Server provisioning for distributed interactive applications

Zheng, Han Ying

2014

Zheng, H. Y. (2014). Server provisioning for distributed interactive applications. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/61880>

<https://doi.org/10.32657/10356/61880>

NANYANG
TECHNOLOGICAL
UNIVERSITY

**SERVER PROVISIONING FOR
DISTRIBUTED INTERACTIVE
APPLICATIONS**

ZHENG HANYING
SCHOOL OF COMPUTER ENGINEERING
2014

SERVER PROVISIONING FOR DISTRIBUTED INTERACTIVE APPLICATIONS

ZHENG HANYING

Supervisor: Dr. Tang Xueyan

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in fulfilment of the requirement for the degree of

Doctor of Philosophy

2014

Dedicated to the four most important persons in my life.

Had I not seen the Sun, I could have borne the shade.

Acknowledgement

I would like to offer my sincerest gratitude to my supervisor, Dr. Tang Xueyan, for his guidance. This thesis would not have existed without his help and support. Thanks also to all my friends and colleagues in PDCC laboratory for their friendship. Finally, I would like to express my deepest gratitude to my family for their unfailing love, support and encouragement.

Contents

Acknowledgement	ii
Contents	iii
List of Figures	vi
List of Tables	ix
Abstract	x
1 Introduction	1
1.1 Server Provisioning for DIAs	2
1.2 Thesis Contribution	4
1.3 Thesis Organization	5
2 Literature Review	6
2.1 Architectures	6
2.2 Server Provisioning	11
2.3 Client Assignment	14
2.4 Summary	15
3 Server Provisioning Problem for Discrete DIAs	16
3.1 Problem Formulation	16
3.2 Hardness of the Problem	19
3.2.1 Hardness for Networks without Triangle Inequality	19
3.2.2 Hardness for Restricted Choices of Server Locations	24
3.2.3 Hardness for Limited Number of Server Locations to Select	26

3.3	k -Median Server Placement	30
3.4	NEAREST Server Placement	35
3.5	T-GREEDY Server Placement	37
3.6	T-BETTER Server Placement	43
3.7	Analysis of Server Placement in Networks with α -Triangle Inequality .	44
3.7.1	k -Median Server Placement	45
3.7.2	NEAREST Server Placement	53
3.7.3	T-GREEDY Server Placement	55
3.7.4	T-BETTER Server Placement	59
3.8	Experimental Evaluation	65
3.8.1	Performance Comparison for Different Algorithms	67
3.8.2	Impact of Server Capacity	73
3.8.3	Impact of Dynamics in Network Conditions	75
3.9	Summary	78
4	Server Provisioning Problem for Continuous DIAs	79
4.1	Problem Formulation	79
4.2	Hardness of the Problem	82
4.2.1	Hardness for Networks without Triangle Inequality	83
4.2.2	Hardness for Restricted Choices of Server Locations	87
4.2.3	Hardness for Limited Number of Server Locations to Select . . .	91
4.3	k -Center Server Placement	94
4.4	NEAREST Server Placement	97
4.5	M-GREEDY Server Placement	99
4.6	M-BETTER Server Placement	106
4.7	Analysis of Server Placement in Networks with α -Triangle Inequality .	111
4.7.1	k -Center Server Placement	111

4.7.2	NEAREST Server Placement	115
4.7.3	M-GREEDY Server Placement	118
4.7.4	M-BETTER Server Placement	121
4.8	Experimental Evaluation	124
4.8.1	Performance Comparison for Different Algorithms	126
4.8.2	Impact of Server Capacity	132
4.8.3	Impact of Dynamics in Network Conditions	134
4.9	Summary	136
5	Conclusion and Future Work	138
5.1	Conclusion	138
5.2	Future Work	140
5.2.1	Multicast Server Provisioning Problem	140
5.2.2	Client Assignment Problem for Long Tail Interactions in DIAs .	142
	Bibliography	144

List of Figures

2.1	Single server model	6
2.2	Peer-to-peer model	7
2.3	Multi-server model	9
3.1	Example instances of the set cover problem and the server provisioning problem for discrete DIAs	20
3.2	Example instances of the 3DM problem and the server provisioning problem for discrete DIAs	26
3.3	The approximation ratio 3 of the k -median placement is tight.	34
3.4	The approximation ratio 2 of the T-GREEDY algorithm is tight.	41
3.5	The approximation ratio of the k -median placement is tight: tight ex- ample for $\frac{1}{2}\alpha^3 + \frac{3}{2}\alpha^2 + 1$	50
3.6	The approximation ratio of the k -median placement is tight: tight ex- ample for $\frac{2}{3}\alpha^3 + \alpha^2 + \frac{2}{3}$	52
3.7	The approximation ratio 2α of the T-GREEDY algorithm is tight.	57
3.8	Approximation ratios of different server placement algorithms for dis- crete DIAs under the α -triangle inequality	64
3.9	Cumulative distribution of the extent of triangle inequality violations for all triangles in the Meridian dataset	66
3.10	Normalized interactivity of different algorithms when all nodes are can- didate server locations	67

3.11	Normalized interactivity of different algorithms for different numbers of candidate server locations	69
3.12	Cumulative distribution of normalized interactivity for 1000 simulation runs	70
3.13	Performance of different algorithms normalized by the same lower bound value for each client set	71
3.14	Normalized interactivity of different algorithms for different individual server capacities	73
3.15	Normalized interactivity of different algorithms for different total server capacities	74
3.16	Cumulative distribution of the standard deviations of network latencies for all node pairs in the PlanetLab All Pairs Pings dataset	75
3.17	Cumulative distribution of normalized interactivity by different algo- rithms under dynamic network latency	76
4.1	An example of minimum achievable interaction time.	81
4.2	Example instances of the set cover problem and the server provisioning problem for continuous DIAs	84
4.3	Some triangles in the constructed network for server provisioning in- stance $\mathcal{X}$	90
4.4	Example instances of the X3C problem and the server provisioning problem for continuous DIAs	91
4.5	The approximation ratio 3 of the k -center placement is tight.	97
4.6	The approximation ratio 2 of the M-GREEDY algorithm is tight.	105
4.7	The approximation ratio $\frac{5}{3}$ of the M-BETTER heuristic is tight.	110
4.8	The approximation ratio 2α of the M-GREEDY algorithm is tight.	120

4.9	Approximation ratios of different server placement algorithms for continuous DIAs under the α -triangle inequality	125
4.10	Normalized interactivity of different algorithms when there are 300 candidate server locations	127
4.11	Normalized interactivity of different algorithms for different numbers of candidate server locations.	128
4.12	Cumulative distribution of normalized interactivity for 1000 simulation runs	130
4.13	Normalized interactivity of different algorithms using the same lower bound value for each client set	131
4.14	Normalized interactivity of different algorithms for different individual server capacities.	132
4.15	Normalized interactivity of different algorithms for different total server capacities.	133
4.16	Cumulative distribution of normalized interactivity by different algorithms under dynamic network latency	135

List of Tables

3.1	Summary of different server placement algorithms for discrete DIAs . . .	30
3.2	Network latency matrix for tight example of $\frac{1}{2}\alpha^3 + \frac{3}{2}\alpha^2 + 1$	51
3.3	Network latency matrix for tight example of $\frac{2}{3}\alpha^3 + \alpha^2 + \frac{2}{3}$	53
3.4	Average running times of different algorithms for discrete DIAs (in seconds)	72
4.1	Summary of different server placement algorithms for continuous DIAs	94
4.2	Average running times of different algorithms for continuous DIAs (in seconds)	131

Abstract

Recent years have witnessed the enormous popularity of distributed interactive applications (DIAs), which allow participants that are distributed in the network to interact with each other simultaneously. The rapid growth of DIAs has raised stringent requirements on providing realistic sense of interaction between participants. To deliver high quality service and guarantee good experiences for participants, it is necessary for the DIA operators to deploy the servers hosting DIAs in a distributed manner. In such model, the information exchanged between clients is transmitted through servers. Servers communicate with each other to maintain consistent and cohesive application states. Since DIAs are latency sensitive, message transmission between clients should be performed in a timely manner. Otherwise, participants may experience inconsistent states and lags in response. However, due to the presence of inevitable network latencies between clients and servers as well as among servers, it is a critical challenge to guarantee the quality of interactions.

To tackle the impact of network latencies on consistency, techniques like local-lag and dead reckoning have been animatedly explored. Unfortunately, these approaches could not actually reduce network latencies, and they may introduce even more delays or additional computational overheads to the network. For example, the local-lag approach maintains consistency by prolonging the waiting time of operation execution on the servers. In this way, the interactivity performance could not be improved due to the decrease in responsiveness.

Although it is unrealistic to completely eliminate the network latency involved in

the interaction, it is possible to reduce it by a smart combination of server locations. This is because the locations of servers influence not only the inter-server latencies, but also the latencies from clients to servers. Thus, the placement of servers is an important factor to the overall performance of distributed interactive applications.

In this thesis, we study the problem of how to place servers appropriately on the network to host DIAs, with the goal of improving the interactivity performance of DIAs. We consider two different types of DIAs, i.e., discrete DIAs and continuous DIAs, which are differentiated by their ways of updating the application states. We formulate the server provisioning problems for both discrete DIAs and continuous DIAs by taking their specific Quality of Service requirements into account. We show that the two server provisioning problems are challenging by presenting their NP-hardness and non-approximability results under several conditions which may be prevalent in practice: (i) the network latencies do not satisfy the triangle inequality; or (ii) the locations where servers can be placed in the network are restricted; or (iii) the number of server locations to select is limited. For each problem, we present four server placement algorithms and theoretically analyze their approximation ratios in networks satisfying the triangle inequality as well as networks violating the triangle inequality. To the best of our knowledge, this is the first work that analyzes the asymptotic approximability of server placement algorithms with respect to the extent of triangle inequality violations. The performance of the algorithms is also studied by experimental evaluations using real Internet latency data. The results show that our proposed algorithms produce near-optimal server placements.

Chapter 1

Introduction

An increasing number of Distributed Interactive Applications (DIAs) are emerging in recent years to provide people with new ways of collaboration and entertainment. In these applications, participants dispersed at different locations interact with each other through the network in real time. Examples of DIAs include online gaming [1], instant messaging [2], collaborative computer-aided design and engineering [3], and distributed e-learning [4].

DIAs usually operate with client-server architectures [5], in which the *servers* maintain the application states (such as the virtual game worlds in online gaming and the shared workspaces in collaborative design tools), and execute the operations submitted by the participants who are known as *clients*. Owing to different ways of maintaining the application states, DIAs can be classified into two categories: discrete DIAs and continuous DIAs [6]. In discrete DIAs, the application states are only updated due to the operations performed by clients. Representative discrete DIAs include instant messaging and collaborative computer-aided design. In continuous DIAs, the application states not only change because of executing the client-initiated operations, but also evolve along with the elapse of time. A typical example of continuous DIAs is networked gaming, where the states of virtual game worlds are normally updated at a fast pace even when there is no operation input from the players.

1.1 Server Provisioning for DIAs

A critical Quality of Service measure for DIAs is the time lag experienced by the participants during their interaction. To host a DIA, a group of interconnected servers are often deployed in the network, each of which maintains a copy of the state of the DIA. Each client connects to one of the servers, and interacts with other clients through its connecting server. When issuing an operation, a client sends the operation to its connecting server, which then relays it to the rest servers. On receiving the operation, the servers execute the operation to update their application states accordingly and disseminate the state update to the clients to reflect the change. Thus, the time lag in the interaction includes the network latency in the communication, and the processing and queuing delays at the servers. Since DIAs are latency sensitive, the above interaction process must be performed timely in order to achieve high quality of experience in DIAs. For example, in the context of networked gaming, it has been shown that delays around 100ms are already noticeable and it may become annoying when delays increase to above 200ms [7].

A major barrier to enhancing the quality of experience in DIAs is the network latency involved in the interaction. Wide geographical spreads of clients in large-scale DIAs necessitate distributed deployment of servers to support the interactions among clients [5]. Servers can be placed not only at locations in the center of the network (e.g. cloud data centers), but also near the edges of the network such as nano data centers [8] and smart edge nodes [9]. The increasing elasticity of computing resources at these locations allows the DIA operators to quickly scale up or down the capacity on demand for hosting their applications. This makes the server-side processing and queuing delays much easier to cut than the network latency. On the other hand, even with distributed server infrastructures, the network latency cannot be completely eliminated from the interactions between clients in DIAs. The net-

work latency involved in client interactions is directly affected by the locations where servers are placed. Excessive network latency can severely degrade the participant's quality of experience in DIAs. Thus, server placement is of crucial importance to the interactivity performance of DIAs.

Besides interactivity, network latency also challenges the consistency and fairness of DIAs. Consistency means to create shared common views of the application state among all clients to enable meaningful interactions [10]. Significant divergence of the application state can seriously affect the behavior and decisions of the participants. Fairness is to ensure that all clients have equal chances to participate despite their geographically dispersed locations in the network [11, 12]. This is particularly important for those DIAs in which the participants compete with each other intensely. For example, in many online games, players would not like to join competitions in which others have unfair advantages against them. Due to constant state updates, fulfilling the consistency and fairness requirements is much tougher in continuous DIAs than in discrete DIAs.

In this thesis, we focus on reducing the network latency involved in the interaction between participants by considering where to place servers in the network. We investigate the server placements for both discrete DIAs and continuous DIAs. With distributed server infrastructures, the interaction between clients involves both the network latencies between clients and servers and the network latencies among the servers. Thus, the locations where the servers are placed must be carefully chosen in order to optimize the interactivity of DIAs. We show by both theoretical analysis and experimental evaluation that aggressively reducing the client-to-server latency alone is not effective for improving the interactivity of DIAs. So, the k -center server placement that aims to shorten the maximum latency from clients to servers, and the k -median server placement that aims to shorten the total latency from clients to servers often make the interactivity between clients far worse than optimum. A

decent server placement must consider the trade-off between the latencies from clients to servers and the latencies among servers.

1.2 Thesis Contribution

The major contributions of this thesis are summarized as follows.

- We formulate the server provisioning problems for discrete DIAs and continuous DIAs as combinatorial optimization problems. We show that both problems are challenging by analyzing their hardness under the following conditions which may be prevalent in practice: (i) the network latencies do not satisfy the triangle inequality; or (ii) the locations where servers can be placed in the network are restricted; or (iii) the number of server locations to select is limited. A number of non-approximability results are further derived for both problems.
- We present four different server placement algorithms for each server provisioning problem. These include a nearest server placement heuristic, efficient greedy placement algorithms, and algorithms that combine these server placements on a selective basis. We conduct theoretical analysis on the approximation ratios of the four server placements for networks with triangle inequality and the asymptotic approximability of these server placements with respect to the extent of triangle inequality violations in the networks. We also demonstrate the tightness of the approximability results where possible.
- We experimentally evaluate the performance of the four server placement algorithms for each server provisioning problem using real Internet latency data. The results show that our proposed algorithms significantly outperform other baseline server placements. We also study the impact of the changes in network latency on the performance of the server placement algorithms. The results

show that our proposed algorithms are more resilient to the dynamics in network latency.

1.3 Thesis Organization

This thesis comprises five chapters. The rest of the thesis is organized as follows. Chapter 2 presents a review of the related works. Chapter 3 formulates and investigates the server provisioning problem for discrete DIAs. Chapter 4 formulates and studies the server provisioning problem for continuous DIAs. Finally, we conclude in Chapter 5 by summarizing the research contributions along with a discussion on the directions for future work.

Chapter 2

Literature Review

This chapter presents a survey of server provisioning and related issues in DIAs. In Section 2.1, we provide an introduction to the architectures that support DIAs. In Section 2.2, we review previous works on the server placement problem. After that, Section 2.3 gives a quick overview of the client assignment problem for DIAs.

2.1 Architectures

Figure 2.1: Single server model

There are three types of architecture models to support DIAs: the single server model, the peer-to-peer model, and the multi-server model. As shown in Figure 2.1, in the single server model [13], a central server executes all the operations generated by the clients and distributes the resultant state updates to the clients. Thus, there is no consistency issue in the single server model. However, the risk of the single-point failure and the challenge of the performance bottleneck of the central server make it

unattractive for large scale distributed interactive applications.

Figure 2.2: Peer-to-peer model

The peer-to-peer model [14–16] is a decentralized model to address the above limitations. In this model (see Figure 2.2), there are no dedicated servers in the network, and all the computations are carried out by the clients. The resultant state updates are transmitted among the clients directly. It avoids the central bottleneck and improves the reliability of the system. However, maintaining a huge number of direct network connections at every client is likely to exceed the available bandwidths at the clients and deteriorate the communication performance.

Several works [17–19] have attempted to resolve this issue by the technique called interest management. Interest management is based on the observation that each player in a Massively Multiplayer Online Game (MMOG) only needs to be aware of a small region of the virtual game world surrounding the player called Area Of Interest (AOI). Therefore, it is sufficient to have each client to receive only the state update messages of the avatars within its AOI and send its own state update messages to the clients whose AOIs cover it. In this way, the volume of messages transmitted in the network could be significantly reduced.

Network latency has direct impact on the performance of interest management.

Since the AOI changes along with the movement of the player, the avatars located near the boundary of the AOI may not be discovered by the local player in time due to the delay of message transmission [15, 20, 21], thereby causing inconsistency. An overlay network called VON addresses this problem by using the Voronoi diagrams [22] to define the spatial relationship among avatars in the virtual game world such that a local avatar could find its nearest neighbours efficiently [21]. Avatars located near the boundary of a local avatar's AOI constantly notify the local avatar about the state information of their neighbours that may enter the AOI. In this way, the local avatar could discover the new avatars around it timely as it moves around. The COVER method [20] provides neighbour discovery by means of supernodes which are responsible for notifying clients about the state information of the potential missing neighbours in their AOIs.

Compared to the traditional interest management techniques, the peer-to-peer MMOG system called Donnybrook [23] further limits the number of players to whom a local player pays attention, and defines the set of such players as the interest set of a local player. In Donnybrook, each client executes the operations it generates, and sends the resultant status update messages to a small set of players that pay attention to its avatar at high frequency. The rest players receive only low fidelity status information at much lower frequency. The Donnybrook system also allows the clients with low average latencies to others to forward messages for those with insufficient network capacities. By these means, Donnybrook could significantly reduce the volume of state update messages and alleviate network congestion, and therefore support fast-paced games that are sensitive to network latency.

Several works [19, 24, 25] have proposed to construct overlay networks for online games based on Distributed Hash Table (DHT). In such systems, each client maintains only a small portion of direct connections to other clients, thereby alleviating the high bandwidth consumption problem of the peer-to-peer model. However, since the

requests for receiving the state updates must be relayed by multiple intermediate nodes, these systems introduce multi-hop latency between clients and significantly deteriorate the responsiveness of the DIAs. As a result, they are not suitable for hosting DIAs with stringent latency requirement.

Figure 2.3: Multi-server model

The multi-server model [26–30] is a mixture of the single server model and the peer-to-peer model. In the multi-server model (see Figure 2.3), multiple servers are deployed and interconnected in a peer-to-peer manner via well provisioned network links. Each server maintains a portion or a full copy of the application state, and is responsible for executing the operations initiated by the clients. Message exchange between servers is indispensable to keep track of the state changes in different servers so as to maintain a consistent and cohesive application state. However, due to the variance of network latencies, the copies of an event message may arrive at different servers at different times and the consistent orders of operation executions may not be guaranteed. Therefore, consistency maintenance among servers becomes a critical issue in the multi-server model.

A significant body of work has investigated how to solve this problem [6, 27, 31, 32]. The local-lag technique proposed by Mauve et al. [6, 33] deliberately delays the execution of an operation sent from a local server for a certain period of time so that

there is enough time for the operation to be transmitted to the other servers. After that, all the servers execute the operation at the same time. Therefore, it ensures that all the operations are executed in the same order at all servers. If the amount of local lag is insufficient and an operation arrives late at a server, then a mechanism called timewarp [6] is used. The timewarp mechanism rolls back the application state at the server to the one immediately preceding the missing operation, and re-executes all the subsequent operations after executing the missing operation. To maintain the consistency among different servers, the local-lag technique introduces additional synchronization delays and therefore decreases the responsiveness. Besides, if there are bursts of delayed events, timewarp has to perform multiple rollbacks in rapid succession, increasing the overhead of the mechanism. The trailing state synchronization (TSS) [27] concurrently maintains several copies of the application state running with different synchronization delays in each server. The state with the shortest synchronization delay (known as the leading state) is rendered on the user's screen while the other states called trailing states are used to detect inconsistency and support rollbacks. In each server, if one of the trailing states executes an operation that is missed by its preceding state, then all the states that have shorter synchronization delays perform rollbacks in a cascade manner and eventually correct the leading state. In this way, TSS is able to re-execute more than one delayed operations in a single rollback.

Assiotis et al. [34] develop a locking mechanism to tackle inconsistency caused by the delay between servers. In this mechanism, each server manages a disjoint portion of the virtual world called region, and executes the operations generated from players located in that region. If a server executes an operation that may affect other regions, then it should acquire authorization from the involved servers and lock those regions first. Servers without authorization should queue if the regions they request are locked by others. Such mechanism may increase the interaction delay significantly because

the locking operation involves several message exchanges between servers.

In the peer-to-peer model, the clients usually have equal roles and the same responsibilities. Several works [35, 36] have investigated a variation of the peer-to-peer model called the super-peer model, which promotes a portion of clients as super peers for operation execution. In this way, the super peers act like the dedicated servers. The difference of this model to the multi-server model is that the super peers in the super-peer model also operate as clients themselves, while in the multi-server model, the servers are separate from the clients.

Without loss of generality, in this thesis, we model the server as the computational unit that takes up the role of operation execution, and the candidate server locations are the locations where the servers can be allocated to. For example, in the super-peer model, if any client has the potential to be promoted, the candidate server locations are the same as the set of clients. By doing this, our server provisioning problems can be adapted to both the super-peer model and the multi-server model.

2.2 Server Provisioning

All the aforementioned approaches for the multi-server model tackle the impact of network delay with the assumption that servers are already placed in the network. Since the locations of the servers influence both the inter-server latencies and the latencies between clients and servers, we can further improve the performance of distributed interactive applications by carefully selecting the locations of the servers. This is referred to as the server provisioning problem or the server placement problem [37–41]. Placing servers without considering the effect of server locations to the network latency involved in the interaction may not be beneficial to optimizing the performance of DIAs. Moreover, it is challenging to find a decent server placement for DIAs because the servers hosting the DIAs can be globally distributed to locations not only in the center of the network, but also near the edges of the network such

as nano data centers [8] and smart edge nodes [9]. Zoom-In-Zoom-Out (ZIZO) is a distributed algorithm presented by Lee et al. [40] for adaptive server selection in the MMOG. Given a set of pre-deployed servers and a set of clients performing interactions, the purpose of ZIZO is to find a minimum subset of servers for the clients such that the maximum synchronization delay satisfies a given real-time delay requirement. The major operations in the ZIZO algorithm are the Zoom-In operation and the Zoom-Out operation. The former operation reduces the number of servers in use by migrating them towards the core server which has the minimum value of the maximum distance to all the clients. After that, based on the observation that the servers close to the core server may introduce more synchronization delays, the Zoom-Out operation intends to remove some servers obtained in the previous procedure by assigning clients to the servers far away from the core server. Since it does not attempt to minimize the synchronization delay, ZIZO does not provide the optimal quality of interaction between different clients.

Laoutaris et al. [42] propose a distributed algorithm to place service facilities so as to minimize the distances between clients and their closest facilities in large scale networks. In their algorithm, each node does not require global knowledge of the network topology, but demands only the information about the neighbourhood that is within certain hops from it. The algorithm starts with a batch of facilities that are randomly placed, and then in each iteration, it randomly pick a facility f . Then, f together with those facilities who have neighbouring nodes that are also the neighbour of f run a local re-optimization algorithm to calculate new facility locations, and mark themselves as processed. After all the original facilities in the batch are processed, the new batch of facilities obtained will repeat this process until no beneficial relocation could be found. By reducing the amount of information required for each node, such distributed algorithm overcomes the scalability limitation compared to a centralized solution.

Briceno et al. [43] describe five heuristics to solve the following server selection problem: In the single server model, some clients could be promoted as secondary servers to shift and offload the computation from the central server if the volume of computation exceeds a certain threshold. Their problem aims to select a set of clients to be secondary servers such that the maximum response time of all operations, i.e., the maximum time difference between the time when the operations are sent from the clients to the servers and the time when the response messages are returned to the senders, is minimized.

The classical k -center and k -median placements have been strongly advocated for server provisioning in the Internet [40, 42, 44–46]. These two placements aim to place k servers in the network to minimize the maximum network latency and the total network latency from the clients to their nearest servers respectively. They well suit the need of web content delivery whose performance is primarily determined by how fast the contents stored on the servers are delivered to the clients. They are also considered to be closely related to the server placement in DIAs. Both the k -center and k -median problems are NP-hard [47]. There have been numerous studies investigating these two problems [42, 48, 49].

The k -median and k -center server placements are quite successful for web content delivery [45, 46]. However, DIAs are fundamentally different from web content delivery. The clients in the web just download contents from web servers. Thus, their access performance can be optimized by simply minimizing the client-to-server latency. In contrast, the clients in DIAs are engaged in mutual interactions among themselves. Each client connects to one server through which it interacts with all the other clients. Therefore, the interaction time between clients must include not only the network latencies from the clients to their connected servers but also the latencies between their connected servers. From the computability perspective, the mutual interaction feature also makes server provisioning for DIAs much more challenging than

that for web content delivery. With the client-to-server latency as the sole optimization objective, placing more servers in the network can only improve the web access performance. Nevertheless, deploying more servers in DIAs does not necessarily imply shorter network latency in the interaction between clients. Moreover, neither the k -center placement nor the k -median placement gives any consideration to maintaining consistency and fairness for continuous DIAs. It has been shown to be necessary to account for the combined effect of client-to-server latency and inter-server latency in order to fulfill the consistency and fairness requirements [40, 50].

2.3 Client Assignment

If the servers are already placed in the network, the problem of deciding how to assign clients to servers is referred to as the client assignment problem [50, 51]. This is an orthogonal issue to the server provisioning problems studied in this thesis. Zhang and Tang [50, 51] define the relationship between the assignment of clients and the interactivity performance of DIA, and formally formulate the client assignment problem as a combinational optimization problem. They prove the NP-completeness of the problem, and present a centralized greedy algorithm and a distributed greedy algorithm, both of which have been shown to generate solutions that are close to the optimal assignment in their experiments. However, in many prevalent DIAs, the clients naturally connect to their preferred nearest servers in terms of network latency [27, 30, 40, 52]. In Section 3.8.3 of Chapter 3 and Section 4.8.3 of Chapter 4, we shall study the impact of assigning clients to their non-nearest servers on server provisioning for DIAs under dynamic network conditions. We show that the improvement of interactivity achieved by carefully choosing the server locations is much more significant than adjusting the connections from clients to servers according to the change of network latency.

A variant of the client assignment problem is the partitioning problem in distributed virtual environments [53], whose purpose is to dynamically partition the

virtual game world into disjoint regions and assign these regions to different servers so as to balance the computation workload and reduce the inter-server communication cost. Since the avatars are assigned to the servers managing the regions where they locate, the partition problem in distributed virtual environments is a variant of the client assignment problem in essence. In [53], Lui et al. prove that the partitioning problem is NP-complete by a reduction from the subset sum problem, and develop a centralized algorithm and a parallel algorithm to address this issue. They also illustrate that, by executing the algorithms periodically, their algorithms could be used in dynamic scenarios where avatars can move, join or leave.

2.4 Summary

Most of the existing server placement algorithms have focused on web content delivery. They do not consider the real-time mutual interactions among the clients of DIAs as a performance measurement. They focus on minimizing a latency metric between clients and servers, bypassing the non-neglectable inter-server latencies involved in the interaction among clients. On the other hand, the works that study the synchronization delay in the interaction do not explicitly optimize the interactivity performance of DIAs. Moreover, none of the aforementioned studies take into consideration the consistency and fairness requirements for continuous DIAs. In this thesis, we aim to optimize the interactivity performance of DIAs by judiciously finding the server locations, while considering the consistency and fairness constraints.

Chapter 3

Server Provisioning Problem for Discrete DIAs

In this chapter, we study server provisioning for discrete DIAs. Section 3.1 formulates the server provisioning problem for discrete DIAs. Section 3.2 investigates the hardness of the problem. Sections 3.3 to 3.7 present four server provisioning algorithms and analyze their approximation ratios. Experimental evaluations are elaborated in Section 3.8. Finally, Section 3.9 summarizes the chapter. Part of the work in this chapter has been published in [54].

3.1 Problem Formulation

We model the network infrastructure underlying the DIA as a graph $G = (V, E)$, where V is the set of nodes and E is the set of links. For each pair of nodes $(u, v) \in V \times V$, we denote by $d(u, v)$ the latency of the network path between nodes u and v . We assume that the latency between each pair of nodes is symmetric, i.e., for any two nodes u and v , it holds that $d(u, v) = d(v, u)$. We also define $d(v, v) = 0$ for each node $v \in V$.

Without loss of generality, we assume that servers can be placed only at a particular set of nodes $Z \subseteq V$ in the network and refer to these nodes as *candidate server locations*. Z can be defined in different ways to address different server provisioning scenarios. For example, in the client-server model, the application state is maintained

by dedicated servers. Participants, known as clients, are responsible for sending user-initiated operations to the servers for execution and receiving the state updates from the servers. In this case, the candidate server locations (e.g., the data centers operated by cloud providers) are normally separate from client locations. In the peer-to-peer model, on the other hand, the clients are responsible for executing operations and maintaining the application state by themselves. In this case, to select which clients to take up the role of operation execution, the candidate server locations Z can be modeled as the set of clients.

Let $C \subseteq V$ be the set of clients in the network. To participate in the DIA, each client $c_i \in C$ needs to connect to one server. The clients are often autonomous in deciding which servers to connect to. An intuitive and widely used strategy in many applications is for the clients to connect to their nearest servers, i.e., the servers with the shortest network latency to them [30, 40, 52, 55]. Suppose that a set of locations $S \subseteq Z$ are selected to place servers. For each client $c_i \in C$, we denote by $n(c_i, S)$ the nearest server to c_i , i.e., $d(c_i, n(c_i, S)) = \min_{s \in S} d(c_i, s)$. Then, all the operations issued by client c_i would be sent to $n(c_i, S)$. On receiving the operations, server $n(c_i, S)$ forwards the operations to all the other servers for execution. Then, all the clients in C would receive the resultant state updates from their nearest servers.

To be more specific, consider the interaction between a client c_i and another client c_j that receives the status update due to c_i 's operation. Their interaction goes through their connected servers. On receiving an operation issued by c_i , its server $n(c_i, S)$ forwards the operation to c_j 's connected server $n(c_j, S)$ if they are different. Then, $n(c_j, S)$ executes the operation¹ and delivers the resultant state update to c_j to present the effect of c_i 's operation. Thus, the interaction process involves the paths from c_i to $n(c_i, S)$, from $n(c_i, S)$ to $n(c_j, S)$, and from $n(c_j, S)$ to c_j . We refer to the

¹In some applications such as instant messaging, operation execution is as simple as forwarding the message typed by the user. In other applications like online gaming, operation execution could involve more complex computation and modification to the application state.

concatenation of these paths as the *interaction path* from c_i to c_j . The length of the interaction path is given by $d(c_i, n(c_i, S)) + d(n(c_i, S), n(c_j, S)) + d(n(c_j, S), c_j)$ and represents the network latency involved in the interaction between c_i and c_j . Note that the interaction path from a client c_i to itself is the round trip between c_i and its connected server $n(c_i, S)$, whose length indicates the network latency involved for c_i to see the effect of its own operation.

We measure the interactivity performance of the DIA by the average interaction path length between all client pairs:

$$\frac{1}{|C|^2} \sum_{c_i \in C} \sum_{c_j \in C} \left(d(c_i, n(c_i, S)) + d(n(c_i, S), n(c_j, S)) + d(n(c_j, S), c_j) \right).$$

The shorter the average interaction path length, the higher the interactivity performance of the DIA. Since the total number of client pairs is fixed given the client set, to minimize the average interaction path length, it is equivalent to minimize the total interaction path length between all client pairs. Therefore, the server provisioning problem for discrete DIAs is defined as follows.

Server Provisioning Problem for Discrete DIAs. *Given a set of clients C and a set of candidate server locations Z , select a set of locations $S \subseteq Z$ to place servers for the DIA that minimizes the total interaction path length between all client pairs, i.e.,*

$$\text{minimize } \sum_{c_i \in C} \sum_{c_j \in C} \left(d(c_i, n(c_i, S)) + d(n(c_i, S), n(c_j, S)) + d(n(c_j, S), c_j) \right),$$

where $n(c_i, S)$ refers to the nearest server in S to client c_i .

