)-\frac{\rho}{2}\delta(r_i,t_{i-1})-\frac{\rho}{2}\delta(r_{i-1},t_{i-1})\\ & -\left(\frac{\rho}{2}-1\right)\delta(s_{i-1},r_{i-1})\\ \leq {} & \frac{\rho}{2}\delta(s_{i-1},r_i)-\frac{\rho}{2}\delta(r_{i-1},r_i)-\left(\frac{\rho}{2}-1\right)\delta(s_{i-1},r_{i-1})\\ = {} & \left(1-\frac{\rho}{2}\right)x+\frac{\rho}{2}y-\frac{\rho}{2}z. \quad (7)\end{aligned}$$

Here, we used the triangle inequality $\delta(r_i,t_{i-1})+\delta(r_{i-1},t_{i-1})\geq\delta(r_{i-1},r_i)$. We note that this upper bound of Δ_2 is used for Cases C and F.

Analysis for Cases A, B, and C:

In Case A, since $y=x-z\ \leq\ x$ and $\rho>3$, it follows from (5) that

$$\Delta_2 \leq (1-\rho)x+2y \leq (3-\rho)x < 0. \quad (8)$$

In Case B, since $z=y-x$ and $\rho>3$, it follows from (6) that

$$\Delta_2 \le \left(2-\frac{\rho}{2}\right)x + \left(1-\frac{\rho}{2}\right)y + \left(\frac{\rho}{2}-1\right)z$$

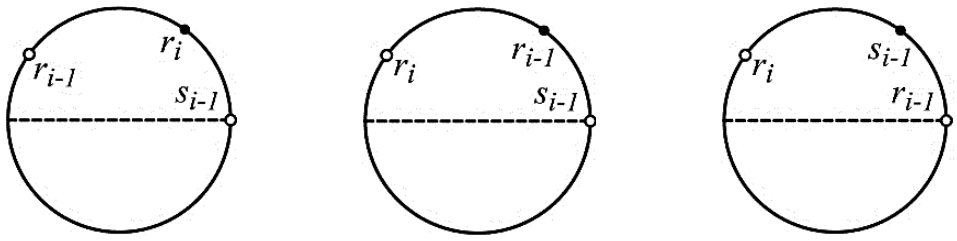
$$= \left(2-\frac{\rho}{2}\right)x + \left(1-\frac{\rho}{2}\right)y + \left(\frac{\rho}{2}-1\right)(y-x) = (3-\rho)x < 0. \quad (9)$$

In Case C, since $z = x + y$ and $\rho > 1$, it follows from (7) that

$$\Delta_2 \le \left(1-\frac{\rho}{2}\right)x + \frac{\rho}{2}y - \frac{\rho}{2}z = \left(1-\frac{\rho}{2}\right)x + \frac{\rho}{2}y - \frac{\rho}{2}(x+y)$$

$$= (1-\rho)x < 0. \quad (10)$$

Therefore, Theorem 12 holds in Cases A, B, and C.



Case A

server migrates from s_{i-1} to r_i

Case B

server migrates from s_{i-1} to r_{i-1}

Case C

no migration from s_{i-1}

Figure 13. Examples of Cases A, B, and C. Filled circles represent destinations of the server, i.e., s_i, upon r_i

Analysis for Cases D and E:

For Cases D and E, we have $z = L - x - y$. The conditions of these cases are defined using four functions y_1, y_2, y_3, and y_4 of x, where

$$y_1 = -\frac{\rho-3}{\rho-2}x + \frac{L}{2}, \quad (11)$$

$$y_2 = \frac{2}{\rho}x + \frac{\rho-2}{\rho}\cdot\frac{L}{2}, \quad (12)$$

$$y_3 = \frac{\rho-1}{2}x, \text{ and} \quad (13)$$

$$y_4 = \frac{\rho}{\rho-2}\left(\frac{L}{2}-x\right). \quad (14)$$

In Figure 14, the separate regions represent the conditions of Cases D, E, and F.

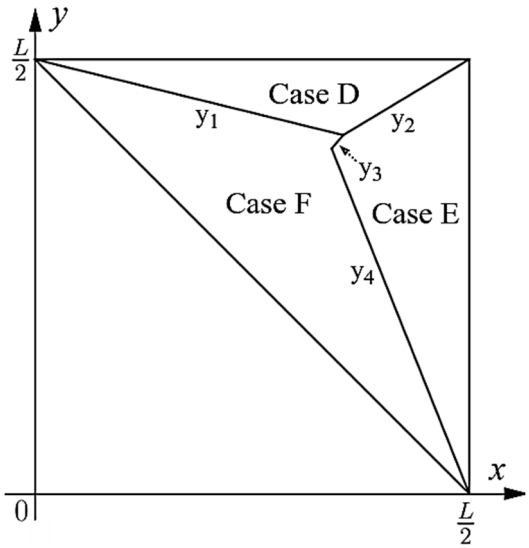


Figure 14. Conditions of Cases D, E, and F

In Case D, since $y \geq y_1$, it follows from (6) and (11) that

$$\begin{aligned} \Delta_2 &\leq \left(2-\frac{\rho}{2}\right)x + \left(1-\frac{\rho}{2}\right)y + \left(\frac{\rho}{2}-1\right)(L-x-y) \\ &= (3-\rho)x + (2-\rho)y + \left(\frac{\rho}{2}-1\right)L \\ &\leq (3-\rho)x + (2-\rho)\left(-\frac{\rho-3}{\rho-2}x + \frac{L}{2}\right) + \left(\frac{\rho}{2}-1\right)L \\ &= 0. \end{aligned} \tag{15}$$

