Cache-Aided Delivery Networks with Correlated Content in a Shared Cache Framework

Behnaz Merikhi

A Thesis
In the Department
of
Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of
Doctor of Philosophy (Electrical and Computer Engineering) at
Concordia University
Montreal, Quebec, Canada

October 2022

CONCORDIA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: Behnaz Merikhi

Entitled: Cache-Aided Delivery Networks with Correlated Content in a Shared Cache Framework

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Electrical and Computer Engineering)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Chair
Dr. Wen-Fang Xie

_____ External Examiner
Dr. Benoit Champagne

_____ External to Program
Dr. Hovhannes Harutyunyan

_____ Examiner
Dr. Walaa Hamouda

_____ Examiner
Dr. Yousef R. Shayan

_____ Thesis Supervisor
Dr. Mohammad Reza Soleymani

Approved by:

_____
Dr. Jun Cai, Graduate Program Director

October 6, 2022

_____
Dr. Mourad Debbabi, Dean
Gina Cody School of Engineering and Computer Science

Abstract

Cache-Aided Delivery Networks with Correlated Content in a Shared Cache Framework

Behnaz Merikhi, Ph.D.
Concordia University, 2022

Internet traffic is growing exponentially due to the penetration of powerful internet-connected devices and cutting-edge technologies. Additionally, the rise in internet usage has coincided with a shift in the nature of data traffic from voice-based to content-based usage, putting significant stress on delivery networks. Despite the infrastructural advancements in communication networks over the past few years, content delivery networks (CDNs) still face challenges in keeping up with the high delivery data rates and suffer from the imbalanced network load between off-peak hours and peak hours.

In this regard, *content caching* has emerged as an efficient technique to combat the high delivery date rates and maintain a balanced network load while improving the quality of services (QoS) by storing some popular content close to the end users. Caching networks operate in two phases; the *placement phase* during off-peak hours before users reveal their demands and the *delivery phase*, which is accomplished when users' demands are revealed to the server during peak hours. As the server is unaware of the demands during the placement phase, this phase must be designed carefully to minimize the delivery rate regardless of the requested content during peak hours.

This dissertation studies cache-aided delivery networks with correlated content in a shared cache framework. A shared cache framework is beneficial in the current and next-generation wireless networks as it provides a local cache to all users within small base stations (SBSs), relieving strain on the backhaul. Furthermore, the library of a caching network could consist of content with a high degree of similarity in many practical applications; Therefore, exploiting the similarity among library content can also be leveraged to reduce the delivery rate in such networks.

In this dissertation, we look at the proposed caching network from an information-theoretic perspective and formulate it as a distributed source coding problem with side information at the decoder. Then, the critical question arises as to what should be cached as side information to reduce the delivery rate of the network efficiently.

To answer this question, we propose an automatic clustering scheme using artificial intelligence (AI)-based optimization techniques to identify the selected side information for the entire library. We comprehensively evaluate the performance of the general clustering framework in a separate chapter by considering different datasets and distance measures. The general clustering framework enables us to develop two novel clustering schemes as a part of the placement phase of the proposed caching networks under different settings throughout this study, considering both the similarity and popularity of the library content.

Upon identifying the selected side information for such networks, the next question that should be answered is how we should place the side information into caches; And consequently, what is the delivery strategy for this placement scheme? We have furnished our answer to these questions by considering three different caching networks: first, a network in a single shared cache framework under lossy caching. Next is a network with multiple shared caches under uniform popularity, and finally, a network with multiple shared caches under non-uniform preferences. In such networks, we address the placement and delivery strategy to show the trade-off between the delivery rate and the memory size of the system. We calculate the peak and expected rates of the studied networks by considering the rate-distortion function and caching strategy. We also introduce the optimum library partitioning formulated to minimize the peak delivery rate in the system.

The performance analysis and extensive simulations of the proposed solution confirm that our scheme provides a considerable boost in network efficiency compared to legacy caching schemes due to our problem formulation and the careful extraction of side information during the placement phase.

To my wonderful husband, Sina

Whose endless love, support, and encouragement embrace me every day

To my lovely parents, Fatemeh and Manouchehr

For their unconditional love

And

To my dear brothers, Babak and Behrooz

Who have always been a constant source of inspiration to me

Acknowledgments

First and foremost, I wish to express my sincere appreciation to my supervisor, Professor Mohammad Reza Soleymani, for providing me with this wonderful opportunity in his research group. Being a part of his research group is an honor that will stay with me forever. I am forever grateful to him for his continuous encouragement and guidance throughout my Ph.D. journey. This dissertation would not have been possible without his support and constructive feedback.

I would also like to express my deepest gratitude to my Ph.D. committee members, Professor Yousef R. Shayan, Professor Walaa Hamouda, and Professor Hovhannes Harutyunyan, for their time and helpful feedback. My special gratitude goes to Professor Benoit Champagne from McGill University for his time as my external examiner.

I would like to thank Professor Ali Mirjalili from Torrens University Australia and Dr. Mohammad Mirjalili from Polytechnique Montréal. I am thankful to them for providing me with the great opportunity to collaborate and learn the fundamentals of artificial intelligence and optimization algorithms, which was invaluable to my research endeavors.

I would like to thank my dearest parents, Fatemeh and Manouchehr, and my brothers, Babak and Behrooz, for all the selfless love and support they have given me throughout my life. My gratitude for their devotion and support is beyond words.

Last but not least, I am most thankful to the love of my life, Sina, who has always believed in me and supported me every step of the way. He lived every single minute of the challenges and achievements with me throughout this journey, and I could not succeed without his love, patience, and encouragement.

Table of Content

List of Figures

List of Tables

List of Abbreviations

ACDE	Automatic Clustering Using Differential Evolution
ACDE-O	Automatic Clustering Using Differential Evolution with Oscillation
ACPAHS	Automatic Clustering using Parameter Adaptive Harmony Search Algorithm
ACPSO	Automatic Clustering based on Particle Swarm Optimization
ADEFC	Automatic Differential Evolution Based Fuzzy Clustering
AGCUK	Automatic Genetic Clustering for Unknown K
AI	Artificial Intelligence
AKC-BCO	Automatic Kernel Clustering with Bee Colony Optimization
AP	Access Point
BA	Binary Address
BBA	Binary Bat Algorithm
BDA	Binary Dragonfly Algorithm
BGA	Binary Genetic Algorithm
BPSO	Binary Particle Swarm Optimization
BSC	Binary Symmetric Channel
CACS	Correlation-Aware Clustering Scheme
CC	Coded Caching
CDN	Content Delivery Network
CSI	Clusters' side information
CS-index	Compact-Separated index
DA	Decimal Address
DB-index	Davis-Bouldin index
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DCPSO	Dynamic PSO
DE	Differential Evaluation
DSC	Distributed Source Coding
EM	Expectation-Maximization
FAPSO	Firefly Algorithm and Particle Swarm Optimization
FCM	Fuzzy C-Mean
GA	Genetic Algorithm
gBest	Global Best

GCUK	Genetic-based clustering with an unknown number of clusters
HetNet	Heterogeneous Network
HPF	Highest Popularity First
i.i.d	Independent and identically distributed
iABC	Improved Artificial Bee Colony
IC	Index Coding
IWO	Invasive Weed Optimization
LFU	Least Frequently Used
LRU	Least Recently Used
MBS	Macro Base Station
MinPts	Minimum number of neighbors within epsilon radius in DBSCAN
MOIMPSO	Multi-objective Immunized Particle Swarm Optimization
NC	Network Coding
NPIR	The nearest point with the indexing ratio
PB-CACS	Popularity-Based Correlation-Aware Clustering Scheme
PSI	Popular side information
PSO	Particle Swarm Optimization
QoE	Quality of Experience
QoS	Quality of Service
SA	Simulated Annealing
SBS	Small Base Stations
SMC	Simple Matching Coefficient
SW	Slepian-Wolf
TGCA	Two-stage Genetic Clustering Algorithm
WZ	Wyner-Ziv
ZB	Zettabyte

