Optimal Data Allocation Method Considering Privacy Enhancement using E-CARGO

by

Chengyu Peng

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science (MSc) in Computational Sciences

The Office of Graduate Studies
Laurentian University
Sudbury, Ontario, Canada

THESIS DEFENCE COMMITTEE/COMITÉ DE SOUTENANCE DE THÈSE
Laurentian Université/Université Laurentienne
Office of Graduate Studies/Bureau des études supérieures

Title of Thesis Titre de la thèse	Optimal Data Allocation Method Considering Privacy Enhancement using E-CARGO
Name of Candidate Nom du candidat	Peng, Chengyu
Degree Diplôme	 Master of Science
Department/Program Département/Programme	Computational Science
Date of Defence Date de la soutenance	April 19, 2023

APPROVED/APPROUVÉ

Thesis Examiners/Examinateurs de thèse:

Dr. Ratvinder Grewal
(Co-Supervisor/co-directeur(trice) de thèse)

Dr. Haibin Zhu
(Co-Supervisor/Co-directeur(trice) de thèse)

Dr. Dongning Zhu
(Committee member/Membre du comité)

Dr. Wengeng Li
(External Examiner/Examinateur externe)

Approved for the Office of Graduate Studies
Approuvé pour le Bureau des études supérieures
Tammy Eger, PhD
Vice-President Research (Office of Graduate Studies)
Vice-rectrice à la recherche (Bureau des études supérieures)
Laurentian University / Université Laurentienne

ACCESSIBILITY CLAUSE AND PERMISSION TO USE

Abstract

With the rise in popularity of cloud computing, there is a growing trend toward the storage of data in a cloud environment. However, there is a significant increase in the risk of privacy information leakage, and users could face serious challenges as a result of data leakage. In this paper, we propose an allocation scheme for the storage of data in a collaborative edge-cloud environment, with a focus on enhanced data privacy. In addition, we explore an extended application of the approach to sourcing. Specifically, we first evaluate the datasets and servers. We then introduce several constraints and use the Environments-Classes, Agents, Roles, Groups, and Objects (E-CARGO) model to formalize the problem. Based on the qualification value, we can find the optimal allocation using the IBM ILOG CPLEX Optimization (CPLEX) Package. At a given scale, the allocation scheme scores based on our method improve by about 50% compared to the baseline method and the trust-based method. Moreover, we use a similar approach to analyze procurement issues in the supply chain to help companies reduce the carbon emissions. This shows that our proposed solution can store data in servers that better suit their requirements and is adaptable to other problems.

Keywords

Allocation Strategy, Collaboration, E-CARGO, Group Role Assignment (GRA), Privacy, Green Computing

Acknowledgments

I would first like to thank Dr. Haibin Zhu, Nipissing University, Canada, whose expertise was invaluable in formulating the research questions and methodology. Your insightful feedback pushed me to sharpen my thinking and brought my work to a higher level.

I would also like to thank Dr. Ratvinder Grewal, Laurentian University, Canada, for his valuable guidance throughout my studies. You provided me with the tools that I needed to choose the right direction and successfully complete my dissertation.

Whenever I felt confused, my two supervisors always guided me and showed me the way. Their rigorous academic attitude and profound knowledge deserve me to learn from all my life.

I would particularly like to acknowledge Dr. Linyuan Liu, Nanjing Audit University, China, for all his help with my studies, without his guidance I would not have been able to complete my research smoothly.

In addition, I would like to thank Dr. Kalpdrum Passi and Dr. Waldemar Koczkodaj, Laurentian University, Canada, for their kind help and their support and understanding of my life.

Finally, I could not have completed this dissertation without the support of my parents and friends, who provided wise counsel as well as happy distractions to rest my mind outside of my research.

Table of Contents

List of Tables

List of Figures

List of Appendices

Chapter 1

1 Introduction

1.1 Background

The application of various cloud computing technology is expanding as the technology develops. The deployment of distributed storage and computing frameworks in the cloud is supported by a large number of cloud service providers (CSPs), including Amazon, Microsoft, Google, and other well-known CPS. However, there is a significant risk of privacy information leaking in this situation of outsourcing data storage [45]. The cost of a data breach report [29] claims that the average total cost of data breach has increased by about 10% yearly, the largest single year cost increase in the last seven years and the hybrid cloud model had the lowest average total cost of a data breach. If data is stored in the public cloud, an untrustworthy cloud service provider or server attacker can directly snoop on some or even all of the data and further speculate on individual privacy information, resulting in the leakage of users' privacy information [18]. Consequently, a crucial concern now is how to guarantee the privacy of data kept on the cloud servers [31], [32], [49]. A considerable amount of work is focused on encrypting data to protect data privacy, but these methods have the disadvantages of cumbersome data recovery and high data transfer costs [10], [61].

Edge-cloud architecture [3] that disperses data partially in both edge servers and public cloud servers is an effective strategy to enhance data privacy. It is well known that by relying on public cloud servers, users inevitably give up a degree of control, while edge server users have better control over their data, making edge servers more suitable for handling or storing sensitive data that can easily infer private information, while relatively less important data can be stored in cloud servers. Numerous studies have concentrated on this. Edge servers [13] can improve data storage in terms of privacy, and edge servers can also be fully utilized. However, since the quantity and capacity of edge servers are constrained in practice, we need practical ways to divide the data so that one part of the data is stored in the edge servers and the other part is stored in the public cloud servers.

We may solve this problem by evaluating the data sensitivity [8], [11], [58] and server trust [4], [7], [40], [42], [53], so that we are able to classify the sensitive data and store them in edge servers and others in public cloud servers, respectively. In addition, each server has a unique trust level that can be used as a reference in order to store certain data in a more appropriate server. Although privacy should be taken into account when selecting servers for data storage, there are other considerations that should not be ignored as well, such as user requirements for servers [50], [60] and storage costs [30], [33], [57], etc. Notably, selecting appropriate servers to store the data while keeping a number of restrictions is a very complex problem.

Fortunately, the Environments-Classes, Agents, Roles, Groups, and Objects (E-CARGO) model can assist us in successfully overcoming this difficulty. It is common to ask why we use E-CARGO. E-CARGO has been verified as a significant tool for investigating challenging problems in collaborative and complex systems. It has a set of clarified components for a system or a problem and provides quite a few validated symbols to express the elements of a problem. E-CARGO provides consistency in concepts, conciseness in structures, and expressiveness in notations. The proposed problem is in fact a natural result of the E-CARGO research [22], [23], [24], [25], [26]. In addition, the E-CARGO model has a wide range of applications and it can give us insight into problems in many areas. We note that human activities have significantly increased emissions of carbon dioxide and other greenhouse gases, posing a severe threat to our living environment [48]. In response to this issue, the entire globe has endorsed reducing carbon emissions, and businesses are no exception. They are under pressure to reduce carbon emissions [14]. In a supply chain [1], procurement plays a crucial role [15]. By reducing carbon emissions during the procurement process, upstream and downstream carbon emissions can be successfully influenced [51]. As it happens, the method we propose is applicable to the optimization of procurement solutions. The success or failure of the selected suppliers impacts every part of the business. The decision is influenced by a number of variables [12], including the cost of the purchase, the quality of an item, the timing of delivery, etc. According to the Scope 3 emissions [41] of the international emissions accounting tool Greenhouse Gas (GHG) accounting system, all indirect emissions that happen along the reporting company's value chain, including upstream and downstream emissions, must be taken into account when calculating carbon emissions. Companies must think about how to cut the carbon emissions produced in their supply chains as much as feasible in light of the tendency to minimize carbon emissions. In addition, the

productivity of each supplier varies in the actual procurement situation, which makes it more challenging to obtain an optimal procurement solution [55]. In terms of how to reduce carbon emissions in the procurement process, a lot of efforts are focused on supplier selection, but it is easy to fail to find the most reasonable procurement solution from a global perspective. Therefore, we can apply our proposed allocation method to the development of a procurement plan. Similar to the previous method, the E-CARGO model [25] can assist us in overcoming this difficulty when dealing with many intricate influences. We can first determine the ratings of the various items from the various suppliers we intend to purchase. This score can be made up of different evaluation indicators, and depending on the requirements of the purchaser, the weights of the different indicators can be calculated using the analytic hierarchy process (AHP) [16], [17]. Then, based on the suppliers' production capacities, the hours they can provide, and the procurement volume, we rate the hours of the providers as qualification values, then use the sum of optimally assigned hours' qualifications as the procurement solution. There are also several practical constraints to take into account [44], such as purchasing budgets, carbon emission caps, and the least requirement on product quality. The final solution for the allocation of working hours that meets the requirements is the procurement solution.

In summary, this paper focuses on developing a storage method that takes into account various requirements and can effectively enhance data privacy and extending the application of the method to the development of procurement schemes. Specifically, the main contributions of this paper are as follows:

(1) We propose a method to divide data into fields and distribute storage based on the fitness of data field sensitivity and server trust in order to protect the privacy of decentralized data storage.

(2) We use the E-CARGO model [22], [23], [24], [25], [26] to formalize and analyze the server allocation problem for data storage and introduce constraints such as server performance, user budgets and service provider into the model.

(3) We make a large number of simulations, and based on the outcomes, we analyze the operational efficiency, feasibility, and advantages of the method.

(4) We further improve the proposed method to enhance its performance and compare the improved method with other methods. We also propose solutions for scenarios dealing with large scale data and backup.

(5) We apply the above method to other areas. Starting from three factors: procurement cost, item quality and carbon emission involved in procurement, and using AHP to determine their weights to help users develop procurement strategies. We use the E-CARGO model to formalize and analyze the procurement problem for low carbon emissions and introduce constraints such as procurement budget, carbon emission caps and least quality requirements of items into the model.

1.2 Thesis Outline

The thesis is organized as follows:

Chapter 2 presents the work related to the research topic.

Chapter 3 presents a real-world scenario related to the proposed problem and the problem description of data storage.

The formalization of the problem using the E-CARGO model is presented in Chapter 4.

Chapter 5 conducts simulation experiments and analyzes the results based on our proposed method.

Chapter 6 discusses the improvement and illustrates the methods for handling large scale data and backup.

Chapter 7 presents an extended application of the mentioned method to the development of a procurement plan and the formalization of the problem using the E-CARGO model.

Chapter 8 conducts simulation experiments and gives guidance to users for developing a procurement plan.

The conclusion of this paper is in Chapter 9.

Chapter 2

2 Literature Review

2.1 Cloud Storage

Storing data in edge-cloud environment to ensure privacy is a very important and comprehensive issue. It is a challenge to allocate server storage data appropriately, and many efforts are focused on this.

Cloud storage: A lightweight hybrid distributed edge/cloud storage framework was presented by Makris et al. [3] to enhance end-user Quality of Experience (QoE) by relocating data close to them and thus minimizing network use and data transfer delays. Nitti et al. [40] proposed an integrity verification framework to ensure Ternary Hash Tree (THT) and Replica based Ternary Hash Tree (R-THT) based cloud storage. The proposed secure cloud auditing framework is highly secure and efficient in terms of storage, communication and computational cost. Zhuang et al. [28] suggested StoreSim multi-cloud storage system. They created the effective storage plan generating method SPClustering for dispersing user data to various clouds based on the information leakage detected by BFSMinHash. Zhang et al. [60] propose a service curve-based QoS algorithm to support latency guarantee applications, IOPS guarantee applications and best-effort applications at the same storage system, which not only provides a QoS guarantee for applications, but also pursues better system utilization. Al Nuaimi et al. [35] addressed both difficulties by improving the dual direction load balancing technique's effectiveness through increased replication and by reducing the storage requirements of a smart self-controlled technique (ssCloud) for deleting redundant data from each cloud server. Abhishek et al. [19] presented a unique hybrid cloud storage approach that enables users to use their computer clusters' quick primary storage as a caching tier in front of their slow secondary storage tier. Patil Rashmi et al. [47] used deduplication technology for data storage. This study's foundation is the homomorphic hash algorithm. The accuracy of the data that is saved on a cloud server is determined by a third party auditor who verifies the user's data for accuracy. There is less communication and computational overhead. When a malicious server launches a replacement attack, their framework is efficient and safe. Although all of these approaches

mentioned above enhance certain aspects, most do not sufficiently integrate edge servers with cloud servers and focus on the privacy of data.

2.2 Data Sensitivity and Server Trust

Data sensitivity: Lang et al. [9] provided an overview of data collecting parsimony, along with a novel approach for quantifying the idea using empirical Bayes estimates. Finally, the metric is evaluated using actual data. This method's advantages and disadvantages are explored from a theoretical and empirical perspective. They concluded that this metric is in many ways superior to others for model creation, but there are certain barriers to its adoption. Idar et al. [20] provided a dynamic architecture to automatically determine data sensitivity without the Data Owner's involvement. Sensitive data must be protected for as long as it is housed in the Hadoop cluster in order to prevent access from unauthorized users. Data sensitivity changes over time based on scenarios offered by their scalable framework. Park et al. [58] introduced their work toward fine-grained data centric security, which semi-automatically determines the sensitivity of enterprise data. In order to automatically find sensitive material in corporate data, they applied a range of text analytics and classification algorithms. This tool suite also provides estimations for the sensitivity of individual data. They created a proof-of-concept system that crawls every file on a computer and calculates the sensitivity of each file as well as the computer's overall sensitivity level. They ran a pilot test using laptops belonging to staff at a big IT organization. The method provides inspiration and ideas for obtaining data sensitivity in the article.

Server trust: Zhang et al. [34] present a trust-based secure multi-cloud collaboration framework for Cloud-Fog-Assisted IoT systems. They specifically created a role-based trust evaluation mechanism to raise the credibility of MCSC and ensured the security of users. They designed an effective user authentication scheme and a secure collaboration scheme to provide collaborative user authentication and access control mechanisms for MCSC in order to maintain the security of the services. Yang and Peng [59] present a novel scheduling technique based on trust and address the scheduling problem for workflow applications with trust restrictions. The suggested heuristic scheduling algorithm can more efficiently identify the most reliable execution flow than the conventional technique for scheduling application workflows, according to experimental results. Dang et al. [53] presented a trusted-based resource scheduling scheme. For big data applications processing sensitive data, they suggested a trust aware framework that

enables CSPs to use highly trusted resources. And they developed a trust measure that allows a CSP to request pertinent trust resources from other CSPs. The scheme they proposed benefits CSPs and customers, enabling inexpensive data processing but achieving high gains in security. Their proposed method provides ideas for determining server trust levels used in the article. However, this approach still does not sufficiently consider the server and storage field fitness.

2.3 Server Allocation Strategy

Server allocation strategy: Halabian et al. [21] provided supplementary findings on delay-optimal server placement in multi-queue multi-server (MQMS) systems with asymmetric connectivity. They demonstrated that MWM minimizes a variety of cost functions of the queue lengths, such as total queue occupancy, in the stochastic ordering sense for a system with Bernoulli arrivals and connectivity (which implies minimization of average queueing delay). Through simulations, they demonstrated that this strategy performs quite similarly to the ideal strategy in terms of the average queueing delay. Sawa et al. [54] proposed an approach called Minimizing the Maximum Delay (MMD) for the distributed server allocation problem, where the best result is attained when all server-server delays have the same fixed value. They demonstrated that MMD achieves an ideal result with polynomial time complexity. Zhang et al. [11] introduced tagged-MapReduce that provides safe processing with mixed-sensitivity data on hybrid clouds. They provided a general security framework that captures how dataflow can leak information via execution for analyzing MapReduce computations in the hybrid cloud. Karim et al. [46] proposed a mechanism to map the users' QoS requirements of cloud services to the right QoS specifications of SaaS. They also proposed a set of rules to perform the mapping process and hierarchically model the QoS specifications of cloud services using the Analytic Hierarchy Process (AHP) method. The AHP based model helps to facilitate the mapping process across the cloud layers, and to rank the candidate cloud services for end users. Vimercati et al. [49] defined a security model based on abstract security features, and they suggested a method for allocating resources to cloud services. The owner's encryption was taken into account when allocating resources to services, as were any global criteria the owner might have set to minimize overhead and over-segmentation of resources among different services. Although all the above approaches to data allocation improve user experience or security to a greater or lesser extent, most of them do not involve data privacy enhancement, and even if they do, they do not prevent critical information

from being inferred, and none of them consider constraints such as budget and cooperation between server providers while allocating the most appropriate server for the data.

2.4 Supplier Selection

Reducing carbon emissions when developing procurement programs is a crucial and comprehensive issue, and it is also a challenge to consider all aspects in an integrated manner. Therefore, many works have made trials from this perspective.

Cui et al. [38] investigated a manufacturer's multi-period, multi-product supplier selection and order allocation challenge. The manufacturer has two options: invest in current suppliers, or find new ones. The development of a new mixed integer programming mathematical model takes into account the diverse requirements of many suppliers. Carvalho Fagundes et al. [43] proposed a computational method for supplier selection using the "Fuzzy Extended Analytic Hierarchy Process (Fuzzy AHP)". The study's findings highlighted significant possibilities for enhancing supplier selection while taking risks into account, including choice rationalization, adaptable decision variable modeling, and automation of subjective assessments and evaluations by decision-makers. The competitiveness of the supply chain may be enhanced by these advantages, which may help reduce purchasing costs and enhance the overall assessment of supply risks. Xia et al. [52] developed a set of electric power industry characteristics of a green supplier evaluation index system against the backdrop of carbon green supplier evaluation value. It also looks at the various applications of green evaluation criteria, which give the power industry a foundation for comparison when choosing suppliers and are crucial for driving the green development of the industrial chain. Lamba and Singh [36] proposed a model for supplier selection and order allocation in a dynamic environment with multiple products, periods and suppliers. And the carbon emissions due to purchasing, inventory, ordering and transportation are integrated into the model. They map the parameters used in the proposed model to various dimensions of big data, and analyze the model numerically and demonstrate its prospects using two randomly generated datasets with big data characteristics. Singh et al. [5] proposed a big data cloud computing framework for carbon minimization. And they show how slaughterhouses and processing plants can use the captured carbon footprint information for eco-friendly supplier selection of beef cattle, which can help agri-food industry players curb their emissions. At the same time, the framework has universal properties that can be emulated and configured for any food supply

chain. Kumar et al. [2] proposed a generic and integrated approach for flexible supplier selection considering both cost reduction and environmental efficiency objectives. The model has the new ability to enforce emission standards and can be adapted to region-specific emission limits. They studied the optimal supplier selection problem in a fashion apparel supply chain in the presence of a carbon tax. Choi [56] proposed a two-stage optimal supplier selection scheme, in which the first stage filters inferior suppliers and the second stage selects the best supplier among the set of non-inferior suppliers through multi-stage stochastic dynamic programming. These studies have proposed supplier selection options under certain specific scopes, but each of them still lacks certain important impression factors that may be involved in real-life situations, such as quality standards of items, working hours that suppliers can provide, matching relationships between items, etc.

2.5 GRA Related Problems

GRA related problems: Zhu [22] formalized the group role assignment problem when faced with the constraint of conflicting agents, proved that such a problem is a subproblem of the extended integer linear programming (x-ILP) problem, proposed a practical approach to the solution, and assured performance based on the results of experiments. Zhu [23] presented and formalized a challenging role assignment problem with budget constraints (GRABC), provided a set of practical solutions to different forms of this problem by using the IBM ILOG CPLEX optimization package, established a set of necessary conditions for these problems to possess feasible solutions to improve the CPLEX solutions and verified the practicability and efficiency of the proposed solutions. Zhu et al. [26] clarified the group role assignment problem (GRAP), described a general assignment problem (GAP), converted a GRAP to a GAP, proposed an efficient algorithm based on the Kuhn-Munkres (K-M) algorithm, conducted numerical experiments, and analyzed the solutions' performances. Zhu et al. [27] presented a challenging problem called group role assignment with cooperation and conflict factors (GRACCFs). The authors formalized the proposed problem, identified the problem’s complexity, and solved the problem using the IBM ILOG CPLEX optimization package (ILOG). The authors verified the benefits of solving the GRACCF problem through simulations. These works provide great support and help formalize and model the problems in the article.