In Case E, since $y \leq y_3$, it follows from (5) and (13) that

$$\begin{aligned} \Delta_2 &\leq (1-\rho)x + 2y \\ &\leq (1-\rho)x + 2\left(\frac{\rho-1}{2}x\right) = 0. \end{aligned} \tag{16}$$

Therefore, Theorem 12 holds in Cases D and E.

Analysis for Case F:

For Case F, we also have $z = L - x - y$ and the conditions of Case F as shown in Figure 14. It follows from (7) that

$$\Delta_2 \le \left(1-\frac{\rho}{2}\right)x+\frac{\rho}{2}y-\frac{\rho}{2}z$$

$$=\left(1-\frac{\rho}{2}\right)x+\frac{\rho}{2}y-\frac{\rho}{2}(L-x-y)=\rho y+x-\frac{\rho}{2}L. \quad (17)$$

This is at most 0 if $y \le y_5$, where

$$y_5=\frac{L}{2}-\frac{x}{\rho}. \quad (18)$$

Therefore, Theorem 12 holds if $y \le y_5$ in Case F. However, if $y > y_5$, i.e., if x and y are in the grey region in Figure 15, then $\Delta_2 > 0$. Instead of bounding Δ_2, for the case $y > y_5$, we bound the aggregation of Δ_2 and Δ_2' , which is defined as the value of Δ_2 for the next request r_{i+1}. Specifically,

$$\Delta_2' = \delta(s_i, r_{i+1}) + \delta(s_i, s_{i+1}) + \Phi(s_{i+1}, r_{i+1}, t_i) - \Phi(s_i, r_i, t_i) - \rho \cdot \delta(t_i, r_{i+1}).$$

It should be noted that Theorem 12 holds if $\Delta_2 + \Delta_2' < 0$ for all six cases of r_{i+1}, and for all x and y in the grey region in Figure 15. We also note that

$$s_i = s_{i-1} \quad (19)$$

in Case F for r_i. In the rest of the proof, we show $\Delta_2 + \Delta_2' \le 0$ for all six cases of r_{i+1} and for all x and y with $\Delta_2 > 0$.

In Cases A, B, and C for r_{i+1}, it follows from (8), (9), (10) and (19) that $\Delta_2' \le (3-\rho)\delta(s_i, r_i) = (3-\rho)\delta(s_{i-1}, r_i) = (3-\rho)y$. Therefore, since $y \le y_1$, it follows from (17) and (11) that

$$\Delta_2 + \Delta_2' \le \rho y + x - \frac{\rho}{2}L - (\rho - 3)y = 3y + x - \frac{\rho}{2}L$$

$$\le 3\left(-\frac{\rho - 3}{\rho - 2}x + \frac{L}{2}\right) + x - \frac{\rho}{2}L$$

$$= \frac{7 - 2\rho}{\rho - 2}x - (\rho - 3)\frac{L}{2}$$

$$\le \frac{7 - 2\rho}{\rho - 2} \cdot \frac{L}{2} - (\rho - 3)\frac{L}{2}$$

$$= -\frac{\rho^2 - 3\rho - 1}{\rho - 2} \cdot \frac{L}{2},$$

which is negative since $\rho > \frac{3+\sqrt{13}}{2} \approx 3.303$.

In Case D for r_{i+1}, it follows from (15) and (19) that

$$\Delta_2' \le -(\rho - 2)\delta(s_i, r_{i+1}) - (\rho - 3)\delta(s_i, r_i) + \left(\frac{\rho}{2} - 1\right)L$$

$$= -(\rho - 2)\delta(s_i, r_{i+1}) - (\rho - 3)\delta(s_{i-1}, r_i) + \left(\frac{\rho}{2} - 1\right)L$$

$$= -(\rho - 2)\delta(s_i, r_{i+1}) - (\rho - 3)y + \left(\frac{\rho}{2} - 1\right)L. \quad (20)$$

Moreover, it follows from (12) and (19) that

$$\delta(s_i, r_{i+1}) \ge \frac{2}{\rho}\delta(s_i, r_i) + \frac{\rho - 2}{\rho} \cdot \frac{L}{2} = \frac{2}{\rho}\delta(s_{i-1}, r_i) + \frac{\rho - 2}{\rho} \cdot \frac{L}{2}$$

$$= \frac{2}{\rho}y + \frac{\rho - 2}{\rho} \cdot \frac{L}{2}. \quad (21)$$

Therefore, it follows from (17), (20), and (21) that

$$\Delta_2 + \Delta_2' \le \rho y + x - \frac{\rho}{2}L - (\rho - 2)\delta(s_i, r_{i+1}) - (\rho - 3)y$$

$$+ \left(\frac{\rho}{2} - 1\right)L$$

$$\le 3y + x - L - (\rho - 2)\left(\frac{2}{\rho}y + \frac{\rho - 2}{\rho} \cdot \frac{L}{2}\right)$$

$$= \frac{\rho+4}{\rho} y + x - \frac{\rho^2 - 2\rho + 4}{\rho} \cdot \frac{L}{2}. \tag{22}$$