List of Symbols

A	Loudness of emitted sound in BBA
$\boldsymbol{\mathcal{A}}_{m\times l}$	Dataset containing m patterns with l features
B	Length of a binary vector
β	Maximum number of requested representatives in a cache across all caches
$\mathrm{c}_1, \mathrm{c}_2$	Acceleration coefficients of the BPSO algorithm
c_i	Cluster i
C	Set of achieved clusters
$\boldsymbol{C}$	Selected clustering solution upon performing the clustering scheme
$\boldsymbol{C}^S$	Clustering solution S
ξ	Zipf distribution parameter
$D_{k,i}$	Maximum allowable distortion for file i within cluster k
$d(.,.)$	Distance measure
$d_{i,j}^H$	Hamming distance between two vectors i and j
$\bar{d}_C^{max}$	Average of the maximum distance over all clusters
$d(.,.)$	Distortion measure
$\boldsymbol{\delta}$	Given distortion
δ_C^{max}	Maximum distance within clusters in clustering solution C
δ^{c_i}	Maximum intra-cluster distance in a cluster $c_i \in C$
$\delta_{k,i}^2$	Distance of a content file i to its representative within cluster k
Δ_C	Distortion deviation over all clusters in the clustering solution C
ΔP_C	Deviation of aggregate popularity in the clustering solution C
$\bar{E}_C^{min}$	Average of minimum inter-cluster distances over all clusters
η	Number of caches which has a request from the server in a set of β
$Fr_i(t)$	Frequency of particle i^{th} in the optimization algorithm at iteration t
$\boldsymbol{f}(.)$	Objective function
$\mathcal{F}$	Set of library content files
f_i	Content file i in the library
$\hat{F}_i$	The representative of cluster i
$\boldsymbol{F}$	Set of all representatives by the achieved clustering solution
f_i^j	Requested file f_i by user j

$F_{k,i}$	Notation of file i located in cluster k (after clustering)
$F_{k,i}^{j_z}$	Notation of file i located in cluster k requested by user j from cache z
$\varphi_i(t)$	Position of particle i^{th} in the optimization algorithm at iteration t
$H(f_i)$	The entropy of content file f_i
$H(.\|.)$	Conditional entropy between two random variables
$H(f_1, \dots, f_m)$	Joint entropy of m content files
K	Number of clusters
$K_{\boldsymbol{C}}$	Achieved number of clusters /representatives in clustering solution $\boldsymbol{C}$
K_{min}/K_{max}	Minimum/Maximum number of clusters in a clustering solution
L	Number of bits to address cluster numbers in the general clustering scheme
λ	Optimal allowable distortion introduced to the files
$\Lambda_{i,j}$	Inter-cluster distance between cluster i^{th} and j^{th} for the DB-index validity measure
m	Number of content files in the library
m_k	Number of files within cluster k
m_C	Content files in the PB-CACS (clustered files)
m_H	Content files with high popularity (PSIs)
M	Size of each shared cache
M_R	Required memory size to store CSIs per shared cache
N	Total number of users in the network
$\tilde{n}$	Binary random variable
p_i	Popularity of file i
$\bar{P}_C$	Average aggregate popularity on all clusters in the clustering solution C
P_k	Aggregate popularity of cluster k
$\mathcal{P}(.,.)$	Joint probability
$\mathcal{P}_{\mathcal{F}}$	Joint distribution of the library content files $\mathcal{F}$
q_i	Demand for file $i \in \mathcal{F}$
q_i	The probability of requesting the i^{th} file according to the Zipf distribution
Q_z	All demands of cache Y_z
Q	Unique representatives requested across all caches
Q'	Unique clustered files requested across all caches
R_{CM}	Delivery rate required for the coded multicast messages
R_{RS}	Delivery rate required for the refinement segments
R_{CM}^P	Peak delivery rate required for the coded multicast messages
R_{RS}^P	Peak delivery rate required for the refinement segments

r_i	Pulse rate in BBA
ρ_0	Crossover probability
ρ	Correlation parameter
Σ	Covariance Matrix
ψ	Set of all feasible ways of clustering a given dataset
T	An integer parameter for the sake of symmetry in splitting files into segments
ϑ	A random number from a uniform distribution in $[0,1]$
U_i	Number of users in SBS i connected to the cache Y_i
Υ_i	Intra-cluster distances in the i^{th} cluster for the DB-index validity measure
$V_i(t)$	Velocity of particle i^{th} in the optimization algorithm at the iteration t
ϖ	Inertia weight
X_i	Encoded message of file i
Y_i	Shared cache i in the network
y_z	Mapped vector for requests of SBS z
Z	Number of shared caches in the network
ζ	Maximum number of requested representatives per shared cache

Chapter 1

Introduction

During the last two decades, wireless networks have witnessed a dramatic rise in content-related data traffic due to the continuous development of cutting-edge technologies and smart applications along with the increasing number of internet users demanding high-quality services.

According to Cisco studies, internet traffic has already exceeded the zettabyte (ZB) level and will reach 4.8 ZB per year by the end of 2022 [1], while over 72% of this traffic is predicted to be content-related due to the frequent use of streaming services, social media and smart applications in recent years [2].

Figure 1.1 shows the IP video traffic predicted by the Cisco studies for 2017 to 2022.

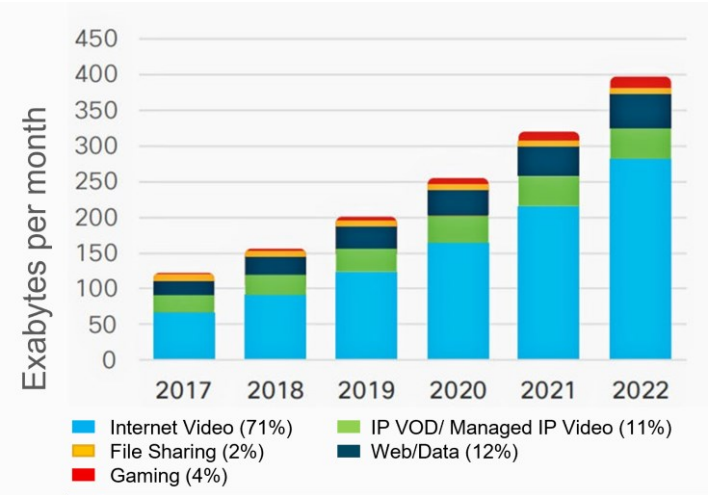


Figure 1.1: Global IP video traffic forecast by Cisco VNI from 2017 to 2022 [2]

1.1. Research Motivation

As content-related internet traffic grows exponentially, delivery networks are forced to handle massive volumes of content, which is a serious challenge for current content delivery networks, especially during peak hours. Therefore, providing high-quality services for the end-users requires a solution beyond traditional content delivery networks, which is why designing efficient content delivery solutions have become increasingly important in the current and next-generation wireless networks [3].

In recent years, content caching has proved to be an efficient technique to improve the quality of services and reduce the high delivery data rate by storing some popular content close to the end-users. Caching networks generally operate in two phases; the placement phase during off-peak hours before users reveal their demand and the delivery phase, which is accomplished when users' demand is revealed to the server during peak hours.

As the server is unaware of the demands during the placement phase, this phase must be carefully designed and implemented to improve the quality of services and maximize the network performance regardless of the requested content [4].

Conventionally, the most popular files were stored in local memories to provide a local caching gain in the network. This strategy randomly allocates some files to the memory in cases with a

uniform popularity demand distribution. A few years ago, Maddah-Ali and Niesen [5] introduced the *coded caching* (CC) strategy and improved the conventional results to reach a global gain by creating multicast opportunities under the assumption that each user is equipped with an individual cache in the network. Since then, this setting has been studied extensively from different angles [3]-[10], considering various system settings, demand characteristics, and source distribution.

The advancement of caching networks with users having individual caches in the past few years has led to focusing on another aspect of caching networks which considers shared caches for a group of users [11]-[14]. A shared cache framework is highly beneficial in next-generation wireless networks as it allows users in a small base station (SBS) to access a nearby cache in the cell. Also, it is useful in a heterogeneous network (HetNet) environment [15] or as an upper layer of hierarchical caching networks in IoT-based applications [16].

A few recent studies presented promising results in a shared caching setting, but their study has been focused on caching networks with independent sources. As discussed in [17], a library of a caching network could consist of sources with a high degree of similarity in many practical applications. Any event recordings with common scenes and background, news updates, different chunks of a video in video summarization, repeated measurements, new updates, augmented and virtual reality [17]-[19], and crowdsourced multi-view videos [20] are examples of such applications. Figure 1.2 illustrates an example of a scene of interest captured by multiple cameras in the field, which has the same scene and background in all videos from different views.

