Swarm Intelligence for Cloud Computing
Swarm Intelligence for Cloud Computing

Edited by
Indrajit Pan
Department of Information Technology, RCC Institute of Information Technology, India

Mohamed Abd Elaziz
Department of Mathematics, Zagazig University, Egypt

Siddhartha Bhattacharyya
Department of Computer Science and Engineering, CHRIST (Deemed to be University), India
Indrajit Pan would like to dedicate this book to his friends of Friend-o-Mania group, Manash Sinharoy, Avijit Das, Palash Kumar Ghosh, David Thomas and Ratan Biswas, for their unconditional support as and when needed as an extended family members and part of an emphatic support system.

Mohamed Abd Elaziz would like to dedicate this book to his family and Rehab Ali for supporting him and giving more motivation in his life.

Siddhartha Bhattacharyya would like to dedicate this book to his father Late Ajit Kumar Bhattacharyya, his mother Late Hashi Bhattacharyya, his beloved wife Rashni and his research scholars Sourav, Sandip, Pankaj, Hrishikesh, Debanjan, Koyel, Alokananda, and Tulika.
Contents

Preface ix
Editors xiii
Contributors xvii

1 A Survey of Swarm Intelligence for Task Scheduling in Cloud Computing
   Ahmed A. Ewees, Neggaz Nabil, Rehab A. Ibrahim, and Rania M. ELEraky 1

2 Techniques for Resource Sharing in Cloud Computing Platform
   B.L. Radhakrishnan, S. Sudhakar, R.V. Belfin, P. Karthikeyan, E. Kirubakuran, and K. Martin Sagayam 39

3 Swarm Intelligent Techniques for Cloud Service Provider Selection in a Multi-cloud Environment
   Amany M. Mohamed and Hisham M. Abdelsalam 73

4 Reliable Data Auditing and ACO-Based Resource Scheduling for Cloud Services
   Soumitra Sasmal and Indrajit Pan 107

5 TS-GWO: IoT Tasks Scheduling in Cloud Computing Using Grey Wolf Optimizer
   Laith Abualigah, Mohammad Shehab, Mohammad Alshinwan, Hamzeh Alabool, Hayfa Y. Abuaddous, Ahmad M. Khasawneh, and Mofleh Al Diabat 127

6 Fact-Checking: Application-Awareness in Data Centre Resource Management
   Aaqif Afzaal Abbasi, Almas Abbasi, Mohammed A. A. Al-qaness, Ammar Hawbani, Ahmed A. Ewees, and Yousif A. Alhaj 153

7 Bio-inspired Optimization Algorithms for Multi-objective Task Scheduling in Cloud Computing Systems
   A. S. Ajeena Beegom and M. S. Rajasree 165

Index 187
Preface

Cloud computing is a newly developed platform for providing IT services. Like the subscribers of traditional utility services, subscribers of cloud computing pay only for the services and resources that they use. Major IT companies, such as Amazon, Google, and Microsoft, are currently providing cloud services and resources. These companies consider cloud computing as the future of IT operations. Subscribers do not have to build or maintain their own IT infrastructures. Cloud computing allows subscribers to rapidly and inexpensively re-provision IT resources for various needs.

Due to flexibility, security, availability, scalability and affordability, cloud computing has begun to attract the attention of IT stakeholders. The features provided by cloud computing are extremely beneficial for small organizations that lack funding to maintain their own infrastructure. Nowadays, many organizations are leaning toward this new technology for various IT-related infrastructural and application needs. So far, cloud computing has been explored up to an extent of its infrastructural and security-related know how; However, a wide scope remains for the research.

Primitive technological development needs to be augmented with high-end intelligent approaches so that the cloud setup becomes more robust and efficient. People are still concerned about its sustainability and stability of infrastructure and security of services and data over the cloud. Since it is run over a distributed platform, many complexities and constraints have to be considered at practical level implementation.

This book will present some state-of-the-art research involving swarm and hybrid swarm intelligence for cloud security, reliability, and infrastructural stability. Security-related task will focus on public auditing of data and services. Infrastructural development will be focused on various aspects like task scheduling, virtual machine allocation, load balancing and optimization, deadline handling, power-aware profiling, fault resilience, cost effective design, and energy efficiency.

In the last few decades, a novel branch of intelligent computation algorithm has been inspired by a swarm intelligence theory that imitates the behavior of animals. These algorithms are successfully applied to solve many kinds of simple and complex problems in several fields such as optimization problems, pattern recognition, image processing, features section, and task scheduling in cloud and parallel computing. Cloud computing has become the preferred environment for several companies. Moreover, it has helped several companies to overcome many server issues by utilizing their characteristics such as
reliability, flexibility, high scalability, and security. Therefore, many intelligent computation algorithms are employed to improve this environment. In Chapter 1, an overview of swarm intelligence for solving the scheduling problems of tasks in cloud computing is presented, including particle swarm optimization, cat optimization algorithm, artificial bee colony, lion optimization algorithm, whale optimization algorithm, Bat algorithm, gray wolf optimizer, cuckoo search algorithm, hybrid swarm algorithms, and multi-objective swarm optimization. All these algorithms are described and presented with their achievements in solving task scheduling issues in cloud computing.

Cloud Computing has revolutionized the way Information Technology (IT) operates. Resource sharing in the cloud enables multiple users or providers to share resources, which improves the utilization of their resources. In the cloud, aggregated resources create a resource pool. The resource pools handle the dynamic demand of IT. Resource sharing enables cloud providers and customers to reduce their capital expenditure, operational expenditure, and total cost of ownership. Multiple cloud service providers connect to form a federation to share their resources among themselves to meet the dynamic demand of their customers. Chapter 2 discusses resource sharing in intercloud, social cloud, Mobile Cloud, Manufacturing Cloud, Vehicular cloud, and Green cloud. In each cloud, various resource sharing approaches, strength and challenges are discussed.

In the cloud computing environment, comparing providers and selecting the best one among them is one of the most important issue/decisions to customers/clients. The difficulty of provider selection decision increases with increasing number of cloud providers and its offering. However, from the customers’ perspective, this enables them to take the advantage of getting its requirement from multiple cloud providers and reducing the dependency on single cloud provider. Multi-cloud has been the focus of recent research—a strategy that assists customers to avoid vendor lock-in problem. Chapter 3 presents a general model for selecting providers in a multi-cloud environment, considering any number of IaaS services based on two evaluation criteria: cost and performance. The problem was formulated as integer programming and it was approved to be a NP hard problem. Consequently, to solve it, three metaheuristic algorithms have been used: Genetic Algorithm (GA), Harmony Search (HS) and Particle Swarm Optimization (PSO). In order to test and compare the performance of the proposed algorithms, a case study was generated.

Twenty-first century has observed prolific progress of cloud computing paradigm. Many individuals and organizations are now heavily dependent on cloud services to run their operations and businesses. Many aspects of cloud computing are still in development under intense observation of research community. The underlying requirement of cloud computing is a network that empowers the connection among stakeholders and resources. Security of resources, specifically data, is a key concern along with right deployment of resources to deliver optimal performance. Chapter 4 proposes two efficient
approaches together toward reliable service execution of cloud. One part provides an ant colony optimization approach for dynamic resource scheduling and the other part ensures secure auditing of data during dynamic resource allocation. Experimental analysis shows performance improvement in terms of request processing time, auditing time, successful resource allocation and average utilization of servers with different volumes of clients among all available servers.

The most critical action in the Internet of Things (IoT), particularly in the cloud computing (CC) ecosystem, is job/task scheduling. The rule of the task scheduling (TS) means that how to arrange/schedule the jobs over the given virtual machines (VMs) by decreasing the Makespan measured (Mm) values and the cost need. Various scholars propose several scheduling algorithms for solving this problem of scheduling the tasks in cloud computing ecosystems. In Chapter 5, a task scheduling method is proposed using a multi-objective design model and the Gray Wolf Optimizer (GWO), called TS-GWO. First, the multi-objective criterion measures the fitness function by determining the cost value of the CPU ratio and memory size (Me). The fitness function is determined by calculating the Makespan value and demand value. The proposed method (TS-GWO) can expertly arrange the given jobs to the offered VMs while keeping the smallest Makespan value and cost value. Conclusively, the performance of the proposed method (TS-GWO) is investigated and matched with the basic GWO for the testing measures: Makespan ration and cost value.

Due to the recent developments witnessed in the world of computing, IT trends are continuously changing. This includes changes in the way information is gathered, managed and processed. In this regard, cloud computing is considered to be the state-of-the-art technology to host and manage stored information on a network of shared resources. With the arrival of SDN technology, the intelligence from hardware devices shifted from vendors to application awareness concepts. This also led to the ease of development of network management functions and features. In Chapter 6, the authors give a quick look at the various aspects of application awareness in data center resource management.

Biologically inspired optimization techniques are applied in various fields for finding a quick optimal or near optimal solution to computationally hard problems. Study on the application of these techniques for task scheduling in cloud computing systems is worth due to the dynamic and unpredictable nature of task arrival. The quality of a task schedule in cloud computing systems depends on the response time that a user of the cloud system experiences and the cost of the service at both cloud service provider’s end and at the user side. Optimizing such conflicting factors in a dynamic environment is challenging since it is a multi-objective optimization problem. Many scientific and large-scale applications are characterized by a huge number of homogeneous tasks that can be executed in parallel. Chapter 7 considers the scheduling of such tasks. Three swarm intelligence algorithms namely Particle
Swarm Optimization (PSO) technique, Artificial Bee Colony (ABC) algorithm and Ant Colony Optimization (ACO) algorithm and an evolutionary computation based algorithm namely Genetic Algorithm (GA) are applied to solve the problem. Experiments are done on different types of data sets and a comparative study on the results is performed to determine the suitability of these algorithms in the cloud environment.

This volume explores different research perspectives based on swarm intelligence or hybrid swarm intelligence based techniques to deliver robust and stable mechanism for cloud infrastructural support and cloud security.

India, Egypt

Indrajit Pan
Mohamed Abd Elaziz
Siddhartha Bhattacharyya
Editors

Dr. Indrajit Pan has done his Bachelor of Engineering (BE) in Computer Science and Engineering with Honors from The University of Burdwan in 2005 and completed Master of Technology (M.Tech) in Information Technology from Bengal Engineering and Science University, Shibpur in 2009. He was the recipient of University Medal for his academic performance during Masters. He obtained Ph.D. in Engineering from Department of Information Technology, Indian Institute of Engineering Science and Technology, Shibpur in 2015. His research title was Design and Analysis of Droplet Routing Algorithms in Digital Microfluidic Biochip.

His present research interest includes social network analysis and cloud computing. Indrajit is now serving as Associate Professor of Information Technology Department at RCC Institute of Information Technology, Kolkata with an experience of 14 years in teaching.

Indrajit takes active interest in organizing international conferences, symposiums, seminars and faculty development programs at different capacities. He was the member of different such organizing committees and also served in the capacity of General chair, organizing secretary and program chair at multiple occasions. He has also served as technical program committee members in many international conferences and reviewer in some IEEE transactions and Elsevier journals.

He has around forty research publications in different International journals, edited books and conference proceedings. Three research scholars are enrolled and working under his supervision for their doctoral dissertation.

Indrajit has coauthored five published research volumes with prestigious publishers like John Wiley, CRC Press, De Gruyter and IGI Global. In addition to these he has coauthored four international journal proceedings with IEEE, USA and two International symposium proceedings with Springer Nature. He served as guest editor in International Journal of Hybrid Intelligence for special issue on Hybrid computational intelligence in big data analytics and cloud computing.

Indrajit is the Senior member of IEEE, USA and Member of ACM, USA.
Mohamed Abd Elaziz was received the B.S. and M.S. degrees in Computer Science from the Zagazig University, in 2008 and 2011, respectively. He received Ph.D. degree in mathematics and computer science from Zagazig University, Egypt in 2014. From 2008 to 2011, he was Assistant lecturer in Department of Computer Science. He is currently an Associate Professor with the Mathematical Department, Zagazig University. He is the author of more than 100 articles. His research interests include machine learning, signal processing, image processing, cloud computing, Evolutionary algorithms. Mohamed has coauthored two published books.

Siddhartha Bhattacharyya did his Bachelor’s in Physics, Bachelor’s in Optics and Optoelectronics and Masters in Optics and Optoelectronics from University of Calcutta, India in 1995, 1998 and 2000 respectively. He completed PhD in Computer Science and Engineering from Jadavpur University, India in 2008. He is the recipient of the University Gold Medal from the University of Calcutta for his Masters. He is the recipient of several coveted awards including the Distinguished HoD Award and Distinguished Professor Award conferred by Computer Society of India, Mumbai Chapter, India in 2017, the Honorary Doctorate Award (D. Litt.) from The University of South America and the South East Asian Regional Computing Confederation (SEARCC) International Digital Award ICT Educator of the Year in 2017. He has been appointed as the ACM Distinguished Speaker for the tenure 2018–2020.

He is currently serving as a Professor in the Department of Computer Science and Engineering of Christ University, Bangalore. He served as the Principal of RCC Institute of Information Technology, Kolkata, India during 2017–2019. He has also served as a Senior Research Scientist in the Faculty of Electrical Engineering and Computer Science of VSB Technical University of Ostrava, Czech Republic (2018–2019). Prior to this, he was the Professor of Information Technology of RCC Institute of Information Technology, Kolkata, India. He served as the Head of the Department from March, 2014 to December, 2016. Prior to this, he was an Associate Professor of Information Technology of RCC Institute of Information Technology, Kolkata, India from 2011 to 2014. Before that, he served as an Assistant Professor in Computer Science and Information Technology of University Institute of Technology,
The University of Burdwan, India from 2005 to 2011. He was a Lecturer in Information Technology of Kalyani Government Engineering College, India during 2001–2005. He is a co-author of 5 books and the co-editor of 54 books and has more than 280 research publications in *International Journals and Conference Proceedings* to his credit. He has got two PCTs to his credit. He has been the member of the organizing and technical program committees of several National and International Conferences. He is the founding Chair of ICCICN 2014, ICRCICN (2015–2018; 2020), ISSIP (2017, 2018; 2020) (Kolkata; Bangalore, India; Zagreb, Croatia), DoSIER (2019; 2020) (Kolkata; Santiniketan, India). He was the General Chair of several international conferences like WCNSSP 2016 (Chiang Mai, Thailand), ICACC (2017, 2019) (Sikkim, India) and ICICC (2018; 2020) (New Delhi, India) and ICICC 2019 (Ostrava, Czech Republic).

He is the Associate Editor of several reputed journals including Applied Soft Computing, IEEE Access, Evolutionary Intelligence and IET Quantum Communications. He is the editor of *International Journal of Pattern Recognition Research* and the founding Editor in Chief of *International Journal of Hybrid Intelligence, Inderscience*. He has guest edited several issues with several International Journals. He is serving as the Series Editor of IGI Global Book Series Advances in Information Quality and Management (AIQM), De Gruyter Book Series Frontiers in Computational Intelligence (FCI), CRC Press Book Series on Computational Intelligence and Applications, CRC Press Book Series on Quantum Machine Intelligence, Wiley Book Series Intelligent Signal and Data Processing, Elsevier Book Series Hybrid Computational Intelligence for Pattern Analysis and Understanding and Springer Tracts on Human Centered Computing.

His research interests include hybrid intelligence, pattern recognition, multimedia data processing, social networks and quantum computing.

Dr. Bhattacharyya is a life fellow of Optical Society of India (OSI), India, life fellow of International Society of Research and Development (ISRD), UK, a fellow of Institute of Electronics and Telecommunication Engineers (IETE), India and a fellow of Institution of Engineers (IEI), India. He is also a senior member of Institute of Electrical and Electronics Engineers (IEEE), USA, International Institute of Engineering and Technology (IETI), Hong Kong and Association for Computing Machinery (ACM), USA. He is a life member of Cryptology Research Society of India (CRSI), Computer Society of India (CSI), Indian Society for Technical Education (ISTE), Indian Unit for Pattern Recognition and Artificial Intelligence (IUPRAI), Center for Education Growth and Research (CEGR), Integrated Chambers of Commerce and Industry (ICCI), and Association of Leaders and Industries (ALI). He is a member of Institution of Engineering and Technology (IET), UK, International Rough Set Society, International Association for Engineers (IAENG), Hong Kong, Computer Science Teachers Association (CSTA), USA, International Association of Academicians, Scholars, Scientists and Engineers (IAASSE), USA, Institute of Doctors
Engineers and Scientists (IDES), India, The International Society of Service Innovation Professionals (ISSIP) and The Society of Digital Information and Wireless Communications (SDIWC). He is also a certified Chartered Engineer of Institution of Engineers (IEI), India. He is on the Board of Directors of International Institute of Engineering and Technology (IETI), Hong Kong.
Contributors

Aaqif Afzaal Abbasi
Department of Software Engineering
Foundation University
Islamabad, Pakistan

Almas Abbasi
Department of Computer Science
International Islamic University
Islamabad, Pakistan

Hisham M. Abdelsalam
Faculty of Computers and Information
Cairo University
Cairo, Egypt

Hayfa Y. Abuaddous
Faculty of Computer Sciences and Informatics
Amman Arab University
Amman, Jordan

Laith Abualigah
Faculty of Computer Sciences and Informatics
Amman Arab University
Amman, Jordan

Hamzeh Alabool
Department of Information Technology
Saudi Electronic University
Riyadh, Saudi Arabia

Yousif A. Alhaj
School of Computer Science and Technology
Wuhan University of Technology
Wuhan, China

Mohammed A. A. Al-qaness
School of Computer Science
Wuhan University
Wuhan, China

Mohammad Alshinwan
Faculty of Computer Sciences and Informatics
Amman Arab University
Amman, Jordan

A. S. Ajeena Beegom
Department of Computer Science and Engineering
College of Engineering Trivandrum
Trivandrum, India

R. V. Belfin
Department of Computer Science and Engineering
Karunya Institute of Technology and Sciences
Coimbatore, India

Mofleh Al Diabat
Department of Computer Science
Al Albayt University
Mafraq, Jordan
**Contributors**

**Rania M. ELeraky**  
Computer Curriculum and Methods Department  
University of Bisha  
Bisha, Saudi Arabia  
and  
Faculty of Specific Education  
Damietta University  
Damietta, Egypt

**Ahmed A. Ewees**  
Computer Department  
Damietta University  
Damietta, Egypt

**Ammar Hawbani**  
School of Computer Science and Technology  
University of Science and Technology of China  
Hefei, China

**Rehab A. Ibrahim**  
School of Computer Science and Technology  
Huazhong University of Science and Technology  
Wuhan, China

**E. Kirubakaran**  
Department of Computer Science and Engineering  
Karunya Institute of Technology and Sciences  
Coimbatore, India

**K. Martin Sagayam**  
Department of Electronics and Communication Engineering  
Karunya Institute of Technology and Sciences  
Coimbatore, India

**Aman M. Mohamed**  
Faculty of Computers and Information  
Cairo University  
Cairo, Egypt

**Neggaz Nabil**  
Faculté des Mathe’matiques et Informatique, Département d’informatique  
Université des Sciences et de la Technologie d’Oran Mohamed Boudiaf, Laboratoire Signal Image PArôle (SIMPA)  
Oran, Algérie

**Indrajit Pan**  
Department of Information Technology  
RCC Institute of Information Technology  
Kolkata, India

**B. L. Radhakrishnan**  
Department of Computer Science and Engineering  
Karunya Institute of Technology and Sciences  
Coimbatore, India
Contributors

M. S. Rajasree
Department of Computer Science
Government Engineering College
Trivandrum, India

Soumitra Sasmal
Department of Information Technology
Techno Main, Salt Lake
Kolkata, India

Mohammad Shehab
Computer Science Department
Aqaba University of Technology
Aqaba, Jordan

S. Sudhakar
Department of Computer Science and Engineering
Karunya Institute of Technology and Sciences
Coimbatore, India
A Survey of Swarm Intelligence for Task Scheduling in Cloud Computing

Ahmed A. Ewees  
*Damietta University*

Neggaz Nabil  
*Université des Sciences et de la Technologie d’Oran Mohamed Boudiaf, Laboratoire Signal Image PArole (SIMPA)*

Rehab A. Ibrahim  
*Huazhong University of Science and Technology*

Rania M. ELEraky  
*University of Bisha. Damietta University*

CONTENTS

1.1 Introduction .......................................................... 2  
1.2 Task Scheduling Problem Formulation ............................ 4  
1.3 The Swarm Intelligence Application for Cloud Computing ...... 5  
1.3.1 Particle Swarm Optimization .................................. 6  
1.3.2 Artificial Bee Colony Algorithm .............................. 8  
1.3.3 Lion Optimization Algorithm ................................. 10  
1.3.4 Whale Optimization Algorithm ............................... 12  
1.3.5 Gray Wolf Optimization .................................... 13  
1.3.6 Bat Algorithm ................................................. 15  
1.3.7 Cat Swarm Optimization ..................................... 18  
1.3.8 Cuckoo Search Algorithm ................................. 20  
1.3.9 Hybrid Swarm Algorithm ................................. 23  
1.3.10 Multi-Objective Swarm Optimization ...................... 24  
1.4 Conclusion .......................................................... 26  
Bibliography .................................................................. 27
1.1 Introduction

Cloud computing (CC) has become the preferred environment for large-scale applications which need diverse platforms. It is an evolution of grid, cluster, and distributed systems [12]. It has many benefits, such as access to resources on demand; low cost due to shared resources; large storage; application of service level agreement, and use of virtual machines (VMs) to increase the resources’ utilization where applications, systems, and infrastructure are completely provided as a service; and customers can access these systems anytime from anywhere [12]. Figure 1.1 shows the main idea of CC [71].

The benefits of CC, in the recent years, have helped several companies to overcome many server-related issues by utilizing their characteristics, such as reliability, flexibility, high scalability, multi-tenancy, security, data flow, quality of service (QoS), and providing better maintenance and support [12,59]. The main goal of CC is to decrease the computation cost for consumers who are using these services [12]. QoS requires companies to provide the desired performance to their customers such as sending jobs to appropriate resources, scheduling, and producing the outputs. All these points should be efficiently performed within a short time with satisfying results [46].

In addition, there are several challenges and issues that should be considered in providing the services to the customers in CC systems, for example, task scheduling, load balancing, security issues, VMs placement, energy consumption, and storage space. However, job or task scheduling is still considered as an essential phase in CC since it works to minimize the cost toward resource utilization.

Job scheduling in CC is one of the well-known optimization problems and plays an essential role in improving reliable and flexible platforms [75]. Job
scheduling works to map the resources to jobs. The optimal scheduling method should guarantee a QoS and good performance of the CC by taking into account some criteria, such as computation time and cost [86]. It should maximize the utilization of VMs while minimizing their computation time [75]. Figure 1.2 illustrates a sample scenario of task scheduling problem in CC [59].

Jobs can be dependent or independent. If they have precedence orders, their scheduling process is called dependent or workflow scheduling, otherwise it is called independent scheduling [46]. The diversity of CC’s resources and the varying types of jobs lead to make the scheduling method an NP-hard problem [86]. VM should process the jobs as early as possible, but because of the huge numbers of jobs and VMs in CC, it may not schedule the jobs to resources optimally. Therefore, efficient job-scheduling methods are required to handle this issue [29]. There is no optimal method to solve all kinds of CC’s issues. Exhaustive search methods are not efficient as they consume high costs in generating schedules. In contrast metaheuristics methods handle these issues by producing near-optimal solutions in a sensible time. However, there are many techniques used to handle these problems, but the metaheuristics optimization techniques are still an effective method to deal with this challenge [46]. Swarm intelligence is a type of metaheuristics optimization method that simulates the natural behavior of insects, animals, or birds in swarming, preying, or flocking such as Bat algorithm [98], particle swarm optimization (PSO) [25], whale optimization algorithm (WOA) [65], gray wolf optimization (GWO) [66], and artificial bee colony (ABC) [48].

![Figure 1.2](image-url)

**FIGURE 1.2**
A sample scenario of a task scheduling problem in CC [59].
These methods are applied successfully in several fields because of their effectiveness and efficiency in dealing with complex and large problems such as image processing [26,33,34,40], features selection [28,31], and prediction problems [4–6,30,81].

In this chapter, we introduce a review of swarm intelligence applied for CC specially in tasks/jobs scheduling based on metaheuristic methods, namely PSO, lion optimization algorithm (LOA), ABC, GWO, Bat algorithm (BAT), cat swarm optimization (CSO), WOA, cuckoo search algorithm (CSA), hybrid algorithms, and multi-objective algorithms based on these methods. We reviewed more than 70 studies published from 2010 to 2019 in this field. Figure 1.3 summarizes these methods and their distribution over the years. From this figure, we can notice that the hybrid and multi-objective algorithms are the most popular methods in solving CC issues and the efforts to use these methods were increased in 2017 and 2018. Therefore, we predict that the use of these methods will also increase in 2019.

The rest of the sections of this chapter are arranged as follows. Section 1.2 shows the problem definition of task/job scheduling in CC. Section 1.3 presents a review of swarm intelligence methods as well as their hybridization and multi-objective versions to solve task scheduling issues in CC. Section 1.4 concludes the paper.

1.2 Task Scheduling Problem Formulation

In this section, the overview of a task scheduling problem in CC is presented. A task scheduling issue can occur if a program has N tasks, $T = t_1, t_2, \ldots, t_N$, each of which needs to be assigned to a VM to be processed; and there are VMs
in the environment (VMs = VM₁, VM₂, ..., VMₘ; \( M \) is the total number of VMs); whereas, this assignment requires times, called computation time, to be completed. Therefore, each task has an arrival time and firm end time. Due to heterogeneity of VMs, each VM has different cost of usage, different computation speed, and limited capacity; therefore, the cost and computation time of tasks are different. Equations 1.1 and 1.2 represent the matrix of the computation time (\( ct \)) of task (\( t_n \)) and its cost (\( \text{cost} \)), respectively, as follows [21]:

\[
CT = (ct_{nm})(N \times M) \tag{1.1}
\]

where, \( ct_{nm} \) denotes the computation time of task (\( t_n \)) on VMₘ machine.

\[
\text{Cost} = (\text{cost}_{nm})(N \times M) \tag{1.2}
\]

where, \( \text{cost}_{nm} \) is the cost usage of task (\( t_n \)) on VMₘ machine.

The aim of task scheduling is to allocate the \( n \) tasks to the appropriate \( m \) VMs to minimize the makespan of the task as well as balance the loads on CC machines. Thus, Equations 1.3 and 1.4 are used to calculate the total computation time and costs for \( n \) tasks, respectively [21]:

\[
CT_{\text{total}} = \Sigma_{n=1}^{N} CT_{nm} \tag{1.3}
\]

\[
\text{Cost}_{\text{total}} = \Sigma_{n=1}^{N} \text{Cost}_{nm} \tag{1.4}
\]

In addition, the task scheduling method should consider the load balance to effectively allocate tasks. Equation 1.4 calculates the load balance \( \text{Load}_{\text{bal}} \) of a VM.

\[
\text{Load}_{\text{bal}} = \Sigma_{n=1}^{M} |b_m - \bar{b}| \tag{1.5}
\]

where \( b_m \) indicates the load of VMₘ and \( \bar{b} \) represents the mean load of all VMs. Consequently, the task scheduling problem can be described as follows in Equation 1.6:

\[
\text{Minimize}(CT_{\text{total}}, \text{Cost}_{\text{total}}, \text{Load}_{\text{bal}}) \tag{1.6}
\]

### 1.3 The Swarm Intelligence Application for Cloud Computing

Swarm intelligence methods vary from simple to complex. Such methods are applied to obtain a best solution in a small computation time. Therefore, various swarm intelligence methods for task scheduling in CC can be used. This section illustrates an overview of the most applied swarm intelligence methods for scheduling tasks in the cloud, such as PSO, ABC algorithm, LOA, WOA, CSA, GWO, cat optimization algorithm, Bat algorithm, hybrid swarm algorithms, and multi-objective swarm optimization.
1.3.1 Particle Swarm Optimization

The PSO was proposed by the authors of [25]. It was designed to simulate the communication behavior of the bird’s group in nature to share their knowledge in migrating, flocking, and hunting. The bird group is called swarm and each member is called particle. Therefore, a swarm consists of a set of particles which denote a solution. The particle’s position is changed based on its own experience and neighbors. In the mathematical model, a swarm begins by generating random particles, each of which has a position and a velocity in the solutions dimension.

These values are updated in each loop to find the best position by applying a fitness function to maintain the solution and try to move it toward the optimal value of the given problem. This sequence is repeated until meeting the desired solution. In the job scheduling issue, the particles are symbolized as \((x \times y)\), where \(y\) indicates the tasks’ numbers and \(x\) denotes the resources’ number to perform the jobs scheduling issue. Figure 1.4 shows the flowchart of PSO [91].

PSO is widely used to solve jobs scheduling problem in CC. In Ref. [43], the authors used a modified PSO for solving the length of average scheduling and successful execution ratio in a cloud. Different numbers of tasks were used, namely 100, 300, 500, 700, and 900. The experiments were performed using CloudSim platform and they showed better results than other methods. PSO was also applied in task scheduling in CC [57]. The PSO was improved by adaptive weights to change the weight with increasing the iterations’ number and produce random weights in the next stage to avoid PSO from getting trapped in local optima. The CloudSim platform was used. The authors used different numbers of tasks, namely 100, 200, and 500 as well as 10 VMs.

In the same trend, PSO was improved in [97]; however, only 6 VMs and 15 tasks were tested using CloudSim platform. In addition, the authors of [21] reported that they effectively applied PSO in job scheduling to obtain better load balancing and minimize the ratio of deadline missing. The experiment used 60 VMs and 250, 300, 350, 400, and 450 tasks. In Ref. [13] the PSO was used to reduce the makespan and cost simultaneously. It used 10 VMs and 99 tasks, but the experiment environment was not mentioned. In Ref. [7] PSO improved based on a ranking strategy and multi-objective concept to schedule tasks to VMs in order to maximize the throughput of the system and minimize the waiting time. The experiment used 500–2,000 VMs and 500–4,000 tasks. The authors reported that the throughput was improved up to 40% and the waiting time was reduced to 30%.

Another effort in solving job scheduling issue by PSO is presented in [76]. This study suggested transferring only overloaded tasks from VM to decrease the downtime in the migration process. The experiment was performed on the CloudSim platform and 5 VMs and 10 cloudlets (as overloaded VM) were applied. The results showed that the study method could reduce the task scheduling time than traditional methods and increased the QoS by
cloud clients. In [19], the PSO was compared with Naïve algorithm to schedule the tasks in CC in smallest computation. The experiment was performed on the CloudSim platform and 5 VMs and 100 cloudlets were applied. The results showed that the PSO had the smallest computation time, whereas the naïve algorithm was computationally expensive when applying on large datasets.
1.3.2 Artificial Bee Colony Algorithm

The ABC algorithm is a type of swarm intelligence method; it mimics the foraging behavior of the colony of real honey bees. ABC algorithm was proposed in 2005 by Karaboga [48]. ABC algorithm consists of three types of bees’ groups: (1) the worker bees group which search for food's sources near the recently visited food sources; (2) the onlooker bees group which work to find a food source using the information provided by the first group; and (3) the scout bees group search randomly to discover more food locations with a high amount of nectar. In an optimization method, the solution is represented as a food source and the solution’s quality is represented by the amount of nectar [48]. Figure 1.5 illustrates the flowchart of the ABC algorithm [71].

The ABC algorithm was applied to deal with several CC optimization problems; some of these works are presented in the following. The ABC algorithm in [83] was improved and utilized to solve task scheduling problem in CC to map tasks to available resources. It worked to improve the computation performance and the resource cost by minimizing execution time and costs. In addition, it grouped the tasks and then sent them to the cloud resources. The experiments were performed using the CloudSim platform to simulate cloud’s platform. Four cloudlets were examined (i.e., 25, 50, 75, and 100). The results of the ABC algorithm outperformed activity-based costing algorithm.

Another effort to solve the problem of task scheduling has been found in [71], in which the system’s production was enhanced by scheduling tasks efficiently to VMs using the ABC algorithm. The experiments were performed using the CloudSim platform. The experiments tested a different number of VMs (from 20 to 100) and a different number of cloudlets (from 100 to 900); however, its results were not compared with the state-of-the-art algorithms.

The authors of [10] presented a version of the ABC method in order to maintain load balancing and migration of tasks across VMs as well as minimize the makespan. The food sources in this algorithm were represented as underloaded machines. The experiments showed the superiority of the proposed method and were performed using different numbers of cloudlets (i.e., 10, 15, 20, 25, and 30); then the output makespan and the number of jobs migration between the ABC and the modified ABC were compared. Similarly, the ABC algorithm was applied in [52] to improve load balancing and tasks migration through VMs. The overload tasks that were deleted from the VMs were represented as honey bees and the priority of these tasks was taken into account. The experiments tested 10 to 40 tasks and calculated using the CloudSim platform. The results showed an improvement in computation time and minimizing the response time of VMs. In the same trend, the authors of [85] maintain load balancing and migration of tasks using different numbers of tasks, namely 20, 30, 40, 50, and 60. The experiments showed a cost reduction of using VM instances. Another version of the ABC algorithm was proposed by the authors of [58]. It combined the ABC and ant colony optimization (ACO) algorithms to solve task scheduling in CC. The experiments were applied using
The ABC algorithm was also combined with the GA algorithm to perform energy-efficient task scheduling [53]. The experiments were applied over the CloudSim platform and six cloudlets were used (i.e., 10, 100, 200, 300, 500, and 1,000). The proposed method showed improvement equals 11% than the classical version of the ABC algorithm.
the CloudSim platform and the number of tasks and VMs varied between 10 and 40. The results depicted that the proposed method showed good performance in energy consumption and minimum makespan than the GA algorithm.

1.3.3 Lion Optimization Algorithm

The LOA is an optimization method which was developed in 2015 by the authors of [101]. It aims to simulate the behavior and lifestyle of lions, based on hunting, mating, and defense [101]. The lion’s Kingdom is composed of two kinds of life organisms known as resident and nomads. The term “Pride” represents a set of residents who live together. We note that the residents can switch their lifestyle (i.e., the resident can become a nomad and vice versa). The hunting principle used for lions is specific because they seek the prey by cooperation. The hunting concept consists to assign three roles for lioness as left, center, and right wing as shown in Figure 1.6. During the chase, each lioness updates its location based on their placement and the locations

**FIGURE 1.6**
The natural behavior of lions hunting.
of all elements. The lionesses run very quickly in an exponential way which facilitates the task of hunting prey so that the lionesses work together to catch the victim. For this, they encircle the prey and quickly attack the victim from the opposite position. Due to this reason, the operator of opposition-based learning (OBL) is used in artificial lion algorithm [101].

The LOA is based on the following three important operators [101]:

- Mating: the purpose of this operator consists to breed new descendants using crossover and mutation. Lionesses can proliferate with one or several resident males while the nomad female shares their desire with a single male that are chosen arbitrary. The process of mating is considered as a linear crossover between parents for generating two descendants.

- Territorial defense: this operator consists to estimate the strength of resident lion and nomad lion according to their fitness. If the fitness of the nomad lion is better than the fitness of the resident lion, then we accept this nomad in the pride and the resident must migrate to another area.

- Territorial takeover: in the natural process, the old territorial males and the new territorial males fight among themselves and the strongest remain in the pride. In artificial life of the resident lions (males and cubs), we evaluate all resident lions based on their fitness and the weakest male leave the troop and emerge as nomad while the rest of the males be converted to resident males.

The flowchart of LOA is described in Figure 1.7.

In Ref. [8], the authors aim to order the jobs by LOA in CC. The goal of this study consists to reduce the makespan of the agent that indicates the objective value for each individual. We initialized randomly the population of lions by devoting cloud jobs to VMs where (%N) is the rate of nomads and the rest of the population is residents which is randomly divided through P subgroups appealed Prides. From this subset, we select S portion of females and the rest is considered as males. During the search process, each pride marks its best-visited position well known as a territory. Furthermore, the strategy of hunting consists to divide the lionesses into seven stalking roles, grouping by left wing, center, and right wing position. During the hunting, the OBL is utilized for encircling and attacking the prey. Other operators are used in LOA such as Mating, which produces new offspring, defenses territorial which allow the emergence of new mature males and the weakest males become nomads. Finally, the concept of migration process forces the nomads to leave the kingdom. Devagnanam and Elango [23] proposed an algorithm called E-LOA for allocating optimal resources in the cloud. This paper combines the LOA with an exponential weighted moving average for providing the optimal solution.
FIGURE 1.7
The flowchart of lion optimization algorithm [73].

1.3.4 Whale Optimization Algorithm

The WOA is an optimization algorithm developed by the authors of [65] in 2016. The WOA simulates the hunting process of the whales that live either in packs or individually. Baleen whales have their own behavior in hunting prey (usually a school of krill or small fish), by generating different bubbles in a circular shape. This method is called bubble-net feeding [96]. The hunting process starts when a whale recognizes the prey’s location (the current best solution), then encircles it by creating water balls in a spiral way surrounding the target and swims near to the apparent area. In the WOA, when the near-optimal solution (prey) is reached, other whales try to modify their locations considering the position of this hunt among a shrinking surrounding and across a spiral-form concomitantly, depending on a half percent of probability to select among these two processes. Figure 1.8 illustrates the flowchart of the WOA.

