Replica Placement Algorithms for Efficient Internet Content Delivery

by

Shihong Xu

Thesis submitted in fulfillment of the
requirements for the degree of

Doctor of Philosophy

in

Computer Science

School of Computer Science
The University of Adelaide, Australia
November 26th, 2009

To my parents and my wife Jie,

who made all of this possible

with their endless encouragement and patience.

Contents

Abstract vi

Statement of Originality viii

Publications ix

Acknowledgments xi

1 Introduction **1**

1.1 Content Delivery in the Internet 2

1.1.1 Content Delivery Systems 2

1.1.2 Evolution of Techniques 4

1.1.2.1 Single proxy caching 4

1.1.2.2 Cooperative web caching 5

1.1.2.3 Content Distribution Networks 6

1.2 Issues and Challenges 6

1.2.1 Research Issues in Content Delivery 7

1.2.2 Challenges 9

1.3 Our Work 11

1.3.1 Issues Addressed in This Thesis 11

1.3.2 Overview of Our Work 13

1.3.3 Summary of Contributions 15

1.4 Thesis Outline 16

2 Replication for Content Delivery — An Overview **18**

2.1 Content Distribution Networks 18

2.1.1 Replica Server Placement 19

2.1.2 Content Replication 20

2.1.3 Other Topics 22

2.2 Web Caching 22

2.2.1 Caching Architecture 22

2.2.2 Object Replacement Algorithms 24

2.2.3 Coordinated En-Route Web Caching 25

2.3 Approaches and Methods 26

2.3.1 Linear Programming and Duality Theory 26

2.3.2 Approximation Algorithms 30

2.3.3 Dynamic Programming 31

3 Fault Tolerant Facility Allocation **33**

3.1 Introduction 33

3.1.1 Related Work 36

3.1.2 Our Technique 37

3.1.3 Our Results 38

3.2 Single-Factor Approximation 39

3.2.1 The Algorithm 39

3.2.2 Inverse Dual Fitting Based Analysis 44

3.2.2.1 High level analysis 44

3.2.2.2 The *p-th* phase 47

3.2.2.3 Performance ratio 49

3.3 Bi-Factor Approximation and Extension 50

3.3.1 The Algorithm 50

3.3.2 Inverse Dual Fitting Based Analysis 52

3.3.3 Scaling and Greedy Augmentation 57

3.4 Fault-Tolerant k-Facility Allocation 58

3.4.1 Bisection Search and Combination 60

3.4.2 Randomized Procedure for Rounding 63

3.4.2.1 Facility opening 63

3.4.2.2 Connection establishment 63
3.4.2.3 Approximation factor 66
3.4.3 Derandomization 66
3.5 Discussion 67
3.5.1 Dealing with Demand 67
3.5.2 Fault-Tolerant Network Design 67

4 QoS-Aware Content Replication for Parallel Access 69

4.1 Introduction 69
4.2 Problem Formulation 72
4.2.1 Object Placement for Parallel Access 72
4.2.2 An Alternative Integer Programming Formulation 74
4.2.3 Issues in the FTFL Problem 76
4.2.3.1 Triangle inequality 76
4.2.3.2 Access frequency 76
4.3 A Distributed ($|\mathcal{R}|$,2)-Approximation Algorithm 77
4.3.1 Facility/Client Side Pseudocodes 77
4.3.2 An alternative centralized algorithm 79
4.3.3 An upper bound of performance ratio 81
4.4 Numerical Results 85
4.4.1 Results on Grids 85
4.4.1.1 Impact of connectivity 85
4.4.1.2 Impact of facility costs 86
4.4.2 Results on Network Models 87
4.4.2.1 Random graph 87
4.4.2.2 GT-ITM model 88
4.4.3 Factors of Performance Ratio 88
4.5 Related Work 89
4.6 Conclusion and Discussion 91

5 Coordinated En-Route Web Caching in Multi-Server Networks 93

5.1 Introduction 93

5.2 Related Work 95

5.3 System Model 98

5.3.1 Calculation of Caching Gain 99

5.3.2 Problem Formulation 102

5.4 Unconstrained p-Server CERC 103

5.4.1 Exhaustive Algorithm 104

5.4.2 Dynamic Programming Solution 104

5.4.3 Caching Scheme 109

5.4.4 Discussion 110

5.4.4.1 Optimization 110

5.4.4.2 Impact of Routing asymmetry 111

5.4.4.3 Server selection and cache consistency 111

5.5 QoS-Aware p-Server CERC 111

5.5.1 QoS Constraints 112

5.5.1.1 Individual Latency Constrained p-Server CERC . . . 113

5.5.1.2 Average Latency Constrained p-Server CERC 115

5.5.2 Extended Constraints on Copy Number 118

5.5.2.1 At most k copies 118

5.5.2.2 At least k copies 119

5.5.3 QoS-Aware Caching Schemes 119

5.6 Simulation Experiments 120

5.6.1 Simulation Model 120

5.6.1.1 Parameter settings 120

5.6.1.2 Caching schemes evaluated 121

5.6.2 Result of Experiments 122

5.6.2.1 Impact of cache size 122

5.6.2.2 Impact of access pattern 124

5.6.2.3 Impact of the number of cached copies 125

5.6.2.4 Impact of the number of servers 127

5.7 Concluding Remarks 128

6 Conclusion and Future Work 130

6.1 Conclusion 130

6.2 Future Work – Towards Smart Content Delivery 132

6.2.1 Novel Architectures 132

6.2.2 Security Considerations 133

6.2.3 Employment of Data Mining Techniques 133

Bibliography 135

Appendix A: List of Symbols 152

Appendix B: Curriculum Vitae 156

Abstract

This thesis covers three main issues in content delivery with a focus on placement algorithms of replica servers and replica contents. In a content delivery system, the location of replicas is very important as perceived by a quotation: *Closer is better*. However, considering the costs incurred by replication, it is a challenge to deploy replicas in a cost-effective manner. The objective of our work is to optimally select the location of replicas which includes sites for replica server deployment, servers for replica contents hosting, and en-route caches for object caching. Our solutions for corresponding applications are presented in three parts of the work, which makes significant contributions for designing scalable, reliable, and efficient systems for Internet content delivery.

In the first part, we define the *Fault-Tolerant Facility Allocation (FTFA)* problem for the placement of replica servers, which relaxes the well known *Fault-Tolerant Facility Location (FTFL)* problem by allowing an integer (instead of binary) number of facilities per site. We show that the problem is NP-hard even for the metric version, where connection costs satisfy the triangle inequality. We propose two efficient algorithms for the metric *FTFA* problem with approximation factors 1.81 and 1.61 respectively, where the second algorithm is also shown to be (1.11,1.78)- and (1,2)-approximation through the proposed inverse dual fitting technique. The first bi-factor approximation result is further used to achieve a 1.52-approximation algorithm and the second one a 4-approximation algorithm for the metric *Fault-Tolerant k-Facility Allocation* problem, where an upper bound of facility number (i. e. *k*) applies.

In the second part, we formulate the problem of QoS-aware content replication for parallel access in terms of combined download speed maximization, where each client has a given degree of parallel connections determined by its QoS requirement. The problem is further converted into the metric *FTFL* problem and we propose an approximation algorithm which is implemented in a distributed and asynchronous manner of communication. We show theoretically that the cost of our solution is no more than $2F^* + RC^*$, where F^* and C^* are two components of any optimal solution while *R* is the maximum number of parallel connections. Numerical experiments show that the cost of our solutions is comparable (within 4% error) to the optimal solutions.

In the third part, we establish mathematical formulation for the en-route web caching problem in a multi-server network that takes into account all requests (to any server) passing through the intermediate nodes on a request/response path. The problem is to cache the requested object optimally on the path so that the total system gain is maximized. We consider the unconstrained case and two QoS-constrained cases respectively, using efficient dynamic programming based methods. Simulation experiments show that our methods either yield a steady performance improvement (in the unconstrained case) or provide required QoS guarantees.

Statement of Originality

Papers Published

[1] Hong Shen and Shihong Xu. Coordinated En-Route Web Caching in Multi-Server Networks. *IEEE Transactions on Computers*, vol.58, no.5, pp.605-619, May 2009.

[2] Shihong Xu and Hong Shen. A Distributed (R, 2)-Approximation Algorithm for Fault Tolerant Facility Location. In *The 10th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT 2009)*, Hiroshima, Japan, Dec. 8-11, 2009.

[3] Shihong Xu and Hong Shen. The Fault Tolerant Facility Allocation Problem. In *The 20th International Symposium on Algorithms and Computation (ISAAC 2009)*, Hawaii, USA, Dec. 16-18, 2009.

[4] Shihong Xu and Hong Shen. QoS-Oriented Content Delivery in E-Learning Systems. In *The 2nd International Symposium on Information Technology in Medicine and Education (ITME 2009)*, pages 665-670, Jinan, China, Aug. 2009, invited paper.

[5] Shihong Xu and Hong Shen. An Efficient Method for p-Server Coordinated En-Route Web Caching. In *The 8th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT 2007)*, pages113-117, Adelaide, Australia, Dec. 2007.

[6] Shihong Xu and Hong Shen. An O(nh) Algorithm for Dual-Server Coordinated En-Route Web Caching. In *The 7th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT 2006)*, pages

399-404, Taipei, Taiwan, Dec. 2006.

Papers under Review

[1] Shihong Xu and Hong Shen. Fault Tolerant Facility Allocation. Submitted to *SIAM Journal on Computing*, Aug. 2009.

[2] Shihong Xu and Hong Shen. QoS-Aware Content Replication for Parallel Access in the Internet. Submitted to *IEEE Transactions on Parallel and Distributed Systems*, Aug. 2009.

Acknowledgments

I am deeply grateful to my principal supervisor, Prof. *Hong Shen*, for guiding my research during the PhD candidateship which started at the Japan Advanced Institute of Science of Technology (JAIST) since year 2005. From the discussion of my proposal to the last corrections of this document, he has been part of my research in all the stages. And it is from him that I learned critical thinking and skills in technical writing, which laid solid foundation for my future research. Furthermore, I would like to thank my co-supervisor, Dr. *Michael Sheng*, for his support on the two and a half years study at the University of Adelaide.

I owe great thanks to the School of Computer Science at the University of Adelaide and the JAIST Foundation for granting me scholarships, which make it possible for me to pursue the degree. Thank the staff in the two institutions for providing their selfless help and professional assistance in my work and life.

I would like to thank all the people who either directly or indirectly provide me knowledge, experience and support. I would like to thank my friends *Zonghua Zhang*, *Yingpeng Sang* and *Hui Tian* for sharing their experience in research. Especially, I would like to thank my colleagues: *Haibo Zhang* and *Yawen Chen*, with whom I traveled around Japan and Australia; *Yidong Li*, who shared with me the office room and those stressful times when struggling on a research problem; and *Donglai Zhang*, who initiated the regular tea time in our group and shared his precious insights into technologies. I will always remember that it is these brilliant people that accompanied me intellectually and personally in these years.

I would like to thank my family for their support and love which make my life enjoyable even in those stressful times. I thank my parents for making it possible for

me to receive high-level education, and my wife Jie Feng for supporting my research faithfully all the time. They have been and will always be the reasons that I strive for excellence.

Last but not the least, I would like to express my gratitude to the anonymous examiners of this thesis. Review of a PhD thesis contains considerable amount of work which requires reading the full document and looking for problems and potential enhancements with a critical mind. I would like to thank them for their precious time and constructive comments on this thesis.

List of Figures

3.1 Credit Offers for Opening a Facility 43
3.2 Ranking of Contributions 48
3.3 Randomized Procedure 63

4.1 Factors of Performance Ratio 90

5.1 System Model in a 3-Server Network 98
5.2 An Example of Hardness for Problem Decomposition 103
5.3 Decomposition of Caching Gain 106
5.4 Reuse of Intermediate Results 108
5.5 Impact of Cache Size 123
5.6 Impact of Access Pattern 125
5.7 Impact of the Number of Copies 126
5.8 Impact of the Number of Servers 128

List of Tables

4.1 Performance Ratios Using Random Points on a Grid (with various maximum connectivity) 86

4.2 Performance Ratios Using Random Points on a Grid (with various maximum facility cost) 86

4.3 Performance Ratios on Network Models 88

5.1 Process of Algorithm (in terms of $G_x(x_r)$ and $P_x(x_r)$) 108

5.2 Parameter Settings 121

5.3 Groups of Experiments 122

List of Algorithms

3.1 1.861-Approximation Algorithm 41

3.2 Algorithm for the *p-th* Phase 42

3.3 Restatement of the *p-th* Phase Algorithm 44

3.4 Multi-Factor Approximation Algorithm 51

3.5 Improved Algorithm for the *p-th* Phase 51

3.6 1.52-Approximation Algorithm 57

4.1 Facility-Side Pseudocode 78

4.2 Client-Side Pseudocode 78

4.3 Extended JMS Algorithm [47, 52] 80

5.1 Unconstrained *p*-Server CERC 107

5.2 Individual Latency Constrained *p*-Server CERC 114

5.3 Average Latency Constrained *p*-Server CERC 117

Chapter 1

Introduction

Hormann and Beaumont described delivery systems in real world as the following [82]:

"Squirrels gather nuts throughout a large territory to store them near their nest for easy winter access. Dairy farmers collect milk from crows every day. The milk is trucked to various processing plants and distribution points before it arrives the shelf at the neighborhood supermarket. A few times a week, the family shopper travels a mile or so to the market to buy milk and bring it home. The milk is stored in the kitchen refrigerator, only a few feet away from the kitchen table. Each morning, a container of milk is moved from the refrigerator and placed on the kitchen table, only a few inches from the family's breakfast cereal bowls where it is actually used."

Content delivery systems on the Internet are similar (metaphorically) to their counterpart in the real world: Comparable to nuts gathering, a simple content delivery system transfers the content, say a web page, from the server to the client directly, while an advanced system has a complex structure to facilitate the process of delivery and a group of nodes are involved in serving a request. It is obvious that the network performance is maximized when the requested content is stored throughout the network because it is faster and easier to use information stored locally than to retrieve it repeatedly from a remote source. However, this requires enormous cache memory which is unpractical in reality and therefore it becomes indispensable to find a balanced solution between network performance and storage cost.

In this thesis, we have developed novel methods to optimize the location of replicas, including replica servers and replica objects, regarding their respective models. The purpose of this work is to improve the efficiency of content delivery and enhance the experience of network access utilizing the existing investment in network facility. Our approaches are independent of specific facilities used in a network, conforming to a general principle proclaimed by Maupertuis: *If there occurs some change in nature, the amount of action necessary for this change must be as small as possible*.

1.1 Content Delivery in the Internet

As the Internet gradually becomes a critical infrastructure, a growing number of businesses, organizations and institutions rely on its operation. Efficient content delivery, as one of the uppermost functions of the Internet, is becoming increasingly important. As such, in this chapter we first investigate related techniques for improving the efficiency of Internet content delivery.

1.1.1 Content Delivery Systems

Just like the difference of milk delivering and nuts gathering, the process of content delivery varies depending on the specific systems chosen by users. Here, we discuss three dominant content delivery systems, i. e. traditional (simple) client/server networks, Content Distribution Networks, and Peer-to-Peer networks.

◇ The most prominent instance of traditional client/server networks is World Wide Web (WWW). WWW enabled the spread of small size information through an easy-to-use and flexible format by providing a standard protocol — Hypertext Transfer Protocol (HTTP) to transfer information over a computer network supporting Internet Protocol (i. e. Internet). Internet was designed according to the end-to-end principle [113] by keeping the core network relatively simple and moving the intelligence as much as possible to the network end-points. For example, a web server is responsible for accepting HTTP requests from clients and serving them HTTP responses along with data contents. Excluding this, a web server also caches the result of some script files like ASP/PHP files (called dynamic/active pages) as a static page to service future requests in order to speed the process. As such, a web server is much more complex than any node in a core network which is usually tailored to the tasks of routing and forwarding messages.

◇ Content Distribution Networks (CDNs) [51, 93, 101] could be regarded as advanced client/server networks which further apply the end-to-end principle by adding more intelligence to the edge-servers. A CDN replicates contents from the origin server to the surrogates — some edge-servers that act on behalf of the origin server. CDNs essentially enhance client/server networks by comprising a system of computers across the Internet that cooperate transparently and forward content to positions closer to end users for performance improvement. Popular examples of CDNs include Akamai [39], Radar [99], SPREAD [110] and Globule [93].

◇ Peer-to-Peer (P2P) networking [82] is an approach for delivering network services where participants share a portion of their own resources, such as processing power, disk storage, network bandwidth and printing facilities. Different from the above networks, resources in P2P networks are provided directly to other participants without intermediary network hosts or servers [116] and this implies that a P2P network participant is a provider and also a consumer of the network services. Instances of P2P Networks include file sharing systems like BitTorrent [97, 74].

We notice that P2P systems, though could also be regarded as caching systems, are quite different from others. Due to the fact that a P2P system does not distinguish the providers and consumers of contents, regarding techniques are not used extensively in commercial environment due to potential security risks. As such, we mainly focus on the first two systems which are used more extensively. Considering that P2P networks are very popular due to their prominent characteristics in applications like file sharing, we will analyze the opportunity of leveraging related techniques, e.g. parallel access, to enhance the existing systems. We also investigate related research topics such as decentralized architecture using DHT (Distributed Hash Table) as part of the future work in the last chapter of this thesis.

Despite the diversity of application technologies, the above systems share a common principle — replicating and distributing contents across the network in order to ease access for end-users. As perceived by the common knowledge: *Closer is better*, these techniques involve caching web contents in a proxy server, cooperative web caching in a group of federated caches and content replication among surrogate servers. We introduce these techniques including proxy caching [31, 118], cooperative web caching [67, 43] and content distribution networks [149] in the next subsection.

1.1.2 Evolution of Techniques

1.1.2.1 Single proxy caching

The idea of using proxy servers [79] to cache web objects arose when they were first used for Internet access by clients within a firewall. For security reasons, companies run a special type of HTTP servers called "proxy" on their firewall machines. A proxy server typically processes requests by forwarding them to the remote servers, intercepting the responses and sending the replies back to the clients [135]. Since the same proxy servers are typically shared by all clients inside the firewall, eventually this leads to a question of the effectiveness of using these proxies to cache documents.

Proxy caching [36] has been proved to be effective to improve the web performance and reduce the network traffic by caching frequently accessed copies of contents at a local proxy to service future requests. Due to the limit of cache memory, single proxy caching focuses on local replacement algorithms to accommodate the new requested objects [30, 64, 95]. Caching of web contents (e. g. HTML pages, images) can reduce bandwidth usage, server load and user's perceived latency since these contents can be fetched from proxy caches, which are located more closely than servers to users. In order to make full use of caching, significant research has been made in this area which extends the regarding techniques to a more advanced version, i. e. cooperative Web caching.

1.1.2.2 Cooperative web caching

Cooperative web caching [35, 100], where caches cooperating with each other to serve requests, is a powerful paradigm to improve cache effectiveness [35, 64, 63]. In related work [69, 131], emphasis has been put on maximizing the benefits of cooperative caching for distributed systems and large-scale systems. In [147], wide-area cache cooperation was studied under a simple model, in which distances among all nodes in the network are assumed to be the same. In [63], the authors examined three practical cooperative placement algorithms for large-scale distributed caches and showed that cooperative object management could significantly improve performance compared to single proxy caching. In order to make caches cooperate on a large scale and effectively increase caching population, several caching architectures were proposed [108], including hierarchical architecture [20], distributed architecture [131, 96] and hybrid architecture [131, 77].

Cooperative web caching has been recognized as one of the effective solutions to alleviate web service bottlenecks, reduce bandwidth consumption, access latency, as well as server load. There are two important issues in cooperative caching: cache location [66] and content management [127, 72, 117]. The first issue studies the problem of where to place network caches while the second focuses on the object placement and replacement, which attempts to make optimized storage decisions for given size cache memory to achieve maximized performance improvement through cache collaboration. Recent advances in caching technology [38, 66] have presented a new form of caching architecture, namely en-route web caching [11, 110]. In en-route (web) caching, copies of web contents are selectively placed in transparent en-route caches [66] along each response path and requests passing through later are satisfied by an en-route cache if the requested object is stored or forwarded to the server otherwise. Different from traditional en-route caching that does not take into account the loss resulted by object replacement when making a placement decision, coordinated en-route caching [127] integrates both object placement and replacement policies into a caching scheme that produces the maximum net gain after loss deduction. Since coordinated en-route caching makes caching decisions in a coordinated fashion between object placement and replacement, it outperforms traditional caching schemes.

1.1.2.3 Content Distribution Networks

With the development of web caching, researchers progressively put their effort on the global effect of deploying multiple sources, including contents in origin servers and surrogate servers across the network, using content replication techniques. Content replication [33, 61, 73, 101], usually in the form of CDN, optimizes content delivery in the way of moving servers and their contents closer to end users to ease access. Typically serving content from a local replica server has better performance than from the origin server and the cost of deploying required number of servers is less than the communication cost over long distance. These evidences suggest people to deploy CDNs in the Internet to achieve performance improvement and CDN gradually becomes indispensable especially for objects/services which are not cacheable but replicable such as some dynamical pages or personalized objects like “cookies”.

In business, CDNs render appreciable benefits to both content providers and content consumers. In technology, CDNs have also been proved to be an effective approach to alleviate congestion on the Internet and make the Internet more responsive. However, though CDNs are very desirable, design an efficient CDN is a non-trivial task and in the next section we will discuss some issues and challenges in the design of Content Distribution Networks.

1.2 Issues and Challenges

As stated in [105], all issues in content delivery can be roughly categorized into content distribution, cache management and request routing. Content distribution includes the placement of surrogate servers strategically, content selection and distribution based on user remands and content outsourcing methodologies. Cache management includes cache organization and maintenance as well as cache content management. Request routing is a technique that redirects a request optimally to a server hosting the required object.

1.2.1 Research Issues in Content Delivery

The following research issues are vital in a CDN for efficient content delivery and overall performance.

1. *Surrogate Server Placement:* A CDN involves creating and deploying server replicas (surrogates) to provide optimized performance to access contents in these servers. Server replicas are used usually deployed selectively among candidate sites and the objective of the problem is to achieve minimized cost, maximized performance improvement or other desired properties such as fault tolerant capability as studied in Chapter 3.

2. *Content Selection and Outsourcing Mechanisms:* This issue is to decide which objects should be replicated and how to distribute them to surrogate servers. Considering the huge amount of data in the Internet, the performance of a delivery system is highly dependent on the selection of contents to be replicated. Previous studies on Web workloads [5, 88, 26] and characterizing web objects [40, 15] suggested that most web objects are small (5-10KB), but the distribution of object sizes is heavy-tailed. This observation suggests replication of small objects with high hit rates. Content outsourcing involves the mechanism to trigger the process of content distribution and traditional content outsourcing mechanisms include cooperative push-based approaches, non-cooperative pull-based approaches and cooperative pull-based approaches.

3. *Content Distribution:* For given contents and surrogates (through the first two steps), it is natural to ask which surrogate(s) should be chosen to store the selected contents and this comprises an optimization problem with respect to different objective functions such as minimized storage cost or maximized throughput as studied in Chapter 4.

4. *Cache Organization and Management:* Between a surrogate server and a client,

there are some intermediary nodes — caches/proxy servers, how to organize these cache servers and how to manage contents in these caches is very important for the performance of the cache system. Web caching can be regarded as a special replication technique applied in a local proxy server or a group of proxy servers on the route that a request/response travels. That is, web caching studies how to replicate an object requested by the client among local proxy servers while content replication involves all surrogate servers in the network. Note that, we use term *object replica* and *copy* interchangeably in this thesis, whose concrete examples include but not limited to web pages, audio files, images, videos and server side scripts like PHP/ASP pages. In Chapter 5, we will study how to cache objects smartly in the intermediary nodes on a delivery path in order to serve future requests.

5. *Consistency Enforcement:* A CDN involves creating copies of web documents and placing these copies at well-chosen locations. It ensures (possibly different levels of) consistency when a replicated document is updated, and redirects a request to a server hosting updated document. Content replica consistency can be managed with a validation based scheme or an invalidation based scheme [110]. Among the existing consistency mechanisms, the TTL based consistency scheme is most widely used. *Tang et al.* [129] investigated the problem of placing object replicas under the TTL-based consistency scheme in a tree network and proposed a polynomial time algorithm to compute the optimal placement of object.

6. *Request Routing:* A request routing scheme redirects a client to a server hosting a document copy so that requests issued from the client are optimally served. When a document is placed at multiple sites, for a request, choosing the best site to access and routing the request is not easy — the resulting performance can dramatically vary depending on the selected site and the route of the request. The hardness of optimal server selection and request routing is mainly due to system's dynamics and different content delivery models. Intelligent content replication and traffic routing use proper technologies to intelligently place content replicas

(among proxies) and route network traffic to optimize network performance.

Among all these issues, we mainly consider issues 1, 3 and 4 which share a common objective — to place replicas optimally which includes replica server placement and replica content placement while the latter can further be classified as static replica object placement among surrogate servers and dynamic replica object placement among proxy server (or caches). Note that we perceive web caching as a special technique of content replication which takes place at proxy servers during the process of serving a request while the term content replication being used in the literature refers to the replication of content among surrogate servers which is initiated by the server. The server replica placement problem must be addressed during the initial infrastructure installation or hosting infrastructure upgrading. Content replication is triggered from time to time using system probing techniques to ensure that content placement is cost effective while web caching is carried out during the process of severing a request. These three issues share similar mathematical models but differ in application backgrounds.

1.2.2 Challenges

Sine Internet is shared by a huge number of users, its resources, particularly bandwidth over communication links, are extremely valuable and should be used in a most effective way. Therefore achieving efficient content delivery is one of the uppermost objectives. However, the over-evolving nature of the Internet brings challenges in managing and delivering content to users. As an example, a sudden spike in Web content requests may cause heavy workload on particular web servers, and as a result a hotspot [90] can be generated. Here, we discuss desired properties of a content delivery system which include high availability of service, Quality of Service (QoS), adaptivity and scalability.

◇ *High Availability of Service:* Availability of services is the heart of service excellence. What makes this characteristic particularly important is that no other element of service, including quality or timeliness, matters if services are not accessible. A desirable content delivery system guarantees provided functionality in unexpected circumstances of component faults. Thus, replication techniques with the character supporting high availability of service are essential to achieve this. Regarding techniques include server replication and content replication where replicas act as backups to provide service when one of them fails, which is likely caused by devastating implications of a virus outbreak or security breach.

◇ *Quality of Service (QoS):* QoS [140] is a set of technologies for managing network traffic in a cost effective manner to enhance user experiences. QoS technologies involve bandwidth measurement, detection of changing network conditions (such as congestion or availability of bandwidth), and traffic prioritization. QoS guarantees are especially important for latency-sensitive applications. QoS technologies can be applied to prioritize traffic for latency-sensitive applications (such as voice or video) and to control the impact of latency-insensitive traffic (such as bulk data transfers). In this thesis, we present QoS-aware caching schemes to meet user requirements according to their priorities using differentiated service.

◇ *Adaptability:* Today's Internet periodically suffers from hot spots, also known as flash crowds [111]. A hot spot is very common in a system providing on-demand media streaming which is typically triggered by an unanticipated news event. An adaptive system has the capability to handle flash crowds which can cause enormous network bandwidth, temporarily overwhelming a site's delivery capabilities. Due to the large majority of users attempting to get a small group of resources, how to utilize the contents already downloaded became a critical problem and P2P networking is helpful to achieve the adaptability of the system in this circumstance.

◇ *Scalability:* Scalability is a desirable property of a system, a network, or a process, which indicates its ability to handle growing amounts of work in a graceful manner [13]. In a content delivery system, scalability refers to the capability of a system to increase total throughput under an increased load when resources (typically number of servers) are added. In this thesis, we consider content replication techniques suitable for parallel access and multi-server environment which provide a capacity to serve any desired number of users and ensure QoS.

These desired properties pose great technical challenges in content delivery system design. Addressing all these issues and challenges is clearly out of the scope of this thesis and we hence choose some issues to study in the thesis.

1.3 Our Work

1.3.1 Issues Addressed in This Thesis

In this thesis, we address several key research issues to satisfy the aforementioned requirements which are desirable in designing a content delivery system. All the issues addressed involves location problem of replicas which specifically include site selection for placing replica servers, replica server selection for hosting replica contents and en-route caches selection on a delivery path for caching objects. These issues are summarized as the following:

◇ *Replica Server Placement:* This issue involves placing surrogate servers across the networks strategically before content distribution. Replica server placement usually incurs the cost of deploying these servers as well as the costs of clients for accessing services from these servers and typically the objective of this kind of problem is to minimize the sum of the above costs under certain condition. In a content delivery system, various faults, resulted from adversarial and inconsistent behaviors, occur at any time. A reliable system should perform and maintain functions not only in the ordinary situation but also in unexpected circumstances, including server crash and connection interception. In Chapter 3, we focus on reliability model for server replication to provide fault tolerance capability in case of server or connection failures. We achieve fault-tolerance capability by meeting fault-tolerant requirement for each client — if a connection from a client to a facility fails, the other facilities assigned could be used to serve the client. We proposed *Fault-Tolerant Facility Allocation* in [141, 142, 143] which minimizes the sum of facility cost and connection cost subject to the fault tolerance capability of each client.

◇ *Content Distribution:* This issue is to distribute content replicas among replica servers in a cost-effective manner to achieve best performing improvement. Traditional content replication scheme is optimized for single-access model, which suffers from the low service availability as well as quality of service. The challenge of this problem is to determine the best strategy so that the perceived performance is improved while the resource used won't increase too much. Parallel access provides a means to tradeoff between the system overhead and user perceived performance, so it's a more flexible way to deliver content efficiently. On the other hand, as most existing solutions are centralized in operation which requires a central server periodically update all nodes with global access information, a great challenge explored in this thesis is to investigate distributed solutions that do not require nodes to have global access information. We proposed QoS-aware content replication for parallel access in [144, 145, 117]. Our focus, as shown in Chapter 4, is on developing an efficient distributed algorithm with predictable performance guarantee, which is fully validated through comparison to optimal strategies.

◇ *En-Route Caching:* This issue is to optimize the caching of content along the path of a request/response travels to achieve maximized benefit in serving future requests. System scalability and efficiency are the major problems in existing en-route caching techniques due to the assumption of single server [127, 129, 73]. In a prospect system, there are multiple servers in operation that work collaboratively to provide scalability, efficiency and QoS. Web caching in multi-server systems is much more complex than that in the traditional single-server systems, because multiple traffic flows (requests/responses) for the same content, overlap on links. As a result, replication shall take into account all flows and result in a maximum benefit to all servers. Clearly it is more difficult to compute the optimal location of replicas in a multi-server system. A trivial approach is to decompose a multi-server network into several single-server networks and obtain a multi-server solution by combining individual single-server solutions together. This will been shown, unfortunately, incorrect due to the contradiction of different placement strategies in different single-server networks. We present corresponding solutions to the problem in Chapter 5 which first appeared in [144, 145, 117].

1.3.2 Overview of Our Work

This thesis covers solutions to the aforementioned three issues. These solutions bring significant contributions to applications, theories and methods for replica placement (and replacement) in devising a scalable, reliable and efficient system for Internet content delivery. In this subsection, we give an overview of our work as follows.

◇ *Fault-Tolerant Facility Allocation* (*FTFA*) [141, 142, 143]: Given n_f sites, each equipped with one facility, and n_c clients, the well-known *Fault-Tolerant Facility Location* (*FTFL*) problem [55] requires a minimum-cost connection scheme such that each client connects to a specified number of facilities. The *FTFL* problem is NP-hard and the best solution has an approximation ratio 2.076 obtained by applying the LP rounding technique [125]. In Chapter 3, we extend the *FTFL* problem to the *Fault-Tolerant Facility Allocation* (*FTFA*) problem by allowing each site to contain multiple replicas of the same facility and show that we can obtain better solutions for this problem. We give two algorithms with 1.81 and 1.61 approximation ratios within $O(mR\log m)$ and $O(Rn^3)$ running time respectively, where R is the maximum number of facilities required by any client, $m = n_f n_c$ and $n = \max\{n_f, n_c\}$. Instead of applying the dual fitting technique that reduces dual problem's solution to fit the original problem as used in the literature [52, 53, 84], we propose a method called inverse dual fitting that alters the original problem to fit the dual solution and show that this method is more effective for obtaining solutions of multi-factor approximation. We show that using factor-revealing technique our second algorithm is also (1.11,1.78)- and (1,2)-approximation simultaneously, these results can be further used to achieve solutions of 1.52-approximation to *FTFA* and 4-approximation to the *Fault-Tolerant* *k-Facility Allocation* problem in which an upper bound (*k*) of total number of replicas applies. We believe the inverse dual fitting technique has the potential to be applied in approximation analysis of similar problems in the future.

◇ *QoS-Aware Content Replication for Parallel Access* [144, 145, 117]: Parallel access to replicated contents enables simultaneous download of different portions of an object from multiple sources, and hence has been applied extensively to offload origin servers and improve end-user experience. In Chapter 4, we study the *QoS-aware object replication problem for parallel access* in which each client has a given degree of parallel access determined by its QoS requirement or priority. We formulate the problem as a maximization problem of combined download speed of all parallel connections at all clients, and then convert it into the metric *Fault Tolerant Facility Location* (*FTFL*) problem to minimize the total cost assuming shortest-path routing is deployed. We propose an approximation algorithm using primal-dual schema for the problem which is implemented in a distributed and asynchronous manner within $O(n)$ rounds of communication, where n is the number of surrogate servers in the network. As far as we know, the approximation factor of existing centralized algorithms using primal-dual schema remains unknown except a special case where all clients have a uniform degree of parallelism (i .e., $|\mathcal{R}| = 1$, where $\mathcal{R}$ is the set of parallel connection degrees). We prove that the cost of our solution is no more than $|\mathcal{R}| \cdot F^* + 2 \cdot C^*$ in the general case for both the distributed and centralized algorithms, where F^* and C^* are respectively the two components of cost in an optimal solution. Extensive numerical experiments showed that the quality of our solutions is comparable (within 4% error) to optimal solutions in all evaluated cases.

◇ *Coordinated En-Route Web Caching in Multi-Server Networks* [144, 145, 117]: With the emergence of various advanced networks that comprise a group of geographically distributed servers, such as Content Delivery Networks (CDNs) and Peer-to-Peer (P2P) systems, coordinated en-route web caching in multi-server networks becomes increasingly attractive but remains of great challenge as solutions for single-server networks become invalid here. In Chapter 5, we first establish mathematical formulation for this problem that takes into account all requests (to any server) that pass through the intermediate nodes on a response path and caches the requested object optimally among these nodes so that system gain is maximized. Then we derive efficient dynamic programming based methods for finding optimal solutions to the problem for the unconstrained case and two QoS-constrained cases respectively. For each case, we present a caching scheme to illustrate application of the corresponding method. Finally we evaluate the proposed schemes on different performance metrics through extensive simulation experiments. The experiment results show that our proposed schemes can yield a steady performance improvement and achieve desired QoS in a multi-server network. To the best of our knowledge, these are the first results for solving the problem of coordinated en-route web caching in multi-server networks.

1.3.3 Summary of Contributions

In summary, our work brings conceptual advances in problem modeling, theories and methods for replica placement (and replacement) in designing a scalable, reliable, and efficient system for content delivery in the Internet. The main contributions of our work are summarized as follows:

◇ We propose the *Fault-Tolerant Facility Allocation* problem for the placement of replica servers, which is shown to be NP-hard and we present three polynomial-time algorithms respectively with approximation factors 1.861, 1.61 and 1.52.

◇ We study an extension of the aforementioned problem called *k-Facility Allocation* problem and provide with a 4-approximation algorithm.

◇ We propose QoS-aware content replication for parallel access to satisfy QoS requirements of clients and convert the problem into the well-studied metric *Fault Tolerant Facility Location* problem.

- $\diamond$ We present a distributed $(R, 2)$-approximation algorithm for the metric *Fault-Tolerant Facility Location* problem which produces solutions comparable (within 4% error) to the optimal solutions within $O(n)$ rounds of communication.

- $\diamond$ We extended the *Coordinated En-Route Web Caching* to the case of multi-server network and present a dynamic programming algorithm to compute the optimal caching strategies in polynomial time.

- $\diamond$ We also consider the QoS constraints in the above problem, respectively on individual latency and average latency. We also provide optimal methods to computer the adapted problem.

On the methodology of our work, we contribute an inverse dual fitting technique which is used to design and analyse algorithms for the *Fault-Tolerant Facility Allocation* problem. Instead of shrinking the dual solution by a factor as in the dual fitting technique, the main step of dual fitting is to compose an extra instance of *FTFA* and a feasible solution to its dual problem. This technique could be regarded as an inverse process of dual fitting technique. Actually, the two techniques are the same in single-factor approximation analysis but our technique has the advantage to provide multi-factor approximation analysis as well. We believe this technique has the potential to be applied in approximation analysis of similar problems in the future.

1.4 Thesis Outline

This thesis comprises 6 chapters. In this chapter we introduce the status of Internet content delivery including research issues and challenges, followed by a brief introduction of our work. The remainder of this thesis is organized as follows.

In Chapter 2, we give an overview on replication techniques, including a literature review on content distribution networks and web caching as well as an introduction of fundamental methods and approaches used in related work.

Chapter 3, *Fault-Tolerant Facility Allocation*, presents a facility allocation model which differs from the classical fault-tolerant facility location problem by allowing multiple instances of facility to be opened at each location (site). We also consider an extension called *Fault-Tolerant k-Facility Allocation* that limits the total number of facilities by placing an upper bound *k*. Both problems are NP hard and a couple of algorithms are given to compute approximation solutions in polynomial time.

In Chapter 4, we propose QoS-aware content replication for parallel access in which each client has a given degree of parallel connections determined by its QoS requirement or priority. We formulate the problem as an optimization problem and present an approximation algorithm which is implemented deliberately in a distributed and asynchronous manner of communication.

Chapter 5, *Coordinated En-Route Web Caching in Multi-Server Networks*, presents optimal methods for coordinated en-route web caching in multi-server networks. Constraints regarding to QoS requirements are also considered to enhance the caching schemes for delay-sensitive applications.

The above three chapters constitute the main body of our work and Chapter 6 concludes the thesis which is complemented by the discussion of future work.

Chapter 2

Replication for Content Delivery — An Overview

Replication is one of the oldest and most important techniques in the area of distributed systems. Replication usually involves providing redundant resources to improve reliability, efficiency, and accessibility of a system or service. Data replication in distributed database systems [50, 1, 137] is the most notable example of the replication techniques. In this thesis we focus on replication techniques for performance improvement in Internet content delivery. Replication for content delivery involves replica servers deployment, replica content distribution and cache management. This thesis focuses on replica placement algorithms including the placement of replica server, placement of replica content in static global model and dynamic en-route model. In this chapter, we investigate existing methods and techniques in related models and give a critical review on related work.

2.1 Content Distribution Networks

A content distribution network comprises two core components: facility (servers) placement and content management. The problem of facility placement is to select a number of sites among candidates to deploy facility so that the objective function is optimized for the given network topology, client population and access patterns [120]. Replica content distribution further involves replication creation mechanisms and replica content placement. The problem of replica content placement is to select a number of servers to deploy contents so that certain objective function is optimized. Both replica (server and content) placement problems focus on the optimal locations of replicas in a network. Applications of these techniques can be found in some CDN projects including Akamai [39], Radar [99], SPREAD [110] and Globule [93].

2.1.1 Replica Server Placement

Li *et al.* [70] assumed that the underlying network topologies are trees and approached the proxy placement problem by solving a dynamic programming problem. They obtained an $O(M^2N^3)$ algorithm for placing M proxies optimally among N potential sites in terms of a given performance measure subject to system resource and traffic pattern. Jamin *et al.* [57] examined the placement problem for Internet instrumentation. They investigated both graph theoretic methods and heuristics for instrumenting the Internet to obtain distance maps. They showed that an Internet distance map service based on their placement techniques can offer useful hints for server selection by clients. Qiu *et al.* [98] explored the problem of Web server replica placement and develop placement algorithms that use workload information to make informed placement decisions. Their evaluation using both synthetic and real network topologies, as well as Web server traces show that these placement decisions are crucial to CDN performance. Radoslavov *et al.* [104] proposed two replica server placement algorithms which require no knowledge of client location but decide on replica location based on the network topology alone. Though the performance studies show that their algorithms performs only 1.1 to 1.2 times worse than that of the greedy algorithm proposed in Qiu *et al.* [98], it must be noted that these algorithms assumed that the clients are uniformly spread throughout the network, which may not be true. Lin and Yang [75] studied the placement of proxy servers to support server-based reliable multicast and presented the *k-maximum shortest path count* (*KMPC*) heuristic which places proxies on the nodes through which the most number of shortest paths pass through. Li and Shen [71] studied optimal methods for proxy placement in coordinated en-route web caching which outperforms the *KMPC* model with respect to performance metrics considered.

Similar but distinct from server placement, Danzig *et al.* [36] studied the placement problem of caches. They showed that the overall reduction in network FTP traffic is higher with caches inside the backbone (core nodes) rather than on the backbone edges (external nodes). Krishinan *et al.* [66] formulated the cache location problem by modeling the flow of data as flows on a graph and presented optimal methods for both linear topology and single-source tree topology.