2.6 Summary

In this section, we present the work related to data storage in the cloud, assessing data sensitivity and server trust, server allocation policies, supplier selection and problems related to GRA and present their features and shortcomings.

Chapter 3

3 Statement and Formalization of the Problem

Chapter 3 first presents a real-world scenario, after which we introduce the general solution strategy of the proposed method in this scenario, and then give a detailed description of the method and the processing steps.

3.1 Real-World Scenario

Eddie, the Chief Information Officer (CIO) of X, a technological company, uses the cloud to store the company’s data. However, a recent attack on the cloud server resulted in the disclosure of some employees' personal data. Bob, the Chief Executive Officer (CEO) of company X asked Eddie to find a data storage solution that would safeguard their private data to store the new data generated by the company and set aside a budget of $30,000 per year for the storage solution. Within this budget, the company also requested that different data fields need to be stored among different service providers.

This is undoubtedly a challenge for Eddie. Eddie needs to consider the budget, privacy protection, data access performance, trust of servers, and potential data sharing of servers between service providers. Eddie tried a few existing methods and noticed that none of them satisfied all his requirements.

Table 3.1: The Price of Each Server

Servers	Price
Cloud Server 1	$3500
Cloud Server 2	$3200
Edge Server 1	$5500
Edge Server 2	$5800
Edge Server 3	$5000
Edge Server 4	$4500
Edge Server 5	$4900
Edge Server 6	$5000
Edge Server 7	$4500
Edge Server 8	$4800

Edge Server 9	$6200
Edge Server 10	$5800

Table 3.2: The Budgets for Data Fields

Data Fields	**Full Name**	**Date of Birth**	**Telephone**	**Address**
Budget Limits	$6000	$5000	$9000	$10000

Table 3.3: The Conflict Relations Between Servers

Servers	**Conflicts**	
Edge Server 1	Edge Server 2	
Edge Server 2	Edge Server 1	
Edge Server 3	Edge Server 6	Edge Server 9
Edge Server 5	Edge Server 7	
Edge Server 6	Edge Server 3	Edge Server 9
Edge Server 7	Edge Server 5	
Edge Server 9	Edge Server 3	Edge Server 6

Table 3.4: Qualifications of Servers for Data Fields

Servers	**Data Fields**			
	Full Name	Date of Birth	Telephone	Address
Cloud Server 1	**0.72**	**0.71**	0.33	0.28
Cloud Server 2	0.64	0.70	0.24	0.31
Edge Server 1	0.56	0.55	0.55	0.61
Edge Server 2	0.61	0.59	0.58	0.66
Edge Server 3	0.32	0.44	0.72	0.75
Edge Server 4	0.25	0.45	**0.85**	0.72
Edge Server 5	0.43	0.35	0.67	0.87
Edge Server 6	0.53	0.46	0.76	**0.76**
Edge Server 7	0.43	0.37	**0.88**	0.75
Edge Server 8	0.22	0.29	0.59	**0.86**
Edge Server 9	0.24	0.26	0.63	0.65
Edge Server 10	0.38	0.44	0.71	0.73

We propose a new way to help him meet his requirements. We first separate the data to be saved into four data fields. Assuming he can look up these fields from a system, we then calculate the sensitivity of each data field using the evaluation strategy and find that the phone number and address are sensitive fields, and the rest are non-sensitive fields. Then, based on the servers' locations, we selected two public cloud servers and ten edge servers for data storage, and calculated the price of each server as shown in Table 3.1. Additionally, we assess the trust and

performance rankings of all available servers to determine the trust score and performance score, and then combine these with the data field sensitivity to calculate a server's qualification value for storing the fields as shown in Table 3.4. Based on the sensitivity of the data fields, we allocate the total budget to each field proportionally as shown in Table 3.2. The unit of data in the table is $1000, and the listed conflicts between servers due to service providers are shown in Table 3.3. We intend to store just one sensitive field per edge server to protect the privacy of sensitive fields, although up to four non-sensitive fields can be stored on each cloud server. Sensitive fields require an additional backup. Finally, we set a series of constraints and obtain the optimized assignment. In this scenario, it is necessary to assign the fields with different sensitivities to different types of servers while taking into account the capacity, performance, price and conflicts of the servers, etc. The final optimal assignment result is shown in Table 3.4. The total sum of assigned qualification values is 4.78.

3.2 Data Storage in Edge-cloud Environment

Based on the above scenario, we give the following data storage scheme in an edge-cloud environment. Firstly, we partition the data into distinct data fields for storage, which prevents others from obtaining the whole data and deducing all the information from the leaked data compared to the method of splitting the data into multiple smaller datasets. In order to find an optimal data storage solution, we need to introduce two parameters: the sensitivity of the data fields and the trust of the server. The sensitive fields must be stored on servers with high server trust levels, while the corresponding non-sensitive fields may be stored on servers with low server trust levels. We believe that the edge servers have better performance in privacy protection, due to their properties of self-controllability and less accessibility to the public.

It is possible to determine the sensitivity of data fields by referring to the method in [58]. First, the category of sensitive data and their relative sensitivities are determined by data specialists in the corresponding fields. A set of text analysis and classification tools are then applied to automatically find sensitive information in the data, estimate the sensitivity level of each data field, and finally map the sensitivity level to a value between 0 and 1. Alternatively, a more straightforward strategy is also an option. For instance, we can develop various data sensitivity levels based on a sensitive data classification strategy and label each data field with a sensitivity level in accordance with the rules. We can assess server trust using three factors by

referring to the calculation method in [53]: the ranking of cloud service providers for their own resources, the ranking of cloud service providers among themselves, and the rating of cloud service providers by third-party organizations.

Users can set a sensitivity threshold based on their needs after obtaining the sensitivity and trust values. Data fields that are over this threshold are sensitive fields and need to be stored in the edge servers. The privacy of the data can be better protected by storing it fully on edge servers. However, in practice, this storage approach is too expensive and the scale of the edge server may not meet the demand because we do not include unstable edge servers, such as some mobile devices, in the storage function. Additionally, in order to maximize data privacy, we assume that each edge server only stores one data field. This is because the more fields stored together, the easier it is to infer privacy information based on these fields [39]. For data saved on public cloud servers, the user can customize the number of data fields to be stored based on their requirements or the performance of servers [50].

When assigning servers to store the data fields, we primarily examine establishing an optimal assignment strategy so that each data field is saved on the most appropriate server, namely, with the best degree of fitting in sensitivity and trust from the perspective of the whole data to be stored. We can first map the server's trust value to the sensitivity value required by the data field according to the sensitivity categorization in order to determine the fitness. The smaller the difference between the two sensitivities, the better the fitness. We also consider the data's performance requirements for the server while allocating resources. We aim to store the data in the server that best satisfies its performance requirements, similar to sensitivity and trust. We concentrate on the server's CPU, memory and bandwidth as the three critical components of server performance. These three metrics are ranked across all discovered servers to get three sets of scores, and the corresponding performance requirement scores are obtained based on the performance requirement mapping of the data fields. A smaller final difference between the two indicates that the server's performance meets the requirement.

In addition, we introduce several limitations that might be present in light of the realistic scenario, including the physical location of the servers, the user's budget, and the server conflicts with the same service provider. Users might exclude some cloud servers from their lists of

options based on their requirements because the frequency of network latency and attacks varies depending on the server's location. Each server will have a varied price due to variations in performance, geographic location, security, and other factors. As a typical consideration of customers, a budget should also be regarded as a key limitation. Too many different fields stored by servers belonging to the same service provider may raise issues regarding data privacy. Suppose service provider A owns two servers, and if the two servers store different fields, then A may obtain information about the data in both servers and infer more privacy details. Thus, this situation is also necessary to avoid. It is also important to note that all available servers are optional for users. And there is a cache-level storage workspace for end users to query data.

3.3 Condensed/Extended E-CARGO Model

The E-CARGO model can assist us in the study related to role assignment, through which a system $\sum$ can be described as a nine-tuple $\sum ::=< C, O, A, M, R, E, G, s_0, H >$. In the model, C denotes a set of classes, O denotes a set of objects, A denotes a set of agents, M denotes a set of messages, R denotes a set of roles, E denotes a set of environments, G denotes a set of groups, s_0 denotes the initial state of the system, and H is a set of users. Our study will then concentrate on A and R, our servers and data fields.

In the discussed scenario, we take servers as agents, and data fields as roles. Then, the number of servers that can be expressed by the non-negative integer $m\ (= |A|)$, likewise, $n\ (= |R|)$ is the number of data fields that we can express, $i, i_1, i_2, \ldots$ are the indices of servers, and $j, j_1, j_2, \ldots$ are the indices of data fields. The servers are divided into two categories: cloud servers and edge servers. The number of edge servers is denoted by m_e, and the number of cloud servers is denoted by m_c, therefore we can deduce that $m = m_e + m_c$. Similarly, data fields are separated into sensitive and non-sensitive groups. A threshold *t* for sensitivity score can be set by the user. Sensitive fields are those with values greater than or equal to this threshold, and vice versa are non-sensitive fields. The number of sensitive fields is denoted by n_s, and the number of non-sensitive fields is denoted by n_n, thus $n = n_s + n_n$.

Definition 3.1: The role range vector L [23], [24] is an n-vector that contains the lower bounds of the ranges of roles in the environment of a group. In our scenario, it reflects the number of backups made for the data fields, i.e., $L[j] \geq 1 (0 \leq j < n)$. If no backup is required, $L[j] = 1$, and if another backup is required, $L[j] = 2$.

Definition 3.2: The agent ability vector [22], [23] is an m-vector L^a where $L^a[i] \in \mathbb{N}$ indicates the maximum number of data fields that can be stored per server i $(0 \leq i < m)$.

Definition 3.3: The data field sensitivity score SS is a n-vector, where $SS[j] \in [0,1]$ represents the sensitivity score of field $j \in \mathbb{N}$ $(0 \leq j < n)$.

Definition 3.4: The server trust score TS is an m-vector, where $TS[i] \in [0,1]$ represents the trust score of server i $(0 \leq i < m)$. A function $f(TS[i]) = SS',\ SS' \in [0,1]$ based on user needs can convert a TS value to a data field's required sensitivity score SS' for a subsequent fit score calculation.

Definition 3.5: The fitness score matrix FS is an $m \times n$ matrix, where $FS[i,j] \in [0,1]$ represents the degree of fit of the trust level of server i $(0 \leq i < m)$ and the sensitivity of data field j $(0 \leq j < n)$,

$$FS[i,j] = \begin{cases} 0, & SS'[i] < SS[j] \\ 1 - (SS'[i] - SS[j]) \end{cases}. \tag{3.1}$$

Definition 3.6: The performance score matrix PS is an m-vector, where $PS[i] \in [0,1]$ represents the matching of the performance of server i $(0 \leq i < m)$ to the requirement of dataset. PS consists of three primary parts, namely, CPU, memory and bandwidth. The performance levels of all servers are sorted from low to high according to these components, and then mapped into three scores PS_1, PS_2, PS_3 between 0 and 1 according to different levels. The performance requirements of the data fields for the server are also mapped into three scores PS'_1, PS'_2, PS'_3 between 0 and 1, and then

$$PS[i] = \begin{cases} 0, PS_1[i] < PS'_1 \text{ or } PS_2[i] < PS'_2 \text{ or } PS_3[i] < PS'_3 \\ 1 - ((PS_1[i] - PS'_1) + (PS_2[i] - PS'_2) + (PS_3[i] - PS'_3))/3 \end{cases}. \tag{3.2}$$

Definition 3.7 [26]: The qualification matrix Q is an $m \times n$ matrix, where $Q[i,j] \in [0,1]$ represents the qualification value of server i $(0 \leq i < m)$ for field j $(0 \leq j < n)$.

$$Q[i,j] = u \times FS[i,j] + (1-u) \times PS[i], \tag{3.3}$$

where $u \in [0,1]$ is a coefficient indicating the weight of FS in the Q. Note that if the value of FS is 0 or the value of PS is 0 then Q is also 0.

Definition 3.8: The storage size of the server S is an m-vector, $S[i] \in \mathbb{N}\ (0 \le i < m)$, which is measured in gigabytes.

Definition 3.9: The size of each data field D is an n-vector, $D[j] \in \mathbb{N}\ (0 \le j < n)$, which is measured in gigabytes.

Definition 3.10 [26]: The definition of the role assignment matrix T is an $m \times n$ matrix, where $T[i,j] \in \{0,1\}\ (0 \le i < m, 0 \le j < n)$ indicates whether the server i is assigned to the field j. If $T[i,j] = 1$, it indicates "yes" and 0 indicates "no".

Definition 3.11 [26]: The qualities of the allocated agents are added together to determine the group performance σ of group g, which is

$$\sigma = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} Q[i,j] \times T[i,j]. \quad (3.4)$$

Definition 3.12 [26]: If role j has been given a sufficient number of agents, it is workable in the group, i.e., $\sum_{i=0}^{m-1} T[i,j] \ge L[j]$.

Definition 3.13 [26]: If each role in T is workable, i.e., $\forall (0 \le j < n) \sum_{i=0}^{m-1} T[i,j] \ge L[j]$, then T is workable. If T is workable, then g is as well.

Definition 3.14: Using the definitions listed above, our data allocation problem can be defined to determine a matrix T to obtain

$$\sigma_0 = max\left\{\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} Q[i,j] \times T[i,j]\right\}$$

$$\text{subject to } T[i,j] \in \{0,1\}\ (0 \le i < m, 0 \le j < n) \quad (3.5)$$

$$\sum_{i=0}^{m-1} T[i,j] = L[j]\ (0 \le j < n) \quad (3.6)$$

$$\sum_{j=0}^{n-1} T[i,j] \le L^a[i]\ (0 \le i < m) \quad (3.7)$$

$$\sum_{i=0}^{m_c-1} T[i,j] = 0\ (n_s \le j < n) \quad (3.8)$$

$$\sum_{j=0}^{n-1} T[i,j] \times D[j] \le S[i]\ (0 \le i < m) \quad (3.9)$$

Where constraint (3.5) indicates whether a server is assigned; (3.6) makes the group workable; (3.7) indicates that each server is only assigned to a limited number of fields; (3.8)

indicates that cloud servers cannot be assigned to store sensitive data fields; (3.9) indicates that the size of all data fields stored by the server cannot be larger than the server storage size. With these constraints we can find an optimal allocation.

Note that, in the above definitions, we assume that the server set is ordered without the loss of generality, i.e., all the cloud servers are indexed between 0 and $m_c - 1$, inclusively, and all the edge servers are indexed between m_c and m-1, inclusively.

3.4 Allocation Problem with Conflict and Budget

In reality, the problem is not simple as that in Definition 3.14. We need to introduce two new constraints, budget and conflict. For the budget constraint, taking into account that the importance of data fields with different sensitivities may vary, users may assign higher budgets to highly sensitive fields to enhance data privacy, and therefore assign different budgets for fields with varying sensitivities. The price of the server used for storage cannot exceed the budget for the corresponding field. The definitions are as follows.

Definition 3.15 [23]: The budget vector is an n-vector B, where $B[j] \in \mathbb{N}$ denotes the maximum cost for storing each field j $(0 \leq j < n)j$ $(0 \leq j < n)$.

Definition 3.16: The price vector is an m-vector P, where $P[i] \in \mathbb{N}$ denotes the cost per server i $(0 \leq i < m)$ usage.

Definition 3.17: Service providers SPs possess all cloud and edge servers for data storage, each server must belong to an SP, and different servers may belong to the same SP. For instance, $SP_1 = \{i_1, i_2, i_3, i_4\}$ indicates that server i_1, i_2, i_3, i_4 belong to SP_1.

Definition 3.18: If two servers belong to the same SP, the two servers are not allowed to store data fields that are different from those already stored, i.e., there is a conflict between the servers.

Definition 3.19 [22]: The definition of the matrix of conflicting agents is an $m \times m$ matrix A^c $(A^c[i_1, i_2] \in \{0,1\},\ 0 \leq i_1, i_2 < m,\ i_1 \neq i_2)$, where $A^c[i_1, i_2] = 1$ denotes that server i_1 conflicts with server i_2, and 0 denotes not.

Using the definitions listed above, we can redefine our data allocation problem to find a workable T to

$$\sigma_0 = max\left\{\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} Q[i,j] \times T[i,j]\right\}$$

subject to (3.5) - (3.9) and

$$\sum_{i=0}^{m-1} T[i,j] \times P[i] \leq B[j] \ (0 \leq j < n) \quad (3.10)$$

$$A^c[i_1,i_2] \times \ (T[i_1,j_1] + T[i_2,j_2]) \leq 1 (0 \leq i_1 \neq \ i_2 < m, \ 0 \leq j_1 \neq j_2 < n) \quad (3.11)$$

Where constraint (3.10) indicates a field's budget cannot be exceeded by the cost of all servers utilized to store that field; (3.11) indicates that if a server is present in the allocation scheme, then another server that conflicts with it cannot be used to store other data fields.

The proposed problem is very similar to the knapsack and bin packing problems in that it is also about finding an optimal value under constraints. This is reflected in finding the allocation solution that maximizes the sum of the assigned qualification values in Q under certain budget conditions. However, it is distinguished from the knapsack or bin packing problems in that we can incorporate some of the constraints into the process of computing Q. For example, if the SS' is lower than the SS then FS is 0, then the corresponding Q value is also 0. If any of the server performance indicators do not meet the demand then PS is 0, then the Q value is also 0. This can reduce a large number of constraints on the judgment. In addition, there is a server conflict constraint in our problem, which means that if a server exists in this allocation scheme, servers that conflict with that server will be excluded.

According to the proof in [22], GRA with conflicting agents in a group (GRACAG) problem is a subproblem of the x-ILP problem but may not be NP-complete. It informs that our problem is likewise a subproblem of the x-ILP problem.

Constraints (3.5), (3.6) and (3.8) apply to our proposed problem and are consistent with the constraints in GRACAG.

Constraint (3.7) adds a new condition check,

for i from 0 to $m_c - 1$ do

if ($\sum_{j=0}^{n_n-1} T[i,j] > L^a[i]$) return false;

return true;

For constraint (3.8), the algorithm is:

for i from 0 to $m_c - 1$ do

  for j from n_n to $n - 1$ do

    if ($T[i, j] > 0$) return false;

return true;

For constraint (3.10), the algorithm is:

for j from 0 to $n - 1$ do

    if ($\sum_{i=0}^{m-1} T[i, j] \times P[i] > B[j]$) return false;

return true;

Constraint (3.11) is the same as that of GRACAG.

It can be observed that our problem is based on a search space similar to that of the GRACAG problem, and requires judgments on additional constraints. It advises that our problem is more sophisticated than the GRACAG problem, making it a subproblem of the x-ILP problem.