Sreenu and Sreelatha [89] proposed a technique for scheduling tasks in CC known as W-scheduler, based on the WOA and the multi-objective model which is calculated by cost and makespan. Another enhanced version of the scheduling-based WOA are presented in the literature; for example, we found the work of Abdel-Basset et al. who merged the local method with the WOA.
FIGURE 1.8
The flowchart of the WOA [27].

for solving the configuration of schedule flow shop [1]. Recently, the emergence of chaotic theory attracts many researchers in the field of metaheuristics for finding a solution of workflow scheduling in computational cloud.

1.3.5 Gray Wolf Optimization

In 2013, Mirjalili et al., introduced a swarm intelligence technique called GWO [66]. Their idea is based on mimicking the hunting behavior of wolves and the leadership hierarchy. For simulating the leadership hierarchy, we distinguish
mainly four categories of gray wolves. Furthermore, the major steps of hunting are focused on tracking, encircling, hunting, and attacking the prey. For applying this algorithm, a random population of wolves is generated wherein each wolf is represented as a vector in N dimension search space. According to their fitness, the four best wolves are assigned to the alpha, beta, and delta groups. For each generation, the location of wolves is changed and the process will be executed for a maximum iterations’ number. Figure 1.9 illustrates the flowchart of the GWO.

In task scheduling, the GWO is widely used as an example. Khalili & Belbamir proposed a version named Pareto-GWO for solving dependent tasks of workflow scheduling using multi-objective algorithm [51]. As fitness functions, the authors seek to minimize makespan and cost, and maximize the throughput of a provider’s resources. In task scheduling, each solution corresponds to task-resource mapping while GWO is developed for continuous optimization problem whereas the task-resource mapping is a discrete problem. The authors suggest applying the smallest position value propriety for seeking the position of wolves. Natesana and Arun proposed an enhanced version of GWO called mean GWO for obtaining a solution of task scheduling in

![Flowchart of the GWO](image-url)

**FIGURE 1.9**
The flowchart of the GWO [32].
heterogeneous cloud [68]. The authors used energy consumption and makespan as a fitness function. Learning-based opposition is integrated into GWO for parallel scheduling in a cloud platform [67].

1.3.6 Bat Algorithm

Bats are considered mammals, but not much is known about them. However, they have fascinated the greatest researchers and are now of interest to scientists. Their ability to fly, unique in mammals, allow them to devolve at night avoiding obstacles, even at high speed, thanks to their sophisticated echolocation system. They are one of the animals with a very high number of species; there are about 996 species of bats. Their sizes vary between the so-called microchiropter or “microbat” weighing about 1.5–2 g up to the giant bat, called megachiropter or “megabat” which has 2 m of wingspan and weighs over 1 kg. Microchiroptera have a body length ranging from 2 to 11 cm. Most bats use echolocation to some degree among all species. The majority of microchiroptera are insectivores. They emit very loud pulsations through the mouth or the nose (ultrasound). As soon as this ultrasound observes a hurdle (plant, prey), it bounces toward the bat. Their brain can calculate the location, velocity, shape, and the distance of the target to be detected [38]. This principle of bats has helped many scientists to create an algorithm called the Bat algorithm which could be noted as BA. It is considered as a type of the population-based metaheuristic algorithm. Xin-She in 2010 proposed for some optimization problems processing [98]. This is performed through the simulation of the echolocation attitude of bats. Echolocation is a biological sonar (because of lack of vision) that can detect distance, and they also have the ability to distinguish the difference between food/prey and obstacles.

Bats whiffle with a random way of possessing a location, certain frequency, certain speed, nuisance, and a rate for emitting the pulse to search victim. Updating bat velocities and positions are similar to the normal procedure for PSO algorithm. BA can be recognized as a hybrid of the PSO and the exhaustive algorithm of the local search depended on the rate of emitting the pulse and nuisance [41]. The BA is applied for resource scheduling in CC by Jacob for minimizing the makespan, and the author has found that BA showed its efficiency in terms of accuracy than the genetic algorithm (GA) [42]. A combination between the BA and the algorithm of the gravitational scheduling was introduced for task scheduling in CC with taking into account some constraints called deadline ones, in addition to a model called trust model [55].

In Ref. [54], the authors developed a hybrid algorithm between the BA and the harmony search called BATS-HS for processing some problems of the scheduling through the CC. Where selection of a resource depended on the trust; by using the proposed hybrid algorithm the time of computation for the functions or tasks was reduced by about 10.4% versus the selection of a resource in a random way by using the BA. Bat algorithm is further
enhanced by chaos theory for the process of scheduling for some functions of tasks through the CC [35]. Bi-objective measures are used for evaluating the performance of the Bat algorithm such as energy consumption and makespan. A flowchart for the BA algorithm is displayed in Figure 1.10 [102].

Raghavan et al. [72] suggested a method that was used as a binary version of the Bat algorithm for scheduling work process in computing the clouds. The mapping of functions or tasks and resources was applied using this method. The proposed method helped in selecting the optimal resources, where the

FIGURE 1.10
The flowchart of the BA algorithm [102].
cost of the workflow of this method was the minimum against the compared methods. Marichelvam et al. [62] explored the scheduling problems of the hybrid flow shop (HFS) for multistage using the suggested technique. The problem of the HFS can be considered as the general case of flow shop where there are several machines.

HFS can enter in many industries, such as chemical; textile; and ceramic, iron, and steel, and it can be recognized as an NP-hard problem. The authors improved the BA to be used for solving the HFS problems. They tested their proposed technique on some benchmark functions. The authors compared the proposed Bat algorithm with some popular algorithms such as the GA in addition to the PSO, and the proposed method was proved to have the superiority. Sathya et al. [82] dealt with the process of controlling the load frequency through utilizing the BA depended on the dual mode through scheduling of the controllers of type PI for the power systems that are interconnected. Wang et al. [95] proposed using the Bat algorithm for developing the revenue and energy-aware, in addition to developing combination scheduling method through data center of cloud. The author applied the BA depended on the controller of type dual mode PI for the power system which can be classified as interconnected thermal and multi-area for the parameter PI controllers tuning. The authors tried to compare their controller with some others from the traditional controllers of type PI and their scheduling with the fuzzy gain. The experimental results declared that the proposed BA, which used the scheduling through the dual mode gain of controllers of type PI (BIDPI), provided a suitable transient. The results showed that the new controller has a lower sensitivity in the system parameters changes.

Sagnika et al. [80] presented a solution for the workflow or the data-intensive problem using the BA algorithm through scheduling through the system of the computation of clouds. The proposed algorithm showed a superiority compared with the PSO and CSO algorithms. Such technique gave a cost, with the best convergence, to be the optimal and with suitable distribution. Chaturvedi et al. [20] applied a developed version of the BA to deal with scheduling of workflow in case of the multi-objectives through some clouds which leads to reduce the time of the execution and increasing the reliability through maintaining the budget through a limit of user-specified. This technique was compared with the randomized evolutionary algorithm (BREA), where it depends on using the method of greedy for allocating resources to be suitable for tasks of the workflow, high reliability with low time execution. The experiments proved the superiority of the BA algorithm during its comparison with other evolutionary algorithms. Rani and Kannan [78] focused on the optimal utilization of VMs, modeled around directed acyclic graph with topological ordering to schedule task using nature-inspired Bat algorithm. It can minimize the time waiting of a task and the idle time of VMs due to its high convergence. The authors gave results of the comparison of the Bat algorithm through MATLAB® with First-In-First-out (FIFO), PSO, and Harmonic Search (HS), and it proved the superiority.
Aguirre et al. [3] used the BA to be utilized in the scheduling problems and allocation of work through a multiprocessor, where the makespan and tardiness were the need for the minimization. There is a scheduler for dividing the processors’ time among the processes needed to be executed. One task can be executed by a processor, and the scheduler can determine the task that will be in the next process, and the task without restriction of dependence. Malakooti et al. [60] used the Bat algorithm in solving the scheduling problems of the multiprocessor that is energy-aware. The authors performed a comparison between the BA and the GA which proved an efficiency in solving the scheduling issue for multiprocessors and with the objective which is single and its fitness function may be the consumption of energy, or tardiness, or makespan alone or as combination of them for solving the bi-objective multiprocessor or tri-objective multiprocessor through the combination of three of them, energy with makespan, and energy with tardiness. The Normalized Weighted Additive Utility Function can be an effective alternative. The experimental results clarified that the proposed algorithm can find solutions that correspond to the assigned weights in an efficient way.

1.3.7 Cat Swarm Optimization

The cats’ behavior has interested the engineers and scientist, hence the birth of an algorithm called “cats optimization algorithm (COA)” introduced by Chu and Tsai [22]. This algorithm is inspired by swarm theory that was applied in the continuous domain of optimization. Two processes are used to understand the behavior of natural cats known as global and local search. In a global search, the seeking mode is used, whereas the local search employed tracing mode. During resting, the cats are highly alert and they react slowly toward their environment; this search is known as seeking mode. As soon as the cats hear the presence of a prey, they react quickly and attack to reach the target; this research is known as tracing mode. In the CSO algorithm, each artificial cat has a position vector and their velocities are updated during the course of the iteration. An active bit is employed for determining the life mode of cat based on their fitness. A simple flowchart for the COA is shown in Figure 1.11 [22].

The authors of [22] proved that the CSO provided better convergence than PSO. Several contribution-based CSOs are developed for a jobs and workflow scheduling. For example, the authors of [15] presented a CSO-based optimization scheduling in order to devote the jobs of an application onto unoccupied devices. The proposed algorithm takes into consideration the cost of the data transfer among the two dependent devices and the cost of jobs on various machines. They tested the proposed CSO using a hypothetical workflow also and made a comparison between the results of the workflow scheduling with the PSO. Their evaluation proved that, firstly, the CSO reached an optimal task-to-resource (TOR) which leads to reduce the total cost. Secondly, the
CSO had an advantage over the existing PSO through iterations, finally, the CSO assumed a fair load distribution on the available resources.

Gabi et al. [36] presented an optimization technique known as OTB-CSO, in order to schedule jobs and to reduce the complete time processing using the orthogonal Taguchi (OT) with CSO. The aim of that study was optimizing the task scheduling based on the proposed approach for reducing the total task execution delay. They used the Taguchi orthogonal approach for the
resting mode of CSO in order to devote jobs on VMs which have a reduced
time of execution. Also, they used the tool of the CloudSim platform for
implementing the proposed algorithm whose impact was tested with 5, 10, and
20 VMs with input tasks and performed by makespan and the measures of the
degree of imbalance. For 20 VMs, the proposed approach, OTB-CSO, reduced
the makespan of the overall scheduled jobs with 2.6% enhancement closed
minimum and maximum jobs first. In addition, the proposed method was
compared with PSO-LDIW and HPSO-SA based on the degree of imbalance
and the obtained result showed an enhancement over existing techniques. Also,
it was concluded that the OTB-CSO has the superiority for optimizing task
scheduling and enhancing the performance of CC by reducing the task delayed
of execution, and a better utilization of the system.

In Ref. [84], they proposed CSO for load distribution of workloads in computa-
tional cloud. The principal goal of this study was to achieve higher cus-
tomer satisfaction and resource utilization ratio. The obtained results were
evaluated using a load balancing in a large-scale cloud and showed high
performance of computing compared to other algorithms.

1.3.8 Cuckoo Search Algorithm

Yang and Deb [99] suggested the Cuckoo Search (CS) technique, which takes
its inspiration from the cuckoos reproduction strategy, who they lay their eggs
within a nest, and for increasing the probability of hatching their own eggs,
they try to hide the eggs of the others too. The flowchart of the CS algorithm
is presented in Figure 1.12 [14].

Many researchers used this technique for scheduling of tasks in the CC.
Agarwal and Srivastava [2] suggested using the CS for dealing with the prob-
lems of scheduling of tasks, where this can lead to dividing the tasks among
the VMs in an efficient manner, in addition to maintaining the minimum of the
response time totally. This technique determines the tasks for the VMs based
on the processing power, that means that instructions which are almost million
per seconds and the tasks’ length. They compared the proposed CS approach
with the FIFO and greedy algorithms using the simulator of the CloudSim
platform. The experiments of this paper clarified that the CS proved to be
more efficient than the other algorithms.

Amtade and Miyamoto [9] used the CC to evaluate the CS algorithm
against GA. The experimental results showed that the CS proved to be more
efficient than the GA according to the objective value.

Mandal and Acharyya [61] proposed swarm intelligence and metaheuristic
methods like firefly algorithm (FA), simulated annealing (SA), and CS; such
techniques were used for reaching an optimal solution for the scheduling of
tasks, and reaching the best resources to try to reduce the total time of pro-
cessing the VMs. They focused on making the shared resources to be more
efficient. The objective of such techniques is reducing the time of processing
FIGURE 1.12
The flowchart of CS [14].

The VMs which can perform a set of tasks. The results showed that the FA outperforms better than the SA and CS Algorithm.

Marichelvam et al. [63] supposed a version of the CS algorithm for real flow shop scheduling problems as an industrial hybrid by reducing the HFS scheduling problems makespan. They considered the hybrid flow shop in case of the multistage (HFS) scheduling problems. Such problems are recorded as NP-hard. In addition CS algorithm was used for reducing the makespan. Another heuristic algorithm, is the Nawaz-Enscore-Ham (NEH) algorithm that can be associated through the initial solutions for trying to reach the optimal solutions more quickly in the enhanced algorithm. Where the enhanced algorithm was built based on the data from a company of leading furniture manufacturing. The experiments proved that the improved algorithm outperforms the others.

Wang et al. [94] presented an algorithm called NCS, for the scheduling problems of flow shop (FSSP). The smallest position value rule was applied...
for dealing with the discrete variables and convert continuous solutions into ones. Also, a method based on the NEH algorithm was applied for initializing populations to generate better initial solutions. The experimental study was implemented on a group of Taillard’s benchmarks. These experiments proved that the NCS outperformed the standard CS. Burnwal and Deb [11] proposed using the CS for scheduling jobs of a system through reducing the cost of the penalty resulted from the manufacturing delay and the maximization of the time for using the machine. This technique was enhanced using the Lévy flight operator due to which the solution had a discrete nature on a scheduling FMS problem which had 16 machines with 43 jobs. MATLAB was used to perform all experiments. The results were compared to the PSO and GA. The CS showed good performance than the other algorithms.

Ouaarab et al. [70] presented the Discrete Cuckoo Search (DCS) for dealing with the scheduling problems of the Job Shop (JSSP), where they used the Lévy flight operator for enhancing the DCS algorithm through three stages to complete the search space. It applied the local and the global random walk and showed that the DCS can be checked in its performance within benchmarks’ set of JSSP from the library named OR. The authors also tested it by comparing it with the PSO algorithm for proving that the DCS had the superiority than the PSO.

Li and Yin [56] used CS based with the memetic algorithm as a hybrid denoted as HCS, for the problem of the permutation flow shop (PFSSP). Also, the authors used a rule of the largest-ranked-value (LRV) which depends on the random key for the CS be suitable for the PFSSP, for converting the continuous location of the CS to a permutation of discrete job. Then hybridized the NEH with the population generated randomly with a specific quality and diversity. Then, the CS was used for including the nest vectors for the process of exploration and improving the capability of the local exploitation. The authors tested the proposed approach performance for solving the PFSSP using the benchmarks, and this proved effectiveness and superiority of the proposed approach.

Hanoun et al. [39] suggested the CS approach for solving the scheduling problem of the job shop in the multi-objective which is single with minimizing the waste of the material and the tardiness time totally. The waste in the material was determined according to some factors to show the material reduction that van be done when producing two jobs through using same materials in a sequence way. There was an estimated factor for computing the cost of the saved processes for each job according to its material type. A formulation of multi-objective optimization problems was adapted to generate the set of schedules that maximize the cost savings and minimize the computation time; all trade-offs were considered for the fitness functions which were conflicting each other. This was performed with developing the CS by using Pareto for a Pareto Archived Multi-Objective Cuckoo Search (PAMOCS) to search for the nondominated optimal solutions of type Pareto. The accuracy of the PAMOCS performance was displayed through the closeness comparison
for the resulted solutions of the true Pareto front of the method of enumeration. Experimental results assumed the superiority of the CS for solving such types of problems.

1.3.9 Hybrid Swarm Algorithm

In this context, we distinguish two types of hybridization, the first type consists of combining the global approach with local method known as memetic algorithm (MA), in view to enhance the convergence criteria, whereas the second type combines two or more approaches to global optimization known as cooperative metaheuristics in order to give more flexibility for exploitation and variety of the swarm. The concept of MA consists to apply firstly the global optimization method on the population which are initialized randomly and then applying the local approach on the best individual. For local approach, we found different approaches such as SA, Nelder and Mead’s (NM), and tabu search (TS). The efficiency of the MA gave the opportunity to the researchers to test them in real field of application such as the scheduling of jobs and workflow scheduling in cloud.

In Ref. [103], the authors presented a memetic algorithm which combines PSO and SA for reducing the schedule length. The PSO was applied first and then SA was applied for the purpose of improving the convergence. In Ref. [90], the authors described a combination based on PSO and Tabu search for task scheduling. The proposed method consists to split the particles into two groups: the first one updates the position and velocity of particles using global search PSO, whereas the second group updates it using local search (TS) and then we combined both groups in order to find the optimal solution.

Navimipour et al. [69] solved the problem of jobs scheduling using the technique of CS. Their paper aimed to devote the jobs in CC based on the combination between CS and the lévy flight operator. The paper realized three experiments according to the variation of the probability (Pa) from 0.2 to 0.6. They concluded that the lower value of Pa provided high performance in terms of the convergence and the computation time. The authors of [64] tried to deal the problem of energy in jobs scheduling. As solution, Mezmaz et al investigated a parallel method based on GA and DVS (dynamic voltage scaling) for reducing the makespan and the energy. The experimental results showed that the proposed technique accelerated the process of convergence and the rate of energy consumption was enhanced by 47.5% as compared to other algorithms, whereas the makespan was ameliorated by 12% in contrast to other works of the literature. Raju et al. [74] proposed a hybrid algorithm that merged the advantage of ACO and CS, for dealing the problem of task scheduling. In order to allocate the adequate resources, this hybrid algorithm decreased the makespan, since the tasks had been performed within the specified time.

Babukartik and Dhavachelvan [11] developed a method for job scheduling that combined the merits of ACO and CS. Kanagaraj et al. [47] presented an efficient hybrid algorithm based on two global search known as GA and
CS. Their paper aimed to solve the global optimization benchmarks and three design-constrained problems. The authors compared the proposed hybrid with other recent ones and used some effective measures for evaluating the performance, from which, the superiority of the proposed one came. The authors of [93] proposed two functions known as ranging and tuning functions based on PSO for solving the inertia weight. For avoiding the problem of convergence of PSO, the authors aimed to combine BA with PSO for finding the key to the scheduling jobs in cloud.

In the second type of hybridization, the authors of [50] proposed a hybrid algorithm called SFLA-GA which combines basic GA with SFLA (shuffled frog leaping algorithm) for seeking the ideal allocation of resources in CC. Because of the easier implementation and faster convergence of SFLA, this algorithm was widely used in the area of CC with the algorithm of CS for allocating the optimal resources [24]. Recently, Torabi and Safi-Esfahani proposed a solution for independent jobs scheduling in dynamic environment using a hybrid algorithm known as IRRO-CSO [92]. Their method tried to combine improved RRO (Raven Roosting Optimization) with CSO (Chicken Swarm Optimization) with the purpose of reducing the time of response and execution and boosting the rate of throughput computing. In this paper, CSO was employed for ensuring the harmony among the global and local searches, whereas the IRRO was used for finding a global solution.

1.3.10 Multi-Objective Swarm Optimization

Recently, some works are proposed as multi-objective optimization problems for solving workflow and job scheduling in CC. Reddy and Kumar in 2017 proposed a method for scheduling tasks based on multi-objective WOA [79]. In their study, the quality function was controlled by three factors known as energy, the use of machine, and the QoS. The performance of MOWOA was quantified by the minimum value of fitness. The simulation was realized using the CloudSim tool with JAYA. The results of the simulation present a superior performance compared with other metaheuristics algorithms such as ACO and PSO in terms of the time, the availability of resources, the scheduling cost, and energy.

In Ref. [44], Jena aimed to schedule jobs using a version of multi-objective PSO in CC called Nested PSO. Their study was focused on task scheduling by optimizing the processing time and energy. Jena used CloudSim platform for simulation and the obtained results showed superior quality according to multi-criteria. In Ref. [49], the authors proposed an algorithm for multi-objective scheduling based on multi-objective bacteria foraging optimization algorithm (MOBFOA). Their study consists of schedule independent jobs by minimizing flow time, makespan, and the cost of resource utilization. The proposed method showed a good performance compared with NSGA-II and optimized MOPSO in terms of measured criteria.
Another study for task scheduling in CC was developed recently by Srichandan et al. in 2018, based on hybrid MOBFOA in heterogeneous cloud environment [87]. The authors aimed to reduce both, the criteria of makespan and consumption of energy, simultaneously. The hybrid algorithm employed the operators of exploration/exploitation used in the GA to BFA with the purpose to enhance the convergence. The obtained results depicted that the hybrid MOBFOA provided high quality in contrast to other approaches according to the convergence, stability, and the diversity of the population.

In Ref. [88], the authors proposed NSGA-II based multi-objective job scheduling for CC. Their work aimed to decrease the use of energy and to minimize the makespan. Furthermore, the neural network was used for predicting the accessibility of the resource in a cloud. The numerical results of single objective based on makespan and consumption energy showed that the GA decreased effectively the energy consumption by incorporating the Dynamic Voltage Frequency Scaling (DVFS) in optimization procedure, however, the makespan increased. In addition, the multi-objective NSGA-II showed that the optimization procedure boosted by neural network provided a better solution. In Ref. [77], the authors considered four conflicting measures, known as minimizing job transfer time, job execution cost, energy consumption, and job queue length for devoting task based two multi-objective evolutionary algorithms called MOPSO and MOGA. From their study, the authors concluded that MOPSO was rapid and more efficient than MOGA for dealing job scheduling in CC.

Gabi et al. [37] proposed a multi-objective approach (MOA) based on a combination between the cat swarm algorithm and SA known as CSM-CSOSA. They used the OT technique with the purpose of enhancing the SA and providing better performance. In addition, they used a MO-QoS model for evaluating the performance of the CSM-CSOSA by CloudSim under two datasets. Quantitative analysis of CSM-CSOSA was evaluated with measures of cost, QoS, time, and performance enhancement rate. Meantime, the scalability analysis of CSM-CSOSA utilizing Isospeed-efficiency scalability metric was evenly defined. The experimental study showed that the CSM-CSOSA outperformed MOGA, MOACO, and MOPSO by providing lower values of time and cost. The presented research in [45] was focused on job scheduling for CC using multi-objective ABC (MO-ABC). Their objective was to reduce the processing time, energy, and cost resource. The proposed algorithm (TA-ABC) was tested by CloudSim and a comparative study was realized between TA-ABC and other algorithms. The experimental study showed that the MO-ABC reduced energy consumption and time execution by 32% and 26%, respectively in comparison with the MASA and the RSA.

In Ref. [100], the authors proposed a multi-swarm optimization algorithm (MSMOOA) for parallel dependent workflow scheduling. Their work aims to find a compromise between makespan, cost, and energy consumption. Two classes of particles in each swarm are generated. The first class communicates
with several swarms at the same time, whereas the second class can only permute the information within particles in the same swarm. In Ref. [104], the authors presented a MOACO for task scheduling. The authors considered two constraints of cost and deadline in CC. The experimental results showed a high performance in terms of makespan, the cardinal of a deadline, cost, and the rate of private machine employment. Chandrasekaran and Simon [14] proposed a hybridization method between CS algorithm and fuzzy system for solving MO-unit commitment problem. For solving such problem, two versions of CSA were generated—binary version of CSA and real version of CSA. The real version of CS can deal with the economic dispatch problem. The proposed technique was evaluated for single and MO-problem. The superiority of the proposed algorithm was identified on several unit test systems through the comparison of it with some other recent techniques via performance.

Bilgaiyan et al. [20] applied cat swarm-based MOOA for workflows scheduling in CC environment. Their work aimed to decrease the CPU time, cost, and makespan. The performance of the proposed algorithm was proved by a comparison with MOPSO technique. The author proved that the swarm-based solutions for such problems were the best suitable due to the requirements of users. So, MOOA became more used for workflow scheduling.

1.4 Conclusion

Task/Job scheduling in CC is a popular issue and plays an essential role in improving reliable and flexible platforms. Therefore, the researchers considered it as an optimization problem. Jobs scheduling works to map the resources to jobs, whereas the optimal scheduling method should guarantee a QoS and good performance of the CC by considering some criteria such as computation time and cost. The good results of swarm intelligence methods in solving many optimization problems led for their application in this field. Various swarm intelligence methods are applied for solving task scheduling issues in CC platforms. In this chapter, an overview of swarm intelligence for solving the problem of task scheduling in CC was presented including PSO, ABC, LOA, WAO, Bat algorithm, GWO, CSA, cat optimization algorithm, hybrid swarm algorithms, and multi-objective swarm optimization. The experimental results of the previous studies showed their ability to solve different types of scheduling tasks in CC. Most of these studies used a simulated environment to perform their experiments. The most used environments were CloudSim and CloudGrid. In addition, the most used number of VM varied from 5 to 2000 and the tasks ranged from 10 to 4000. Several studies solved this optimization problem using one swarm intelligence method, whereas the others used two or more methods to solve this problem as well.
as some studies treated with task scheduling as a multi-objective problem because they tried to minimize many factors along with makespan value. All the previous studies did not reach the optimum results and the field is still wide to perform more and more experiments in the future to enhance and improve the services of CC.

Bibliography


Techniques for Resource Sharing in Cloud Computing Platform

B.L. Radhakrishnan, S. Sudhakar, R.V. Belfin, P. Karthikeyan, E. Kirubakaran, and K. Martin Sagayam

Karunya Institute of Technology and Sciences

CONTENTS

2.1 Introduction ................................................. 40
2.2 Intercloud .................................................. 41
   2.2.1 Resource Sharing in Federated Cloud ................. 42
      2.2.1.1 Agent-Based (Centralized) Resource Sharing in Federated Cloud ............... 42
      2.2.1.2 Peer-to-Peer Resource Sharing in Federated Cloud .............................. 44
   2.2.2 Resource Sharing in Multi-Cloud ..................... 46
      2.2.2.1 Agent-Based Service Selection Approaches in Multi-Cloud .................. 46
      2.2.2.2 QoS-Based Service Selection in Multi-Cloud ................................. 47
2.3 Social Cloud ................................................ 47
2.4 Mobile Cloud .............................................. 50
   2.4.1 Incentive Mechanisms in Mobile Cloud ............ 53
2.5 Cloud Manufacturing ....................................... 54
   2.5.1 Resource Sharing in CM .............................. 54
2.6 Vehicular Cloud ........................................... 57
   2.6.1 Vehicular Cloud Architecture ........................ 57
      2.6.1.1 Inside-Vehicle Layer ................................. 57
      2.6.1.2 Communication Layer ................................. 57
      2.6.1.3 Cloud Layer ....................................... 58
   2.6.2 Resource Sharing in Traffic Management .......... 59
   2.6.3 Resource Sharing in Multimedia Services ........ 59
   2.6.4 Resource Sharing in Smart Parking ................ 60
   2.6.5 Vehicle-Assisted Cloud Computing (VACC)
      for Smartphones ........................................ 60
   2.6.6 Resource Sharing in Vehicular Fog Computing .... 60
2.7 Green Cloud ................................................ 61
2.1 Introduction

Resource sharing is a way to share the resources with other users or providers in the cloud. Cost of IT involves capital expenditure, operational expenditure, and total cost of ownership. This cost of resources is a barrier to the small- and medium-sized service providers or customers. Also, the underutilization of the resources brings down the profit of the provider or customer. The service providers and customers share their resources to reduce the cost and increase the resource utilization. The small- and medium-size providers are unable to meet the dynamic resource requirement due to lack of resources. Multiple providers or customers aggregate their unused resources to create a resource pool. The dynamic demand of IT can be meet out easily by the resource pool. The resource sharing at different IT can be depicted in Figure 2.1.

FIGURE 2.1
Resource sharing at different clouds.
Inter Cloud (IC) integrates many cloud providers to exchange resources among themselves. IC operates in two ways: (1) cloud Federation (CF) and (2) Multi-Cloud (MC). CF is a group of service providers that voluntarily share their resources. MC customers use different service provider’s resources to host their applications. Social media platform enables people to collaborate for resource sharing. Like-minded people create groups in social media to share, rent, and barter their resources that forms Social Cloud (SC). So far smartphone market has attracted more than 2.5 billion people worldwide. Smartphone has resource limitations, which is overcome by collaboration among smart devices and Cloud. Mobile cloud extends the mobile computing by using cloud providers, and wireless network operators to enrich the functionality.

The Small and Medium Enterprises (SMEs) in manufacturing have resource and capacity shortage. Sharing resources increases productivity and reduces the cost of production. Manufacturing Cloud (MC) simplifies the resource sharing among SMEs. Resource sharing in MC happens in two environments: (1) IT and (2) Manufacturing. Modern vehicles possess reasonable computing devices. Pooling of computing resources increases the computational capability of vehicles. The resource pool is used to perform the computation of traffic management, multimedia services, and smart parking. Vehicular Cloud (VC) enables the connected vehicles to process and share information with others. Energy saving is significant in today’s world. Energy-aware resource utilization reduces energy consumption and pollution. supports energy-aware resource sharing, which saves energy, reduces carbon emission and water consumption. The following sections describe various resource sharing methods in clouds.

The remaining of the chapter is organized as follows: Section 2.2 discusses IC, Section 2.3 describes SC, Section 2.4 explains mobile cloud, Section 2.5 discusses manufacturing cloud, Section 2.6 describes VC, and Section 2.7 explains green cloud.

### 2.2 Intercloud

Interconnects clouds together to augment its capacity and resource utilization. It enables cloud providers to share underutilized resources to maximize profit. Based on the party collaboration, IC is categorized as volunteer federation (federated cloud) and independent cloud (multi-cloud) [1]. In a federated cloud, providers join together to share and exchange their resources. Federated cloud is subdivided into centralized cloud (agent-based resource allocation) and cloud (peer allocates resources). In a multi-cloud, providers are not needed to join together to exchange resources and customers are responsible for gathering and controlling resources from multiple providers. Services and libraries are the further classifications of multi-cloud. Figure 2.2 shows the classification of IC.
2.2.1 Resource Sharing in Federated Cloud

2.2.1.1 Agent-Based (Centralized) Resource Sharing in Federated Cloud

An agent is an autonomous software system that discovers, matches, selects, negotiates, schedules, and monitors resources in the cloud [66]. This section discusses various approaches to agent-based resource sharing in the federated cloud environment.

A Multi-Agent System for Cloud Resource Enhancement: this Cloud Resource Bartering System (CRBS) implemented by Goher et al. is a platform for IaaS service provider to barter underutilized pooled resources without financial transaction [82]. Agents in CRBS enable resource exchange among multiple providers and handle the dynamic demand. Customer submits a resource request along with the following parameters to the user agent: resource type, number of resources required, maximum cost, resource urgency, and resource region. Multi-criteria based resource selection algorithm is used to handle the submitted request. The resource selection process considers the following factors, such as resource type, number of resources required, exchange time, maximum cost, resource urgency, and provider’s rank while selecting resources. Bartering agent uses a price determination algorithm to satisfy both the customer and the provider during price negotiation. Finally, the control of the selected resources is transferred to the customer after price negotiation and service level agreement.

Automated Bartering System: this multi-agent based e-barter system developed by Demirkol et al. is an online system for IaaS service provider to exchange resources between customer and provider [22]. This e-barter system uses many intelligent software agents for trade management and resource matching. The resource matching in system uses ontology-based comparison. It uses an intelligent Semantic Web Service (SWS) agent to deduce closeness between the quoted price and the selling price. The customer agent adds
resource request by sending a request for proposal message to the system. Barter agent discovers matching resource for the request and sends matched resource details to the customer agent. Then, negotiation between customer and provider is initiated. Customer agent sends, accept, or reject message to the system after the negotiation. Cargo agent is responsible for shipping resource control to the customer after a successful negotiation.

For Resource Matching: a Belief-Desire-Intension (BDI) agent-based semantic e-barter system proposed by Cakmaz et al. is a multi-agent system for IaaS Service provider to share resources [12]. uses a JACK framework to build agents in the system. When the customer submits a request, the BDI uses two ways to match resources—(1) exact matching: matching of resources as per request submitted by the customer and (2) semantic matching: matching of resources which are close to the request. First, the matchmaker agent applies the exact matching technique. If an exact match fails, then it applies the semantic matching. Second, the customer agent executes resource exchange plan; if the plan type is “no negotiation”, it executes a negotiation plan using three ratios: minimum ratio, actual ratio, and an inverse ratio. Minimum ratio is the minimum ratio between the cost of the customer to the resource and the quoted cost of the provider. The actual ratio is the ratio between the quoted cost of the customer to the resource and quoted cost of the provider. The inverse ratio is the ratio between the quoted cost of the provider and the quoted customer cost to the resource. User-agent checks the customer Requested Ratio (RR) with the New Actual Ratio (NRR) using the following steps:

step 1: NRR \leq RR,
if true bartering successful resource swapping happens else go to step 2
step 2: upper-ratio \leq NRR
if true then bartering unsuccessful and the process terminated,
else select a new offer and repeat step 1.

The Multi-Agent System for Resource Selection: cloud service selection using an adaptive learning method developed by Wang et al. optimizes resource selection dynamically and gives a consolidated solution to the customer [74]. It executes many resource selection algorithms to select the best matching resources from the resource pool. The customer agent accepts the resource request, decomposes the submitted request into distinct categories of sub-request, and sends to an intermediate agent. Adaptive learning algorithms are used by the intermediate agent to perform resource selection and return selected resources to the customer agent for integration. Service providers register with federation through cloud service agent and submit the resource availability information. Cloud service agent monitors the resource performance and records performance after transferring.
2.2.1.2 Peer-to-Peer Resource Sharing in Federated Cloud

Enables peers to share resources among themselves without any centralized agent or third party. The following section describes various techniques used in peer-to-peer resource sharing in the federated cloud.

Resource Sharing in the Cross-Federated Cloud: cross-cloud federation (CCF) developed by Celesti et al. is a cloud testbed for horizontal federation [15]. CCF supports IaaS service providers to expand resource provisioning capabilities dynamically using Peer-to-Peer (P2P) approach. It consists of two clouds namely home cloud and foreign cloud. Home cloud makes resource request to the foreign cloud. The cloud manager in uses three agents such as discovery agent, matchmaking agent, and authentication agent for resource sharing. Discovery manager is responsible for preparing and publishing message across cloud providers. It uses “publish and subscribe” technique to acquire resources from other clouds. Match-making manager is responsible for selecting a correct resource, manage, and enforce policies of the cloud providers. The authentication manager creates, implements, and ensures security policies among cloud providers. It uses Single Sign On (SSO) technique to implement authentication in CCF.

Resource Sharing Using Incentive Technique: Cluster-Based Incentive Mechanism (CBIM) proposed by Zhang and Antonopoulos is an incentive-based resource sharing method [84]. CBIM creates resource providers cluster to share resources for servicing a resource request. The consumer has to initiate a transaction by sending a resource request query to all the providers in the cluster. The query consists of consumer id, address, resource request, and resource provision. The consumer has to share the same amount of other types of resource to other cloud providers in the cluster. uses the Eulerian graph forming technique to form a cluster. Once the cluster is formed, the resource sharing starts. The reputation value of the consumer is calculated based on the past resource sharing history. Each consumer is responsible for calculating the reputation of the other cloud providers in the ring before agreeing to share resources.

Resource Sharing Using Indirect Reciprocity: Fairness Driven Networks of Favor (FD-NoF) approach proposed by Falcao et al. enhances peer-to-peer cooperation while sharing resources with peers [26]. Cloud providers increase resource sharing capacity based on their resource sharing fairness value (using transaction history). A resource consumer in the group begins the transaction by sending a resource request to all the other members in the group. All the members of the group maintain transaction record of other members. The fairness value is obtained by considering the past resource contribution of the consumer. Satisfaction value is obtained by the amount of resource requests by the consumer and available resource with the provider. The “fairness value” and “satisfaction value” are used to process the resource request from the customer. Based on the calculated “fairness value” and “satisfaction value” providers share resources with other members. Finally, the
transaction record is shared to the members of the group for maintaining history.

Multi-Tenant Based Resource Sharing: Reciprocal Resource Fairness (RRF) based resource sharing model proposed by Liu et al. enables fair resource sharing among multiple IaaS cloud customers [49]. RRF is a measure of the ratio between amounts of resources shared and amounts of resource consumed by the customer in a federation. Inter-tenant Resource Trading (IRT) is a resource trading technique which allows the exchange of resources among the customers. RRF uses IRT technique to share resources among customers. In RRF, initially, customers acquire the required resources from many cloud service providers. The acquired resources may be surplus or insufficient to meet the customer’s actual requirement. Then, customers make a trade of surplus resource to meet an insufficient resource using IRT technique. RRF guarantees fairness in IRT trading (resource sharing). Intra-tenant Weight Adjustment (IWA) distributes workloads evenly to all the resources to avoid overloading a particular resource. Table 2.1 shows the comparison of centralized and peer-to-peer resource sharing techniques.

