2014

Bio-inspired cost-aware optimization for data-intensive service provision

Lijuan Wang
University of Wollongong

School of Information Systems and Technology

Bio-Inspired Cost-Aware Optimization for Data-Intensive Service Provision

Lijuan Wang

Supervisor

Dr. Jun Shen

This thesis is presented as part of the requirements for the

Award of the Degree of

Doctor of Philosophy

of the

University of Wollongong

July 2014

Declaration

I, Lijuan Wang, declare that this thesis, submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy, of the School of Information Systems and Technology, University of Wollongong, is wholly my own work unless otherwise referenced or acknowledged. The document has not been submitted for qualifications at any other academic institution.

Lijuan Wang, July 2014

Abstract

The rapid proliferation of enormous sources of digital data and the development of cloud computing have led to greater dependence on data-intensive services. Each service may actually request or create a large amount of data sets. The scope, number, and complexity of data-intensive services are all set to soar in the future. To compose these services will be more challenging. Issues of autonomy, scalability, adaptability, and robustness, become difficult to resolve. Bio-inspired algorithms can overcome the new challenging requirements of data-intensive service provision. It is useful for the provision of data-intensive services to explore key features and mechanisms of biological systems and accordingly to add biological mechanisms to services.

This thesis discusses the key concepts and approaches behind Web service concretization. It presents a hierarchical taxonomy of Web service concretization approaches and provides a detailed analysis of each approach. The taxonomy helps us not only to identify existing algorithms developed in service concretization but also to evaluate the applicability of different approaches to the data-intensive service provision problem.

This thesis concentrates on the bio-inspired algorithms in the taxonomy. To this end, it conducts a systematic review of Web service concretization based on bio-inspired algorithms. The systematic review investigates the extent of applications of bio-inspired algorithms to QoS-based Web service concretization, which is not covered by previous studies. Then the comprehensive applications of an ant colony system and a genetic algorithm for cost-aware data-intensive service provision are introduced. The performance of the two proposed algorithms is evaluated and compared with other traditional methods. Further, this thesis investigates an ant colony system and a genetic algorithm for the multi-objective data-intensive service

provision. Both the algorithms get a set of Pareto-optimal solutions by considering the total cost and the total execution time of a composite service at the same time. Experiments using many different scenarios with respect to five performance metrics have been conducted to evaluate the proposed algorithms.

This thesis then further studies an ant colony system for the dynamic data-intensive service provision problem. In the dynamic environment, the service composition optimization process should be conducted repeatedly when the changes of the states of services occur. In order to adapt the ant colony system to handle the dynamic scenarios, several pheromone modification strategies in reaction to changes are discussed. The aim of the strategies is to find a balance between preserving enough old pheromone information to speed up the search process, and resetting enough new pheromone information to facilitate the ants to find a good solution for the changed scenarios. The strategies differ in their degree of reinitialized pheromone values with respect to the information that has been used to decide the amount of pheromone values. Moreover, the behavior of different strategies for modifying pheromone information is compared.

Data-intensive services are typically used in a dynamic and changing environment, and different providers typically have conflicting objectives. In order to automate the process of reaching an agreement in the data-intensive service provision, this thesis finally proposes an ant-inspired negotiation approach. It uses a group of agents automatically negotiating to establish agreeable service contracts. The lifetime of the complete data-intensive service provision and the two-stage negotiation procedures are described. A multi-phase, multi-party negotiation protocol is also designed. The negotiation approach is evaluated through simulations.

This thesis also points out future research topics when it concludes its contributions.

Acknowledgments

The PhD study is a long journey, throughout which I have received support from many people whom I wish to acknowledge herein. First and foremost, I would like to express my sincere gratitude to my supervisor, Dr. Jun Shen, for his encouragement, guidance, support, and patience during my doctoral study. I thank him for proactively extending support to me during a low point in my research. His constant pursuits for perfection and valuable suggestions have guided me in every step of my study. This dissertation would not have been possible without his invaluable advices and guidance. I feel so lucky to have Dr. Jun Shen as my supervisor.

I would like to thank anonymous reviewers for their valuable comments on earlier drafts of my publications. I would like to thank Dr. Madeleine Cincotta for the help in the final language editing of earlier drafts of my publications. The staff and other PhD students in the School of Information Systems and Technology also provided a lot of helpful advices on my research.

I would like to thank Demin Gao, Zhengrong Huang, and Lei Lv, for their friendship. We started our PhD candidatures together and have shared many experiences that will be cherished memories for many years. I would like to thank Simon Bailey, Jessica Bailey, and Crystal Jade Wickett for teaching me the Bible. I would also like to thank the brothers and sisters in the Kingdom Hall for sharing and discussing the Bible with me.

I owe a huge debt of gratitude to my grandfather, my parents, and my brother, for their constant support and encouragement. They have always been there whenever I needed them. Without their endless love and support, I could not have reached this far. I especially should thank my husband, Mr. Qiaojin Xing, who supported my three years of research with an open heart and mind and has encouraged me in every step of my career and life. In particular, I thank him sincerely for visiting my

parents when I was abroad.

Finally, I would like to express my sincere appreciation to the University of Wollongong for offering the International Postgraduate Tuition Award to financially support my study. Also, I would like to express my gratitude to the China Scholarship Council and the Ministry of Education of the P. R. China for awarding me a scholarship to pursue my study in Australia as a PhD student. This has given me an opportunity to study abroad.

Contents

Abstract i
Acknowledgments iii
Table of contents v
List of figures ix
List of tables xii
1 Introduction 1
1.1 Background 1
1.1.1 Big Data and Cloud Computing 1
1.1.2 Web Services 4
1.1.3 Service Composition 6
1.2 Motivation 8
1.3 Problem Description and Challenges 10
1.4 Contributions 13
1.5 Thesis Overview 15
2 Web Service Concretization 16
2.1 Terms and Definitions 16
2.2 The Web Service Concretization Architecture 18
2.3 QoS Attributes in Web Service Concretization 19
2.3.1 Service Level Agreement 19
2.3.2 Definition of QoS Attributes 20
2.3.3 Classification of QoS Attributes 20
2.3.4 Normalization of QoS Attributes 20
2.4 Web Service Concretization as a Workflow Design Problem 22
2.4.1 Web Service Concretization Structures 22
2.4.2 QoS Attributes of Composite Services 23

2.5 Web Service Concretization Approaches . . . 24
2.5.1 Local Optimization Approaches . . . 26
2.5.2 Global Optimization Approaches . . . 27
2.5.3 Analysis of the Existing Web Service Concretization Approaches 28
2.6 Summary . . . 29
3 A Systematic Review of Bio-Inspired Service Concretization . . . 31
3.1 Key Attributes of Biological Systems . . . 31
3.2 Method of Systematic Review . . . 33
3.2.1 Research Questions . . . 33
3.2.2 Specification for Literature Review Strategy . . . 33
3.3 Results and Synthesis of Findings . . . 35
3.3.1 Ant Colony Optimization Algorithms . . . 35
3.3.2 Genetic Algorithms . . . 39
3.3.3 Particle Swarm Optimization Algorithms . . . 44
3.3.4 Combined Bio-Inspired Algorithms . . . 46
3.3.5 Synthesis of Findings . . . 47
3.4 Summary . . . 49
4 Cost-Aware Data-Intensive Service Provision . . . 51
4.1 Introduction . . . 51
4.2 Data-Intensive Service Provision . . . 52
4.2.1 An Economic Model . . . 52
4.2.2 An Extensible QoS Model . . . 54
4.2.3 Service Provision Model Using an AND/OR Graph . . . 55
4.2.4 Objective Function . . . 56
4.3 Data-Intensive Service Provision Based on an Ant Colony System . . . 57
4.3.1 State Transition Rule . . . 58
4.3.2 Local Updating Rule . . . 59
4.3.3 Global Updating Rule . . . 59
4.3.4 Ant Replication and Death Rule . . . 60
4.4 Data-Intensive Service Provision Based on a Genetic Algorithm . . . 60
4.4.1 The Implementation of a Genetic Algorithm . . . 61
4.5 Experiments and Analysis . . . 64

4.5.1 The Parameters . . . 66
4.5.2 Evaluation Methodology . . . 66
4.5.3 The Dataset . . . 67
4.5.4 Results Analysis . . . 67
4.6 Related Work . . . 71
4.7 Summary . . . 75
5 Multi-Objective Data-Intensive Service Provision . . . 76
5.1 Background . . . 77
5.1.1 Multi-Objective Ant Colony Optimization Algorithms . . . 77
5.1.2 Multi-Objective Genetic Algorithms . . . 78
5.1.3 Performance Metrics . . . 79
5.2 Multi-Objective Data-Intensive Service Provision with Global QoS Constraints . . . 81
5.3 Multi-Objective Data-Intensive Service Provision Based on an Ant Colony System . . . 83
5.3.1 State Transition Rule . . . 84
5.3.2 Local Updating Rule . . . 84
5.3.3 Global Updating Rule . . . 85
5.4 Multi-Objective Data-Intensive Service Provision Based on a Genetic Algorithm . . . 86
5.5 Experiments and Analysis . . . 87
5.5.1 Test Case Generation . . . 87
5.5.2 Results Analysis . . . 88
5.6 Related Work . . . 94
5.7 Summary . . . 97
6 Ant Colony System for Dynamic Data-Intensive Service Provision 99
6.1 Introduction . . . 99
6.2 Dynamic Ant Colony System . . . 100
6.2.1 Pheromone Modification Approaches . . . 101
6.2.2 Pheromone Modification Strategies . . . 102
6.3 Experiments and Analysis . . . 104
6.3.1 Results of Case Study One . . . 105

6.3.2 Results of Case Study Two . . . 111
6.4 Related Work . . . 115
6.5 Summary . . . 116
7 Ant-Inspired Negotiations in the Data-Intensive Service Provision 117
7.1 Introduction . . . 118
7.2 The Lifetime of Data-Intensive Service Provision . . . 119
7.2.1 First Step: Data-Intensive Service Composition . . . 120
7.2.2 Second Step: Two Negotiation Processes . . . 123
7.3 The Extended Negotiation Protocol . . . 123
7.4 Decision Making Model . . . 127
7.5 Experiments and Analysis . . . 130
7.5.1 Test Case Generation . . . 131
7.5.2 Results Analysis . . . 131
7.6 Related Work . . . 132
7.6.1 Data Replica Selection . . . 132
7.6.2 Negotiation in Service Provision . . . 137
7.7 Summary . . . 137
8 Discussions and Future Work . . . 139
8.1 Main Implications of this Research . . . 139
8.2 Limitations of the Current Work . . . 141
8.3 Possible Solutions . . . 142
8.3.1 Strategies to Lower the Total Cost . . . 142
8.3.2 Economic Mechanisms for Data-Intensive Service Provision . . . 149
8.4 Summary . . . 150
9 Conclusions . . . 151
A List of Articles Published During the Candidature . . . 154
References . . . 157

List of Figures

1.1 Some of the streams of Big Data 2
1.2 Cloud computing logical diagram 3
1.3 The general architectural model for Web services 6
1.4 AMS-02 SOC massive data handling processes 9
1.5 AMS-02 massive data processing framework 10
1.6 Selection of services and data replicas for a data-intensive application 11

2.1 Service functional graph 17
2.2 Service candidate graph 18
2.3 Web service concretization architecture 18
2.4 Four basic composition structures 23
2.5 A hierarchical taxonomy of Web service concretization approaches . . 25

3.1 A service composition graph when using an ACO algorithm 36
3.2 An example of how the genome is encoded when using a GA 40

4.1 Service and data set usage and charging relationship 53
4.2 An AND/OR graph for data-intensive service provision 56
4.3 Data-intensive service provision based on an ant colony system 61
4.4 An example of infeasible individuals generated by the crossover operator 63
4.5 An example of feasible and infeasible individuals generated by the mutation operator 64
4.6 Data-intensive service provision based on a genetic algorithm 65
4.7 The evolution of utility/fitness value in ACS and GA 68
4.8 The performance of ACS, GA, and MIP 69
4.9 The optimality of the two types of genetic algorithms 72
4.10 The values of FRIT of the two types of genetic algorithms 73

5.1 Dominated space 80
5.2 Median summary attainment surface vs. number of concrete services - part I 91
5.3 Median summary attainment surface vs. number of concrete services - part II 92
5.4 Median summary attainment surface vs. number of abstract services . 93

6.1 A graph for the data-intensive service provision problem 102
6.2 The optimization behavior of R-strategy with all parameters in case study one 106
6.3 The optimization behavior of η-strategy with all parameters in case study one 106
6.4 The optimization behavior of τ-strategy with all parameters in case study one 107
6.5 The optimization behavior of G-strategy in case study one 107
6.6 The loss in quality of the best found solution of all strategies for the case of $(k, t) = (1, 50)$ in case study one 108
6.7 The loss in quality of the best found solution of R-strategy with different values of γ_R, k and t in case study one 109
6.8 The loss in quality of the best found solution of η-strategy with different values of γ_E, k and t in case study one 109
6.9 The loss in quality of the best found solution of τ-strategy with different values of γ_T, k and t in case study one 110
6.10 The loss in quality of the best found solution of G-strategy with different values of k and t in case study one 110
6.11 The optimization behavior of R-strategy with all parameters in case study two 111
6.12 The optimization behavior of η-strategy with all parameters in case study two 112
6.13 The optimization behavior of τ-strategy with all parameters in case study two 112
6.14 The optimization behavior of G-strategy in case study two 113

6.15 The loss in quality of the best found solution of R-strategy with different values of γ_R, k and t in case study two 113
6.16 The loss in quality of the best found solution of η-strategy with different values of γ_E, k and t in case study two 114
6.17 The loss in quality of the best found solution of τ-strategy with different values of γ_T, k and t in case study two 114
6.18 The loss in quality of the best found solution of G-strategy with different values of k and t in case study two 115

7.1 The lifetime of the data-intensive service provision 119
7.2 Protocol to support first phase of negotiation 125
7.3 Protocol to support second phase of negotiation 125
7.4 Protocol to support third phase of negotiation in case of successful negotiation 126
7.5 Protocol to support third phase of negotiation in case of failed negotiation 126
7.6 Success rate of the negotiation approach and the MIP approach . . . 133
7.7 Number of negotiation rounds 134
7.8 Computation time per negotiation round 135

8.1 Before data partition and replication takes place 145
8.2 After data partition and replication has taken place 145

List of Tables

2.1 QoS aggregation functions for different composition structures 24

3.1 Distribution of papers among different publication sources 34
3.2 Distribution of primary studies 35
3.3 A summary of results on applying ACO algorithms to Web service concretization problems 38
3.4 A summary of results on applying GAs to Web service concretization problems 42
3.5 A summary of results on applying PSO algorithms to Web service concretization problems 46
3.6 A summary of results on applying combined bio-inspired algorithms to Web service concretization problems 47
3.7 Main control parameters in three bio-inspired algorithms 48

4.1 Means of optimality 70
4.2 Means of FRIT 71

5.1 MOACO algorithmic components and values 78
5.2 Means of the computation time 88
5.3 Means of ONVG 89
5.4 Means of $PS(A)$ 90
5.5 Means of C Metric 90
5.6 The comparison of the MOACS and MOGA with other studies 97

Chapter 1

Introduction

This chapter introduces a high-level description of the research presented in this thesis. It begins with an introduction to the general area of cloud computing and Big Data, and presents the motivations and problem descriptions for composing data-intensive services in such context. Then it describes the primary contributions of this thesis. The chapter ends with an overview of the organization of the thesis.

1.1 Background

1.1.1 Big Data and Cloud Computing

The world is filled with an unimaginable amount of digital information, which is getting even more vast [56]. Huge collections of data have been created due to the advances in digital sensors, communications, computation, and storage. Every area of the global economy is feeling the impact of data explosion. Scientists and computer engineers have coined a new term for this phenomenon: "Big Data". According to the EMC-sponsored IDC Digital Universe study, the world's data is doubling every two years, with 1.8 trillion gigabytes being created and replicated in 2011. Overall data will grow 50 fold by 2020 [74]. Figure 1.1 depicts some of the streams of Big Data. This data deluge exhibits not just volume and velocity but also variability and diversity of structure, completeness, and domain [48]. The McKinsey Global Institute argues, in a suitably fact-packed new report, that data is becoming a factor in production, like human capital or physical capital [121]. A report by the forum "Big Data, Big Impact" declared data as a new class of economic asset, like currency

Figure 1.1: Some of the streams of Big Data (Credit: image courtesy of Miller [126])

or gold [107]. The impact of enormous new sources of data extends to many areas of society, far beyond business, industry, government, science, sports, advertising and public health, with no area being untouched. The IT services industry is already abuzz about Big Data. Big Data allows us to discover much more knowledge about the world, not only with structured internal data but also using unstructured data from external sources. Meanwhile, industries can now make key business decisions by predicting how customers and competitors' customers will behave and how their behavior may change. Obtaining deeper insights into such key business decisions is critical for better-targeted marketing. Furthermore, Big Data helps to achieve a larger share of profit in terms of both market and customer. The Gartner Group predicts that, "The need to access and the ability to identify individual customers, products and suppliers referenced in 'Big Data' sources such as social networks, and to link them to internally authored master data, will provide new opportunities to improve marketing, sales and customer service" [165]. The data captures and presents valuable information to business, science, government, and general society. For example, search engine companies such as Google, Yahoo!, and Microsoft have created an entirely new business by capturing the information freely available on the World Wide Web and providing it to people in useful ways [29]. These companies continually add new services by collecting trillions of bytes of data every day. While

the manufacturing industry is hastening their transformation into the service industry, a number of data-oriented businesses are growing and developing. Mostly, their motivation is to create various connections with the data that will bring a deeper economic value. Analysts in the cloud sphere think that the next few years will see an explosion of Big Data enterprise services [165].

Cloud computing is a general term for anything that is delivered as a service over the Internet. These services are broadly divided into three categories: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Figure 1.2 shows an overview of cloud computing. The name cloud computing was inspired by the cloud symbol that is often used to represent the Internet in flowcharts and diagrams [149]. Cloud computing means different things to different people. Armbrust et al. [13] clarified that: "Cloud computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the data centers that provide those services". This definition is given

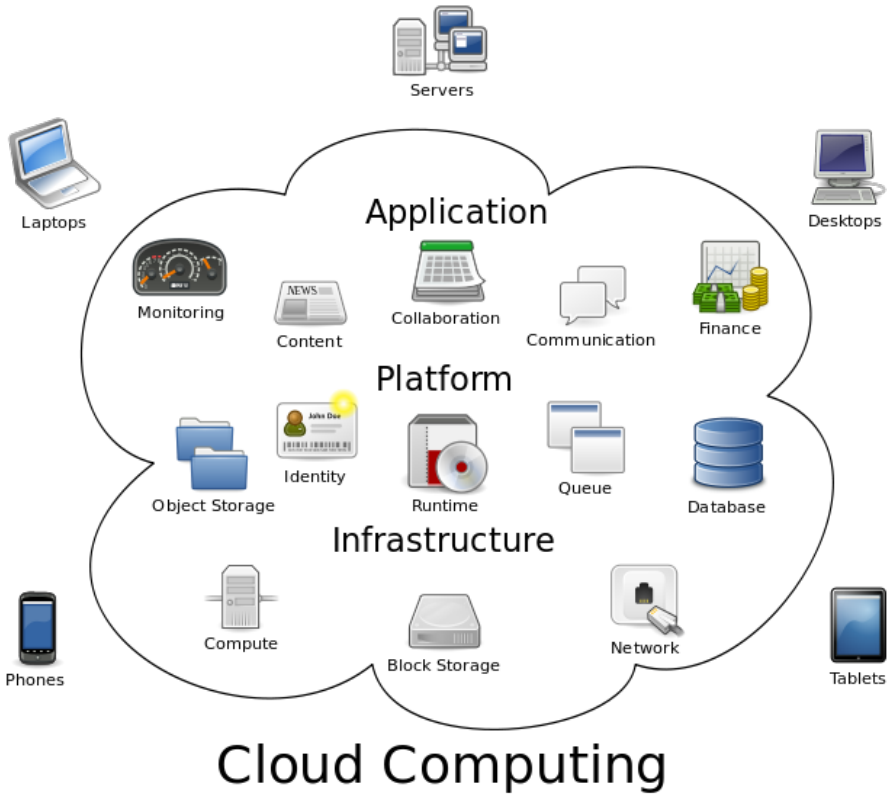


Figure 1.2: Cloud computing logical diagram (Credit: image courtesy of Johnston [97])

from both a software and a hardware point of view and stresses that both of them can be accessed through the Internet. Buyya et al. [30] proposed the following definition: "A cloud is a type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service level agreements established through negotiation between service providers and consumers". This definition is given from a market-oriented perspective and stresses the delivery of high performance IT services as computing utilities.

Cloud computing provides unlimited resources on demand. Considering Big Data and cloud together will accelerate the availability and acceptability to analyze the data. Big Data offers unprecedented business opportunities to enterprises. Being able to handle a large amount of information is a priority for big enterprises in the industry, because they will gain a competitive advantage by turning a huge amount of information into business value. Cloud computing provides many data services in the area of Big Data analytics. Cloud infrastructure and platforms will play a huge role in accessing, processing, and analyzing massive amounts of data.

1.1.2 Web Services

The term "Web service" is a buzzword of our era, it means different things to different people. Singh and Huhns [156] pointed out that the definitions of Web service given by different communities reflected their backgrounds and concerns. The World Wide Web Consortium (W3C) uses the following definition: "A Web service is a software system designed to support inter-operable machine-to-machine interaction over a network. It has an interface described in a machine-processable format. Other systems interact with the Web service in a manner prescribed by its description using SOAP (Simple Object Access Protocol) messages, typically conveyed using HTTP with an XML (eXtensible Markup Language) serialization in conjunction with other Web-related standards" [26]. This definition has been published by the W3C, in the Web services architecture document.

Web services perform functions that can range from answering simple requests (such as a weather report, or credit checking and authorization) to executing sophisticated business processes (such as a travel planner, or insurance brokering system)

requiring peer-to-peer relationships between service consumers and providers [57]. Service Oriented Computing (SOC) refers to a computing paradigm that utilizes services to support the development of rapid, low-cost, inter-operable, and distributed applications [132]. The aim of SOC is to create dynamic business processes and agile applications covering different organizations and computing platforms by assembling application components into a loosely coupled network of services. Jamil [95] points out that the redefinition of data is making service oriented computing model come back. For example, given the fact that data sizes are large, data is distributed, data changes frequently, and the data backup requirement.

SOC leads to the term Service Oriented Architecture (SOA), an architecture which focuses on building systems based on services [61]. SOA is a design pattern which is composed of loosely coupled, discoverable, reusable, inter-operable platform agnostic services in which each of these services follow a well-defined standard. Each of these services can be bound or unbound at any time and as needed [95]. SOA describes an architectural style and it does not limit itself to one implementation technology [57]. Web services are one set of technologies and composable standards with well-defined interfaces, which is one of the most active and widely adopted implementations of SOA [57, 95]. The advent of Web services standards boosted the proliferation of the principles as well as the actual implementations of service orientation. Service orientation is founded on the idea of composing business processes or workflows by discovering and invoking the most suitable services such as based on cost or other quality of service attributes, rather than building new applications to satisfy a particular domain or business requirement [57].

The general architectural model for Web services is shown in Figure 1.3. It consists of three types of participants: the service providers, the service repository, and the service requesters. The service provider describes Web services and publishes them to a service repository. The service requester finds services in the service repository. Service providers and service requesters use a common language provided by the XML to connect and exchange information. WSDL (Web Services Description Language), UDDI (Universal Description, Discovery and Integration), and SOAP are three XML-based standards developed for the service description, discovery, and communication. WSDL is used to describe the attributes of a Web service, such as

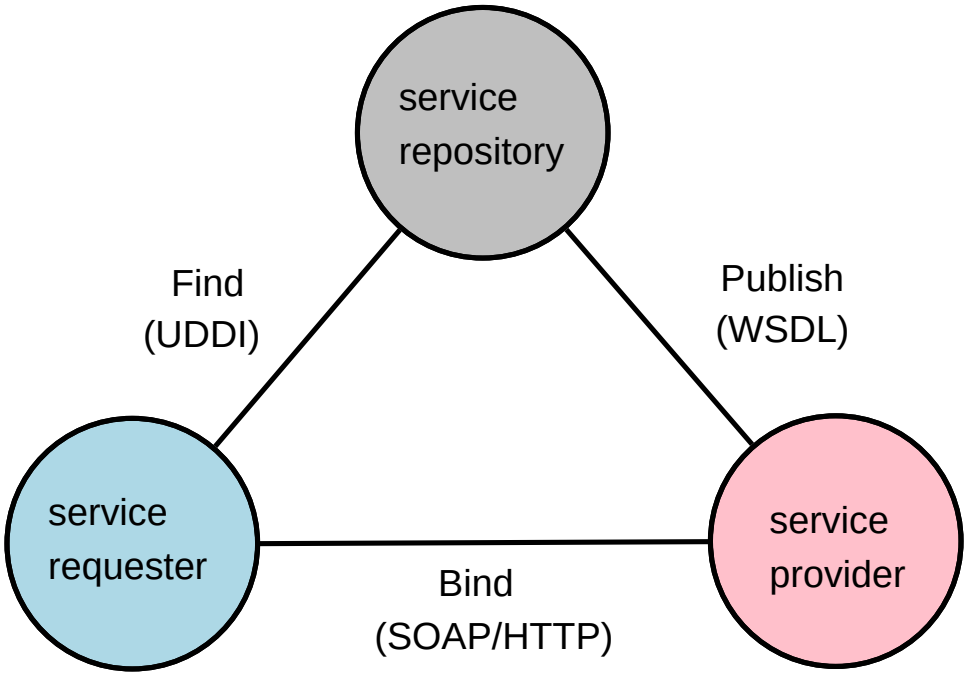


Figure 1.3: The general architectural model for Web services (Credit: image courtesy of Singh and Huhns [156])

the name, function, parameter, and how to invoke it. UDDI is used to register and discover Web services. SOAP is used to transfer messages between different systems in HTTP and XML format.

Services may be implemented on a single machine or a large number and variety of devices, and be distributed on a local area network or more widely across several wide area networks [57]. When the services use the Internet as the communication mechanism, the resulting Web services require special consideration as a result of using a public, insecure, low-fidelity mechanism for inter-service interactions [28]. In this research, the term 'service' is simply used in the place of 'Web service'.

1.1.3 Service Composition

The implementation of a basic service refers to the simple interactions between a client and a server. If the implementation of a service invokes other services, it is necessary to combine the functionalities of several services and this service is referred to as a composite service [7]. The process of developing a composite service in turn is called service composition [57]. Service composition can be performed by composing either elementary services or composite services. Composite services in turn are recursively defined as an aggregation of elementary services and composite

services [57, 99]. Service composition accelerates rapid application development, offers service reuse possibilities, ensures complex service consummation from the service provider's perspective, and provides seamless access to a variety of complex services from the service consumer's perspective [124].

When composing services, the functionalities of the client's request are implemented by several services. This is analogous to the workflow management, where the application is represented as a workflow structure, which consists of tasks, data flows, and control flows. Many scientific applications in the field of astronomy [24], life-sciences [32], high-energy physics [53], climate and weather modeling [129], and gravitational-wave physics [168] have used workflow technologies to carry out large-scale experiments. The scientists in these areas share and collaborate on a variety of resources in the pursuit of common goals. These resources are distributed and may encompass people, heterogeneous tools, data from different sources, and machines spread across the world. The service oriented architecture with its implementation in the Web services, seems a viable approach to execute scientific applications by providing platform-independent and language-neutral service interfaces, that hide the complexity while providing a well-defined and high performance Quality of Service (QoS) attributes [103, 138].

The scientific applications usually have a large number of tasks and require the manipulation of a large amount of data, many of them are implemented as workflows of services [9, 91, 98, 127, 131, 171]. To provide reusable analysis solutions for the large data resources, it is necessary to compose workflows of distributed services. The cost of the service composition process becomes extremely complicated in scientific data-intensive applications, as services produce and process terabytes or petabytes of data sets. During the execution of a data-intensive service composition process, services need more time to transfer and store data, and the access cost of each data set depends on the data requester's location and the amount of transferred data [111]. Therefore, how to select the data replica for each service to lower the access cost and the transferred cost and to optimize the service performance simultaneously is a challenging and emergent problem.

1.2 Motivation

Big Data has attracted much research attention. The Gartner Group listed Big Data in the "10 Critical Tech Trends for the Next Five Years" [151]. Data-intensive science is emerging as the fourth scientific paradigm, and new techniques and technologies for the new scientific paradigm are needed [18]. On the one hand, Big Data provides opportunities and potential values. On the other hand, many challenges are arising with respect to the data capture, data storage, data analysis, data searching, data sharing, and data visualization [136]. The applications of the Big Data problems in scientific research fields are briefly introduced. Other applications such as in society administration, commence and business can refer to [136].

The National Aeronautics and Space Administration (NASA[a]) conducts the Earth Science program[b] to develop a scientific understanding of the global integrated earth system using space-based observations. The goal of the program is to understand the interactions among the atmosphere, oceans, land, ecosystems, and how human activities affect the environment. The Earth Observing System Data and Information System (EOSDIS[c]) is a major core data system, and it has twelve Distributed Active Archive Centers (DAAC[d]). Data from NASA's past and current Earth Observing System (EOS) satellites and field measurement programs are processed, archived, documented, and distributed to researchers and the general public by DAACs. NASA aircraft, in recording approximately four terabytes of data every day, maintains the world's largest scientific data and information system for collecting, processing, archiving, and distributing data from Earth system to worldwide users. The size of climate change data repositories alone are projected to grow to nearly 350 petabytes by 2030[e].

Another example is the Alpha Magnetic Spectrometer (AMS)[f] experiment, which uses cloud computing to process a huge amount of data. The AMS, also designated AMS-02, is a particle physics experiment module that is mounted on the International Space Station (ISS). The purpose of the AMS experiment is to advance

[a] http://www.nasa.gov
[b] http://science1.nasa.gov/earth-science/
[c] http://earthdata.nasa.gov/about-eosdis
[d] http://earthdata.nasa.gov/about-eosdis/system-description/eosdis-data-centers
[e] http://open.nasa.gov/blog/2012/10/04/what-is-nasa-doing-with-big-data-today/
[f] http://ams.nasa.gov/

knowledge of the universe and lead to the understanding of its origin by searching for antimatter and dark matter while performing precision measurements of cosmic rays composition and flux. The ground AMS lab is based at CERN[g] in Switzerland. The key technology for accessing the data collected from AMS relies on data services based on cloud computing. The AMS-02 SOC (Science Operation Center) at Southeast University in China (labeled as AMS-02 SOC@SEU) is supported by IBM Cloud Computing Center with 3500 CPU core and 500TB storage. The AMS-02 SOC@SEU typically receives 200GB of data from AMS and generates 700GB of data after processing them, on each day. Figure 1.4 shows the massive data handling processes of AMS-02 SOC, and Figure 1.5 presents the AMS-02 massive data processing framework.

Scientists and remote users deploy different processes, such as data mining, image processing, thematic map generation, or data query on a large amount of Earth Science data at existing NASA data provider sites and on data at AMS-02 SOC. A set of operations is often necessary to provide an appropriate solution to complex scientific applications. For example, in order to understand the regional scale impacts of fires, or the long-range transportation of pollutants on air quality, or even the implications of climate change, scientists need to combine data from aircraft and satellites. The use of Web services technologies provides valuable solutions to speed up the scientific data analysis [17, 23, 47, 116, 186, 189]. A composition of a set of services as a composite service can be reused by other researchers. For science problems involving a large amount of data, cost-effective mechanisms for data-intensive service provision are needed.

[g]http://home.web.cern.ch/about/experiments/ams

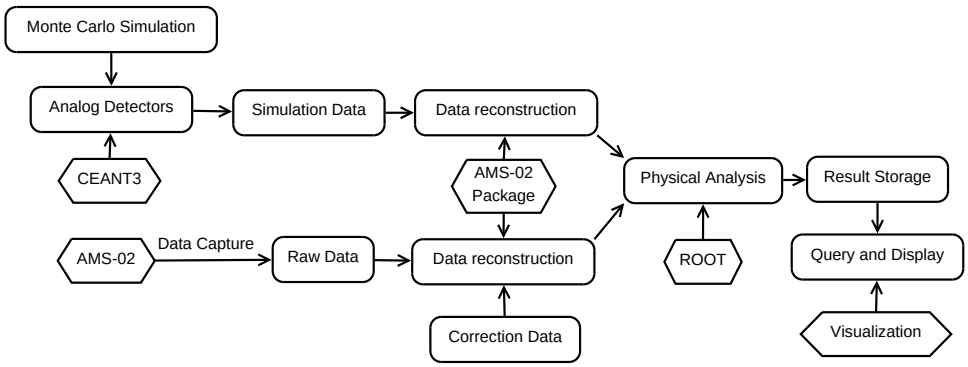


Figure 1.4: AMS-02 SOC massive data handling processes

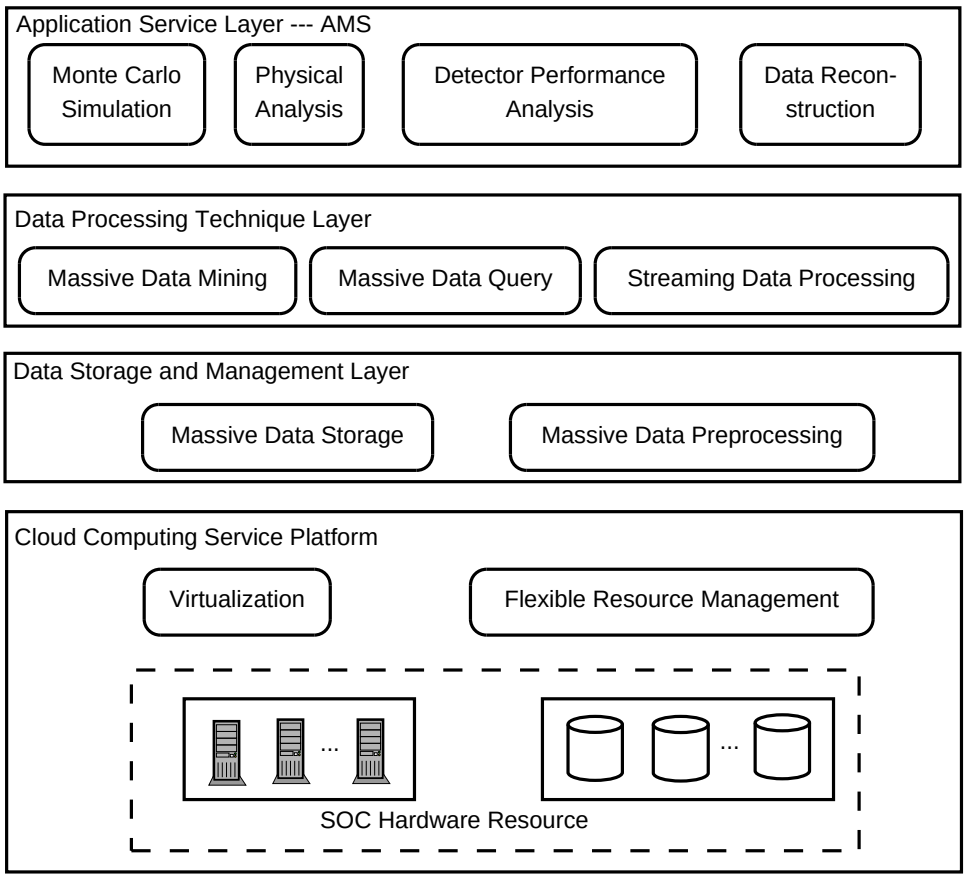


Figure 1.5: AMS-02 massive data processing framework

A survey of the challenges, techniques, and technologies of data-intensive applications was done by Philip Chen and Zhang [136] showing that bio-inspired computing was one of the ongoing or underlying techniques and technologies to harness Big Data. The authors stated that biological computing models were better appropriate for Big Data because they had mechanisms with high-efficiency to organize, access, and process data. Hence, this thesis focuses on how bio-inspired algorithms facilitate the data-intensive service provision.

1.3 Problem Description and Challenges

Generally, the notion of data-intensive services is defined as Web services that request or create very large data sets as inputs or outputs. Services use data from data providers, and they also use data from other services. The data from other services sometimes consists of data which is exchanged between services within a common context. This research focuses on the data from typical big data providers

as this type of data is generally much larger.

For data-intensive applications, a variety of services for data mining, data storage, data placement, data replication, data transfer, and data movement have been deployed in distributed computing environments. For a complex problem, it needs to compose various services into a composite service which can be reused by other programs. Data sets required by these services can be retrieved from several data centers as there are data replicas. For one data set, the access cost of its replica at one data center is different from other data centers [170]. Data-intensive service composition primarily focuses on some of the objective functions or a combination of them, such as minimizing the overall cost of the composite service, or providing service composition solutions under the time and budget constraints. The problem becomes complex with a large number of data sets and increasing functionality of related services. Figure 1.6 demonstrates the complexity of selecting services and data replicas for a data-intensive application. A data-intensive application is designed as a workflow of n abstract services, and there are m concrete services for each abstract service (the definitions of abstract services and concrete services are given in Chapter 2). This gives us m^n possible sets of services. Each abstract service requires k data sets, and each data set has l data replicas. This gives us l^k possible sets of data sets. The total possible sets for a composite service would be $m^n l^k$, and it becomes a challenging problem to find a composition solution. This problem becomes highly complex when there are constraints involved, such as QoS requirements, network costs, and storage limits.

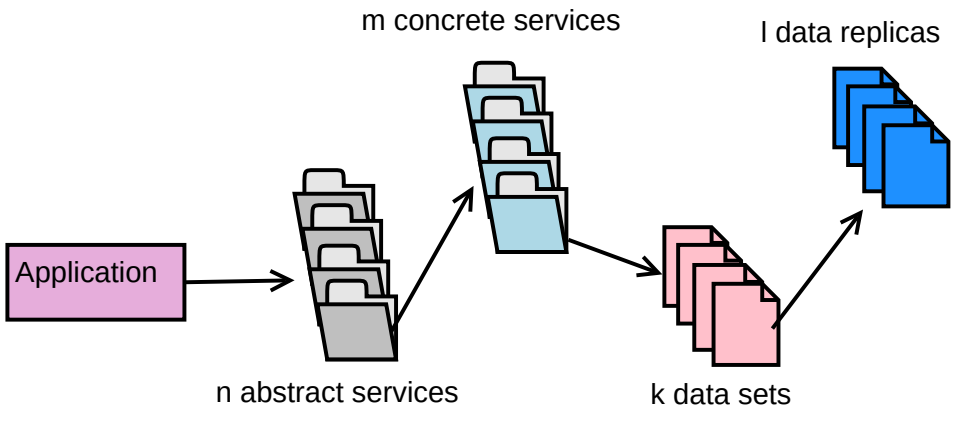


Figure 1.6: Selection of services and data replicas for a data-intensive application

The problem of data-intensive service composition faces many new challenges:

1. Data-intensive service composition mechanism. A composite service's overall cost and response time depends on the QoS attributes of the selected candidate services. The QoS attributes of each selected service depend on the cost and response time of the data sets required by the service. The traditional service composition approaches are not suitable to the data-intensive service composition problems, since the data-intensive services handle a huge amount of data sets. These data sets may each be replicated at different data centers. Mass data movement influences the performance of data-intensive applications, as data plays the dominant role in the composition process. Specifically, data providers charge data consumers based on the amount of transferred data and the consumers' location. The questions that need to be answered are:

   1.1 What approaches are still useful to solve these new service composition problems?

   1.2 Which approaches can be used to solve data-intensive service provision problems?

   1.3 How can candidate services that optimize the overall cost or time be selected?

   1.4 How can data replicas be selected?

2. Users' QoS requirements. The end-to-end QoS constraints such as budget (cost for using services) and response time (time taken for the response of service request), should also be taken into account while composing data-intensive services. For example, a user may require minimizing the execution time under the constraints with respect to price and reputation, while another user may focus more on price than execution time. The questions are:

   2.1 How can the QoS attributes of data-intensive services be computed?

   2.2 How can users' QoS requirements be incorporated into the data-intensive service provision mechanism so that either or both could be optimized?

3. Providers' objective. The data-intensive service composition will be cooperatively supported by three stakeholders: the service composers, the service

providers, and the data providers. Providers need an approach to regulate and price their resources. They all want to have a good market position maximizing their profits. The decisions of all three stakeholders depend on each other. Meanwhile, data-intensive services are typically used in a dynamic and changing environment, and different providers typically have conflicting objectives. The location of service requesters, service composers, service providers, and data providers will affect the total cost of service provision. The questions are:

3.1 How can the dynamic data-intensive service provision problem be solved?

3.2 How can different providers, for services, data or composite services, reach an agreement when they have conflicting objectives?

1.4 Contributions

To address the above challenges, this thesis makes several contributions towards improving the understanding of data-intensive service provision and towards advancing the area of data-intensive service composition based on bio-inspired algorithms:

1. This thesis discusses the key concepts behind the Web service concretization. It presents a hierarchical taxonomy of Web service concretization approaches and provides a detailed analysis of each approach. The proposed taxonomy, being the most comprehensive one to date, is used to identify algorithms developed in related areas that can be applied to the target research area. This also helps evaluate the applicability of different approaches to the data-intensive service composition problems.

2. This thesis conducts a systematic review of Web service concretization based on bio-inspired algorithms. The detailed investigation indicates that bio-inspired algorithms can overcome the new challenging requirements of the data-intensive service provision. It is useful for the provision of data-intensive services to explore key features and mechanisms of biological systems and accordingly to add biological mechanisms to services. The objective of the systematic review is to investigate the extent of applications of bio-inspired

algorithms to QoS-based Web service concretization, which is not covered by previous studies. This effort shows us an overview on the existing Web service concretization problems based on bio-inspired algorithms.

3. This thesis establishes an economic model and an extensible QoS model for the data-intensive service provision in order to analyze the interactions among the service composers, the service providers, and the data providers. The QoS model takes into account both the economic costs of provisioning a composite service, and the execution time for transferring the required data sets from different data servers to the service platform. Then the comprehensive applications of an ant colony system and a genetic algorithm for data-intensive service provision are studied. The objective functions of the two bio-inspired algorithms are to minimize the total cost of the composite service. Both algorithms are evaluated via simulations and compared with other traditional methods.

4. This thesis investigates a multi-objective ant colony system and a multi-objective genetic algorithm for the problem. Both the algorithms for a multi-objective context get a set of Pareto-optimal solutions by considering two objectives at the same time, i.e., the total cost and the total execution time of a composite service. The proposed algorithms are evaluated by experiments using many different scenarios with respect to five performance metrics.

5. This thesis presents an ant colony system for the dynamic data-intensive service provision. The service composition optimization process should be conducted repeatedly when the changes of the states of services occur, such as the changes of QoS attributes, the discontinuation of services, and the increase of new services. In order to adapt the ant colony system to handle the dynamic scenarios, this thesis discusses several strategies for modifying the pheromone information. The performance of each pheromone modification strategy is evaluated.

6. This thesis finally proposes an ant-inspired negotiation approach. The lifetime of the data-intensive service provision and two-stage negotiation procedures are described. It also designs a multi-phase, multi-party negotiation protocol,

where the ant colony system is applied for selecting the services. The proposed approach is evaluated via simulations and compared with other methods.

1.5 Thesis Overview

The research methodology of this thesisis is fully aligned with its main goals: investigating how bio-inspired algorithms facilitate the data-intensive service provision and designing mechanisms. The main objective of the literature review is to gain the fundamental of service provision, and also to have a comprehensive point of view about the related work. The literature focused on QoS-aware service composition for data-intensive applications, bio-inspired algorithm-based service composition, and cost-based service composition. Following a thorough literature review, there are three types of activities: model construction, design of optimization algorithms, and experimentation.

The rest of the thesis is organized as follows. Chapter 2 describes an overview of Web service concretization. It also presents a hierarchical taxonomy of Web service concretization approaches and provides a detailed analysis of each approach. This is followed by Chapter 3 which conducts a systematic review of Web service concretization based on bio-inspired algorithms. The thesis then concentrates on the cost-aware data-intensive service provision and introduces the comprehensive applications of an ant colony system and a genetic algorithm in Chapter 4. Chapter 5 investigates an ant colony system and a genetic algorithm for multi-objective data-intensive service provision. Chapter 6 then studies an ant colony system for the dynamic data-intensive service provision, while Chapter 7 proposes an ant-inspired negotiation approach for the data-intensive service provision. Finally, the thesis points out ideas and topics for future work in Chapter 8 and concludes in Chapter 9.

Chapter 2

Web Service Concretization

In service oriented computing, Web service selection is an important part of Web service composition. The Web service composition is achieved by solving the well-known Web service concretization problem. This chapter provides a general overview of Web service concretization, and covers topics such as Web service concretization architecture, QoS attributes in Web service concretization, and workflow-based Web service concretization. Meanwhile, this chapter presents a hierarchical taxonomy of Web service concretization approaches. It ends with a brief discussion on the existing Web service concretization approaches.

2.1 Terms and Definitions

Definition 2.1 (Abstract Services) *Abstract services represent the functional descriptions of services. These descriptions are extracted from the expression of a user's task which is combined with the user's preferences [85]. Abstract services have standard service interfaces across different service providers. An abstract service typically corresponds to a workflow task. Here* $\{AS_1, AS_2, \ldots, AS_n\}$ *is used to represent a set of abstract services.*

Definition 2.2 (Concrete Services) *Concrete services, also called service instances, are published by service providers. They represent the existing services which are available for potential invocation of their functionalities and capabilities [85]. Concrete services may have a similar functionality, for example, checking available tickets, making reservations, or planning meetings, but they differ with each other on*

non-functional qualities such as response time, service cost, and reliability.

Definition 2.3 (Candidate Relationship) *When the functions of several concrete services $cs_{i,1}$, $cs_{i,2}$, ..., $cs_{i,m}$ are consistent with the functional description of an abstract service AS_i, we state that concrete services $cs_{i,1}$, $cs_{i,2}$, ..., $cs_{i,m}$ and abstract service AS_i have a candidate relationship, and $cs_i = \{cs_{i,1}, cs_{i,2}, \ldots, cs_{i,m}\}$ is called the service candidate set of AS_i.*

In the context of Web service concretization, a directed graph is used to represent the dependencies between services, and it is referred to as a service functional graph. To compose a service is, essentially, to search for a solution in a solution space represented by a service candidate graph.

Definition 2.4 (Service Functional Graph) *To fulfill a user's requirements in functionalities, we construct abstract services as a workflow, which is represented as a directed graph G=(V,E,start,end), where $V = \{AS_1, AS_2, \ldots, AS_n\}$ and E represent the vertices and edges of the graph, respectively. There are only two virtual vertices, the start vertex which has no predecessors, and the end vertex which has no successors. Each edge (AS_i, AS_j) represents a relationship between AS_i and AS_j, which means that AS_i has to finish prior to invoking AS_j. This directed graph is called a service functional graph.*

Definition 2.5 (Service Candidate Graph) *By replacing abstract services in the service functional graph with their corresponding service candidate sets, and constructing a full set of connection links between concrete services of any two connected service candidate sets, the service candidate graph is created.*

Figure 2.1 shows a service functional graph, and Figure 2.2 shows a service candidate graph.