In the above formulation, we focus on minimizing the network latencies involved in the interaction. No server capacity limitation is assumed at the candidate server locations. We shall discuss and evaluate how to deal with server capacity constraints in our proposed algorithms in Section 3.8.2. Furthermore, the above formulation assumes uniform weights on the interaction path lengths between all client pairs. Our

analysis and proposed algorithms can be easily extended to deal with different weights on the interaction path lengths of different client pairs.

3.2 Hardness of the Problem

The server provisioning problem for discrete DIAs is trivial if (i) the network latencies among the nodes satisfy the triangle inequality; (ii) all the nodes in the network are candidate server locations; and (iii) there is no limit on the number of server locations to select (or more precisely, the number of server locations selected can be as large as the number of clients because there is certainly no need to place more servers than the number of clients).

The triangle inequality implies that for any two clients c_i and c_j , and any two servers s_a and s_b , we have $d(c_i, s_a) + d(s_a, s_b) + d(s_b, c_j) \geq d(c_i, c_j)$. Therefore, under the above three assumptions, the optimal server provisioning solution is to place servers at all the nodes where the clients are located. In this way, each client connects to the server co-located with it so that the latencies between all clients and their nearest servers are 0. Thus, the interaction path length between any two clients c_i and c_j is simply $d(c_i, c_j)$, which is the shortest possible.

Interestingly, if *any* one of the above assumptions is relaxed, the server provisioning problem for discrete DIAs becomes NP-hard. In Section 3.2.1, we first show that the server provisioning problem is NP-hard when assumption (i) does not hold. In Sections 3.2.2 and 3.2.3, we further prove the NP-hardness of the server provisioning problem when assumptions (ii) and (iii) do not hold respectively.

3.2.1 Hardness for Networks without Triangle Inequality

Triangle inequality violations are not uncommon as Internet routing is often based on business policies and is not optimal in terms of network latency [56]. If the network latency does not satisfy the triangle inequality, we can show the NP-hardness of the

Figure 3.1: Example instances of the set cover problem and the server provisioning problem for discrete DIAs

server provisioning problem by a polynomial reduction from the minimum set cover problem [47]. Given a finite set P and a collection $\mathbb{Q}$ of its subsets, and a positive integer $k \leq |\mathbb{Q}|$, the decision version of the minimum set cover problem is to find out whether $\mathbb{Q}$ contains a subcollection $\mathbb{Q}'$ of at most k subsets such that $\bigcup_{Q \in \mathbb{Q}'} Q = P$.

Consider an instance $\mathcal{R}$ of the minimum set cover problem. Suppose that set P contains n elements: $P = \{p_1, p_2, \dots, p_n\}$, and collection $\mathbb{Q}$ contains m subsets: $\mathbb{Q} = \{Q_1, Q_2, \dots, Q_m\}$. We construct a network $G_{\mathcal{R}}$ consisting of the node set $V_C \cup V_S$, where V_C contains n nodes $V_C = \{c_1, c_2, \dots, c_n\}$, and V_S contains k groups of nodes $V_S = \bigcup_{i=1}^k V^i$. Each node $c_i \in V_C$ corresponds to an element p_i in set P . Each group V^i contains m clusters of nodes $V^i = \bigcup_{j=1}^m V_j^i$. Each cluster V_j^i corresponds to a subset Q_j , and contains $|Q_j|$ nodes that correspond to the elements of Q_j . The nodes in different clusters are separate and they can correspond to the same element in P . Thus, there are a total of $k \cdot \sum_{j=1}^m |Q_j|$ nodes in V_S .

Among the nodes in V_S , the latency between any two nodes in the same cluster or in different groups is set to 1. The latency between any two nodes in different clusters of the same group is set to L , where $L > 3n^2$. The latency between any two nodes in V_C is also set to L . The latency from a node $c_i \in V_C$ to a node $v \in V_S$ is set to 1 if v corresponds to element $p_i \in P$, and is set to L otherwise. Figure 3.1 illustrates an example of network $G_{\mathcal{R}}$, where each node pair connected by a link has latency 1, and each pair not connected by a link has latency L .

An instance $\mathcal{T}$ of the server provisioning problem in the decision version is then defined on the constructed network $G_{\mathcal{R}}$ as follows: Suppose that a client is located at each node in V_C , all the nodes in the network are candidate server locations, and there is no limit on the number of server locations to select. Can we select a set of server locations such that the total interaction path length between all client pairs is bounded by $B = 3n^2$?

We first prove that if $\mathbb{Q}$ contains a set cover of size at most k for instance $\mathcal{R}$, there must exist a server placement with total interaction path length bounded by B for instance $\mathcal{T}$. Let $\mathbb{Q}' = \{Q_{x_1}, Q_{x_2}, \dots, Q_{x_l}\}$ (where $1 \leq l \leq k$) be a set cover of size not exceeding k for instance $\mathcal{R}$. Then, placing servers at all the nodes in clusters $V_{x_1}^1, V_{x_2}^2, \dots, V_{x_l}^l$ is a valid server provisioning solution for instance $\mathcal{T}$. In fact, since $\mathbb{Q}'$ is a set cover, each element $p_i \in P$ is contained in at least one subset among $Q_{x_1}, Q_{x_2}, \dots, Q_{x_l}$. Thus, for each client $c_i \in V_C$, there exists at least one node in $V_{x_1}^1 \cup V_{x_2}^2 \cup \dots \cup V_{x_l}^l$ having latency 1 to c_i . As a result, the latencies between all the clients and their nearest servers are 1. In addition, the latency between any two servers in $V_{x_1}^1 \cup V_{x_2}^2 \cup \dots \cup V_{x_l}^l$ is 1 because they either come from the same cluster or from different groups. Therefore, the interaction path length between each pair of clients is bounded by $1 + 1 + 1 = 3$. Since there are n^2 interaction paths in total, the total interaction path length is bounded by $3n^2 = B$.

Next, we prove that if a valid server placement solution can be found for instance

$\mathcal{T}$, there must exist a set cover of size at most k for instance $\mathcal{R}$. Let $S \subseteq V_C \cup V_S$ be the set of server locations selected in a valid solution of instance $\mathcal{T}$. Without loss of generality, we assume that the server at each location in S is connected by at least one client.² Since each client connects to its nearest server, if a server is placed at a node in V_C , the client at that node should connect to this server. Note that the latency between any two nodes in V_C is L . If servers are placed at more than one node in V_C , the interaction path length between the clients at these nodes becomes $L > 3n^2 = B$. Therefore, at most one node in V_C can be selected to place a server, i.e., $|S \cap V_C| \leq 1$.

If $|S \cap V_C| = 1$, there is exactly one server located in V_C and the remaining servers are all located in V_S . Let the server in V_C be located at node c_x . Since the latency between any two nodes in V_C is L , to bound the total interaction path length by $B < L$, the clients in $V_C \setminus \{c_x\}$ should all connect to the servers in V_S . Moreover, each of these clients must have latency 1 to its nearest server because the latency from a client to a server in V_S is either 1 or L . This implies that the connecting server of each client $c_y \in V_C \setminus \{c_x\}$ must correspond to the element $p_y \in P$. Since $c_x \neq c_y$, we have $p_x \neq p_y$. Therefore, the latency from c_x to the connecting server of c_y must be L since they correspond to different elements in P . This implies that the interaction path length between clients c_x and c_y exceeds the bound B . So, there is no valid solution of instance $\mathcal{T}$ satisfying $|S \cap V_C| = 1$.

If $|S \cap V_C| = 0$, all the servers are placed in V_S . Similar to the former case, since the total interaction path length does not exceed B , each client c_i must have latency 1 to its nearest server. This implies that each element $p_i \in P$ is covered by the subset Q_j corresponding to the node cluster of c_i 's nearest server. Thus, $\mathcal{Q}' = \{Q_j \mid \exists i, V_j^i \cap S \neq \emptyset\}$ is a set cover for P . Furthermore, to cap the total interaction path length by B , all the servers must have latency 1 from each other.

²The server locations not connected by any clients can simply be removed from S .

Thus, for each group of nodes V^i , all the servers in $V^i \cap S$ must come from the same cluster of V^i . Since there are k groups of nodes $V^1, V^2, \dots, V^k$ in V_S , we have $|\mathbb{Q}'| \leq k$. Therefore, $\mathbb{Q}'$ is a set cover of size at most k .

Hence, a set cover of size at most k can be found for instance $\mathcal{R}$ if and only if there exists a server placement with total interaction path length bounded by B for instance $\mathcal{T}$. Thus, the server provisioning problem for discrete DIAs for networks without the triangle inequality is NP-hard.

The above hardness analysis also leads to the following non-approximability result.

Theorem 1. *If $P \neq NP$, no polynomial-time algorithm can achieve any constant approximation ratio for the server provisioning problem for discrete DIAs on networks without the triangle inequality.*

Proof. For any instance of the minimum set cover problem, if there exists a set cover of size at most k , the total interaction path length of an optimal server placement must be bounded by $B = 3n^2$ in the network constructed above. Any non-optimal server placement, however, produces a total interaction path length greater than L . This is because in any non-optimal placement, either a client connects to a server that has a latency L to it, or the latency between a certain pair of servers is L . In either case, there is at least one interaction path longer than L , so the total interaction path length exceeds L . Similarly, if there does not exist any set cover of size at most k in the set cover problem, the total interaction path length of an optimal server placement in the constructed network is also greater than L .

Assume on the contrary that there exists a polynomial-time server provisioning algorithm with a constant approximation ratio of β . For any instance of the set cover problem, we set $L = \beta B = 3n^2\beta$ in the constructed network and then run the β -approximation algorithm. If the output server placement produces a total interaction path length larger than L , it indicates that the optimal server placement has a total

interaction path length larger than B . So, there does not exist any set cover of size at most k . On the other hand, if the output server placement has a total interaction path length within L , it must be bounded by B according to the above analysis. Thus, the optimal server placement would also have a total interaction path length within B so that there exists a set cover of size at most k . Therefore, this implies a polynomial-time algorithm for the set cover problem, which contradicts to $P \neq NP$.

Hence, the theorem is proven. $\square$

3.2.2 Hardness for Restricted Choices of Server Locations

If not all the nodes in the network are candidate server locations, the server provisioning problem becomes NP-hard as well. We can again prove it by a polynomial reduction from the minimum set cover problem.

Given an instance $\mathcal{R}$ of the minimum set cover problem, we construct the same network $G_{\mathcal{R}}$ as in the previous section, except that the node pairs that had latency L earlier have latency 2 now to satisfy the triangle inequality. That is, in the illustration of Figure 3.1, a node pair has latency 1 if they are connected by a link and has latency 2 otherwise. Suppose that a client is located at each node in V_C , only the nodes in V_S are candidate server locations, and there is no limit on the number of server locations to select. An instance $\mathcal{Y}$ of the server provisioning problem is then defined on network $G_{\mathcal{R}}$ by setting a bound $H = 3n^2 - n$ on the total interaction path length between all client pairs.

We first show that a set cover of size at most k for instance $\mathcal{R}$ gives rise to a valid server provisioning solution for instance $\mathcal{Y}$. Again, let $\mathbb{Q}' = \{Q_{x_1}, Q_{x_2}, \dots, Q_{x_l}\}$ (where $1 \leq l \leq k$) be a set cover of size not exceeding k . Then, a valid solution for instance $\mathcal{Y}$ is to place servers at all the nodes in clusters $V_{x_1}^1, V_{x_2}^2, \dots, V_{x_l}^l$. Similar to the argument in Section 3.2.1, under such server placement, the latencies between all clients and their nearest servers are 1, and the latency between any two servers is

also 1. Thus, the interaction path from a client to itself has length $1 + 1 = 2$, and the interaction path length between two distinct clients is bounded by $1 + 1 + 1 = 3$. Note that there are $n(n - 1)$ pairs of distinct clients. Therefore, the total interaction path length is bounded by $3n(n - 1) + 2n = 3n^2 - n = H$.

Next, we show that a valid server provisioning solution for instance $\mathcal{Y}$ gives rise to a set cover of size at most k for instance $\mathcal{R}$. Let $S \subseteq V_S$ be a set of server locations that produces a total interaction path length not exceeding H . Without loss of generality, assume that the server at each location in S is connected by at least one client. Note that each node in V_S has latency 1 to exactly one client in V_C and has latency 2 to all the other clients. As a result, if two distinct clients in V_C connect to the same server in S , their interaction path length is at least $1 + 2 = 3$. On the other hand, the latency between any two nodes in V_S is at least 1. Thus, if two distinct clients connect to different servers in S , their interaction path length is at least $1 + 1 + 1 = 3$. Therefore, regardless of server placement, the interaction path length between any two distinct clients is at least 3. Since there are $n(n - 1)$ pairs of distinct clients, this implies that the total interaction path length from all clients to themselves under server placement S cannot exceed $H - 3n(n - 1) = 2n$.

Since the shortest latency from each client to the nodes in V_S is 1, it follows that the latencies from all the clients to their nearest servers must be exactly 1. Therefore, each element $p_i \in P$ is covered by the subset Q_j corresponding to the node cluster of c_i 's nearest server. That is, $\mathbb{Q}' = \{Q_j \mid \exists i, V_j^i \cap S \neq \emptyset\}$ is a set cover for P .

To limit the total interaction path length by H , the interaction path length between any two distinct clients must now be exactly 3. This implies that the latency between any two servers in S must be 1. Thus, for each group of nodes V^i , all the servers in $V^i \cap S$ must come from the same cluster of V^i . It follows that $|\mathbb{Q}'| \leq k$. Therefore, $\mathbb{Q}'$ is a set cover of size at most k .

In summary, there exists a set cover of size at most k for instance $\mathcal{R}$ if and only if

Figure 3.2: Example instances of the 3DM problem and the server provisioning problem for discrete DIAs

a server placement with total interaction path length bounded by H can be found for instance $\mathcal{Y}$. Thus, the server provisioning problem for discrete DIAs with restricted choices of server locations is NP-hard.

3.2.3 Hardness for Limited Number of Server Locations to Select

If the DIA operator can deploy only a limited number of servers due to budget restriction, a cap can be set on the number of server locations to select. If the cap is less than the number of clients, the server provisioning problem is also NP-hard. This can be proved by a polynomial reduction from the 3-Dimensional Matching (3DM) problem [47]. The decision version of the 3DM problem is defined as follows: Given three disjoint sets W , X and Y each having k elements, and a set of triples $M \subseteq W \times X \times Y$, find out whether M contains a 3-dimensional matching, i.e., whether there exists a subset $M' \subseteq M$ such that $|M'| = k$ and any two triples (w_a, x_a, y_a) and $(w_b, x_b, y_b) \in M'$ satisfy $w_a \neq w_b$, $x_a \neq x_b$ and $y_a \neq y_b$.

Given an instance $\mathcal{U}$ of the 3DM problem, we construct a network $G_{\mathcal{U}}$ consisting of the node set $V_C \cup V_M$. The set V_C contains $3k$ nodes, each corresponding to an element in $W \cup X \cup Y$ of instance $\mathcal{U}$. The set V_M contains the same number of

nodes as the size of M in instance $\mathcal{U}$. Each node in V_M corresponds to a triple in M , and establishes links of latency 2 to the three nodes in V_C corresponding to its three coordinates. For example, in Figure 3.2, node v_4^m , which corresponds to the triple $m_4 = (w_1, x_2, y_3)$, has three links to nodes v_1^w , v_2^x and v_3^y in V_C . In addition, all the nodes in V_M are inter-connected with each other via links of latency 1. Suppose that a client is located at each node in V_C , and all the nodes in network $G_{\mathcal{U}}$ are candidate server locations. An instance $\mathcal{F}$ of the server provisioning problem is then defined on network $G_{\mathcal{U}}$ by capping the number of server locations to select at k and setting a bound $A = 45k^2 - 9k$ on the total interaction path length between all client pairs.

We first prove that if M contains a matching for instance $\mathcal{U}$, there must exist a server placement with total interaction path length bounded by A for instance $\mathcal{F}$. Suppose that $M' \subseteq M$ is a matching. Let S be the set of k nodes in V_M that represent the triples in M' (for example, nodes v_1^m , v_2^m and v_3^m in Figure 3.2). We construct a server placement by selecting all the k nodes in S as the server locations. Then, each client in V_C has exactly one adjacent node in S . Thus, each client connects to the server placed at its adjacent node in S and the latency between them is 2. Since all the servers in S are inter-connected by links of latency 1, and each server has exactly three adjacent clients, the total interaction path length under placement S is given by

$$\begin{aligned} & (2 + 0 + 2) \cdot 9k + (2 + 1 + 2) \cdot (9k^2 - 9k) \\ & = 45k^2 - 9k = A. \end{aligned}$$

Next, we prove that M contains a matching for instance $\mathcal{U}$ if a valid server placement solution can be found for instance $\mathcal{F}$. Consider a set S of up to k server locations in network $G_{\mathcal{U}}$. Let b_i ($i \geq 0$) be the number of clients that have latency i to their nearest servers in S . Since there are a total of $3k$ clients and all the links incident on clients have latency 2, we have $\sum_{i \geq 0} b_i = 3k$ and $b_1 = 0$. Note that there are b_0 servers of S located in V_C . So, the number of servers located in V_M is at most $k - b_0$.

If a client has latency 2 to its nearest server, this server must be adjacent to the client and thus is located in V_M . Since each server in V_M has three adjacent clients, we have $b_2 \leq 3 \cdot (k - b_0)$. Thus, the total length of the interaction paths from the clients to themselves is

$$\begin{aligned} \sum_{i \geq 0} 2i \cdot b_i &= 0 \cdot b_0 + 4 \cdot b_2 + \sum_{i \geq 3} 2i \cdot b_i \\ &\geq 4 \cdot b_2 + 6 \cdot \sum_{i \geq 3} b_i \end{aligned} \quad (3.1)$$

$$\begin{aligned} &= 4 \cdot b_2 + 6 \cdot (3k - b_2 - b_0) \\ &= 18k - 2 \cdot b_2 - 6 \cdot b_0 \\ &\geq 18k - 2 \cdot (3 \cdot (k - b_0)) - 6 \cdot b_0 \\ &= 12k. \end{aligned} \quad (3.2)$$

Since all the links incident on clients have latency 2 and the clients are not adjacent to each other, the shortest possible interaction path between two distinct clients has length 4. Let p_i ($i \geq 4$) be the number of pairs of distinct clients whose interaction path length is i under server placement S . Since there are $3k(3k - 1)$ pairs of distinct clients in total, we have $\sum_{i \geq 4} p_i = 3k(3k - 1)$. If any two distinct clients have an interaction path length of 4, there must be a server in V_M that is adjacent to both clients. Each server in V_M has three adjacent clients and can thus support at most $3 \cdot 2 = 6$ interaction paths of length 4 between distinct clients. Since there are at most k servers, we have $p_4 \leq 6k$. Therefore, the total length of the interaction paths between distinct clients is

$$\begin{aligned} \sum_{i \geq 4} i \cdot p_i &= 4 \cdot p_4 + \sum_{i \geq 5} i \cdot p_i \\ &\geq 4 \cdot p_4 + 5 \cdot \sum_{i \geq 5} p_i \end{aligned} \quad (3.3)$$

$$\begin{aligned} &= 4 \cdot p_4 + 5 \cdot (3k(3k - 1) - p_4) \\ &= 45k^2 - 15k - p_4 \\ &\geq 45k^2 - 15k - 6k \end{aligned} \quad (3.4)$$

$$= 45k^2 - 21k.$$

Thus, the total interaction path length is

$$\begin{aligned} \sum_{i \geq 0} 2i \cdot b_i + \sum_{i \geq 4} i \cdot p_i &\geq 12k + 45k^2 - 21k \\ &= 45k^2 - 9k = A. \end{aligned}$$

Therefore, if server placement S has a total interaction path length bounded by A , the equality must be satisfied at all the steps (3.1), (3.2), (3.3) and (3.4) in the above derivation.

Equality at step (3.3) implies that $\sum_{i \geq 5} p_i = 0$. So, at most one client can have latency 3 to its nearest server in S , i.e., $b_3 \leq 1$. Equality at step (3.1) indicates $\sum_{i \geq 3} b_i = 0$. Thus, $b_0 + b_2 + b_3 = 3k$. It follows from equality at step (3.2) (i.e., $b_2 = 3(k - b_0)$) that $b_3 = 2 \cdot b_0$, so b_3 is an even number. Therefore, b_3 must be 0, and b_0 is also 0. As a result, we have $b_2 = 3k$, i.e., the latency from each client to its nearest server in S must be 2. Thus, all the server locations in S are selected from node set V_M . Since each node in V_M has only three adjacent clients, exactly k server locations must be selected from V_M and these server locations cannot have any adjacent client in common. Therefore, the k triples of M corresponding to the k server locations selected in V_M form a matching.

Hence, M contains a matching for instance $\mathcal{U}$ if and only if there exists a server placement with total interaction path length bounded by A for instance $\mathcal{F}$. Thus, the server provisioning problem for discrete DIAs with limited number of server locations to select is NP-hard.

In the following sections, we present four different server placements for the server provisioning problem for discrete DIAs. The computation of these four algorithms is based simply on the network latencies between clients and candidate server locations, which can be acquired with existing tools like ping and King [57]. We start with the classical k -median server placement, which is related to our server provisioning problem for discrete DIAs. We show that for networks with the triangle inequality,

Table 3.1: Summary of different server placement algorithms for discrete DIAs

	approximation ratio for networks with triangle inequality	approximation ratio for networks with α -triangle inequality	time complexity
k -median	3	$\max \left\{ \frac{1}{2}\alpha^3 + \frac{3}{2}\alpha^2 + 1, \frac{2}{3}\alpha^3 + \alpha^2 + \frac{2}{3} \right\}$	NP-hard
NEAREST	3	$\max \left\{ \frac{1}{2}\alpha^3 + \frac{3}{2}\alpha^2 + 1, \frac{2}{3}\alpha^3 + \alpha^2 + \frac{2}{3} \right\}$	$O(Z C)$
T-GREEDY	2	2α	$O(k Z (C + k^2))$
T-BETTER	$\frac{5}{3}$	$\min \left\{ \frac{2\alpha^4 + \alpha^3 + 2\alpha}{\alpha^3 + \alpha + 1}, 2\alpha \right\}$	$O(k Z (C + k^2))$

the k -median server placement achieves an approximation ratio of 3 for our server provisioning problem. However, considering the difficulty in finding the k -median placement due to its NP-hardness, we further propose a NEAREST server placement that achieves the same approximation ratio as the k -median placement, and can be computed in polynomial time. Then, we improve the approximation ratio to 2 by proposing an efficient T-GREEDY server placement heuristic. This is followed by the fourth server placement algorithm called T-BETTER, which further improves the approximation ratio to $\frac{5}{3}$. After that, we conduct theoretical analysis on the asymptotic approximability of these four different server placements for networks with triangle inequality violations. Let $|C|$ denote the number of clients, $|Z|$ denote the number of candidate server locations in the network, and k denote the cap on the number of server locations to select. Table 3.1 summarizes the approximation ratios and the computational complexities of the four algorithms.

3.3 k -Median Server Placement

In this section, we study the performance of the classical k -median server placement for DIAs. The k -median problem has been widely used to model server placement in the Internet [44–46]. The k -median problem seeks to select a given number of k locations in the network to host servers such that the total distance (latency) from the clients to their nearest servers is minimized. Our server provisioning problem for

discrete DIAs shares some similarity with the k -median problem in that we also aim to find server placement that minimizes an aggregate latency metric. However, in our problem, the interaction time between the clients in DIAs includes not only the latencies from the clients to their connected servers but also the latencies between their connected servers. When a limit k is set on the number of server locations to select in our server provisioning problem, we have the following approximability result of the k -median server placement.

Theorem 2. *For networks with the triangle inequality, the k -median server placement has an approximation ratio of 3 for the server provisioning problem for discrete DIAs with a limit k on the number of server locations to select.*

Proof. Consider a network satisfying the triangle inequality. Let $C = \{c_1, c_2, \dots, c_n\}$ be the set of clients, and Z be the set of candidate server locations in the network. Suppose that $S_O \subseteq Z$ is a set of optimal server locations for our server provisioning problem. We denote by $o_i \in S_O$ as the nearest server of each client c_i ($1 \leq i \leq n$) under placement S_O .

Suppose that the k -median server placement is $S_M \subseteq Z$. That is, S_M is the set of k server locations that minimize the total distance from all the clients to their nearest servers. We denote by $m_i \in S_M$ as the nearest server of each client c_i under placement S_M .

If the k -median server placement S_M is applied to our server provisioning problem, the total interaction path length is given by

$$\begin{aligned} T_M &= \sum_{i=1}^n \sum_{j=1}^n (d(c_i, m_i) + d(m_i, m_j) + d(m_j, c_j)) \\ &= 2n \sum_{i=1}^n d(c_i, m_i) + \sum_{i=1}^n \sum_{j=1}^n d(m_i, m_j). \end{aligned}$$

By the triangle inequality, for any two clients c_i and c_j , we have

$$d(m_i, m_j) \leq d(m_i, c_i) + d(c_i, c_j) + d(c_j, m_j).$$

Thus,

$$\begin{aligned}
 T_M &\leq 2n \sum_{i=1}^n d(c_i, m_i) \\
 &\quad + \sum_{i=1}^n \sum_{j=1}^n (d(m_i, c_i) + d(c_i, c_j) + d(c_j, m_j)) \\
 &= 4n \sum_{i=1}^n d(c_i, m_i) + \sum_{i=1}^n \sum_{j=1}^n d(c_i, c_j).
 \end{aligned}$$

Based on the triangle inequality, we also have

$$d(c_i, c_j) \leq d(c_i, o_i) + d(o_i, o_j) + d(o_j, c_j).$$

Therefore,

$$\begin{aligned}
 T_M &\leq 4n \sum_{i=1}^n d(c_i, m_i) \\
 &\quad + \sum_{i=1}^n \sum_{j=1}^n (d(c_i, o_i) + d(o_i, o_j) + d(o_j, c_j)) \\
 &= 4n \sum_{i=1}^n d(c_i, m_i) \\
 &\quad + 2n \sum_{i=1}^n d(c_i, o_i) + \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j). \tag{3.5}
 \end{aligned}$$

The definition of the k -median placement implies that $\sum_{i=1}^n d(c_i, m_i)$ is the minimum achievable total distance from all the clients to their nearest servers among all possible placements of up to k servers. Thus, we have

$$\sum_{i=1}^n d(c_i, m_i) \leq \sum_{i=1}^n d(c_i, o_i).$$

Therefore,

$$\begin{aligned}
 T_M &\leq 4n \sum_{i=1}^n d(c_i, o_i) + 2n \sum_{i=1}^n d(c_i, o_i) + \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j) \\
 &= 6n \sum_{i=1}^n d(c_i, o_i) + \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j).
 \end{aligned}$$

This implies that

$$T_M \leq 3 \cdot \left(2n \sum_{i=1}^n d(c_i, o_i) + \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j) \right).$$

Note that the total interaction path length under the optimal server placement S_O is given by

$$\begin{aligned} T_O &= \sum_{i=1}^n \sum_{j=1}^n (d(c_i, o_i) + d(o_i, o_j) + d(o_j, c_j)) \\ &= 2n \sum_{i=1}^n d(c_i, o_i) + \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j). \end{aligned}$$

It follows that

$$T_M \leq 3 \cdot T_O.$$

Hence, the theorem is proven. $\square$

The proof of Theorem 2 gives rise to the following result for the server provisioning problem when there is no limit on the number of server locations to select.

Corollary 1. *For each instance of the server provisioning problem for discrete DIAs on a network satisfying the triangle inequality, if an optimal server placement selects k server locations, then the k -median server placement produces a total interaction path length within three times of that in the optimal server placement.*

The approximation ratio 3 of the k -median server placement is tight. Figure 3.3 shows a tight example. There are three disjoint sets of nodes $V_C = \{c_1, c_2, \dots, c_n\}$, $V_M = \{m_1, m_2, \dots, m_n\}$ and $V_O = \{o_1, o_2, \dots, o_n\}$ in the network. Each node c_i has latency 1 to node m_i and latency $1 + \varepsilon$ to node o_i , where $0 < \varepsilon < 1$. The latency between any two nodes o_i and o_j ($i \neq j$) is ε . The latency between any two nodes c_i and c_j ($i \neq j$) is 2. The remaining node pairs have network latencies given by the lengths of shortest paths. Due to space limitations, only the network latencies between some node pairs are shown in the figure. Apparently, the network satisfies the triangle inequality. Suppose that a client is located at each node in V_C , and all the nodes in V_M and V_O are candidate server locations. The optimal server placement for our server provisioning problem is to place servers at all the n nodes in V_O . In

Figure 3.3: The approximation ratio 3 of the k -median placement is tight.

this case, each client c_i connects to the server located at o_i and the resultant total interaction path length is

$$\begin{aligned}
 & 2n \sum_{i=1}^n d(c_i, o_i) + \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j) \\
 &= 2n^2(1 + \varepsilon) + n(n-1)\varepsilon \\
 &= 2n^2 + (3n^2 - n)\varepsilon.
 \end{aligned}$$

On the other hand, the n -median server placement selects the n nodes in V_M to place servers. Under such placement, each client c_i connects to its nearest server at m_i .

This results in a total interaction path length of

$$\begin{aligned}
 & 2n \sum_{i=1}^n d(c_i, m_i) + \sum_{i=1}^n \sum_{j=1}^n d(m_i, m_j) \\
 &= 2n^2 + 4n(n-1) \\
 &= 6n^2 - 4n.
 \end{aligned}$$

Therefore, the ratio between the results of the n -median placement and the optimal

server placement is $\frac{6n^2-4n}{2n^2+(3n^2-n)\varepsilon}$. When n goes towards infinity and ε approaches 0, this ratio can be arbitrarily close to

$$\begin{aligned} & \lim_{n \rightarrow \infty} \lim_{\varepsilon \rightarrow 0} \frac{6n^2 - 4n}{2n^2 + (3n^2 - n)\varepsilon} \\ &= \lim_{n \rightarrow \infty} \frac{6n^2 - 4n}{2n^2} \\ &= 3. \end{aligned}$$

3.4 Nearest Server Placement

Although the k -median server placement achieves an approximation ratio 3 for the server provisioning problem, the k -median problem itself is also NP-hard [47] and difficult to solve. In this section, we introduce a simple server placement called NEAREST that achieves the same approximation ratio as the k -median placement, and can be computed in polynomial time.

Given a set of clients C , and a set of candidate server locations Z , we find the nearest candidate server location $n_i \in Z$ to each client $c_i \in C$, i.e., $d(c_i, n_i) = \min_{s \in Z} d(c_i, s)$. The NEAREST server placement S_N selects the collection of nearest candidate server locations of all the clients, i.e., $S_N = \bigcup_{c_i \in C} \{n_i\}$. Since the nearest candidate server location of each client can be computed in $O(|Z|)$ time, the time complexity of the NEAREST server placement is $O(|Z| \cdot |C|)$.

Theorem 3. *For networks with the triangle inequality, the NEAREST server placement has an approximation ratio of 3 for the server provisioning problem for discrete DIAs.*

Proof. Suppose that $C = \{c_1, c_2, \dots, c_n\}$ is the set of clients and Z is the set of candidate server locations in the network. Let n_i be the nearest candidate server location in Z to client c_i ($1 \leq i \leq n$). Under the NEAREST server placement S_N , each client c_i naturally connects to the server at n_i . Thus, the total interaction path

length under placement S_N is given by

$$\begin{aligned} T_N &= \sum_{i=1}^n \sum_{j=1}^n (d(c_i, n_i) + d(n_i, n_j) + d(n_j, c_j)) \\ &= 2n \sum_{i=1}^n d(c_i, n_i) + \sum_{i=1}^n \sum_{j=1}^n d(n_i, n_j). \end{aligned}$$

Similar to the derivation of (3.5) in Theorem 2, we have

$$T_N \leq 4n \sum_{i=1}^n d(c_i, n_i) + 2n \sum_{i=1}^n d(c_i, o_i) + \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j),$$

where o_i is the nearest server of each client c_i under an optimal server placement S_O .

For each client c_i , since $d(c_i, n_i) = \min_{s \in Z} d(c_i, s) \leq d(c_i, o_i)$, it follows that

$$\sum_{i=1}^n d(c_i, n_i) \leq \sum_{i=1}^n d(c_i, o_i).$$

Therefore,

$$\begin{aligned} T_N &\leq 4n \sum_{i=1}^n d(c_i, o_i) + 2n \sum_{i=1}^n d(c_i, o_i) + \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j) \\ &= 6n \sum_{i=1}^n d(c_i, o_i) + \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j). \\ &\leq 3 \cdot \left(2n \sum_{i=1}^n d(c_i, o_i) + \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j) \right) \\ &= 3 \cdot T_O, \end{aligned} \tag{3.6}$$

where T_O is the total interaction path length under the optimal server placement S_O .

Hence, the theorem is proven. $\square$

The approximation ratio 3 of the NEAREST server placement is also tight. The example given in Figure 3.3 is also applicable to the NEAREST server placement. In this example, since the nearest candidate server location of each client c_i is m_i , the NEAREST server placement would place servers at all the n nodes in V_M , which is the same as the n -median server placement. Therefore, the example in Figure 3.3 also shows the tightness of the approximation ratio 3 for the NEAREST server placement.