Most of the above works assumed there is at most one standard server (or any facility) at each potential site which is a little obsolete due to the development of infrastructure techniques such as server farms and virtualization technology. In those environments, the capacity of a facility is customizable according to the demand of needs. As such, we define in the thesis a problem called *Fault Tolerant Facility Allocation* (*FTFA*) for the placement of replica servers among a set of candidate sites. The *FTFA* problem extends the well-studied *Fault Tolerant Facility Location* (*FTFL*) problem by allowing multiple replicas of the same facility at each site. The *FTFA* problem is less constrained and hence incurs a smaller total cost than *FTFL*. Corresponding algorithms and extensions are proposed in Chapter 3.

2.1.2 Content Replication

Considerable research has been conducted on developing novel protocols and mechanisms [149] for fast data delivery. Content replication [25, 45] as an approach for efficient content distribution has the advantages of broad flexibility and extendibility [78]. Web content replication [33] involves creating copies of a site's documents and placing these document copies at well-chosen locations. Different from the cache replacement algorithms which only consider objects in one cache, replica placement algorithms need to consider all possible locations (nodes) in the network and decide the optimal or near-optimal locations to host the object. Through replication in the Internet, content is closer to end-users and as a result the performance, especially the access latency perceived by a client, is significantly improved. On the server side, replication shifts the load from congested servers (or links) to less loaded servers so that all the servers are able to respond to requests quickly. At the same time, the service becomes more reliable and the system obtains improved fault-tolerant capability by deploying multiple replicas across the network.

Object placement [41] is one of the main problems in content replication which is usually formulated as a maximization problem regarding to the caching gain [127, 72, 117] or a minimization problem with respect to the total cost [129, 126]. Kangasharju *et al.* [61] modeled the content placement problem as an optimization problem. The problem is to place k objects in some of N servers, in an effort to minimize the average number of inter-AS hops a request must traverse to be serviced, meeting the storage constraints of each server. The problem was showed to be NP-complete and three heuristics ware proposed to address this problem. Tang and Chanson proposed coordinated en-route caching for object placement (and replacement) in a linear array [127], [128] to maximize the gain of communication cost reduction. Li *et al.* [72] presented a dynamic programming-based solution to the placement of content replica in a tree network. Their method can find the optimal solution in low time complexity. However, the same problem for general network topologies was showed to be NP-complete. Qiu *et al.* [98] experimentally compared several heuristic solutions and found that a simple greedy algorithm performs well. Baev *et al.* [8] developed a polynomial time constant-factor approximation algorithm for the replica placement problem in general networks.

Most of the existing works on content replication assumed single access of each client. However, with some popular content replicated across the Internet, parallel access to the replicated content enables simultaneous download of different portions of an object from multiple sources and hence improved end-user experience. In this thesis, we study the QoS-aware object replication problem for parallel access in which each client has a given degree of parallel access determined by its QoS requirement or priority. We formulate the problem as a maximization problem of combined download speed of all parallel connections at all clients, and provide a corresponding algorithm which is implemented in a distributed and asynchronous manner to work in a distributed environment.

2.1.3 Other Topics

Other topics in content distribution include content consistency enforcement, request routing and adaptive algorithms. For the first topic, content distribution mechanisms, which define how replica servers exchange updates, are very important. These mechanisms include cooperative push-based, non-cooperative pull-based and cooperative pull-based approaches [89]. Request routing redirects a client to the best replica server for dealing with its request. Most existing systems use the Domain Name System (DNS) for this purpose, but in different ways. Sivasubramanian [121] investigated the usage of adaptive replication algorithms for both static and dynamic Web sites, respectively using dynamic selection of replication strategies and automatic replication of application data. Wujuan and Veeravalli [139] proposed an adaptive object replication algorithm for distributed network systems and analyzed its performance from both theoretical and experimental standpoints. Content replication and distribution techniques can also be extended to be applicable in the wireless network environments [10, 60].

2.2 Web Caching

2.2.1 Caching Architecture

To make caches cooperate on a large scale and effectively increase caching population, a group of caches are usually organized in a federated architecture. There are three commonly used caching architectures [131, 108]: *hierarchical architecture* [20], *distributed architecture* [131, 96] and *hybrid architecture* [131, 77]. Hierarchical caching architecture organizes caches by placing them in multiple levels of a network. While in distributed caching architecture, caches are all placed at the bottom level and there are no intermediate caches. Hybrid caching architecture is a mixed architecture in which the above two architectures are mixed and matched to form a complementary and coordinated architecture. For these three architectures, each one has its advantages and disadvantages. Generally speaking, *hierarchical architecture* has shorter connection times than *distributed architecture* because additional copies at intermediate levels reduce retrieval latency for small documents. On the other hand, *Distributed architecture* has shorter transmission times and higher bandwidth usage. A well configured *hybrid architecture* can reduce both connection time and transmission time.

Currently, the popular locations for caches are at the edge of networks as part of cache hierarchies. Danzig *et al.* [36] observed the advantage of placing caches inside the backbone rather than at its edges. They showed that the overall reduction in network FTP traffic is higher with caches inside the backbone (core nodes) than with caches on the backbone edges (external nodes). A method to optimize the cache location inside the backbone was proposed by Krishinan *et al.* [66]. They formulated the cache location problem by regarding the network as a graph and modeling the flow of data from servers to clients as flows on this graph. They presented optimal solutions in linear topologies and single-source networks.

Example: *En-Route Caching*

Transparent En-Route Cache (TERC [66]) is a special type of caches placed transparently between servers and clients. The optimization problem of cache location for TERCs was addressed by Krishinan *et al.* [66] in a single-source network. A TERC intercepts a request that passes through itself, and either satisfies the request or forwards the request toward the server along the regular routing path: If the requested object is in the cache, the object is sent to the client and the request will not be propagated further upstream. Otherwise, the routing node forwards the request along the regular routing path toward the content server. If no en-route cache is found to contain the target object, the request will be eventually serviced by the content server. Since TERCs are transparent, only the caches along the route from a client to a server can benefit the request. Therefore, TERCs are easier to implement [32, 37] and manage than the replicated web servers since they are oblivious both to the end-user and the server. The effectiveness of TERCs depends on the Internet routing stability during the connection lifetime of an HTTP session. Measurement results [91, 68, 66] show that for the short duration of an

HTTP connection, routing is mostly stable. Using TERCs, en-route web caching can be implemented by a number of light-weight techniques such as extending the standard TCP or HTTP protocol in existing IP networks [107, 102] or using an active network where the routers can manipulate the messages flowing through and perform customized computations [130]. En-route web caching has a number of advantages as pointed out by [127]: First, it is transparent to both content servers and clients. Second, since no request is detoured off the regular routing path, the additional bandwidth consumption and network delay for cache miss are minimized. Moreover, it eliminates the extra overhead of locating the objects such as sending broadcast queries [136] and maintaining directories [43, 100].

2.2.2 Object Replacement Algorithms

Due to the limitation of cache resources, many object replacement algorithms [11, 114] have been proposed in the past in order to utilize caches effectively. Cache replacement algorithms [11, 114] usually maximize the cache hit ratio by attempting to cache the data items which are most likely to be referenced in the future. Since the future data reference pattern is typically difficult to predict, a common approach is to extrapolate from the past by caching the data items which were referenced most frequently. This approach is exemplified by the LRU (Least Recently Used) algorithm which evicts the object being requested the least recently and LFU (Least Frequently Used) which evicts the object being accessed least frequently. Excluding these traditional replacement algorithms, advanced algorithms were also proposed, which attempt to minimize various cost metrics, such as hit rate, byte hit rate, average latency and total cost. These replacement algorithms can be classified into key-based algorithms and cost-based algorithms as suggested in [4].

Key-based replacement algorithms [11] sort objects based upon a primary key, break ties based on a secondary key, break remaining ties based on a tertiary key and so on. These replacement algorithms include Size [2] which evicts the largest object and LRU-MIN [3] which evicts the small object that was least recently used from the cache. Similar algorithms include LRU-Threshold [3] and Hyper-G [2] and Lowest Latency First [138].

Cost-based replacement algorithms employ a potential cost function derived from different factors such as the time since last access, entry time of the object in the cache, transfer time cost, object expiration time and so on. For example, GreedyDual-Size (GD-Size) associates a cost with each object and evicts object with the lowest cost/size. Least Normalized Cost Replacement (LNC-R) [115] employs a rational function of the access frequency, the transfer time cost and the size. Similar examples include Hybrid [138], Bolot/Hoschka [12], Size-Adjusted LRU (SLRU) [4], Server-assisted scheme [30] and Hierarchical GreedyDual (Hierarchical GD) [64].

2.2.3 Coordinated En-Route Web Caching

In traditional caching schemes, object replacement in each cache is carried out independently. Tang *et al.* [127] integrated object replacement into a scheme of object placement and proposed coordinated en-route Web caching. Coordinated en-route Web caching regards the removal of objects at intermediate nodes as the loss of a caching decision (because it disables future access to these objects) and then optimizes the decision on a response path by maximizing the caching gain with the loss deducted. Coordinated en-route caching outperforms other schemes significantly in the experiments [127], though it only optimizes the caching decision in a linear array. Li *et al.* [72] extended coordinated en-route caching from a linear array to a tree and presented optimal methods for both constrained and unconstrained coordinated en-route caching in a tree network.

Aforementioned work all assumed that there is only one server in the system. This assumption became obsolete with the emergence of various state-of-art networks that contain a group of servers distributed geographically, such as CDNs [39] and P2P file sharing systems [123] which allow file transfer to be performed bi-directionally. However, we observed that the existing caching schemes cannot be applied directly to these systems. A naive idea is to decompose a multi-server system into multiple sub-systems each with one server and solve the problem in these single-server systems individually. Unfortunately, this approach does not work because solutions to sub-systems may contradict each other and simply combining them cannot yield an optimal solution to the original system. Thus, it remains a challenging task to find a way to enable coordinated en-route caching in multi-server networks that considers all servers in an integrated fashion and utilizes the advantage of multiple servers to provide QoS guarantees to the customers. In Chapter 5, we address the problem by presenting dynamic programming based solutions for the unconstrained case and two QoS-constrained cases.

2.3 Approaches and Methods

In this section, we give a brief introduction of related approaches and methods used in this thesis including linear programming, approximation algorithms and dynamic programming. These methods have seen extensive applications in the literature.

2.3.1 Linear Programming and Duality Theory

In mathematics, *linear programming* (LP) is a technique for optimization of a linear objective function subject to linear equality or inequality constraints. Formally, given a function

$$f(x_1, x_2, ..., x_n) = c_1x_1 + c_2x_2 + ... + c_nx_n + d$$

defined on a polytope (for example, a polygon or a polyhedron), a linear programming method will find a point in the polytope where this function has the smallest (or largest) value. Such points may not exist, but if they do, searching through the polytope vertices is guaranteed to find at least one of them. Let $\boldsymbol{c}$ be the vectors of coefficients and $\boldsymbol{x}$ the vector of variables (real numbers). The function to be maximized or minimized is called the objective function which can be expressed simply by $\boldsymbol{c}^T\boldsymbol{x}$. A linear program is a problem that can be expressed in canonical form:

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{c}^T\boldsymbol{x} \\ \text{subject to} \quad & A\boldsymbol{x} \geq \boldsymbol{b}, \end{aligned}$$

where $\boldsymbol{b}$ is a vector and A is a matrix of parameters. The equation $A\boldsymbol{x} \leq \boldsymbol{b}$ denotes the set of constraints which specify a convex polyhedron over which the objective function is to be optimized. Linear programming is used in this thesis to find a lower bound of the optimal solution in order to obtain the performance ratio of proposed algorithm.

There is a type of linear programming called integer programming, a.k.a. integer linear programming, which requires that all variables, i.e. $\boldsymbol{x}$ are constrained to take on integer values. A notable case of integer programming is 0-1 integer programming or binary integer programming (BIP) which requires all variables to take on binary integer values, i. e. 0 or 1. In Chapter 3 and Chapter 4, integer programming based formulation (or integer program) is used to define related problems.

Example 1: *Uncapacitated facility location problem*

Given a set of geographically distributed facilities, operating (opening) a facility $i \in \mathcal{F}$ incurs a cost of f_i. For a given set of clients $\mathcal{C}$, each client $j \in \mathcal{C}$ must be assigned to one facility, incurring a cost of c_{ij} which is associated with the distance between i and j. The objective of the facility location problem is to open a subset of facilities such that the combined cost including facility operating cost and connection cost is minimized. Facility location problem can be formulated as the following 0-1 integer program:

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{i\in\mathcal{F}} f_i y_i + \sum_{i\in\mathcal{F}}\sum_{j\in\mathcal{C}} c_{ij}x_{ij} \\ \text{subject to} \quad & \forall j \in \mathcal{C} : \textstyle\sum_{i\in\mathcal{F}} x_{ij} \geq 1 \\ & \forall i \in \mathcal{F}, j \in \mathcal{C} : y_i \geq x_{ij} \\ & \forall i \in \mathcal{F}, j \in \mathcal{C} : x_{ij}, y_i \in \{0, 1\} \end{aligned}$$

The first constraint ensures that each client $j \in \mathcal{C}$ is assigned to a facility, and the second constraint ensures that only open facilities are able to serve a client.

Example 2: *k-median problem*

Given a set of geographically distributed centers $\mathcal{F}$, we must select *k* ($k \leq |\mathcal{F}|$) centers to open so that each client, say j, in a given set $\mathcal{C}$ can be assigned to the open center which is most close to the client. Suppose the connection cost between facility i and client j is c_{ij}, the problem is to select the *k* open centers such that the sum of the connection costs is minimized. The *k*-median problem can be formulated as the following 0-1 integer program:

$$\begin{array}{ll} \text{minimize} & \sum_{i\in\mathcal{F}}\sum_{j\in\mathcal{C}} c_{ij}x_{ij} \\ \text{subject to} & \forall j\in\mathcal{C}:\ \sum_{i\in\mathcal{F}} x_{ij} \geq 1 \\ & \forall i\in\mathcal{F}, j\in\mathcal{C}:\ y_i \geq x_{ij} \\ & \sum_{i\in\mathcal{F}} y_i \leq k \\ & \forall i\in\mathcal{F}, j\in\mathcal{C}:\ x_{ij}, y_i \in \{0,1\} \end{array}$$

The first two constraints are the same to those in the facility location problem. The third constraint ensures that there are totally no more than *k* open centers. Note that the cost for opening centers is not considered in the *k*-median problem; instead it uses an upper bound of total center number *k*, which is supplied as part of the input. The center property of the open facilities is implied by the objective function as the facilities collectively should have the minimum access cost to its clients.

One of the most important theories in linear programming is *duality theory*. Every linear programming problem, referred to as a *primal problem*, can be converted into a *dual problem*, which provides an upper bound to the optimal value of the primal problem. In matrix form, we can express the primal (problem) and its dual (problem) as:

$$\begin{array}{llcll} \text{Primal :} & & & \text{Dual :} & \\ \text{minimize} & \boldsymbol{c}^T\boldsymbol{x} & & \text{maximize} & \boldsymbol{b}^T\boldsymbol{\alpha} \\ & & \text{and} & & \\ \text{subject to} & A\boldsymbol{x} \geq \boldsymbol{b} & & \text{subject to} & A^T\boldsymbol{\alpha} \leq \boldsymbol{c} \\ & \boldsymbol{x} \geq \boldsymbol{0} & & & \boldsymbol{\alpha} \geq \boldsymbol{0} \end{array}$$

In the dual problem, $\boldsymbol{\alpha}$ is used instead of $\boldsymbol{x}$ as variable vector. As we can see, the dual of a minimization problem is a maximization problem, and the variable vector in the primal is transformed into the right side of the constraints in the dual. This implies that the number of variables in the primal is equal to the number of constraints in the dual . Correspondingly, the right side of the constraints in the primal is transformed into the variable vector in the dual. An interesting observation is that the dual of the dual problem is the original primal problem.

Duality theory states that every feasible solution for a linear program gives a bound on the optimal value of the objective function of its dual. For the simplicity, we assume the primal problem is a minimization problem. The theorem states that the objective function value of the dual at any feasible solution is always greater than or equal to the objective function value of the primal at any feasible solution. The strong duality theorem states that if the primal has an optimal solution, $\boldsymbol{x}^*$, then the dual also has an optimal solution $\boldsymbol{\alpha}^*$ such that $\boldsymbol{c}^T\boldsymbol{x}^* = \boldsymbol{b}^T\boldsymbol{\alpha}^*$.

Example: *Primal and dual problem of uncapacitated facility location*

LP relaxation of the facility location problem is obtained by allowing x_{ij} and y_i to be non-negative real numbers, i.e.

$$\begin{array}{ll} \text{minimize} & \sum_{i\in\mathcal{F}} f_i y_i + \sum_{i\in\mathcal{F}}\sum_{j\in\mathcal{C}} c_{ij}x_{ij} \\ \text{subject to} & \forall j \in \mathcal{C}:\ \sum_{i\in\mathcal{F}} x_{ij} \geq 1 \\ & \forall i \in \mathcal{F}, j \in \mathcal{C}:\ y_i - x_{ij} \geq 0 \\ & \forall i \in \mathcal{F}, j \in \mathcal{C}:\ x_{ij}, y_i \geq 0. \end{array}$$

And the dual problem of the LP relaxation is:

$$\begin{array}{ll} \text{maximize} & \sum_{j\in\mathcal{C}} \alpha_j \\ \text{subject to} & \forall i \in \mathcal{F}:\ \sum_{j\in\mathcal{C}} \beta_{ij} \leq f_i \\ & \forall i \in \mathcal{F}, j \in \mathcal{C}:\ \alpha_j - \beta_{ij} \leq c_{ij} \\ & \forall i \in \mathcal{F}, j \in \mathcal{C}: \alpha_j, \beta_{ij} \geq 0, \end{array}$$

where the dual variables will be explained in Chapter 3.

Linear programming and duality theory is the foundation of many approximation algorithms. In Chapter 3 and Chapter 4, we apply the linear programming based formulation to express our problems and also used LP technique to solve the LP relaxed problems to get a lower bound of optimal solution. We also use duality theory in Chapter 3 to design and analyze the proposed algorithms.

2.3.2 Approximation Algorithms

A large number of optimization problems required to solve in practice are NP-hard. Complexity theory tells us that it is impossible to find efficient algorithms for such problems unless NP=P which is unlikely to be true [86]. In computer science, approximation algorithms are algorithms used to find the approximate solutions to NP-hard problems. Since it is unlikely that there can ever be efficient algorithms solving NP-hard problems, researchers settle for suboptimal solutions, and require them to be found in polynomial time. Different from heuristic algorithms, which usually only find practically good and fast solutions, provable solution quality and provable run time bounds are necessary in approximation algorithms. The approximation is ideally optimal up to a small constant factor, e. g. ρ. Throughout this thesis, a ρ-approximation algorithm is a polynomial-time algorithm that always finds a feasible solution with objective function value within a factor of ρ times of the optimal solution. It should be noted that approximation algorithms are increasingly being used for problems when polynomial algorithms are known but are too expensive due to the sizes of the data sets.

Both facility location problem and *k*-median problem are NP-hard, but there have been a number of constant-approximation algorithms developed for the metric version of problems in which the costs satisfy the triangle inequality. The best approximation algorithm known today for facility location problem was due to Jaroslow [18]. They improved the Chudak and Shmoys's algorithm [28] and obtained a 1.5-approximation factor in the new algorithm. Guha and Khuller [112] showed that there is no 1.462-approximation algorithm for the facility location problem, unless NP $\subseteq$ DTIME$[n^{O(\log\log n)}]$. The best known approximation algorithm for the *k*-median problem, due to Arya *et al.* [6], achieves a factor of $3+\epsilon$. It is also straightforward to adapt the proof of hardness of the facility location problem [112] to show that there is no $(1+\frac{2}{e}-\epsilon)$- approximation algorithm for *k*-median problem, unless

NP $\subseteq$ DTIME[$n^{O(\log\log n)}$]. Mainly, there are two types of techniques in approximation algorithm design, i. e. LP rounding and primal-dual schema. The drawback of LP rounding algorithms is that they need to solve large linear programs and so usually have prohibitive running times for most applications [56]. Therefore in this thesis, we use primal-dual schema to devise approximation algorithms that can be easily adopted in applications.

A primal-dual algorithm is an algorithm that is iteratively making primal and dual updates using the linear programming relaxation of the problem and its dual. Assume the primal and dual solutions have the same value for object function in the process of evolution. As pointed out in the literature, the dual solution produced under this condition is, in general, infeasible to the dual problem (otherwise, we would be able to find an optimal solution for the primal problem). The dual fitting technique shrink the dual solution by a factor to make it fit the problem and this factor is exactly the approximation ratio of the algorithm. In order to find such constants which is reasonable good, factor-revealing LPs are used in the literature [53, 52, 84].

2.3.3 Dynamic Programming

For a problem which can be solved optimally in a polynomial time, the quality of corresponding algorithms mainly depends on the time complexity of the algorithms. Here we introduce an important optimization method — dynamic programming, which is used in Chapter 5. A method using dynamic programming usually takes much less time than naive methods and therefore dynamic programming has been used extensively in optimization of location problems [127, 72] when applicable.

Dynamic programming is a technique to simplify a complicated problem by breaking it down into simpler subproblems in a recursive manner. For a decision problem, dynamic programming simplifies a decision by breaking it down into a sequence of decision steps over time. A dynamic programming based method usually has the following characteristics as outlined by Trick [132]:

◇ The problem can be divided into stages with a decision required at each stage.

- ◇ Each stage has a number of states associated with it.
- ◇ The decision at one stage transforms one state into a state in the next stage.
- ◇ Given the current state, the optimal decision for each of the remaining states does not depend on the previous states or decisions.
- ◇ There exists a recursive relationship that identifies the optimal decision for stage j, given that stage $j+1$ has already been solved. The final stage must be solvable by itself and the optimal values of the decision variables are recovered one by one, by tracking back the calculations already performed.

The last two properties are tied up in the recursive relationships given above. The skills in dynamic programming are to determine stages and states so that all of the above hold.

Chapter 3

Fault Tolerant Facility Allocation

In this chapter, we define a problem called *Fault Tolerant Facility Allocation* (*FTFA*) for the placement of replica servers among a set of candidate sites in a content distribution network. The *FTFA* problem extends the well-studied *Fault Tolerant Facility Location* (*FTFL*) problem by allowing multiple replicas of the same facility at each site. Because a site may hold as many replicas of a facility as necessary, the *FTFA* problem is less constrained and hence incurs a smaller total cost than *FTFL*. *FTFA* is, however, harder than the *Uncapacitated Facility Location* (*UFL*) problem which is NP hard, and in this chapter we show how to obtain smaller approximation ratios for *FTFA* than the best-known approximation ratio (2.076) for *FTFL*. We also consider the *Fault-Tolerant k-Facility Allocation* problem which has an upper bound *k* for the total number of replicas that can be deployed.

3.1 Introduction

The classical facility location problem [94] has been widely studied in the field of operations research. In this problem, we are given a set $\mathcal{F}$ of n_f sites where each site $i \in \mathcal{F}$ is associated with a non-negative cost f_i for operating (opening) a facility at this site, and a set $\mathcal{C}$ of n_c clients where each client requires to access one facility and a connection between client $j \in \mathcal{C}$ and facility $i \in \mathcal{F}$ incurs connection cost c_{ij}. The objective of facility location problem is to find a subset of $\mathcal{F}$ and deploy one facility at each site in the subset so that the total combined cost for operating and accessing these facilities is minimized.

In this chapter, we study a generalization of the facility location problem called *Fault-Tolerant Facility Allocation* (*FTFA*). In this problem, a site holds an unlimited number of identical facilities (replicas) and a client requires a prespecified number of connections to facilities for the fault tolerance purpose. Specifically, each client $j \in \mathcal{C}$ requires r_j connections so that it can tolerate up to $r_j - 1$ connection (or facility) failures. The *FTFA* problem requires allocating each site a proper number of replicas and further each client the required number of replicas so that the total combined cost for operating and accessing these facilities is minimized. The *FTFA* problem can be formulated by the following integer linear program.

$$\begin{array}{ll} \text{minimize} & \sum_{i\in\mathcal{F}} f_i y_i + \sum_{i\in\mathcal{F}}\sum_{j\in\mathcal{C}} c_{ij}x_{ij} \\ \text{subject to} & \forall j \in \mathcal{C}:\ \sum_{i\in\mathcal{F}} x_{ij} \geq r_j \\ & \forall i \in \mathcal{F}, j \in \mathcal{C}:\ y_i \geq x_{ij} \\ & \forall i \in \mathcal{F}, j \in \mathcal{C}:\ x_{ij}, y_i \in \mathbb{Z}^+ \end{array} \tag{3.1}$$

In this formulation, non-negative integer y_i indicates how many replicas are deployed at site i and x_{ij} indicates how many connections between site i and client j are established. The cost for operating facilities is calculated by $\sum_{i\in\mathcal{F}} f_i y_i$ and the cost for accessing facilities by $\sum_{i\in\mathcal{F}}\sum_{j\in\mathcal{C}} c_{ij}x_{ij}$. The objective is to find $\boldsymbol{y} = \{y_i\}$ and $\boldsymbol{x} = \{x_{ij}\}$ so that the combined cost (the sum of the above two items) is minimized. Note that the first constraint ensures fault tolerance of connectivity — there should be at least r_j connections for any client $j \in \mathcal{C}$. The second constraint secures enough replicas at each site for the requirement of connections from any client.

FTFA is similar to the well studied *Fault-Tolerant Facility Location (FTFL)* problem [55, 47, 48, 125] which has the same objective function and constraints as *FTFA* except the variants: for any $i \in \mathcal{F}, j \in \mathcal{C}$, x_{ij} and y_i are non-negative integers (i. e. $\mathbb{Z}^+$) in *FTFA* but binary (i. e. 0 or 1) in *FTFL*. Without the restriction on the maximum number of replicas that can be opened at each site, *FTFA* is less constrained and hence incurs a smaller total cost than *FTFL*. Note that the *FTFA* problem is also applicable in a scenario where the capacity of a facility is customizable according to the demand of needs which has been realistic due to the development of infrastructure techniques such as server farms and virtualization technology. As an example, we use the *FTFA* problem deploy surrogate server in a content distribution network. In such an application, multiple replicas (surrogate servers) can be deployed at one site if necessary, which share the duty to serve clients together. We notice that the *FTFA* problem becomes the classical *Uncapacitated Facility Location (UFL)* problem when connectivity requirement $r_j = 1$ for all $j \in \mathcal{C}$. It is not hard to see that the hardnesses of *UFL*, *FTFA* and *FTFL* comply with the following relation:

$$\textit{UFL} \subseteq \textit{FTFA} \subseteq \textit{FTFL}.$$

Here, the second inclusion is implied by a special *FTFL* problem which has a set of facilities distributed by groups. Let $\mathcal{F}' = \mathcal{F} \times \{1, 2, ..., R\}$, where $R = \max_{j \in \mathcal{C}} r_j$ is the number of identical replicas in each group. Using this setting, the *FTFA* problem can be solved by *FTFL* algorithms because the number of replicas at any site is no more than R in any optimal solution of the *FTFA* problem. However, we notice that the existing algorithms for *FTFL* are not as efficient as the algorithms for *UFL* , in both approximation ratio and time complexity: most *FTFL* algorithms employ an LP routine which is rather time consuming and the best known approximation ratio for *FTFL* is 2.09 which is considerably worse than the best known 1.5 approximation ratio for *UFL*. Therefore, in order to achieve a better result than that from applying *FTFL* algorithms directly, we must take the full advantage of existing methods for *UFL* in solving the *FTFA* problem. For a given *FTFA*, consider a *UFL* problem with the aforementioned $\mathcal{F}'$ and client set $\mathcal{C}' = \{(j, p), j \in \mathcal{C}, 1 \leq p \leq r_j\}$, where (j, p) is the *p-th* port of client j, we are able to regard the *FTFA* problem as a *UFL* problem with an additional constraint: no parallel connections are made between any replica-client pair, i. e., different ports of a client must be connected with different replicas. However, this constraint is nontrivial and as a result, *FTFA* becomes harder to solve than *UFL*, yielding the first inclusion. In the subsequent sections, we will show how to deal with this constraint and obtain better single-factor and bi-factor approximation solutions to *FTFA* than that for *FTFL*.

3.1.1 Related Work

The facility location problem and its variants occupy a central place in operations research [94]. For the simplest problem — the maximization variant of *Uncapacitated Facility Location*, Cornuejols *et al.* [34] obtained a $(1 - e^{-1})$-approximation algorithm. The first approximation algorithm for the minimization variant is a greedy algorithm due to [49], which is $O(\log n)$-approximation in the general (nonmetric) case. The *UFL* problem has found extensive applications since these works and one of the most widely studied variants of the problem is *metric UFL*. In this variant, the function of connection cost forms a metric, i. e., the connection costs between facilities and clients satisfy triangle inequality. Existing algorithms for the *metric UFL* problem mainly use two types of techniques, i. e. LP rounding and primal-dual algorithms.

The first constant-factor approximation algorithm for the *metric UFL* problem was due to Shmoys *et al.* [119]. They gave a 3.16-approximation algorithm using the filtering technique of Lin and Vitter [76] to round the optimal solution of a linear programming relaxation. Chudak, Williamson and Sviridenko improved the approximation ratio to 1.736 [29] and 1.582 [124] by rounding an optimal fractional solution to a linear program.

Charikar and Guha obtained 1.853-approximation and 1.728-approximation algorithms [22] by using primal-dual theory and greedy augmentation; Jain *et al.* presented greedy algorithms based on dual fitting and factor-revealing LP technique, achieving approximation guarantee 1.861 [52, 84] and 1.61 [53, 52]; Different from traditional primal-dual schema [54, 133], dual fitting relax the feasibility of the dual solution. Suppose the dual solution become feasible after shrunk by a factor and then this factor is the approximation factor of the algorithm. They also studied the tradeoff [52] between facility cost and connection cost and present a series of bi-factor approximation guarantees. Mahdian *et al.* further improved the approximation ratio to 1.52 [80] by adding a scaling and greedy augmentation procedure to the JMS algorithm. Byrka [18] modified the

Chudak and Shmoys's algorithm [28] and obtained a new algorithm which is the first one that touches the approximability limit. Their new approach gives a 1.5-approximation algorithm, which are currently the best known solution for the problem.

Fault Tolerant Facility Location [55] is a generalization of *UFL*, where connectivity at a client (e. g. the number of distinct replicas that serve a client) is prespecified to meet fault-tolerant requirements. Guha *et al.* obtained a 3.16-approximation algorithm by rounding the optimal fractional solution to the problem and further improve the result to 2.41 by employing a greedy local improvement step [48]. Recently, Swamy and Shmoys presented a 2.076-approximation by using LP rounding [125]. All these results hold for both uniform connectivity case and nonuniform connectivity case (general case). Guha and Khuller proved that the best approximation ratio (lower bound) to *UFL* is 1.463 [112], assuming NP $\nsubseteq$ DTIME$[n^{O(\log \log n)}]$, this result also holds for fault-tolerant version of the problem.

The *k-Median* problem [76] has also been studied extensively [6, 21, 23] and the best known approximation ratio for this problem is $3+\varepsilon$ [6]. Jain *et al.* studies a new problem called *k-Facility* which is a combination of the *k-Median* and *UFL* problems and achieve a 6-approximation algorithm [56] and further a 4-approximation [52] based on the JMS algorithm [53, 52]. The *Fault Tolerant k-Facility* problem was also studied by Swamy and Shmoys [125] and they achieved a 4-approximation algorithm for a special case of the problem where all clients have an identical connectivity requirement. Performance of their algorithm is unknown for the general case.

3.1.2 Our Technique

Consider an integer linear program containing $k \geq 1$ items in the objective function. For the facility location problem, $k = 2$ — the facility cost and connection cost. Suppose an optimal solution $OPT_1 = \sum_{p=1}^{k} I_p^*$, i.e., I_p^* is the cost for the *p-th* item in the optimal solution. We say a solution SOL is $(\lambda_1, ...\lambda_k)$-approximation if $SOL \leq \sum_{p=1}^{k} \lambda_p I_p^*$ for any optimal solution. When $k = 1$ or $\lambda_1 = \lambda_2 =, ... = \lambda_k$, we say it is a single-factor approximation solution, otherwise multi-factor approximation solution.

Consider a minimization problem and a primal-dual algorithm — an algorithm that is iteratively making primal and dual updates using the linear programming relaxation of the problem and its dual. Let the primal solution and dual solution have the same value for object function in the process of evolution. As pointed out in the literature, the dual solution produced under this condition is, in general, infeasible to the dual problem (otherwise, we would be able to find an optimal solution for the primal problem). Instead of shrinking the dual solution by a factor in order to make it fit the original problem as used in the dual fitting technique [53, 52, 84], we propose a method called *inverse dual fitting* that constructs an additional instance of the original problem to make the problem fit its dual solution. Inverse dual fitting has the same effect as dual fitting for single-factor approximation, but is more powerful in multi-factor approximation analysis as shown below.

Formally, for a primal problem of minimization, we scale the coefficients in the objective function (i.e. the right side of the constraints in the dual problem) with constant λ_p for the *p-th* item and obtain $OPT_2 \leq \sum_{p=1}^{k} \lambda_p I_p^*$, where OPT_2 is the optimal solution to the scaled instance of the primal problem. Actually, we can regard the original optimal solution (now with total cost $\sum_{p=1}^{k} \lambda_p I_p^*$) as a feasible solution to the scaled instance. Due to the duality theory which states that the maximum of the dual problem is no more than the minimum of the primal problem, we have $SOL_D \leq OPT_2$. As such, we only need to ensure that $SOL_P = SOL_D$ and SOL_D is a feasible dual solution to the scaled instance to achieve the result of $(\lambda_1, ...\lambda_k)$-approximation, i.e. $SOL_P \leq \sum_{p=1}^{k} \lambda_p I_p^*$. The first condition is usually ensured by the algorithm — for our algorithms, the total cost is equal to the total credit paid by all clients. In order to ensure the feasibility of SOL_D (to the scaled instance), we need proper constants $\lambda_p, 1 \leq p \leq k$. Factor-Revealing LPs are usually used to derive such constants.

3.1.3 Our Results

We use inverse dual fitting technique to design and analyze two algorithms for the *metric FTFA* problem. Both algorithms run through R phases and in each phase employ a subroutine to pick the most cost-effective star iteratively. The concept of cost efficiency is used by the MMS algorithm [84] which is a one phase algorithm for *UFL*. The difference here is that our algorithms comprise multiple phases and in each phase deal with a distinct constraint to ensure the feasibility of the solution. To satisfy the constraint, our algorithms need to process three types of events: One for replicas opened in a previous phase, one for replicas opened in the current phase and another for opening a new replica in the current phase. Combined with Factor-Revealing LPs in the literature, our algorithms achieve 1.81 and 1.61 approximation factors within running time $O(mR\log m)$ and $O(Rn^3)$ respectively, where $m = n_f n_c$ and $n = \max\{n_f, n_c\}$.

The second algorithm aforementioned is also shown to be (1.11,1.78)- and (1,2)-approximation simultaneously by applying the techniques of inverse dual fitting and Factor-Revealing LPs cooperatively. The first result is further used to obtain a 1.52-approximation algorithm to *FTFA* and the second one a 4-approximation algorithm for the *Fault-Tolerant k-Facility Allocation* problem, which has an upper bound on the total number of replicas, i. e. *k*, on the base of *FTFA*.

The remainder of the chapter is organized as follows. In Section 3.2 and Section 3.3, we present the single-factor and bi-factor approximation algorithms for solving the *FTFA* problem. In Section 3.4, we show how to extend the bi-factor approximation solution for *FTFA* to solve the problem of *Fault-Tolerant k-Facility Allocation* (*FTKFA*) in which the number of replicas has an upper bound *k*. Section 3.5 discusses a generalization of *FTFA* by allowing prespecified demand for each client and other applications.

3.2 Single-Factor Approximation

3.2.1 The Algorithm

Without loss of generality, assume the set of connectivity requirements $\mathcal{R} = \{1, 2, 3, ..., R\}$, otherwise we may add dummy clients for the missing connectivity requirements. Further we assume there are r_j ports at each client and R replicas at each site. All ports of a client must be connected in the order from 1 to r_j and all replicas at a site can be opened, if necessary, in the order from 1 to R. We use vector $\boldsymbol{y}^p$ to denote whether the *p-th* replica is opened for any site in $\mathcal{F}$ and $\boldsymbol{x}^p$ whether the *p-th* port of a client is connected with a replica at any site. It is clear that $\boldsymbol{y} = \sum_{p=1}^{R} \boldsymbol{y}^p$, $\boldsymbol{x} = \sum_{p=1}^{R} \boldsymbol{x}^p$ and $x_{ij}^p = 0$ if $p > r_j$. Program 3.1 can be rewritten as

$$\begin{array}{rl} \text{minimize} & \sum_{i\in\mathcal{F}}\sum_{p\in\mathcal{R}}(f_i y_i^p + \sum_{j\in\mathcal{C}} c_{ij}x_{ij}^p) \\ \text{subject to} & \forall j \in \mathcal{C}, 1 \le p \le r_j :\ \sum_{i\in\mathcal{F}} x_{ij}^p \ge 1 \\ & \forall j \in \mathcal{C}, i \in \mathcal{F} :\ \sum_{p=1}^{R} y_i^p - \sum_{p=1}^{R} x_{ij}^p \ge 0 \\ & \forall j \in \mathcal{C}, i \in \mathcal{F}, p \in \mathcal{R} :\ x_{ij}^p, y_i^p \in \{0,1\}. \end{array} \tag{3.2}$$

The LP-relaxation of this program can be obtained by allowing x_{ij} and y_i to be non-negative real numbers. The dual problem of the LP relaxation is

$$\begin{array}{rl} \text{maximize} & \sum_{j\in\mathcal{C}}\sum_{p=1}^{r_j}\alpha_j^p \\ \text{subject to} & \forall i \in \mathcal{F}, p \in \mathcal{R} :\ \sum_{j\in\mathcal{C}} \beta_{ij}^p \le f_i \\ & \forall i \in \mathcal{F}, j \in \mathcal{C}, p \in \mathcal{R} :\ \alpha_j^p - \beta_{ij}^p \le c_{ij} \\ & \forall i \in \mathcal{F}, j \in \mathcal{C}, p \in \mathcal{R} : \alpha_j^p, \beta_{ij}^p \ge 0. \end{array} \tag{3.3}$$

We use the same interpretation for dual variables, i. e. α_j^p and β_{ij}^p as in [84, 52] which will be explained in detail later. An interesting observation is that we can extract $p \in \mathcal{R}$ in the constraints and the objective function of the dual problem if we have $\alpha_j^p = 0$ when $p > r_j$. We utilize this observation to design a greedy algorithm which decomposes the problem into R subproblems and solves them in order. For the sake of simplicity, let vector $\boldsymbol{X}^b = \sum_{p=1}^{b} \boldsymbol{x}^p$ and $\boldsymbol{Y}^b = \sum_{p=1}^{b} \boldsymbol{y}^p, 1 \le b \le R$, our algorithm evolves the solution from the initial stage (suppose $\boldsymbol{X}^0$ and $\boldsymbol{Y}^0$), through R phases, to $\boldsymbol{X}^R$ and $\boldsymbol{Y}^R$. In each phase $p \in \mathcal{R}$, the algorithm establishes one connection for each client if it is not-fully-connected, i. e. clients in $\mathcal{C}_p = \{j \in \mathcal{C} : r_j \ge p\}$. A replica opened in one phase at site i can be used for free by any client j in the next phase, suppose p, if this usage does not violate the constraint $X_{ij}^p \le Y_i^p$.

The process of our algorithm is presented in Algorithm 3.1: In the *p-th* phase, the solution inherited from the last phase, i. e. $(\boldsymbol{X}^{p-1}, \boldsymbol{Y}^{p-1})$, as well as $\mathcal{F}$ and $\mathcal{C}_p$ are used as the input of the subroutine. Note that clients with $r_j < p$ is already fully-connected and therefore not included in $\mathcal{C}_p$. Suppose the new opened replicas and new established connections are denoted by $(\boldsymbol{x}^p, \boldsymbol{y}^p)$, then in the next phase, we have $\boldsymbol{X}^p = \boldsymbol{X}^{p-1} + \boldsymbol{x}^p$ and $\boldsymbol{Y}^p = \boldsymbol{Y}^{p-1} + \boldsymbol{y}^p$ as part of the input. The algorithm ends when all R phases are finished.