3.5 Experiment settings

Based on the analysis of the complexity, we know that it is challenging to solve such a complicated problem, and we need to discover a way that can do it in a reasonable amount of time. The IBM ILOG CPLEX (CPLEX) platform is a tool that can be utilized to tackle x-ILP problems. In order to resolve the issue, we employ the CPLEX package. First, we must list all the components that make up the CPLEX package, where the variables of matrix T have upper and lower bounds of 1 and 0, respectively, and matrix Q represents the coefficients of the objective function. Subsequently, we add the expressions of the objective and constraints and find the optimal solution, where these expressions can be represented by some formulas. In this section, we simulate different real-world scenarios and design two types of experiments to analyze the effect of the solution. All experiments were conducted in the environment shown in Table 3.5.

Table 3.5: Experiment Environment

Hardware	
CPU	Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz 2.50 GHz
MM	12 GB
Software	
OS	Windows 10 64-bit
IDEA	Version: IntelliJ IDEA 2020.1.3 (Ultimate Edition)
JDK	Java 1.8.0_271

3.6 Performance Experiments

We simulated finding the optimal solution for various numbers of servers and field counts, then we tracked the program's execution time to assess how well the solution performs in a real-world scenario. The size of the simulated trials is assumed to expand 10 fields at a time beginning with 10 data fields until the runtime is unacceptable. Accordingly, 80% of these are public cloud servers. $L[j]$ =1 $(0 \leq j < n)$, i.e., there are no further backups needed for any data fields. $L^a[i] = 10\ (0 \leq i < m_c)$ and edge servers can only store up to one field, which means that $L^a[i] = 1(m_c \leq i < m)$. During the allocation process, if server 1 belonging to SP_1 and server 2 belonging to SP_1 are the most suited to store fields 1 and 2, respectively, but the two servers belong to the same SP, i.e., there is a conflict between server 1 and server 2, then we must avoid this allocation. Assuming that the sensitivity threshold $t = 0.5$, sensitive data fields are those that have a sensitivity score of at least 0.5. The sensitivity and size of the data are randomly generated, and the trust level of the server is determined by a set of random values that follow a normal distribution. The capacity, pricing, and performance are established at random with reference to the Microsoft Azure Cloud Server data, the probability of conflict between servers was set to 10%.

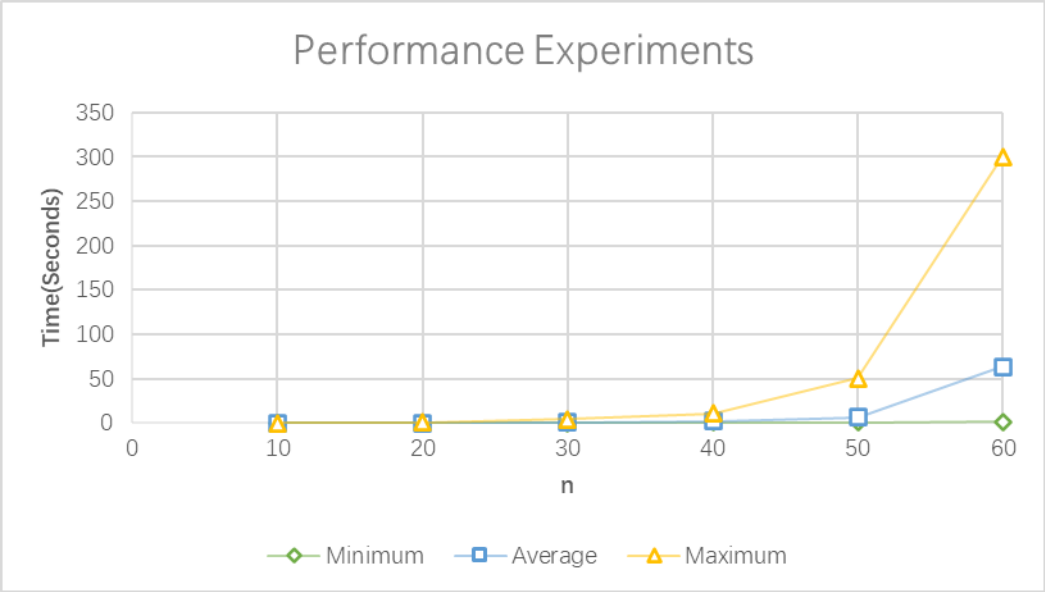

Figure 3.1: Performance Experiments Result.

We conducted 100 sets of experiments with various fields and server sizes, counting the minimum, average and maximum running times in each case. The results are displayed in Fig. 3.1. According to the results, when $n = 60$ which means the number of data fields is 60, the average running time is approximately 63.5 s and the maximum running time is 300 s. Although the average runtime is low when n is 10-50, the increasing trend of maximum and average runtime is quite significant when n increases to 60 fields, and it is projected that when n surpasses 100, the average running time will be measured in hours. As a result, in order to increase efficiency, we need further improvements to the solution.

3.7 Feasibility Experiments

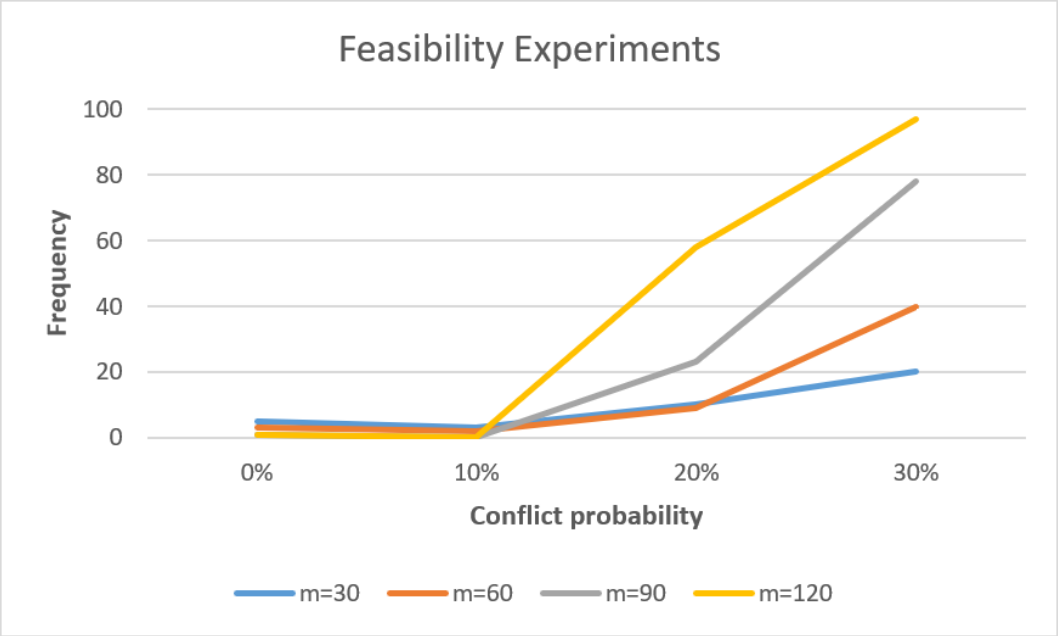

Figure 3.2: Feasibility Experiments Result.

In order to evaluate the feasibility of the solution, we conducted 100 experiments with various server conflict rates and server sizes and counted the frequency of infeasible solutions, and *m* is still three times *n* which means the number of servers is three times the number of data fields. The experimental results are shown in Fig. 3.2, where the vertical coordinates represent the number of infeasible solutions, and the horizontal coordinates represent the probability of conflict between servers. The figure illustrates how the number of infeasible solutions rises considerably when the conflict rate exceeds 10%. It also shows how this number rises as server size increases. For example, when there are 120 servers and the conflict rate is 30%, 97 out of 100 solutions are infeasible. Therefore, in real-world applications, especially when *m* is large, we should attempt to minimize the conflict rate between them below roughly 10%. High conflict rates both indicate that the scenario does not make sense in reality or additional service providers may need to be employed to complete the storage.

3.8 Summary

In Chapter 3 we present a real-world scenario and assume data and constraints, and introduce a method to solve the scenario. We then further describe the problem and briefly explain the

solution. We model the problem using the E-CARGO model, introducing different notations to describe the problem and giving definitions. For the calculation of the different metrics, we give detailed formulas. In addition we formalize the constraints in the model and explain the complexity of the problem. We use CPLEX to first conduct performance experiments, through which we find that the runtime is relatively small at a certain scale, but unacceptable when the number of data fields exceeds 100. We then test the feasibility of the method and find that the conflict rate of the server is around 10% which is reasonable, a higher conflict rate will lead to a large number of infeasible solutions and a high conflict rate is not meaningful in practice.

Chapter 4

4 Improvement of the Allocation Scheme

We can observe from the performance experiment results that the performance needs to be improved. Through experiments, we discovered that overly complicated server conflict constraints were the primary reason. In the same setting as the earlier performance experiments, we eliminated the server conflict constraint and ran the experiments for various numbers of fields independently. Detailed results are shown in the appendix. The runtime without adding conflict constraints is significantly lower compared to the method with conflict constraints. Therefore, in order to improve the performance, we need to focus on improving the conflict constraint while reducing the overall operation size as much as possible. We made a series of initial trials to improve the method, but none of them achieved a good result, see the appendix for details.

4.1 Genetic Algorithm

After trying some improvement methods without satisfactory results, we also tried to use genetic algorithms [30] to explore whether performance improvements could be achieved.

In this method, the number of individuals is the size of the role assignment matrix T, i.e., $m \times n$. Each generation uses all the allocation process constraints as screening requirements, and the individual with the greatest Q is then chosen for future evolution after the screening process is complete. In the initial experiments, we discovered that each initialized population could not satisfy the constraints we set from the beginning due to the complexity of the constraints, and the experiments could not be carried out. To solve this issue, we use CPLEX to first find a feasible solution for the initial population so that we can further search for individuals with higher Q. When the mutation probability and crossover probability are both equal to 0.7 after running experiments with various parameter values, we can get better results; otherwise, the evolution can rarely move forward. We conducted 100 experiments with m = 120 and n = 40 which means the number of severs is 120 and the number of data fields is 40. The other settings remained the same as those in the previous experiments. A population size of 1000, and an evolutionary generation of 5000 were set. On average, the optimal individual appeared in the 4724th generation, with an average maximum value of 20.11 and an average running time of 233.39 s. The value of the optimal solution using the improved method under the same conditions was 32.37. From the results, we can see that it is possible to find the highest value only if the

population size and the number of genetic generations are large enough, and the performance of excellent individuals evolved with this strategy is very low.

To further improve it, we attempt to set the number of individuals to the number of servers in conflict, initialize the population with CPLEX as before, where each mutated individual actually represents the available servers in conflict, combine each individual to other servers to find a feasible solution using CPLEX, and select the largest value in the population. For the following evolution, the individual with the highest value in the population is chosen. After performing the same experiments on the modified genetic method, we found that while the number of generations needed to produce better individuals did decrease significantly, the time to find the best solution was still far from satisfactory. Each generation requires a large number of computations, and if the population size is 100, then each evolutionary generation requires 100 computations using CPLEX, while the average run time per generation is about 1.7s. Therefore, if we set hundreds of evolutionary generations, the final total time will be very long and far from acceptable.

4.2 Greedy Algorithm

We also attempted to use the greedy algorithm [6], a common method for finding the optimal solution, to tackle the problem. Following the principles of the greedy algorithm, we choose the server with the highest Q value for each field, and we set all the constraints that must be met by that server. If there is a server that conflicts with that server, then the conflicting server is flagged, and the field is not allowed to be assigned to it. We conducted 100 experiments with 120 servers, 40 data fields. Detailed experimental results can be found in the appendix. According to the statistics of the experimental results, the values obtained by the greedy algorithm were on average 1.93% less than those obtained by CPLEX, and in addition, there were 8 cases of no solution in 100 experiments. Subsequently, to further improve the results of the greedy algorithm, we computed the edges from each field to all servers and selected the largest edges for assignment. The experimental results show that the value obtained by the modified greedy algorithm is 1.02% less than the value of the CPLEX result. Additionally, there are 15 instances in 100 experiments when there is no solution. Although the difference between the result value of the greedy algorithm and the result value calculated by CPLEX is not large, the greedy algorithm generally has trouble finding the real optimal solution and there are many cases where no solution can be found.

4.3 Generalizing Conflict Relationships

When analyzing the conflict constraints, we consider that the definition of server conflicts actually comes from the service providers, i.e., SPs. The computation of conflict constraints can be reduced by generalizing conflicts between servers to conflicts between servers and the entire set of servers belonging to the SP.

Definition 4.1: $A^{c\prime}$ is a list of all servers provided by SPs, e.g. $A^{c\prime} = [< i_1, i_2 >, < i_3, i_4 >]$ indicates that SP_1 provides two servers 1 and 2, SP_2 provides servers 3 and 4. The number of servers provided by each SP is m_{sp}. Constraint (3.11) needs to be changed to:

$$if \sum_{j=0}^{n-1} T\left[A^{c\prime}[k][i], j\right] \geq 1 \rightarrow \sum_{i\prime=0, i\neq i\prime}^{m_{sp}} \sum_{j'=0, j\neq j\prime}^{n-1} T\left[A^{c\prime}[k][i'], j'\right] = 0 \ (0 \leq k < n_{sp}, 0 \leq i < m_{sp}) \tag{3.11}$$

Where n_{sp} is the number of service providers. In the same setting as before, we conducted 100 experiments with various server and field sizes. We assume that each SP provides 3 servers, i.e., m_{sp}=3, and the number of cloud servers is 80% of the number of all servers m, i.e., m_c= m×80%, and the remaining servers are edge servers. Each cloud server and each edge server can store up to 10 fields and 1 field, respectively, without a backup. According to the assumption that there are conflicting relationships between the three servers provided by each SP. The runtime at various sizes is shown in Fig. 4.1. In contrast to the results in Fig. 3.1, the running time of the improved method is significantly lower compared to the original method, where the average running time of the original method is about 63.5 s while the average running time of the improved method is about 0.5 s when the number of data fields is 60, which is only 0.8% of the results of the original method. And the final result value is the same as the original method is the result value of the optimal solution.

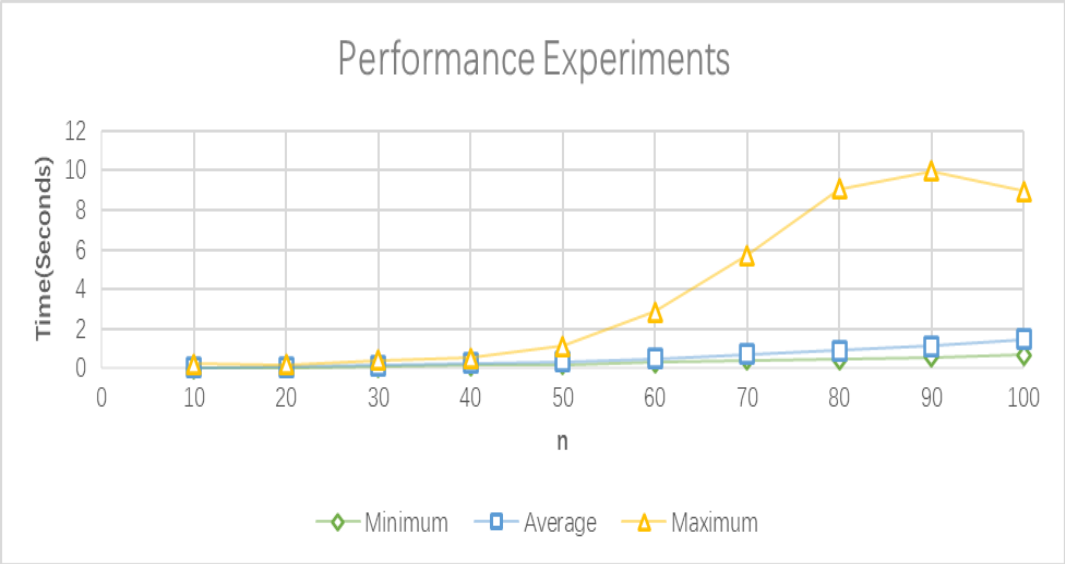

Figure 4.1: Performance Experiments Result of the Improved Method.

4.4 Conflict Constraints in Realistic Scenario

According to the results of the above improvement, it can be seen that the efficiency of the solution has been greatly improved, but there are also unreasonable aspects of the scenario. If we need to store many data fields and restrict too many fields to be stored in the same SP's servers, then we need at least the same number of SPs as that of data fields, which is unrealistic. In addition, in this scenario we believe that any two sensitive fields can infer critical privacy information, but in practice the number of such fields is often very few. Therefore, we need to further adapt the experimental scenario and modify the method to meet the realistic situation.

In order to satisfy practical needs, we further segment the field categories and divide sensitive fields into highly sensitive fields and low sensitive fields. Highly sensitive fields refer to the fields that are highly likely to be inferred from privacy information when combined with other sensitive fields, and low sensitive fields are those having a certain probability of inferring privacy information from them, and the previously mentioned non-sensitive fields are fields that have very low probability or even will not be inferred from privacy information. The number of highly sensitive fields is denoted by $n_{\acute{h}}$, and the number of low sensitive fields is denoted by n_l, $n_s = n_{\acute{h}} + n_l$. Considering the fact that the number of highly sensitive fields is often very small in practice, we can assume that the number of servers can meet the storage requirements in the usual case. In addition, due to the specificity of highly sensitive fields, we only allow them to be stored in the edge server, and since the edge server has a high level of security we can ignore conflicts caused by service providers in the edge server. Thus, in this scenario we consider

removing the constraint (3.12). However, for low sensitive fields, we need to introduce a new matrix to help us measure the impact on information privacy when multiple low sensitive fields are stored on the cloud servers of the same SP.

Definition 4.2 [27]: The matrix C^{cf} is an $n_c \times 5$ matrix which is used to represent the impact on data storage privacy, i.e., Q values, when a certain two low sensitive fields are stored in the same SP 's server. Symbol n_c denotes the number of impact factors, i.e., $m_c \times n_l \times m_{sp} \times (n_l - 1)$. $C^{cf}[0]$ and $C^{cf}[2]$ denote servers, $C^{cf}[1]$ and $C^{cf}[3]$ denote stored fields, $C^{cf}[4]$ denotes the degree of impact on privacy and $C^{cf}[4] \in [-1, 0]$.

For example, when server i and server i' belong to the same SP and store the low sensitive fields j and j' with a privacy impact of -0.3, then $C^{cf}[k,\ 0] = i, C^{cf}[k,\ 1] = j, C^{cf}[k,\ 2] = i', C^{cf}[k,\ 3] = j', C^{cf}[k,\ 4] = -0.3$, then $Q[i,\ j] + Q[i',\ j']$ changes to $Q[i,\ j] + Q[i',\ j'] - 0.3 \times (Q[i,\ j] + Q[i',\ j'])$.