Despite current advancements in caching networks with correlated sources, all these studies also assumed a setting where each user has been equipped with an individual cache. Thus, an efficient solution is still needed for a network with correlated content in a shared cache framework.

Motivated by the aforementioned issues, our research explores a cache-aided delivery network with correlated content based on a shared cache framework to fill the gap between recent studies on shared caches with independent sources and individual caches with correlated sources. We show that the efficiency of current shared cache networks with independent sources does not carry over to a network with correlated sources in a shared setting. Therefore, finding an efficient solution to fill this gap is still necessary.

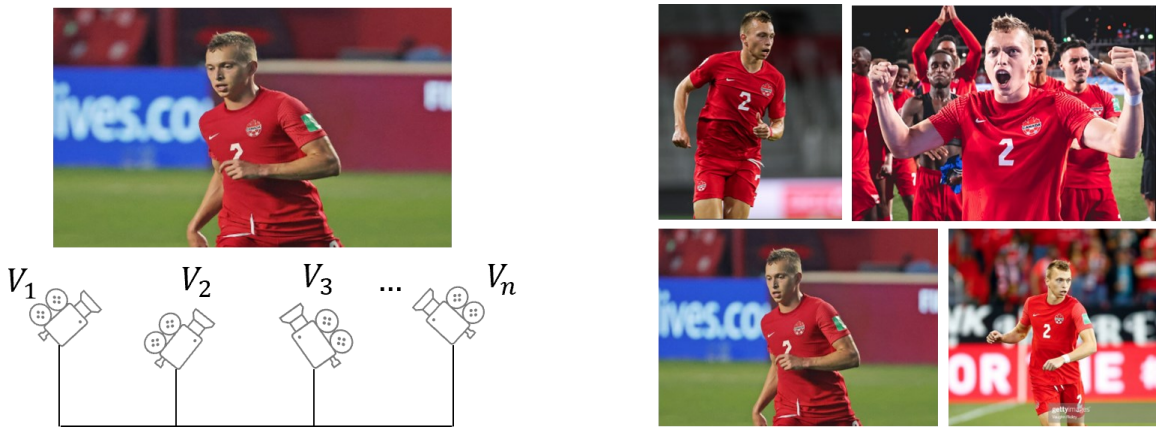


Figure 1.2: Different videos of a person of interest from the same event with the same scene and background[1]

1.2. System Model and Problem Statement

This research considers a centralized cache-aided delivery network with correlated content in a shared cache framework. In line with current studies, we consider a network consisting of one server, e.g., Macro base station (MBS), and multiple receivers, e.g., SBSs, over a shared error-free broadcast link. Each SBS is equipped with a shared cache and serves multiple users. It is assumed that each user is connected to only one SBS at the same time and can receive messages from the server and its SBS. The caching network operates in two phases;

In the placement phase, a set of side information is extracted and placed into the caches according to a proposed placement strategy considering network settings.

In the delivery phase, users reveal their requests independently from each other according to a uniform (Chapters 4 and 5) or non-uniform (Chapter 6) popularity demand distribution. If the connected cache to the user already has (some of) the content, the request (or a portion of the request) can be served locally. Otherwise, the server must transmit requested content not available

[1] The person of interest in these scenes is Alistair Johnston, #2 of the Canada soccer team during the 2022 World Cup Qualifying matches. Images vailable on: https://www.canadasoccer.com/news/canada-announces-squad-for-september-2022-matches/

in the shared cache via broadcast index coding messages and unicast/multicast of the encoded messages[2].

The goal is for every user to be able to reconstruct the file that it requests with the information received from the server and the cached content in the shared cache while the delivery rate of the system is also improved. It should be mentioned that the transmission rate is only considered for the delivery transmission and not for the placement phase, as the placement phase is accomplished during off-peak hours.

Figure 1.3 illustrates the network model of a cache-aided delivery system in a shared cache Framework where each cache is connected to a different group of users.

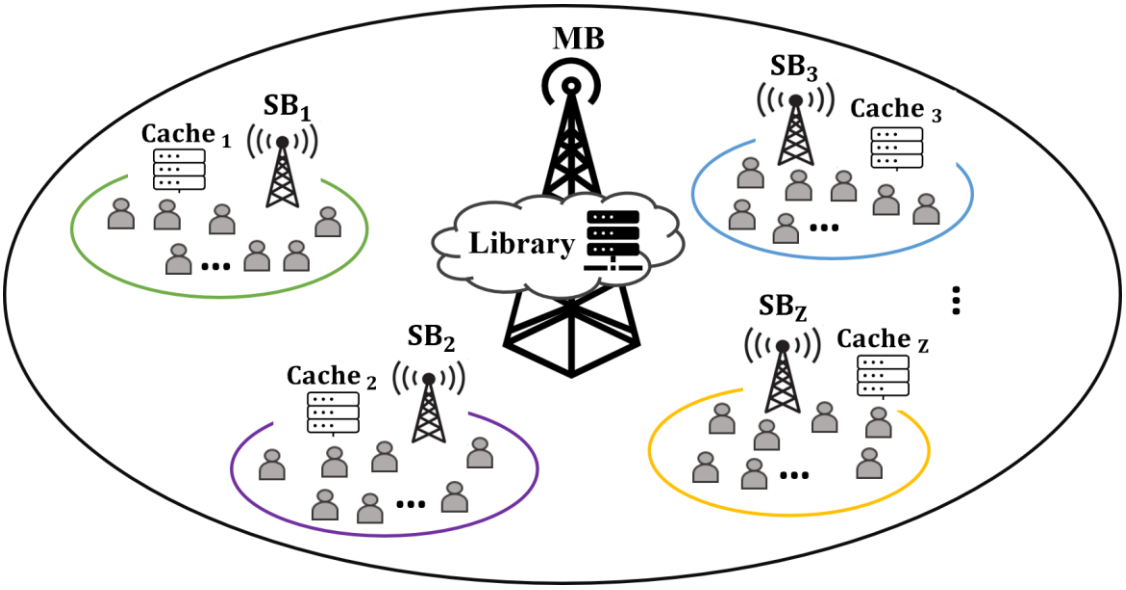


Figure 1.3: Cache-aided delivery system in a shared cache framework with multiple caches in the network

[2] We prefer to refer to the encoded messages as refinement segments throughout this research, and these two terms may be used interchangeably in the following chapters.

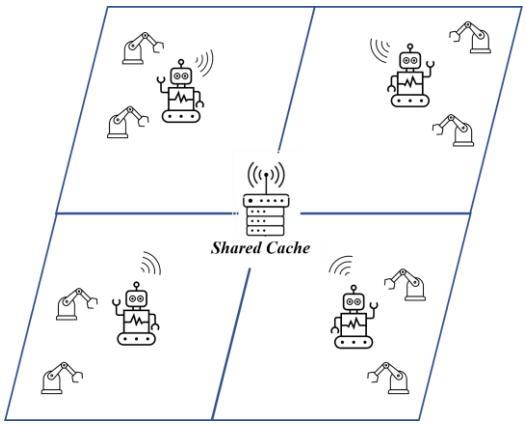


Figure 1.4: Example of a robot factory where all laborers are connected to a shared cache

First, we explore the proposed networks by assuming a single shared cache under lossy caching, and then we extend our study to networks with multiple shared caches under uniform and non-uniform popularity demands. Connecting users to a single shared cache is beneficial where they have a very small memory size. Consider robots as laborers in a factory or drones as operators in deserted areas. In such settings, providing a shared cache in the access point to be filled with the most useful content for the entire network during the placement phase leads to only transmitting small updates (e.g., recent maps and frequent updates of the locations under their coverage) during the delivery phase. Figure 1.4 illustrates a simple example of such networks.

In this dissertation, we study the cache-aided delivery problem from an information-theoretic perspective and formulate it as distributed source coding with side information at the decoder. We focus on the content placement phase to reduce the peak delivery rate and the users' expected distortion during the delivery time in the proposed network. More specifically, we are seeking the answer to the following questions throughout this research:

- What content should be placed into the memory during the placement phase?
- How should we place the selected content into the caches?
- And what content should be transmitted during the delivery phase to serve all possible demands at a low delivery rate in the caching network?
- Moreover, what is the trade-off between the cache size, delivery rate, and the user's expected distortion under different settings in such a system?