Algorithm 3.1 1.861-Approximation Algorithm

```
Input: $f_i, r_j, c_{ij}$ for any $i \in \mathcal{F}, j \in \mathcal{C}$.
Output: $x_{ij}, y_i$ for any $i \in \mathcal{F}, j \in \mathcal{C}$.
(1) Initially set vector $\boldsymbol{X}^0 \leftarrow \mathbf{0}, \boldsymbol{Y}^0 \leftarrow \mathbf{0}$ and the number of current phase $p \leftarrow 1$.
(2) While $p \leq R$:
    (a) Invoke the algorithm for the p-th phase with input $(\boldsymbol{X}^{p-1}, \boldsymbol{Y}^{p-1}, \mathcal{F}, \mathcal{C}_p)$, suppose the output is $(\boldsymbol{X}^p, \boldsymbol{Y}^p)$.
    (b) Set $p \leftarrow p + 1$.
(3) Set $\boldsymbol{x} = \boldsymbol{X}^R$ and $\boldsymbol{y} = \boldsymbol{Y}^R$
```

In the algorithm for the *p-th* phase, we use a notation of star and a definition of cost efficiency. A star is composed of a replica and a group of clients that are connected with the replica. Considering the time before the new star is selected, the *cost efficiency* of a star is defined to be

$$\text{eff}(i, p, C') = \frac{f_i^p + \sum_{j \in C'} c_{i,j}}{|C'|}, \tag{3.4}$$

where f_i^p is the cost paid to open a replica at site i in phase p and C' the set of client members in the star. The two items in the numerator represent the total cost of the star and therefore the cost efficiency of a star is actually the average payment of all client members to establish the star. Let $U \subseteq \mathcal{C}_p$ be the set of not-fully-connected clients in phase p, $C' \subseteq U$ is a set of clients chosen by the algorithm to be connected with the replica. As an open replica can be accessed for free under certain condition, the cost paid to the site is equal to zero if no new replica have to be opened. Formally,

$$f_i^p = \begin{cases} f_i & \text{if a new replia of } i \text{ must be opened at phase } p; \\ 0 & \text{otherwise.} \end{cases}$$

Algorithm 3.2 Algorithm for the *p-th* Phase

```
(1) Let $U \subseteq \mathcal{C}_p$ be the set of not-fully-connected clients, initially set $U \leftarrow \mathcal{C}_p$.
(2) While $U \neq \phi$:
    (a) Find the most cost efficient star $(i, p, C')$ according to Formula (3.4).
    (b) Open a replica at site $i$, if it is not already open, and establish a connection to
        the replica for all clients in $C'$.
    (c) Set $f_i \leftarrow 0, U \leftarrow U \setminus C'$.
```

Now the dual variables i. e. α_j^p and β_{ij}^p can be used to find the most cost efficient star. We use the same interpretation as in [84, 52]: α_j^p is the total cost (including the connection cost and the contribution to open replicas), increasing simultaneously with time, paid by the *p-th* port of client j and β_{ij}^p is the contribution received by site i from client j at the *p-th* phase. As such, the most cost efficient star in each iteration of the subroutine can be found in this way: if the dual variables of all unconnected clients are raised simultaneously, the most cost efficient star will be the first star (i, p, C') formed at time t such that

$$\sum_{j \in C'} \max(t - c_{ij}, 0) = f_i^p.$$

The algorithm for *p-th* phase opens the most cost efficient star repeatedly until all the clients in $\mathcal{C}_p$ are *connected* with a replica. Once a client is connected, it is removed from U; in contrast, a replica is never removed, instead it can be reused for free under certain condition. In fact, the subroutine is very close to the MMS algorithm proposed in [52, 84] for the *UFL* problem. The difference is, here we need to ensure the feasibility of the solution by maintaining a distinct constraint. For the sake of simplicity, we set $y_i^p \leftarrow 1$ to mean a new replica at site i is opened and $x_{ij}^p \leftarrow 1$ a new connection between client j and site i is established. In order to maintain the feasibility of a solution, i. e. $X_{ij}^p \leq Y_i^p$, we consider three cases for any client $j \in \mathcal{C}_p$:

1. $X_{ij}^{p-1} < Y_i^{p-1}$: Set $x_{ij}^p \leftarrow 1$ as site i is *eligible* to be connected by client j. There is no need to open a new replica at site i and there $f_i^p = 0$.

2. $X_{ij}^{p-1} = Y_i^{p-1}$ and $y_i^p = 0$: In this case, we must open a new replica at site i, i. e.

set $y_i^p \leftarrow 1$ in order to establish a new connection for client j and therefore $f_i^p = f_i$. The operating cost is shared between clients in C' and any client contributed to opening the replica can be connected.

3. $X_{ij}^{p-1} = Y_i^{p-1}$ and $y_i^p = 1$: This case is the result of the second case. The feasibility of the solution is maintained if we set $x_{ij}^p \leftarrow 1$. We do not have to open a new replica, i. e. $f_i^p = 0$.

In all three cases, only the second one involves more than one client. Suppose each site has a list of clients sorted according to their connection costs to the site. As shown by Figure 3.1, the most cost efficient star will consist of a replica and a set, containing the first k clients in this order, for some k. Therefore the algorithm can be finished efficiently in polynomial time.

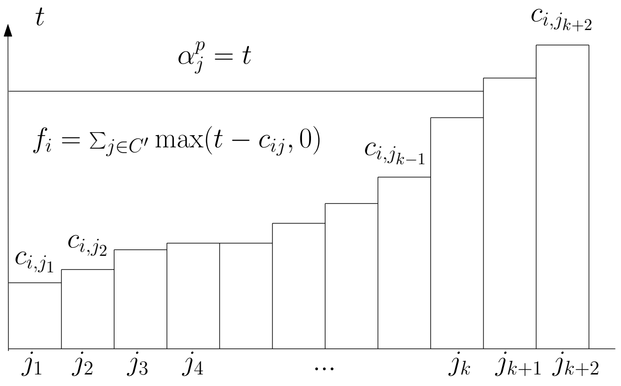


Figure 3.1: Credit Offers for Opening a Facility

Here, we use three types of events to process these cases respectively. Note that $f_i^p = 0$ implies any client $j \in U$ forms a star with the site once the client has enough credit to pay the connection cost, i. e. $c_{ij} = t$. We restate the algorithm for the *p-th* phase based on this observation.

Remark 3.1. Algorithm 3.1 is independent of the order of the clients processed, e. g., we can also process clients in order $\mathcal{C}_R, \mathcal{C}_{R-1}..., \mathcal{C}_1$.

Algorithm 3.3 Restatement of the *p-th* Phase Algorithm

```
(1) At the beginning, all clients are unconnected, i. e., $t \leftarrow 0, U \leftarrow \mathcal{C}_p$. Assume client
    $j \in U$ has $r_j$ ports each with some credit which increases from zero simultaneously
    with time before the port is connected.
(2) While $U \neq \phi$, increase time $t$ until an instance of Event-1 or Event-2 or Event-3
    occurs. If two events occur at the same time, process them in an arbitrary order.
    (a) Event-1: A client $j \in U$ has enough credit to be connected with an eligible site,
        suppose $i$, i. e. $t = c_{ij}$ and $X_{ij}^{p-1} < Y_i^{p-1}$. Set $X_{ij}^p \leftarrow X_{ij}^{p-1} + 1$ in this case.
    (b) Event-2: A site $i \in \mathcal{F}$ receives enough payment from clients in $U$ to open its
        p-th replica, i. e. $\sum_{j \in U} \max(t - c_{ij}, 0) = f_i$. In this case, let $C' = \{j \in U :
        c_{ij} \leq t\}$, set $Y_i^p \leftarrow Y_i^{p-1} + 1$ and $X_{ij}^p \leftarrow X_{ij}^{p-1} + 1$ for any $j \in C'$.
    (c) Event-3: A client $j \in U$ has enough credit to be connected with a new opened
        replica, i. e. $t = c_{ij}$. Set $X_{ij}^p \leftarrow X_{ij}^{p-1} + 1$ in this case.
    (d) For any client $j \in U$, set $\alpha_j^p \leftarrow t$ and remove client $j$ from $U$ if it is connected
        with a replica in phase $p$.
```

3.2.2 Inverse Dual Fitting Based Analysis

3.2.2.1 High level analysis

In order to show the performance of Algorithm 3.1, we claim that the maximum cost ratio in each phase is bounded by a constant for any instance of the problem. Formally, let $\mathcal{I}(\mathcal{F}, \mathcal{C}, \boldsymbol{f}, \boldsymbol{c}, \boldsymbol{r})$ be an instance of the *FTFA* problem and $p \in \mathcal{R}$ a phase when solving the problem, we define the maximum cost ratio with respect to any possible star (i, p, C') as

$$\lambda_{\mathcal{I}} = \max_{i \in \mathcal{F}, p \in \mathcal{R} C' \subseteq \mathcal{C}_p} \frac{\sum_{j \in C'} \alpha_j^p}{f_i + \sum_{j \in C'} c_{ij}}.$$

Claim 3.1. The cost of the solution in each phase is equal to $\sum_{j \in \mathcal{C}_p} \alpha_j^p$ and the maximum cost ratio $\lambda_{\mathcal{I}}$ is bounded by a constant λ for any instance $\mathcal{I}$ of the problem.

We use the inverse dual fitting technique here to analyze the approximation factor of the algorithm. We do this by composing an extra instance of the problem which has the same size as the original problem but different values of facility cost and connection cost. We achieve this by scaling the facility cost and connection cost by constant λ: $f_i' \leftarrow \lambda f_i$ and $c_{ij}' \leftarrow \lambda c_{ij}$. Instead of shrinking the dual variable as in the dual fitting [84, 53, 52] technique to achieve a feasible solution to the unscaled dual problem, we use the unshrunk dual solution which is feasible to the composed instance of the dual problem and achieve a λ-approximation factor based on Claim 3.1. It is not hard to see that the same result can be achieved by using the dual fitting technique. However, we argue that our inverse dual fitting technique is more powerful in multi-factor approximation analysis as shown in the next section.

Theorem 3.1. *If the algorithm for the p-th phase satisfies Claim 3.1, Algorithm 3.1 is a λ-approximation algorithm to the FTFA problem.*

Proof. First we check the feasibility of the solution. According to the subroutine (the algorithm for the *p-th* phase) in each phase, we have $\forall i \in \mathcal{F}, j \in \mathcal{C}_p : X_{ij}^p \leq Y_i^p$. In fact, this is required by the subproblem (3.2) in each phase. It is not hard to see that X_{ij}^p stops increasing when $p > r_j$ because a client j is included in $\mathcal{C}_p$ only when $p \leq r_j$ and not yet processed when $p > r_j$. Therefore, we have $\forall i \in \mathcal{F}, j \in \mathcal{C} : X_{ij}^R \leq Y_i^R$ because Y_i^p is increasing monotonously (we never close a replica). The feasibility of the solution is proved.

In order to show the cost ratio, we compose an extra instance of the problem and its feasible dual solution. Let $\beta_{ij}^p = \max(\alpha_j^p - \lambda c_{ij}, 0)$ for any $i \in \mathcal{F}, j \in \mathcal{C}, p \in \mathcal{R}$ and $C' = \{j \in \mathcal{C} : \alpha_j^p \geq \lambda c_{ij}\}$, we have $\sum_{j\in\mathcal{C}} \beta_{ij}^p = \sum_{j\in C'} \beta_{ij}^p = \sum_{j\in C'} (\alpha_j^p - \lambda c_{ij})$. According to Claim 3.1, we have

$$\sum_{j\in C'} (\alpha_j^p - \lambda c_{ij}) \leq \lambda f_i$$

for any $i \in \mathcal{F}, p \in \mathcal{R}$. That is, there exist dual variables $\beta_{ij}^p \geq 0$ such that

$$\forall i \in \mathcal{F}, p \in \mathcal{R} : \textstyle\sum_{j\in\mathcal{C}} \beta_{ij}^p \leq \lambda f_i \tag{3.5}$$

$$\text{and} \quad \forall i \in \mathcal{F}, j \in \mathcal{C}, p \in \mathcal{R} : \alpha_j^p - \beta_{ij}^p \leq \lambda c_{ij}. \tag{3.6}$$

We note that the above inequalities are exactly the constraints of the dual problem (3.3). Therefore, we can compose an instance of the *FTFA*, suppose $\mathcal{I}'$, with facility cost $f_i' = \lambda f_i$ and connection cost $c_{ij}' = \lambda c_{ij}$. Let OPT_2 be the optimal solution to the primal problem of $\mathcal{I}'$, and OPT_1 the optimal solution to the primal problem of $\mathcal{I}$. It is clear that

$$OPT_2 = \lambda OPT_1. \tag{3.7}$$

From Inequality (3.5) and (3.6), we know (α, β) is a feasible solution to the dual problem of $\mathcal{I}'$. Due to the weak duality theorem, which states that the optimum of the dual problem (in a form of maximization problem) is no more than the optimum of the primal problem (in a form of minimization problem), we have

$$\sum_{j \in \mathcal{C}} \sum_{p=1}^{r_j} \alpha_j^p \leq OPT_2. \tag{3.8}$$

On the other hand, let SOL be the solution derived by the algorithm, we have

$$SOL = \sum_{p=1}^{R} \sum_{j \in \mathcal{C}_p} \alpha_j^p \tag{3.9}$$

according to the first part of the claim. Combining (3.7), (3.8) and (3.9), we have

$$SOL \leq \lambda OPT_1$$

because $\sum_{p=1}^{R} \sum_{j \in \mathcal{C}_p} \alpha_j^p = \sum_{j \in \mathcal{C}} \sum_{p=1}^{r_j} \alpha_j^p$. The theorem is established. □

Remark 3.2. The same result can be achieved by using dual fitting, i. e. shrinking the dual solution λ times to make it 'fit' the original problem. Another interesting observation is that the greedy algorithm for the *UFL* problem, like the MMS algorithm [84, 52] or JMS algorithm [53, 52], can also be analyzed through decomposing the optimal solution into a group of stars. This is true because any solution to *UFL* can be decomposed into stars without overlap or interference. However in the *FTFA* problem, a client is involved in multiple stars and how to assign their costs to achieve the balance between regarding stars is a nontrivial task. Fortunately, by using the (inverse) dual fitting technique, it provides an alternative approach to reveal the approximation factor.

From Algorithm 3.1, we can see that all payments are either for connection cost or facility operating cost. Therefore the first part of the claim is complied by the algorithm.

Now, we only need to find a proper value of $\lambda \geq 1$ such that for any instance $\mathcal{I}$ of the *FTFA* problem

$$\max_{i \in \mathcal{F}, p \in \mathcal{R}, C' \subseteq \mathcal{C}_p} \frac{\sum_{j \in C'} \alpha_j^p}{f_i + \sum_{j \in C'} c_{ij}} \leq \lambda.$$

It is clear that we only need to consider clients with $\alpha_j^p \geq \lambda c_{ij}$. Without loss of generality, suppose there are k clients in $\mathcal{C}_p$ and further $\alpha_1^p \leq \alpha_2^p \leq ... \leq \alpha_k^p$. We consider some important properties of the algorithm for the *p-th* phase before finding a proper value of λ.

3.2.2.2 The *p-th* phase

First, we have the following lemma on the contribution received by a site according to event-2 and event-3.

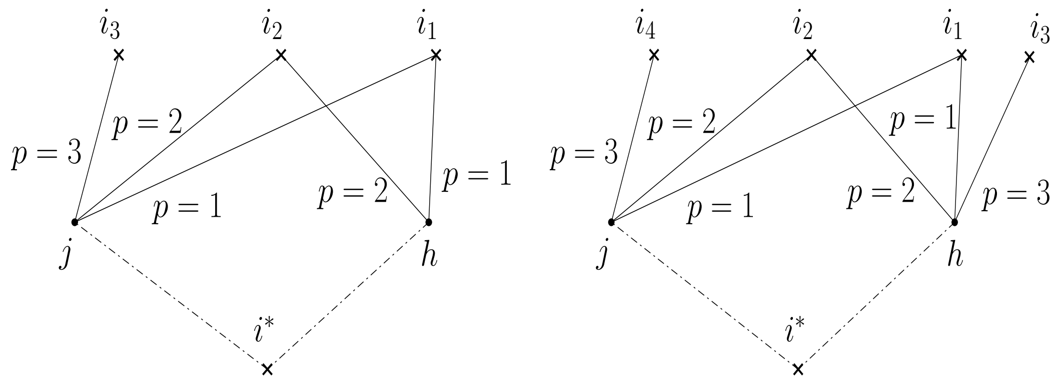
Lemma 3.1. For any instance $\mathcal{I}$ and phase $p \in \mathcal{R}$, $\sum_{j=h}^{k} \max(\alpha_h^p - c_{ij}, 0) \leq f_i$ for any site $i \in \mathcal{F}$ and any client $h, 1 \leq h \leq k$.

Proof. Assume $\sum_{j=h}^{k} \max(\alpha_h^p - c_{ij}, 0) > f_i$, then a new replica at site i must be opened at time $t = \alpha_h^p - \epsilon$ according to event-2 because any j with $\alpha_j^p \geq \alpha_h^p$ is still contributing to open replicas at time t. According to the assumption, there is at least one client, suppose j', such that

$$\alpha_{j'}^p \geq \alpha_h^p \text{ and } \alpha_h^p > c_{ij'}.$$

That is, $\alpha_{j'}^p > c_{ij'}$. Actually, j' can be connected with site i at least at time t according to event-3 of the algorithm which implies $\alpha_{j'}^p \leq c_{ij'}$. The contradiction establishes the lemma. $\square$

It is natural to follow the approach proposed by Mahdian *et al.* [52, 84] to obtain a property regarding the triangle inequality. However in the fault-tolerant context, this becomes more complex. In fact, we are not able to conclude that a contribution is less than the connection cost to any open replica. As shown by the first graph of Figure 3.2(a), neither $\alpha_j^3 \leq c_{i_1 j}$ or $\alpha_j^3 \leq c_{i_2 j}$ can be achieved even if i_1 and i_2 are opened. This is because h is already connected with replica i_1 and i_2 before making its third contribution. Fortunately, in our algorithm, only ports of the same rank are processed in a phase and this makes an important difference. In fact, if there are p open replicas, the *p-th* contribution of a client is no more than the maximum connection cost from the client to these replicas. As shown by the second graph of Figure 3.2(b), $\alpha_j^3 \leq c_{i_3 j}$. Formally, we have the following lemma.



(a) $p = 3, r_h = 2$ (b) $p = 2, r_h = 2$

Figure 3.2: Ranking of Contributions

Lemma 3.2. For any instance $\mathcal{I}$ and phase $p \in \mathcal{R}$, $\alpha_j^p \leq \alpha_h^p + c_{ij} + c_{ih}$ for any site $i \in \mathcal{F}$, clients h and $j, 1 \leq h, j \leq k$.

Proof. Assume $\alpha_j^p > \alpha_h^p$, otherwise the lemma is obvious. Let H be the set of replicas that are connected with client h at time $t = \alpha_j^p - \epsilon$, so we have $|H| = p$ because h is already connected in the *p-th* phase. Hence, there must exists a replica among H which is not connected with j at the moment in phase p. Suppose this is a replica at site i', we have $X_{i'j}^{p-1} < Y_{i'}^p$. Therefore j can be connected with i' without paying operating cost. Considering two cases, i. e., the replica is respectively opened in an early phase or opened in phase p, we have $\alpha_j^p \leq c_{i'j}$ for both cases due to event-1 and event-3. Further we have $\alpha_j^p \leq c_{ij}$ if $i = i'$; otherwise combining the triangle inequality $c_{i'j} \leq c_{i'h} + c_{ij} + c_{ih}$, we have $\alpha_j^p \leq \alpha_h^p + c_{ij} + c_{ih}$ because $\alpha_h^p \geq c_{i'h}$ for any i' connected with client h. The lemma follows. $\square$

3.2.2.3 Performance ratio

The above lemmas present some important properties of the algorithm for the *p-th* phase and the following result turns out that they are enough to bound the ratio of the total cost of a derived solution to that of an optimal solution. Let λ_k be the maximum of the following LP

$$\begin{array}{ll} \text{maximize} & \frac{\sum_{j=1}^{k}\alpha_j}{f+\sum_{j=1}^{k}c_j} \\ \text{subject to} & \forall 1 \le j < h \le k:\ \alpha_h \le \alpha_j + c_j + c_h \\ & \forall 1 \le h \le k:\ \sum_{j=h}^{k}\max(\alpha_h - c_j, 0) \le f \\ & \forall 1 \le j \le k:\ \alpha_j, c_j, f \ge 0. \end{array} \tag{3.10}$$

If λ_k has an upper bound with respect to any integer k, we are able to choose this upper bound as the value of λ with respect to the claim. LPs like program 3.10 are also called factor revealing LPs in the literature [83, 84, 53, 52].

Corollary 3.1. *Algorithm 3.1 is a 1.861-approximation algorithm for the metric FTFA problem.*

Proof. Let α_j^p be denoted by α_j , c_{ij} by c_j, and f_i by f, it is clear that Lemma 3.1 and Lemma 3.2 imply the two constraints of program 3.10. As a result, we have

$$\lambda \le \sup_{k\ge 1}\{\lambda_k\}.$$

In fact, Mahdian *et al.* [84, 52] showed that program 3.10 has an upper bound 1.861 in their analysis for the MMS algorithm. Combined with Theorem 3.1, the corollary follows. □

In each phase, there are at most $n_f \cdot |\mathcal{C}_p|$ events for which the algorithm needs $n_f \cdot |\mathcal{C}_p| \log(n_f \cdot |\mathcal{C}_p|)$ time to sort these events. Considering that the algorithm runs R phases and in each phase $|\mathcal{C}_p| \le |\mathcal{C}|$, we have the following lemma.

Lemma 3.3. *Time complexity of Algorithm 3.1 is $O(mR\log m)$, where $m = n_c n_f$.*

3.3 Bi-Factor Approximation and Extension

Interesting enough, the dual fitting technique is not used except the single-factor approximation analysis (i. e. 1.861- and 1.61- approximation) as presented in [84, 52, 53]. In fact, the JMS algorithm is (1.11,1.78)-approximation [80] and (1,2)-approximation [53, 52] at the same time for the *UFL* problem. However, these results were achieved through proving an upper bound of total cost paid by all clients in any possible star s, i. e. $\lambda_f f_i + \lambda_c \sum_{j \in s \cap \mathcal{C}} c_{ij}$, rather than by applying the dual fitting technique. This approach is straightforward to the *UFL* problem because a solution can be decomposed into a group of stars without overlap which is not true in the fault tolerant variants of the problem. In this section, we demonstrate an alternative approach — inverse dual fitting based bi-factor approximation, to achieve similar results for the *FTFA* problem.

3.3.1 The Algorithm

We note that once a client in Algorithm 3.1 is fully-connected, it is not processed any more even there is a facility (replica) opened with a smaller connection cost. It is obvious that, we are able to improve the algorithm by establishing connections, for each client, to replicas with smallest connection costs. We do this by switching two connections of a client, an old one with higher connection cost and a new one with smaller connection cost. Considering the reduction in total cost by connection switching, we redefine the cost efficiency of a star at the time before the new star is selected by

$$\text{eff}(i, p, C') = \frac{f_i^p + \sum_{j \in C'} c_{ij} - \sum_{j \in \mathcal{C}_p \setminus U} \max(c_{i'j} - c_{ij}, 0)}{|C'|}, \tag{3.11}$$

where $\mathcal{C}_p \setminus U$ is the set of clients which are already connected in phase p and $c_{i'j}$ the maximum connection cost of client j. The first two items in the numerator represent the total cost of the star which is the same as in Algorithm 3.1. The third item is the contribution made by connected clients via connection exchange.

The new algorithm has the same structure as Algorithm 3.1, with a new cost efficiency used in the subroutine as redefined by (3.11). We also use the same interpretation of dual variables as in Algorithm 3.1. So, the most cost efficient star in each iteration of the algorithm can be found in a similar way: if the dual variables of all unconnected clients are raised simultaneously, the most cost efficient star will be the first star (i, p, C') for which

$$\sum_{j\in C'} \max(t - c_{ij}, 0) + \sum_{j\in \mathcal{C}_p\setminus U} \max(c_{i'j} - c_{ij}) = f_i^p.$$

When $f_i^p = 0$, the improved algorithm for the *p-th* phase is the same as the original. When $f_i^p = f_i$ it receives payment from unconnected clients as well as payment from connected clients through connection switch and once its amount is equal to the facility cost, it opens a new replica at site i. Same as Algorithm 3.1, the new algorithm is also stated through three types of events given in Algorithm 3.4.

Algorithm 3.4 Multi-Factor Approximation Algorithm

```
(1) Initially set vector $\boldsymbol{X}^0 \leftarrow \boldsymbol{0}, \boldsymbol{Y}^0 \leftarrow \boldsymbol{0}$ and the number of current phase $p \leftarrow 1$.
(2) While $p \le R$:
    (a) Invoke the improved algorithm for the p-th phase with input
        $(\boldsymbol{X}^{p-1}, \boldsymbol{Y}^{p-1}, \mathcal{F}, \mathcal{C}_p)$, suppose the output is $(\boldsymbol{X}^p, \boldsymbol{Y}^p)$.
    (b) Set $p \leftarrow p + 1$.
(3) Set $\boldsymbol{x} \leftarrow \boldsymbol{X}^R$ and $\boldsymbol{y} \leftarrow \boldsymbol{Y}^R$
```

Algorithm 3.5 Improved Algorithm for the *p-th* Phase

```
(1) At the beginning, all clients are unconnected: $t \leftarrow 0, U \leftarrow \mathcal{C}_p$.
(2) While $U \neq \phi$, increase time $t$ until an instance of Event-1 or Event-2 or Event-3
    occurs. If two events occur at the same time, process them in an arbitrary order.
    (a) Event-1: A client $j \in U$ has enough credit to be connected with an eligible site,
        suppose $i$, i.e. $t = c_{ij}$ and $X_{ij}^{p-1} < Y_i^{p-1}$. Set $X_{ij}^p \leftarrow X_{ij}^{p-1} + 1$ in this case.
    (b) Event-2: A site $i \in \mathcal{F}$ receives enough credit from clients in $U$ to open its
        p-th replica, i.e. $\sum_{j\in U} \max(t - c_{ij}, 0) + \sum_{j\in \mathcal{C}_p\setminus U} \max(c_{i'j} - c_{ij}) = f_i$. In
        this case, let $C'_1 = \{j \in U : c_{ij} \le t\}$ and $C'_2 = \{j \in \mathcal{C}_p \setminus U : c_{ij} \le c_{i'j}\}$,
        set $Y_i^p \leftarrow Y_i^{p-1} + 1$ and $X_{ij}^p \leftarrow X_{ij}^{p-1} + 1$ for any $j \in C'_1$ or $j \in C'_2$, set
        $X_{ij}^p \leftarrow X_{ij}^{p-1} - 1$ for any $j \in C'_2$.
    (c) Event-3: A client $j \in U$ has enough credit to be connected with the new opened
        replica, i.e. $t = c_{ij}$. Set $X_{ij}^p \leftarrow X_{ij}^{p-1} + 1$.
    (d) For any client $j \in U$, set $\alpha_j^p \leftarrow t$ and remove client $j$ from $U$ if it is connected
        with a replica in phase $p$.
```

According to the subroutine in each phase, we have $\forall i \in \mathcal{F}, j \in \mathcal{C}_p : X_{ij}^p \le Y_i^p$ as required by the subproblem in each phase. It is not hard to see that X_{ij}^p stops increasing when $p > r_j$ because a client j is included in $\mathcal{C}_p$ only when $p \leq r_j$ and not been processed when $p > r_j$. Therefore, we have $\forall i \in \mathcal{F}, j \in \mathcal{C} : X_{ij}^R \leq Y_i^R$ because Y_i^p is increasing monotonously (we never close a replica). Feasibility of the solution is ensured.

In Algorithm 3.4, the amount of credit paid for a connection can be divided further — part for a connection with a smaller cost, remaining for opening other replicas. Despite of the difference to Algorithm 3.1, it is still true that all payments of a client are either used to open replicas or to establish connections, therefore the total cost of the solution is still $\sum_{j\in\mathcal{C}} \sum_{p=1}^{r_j} \alpha_j^p$. This results in the following lemma.

Lemma 3.4. *A solution produced by Algorithm 3.4 is feasible to the FTFA problem and its total cost is equal to* $\sum_{j\in\mathcal{C}} \sum_{p=1}^{r_j} \alpha_j^p$.

3.3.2 Inverse Dual Fitting Based Analysis

Let $\lambda_f \geq 1$ be a constant (to be fixed later), and define the maximum connection cost ratio with respect to any possible star (i, p, C') as

$$\lambda'_{\mathcal{I}} = \max_{i\in\mathcal{F}, p\in\mathcal{R}, C'\subseteq\mathcal{C}_p} \frac{\sum_{j\in C'} \alpha_j^p - \lambda_f \cdot f_i}{\sum_{j\in C'} c_{ij}}.$$

Claim 3.2. The maximum cost ratio $\lambda'_{\mathcal{I}}$ is bounded by a constant λ_c for any instance $\mathcal{I}$ of the *FTFA* problem.

Theorem 3.2. *If the improved algorithm for the p-th phase satisfies Claim 3.2, Algorithm 3.4 produces a solution within cost $\lambda_f F^* + \lambda_c C^*$, where F^* and C^* are respectively the facility cost and connection cost of an optimal solution to the FTFA problem.*

Proof. Let $\beta_{ij}^p = \max(\alpha_j^p - \lambda_c c_{ij}, 0)$ for any $i \in \mathcal{F}, j \in \mathcal{C}, p \in \mathcal{R}$ and $C' = \{j \in \mathcal{C} : \alpha_j^p \geq \lambda_c c_{ij}\}$, we have $\sum_{j\in\mathcal{C}} \beta_{ij}^p = \sum_{j\in C'} \beta_{ij}^p = \sum_{j\in C'} (\alpha_j^p - \lambda_c c_{ij})$. According to the Claim 3.2, we have

$$\sum_{j\in C'} (\alpha_j^p - \lambda_c c_{ij}) \leq \lambda_f f_i$$

for any $i \in \mathcal{F}, p \in \mathcal{R}$. That is, there exist dual variables $\beta_{ij}^p \geq 0$ such that

$$\forall i \in \mathcal{F}, p \in \mathcal{R} : \textstyle\sum_{j \in \mathcal{C}} \beta_{ij}^p \leq \lambda_f f_i \tag{3.12}$$

$$\text{and} \quad \forall i \in \mathcal{F}, j \in \mathcal{C}, p \in \mathcal{R} : \alpha_j^p - \beta_{ij}^p \leq \lambda_c c_{ij}. \tag{3.13}$$

We note that the above inequalities share the same form as the constraints of the dual problem (3.3). Therefore, we can compose an instance of the *FTFA*, suppose $\mathcal{I}'$, with facility cost $f_i' = \lambda_f f_i$ and connection cost $c_{ij}' = \lambda_c c_{ij}$. Let OPT_2 be the optimal solution to the primal problem of $\mathcal{I}'$, and OPT_1 the optimal solution to the primal problem of $\mathcal{I}$. It is clear that

$$OPT_2 \leq \lambda_f F^* + \lambda_c C^* \tag{3.14}$$

because the optimal solution to $\mathcal{I}$ is also a feasible solution to $\mathcal{I}'$ (its cost is equal to the left side of the above inequality). From Inequality (3.12) and (3.13), we know (α, β) is a feasible solution to the dual problem of $\mathcal{I}'$. Due to the weak duality theorem, which states that the maximum of the dual problem is no more than the minimum of the primal problem, we have

$$\sum_{j \in \mathcal{C}} \sum_{p=1}^{r_j} \alpha_j^p \leq OPT_2. \tag{3.15}$$

On the other hand, let SOL be the solution produced by the algorithm, we have

$$SOL = \sum_{j \in \mathcal{C}} \sum_{p=1}^{r_j} \alpha_j^p \tag{3.16}$$

according to Lemma 3.4. Combine Inequality (3.14), (3.15) and (3.16), we have

$$SOL \leq \lambda_f F^* + \lambda_c C^*.$$

And the theorem follows. □

Remark 3.3. As far as we know, dual fitting technique is not effective for deriving similar results in multi-factor approximation even for the *UFL* problem. Instead, related results [53, 52] are achieved through a combinatorial approach by decomposing a solution into a group of stars and proving an upper bound of the total cost paid by all clients in each star. This approach also works here for the *FTFA* problem with an extra step to assign the multiple costs paid by a client deliberately to stars. However, our inverse dual fitting technique can simplify algorithm design and explain the dual variables intuitively.

For a *FTFA* problem, we say a solution is (λ_f, λ_c)-approximation to *FTFA* if its cost is no more than $\lambda_f F^* + \lambda_c C^*$, where F^* and C^* are respectively the facility cost and connection cost of any optimal solution. Now, we only need to find a proper value of $\lambda_c \geq 1$ such that for any instance $\mathcal{I}$ of the *FTFA* problem

$$\max_{i\in\mathcal{F},p\in\mathcal{R},C'\subseteq\mathcal{C}_p} \frac{\sum_{j\in C'} \alpha_j^p - \lambda_f f_i}{\sum_{j\in C'} c_{ij}} \leq \lambda_c.$$

Again, we only need to consider clients with $\alpha_j^p \geq \lambda_c c_{ij}$. Without loss of generality, suppose there are k clients in $\mathcal{C}_p$ and further $\alpha_1^p \leq \alpha_2^p \leq ... \leq \alpha_k^p$. We consider some important properties of the improved algorithm for the *p-th* phase before finding a proper value of λ_c.

Consider time $t = \alpha_h^p - \epsilon$ ($\epsilon \longrightarrow 0$) and define

$$u_{jh} = \begin{cases} t & \alpha_j^p = \alpha_h^p \\ c_{i^*j} & \alpha_j^p < \alpha_h^p \end{cases}$$

for any $1 \leq j \leq h \leq k$, where c_{i^*j} is the maximum connection cost of client j at time t. We have the following properties for Algorithm 3.4.

Lemma 3.5. For a given instance $\mathcal{I}$ and any phase $p \in \mathcal{R}$, $\alpha_h^p \leq u_{jh} + c_{ij} + c_{ih}$ for any $1 \leq j < h \leq k$.

Proof. If $\alpha_j^p = \alpha_h^p$, $u_{jh} \rightarrow \alpha_h^p$ according to the definition, the inequality is obvious; otherwise $\alpha_j^p < \alpha_h^p$. Let H be the set of replicas connected with client j at the moment t, we have $|H| = p$. Hence, there must exists a replica in H which is not connected with h. Suppose it is a replica at site i', we have $X_{i'h}^{p-1} < Y_{i'}^p$. Therefore h can be connected with i', without paying the operating cost. Considering two cases when the replica is opened in a previous phase and in phase p respectively, we have $\alpha_h^p \leq c_{i'h}$ for both cases due to event-1 and event-3. Further we have $\alpha_h^p \leq c_{ih}$ if $i = i'$; otherwise combining the triangle inequality $c_{i'h} \leq c_{i'j} + c_{ij} + c_{ih}$, immediately yields $\alpha_h^p \leq u_{jh} + c_{ij} + c_{ih}$ because u_{jh} is the maximum connection cost of client j at time t when $\alpha_j^p < \alpha_h^p$. The lemma follows. □

At time $t = \alpha_h^p$, the amount of contribution that client j offers to open a replica at site i is equal to

$$\begin{aligned} &\max(u_{jh} - c_{ij}, 0) \quad \text{if } j < h, \text{and} \\ &\max(\alpha_h^p - c_{ij}, 0) \quad \text{if } j \geq h. \end{aligned}$$

Notice that by the definition of u_{jh} this holds even if $\alpha_j^p = \alpha_h^p$. It is clear that the total offer of clients to a site can never become larger than the operating cost at this site. Therefore, we have $\sum_{j=1}^{h-1} \max(u_{jh} - c_{ij}, 0) + \sum_{j=h}^{k} \max(\alpha_h^p - c_{ij}, 0) \leq f_i$. This results in the following lemma.

Lemma 3.6. For a given instance $\mathcal{I}$ and any phase $p \in \mathcal{R}$, $\sum_{j=1}^{h-1} \max(u_{jh} - c_{ij}, 0) + \sum_{j=h}^{k} \max(\alpha_h^p - c_{ij}, 0) \leq f_i$ for any $1 \leq h \leq k$.

The above lemmas presents some properties of Algorithm 3.4 and the following theorem shows that they are enough to prove Claim 3.2. Let λ_c^k be the maximum of the following Factor Revealing LP

$$\begin{aligned} \text{maximize} \quad & \frac{\sum_{j=1}^{k} \alpha_j - \lambda_f \cdot f}{\sum_{j=1}^{k} c_j} \\ \text{subject to} \quad & \forall 1 \leq j \leq h \leq k: \ \alpha_h \leq u_{jh} + c_j + c_h \\ & \forall 1 \leq h \leq k: \ \sum_{j=1}^{h-1} \max(u_{jh} - c_j, 0) + \\ & \sum_{j=h}^{k} \max(\alpha_h - c_j, 0) \leq f \\ & \forall 1 \leq j \leq h \leq k: \ \alpha_j, c_j, u_{jh}, f \geq 0. \end{aligned} \tag{3.17}$$

It is clear that Lemma 3.5 and Lemma 3.6 imply the two constraints of program 3.17. As a result, we have

$$\lambda_c \leq \sup_{k \geq 1} \{\lambda_c^k\}$$

regarding to Claim 3.2. Actually, for different $\lambda_f \geq 1$, there is a unique upper bound $\sup_{k\geq 1}\{\lambda_c^k\}$ as shown on the approximation curve in [52]. Furthermore, we have the following theorem according to existing results on the Factor Revealing LP.

Theorem 3.3. *For λ_c defined in Claim 3.2, we have:*

1) if $\lambda_f = 1.61$, then $\lambda_c \leq 1.61$ (see [53]);

2) if $\lambda_f = 1.11$, then $\lambda_c \leq 1.78$ (see [80]);

3) if $\lambda_f = 1$, then $\lambda_c \leq 2$ (see [52]).

Proof. Proofs of these results can be obtained directly from the cited references. Here, for the third result we give an alternative proof which is simpler than that given in [52]. We first relax the second constraint as

$$\forall 1 \leq h \leq k: \sum_{j=1}^{h-1}(u_{jh} - c_j) + \sum_{j=h}^{k}(\alpha_h - c_j) \leq f.$$

According to the first constraint, we are able to use $\alpha_h - c_j - c_h$ to replace u_{jh} in the above inequality. After moving some items to the right side, we have

$$\forall 1 \leq h \leq k: \sum_{j=1}^{k}(\alpha_h - c_j) \leq f + \sum_{j=1}^{h-1}(c_j + c_h).$$

For the above inequality combining all cases for $1 \leq h \leq k$, we have

$$\sum_{h=1}^{k}(k\alpha_h - kc_h) \leq k \cdot f + \sum_{h=1}^{k}\sum_{j=1}^{h-1}(c_j + c_h).$$

Noting $\sum_{h=1}^{k}\sum_{j=1}^{h-1}(c_j + c_h) = (k-1)\sum_{h=1}^{k} c_h$, we have

$$k\sum_{h=1}^{k}\alpha_h \leq kf + (2k-1)\sum_{h=1}^{k} c_h,$$

that is

$$\frac{\sum_{j=1}^{k}\alpha_j - f}{\sum_{j=1}^{k} c_j} \leq \frac{2k-1}{k} < 2.$$

This yields the third result. □

From Theorem 3.3 the following corollary is immediate:

Corollary 3.2. *Algorithm 3.4 is a 1.61-, (1.11,1,78)- and (1,2)-approximation algorithm for the* metric FTFA *problem.*

Different from Algorithm 3.1, Algorithm 3.4 has to traverse all *fully-connected* clients for each site, i. e. $\mathcal{C}_p \setminus U$, to know their maximum connection costs. Therefore it needs $O(|\mathcal{C}_p| \cdot n_f)$ time to know the time that the next event occurs. So, totally Algorithm 3.4 needs at most $O(|\mathcal{C}_p| \cdot n_f^2)$ steps to complete each phase because event-2 occurs at most n_f times.

Lemma 3.7. *The time complexity of Algorithm 3.4 is $O(Rn^3)$, where n is the maximum of n_f and n_c.*

3.3.3 Scaling and Greedy Augmentation

Guha *et al.* [112, 21] showed that it is possible to improve the performance of JMS algorithm by using scaling and greedy augmentation. Similarly, we use the same technique to improve Algorithm 3.4. The combined algorithm is as follows.

Algorithm 3.6 1.52-Approximation Algorithm

```
(1) Scale the facility costs by $\delta$: $f_i \leftarrow \delta f_i$ .
(2) Run *Algorithm 3.4* on the scaled instance of fault-tolerant facility allocation problem.
(3) Scale back the facility costs and perform greedy augmentation. Define the gain of
    replica $i$, $gain(i)$, to be the reduction in total cost obtained by adding replica $i$ to the
    current solution ($gain(i) = 0$ if the total cost does not decrease). While there exist
    replicas with positive gains, choose the replica $i$ for which $\frac{gain(i)}{f_i}$ is maximized and
    add it to the current solution.
```

The next lemma was first proved in [112, 21] for the *UFL* problem and then in [48] for the *FTFL* problem. Noting that the *FTFA* problem is a special case of the *FTFL* problem, we have

Lemma 3.8. *[125, 48] Let F^* and C^* be the facility cost and connection cost, respectively, of an optimal solution to the FTFA problem. Greedy augmentation, when applied to a solution with initial facility cost F and connection cost C, produces a solution of cost at most $F + \max\{0, \ln(\frac{C-C^*}{F^*})\} \cdot F^* + F^* + C^*$.*

The above lemma implies the following result:

Lemma 3.9. *[22, 81, 125] If Algorithm 3.4 is a (λ_f, λ_c)-approximation algorithm, Algorithm 3.6 with parameter $\delta \geq 1$, gives a $(\lambda_f + \ln \delta, 1 + \frac{\lambda_c - 1}{\delta})$-approximation solution for any instance of* the *FTFA problem.*

As shown by Mahdian *et al.* [80], we get $\lambda_f + \ln \delta = 1 + \frac{\lambda_c - 1}{\delta} = 1.52$ taking $(\lambda_f, \lambda_c) = (1.11, .178)$ and $\delta = 1.504$, which implies that Algorithm 3.6 is a 1.52-approximation algorithm.