<table>
<thead>
<tr>
<th>Federation Type</th>
<th>Techniques</th>
<th>Strengths</th>
<th>Challenges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Centralized</td>
<td>CRBS</td>
<td>Easier service discovery</td>
<td>Implementation complexity</td>
</tr>
<tr>
<td></td>
<td>MAS based e-barter system</td>
<td>High security</td>
<td>Free riders</td>
</tr>
<tr>
<td></td>
<td>Semantic e-barter system</td>
<td>Trust worthiness</td>
<td>Ensuring fairness</td>
</tr>
<tr>
<td></td>
<td>Adaptive learning mechanism</td>
<td>Easier transaction maintenance</td>
<td>New comers</td>
</tr>
<tr>
<td>Peer to Peer</td>
<td>CCF</td>
<td>Quality of service incentive for resource sharing</td>
<td>Finding free resources</td>
</tr>
<tr>
<td></td>
<td>CBIM</td>
<td>Avoids free riders</td>
<td>Maintaining transaction record</td>
</tr>
<tr>
<td></td>
<td>FDNoF</td>
<td>Less implementation complexity</td>
<td>Security</td>
</tr>
<tr>
<td></td>
<td>RRF</td>
<td>Motivates new comers</td>
<td>Trust worthiness</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Difficulties in ensuring quality</td>
</tr>
</tbody>
</table>
2.2.2 Resource Sharing in Multi-Cloud

2.2.2.1 Agent-Based Service Selection Approaches in Multi-Cloud

Self-coordinating multi-agent based service selection model proposed by Gutierrez-Garcia et al. collects services from multiple service providers and consolidates them as a single service [32]. It uses multiple agents to orchestrate, coordinate, and control services from many service providers. Service Composition Table (SCT) consists of information such as participants list, services list, services status, and agents list. Semi-recursive contract net protocol (SR-CNP) augmented with a service composition table technique is proposed in this system. Every agent has its own SCT table. The Customer Agent (CA) prepares service request using the service ontology and sends to the Brokering Agent (BA). BA refers the SCT table for service availability and sends the "request for proposal" message to the Service Provider Agents (SPAs). SPA checks the resource status (available, busy, and failed) in consultation with the Resource Agents (RAs) and sends the reply back to the BA. BA manages all the SPAs communications, makes services from multiple SPAs as a single service, and gives it to the customer.

Focused Selection Contract Net Protocol (FSCNP) proposed by Kwang Mong Sim gathers services from many service providers and makes as a single service [65]. FSCNP uses many agents to discover, find similarities, filter, and select services from many providers. Cloudle is a search engine based on MAS for finding services in the cloud. Customers request their services using Cloudle. The customer request consists of functional requirements, technical requirements, and cost of the services. The Service Discovery Agent (SDA) performs the similarity matching using the keyword taken from the request. Then, the matched service is selected by the SDA. Cloud crawler is an agent which accumulates services from many providers. SDA uses cloud crawler to aggregate service provider information. BA determines offered price using two techniques: (1) market-driven negotiation strategy, and (2) bargaining position gaining strategy. Finally, BA selects the service provider based on the best offer and sends the details to the customer.

Ontology-based service discovery: an ontology-based recommendation approach for matching service provider proposed by Kang et al. helps customers to find suitable service provider thorough intelligent brokering system [25]. Cloud ontology is a concept used to find similarities among cloud services. The customer submits a service request to the system using CA. BA uses cloud ontology to find similarities between requested services and available services. BAs use three types of similarities (concept similarity, object property similarity, and database type similarity) to select the most suitable service providers from the providers’ list. Initially, BA selects services based on service availability. Secondly, it evaluates selected services using similarities relationship. Thirdly, it removes services which have low similarity value from the initially selected services. Finally, the selected service is sent to the customer, if BA selects services successfully, else BA send a service
request to the other BAs which are capable of connecting to more service providers.

2.2.2.2 QoS-Based Service Selection in Multi-Cloud

Quality of service (QoS) and customer satisfaction level vary depending upon the hosting environment and availability zone. The above problem leads to uncertainty in QoS and user satisfaction levels. Kernel Density Estimation (KDE) based recommendation system developed by Haithem solved this uncertainty problem when services are hosted in multi-cloud [55]. KDE is an estimation method used to predict the probability of occurrence. Customer satisfaction level in each availability zone is computed and filtered using KDE. Then, the low-ranked service providers are eliminated, and the correlation between availability zones and services are examined using Formal Concept Analysis Clustering (FCAC) method. Finally, the service selection for the customer is made based on customer ratings and probability of suitability of cloud zone.

Genetic Algorithm (GA) based service selection method proposed by Li Liu optimizes service selection as stated in customer service request [86]. GA uses an incremental approach to obtain the most favorable solutions in wide range spaces. The fitness value is a measure of how good is the selected solution to the problem. GA-based approach applies in three steps: (1) selection, (2) crossover, and (3) mutation to filter services based on fitness value and eliminating poor services. Initially, services are ranked based on fitness value, and then the selection process selects the top-ranked fixed number of services. Next, a pair of services is selected randomly to form service composition and categorized into higher and lower services. Finally, GA chooses a random point to generate better service composition. Table 2.2 exhibits techniques for resource sharing and comparison of agent-based service composition and QoS-based service composition.

2.3 Social Cloud

The advancement in the Internet of Things (IoT) is reconstructing our times into a cyber-physical-social hyperspace and altering what it intends to be social. Prominent cases include smartphones, tablets, and all sorts of wearable gadgets, which are uniting people, both directly and indirectly, through several applications and social networks such as Twitter, Facebook, and WeChat. People tend to establish new connections every day, and it forms a social network. extend the high potential for fostering collaboration among individuals and among groups. This possible collaborative atmosphere is not only suitable for entertainment, but can also contribute significant value to various
TABLE 2.2
Agent-Based Service Composition Versus QoS-Based Service Composition

<table>
<thead>
<tr>
<th>Selection Criteria</th>
<th>Techniques</th>
<th>Strengths</th>
<th>Challenges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent based</td>
<td>Self-coordinating MAS</td>
<td>Easier service discovery</td>
<td>Ensuring quality of services and customer satisfaction</td>
</tr>
<tr>
<td></td>
<td>FSCNP</td>
<td>Trust worthiness</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ontology based recommendation system</td>
<td>Transaction based price negotiation</td>
<td></td>
</tr>
<tr>
<td>QoS based</td>
<td>KDE based recommendation system</td>
<td>Trust worthiness</td>
<td>Finding similar service</td>
</tr>
<tr>
<td></td>
<td>GA based service selection</td>
<td>More customer satisfaction</td>
<td>Selecting price worthy services</td>
</tr>
</tbody>
</table>

research societies. For this purpose, scientists are frequently employing social networking theories in projects to create groups, share knowledge, advertise their work, and communicate with their companions [11].

Concerning social networking, many techniques are proposed for resource sharing. The Automated Service Provisioning Environment (ASPEN) [9] is one of the methods that provide a combination of Web 2.0, social networks, and the cloud. ASPEN presents applications and shares data inside a business using a social network. It concentrates on the particular research communities which help based on the social features of collaboration. Therefore, the current social networks are possibly suitable platforms for sharing cloud resources in a more general way. There are numerous good examples of cloud and social networks serving synchronically, but in most cases, the cloud is merely entertaining the applications of social networks, for example, a user can develop a Facebook application hosted by Amazon’s AWS service.

The SC [11] assists individuals or institutions to share the capacity of their computing resources utilizing virtual machines (VMs) borrowed through the social network. Actors of the SC can share, demand, and utilize VMs from other actors. It is the virtual organization among the group members, but the modern work on the SC is not establishing a proper economic model for, such as incentive mechanisms for benefaction and various economic allocations [3]. The general architecture of the SC is illustrated in Figure 2.3.

There are two different economic models in the SC; firstly, the pricing models and secondly, the bartering models. The pricing models in the cloud resource sharing, deal with the cash for sharing the resources. However, in the case of the bartering models, there is no cash involved in the sharing process.
There is some work related to the creation of cloud and social network model. A paradigm is designed in [18] for an economic model and an auction-based model in the SC. This model is proposed for sharing storage resources in the cloud.

SC systems utilize financial models, such as auction, bartering, and pricing models. Financial models are classified into two fundamental divisions based on the sharing methods. The cloud industry has effectively used various pricing models for resource sharing [2], but bartering models are not popular because of the trust factor among judicious users. In massively distributed systems, several computational competent resource bartering models are developed in the existence of judicious SC actors. The method in [4] serves a community of colleagues to share cloud resources without including any currency exchange between them. Bartering methods are divided into three distinct classes called as TiT for TaT (BitTorrent) [58], Volunteer Computing [6], and Network of Favors (OurGrid) [8]. BitTorrent aides reducing the capacity on clogged cloud providers by tracing the actions of users. OurGrid encourages the group of people in site sharing. A site provides unused resources based on the centralized preference.
The SC model grouping the concepts of cloud, volunteer computing, and social networking is discussed in [19]. This work explains the resource trading of storage space among friends in Facebook. The problem of freeloading is discussed with the help of a credit-based dealing strategy where the users may rent their resources with a particular user in the social network based on the posted price market, or compete within an auction market. The forum formed around a social group encourages a method of trading, exchanging of resources, and various concurrent auctions [33]. The work by the authors of [18] created a dynamic cloud environment by using the mutual connection naturally available in the social network. The social marketplace concept proposed in the work by the authors of [19] regulates sharing demonstrated social storage cloud implemented in Facebook. Comparison of existing works with various factors is given in Table 2.3.

2.4 Mobile Cloud

In recent times, there is a massive growth in wireless communication networks, where various models of smart mobile devices and lots of smart apps are in use. Mobile devices face the power and resource limitation problems because of energy-hungry and computation-intensive apps. The cloud resources in this context are not only engaged in computation, but also processing, storage, and communications [87]. Despite the rapid growth in the hardware technologies, the problem of resource scarcity persists due to the increasing computation need of the users. Meanwhile, cloud technologies are moving fast in giving an excellent experience to the users in virtualization and sharing the resources. Solving the problem of resource limitation in the mobile devices owns many works by researchers which discuss the well-organized computation offloading mechanism for portable devices to remotely available cloud resources or closely available computing resources called cloudlets. The general computing architecture with cloudlet is depicted in Figure 2.4. There are few of the literature which discusses the various techniques of mobile cloud computing (MCC) [29,41,77].

MCC can play an essential role in providing access to a mobile device for offloading considerable computational responsibilities to the connected cloudlet servers. It ensures the QoS demands of the mobile device users. However, the connection between the mobile access point and the mobile device is slightly irregular with the unstable signal strengths. Additionally, the resource heterogeneity of the cloudlet and the massive data application requirements creates further problems in running the best possible code [24].

Some of the related works that have proposed to support the MCC are given in this section. The works by the authors of [10,29,48,56,59,90] discuss the idea of clustering mobile devices and linking them to the remote cloud.
### TABLE 2.3
Related Work Comparison

<table>
<thead>
<tr>
<th>Methods</th>
<th>Resource</th>
<th>Uses</th>
<th>Integration</th>
<th>Allocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Social Compute Cloud [14]</td>
<td>Compute</td>
<td>Authentication, social graph extraction</td>
<td>Facebook API, clearinghouse, Django Social Auth plugin</td>
<td>Matching the preference</td>
</tr>
<tr>
<td>Social Storage Cloud [19]</td>
<td>Storage</td>
<td>Authentication, social-graph extraction</td>
<td>Integrated Facebook application</td>
<td>Economic - pricing or auction</td>
</tr>
<tr>
<td>Social Content Delivery Network (S-CDN) [43]</td>
<td>Storage</td>
<td>Authentication, and social-graph extraction</td>
<td>Facebook API and coauthorship network</td>
<td>Social network analysis</td>
</tr>
<tr>
<td>FriendBox [30]</td>
<td>Storage</td>
<td>Authentication, and social-graph extraction</td>
<td>Facebook API</td>
<td>Allocate equally among friends resources</td>
</tr>
<tr>
<td>Subdivision Social Cloud [3]</td>
<td>Compute</td>
<td>Authentication, social-graph extraction</td>
<td>Integrated Facebook application</td>
<td>Economic - bartering</td>
</tr>
<tr>
<td>Social Cloud [57]</td>
<td>None</td>
<td>Social-graph extraction</td>
<td>None</td>
<td>Scheduling-based model</td>
</tr>
<tr>
<td>Cycle Sharing in Social Networks (CSSN)[9]</td>
<td>Compute</td>
<td>Social graph extraction, create a social network for message transport</td>
<td>Facebook API</td>
<td>Social postings</td>
</tr>
</tbody>
</table>

(Continued)
<table>
<thead>
<tr>
<th>Methods</th>
<th>Resource</th>
<th>Uses</th>
<th>Integration</th>
<th>Allocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-community-cloud collaboration (MC3) [9]</td>
<td>Compute</td>
<td>Collaboration across community clouds with social networks</td>
<td>None</td>
<td>Social network analysis</td>
</tr>
<tr>
<td>CONFINE [62]</td>
<td>Compute</td>
<td>Ad hoc extension to existing network topologies</td>
<td>None</td>
<td>Scheduling-based techniques and community economies</td>
</tr>
<tr>
<td>Social Community Clouds [62]</td>
<td>Compute/data</td>
<td>Authentication and social-graph extraction</td>
<td>Federated cloud scenario using the CometCloud system</td>
<td>Economic (price/reputation)</td>
</tr>
<tr>
<td>Cybernetics Social Cloud [16]</td>
<td>Compute/data</td>
<td>Social-graph extraction</td>
<td>Facebook Graph API, hybrid Cloud, and BOINC</td>
<td>Resource sharing between communities.</td>
</tr>
<tr>
<td>Facebook-based cloud resource sharing [4]</td>
<td>Compute</td>
<td>Authentication, social graph extraction</td>
<td>Container based CRB model- Facebook API</td>
<td>Economic (bartering)</td>
</tr>
<tr>
<td>Altruistic Device Sharing [37]</td>
<td>Data/Power</td>
<td>Authentication, social-graph extraction</td>
<td>Mobile network users and their social closeness</td>
<td>Economic (bartering)</td>
</tr>
</tbody>
</table>
Fernando et al. [28] tested the scenario of MCC to utilize local resources to resolve the resource scarcity. A concept of cooperative computing platform named as “Transient cloud” is projected in [56]. This work allows close-by devices to form a network and offer a variety of resources as a service to the actors in the system—tasks assigned to the devices done by the Hungarian technique. Li et al. introduced a best possible solution to distribute the functions to the devices. Zhou et al. [90] designed a model that considers several cloud resources like mobile ad hoc cloud, and cloudlet to provide a MCC. Recently, mobility aware best possible resource provision method, Mobi-Het proposed for remote data-intensive process execution in the mobile cloud with high efficiency and reliability [24]. A broker-based double-sided bidding mechanism for the mobile cloud is introduced in [68]. Few of the use cases of MCC are Smart City [24] and Health Care [35].

2.4.1 Incentive Mechanisms in Mobile Cloud

Although the MCC previously discussed is promising, the proper incentive mechanism encourages the users to share their resources. The following incentive methods are available for MCC: (1) mechanism, (2) bidding, and (3) crowdsourcing. The literature for the incentive mechanisms is given in this section. A fundamental method for auction method is Vickrey-Clarke-Groves (VCG) auction [31,60,73]. There are lots of variation emerged from this primary method, for example, Vickrey-based double auction [60]. McAfee proposed a double auction method [53]. There are few other methods like truthful double auction mechanism (TASC) [80], mClouds [56], CellCloud [59], optimal relay assignment algorithm [63], the maximum weighted matching
algorithm [75], and the incentive mechanism based on all-pay auctions [51] proposed in the literature. Double-sided bidding method for mobile cloud resource allocation is proposed in [68]. Incentive mechanisms using crowdsourcing technique is referred in [20,42,51,81]. Finally, some of the resource allocation optimization algorithms are studied in optimal relay assignment algorithm [63], maximum weighted matching algorithm [75], platform-centric model and user-centric model [81], and optimal incentive mechanism [42]. Thus the mobile cloud resource sharing has lots of open market in the future to the end users.

2.5 Cloud Manufacturing

Cloud intruded into every field. In the manufacturing world, there is a constant pressure to reduce the cost, improve the quality, optimal usage of manufacturing resources, and reduce the impact on the environment. The adoption of has brought many innovations in the manufacturing field. Cloud Manufacturing (CM) enables us to share and use resources optimally with the help of resource sharing. The architecture diagram for CM is shown in Figure 2.5.

CM, irrespective of IT resources and manufacturing resources, and based on its ability, can be classified into seven types which are shown in Figure 2.6 [69].

2.5.1 Resource Sharing in CM

Different manufacturers collaborate to share their capabilities and resources. Two different approaches are used to schedule the jobs when multiple jobs $j_1, j_2, \ldots, j_n$ are placed in the group of machines $m_1, m_2, \ldots, m_n$: (1) perform the jobs in the same order; (2) jobs performed in any order depending on the availability of resources. It is solved by (GA) using various arguments, such as processing time needed, material setup time needed, etc. A service-oriented cloud-based architecture is proposed to solve flow shop scheduling [34].

Complex network-based Cloud Manufacturing Platform (CMP) diffusion model by the authors of [17] was proposed to improve resource sharing. CMP uses theoretical analysis, simulation, and evaluation to match the supply and demand, which facilitate better resource sharing.

In the CM scenario, the resource requester has different choices of providers. The Gale-Shapley algorithm is used to pair the resource requester and provider based on preference degree. A utility function calculates the preference degree based on, price, execution time, reliability, availability, etc. The Gale-Shapley algorithm uses a preference metrics to match the providers [50].

CM tasks are of two types: (1) single resource-based task; and (2) multi-resource based task. Optimal service selection in single resource-based task
is simple and straight. In a broad scale, Service Composition and Optimal Selection (SCOS) for multi resource-based task with various constraints and objectives is difficult. Intelligent algorithms are solving multiobjective problems, but they increase the number of iterations and fails to provide efficient resource sharing. FC-PACO-RM, a parallel algorithm [70] is proposed to provide efficient resource sharing decision.

Data-driven smart manufacturing proposed [71] to improve the resource utilization among SMEs. Throughout the product life cycle, a massive amount
of data get generated. The proven data analysis techniques are used to envision better resource sharing. Also, it supports proactive maintenance and quality control by learning.

In CM usage of robots is inevitable for specific tasks. Robots need to be positioned into the specific areas to carry out the assigned tasks. Task still is challenging in CM robots. The optimal methods, such as Load Balance Scheduling (LBS), Cost Minimized Scheduling (CMS), and Delay Minimized Scheduling (DMS) provide optimal performance, cost, and processing time. Location of the robot influence in the overall cost. A CM scheduling model is proposed for better resource sharing using the optimal deployment of robots and task scheduling methods. The Location Aware Deployment (LA-DP) reduces the transportation cost, and Function Balanced Deployment (FB-DP) reduces the overall processing time [47].

In CM paradigm the traditional approaches for focus on optimal resource selection. The optimal resource sharing is adopted to achieve operational efficiency. The hybrid energy-aware resource allocation algorithm proposed in [88] to support the demanders uses energy efficient manufacturing. It follows three steps to achieve it. First, it uses fuzzy similarity degree for selecting correct providers based on QoS and then filters the providers based on customer constraints. Second, the non-dominated sorting genetic algorithm (NSGA-II) optimizes the selection based on QoS and energy awareness. Lastly, TOPSIS algorithm is used to select the optimal provider.

Integration of cloud with the manufacturing never changes the dynamic behavior of the cloud. The traditional resource sharing models have a specific problem called semantic heterogeneities. A semantic model is proposed to match the dynamic nature of the cloud. The semantic model enables the flexible access and manipulation of resources [78].

In a manufacturing environment, the production cycle goes like a. The workflow needs a resource to finish a task, and that resource may not be available in a real production environment when it is in a shared environment. The resource sharing and utilization improved if that is managed well. Most existing methods have failed to use the temporal relationship between the resources, which creates an in-efficient utilization. Hence a Resource Service Chain Composition Algorithm (RSCCA) was proposed [46] to build a temporal relationship among the resources to improve resource utilization.

The primary challenge in CM is envisioning the intelligent resource perception and access. IoT plays a significant role in performing intelligent-perception in the shared environment. Identification of various manufacturing resources, services, and their relationship is essential in resource management. IoT-based framework with five layers was proposed for intelligent-view and access to resources of manufacturing [72]. The intelligent-resource perception helps to identify the present state of a resource, which improves the efficiency in sharing.

Mass production reduces the cost of a product in manufacturing. The cloud 3D printing system integrates various 3D printing facilities across the
city to reduce manufacturing cost. The cloud 3D receives the order from the
demander and distributes it to multiple locations. Location-aware printing
helps to finish the work effectively. The existing methods have two problems:
(1) to minimize the makespan; and (2) shortest total length path. An agile
and ubiquitous additive manufacturing system has been proposed to address
the problems mentioned above [21].

2.6 Vehicular Cloud

Vehicle employed with smart devices forms vehicle ad hoc networks (VANET).
An enlargement of VANET is VC. Vehicle cloud uses resources like processing,
storage, and cyberspace to take a decision which improves traffic management,
smart parking, and road safety. Amazingly the high-tech cars are using
VCC. The idle resources in the VCC are pooled together and used when their
resource demand goes high [54].

2.6.1 Vehicular Cloud Architecture

VCC architecture is visualized as a layered architecture. The VCC architec-
ture is majorly classified into three primary layers, namely inside-vehicle,
communication, and cloud [76]. The VCC architecture is shown in Figure 2.7.

2.6.1.1 Inside-Vehicle Layer

The inside-vehicle layer uses smart sensors to observe the health, mood and,
STP (Standard Temperature and Pressure) inside the vehicle. It also observes
the driver behavior to find his or her unconsciousness and plans. Gathered
sensor data is sent to the cloud for further processing.

Roadside units (RSUs) and On-Board Units (OBUs) have transceivers and
transponders. The RSU manages OBUs of traveling vehicles via the DSRC
technology. DSRC system acquires the needed traffic information, speed, and
vehicle location in an instant time. The OBUs transport data through mobile
devices.

2.6.1.2 Communication Layer

The communication layer has two components: (1) the vehicular-to-vehicular
(V2V) systems via DSRC, and (2) vehicle-to-infrastructure (V2I). V2V recog-
nizes an unusual behavior like exceeding the speed limit, shifting of direction
dramatically, or detecting a mechanical failure in the vehicle. On detecting
unusual behavior, an alert message is sent to the cloud and nearby vehicles.
Alert messages include the geographical location, acceleration, direction of
the vehicle, and speed. layer transfers the functional data among vehicles,
infrastructures, and the cloud over 4G, internet, or satellite. It ensures vehicle safety on roadways automatically.

2.6.1.3 Cloud Layer

The cloud layer has two components: (1) platform, and (2) application services. Government and related agencies create reports and perform investigation using the data stored in the cloud. The cloud platform has two subcomponents: (1) storage, and (2) processing. Data collected by IVL is moved to the storage and compute component performs a task for effective management of traffic, smart parking, and multimedia services. Application services component incorporate Real-Time Application (RTA) and Cloud Primary Application Services (CPAS). Vehicle drivers access RTA to identify inside and outside human activity with the help of CPAS. CPAS comprises of services, such as NaaS, CaaS, STaaS, INaaS, and ENaaS [45,89].

The components of the CPAS layer are described in Table 2.4.
Techniques for Resource Sharing

TABLE 2.4
Primary Application Service Components

<table>
<thead>
<tr>
<th>Components of CPASL</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>NaaS</td>
<td>Shares the internet with other vehicles</td>
</tr>
<tr>
<td>STaaS</td>
<td>Storage for vehicles when additional storage needed to run their applications over a network</td>
</tr>
<tr>
<td>CaaS</td>
<td>Provides a list of services such as traffic information, road condition, driver safety, and parking availability</td>
</tr>
<tr>
<td>Computing</td>
<td>Aggregate the idle computer and storage among parked vehicles and offer it as a service to the demander</td>
</tr>
<tr>
<td>Pictures on a wheel</td>
<td>Based on the customer demand takes the photo or video of the landscape and share it</td>
</tr>
<tr>
<td>INaaS</td>
<td>Set of information such as road conditions, emergency situation, etc. shared to the drivers for a safe journey</td>
</tr>
<tr>
<td>ENaaS</td>
<td>Advertisements and promotions delivered in the screen of the car as like beside the road</td>
</tr>
</tbody>
</table>

2.6.2 Resource Sharing in Traffic Management

Good traffic management leads to smooth transportation. Because of the increase in number of cars on the streets, traffic management became a complicated process. One of the most important things to overcome this problem is to share required resources rather than allocating excess resources, hence utilization of the resources can be improved [52].

Consider the following scenario, a cricket match is scheduled in the city. Many people enter and leave the city to watch the match simultaneously, which causes a traffic jam. Various investigations are carried out to address the above problem based on VANETs and IT system, but failed to provide a traffic moderation plan. Hence, it is necessary to find a solution, which enables the vehicles to pool resources and create a powerful super-computer that, reschedules the traffic lights that regulate the traffic without congestion [23].

2.6.3 Resource Sharing in Multimedia Services

The driving recorder is used in Intelligent Transport Systems (ITS) to record the video of the vehicle. The recorded videos are pushed to the cloud in real time. Pushing multiple videos from multiple Driving Recorders Video (DRVs) creates a competition for the wireless connectivity for uplink. A solution to the above problem comprises of two objectives: (1) assignment of bit rate, and (2) real time slot assignment. The objectives (1) and (2) solved by utility-based priority function was proposed in [39].
2.6.4 Resource Sharing in Smart Parking

(SPS) play a vital role in smart cities. Effective management of traffic and parking increase the satisfaction of commuters. The intelligent car parking systems are getting a lot of recognition and momentum.

The Smart Car Parking System (SCPS) directs vehicles toward the best path for parking in the city using vehicular road sensing and intelligent display. VCC mixes MCC, and vehicular networking (VN) to create better communication and computing. The applications like Netflix is used in vehicular clients. Also, it is used in SCPS. However, the streaming nature of Netflix-like apps brings jitter and delay, which, affects the SPCS. By using technology advancements like fog and edge, the SPCS solves the traffic-related issues [38].

To reduce the parking-related problems, SPCS checks for a vacant slot and then directs the vehicle to the parking if there is a space. Car theft is avoided by using a fingerprint scanner and camera in the parking lot. Introduction of fog improves the of SPCS [7].

2.6.5 Vehicle-Assisted Cloud Computing (VACC) for Smartphones

Even though standard resource exists in the smartphones to process its applications, it needs more processing power based on the apps used such as gaming and mobile data analytics. The sudden resource demand for a smartphone is provided by the Vehicle assisted cloud computing (VACC).

VACC comprises of a central cloud, a cloudlet, and a VC. In VACC architecture, MAN connects the components which need network connectivity. The VC is connected to a cloudlet for resources. Smartphones are connected to the cloudlets through mobile networks or Wi-Fi. The cloud nodes are heterogeneous based on their capacity and resources. The central cloud has large storage and high computational power and less communication rate because it is connected to WAN. The cloudlet has high communication rate, less storage, and less computation [67].

When a smartphone needs an extra resource for its application first, it tries to get from VCC. If VCC does not have the required resources, next, it tries to get from the cloudlet. If this also fails, then it gets the resource form the central cloud [83].

2.6.6 Resource Sharing in Vehicular Fog Computing

The smart vehicles possess decent devices for computation and communication. Vehicular Fog Computing (VFC) is a collaboration of distinct nearby edge devices to perform computation and communication. The nearby moving vehicle can communicate and share the resources with each other to form a
Techniques for Resource Sharing

TABLE 2.5
Comparison of Resource Sharing Techniques in VCC

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Resource Utilization</th>
<th>Response Time</th>
<th>Integration</th>
<th>Security/Privacy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traffic Management in VCC</td>
<td>Medium</td>
<td>High</td>
<td>Complicated</td>
<td>Medium</td>
</tr>
<tr>
<td>Multimedia Services</td>
<td>Low</td>
<td>More short</td>
<td>Easy</td>
<td>Low</td>
</tr>
<tr>
<td>Smart Parking</td>
<td>Medium</td>
<td>Short</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>Vehicle-Assisted</td>
<td>Low</td>
<td>High</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>Vehicular Fog</td>
<td>Very High</td>
<td>High</td>
<td>Complicated</td>
<td>High</td>
</tr>
</tbody>
</table>

good computing power. The parked vehicles can also establish communication for computation [5,36,85]. Table 2.5 shows the comparison of resource sharing techniques in VCC.

2.7 Green Cloud

Green cloud computing enables us to use computer resources in an environmentally responsible way. It is possible by redesigning the data center to minimize power consumption, resource utilization, and the usage of natural resources. Cloud data centers are also adopting green computing to save energy, reduce cost, and save environment. In Computing (GCC), plays a vital role [61].

2.7.1 Resource Sharing in GCC

In cloud data centers thousands of physical machines are used to power up user’s VM. consolidation is the process of moving the VMs to the few physical machines, which reduces the consumption of energy. In cloud data center VM consolidation is used. Practically the VM consolidation is an NP-hard problem. Ant Colony based VM consolidation minimizes the energy utilization while maintaining the [27].

The main focus of GCC is using less energy in cloud data center. A clonal selection algorithm [64] proposed to reduce the energy consumption, time, and cost. A scheduling algorithm based on multi-step heuristic proposed in [13] to reduce the CO₂ emission and improve the resource utilization. When the customer request for a resource with a deadline, the request is assigned to one
of the data centers, which has less environmental effects without compromising the deadline.

Adaptive energy-aware VM migration technique proposed by Zhang et al. decreases energy utilization and service level agreement (SLA) violation in green cloud computing [79]. The data center performance increases because the technique proposed by Zhang et al. takes four parameters (number of host shutdowns, number of SLA violations, number of VM migration, and energy consumption) while selecting a host for placing new VMs. For the placement of new VMs, it monitors the data center and saves energy. The technique mentioned above uses a central controller (CC), local controller (LC), and VM controller (VMC). LC monitors CPU utilization and VM in the local host and sets any one of the statuses such as under-load, overload, and normal. CC collects statistics from LCs, maintains the overall status of the data center, and has a plan for deployment of the new VMs. VMC migrates and resizes the VMs. The earlier energy-aware VM selection techniques do not consider SLA violation and network bandwidth that reduces the performance of the data center.

Locust-inspired scheduling technique proposed by Kurdi et al. decreases energy consumption by applying Locusts behavior [44]. It places new VMs on the host by executing three phases (mapping, consolidation, and migration). In the mapping phase, an entered VM request is received and assigned to the server with enough resources. Consolidation phase consolidates VMs for reducing the count of servers active. During the migration phase, VMs from underload server is moved to other servers for reducing the number of running servers. LACE algorithm is applied in both mapping and consolidation phases.

2.8 Resource Scheduling Using Meta-Heuristic Techniques

Many works have been done in the area of resource scheduling using meta-heuristic techniques. There are many types of meta-heuristic techniques used in resource scheduling in the cloud computing, such as the Ant Colony Optimization (ACO), Genetic Algorithm (GA), and Particle Swarm Optimization (PSO). ACO is used to optimize the selection of pooled resources. It is mainly used in the VC and SCC. The GA is used to select the quality service providers based on QoS parameters. It is majorly used in task distribution in the CM. The PSO is majorly used in workflow processing and service selection in the cloud. Application of PSO in CM improves the efficiency of the manufacturing process to the next level [40].
Future Scope

In this chapter, the benefits of resource sharing techniques in the cloud computing platform and challenges were discussed. However, some open issues exist that can be a beginning point for future research work. In an environment, selecting a trustworthy and high-quality provider, neglecting selfish providers, and creating transaction histories for new providers are the real challenges. Artificial intelligence techniques can be used to get better performance than using traditional federation management techniques. Numerous difficulties emerge while moving from a cloud model to a SC model. Ensuring trust and long-term service in the SC is a challenging task. Scalability, maintenance, privacy, and security are the other concerns in the SC. Incentive-based resource sharing can solve durability and trustworthiness. Currently, extensive research is going on the areas mentioned above. Open problems are present in the MC resources sharing techniques that can be addressed by solving congestion issues in mobile communication and improving the energy-efficient transmission. The future CM can be constructed by the sustainable manufacturing process to reduce the impact of climate and energy. CM can be improved by incorporating efficient, automated, energy-aware techniques which maximize the production and minimize the waste at a reduced cost. In VCC, many areas are still unexplored for researchers such as security, privacy, responsive resource sharing, highly reliable data access, the mobility of resources sharing, unbalanced message links. Further, VCC can be extended to intelligent VC computing like driverless vehicles, machine learning based traffic control and route prediction system. Scheduling and resource management play a vital role in GCC. In GCC, environment-aware computing is made possible by energy-aware VM allocation, scheduling, and resource management. Incorporating machine learning techniques in scheduling and resource management can bring better results.

Summary

This chapter states the different resource sharing techniques. It compares various resource sharing techniques from the literature and compares their strengths and challenges. Moreover, the revenue generation models for resource sharing are discussed. The use cases of cloud resource sharing using the mobile, social network, and vehicular network are elaborated. Also, the effect of the green cloud for sustainable development is also discussed.
Bibliography


Techniques for Resource Sharing


3

Swarm Intelligent Techniques for Cloud Service Provider Selection in a Multi-cloud Environment

Amany M. Mohamed
Cairo University

Hisham M. Abdelsalam
Cairo University

CONTENTS
3.1 Introduction ................................................................. 74
3.2 Related Work ............................................................... 75
3.3 Problem Description ..................................................... 76
  3.3.1 Context ................................................................. 76
  3.3.2 Model Formulation .................................................. 76
  3.3.3 Decision Variable ................................................. 77
  3.3.4 Objective Functions .............................................. 77
3.4 Solution Algorithms ...................................................... 80
  3.4.1 Solution Evaluation .............................................. 81
  3.4.2 Genetic Algorithm (GA) ......................................... 82
  3.4.3 Particle Swarm Optimization (PSO) ........................... 85
  3.4.4 Harmony Search (HS) .......................................... 87
3.5 Numerical Experiments .................................................. 89
3.6 Algorithms’ Performance .............................................. 93
3.7 Conclusion and Future Work .......................................... 99
Appendix A: Example of Case Study Data ............................. 99
Bibliography ................................................................. 102
3.1 Introduction

Provider selection is one of the most imperative decisions when dealing with cloud computing [1,2]. Comparing providers and deciding which provider is the best represents an important issue to the client/customer and providers as well. Client concerns about selecting provider that best fits its needs, while the providers need to benchmark themselves with others to improve their services [3]. This research focuses on cloud provider selection problem from the client/customer perspective.

Due to the growth of public cloud service offerings and its diversity, selecting the right provider that can satisfy the Quality of Service (QoS) requirements of the customer become an important challenge [4]. The diversity comes from that similar services are offered at different prices and performance levels. Consequently, three main questions should be answered when comparing cloud providers: (1) what are the criteria that will be used to compare cloud providers?, (2) how to measure these criteria?, and (3) how to rank cloud provider based on these criteria? [5].

Cost is considered as an essential factor for comparing cloud providers. Numerous research papers proposed several decision models that were mainly based on cost [3–10]. However, because of the diversity of cloud services offerings, in addition to the increasing number of cloud providers, customer faces a challenge of selecting cloud provider considering various criteria. Consequently, research papers tried to identify the main criteria from customer’s perspective [10,11]. In this context, the Cloud Service Measurement Index Consortium (CSMIC) has designed the Service Measurement Index (SMI) to provide a standardization method of measuring and comparing the business service [12].

SMI involves various categories that helps customers to compare different cloud service providers which lead them to consider multiple criteria through the provider selection process. Therefore, to address this problem, previously published research papers proposed multi-criteria decision models. The proposed models were solved using weighted methods, such as Analytic Hierarchy Process (AHP), Ranked Voting Method, or Technique of Order Preference by Similarity to Ideal Solution (TOPSIS). However, the main problem with these methods is the dependency on user experience and preferences which make the selection decision inaccurate. Moreover, weighted methods were used for the purpose of rank providers which means selecting a single provider to satisfy customer’s requirements (vendor lock-in problem). Consequently, the customer is forced to accept the provider’s capabilities and pricing.

Multi-cloud environment has attracted the attention of the recent research papers—a strategy that assists customers to avoid vendor lock-in problem. Therefore, this chapter presents a general model for selecting providers in multi-cloud environment considering any number of Infrastructure as a service (IaaS) based on two evaluation criteria: cost and performance. The problem
was formulated as integer programming and was approved to be a NP-hard problem. Consequently, to solve it, three meta-heuristic algorithms were used: Genetic Algorithm (GA), Harmony Search (HS), and Particle Swarm Optimization (PSO). In order to test and compare the performance of the proposed algorithms, a case study was generated. Results showed improved performance of the PSO compared to the GA and the HS.

The rest of this chapter is organized as follows: Section 3.2 provides related work, while Section 3.3 deals with problem description; it starts with context and proceeds with model formulation. The details of the solution algorithms are presented in Section 3.4. Section 3.5 discusses case study, while Section 3.6 tests the performance of the proposed solution algorithms. Finally, conclusion and future work are given in Section 3.7.

### 3.2 Related Work

Early papers cited, relevant to this research direction, focused on supporting customer/decision maker (DM) in evaluating only one cloud provider. Khajehhosseini et al., in [6] provided cost modeling that helped DM in obtaining precise cost estimates of running a server infrastructure on the cloud. Then, the idea was extended to include a group of providers; Khajehhosseini et al., in [7] extended their work discussed in [6] in order to use it to compare the costs of different cloud providers and different types of instances. Besides, Li et al., in [3] developed CloudCmp tool to compare cloud providers based on performance and cost and provide this information to the customer to select the right provider. However, all previous researches were restricted to a few cloud providers that dominate the cloud market and the common services offered by them.