To answer the above questions, we proposed two automatic clustering solutions using AI-based optimization techniques to extract the most efficient set of side information for the entire library considering the similarity among sources and the popularity demand distribution. Then, we utilize the side information for the caching strategy and proceed with the delivery phase under different popularity demands to serve all requests.

1.3. Literature Review

A literature review related to our study is presented in this section in two parts: first, we discuss related works on caching networks, and then we present related works on data clustering.

1.3.1. Caching Networks

The increasing number of internet users and IoT devices, in addition to the widespread use of social media and streaming services, has put significant stress on current content delivery networks. Content caching has emerged as an effective solution to combat this stress and reduce the high delivery data rate to improve the quality of services for the end-users in such networks.

Over the past few years, many studies have been conducted to face the key challenges of caching networks from different angles. In general, one perspective is optimizing the delivery phase for specific demands and fixed cache content [21][22]. The other point of view is optimizing the content placement for a fixed delivery phase consisting of unicast or multicast transmissions [23]-[25].

In the simplest form of content caching, the most popular files are stored in local memories close to the end-users; therefore, such requests will be locally served without requiring an extra transmission rate. Depending on the cache size, this approach selects some files randomly for local caching when demand distribution is uniform. Therefore, the network experiences a "*local caching gain*." By leveraging the multicast nature of the shared link, the coded caching strategy with uncoded placement [5] revolutionized conventional results even for caches with distinct demands by reaching a "*global caching gain*." While the local caching gain is proportional to the size of the

user's cache, the global caching gain is proportional to the aggregate size of all the available caches in the network.

The key to increasing multicast opportunities in the coded caching strategy is to involve diverse parts of the library content in the cache placement and store them symmetrically across different users (coding principle) [12][15]. However, in a caching network with non-uniform content popularity demand distribution caching higher popular content is preferable as it significantly reduces the delivery rate (popularity principle) [6][12][15].

Even though the popularity principle reduces delivery rates, it forces caches to store almost the same content and prevents creating multicast opportunities. In this regard, developing a placement strategy that can simultaneously benefit from both local and global caching gains is extremely challenging since these principles conflict with one another.

Figure 1.5 shows the tension between the popularity principal and the coding principle, which are moving in opposite directions.

Figure 1.5: Tension between the popularity principle and the coding principle in cache placement

Over the past few years, various studies have been conducted on developing placement schemes assuming an individual cache for each user under different settings, including different source distributions [5]-[10] and demand specifications [26]-[28], to address this challenge.

Recent studies are focused on another aspect of caching networks that considers shared caches for a group of users. In such settings, considering a shared cache in the access points to be filled with the most useful content during the placement phase leads to transmitting only the small updates during the delivery phase.

According to [5], when multiple caches are available in the network, coded caching rather than replication is the best strategy as it provides multicast opportunities and global caching gain.

However, multicast opportunities cannot be created for the connected users in a single shared cache setting as the framework is not distributed for that group of users. Thus, designing a cache-aided delivery network with shared caches is still being investigated from different aspects.

In this regard, [11] has studied a network consisting of a server and several small base stations, each connected to a shared cache. They proposed a combination of coded-uncoded placement strategies and extracted the expected delivery rate of the system. Also, they formulated the near-optimum partitioning placement for such systems. However, they assumed a homogenous number of users connected to each cache.

Authors in [12] have also considered a network with a non-uniform popularity demand consisting of a server and multiple shared caches under homogenous and heterogenous user behavior. In this sense, they proposed a coded-uncoded placement strategy to consider the trade-off between the coding and popularity principles. They extracted a closed-form expression of the transmission rate by formulating the problem as a constraint optimization problem. They have shown that the proposed placement strategy significantly reduces the transmission rate compared to the pure-coded or pure-uncoded schemes in the literature.

A similar setting with shared caches is described in [13]. They proposed a coded placement strategy that considers the asymmetry in the number of users connected to each cache. They characterized the gain for the two-cache system and then extended it to a larger network. They concluded that coded placement would achieve more gains if the users' connectivity were more asymmetric. It has been shown that their method is more effective than [14], which considers a similar setting with uncoded placement.

Assuming each user can only request a subset of files, [27] analyzes the average rate for three placement schemes considering a heterogeneous user profile. They divided users into groups according to the request pattern of the shared and individual files. They showed that the lowest average rate under a small cache capacity is found when caching, and delivery of shared and individual files are decoupled.

While recent studies presented promising results in a shared caching setting, they have investigated caching networks with independent sources. As discussed in [17]-[19], a library of a caching network could consist of sources with a high degree of similarity in many practical

applications. Any event recordings with common scenes and background, repeated measurements, news updates, different chunks of a video in video summarization, augmented and virtual reality [17]-[19],[30]-[34], and crowdsourced multi-view videos [20] are examples of such applications.

The authors in [17] showed that exploiting the correlation among the sources can be considered leverage to reduce the delivery rate and improve network efficiency during peak hours. They have assumed a system consisting of a server with correlated sources and multiple users where each user is equipped with an individual cache in the network. They proposed a correlation-aware scheme for the placement phase that enables users to store some segments of the files based on their similarity to the rest of the library files. The delivery phase then considers transmitting the compression version of the requested files. They have studied the same system with an alternative for the placement phase in [31], in which a compressed version of some selected segments of each file is stored in each user's cache. They have shown how joint compression of the selected segments prior to the placement phase can reduce load compared to other solutions in the same setup.

A caching network consisting of a library with three correlated files and two receivers, each having an individual cache, is considered in [32]. They have proposed a two-step scheme in which the library files are compressed using Gray-Wyner source coding. Then the encoded segments are treated as independent content using a multiple-request cache-aided coded multicast scheme. They have characterized the rate-memory trade-off and examined the limits for the worst-case rate.

The authors in [33] studied a coded caching scheme considering the correlation among the library files in a network where each user is equipped with an individual cache of the same size. They assumed that each file is divided into some subfiles shared between several users. They defined the level of commonness as the number of files that include a certain subfile and proposed one uncoded and one coded caching scheme for the placement phase, which has been optimized in terms of the achievable delivery rate.

In [34], the same network setup with one server consisting of correlated sources and multiple users, each having an individual cache, is studied for applications with updated versions of dynamic data. They have considered a caching scheme in which users cache content pieces based on popularity and correlation with the rest of the library files. Then they receive compressed

versions of the requested files according to the information distributed across the network and their joint statistics during delivery time.

Despite current advancements in caching networks with correlated sources, all these studies also assumed a setting where each user has been equipped with an individual cache. Thus, the challenges mentioned above have remained unsolved for a shared caching framework. Therefore, a solution is still needed for a network with correlated sources in a shared cache framework, which is the focus of this study.

1.3.2. Data Clustering

As Artificial Intelligence technologies expand and smart sensors and internet-connected devices are used more widely, wireless networks are faced with a large volume of unstructured data. This massive volume of content is highly beneficial in data analysis, social sciences, and many other modern applications if appropriately classified and analyzed in a meaningful way [36][37].

Clustering is a popular unsupervised data analysis technique that captures the natural structure of a dataset and places similar objects into a set of groups to simplify the process of analyzing and understanding information coming from different sources. As a result, objects within a cluster have a higher degree of similarity. Nowadays, clustering analysis methods are widely used in many fields, such as wireless sensor networks, mobile networks, image and video processing, and data summarization [38]. In some applications of wireless sensor networks, having clusters with approximately the same distance is desirable, as it helps to introduce the optimum allocation of the maximum allowable distortion to the receivers and reduce the transmission rate. In most cases, however, there is no prior information specifying the number of clusters, which makes clustering difficult.

Over the years, many optimization algorithms have been proposed to overcome the problems caused by traditional algorithms in cluster analysis. Tabu search algorithm [41], the simulated annealing (SA) algorithm [42], and the particle swarm optimization (PSO) algorithm [43], which is one of the most powerful optimization algorithms for data clustering in complex problems, are some of the well-known examples in this area. Van der Merwe and Engelbrecht [44] have

investigated the capability of several swarm intelligence algorithms in partitioning different types of datasets. They have also proposed a novel approach for clustering different datasets into an optimal number of clusters through an optimization process. In [45], the data clustering problem has been formulated as a single objective problem, and the standard *global best (gbest)* of the PSO algorithm has been used to identify the centroid of the clusters.