The value $C = \frac{\rho+4}{\rho} y + x$ is maximized at p, at which y_1 and y_3 intersect in Figure 15. This is because for the function $y = -\frac{\rho}{\rho+4} x + \frac{\rho}{\rho+4} C$, its slope $-\frac{\rho}{\rho+4}$ is negative and less than the slope $-\frac{\rho-3}{\rho-2}$ of y_1 for $\rho \approx 3.326$. Therefore, we have

$$\begin{aligned} \Delta_2 + \Delta_2' &\leq \frac{\rho+4}{\rho} \cdot \frac{\rho^2 - 3\rho + 2}{\rho^2 - \rho - 4} \cdot \frac{L}{2} + \frac{\rho - 2}{\rho^2 - \rho - 4} L - \frac{\rho^2 - 2\rho + 4}{\rho} \cdot \frac{L}{2} \\ &= \frac{-\rho^4 + 4\rho^3 + \rho^2 - 18\rho + 24}{\rho(\rho^2 - \rho - 4)} \cdot \frac{L}{2}, \end{aligned}$$

which is equal to 0 by the assumption of ρ in Theorem 12.

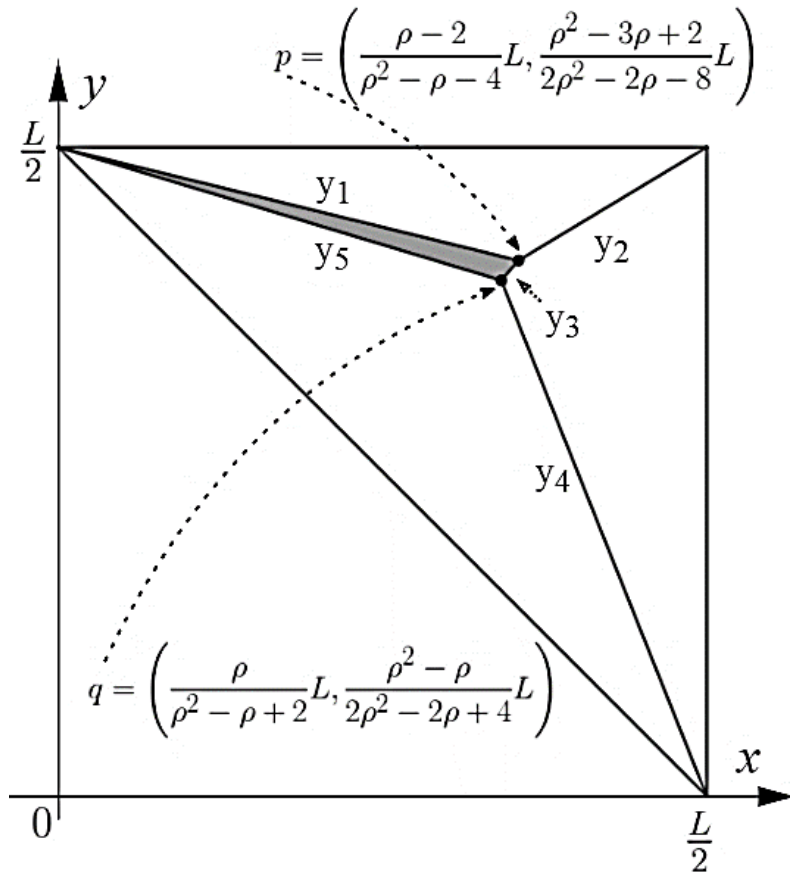


Figure 15. Representation of positive region of Δ_2 in Case F

In Case E for r_{i+1}, it follows from (16) and (19) that

$$\Delta_2' \le 2\delta(s_i, r_{i+1}) - (\rho-1)\delta(s_i, r_i) = 2\delta(s_i, r_{i+1}) - (\rho-1)\delta(s_{i-1}, r_i)$$
$$= 2\delta(s_i, r_{i+1}) - (\rho-1)y. \tag{23}$$

Moreover, it follows from (12) and (19) that

$$\delta(s_i, r_{i+1}) \le \frac{2}{\rho}\delta(s_i, r_i) + \frac{\rho-2}{\rho}\cdot\frac{L}{2}$$
$$= \frac{2}{\rho}\delta(s_{i-1}, r_i) + \frac{\rho-2}{\rho}\cdot\frac{L}{2}$$
$$\le \frac{2}{\rho}y + \frac{\rho-2}{\rho}\cdot\frac{L}{2}. \tag{24}$$

Therefore, it follows from (17), (23), and (24) that

$$\Delta_2 + \Delta_2' \le \rho y + x - \frac{\rho}{2}L + 2\delta(s_i, r_{i+1}) - (\rho-1)y$$
$$\le y + x - \frac{\rho}{2}L + 2\left(\frac{2}{\rho}y + \frac{\rho-2}{\rho}\cdot\frac{L}{2}\right)$$
$$= \frac{\rho+4}{\rho}y + x - \frac{\rho^2-2\rho+4}{\rho}\cdot\frac{L}{2},$$

which is identical with (22). Hence, we can obtain $\Delta_2 + \Delta_2' \le 0$ as done for (22).