Theorem 3.4. *Algorithm 3.6 is a 1.52-approximation algorithm with running time $O(Rn^3)$ for FTFA .*

3.4 Fault-Tolerant k-Facility Allocation

In this section ,we consider the *Fault-Tolerant k-Facility Allocation* (*FTKFA*) problem which can be seen as a combination of the *k-Median* problem and the *FTFA* problem

The *k-Median* problem [76] has also been studied extensively [6, 21, 23]. This problem requires to open no more than *k* medians in a set of geographically distributed candidate sites and connect each client with the closest open median so that the total connection cost of all clients is minimized. The *k-Facility* problem differs from the *k-Median* problem by considering the specified operating cost for each facility and minimize the combined cost for both facility operating and connection establishing. The *FTKFA* problem is a further generalization of *k-Facility* problem, where the connectivity at each client is not necessarily equal to one. *FTKFA* is also an extension of *FTFA* by applying an extra upper bound on total open facility numbers, i. e. *k*.

Jain and Vazirani [54] reduced the *k-Facility* problem to the *UFL* in the following way: Suppose $\mathcal{A}$ is an approximation algorithm for the facility location problem. Consider an instance $\mathcal{I}$ of the problem with optimum cost OPT, and let F and C be the facility and connection costs of the solution found by $\mathcal{A}$. Algorithm $\mathcal{A}$ is called a Lagrangian Multiplier Preserving λ-approximation (or LMP λ-approximation for short) if for every instance $\mathcal{I}, C/\lambda + F \leq OPT$. Jain and Vazirani [54] proposed that an LMP λ-approximation algorithm for the metric *UFL* problem gives rise to a 2λ-approximation algorithm for the metric k*-Facility* problem. In this chapter, we consider the fault-tolerant version of the problem. Instead of using the concept of LMP λ-approximation, we use bi-factor approximation. We use a (1,λ)-approximation algorithm to *FTFA* as a subroutine to obtain a $(\lambda + \frac{1}{n_f})(2 - \frac{1}{n_f})$ approximation algorithm for the metric *FTKFA* problem. This result is better than 2λ when $\lambda \geq 2$ but worse than 2λ otherwise. Applying the result on bi-factor approximation given in the last section, we know Algorithm 3.4 is (1,2)-approximation to *FTFA* and therefore the result we have has a $4 - 1/n_f^2$ approximation factor for the *FTKFA* problem. The algorithm has the virtue of simplicity and can be completed efficiently in strong polynomial time.

We also assume each client contains r_j ports, let $\mathcal{P}$ denote the set of all ports of all clients. Let s be a star composed of a replica and a group of ports connected with the replica. Let $\mathcal{S}$ be all possible stars and $\mathcal{S}_i$ all possible stars centered at site i. The *FTKFA* problem can be formulated by

$$\begin{array}{ll} \text{minimize} & \sum_{s \in \mathcal{S}} c_s x_s \\ \text{subject to} & \sum_{s \in \mathcal{S}} x_s \leq k \\ & \forall l \in \mathcal{P} : \sum_{s: l \in s} x_s \geq 1 \\ & \forall i \in \mathcal{F}, j \in \mathcal{C} : \sum_{l \in \mathcal{P}_j} \sum_{s: l \in s} x_s \leq \sum_{s \in \mathcal{S}_i} x_s \\ & \forall s \in \mathcal{S} : x_s \in \{0, 1\}. \end{array} \tag{3.18}$$

In the above ILP, $\mathcal{P}_j$ is the set of all ports of client j. The first constraint ensures at most k-replicas are opened in total; the second one at least one connection for each port and the third constraint ensures enough open replicas at each location so that connections between any site-client pair can be assigned to distinct replicas.

Suppose the number of the replicas opened by an algorithm for *FTFA* is k'. It is clear that the solution can be used directly if $k' \leq k$; in the rest of the chapter, we assume $k' > k$. In this case, in order to minimize the total cost, we can always open k replicas, i. e. $\sum_{s \in \mathcal{S}} x_s = k$. Let $\tilde{x}_s$ be the optimal solution of the original problem (with facility cost f_i). We set the cost of operating a facility at site i to $f_i + z$, and let $c_s^- = \sum_{l \in s \cap \mathcal{P}} c_{il}$

and $c_s = c_s^- + f_{i(s)}$. Suppose an algorithm $\mathcal{A}$ is a $(1, \lambda)$-approximation algorithm and it happens to open k replicas, we have

$$\sum_{s \in \mathcal{S}} (c_s + z)x_s \leq \sum_{s \in \mathcal{S}} (f_{i(s)} + z)\tilde{x}_s + \lambda \sum_{s \in \mathcal{S}} c_s^- \tilde{x}_s,$$

and

$$\sum_{s \in \mathcal{S}} x_s = k \geq \sum_{s \in \mathcal{S}} \tilde{x}_s.$$

That is,

$$\sum_{s \in \mathcal{S}} c_s x_s \leq \sum_{s \in \mathcal{S}} (c_s + z)x_s - \sum_{s \in \mathcal{S}} z\tilde{x}_s \leq \sum_{s \in \mathcal{S}} f_{i(s)}\tilde{x}_s + \lambda \sum_{s \in \mathcal{S}} c_s^- \tilde{x}_s \leq \lambda \sum_{s \in \mathcal{S}} c_s \tilde{x}_s. \quad (3.19)$$

We can conclude that the solution is a λ-approximation. However, this result relies on the assumption that the algorithm for *FTKFA* opens exactly k replicas. In the rest of the thesis, we assume such an algorithm does not exist, and instead we combine two solutions with k_1 and k_2 replicas respectively, $k_1 < k < k_2$, to achieve a solution with k replicas.

3.4.1 Bisection Search and Combination

Jain *et al.* proposed an approach to get a 2λ-approximation algorithm for the metric k-*Facility* problem by using an LMP λ-approximation algorithm for the metric *UFL* problem [54]. They achieved a 6-approximation algorithm using a LMP 3-approximation algorithm [56, 54] and further a 4-approximation algorithm for *UFL* in [53]. Their approaches are based on the concept of LMP λ-approximation and as a result need an extra step described in [53] to transform the JMS algorithm which is $(1, \lambda)$-approximation to *UFL* into an LMP λ-approximation algorithm. Our approach simplifies this process by eliminating the middle step and using a $(1, \lambda)$-approximation algorithms directly. Note that our approach is for the fault-tolerant extension of their problem. We first prove that two $(1, \lambda)$-approximation solutions to *FTFA* can be combined to achieve a $(\lambda + \frac{1}{n_f})$-

approximation fractional solution to *FTKFA*. In the next subsection, we will round the fractional solution, losing a small factor.

Consider an algorithm using a bisection search to approximate the value of z , i. e. facility cost $f_i + z$. Let $c_{\max}$ be the maximum of all connection costs, it is clear that $\sum_{s\in\mathcal{S}} x_s = \max_{j\in\mathcal{C}} r_j \leq k$ when $z = n_c c_{\max}$ and $\sum_{s\in\mathcal{S}} x_s \geq k$ when $z = 0$. Instead of using $n_c c_{\max}$ and 0 directly, we find two values of z which are very close and then combine corresponding solutions together. Assume the solutions are $\boldsymbol{x}^1$, $\boldsymbol{x}^2$ respectively for z_1 and z_2, and $\sum_{s\in\mathcal{S}} x_s^1 = k_1$ and $\sum_{s\in\mathcal{S}} x_s^2 = k_2$. The combined solution $\boldsymbol{x} = a\boldsymbol{x}^1 + b\boldsymbol{x}^2$, where $a = (k - k_1)/(k_2 - k_1)$ and $b = (k_2 - k)/(k_2 - k_1)$. Now the problem is how efficient we can find the values of z_1 and z_2 such that they are close enough to ensure the quality of the combined solution and how we can get an integer solution from the combined fractional solution. We have the following lemma for the first problem.

Lemma 3.10. *The cost of the combined solution is within* $(\lambda + \frac{1}{n_f})$ *times of that for an optimal fractional solution to FTKFA if* $z_1 - z_2 \leq \frac{Rf_{\min} + n_c c_{\min}}{kn_f}$.

Proof. Suppose the primal solution and the dual solution derived by Algorithm 3.4 are $(\boldsymbol{x}^1, \boldsymbol{\alpha}^1)$ and $(\boldsymbol{x}^2, \boldsymbol{\alpha}^2)$ respectively. Let $\tilde{x}_s$ be the optimal solution of the original problem (with facility cost f_i). We have

$$\sum_{s\in\mathcal{S}} (c_s + z_1) x_s^1 \leq \sum_{s\in\mathcal{S}} (f_i + z_1)\tilde{x}_s + \lambda \sum_{s\in\mathcal{S}} c_s^- \tilde{x}_s$$

according to the definition of $(1, \lambda)$-approximation. Considering $\sum_{s\in\mathcal{S}} \tilde{x}_s \leq k$, we have

$$\sum_{s\in\mathcal{S}} c_s x_s^1 \leq z_1(k - k_1) + \lambda \sum_{s\in\mathcal{S}} c_s \tilde{x}_s. \tag{3.20}$$

Similarly we have

$$\sum_{s\in\mathcal{S}} c_s x_s^2 \leq z_2(k - k_2) + \lambda \sum_{s\in\mathcal{S}} c_s \tilde{x}_s. \tag{3.21}$$

Now replace z_1 with z_2 in the first item of Inequality (3.20) using the fact that $z_1 - z_2 \leq \frac{Rf_{\min} + n_c c_{\min}}{kn_f} \leq \frac{\sum_{s\in\mathcal{S}} c_s \tilde{x}_s}{kn_f}$, we have

$$\sum_{s\in\mathcal{S}} c_s x_s^1 \le z_2(k-k_1) + (\lambda + \frac{1}{n_f}) \cdot \sum_{s\in\mathcal{S}} c_s \tilde{x}_s, \tag{3.22}$$

Multiplying Inequality (3.21) with constant $a = (k-k_1)/(k_2-k_1)$ and Inequality (3.22) with constant $b = (k_2-k)/(k_2-k_1)$, we have z_2 eliminated, i. e.,

$$\sum_{s\in\mathcal{S}} c_s x_s \le [a\lambda + b(\lambda + \frac{1}{n_f})] \sum_{s\in\mathcal{S}} c_s \tilde{x}_s \le (\lambda + \frac{1}{n_f}) \sum_{s\in\mathcal{S}} c_s \tilde{x}_s.$$

The lemma follows. □

Since the total range of z is $n_c c_{\max}$ and the interval between z_1 and z_2 is required to be less than $(Rf_{\min} + n_c c_{\min})/kn_f$, so the total number of probing steps is $\log \frac{kn_f n_c c_{\max}}{Rf_{\min}+n_c c_{\min}}$. Letting $L = c_{\max}/(Rf_{\min} + n_c c_{\min})$ and $n = \max(n_c, n_f)$, we have the following lemma.

Lemma 3.11. *After $O(\log(nL))$ probe of z using a bisection search, z_1 and z_2 are so close that $z_1 - z_2 \le (Rf_{\min} + nc_{\min})/kn$ and $k_1 \le k \le k_2$.*

We notice that Smamy and Shmoys [125] achieved a similar result. Our solution applies a similar approach to that in [125] but differs in three aspects:

1. Their result only applies the uniform connectivity case for the *Fault Tolerant k-Facility Location* problem where each site allows at most one facility while ours applies for both the uniform case and general case and each site allows unlimited number of replicas.

2. Their bisection search needs $O(\frac{poly(n)}{L'})$ steps, where $L' = \log(c_{\max})$, while ours only needs $O(\log n + \log L)$, where $L = c_{\max}/(Rf_{\min} + nc_{\min})$. This is because, we do not require the corresponding dual solutions are identical for the two primal solutions, as a result the length of search interval is substantially greater.

3. Their approach depends on how to break ties between events in the primal-dual algorithm, while ours does not.

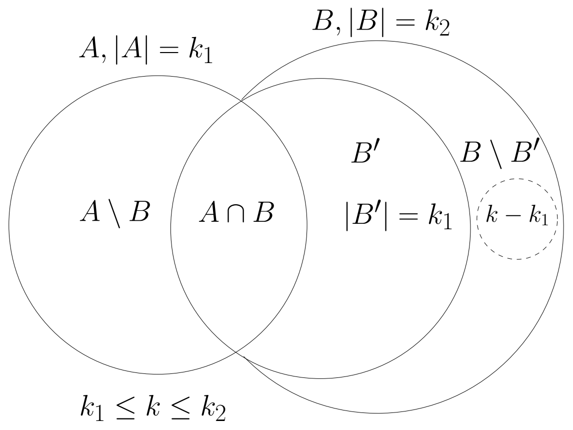


Figure 3.3: Randomized Procedure

3.4.2 Randomized Procedure for Rounding

3.4.2.1 Facility opening

We use the same randomized procedure as in [56] to open replicas. We show that similar result can also be achieved in the fault tolerance context.

Let A and B be the sets of open replicas in the two solutions, $|A| = k_1$ and $|B| = k_2$. For each facility in A, find the closest facility in B, which are not required to be distinct to each other. Let $B' \subset B$ be these replicas. If $|B'| \leq k_1$, arbitrarily include additional replicas from $B \setminus B'$ into B' until $|B'| = k_1$. Now we open all replicas in A with probability a and open all replicas in B' with probability $b = 1 - a$. In addition, a set of cardinality $k - k_1$ is picked randomly from $B \setminus B'$ and replicas in this set are opened. Furthermore, each facility in B is opened with probability b. The procedure is demonstrated in Figure 3.3. For convenience, we use $\hat{y}_i$ and $\hat{x}_{ij}, i \in \mathcal{F}, j \in \mathcal{C}$ to denote the integer solution in which there are totally k open replicas, we have the following lemma.

Lemma 3.12. *The expected facility cost $E[\sum_{i \in \mathcal{F}} f_i \hat{y}_i]$ is no more than $a \sum_{i \in A} f_i x_i + b \sum_{i \in B} f_i x_i$.*

3.4.2.2 Connection establishment

Instead of connecting a client with the r_j nearest open replicas, we consider a suboptimal approach for the sake of approximation factor revealing. The approach is first proposed in [125]. Our analysis follows the same idea but leads to a more strict result because we can only lose a factor $(2 - \frac{1}{n_f})$ instead of 2 to achieve a less than 4 approximation factor. This is because the factor we lost in the last step is $(2 + \frac{1}{n_f})$ for the sake of time complexity. We introduce the procedure briefly as follows. Note that this approach is used only in the analysis and not in the algorithm because the optimal approach, i. e. connecting a client with the r_j nearest open replicas, is always preferred.

Let A_j be the set of replicas in A to which client $j \in \mathcal{C}$ is connected, namely, $A_j = \{i \in A : x_{ij} = 1\}$. Similarly, let B_j be the set of replicas in B that serve j. Clearly $|A_j| = |B_j| = r_j$. For each port l of client j, , we define a set of replicas T_l, and l will only be connected to a facility in T_l. First, we arbitrarily assign each facility $i \in A_j$, and the facility in B_j to which it is matched (which could be the same as i), to a distinct set T_l. Observe the important fact that the sets T_l are disjoint, since distinct replicas in A_j are matched to distinct replicas in B. Let $m(A_j) \subseteq B$ denote the set of replicas that are matched to replicas in A_j. Then $|m(A_j)| = |A_j| = |B_j| \Rightarrow |m(A_j)\backslash B_j| = |B_j\backslash m(A_j)|$, so the number of sets T_l not containing a facility from B_j after the first step is equal to the number of unmatched replicas in B_j . We assign a distinct unmatched facility of B_j to each set T_l which does not already contain a facility from B_j. Note that the sets T_l remain disjoint; so if we connect each port l to a facility in T_l, we will get a feasible solution.

Lemma 3.13. *After the above randomized procedure, a client j is connected with r_j distinct replicas.*

Furthermore, we have the following lemma on the connection cost.

Lemma 3.14. *The expected connection cost for a client j, i. e. $E[cost(j)]$ is no more than $(1 + \max(a, b)) \sum_{i\in\mathcal{F}} c_{ij}x_{ij}$.*

Proof. For convenience, if facility $i \in A_j$ is matched with i', we will consider i and i' as two different replicas even if $i = i'$. Let the service cost of client $j \in \mathcal{C}$ be $cost(j)$ and the service cost of port l be $cost(l)$. The set T_l contains at least one small facility $i_1 \in A_j$ and one large facility i_2 such that i_1 is matched to i_2.

If these are the only two replicas then it must be that $i_2 \in B_j$. Either i_1 or i_2 is open, and we assign l to that open facility. So $E[cost(l)] = ac_{i_1j} + bc_{i_2j}$.

Otherwise, T_l contains a third facility $i_3 \in B_j$ such that i_3 is unmatched and $i_2 \in B_j$ .We assign l to i_3 if it is open, and to i_1 or i_2, whichever is open, otherwise. So $E[cost(l)] = a(ac_{i_1j} + bc_{i_2j}) + bc_{i_3j}$. Since i_1 is matched with i_2 and i_3 is unmatched, it must be that i_2 is closer to i_1 than i_3. So,

$$c_{i_2j} \leq c_{i_1j} + c_{i_1i_2} \leq c_{i_1j} + c_{i_1i_3} \leq 2c_{i_1j} + c_{i_3j}.$$

Therefore

$$\begin{aligned} E[cost(l)] &\leq bc_{i_3j} + a(ac_{i_1j} + 2bc_{i_1j} + bc_{i_3j}) \\ &= a(1+b)c_{i_1j} + b(1+a)c_{i_3j}. \end{aligned}$$

Thus for every port l, if $i, i' \in T_l$, where $i \in A_j$ and $i' \in B_j$, we have

$$E[cost(l)] \leq a(1+b)c_{ij} + b(1+a)c_{i'j}).$$

For both cases, we have $E[cost(l)] \leq (1 + \max(a,b))(ac_{ij}x^1_{ij} + bc_{i'j}x^2_{i'j})$, since $x^1_{ij} = x^2_{i'j} = 1$. So, summing up the costs for all ports l, since the set of all replicas i for the first component of the last item is precisely A_j and the set of all replicas i for the second component is the set B_j, we get

$$\begin{aligned} E[cost(j)] &\leq (1 + \max(a,b))(\sum_{i \in A_j} ac_{ij}x^1_{ij} + \sum_{i \in B_j} bc_{ij}x^2_{ij}) \\ &= (1 + \max(a,b)) \sum_{i \in \mathcal{F}} c_{ij}(ax^1_{ij} + bx^2_{ij}), \end{aligned}$$

where the last equality holds since $x^1_{ij} = 0$ if $i \notin A_j$ and $x^2_{ij} = 0$ if $i \notin B_j$. The lemma follows because $x_{ij} = (ax^1_{ij} + bx^2_{ij})$. $\square$

3.4.2.3 Approximation factor

We have the following theorem.

Theorem 3.5. *A* $(1,\lambda)$*-approximation algorithm for FTFA can result in a* $(\lambda+\frac{1}{n_f})(2-\frac{1}{n_f})$*-approximation stochastic algorithm for* FTKFA.

Proof. According to Lemma 3.14, we have

$$E[\sum_{i\in\mathcal{F}}\sum_{j\in\mathcal{C}} c_{ij}\hat{x}_{ij}] \le (1+\max(a,b))\sum_{i\in\mathcal{F}}\sum_{j\in\mathcal{C}} c_{ij}x_{ij}.$$

According to Lemma 3.12, we have

$$E[\sum_{i\in\mathcal{F}} f_i\hat{y}_i] \le (a+b)\sum_{i\in A\cup B} f_iy_i \le (1+\max(a,b))\sum_{i\in\mathcal{F}} f_iy_i.$$

Combining them together, we have

$$E[\sum_{s\in\mathcal{S}} c_s\hat{x}_s] \le (1+\max(a,b))\sum_{s\in\mathcal{S}} c_sx_s.$$

On the other hand, its easy to see that $a \le 1-1/n_f$ (this happens for $k_1 = k-1$ and $k_2 = n_f$) and $b \le 1-1/k$ (this happens for $k_1 = 1$ and $k_2 = k+1$). Therefore, $1+\max(a,b) \le 2-1/n_f$. Combined with Lemma 3.10, the theorem follows. □

3.4.3 Derandomization

Due to the fact that, the randomization procedure is only used to open replicas (we always connect a client to the nearest open replicas), the derandomization technique proposed by Jain *et al.* [56] can be applied here directly. We have the following result.

Lemma 3.15. *The bisection search based deterministic algorithm which employs Algorithm 3.4 as a subroutines is a* $(4-\frac{1}{n_f^2})$*-approximation algorithm for* FTKFA *and its time complexity is* $O(Rn^3\log(nL))$*, where* $L = c_{\max}/(Rf_{\min}+nc_{\min})$ *and* $n = \max(n_c, n_f)$.

3.5 Discussion

3.5.1 Dealing with Demand

As mentioned before, the *FTFA* problem with nonuniform demands (access frequencies) is equivalent to the *FTFA* problem with independent demands. Suppose the demand of client j is d_j, the problem becomes

$$\begin{array}{ll} \text{minimize} & \sum_{i\in\mathcal{F}} f_i y_i + \sum_{i\in\mathcal{F}}\sum_{j\in\mathcal{C}} c_{ij} d_j x_{ij} \\ \text{subject to} & \forall j\in\mathcal{C}: \sum_{i\in\mathcal{F}} x_{ij} \ge r_j \\ & \forall i\in\mathcal{F}, j\in\mathcal{C}: x_{ij}\le y_i \\ & \forall i\in\mathcal{F}, j\in\mathcal{C}: x_{ij}, y_i \in \mathbb{Z}^+ \end{array}$$

When d_j is an integer, cost $d_j \cdot c_{i,j}$ implies that there are d_j copies of client j at the same location. It is clear that the new problem can be transformed into the *FTFA* problem with each client being replicated d_j copies. When d_j is not an integer, we scale d_j and f_i together so that d_j becomes an integer. It is not hard to show that the new problem has the same solution as the original problem. By applying the approach as above, the problem can be transformed into an *FTFA* problem.

3.5.2 Fault-Tolerant Network Design

The *FTFA* problem is useful in the fault-tolerant network design. Suppose the downtime ratio is uniformly σ for each facility and the usability required by client j is μ_j (percent of time that a client is serviced). If the downtime of replicas (or links) is predicable (deterministic), for example, in a system where each facility needs a fraction of time to 'rest', the corresponding network design problem can be modeled as a *FTFA* problem with r_j set to $\lceil \mu_j/(1-\sigma)\rceil$. If the downtime is unpredictable (stochastic), then r_j should be set to $\lceil \log_\sigma(1-\mu_j)\rceil$. In both cases, the proposed algorithms are able to solve the problem. However, if replicas or links have nonuniform downtimes, the constraint on connectivity becomes $\sum_{i\in\mathcal{F}}(1-\sigma_{ij})x_{ij} \ge \mu_j$ for the deterministic model and $\prod_{i\in\mathcal{F}:x_{ij}=1}\sigma_{ij} \le 1-\mu_j$ for the stochastic model, where σ_{ij} is the downtime ratio of connection (i, j). For these cases our *FTFA* algorithms cannot be directly applied to solve the problem. We shall leave them as open problems for future study.

Chapter 4

QoS-Aware Content Replication for Parallel Access

After deploying replica servers, it comes to the problem of content management in these servers. Due to the reason that a content distribution network is shared by a number of content providers (as the clients of a CDN owner), it is impractical to store all the contents in each surrogate server. Therefore, in this chapter we study how to replicate contents in these servers in a cost-effective manner. We propose QoS-aware content replication technique for parallel access in which each client has a given degree of parallel connections determined by its QoS requirement (i. e. delay and priority). We study the optimization problem to maximize the combined download speed of all parallel connections at all clients and provide a distributed $(|\mathcal{R}|, 2)$-approximation algorithm which produces solutions comparable (within 4% error) to the optimal solutions in practice.

4.1 Introduction

As the Internet gradually becomes an essential infrastructure, a growing number of business and organizations rely on its operations and performance. Among all the functions of the Internet, efficient content delivery is most fundamental for the infrastructure. However, it is unlikely that the approach of ever increasing the number and capacity of facilities would improve the performance of network without a cost-effective solution for content management. Consequently, the replication and distribution of popular contents become critical for further performance improvement, especially in the practice to offload Web servers, improve end-user experience and increase system's reliability. Content replication [62, 122, 103] involves object replicas creation, deletion, and migration among hosts in response to changing usage patterns. Through replication in the Internet, contents draw closer to clients (end-users) and as a result the performance, especially the access latency perceived by a client, is much more improved. On the server side, replication shifts the load from congested servers (or links) to less loaded servers so that all servers are able to respond to requests quickly. At the same time, the service becomes more reliable and the system obtains improved fault-tolerant capability due to those replicas across the network. As a result, content replication has been used extensively with much success in the area of distributed computing [78].

With the popular contents replicated in multiple places over the network, parallel access to replicated contents [106, 92] which enables clients to fetch different portions of an object from multiple sources simultaneously and reassemble them locally, becomes a natural choice to improve access efficiency. By opening multiple connections to a chosen set of sources, it speeds up data transfer and reduces latency. At the same time, it also improves the resilience of the system inherently to route/link failures and traffic fluctuation. Parallel access to multiple servers usually involves mirror sites [109, 106] of a web site, surrogate servers in a Content Distribution Network, or peers in a P2P application [16, 58]. Byers *et al.* [17] proposed to access multiple servers in parallel using an open-loop multi-cast distribution where different servers generate different sets of parity blocks and cyclically transmit parities and originals. Rodriguez and Biersack [109] proposed dynamic parallel access to replicated contents using a parallel-access scheme which automatically shifts the load from congested locations to less loaded parts. In our earlier work [146], we studied QoS-oriented content delivery in e-learning systems through parallel access.

In this chapter, we study the object replication problem for QoS-aware object replication for parallel access in which each client has a given degree of parallel access determined by its QoS requirement or priority. The problem is formulated as a maximization problem of combined download speed of all parallel connections at all clients. Combined with a cost function which forms a metric when shortest-path routing is deployed, the problem is further converted into the metric *Fault Tolerant Facility Location* (*FTFL*) problem [55, 125] to minimize the total cost. Though there exist different algorithms for the *FTFL* problem including LP rounding algorithms [55, 125, 48, 47] and primal-dual algorithms [55, 125], these algorithms are all implemented in a centralized manner and not suitable to work in a distributed environment lacking global knowledge — because collecting global state information of a large network is impractical and the collected information may be outdated when it is ready for use. As such, a distributed *FTFL* algorithm is essential for content replication and it is also helpful in offloading the central server and reducing the traffic amount in the network.

Due to the NP-hardness of the *FTFL* problem, it is unlikely to find optimal solutions in polynomial time. Instead, researchers have developed approximation algorithms to solve the problem suboptimally. Mainly, there are two types of techniques used in approximation algorithm for *FTFL*, i. e. LP rounding and primal-dual schema. The drawback of LP rounding algorithms is that they need to solve large linear programs and so have prohibitive running times for most applications [56]. Therefore in this chapter, we propose an approximation algorithm for the *FTFL* problem using primal-dual schema. The algorithm is implemented in a distributed and asynchronous manner within $O(n)$ rounds of communication, where n is the number of surrogate servers in the network. As far as we know, the approximation factor of similar centralized algorithms (using primal-dual schema) remains unknown except a special case where all clients have a uniform degree of parallelism (i .e., $|\mathcal{R}| = 1$, where $\mathcal{R}$ is the set of parallel connection degrees). We prove that the cost of our solution is no more than $|\mathcal{R}| \cdot F^* + 2 \cdot C^*$ in the general case, where F^* and C^* are respectively the two components of cost regarding any optimal solution. Though this theoretical result is worse than the best-known 2.076-approximation algorithm which is obtained by a centralized LP rounding algorithm [125], extensive numerical experiments showed that the quality of our solutions is comparable (within 4% error) to optimal solutions in all cases we evaluated.

4.2 Problem Formulation

In a content delivery network composed of an origin server, a number of surrogate servers and clients (end-users or proxy servers), the origin server holds all objects that can be requested by clients and a surrogate server have the functionality of hosting part of these objects and serving a request if it holds the desired object. The origin server also maintains a table of replica locations in the network and redirects all the requests to the surrogate servers holding the required content which are most close to the client. Since only servers holding the desired object are capable of serving a request, we are interested in finding the locations (surrogate servers) to host the object so that certain objective is satisfied.

4.2.1 Object Placement for Parallel Access

In a parallel-access enabled network, the download speed of a client for an object is defined as the sum of the download speeds on all connections to multiple sources holding the object, and the throughput of a content delivery network is defined as the combined download speeds for all clients. Content replication and parallel access are two primary approaches to improve the throughput for a given network. Through object replication, an object is duplicated across the network to facilitate access at the cost of some additional cost, e. g. the expense to provide the storage space. It is clear that the throughput is maximized when the object is replicated at all surrogate servers but this obviously results in an impractically large storage cost. Therefore a well-chosen set of locations to hold the object is desired. We start this by calculating the gain for hosting an object in a given set of locations.

Consider graph $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of all nodes and $\mathcal{E}$ is the set of links between these nodes. Assume $\mathcal{V}$ comprises three disjoint sets $\{s\}$, $\mathcal{C}$ and $\mathcal{F}$, where s is the origin server, $\mathcal{C}$ is the set of clients and $\mathcal{F}$ is the set of surrogate servers. We want to find a subset $P \subseteq \mathcal{F}$ (called placement) to hold replicas so that certain property is obtained. Let $b_{ij}, i \in \mathcal{F} \cup \{s\}, j \in \mathcal{C}$ be the download speed of client j obtained from server i, and $P_j \subseteq P \cup \{s\}$ be the set of r_j distinct servers which are most close to j

among all servers holding the object. Then the download speed of j is $\sum_{i \in P_j} b_{ij}$ and the throughput of the network is $\sum_{j \in \mathcal{C}} \sum_{i \in P_j} b_{ij}$. We denote the normalized storage cost at server $i \in \mathcal{F}$ by f_i and there are different ways to calculate the storage cost. For example, in a memory-sensitive system, storage cost could be the loss of download speed for necessary object removal or the cost to deploy additional memory if possible. While in an update-frequent system, storage cost could be the cost used to maintain the freshness of replicas.

The objective of our problem is to maximize the gain by holding replicas of the object in set P, which is calculated by

$$\sum_{j \in \mathcal{C}} \sum_{i \in P_j} b_{ij} - \sum_{i \in P} f_i.$$

Due to the redirection of origin server, a request is served by the nearest sources from the client, i. e. $P_j \subseteq P \cup \{s\}$ to maximize the download speed of the client, i. e. $\max_{P_j \subseteq P \cup \{s\}} \sum_{i \in P_j} b_{ij}$. Let $\mathcal{R}$ represent all available degrees of parallel access and $r_j \in \mathcal{R}$ (i. e. the size of set P_j) reflects the degree of parallelism (or connectivity) for client j. Since the objective is to maximize the throughput of the network, the problem of object placement for parallel access can be modeled as the following *GMax* problem.

Definition 4.1. *Gain Maximization* (*GMax*) problem: In a graph $G(\mathcal{V}, \mathcal{E}), \mathcal{V} = \{s\} \cup \mathcal{C} \cup \mathcal{F}$, s is the origin server, $\mathcal{C}$ is the set of clients, and $\mathcal{F}$ is the set of surrogate servers. For any $j \in \mathcal{C}$, $r_j \in \mathcal{R}$ is the degree of parallel access for client j, $f_i, i \in \mathcal{F}$ is the cost of hosting the object at server i; $b_{ij}, i \in \mathcal{F} \cup \{s\}$ is the download speed between i and j. The *GMax* problem is to find a set of nodes, i. e. placement $P \subseteq \mathcal{F}$, to cache the object so that the gain of the placement is maximized. Mathematically, the problem can be formulated as

$$\begin{array}{ll} \max\limits_{P \subseteq \mathcal{F}} & \{\sum\limits_{j \in \mathcal{C}} \max\limits_{P_j \subseteq P \cup \{s\}} \sum\limits_{i \in P_j} b_{ij} - \sum\limits_{i \in P} f_i\} \\ s.t. & \qquad \forall j \in \mathcal{C} : \, |P_j| = r_j. \end{array}$$

Note the constraint reflects the degrees of parallel access, i. e. the number of connections required by each client.

4.2.2 An Alternative Integer Programming Formulation

Now, we provide an alternative formulation for the problem using integer programming. Let M be a large constant and define a cost function $c_{ij} = M - b_{ij}$ for any connection between $i \in \mathcal{F} \cup \{s\}$ and $j \in \mathcal{C}$. First, we define a *Cost Minimization* problem as follows.

Definition 4.2. The *Cost Minimization* (*CMin*) problem shares the same model as the *Gmax* problem except that the objective of the *CMin* problem is to minimize the total cost including connection cost and storage cost, i.e.,

$$\begin{aligned} &\min_{P \subseteq \mathcal{F}} \quad \{\sum_{j \in \mathcal{C}} \min_{P_j \subseteq P \cup \{s\}} \sum_{i \in P_j} c_{ij} + \sum_{i \in P} f_i\} \\ &s.t. \qquad \forall j \in \mathcal{C} : |P_j| = r_j, \end{aligned}$$

Clearly the *GMax* problem is equivalent to the *CMin* problem because

$$\begin{aligned} &\sum_{j \in \mathcal{C}} \max_{P_j} \sum_{i \in P_j} b_{ij} - \sum_{i \in P} f_i \\ &= \sum_{j \in \mathcal{C}} r_j \cdot M - \{\sum_{j \in \mathcal{C}} \min_{P_j} \sum_{i \in P_j} c_{ij} + \sum_{i \in P} f_i\}, \end{aligned}$$

where the first item is a constant. It is not hard to see that the *Cmin* problem can be further converted into the *Fault Tolerant Facility Location* problem.

Facility location problem [94] has been studied extensively in operations research. In the *Uncapacitated Facility Location (UFL)* problem, we are given a set of facilities (servers) $\mathcal{F}$ and a set of clients $\mathcal{C}$. For every facility $i \in \mathcal{F}$, a non-negative number f_i is given as the opening cost of facility i; and for every facility-client pair (i, j) a connection cost c_{ij} between facility i and client $j \in \mathcal{C}$. The objective of the problem is to open a subset of the facilities in $\mathcal{F}$, and connect each client to an open facility so that the total cost is minimized. In *UFL*, only one connection is required for each client which is not the case in many application scenarios and therefore a more general problem called *Fault-Tolerant Facility Location (FTFL)* was proposed [55]. In *FTFL*, each client $j \in \mathcal{C}$

has a prespecified connectivity r_j and has to be assigned to r_j distinct facilities instead of just one to achieve fault-tolerance capability in case of connection or facility fails.

Definition 4.3. [55] The *FTFL* problem can be formulated by the following ILP (integer linear program).

$$\begin{array}{ll} \text{minimize} & \sum_{i \in \mathcal{F}} f_i y_i + \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{C}} c_{ij} x_{ij} \\ \text{subject to} & \forall j \in \mathcal{C} : \sum_{i \in \mathcal{F}} x_{ij} \geq r_j \\ & \forall i \in \mathcal{F}, j \in \mathcal{C} : y_i \geq x_{ij} \\ & \forall i \in \mathcal{F}, j \in \mathcal{C} : x_{ij}, y_i \in \{0, 1\} \end{array} \tag{4.1}$$

In the above formulation, binary y_i indicates whether facility i is opened, and x_{ij} whether i and j are connected. The objective of the problem is to open proper number facilities among $\mathcal{F}$ and assign each client j to r_j distinct facilities so as to minimize the total cost for facility opening and connections establishing. Note that the first constraint ensures the number of connections required by each client and the second constraint reflects that a client can only be assigned to an open facility. Let $\mathcal{R}$ be all possible values of r_j for all $j \in \mathcal{C}$, the problem becomes the classical *UFL* problem when $\mathcal{R} = \{1\}$. The *FTFL* problem can further be combined with demands by replacing c_{ij} with $d_j c_{ij}$, where d_j is the demand of client j. Note that the requirement of multiple distinct connections for a client violates with the cost minimization objective and therefore need to be treated carefully, since a minimized cost for multiple connections requires to establish all the connections to the same facility closest to the client. We have the following lemma.

Lemma 4.1. *The CMin problem is equivalent to the FTFL problem.*

Considering the NP-hardness of the facility location problem, the following theorem is immediate.

Theorem 4.1. *Both the GMax and CMin problems are NP-hard.*

4.2.3 Issues in the FTFL Problem

4.2.3.1 Triangle inequality

For the facility location problems (*UFL* and *FTFL*), there exist several constant-approximation algorithms [124, 56, 52, 55, 125, 48, 47] when connection cost forms a metric (i.e. follows the triangle inequality). However, if the connection costs are unrestricted, the hardness of the problem is the same as the set cover problem and therefore it cannot be solved better than $O(\log n)$-approximation in polynomial time. Fortunately, the cost function in the problem of object placement has certain property due to the deployment of shortest-path routing. We have the following lemma.

Lemma 4.2. *Cost function **c** forms a metric when the shortest-path routing is deployed.*

Proof. Let Δt be the delay in the LAN centered at a representative client and assume Δt is a constant. Let t_{ij} be the time used to transmit the whole object (with certain size) between i and j. We can calculate the download speed between i and j , i.e. b_{ij} by $size/(\Delta t + t_{ij})$. Let $M = size/\Delta t$, then

$$c_{ij} = \frac{size}{\Delta t} - \frac{size}{\Delta t + t_{ij}} = \frac{size \cdot t_{ij}}{\Delta t \cdot (\Delta t + t_{ij})} = \frac{M \cdot t_{ij}}{\Delta t + t_{ij}}.$$

Clearly, when the shortest-path routing is deployed, ***t*** forms a metric and in this case the cost function ***c*** also forms a metric because M and Δt are both constants. □

4.2.3.2 Access frequency

When access frequencies are considered, it is not hard to see that the *CMin* problem is equivalent to the *FTFL* problem with demands. Both enhanced problems are able to be solved using the approaches for the original problems. Suppose the access frequency (or demand) of $j \in \mathcal{C}$ is d_j. When d_j is an integer, cost $d_j \cdot c_{ij}$ implies that there are d_j copies of client j at the same location. It is clear that the new problem can be transformed into the *FTFL* problem with each client being replicated d_j copies. When d_j is not an integer, we scale d_j and f_i together until d_j becomes an integer. It is clear that the new problem has the same solution as the original problem. By applying the approach as above, the problem can be transformed into an *FTFL* problem.

4.3 A Distributed ($|\mathcal{R}|$,*2*)-Approximation Algorithm

Similar to the algorithms in [52], our algorithm also uses a notation of star which is composed of a facility and a group of clients that connected with the facility. Given two sets $\mathcal{F}$ and $\mathcal{C}$ as well as f_i, r_j and c_{ij} for any $i \in \mathcal{F}$ and $j \in \mathcal{C}$, the algorithm selects the most cost-efficient star repeatedly and updates related variants until all the clients are *fully-connected* (i. e., there are r_j connections for each $j \in \mathcal{C}$). Specifically, let U be the set of *not-fully-connected* clients and $\mathcal{C} \setminus U$ the set of *fully-connected* clients, and the *cost efficiency* eff(i, C) of a star $(i, C), C \subseteq U$ is defined to be

$$\frac{f_i + \sum_{j \in C} c_{ij} - \sum_{j \in \mathcal{C} \setminus U} \max(c_{i'j} - c_{ij}, 0)}{|C|}, \tag{4.2}$$

where $c_{i'j}$ is the maximum connection cost of client $j \in \mathcal{C} \setminus U$ at the moment before the star (i, C) is selected. The first two items in the numerator represent the total cost of the star and the last item is the contribution made by *fully-connected* clients via connection exchange. Each facility finds the most *cost efficient* star, that is eff$(i, C') = \min_{C \subseteq U}$ eff(i, C) in each iteration.

4.3.1 Facility/Client Side Pseudocodes

In the distributed settings, each facility has a list of clients which is not yet *fully-connected* and each client has a list of facilities which have the potential to be connected with. Different from centralized settings, a facility $i \in \mathcal{F}$ only knows the value of f_i and $c_{ij}, j \in \mathcal{C}$ without the knowledge of $c_{i'j}, i' \neq i$. Assume c_{ij} is a distance function for facility-client pair (i, j) that forms a metric. Actually, this assumption is reasonable as in most cases that shortest-path routing is deployed. In our algorithm, a client does not have to know the information on connection cost. Our facility-side and client-side algorithms are given in Algorithm 4.1 and Algorithm 4.2 respectively. Actually, these algorithms can be seen as a fault-tolerant extension to Frank and Romer's algorithms for *UFL* [44]. Compared with Frank and Romer's algorithms for *UFL* [44], our algorithms have the feature of asynchronous communication which is implemented using a round number to ensure the coordination between clients and facilities. This is very important in an environment composed of heterogeneous networks to achieve performance acceleration.