As a consequence of the diversity of cloud services offerings and the increasing number of cloud providers, research papers proposed decision models in order to compare different providers considering multiple criteria. Various criteria are considered in the literature, but many of the research papers revolve around cost criteria [3–10]. Furthermore, based on SMI, Garg et al., [5] provided a framework to measure various SMI attributes and rank the cloud services based on it.

As a result of the continually increasing number of cloud service provider’s evaluation criteria, research papers depended on user experience and preferences to select the best cloud service provider. Analytic Hierarchy Process (AHP) was the most used method to rank cloud providers [4,5,8–10,14,15]. AHP is a flexible and powerful tool for dealing with complex decision-making. It depends on three principles: (1) decomposition of the problem, (2) comparative judgment of the elements, and (3) synthesis of the priorities. However, when a large number of criteria is considered, applying AHP becomes
time-consuming. Another weighted methods were used to solve the provider selection problem. Baranwal and Vidyarthi [16] and Shirur and Swamy [17] proposed Ranked Voting Method, while Upadhyay [18] proposed Technique of Order Preference by Similarity to Ideal Solution (TOPSIS).

All the previously published research work share three common shortcomings: (1) focusing on single-cloud provider perspective; the proposed models and frameworks tried to evaluate different criteria and rank different providers to select only one of them to satisfy customer’s requirements and this force the customer to accept the provider’s capabilities and pricing, (2) ignoring the constraints for each criterion, and (3) the solution of the problem was based on weighted methods. Recent researches tried to take advantage of multi-cloud to secure customer data getting customer’s data storage requirements from multiple cloud providers [19–22].

3.3 Problem Description

3.3.1 Context

Consider a set of autonomous cloud providers. Each provider offers various infrastructure services with different configurations. Each configuration is defined by different parameters. Besides, a client has a set of different infrastructure services to be rented. Based on the type of service, the parameters that defines each service can be numerical or non-numerical. The client needs to satisfy its requirements with minimum cost and maximum performance. The aim of this study is to select the optimal group of cloud providers for all clients’ requirements in order to minimize cost and maximize performance subject to a set of constraints.

3.3.2 Model Formulation

For the convenience of formulation, the parameters and variables for provider selection are defined as follows:

- \( S^{(R)} = S^{(R)}_i \quad |i=1...N \) a set of service requirements of client
- \( S^{(RP)} = S^{(RP)}_{ia} \quad |a=1,...,A_i \) a set of parameters that define required service \( i \)
- \( P = P_j |j=1,...,M \) a set of independent cloud service providers
- \( S^{(P)}_j = S^{(P)}_{k=1,...,K} \) a set of infrastructure service that offered by provider \( j \)
- \( S^{(PP)}_{kj} = S^{(PP)}_{bjk} \quad |b=1...B_{kj} \) a set of parameters that define service \( k \) that offered by provider \( j \)
Techniques for Service Provider Selection

\[ C_{ij} \] cost of required service \( i \) when it is rented from provider \( j \)

\[ \text{BU} \] budget of client

\[ \text{PF}_{ij} \] performance when rented requirement \( i \) from provider \( j \)

\[ \text{ART}_{ij}^{(P)} \] the provider \( j \) response time for the service \( i \)

\[ T_{uij} \] time between when user \( u \) requested for service \( i \) from provider \( j \) and when it is actually available

\[ n_{ij} \] total number of service requests of service \( i \) from provider \( j \)

\[ \text{ART}_{i}^{(R)} \] client’s maximum acceptable response time

\[ \text{Suit}_{ij} \] value of suitability for service \( i \) when rented from provider \( j \)

\[ F_{ij}^{(P)} \] number of essential and non-essential features provided by the service \( i \) at provider \( j \)

\[ F_{ij}^{(R)} \] number of essential and non-essential features required by the client for service \( i \)

3.3.3 Decision Variable

Given a set of requirements and the set of cloud providers, the decision to be make is which provider to select in order to satisfy each of clients’ requirements. So, the decision variable is formulated as integer as follows:

\[ X_{ij} = \begin{cases} 1, & \text{if required service } i \text{ is rented from cloud provider } j. \\ 0, & \text{otherwise.} \end{cases} \quad (3.1) \]

3.3.4 Objective Functions

1. Cost

Cost is one of the important attributes for IT and the business. It is the most quantifiable metric through which the organization can answer the question whether switching to cloud computing is cost-effective or not [5]. Nowadays, various pricing models/schemes are used by cloud providers, however, “pay-as-you-go” model is the foremost common model used in cloud computing [23,24]. In this study, the cost formulation is based on “pay-as-you-go” model. Majority of services have a constant coefficient of calculation because of the fixed relationship between unit-price and consumption [4]. However, the cost/price of some services will be changed with the increase of
consumption. In this context, a dynamic function will be used to calculate the service cost. For example, the Windows Azure storage service pricing policy for the Geographically Redundant Storage (GRS) option for the Central-US region with the defined cost of $0.01 per 100 K transactions is shown in Figure 3.1.

From the pricing details, we can notice that the unit-price of storage capacity is constant, but the cost will decrease with the increase of amount. Subsequently, it is necessary to set calculation, respectively for each amount range. Thus, given that $S_1^{(PP)}$ is the storage capacity in TB and $S_2^{(PP)}$ is the number of storage transactions, the cost can be formulated as follows:

$$
f \left( S_1^{(PP)}, S_2^{(PP)} \right) = \begin{cases} 
(0.048 + \frac{S_2^{(PP)}}{1,000}) + \frac{(0.01 \times S_2^{(PP)})}{100,000}, & S_1^{(PP)} \in [0, 1] \\
48 + \left(0.0472 \times \frac{S_1^{(PP)} - 1}{1,000}\right) + \frac{(0.01 \times S_2^{(PP)})}{100,000}, & S_1^{(PP)} \in (1, 50] \\
2360.8 + \left(0.0464 \times \frac{S_1^{(PP)} - 50}{1,000}\right) + \frac{(0.01 \times S_2^{(PP)})}{100,000}, & S_1^{(PP)} \in (50, 500] \\
\ldots \ldots & 
\end{cases}
$$

Consequently, based on the parameters of each service, the cost function can be formulated as follows:

$$C_{ij} = f \left( S_1^{(PP)}_{ij}, S_2^{(PP)}_{ij}, \ldots, S_k^{(PP)}_{ij} \right)$$

2. Performance

It is necessary for the clients to understand how well their applications will perform on the different clouds and whether these deployments meet their expectations [5]. Performance is one of the defined categories of SMI. It contains five attributes that are used to measure the features and functions of the provided services. In this

<table>
<thead>
<tr>
<th>Storage Capacity</th>
<th>GRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>First 1 TB / Month</td>
<td>$0.048 per GB</td>
</tr>
<tr>
<td>Next 49 TB (1 to 50 TB) / Month</td>
<td>$0.0472 per GB</td>
</tr>
<tr>
<td>Next 450 TB (50 to 500 TB) / Month</td>
<td>$0.0464 per GB</td>
</tr>
<tr>
<td>Next 500 TB (500 to 1000 TB) / Month</td>
<td>$0.0456 per GB</td>
</tr>
<tr>
<td>Next 4000 TB (1000 to 5000 TB) / Month</td>
<td>$0.0448 per GB</td>
</tr>
<tr>
<td>Over 5000 TB / Month</td>
<td>Contact us</td>
</tr>
</tbody>
</table>
study, two convenient attributes for IaaS services were chosen: Average Response Time (ART) and Suitability (Suit).

(a) Average Response Time
ART is a high-level measure that can be used to quantify service efficiency by calculating the time interval between the service request and the response of the service; how quick the service can be made accessible for utilization [12]. ART, for an existing service, can be calculated using the arithmetic mean (average) of the individual response times over a period of 6 months [12]. ART for required service $i$ when rented from provider $j$ can be determined mathematically using the following equation:

$$ART_{ij}^{(P)} = \sum_u \frac{T_{uij}}{n_{ij}}$$  \hspace{1cm} (3.4)

From the customer’s perspective, an acceptable maximum value for the response time need to be determined, thus the selected provider must meet this value. This constraint is formulated as follows: $ART_{ij}^{(P)} \geq ART_{i}^{(R)}$.

(b) Suitability
The client may have fundamental and unnecessary features. Suitability measure is used to answer a question, such as how nearly the provided service matches the clients’ needs? CSMIC determine a range from 0 to 10 to express the measure. Consequently, suitability of the required service $i$ that will be rented from the provider $j$ can be presented mathematically as follows:

$$Suit_{ij} = \begin{cases} 10, & \text{if all the essential features are satisfied} \\ \frac{F_{ij}^{(P)}}{F_{ij}^{(R)}}, & \text{if all essential features are satisfied and some} \\ & \text{of the non-essential features are not satisfied} \\ 0, & \text{if any of the essential features is not satisfied} \end{cases}$$  \hspace{1cm} (3.5)

This measure is customer focused [5] and needs to be performed at the initial selection of the cloud provider [12]. Zero is unacceptable value of suitability, so the suitability of provider must be greater than zero ($Suit_{ij} > 0$).

In the provider selection process, the provider with the minimum ART will be preferable and the provider with maximum suitability will be preferable. In order to sum the two measures, ART will be multiplied by $(-1)$. Given the weight of each attribute/measure, performance of the service $i$ when rented from the provider $j$ will be calculated according to the following equation:

$$PF_{ij} = w_1^{(r)} \times Suit_{ij} - w_2^{(r)} \times ART_{ij}^{(P)}$$  \hspace{1cm} (3.6)
where $w_1^{(r)}$ is the weight of suitability and $w_2^{(r)}$ is the weight of average response time attribute. Finally, the formulation of provider’s selection in multi-cloud environment is presented as follows:

$$\text{Minimize} \sum_{i}^{N} \sum_{j}^{M} C_{ij} \times X_{ij} \quad (3.7)$$

$$\text{Maximize} \sum_{i}^{N} \sum_{j}^{M} PF_{ij} \times X_{ij} \quad (3.8)$$

Subject to:

$$\sum_{i}^{N} \sum_{j}^{M} X_{ij} = 1 \quad (3.9)$$

$$\sum_{i}^{N} \sum_{j}^{M} C_{ij} \times X_{ij} \leq BU \quad (3.10)$$

$$\text{ART}_{ij}^{(P)} \geq \text{ART}_{ij}^{(R)}, \forall i \in N \quad (3.11)$$

$$\text{Suit}_{ij} > 0, \forall i \in N \quad (3.12)$$

$$X_{ij} \in 0, 1 \quad (3.13)$$

The constraint in Equation (3.9) ensures that each required service is rented from only one cloud provider. In Equation (3.10), the total cost cannot be larger than the client budget. In Equation (3.11), the provider response time must meet the client’s maximum acceptable response time. The constraint in Equation (3.12) ensures that the suitability of provider must be greater than zero. The last constraint ensures that $X_{ij}$ is a binary variable.

### 3.4 Solution Algorithms

The multi-cloud provider selection is similar to the multi-item vendor selection (MIVS) problem discussed in [25]; MIVS problem is a problem of procuring a number of different discrete items ($N$) from different vendors ($M$) under the assumptions that each item is bought from a single vendor and all vendors offer all items at different costs. MIVS problem is formulated as an integer programming model which is similar to the facility location problem, a well-known NP-complete problem. Given the assumption that each item is bought
from a single provider, and the number of candidate solutions is $M^N$. Consequently, solution space may grow into very large number with the increase in the number of vendors and the number of items to be purchased.

The presented problem is similar to MIVS problem; each of the IaaS requirements is considered as a discrete item and is rented from a single cloud provider. It is formulated as integer programming which is similar to the facility location problem. So, multi-cloud provider selection is NP-complete problem and to solve it three swarm algorithms were applied: Genetic Algorithm (GA), Particle Swarm Optimization Algorithm (PSO), and Harmony Search Algorithm (HS). Due to the difference in the nature of the presented solution algorithms, each algorithm has its own solution representation. All algorithms are common in solution evaluation.

### 3.4.1 Solution Evaluation

The presented solution algorithms have different solution representation. However, the three algorithms are common in solution representation as shown in Figure 3.2. Thus, for the implementation purpose, a general evaluation function was coded based on the common solution representation.

In the presented model, we have two objective functions: cost and performance.

1. **Cost**
   
   Each of the required services has its own function to calculate its cost. For example, the cost function of renting Virtual Machine (VM) service is different from the cost function of renting storage service. Even with the same type of infrastructure service, different cost functions may be used based on the parameters that define each service. For example, for storage service, file storage service and blob storage service have different cost function with different parameters.

   Consequently, in order to calculate the cost of each required service, for a single solution, the first step is determining the corresponding cost function based on the service type. The second step is finding the corresponding cost data for the provider assigned to the required service. Finally, the cost of the required service when renting from the corresponding provider can be measured by

   ```
   s_1^{(R)} \quad s_2^{(R)} \quad s_3^{(R)} \quad \ldots \quad s_N^{(R)}
   
   3 \quad 10 \quad 20 \quad \ldots \quad 6
   ```

   **FIGURE 3.2**
   Common solution representation.
substituting in the service cost function. The previous steps are repeated until all required services are finished. The total cost of the current solution is the summation of the cost of each required service.

2. Performance

In the presented model, performance has two attributes: suitability and average response time.

(a) For suitability, it is calculated using Equation (3.5). For simplicity and without loss generality, all the essential features of the required service are assumed to be satisfied by all providers. Thus the suitability of each required service in the current solution is equal to 10 and the total suitability of the current solution is \((10 \times N)\), where \(N\) is the number of required services.

(b) For average response time (ART), each provider claim \(ART\) for each offered services based on the service type. ART of each required service is the ART of the assigned provider and ART of the current solution is the summation of \(ART\) of each required service.

The performance of the current solution is the summation of the calculated value of each attribute multiplied by weighting coefficient.

Note that each attribute has different unit; suitability is represented by numbers ranged from 0 to 10, while ART is measured in seconds. Thus, linear normalization (II) [26] is used to unify the units of attributes. Using Equation 3.14, each attribute is normalized based on its lower bound and upper bound, where \(r_k\) is the normalized value of attribute \(k\), \(k = 1, 2, \ldots, K\), \(V_k\) is the calculated value of attribute \(k\), and \(v^-\) and \(v^+\) are lower bound and upper bound of attribute \(k\), respectively

\[
    r_k = \frac{v_k - v^-}{v^+ - v^-}
\]

The weighted sum method was selected to convert the presented multi-objective optimization problem to a single-objective optimization by summing the cost and performance function multiplied by weighting coefficients. Note that, the performance of the current solution is normalized, so before applying the weighted sum method, cost is needed to be normalized using linear normalization (II).

3.4.2 Genetic Algorithm (GA)

GA is a heuristic search based on the Darwin’s theory of natural evolution. “Survival of the fittest” is the basic concept of GA. As in natural, the strong remains fit, while the weak is removed completely. In the same context, solution with the highest fitness value will be survived during different generations, while the solution with the lowest fitness value will be eliminated [27].
GA is a population-based algorithm. It starts with generating an initial feasible population (set of chromosomes/solutions) randomly or by the use of some heuristics. Fitness function (objective function) is used to measure the fitness of each solution. Then, a new population is created using the three operators of GA: (1) selection/reproduction, (2) crossover, and (3) mutation. The selection operator chooses the fitter solutions; the crossover operator selects two solutions randomly in order to exchange genes and mate them; and the mutation changes some of chromosomes’ genes randomly [28]. Figure 3.3 shows pseudo-code of GA.

1. Chromosome Encoding
   For the presented problem, integer encoding representation is suitable. Every chromosome is a string of discrete integer values. Each gene contains information about the provider ID that is potentially assigned to the corresponding client’s requirement. Each gene takes values from 1 to $M$ ($M$, the total number of cloud providers). Chromosome encoding is represented in Figure 3.4

2. New Population
   In order to generate a new population, GA uses three common operators: selection, crossover, and mutation. Selection step involves the

---

**FIGURE 3.3**

Pseudo-code of GA.

| GENERATE random set of chromosomes | // initial population |
| EVALUATE the fitness of each chromosome | |
| REPEAT | |
| SELECT two parents | |
| CROSSOVER the parents | // with crossover probability |
| MUTATE the new offspring | |
| PLACE new offspring in new population | |
| EVALUATE the new population | |
| FIND the best | |
| END IF | |
| UNTIL stopping criteria | |
| RETURN the best | |

**FIGURE 3.4**

GA representation.
decision of how to choose two or more parents from the population for crossing. It is an important decision that affects the convergence of GA. According to the Darwin’s theory of evolution, the individuals with higher fitness value have more chance to be selected. There are many methods for selection. Some of them depend on the relative fitness value of the individuals, while the others are based on the individuals rank with the population. In this study, roulette wheel selection method was selected. In roulette wheel, the sizes of segments are different based on the relative fitness of each individual. Consequently, the individual with highest fitness will have more probability for selection.

After selecting parent individuals/chromosomes, crossover is used to create new offspring. In this study, single point crossover was applied by selecting crossover point randomly and then the sections of two parents are exchanged. Assuming eight requirements and crossover point equal to three, the parents before and after applying crossover are shown in Figure 3.5.

Finally, in order to maintain diversity within the population, mutation was applied by selecting one gene randomly and generate a new provider ID randomly. As shown in Figure 3.6, assume that the selected gene for mutation is the fifth gene that contains provider with ID = 37. After mutation the fifth gene will contain provider ID = 55 that generated randomly.

FIGURE 3.5
Single point crossover.

FIGURE 3.6
Example of mutation.
3.4.3 Particle Swarm Optimization (PSO)

PSO is a population-based stochastic optimization algorithm introduced by Kennedy and Eberhart in 1995 [29]. It is inspired by observing the behavior of bird flocking and fish schooling and is mainly based on the guidance of particle to search for global optimal solutions. Each particle represents a potential solution to the problem and has coordinates and rate of change (velocity) in a D-dimensional space. Particles’ velocities are one of the important parameters for adjusting the directions of the particle toward their target. Particles’ velocities are determined by two factors: (1) their own best experience (cognition part), and (2) the best experience of all other particles (social part).

The original version of PSO is limited in real number space. Therefore, for discrete optimization problem, Kennedy and Eberhart [30] developed a discrete version of PSO. Discrete PSO allows the particle to be composed of binary variable. Besides, to update the particle velocity is transformed into the change of probability, which is the chance of binary variable taking a value one. Figure 3.7 shows the pseudo-code of PSO.

1. Particle Definition
   The solution space of the presented problem is discrete. Therefore, to be able to use a discrete PSO, the solution is represented in different way. As in GA, the initialization of the initial population/particles is generated randomly; for each required service, a random cloud provider is selected. However, to apply the discrete PSO, the generated population is expressed in binary notation. As shown in Figure 3.8, a particle is represented by a matrix with $M$ rows and $N$ columns, where $M$ is the number of cloud providers and $N$ is the number of required services.

   For example, in order to represent that cloud provider with ID = 4 is the candidate provider that will satisfy the first required service ($s^{(R)}_1$), the element at the fourth row and first column will contain 1 and the remaining elements in the first column will contain 0.

---

**Figure 3.7**

Pseudo-code of PSO.

| INITIALIZE population, velocity, best position, global position, and PSO parameters |
| EVALUATE population  |
| REPEAT               |
| UPDATE velocity      |
| UPDATE position      |
| EVALUATE population  |
| FIND best position   |
| FIND global best     |
| UNTIL stopping criteria |

---
contain 0 to satisfy constraint (3.9); each required service is rented from a single provider.

2. Generate New Population

A new population is created by updating the velocity and position of all particles. Velocity is updated as in continuous PSO according to cognition part and social part. Mathematically, using Equation (3.15)

\[ v_{id}^{t+1} = wv_{id}^t + \varphi_1 r_1 (pbest_{id}^t - x_{id}^t) + \varphi_2 r_2 (gbest_{gd}^t - x_{id}^t) \]  

(3.15)

where \( \varphi_1 \) is the cognition learning factor, \( \varphi_2 \) is the social learning factor, \( r_1 \) and \( r_2 \) are random numbers uniformly distributed in \([0, 1]\), \( pbest_{id}^t \) is the individual’s \( i \) best position found so far at site \( d \), \( gbest_{gd}^t \) is the neighborhood best state found so far at site \( d \), \( x_{id} \) is the current position at site \( d \) of particle \( i \), \( v_{id}^t \) is the current velocity at site \( d \) of particle \( i \), and \( w \) is inertia weight.

The new velocity of each particle is used to update its position by using Equation (3.16) which present the sigmoid function to calculate the probability of \( x_{id}^t \) bit taking 1

\[ s(v_{id}^t) = \frac{1}{1 + exp(-v_{id}^t)} \]  

(3.16)

After updating the position of all particles, we need to be sure that new particles/solutions are feasible solutions. As mentioned before, each required service is rented from one provider. Consequently, a simple heuristic is used to check the feasibility of the new particles as follows: for each required service, if more than one provider was assigned, one of them is selected randomly and assign “1” in the corresponding bit and “0” is assigned to the remaining providers. The heuristic for one particle is explained using Figure 3.9. For simplicity, let us assume that client’s requirements are three services and ten providers are available to satisfy the requirements. As shown in Figure 3.9, according to the unfeasible particle, the required service \( s_1^{(R)} \) can be rented from providers with IDs (3, 4, 6, 7, and 9).
Thus, a random provider ID is selected from the assigned IDs. For example, let the selected random provider be provider with ID = 6. Consequently, “1” is assigned to the selected provider and “0” is assigned to the remaining providers.

3.4.4 Harmony Search (HS)

HS is a music-inspired algorithm developed by Zong Woo Geem [31] through simulating the musicians’ behavior in producing a piece of music with perfect harmony. Skilled musician rely on improvisation through three possible choices: (1) use his memory to play a known piece of music exactly as it is (harmony memory); (2) play a known piece with adjusting the pitch a little (pitch adjusting); or (3) create a piece of music randomly (randomization). Lee and Geem [32] used the same concepts of the three choices of improvisation process to formalize it into quantitative optimization process.

Harmony memory plays an important role to ensure that the best harmonies will be moved to the new harmony memory. The selection is controlled by the harmony consideration rate (HMCR). With the low rate of the HMCR, few best harmonies are selected, whereas with the high rate, almost all harmonies are chosen. Low HMCR results in slow convergence, whereas high HMCR results in poor exploration. Therefore, HMCR are always >70%. For each selected harmonies, pitch adjustment is applied according to pitch adjusting rate (PAR). In order to increase the diversity of the solutions, randomization is applied with probability (100 HMRC) %. Figure 3.10 shows the pseudo-code of the HS algorithm.

1. Harmony Representation

Harmony representation is similar to chromosome representation discussed in Section 3.4.2 Figure 3.11 shows solution representation in terms of the HS algorithm.
2. Create New Harmony Memory
   A new harmony memory is created by applying three different options: (1) harmony memory, (2) pitch adjusting, or (3) randomization. Let us assume that we have three required services and ten providers. Given harmony memory size equal to 5, the initial harmony memory is created as shown in Figure 3.12.

   Let HMCR = 75% and PAR = 20%. For the first harmony, random number is generated (HMCRRnd=0.326). HMCRRnd is less than HMCR. Thus, the first value is kept, then another random number is generated to test the probability of changing current pitch value (PARRnd=0.633). PARRnd is greater than PAR, thus
the same value is copied into new harmony memory. For the second pitch, HMCRRnd=0.207 and HMCRRnd < 0.75, then the same value is moved to the new memory. When PARRnd=0.186 and PARRnd < 0.2, then a little change is applied to the pitch value. In this research, the change is suggested to be as follows: generate a new provider ID randomly and add this value to the current pitch value. Assume that the generated provider ID is “5,” the new pitch value will be (1+5 = 6). Note that before adding the new pitch value to the new memory, we need to ensure that it is a valid provider ID with the range \([1, M]\), where \(M\) is the number of providers. In this example “6” is a valid ID.

For the last requirement, HMCRRnd = 0.78 and HMCRRnd > 0.75, so randomization option is applied. Let the random value be “7.” After applying the three options to the first harmony, the new harmony/solution is 3, 6, and 7. By applying the same procedure to the rest of harmonies, the new harmony memory is generated as shown in Figure 3.13.

### FIGURE 3.13
New harmony memory.

<table>
<thead>
<tr>
<th>(s_1^{(R)})</th>
<th>(s_2^{(R)})</th>
<th>(s_3^{(R)})</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

#### 3.5 Numerical Experiments

For the purpose of testing the performance of the three algorithms, an experimental case study was generated; the most popular cloud providers were scanned and the needed information was collected. Three IaaS services were selected as common service offered by the scanned providers: (1) virtual machine (VM), (2) blob storage (SB), and (3) file storage (SF). Each service is defined by different set of parameters—VM is defined by the following parameters: instance configuration (Conf), operating system (OS), OS type, region (R), while SB is defined by region, capacity, access tier, and redundancy; and SF is defined by region, capacity and redundancy. Ten regions were selected as listed in Table A.1. For VM service 49 common configurations were chosen and six different OS types were selected as presented in Tables 3.1 and 3.2, respectively.
### TABLE 3.1
VM Configurations

<table>
<thead>
<tr>
<th>ID</th>
<th>Cores</th>
<th>RAM Sizes/GB</th>
<th>Disk Sizes/GB</th>
<th>ID</th>
<th>Cores</th>
<th>RAM Sizes/GB</th>
<th>Disk Sizes/GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conf1</td>
<td>1</td>
<td>0.75</td>
<td>20</td>
<td>Conf26</td>
<td>8</td>
<td>16</td>
<td>128</td>
</tr>
<tr>
<td>Conf2</td>
<td>1</td>
<td>1.75</td>
<td>40</td>
<td>Conf27</td>
<td>16</td>
<td>32</td>
<td>256</td>
</tr>
<tr>
<td>Conf3</td>
<td>2</td>
<td>3.5</td>
<td>60</td>
<td>Conf28</td>
<td>2</td>
<td>14</td>
<td>100</td>
</tr>
<tr>
<td>Conf4</td>
<td>4</td>
<td>7</td>
<td>120</td>
<td>Conf29</td>
<td>4</td>
<td>28</td>
<td>200</td>
</tr>
<tr>
<td>Conf5</td>
<td>8</td>
<td>14</td>
<td>240</td>
<td>Conf30</td>
<td>8</td>
<td>56</td>
<td>400</td>
</tr>
<tr>
<td>Conf6</td>
<td>1</td>
<td>2</td>
<td>10</td>
<td>Conf31</td>
<td>16</td>
<td>112</td>
<td>800</td>
</tr>
<tr>
<td>Conf7</td>
<td>2</td>
<td>4</td>
<td>20</td>
<td>Conf32</td>
<td>20</td>
<td>140</td>
<td>1,000</td>
</tr>
<tr>
<td>Conf8</td>
<td>4</td>
<td>8</td>
<td>40</td>
<td>Conf33</td>
<td>32</td>
<td>256</td>
<td>512</td>
</tr>
<tr>
<td>Conf9</td>
<td>8</td>
<td>16</td>
<td>80</td>
<td>Conf34</td>
<td>64</td>
<td>432</td>
<td>864</td>
</tr>
<tr>
<td>Conf10</td>
<td>2</td>
<td>16</td>
<td>20</td>
<td>Conf35</td>
<td>2</td>
<td>28</td>
<td>384</td>
</tr>
<tr>
<td>Conf11</td>
<td>4</td>
<td>32</td>
<td>40</td>
<td>Conf36</td>
<td>4</td>
<td>56</td>
<td>768</td>
</tr>
<tr>
<td>Conf12</td>
<td>8</td>
<td>64</td>
<td>80</td>
<td>Conf37</td>
<td>8</td>
<td>112</td>
<td>1,536</td>
</tr>
<tr>
<td>Conf13</td>
<td>2</td>
<td>7</td>
<td>100</td>
<td>Conf38</td>
<td>16</td>
<td>224</td>
<td>3,072</td>
</tr>
<tr>
<td>Conf14</td>
<td>4</td>
<td>14</td>
<td>200</td>
<td>Conf39</td>
<td>32</td>
<td>448</td>
<td>6,144</td>
</tr>
<tr>
<td>Conf15</td>
<td>8</td>
<td>28</td>
<td>400</td>
<td>Conf40</td>
<td>64</td>
<td>1,750</td>
<td>2,000</td>
</tr>
<tr>
<td>Conf16</td>
<td>16</td>
<td>56</td>
<td>800</td>
<td>Conf41</td>
<td>128</td>
<td>2,000</td>
<td>4,000</td>
</tr>
<tr>
<td>Conf17</td>
<td>2</td>
<td>8</td>
<td>16</td>
<td>Conf42</td>
<td>4</td>
<td>32</td>
<td>678</td>
</tr>
<tr>
<td>Conf18</td>
<td>4</td>
<td>16</td>
<td>32</td>
<td>Conf43</td>
<td>8</td>
<td>64</td>
<td>13,88</td>
</tr>
<tr>
<td>Conf19</td>
<td>8</td>
<td>32</td>
<td>64</td>
<td>Conf44</td>
<td>16</td>
<td>128</td>
<td>2,807</td>
</tr>
<tr>
<td>Conf20</td>
<td>16</td>
<td>64</td>
<td>128</td>
<td>Conf45</td>
<td>32</td>
<td>256</td>
<td>5,630</td>
</tr>
<tr>
<td>Conf21</td>
<td>32</td>
<td>128</td>
<td>256</td>
<td>Conf46</td>
<td>8</td>
<td>56</td>
<td>1,000</td>
</tr>
<tr>
<td>Conf22</td>
<td>64</td>
<td>256</td>
<td>512</td>
<td>Conf47</td>
<td>16</td>
<td>112</td>
<td>2,000</td>
</tr>
<tr>
<td>Conf23</td>
<td>1</td>
<td>2</td>
<td>16</td>
<td>Conf48</td>
<td>8</td>
<td>112</td>
<td>1,000</td>
</tr>
<tr>
<td>Conf24</td>
<td>2</td>
<td>4</td>
<td>32</td>
<td>Conf49</td>
<td>16</td>
<td>224</td>
<td>2,000</td>
</tr>
<tr>
<td>Conf25</td>
<td>4</td>
<td>8</td>
<td>64</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### TABLE 3.2
OS Types

<table>
<thead>
<tr>
<th>ID</th>
<th>OS Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS1</td>
<td>Red Hat Enterprise Linux</td>
</tr>
<tr>
<td>OS2</td>
<td>SUSE Linux Enterprise</td>
</tr>
<tr>
<td>OS3</td>
<td>Windows OS</td>
</tr>
<tr>
<td>OS4</td>
<td>Windows with SQL Standard</td>
</tr>
<tr>
<td>OS5</td>
<td>Windows with SQL Web</td>
</tr>
<tr>
<td>OS6</td>
<td>Windows with SQL Enterprise</td>
</tr>
</tbody>
</table>

In order to calculate the cost and performance of each service, prices and average response times offered by providers were scanned and the absolute minimum and the absolute maximum were collected. Example of the prices intervals per hour for the VM in dollars, storage prices per GB in dollars, and ART for VM and storage services are presented in Tables 3.3–3.5,
TABLE 3.3
Example of the Price Ranges Per Hour for the VM in Dollars

<table>
<thead>
<tr>
<th></th>
<th>R1</th>
<th></th>
<th>R2</th>
<th></th>
<th>R3</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min</td>
<td>Max</td>
<td>Min</td>
<td>Max</td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>OS1</td>
<td>0.078</td>
<td>0.312</td>
<td>0.082</td>
<td>0.328</td>
<td>0.084</td>
<td>0.336</td>
</tr>
<tr>
<td>OS2</td>
<td>0.118</td>
<td>0.472</td>
<td>0.122</td>
<td>0.488</td>
<td>0.124</td>
<td>0.496</td>
</tr>
<tr>
<td>Conf1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OS3</td>
<td>0.018</td>
<td>0.072</td>
<td>0.022</td>
<td>0.088</td>
<td>0.024</td>
<td>0.096</td>
</tr>
<tr>
<td>OS4</td>
<td>0.418</td>
<td>1.672</td>
<td>0.422</td>
<td>1.688</td>
<td>0.424</td>
<td>1.696</td>
</tr>
<tr>
<td>OS5</td>
<td>0.05</td>
<td>0.2</td>
<td>0.054</td>
<td>0.216</td>
<td>0.056</td>
<td>0.224</td>
</tr>
<tr>
<td>OS6</td>
<td>1.518</td>
<td>6.072</td>
<td>1.522</td>
<td>6.088</td>
<td>1.524</td>
<td>6.096</td>
</tr>
<tr>
<td>OS1</td>
<td>0.097</td>
<td>0.388</td>
<td>0.092</td>
<td>0.368</td>
<td>0.092</td>
<td>0.368</td>
</tr>
<tr>
<td>OS2</td>
<td>0.137</td>
<td>0.548</td>
<td>0.132</td>
<td>0.528</td>
<td>0.132</td>
<td>0.528</td>
</tr>
<tr>
<td>Conf2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OS3</td>
<td>0.05</td>
<td>0.2</td>
<td>0.041</td>
<td>0.164</td>
<td>0.041</td>
<td>0.164</td>
</tr>
<tr>
<td>OS4</td>
<td>0.45</td>
<td>1.8</td>
<td>0.441</td>
<td>1.764</td>
<td>0.441</td>
<td>1.764</td>
</tr>
<tr>
<td>OS5</td>
<td>0.082</td>
<td>0.328</td>
<td>0.073</td>
<td>0.292</td>
<td>0.073</td>
<td>0.292</td>
</tr>
<tr>
<td>OS6</td>
<td>1.55</td>
<td>6.2</td>
<td>1.541</td>
<td>6.164</td>
<td>1.541</td>
<td>6.164</td>
</tr>
<tr>
<td>OS1</td>
<td>0.155</td>
<td>0.62</td>
<td>0.169</td>
<td>0.676</td>
<td>0.164</td>
<td>0.656</td>
</tr>
<tr>
<td>OS2</td>
<td>0.295</td>
<td>1.18</td>
<td>0.309</td>
<td>1.236</td>
<td>0.304</td>
<td>1.216</td>
</tr>
<tr>
<td>Conf3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OS3</td>
<td>0.137</td>
<td>0.548</td>
<td>0.145</td>
<td>0.58</td>
<td>0.167</td>
<td>0.668</td>
</tr>
<tr>
<td>OS4</td>
<td>0.537</td>
<td>2.148</td>
<td>0.545</td>
<td>2.18</td>
<td>0.567</td>
<td>2.268</td>
</tr>
<tr>
<td>OS5</td>
<td>0.169</td>
<td>0.676</td>
<td>0.177</td>
<td>0.708</td>
<td>0.199</td>
<td>0.796</td>
</tr>
<tr>
<td>OS6</td>
<td>1.637</td>
<td>6.548</td>
<td>1.645</td>
<td>6.58</td>
<td>1.667</td>
<td>6.668</td>
</tr>
</tbody>
</table>

TABLE 3.4
Example of Storage Prices per GB in Dollars

<table>
<thead>
<tr>
<th></th>
<th>R1</th>
<th></th>
<th>R2</th>
<th></th>
<th>R3</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min</td>
<td>Max</td>
<td>Min</td>
<td>Max</td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>First 1 TB/Month</td>
<td>0.024</td>
<td>0.096</td>
<td>0.0326</td>
<td>0.1304</td>
<td>0.0264</td>
<td>0.1056</td>
</tr>
<tr>
<td>SB</td>
<td>0.0236</td>
<td>0.0944</td>
<td>0.032</td>
<td>0.128</td>
<td>0.026</td>
<td>0.104</td>
</tr>
<tr>
<td>Next 49 TB/Month</td>
<td>0.0232</td>
<td>0.0928</td>
<td>0.0315</td>
<td>0.126</td>
<td>0.0255</td>
<td>0.102</td>
</tr>
<tr>
<td>Next 450 TB/Month</td>
<td>0.0228</td>
<td>0.0912</td>
<td>0.0309</td>
<td>0.1236</td>
<td>0.0251</td>
<td>0.1004</td>
</tr>
<tr>
<td>SF</td>
<td>0.08</td>
<td>0.32</td>
<td>0.109</td>
<td>0.436</td>
<td>0.088</td>
<td>0.352</td>
</tr>
</tbody>
</table>

respectively. The price interval of each VM is determined according to VM configuration, OS type, and the region located in it. For example, the price interval of VM with Conf1 [1 Cores(s), 0.75 GB RAM, 20 GB Temporary Storage, and Red Hat Enterprise Linux] located in the Southeast Asia is [0.078, 0.312].

Using the selected IaaS services, a case study of eight client’s requirement is generated. It consists of four VM, one SB, and three SF as presented in Table 3.6 In addition, data of 60 providers were generated randomly by using
the absolute minimum and the absolute maximum. Example of provider’s data is presented in Appendix A. The generated data for the three IaaS services for four providers are presented in Tables A.2 and A.3, respectively. For example, the prices per hour for VM with {Conf1, OS} located in R1 from provider 1, provider 2, provider 3, and provider 4 are $0.2431, $0.2089, $0.1539 and $0.3016, respectively, where ART(s) are 703.79/s, 708.65/s, 670.65/s, and 731.48/s, respectively.
3.6 Algorithms’ Performance

Theoretically, when the number of generations is increased, the quality of the obtained solutions will be improved. In practice, this rule is not always true because of the stochastic nature of these algorithms. So, the quality of solutions of the three algorithms was checked with different number of generations. Three different generation numbers (100, 150, and 200) were chosen in order to check the effectiveness of changing parameters values on the properties of the algorithms.