In Case F for r_{i+1}, it follows from (17) and (19) that

$$\Delta_2' \le \rho\delta(s_i, r_{i+1}) + \delta(s_i, r_i) - \frac{\rho}{2}L$$
$$= \rho\delta(s_i, r_{i+1}) + \delta(s_{i-1}, r_i) - \frac{\rho}{2}L$$
$$= \rho\delta(s_i, r_{i+1}) + y - \frac{\rho}{2}L. \tag{25}$$

For x and y in the grey region in Figure 15, it follows that $y \ge \frac{\rho^2-\rho}{\rho^2-\rho+2}\cdot\frac{L}{2}$, 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 $\frac{\rho-2}{\rho^2-\rho-4}L$ of p. We can verify this by $\delta(s_i, r_i) = \delta(s_{i-1}, r_i) = y$ and

$$\frac{\rho^2-\rho}{\rho^2-\rho+2}\cdot\frac{L}{2}-\frac{\rho-2}{\rho^2-\rho-4}L$$

$$=\frac{\rho^4-4\rho^3+3\rho^2-4\rho+8}{(\rho-2)(\rho+1)(\rho^2-\rho-4)}\cdot\frac{L}{2}$$

$$=\frac{(\rho^4-4\rho^3-\rho^2+18\rho-24)+(4\rho^2-22\rho+32)}{(\rho-2)(\rho+1)(\rho^2-\rho-4)}\cdot\frac{L}{2}$$

$$=\frac{4\rho^2-22\rho+32}{(\rho-2)(\rho+1)(\rho^2-\rho-4)}\cdot\frac{L}{2},$$

which is positive for $\rho>\frac{1+\sqrt{17}}{2}\approx 2.56$. Here, we have used the assumption $\rho^4-4\rho^3-\rho^2+18\rho-24=0$ in Theorem 12. Therefore, it follows from (14) that

$$\begin{aligned}\delta(s_i,r_{i+1}) &\le \frac{\rho}{\rho-2}\left(\frac{L}{2}-\delta(s_i,r_i)\right)\\ &=\frac{\rho}{\rho-2}\left(\frac{L}{2}-\delta(s_{i-1},r_i)\right)\\ &=\frac{\rho}{\rho-2}\left(\frac{L}{2}-y\right).\end{aligned}\tag{26}$$

Therefore, it follows from (17), (25), and (26) that

$$\begin{aligned}\Delta_2+\Delta_2' &\le \rho y+x-\frac{\rho}{2}L+\rho\delta(s_i,r_{i+1})+y-\frac{\rho}{2}L\\ &\le(\rho+1)y+x-\rho L+\rho\cdot\frac{\rho}{\rho-2}\left(\frac{L}{2}-y\right)\\ &=-\frac{\rho+2}{\rho-2}y+x-\frac{\rho(\rho-4)}{\rho-2}\cdot\frac{L}{2}.\end{aligned}$$

The value $C=-\frac{\rho+2}{\rho-2}y+x$ is maximized at q. This is because for the function $y=\frac{\rho-2}{\rho+2}x-\frac{\rho-2}{\rho+2}C$, its slope $\frac{\rho-2}{\rho+2}$ is positive and less than the slope $\frac{\rho-1}{2}$ of y_3. Therefore, we have

$$\Delta_2 + \Delta_2' \leq -\frac{\rho+2}{\rho-2} \cdot \frac{\rho(\rho-1)}{\rho^2-\rho+2} \cdot \frac{L}{2} + \frac{\rho}{\rho^2-\rho+2} L - \frac{\rho(\rho-4)}{\rho-2} \cdot \frac{L}{2}$$

$$= -\frac{\rho(\rho-3)}{\rho-2} \cdot \frac{L}{2} < 0.$$

Thus, the proof of Theorem 12 is completed. ■

3.3 Tight Analysis

In Section 3, we prove that TriAct is ρ-competitive with $\rho \approx 3.326$. In this section, we show that the lower bound of the algorithm is also ρ. This mseans that the exact competitive ratio of the algorithm is ρ. We introduce an adversary through Theorem 13 and prove the existence of such a lower bound. The adversary makes a special sequence of requests on four nodes on a ring network against TriAct, such that upon any request either the condition of Case B or the condition of Case E holds.