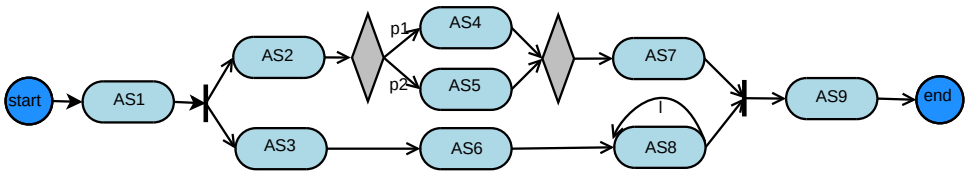


Figure 2.1: Service functional graph

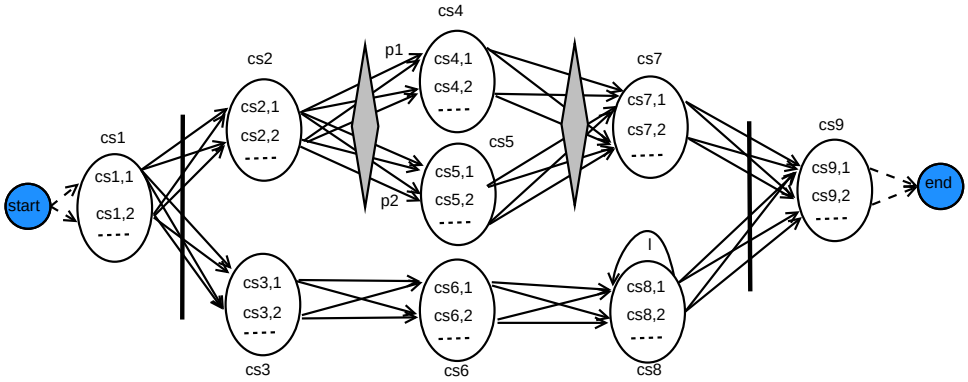


Figure 2.2: Service candidate graph

2.2 The Web Service Concretization Architecture

Claro et al. [44] proposed a Web service composition architecture which included four fundamental phases: planning, discovery, selection & optimization, and execution. This architecture is extended by adding corresponding components, which is depicted in Figure 2.3. When a composite service is requested, the first phase is to specify a workflow of the abstract services. After the workflow is specified, we can get a service functional graph. The second phase is discovery that aims at finding a set of service candidates from a service repository for each abstract service in the workflow. After this phase, a service candidate graph is created. The third phase aims at service selection and optimization, which is the focus of this

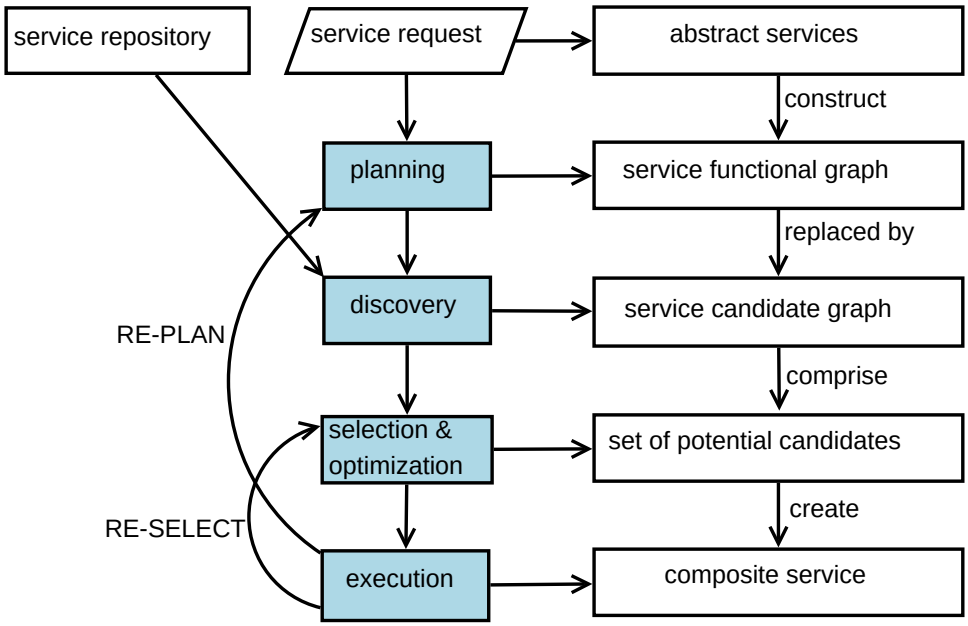


Figure 2.3: Web service concretization architecture (Adapted from [44])

research. After this phase, a set of potential service candidates is chosen to create a composite service. The fourth phase concerns execution, in which we attain the real composite service. Because services are in a highly dynamic environment, and different service requesters may have different requirements and preferences regarding QoS attributes, we need to re-plan the workflow or re-select service candidates if the concrete services are not found or the composite service does not satisfy the requirements.

2.3 QoS Attributes in Web Service Concretization

Prior to composing services, candidate services need to be discovered first and then selected. Discovering services means to match a user's request with service functionalities. During the process of discovering services, there are a set of services with similar functionality, which correspond to a user's request. Hence, we should consider criteria other than functionality to make a distinction between the selection process and the discovery process. QoS attributes are naturally deemed as such non-functional criteria. The following subsections will give a short introduction of the service level agreement, and the definition, classification, and normalization of QoS attributes.

2.3.1 Service Level Agreement

Typically the QoS attributes are clearly defined in a Service Level Agreement (SLA) [88]. A SLA can be regarded as a formal or an informal contract between service requesters and service providers. The service definition and the service performance are commonly recorded in SLAs. The SLA defines the agreed performance characteristics and also specifies how they will be evaluated and measured. The process of maintaining SLAs gives an opportunity to service providers to improve their service performance. It also allows service requesters to review service priorities. Hence, the SLA facilitates, and also constrains, the cooperation between service requesters and service providers.

2.3.2 Definition of QoS Attributes

QoS attributes for services refer to various non-functional attributes that are used for distinguishing services with identical functionality. Most approaches to the service concretization problem are trying to find a combination of concrete services based on QoS attributes. There is no standard mechanism, however, to include all or any subsets of the non-functional properties when developing services. In the International Standard ISO 8402, quality is defined as the totality of an entity that bear on its ability to satisfy stated or implied needs [104]. A set of thirteen quality attributes has been identified as important factors in the design of SOAs [195]. They are: interoperability, reliability, availability, usability, security, performance, scalability, extensibility, adaptability, testability, auditability, operability, deployability, and modifiability.

2.3.3 Classification of QoS Attributes

QoS attributes can be divided into two groups: quantitative attributes and qualitative attributes [120]. The former attributes such as response time and availability are quantitatively measured using metrics, whereas the latter attributes such as security cannot be measured quantitatively. For the sake of simplicity, and without loss of generality, in this research only quantitative QoS attributes are considered, since qualitative attributes can be represented as quantitative attributes by using specific functions. For example, the flexibility attribute has the value type {inflexible, flexible, very flexible}, which can be mapped to the set {0, 1, 2}. The quantitative QoS attributes are in turn divided into two classes: negative attributes and positive attributes. The first class of attributes such as response time and cost have a negative effect on QoS quality, hence they should be optimally minimized because the higher the value, the lower the quality. On the contrary, positive QoS attributes such as availability and reliability should be maximized.

2.3.4 Normalization of QoS Attributes

For an abstract service AS_i, there is a service candidate set of $cs_i = \{cs_{i,1}, \ldots, cs_{i,m}\}$ that can be used to implement it. Each concrete service $cs_{i,j}$ is associated with a

QoS vector $q_{cs_{i,j}} = [q^1_{cs_{i,j}}, q^2_{cs_{i,j}}, \ldots, q^r_{cs_{i,j}}]$ with r QoS parameters ($i \in \{1, 2, \ldots, n\}$, $j \in \{1, 2, \ldots, m\}$), where n is the number of abstract services in the composite service, and m is the number of concrete services in each service candidate set. In order to evaluate the multidimensional quality of concrete service $cs_{i,j}$, an evaluation function is used. The function maps the QoS vector $q_{cs_{i,j}}$ into a single real value to rank service candidates. In this research, a multiple attribute decision-making approach for the evaluation function is used, namely, the simple additive weighting (SAW) method [198]. There are two phases in applying SAW:

1. Scaling phase. As already mentioned, QoS attributes can be either negative or positive.

$$V^k_{cs_{i,j}} = \begin{cases} \dfrac{Q^{max}_{k,i} - q^k_{cs_{i,j}}}{Q^{max}_{k,i} - Q^{min}_{k,i}} & \text{if } Q^{max}_{k,i} - Q^{min}_{k,i} \neq 0 \\ 1 & \text{if } Q^{max}_{k,i} - Q^{min}_{k,i} = 0 \end{cases} \tag{2.1}$$

$$V^k_{cs_{i,j}} = \begin{cases} \dfrac{q^k_{cs_{i,j}} - Q^{min}_{k,i}}{Q^{max}_{k,i} - Q^{min}_{k,i}} & \text{if } Q^{max}_{k,i} - Q^{min}_{k,i} \neq 0 \\ 1 & \text{if } Q^{max}_{k,i} - Q^{min}_{k,i} = 0 \end{cases} \tag{2.2}$$

   For negative QoS attributes, values are scaled according to (2.1). For positive QoS attributes, values are scaled according to (2.2). In (2.1) and (2.2), $Q^{max}_{k,i}$ and $Q^{min}_{k,i}$ ($k \in \{1, 2, \ldots, r\}$) are used to represent the maximal value and the minimal value of the k-th QoS attributes of all concrete services in candidate set cs_i, which are given by (2.3).

$$\begin{aligned} Q^{min}_{k,i} &= \min_{\forall cs_{i,j} \in cs_i} q^k_{cs_{i,j}} \\ Q^{max}_{k,i} &= \max_{\forall cs_{i,j} \in cs_i} q^k_{cs_{i,j}} \end{aligned} \tag{2.3}$$

2. Weighting phase. The overall quality score of $cs_{i,j}$ is computed according to (2.4).

$$score_{cs_{i,j}} = \sum_{k=1}^{r} (V^k_{cs_{i,j}} * W_k) \tag{2.4}$$

The variable W_k ($W_k \in [0,1]$, $\sum_{k=1}^{r} W_k = 1$) represents the weight of k-th quality attribute, which is normally provided by the service requesters based on their own preferences.

2.4 Web Service Concretization as a Workflow Design Problem

According to Casati et al. [38], a composite service is similar to a workflow. The main specification of a composite service is the flow of service invocations (a set of atomic services, the control flow, and the data flow among these services). Similarly, the specification of a workflow is the flow of work (a set of work items, the control flow, and the data flow among these work items). Cardoso et al. [37] proposed a model for service composition using a workflow, and they argued that tasks and services should be treated as the same. They maintained that, since workflows and processes had precisely the same characteristics, processes could be viewed as workflows that managed services instead of tasks. Thus, in a workflow-based service concretization model, a workflow is actually a composite service.

2.4.1 Web Service Concretization Structures

A set of basic workflow patterns commonly used by concretization approaches [200, 203], which cover most of the structures specified by composition languages (such as YAWL [169] and BPEL [130]), are considered in this research. The general patterns are: sequence, parallel split (AND-split), synchronization (AND-join), exclusive choice (XOR-split), simple merge (XOR-join), and loop. Figure 2.4 gives the four basic Web service composition structures, and other complex structures can be considered as a combination of the four basic structures. Figure 2.1 is actually a simplified workflow.

Using the conditional probability (p_1 and p_2) of XOR branches and the maximum iteration numbers (l) of the loop structure, the workflow of a composite service can be transformed to a directed acyclic graph (DAG) through the use of loop pealing [14]. From the transformed DAG, a set of execution paths that identify all

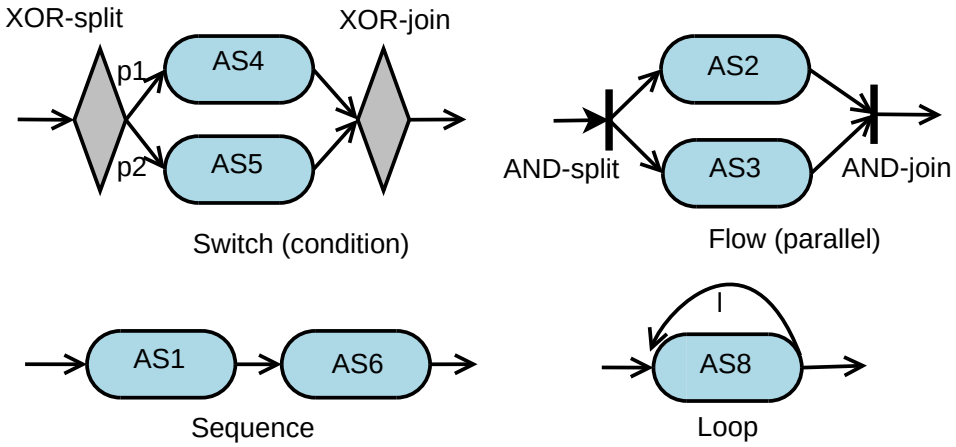


Figure 2.4: Four basic composition structures

possible execution scenarios of the composite service can be derived. An execution path is a sequence of abstract services $\{AS_1, AS_2, \ldots, AS_n\}$, where AS_1 and AS_n are the initial and final abstract service, and no abstract services AS_{i_1} and AS_{i_2} $(i_1, i_2 \in \{1, 2, \ldots, n\})$ belong to alternative branches. Every execution path has an associated probability that can be evaluated as the product of the probability of the conditional branches included in the execution path. For example, there are two execution paths, called ep_1 and ep_2, in Figure 2.1.

$ep_1 : \{AS_1, AS_2, AS_3, AS_4, AS_6, AS_7, l * AS_8, AS_9\}$, with a probability of p1;

$ep_2 : \{AS_1, AS_2, AS_3, AS_5, AS_6, AS_7, l * AS_8, AS_9\}$, with a probability of p2.

For each abstract service AS_i of an execution path, a concrete service is selected from the service candidate set associated with AS_i, then a concrete execution path is derived.

2.4.2 QoS Attributes of Composite Services

The evaluation of QoS attributes of a composite service depends on the composition structures and the QoS aggregation functions. The overall QoS attributes of a composite service can be computed by applying the functions described in Table 2.1, which shows an aggregation formula for each composition structure and QoS attribute. For a sequence structure of services (s_i, $i = 1, 2, \ldots, ns$, ns is the number of services in the sequence structure), the response time (RT) and cost (C) functions are additive while availability (Av) and reliability (Re) functions are multiplicative.

Table 2.1: QoS aggregation functions for different composition structures (Credit: table courtesy of Canfora et al. [34])

QoS Attributes	Composition Structures			
	Sequence	XOR	AND	Loop
Response time (RT)	$\sum_{i=1}^{ns} RT(s_i)$	$\sum_{i=1}^{nc} pc_i * RT(s_i)$	$Max\{RT(s_i)_{i\in\{1,\ldots,np\}}$	$l * RT(s)$
Cost (C)	$\sum_{i=1}^{ns} C(s_i)$	$\sum_{i=1}^{nc} pc_i * C(s_i)$	$\sum_{i=1}^{np} C(s_i)$	$l * C(s)$
Availability (Av)	$\prod_{i=1}^{ns} Av(s_i)$	$\sum_{i=1}^{nc} pc_i * Av(s_i)$	$\prod_{i=1}^{np} Av(s_i)$	$[Av(s)]^l$
Reliability (Re)	$\prod_{i=1}^{ns} Re(s_i)$	$\sum_{i=1}^{nc} pc_i * Re(s_i)$	$\prod_{i=1}^{np} Re(s_i)$	$[Re(s)]^l$

For a XOR (also called conditional) structure of services (s_i, $i = 1, 2, \ldots, nc$, nc is the number of services in the conditional structure), the QoS attributes are always evaluated as a sum of the attribute value of each service times the probability of the branch (pc_i, and $\sum_{i=1}^{nc} pc_i = 1$) to which the service belongs. The aggregation functions for the AND (also called parallel) structure are essentially the same as those for the sequence structure, except for the RT attribute where it is the maximum response time of the parallel services (s_i, $i = 1, 2, \ldots, np$, np is the number of services in the parallel structure). Finally, a loop structure with l iterations of service s is equivalent to a sequence structure of l copies of service s. Table 2.1 only contains four QoS attributes. It should be noted that other attributes and their aggregation functions can also be specified similarly.

2.5 Web Service Concretization Approaches

The final goal of the composite service construction is achieved by solving the well-known service concretization problem. The best service available at runtime has to be selected from all candidate ones for each abstract service, taking into consideration the global and local QoS constraints given by the service requester. The objective of the problem is to maximize the aggregated quality by considering all possible execution paths and their corresponding execution probability. Each execution path ep_z is associated with a QoS vector $q_{ep_z} = [q^1_{ep_z}, q^2_{ep_z}, \ldots, q^r_{ep_z}]$ and an execution probability p_{ep_z}. The global QoS constraints define requirements regarding the aggregated QoS values of the requested composite service, which are expressed by a vector $Q_c = [Q^1_c, Q^2_c, \ldots, Q^u_c]$ $(1 \leq u \leq r)$. The service concretization problem

can be stated by (2.5).

$$\max \sum_{z=1}^{Z} p_{ep_z} * score_{ep_z} \tag{2.5}$$

subject to:

$$q^k_{ep_z} \leq Q^k_c, \quad \forall Q^k_c \in Q_c, \quad \text{k is negative attributes;}$$
$$q^k_{ep_z} \geq Q^k_c, \quad \forall Q^k_c \in Q_c, \quad \text{k is positive attributes.}$$

The variable $score_{ep_z}$ is the overall quality score of execution path ep_z and Z is the number of all possible execution paths.

The quality score of an execution path is also computed according to the SAW method, which is similar to (2.1) - (2.4). When scaling the aggregated values of QoS attributes for an execution path, the minimum and maximum values of the *k-th* QoS attribute given by (2.3) should be replaced by (2.6).

$$\begin{aligned} Q^{MIN}_k &= F(k)^n_{i=1} Q^{min}_{k,i} \\ Q^{MAX}_k &= F(k)^n_{i=1} Q^{max}_{k,i} \end{aligned} \tag{2.6}$$

The function $F(k)$ denotes the aggregation function of the *k-th* QoS attribute, which is referred to Table 2.1.

The literature presents two types of Web service concretization approaches: local optimization approaches and global optimization approaches. Figure 2.5 shows a hierarchical taxonomy of Web service concretization approaches.

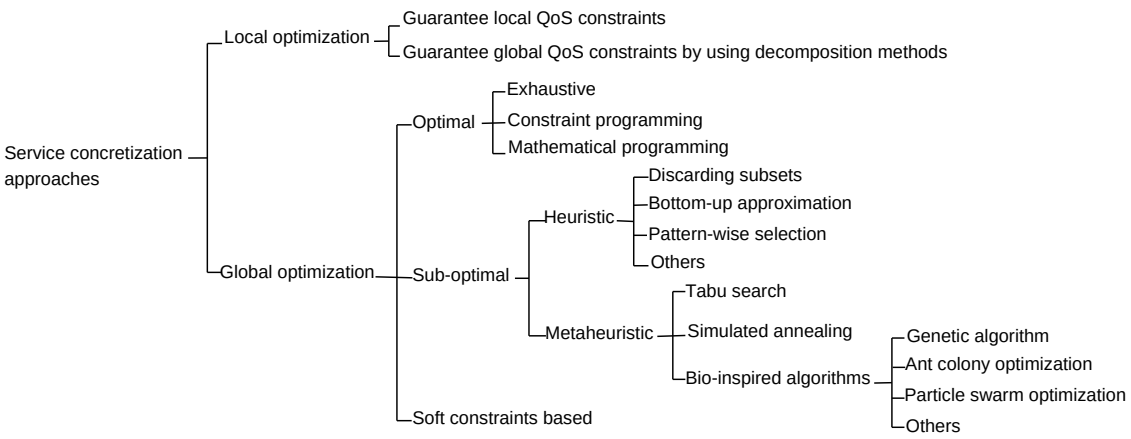


Figure 2.5: A hierarchical taxonomy of Web service concretization approaches

2.5.1 Local Optimization Approaches

In local optimization approaches, service selection is undertaken independently for each abstract service [20, 108, 134, 203]. This type of approach is very efficient in terms of computation time, as the time complexity is $O(m)$ by using a service broker, peer-to-peer, or agent-based architecture to perform selection in parallel. Even if local optimization approaches are useful in decentralized and dynamic environments, they are not suitable for QoS-based service selection with global QoS constraints, since they can only guarantee local QoS constraints and cannot satisfy the global QoS constraints.

From the viewpoint of computational time, the local optimization approaches can be appropriate when the global QoS constraints are transformed, i.e., decomposed, into local QoS constraints in order to overcome their drawbacks. A few decomposition methods have been proposed.

Alrifai et al. [8] used the concept of local quality level and modeled the decomposition problem as an optimization problem, which was formulated as a mixed integer programming model. The local quality levels were a set of discrete representative values, which were extracted from the quality properties of all service candidates. The local constraints were the selected quality levels. The decomposition problem proposed by Mardukhi et al. [122], was solved by a genetic algorithm. The authors used the concept of quality degree. The quality degrees were a set of discrete quality values, which were extracted from partitioning each quality property. The local constraints were the selected quality degrees. Sun and Zhao [164] used the mean and standard deviation of the values of QoS attributes to decompose global QoS constraints. The time complexity of the decomposition approach proposed by Alrifai et al. [8] is $O(n * c * d)$, that proposed by Mardukhi et al. [122] is $O(n * c * q)$, and that proposed by Sun and Zhao [164] is $O(n * c * m)$, where c is the number of global QoS constraints, d is the number of quality levels, and q is the number of quality degrees.

The main drawbacks of these decomposition methods are two-fold: the local selection relies on a greedy method, and the decomposition deals with QoS parameters independently so it does not take into account potential correlations and dependencies among them. Meanwhile, in these decomposition methods there is no analysis

for cases when the requirements of the global QoS constraints are over-constrained. This often leads to a very restrictive decomposition of the global QoS constraints that cannot be satisfied by any of the service candidates, even if there were feasible solutions otherwise.

2.5.2 Global Optimization Approaches

Global optimization approaches can solve Web service selection problems at both local and global service levels [12, 33, 93]. When using global optimization approaches, service selection problems have been modeled as 0-1 knapsack problems [94, 200], constraint optimization problems [70, 202], multi-dimensional and/or multi-objective optimization problems [110, 118, 206], weighted directed acyclic graph problems [137, 196], and mathematical programming problems [2, 11, 12, 36, 87, 139, 203]. Accordingly, many algorithms have been proposed. There are three types of algorithmic methods in the global optimization approaches: optimal methods, sub-optimal methods, and soft constraints-based methods.

Optimal methods, such as the exhaustive algorithm [62], constraint programming-based algorithms [90, 150], and mathematical programming-based algorithms (integer programming (IP) [2, 203], linear programming (LP) [36], or mixed integer programming (MIP) [11, 12, 139]), are designed to find optimal solutions. Ruiz-Cortés et al. [150] explained that constraint programming-based algorithms were essentially very simple and sometimes they could find a solution faster than more complex mathematical programming-based algorithms.

Sub-optimal methods, such as heuristic-based algorithms (discarding subsets [94], H1_RELAX_IP [21], shortest path heuristic [25], and other heuristics [120, 200]), and metaheuristic-based algorithms (tabu search [133], simulated annealing [64], genetic algorithms [33, 119], ant colony optimization algorithms [185, 206], particle swarm optimization algorithms [64, 207], and others [102]), are designed to find an optimal or a near-optimal solution. Metaheuristics provide both a general structure and strategy guidelines for developing a heuristic to solve service concretization problems. They provide an efficient way to move towards a very good solution, though it might not necessarily be the best one.

Soft constraints-based algorithms are adopted in order to allow constraint vio-

lations in less likely feasible compositions when the requirements of the global QoS constraints are overly strict [70, 76, 135, 202].

2.5.3 Analysis of the Existing Web Service Concretization Approaches

The Web service concretization approaches should be evaluated with respect to their optimality, their computational efficiency, and their dynamic complexity. The optimality is the extent to which the approach finds the best solution. The computational efficiency is measured by the time that the approach takes to find a solution. The dynamic complexity is referred to the time complexity that the approach deals with the re-optimization. By analyzing the existing service concretization approaches with respect to the above three aspects, the author observed that:

1. If there is no requirement specifying global constraints, the local optimization approaches are preferable.

2. Optimal methods, such as mathematical programming-based algorithms, which find optimal solutions, are often practical for small scale problems. When the time requirements become excessive, sub-optimal methods must be employed. Continuing efforts have demonstrated that sub-optimal methods for QoS-aware service composition usually require much less computation time than optimal methods, and they are consistently capable of achieving near-optimal solutions [21, 25, 45, 62, 92, 94].

3. The optimal methods need more computation time, especially in dynamic environments. This is exacerbated in a situation where the QoS-aware composition has to be re-planned at runtime because the computation time of the composition becomes crucial. This was confirmed by the existing studies [33, 62, 64, 200, 203].

4. IP, LP and MIP-based algorithms need linear aggregation functions for the QoS attributes of services. Some standard attributes such as availability and reliability are not linear initially. While linearization can be adopted [203], it is quite difficult in other cases, such as computing the response time for the paral-

lel structures. An alternative would be the use of non-linear IP, however, scalability problems would still arise [33]. When using constraint programming-based algorithms, the constraints can be expressed without translating them into linear inequalities [150].

5. In the case of the sub-optimal methods, for example, heuristic or metaheuristic, there is a trade-off between the computation cost and the quality of the solution. That is to say, how much worse the sub-optimal methods would be when compared with an optimal method that always finds the optimal solution.

6. Soft constraints-based methods are proposed to predict run-time service level agreement violations for composite services. They work around the over-constrained QoS-aware service composition problem and offer a feasible solution. The choice of evaluation functions, the definition of penalties, and the relation between the assigned penalties and the violated guarantees, are key factors in this type of methods.

The QoS-aware Web service composition problem is well known as a NP-hard problem with no known polynomial-time algorithms to solve them [92]. Besides, the composition can be either static or dynamic. In the static composition, all services are available before the composition process. No runtime reconfigurations are possible once the composition process is launched. The static composition is not concerned about the complexity but the focus is more on the accuracy or optimality of the approach. The dynamic composition is more concerned about the complexity. In the dynamic composition, physical nodes, services, and service implementations can be replaced or reconfigured, all of these lead QoS attributes of services to change. Thus, the service composition has to be re-planned.

2.6 Summary

The existing QoS-based Web service concretization approaches can be categorized into two groups: local optimization approaches and global optimization approaches. The local optimization approaches cannot solve service selection problems with

global QoS constraints, but they can be used by exploiting decomposition methods. The global optimization approaches construct composite services from a global level to satisfy the QoS constraints. Accordingly, several optimal and sub-optimal algorithms have been proposed, and in particular, metaheuristic-based algorithms intend to find near-optimal solutions. The sub-optimal algorithms demonstrate good scalability and can decrease the computational complexity when compared with the optimal algorithms.

Based on the detailed investigation conducted on Web service concretization approaches, the Web service concretization problem based on bio-inspired algorithms has been reviewed and is discussed in the next chapter.

Chapter 3

A Systematic Review of Bio-Inspired Service Concretization

This chapter investigates the existing studies using bio-inspired algorithms to deal with QoS attributes, optimality, and dynamicity in Web service concretization problems. Although numerous research studies have addressed Web service concretization problems (including bio-inspired algorithms), service discovery and composition mechanisms will face new challenges as cloud computing, big data, and the next generation Internet change the whole scene. The author believes that applying biologically inspired approaches to services can make them adapt to dynamic service environments. For example, Balasubramaniam et al. [15] already proved that it was useful for service management and discovery to add biological mechanisms to services. The key attributes of biological systems are given in Section 3.1. Then Section 3.2 describes the method of the systematic review on literature in this area, while Section 3.3 presents the results.

3.1 Key Attributes of Biological Systems

Biological systems present features such as autonomy, scalability, adaptability, and robustness. They are autonomous entities and often self-organized without a central controller. Typical life cycles and behaviors of biological organisms include: birth

and death, migration, and replication (or division) of a group of organisms [15]. The biological environment provides a medium that allows biological organisms to interact and mobilize. For example, ants release chemical signals, creating chemical gradients in the environment after finding a good path from the nest to a food source, so that letting other ants sense the chemical and follow the path to food.

Nakano [128] pointed out four key attributes of biological systems, which can be applied to the design of biological inspired systems.

1. A large number of redundant components: a biological system is composed of massive numbers of redundant components. For example, an ant colony may contain millions of ants. A biological system is robust under perturbations or conditions of uncertainty because of the large number of redundant components.

2. Local interactions and collective behavior: an individual ant is not able to find the shortest path to food source, but a group of ants can find the shortest path via interactions among individual ants through the placement of pheromones.

3. Stochastic or probabilistic nature: the probabilistic nature of biological systems helps them explore large search space. For example, mutation in genetic evolution randomly changes genetic information to enable individuals to adapt to the environment. Ants find the shortest path via the probability of choosing different paths.

4. Feedback-based control: biological systems frequently use positive and negative feedback control. For example, ants use positive feedback to deposit the pheromone trails and to recruit more ants towards a particular path.

These characteristics will be discussed in the following sections in three biological systems, namely, insect colonies, genetic evolution, and swarms. Three bio-inspired algorithms are considered: ant colony optimization (ACO) algorithms, genetic algorithms (GA), and particle swarm optimization (PSO) algorithms. It is both important and interesting to know the extent of applications of bio-inspired algorithms to QoS-based Web service concretization problems, which is not covered by previous studies. This effort allows us to provide an overview on the existing studies.

3.2 Method of Systematic Review

This section describes the systematic review protocol, consisting of several steps, as outlined by Kitchenham and Charters [100].

3.2.1 Research Questions

In order to find principles for applying bio-inspired algorithms to solve Web service concretization problems, the following research questions need to be answered:

RQ 1. How are the three types of bio-inspired algorithms applied to solve Web service concretization problems?
After knowing this, three additional research questions are applicable in each type of algorithm:

RQ 2. What are the main issues that need to be addressed if the bio-inspired algorithms are to be applied?

RQ 3. What are the strategies proposed to address the issues?

RQ 4. What are the current challenges or limitations in the application of the bio-inspired algorithms to solve Web service concretization problems?

The domain of the review is the Web service composition and selection based on ant colony optimization algorithms, genetic algorithms, and particle swarm optimization algorithms. The research questions are not aiming at making a comparison among the three algorithms, but the comparisons within the scope of each primary study will be discussed to support the argument from the obtained results.

3.2.2 Specification for Literature Review Strategy

The following terms were used to search the literature:

1. Web service composition, Web service selection.

2. Ant colony algorithm, ant colony optimization, ant-inspired, ant system, ant-based, genetic algorithm, particle swarm optimization, particle swarm algorithm.

Table 3.1: Distribution of papers before and after duplication removal among different publication sources

Source	Count
ACM Digital Library	10(10)
ScienceDirect	11(11)
Google Scholar	55(54)
IEEE Xplore	154(120)
EI Compendex	224(83)
Total	454(278)

Five electronic databases were used, namely, ACM Digital Library, ScienceDirect, Google Scholar, IEEE Xplore, and EI Compendex as data sources, and papers were selected for review if they had a key term (or synonym) in the title, abstract or keywords list. 2005 is selected as the starting year for the search since this year marked the first publication of the application of genetic algorithm to Web service composition, and the ending year is 2012. The search resulted in a total of 454 papers. After eliminating duplicates found by more than one electronic database, 278 papers were left. Table 3.1 shows the distribution of papers before and after removal of duplications among different sources.

The following exclusion criteria are applicable in this review to exclude studies that:

1. Do not relate to Web service selection and composition.

2. Do not report application of bio-inspired algorithms.

3. Do not report experiments and simulations.

4. Are related to semantic Web service composition and selection.

The application of detailed exclusion criteria resulted in 120 remaining references, which were further filtered out by reading full-text. A final figure of 22 primary studies was reached after excluding similar studies that were published in different venues. Relevant information describing the distribution of primary studies within the four QoS attributes is shown in Table 3.2. The four QoS attributes which are the most commonly used are: response time, cost, reliability, and availability,

Table 3.2: Distribution of primary studies

Bio-inspired algorithm	Articles	Year	Cost (C)	Time (T)	Availability (Av)	Reliability (Re)
ACO	[208]	2007	∨	∨	∨	
	[192]	2008	∨	∨	∨	∨
	[174]	2010	∨	∨		∨
	[206]	2010	∨	∨	∨	∨
	[185]	2011	∨	∨	∨	∨
GA	[33]	2005	∨	∨	∨	∨
	[35]	2005	∨			
	[205]	2006	∨	∨	∨	∨
	[75]	2007	∨	∨	∨	∨
	[3]	2008	∨	∨	∨	∨
	[133]	2008	∨	∨		∨
	[119]	2008	∨	∨	∨	∨
	[114]	2009	∨	∨	∨	∨
	[167]	2010	∨	∨	∨	∨
	[173]	2012		∨	∨	∨
PSO	[65]	2009				
	[146]	2010	∨	∨		
	[183]	2012	∨	∨	∨	
	[207]	2012	∨	∨	∨	∨
Combination of GA and PSO	[117]	2007				
Combination of ACO and GA	[196]	2010	∨	∨		
Combination of PSO and SA	[64]	2011				

and according to Jaeger and Muehl [93], they represent a selection of the most relevant characteristics in the field of services. Response time, run time, and execution time might have the same or different definitions, and Table 3.2 places them into the single time column. In addition, the studies [64, 65, 117] did not provide details about whether the four QoS attributes were considered or not.

3.3 Results and Synthesis of Findings

3.3.1 Ant Colony Optimization Algorithms

The ant colony optimization algorithm modeling the behavior of real ants is widely used for solving combinatorial optimization problems. The features of ACO algorithms include positive feedback and local heuristics. When ACO algorithms are

used to solve a Web service concretization problem, the problem is modeled as a weighted directed acyclic graph with a start vertex and an end vertex. Then the start vertex is set as the ants' nest and the end vertex is set as the food source, and the QoS attributes are regarded as the weights of the edges. Thus, the problem is transformed from finding feasible solutions to the optimization problem into selecting optimal paths through the weighted graph. Figure 3.1 describes a service composition graph when using ACO algorithms. In the graph, all ants are initially positioned at the start vertex and the task of each ant is to find a path from the start vertex to the end vertex. The whole ant colony's foraging behavior creates optimal paths from the start vertex to the end vertex. The details of a simple ACO algorithm, which adapts the real ants' behavior to the problem of optimal paths through a weighted graph, are shown as follows.

Given a graph G=<V, E, QoS>, two vertices $i, j \in V$ are neighbors if there exists an arc $(i, j) \in E$. Beginning with the start vertex, each ant builds up a solution iteratively by always selecting the next vertex based on pheromone trails and heuristic information. The pheromone trail which is denoted by τ_{ij}, and the heuristic information which is denoted by η_{ij}, are indicators of tendency to move from vertex i to vertex j. All edges of G are initialized with a certain amount of pheromone at the beginning of the search process, i.e., $\tau_{ij} = \tau_0$ (τ_0 is a constant, $\forall(i, j) \in E$). In the process of searching, when located at a vertex i, each ant k

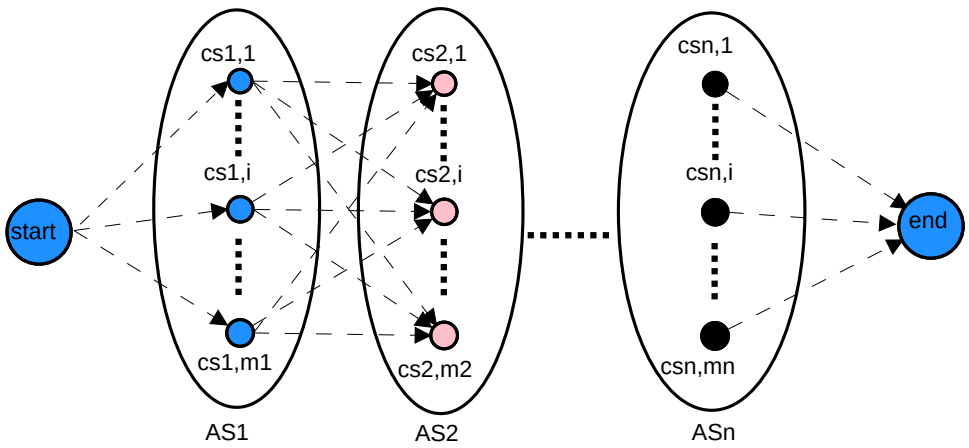


Figure 3.1: A service composition graph when using an ACO algorithm

chooses the next vertex j according to the probability distribution defined by (3.1).

$$p_{ij}^k = \begin{cases} \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta}, & \text{if } j \in N_i^k \ ; \\ 0, & \text{otherwise} \ . \end{cases} \tag{3.1}$$

The parameters α and β control the influence of pheromone value and heuristic value. The variable N_i^k is used to denote the set of unvisited vertices, which contains all the direct successors of vertex i in the graph G. Each ant changes the pheromone value τ_{ij} according to (3.2).

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \Delta\tau^k, \quad \forall (i, j) \in E. \tag{3.2}$$

The variable ρ is the pheromone evaporation coefficient, and $\Delta\tau^k$ is the amount of pheromone deposited by ant k. The choice of $\Delta\tau^k$ is an important aspect of the algorithm. It can be the same constant value for all the ants. Generally, it is required that the amount of pheromone deposited by an ant to be a non-increasing function of the path length [55].

An ACO algorithm iteratively performs a loop containing two basic procedures: one is how the ants construct solutions to the problem, and the other is how to update the pheromone trails [157]. When using ACO algorithms to solve the combinatorial optimization problem, it requires a representation of the problem and the definition of the pheromone information as well as the heuristic information. Then a specific ACO algorithm is chosen and the related parameters should be tuned. The basic operational flow in an ACO algorithm is as follows.

1. *Initialize ACO parameters;*
2. *Generate random solutions from each ant's random walk;*
3. *Update pheromone values;*
4. *Steps 2 to 3 are repeated until a termination condition has been satisfied.*

A summary of results on applying ACO algorithms to Web service concretization problems is given in Table 3.3. Besides the listed items, the references in Table 3.3

Table 3.3: A summary of results on applying ACO algorithms to Web service concretization problems

Articles	Heuristic information used	Pheromone updating was applied to	Key parameters	Limitations and highlights
[208]	C, T, Av	the best solution	$\alpha = 1$, $\beta = 2$, $\rho = 0.05$	Effects of the number of abstract services are not taken into account. Further investigation is required to deal with all situations of clone ants.
[192]	C, T, Av, Re	the most satisfying solution	$\alpha = 2$, $\beta = 2$	The setting of parameter is not given and further performance evaluation is required.
[174]	C, T, Re	the solution in known Pareto front	$\alpha = 1$, $\beta = 1$, $\rho = 0.3$	There is a need to use different metrics to measure the performance of multi-objective optimization.
[206]	C, T, Av, Re	all the solutions	$\alpha \in [2, 3, 4, 5]$, $\beta \in [4, 8, 9]$, $\rho = 0.7$	All the abstract execution paths are not consistent with the definition, especially those in the experimental part. The optimal of abstract execution paths cannot guarantee the global QoS constraints of a composite service.
[185]	C, T, Re, Av, and five domain QoS attributes	the top solutions and other remaining solutions used different updating rule	$\alpha = 1$, $\beta = 4$, $\rho = 0.5$	Effects of the number of abstract services and the number of concrete services are not taken into account. There is a need to devise different scenarios to test scalability.

differ from each other with respect to the single objective ACO algorithm which they were based on, such as the ant colony system adopted by Zheng et al. [208], the ant system adopted by Wang and He [174], Xia et al. [192], Zhang et al. [206], and the max-min ant system adopted by Wang et al. [185].

Zheng et al. [208] presented a utility function to measure the users' satisfaction. The method was compared with an exhaustive searching method, and a test scenario with nine abstract services was used in the evaluation. The simulation results showed that the execution time of the proposed algorithm was less than that of the exhaustive searching method.

Xia et al. [192] used different pheromones to denote different QoS attributes. The simulation used eight abstract services, and the number of concrete services corresponding to each abstract service was randomly selected from 0 to 25. The experimental results showed that the dynamic ACO algorithm had better performance than the typical ACO algorithm.

Wang and He [174] modeled the Web service selection problem as a multi-

objective optimization problem, and proposed a multi-objective chaos ACO algorithm to solve it. The chaos variable was used to improve the efficiency of the ACO algorithm. To evaluate the proposed algorithm, the authors compared it with a multi-objective GA and a multi-objective ACO algorithm based on three test groups. The evaluation criteria were the number of optimal solutions and the running time. The simulation indicated that the multi-objective chaos ACO algorithm was able to find more optimal solutions and used less time than the multi-objective GA and the multi-objective ACO algorithm.

Zhang et al. [206] used a k-tuple pheromone to represent k objectives. A strategy was adopted to decompose a composite service with a general composition structure into parallel execution paths. The experimental results showed that the proposed multi-objective ACO algorithm could find near-optimal solutions.

Wang et al. [185] integrated the max-min ant system into the framework of culture algorithm to solve the Web service selection problem. A comprehensive evaluation model based on generic QoS attributes and domain QoS attributes was designed. The generic QoS model was used to evaluate the QoS attributes of composite services, and the domain QoS model was used to conquer the over-constrained problem. A scenario with 10 abstract services and 50 concrete services for each abstract service was used to test the performance of the proposed algorithm. The experimental results showed that the solutions found by the proposed algorithm were much better than that of an ACO algorithm and a max-min ant system.

3.3.2 Genetic Algorithms

Genetic algorithms belong to the larger class of evolutionary algorithms, which generate approximate solutions to optimization and search problems using techniques inspired by the principles of natural evolution: selection, crossover, and mutation. The genetic algorithm is a powerful tool to solve combinatorial optimization problems [159]. It is an iterative procedure based on a constant-size population. In a genetic algorithm, a population of candidate solutions to an optimization problem is evolved towards better solutions.

Figure 3.2 shows an example, which uses an integer array to encode the genome. Each genome is associated with a fitness value based on a fitness function that

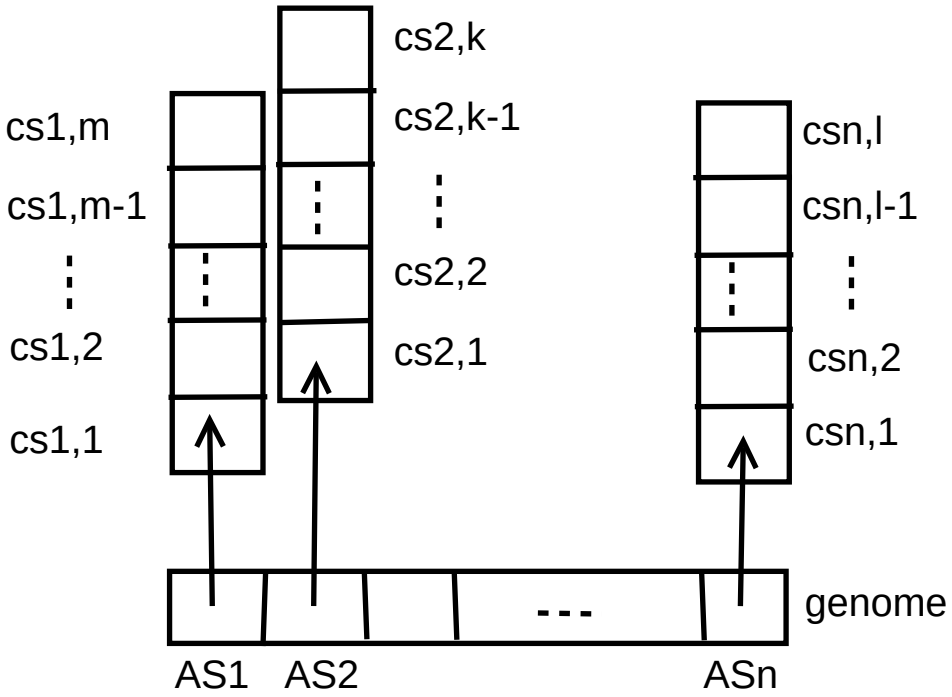


Figure 3.2: An example of how the genome is encoded when using a GA

indicates how close it comes to meeting the overall requirement, when compared to other genomes in the population. The fitness value of a genome is also an indication of its chances of survival and reproduction in the next generation. The fitness can be measured in a variety of ways: a distance, an error, or a time interval. When using GA to solve Web service concretization problems, the fitness function always corresponds to QoS attributes. An example of fitness function for a genome g is given by (3.3).

$$F(g) = \frac{w_1 * C(g) + w_2 * T(g)}{w_3 * Av(g) + w_4 * Re(g)} + w_5 * pf \tag{3.3}$$

The QoS attributes (C(g), T(g), Av(g), Re(g)) are normalized in the interval $[0, 1]$. The positive variables $w_1, \ldots, w_5$ represent weighting factors. In particular, $w_1, \ldots, w_4$ indicate the importance of each QoS attribute while w_5 weights the penalty factor pf. The penalty factor for an individual is defined as the distance between the QoS attributes of the individual and the constraints.

A typical GA requires definition of two things: one is a genetic representation of the solution domain, and the other is a fitness function to evaluate the solution domain. Once the genetic representation and the fitness function are defined, a GA executes five steps:

1. *Initialization. Traditionally, the population is formed by a group of randomly generated individuals.*

2. *Evaluation. The fitness of each individual in the population is evaluated.*

3. *Selection. Individuals are selected based on the fitness value to breed a new generation.*

4. *Evolution. New individuals are created through crossover and mutation operations. The new population is composed of the individuals in the new generation and a few individuals from the previous generation.*

5. *Termination. Steps 2 to 4 are repeated until a termination condition has been reached. The termination conditions [191] are:*

   - *A solution that satisfies the specified criteria is found.*
   - *A fixed number of generations is reached.*
   - *The allocated budget (computation time/money) is reached.*
   - *The highest ranking solution's fitness is reaching or has reached a plateau such that successive iterations no longer produce better results.*
   - *Manual inspection.*
   - *Combinations of the above conditions.*

A summary of results on applying GAs to Web service concretization problems is given in Table 3.4.

Canfora et al. [33] proposed a GA with static and dynamic penalty strategies in the fitness function. The experimental results showed that the GA was preferred when a large number of concrete services were available for each abstract service. When the number of concrete services was in a small scale, the integer programming algorithm would be preferred.

Cao et al. [35] presented a GA for cost-driven Web service selection, which only considered cost of concrete services. A scenario with 20 abstract services was used to test the performance of the GA. The experimental results showed that the GA could work and get better performance when compared with a local service selection method.

Table 3.4: A summary of results on applying GAs to Web service concretization problems

Articles	Fitness function used	Encoding scheme	Selection operator	Crossover operator	Mutation operator	Limitations and highlights
[33]	C, T, Re, Av, the current generation and the maximum number of generations	integer array coding	roulette wheel selection and elitism selection	two points crossover, probability was 0.7	randomly selected a gene and randomly replaced with another one, probability was 0.01	The randomly replacement in mutation operator is not efficient.
[35]	C, and a constant	integer array coding	roulette wheel selection and elitism selection	single point crossover, probability was 0.8	The probability of mutation was for the locus of chromosome, mutation probability was 0.05	It is difficult to decide the value of the constant in the fitness function. Further investigation is required to design the fitness function and other QoS attributes should be considered. The experiments are too simple.
[205]	C, T, Re, Av	relation matrix coding	roulette wheel selection	special crossover operator, probability was 0.7	randomly selected a gene in each chromosome and mutated, probability was 0.1	The crossover and mutation operators may generate infeasible individuals frequently, and the proposed algorithm does not treat distributed service deployment.
[75]	C, T, Re, Av, and a penalty item	tree-coding	roulette wheel selection	single point crossover, probability was 0.7	randomly selected a gene in each chromosome and mutated, probability was 0.01	Each gene has to record the information of its children and father, which makes the length of chromosomes longer and more complicated.
[3]	C, T, Re, Av, and reputation	integer array coding	rank-based selection and elitism selection	single point crossover, probability was 0.9	randomly selected a gene in each chromosome and mutated, probability was 0.08	The repairing operator randomly selects and assigns values, the efficiency is low and it cannot guarantee the finding of a feasible solution in the maximal repairing times.
[133]	C, T, Re, reputation, security, and a penalty item	integer array coding	tournament selection	two points crossover, probability was 0.7	randomly selected a gene in each chromosome and mutated, probability was 0.2	The hybrid genetic algorithm is based on exploration of the neighborhood, and the percentage of the neighborhood is a key factor to determine.
[119]	C, T, Re, Av, and a penalty item	relation matrix coding	roulette wheel selection	special crossover operator, probability was 0.7	randomly selected a gene in each chromosome and mutated, probability was 0.1	More scenarios are required to test the proposed algorithm. The investigation is required to escape from frequently created infeasible individuals by the crossover and mutation operator.
[114]	C, T, Re, Av, and reputation	not provided	not provided	constrained single point crossover, probability was 0.8	the selection of mutation focused on one specific gene of the entire chromosome, probability was 0.001	The proposed algorithm is required to test more scenarios and further comparisons are needed.
[167]	C, T, Re, Av, total number of constraint violations in a solution, and the maximal number of possible constraint violations	integer array coding	roulette wheel selection	a knowledge-based one, probability was 0.9	randomly selected a gene in each chromosome and mutated, probability was 0.15	The performance is not stable, further improvement is required to modify the local optimizer.
[173]	C, T, Re, Av, the reversed index number of qualitative pattern, and some violation value with respect to quantitative constraints	binary string coding	according to a defined probability to select individuals	uniform crossover	randomly chose a single bit of a string to mutate	The structure of execution plan is sequential flow. Large number of service candidates leads to long length of the gnome, and the readability of the chromosomes is fairly weak.

Zhang et al. [205] conducted an initial population policy and a mutation policy to direct the evolution of genetic algorithms. The performance was evaluated by comparing a GA using the relation matrix coding scheme with a GA using the one-dimension coding scheme, comparing a GA using the initial population policy with a GA using the randomly created population, and comparing a GA using the proposed mutation policy with a GA using other policies. The experimental results showed that genetic algorithms using the proposed policies could get more excellent solutions than standard genetic algorithms.

Gao et al. [75] investigated a novel tree-coding GA for QoS-aware service composition. The tree-coding GA could simultaneously express multiple types of composite relationships and could re-plan process at runtime. The performance was evaluated by comparing the tree-coding GA with a one-dimensional coding GA. The experimental results showed that the tree-coding GA was effective and faster than the one-dimensional coding GA.

Ai and Tang [3] designed a repair GA to address the Web service composition problem in the presence of domain constraints and inter service dependencies. The proposed GA used a repair operator to fix up the infeasible individuals in order to make all the individuals in the population be always feasible. The repair operator was based on an iterative heuristic approach, which randomly selected a gene and randomly chose a value from the set of minimal conflict values for the selected gene. The performance was evaluated by testing the effectiveness and scalability of the proposed repair GA, and compared it with a GA without the repair operator. The experimental results showed that the repair GA was scalable and effective.