Algorithm 4.1 Facility-Side Pseudocode

```
01: set $U \leftarrow \mathcal{C}$ and round number $q \leftarrow 1$.
02: while $U \neq \phi$ do
03:     find the most cost-efficiency star $(i, C'), C' \subseteq U$ .
04:     send message $(q, i, \text{eff})$ to all $j \in C'$ .
05:     receive requests from all clients in $C'$, suppose the total number is $req$.
06:     if $req = |C'|$ then
07:         open the facility $i$, if it is not already open, set $f_i \leftarrow 0$ and $y_i \leftarrow 1$.
08:         send signal "open" to all $j \in C'$, set $x_{ij} \leftarrow 1$ for all $j \in C'$.
09:     else
10:         send signal "not-open" to all $j \in C'$.
11:     endif
12:     receive signal "fully-connected" from all clients in $U$ , let $C$" be the set of those
        clients.
13:     set $U \leftarrow U \setminus C$" and $q \leftarrow q + 1$.
14: enddo
```

Algorithm 4.2 Client-Side Pseudocode

```
01: set connection number $p \leftarrow 1$ and round number $q \leftarrow 1$.
02: while $p \leq r_j$ do
03:     receive message $(q, i, \text{eff})$ from all $i \in \mathcal{F}$.
04:     set $i^* \leftarrow \text{argmin}_{i \in \mathcal{F}} \text{eff}_{q,i}$ and $q \leftarrow q + 1$, if two stars have the same cost efficiency,
        break ties according to facility IDs.
05:     send request to $i^*$ to be connected and signal "not-selected" to others.
06:     receive signal from $i^*$.
07:     if the signal is "open" then
08:         set $p \leftarrow p + 1$.
09:         if $p = r_j$ then
10:             send signal "fully-connected " to all facilities.
11:         endif
12:     endif
13: enddo
```

In the algorithm, each facility maintains a list of clients which are not *fully-connected* and the facility starts a conversation by sending the cost efficiency value of a regarding star centered at itself to all the members clients. Each client compares the values of efficiency sent from all facilities and requires to be connected with the facility with the minimum cost efficiency. If a facility receives connection request from all of its members, the facility is opened with a corresponding signal sent to the member clients, otherwise the facility sends signal "*not-open*" to its member clients. Once a client receives a signal "*open*" from a facility which is requested to be connected by itself, they are connected and, if this happens to be the r_j-*th* connection, the client also sends signal "*fully-connected*" to all facilities. When a facility receives a "*fully-connected*" signal from a client, it excludes the client from its list. A facility quits when its list is empty and the algorithms ends when all facilities have quitted.

Consider the same worst case as in [44] in which a chain of facilities interconnected by at least one client. In this case, only the facility at one end can be opened in one round if the cost efficiencies are monotonously increasing along the chain. It is clear that the number of rounds required is at most n, where n is the number of surrogate servers, i e. $|\mathcal{F}|$. So we have the following lemma whose proof is similar to [44].

Lemma 4.3. *The proposed distributed algorithm can be completed in $O(n)$ rounds of communication with each message no more than $O(\log n)$ bits, where n is the number of surrogate servers.*

4.3.2 An alternative centralized algorithm

In order to reveal the approximation factor of the distributed algorithm, we extend a centralized algorithm which was first proposed by Jain *et al.* [52] and further studied by Swamy and Shmoys [125]. We will show later that the distributed algorithm and the centralized algorithm share the same approximation factor.

In the above algorithm, all clients have certain *credits* to offer which are equal to zero at the beginning and increase simultaneously in time until their respective clients get *fully-connected*. Once a client is *fully-connected* it is removed from set U and we suppose the maximum connection cost of client j is $c_{i'j}$. All clients offer their contributions to open facilities, a client in U with amount $\max(t-c_{ij}, 0)$ and a client in $\mathcal{C}\backslash U$ with

Algorithm 4.3 Extended JMS Algorithm [47, 52]

```
Given: $\mathcal{F}, \mathcal{C}$ and $f_i, c_{ij}, r_j$ for any $i \in \mathcal{F}, j \in \mathcal{C}$
Output: $x_{ij}, y_i, \alpha_j^p$ for any $i \in \mathcal{F}, j \in \mathcal{C}, 1 \leq p \leq r_j$
(1) At the beginning ($t \leftarrow 0$), all facilities are *unopened* ($y_i \leftarrow 0$), all clients are
    *unconnected* ($U \leftarrow \mathcal{C}, x_{ij} \leftarrow 0$) and their credit is equal to zero but will increase
    simultaneously with time. At every moment, each client ($j \in U$) offers some money
    from its credit to *unopened* facilities, a not-fully-connected with amount $\max(t -
    c_{ij}, 0)$; and a fully-connected client ($j \in \mathcal{C} \setminus U$) with amount $\max(c_{i'j} - c_{ij}, 0)$,
    where $c_{i'j}$ is the maximum connection cost of $j$ at the moment.
(2) While $U \neq \phi$, increase the credit of each client $j \in U$ at the same rate until *Event
    (a)* or *Event (b)* occurs. If two events occur at the same time, we break ties according
    to facility IDs.
    (a) *Event (a)*: Some unopened facility $i$ receive enough money to open itself, that
        is $\sum_{j \in U} \max(t - c_{ij}, 0) + \sum_{j \in \mathcal{C} \setminus U} \max(c_{i'j} - c_{ij}, 0) = f_i$. In this case, set
        $y_i \leftarrow 1, l_i \leftarrow 0$, and let $C'_1 = \{j \in U : t \geq c_{ij}\}$, set $x_{ij} \leftarrow 1$ for all $j \in C'_1$; and
        $C'_2 = \{j \in \mathcal{C} \setminus U : c_{i'j} > c_{ij} 0\}$, set $x_{i'j} \leftarrow 0, x_{ij} \leftarrow 1$ for all $j \in C'_2$.
    (b) *Event (b)*: For some client $j \in U$, its credit is enough to connect an open facility
        $i$ which is not connected before, i. e. $t = c_{ij}$. In this case, set $x_{ij} \leftarrow 1$.
    (c) For any new established connection of client $j$, set $\alpha_j^{p_j} \leftarrow t$ and $p_j \leftarrow p_j + 1$.
        If a client is fully connected, i. e. $p_j = r_j$, delete client $j$ from $U$.
```

amount $\max(c_{i'j} - c_{ij}, 0)$. The algorithm opens the most *cost-efficient* star repeatedly and updates connections correspondingly. In this process, the maximum connection cost of a fully connected client is to be exchanged with a cheaper connection. Note that the amount of credit that client j owns is equal to t before the client is *fully-connected*. Using the definition on cost-efficiency, i. e. Formula (4.2), the most cost-efficient star is the first star that achieves $\sum_{j \in U} \max(t - c_{ij}, 0) + \sum_{j \in \mathcal{C} \setminus U} \max(c_{i'j} - c_{ij}, 0) = f_i$ if facility i is not opened. If a facility i is already open, the facility cost f_i is set to zero and thus the above cost-efficiency formula suggests that i can be connected with any client which is not *fully-connected* only if $t = c_{ij}$.

It is not hard to see that if we order the stars opened in the distributed algorithm according to their cost efficiencies, it would be exactly the same to those opened in the centralized algorithm. As such we have the following lemma and its proof is similar to that in [44].

Lemma 4.4. *As far as the derived solutions are concerned, the distributed algorithm is equivalent to the centralized algorithm.*

As far as we know, approximation factor of Algorithm 4.3 remains unknown for the general case. We prove in the next subsection that the solution of Algorithm 3 is no more than $|\mathcal{R}| \cdot F^* + 2 \cdot C^*$, where F^* and C^* are respectively the facility cost and connection cost in any optimal solution. For example, the algorithm is at least 2-approximation when $|\mathcal{R}| = 2$, and $(1, 2)$-approximation when $|\mathcal{R}| = 1$, i. e. the special case of uniform connectivity. All these are competitive to the existing results.

4.3.3 An upper bound of performance ratio

In Algorithm 4.3, the amount of credit paid for a connection can be divided further — part for a connection with a smaller cost, remaining for opening other replicas. Despite this, it is true that all payments of a client are either used to open replicas or to establish connections, therefore the total cost of the solution is still $\sum_{j \in \mathcal{C}} \sum_{p=1}^{r_j} \alpha_j^p$. This results in the following lemma.

Lemma 4.5. *A solution produced by Algorithm 4.3 is feasible to the FTFL problem and its total cost is equal to* $\sum_{j \in \mathcal{C}} \sum_{p=1}^{r_j} \alpha_j^p$.

We need to capture other properties of the algorithm to provide an upper bound of the cost. We denote the optimal solution by a set of stars, i. e. $\mathcal{S}^*$, and for any star $s \in \mathcal{S}^*$ centered at facility i and any client j in the star, we define $\alpha_j^s = \alpha_j^{p(i,j)}$ and

$$p(i,j) = \begin{cases} p_j & \text{if } x_{ij} = 1, \\ r_j & \text{otherwise,} \end{cases}$$

where p_j is the rank of the contribution made to facility i among all contributions made by the client (or the port number via which connection (i, j) is set up). We can see that $\alpha_j^{p(i,j)+1} \geq \alpha_j^{p(i,j)}$ if $p(i, j) + 1 \leq r_j$ because time t is increasing, therefore the sum of α_j^s is an upper bound on the total cost.

Lemma 4.6. *The cost of a solution derived by Algorithm 4.3 is no more than* $\sum_{s \in \mathcal{S}^*} \sum_{j \in s \cap \mathcal{C}} \alpha_j^s$.

Consider a star s which is centered at facility i, we omit s and i for the sake of simplicity if no confusion is caused. Assume without loss of generality that the star is composed of k clients numbered from 1 through k and $\alpha_1 \leq \alpha_2 \leq \ldots \leq \alpha_k$. For any clients j and h, $1 \leq j < h \leq k$, we define u_{jh} as

$$u_{jh} = \begin{cases} \alpha_j & \text{if } \alpha_j = \alpha_h, \\ c_{i'j} & \text{otherwise,} \end{cases} \tag{4.3}$$

where $c_{i'j}$ is the maximum connection cost of j before time t. Now consider time $t = \alpha_h - \epsilon, \epsilon \rightarrow 0$. At this time, the amount of money that client j offers to open a facility is equal to

$$\begin{aligned} &\max(u_{jh} - c_{ij}, 0) && \text{if } j < h, \text{and} \\ &\max(\alpha_h - c_{ij}, 0) && \text{otherwise.} \end{aligned}$$

Note that by the definition of u_{jh} this holds even if $\alpha_j = \alpha_h$. It is clear that the total offer of clients to a facility can never become larger than the opening cost of the facility. Therefore, we have the following lemma.

Lemma 4.7. *In an optimal star s, $\sum_{j=1}^{h-1} \max(u_{jh} - c_{ij}, 0) + \sum_{j=h}^{k} \max(\alpha_h - c_{ij}, 0) \leq f_i$ for any client h in the star.*

Similar to the lemmas regarding triangle inequality in the last chapter, we have the following lemma.

Lemma 4.8. *In an optimal star s, $\alpha_h \leq u_{jh} + c_{ij} + c_{ih}$ for any two clients h and j, if $p(i,j) \leq p(i,h)$ and $j < h$.*

Proof. If $\alpha_h = \alpha_j$, the lemma holds obviously according to the definition; otherwise $\alpha_h > \alpha_j$. Let F_1 be the set of open facilities connected with client j before time $t = \alpha_h$, then $|F_1| \geq p(i,j)$ because $\alpha_h > \alpha_j$ and the number of facilities connected with a client is monotonically increasing. Now, consider two cases:

1) $i \notin F_1$. In this case, contribution α_h could be made to at least one facility in F_1, suppose i'. That is, there exists $i' \in F_1, (i' \neq i)$ such that $\alpha_h \leq c_{i'h}$. Further

combine the triangle inequality $c_{i'h} \leq c_{i'j} + c_{ij} + c_{ih}$ and $u_{jh} = \max_{i' \in F_1} c_{i'j}$, we have $\alpha_h \leq u_{jh} + c_{ij} + c_{ih}$.

2) $i \in F_1$, i. e. i is opened before time t. It is clear that h could be connected with i at least at the time when it made contribution α_h, we have $\alpha_h \leq c_{ih}$. □

Let F^* and C^* be the total facility cost and connection cost respectively in the optimal solution and SOL the total cost of the derived solution, we have $F^* = \sum_{s \in \mathcal{S}^*} f_i$, $C^* = \sum_{s \in \mathcal{S}^*} \sum_{j \in s \cap \mathcal{C}} c_{ij}$, and $SOL \leq \sum_{s \in \mathcal{S}^*} \sum_{j \in s \cap \mathcal{C}} \alpha_j^s$. Now if there is constant λ_f and λ_c (to be fixed later) such that

$$\max_{s \in \mathcal{S}^*} \frac{\sum_{j \in s \cap \mathcal{C}} \alpha_j^s - \lambda_f \cdot f_i}{\sum_{j \in s \cap \mathcal{C}} c_{ij}} \leq \lambda_c,$$

then it is clear that $SOL \leq \lambda_f \cdot F^* + \lambda_c \cdot C^*$.

According to Lemma 4.8, we know that all contributions with the same rank i. e. $p(i,j)$, follow triangle inequality, so we decompose each optimal star into a set of substars $\{s_p, 1 \leq p \leq |\mathcal{R}|\}$ and each substar has a number of clients with same value of $p(i,j)$. We duplicate the facility cost into each substar and set $\lambda_f = |\mathcal{R}|$. For each substar, define $k = |s_p|$ and $\lambda = \sup_{k \in Z^+} \lambda_k$, where λ_k is the maximum of the following LP.

$$\begin{array}{ll}
\text{maximize} & \frac{\sum_{j=1}^{k} \alpha_j - f}{\sum_{j=1}^{k} c_j} \\
\text{subject to} & \forall 1 \leq j < h \leq k: \ \alpha_h \leq u_{jh} + c_j + c_h \\
& \forall 1 \leq h \leq k: \ \sum_{j=1}^{h-1} \max(u_{jh} - c_j, 0) + \\
& \sum_{j=h}^{k} \max(\alpha_h - c_j, 0) \leq f \\
& \forall 1 \leq j < h \leq k: \ \alpha_j, c_j, u_{jh}, f \geq 0.
\end{array} \tag{4.4}$$

Lemma 4.9. *For any integer k, $\lambda_k \leq 2$ and therefore the solution of Algorithm 4.3 is no more than $|\mathcal{R}| \cdot F^* + 2 \cdot C^*$*, where F^* and C^* are respectively the facility cost and connection cost in an optimal solution.

Proof. Adding up the two constraints of program (4.4), we get

$$k \cdot \alpha_h \leq f + \sum_{j=1}^{k} c_j + \sum_{j=1}^{h-1} (c_j + c_h)$$

Adding up both sides of the above inequalities for all $1 \leq h \leq k$, we have

$$k \cdot \sum_{h=1}^{k} \alpha_h = k \cdot (f + \sum_{j=1}^{k} c_j) + (k-1) \cdot \sum_{j=1}^{k} c_j,$$

that is

$$\frac{\sum_{h=1}^{k} \alpha_h - f}{\sum_{j=1}^{k} c_j} \leq \frac{2k-1}{k} \leq 2$$

for all possible values of α_j, c_j and f and sizes of sub-star $s_p, 1 \leq p \leq |\mathcal{R}|$. It is clear that $\lambda_k \leq 2$ for any integer k.

Now set $c_j = c_{ij}, f = f_i$ and $\alpha_j = \alpha_j^s$ and $k = |s_p|$. Since all clients in a sub-star have the same value of $p(i,j)$, Lemma 4.7 and Lemma 4.8 respectively imply the two constraints of program (4.4). Because the facility cost is duplicated at most $|\mathcal{R}|$ times (each in a sub-star), we have

$$\max_{s \in \mathcal{S}^*} \frac{\sum_{j \in s \cap \mathcal{C}} \alpha_j^s - |\mathcal{R}| \cdot f_i}{\sum_{j \in s \cap \mathcal{C}} c_{ij}} \leq 2. \tag{4.5}$$

The theorem follows after adding up the above inequalities in all stars in the optimal solution. □

We say a solution $SOL \leq |\mathcal{R}| \cdot F^* + 2 \cdot C^*$, where F^* and C^* are respectively the facility cost and connection cost in an optimal solution, is a $(|\mathcal{R}|, 2)$-approximation solution to the *FTFL* problem. As a result, Algorithm 4.3 is $(|\mathcal{R}|, 2)$-approximation at most and we do not know the exact approximation factor of the algorithm. Combine Lemma 4.3, Lemma 4.4and Theorem 4.9, we have the following theorem.

Theorem 4.2. *The distributed algorithm is at most* $(|\mathcal{R}|, 2)$*-approximation to the GMax problem and the algorithm can be finished within* $O(n)$ *rounds of communication with each message no more than* $O(\log n)$ *bits.*

4.4 Numerical Results

Though our algorithm has a larger approximation factor than the best-known 2.076-approximation [125] which was obtained in a centralized manner using LP rounding technique, the proposed algorithm performs very well in the average. In fact, our numerical experiments show that the derived solutions are comparable to optimal solutions in all cases we evaluated.

We have implemented the proposed distributed algorithm and run the algorithm in three groups of experiments. In all cases, the solution of the proposed algorithm is compared with optimal solution of the LP-relaxation of the problem which is computed using the commercial software package CPLEX in order to get the performance ratios.

4.4.1 Results on Grids

Due to the fact that our algorithm could be regarded as a distributed version of the extended JMS algorithm which was originally proposed to solve the *UFL* problem, we would like to compare our results (for the *FTFL* problem) with the results of JMS algorithm for the *UFL* problem. (The performance ratios for the *UFL* problem are cited from [52] directly which are close to the results using our implementation of the JMS algorithm). In this group of experiments, facilities numbers and client numbers are chosen to be the same as those in [52].

4.4.1.1 Impact of connectivity

First we set the facility cost $f_i, i \in \mathcal{F}$ to be a random integer between 0 and 30000, that is $f_i = \text{rand}(0, 30000)$. The connectivity is set to be a random integer between 1 and $r_{\max}$, that is $r_j = \text{rand}(1, r_{\max})$, where $r_{\max}$ ranges from one fourth of the total facility number to 100% of the total facility number. The performance ratios are listed in Table 4.1 together with the performance ratios for the *UFL* problem as cited from [52].

Table 4.1: Performance Ratios Using Random Points on a Grid (with various maximum connectivity)

$\|\mathcal{C}\|$	$\|\mathcal{F}\|$	*UFL* see [52]	*FTFL*			
			$r_{\max} = 25\% \cdot \|\mathcal{F}\|$	$50\% \cdot \|\mathcal{F}\|$	$75\% \cdot \|\mathcal{F}\|$	$100\% \cdot \|\mathcal{F}\|$
50	20	1.0041	1.0074	1.0241	1.0142	1.0044
100	20	1.0019	1.0082	1.0079	1.0032	1.0013
100	50	1.0002	1.0326	1.0238	1.0057	1.0029
200	50	1.0035	1.0161	1.0059	1.0018	1.0008
200	100	1.0060	1.0373	1.0061	1.0017	1.0009
300	50	1.0054	1.0138	1.0018	1.0009	1.0003
300	80	1.0053	1.0212	1.0034	1.0010	1.0004
300	100	1.0042	1.0163	1.0023	1.0010	1.0005
300	150	1.0019	1.0222	1.0035	1.0013	1.0005
400	50	1.0035	1.0063	1.0009	1.0004	1.0001
400	100	1.0026	1.0106	1.0014	1.0005	1.0003
400	150	1.0013	1.0142	1.0017	1.0007	1.0003

Table 4.2: Performance Ratios Using Random Points on a Grid (with various maximum facility cost)

$\|\mathcal{C}\|$	$\|\mathcal{F}\|$	*UFL* see [52]	*FTFL*			
			$f_{\max} = 10000$	30000	50000	70000	90000
50	20	1.0041	1.0058	1.0150	1.0175	1.0132	1.0116
100	20	1.0019	1.0035	1.0132	1.0142	1.0178	1.0126
100	50	1.0002	1.0095	1.0106	1.0161	1.0069	1.0122
200	50	1.0035	1.0084	1.0111	1.0098	1.0144	1.0154
200	100	1.0060	1.0101	1.0109	1.0121	1.0119	1.0158
300	50	1.0054	1.0057	1.0128	1.0103	1.0128	1.0137
300	80	1.0053	1.0086	1.0096	1.0141	1.0128	1.0121
300	100	1.0042	1.0096	1.0104	1.0095	1.0126	1.0117
300	150	1.0019	1.0108	1.0122	1.0116	1.0104	1.0112
400	50	1.0035	1.0039	1.0094	1.0136	1.0114	1.0111
400	100	1.0026	1.0084	1.0123	1.0120	1.0107	1.0140
400	150	1.0013	1.0108	1.0101	1.0125	1.0113	1.0114

4.4.1.2 Impact of facility costs

In the consequent experiments, we set $r_j = \text{rand}(1, 10)$ and $f_i = \text{rand}(0, f_{\max})$, where $f_{\max}$ ranges from 10000 to 90000. Note that the facility cost for the *UFL* problem in [52] is set to be a random integer between 0 and 9999. The performance ratios are listed in Table 4.2.

We notice that our solution should be optimal (i. e., approximation ratio should be 1) when $|\mathcal{R}| = |\mathcal{F}|$ in which case all facilities should be opened. The reason that our result is greater than 1 when $r_{\max} = 100\% \cdot |\mathcal{F}|$ is that not all integers from 1 to $r_{\max}$ are enumerated by the random number generator in some runs, due to the small size of example set (i. e. set $\mathcal{C}$).

As shown in both tables, the proposed algorithm behaves extremely well and its performance ratios to the *FTFL* problem are in the same order of magnitude as the performance ratios of JMS algorithm to the *UFL* problem. In all the cases we evaluated, the maximum performance ratio is 1.0373 which happens in the *5-th* group of experiments as shown in Table 4.1. Therefore we are able to conclude that the quality of our solutions is comparable (within 4% error) to that of optimal solutions in all evaluated cases.

4.4.2 Results on Network Models

In this group of experiments, we generate instances of the problem based on network models. We first use the most simple model — random graph and then GT-ITM model to simulate a network. The instance sizes, in terms of client number and facility number, vary from 100 to 416 and 20 to 100 respectively which are the same to those instances in [52].

4.4.2.1 Random graph

We have generated random graphs according to the distribution $G(n, p)$ and assigned uniform random weights on the edges. Clients and facilities correspond to the nodes in this graph, and the connection cost between a client and a facility is defined to be the shortest path between the corresponding nodes. The facility opening costs are generated at random. For each size, 15 instances are generated and the average error of the algorithm (compared to the lower bound obtained from the LP relaxation of the problem) is computed. The results of these experiments are shown in Table 4.3.

4.4.2.2 GT-ITM model

GT-ITM [148, 19] is a software package to generate network topologies. In this model, we consider transit node as potential facilities and stub nodes as clients. The connection cost is set to be the shortest distance and facility opening cost a random number. Connectivity of a client is set to be a random number between 1 and 10. This model is also used in the applications of facility location problems such as placing web server replicas [98]. Again, 15 instances are generated for each size and the average error of the algorithm compared to the lower bound obtained from the LP relaxation of the problem is computed. The results are also shown in Table 4.3.

Table 4.3: Performance Ratios on Network Models

$\|\mathcal{C}\|$	$\|\mathcal{F}\|$	Random Graph	GT-ITM Model
100	20	1.009	1.0032
160	20	1.0072	1.0012
160	40	1.0091	1.0029
208	52	1.009	1.0049
240	60	1.0082	1.003
300	75	1.007	1.003
312	52	1.0091	1.002
320	32	1.0072	1.002
400	100	1.0085	1.003
416	52	1.0091	1.003

4.4.3 Factors of Performance Ratio

The third group of experiments is to reveal the factors of performance ratio, where all instances are generated randomly on a 1000*1000 grid: In each instance, clients and facilities are points selected randomly from the grid and the connection cost is set to be the Euclidean distance of the corresponding points. There are totally 100 facilities and 200 clients in this group of experiments.

First we want to show the relation between performance ratio and size of $|\mathcal{R}|$. For the sake of simplicity, we set facility cost a random between 1 to 30000 and the connectivity for a client, as a random integer between 1 to $r_{\max}$. The result is pictured in

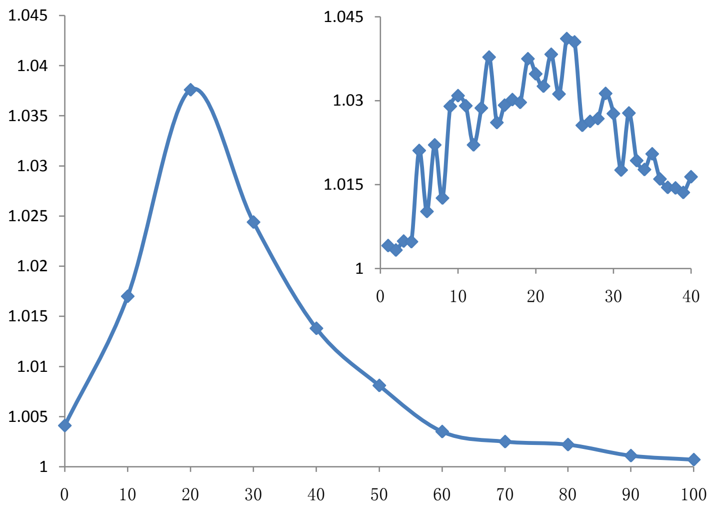
Figure 4.1(a), where the horizontal axis presents the value of $r_{\max}$ ranging from 1 to 100. The figure shows that the performance ratio increase together with the maximum connectivity before it reaches 20, and then decrease all the way until it reaches 100.

In order to show the impact of facility costs, we set the maximum connectivity $r_{\max} = 5$ and run the same experiments a number of times with facility costs set as a random between $\frac{1}{2}f_{\max}$ and $f_{\max}$. Figure 4.1(b) shows the performance ratios with the maximum facility cost $f_{\max}$ ranges from 100 to 52428800. The result reflects that the performance ratio reaches its maximum when the maximum facility cost is around 200000 and approaches one either when the facility cost becomes larger or becomes smaller.

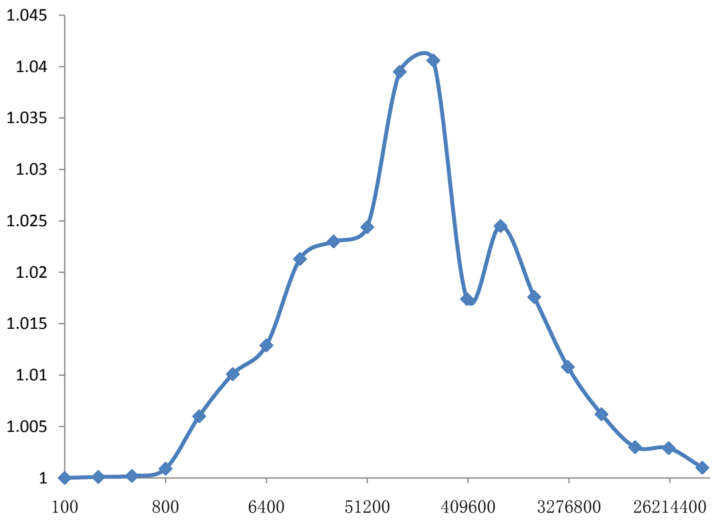
4.5 Related Work

The facility location problem and its variants occupy a central place in operations research [94]. For the simplest *UFL* problem, the first approximation algorithm was built by Cornuejols *et al.* [34]. They obtained $(1 - e^{-1})$-approximation algorithm for the maximization variant of *UFL*. The first approximation algorithm for the minimization variant, is a greedy algorithm achieving a guarantee of $O(\log n)$ in the general (non-metric) case due to [49]. Since these works dated back to almost 20 years ago, *UFL* has found extensive application and has been studied widely. Existing algorithms for *UFL* mainly use LP rounding technique or primal-dual schema.

Fault Tolerant Facility Location [55] is a generalization of *UFL*, where connectivity at different clients (i. e. the number of distinct facilities that serve a client) are specified to meet fault-tolerant requirements. The *FTFL* problem has been studied extensively in the recent years [55, 47, 48, 125]. Guha *et al.* obtained a 3.16-approximation algorithm by rounding the optimal fractional solution to the problem and further improved the result to 2.41 by employing a greedy local improvement step [48]. Recently, Swamy and Shmoys presented a 2.076-approximation by using LP rounding [125]. All these results hold for both uniform connectivity case and non-uniform connectivity case (general



(a) Impact of The Maximum Connectivity



(b) Impact of The Maximum Facility Cost

Figure 4.1: Factors of Performance Ratio

case). Guha and Khuller proved that the best approximation factor (lower bound) to *UFL* is 1.463 [112], assuming NP $\not\subseteq$ DTIME$[n^{O(\log\log n)}]$. This result also holds for fault-tolerant version of the problem.

Researchers have attempted to devise distributed algorithms for the *UFL* problem: Moscibroda and Wattenhofer [85] presented a distributed algorithm for the standard non-metric facility location problem in the $\mathcal{CONGEST}$ model that, for every k, achieves an $O(\sqrt{k}(n\rho)^{1/\sqrt{k}}\log(n))$-approximation in $O(k)$ communication rounds, where n is the total number of facilities and clients, and ρ is a coefficient that depends on the opening costs and connection costs as part of the input. Frank and Römer [44] considered the metric facility location on multi-hop networks, using the 1.61-approximation due to Jain *et al.* [52], they showed how to implement the algorithm in a distributed setting without any degradation in the approximation factor. Gehweiler *et al.* [46] presented a constant-approximation, constant-round distributed algorithm using only $O(\log n)$-bits per message, for the uniform facility location problem, where all the opening costs are identical and the underlying network is a clique. As to the fault-tolerant version of the facility location problem (i. e. the *FTFL* problem) in the general case of non-uniform connectivity, the approximation factor of existing centralized algorithms using primal-dual schema is unknown and no distributed algorithm is known either as far as we know. However, we are able to show in this chapter how to applying the primal-dual schema to tackle the *FTFL* problem. Particularly, we extend the JMS algorithm [44] by charging each connection of a client so that the total cost is paid collaboratively by all connections at all clients. We provide approximation analysis for the new algorithm in the general case wherein all clients have non-uniform connectivity and further present a distributed implementation of the algorithm.

4.6 Conclusion and Discussion

The essential of fault-tolerant facility location problem is to set up multiple connections for each client in an optimized manner and possible purpose of doing this includes providing fault tolerant ability in the case of facility or connection errors, improving performance via parallel access when the capability of one connection is limited, or providing the ability to configure system dynamically without stopping service etc. In this chapter, we studied the QoS-aware object replication for parallel access in the Internet which is formulated as a problem of maximizing the combined download speed for all parallel connections at all clients. Combined with a cost function, the problem is further converted into the metric *Fault Tolerant Facility Location* (*FTFL*) problem to minimize the total cost assuming shortest-path routing is adopted. Due to the NP-hardness of the problem, we proposed an approximation algorithm which is implemented in a distributed and asynchronous manner within $O(n)$ rounds of communication, where n is the number of surrogate servers in the network. The cost of our solution is no more than $|\mathcal{R}| \cdot F^* + 2 \cdot C^*$ through theoretical analysis, where F^* and C^* are respectively the facility cost and connection cost in an optimal solution. Extensive numerical experiments show that the quality of our solutions is comparable to optimal solutions in all cases we evaluated.

Chapter 5

Coordinated En-Route Web Caching in Multi-Server Networks

In the last chapter, we considered the content replication problem for parallel access in a content distribution network. Different from content replication, web caching is to store web objects dynamically at locations that see the object even if the caching scheme is not deployed. In this chapter, we consider the en-route web caching problem in a multi-server network that takes into account all requests (to any server) that pass through the intermediate nodes (caches) on a response path. The objective is to cache the requested object optimally among those caches so that system's total gain is maximized. We derive efficient dynamic programming based methods for finding optimal solutions to the problem for the unconstrained case and two QoS-constrained cases respectively.

5.1 Introduction

Web caching [36] is an important technology to enhance the scalability of Web services. Caching of Web contents (e.g., HTML pages, images) can reduce bandwidth usage, server load and user's perceived latency since these contents can be fetched from proxy caches, which are located more closely than servers to users.

In order to make full use of Web caching, significant research has been conducted in the past decade. These studies include performance optimization in a single cache [31, 118], cooperation among multiple caches [67, 43] and caching architectures [131, 108]. Recent advances in caching technology [38, 66] have presented a new form of caching architecture, namely en-route web caching [11, 110]. In en-route (web) caching, copies of Web contents are selectively placed in transparent en-route caches [66] along each response path, requests passing through later are satisfied by an en-route cache if the requested object is stored, and forwarded to the server otherwise. Different from traditional en-route caching that does not take into account the loss resulted by object replacement when making a placement decision, coordinated en-route caching [127] integrates both object placement and replacement policies into a caching scheme that produces the maximum net gain after loss deduction. Since coordinated en-route caching makes caching decisions in a coordinated fashion between object placement and replacement, it outperforms traditional caching schemes.

Similar to Web caching, deploying multiple servers in a system is an alternative solution to improve the performance of services. Since deploying multiple servers in a network (e.g., CDNs [134]) can alleviate both the server load and network latency, it improves the performance of a network significantly. In addition to this, a multi-server network has the merit of providing fault tolerance and high reliability. However, the complex topology of multi-server networks makes Web caching in this context more challenging.

As far as we know, little work has been done on en-route caching in multi-server networks. Our previous work in [144] studied coordinated en-route caching in dual-server networks, where all requests received by nodes on a response path are destined to two servers at the ends of the path. This chapter extends our previous work by taking into account all requests (to any server, including those not on the path) that pass through the intermediate nodes on a response path.

The main contributions of this chapter are: (1) Establish mathematical formulation for the p-server coordinated en-route caching problem, where p is an arbitrary number of servers. The problem requires to place the requested object optimally among intermediate nodes on a response path so that the maximum benefit to the whole system is achieved wrt all requests (to any server) that pass through these nodes. (2) Derive efficient dynamic programming based methods for finding optimal solutions to the problem for the unconstrained case and two QoS-constrained cases respectively. (3) Present caching schemes to show the application of our methods, which are evaluated on different performance metrics through extensive simulation experiments. The experiment results show that applying our proposed schemes yields a steady performance improvement and achieves desired QoS in a multi-server network.

The rest of the chapter is organized as follows. In Section 5.2, related work on Web caching is discussed. In Section 5.3, system model and formulation of the problem are given. A dynamic programming based method for finding the optimal solution to the unconstrained p-server coordinated en-route caching problem is presented in Section 5.4 and the methods for finding the optimal solution to QoS-constrained p-server coordinated en-route caching problem are presented in Section 5.5. Section 5.6 describes the experimental results and Section 5.7 is the summary of our work.

5.2 Related Work

En-route Web caching is a prospective technology, which is easy to implement and costs little additional bandwidth [37]. Despite its simplicity, en-route caching effectively improves the performance of services by utilizing cache collaboration [7] and selectively placing copies of Web contents in en-route caches.

On cache collaboration that enables requests unsatisfied in one cache to be satisfied in other caches, research was focused on the benefits of cooperative caching for distributed systems and large-scale systems [69, 131]. In [147], wide-area cache cooperation was studied under a simple model, in which distances among all nodes in the network are assumed to be the same. In [63], the authors examined three practical cooperative placement algorithms for large-scale distributed caches and showed that cooperative object placement could significantly improve performance compared to local replacement algorithms, particularly when the sizes of individual caches were small compared to the total size of all cacheable objects. In order to make caches cooperate on a large scale and effectively increase caching population, several caching architectures has been proposed [108], including hierarchical architecture [20], distributed architecture [131, 96] and hybrid architecture [131, 77]. En-route Web caching is based on the hierarchical architecture in a backbone network.

Web caching in a specific architecture comprises two core components: cache placement and content management. Cache placement focuses on the optimal locations of caches in a network. Danzig *et al.* [36] observed the advantage of placing caches inside the backbone rather than at its edges. They showed that the overall reduction in network FTP traffic is higher with caches inside the backbone (core nodes) rather than on the backbone edges (external nodes). The optimal method for cache location inside a backbone was proposed by Krishinan et al [66]. They formulated the cache location problem by modeling the flow of data as flows on a graph and presented optimal methods for both the linear topology and single-source topology.

Content management is the other core component in Web caching. Since the size of cache memory is limited, some contents should be removed to accommodate new contents when a cache gets full. Many replacement policies [11, 114] have been proposed in the past, including LRU (Least Recently Used), LFU (Least Frequently Used) and key-based policies [11]. Cost-based policies were studied recently, which optimizes the replacement by comparing the loss of gain for different decisions. LNC-R [114] is a cost-based policy, which uses the normalized loss of gain to select replacement candidates. LNC-R has been shown to be effective in the context of a single proxy server.

In traditional caching schemes, object replacement in each cache is carried out independently. Tang et al [127] integrated object replacement with placement and proposed coordinated en-route Web caching. Coordinated en-route Web caching regards the removal of objects at intermediate nodes as the loss of a caching decision (because it disables future access to these objects) and then optimizes the decision on a response path by maximizing the net caching gain. It was shown [127] that coordinated en-route caching outperforms other schemes significantly in a linear array. Li et al [72] extended coordinated en-route caching from linear arrays to trees and presented optimal methods for both unconstrained and constrained cases.

Aforementioned work all assumed that there is only one server in the system. This assumption became obsolete with the emergence of various state-of-art networks that contain a group of servers distributed geographically, such as *content distribution networks* (CDNs) [39] and *peer-to-peer* (P2P) file sharing systems [123] which allow file transfer to be performed bi-directionally. A CDN involves creating copies of Web documents and placing these copies at well-chosen servers, with an assurance of different levels of consistency when a replica is updated, and redirecting a client to a server such that the client is optimally served. Qiu et al [98] developed replica placement algorithms that use workload information to make placement decisions which is crucial to a CDN's performance. Mundur and Arankalle [87] addressed the server selection problem for streaming applications by minimizing the cost of serving a video request as measured by network distance. Tang et al [129] investigated the problem of placing replicas under a TTL-based consistency scheme in a tree network and proposed an $O(n^2)$-time algorithm to compute the optimal placement of content replicas. Several research efforts on CDN design are compared, and features of major CDN projects, including Akamai [39], Radar [99], SPREAD [110] and Globule [93], are summarized by Sivasubramanian *et al.* [120].

However, we observed that the existing caching schemes cannot be applied directly to these systems. A naive idea is to decompose a multi-server system into multiple sub-systems each with one server and solve the problem in these single-server systems individually. Unfortunately, this approach does not work because solutions to sub-systems may contradict each other and simply combining them cannot yield an optimal solution to the original system (see the example in Section 5.3).

Thus, it remains a challenging task to find a way to enable coordinated en-route caching in multi-server networks that considers all servers in an integrated fashion and utilizes the advantage of multiple servers to provide QoS guarantees to customers. This chapter addresses the problem by presenting dynamic programming based solutions for both the unconstrained case and two QoS-constrained cases. In the next section, we describe the system model and formulate the problem as that to maximize system's total gain.

5.3 System Model

In a network comprising servers (denoted by set S) and routers (each equipped with a cache), denote the set of all nodes by $\mathcal{V}$ and the set of links by $\mathcal{E}$. When a request that travels along a path is being served by a server/cache s^*, suppose that the intermediate nodes passed by the request are denoted by their indices $\mathcal{A} = \{1, 2, \ldots, n\}$ in increasing order. Note that $\mathcal{A}$ is a path determined dynamically by the deployed request routing scheme according to network status, and may be different in different times. Let $f_s(x)$, $x \in \mathcal{A}, s \in S$, be the *access frequency* to server s observed at node x, that equals the sum of access frequencies of all incoming request flows including those forwarded from x's neighbors and that issued at x.

Figure 5.1 illustrates such a snapshot in a network containing three servers. Assume that before (en-route) placement of cached copies no node except three servers holds the object, and after placement the object is placed at nodes 2, 4 and 5. Request-flows toward a server form a tree rooted at the server. That is, toward a server each node has a unique upstream path and possibly multiple downstream paths. As the result, a node has a unique nearest upstream node holding a copy of the object, but may have several such nearest downstream nodes, each on one downstream path. For example, toward server w node 4 has nearest downstream nodes 2 and 5 holding an object copy after placement. For notational simplicity, we assume in the snapshot that each node issues no request (access frequency 0).

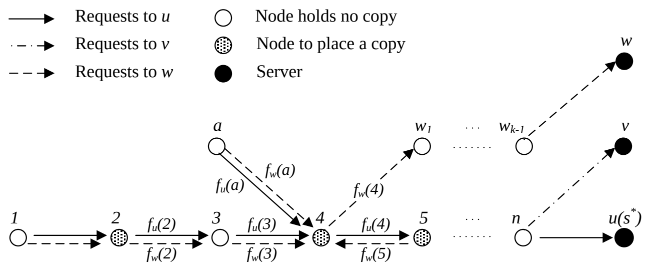


Figure 5.1: System Model in a 3-Server Network

Our objective is to cache an object in a set of nodes in $\mathcal{A}$ along a response path so that, with respect to (wrt) all requests (to all servers) for the object that pass through the nodes on the response path, the total gain of all cached copies is maximized.