Evolutionary algorithms have also been among the most frequently used metaheuristic algorithms for the clustering problem [46]. Hence, different types of this algorithm have been studied in the literature, ranging from a straightforward encoding to a more sophisticated solution similar to Falkenauer's grouping genetic algorithm (GA) [47].

In [48], the automatic data clustering problem has been addressed using a hybrid solution called FAPSO based on an improved firefly algorithm and the particle swarm optimization algorithm. The authors also investigate the applicability of the proposed solution in detecting the correct number of clusters according to the Davis-Bouldin index (DB-index) [49] and the compact-separated index (CS-index) [50] as the validity measure. The proposed algorithm's performance has been evaluated on thirteen benchmark datasets, and it has also been compared to other well-known clustering algorithms. The experimental results indicated that the FAPSO outperforms the comparative studies in most cases in terms of the accuracy of the results.

In [51], an automatic data clustering algorithm using an improved PSO algorithm (ACPSO) has been proposed to address the clustering problem. The proposed algorithm determines the correct number of clusters and adjusts the centroids. The authors have considered the K-means algorithm and a sigmoid function to adjust the cluster centroids and manage the infeasible solutions. This algorithm has been evaluated by considering four benchmark datasets in terms of consistency and accuracy.

Abraham et al. [52] proposed a kernel-based automatic clustering using a modified PSO algorithm. This approach employs a kernel-induced similarity measure instead of the sum of square distances. They believed that using a kernel function in this solution leads to clustering linearly non-separable data into homogenous clusters in a high dimensional feature space transformation. This algorithm has been evaluated by considering five synthetic and three real-life datasets in terms of convergence, accuracy, and robustness.

In [53], Nanda and Panda proposed a multi-objective automatic clustering algorithm called MOIMPSO to classify the actions of 3D human models. This algorithm provides a Pareto optimal archive for automatic clustering problems by considering a developed hybrid evolutionary algorithm immunized PSO and two objective functions. Besides, a single best solution from the Pareto optimal archive has been provided to satisfy the users' requirements. They have also evaluated the proposed algorithm on eleven benchmark datasets in terms of computation time and accuracy.

Liu et al. [54] proposed a solution based on the genetic algorithm with unknown K called AGCUK. They employed the DB-index as the validity measure of clusters. The performance of this algorithm has been evaluated on several artificial and real-life datasets in terms of determining the correct number of clusters and the accuracy of the clustering.

Then, He and Tan [55] proposed a novel two-stage genetic algorithm called TGCA to solve the clustering problem. This algorithm uses the selection and mutation operators of the original genetic algorithm. The TGCA algorithm attempts to gradually reach globally optimal cluster heads by focusing on determining the correct number of clusters for each input. The experimental results on four artificial and seven real-life datasets in this study indicate that this algorithm shows high performance in determining the number of clusters and the accuracy of the clustering solution.

In [56], the application of the differential evaluation (DE) algorithm is described for the clustering problem with an un-labeled large dataset. The proposed algorithm is called the ACDE algorithm and uses an improved differential evaluation algorithm for the data clustering problem. The ACDE algorithm has been evaluated by considering five benchmark datasets via DB-measure and CS-measure. The authors also reported the application of the ACDE algorithm to the automatic segmentation of images.

Then, in [57], a new hybrid algorithm based on differential evaluation and fuzzy clustering called ADEFC is proposed to solve the automatic clustering problem. In this algorithm, the cluster heads are encoded in the vectors. The data points are then assigned to different clusters based on the Xie-Beni index, a validity measure for the clustering validation. The performance of the ADEFC algorithm has been evaluated on two synthetic and two real-life datasets. It has also been compared to the fuzzy C-mean algorithm and the variable-length genetic algorithm based on fuzzy

clustering. The authors indicated that the ADEFC has the capability of being considered for micro-array data clustering.

An improved differential evaluation algorithm with cluster number oscillation called ACDE-O has been proposed in [58]. Since poor initial guesses lead to inefficient clusters, a cluster number oscillation mechanism is used in this algorithm to improve searching and finding more possible clusters. This algorithm's efficiency has been evaluated on three real-life datasets compared to the ACDE algorithm and reported better performance.

Kuo et al. [59] proposed automatic kernel clustering with bee colony optimization (AKC-BCO) to address the weaknesses of the automatic clustering problem in determining the number of clusters and the accuracy of the clustering. The authors employed a kernel function to increase the capability of the clustering algorithm. The performance of the AKC-BCO has been evaluated on several benchmark datasets compared to three other clustering algorithms. The experimental results indicate that the AKC-BCO algorithm demonstrates superior performance in terms of not trapping in local optima, convergence speed, and accurate and stable clustering results.

Then, in [60], Kuo and Zulvia proposed a hybrid solution of an improved artificial bee colony optimization and the K-means algorithm called iABC for the automatic clustering and customer segmentation problems. In this study, the onlooker bee exploration in the original ABC algorithm is improved by guiding their movements to a better location, leading to a better initial centroid in the K-means algorithm. Then, to increase the algorithm's efficiency, only the worst cluster centroid will be improved through an updating process. The experimental results on several benchmark datasets show that the iABC algorithm provides better performance than the classical ABC algorithm. It has been discussed that the average value of the computational time for some datasets is less than the original ABC algorithm. The reason is that the iABC algorithm generates better solutions compared to the original ABC algorithm. However, its performance is not faster than the PSO and GA-based algorithms.

In [61], the harmony search algorithm has been employed by Kumar et al. to present a parameter adaptive harmony search algorithm called ACPAHS for automatic data clustering problems. The number of clusters in the proposed algorithm is determined by using a real-coded variable-length harmony vector. The data points are assigned to clusters according to the developed weighted

Euclidean distance. The authors also reported the application of the proposed algorithm to the automatic segmentation of images. The efficiency of the ACPAHS algorithm has been evaluated on several real-life datasets and compared to four other well-known clustering techniques in terms of the determined number of clusters and the accuracy of the clustering solution.

In [62], the problem of automatic data clustering has been solved based on an evolutionary metaheuristic algorithm known as invasive weed optimization (IWO). This algorithm can perform the clustering task without requiring any prior knowledge of the datasets and employs the genetic algorithm's fitness function as the validity measure. The algorithm's proficiency has been evaluated on nine artificial and four real-life datasets, and the results are compared to three other clustering algorithms. The experimental results have indicated that the IWO algorithm shows great performance in population size and computation time.

Qaddoura et al. in [63] proposed an open-source, cross-platform framework called EvoCluster for data clustering. EvoCluster is a customizable framework that can employ various objective functions in addition to different well-known nature-inspired optimizers developed by other researchers to perform partitional clustering tasks. It can also evaluate the result according to different validity measures such as Purity, Entropy, squared error sum, and other common validity measures. Since this framework covers different algorithms and measures, it can be useful in different applications.

A comprehensive survey on data clustering using metaheuristic algorithms can also be found in [56][64][65][66][67]. We have reviewed many related studies that focus on the clustering problem from different aspects in this section. Since cluster analysis is being exploited in diverse research fields, a unique algorithm cannot be a solution to all clustering scenarios due to the differences in the nature of the patterns and applications. Hence, considering the main purpose of this research, we remain focused on reaching clusters with approximately the same maximum distance while the number of clusters is in accordance with the number of inputs.

Reaching clusters with the same radius can also be discussed under the cluster-level constraint and has many practical applications. The cluster-level constraint considers some available information about the underlying cluster structures in the form of limitations [68][69].

The facility location problem discussed in [70] is similar to the clustering problem with a cluster-level constraint. In this study, the authors propose two heuristic algorithms for the facility location problems that can be interpreted as clustering problems with upper bounds on the radius of the clusters.

In [71], the authors study two types of cluster-level constraints in a search-based agglomerative hierarchical clustering algorithm. This algorithm forms initial partitioning using the must-link constraints and then merges some groups by taking constraints into account to meet the stopping criteria. Finally, they mentioned creating a feasible dendrogram is intractable since solving the clustering problem with unspecified K is NP-complete under these constraints.