Parejo et al. [133] proposed a hybrid GA by using a local improvement procedure, which was based on an iterative neighborhood search. The hybrid GA was compared with a standard GA. The experimental results showed that the hybrid GA performed better than the standard GA for small and medium problem sizes.

Ma and Zhang [119] adopted an enhanced initial population policy and an evolution policy to improve the convergence of the GA. An expectation value was used to control the population diversity. The performance was evaluated with regard to the coding scheme, the population diversity, the enhanced initial population, and the evolution policy. The GA with the proposed policies was compared with a stan-

dard GA. The experimental results showed that the proposed GA could improve the fitness value and promote the convergence rate.

Liang and Huang [114] incorporated a GA with the rough set theory, and exemplified by solving the Web service composition problem. The proposed method was evaluated by comparing with a standard GA and an exhaustive enumeration method. The experimental results indicated that the execution time of the proposed GA was not limited to the number of abstract services, and its convergence rate and hit rate were much better than that of the other two methods.

Tang and Ai [167] considered the dependency constraint and conflict constraint between Web services in their hybrid GA. A local optimizer was used to improve the individuals in the initial population. The proposed GA was compared with a penalty-based GA and a repairing-based GA. Three sets of scenarios with different number of abstract services, concrete services, and constraints were tested. The experimental results showed that the hybrid GA was as scalable as the other two genetic algorithms, and could find better solutions. The computation time of the hybrid GA did not change as the number of concrete services increased, and linearly increased as the number of abstract services or the number of constraints increased.

Wang et al. [173] showed a service composition model considering quantitative and qualitative non-functional properties. A global optimization and a GA were proposed to conduct the service composition. The performance was evaluated by using a public available dataset and a synthetic dataset. The experimental results showed that the quality of the GA was better than that of the global optimization in most cases.

3.3.3 Particle Swarm Optimization Algorithms

Particle swarm optimization is one of the evolutionary computational techniques. It has been widely used to solve optimization problems because it is very robust, it has a small number of parameters, and it is simple and easy to implement. The PSO algorithm finds the optimal solution through iterations after initializing a group of

random particles. The particles update their position and speed according to (3.4).

$$\begin{aligned} x_{id}(t+1) &= x_{id}(t) + v_{id}(t+1) \\ v_{id}(t+1) &= \omega v_{id}(t) + c_1\gamma_1(P_{id}(t) - x_{id}(t)) + c_2\gamma_2(P_{gd}(t) - x_{id}(t)) \end{aligned} \tag{3.4}$$

The variables x_{id} and v_{id} are the position and speed of the d-dimensional particle i, and t means the *t-th* generation. The variable ω is the inertia weight that controls the exploration and exploitation of the search space. γ_1 and γ_2 are two mutually independent random functions. The variables c_1 (cognition coefficient) and c_2 (social coefficient) are the acceleration constants, which change the velocity of particle i towards P_{id} and P_{gd} [157]. The variable P_{id} is an individual best position, and the variable P_{gd} is a global best position. The basic operation of a PSO algorithm is as follows.

1. *Initialization. The swarm population is formed.*
2. *Evaluation. The fitness of each particle is evaluated.*
3. *Modification. The best position of each particle, the velocity of each particle, and the best position of the whole swarm are modified.*
4. *Update. Each particle is moved to a new position.*
5. *Termination. Steps 2 to 4 are repeated until a termination condition has been satisfied.*

A summary of results on applying PSO algorithms to Web service concretization problems is given in Table 3.5.

Fan et al. [65] compared a multi-objective discrete PSO algorithm with an exhaustive method. The experimental results showed that the PSO algorithm could get lower computation cost and higher quality solutions.

Rezaie et al. [146] compared a multi-objective discrete PSO algorithm with a multi-objective GA. The experimental results showed that the proposed PSO algorithm used more execution time and created more Pareto solutions than the multi-objective GA.

Wang et al. [183] designed a non-uniform mutation strategy, an adaptive weight adjustment strategy, and a local best first strategy for a PSO algorithm. These

Table 3.5: A summary of results on applying PSO algorithms to Web service concretization problems

Articles	Fitness function used	The presentation of particle	Key parameter	Limitations and highlights
[65]	did not provide	N-tuple	ω was computed according to constraints	The experiments cannot show the efficiency of the proposed algorithm, and the fitness function and some parameters are not provided.
[146]	C, T	N-dimensional vector	$\omega = 0.0275$, $c_1 = 0.3$, $c_2 = 0.49$	Further performance evaluation is required, especially various performance metrics are needed to measure performance of multi-objective optimization algorithms.
[183]	C, T, Av, and reputation	N-dimensional vector	ω, c_1 and c_2 were self-adaptive	The adaptive termination of algorithms is not considered.
[207]	C, T, Re, Av	N-dimensional vector	ω, c_1 and c_2 were self-adaptive	Further experiments are required to make sure the results are not biased by the QoS attributes of concrete services.

strategies were used to enhance the population diversity and to improve the convergence rate. The performance of each strategy was evaluated. The experimental results indicated that the proposed strategies could improve the convergence rate and fitness values.

Zhao et al. [207] proposed an immune optimization algorithm based on a PSO algorithm. An improved local best strategy and a perturbing global best strategy were also discussed. The proposed algorithm was compared with a standard PSO algorithm and a genetic algorithm. The mean and standard deviation of the optimal fitness values were computed. The experimental results showed that the proposed algorithm was better than the other two algorithms in terms of the search ability, the convergence rate, and the stability.

3.3.4 Combined Bio-Inspired Algorithms

A summary of results on applying combined bio-inspired algorithms to Web service concretization problems is given in Table 3.6.

Liu et al. [117] designed a hybrid genetic particle swarm algorithm to balance the iterations of the global optimization and the local optimization. The proposed algorithm was compared with a balanced multi-objective GA. The experimental results indicated that the hybrid algorithm improved the solutions and enhanced the execution efficiency.

Table 3.6: A summary of results on applying combined bio-inspired algorithms to Web service concretization problems

Articles	Fitness function used	Reason(s) for combining	Limitations and highlights
[117]	did not provide details	The GA was used to search the problem space, and the PSO was used to enhance the local search ability.	There is a need to show more details of the experiments, and find most appropriate and robust parameters for testing.
[196]	C, T	The GA was used to set the key parameters of ACO	The experiment cannot indicate the performance of the algorithms. The QoS dataset is not provided. There is a need to design more scenarios for performance testing.
[64]	did not provide details	The swarm in a stochastic PSO was often not convergent to the global optimal position but a local best position, and a SA was convergent to the optimum but its convergent speed was slow.	The parameters of the proposed algorithm and the QoS dataset are not provided. Further investigations are required to test the performance of the proposed algorithms in terms of the changing number of abstract services and concrete services.

Yang et al. [196] presented a dynamic algorithm by combining an ACO algorithm and a GA. The proposed algorithm was compared with a genetic algorithm and an immune algorithm. The test scenario had 10 abstract services and each abstract service had 35 concrete services. The experimental results showed that the proposed algorithm had better performance than the other two algorithms in terms of the optimal solutions and the iteration numbers.

Fan et al. [64] proposed a cooperative evolution algorithm that consisted of a stochastic PSO and a simulated annealing (SA). The stochastic PSO was used to rapidly search the local optimum, while the SA was adopted to make the individual jump off the local optimum. The cooperative algorithm was compared with a stochastic PSO and a SA in terms of the quality of solution, the computation complexity, and the convergence rate. The experimental results indicated that the cooperative algorithm was better than the other two.

3.3.5 Synthesis of Findings

A variety of studies suggest that the parameters of bio-inspired algorithms have a strong impact on the performance of the service concretization. They argue that inept setting of parameters may result in bad performance, but only a few of the existing studies [183, 185, 206, 207] gave an explicit investigation about the influence of parameters. For example, the design of a GA has a great influence on its behavior

Table 3.7: Main control parameters in three bio-inspired algorithms

Algorithm	Main control parameters
ACO	number of ants
	number of iterations
	influence of heuristic (β)
	influence of pheromone (α)
	pheromone evaporation coefficient (ρ)
	definition of heuristic factor (η)
	pheromone updating strategy
GA	population size
	number of generations
	encoding scheme
	selection operator
	crossover operator
	crossover probability
	mutation operator
	mutation probability
	definition of fitness function
PSO	population size
	inertia weight (ω)
	number of generations
	particle initial position
	particle initial velocity
	acceleration factor (c_1 and c_2)
	definition of fitness function

and performance [148]. It is necessary for a GA to accord with the characteristic of the service concretization in order to get better performance. By reviewing a number of studies on bio-inspired algorithms-based Web service concretization, the main control parameters involved in each type of algorithm are summarized in Table 3.7.

The ACO algorithm focuses on theoretical problems, such as parameter initialization and information updating. It can be run continuously and is generally capable of adapting to changes of an optimization problem. It may have some advantages over the PSO algorithm and the GA in dynamic environments. On the other hand, the efficiency of the ACO algorithm is somewhat low. In the GA, the

mutation operator and the crossover operator are relatively fixed, and the random search without guide will cause degeneration. The ACO and PSO algorithms share many common points with the GA, such as a randomly generated population, and a fitness function to evaluate individuals. Unlike a GA, however, ACO and PSO algorithms have no evolution operators such as the crossover operator and the mutation operator. In addition, the PSO algorithm has a few more parameters to adjust and it is easy to implement. All these bio-inspired algorithms have the problem of premature convergence, and they tend to be easily trapped into the local optimality. So many efforts have been made to improve their performance by combining them.

Normally, the Web service concretization needs to be performed and completed in a short time period, and the quality of solutions needs to fully satisfy service requesters' requirements. Many studies into Web service concretization based on bio-inspired algorithms are attempting to reduce the processing time for a near-optimal solution, to improve the quality of solutions, and in particular, to avoid being trapped into local optimization. Some studies have carried out comparisons among different versions of the same type of algorithms, such as comparing an ACO algorithm with a chaos ACO algorithm [174], or comparing an improved GA with a standard GA [3, 75, 114, 119, 133, 167, 205]. Other studies have compared different types of algorithms, such as comparing the combination of an ACO algorithm and a GA with a standard GA [196].

3.4 Summary

In this chapter, the Web service concretization based on bio-inspired algorithms was discussed. The ant colony optimization algorithms, the genetic algorithms, and the particle swarm optimization algorithms had been described in detail. While some progresses have been made in recent years in the area of bio-inspired Web service concretization, there remains much potential for further efforts to be made in this area.

One future challenge is to establish a standard framework for designing and testing bio-inspired Web service concretization. Currently, there is no such framework available. The existing studies presenting comparisons among bio-inspired

algorithms had limited themselves to their own simulated environment and test data set. Their experimental results represented only a small number of specific scenarios. We cannot tell which algorithm performs better in a certain scenario. Guidelines are also needed for service developers to choose the optimal algorithm to offer composite services, which could enable the service requesters to get better performance based on their own requirements and preferences.

A second major challenge is to explore key features and mechanisms of biological systems to be applied in cloud computing and big data areas. Due to the deluge of enormous sources of data and the development of cloud computing, applications based on data-intensive services have become one of the most challenging applications in service oriented computing. Data access, data analysis, and data manipulation may require a nontrivial composition of a series of data retrieval and process executions. The service-based strategy provides maximal flexibility when designing data-intensive applications. These data services are different from traditional services because they handle huge amount of data, and their QoS attributes might be quite different. Because these data-intensive services are normally on a larger scale and are usually more complex, the author will attest that bio-inspired mechanisms would also facilitate the data-intensive service provision. The applications of the ant colony optimization algorithm and the genetic algorithm to solve the cost-aware service provision problem are discussed in the next chapter.

Chapter 4

Cost-Aware Data-Intensive Service Provision

This chapter introduces the comprehensive applications of an ant colony optimization algorithm and a genetic algorithm to the data-intensive service provision from the perspective of costs. Due to the explosion in digital data, the distributed nature of cloud computing, as well as the large increase of providers to the market, providing efficient cost models for composing data-intensive services will become central to this dynamic market. The location of users, service composers, service providers, and data providers will affect the total cost of service provision. Different providers need to make decisions about how to price and pay for resources. Each of them wants to have a good market position maximizing profit. An economic model is proposed for the data-intensive service provision. An extensible QoS model mainly considering the cost is also presented to calculate the QoS attributes of data-intensive services. Then an ant colony system and a genetic algorithm are applied to compose data-intensive services. Finally, the performance of the two proposed algorithms is compared.

4.1 Introduction

Chapter 2 presents a hierarchical taxonomy of Web service concretization approaches and provides a detailed analysis of each approach. Then the author found that bio-inspired algorithms could overcome the challenging requirements of the data-

intensive service provision. It is useful for the provision of data-intensive services to explore key features and mechanisms of biological systems and to add biological mechanisms to services. For example, to solve a complex scientific problem in the AMS experiment, such as the massive data mining, massive data query, and the physical analysis, scientists need to combine data from various sources. Such a data-intensive process is usually composed as a workflow of various data-intensive services using Web service technologies. The cost and response time of each service are critical for the composite service. The data-intensive service provision has the following challenges.

1. The size and the number of distributed data sets increase the communication and storage costs, which affect the performance of the whole data-intensive application.

2. Large numbers of data sets and increases of functionally equivalent services make the composition become complex. Furthermore, the development of cloud computing has made increasingly diverse services with different quality levels available for both the data itself and services based on it.

3. Current businesses using cloud computing have begun to provide data management services, data transfer services, and data storage services. The delivery of Internet-scale data-intensive services has brought a number of challenges, such as, the maintenance of QoS attributes as well as excellent opportunities for businesses to gain market share, and for scientific programs to save on cost and energy as well as on space for data management.

This chapter addresses the challenges listed above and applies two algorithms to solve them.

4.2 Data-Intensive Service Provision

4.2.1 An Economic Model

In general, data-intensive service provision will be cooperatively supported by three stakeholders: the service composers, the service providers, and the data providers.

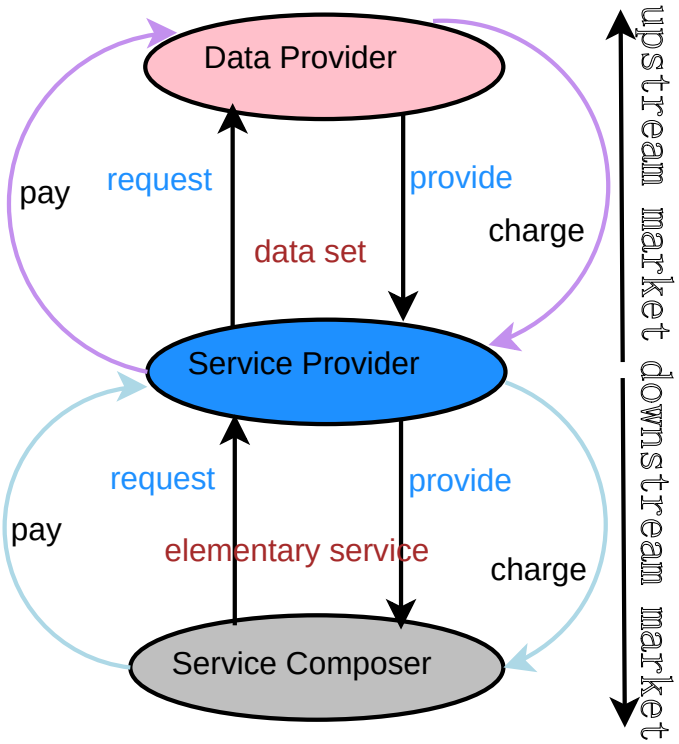


Figure 4.1: Service and data set usage and charging relationship

Providers need an approach to regulate and price their resources, either services or data or their combination. They all want to have a good market position maximizing their profits. It assumes that an economic model is an accurate representation of the reality, and it will offer a suitable way to regulate the interactions among the three stakeholders. As shown in Figure 4.1, in the downstream market, the service composers seek optimal strategies to select elementary services provided by multiple service providers who compete on the basis of price and QoS attributes. From the service composers' point of view, it is important to be able to assess the value of the needed services and how much they want to pay for them to satisfy their users' requirements as well as to maximize their profit. From the service providers' perspective, it is important to be able to analyze their competitors' position and improve their offers if they are to win contracts with the service composers. In the upstream market, the service providers request the data from the data providers. The price of the data may affect the total cost and the price of services. Therefore, the prices of both service and data have a crucial impact on the service composers' and the service providers' profits.

Here the author makes a distinction between cost and other QoS attributes because cost is usually related to other quality attributes and it becomes more impor-

tant in the data-intensive service provision. In the traditional service composition, executable services and their input or output data are usually on the same platform. Thus, the cost for data storage can be neglected or the cost is a constant determined before execution, and service selection algorithms need not consider it. However, in data-intensive service composition, service providers charge service requesters depending on the requesters' location and the amount of transferred data [111]. Each service requests data sets from the storage resources (or data servers). Each of these data sets may be replicated at several locations that are connected to each other and to the service platform through networks of varying capability [125, 158, 188]. When composing data-intensive services, optimizing the cost of data is a priority, as data plays the dominant role.

4.2.2 An Extensible QoS Model

Consider a data-intensive service $\tilde{s}$ on platform y has been chosen to implement abstract service AS_i, which is connected by links of different bandwidths with all the data servers. All data sets required by $\tilde{s}$ is denoted by DT^i. The price of data set dt is denoted by p_{dt}, which is the fee that a data user has to pay to the data provider for the data usage. The size of data set dt is denoted by $size(dt)$. $C_{ac}(\tilde{s})$ and $C_{tr}(\tilde{s})$ are used to denote the access cost and the transfer cost of all data sets required by $\tilde{s}$, respectively. $C_{sr}(\tilde{s})$ is used to denote the service related cost, which mainly includes the cost to provide the service and the cost to process the data sets. The data transfer time, $T_t(dt, d_{dt}, y)$, is the time to transfer the data set $dt \in DT^i$ from the remote platform d_{dt} that houses the data to the local platform y, which has the service requesting the data. The cost for service $\tilde{s}$, $Cost(\tilde{s})$, can be described by (4.1).

$$
\begin{aligned}
&Cost(\tilde{s}) = C_{ac}(\tilde{s}) + C_{tr}(\tilde{s}) + C_{sr}(\tilde{s}) \\
&C_{ac}(\tilde{s}) = \sum_{dt \in DT^i} p_{dt} \\
&C_{tr}(\tilde{s}) = \sum_{dt \in DT^i} T_t(dt, d_{dt}, y) * tcost \\
&T_t(dt, d_{dt}, y) = size(dt)/bw(d_{dt}, y)
\end{aligned}
\tag{4.1}
$$

The variable *tcost* is the cost of data transfer per time unit, $bw(d_{dt}, y)$ is the network bandwidth between data server d_{dt} and service platform y, and $size(dt)/bw(d_{dt}, y)$ denotes the practical transfer time.

The estimated execution time for service $\tilde{s}$, $T_{et}(\tilde{s})$, includes the time for processing data sets, which is denoted by $T_p(\tilde{s})$, and the time for accessing data sets, which is denoted by $T_{ad}(\tilde{s})$. $T_{ad}(\tilde{s})$ is the maximum value of response time for accessing all data sets required by service $\tilde{s}$. The access response time of data set dt, which is denoted by $T_{rt}(dt)$, includes the data transfer time $T_t(dt, d_{dt}, y)$, the storage access latency $T_{sal}(d_{dt})$, and the request waiting time $T_{wt}(d_{dt})$. Thus, $T_{et}(\tilde{s})$ can be described by (4.2).

$$
\begin{aligned}
&T_{et}(\tilde{s}) = T_p(\tilde{s}) + T_{ad}(\tilde{s}) \\
&T_{ad}(\tilde{s}) = \max_{dt \in DT^i} \Big(T_{rt}(dt) \Big) \\
&T_{rt}(dt) = T_t(dt, d_{dt}, y) + T_{sal}(d_{dt}) + T_{wt}(d_{dt}) \\
&T_{sal}(d_{dt}) = size(dt)/sp(d_{dt}) \\
&T_{wt}(d_{dt}) = \sum_{i=1}^{nr} \Big(size(dt^i)/sp(d_{dt}) \Big)
\end{aligned}
\tag{4.2}
$$

The variable $sp(d_{dt})$ is the storage media speed of d_{dt}, and nr is the number of data requests waiting in the queue prior to the underlying request for dt. The data transfer time depends on the network bandwidth and the size of the data. The storage access latency is the delayed time for the storage media to serve the requests, and it depends on the size of the data and storage type [4]. Each storage media has many requests at the same time and it serves only one request at a time. The current request needs to wait until all requests that are prior to it have finished.

4.2.3 Service Provision Model Using an AND/OR Graph

An AND/OR graph is used to form the service composition model. As shown in Figure 4.2, there is a logical AND relationship between abstract service AS_2 and AS_3, and there is a logical OR relationship between abstract service AS_4 and AS_5. The graph is denoted by $G = (V, E, D, start, end)$, where $V = \{V_{and} \cup V_{or} \cup V_{other}\}$ and E represent the vertices and edges of the graph, respectively. V_{and} is the set of

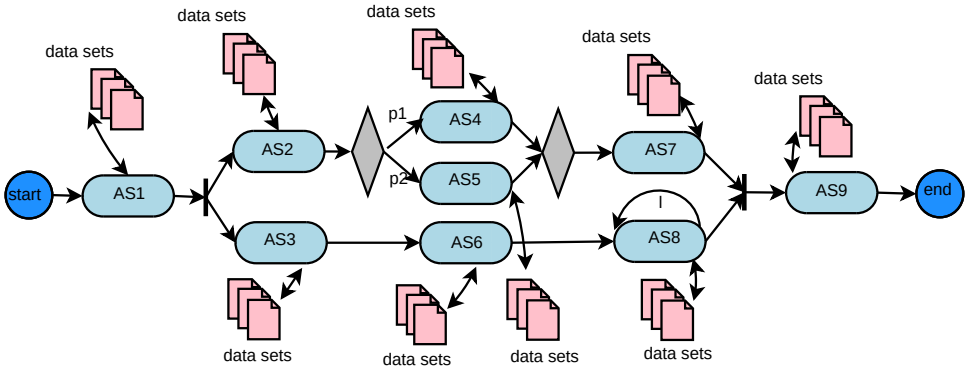


Figure 4.2: An AND/OR graph for data-intensive service provision

AND vertices, V_{or} is the set of OR vertices, and V_{other} is the set of remaining vertices. D represents a set of data servers that house data sets. Two virtual vertices *start* and *end* are used to indicate the beginning and ending of the composition process.

In graph G, a vertex is called an 'AND split' vertex if there is a logical AND relationship between its direct successors, such as the vertex AS_1. A vertex is called an 'OR split' vertex if there is a logical OR relationship between its direct successors, such as the vertex AS_2. Meanwhile, a vertex is called an 'AND join' vertex if there is a logical AND relationship between its direct predecessors, such as the vertex AS_9. A vertex is called an 'OR join' vertex if there is a logical OR relationship between its direct predecessors, such as the vertex AS_7. All 'AND split' and 'AND join' vertices are called AND vertices, and all 'OR split' and 'OR join' vertices are called OR vertices. Each 'AND split' vertex and its 'AND join' vertex are a pair, and each 'OR split' vertex and its 'OR join' vertex are a pair. For example, AS_1 and AS_9 are AND vertices, AS_2 and AS_7 are OR vertices.

4.2.4 Objective Function

As mentioned in Chapter 2, the goal of the service provision problem is to maximize the aggregated quality by considering all possible execution paths and their corresponding execution probability. For example, there are two execution paths in Figure 4.2: ep_1 includes all abstract services except AS_5, and ep_2 includes all abstract services except AS_4. The objective function can be stated by (4.3).

$$\max U_{total} = p_{ep_1} * U_{ep_1} + p_{ep_2} * U_{ep_2} \tag{4.3}$$

U_{total} is the total utility of all paths, U_{ep_1} and U_{ep_2} are the overall utility of execution path ep_1 and ep_2, respectively. p_{ep_1} and p_{ep_2} are the execution probability of ep_1 and ep_2.

The cost and execution time of data-intensive services is considered. Then the overall utility of the execution path, U_{ep_z} ($z \in \{1, 2\}$), is given by (4.4).

$$U_{ep_z} = \frac{Q_c^{MAX} - Cost(ep_z)}{Q_c^{MAX} - Q_c^{MIN}} * W_c + \frac{Q_t^{MAX} - Time(ep_z)}{Q_t^{MAX} - Q_t^{MIN}} * W_t \tag{4.4}$$

The variable W_c and W_t ($W_c + W_t = 1$) represent weights of cost and execution time. Q_c^{MAX} and Q_c^{MIN} represent the maximal value and the minimal value of cost of all the possible paths. Q_t^{MAX} and Q_t^{MIN} represent the maximal value and the minimal value of execution time of all the possible paths.

The cost and execution time of ep_1 and ep_2 are given by (4.5) and (4.6).

$$\begin{aligned} Cost(ep_1) &= c_1 + c_2 + c_3 + c_4 + c_6 + c_7 + l * c_8 + c_9 \\ Time(ep_1) &= t_1 + max\Big(t_2 + t_4 + t_7, t_3 + t_6 + l * t_8\Big) + t_9 \end{aligned} \tag{4.5}$$

$$\begin{aligned} Cost(ep_2) &= c_1 + c_2 + c_3 + c_5 + c_6 + c_7 + l * c_8 + c_9 \\ Time(ep_2) &= t_1 + max\Big(t_2 + t_5 + t_7, t_3 + t_6 + l * t_8\Big) + t_9 \end{aligned} \tag{4.6}$$

The variable l is the loop count in the loop structure. t_i and c_i ($i \in \{1, 2, \ldots, 9\}$) represent the execution time and cost of the concrete service that is selected to implement AS_i, and they are computed according to (4.1) and (4.2), respectively. The constraint is that only one concrete service is selected to implement each abstract service during the process of service composition.

4.3 Data-Intensive Service Provision Based on an Ant Colony System

The ant colony system (ACS) is an algorithm inspired by the ant system (AS), but it differs from AS in three main aspects [54]. First, the state transition rule provides a direct way to balance between exploration of new edges and exploitation of a priori and accumulated knowledge about the problem. Second, the global updating rule

is applied only to edges which belong to the best path. Third, a local pheromone updating rule is applied while ants construct a solution.

As discussed, considering different workflow structures and an optimized function with QoS attributes, the service composition problem is modeled as an AND/OR graph. In the graph, ants are initially positioned at the *start* vertex. The task of each ant is to find a path from the *start* vertex to the *end* vertex. If the ant arrives at an 'AND split' vertex, it will clone several new ants and each ant will choose one of the successors according to the state transition rule. If the ant arrives at an 'OR split' vertex, it only chooses one of the branches according to the execution probability, then it will apply the state transition rule to choose the successor. While finding the path, an ant uses a local updating rule to modify the pheromone on the edges it has passed. Once all ants arrive at the *end* vertex, the global updating rule will be applied to modify the pheromone on the edges of the best path. The key to ACS is how to determine the state transition rule, the local updating rule, the global updating rule, and the ant replication and death rule.

4.3.1 State Transition Rule

When ant k arrives at vertex i, it will choose successor j to move to by applying the rule given by (4.7).

$$j = \begin{cases} \arg\max_{j \in N_i^k} \{[\tau_{ij}][\eta_{ij}]^\beta\}, & \text{if } q \leq q_0; \\ \text{randomly selected from } N_i^k \text{ according to } p_{ij}^k, & \text{otherwise.} \end{cases} \tag{4.7}$$

The variable q is a random number uniformly distributed in $[0, 1]$, and q_0 $(0 \leq q_0 \leq 1)$ is a parameter. If $q > q_0$, j is randomly selected according to the probability distribution given by (4.8).

$$p_{ij}^k = \begin{cases} \frac{[\tau_{ij}][\eta_{ij}]^\beta}{\sum_{j \in N_i^k} [\tau_{ij}][\eta_{ij}]^\beta}, & \text{if } j \in N_i^k \ ; \\ 0, & \text{otherwise.} \end{cases} \tag{4.8}$$

The set of unvisited vertices, N_i^k, contains all the direct successors of vertex i when ant k is at it. The variable τ_{ij} is the pheromone density on edge (i, j). The variable

η_{ij} is the heuristic information and is set as the quality score of vertex j, which is computed according to the weighting phase described in Chapter 2. The parameter β controls the influence of η_{ij}.

4.3.2 Local Updating Rule

When building up a solution to the problem, i.e., a path through the graph, the ants use a local pheromone updating rule that they apply immediately after having crossed an edge (i, j), which is shown by (4.9).

$$\tau_{ij} = (1 - \xi)\tau_{ij} + \xi\tau_0, \forall(i, j) \in E. \tag{4.9}$$

The variable ξ $(0 < \xi < 1)$ is used to determine the local evaporation rate. At the beginning of the search process, a constant amount of pheromone is assigned to all edges, $\tau_{ij} = \tau_0$ ($\forall(i, j) \in E$, τ_0 is a constant). The local updating rule will reduce the pheromone trail on edge (i, j) after an ant has passed it. In other words, it allows an increase in the exploration of edges that have not been visited yet and, in practice, has the effect that the algorithm does not show stagnation behavior [54].

4.3.3 Global Updating Rule

In the iteration, after all ants arrive at the *end* vertex, a global pheromone updating rule is performed to the best path found so far, which is given by (4.10).

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \rho\Delta\tau_{ij} \tag{4.10}$$

The variable ρ $(0 < \rho < 1)$ denotes the global pheromone evaporation rate, and

$$\Delta\tau_{ij} = \begin{cases} U, & \text{if } \forall(i, j) \in P^{bs} \ ; \\ 0, & \text{otherwise .} \end{cases}$$

U is the overall utility of the best path P^{bs}, which is computed according to (4.4). This rule indicates that both the evaporation and the new pheromone deposit, only apply to every edge of the best path P^{bs}. There are two types of best paths: the best path in the current iteration of the trial, called iteration-best, and the best

path from the beginning of the trial, called global-best. Experiments have shown that the global-best is slightly better, which is therefore used in this research.

4.3.4 Ant Replication and Death Rule

When ant k is at an 'AND split' vertex i which has f direct successors, ant k will make $(f-1)$ copies of itself. Then ant k and the $(f-1)$ ants will choose one of the direct successors of vertex i to move to, according to the state transition rule that described by (4.7), and different ants choose different successors. During the path construction, each ant applies the local updating rule to the edge that it has passed. Ant k and all of its copies will keep the vertices they have visited in the list of ant k. When the $(f-1)$ ants arrive at the 'AND join' vertex of i, all of them will die. Then ant k will determine the current 'AND join' vertex. The whole process is recursive. All the copies of ant k construct paths using the same rule as ant k. That is to say, a copy ant of ant k will make copies of itself when it arrives at an 'AND split' vertex. All copy ants will die when they arrive at the 'AND join' vertex, which pairs with the 'AND split' vertex where they are replicated.

The ant colony system for the data-intensive service provision is given in Figure 4.3.

4.4 Data-Intensive Service Provision Based on a Genetic Algorithm

To use a genetic algorithm to search for a solution of a problem, the first step is to encode the problem with a suitable genome. The integer array coding scheme is chosen, i.e., every chromosome is represented by an integer array with a number of items. Using the integer array coding scheme, any change in the number of concrete services would not influence the length of the genome. Also, this type of encoding scheme is human-readable and straightforward to represent the service composition solution. Each gene in the chromosome represents an abstract service in the composite service, and the value of the gene represents an assignment of a concrete service for the abstract service. A genetic algorithm works with a set of chromosomes called a population. The whole population moves like one group towards an optimal area,

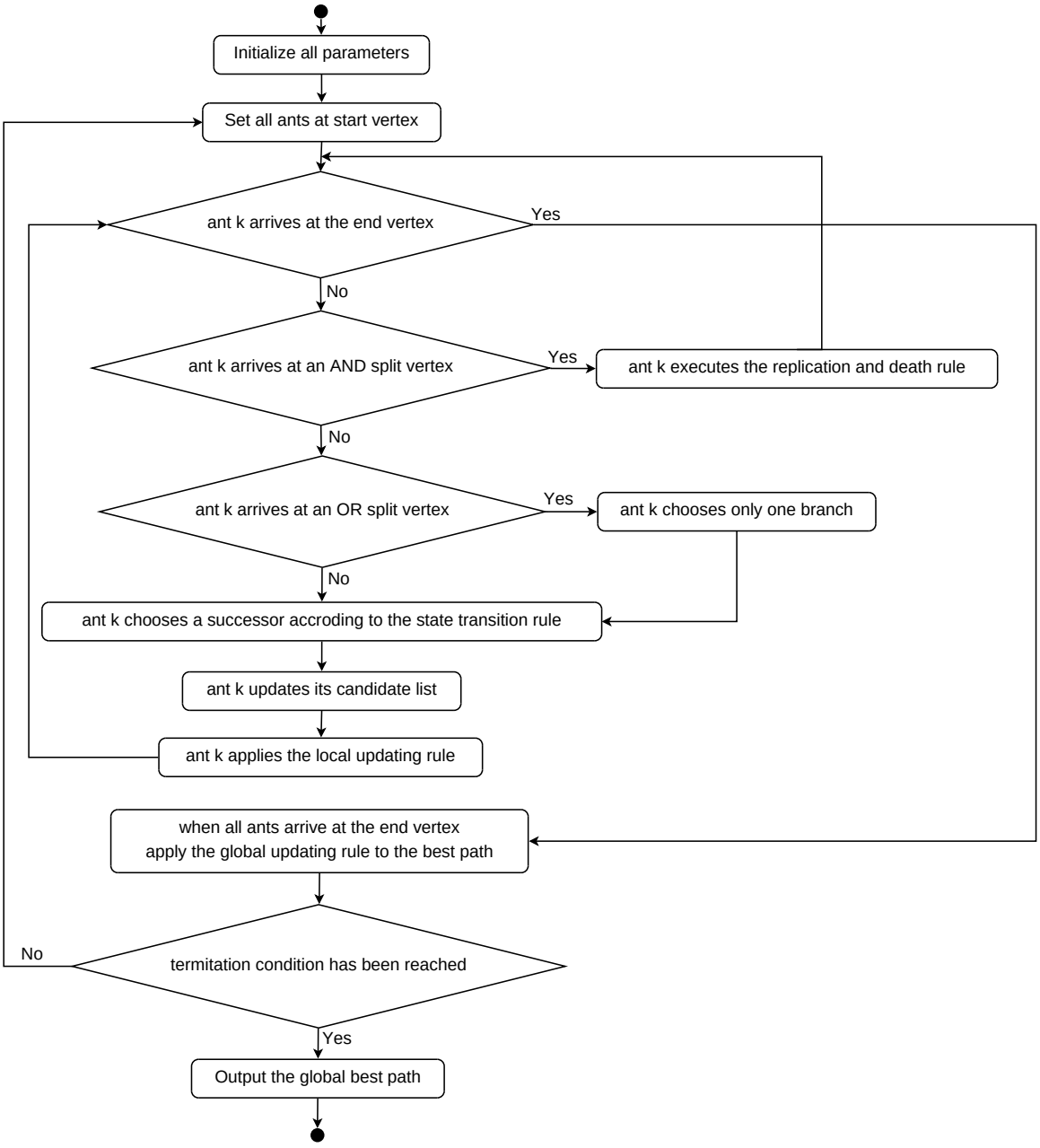


Figure 4.3: Data-intensive service provision based on an ant colony system

so a genetic algorithm searches from a population of solutions rather than a single solution.

4.4.1 The Implementation of a Genetic Algorithm

The implementation of the genetic algorithm applied to the data-intensive service provision is as follows.

Initialization. Once a suitable representation for the chromosomes has been decided, it is necessary to create an initial population to serve as the first generation.

Here, the initial population is randomly created according to the AND/OR graph. Only one branch of the conditional structure is selected according to the execution probability. When creating the initial population, if one of the branches of the conditional structure is not selected, the value of the gene is set to zero. That is to say, the related abstract service is not included in the composite service.

Evaluation. After the initial population is created, every individual in the population is evaluated by a fitness function. Fitness denotes a measure that is used to select individuals for further evolution. Applied to the service selection problem, fitness is a measure of the overall utility of the composition solution resulting from the selection of service candidates for each abstract service. Thus, the fitness function is the same as the utility function, which is given by (4.4).

Selection. After the fitness evaluation, a set of individuals are selected and they are then used by the crossover operator to produce offspring for the new generation. The selection operator is the combination of the elitism selection and the tournament selection. The elitism selection involves copying a few best individuals, unchanged, into the next generation. The tournament selection involves selecting a set of pairs of individuals as parents to breed the remainder of the next generation. Two separate tournaments were performed to choose father and mother, respectively. After a set of pairs of parents has been selected, a crossover operation will be executed to produce offspring.

Crossover operation. The crossover operator here is the single point crossover. Suppose there are $Npop$ individuals in the population and the crossover probability is PC, then there are $Npop * PC$ individuals which are replaced by the new offspring, and there are $Npop * (1 - PC)$ individuals which are able to survive to the next generation. The crossover point is created randomly, but must be checked as to whether it will create infeasible solutions. For example, suppose there are twenty service candidates for each abstract service in Figure 4.2 and the crossover point is four. The parents and the offspring are given in Figure 4.4. The two offspring are infeasible because all conditional branches are selected in the first offspring, and none of the conditional branches is selected in the second offspring. The checking rule is that there is one and only one branch of each conditional structure to be selected.

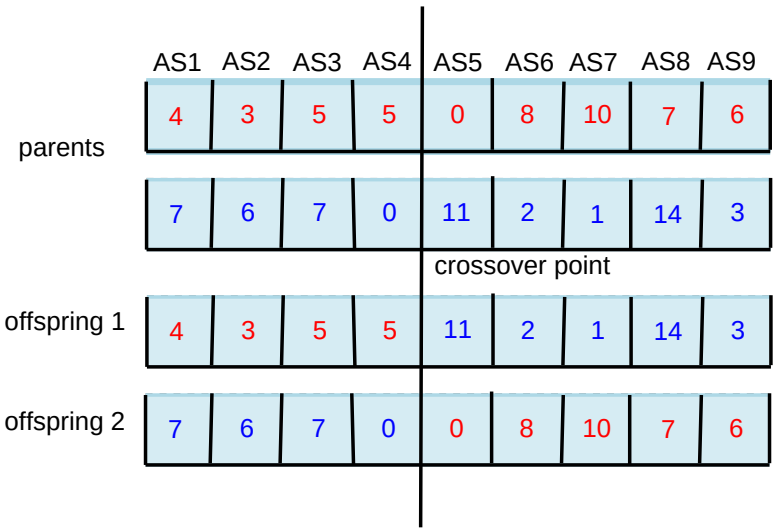


Figure 4.4: An example of infeasible individuals generated by the crossover operator

Mutation operation. If only the crossover operator was used to produce offspring, unexplored service composition solutions cannot be added into the population. Also, the same value of the gene at the same position of all individuals in the initial population will be kept in future offspring. Thus, it is necessary to adopt a mutation operator. The mutation policy is proposed as follows. The probability of mutation is PM, which is for the locus. The locus for each gene represents its own position in the chromosome. Every locus in each chromosome that was created by the crossover operation is checked for possible mutation by generating a random number between zero and one. If this random number is less than or equal to the given mutation probability, then the value of the gene will be replaced by the assignment of another concrete service in the service candidate set, according to the local selection approach. The local selection approach is based on the utilities of the concrete services. Prior to the mutation operation, the utility of each concrete service in each service candidate set is computed. Then all the concrete services in each service candidate set are sorted in descending order according to their utilities. When the mutation operation is applied, the replacement process will search another service candidate from the beginning of the service candidate set until the assignment is different from the old assignment, and will then replace it.

Before the mutation operation, it is necessary to check whether the value of the gene equals to zero. If the value of the gene equals to zero, it means the related abstract service is in a conditional branch and it is not selected, so the mutation

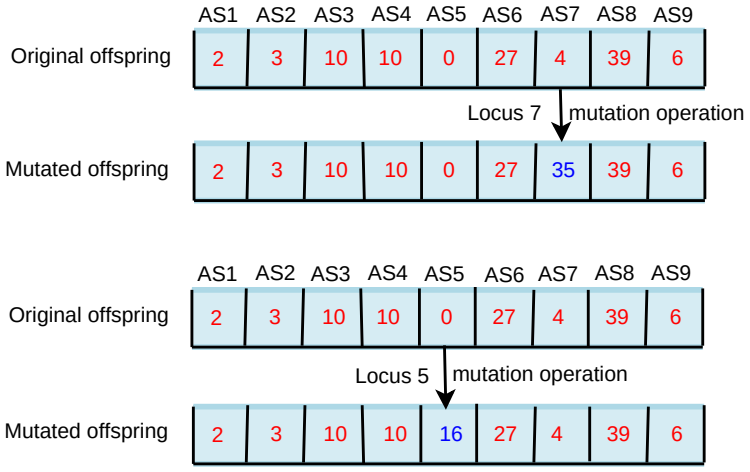


Figure 4.5: An example of feasible and infeasible individuals generated by the mutation operator

operator will not be applied to this locus. For example, suppose there are forty service candidates for each abstract service in Figure 4.2. The original offspring and mutated offspring are given in Figure 4.5. The mutated offspring is feasible when the locus is seven, but it is infeasible when the locus is five because the value of the fifth gene equals to zero. After the mutation operation, only the chromosome with the greater fitness value will survive between the mother and child chromosome.

Iteration and termination. After the crossover and the mutation operations, the new population is composed of the individuals in the new generation and a few individuals from the previous generation. The fitness of each individual in the new population is evaluated and the whole procedure is repeated until a termination condition has been reached.

The genetic algorithm for the data-intensive service provision is given in Figure 4.6.

4.5 Experiments and Analysis

The aim of this evaluation is to analyze the performance of the proposed algorithms: 1) observing the evolution of the utility over the ACS iteration numbers; 2) observing the evolution of the fitness value over the GA generations; 3) comparing the proposed GA with the proposed ACS; 4) comparing the proposed GA and ACS with the MIP approach [8, 142]; and 5) comparing the proposed GA with the GA-based random selection approach [33, 75, 93, 167]. All the experiments are conducted on

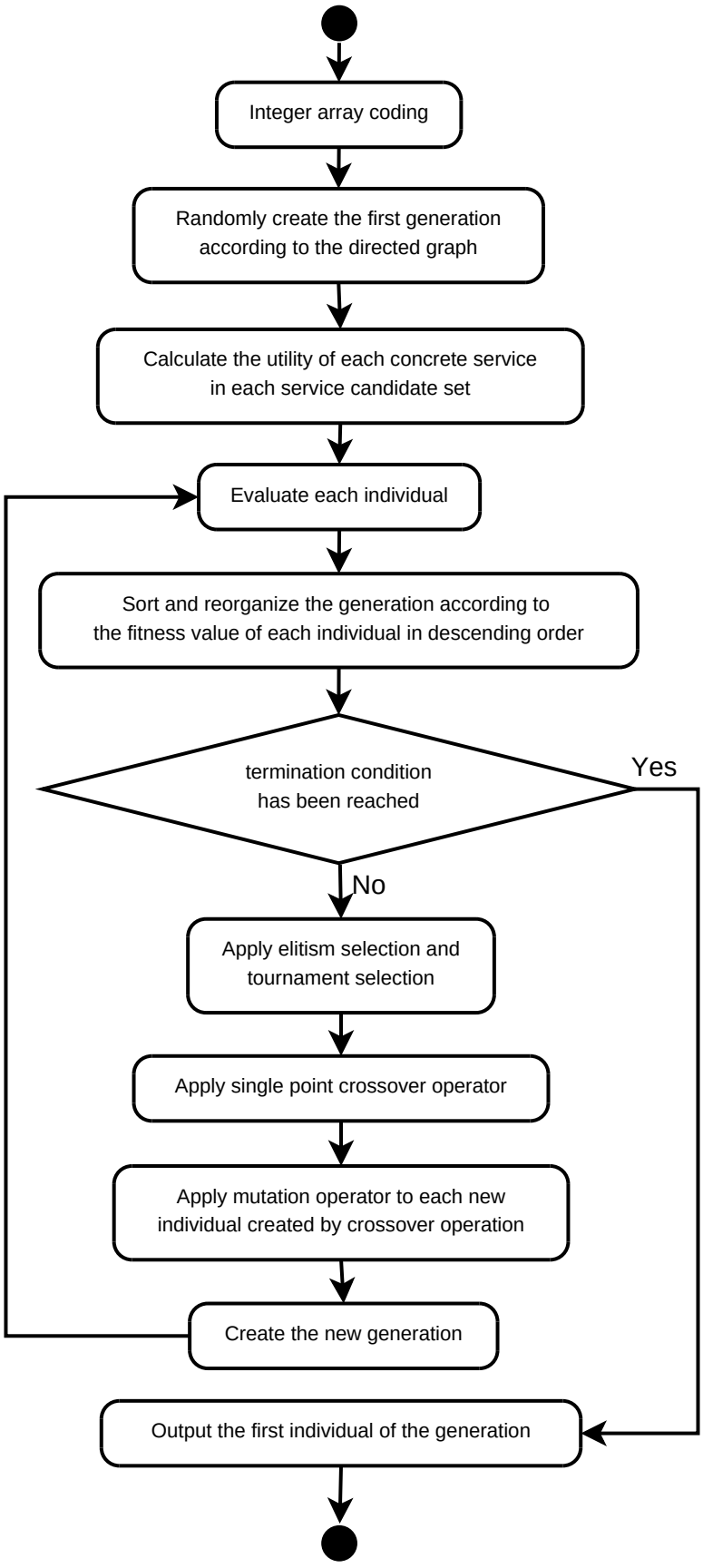


Figure 4.6: Data-intensive service provision based on a genetic algorithm

a computer with Inter Core i5 2500 CPU (3.3GHz and 8 GB RAM).

4.5.1 The Parameters

In the experiments, a trial testing method is adopted to determine the most suitable values for all parameters of ACS and GA, considering other researchers' earlier experiments. Finally, the parameters of ACS used in the experiment are: $\beta = 2$, $q_0 = 0.9$, $\tau_0 = 0.1$, $\rho = 0.1$, $\xi = 0.1$, and the number of ants is 20. The parameters of GA in the experiment are: $PC = 0.7$, $PM = 0.1$, and the number of individuals is 20. The weights for cost and response time are 0.8 and 0.2, respectively. The loop structure can be unfolded by cloning the vertices involved in the structure as many times as the maximal loop count [203]. In addition, two termination conditions were defined for both algorithms: iterate until a maximum iteration numbers ($MaxIt = 1000$) is reached, alternatively, iterate until the best utility remains unchanged for a given iteration numbers ($EIT = 50$).

4.5.2 Evaluation Methodology

The performance of the GA and the ACS is affiliated to the size of the problem. The size of the problem depends on the number of abstract services in the composite service, and the number of concrete services for each abstract service. Here, the influence of the number of data sets is not considered.

For the purpose of the evaluation, different scenarios are considered where a composite service consists of n abstract services, and n varies in the experiments between 10 and 50, in increments of 10. There are m concrete services in each service candidate set, and m varies in the experiments between 100 and 1000, in increments of 100. Each abstract service requires a set of k data sets, and k is fixed at 10 in the experiments. A scenario generation system is designed to generate all scenarios for the experiments. The system first determines a basic scenario, which includes sequence, conditional and parallel structures. With this basic scenario, other scenarios are generated by either placing an abstract service into it or adding another composition structure as substructure. This procedure continues until the scenario has the predefined number of abstract services.

Three performance factors were evaluated: 1) the required computation time;

2) the quality of the solution; and 3) the value of FRIT, which is the number of the iterations when the best utility appeared, and from this iteration the best utility will not change until the termination condition has been reached. As ACS and GA are sub-optimal, the solutions obtained by these two bio-inspired algorithms have been evaluated through comparing them with the optimal solutions obtained by the MIP approach. The overall utility of the solution obtained by ACS or GA is denoted by U_{bio}, and the overall utility of the solution obtained by the MIP approach is denoted by U_{global}. Then the optimality of the solution created by ACS or GA is computed as $optimality = U_{bio}/U_{global}$. The open source integer programming system *lpsolve* version 5.5 [22] was used for the MIP approach.