Theorem 13. *For a sufficiently large integer n, there exists a request sequence $\sigma = r_1, \ldots, r_{4n}$ and four request nodes s, a, b, and c on a ring network, such that $cost_{TriAct}(s, \sigma) \geq \rho \cdot cost_{OPT}(s, \sigma)$ for $\rho \approx 3.326$ and the initial server node is s.*

Proof. The adversary sets $\delta(s, b) = \frac{\rho^2-3\rho+2}{2\rho^2-2\rho-8} L$ and $\delta(s, a) = \delta(b, c) = \frac{\rho-2}{\rho^2-\rho-4} L$, such that neither $\pi(s, a)$ nor $\pi(b, c)$ has an edge of $\pi(s, b)$. Moreover, request r_{4k+1} is at a, request r_{4k+2} is at b, request r_{4k+3} is at c, and request r_{4k+4} is at s, where $0 \leq k < n$ (see Figure 16).

We first calculate the total cost incurred by TriAct, $cost_{TriAct}(s, \sigma)$. The request r_{4k+1} is served with the cost of $\delta(s, a)$ and the server does not move from s, because Case B holds. The request r_{4k+2} is served with the cost of $\delta(s, b)$ and the server migrates from s to b with the cost of $\delta(s, b)$, because Case E holds. The request r_{4k+3} is served with the cost of $\delta(b, c)$ and the server does not move from b, because Case B

holds. The request r_{4k+4} is served with the cost of $\delta(b,s)$ and the server migrates from b to s with a cost of $\delta(b,s)$, because Case E holds. Therefore, we have

$$cost_{TriAct}(s,\sigma) = n\big(\delta(s,a) + 2\delta(s,b) + \delta(b,c) + 2\delta(b,s)\big)$$
$$= n\big(2\delta(s,a) + 4\delta(s,b)\big).$$

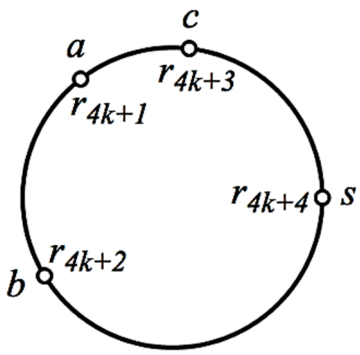


Figure 16. Representation of adversary model against TriAct

Now we calculate the total cost incurred by a specific algorithm ALG, $cost_{ALG}(s,\sigma)$ that is at least $cost_{OPT}(s,\sigma)$. ALG 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, we have

$$cost_X(s,\sigma) = \delta(s,a) + n\big(\delta(a,b) + \delta(a,c) + \delta(c,s) + \delta(c,a)\big)$$
$$= \delta(s,a) + n\big(\delta(b,c) + \delta(s,a)\big) = \delta(s,a) + n\big(2\delta(s,a)\big).$$

Since

$$\lim_{n\to\infty}\frac{n\big(2\delta(s,a)+4\delta(s,b)\big)}{\delta(s,a)+n\big(2\delta(s,a)\big)}=\frac{\delta(s,a)+2\delta(s,b)}{\delta(s,a)}$$

$$=\frac{\frac{\rho-2}{\rho^2-\rho-4}L+2\cdot\frac{\rho^2-3\rho+2}{2\rho^2-2\rho-8}L}{\frac{\rho-2}{\rho^2-\rho-4}L}=\rho$$

then $\frac{cost_{TriAct}(s,\sigma)}{cost_{ALG}(s,\sigma)}$ is at least ρ. This completes the proof. ■

3.4 Competitiveness

The exact value of ρ is the positive solution of the equation $-\rho^4+4\rho^3+\rho^2-18\rho+24=0$. The strict solution is

$$\rho=1-\frac{\sqrt{9\sqrt[3]{\lambda}^{\frac{3}{2}}+42\sqrt[3]{\lambda}-71}}{6\sqrt[6]{\lambda}}+\frac{\sqrt{-\sqrt[3]{\lambda}+\frac{48\sqrt[6]{\lambda}}{\sqrt{9\sqrt[3]{\lambda}^{\frac{3}{2}}+42\sqrt[3]{\lambda}-71}}+\frac{71}{9\sqrt[3]{\lambda}}+\frac{28}{3}}}{2}$$

where $\lambda=\frac{2\sqrt{13438}}{3}-\frac{1999}{27}$.

A more accurate approximation for this ratio would be $\rho\approx 3.325722333398888$.

3.5 Remarks

In this chapter, we proposed a deterministic algorithm for the uniform page migration problem in ring networks. The competitive ratio $\rho\approx 3.326$ of the algorithm is an improvement on the previous competitiveness of 3.414 in our setting for general ring networks. We think similar technique can be utilized to design novel algorithms for improving current solution on restricted networks such as 4-node and 5-node rings. Moreover, we conjecture the optimality of proposed algorithm for general ring networks. A tight example was found to express a lower bound of ρ for the algorithm.

If possible, one could seek to find an algorithm to cover a larger D with better competitiveness than 4.