Now our problem is transformed into finding a T to

$$\sigma_0 = max\left\{\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} Q[i,j] \times T[i,j] + \sum_{k=0}^{n_c-1} C^{cf}[k,4] \times Q[C^{cf}[k,0], C^{cf}[k,1]] \times T'[k]\right\}$$

$T'[k] = T[i,\ j] \times T[i',\ j']$, and $T'[k] = 1$ if both $T[i,\ j]$ and $T[i',\ j']$ are 1 and $T'[k] = 0$, otherwise. Therefore, except for constraints (3.5) – (3.12) the problem also needs to satisfy the following constraints:

$$2T'[k] \le T\left[C^{cf}[k,0],\ C^{cf}[k,1]\right] + T\left[C^{cf}[k,2],\ C^{cf}[k,3]\right] \quad (4.1)$$

$$T\left[C^{cf}[k,0],\ C^{cf}[k,1]\right] + T\left[C^{cf}[k,2],\ C^{cf}[k,3]\right] \le T'[k] + 1 \quad (4.2)$$

To verify the feasibility of the method, we assume that $m = 100,\ n = 200$, $n_n = 180, n_l = 15, n_h = 5$, the number of SP is 10 and $m_{sp} = 10$, and perform one hundred simulations under that condition. The maximum run time in one hundred experiments is 64.05 s and the average run time is 25.95 s. We also compared the results with the same conditions without adding the impact factor constraint and found that the total qualification value of the assignment without this constraint was reduced by 15.26% on average compared to our method. Additionally, we conducted experiments with a larger number of data fields while keeping the

same number of servers, and the results are shown in Fig. 4.2. When the number of data fields is 400, the complete result is not obtained due to computer performance, so the maximum number of fields that can be calculated in this environment is around 350 when the number of servers is 100. The experiment shows that our method modified for the real-world scenario can effectively enhance the privacy of the data and is relatively acceptable in terms of operational efficiency and can obtain the optimal solution.

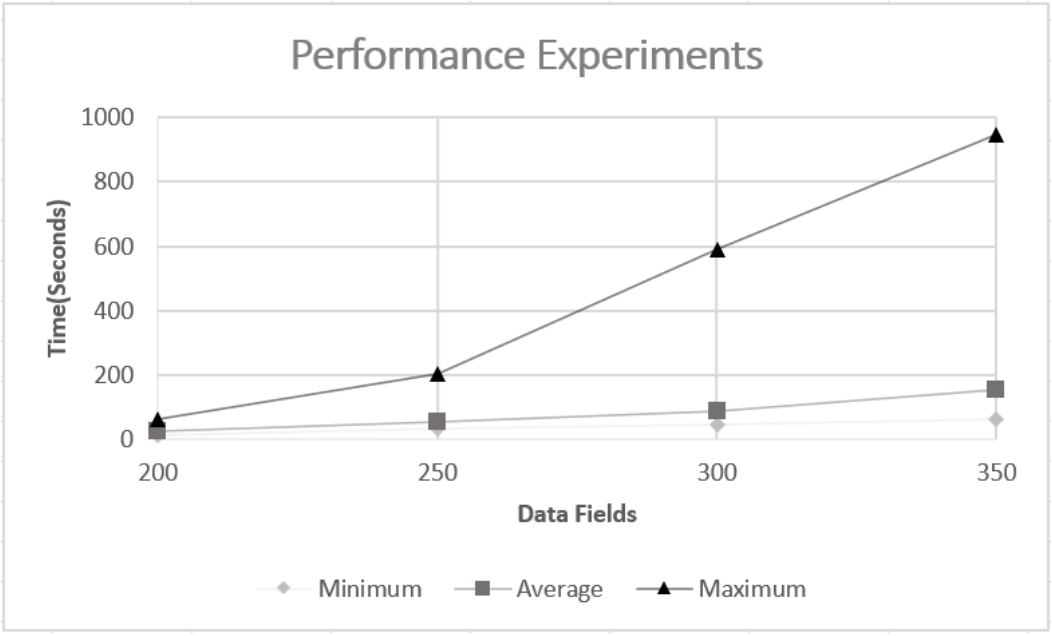

Figure 4.2: Performance Experiments Result of the Method for the Realistic Scenario.

4.5 Comparison Experiments

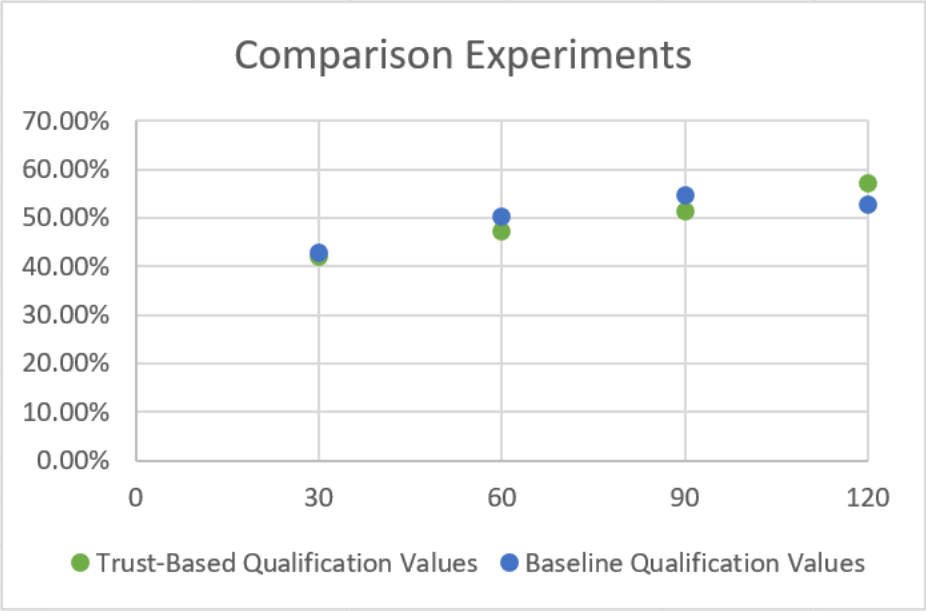

Figure 4.3: Comparison Experiments Result of Qualification Values.

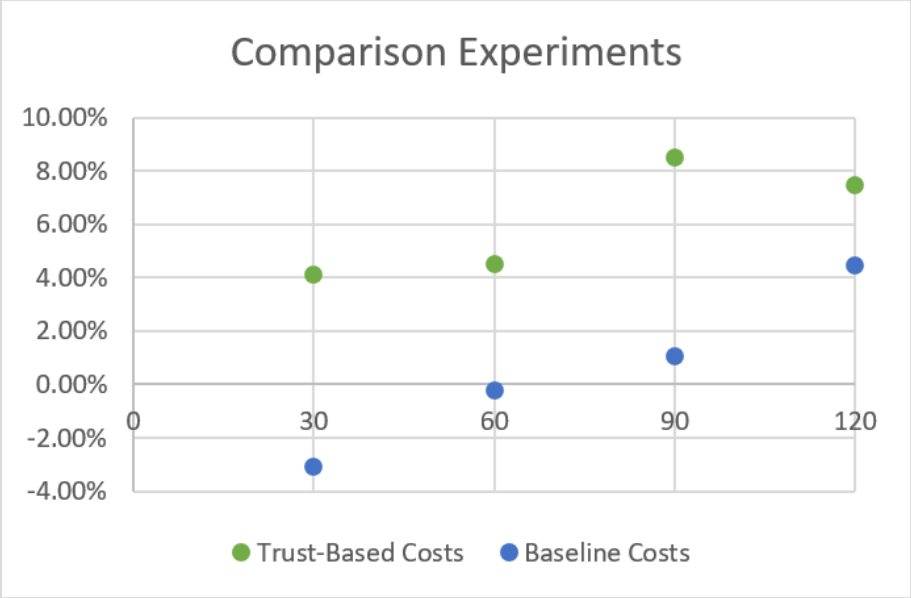

Figure 4.4: Comparison Experiments Result of Cost.

To test the effectiveness of our proposed method, we compare the improved method with the trust-based method and the baseline method. In the comparison experiments, we contrast our proposed method with the other two methods in terms of, respectively, qualification value Q and

cost. The fundamental idea of the trust-based strategy is to allocate the server with the highest trust level to the data, and the baseline method is a randomly found feasible assignment scheme that satisfies the conditions, and all approaches are executed in the same environment and under the same constraints. As in the other experiments, we ran 100 experiments with varied server sizes of 30, 60, 90, and 120, as well as data field sizes of one-third of the number of servers, and we tallied the average qualification values and costs and compared them. The experimental setting and conditions are similar to those of the earlier experiments. Fig. 4.3 demonstrates the percentage increase in the sum of the assigned qualification values, and the percentage decrease in the cost of our method compared to the other two methods is shown in Fig. 4.4. As can be seen, the sum of the assigned qualification values of our proposed method is 57.15% higher than the trust-based method and 52.75% higher than the baseline method, the costs of the trust-based method and the baseline method are, respectively, 7.48% and 4.47% higher than the cost of our method when the number of servers is 120. The results show that our proposed method outperforms the other two methods, and the effects are more pronounced the larger the server size. Overall, our approach allows for a better assignment of the most appropriate server to store the fields, while focusing on data privacy and reducing expenses.

We have to clarify that we may not be able to compare ours with existing methods in the related work (Chapter 2), because the defined data allocation problem is original, and no existing work can be used directly to solve such a problem.

4.6 Large Scale Data and Backups

We can observe from the performance experiment results that the performance needs to be improved. Through experiments, we discovered that overly complicated server conflict constraints were the primary reason. In the same setting as the earlier performance experiments, we eliminated the server conflict constraint and ran the experiments for various numbers of fields independently. Detailed results are shown in the appendix. The runtime without adding conflict constraints is significantly lower compared to the method with conflict constraints. Therefore, in order to improve the performance, we need to focus on improving the conflict constraint while reducing the overall operation size as much as possible. We made a series of initial trials to improve the method, but none of them achieved a good result, see the appendix for details.

In real-world scenarios, it occasionally occurs to store massive amounts of data. There could be thousands of different data fields in the dataset, and for non-sensitive or low sensitive fields, the cloud servers usually have enough resources to handle large-scale data. But in the case of highly sensitive fields, if the data contains a very large number of highly sensitive fields, then finding sufficient service providers and edge servers while keeping high privacy can be a challenge and we need to come up with some feasible solutions to solve this problem.

First, we can find all the duplicate data fields and remove them, and then determine if the numbers of the server and SPs meet the requirement of the data fields. If it is still impossible to guarantee that each edge server stores only one highly sensitive data field and different SPs store different highly sensitive data field, then we must sacrifice some privacy and permissions and let each edge server store multiple highly sensitive fields in order for the entire allocation scheme to be feasible.

Therefore, we consider compressing the field size. For instance, users can set multiple sensitivity ranges, and we can treat some data fields with similar sensitivity as a storage unit and designate a server to store these storage units. This makes it possible for a large number of data fields to be compressed into just a few hundred storage units. It is crucial to be aware that doing so can result in the inference of some private information, since if data stored on a server is compromised, others might well infer more private information based on these fields. Therefore, we advise combining multiple data fields into a single storage unit with the help of strategies based on correlations between the data fields or consulting experts in the area to which the data relates, ensuring as much as possible that no private information can be inferred from these consolidated fields. In addition, in order to ensure that the amount of data does not exceed the server's capacity, we should also obtain as much information as we can about the server configuration and evaluate the amount of data the servers can carry before assigning a server to each storage unit. If the volume of the data unit exceeds the capacity, we can determine the load range of the server and iterate the allocation process. The runtime after the field size reduction can be referred to the experimental results in Fig. 4.1. In addition, due to the limited storage capacity of the server, when data fields are compressed for storage, it may result in infeasible solutions. In the feasibility test we keep both the number of servers and the number of storage units at 100, and keep increasing the number of data fields to test the number of infeasible solutions for the case of a lower field compression rate. When the data field is 50,000, i.e., the

compression rate is 0.2%, 8 infeasible solutions are obtained in 100 solves. More infeasible solutions are generated when the compression rate is lower. Hence, it should be avoided that the field compression rate is too low.

Moreover, when the data fields need to be backed up additionally, it is equivalent to increasing the size of the data. To test the feasibility of our modified method, we use similar conditions as those in the previous experiments $m = 100$, $n = 100$, $L = 2$, which means that all fields need an additional backup, and then we reduce the number of fields by 10 in turn to test their feasibility at 100 servers.

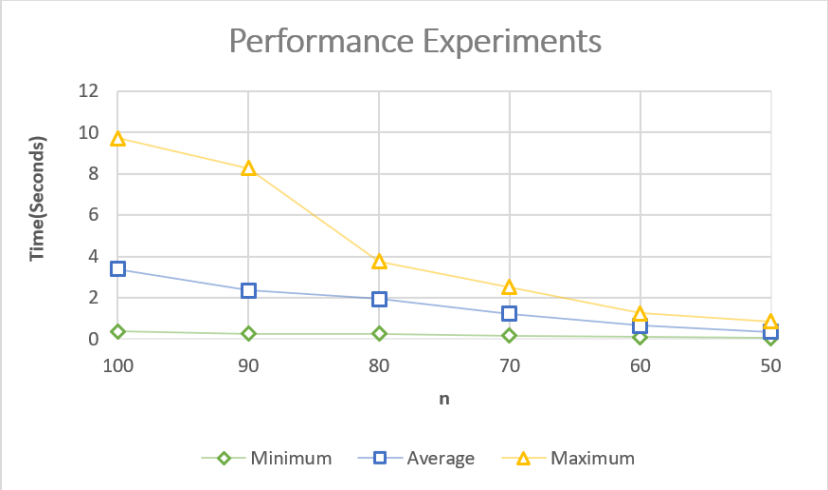

Figure 4.5: Performance Experiments Result for Additional Data Backup.

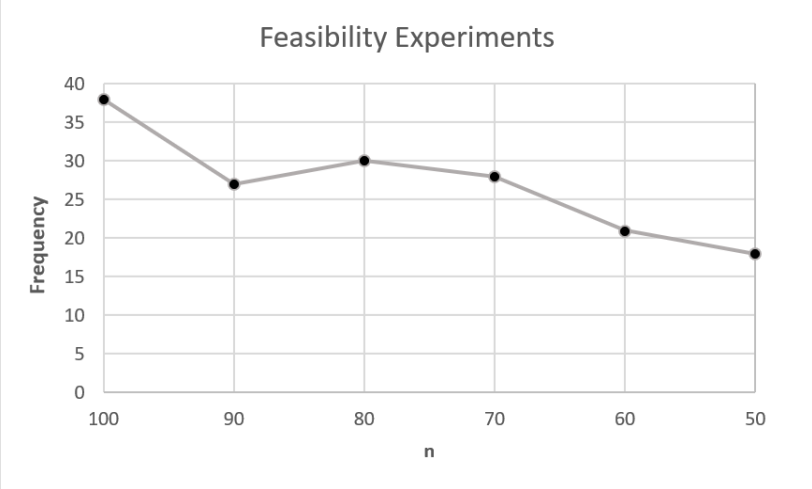

Figure 4.6: Feasibility Experiments Result for Additional Data Backup.

From the experimental results shown in Figs. 4.5 and 4.6, we can see that when the number of servers and data fields is 100, the maximum running time is nearly 10 s, the average running time is less than 4 seconds and its running efficiency is relatively acceptable. The rate of infeasible solutions is reduced from 38% to 18% by reducing the size of the fields while keeping the budget and the number of servers constant. Therefore, in order to improve the feasibility of the method with a certain server size, it is necessary to reduce the size of the fields, or it is necessary to increase the number of servers and service providers. In addition, from the experiment, when the number of fields is reduced to a certain number, it is difficult to continue to reduce the rate of infeasible solutions, and then it is necessary to increase the budget. When we double the budget, the number of fields is 50 and 60, and there are no infeasible cases. The budget for using servers increases where additional backups are needed, and it is difficult to find a feasible solution if the original budget is maintained. Hence, increasing the budget also improves the feasibility of the method in this situation.

4.7 Summary

In this chapter, we improve the proposed method. Specifically, we first try to solve the allocation problem using a genetic algorithm and a greedy algorithm, respectively, but the results of both are not as satisfactory. We then delve into the conflict constraints in the allocation problem and improve the constraints to improve the running speed significantly. We also compare the improved method with the trust-based method and the baseline method, and arrive at the result that the improved method is superior. Finally, we further adapt the method to scenarios with a larger number of servers and data fields as well as where data needs to be backed up.

Chapter 5

5 Extended Application of the E-CARGO Model

Nowadays, in the context of low carbon emissions, the procurement chain affects the carbon emissions in the supply chain. Inspired by the above approach, we realized the wide range of applications that the E-CARGO model can be applied to help us develop strategies to optimize the carbon footprint involved in procurement. Below is a sourcing scenario and a description of the problem.

5.1 Real-World Procurement Scenario

Eddie works for a company as a procurement officer and is in charge of developing the procurement program. In recent years, the company has established a series of emission reduction targets in response to efforts from the government and the community to reduce its carbon footprint while building a positive corporate image. Eddie is requested to develop a procurement strategy that takes into account three factors: cost, carbon emission and item quality. Eddie is instructed to consider all three factors as equally important, and is required to spend no more than $100,000 on the purchase and no more than 200,000 kg on the carbon emission. Procure 1,000 of each item. Under this condition, Eddie has to develop an optimal procurement strategy to satisfy the requirements of the company.

Table 5.1: The qualification Values for Items from Different Suppliers

Suppliers	Item 1	Item 2	Item 3	Item 4	Item 5
Supplier 1	0.72	0.71	0.53	0.78	0.58
Supplier 2	0.64	0.70	0.64	0.61	0.64
Supplier 3	0.52	0.74	0.72	0.75	0.57
Supplier 4	0.65	0.65	0.85	0.72	0.48
Supplier 5	0.43	0.45	0.67	0.87	0.63

Table 5.2: The Number of Items Produced Per Hour

Suppliers	Item 1	Item 2	Item 3	Item 4	Item 5
Supplier 1	1	2	1	3	5
Supplier 2	1	1.5	2	2	4

Suppliers	Item 1	Item 2	Item 3	Item 4	Item 5
Supplier 3	1	3	2	3	5
Supplier 4	0.5	2	1	2	4
Supplier 5	1	2	1.5	2	4

Table 5.3: The Matching Relationship Between Items

Item	Matching Relationship	
Supplier 1 Item 1	Supplier 2 Item 3	Supplier 3 Item 3
Supplier 2 Item 3	Supplier 1 Item 1	Supplier 3 Item 3
Supplier 3 Item 3	Supplier 1 Item 1	Supplier 2 Item 3

This is undoubtedly a challenge for Eddie. We propose a new way for him to develop an optimal procurement solution. Prior to making any decisions, we need to evaluate the average cost of purchasing each item, the average carbon emissions generated during the production and transportation of each item, and the average quality of each item as shown in Table 5.1. The carbon emission can be found using the life-cycle assessment (LCA) method and calculated based on different carbon emission factors. Then, the three indicators are normalized to a value between 0 and 1, and the quality of each item is converted to a value between 0 and 1, where a smaller value means a better case, and vice versa. Next, based on these three types of indicators, we calculate the overall evaluation value of each supplier for each type of item, and since these three factors are equally important, they have the same weight of one-third in the overall evaluation value. It is also necessary to calculate the number of working hours, the production capacity as shown in Table 5.2 and the minimum purchasable quantity that the supplier can provide for the different items. In addition, the fit between items produced by different suppliers should be taken into account, as shown in Table 5.3, item A only fits items B and C. If item A is purchased, it can only be chosen from C and D of the supplier. Finally, the constraints proposed by the company are added to find a purchasing solution with the minimum evaluation value. The final optimized allocation qualification value of 2500.

5.2 Procurement Strategy for Reducing Carbon Emissions

First, we need to collect all the costs involved in purchasing different types of items from different suppliers and calculate the average cost of purchasing each item. Then we need to evaluate the carbon footprint of each item purchased, and we may calculate all the direct and

indirect carbon emissions produced by the item based on different carbon emission factors. Finally, we also need to assess the average product quality of each item. The three types of data collected were aggregated and normalized to a value between 0 and 1. In addition, item quality necessitates further the conversion of the order of the maximum and minimum values, and after subtracting the item quality score from 1, smaller values indicate higher item quality scores. This conversion is to facilitate the subsequent identification of the least cost and carbon emission procurement options. After obtaining these three types of scores, we can finally calculate the qualification value of different items from different suppliers, which can be obtained by multiplying these three scores by their respective weights cumulatively. AHP can effectively help us to derive their weights analytically [17].