3.5 T-Greedy Server Placement

In this section, we present an efficient greedy algorithm called T-GREEDY for server provisioning, which further improves the approximation ratio to 2 for networks with the triangle inequality.

Without loss of generality, the T-GREEDY algorithm takes a parameter k that indicates the cap on the number of server locations to select. In the case where there is no limit on the number of server locations to select, k may simply be set to the number of candidate server locations. The T-GREEDY algorithm starts with an empty set of server locations S . In each iteration, the algorithm adds to S an unselected candidate location that leads to the maximum reduction in the total interaction path length. The algorithm terminates when no new server location can be further selected to reduce the total interaction path length or the number of server locations selected reaches k .

Algorithm 1 shows the pseudo code of the T-GREEDY algorithm. Suppose that C is the set of clients and Z is the set of candidate server locations in the network. In each iteration, the algorithm evaluates each unselected candidate location $s \in Z \setminus S$ and calculates the total interaction path length D if s is added to S (lines 6 to 17). The candidate location that results in the minimum value of D and the corresponding D value are recorded in s^* and D^* respectively (lines 16 to 17). At the end of each iteration, the algorithm keeps the nearest servers of all clients under the current placement S in an array $n[\cdot]$ (lines 21 to 23). When evaluating a candidate server location s in the next iteration, we can find out the nearest server of each client c by simply comparing the latency from c to $n[c]$ with the latency from c to s (lines 11 to 14). In this way, the efficiency of the algorithm is improved by avoiding repeatedly comparing the latencies between each client and all the selected server locations.

Suppose that under a server placement S , the number of clients connecting to each

```

1:  $S \leftarrow \emptyset$ ,  $D_{current} \leftarrow \infty$ ;
2: for all  $c \in C$  do
3:    $n[c] \leftarrow \emptyset$ ; //nearest server to  $c$ 
4:   for  $i = 1$  to  $k$  do
5:      $D^* \leftarrow \infty$ ;
6:     for all  $s \in Z \setminus S$  do
7:       for all  $s_i \in S \cup \{s\}$  do
8:          $w[s_i] \leftarrow 0$ ; //number of clients connecting to  $s_i$ 
9:          $D \leftarrow 0$ ; //total interaction path length when evaluating  $s$ 
10:        for all  $c \in C$  do
11:          if  $n[c] = \emptyset$  or  $d(c, s) < d(c, n[c])$  then
12:             $w[s] \leftarrow w[s] + 1$ ;  $D \leftarrow D + 2 \cdot |C| \cdot d(c, s)$ ;
13:          else
14:             $w[n[c]] \leftarrow w[n[c]] + 1$ ;  $D \leftarrow D + 2 \cdot |C| \cdot d(c, n[c])$ ;
15:           $D \leftarrow D + \sum_{s_i \in S \cup \{s\}} \sum_{s_j \in S \cup \{s\}} w[s_i] \cdot w[s_j] \cdot d(s_i, s_j)$ ;
16:          if  $D < D^*$  then
17:             $D^* \leftarrow D$ ;  $s^* \leftarrow s$ ;
18:        if  $D^* \geq D_{current}$  then
19:          break; //stop if cannot further reduce interaction path length
20:         $S \leftarrow S \cup \{s^*\}$ ;  $D_{current} \leftarrow D^*$ ;
21:        for all  $c \in C$  do
22:          if  $n[c] = \emptyset$  or  $d(c, s^*) < d(c, n[c])$  then
23:             $n[c] \leftarrow s^*$ ; //update the nearest server of client  $c$ 
24: output  $S$ ;
    
```

Algorithm 1: T-GREEDY server provisioning algorithm

server $s_i \in S$ is $w(s_i)$. Then, the total interaction path length between all client pairs can be rewritten as

$$\begin{aligned}
 & \sum_{c \in C} \sum_{c' \in C} \left(d(c, n(c, S)) + d(n(c, S), n(c', S)) \right. \\
 & \quad \left. + d(n(c', S), c') \right) \\
 &= |C| \cdot \sum_{c \in C} d(c, n(c, S)) + |C| \cdot \sum_{c' \in C} d(n(c', S), c') \\
 & \quad + \sum_{c \in C} \sum_{c' \in C} d(n(c, S), n(c', S)) \\
 &= 2|C| \sum_{c \in C} d(c, n(c, S)) \\
 & \quad + \sum_{s_i \in S} \sum_{s_j \in S} w(s_i) \cdot w(s_j) \cdot d(s_i, s_j). \tag{3.7}
 \end{aligned}$$

Thus, to calculate the total interaction path length resulting from adding a candidate location s to S , the T-GREEDY algorithm first counts the number of clients connecting to each server $s_i \in S \cup \{s\}$ and records it in $w[s_i]$. Then, the total interaction

path length is calculated according to the above formula (3.7) (lines 10 to 15). The computational complexity of the total interaction path length is $O(|C| + k^2)$. Since the number of candidate server locations to evaluate in each iteration is capped by $|Z|$ (where Z is the set of candidate server locations) and there are at most k iterations, the total time complexity of the T-GREEDY algorithm is $O(k|Z|(|C| + k^2))$.

We can show that when the network latency satisfies the triangle inequality, the above T-GREEDY algorithm produces a total interaction path length within two times of that in an optimal server placement, irrespective of whether the choices of server locations are restricted and the cap on the number of server locations to select (if any).

Theorem 4. *For networks with the triangle inequality, the T-GREEDY server placement has an approximation ratio of 2 for the server provisioning problem for discrete DIAs.*

Proof. Suppose that $C = \{c_1, c_2, \dots, c_n\}$ is the set of clients and Z is the set of candidate server locations in the network. Let $S_O \subseteq Z$ be the set of server locations selected by an optimal server placement. For each client $c_i \in C$, denote by o_i the nearest server of c_i in S_O . Then, the total interaction path length under the optimal server placement S_O is given by

$$\begin{aligned} T_O &= \sum_{i=1}^n \sum_{j=1}^n \left(d(c_i, o_i) + d(o_i, o_j) + d(o_j, c_j) \right) \\ &= 2n \sum_{i=1}^n d(c_i, o_i) + \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j). \end{aligned}$$

If only one server is placed at a location s in the network, all the clients would connect to that server and thus the total interaction path length is given by $2n \cdot \sum_{c \in C} d(c, s)$.

Suppose that s^* is the first server location selected by the T-GREEDY algorithm. Since the T-GREEDY algorithm always selects the best known server location in each

iteration, the total interaction path length resulting from selecting s^* must not exceed that of selecting any other candidate server location. Thus,

$$\begin{aligned} 2n \cdot \sum_{i=1}^n d(c_i, s^*) &\leq 2n \cdot \sum_{i=1}^n d(c_i, o_1), \\ 2n \cdot \sum_{i=1}^n d(c_i, s^*) &\leq 2n \cdot \sum_{i=1}^n d(c_i, o_2), \\ &\dots\dots \end{aligned}$$

and

$$2n \cdot \sum_{i=1}^n d(c_i, s^*) \leq 2n \cdot \sum_{i=1}^n d(c_i, o_n),$$

Taking an average of the interaction path lengths on the right sides of the above inequalities, we have

$$\begin{aligned} 2n \cdot \sum_{i=1}^n d(c_i, s^*) &\leq \frac{1}{n} \cdot \sum_{j=1}^n \left(2n \cdot \sum_{i=1}^n d(c_i, o_j) \right) \\ &= 2 \cdot \sum_{j=1}^n \sum_{i=1}^n d(c_i, o_j) \\ &= 2 \cdot \sum_{i=1}^n \left(d(c_i, o_i) + \sum_{j \neq i} d(c_i, o_j) \right). \end{aligned} \tag{3.8}$$

By the triangle inequality, for any client c_i and any o_j where $j \neq i$, we have

$$d(c_i, o_j) \leq d(c_i, o_i) + d(o_i, o_j).$$

Therefore,

$$\begin{aligned} &2n \cdot \sum_{i=1}^n d(c_i, s^*) \\ &\leq 2 \cdot \sum_{i=1}^n \left(d(c_i, o_i) + \sum_{j \neq i} (d(c_i, o_i) + d(o_i, o_j)) \right) \\ &= 2 \cdot \sum_{i=1}^n \left(n \cdot d(c_i, o_i) + \sum_{j \neq i} d(o_i, o_j) \right) \\ &= 2 \cdot \sum_{i=1}^n \left(n \cdot d(c_i, o_i) + \sum_{j=1}^n d(o_i, o_j) \right) \\ &= 2n \sum_{i=1}^n d(c_i, o_i) + 2 \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j). \end{aligned} \tag{3.9}$$

Figure 3.4: The approximation ratio 2 of the T-GREEDY algorithm is tight.

It follows that

$$\begin{aligned} 2n \cdot \sum_{i=1}^n d(c_i, s^*) &\leq 2 \cdot \left(2n \sum_{i=1}^n d(c_i, o_i) + \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j) \right) \\ &= 2 \cdot T_O. \end{aligned}$$

This indicates that the total interaction path length on selecting the first server location in the T-GREEDY algorithm is within two times of that in an optimal server placement. Since the T-GREEDY algorithm adds a new server location in each subsequent iteration only if it reduces the total interaction path length, the total interaction path length T_G eventually produced by T-GREEDY must also be within two times of the optimum, i.e.,

$$T_G \leq 2n \cdot \sum_{i=1}^n d(c_i, s^*) \leq 2 \cdot T_O.$$

Hence, the theorem is proven. $\square$

The approximation ratio 2 of the T-GREEDY algorithm is tight. Figure 3.4 presents a tight example in which the network contains two client groups $C_A = \{c_1, c_2, c_3\}$ and $C_B = \{c_4, c_5, c_6\}$, and an additional node g . The latency between any two nodes in the same client group is 1. The latency between any two nodes in

different client groups is $\frac{4}{3} + 3\delta$, where $\delta > 0$. All the nodes in C_A and C_B have latencies $1 + \delta$ to node g . It is easy to infer that the network satisfies the triangle inequality when $\delta \leq \frac{2}{3}$. Suppose that all the nodes in the network are candidate server locations and there is no limit on the number of server locations to select. Then, the optimal placement is to select all the nodes in $C_A \cup C_B$ as server locations. Under this placement, each client connects to the server co-located with it, so the total interaction path length is

$$T_O = \sum_{i=1}^6 \sum_{j \neq i} d(c_i, c_j) = 12 \cdot 1 + 18 \cdot \left(\frac{4}{3} + 3\delta\right) = 36 + 54\delta.$$

On the other hand, the first server location selected by the T-GREEDY algorithm is g . This is because placing a server at g leads to the total interaction path length of

$$D_1 = 12 \cdot \sum_{i=1}^6 d(c_i, g) = 72 + 72\delta,$$

whereas if a server is placed at a node in $C_A \cup C_B$, the total interaction path length is

$$12 \cdot \left(2 \cdot 1 + 3 \cdot \left(\frac{4}{3} + 3\delta\right)\right) = 72 + 108\delta > D_1.$$

In the second iteration of the T-GREEDY algorithm, if a new server location c is further selected from $C_A \cup C_B$, all the three clients in c 's group would connect to the server at c and the other three clients remain connecting to the server at g . Therefore, the total interaction path length becomes

$$D_2 = 12 \cdot (2 \cdot 1 + 3 \cdot (1 + \delta)) + 18 \cdot (1 + \delta) = 78 + 54\delta.$$

When $\delta < \frac{1}{3}$, we have $D_2 > D_1$. Thus, the T-GREEDY algorithm does not add any new server location and terminates at the second iteration. As a result, the final set of server locations selected is $\{g\}$. When δ approaches 0, the ratio between the interaction path lengths produced by the T-GREEDY algorithm and the optimal server placement is

$$\lim_{\delta \rightarrow 0} \frac{D_1}{T_O} = \lim_{\delta \rightarrow 0} \frac{72 + 72\delta}{36 + 54\delta} = 2.$$

3.6 T-Better Server Placement

Somewhat surprisingly, we can further improve the approximation ratio by simply combining the NEAREST server placement and the T-GREEDY algorithm. The algorithm, which shall be called T-BETTER, works as follows. Given an instance of the server provisioning problem, suppose that S_N is the NEAREST server placement, and S_G is the server placement produced by the T-GREEDY heuristic. Denote by T_N and T_G respectively the total interaction path lengths under server placements S_N and S_G . The T-BETTER algorithm chooses the “better” placement between S_N and S_G . That is, it selects the candidate server locations in S_N if $T_N < T_G$, and selects the candidate server locations in S_G otherwise. As a result, the total interaction path length T_B produced by the T-BETTER algorithm is $\min\{T_N, T_G\}$. It is easy to infer that the T-BETTER algorithm has the same time complexity as the T-GREEDY heuristic because the T-GREEDY heuristic has a higher time complexity than the NEAREST server placement.

Theorem 5. *For networks with the triangle inequality, the T-BETTER server placement has an approximation ratio of $\frac{5}{3}$ for the server provisioning problem for discrete DIAs.*

Proof. Let T_B , T_N and T_G be the total interaction path lengths produced by the T-BETTER, NEAREST and T-GREEDY server placements respectively. It follows from (3.6) in Theorem 3 that

$$T_N \leq 6n \sum_{i=1}^n d(c_i, o_i) + \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j).$$

It also follows from (3.9) in Theorem 4 that,

$$T_G \leq 2n \sum_{i=1}^n d(c_i, o_i) + 2 \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j).$$

Thus,

$$T_B = \min\{T_N, T_G\}$$

$$\begin{aligned}
 &\leq \frac{1}{3} \cdot (T_N + 2T_G) \\
 &\leq \frac{1}{3} \cdot \left(6n \sum_{i=1}^n d(c_i, o_i) + \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j) \right. \\
 &\quad \left. + 4n \sum_{i=1}^n d(c_i, o_i) + 4 \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j) \right) \\
 &= \frac{1}{3} \cdot \left(10n \sum_{i=1}^n d(c_i, o_i) + 5 \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j) \right) \\
 &= \frac{5}{3} \cdot \left(2n \sum_{i=1}^n d(c_i, o_i) + \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j) \right).
 \end{aligned}$$

Note that the total interaction path length T_O under an optimal server placement is given by

$$T_O = 2n \sum_{i=1}^n d(c_i, o_i) + \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j).$$

Therefore,

$$T_B \leq \frac{5}{3} \cdot T_O.$$

Hence, the theorem is proven. $\square$

3.7 Analysis of Server Placement in Networks with α -Triangle Inequality

While many works in the field of network coordinate systems [58, 59] have assumed that the triangle inequality holds in the Internet, several other works have argued that triangle inequality violations are not uncommon due to the restrictions of the Internet structure and the routing policies of ISPs [56, 60]. Thus, besides networks with the triangle inequality, we also concern about the performance of server placements in networks with triangle inequality violations. To this end, we relax the triangle inequality constraint by introducing an additional parameter α and define the α -triangle inequality as follows.

α -Triangle Inequality. *Given a constant α ($\alpha \geq 1$), a network is said to satisfy the α -triangle inequality if for any three different nodes u , v and w in the network, it holds that*

$$d(u, v) \leq \alpha \cdot (d(u, w) + d(w, v)).$$

Note that for any two different nodes u and v in the network, it is obvious that $d(u, v) \leq \alpha \cdot d(u, v) = \alpha \cdot (d(u, v) + d(v, v))$. Therefore, if a network satisfies the α -triangle inequality, any nodes u , v and w in the network fulfills $d(u, v) \leq \alpha \cdot (d(u, w) + d(w, v))$.

The α -triangle inequality is generic because for any network, a value of α can always be found for the network latencies to satisfy the α -triangle inequality. Suppose that V is the set of nodes in a network. Then, α can be set at

$$\max \left\{ 1, \max_{\substack{u, v, w \in V \\ u \neq v \neq w}} \frac{d(u, v)}{d(u, w) + d(w, v)} \right\}.$$

In essence, the parameter α characterizes the extent of triangle inequality violations. The larger the value of α , the more severe the triangle inequality violations. When $\alpha = 1$, the α -triangle inequality degenerates to the original triangle inequality. Given a constant $\alpha > 1$, the networks satisfying the α -triangle inequality but not the original 1-triangle inequality constitute a *subset* of all the networks with triangle inequality violations. Although Theorem 1 has given a non-approximability result for networks with triangle inequality violations, it is possible to approximate the server provisioning problem for networks satisfying the α -triangle inequality.

Next, we analyze the approximability of the four server placement algorithms described earlier under the α -triangle inequality.

3.7.1 k -Median Server Placement

We start with the k -median server placement. When a limit k is set on the number of server locations to select in the server provisioning problem, we have the following

approximability result of the k -median server placement.

Theorem 6. *For networks satisfying the α -triangle inequality ($\alpha \geq 1$), the k -median server placement has an approximation ratio of $\max\{\frac{1}{2}\alpha^3 + \frac{3}{2}\alpha^2 + 1, \frac{2}{3}\alpha^3 + \alpha^2 + \frac{2}{3}\}$ for the server provisioning problem for discrete DIAs with a limit k on the number of server locations to select.*

Proof. Consider a network satisfying the α -triangle inequality ($\alpha \geq 1$). Assume that there are n clients $c_1, c_2, \dots, c_n$ in the network. Let Z be the set of candidate server locations in the network. Suppose that the set of optimal server locations for our server provisioning problem is $S_O \subseteq Z$. We denote by $o_i \in S_O$ the nearest server of each client c_i ($1 \leq i \leq n$) under placement S_O . Suppose that the k -median server placement is $S_M \subseteq Z$. We denote by $m_i \in S_M$ the nearest server of each client c_i under placement S_M .

If we apply the k -median server placement S_M to our server provisioning problem, its total interaction path length is given by

$$\begin{aligned} T_M &= \sum_{i=1}^n \sum_{j=1}^n (d(c_i, m_i) + d(m_i, m_j) + d(m_j, c_j)) \\ &= 2n \sum_{i=1}^n d(c_i, m_i) + \sum_{i=1}^n \sum_{j=1}^n d(m_i, m_j). \end{aligned}$$

By the α -triangle inequality, for any two clients c_i and c_j , we have

$$\begin{aligned} d(m_i, m_j) &\leq \alpha(d(m_i, o_i) + d(o_i, m_j)) \\ &\leq \alpha(\alpha(d(m_i, c_i) + d(c_i, o_i)) + d(o_i, m_j)). \end{aligned} \tag{3.10}$$

By the α -triangle inequality, we also have

$$\begin{aligned} d(o_i, m_j) &\leq \alpha(d(o_i, o_j) + d(o_j, m_j)) \\ &\leq \alpha(d(o_i, o_j) + \alpha(d(o_j, c_j) + d(c_j, m_j))), \end{aligned}$$

and

$$\begin{aligned} d(o_i, m_j) &\leq \alpha(d(o_i, c_j) + d(c_j, m_j)) \\ &\leq \alpha(\alpha(d(o_i, o_j) + d(o_j, c_j)) + d(c_j, m_j)). \end{aligned}$$

Therefore,

$$d(o_i, m_j) \leq \alpha^2 \cdot d(o_j, c_j) + \alpha \cdot \min \left\{ d(o_i, o_j) + \alpha \cdot d(c_j, m_j), \right. \\ \left. \alpha \cdot d(o_i, o_j) + d(c_j, m_j) \right\}. \quad (3.11)$$

It follows from (3.10) and (3.11) that,

$$\begin{aligned} T_M &\leq 2n \sum_{i=1}^n d(c_i, m_i) + \alpha \sum_{i=1}^n \sum_{j=1}^n \left(\alpha(d(m_i, c_i) + d(c_i, o_i)) \right. \\ &\quad \left. + d(o_i, m_j) \right) \\ &\leq 2n \sum_{i=1}^n d(c_i, m_i) \\ &\quad + \alpha \sum_{i=1}^n \sum_{j=1}^n \left(\alpha \cdot d(m_i, c_i) + \alpha \cdot d(c_i, o_i) \right. \\ &\quad \left. + \alpha^2 \cdot d(o_j, c_j) + \alpha \cdot \min \left\{ d(o_i, o_j) + \alpha \cdot d(c_j, m_j), \right. \right. \\ &\quad \left. \left. \alpha \cdot d(o_i, o_j) + d(c_j, m_j) \right\} \right) \\ &= 2n \sum_{i=1}^n d(c_i, m_i) + \alpha^2 n \sum_{i=1}^n d(m_i, c_i) \\ &\quad + \alpha^2 n \sum_{i=1}^n d(c_i, o_i) + \alpha^3 n \sum_{j=1}^n d(c_j, o_j) \\ &\quad + \alpha^2 \cdot \sum_{i=1}^n \sum_{j=1}^n \min \left\{ d(o_i, o_j) + \alpha \cdot d(c_j, m_j), \right. \\ &\quad \left. \alpha \cdot d(o_i, o_j) + d(c_j, m_j) \right\} \\ &\leq (2n + \alpha^2 n) \sum_{i=1}^n d(c_i, m_i) + (\alpha^2 n + \alpha^3 n) \sum_{i=1}^n d(c_i, o_i) \\ &\quad + \alpha^2 \cdot \min \left\{ \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j) + \alpha n \sum_{j=1}^n d(c_j, m_j), \right. \\ &\quad \left. \alpha \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j) + n \sum_{i=1}^n d(c_i, m_i) \right\}. \end{aligned} \quad (3.12)$$

The definition of the k -median placement implies that $\sum_{i=1}^n d(c_i, m_i)$ is the minimum achievable total distance from all the clients to their nearest servers among all possible placements of up to k servers. Thus, we have

$$\sum_{i=1}^n d(c_i, m_i) \leq \sum_{i=1}^n d(c_i, o_i).$$

As a result,

$$\begin{aligned}
 T_M \leq & n(2 + 2\alpha^2 + \alpha^3) \sum_{i=1}^n d(c_i, o_i) \\
 & + \alpha^2 \cdot \min \left\{ \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j) + \alpha n \sum_{i=1}^n d(c_i, o_i), \right. \\
 & \left. \alpha \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j) + n \sum_{i=1}^n d(c_i, o_i) \right\}.
 \end{aligned}$$

We define a new variable

$$x = \frac{\sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j)}{n \cdot \sum_{i=1}^n d(c_i, o_i)}.$$

Then, we have

$$\begin{aligned}
 T_M \leq & n \sum_{i=1}^n d(c_i, o_i) \cdot \left(2 + 2\alpha^2 + \alpha^3 \right. \\
 & \left. + \alpha^2 \cdot \min \left\{ x + \alpha, \alpha x + 1 \right\} \right). \tag{3.13}
 \end{aligned}$$

If $x \geq 1$, since $\alpha \geq 1$, we have $x - 1 \leq \alpha(x - 1)$. It follows that $\min\{x + \alpha, \alpha x + 1\} = x + \alpha$. Thus,

$$T_M \leq n \sum_{i=1}^n d(c_i, o_i) \cdot (2 + 2\alpha^2 + \alpha^3 + \alpha^2(x + \alpha)).$$

Note that the total interaction path length under the optimal server placement S_O is

$$\begin{aligned}
 T_O &= 2n \sum_{i=1}^n d(c_i, o_i) + \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j) \\
 &= n \sum_{i=1}^n d(c_i, o_i) \cdot (2 + x). \tag{3.14}
 \end{aligned}$$

Therefore, the largest possible ratio between T_M and T_O is

$$\begin{aligned}
 r(x \geq 1) &= \max_{x \geq 1} \frac{2 + 2\alpha^2 + \alpha^3 + \alpha^2(x + \alpha)}{2 + x} \\
 &= \max_{x \geq 1} \left(\alpha^2 + \frac{2\alpha^3 + 2}{2 + x} \right) \\
 &= \alpha^2 + \frac{2\alpha^3 + 2}{3}
 \end{aligned}$$

$$= \frac{2}{3}\alpha^3 + \alpha^2 + \frac{2}{3}. \quad (3.15)$$

Similarly, if $0 \leq x \leq 1$, since $\alpha \geq 1$, we have $x - 1 \geq \alpha(x - 1)$ and thus $\min\{x + \alpha, \alpha x + 1\} = \alpha x + 1$. In this case, the largest possible ratio between T_M and T_O is

$$\begin{aligned} r(0 \leq x \leq 1) &= \max_{0 \leq x \leq 1} \frac{2 + 2\alpha^2 + \alpha^3 + \alpha^2(\alpha x + 1)}{2 + x} \\ &= \max_{0 \leq x \leq 1} \left(\alpha^3 + \frac{2 + 3\alpha^2 - \alpha^3}{2 + x} \right). \end{aligned}$$

If $2 + 3\alpha^2 - \alpha^3 \geq 0$, i.e., $1 \leq \alpha \leq 1 + \sqrt[3]{2 + \sqrt{3}} + \sqrt[3]{2 - \sqrt{3}}$, we have

$$\begin{aligned} &\max_{0 \leq x \leq 1} \left(\alpha^3 + \frac{2 + 3\alpha^2 - \alpha^3}{2 + x} \right) \\ &= \alpha^3 + \frac{2 + 3\alpha^2 - \alpha^3}{2} \\ &= \frac{2 + 3\alpha^2 + \alpha^3}{2}. \end{aligned}$$

If $2 + 3\alpha^2 - \alpha^3 < 0$, i.e., $\alpha > 1 + \sqrt[3]{2 + \sqrt{3}} + \sqrt[3]{2 - \sqrt{3}}$, we have

$$\begin{aligned} &\max_{0 \leq x \leq 1} \left(\alpha^3 + \frac{2 + 3\alpha^2 - \alpha^3}{2 + x} \right) \\ &= \alpha^3 + \frac{2 + 3\alpha^2 - \alpha^3}{3} \\ &= \frac{2 + 3\alpha^2 + 2\alpha^3}{3}. \end{aligned}$$

Thus,

$$r(0 \leq x \leq 1) = \begin{cases} \frac{1}{2}\alpha^3 + \frac{3}{2}\alpha^2 + 1 & \text{if } 2 + 3\alpha^2 \geq \alpha^3, \\ \frac{2}{3}\alpha^3 + \alpha^2 + \frac{2}{3} & \text{if } 2 + 3\alpha^2 < \alpha^3, \end{cases}$$

which can actually be rewritten as

$$r(0 \leq x \leq 1) = \max \left\{ \frac{1}{2}\alpha^3 + \frac{3}{2}\alpha^2 + 1, \frac{2}{3}\alpha^3 + \alpha^2 + \frac{2}{3} \right\}. \quad (3.16)$$

Combining (3.15) and (3.16), if $\alpha \geq 1$, the approximation ratio of the k -median placement for our server provisioning problem is given by

$$\max \left\{ \frac{1}{2}\alpha^3 + \frac{3}{2}\alpha^2 + 1, \frac{2}{3}\alpha^3 + \alpha^2 + \frac{2}{3} \right\}.$$

Hence, the theorem is proven. $\square$

Figure 3.5: The approximation ratio of the k -median placement is tight: tight example for $\frac{1}{2}\alpha^3 + \frac{3}{2}\alpha^2 + 1$.

The proof of Theorem 6 gives rise to the following result for the server provisioning problem when there is no limit on the number of server locations to select.

Corollary 2. *For each instance of the server provisioning problem for discrete DIAs on a network satisfying the α -triangle inequality, if an optimal server placement selects k server locations, then the k -median server placement produces a total interaction path length within $\max\{\frac{1}{2}\alpha^3 + \frac{3}{2}\alpha^2 + 1, \frac{2}{3}\alpha^3 + \alpha^2 + \frac{2}{3}\}$ times of that in the optimal server placement.*

The approximation ratio derived in Theorem 6 is tight. To demonstrate, we present an example in which the ratio between the total interaction path lengths of the k -median placement and optimal placement can be made arbitrarily close to $\frac{1}{2}\alpha^3 + \frac{3}{2}\alpha^2 + 1$, and another example in which the ratio can be made arbitrarily close to $\frac{2}{3}\alpha^3 + \alpha^2 + \frac{2}{3}$. Both examples are constructed based on a network topology consisting

Table 3.2: Network latency matrix for tight example of $\frac{1}{2}\alpha^3 + \frac{3}{2}\alpha^2 + 1$.

$\forall i, j$ ($i \neq j$)	c_i	m_i	o_i
c_i	0	1	$1 + \varepsilon$
m_i	1	0	2α
o_i	$1 + \varepsilon$	2α	0
c_j	$\alpha^2 + \alpha$	$2\alpha^2 + \alpha$	$\alpha(1 + 2\varepsilon)$
m_j	$2\alpha^2 + \alpha$	$\alpha^3 + 3\alpha^2$	$\alpha^2 + \alpha$
o_j	$\alpha(1 + 2\varepsilon)$	$\alpha^2 + \alpha$	ε

of three disjoint subsets of nodes $V_C = \{c_1, c_2, \dots, c_n\}$, $V_M = \{m_1, m_2, \dots, m_n\}$ and $V_O = \{o_1, o_2, \dots, o_n\}$. Figure 3.5 illustrates the first example. Due to space limitations, only the network latencies between some node pairs are shown in the figure. Table 3.2 lists the network latencies between all pairs of nodes, where $\alpha \geq 1$ and $0 < \varepsilon < 1$. It is easy to verify that the network latencies satisfy the α -triangle inequality. Suppose that a client is located at each node in V_C , and all the nodes in $V_M \cup V_O$ are candidate server locations. Then, the optimal solution to our server provisioning problem is to place servers at all the n nodes in V_O . In this case, each client c_i connects to the server at o_i , and this gives the minimum total interaction path length of

$$\begin{aligned}
 & 2n \sum_{i=1}^n d(c_i, o_i) + \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j) \\
 &= 2n^2(1 + \varepsilon) + n(n-1)\varepsilon.
 \end{aligned}$$

On the other hand, the n -median placement is to place servers at all the n nodes in V_M . Under such placement, the nearest server of each client c_i is at m_i , so the total latency from the clients to their nearest servers is minimized at n . In this case, the total interaction path length is given by

$$\begin{aligned}
 & 2n \sum_{i=1}^n d(c_i, m_i) + \sum_{i=1}^n \sum_{j=1}^n d(m_i, m_j) \\
 &= 2n^2 + n(n-1)(\alpha^3 + 3\alpha^2).
 \end{aligned}$$

Therefore, the ratio between the results of the n -median and optimal placements is $\frac{2n^2 + n(n-1)(\alpha^3 + 3\alpha^2)}{2n^2(1 + \varepsilon) + n(n-1)\varepsilon}$. As the number of clients n goes toward infinity and ε approaches

Figure 3.6: The approximation ratio of the k -median placement is tight: tight example for $\frac{2}{3}\alpha^3 + \alpha^2 + \frac{2}{3}$.

0, this ratio approaches

$$\begin{aligned}
 & \lim_{n \rightarrow \infty} \lim_{\varepsilon \rightarrow 0} \frac{2n^2 + n(n-1)(\alpha^3 + 3\alpha^2)}{2n^2(1+\varepsilon) + n(n-1)\varepsilon} \\
 &= \lim_{n \rightarrow \infty} \frac{2n^2 + n(n-1)(\alpha^3 + 3\alpha^2)}{2n^2} \\
 &= \frac{1}{2}\alpha^3 + \frac{3}{2}\alpha^2 + 1.
 \end{aligned}$$

Similarly, Figure 3.6 illustrates the second example and Table 3.3 shows the network latencies between all pairs of nodes in this example, where $\alpha \geq 1$ and $0 < \varepsilon < 1$. It can be checked that the network latencies satisfy the α -triangle inequality. Again, suppose that a client is located at each node in V_C , and all the nodes in $V_M \cup V_O$ are candidate server locations. The optimal placement for our server provisioning problem is to place a server at each node in V_O so that each client c_i connects to the

Table 3.3: Network latency matrix for tight example of $\frac{2}{3}\alpha^3 + \alpha^2 + \frac{2}{3}$.

$\forall i, j$ ($i \neq j$)	c_i	m_i	o_i
c_i	0	1	$1 + \varepsilon$
m_i	1	0	2α
o_i	$1 + \varepsilon$	2α	0
c_j	$2\alpha^2$	$2\alpha^2 + 3\alpha$	2α
m_j	$2\alpha^2 + 3\alpha$	$2\alpha^3 + 3\alpha^2$	$2\alpha^2 + \alpha$
o_j	2α	$2\alpha^2 + \alpha$	1

server at o_i . This gives the minimum total interaction path length of

$$2n \sum_{i=1}^n d(c_i, o_i) + \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j) = 2n^2(1 + \varepsilon) + n(n-1).$$

Meanwhile, the n -median placement is to select all the n nodes in V_M as server locations. In this case, each client c_i connects to its nearest server at m_i . Thus, the total interaction path length is

$$\begin{aligned} & 2n \sum_{i=1}^n d(c_i, m_i) + \sum_{i=1}^n \sum_{j=1}^n d(m_i, m_j) \\ &= 2n^2 + n(n-1)(2\alpha^3 + 3\alpha^2). \end{aligned}$$

Therefore, the ratio between the results of the n -median and optimal placements can be made arbitrarily close to

$$\begin{aligned} & \lim_{n \rightarrow \infty} \lim_{\varepsilon \rightarrow 0} \frac{2n^2 + n(n-1)(2\alpha^3 + 3\alpha^2)}{2n^2(1 + \varepsilon) + n(n-1)} \\ &= \lim_{n \rightarrow \infty} \frac{2n^2 + n(n-1)(2\alpha^3 + 3\alpha^2)}{2n^2 + n(n-1)} \\ &= \frac{2}{3}\alpha^3 + \alpha^2 + \frac{2}{3}, \end{aligned}$$

as n goes toward infinity and ε approaches 0.