A population of 100 solutions was generated randomly and used as an initial population for the three algorithms. The parameters values of each algorithm were chosen to ensure that all algorithms evaluate the same number of solutions. Each algorithm was run 30 times and the analysis was done based on the average weighted objective values of the best solution obtained through all runs. The performance analysis was done in two phases. In the first phase, the performance of each algorithm was checked with the three numbers of generation. While, in the second phase, the performance of the three algorithms was checked with each number of generation. The proposed algorithms was implemented using Visual Basic for Application (VBA) on Excel. All experiments were conducted on machine with Intel Core Duo CPU (2.27 GHz) and 4 GB RAM.

The progress of the best solution for GA, HS, and PSO is shown in Figures 3.14–3.16, respectively. It is noticed that the quality of the solutions of the algorithm was improved when the number of generation increased. For GA and HS, the quality of solutions was improved, but with a little change in the best solution objective value, while PSO continued to improve its solutions with remarkable change.

The best solution progress of the three algorithms with 100 generations is shown in Figure 3.17. In the beginning, it was noticed that the solutions obtained by HS is better than the solutions obtained by GA. However, GA started to improve its solutions after approximately 7,500 solution evaluation and achieve best solution with better objective value than HS. While, the solutions obtained by PSO are better than the solutions obtained by GA and HS.

The box-plot of the three algorithms with 100 generations is shown in Figure 3.18. We can notice that there is a remarkable difference in the median value of the three algorithms. As mentioned before, the algorithms were applied to a minimization problem and based on the presented information from the box-plot, it is notice that PSO has the better properties than GA and HS. This is because PSO has the minimum value and quartile Q0.25 and Q0.75 are closer to the minimum than GA and HS. GA achieve minimum value better than HS, however, HS has better properties than GA as the median
FIGURE 3.14
GA with different generations.

FIGURE 3.15
HS with different generations.
FIGURE 3.16
PSO with different generations.

FIGURE 3.17
Best solution progress of the three algorithms with 100 generations.
value of HS is smaller than median value of GA and quartile Q0.25 and Q0.75 of HS are closer to the minimum than GA.

The best solution progress of the three algorithms with 150 generations is shown in Figure 3.19. There is no significant difference in the solutions obtained by GA and HS, but HS achieve the minimum value. PSO has remarkable difference in its obtained solutions in comparison with the solutions obtained by GA and HS. From the information of the box-plot of the three algorithms with 150 generations shown in Figure 3.20, we can notice that the minimum value achieved by HS is smaller than the minimum value obtained by GA, however, quartile Q0.25 and Q0.75 of the two algorithms are close. Also, there is a significant difference in the median of PSO; it is smaller than the median of GA and the median of HS and quartile Q0.75 of PSO is smaller than median of GA and HS. Consequently, the solutions acquired by PSO are the best.

Figure 3.21 shows the progress of the three algorithms with 200 generations. The solutions achieved by GA are the worst, while the solutions obtained by PSO are the best. From Figure 3.22, PSO has the minimum value and quartile Q0.25 is close to it, but quartile Q0.50 and quartile Q0.75 are not close. However, PSO has better properties than GA and HS.

From the analysis of the behavior of the three algorithms with different generations, we can conclude that PSO is better than GA and HS; the
FIGURE 3.19
Best solution progress of the three algorithms with 150 generations.

FIGURE 3.20
Box-plot of the three algorithms with 150 generations.
FIGURE 3.21
Best solution progress of the three algorithms with 200 generations.

FIGURE 3.22
Box-plot of the three algorithms with 200 generations.
behavior of PSO during different generations is better than GA and HS and it always achieves the minimum value. Besides, the progress of the best solutions obtained by PSO is more robust and stable than the best solutions obtained by GA and HS.

3.7 Conclusion and Future Work

With the diversity of cloud providers, cloud customers faces a challenge of discovering the right providers that can satisfy their requirements. Based on the variation of customer’s needs, many research papers proved that dealing with single provider is not sufficient. Thus, this paper presented a general mathematical model to select multiple providers based on minimum cost and maximum performance in order to satisfy customer’s requirements and needs. Three heuristics algorithms were proposed to solve the model. Results showed that better solutions obtained by PSO are better than the solutions achieved by GA and HS.

The major limitation of this paper is that the data of the case study were generated randomly and it will be better to test the model on real data. In addition, each of the proposed algorithms has a set of parameters that have a significant influence on the algorithm performance. So, studying the effects of changing values of the different parameters is an important issue. Therefore, further analysis is needed to determine the optimal parameters set.

Appendix A: Example of Case Study Data

<table>
<thead>
<tr>
<th>Region ID</th>
<th>Region Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>Southeast Asia</td>
</tr>
<tr>
<td>R2</td>
<td>Japan</td>
</tr>
<tr>
<td>R3</td>
<td>Australia Southeast</td>
</tr>
<tr>
<td>R4</td>
<td>Korea South</td>
</tr>
<tr>
<td>R5</td>
<td>West India</td>
</tr>
<tr>
<td>R6</td>
<td>West Europe</td>
</tr>
<tr>
<td>R7</td>
<td>Germany Central</td>
</tr>
<tr>
<td>R8</td>
<td>UK South</td>
</tr>
<tr>
<td>R9</td>
<td>US Gov Virginia</td>
</tr>
<tr>
<td>R10</td>
<td>Canada Central</td>
</tr>
</tbody>
</table>
### TABLE A.2
Random Data for Virtual Machine Service for Four Providers

<table>
<thead>
<tr>
<th>Provider</th>
<th>Conf1</th>
<th>Conf2</th>
<th>Conf3</th>
<th>ART/s</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\text{price/h}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Provider 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OS1</td>
<td>0.2431</td>
<td>0.2132</td>
<td>0.23</td>
<td></td>
</tr>
<tr>
<td>OS2</td>
<td>0.1341</td>
<td>0.2735</td>
<td>0.4449</td>
<td></td>
</tr>
<tr>
<td>OS3</td>
<td>0.0463</td>
<td>0.0726</td>
<td>0.0279</td>
<td></td>
</tr>
<tr>
<td>OS4</td>
<td>1.4586</td>
<td>1.4659</td>
<td>1.1734</td>
<td>703.79</td>
</tr>
<tr>
<td>OS5</td>
<td>0.066</td>
<td>0.2159</td>
<td>0.1696</td>
<td></td>
</tr>
<tr>
<td>OS6</td>
<td>2.8649</td>
<td>3.2663</td>
<td>2.9</td>
<td></td>
</tr>
<tr>
<td>Provider 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OS1</td>
<td>0.2163</td>
<td>0.2059</td>
<td>0.2887</td>
<td></td>
</tr>
<tr>
<td>OS2</td>
<td>0.5093</td>
<td>0.2355</td>
<td>0.4429</td>
<td></td>
</tr>
<tr>
<td>OS3</td>
<td>0.1342</td>
<td>0.1264</td>
<td>0.1534</td>
<td></td>
</tr>
<tr>
<td>OS4</td>
<td>1.1435</td>
<td>1.0535</td>
<td>0.9086</td>
<td>841.59</td>
</tr>
<tr>
<td>OS5</td>
<td>0.1718</td>
<td>0.1803</td>
<td>0.1071</td>
<td></td>
</tr>
<tr>
<td>OS6</td>
<td>3.9033</td>
<td>3.3461</td>
<td>2.0374</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(Continued)
<table>
<thead>
<tr>
<th>Provider 3</th>
<th>Conf1</th>
<th>Conf2</th>
<th>Conf3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OS1</td>
<td>OS1</td>
<td>OS1</td>
</tr>
<tr>
<td></td>
<td>0.1539</td>
<td>0.1997</td>
<td>0.369</td>
</tr>
<tr>
<td></td>
<td>0.1404</td>
<td>0.2548</td>
<td>0.5606</td>
</tr>
<tr>
<td></td>
<td>0.1563</td>
<td>0.2755</td>
<td>0.3381</td>
</tr>
<tr>
<td></td>
<td>0.1599</td>
<td>0.448</td>
<td>0.4046</td>
</tr>
<tr>
<td></td>
<td>0.0563</td>
<td>0.4751</td>
<td>0.4069</td>
</tr>
<tr>
<td></td>
<td>0.058</td>
<td>0.0569</td>
<td>0.2968</td>
</tr>
<tr>
<td></td>
<td>0.0396</td>
<td>0.0899</td>
<td>0.5594</td>
</tr>
<tr>
<td></td>
<td>0.0586</td>
<td>0.1206</td>
<td>0.058</td>
</tr>
<tr>
<td></td>
<td>0.1525</td>
<td>0.0596</td>
<td>0.0387</td>
</tr>
<tr>
<td></td>
<td>0.2078</td>
<td>0.2302</td>
<td>0.1712</td>
</tr>
<tr>
<td></td>
<td>4.2423</td>
<td>3.629</td>
<td>5.1745</td>
</tr>
<tr>
<td></td>
<td>2.3048</td>
<td>4.8864</td>
<td>3.8522</td>
</tr>
<tr>
<td></td>
<td>3.0891</td>
<td>2.9123</td>
<td>1.6205</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Provider 4</td>
<td>Conf1</td>
<td>Conf2</td>
<td>Conf3</td>
</tr>
<tr>
<td></td>
<td>OS1</td>
<td>OS1</td>
<td>OS1</td>
</tr>
<tr>
<td></td>
<td>0.3016</td>
<td>0.3054</td>
<td>0.1892</td>
</tr>
<tr>
<td></td>
<td>0.183</td>
<td>0.1807</td>
<td>0.2677</td>
</tr>
<tr>
<td></td>
<td>0.1639</td>
<td>0.3013</td>
<td>0.1886</td>
</tr>
<tr>
<td></td>
<td>0.3708</td>
<td>0.2797</td>
<td>0.654</td>
</tr>
<tr>
<td></td>
<td>0.2242</td>
<td>0.3207</td>
<td>1.0577</td>
</tr>
<tr>
<td></td>
<td>0.46</td>
<td>0.4378</td>
<td>0.5119</td>
</tr>
<tr>
<td></td>
<td>0.058</td>
<td>0.1734</td>
<td>0.3865</td>
</tr>
<tr>
<td></td>
<td>0.0387</td>
<td>0.155</td>
<td>0.4987</td>
</tr>
<tr>
<td></td>
<td>0.0566</td>
<td>0.0919</td>
<td>0.5616</td>
</tr>
<tr>
<td></td>
<td>1.3302</td>
<td>1.197</td>
<td>1.9231</td>
</tr>
<tr>
<td></td>
<td>0.6288</td>
<td>0.7468</td>
<td>1.0817</td>
</tr>
<tr>
<td></td>
<td>1.3507</td>
<td>1.2853</td>
<td>0.6381</td>
</tr>
<tr>
<td></td>
<td>0.115</td>
<td>0.1585</td>
<td>0.4356</td>
</tr>
<tr>
<td></td>
<td>0.0889</td>
<td>0.1658</td>
<td>0.4418</td>
</tr>
<tr>
<td></td>
<td>0.1712</td>
<td>0.1363</td>
<td>0.5906</td>
</tr>
<tr>
<td></td>
<td>4.8864</td>
<td>2.2919</td>
<td>4.3998</td>
</tr>
<tr>
<td></td>
<td>2.9123</td>
<td>4.9028</td>
<td>5.2755</td>
</tr>
<tr>
<td></td>
<td>1.6205</td>
<td>3.6486</td>
<td>4.3069</td>
</tr>
</tbody>
</table>
### TABLE A.3
Random Data for Storage Services for Four Providers

<table>
<thead>
<tr>
<th>Provider</th>
<th>First 1 TB/Month</th>
<th>Next 49 TB/Month</th>
<th>Next 450 TB/Month</th>
<th>Next 500 TB/Month</th>
<th>ART/s</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>R1</strong></td>
<td><strong>R2</strong></td>
<td><strong>R3</strong></td>
<td><strong>R4</strong></td>
<td><strong>R5</strong></td>
<td></td>
</tr>
<tr>
<td>Provider 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB</td>
<td>0.0797</td>
<td>0.1232</td>
<td>0.0743</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SF</td>
<td>0.2534</td>
<td>0.4053</td>
<td>0.2706</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Provider 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB</td>
<td>0.0617</td>
<td>0.1082</td>
<td>0.0979</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SF</td>
<td>0.2233</td>
<td>0.336</td>
<td>0.145</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Provider 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB</td>
<td>0.0869</td>
<td>0.1135</td>
<td>0.0701</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SF</td>
<td>0.2201</td>
<td>0.2517</td>
<td>0.0899</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Provider 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SB</td>
<td>0.0596</td>
<td>0.09</td>
<td>0.0685</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SF</td>
<td>0.0924</td>
<td>0.365</td>
<td>0.3103</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bibliography**


Techniques for Service Provider Selection


Reliable Data Auditing and ACO-Based Resource Scheduling for Cloud Services

Soumitra Sasmal
Techno Main

Indrajit Pan
RCC Institute of Information Technology

CONTENTS

4.1 Introduction ...................................................... 108
4.2 Basic Concepts ................................................... 111
   4.2.1 Basic Outline of Ant Colony Optimization Method .... 111
   4.2.2 Concept of Merkle Tree ................................ 112
4.3 Problem Definition ............................................ 112
4.4 Proposed Solution .............................................. 113
   4.4.1 Dynamic Resource Scheduling ............................ 113
   4.4.2 Ant Colony Optimization-Based Resource Scheduling ............................................ 115
      4.4.2.1 Pheromone Function ............................. 115
      4.4.2.2 Decision on Resource Scheduling ............... 115
   4.4.3 Auditing of User Data ................................... 116
4.5 Results and Discussion ........................................ 117
   4.5.1 Metadata Generation Time for Data Files of Varying Sizes ...................................... 117
   4.5.2 Data Verification with Varying Numbers of Tenants .... 119
   4.5.3 Successful Resource Allocation with Different Volumes of Active Users .......................... 120
   4.5.4 Balanced Allocation of Server Resources ............. 120
4.6 Conclusion ........................................................ 122
Bibliography ............................................................ 123
4.1 Introduction

Virtualization of expert services is in trend over the decades. It helps people to work on different domains even without in-depth knowledge of technology [22]. One of the fastest growing computing paradigms is cloud computing. Cloud computing has gained much attention by empowering people through virtualization of computing infrastructure and services. It can facilitate people or organizations with huge computing infrastructure like CPU core, CPU bandwidth, network infrastructure and peripheral supports. Similarly, it extends support for storage of massive volumes of data and provides a high-end software framework to develop a new application or to simulate and execute some highly resource-centric processes [24].

Figure 4.1 illustrates a classic cloud computing architecture in which the back-end resources combine server, virtual desktop, software platforms, storage or data servers and application software framework [7]. End users are part of the front end and usually connected with the back end via internet through routers and switches. Majority of cloud users can avail the data storage services of cloud infrastructure. Some of the data owners store their data files on cloud storage servers which are later used by many of its associated users [7]. One of the major concerns in maintaining these files in cloud storage is their security. Files stored on a distant virtual storage have a high

FIGURE 4.1
Cloud computing architecture.
probability of being distorted. Sometimes, one can even upload some content that violates the ethical norms of cloud usage. Thus, auditing of these files is very important both while uploading and during subsequent accesses [12]. In literature, there are many reported works on public auditing of cloud data. A data dynamics-based cloud auditing protocol has been proposed by the work carried out in [12]. The authors have used signature scheme of homomorphic network to perform public auditing and privacy preservation of cloud data. They have implemented an RSA (Rivest–Shamir–Adleman)-based data integration and auditing scheme to execute third-party auditing without revealing any detail of the files content. In another work [15], a challenge–response-based method has been proposed for auditing data files on cloud. Here, the authors have designed two algorithms: the first-phase algorithm uses standard cryptographic keyGen algorithm for signature generation which is later used to populate metadata. The second algorithm performs challenge–response-based auditing of the data. A continuous auditing scheme has been proposed for various cloud services through third-party auditor (TPA) in a work reported in [16]. This scheme monitors on-demand access provisioning, competencies of hardware and software components of the servers, and changes in the rule base to maintain reliability and transparency of services. The authors have proposed a secured data storage mechanism here by generating a signature through MD5 algorithm. This signature is appended with each encrypted data blocks. Base 64 algorithm is proposed for encrypting data blocks which will be uploaded in cloud server. This method also includes a mechanism for checking data integrity by TPA where the auditor can generate a report if any intrusion is detected within the stored data file. A method for encrypting data blocks of each file through advanced encryption standard (AES) algorithm and protecting those encrypted blocks through a message digest generated by secure hash algorithm 2 (SHA-2) is proposed in [17]. This is an effective method that claims to be beneficial for third-party auditing, and using this method, auditing can be completed within a fixed amount of time irrespective of the type and size of files.

The works available in literature have each focused on different approaches to minimize the complexity in auditing of data files and the task of resource allocation. Resource allocation is also a very challenging task in cloud computing environment [21,23,25]. This task has a close relation with auditing procedures of data files. Checking the integrity of data files while accessing them is an important process. The process of resource allocation also guides the factors related to file access. It derives the physical location of servers that will be assigned to offer the file access for a particular request. An efficient yet simple resource allocation strategy will ease out verification of data files integrity before granting the access.

There are many research reports available in literature which have worked on different resource allocation strategies in cloud computing environment. A contract-based resource sharing algorithm for geographically distributed and
federated cloud infrastructure is proposed in [14]. A scheduling request needs to be submitted to cloud service provider before 24 hours, and based on all requisitions received in a given span of time, this proposed method of [14] performs job scheduling. A load balancing mechanism in virtual machines and resource scheduling is proposed in [18]. This work has taken several simulation setups into consideration to monitor the resource sharing performances by balancing the load of requests through different trials. Swarm optimization approach has been adopted for resource sharing in the work carried out in [19]. They have designed a hybrid method with particle swarm optimization technique and ant colony optimization technique. Different resource units in the system were assumed to be the ants for the method. These ants were primarily initialized with a pheromone value to initiate ant colony optimization approach. Pheromone density on each ant was decided through a particle swarm optimization method. Subsequently, an ant optimization strategy based on pheromone trails was adopted for optimum resource allocation. The authors in [20] have proposed a mechanism for classification of user files to optimize the storage space utilization. They have performed this classification on the basis of usage rate of different files. Mainly the files were categorized into three different clusters (as (i) files which are very frequently used, (ii) files which are averagely used and (iii) files those are rarely used), and different compression techniques were applied on them. Files with heavy usage were given the highest compression. Huffman encoding and Lempel-Ziv algorithms were used for compression.

Optimization of resource allocation in cloud is always a challenging and most talked about issue. Researchers are rigorously coming up with many proposals and better solutions to address this problem. However, there is lack of in-depth studies on application of swarm-inspired methods at a large extent in optimization of cloud resource scheduling, but application of swarm optimization methods in cloud resource scheduling can be effective, considering the association of multiple factors.

This work will propose a combined dynamic resource allocation and data auditing approach to enhance the reliability of cloud infrastructure in an efficient manner. The work reported in this chapter has utilized ant colony optimization (ACO)-based resource sharing technique along with a novel data auditing scheme. The intended auditing process operates through a safe and sound data upload and verification system. Integrity of the auditing process is ensured through a message digest generated by Merkle tree representation of main data file. Algorithms have been implemented in Amazon EC2 framework and compared with the results of some existing research works reported in [12–14] to establish the efficacy of the proposed method.

The next section presents a preliminary discussion on basic ant colony optimization technique and the concept of Merkle tree. Section 4.3 defines the problem, and the proposed method is discussed in Section 4.4. Section 4.5 contains the results and discussion followed by the conclusion in Section 4.6.
4.2 Basic Concepts

4.2.1 Basic Outline of Ant Colony Optimization Method

ACO algorithm is an optimization technique based on the behavior of ants while searching for food or other resources [1]. Ants often move in groups, and they usually try to find out a suitable path between their original location and resource location by a process of evolution. In this biological behavior of ants, they deposit a pheromone along the paths they travel. Initially, they might explore multiple path options to reach their target. Subsequently, a few or more ants start to choose a particular path among the other options. These ants keep on depositing pheromone on the path they travel. Normally, increased amount of pheromone attracts other ants as well. Gradually, the path having dense deposition of pheromone will drag other ants to follow that path. Thus, an optimum path evolves. Dorigo first proposed an optimization technique based on an evolutionary approach in his research work [2]. This evolution was based on the movement behavior of ant colonies. Algorithmic implementation of this method often conceptualizes multiple ants as a multi-agent system where the agents are searching for some optimum results. Mathematical model of ACO suggests that the movement probability ($\Pi_{mn}^a$) of such agent ($a$) from a state $m$ to $n$ depends on two factors. These two factors are the impact of pheromone trail ($t_{mn}$) upon the connector between $m$ and $n$, and the logical need of an agent’s movement from $m$ to $n$, which is denoted by ($\delta_{mn}$).

$$t_{mn} = (1 - \rho) \cdot t_{mn} + \Sigma_k \cdot \Delta t^k_{mn} \quad (4.1)$$

This equation represents pheromone update model, where $\rho$ is the evaporation coefficient, $\Sigma_k \cdot \Delta t^k_{mn}$ is the summation of pheromone deposited by all $k$ number of ants on the connection between $m$ and $n$. $\Delta t^k_{mn}$ is the amount of pheromone deposited by $k^{th}$ ant, where Equation 4.2 shows the rate of $\Delta t^k_{mn}$.

$$\Delta t^k_{mn} = \begin{cases} 0, & \text{if } k^{th} \text{ ant not use } m \text{ and } n \text{ connector} \\ \frac{C}{L_k}, & \text{otherwise} \end{cases} \quad (4.2)$$

In the previous equation, $C$ is a constant and $L_k$ is the total travel length of the $k^{th}$ ant.

$$\Pi_{mn}^a = t_{mn}^\alpha \times \delta_{mn}^\beta \quad (4.3)$$

This equation represents the probability ($\Pi$) of agent ($a$) to select the path between $m$ and $n$. This is governed by the influence ($\alpha$) of deposited pheromone on path $mn$ ($t_{mn}$) and the influence ($\beta$) of the logical need of adopting the path ($\delta_{mn}$).
4.2.2 Concept of Merkle Tree

Merkle tree is a hash tree used for cryptographic purposes [3]. It follows the structure of complete binary tree [4]. Every leaf node of Merkle tree contains a data block. Subsequent parent node contains hash of that data contained within the data block. Cryptographic hash function [5] is used to hash these data blocks. A Merkle tree is shown in Figure 4.2. The four leaf nodes are representing four data blocks ($D_1$, $D_2$, $D_3$ and $D_4$). First, a cryptographic hash function is used to individually hash these data blocks. Penultimate ancestor blocks of the leaf nodes hold the hash of their respective child blocks (e.g. Hash ($D_1$) is $H_1$). Afterward, two adjacent hashed blocks are taken together from left to right direction, and they are hashed (e.g. parent blocks of $H_1$ and $H_2$ contain their hash $H_{12}$). Finally, the root contains the final hash block of whole tree. The root with $H_{1234}$ in Figure 4.2 contains Merkle hash of all data blocks.

4.3 Problem Definition

Data auditing within cloud data store is an important task. Optimized dynamic resource allocation to maintain the quality of performance is imperative as well. This proposed work will perform ACO-based dynamic resource allocation and data center auditing to deliver reliable services to cloud users.

Dynamic resource allocation phase has to provide an optimized resource allocation for cloud users. Resources are either infrastructural (CPU and peripheral devices) or service oriented (data center access or application software framework). An optimized solution is needed to allocate these resources
among several ongoing tasks depending upon their requisitions. Auditing is associated with data center services. This phase has to deal with the correctness and authenticity of the data stored within the cloud data server to protect cloud users from fraudulent adversaries. Data auditing is mainly performed by a TPA, but the task of TPA becomes challenging under dynamic access allocation. Auditing becomes more crucial to maintain the quality of service for cloud users.

4.4 Proposed Solution

The proposed solution deals with two different tasks. It performs dynamic resource allocation to cloud users and also verifies correctness and authenticity of the data stored in cloud servers. The later part is also known as cloud data auditing. Working principle of the proposed solution along with its primary basics has been discussed in this section.

ACO technique, which is a swarm-based evolutionary optimization process, is applied for efficient and dynamic resource allocation. On the other hand, Merkle tree-based cryptographic hash procedure is used for the purpose of auditing user data. Basic concepts of ACO and cryptographic hash operation based on Merkle tree are briefly described in the following subsections.

4.4.1 Dynamic Resource Scheduling

Resource allocation task primarily performs infrastructure management and service management. Cloud users submit their requests to a task resolver. The task resolver maintains a liaison with a resource manager. The resource manager undertakes the responsibility of resource scheduling [6]. Underlying concept of cloud computing is based on virtualization. The terms coined like task resolver or resource manager are virtual, and these are software based. This framework is illustrated in Figure 4.3. The figure illustrates that a user of cloud services generally interacts with the main task resolver of the cloud through a request submitter module. The task resolver categorizes these submitted requests and forwards them to the resource manager. The resource manager performs the task of scheduling or allocation based on some best practices, and the requisite services are granted to the users subject to satisfying other pre-conditions of availing services [6].

There are multiple resources in the form of CPU core, peripheral units, and some application software frameworks and data centers. The proposed model conceptualizes a system architecture where these units will primarily form an individual sub-cloud network which will be connected with the main cloud. Cloud service provider of the main cloud will control the overall operation. This system model is explained in Figure 4.4.
FIGURE 4.3
Schematic diagram of resource management.

FIGURE 4.4
Proposed resource distribution model.

This figure shows a conceptual view of the proposed resource distribution model. This diagram is mainly focused on the resource management part of the cloud service provider. The proposed work separates four major types of resources (CPU core, peripheral units, data centers and application software...
frameworks) into four different sub-clouds (namely, \( S_C^1 \), \( S_C^2 \), \( S_C^3 \) and \( S_C^4 \)). Each of these sub-clouds virtually contains many such resources registered under the same cloud service provider. This work also introduces four sub-schedulers for optimum scheduling of these four types of resources. These four sub-schedulers are ant CPU scheduler (ACS), ant process scheduler (APS), ant data scheduler (ADS) and ant software scheduler (ASS). ACO-based resource scheduling algorithm will work under these respective modules. Decisions of these schedulers are forwarded to resource manager for optimum and dynamic allocation of resources. Working principle of ACO-based resource scheduling along with its pheromone update function is illustrated in the next subsection.

4.4.2 Ant Colony Optimization-Based Resource Scheduling

4.4.2.1 Pheromone Function

Ant resource scheduling method for every resource type (\( \mu \)) initializes pheromone (\( t^k_{\mu} \)) on each instance (\( k \)) of the said resource type (\( \mu \)) upon receiving its requisition by any user (\( u \)) through the resource scheduler. Pheromone deposition on every \( k^{th} \) instance of a particular resource (\( \mu \)) is represented by the following equation:

\[
t^k_{\mu} = (1 - \rho) \cdot t^k_{\mu} + \Delta \cdot t^{i-1}_{\mu}
\]

\[
\rho^k = \frac{x^k}{y_{\mu}}
\]

In Equation 4.4, \( \rho \) is the standard pheromone evaporation. Evaporation rate for \( k^{th} \) ant (\( \rho^k \)) is given in Equation 4.5. This is a ratio of the number of allocations (\( x^k \)) presently under use for \( k^{th} \) instance of the said resource type to the total number of allocations under use for resource type \( \mu \), represented by (\( y_{\mu} \)). If there are a total \( i \) number of \( \mu \)-type resources, then \( \Delta \cdot t^{i-1}_{\mu} \) represents pheromone deposited on each of other (\( i1 \)) ants in the present allocation cycle. This deposition value will be 0 if the resource is already allocated to at least one ongoing process and will be (\( \theta/\omega \)) if it is free. \( \theta \) is a reputation parameter for a resource instance, which is constant and based on the previous performance feedback of that resource. \( \omega \) is the total number of idle instances of resource type \( \mu \) during that allocation cycle.

4.4.2.2 Decision on Resource Scheduling

Decision (\( \Pi \)) on the final selection of \( k^{th} \) resource instance (\( \mu^k \)) will be governed by Equation 4.6. Selection is based on the factor of two parameters. One parameter is related to pheromone trail represented by (\( (t^k_{\mu})^{\alpha} \)), where \( \alpha \) is an influence and reputation of that \( k^{th} \) instance (\( (\delta^k_{\mu})^{\beta} \)), which is based on its earlier performance feedback. \( \beta \) is the influence of that feedback.

\[
\Pi^k_{\mu} = (t^k_{\mu})^{\alpha} \times (\delta^k_{\mu})^{\beta}
\]
Overall ACO-based scheduling process is further illustrated in Algorithm 1:

**Algorithm 1: ACO-based cloud resource scheduling**

A request is received in resource manager of CSP. The request is forwarded to respective ant resource ($\mu$) scheduler. The ant resource scheduler connects to sub-cloud of the said resource ($S^\mu$). Virtual ant is initialized on all active instances of resource ($\mu$) to compute pheromone following Equation 4.4. Reputation ($\delta^k_\mu$) of each active instance $k$ is computed. Selection of resource ($\mu$) instance ($k$) is performed through equation 4.6.

### 4.4.3 Auditing of User Data

Major use of cloud computing is associated to its service as huge data centers [7]. A group of users store various data files on cloud data storage for future use. These files are used afterward either by the same users or by a group of other users associated with them. Users of these data files are normally classified into data owners and clients. Data owners store data files on cloud through their authenticated access. Later clients utilize these files for various purposes [8]. As the control and ownership of the files that are transferred to cloud storage is given to a third party, their confidentiality and reliability become major concerns [7]. Confidentiality refers to the concern of data management. Cloud service providers need to ensure that anyone other than the owner or its authorized users won’t be able to tamper the files stored in cloud or infringe their contents [9] and only authorized users will get on demand reliable access to these files [10].

The proposed method suggests a data authentication mechanism during any types of data transactions. These transactions may be either upload of user-owned files into the cloud storage servers or subsequent access of those files. This proposed auditing process generates an initial entry-level identity for the files when they are attempted to upload on the server. An owner attempting to upload any file first needs to submit that file to a virtual cloud file manager (CFM). This virtual CFM is an application associated with data upload process. CFM will perform an initial check regarding structural orientation of the file. Then, the file will be split into equal-sized blocks. These blocks will be treated as leaf nodes of Merkle tree (described in Section 4.2) to generate a message digest through Merkle hash. This message digest is unique for every new file. This message digest will be stored under the profile of the owner who has uploaded the file. This method is illustrated in Figure 4.5.

Subsequently, when any user attempts to access that file, another message digest is generated from the then version of the file. This is called user copy of message digest. This user copy of digest and previously saved owners copy of
digest will be sent to a TPA to compare and adjudge the integrity. If both these
digests found to be the same, then access will be granted to the requesting
user. Otherwise, the system will assume that the file has been intruded and
corrupted.

After successful integrity check, CFM will grant permission to the user to
access the file either in write mode or in read mode. The user will only be
able to view the file in read-mode access. On the other hand, the user will
be given the permission to modify the file in write-mode access. After gaining
write-mode access, the user will become the recent owner of the file and the
file will be saved with a new message digest. This new message digest will be
tagged with the recent user. Workflow of this access process is illustrated in
Figure 4.6.

4.5 Results and Discussion

This proposed system was fully implemented on Amazon EC2 [11] platform.
Virtual setup was done on Microsoft Windows Server instance. Initial simula-
tion involved uploading of some data files on the data servers following the
process of message digest generation. The proposed Merkle tree method was used
to generate this message digest. The other end of the system implemented login
by client users in demand of some peripheral and data request. These resources
were allocated using ACO-based resource scheduling algorithm. Performance
of auditing was measured in terms of following two metrics.

4.5.1 Metadata Generation Time for Data Files
of Varying Sizes

In this module, files of different sizes were taken into the process of uploading.
System-generated times to compute its metadata at the time of upload were
recorded. These times were recorded in milliseconds, and the files sizes are
given in kilobytes. Performance details are given in Table 4.1. This table also
contains the performance records of the existing works in [12] and [13] on the
same parameters. A graphical representation of the observation is given in
Figure 4.7.
FIGURE 4.6
Auditing of data file access by user.

TABLE 4.1
Time Taken to Generate Metadata Information on Different File Sizes

<table>
<thead>
<tr>
<th>File Sizes (Kilobytes)</th>
<th>Proposed Method</th>
<th>[12]</th>
<th>[13]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,000</td>
<td>90</td>
<td>200</td>
<td>360</td>
</tr>
<tr>
<td>5,000</td>
<td>220</td>
<td>360</td>
<td>570</td>
</tr>
<tr>
<td>10,000</td>
<td>400</td>
<td>620</td>
<td>1,200</td>
</tr>
<tr>
<td>25,000</td>
<td>560</td>
<td>850</td>
<td>1,620</td>
</tr>
<tr>
<td>50,000</td>
<td>700</td>
<td>940</td>
<td>1,900</td>
</tr>
<tr>
<td>100,000</td>
<td>950</td>
<td>1,150</td>
<td>2,500</td>
</tr>
</tbody>
</table>
4.5.2 Data Verification with Varying Numbers of Tenants

This process records the time taken for data verification when a client requests the access of certain data files. Data verification process checks the integrity of requested data files under different volumes (numbers) of active tenants (clients). Observations are recorded in Table 4.2 along with a comparative performance with the method proposed in [13]. Verification of files integrity measures the average time consumed to take the decision to give access to one requested file. However, files sizes are not taken into consideration while computing this average time. A graphical representation is given in Figure 4.8.

ACO-based dynamic resource allocation task was implemented with the configuration of data centers as depicted in Table 4.3.

**TABLE 4.2**

Time Taken to Verify File Integrity under Different Volumes of Users

<table>
<thead>
<tr>
<th>Volume of Users (number)</th>
<th>Average Time Consumed (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Proposed Method</td>
</tr>
<tr>
<td>5,000</td>
<td>2,250</td>
</tr>
<tr>
<td>10,000</td>
<td>2,675</td>
</tr>
<tr>
<td>25,000</td>
<td>3,210</td>
</tr>
<tr>
<td>30,000</td>
<td>3,540</td>
</tr>
</tbody>
</table>
Implementation of the resource allocation task recorded two different performance metrics.

4.5.3 Successful Resource Allocation with Different Volumes of Active Users

This implementation recorded the percentage (%) of successful resource allocation under different volumes of active users. Observations are recorded in Table 4.4 along with a comparative performance analysis with the work proposed in [14]. Experimental results are also graphically shown on a comparative analysis in Figure 4.9.

4.5.4 Balanced Allocation of Server Resources

According to Table 4.3, the maximum number of available servers is \((500 \times 15) = 7,500\). This section records the average utilization of servers with 25,000 active users under varying counts of servers. Here the average utilization represents the percentage of servers allocated with at least one job request. The count of servers under different service providers varies in real-time situation.
FIGURE 4.9
Comparative on success rate (%) in resource allocation between the proposed method and [14].

Challenge in resource allocation increases when the number of available servers is less due to some reasons. Table 4.5 records a performance metrics on allocation when the count of servers varies due to different reasons. It also contains a performance of work [14] under parallel simulation context. Figure 4.10 shows the comparative analysis.

TABLE 4.4
Success Rate (%) in Resource Allocation with Different Volumes of Clients

<table>
<thead>
<tr>
<th>Active Users</th>
<th>Proposed Method</th>
<th>[14]</th>
</tr>
</thead>
<tbody>
<tr>
<td>5,000</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>10,000</td>
<td>100</td>
<td>97</td>
</tr>
<tr>
<td>25,000</td>
<td>98</td>
<td>92</td>
</tr>
<tr>
<td>30,000</td>
<td>98</td>
<td>88</td>
</tr>
</tbody>
</table>

TABLE 4.5
Average Utilization of Servers under Varying Counts of Active Servers

<table>
<thead>
<tr>
<th>Active Servers</th>
<th>Proposed Method</th>
<th>[14]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,000</td>
<td>90</td>
<td>40</td>
</tr>
<tr>
<td>5,000</td>
<td>65</td>
<td>35</td>
</tr>
<tr>
<td>10,000</td>
<td>34</td>
<td>35</td>
</tr>
<tr>
<td>15,000</td>
<td>22</td>
<td>27</td>
</tr>
<tr>
<td>20,000</td>
<td>17</td>
<td>25</td>
</tr>
</tbody>
</table>
4.6 Conclusion

Data auditing and resource allocation are two major activities performed under cloud computing system. Proper functioning of these two tasks is very important for any cloud service provider and cloud framework to sustain in the market as a trusted and reliable service provider. These two activities are closely associated in leveraging the commitment made under service-level agreement (SLA). The present work has proposed a new solution for these two major activities associated with cloud computing. A Merkle tree-based message digest generation technique has been used to facilitate the auditing of the data stored in the cloud storage server during upload and subsequent access operations. Furthermore, an ACO-based dynamic resource allocation scheme has also been proposed for smooth and efficient resource allocation on cloud. These two techniques have been simulated in Microsoft Windows Server virtual instances on Amazon EC2 platform. The proposed auditing technique has been implemented to measure its efficiency in terms of the time taken by the technique to generate message digest during data file upload and the time taken for validation during file access under different volumes of clients. Experimental results show that the performance of the proposed auditing technique is notable and better than a few recent works reported in literature. The ACO-based resource allocation technique has also been implemented to check its proficiency in terms of successful resource allocation and distributing resource requests among available servers. Performance of the proposed ACO-based technique is better than a proposal of one recently published research report. It has been observed that distribution of resource requests among the available servers is wider in the proposed technique. However, a drawback of this distribution may be the power utilization, which has not been explored in this study. Engaging a more number of servers under a lesser load may cause
high consumption of power. Still the benefit of wider distribution is reduced
failure rate during a high-demand situation when the number of clients and
their requests is more. Future work might explore the scope of optimized power
utilization under this ACO-based resource allocation scheme. Also a swarm
intelligent algorithm-based key generation protocol can be used to implement
a public–private key-based digital signature system to protect the privacy and
integrity of the uploaded data.