CHAPTER 4

Telecommunicational Servers Problem

The traditional problems of managing data (see sections 1.5.2 and 01.6) arise for optimizing the 'load' of network. As a subsequence, the online algorithms are designed in a load-efficient approach, and no similar studies are known on time-efficient methods on networks. Recall the general problem of data management (section 1.5.2), in which the objective function of optimization $\sum_{i=1}^{m}(r_i(s_{i-1}) + \gamma(s_{i-1}, s_i))$ is defined for a sequence of m requests and a state transition function of γ. For example, the online algorithm working on the first request on the network of Figure 17, costs the 'summation' of 'serving' the request through slightly shaded edges as well as 'moving' a copy of data through the strongly shaded edge. 'Summation' of costs matters when the 'load' is the subject of optimization.

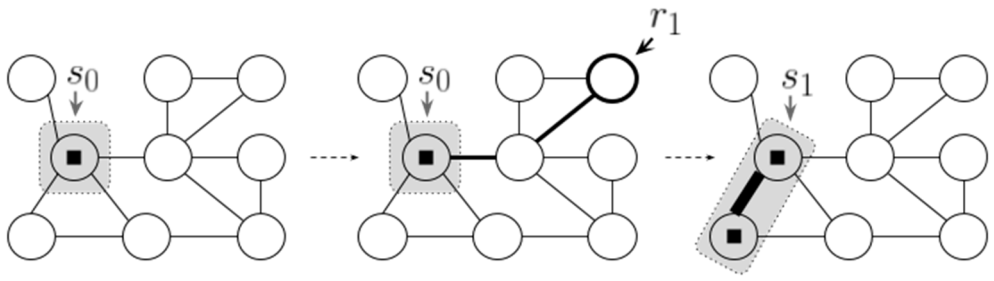


Figure 17. An example to show the importance of telecommunicational servers problem

How about if the 'time' matters for optimization? In the network of the example, both actions of service and move can be done simultaneously, so that we can take the maximum of them for optimization, since we do not see a reason for movement to wait and start after service, because there is no overlap between two actions.

In modern networks such as telecommunications, the bandwidth of interconnections has been dramatically developed, and it is not hard to imagine that in the case of overlapping, each interconnection is able to handle both of ‘service’ and ‘move’ tasks, simultaneously. We suppose each edge of the corresponding graph is capable to carry service and move without interference.

In the example of Figure 17, suppose the transition cost from the initial state s_0 to the next state s_1 equals to $\gamma(s_0, s_1) = 3$ incurred by moving a copy of data through the strongly shaded edge. In addition, the cost of serving the first request equals to $r_1(s_0) = 2$ which is incurred by the distance of lightly shaded edges. For our telecommunicational servers problem, the total cost for the first request equals to $\max\{3,2\} = 3$, since the cost is time and, we assume, service and move are done simultaneously. Note that in the data management problem (load-efficient version of telecommunicational servers problem), the total cost (i.e. load) equals to $3 + 2 = 5$.

Here we define a time-efficient problem, *telecommunicational servers problem*, with the objective of $\sum_{i=1}^{m} \max\{r_i(s_{i-1}), \gamma(s_{i-1}, s_i)\}$. Similarly, as a time-efficient version of page migration, we define *telecommunicational server problem*, with the objective function $\sum_{i=1}^{m} \max\{\delta(s_{i-1}, r_i), D \cdot \delta(s_{i-1}, s_i)\}$ for the distance metric of δ in the network with a single page of size D. In these problems, we suppose each request is issued after finishing the service and move (or migration) of the previous request. Taking the maximization in the objective functions is due to the waiting time until finishing the accomplishment of both actions.

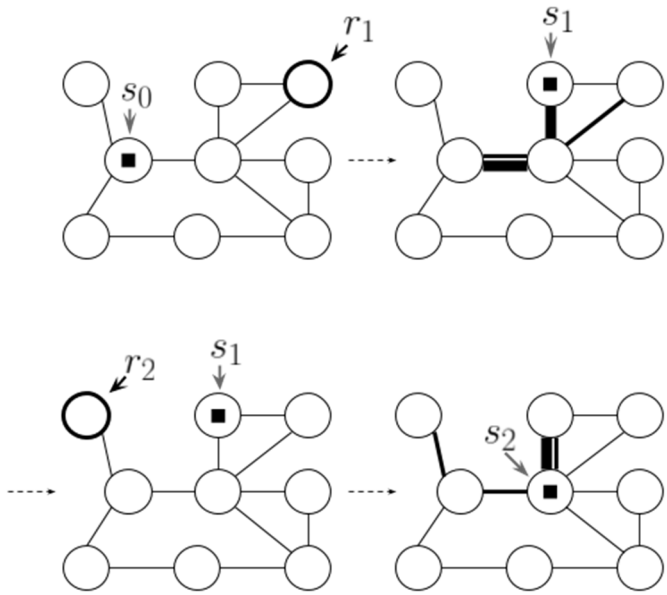


Figure 18. An example of an online algorithm for the telecommunicational server problem

Figure 18 shows an example of online algorithm to solve a telecommunicational server problem. This example can be recognized as a time-efficient version of former example shown in Figure 7 (see Section 0, Page 22) for the page migration problem. In this example here, suppose the page has size of $D = 2$. Therefore, the first request incurs a cost of 4, and the second request costs 3, with the assumption that all edges have unit distances.