3.7.2 Nearest Server Placement

Similar to the case of networks with the triangle inequality, the NEAREST server placement proposed in Section 3.4 has the same approximation ratio as that of the k -median placement for networks with the α -triangle inequality.

Theorem 7. *For networks satisfying the α -triangle inequality ($\alpha \geq 1$), the NEAREST server placement has an approximation ratio of $\max\{\frac{1}{2}\alpha^3 + \frac{3}{2}\alpha^2 + 1, \frac{2}{3}\alpha^3 + \alpha^2 + \frac{2}{3}\}$ for the server provisioning problem for discrete DIAs.*

Proof. Suppose that $C = \{c_1, c_2, \dots, c_n\}$ is the set of clients, and Z is the set of candidate server locations in a network with the α -triangle inequality. Let $S_N \subseteq Z$ be the set of server locations selected by the NEAREST server placement, and n_i be the nearest server of each client c_i ($1 \leq i \leq n$) in S_N . Then, the total interaction path length under placement S_N is given by

$$\begin{aligned} T_N &= \sum_{i=1}^n \sum_{j=1}^n (d(c_i, n_i) + d(n_i, n_j) + d(n_j, c_j)) \\ &= 2n \sum_{i=1}^n d(c_i, n_i) + \sum_{i=1}^n \sum_{j=1}^n d(n_i, n_j). \end{aligned}$$

Similar to the derivation of (3.12) in Theorem 6, we have

$$\begin{aligned} T_N &\leq (2n + \alpha^2 n) \sum_{i=1}^n d(c_i, n_i) + (\alpha^2 n + \alpha^3 n) \sum_{i=1}^n d(c_i, o_i) \\ &\quad + \alpha^2 \cdot \min \left\{ \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j) + \alpha n \sum_{j=1}^n d(c_j, n_j), \right. \\ &\quad \left. \alpha \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j) + n \sum_{i=1}^n d(c_i, n_i) \right\}, \end{aligned} \tag{3.17}$$

where o_i is the nearest server of each client c_i under an optimal server placement S_O .

For each client c_i , since n_i is the nearest candidate server location to c_i among all the candidate server locations in Z , we have $d(c_i, n_i) \leq d(c_i, o_i)$. Thus,

$$\begin{aligned} T_N &\leq (2n + \alpha^2 n) \sum_{i=1}^n d(c_i, o_i) + (\alpha^2 n + \alpha^3 n) \sum_{i=1}^n d(c_i, o_i) \\ &\quad + \alpha^2 \cdot \min \left\{ \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j) + \alpha n \sum_{i=1}^n d(c_i, o_i), \right. \\ &\quad \left. \alpha \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j) + n \sum_{i=1}^n d(c_i, o_i) \right\} \\ &= n \sum_{i=1}^n d(c_i, o_i) \cdot \left(2 + 2\alpha^2 + \alpha^3 \right) \end{aligned}$$

$$+\alpha^2 \cdot \min \left\{ x + \alpha, \alpha x + 1 \right\}, \quad (3.18)$$

where

$$x = \frac{\sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j)}{n \cdot \sum_{i=1}^n d(c_i, o_i)}.$$

By considering the ratio between T_N and the optimal total interaction path length T_O as a function of α and x , we can follow the same mathematical reasoning as in Theorem 6 and obtain that the approximation ratio of the NEAREST server placement is $\max\{\frac{1}{2}\alpha^3 + \frac{3}{2}\alpha^2 + 1, \frac{2}{3}\alpha^3 + \alpha^2 + \frac{2}{3}\}$ for the server provisioning problem.

Hence, the theorem is proven. $\square$

The tightness of the approximation ratio derived in Theorem 7 can also be shown by the examples given in Figures 3.5 and 3.6. In these examples, the NEAREST server placement is given by all the n nodes in V_M , which is the same as the n -median server placement. Thus, these examples are also applicable to the NEAREST server placement.

3.7.3 T-Greedy Server Placement

In networks with the α -triangle inequality, the T-GREEDY algorithm proposed in Section 3.5 produces a total interaction path length within 2α times of that in an optimal server placement, irrespective of whether the choices of server locations are restricted and the cap on the number of server locations to select (if any).

Theorem 8. *For networks satisfying the α -triangle inequality ($\alpha \geq 1$), the T-GREEDY server placement has an approximation ratio of 2α for the server provisioning problem for discrete DIAs.*

Proof. Suppose that $C = \{c_1, c_2, \dots, c_n\}$ is the set of clients. Let s^* be the first server location selected by the T-GREEDY algorithm. It follows from (3.8) in Theorem 4

that

$$2n \cdot \sum_{i=1}^n d(c_i, s^*) \leq 2 \cdot \sum_{i=1}^n \left(d(c_i, o_i) + \sum_{j \neq i} d(c_i, o_j) \right),$$

where o_i is the nearest server of each client c_i under an optimal server placement S_O .

By the α -triangle inequality, for any client c_i and any o_j ($j \neq i$), we have

$$d(c_i, o_j) \leq \alpha \cdot (d(c_i, o_i) + d(o_i, o_j)).$$

Therefore,

$$\begin{aligned} & 2n \cdot \sum_{i=1}^n d(c_i, s^*) \\ & \leq 2 \cdot \sum_{i=1}^n \left(d(c_i, o_i) + \alpha \cdot \sum_{j \neq i} (d(c_i, o_i) + d(o_i, o_j)) \right) \\ & \leq 2 \cdot \sum_{i=1}^n \left(n\alpha \cdot d(c_i, o_i) + \alpha \cdot \sum_{j \neq i} d(o_i, o_j) \right) \\ & = 2 \cdot \sum_{i=1}^n \left(n\alpha \cdot d(c_i, o_i) + \alpha \cdot \sum_{j=1}^n d(o_i, o_j) \right) \\ & = 2\alpha \cdot n \cdot \sum_{i=1}^n d(c_i, o_i) + 2\alpha \cdot \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j) \tag{3.19} \\ & = 2\alpha \cdot \left(n \sum_{i=1}^n d(c_i, o_i) + \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j) \right) \\ & \leq 2\alpha \cdot \left(2n \sum_{i=1}^n d(c_i, o_i) + \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j) \right) \\ & = 2\alpha \cdot T_O, \end{aligned}$$

where T_O is the total interaction path length under the optimal server placement S_O .

This implies that the total interaction path length on selecting the first server location in the T-GREEDY algorithm is within 2α times of that in an optimal server placement. Since the T-GREEDY algorithm adds a new server location in each subsequent iteration only if it reduces the total interaction path length, the total interaction path length T_G eventually produced by T-GREEDY must also be within 2α times of that in an optimal server placement, i.e.,

$$T_G \leq 2n \sum_{i=1}^n d(c_i, s^*) \leq 2\alpha \cdot T_O.$$

Figure 3.7: The approximation ratio 2α of the T-GREEDY algorithm is tight.

Hence, the theorem is proven. $\square$

The approximation ratio 2α of the T-GREEDY algorithm is tight. Figure 3.7 presents a tight example in which the network contains two node groups $G_A = \{c_1, c_2, c_3, s_1, s_2, s_3\}$ and $G_B = \{c_4, c_5, c_6, s_4, s_5, s_6\}$, and an additional node g . Each node c_i ($1 \leq i \leq 6$) has latency ε to node s_i , latency α to the rest nodes in the same group, and latency $\alpha(\frac{4}{3} + 3\delta)$ to all the nodes in the other group, where $\delta, \varepsilon > 0$. Figure 3.7 illustrates the latencies between c_3 and the other nodes. In addition, the latency between two nodes s_i and s_j ($i \neq j$) is 1 if they are in the same group, and is $\frac{4}{3} + 3\delta$ otherwise. All the nodes in G_A and G_B have latency $\alpha(1 + \delta)$ to node g . It is easy to infer that the network latencies satisfy the α -triangle inequality when $\delta \leq \frac{2}{3}$ and $\varepsilon \leq \alpha(1 + \alpha)$. Suppose that a client is located at each node c_i ($1 \leq i \leq 6$), nodes $g, s_1, s_2, \dots, s_6$ are candidate server locations, and there is no limit on the number of server locations to select. Then, the optimal placement is to select nodes $s_1, s_2, \dots, s_6$ as server locations. Under such server placement, each client c_i connects to the server at s_i , so the total interaction path length is

$$T_O = 12 \cdot \sum_{i=1}^6 d(c_i, s_i) + \sum_{i=1}^6 \sum_{j \neq i}^6 d(s_i, s_j)$$

$$\begin{aligned}
 &= 12 \cdot 6\varepsilon + 12 \cdot 1 + 18 \cdot \left(\frac{4}{3} + 3\delta\right) \\
 &= 36 + 54\delta + 72\varepsilon.
 \end{aligned}$$

On the other hand, the first server location selected by the T-GREEDY algorithm is g . This is because placing a server at g leads to the total interaction path length of

$$\begin{aligned}
 D_1 &= 12 \cdot \sum_{i=1}^6 d(c_i, g) \\
 &= 12 \cdot 6 \cdot \alpha(1 + \delta) \\
 &= 72\alpha + 72\alpha\delta,
 \end{aligned}$$

whereas if a server is placed at any node s_i , the total interaction path length is

$$\begin{aligned}
 12 \cdot \left(\varepsilon + 2 \cdot \alpha + 3 \cdot \alpha\left(\frac{4}{3} + 3\delta\right)\right) &= 72\alpha + 108\alpha\delta + 12\varepsilon \\
 &> D_1.
 \end{aligned}$$

In the second iteration of the T-GREEDY algorithm, if a new server location s is further selected from $\{s_1, s_2, \dots, s_6\}$, all the three clients in s 's group would connect to the server at s and the other three clients remain connecting to the server at g . Therefore, the total interaction path length becomes

$$\begin{aligned}
 D_2 &= 12 \cdot (\varepsilon + 2 \cdot \alpha + 3 \cdot \alpha(1 + \delta)) + 18 \cdot \alpha(1 + \delta) \\
 &= 78\alpha + 54\alpha\delta + 12\varepsilon.
 \end{aligned}$$

When $\delta < \frac{1}{3} + \frac{2\alpha}{3}\varepsilon$, we have $D_2 > D_1$. Thus, the T-GREEDY algorithm does not add any new server location and terminates at the second iteration. As a result, the final set of server locations selected is $\{g\}$. When both δ and ε approach 0, the ratio between the total interaction path lengths produced by the T-GREEDY algorithm and the optimal server placement is

$$\begin{aligned}
 &\lim_{\delta \rightarrow 0} \lim_{\varepsilon \rightarrow 0} \frac{D_1}{T_O} \\
 &= \lim_{\delta \rightarrow 0} \lim_{\varepsilon \rightarrow 0} \frac{72\alpha + 72\alpha\delta}{36 + 54\delta + 72\varepsilon}
 \end{aligned}$$

$$= 2\alpha.$$

For any $\alpha \geq 1$, we have

$$\begin{aligned} 2\alpha &\leq \frac{1}{2}\alpha^3 + \frac{3}{2}\alpha^2 \\ &\leq \max \left\{ \frac{1}{2}\alpha^3 + \frac{3}{2}\alpha^2 + 1, \frac{2}{3}\alpha^3 + \alpha^2 + \frac{2}{3} \right\}. \end{aligned}$$

Therefore, the approximation ratio of the T-GREEDY algorithm (Theorem 8) is much lower than those of the k -median server placement (Theorem 6) and the NEAREST server placement (Theorem 7). Asymptotically, the approximation ratio of T-GREEDY grows just linearly with α , whereas the approximation ratios of the k -median and NEAREST server placements grow cubically with α . This indicates that the T-GREEDY algorithm is far more resilient to the extent of triangle inequality violations than the k -median and NEAREST server placements.

3.7.4 T-Better Server Placement

The following theorem presents the approximability result of the T-BETTER algorithm proposed in Section 3.6 under the α -triangle inequality.

Theorem 9. *For networks satisfying the α -triangle inequality ($\alpha \geq 1$), the T-BETTER server placement has an approximation ratio of $\min \left\{ \frac{2\alpha^4 + \alpha^3 + 2\alpha}{\alpha^3 + \alpha + 1}, 2\alpha \right\}$ for the server provisioning problem for discrete DIAs.*

Proof. Let S_O be a set of optimal server locations for the server provisioning problem, and o_i be the nearest server of each client c_i under placement S_O . Again, define

$$x = \frac{\sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j)}{n \cdot \sum_{i=1}^n d(c_i, o_i)}.$$

According to (3.19) in Theorem 8, the total interaction path length T_G produced by the T-GREEDY heuristic is bounded by

$$T_G \leq 2\alpha \cdot n \cdot \sum_{i=1}^n d(c_i, o_i) + 2\alpha \cdot \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j)$$

$$= (2\alpha + 2\alpha x) \cdot n \cdot \sum_{i=1}^n d(c_i, o_i). \quad (3.20)$$

Denote by T_N the total interaction path length produced by the NEAREST server placement. According to (3.18) in Theorem 7, we have

$$\begin{aligned} T_N &\leq n \sum_{i=1}^n d(c_i, o_i) \cdot \left(2 + 2\alpha^2 + \alpha^3 \right. \\ &\quad \left. + \alpha^2 \cdot \min \left\{ x + \alpha, \alpha x + 1 \right\} \right) \\ &= \min \left\{ 2 + 2\alpha^2 + 2\alpha^3 + \alpha^2 x, 2 + 3\alpha^2 + \alpha^3 + \alpha^3 x \right\} \\ &\quad \cdot n \cdot \sum_{i=1}^n d(c_i, o_i). \end{aligned} \quad (3.21)$$

Recall that the T-BETTER algorithm chooses the better solution produced by the NEAREST server placement and the T-GREEDY heuristic. Therefore, the total interaction path length T_B under the T-BETTER server placement is given by

$$T_B = \min\{T_N, T_G\}.$$

It follows from (3.20) and (3.21) that

$$\begin{aligned} T_B &\leq \min \left\{ 2\alpha + 2\alpha x, 2 + 2\alpha^2 + 2\alpha^3 + \alpha^2 x, \right. \\ &\quad \left. 2 + 3\alpha^2 + \alpha^3 + \alpha^3 x \right\} \cdot n \cdot \sum_{i=1}^n d(c_i, o_i). \end{aligned}$$

When $x \leq 1$, we have

$$\begin{aligned} &(2 + 2\alpha^2 + 2\alpha^3 + \alpha^2 x) - (2 + 3\alpha^2 + \alpha^3 + \alpha^3 x) \\ &= (\alpha^3 - \alpha^2) \cdot (1 - x) \\ &\geq 0. \end{aligned}$$

In addition, $x \leq 1$ implies $x + 1 \leq 2$. Thus, we have

$$\begin{aligned} &(2 + 3\alpha^2 + \alpha^3 + \alpha^3 x) - (2\alpha + 2\alpha x) \\ &= 2 + 3\alpha^2 + (1 + x)(\alpha^3 - 2\alpha) \end{aligned}$$

$$\begin{aligned}
 &> (1+x) + (1+x)\alpha^2 + (1+x)(\alpha^3 - 2\alpha) \\
 &= (1+x)(1 + \alpha^2 + \alpha^3 - 2\alpha) \\
 &= (1+x)(\alpha^3 + (\alpha - 1)^2) \\
 &> 0.
 \end{aligned}$$

Since $2 + 2\alpha^2 + 2\alpha^3 + \alpha^2x \geq 2 + 3\alpha^2 + \alpha^3 + \alpha^3x > 2\alpha + 2\alpha x$, it follows that $\min \{2\alpha + 2\alpha x, 2 + 2\alpha^2 + 2\alpha^3 + \alpha^2x, 2 + 3\alpha^2 + \alpha^3 + \alpha^3x\} = 2\alpha + 2\alpha x$. Therefore, T-BETTER's total interaction path length T_B is bounded by $(2\alpha + 2\alpha x) \cdot n \cdot \sum_{i=1}^n d(c_i, o_i)$ when $x \leq 1$. Note that the total interaction path length under the optimal server placement S_O is

$$\begin{aligned}
 T_O &= \sum_{i=1}^n \sum_{j=1}^n (d(c_i, o_i) + d(o_i, o_j) + d(o_j, c_j)) \\
 &= 2n \sum_{i=1}^n d(c_i, o_i) + \sum_{i=1}^n \sum_{j=1}^n d(o_i, o_j) \\
 &= (2+x) \cdot n \cdot \sum_{i=1}^n d(c_i, o_i).
 \end{aligned}$$

Thus, when $x \leq 1$, the largest possible ratio between T_B and T_O is

$$\begin{aligned}
 r(x \leq 1) &= \max_{x \leq 1} \frac{2\alpha + 2\alpha x}{2+x} \\
 &= \max_{x \leq 1} \left(2\alpha - \frac{2\alpha}{2+x} \right) \\
 &= 2\alpha - \frac{2}{3}\alpha \\
 &= \frac{4}{3}\alpha.
 \end{aligned} \tag{3.22}$$

When $x \geq 1$, we have

$$\begin{aligned}
 &(2 + 2\alpha^2 + 2\alpha^3 + \alpha^2x) - (2 + 3\alpha^2 + \alpha^3 + \alpha^3x) \\
 &= (\alpha^3 - \alpha^2) \cdot (1 - x) \\
 &\leq 0,
 \end{aligned}$$

i.e.,

$$2 + 2\alpha^2 + 2\alpha^3 + \alpha^2x \leq 2 + 3\alpha^2 + \alpha^3 + \alpha^3x. \tag{3.23}$$

Now, consider

$$\begin{aligned} d &= (2 + 2\alpha^2 + 2\alpha^3 + \alpha^2 x) - (2\alpha + 2\alpha x) \\ &= 2 + 2\alpha^2 + 2\alpha^3 - 2\alpha + (\alpha^2 - 2\alpha)x. \end{aligned}$$

When $\alpha \geq 2$, we have $\alpha^2 - 2\alpha \geq 0$ and hence $d \geq 0$. This implies that

$$2 + 2\alpha^2 + 2\alpha^3 + \alpha^2 x \geq 2\alpha + 2\alpha x. \quad (3.24)$$

Combining (3.23) and (3.24), T-BETTER's total interaction path length T_B is bounded by $(2\alpha + 2\alpha x) \cdot n \cdot \sum_{i=1}^n d(c_i, o_i)$. Therefore, in the case of $x \geq 1$ and $\alpha \geq 2$, the largest possible ratio between T_B and T_O is

$$\begin{aligned} r(x \geq 1, \alpha \geq 2) &= \max_{x \geq 1} \frac{2\alpha + 2\alpha x}{2 + x} \\ &= \max_{x \geq 1} \left(2\alpha - \frac{2\alpha}{2 + x} \right) \\ &= 2\alpha. \end{aligned} \quad (3.25)$$

In case of $1 \leq x \leq \frac{2+2\alpha^2+2\alpha^3-2\alpha}{2\alpha-\alpha^2}$ and $\alpha \leq 2$, we have $(2\alpha - \alpha^2) \cdot x \leq 2 + 2\alpha^2 + 2\alpha^3 - 2\alpha$ and hence $d \geq 0$. Thus, similar to the above case, T-BETTER's total interaction path length T_B is bounded by $(2\alpha + 2\alpha x) \cdot n \cdot \sum_{i=1}^n d(c_i, o_i)$. So, the largest possible ratio between T_B and T_O is

$$\begin{aligned} &r\left(1 \leq x \leq \frac{2 + 2\alpha^2 + 2\alpha^3 - 2\alpha}{2\alpha - \alpha^2}, \alpha \leq 2\right) \\ &= \max_{1 \leq x \leq \frac{2+2\alpha^2+2\alpha^3-2\alpha}{2\alpha-\alpha^2}} \frac{2\alpha + 2\alpha x}{2 + x} \\ &= \max_{1 \leq x \leq \frac{2+2\alpha^2+2\alpha^3-2\alpha}{2\alpha-\alpha^2}} \left(2\alpha - \frac{2\alpha}{2 + x} \right) \\ &= 2\alpha - \frac{2\alpha}{2 + \frac{2+2\alpha^2+2\alpha^3-2\alpha}{2\alpha-\alpha^2}} \\ &= 2\alpha - \frac{2\alpha}{\frac{2+2\alpha^3+2\alpha}{2\alpha-\alpha^2}} \\ &= 2\alpha - \frac{\alpha \cdot (2\alpha - \alpha^2)}{\alpha^3 + \alpha + 1} \\ &= \frac{2\alpha^4 + \alpha^3 + 2\alpha}{\alpha^3 + \alpha + 1}. \end{aligned} \quad (3.26)$$

In the case of $x \geq \frac{2+2\alpha^2+2\alpha^3-2\alpha}{2\alpha-\alpha^2}$ and $\alpha \leq 2$, we have $(2\alpha - \alpha^2)x \geq 2 + 2\alpha^2 + 2\alpha^3 - 2\alpha$ and hence $d \leq 0$. This indicates that

$$2 + 2\alpha^2 + 2\alpha^3 + \alpha^2 x \leq 2\alpha + 2\alpha x. \quad (3.27)$$

Combining (3.23) and (3.27), T-BETTER's total interaction path length T_B is bounded by $(2 + 2\alpha^2 + 2\alpha^3 + \alpha^2 x) \cdot n \cdot \sum_{i=1}^n d(c_i, o_i)$. In this case, the largest possible ratio between T_B and T_O is

$$\begin{aligned} & r\left(x \geq \frac{2 + 2\alpha^2 + 2\alpha^3 - 2\alpha}{2\alpha - \alpha^2}, \alpha \leq 2\right) \\ &= \max_{x \geq \frac{2+2\alpha^2+2\alpha^3-2\alpha}{2\alpha-\alpha^2}} \frac{2 + 2\alpha^2 + 2\alpha^3 + \alpha^2 x}{2 + x} \\ &= \max_{x \geq \frac{2+2\alpha^2+2\alpha^3-2\alpha}{2\alpha-\alpha^2}} \left(\alpha^2 + \frac{2\alpha^3 + 2}{2 + x} \right) \\ &= \alpha^2 + \frac{2\alpha^3 + 2}{2 + \frac{2+2\alpha^2+2\alpha^3-2\alpha}{2\alpha-\alpha^2}} \\ &= \alpha^2 + \frac{2\alpha^3 + 2}{\frac{2+2\alpha^3+2\alpha}{2\alpha-\alpha^2}} \\ &= \alpha^2 + \frac{(\alpha^3 + 1) \cdot (2\alpha - \alpha^2)}{\alpha^3 + \alpha + 1} \\ &= \alpha^2 + \frac{2\alpha^4 + 2\alpha - \alpha^5 - \alpha^2}{\alpha^3 + \alpha + 1} \\ &= \frac{2\alpha^4 + \alpha^3 + 2\alpha}{\alpha^3 + \alpha + 1}. \end{aligned} \quad (3.28)$$

Combining (3.22), (3.26) and (3.28), when $\alpha \leq 2$, the largest possible ratio between T_B and T_O is given by

$$r(\alpha \leq 2) = \max \left\{ \frac{4}{3}\alpha, \frac{2\alpha^4 + \alpha^3 + 2\alpha}{\alpha^3 + \alpha + 1} \right\}.$$

Note that

$$\begin{aligned} & \frac{2\alpha^4 + \alpha^3 + 2\alpha}{\alpha^3 + \alpha + 1} - \frac{4}{3}\alpha \\ &= \frac{6\alpha^4 + 3\alpha^3 + 6\alpha - 4\alpha^4 - 4\alpha^2 - 4\alpha}{3\alpha^3 + 3\alpha + 3} \\ &= \frac{2\alpha^4 + 3\alpha^3 - 4\alpha^2 + 2\alpha}{3\alpha^3 + 3\alpha + 3} \end{aligned}$$

$$\begin{aligned}
 &= \frac{2\alpha^4 + \alpha^3 + 2\alpha \cdot (\alpha - 1)^2}{3\alpha^3 + 3\alpha + 3} \\
 &\geq 0.
 \end{aligned}$$

Therefore,

$$r(\alpha \leq 2) = \frac{2\alpha^4 + \alpha^3 + 2\alpha}{\alpha^3 + \alpha + 1}.$$

Combining (3.22) and (3.25), when $\alpha \geq 2$, the largest possible ratio between T_B and T_O is given by

$$\begin{aligned}
 r(\alpha \geq 2) &= \max \left\{ \frac{4}{3}\alpha, 2\alpha \right\} \\
 &= 2\alpha.
 \end{aligned}$$

Therefore, for networks satisfying the α -triangle inequality, the T-BETTER heuristic achieves an approximation ratio of $\frac{2\alpha^4 + \alpha^3 + 2\alpha}{\alpha^3 + \alpha + 1}$ if $\alpha \leq 2$, and an approximation ratio of 2α if $\alpha \geq 2$. It is easy to infer that the approximation ratio can be rewritten in a uniform format as $\min \left\{ \frac{2\alpha^4 + \alpha^3 + 2\alpha}{\alpha^3 + \alpha + 1}, 2\alpha \right\}$.

Hence, the theorem is proven. $\square$

Figure 3.8: Approximation ratios of different server placement algorithms for discrete DIAs under the α -triangle inequality

Figure 3.8 illustrates the approximation ratios of the NEAREST, T-GREEDY and T-BETTER algorithms as a function of α for networks satisfying the α -triangle inequality ($\alpha \geq 1$). For ease of reading, the approximation ratio of the k -median server placement is not shown in the figure since it is identical to that of the NEAREST server placement. It can be seen from Figure 3.8 that, although T-GREEDY achieves a much lower approximation ratio than that of NEAREST, the T-BETTER algorithm further improves the approximation ratio compared to the T-GREEDY algorithm when $\alpha \leq 2$.

3.8 Experimental Evaluation

In this section, we evaluate the performance of the proposed server provisioning algorithms and compare their performance with some baseline algorithms using real-world data of network latency. We employ the Meridian dataset [61], which contains the pairwise latency measurements between 2500 nodes in the real Internet. The measurements for some node pairs are not available in the dataset. We discard the nodes involved in the missing measurements, and use the complete pairwise latency matrix among the remaining 1796 nodes as our simulated network. The simulated network does not satisfy the triangle inequality. To examine the extent of triangle inequality violations, we check all the triangles among the nodes. For each triangle (u, v, w) , we compute the ratio of $\frac{d(u,v)}{d(u,w)+d(w,v)}$, where $d(u, v)$ is the longest edge among the three. If the ratio is less than or equal to 1, the triangle satisfies the triangle inequality. If the ratio exceeds 1, its value characterizes the extent of triangle inequality violation. Figure 3.9 shows the cumulative distribution of the ratio. It can be seen that 78% of the triangles satisfy the triangle inequality and for the remaining triangles, their extent of triangle inequality violations has a heavy-tailed distribution.

We compare the four server placements introduced above. To quantify the relative performance of the algorithms, we normalize the total interaction path lengths produced by different algorithms by a theoretical lower bound, which is derived as

Figure 3.9: Cumulative distribution of the extent of triangle inequality violations for all triangles in the Meridian dataset

follows. Given a set of candidate server locations Z , the shortest possible interaction path length between two clients c_i and c_j is given by $\min_{s_a, s_b \in Z} (d(c_i, s_a) + d(s_a, s_b) + d(s_b, c_j))$. Therefore, the total length of interaction paths between all client pairs has a lower bound of

$$\sum_{c_i \in C} \sum_{c_j \in C} \left(\min_{s_a, s_b \in Z} (d(c_i, s_a) + d(s_a, s_b) + d(s_b, c_j)) \right). \quad (3.29)$$

Note that this bound is a super-optimum, which may not be achievable by any real server placement, because it does not enforce each client to connect to its nearest server or even a single server through which it interacts with all the other clients. The total interaction path length normalized by the above bound shall be called the *normalized interactivity*.

Considering the difficulty of finding the optimal k -median placement due to its NP-hardness, we implement a k -median heuristic [45] in our experiments for the purpose of comparison. The heuristic works for k iterations. In each iteration, it adds a new server location that results in the largest reduction in the total latency from the clients to their nearest servers. Besides the four server placements introduced earlier, we also implement a k -favourable heuristic that evolves from the above derivation of the lower bound. Specifically, for each pair of clients c_i and c_j , the candidate server locations giving rise to the shortest possible interaction path between c_i and c_j are recorded.

Figure 3.10: Normalized interactivity of different algorithms when all nodes are candidate server locations

Then, the k -favourable heuristic selects the top k candidate server locations that are most frequently involved in the shortest possible interaction paths between all client pairs.

3.8.1 Performance Comparison for Different Algorithms

We start by not placing any restriction on the available choices of server locations. Specifically, we assume that all 1796 nodes in our simulated network are candidate server locations, and a client is located at each node in the network. Figure 3.10 shows the performance of three algorithms k -median, k -favourable and T-GREEDY for different caps on the number of server locations to select. Note that the NEAREST server placement and the T-BETTER server placement do not consider any limitation on the number of server locations to select and therefore they are not included in this experiment.

It can be seen from Figure 3.10 that server placement has great impacts on the interactivity performance. When only one server location is selected, the T-GREEDY algorithm degenerates to the 1-median placement. Figure 3.10 shows that by selecting additional server locations, T-GREEDY substantially reduces the network latency involved in the interaction between clients over the 1-median placement. This

demonstrates the great potential of distributed server deployment to enhance the interactivity performance of discrete DIAs. However, deploying more servers does not always result in better interactivity between clients—the server locations must be chosen carefully. For example, in the k -median heuristic, the latencies from the clients to their connected servers normally decrease with increasing number of server locations selected since each client connects to its nearest server. Thus, the k -median heuristic always places servers up to the cap number. However, aggressively reducing client-to-server latencies alone may result in longer latencies between the servers of different clients and consequently increase the interaction path length between clients. As seen from Figure 3.10, the interaction path lengths of the k -median heuristic do not always reduce with increasing number of server locations selected because it does not consider inter-server latencies in server provisioning. By taking inter-server latencies into account, the k -favourable heuristic performs much better than the k -median heuristic when a fair number of servers are allowed to be selected. However, its interactivity performance is still considerably worse than our T-GREEDY algorithm. The T-GREEDY algorithm stops selecting new server locations when the total interaction path length cannot be further reduced. Therefore, its interactivity performance does not deteriorate with a higher cap on the number of server locations to select. Figure 3.10 shows that the T-GREEDY algorithm significantly outperforms the other two placements. It can also be seen that T-GREEDY dramatically improves the interactivity performance with increasing number of server locations selected up to 20, and it still provides marginal improvement on the interactivity by selecting more server locations when the number of server locations selected is larger than 20. This implies that using our T-GREEDY algorithm, the DIA operators can flexibly explore the trade-off between interactivity performance and service cost. For example, if there is insufficient budget and the number of server locations that can be selected is very limited, the DIA operators can still achieve reasonably good interactivity performance

Figure 3.11: Normalized interactivity of different algorithms for different numbers of candidate server locations

using the T-GREEDY algorithm.

Next, we restrict the choices of server locations in the network. We assume that a client is located at each of 896 randomly selected nodes. Then, we randomly choose subsets of 75, 150, 300, 600 and 900 nodes from the remaining nodes in the network as the candidate server locations. In this experiment, no limit is set on the number of server locations for our T-GREEDY algorithm to select. We run the T-GREEDY algorithm until it terminates and record the number of server locations actually selected by T-GREEDY. This number is then used as the number of server locations to select in executing the k -median and k -favourable heuristics. In this way, these three algorithms select the same number of server locations in the network to allow for fair comparison. On the other hand, the NEAREST and T-BETTER server placements are not restricted by any stipulated number of server locations to select. For each set size, we perform 1000 simulation runs using 1000 different sets of candidate server locations chosen at random. Figure 3.11 shows the average performance of the algorithms for different numbers of candidate server locations. Here, the total interaction path lengths are normalized by the lower bound (3.29) derived using the respective sets of candidate server locations chosen. Thus, the results of various simulation runs are normalized by different lower bound values. It can be seen from Figure 3.11 that the

Figure 3.12: Cumulative distribution of normalized interactivity for 1000 simulation runs

Figure 3.13: Performance of different algorithms normalized by the same lower bound value for each client set

algorithms considering both client-to-server latencies and inter-server latencies (i.e., k -favourable, T-GREEDY and T-BETTER) produce better interactivity than those considering client-to-server latencies only (i.e., NEAREST and k -median). In our experiments, the T-GREEDY heuristic outperforms the NEAREST server placement in most of the simulation runs, so the T-BETTER algorithm normally produces the same server placement as the T-GREEDY heuristic. Therefore, the difference in the interactivity performance produced by these two server placements is little. They both result in much better interactivity than the other three algorithms. We also plot the cumulative distribution of the normalized interactivity for the simulation runs of different numbers of candidate server locations in Figure 3.12. Since the T-BETTER server placement produces a curve almost the same as that of the T-GREEDY server placement, it is not shown in the figure for ease of reading.