Bibliography

colony of cooperating agents, *IEEE Transactions on Systems, Man,


[3] R. C. Merkle, A digital signature based on a conventional encryption func-
tion, Advances in Cryptology CRYPTO ’87, *Lecture Notes in Computer


tions: Recent design trends and security notions, *In Short Paper Pro-
cedings of Inscrypt 10*, China, Science Press of China, pp. 133–150,
2010.

[6] E. Meriam and N. Tabbane, A survey on cloud computing schedul-
ing algorithms, *Proceedings of 2016 Global Summit on Computer and

[7] J. Spillner, J. Müller, and A. Schill, Creating optimal cloud storage sys-
tems, *Future Generation Computer Systems*, vol. 29(4), pp. 1062–1072,
2012.

secure and dependable cloud storage based on user defined workflows,*
Proceedings of 2015 2nd IEEE Conference on Cyber Security and Cloud

[9] H. Xiong, X. Zhang, W. Zhu, and D. Yao, CloudSeal: End to-end con-
tent protection in cloud-based storage and delivery services, 96 LNICST,


5

TS-GWO: IoT Tasks Scheduling in Cloud Computing Using Grey Wolf Optimizer

Laith Abualigah
Amman Arab University

Mohammad Shehab
Aqaba University of Technology

Mohammad Alshinwan
Amman Arab University

Hamzeh Alabool
Saudi Electronic University

Hayfa Y. Abuaddous
Amman Arab University

Ahmad M. Khasawneh
Amman Arab University

Mofleh Al Diabat
Al Albayt University

CONTENTS

5.1 Introduction ................................................................. 128
5.2 Related Works ............................................................. 129
  5.2.1 Challenges ............................................................ 133
5.3 System Model ............................................................. 133
  5.3.1 Multi-objective Design Model .................................... 134
  5.3.2 Solution Encoding ................................................ 136
  5.3.3 Gray Wolf Optimizer for Solving the TS .................... 136
    5.3.3.1 Inspiration Source ....................................... 137
    5.3.3.2 Mathematical Model and Algorithm ................. 137
5.4 Experimental Results and Discussions ................................. 140
  5.4.1 Evaluation Measurements ....................................... 140
Cloud computing (CC) is the usual prominent solution to deliver large-scale and elastic IT resources (e.g., CPU, Me, and storage) as a service over the internet [2,35]. CC provides a huge pool of IT resources that need to be managed carefully. This is because the performance of CC highly depends on the efficiency of management of cloud service resources [17]. One of the main problems that directly affect the performance of cloud services is tasks scheduling (TS) [15,54]. Decision-making with regard to allocation of the best resources for executing tasks with the consideration of quality of service (QoS) criteria is important not only for meeting cloud users’ requirements but also for improving the performance of the whole cloud service. However, TS based on different QoS criteria is described as an NP-hard problem. For example, queries of tens of thousands of end users with different QoS requirements (e.g., performance and cost) need to be distributed across thousands of servers to be served [14].

In the cloud-computing environment, power consumption becomes a critical bottleneck for computer hardware and network infrastructure [19]. Hence, cloud resources require to be designated to save energy and reduce execution time. Consequently, TS and capacity balancing are vital to increasing the performance of cloud service utilizing inadequate resources. The type of TS in CC significantly affects the energy consumption of a cloud data-center. Thus, many researchers have proposed different heuristics [52].

In 2014, a new population-based meta-heuristic algorithm, namely gray wolf optimizer (GWO), was proposed by Mirjalili et al. [36]. GWO imitates the grey wolves’ hunting mechanism, which comprises three main steps: starting by tracking the prey, then encircling it, and finally attacking it. This algorithm has shown excellent results in solving several optimization problems in several fields [46]. Many problems related to, for example, data classification [13], feature selection [4], information retrieval [11], dimension reduction [5], data clustering [9], text clustering [6,7], text analysis [10], and benchmark functions [8] can be solved using GWO. There are also other algorithms that can be used to solve these problems [3,48].

In this regard, the arrangement time of tasks that are working on resources is a critical task. Allocating of several tasks with different requirements to be served with a suitable cloud resource is also a daunting and crucial task. This is because resources and tasks in CC are all diverse. Therefore, a multi-objective optimization scheduling model based on gray wolf optimization (GWO) algorithm, called TS-GWO, is proposed. Gray wolf optimization (GWO) method
is chosen as it is identical to the combinatorial optimization problem. This method has advantages of enhancing the performance of the TS model and improving the validity of the results obtained. This is due to its capacity to provide high performance even in unknown, challenging search spaces by selecting the optimal solution. Therefore, GWO is applied in the current research to measure the resources and then select the optimal resource for a specific task taking QoS criteria into consideration. The effectiveness of the proposed TS-GWO is assessed based on measurements: Mm values and average cost values.

This chapter contributes to the TS in CC context by:

- Proposing a multi-objective optimization scheduling model based on gray wolf optimization (GWO) algorithm
- Reviewing the more related work in this domain
- Validating the performance of the proposed TS model.

This chapter is organized as follows: Section 5.2 presents the related works. Section 5.3 presents the system model. Section 5.4 presents experimental results and discussions, and finally the conclusion and future work are presented in Section 5.5.

### 5.2 Related Works

Zuo et al. in [54] introduced a resource cost system for CC by determining the order of jobs on devices with more features. This system builds a connection between the client cost and the demand cost; moreover, client cost demand, and Mm will be the constraints in optimizing problems. Enhanced ant colony algorithm suggested evaluating the performance and the demand cost. To assess the effectiveness of the proposed system, several factors have been considered such as cost, Mm, violation time, and resource utilization. The experimental results demonstrate that the proposed system shows better results compared with other systems.

Reference [15] present a new version of the particle swarm optimization (PSO) algorithm for TS allocated in CC based on availability, reliability, transportation time, performance time, round-trip time, and transmission cost. The proposed method is called load balancing mutation PSO (LBMPSO). LBMSO has achieved the reliability of the cloud system by using the available resources rule task that fails to designate. The results show a better performance compared with other algorithms. Li, Yibin, et al. suggest a system to reduce the total energy expenditure for smart phones to deal with CC. This method is called dynamic voltage scaling (DVC), which supports power management by reducing the supply voltage and repetition of processors. Li, Yibin, et al. propose an algorithm, called energy-aware dynamic TS (EDTS), to minimize
the consumption of energy system for mobile phones. The results show that the proposed work substantially decreases energy expenditure [31].

To reduce the computation cost of CC, the tasks re-allocated are used to the available cloud resources. Reference [50] present a modified PSO (MPSO) algorithm to enhance cloud resource allocation based on price and time. The experiment results show a significant improvement in performance with normal PSO, in terms of utilization, speed, and effectiveness. An algorithm to schedule tasks is proposed in [12] using VM in the cloud environment with reduced time and increased system throughput. This work uses a combination of multi-objective optimization (MOO) and a PSO algorithm called MOPSO for cloud resource allocation. The experimental results show a decrease of 20% and 30% in execution time and waiting time, respectively. Moreover, the algorithm shows an increase of up to 40% in throughput.

In Ref. [47] introduce an algorithm correlates benefit-fairness method depend on weighted-fair queuing system for solving cloud TS problems. The proposed algorithm grouped the optimization problem constraints at a minimum cost and deadline. Then, it applied effective optimization and preference of fairness queue (low, mid, high). The simulation results based on CloudSim simulation show better performance and fairness at the preference level. To overcome the TS challenges in CC, [42] introduce a priority-based scheduling optimization approach. All tasks are grouped based on their priority. To optimize the resource allocation, two constraints have been considered, namely, cost and round-trip time. The results show that the proposed work enhances the allocation operations by decreasing the average round-trip time and cost.

Sreenu and Sreelatha in [49] present a multi-objective algorithm based on whale optimization algorithm (WOA), called W-Scheduler. The fitness value is defined as a multi-objective algorithm by determining the cost of CPU ration and Me. Based on the price and Mm, the WOA will find the optimal scheduling of tasks in the VMs. The proposed algorithm is compared with various algorithms such as PBACO, SPSO-SA, and SLPSO-SA. Experimental results demonstrate that the W-Scheduler shows outstanding performance with a minimum cost and Mm. Based on the ant colony algorithm (ACO), which has a strong convergence rate, [32] propose a genetic algorithm (GA-ACO) for TS in cloud. The simulation results show that the proposed method is considered as the efficient TS method in CC. To optimize the cost and Mm of CC scheduling, a variant of endless PSO algorithm is proposed, named Integer-PSO (IPSO) [18].

Azad and Navimipour in [16] introduce an algorithm based on ant colony and social optimization algorithm, to optimize the energy consumption and the Mm in the cloud environment. The experimental outcomes in the Azure platform show that the introduced algorithm has a significant performance in term of Mm and energy consumption. To solve the multi-QoS problem in cloud scheduling, [22] propose a method with several constraints like security, time-consuming, reliability, and expense in the scheduling process. The technique combines the genetic algorithm (GA) with the ACO algorithm, namely
MQoS-GAAC. The GA algorithm will calculate the initial pheromone values for ACO. Then, ACO is used to find out the optimal resource. Also, to minimize the Mm in TS for CC, [51] propose to use the ACO algorithm. A novel resource scheduling method is proposed in [37] for CC. This work adopts ACO to assign tasks to cloud VMs, namely slave ACO (SACO). The proposed work solves optimization problems by avoiding slave ants that use long ways which may be miscalculated by leaders. Moreover, this work uses diversity and support plans of slave ants. The experimental results show significant performance in cloud server utilization. Based on a small position value rule of the PSO algorithm, [26] introduce an algorithm to reduce the cost of CC processing. The proposed work is compared with the normal PSO, and the experimental results show that the enhanced PSO is more suitable for CC than the normal PSO.

Ramezani and Hussain in [45] developed a multi-objective model based on multi-objective particle swarm optimization (MOPSO) algorithm for optimizing TS and QoS including response time and service cost. To implement and evaluate their model, they extended the Jswarm and Cloudsim packages. In this work, the Jswarm package was extended to multi-objective Jswarm (MO-Jswarm) package by converting the PSO algorithm to MOPSO algorithm. After that, they applied the MO-Jswarm in the Cloudsim toolkit as the task scheduling algorithm. Their simulation results showed that the proposed model significantly increases the QoS. Navimipour and Milani in [40] proposed the cuckoo search algorithm (CSA) to address task scheduling problems in a homogeneous cloud infrastructure. However, the performance of the CSA was not compared with other algorithms such as GA and bee colony. Bitam et al. in [20] proposed the bees’ life algorithm (BLA) for cloud scheduling that overtakes GA in terms of execution time. Choudhary and Peddoju in [21] presented a scheduling algorithm that shows a significant improvement in cost and task completion time over sequential assignment. Reference [25] proposed a GA-based load balancing technique for task-level scheduling in Hadoop MapReduce, which outperforms the FIFO algorithm and in terms of delay scheduling and total cost time.

Guo-ning et al. in [24] suggested an optimized algorithm based on genetic simulated annealing algorithm to complete TS in an inefficient way. In their paper [29], Kaur and Sharma presented an improved particle swarm optimization (IPSO) algorithm, which consumes less execution time compared to PSO and simulated annealing algorithms. Due to the rapid growth of mobile devices, various applications have emerged which require high computing power and low latency. Thus, new network architectures have appeared such as fog computing and multi-access edge computing (MEC). Liu et al. in [33] proposed a latency-optimal TS policy for MEC based on Markov decision process (MDP) theory. This policy has achieved a significant reduction in the execution delay compared to the greedy scheduling policy. More related studies can be found in [3,30,41,44,52], and more details are shown in Table 5.1.
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Description</th>
<th>Problem</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC</td>
<td>Introduced a resource cost system for CC by determining the order of jobs on devices with more features</td>
<td>Jobs in cloud computing</td>
<td>Zuo et al. [54]</td>
</tr>
<tr>
<td>PSO</td>
<td>Presented a new version of the PSO algorithm for TS and allocation in CC</td>
<td>Allocation in cloud computing</td>
<td>Awad et al. [15]</td>
</tr>
<tr>
<td>EDTS</td>
<td>Proposed an algorithm to minimize the energy system energy-aware dynamic TS (EDTS) consumption for mobile phones</td>
<td>Mobile phones</td>
<td>Li et al. [31]</td>
</tr>
<tr>
<td>PSO</td>
<td>Pso optimization algorithm for task scheduling on the cloud computing environment</td>
<td>Task scheduling</td>
<td>Tarek et al. [50]</td>
</tr>
<tr>
<td>CSSA</td>
<td>Chaotic social spider algorithm for load balance aware task scheduling in cloud computing</td>
<td>Task scheduling</td>
<td>Xavier et al. [52]</td>
</tr>
<tr>
<td>GSA</td>
<td>Genetic and static algorithm for task scheduling in cloud computing</td>
<td>Task scheduling</td>
<td>Matos et al. [35]</td>
</tr>
<tr>
<td>LBS</td>
<td>Location-based secured task scheduling in cloud</td>
<td>Allocation in cloud computing</td>
<td>Basu et al. [17]</td>
</tr>
<tr>
<td>MMS</td>
<td>Multi-objective virtual machine selection for task scheduling in cloud computing</td>
<td>Task scheduling</td>
<td>Naik et al. [39]</td>
</tr>
<tr>
<td>OUR</td>
<td>Optimized utilization of resources using improved particle swarm optimization-based task scheduling algorithms in cloud computing</td>
<td>Task scheduling</td>
<td>Kaure et al. [29]</td>
</tr>
<tr>
<td>DCT</td>
<td>Delay optimal computation task scheduling for mobile-edge computing systems</td>
<td>Mobile-edge computing systems</td>
<td>Liu et al. [33]</td>
</tr>
</tbody>
</table>
5.2.1 Challenges

Significant difficulties emerge in the CC field when scheduling tasks/jobs to the devices. Currently, there are no methods that can address all critical factors associated with the job scheduling dilemma. The conditions needed for designating the resources raises when the amount of jobs increases. Consequently, optimization methods are not meeting the requirements of cloud cities with a high volume of information. Variety in time-alignment is considered as a complicated process during TS. Most of the current methods solve issues in job scheduling by setting jobs. In unusual circumstances, the expected arrangement was less than the original workloads of the operation, which leads to evil performance of job scheduling based on optimization algorithms. The unexpected complications in evaluation of workloads also affect the remaining parts of the scheduling method [39,53].

5.3 System Model

This area depicts the systemization of the suggested TS component within the cloud ecosystem [27]. In this chapter, we describe a framework that demonstrates assignment planning of CC. The errand supervisor gets the assignments from many users. In this model, the process includes multiple tasks that are arranged and scheduled optimally by the task manager. Each cloud has several VMs, and each one can perform some jobs.

The clients/users yield the assignment demands to the errand director. The assignment chief oversees the database to store each client task. The assignment chief organizes the client errands/tasks and gives the possibility of the assignment to the client. The assignment director contains the data approximate to the state of VMs. The assignment director provides these errands with demands to the errand scheduler. Assignment scheduler could be a gadget that presents the need for approaching chores. The errand scheduler analyzes the (Me) necessity, complications, due date, and the desired resources of the jobs. The cloud environment contains numerous physical devices. The VM display within the electrical device can handle many assignments. The errand scheduler apportions errands to the VMs displayed within the cloud ecosystem.

Consider that there are 100 physical machines in a cloud ecosystem and each electrical device comprises 10 VMs. This could be described as follows:

$$\text{Cloud}, M = \{P_1, P_2, P_3, \ldots, P_{100}\}$$  \hspace{1cm} (5.1)

where $C$ denotes to the cloud and $P_1, P_2, P_3, \ldots, P_{100}$ denotes the physical machines displayed within the cloud. The following condition can express the physical machine $P_1$: 
Swarm Intelligence for Cloud Computing

Tasks = \{T_1, T_2, T_3, \ldots, T_i, \ldots, T - 100\} \quad (5.2)

where $T_1$ and $T_2$ are the first and second errands, respectively. $T_i$ denotes the assignment number $i$, and $T_{100}$ denotes the 100 jobs. The job supervisor gives the assignments to the controller. Any errand given by the client includes adjustable parameters that are taken, demand, cost demand, due date, and the wanted demand. Consequently, the errand controller prioritizes the errand and plans it in like manner. The prerequisite, complications, due date, and the desired demand of the assignment $T_i$ are analyzed by the scheduler and given to the VM for arranging the tasks [1,35].

As shown in the figure, the term $CU_i$ denotes the take a toll of CPU of the assignment $T_i$ which was characterized by the clients, $MU_i$ denotes the fetched of the Me of assignment $T_i$ characterized by the clients, $DU_i$ denotes the due date for the errand $T_i$, and $BU_i$ is the demand fetched of $T_i$.

5.3.1 Multi-objective Design Model

The multi-objective design [54] to solve the TS in the VMs is represented here. The multi-objective design determines the cost estimations of Me and CPU of the group the given VMs provided in the CC ecosystem and determines the demand price. The demand price is determined by calculating the cost value of CPU ration and Me together [39]. Later, the fitness value is determined by combining the demand cost and the Mm value of the arrangement rule.

The fitness function value is determined to evaluate the candidate solutions and choose the optimal one. In the beginning, the cost values of Me and CPU are measured. The subsequent equations are utilized to determine the cost values of CPU and Me in the given VM $V_j$.

$$C(x) = \sum_{j=1}^{\text{\mid VM\mid}} C_{\text{cost}}(j) \quad (5.3)$$

where $C_{\text{cost}}(j)$ denotes the cost value of CPU in the VM number $j$ ($V_j$), $\mid \text{VM\mid}$ denotes the total number of VMs. Next, the $C_{\text{cost}}(j)$ is determined by the following equation:

$$C_{\text{cost}}(j) = C_{\text{base}} \times C_i \times C_{ij} \times C_{\text{Trans}} \quad (5.4)$$

where $C_{\text{Trans}}$ denotes the base cost, $C_{ij}$ denotes the CPU of the VM $V_j$, and $C_{ij}$ denotes the time where the task $T_i$ is preformed in the device $R_j$. $C_{\text{Trans}}$ denotes the transmission cost of the CPU. Here, $C_{\text{base}}$ and $C_{\text{Trans}}$ are constants (which are the same as in Equations (5.5) and (5.6)).

$$C_{\text{base}} = 17/h \quad (5.5)$$

$$C_{\text{Trans}} = 0.0005 \quad (5.6)$$
The cost function value of the Me is determined by the following equation:

\[ M(x) = \sum_{j=1}^{\lvert VM\rvert} M_{\text{cost}}(j) \]  \hspace{1cm} (5.7)

where \( M_{\text{cost}}(j) \) denotes the cost value of the Me of the VM number \( j \) \( V_j \), \( |VM| \) denotes the total number of the given VMs. Then, \( M_{\text{cost}}(j) \) is determined by the following equation:

\[ M_{\text{cost}}(j) = M_{\text{base}} \times M_j \times t_{ij} \times M_{\text{Trans}} \]  \hspace{1cm} (5.8)

where \( M_{\text{base}} \) denotes the post cost value of the Me, \( M_j \) denotes the Me value of the VM \( V_j \), and \( t_{ij} \) denotes the time in which the job \( T_i \) is prepared in the resource \( R_j \). \( M_{\text{Trans}} \) denotes the transportation price value of the Me. The amounts of \( M_{\text{base}} \) and \( M_{\text{Trans}} \) are two fixed variables (which are the same as in Equations (5.9) and (5.10)).

\[ M_{\text{base}} = 0.05 \text{GB/h} \]  \hspace{1cm} (5.9)

\[ M_{\text{Trans}} = 0.5 \]  \hspace{1cm} (5.10)

Then, the demand cost value of the agent can be determined using the price need of each VM for both CPU ratio and Me.

\[ B(x) = C(x) + M(x) \]  \hspace{1cm} (5.11)

where \( B(x) \) denotes the demand price value of the agent, \( C(x) \) denotes the price value of the CPU, and \( M(x) \) denotes the cost value of the Me. Next, the fitness function value is determined as follows (5.12):

\[ H(x) = F(x) + B(x) \]  \hspace{1cm} (5.12)

where \( F(x) \) denotes the Mm function, which must be less than or equal to the finishing of the given task. Mm is determined by the following equation:

\[ F(x) = \leq \sum_{i=1}^{\lvert T\rvert} D_i \]  \hspace{1cm} (5.13)

where \( D_i \) denotes the deadline for the given task \( T_i \). In Equation (5.12), \( B(x) \) denotes the demand cost value (it comes from the demand price value) of the tasks that include the CPU ratio and Me cost, and it must be less than or equal to the agent’s demand price. The demand price function is determined by the following equation:

\[ B(x) = \leq \sum_{i=1}^{\lvert T\rvert} B_i \]  \hspace{1cm} (5.14)

where \( B_i \) denotes the agent’s demand price of given tasks as \( T_i \).
5.3.2 Solution Encoding

The main object of TS is to designate a group of tasks (i.e., 50) to the given VMs (i.e., 10) in accordance with the estimated time of the assignment and the demand cost of the functions. Suppose that there are 50 tasks, and each one is initialized to the conditions from 1 to 10, as shown in Table 5.2. If the first position is 1, then this task 1 is designated to the VM (i.e., V1). If the first position is 8, then the task number 1 is allocated to the (VM) (i.e., V8). Furthermore, all the given tasks are assigned to the given VMs (i.e., V1V10).

5.3.3 Gray Wolf Optimizer for Solving the TS

This section explains the proposed method (GWO for solving the tasks scheduling in CC, called TS-GWO) for scheduling tasks using the given number of VMs in the CC ecosystems. The proposed method is introduced with a new multi-objective design model and the GWO. Figure 5.1 presents the description of the proposed TS-GWO for arrangement in cloud ecosystems. The multi-objective design determines the cost use of Me and CPU of all the given VMs and also determines the demand price function through calculating the cost values of Me and CPU [23,28]. The fitness function is determined by calculating both the Mm and the demand price function together. Using the fitness, the designate of the jobs to the given VMs.

<table>
<thead>
<tr>
<th>Table 5.2</th>
<th>Solution Representation for the GWO</th>
</tr>
</thead>
<tbody>
<tr>
<td>T₁</td>
<td>T₂</td>
</tr>
<tr>
<td>V₁</td>
<td>V₃</td>
</tr>
</tbody>
</table>

FIGURE 5.1
Description of the proposed TS-GWO for TS problem.
5.3.3.1 Inspiration Source

In 2014, a new population-based metaheuristic algorithm, namely GWO, was proposed by Mirjalili et al. [36]. GWO imitates the grey wolves’ hunting mechanism, which comprises three main steps: starting by tracking the prey, then encircling it, and finally attacking it. Alpha (α), beta (β), delta (δ) and omega (ω) refer to the grey wolves’ types.

α refers to the leader which is responsible for determining the hunting and sleeping times. Then follows β which is the deputy leader that aids α in making decisions and ensures that the other wolves are doing their tasks. It’s worth to mention that β is the strongest candidate to become the leader after the current leader gets old or dies. The third level is δ which serves as connectors between α, β, and ω. In other words, δ monitors ω and sends the information to α and β (i.e., spy). Finally, ω is at the lowest level of the hierarchy, which includes the other wolves except α, β, and δ.

Here, we illustrate and describe the main phases of grey wolf hunting process [34]:

- Approaching the prey
- Pursuing, encircling, and harassing the prey until it stops moving
- Attacking the prey.

5.3.3.2 Mathematical Model and Algorithm

This section shows the mathematical models of GWO phases, for example, social hierarchy, tracking, encircling, and attacking prey. Then follows the pseudo-code of GWO algorithm.

1. Social hierarchy

As mentioned in Section 5.3.3.1, the hierarchical order of gray wolves’ types is α, β, δ, and ω. So, the wolves can be mathematically modeled as α for the best solution, β for the second-best solution, δ for the third-best solution, and finally ω for the worst solution.

2. Encircling the prey

As mentioned previously, in hunting, grey wolves chase and encircle the prey. Equations 5.15 and 5.16 present the mathematical model for this phase.

\[
\bar{D} = \left| \bar{C} \cdot \bar{X}_p - \bar{X}(t) \right| \tag{5.15}
\]

\[
\bar{X}(t + 1) = \bar{X}_p(t) - \bar{A} \cdot \bar{D} \tag{5.16}
\]

where \( t \) and \( t+1 \) refer to the current and next iterations, respectively. \( \bar{A} \) and \( \bar{C} \) indicate coefficient vectors (see Equations 5.17 and 5.18), \( \bar{X} \) denotes the location vector of a grey wolf, and \( \bar{X}_p \) denotes the location vector of the prey.

\[
\bar{A} = 2\bar{a} \cdot \bar{r}_1 - \bar{a} \tag{5.17}
\]
\[ \vec{C} = 2 \cdot \vec{r}_2 \] (5.18)

where the values of \( \vec{a} \) are linearly reduced from 2 to 0 over the course of iterations, as well as \( r_1 \) and \( r_2 \) are random vectors in [0, 1].

It can be noticed that the location of the grey wolf \((X, Y)\) can be updated based on the prey location \((X^*, Y^*)\). It’s worth to mention that setting the values of \( \vec{A} \) and \( \vec{C} \) vectors can help to achieve a better location. Moreover, the potential updated locations of a grey wolf are shown using 3D space. In this case, vectors \( r_1 \) and \( r_2 \) allow the grey wolves moving to any location between the points.

3. **Hunting**

The next step after estimating the prey location is hunting, where the hunting technique of the grey wolf is based on the locations of \( \alpha, \beta, \) and \( \delta \), knowing that the prey’s location isn’t determined exactly. Thus, the values of \( \alpha, \beta, \) and \( \delta \) are considered as the best solutions. In the case of \( \omega \), they’ll update their location according to the three obtained solutions. The following equations illustrate the mathematical model of the hunting mechanism:

\[ D_{\alpha} = |C_1 \cdot \vec{X}_\alpha - \vec{X}|, D_{\beta} = |C_2 \cdot \vec{X}_\beta - \vec{X}|, D_{\delta} = |C_3 \cdot \vec{X}_\delta - \vec{X}| \] (5.19)

\[ \vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \cdot D_{\alpha}, \vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \cdot D_{\beta}, \vec{X}_3 = \vec{X}_\delta - \vec{A}_3 \cdot D_{\delta} \] (5.20)

\[ \vec{X}(t + 1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \] (5.21)

Here, we show the procedures for updating the search locations of the wolves using two-dimensional search space which is based on the locations of \( \alpha, \beta, \) and \( \delta \). It can be observed that the final location is determined randomly in a circle using the locations of \( \alpha, \beta, \) and \( \delta \).

4. **Attacking the prey (utilization)**

This phase is also called the exploitation phase. After determining the prey’s location, it stops moving. On the other hand, the gray wolves change their locations. The mathematical modal represents this process by decreasing the value of \( \vec{A} \) which leads to decrease of \( \vec{A} \). We illustrate that \(|\vec{A}| < 1\) allow the wolves to attack across the prey. Therefore, the grey wolves (i.e., \( \omega \)) change their locations depending on the locations of \( \alpha, \beta, \) and \( \delta \). These processes work well, but are prone to stuck in the local optima. So, it is necessary to enhance the exploitation search of GWO.

5. **Searching for prey**

This subsection presents the mathematical model for increasing the exploration search of GWO algorithm. To achieve this goal, we
should use the value of $a$ with a random value $> 1$ or $<-1$; in this case, the grey wolves (i.e., $\omega$) were forced to be distant from the prey [38]. We show the inequality $|A| > 1$. In the end, the GWO algorithm is finished by satisfying an end criterion.

Figure 5.2 shows the pseudo-code of the GWO algorithm. The following points describe theoretically how to fix optimization problems:

- Usage of the social hierarchy helps the algorithm to keep the best solutions gained so far over the track of an iteration.
- The surrounding technique determines that a circle-shaped neighborhood includes all solutions that can be expanded to higher dimensions.
- $A$ and $C$ help selected solutions to have hyper-spheres with various random radii.
- The hunting mechanism allows the selected solution to define the probable location of the prey.
- The exploitation and exploration can be controlled by adjusting the values of $\alpha$ and $A$.
- Using a large value of $A$ helps to improve the exploration search (i.e., $|A| \geq 1$). However, a small value of $A$ is used to enhance the exploitation search (i.e., $A < 1$).
- $\alpha$ and $C$ are considered as main parameters of the GWO algorithm.

---

**FIGURE 5.2**  
Pseudo-code of the GWO algorithm. [43]
5.4 Experimental Results and Discussions

This section shows the experimental results of the proposed TS in CC using GWO (TS-GWO) for arranging jobs over the practical devices in the CC ecosystems and the analysis of the introduced algorithm with the other similar approaches, like improved ACO-based scheduling approach (PBACO) [54], self-adaptive learning PSO-based scheduling approach (SLPSO-SA) [55], standard PSO-based scheduling approach (SPSO-SA) [55], and WOA-based scheduling approach (W-Scheduler) [49].

The experimentation of the proposed TS-GWO is implemented in a personal computer (PC) with processor Core i5 and 8GB Me using Windows 10 operating system. The introduced method (TS-GWO) is applied using Java deprogramming with Cloudsim, and the effectiveness of the proposed TS-GWO is assessed using measurements: Mm values and average cost values.

5.4.1 Evaluation Measurements

The evaluation metrics examined for investigating the performance of the proposed TS-GWO method are Mm and cost.

- Mm denotes the total time required for completing all the given assignments. The Mm of the task scheduler methods must be the smallest (minimum Mm).
- The cost denotes the total cost required for programming the tasks to the practical devices (VM).

5.4.2 Results and Discussions

The proposed TS-GWO is assessed based on the testing measurements (Mm and average cost) and also compared with the basic GWO.

Figures 5.3–5.5 show the Mm values using 25 iterations when physical machines = 10 for the population sizes (Pop) of 10, 15, and 20; the Mm values using 50 iterations when virtual machines = 20 for the Pop of 10, 15, and 20; and the Mm values using 50 iterations when physical devices = 30 for the Pop of 10, 15, and 20, respectively.

As shown in Figure 5.3, the numbers of assignments are 100, 200, 300, and 400. In terms of 25 iterations, the proposed TS-GWO got the best Mm results when the tasks are 100 using 20 solutions. When the tasks are 200, the proposed TS-GWO got the best Mm results using 20 solutions. When the tasks are 300, the proposed TS-GWO got the best Mm results using 10 solutions. When the tasks are 400, the proposed TS-GWO got the best Mm results using 20 solutions. From this figure, we conclude that the proposed method (TS-GWO) obtains all the best results when physical machines = 10.
FIGURE 5.3
The Mm values using 25 iterations when physical machines = 10 for the Pop of 10, 15, and 20.

FIGURE 5.4
The Mm values using 50 iterations when physical machines = 20 for the Pop of 10, 15, and 20.

in comparison with other similar purposes, especially when the number of solutions is 20.

As shown in Figure 5.4, the numbers of assignments are 100, 200, 300, and 400. In terms of 50 iterations, the proposed TS-GWO got the best Mm results when the tasks are 100 using 10 solutions. When the tasks are 200,
FIGURE 5.5
The Mm values using 50 iterations when physical machines = 30 for the Pop of 10, 15, and 20.

The proposed TS-GWO got the best Mm results using 15 solutions. When the tasks are 300, the proposed TS-GWO got the best Mm results using 10 solutions. When the tasks are 400, the proposed TS-GWO got the best Mm results using 10 solutions. From this figure, we conclude that the proposed method (TS-GWO) obtains all the best results when physical machines = 30 in comparison with other similar purposes, especially when the number of solutions is 10.

As shown in Figure 5.5, the numbers of assignments are 100, 200, 300, and 400. In terms of 50 iterations, the proposed TS-GWO got the best Mm results when the tasks are 100 using 15 solutions. When the tasks are 200, the proposed TS-GWO got the best Mm results using 10 solutions. When the tasks are 300, the proposed TS-GWO got the best Mm results using 15 solutions. When the tasks are 400, the proposed TS-GWO got the best Mm results using 15 solutions. From this figure, we conclude that the proposed method (TS-GWO) obtains all the best results when physical machines = 30 in comparison with other similar purposes, especially when the number of solutions is 15.

Figures 5.6–5.8 show the demand cost values using 25 iterations when VMs = 10 for the Pop of 10, 15, and 20; the demand cost values using 50 iterations when VMs = 20 for the Pop of 10, 15, and 20; and the demand cost values using 50 iterations when physical devices = 30 for the Pop of 10, 15, and 20, respectively.

As shown in Figure 5.6, the numbers of assignments are 100, 200, 300, and 400. In terms of 25 iterations, the proposed TS-GWO got the best demand cost for all tasks sizes (i.e., 100, 200, 300, and 400). From this figure, we conclude
that the proposed method (TS-GWO) obtains all the best demand costs for all tasks sizes (i.e., 100, 200, 300, and 400) when physical machines = 10 in comparison with other similar methods, especially when the number of solutions is 25.
The demand cost values using 50 iterations when physical machines = 30 for the Pop of 10, 15, and 20.

As shown in Figure 5.7, the numbers of assignments are 100, 200, 300, and 400. In terms of 50 iterations, the proposed TS-GWO got the best demand cost for all tasks sizes (i.e., 100, 200, 300, and 400). From this figure, we concluded that the proposed method (TS-GWO) obtained all the best demand costs for all tasks sizes (i.e., 100, 200, 300, and 400) when physical machines = 30 in comparison with other similar methods, especially when the number of solutions is 50.

As shown in Figure 5.8, the numbers of assignments are 100, 200, 300, and 400. In terms of 25 iterations, the proposed TS-GWO got the best demand cost for all tasks sizes (i.e., 100, 200, 300, and 400). From this figure, we conclude that the proposed method (TS-GWO) obtains all the best demand costs for all tasks sizes (i.e., 100, 200, 300, and 400) when physical machines = 30 in comparison with other similar methods, especially when the number of solutions is 50.

Figures 5.9–5.11 show the average price values using 25 iterations when VMs = 10 for the Pop of 10, 15, and 20; the average cost values using 50 iterations when VMs = 20 for the Pop of 10, 15, and 20; and the average cost values using 50 iterations when physical devices = 30 for the Pop of 10, 15, and 20, respectively.

As shown in Figure 5.9, the numbers of assignments are 100, 200, 300, and 400. In terms of 25 iterations, the proposed TS-GWO got the average cost for all tasks sizes (i.e., 100 tasks when POP = 15, 200 tasks when POP = 15, 300 tasks when POP = 15, and 400 tasks when POP = 20). From this figure, we conclude that the proposed method (TS-GWO) obtains all the average costs
FIGURE 5.9
The average cost values using 25 iterations when physical machines = 10 for the Pop of 10, 15, and 20.

FIGURE 5.10
The average cost values using 50 iterations when physical machines = 20 for the Pop of 10, 15, and 20.

for all tasks sizes when physical machines = 10 in comparison with other similar methods, especially when the number of solutions is 25.

As shown in Figure 5.10, the numbers of assignments are 100, 200, 300, and 400. In terms of 50 iterations, the proposed TS-GWO got the average cost for all tasks sizes (i.e., 100 tasks when POP = 20, 200 tasks when POP = 20, 300 tasks when POP = 20, and 400 tasks when POP = 10). From this figure,
we conclude that the proposed method (TS-GWO) obtains almost the average costs for all tasks sizes (i.e., 100, 200, and 300) when physical machines = 30 in comparison with other similar methods, especially when the number of solutions is 50.

As shown in Figure 5.11, the numbers of assignments are 100, 200, 300, and 400. In terms of 25 iterations, the proposed TS-GWO got the average cost for all tasks sizes (i.e., 100 tasks when POP = 20, 200 tasks when POP = 20, 300 tasks when POP = 20, and 400 tasks when POP = 20). From this figure, we conclude that the proposed method (TS-GWO) obtains all the average costs for all tasks sizes when physical machines = 30 in comparison with other similar methods, especially when the number of solutions is 50.

5.5 Conclusion and Future Work

This chapter introduces the TS method for scheduling the given jobs or tasks to the available VMs in CC ecosystems. The proposed method is introduced with a new multi-objective design and the GWO algorithm, called TS-GWO. In the beginning, the multi-objective design measures the fitness function by measuring the cost values of the CPU and Me. Next, the demand price value is computed by calculating the price function value of the CPU ratio and Me. Eventually, the fitness function is determined by calculating the Mm value and the demand price value.
The GWO is introduced for finding the optimal scheduling of a group of tasks over the available VMs. The GWO thinks that the currently existing option is optimum to determine the optimal nest option using the best-obtained solution so far. The performance investigation of the proposed scheduling method (TS-GWO) is carried out, and this method is compared with the existing basic GWO for the evaluation analysis using the Mm and cost measure values. According to the empirical outcomes, we conclude that the introduced method can arrange the given jobs to the available VMs optimally, and it produces the minimum Mm values and the minimum average cost values compared with the other.

In future works, we will improve GWO’s abilities in effectively finding the best solution by hybridizing it with other algorithm components. Moreover, the proposed algorithm can be implemented to solve other optimization problems.