To our point of view, the telecommunicational server problem is more interesting than the page migration problem, while the network is defined in Euclidean space. We follow the 'uniform' naming for the unit page size as *uniform telecommunicational server problem* which targets $\sum_{i=1}^{m} \max\{\delta(s_{i-1}, r_i), \delta(s_{i-1}, s_i)\}$. We leave open, the design of online algorithms for these problems in various metrics.

As a starting point, we mention some designing techniques of efficient online algorithms for each of these two problems on general graphs. About designing work function algorithms, the preliminary steps might be to recursively define the problem

and observe the behavior of work functions for some sample input sequences using dynamic programming. On another side, a counter-based approach could be considering counters on the requesting nodes, such that a new request will increment the counter of its corresponding node and will decrement the counter of another node. Some experiments may intuitively suggest deciding to move the servers after reaching a specific number of counts, which is usually a function of a data-property. One other idea may look for a phase-based algorithm by keeping the servers' location throughout a fixed or dynamic number of requests. As for continues spaces like Euclidean space, the evidences appear that deciding to move towards the center point of some of previous requesting points would be a proper choice for an online algorithm.

It would be an interesting future work, to consider designing online algorithms for these problems on general graphs by handling work functions (see section 1.3.7).

CHAPTER 5

Conclusions

We first introduce several frameworks as the basements to conduct the research works of online computation. Metrical task systems are clarified as a general framework for defining online problems, so that we adapt the specific problems of the study to these systems. The standard technique of work functions as well as the useful technique of counting are introduced as methodical approaches to design online algorithms. An abstraction of algorithmic competition, as well as the model of adversary, and the scheme of amortization, are all introduced as analytical tools for measuring the efficiency of online algorithms.

Well-studied network problems are surveyed so that the current states of the art on each problem are shown together with general hints of algorithmic design. The bounds of k-server problem for the maintenance of k servers regarding their mobile locations in the network are stated in general networks, and a long-standing open question for this problem is mentioned. The data management problem for administering replicas of the resource in a real-time responsible network is also introduced. The achievements on general and some restricted graphs are looked up for this problem, in both views of deterministic and randomized algorithms. One variant of this problem, the online Steiner tree problem is reviewed, expressing its strong connection with the origin in the aspect of competitiveness. The page migration problem has to determine an individual server location after satisfying the request of network. This problem is intently surveyed to demonstrate its lower and upper bounds by some algorithms in various topologies against different adversarial models. The

investigation is done more carefully for the uniform model of problem, in which each request occupies the entire resource.

For the uniform page migration problem, we succeed to achieve improvements in two different networks. The network of Euclidean space with arbitrary dimension, as well as the network of general rings, are both considered, and deterministic online algorithms are designed. The algorithms significantly improve the best former algorithms. For the Euclidean space, an efficient algorithm is designed. It is simple to understand. The online algorithm keeps the server at the center of two points, which are initially located at the initial server location. Upon each request, the farthest point moves to the requesting node. This algorithm improves the upper bound of the problem from 2.8 to 2.75, noting that the lower bound of problem is 2.5 and the analysis of adversary against the algorithm gives a lower bound of 2.732 for the algorithm. As for the general rings, a deterministic algorithm is designed. It is hard to explain in some easy words. This algorithm works in six cases and in analysed in eleven cases, achieving an improved and tightened upper bound of 3.326 for the problem, and extending the scheme of amortization for online computation.

The k-server problem can be adopted for both load-efficient and time-efficient subjects of study. But the problem of data management does not make sense from a time-efficient point of view, since there is capacity for performing both actions of service and relocation simultaneously, when they are done through separate interconnections, or when the bandwidth is capable enough for carrying the action of service and movement, at the same time. In this way, we define new online problems of Telecommunicational servers and Telecommunicational server, as the variants of data management and page migration.

REFERENCES

[1] A. Khorramian and A. Matsubayashi, “Uniform Page Migration Problem in Euclidean Space,” *Algorithms*, vol. 9, no. 3, p. 57, 2016.

[2] A. Khorramian and A. Matsubayashi, “Online Page Migration on Ring Networks in Uniform Model,” Dec. 2016.

[3] D. Goldin, P. Wegner, and S. A. Smolka, *Interactive computation: The new paradigm*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.

[4] D. Goldin and P. Wegner, “Principles of Interactive Computation,” in *Interactive Computation*, Springer Berlin Heidelberg, 2006, pp. 25–37.

[5] R. Milner, “Turing, Computing and Communication,” in *Interactive Computation*, Springer Berlin Heidelberg, 2006, pp. 1–8.

[6] G. Japaridze, “Computability Logic: A Formal Theory of Interaction,” in *Interactive Computation*, Springer Berlin Heidelberg, 2006, pp. 183–223.

[7] 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.

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT Press, 2009.

[9] M. M. Fréchet, “Sur quelques points du calcul fonctionnel,” *Rend. del Circ. Mat. di Palermo*, vol. 22, no. 1, pp. 1–72, Dec. 1906.

[10] S. Albers, “Online Algorithms,” in *Interactive Computation*, Springer Berlin Heidelberg, 2006, pp. 143–164.