Before solving, we also need to determine a set of conditions that may be involved in a realistic scenario. Since the procurement strategy is modeled in terms of working hours in this solution, we also need to account for the efficiency of each supplier to produce different items, i.e., the production per hour. The product of the production efficiency and the working hours assigned to the supplier by the purchaser cannot be less than the sourcing requirement. In addition, it is doubtful that the supplier will produce only a small number of items at a time in a practical situation, so we also need to consider the minimum purchasable quantity. Finally, other constraints, such as procurement budget, carbon emission cap and product quality standard, are taken into account to find the optimal working hours allocation solution, i.e., the optimal procurement solution to meet the procurement demand.

5.3 Formalization of the Procurement Problem with E-CARGO Model

In this section we still use the E-CARGO model to analyze the problem, and the following notation and definitions are similar but not related to the previous section. In the new problem A refers to the suppliers and R refers to the items to be purchased. The number of suppliers that can be expressed by the non-negative integer $m\ (= |A|)$, likewise, $n\ (= |R|)$ is the number of items that we can express, $i, i_1, i_2, \ldots$ are the indices of suppliers, and $j, j_1, j_2, \ldots$ are the indices of items.

Definition 5.1 [24]: The role range vector L is an n-vector that contains the lower bounds of the ranges of roles in the environment of a group. It indicates the quantity of each item to be

purchased, i.e., $L[j] \geq 1 (0 \leq j < n)$. If the quantity to be purchased for item 1 is 10,000 then $L[j_1] = 10{,}000$.

Definition 5.2 [24]: The role limit matrix L^a is an $m \times n$ matrix where $L^a[i, j] \in \mathbb{N}$ represents the maximum number of working hours that a supplier can provide to produce an item. It is also related to the production conditions of the supplier. For example, if item 1 and item 2 are produced using separate production lines, the working hours between them will not affect each other.

Definition 5.3: The Production Line Relationship PL is an $m \times n$ matrix that is used to record the production line relationships used by different suppliers to produce different items. $PL[i_1][\, j_1, j_2, j_3, j_4]$ indicates that supplier 1 produces items 1,2,3, and 4 by the same production line, therefore the working hours provided by supplier 1 are shared by these items.

Definition 5.4: The procurement spend matrix PS is an $m \times n$ matrix, where $PS[i, j] \in \mathbb{N}$ represents the cost of purchasing item j $(0 \leq j < n)$ from a single supplier i $(0 \leq i < m)$. Normalize procurement expenditures to get procurement cost score $PS' \in [0, 1]$.

Definition 5.5: The carbon emission matrix CE is an $m \times n$ matrix, where $CE[i, j] \in \mathbb{N}$ represents all the direct and indirect carbon emissions involved in purchasing item j $(0 \leq j < n)$ from individual supplier i $(0 \leq i < m)$. To ease formalization, we normalize carbon emissions to create carbon emission scores $CE' \in [0, 1]$.

Definition 5.6: The quality score matrix QS is an $m \times n$ matrix, where $QS[i, j] \in [0, 1]$ is used to measure the average quality of item j $(0 \leq j < n)$ of supplier i $(0 \leq i < m)$. For the subsequent operation of finding the minimum solution, we use $QS' \in [0, 1]$, $QS'[i, j] = 1 - QS[i, j] (0 \leq i < m, 0 \leq j < n)$. Then the minimum value is used to represent the best quality score.

Definition 5.7 [24]: The qualification matrix Q is an m × n matrix, where $Q[i, j] \in [0, 1]$ represents the qualification value of item j $(0 \leq j < n)$ of supplier i $(0 \leq i < m)$. $Q[i, j] = w_1 PS'[i, j] + w_2 CE'[i, j] + w_3 QS'[i, j]$, where $w_1, w_2, w_3 \in [0, 1]$ are the weights of the three indicators in the qualification value.

To determine the weight of each factor in the qualification value in a concise and systematic way, we use AHP. First the three factors can be evaluated by experts to determine the more appropriate scale between them two by two, and in the example we used the 1-9 scale method. Then we calculate the feature vector value based on the scale, which is the weight value of each

factor, and also obtain the maximum feature root value and consistency index for the consistency test later.

Definition 5.8: The work efficiency matrix WE is an $m \times n$ matrix, where $WE[i,j] \in \mathbb{N}(0 \leq i < m, 0 \leq j < n)$ indicates the number of item j that can be produced by supplier i per hour.

Definition 5.9: The minimum number of working hours available from the suppliers WH is an $m \times n$ matrix, where $WH[i,j] \in \mathbb{N}$ indicates the minimum number of working hours that the purchaser can allocate to the supplier when purchasing item $j(0 \leq j < n)$ produced by supplier $i(0 \leq i < m)$. The minimum available working hours are introduced to match the actual situation of the supplier and to avoid situations where the purchaser requests an excessively low procurement requirement or an unavailable quantity.

Definition 5.10: The carbon emissions cap $CEC \in \mathbb{N}$ indicates the maximum carbon emissions allowed by the procurement scheme.

Definition 5.11: The procurement budget $PB \in \mathbb{N}$ indicates the maximum procurement expenditure that the purchaser can accept.

Definition 5.12: The item quality level requirement QL is an n-vector where $QL[j] \in [0,1]$ represents the minimum quality standard that needs to be met for each type of item.

Definition 5.13: The item matching relationship matrix MR is an $m \times n \times m \times n$ matrix, where $MR[i,j,i',j'] \in \{0,1\}\ (0 \leq i < m, 0 \leq j < n)$ indicates whether there is a matching relationship between item j of supplier i and item j' of supplier i'. That $MR[i_1,j_1,i_1,j_3] = 1$ indicates there is an adaptation relationship between item 1 and item 3 of supplier 1. Hence, if you choose to purchase item 1 of supplier 1 during procurement, you also need to choose item 3 of supplier 1.

Definition 5.14 [24]: The definition of the role assignment matrix T is an $m \times n$ matrix, where $T[i,j] \in [0, \max L^a[i,j]]\ (0 \leq i < m, 0 \leq j < n)$ indicates the purchaser's demand for working hours to produce item j at supplier i. That $T[i,j] = 100$ indicates the purchaser requires 100 hours of production item j from supplier i.

Definition 5.15 [24]: The qualities of the allocated agents are added together to determine the group performance σ of group g, which is $\sigma = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} Q[i,j] \times T[i,j] \times WE[i,j]$. It indicates the final score of the procurement solution.

Definition 5.16 [24]: If role j has been given a sufficient number of agents, it is workable in the group, i.e., $\sum_{i=0}^{m-1} T[i,j] \times WE[i,j] \geq L[j]$. It means that the sum of the product of working hours and work efficiency required by the purchaser for different suppliers needs to be greater than or equal to the quantity purchased.

Definition 5.17 [24]: If each role in T is workable, i.e., $\forall(0 \leq j < n) \sum_{i=0}^{m-1} T[i,j] \times WE[i,j] \geq L[j]$, then T is workable. If T is workable, then g is as well.

Definition 5.18: Using the definitions listed above, our allocation problem can be defined to determine a matrix T to obtain

$$\sigma_0 = min\left\{\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} Q[i,j] \times T[i,j] \times WE[i,j]\right\}$$

$$\text{subject to } T[i,j] \in [0, \max L^a[i,j]] \ (0 \leq i < m, 0 \leq j < n) \quad (5.1)$$

$$\sum_{i=0}^{m-1} T[i,j] \times WE[i,j] \geq L[j] \ (0 \leq j < n) \quad (5.2)$$

$$\sum_{i=0}^{n(PL[i])-1} T[i,j] \leq L^a[i][j] \ (0 \leq i < m) \quad (5.3)$$

$$\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} T[i,j] \times WE[i,j] \times PS[i,j] \leq PB \ (0 \leq i < m, \ 0 \leq j < n) \quad (5.4)$$

$$\sum_{i=0}^{m-1}\sum_{j=0}^{n-1} T[i,j] \times WE[i,j] \times CE[i,j] \leq CEC \ (0 \leq i < m, \ 0 \leq j < n) \quad (5.5)$$

$$QS[i,j] \geq QL[j] \ (0 \leq i < m, \ 0 \leq j < n) \quad (5.6)$$

$$if \ T[i,j] > 0 \rightarrow T[i,j] \geq WH[i,j] \ (0 \leq i < m, 0 \leq j < n) \quad (5.7)$$

$$if MR[i,j,i',j'] > 0 \rightarrow \sum_{i'=0}^{m'-1} T[i',j'] \times WE[i',j'] \geq L[j'] (j \neq j', 0 \leq i < m, 0 \leq j < n, 0 \leq i' < m, 0 \leq j' < n) \quad (5.8)$$

Where constraint (5.1) informs whether the purchaser has a demand for the supplier's item in terms of working hours; (5.2) makes the group workable; (5.3) indicates that the purchaser's demand for working hours cannot exceed the maximum number of working hours that the supplier can provide under certain production conditions, $n(PL[i])$ tells the number of items produced by different production lines of supplier i; (5.4) points out that all expenses for procurement must not exceed the procurement budget; (5.5) constraints that the sum of all carbon emissions involved in the procurement must not exceed the carbon emissions cap; (5.6)

specifies that the quality of each type of item cannot be lower than the item quality standard; (5.7) shows that the required working hours in the procurement program must be greater than or equal to the minimum productive working hours set by the supplier; (5.8) denotes that if there is a matching relationship between two types of items, e.g., if there is a matching relationship between item 1 and item 3, then if item 1 from that supplier is purchased, the purchase of item 3 must be selected from among the suppliers with which it has a matching relationship, m' indicates the number of suppliers with which it has a matching relationship.

5.4 Summary

In this chapter, we extend the application of the allocation scheme to focus on the procurement segment of the supply chain, focusing on reducing the carbon emissions involved in that segment. First, we present a realistic scenario to describe the problem and outline how to achieve an optimization of the sourcing solution. Finally, similar to the previous method, we model the problem using the E-CARGO model and adapt and introduce definitions to fit the problem.

Chapter 6

6 Simulation Experiments to Optimize Procurement Solution

In this section, we simulate different real-world scenarios and design three types of experiments to analyze the effect of the solution. All experiments were conducted in the environment shown in Table 3.5.

Based on the modeling of the problem, we know that it is challenging to solve such a complicated model, and we need to discover a way that can do it in a reasonable amount of time. ILOG is the tool that can be utilized to tackle this complex problem. In order to resolve the issue, we employ the IBM ILOG CPLEX (CPLEX) package. First, we must prepare all the components to meet the requirement of the CPLEX package, where the variables of matrix T have upper and lower bounds of 0 and the maximum value of L^a, respectively, and the matrix Q and WE represent the coefficients of the objective function. Subsequently, we add the expressions of the objective and constraints and find the optimal solution, where these expressions can be represented by some formulas.

6.1 Performance Experiments

We simulate experiments in contexts with different numbers of suppliers and purchased items, and then we track the execution time of the program to evaluate the performance of the solution in real-world scenarios. The size of the simulated experiment is assumed to start with a number of procurement items of 50 and increase by 50 procurement items each step until it approaches 300. Correspondingly, the number of suppliers is one-fifth of the number of item types.

The item quality standard we set to 0.8, which means that items with a quality score lower than 0.8 will not be considered. The upper limit of the procurement budget and the upper limit of the carbon emission are set to 80% of the maximum values of the procurement budget and carbon emission. The production line relationships are randomly generated and the probability of the existence of matching relationships between items is set to 10%.

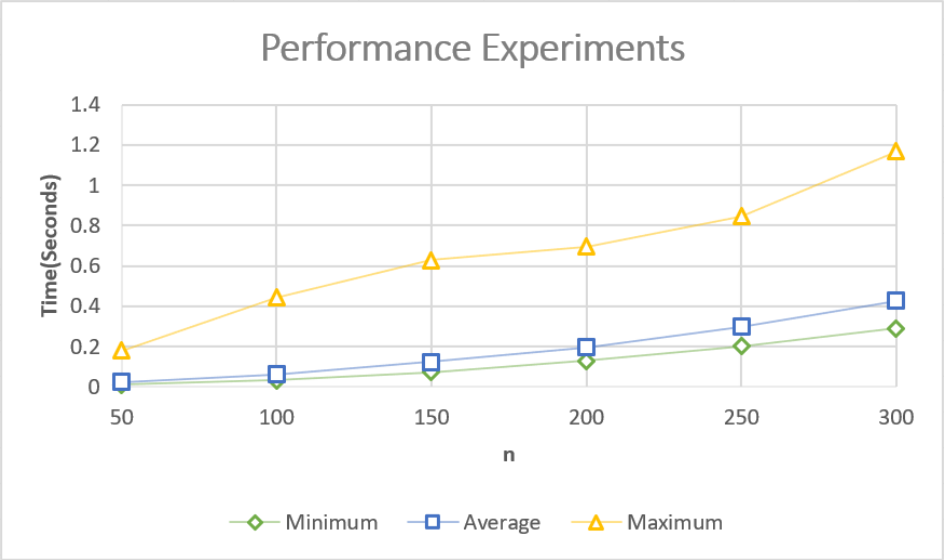

Figure 6.1: Performance Experiments Result.

We conduct 100 sets of experiments with different numbers of suppliers and types of purchased items, counting the minimum, average and maximum running times in each case. The results are displayed in Fig. 6.1. According to the results, when there are 10 suppliers and 50 categories of items to be purchased, the minimum running time is 0.01 s, the average running time is approximately 0.03 s and the maximum running time is 0.18 s. When there are 60 suppliers and 300 categories of items to be purchased, the minimum running time increase to 0.29 s, the average time increase to 0.43 s and the maximum time increase to 1.17 s. The results of the performance experiments show that our proposed method runs efficiently at a certain scale and the running time is relatively acceptable when the scale is larger. Also to further determine the performance efficiency of the method, we modify different constraints in the method and test them. The results show that the constraints of the item matching relationship require more computation time, so we need to further improve the constraints if we want to further improve the efficiency.

6.2 Weighting Ratio Experiments

In order to evaluate the impact of the weight of the carbon emission score on the total sum qualification value, we assume different weights for the carbon emission score and conducted 100 experiments separately. According to AHP, we assume five importance levels among the factors: very weak, weak, strong, very strong, and absolution with other factors, and their

measurement values are 1, 3, 5, 7, and 9, respectively. Subsequently, we calculate the cumulative total scores of procurement expenditure, carbon emission and item quality based on the results of the working hour allocation in the procurement solution and use the average of 100 experimental results as the final reference value. The experiments are conducted at the scale of $n = 100$, $m = 20$, and the other conditions are similar to the previous experiments.

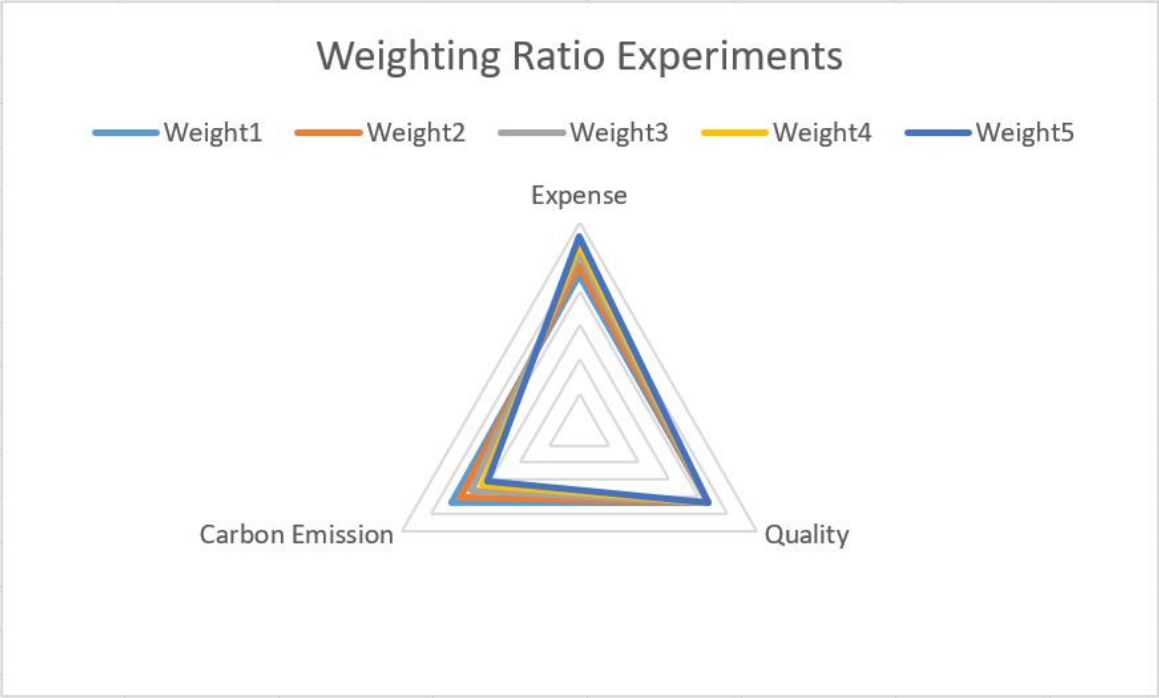

Figure 6.2: Weighting Ratio Experiments Result.

The result of the experiments are shown in Fig. 6.2, where different colors indicate different weights, and the weights from 1 to 5 correspond to the weight of carbon emissions with other factors from very weak to absolution, respectively. When the weight of carbon emissions is absolution, the score of carbon emissions decreases by 28.75% compared to when the weight of carbon emissions and other factors are equally strong, while the procurement expense increases by 23.74%. It can be seen that as the weight of carbon emissions increases, the reduction in carbon emissions becomes more pronounced, although there is also a significant increase in the relative cost of procurement, while at the same time, the quality of the items does not change significantly and is in a state of small fluctuations of increase and decrease. Therefore, in real-world situations, especially when the purchaser is more concerned about carbon emissions, we

should not ignore the cost of procurement and try to adjust the weight of various factors to a reasonable range.

6.3 Comparison Experiments

In the comparison experiments, we compare a baseline method with three methods - minimizing procurement expense, minimizing carbon emissions, and maximizing item quality - to analyze their strengths and weaknesses. The baseline method treats all three factors as equally important, meaning they all have a weight of one-third in the calculation of eligibility values, and then uses that weight to calculate the optimal procurement solution. Minimizing expenditure is to use only the procurement expenditure as an optimization parameter and find the working hour allocation scheme with the lowest expenditure. Minimizing carbon emissions and maximizing item qualities are similar to minimizing expenditure, and finally a procurement scheme is found under certain constraints. The assumptions of the experiment are similar to those of the previous experiments. The results of the experiments are shown in Fig. 6.3 and Fig. 6.4.