Bibliography


Fact-Checking: Application-Awareness in Data Centre Resource Management

Aaqif Afzaal Abbasi
Foundation University

Almas Abbasi
International Islamic University

Mohammed A. A. Al-qaness
Wuhan University

Ammar Hawbani
University of Science and Technology of China

Ahmed A. Ewees
Damietta University

Yousif A. Alhaj
Wuhan University of Technology

CONTENTS

6.1 Introduction ...................................................... 154
6.2 Application-Awareness in Cloud Data Centers .................. 155
6.3 Background ........................................................... 155
6.4 Research Directions .............................................. 158
   6.4.1 Application-Aware Data Center Resource Management ............................................. 158
   6.4.2 Topology of Inference Schemes ......................... 159
       6.4.2.1 Managing Cloud Services Admission .......... 159
       6.4.2.2 Intelligent Resource Administration and Resource Optimization ...................... 159
6.5 Conclusion ............................................................. 160
Bibliography ............................................................ 160
6.1 Introduction

Computing capabilities in the world of IT are increasing at a rapid pace with constant expansion and growth. The ever-growing IT industry aims to make optimum use of the latest computing technologies to keep pace with emerging and ever-increasing computation demands. In this regard, cloud computing [2,34] is of fundamental importance. It is the art of managing network servers and services on a distributed platform. Cloud computing and related services are often available on requirement basis and can be purchased on a “pay-as-you-go” model. This means that the user does not have to pay extra charges for the use of computing services they do not need and only pay for those they utilize. Similarly, different cloud providing services leverage cloud computing solutions to handle the requirements of resource-intensive IT applications.

The main enabling technology behind the development of cloud computing is virtualization. Virtualization [15,28], is the process of separating the physical computing devices. This results in the creation of “virtual” devices. These devices are later managed to perform some computing tasks. The technique is used to create virtual versions of computing hardware, storage devices, and operating systems. The method was first adopted in the 1960s with an aim to divide a systems resource allocation among various applications [5]. Since then, the term has expanded in use and has now garnered much wider recognition.

System virtualization is the development of a virtual machine (VM). A VM works just like a real computer system [1,21]. The software working on these VMs is separated from the hardware system (see Figure 6.1). Therefore, development of cloud computing is related to the evolution of distributed systems-related disciplines. By using cloud computing, users can benefit from various distributed system-related technologies. The cloud computing paradigm reduces operational costs by helping users to focus only on utilizing IT services instead of wasting it on worrying about the ever-changing IT technologies. The agile, efficient, and cost-effective nature of cloud technology led it to be a major technology of the IT infrastructure.

![FIGURE 6.1](image)

VMs and operating systems in a virtualized environment
Swarm intelligence (SI) is a concept used in mathematics and computing to derive the collective and decentralized trends in objects and systems. SI often represents the decentralized behavior of objects. The concept relies on artificial intelligence (AI) and was introduced in 1989 by Gerardo Beni and Jing Wang for management of robot technology. SI concepts have often intermingled with various fields of computing and technology to execute resource administration in cloud computing and data centers. The SI concepts consider various communicating elements of cloud computing and data centers as members of a swarm environment. By bringing in intelligence in data centers, the SI concepts play the same role (i.e. improving the resource administration concepts of data center elements).

In general, the idea of SI couples huge networks using loosely coupled technologies, e.g. Internet of Things and cloud computing. SI is a process of bringing intelligence to computing paradigm. In this context, SI can be used to enable the AI functions in data mining and pattern recognition of a particular trend of data hosted on cloud-based systems.

This chapter presents a brief introduction to cloud computing, application-awareness, and related technologies that influence high-performance computing environments.

6.2 Application-Awareness in Cloud Data Centers
Application-awareness [29] is a networking concept that improves a system’s capability to keep information about applications to optimize its functionality. The application-aware network makes the best use of available information by using centralized software-defined networking (SDN) features. This enables the network to effectively handle resource allocation challenges in performing various operations [18,25]. Application-aware storage systems often rely on intelligent built-in patterns. If the storage mechanism facilitates application usage conditions, optimized data layouts and caching behaviors can be administered to reach the needed quality of service (QoS) levels.

6.3 Background
Cloud computing [27] technology appeared as one of the important computing prototypes built with the objectives of reducing investments and delivering services on pay-per-usage concepts. Despite attaining success in numerous domains of computing services, cloud computing technologies are still developing. In this regard, there are various workgroups, and industry-standard
regulatory authorities are keep trying to address the gaps emerging in the future development and growth. Cloud computing concepts are also vital in helping companies to reduce costs related to infrastructure development and maintenance [8]. Due to the flexible nature of adoption, it allows industries and enterprises to improve their applications to run faster, and also improve their manageability with lower maintenance costs. It also enables cloud management to adjust their resources to meet the ever-fluctuating and un-predictable demands of cloud business models. By using a typical “pay as you go” model, the administrator asks users to choose an appropriate cloud-pricing model. The recent easy availability of high-performance networks and low-cost computing and storage devices has led to the widespread adoption of hardware-depended virtualization function features, cloud-oriented service-centric architectures, and utility computing, which accelerates the explosive development of cloud computing technology.

Cloud computing delivers services through service-oriented architectures, also called service models. In this regard, National Institute of Standards and Technology (NIST) [27] has proposed three primary service models for cloud services, namely Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS), as shown in Figure 6.2.

**FIGURE 6.2**
Cloud service models.
These models are used to deliver abstraction services for cloud functions. According to the Internet Engineering Task Force (IETF), a cloud service is described as a service that offers computing infrastructure for VMs. The same services and resources are provided to subscribers as well. In this regard, IaaS refers only to the services that can be abstracted from the infrastructure details such as the computing machine, its resources, and its location. These services run through VMs by a hypervisor. A large pool of hypervisors residing in a cloud provide support to a large number of VMs. This helps service providers by improving their capacity to scale services as per user needs. The PaaS model offers a platform for development purposes. It also enables application developers to operate their software solutions on a cloud platform without involving users in the complexities of buying and managing an IT infrastructure. In the SaaS service model, users gain system resources access, while cloud service providers manage infrastructure and system applications. The SaaS model is often referred to as “on-demand software”. It is also charged on a pay-per-use basis or with a fixed subscription rate.

SDN is a concept in data center networking, where network managers can manage network services by using the abstraction of lower-level functionality. SDN is widely recognized as the future of networking [31]. SDN concept involves the separation of a networks control plane from its data plane. This improves the ease of administration and allows remote access to the data center switches for improved traffic management [32], as shown in Figure 6.3. This feature distinguishes SDN-enabled networks from other

![SDN Concepts](image)
network administration concepts. SDN-enabled networks are vital in dealing with multi-tiered applications as they require definite bandwidth allocation between server instances to assure that user transactions are accomplished within a predefined timeframe under approved service-level agreements (SLAs). The continuation of manual systems for configuration via command-line interfaces (CLIs) has restricted networking from exploring the full features of virtualization. It also increases the operational costs, requires time-consuming network update processes, and is susceptible to introduction of errors [36]. These issues were briefly discussed in [16]. Since their inception, SDN and its first industrial standard OpenFlow [30] have tried to solve many network-related issues such as the network functions virtualization (NFV) [14,17], data centers and cloud networking [26], acceleration in value-added networks, services delivery [3], and network management platforms [9,19].

The software-defined data center (SDDC) concept extends the virtualization concepts to all data center resources. In an SDDC, all functional components of data center, i.e. internetworking, storage, security, and traffic management, are virtualized. According to IT experts, SDDCs will witness a strong market demand in near future [23]. SDDC concept actually comprises a variety of data center concepts that provide, operate and administer data operations using an application program interface (API). The biggest advantage of SDDCs is that cloud service providers need no longer to depend on specialized hardware equipment. All the applications in an SDDC are deployed and run through “logical applications”. A large number of vendors, such as VMware, NEC, Microsoft, IBM, Cisco, Juniper, and Hewlett-Packard, are working to develop and establish SDDC approaches. The most commonly cited advantage of SDDCs is the improved efficiency brought about by extending virtualization features throughout the data center, resulting in better administration of network and policy-based services. Several efforts are underway to make the standardization process uniform and to bring it in line with industry standards.

6.4 Research Directions

Application-awareness can be improved in the following major streams.

6.4.1 Application-Aware Data Center Resource Management

Use of SDN to re-engineer data center infrastructure helps to maximize the data center resource utilization. This helps to address issues related to network management systems [22] and network configuration syntax [11], and it
often delivers improved network management functions. While the industry is currently witnessing major developments in the areas of network-path optimization [20,24], traffic routing [13,33], latency management [12], and network functions management [35],¹ there is a huge research market for the development of performance fine tuning and related network applications especially designed for data centers. Therefore, the absolute necessity to improve data center infrastructure management has been agreed upon by experts who believe that data center infrastructure management systems are essential for the development of data center efficiency.

6.4.2 Topology of Inference Schemes

Inference schemes use internet exchange providers (IXPs) to provide services and facilities to internet service providers (ISPs). Similar concepts are used for content delivery networks (CDNs) where internet traffic is exchanged between two networks containing featured content. Implementing the SDN concepts through SDDCs at IXPs offers new avenues for solving network domain routing issues. The software-defined exchange (SDX) controller [7,10] is a vital development in this regard as it provides a sequential composition of policies that change inter-domain routing.

6.4.2.1 Managing Cloud Services Admission

Due to the massive amounts of data transfers in cloud computing systems, the huge amount of traffic generated by these services creates serious challenges for cloud infrastructure and service users. Cloud admission control solutions are even more important due to mobile-based applications and services. State-of-the-art research directions regarding admission control of cloud services have been addressed in [4,6]. SDNs are widely used for policy enforcement and isolation to boost cloud service admission in cloud data centers. Services admission control issues in data centers can be resolved through various application-awareness model-based schemes.

6.4.2.2 Intelligent Resource Administration and Resource Optimization

SDNs are used to administer data center resource usage in general. The SI optimization techniques such as ant colony optimization and grey wolf optimization techniques may be used to improve the resource utilization pattern in cloud-based environments.

6.5 Conclusion

Application-awareness-related challenges in data centers are gaining popularity. However, there is lack of studies in literature on this exciting area of interest in terms of cloud computing. IT and computing professionals over the years have been developing systems that support data center resource consolidation methods. In traditional data centers, traffic flows are assigned with different priorities. This means that the traffic flows dedicated to important applications will get more bandwidth. By doing so, these applications will be processed quickly and thus will spend less time in processing queues. Therefore, the traffic with higher priority will suffer lower data transfer delays. Using this simplistic approach may create issues in resource-hungry applications where several applications compete for bandwidth in the same class. This can also lead to congestion and poor application performance. In addition, this approach only addresses bandwidth allocation issues and misses out latency-related issues.

In view of the above, application-aware networking is a concept that refers to the network-dependent processing of application-layer information. This ensures that the network categorizes the content on the basis of application-layer attributes. This will help network managers to provide optimized application delivery to their clients, guaranteeing application performance, availability, and costs. Applications running in a conventional datacenter sometimes may not require enough bandwidth to justify the occupancy of a switching slot. Therefore, to ensure flexibility, all workloads should be flexible enough to move among servers freely. The same case applies to memory when it migrates outside VM jurisdiction to contact another network service.

In this research study, we try to fill the gaps that are considered as obstacles in adopting a methodical approach for implementing application-awareness concepts in data centers. The recent developments in SDN concepts and their implementation in data centers successfully have paved the way for a new generation of cloud networking paradigm and have also spawned innovation that has achieved key advances in several areas of network resource management. It is evident from these facts that application-awareness in data centers will continue to grow in the future.

Bibliography


[27] Peter Mell and Tim Grance The nist definition of cloud computing. 2011.


Bio-inspired Optimization Algorithms for Multi-objective Task Scheduling in Cloud Computing Systems

A. S. Ajeena Beegom
College of Engineering

M. S. Rajasree
Government Engineering College

CONTENTS

7.1 Introduction ...................................................... 165
7.2 Related Work ..................................................... 167
  7.2.1 Contribution of Chapter ................................ 168
7.3 Mathematical Formulation ..................................... 168
  7.3.1 Multi-objective Optimization ............................ 169
7.4 Solution Methodologies ......................................... 169
  7.4.1 Particle Swarm Optimization Technique .............. 170
  7.4.2 Artificial Bee Colony Algorithm ....................... 171
  7.4.3 Genetic Algorithm ...................................... 174
  7.4.4 Ant Colony Optimization Algorithm .................. 174
7.5 Experimental Settings ......................................... 177
7.6 Results and Discussion ........................................ 179
7.7 Conclusion ........................................................ 183
Bibliography ............................................................ 183

7.1 Introduction

Cloud computing is a recent advancement in the technology field, which allows leasing of computing resources over the Internet for meeting the computing/storage requirements of an individual, small-scale companies, and research organizations. In public cloud platforms, the ownership of the
resources resides with an external party, whereas in private cloud systems, it is owned and managed internally within an organization or a university. Benefits of using public cloud systems include reduced infrastructure cost, reduced overhead for the end user, and pay only for the components that are used in the given amount of time. Here scheduling plays an important role, which is the process of mapping the jobs or tasks to the resources available so that the quality of service is guaranteed with a minimum cost of usage.

In a public cloud platform, there are hundreds and thousands of tasks to be scheduled per unit time. Consider the problem of scheduling 20 tasks to 20 virtual machines (VM) considering a single objective function, say minimizing cost. There are $20! = 2,432,902,008,176,640,000$ possible solutions to the problem. Examining all these solutions to find the best solution giving minimum cost is even computationally infeasible. For a cloud computing environment, the variability in task length is also unpredictable. This is because some of the applications that are hosted on the cloud platform may be highly active for a certain number of days of a month or a few hours in a day. Here the pattern of task arrival rate, as well as task length, shows a normal or log-normal distribution trend, but for other days or hours, it shows an entirely random distribution [10]. To model such scenarios, studies on optimal scheduling algorithms are needed since they determine the efficient utilization of resources in the cloud, thereby reducing infrastructure and management cost at cloud service provider’s (CSP) end and desirable throughput at the end user’s side.

Scheduling large-scale scientific applications to distributed and heterogeneous resources such that certain objective functions such as execution time of tasks (makespan) or running and management cost (cost in short) are optimized subject to constraints such as storage and processing power, is a well-known problem in the NP-complete category. The problem is a constraint multi-objective optimization problem, where no single solution that is optimal for all the objectives exists. Instead, a trade-off solution can be found, with the property that solutions within this set cannot be improved further in any of the considered objectives without degrading another objective function value. Proposing algorithms and solutions to such problems, especially in the context of cloud computing, is a real challenge to the engineering and scientific community.

For task scheduling in the cloud computing platform, apart from running time or management cost, other considerations like network traffic, user satisfaction in terms of Quality of Service (QoS), and CSP’s profit are to be addressed. Many of the existing task scheduling research in cloud computing consider one factor alone as done in [5,6,9,12,14], and tries to find an optimal schedule based on that factor. But single-objective optimization solutions try to optimize one objective function value making another key factor to worse; hence, multiple objective functions characterizing the domain need to be considered. In this work, bi-objective optimization based on makespan at the user end and management cost at the CSP’s end for task scheduling
is considered. The cost includes computation cost, communication cost, the overall maintenance cost, and power consumption cost. The proposed solution methodology can be applied to any number of objective functions. We have focused on the scheduling of a large set of parallel independent tasks of different sizes, as they form the bottleneck in many of the scientific applications. The weighted sum approach for multi-objective optimization is used, and a comparative study on the performance of nature-inspired optimization algorithms such as Particle Swarm Optimization (PSO) algorithm, Artificial Bee Colony (ABC) algorithm, Genetic Algorithm (GA), and Ant Colony Optimization (ACO) algorithm is done. These algorithms are applied to the same sets of task data and are compared against Greedy Randomized Adaptive Search Procedure (GRASP), which is a list-based meta-heuristic search technique.

7.2 Related Work

A comprehensive study on the evolutionary computation approaches for resource scheduling in the cloud computing systems is presented by Zhan et al. [18]. A taxonomy of various approaches in the literature for task scheduling using GA, PSO, and ACO in the SaaS layer is presented in detail in this work. Since our work falls into scheduling for negotiations category, the literature review is focused on this type of work alone. Our previous work [2] uses PSO technique for bi-objective task scheduling in the cloud environment using weighted sum approach and proposed Integer-PSO technique. Here a method to find discrete values after the velocity and position update of particles is proposed. M. Feng et al. [7] use the PSO algorithm to solve resource allocation in cloud computing. They have considered total execution time, resource reservation, and QoS of each task as optimization objectives and used the pareto dominance principle to find optimal solutions. L Guo et al. [8] address the task assignment problem in cloud computing considering makespan and cost using PSO algorithm. E.S. Alkayal [1] applies the PSO technique using a ranking strategy for minimizing waiting time and maximizing system throughput.

A hyper-heuristic approach to scheduling in the cloud is proposed in [16], which uses an integration of different heuristics algorithms such as simulated annealing, GA, PSO algorithm, and ACO algorithm, and in each iteration of the algorithm, any one of these techniques is selected for finding the schedule. MOEA/D technique is used by X. Wang et al. [17] for bi-level multi-objective task scheduling which uses map-reduce framework. G.F. Elhady and M.A. Tawfeek [6] have done a comparison between ACO, PSO, and ABC algorithms for cloud task scheduling considering a single objective function. Application of the GA framework for task scheduling in cloud computing systems is addressed in our prior work [3]. Bi-objective optimization using NSGA-II for optimizing makespan and cost for hybrid cloud platform is done by V.A. Leena et al. [11].
7.2.1 Contribution of Chapter

1. Analysis of the suitability of bio-inspired optimization algorithms such as ABC algorithm, PSO algorithm, GA, and ACO algorithm for the cloud environment.

2. Study on multi-objective optimization for cloud task scheduling for parallel independent tasks considering makespan at the user’s end and cost at the CSP’s end, two conflicting objective functions.

3. Experimental analysis based on different data sets generated using an established procedure to characterize normal traffic and bursty traffic in the cloud environment.

7.3 Mathematical Formulation

For a better understanding of the mathematical model, the variables used are presented in Table 7.1 along with a brief description.

Assuming that an application consists of $N$ tasks which are independent, then $n$ out of $N$ are scheduled to $m$ VMs in each time window, where the value of $n$ is limited by $m$. To completely execute all tasks in the application, $k = \frac{N}{n}$ similar epochs are needed. For each type of VM instances say small, medium, large, etc., cost of usage and the processing power are different. Let $P_j$ represent the processing power of $j^{th}$ VM instance type, where $j$ ranges from 1 to $m$ and $E_j$ represents its execution cost for unit time. Given the task length $T$ needed to execute each task in a ‘small’-type VM, the optimization objectives for $N$ tasks are as follows:

\[
\begin{align*}
\text{Minimize} & \quad M = \sum_{p=1}^{k} \sum_{i=1}^{n,m} T_{pi} \times P_j \times z_{ij} \\
\text{Minimize} & \quad C = \sum_{p=1}^{k} \sum_{i=1}^{n,m} E_j \times T_{pi} \times P_j \times z_{ij}
\end{align*}
\]

(7.1) (7.2)

<table>
<thead>
<tr>
<th>TABLE 7.1</th>
<th>Variables and Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variables</td>
<td>Description</td>
</tr>
<tr>
<td>$N$</td>
<td>Total number of tasks</td>
</tr>
<tr>
<td>$m$</td>
<td>No. of virtual machines</td>
</tr>
<tr>
<td>$P$</td>
<td>Processing power</td>
</tr>
<tr>
<td>$T$</td>
<td>Estimated task length</td>
</tr>
<tr>
<td>$M$</td>
<td>Makespan function</td>
</tr>
<tr>
<td>$Z$</td>
<td>Objective function</td>
</tr>
</tbody>
</table>
with constraints

\[
 z_{ij} = \begin{cases} 
 1 & \text{if TASK}_i \text{ scheduled to VM}_j \\
 0 & \text{otherwise} 
\end{cases} 
\]  

(7.3)

\[ n \leq m \]  

(7.4)

and

\[ \sum_{i=1}^{n} z_{ij} = m \]  

(7.5)

Equation (7.3) ensures that any task in the task list is assigned to some VM, and Equation (7.4) for decision variable \( z_{ij} \) ensures that there exist a sufficient number of VMs to schedule each task. Constraint function (7.5) ensures that all tasks in the task list are scheduled.

7.3.1 Multi-objective Optimization

A multi-objective optimization problem consists of optimizing a vector of size \( n \) where the objective function \( F(x) = (f_1(x), f_2(x), \ldots, f_n(x)) \). Classical approaches to solving a multi-objective optimization problem include combining multiple objectives into a single objective function, by assigning weights to different objective functions [13,15]. The advantage is that they generate a single solution to the problem based on the assigned weights, and this approach is easy to solve. But the relative weight among objective functions needs to be known in advance. To find an optimal task schedule considering the two objective functions \( M \) (Equation (7.1)) and \( C \) (Equation (7.2)) simultaneously using the weighted sum approach, the optimization problem is represented using the following formula:

\[
\text{Minimize} \quad Z = \theta \ast C + (1 - \theta) \ast M 
\]  

(7.6)

where \( \theta \) represents the relative preference of one objective function over the other in the range [0, 1]. When \( \theta = 0 \), the optimization problem becomes that of minimizing \( M \) alone, and when \( \theta = 1 \), the problem becomes minimizing \( C \) alone, producing makespan-optimal task schedule and cost-optimal task schedule, respectively.

7.4 Solution Methodologies

The different optimization techniques used in this study and the corresponding algorithm to solve the task scheduling problem are explained in this section.
7.4.1 Particle Swarm Optimization Technique

PSO algorithm is one of the population-based stochastic optimization techniques. Initially, the PSO algorithm generates randomly a set of $N$ solutions called *particles* in the $d$-dimensional search space. Each *particle* is represented by a $D$-dimensional vector $X_i$ which stands for its position $(x_{i1}, x_{i2}, \ldots, x_{id})$ in space. Each *particle* maintains its position and velocity. It also maintains the best fitness value it has achieved so far during the search and the position that achieved this fitness (pbest). The PSO algorithm also maintains the best fitness value achieved among all *particles* in the swarm and the corresponding position (gbest). This algorithm was proposed initially for solving problems in the continuous domain. Since the task scheduling problem is in the discrete domain, Integer-PSO technique [2] is used in this work. The velocity ($v$) update, constrained by $v_{\text{min}}$ and $v_{\text{max}}$, and position ($x$) update mechanism of each *particle* are given in Equations (7.7)–(7.10).

\begin{align*}
v_{id}^{n+1} &= w \cdot v_{id}^n + c_1 \cdot r_1 \cdot (\text{pbest}_{id}^n - x_{id}^n) + c_2 \cdot r_2 \cdot (\text{gbest}_d^n - x_{id}^n) \quad (7.7) \\
Y_{id}^{n+1} &= \text{ceil}((x_{id}^n + v_{id}^{n+1}) \cdot \beta) \text{ where } \beta = 10^y \\
\text{Pos}_{id}^{n+1} &= (Y_{id}^{n+1}) \mod m \quad (7.9) \\
x_{\text{c}id}^{n+1} &= \begin{cases} 
\text{Pos}_{id}^{n+1} & \text{if } \text{Pos}_{id}^{n+1} > 0 \\
\frac{\text{Pos}_{id}^{n+1}}{m} & \text{otherwise} 
\end{cases} \quad (7.10)
\end{align*}

where $i = 1, 2, \ldots, N$, $n = 1, 2, \ldots, \text{max}$, $w$ is the inertia weight, $c_1$ and $c_2$ are acceleration coefficients, and $r_1$ and $r_2$ are two uniformly distributed random numbers in the interval $[0, 1]$. New position $x_{id}^{n+1}$ is obtained from $x_{\text{c}id}^{n+1}$ by removing duplicate entries, if any, in the assignment generating (Task, VM) schedule for the next iteration.

PSO-Based Task Scheduling Algorithm is given as Algorithm 1. Permutation encoding is used to represent each population. For example, $(5, 4, 1, 3, 2)$ represents a (Task, VM) assignment schedule showing Task$_1$ assigned to VM$_5$, Task$_2$ to VM$_4$, etc. In step 3–6 of the algorithm, $N$ initial populations (schedules) are generated randomly and are checked to see that all the constraints are satisfied. The velocity of all *particles* is also initialized. In steps 9 and 10 depending on task length $T_i$, processing power $P_j$ of VM$_j$ and cost $E_j$ of VM$_j$, the values of $M$ (Equations (7.1)) and $C$ (Equation (7.2)) are calculated. After appropriate scaling of these values, the value of $Z$ (equation (7.6)) of each of these populations is also calculated for the given $\theta$ value. From this set, pbest and gbest are assigned (steps 11–16).

Steps 17 and 18 of the algorithm generates new schedules from the current population using the velocity and position update rules (Equations (7.7)–(7.10)). If $(5, 4, 1, 3, 2)$ is one *particle* with pbest as $(1, 3, 4, 2, 5)$ and gbest as $(1, 3, 2, 4, 5)$ and if $c_1 = c_2 = 0.2$ and $w = 0.6$, $r_1 = 0.5$, $r_2 = 0.3$ and current velocity $v_{id}^n = 1.4$, then $v_{id}^{n+1} = (0.2, 0.68, 1.2, 0.8, 1.32)$ and
Algorithm 1 PSO-Based Task Scheduling Algorithm

1: Initialize $N$, max, $w$, $c_1$, $c_2$, $v_{\text{min}}$, $v_{\text{max}}$, $\theta$, High
2: $n = 1$, $i = 1$, $Z_{\text{gbest}} = \text{High}$, $\text{pbest}(n)(i) = \text{null}$, $\text{gbest} = \text{null}$
3: for ($i = 1$ to $N$) do
4: $P(n)(i) \leftarrow$ Generate random schedule
5: Initialize $v(n)(i)$ randomly in the range $[v_{\text{min}}, v_{\text{max}}]$
6: end for
7: while ($n < \text{max}$) do
8: for (each $P(n)(i), i \leq N$) do
9: Calculate $M(i)$ and $C(i)$ as per Equations (7.1)–(7.2)
10: $Z(i) = \theta \times C(i) + (1 - \theta) \times M(i)$
11: if ($Z_{\text{pbest}}(i) > Z(i)$) then
12: $\text{pbest}(i) = P(n)(i), Z_{\text{pbest}}(i) = Z(i)$
13: end if
14: if ($Z(i) < Z_{\text{gbest}}$) then
15: $\text{gbest} = \text{pbest}(i), Z_{\text{gbest}} = Z(i)$
16: end if
17: $v(n+1)(i) \leftarrow$ Update $v(n)(i)$ as per Equation (7.7)
18: $xc(n+1)(i) \leftarrow$ Update $xc(n)(i)$ as per Equations (7.8)–(7.10)
19: $x(n+1)(i0) \leftarrow$ Remove duplicate from $xc(n+1)(i)$
20: $P(n+1)(i) = x(n+1)(i), i = i + 1$
21: end for
22: $i = 1$, $n = n + 1$
23: end while
24: Output $\text{gbest}$

$Y_{id}^{n+1} = (5.2, 4.68, 2.2, 3.8, 3.32)$. If $y = 2$, then $xc_{id}^{n+1} = (5, 3, 5, 5, 2)$ which is changed to $(5, 3, 1, 4, 2)$ on duplicate removal to ensure that every VM is assigned a task (step 19). Other constraints are also checked for each particle, and if there are any violations on constraints, that particle is replaced with its previous iteration value else new position is assigned (step 20). Steps 7–23 are repeated until max iteration is reached. On completion, gbest is returned, as it gives the best schedule achieved among all the particles in the population in all generations.

7.4.2 Artificial Bee Colony Algorithm

ABC algorithm is another population-based meta-heuristics algorithm that simulates the foraging behavior of honey bees in nature. A honey bee swarm consists of food sources, employed bees, onlooker bees and scout bees. Each food source represents a solution to the optimization problem to be solved and have a fitness value representing its nectar amount. The solutions are subjected to repeated update through employed bees, onlooker bees and scouts
for a predetermined number of cycles. A candidate food position \( v_{ij} \) is computed from the current food position \( x_{ij} \) by employed bees using the following formula:

\[
v_{ij} = x_{ij} + \phi_{ij} \ast (x_{ij} - x_{kj})
\]  
(7.11)

where \( k, j \) and \( \phi \) are three random numbers in the range \([1, E]\), \([1, U]\) and \([-1, 1]\), respectively, with \( E \) as the number of employed bees. To get \( v_{ij} \) values as discrete quantities with \( m \) as the number of available VMs, the following equations are applied:

\[
Y_{ij} = \text{round}(|v_{ij}|) \mod m
\]  
(7.12)

\[
v_{ij} = \begin{cases} 
Y_{ij} & \text{if } Y_{ij} > 0 \\
m & \text{otherwise}
\end{cases}
\]  
(7.13)

Fitness value of the new solution is computed, and if it is better than the current food position, the employed bee replaces the current food position with the new position else retains the current food position. When all employed bees complete the update process, they share this information with onlooker bees (waggle dance). An onlooker bee chooses a food source probabilistically using equation (7.14), where \( N \) denotes the total number of bees and \( f \) their fitness.

\[
p_i = \frac{f_i}{\sum_{n=1}^{N} f_n}
\]  
(7.14)

When a food source is not profitable after a limit number of cycles that food source is abandoned. Now, scout bees are sent out in search for new solutions that enable exploration. The best food source found by all the bees is output as the solution to the problem on completion of maximum iterations.

ABC-Based Task Scheduling Algorithm is given in Algorithm 2. Steps 2–4 of the algorithm are used to generate bee positions randomly, satisfying the constraints mentioned in the mathematical model, and their nectar amounts are also calculated. Steps 8–16 find the new position and the corresponding nectar amount for each employed bee. For example, if \( 5, 4, 1, 3, 2 \) is the current solution \( x_i \) and if \( x_k \) is \((4,1,2,3,5)\) and \( \phi_i = 0.5 \), then \( v_{ij} = (5.5, 5.5, 0.5, 3, 0.5) \), which is converted to discrete values \((1, 1, 1, 3, 1)\) with \( m = 5 \) using the equations (7.12) and (7.13) from which duplicate entries are removed to get \((1, 2, 4, 3, 5)\). The nectar amount \( Z \) (Equation (7.6)) of each of these bees is computed for the given \( \theta \) value, and the position is updated if it finds to be better.

In steps 18–31, onlooker bees compute the probability of solution fitness considering its own fitness and fitness of all the bees in the swarm. If the computed value is higher than a predefined value, the position is updated else it will retain its previous position. In every iteration of the algorithm (limit = 1), a scout bee is the one that stores the solution with the maximum \( Z \) value. A new scout solution is then generated by randomly interchanging one or two (Task, VM) assignments from the current scout solution that helps
Algorithm 2 ABC-Based Task Scheduling Algorithm

1: Initialize $E$, $U$, $p_b$, $\theta$, max, $\phi$, $n = 1$, $i = 1$
2: $EB(n) \leftarrow$ Generate $E$ random schedules
3: $OB(n) \leftarrow$ Generate $U$ random schedules
4: Calculate $M$, $C$ and $Z(n)$ for each schedule /as in Equations (7.1), (7.2), (7.6)
5: $n = n + 1$
6: while $(n \leq$ max) do
7:      $i = 1$, $T_f = 0$
8:      for (each $EB(n)(i), i \leq E$) do
9:          $k = \text{random}(E)$
10:         $EB(n)(i) = EB(n - 1)(i) + \phi \ast EB(n - 1)(k)$
11:     $EB(n)(i) \leftarrow$ Apply Equations (7.12) and (7.13) on $(EB(n)(i))$
12:     Calculate $M(i)$, $C(i)$ and $ZEB(n)(i)$
13:     if $(ZEB(n)(i) > ZEB(n - 1)(i))$ then
14:         $EB(n)(i) = EB(n - 1)(i)$
15:     end if
16:      end for
17:      Sort $EB(n)$ based on $ZEB(n)$
18:      for $(i = 1$ to $E$) do
19:          $T_f = T_f + ZEB(n)(i)$
20:      end for
21:      for $(i = 1$ to $U$) do
22:          $T_f = T_f + ZOB(n - 1)(i)$
23:      end for
24:      for (each $OB(n)(i), i \leq U$) do
25:          $p = ZOB(n - 1)(i)/T_f$
26:          if $(p \geq p_b)$ then
27:              $OB(n)(i) = EB(n)(i)$
28:          else
29:              $OB(n)(i) = OB(n - 1)(i)$
30:      end if
31:      end for
32:      Scout$(n) \leftarrow EB(n)$ or $OB(n)$ with maximum $Z$
33:      Scout$(n) \leftarrow$ Interchange two positions of Scout$(n)$ randomly
34:      Calculate $M$, $C$ and $Z$ of Scout$(n)$
35:      Replace corresponding $EB(n)$ or $OB(n)$ with Scout$(n)$, $n = n + 1$
36:  end while
37:  Output Best among $EB(n)$ and $OB(n)$

for exploration, and the fitness value is calculated (steps 32–35). For example, if the scout bee is identified to be $(5,4,1,3,2)$, then exchanging 5 with 1 (position 1 with position 3) yields a new solution $(1,4,5,3,2)$, which is added to the solution set. Steps 6–36 are repeated for max iterations, and on completion,
the algorithm outputs the best bee position among all the bees as the solution to the optimization problem.

7.4.3 Genetic Algorithm

GA is based on genetic science and natural selection. It is also a population-based meta-heuristic search technique. It works under the principle of 'survival of the fittest'. The chance of being selected as the parent of the next generation depends upon the fitness value of individuals. At each iteration, selected parent solutions are recombined (crossover operation) to generate new offspring solutions. The mutation operator is then applied to a very small set of offspring solutions. The new solutions replace other solutions according to a replacement strategy.

Task scheduling using GA is given as Algorithm 3. A solution or a chromosome is composed of a sequence of \( n \) genes, where each gene represents a task to VM assignment, represented using permutation encoding. Initial solutions are randomly generated meeting constraints, and their fitness is evaluated (steps 2–3). Steps 6–19 of the algorithm select parent individuals for generating offsprings for the next generation probabilistically through Roulette wheel selection method. The scaling method applied here is exponential scaling. The selected chromosomes are recombined to produce offsprings. The recombination procedure adopted is one point crossover technique on 90\% of these individuals, where the crossover point is determined randomly on each generation (steps 21–23). For example, if the point of crossover selected is 2 and if the parent chromosomes are \( (5,4,1,3,2) \) and \( (4,1,2,3,5) \), then new offsprings are \( (5,4,2,3,5) \) and \( (4,1,1,3,2) \). The offsprings after duplicate removal are \( (5,4,2,3,1) \) and \( (4,1,5,3,2) \). Mutation operation is performed on a small set of offspring chromosomes (5\%) for exploitation. The mutation strategy adopted is an interchange of two randomly selected genes (step 24). For example, if \( (5,4,1,3,2) \) is the offspring chromosome that is selected for mutation, two positions are randomly selected, say 2 and 4, then the chromosome after mutation will be \( (5,3,1,4,2) \). These offspring chromosomes are evaluated to find the value of \( Z (\text{fitness}) \) for the given \( \theta \) value. Steps 25–27 of the algorithm removes duplicates and find the fitness value of each offspring chromosome. The replacement strategy adopted is generational elitism, where the best \( N \) solutions (solutions with a minimal \( Z \) value) of parent and offspring chromosomes are selected for the next generation (step 28). Steps 4–29 are repeated for \( \text{max} \) generations, and on completion, the best chromosome is output as the solution to the problem. An initial proposal on the above algorithm is presented in our earlier work [3].