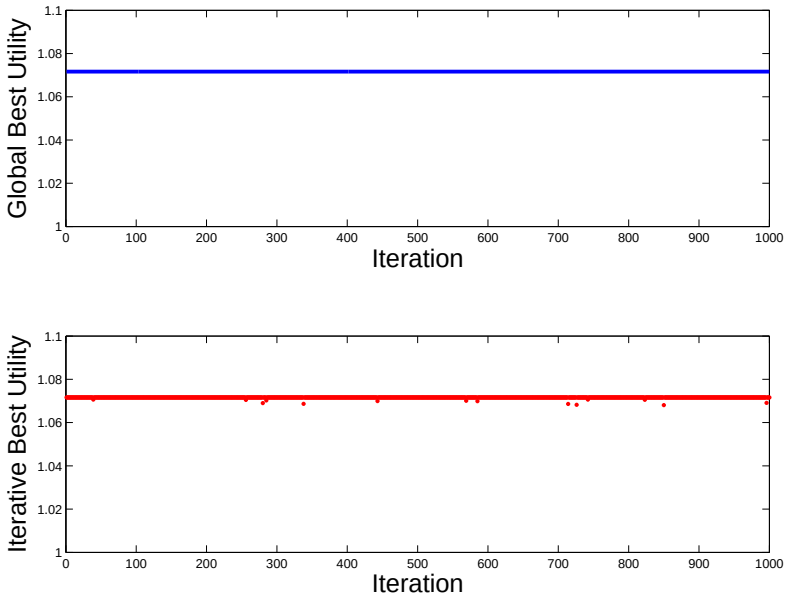
4.5.3 The Dataset

The synthetic datasets were experimented. For each scenario, the price of a data set, the network bandwidth (Mbps) between each data server and the service platform, the storage media speed (Mbps), the size (MB) of a data set, and the number of data requests in the waiting queue were randomly generated from the following intervals: [1,100], [1,100], [1,100], [1000,10000] and [1,10]. Then every scenario was performed with 50 runs. All runs of the same scenario use the same data, and the average results over 50 independent runs are reported.

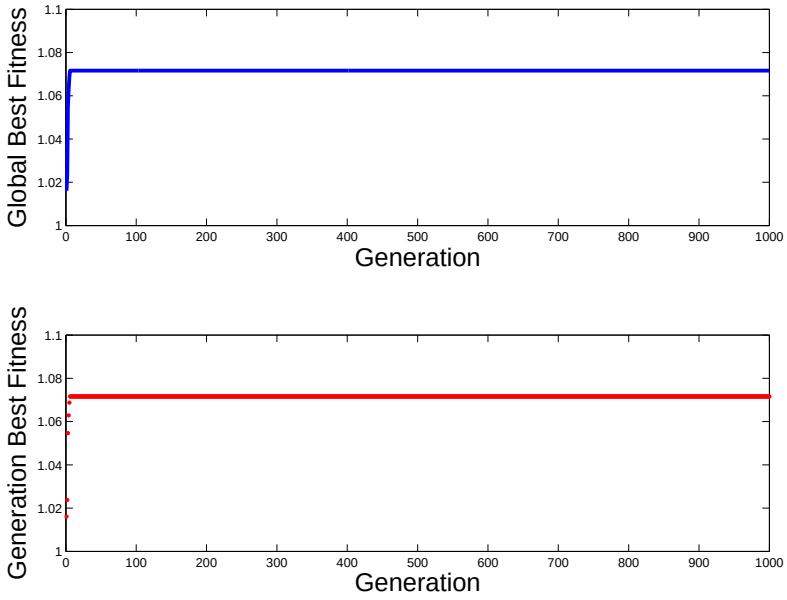
4.5.4 Results Analysis

Two examples of all experiments are given to show the evolution of utility value over the ACS iteration and the evolution of fitness value over the GA generation, which are shown in Figure 4.7. The figure shows a scenario where a composite service consists of 10 different abstract services, and each abstract service has 200 service candidates. In the figure, the short blue line represents the global best utility (fitness) value, and the red point represents the best utility (fitness) value in the current iteration (generation). The results indicate that the ants find the best utility in the first iteration, and the genetic algorithm finds the best fitness value in the seventh generation.

In Figure 4.8, the performance of ACS, GA, and MIP are compared with respect to the number of candidate services and the number of abstract services. In Fig-



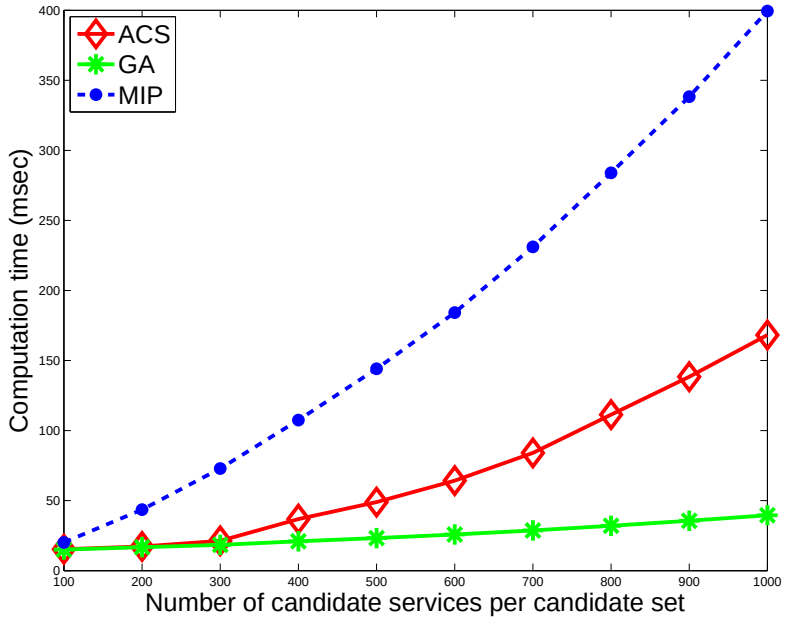
(a) The evolution of utility value over the ant colony system iteration



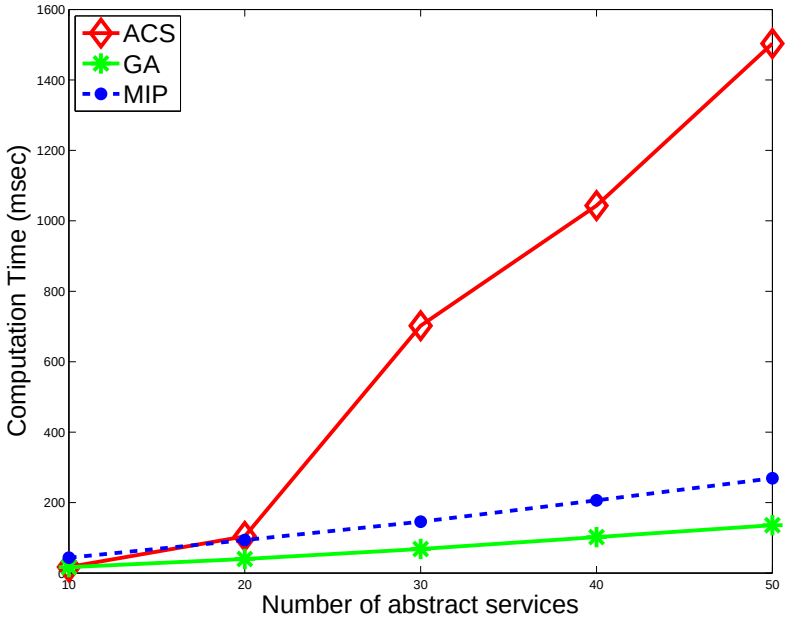
(b) The evolution of fitness value over the genetic algorithm generation

Figure 4.7: The evolution of utility/fitness value in ACS and GA

ure 4.8(a), the number of candidate services for abstract service varies from 100 to 1000, while the number of abstract service is set to 10. The results indicate that the proposed ACS and GA are faster than the MIP method. By increasing the number of candidate services, the required computation time of GA increases very slowly, this makes GA be more scalable. In Figure 4.8(b), the number of abstract services



(a) Performance vs. number of candidate services



(b) Performance vs. number of abstract services

Figure 4.8: The performance of ACS, GA, and MIP

varies from 10 to 50, while the number of candidate services for each abstract service is fixed at 200. The results of this experiment indicate that the performance of all three methods degrades as the number of abstract services increases. Again, we observe that GA outperforms ACS and MIP. The reason is that each ant needs to compute the probability of next vertex while finding the path, and such computa-

Table 4.1: Means of optimality (%)

Scenarios	***ACS***	***GA***
n is fixed at 10, m varies between 100 and 1000, in increments of 100	100	100
	100	100
	100	100
	99.9908	100
	100	100
	100	100
	100	100
	99.9908	100
	99.9907	100
	100	100
m is fixed at 100, n varies between 10 and 50, in increments of 10	100	100
	99.9804	100
	99.9111	100
	99.8405	100
	99.7197	100

tion costs much time. Meanwhile, we observe that ACS outperforms MIP when the number of abstract services is small, in this case, less than 20.

Next, the quality of the solutions obtained by ACS and GA is presented. Table 4.1 shows that the optimality achieved by ACS and GA with respect to a varying number of candidate services and a varying number of abstract services. The results indicate that GA and ACS are able to achieve the optimal solutions, and GA slightly outperforms ACS.

Table 4.2 gives the means of FRIT obtained by ACS and GA with respect to a varying number of candidate services and a varying number of abstract services. The results show that both algorithms need more iterations to get the best utility value when the number of abstract services increases. When the number of concrete services increases, the values of FRIT of both algorithms does not change and ACS can get the best utility value almost in the first iteration.

The following presents the quality of the solutions obtained by the GA with local selection approach and the GA with random selection approach. Figure 4.9 shows the optimality achieved by the two types of GAs in terms of a varying number of candidate services and a varying number of abstract services. The results indicate that GA with local selection approach outperforms GA with random selection

Table 4.2: Means of FRIT

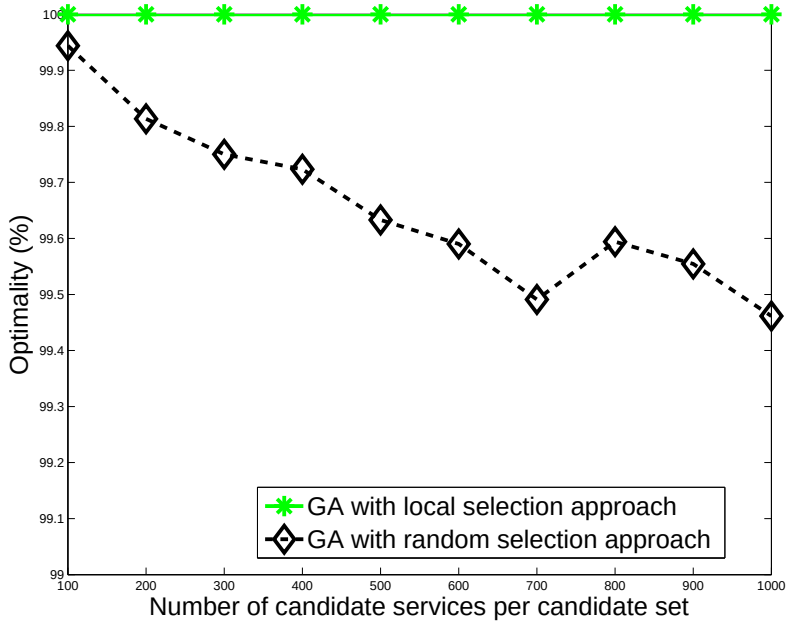
Scenarios	***ACS***	***GA***
n is fixed at 10, m varies between 100 and 1000, in increments of 100	1	7
	1	7
	1	7
	1	7
	1	7
	1	7
	1	7
	1	7
	1	7
	1	7
m is fixed at 100, n varies between 10 and 50, in increments of 10	1	7
	7	10
	11	12
	18	13
	26	15

approach.

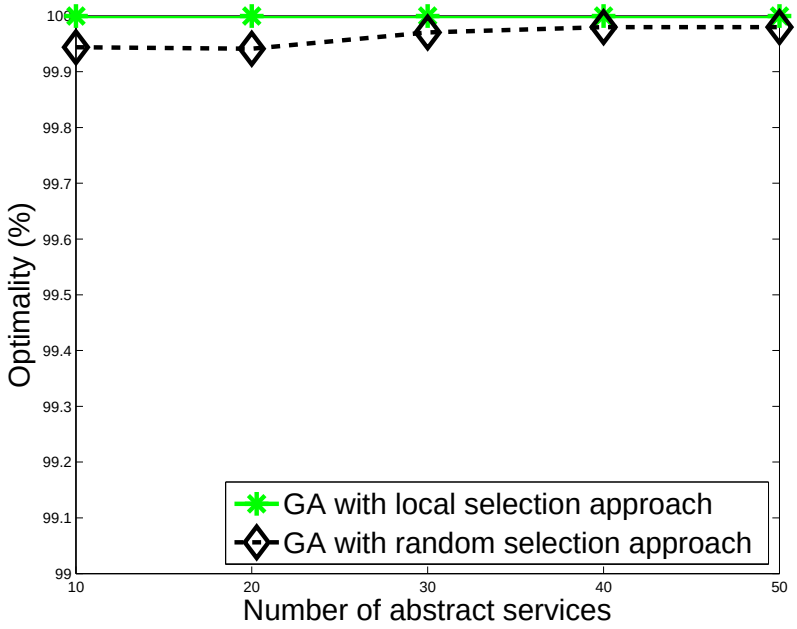
Figure 4.10 gives the mean values of FRIT of the two types of GAs in terms of a varying number of candidate services and a varying number of abstract services. The results show that when the number of concrete services and the number of abstract service increase, the values of FRIT of the GA with random selection approach increases more quickly than the GA with local selection approach. Again, we observe that the GA with local selection approach outperforms the GA with random selection approach.

4.6 Related Work

A variety of studies adopted ant colony optimization algorithms to solve Web service composition problems. The studies [182, 184, 192, 196] only considered sequential structure, and did not address the parallel, conditional, or loop structure, which is very common in business processes or workflows. Yu and Liu [199] modeled the QoS-aware Web service selection problem as an AND/OR graph, but the authors did not design the ant replication and death rule to deal with the AND vertices. Liang and Su [113] used an AND/OR graph to represent the dependencies between services



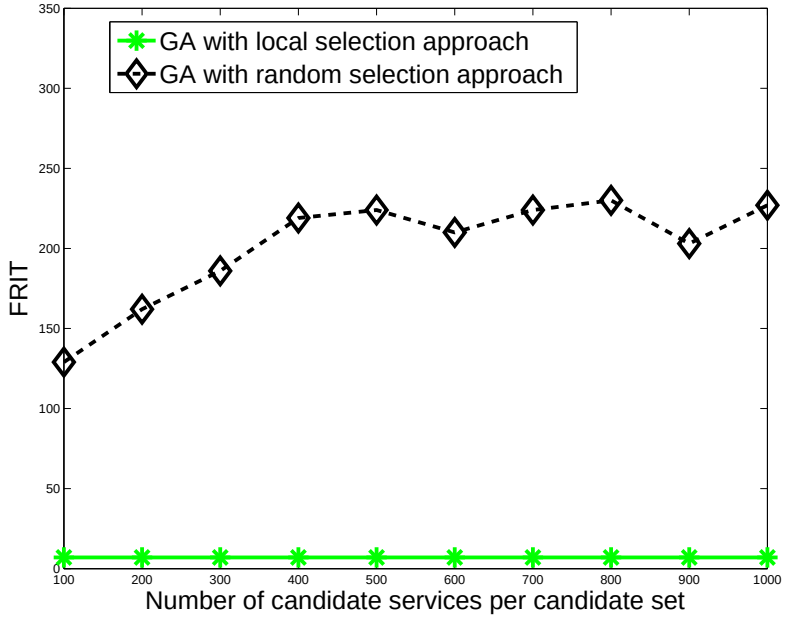
(a) Optimality vs. number of concrete services



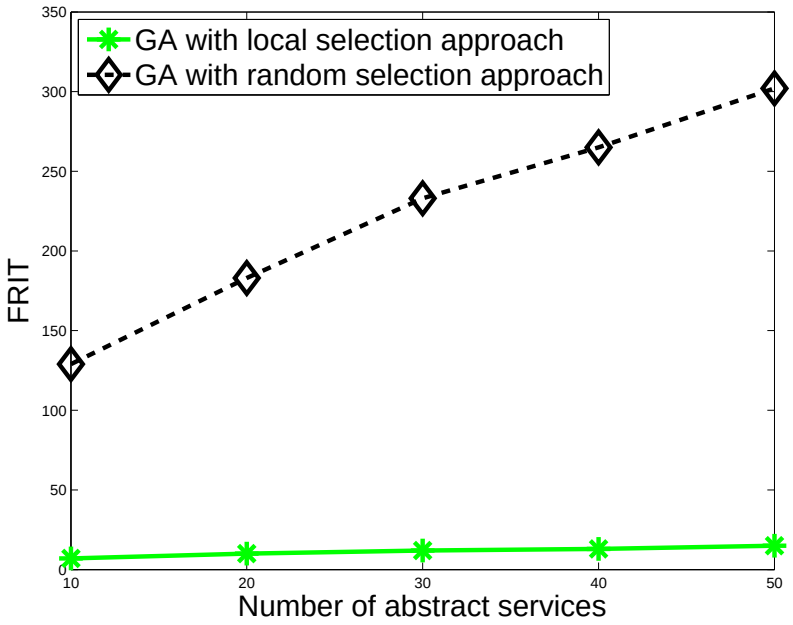
(b) Optimality vs. number of abstract services

Figure 4.9: The optimality of the two types of genetic algorithms

and presented a search algorithm for Web service composition. However, the authors focused on how to construct the graph to capture the input or output dependencies between the Web services, and they neglected the nonfunctional characteristics of services. Zheng et al. [208] modeled the service selection problem as an AND/OR graph and designed the ant clone rule. However, there were some issues with the



(a) FRIT vs. number of concrete services



(b) FRIT vs. number of abstract services

Figure 4.10: The values of FRIT of the two types of genetic algorithms

graph and the ant clone rule. First, the authors did not consider that the successors of an AND/OR vertex might also contain an AND/OR vertex. Second, the authors did not consider how to deal with all the situations of clone ants, such as when the clone ants finished their paths, and how to select candidate services when all clone ants arrived at the same vertex. These issues were solved in this chapter.

Many other studies only considered local QoS constraints [137, 206, 208], and they would not be suitable to resolve the Web service selection problem with global QoS constraints. Global QoS constraints were considered and used to decide whether the solution found by an ant was feasible [66, 174]. If the aggregated QoS attributes of the solution satisfies all the global constraints, then it is accepted, otherwise it will not be considered. This type of constraints-handling is simple and cannot provide useful information for ants to find paths under satisfying constraints. Particularly, the method presented by Xia et al. [192], every time an ant added a new service to its path, the aggregated QoS of the current path would be checked. If the aggregated QoS attributes did not satisfy the global constraints, then the ant would choose another service. This type of constraints-handling is time-consuming when the number of abstract services and concrete services are very large. So constraints-handling in ant colony optimization algorithms is also worthy of exploration.

Meanwhile, most of the existing studies have adopted the roulette wheel selection when they used genetic algorithms to solve the Web service composition problem. However, the selection has problems when the fitness values vary widely [33, 75, 93, 162, 167]. The selection operator in the proposed genetic algorithm is the combination of the elitism selection and the tournament selection, by first copying a few fittest individuals into the next generation, and then selecting a set of pairs of individuals as parents to produce the remainder of the next generation. The elitism selection can increase the performance of the GA very rapidly. The tournament selection is among the most widely used selection strategies in evolutionary algorithms. The combination of the two selection methods can guarantee the algorithm is not being stuck in local optima.

In order to escape from local optima, a local selection method and modifications of the crossover and the mutation operations are adopted. The local selection method can improve the convergence speed of GA. The modifications of the crossover and the mutation operators can prevent the creation of infeasible solutions.

Very few studies have considered the effect of data intensity and the communication cost of mass data transfer on service composition. This work is the first effort to address data-intensive service composition using an ant colony optimization algorithm and a genetic algorithm.

4.7 Summary

In this chapter, an economic model was presented for the data-intensive service provision. An extensible QoS model was also proposed to calculate the QoS attributes of the data-intensive services. The problem was modeled as an AND/OR graph, which was not only able to deal with sequential and parallel structures, but was also able to deal with conditional structures. An ant colony system and a genetic algorithm were conducted, and the performance of the two algorithms had been evaluated by simulations.

Both algorithms are proposed to solve the single-objective problem. The multi-objective ant colony system and the multi-objective genetic algorithm for data-intensive service provision are investigated in the next chapter.

Chapter 5

Multi-Objective Data-Intensive Service Provision

This chapter will evaluate two bio-inspired algorithms for the data-intensive service provision, which is modeled as a multi-objective optimization problem. Both the algorithms for a multi-objective context will get a set of Pareto-optimal solutions by considering two objectives at the same time, the total cost and the total execution time of a composite service. Chapter 4 describes two algorithms for the single-objective optimization problem. In the multi-objective algorithms there are two goals, one is to converge to the Pareto-optimal solutions and second is to maintain the diversity of the Pareto-optimal solutions. These two goals cannot be measured by one single performance metric. Also, it needs to define all the algorithmic components in the multi-objective ant colony system, and to explore the constraints-handling approach in the multi-objective genetic algorithm.

To evaluate the proposed algorithms, experiments on many different scenarios were conducted with respect to five performance metrics. Then a comprehensive comparison of the two algorithms are provided. The lessons learned from the experimental results are that, when we have a large number of concrete services available for each abstract service, a multi-objective genetic algorithm can achieve better solutions. On the other hand, whenever the number of concrete services available is small, such as in some simple and repetitive scientific computation, a multi-objective ant colony system is to be preferred to a multi-objective genetic algorithm.

5.1 Background

The goal of the majority of existing multi-objective optimization algorithms is to find Pareto-optimal solutions. The concept of dominance is used to relate the solutions found in these algorithms.

Definition 5.1 (Dominance) *In a minimization problem for all objectives, a solution x_1 dominates another solution x_2 (denotes as $x_1 \prec x_2$) if and only if the two following conditions are true: 1) x_1 is no worse than x_2 in all objectives, namely $F_i(x_1) \leq F_i(x_2)$ ($\forall i \in \{1, 2, \ldots, N\}$, N is the number of objective functions), and 2) x_1 is strictly better than x_2 in at least one objective, namely $F_j(x_1) < F_j(x_2)$ ($\exists j \in \{1, 2, \ldots, N\}$).*

Definition 5.2 (Cover) *In a minimization problem for all objectives, a solution x_1 is said to cover another solution x_2 (denotes as $x_1 \preceq x_2$) if one of the two following conditions is true: 1) x_1 dominates x_2, namely $x_1 \prec x_2$, or 2) x_1 is equal to x_2 in all objectives, namely $F_i(x_1) = F_i(x_2)$ ($\forall i \in \{1, 2, \ldots, N\}$, N is the number of objective functions).*

Definition 5.3 (Non-dominated set) *Among a set of solutions, the non-dominated set of solutions are those that are not dominated by any member of the set.*

A solution is said to be Pareto-optimal if it is not dominated by any other possible solutions. Thus, the Pareto-optimal solutions to a multi-objective optimization problem form the Pareto front or Pareto-optimal set [166]. Pareto-optimal sets are the solutions that cannot be improved in one objective function without deteriorating their performance in at least one of the remaining objective functions.

5.1.1 Multi-Objective Ant Colony Optimization Algorithms

Various alternatives to the implementation of multi-objective ant colony optimization (MOACO) algorithms have been proposed in the literature. They usually differ from each other with respect to the single-objective ACO algorithms on which they are based, such as the ant system, the ant colony system, and the max-min ant system, as well as the variations in their algorithmic components. Angelo and Barbosa [10] provided a comprehensive review of the use of ACO algorithms in the realm of multiple-objective problems, which is illustrated in Table 5.1.

Table 5.1: MOACO algorithmic components and values (Adapted from [10])

Components	***Values***	***Illustration***
Ant colony	Single	Multiple colonies can cooperate with each other by exchanging solutions or sharing solutions.
	Multiple	
Pheromone information	Single matrix	The pheromone information associated with each objective are combined.
	Multiple matrices	Each matrix corresponds to one objective.
Heuristic information	Single matrix	The heuristic information associated with each objective is combined.
	Multiple matrices	Each matrix corresponds to one objective.
Pheromone and heuristic aggregation	Weighted sum	$\sum_{f=1}^{N} \lambda_f \eta_{ij}^f$, $\sum_{f=1}^{N} \lambda_f = 1$, where N is the number of objectives. $\big($details can be found from (5.7)$\big)$
	Weighted product	$\prod_{f=1}^{N} (\eta_{ij}^f)^{\lambda_f}$, $\sum_{f=1}^{N} \lambda_f = 1$, where N is the number of objectives. $\big($details can be found from (5.7)$\big)$
	Random	a random objective is selected to be optimized.
Weight setting	Dynamic	Each ant may be assigned a different weight from the other ants in the iteration.
	Fixed	The weights can be set a priori and each objective has the same importance during the entire algorithm run.
Pheromone update	Elite	The iteration-best or best-so-far solution is used to update the pheromone.
	Best-of-objective	The iteration-best or best-so-far solutions with respect to each objective is used to update the pheromone.
	Non-dominated set	The solutions in the non-dominated set are allowed to update the pheromone.
	All	All ants are allowed to update the pheromone.
Pareto archive	Off-line	The non-dominated solutions can be stored in an external set used to update the pheromone.
	On-line	The Pareto set is connected to the pheromone update procedure at any time.
	No-archive	The Pareto set is not used to update pheromone but it is used as a final solution.

5.1.2 Multi-Objective Genetic Algorithms

Multi-objective genetic algorithms (MOGAs) belong to the class of multi-objective evolutionary algorithms, which are stochastic optimization methods and usually use a population-based approach to find Pareto-optimal solutions [209]. The non-dominated sorting genetic algorithm (NSGA) proposed by Srinivas and Deb [160] was one of the first multi-objective evolutionary algorithms. The NSGA was not an efficient algorithm because of 1) the high computational complexity of non-dominated sorting, 2) its lack of elitism, and 3) the need for specifying the sharing parameter. Several years later, Deb et al. [52] proposed an improved version of

NSGA, called NSGA-II, to solve the problems of NSGA. The three new mechanisms of NSGA-II are the fast non-dominated sorting procedure, the crowded-comparison procedure, and the elitism mechanism.

5.1.3 Performance Metrics

There are two goals in a multi-objective optimization: the convergence to an approximation set (the Pareto-optimal set found in a single run) and the maintenance of diversity in solutions of the Pareto-optimal set [52]. These two goals cannot be measured adequately with one single performance metric. Meanwhile, the outcome of a multi-objective optimization run will generally consist of a varying number of non-dominated solutions. Various performance metrics to measure these two goals have been suggested [16, 50, 101, 209]. Here, the following five performance metrics were chosen: 1) the computation time, 2) the overall non-dominated vector generation (ONVG), 3) the comparison metric (C metric), 4) the size of the dominated space, and 5) the summary attainment surface. The first four metrics measure the convergence of the Pareto-optimal solutions, while the fifth metric measures the distribution of the Pareto-optimal set obtained by a multi-objective optimization algorithm.

The Computation Time

The computation time, also called running time, is the length of time required to perform the algorithm.

The ONVG Metric

The ONVG metric measures the number of distinct non-dominated solutions in the calculated Pareto-optimal set GP. It is defined as $ONVG = |GP|_c$, where $||_c$ denotes cardinality. The larger the value of the ONVG, the more we know about the details of the Pareto-optimal set.

The C Metric

The C metric is based on comparing a pair of non-dominated sets by computing the fraction of each set that is covered by the other. C maps the ordered pair (A, B)

into the interval $[0, 1]$:

$$C(A, B) = \frac{|\{b \in B, \exists a \in A : a \preceq b\}|}{|B|} \tag{5.1}$$

where $|B|$ means the number of solutions in the non-dominated set B, and $a \preceq b$ means a covers b. $C(A, B) = 1$ means that all solutions in B are dominated by A. The opposite, $C(A, B) = 0$ means that none of the solutions in B is dominated by A. It is important to note that both $C(A, B)$ and $C(B, A)$ have to be considered, since $C(A, B)$ is not necessarily equal to $1 - C(B, A)$. $C(A, B) > C(B, A)$ means that the non-dominated set A has better solutions than B.

The Size of the Dominated Space

The size of the dominated space $S(A)$ indicates how well the Pareto-optimal set is approximated by the non-dominated set A of the algorithm [209]. For each non-dominated solution in A, we can compute the values of all objective functions. These values comprise a point in the solution space. Figure 5.1 illustrates an example of a dominated space, which is separated by four points (x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4). Since the optimization involves the minimization of two objectives, a reasonable maximum value for each objective (maxI and maxII) has to be chosen to determine the size of the dominated space. The greater the size of the space dominated by

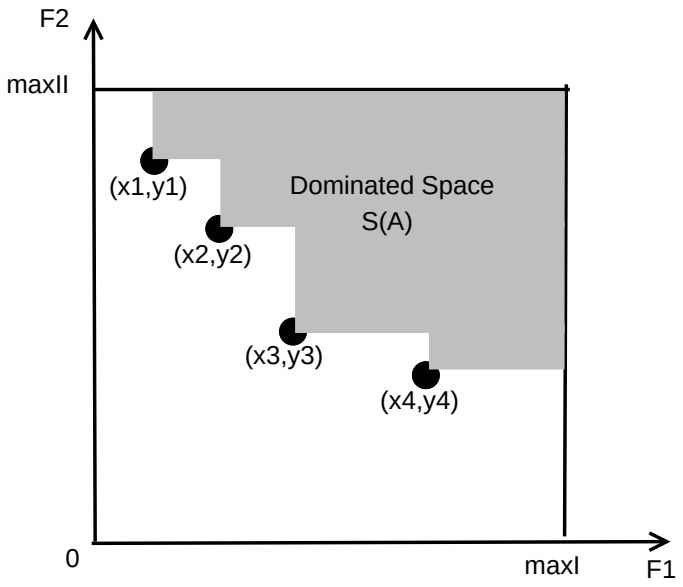


Figure 5.1: A space dominated by a non-dominated set for a minimization of two objectives

the non-dominated set, the closer the solutions are to the Pareto-optimal set. Here, $PS(A)$ is used to indicate this metric.

$$PS(A) = \frac{S(A)}{maxI * maxII} * 100\% \tag{5.2}$$

where

$$S(A) = (maxI - x_1) * (maxII - y_1) + (maxI - x_2) * (y_1 - y_2) + (maxI - x_3) * (y_2 - y_3) + (maxI - x_4) * (y_3 - y_4)$$

The Summary Attainment Surface

A summary attainment surface is a visual approach to summarizing a number of runs of a multi-objective optimizer. An attainment surface is a boundary in the objective space that separates those points that are dominated by or equal to at least one of the points, from those that no point dominates or equals [72]. The attainment surface emphasizes the achieved distribution and indicates the quality of the individual solutions. Knowles [101] proposed an algorithm to plot approximate summary attainment surfaces with any number of objectives, and he suggested that it was more useful to plot the median summary attainment surface to compare the performance of the optimizers. The median summary attainment surface, also called 50%-attainment surface of the optimizer, is the attainment surface on which all points are attained in exactly 50% of the runs [72, 101]. For a two-objective problem, the more the points of the median attainment surface of an algorithm close to the origin of the rectangular coordinate system, the better the solutions of the algorithm are.

5.2 Multi-Objective Data-Intensive Service Provision with Global QoS Constraints

The data-intensive service provision with global QoS constraints (DISP_GQoSC) is an extension of the traditional service provision problem, in which data sets, as the inputs and outputs of services, are incorporated. As mentioned in Chapter 2, a directed graph is used to represent the dependencies between services. The directed

graph of DISP_GQoSC is denoted as $G = \{V, E, D, start, end\}$, and is mathematically stated as:

Minimize an objective function F, given:

1. $V = \{AS_1, AS_2, \ldots, AS_n\}$ represents a set of n abstract services, and AS_i ($i \in \{1, 2, \ldots, n\}$) represents abstract service i. The *start* vertex and *end* vertex represent two virtual vertices, which are used to indicate the beginning and ending of the composition process;

2. $cs_i = \{cs_{i,1}, cs_{i,2}, \ldots, cs_{i,m}\}$ is the service candidate set of AS_i, which includes all concrete services that can be used to implement AS_i;

3. $q_{cs_{i,j}} = [q^1_{cs_{i,j}}, q^2_{cs_{i,j}}, \ldots, q^r_{cs_{i,j}}]$ with r QoS parameters, is the QoS vector of concrete service $cs_{i,j}$;

4. E represents the edges of the graph, which includes all links between concrete services of any two connected service candidate sets;

5. $D = \{d_1, d_2, \ldots, d_z\}$ represents a set of z data servers;

6. $DT^i = \{dt^1, dt^2, \ldots, dt^k\}$ represents a set of k data sets which are required by abstract service AS_i, and these data sets are distributed on a subset of D;

7. $Q_c = [Q^1_c, Q^2_c, \ldots, Q^u_c]$ $(1 \leq u \leq r)$ represents a set of global QoS constraints, which define requirements regarding the aggregated QoS values of the requested composite service.

In the traditional service composition problem, a single-objective function F may be chosen from any of the following ones:

1. *Inverse of overall utility*

$$F_1 = 1/\sum_{i=1}^{n}\sum_{j=1}^{m}\Big(U(cs_{i,j})x_{i,j}\Big) \tag{5.3}$$

where $U(cs_{i,j})$ is the utility of concrete service $cs_{i,j}$. The variable $x_{i,j}$ ($\sum_{j=1}^{m} x_{i,j} = 1$) represents only one concrete service is selected to implement each abstract service during the process of service composition, where $x_{i,j}$ is set to 1 if $cs_{i,j}$ is selected to implement abstract service AS_i and 0 otherwise.

2. *Overall Cost*

$$F_2 = \sum_{i=1}^{n}\sum_{j=1}^{m}\Big(Cost(cs_{i,j})x_{i,j}\Big) \tag{5.4}$$

where $Cost(cs_{i,j})$ is the cost of concrete service $cs_{i,j}$.

3. *Overall execution time*

$$F_3 = \sum_{i=1}^{n}\sum_{j=1}^{m}\Big(T_{et}(cs_{i,j})x_{i,j}\Big) \tag{5.5}$$

where $T_{et}(cs_{i,j})$ is the execution time of concrete service $cs_{i,j}$.

For the multi-objective context of the present work, the objective function F is considered as a two-dimensional vector $F = [F_2 \quad F_3]^T$ with no objective function considered as more important than the other. In this case, a set of Pareto-optimal solutions should be found.

5.3 Multi-Objective Data-Intensive Service Provision Based on an Ant Colony System

The proposed MOACO algorithm is based on ACS, which uses a unique ant colony to simultaneously minimize all functions. All objectives share the same pheromone trails. In every iteration, an ant k ($\forall k \in \{1, 2, \ldots, Nants\}$, $Nants$ is the number of ants), starts at the *start* vertex and successively chooses a next vertex from N_i^k, where subindex i represents that ant k is at vertex s_i (s_i is the concrete service which is chosen to implement abstract service AS_i). The set of unvisited vertices N_i^k, only includes all the direct successors of vertex s_i. For each ant k, a solution is found once it arrives at the *end* vertex. The key to MOACS for DISP_GQoSC is how to determine the state transition rule, the local updating rule, and the global updating rule.

5.3.1 State Transition Rule

When ant k arrives at vertex s_i, it will choose successor s_j to move to by applying the rule given by (5.6).

$$j = \begin{cases} \arg\max_{j \in N_i^k} \{ \left[\tau_{ij}\right] \left[[\eta_{ij}^C]^{\lambda} [\eta_{ij}^T]^{(1-\lambda)} \right]^{\beta} \}, & \text{if } q \leq q_0; \\ \text{randomly selected from } N_i^k \text{ according to } p_{ij}^k, & \text{otherwise.} \end{cases} \tag{5.6}$$

The variable q is a random number uniformly distributed in $[0, 1]$, q_0 $(0 \leq q_0 \leq 1)$ is a parameter, $\lambda = k/Nants$, β weights the relative importance of the objectives with respect to the heuristic information, and τ_{ij} is the pheromone density on edge (s_i, s_j). If $q > q_0$, j is randomly selected from N_i^k according to the probability distribution p_{ij}^k, which is given by (5.7).

$$p_{ij}^k = \begin{cases} \dfrac{\left[\tau_{ij}\right] \left[[\eta_{ij}^C]^{\lambda} [\eta_{ij}^T]^{(1-\lambda)} \right]^{\beta}}{\sum_{j \in N_i^k} \left[\tau_{ij}\right] \left[[\eta_{ij}^C]^{\lambda} [\eta_{ij}^T]^{(1-\lambda)} \right]^{\beta}}, & \text{if } j \in N_i^k \ ; \\ 0, & \text{otherwise.} \end{cases} \tag{5.7}$$

The heuristic information for the objective function F_2 considering service cost, and for the objective function F_3 considering service execution time, are calculated according to (5.8) and (5.9), respectively.

$$\eta_{ij}^C = 1/Cost(s_j) \tag{5.8}$$

$$\eta_{ij}^T = 1/T_{et}(s_j) \tag{5.9}$$

The cost and execution time of concrete service s_j, $Cost(s_j)$ and $T_{et}(s_j)$, are computed according to the QoS model as described in Chapter 4.

5.3.2 Local Updating Rule

When building up a solution of the problem, the ants apply a local pheromone updating rule, as shown in (5.10), immediately after they have crossed an edge (s_i, s_j).

$$\tau_{ij} = (1 - \xi)\tau_{ij} + \xi\tau_0 \tag{5.10}$$

The variable ξ $(0 < \xi < 1)$ is used to determine the local evaporation rate. At the beginning of the search process, a constant amount of pheromone is assigned to all edges, $\tau_{ij} = \tau_0$ $(\forall (i,j) \in E)$. τ_0 is initially calculated by $\tau_0 = 1/\Big(NN{*}C(S_0){*}T(S_0)\Big)$, as NN is the number of vertices in the directed graph, S_0 is the solution generated by the nearest neighbor heuristic [55]. $C(S_0)$ represents the overall cost of S_0, and $T(S_0)$ represents the overall execution time of S_0. This is due to the fact that it is a good practice to set the initial pheromone concentration to a value that is slightly higher than the expected amount of pheromone deposited by the ants in the iteration.

5.3.3 Global Updating Rule

The global non-dominated set of solutions, which the ants found from the beginning of the trial, is stored in the Pareto-optimal set GP. In each iteration, the solution found by each ant is recorded to a set P. After all ants arrive at the *end* vertex, the non-dominated set of solutions LP, are found from P. Each solution in LP is compared with the solutions in GP in order to check if it is non-dominated. If it is a new Pareto-optimal solution, it is added to GP and all solutions dominated by the added one are erased from GP. Since all non-dominated solutions are considered as best solutions for a multi-objective optimization problem, we suppose that all non-dominated solutions have the same quality. Therefore, for each solution $\psi^{GP} \in GP$, the pheromone information is globally updated according to (5.11).

$$\tau_{ij} = (1-\rho)\tau_{ij} + \rho/\Big(C(\psi^{GP}) * T(\psi^{GP})\Big), \quad \forall (i,j) \in \psi^{GP} \tag{5.11}$$

The variable ρ $(0 < \rho < 1)$ is the pheromone evaporation rate, $C(\psi^{GP})$ is the value of F_2 for a given solution ψ^{GP} while $T(\psi^{GP})$ is the corresponding value of F_3.

The implementation of the proposed MOACS is given in Algorithm 5.1.

Algorithm 5.1 Multi-objective data-intensive service composition based on MOACS

```
1: MaxIt = 300; // the maximum number of iterations
2: step = 0; // iteration counter
3: GP = ∅; // the global non-dominated set
4: while 1 do
5:    step = step + 1;
6:    P = ∅; // The solutions found by ants in each iteration
7:    set all ants at the start vertex;
8:    for each ant k do
9:       while ant k is not at the end vertex do
10:         construct a solution using (5.6), (5.7), (5.8), and (5.9);
11:         record the solution to P;
12:         apply the local updating rule (5.10);
13:      end while
14:   end for
15:   when all ants arrive at the end vertex, find the non-dominated set from P;
16:   update the global non-dominated set GP;
17:   apply the global updating rule (5.11) to GP;
18:   if step > MaxIt then
19:      break;
20:   end if
21: end while
22: output all solutions in the global non-dominated set.
```

5.4 Multi-Objective Data-Intensive Service Provision Based on a Genetic Algorithm

The MOGA for DISP_GQoSC is based on NSGA-II, but it needed to be customized and modified in order to handle the problem. In the proposed MOGA, the integer array coding scheme is used to encode chromosomes. The initial population is randomly created according to a directed graph. Only one branch of the conditional structures is selected according to the execution probability. The binary tournament selection operator and the single-point crossover operator are adopted in the proposed genetic algorithm. The mutation operator for each chromosome replaces the value of the gene with the assignment of another concrete service in the service candidate set randomly. The implementation of the proposed MOGA is given in Algorithm 5.2.

At the end of the algorithm, it needs to remove the duplication and then output the remainder of the population. This is because it is possible that the crossover

Algorithm 5.2 Multi-objective data-intensive service composition based on MOGA

1: $MaxIg = 300$; //the maximum number of generations
2: $iga = 0$; //generation counter
3: randomly create an initial *Population* with *Npop* individuals;
4: apply the fast non-dominated sort procedure to *Population*; //get the non-dominated fronts and the fitness value of each individual
5: calculate the crowding-distance value of each individual in *Population*;
6: **while** 1 **do**
7: $iga = iga + 1$;
8: select parents from *Population* using binary tournament selection strategy;
9: perform the single-point crossover operator and the mutation operator to the selected parents to create an *Offspring Population*;
10: combine *Population* and *Offspring Population* to get a *Combined Population*;
11: apply the fast non-dominated sort procedure to the *Combined Population*;
12: calculate the crowding-distance value of each individual in the *Combined Population*;
13: use the elitism mechanism to select *Npop* individuals from the *Combined Population* according to the fitness value and the crowding-distance value of each individual;
14: the new selected individuals create a new *Population*;
15: **if** $iga > MaxIg$ **then**
16: break;
17: **end if**
18: **end while**
19: output all individuals after removing the duplication from *Population*.

operator is applied to an individual and to itself.

5.5 Experiments and Analysis

5.5.1 Test Case Generation

The parameters of MOACS are: $\beta = 2$, $q_0 = 0.9$, $\rho = 0.1$, $\xi = 0.1$, $Nants = 100$. There are two branches in all the conditional patterns with the probability of 0.5. The parameters of MOGA are: the population is 100, the crossover probability is 0.9, and the mutation probability is $1/n$, where n is the number of abstract services in a composite service. For MOACS and MOGA, a reasonable set of parameter values was chosen. The author will try to find better parameter settings in future work to be discussed in Chapter 8.

For the purpose of the evaluation, different scenarios were considered where a

composite application consists of n abstract services, and n varies in the experiments between 10 and 50, in increments of 10. There are m concrete services in each service candidate set, and m varies in the experiments between 10 and 100, in increments of 10. Each abstract service requires a set of k data sets, and k is fixed at 10 in the experiments.

A scenario generation system is used to generate all scenarios, as described in Section 4.5. Then every scenario was performed with 21 runs (with 11 being the median line of all 21 attainment surfaces), and every run was stopped after 300 generations. All runs of the same scenario use the same data, and the average results over 21 independent runs are reported.

5.5.2 Results Analysis

Table 5.2 shows the means of the computation time of each scenario. In the upper half of Table 5.2, the second column indicates that the MOACS needs more computation time when the number of concrete services increases, while the third column shows the computation time of MOGA remains almost steady as the number of concrete services increases. This is because, by using the integer array coding scheme, the change in the number of concrete services will not influence the length

Table 5.2: Means of the computation time

Scenarios	***MOACS***	***MOGA***
n is fixed at 10, m varies between 10 and 100, in increments of 10	22.3333	29.6667
	23.6190	30.0952
	24.4762	29.8571
	25.0952	31.3333
	26.5238	29.9524
	27.1905	31.2857
	27.6667	31.1429
	29.5238	30.9048
	30.6667	30.7143
	30.4286	30.2381
m is fixed at 50, n varies between 10 and 50, in increments of 10	26.5238	29.9524
	61.2381	30.7143
	98.9048	30.2381
	141.9524	31.1905
	285.5714	31.7143

of the genome. The computation time of both MOACS and MOGA increases when the number of abstract services increases, which is indicated by the lower half of Table 5.2. The upper half of Table 5.2 indicates that when the number of abstract services and concrete services is small, MOACS is better than MOGA, since the means of the computation time of MOACS are lower than those of MOGA except in the scenario where $n = 10$ and $m = 100$. Meanwhile, the lower half of Table 5.2 indicates that MOGA is more scalable than MOACS when there is a large number of concrete services and abstract services.

Table 5.3 gives the means of the ONVG. By comparing the second and third columns of the upper half of Table 5.3, we conclude that MOGA can get more non-dominated solutions than MOACS except in the scenario where $n = 10$ and $m = 10$. On the other hand, the lower half of Table 5.3 indicates that MOACS can find more non-dominated solutions than MOGA when the number of abstract services increases except in the scenario where $n = 10$ and $m = 50$.

Table 5.4 provides the means of the $PS(A)$. By comparing the second and third columns of Table 5.4, we conclude that MOACS is better than MOGA since MOACS always leads to a higher value of $PS(A)$.

Table 5.5 presents the means of the C metric. The value in the second column

Table 5.3: Means of ONVG

Scenarios	***MOACS***	***MOGA***
n is fixed at 10, m varies between 10 and 100, in increments of 10	14.9048	14.8571
	32.0000	36.0476
	27.4762	34.4286
	22.0952	27.5714
	35.8571	49.8045
	25.3333	28.0952
	20.1905	27.4286
	38.5238	46.0476
	34.8095	38.3333
	15.3333	16.5714
m is fixed at 50, n varies between 10 and 50, in increments of 10	35.8571	49.8045
	75.4762	69.6196
	90.7619	83.1905
	114.9524	82.3810
	131.5714	84.1429

Table 5.4: Means of $PS(A)$

Scenarios	***MOACS***	***MOGA***
n is fixed at 10, m varies between 10 and 100, in increments of 10	82.65%	82.64%
	84.86%	84.77%
	85.29%	85.25%
	84.65%	84.61%
	85.88%	85.87%
	86.30%	86.12%
	86.17%	86.02%
	86.12%	85.92%
	86.72%	86.38%
	86.33%	86.28%
m is fixed at 50, n varies between 10 and 50, in increments of 10	85.88%	85.87%
	68.96%	68.31%
	51.77%	49.81%
	38.01%	35.96%
	24.74%	21.90%

Table 5.5: Means of C Metric

Scenarios	***C(MOACS,MOGA)***	***C(MOGA,MOACS)***
n is fixed at 10, m varies between 10 and 100, in increments of 10	0.9524	0.9524
	0.6298	0.6298
	0.6037	0.6037
	0.8429	0.8429
	0.8311	0.8311
	0.5982	0.5982
	0.6219	0.6219
	0.5127	0.5127
	0.4523	0.4523
	0.5189	0.5189
m is fixed at 50, n varies between 10 and 50, in increments of 10	0.8311	0.8311
	0.4530	0.4530
	0.4452	0.4452
	0.2338	0.2338
	0.2094	0.2094

is equal to the value in the third column of Table 5.5. The results indicate that the convergence of the Pareto-optimal solutions of MOACS and MOGA has never been different, so we cannot say one is better than the other with respect to the C metric.

Figure 5.2 and Figure 5.3 depict the median summary attainment surface of MOACS and MOGA with respect to a varying number of concrete services. Fig-

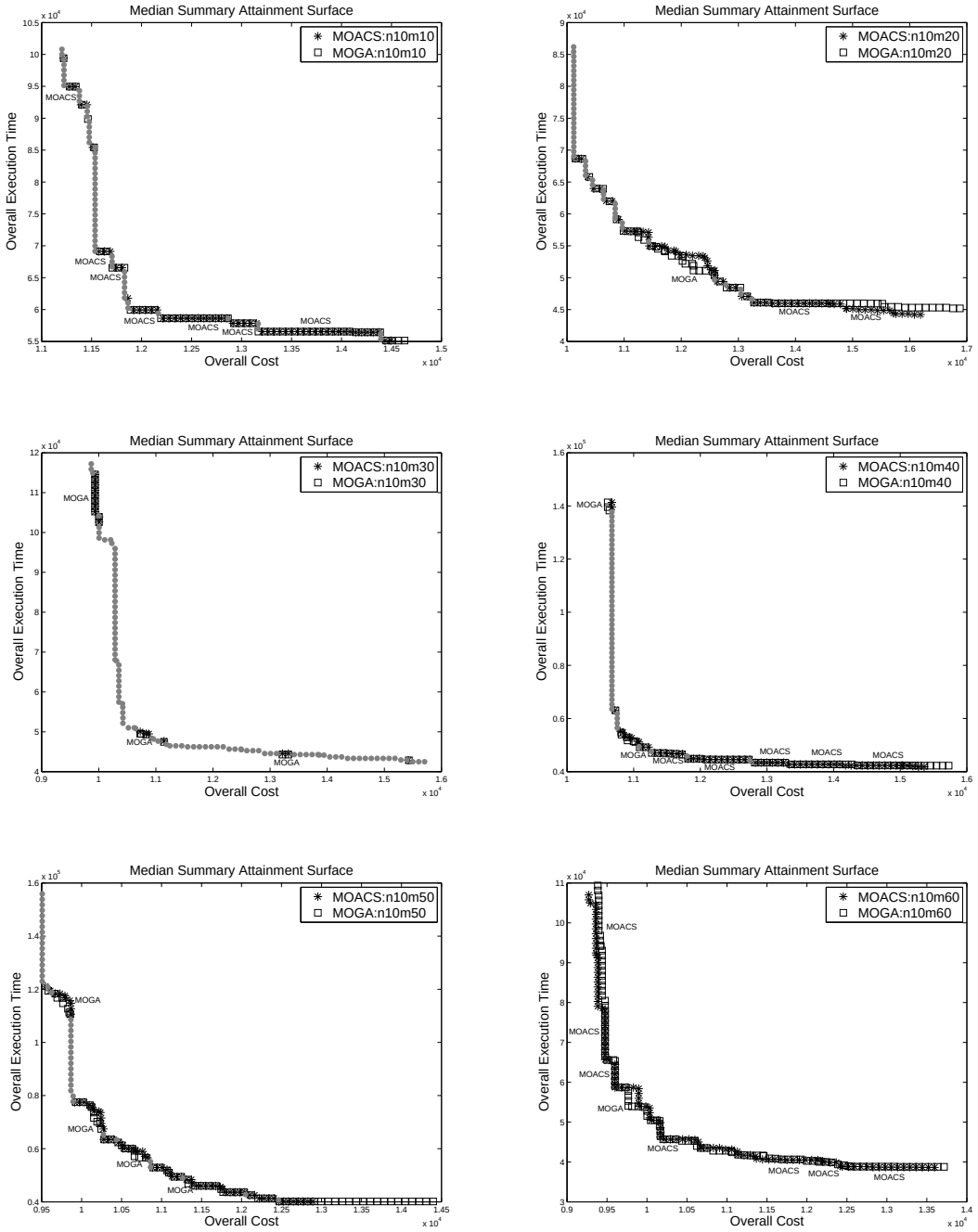


Figure 5.2: Median summary attainment surface vs. number of concrete services - part I

ure 5.4 shows the median summary attainment surface of MOACS and MOGA with respect to a varying number of abstract services.