We also normalize the total interaction path lengths of different algorithms by the lower bound (3.29) derived by setting Z as the set of all 900 nodes where clients are not located. In this way, the results of the simulation runs using different subsets of candidate server locations are normalized by the same lower bound value if they have the same client set. Figure 3.13 shows the results with such normalization. It can be

seen that when the number of candidate server locations reduces, the new normalized interactivity becomes worse for all the five algorithms. This is intuitive since a smaller candidate set restricts the choices of server locations more severely, thereby degrading the absolute interactivity performance.

Table 3.4: Average running times of different algorithms for discrete DIAs (in seconds)

Number of candidate server locations	NEAREST	k -median	k -favourable	T-GREEDY	T-BETTER
75	0.026s	0.424s	0.295s	0.285s	0.311s
150	0.028s	0.712s	0.664s	0.475s	0.504s
300	0.029s	1.222s	2.183s	0.846s	0.878s
600	0.030s	2.109s	10.876s	1.945s	1.984s
900	0.031s	2.856s	18.807s	3.413s	3.453s

Table 3.4 reports the average running times of the algorithms for different numbers of candidate server locations. They are measured on a machine with Intel Xeon 3.2GHz CPU and 16GB RAM. In general, our T-GREEDY algorithm and T-BETTER algorithm have similar running times to the k -median placement, and are much faster than the k -favourable algorithm. The running times of our proposed algorithms are very acceptable in that server provisioning is often planned on mid- to long-term basis since deploying new servers may involve amendment to hardware infrastructures or lease agreements with third-party service providers. Moreover, in our T-GREEDY algorithm, the evaluation of each unselected candidate server location in an iteration (lines 6 to 17 in Algorithm 1) can be carried out in parallel. Therefore, the running times of the T-GREEDY algorithm and the T-BETTER algorithm could be further improved by utilizing parallel computing technology. Parallelizing the evaluations in each iteration would not affect the performance results of interactivity and the analysis on the approximation ratios of the algorithms.

Figure 3.14: Normalized interactivity of different algorithms for different individual server capacities

3.8.2 Impact of Server Capacity

All the above experiments have assumed that there is no capacity limitation of the servers. This is the case when we have sufficient server capacities available to be allocated on demand at the candidate server locations. In this section, we further study the impact of server capacity on the performance of the server placement algorithms. Suppose that we have a certain number of servers to allocate, each of which has a given capacity in terms of the maximum number of clients that can connect to it. To place the servers, we first execute each server placement algorithm to select the server locations. Then, we allocate the servers to each selected server location approximately in proportion to the number of clients having it as their nearest server locations. The total capacity of the servers allocated to a server location forms the capacity of this server location. When a server location has been connected by enough clients and filled to its capacity, it would not accept any new connections from the clients, and the affected clients have to turn to their next nearest server locations with spare capacities.

We assume 896 clients and 300 candidate server locations in each simulation run. First, we study the impact of the individual server capacity by testing 36, 18, 12, 9, 6, 4 and 3 servers each with the capacity of 25, 50, 75, 100, 150, 225 and 300

Figure 3.15: Normalized interactivity of different algorithms for different total server capacities

respectively. Note that under these settings, the total server capacity is 900 and just suffices to accommodate all the 896 clients. Figure 3.14 shows the performance of different server placement algorithms. It can be seen that the T-GREEDY and T-BETTER algorithms achieve the best interactivity for all the cases tested. Figure 3.14 also shows the trend that the performance of each algorithm slightly deteriorates when the individual server capacity increases. This is because the number of servers needed to support all the clients decreases with increasing individual server capacity. As a result, the number of locations that are allocated servers decreases accordingly, raising the number of clients that cannot connect to their preferred nearest server locations.

Next, we study the impact of the total available server capacity. In this experiment, the capacity of each server is fixed at 50. Figure 3.15 shows the performance results for different numbers of servers available for allocation. Again, the T-GREEDY and T-BETTER algorithms outperform the other server placements. We can also observe from the figure that the interactivity performance of each server placement generally improves with increasing number of servers available. This is because a larger number of servers increase the maximum number of connections that each server location can accept. Thus, more clients can connect to their preferred nearest server locations.

Figure 3.16: Cumulative distribution of the standard deviations of network latencies for all node pairs in the PlanetLab All Pairs Pings dataset

3.8.3 Impact of Dynamics in Network Conditions

The network latency in real networks may vary over time due to dynamics in network conditions. In this section, we study the impact of the changes in network latency on the performance of the server provisioning algorithms. Since the Meridian dataset does not contain the measurements of network latencies at different times, we use the PlanetLab All Pairs Pings dataset [62] in this experiment. This dataset contains the periodic pairwise latency measurements between PlanetLab nodes at intervals of 15 minutes over a one-day time span. Our experiments with the PlanetLab datasets of different days give similar performance trends. We present here the results for a sample dataset of 4 June 2005. There are 96 measurements in this dataset and 195 nodes consistently involved in all the measurements. Thus, we simulate a network of 195 nodes and emulate the dynamics in the network latencies among the nodes using the data of these 96 measurements. To show the characteristics of latency dynamics in the simulated network, we calculate the standard deviation of the network latency between each node pair based on all the measurements over the one-day time span. Figure 3.16 shows the cumulative distribution of the standard deviations for all node

(a) The connected servers of clients are fixed.

(b) The connected servers of clients change along with the variation of network latency.

Figure 3.17: Cumulative distribution of normalized interactivity by different algorithms under dynamic network latency

pairs. It can be seen that 90% of the node pairs have standard deviations within 25ms and they have relatively small latency fluctuation, while the rest 10% of the node pairs have wider latency fluctuation.

We randomly select 80 nodes as candidate server locations, and assume that a client is located at each of the remaining 115 nodes. We decide the server placement based on the latency data of the first measurement and keep the server placement unchanged throughout the simulation run. Two scenarios of client connections are then simulated. In the first scenario, each client connects to a fixed server (the nearest server as indicated by the latency data of the first measurement) throughout the simulation run. That is, the connected servers of the clients do not change with the network latency. In the second scenario, each client always connects to the nearest server based on the latest latency measurement. As a result, the connected servers of the clients change over time along with the variation of network latency. The theoretical lower bound on the total interaction path length is recalculated at each latency measurement for computing the normalized interactivity. Figure 3.17 shows the cumulative distribution of the normalized interactivity at the times of the 96 latency measurements for 1000 different simulation runs. Since the T-GREEDY server placement usually gives better performance than the NEAREST server placement when using the latency data of the first measurement, the T-BETTER server placement is very similar to T-GREEDY in this experiment. For ease of reading, the results of T-BETTER are not shown in the figure. As can be seen from Figure 3.17, our T-GREEDY algorithm achieves the best interactivity in both scenarios despite the changes in network latency. In addition, the T-GREEDY algorithm also has the minimum fluctuation in the normalized interactivity among the four server provisioning algorithms. For instance, if clients connect to fixed servers (see Figure 3.17(a)), T-GREEDY produces a normalized interactivity within 1.2 in 95% of the times, while the k -favourable, k -median and NEAREST algorithms achieve it in only 79%, 58% and

51% of the time respectively. This implies that the T-GREEDY algorithm is more resilient to the variation of network latency.

Comparing Figures 3.17(a) and 3.17(b), it can be observed that the performance of each server provisioning algorithm is slightly improved when the clients adjust their connections according to the up-to-date latency measurements. These adjustments reduce the latency between each client and its nearest server.

To summarize, the results of this experiment show that good server placement for discrete DIAs is relatively stable over time and adaptive client connections can assist to compensate for the interactivity loss due to short-term variation of network latency.

3.9 Summary

In this chapter, we have investigated the server provisioning problem for discrete DIAs. We have proved that this problem is NP-hard when (i) the network latency does not satisfy the triangle inequality; or (ii) the choices of server locations in the network are restricted; or (iii) the number of server locations to select is limited. We have presented four different server provisioning algorithms with progressively improved approximation ratios. Experimental evaluations are also conducted using real Internet latency data. The results show that aggressively reducing the client-to-server latency alone is not effective for improving the interactivity of discrete DIAs. Our proposed T-GREEDY and T-BETTER algorithms substantially outperform the k -median and other baseline server placements regardless of the server capacity constraint and they are also more resilient to the dynamics in network latency. Moreover, since our T-GREEDY algorithm selects the server locations incrementally, it can be applied to a variety of scenarios to select the server locations. For instance, in presence of server failures, the T-GREEDY algorithm can be used directly to select new server locations to substitute the failed ones while keeping the existing live servers.

Chapter 4

Server Provisioning Problem for Continuous DIAs

In this chapter, we study server provisioning for continuous DIAs. We first give the formulation of the server provisioning problem for continuous DIAs in Section 4.1. Then, the hardness and non-approximability results are presented in Section 4.2. We propose four server provisioning algorithms and study their approximation ratios in Sections 4.3 to 4.7. The proposed algorithms are experimentally evaluated using real Internet latency data in Section 4.8. Finally, the chapter is summarized in Section 4.9.

4.1 Problem Formulation

Similar to Chapter 3, we model the underlying network of the DIA as a graph, whose node set includes a set of clients C and a set of servers S . A distance $d(u, v)$ is associated with each pair of nodes u and v , representing the network latency of the routing path between them.

Each client is connected to a server to participate in the DIA and interact with other clients [27, 40]. When a client performs an operation, the application state of the DIA should be updated accordingly by executing the operation at each server and then the state update should be disseminated to all clients to reflect the change. Consider the interaction between two clients c and c' . In order to notify c' of the

change in the application state caused by an operation initiated by c , c should first send its operation to its connected server s . Then, s would forward the operation to the connected server s' of client c' if s and s' are different. After receiving the operation, s' executes it and sends the resultant state update to c' to complete the interaction. Therefore, the interaction involves the paths from c to s , from s to s' , and from s' to c' . We define the concatenation of these three paths as the *interaction path* from c to c' . The network latency involved in the interaction is given by the length of the interaction path, i.e., $d(c, s) + d(s, s') + d(s', c')$. Note that the interaction path from a client c to itself has length $2d(c, s)$, which is the network latency involved for c to see the effect of its own operation.

In continuous DIAs, the application states change due to not only client-initiated operations but also time passing [6]. Thus, the progress of the application state is often measured along a synthetic timescale known as the *simulation time* (for example, the time elapsed in the virtual game world of an online game) [50]. To ensure that all clients consistently observe the same effect caused by a user operation, the operation must be executed by all servers at the same simulation time. As a result, the servers may not be able to execute user-initiated operations immediately upon receiving them. They may have to wait and give enough time for the operations to be delivered to other servers [5, 27, 31]. In addition, fairness is concerned with the order of executing user-initiated operations [12]. To guarantee that all clients have equal chance of participation regardless of their network conditions, the user operations must be executed in the simulation time order of their initiations at the clients. The execution of operations must also preserve the simulation time interval between the operation initiations. Due to diverse network latencies, maintaining consistency and fairness in continuous DIAs introduce artificial synchronization delays in the interactions among clients.

Zhang et al. [50] have proved that the minimum achievable interaction time be-

(a) Interaction time when all servers have synchronized simulation times

(b) Minimum achievable interaction time

Figure 4.1: An example of minimum achievable interaction time.

tween clients for fulfilling the consistency and fairness requirements is given by the length of the longest interaction path among all clients. Figure 4.1 shows a simple example to illustrate the minimum achievable interaction time. Suppose that two clients c_1 and c_2 are connected to servers s_1 and s_2 respectively. The latencies from c_1 to s_1 , from c_2 to s_2 , and the inter-server latency are 5, 3 and 2 respectively. A straightforward simulation time setting is to synchronize the simulation times at the two servers (see Figure 4.1(a)). In this case, if an operation is issued at simulation time 0, considering all possible initiating clients, the latest possible time for all the servers to receive the operation is at simulation time 12. Thus, the earliest simulation

time for the servers to execute the operation is 12 in order to guarantee the consistency and fairness. Therefore, the resultant state update can only be seen by the clients after a time lag of 12. On the other hand, if the relative offset of simulation times at s_1 and s_2 is set to 2 as shown in Figure 4.1(b), all the servers would be able to receive an operation issued at simulation time 0 by time 10 at the latest, irrespective of the client initiating the operation. As a result, the interaction time is reduced to 10. This is the minimum achievable interaction time, which is equal to the maximum interaction path length.

In many DIAs, the clients are naturally connected to their nearest servers, i.e., the servers having the shortest network latency to them [27,30,40,52]. Thus, given a set of clients, the longest interaction path is primarily determined by where the servers are placed to support client interactions. Suppose that there is a set of candidate server locations Z in the network where servers can be placed. Then, the server provisioning problem for continuous DIAs is formulated as follows.

Server Provisioning Problem for Continuous DIAs. *Given a set of clients C and a set of candidate server locations Z , find a set of locations $S \subseteq Z$ to place servers so that the maximum interaction path length among all clients, i.e.,*

$$\max_{c,c' \in C} \{d(c, n(c, S)) + d(n(c, S), n(c', S)) + d(n(c', S), c')\},$$

is minimized, where $n(c, S)$ denotes client c 's nearest server in S .

Again, in the above problem definition, we do not assume any server capacity restriction at the candidate server locations. Extending our proposed algorithms to handle server capacity restrictions shall be discussed in Section 4.8.2.

4.2 Hardness of the Problem

If the network latency satisfies the triangle inequality, for any pair of clients c and c' , it is obvious that $d(c, c') \leq d(c, s) + d(s, s') + d(s', c')$ for any two servers s and

s' regardless of their locations. Thus, the best arrangement in this case is to have a server co-located with each client so that all the clients connect to their co-located servers, producing the shortest possible interaction path length between each pair of clients. However, Internet routing is not optimal in terms of network latency due to business-oriented strategies [56]. Normally, it is also impossible to place servers at the same locations as clients. For example, DIA operators who cannot afford to set up their own server infrastructures may have to run servers at the data centers operated by cloud providers. In addition, there might also be a limit on the number of server locations that a DIA operator can choose due to its budget constraints. Under *any* one of the above conditions, the server provisioning problem is NP-hard. We first show the NP-hardness of the problem for networks without the triangle inequality in Section 4.2.1. Then, Section 4.2.2 presents the NP-hardness of the server provisioning problem when there are restricted choices of server locations. Finally, we show that the problem is also NP-hard when there is limited number of server locations to select in Section 4.2.3.

4.2.1 Hardness for Networks without Triangle Inequality

In the case that the network latency does not satisfy the triangle inequality, the NP-hardness of the server provisioning problem for continuous DIAs can be proved by a polynomial reduction from the minimum set cover problem which is NP-hard [47]. The decision version of the set cover problem is defined as follows: Given a finite set $P = \{p_1, p_2, \dots, p_n\}$ and a collection $\mathbb{Q} = \{Q_1, Q_2, \dots, Q_m\}$ of its subsets, and a positive integer $k \leq m$, decide whether $\mathbb{Q}$ contains a subcollection $\mathbb{Q}'$ of at most k subsets such that $\bigcup_{Q \in \mathbb{Q}'} Q = P$.

The polynomial reduction is similar to that in Sections 3.2.1 and 3.2.2 of Chapter 3, but we deliberately tune the latencies here for yielding the strongest non-approximability results later on. Given an instance $\mathcal{R}$ of the set cover problem, we

$$P = \{p_1, p_2, p_3\}, Q_1 = \{p_1, p_3\}, Q_2 = \{p_2\}, Q_3 = \{p_3\}$$

$$\mathbb{Q} = \{Q_1, Q_2, Q_3\}, k = 2. \text{ Set Cover: } \mathbb{Q}' = \{Q_1, Q_2\}$$

Figure 4.2: Example instances of the set cover problem and the server provisioning problem for continuous DIAs

first construct a network comprising a set V_C of n nodes and a set V_S of k disjoint groups of nodes as shown in Figure 4.2. The n nodes of V_C map onto to the elements in the set P of instance $\mathcal{R}$. Each node group of V_S contains m cliques of sizes $|Q_1|, |Q_2|, \dots$, and $|Q_m|$, each of which corresponds to a subset in the collection $\mathbb{Q}$ by mapping each node in the clique onto an element in the corresponding subset.

In the constructed network, any two nodes in different groups of V_S have a latency t_2 to each other. Any two nodes in the same group of V_S have a latency t_2 if they belong to the same clique or map onto the same element in P . Otherwise, they have a latency t_4 . For any two nodes $u \in V_S$ and $v \in V_C$, they have a latency t_1 if both u and v map onto the same element in P . Otherwise, they have a latency t_3 . Any two nodes in V_C have a latency t_4 to each other. For ease of reading, Figure 4.2 illustrates only the network latencies relevant to a representative node in V_S and the latencies among the nodes in V_C .

We assume that

- $t_4 > 2t_1 + t_2$,

- $t_3 > t_1 + t_2$.

The second inequality implies that the constructed network does not satisfy the triangle inequality due to a triangle with sides t_1 , t_2 and t_3 as shown in Figure 4.2.

We then define an instance $\mathcal{T}$ of the server provisioning problem in its decision version as follows: Given the constructed network, assuming that a client is located at each node of V_C and all the nodes in the network (i.e., $V_C \cup V_S$) are candidate server locations, can we select a set of server locations to bound the maximum interaction path length among all clients by $2t_1 + t_2$?

We first prove that if there is an acceptable set cover for instance $\mathcal{R}$, then there must exist a valid solution for the server provisioning instance $\mathcal{T}$. Suppose that $\mathbb{Q}' = \{Q_{x_1}, Q_{x_2}, \dots, Q_{x_l}\}$ (where $1 \leq l \leq k$) is a set cover. We can construct a server placement S by selecting all the nodes in the x_1 -th clique in the first group of V_S , the x_2 -th clique in the second group of V_S , $\dots$, and the x_l -th clique in the l -th group of V_S as server locations (see Figure 4.2 for an example). Note that each element of P is covered by at least one subset in $\mathbb{Q}'$. Therefore, for each client in V_C , at least one clique selected by S should contain a server having latency t_1 to the client. Our assumption $t_3 > t_1 + t_2$ implies that $t_3 > t_1$. So, each client has latency t_1 to its nearest server in S . Furthermore, placement S selects at most one clique from each group of V_S . Thus, all the inter-server latencies in placement S are t_2 . As a result, the maximum interaction path length between any two clients is $2t_1 + t_2$. Therefore, server placement S is a valid solution for instance $\mathcal{T}$.

Next, we prove that if there is a valid solution for the server provisioning instance $\mathcal{T}$, there must exist a set cover of size at most k for instance $\mathcal{R}$. We start by showing that in any valid solution of instance $\mathcal{T}$, each client and its nearest server must map onto the same element in P . If a client and its nearest server do not map onto the same element, the latency between them would be t_3 if $s \in V_S$ or would be t_4 if $s \in V_C$.

Then, the interaction path from the client to itself is $2t_3$ or $2t_4$ long, which exceeds the bound $2t_1 + t_2$ since $t_3 > t_1 + t_2$ and $t_4 > 2t_1 + t_2$. Therefore, each client and its nearest server must map onto the same element. This implies that: (a) if the nearest server s of a client c is located in V_C , the latency between c and s can only be 0, i.e., s and c must be located at the same node; (b) if the nearest server s of a client c is located in V_S , the latency between them must be t_1 .

If two clients both have their nearest servers in V_C , based on (a), their interaction path length would be $0 + t_4 + 0 = t_4 > 2t_1 + t_2$. If two clients have their nearest servers in V_C and V_S respectively, the inter-server latency must be t_3 since the two servers map onto different elements. As a result, based on (a) and (b), the interaction path length between the two clients is $0 + t_3 + t_1 > (t_1 + t_2) + t_1 = 2t_1 + t_2$. Therefore, all the clients must have their nearest servers in V_S in any valid solution of instance $\mathcal{T}$. Let $S \subseteq V_S$ be a valid solution. Without loss of generality, assume that each server in S is connected by at least one client (any server not connected by any clients can simply be removed from S). According to (b), to cap the maximum interaction path length by $2t_1 + t_2$, the latency between any two distinct servers in S must be bounded by $(2t_1 + t_2) - t_1 - t_1 = t_2$. Our assumption $t_4 > 2t_1 + t_2$ implies that $t_4 > t_2$. Thus, servers located in the same group of V_S must come from the same clique therein. Since there are k groups in V_S , the number of cliques with servers placed is at most k . Therefore, the subsets in $\mathbb{Q}$ that correspond to the cliques with servers placed form a set cover of size at most k for instance $\mathcal{R}$.

The above analysis establishes that a set cover of size at most k can be found for instance $\mathcal{R}$ if and only if there exists a valid solution for instance $\mathcal{T}$. Therefore, the server provisioning problem for networks without the triangle inequality is NP-hard.

Theorem 10. *If $P \neq NP$, no polynomial-time algorithm can achieve any constant approximation ratio for the server provisioning problem for continuous DIAs on networks*

without the triangle inequality.

Proof. This result can be proved based on the server provisioning instance constructed above. If there exists a set cover of size at most k , the maximum interaction path length under the optimal server placement is $2t_1 + t_2$. According to the above analysis, in a non-optimal server placement, either (1) some client and its nearest server do not map onto the same element, or (2) some client has its nearest server located in V_C , or (3) two servers in V_S have latency t_4 to each other. In case (1), the interaction path from that client to itself is $2t_3$ or $2t_4$ long. In case (2), there exists an interaction path with length t_4 or $t_3 + t_1$. In case (3), there exists an interaction path with length at least t_4 . Since $t_3 > t_1 + t_2 > t_1$, the maximum interaction path length under a non-optimal server placement is at least $\min\{2t_3, 2t_4, t_4, t_3 + t_1\} = \min\{t_4, t_3 + t_1\}$. If a polynomial-time server provisioning algorithm achieves a constant approximation ratio γ , by setting $t_3 = t_4 = (2t_1 + t_2) \cdot \gamma$ in the constructed network, it is guaranteed to find a set cover of size at most k if one exists, which contradicts $P \neq NP$. Thus, server provisioning for networks without the triangle inequality is not approximable within any constant factor.

Hence, the theorem is proven. □

4.2.2 Hardness for Restricted Choices of Server Locations

In the case that the choices of server locations are restricted to a subset of nodes in the network, the server provisioning problem is also NP-hard. Since server provisioning for networks without the triangle inequality has been shown to be NP-hard in the previous section, we shall focus our discussion here on server provisioning with restricted choices of server locations on networks with the triangle inequality only.

The NP-hardness can be proved by a similar reduction from the minimum set cover problem to that in the previous section. Given an instance $\mathcal{R}$ of the set cover problem, we construct the same network as shown in Figure 4.2. Suppose that the latency values

t_1, t_2, t_3 and t_4 satisfy the triangle inequality for the constructed network. In addition, we also assume that

- $2t_3 > 2t_1 + t_2$,
- $t_4 > t_2$.

We restrict the choices of server locations to the nodes in V_S only and define an instance $\mathcal{X}$ of the server provisioning problem as follows: assuming that a client is located at each node of V_C , can we select a set of server locations from V_S to bound the maximum interaction path length by $2t_1 + t_2$?

It can be proved in the same way as in the previous section that a valid solution for the server provisioning instance $\mathcal{X}$ can always be derived from a set cover of size at most k . Next, we show that a set cover of size at most k can always be derived from a valid solution for instance $\mathcal{X}$.

Note that the latency from any node in V_C to any node in V_S is either t_1 or t_3 . If any client has latency exceeding t_1 to its nearest server, it must have a latency of t_3 to its nearest server. In consequence, the interaction path length from such client to itself is $2t_3 > 2t_1 + t_2$. Thus, in any valid solution for instance $\mathcal{X}$, the latency from each client to its nearest server can only be t_1 . This implies that each client and its nearest server should map onto the same element in set P of the set cover instance $\mathcal{R}$. Consider a valid server placement S in which each server is connected by at least one client. To cap the maximum interaction path length by $2t_1 + t_2$, the latency between any two distinct servers in S must be bounded by $(2t_1 + t_2) - t_1 - t_1 = t_2$. Since $t_4 > t_2$, the servers located in the same group of V_S must come from the same clique therein. Since there are k groups in V_S , the number of cliques with servers placed is at most k . Therefore, the subsets in $\mathbb{Q}$ that correspond to the cliques with servers placed form a set cover of size at most k for instance $\mathcal{R}$.

Thus, there exists a set cover of size at most k if and only if there exists a valid server provisioning solution. Therefore, server provisioning with restricted choices of server locations on networks with the triangle inequality is NP-hard. Next, we analyze the non-approximability for server provisioning in this case.

Theorem 11. *If $P \neq NP$, no polynomial-time algorithm can achieve an approximation ratio within $3/2$ for the server provisioning problem for continuous DIAs with restricted choices of server locations on networks with the triangle inequality.*

Proof. Consider the server provisioning instance constructed above. According to the above analysis, if there exists a set cover of size at most k , the maximum interaction path length under the optimal server placement is $2t_1 + t_2$. On the other hand, the maximum interaction path length is at least $\min\{2t_3, 2t_1 + t_4\}$ under a non-optimal server placement. This is because in a non-optimal server placement, either some client and its nearest server do not map onto the same element in set P , or the latency between some pair of servers is t_4 . In the former case, that client has latency t_3 to its nearest server, so the interaction path from that client to itself is $2t_3$ long. In the latter case, there exists an interaction path with length at least $t_1 + t_4 + t_1 = 2t_1 + t_4$. No server placement can produce a maximum interaction path length between $2t_1 + t_2$ and $\min\{2t_3, 2t_1 + t_4\}$. Therefore, if a polynomial-time server provisioning algorithm is able to achieve an approximation ratio within $\frac{\min\{2t_3, 2t_1 + t_4\}}{2t_1 + t_2}$, it is guaranteed to find a set cover of size at most k if there exists one. This contradicts to the assumption of $P \neq NP$ since the set cover problem is NP-hard. Thus, server provisioning with restricted choices of server locations is not approximable within a ratio of $\frac{\min\{2t_3, 2t_1 + t_4\}}{2t_1 + t_2}$.

To yield the strongest non-approximability result, the latency values t_1 , t_2 , t_3 and t_4 can be tuned to make the ratio $\frac{\min\{2t_3, 2t_1 + t_4\}}{2t_1 + t_2}$ as large as possible. Note that there exist a triangle with sides t_1 , t_2 and t_3 and another triangle with sides t_2 , t_2 and t_4 in the constructed network (see Figure 4.3). By the triangle inequality, we have

Figure 4.3: Some triangles in the constructed network for server provisioning instance $\mathcal{X}$.

$2t_3 \leq 2t_1 + 2t_2$ and $t_4 \leq 2t_2$. Thus,

$$\frac{\min\{2t_3, 2t_1 + t_4\}}{2t_1 + t_2} \leq \frac{2t_1 + 2t_2}{2t_1 + t_2}.$$

Furthermore, the presence of a triangle with sides t_1 , t_1 and t_2 in the constructed network (see Figure 4.3) implies that $t_2 \leq 2t_1$. Therefore, $t_2 \leq \frac{1}{2}(2t_1 + t_2)$. It follows that

$$\begin{aligned} \frac{\min\{2t_3, 2t_1 + t_4\}}{2t_1 + t_2} &\leq \frac{2t_1 + 2t_2}{2t_1 + t_2} \\ &= 1 + \frac{t_2}{2t_1 + t_2} \\ &\leq 1 + \frac{1}{2} = \frac{3}{2}. \end{aligned}$$

So, the largest possible ratio of $\frac{\min\{2t_3, 2t_1 + t_4\}}{2t_1 + t_2}$ is $\frac{3}{2}$. This can be achieved by setting $t_1 = 1$, $t_2 = 2$, $t_3 = 3$ and $t_4 = 4$. It is easy to verify that such latency setting satisfies the triangle inequality for the constructed network. Therefore, server provisioning with restricted choices of server locations on networks with the triangle inequality cannot be approximated with a ratio of $\frac{3}{2}$.

Hence, the theorem is proven. $\square$

Figure 4.4: Example instances of the X3C problem and the server provisioning problem for continuous DIAs

4.2.3 Hardness for Limited Number of Server Locations to Select

When there is a limit on the number of server locations to select, the server provisioning problem is again NP-hard even if the network latency satisfies the triangle inequality, and there is no restriction on the choices of server locations. This can be proved by a polynomial reduction from the Exact Cover By 3-Sets (X3C) problem [47]. Given a set X of $3q$ elements, and a collection $\mathbb{O}$ of X 's 3-element subsets, the X3C problem is to decide whether $\mathbb{O}$ contains an exact cover for X , i.e., a subcollection $\mathbb{O}' \subseteq \mathbb{O}$ such that every element of X occurs in exactly one subset in $\mathbb{O}'$.

Given an instance $\mathcal{U}$ of the X3C problem, we first construct a network composed of node sets V_C and V_S as shown in Figure 4.4. V_C is a set of $3q$ nodes, each corresponding to an element in set X . V_S is a set of $|\mathbb{O}|$ nodes, each corresponding to a subset in $\mathbb{O}$. The nodes in V_S are connected to each other by links of latency 1, and each node in V_S has links of latency 1 to the three nodes in V_C that represent the three elements of its corresponding subset. Unmarked node pairs have network latencies given by the lengths of shortest paths. Apparently, the network latencies satisfy the triangle inequality.

An instance $\mathcal{F}$ of the server provisioning problem is then defined as follows: Given the constructed network, assuming that a client is located at each node in V_C , and all the nodes in the network are candidate server locations, can we select up to q locations to place servers such that the maximum interaction path length among all clients is bounded by 3?

First, we show that if there is an exact cover $\mathbb{O}' \subseteq \mathbb{O}$ for instance $\mathcal{U}$, a valid solution must exist for instance $\mathcal{F}$. Since each subset in $\mathbb{O}$ has exactly 3 elements, it is obvious that $|\mathbb{O}'| = q$. Let S be the q nodes in V_S that represent the subsets in $\mathbb{O}'$. We select these nodes as server locations (see Figure 4.4 for an example). Since $\mathbb{O}'$ is an exact cover, each client has exactly one link of latency 1 to a server in S and connects to that server. Since all the nodes in V_S are interconnected with each other by links of latency 1, the maximum interaction path length among all clients is bounded by $1 + 1 + 1 = 3$. Thus, placement S is a valid solution for instance $\mathcal{F}$.

On the other hand, if we have a valid solution for instance $\mathcal{F}$, an exact cover for instance $\mathcal{U}$ can be found accordingly. We first show that selecting any nodes in V_C to place servers would make the maximum interaction path length exceed the bound of 3. Suppose that b ($b > 0$) nodes in V_C are selected as server locations. Then, the clients located at these nodes would connect to their co-located servers. The other clients have latency at least 2 to these servers. Among the nodes in V_S , at most $(q - b)$ of them could be selected to place servers. Since each node in V_S has latency 1 to exactly three nodes in V_C , at most $3(q - b)$ clients can have latency 1 to the servers placed in V_S . The remaining $3q - 3(q - b) - b = 2b$ clients would have to connect to servers with latency at least 2 from them. So, the interaction path lengths among these $2b$ clients are at least $2 + 2 > 3$. Therefore, in a valid solution for instance $\mathcal{F}$, all the servers must be placed in V_S . In order to cap the maximum interaction path length at 3, each client must have latency 1 to its nearest server. Since there are $3q$ clients in total, q server locations must be selected in V_S and their corresponding

3-element subsets form an exact cover.

The above analysis shows that an exact cover can be found for instance $\mathcal{U}$ if and only if there exists a valid solution for instance $\mathcal{F}$. So, the server provisioning problem with a limit on the number of server locations to select is NP-hard.