As our focus is on en-route caching, we use existing schemes for server selection and request routing. We also assume *routing symmetry* as did by many others, including recently published work [127, 72, 14, 59], i.e., object delivery follows the same path as its request (in opposite direction). Because of the relative stability of network topology and traffic pattern during the short duration of an object delivery session in wired networks, this assumption is reasonable in most cases.

Though the system model of *p*-server coordinated en-route caching is similar to its single-server version [127], the problem becomes more complex when the data flows from each node are multi-directional. We start formulation of this problem from calculation of caching gain in the next subsection.

5.3.1 Calculation of Caching Gain

For requests towards server $s \in S$, the *miss penalty* of object *O* at node x, denoted by $m_s(O, x)$, is defined as the additional cost (e.g. delay) of accessing the object if it is not cached at the local node x. Clearly, this additional cost equals the cost of traveling from x to x's nearest upstream node that holds a copy of object O before placement. Denote by $UE_s(x) \subseteq \mathcal{E}$ the set of links from x to the nearest upstream node holding a copy of O toward s, and by $C(O, e)$ the cost on link e wrt O. We can compute $m_s(O, x)$ by accumulating the cost on each link before the request is satisfied, i.e.

$$m_s(O, x) = \sum_{e \in UE_s(x)} C(O, e). \tag{5.1}$$

Since only the placement of one object is considered, we omit argument O in all parameters throughout the chapter for notational tidiness if no confusion arises.

E.g., in Figure 5.1. $UE_u(4) = \{\langle 4,5\rangle, \langle 5,6\rangle, \ldots, \langle n,u\rangle\}$, $UE_v(4) = \{\langle 4,5\rangle, \langle 5,6\rangle, \ldots, \langle n,v\rangle\}$ and $UE_w(4) = \{\langle 4,w_1\rangle, \langle w_1,w_2\rangle, \ldots, \langle w_{k-1},w\rangle\}$. Assuming that object delivery requires unit cost for each link, we have $m_u(4) = m_v(4) = n-3$ and $m_w(4) = k$.

Let $DV_s(x) \subset \mathcal{V}$ be the set of nodes between x and its nearest downstream node

x' holding a copy of O on every path toward s after placement, where x' is included if $x' \in P$ and excluded otherwise. For the special case when no such x' exists on a path, x' is set to be the first node on that path and included in $DV_s(x)$. E.g., in Figure 5.1, since all nodes issue no request, $f_u(4) = f_u(3) + f_u(a)$, $f_v(4) = 0$, $f_w(4) = f_w(3) + f_w(5) + f_w(a)$; $DV_u(4) = DV_v(4) = \{2, 3, w_1, \ldots, w_{k-1}, a\}$, as there are three paths $[2, 4)$, $(w, 4)$ and $[a, 4)$ via 4 toward servers u, v respectively, $DV_w(4) = \{2, 3, 5\}$.

Suppose $P \subseteq \mathcal{A}$ is a set of nodes to place object O (at each node) and for node $x \in \mathcal{A}$, $x_l \in P \cup \{0\}$ and $x_r \in P \cup \{n+1\}$ are the nearest nodes on the left and right sides of x. (Suppose $1_l = 0$ — dummy node and $n_r = n+1$ — s^*.) Clearly for any x and s, at least one of x_l and x_r is contained in $DV_s(x)$. We denote by $f'_s(x_l, x, x_r)$ the part of $f_s(x)$ that is forwarded by x_l and x_r, and satisfied by the cached copies at x_l and x_r. That is,

$$\begin{aligned} f'_s(x_l, x, x_r) &= \sum_{y \in \{x_l, x_r\} \cap DV_s(x)} f_s(y) \\ &= \begin{cases} f_s(x_l) & if\, x_l \in DV_s(x), x_r \notin DV_s(x), \\ f_s(x_r) & if\, x_l \notin DV_s(x), x_r \in DV_s(x), \\ f_s(x_l) + f_s(x_r) & if\, x_l, x_r \in DV_s(x). \end{cases} \end{aligned} \tag{5.2}$$

E.g., in Figure 5.1 we have $f'_u(2, 4, 5) = f_u(2)$, $f'_v(2, 4, 5) = f_v(2) = 0$ and $f'_w(2, 4, 5) = f_w(2) + f_w(5)$.

Define the *caching benefit* at node x for a given object to be the cost reduction for all requests for the object to all servers. Because all requests at x_l and from its downstream nodes are satisfied by x_l, and those at x_r and from its downstream nodes are satisfied by x_r, caching the object at x will benefit only those requests forwarded from nodes in the segments $(x_l, x]$ ($x_l \rightarrow x$) and $[x, x_r)$ ($x_r \rightarrow x$) respectively. More precisely, it will benefit each of these requests by a cost of $m_s(x)$ given by equation 5.1, since otherwise the request must go to x's nearest upstream node holding a copy. Since there are $f_s(x) - f'_s(x_l, x, x_r)$ requests within these segments in unit time, we have immediately the following equation to compute caching benefit at node x wrt all servers in S:

$$b(x_l, x, x_r, S) = \sum_{s \in S} \{m_s(x) \cdot [f_s(x) - f'_s(x_l, x, x_r)]\}. \tag{5.3}$$

E.g., in Figure 5.1 under the unit link-cost assumption we have

$$\begin{aligned} b(2,4,5,\{u,v,w\}) &= (n-3) \cdot (f_u(4) - f'_u(2,4,5)) + 0 + k \cdot (f_w(4) - f'_w(2,4,5)) \\ &= (n-3)(f_u(4) - f_u(2)) + k(f_w(4) - f_w(2) - f_w(5)) \\ &= (n-3)f_u(a) + kf_w(a). \end{aligned}$$

Because of the limit of cache memory, some objects should be removed from the cache to accommodate a new object and this leads to miss penalties when these removed objects are requested later. We denote the removed objects at x by set Q and the corresponding *caching loss* at x wrt server s by $l_s(x)$ and wrt all servers in S by $l(x, S)$, then the net gain of caching the object in P can be computed by formula

$$G(P, S) = \sum_{x \in P} [b(x_l, x, x_r, S) - l(x, S)]$$

where

$$l(x, S) = \sum_{s \in S} l_s(x) = \sum_{s \in S} \sum_{O' \in Q} m_s(O', x) \cdot f_s(O', x). \tag{5.4}$$

In Figure 5.1, consider the situation of placing object O at node 4. Assume $Q = \{O'\}$, and node 4's nearest upstream nodes holding the removed object O' are n towards servers u and v (happen to be the same) and w_1 toward server w. By equation 5.4 we have

$$l(4, \{u,v,w\}) = \sum_{i=u,v,w} f_i(O', 4)m_i(O', 4) = (n-3)(f_u(O', 4) + f_v(O', 4)) + f_w(O', 4).$$

To determine set Q, we simply use LNC-R replacement policy [114] for the unconstrained p-server caching scheme and the scheme with individual latency constraint. For object O' to be replaced, we denote its size by $size(O')$, the access frequency and miss penalty wrt server s by $f_s(O')$ and $m_s(O')$, then LNC-R replacement policy applies the function

$$\frac{\sum_{s\in S} f_s(O') \cdot m_s(O')}{size(O')} \tag{5.5}$$

to select replacement candidates, i.e., remove object O' with smallest value of $\sum_{s\in S} f_s(O')m_s(O')/size(O')$ from the cache and repeat this until enough space is available to accommodate the new object. We also give a priority based method in Subsection 5.5.3 for the average latency constraint.

5.3.2 Problem Formulation

Given network $(\mathcal{V}, \mathcal{E})$, a set of p servers $S \subset \mathcal{V}$, let $\mathcal{A} \subseteq \mathcal{V} \setminus S$ be a set of nodes passed by a request for object O to server s^*. For any node $x \in \mathcal{A}$, the access frequency and miss penalty for server s, observed at x, are given by $f_s(x)$ and $m_s(x)$ respectively; the caching benefit and caching loss wrt all servers, $b(x_l, x, x_r, S)$ and $l(x, S)$, are computed by equations 5.3 and 5.4 respectively.

Definition 5.1. The p-Server Coordinated En-Route Caching (p-Server CERC) problem is to find set P^* such that

$$\begin{cases} G(P^*, S) = \max\limits_{P\subseteq\mathcal{A}}\{G(P, S)\} = \max\limits_{P\subseteq\mathcal{A}}\{\sum\limits_{x\in P}[b(x_l, x, x_r, S) - l(x, S)]\}, \\ \text{subject to } \xi. \end{cases} \tag{5.6}$$

where ξ is a set of constraints, $|S| = p$, and P^* is called an optimal solution to formula 5.6.

The setting of ξ will be discussed in Section 5.5 after we address the unconstrained *p-Server CERC* problem ($\xi = \phi$) in Section 5.4.

We note that when $p = 1$, this problem equals the single-server problem solved by [127] and [72]. However, their methods cannot be used directly to address p-server problem ($p \geq 2$) since the solutions wrt all servers may contradict each other and simply combining these solutions cannot get the optimal solution to the original network. For example, we consider a simple linear array with two servers at ends as illustrated in Figure 5.2. Suppose $f_0(1) = 4, f_0(2) = 1, f_0(3) = 0, f_3(0) = 0, f_3(1) = 1, f_3(2) =$

2 and $l(1, \{0, 3\}) = 3, l(2, \{0, 3\}) = 5$. If we decompose the access frequency and caching loss into two directions and then solve the two subproblems each with single server as illustrated in subfigures (b) and (c), we get solution $\{1, 2\}$ after combining solutions $\{1\}$ and $\{2\}$ to the subproblems, which differs from the optimal solution $P^* = \{1\}$.

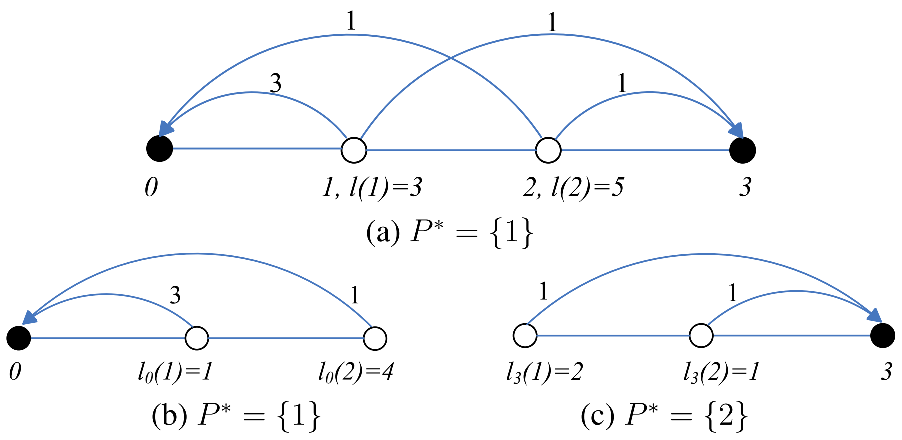


Figure 5.2: An Example of Hardness for Problem Decomposition

5.4 Unconstrained *p*-Server CERC

In this section, we first derive two algorithms to address the unconstrained *p-Server CERC* in Subsection 5.4.1 and 5.4.2, then we present a caching scheme in Subsection 5.4.3 to show the application of our method in practical applications.

Let $\mathcal{A}_x = (1, 2, \ldots, x), x \leq n$ be the first x elements in $\mathcal{A}$ and $P_x(x_r)$ the part of optimal solution which falls in $\mathcal{A}_x$ when $x \in P^*$, i.e., $P_x(x_r) = \{\ldots, x\} = P^* \cap \mathcal{A}_x$, where x_r indicates the next node to expand the solution. According to formula 5.6, the total gain of copies in $P_x(x_r)$ can be denoted by $G(P_x(x_r), S)$. For simplicity of notations, we use $G_x(x_r)$ instead of $G(P_x(x_r), S)$, use $l(x)$ and $b(x_l, x, x_r)$ instead of $l(x, S)$ and $b(x_l, x, x_r, S)$ respectively.

5.4.1 Exhaustive Algorithm

We observe $P_x(x_r) = P_{x_l}(x) \cup \{x\}$, i.e., if $\{\dots, x_l, x\}$ is an optimal solution to the problem in $\mathcal{A}_x$, then $\{\dots, x_l\}$ must be an optimal solution to the problem in $\mathcal{A}_{x_l}$. Considering all possible value of x_l, we have

$$G_x(x_r) = \max_{1 \le x_l < x} \{G_{x_l}(x) + b(x_l, x, x_r) - l(x)\}. \tag{5.7}$$

Note that $l(x)$ will not be affected by removing objects in other location. Thus, we are able to design a simple exhaustive algorithm which builds a solution $P_x(x_r)$ from a smaller solution $P_{x_l}(x)$ recursively and enumerates all combinations of x_l, x and x_r to compute the optimal solution P^*. Although the exhaustive algorithm produces an optimal solution, it does not have an optimal runtime and therefore, we devise a dynamic programming algorithm in the following subsection for better performance.

5.4.2 Dynamic Programming Solution

For any node, there are only two possible caching decisions — placing a copy or not. Therefore we use $P_{\overline{x}}(x_r) = P^* \cap \mathcal{A}_x$ to denote the part of optimal solution which falls in $\mathcal{A}_x$ when $x \notin P^*$ and use $G_{\overline{x}}(x_r)$ to denote the total gain of copies in $P_{\overline{x}}(x_r)$. Using the above definition, we have the following proposition.

Proposition 5.1. *Given node x in path $\mathcal{A} = (1, 2, \dots, n)$ and its nearest node x_r on the right side that holds a copy $(2 \le x_r \le n+1)$, the part of optimal solution to formula 5.6, i.e. $P_x^*(x_r)$, can be obtained by equation*

$$P_x^*(x_r) = \begin{cases} P_{\overline{x}}(x_r) & \text{if } G_{\overline{x}}(x_r) \ge G_x(x_r), \\ P_x(x_r) & \text{otherwise.} \end{cases} \tag{5.8}$$

That is, node x is included in the optimal solution only if placing a copy at x will give more gain. We now need to show how to compute $G_{\overline{x}}(x_r)$ and $G_x(x_r)$ efficiently using dynamic programming techniques.

Let $last(P_{\overline{x}}(x_r))$ be the right-most node in $P_{\overline{x}}(x_r)$, we observe that when x holds a copy, the nearest node in P on the left side of x can either be $x-1$ or be $last(P_{\overline{x-1}}(x))$; when x holds no copy, it can either be $x-1$ or be $last(P_{\overline{x-1}}(x_r))$. We have the following two lemmas to compute $G_{\bar{x}}(x_r)$ and $G_x(x_r)$.

Lemma 5.1. *Given node x in path $\mathcal{A} = (1, 2, \ldots, n)$ and its nearest node x_r on the right side that holds a copy $(2 \leq x_r \leq n+1)$, gain $G_{\bar{x}}(x_r)$ equals to $G^*_{x-1}(x_r)$, i.e.*

$$G_{\bar{x}}(x_r) = max\{G_{\overline{x-1}}(x_r), G_{x-1}(x_r)\}. \tag{5.9}$$

Lemma 5.1 shows that the gain of placing the object optimally in $\mathcal{A}_x$, without a copy placed at x, equals the maximum of that in $\mathcal{A}_{x-1}$ without and with a copy placed at $x-1$.

Proof. When node x holds no copy, we have $(x-1)_r = x_r$ and $P_{\bar{x}}(x_r) = P^*_{x-1}(x_r)$. According to Proposition *5.1*, we have the lemma. □

Lemma 5.2. *Given node x in path $\mathcal{A} = (1, 2, \ldots, n)$ and its nearest node x_r on the right side that holds a copy $(2 \leq x_r \leq n+1)$, gain $G_x(x_r)$ can be computed by equation*

$$\begin{aligned} G_x(x_r) = max\{&G_{\overline{x-1}}(x) + b(last(P_{\overline{x-1}}(x)), x, x_r) - l(x), \\ &G_{x-1}(x) + b(x-1, x, x_r)) - l(x)\}. \end{aligned} \tag{5.10}$$

Lemma 5.2 states that, when a copy is placed at x, the gain of placing the object optimally in $\mathcal{A}_x$ equals the maximum between the gain in $\mathcal{A}_{x-1}$ with a copy at $x-1$ plus the net gain brought by x and that without a copy at $x-1$ plus the net gain brought by x.

Proof. When node x holds a copy, the nearest copy in P on the right side of $x-1$ must be x, i.e. $(x-1)_r = x$. Considering the two possibilities of node $x-1$, as illustrated in Figure 5.3, we have $x_l = last(P_{\overline{x-1}}(x))$ when x does not hold a copy or $x_l = x-1$ otherwise. So $G_x(x_r)$ equals the maximum gain of these two cases. □

Since $last(P_{\overline{x-1}}(x_r))$ and $last(P_{\overline{x-1}}(x))$ can be obtained from $P_{\overline{x-1}}(x_r)$ and $P_{\overline{x-1}}(x)$ respectively, we now can build an optimal solution to the sub-array $\mathcal{A}_x$ from a shorter sub-array $\mathcal{A}_{x-1}$. A dynamic programming algorithm eliminates the need of

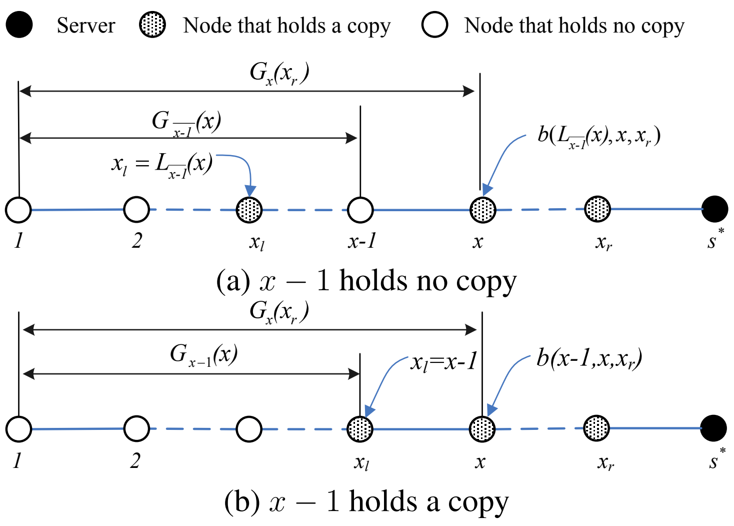


Figure 5.3: Decomposition of Caching Gain

enumerating x_l, which is required in the exhaustive algorithm, by storing the values of $P_{\overline{x}}(x_r)$ and $P_x(x_r)$ for all combination of x and x_r. For notational simplicity, let $\beta(x)$ denote the difference of the two possible values of $G_x(x_r)$ as shown in equation 5.10.

$$\beta(x) = G_{\overline{x-1}}(x) + b(last(P_{\overline{x-1}}(x)), x, x_r) - G_{x-1}(x) - b(x-1, x, x_r). \tag{5.11}$$

Using the above results, we can now show how to compute the optimal solution by left-ward recursion (right-ward iterative expansion) in the following theorem:

Theorem 5.1. *In path $\mathcal{A} = (1, 2, \ldots, n)$, the optimal solution to formula 5.6 is $P_{\overline{n}}(n+1)$ if $G_{\overline{n}}(n+1) \geq G_n(n+1)$, and $P_n(n+1)$ otherwise, where $P_{\overline{n}}(n+1)$ and $P_n(n+1)$ can be computed recursively by the following two equations:*

$$P_{\overline{x}}(x_r) = \begin{cases} P_{\overline{x-1}}(x_r) & if\; G_{\overline{x-1}}(x_r) \geq G_{x-1}(x_r), \\ P_{x-1}(x_r) & otherwise. \end{cases} \tag{5.12}$$

$$P_x(x_r) = \begin{cases} P_{\overline{x-1}}(x) \cup \{x\} & if\; \beta(x) \geq 0, \\ P_{x-1}(x) \cup \{x\} & otherwise. \end{cases} \tag{5.13}$$

Proof. Equation 5.2 can be obtained from Proposition *5.1* and Lemma *5.1*, equation 5.13 can be obtained from Proposition *5.1* and Lemma *5.2*. □

Equation 5.2 shows that the optimal solution in $\mathcal{A}_x$, without a copy at x, equals that in $\mathcal{A}_{x-1}$, without or with a copy at $x-1$, which has the maximum gain. Similarly, equation 5.13 gives the optimal solution in $\mathcal{A}_x$ with a copy at x, i.e., the optimal solution in $\mathcal{A}_{x-1}$ with the addition of node x. Note that in this case the gain calculation follows equation 5.10.

Algorithm 5.1 Unconstrained *p*-Server CERC

```
01: *Step 1. Initialization*
02: $G_{\overline{0}}(x_r) = G_0(x_r) = 0$ for any $x_r$, $1 \le x_r \le n+1$;
03: $P_{\overline{0}}(x_r) = P_0(x_r) = \phi$ for any $x_r$, $1 \le x_r \le n+1$;
04: *Step 2. Iterative procedure*
05: **for** $x = 1$ **upto** $n$
06:     **for** $x_r = x+1$ **upto** $n+1$
07:         *// According to equation 5.12 in Theorem 5.1*
08:         **if** $G_{\overline{x-1}}(x_r) \ge G_{x-1}(x_r)$ **then**
09:             $P_{\overline{x}}(x_r) = P_{\overline{x-1}}(x_r)$;  $(G_{\overline{x}}(x_r) = G_{\overline{x-1}}(x_r))$
10:         **else**
11:             $P_{\overline{x}}(x_r) = P_{x-1}(x_r)$;  $(G_{\overline{x}}(x_r) = G_{x-1}(x_r))$
12:         **endif**
13:         *// According to equation 5.13 in Theorem 5.1*
14:         **if** $\beta(x) \ge 0$ **then**
15:             $P_x(x_r) = P_{\overline{x-1}}(x) \cup \{x\}$;  $(G_x(x_r) = G_{\overline{x-1}}(x) + b(last(P_{\overline{x-1}}(x)), x, x_r))$
16:         **else**
17:             $P_x(x_r) = P_{x-1}(x) \cup \{x\}$;  $(G_x(x_r) = G_{x-1}(x) + b(x-1, x, x_r))$
18:         **endif**
19:     **endfor**
20: **endfor**
21: *Step 3. Get the optimal solution according to Theorem 5.1*
22: **if** $G_{\overline{n}}(n+1) \ge G_n(n+1)$ **then**
23:     $P^* = P_{\overline{n}}(n+1)$;
24: **else**
25:     $P^* = P_n(n+1)$;
26: **endif**
```

By Theorem *5.1*, the algorithm has a simple form as depicted by Algorithm 5.1:

(1) For the base case $x = 0$, we can suppose $G_{\overline{0}}(x_r) = G_0(x_r) = 0$ and $P_{\overline{0}}(x_r) = P_0(x_r) = \phi$ for any x_r, $1 \le x_r \le n+1$.

(2) For $x \ge 1$, we can apply Theorem *5.1* to obtain the solution from a smaller problem.

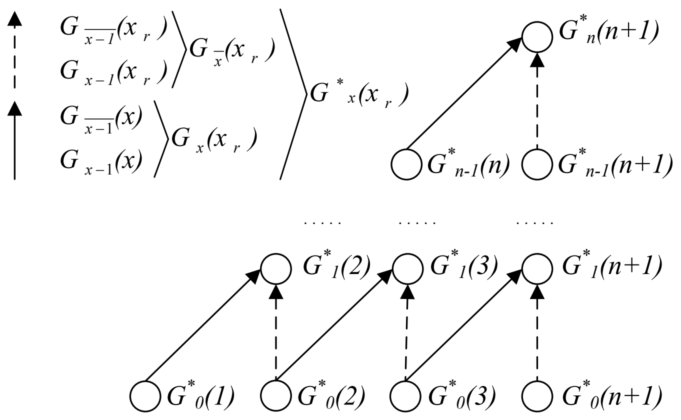


Figure 5.4: Reuse of Intermediate Results

Table 5.1: Process of Algorithm (in terms of $G_x(x_r)$ and $P_x(x_r)$)

x	$x_r = 1$	2	3
0	$0, \phi$	$0, \phi$	$0, \phi$
$\bar{1}$		$0, \phi$	$0, \phi$
1		$1, \{1\}$	$3, \{1\}$
$\bar{2}$			$3, \{1\}$
2			$0, \{1, 2\}$

Algorithm 5.1 saves the results of function calls (G and P) for later reuse, rather than recompute them at each invocation. Generally, when *Algorithm* 5.1 computes $G_{\bar{x}}(x_r)$, the result of $G_{\overline{x-1}}(x_r)$ and $G_{x-1}(x_r)$ can be reused, while it computes $G_x(x_r)$, the result of $G_{\overline{x-1}}(x)$ and $G_{x-1}(x)$ can be reused. The reuse of intermediate results is shown in Figure 5.4. In this figure, horizontal axis delegates x_r, vertical axis delegates x. Intermediate results $G_{\bar{x}}(x_r)$ and $G_x(x_r)$ are computed level by level as the value of x increases, from left to right at each level as x_r increases. This shows that $G_{\bar{n}}(n+1)$ and $G_n(n+1)$ can be computed in $O(n^2)$ steps by a traversal of the tree in Figure 5.4 starting from $G_{\bar{0}}(1)$ and $G_0(1)$. So the time complexity of *Algorithm 5.1* is dominated by the computation of $\beta(x)$ according to equation 5.11. Since $G_{\overline{x-1}}(x)$ and $G_{x-1}(x)$ have already been computed before computing $\beta(x)$, $b(last(P_{\overline{x-1}}(x)), x, x_r)$ and $b(x-1, x, x_r)$ can be computed in p steps according to formula 5.3, computing $\beta(x)$ within each iteration thus requires $O(p)$ time. This yields a total time complexity of $O(pn^2)$ for *Algorithm 5.1*.

We take the example in Figure 5.2 to show the process of *Algorithm* 5.1. Results of each step of the process are described in Table 5.1. Since the maximum caching gain $G(P, S) = max\{G_2(3), G_{\overline{2}}(3)\} = 3$, the optimal solution $P^* = P_{\overline{2}}(3) = \{1\}$.

5.4.3 Caching Scheme

In this subsection, we describe how to implement our above solution in a caching scheme that is ready to work in practical applications.

In *p-Server CERC*, each cache maintains some information on an object (called object descriptor), including the values of $f_s(x), m_s(x), s \in S$ and $l(x)$. Among them, $f_s(x)$ is estimated locally by a "sliding window" technique [118] based on request history and $m_s(x)$ is recorded by the last response message from server s. Caching loss $l(x)$ is estimated according to the LNC-R replacement policy [114]. Since it is technically infeasible to record all downstream nodes of any node, we record only the last node in $\mathcal{A}$ toward each server, denoted by x_s, and compute $f'_s(x_l, x, x_r)$ simply by checking whether $x_l \leq x_s$ and $x_r \geq x_s$ because $x_l \in DV_s(x)$ if $x_l \leq x_s$ and $x_r \in DV_s(x)$ if $x_r \geq x_s$.

Our caching scheme works as follows: When a request passes through a node, the node checks the requested object in its cache. If the object is found, the node makes a caching decision and sends a response message to the client; otherwise, the node attaches the information of $f_s(x), m_s(x), s \in S$ and $l(x)$ to the request message and forwards it to the next node until a server (or a cache that holds the requested object) is reached. At a node holding the requested object, a caching decision is made by executing *Algorithm 5.1* with the object information at all intermediate nodes being given. Then, the node packs the caching decision together with the required object into a response message and sends it back to the client. When the response message passes through an intermediate node, the node adjusts the contents in its cache according to the caching decision. In this process, miss penalties of both the requested object and the removed objects should be updated. The miss penalty of the requested object at each node is updated by accumulating the cost on each link that the response has traveled since the last node which was instructed to hold a copy. To avoid unnecessary communication overhead, miss penalties of the removed objects in the neighboring servers are updated when subsequent requests (responses) passing through the current node reaches these servers, by accumulating the cost of the links the request (response) has traversed.

5.4.4 Discussion

5.4.4.1 Optimization

The method in Subsection 5.4.2 looks for optimal solutions in a candidate set comprising all intermediate nodes that the request message has passed through. Here we optimize the method by reducing the size of the candidate set.

First, we give a proposition to show the property of nodes which are instructed to hold a copy in a caching decision.

Proposition 5.2. *If x is a node in the optimal solution to formula 5.6, then we have $\sum_{s\in S} m_s(x)f_s(x) - l(x) \geq 0$.*

Proof. Suppose there is a node $c \in P$ such that $\sum_{s\in S} m_s(c)f_s(c) - l(c) < 0$. Let node b be the nearest node in $P \cup \{0\}$ on the left side and node d the nearest node in $P \cup \{n+1\}$ on the right side. We define $b(0_l, 0, 0_r) = l(0) = b(n+1_l, n+1, n+1_r) = l(n+1) = 0$. Then we have $b(b, c, d) - l(c) = \sum_{s\in S} m_s(c)(f_s(c) - f'_s(b, c, d)) - l(c) < 0$, $b(b_l, b, c) < b(b_l, b, d)$ and $b(c, d, d_r) < b(b, d, d_r)$. So, we have

$$\begin{aligned} G(P) &= \sum_{x<b\vee x>d} (b(x_l, x, x_r) - l(x)) + b(b_l, b, c) - l(b) \\ &\quad +b(b, c, d) - l(c) + b(c, d, d_r) - l(d) \\ &< \sum_{x<b\vee x>d} (b(x_l, x, x_r) - l(x)) + b(b_l, b, d) - l(b) \\ &\quad +b(b, d, d_r) - l(d) \\ &= G(P \setminus \{c\}). \end{aligned}$$

From the inequality, we know that $P \setminus \{c\}$ is a better solution than P, which contradicts the fact that P is an optimal solution. The proposition is proven. □

We say that nodes satisfying this proposition are *locally beneficial*. The proposition implies that we should only consider placing copies among theses *locally beneficial* nodes.

5.4.4.2 Impact of Routing asymmetry

Routing asymmetry, i.e. response may not follow the same path as the request, arises when a network has a frequently changing topology and traffic pattern, e,g, a wireless network containing a large number of mobile nodes. Our scheme can still be used when the routing is asymmetrical, however the solution deployed practically in this case is not optimal and optimal caching scheme in this case is still an open problem. Specifically, for a request path $\mathcal{A}$ from a client to a server/cache holding the desired object, let $\mathcal{B}$ denote the response path from that server/cache to the client. Clearly $P \subseteq (\mathcal{A} \setminus \mathcal{B}) \cup (\mathcal{A} \cap \mathcal{B})$. For each node $x \in P$, caching remains the same as in the routing symmetry case if $x \in \mathcal{A} \cap \mathcal{B}$, and is made at a node in $\mathcal{B}$ that is of the best interest (e.g. closest) to x instead at x in order to avoid additional transfer and as well as retain a maximal amount of the original benefit of caching at x otherwise.

5.4.4.3 Server selection and cache consistency

Our scheme can be combined with a simple strategy of best server selection: select the server that has the minimum access cost (to the request issuer) among all servers. This cost can be accumulated distance, delay, miss penalty or a combination of them as desired by applications. The best server is chosen by each request dynamically at the time prior to request routing. Cache consistency can be maintained by deploying a server invalidation scheme [65] that requires each server to keep track of a list of caches and send an update message to them upon detecting any change.

5.5 QoS-Aware *p*-Server CERC

In unconstrained *p-Server CERC*, only the access frequency of an object is considered. This is appropriate in a best-effort network because Web contents having the same access frequency are treated equally. However, it is not the case when the priorities of different Web contents are concerned in a capacity-limited network. In this situation, quality of service should be introduced.

Quality of Service (QoS) [140] refers to control mechanisms that can provide different priorities to different users and guarantee a certain level of performance to a data flow in accordance with requests from applications. QoS guarantees are very important in delay-sensitive applications and capacity-limited networks. As far as an ISP (Internet Service Provider) is concerned, there are two types of QoS provided, which are respectively designed for satisfying the requirements of content providers and the requirements of content consumers. Generally speaking, QoS requirements of these two types of customers are different. Content consumers are concerned with the perceived latency on individual customers, while content providers are concerned with the average latency perceived by all customers. In this section, we propose two constraint-based approaches to solve *p*-Server CERC wrt these QoS requirements.

5.5.1 QoS Constraints

We consider the following settings of ξ, i.e. set of constraints (see Section 5.3), for QoS requirements of individual latency and average latency:

- ◇ $\forall x \in \mathcal{A} \setminus P$, $\alpha(x, x_r) = d(x, x_r) - q(x) \leq 0$.

  In this constraint, $d(x, x_r)$ is the cost (e.g., delay) of all links between x and x_r, $q(x)$ is the QoS requirement at node x. This constraint states that each cost perceived by an individual node should be smaller than the QoS requirement at that node.

- ◇ $\overline{d}(x, x_r) = \frac{m^*}{|P|+1} \leq q(O), x \in \{0\} \cup P$.

  In this constraint, node 0 is a dummy node, $m^* \approx m_{s^*}(1)$ is the total miss penalty on the whole path, $|P|$ is the number of placed copies in solution P and $q(O)$ is the QoS requirement of object O. This constraint states that the average cost perceived by all nodes should be smaller than the QoS requirement of object O.

5.5.1.1 Individual Latency Constrained *p*-Server CERC

When the QoS constraint on individual latency is concerned, the problem can be formulated as

$$\begin{cases} G(P^*, S) = \max_{P \subseteq \mathcal{A}} \{ \sum_{x \in P} [b(x_l, x, x_r) - l(x)] \}. \\ \text{subject to } \forall x \in \mathcal{A} \setminus P^*, \alpha(x, x_r) \leq 0. \end{cases} \tag{5.14}$$

We use $P_x(x_r) = P^* \cap \mathcal{A}_x$ to denote the part of optimal solution which falls in $\mathcal{A}_x$ when $x \in P^*$ and $P_{\bar{x}}(x_r)$ the part of optimal solution when $x \notin P^*$. The gains of copies in $P_x(x_r)$ and $P_{\bar{x}}(x_r)$ are denoted respectively by $G_x(x_r)$ and $G_{\bar{x}}(x_r)$, then we have the following proposition.

Proposition 5.3. *Given node x in path $\mathcal{A} = (1, 2, \ldots, n)$ and its nearest node x_r on the right side that holds a copy $(2 \leq x_r \leq n+1)$, the part of optimal solution to formula 5.14, i.e. $P_x^*(x_r)$, can be obtained from equation*

$$P_x^*(x_r) = \begin{cases} P_{\bar{x}}(x_r) & \textit{if } G_{\bar{x}}(x_r) \geq G_x(x_r) \textit{ and } \alpha(x, x_r) \leq 0, \\ P_x(x_r) & \textit{otherwise}. \end{cases} \tag{5.15}$$

Using the same definition of $\beta(x)$ as in equation 5.11, we have the following two lemmas to compute $G_{\bar{x}}(x_r)$ and $G_x(x_r)$.

Lemma 5.3. *Given node x in path $\mathcal{A} = (1, 2, \ldots, n)$ and its nearest node x_r on the right side that holds a copy $(2 \leq x_r \leq n+1)$, gain $G_{\bar{x}}(x_r)$ can be computed by equation*

$$G_{\bar{x}}(x_r) = \begin{cases} G_{\overline{x-1}}(x_r) & \textit{if } G_{\overline{x-1}}(x_r) \geq G_{x-1}(x_r) \textit{ and } \alpha(x-1, x_r) \leq 0, \\ G_{x-1}(x_r) & \textit{otherwise}. \end{cases} \tag{5.16}$$

Proof. When node x holds no copy, we have $P_{\bar{x}}(x_r) = P_{x-1}^*(x_r)$ because $(x-1)_r = x_r$. According to Proposition 5.3, we have the lemma: □

Lemma 5.4. *Given node x in path $\mathcal{A} = (1, 2, \ldots, n)$ and its nearest node x_r on the right side that holds a copy $(2 \leq x_r \leq n+1)$, gain $G_x(x_r)$ can be computed by $G_{\overline{x-1}}(x) + b(last(P_{\overline{x-1}}(x)), x, x_r) - l(x)$ if $\beta(x) \geq 0$ and $\alpha(x-1, x) \leq 0$, otherwise $G_{x-1}(x) + b(x-1, x, x_r) - l(x)$.*

Proof. When node x holds a copy, x must be the nearest node at the right side of $x-1$ in P, i.e. $(x-1)_r = x$. Consider the two possibilities of node $x-1$, we have $x_l = last(P_{\overline{x-1}}(x))$ if no copy at $x-1$ or $x_l = x-1$ otherwise. The lemma follows immediately by Proposition *5.3*. □

Now, we can build an optimal solution to the sub-array $\mathcal{A}_x$ from a shorter sub-array $\mathcal{A}_{x-1}$. We have thus the following theorem:

Algorithm 5.2 Individual Latency Constrained *p*-Server CERC

```
01: Step 1. Initialization
02: $G_{\overline{0}}(x_r) = G_0(x_r) = 0$ for any $x_r$, $1 \leq x_r \leq n+1$;
03: $P_{\overline{0}}(x_r) = P_0(x_r) = \phi$ for any $x_r$, $1 \leq x_r \leq n+1$;
04: Step 2. Iterative procedure
05: for $x = 1$ upto $n$
06:     for $x_r = x+1$ upto $n+1$
07:         // According to equation 5.17 in Theorem 5.2
08:         if $G_{\overline{x-1}}(x_r) \geq G_{x-1}(x_r)$ and $\alpha(x-1, x_r) \leq 0$ then
09:             $P_{\overline{x}}(x_r) = P_{\overline{x-1}}(x_r)$;
10:         else
11:             $P_{\overline{x}}(x_r) = P_{x-1}(x_r)$;
12:         endif
13:         // According to equation 5.18 in Theorem 5.2
14:         if $\beta(x) \geq 0$ and $\alpha(x-1, x) \leq 0$ then
15:             $P_x(x_r) = P_{\overline{x-1}}(x) \cup \{x\}$;
16:         else
17:             $P_x(x_r) = P_{x-1}(x) \cup \{x\}$;
18:         endif
19:     endfor
20: endfor
21: Step 3. Get the optimal solution according to Theorem 5.2
22: if $G_{\overline{n}}(n+1) \geq G_n(n+1)$ and $\alpha(n, n+1) \leq 0$ then
23:     $P^* = P_{\overline{n}}(n+1)$;
24: else
25:     $P^* = P_n(n+1)$;
26: endif
```

Theorem 5.2. *In path $\mathcal{A} = (1, 2, \ldots, n)$, the optimal solution of formula 5.14 is $P_{\overline{n}}(n+1)$ if $G_{\overline{n}}(n+1) \geq G_n(n+1)$ and $\alpha(n, n+1) \leq 0$, and $P_n(n+1)$ otherwise, where $P_{\bar{n}}(k, n+1)$ and $P_n(k, n+1)$ can be computed recursively by the following* *two equations:*

$$P_{\overline{x}}(x_r) = \begin{cases} P_{\overline{x-1}}(x_r) & \textit{if } G_{\overline{x-1}}(x_r) \geq G_{x-1}(x_r) \textit{ and } \alpha(x-1, x_r) \leq 0, \\ P_{x-1}(x_r) & \textit{otherwise}. \end{cases} \tag{5.17}$$

$$P_x(x_r) = \begin{cases} P_{\overline{x-1}}(x) \cup \{x\} & \textit{if } \beta(x) \geq 0 \textit{ and } \alpha(x-1, x) \leq 0, \\ P_{x-1}(x) \cup \{x\} & \textit{otherwise}. \end{cases} \tag{5.18}$$

Proof. Equation 5.17 can be obtained from Proposition *5.3* and Lemma *5.3*, equation 5.18 can be obtained from Proposition *5.3* and Lemma *5.4*. □

Our algorithm for *the individual-latency-constrained p-Server CERC* is depicted by Algorithm 5.2. The structure of *Algorithm 5.2* is same as *Algorithm 5.1* except that constraint α is added in all conditions. So the time complexity of *Algorithm 5.2* is also $O(pn^2)$.

5.5.1.2 Average Latency Constrained *p*-Server CERC

QoS constraint on average latency $\overline{d}(x, x_r) = \frac{m^*}{|P|+1} \leq q(O)$ can be transferred to a constraint on the number of copies, i.e. $|P| \geq m^*/q(O) - 1$. We first consider *exactly-k-copies* constraint and formulate the problem as

$$\begin{cases} G(P^*, S) = \max\limits_{P \subseteq \mathcal{A}} \{\sum\limits_{x \in P} [b(x_l, x, x_r) - l(x)]\}, \\ \text{subject to } |P| = k(m^*, q(O)), \end{cases} \tag{5.19}$$

where the required number of copies, i. e. k, is no more than the total number of nodes on the path (otherwise there is no solution to the problem). The specific value of k is usually given by a table according to the value of m^* and $q(O)$. Further, we define $P_x(j, x_r), j \leq min\{x, k\}$ as the part of optimal solution comprising j copies that falls in $\mathcal{A}_x$ when $x \in P^*$ and $P_{\overline{x}}(j, x_r), j \leq min\{x-1, k\}$ as the part of optimal solution when $x \notin P^*$. $G_{\overline{x}}(j, x_r)$ and $G_x(j, x_r)$ are the total gains respectively to $P_{\overline{x}}(j, x_r)$ and $P_x(j, x_r)$. Then, we have the following proposition.