In Figures 5.2-5.4, the regions where there is no difference between the points of the median attainment surfaces of the two algorithms are indicated in gray dots, whereas those regions where the points of the two surfaces are found to be different from each other are plotted in stars and squares, respectively.

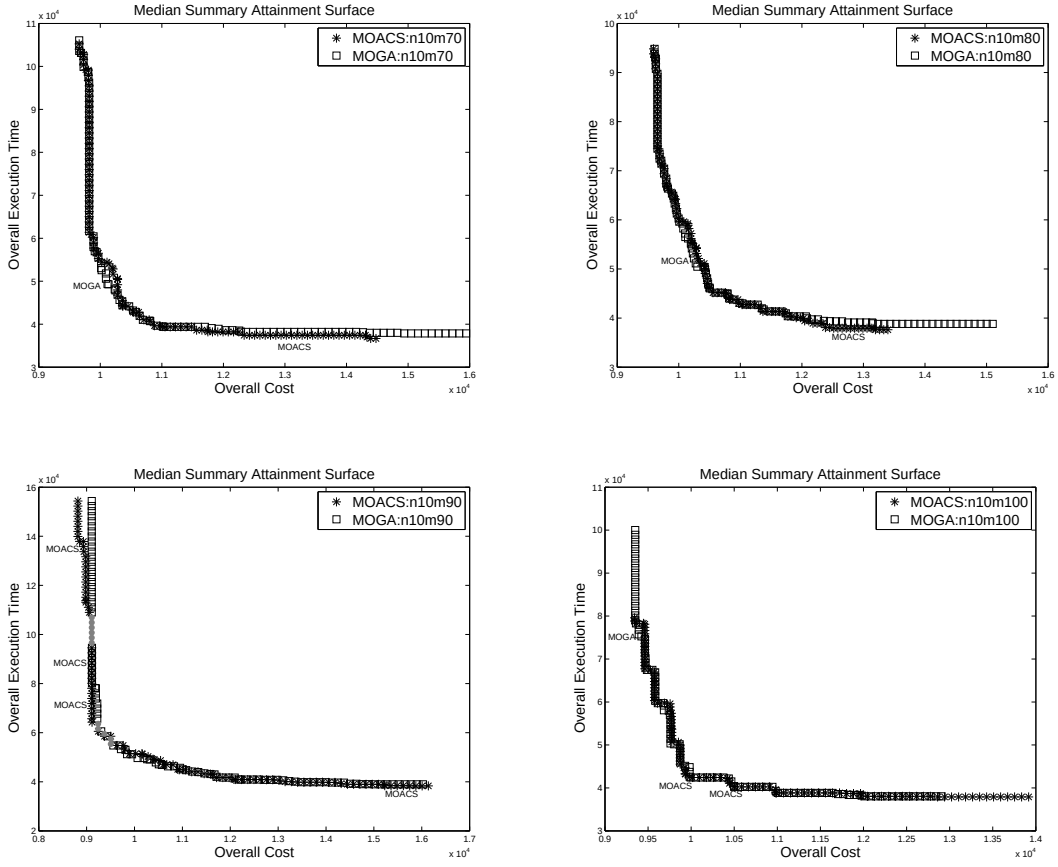


Figure 5.3: Median summary attainment surface vs. number of concrete services - part II

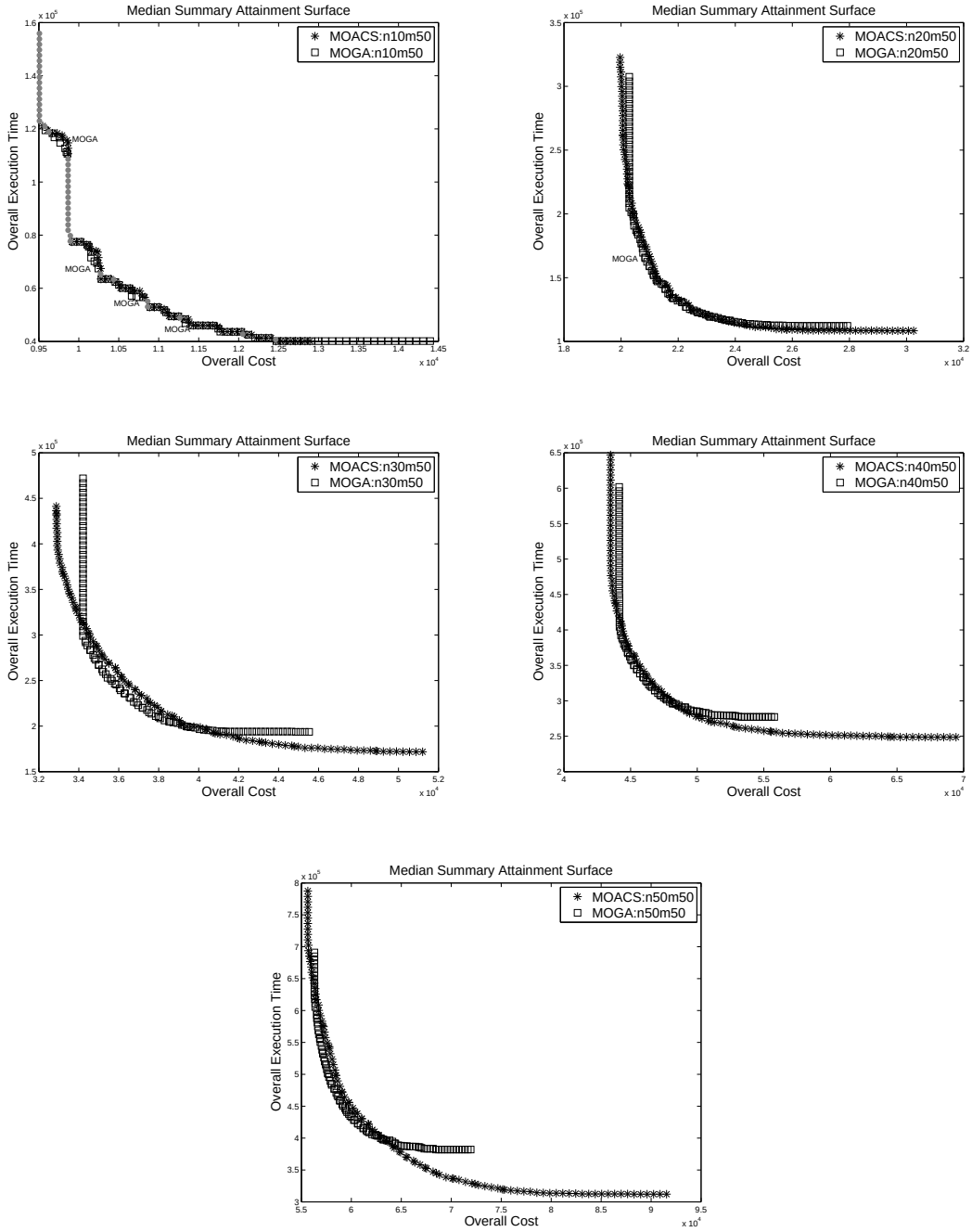


Figure 5.4: Median summary attainment surface vs. number of abstract services

In the regions where the points of the two surfaces are found to be different from each other, there are three situations: 1) if the points of the median attainment surfaces of MOACS dominate those of MOGA, then the label MOACS is put near the points, 2) if the points of the median attainment surfaces of MOGA dominate those of MOACS, then the label MOGA is put near the points, and 3) if there is no domination relationship between the points of the median attainment surfaces of

MOACS and MOGA, then no label was put.

According to Figure 5.2 and Figure 5.3, MOACS is better than MOGA except for a small number of points of all the median attainment surfaces, since the points of the median attainment surface of MOACS are closer to the origin of the rectangular coordinate system. According to Figure 5.4, both of MOACS and MOGA have some points where there is no difference between them in the scenario where $n = 10$ and $m = 50$. In the scenario where $n = 20$ and $m = 50$, there are some points where MOGA is better than MOACS. In the remaining scenarios of Figure 5.4, there is no domination relationship between the points of the median attainment surfaces of the two algorithms. But it is clear that MOGA generates more useful solutions than MOACS, which is indicated by the middle parts of the median attainment surfaces of both algorithms.

5.6 Related Work

Most of the past studies applied the aggregated single-objective ant colony optimization algorithms and genetic algorithms, which combined several objectives into a single objective value to tackle service composition problems.

Fang et al. [66] used one pheromone matrix in their MOACO algorithm, and the heuristic information was the value of a function of all objectives. Based on this study, Wang and He [174] adopted the chaos variable and proposed a chaos MOACO algorithm. On the other hand, Xia et al. [192] investigated a dynamic ACO algorithm with multi-pheromone to optimize service composition. They set various pheromones to denote different QoS attributes.

A few research studies have addressed the QoS-aware service composition problems by using MOGAs [40, 43, 118, 166], but their service composition models were relatively simple. For example, a few studies did not consider global QoS constraints [40, 43, 166], and the study [118] did not show details about solutions for different objective functions and did not present experimental results, either.

Wada et al. [172] designed an optimization framework to address SLA-aware service composition and conducted two MOGAs to produce a set of Pareto-optimal solutions. Each concrete service was characterized with three QoS attributes: through-

put, latency, and cost. In order to improve the service's throughput and fault tolerance, a set of service instances for an abstract service were deployed in redundant parallel and they were used with equal probability. However, their simulation used a workflow which consisted of only four abstract services, and each abstract service was associated with three concrete services, which could not address the scalability of the algorithms. Moreover, each gene was encoded as the number of instances for each concrete service, hence the greater the number of concrete services, the longer the length of the genome will be.

Li et al. [110] applied an improved version of NSGA-II to solve QoS-based service composition, but they only considered two simple scenarios. One scenario had four abstract services and each abstract service was associated with 90 concrete services, the other had 12 abstract services and each abstract service was associated with 30 concrete services. They only measured the number of non-dominated solutions, and their experimental results only indicated that NSGA-II could be used for service composition. Similarly, Yao and Chen [197] applied NSGA-II to solve a similar problem, but the experimental part was also too simple and we could not get sufficient information about the performance of NSGA-II. A comprehensive evaluation of NSGA-II for service composition is much needed.

Furthermore, GAs are unconstrained optimization algorithms. The mutation operator and the crossover operator are blind to constraints, and they might result in infeasible individuals. Therefore, it is necessary to develop approaches to handling constraints. Constraint-handling techniques in GAs are mainly based on four different methods: penalty functions, special operators, repair algorithms, and separation of constraints and objectives. Deb et al. [52] proposed a constraint-handling method for NSGA-II, which was based on the tournament selection algorithm and it separated the constraints and objectives. The tournament selection algorithm is much better than a number of other existing constraint-handling approaches, as confirmed in other studies [49, 51]. In the MOGA of this research, this constraint-handling method was adopted.

The penalty function is the most common approach in GAs to handle constraints. It transforms a constrained-optimization problem into an unconstrained one by adding or subtracting a certain value to or from the objective function based

on the extent of constraint violation in a certain solution. This method has been adopted by a few studies [33, 75, 119, 133, 162, 167, 204]. The special operator is designed to deal with infeasible individuals in order to preserve feasibility. Claro et al. [44] adopted a crowded-comparison operator to deal with infeasible individuals. On the other hand, the repair algorithm is used on the obtained solutions to move infeasible solutions closer to the feasible solution space. A repair algorithm was designed in a few studies [115, 193].

The comparison of the proposed algorithms with other studies [40, 43, 66, 110, 118, 172, 174, 192, 197, 206, 208] is shown in Table 5.6. The studies listed in Table 5.6 used ACO algorithms or GAs to solve a similar problem to this research. The contributions of this work which differentiate it from these studies are as follows.

1. Compared with the studies [43, 66, 110, 118, 174, 206], the sequential, parallel, and conditional structures are considered in the proposed MOACS and MOGA.

2. The proposed MOACS and MOGA are suitable to resolve the service selection problem with global QoS constraints, which is different from the studies [40, 43, 118, 197, 206].

3. In the proposed MOACS, all objectives share the same pheromone trails and all the objectives have the same importance, in this case, a set of Pareto-optimal solutions are found. Only single solutions were given by other studies [40, 66, 192, 208].

4. Five performance metrics were adopted to measure the Pareto-optimal solutions, which is different from all the other studies. One performance metric cannot adequately measure the performance of multi-objective optimization algorithms, as explained in Section 5.1.3.

5. The NSGA-II was customized and modified in order to handle the multi-objective data-intensive service provision problem. A comprehensive evaluation of MOGA for service composition, which was not presented in other studies [40, 43, 110, 118, 172, 197], is provided in this work.

Table 5.6: The comparison of the MOACS and MOGA with other studies

Articles	Composition structure	Global QoS constraints	Performance metric(s)	Method of evaluation	Test scenarios	Outcomes of the experiment
[66]	Sequential	Considered	The number of non-dominated solutions and the computation time	Comparison with a MOGA	four scenarios	The MOACO algorithm was better than the MOGA.
[174]	Sequential	Considered	The number of non-dominated solutions and the computation time	Comparison with a MOACO and a MOGA	three scenarios	The multi-objective chaos ACO algorithm was better than the MOACO algorithm and the MOGA.
[206]	Sequential	None	Computation time	Comparison with a MOGA	10 scenarios	The MOACO algorithm could find near-optimal solutions and was scalable.
[208]	Complex structures	Considered	Computation time	Comparison with the exhaustive searching method	varying number of concrete services	ACS had higher convergence speed and needed fewer iteration numbers.
[192]	Complex structures	Considered	Computation time	Comparison with a GA and an ACO algorithm	varying number of concrete services	Dynamic ACO algorithm had better performance than the ACO algorithm and the GA.
[40]	Complex structures	None	Applicability	None	Single scenario	The applicability of the multi-objective evolutionary algorithm for service composition was verified.
[43]	Sequential	None	The number of non-dominated solutions	None	Two test groups	The results obtained from the NSGA-II validated the feasibility of the proposed service selection approach.
[110]	Sequential	Considered	The number of non-dominated solutions	None	two scenarios	The results indicated that the NSGA-II was fit for solving the service composition problem.
[172]	Complex structures	Considered	The quality of solutions	Comparison with the NSGA-II	Single scenario	The performance of the proposed algorithms outperformed the linear programming method and the NSGA-II.
[118]	Sequential	None	None	None	None	The proposed MOGA was not simulated with experiments.
[197]	Complex structures	None	The fitness value	Comparison with a GA	Single scenario	NSGA-II had quicker convergence than the GA.
This work	Complex structures	Considered	Five performance metrics	Comparison the MOACS with the MOGA	15 different scenarios	When a large number of concrete services are available for each abstract service, the MOGA can achieve better solutions. On the other hand, whenever the number of concrete services available is small, the MOACS should be preferred.

5.7 Summary

This chapter proposed a multi-objective ant colony system and a multi-objective genetic algorithm for the data-intensive service provision problem. The goal of the

problem was to efficiently obtain a set of non-dominated solutions that simultaneously minimized the total cost and the total execution time. Because ant colony optimization algorithms and genetic algorithms are mainly used to solve multi-objective service composition problems, detailed comparisons of the proposed algorithms with other studies based on the two algorithms were provided. Both algorithms were simulated with many different scenarios, and they were compared according to five performance metrics.

In the provision of dynamic data-intensive services, the states of services and the cost and response time of data sets may change over time. The ant colony system will be exploited to tackle this issue in the next chapters.

Chapter 6

Ant Colony System for Dynamic Data-Intensive Service Provision

This chapter conducts an investigation in applying an ant colony system to the dynamic data-intensive service provision problem. Specifically, the author considers changing the QoS attributes of services and replacing a certain number of services with new ones at different frequencies. In order to adapt the ant colony system to handle the dynamic scenarios, several pheromone modification strategies in reaction to changes of the optimization scenarios are investigated. The aim of the strategies is to find a balance between preserving enough old pheromone information to speed up the search process, and resetting enough new pheromone information to facilitate the ants to find a new solution for the changed scenarios. The strategies differ in their degree of reinitialized pheromone values with respect to the information that has been used to decide the amount of pheromone values. Moreover, the behaviors of different strategies for modifying pheromone information are compared.

6.1 Introduction

According to the hierarchical taxonomy of Web service concretization approaches described in Chapter 2, many methods have been applied to tackle Web service selection and composition problems, such as the local optimization methods [20, 108, 134, 203], the MIP approaches [12, 139], the heuristic-based algorithms [21, 25, 120], and the bio-inspired algorithms [33, 119, 185, 206]. By a detailed analysis of each

method, the author found that bio-inspired algorithms could overcome the challenging requirements of the typical data-intensive service provision. In the bio-inspired algorithms, the ant colony optimization algorithms can run continuously and are generally capable of adapting to changes of an optimization problem. The former two chapters has presented the ant colony optimization algorithms and the genetic algorithms to tackle the single-objective and multi-objective data-intensive service provision problems. Compared with genetic algorithms and the MIP approach, the ant colony optimization algorithms can find solutions more quickly in dynamic environments without restarting the optimization process, since the ants can react explicitly to the changes based on the pheromone information. This chapter explores strategies to apply ACS to solving the dynamic scenarios, where services change at certain intervals.

In a dynamic environment, the service composition optimization process should be conducted repeatedly when the changes of the states of services occur, such as the changes of QoS attributes of services, the discontinuation of services, and the increase of new services. If each change in the composition is not too significant, it is likely that the solution to the changing optimization scenario will be related to the old ones to some extent. The old optimal solution can be reused to find a good solution quickly after a change occurred. A simple restart of the optimization process, which discards the former solution after a change has occurred, might not be a good strategy in most cases. On the other hand, if too much old pheromone information is maintained, the ants will be stuck in a local optimal solution. The key is to find a trade-off between preserving old pheromone information and modifying enough new pheromone information to allow the ants to find solutions for the new search space in later iterations.

6.2 Dynamic Ant Colony System

This section discusses strategies for modifying the pheromone information to adapt ACS to handle the dynamic scenarios, when new services are provided, some services are discontinued, or the QoS attributes of some services are changed. When a change of the search space occurred, it is necessary to initialize the pheromone information

for the new services and to modify the pheromone information for the old services that are still in provision. The solutions that were bad before a change, might be good afterwards. If ACS has converged to a path on which the amount of pheromone trails becomes much higher than on all other paths, this path prevents the ants from taking another path when a change occurs on it, and the result will be a sub-optimal solution. We need to modify the pheromone information to find a balance between preserving enough old information to speed up the search process, and modifying enough new information to facilitate the ants to find a new solution for the changed scenarios.

6.2.1 Pheromone Modification Approaches

The literature presents three types of pheromone modification approaches: global approaches, local approaches, and combined approaches.

The global pheromone modification approaches, such as the approach to reinitializing the whole pheromone matrix, and the approach to increasing the pheromone values proportionately to their difference to the maximum pheromone value, are designed to reset all the pheromone values to a certain degree [63, 77, 78]. However, the global approaches do not take into account where the changes of the search space actually occurred. Moreover, when we have a big search space, if a change occurs somewhere near the edge of the space, too much information might be lost by using global pheromone modification approaches.

The local pheromone modification approaches are designed to reset the pheromone values near the changes. Often, solutions to the changed scenarios will differ only locally from solutions to the old one. Therefore, the resetting of pheromone values should be performed in the close vicinity of the changes. Usually, the pheromone values are modified based on the factors contributing to an ant's local decisions. That is to say, modifying pheromone information is based on the heuristic information or the pheromone information.

The combined pheromone modification approaches could be advantageous in a situation where a local resetting is necessary to a change while a global resetting is needed to facilitate the algorithm.

6.2.2 Pheromone Modification Strategies

The general approach of ACS for the problem in this research is described in Chapter 4. The new strategies applied to the dynamic scenarios are presented in this section. As described in Chapter 4, the problem is modeled as a graph with a *start* vertex and an *end* vertex. In Figure 6.1, $cs_{r,s}$ represents the s^{th} concrete service in the corresponding service candidate set cs_r for abstract service r. ACS consists of several iterations where each ant in the ant colony individually constructs a candidate solution for the problem in each iteration. Beginning with the *start* vertex, each ant builds up a solution iteratively by always selecting the next vertex based on the pheromone information τ_{ij} and the heuristic information η_{ij}. All edges of the graph are initialized with a certain amount of pheromone τ_0 at the start.

The strategies the author follows to modify the pheromone values are inspired by the study [77]. Suppose a change occurs at vertex $j \in cs_{cj}$, the set of its direct successors is denoted by cs_{sj} and the set of its direct predecessors is denoted by cs_{pj}. A new parameter as the reset-value, denoted by γ_i ($\gamma_i \in [0, 1]$), is introduced to determine the amount of reinitialized pheromone values on edges adjacent to vertex i ($\forall i \in cs_{cj}$), according to (6.1) and (6.2).

$$\tau_{ik} = (1 - \gamma_i)\tau_{ik}^{old} + \gamma_i \tau_0, \quad \text{if } k \in cs_{sj}. \tag{6.1}$$

$$\tau_{ki} = (1 - \gamma_i)\tau_{ki}^{old} + \gamma_i \tau_0, \quad \text{if } k \in cs_{pj}. \tag{6.2}$$

The variables τ_{ik}^{old} and τ_{ki}^{old} are the pheromone values on edge (i, k) and edge (k, i)

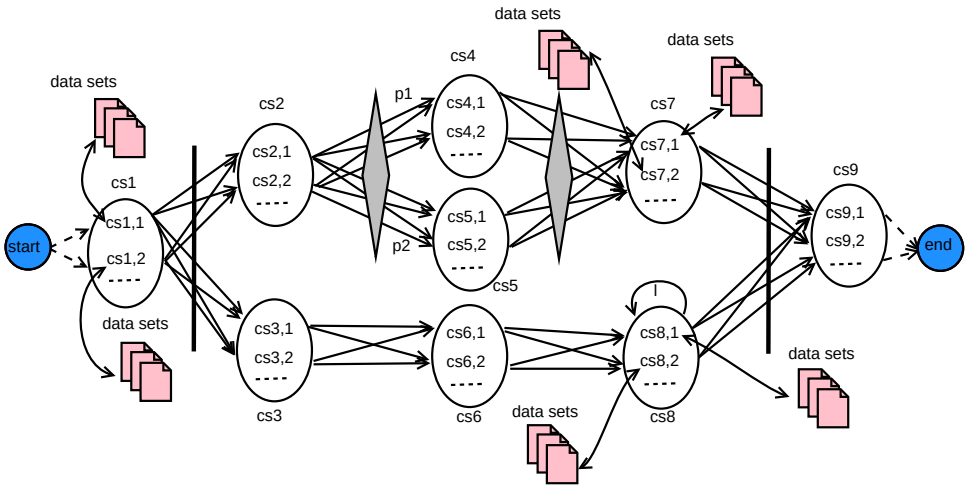


Figure 6.1: A graph for the data-intensive service provision problem

before a change occurs. In fact, pheromone values are not completely modified, but a trace of old values remains. If there is a new service, namely s, it receives reset-value of $\gamma_s = 1$. More details are described in the following subsections about how the different strategies assign the values γ_i.

R-strategy

The first strategy is denoted as the R-strategy, which reinitializes all the pheromone values by the same degree. Each vertex i is assigned a strategy-specific parameter $\gamma_R \in [0, 1]$ as its reset-value, namely $\gamma_i = \gamma_R$.

η-strategy

The second strategy is denoted as the η-strategy, which uses the heuristic information to decide the amount of pheromone trails. Each vertex i is given a value γ_i proportionate to its utility related to vertex k, according to (6.3) and (6.4).

$$\gamma_i = \max\{0, U_{ik}^{\eta}\}, \quad \text{if } k \in cs_{sj} \tag{6.3}$$

$$\gamma_i = \max\{0, U_{ki}^{\eta}\}, \quad \text{if } k \in cs_{pj} \tag{6.4}$$

where

$$U_{ik}^{\eta} = 1 - \frac{\eta_{avg}}{\gamma_E * \eta_{ik}}, \quad \eta_{avg} = \frac{\sum_{k=1}^{m_1} \eta_{ik}}{m_1}$$

$$U_{ki}^{\eta} = 1 - \frac{\eta_{avg}}{\gamma_E * \eta_{ki}}, \quad \eta_{avg} = \frac{\sum_{k=1}^{m_2} \eta_{ki}}{m_2}$$

The variables m_1 and m_2 are the number of concrete services in the candidate sets cs_{sj} and cs_{pj}. The strategy-specific parameter $\gamma_E \in (0, \infty)$ scales the width of the utility proportion.

τ-strategy

The third strategy is denoted as the τ-strategy, which used the pheromone information to decide the reset-values, according to (6.5) and (6.6).

$$\gamma_i = \min\{1, \gamma_T * \tau_{ik}\}, \quad \text{if } k \in cs_{sj} \tag{6.5}$$

$$\gamma_i = \min\{1, \gamma_T * \tau_{ki}\}, \quad \text{if } k \in cs_{pj} \tag{6.6}$$

The strategy-specific parameter $\gamma_T \in (0, \infty)$ is used to limit the result to 1 for application of (6.1) and (6.2).

G-strategy

The fourth strategy is denoted as the G-strategy, which belongs to the global pheromone modification approach and reinitializes the whole pheromone matrix. That is to say, $\gamma_i = 1$, $\forall i, k \in V$, and V is the set of vertices of the graph.

6.3 Experiments and Analysis

For the purpose of this evaluation, the dynamic ACS with two different case studies were tested: one with nine abstract services as shown in Figure 6.1, the other with thirty abstract services. The second case study was created by either placing a candidate set into Figure 6.1 or adding another composition structure as substructure. For each graph, it is supposed that there are twenty concrete services in each service candidate set and each concrete service requires a set of ten data sets. Then ten concrete services in each service candidate set are reserved to form a spare pool of services before the start of the algorithm, leaving each service candidate set with ten concrete services. The criteria to measure the performance of ACS are the utility of the best solution and the loss in quality of the solution compared with the MIP approach.

As mentioned in Section 6.1, the dynamical changes that can occur to the problem are the changes of QoS attributes of services (replacement of vertices), the discontinuation of services (deletion of vertices), and the increase of new services (insertion of vertices). During the run of the algorithms, the actual scenario was changed

every t iterations by replacing k services between the actual instance and the spare pool. All combinations of parameter values $k \in \{1, 5, 10\}$ and $t \in \{50, 100, 200\}$ were tested. For each configuration (k, t), 20 test runs were performed and every run was stopped after 1000 iterations. Then the average values over these 20 runs were used for comparison. When deciding which vertex to replace, the concrete service was chosen at random. This procedure continues until the number of replacing services is k.

For all combinations of parameters (k, t), ACS with the global pheromone modification approach and the local pheromone modification approach were tested. For the two approaches, four strategies were compared:

1. R-strategy with all strategy-specific parameters $\gamma_R \in \{0.25, 0.5, 0.75, 1.0\}$.
2. η-strategy with all strategy-specific parameters $\gamma_E \in \{0.5, 1.0, 2.0, 5.0\}$.
3. τ-strategy with all strategy-specific parameters $\gamma_T \in \{0.5, 1.0, 2.0, 5.0\}$.
4. G-strategy with no strategy-specific parameter in it.

6.3.1 Results of Case Study One

A detailed view of the optimization behavior for each individual strategy and its strategy-specific parameter are shown in Figures 6.2-6.5. These figures show the case of $(k, t) = (1, 50)$, small changes occur frequently. In particular, the effect on increasing the strategy-specific parameter values can be observed. After each change occurs, we see a hop of the utility of the best solution having been found in these figures. For example, after $50th$ iteration a change occurs for the first time and after $100th$ iteration a change occurs for the second time. Then the ants need to find a solution for the new scenario between the $51st$ iteration and the $100th$ iteration and let fi be the iteration where the best utility for the new scenario presents the first time. The smaller value of $fi - 50$ suggests a quick recovery from the change occurred in the $51st$ iteration. From these figures, we observe that the best utility is the same after changes occurred in the interval $[1, 200]$, $[251, 350]$, $[451, 700]$, and $[801, 1000]$. That is to say, the old feasible solution is the same as the new feasible solution after changes occurred in these intervals. For the other combination of k

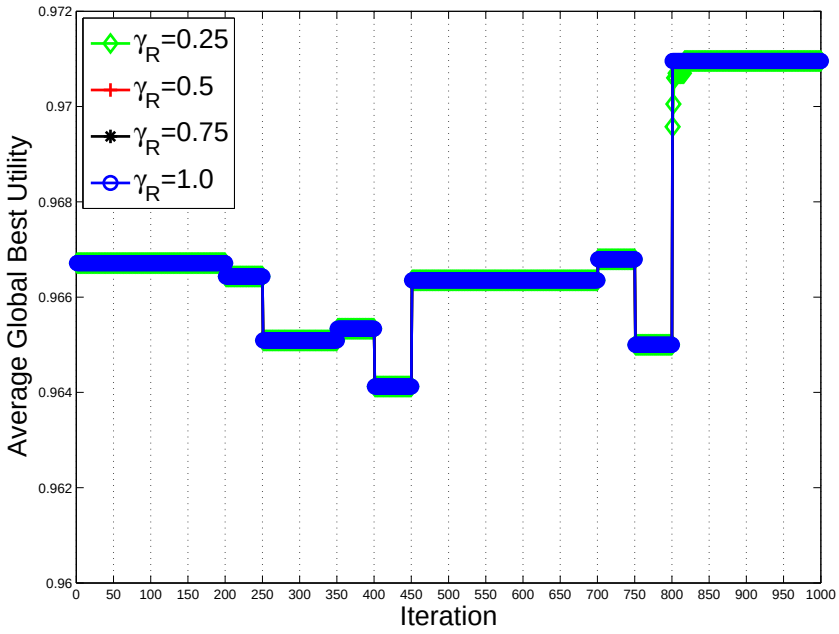


Figure 6.2: The optimization behavior of R-strategy with all parameters in case study one

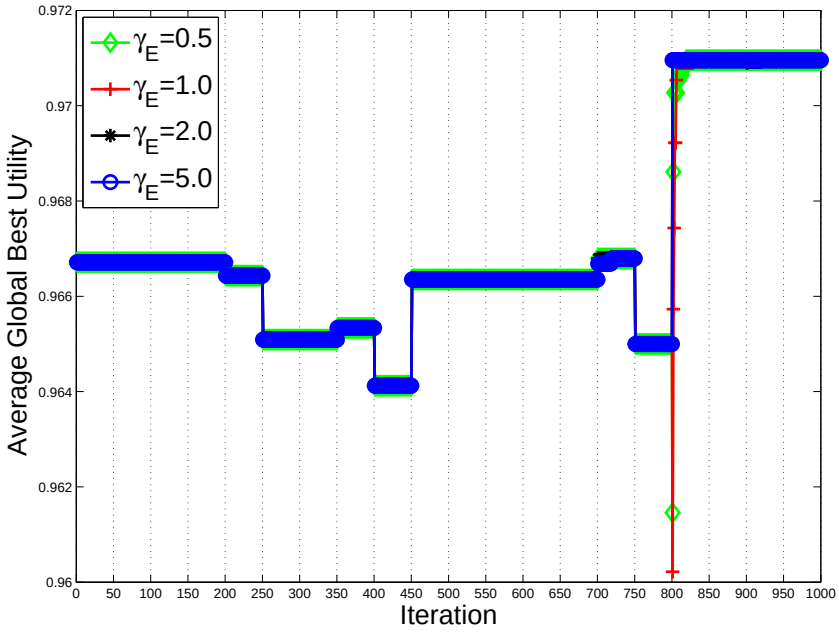


Figure 6.3: The optimization behavior of η-strategy with all parameters in case study one

and t, the optimization behaviors for each individual strategy is similar with those in Figures 6.2-6.5.

Besides the optimization behavior of each strategy, the loss in quality of the

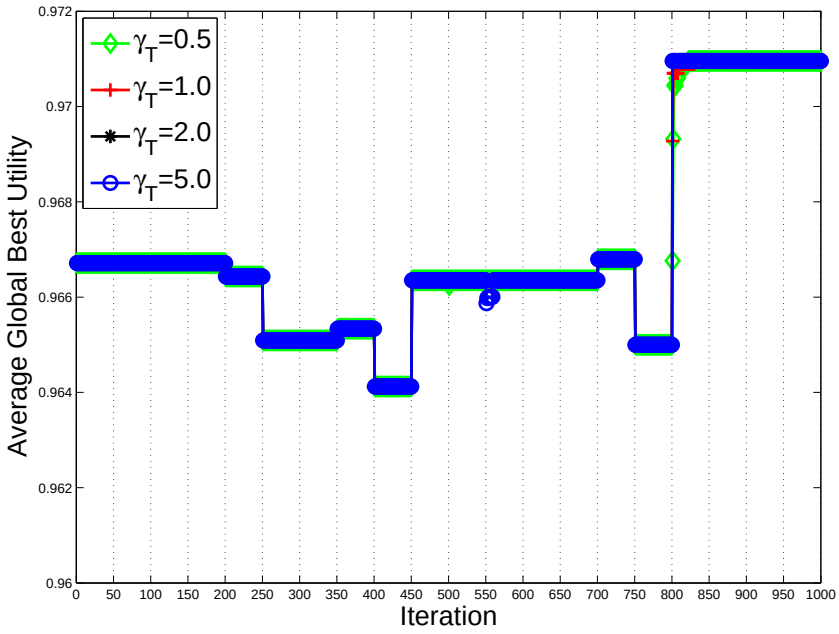


Figure 6.4: The optimization behavior of τ-strategy with all parameters in case study one

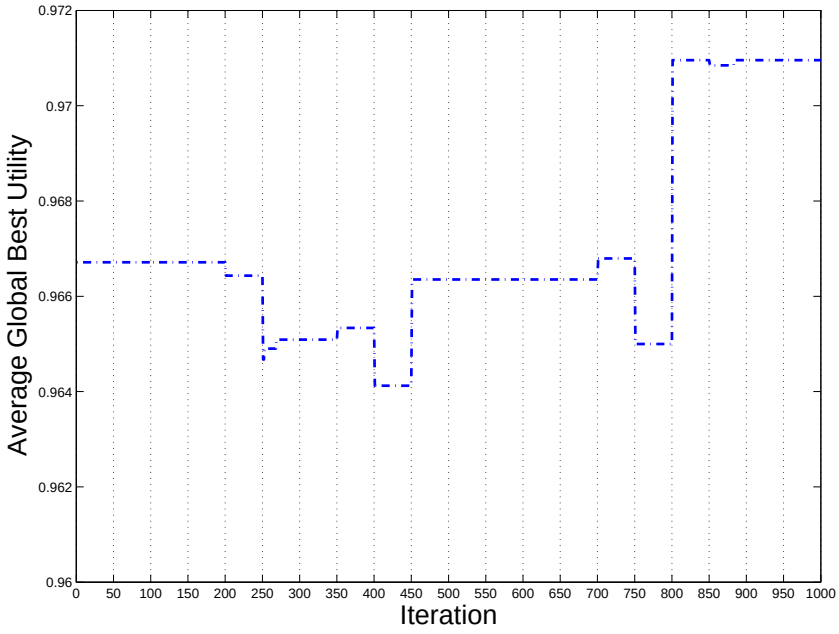


Figure 6.5: The optimization behavior of G-strategy in case study one

best solution was also recorded, compared with the solution found by the MIP approach. As ACS is sub-optimal, the quality of the solution obtained by ACS has been evaluated through comparing it with the optimal solution obtained by the MIP approach. The global best utility of the solution obtained by the MIP approach is

denoted by U_{global}, and the global best utility of the solution obtained by ACS is denoted by U_{acs}. Then the loss in the quality is computed, which is given by (6.7).

$$loss = \frac{U_{global} - U_{acs}}{U_{global}} * 100\% \tag{6.7}$$

For each strategy and its strategy-specific parameter, the average results of 20 runs of 1000 iterations was recorded. Figure 6.6 gives a more detailed view of the performance of each strategy and its different strategy-specific parameters for the case of $(k, t) = (1, 50)$. The upper row shows the loss of the quality for R-strategy with the strategy-specific parameter $\gamma_R \in \{0.25, 0.5, 0.75, 1.0\}$ from left to right, the middle two rows show the loss of the quality for η-strategy and τ-strategy with the strategy-specific parameter $\gamma_E, \gamma_T \in \{0.5, 1.0, 2.0, 5.0\}$ from left to right, and the whole lower row shows the loss of the quality for G-strategy since there is no strategy-specific parameter in it. The smaller the loss's value, the brighter the grid will be. Judging from the gray scale of the grid, the best utility is the R-strategy with $\gamma_R \in \{0.5, 0.75, 1.0\}$ and τ-strategy with $\gamma_T = 2.0$. The η-strategy with $\gamma_E = 2.0$ is also able to achieve good solutions. The G-strategy is better than the η-strategy with $\gamma_E \in \{0.5, 1.0\}$ and τ-strategy with $\gamma_T = 0.5$.

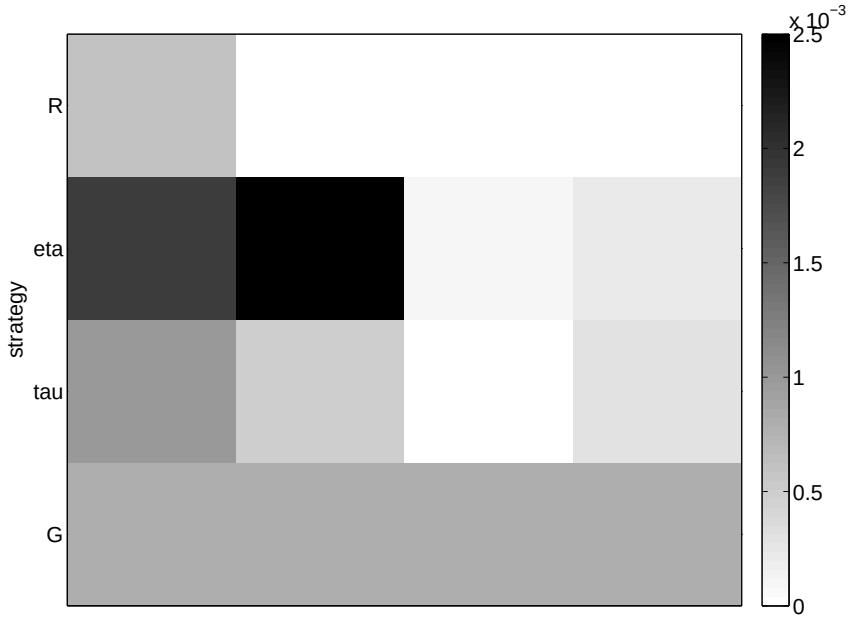


Figure 6.6: The loss in quality of the best found solution of all strategies for the case of $(k, t) = (1, 50)$ in case study one

Figures 6.7-6.10 show the performance of each strategy and its different strategy-specific parameters for all the cases of the combination of k and t. Judging from the "average darkness", the best overall strategy is the R-strategy with $\gamma_R = 1.0$,

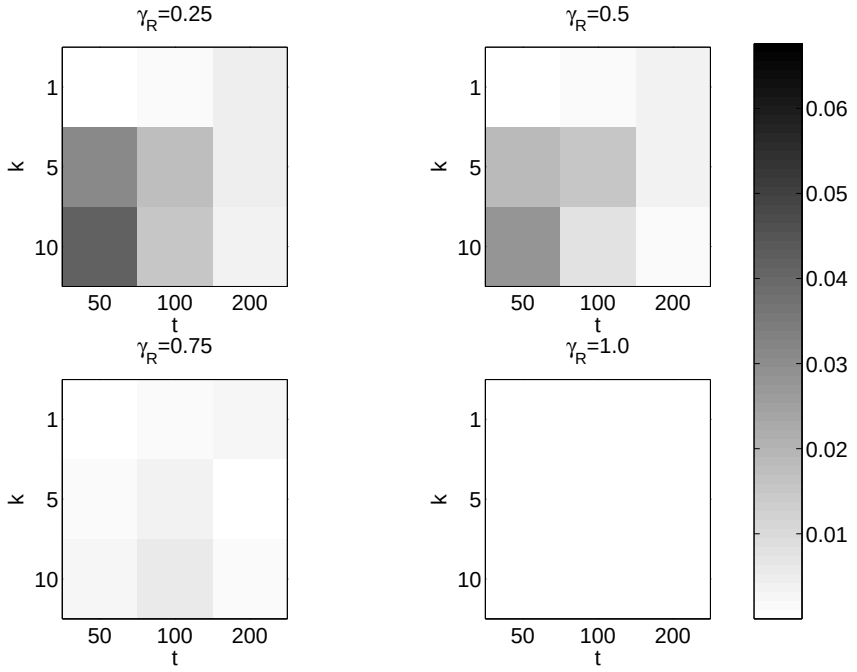


Figure 6.7: The loss in quality of the best found solution of R-strategy with different values of γ_R, k and t in case study one

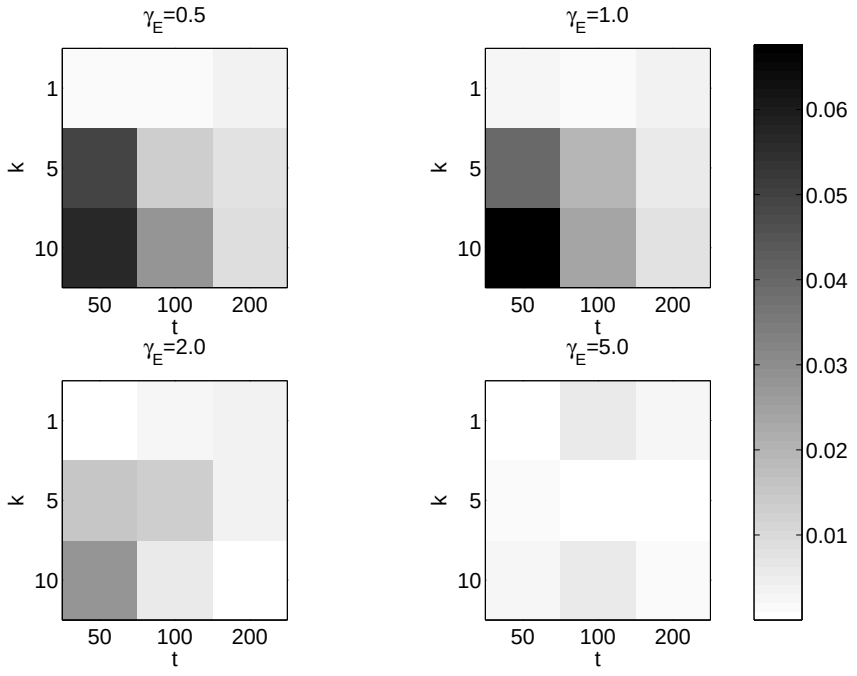


Figure 6.8: The loss in quality of the best found solution of η-strategy with different values of γ_E, k and t in case study one

τ-strategy with $\gamma_T = 5.0$, and the G-strategy. The η-strategy with $\gamma_E = 5.0$ provides good solutions when $k = 5$ and $t \in \{100, 200\}$. The loss in G-strategy is almost zero because when the number of abstract services is small, the ants can find the best

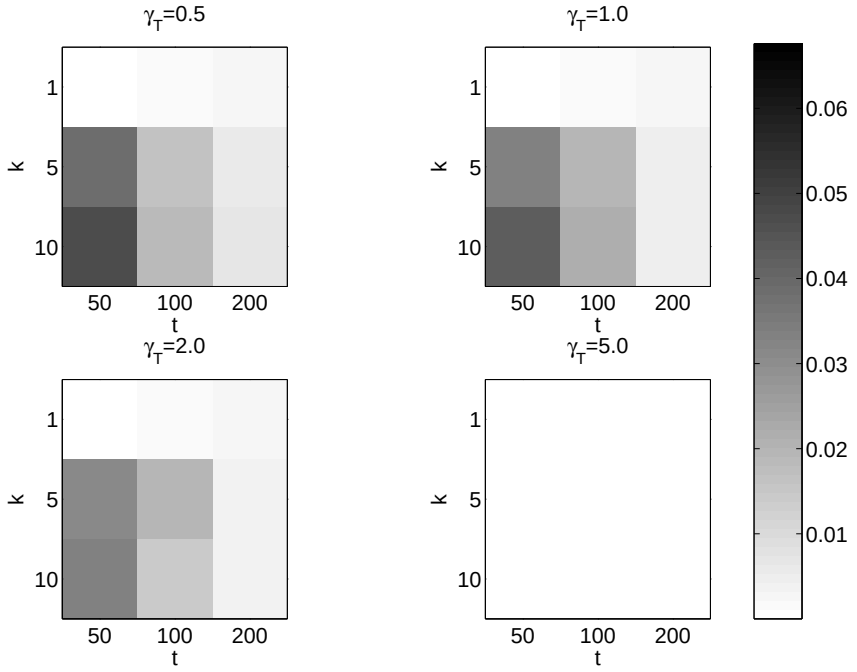


Figure 6.9: The loss in quality of the best found solution of τ-strategy with different values of γ_T, k and t in case study one

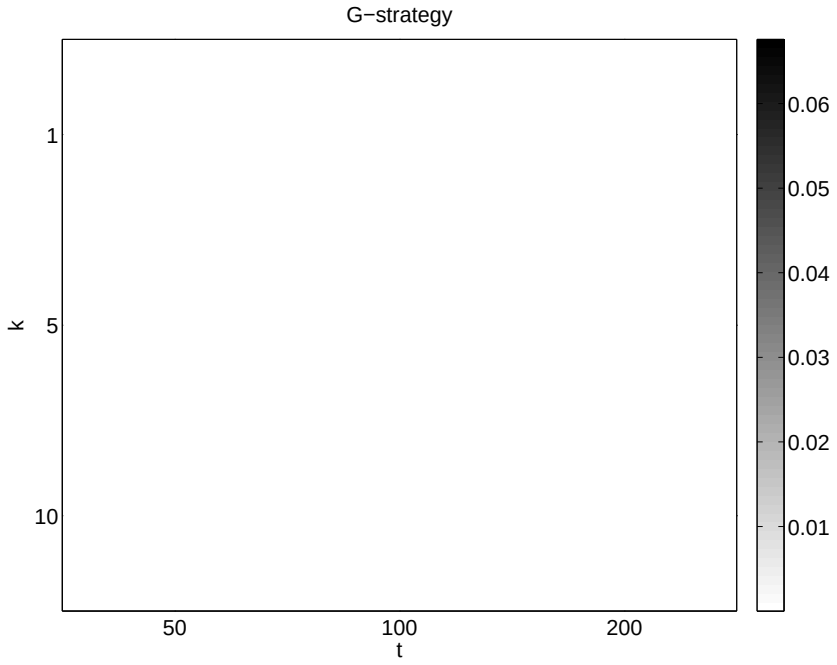


Figure 6.10: The loss in quality of the best found solution of G-strategy with different values of k and t in case study one

utility in the first iteration. Although no old pheromone information is preserved, the ants can find new solution quickly. This was illustrated by the experimental results of [178].

6.3.2 Results of Case Study Two

For case study two, the optimization behavior for each individual strategy and its strategy-specific parameter are shown in Figures 6.11-6.14. These figures are for the case of $(k, t) = (1, 50)$. From these figures, we observe that $\gamma_R = 1.0$ in R-strategy, $\gamma_E = 5.0$ in η-strategy, and $\gamma_T = 5.0$ in τ-strategy give better results than other strategy-specific parameters for the individual strategies in general. Compared with the optimization behavior for each individual strategy given in Figures 6.2-6.5, the average global utility at each interval gradually increase. That is because in case study one the ants can find the best utility in just a few iterations after a change occurred, but in case study two, the number of abstract services increases so the ants need more iterations to find the best solutions.