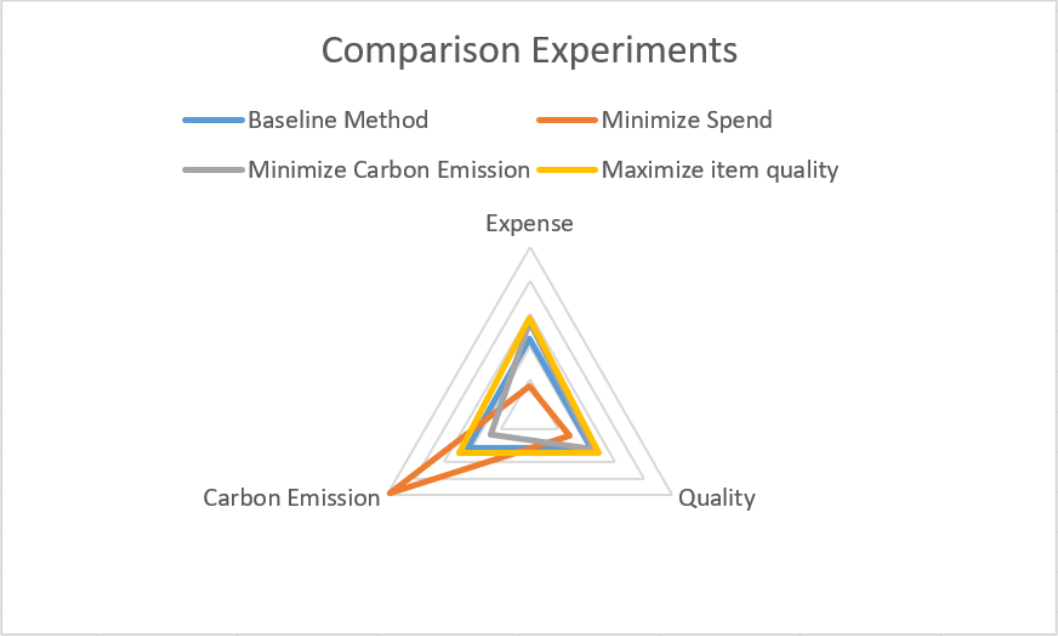

Figure 6.3: Comparison Experiments Result.

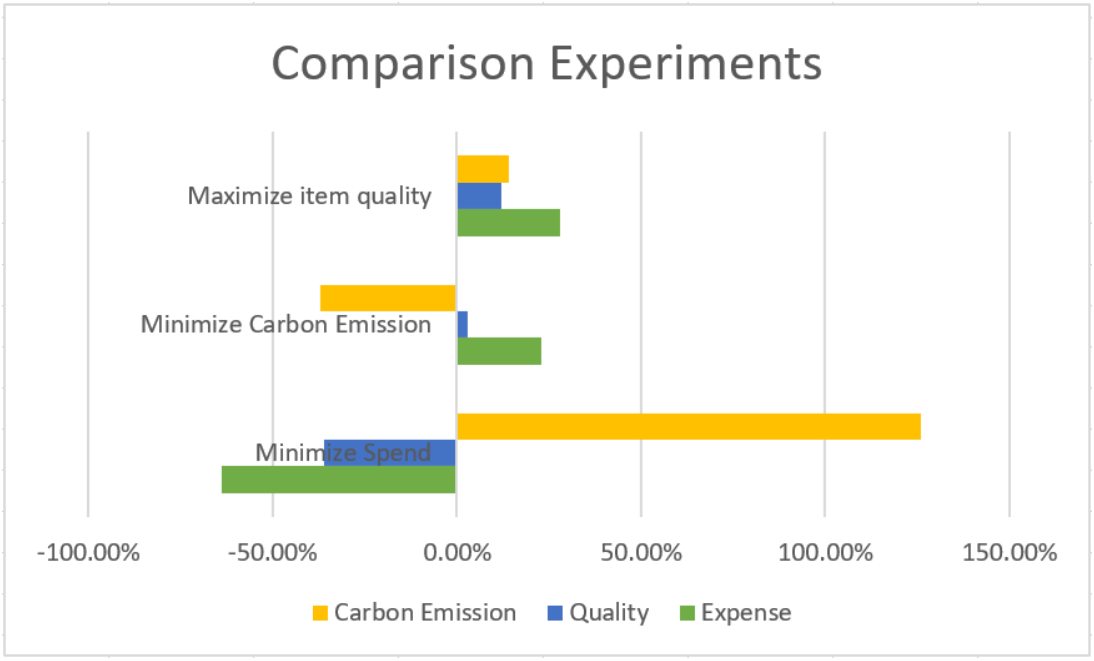

Figure 6.4: Comparison Experiments Result.

Fig. 6.3 shows the differences between the three methods compared to the basic method in terms of three factors: expenditure, carbon emissions and quality. The figure shows that the method that focuses only on procurement spend performs best in terms of overhead but has the highest carbon emission and the lowest average quality of items. The approach that focuses only on carbon emissions significantly reduces carbon emissions compared to the other methods but also increases costs. The method that only considers the quality of the items increases in all aspects compared to the basic solution. Fig. 6.4 shows a more specific percentage increase or decrease for the three methods compared to the baseline method.

6.4 Summary

In this chapter, we have conducted performance experiments, weight ratio experiments and comparison experiments. Based on the experimental results, we suggest that companies need to consider the impact of various factors in a more aggregate manner when purchasing items, and that over-consideration of one factor may have a more serious impact on the results. When developing a purchasing plan, we try to adjust the weights of different factors to find a relatively balanced optimal purchasing plan, taking into account the results of previous experiments. We must clarify that the constraints of work efficiency, working hours, and procurement

requirements in the experiments are hypothetical data, and further research may be needed to confirm the generality of the experimental results for specific areas or situations.

Chapter 7

7 Conclusion

In this paper, we propose an optimal allocation solution for decentralized data storage in an edge-cloud environment that focuses on enhancing data privacy. We propose a data partitioning scheme by field to avoid as much as possible the inference of important information due to data leakage. We introduce a degree of fit metric that enables data fields to be stored in the most appropriate servers. We also introduce the E-CARGO model and other constraints to model the problem and use the CPLEX optimization package to find the optimal allocation scheme. We refine the method to improve its performance and reduce the running time to an acceptable range based on realistic scenarios. According to the results of the simulation experiments, our proposed solution allows the data to be stored in servers that better match their requirements compared with the trust-based and baseline methods, making full use of the server resources while reducing the overhead to some extent. We further optimize the method by proposing some strategies to adapt it to scenarios with a large number of servers and data volume and data backup.

Furthermore, we extend the application of the approach to other areas, and we propose a working hour allocation scheme for item procurement that focuses on optimizing carbon emissions. In dealing with this proposed scheme, we facilitate the modeling and analysis of the problem using the E-CARGO model. Following the E-CARGO model, we introduce a comprehensive evaluation score to assess the advantages and disadvantages of the procurement scheme in terms of procurement overhead, carbon emissions and quality of the items. We also introduce other constraints to model the problem and use the CPLEX optimization package to find the best allocation scheme. Based on the results of the simulation experiments, our proposed solution enables the purchaser to develop a procurement solution that better meets their needs, reducing the carbon emissions of the procurement process while reducing the overhead to a certain extent while ensuring the quality of the purchased items.

Our next study will focus on further examining the partitioning of data fields for large-scale scenarios, data field conflicts, and the introduction of other constraints; further improving the

performance of the method; adjusting the adaptability of the method to fit more scenarios and constraints; and designing the system prototype.

References

[1] A. Acquaye, A. Genovese, J. Barrett, and S. C. Lenny Koh, ‘Benchmarking carbon emissions performance in supply chains’, Supply Chain Management: An International Journal, vol. 19, no. 3, pp. 306–321, May 2014, doi: 10.1108/SCM-11-2013-0419.

[2] A. Kumar, V. Jain, and S. Kumar, ‘A comprehensive environment friendly approach for supplier selection’, Omega, vol. 42, no. 1, pp. 109–123, Jan. 2014, doi: 10.1016/j.omega.2013.04.003.

[3] A. Makris, E. Psomakelis, T. Theodoropoulos, and K. Tserpes, ‘Towards a Distributed Storage Framework for Edge Computing Infrastructures’, in Proceedings of the 2nd Workshop on Flexible Resource and Application Management on the Edge, Minneapolis MN USA, Jul. 2022, pp. 9–14. doi: 10.1145/3526059.3533617.

[4] A. Sharma and H. Banati, ‘A Framework for Implementing Trust in Cloud Computing’, in Proceedings of the International Conference on Internet of things and Cloud Computing, Cambridge United Kingdom, Mar. 2016, pp. 1–7. doi: 10.1145/2896387.2896391.

[5] A. Singh, S. Kumari, H. Malekpoor, and N. Mishra, ‘Big data cloud computing framework for low carbon supplier selection in the beef supply chain’, Journal of Cleaner Production, vol. 202, pp. 139–149, Nov. 2018, doi: 10.1016/j.jclepro.2018.07.236.

[6] B. A. Julstrom, ‘Greedy, genetic, and greedy genetic algorithms for the quadratic knapsack problem’, in Proceedings of the 2005 conference on Genetic and evolutionary computation - GECCO ’05, Washington DC, USA, 2005, p. 607. doi: 10.1145/1068009.1068111.

[7] C. Jia, K. Lin, and J. Deng, ‘A Multi-property Method to Evaluate Trust of Edge Computing Based on Data Driven Capsule Network’, in IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Toronto, ON, Canada, Jul. 2020, pp. 616–621. doi: 10.1109/INFOCOMWKSHPS50562.2020.9163069.

[8] C. Lang, C. Woo, and J. Sinclair, ‘Quantifying data sensitivity: precise demonstration of care when building student prediction models’, in Proceedings of the Tenth International Conference on Learning Analytics & Knowledge, Frankfurt Germany, Mar. 2020, pp. 655–664. doi: 10.1145/3375462.3375506.

[9] C. Lang, C. Woo, and J. Sinclair. 2020. Quantifying data sensitivity: precise demonstration of care when building student prediction models. In Proceedings of the Tenth International Conference on Learning Analytics & Knowledge (LAK '20). Association for Computing Machinery, New York, NY, USA, 655–664. https://doi.org/10.1145/3375462.3375506

[10] C. Niu, Z. Zheng, F. Wu, X. Gao, and G. Chen, ‘Achieving Data Truthfulness and Privacy Preservation in Data Markets’, IEEE Trans. Knowl. Data Eng., vol. 31, no. 1, pp. 105–119, Jan. 2019, doi: 10.1109/TKDE.2018.2822727.

[11] C. Zhang, E.-C. Chang, and R. H. C. Yap, ‘Tagged-MapReduce: A General Framework for Secure Computing with Mixed-Sensitivity Data on Hybrid Clouds’, in 2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Chicago, IL, USA, May 2014, pp. 31–40. doi: 10.1109/CCGrid.2014.96.

[12] C.-T. Chang, C.-L. Chin, and M.-F. Lin, ‘On the single item multi-supplier system with variable lead-time, price-quantity discount, and resource constraints’, Applied Mathematics and Computation, vol. 182, no. 1, pp. 89–97, Nov. 2006, doi: 10.1016/j.amc.2006.03.038.

[13] D.-I. Sirbu, C. Negru, F. Pop, and C. Esposito, ‘Storage service for edge computing’, in Proceedings of the 36th Annual ACM Symposium on Applied Computing, Virtual Event Republic of Korea, Mar. 2021, pp. 1165–1171. doi: 10.1145/3412841.3441991.

[14] E. D. Gemechu, I. Butnar, M. Llop, and F. Castells, ‘Environmental tax on products and services based on their carbon footprint: A case study of the pulp and paper sector’, Energy Policy, vol. 50, pp. 336–344, Nov. 2012, doi: 10.1016/j.enpol.2012.07.028.

[15] E. Pourjavad and R. V. Mayorga, ‘Optimization of a sustainable closed loop supply chain network design under uncertainty using multi-objective evolutionary algorithms’, Adv produc engineer manag, vol. 13, no. 2, pp. 216–228, Jun. 2018, doi: 10.14743/apem2018.2.286.

[16] F. T. S. Chan and H. K. Chan, ‘An AHP model for selection of suppliers in the fast changing fashion market’, Int J Adv Manuf Technol, vol. 51, no. 9–12, pp. 1195–1207, Dec. 2010, doi: 10.1007/s00170-010-2683-6.

[17] F. T. S. Chan, N. Kumar, M. K. Tiwari, H. C. W. Lau, and K. L. Choy, ‘Global supplier selection: a fuzzy-AHP approach’, International Journal of Production Research, vol. 46, no. 14, pp. 3825–3857, Jul. 2008, doi: 10.1080/00207540600787200.

[18] G. D. Samaraweera and J. M. Chang, ‘Security and Privacy Implications on Database Systems in Big Data Era: A Survey’, IEEE Trans. Knowl. Data Eng., vol. 33, no. 1, pp. 239–258, Jan. 2021, doi: 10.1109/TKDE.2019.2929794.

[19] Gupta A, Spillane R, Wang W, et al. Hybrid cloud storage: Bridging the gap between compute clusters and cloud storage[J]. ACM SIGOPS Operating Systems Review, 2017, 51(1): 48-53.

[20] H. A. Idar, K. Aissaoui, H. Belhadaoui and R. F. Hilali, "Dynamic Data Sensitivity Access Control in Hadoop Platform," 2018 IEEE 5th International Congress on Information Science and Technology (CiSt), Marrakech, Morocco, 2018, pp. 105-109, doi: 10.1109/CIST.2018.8596381.

[21] H. Halabian, I. Lambadaris and Y. Viniotis, "Optimal server assignment in multi-server queueing systems with random connectivities," in Journal of Communications and Networks, vol. 21, no. 4, pp. 405-415, Aug. 2019, doi: 10.1109/JCN.2019.000023.

[22] H. Zhu, ‘Avoiding Conflicts by Group Role Assignment’, IEEE Trans. Syst. Man Cybern, Syst., vol. 46, no. 4, pp. 535–547, Apr. 2016, doi: 10.1109/TSMC.2015.2438690.

[23] H. Zhu, ‘Maximizing Group Performance While Minimizing Budget’, IEEE Trans. Syst. Man Cybern, Syst., vol. 50, no. 2, pp. 633–645, Feb. 2020, doi: 10.1109/TSMC.2017.2735300.

[24] H. Zhu, D. Liu, S. Zhang, S. Teng, and Y. Zhu, ‘Solving the Group Multirole Assignment Problem by Improving the ILOG Approach’, IEEE Trans. Syst. Man Cybern, Syst., vol. 47, no. 12, pp. 3418–3424, Dec. 2017, doi: 10.1109/TSMC.2016.2566680.

[25] H. Zhu, E-CARGO and role-based collaboration: modeling and solving problems in the complex world, First edition. Hoboken, New Jersey: John Wiley & Sons, Inc, 2021.

[26] H. Zhu, M. Zhou, and R. Alkins, 'Group Role Assignment via a Kuhn–Munkres Algorithm-Based Solution', IEEE Trans. Syst., Man, Cybern. A, vol. 42, no. 3, pp. 739–750, May 2012, doi: 10.1109/TSMCA.2011.2170414.

[27] H. Zhu, Y. Sheng, X. Zhou and Y. Zhu, "Group Role Assignment With Cooperation and Conflict Factors," in IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 48, no. 6, pp. 851-863, June 2018, doi: 10.1109/TSMC.2016.2633244.

[28] H. Zhuang, R. Rahman, P. Hui, and K. Aberer, 'Optimizing Information Leakage in Multicloud Storage Services', IEEE Trans. Cloud Comput., vol. 8, no. 4, pp. 975–988, Oct. 2020, doi: 10.1109/TCC.2018.2808275.

[29] IBM, 'Cost of a Data Breach Report', Computer Fraud & Security, vol. 2021, no. 8, pp. 4–4, Jan. 2021, doi: 10.1016/S1361-3723(21)00082-8.

[30] J. E. Rowe, 'Genetic algorithm theory', in Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference companion - GECCO Companion '12, Philadelphia, Pennsylvania, USA, 2012, p. 917. doi: 10.1145/2330784.2330923.

[31] J. Hur, 'Improving Security and Efficiency in Attribute-Based Data Sharing', IEEE Trans. Knowl. Data Eng., vol. 25, no. 10, pp. 2271–2282, Oct. 2013, doi: 10.1109/TKDE.2011.78.

[32] J. Koo, G. Kang, and Y.-G. Kim, 'Security and Privacy in Big Data Life Cycle: A Survey and Open Challenges', Sustainability, vol. 12, no. 24, p. 10571, Dec. 2020, doi: 10.3390/su122410571.

[33] J. Yang, H. Zhu, and T. Liu, 'Secure and economical multi-cloud storage policy with NSGA-II-C', Applied Soft Computing, vol. 83, p. 105649, Oct. 2019, doi: 10.1016/j.asoc.2019.105649.

[34] J. Zhang, T. Li, Z. Ying and J. Ma, "Trust-Based Secure Multi-Cloud Collaboration Framework in Cloud-Fog-Assisted IoT," in IEEE Transactions on Cloud Computing, doi: 10.1109/TCC.2022.3147226.

[35] K. Al Nuaimi, N. Mohamed, M. Al Nuaimi and J. Al-Jaroodi, "ssCloud: A Smart Storage for Distributed DaaS on the Cloud," 2015 IEEE 8th International Conference on Cloud Computing, New York, NY, USA, 2015, pp. 1049-1052, doi: 10.1109/CLOUD.2015.149.

[36] K. Lamba and S. P. Singh, 'Dynamic supplier selection and lot-sizing problem considering carbon emissions in a big data environment', Technological Forecasting and Social Change, vol. 144, pp. 573–584, Jul. 2019, doi: 10.1016/j.techfore.2018.03.020.

[37] K. Shaw, R. Shankar, S. S. Yadav, and L. S. Thakur, 'Supplier selection using fuzzy AHP and fuzzy multi-objective linear programming for developing low carbon supply chain', Expert Systems with Applications, vol. 39, no. 9, pp. 8182–8192, Jul. 2012, doi: 10.1016/j.eswa.2012.01.149.

[38] L. X. Cui, L. Bai and Z. P. Cui, "Optimal multi-period multi-product supplier selection and order allocation: Balancing supplier development and supplier switching," 2017 IEEE

International Conference on Industrial Engineering and Engineering Management (IEEM), Singapore, 2017, pp. 1985-1989, doi: 10.1109/IEEM.2017.8290239.

[39] M. Gong, K. Pan, and Y. Xie, ‘Differential privacy preservation in regression analysis based on relevance’, Knowledge-Based Systems, vol. 173, pp. 140–149, Jun. 2019, doi: 10.1016/j.knosys.2019.02.028.

[40] M. Nitti, R. Girau, and L. Atzori, ‘Trustworthiness Management in the Social Internet of Things’, IEEE Trans. Knowl. Data Eng., vol. 26, no. 5, pp. 1253–1266, May 2014, doi: 10.1109/TKDE.2013.105.

[41] M. S. Asif, H. Lau, D. Nakandala, Y. Fan, and H. Hurriyet, ‘Case study research of green life cycle model for the evaluation and reduction of scope 3 emissions in food supply chains’, Corp Soc Responsibility Env, vol. 29, no. 4, pp. 1050–1066, Jul. 2022, doi: 10.1002/csr.2253.

[42] M. Thangavel and P. Varalakshmi, ‘Enabling Ternary Hash Tree Based Integrity Verification for Secure Cloud Data Storage’, IEEE Trans. Knowl. Data Eng., vol. 32, no. 12, pp. 2351–2362, Dec. 2020, doi: 10.1109/TKDE.2019.2922357.

[43] M. V. Carvalho Fagundes, Á. Cheles Keler, E. Oliveira Teles, S. A. B. Vieira de Melo and F. G. Mendonça Freires, "Multicriteria Decision-Making System for Supplier Selection Considering Risk: A Computational Fuzzy AHP-Based Approach," in IEEE Latin America Transactions, vol. 19, no. 9, pp. 1564-1572, Sept. 2021, doi: 10.1109/TLA.2021.9468610.