Proposition 5.4. *Given node x in path $\mathcal{A} = (1, 2, \ldots, n)$ and its nearest node x_r on the right side that holds a copy $(2 \leq x_r \leq n+1)$, the part of optimal solution to formula 5.19, i.e. $P^*_x(j, x_r)$, can be obtained from equation*

$$P^*_x(j, x_r) = \begin{cases} P_{\overline{x}}(j, x_r) & \text{if } G_{\overline{x}}(j, x_r) \geq G_x(j, x_r), \\ P_x(j, x_r) & \text{otherwise.} \end{cases} \tag{5.20}$$

We have the following two lemmas to compute $G_{\bar{x}}(x_r)$ and $G_x(x_r)$.

Lemma 5.5. *Given node x in path $\mathcal{A} = (1, 2, \ldots, n)$ and its nearest node x_r on the right side that holds a copy $(2 \leq x_r \leq n+1)$, gain $G_{\overline{x}}(j, x_r)$ equals to $G^*_{x-1}(j, x_r)$, i.e.*

$$G_{\overline{x}}(j, x_r) = max\{G_{\overline{x-1}}(j, x_r), G_{x-1}(j, x_r)\}. \tag{5.21}$$

Proof. When node x holds no copy, since $(x-1)_r = x_r$, we have $P_{\overline{x}}(j, x_r) = P^*_{x-1}(j, x_r)$. The lemma follows directly by Proposition 5.4. □

Lemma 5.6. *Given node x in path $\mathcal{A} = (1, 2, \ldots, n)$ and its nearest node x_r on the right side that holds a copy $(2 \leq x_r \leq n+1)$, gain $G_x(x_r)$ can be computed by equation*

$$\begin{aligned} G_{x-1}(j, x_r) = \; & max\{G_{\overline{x-1}}(j-1, x) + b(last(P_{\overline{x-1}}(x)), x, x_r) - l(x), \\ & G_{x-1}(j-1, x) + b(x-1, x, x_r) - l(x)\}. \end{aligned} \tag{5.22}$$

Proof. When node x holds a copy, x must be the nearest node on the right side of $x-1$ in P, i.e. $(x-1)_r = x$. Considering the two possibilities of node $x-1$, we have $x_l = last(P_{\overline{x-1}}(j, x))$ or $x_l = x - 1$. So the lemma holds according to Proposition 5.4. □

Now, we can build an optimal solution to the sub-array $\mathcal{A}_x$ from a shorter sub-array $\mathcal{A}_{x-1}$. Suppose $\beta(j, x) = G_{\overline{x-1}}(j-1, x_r) + b(last(P_{\overline{x-1}}(x_r)), x, x_r) - G_{x-1}(j-1, x_r) - b(x-1, x, x_r)$, we have the following theorem.

Theorem 5.3. *In path $\mathcal{A} = (1, 2, \ldots, n)$, the optimal solution of formula 5.19 is $P_{\overline{n}}(k, n+1)$ if $G_{\overline{n}}(k, n+1) \geq G_n(k, s^*)$, and $P_n(k, s^*)$ otherwise, where $P_{\bar{n}}(k, n+1)$*

and $P_n(k, n+1)$ *can be computed recursively by the following two equations:*

$$P_{\overline{x}}(j, x_r) = \begin{cases} P_{\overline{x-1}}(j, x_r) & if\, G_{\overline{x-1}}(j, x_r) \geq G_{x-1}(j, x_r), \\ P_{x-1}(j, x_r) & otherwise. \end{cases} \tag{5.23}$$

$$P_x(j, x_r) = \begin{cases} P_{\overline{x-1}}(j-1, x) \cup \{x\} & if\, \beta(j, x) \geq 0, \\ P_{x-1}(j-1, x) \cup \{x\} & otherwise. \end{cases} \tag{5.24}$$

Proof. Equation 5.23 can be obtained from Proposition *5.4* and Lemma *5.5*, equation 5.24 can be obtained from Proposition *5.4* and Lemma *5.6*. □

Algorithm 5.3 Average Latency Constrained *p*-Server CERC

```
01: Step 1. Initialization
02: $G_{\overline{0}}(0, x_r) = G_0(0, x_r) = 0$ for any $x_r$, $1 \leq x_r \leq n+1$;
03: $P_{\overline{0}}(0, x_r) = P_0(0, x_r) = \phi$ for any $x_r$, $1 \leq x_r \leq n+1$;
04: Step 2. Iterative procedure
05: for $x = 1$ upto $n$
06:     for $x_r = x+1$ upto $n+1$
07:         for $j = 0$ upto $\min\{x, p\}$ do
08:         // According to equation 5.23 in Theorem 5.3
09:         if $G_{\overline{x-1}}(j, x_r) \geq G_{x-1}(j, x_r)$ then
10:             $P_{\overline{x}}(j, x_r) = P_{\overline{x-1}}(j, x_r)$;
11:         else
12:             $P_{\overline{x}}(j, x_r) = P_{x-1}(j, x_r)$;
13:         endif
14:         // According to equation 5.24 in Theorem 5.3
15:         if $\beta(j, x) \geq 0$ then
16:             $P_x(j, x_r) = P_{\overline{x-1}}(j-1, x) \cup \{x\}$;
17:         else
18:             $P_x(j, x_r) = P_{x-1}(j-1, x) \cup \{x\}$;
19:         endif
20:         endfor
21:     endfor
22: enddo
23: Step 3. Get the optimal solution according to Theorem 5.3
24: if $G_{\overline{n}}(k, n+1) \geq G_n(k, n+1)$ then
24:     $P^* = P_{\overline{n}}(k, n+1)$;
26: else
27:     $P^* = P_n(k, n+1)$;
28: endif
```

Now, the algorithm becomes straightforward:

(1) For the base case $x = 0$, we can suppose $G_{\bar{0}}(0, x_r) = G_0(0, x_r) = 0$ and $P_{\bar{0}}(0, x_r) = P_0(0, x_r) = \phi$ for any x_r, $1 \leq x_r \leq n + 1$;

(2) For $x \geq 1$, we can apply Theorem *5.3* to obtain the solution from a smaller sub-problem.

Our algorithm for *the average-latency-constrained p-Server CERC (exactly-k-copies)* is depicted by Algorithm 5.3.

The structure of *Algorithm 5.3* is similar to *Algorithm 5.1*, except that it has one more nested loop on j. So the time complexity of *Algorithm 5.3* is at most k times of that of *Algorithm 5.1*, i.e., $O(pkn^2)$. We also note that, in *Algorithm 5.3*, not only the solution corresponding to constraint *exactly-k-copies* is computed, but also for any *exactly-i-copies* constraint ($i \leq k$), the solution is computed.

5.5.2 Extended Constraints on Copy Number

The previous policy controls the quality of service by specifying the number of copies needed on a response path. However, sometimes we may need more flexible policies to control a system. This can be achieved by adapting previous algorithm. For example, our algorithm can be easily adapted to the following QoS constraints.

5.5.2.1 At most k copies

To limit the resources occupied by an object so that other objects have sufficient resources as well, we propose constraint *at-most-k-copies*. From the previous subsection, we know that wrt a constraint *exactly-k-copies*, the solutions satisfying *exactly-i-copies* ($i \leq k$) are all obtained by *Algorithm 5.3*, so the only task left is to find the maximum gain of these solutions. The total time complexity of the algorithm under constraint *at-most-k-copies* follows *Algorithm 5.3*, i.e., $O(pkn^2)$. We also note that this constraint has the same effect as the unconstrained case when $k \geq k^*$, where k^* is the number of copies in the optimal solution to the unconstrained case. However, It's a little surprising when we learn from the experiments that this constraint does not always have the same effect as *exactly-k-copies* when $k < k^*$, although most of the time it is true.

5.5.2.2 At least k copies

As illustrated in the previous subsection, the QoS requirement on average latency can be transformed to *at-least-k-copies*. In this case, we set $k = n$, then the solutions satisfying *exactly-i-copies* ($i \leq n$) are all obtained by *Algorithm 5.3* and the only task left is to find the maximum gain among these solutions which can be done in time $O(n)$. So the total time complexity of the algorithm under constraint *at-least-k-copies* equals that of *Algorithm 5.3* when $k = n$, i.e., $O(pn^3)$. We also note that this constraint has the same effect as the unconstrained case when $k \leq k^*$, where k^* is the number of copies in the optimal solution in the unconstrained case. However, the experiments show again that this constraint does not always have the same effect as *exactly-k-copies* when $k > k^*$.

5.5.3 QoS-Aware Caching Schemes

A QoS-aware caching scheme also needs to maintain object descriptors in each cache, including *access frequency, miss penalty* and *caching loss*. When the constraint on individual latency is concerned, in addition to $f_s(x), m_s(x)$ and $l(x)$, $q(x)$ should also be maintained at each node and attached to a request message when it passes through. For the constraint on average latency, a performance table should be estimated in advance to show the number of copies needed as a function of the total miss penalty and the QoS requirement of the object, denoted by $k(m^*, q(O))$. In this case, our QoS-aware caching scheme works as follows:

When a request is being forwarded to a server, each node on the path piggybacks the required information (including $f_s(x), m_s(x)$ and $l(x)$) on the request message. When the request arrives at a server (or a cache holding the requested object), the server first computes the value of k according to function $k(m^*, q(O))$ and then executes *Algorithm 5.3* to obtain a caching decision based on the piggybacked information. Finally, the server sends the decision, together with the requested object, back to the client node. Along the way, intermediate nodes on the path adjust their cache contents according to the caching decision. If a node is instructed to cache the object, then the content of its cache is updated according to the replacement policy deployed.

We note that traditional replacement policy does not take QoS requirement (or priority) into account. For example, LNC-R replacement policy [114] applies the formula 5.5 to select replacement candidates. In such a policy, an object O' of higher priority will be replaced before those of lower priorities since its miss penalty $(m(O'))$ is smaller for the same access frequency $(f(O'))$ and size $(s(O'))$. Here, we normalize formula 5.5 by the priority (i.e. QoS requirement) of the object and use function *LNP (loss normalized by priority)*

$$\frac{\sum_{s \in S}(f_s(O')m_s(O'))}{size(O')q(O')} \tag{5.25}$$

to select replacement candidates. According to this policy, those objects with smallest value of expression (5.25) will be removed from the cache until enough space is available to accommodate a new object.

5.6 Simulation Experiments

5.6.1 Simulation Model

5.6.1.1 Parameter settings

We implemented our schemes using simulation experiments as did in the literature by others [126, 129, 14, 59] on networks comprising multiple servers and a large number of routers. As there is no real trace data available in the open literature which we can use to simulate our caching schemes, we generate the topologies of the network from empirical results by the Tier program [19], request rate at each router randomly under uniform distribution, and access pattern (frequency to a specific object) following Zipf parameters. The network is composed of a Wide Area Network (WAN) and a number of Metropolitan Area Networks (MANs, a term used in Tiers). The WAN is used as a core network and each MAN acts as an edge network (or access network). Table 5.2 shows the parameters and their values used in our experiments. These values are chosen similar to those in [134].

Table 5.2: Parameter Settings

Parameters	*Value*
Total number of nodes	320
WAN nodes / MAN nodes	1:1
Number of network links	506
Number of sites	32
Average delay of WAN links	0.5 second
Average delay of MAN links	0.06 second
Number of objects per site	2000
Relative cache size	4%
Request rate	U(1,9)/sec
Object size	average size: 30KB
	hybrid distribution [9]
Access frequency	server: Zipf, $1/i^{\alpha}$, $\alpha = 0.8$
	object: Zipf, $1/j^{\beta}$, $\beta = 0.8$

Among these parameters, cache size is described as the total relative size of all objects available in the content server and the object sizes follow the distribution described in [9] with the average size *30KB*. To simulate the requests made by the clients, a continuous request stream is randomly generated at each node and the average request rate of each node follows the distribution of *U(1, 9)*, where *U(x, y)* denotes the uniform distribution between *x* and *y*. User access pattern *[27]* is the character of user behavior when requesting documents. Some early studies [15] gave evidence that the relative frequency of web pages follows Zipf's law, which states that the relative frequency of a request for the i^{th} most popular web page is proportional to $1/i$.

In the experiments, routing paths from all nodes to a given server are set to be the shortest-path tree rooted at the server. The server to respond is selected dynamically according to a minimum cost as described in Section IV-D. The cached copies are assumed to be consistent with those on the servers to facilitate performance comparison among different caching schemes.

5.6.1.2 Caching schemes evaluated

The objective of simulation experiments is to evaluate the feasibility, effectiveness and scalability of our proposed schemes. We design four groups of experiments to evaluate

Table 5.3: Groups of Experiments

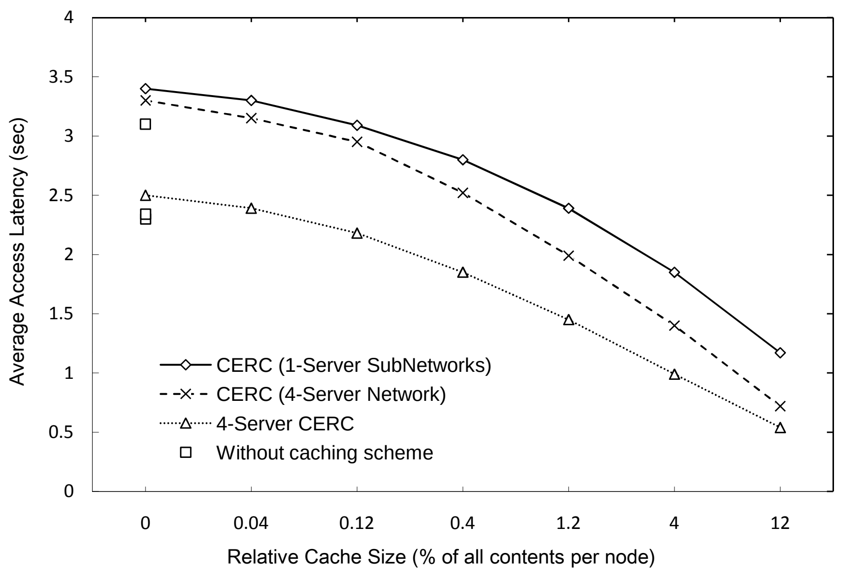
Group	*Parameters*	*Constraints*	*# of Servers*
1	Cache size	Unconstrained	4
2	Access pattern	Unconstrained	4
	# of Copies	Exactly-k-copies	2
		At-most-k-copies	2
		At-least-k-copies	2
4	# of Servers	Unconstrained	2,4,8,16

the performance of our schemes in different aspects. The effectiveness of our schemes are shown in the first two groups of experiments through comparisons with applying single-server en-route caching in a 4-server network and its decomposed 1-server sub-networks. First, we evaluate the performance improvement with the increase of cache size to show the feasibility of deploying our schemes in a 4-server network. Then we perform similar comparisons for different access patterns measured by Zpif parameters. The impacts of the number of cache copies and number of deployed servers on our schemes are also evaluated in the last two groups of experiments. In the third group, we evaluate the QoS performance on specific number of (cache) copies for deployment of our schemes with different constraints. Finally, the performance on a wide range of server numbers is evaluated to show the scalability of proposed schemes. Our comparisons are made on the metrics of average access latency, cache hit ratio, highest server load and average bandwidth consumption because they are the major performance parameters in the caching literature.

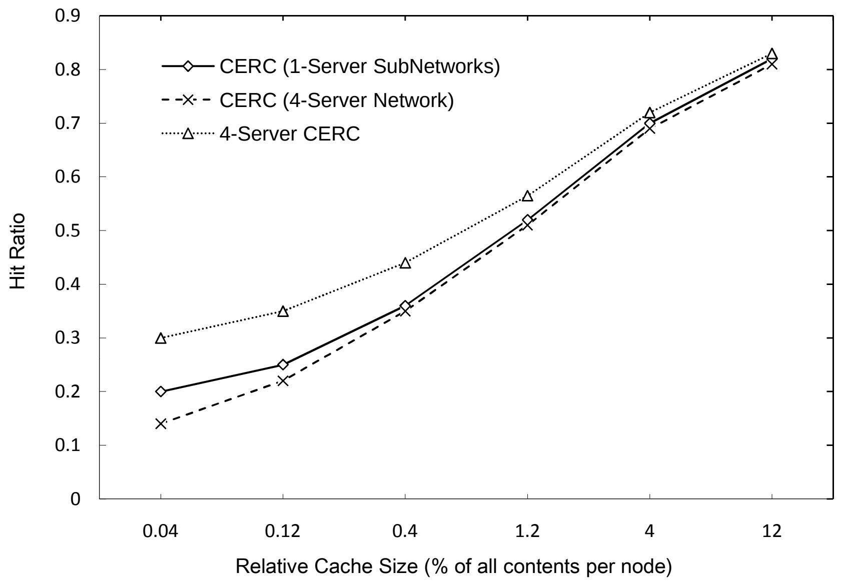
As all our caching schemes exhibit similar properties in performance, for illustration simplicity and as well as coverage of all schemes, we use the unconstrained scheme in group 1, 2 and 4 experiments, and three constrained schemes in group 3 experiments. The detailed information of the four groups is listed in Table 5.3.

5.6.2 Result of Experiments

5.6.2.1 Impact of cache size



(a)



(b)

Figure 5.5: Impact of Cache Size

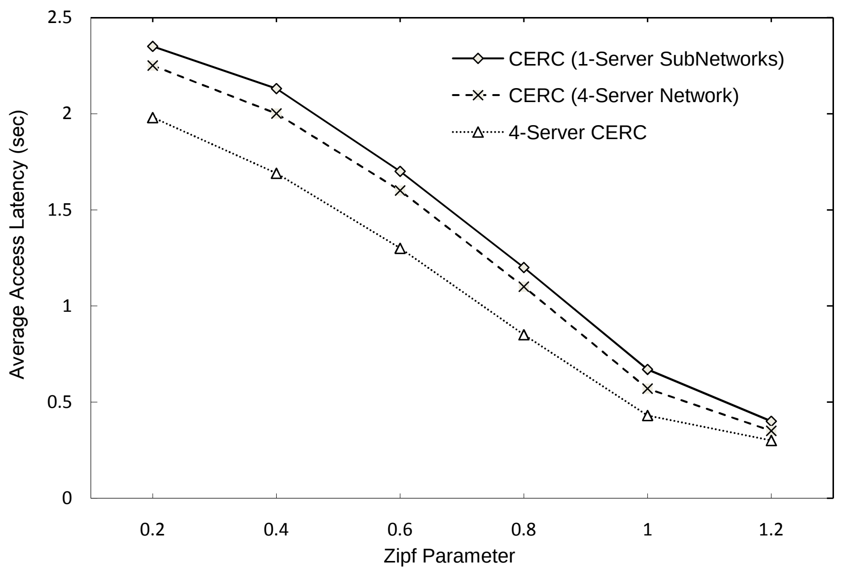
In this experiment, we compare the performances of three different schemes. We implement the currently known best single-server coordinated en-route caching scheme [127] in a 4-server network and its decomposed 1-server subnetworks, as two direct applications of existing single-server caching techniques in a multi-server environment, and our multi-server caching scheme in the same network. The performances of these schemes on average access latency and hit ratio vs relative cache size are evaluated and compared in Figure 5.5.

First, the experiments were made across a wide range of cache sizes, from *0.04* percent to *12* percent. Metrics of the average access latency and hit ratio are chosen to evaluate the performance of our unconstrained scheme, where access latency is the latency perceived by users and hit ratio is the ratio of the number of requests served by caches to the number of total requests. For the purpose of comparison, average access latency is also evaluated when no caching scheme is employed as shown by square points in Figure 5.5. Figure 5.5(a) shows that deployment of caching scheme is beneficial when the relative cache size reaches around 0.04% and performance improvement increases with the increase of cache size. Figure 5.5(b) shows that hit ratio increases with the increase of relative cache size. Hit ratio becomes zero when no caching scheme is deployed. But considering the expensive cost of cache memory, an optimal method will involve an appropriate trade-off between cache cost and cache performance.

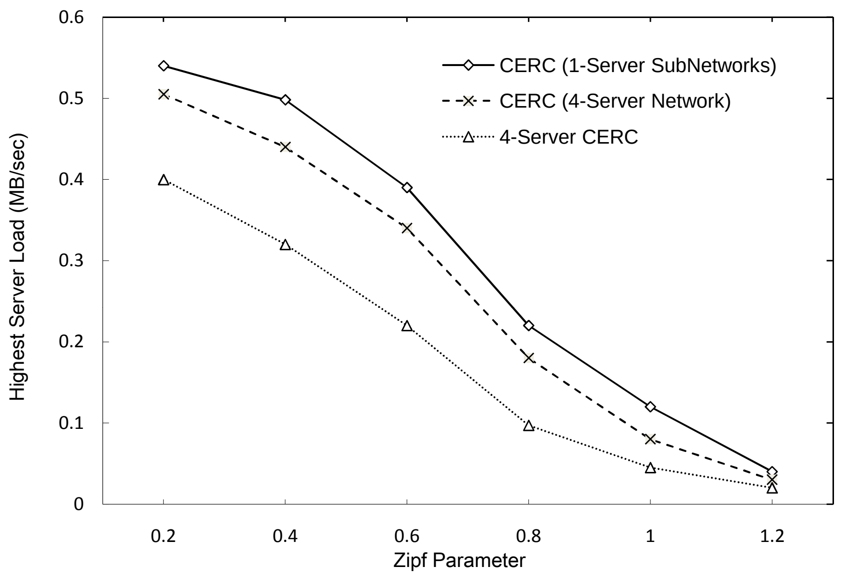
5.6.2.2 Impact of access pattern

Here, we do the same comparison in a variety of access frequency distributions measured by Zipf parameters to show the impact of access pattern. The Zipf parameters for servers and objects (i.e., α and β) are assigned equal values between 0.2 and 1.2. Figure 5.6(a) shows that our scheme has significantly lower average access latency than that of two direct applications of single-server caching. Similarly, Figure 5.6(b) shows that the hit ratio for our scheme is consistently higher than that of other two schemes. In this two figures, we find that the relative performance difference among the caching schemes is not very large when Zipf parameter is very small or very large. This is because all objects have similar access frequencies when the parameter is very small and only a few objects are accessed very frequently when the parameter is very large.

From the above figures, we can conclude that our scheme provides a steady performance improvement over direct applications of single-server caching for different cache sizes.



(a)

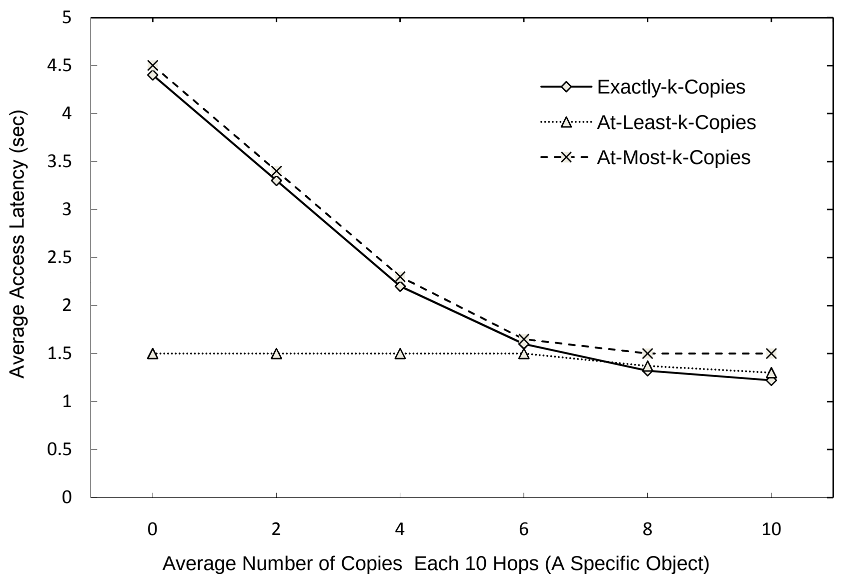


(b)

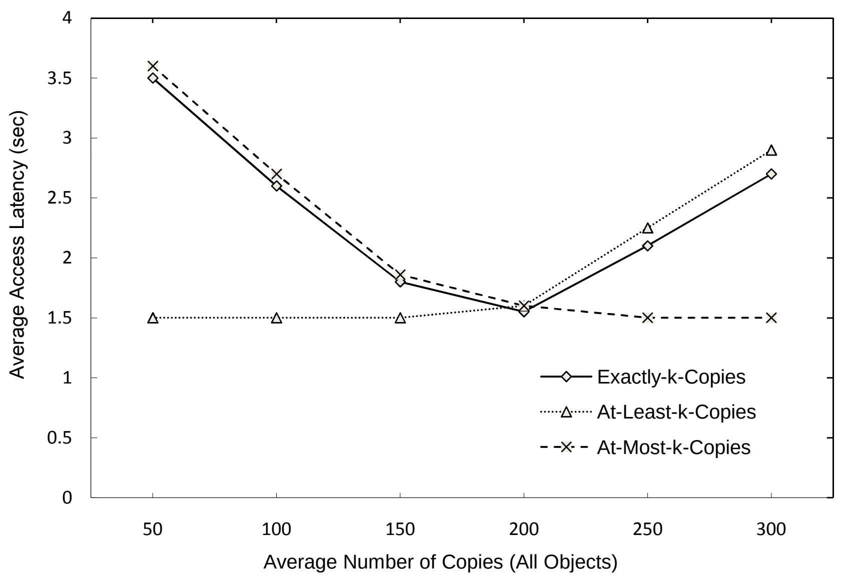
Figure 5.6: Impact of Access Pattern

5.6.2.3 Impact of the number of cached copies

In this group of experiments, we evaluate the QoS performance of our constrained schemes wrt different numbers of cached copies of object(s). To facilitate measurement, we assume all copies are distributed uniformly in the network. Figure 5.7 shows the relation between the average access latency and the number of copies, where three types



(a)



(b)

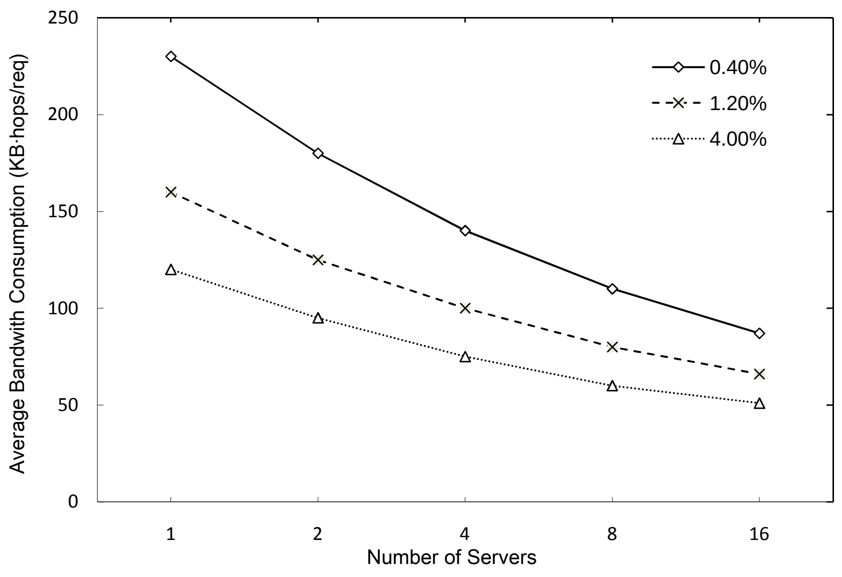
Figure 5.7: Impact of the Number of Copies

of constraints are compared. The experiments in this group are made with two servers and 0.4% relative cache size.

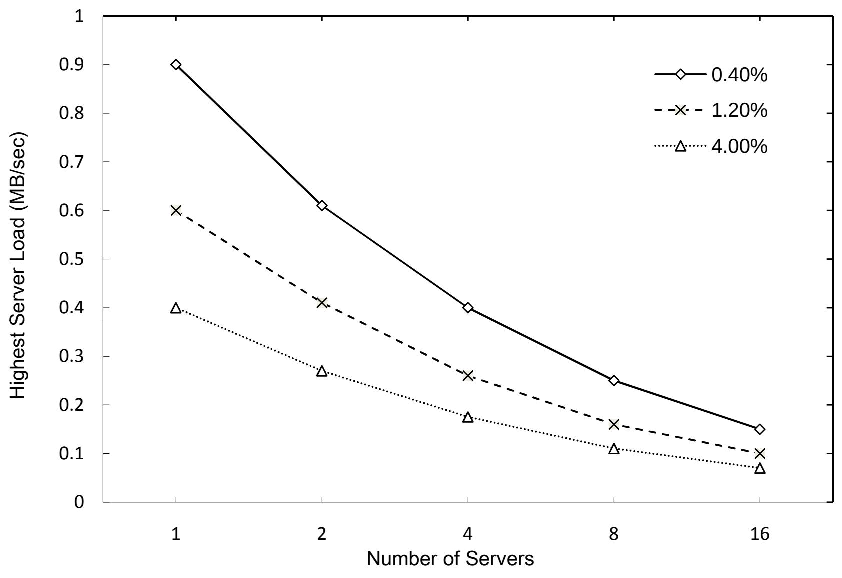
Figure 5.7(a) shows individual scheme's latency for different numbers of copies within 10 hops from which it is easy to see that the performances of all schemes increase steadily with the increase of the number of copies. However, if the number of copies of all objects increases continually, system performance fluctuates at some degree as shown in Figure 5.7(b). The reason of this phenomenon is that system performance is also affected by the speed of content exchange in caches when the average number of copies of all objects increases continually. In this case, we find that the constraint *at-most-k-copie*s is very useful to avoid system unsteadiness. As we can see from Figure 5.7, the matrices corresponding to *exactly-k-copies* reaches an extreme value when the average number of copies is set around *200*. This observation indicates that the optimal number of copies in the unconstrained case may be a value in this region. Therefore, the system can work in a good condition when fewer than 250 copies are deployed on average. We also note that the performance of the constrained schemes are always inferior to the unconstrained scheme in the same environment, this is due to the computational overhead brought in to the scheme by each constraint.

5.6.2.4 Impact of the number of servers

Our last group of experiments is designed to show the scalability of our schemes and their applicability in networks deploying different numbers of servers. We evaluate the performance of our unconstrained scheme, in the metrics of bandwidth consumption in Figure 5.8(a) and server load in Figure 5.8(b) respectively, wrt deployment of different numbers of servers. The experiments are done for three different settings of cache size: 0.4%, 1.2% and 4%. The results in 5.8 show clearly that in both bandwidth consumption and server load the performance of our scheme improves consistently with the increase of servers deployed. This improvement becomes weaker as the number of deployed servers is greater. This is because a system deploying more servers has the capability of storing an object in more locations and thus receives less benefit from caching the object by our scheme. Another interesting observation we can get from Figure 5.8 is that the increase of servers brings more benefit when the cache size is small than that when the cache size is large. This is due to the functional similarity of caches and servers in providing service to clients. When cache size is large, caches as proxies of servers are more powerful and capable of providing some services which otherwise have to be supplied by servers, making performance improvement by adding more servers less



(a)



(b)

Figure 5.8: Impact of the Number of Servers

significant.

5.7 Concluding Remarks

Deploying multiple servers and caching an object at selective sites are important technologies to improve the efficiency of content delivery and the scalability of network services. Existing single-server caching schemes in general do not work in multi-server systems. In this chapter, we formulated the coordinated en-route caching problem in a multi-server network, which takes into account all requests that pass through the intermediate nodes of a response path, as a system gain maximization problem. Applying dynamic programming techniques, we developed efficient methods for finding optimal solutions to this problem for the unconstrained case and two QoS-constrained cases respectively. For each case, we presented a caching scheme as application of the corresponding method. We evaluated the proposed schemes on different performance metrics through extensive simulation experiments. The experiment results show that these schemes all yield a steady performance improvement and achieve desired QoS in a multi-server network.

Our caching schemes can be directly applied in CDNs and P2P systems to improve system performance and service scalability. Similar to the pioneering work of Tang *et al.* in single-server networks [127], our proposed methods for multi-server networks can also be extended for solving the coordinated en-route web caching problem in different system settings, such as specific network topologies (e.g. tree) and networks with transcoding-enabled proxies.

If caching is performed in non en-route fashion, i.e., the response path may not follow the request path, our proposed caching schemes still work correctly provided all nodes on the response path together with their request frequencies for the object are known at the time of object delivery. This is because the performance of our schemes depends only on the knowledge of request frequency recorded at every node on the response path which can be completely different from the request path. If the response path is not known, which may occur in many practical situations, it is desirable to perform caching in an ad hoc manner that makes decision based on the knowledge of only those nodes already passed as delivery goes. Applying appropriate techniques in approximation and online algorithms, we can make the caching achieve local optimum with a good approximation ratio. Our proposed caching schemes will provide useful knowledge and experience for further developing effective caching methods in this kind of new environments.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

With the proliferation of the Internet, popular Web services often suffer from congestion due to large demands for access. Replicating these services across the network strategically is an effective method to improve performance and achieve scalability. Content replication techniques are critical for the content delivery infrastructure, especially to bandwidth-hungry applications like online video casting, high resolution photo sharing, query based interactions, and multimedia content downloading. Significant work has been done in this area to improve the effectiveness of relevant techniques in content delivery.

This thesis covers three issues in content delivery: replica server (facility) placement, content replication and en-route caching. Traditionally, these problems are studied in operations research, Content Distribution Network and web caching. In this thesis, we combine related work in these fields into a single research theme on efficient content delivery and put focus on algorithms for replica placement of servers and contents. In essence, these issues share a basic objective — to optimize the location of replicas and improve the efficiency of content delivery, which includes selection of sites to deploy replica servers, selection of replica servers to host replica contents, and selection of en-route caches on the delivery path to store contents dynamically. Considering the requirement of stakeholders including high availability of services, scalability of infrastructure and efficiency in content management, conventional techniques face serious challenges to be widely used in the future.

In order to meet aforementioned requirements, we specifically studied three topics in this thesis: replica server placement for fault tolerance, content replication for parallel access and en-route caching in multi-server networks. On the first topic, we focused on the placement of replica servers to achieve fault tolerance capability by defining an optimization problem called *Fault Tolerant Facility Allocation (FTFA)*. The problem is distinct from the well-know *Fault-Tolerant Facility Location (FTFL)* by relaxing the number of facilities that can be deployed at each site. Using this model usually results to cheaper cost in network design than the *FTFL* model because of the relaxation. Due to the NP-hardness of the problem, we propose three polynomial-time algorithms which achieve approximation factor 1.861, 1.61 and 1.52 respectively. These results are based on the existing factor-revealing LPs in the literature, which are better than the best-know approximation factor for *FTFL* (2.076). We also studied a variant of the problem, *Fault Tolerant k-Facility Allocation*, by specifying an upper bound of the total facility numbers that can be deployed and we present a 4-approximation algorithm for the problem.

On the second topic, we studied QoS-aware replication technique for parallel access to the replicated content in the Internet and show that this problem can be modeled as the *Fault-Tolerant Facility Location (FTFL)* problem where facilities to be deployed are digital contents. We propose a distributed algorithm to find potential locations to deploy contents in order to suit the environment where the global knowledge of network status is impractical to obtain. As far as we know, performance guarantee of similar algorithms based on primal-dual schema (including centralized algorithms) for *FTFL* remains unknown except a special case in which all cities have a uniform connectivity requirement. We provide an upper bound of our solution which is between *2* and *R* times of the optimal solution in the nonuniform case, where *R* is the maximum number of parallel connections. Extensive numerical experiments show that the cost of our solutions is comparable (within 4% error) to the optimal solutions.

On the third topic, we studied the problem of en-route web caching problem in a multi-server network. The problem differs from the single-server version as it needs to consider all requests (to any server for an object) which pass through the intermediate nodes on a request/response path. We derive efficient dynamic programming based methods for finding optimal locations which maximize the system's total gain by deploying objects among these locations. Different from the first two problems, here the objects are deployed dynamically when a response is forwarded to the client, and coordinately by integrating both object placement and replacement policies together. The problem is considered in the unconstrained case and two QoS-constrained cases respectively.

In conclusion, this thesis established new models for replica placement concerning scalability, reliability and efficiency. These models are effective to enhance reliability of existing information communication infrastructures and availability of content delivery service through server replication, as well as scalability and efficiency via content replication for parallel access and caching in multi-server networks. Both experiments and theoretical analysis show that the proposed methods for these models are crucial for designing an efficient content delivery system.

6.2 Future Work – Towards Smart Content Delivery

As stated in the first chapter, this thesis focuses on the replica placement algorithms which bring significant solutions to part of those issues in content delivery. Other important issues in content delivery include content consistency enforcement, data security, and system architecture etc. Future work may involve proposing novel techniques for these issues, and improving the techniques for addressing the issues mentioned in this thesis as well. One possible direction in the future work is to integrate existing techniques in such as architectures, security and data mining and design a state-of-the-art application framework for smart content delivery.

6.2.1 Novel Architectures

Due to the development of P2P networks, the concept of decentralized architecture is deeply rooted in practice for the benefit of high scalability. However, the drawbacks of existing P2P systems impede the application in commercial environment, which notably include the potential piracy of intellectual property. Essentially, this issue is due to the lack of characterization of roles for participants such as content providers and content consumers. Fortunately, these drawbacks are avoidable using a deliberate combination of existing techniques which aimed at a state-of-the-art integrated framework for content delivery. Architecture is the first consideration in designing such a framework in the future. A desired architecture maintains the excellence of existing systems and at the same time exploits the capability and benefits of other techniques, like P2P networking for fault-tolerance and inherent reliability. As an example, a server-side overlay network using DHT (Distributed Hash Table), which is capable of dealing with system's dynamics and also coping with client-side access, is a point for further advance in the study.

6.2.2 Security Considerations

Existing mechanisms for content outsourcing include: cooperative push-based approach, uncooperative pull-based approach, and cooperative pull-based approaches [105, 89]. These approaches mainly focus on the efficiency of content delivery while consider little on data security. However, security mechanisms are indispensable for a content delivery system to protect participants including both content consumers and providers.

On the other hand, digital rights management (DRM) technologies have spawned widespread controversy [42]. The collection of information about access and use of creative works raises obvious privacy concerns. As such, appropriate approaches for protecting content providers under specific privacy policy are one of the major concerns of current research in content delivery.

6.2.3 Employment of Data Mining Techniques

Content delivery involves many issues and difficulties in large-scale data management [24] and data mining provide an effective way of dealing with these difficulties. Researchers have exploited data mining in solutions for service pricing and content outsourcing. In fact, various data mining techniques such as clustering based on similarities of contents, links and models may be used to facilitate content outsourcing. As pointed out by Pallis and Vakali [89], clustering of pages is essential in the optimization of content outsourcing which aims to address the content selection problem by clustering content for outsourcing. Clustering of users is another possible practice in sketching pricing strategies which involves defining clusters of users in order to facilitate content personalization and differentiated services. Theoretical models like Bayesian networks or Markov models can be utilized here for classifying users over clusters.

Bibliography

[1] Bharat B. Bhargava Abdelsalam A. Helal, Abdelsalam A. Heddaya. *Replication Techniques In Distributed Systems*. Kluwer Academic Publishers, 1996.

[2] Marc Abrams, Charles R. Standridge, Ghaleb Abdulla, Edward A. Fox, and Stephen Williams. Removal policies in network caches for World-Wide Web documents. *SIGCOMM Comput. Commun. Rev.*, 26(4):293–305, 1996.

[3] Marc Abrams, Charles R. Standridge, Ghaleb Abdulla, Stephen Williams, and Edward A. Fox. Caching proxies: Limitations and potentials. Technical report, Blacksburg, VA, USA, 1995.

[4] Charu Aggarwal, Joel L. Wolf, and Philip S. Yu. Caching on the World Wide Web. *IEEE Trans. on Knowl. and Data Eng.*, 11(1):94–107, 1999.

[5] Jussara Almeida, Virgilio Almeida, and David Yates. Measuring the behavior of a World-Wide Web server. Technical report, Boston, MA, USA, 1996.

[6] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristic for k-median and facility location problems. In *STOC '01: Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 21–29, New York, NY, USA, 2001. ACM.

[7] Awerbuch, Bartal, and Fiat. Distributed paging for general networks. In *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*, 1996.

[8] Ivan D. Baev and Rajmohan Rajaraman. Approximation algorithms for data placement in arbitrary networks. In *SODA '01: Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 661–670, Philadelphia, PA, USA, 2001. Society for Industrial and Applied Mathematics.

[9] Paul Barford and Mark Crovella. Generating representative web workloads for network and server performance evaluation. In *Measurement and Modeling of Computer Systems*, pages 151–160, 1998.

[10] Abdelkrim Beloued, Jean-Marie Gilliot, Maria-Teresa Segarra, and Françoise André. Dynamic data replication and consistency in mobile environments. In *DSM '05: Proceedings of the 2nd international doctoral symposium on Middleware*, pages 1–5, New York, NY, USA, 2005. ACM.

[11] Samrat Bhattacharjee, Kenneth L. Calvert, and Ellen W. Zegura. Self-organizing wide-area network caches. In *Proceedings of IEEE INFOCOM '98*, pages 600–608, 1998.