One of the other exciting applications using cluster-level constraint approaches is discussed in [72] for a distributed sensor network. In such applications, each sensor has one master node, and the aim is to find balanced clusters of sensor nodes while attempting to minimize the distance between master and sensor nodes. The authors formulated the problem as a minimum cost flow problem and optimally solved it.

Although many optimization algorithms have been used to solve traditional clustering problems, few studies have focused on applying AI-based optimization techniques to solve automatic clustering problems, and none of them concentrate on having clusters with the same maximum distance. Due to the requirement of an appropriate clustering solution in such applications, as part of this research, we proposed an automatic clustering framework using AI-based optimization techniques to create clusters with approximately the same maximum distance without knowing the exact number of clusters.

1.4. Research Contributions

This research studies a cache-aided delivery network in a shared cache framework with correlated content under different settings. We focus on the content placement phase to reduce the delivery rate and the users' expected distortion in the delivery phase of the caching network. The contributions to this dissertation are summarized as follows:

- **Automatic data clustering using AI-based optimization techniques in a general framework:** To extract the most efficient side information for the placement phase, we propose a systematic framework for automatic data clustering using AI-based optimization techniques. Our proposed clustering solution partitions the library content files into compact and well-separated clusters with approximately the same maximum distance without requiring prior knowledge about the exact number of clusters. The proposed solution has been designed in a general framework and can successfully cope with different types of datasets and distance measures. The proposed clustering scheme is highly effective in high-dimensional data and binary datasets with a high degree of similarity.
- **Single shared cache Scenario**: As the first step toward this research, we study the proposed network in a single shared cache scenario. To address the key challenge of content placement, we propose a *correlation-aware clustering scheme (CACS)*, developed based on the general framework, to extract the most efficient side information for the placement phase. This scheme categorizes content files with a high degree of similarity into the same group with approximately the same maximum distance per cluster, considering the distortion constraint of the system. We formulate the expected delivery rate by joint consideration of the rate-distortion function and caching strategy, where the limit for the maximum allowable distortion in the system is determined based on the Lagrange multipliers technique and reverse water-filling algorithm. Our simulation results show a considerable boost in network efficiency compared to legacy caching schemes.
- **Shared cache framework with multiple caches under uniform popularity demand**: We study the proposed network in a shared cache framework with multiple caches where each cache is connected to a different group of users in the network. We utilize the proposed CACS to extract the side information for the entire library along. Then we address the caching placement and delivery scheme in a distributed platform. The placement phase considers splitting the representatives into segments and placing the segments according to codded caching strategy to maintain symmetry and reach the global caching gain. The delivery phase involves constructing the clusters'

representatives and clustered files by transmitting codded multicast messages and refinement segments. Moreover, we introduce the optimum library partitioning formulated to minimize the worst-case delivery rate in the system.

- **Shared cache framework with multiple caches under non-uniform popularity demand**: We propose a cache-aided delivery network in a shared cache framework with correlated content under non-uniform popularity demand distribution. To address the content placement challenge, we propose a *popularity-based correlation-aware clustering scheme (PB-CACS)* that considers both the popularity and similarity of library content to extract the most efficient side information for the entire library. Then, we consider a hybrid [3] placement, in which the popular side information is stored completely in all caches, and the clusters' side information is split among different caches. In the delivery phase, coded multicast messages and refinement segments are transmitted to construct the requested cluster representatives and clustered files. Finally, we discuss the trade-offs between the cache size and the delivery rate in our system.

1.5. Organization of Dissertation

This thesis is organized as follows. Chapter 2 fully represents the fundamentals and theoretical backgrounds behind the proposed cache-aided content delivery network in this research. Chapter 3 introduces the proposed automatic clustering solution in a general framework. In Chapter 4, our proposed solution to the cache-aided delivery network in a single shared cache is presented. Chapter 5 introduces the cache-aided delivery network in a shared framework with multiple caches under uniform demand distribution. Chapter 6 discusses our proposed solution considering the non-uniform demand popularity, and finally, Chapter 7 represents the conclusions and directions for future works.

[3] Hybrid placement in this model means that the placement strategy benefits from two models of placement simultaneously; some content is fully cached into the memory, while others are cached only in segments according to the coded caching strategy.

Chapter 2

Background

This chapter introduces the theoretical concepts and the fundamental background used in this research for the proposed cache-aided delivery network under different settings.

2.1. Centralized vs. Decentralized Caching

Cache-aided networks are categorized into two main classes: *centralized* [5] and *decentralized* [9] caching. A centralized cache-aided network assumes that all users, and consequently caches, are available during both phases of caching; therefore, they can all be counted on for the placement phase. However, a decentralized caching network is designed for applications where some of the

users are not available during the placement phase (For example, due to the mobility of users) and cannot participate in the placement phase. In this research, we consider a centralized approach for the proposed network.

2.1.1. Centralized Coded Caching Strategy with Uncoded Placement

The centralized coded caching strategy with uncoded placement [5] has emerged as a major breakthrough to combat the high delivery data rate requirement and improve the quality of experience (QoE) for the end-users in delivery networks. This strategy aims to minimize the peak load on the broadcast link through the caching and delivery scheme.

This strategy assumes a network consisting of a single server with access to m independent library files and N users, where each user is equipped with an individual cache of size M files. It is assumed that the server is connected to the users through a shared error-free broadcast link, and files follow a uniform popularity demand.

This scheme splits each content file into $\binom{N}{T}$ nonoverlapping subfiles, where $T = NM/m$ and caches each subfile in a distinct group of T users in the placement phase. In the delivery phase, one coded message is sent to each subset of $T + 1$ users by the server. The coded message is the bitwise XOR of all $T + 1$ request subfiles. Therefore, the delivery rate to serve all users with one request for any $1 < M < m$ is calculated as:

$$R(M) = N\left(1 - \frac{M}{m}\right) min\left(\frac{1}{1+N\,M/m}, \frac{m}{N}\right) \tag{2.1}$$

Where $N\left(1 - \frac{M}{m}\right)$ is known as the local caching gain and $\frac{1}{1+ N\,M/m}$ is known as the global caching gain.

2.2. Placement and Delivery Phases Strategies

Content placement strategy identifies selected content for placing into the cache during the placement phase. Generally, two kinds of placement strategies are discussed in studies: *uncoded placement* and *coded placement* [73][74]. In the uncoded placement strategy, some actual files or

subfiles are placed into the caches without manipulation according to the cache size. While in the coded strategy, the library files are divided into subfiles and processed by a certain coding method. Then the coded segments are placed into the caches according to the cache size.

A *content replacement* is another placement strategy that replaces the old content with updated content. In this regard, a frequency-based strategy called *least frequently used (LFU)* and a recency-strategy called *least recently used (LRU)* are the most famous replacement strategies in studies [6][75].

The *caching delivery strategy* determines how requested files are delivered to receivers during the delivery phase considering the placement strategy. The *linear network coding (NC)* [77][78] or equivalently *index coding (IC)* technique [22][79] is typically used to create multicast opportunities and reduce bandwidth usage in the delivery phase. In cases where multicasting cannot improve the unicast rate, unicasting can also be considered [5][26][72].

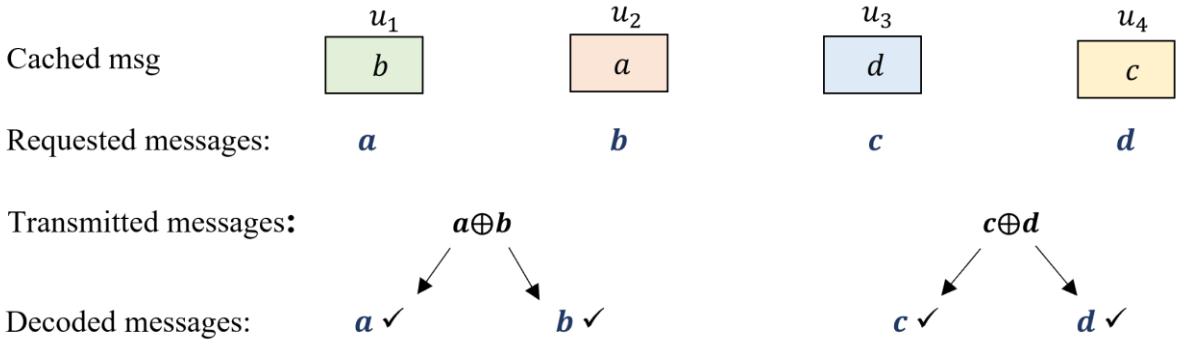
As the content placement and delivery strategies directly affect each other, jointly designing placement and delivery strategies is another point of view to improve network efficiency [15][17][22][72]. This strategy aims to optimize both the placement and delivery phases simultaneously; hence, the placement phase will be designed in a way to minimize the delivery rate.