[11] N. Nishimura, “Introduction to Reconfiguration,” 2017.

[12] B. E. Illy, “The complexity of change,” in *Surveys in combinatorics*, vol. 409, no. June, 2002, pp. 1–21.

[13] T. Ito *et al.*, “On the complexity of reconfiguration problems,” *Theor. Comput. Sci.*, vol. 412, no. 12–14, pp. 1054–1065, Mar. 2011.

[14] J. Abernethy, P. L. Bartlett, N. Buchbinder, and I. Stanton, “A Regularization Approach to Metrical Task Systems,” Springer, Berlin, Heidelberg, 2010, pp. 270–284.

[15] N. Buchbinder, S. Chen, and J. S. Naor, “Competitive analysis via regularization,” in *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, 2014, pp. 436–444.

[16] 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.

[17] S. Ben-David *et al.*, “On the power of randomization in on-line algorithms,” *Algorithmica*, vol. 11, no. 1, pp. 2–14, Jan. 1994.

[18] A. V Aho and J. E. Hopcroft, *The design and analysis of computer algorithms*. Pearson Education India, 1974.

[19] R. E. Tarjan, “Amortized Computational Complexity,” *SIAM J. Algebr. Discret. Methods*, vol. 6, no. 2, pp. 306–318, Apr. 1985.

[20] A. Borodin, N. Linial, and M. E. Saks, “An optimal on-line algorithm for metrical task system,” *J. ACM*, vol. 39, no. 4, pp. 745–763, Oct. 1992.

[21] A. S. Tanenbaum and D. (David) Wetherall, *Computer networks*. Pearson Prentice Hall, 2011.
[22] B. Bicsi, *Network Design Basics for Cabling Professionals*. McGraw-Hill, 2002.
[23] E. Torng, "A unified analysis of paging and caching," *Algorithmica*, vol. 20, no. 2, pp. 175–200, 1998.
[24] 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.
[25] 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.
[26] 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.
[27] E. Koutsoupias and C. H. Papadimitriou, "On the k-server conjecture," *J. ACM*, vol. 42, no. 5, pp. 971–983, 1995.
[28] 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.
[29] 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.
[30] Y. Bartal, A. Fiat, and Y. Rabani, "Competitive algorithms for distributed data management," *J. Comput. Syst. Sci.*, pp. 341–358, 1995.
[31] B. Awerbuch, Y. Bartal, and A. Fiat, "Competitive distributed file allocation," *Inf. Comput.*, vol. 185, no. 1, pp. 1–40, 2003.
[32] M. Imase and B. M. Waxman, "Dynamic Steiner tree problem," *SIAM J. Discret. Math.*, vol. 4, no. 3, pp. 369–384, 1991.
[33] 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.
[34] 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.
[35] D. L. Black and D. D. Sleator, "Competitive Algorithms for Replication and Migration Problems," Pittsburgh, PA, USA, 1989.
[36] R. M. Karp, "A 2k-competitive algorithm for the circle," *Manuscript, August*, vol. 5, 1989.
[37] A. Gupta, I. Newman, Y. Rabinovich, and A. Sinclair, "Cuts, trees and ℓ_1-embeddings of graphs," *Combinatorica*, vol. 24, no. 2, pp. 233–269, 2004.
[38] A. Matsubayashi, "Non-greedy Online Steiner Trees on Outerplanar Graphs," Springer, Cham, 2017, pp. 129–141.
[39] 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.
[40] W. Glazek, "Online algorithms for page replication in rings," *Theor. Comput. Sci.*, vol. 268, no. 1, pp. 107–117, 2001.

[41] A. Matsubayashi, “A 3-Competitive Page Migration Algorithm on Trees,” 2015.
[42] 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.
[43] 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.
[44] J. Westbrook, “Randomized algorithms for multiprocessor page migration,” *On-line Algorithms*, vol. 7, no. 5, pp. 135–150, 1992.
[45] 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.
[46] A. Matsubayashi, “Asymptotically Optimal Online Page Migration on Three Points,” *Algorithmica*, vol. 71, no. 4, pp. 1035–1064, 2015.
[47] A. Matsubayashi, “UNIFORM PAGE MIGRATION ON GENERAL NETWORKS,” *Int. J. Pure Appl. Math.*, vol. 42, no. 2, pp. 161–168, 2008.
[48] 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.
[49] J. Kleinberg and E. Tardos, *Algorithm Design*. Pearson Education, 2011.
[50] J. A. (John A. Bondy and U. S. R. Murty, *Graph theory with applications*. American Elsevier Pub. Co, 1976.
[51] D. B. West, *Introduction to graph theory*. Prentice Hall, 2001.
[52] N. Reingold, J. Westbrook, and D. D. Sleator, “Randomized competitive algorithms for the list update problem,” *Algorithmica*, vol. 11, no. 1, pp. 15–32, 1994.
[53] A. Matsubayashi, “A 3+Omega(1) Lower Bound for Page Migration,” *2015 Third Int. Symp. Comput. Netw.*, no. 1, pp. 314–320, 2015.