[44] N. Absi, S. Dauzère-Pérès, S. Kedad-Sidhoum, B. Penz, and C. Rapine, ‘Lot sizing with carbon emission constraints’, European Journal of Operational Research, vol. 227, no. 1, pp. 55–61, May 2013, doi: 10.1016/j.ejor.2012.11.044.

[45] P. K. Fong and J. H. Weber-Jahnke, ‘Privacy Preserving Decision Tree Learning Using Unrealized Data Sets’, IEEE Trans. Knowl. Data Eng., vol. 24, no. 2, pp. 353–364, Feb. 2012, doi: 10.1109/TKDE.2010.226.

[46] R. Karim, C. Ding and A. Miri, "An End-to-End QoS Mapping Approach for Cloud Service Selection," 2013 IEEE Ninth World Congress on Services, 2013, pp. 341-348, doi: 10.1109/SERVICES.2013.71.

[47] R. Patil Rashmi, Y. Gandhi, V. Sarmalkar, P. Pund and V. Khetani, "RDPC: Secure Cloud Storage with Deduplication Technique," 2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, India, 2020, pp. 1280-1283, doi: 10.1109/I-SMAC49090.2020.9243442.

[48] S. Adams and A. O. Acheampong, ‘Reducing carbon emissions: The role of renewable energy and democracy’, Journal of Cleaner Production, vol. 240, p. 118245, Dec. 2019, doi: 10.1016/j.jclepro.2019.118245.

[49] S. De Capitani di Vimercati, S. Foresti, G. Livraga, V. Piuri, and P. Samarati, ‘Security-Aware Data Allocation in Multicloud Scenarios’, IEEE Trans. Dependable and Secure Comput., pp. 1–1, 2021, doi: 10.1109/TDSC.2019.2953068.

[50] S. De Capitani Di Vimercati, S. Foresti, G. Livraga, V. Piuri, and P. Samarati, ‘Supporting User Requirements and Preferences in Cloud Plan Selection’, IEEE Trans. Serv. Comput., pp. 1–1, 2019, doi: 10.1109/TSC.2017.2777977.

[51] S. Elhedhli and R. Merrick, ‘Green supply chain network design to reduce carbon emissions’, Transportation Research Part D: Transport and Environment, vol. 17, no. 5, pp. 370–379, Jul. 2012, doi: 10.1016/j.trd.2012.02.002.

[52] S. Xia, Y. Zhang and F. Yang, "Green evaluation design and application research of power grid suppliers under the carbon peaking and carbon neutrality goals," 2022 Asian Conference on Frontiers of Power and Energy (ACFPE), Chengdu, China, 2022, pp. 308-312, doi: 10.1109/ACFPE56003.2022.9952353.

[53] T. D. Dang, D. Hoang, and D. N. Nguyen, ‘Trust-Based Scheduling Framework for Big Data Processing with MapReduce’, IEEE Trans. Serv. Comput., vol. 15, no. 1, pp. 279–293, Jan. 2022, doi: 10.1109/TSC.2019.2938959.

[54] T. Sawa, F. He, A. Kawabata and E. Oki, "Polynomial-time Algorithm for Distributed Server Allocation Problem," 2019 IEEE 8th International Conference on Cloud Networking (CloudNet), Coimbra, Portugal, 2019, pp. 1-3, doi: 10.1109/CloudNet47604.2019.9064128.

[55] T. Zouadi, A. Yalaoui, M. Reghioui, and K. E. El Kadiri, ‘Hybrid manufacturing/remanufacturing lot-sizing problem with returns supplier’s selection under, carbon emissions constraint’, IFAC-PapersOnLine, vol. 49, no. 12, pp. 1773–1778, 2016, doi: 10.1016/j.ifacol.2016.07.839.

[56] T.-M. Choi, ‘Optimal apparel supplier selection with forecast updates under carbon emission taxation scheme’, Computers & Operations Research, vol. 40, no. 11, pp. 2646–2655, Nov. 2013, doi: 10.1016/j.cor.2013.04.017.

[57] V. Kantere, D. Dash, G. Francois, S. Kyriakopoulou, and A. Ailamaki, ‘Optimal Service Pricing for a Cloud Cache’, IEEE Trans. Knowl. Data Eng., vol. 23, no. 9, pp. 1345–1358, Sep. 2011, doi: 10.1109/TKDE.2011.35.

[58] Y. Park, S. C. Gates, W. Teiken, and P.-C. Cheng, ‘An experimental study on the measurement of data sensitivity’, in Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security - BADGERS ’11, Salzburg, Austria, 2011, pp. 70–77. doi: 10.1145/1978672.1978681.

[59] Y. Yang and X. Peng, "Trust-Based Scheduling Strategy for Workflow Applications in Cloud Environment," 2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Compiegne, France, 2013, pp. 316-320, doi: 10.1109/3PGCIC.2013.53.

[60] Y. Zhang, Q. Wei, C. Chen, M. Xue, X. Yuan, and C. Wang, ‘Dynamic Scheduling with Service Curve for QoS Guarantee of Large-Scale Cloud Storage’, IEEE Trans. Comput., vol. 67, no. 4, pp. 457–468, Apr. 2018, doi: 10.1109/TC.2017.2773511.

[61] Z. Liu, B. Li, Y. Huang, J. Li, Y. Xiang, and W. Pedrycz, ‘NewMCOS: Towards a Practical Multi-Cloud Oblivious Storage Scheme’, IEEE Trans. Knowl. Data Eng., vol. 32, no. 4, pp. 714–727, Apr. 2020, doi: 10.1109/TKDE.2019.2891581.

Appendix I

In order to reduce the size of the computation for conflict constraints, we use a list of all conflicting server combinations A^{cl}, which is a list with the size of the number of groups of all conflicting servers, with each section of the list recording the index of each group of conflicting servers. Then we use it as the parameter for the constraint calculation, which will reduce the size of the operation moderately compared with setting constraints for the $m \times m$ conflict matrix. However, experiments reveal that this method only cuts the time by less than 100 ms, which is far from satisfaction.

Then we tried to compress the number of conflicting servers by selecting one server from the conflicting servers to represent all the conflicting servers, which is equivalent to removing all the conflicting constraints. Initially, we counted the Q of all fields of the server, and if the server is a cloud server, only the lowly sensitive fields are counted. Then, we selected a server with the highest average Q as a representative of these servers to participate in the allocation process. Despite significantly reducing the runtime, with a server count of 120, a field count of 40, and a server conflict rate of 10%, there were still many conflicting servers. This also lead to a significant reduction in the number of available servers, resulting in an extremely high number of unresolved situations.

Another method is to find an optimal solution without setting conflicting constraints, and then check whether this solution contains conflicting servers. If not, this solution is optimal; if it does, add additional constraints and solve the problem again. But after carrying out 100 experiments with a conflict rate of 10%, we found that the optimal solution without setting the conflict constraint basically contains some conflicting servers, and only less than 5% of experiments have a conflict-free solution. The results are equally unsatisfactory.

The method of reassigning conflicting servers is: 1) finding an optimal allocation scheme without adding conflict constraints; 2) selecting all the servers in the scheme with conflicting relationships and the fields assigned to them; 3) mixing those servers with other servers that are not assigned data fields; 4) adding conflict constraints; and 5) solving once more to find a solution for the selected fields that does not conflict the severs. The final solution is then produced by combining the allocation strategy with the servers and fields that have previously been assigned. Even though the gap between the acquired result and the value of the ideal

solution is not great, the percentage of situations with infeasible solution is too high, on average, there are over 90 infeasible solutions for every 100 experiments.

Experiment Data

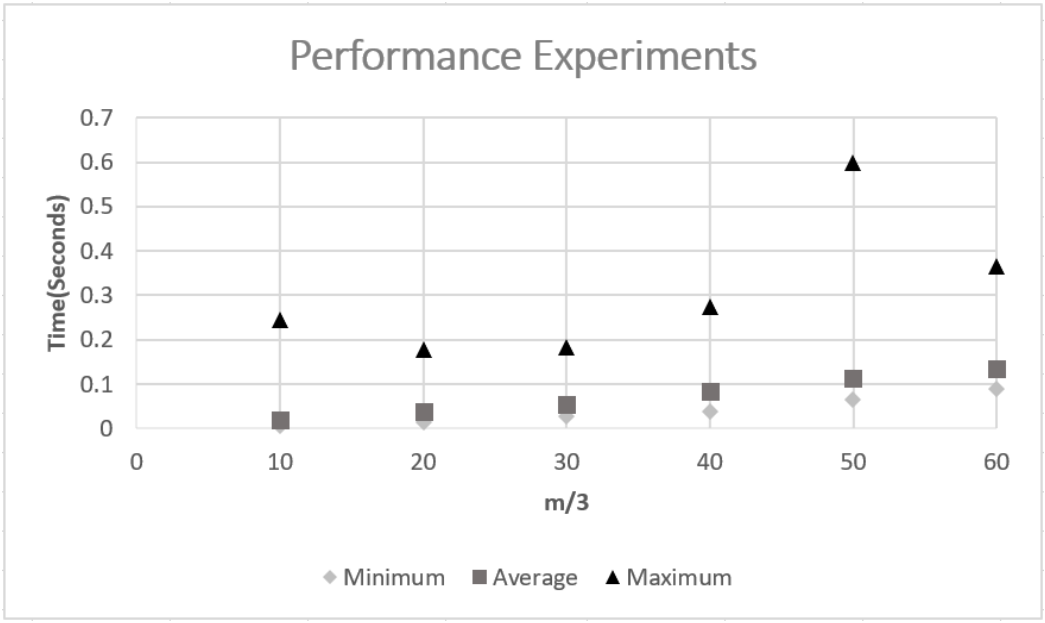

Figure 1: Performance Experiments Result without Conflict Constraints.

Fig. 1 shows the runtime of the original method when the number of data fields is incremented sequentially from 10 to 60 and the number of servers is three times the number of fields, without adding server conflicts.

Fig. 2 shows the runtime of the improved method using MIP start. We believe that if we can offer a decent initial solution to the issue at the outset of the solution, we can significantly minimize the computational work of the solution process and accelerate the resolution, and a mixed integer programming (MIP) start of CPLEX can assist us in achieving this objective. When we are solving an MIP problem, we can provide hints to help CPLEX find an initial solution. These hints consist of pairs of variables and values and are called MIP starts. First, we randomly find a feasible solution using CPLEX, after that, we utilize it as the starting point for a MIP start to determine the best assignment. The running time remains essentially the same compared with the original solution, and at some scales it even takes more time than the original solution.

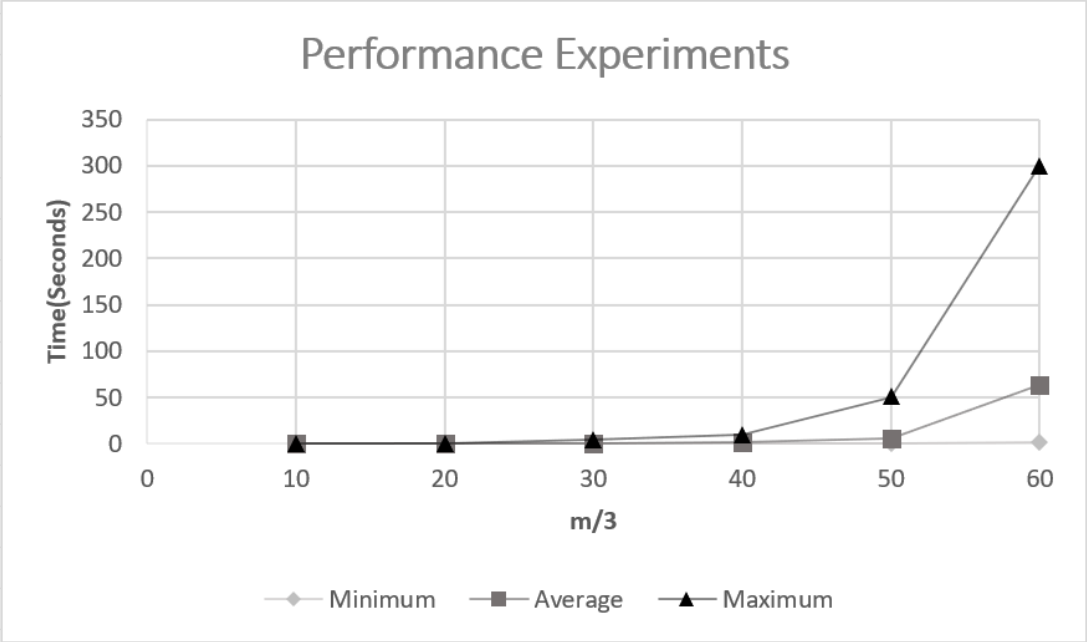

Figure 2: Performance Experiments Result Using MIP Start.

Figure 3 shows the results of the experiments using the greedy algorithm, which we performed 100 times with 120 servers and 40 data fields. The blue points in the figure represent the values of the solutions found by the greedy algorithm, the red points represent the values of the optimal solutions obtained by the CPLEX calculation, and the points with a limited value of 0 indicate that no feasible solution was found.

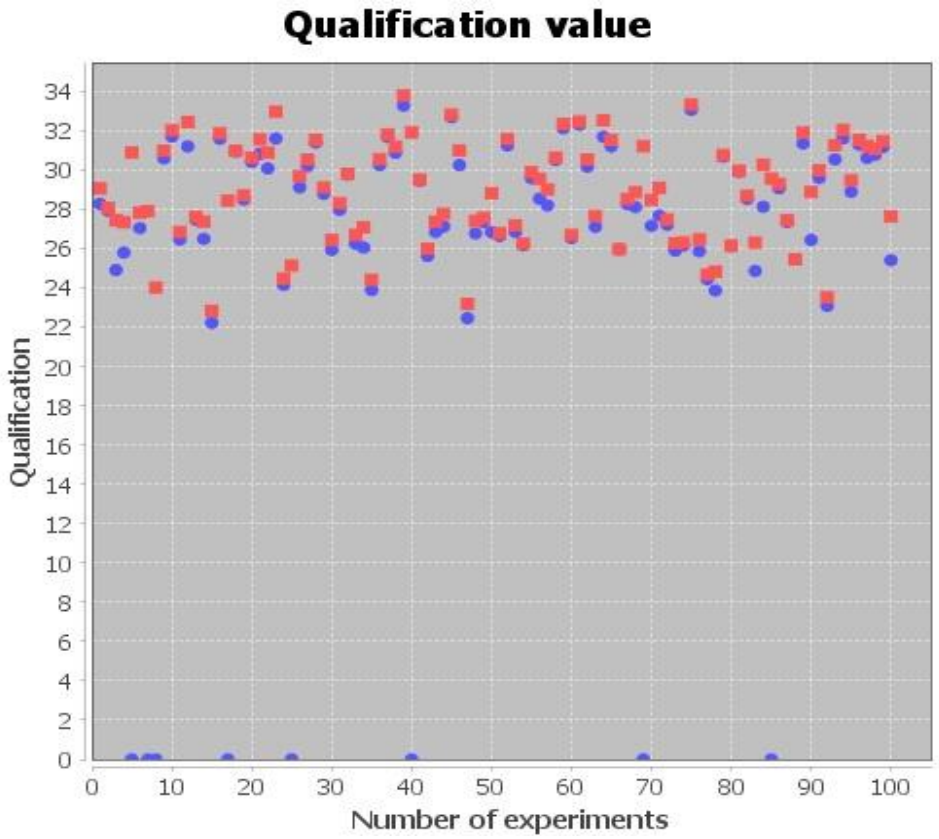

Figure 3: Qualification Values Computed by Greedy Algorithm and CPLEX.

Figure 4 shows that in the experimental results of the modified greedy algorithm, the blue dots represent the values of the solutions derived from the modified greedy algorithm and the red dots represent the values of the optimal solutions obtained from the CPLEX solutions. As can be seen from the figure, the difference between the red and blue dots is not significant, and the value obtained by the modified greedy algorithm is 1.02% less than the value obtained by the CPLEX result. In addition, there were 15 instances without a solution out of 100 experiments.

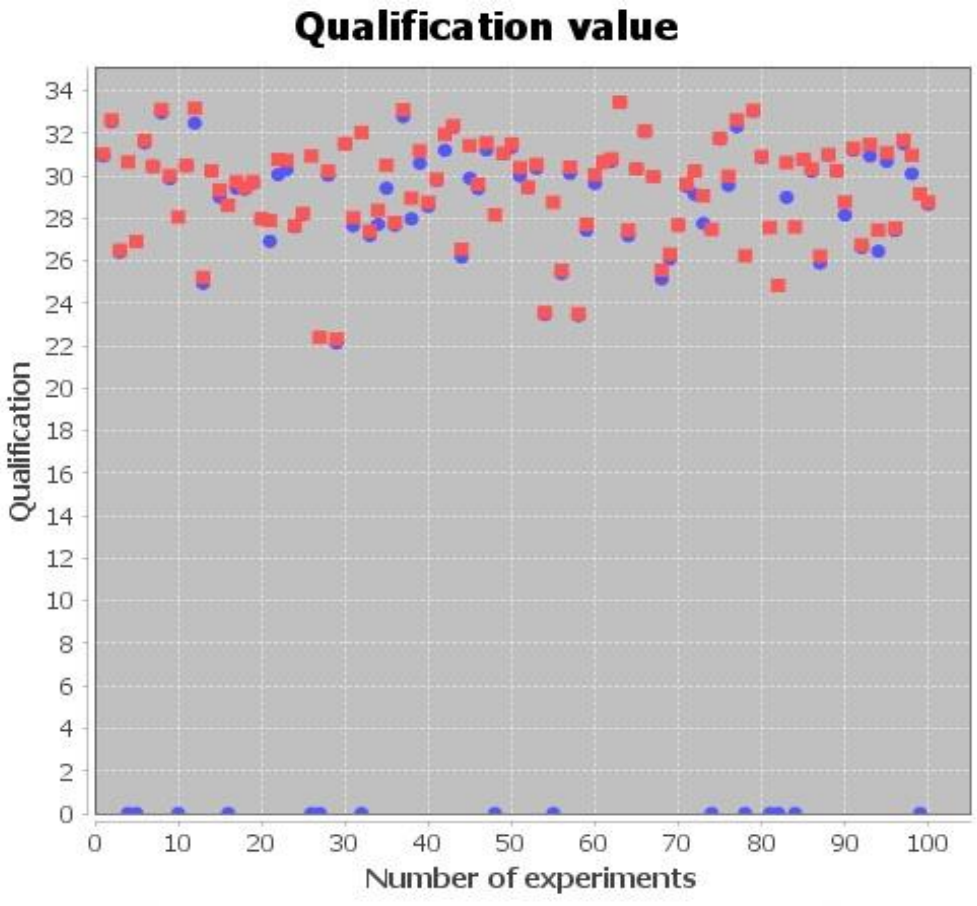

Figure 4: Qualification Values Computed by Modified Greedy Algorithm and CPLEX.

Fig. 5 shows the runtime required using the original method in the case of data requiring an additional backup. We assume up to 300 available servers with an additional backup of all data, starting with 100 storage units and decreasing by 10 storage units in sequence.