7.4.4 Ant Colony Optimization Algorithm

ACO algorithm is a construction-based search algorithm for an optimization problem. They use the foraging behavior of ant colonies. Ants deposit a special
Algorithm 3 GA-Based Task Scheduling

1: Initialize $N$, $K$, $\theta$, max, $G$, $n = 1$, $i = 1$
2: $P(n) \leftarrow$ Generate $N$ random schedule
3: Calculate $M$, $C$ and $Z(n)$ for $P(n)$ as in Equations (7.1), (7.2), (7.6)
4: while $(n <= \text{max})$ do
5:  $\text{sum} = 0$
6:  for (each $P(n)(i), i \leq N$) do
7:   $sc(i) = \exp(Z(n)(i) \ast K)$, $\text{sum} = \text{sum} + sc(i)$
8:  end for
9:  for ($i = 1$ to $N$) do
10:  $pb(i) = sc(i)/\text{sum}$
11: end for
12: for ($i = 2$ to $N$) do
13:  $cpb(i) = cpb(i - 1) + pb(i)$ with $cpb(1) = pb(1)$
14: end for
15: for ($k = 1$ to $N$) do
16:  for ($i = 1$ to $N$) do
17:   $Pt(n)(k) \leftarrow P(n)(i)$ depending on random($N$) & $cpb(i)$
18:  end for
19: end for
20: for ($k = 1$ to $N$, $k = k+2$) do
21:  $i = 1$, $j = \text{random}(N)$
22:  Do crossover at $j$ on $Pt(n)(k)$ & $Pt(n)(k+1)$ to get $ch(k)$ & $ch(k+1)$
23: end for
24: $j = \text{random}(G)$, $k = \text{random}(G)$, Interchange ($ch(N : j)$, $ch(N : k)$)
25: for (each $ch(i), i \leq N$) do
26:  Remove duplicates from $ch(i)$, Calculate $M(i)$, $C(i)$ and $Z(i)$
27: end for
28: $P \leftarrow$ Best $N$ among $P(n)$ and $ch$ based on $Z$, $n = n + 1$
29: end while
30: Output $P$

kind of chemical substance called pheromones on the ground while moving towards a food source. Ants construct solutions, and these are used to perform a pheromone update to modify the pheromone trails. The pheromone trails affect the construction of solutions in the next iteration. Algorithm for task scheduling based on ACO technique is given as Algorithm 4. During a tour, the ants build the solution to the cloud task scheduling problem by adding one VM from a set of VMs for a particular task, and the process repeats until all tasks are assigned some VMs, thus completing a tour. Initially, one VM is assigned randomly for each ant (steps 3–5). Then, the selection of a particular VM for a given task is determined by a selection probability (steps 6–12 of the algorithm). Let $\tau_{ij}$ be the pheromone concentration on the path from $Task_i$ to $VM_j$ and $\eta_{ij}$ be the heuristic information associated on the same path, then
Algorithm 4 ACO-Based Task Scheduling Algorithm

1: Initialize $N_a$, $\alpha$, $\beta$, $\theta$, $\rho$, $Q$, $N$, max, $k = 1$, $S = NULL$
2: while ($k < \text{max}$) do
3: \hspace{0.5em} for ($i = 1$ to $N_a$) do
4: \hspace{1em} $S(i) \leftarrow$ Assign 1 VM randomly
5: \hspace{0.5em} end for
6: \hspace{0.5em} $t = 2$
7: \hspace{0.5em} while ($t \leq N$) do
8: \hspace{1em} for (each ant, $i = 1$ to $N_a$) do
9: \hspace{2em} $S(i) \leftarrow$ Choose next VM using Equation (7.15)
10: \hspace{1em} end for
11: \hspace{1em} $t = t + 1$
12: \hspace{0.5em} end while
13: Update pheromone trails using Equation (7.16)
14: Apply pheromone evaporation using Equation (7.17)
15: \hspace{0.5em} for ( $i = 1$ to $N_a$) do
16: \hspace{1em} Calculate $M(i)$, $C(i)$ and $Z(i)$ of $S(i)$
17: \hspace{1em} end for
18: $S^* \leftarrow$ Best solution from $S$
19: Update global pheromone evaporation using Equation (7.18)
20: \hspace{0.5em} $k = k + 1$
21: end while
22: Return $S^*$

the probability of selecting $j^{th}$ VM for Task$_i$ for ant $k$ is determined using the following formula:

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha * [\eta_{ij}]^\beta}{\sum_{l \in N_i} [\tau_{il}]^\alpha * [\eta_{il}]^\beta} \quad (7.15)$$

where $\alpha$ and $\beta$ are parameters that determine the impact of pheromones and heuristic information, respectively, and $l$ is the set of allowed VMs for ant $k$ in the current tour. The inverse of objective function value is taken as heuristic information ($\eta$). With the selection of a particular path, each ant lays a quantity of pheromone ($\Delta \tau_{ij}$) on that path, determined by Equation (7.16) where $Q$ is an adaptive parameter and $Z_{ij}$ is the objective function value for Task$_i$ on VM$_j$ (step 13). In step 14, the pheromone update mechanism defined by Equation (7.17) is done where $\rho$ is a parameter to control pheromone evaporation.

$$\Delta \tau_{ij} = \frac{Q}{Z_{ij}} \quad (7.16)$$

$$\tau_{ij} = (1 - \rho) * \tau_{ij} + \Delta \tau_{ij} \quad (7.17)$$

When all ants complete the solution construction process, the fitness of the solutions is evaluated and the best-found solution is updated from among all
the ants (steps 15–18). The $\tau_{ij}$ for the best solution path is again updated with a global perspective, using Equation (7.18), which helps to bias or broaden the search in the next iteration (step 19). The algorithm iterates for $\text{max}$ number of times (steps 7–21) and on completion returns the best solution.

$$\Delta \tau_{ij} = \frac{Q}{Z_{\text{best}}}$$ (7.18)

### 7.5 Experimental Settings

Three types of synthetic data sets, $\text{Type 1}$, $\text{Type 2}$ and $\text{Type 3}$, are used to analyze the performance of the above four algorithms. In $\text{Type 1}$ data set, a task set of 99 tasks is randomly generated with a task length in the range of 1–9, and 9 VMs of different capability levels are taken for scheduling. $\text{Type 2}$ data sets and $\text{Type 3}$ data sets are generated according to the procedure for Expected Time to Compute (ETC) matrix described by Braun et al. [4]. $\text{Type 2}$ data sets have the property that the variability in task length exhibits random probability distribution and $\text{Type 3}$ data sets have task length variability of normal probability distribution. Each $i^{th}$ entry represents the estimated execution time of task $i$ on a ‘small’-type VM. The characteristics of the ETC matrix is varied to represent task heterogeneity ($\phi_b = 100, 1,000, 3,000$) and machine heterogeneity ($\phi_r = 10, 100, 1,000$). Sample $4 \times 4$ excerpt from one of the 512 ETC matrices is shown in Table 7.2. Experiments are done with 9 ETC matrices of size 512, another 9 of size 1024 and another 9 having 2048 entries. For an ETC matrix of 512 entries, 16 VMs are considered, and for 1024 and 2048 tasks, 32 VMs of different capability levels are considered. For comparing the performance of the biologically inspired algorithms, another meta-heuristic algorithm, namely, Greedy Randomized Adaptive Search Procedure (GRASP) is also simulated. Each of these five algorithms is trying to minimize the value of the $M$ function (Equation (7.1)) and $C$ function (Equation (7.2)) simultaneously, so as to have a minimum value of $Z$ (Equation (7.6)) satisfying constraint equations presented in Section 7.3.

$\text{Type 1}$ and $\text{Type 2}$ data sets have sets of tasks showing random probability distribution. These data sets are used to characterize cloud computing systems

<table>
<thead>
<tr>
<th>Sample 4 × 4 Type 2 Data Set</th>
<th>167177.182</th>
<th>19862.324</th>
<th>27226.802</th>
<th>116593.838</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>75809.844</td>
<td>80844.535</td>
<td>2957.051</td>
<td>69479.025</td>
</tr>
<tr>
<td></td>
<td>65871.960</td>
<td>970.218</td>
<td>78680.239</td>
<td>34823.485</td>
</tr>
<tr>
<td></td>
<td>103666.221</td>
<td>191988.413</td>
<td>51695.280</td>
<td>66.433</td>
</tr>
</tbody>
</table>
where the task arrival pattern is dynamic and unpredictable. For most of the
days or hours, the behavior of the system is purely random. Type 3 data
set tasks are generated to characterize bursty traffic in the cloud computing
systems, where there will be a huge demand for certain applications running
in the cloud, giving spikes. For this period, the task arrival rate and task
length show a bell-shaped curve. Also, it is reported in the literature that
Map-Reduce jobs in the cloud can be better characterized through log-normal
probability distribution. Hence, the data distribution in Type 3 data set is
chosen to be of a normal probability distribution. The probability distribution
of the sample data sets for Type 2 (random) and Type 3 (normal) data sets
is shown in Figures 7.1a and b, respectively.

Initial solutions are randomly generated for PSO, ABC, and GA tech-
niques. In the solution, it is ensured that every VM is assigned only one task
from the task row and more than one task is not assigned to the same VM in
an iteration. Experiments are done by varying $\theta$ from 0 to 1 at an interval of
0.1, on Type 1 data sets to study the behavior of each of the algorithms. When
$\theta = 0$, the optimization algorithm finds running time optimal schedule, and
when $\theta = 1$, the algorithm determines cost optimal schedule which is equiva-
lent to finding the solution for the single-objective optimization problem. Each
of the algorithms is tested on Type 1 data sets for a different number of tasks,
a different number of VMs and different types of VMs. For further analysis,
Type 2 data sets and Type 3 data sets are used. The termination criteria for
all algorithms are set as 500 generations, which is determined from the con-
vergence study. Each of these algorithms is run ten times, and the average
values of cost and makespan are presented. The parameters needed for each
of the algorithms are also set as given below:

**PSO Settings:** The inertia weight $w = 0.6$, $c_1 = c_2 = 0.2$, $v_{\min} = -4.0$,
$v_{\max} = 4.0$ and $N = 25$.

![FIGURE 7.1](image)
Probability distribution of data sets. (a) Type 1 & Type 2, (b) Type 3.
**ABC Settings:** The number of food sources $N = 25$, the number of employed bees $E = 13$, number of onlooker bees $U = 12$ and limit = 1.

**GA Settings:** The population size $N = 25$. The crossover probability is 90% and the mutation probability is 5%.

**ACO Settings:** Initial pheromone value $\tau = 0.5$, $N_a = 25$, $Q = 1/10$, $\alpha = 0.3$, $\beta = 1$ and $\rho = 0.4$.

### 7.6 Results and Discussion

Figure 7.2a represents the average convergence values on cost for different values of $\theta$, and Figure 7.2b is that of makespan on the Type 1 data set. In both these figures, it is seen that the values obtained by the GRASP and ACO algorithms are much higher when compared to GA, PSO, and ABC algorithms for different values of $\theta$. Hence, for further analysis on the Type 1 data set, only the performances of GA, PSO, and ABC algorithms are compared. The convergence characteristics of these algorithms for $\theta = 0.9$ are further studied. For this $\theta$ value, the PSO algorithm finds an optimal cost value in 90% of the time, but both GA and ABC algorithms have not converged to the optimal cost value on multiple trials.

A detailed performance analysis of these algorithms is done by varying the number of tasks from 18 to 99 (refer Figures 7.3a and b), with a different number of VMs from 6 to 9 varying the number of task rows from 17 to 11 with appropriate padding (refer Figures 7.4a and b) and various types of VMs (refer Figures 7.5a and b) on the Type 1 data set. The average results of 10 independent trials are presented in these figures. It can be seen that the

![FIGURE 7.2](image)
Pareto-optimal solutions - Type 1 data sets. (a) Cost, (b) Makespan.
performance of PSO algorithm is better than the GA, and both are better than the ABC algorithm for most of the cases.

To study the performance of these algorithms on a more number of tasks and VM pairs, all these algorithms are executed on nine, Type 2 data sets having a random probability distribution and on nine, Type 3 data sets having normal probability distribution on task length, characterizing regular traffic and bursty traffic, respectively. The average cost and makespan of each of these algorithms are compared against GRASP algorithm, and the average performance improvement in % of 10 independent trials is presented in Table 7.3. The % performance benefit of Algorithm X on factor a (denoted as $X_a$) with respect to Algorithm Y on the same factor (denoted as $Y_a$) is calculated using the following equation:
FIGURE 7.5
Type 1 data set - different types of VMs. (a) Cost Vs VM Types, (b) Makespan Vs VM Types.

TABLE 7.3
Result Comparison with GRASP

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Type 2</th>
<th>Type 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cost %</td>
<td>Makespan %</td>
</tr>
<tr>
<td>ACO</td>
<td>0.102</td>
<td>-4.212</td>
</tr>
<tr>
<td>ABC</td>
<td>1.491</td>
<td>16.687</td>
</tr>
<tr>
<td>GA</td>
<td>5.751</td>
<td>31.281</td>
</tr>
<tr>
<td>PSO</td>
<td>9.779</td>
<td>57.356</td>
</tr>
</tbody>
</table>

\[ a\% = \frac{(Y_a - X_a)}{Y_a} \times 100 \]  
(7.19)

In Table 7.3, it is seen that the performance of ACO algorithm is poor when compared to GRASP algorithm. All the other algorithms are better than GRASP in both the objective function values. Also, both GA and PSO algorithm produce better results than the other three algorithms.

The three better-performing algorithms (GA, ABC, and PSO) are again analyzed for Type 2 and Type 3 data sets having a more number of tasks with double the number of VMs. The performance of PSO algorithm and GA is compared against the ABC algorithm, and the average % improvement in cost and makespan is shown in Table 7.4 for \( \theta = 0.9 \). We have considered nine, 1024 task set and nine, 2048 task set with 32 VMs each, and their average results are presented. As it is evident from the table, the % benefit of the PSO algorithm is far better than GA for any data set types. Also, there are no significant performance benefits among Type 2 or Type 3 data sets for PSO algorithm or GA implying that these algorithms perform similarly on any type of traffic in the cloud systems.
TABLE 7.4
Result Comparison with ABC Algorithm

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Data Set</th>
<th>Type 2</th>
<th>Type 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Cost %</td>
<td>Makespan %</td>
</tr>
<tr>
<td>GA</td>
<td>1024</td>
<td>2.145</td>
<td>9.055</td>
</tr>
<tr>
<td></td>
<td>2048</td>
<td>2.129</td>
<td>8.623</td>
</tr>
<tr>
<td>PSO</td>
<td>1024</td>
<td>6.68</td>
<td>36.620</td>
</tr>
<tr>
<td></td>
<td>2048</td>
<td>6.718</td>
<td>35.963</td>
</tr>
</tbody>
</table>

The performance of each of these algorithms is analyzed using two quality parameter measures, namely the Degree of Imbalance (DOI) [6,12] and Standard Deviation (σ) [6,19] which are defined in Equations (7.20) and (7.21), respectively. DOI is a measure to know the load balancing ability among heterogeneous VMs. A small value of DOI is preferred. Standard Deviation on task completion time is an indication of the search efficiency of the algorithm in finding optimal makespan. A small value of σ is an indication of the algorithm’s effectiveness in finding stable solutions.

\[
DOI = \frac{(T_{\text{max}} - T_{\text{min}})}{T_{\text{avg}}} \tag{7.20}
\]

\[
σ = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})^2} \tag{7.21}
\]

where \( T_{\text{max}} \), \( T_{\text{min}} \) and \( T_{\text{avg}} \) are the maximum, minimum, and average task lengths \( T_i \), respectively, submitted among all VMs in an epoch; \( x_i \) is the task completion time of \( VM_i \); and \( \bar{x} \) is the mean task completion time of all VMs.

Tables 7.5 and 7.6 show a comparison of DOI and σ for different algorithms. It can be seen that the load of VMs is more balanced in PSO technique as they have the small DOI value and comparatively better in GA than in other algorithms. Similarly, the value of σ is comparatively small for PSO algorithm, showing its capacity to search better in the objective space.

From the analysis, it can be concluded that the GA and PSO algorithm are comparatively better for task scheduling in the cloud computing environment, irrespective of the data characteristics. ACO and GRASP algorithms are unable to achieve an optimal value in any of the cases studied. The ABC algorithm performs better than both GRASP and ACO algorithms, but its

TABLE 7.5
DOI & σ for Type 1 Data Sets

<table>
<thead>
<tr>
<th></th>
<th>GRASP</th>
<th>ACO</th>
<th>ABC</th>
<th>GA</th>
<th>PSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOI</td>
<td>2.054</td>
<td>2.181</td>
<td>2.241</td>
<td>2.06</td>
<td>1.809</td>
</tr>
<tr>
<td>σ</td>
<td>1.792</td>
<td>2.082</td>
<td>1.912</td>
<td>1.722</td>
<td>1.32</td>
</tr>
</tbody>
</table>
performance is poor when compared to GA and PSO techniques. The PSO technique outperforms all the other algorithms in any of the aspects considered. The running time needed for the convergence of PSO techniques is also less when compared to other bio-inspired algorithms.

7.7 Conclusion

Scheduling tasks in the cloud is a challenging process as it involves many factors such as cost, energy-efficient management and profit considerations at CSP’s end, and execution time and other QoS parameters at the user’s end. In the cloud environment, the task arrival pattern is also highly unpredictable and dynamic in nature. For scheduling tasks in this complex environment, the problem is modeled as a constraint bi-objective optimization problem, where the objectives considered are makespan and cost and have used five meta-heuristic search algorithms. A weighted sum–based multi-objective optimization approach is employed to find the optimal solutions. The experimental result shows that the performance of PSO technique is far better than other bio-inspired optimization algorithms irrespective of the task arrival patterns and length of tasks. This study provides a comparison and gives an insight into the suitability of each of these techniques in the cloud domain. As a future work, the performance of these algorithms in the scheduling of workflow applications needs to be explored. Similarly other methods to find pareto-optimal solutions are to be experimented.

Bibliography


Index

A
ABC, see Artificial bee colony algorithm (ABC)
ACO, see Ant colony optimization (ACO)
Adaptive learning algorithms, 43
Advanced encryption standard (AES), 109
Agent-based resource sharing
  in inter cloud, 42–43
  in multi-cloud, 46–47
AHP, see Analytic Hierarchy Process (AHP)
AI, see Artificial intelligence (AI)
Amazon EC2 framework, 110, 117, 122
Analytic Hierarchy Process (AHP), 75–76
Ant colony algorithm, 130, 131
Ant colony optimization (ACO), 23, 110
  balanced allocation, of server resources, 120–122
  description of, 111
  performance analysis, 177–179
  resource scheduling decision on, 115–116
  pheromone function, 115
  user data auditing, 116–117
  successful resource allocation, 119, 120
  with different volumes of active users, 120
  for task scheduling, 174, 175–177
Application-awareness
  in data centre resource management, 155, 158–159
  inference schemes topology, 159
  service models, 156–157
Application-layer information, 160
ART, see Average response time (ART)
Artificial bee colony algorithm (ABC), 8–10
  performance analysis, 177–179
  for task scheduling, 171–174
Artificial intelligence (AI), 155
ASPEN, see Automated Service Provisioning Environment (ASPEN)
Auditing approach, 109, 110, 112–113; see also Data auditing approach; Public auditing
Automated bartering system, 42
Automated Service Provisioning Environment (ASPEN), 48
Average response time (ART), 79, 80, 82

B
BA, see Bat algorithm (BA);
  Brokering Agent (BA)
Bartering models, 48, 49
Base 64 algorithm, 109
Bat algorithm (BA), 15–18
BATS-HS, 15
BDI, see Belief-Desire-Intension (BDI)
Bees’ life algorithm (BLA), 131
Belief-Desire-Intension (BDI), 43
Binary version, bat algorithm, 16, 17
Bi-objective optimization for task scheduling, 166, 167
ant colony optimization, 174, 175–177
artificial bee colony algorithm, 171–174
genetic algorithm, 174, 175
mathematical formulation, 168–169
multi-objective optimization, 169
overview of, 165–167
particle swarm optimization, 167, 170–171
performance analysis, 177–179
results, 179–183
BLA, see Bees’ life algorithm (BLA)
Box-plot, of algorithms, 93, 96–98
Brokering Agent (BA), 46–47
Bubble-net feeding, 12

C
CA, see Customer Agent (CA)
Cat swarm optimization (CSO), 18–20
CBIM, see Cluster-Based Incentive Mechanism (CBIM)
CC, see Cloud computing (CC)
CCF, see Cross-cloud federation (CCF)
CDNs, see Content delivery networks (CDNs)
CF, see Cloud Federation (CF)
CFM, see Cloud file manager (CFM)
Challenge–response-based method, 109
Chaos theory, 16
Chromosome encoding, 83
CloudCmp tool, 75
Cloud computing (CC), 2, 128
overview of, 155–158
service models, 156–157
task scheduling in, 2, 3
formulation, 4–5
swarm intelligence methods, 5, 8–27 (see also Swarm intelligence methods)
Cloud computing architecture, 108, 109
Cloud data auditing, 109, 113; see also Data auditing approach
Cloud data centers, 61–62, 128
application-awareness in, 155, 158–159
Cloud 3D printing, 56–57
Cloud Federation (CF), 41
Cloud file manager (CFM), 116, 117
Cloud layer, VCC, 58
Cloud manufacturing (CM), resource sharing in, 54–57, 63
Cloud ontology, 46
Cloud Primary Application Services (CPAS), 58, 59
Cloud Resource Bartering System (CRBS), 42
Cloud service admission, 159
Cloud Service Measurement Index Consortium (CSMIC), 74, 79
Cloud service providers (CSP), 116, 166
Cloud service providers, in multi-cloud
numerical experiments, 89–92, 99–102
overview of, 74–75
problem description context, 76
decision variable, 77
model formulation, 76–77
objective functions, 77–80
solution algorithms, 80–81
cost, 81–82
genetic algorithm, 82–84
harmony search, 87–89
particle swarm optimization, 85–87
performance of, 82, 93–99
CloudSim platform, 8–10, 20, 24, 25
Cluster-Based Incentive Mechanism (CBIM), 44
Communication layer, VCC, 57, 58
Content delivery networks (CDNs), 159
Contract-based resource sharing algorithm, 109–110
Cost
cloud service providers, 77–78, 81–82
function value, 135
modeling, 75
CPAS, see Cloud Primary Application Services (CPAS)
CRBS, see Cloud Resource Bartering System (CRBS)
Cross-cloud federation (CCF), 44
Cryptographic keyGen algorithm, 109
CS algorithm, see Cuckoo search algorithm (CS)
CSMIC, see Cloud Service Measurement Index Consortium (CSMIC)
CSO, see Cat swarm optimization (CSO)
Cuckoo search algorithm (CSA), 20–23, 26, 131
Customer Agent (CA), 46

D
Darwin’s theory of evolution, 84
Data and service auditing, 116–117
Data auditing approach, 109, 110, 112–113
data verification, 119–120
metadata generation time, on different file sizes, 117–119
Data center, 119, 120, 155, 158
application-awareness in, 155, 158–159
Data verification, 119–120
DCS, see Discrete Cuckoo Search (DCS)
Decision variable, cloud service providers, 77
Degree of Imbalance (DOI), 182, 183
Demand price function, 135
Dependent/workflow scheduling, 3
Discrete Cuckoo Search (DCS), 22
DOI, see Degree of Imbalance (DOI)
DVC, see Dynamic voltage scaling (DVC)
Dynamic resource allocation, 110, 112, 113, 122
Dynamic resource scheduling, 112–115
Dynamic voltage scaling (DVC), 129

E
E-barter system, 42
Echolocation, 15
Energy-aware dynamic TS (EDTS), 129
ETC, see Expected Time to Compute (ETC)
Evolutionary optimization, 111, 167; see also Ant colony optimization (ACO); Genetic algorithm (GA)

Expected Time to Compute (ETC), 177

F

Fairness Driven Networks of Favor (FD-NoF), 44
FCAC, see Formal Concept Analysis Clustering (FCAC)
FD-NoF, see Fairness Driven Networks of Favor (FD-NoF)

Federated cloud, peer-to-peer resource sharing, 44–45
Fitness function value, 134, 135
Focused Selection Contract Net Protocol (FSCNP), 46

Formal Concept Analysis Clustering (FCAC), 47
FSCNP, see Focused Selection Contract Net Protocol (FSCNP)

G

GA, see Genetic algorithm (GA)
Gale-Shapley algorithm, 54
GCC, see Green cloud computing (GCC)
Genetic algorithm (GA), 47, 62
chromosome encoding, 83
description of, 82–83
new population generation, 83, 84
performance analysis, 177–179
performance of, 93–99
for task scheduling, 174, 175

Geographically Redundant Storage (GRS), 78

GRASP, see Greedy Randomized Adaptive Search Procedure (GRASP)
Gray wolf optimization (GWO), 13–15, 128, 129, 147
mathematical models of, 137–139
pseudo-code of, 139
Greedy Randomized Adaptive Search Procedure (GRASP), 177, 179–182

Green cloud computing (GCC), 61–63

GRS, see Geographically Redundant Storage (GRS)

GWO, see Gray wolf optimization (GWO)

H

Harmony consideration rate (HMCR), 87
Harmony search (HS)
description of, 87
harmony representation, 87, 88
new harmony memory creation, 88–89
performance of, 93–99

HFS, see Hybrid flow shop (HFS)
HMCR, see Harmony consideration rate (HMCR)

Homomorphic network, 109

HS, see Harmony search (HS)

Hybrid flow shop (HFS), 17, 21
Hybridization method, 26

Hybrid swarm algorithm, 23–24

I

IaaS, see Infrastructure as a service (IaaS)

IC, see Inter Cloud (IC)
IETF, see Internet Engineering Task Force (IETF)

Improved particle swarm optimization (IPSO) algorithm, 131

Incentive mechanisms
in mobile cloud computing, 53–54
resource sharing, 44, 63

Indirect reciprocity, 45

Inference schemes, 159

Infrastructure as a service (IaaS), 74–75, 81, 92, 99–102, 157
attributes for average response time, 79, 80, 82
suitability measure, 79–80
by scanned providers, 89, 92

Inside-vehicle layer, VCC, 57

Innovation source, TS-GWO, 137

Integer-PSO technique, 167

Intelligent resource administration, 159

Intelligent Transport Systems (ITS), 59

Inter Cloud (IC), 41, 42
federated cloud, resource sharing in agent-based, 42–43
peer-to-peer, 44–45

Internet Engineering Task Force (IETF), 157

Internet exchange providers (IXPs), 159

Internet of Things (IoT), 47, 56

Inter-tenant Resource Trading (IRT), 45

IoT, see Internet of Things (IoT)

IPSO algorithm, see Improved particle swarm optimization (IPSO) algorithm

IRT, see Inter-tenant Resource Trading (IRT)

ITS, see Intelligent Transport Systems (ITS)

IXPs, see Internet exchange providers (IXPs)

J

Job scheduling, 3, 4, 6, 23–26;
see also Task scheduling, in CC

JSSP, see Scheduling problems of the Job Shop (JSSP)

K

KDE, see Kernel Density Estimation (KDE)

Kernel Density Estimation (KDE), 47

L

LBMPSO, see Load balancing mutation PSO (LBMPSO)

Lion optimization algorithm (LOA), 10–12

LOA, see Lion optimization algorithm (LOA)

Load balance, 5

Load balancing mechanism, 110

Load balancing mutation PSO (LBMPSO), 129

Locust-inspired scheduling, 62

M

MA, see Memetic algorithm (MA)

Makespan, 18, 20, 23, 25, 26, 166, 167, 169, 179–183

Manufacturing Cloud (MC), 41

Mathematical model, GWO, 137–139

MC, see Manufacturing Cloud (MC); Multi-Cloud (MC)

MCC, see Mobile cloud computing (MCC)

MEC, see Multi-access edge computing (MEC)
Index

Megachiropter/megabat, 15
Memetic algorithm (MA), 23
Merkle tree concept, 110, 112, 113, 117, 122
Message digest, 115–116
Metaheuristics optimization techniques, 3
Meta-heuristic techniques, resource sharing, 62
Microchiropter/microbat, 15
MIVS problem, see Multi-item vendor selection (MIVS) problem
MOA, see Multi-objective approach (MOA)
MOBFOA, see Multi-objective bacteria foraging optimization algorithm (MOBFOA)
Mobile cloud computing (MCC), 41, 50, 53
incentive mechanisms in, 53–54
Model formulation, cloud service providers, 76–77
MO-Jswarm, see Multi-objective Jswarm (MO-Jswarm)
MOPSO, see Multi-objective particle swarm optimization (MOPSO)
MOWOA, see Multi-objective swarm optimization (MOWOA)
MSMOOA, see Multi-swarm optimization algorithm (MSMOOA)
Multi-access edge computing (MEC), 131
Multi-agent system for cloud resource enhancement, 42
Multi-cloud (MC), 41
agent-based service selection approaches, 46–47
QoS-based service selection, 47, 48
Multi-cloud environment, 74, 80, 81
cloud service providers in, 74, 80–81
algorithms performance, 82, 93–99
genetic algorithm, 82–84
harmony search, 87–89
numerical experiments, 89–92, 99–102
overview of, 74–75
particle swarm optimization, 85–87
problem description, 76–80
Multi-item vendor selection (MIVS) problem, 80–81
Multimedia services, resource sharing in, 59
Multi-objective approach (MOA), 25
Multi-objective bacteria foraging optimization algorithm (MOBFOA), 24, 25
Multi-objective design, TS-GWO, 134–135
Multi-objective Jswarm (MO-Jswarm), 131
Multi-objective optimization, 167, 169; see also Bi-objective optimization
Multi-objective particle swarm optimization (MOPSO), 131
Multi-objective swarm optimization (MOWOA), 24–26
Multiprocessor scheduling problems, 18
Multi-step heuristic scheduling algorithm, 61–62
Multi-swarm optimization algorithm (MSMOOA), 25–26
Index

Multi-tenant based resource sharing, 45

N
National Institute of Standards and Technology (NIST), 156
Nawaz-Enscore-Ham (NEH) algorithm, 21, 22
New Actual Ratio (NRR), 43
New population, GA, 83, 84
NIST, see National Institute of Standards and Technology (NIST)
Non-dominated sorting genetic algorithm (NSGA-II), 24, 25, 56, 167
NRR, see New Actual Ratio (NRR)
NSGA-II, see Non-dominated sorting genetic algorithm (NSGA-II)
Numerical experiments, cloud service providers, 89–92, 99–102

O
OBL, see Opposition-based learning (OBL)
On-Board Units (OBUs), 57
“On-demand software,” 157
Ontology-based service discovery, 46
Opposition-based learning (OBL), 11
Optimal methods, cloud manufacturing, 56
Orthogonal Taguchi (OT), 19–20
OTB-CSO, 19–20

P
PaaS, see Platform as a Service (PaaS)
PAMOCS, see Pareto Archived Multi-Objective Cuckoo Search (PAMOCS)
PAR, see Pitch adjusting rate (PAR)
Pareto Archived Multi-Objective Cuckoo Search (PAMOCS), 22–23
Pareto-GWO, 14
Pareto-optimal solutions, 167, 179
Particle swarm optimization (PSO), 6–7, 23, 24, 129, 131
description of, 85
new population creation, 86–87
performance analysis, 177–179
performance of, 93–99
solution representation, 85, 86
for task scheduling, 167, 170–171
“Pay-as-you-go” model, 77, 154, 156
Peer-to-peer resource sharing, 44–45
Performance, cloud service providers, 78–80, 82
PFSSP, see Problem of the permutation ow shop (PFSSP)
Pheromone function, ACO, 115
Pheromones, 174–176
Pitch adjusting rate (PAR), 87
Platform as a Service (PaaS), 157
Power utilization, 122–123
Pricing models, 48
Problem of the permutation ow shop (PFSSP), 22
Proposed solution
ant colony optimization pheromone function, 115
resource scheduling decision, 115–116
user data auditing, 116–117
dynamic resource allocation, 112–113
PSO, see Particle swarm optimization (PSO)
Public auditing, 109; see also Data auditing approach

Q
Quality of service (QoS), 2, 47, 48

R
RAs, see Resource Agents (RAs)
Reciprocal Resource Fairness (RRF), 45
Reliability, 17, 109, 110, 116, 129
Reliable data auditing, 109, 110, 112–113
data verification, 119–120
metadata generation time, on different file sizes,
117–119
Requested Ratio (RR), 43
Resource Agents (RAs), 46
Resource allocation, 109, 110
Resource cost system, 129
Resource-hungry applications, 160
Resource management, 56, 63
data centre, 153–160
Resource optimization, 159
Resource scheduling
ant colony optimization decision on, 115–116
pheromone function, 115
user data auditing, 116–117
Resource Service Chain Composition Algorithm (RSCCA), 56
Resource sharing
in cloud manufacturing, 54–57
in federated cloud
agent-based, 42–43
peer-to-peer, 44–45
in green cloud computing, 61–63
in inter cloud, 41, 42
meta-heuristic techniques, 62
in multi-cloud
agent-based service selection approaches, 46–47
QoS-based service selection, 47, 48
overview of, 40–41
social cloud, 47, 48–50
in vehicular cloud computing, 58–61, 63 (see also Vehicular cloud computing (VCC))
Rivest–Shamir–Adleman (RSA), 109
Roadside units (RSUs), 57
RR, see Requested Ratio (RR)
RRF, see Reciprocal Resource Fairness (RRF)
RSA, see Rivest–Shamir–Adleman (RSA)
RSA-based data integration and auditing scheme, 109
RSCCA, see Resource Service Chain Composition Algorithm (RSCCA)
RSUs, see Roadside units (RSUs)

S
SaaS, see Software as a Service (SaaS)
SC, see Social cloud (SC)
Scheduling, 166; see also Task scheduling
Scheduling problems of the Job Shop (JSSP), 22
SCPS, see Smart Car Parking System (SCPS)
SCT, see Service Composition Table (SCT)
SDA, see Service Discovery Agent (SDA)
SDDC, see Software-defined data center (SDDC)
SDN, see Software-defined networking (SDN)
Index

SDX, see Software-defined exchange (SDX)

Secure hash algorithm 2 (SHA-2), 109

Semantic Web Service (SWS), 42

Semi-recursive contract net protocol (SR-CNP), 46

Service Composition Table (SCT), 46

Service Discovery Agent (SDA), 46

Service Measurement Index (SMI), 74, 75

Service Provider Agents (SPAs), 46

SFLA, see Shuffled frog leaping algorithm (SFLA)

SHA-2, see Secure hash algorithm 2 (SHA-2)

Shuffled frog leaping algorithm (SFLA), 24

Single-cloud provider, 76

Small and Medium Enterprises (SMEs), 41

Smart Car Parking System (SCPS), 60

SMEs, see Small and Medium Enterprises (SMEs)

SMI, see Service Measurement Index (SMI)

Social cloud (SC), 47–52

Software as a Service (SaaS), 157

Software-defined data center (SDDC), 158

Software-defined exchange (SDX), 159

Software-defined networking (SDN), 155, 157, 159, 160

Solution algorithms
cloud service providers, 80–81
cost, 81–82
genetic algorithm, 82–84
harmony search, 87–89

particle swarm optimization, 85–87
performance of, 82, 93–99

Solution encoding, 136

TS-GWO, 136

SPAs, see Service Provider Agents (SPAs)

SR-CNP, see Semi-recursive contract net protocol (SR-CNP)

Standard Deviation, 182, 183

Suitability measure, 79–80, 82

Swarm intelligence methods, 5, 26–27

artificial bee colony algorithm, 8–10
bat algorithm, 15–18
cat swarm optimization, 18–20
cuckoo search algorithm, 20–23
gray wolf optimization, 13–15
hybrid swarm algorithm, 23–24
lion optimization algorithm, 10–12
multi-objective swarm optimization, 24–26
particle swarm optimization, 6–7, 23, 24

Swarm optimization approach, 110

SWS, see Semantic Web Service (SWS)

T

Task resolver, 113

Task scheduling (TS), 128, 132, 166

bi-objective optimization for, 166, 167
ant colony optimization, 174, 175–177
artificial bee colony algorithm, 171–174
Task scheduling (TS) (cont.)
genetic algorithm, 174, 175
mathematical formulation, 168–169
multi-objective optimization, 169
overview of, 165–167
particle swarm optimization, 167, 170–171
performance analysis, 177–179
results, 179–183
Task scheduling, in CC formulation, 4–5
swarm intelligence methods, 5, 26–27
artificial bee colony algorithm, 8–10
bat algorithm, 15–18
cat swarm optimization, 18–20
cuckoo search algorithm, 20–23
gray wolf optimization, 13–15
hybrid swarm algorithm, 23–24
lion optimization algorithm, 10–12
multi-objective swarm optimization, 24–26
particle swarm optimization, 6–7, 23, 24
Task-to-resource (TOR), 18
Third-party auditor (TPA), 109, 113
TOR, see Task-to-resource (TOR)
TPA, see Third-party auditor (TPA)
Traffic flows, 160
Traffic management, resource sharing in, 59
Transient cloud, 53
TS, see Task scheduling (TS)

TS-GWO
challenges of, 133
cost, 140, 143–146
experimental results, 140–146
minimum values (Mm), 140–142
multi-objective design model, 134–135
overview of, 128–129
system model, 133–134
inspiration source, 137
mathematical model, 137–139
multi-objective design, 134–135
solution encoding, 136

U
User copy of message digest, 115–116
User data auditing, 116–117
User files classification, 110

V
VACC, see Vehicle-Assisted Cloud Computing (VACC)
VANET, see Vehicle ad hoc networks (VANET)
VCC, see Vehicular cloud computing (VCC)
VCG auction, see Vickrey-Clarke-Groves (VCG) auction
Vehicle ad hoc networks (VANET), 57
Vehicle-Assisted Cloud Computing (VACC), 60
Vehicular cloud computing (VCC), 41, 57
architecture
cloud layer, 58
communication layer, 57, 58
inside-vehicle layer, 57
resource sharing in, 63, 66
multimedia services, 59
smart parking, 60
traffic management, 59
Vehicle-Assisted Cloud Computing, 60
Vehicular Fog Computing, 60
Vehicular Fog Computing (VFC), 60
Vehicular-to-vehicular (V2V), 57
VFC, see Vehicular Fog Computing (VFC)
Vickrey-Clarke-Groves (VCG) auction, 53
Virtualization, 154
Virtual machines (VMs), 48, 61, 62, 133, 154
artificial bee colony algorithm, 8
average response time for, 92
configurations, 89, 90
cuckoo search algorithm, 20, 21
operating system types, 90
price ranges per hour for, 91
services for providers, 100–101
storage prices per GB, 91
task scheduling problem, 3–5
V2V, see Vehicular-to-vehicular (V2V)

W
Weighted sum method, 82
Whale optimization algorithm (WOA), 12–13, 130
W-scheduler, 12, 130
Taylor & Francis eBooks

www.taylorfrancis.com

A single destination for eBooks from Taylor & Francis with increased functionality and an improved user experience to meet the needs of our customers.

90,000+ eBooks of award-winning academic content in Humanities, Social Science, Science, Technology, Engineering, and Medical written by a global network of editors and authors.

TAYLOR & FRANCIS EBOOKS OFFERS:

A streamlined experience for our library customers

A single point of discovery for all of our eBook content

Improved search and discovery of content at both book and chapter level

REQUEST A FREE TRIAL

support@taylorfrancis.com