Theorem 12. *If $P \neq NP$, no polynomial-time algorithm can achieve an approximation ratio within $4/3$ for the server provisioning problem for continuous DIAs with a limit on the number of server locations to select.*

Proof. Consider the server provisioning instance $\mathcal{F}$ constructed above. If there exists an exact cover, an optimal server placement has a maximum interaction path length 3. On the other hand, there always exist clients having latency at least 2 to their nearest servers in non-optimal placements, giving rise to interaction paths of length at least 4. This implies that no server placement can produce a maximum interaction path length between 3 and 4. Therefore, if a polynomial-time algorithm can achieve an approximation ratio within $4/3$, it can always find a server placement with maximum interaction path length 3 if there exists one, following which an exact cover can be derived. Since the X3C problem is NP-hard, this leads to contradiction to the assumption $P \neq NP$. Hence, the theorem is proven. $\square$

Using similar methodologies to Chapter 3, we present four server placements for the server provisioning problem for continuous DIAs. The network latencies between clients and candidate server locations, which are needed for the computation of these four algorithms, can be acquired with existing tools like ping and King [57]. We first analyze the classical k -center server placement which is related to our server provisioning problem. We show that, in networks with the triangle inequality, the k -center server placement has an approximation ratio of 3 for our server provisioning problem. Note that the k -center placement is difficult to find owing to its NP-hardness. We then propose a NEAREST server placement that achieves the same approximation ra-

Table 4.1: Summary of different server placement algorithms for continuous DIAs

	approximation ratio for networks with triangle inequality	approximation ratio for networks with α -triangle inequality	time complexity
k -center	3	$\frac{3}{4}\alpha^3 + \frac{5}{4}\alpha^2 + 1$	NP-hard
NEAREST	3	$\max\left\{\frac{1}{2}\alpha^3 + \frac{3}{2}\alpha^2 + 1, \frac{2}{3}\alpha^3 + \alpha^2 + \frac{2}{3}\right\}$	$O(Z C)$
M-GREEDY	2	2α	$O(k Z (C + k^2))$
M-BETTER	$\frac{5}{3}$	$2\alpha \cdot \min\left\{1, \frac{2\alpha^3 + 4\alpha^2 + 4}{\alpha^3 + 3\alpha^2 + 4\alpha + 4}\right\}$	$O(k Z (C + k^2))$

tio as that of the k -center placement, and can be computed in polynomial time. After that, we propose an efficient M-GREEDY algorithm that improves the approximation ratio to 2. Finally, the fourth algorithm called M-BETTER is proposed to further improve the approximation ratio to $\frac{5}{3}$. In Section 4.7, we conduct theoretical analysis on the asymptotic approximability of these four different server placements for networks with triangle inequality violations. Suppose that $|C|$ is the number of clients and $|Z|$ is the number of the candidate server locations in the network. Let k be the cap on the maximum number of server locations to select. Table 4.1 summarizes the approximation ratios and the computational complexities of the four algorithms.

4.3 k -Center Server Placement

Our server provisioning problem for continuous DIAs bears some similarity to the classical k -center problem in that both problems aim to find server placement that minimizes a maximum latency metric in a network of clients and servers. However, different from our problem, the k -center problem targets at placing a given number of k servers in the network to minimize the maximum latency between the clients and their nearest servers. It does not consider the latencies among the servers. When a limit k is set on the number of server locations to select in our server provisioning problem, we have the following approximability result of the k -center server placement.

Theorem 13. *For networks with the triangle inequality, the k -center server placement*

has an approximation ratio of 3 for the server provisioning problem for continuous DIAs with a limit k on the number of server locations to select.

Proof. Let C be the set of clients. Suppose that the sets of optimal server locations for our server provisioning problem and the k -center problem are S_O and S_R respectively. For each client $c \in C$, denote by $o(c)$ and $r(c)$ the nearest servers of c under placements S_O and S_R respectively. Then, for any two clients c_i and $c_j \in C$, their interaction path length under placement S_R is given by

$$l_R(c_i, c_j) = d(c_i, r(c_i)) + d(r(c_i), r(c_j)) + d(r(c_j), c_j).$$

By the triangle inequality, we have

$$d(r(c_i), r(c_j)) \leq d(r(c_i), c_i) + d(c_i, c_j) + d(c_j, r(c_j)).$$

Thus,

$$\begin{aligned} l_R(c_i, c_j) &\leq d(c_i, r(c_i)) + \left(d(r(c_i), c_i) + d(c_i, c_j) \right. \\ &\quad \left. + d(c_j, r(c_j)) \right) + d(r(c_j), c_j) \\ &= 2d(c_i, r(c_i)) + 2d(c_j, r(c_j)) + d(c_i, c_j). \end{aligned}$$

Let $m = \max_{c \in C} d(c, r(c))$ be the maximum latency from the clients to their nearest servers under the k -center placement S_R . It follows that

$$l_R(c_i, c_j) \leq 4m + d(c_i, c_j).$$

By the definition of the k -center placement, m is the minimum achievable value of the maximum latency from the clients to their nearest servers among all possible placements of k servers and hence among all possible placements of up to k servers. Thus, there must exist a client $c_x \in C$ satisfying $d(c_x, o(c_x)) \geq m$ under the optimal placement S_O . Therefore, by the triangle inequality, we have

$$\begin{aligned} l_R(c_i, c_j) &\leq 2 \cdot (2 \cdot d(c_x, o(c_x))) + d(c_i, c_j) \\ &\leq 2 \cdot (d(c_x, o(c_x)) + d(o(c_x), o(c_x)) + d(o(c_x), c_x)) \end{aligned}$$

$$+d(c_i, o(c_i)) + d(o(c_i), o(c_j)) + d(o(c_j), c_j).$$

Denote by L_O the maximum interaction path length under the optimal server placement S_O . Then,

$$d(c_x, o(c_x)) + d(o(c_x), o(c_x)) + d(o(c_x), c_x) \leq L_O,$$

and

$$d(c_i, o(c_i)) + d(o(c_i), o(c_j)) + d(o(c_j), c_j) \leq L_O.$$

It follows that

$$l_R(c_i, c_j) \leq 3 \cdot L_O. \tag{4.1}$$

The above inequality holds for any two clients c_i and c_j . As a result, the maximum interaction path length L_R under the k -center server placement satisfies

$$\begin{aligned} L_R &= \max_{c_i, c_j \in C} l_R(c_i, c_j) \\ &\leq 3 \cdot L_O. \end{aligned}$$

That is, the maximum interaction path length produced by the k -center server placement S_R is within three times of that in the optimal server placement S_O .

Hence, the theorem is proven. $\square$

The proof of Theorem 13 also leads to the following result for the server provisioning problem when there is no limit on the number of server locations to select.

Corollary 3. *For each instance of the server provisioning problem for continuous DIAs on a network satisfying the triangle inequality, if an optimal server placement selects k server locations, then the k -center server placement produces a maximum interaction path length within three times of that in the optimal server placement.*

Figure 4.5: The approximation ratio 3 of the k -center placement is tight.

The approximation ratio 3 of the k -center server placement is tight. A tight example is given in Figure 4.5, which consists of four candidate server locations $\{s_1, s_2, o_1, o_2\}$ and two clients $\{c_1, c_2\}$. The latencies between some node pairs are marked in the figure. The latencies between unmarked node pairs are given by the lengths of their shortest paths. Suppose that $\varepsilon > 0$ and up to 2 server locations can be selected. Then, the set of optimal server locations for the server provisioning problem is $\{o_1, o_2\}$. Under such placement, clients c_1 and c_2 are connected to servers o_1 and o_2 respectively, so the maximum interaction path length is $2 + 3\varepsilon$. On the other hand, the 2-center placement is $\{s_1, s_2\}$, which gives a maximum latency of 1 between the clients and their nearest servers. In this case, the maximum interaction path length is $6 + 3\varepsilon$. Thus, the ratio between the two results is $\frac{6+3\varepsilon}{2+3\varepsilon}$, which can be made arbitrarily close to

$$\lim_{\varepsilon \rightarrow 0} \frac{6 + 3\varepsilon}{2 + 3\varepsilon} = 3$$

when ε approaches 0.

4.4 Nearest Server Placement

The k -center problem is NP-hard [47]. So, it is difficult to compute the k -center server placement. In this section, we propose a simple server placement called NEAREST that has the same approximation ratio as the k -center placement for networks with the triangle inequality, and can be computed in polynomial time.

Suppose that C is the set of clients and Z is the set of candidate server locations in the network. For each client $c \in C$, let $n(c)$ be the nearest candidate server location to c , i.e., $d(c, n(c)) = \min_{s \in Z} \{d(c, s)\}$. As defined in Section 3.4 of Chapter

3, the NEAREST server placement S_N is to select the collection of all clients' nearest candidate server locations, i.e., $S_N = \bigcup_{c \in C} n(c)$. Note that the time complexity to find $n(c)$ for each client c is $O(|Z|)$. Thus, the overall time complexity of the NEAREST server placement is $O(|Z||C|)$.

Theorem 14. *For networks with the triangle inequality, the NEAREST server placement has an approximation ratio of 3 for the server provisioning problem for continuous DIAs.*

Proof. Let S_N be the NEAREST server placement. For any two clients c_i and $c_j \in C$, their interaction path length under placement S_N is given by

$$l_N(c_i, c_j) = d(c_i, n(c_i)) + d(n(c_i), n(c_j)) + d(n(c_j), c_j).$$

By the triangle inequality, we have

$$\begin{aligned} l_N(c_i, c_j) &\leq d(c_i, n(c_i)) + \left(d(n(c_i), c_i) + d(c_i, c_j) \right. \\ &\quad \left. + d(c_j, n(c_j)) \right) + d(n(c_j), c_j) \\ &= 2d(c_i, n(c_i)) + 2d(c_j, n(c_j)) + d(c_i, c_j). \end{aligned}$$

Let c_x be the client that is furthest away from its nearest server under placement S_N , i.e., $d(c_x, n(c_x)) = \max_{c \in C} d(c, n(c))$. It follows that

$$l_N(c_i, c_j) \leq 4 \cdot d(c_x, n(c_x)) + d(c_i, c_j).$$

By the definition of S_N , $n(c_x)$ is the nearest candidate server location to c_x among all candidate server locations. Let $o(c_x)$ be the nearest server of client c_x under an optimal server placement S_O . Then, we have $d(c_x, n(c_x)) \leq d(c_x, o(c_x))$. Therefore,

$$l_N(c_i, c_j) \leq 4 \cdot d(c_x, o(c_x)) + d(c_i, c_j).$$

Similar to the derivation of (4.1) in Theorem 13, we have

$$l_N(c_i, c_j) \leq 3 \cdot L_O,$$

where L_O is the maximum interaction path length under the optimal server placement S_O . It follows that the maximum interaction path length L_N under the NEAREST server placement satisfies

$$\begin{aligned} L_N &= \max_{c_i, c_j \in C} l_N(c_i, c_j) \\ &\leq 3 \cdot L_O. \end{aligned}$$

Hence, the theorem is proven. $\square$

The approximation ratio 3 of the NEAREST server placement is also tight. Consider again the example given in Figure 4.5. Note that the nearest candidate server locations for clients c_1 and c_2 are s_1 and s_2 respectively. This implies that the NEAREST server placement is $\{s_1, s_2\}$, which is identical to the 2-center placement. Thus, this example is also applicable to showing the tightness of the approximation ratio 3 for the NEAREST server placement.

4.5 M-Greedy Server Placement

In this section, we develop an efficient heuristic called M-GREEDY for our server provisioning problem. For networks with the triangle inequality, it improves the approximation ratio to 2. Algorithm 2 shows the pseudo code of the M-GREEDY algorithm. Denote by S the set of server locations selected. Starting from an empty set S , the algorithm iteratively adds a candidate server location $s^* \in Z \setminus S$ to S in a greedy manner that results in the largest reduction in the maximum interaction path length M (lines 12, 13 and 18). If adding any server location from $Z \setminus S$ cannot further shorten the maximum interaction path length (lines 16-17), or when $|S|$ reaches a given limit k (if any) on the number of server locations to select (line 7), the algorithm terminates and outputs the final server placement S (line 27).

A brute force calculation of the maximum interaction path length has a time complexity of $O(|C|^2)$, where $|C|$ is the number of clients. Thus, a naive implementation

```

1:  $S \leftarrow \emptyset$ ; //server locations selected
2:  $M_p \leftarrow \infty$ ; //maximum interaction path length for placement  $S$ 
3: for all  $z \in Z$  do
4:    $\mathbb{C}(z) \leftarrow C$ ;  $R_p(z) \leftarrow 0$ ;  $L_p(z) \leftarrow 0$ ;  $E_p(z) \leftarrow \emptyset$ ;
5:   for all  $c \in C$  do
6:      $n(c) \leftarrow \emptyset$ ;
7:   for  $i = 1$  to  $k$  do
8:      $M^* \leftarrow \infty$ ,  $s^* \leftarrow \emptyset$ ;
9:     for all  $z \in Z \setminus S$  do
10:      updateLR( $z$ ); //invoke Algorithm 3
11:       $M \leftarrow \max_{s \in S \cup \{z\}} \{L(s) + R(s)\}$ ;
12:      if  $M < M^*$  then
13:         $M^* \leftarrow M$ ,  $s^* \leftarrow z$ ;
14:        for all  $s \in S \cup \{z\}$  do
15:           $R^*(s) \leftarrow R(s)$ ;  $L^*(s) \leftarrow L(s)$ ;  $E^*(s) \leftarrow E(s)$ ;
16:      if  $M^* \geq M_p$  then
17:        break;
18:       $S \leftarrow S \cup \{s^*\}$ ;  $M_p \leftarrow M^*$ ;
19:      for all  $c \in \mathbb{C}(s^*)$  do
20:         $\mathbb{C}(n(c)) \leftarrow \mathbb{C}(n(c)) \setminus \{c\}$ ;  $n(c) \leftarrow s^*$ ;
21:      for all  $z \in Z \setminus S$  do
22:        for all  $c \in \mathbb{C}(z)$  do
23:          if  $d(c, z) > d(c, s^*)$  then
24:             $\mathbb{C}(z) \leftarrow \mathbb{C}(z) \setminus \{c\}$ ;
25:      for all  $s \in S$  do
26:         $R_p(s) \leftarrow R^*(s)$ ;  $L_p(s) \leftarrow L^*(s)$ ;  $E_p(s) \leftarrow E^*(s)$ ;
27: return  $S$ ;
    
```

Algorithm 2: M-GREEDY server provisioning algorithm

of the above greedy algorithm has a computational complexity of $O(|Z| \cdot |C|^2)$ for each iteration and $O(k \cdot |Z| \cdot |C|^2)$ in total. To efficiently compute the maximum interaction path length when testing and selecting new server locations, we decompose the maximum length of the interaction paths involving each server s into two parts: a local latency $L(s)$ and a remote latency $R(s)$. The local latency is defined as that from s to the furthest client connecting to it, i.e.,

$$L(s) = \max_{n(c, S)=s} d(c, s),$$

where $n(c, S)$ denotes client c 's nearest server in S . The remote latency $R(s)$, on the other hand, is defined as the maximum latency from s to all the clients through their

connected servers, i.e.,

$$R(s) = \max_{c \in C} \{d(c, n(c, S)) + d(n(c, S), s)\}.$$

In essence, $R(s)$ represents the longest possible latency for server s to receive an operation issued by a client, and $L(s)$ represents the longest possible latency for s to deliver a state update to its connected clients. The maximum interaction path length among all clients can be rewritten as

$$\begin{aligned} & \max_{c, c' \in C} \{d(c, n(c, S)) + d(n(c, S), n(c', S)) + d(n(c', S), c')\} \\ &= \max_{c \in C} \left\{ d(c, n(c, S)) + \max_{c' \in C} \{d(n(c, S), n(c', S)) \right. \\ & \quad \left. + d(n(c', S), c')\} \right\} \\ &= \max_{s \in S} \left\{ \max_{n(c, S)=s} \{d(c, s)\} \right. \\ & \quad \left. + \max_{c' \in C} \{d(s, n(c', S)) + d(n(c', S), c')\} \right\} \\ &= \max_{s \in S} \{L(s) + R(s)\}. \end{aligned}$$

The M-GREEDY algorithm makes use of the above decomposition in computing the maximum interaction path length (line 11). In our algorithm, the local and remote latencies of all servers are incrementally updated across iterations. In addition, the algorithm also maintains, for each client c , its nearest server $n(c)$ under the current server placement S ; for each selected server location $s \in S$, the set of clients $\mathbb{C}(s)$ whose nearest servers are s under placement S ; and for each unselected candidate server location $z \in Z \setminus S$, the set of clients $\mathbb{C}(z)$ whose nearest servers would become z if z is added to the current placement S . Initially, $n(c) = \emptyset$ for all clients (line 6) and $\mathbb{C}(z) = C$ (the whole client set) for all candidate server locations (line 4). Since the clients always connect to their nearest servers, if a candidate server location s^* is added to the current placement S , for each client $c \in \mathbb{C}(s^*)$, c shall no longer connect to its current server $n(c)$ and is thus removed from $\mathbb{C}(n(c))$ (lines 19-20). Meanwhile, for each remaining candidate location $z \in Z \setminus S$, all the clients c satisfying

```

updateLR( $z$ )
1:  $\Delta S \leftarrow \{n(c) \mid c \in \mathbb{C}(z)\};$ 
2: for all  $s \in S \setminus \Delta S$  do
3:    $L(s) \leftarrow L_p(s);$ 
4: for all  $s \in \Delta S$  do
5:    $L(s) \leftarrow \max_{c \in \mathbb{C}(s) \setminus \mathbb{C}(z)} d(c, s);$ 
6:   if  $L(s) = L_p(s)$  then
7:      $\Delta S \leftarrow \Delta S \setminus \{s\};$ 
8:  $L(z) \leftarrow \max_{c \in \mathbb{C}(z)} d(c, z);$ 
9: for all  $s \in S$  do
10:  if  $d(s, z) + L(z) > R_p(s)$  then
11:     $R(s) \leftarrow d(s, z) + L(z); E(s) \leftarrow \{z\};$ 
12:  else if  $E_p(s) \setminus \Delta S \neq \emptyset$  then
13:     $R(s) \leftarrow R_p(s); E(s) \leftarrow E_p(s) \setminus \Delta S;$ 
14:    if  $d(s, z) + L(z) = R_p(s)$  then
15:       $E(s) \leftarrow E(s) \cup \{z\};$ 
16:  else
17:     $R(s) \leftarrow \max_{x \in S \cup \{z\}} \{d(s, x) + L(x)\};$ 
18:     $E(s) \leftarrow \{x \in S \cup \{z\} \mid d(s, x) + L(x) = R(s)\};$ 
19:  $R(z) \leftarrow \max_{x \in S \cup \{z\}} \{d(z, x) + L(x)\};$ 
20:  $E(z) \leftarrow \{x \in S \cup \{z\} \mid d(z, x) + L(x) = R(z)\};$ 
    
```

Algorithm 3: Update local and remote latencies

$d(c, z) > d(c, s^*)$ shall be removed from $\mathbb{C}(z)$ as it would no longer be possible for them to connect to z even if z is selected later (lines 21-24).

Algorithm 3 shows the subroutine for evaluating the change in the local and remote latencies caused by adding a new server location z . To update the local latencies, we start by finding the set of existing servers ΔS which have at least one client that will re-connect to z if z is selected as a server location (line 1). Only the local latencies of the servers in ΔS need to be recalculated (lines 4-5). Then, we remove from ΔS the servers whose local latencies are not changed (lines 6-7). So, ΔS eventually constitutes the servers whose local latencies change due to the addition of z . It is obvious that the local latencies of the servers in ΔS can only decrease after z is added. Line 8 calculates z 's own local latency. The update of remote latencies leverages the values of local latencies. Note that the remote latency $R(s)$ of each server s can be rewritten as

$$R(s) = \max_{x \in S} \{d(s, x) + L(x)\}.$$

For each server s , we also maintain the list of servers $E(s)$ which give rise to the maximum remote latency of s , i.e.,

$$E(s) = \{x \in S \mid d(s, x) + L(x) = R(s)\}.$$

Let $L_p(s)$, $R_p(s)$ and $E_p(s)$ be the information of each server s before z is added, and let $L(s)$ be the updated local latency of s after z is added. Since the local latencies of existing servers cannot increase due to the addition of a new server, after z is added, the remote latency of each existing server $s \in S$ is bounded by $\max\{d(s, z) + L(z), R_p(s)\}$. If $d(s, z) + L(z) > R_p(s)$, the $R(s)$ and $E(s)$ after adding z are simply given by $d(s, z) + L(z)$ and $\{z\}$ respectively (lines 10-11). Otherwise, if $E_p(s) \setminus \Delta S \neq \emptyset$, the remote latency of s remains unchanged after the addition of z (lines 12-13). In this case, z needs to be added to $E(s)$ if $d(s, z) + L(z) = R_p(s)$ (i.e., z also gives rise to the remote latency of s) (lines 14-15). If $E_p(s) \setminus \Delta S = \emptyset$, the remote latency of s is recalculated as $R(s) = \max_{x \in S \cup \{z\}} \{d(s, x) + L(x)\}$ (line 17). Finally, lines 19-20 calculate z 's own remote latency. The subroutine of Algorithm 3 has a time complexity of $O(|C| + k^2)$, where $O(|C|)$ is the total computational complexity of lines 4-5 for all selected server locations s , $O(k^2)$ is that of line 17 for all selected server locations s , and k is the number of iterations in Algorithm 2. Therefore, the total time complexity of the M-GREEDY algorithm is $O(k \cdot |Z| \cdot (|C| + k^2))$.

Theorem 15. *For networks with the triangle inequality, the M-GREEDY server placement has an approximation ratio of 2 for the server provisioning problem for continuous DIAs.*

Proof. When only one server is placed in the network, there is no inter-server latency involved in the interactions between clients. In this case, the longest interaction path is the one from the client furthest away from the server to itself.

Suppose that s^* is the first server location selected by the M-GREEDY algorithm. Let c_i be the client furthest away from s^* . Then, after selecting s^* , the maximum

interaction path length is $2 \cdot d(c_i, s^*)$. Denote by $o(c_i)$ the nearest server of c_i in an optimal server placement S_O . There must exist a client c_j (which can be the same as c_i) satisfying

$$d(c_j, o(c_i)) \geq d(c_i, s^*) = \max_{c \in C} d(c, s^*).$$

This is because otherwise, $o(c_i)$ would have been the first server location selected by the M-GREEDY algorithm. Denote by $o(c_j)$ the nearest server of c_j in the optimal server placement S_O . Let L_O be the maximum interaction path length in the optimal server placement. Then,

$$L_O \geq d(c_j, o(c_j)) + d(o(c_j), o(c_i)) + d(o(c_i), c_i).$$

By the triangle inequality, we have

$$d(c_j, o(c_j)) + d(o(c_j), o(c_i)) \geq d(c_j, o(c_i)).$$

Therefore,

$$\begin{aligned} L_O &\geq d(c_j, o(c_i)) + d(o(c_i), c_i) \\ &\geq d(c_j, o(c_i)). \end{aligned}$$

Thus,

$$2d(c_i, s^*) \leq 2d(c_j, o(c_i)) \leq 2 \cdot L_O.$$

That is, $2d(c_i, s^*)$ is within two times of the maximum interaction path length in the optimal server placement S_O . Since the maximum interaction path length finally produced by the M-GREEDY algorithm cannot be longer than that after selecting the first location s^* , it must also be within two times of that in S_O .

Hence, the theorem is proven. $\square$

The approximation ratio 2 of the M-GREEDY algorithm holds regardless of whether the choices of server locations are restricted and whether there is a limit on the number of server locations to select. The approximation ratio is also tight. Figure 4.6 shows a tight example which consists of 5 nodes $\{c_1, c_2, c_3, c_4, s\}$. The latency between c_1 and

Figure 4.6: The approximation ratio 2 of the M-GREEDY algorithm is tight.

c_4 as well as the latency between c_2 and c_3 are $(1 - \varepsilon)$, where $0 < \varepsilon < 1$. The latencies between the remaining node pairs out of $\{c_1, c_2, c_3, c_4\}$ are all $(1 + \varepsilon)$. Meanwhile, node s has latency 1 to all the other nodes. It is obvious that the network satisfies the triangle inequality. Suppose that a client is located at each of the nodes c_1 , c_2 , c_3 and c_4 , and all the nodes in the network are candidate server locations. Then, the first server location selected by the M-GREEDY algorithm is s , because the maximum interaction path length is 2 if all the clients are connected to s . If the first server is placed at any other location, the maximum interaction path length is $2 + 2\varepsilon > 2$. After s is selected, further placing a new server at any node c_i ($i = 1, 2, 3, 4$) would result in some client located at $c_j \neq c_i$ to connect to the server at c_i with latency $1 - \varepsilon$. As a result, the maximum interaction path length becomes $(1 - \varepsilon) + 1 + 1 = 3 - \varepsilon > 2$. Therefore, the M-GREEDY algorithm stops at the second iteration and outputs the server placement $\{s\}$. On the other hand, the optimal placement of this example is to place servers at nodes $\{c_1, c_2, c_3, c_4\}$. In this way, each client is connected to the server at the same location. Thus, the maximum interaction path length is $1 + \varepsilon$. When ε approaches 0, the M-GREEDY algorithm and the optimal server placement differ by a ratio of

$$\lim_{\varepsilon \rightarrow 0} \frac{2}{1 + \varepsilon} = 2$$

in the maximum interaction path length.

4.6 M-Better Server Placement

In this section, we propose an algorithm called M-BETTER that achieves an even better approximation ratio than the M-GREEDY server placement for our server provisioning problem. Given an instance of the server provisioning problem, the M-BETTER algorithm combines the NEAREST server placement S_N and the M-GREEDY server placement S_G to produce its own solution. Suppose that L_N and L_G are the maximum interaction path lengths under server placements S_N and S_G respectively. The M-BETTER algorithm selects the candidate server locations in S_N if $L_N < L_G$, and selects the candidate server locations in S_G otherwise. In this way, the M-BETTER server placement produces a maximum interaction path length $L_B = \min\{L_N, L_G\}$. Since the M-GREEDY server placement has higher computational complexity than the NEAREST server placement, the time complexity of the M-BETTER algorithm is the same as that of the M-GREEDY algorithm.

Theorem 16. *For networks with the triangle inequality, the M-BETTER server placement has an approximation ratio of $\frac{5}{3}$ for the server provisioning problem for continuous DIAs.*

Proof. Let C be the set of clients, and let Z be the set of candidate server locations in the network. Suppose that $S_O \subseteq Z$ is a set of optimal server locations for our server provisioning problem. For each client $c \in C$, denote by $o(c) \in S_O$ the nearest server of c under placement S_O .

Let S_N be the set of server locations selected by the NEAREST server placement. For each client $c \in C$, denote by $n(c) \in S_N$ the nearest server of c under placement S_N . Then, $n(c)$ is the nearest candidate server location to c . Let $r = \max_{c \in C} d(c, n(c))$ be the maximum latency from all the clients to their nearest servers under server placement S_N . Denote by L_N the maximum interaction path length under placement S_N . Suppose that L_N is produced by the interaction path between two clients c_x and

$c_y \in C$. According to the triangle inequality, we have

$$\begin{aligned}
 L_N &= d(c_x, n(c_x)) + d(n(c_x), n(c_y)) + d(n(c_y), c_y) \\
 &\leq d(c_x, n(c_x)) + \left(d(n(c_x), c_x) + d(c_x, c_y) + d(c_y, n(c_y)) \right) \\
 &\quad + d(n(c_y), c_y) \\
 &= 2d(c_x, n(c_x)) + d(c_x, c_y) + 2d(c_y, n(c_y)) \\
 &\leq 2r + d(c_x, c_y) + 2r \\
 &\leq 4r + \left(d(c_x, o(c_x)) + d(o(c_x), o(c_y)) + d(o(c_y), c_y) \right) \\
 &\leq 4r + L_O,
 \end{aligned} \tag{4.2}$$

where L_O is the maximum interaction path length under the optimal server placement S_O .

Next, consider the M-GREEDY server placement. Suppose that the first server location selected by the M-GREEDY algorithm is $s^* \in Z$. Let $t = \max_{c \in C} d(c, s^*)$ be the latency from s^* to the furthest client. Then, the maximum interaction path length given by connecting all the clients to s^* is $2t$. Note that the M-GREEDY algorithm can only improve the maximum interaction path length by selecting additional server locations. Thus, the maximum interaction path length L_G eventually produced by the M-GREEDY algorithm cannot be longer than $2t$, i.e.,

$$L_G \leq 2t. \tag{4.3}$$

Since the M-BETTER heuristic produces a maximum interaction path length $L_B = \min\{L_N, L_G\}$, it follows that

$$L_B \leq 2t. \tag{4.4}$$

By the design of the M-GREEDY algorithm, t is the lowest achievable value of the maximum latency from all the clients to any single server. Therefore, for the nearest server $o(c_i)$ of each client c_i under the optimal placement S_O , there must exist a client $c_j \in C$ (which can be the same as c_i) fulfilling $d(c_j, o(c_i)) \geq t$.

If there exists a client c_i such that $d(c_i, o(c_i)) \geq t$, then we have

$$L_O \geq 2d(c_i, o(c_i)) \geq 2t.$$

It follows from (4.4) that

$$L_O \geq L_B.$$

Thus, it must hold that $L_B = L_O$ since L_O is the lowest achievable value of the maximum interaction path length among all possible server placements. This implies that the M-BETTER heuristic produces an optimal server placement in this case.

Otherwise, if all the clients c_i fulfill $d(c_i, o(c_i)) < t$, then for each client c_i , there must exist another client c_j ($c_j \neq c_i$) satisfying $d(c_j, o(c_i)) \geq t$. Consider the client c_k that is furthest away from its nearest server under the optimal placement S_O , i.e.,

$$d(c_k, o(c_k)) = \max_{c \in C} d(c, o(c)).$$

Let c_h ($c_h \neq c_k$) be a client satisfying $d(c_h, o(c_k)) \geq t$. Then, by the definition of L_O and the triangle inequality, we have

$$\begin{aligned} L_O &\geq d(c_h, o(c_h)) + d(o(c_h), o(c_k)) + d(o(c_k), c_k) \\ &\geq d(c_h, o(c_k)) + d(o(c_k), c_k) \\ &\geq t + d(o(c_k), c_k). \end{aligned}$$

Note that for each client c , $n(c)$ is the nearest candidate server location to c . Thus, $d(c, o(c)) \geq d(c, n(c))$. It follows that

$$\begin{aligned} d(c_k, o(c_k)) &= \max_{c \in C} d(c, o(c)) \\ &\geq \max_{c \in C} d(c, n(c)) \\ &= r. \end{aligned}$$

Therefore,

$$L_O \geq t + r.$$

According to (4.2), we have

$$r \geq \frac{1}{4} \cdot (L_N - L_O).$$

It follows that

$$L_O \geq t + \frac{1}{4}L_N - \frac{1}{4}L_O.$$

Thus,

$$L_O \geq \frac{4}{5} \cdot \left(t + \frac{1}{4}L_N\right).$$

Based on (4.3), we have

$$\begin{aligned} L_O &\geq \frac{4}{5} \cdot \left(\frac{1}{2}L_G + \frac{1}{4}L_N\right) \\ &= \frac{1}{5} \cdot \left(2L_G + L_N\right). \end{aligned}$$

Since

$$L_G \geq \min \{L_N, L_G\}$$

and

$$L_N \geq \min \{L_N, L_G\},$$

we have

$$\begin{aligned} L_O &\geq \frac{1}{5} \cdot (2L_G + L_N) \\ &\geq \frac{1}{5} \cdot (2 \cdot \min \{L_N, L_G\} + \min \{L_N, L_G\}) \\ &= \frac{3}{5} \cdot \min \{L_N, L_G\}. \end{aligned}$$

Recall that the M-BETTER server placement produces a maximum interaction path length $L_B = \min \{L_G, L_N\}$. It follows that

$$L_O \geq \frac{3}{5} \cdot L_B.$$

Therefore,

$$L_B \leq \frac{5}{3} \cdot L_O.$$

Hence, the theorem is proven. $\square$

Figure 4.7: The approximation ratio $\frac{5}{3}$ of the M-BETTER heuristic is tight.

The approximation ratio $\frac{5}{3}$ of the M-BETTER heuristic is tight. Figure 4.7 illustrates a tight example. Due to space limitations, only the network latencies between some node pairs are shown in the figure. The latencies between unmarked node pairs are given by the lengths of their shortest paths. It is easy to verify that the network satisfies the triangle inequality when $0 < \varepsilon < 4$. Suppose that two clients are located at nodes c_1 and c_2 respectively, and the remaining nodes in the network are all candidate server locations. Since each client c_i ($i = 1, 2$) has a latency of $1 + (4 + \varepsilon) = 5 + \varepsilon > 5$ to node o_j ($j \neq i$) and a latency of $(1 - \frac{\varepsilon}{4}) + (8 + \frac{\varepsilon}{2}) = 9 + \frac{\varepsilon}{4} > 5$ to node m_j ($j \neq i$), the first server selected by the M-GREEDY algorithm is s^* . Connecting both clients to s^* gives a maximum interaction path length of $5 + 5 = 10$. If one node o_i or m_i ($i = 1, 2$) is further selected, client c_i would re-connect to the newly selected node, and this results in an interaction path length between c_1 and c_2 greater than 10. Therefore, the M-GREEDY algorithm terminates at the second iteration and produces the server placement $\{s^*\}$. On the other hand, the nearest candidate server locations of clients c_1 and c_2 are m_1 and m_2 respectively. So, the maximum interaction path length under the NEAREST server placement $\{m_1, m_2\}$ is

$$\left(1 - \frac{1}{4}\varepsilon\right) + \left(8 + \frac{1}{2}\varepsilon\right) + \left(1 - \frac{1}{4}\varepsilon\right) = 10.$$

Therefore, the M-BETTER heuristic also produces a maximum interaction path length of 10. In this example, the optimal server placement is to select nodes o_1 and o_2 as server locations. Under such server placement, clients c_1 and c_2 are connected to

servers o_1 and o_2 respectively. As a result, the maximum interaction path length is

$$1 + (4 + \varepsilon) + 1 = 6 + \varepsilon.$$

When ε approaches 0, the ratio between the maximum interaction path lengths produced by the M-BETTER heuristic and the optimal server placement is

$$\lim_{\varepsilon \rightarrow 0} \frac{10}{6 + \varepsilon} = \frac{5}{3}.$$

4.7 Analysis of Server Placement in Networks with α -Triangle Inequality

Due to the restrictions of the Internet structure and the routing policies of ISPs [56,60], the triangle inequality does not always hold in the Internet. As discussed in Section 3.7 of Chapter 3, we can generalize the triangle inequality to the α -triangle inequality to capture the extent of triangle inequality violations. In this section, we analyze the performance of the server placement algorithms under the α -triangle inequality.