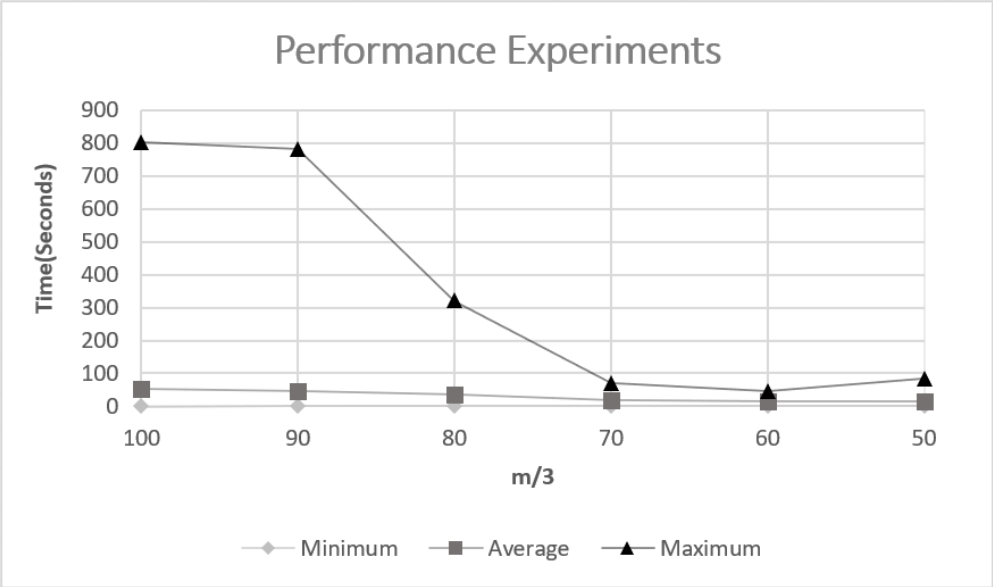

Figure 5: Performance Experiments Result for Additional Data Backup.

Appendix II

The implementation code of the improved method is as follows:

1) Determine the server trust score TS, SS' data field sensitivity score SS, sensitivity threshold t, server performance score PS and coefficient u. Servers are sorted according to public cloud servers first and then edge servers, fields are sorted in ascending order of sensitivity, the number of non-sensitive fields below the threshold is n_n and the number of sensitive fields above the threshold is n_s. Calculate the qualification value Q based on TS, SS, PS.

2) Determine other solution elements, number of backups L, maximum number of server storage fields L^a, server price P, budget B, storage size of the server S, size of each data field D, a list of all servers provided by SPs $A^{c\prime}$, matrix C^{cf} and assignment matrix T, T'.

3) The matrices Q, T and T' are expressed in a one-dimensional array form and all parameters are passed into the ILOG package for solving.

4) Declare objects that requires optimization.

```
IloIntVar[] x = cplex.intVarArray(m * n, 0, 1)
IloIntVar[] y = cplex.intVarArray(nc, 0, 1);
Declare qualification values that have an impact on data privacy.
double[] QNC = new double[nc];
for (int k = 0; k < nc; k++) {
        QNC[k] = CCF[k][4] * Q[(int) (CCF[k][0] * n + CCF[k][1])];
}
Then we need to add the optimization target, and we invoke the following method.
cplex.addMaximize(cplex.sum(cplex.scalProd(x, Q), cplex.scalProd(y, QNC)));
5) Add all constraints.
For constraint (6):
for (int j = 0; j < n; j++) {
        IloLinearNumExpr constraint2 = cplex.linearNumExpr();
        for (int i = 0; i < m; i++) {
```

```
                constraint2.addTerm(1, x[i * n + j]);
        }
        cplex.addEq(constraint2, L[j]);
}
For constraint (7):
for (int i = 0; i < m; i++) {
        IloLinearNumExpr constraint3 = cplex.linearNumExpr();
        for (int j = 0; j < n; j++) {
                constraint3.addTerm(1, x[i * n + j]);
        }
        cplex.addLe(constraint3, LA[i]);
}
For constraint (8):
for (int j = ns; j < n; j++) {
        IloLinearNumExpr constraint4 = cplex.linearNumExpr();
        for (int i = 0; i < mc; i++) {
                constraint4.addTerm(1, x[i * n + j]);
        }
        cplex.addEq(constraint4, 0);
}
For constraint (9):
for (int i = 0; i < m; ++i) {
        IloLinearNumExpr storeConstrain = cplex.linearNumExpr();
        for (int j = 0; j < n; ++j) {
                storeConstrain.addTerm(D[j], x[i * n + j]);
        }
        cplex.addLe(storeConstrain, S[i]);
}
For constraint (10):
for (int j = 0; j < n; j++) {
        IloLinearNumExpr constraint5 = cplex.linearNumExpr();
        for (int i = 0; i < m; i++) {
```

```
                constraint5.addTerm(P[i], x[i * n + j]);
        }
        cplex.addLe(constraint5, B[j]);
}
For constraint (12):
for (int i = 0; i < serverProviders.size(); i++) {
        List<Integer> servers = serverProviders.get(i);
        for (int j = 0; j < servers.size(); j++) {
                int server = servers.get(j);
                for (int k = 0; k < servers.size(); k++) {
                        if (servers.get(k) != server && server >= m1 && servers.get(k) >= m1) {
                                IloLinearNumExpr server1 = cplex.linearNumExpr();
                                IloLinearNumExpr others = cplex.linearNumExpr();
                                for (int l = n1 + n2; l < n; l++) {
                                        server1.addTerm(1, x[server * n + l]);
                                        others.addTerm(1, x[servers.get(k) * n + l]);
                                }
                                cplex.add(cplex.ifThen(cplex.ge(server1, 1), cplex.eq(others, 0)));
                        }
                }
        }
}
For constraint (13):
for (int k = 0; k < nc; k++){
        IloLinearNumExpr exprTConstraint = cplex.linearNumExpr();
        exprTConstraint.addTerm(1, x[(int) CCE[k][0] * n + (int) CCE[k][1]]);
        exprTConstraint.addTerm(1, x[(int) CCE[k][2] * n + (int) CCE[k][3]]);
        exprTConstraint.addTerm(-1, y[k]);
        cplex.addLe(exprTConstraint, 1);
}
for (int k = 0; k < nc; k++) {
        IloLinearNumExpr exprTConstraint1 = cplex.linearNumExpr();
```

```
        exprTConstraint1.addTerm(1, x[(int) CCE[k][0] * n + (int) CCE[k][1]]);
        exprTConstraint1.addTerm(1, x[(int) CCE[k][2] * n + (int) CCE[k][3]]);
        exprTConstraint1.addTerm(-2, y[k]);
        cplex.addGe(exprTConstraint1, 0);
}
```

6) Finally, the cplex.solve() method is used to determine whether there is an optimal solution and to derive a specific solution.

Code for solving in the case of large-scale data and backups:

```
IloCplex cplex = new IloCplex();
IloIntVar[] x = cplex.intVarArray(m * n, 0, 1);
 IloIntVar[] y = cplex.intVarArray(nc, 0, 1);
double[] QNC = new double[nc];
 for (int k = 0; k < nc; k++) {
        QNC[k] = CCE[k][4] * Q[(int) (CCE[k][0] * n + CCE[k][1])];
}
cplex.addMaximize(cplex.sum(cplex.scalProd(x, Q), cplex.scalProd(y, QNC)));
cplex.setOut(null);
for (int j = n1 + n2; j < n; j++) {
        IloLinearNumExpr sensitivityConstrain2 = cplex.linearNumExpr();
        for (int i = 0; i < m1; i++) {
                sensitivityConstrain2.addTerm(1, x[i * n + j]);
        }
        cplex.addEq(sensitivityConstrain2, 0);
}
for (int j = 0; j < n; j++) {
        IloLinearNumExpr exprReqConstrain = cplex.linearNumExpr();
        for (int i = 0; i < m; i++) {
                exprReqConstrain.addTerm(1, x[i * n + j]);
        }
        cplex.addEq(exprReqConstrain, L[j]);
}
for (int i = 0; i < m; i++) {
```

```
        IloLinearNumExpr AgentAbilityConstraint = cplex.linearNumExpr();
        for (int j = 0; j < n; j++) {
                AgentAbilityConstraint.addTerm(1, x[i * n + j]);
        }
        cplex.addLe(AgentAbilityConstraint, LA[i]);
}
for (int j = 0; j < n; j++) {
        IloLinearNumExpr budgetConstrain = cplex.linearNumExpr();
        for (int i = 0; i < m; i++) {
                budgetConstrain.addTerm(P[i], x[i * n + j]);
        }
        cplex.addLe(budgetConstrain, B[j]);
}
for (int i = 0; i < m; ++i) {
        IloLinearNumExpr storeConstrain = cplex.linearNumExpr();
        for (int j = 0; j < n; ++j) {
                storeConstrain.addTerm(DQ[j], x[i * n + j]);
        }
        cplex.addLe(storeConstrain, SC[i]);
}
for (int i = 0; i < serverProviders.size(); i++) {
        List<Integer> servers = serverProviders.get(i);
                for (int j = 0; j < servers.size(); j++) {
                        int server = servers.get(j);
                        for (int k = 0; k < servers.size(); k++) {
                                if (servers.get(k) != server && server >= m1 && servers.get(k) >=
                        m1) {
                                        IloLinearNumExpr server1 = cplex.linearNumExpr();
                                        IloLinearNumExpr others = cplex.linearNumExpr();
                                        for (int l = n1 + n2; l < n; l++) {
                                                server1.addTerm(1, x[server * n + l]);
                                                others.addTerm(1, x[servers.get(k) * n + l]);
```

```
                                        }
                                        cplex.add(cplex.ifThen(cplex.ge(server1, 1),cplex.eq(others,
                                        0)));
                                }
                        }
                }
        }
for (int k = 0; k < nc; k++) {
        IloLinearNumExpr exprTConstraint = cplex.linearNumExpr();
        exprTConstraint.addTerm(1, x[(int) CCE[k][0] * n + (int) CCE[k][1]]);
        exprTConstraint.addTerm(1, x[(int) CCE[k][2] * n + (int) CCE[k][3]]);
        exprTConstraint.addTerm(-1, y[k]);
        cplex.addLe(exprTConstraint, 1);
}
for (int k = 0; k < nc; k++) {
        IloLinearNumExpr exprTConstraint1 = cplex.linearNumExpr();
        exprTConstraint1.addTerm(1, x[(int) CCE[k][0] * n + (int) CCE[k][1]]);
        exprTConstraint1.addTerm(1, x[(int) CCE[k][2] * n + (int) CCE[k][3]]);
        exprTConstraint1.addTerm(-2, y[k]);
        cplex.addGe(exprTConstraint1, 0);
}
if (cplex.solve()) {
        bILOG_result = true;
        optimized_result = cplex.getObjValue();
        double[] val = cplex.getValues(x);              for (int j = 0; j < val.length; j++) {
                A[j / n][j % n] = (int) (val[j] + 0.000001);
                TR[j / n][j % n] = A[j / n][j % n];
        }
        cplex.end();
}
```

Genetic Algorithm:

```
void initGroup() {
```

```
        int[] o = new int[LL];
        for (int i = 0; i < m; i++) {
                for (int j = 0; j < n; j++) {
                        if (firstT[i][j] == 1) {
                                o[i] = 1;
                                break;
                        }
                }
        }
        for (int i = 0; i < LL; i++) {
                oldPopulation[0][i] = o[i];
        }
}
public double evaluate(int[] chromosome) {
        int[][] A = new int[m][n];
        try {
                IloCplex cplex = new IloCplex();
                IloIntVar[] x = cplex.intVarArray(m * n, 0, 1);
                for (int j = 0; j < n; j++) {
                        IloLinearNumExpr sensitivityConstrain1 = cplex.linearNumExpr();
                        for (int i = 0; i < m; i++) {
                                if (chromosome[i] == 0) {
                                        sensitivityConstrain1.addTerm(1, x[i * n + j]);
                                }
                        }
                        cplex.addEq(sensitivityConstrain1, 0);
                }
                for (int j = 0; j < n1; j++) {
                        IloLinearNumExpr sensitivityConstrain1 = cplex.linearNumExpr();
                        for (int i = m1; i < m; i++) {
                                sensitivityConstrain1.addTerm(1, x[i * n + j]);
                        }
```

```
            cplex.addEq(sensitivityConstrain1, 0);
    }
    for (int j = n1; j < n; j++) {
            IloLinearNumExpr sensitivityConstrain2 = cplex.linearNumExpr();
            for (int i = 0; i < m1; i++) {
                    sensitivityConstrain2.addTerm(1, x[i * n + j]);
            }
            cplex.addEq(sensitivityConstrain2, 0);
    }
    for (int j = 0; j < n; j++) {
            IloLinearNumExpr exprReqConstrain = cplex.linearNumExpr();
            for (int i = 0; i < m; i++) {
                    exprReqConstrain.addTerm(1, x[i * n + j]);
            }
            cplex.addEq(exprReqConstrain, L[j]);
    }
    for (int i = 0; i < m; i++) {
            IloLinearNumExpr AgentAbilityConstraint = cplex.linearNumExpr()
            for (int j = 0; j < n; j++) {
                    AgentAbilityConstraint.addTerm(1, x[i * n + j]);
            }
            cplex.addLe(AgentAbilityConstraint, LA[i]);
    }
    for (int i = 0; i < m; ++i) {
            IloLinearNumExpr storeConstrain = cplex.linearNumExpr();
            for (int j = 0; j < n; ++j) {
                    storeConstrain.addTerm(DQ[j], x[i * n + j]);
            }
            cplex.addLe(storeConstrain, SC[i]);
    }
    for (int j = 0; j < n; j++) {
            IloLinearNumExpr budgetConstrain = cplex.linearNumExpr();
```

```
                    for (int i = 0; i < m; i++) {
                        budgetConstrain.addTerm(price[i], x[i * n + j]);
                    }
                    cplex.addLe(budgetConstrain, budget[j]);
                }
                for (int i = 0; i < conList.size(); i++) {
                    List<Integer> servers1 = serverProviders.get(conList.get(i).get(0));
                    List<Integer> servers2 = serverProviders.get(conList.get(i).get(1));
                    IloLinearNumExpr conflict1 = cplex.linearNumExpr();
                    IloLinearNumExpr conflict2 = cplex.linearNumExpr();
                    for (int j = 0; j < servers1.size(); j++) {
                        for (int k = 0; k < n; k++) {
                            conflict1.addTerm(1, x[servers1.get(j) * n + k]);
                        }
                    }
                    for (int j = 0; j < servers2.size(); j++) {
                        for (int k = 0; k < n; k++) {
                            conflict2.addTerm(1, x[servers2.get(j) * n + k]);
                        }
                    }
                    cplex.add(cplex.ifThen(cplex.ge(conflict1, 1), cplex.eq(conflict2, 0)));
                }
            }
        }
        void countRate() {
            int k;
            double sumFitness = 0;
            double[] tempf = new double[scale];
            for (k = 0; k < scale; k++) {
                tempf[k] = fitness[k];
                sumFitness += tempf[k];
            }
```

```
        Pi[0] = (float) (tempf[0] / sumFitness);
                for (k = 1; k < scale; k++) {
                        Pi[k] = (float) (tempf[k] / sumFitness + Pi[k - 1]);
                }
}
public void selectBestGh() {
        int k, i, maxid;
        double maxevaluation;
        maxid = 0;
        maxevaluation = fitness[0];
        for (k = 1; k < scale; k++) {
                if (maxevaluation < fitness[k]) {
                        maxevaluation = fitness[k];
                        maxid = k;
                }
        }
        if (bestLength < maxevaluation) {
                bestLength = maxevaluation;
                bestT = t;
                for (i = 0; i < LL; i++) {
                        bestTour[i] = oldPopulation[maxid][i];
                }
        }
        copyGh(0, maxid);
}
public void copyGh(int k, int kk) {
        int i;
        for (i = 0; i < LL; i++) {
                newPopulation[k][i] = oldPopulation[kk][i];
        }
}
public void select() {
```

```
        int k, i, selectId;
        float ran1;
        for (k = 1; k < scale; k++) {
                ran1 = (float) (random.nextInt(65535) % 1000 / 1000.0);
                for (i = 0; i < scale; i++) {
                        if (ran1 <= Pi[i]) {
                                break;
                        }
                }
                selectId = i;
                copyGh(k, selectId);
        }
}
public void evolution() {
        int k;
        selectBestGh();
        select();
        float r;
        for (k = 0; k < scale; k = k + 2) {
                r = random.nextFloat();
                if (r < Pc) {
                        OXCross(k, k + 1);
                }
                else {
                        r = random.nextFloat();
                        if (r < Pm) {
                                OnCVariation(k);
                        }
                        r = random.nextFloat();
                        if (r < Pm) {
                                OnCVariation(k + 1);
                        }
```

```
                }
        }
}
void OXCross(int k1, int k2) {
        int i, j, flag;
        int ran1, ran2, temp = 0;
        ran1 = random.nextInt(65535) % LL;
        ran2 = random.nextInt(65535) % LL;
        while (ran1 == ran2) {
                ran2 = random.nextInt(65535) % LL;
        }
        if (ran1 > ran2) {
                temp = ran1;
                ran1 = ran2;
                ran2 = temp;
        }
        flag = ran2 - ran1 + 1;
        for (i = 0, j = ran1; i < flag; i++, j++) {
                temp = newPopulation[k1][j];
                newPopulation[k1][j] = newPopulation[k2][j];
                newPopulation[k2][j] = temp;
        }
}
public void OnCVariation(int k) {
        int ran1, ran2, temp;
        int count;
        count = random.nextInt(65535) % LL;
        for (int i = 0; i < count; i++) {
                ran1 = random.nextInt(65535) % LL;
                ran2 = random.nextInt(65535) % LL;
                while (ran1 == ran2) {
                        ran2 = random.nextInt(65535) % LL;
```

```
                }
                temp = newPopulation[k][ran1];
                newPopulation[k][ran1] = newPopulation[k][ran2];
                newPopulation[k][ran2] = temp;
        }
}
public void solve() {
        initGroup();
        fitness[0] = firstR;
        countRate();
        for (t = 0; t < MAX_GEN; t++) {
                evolution();
                for (int k = 0; k < scale; k++) {
                   for (int i = 0; i < LL; i++) {
                           oldPopulation[k][i] = newPopulation[k][i];
                   }
                }
                for (int k = 0; k < scale; k++) {
                        fitness[k] = evaluate(oldPopulation[k]);
                }
                countRate();
        }
}
```

Curriculum Vitae

Name:	Chengyu Peng
Post-secondary Education and Degrees:	Jiangxi University of Finance and Economics Nanchang, Jiangxi, China 2016-2020 B.E. Laurentian University Sudbury, Ontario, Canada 2021-2023 M.Sc.
Related Work Experience	Teaching Assistant Laurentian University 2021-2023

Publications:

C. Peng, H. Zhu, L. Liu and R. Grewal, "Optimal Data Allocation in the Environment of Edge and Cloud Servers," 2022 IEEE International Conference on Networking, Sensing and Control (ICNSC), Shanghai, China, 2022, pp. 1-6, doi: 10.1109/ICNSC55942.2022.10004065.

C. Peng, H. Zhu, L. Liu and R. Grewal, "Optimal Data Allocation in Edge-cloud Environment Considering Privacy Enhancement," in IEEE Transactions on Knowledge and Data Engineering, Under Review.

C. Peng, H. Zhu, L. Liu and R. Grewal, "Optimal Procurement in Consideration of Carbon Emissions," 2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Maui, Hawaii, 2023, Under Review.