2.3. Index Coding Technique

Birk and Kol introduced index coding [22][79] as a transmission technique for a noiseless broadcast channel consisting of a transmitter and multiple receivers. In this technique, the transmitter wishes to deliver multiple packets to the receivers over a shared error-free broadcast link, where each receiver has already stored some packets in its own cache and demands a subset of the remaining packets. The goal is to minimize the required transmissions so that all users can decode the desired packets reliably with their own side information and received packets.

Example 2.1: Consider a simple network with four users u_1, u_2, u_3, u_4 each with an individual cache, requesting messages $a, b, c, d,$ respectively. Assume user u_1 has message b in its cache, user u_2 has message a in its cache, user u_3 has message d in its cache, and user u_4 has message c

in its cache. As an alternative to transmitting all four messages a, b, c, d separately, according to the index coding technique, we can transmit just two coded messages $a \oplus b$ and $c \oplus d$, which reduces the delivery rate by $1/2$. The coded messages are the result of bitwise XOR operation between the two desired messages denoted by $\oplus$.



The IC technique has recently gained significant attention in the delivery phase of caching problems. The difference between caching and IC problems is the available side information in the receivers; In the IC, the cached side information is fixed for the problem, while in the caching scenarios, the cached content is not predefined and should be properly designed during the placement phase. Moreover, in IC problems, demands are also assumed to be fixed for each receiver, while in caching scenario, one might be interested in all possible demands.

2.4. Popularity Demand Distribution

Generally, cache-aided delivery networks can be investigated under two content popularity distributions: uniform content popularity and non-uniform content popularity.

In uniform content popularity demand, the assumption is that all library content files are requested according to a uniform distribution, and users have no preference in choosing one file over the others [5]. The traditional approach for the placement phase under the uniform popularity is randomly placing some of the content into the cache according to the memory size. In such cases, the worst-case demand delivery rate is usually evaluated as a performance criterion.

Non-uniform content popularity demand is of great importance in the caching literature as it is more likely to occur in the real world. According to this model, only a small number of files are

highly requested by a large number of users, and the rest of the files will be demanded only from time to time in the network. In this sense, different popularity models have been considered in the literature [18][44], in which *Zipf distribution* is discussed as one of the most famous models in multimedia content popularity.

The Zipf distribution models the relative popularity of a few population numbers and the relative obscurity of others. Based on this distribution, $q_i \sim \text{Zipf}(\xi, m)$, the probability of requesting the i^{th} file among all m library content files with the parameter ξ is modeled as follows.

$$q_i = \frac{1/i^{\xi}}{\sum_{j=1}^{m} 1/j^{\xi}} \tag{2.2}$$

Lower values of parameter γ indicate a more uniform distribution, while high values indicate that a large number of users are interested in a small number of files.

Conventionally, caches are allocated to the highest popular content under the non-uniform content popularity demand. Dividing files into groups with similar probability [6], categorizing library files into only two groups of popular and non-popular content files [10], and the multi-level popularity model [19] are the most well-known approaches used in designing caching schemes under non-uniform popularity demand.

2.5. Distributed Source Coding with Side Information at the Decoder

As mentioned earlier, we have an information-theoretic perspective on the caching scheme and formulate this problem in the context of distributed source coding (DSC) with side information at the decoder.

DSC considers independently compressing and jointly decompressing the library sources by exploiting mutual correlation among them. These sources should be statistically dependent but physically separated [80]. The basic idea of the distributed source coding with a decoder side information problem is compressing the sources conditionally according to the available side information in the decoder so that less delivery rate is needed for the reconstruction of the sources at the decoder [81]. Lossless DSC considers discrete sources and perfect reconstruction at the

decoder, while lossy DSC allows for some distortion in the decoder, and the problem turns to a rate-distortion function.

2.5.1. Lossless Source Coding

The Slepian-Wolf (SW) coding is the simplest case of lossless distributed source coding, which considers the separate compression and joint lossless decompression of correlated discrete sources.

Slepian-Wolf Theorem: Let $\mathcal{U}$ and $\mathcal{V}$ be two correlated random sources drawn independently and identically distributed, i.i.d, from joint distribution $\mathcal{P}(\mathcal{u}, \mathcal{v})$. Therefore, the achievable rate region for compressing $\mathcal{U}$ and $\mathcal{V}$ with compressing rate $\mathcal{R}_{\mathcal{u}}$ and $\mathcal{R}_{\mathcal{v}}$ respectively is as follows.

$$\mathcal{R}_{\mathcal{u}} \geq H(\mathcal{U}|\mathcal{V})$$
$$\mathcal{R}_{\mathcal{v}} \geq H(\mathcal{V}|\mathcal{U})$$
$$\mathcal{R} = \mathcal{R}_{\mathcal{u}} + \mathcal{R}_{\mathcal{v}} \geq H(\mathcal{U}, \mathcal{V}) \tag{2.3}$$

It is shown that if we compress the first source up to its entropy, the second source can then be compressed as low as its conditional entropy given the first source and vice versa.

Figure 2.1 illustrates this achievable rate region. As we can see, the SW rate region has two corner points $(H(\mathcal{U}), H(\mathcal{V}|\mathcal{U}))$ and $(H(\mathcal{U}|\mathcal{V}), H(\mathcal{V}))$.

The asymmetric SW coding scheme is a scheme that attempts to approach one of these two corner points. In Asymmetric coding, a source is compressed to its entropy rate while the second source is compressed at the minimum possible rate. Therefore, during the decompression process, the first source is decompressed, and then the second source is decompressed with the help of the first source. This point of view is equivalent to the source coding with decoder side information case where a source is already available at the decoder, and the other source is compressed to the minimum possible rate.

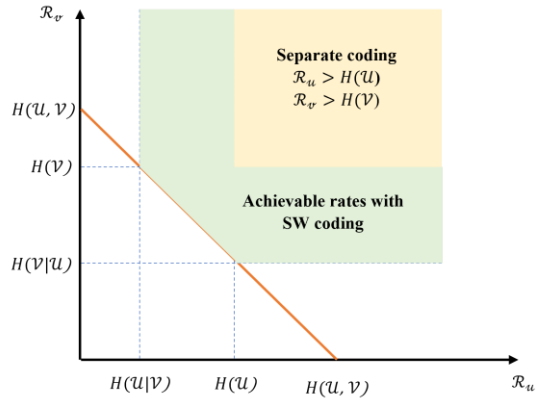


Figure 2.1: The achievable rate region in SW coding

2.5.2. Lossy Source Coding

Wyner-Ziv (WZ) coding generalizes SW coding from the lossless to the lossy case, which is similar to the generalization of the classic source coding problem to the rate-distortion problem [80][81].

Assuming $\mathcal{U}$ and $\mathcal{V}$ be two joint i.i.d sources, the WZ problem aims to compress a block of $\mathcal{U}$, lossily and recover an estimate $\hat{\mathcal{U}}$ at the decoder with the help of $\mathcal{V}$.

The fidelity of $\hat{\mathcal{U}}$ from $\mathcal{U}$ is characterized using a distortion measure $d(.,.)$. In this regard, we are interested in finding the smallest rate required to ensure the average distortion $\mathbb{E}[d(\mathcal{U},\hat{\mathcal{U}})]$ is smaller than some predetermined value $\boldsymbol{\delta}$.

Figure 2.2 shows the setup of the WZ problem.