4.7.1 k -Center Server Placement

We first analyze the performance of the k -center server placement for networks with the α -triangle inequality. Suppose that the number of server locations to select for our server provisioning problem is capped by k , then we have the following approximability result of the k -center server placement.

Theorem 17. *For networks satisfying the α -triangle inequality ($\alpha \geq 1$), the k -center server placement has an approximation ratio of $(\frac{3}{4}\alpha^3 + \frac{5}{4}\alpha^2 + 1)$ for the server provisioning problem for continuous DIAs with a limit k on the number of server locations to select.*

Proof. Let C be the set of clients. For each client $c \in C$, denote by $o(c)$ and $r(c)$ the nearest servers of c under the optimal server placement and the k -center server

placement respectively. Then, for any two clients c_i and $c_j \in C$, their interaction path length under the k -center server placement is given by

$$l_R(c_i, c_j) = d(c_i, r(c_i)) + d(r(c_i), r(c_j)) + d(r(c_j), c_j).$$

By the α -triangle inequality, we have

$$\begin{aligned} l_R(c_i, c_j) &\leq d(c_i, r(c_i)) + \alpha \cdot \left(d(r(c_i), o(c_i)) \right. \\ &\quad \left. + d(o(c_i), r(c_j)) \right) + d(r(c_j), c_j) \\ &= d(c_i, r(c_i)) + \alpha \cdot d(r(c_i), o(c_i)) \\ &\quad + \alpha \cdot d(o(c_i), r(c_j)) + d(r(c_j), c_j) \\ &\leq d(c_i, r(c_i)) + \alpha^2 \cdot \left(d(r(c_i), c_i) + d(c_i, o(c_i)) \right) \\ &\quad + \alpha \cdot d(o(c_i), r(c_j)) + d(r(c_j), c_j) \\ &= (\alpha^2 + 1) \cdot d(c_i, r(c_i)) + \alpha^2 \cdot d(c_i, o(c_i)) \\ &\quad + \alpha \cdot d(o(c_i), r(c_j)) + d(c_j, r(c_j)). \end{aligned} \tag{4.5}$$

By the α -triangle inequality, we also have

$$\begin{aligned} d(o(c_i), r(c_j)) &\leq \alpha \cdot \left(d(o(c_i), o(c_j)) + d(o(c_j), r(c_j)) \right) \\ &\leq \alpha \cdot \left(d(o(c_i), o(c_j)) + \alpha \cdot (d(o(c_j), c_j) + d(c_j, r(c_j))) \right) \\ &= \alpha \cdot d(o(c_i), o(c_j)) + \alpha^2 \cdot d(o(c_j), c_j) + \alpha^2 \cdot d(c_j, r(c_j)), \end{aligned}$$

and

$$\begin{aligned} d(o(c_i), r(c_j)) &\leq \alpha \cdot \left(d(o(c_i), c_j) + d(c_j, r(c_j)) \right) \\ &\leq \alpha \cdot \left(\alpha \cdot (d(o(c_i), o(c_j)) + d(o(c_j), c_j)) + d(c_j, r(c_j)) \right) \\ &= \alpha^2 \cdot d(o(c_i), o(c_j)) + \alpha^2 \cdot d(o(c_j), c_j) + \alpha \cdot d(c_j, r(c_j)). \end{aligned}$$

Thus, it follows from (4.5) that

$$l_R(c_i, c_j) \leq (\alpha^2 + 1) \cdot d(c_i, r(c_i)) + \alpha^2 \cdot d(c_i, o(c_i))$$

$$\begin{aligned}
 & +\alpha^2 \cdot d(o(c_i), o(c_j)) + \alpha^3 \cdot d(o(c_j), c_j) \\
 & +\alpha^3 \cdot d(c_j, r(c_j)) + d(c_j, r(c_j)) \\
 = & (\alpha^2 + 1) \cdot d(c_i, r(c_i)) + \alpha^2 \cdot d(c_i, o(c_i)) \\
 & +\alpha^2 \cdot d(o(c_i), o(c_j)) + \alpha^3 \cdot d(o(c_j), c_j) \\
 & +(\alpha^3 + 1) \cdot d(c_j, r(c_j)), \tag{4.6}
 \end{aligned}$$

and

$$\begin{aligned}
 l_R(c_i, c_j) \leq & (\alpha^2 + 1) \cdot d(c_i, r(c_i)) + \alpha^2 \cdot d(c_i, o(c_i)) \\
 & +\alpha^3 \cdot d(o(c_i), o(c_j)) + \alpha^3 \cdot d(o(c_j), c_j) \\
 & +\alpha^2 \cdot d(c_j, r(c_j)) + d(c_j, r(c_j)) \\
 = & (\alpha^2 + 1) \cdot d(c_i, r(c_i)) + \alpha^2 \cdot d(c_i, o(c_i)) \\
 & +\alpha^3 \cdot d(o(c_i), o(c_j)) + \alpha^3 \cdot d(o(c_j), c_j) \\
 & +(\alpha^2 + 1) \cdot d(c_j, r(c_j)). \tag{4.7}
 \end{aligned}$$

By symmetry, we also obtain

$$\begin{aligned}
 l_R(c_i, c_j) \leq & (\alpha^3 + 1) \cdot d(c_i, r(c_i)) + \alpha^3 \cdot d(c_i, o(c_i)) \\
 & +\alpha^2 \cdot d(o(c_i), o(c_j)) + \alpha^2 \cdot d(o(c_j), c_j) \\
 & +(\alpha^2 + 1) \cdot d(c_j, r(c_j)), \tag{4.8}
 \end{aligned}$$

and

$$\begin{aligned}
 l_R(c_i, c_j) \leq & (\alpha^2 + 1) \cdot d(c_i, r(c_i)) + \alpha^3 \cdot d(c_i, o(c_i)) \\
 & +\alpha^3 \cdot d(o(c_i), o(c_j)) + \alpha^2 \cdot d(o(c_j), c_j) \\
 & +(\alpha^2 + 1) \cdot d(c_j, r(c_j)). \tag{4.9}
 \end{aligned}$$

Taking the average of (4.6) and (4.8), we have

$$l_R(c_i, c_j) \leq \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2 + 1 \right) \cdot d(c_i, r(c_i)) + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2 \right) \cdot d(c_i, o(c_i))$$

$$\begin{aligned}
 & +\alpha^2 \cdot d(o(c_i), o(c_j)) + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2\right) \cdot d(o(c_j), c_j) \\
 & + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2 + 1\right) \cdot d(c_j, r(c_j)).
 \end{aligned} \tag{4.10}$$

Similarly, taking the average of (4.7) and (4.9), we have

$$\begin{aligned}
 l_R(c_i, c_j) & \leq (\alpha^2 + 1) \cdot d(c_i, r(c_i)) + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2\right) \cdot d(c_i, o(c_i)) \\
 & + \alpha^3 \cdot d(o(c_i), o(c_j)) + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2\right) \cdot d(o(c_j), c_j) \\
 & + (\alpha^2 + 1) \cdot d(c_j, r(c_j)).
 \end{aligned} \tag{4.11}$$

Now, taking the average of (4.10) and (4.11), we obtain

$$\begin{aligned}
 l_R(c_i, c_j) & \leq \left(\frac{1}{4}\alpha^3 + \frac{3}{4}\alpha^2 + 1\right) \cdot d(c_i, r(c_i)) + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2\right) \cdot d(c_i, o(c_i)) \\
 & + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2\right) \cdot d(o(c_i), o(c_j)) + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2\right) \cdot d(o(c_j), c_j) \\
 & + \left(\frac{1}{4}\alpha^3 + \frac{3}{4}\alpha^2 + 1\right) \cdot d(c_j, r(c_j)).
 \end{aligned}$$

Let $m = \max_{c \in C} d(c, r(c))$ be the maximum latency from the clients to their nearest servers under the k -center server placement. It follows that

$$\begin{aligned}
 l_R(c_i, c_j) & \leq 2 \cdot \left(\frac{1}{4}\alpha^3 + \frac{3}{4}\alpha^2 + 1\right)m \\
 & + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2\right) \cdot \left(d(c_i, o(c_i)) + d(o(c_i), o(c_j)) + d(o(c_j), c_j)\right).
 \end{aligned}$$

By the definition of the k -center placement, m is the minimum achievable value of the maximum latency from the clients to their nearest servers among all possible placements of up to k servers. Thus, there must exist a client $c_x \in C$ fulfilling $d(c_x, o(c_x)) \geq m$ under the optimal server placement. Therefore, we have

$$\begin{aligned}
 l_R(c_i, c_j) & \leq \left(\frac{1}{4}\alpha^3 + \frac{3}{4}\alpha^2 + 1\right) \cdot 2d(c_x, o(c_x)) \\
 & + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2\right) \cdot \left(d(c_i, o(c_i)) + d(o(c_i), o(c_j)) + d(o(c_j), c_j)\right) \\
 & = \left(\frac{1}{4}\alpha^3 + \frac{3}{4}\alpha^2 + 1\right) \cdot \left(d(c_x, o(c_x)) + d(o(c_x), o(c_x))\right. \\
 & \quad \left.+ d(o(c_x), c_x)\right) + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2\right) \cdot \left(d(c_i, o(c_i))\right. \\
 & \quad \left.+ d(o(c_i), o(c_j)) + d(o(c_j), c_j)\right)
 \end{aligned}$$

$$\begin{aligned}
 &\leq \left(\left(\frac{1}{4}\alpha^3 + \frac{3}{4}\alpha^2 + 1 \right) + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2 \right) \right) \cdot L_O \\
 &= \left(\frac{3}{4}\alpha^3 + \frac{5}{4}\alpha^2 + 1 \right) \cdot L_O,
 \end{aligned} \tag{4.12}$$

where L_O denotes the maximum interaction path length under the optimal server placement. As a result, the maximum interaction path length L_R produced by the k -center server placement satisfies

$$\begin{aligned}
 L_R &= \max_{c_i, c_j \in C} l_R(c_i, c_j) \\
 &\leq \left(\frac{3}{4}\alpha^3 + \frac{5}{4}\alpha^2 + 1 \right) \cdot L_O.
 \end{aligned}$$

Hence, the theorem is proven. $\square$

The proof of Theorem 17 also results in the following proposition for the server provisioning problem when there is no limit on the number of server locations to select.

Corollary 4. *For each instance of the server provisioning problem for continuous DIAs on a network satisfying the α -triangle inequality, if an optimal server placement selects k server locations, then the k -center server placement produces a maximum interaction path length within $\left(\frac{3}{4}\alpha^3 + \frac{5}{4}\alpha^2 + 1 \right)$ times of that in the optimal server placement.*

4.7.2 Nearest Server Placement

In this section, we study the performance of the NEAREST server placement for networks with the α -triangle inequality.

Theorem 18. *For networks satisfying the α -triangle inequality ($\alpha \geq 1$), the NEAREST server placement has an approximation ratio of $\max \left\{ \frac{1}{2}\alpha^3 + \frac{3}{2}\alpha^2 + 1, \frac{2}{3}\alpha^3 + \alpha^2 + \frac{2}{3} \right\}$ for the server provisioning problem for continuous DIAs.*

Proof. Let C be the set of clients and Z be the set of candidate server locations. Denote by S_N and $n(c)$ respectively as the NEAREST server placement and the nearest

server of each client $c \in C$ under placement S_N . Under the NEAREST server placement S_N , the interaction path length between any two clients c_i and $c_j \in C$ is given by

$$l_N(c_i, c_j) = d(c_i, n(c_i)) + d(n(c_i), n(c_j)) + d(n(c_j), c_j).$$

Let $o(c)$ be the nearest server of each client c under the optimal server placement S_O .

Similar to the derivations of (4.10) and (4.11) in Theorem 17, we have

$$\begin{aligned} l_N(c_i, c_j) &\leq \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2 + 1\right) \cdot d(c_i, n(c_i)) + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2\right) \cdot d(c_i, o(c_i)) \\ &\quad + \alpha^2 \cdot d(o(c_i), o(c_j)) + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2\right) \cdot d(o(c_j), c_j) \\ &\quad + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2 + 1\right) \cdot d(c_j, n(c_j)), \end{aligned} \quad (4.13)$$

and

$$\begin{aligned} l_N(c_i, c_j) &\leq (\alpha^2 + 1) \cdot d(c_i, n(c_i)) + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2\right) \cdot d(c_i, o(c_i)) \\ &\quad + \alpha^3 \cdot d(o(c_i), o(c_j)) + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2\right) \cdot d(o(c_j), c_j) \\ &\quad + (\alpha^2 + 1) \cdot d(c_j, n(c_j)). \end{aligned} \quad (4.14)$$

By the definition of the NEAREST server placement, for each client c , $n(c)$ is the nearest candidate server location to c among all candidate server locations in Z . Thus,

$$d(c_i, n(c_i)) \leq d(c_i, o(c_i)),$$

and

$$d(c_j, n(c_j)) \leq d(c_j, o(c_j)).$$

It follows from (4.13) and (4.14) that

$$\begin{aligned} l_N(c_i, c_j) &\leq \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2 + 1\right) \cdot d(c_i, o(c_i)) + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2\right) \cdot d(c_i, o(c_i)) \\ &\quad + \alpha^2 \cdot d(o(c_i), o(c_j)) + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2\right) \cdot d(o(c_j), c_j) \\ &\quad + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2 + 1\right) \cdot d(c_j, o(c_j)) \\ &= (\alpha^3 + \alpha^2 + 1) \cdot d(c_i, o(c_i)) + \alpha^2 \cdot d(o(c_i), o(c_j)) \end{aligned}$$

$$+(\alpha^3 + \alpha^2 + 1) \cdot d(c_j, o(c_j)), \quad (4.15)$$

and

$$\begin{aligned} l_N(c_i, c_j) &\leq (\alpha^2 + 1) \cdot d(c_i, o(c_i)) + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2\right) \cdot d(c_i, o(c_i)) \\ &\quad + \alpha^3 \cdot d(o(c_i), o(c_j)) + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2\right) \cdot d(o(c_j), c_j) \\ &\quad + (\alpha^2 + 1) \cdot d(c_j, o(c_j)) \\ &= \left(\frac{1}{2}\alpha^3 + \frac{3}{2}\alpha^2 + 1\right) \cdot d(c_i, o(c_i)) + \alpha^3 \cdot d(o(c_i), o(c_j)) \\ &\quad + \left(\frac{1}{2}\alpha^3 + \frac{3}{2}\alpha^2 + 1\right) \cdot d(o(c_j), c_j). \end{aligned} \quad (4.16)$$

We define a new variable

$$x = \frac{2 \cdot d(o(c_i), o(c_j))}{d(c_i, o(c_i)) + d(c_j, o(c_j))}.$$

Then, (4.15) and (4.16) can be rewritten as

$$l_N(c_i, c_j) \leq \left(\alpha^3 + \alpha^2 + 1 + \frac{1}{2}\alpha^2 x\right) \cdot \left(d(c_i, o(c_i)) + d(c_j, o(c_j))\right), \quad (4.17)$$

and

$$l_N(c_i, c_j) \leq \left(\frac{1}{2}\alpha^3 + \frac{3}{2}\alpha^2 + 1 + \frac{1}{2}\alpha^3 x\right) \cdot \left(d(c_i, o(c_i)) + d(c_j, o(c_j))\right). \quad (4.18)$$

Let L_O be the maximum interaction path length under the optimal server placement S_O . We have

$$\begin{aligned} L_O &\geq d(c_i, o(c_i)) + d(o(c_i), o(c_j)) + d(o(c_j), c_j) \\ &= \left(1 + \frac{1}{2}x\right) \cdot \left(d(c_i, o(c_i)) + d(c_j, o(c_j))\right). \end{aligned} \quad (4.19)$$

Combining (4.17), (4.18) and (4.19), we obtain

$$\begin{aligned} l_N(c_i, c_j) &\leq \frac{\min\left\{\alpha^3 + \alpha^2 + 1 + \frac{1}{2}\alpha^2 x, \frac{1}{2}\alpha^3 + \frac{3}{2}\alpha^2 + 1 + \frac{1}{2}\alpha^3 x\right\}}{1 + \frac{1}{2}x} \cdot L_O \\ &= \frac{2 + 2\alpha^2 + \alpha^3 + \alpha^2 \cdot \min\{x + \alpha, \alpha x + 1\}}{2 + x} \cdot L_O. \end{aligned}$$

The coefficient on the right side of the above inequality is the same as that in Theorem 6 of Chapter 3 (i.e., (3.13) and (3.14)). Following the derivation in Theorem 6, the maximum possible value of $\frac{2 + 2\alpha^2 + \alpha^3 + \alpha^2 \cdot \min\{x + \alpha, \alpha x + 1\}}{2 + x}$ is given by

$$\max \left\{ \frac{1}{2}\alpha^3 + \frac{3}{2}\alpha^2 + 1, \frac{2}{3}\alpha^3 + \alpha^2 + \frac{2}{3} \right\}.$$

Thus,

$$l_N(c_i, c_j) \leq \max \left\{ \frac{1}{2}\alpha^3 + \frac{3}{2}\alpha^2 + 1, \frac{2}{3}\alpha^3 + \alpha^2 + \frac{2}{3} \right\} \cdot L_O.$$

As a result, the maximum interaction path length L_N under the NEAREST server placement fulfills

$$\begin{aligned} L_N &= \max_{c_i, c_j \in C} l_N(c_i, c_j) \\ &\leq \max \left\{ \frac{1}{2}\alpha^3 + \frac{3}{2}\alpha^2 + 1, \frac{2}{3}\alpha^3 + \alpha^2 + \frac{2}{3} \right\} \cdot L_O. \end{aligned}$$

Hence, the theorem is proven. $\square$

The examples of Figures 3.5 and 3.6 in Chapter 3 are also applicable to showing the tightness of the above approximation ratio. In these examples, the NEAREST server placement would place servers at all the nodes in V_M , and the optimal server placement for minimizing the maximum interaction path length is to place servers at all the nodes in V_O . Thus, these examples show the tightness of the approximation ratio derived in Theorem 18 as well.

4.7.3 M-Greedy Server Placement

For networks with the α -triangle inequality, the approximation ratios of the k -center server placement and the NEAREST server placement both grow cubically with α . In this section, we show that the M-GREEDY algorithm proposed in Section 4.5 outperforms these two server placements by achieving an approximation ratio that grows linearly with α .

Theorem 19. *For networks satisfying the α -triangle inequality ($\alpha \geq 1$), the M-GREEDY server placement has an approximation ratio of 2α for the server provisioning problem for continuous DIAs.*

Proof. When only one server is placed in the network, the longest interaction path is the one from the client furthest away from this server to itself. Let s^* be the first server location selected by the M-GREEDY algorithm, and let c_i be the client furthest away from s^* . Then, after selecting s^* , the maximum interaction path length is $2 \cdot d(c_i, s^*)$. Denote by $o(c_i)$ the nearest server of c_i in the optimal server placement. There must exist a client c_j satisfying

$$d(c_j, o(c_i)) \geq d(c_i, s^*) = \max_{c \in C} d(c, s^*),$$

since otherwise, $o(c_i)$ would have been the first server location selected by the M-GREEDY algorithm. Denote by $o(c_j)$ the nearest server of c_j in the optimal server placement. Let L_O be the maximum interaction path length in the optimal server placement. Then, by the α -triangle inequality, we have

$$\begin{aligned} \alpha \cdot L_O &\geq \alpha \cdot (d(c_j, o(c_j)) + d(o(c_j), o(c_i)) + d(o(c_i), c_i)) \\ &\geq \alpha \cdot (d(c_j, o(c_j)) + d(o(c_j), o(c_i))) \\ &\geq d(c_j, o(c_i)). \end{aligned}$$

Thus,

$$2d(c_i, s^*) \leq 2d(c_j, o(c_i)) \leq 2\alpha \cdot L_O.$$

Since the maximum interaction path length finally produced by the M-GREEDY algorithm cannot be longer than that after selecting the first location s^* , it must also be within $2\alpha \cdot L_O$. Hence, the theorem is proven. $\square$

The approximation ratio 2α of the M-GREEDY algorithm is tight. Figure 4.8 shows a tight example which consists of 5 nodes $\{c_1, c_2, s_1, s_2, s\}$. The latency between c_1 and s_1 as well as the latency between c_2 and s_2 are ε , where $0 < \varepsilon < \alpha - \frac{1}{2}$. The

Figure 4.8: The approximation ratio 2α of the M-GREEDY algorithm is tight.

latency between s_1 and s_2 is 1, and that between c_1 and c_2 is α . The latency between c_1 and s_2 , and the latency between c_2 and s_1 are both $\alpha + \varepsilon$. Meanwhile, node s has latency α to all the other nodes. It is easy to verify that the network satisfies the α -triangle inequality. Suppose that two clients are located at nodes c_1 and c_2 respectively, and nodes s_1 , s_2 and s are candidate server locations. Then, the first server location selected by the M-GREEDY algorithm is s , because the maximum interaction path length is 2α if both clients are connected to s . If the first server is placed at either s_1 or s_2 , the maximum interaction path length is $2\alpha + 2\varepsilon > 2\alpha$. After s is selected, further placing a new server at either node s_i ($i = 1, 2$) would result in the client located at c_i to connect to the server at s_i with latency ε . As a result, the maximum interaction path length becomes $\alpha + \alpha + \varepsilon = 2\alpha + \varepsilon > 2\alpha$. Therefore, the M-GREEDY algorithm stops at the second iteration and outputs the server placement $\{s\}$. On the other hand, the optimal placement of this example is to place servers at nodes $\{s_1, s_2\}$. In this way, each client c_i ($i = 1, 2$) is connected to the server at s_i . Thus, the maximum interaction path length is $\varepsilon + 1 + \varepsilon = 1 + 2\varepsilon < 2\alpha$. When ε approaches 0, the M-GREEDY algorithm and the optimal server placement differ in the maximum interaction path length by a ratio of

$$\lim_{\varepsilon \rightarrow 0} \frac{2\alpha}{1 + 2\varepsilon} = 2\alpha.$$

The approximation ratio 2α of the M-GREEDY server placement holds regardless of whether the choices of server locations are restricted and any limit on the number of server locations to select.

4.7.4 M-Better Server Placement

In this section, we analyze the approximation ratio of the M-BETTER server placement for networks with the α -triangle inequality.

Theorem 20. *For networks satisfying the α -triangle inequality ($\alpha \geq 1$), the M-BETTER server placement has an approximation ratio of $2\alpha \cdot \min \left\{ 1, \frac{2\alpha^3 + 4\alpha^2 + 4}{\alpha^3 + 3\alpha^2 + 4\alpha + 4} \right\}$ for the server provisioning problem for continuous DIAs.*

Proof. Let C be the set of clients and Z be the set of candidate server locations. Suppose that $S_O \subseteq Z$ is a set of optimal server locations for the server provisioning problem. For each client $c \in C$, denote by $o(c) \in S_O$ the nearest server of c under placement S_O .

Let S_N be the set of server locations selected by the NEAREST server placement. For each client $c \in C$, denote by $n(c)$ the nearest server to c in S_N . Let $r = \max_{c \in C} d(c, n(c))$ be the maximum latency from all the clients to their nearest servers under server placement S_N . Denote by L_N the maximum interaction path length under placement S_N . Suppose that L_N is produced by the interaction path between two clients c_x and $c_y \in C$. Similar to the derivations of (4.13) and (4.14) in Theorem 18, we have

$$\begin{aligned}
 L_N &\leq \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2 + 1 \right) \cdot d(c_x, n(c_x)) + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2 \right) \cdot d(c_x, o(c_x)) \\
 &\quad + \alpha^2 \cdot d(o(c_x), o(c_y)) + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2 \right) \cdot d(o(c_y), c_y) \\
 &\quad + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2 + 1 \right) \cdot d(c_y, n(c_y)) \\
 &\leq (\alpha^3 + \alpha^2 + 2) \cdot r + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2 \right) \cdot d(c_x, o(c_x)) \\
 &\quad + \alpha^2 \cdot d(o(c_x), o(c_y)) + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2 \right) \cdot d(o(c_y), c_y), \tag{4.20}
 \end{aligned}$$

and

$$L_N \leq (\alpha^2 + 1) \cdot d(c_x, n(c_x)) + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2 \right) \cdot d(c_x, o(c_x))$$

$$\begin{aligned}
 & +\alpha^3 \cdot d(o(c_x), o(c_y)) + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2\right) \cdot d(o(c_y), c_y) \\
 & +(\alpha^2 + 1) \cdot d(c_y, n(c_y)) \\
 & \leq (2\alpha^2 + 2) \cdot r + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2\right) \cdot d(c_x, o(c_x)) \\
 & +\alpha^3 \cdot d(o(c_x), o(c_y)) + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2\right) \cdot d(o(c_y), c_y). \tag{4.21}
 \end{aligned}$$

Taking the average of (4.20) and (4.21), we obtain

$$\begin{aligned}
 L_N & \leq \left(\frac{1}{2}\alpha^3 + \frac{3}{2}\alpha^2 + 2\right) \cdot r + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2\right) \cdot d(c_x, o(c_x)) \\
 & + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2\right) \cdot d(o(c_x), o(c_y)) + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2\right) \cdot d(o(c_y), c_y) \\
 & = \left(\frac{1}{2}\alpha^3 + \frac{3}{2}\alpha^2 + 2\right) \cdot r + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2\right) \cdot \left(d(c_x, o(c_x))\right. \\
 & \quad \left.+ d(o(c_x), o(c_y)) + d(o(c_y), c_y)\right) \\
 & \leq \left(\frac{1}{2}\alpha^3 + \frac{3}{2}\alpha^2 + 2\right) \cdot r + \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2\right) \cdot L_O, \tag{4.22}
 \end{aligned}$$

where L_O is the maximum interaction path length under the optimal server placement S_O .

Now, consider the M-GREEDY server placement. Suppose that the first server location selected by M-GREEDY is $s^* \in Z$. Let $t = \max_{c \in C} d(c, s^*)$ be the latency from s^* to the furthest client. Let L_G and L_B be the maximum interaction path lengths produced by the M-GREEDY and M-BETTER algorithms respectively. Following the same reasoning behind (4.3) and (4.4) in Theorem 16, we have

$$L_G \leq 2t, \tag{4.23}$$

and

$$L_B \leq 2t. \tag{4.24}$$

The design of the M-GREEDY algorithm implies that t is the lowest achievable value of the maximum latency from all the clients to any single server. Therefore, for the nearest server $o(c_i)$ of each client c_i under the optimal placement S_O , there must exist a client $c_j \in C$ satisfying $d(c_j, o(c_i)) \geq t$.

If there exists a client c_i such that $d(c_i, o(c_i)) \geq t$, then we have

$$L_O \geq 2d(c_i, o(c_i)) \geq 2t \geq L_B.$$

In this case, it follows that $L_B = L_O$ since S_O is an optimal server placement.

Otherwise, if all the clients c_i satisfy $d(c_i, o(c_i)) < t$, then for each client c_i , there must exist another client c_j ($c_j \neq c_i$) satisfying $d(c_j, o(c_i)) \geq t$. Consider the client c_k that is furthest away from its nearest server under the optimal placement S_O , i.e.,

$$d(c_k, o(c_k)) = \max_{c \in C} d(c, o(c)).$$

Let c_h ($c_h \neq c_k$) be a client satisfying $d(c_h, o(c_k)) \geq t$. Then, by the definition of L_O and the α -triangle inequality, we have

$$\begin{aligned} L_O &\geq d(c_h, o(c_h)) + d(o(c_h), o(c_k)) + d(o(c_k), c_k) \\ &\geq \frac{1}{\alpha} \cdot d(c_h, o(c_k)) + d(o(c_k), c_k) \\ &\geq \frac{1}{\alpha} \cdot t + d(o(c_k), c_k). \end{aligned}$$

Note that for each client c , $n(c)$ is the nearest candidate server location to c . Thus, $d(c, o(c)) \geq d(c, n(c))$. It follows that

$$\begin{aligned} d(c_k, o(c_k)) &= \max_{c \in C} d(c, o(c)) \\ &\geq \max_{c \in C} d(c, n(c)) \\ &= r. \end{aligned}$$

Therefore,

$$L_O \geq \frac{1}{\alpha} \cdot t + r.$$

According to (4.22) and (4.23), we have

$$r \geq \frac{L_N - \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2\right) \cdot L_O}{\frac{1}{2}\alpha^3 + \frac{3}{2}\alpha^2 + 2},$$

and

$$t \geq \frac{1}{2} \cdot L_G.$$

It follows that

$$L_O \geq \frac{1}{2\alpha} \cdot L_G + \frac{L_N - \left(\frac{1}{2}\alpha^3 + \frac{1}{2}\alpha^2\right) \cdot L_O}{\frac{1}{2}\alpha^3 + \frac{3}{2}\alpha^2 + 2}.$$

Thus,

$$\begin{aligned} (\alpha^3 + 2\alpha^2 + 2) \cdot L_O &\geq \left(\frac{1}{4}\alpha^2 + \frac{3}{4}\alpha + \frac{1}{\alpha}\right) \cdot L_G + L_N \\ &\geq \left(\frac{1}{4}\alpha^2 + \frac{3}{4}\alpha + \frac{1}{\alpha} + 1\right) \cdot \min\{L_N, L_G\} \\ &= \left(\frac{1}{4}\alpha^2 + \frac{3}{4}\alpha + \frac{1}{\alpha} + 1\right) \cdot L_B. \end{aligned}$$

Therefore, we have

$$\begin{aligned} L_B &\leq \frac{\alpha^3 + 2\alpha^2 + 2}{\frac{1}{4}\alpha^2 + \frac{3}{4}\alpha + \frac{1}{\alpha} + 1} \cdot L_O \\ &= 2\alpha \cdot \frac{2\alpha^3 + 4\alpha^2 + 4}{\alpha^3 + 3\alpha^2 + 4\alpha + 4} \cdot L_O. \end{aligned}$$

Recall that the M-GREEDY placement produces a maximum interaction path length L_G within $2\alpha \cdot L_O$ (Theorem 19). So, we have

$$\begin{aligned} L_B &\leq \min \left\{ 2\alpha, 2\alpha \cdot \frac{2\alpha^3 + 4\alpha^2 + 4}{\alpha^3 + 3\alpha^2 + 4\alpha + 4} \right\} \cdot L_O \\ &= 2\alpha \cdot \min \left\{ 1, \frac{2\alpha^3 + 4\alpha^2 + 4}{\alpha^3 + 3\alpha^2 + 4\alpha + 4} \right\} \cdot L_O. \end{aligned}$$

Hence, the theorem is proven. $\square$

Figure 4.9 illustrates the approximation ratios of the NEAREST, M-GREEDY and M-BETTER algorithms as a function of α for networks satisfying the α -triangle inequality ($\alpha \geq 1$). It can be seen from Figure 4.9 that the M-BETTER server placement achieves a better approximation ratio than the M-GREEDY server placement when $\alpha \leq \frac{\sqrt{17}-1}{2} \approx 1.56$. They are both more resilient to the extent of triangle inequality violations than the NEAREST server placement.

4.8 Experimental Evaluation

In this section, we evaluate the four server placements with the Meridian dataset [61], which is a real Internet latency dataset collected using the King technology [57]. The

Figure 4.9: Approximation ratios of different server placement algorithms for continuous DIAs under the α -triangle inequality

dataset provides the measurements of pairwise network latencies among 2500 nodes. Since some measurements are not available, using the same method as in Section 3.8 of Chapter 3, we discard those nodes involved in the unavailable measurements and keep the remaining 1796 nodes that form a complete matrix of pairwise latencies.

We compare the four server placements introduced in Sections 4.3 to 4.6. To quantify the performance difference of the algorithms, we normalize their maximum interaction path lengths by a theoretical lower bound. Given a set of candidate server locations Z , and a set of clients C , a lower bound on the maximum interaction path length is given by

$$\max_{c_i, c_j \in C} \left\{ \min_{s_a, s_b \in Z} \{d(c_i, s_a) + d(s_a, s_b) + d(s_b, c_j)\} \right\}. \quad (4.25)$$

The above bound makes two relaxations. First, a server is assumed to be available at every candidate server location. Second, each client can connect to different servers for interacting with different clients. Thus, this bound is a super-optimum and may not really be achievable. We refer to the normalized maximum interaction path length as the *normalized interactivity*.