Figures 6.15-6.18 show the performance of each strategy and its different strategy-specific parameters for all the cases of the combination of k and t. From these figures,

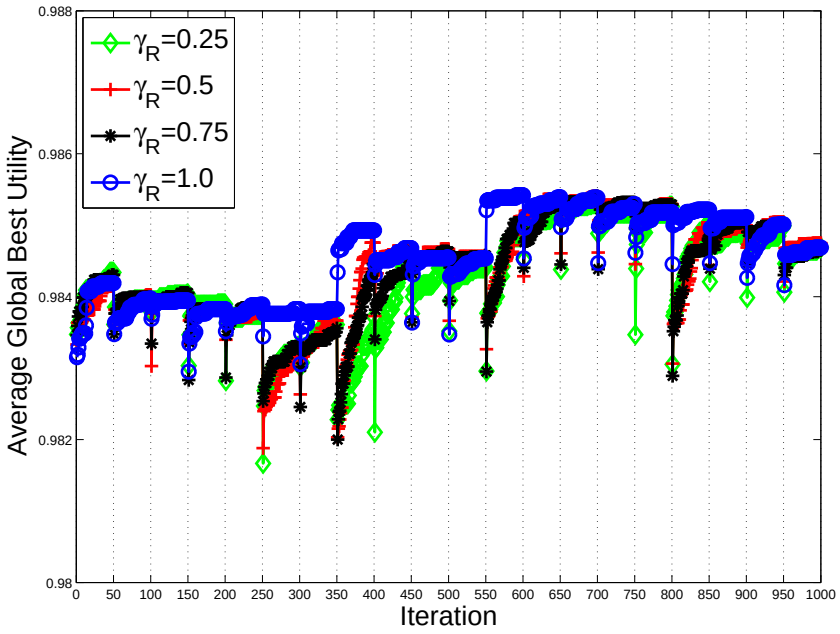


Figure 6.11: The optimization behavior of R-strategy with all parameters in case study two

we observe that R-strategy with $\gamma_R = 0.5$ and η-strategy with $\gamma_E = 2.0$ give better solutions when $(k, t) = (1, 200)$, τ-strategy with $\gamma_T = 2.0$ gives better solutions when $(k, t) = (5, 200)$. The G-strategy gives better solutions than $\gamma_E = 5.0$ in η-strategy

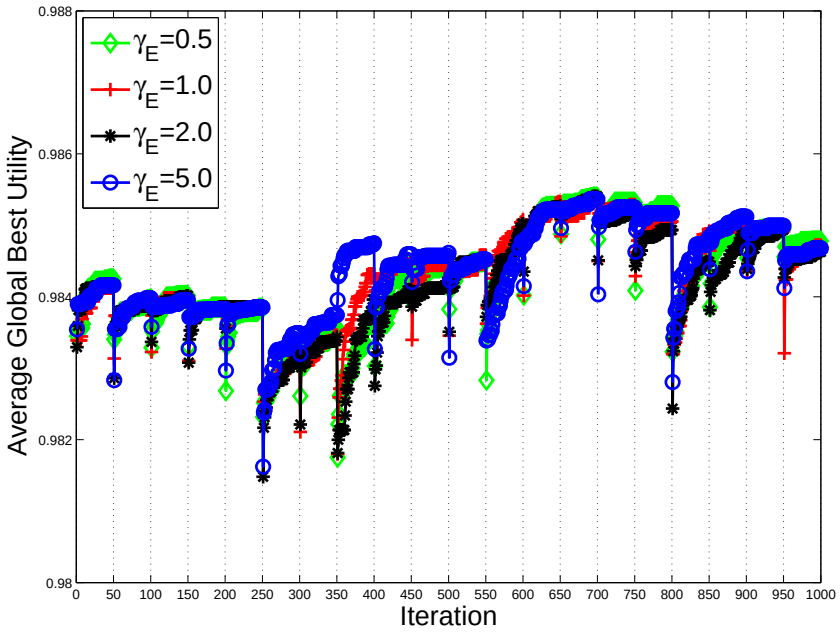


Figure 6.12: The optimization behavior of η-strategy with all parameters in case study two

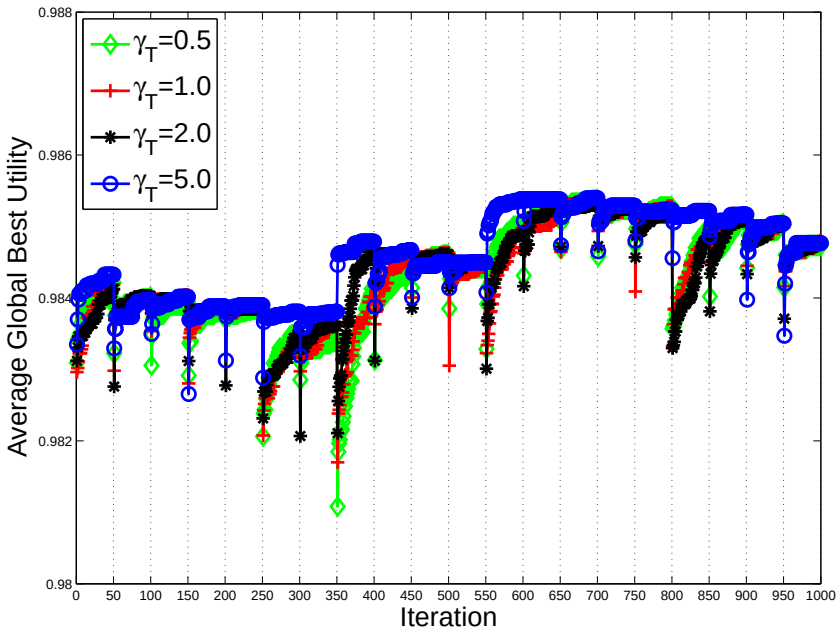


Figure 6.13: The optimization behavior of τ-strategy with all parameters in case study two

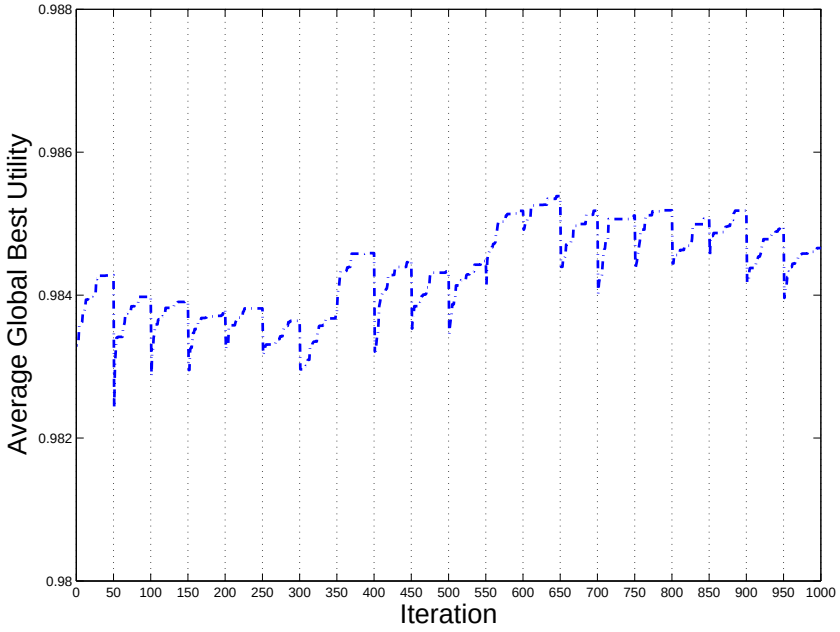


Figure 6.14: The optimization behavior of G-strategy in case study two

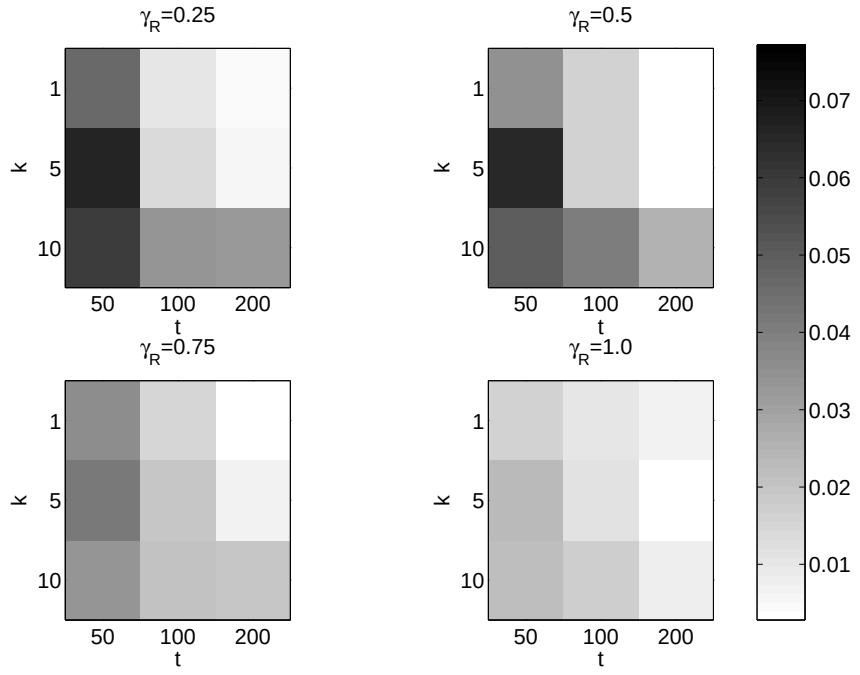


Figure 6.15: The loss in quality of the best found solution of R-strategy with different values of γ_R, k and t in case study two

when $k \in \{5, 10\}$ and $t = 50$. Judging from the "average darkness", R-strategy with $\gamma_R = 1.0$ gives better solutions when $k \in \{5, 10\}$, and τ-strategy with $\gamma_T = 5.0$ gives better solutions when $k = 1$.

The results of the two case studies indicate that $\gamma_R = 1.0$ in R-strategy, $\gamma_E = 5.0$

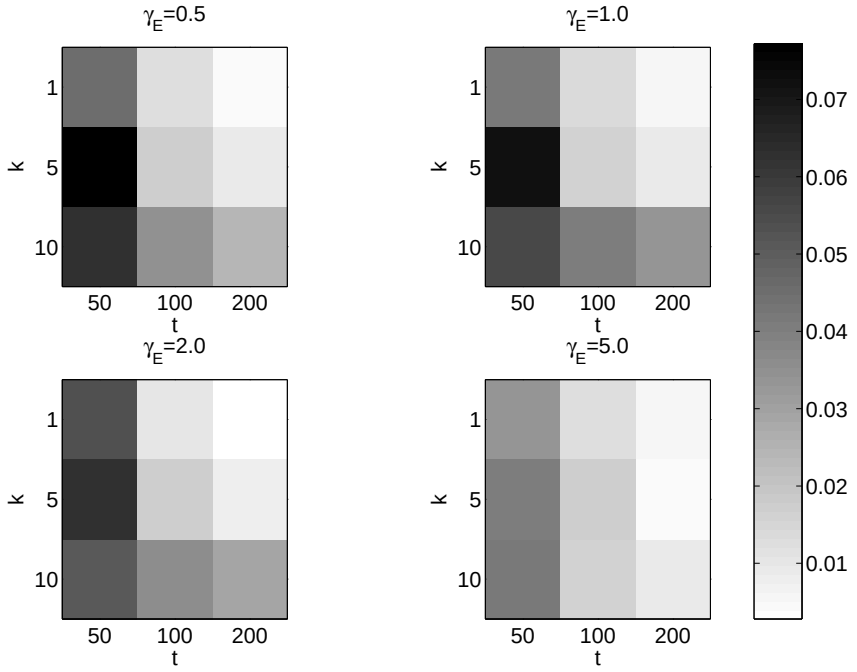


Figure 6.16: The loss in quality of the best found solution of η-strategy with different values of γ_E, k and t in case study two

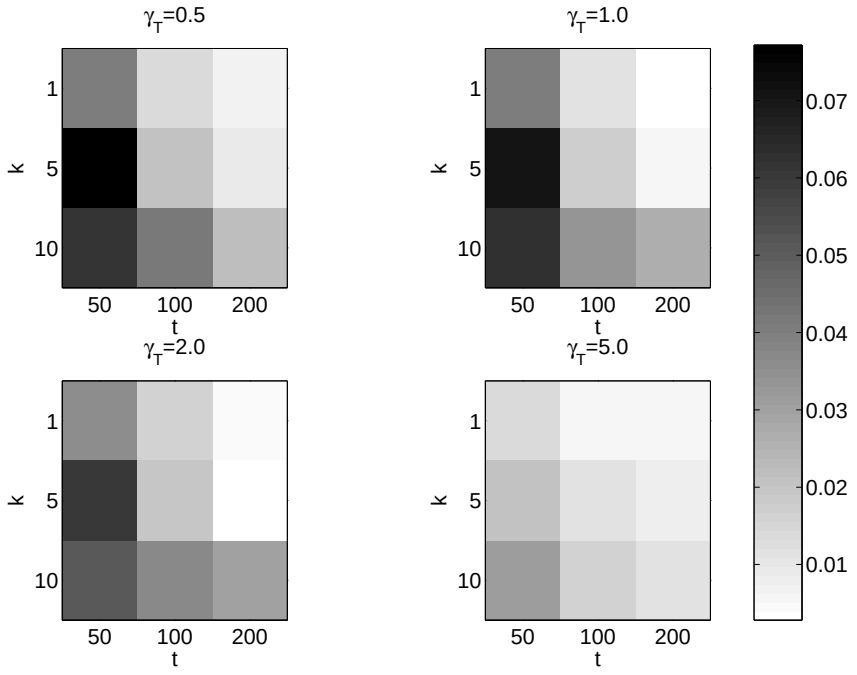


Figure 6.17: The loss in quality of the best found solution of τ-strategy with different values of γ_T, k and t in case study two

in η-strategy, and $\gamma_T = 5.0$ in τ-strategy give better results than other strategy-specific parameters for the individual strategies. The G-strategy could give better results when the number of abstract services is very small, in my case is 10. When

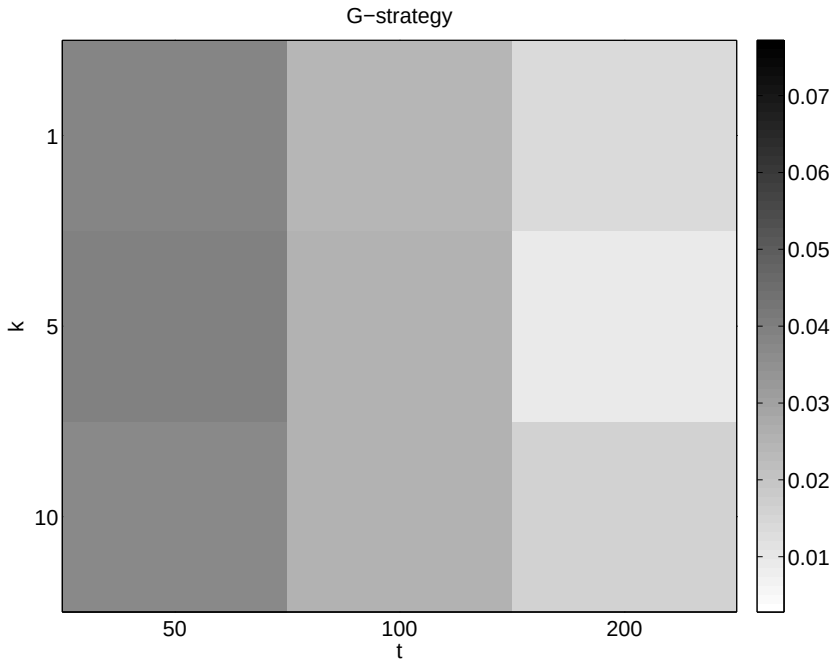


Figure 6.18: The loss in quality of the best found solution of G-strategy with different values of k and t in case study two

the number of abstract services becomes bigger, the R-strategy could give better results in the case of many changes, and the τ-strategy could give better results in the case of a small number of changes.

6.4 Related Work

Many research studies have been carried out on dynamic service composition problems. El Hadad et al. [60] proposed a design-time selection algorithm for automatic Web service composition, where transactional properties and QoS characteristics were both integrated in the selection process. Fujii and Suda [73] designed a semantics-based composition architecture, which consisted of a semantics-aware component model, a middleware, and a semantics-based service composition mechanism. The proposed architecture could dynamically compose the requested services based on the semantics. Lécué et al. [106] introduced a framework for performing dynamic service composition by exploiting the semantic matchmaking between service parameters to enable their interconnection and interaction. Brahmi and Gammoudi [27] presented a service composition approach based on a multi-agent system. The agents cooperated together to find the feasible solutions.

Wu and Zhu [190] formulated the QoS-aware service composition problem with transactional constraints, and users could specify the transactional constraints and QoS attributes in their requests. Then the problem was modeled as a constrained directed acyclic graph, and the ant colony optimization algorithm was used to search for a near-optimal solution. Similarly, the studies [112, 196, 206] applied the ant colony optimization algorithms to solve dynamic service selection and composition problems. However, these studies focused on the dynamic binding of concrete services at running time, and they did not present strategies to adapt the ant colony optimization algorithms to handle the dynamic scenarios. To the best of my knowledge, this work is the first to study the pheromone modification strategies and to apply an ant colony system to solve the dynamic data-intensive service provision problem.

6.5 Summary

In the dynamic data-intensive service provision problem, the states of services change at certain intervals. In order to adapt the ant colony system to handle the dynamic changes, this chapter discussed four strategies to modify the pheromone information. Two case studies were used to evaluate the proposed strategies. For each case study, the performance of the four strategies with respect to the optimization behavior and the loss in quality of the best found solution were compared.

Dynamic service price-setting models for composite services are increasingly negotiation-based. The algorithm for negotiation processes among the service composers, the service providers, and the data providers, when they source and buy services and data, is investigated in next chapter.

Chapter 7

Ant-Inspired Negotiations in the Data-Intensive Service Provision

Negotiation has been adopted to establish service contracts within the agent community. Intelligent agent technology is used to conduct automatic negotiations and to establish multi-agent systems. In the context of service provision, if the QoS attributes of a composite service cannot satisfy the global QoS constraints, it is useful to exploit the competition among service providers to help the service composer obtain favorable services and find feasible solutions. To reach an agreement in data-intensive service provision, the service composers, the service providers, and the data providers are represented by a set of agents who negotiate the attributes of services and data sets.

This chapter firstly describes the lifetime of the data-intensive service provision. Two-stage negotiation processes are specified in the lifetime, which will provide effective and efficient service selection for service composers. It then presents a multi-phase, multi-party negotiation protocol and discusses the decision making model. The experimental results show that the proposed negotiation-based approach, compared with the traditional non-dynamic method, can facilitate the data-intensive service provision with a better outcome.

7.1 Introduction

As described in Chapter 4, the decisions of the service composers, the service providers, and the data providers depend on each other. The data provider sells data sets to multiple service providers in order to maximize the data usage and the profit. The cost and response time of data sets for one service provider are affected by the demand of the others. The service providers also play the requester role with respect to the data sets. Thus, service providers will have two aims, one is to lower the access cost and response time of data sets and the other is to maximize their profit and service usage. Also, the service providers compete with other service providers to initiate or maintain a contract with the service composers and are invariably interested in cost saving. The actual usage of services typically encourages the composers to have a long term contract with the service providers. They also select concrete services that best match the QoS requirements. Meanwhile, data-intensive services are typically used in a dynamic and changing environment, and different providers typically have conflicting objectives. In order to automate the process of reaching an agreement in the problem of this research, a group of agents was exploited to establish agreeable service contracts.

A negotiation process is the interplay of offers and counter-offers between a buyer and a seller, with different criteria and goals, working to identify a mutually acceptable solution. Automated negotiations normally follow negotiation protocols, exchange negotiation objects, and are driven by decision making models [96]. Negotiation protocols govern the negotiation by defining rules such as when the negotiation process ends, what deals can be made, and what sequences of offers are allowed. Negotiation objects are the issues such as price and time over which the negotiation takes place. Decision making models are used for evaluating and generating offers and counter-offers.

During the negotiation processes, the service composition optimization process should be conducted repeatedly when the changes of the states of services occur, such as some service providers improve the QoS attributes, some service providers exit the negotiation, and some new service providers enter the negotiation. The dynamic ACS that presented in Chapter 6 is applied to select services in the negotiation approach.

7.2 The Lifetime of Data-Intensive Service Provision

The lifetime of the problem framework is described in Figure 7.1. The first step is that the service composer tries to select a set of service candidates while the data provider provides data sets, and the second step is that if a feasible solution which satisfies the service composer's local and global QoS constraints does not exist, negotiations are performed in order to determine new quality values for each involved service. In the lifetime, two-stage negotiation processes are used. In the first stage, a service composer negotiates with multiple service providers over each service in a structured one-to-many negotiation process. In the second stage, each service provider negotiates with a data provider over a set of data sets in a structured one-to-one negotiation process. For the sake of simplicity, the author restricts her attention to the following two QoS attributes but other attributes could be easily

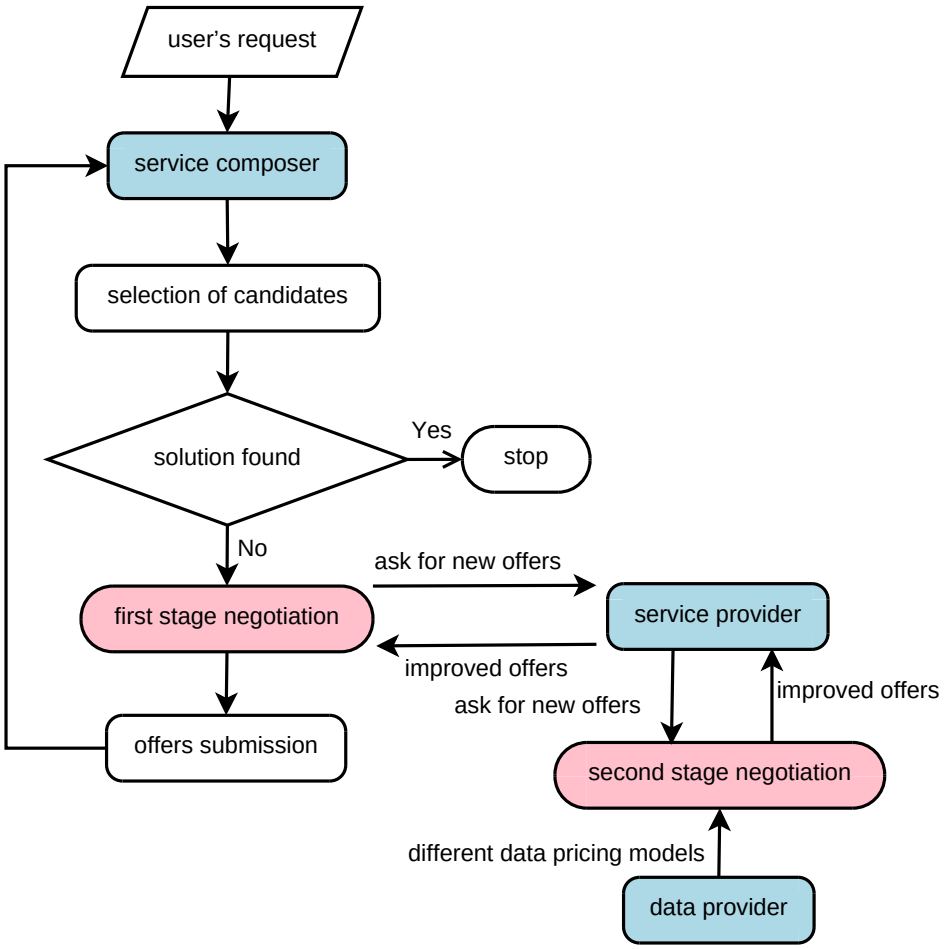


Figure 7.1: The lifetime of the data-intensive service provision

added to the negotiation approach:

- execution time: the interval of time elapsed from the invocation to the completion of a service;
- cost: the price charged for each invocation of a service.

7.2.1 First Step: Data-Intensive Service Composition

In the first step, the data provider provides data sets in the usage-based pricing model and the package-based pricing model. These two pricing models are called short-term options. In the usage-based pricing model, data users need to pay for data sets in respective usage. In the package-based pricing model, the data provider offers a certain amount of data sets as a package to data users, and the price of the package is lower than the sum of the price of individual data set in the package. Here the author makes the implicit assumption that the data provider always sets a reasonable price and package for data sets to accommodate all users' requirements, otherwise the data provider would have an optimization problem in the first step. In the following, the optimization problem of the service provider and the service composer will be presented.

Service Provider

Each service provider independently determines the optimal number of data sets paid in short-term options. If data sets have data replicas which distributed in different data servers, each service provider also decides where to access data sets. The cost and execution time of service $\tilde{s}$, $Cost(\tilde{s})$ and $T_{et}(\tilde{s})$, are given in the extensible QoS model, which is described in Chapter 4. The variable T is used to denote the overall period over which there exist SLA contracts for the composite service. The optimization problem of each service provider is to maximize the total profit, which

is given by (7.1).

$$\max \Theta_{sp}^{T} = \varphi_{\tilde{s}} * (p(\tilde{s}) - Cost(\tilde{s})) \tag{7.1}$$

subject to:

$$\varphi_{\tilde{s}} \geq 0$$

$$0 \leq f(q^{\tilde{s}}) \leq p(\tilde{s}) \leq p^{max}(\tilde{s})$$

The variable $\varphi_{\tilde{s}}$ is the number of invocations of service $\tilde{s}$, $p(\tilde{s})$ is the price of $\tilde{s}$, $q^{\tilde{s}} = [Cost(\tilde{s}), T_{et}(\tilde{s})]$ denotes the QoS attributes of service $\tilde{s}$, and $p^{min}(\tilde{s})$ ($p^{min}(\tilde{s}) = f(q^{\tilde{s}})$) and $p^{max}(\tilde{s})$ denote the minimum and maximum price of service $\tilde{s}$. It is obvious that both the invocations and the price of a service must be positive numbers. Furthermore, it is necessary to impose an upper and lower bound on the price. The invocations and the price of a service are connected variables: intuitively, as the price for a service increases, the demand decreases, and vice versa. In order to have a contract with the service composer, the service provider needs to lower the price by decreasing the total cost of the service. According to El Azouzi et al. [59], the dependence of $p^{min}(\tilde{s})$ on $q^{\tilde{s}}$ through the function f will eliminate explicitly policies that will result in negative profits for a service provider. The maximum price reflects the fact that beyond some reasonable price, the demand will be zero whatever the price and QoS attributes are.

If a data set has more than one data replica, each service provider needs a data replica selection strategy to select data replicas. The selection can be done off-line before the invocation of services. The selection of data replicas can be performed by a set of data replica agents. The data replica agents are located on each service platform. They use an optimization approach to select the optimal replica of a data set, and use a caching function to make informed decisions about local data caching. Ranganathan and Foster [144] distinguished caching and replication: replication was assumed to be a server side phenomenon that the server decided when and where to create a replica of its data, while caching was defined as a user side phenomenon that the user selected the best replica and cached a copy of the replica at the local machine. A data replica agent is invoked by a service that needs to access a data set. If the service platform does not hold the data set, the data replica agent consults

the replica location service [147] for replica locations. Then the data replica agent uses the optimization approach to select the optimal replica of that data set, and it might invoke the caching function. The purpose of the caching function is to create, on the corresponding service platform, a copy of the requested data set. A caching function is only invoked if the agent decides that having a local copy of the data set is economically beneficial, in the situation that the service platform has space to store the data set.

A data replica selection optimization algorithm based on an ant colony system was presented in [175]. The optimization approach is similar with the one described in Chapter 4, however, the data replica selection process regards the response time and the cost of a data replica as the criteria.

Service composer

The service composer needs to select an overall best set of candidate services with sufficient quality at a reasonable price, according to a utility function and a set of QoS constraints. The utility function expresses preferences that prioritize values of the QoS attributes, typically it is the weighted sum of the normalized QoS attributes. The constraints define requirements regarding the aggregated QoS values of the requested composite service. The service composer can define different optimization goals according to requesters' requirements. Then the problem of a service composer is turned into a constraint optimization problem, which is given by (7.2).

$$\max \Theta_{sc}^{T} = p(CS) - F_c(CS) \tag{7.2}$$

subject to:

$$Cost(CS) \leq C_{CS}^{max}$$

$$RT(CS) \leq RT_{CS}^{max}$$

$$0 \leq f(q^{CS}) \leq p(CS) \leq p^{max}(CS)$$

The variable $p(CS)$ is the price of composite service CS. $q^{CS} = [Cost(CS), RT(CS)]$ denotes the QoS aggregate attributes of CS. C_{CS}^{max} and RT_{CS}^{max} are the upper bound of the cost and the upper bound of the response time of CS respectively. It is also necessary to impose an upper and lower bound on the price of CS, denoted by

$p^{max}(CS)$ and $p^{min}(CS)$ ($p^{min}(CS) = f(q^{CS})$).

Chapter 4 presents an ant colony system and a genetic algorithm to solve this problem in the first step.

7.2.2 Second Step: Two Negotiation Processes

In the second step, the service composer negotiates with multiple service providers about the price and execution time of services, and the service providers negotiate with a data provider about the price and response time of data sets.

The negotiators, namely, the service composer, the service provider, and the data provider will be represented by agents. The agents acting on behalf of service composers are referred as composer agents (CAs), agents representing service providers as service agents (SAs), and agents representing data providers as data agents (DAs). Basically, an agent starts by offering a value for the negotiation object (say, a price) to its opponent. The opponent can accept the offer, exit the negotiation, or make a counter-offer. The negotiation may be iterative as several rounds of offers and counter-offers will occur until one of the agents accepts, or exits the negotiation. The agent might exit the negotiation when the time deadline is reached without an agreement being in place. The iteration is referred to as a negotiation round, and the time deadline is referred to as the number of allowed rounds.

7.3 The Extended Negotiation Protocol

A multi-phase, multi-party negotiation protocol for the problem has been designed. The proposal is based on the iterated contract net interaction protocol, which is specified by the Foundation of Intelligent Physical Agents (FIPA) [71]. The FIPA protocol is the most widely used negotiation protocol for one-to-many negotiation in the agent community [5, 194]. It is also applicable in one-to-one negotiation, as it is considered to be a particular case of one-to-many negotiation. The protocol supports recursive negotiation and allows multi-round iterative negotiation to reach an agreement. However, the FIPA standard protocol is insufficient in supporting QoS negotiation for the problem in this research, as it is unable to allow multi-party negotiation or combine multiple negotiation processes. It is hence extended to allow

a CA to aggregate results of the individual services and to perform the confirmation to instruct SAs in the negotiations. Also, the extended protocol should allow the negotiations between SAs and DAs.

There are three phases in the extended protocol:

1. Phase 1, which allows us to find out how many SAs are available to enter the negotiation, and their offers over the objects to be negotiated. This phase includes one-shot interaction between the CA and all SAs, and it also includes one-shot interaction between all SAs and the DA. The CA is involved in simultaneous negotiations with multiple SAs. Each negotiation between the CA and a SA is private and holds a lot of information. Each SA is unaware of its competitor's status in the current composition. If the CA sends the current winning offers to potential SAs, SAs can analyze their positions and adapt quickly [81, 83]. This in turn significantly reduces the search space by guiding the negotiation process to proceed in the right direction towards optimal solutions. The SAs process the request from the CA and may decide to negotiate with the DA. Before a SA answers the CA, it evaluates the offers received from the DA. The DA may accept the counter-offer from a SA, or the DA may refuse to participate in the negotiation, or the DA may provide a new offer to the SA. Thus, the SA can answer the CA based on the results of the nested negotiation offers. The type of response generated by a SA in this phase is based on the following conditions.

   (a) proposal, the SA will respond positively and the process will terminate.

   (b) refusal, the SA refuses to participate in the negotiation.

   (c) counter-proposal, the SA will respond to the CA with its new offer.

   The negotiation protocol is shown in Figure 7.2.

2. Phase 2, which allows us to iterate the negotiation. In case the first negotiation phase ends up with counter-proposals from SAs, the CA will initiate negotiation with the SAs and send them new proposals with the set of current winning offers and counter-offers as its content. The following response options are available to SAs.

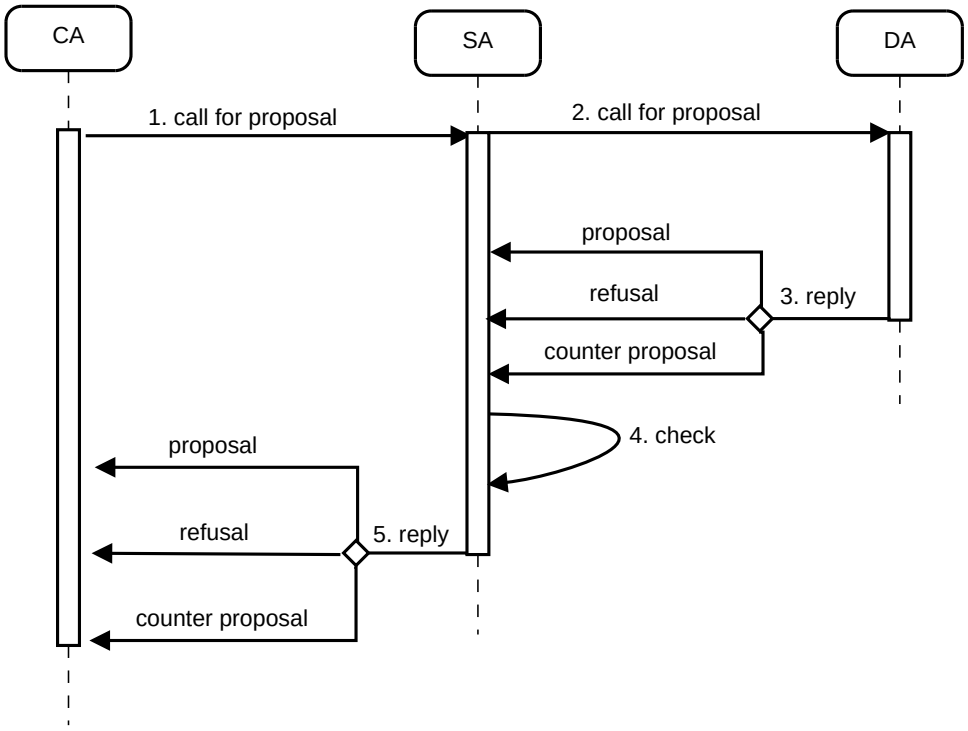


Figure 7.2: Protocol to support first phase of negotiation

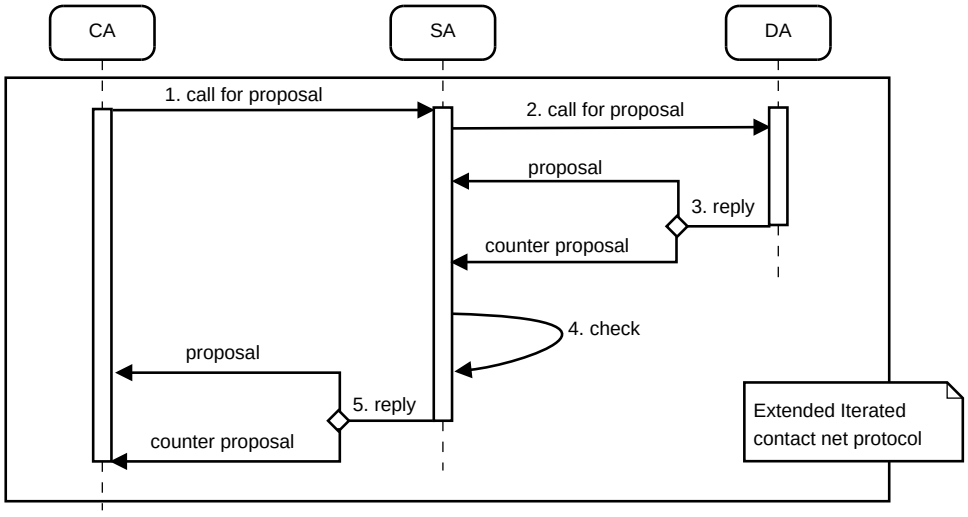


Figure 7.3: Protocol to support second phase of negotiation

(a) proposal, the SA will respond positively and the process will terminate.

(b) counter-proposal, the SA responds with a counter-offer.

This negotiation phase is repeated until there is no counter-proposal from SAs or the time deadline is reached. The negotiation protocol is shown in Figure 7.3.

3. Phase 3, which allows us to end the negotiation. If the second negotiation

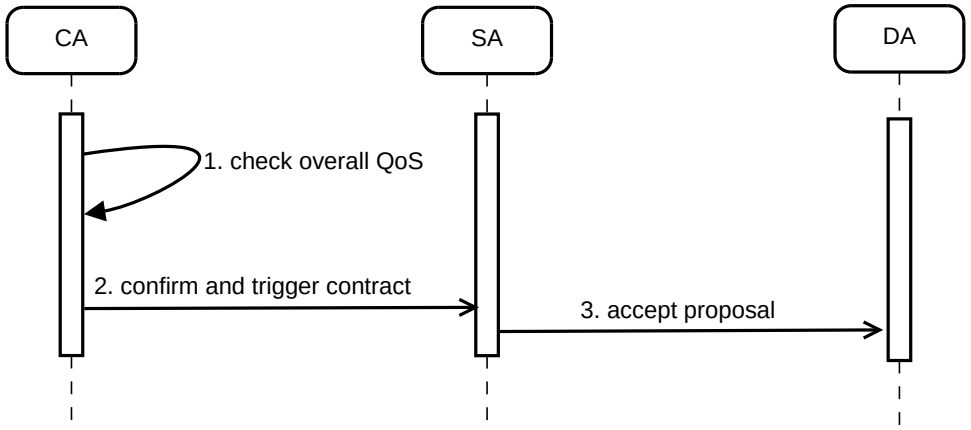


Figure 7.4: Protocol to support third phase of negotiation in case of successful negotiation

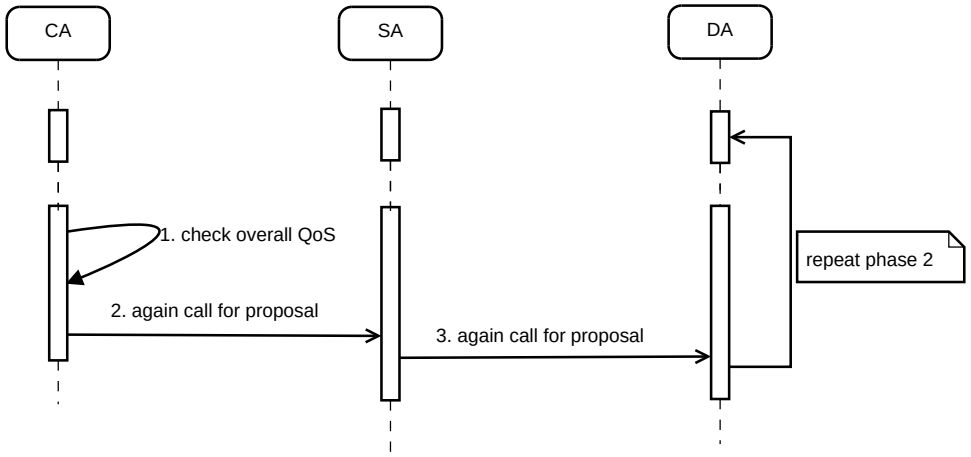


Figure 7.5: Protocol to support third phase of negotiation in case of failed negotiation

phase ends up with proposals, the CA will evaluate the overall QoS attributes. If the overall QoS requirements are satisfied, the CA informs all SAs about the acceptance of the offers. Then SAs inform the DA about the acceptance of the dependent offers. The negotiation leads to a signed contractual agreement since it ends with an optimal solution that satisfies the requirements. The negotiation protocol is shown in Figure 7.4. In the case that the overall QoS requirements are not satisfied based on the current deals, the CA will restart the second negotiation phase with respect to one or many abstract services. Then each corresponding SA sends proposals to the DA and commences a new negotiation process. The negotiation protocol is shown in Figure 7.5.

7.4 Decision Making Model

The negotiation process between two agents is a bilateral interaction that consists of a succession of offers and counter-offers. Let a ($a \in \{CA, SA, DA\}$) represent the negotiation agent and o ($o \in \{1, 2, \ldots, r\}$) the negotiation object (or issue). Each agent has a defined delimited range to consider the value of an issue. For example, the CA has the maximum price at which it would buy the service and the SA has the minimum price at which it would sell the service. Let $x_o \in [min_o^a, max_o^a]$ be a value for issue o acceptable to agent a. The agent acts alternately, making or accepting offers and counter-offers on the value x_o, or abandoning the negotiation when the time deadline is reached without an agreement being in place. Each agent a has a utility function $U_a(x_o) : [min_o^a, max_o^a] \rightarrow [0, 1]$, representing its satisfaction of the value x_o. For convenience, the utility values are kept in the interval $[0, 1]$. Linear utility function was adopted as it was used in many cost-benefit models in the literature [46, 67, 210]. Each negotiator can also define the utility functions more explicitly [187].

To negotiate on multiple object (say r), a normalized weight w_o^a ($\sum_{o=1}^{r} w_o^a = 1$) representing the relative importance of each object o for agent a is assigned under negotiation. Then an agent's utility function for an offer $x = (x_1, x_2, \ldots, x_r)$ in the multi-dimensional space is given by (7.3).

$$U_a(x) = \sum_{o=1}^{r} w_o^a * U_a(x_o) \tag{7.3}$$

The variable $x_{b \rightarrow a}^t$ is used to denote the offer x that agent a received from agent b at time t, and a's interpretation of $x_{b \rightarrow a}^t$ at time t' ($t < t'$) is given by (7.4).

$$I^a(t', x_{b \rightarrow a}^t) = \begin{cases} reject & \text{if } t' > t_{max}^a, \\ accept & \text{if } U_a(x_{a \rightarrow b}^{t'}) \leq U_a(x_{b \rightarrow a}^t), \\ x_{a \rightarrow b}^{t'} & \text{otherwise.} \end{cases} \tag{7.4}$$

The variable $x_{a \rightarrow b}^{t'}$ is used to denote the offer that agent a is ready to send to agent b at time t'. A constant t_{max}^a is used to represent the time by which agent a must have completed the negotiation.

In order to prepare a counter-offer, $x^{t'}_{a\to b}$, agent a uses a set of tactics that generate new values for each issue in the negotiation set. The time dependent tactics, which use time-based decision functions $\alpha^a_o(t)$ $(0 \leq \alpha^a_o(t) \leq 1)$, are often used in negotiation systems [42, 46, 67, 109, 210]. In this case, the counter-offer generated from agent a to b at time t' with respect to each object o, $x^{t'}_{a\to b}[o]$, is given by (7.5).

$$x^{t'}_{a\to b}[o] = \begin{cases} min^a_o + \alpha^a_o(t)(max^a_o - min^a_o) & \text{if } U_a(x(o)) \text{ is decreasing,} \\ max^a_o - \alpha^a_o(t)(max^a_o - min^a_o) & \text{if } U_a(x(o)) \text{ is increasing.} \end{cases} \tag{7.5}$$

The counter-offer $x^{t'}_{a\to b}[o]$ will always be between the value range $[min^a_o, max^a_o]$. When the time deadline is reached, the tactic will suggest offering the reservation value. The reservation value for issue o of agent a represents the value that provides the smallest utility. If the utility function $U_a(x(o))$ is monotonically increasing, the reservation value is min^a_o, otherwise if $U_a(x(o))$ is decreasing, the reservation value is max^a_o.

The time-based decision function $\alpha^a_o(t)$ can be defined as the exponential function, polynomial function, and sigmoid function [67, 109, 210]. For example, the polynomial function can be given by (7.6).

$$\alpha^a_o(t) = \kappa^a_o + (1 - \kappa^a_o) * \Big(\frac{\min(t, t^a_{max})}{t^a_{max}}\Big)^{\frac{1}{\delta}} \tag{7.6}$$

The variable δ is a parameter used to control the concession rate, and κ^a_o is the initial concession at $t = 0$, where $\alpha^a_o(0) = \kappa^a_o$ $(0 \leq \kappa^a_o \leq 1)$ and $\alpha^a_o(t^a_{max}) = 1$.

For each type of agent, the above time dependent tactic, which uses the polynomial function, is used to generate counter-proposals. For each bilateral negotiation between the CA and a SA or between a SA and the DA, an agreement is possible if there is some degree of intersection between the reservation intervals of the two agents. After CA and SA finish their negotiation process, it is necessary to modify the pheromone information of ACS. The CA can choose certain pheromone modification strategy since the CA knows the states of all services in the composition. The data-intensive service composition based on ACS is given in Function ACS(G), and the negotiation-based approach is illustrated in Algorithm 7.1. All the variables

Function ACS(G)
Initialization:
$MaxIt$: the maximum number of iterations;
noa: the number of artificial ants;
Input:
G: the graph;
Output:
S: the best path;
$optimal$: a variable used to indicate whether S is a feasible solution;

$S = \emptyset$;
$optimal = 0$;
$step = 0$;
while $step < MaxIt$ **do**
  $step = step + 1$;
  set all ants at $start$ vertex;
  for each ant k in noa **do**
    $list_k = \emptyset$; //candidate list for each ant
    while ant k is not at the end vertex **do**
      if $q \leq q_0$ **then**
        ant k chooses a successor j which maximizes $[\tau_{ij}][\eta_{ij}]^{\beta}$;
      else
        ant k chooses a successor j according to the probability distribution $p_{ij}^k = \frac{[\tau_{ij}][\eta_{ij}]^{\beta}}{\sum_{j \in N_i^k} [\tau_{ij}][\eta_{ij}]^{\beta}}$;
      end if
      ant k updates its candidate list $list_k$;
      ant k applies the local updating rule to the edge (i, j), $\tau_{ij} = (1-\xi)\tau_{ij} + \xi\tau_0$;
    end while
  end for
  when all ants arrive at the end vertex, find the best path L from all candidate lists;
  compare the utility of L with the utility of S. If the utility of L is larger than the utility of S, then S is replaced by L.
  apply the global updating rule to all edges belong to S, $\tau_{ij} = (1 - \rho)\tau_{ij} + \rho * Utility(S)$;
end while
if S is a feasible solution **then**
  $optimal = 1$;
end if
return S and $optimal$.

and parameters in Function ACS(G), and the computing of the utility of the best path are the same as described in Section 4.3.

Algorithm 7.1 Ant-inspired negotiation approach for the data-intensive service provision

Initialization:
$Tmax$: the maximum number of negotiation rounds;
Input:
G: a graph;
Output:
S and *optimal*;

1: $S = \emptyset$;
2: $iteration = 0$;
3: $[optimal, S] = ACS(G)$;
4: **while** $optimal == 0$ **do**
5: $iteration = iteration + 1$;
6: **if** $iteration > Tmax$ **then**
7: break;
8: **end if**
9: **for** each abstract service **do**
10: find out SAs that are available to enter the negotiation;
11: **for** each concrete service **do**
12: CA sends a counter-offer to SA;
13: SA sends a counter-offer to DA;
14: DA interprets the counter-offer from SA;
15: SA interprets the counter-offer from CA based on the nested offers from DA;
16: steps 12-15 are repeated until no counter-offer is proposed within the time deadline.
17: **if** a change occurs **then**
18: modify the related parameters of G;
19: **end if**
20: **end for**
21: **end for**
22: modify the pheromone matrix of ACS;
23: $[optimal, S] = ACS(G)$;
24: **end while**
25: **return** S and *optimal*.

7.5 Experiments and Analysis

In this experiments, a trial testing method is adopted to determine most suitable values for all parameters of ACS, considering other researchers' earlier experiments. Finally, the parameters for ACS were $\alpha = 1$, $\beta = 2$, $q_0 = 0.9$, $\rho = 0.1$, and $\xi = 0.1$. The number of ants in the colony was 10. To evaluate the negotiation-based approach, a MIP approach was also implemented to be applied to the same problem

and compare with the negotiation-based approach. The open source integer programming system *lpsolve* version 5.5 [22] was used for the MIP. All the experiments are conducted on computers with Inter Core i5 2500 CPU (3.3GHz and 8 GB RAM).

The aim of this evaluation is to analyze the performance of the proposed ant-inspired negotiation approach: 1) observing the effectiveness of the approach, measured by the success rate of finding an optimal solution; 2) investigating the efficiency of the approach, measured by the number of negotiation rounds and the computation time in each negotiation round; and 3) studying the effect of the problem size on the performance of the approach.

7.5.1 Test Case Generation

The performance of the approach is affiliated to the size of the data-intensive service provision problem. The size of the problem depends on the number of abstract services used in the composite service, the number of concrete services for each abstract service, and the number of data sets required by each abstract service. For the purpose of this evaluation, different scenarios were considered, where a composite application comprises services from n abstract services, and n varies in the experiments between 10 and 100, in increments of 10. There are m concrete services for each abstract service, and m varies in the experiments between 10 and 100, in increments of 10. Each abstract service requires a set of d data sets, and d varies in the experiments between 1 and 10, in increments of 1.