[12] Jean-Chrysostome Bolot and Philipp Hoschka. Performance engineering of the World Wide Web: application to dimensioning and cache design. In *Proceedings of the fifth international World Wide Web conference on Computer networks and ISDN systems*, pages 1397–1405, Amsterdam, The Netherlands, The Netherlands, 1996. Elsevier Science Publishers B. V.

[13] André B. Bondi. Characteristics of scalability and their impact on performance. In *WOSP '00: Proceedings of the 2nd international workshop on Software and performance*, pages 195–203, New York, NY, USA, 2000. ACM.

[14] Jürgen Branke, Pablo Funes, and Frederik Thiele. Evolutionary design of en-route caching strategies. *Appl. Soft Comput.*, 7(3):890–898, 2007.

[15] Lee Breslau, Pei Cue, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web caching and Zipf-like distributions: Evidence and implications. In *Proceedings of IEEE INFOCOM '99*, pages 126–134, 1999.

[16] J. Byers, J. Considine, M. Mitzenmacher, and S. Rost. Informed content delivery across adaptive overlay networks. In *SIGCOMM '02*, 2002.

[17] J.W. Byers, M. Luby, and M. Mitzenmacher. Accessing multiple mirror sites in parallel: using tornado codes to speed up downloads. *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 1:275–283 vol.1, Mar 1999.

[18] Jaroslaw Byrka. An optimal bifactor approximation algorithm for the metric uncapacited facility location problem. In *APPROX and RANDOM '07*, 2007.

[19] K. L. Calvert, M. B. Doar, and E. W. Zegura. Modelling internet topology. *IEEE Comm. Magazine*, 35(6):160–163, 1997.

[20] Anawat Chankhunthod, Peter B. Danzig, Chuck Neerdaels, Michael F. Schwartz, and Kurt J. Worrell. A hierarchical internet object cache. In *USENIX Annual Technical Conference*, pages 153–164, 1996.

[21] Moses Charikar and Sudipto Guha. Improved combinatorial algorithms for the facility location and k-median problems. In *FOCS '99: Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, page 378, Washington, DC, USA, 1999. IEEE Computer Society.

[22] Moses Charikar and Sudipto Guha. Improved combinatorial algorithms for facility location problems. *SIAM J. Comput.*, 34(4):803–824, 2005.

[23] Moses Charikar, Sudipto Guha, Eva Tardos, and David B. Shmoys. A constant-factor approximation algorithm for the k-median problem. In *ACM Symposium on Theory of Computing*, pages 1–10, 1999.

[24] Yan Chen, Lili Qiu, Weiyu Chen, Luan Nguyen, and R.H. Katz. Efficient and adaptive web replication using content clustering. *Selected Areas in Communications, IEEE Journal on*, 21(6):979–994, Aug. 2003.

[25] Ludmila Cherkasova and Jangwon Lee. FastReplica: efficient large file distribution within content delivery networks. In *USITS'03: Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems*, pages 7–7, Berkeley, CA, USA, 2003. USENIX Association.

[26] Maureen Chesire, Alec Wolman, Geoffrey M. Voelker, and Henry M. Levy. Measurement and analysis of a streaming-media workload. In *USITS'01: Proceedings of the 3rd conference on USENIX Symposium on Internet Technologies and Systems*, pages 1–1, Berkeley, CA, USA, 2001. USENIX Association.

[27] D. CHEUNG, B. KAO, and J. LEE. Discovering user access patterns on the World Wide Web. In *Proceedings of the 1st Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'97)*, Feb. 1997.

[28] Fabián A. Chudak and David B. Shmoys. Improved approximation algorithms for the uncapacitated facility location problem. *SIAM J. Comput.*, 33(1):1–25, 2004.

[29] Fabián A. Chudak and David P. Williamson. Integer programming and combinatorial optimization. In *Improved Approximation Algorithms for Capacitated Facility Location Problems*, Lecture Notes in Computer Science, pages 99–113. Springer Berlin / Heidelberg, 1999.

[30] Edith Cohen, Balachander Krishnamurthy, and Jennifer Rexford. Evaluating server-assisted cache replacement in the web. In *ESA '98: Proceedings of the 6th Annual European Symposium on Algorithms*, pages 307–319, London, UK, 1998. Springer-Verlag.

[31] Edith Cohen, Balachander Krishnamurthy, and Jennifer Rexford. Improving end-to-end performance of the web using server volumes and proxy filters. In *SIGCOMM '98*, pages 241–253, 1998.

[32] Jupiter Communications. Internet caching resource center. Online. Available: http://www.caching.com/.

[33] I. Cooper, I. Melve, and G. Tomlinson. Internet web replication and caching taxonomy, 2001.

[34] Gerard Cornuejols, Marshall L. Fisher, and George L. Nemhauser. Location of bank accounts to optimize float: An analytic study of exact and approximate algorithms. *Management Science*, 23(8):789–810, 1977.

[35] Michael Dahlin, Randolph Wang, Thomas E. Anderson, and David A. Patterson. Cooperative caching: Using remote client memory to improve file system performance. In *Operating Systems Design and Implementation*, pages 267–280, 1994.

[36] Peter B. Danzig, Richard S. Hall, and Michael F. Schwartz. A case for caching file objects inside internetworks. In *SIGCOMM '93*, pages 239–248, 1993.

[37] B. D. Davison. Proxy cache comparison. Online. Available: http://www.web-caching.com/proxy-comparison.html.

[38] Brian D. Davison. A Web caching primer. *IEEE Internet Computing*, 5(4):38–45, July/August 2001.

[39] J. DILLEY, B. MAGGS, J. PARIKH, H. PROKOP, R. SITARAMAN, and B. WEIHL. Globally distributed content delivery. *IEEE Internet Computing*, 6(5):50–58, 2002.

[40] Fred Douglis, Anja Feldmann, Balachander Krishnamurthy, and Jeffrey Mogul. Rate of change and other metrics: a live study of the World Wide Web. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems on USENIX Symposium on Internet Technologies and Systems*, pages 14–14, Berkeley, CA, USA, 1997. USENIX Association.

[41] Lawrence W. Dowdy and Derrell V. Foster. Comparative models of the file assignment problem. *ACM Comput. Surv.*, 14(2):287–313, 1982.

[42] Natali Helberger (ed.). Digital rights management and consumer acceptability: A multi-disciplinary discussion of consumer concerns and expectations. Technical report, INDICARE, http://www.ivir.nl/publications/helberger/INDICAREStateoftheArtReport.pdf, 2004.

[43] Li Fan, Pei Cao, Jussara Almeida, and Andrei Z. Broder. Summary cache: a scalable wide-area Web cache sharing protocol. *IEEE/ACM Transactions on Networking*, 8(3)(3):281–293, 2000.

[44] Christian Frank and Kay Römer. Distributed facility location algorithms for flexible configuration of wireless sensor networks. In *Proceedings of the 3rd IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS 2007)*, Santa Fe, NM, USA, June 2007.

[45] Samrat Ganguly, Akhilesh Saxena, Sudeept Bhatnagar, Suman Banerjee, and Rauf Izmailov. Fast replication in content distribution overlays. In *Proceedings of IEEE INFOCOM '05*, 2005.

[46] Joachim Gehweiler, Christiane Lammersen, and Christian Sohler. A distributed $o(1)$-approximation algorithm for the uniform facility location problem. In *SPAA '06: Proceedings of the eighteenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 237–243, New York, NY, USA, 2006. ACM.

[47] Sudipto Guha, Adam Meyerson, and Kamesh Munagala. Improved algorithms for fault tolerant facility location. In *SODA '01: Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 636–641, Philadelphia, PA, USA, 2001. Society for Industrial and Applied Mathematics.

[48] Sudipto Guha, Adam Meyerson, and Kamesh Munagala. A constant factor approximation algorithm for the fault-tolerant facility location problem. *J. Algorithms*, 48(2):429–440, 2003.

[49] Dorit S. Hochbaum. Heuristics for the fixed cost median problem. *Mathematical Programming*, 22(1):148–162, 1982.

[50] David J. Dewitt Hui-I Hsiao. A performance study of three high availability data replication strategies. *Distributed and Parallel Databases*, 1(1):53–79, 1993.

[51] S HULL. *Content Delivery Networks*. McGraw-Hill, New York, 2002.

[52] Kamal Jain, Mohammad Mahdian, Evangelos Markakis, Amin Saberi, and Vijay V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *Journal of the ACM*, 50(6):795–824, 2003.

[53] Kamal Jain, Mohammad Mahdian, and Amin Saberi. A new greedy approach for facility location problems. In *STOC '02: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 731–740, New York, NY, USA, 2002. ACM.

[54] Kamal Jain and Vijay V. Vazirani. Primal-dual approximation algorithms for metric facility location and k-median problems. In *IEEE Symposium on Foundations of Computer Science*, pages 2–13, 1999.

[55] Kamal Jain and Vijay V. Vazirani. An approximation algorithm for the fault tolerant metric facility location problem. In *APPROX '00: Proceedings of the Third International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 177–183, London, UK, 2000. Springer-Verlag.

[56] Kamal Jain and Vijay V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation. *Journal of the ACM*, 48(2):274–296, 2001.

[57] S. Jamin, Cheng Jin, Yixin Jin, D. Raz, Y. Shavitt, and Lixia Zhang. On the placement of internet instrumentation. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 295–304 vol.1, 2000.

[58] Jimmy Jernberg, Vladimir Vlassov, Ali Ghodsi, and Seif Haridi. DOH: A content delivery Peer-to-Peer network. In *Proceedings of European Conference on Parallel Computing, (EUROPAR '06)*, page 13, Dresden, Germany, 2006.

[59] Anxiao (Andrew) Jiang and Jehoshua Bruck. Optimal content placement for en-route web caching. In *Proc. the 2nd IEEE International Symposium on Network Computing and Applications*, page 9, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

[60] Shudong Jin and Limin Wang. Content and service replication strategies in multihop wireless mesh networks. In *MSWiM '05: Proceedings of the 8th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 79–86, New York, NY, USA, 2005. ACM.

[61] Jussi Kangasharju, James Roberts, and Keith W. Ross. Object replication strategies in content distribution networks. *Computer Communications*, 25(4):376 – 383, 2002.

[62] S.U. Khan and I. Ahmad. A powerful direct mechanism for optimal WWW content replication. pages 86–86, April 2005.

[63] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman. Placement algorithms for hierarchical coorperative caching. In *Proceedings of 10th Ann. ACM-SIAM Symp. Discrete Algorithms*, pages 586–595, 1999.

[64] Madhukar R. Korupolu and Michael Dahlin. Coordinated placement and replacement for large-scale distributed caches. volume 14, pages 1317–1329, Piscataway, NJ, USA, 2002. IEEE Educational Activities Department.

[65] Balachander Krishnamurthy and Craig E. Wills. Piggyback server invalidation for proxy cache coherency. In *WWW7: Proceedings of the seventh international conference on World Wide Web 7*, pages 185–193, Amsterdam, The Netherlands, The Netherlands, 1998. Elsevier Science Publishers B. V.

[66] P. Krishnan, Danny Raz, and Yuval Shavitt. The cache location problem. *IEEE/ ACM Transactions on Networking*, 8(5):568–582, 2000.

[67] P. Krishnan and Binay Sugla. Utility of co-operating Web proxy caches. *Computer Networks and ISDN Systems*, 30(1–7):195–203, 1998.

[68] C. Labovitz, G. R. Malan, and F. Jahania. Internet routing instability. In *SIGCOMM' 97*, Aug. 1997.

[69] A. Leff, J. L. Wolf, and P. S. Yu. Replication algorithms in a remote caching architecture. *IEEE Trans. Parallel Distrib. Syst.*, 4(11):1185–1204, 1993.

[70] Bo Li, Xin Deng, Mordecai J. Golin, and Kazem Sohraby. On the optimal placement of Web proxies in the Internet: The linear topology. In *HPN '98: Proceedings of the IFIP TC-6 Eigth International Conference on High Performance Networking*, pages 485–495, Deventer, The Netherlands, The Netherlands, 1998. Kluwer, B.V.

[71] Keqiu LI and Hong SHEN. Optimal methods for proxy placement in coordinated en-route web caching. *IEICE Transactions on Communications*, E88-B(4):1458–1466, 2005.

[72] Keqiu Li, Hong Shen, Francis Y. L. Chin, and Si Qing Zheng. Optimal methods for Coordinated En-Route Web Caching for tree networks. *ACM Trans. Inter. Tech.*, 5(3):480–507, 2005.

[73] Keqiu Li, Hong Shen, Francis Y.L. Chin, and Weishi Zhang. Multimedia object placement for transparent data replication. *IEEE Transactions on Parallel and Distributed Systems*, 18(2):212–224, 2007.

[74] Wei-Cherng Liao, Fragkiskos Papadopoulos, and Konstantinos Psounis. Performance analysis of BitTorrent-like systems with heterogeneous users. *Perform. Eval.*, 64(9-12):876–891, 2007.

[75] Hwa-Chun Lin and Kuen-Feng Yang. Placement of repair servers to support server-based reliable multicast. In *Communications, 2001. ICC 2001. IEEE International Conference on*, volume 6, pages 1802–1806 vol.6, 2001.

[76] Jyh-Han Lin and Jeffrey Scott Vitter. e-approximations with minimum packing constraint violation. In *STOC '92: Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 771–782, New York, NY, USA, 1992. ACM.

[77] Zhen Liu, Kai Zheng, and Bin Liu. Hybrid cache architecture for high speed packet processing. In *HOTI '05: Proceedings of the 13th Symposium on High Performance Interconnects*, pages 67–72, Washington, DC, USA, 2005. IEEE Computer Society.

[78] T. Loukopoulos, I. Ahmad, and D. Papadias. An overview of data replication on the internet. pages 27–32, 2002.

[79] Ari Luotonen and Kevin Altis. World-wide web proxies. *Comput. Netw. ISDN Syst.*, 27(2):147–154, 1994.

[80] Mohammad Mahdian, Yinyu Ye, and Jiawei Zhang. A 1.52-approximation algorithm for the uncapacitated facility location problem. In *Proceedings of APPROX '02 LNCS 2462*, pages 229–242, 2002. manuscript.

[81] Mohammad Mahdian, Yinyu Ye, and Jiawei Zhang. Approximation algorithms for metric facility location problems. *SIAM J. Comput.*, 36(2):411–432, 2006.

[82] Leland R. Beaumont Markus Hofmann. *Content Networking: Architecture, Protocols, and Practice*. Morgan Kaufmann Publisher, 2005.

[83] R. McEliece, E. Rodemich, H. Rumsey, and L. Welch. New upper bounds on the rate of a code via the delsarte-macwilliams inequalities. *Information Theory, IEEE Transactions on*, 23(2):157–166, Mar 1977.

[84] Amin Saberi Vijay Vazirani Mohammad Mahdian, Evangelos Markakis. A greedy facility location algorithm analyzed using dual fitting. In *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques*, pages 127–137, 2001.

[85] Thomas Moscibroda and Roger Wattenhofer. Facility location: distributed approximation. In *PODC '05: Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*, pages 108–117, New York, NY, USA, 2005. ACM.

[86] Rajeev Motwani. Lecture notes on approximation algorithms: Volume i. Technical report, Stanford, CA, USA, 1993.

[87] Padmavathi Mundur and Poorva Arankalle. Optimal server allocations for streaming multimedia applications on the Internet. *Comput. Networks*, 50(18):3608–3621, 2006.

[88] Venkata N. Padmanabhan and Lili Qiu. The content and access dynamics of a busy web site: findings and implications. *SIGCOMM Comput. Commun. Rev.*, 30(4):111–123, 2000.

[89] George Pallis and Athena Vakali. Insight and perspectives for content delivery networks. *Commun. ACM*, 49(1):101–106, 2006.

[90] R. Panigrahy. Relieving hot spots on the world wide web. Technical report, Cambridge, MA, USA, 1997.

[91] V. Paxson. End-to-end routing behavior in the Internet. *IEEE/ACM Transactions on Networking*, 6:601–615, Oct. 1997.

[92] S. Philopoulos and M. Maheswaran. Experimental study of parallel downloading schemes for Internet mirror sites. In *13th IASTED International Conference on Parallel and Distributed Computing Systems (PDCS '01)*, pages 44–48, 2001.

[93] Guillaume Pierre and Maarten van Steen. Design and implementation of a user-centered content distribution network. In *WIAPP '03: Proceedings of the 3rd IEEE Workshop on Internet Applications*, page 42, Washington, DC, USA, 2003. IEEE Computer Society.

[94] Richard L. Francis Pitu B. Mirchandani, editor. *Discrete Location Theory*. John Wiley, New York., 1990.

[95] Stefan Podlipnig and Laszlo Böszörmenyi. A survey of Web cache replacement strategies. *ACM Comput. Surv.*, 35(4):374–398, 2003.

[96] D. Povey and J. Harrison. A distributed Internet cache. In *Proceedings 20th Australian Computer Science Conf.,*, Sydney, Australia, Feb. 1997.

[97] Dongyu Qiu and R. Srikant. Modeling and performance analysis of BitTorrent-like peer-to-peer networks. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 367–378, New York, NY, USA, 2004. ACM.

[98] Lili Qiu, Venkata N. Padmanabhan, and Geoffrey M. Voelker. On the placement of Web server replicas. In *Proceedings of IEEE INFOCOM '01*, pages 1587–1596, 2001.

[99] M. Rabinovich and A. Aggarwal. Radar: A scalable architecture for a global Web hosting service. *Computer Network*, 31:1545–1561, 1999.

[100] M. Rabinovich, J. Chase, and S. Gadde. Not all hits are created equal: Cooperative proxy caching over a wide area network. *Computer Networks and ISDN Systems*, 30(22-23):2253–2259, Nov. 1998.

[101] M. Rabinovich and O Spastscheck. *Web Caching and Replication*. Addison-Wesley, MA, 2002.

[102] M. Rabinovich and H. Wang. Dhttp: An efficient and cache-friendly transfer protocol for Web traffic. In *Proceedings of IEEE INFOCOM '01*, pages 1597–1606, April 2001.

[103] Michael Rabinovich. Issues in web content replication. *Data Engineering Bulletin*, 21:21–29, 1998.

[104] Pavlin Radoslavov, Ramesh Govindan, and Deborah Estrin. Topology-informed internet replica placement. *Computer Communications*, 25(4):384 – 392, 2002.

[105] Mukaddim Pathan Rajkumar Buyya and Athena Vakali, editors. *Content Delivery Networks*. Springer, 2008.

[106] P. Rodriguez and W. Ernst Biersack. Dynamic parallel-access to replicated content in the Internet. *IEEE/ACM Transactions on Networking*, 10:455–465, 2002.

[107] P. Rodriguez, S. Sibal, and O. Spatscheck. Tpot: Translucent proxying of TCP. In *Proceedings 5th Int'l Web Caching and Content Delivery Workshop (WCW)*, May 2000.

[108] P. Rodriguez, C. Spanner, and E. Biersack. Analysis of web caching architectures: Hierarchical and distributed caching. *IEEE/ACM Transactions on Networking*, 9:404–418, 2001.

[109] Pablo Rodriguez, Andreas Kirpal, and Ernst Biersack. Parallel-access for mirror sites in the internet. In *Proceedings of IEEE INFOCOM '00*, pages 864–873, 2000.

[110] Pablo Rodriguez and Sandeep Sibal. SPREAD: Scalable platform for reliable and efficient automated distribution. *Computer Networks (Amsterdam, Netherlands: 1999)*, 33(1–6):33–49, 2000.

[111] Dan Rubenstein and Sambit Sahu. Can unstructured p2p protocols survive flash crowds? *IEEE/ACM Trans. Netw.*, 13(3):501–512, 2005.

[112] Guha S. and Khuller S. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms*, 31:228–248(21), April 1999.

[113] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Trans. Comput. Syst.*, 2(4):277–288, 1984.

[114] P. Scheuermann, J. Shim, and R. Vingralek. A case for delay conscious caching of Web documents. *Computer Networks and ISDN Systems*, 29(8-13):997–1005, 1997.

[115] Peter Scheuermann, Junho Shim, and Radek Vingralek. A case for delay-conscious caching of Web documents. *Comput. Netw. ISDN Syst.*, 29(8-13):997–1005, 1997.

[116] R. Schollmeier. A definition of Peer-to-Peer networking for the classification of peer-to-peer architectures and applications. *Peer-to-Peer Computing, IEEE International Conference on*, 0:0101, 2001.

[117] Hong Shen and Shihong Xu. Coordinated En-Route Web Caching in multiserver networks. *IEEE Transactions on Computers*, 58(5):605–619, May 2009.

[118] Junho Shim, Peter Scheuermann, and Radek Vingralek. Proxy cache algorithms: Design, implementation, and performance. *Knowledge and Data Engineering*, 11(4):549–562, 1999.

[119] David B. Shmoys, Eva Tardos, and Karen Aardal. Approximation algorithms for facility location problems. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 265–274, 1997.

[120] S. Sivasubramanian, M. Szymaniak, G. Pierre, and M. Steen. Replication for Web hosting systems. *ACM Computing Surveys*, 36:291–334, 2004.

[121] Swaminathan Sivasubramanian. Adaptive replication for web applications. In *DSM '04: Proceedings of the 1st international doctoral symposium on Middleware*, pages 290–293, New York, NY, USA, 2004. ACM.

[122] Konstantinos Stamos, George Pallis, Charilaos Thomos, and Athena Vakali. A similarity based approach for integrated web caching and content replication in CDNs. In *IDEAS '06: Proceedings of the 10th International Database Engineering and Applications Symposium*, pages 239–242, Washington, DC, USA, 2006. IEEE Computer Society.

[123] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, 2001.

[124] Maxim Sviridenko. An improved approximation algorithm for the metric uncapacitated facility location problem. In *Proceedings of the 9th International IPCO Conference on Integer Programming and Combinatorial Optimization*, pages 240–257, London, UK, 2002. Springer-Verlag.

[125] Chaitanya Swamy and David B. Shmoys. Fault-tolerant facility location. *ACM Trans. Algorithms*, 4(4):1–27, 2008.

[126] Member-Xueyan Tang and Member-Jianliang Xu. QoS-aware replica placement for content distribution. *IEEE Trans. Parallel Distrib. Syst.*, 16(10):921–932, 2005.

[127] Xueyan Tang and Samuel T. Chanson. Coordinated En-Route Web Caching. *IEEE Trans. Comput.*, 51(6):595–607, 2002.

[128] Xueyan Tang and S.T. Chanson. Coordinated management of cascaded caches for efficient content distribution. pages 37–48, March 2003.

[129] Xueyan Tang, Huicheng Chi, and S.T. Chanson. Optimal replica placement under TTL-based consistency. *Transactions on Parallel and Distributed Systems*, 18(3):351–363, March 2007.

[130] D.L. Tennenhouse, J.M. Smith, W.D. Sincoskie, D.J. Wetherall, and G.J. Minden. A survey of active network research. *IEEE Comm. Magazine*, 35(1):80–86, Jan. 1997.

[131] Renu Tewari, Michael Dahlin, Harrick M. Vin, and Jonathan S. Kay. Design considerations for distributed caching on the Internet. In *International Conference on Distributed Computing Systems*, pages 273–284, 1999.

[132] Michael A. Trick. A tutorial on dynamic programming. Online:http://mat.gsia.cmu.edu/classes/dynamic/dynamic.html, June 1997.

[133] Vijay V. Vazirani. *Approximation Algorithms*. Springer-Verlag, Berlin, 2001.

[134] S. Venkataraman, J.F. Naughton, and M. Livny. Remote load-sensitive caching for multi-server database systems. In *14th International Conference on Data Engineering (ICDE '98)*, 1998.

[135] Jia Wang. A survey of web caching schemes for the Internet. *SIGCOMM Comput. Commun. Rev.*, 29(5):36–46, 1999.

[136] D. Wessels and K. Claffy. ICP and the Squid Web cache. *IEEE J. Selected Areas in Comm*, 16(3):345–357, April 1998.

[137] Ouri Wolfson, Sushil Jajodia, and Yixiu Huang. An adaptive data replication algorithm. *ACM Trans. Database Syst.*, 22(2):255–314, 1997.

[138] Roland P. Wooster and Marc Abrams. Proxy caching that estimates page load delays. In *Selected papers from the sixth international conference on World Wide Web*, pages 977–986, Essex, UK, 1997. Elsevier Science Publishers Ltd.

[139] Lin Wujuan and Bharadwaj Veeravalli. Design and analysis of an adaptive object replication algorithm in distributed network systems. *Comput. Commun.*, 31(10):2005–2015, 2008.

[140] X. Xiao and L. M. Ni. Internet QoS: a big picture. *Network, IEEE*, 13(2)(2):8–18, 1999.

[141] Shihong Xu and Hong Shen. Fault tolerant facility allocation. manuscript.

[142] Shihong Xu and Hong Shen. The fault tolerant facility allocation problem. manuscript.

[143] Shihong Xu and Hong Shen. Improved algorithms for fault-tolerant facility allocation by inverse dual fitting. manuscript.

[144] Shihong Xu and Hong Shen. An $o(nh)$ algorithm for dual-server Coordinated En-Route Web Caching. In *The 7th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT 2006)*, pages 399–404, Taipei, Taiwan, 2006.

[145] Shihong Xu and Hong Shen. An efficient method for p-server Coordinated En-Route Web Caching. In *The 8th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT 2007)*, pages 113–117, Adelaide, Australia, 2007.

[146] Shihong Xu and Hong Shen. QoS-oriented content delivery in e-learning systems. In *Information Technology in Medicine and Education, IEEE International Symposium on (ITME 2009)*, accepted, Jinan, China, 2009.

[147] Philip S. Yu and Edward A. MacNair. Performance study of a collaborative method for hierarchical caching in proxy servers. *Computer Networks and ISDN Systems*, 30(1-7):215–224, 1998.

[148] Ellen Zegura, Kenneth Calvert, and Samrat Bhattacharjee. How to model an internetwork. In *Proceedings of IEEE INFOCOM '96*, pages 594–602, 1996.

[149] D. Zeng, FeiYue Wang, and Mingkuan Liu. Efficient Web content delivery using proxy caching techniques. *Systems, Man and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 34(3):270–280, Aug. 2004.

Appendix A: List of Symbols

Symbol	*Meaning*
$\mathcal{F}$	set of sites (Chapter 3) or surrogate servers (chapter 4), $\lvert\mathcal{F}\rvert = n_f$.
$\mathcal{C}$	set of clients, $\lvert\mathcal{C}\rvert = n_c$.
f_i	$i \in \mathcal{F}$, facility operating cost (Chapter 3) at site i or storage cost of an object at server i (Chapter 4).
c_{ij}	$i \in \mathcal{F}, j \in \mathcal{C}$, the connection cost between facility i and client j. The cost function forms a metric (satisfy triangle inequality) if the sum of connection costs for two edges of a triangle is not less than the connection cost for the third edge.
r_j	$j \in \mathcal{C}$, connectivity requirement (Chapter 3) or parallel access degree (Chapter 4) of j.
$\mathcal{R}$	set of connectivity requirements of all cities (Chapter 3) or set of parallel access degrees (Chapter 4).
R,	size of $\mathcal{R}$.
d_j	$j \in \mathcal{C}$, demand of city j or access frequency of client j.
x_{ij}	$i \in \mathcal{F}, j \in \mathcal{C}$, number of connections between i and j . There could be multiple connections for a site-client pair in the *FTFA* problem (Chapter 3) but at most one in the *FTFL* problem (Chapter 4).
y_i	$i \in \mathcal{F}$, number of facilities (replicas) at site i. There could be multiple facilities at a site in the *FTFA* problem (Chapter 3) but at most one in the *FTFL* problem (Chapter 4).

Symbol	*Meaning*
x_{ij}^p	$p \in \mathcal{R}$, number of connections between i and j that is set up in phase p.
y_i^p	$p \in \mathcal{R}$, number of replicas at site i that is opened in phase p.
α_j^p	$p \in \mathcal{R}$, total credit paid by client j in phase p.
α_j^p	$p \in \mathcal{R}$, total credit paid by client j in phase p.
β_{ij}^p	$p \in \mathcal{R}$, contribution from client j in phase p to opening a replica at site i.
U	set of *not-fully-connected* cities.
$\lambda_{\mathcal{I}}$	maximum cost ratio (single-factor approximation) with respect to any possible star in the instance $\mathcal{I}$ of the problem.
λ_f	factor of approximation ratio regarding facility cost in *FTFA* or *FTFL* (the first item in terms of bi-factor approximation).
λ_c	factor of approximation ratio regarding connection cost in *FTFA* or *FTFL* (the second item in terms of bi-factor approximation).
λ_p	factor of approximation ratio regarding the *p-th* item in an optimization problem with totally *k* items in the objective function, $1 \leq p \leq k$.
SOL_P	primal solution to the *FTFA* or *FTFL* problem.
SOL_D	dual solution to the *FTFA* or *FTFL* problem.
OPT_1	optimal solution to the original *FTFA* or *FTFL* problem.
F^*	facility cost in OPT_1.
C^*	connection cost in OPT_1.
OPT_2	optimal solution to the new composed problem with the coefficients of the *p-th* item scaled λ_p times.
s	a star composed of a facility and a group of cities connected with the facility.
$\mathcal{S}$	set of all possible stars.
$\mathcal{S}^*$	set of stars in an optimal solution.
c_s^-	connection cost of star s
c_s	cost of star s, $c_s = c_s^- + f_i$, where i is the regarding facility in star s.

Symbol	*Meaning*
x_s	binary variable indicating whether star s is selected.
z	offset of facility cost, i. e., cost for operating facility i is set to be $f_i + z$.
k	input of the problem, which is an upper bound of the number of all open replicas.
$\boldsymbol{x}^1$	$\{x_s^1, s \in \mathcal{S}\}$, solution of *FTFA* when a facility cost is set to be $f_i + z_1$.
A	set of replicas opened in solution $\boldsymbol{x}^1$, $\|A\| = k_1$, $k_1 < k$.
$\boldsymbol{x}^2$	$\{x_s^2, s \in \mathcal{S}\}$, solution of *FTFA* when a facility cost is set to be $f_i + z_2$.
B	set of replicas opened in solution $\boldsymbol{x}^2$, $\|B\| = k_2$, $k < k_2$.
a	coefficient of $\boldsymbol{x}^1$ in a combined solution, i. e. $\boldsymbol{x} = a\boldsymbol{x}^1 + b\boldsymbol{x}^2$, $a = (k_2 - k)/(k_2 - k_1)$.
b	coefficient of $\boldsymbol{x}^2$ in a combined solution, i. e. $\boldsymbol{x} = a\boldsymbol{x}^1 + b\boldsymbol{x}^2$, $b = (k - k_1)/(k_2 - k_1)$.
Δt	constant denoting the delay in the LAN centered at a representative client.
$size$	size of an object.
t_{ij}	time required to transmit the whole object between i and j.
b_{ij}	download speed of client j obtained from server i.
P	subset of surrogate servers to hold replicas, $P \subseteq \mathcal{F}$.
P_j	set of r_j distinct servers which are nearest from j among all servers holding the object.
$\mathcal{A}$	indices denoting the set of nodes passed by a request, i. e. $\{1, 2, 3...n\}$.
$m_s(O, x)$	miss penalty of object O at node $x \in \mathcal{A}$, with respect to server s.
$UE_s(x)$	set of links from x to the nearest upstream node holding a copy of object O toward server s.
$DV_s(x)$	set of links from x to the nearest downstream node holding a copy of object O toward server s.
$C(O, e)$	additional cost of delivering object O over link e.

Symbol	*Meaning*
$f_s(x)$	access frequency to server s observed at node x.
$f'_s(x_l, x, x_r)$	a fraction of the observed frequency at node x which will be satisfied by node x_l or x_r.
$b(x_l, x, x_r, S)$	caching benefit at node x with respect to all servers in S.
$l(x, S)$	caching loss at node x with respect to all servers in S.
$G(P, S)$	net gain of caching the object in set P with respect to all servers in S.
$\mathcal{A}_x$	the first x elements in $\mathcal{A}$, i. e. $(1, 2, \ldots, x)$.
$P_x(x_r)$	part of optimal solution which falls in $\mathcal{A}_x$ when $x \in P^*$ with x_r as the next node in the solution.
$G_x(x_r)$	total gain of copies in $P_x(x_r)$.
$P_{\overline{x}}(x_r)$	part of optimal solution which falls in $\mathcal{A}_x$ when $x \notin P^*$
$G_{\overline{x}}(x_r)$	total gain of copies in $P_{\overline{x}}(x_r)$.

Appendix B: Curriculum Vitae

Personal Information

Name:	Shihong Xu		
Address:	2A Laverack Rd, North Plympton, SA 5037		
Email:	tellshawn@gmail.com		
TEL:	+61(0)8 8303 6744	Mobile:	+61 (0)4 3068 2820

Profile

Research interest: Internet technology, software engineering, algorithm design and analysis, privacy and security, distributed computing and combinatorial optimization.

Experience in commercial software development: worked as a project owner, researcher and employee in the industry; devised and developed diverse solutions to meet business requirements and numerous mission-critical applications.

Certifications

Certified Software Engineer, by Ministry of Information Industry, China, 2000.

Education

PhD in **Computer Science** 2006 – 2009
University of Adelaide, Australia

MEng in **Computer Application Technology** 2002 – 2004
Dalian University of Technology

BEng in **Mechanical Engineering** 1998 – 2002
Dalian University of Technology, China

Publication

Journal papers published

[1] Hong Shen and Shihong Xu. Coordinated En-Route Web Caching in Multi-Server Networks. *IEEE Transactions on Computers*, vol.58, no.5, pp.605-619, May 2009.

[2] Xiqian Chen, Zhongxian Chi, and Shihong Xu. A Spatial Data Warehouse Framework for Efficient Spatial OLAP. *Journal of Dalian University of Technology*, vol.56, no.6, pp.901-904, June 2004. (In Chinese)

[3] Shihong Xu, Hongfei Teng and Zhigang Tian. Metagalaxy Evolving Simulated Algorithm and its Application in Layout for Cabin of Satellite. *Machine Design*, vol. 29, no.7, pp.69-72, July 2001. (In Chinese)

Journal papers under review

[4] Shihong Xu and Hong Shen. Fault Tolerant Facility Allocation. Submitted to *SIAM Journal on Computing*, Aug. 2009.

[5] Shihong Xu and Hong Shen. QoS-Aware Content Replication for Parallel Access in the Internet. Submitted to *IEEE Transactions on Parallel and Distributed Systems*, Aug. 2009.

Conference papers published

[6] Shihong Xu and Hong Shen. A Distributed (R, 2)-Approximation Algorithm for Fault Tolerant Facility Location. In *The 10th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT 2009)*, Hiroshima, Japan, Dec. 8-11, 2009.

[7] Shihong Xu and Hong Shen. The Fault Tolerant Facility Allocation Problem. In *The 20th International Symposium on Algorithms and Computation (ISAAC 2009)*, Hawaii, USA, Dec. 16-18, 2009.

[8] Shihong Xu and Hong Shen. QoS-Oriented Content Delivery in E-Learning Systems. In *The 2nd International Symposium on Information Technology in Medicine and Education (ITME 2009)*, pages 665-670, Jinan, China, Aug. 2009, invited paper.

[9] Shihong Xu and Hong Shen. An Efficient Method for p-Server Coordinated En-Route Web Caching. In *The 8th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT 2007)*, pages113-117, Adelaide, Australia, Dec. 2007.

[10] Shihong Xu and Hong Shen. An O(nh) Algorithm for Dual-Server Coordinated En-Route Web Caching. In *The 7th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT 2006)*, pages 399-404, Taipei, Taiwan, Dec. 2006.

Thesis/Dissertation

PHD DISSERTATION: Replica Placement Algorithms for Efficient Internet Content Delivery

MASTER'S THESIS: The Design and Implementation of an OLAP Oriented WebGIS

BACHELOR'S THESIS: Metagalaxy Evolving Simulated Algorithm and its Application in Layout for Cabin of Satellite

Grant Proposal Writing

During Dec.2007 ~ Feb. 2008, I wrote a research proposal draft for a 3-year project "Smart Digital Content Delivery Network" under the supervision of Prof. Hong Shen, to apply for ARC Discovery Grant.

Teaching Experience

SCHOOL OF COMPUTER SCIENCE

The University of Adelaide, Australia (Part-Time)

Casual Lecturer: Distributed Database and Data Mining

Semester 1, 2008& 2009, under supervision of Prof. Hong Shen.

My responsibilities include introducing the concept and techniques in database design and deployment, presenting advanced technologies in business intelligence including OLAP, data warehouse (snowflake and/or star scheme) and data mining

Teaching Assistant: Distributed and High Performance Computing

Semester 2, 2007& 2008, under supervision of Prof. Hong Shen (2007 and 2008), Dr. Paul Coddington (2007) and Dr. Andrew Wendelborn (2008).

My responsibilities include tutoring students programming on high-performance computing facilities like clusters and supercomputers and guiding student to develop web service based solutions and parallel computing based applications.

Professional Experience

JAPAN ADVANCED INSTITUTE OF SCI & TECH

Kanazawa, Japan (Part-Time), 2005 – 2006

Contract owner of a 5-person-months project aimed to implement a functional prototype of knowledge management system using leading-edge technologies such as .NET Framework (3.5 beta) and SQL Server 2005. *Key achievements:*

◇ Defined project scope, investigated user requirements and analyzed use cases, organized and scheduled the development, and controlled the progress to meet the deadline.

◇ Reviewed test cases to ensure completeness in terms of requirements.

◇ Applied MVC architecture to provide flexibility and extensibility.

◇ Utilized Windows Presentation Foundation to render graphics and corresponding techniques for performance improvement.

◇ Improved scalability in data management through using SQL Server and XML.

◇ Adopted component-based design to reuse inherited codes.

JUSTSYSTEM SOFTWARE CO., LTD.
Dalian, China, 2004 – 2005

Acted as senior programmer, project analyst and solution architect; initiated requirements gathering and business process refining; leaded development team to enhance and upgrade legacy system to meet business requirements, developed SharePoint solutions to facilitate collaboration and improve productivity. *Selected projects include:*

Team leader of a project aimed to develop teaching management information system for Dongbei University of Finances & Economics. I acted as the primary liaison among clients and company, analyzed the project requirements and their feasibility, devised specific solutions to meet the requirements, and made main contribution to develop the system. I also provided technical leadership to junior engineers and mentored team members to accomplish their tasks.

Team leader of a project aimed to design and implement an innovative e-police processing system for a division of Dalian Public Security Agency, China. I worked throughout the SDLC. Key achievements:

◇ Aligned police information and efficient management within the system.

◇ Increased productivity in information gathering by using XML forms which is deliverable via web browser.

◇ Intuitive rendering of spatial information for specific (crime) events like Google maps in a given area.

◇ Facilitated collaboration between remote office sites by using work flow engine.

◇ Improved efficiency by extending information sharing beyond firewall using SharePoint solution.

◇ Concise design benefited from CLR integration in MS SQL Server.

◇ Efficient data manipulation leveraging T-SQL store procedure.

JOINT RESEARCH CENTER OF SPATIAL DATA

Dalian Uni. of Tech., 2002 – 2004

Developed, maintained and upgraded cutting-edge BI integrated Web-GIS software. Coordinate design and implementation of applications. Collaborate with team leader, team members and client representatives to ensure on-time completion of project deliverables, deployed GIS system in multiple sites. Selected projects include: Team leader of research project: Design an OLAP Oriented WebGIS. *Key accomplishments:*

◇ A State-of-the-art architecture with high scalability and flexibility for location service, designed server-side components and store procedure that reduced server load significantly.

◇ Interface oriented design to facilitate collaboration.

◇ Integrated BI component leveraging Analysis Service in SQL Server to support decision making.

◇ Well authored documents since system analysis and design to detailed implementation. Core member of Digital Dalian Project and my responsibilities include:

◇ Maintain, upgrade and code refactoring an inherited GIS system.

◇ Provide consultation to customers on ETL processes

◇ Make contribution to the Schema for Dalian Spatial Information Infrastructure with regard to data format standards, data storage guidelines and best practices.

◇ Provide post-implementation support to clients, trouble shooting and reporting issues to team leaders.

KAIMING SOFTWARE CO., LTD.

Dalian, China (Part-Time), 2000 – 2002

Acted as an application developer, test case developer, and documentation author in dozens of projects. *Key accomplishments include:*

◇ A Windows application for desktop customization.

◇ GA algorithm improvement in terms of multiple optimization objectives.

◇ User manual authorization for multiple systems.

◇ Heuristic layout algorithm for rendering entries and their relations in given area.

Technical Proficiencies

Database

SQL SERVER 2000/2005/2008: Proficient in data modeling (UML), database design (SQL), store procedure design, and ETL tools like SSIS.

BUSINESS INTELLIGENCE: Data warehouse design (snowflake/star scheme), OLAP provision using Analysis Service, Reporting using Reporting Service.

Software Development

LANGUAGES: C/C++/C#, Java, Python.

PLATFORM: .NET Framework, ASP.NET, Ajax, LING, WCF/WWF/WPF, MOSS 2007, WSS.

METHODOLOGIES: UML models, design patterns, agile development, Object-/Interface-/Component-Oriented design. Familiar with all aspects of Software Development Life Circle.

SOFTWARE PACKAGES: Omnet++, Matlab, CPLEX, Visual Studio System.