The NP-hardness of the k -center problem indicates that it is difficult to find the exact k -center server placement. For comparison purpose, we implement a k -center heuristic [46] in our experiments. Starting from an empty placement, the k -center heuristic iteratively selects a new server location that leads to the largest reduction in the maximum latency from the clients to their nearest servers until exactly k servers are placed. In addition to the four server placements presented earlier, we also implement a k -favourable heuristic, which has been introduced in Section 3.8 of Chapter 3. The k -favourable heuristic evaluates the popularity of each candidate server location by the frequency of its involvement in the shortest possible interaction paths among all client pairs. Then, the k -favourable heuristic selects the k most popular candidate server locations.

4.8.1 Performance Comparison for Different Algorithms

Our experiments assume that the clients and candidate server locations are separate. For each simulation run, we randomly select a total of 896 nodes and assume that a client is located at each of these nodes. Then, 75, 150, 300, 600 or 900 nodes are randomly selected from the remaining nodes as the candidate server locations. For each setting, we perform 1000 simulation runs using different sets of clients and candidate server locations. The results of various simulation runs are normalized by different lower bound values derived using the respective sets of candidate server locations.

We first study the maximum interaction path lengths of different algorithms as a function of the limit on the number of server locations to select. The number of candidate server locations is set to 300 in this experiment. Figure 4.10 shows the average results of 1000 simulation runs for the k -center, k -favourable and M-GREEDY server placements. Since the NEAREST algorithm and the M-BETTER algorithm do not consider any limitation on the number of server locations to select, they are not included

Figure 4.10: Normalized interactivity of different algorithms when there are 300 candidate server locations

in this experiment. Note that the first server location selected by our M-GREEDY algorithm constitutes the 1-center placement. Starting from the 1-center placement, the M-GREEDY algorithm considerably reduces the maximum interaction path length by adding more server locations. This implies that there is significant potential for improving the interactivity performance by distributing the servers hosting continuous DIAs. However, increasing the number of server locations does not guarantee to enhance the interactivity unless the server locations are carefully chosen. For example, it can be seen from Figure 4.10 that the performance of the k -center placement deteriorates when the number of server locations selected become large. This is because the k -center placement does not take into account the inter-server latencies, which may increase substantially and outweigh the reduction in the latencies from clients to servers when the server locations are more widely dispersed. By considering both the client-to-server latencies and the inter-server latencies together, the k -favourable heuristic is able to cut the maximum interaction path length as more server locations are selected. However, its performance is still far worse than the M-GREEDY algorithm. The best achievable result of the M-GREEDY algorithm is 1.16, indicating that it can produce near-optimal server placements.

Figure 4.11: Normalized interactivity of different algorithms for different numbers of candidate server locations.

Next, we study the impact of the number of candidate server locations. We remove the limit on the number of server locations to select and run our M-GREEDY algorithm until it terminates due to possible performance deterioration if more servers are placed. We record the exact number of server locations selected in each simulation run. We then use the recorded number as the number of server locations to pick for the k -center and k -favourable server placements in the same simulation run. This leads to exactly the same number of server locations selected by these three algorithms for fair comparison. The NEAREST and M-BETTER server placements do not have any stipulation on the number of server locations to select. Figure 4.11 shows the average results of 1000 simulation runs. It can be seen that the interactivity performance achieved by the M-BETTER server placement is very close to that of the M-GREEDY server placement. This is because our M-GREEDY algorithm produces better solutions than the NEAREST server placement in most of the simulation runs. The performance improvement of the M-GREEDY and M-BETTER server placements relative to the other three placements increases with the number of candidate server locations. This implies that M-GREEDY and M-BETTER are effective in finding good server placements even when the search space is large. It can also be seen from Fig-

ure 4.11 that the performance of the NEAREST server placement deteriorates sharply with increasing number of candidate server locations. This indicates that aggressively shortening the client-to-server latencies cannot contribute effectively to improving the interactivity performance of continuous DIAs.

Figure 4.12 shows the cumulative distribution of the normalized interactivity for the 1000 simulation runs of different numbers of candidate server locations. For ease of reading, the M-BETTER server placement is not shown in the figure since it produces almost the same curve as the M-GREEDY server placement. When there are only 75 candidate server locations (Figure 4.12(a)), our M-GREEDY algorithm finds the optimal server placement in over 30% of the simulation runs (a normalized interactivity of 1 implies that the output placement must be optimal), whereas the k -center placement finds it in just 15% of the runs and the other two placements find even less. Our detailed investigation indicates that most of the simulation runs in which the k -center placement finds the optimum have only one server location selected in the optimal placement. In this case, the M-GREEDY algorithm outputs the same placement as 1-center. Since the 1-center placement is optimal in only a small portion of all simulation runs, this further suggests the necessity of distributed server placement to achieve better interactivity for continuous DIAs. Similar performance trends are also observed in the results for other numbers of candidate server locations (Figures 4.12(b) to 4.12(e)). As the number of candidate server locations increases, it becomes harder to hit a good solution due to larger search space. Figure 4.12 also shows that the improvement of our M-GREEDY algorithm is even greater in the worst case scenarios. For example, when there are 300 candidate server locations (Figure 4.12(c)), the 95th percentile results of the M-GREEDY, k -center, k -favourable and NEAREST server placements are 1.42, 1.86, 3.86 and 6.32 respectively.

We further compare the interactivity performance of different algorithms by normalizing the maximum interaction path lengths using the same lower bound value

Figure 4.12: Cumulative distribution of normalized interactivity for 1000 simulation runs

Figure 4.13: Normalized interactivity of different algorithms using the same lower bound value for each client set

for each client set. The lower bound value is calculated by formula (4.25), setting Z as the set of all 900 nodes that are not selected to place clients. Figure 4.13 shows the results with such normalization. It can be seen from the figure that our proposed M-GREEDY and M-BETTER algorithms outperform the other algorithms irrespective of the numbers of candidate server locations. The interactivity of M-GREEDY and M-BETTER generally improves with increasing number of candidate server locations.

Table 4.2: Average running times of different algorithms for continuous DIAs (in seconds)

Number of candidate server locations	NEAREST	k -favourable	k -center	M-GREEDY	M-BETTER
75	0.026s	0.289s	0.085s	0.101s	0.127s
150	0.028s	0.670s	0.110s	0.137s	0.165s
300	0.029s	2.123s	0.152s	0.176s	0.205s
600	0.030s	10.382s	0.269s	0.377s	0.407s
900	0.031s	18.301s	0.277s	0.434s	0.465s

Table 4.2 shows the average running times of the algorithms for different numbers of candidate server locations. Similar to Section 3.8.1, these experiments are run on a machine with Intel Xeon 3.2GHz CPU and 16GB RAM. It can be seen that the running times of our M-GREEDY algorithm and M-BETTER algorithm are similar to the k -center placement, and are much shorter than the k -favourable algorithm. This

Figure 4.14: Normalized interactivity of different algorithms for different individual server capacities.

indicates that our proposed algorithms are efficient and their running times are very acceptable.

4.8.2 Impact of Server Capacity

So far, we have not assumed any capacity limitation of the servers. Now, we study the impact of server capacity. Suppose that there are a certain number of servers to be allocated to the locations selected by the algorithms. Each server has a capacity limit on the number of clients that can connect to it. In this case, the algorithms for server placement can be executed first to select server locations. Next, each selected server location is allocated the number of servers roughly proportional to the number of clients having it as their nearest server locations. Then, the capacity of each location is given by the total capacity of the servers allocated to it. If the nearest server location of a client has been filled to its capacity, the client has to turn to the next nearest server location with spare capacity.

First, we assume that the total server capacity is 900, which is just enough to accommodate the 896 clients in our simulation. Figure 4.14 shows the performance results for different server capacities and numbers for the simulation runs of 300

Figure 4.15: Normalized interactivity of different algorithms for different total server capacities.

candidate server locations. We test 36, 18, 12, 9, 6, 4 and 3 servers each with the capacity of 25, 50, 75, 100, 150, 225 and 300 respectively. It can be seen that the M-GREEDY and M-BETTER algorithms consistently outperform the other algorithms for all the cases tested. We can observe from Figure 4.14 that the performance of each algorithm slightly deteriorates with the increase of individual server capacity. This is because the actual number of locations with servers deployed cannot exceed the number of servers available. When the individual server capacity increases, the number of servers needed to support all the clients decreases, which implicitly limits the number of server locations used.

Figure 4.15 shows the impact of the total available server capacity. In this experiment, the capacity of each server is fixed at 50 and we vary the number of servers available for allocation. It can be seen that the M-GREEDY and M-BETTER algorithms achieve better interactivity performance than the other algorithms. In general, for all the algorithms, the interactivity improves when the number of servers available increases. This is because a larger number of servers increase the capacity of each server location and thus allows more clients to connect to their preferred nearest server locations. When there are ample capacities allocated to every selected server lo-

cation, the performance of each algorithm converges to that when there is no capacity limitation at the server locations.

4.8.3 Impact of Dynamics in Network Conditions

The network latency in the real Internet may fluctuate due to the mutability of network conditions. In this section, we study the effect of network latency variation on the performance of different server placements. Since the Meridian dataset does not contain the measurements of network latencies at different times, we again use the PlanetLab All Pairs Pings dataset [62] in this experiment. We have performed simulation runs with the PlanetLab datasets of different days and observed similar performance trends. We present here the results for a sample dataset of 4 June 2005. This dataset contains 96 pairwise latency measurements among 195 PlanetLab nodes at intervals of 15 minutes over a one-day time span. Therefore, we simulate a network of 195 nodes and emulate the dynamics in the network latencies among the nodes using the data of these 96 measurements.

Similar to Section 3.8.3 in Chapter 3, in each simulation run, we randomly choose 80 nodes as the candidate server locations, and assume that a client is placed at each of the remaining 115 nodes. Each server placement is computed based on the network latencies recorded in the first measurement, and remains unchanged throughout the simulation run. We use two mechanisms for connections between clients and servers. The first mechanism is to connect each client to its nearest server as indicated by the network latencies recorded in the first measurement and keep the connection unchanged regardless of the variation of network latency later. The other mechanism is to ensure that each client always connects to its nearest server based on the latest latency measurement, i.e., the connected server of each client changes with the variation of network latency. The theoretical lower bound on the maximum interaction path length is recalculated at each latency measurement for computing the normalized

(a) The connected servers of clients are fixed.

(b) The connected servers of clients change along with the variation of network latency.

Figure 4.16: Cumulative distribution of normalized interactivity by different algorithms under dynamic network latency

interactivity. We perform 1000 different simulation runs and plot the cumulative distribution of the normalized interactivity at the times of the 96 latency measurements in Figure 4.16. Since the M-GREEDY server placement outperforms the NEAREST server placement in most of the simulation runs, the performance of the M-BETTER server placement is close to that of M-GREEDY. For ease of reading, the curve of M-BETTER is not shown in the figure. It can be seen that the M-GREEDY server placement achieves much better interactivity than the other three server placements, for both mechanisms of client connections. Figure 4.16 also indicates that the M-GREEDY server placement is more resilient to the variation of network latency. For example, if the client connections are fixed, the M-GREEDY server placement achieves a normalized interactivity within 1.5 in 92% of the time (see Figure 4.16(a)). In contrast, the k -center, k -favourable and NEAREST server placements achieve it in only 79%, 69% and 68% of the time respectively. A similar trend can also be observed in Figure 4.16(b).

By comparing Figures 4.16(a) and 4.16(b), it can be seen that the interactivity performance of each server placement can be slightly improved by adjusting the connections from clients to servers according to the change of network latency. However, the improvement achieved by carefully choosing the locations of servers (e.g., utilizing the M-GREEDY server placement) is much more significant. This emphasizes the importance of server placement in improving the interactivity performance of continuous DIAs.

4.9 Summary

In this chapter, we have investigated the server provisioning problem for optimizing the interactivity performance of continuous DIAs with joint considerations of their consistency and fairness requirements. We have shown that this is a challenging problem by analyzing its hardness under various conditions. We have proved that

the problem cannot be approximated within any constant factor for networks without the triangle inequality; within a factor of $3/2$ if the choices of server locations are restricted; and within a factor of $4/3$ if there is a limit on the number of server locations to select. From the computability perspective, the mutual interaction feature makes server provisioning for continuous DIAs much more challenging than that for web content delivery. With the client-to-server latency as the sole optimization objective, placing more servers in the network can only improve the web access performance. Nevertheless, deploying more servers in DIAs does not necessarily imply shorter network latency in the interaction between clients. We have presented and analyzed four server provisioning algorithms with progressively reduced approximation ratios. Experiments with real latency data of the Internet show that the M-GREEDY and M-BETTER algorithms produce near-optimal server placements.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

The presence of network latencies poses a challenge to the development of distributed interactive applications. The interaction between the clients of DIAs involves not only the network latencies between clients and servers but also the network latencies among the servers. In this thesis, we have identified the placement of the servers hosting DIAs as an essential factor that significantly affects the network latencies involved in the interaction between clients. We have formulated the server provisioning problems as combinational optimization problems for both discrete DIAs and continuous DIAs, with the goals of optimizing the interactivity performance while maintaining other Quality of Service requirements.

Chapter 3 has focused on the server provisioning problem for discrete DIAs. We have proved the NP-hardness of the server provisioning problem under any one of the following three conditions that may be common in practice: (i) the network latency does not satisfy the triangle inequality; or (ii) the choices of server locations in the network are restricted; or (iii) the number of server locations to select is limited. Then, we have presented four different algorithms for server placement and shown a variety of theoretical results on the approximation ratios of these algorithms. First, we have analyzed the classical k -median server placement which is considered closely related to our server provisioning problem. Second, we have presented a simple NEAREST

server placement that achieves the same approximation ratio as that of the k -median server placement. The NEAREST server placement is easier to compute than the k -median server placement in that it can be acquired in polynomial time while it is NP-hard to compute the k -median server placement. After that, we have proposed our T-GREEDY and T-BETTER algorithms to further improve the approximation ratios. We have studied both networks satisfying the triangle inequality and networks with triangle inequality violations. In particular, we have characterized the extent of triangle inequality violations in the network using the α -triangle inequality property. We have shown that, for discrete DIAs, our proposed T-GREEDY and T-BETTER algorithms achieve approximation ratios that grow linearly with α , which are significantly better than those of the k -median and NEAREST server placements that grow cubically with α . This implies that T-GREEDY and T-BETTER are more resilient to the triangle inequality violations of the network.

Chapter 4 is devoted to the study of the server provisioning problem for continuous DIAs. We have first identified that the difference between continuous DIAs and discrete DIAs is that the application state also changes along with the passing of time in continuous DIAs. Because of this, maintaining the consistency and fairness requirements is much tougher in continuous DIAs than in discrete DIAs. We have investigated the server provisioning problem so as to optimize the interactivity performance, with joint considerations of the consistency and fairness requirements. We have shown the NP-hardness of the problem under various conditions, and have further proved that the server provisioning problem for continuous DIAs cannot be approximated within any constant factor when the network latencies do not satisfy the triangle inequality, within a factor of $\frac{3}{2}$ when the locations where servers can be placed are restricted, and within a factor of $\frac{4}{3}$ when the number of server locations to select is limited. Four server placement algorithms, namely, the k -center, NEAREST, M-GREEDY and M-BETTER server placements, are then presented and

theoretically analyzed. Similar to the analysis in Chapter 3, we have shown that our proposed M-GREEDY and M-BETTER server placements significantly outperform the k -center and NEAREST server placements in terms of approximation ratio, regardless of whether the networks satisfy the triangle inequality or not. It is also worth noting that, for networks with the triangle inequality, our proposed M-BETTER algorithm achieves an approximation ratio of $\frac{5}{3}$, which is rather close to the aforementioned non-approximability results. This implies that M-BETTER is a near optimal algorithm.

For each server provisioning problem, we have conducted experimental evaluations for the server placement algorithms using real Internet latency data. We have demonstrated the necessity of distributed server placements to optimize the interactivity performance of DIAs. However, simply increasing the number of servers does not guarantee better interactivity performance. The locations where the servers are placed must be carefully chosen. We have shown that aggressively reducing the latencies between clients and servers may not contribute effectively to enhancing the interactivity performance. The experimental results have indicated that our proposed algorithms significantly outperform the baseline server placements, and they are also more resilient to the dynamics of network latencies.

5.2 Future Work

While we have investigated the server provisioning problems for both discrete DIAs and continuous DIAs in this thesis, many studies can be further carried out to extend our work. Before closing this thesis, we propose two research directions for further exploration.

5.2.1 Multicast Server Provisioning Problem

The first direction we could consider is the multicast server provisioning problem whose purpose is to figure out the placement of servers to support large scale DIAs

with multicast delivery of state updates. The problem is described as follows. Suppose that each candidate server location has a finite capacity in terms of network bandwidth such that if it is selected to place servers, the maximum number of clients that can connect to it is limited. Due to the budget limitation, only a certain number of server locations can be selected by the DIA operators to host their applications. In this case, if the total number of clients of the DIA exceeds the maximum allowance, then there is no feasible solution for server placement due to the lack of available network bandwidth. More specifically, let C denote the set of clients, Z denote the set of candidate server locations, and k denote the cap on the number of server locations to select. Suppose that each candidate server location $s_i \in Z$ allows a maximum of $m(s_i)$ clients to connect to it if it is selected to place a server. Without loss of generality, assume that $m(s_i) \geq m(s_j)$ for any $i < j$. Then, if $\sum_{i=1}^k m(s_i) < |C|$, any server placement would exceed the total bandwidth capacity of server locations and there is no feasible solution for server placement.

We could overcome this issue by using multicast delivery of state updates from servers to clients. Assume that a set of server locations are already selected in the network. Different from traditional client assignment strategies that establish direct connections between clients and their connecting servers, in the multicast delivery strategy, each server location s constructs a multicast tree rooted at s . Clients assigned to the servers at s join the multicast tree of s , i.e., if a client c is assigned to a server at s , then c is added to the multicast tree of s . The number of clients that are directly connected to s in the multicast tree is bounded by $m(s)$ so that s would not exceed its capacity limitation. All the message transmissions between clients and servers are routed along the multicast trees. Each client receives the state update message generated by its assigned server through its parent node in the multicast tree, and then it forwards the received message to its children. In this way, each server s does not have to directly broadcast the state update messages to all clients assigned to it.

Instead, it sends only a maximum of $m(s)$ copies of messages. Therefore, the capacity limitations of the server locations could be satisfied and it is suitable to support large scale DIAs where there are a huge number of clients in the network. The multicast trees should be constructed in such a way that the total or maximum interaction path length could be minimized.

The purpose of the multicast server provisioning problem is therefore to select a set of server locations $S \subseteq Z$ to place servers ($|S| \leq k$) such that they are applicable to achieving the minimum value of the total or maximum interaction path length when the multicast delivery strategy is used. A significant difference between the two server provisioning problems discussed in this thesis and the multicast server provisioning problem is that in the latter problem, the path lengths from a server to the clients assigned to it are not fixed and are dependent on how the multicast trees are constructed. This implies that the multicast server provisioning problem is much more difficult than the two server provisioning problems discussed in this thesis. In our future work, we would like to first design an efficient multicast delivery strategy, and then based on it, we would like to propose an efficient algorithm to solve the multicast server provisioning problem.

5.2.2 Client Assignment Problem for Long Tail Interactions in DIAs

Suppose that there are $|C|$ clients in the network, then all the $|C|^2$ client pairs in the DIA are involved in the interaction. Given a server placement, we can expect that the interaction path lengths for most of the client pairs rest at a moderate level. It is no surprise that the interactions for a small portion of client pairs (referred to as *Long Tail Interactions* onwards) generate interaction path lengths far beyond the average level regardless of the server placement. Note that the maximum interaction path length is produced by the longest interaction path among the Long Tail Interactions.

As discussed in Chapter 4, the minimum achievable interaction time between clients for fulfilling the consistency and fairness requirements is determined by the maximum interaction path length in continuous DIAs. Therefore, the Long Tail Interactions have significant impact on the interactivity performance of continuous DIAs.

Given the server placement, Zhang and Tang have shown in their recent work [50] that connecting clients to appropriate servers instead of their nearest servers shortens the maximum interaction path. However, the effect of their approaches on reducing the maximum interaction path length is weakened since they do not consider the influence of the Long Tail Interactions. Inspired by [50], in our further work, we would like to measure the impact of the Long Tail Interactions using mathematical models and experimental evaluations, and design a client assignment algorithm that aims at optimizing the interactivity performance of continuous DIAs by taking into consideration the Long Tail Interactions.

Bibliography

- [1] M. Claypool and K. Claypool. Latency Can Kill: Precision and Deadline in Online Games. In *Proc. ACM MMSys*, pages 215–222, 2010.
- [2] R. B. Jennings, E. M. Nahum, D. P. Olshefski, D. Saha, Z.-Y. Shae and C. Waters. A Study of Internet Instant Messaging and Chat Protocols. *IEEE Network*, 20(4):16–21, 2006.
- [3] Agustina, F. Liu, S. Xia, H. Shen and C. Sun. CoMaya: Incorporating Advanced Collaboration Capabilities into 3D Digital Media Design Tools. In *Proc. ACM CSCW*, pages 5–8, 2008.
- [4] L. Ahmad, A. Boukerche, A. Al Hamidi, A. Shadid and R. Pazzi. Web-Based e-Learning in 3D Large Scale Distributed Interactive Simulations using HLA/RTI. In *Proc. IEEE IPDPS*, pages 1–4, 2008.
- [5] F. Safaei, P. Boustead, C. D. Nguyen, J. Brun and M. Dowlatshahi. Latency-Driven Distribution: Infrastructure Needs of Participatory Entertainment Applications. *IEEE Communications Magazine*, 43(5):106–112, 2005.
- [6] M. Mauve, J. Vogel, V. Hilt and W. Effelsberg. Local-Lag and Timewarp: Providing Consistency for Replicated Continuous Applications. *IEEE Transactions on Multimedia*, 6(1):47–57, 2004.

- [7] T. Beigbeder, R. Coughlan, C. Lusher, J. Plunkett, E. Agu and M. Claypool. The Effects of Loss and Latency on User Performance in Unreal Tournament 2003. In *Proc. ACM NetGames*, pages 144–151, 2004.
- [8] V. Valancius, N. Laoutaris, L. Massoulié, C. Diot and P. Rodriguez. Greening the Internet with Nano Data Centers. In *Proc. ACM CoNEXT*, pages 37–48, 2009.
- [9] S. Choy, B. Wong, G. Simon and C. Rosenberg. The Brewing Storm in Cloud Gaming: A Measurement Study on Cloud to End-user Latency. In *Proc. ACM NetGames*, pages 1–6, 2012.
- [10] D. Delaney, T. Ward and S. McLoone. On Consistency and Network Latency in Distributed Interactive Applications: A survey-Part I. *Presence: Teleoperators and Virtual Environments*, 15(2):218–234, 2006.
- [11] J. Brun, F. Safaei and P. Boustead. Managing Latency and Fairness in Networked Games. *Communications of the ACM*, 49(11):46–51, 2006.
- [12] Y.-J. Lin, K. Guo and S. Paul. Sync-MS: Synchronized Messaging Service for Real-Time Multi-Player Distributed Games. In *Proc. IEEE ICNP*, pages 155–164, 2002.
- [13] C. Greenhalgh, J. Purbrick and D. Snowdon. Inside MASSIVE-3: Flexible Support for Data Consistency and World Structuring. In *Proc. ACM CVE*, pages 119–127. ACM, 2000.
- [14] E. Frécon and M. Stenius. DIVE: A Scaleable Network Architecture for Distributed Virtual Environments. *Distributed Systems Engineering*, 5:91–100, 1998.

- [15] Y. Kawahara, T. Aoyama and H. Morikawa. A Peer-to-Peer Message Exchange Scheme for Large-Scale Networked Virtual Environments. *Telecommunication Systems*, 25(3):353–370, 2004.
- [16] M. R. Macedomia, M. J. Zyda, D. R. Pratt, D. P. Brutzman and P. T. Barham. Exploiting Reality with Multicast Groups: a Network Architecture for Large-Scale Virtual Environments. In *Proc. IEEE VRAIS*, pages 2–10, 1995.
- [17] E. S. Liu and G. K. Theodoropoulos. A Continuous Matching Algorithm for Interest Management in Distributed Virtual Environments. In *Proc. IEEE Workshop on PADS*, pages 1–10, 2010.
- [18] K. L. Morse, L. Bic and M. Dillencourt. Interest Management in Large-Scale Virtual Environments. *Presence: Teleoperators and Virtual Environments*, 9(1):52–68, 2000.
- [19] A. Yu and S. T. Vuong. MOPAR: a Mobile Peer-to-Peer Overlay Architecture for Interest Management of Massively Multiplayer Online Games. In *Proc. ACM Workshop on NOSSDAV*, pages 99–104, 2005.
- [20] P. Morillo, W. Moncho, J. Orduña and J. Duato. *Providing Full Awareness to Distributed Virtual Environments Based on Peer-to-Peer Architectures Advances in Computer Graphics*, volume 4035 of *Lecture Notes in Computer Science*, pages 336–347. Springer Berlin / Heidelberg, 2006.
- [21] S.-Y. Hu, J.-F. Chen and T.-H. Chen. VON: a Scalable Peer-to-Peer Network for Virtual Environments. *IEEE Network*, 20(4):22–31, 2006.
- [22] F. Aurenhammer. Voronoi Diagrams - a Survey of a Fundamental Geometric Data Structure. *ACM Computing Surveys*, 23(3):345–405, 1991.

- [23] A. Bharambe, J. R. Douceur, J. R. Lorch, T. Moscibroda, J. Pang, S. Seshan and X. Zhuang. Donnybrook: Enabling Large-Scale, High-Speed, Peer-to-Peer Games. In *Proc. ACM SIGCOMM conference on Data communication*, pages 389–400, 2008.
- [24] A. R. Bharambe, S. Rao and S. Seshan. Mercury: A Scalable Publish-Subscribe System for Internet Games. In *Proc. ACM NetGames*, pages 3–9, 2002.
- [25] T. Iimura, H. Hazeyama and Y. Kadobayashi. Zoned Federation of Game Servers: A Peer-to-Peer Approach to Scalable Multi-Player Online Games. In *Proc. ACM NetGames*, pages 116–120, 2004.
- [26] E. Cronin, B. Filstrup and A. Kurc. A Distributed Multiplayer Game Server System. Technical Report. University of Michigan, 2001.
- [27] E. Cronin, A. R. Kurc, B. Filstrup and S. Jamin. An Efficient Synchronization Mechanism for Mirrored Game Architectures. *Multimedia Tools and Applications*, 23(1):7–30, 2004.
- [28] T. A. Funkhouser. RING: a Client-Server System for Multi-User Virtual Environments. In *Proc. ACM symposium on Interactive 3D Graphics*, pages 85–92, 1995.
- [29] M. Mauve, S. Fischer and J. Widmer. A Generic Proxy System for Networked Computer Games. In *Proc. ACM NetGames*, pages 25–28, 2002.
- [30] S. D. Webb, S. Soh and W. Lau. Enhanced Mirrored Servers for Network Games. In *Proc. ACM NetGames*, pages 117–122, 2007.
- [31] L. Gautier, C. Diot and J. Kurose. End-to-End Transmission Control Mechanisms for Multiparty Interactive Applications on the Internet. In *Proc. IEEE INFOCOM*, pages 1470–1479, 1999.

- [32] L. Pantel and L. C. Wolf. On the Suitability of Dead Reckoning Schemes for Games. In *Proc. ACM NetGames*, pages 79–84, 2002.
- [33] M. Mauve. Consistency in Replicated Continuous Interactive Media. In *Proc. ACM CSCW*, pages 181–190, 2000.
- [34] M. Assiotis and V. Tzanov. A Distributed Architecture for MMORPG. In *Proc. ACM NetGames*. ACM, 2006.
- [35] B. Yang and H. Garcia-Molina. Designing a super-peer network. In *Proc. IEEE ICDE*, pages 49–60, 2003.
- [36] S.-H. Min, J. Holliday and D.-S. Cho. Optimal Super-Peer Selection for Large-Scale P2P System. In *Proc. IEEE ICHIT*, pages 588–593, 2006.
- [37] J. Brun, F. Safaei and P. Boustead. Server Topology Considerations in Online Games. In *Proc. ACM NetGames*. ACM, 2006.
- [38] D. N. B. Ta and S. Zhou. Server Placement for Enhancing the Interactivity of Large-Scale Distributed Virtual Environments. In *Proc. IEEE CW*, pages 123–132, 2006.
- [39] D. N. B. Ta, T. Nguyen, S. Zhou, X. Tang, W. Cai and R. Ayani. Interactivity-Constrained Server Provisioning in Large-Scale Distributed Virtual Environments. *IEEE Transactions on Parallel and Distributed Systems*, 23(2):304–312, 2012.
- [40] K.-W. Lee, B.-J. Ko and S. Calo. Adaptive Server Selection for Large Scale Interactive Online Games. *Computer Networks*, 49(1):84–102, 2005.
- [41] Y.-R. Chen, S. Radhakrishnan, S. K. Dhall and S. Karabuk. Server Selection with Delay Constraints for Online Games. In *Proc. IEEE Workshop on GLOBECOM*, pages 882–887, 2010.

- [42] N. Laoutaris, G. Smaragdakis, K. Oikonomou, I. Stavrakakis and A. Bestavros. Distributed Placement of Service Facilities in Large-Scale Networks. In *Proc. IEEE INFOCOM*, pages 2144–2152, 2007.
- [43] L. D. Briceno, H. J. Siegel, A. A. Maciejewski, H. Ye, B. Lock, M. N. Teli, F. Wedyan, C. Panaccione and Z. Chen. Resource Allocation in a Client/Server Hybrid Network for Virtual World Environments. In *Proc. IEEE. IPDPS*, pages 1–13, 2008.
- [44] M. Kwok. *Performance Analysis of Distributed Virtual Environments*. PhD thesis, University of Waterloo, 2006.
- [45] L. Qiu, V. N. Padmanabhan and G. M. Voelker. On the Placement of Web Server Replicas. In *Proc. IEEE INFOCOM*, pages 1587–1596, 2001.
- [46] E. Cronin, S. Jamin, J. Cheng, A. R. Kurc, D. Raz and Y. Shavitt. Constrained Mirror Placement on the Internet. *IEEE Journal on Selected Areas in Communications*, 20(7):1369–1382, 2002.
- [47] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.
- [48] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala and V. Pandit. Local Search Heuristic for k-Median and Facility Location Problems. In *Proc. ACM STOC*, pages 21–29, 2001.
- [49] M. Chrobak, C. Kenyon and N. Young. The Reverse Greedy Algorithm for the Metric k-Median Problem. *Information Processing Letters*, 97(2):68–72, 2006.
- [50] L. Zhang and X. Tang. The Client Assignment Problem for Continuous Distributed Interactive Applications. In *Proc. IEEE ICDCS*, pages 203–214, 2011.

- [51] L. Zhang and X. Tang. Client Assignment for Improving Interactivity in Distributed Interactive Applications. In *Proc. IEEE INFOCOM*, pages 3227–3235, 2011.
- [52] C. Ding, Y. Chen, T. Xu and X. Fu. CloudGPS: A Scalable and ISP-Friendly Server Selection Scheme in Cloud Computing Environments. In *Proc. IEEE/ACM IWQoS*, 2012.
- [53] J. C. S. Lui and M. F. Chan. An Efficient Partitioning Algorithm for Distributed Virtual Environment Systems. *IEEE Transactions on Parallel and Distributed Systems*, 13(3):193–211, 2002.
- [54] H. Zheng and X. Tang. On Server Provisioning for Distributed Interactive Applications. In *Proc. IEEE ICDCS*, pages 500–509, 2013.
- [55] L. Zhang and X. Tang. Optimizing Client Assignment for Enhancing Interactivity in Distributed Interactive Applications. *IEEE/ACM Transactions on Networking*, 20(6):1707–1720, 2012.
- [56] C. Lumezanu, R. Baden, N. Spring and B. Bhattacharjee. Triangle Inequality and Routing Policy Violations in the Internet. In *Proc. Passive and Active Measurement Conference*, pages 45–54, 2009.
- [57] K. P. Gummadi, S. Saroiu and S. D. Gribble. King: Estimating Latency between Arbitrary Internet End Hosts. In *Proc. ACM SIGCOMM Workshop on Internet Measurement*, pages 5–18, 2002.
- [58] F. Dabek, R. Cox, F. Kaashoek and R. Morris. Vivaldi: A Decentralized Network Coordinate System. In *Proc. ACM SIGCOMM*, pages 15–26, 2004.
- [59] M. Costa, M. Castro, A. Rowstron and P. Key. PIC: Practical Internet Coordinates for Distance Estimation. In *Proc. IEEE ICDCS*, pages 178–187, 2004.

- [60] H. Zheng, E. K. Lua, M. Pias and T. G. Griffin. Internet Routing Policies and Round-Trip-Times. In *Proc. Passive and Active Measurement Conference*, pages 236–250, 2005.
- [61] The Meridian Latency Data Set. <http://www.cs.cornell.edu/People/egs/meridian/>.
- [62] The PlanetLab All Pairs Pings Data Set. <http://pdos.csail.mit.edu/~strib/projects.html>.