The optimization goal of the experiments is to maximize the overall utility, which is specified in Section 4.2. The severe quality constraints were generated according to the method described in [82]. A scenario generation system is used to generate all scenarios, as described in Section 4.5. Each time the number of abstract services, the number of service providers, or the number of data sets was changed, 50 instances of experiments were run and the average results were reported.

7.5.2 Results Analysis

The experimental results of all scenarios are grouped into three test sets. In the first set, n varies from 10 to 100, while $m = 10$ and $d = 5$. In the second set, m varies from 10 and 100, while $n = 10$ and $d = 5$. In the third set, d varies from 1 to 10,

while $n = 10$ and $m = 10$.

To evaluate the effectiveness of the negotiation approach, its success rate was compared with that of the MIP approach. The success rate is the percentage of instances where a solution could be found. The success rate of the negotiation approach and the MIP approach with respect to the number of abstract services, the number of service providers, and the number of data sets are presented in Figure 7.6. The results illustrate that the success rate of the MIP approach remains zero in all scenarios, while the negotiation approach maintains a higher success rate compared with the MIP approach. Specifically, the average success rate of the negotiation approach is 98.40%, 98.00%, 93.20% in the first, second, and third set of experiment.

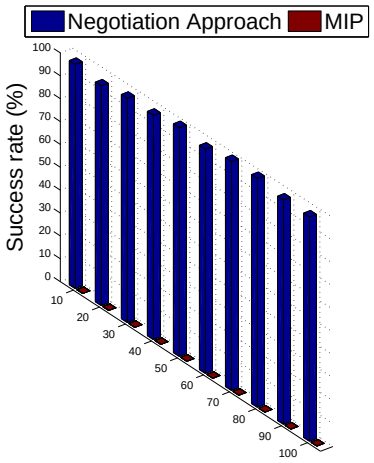
The number of negotiation rounds is a relevant efficiency factor of the negotiation approach. A large value in the negotiation round implies that the problem has to be solved via many iteration numbers in the whole negotiation process. Figure 7.7 shows the negotiation rounds of the three sets. The figure indicates that the number of negotiation rounds increases when the number of service providers increases. When the number of abstract services or the number of data sets increases, the number of negotiation rounds, on the other hand, does not exact increase . On average, it took approximately seven rounds to find an optimal solution in the first and third test sets.

Besides the number of negotiation rounds, the time consumption for each round is also a very important factor with respect to the efficiency. Figure 7.8 presents the average time consumption per negotiation round of the three sets. The results indicate that the time consumption per round increases when the number of abstract services, the number of service providers, or the number of data sets increases. As illustrated, the time consumption per round is linear rather than exponential.

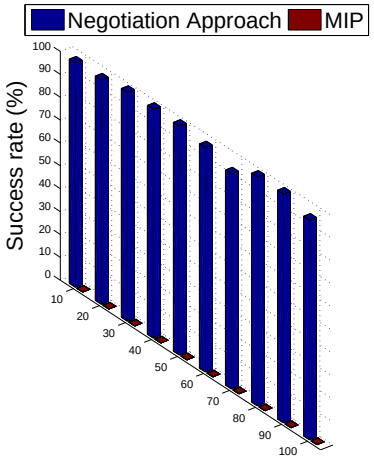
7.6 Related Work

7.6.1 Data Replica Selection

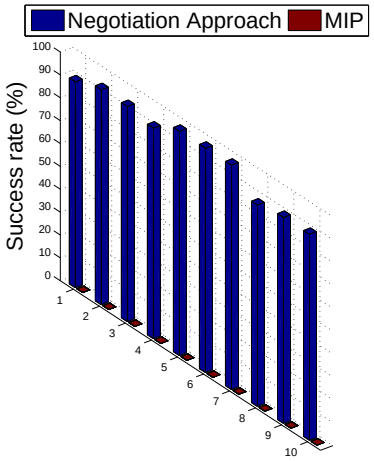
Data replication is considered to be an important technique used in cloud computing to speed up data access, to reduce bandwidth consumption, execution cost and users waiting time, and to increase data availability [86]. Data is replicated across large



(a) varying number of abstract services

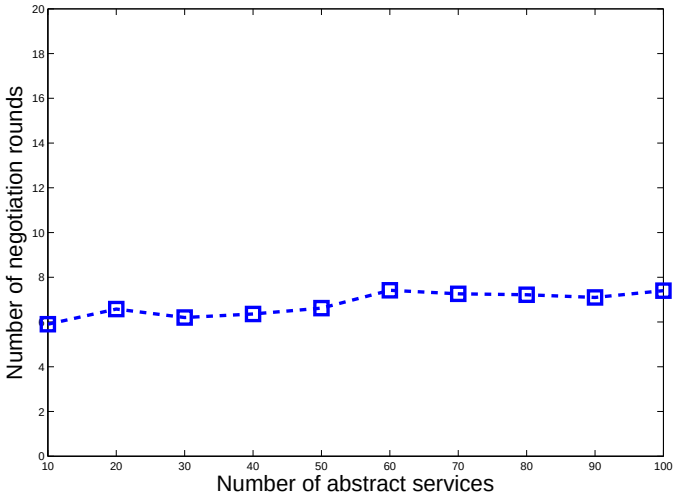


(b) varying number of service providers

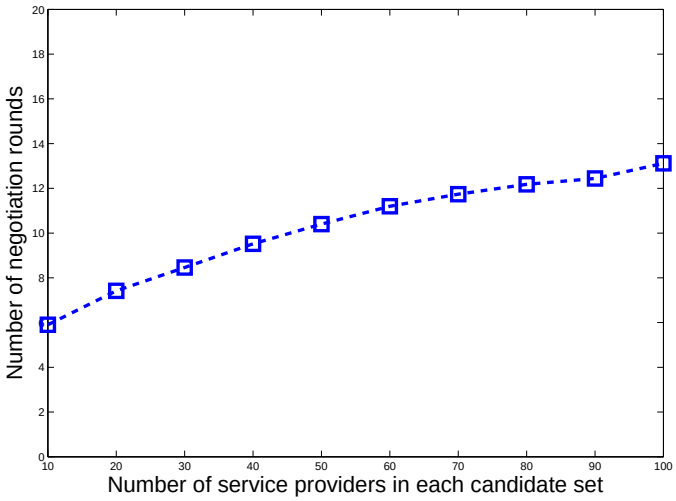


(c) varying number of data sets

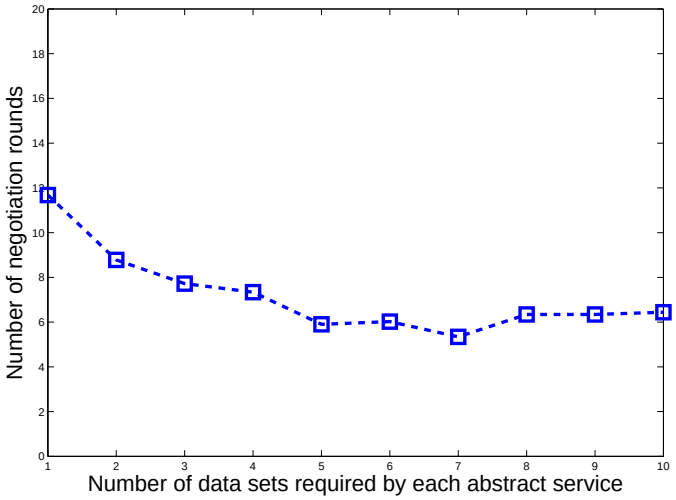
Figure 7.6: Success rate of the negotiation approach and the MIP approach



(a) varying number of abstract services

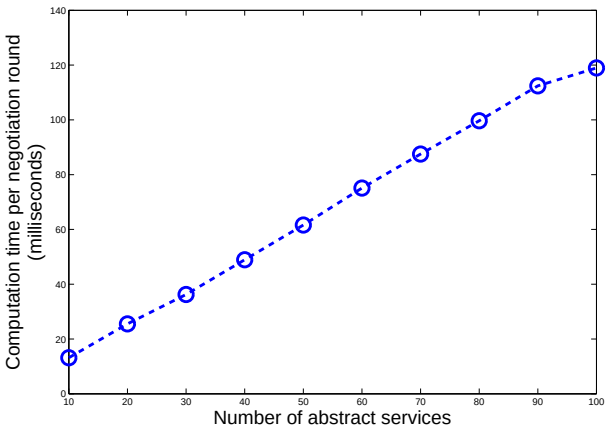


(b) varying number of service providers

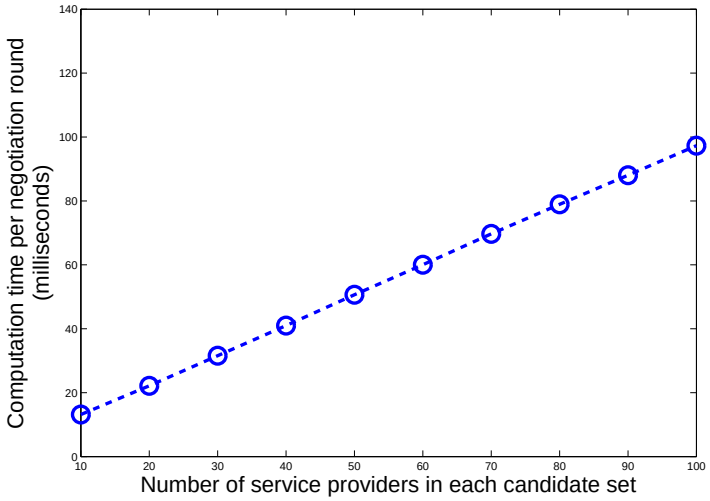


(c) varying number of data sets

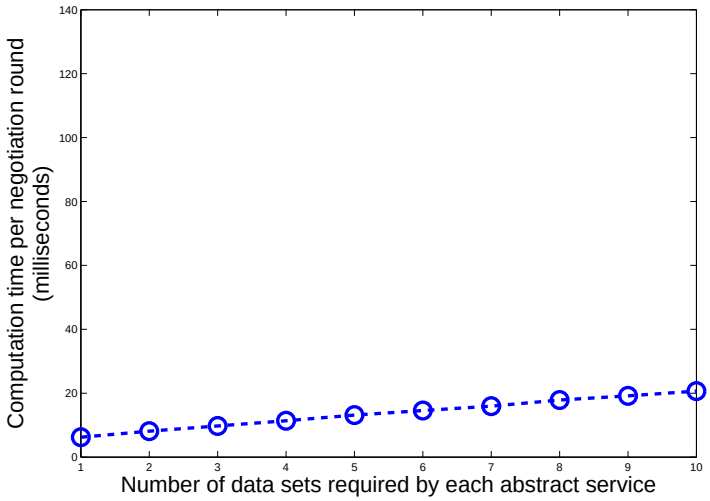
Figure 7.7: Number of negotiation rounds



(a) varying number of abstract services



(b) varying number of service providers



(c) varying number of data sets

Figure 7.8: Computation time per negotiation round

geographic distances in the cloud [1]. Given the dynamic distribution of data and data requests from users, the "best" replica platform for users to access data is determined by the replica selection process. The response time of data is the most relevant metric in the existing studies on replica selection. As this metric cannot be computed in advance, some other factors such as network and server performance are adopted to estimate the response time.

The literature presents two types of approaches for replica selection. The static replica selection approaches select the nearest replica according to static metrics, such as the geographical distance in miles, topological distance in number of hops, and HTTP request latency [68, 79, 80, 153]. However, the static metrics ignore the network dynamic conditions, so they are not sufficient predictors for the expected response time for user requests, which was illustrated by the experimental results [152, 201]. The dynamic replica selection approaches have emerged to improve the estimation of the expected response time based on network factors, such as round-trip time, network bandwidth, and server request latency [39, 69, 84, 140, 141].

Quite a few replica selection strategies are proposed for data Grid. Rahman et al. [140] designed a neural network algorithm to predict the transfer time for different sites that hold replicas, and applied a k-nearest neighbor rule to select the best replica. The goal was to minimize the access latency. Since the states of data replica change dynamically, the neural network algorithm does not always give the right decision. The misclassification in k-nearest neighbor rule will increase when large files are transferred. Also, the rule needs to save all previous requests, which needs time to search [6]. Bell et al. [19] adopted a reverse Vickrey auction to select the cheapest replica. The cost of a replica was proportional to the combination of access time and waiting time in the queue. However, the auction protocol was not efficient for single data request since it performed long-term optimization.

A few studies propose replica selection strategies based on bio-inspired algorithms [105, 123, 163]. Bio-inspired optimization algorithms have been proposed to solve the service provision problem, because of the simplicity of the algorithms and the rapid convergence to optimal or near-optimal solutions [89, 154, 176, 177, 179, 180, 181]. In order to deal with the dynamic changes of data replicas and network conditions in cloud computing, as well as the constraints of different users

and the flexibility of the selection criteria, an ant colony optimization algorithm was proposed to solve the data replica selection problem [175].

7.6.2 Negotiation in Service Provision

Negotiation has been adopted in service provision in order to get better QoS attributes. An iterative negotiation approach for a service composition was presented by He et al. [83]. The aim of the approach was to select services for the service-based systems in the scenarios where the QoS constraints were severe. Yan et al. [194] designed a framework in which the service level agreements for a service composition were established through autonomous agent negotiation. A new negotiation protocol was also proposed to support coordinated negotiation. Ardagna and Pernici [12] introduced an approach for the Web service selection problem with large scale processes and severe QoS constraints. The Web service selection problem was formalized as a mixed integer programming problem and loop peeling was adopted for optimization in that paper.

However, the negotiation approaches in the above studies are not able to effectively solve the problem in this research, in which the data plays the dominant role. The cost and response time of services largely depend on the accessing cost and response time of data sets. The negotiation for the problem is a multi-phase, multi-party process. The service composer should be able to negotiate with multiple service providers over each abstract service, while a service provider should be able to negotiate with the data providers over the data sets required by the services. To address the above issues, this chapter presents an ant-inspired negotiation approach.

7.7 Summary

This chapter proposed an ant-inspired negotiation approach for the data-intensive service provision problem. The two-stage negotiation processes provided effective and efficient service selection for service composers. A multi-phase, multi-party negotiation protocol was also designed, where the dynamic ant colony system was applied for selecting the services. The experimental results indicated that the ant-inspired negotiation approach was able to find a good solution in scenarios where

severe quality constraints were imposed on the problem.

Chapter 8

Discussions and Future Work

8.1 Main Implications of this Research

The data-intensive service composition problem with global QoS constraints is an extension of the traditional service composition problem, in which data sets, as the inputs and outputs of services, are incorporated. Services use data from data providers, and they also use data from other services. This research focuses on the data from data providers as this type of data is generally much larger.

To solve the problem, it is necessary to investigate the approaches to the traditional service composition problems. In service oriented computing, Web service selection is an important part of Web service composition. The Web service composition is typically achieved by solving the Web service concretization problem. The general overview of Web service concretization covers many aspects, such as the architecture, the composition structures, and the optimization approaches. The literature presents two types of Web service concretization approaches: local optimization approaches and global optimization approaches. By analyzing methods in the two types of approaches with respect to their optimality, their computational efficiency, and their dynamic complexity, the author found bio-inspired algorithms for QoS-aware service composition usually required much less computation time for large scale problems, especially in the dynamic environments. Then the applications of ant colony optimization algorithms, genetic algorithms, and particle swarm optimization algorithms to Web service concretization were reviewed. The systematic review based on the three bio-inspired algorithms helps us re-examine the models,

the basic operational flow, the main issues, and the limits of each algorithm.

Based on these foundational investigations, in this thesis an analysis of the cost-aware data-intensive service provision problem was made, which resulted in an economic model. This model enabled an analysis on the service and data usage and charging relationship. Based on the relationship, an extensible QoS model was presented to compute the cost and execution time of data-intensive services. Then the comprehensive applications of an ant colony system and a genetic algorithm were studied. Both algorithms were evaluated and compared with the mixed integer programming approach. The results indicated that both algorithms were able to achieve optimal solutions, and the genetic algorithm slightly outperformed the ant colony system when the problem was on a large scale. The objective functions of both algorithms were set to optimize the total cost of the composite service. It should be noted that here both algorithms were proposed to solve the single-objective optimization problem.

An evaluation of an ant colony system and a genetic algorithm for the multi-objective data-intensive service provision was also carried out. Both algorithms for a multi-objective context got a set of Pareto-optimal solutions by considering the total cost and the total execution time of a composite service. Since the two goals in a multi-objective optimization cannot be measured by one single performance metric, and the outcome of a multi-objective optimization run will generally consist of a varying number of non-dominated solutions, five performance metrics were adopted to evaluate the two algorithms. The experimental results indicated that when a large number of concrete services were available for each abstract service, a multi-objective genetic algorithm could achieve better solutions. On the other hand, whenever the number of concrete services was small, a multi-objective ant colony system would be preferred to a multi-objective genetic algorithm.

The experimental results of the applications of the ant colony system and the genetic algorithm to the single-objective and the multi-objective data-intensive service provision problem indicated that the ant colony system could find solutions more quickly in dynamic environments without restarting the optimization process, since the ants could react explicitly to the changes based on the pheromone information. Four strategies have been designed for modifying the pheromone information

to adapt the ant colony system to handle the dynamic scenarios when new services are provided, some services are discontinued, or the QoS attributes of some services are changed.

The economic model of the data-intensive service provision indicates that the composition will be cooperatively supported by the service composers, the service providers, and the data providers. Their decisions depend on each other. If the QoS attributes of a composite service cannot satisfy the requirements, it is expected that the service composer will negotiate with the service providers in order to improve the quality of services, further, the service providers will negotiate with the data providers in order to improve the quality of data provision. Meanwhile, in the negotiation process, the composition optimization scenarios are being changed dynamically. Thus, an ant-inspired negotiation approach was proposed to solve the dynamic service provision problem with severe constraints.

This thesis improves the understanding of the Web service selection and composition problem based on bio-inspired algorithms and advances the state-of-the-art research through its contributions. The applications of bio-inspired algorithms to the data-intensive service provision problem, when forming service compositions, have been comprehensively explained and analyzed. The results from the conducted simulations allow quantitative assessments on the feasibility of the bio-inspired algorithms, when compared to other algorithms that only guarantee to solve the problem in an optimal way. In the course of designing the bio-inspired mechanisms for the data-intensive service provision problems, the author also found several interesting issues that warrant additional investigations. In the following sections, an overview of some of them is provided and opportunities for future work is presented.

8.2 Limitations of the Current Work

The features and mechanisms of biological systems have been demonstrated to be useful for the provision of data-intensive services. The ant colony optimization algorithm and the genetic algorithm were applied to solve the cost-aware service provision problems. The experimental results indicated that the proposed algorithms outperformed other traditional methods. There are, however, some aspects of the

design of the algorithms and the optimization model that should be rethought when the size of the optimization problem grows further, or more complexity is incorporated in real environments.

First, as discussed in Section 3.3, in each algorithm, some main control parameters are involved. In the experiments, settings from the literature or trial testing methods were used to determine the most suitable values for all parameters of the ant colony system and the genetic algorithm. The performance of both algorithms may be improved if the influences of the parameters regarding the problems are better understood. Hence, the approaches for the adaptation of the parameters of ant colony optimization algorithms and genetic algorithms need to be investigated [58, 143, 161].

Second, as described in Section 4.2, the service composition problem is modeled as an AND/OR graph. From the graph, we can get a set of execution paths that identify all possible execution scenarios of the composite service. In the ant colony system, when an ant arrives at an 'AND split' vertex, it will make several copies of itself and apply the state transition rule to choose a successor. However, the computation of the probability of successors costs much time. In the genetic algorithm, the whole population moves like one group towards an optimal area and there is no extra computation cost. But the genetic algorithm cannot adapt to the dynamic scenarios of the optimization problem, whereas the ants can react to the changes based on the pheromone information. For the dynamic data-intensive service provision, faster models and cooperative algorithms will be required to improve the performance of the related algorithms.

8.3 Possible Solutions

8.3.1 Strategies to Lower the Total Cost

In the ant-inspired negotiation with severe global QoS constraints, the service providers and the data providers improve their offers based on their predefined ranges. This subsection outlines the strategies that may be used by the service providers and the data providers to lower the total cost of the composite service.

Data Pricing Strategy for Service Providers

For each service provider, one strategy to decrease the cost of a service is to choose different data pricing models. Except the usage-based pricing model and the package-based pricing model that described in Chapter 7, the data provider also offers the subscription-based pricing model. In the subscription-based pricing model, the data requesters need to pay once for the data set and afterwards they can access the data set for a period of time. The access cost of the data set in this period is independent of the number of usages. This pricing model is called the long-term option.

One service might require many data sets and one data set might also be required by many services. Let us assume that a service provider needs three data sets dt_1, dt_2, and dt_3, and the demands for these data sets are φ_1, φ_2, and φ_3 during a period of time. The prices of these data sets in the usage-based pricing model are p_1, p_2, and p_3. The data provider may also provide a package of the three data sets with a price p_{pk}, and $p_{pk} < p_1 + p_2 + p_3$. The service provider can choose the usage-based pricing model, the package-based pricing model, or the combination of both pricing models to access the three data sets. If the demands of the three data sets are same, namely, $\varphi_1 = \varphi_2 = \varphi_3$, the service provider prefers the package-based pricing model and the total access cost is $\varphi_1 * p_{pk}$. If the demands of the three data sets are different, and they are paid in the usage-based pricing model, the total access cost is given by (8.1).

$$\varphi_1 * p_1 + \varphi_2 * p_2 + \varphi_3 * p_3 \tag{8.1}$$

If the three data sets are paid in the combination pricing model, and demand φ_1 is paid in the package-based pricing model, and other demands are paid in the usage-based pricing model, there are three situations: 1) $\varphi_1 < \varphi_2$ and $\varphi_1 < \varphi_3$, 2) $\varphi_1 > \varphi_2$ and $\varphi_1 > \varphi_3$, 3) $\varphi_2 < \varphi_1 < \varphi_3$. For case 1, the combination pricing model is always better than the usage-based pricing model (8.1). For case 2, the combination pricing model is better than the usage-based pricing model (8.1) if the conditions

given by (8.2) are true.

$$
\begin{aligned}
&1 < \frac{\varphi_1}{\varphi_2} < \frac{p_2 + \frac{\varphi_3}{\varphi_2} p_3}{p_{pk} - p_1} \\
&1 < \frac{\varphi_1}{\varphi_3} < \frac{p_3 + \frac{\varphi_2}{\varphi_3} p_2}{p_{pk} - p_1} \\
&\frac{\varphi_2}{\varphi_3} < \frac{p_3}{p_{pk} - p_1 - p_2}, \quad \text{if } \varphi_2 > \varphi_3 \\
&\frac{p_{pk} - p_1 - p_3}{p_2} < \frac{\varphi_2}{\varphi_3}, \quad \text{if } \varphi_2 < \varphi_3
\end{aligned}
\tag{8.2}
$$

For case 3, the combination pricing model is better than the usage-based pricing model (8.1) if the condition given by (8.3) is true.

$$
\frac{\varphi_1}{\varphi_2} < \frac{p_2}{p_{pk} - p_1 - p_3} \tag{8.3}
$$

In (8.3), there is no constraint between φ_1 and φ_3. The reason is that for case 3, $\varphi_1 < \varphi_3$, the package pricing model with demand φ_1 for dt_1 and dt_3 is always better than the usage-based pricing model.

Further, a service provider will prefer to choose the subscription-based pricing model rather than a usage-based and package-based pricing model if the combination pricing model is not attractive anymore. During the service composition, the data provider will package the data sets based on the requests. If some data sets are always used together by many services, the data provider will package them and give a discount to the service providers.

Data Placement Strategy for Data Providers

The data provider may improve the quality of data by changing data placement policies. The data placement policies are about how data can be organized physically in order to minimize the cost and access latency of data sets. For example, the data providers may place data near the services or partition a large amount of data sets into several portions (or replicas if necessary). The following presents approaches to analyze the conditions, under which the data provider changes data placement policies.

Before the negotiations, we may assume that the data provider places all data

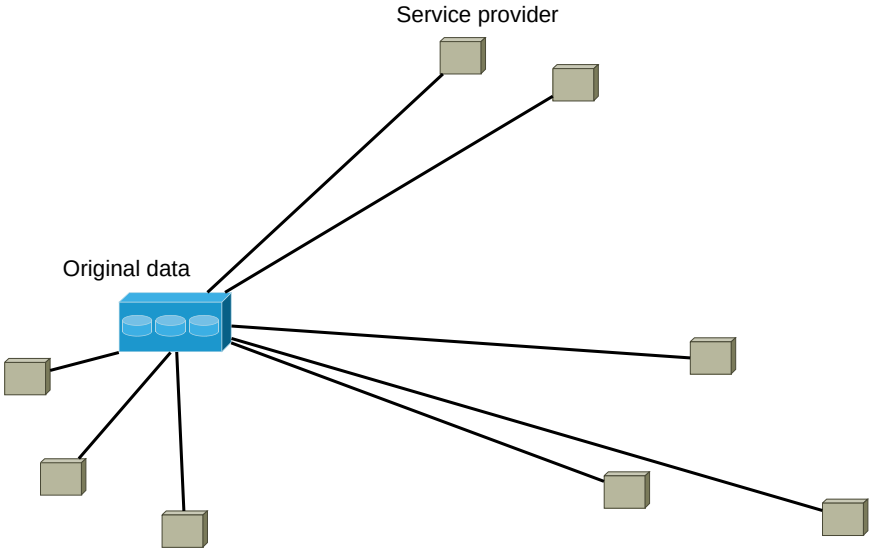


Figure 8.1: Before data partition and replication takes place

sets in a data center, and all service providers access data from this data center. Figure 8.1 illustrates the data placement before the negotiation. During the negotiations, the data provider needs to effectively store all data sets in order to maximize the data usage and the profit. In order to prevent data gathering to one data center and reduce the access response time of data sets, the data provider partitions all data sets and distributes these partitions to different data centers, and replicates portions of the data sets. Figure 8.2 illustrates the data placement after the negotiation. The data provider makes decisions based on the access frequency and update rate of the data sets as well as the locality of service providers. The service providers

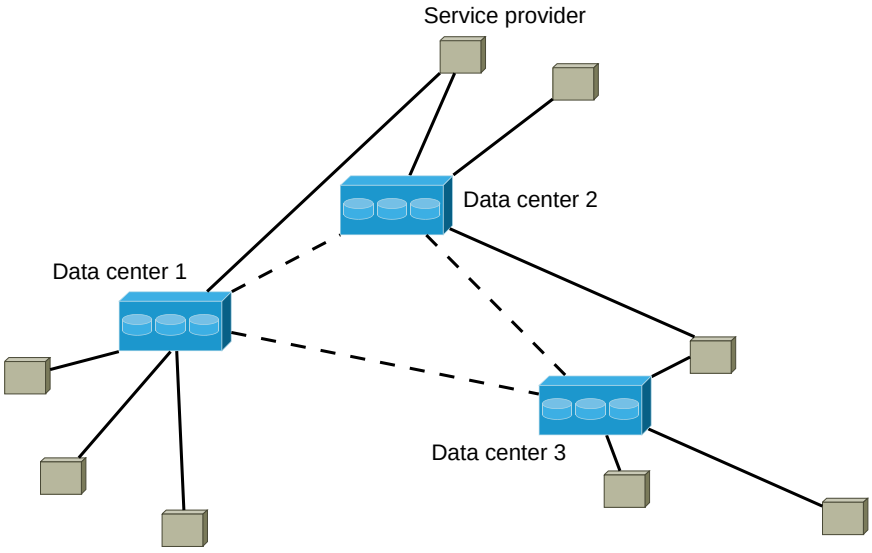


Figure 8.2: After data partition and replication has taken place

prefer the data sets to be placed in the closest data center as much as possible, as this will efficiently minimize the total data movement. If the service cannot find the required data sets from the closest data center, it needs to access them from the other data centers.

The data placement problem is very similar to the minimum K-median problem, especially from a facility location perspective [31, 41]. In a typical facility-location setting, we are given a set of clients with demands and a set of facilities with facility-opening costs. We want to open facilities to satisfy the requirements and assign the demands to facilities where the requirements of the clients can be satisfied, so as to minimize the sum of the facility-opening costs, supplies storage costs, and distance from each client to its closest facility. The data placement problem can be abstracted to such settings as: the data centers represent facilities, the data sets correspond to the different supplies required by the clients, the facility-opening cost models the cost of opening a data center, the supply storage cost models the cost of storing a data set, and the data center capacity imposes a restriction on the number of data sets that may be stored at a data center. The studies [145, 155] had investigated problems closely related to the data placement problem, as motivated by such facility-location applications.

The data placement problem is formalized via the following mathematical formulation. The data provider is given a set of data centers DC, a set of data requesters DR, and a set of data sets DT. The variable i is used to index the data center in DC, j to index the data requester in DR, and k to index the data set in DT. Each data center i can store data sets at most λ^i_{max}, and has a center-opening cost CO_i. Storing a data set k at data center i incurs a storage cost of ds^k_i. Let ω_{ij} denote the cost of transmitting a unit-size data set between data center i and data request j. The data requesters DR periodically issue access requests to the data sets DT, the rate of which is given by the demand function R. Each data requester j has the frequency R_{jk} to access data set k and has to be assigned to a data center that stores k. The cost of assigning data requester j to data center i to access data set k is denoted by $\psi^k_{ij} = R_{jk}\omega_{ij}size(k)$, where $size(k)$ is the size of data set k. The maximum storage usage λ^i_{max} bounds the total size of data sets that may be stored in data center i. The data provider seeks a placement of the data sets to data centers with

respect to the center-opening cost and capacities, and an assignment of requesters to data centers, so as to minimize the total opening cost, storage cost, and data requesters' access cost. More precisely, the data provider wants to determine the number of new data centers to open, a set of data sets $DT_i \subseteq DT$ to place at data center i ($|DT_i| \leq \lambda^i_{max}$), and a set of data center to assign each data requester. The objective is given by (8.4).

$$\min \Phi^1_{dp} = C_o + C_s + C_a \tag{8.4}$$

where

$$C_o = \sum_{i \in DC} CO_i \sigma_i$$

$$C_s = \sum_{i \in DC} \sum_{k \in DT} ds^k_i \sigma^k_i$$

$$C_a = \sum_{j \in DR} \sum_{\substack{k \in DT_i \\ i \in DC}} \psi^k_{ij} \varsigma^k_{ij}$$

subject to:

$$\sum_{i \in DC} \varsigma^k_{ij} = 1, \qquad j \in DR, k \in DT \tag{8.5}$$

$$\varsigma^k_{ij} \leq \sigma^k_i, \qquad i \in DC, j \in DR, k \in DT \tag{8.6}$$

$$\varsigma^k_{ij} \leq \sigma_i, \qquad i \in DC, j \in DR, k \in DT \tag{8.7}$$

$$\sum_{k \in DT} size(k) \sigma^k_i \leq \lambda^i_{max}, \qquad i \in DC \tag{8.8}$$

$$\varsigma^k_{ij}, \sigma_i, \sigma^k_i \in [0, 1], \qquad i \in DC, j \in DR, k \in DT \tag{8.9}$$

The variables C_o, C_s, C_a represent the center-opening cost, storage cost, and access cost. The variable σ_i indicates if data center i is open, σ^k_i indicates if data set k is stored at data center i, and ς^k_{ij} indicates if data requester j is assigned to data center i to access data set k. The constraint (8.5) states that for each data set, the data request must be assigned to a data center. Constraints (8.6) and (8.7) state that if data request j is assigned to data center i to access data set k, then k must be stored at i and i must be open. The constraint (8.8) states that the total size of data sets stored at data center i is at most its capacity λ^i_{max}.

The above formulation (8.4) of the data placement problem seems most suitable for data sets that are rarely updated. In the presence of update-requests, all the

copies of a data set replicated in the various data centers are consistent, and this requires that a write-request updates all the replicas of the data set. Let E_k denote the total number of update-requests for data set k, and c_k denote the cost to update data set k. Let $R_k = \sum_{j \in DR} R_{jk}$ denote the total demands for data set k. Let us assume that the total demands of a data set is larger than its update rate, that is to say, $E_k \leq R_k$. If this is not the case, the data set is stored at one data center. Then the objective is given by (8.10).

$$\min \Phi_{dp}^2 = C_o + C_s + C_a + C_u \tag{8.10}$$

where

$$C_u = \sum_{\substack{k \in DT_i \\ i \in DC}} E_k c_k \varrho_i^k$$

subject to:

$$\varrho_i^k \in [0, 1], \qquad i \in DC, k \in DT \tag{8.11}$$

The variable ϱ_i^k indicates if data set $k \in DT_i$ is updated.

All variables are initialized to 0 before the negotiation processes. During the negotiations, the data provider considers to change the data placement policies. The variable Ω_{ij} is used to denote the budget that the data requester j is willing to spend to get itself assigned to data center i. Intuitively Ω_{ij} is divided into four parts, which is given by (8.12).

$$\Omega_{ij} = \chi_{ij} + \phi_{ij} + \psi_{ij} + \mu_{ij} \tag{8.12}$$

A part of Ω_{ij} goes towards paying for the assignment cost $\psi_{ij} = \sum_{k \in DT_i} \psi_{ij}^k$. The rest of Ω_{ij} is divided into a payment for the data center opening cost χ_{ij}, a payment for the data set storage cost ϕ_{ij}, and a payment for data set update cost μ_{ij}. Before the data provider changes the data placement policies, let i_1 be the data center that data requester j is assigned to access data sets and $\Omega_{i_1,j}$ is the assigning cost. After data partition and replication has taken place, data requester j is assigned to data center i_2 to access data sets and $\Omega_{i_2,j}$ is the assigning cost. If $\Omega_{i_1,j} \geq \Omega_{i_2,j}$ and $\Omega_{i_2,j} \geq \Phi$, then data provider is willing to make changes. The condition $\Omega_{i_1,j} \geq \Omega_{i_2,j}$ states that the data requester wants to lower the cost, and the condition $\Omega_{i_2,j} \geq \Phi$ states

that the data requester needs to pay enough money to support the data provider to make changes, where Φ is the value of (8.10).

8.3.2 Economic Mechanisms for Data-Intensive Service Provision

The explosion of raw data and the dependence on data services are expected to be further amplified, as a result of the enormous proliferation of data-intensive services, for example, in critical areas such as disaster management and health care. Now cloud infrastructure and platforms have become viable mainstream solutions for data accessing, processing, analyzing, storage, and distribution. Because the primary motivation for moving to the cloud is to minimize cost (or maximize earnings), i.e., for economic reasons, we should attempt to find approaches to supporting economic data-intensive services provision from holistic perspectives.

This thesis has investigated the challenges and constraints of the data-intensive service provision problem. Currently, the ant colony optimization algorithm and the genetic algorithm have been proposed for selecting concrete services and data replicas to achieve a cost-effective solution as well as to achieve a set of Pareto-optimal solutions. However, the economic model in Section 4.2 may be extended to include the network providers, since the cost of transferring data and the response time of accessing data depend on the network bandwidth. Various providers need a standardized yet adaptive way to regulate and price their resources, which are specified by the utility of the stakeholders. Hence, the composite service should be constructed by achieving the Pareto optimum under the Nash equilibrium for the data-provider utility, the network-provider utility, the service-provider utility, and the service-composer utility.

As mentioned in Section 7.2, if a data set has more than one data replica, the service provider needs a data replica selection strategy to select data replicas, since mass data-moving influences the effectiveness and efficiency of the application execution. The literature has presented static replica selection approaches and dynamic replica selection approaches. Only very few studies proposed replica selection strategies based on bio-inspired algorithms. In order to achieve an economic solution for data-intensive service provision problems, novel mechanisms are needed to

be developed for selecting data replicas.

This thesis has presented an ant-inspired negotiation approach for the data-intensive service provision. In the real negotiation processes, it is useful for a service composer releases information about the status of the competition to the service providers, enabling them to make decisions about how to purchase the data and how to price their services. More complicated decision making tactics are needed to be investigated to generate counter-proposals for each provider. The overall objective is to build an economic model to ensure enhanced configuration for big data service provision, potentially saving money, energy and space for maintaining huge amount of data in future information infrastructures.

8.4 Summary

After reviewing the research work of this thesis, this chapter discussed possible improvements of the current work. Furthermore, it also provided insights on future work that could be carried out based on the work presented in this thesis. A conclusion with the contributions of this thesis will be presented in next chapter.

Chapter 9

Conclusions

This thesis started by describing, studying, and categorizing the Web service concretization with respect to the architecture, QoS attributes, and optimization approaches. It then created a hierarchical taxonomy of Web service concretization approaches. There are two types of Web service concretization approaches: local optimization approaches and global optimization approaches. The local optimization approaches are not suitable for QoS-based service selection with global QoS constraints, since they can only guarantee local QoS constraints and cannot satisfy the global QoS constraints. From the viewpoint of computational time, the local optimization approaches can be appropriate when the global QoS constraints are transformed into local QoS constraints. The global optimization approaches can solve service selection problems at both local and global service levels. There are three types of algorithmic methods in the global optimization approaches: optimal methods, sub-optimal methods, and soft constraints-based methods. These methods were evaluated with respect to their optimality, their computational efficiency, and their dynamic complexity. The detailed analysis of each method indicated that the bio-inspired algorithms, belonging to the sub-optimal methods, could overcome the new challenging requirements of the data-intensive service provision problem.

Further, this thesis focused on the bio-inspired algorithms-based Web service concretization. A systematic review of Web service concretization based on the ant colony optimization algorithms, the genetic algorithms, and the particle swarm optimization algorithms was conducted. For each type of algorithm, this thesis introduced the primary references, the models, and the basic operational flow. Then,

the main control parameters involved in each type of algorithm were discussed. This exercise disclosed the extent of applications of bio-inspired algorithms to Web service concretization problems.

The findings from the systematic review had been the basis for solving the problems. An economic model was established to represent and regulate the interactions among the service composers, the service providers, and the data providers. An extensible QoS model was also constructed to show the service and data set usage and charging relationship. Then this thesis applied an ant colony system and a genetic algorithm to compose data-intensive services. It also evaluated the two algorithms and compared them with other traditional methods. Both algorithms were proposed to optimize the total cost of a composite service.

This thesis then investigated a multi-objective ant colony system and a multi-objective genetic algorithm for the problem. Both the algorithms got a set of Pareto-optimal solutions by considering the total cost and the total execution time of a composite service at the same time. The two goals in a multi-objective optimization are the convergence to an approximation set and the maintenance of diversity in solutions of the Pareto-optimal set. These two goals cannot be measured adequately by one single performance metric. Experiments on many different scenarios with respect to five performance metrics were conducted to evaluate the proposed algorithms. The experimental results showed that when we had a large number of concrete services available for each abstract service, a multi-objective genetic algorithm could achieve better solutions. On the other hand, whenever the number of concrete services available was on a small scale, such as in some simple and repetitive scientific computations, a multi-objective ant colony system would be preferred to a multi-objective genetic algorithm.

This thesis also studied an ant colony system for the dynamic problem, where the changes of the states of services would occur, such as the changes of QoS attributes, the discontinuation of services, and the increase of new services. When a change of the search space occurred, the service composition optimization process should be conducted again. In order to adapt the ant colony system to handle the dynamic scenarios, this thesis discussed several strategies to modify the pheromone information. The performance of each pheromone modification strategy was eval-

uated. The experimental results indicated that the performance of each strategy depended not only on its strategy-specific parameter, but also on the number of changes and the frequency of occurrence of changes.

Taking further enhancements in the economic model, this thesis proposed an ant-inspired negotiation approach. In the context of Web service composition, if the QoS attributes of a composite service cannot satisfy the global constraints, it is useful to exploit the competitions among service providers to help the service composer obtain favorable services and find feasible solutions. The decisions of the service providers, the service composers, and the data providers depend on each other. Meanwhile, data-intensive services are used in a dynamic and changing environment, and different providers typically have conflicting objectives. In order to automate the process of reaching an agreement, a group of agents were exploited to establish agreeable service contracts. The lifetime of the data-intensive service provision and the two-stage negotiation processes were described. This thesis also designed a multi-phase, multi-party negotiation protocol, where an ant colony system was applied to select services. The experimental results indicated that the ant-inspired negotiation approach was able to find optimal solutions in scenarios where even severe QoS constraints were imposed on the problem.

This thesis has innovatively addressed several aspects related to the bio-inspired cost-aware data-intensive service provision. To solve the problems, it is important to know the extent of applications of bio-inspired algorithms to Web service concretization. Thus, this thesis began with investigating the Web service concretization in terms of architecture, QoS attributes, structures, and approaches. A systematic review of bio-inspired service concretization was conducted. Based on that, this thesis applied the ant colony system and the genetic algorithms to the single-objective and multi-objective cost-aware service provision in data-intensive context. This thesis also investigated the application of an ant colony system to the dynamic service provision. Finally, because different providers would negotiate with each other when the QoS constraints were severe, the thesis proposed an ant-inspired negotiation approach for the data-intensive service provision.

Appendix A

List of Articles Published During the Candidature

Journals

1. **L. Wang**, and J. Shen, "A Systematic Review of Bio-Inspired Service Concretization", submitted to IEEE Transactions on Service Computing, 2013.

2. **L. Wang**, J. Shen, and J. Luo, "Facilitating an Ant Colony Algorithm for Multi-Objective Data-Intensive Service Provision", submitted to Journal of Computer and System Sciences, 2014.

3. **L. Wang**, and J. Shen, "Multi-Phase Ant Colony System for Multi-Party Data-Intensive Service Provision", submitted to IEEE Transactions on Service Computing, 2014.

4. J. Shen, G. Beydoun, G. Low, and **L. Wang**, "Aligning Ontology-Based Development with Service Oriented Systems," *Future Generation Computer Systems*, 32: 263-273, 2014.

Refereed Conference Papers

1. **L. Wang**, J. Shen, and J. Yong, "A Survey on Bio-Inspired Algorithms for Web Service Composition," *Proceedings of the IEEE 16th International Conference on*

Computer Supported Cooperative Work in Design (CSCWD), pp. 569-574, IEEE Computer Society, Washington, DC, USA, 2012.

2. **L. Wang**, and J, Shen, "Towards Bio-Inspired Cost Minimisation for Data-Intensive Service Provision," *Proceedings of the IEEE 1st International Conference on Services Economics (SE)*, pp. 16-23, IEEE Computer Society, Washington, DC, USA, 2012.

3. **L. Wang**, J. Shen, C. Di, Y. Li, and Q. Zhou, "Towards Minimizing Cost for Composite Data Intensive Services," *Proceedings of the IEEE 17th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pp. 293-298, IEEE Computer Society, Washington, DC, USA, 2013.

4. **L. Wang**, J. Luo, J. Shen, and F. Dong, "Cost and Time Aware Ant Colony Algorithm for Data Replica in Alpha Magnetic Spectrometer Experiment," *Proceedings of the IEEE 2nd International Congress on Big Data (BigData Congress)*, pp. 247-254, IEEE Computer Society, Washington, DC, USA, 2013.

5. **L. Wang**, and J. Shen, "Economical Data-Intensive Service Provision Supported With a Modified Genetic Algorithm," *Proceedings of the IEEE 2nd International Congress on Big Data (BigData Congress)*, pp. 355-362, IEEE Computer Society, Washington, DC, USA, 2013.

6. **L. Wang**, J. Shen, and G. Beydoun, "Enhanced Ant Colony Algorithm for Cost-Aware Data-Intensive Service Provision," *Proceedings of the IEEE 9th World Congress on Services (SERVICES)*, pp. 227-234, IEEE Computer Society, Washington, DC, USA, 2013.

7. **L. Wang**, J. Shen, J. Luo, and F. Dong, "An Improved Genetic Algorithm for Cost-Effective Data-Intensive Service Composition," *Proceedings of the 9th International Conference on Semantics, Knowledge and Grids (SKG)*, pp. 105-112, IEEE Computer Society, Washington, DC, USA, 2013.

8. **L. Wang**, and J. Shen, "Bio-Inspired Cost-Effective Access to Big Data," accepted by *The International Symposium for Next Generation Infrastructure (ISNGI)*, Australia, 2013.

9. **L. Wang**, J. Shen, and J. Luo, "Impacts of Pheromone Modification Strategies in Ant Colony for Data-Intensive Service Provision", *Proceedings of the IEEE 21th International Conference on Web Services (ICWS)*, pp. 177-184, IEEE Computer Society, Washington, DC, USA, 2014.

10. **L. Wang**, J. Shen, Q. Zhou, and G. Beydoun, "Ant-Inspired Multi-Phase and Multi-Party Negotiations in the Data-Intensive Service Provision", *Proceedings of the IEEE 11th International Conference on Services Computing (SCC)*, 211-218, IEEE Computer Society, Washington, DC, USA, 2014.

Bibliography

[1] Abadi, D. J. (2009). Data Management in the Cloud: Limitations and Opportunities. *IEEE Data Engineering Bulletin*, *32*(1), 3–12.

[2] Aggarwal, R., Verma, K., Miller, J., and Milnor, W. (2004). Constraint Driven Web Service Composition in Meteor-S. In *Proceedings of IEEE International Conference on Services Computing (SCC '04)*. IEEE Computer Society, Washington, DC, USA, 23–30.

[3] Ai, L. and Tang, M. (2008). QoS-Based Web Service Composition Accommodating Inter-service Dependencies Using Minimal-Conflict Hill-Climbing Repair Genetic Algorithm. In *Proceedings of IEEE 4th International Conference on eScience (eScience '08)*. IEEE Computer Society, Washington, DC, USA, 119–126.

[4] Al-Mistarihi, H. H. E. and Yong, C. H. (2008). Response Time Optimization for Replica Selection Service in Data Grids. *Journal of Computer Science*, *4*(6), 487–493.

[5] Alibhai, Z. (2003). What is Contract Net Protocol. `http://www2.ensc.sfu.ca/research/iDEA/courses/files/ContractNetProtocol1.pdf`.

[6] Almuttairi, R. M., Wankar, R., Negi, A., Chillarige, R. R., and Almahna, M. S. (2010). New Replica Selection Technique for Binding Replica Sites in Data Grids. In *Proceedings of 1st International Conference on Energy, Power and Control (EPC-IQ)*. IEEE Computer Society, Washington, DC, USA, 187–194.

[7] Alonso, G., Casati, F., Kuno, H. A., and Machiraju, V. (2004). *Web Services - Concepts, Architectures and Applications*. Springer-Verlag, Berlin, Heidelberg.

[8] Alrifai, M., Risse, T., and Nejdl, W. (2012). A Hybrid Approach for Efficient Web Service Composition with End-to-End QoS Constraints. *ACM Transactions on the Web*, *6*(2), 7:1–7:31.

[9] Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic Local Alignment Search Tool. *Journal of Molecular Biology*, *215*(3), 403–410.

[10] Angelo, J. S. and Barbosa, H. J. C. (2011). On Ant Colony Algorithms for Multiobjective Problems. In *Ant Colony Optimization - Methods and Applications*, A. Ostfeld, Ed. InTech, Manhattan, NY, USA.

[11] Ardagna, D. and Pernici, B. (2006). Global and Local QoS Guarantee in Web Service Selection. In *Business Process Management Workshops*, C. J. Bussler and

A. Haller, Eds. Lecture Notes in Computer Science Series, vol. 3812. Springer-Verlag, Berlin, Heidelberg, 32–46.

[12] Ardagna, D. and Pernici, B. (2007). Adaptive Service Composition in Flexible Processes. *IEEE Transactions on Software Engineering*, *33*(6), 369–384.

[13] Armbrust, M., Fox, A., Joseph, R. G. A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., and Zaharia, M. (2009). Above the Clouds: A Berkeley View of Cloud Computing. Tech. Rep. UCB/EECS-2009-28, EECS Department, University of California, Berkeley, CA.

[14] Bacon, D. F., Graham, S. L., and Sharp, O. J. (1994). Compiler Transformations for High-Performance Computing. *ACM Computing Surveys*, *26*(4), 345–420.

[15] Balasubramaniam, S., Botvich, D., Carroll, R., Mineraud, J., Nakano, T., Suda, T., and Donnelly, W. (2011). Biologically Inspired Future Service Environment. *Computer Networks*, *55*(15), 3423–3440.

[16] Baran, B. and Schaerer, M. (2003). A Multiobjective Ant Colony System for Vehicle Routing Problem with Time Windows. In *Proceedings of 21st IASTED International Conference on Applied Informatics*. IASTED, Zurich, Switzerland, 97–102.

[17] Bastin, L., McInerney, D., Revez, G., Figueiredo, C., Simonetti, D., Barredo, J., Achard, F., and San-Miguel-Ayanz, J. (2012). Web Services for Forest Data, Analysis and Monitoring: Developments from EuroGEOSS. `http://www.earthzine.org/2012/07/25/`.

[18] Bell, G., Hey, T., and Szalay, A. (2009). Beyond the Data Deluge. *Science*, *323*(5919), 1297–1298.

[19] Bell, W. H., Cameron, D. G., Carvajal-Schiaffino, R., Millar, A. P., Stockinger, K., and Zini, F. (2003). Evaluation of an Economy-Based File Replication Strategy for A Data Grid. In *Proceedings of the 3rd International Symposium on Cluster Computing and the Grid (CCGRID '03)*. IEEE Computer Society, Washington, DC, USA, 661–668.

[20] Benatallah, B., Dumas, M., Sheng, Q. Z., and Ngu, A. H. H. (2002). Declarative Composition and Peer-to-Peer Provisioning of Dynamic Web Services. In *Proceedings of 18th International Conference on Data Engineering (ICDE '02)*. IEEE Computer Society, Washington, DC, USA, 297–308.

[21] Berbner, R., Spahn, M., Repp, N., Heckmann, O., and Steinmetz, R. (2006). Heuristics for QoS-Aware Web Service Composition. In *Proceedings of International Conference on Web Services (ICWS '06)*. IEEE Computer Society, Washington, DC, USA, 72–82.

[22] Berkelaar, M., Eikland, K., and Notebaert, P. (2004). lpsolve: Open Source (mixed-integer) Linear Programming System. `http://lpsolve.sourceforge.net/`.

[23] Berrick, S. W., Leptoukh, G., Farley, J. D., and Rui, H. (2009). Giovanni: A Web Service Workflow-Based Data Visualization and Analysis System. *IEEE Transactions on Geoscience and Remote Sensing*, *47*(1), 106–113.

[24] Berriman, G. B., Deelman, E., Good, J., Jacob, J. C., Katz, D. S., Laity, A. C., Princeomas, T. A., Singh, G., and Su, M. H. (2007). Generating Complex Astronomy Workflows. In *Workflows for e-Science*, I. J. Taylor, E. Deelman, D. B. Gannon, and M. Shields, Eds. Springer-Verlag, Berlin, Heidelberg, 19–38.

[25] Bless, P. N., Klabjan, D., and Chang, S. Y. (2008). Heuristics for Automated Knowledge Source Integration and Service Composition. *Computers and Operations Research*, *35*(4), 1292–1314.

[26] Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., and Orchard, D. (2004). Web Services Architecture. Tech. Rep. W3C Working Group Note, W3C. `http://www.w3.org/TR/ws-arch`.

[27] Brahmi, Z. and Gammoudi, M. M. (2013). QoS-Aware Automatic Web Service Composition Based on Cooperative Agents. In *Proceedings of IEEE 22nd International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*. IEEE Computer Society, Washington, DC, USA, 27–32.

[28] Brown, A., Johnston, S., and Kelly, K. (2002). Using Service-Oriented Architecture and Component Based Development to Build Web Service Applications. Tech. Rep. Rational Software White Paper TP032, Rational Software Corporation.

[29] Bryant, R. E., Katz, R. H., and Lazowska, E. D. (2008). Big-Data Computing: Creating Revolutionary Breakthroughs in Commerce, Science, and Society. Computing Research Initiatives for the 21st Century.

[30] Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., and Brandic, I. (2009). Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility. *Future Generation Computer Systems*, *25*(6), 599–616.

[31] Byrka, J. (2007). An Optimal Bifactor Approximation Algorithm for the Metric Uncapacitated Facility Location Problem. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, M. Charikar, K. Jansen, O. Reingold, and J. D. P. Rolim, Eds. Lecture Notes in Computer Science Series, vol. 4627. Springer-Verlag, Berlin, Heidelberg, 29–43.

[32] Callaghan, S., Deelman, E., Gunter, D., Juve, G., Maechling, P., Brooks, C., Vahi, K., Milner, K., Graves, R., Field, E., Okaya, D., and Jordan, T. (2010). Scaling Up Workflow-Based Applications. *Journal of Computer and System Sciences*, *76*(6), 428–446.

[33] Canfora, G., Penta, M., Espositio, R., and Villani, M. L. (2005). An Approach for QoS-Aware Service Composition Based on Genetic Algorithms. In *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation (GECCO '05)*. ACM, New York, NY, USA, 1069–1075.

[34] Canfora, G., Penta, M., Esposito, R., and Villani, M. L. (2004). A Lightweight Approach for QoS Aware Service Composition. In *Proceedings of 2nd International Conference on Service Oriented Computing (ICSOC '04)*. ACM, New York, NY, USA, 36–47.

[35] Cao, L., Li, M., and Cao, J. (2005). Cost-Driven Web Service Selection Using Genetic Algorithm. In *Internet and Network Economics*, X. Deng and Y. Ye, Eds. Lecture Notes in Computer Science Series, vol. 3828. Springer-Verlag, Berlin, Heidelberg, 906–915.

[36] Cardellini, V., Casalicchio, E., Grassi, V., and Presti, F. L. (2007). Efficient Provisioning of Service Level Agreements for Service Oriented Applications. In *Proceedings of 2nd International Workshop on Service Oriented Software Engineering: in Conjunction with the 6th ESEC/FSE Joint Meeting (IW-SOSWE '07)*. ACM, New York, NY, USA, 29–35.

[37] Cardoso, J., Sheth, A., Miller, J., Arnold, J., and Kochut, K. (2004). Quality of Service for Workflows and Web Service Processes. *Journal of Web Semantics*, *1*(3), 281–308.

[38] Casati, F., Sayal, M., and Shan, M. C. (2001). Developing e-Services for Composing e-Services. In *Proceedings of 13th International Conference on Advanced Information Systems Engineering (CSiSE '01)*. Springer-Verlag, Berlin, Heidelberg, 171–186.

[39] Chang, R. S. and Chen, P. H. (2007). Complete and Fragmented Replica Selection and Retrieval in Data Grids. *Future Generation Computer Systems*, *23*(4), 536–546.

[40] Chang, W., Wu, C., and Chang, C. (2005). Optimizing Dynamic Web Service Component Composition by Using Evolutionary Algorithms. In *Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence*. IEEE Computer Society, Washington, DC, USA, 708–711.

[41] Charikar, M., Guha, S., Tardos, E., and Shmoys, D. B. (2002). A Constant-Factor Approximation Algorithm for the k-Median Problem. *Journal of Computer and System Sciences*, *65*(1), 129–149.

[42] Chhetri, M. B., Lin, J., Goh, S., Zhang, J., Kowalczyk, R., and Yan, J. (2006). A Coordinated Architecture for the Agent-Based Service Level Agreement Negotiation of Web Service Composition. In *Proceedings of the Australian Software Engineering Conference (ASWEC '06)*. IEEE Computer Society, Washington, DC, USA, 90–99.

[43] Claro, D. B., Albers, P., and Hao, J. (2005). Selecting Web Services for Optimal Composition. In *Proceedings of 2nd International Workshop Semantic and Dynamic Web Processes (SDWP 2005)*. IEEE Computer Society, Washington, DC, USA, 32–45.

[44] Claro, D. B., Albers, P., and Hao, J. (2006). Web Services Composition. In *Semantic Web Services, Processes and Applications*, J. Cardoso and A. P. Sheth,

Eds. Semantic Web and Beyond Series, vol. 3. Springer-Verlag, Berlin, Heidelberg, 195–225.

[45] Comes, D., Baraki, H., Reichle, R., Zapf, M., and Geihs, K. (2010). Heuristic Approaches for QoS-Based Service Selection. In *Service-Oriented Computing*, P. P. Maglio, M. Weske, J. Yang, and M. Fantinato, Eds. Lecture Notes in Computer Science Series, vol. 6470. Springer-Verlag, Berlin, Heidelberg, 441–455.

[46] Comuzzi, M. and Pernici, B. (2005). An Architecture for Flexible Web Service QoS Negotiation. In *Proceedings of the 9th IEEE International EDOC Enterprise Computing Conference (EDOC '05)*. IEEE Computer Society, Washington, DC, USA, 70–82.

[47] Cornford, D., Jones, R., Bastin, L., Williams, M., Pebesma, E., and Nativi, S. (2010). UncertWeb: Chaining Web Services Accounting for Uncertainty. Geophysical Research Abstracts, EGU2010-9052.

[48] DataMass (2012). Workshop on Massive Data Analytics on Scalable Systems. `http://ceng.usc.edu/~simmhan/DataMASS2012/`.

[49] Deb, K. (2000). An Efficient Constraint Handling Method for Genetic Algorithms. *Computer Methods in Applied Mechanics and Engineering*, *186*(2-4), 311–338.

[50] Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Chichester, UK.

[51] Deb, K. and Agrawal, R. B. (1995). Simulated Binary Crossover for Continuous Search Space. *Complex Systems*, *9*(2), 115–148.

[52] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197.

[53] Dolgert, A., Gibbons, L., Jones, C. D., Kuznetsov, V., Riedewald, M., Riley, D., Sharp, G. J., and Wittich, P. (2008). Provenance in High-Energy Physics Workflows. *Computing in Science and Engineering*, *10*(3), 22–29.

[54] Dorigo, M. and Gambardella, L. M. (1997). Ant Colony System: a Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation*, *1*(1), 53–66.

[55] Dorigo, M. and Stutzle, T. (2004). *Ant Colony Optimization*. MIT Press, Cambridge, MA, USA.

[56] Duderstadt, J. J. (2001). The Future of the University in the Digital Age. *Proceedings of the American Philosophical Society*, *145*(1), 54–72.

[57] Dustdar, S. and Papazoglou, M. P. (2008). Services and Service Composition - An introduction. *Information Technology*, *50*(2), 86–92.

[58] Eiben, A. E. and Smit, S. K. (2011). Parameter Tuning for Configuring and Analyzing Evolutionary Algorithms. *Swarm and Evolutionary Computation*, *1*(1), 19–31.

[59] El Azouzi, R., Altman, E., and Wynter, L. (2003). Telecommunications Network Equilibrium with Price and Quality-of-Service Characteristics. In *Providing Quality of Service in Heterogeneous Environments Proceedings of the 18th International Teletraffic Congress - ITC-18*, J. Charzinski, R. Lehnert, and P. Tran-Gia, Eds. Teletraffic Science and Engineering Series, vol. 5. Elsevier B.V., Amsterdam, Netherlands, 369–378.

[60] El Hadad, J., Manouvrier, M., and Rukoz, M. (2010). TQoS: Transactional and QoS-Aware Selection Algorithm for Automatic Web Service Composition. *IEEE Transactions on Services Computing*, *3*(1), 73–85.

[61] Erl, T. (2005). *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall, Upper Saddle River, NJ, USA.

[62] Estévez-Ayres, I., Basanta-Val, P., Garcia-Valls, M., Fisteus, J. A., and Almeida, L. (2009). QoS-Aware Real-Time Composition Algorithms for Service-Based Applications. *IEEE Transactions on Industrial Informatics*, *5*(3), 278–288.

[63] Eyckelhof, C. J. and Snoek, M. (2002). Ant Systems for a Dynamic TSP - Ants Caught in a Traffic Jam. In *Proceedings of ANTS 2002 - From Ant Colonies to Artificial Ants: 3rd International Workshop on Ant Algorithms*, M. Dorigo, G. Caro, and M. Sampels, Eds. Lecture Notes in Computer Science Series, vol. 2463. Springer-Verlag, Berlin, Heidelberg, 88–99.

[64] Fan, X. Q., Fang, X. W., and Jiang, C. J. (2011). Research on Web Service Selection Based on Cooperative Evolution. *Expert Systems with Applications*, *38*(8), 9736–9743.

[65] Fan, X. Q., Jiang, C. J., and Fang, X. W. (2009). An Efficient Approach to Web Service Selection. In *Web Information Systems and Mining*, W. Liu, X. Luo, F. Wang, and J. Lei, Eds. Lecture Notes in Computer Science Series, vol. 5854. Springer-Verlag, Berlin, Heidelberg, 271–280.

[66] Fang, Q., Peng, X., Liu, Q., and Hu, Y. (2009). A Global QoS Optimizing Web Services Selection Algorithm Based on MOACO for Dynamic Web Service Composition. In *Proceedings of International Forum on Information Technology and Applications (IFITA '09)*. Vol. 1. IEEE Computer Society, Washington, DC, USA, 37–42.

[67] Faratin, P., Sierra, C., and Jennings, N. R. (1998). Negotiation Decision Functions for Autonomous Agents. *Robotics and Autonomous Systems*, *24*(3-4), 159–182.

[68] Fei, Z., Bhattacharjee, S., Zegura, E. W., and Ammar, M. H. (1998). A Novel Server Selection Technique for Improving the Response Time of a Replicated Service. In *Proceedings of the 17th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '98)*. Vol. 2. 783–791.

[69] Ferdean, C. and Makpangou, M. (2003). A Scalable Replica Selection Strategy Based on Flexible Contracts. In *Proceedings of the 3rd IEEE Workshop on Internet Applications (WIAPP '03)*. IEEE Computer Society, Washington, DC, USA, 95–99.

[70] Ferreira, A. M., Kritikos, K., and Pernici, B. (2009). Energy-Aware Design of Service-Based Applications. In *Service-Oriented Computing*, L. Baresi, C. H. Chi, and J. Suzuki, Eds. Lecture Notes in Computer Science Series, vol. 5900. Springer-Verlag, Berlin, Heidelberg, 99–114.

[71] FIPA. (2002). Iterated Contract Net Interaction Protocol Specification. `http://www.fipa.org/specs/fipa00030/SC00030H.pdf`.

[72] Fonseca, C. M. and Fleming, P. J. (1996). On the Performance Assessment and Comparison of Stochastic Multiobjective Optimizers. In *Parallel Problem Solving from Nature - PPSN IV*, H. M. Voigt, W. Ebeling, I. Rechenberg, and H. P. Schwefel, Eds. Lecture Notes in Computer Science Series, vol. 1141. Springer-Verlag, Berlin, Heidelberg, 584–593.

[73] Fujii, K. and Suda, T. (2005). Semantics-Based Dynamic Service Composition. *IEEE Journal on Selected Areas in Communications*, *23*(12), 2361–2372.

[74] Gantz, J. and Reinsel, D. (2011). Extracting Value from Chaos. Tech. rep., IDC research report, Framingham, MA, USA.

[75] Gao, C., Cai, M., and Chen, H. (2007). QoS-Aware Service Composition Based on Tree-Coded Genetic Algorithm. In *Proceedings of 31st Annual International Computer Software and Applications Conference (COMPSAC 2007)*. IEEE Computer Society, Washington, DC, USA, 361–367.

[76] Guan, Y., Ghose, A. K., and Lu, Z. (2006). Using Constraint Hierarchies to Support QoS-Guided Service Composition. In *Proceedings of the IEEE International Conference on Web Services (ICWS '06)*. IEEE Computer Society, Washington, DC, USA, 743–752.

[77] Guntsch, M. and Middendorf, M. (2001). Pheromone Modification Strategies for Ant Algorithms Applied to Dynamic TSP. In *Applications of Evolutionary Computing*, E. J. W. Boers, Ed. Lecture Notes in Computer Science Series, vol. 2037. Springer-Verlag, Berlin, Heidelberg, 213–222.

[78] Guntsch, M. and Middendorf, M. (2002). Applying Population Based ACO to Dynamic Optimization Problems. In *Ant Algorithms*, M. Dorigo, G. Caro, and M. Sampels, Eds. Lecture Notes in Computer Science Series, vol. 2463. Springer-Verlag, Berlin, Heidelberg, 111–122.

[79] Guyton, J. D. and Schwartz, M. F. (1995). Locating Nearby Copies of Replicated Internet Servers. *SIGCOMM Computer Communication Review*, *25*(4), 288–298.

[80] Gwertzman, J. S. and Seltzer, M. (1995). The Case for Geographical Push-Caching. In *Proceedings of the 5th Workshop on Hot Topics in Operating Systems (HOTOS '95)*. IEEE Computer Society, Washington, DC, USA, 51–55.

[81] Hashmi, K., Alhosban, A., Malik, Z., and Medjahed, B. (2011). WebNeg: A Genetic Algorithm Based Approach for Service Negotiation. In *Proceedings of the IEEE International Conference on Web Services (ICWS)*. IEEE Computer Society, Washington, DC, USA, 105–112.

[82] He, Q., Yan, J., Jin, H., and Yang, Y. (2014). Quality-Aware Service Selection for Service-Based Systems Based on Iterative Multi-Attribute Combinatorial Auction. *IEEE Transactions on Software Engineering*, *40*(2), 192–215.

[83] He, Q., Yang, Y., Yan, J., and Jin, H. (2012). INSC: An Iterative Negotiation Approach for Service Compositions. In *Proceedings of the IEEE 9th International Conference on Services Computing (SCC)*. IEEE Computer Society, Washington, DC, USA, 170–177.

[84] Ho, T. and Abramson, D. (2005). The GriddLeS Data Replication Service. In *Proceedings of the 1st International Conference on e-Science and Grid Computing (E-SCIENCE '05)*. IEEE Computer Society, Washington, DC, USA, 271–278.

[85] Hock-Koon, A. and Oussalah, M. (2010). Many to Many Service Discovery: a First Approach. In *Proceedings of the 4th European Conference on Software Architecture (ECSA '10)*. Springer-Verlag, Berlin, Heidelberg, 449–456.

[86] Hoschek, W., Jaen-Martinez, J., Samar, A., Stockinger, H., and Stockinger, K. (2000). Data Management in an International Data Grid Project. In *Grid Computing – GRID 2000*, R. Buyya and M. Baker, Eds. Lecture Notes in Computer Science Series, vol. 1971. Springer-Verlag, Berlin, Heidelberg, 77–90.

[87] Huang, A. F. M., Lan, C. W., and Yang, S. J. H. (2009). An Optimal QoS-Based Web Service Selection Scheme. *Information Sciences*, *179*(19), 3309–3322.

[88] IBM. (2003). Web Service Level Agreement (WSLA) Language Specification (Version 1.0). `http://www.research.ibm.com/wsla`.

[89] Ismail, A., Yan, J., and Shen, J. (2013). Incremental Service Level Agreements Violation Handling with Time Impact Analysis. *Journal of Systems and Software*, *86*(6), 1530–1544.

[90] Ivanović, D., Carro, M., and Hermenegildo, M. V. (2012). A Constraint-Based Approach to Quality Assurance in Service Choreographies. In *Service-Oriented Computing*, C. Liu, H. Ludwig, F. Toumani, and Q. Yu, Eds. Lecture Notes in Computer Science Series, vol. 7636. Springer-Verlag, Berlin, Heidelberg, 252–267.

[91] Jaeger, E., Altintas, I., Zhang, J., Ludascher, B., Pennington, D., and Michener, W. (2005a). A Scientific Workflow Approach to Distributed Geospatial Data Processing Using Web Services. In *Proceedings of the 17th International Conference on Scientific and Statistical Database Management (SSDBM)*. Lawrence Berkeley Laboratory, Berkeley, CA, US, 87–90.

[92] Jaeger, M. (2007). Optimising Quality-of-Service for the Composition of Electronic Services. Ph.D. thesis, Berlin University of Technology.

[93] Jaeger, M. C. and Muehl, G. (2007). QoS-Based Selection of Services: The Implementation of a Genetic Algorithm. In *Proceedings of KiVS 2007 Workshop: Service-Oriented Architectures and Service-Oriented Computing (SOA/SOC)*, T. Braun, G. Carle, and B. Stiller, Eds. VDE, Bern, Switzerland, 359–370.

[94] Jaeger, M. C., Muehl, G., and Golze, S. (2005b). QoS-Aware Composition of Web Services: An Evaluation of Selection Algorithms. In *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE*, R. Meersman and Z. Tari, Eds. Lecture Notes in Computer Science Series, vol. 3760. Springer-Verlag, Berlin, Heidelberg, 646–661.

[95] Jamil, E. (2009). What Really is SOA. A Comparison With Cloud Computing, Web 2.0, SaaS, WOA, Web Services, PaaS and Others. Tech. rep., Southborough, MA, USA. White Paper.

[96] Jennings, N. R., Faratin, P., Lomuscio, A. R., Parsons, S., Wooldridge, M. J., and Sierra, C. (2001). Automated Negotiation: Prospects, Methods and Challenges. *Group Decision and Negotiation*, *10*(2), 199–215.

[97] Johnston, S. (2009). Diagram Showing Overview of Cloud Computing Including Google, Salesforce, Amazon, Microsoft, Yahoo & Zoho. `http://en.wikipedia.org/wiki/File:Cloudcomputing.svg`.

[98] Kandaswamy, G., Fang, L., Huang, Y., Shirasuna, S., Marru, S., and Gannon, D. (2006). Building Web Services for Scientific Grid Applications. *IBM Journal of Research and Development*, *50*(2/3), 249–260.

[99] Khalaf, R. and Leymann, F. (2003). On Web Services Aggregation. In *Technologies for e-Services*, B. Benatallah and M. C. Shan, Eds. Lecture Notes in Computer Sciences Series, vol. 2819. Springer-Verlag, Berlin, Heidelberg, 1–13.

[100] Kitchenham, B. and Charters, S. (2007). Guidelines for Performing Systematic Literature Reviews in Software Engineering. Tech. Rep. EBSE-2007-001, Keele University and Durham University Joint Report. `http://www.dur.ac.uk/ebse/resources/`.

[101] Knowles, J. (2005). A Summary-Attainment-Surface Plotting Method for Visualizing the Performance of Stochastic Multiobjective Optimizers. In *Proceedings of 5th International Conference on Intelligent Systems Design and Applications (ISDA '05)*. IEEE Computer Society, Washington, DC, USA, 552–557.

[102] Kobti, Z. and Wang, Z. (2007). An Adaptive Approach for QoS-Aware Web Service Composition Using Cultural Algorithms. In *AI 2007: Advances in Artificial Intelligence*, M. A. Orgun and J. Thornton, Eds. Lecture Notes in Computer Science Series, vol. 4830. Springer-Verlag, Berlin, Heidelberg, 140–149.

[103] Krishnan, S., Steam, B., Bhatia, K., Baldridge, K. K., Li, W. W., and Arzberger, P. (2006). Opal: Simple Web Services Wrappers for Scientific Applications (ICWS '06). In *Proceedings of International Conference on Web Services*. IEEE Computer Society, Los Alamitos, CA, USA, 823–832.

[104] Kritikos, K. and Plexousakis, D. (2009). Requirements for QoS-Based Web Services Description and Discovery. *IEEE Transactions on Services Computing*, *2*(4), 320–337.

[105] Kulkarni, S., Ganguly, N., Canright, G., and Deutsch, A. (2006). A New Bio-Inspired Location Search Algorithm for Peer to Peer Network Based Internet Telephony. In *Proceedings of the 1st International Conference on Bio Inspired Models of Network, Information and Computing Systems (BIONETICS '06)*. ACM, New York, NY, USA, 1–5.

[106] Lécué, F., Silva, E., and Pires, L. F. (2008). A Framework for Dynamic Web Services Composition. In *Emerging Web Services Technology, Volume II*, T. Gschwind and C. Pautasso, Eds. Whitestein Series in Software Agent Technologies and Autonomic Computing. Birkhäuser, Basel, Switzerland, 59–75.

[107] Lee, N. (2013). Generation C in the Age of Big Data. In *Facebook Nation*. Springer-Verlag, New York, NY, USA, 77–82.

[108] Li, F., Yang, F., Shuang, K., and Su, S. (2007). Q-Peer: A Decentralized QoS Registry Architecture for Web Services. In *Service-Oriented Computing - ICSOC 2007*, B. Krämer, K. J. Lin, and P. Narasimhan, Eds. Lecture Notes in Computer Science Series, vol. 4749. Springer-Verlag, Berlin, Heidelberg, 145–156.

[109] Li, H., Su, S. Y. W., and Lam, H. (2006). On Automated e-Business Negotiations: Goal, Policy, Strategy, and Plans of Decision and Action. *Journal of Organizational Computing and Electronic Commerce*, *13*(1), 1–29.

[110] Li, L., Yang, P., Ou, L., Zhang, Z., and Cheng, P. (2010). Genetic Algorithm-Based Multi-Objective Optimization for QoS-Aware Web Services Composition. In *Knowledge Science, Engineering and Management*, Y. Bi and M. A. Williams, Eds. Lecture Notes in Computer Science Series, vol. 6291. Springer-Verlag, Berlin, Heidelberg, 549–554.

[111] Li, Y. and Lin, C. (2011). QoS-Aware Service Composition for Workflow-Based Data-Intensive Applications. In *Proceedings of IEEE International Conference on Web Services*. IEEE Computer Society, Washington, DC, USA, 452–459.

[112] Li, Y., Tong, W., Zhi, X., and Ding, D. (2012). Dynamic Service Selection Based on Ant Colony System. In *Applied Mechanics and Materials*, X. Huang, X. Huang, H. Mao, and Z. Yin, Eds. Vol. 182-183. Trans Tech Publications, Dürnten, Switzerland, 2136–2141.

[113] Liang, Q. H. A. and Su, S. Y. W. (2005). ANDOR Graph and Search Algorithm for Discovering Composite Web Services. *International Journal of Web Services Research*, *2*(4), 46–64.

[114] Liang, W. Y. and Huang, C. C. (2009). The Generic Genetic Algorithm Incorporates with Rough Set Theory - An Application of the Web Services Composition. *Expert Systems with Applications*, *36*(3), 5549–5556.

[115] Liepins, G. E. and Vose, M. D. (1990). Representational Issues in Genetic Optimization. *Journal of Experimental and Theoretical Artificial Intelligence*, *2*(2), 4–30.

[116] Lindquist, E. J., McInerney, D., Annunzio, R. D., and Barredo, J. (2012). Improving Access, Utility and Analyses of FAO Forestry Statistics via Geographic Web-Services Earthzine. `http://www.earthzine.org/2012/06/01/`.

[117] Liu, J., Li, J., Liu, K., and Wei, W. (2007). A Hybrid Genetic and Particle Swarm Algorithm for Service Composition. In *Proceedings of 6th International Conference on Advanced Language Processing and Web Information Technology*. IEEE Computer Society, Washington, DC, USA, 564–567.

[118] Liu, S., Liu, Y., Jing, N., Tang, G., and Tang, Y. (2005). A Dynamic Web Service Selection Strategy with QoS Global Optimization Based on Multi-Objective Genetic Algorithm. In *Grid and Cooperative Computing - GCC 2005*, H. Zhuge and G. C. Fox, Eds. Lecture Notes in Computer Science Series, vol. 3795. Springer-Verlag, Berlin, Heidelberg, 84–89.

[119] Ma, Y. and Zhang, C. (2008). Quick Convergence of Genetic Algorithm for QoS-Driven Web Service Selection. *Computer Networks*, *52*(5), 1093–1104.

[120] Mabrouk, N. B., Beauche, S., Kuznetsova, E., Georgantas, N., and Issarny, V. (2009). QoS-Aware Service Composition in Dynamic Service Oriented Environments. In *Middleware 2009*, J. M. Bacon and B. F. Cooper, Eds. Lecture Notes in Computer Science Series, vol. 5896. Springer-Verlag, Berlin, Heidelberg, 123–142.

[121] Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., and Byers, A. H. (2011). Big Data: The Next Frontier for Innovation, Competition, and Productivity. Tech. rep., McKinsey Global Institute.

[122] Mardukhi, F., NematBakhsh, N., Zamanifar, K., and Barati, A. (2013). QoS Decomposition for Service Composition Using Genetic Algorithm. *Applied Soft Computing*, *13*(7), 3409–3421.

[123] Méndez Muñoz, V., Amoróz Vicente, G., García Carballeira, F., and Salt Cairols, J. (2010). Emergent Algorithms for Replica Location and Selection in Data Grid. *Future Generation Computer Systems*, *26*(7), 934–946.

[124] Milanovia, N. and Malek, M. (2004). Current Solutions for Web Service Composition. *IEEE Internet Computing*, *8*(6), 51–59.

[125] Milenkovic, M., Castro-Leon, E., and Blakley, J. R. (2009). Power-Aware Management in Cloud Data Centers. In *Cloud Computing*, M. Jaatun, G. Zhao, and C. Rong, Eds. Lecture Notes in Computer Science Series, vol. 5931. Springer-Verlag, Berlin, Heidelberg, 668–673.

[126] Miller, R. (2011). 'Digital Universe' to Add 1.8 Zettabytes in 2011. `http://www.datacenterknowledge.com/archives/2011/06/28/digital-universe-to-add-1-8-zettabytes-in-2011/`.

[127] Montagnat, J., Glatard, T., and Lingrand, D. (2006). Data Composition Patterns in Service-Based Workflows. In *Proceedings of Workshop on Workflows in Support of Large-Scale Science (WORKS '06)*. ACM, New York, NY, USA, 1–10.

[128] Nakano, T. (2011). Biologically Inspired Network Systems: A Review and Future Prospects. *IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and reviews*, *41*(5), 630–643.

[129] Nefedova, V., Jacob, R., Foster, I. T., Liu, Z., Liu, Y., Deelman, E., Mehta, G., Su, M. H., and Vahi, K. (2006). Automating Climate Science: Large Ensemble Simulations on the TeraGrid with the GriPhyN Virtual Data System. In *Proceedings of the 2nd IEEE International Conference on e-Science and Grid Computing (e-Science '06)*. IEEE Computer Society, Washington, DC, USA, 32–37.

[130] OASIS. (2007). Web Services Business Process Execution Language Version 2.0. `http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html`.

[131] Pandey, S., Barker, A., Gupta, K. K., and Buyya, R. (2010). Minimizing Execution Costs When Using Globally Distributed Cloud Services. In *Proceedings of the 2010 24th IEEE International Conference on Advanced Information Networking and Applications (AINA '10)*. IEEE Computer Society, Washington, DC, USA, 222–229.

[132] Papazoglou, M. P. and Georgakopoulos, A. D. (2003). Introduction: Service-Oriented Computing. *Communications of the ACM*, *46*(10), 24–28.

[133] Parejo, J. A., Fernandez, P., and Cortes, A. R. (2008). QoS-Aware Services Composition Using Tabu Search and Hybrid Genetic Algorithms. *Actas de Talleres de Ingeniera Del Software y Bases de Datos*, *2*(1), 55–66.

[134] Patel, C., Supekar, K., and Lee, Y. (2003). A QoS Oriented Framework for Adaptive Management of Web Service Based Workflows. In *Database and Expert Systems Applications*, V. Mařík, W. Retschitzegger, and O. Štěpánková, Eds. Lecture Notes in Computer Science Series, vol. 2736. Springer-Verlag, Berlin, Heidelberg, 826–835.

[135] Pernici, B., Siadat, S. H., Benbernou, S., and Ouziri, M. (2011). A Penalty-Based Approach for QoS Dissatisfaction Using Fuzzy Rules. In *Service-Oriented Computing*, G. Kappel, Z. Maamar, and H. R. Motahari-Nezhad, Eds. Lecture Notes in Computer Science Series, vol. 7084. Springer-Verlag, Berlin, Heidelberg, 574–581.

[136] Philip Chen, C. L. and Zhang, C. Y. (2014). Data-Intensive Applications, Challenges, Techniques and Technologies: A Survey on Big Data. *Information Sciences*, *275*, 314–347.

[137] Pop, C. B., Chifu, V. R., Salomie, I., Dinsoreanu, M., David, T., and Acretoaie, V. (2010). Ant-Inspired Technique for Automatic Web Service Composition and Selection. In *Proceedings of 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*. IEEE Computer Society, Washington, DC, USA, 449–455.

[138] Puppin, D., Tonellotto, N., and Laforenza, D. (2005). How to Run Scientific Applications over Web Services. In *Proceedings of the 2005 International Conference on Parallel Processing Workshops (ICPPW '05)*. IEEE Computer Society, Washington, DC, USA, 29–33.

[139] Qu, Y., Lin, C., Wang, Y., and Shan, Z. (2006). QoS-Aware Composite Service Selection in Grids. In *Proceedings of 5th International Conference on Grid and*

Cooperative Computing (GCC '06). IEEE Computer Society, Washington, DC, USA, 458–465.

[140] Rahman, R. M., Alhajj, R., and Barker, K. (2008). Replica Selection Strategies in Data Grid. *Journal of Parallel and Distributed Computing*, *68*(12), 1561–1574.

[141] Rahman, R. M., Barker, K., and Alhajj, R. (2007). A Predictive Technique for Replica Selection in Grid Environment. In *Proceedings of the 7th IEEE International Symposium on Cluster Computing and the Grid (CCGRID 2007)*. IEEE Computer Society, Washington, DC, USA, 163–170.

[142] Ramacher, R. and Monch, L. (2012). Cost-Minimizing Service Selection in the Presence of End-to-End QoS Constraints and Complex Charging Models. In *Proceedings of IEEE 9th International Conference on Services Computing (SCC)*. IEEE Computer Society, Washington, DC, USA, 154–161.

[143] Randall, M. (2004). Near Parameter Free Ant Colony Optimisation. In *Ant Colony Optimization and Swarm Intelligence*, M. Dorigo, M. Birattari, C. Blum, L. Gambardella, F. Mondada, and T. Stützle, Eds. Lecture Notes in Computer Science Series, vol. 3172. Springer-Verlag, Berlin, Heidelberg, 374–381.

[144] Ranganathan, K. and Foster, I. (2001). Identifying Dynamic Replication Strategies for a High-Performance Data Grid. In *Grid Computing - GRID 2001*, C. A. Lee, Ed. Lecture Notes in Computer Science Series, vol. 2242. Springer-Verlag, Berlin, Heidelberg, 75–86.

[145] Ravi, R. and Sinha, A. (2004). Multicommodity Facility Location. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '04)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 342–349.

[146] Rezaie, H., NematBaksh, N., and Mardukhi, F. (2010). A Multi-Objective Particle Swarm Optimization for Web Service Composition. In *Networked Digital Technologies*, F. Zavoral, J. Yaghob, P. Pichappan, and E. El-Qawasmeh, Eds. Communications in Computer and Information Science Series, vol. 88. Springer-Verlag, Berlin, Heidelberg, 112–122.

[147] Ripeanu, M. and Foster, I. (2002). A Decentralized, Adaptive Replica Location Mechanism. In *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing (HPDC '02)*. IEEE Computer Society, Washington, DC, USA, 24–32.

[148] Rojas, I., Gonzalez, J., Pomares, H., Merelo, J. J., Castillo, P. A., and Romero, G. (2002). Statistical Analysis of the Main Parameters Involved in the Design of a Genetic Algorithm. *IEEE Transactions on Systems, Man, and Cybernetics-Part. C: Applications and Reviews*, *32*(1), 31–37.

[149] Rouse, M. (2010). Cloud Computing. `http://searchcloudcomputing.techtarget.com/definition/cloud-computing`.

[150] Ruiz-Cortés, A., Martín-Díaz, O., Durán, A., and Toro, M. (2005). Improving the Automatic Procurement of Web Services Using Constraint Programming. *International Journal of Cooperative Information Systems*, *14*(4), 439–467.

[151] Savitz, E. (2012). Gartner: 10 Critical Tech Trends for the Next Five Years. http://www.forbes.com/sites/ericsavitz/2012/10/22/gartner-10-critical-tech-trends-for-the-next-five-years/.

[152] Sayal, M., Breitbart, Y., Scheuermann, P., and Vingralek, R. (1998). Selection Algorithms for Replicated Web Servers. *ACM SIGMETRICS Performance Evaluation Review*, *26*(3), 44–50.

[153] Sayal, M., Scheuermann, P., and Vingralek, R. (2003). Content Replication in Web++. In *Proceedings of the 2nd IEEE International Symposium on Network Computing and Applications (NCA '03)*. IEEE Computer Society, Washington, DC, USA, 33–40.

[154] Shen, J., Beydoun, G., Yuan, S., and Low, G. (2011). Comparison of Bio-Inspired Algorithms for Peer Selection in Services Composition. In *Proceedings of the 2011 IEEE International Conference on Services Computing (SCC)*. IEEE Computer Society, Washington, DC, USA, 250–257.

[155] Shmoys, D. B., Swamy, C., and Levi, R. (2004). Facility Location with Service Installation Costs. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '04)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1088–1097.

[156] Singh, M. N. and Huhns, M. N. (2005). *Service-Oriented Computing: Semantics, Processes, Agents*. John Wiley & Sons Ltd, West Sussex, England.

[157] Sivanandam, S. N. and Deepa, S. N. (2007). *Introduction to Genetic Algorithms*. Springer-Verlag, Berlin, Heidelberg.

[158] Song, Y., Wang, H., Li, Y., Feng, B., and Sun, Y. (2009). Multi-Tiered on Demand Resources Scheduling for VM-Based Data Center. In *Proceedings of 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID '09)*. IEEE Computer Society, Washington, DC, USA, 148–155.

[159] Srinivas, M. and Patnaik, L. M. (1994). Genetic Algorithms: A Survey. *Computer*, *27*(6), 17–26.

[160] Srinivas, N. and Deb, K. (1994). Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, *2*(3), 221–248.

[161] Stützle, T., López-Ibáñez, M., Pellegrini, P., Maur, M., Montes de Oca, M., Birattari, M., and Dorigo, M. (2012). Parameter Adaptation in Ant Colony Optimization. In *Autonomous Search*, Y. Hamadi, E. Monfroy, and F. Saubion, Eds. Springer-Verlag, Berlin, Heidelberg, 191–215.

[162] Su, S., Zhang, C., and Chen, J. (2007). An Improved Genetic Algorithm for Web Services Selection. In *Proceedings of the 7th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems*, J. Indulska and K. Raymond, Eds. Springer-Verlag, Berlin, Heidelberg, 284–295.

[163] Sun, M., Sun, J., Lu, E., and Yu, C. (2005). Ant Algorithm for File Replica Selection in Data Grid. In *Proceedings of the 1st International Conference on*

Semantics, Knowledge and Grid (SKG '05). IEEE Computer Society, Washington, DC, USA, 64–66.

[164] Sun, S. X. and Zhao, J. (2012). A Decomposition-Based Approach for Service Composition with Global QoS Guarantees. *Information Sciences*, *199*(0), 138–153.

[165] Swaminathan, G. (2012). Big Data Solutions to Remain Ahead of the Curve. `http://archive.globalservicesmedia.com/IT-Outsourcing/Infrastructure-Management/Big-Data-Solutions-to-Remain-Ahead-of-the-Curve/22/6/11990/GS1203198210600`.

[166] Taboada, H. A., Espiritu, J. F., and Coit, D. W. (2008). MOMS-GA: A Multi-Objective Multi-State Genetic Algorithm for System Reliability Optimization Design Problems. *IEEE Transactions on Reliability*, *57*(1), 182–191.

[167] Tang, M. and Ai, L. (2010). A Hybrid Genetic Algorithm for the Optimal Constrained Web Service Selection Problem in Web Service Composition. In *Proceeding of the 2010 World Congress on Computational Intelligence*. IEEE Computer Society, Washington, DC, USA, 268–275.

[168] Taylor, I. J., Deelman, E., Gannon, D. B., and Shields, M., Eds. (2006). *Workflows for e-Science: Scientific Workflows for Grids*. Springer-Verlag, Berlin, Heidelberg.

[169] van der Aalst, W. M. P. and ter Hofstede, A. H. M. (2005). YAWL: Yet Another Workflow Language. *Information Systems*, *30*(4), 245–275.

[170] Venugopal, S. and Buyya, R. (2008). An SCP-Based Heuristic Approach for Scheduling Distributed Data-Intensive Applications on Global Grids. *Journal of Parallel and Distributed Computing*, *68*(4), 471–487.

[171] Von Laszewski, G., Gawor, J., Krishnan, S., and Jackson, K. (2003). *Commodity Grid Kits - Middleware for Building Grid Computing Environments* 1 Ed. John Wiley & Sons Ltd, Somerset, NJ, USA, Chapter 25.

[172] Wada, H., Suzuki, J., Yamano, Y., and Oba, K. (2012). E^3: A Multiobjective Optimization Framework for SLA-Aware Service Composition. *IEEE Transactions on Services Computing*, *5*(3), 358–372.

[173] Wang, H., Ma, P., and Zhou, X. (2012a). A Quantitative and Qualitative Approach for NFP-Aware Web Service Composition. In *Proceedings of IEEE 9th International Conference on Services Computing (SCC)*. IEEE Computer Society, Washington, DC, USA, 202–209.

[174] Wang, L. and He, Y. (2010). A Web Service Composition Algorithm Based on Global QoS Optimizing with MOCACO. In *Proceedings of the 2011 International Conference on Informatics, Cybernetics, and Computer Engineering (ICCE 2011)*, L. Jiang, Ed. Advances in Intelligent and Soft Computing Series, vol. 111. Springer-Verlag, Berlin, Heidelberg, 79–86.

[175] Wang, L., Luo, J., Shen, J., and Dong, F. (2013a). Cost and Time Aware Ant Colony Algorithm for Data Replica in Alpha Magnetic Spectrometer Experiment. In *Proceedings of the IEEE 2nd International Congress on Big Data*. IEEE Computer Society, Washington, DC, USA, 254–261.

[176] Wang, L. and Shen, J. (2012). Towards Bio-Inspired Cost Minimization for Data-Intensive Service Provision. In *Proceedings of IEEE 1st International Conference on Services Economics*. IEEE Computer Society, Washington, DC, USA, 16–23.

[177] Wang, L. and Shen, J. (2013). Economical Data-Intensive Service Provision Supported with a Modified Genetic Algorithm. In *Proceedings of IEEE 2nd International Congress on Big Data*. IEEE Computer Society, Washington, DC, USA, 358–365.

[178] Wang, L. and Shen, J. (2014). A Critical Systematic Review of Service Concretization Based on Bio-Inspired Approaches. `http://ro.uow.edu.au/eispapers/1903`.

[179] Wang, L., Shen, J., and Beydoun, G. (2013b). Enhanced Ant Colony Algorithm for Cost-Aware Data-Intensive Service Provision. In *Proceedings of IEEE 9th World Congress on Services*. IEEE Computer Society, Washington, DC, USA, 227–234.

[180] Wang, L., Shen, J., Di, C., Li, Y., and Zhou, Q. (2013c). Towards Minimizing Cost for Composite Data Intensive Services. In *Proceeding of the IEEE 17th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. IEEE Computer Society, Washington, DC, USA, 293–298.

[181] Wang, L., Shen, J., and Yong, J. (2012b). A Survey on Bio-Inspired Algorithms for Web Service Composition. In *Proceeding of the IEEE 16th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. IEEE Computer Society, Washington, DC, USA, 569–574.

[182] Wang, R., Ma, L., and Chen, Y. (2010). The Application of Ant Colony Algorithm in Web Service Selection. In *Proceedings of International Conference on Computational Intelligence and Software Engineering*. IEEE Computer Society, Washington, DC, USA, 1–4.

[183] Wang, W., Sun, Q., Zhao, X., and Yang, F. (2012c). An Improved Particle Swarm Optimization Algorithm for QoS-Aware Web Service Selection in Service Oriented Communication. *International Journal of Computational Intelligence Systems*, *3*(01), 18–30.

[184] Wang, Y. (2009). Application of Chaos Ant Colony Algorithm in Web Service Composition Based on QoS. In *Proceedings of International Forum on Information Technology and Applications*. IEEE Computer Society, Washington, DC, USA, 225–227.

[185] Wang, Z. J., Liu, Z. Z., Zhou, X. F., and Lou, Y. S. (2011). An Approach for Composite Web Service Selection Based on DGQoS. *The International Journal of Advanced Manufacturing Technology*, *56*(9-12), 1167–1179.

[186] Watson, W. A., Bird, I., Chen, J., Hess, B., Kowalski, A., and Chen, Y. (2002). A Web Services Data Analysis Grid. *Concurrency and Computation: Practice and Experience*, *14*, 1303–1311.

[187] Wilkes, J. (2008). Utility Functions, Prices, and Negotiation. Tech. Rep. HPL-2008-81, HP Labs.

[188] Winter, M. (2009). Data Center Consolidation: A Step Towards Infrastructure Clouds. In *Cloud Computing*, M. Jaatun, G. Zhao, and C. Rong, Eds. Lecture Notes in Computer Science Series, vol. 5931. Springer-Verlag, Berlin, Heidelberg, 190–199.

[189] Woolf, A., Haines, K., and Liu, C. (2003). A Web Service Model for Climate Data Access on the Grid. *International Journal of High Performance Computing Applications*, *17*(3), 281–295.

[190] Wu, Q. and Zhu, Q. (2013). Transactional and QoS-Aware Dynamic Service Composition Based on Ant Colony Optimization. *Future Generation Computer Systems*, *29*(5), 1112–1119.

[191] Wyglinski, A. M., Nekovee, M., and Hou, Y. T. (2010). *Cognitive Radio Communications and Networks*. Academic Press Inc, United Kingdom.

[192] Xia, Y., Chen, J., and Meng, X. (2008). On the Dynamic Ant Colony Algorithm Optimization Based on Multi-Pheromones. In *Proceedings of 7th IEEE/ACIS International Conference on Computer and Information Science (ICIS '08)*. IEEE Computer Society, Washington, DC, USA, 630–635.

[193] Xiao, J., Michalewicz, Z., and Trojanowski, K. (1997). Adaptive Evolutionary Planner/Navigator for Mobile Robots. *IEEE Transactions on Evolutionary Computation*, *1*(1), 18–28.

[194] Yan, J., Kowalczyk, R., Lin, J., Chhetri, M. B., Goh, S. K., and Zhang, J. (2007). Autonomous Service Level Agreement Negotiation for Service Composition Provision. *Future Generation Computer Systems*, *23*(6), 748–759.

[195] Yang, M., Li, Y., Li, S., and Li, P. (2008). ANN-Based Fuzzy Reasoning to Determine the Importance of Technical Requirements in QFD. In *Proceedings of 4th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM '08)*. IEEE Computer Society, Washington, DC, USA, 1–5.

[196] Yang, Z., Shang, C., Liu, Q., and Zhao, C. (2010). A Dynamic Web Services Composition Algorithm Based on the Combination of Ant Colony Algorithm and Genetic Algorithm. *Computational Information Systems*, *6*(8), 2617–2622.

[197] Yao, Y. and Chen, H. (2009). QoS-Aware Service Composition Using NSGA-II1. In *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*. ACM, New York, NY, USA, 358–363.

[198] Yoon, K. P. and Land, H. C. (1995). *Multiple Attribute Decision Making: An Introduction*. Quantitative Applications in the Social Sciences Series, vol. 104. SAGE Publications, Inc., CA, Thousand Oaks.

[199] Yu, H. and Liu, M. (2011). A QoS-Aware Web Services Selection Model Using AND/OR Graph. In *Advanced Data Mining and Applications*, J. Tang, I. King, L. Chen, and J. Wang, Eds. Lecture Notes in Computer Science Series, vol. 7120. Springer-Verlag, Berlin, Heidelberg, 124–137.

[200] Yu, T., Zhang, Y., and Lin, K. J. (2007). Efficient Algorithms for Web Services Selection with End-to-End QoS Constraints. *ACM Transactions on the Web*, *1*(1), 1–26.

[201] Zegura, E. W., Ammar, M. H., Fei, Z., and Bhattacharjee, S. (2000). Application-Layer Anycasting: A Server Selection Architecture and Use in a Replicated Web Service. *IEEE/ACM Transactions on Networking*, *8*(4), 455–466.

[202] Zemni, M. A., Benbernou, S., and Carro, M. (2010). A Soft Constraint-Based Approach to QoS-Aware Service Selection. In *Service-Oriented Computing*, P. P. Maglio, M. Weske, J. Yang, and M. Fantinato, Eds. Lecture Notes in Computer Science Series, vol. 6470. Springer-Verlag, Berlin, Heidelberg, 596–602.

[203] Zeng, L., Benatallah, B., Ngu, A. H. H., Dumas, M., Kalagnanam, J., and Chang, H. (2004). QoS-Aware Middleware for Web Services Composition. *IEEE Transactions on Software Engineering*, *30*(5), 311–327.

[204] Zhang, C. and Ma, Y. (2009). Dynamic Genetic Algorithm for Search in Web Service Compositions Based on Global QoS Evaluations. In *Proceedings of International Conference on Scalable Computing and Communications; 8th International Conference on Embedded Computing (SCALCOM-EMBEDDEDCOM '09)*. IEEE Computer Society, Washington, DC, USA, 644–649.

[205] Zhang, C., Su, S., and Chen, J. (2006). A Novel Genetic Algorithm For QoS-Aware Web Services Selection. In *Data Engineering Issues in E-Commerce and Services*, J. Lee, J. Shim, S. Lee, C. Bussler, and S. Shim, Eds. Lecture Notes in Computer Science Series, vol. 4055. Springer-Verlag, Berlin, Heidelberg, 224–235.

[206] Zhang, W., Chang, C. K., Feng, T., and Jiang, H. Y. (2010). QoS-Based Dynamic Web Service Composition with Ant Colony Optimization. In *Proceedings of the 34th Annual IEEE International Computer Software and Applications Conference (COMPSAC '10)*. IEEE Computer Society, Washington, DC, USA, 493–502.

[207] Zhao, X., Song, B., Huang, P., Wen, Z., Weng, J., and Fan, Y. (2012). An Improved Discrete Immune Optimization Algorithm Based on PSO for QoS-Driven Web Service Composition. *Applied Soft Computing*, *12*(8), 2208–2216.

[208] Zheng, X., Luo, J., and Song, A. (2007). Ant Colony System Based Algorithm for QoS-Aware Web Service Selection. In *Proceedings of 4th International Conference on Grid Service Engineering and Management (GSEM)*. Springer-Verlag, Berlin, Heidelberg, 39–50.

[209] Zitzler, E. and Thiele, L. (1999). Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, *3*(4), 257–271.

[210] Zulkernine, F. H. and Martin, P. (2011). An Adaptive and Intelligent SLA Negotiation System for Web Services. *IEEE Transactions on Services Computing*, *4*(1), 31–43.