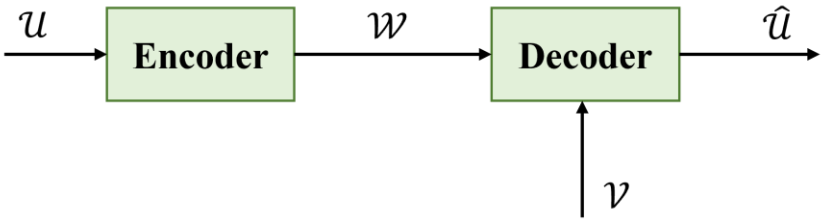


Figure 2.2: The setup of the Wyner-Ziv problem

Then, the rate-distortion function $R_{WZ}(D)$ can be found by solving the optimization problem as follows:

$$R^*(D) \triangleq \min_{q(\mathcal{W}|\mathcal{U}),\hat{\mathcal{U}}} I(\mathcal{U};\mathcal{W}|\mathcal{V}) \tag{2.4}$$

$$s.t. \quad \mathbb{E}[d(\mathcal{U},\hat{\mathcal{U}})] \leq \boldsymbol{\delta}$$

In other words, we have $R_{WZ}(D) = R^*(D)$.

2.6. AI-Based Optimization Techniques

Most clustering problems can be described as typical optimization problem that tries to optimize a criterion and specify the clustering quality [66]. Therefore, using heuristic and meta-heuristic AI techniques in automatic clustering has been proposed as a superb solution to work with different applications and datasets and overcome the weaknesses of the classical approaches in recent years. Heuristic and meta-heuristic AI techniques are optimization techniques that do not require offline training but could provide an optimal or near-optimal solution for the formulated objective function, considering the system objectives and constraints. A comprehensive survey on data clustering using metaheuristic algorithms can also be found in [56][64][65][66][67].

Since assigning a cluster number to a data point has a discrete nature, we have adopted binary metaheuristics optimization algorithms in the AI-optimizer module of the proposed clustering scheme. To this end, we have considered four of the most well-known algorithms, including the binary bat algorithm (BBA) [82], binary particle swarm optimization (BPSO) [83], binary genetic algorithm [84], and binary dragonfly algorithm (BDA) [85].

2.6.1. Binary Bat Algorithm

The bat algorithm is a heuristic optimization algorithm inspired by the echolocation behavior of bats, initially proposed to solve problems with continuous search spaces [86]. In [82], a binary version of this algorithm known as BBA is developed, which has a discrete search space and applies to binary problems.

In the original Bat algorithm, each artificial bat will update the velocity, position, and frequency vector according to (2.5), (2.6), and (2.7) during the course of iterations t.

$$V_i(t+1) = V_i(t) + (\varphi_i(t) - Gbest)Fr_i \tag{2.5}$$

$$\varphi_i(t+1) = \varphi_i(t) + V_i(t+1) \tag{2.6}$$

$$Fr_i = Fr_{min} + (Fr_{max} - Fr_{min})\vartheta \tag{2.7}$$

where Fr_{min} and Fr_{max} are the minimum and maximum frequencies, and ϑ is a random number from a uniform distribution in $[0,1]$. Each bat for a local search carries out the random walk based on the optimal solution as given in (2.8).

$$\varphi_{new} = \varphi_{old} + \varepsilon A^t \tag{2.8}$$

Where ε is a random number from a uniform distribution in $[-1,1]$, φ_{old} is a solution randomly selected from the current optimal solution, and A shows the loudness of emitted sound. The loudness of the pulse emission of the bat A_i and the velocity r_i will be updated as follows.

$$A_i(t+1) = \alpha A_i(t) \tag{2.9}$$

$$r_i(t+1) = r_i(0)[1 - \exp(-\gamma t)] \tag{2.10}$$

Where $0 < \alpha < 1$ and $\gamma > 0$ are constant. In the binary version of the bat algorithm, a V-shaped transfer function is used to define a transformation probability from 0 to 1 for a position vector and oblige the particles with high velocity to switch their positions. Then a new position-updating equation is also used to update the position of the particle. These equations are given in (2.11) and (2.12), respectively.

$$V\left(v_i^k(t)\right) = \left|\frac{2}{\pi}\arctan(\frac{\pi}{2}v_i^k(t))\right| \tag{2.11}$$

$$\varphi_i^k(t+1) = \begin{cases} (\varphi_i^k(t))^{-1} & if\ rand < V\left(v_i^k(t+1)\right) \\ \varphi_i^k(t) & if\ \ rand \geq V\left(v_i^k(t+1)\right) \end{cases} \tag{2.12}$$

where $\varphi_i^k(t)$ and $v_i^k(t)$ indicate the position and velocity of i^{th} particle at t^{th} iteration in k^{th} dimension and $(\varphi_i^k(t))^{-1}$ is the complement of $\varphi_i^k(t)$.

2.6.2. Binary Particle Swarm Optimization

Particle swarm optimization is one of the most widely used evolutionary due to its simplicity and low computation cost in solving a wide range of problems. The binary version of the PSO was originally proposed by Kennedy and Eberhart [87]. In this algorithm, each particle will update the position and velocity vectors as guidance for its movement in the discrete search space. This algorithm utilizes a transfer function to map the velocity vector to a probability vector, and then the probability vector is used to update the position of particles. This transfer function is a sigmoid function that returns 0.5 when the velocity equals 0. At this point, the probability of changing a variable is at the highest level. By contrast, as the velocity value goes toward the positive or negative infinity, the transfer function returns values close to 1 or 0. The related equations are given in (2.13), (2.14), and (2.15).

$$V_i(t+1) = \mathcal{w}V_i(t) + \mathfrak{c}_1\mathfrak{r}_1\left(\varphi_{best(i)}(t) - \varphi_i(t)\right) + \mathfrak{c}_2\mathfrak{r}_2\left(\varphi_{bestglobal(i)}(t) - \varphi_i(t)\right) \quad (2.13)$$

$$\varphi_i(t+1) = \begin{cases} 1 & if\ r < T\big(V_i(t+1)\big) \\ 0 & if\ \ r \geq T\big(V_i(t+1)\big) \end{cases} \quad (2.14)$$

$$T\big(V_i(t+1)\big) = \frac{1}{1+e^{-\left(V_i(t+1)\right)}} \quad (2.15)$$

Over the years, several studies [88][89][90] have improved the original BPSO by enhancing the updating position formula, as the original BPSO suffered from local minima trapping. The authors of [83] discussed how the transfer function affects the efficiency of the algorithm in the binary version for mapping the continuous search space to the discrete search space. They improved the performance and convergence speed of the algorithm by proposing two families of transfer functions and showed excellent performance with a V-shaped transfer function which has been used utilized in our study as well.

2.6.3. Binary Genetic Algorithm

The genetic algorithm is a population-based optimization method that formulates the problem as an objective function using the concept of genes from biology. The GA initializes by generating the initial population; then, three genetic operators are performed (i.e., selection, crossover, and

mutation) on some selected individuals from the current population iteratively to reach either a near-optimal solution or reach a predefined number of generations [91].

For BGA [92], the decision variables are represented as chromosomes $= [G_1, G_2, \dots, G_n]$, while each gene g_i in the chromosome is encoded by a binary vector of length B, in which 0 means not selected feature and 1 means selected feature for the decoding process. BGA uses a group of chromosomes known as population (Npop) and goes through a process of updates or generations to evolve. The cost function can be represented as $\boldsymbol{f}$(chromosome), which needs to be minimized or maximized based on the desired settings. The next step is natural selection and mating, in which a pair of chromosomes are selected to mate to generate further offspring in each generation. BGA first selects a set of fittest chromosomes (Nkeep) from which the parents will be selected, and all other chromosomes are discarded and replaced by generated offsprings. Once the offsprings is generated, the mutation occurs as a random process to mutate the bits in the population. If the new population is ineffective in terms of cost, it will be discarded in the next generation or iteration.

As a final step, BGA checks for convergence conditions as soon as the mutated population is generated. If the convergence conditions are met, the process stops; otherwise, BGA iterates through multiple generations and repeats the decoding, cost calculation, mating, and mutation process.

2.6.4. Binary Dragonfly Algorithm

While the dragonfly algorithm had been proposed for problems with continuous search spaces, the binary version of this algorithm (BDA) [85] has been developed for feature selection and demonstrated excellent performance in discrete search spaces.

DA adopted the step and position vectors to solve the optimization problem. In a continuous search space, the position of dragonflies is updated by adding the step vector to the previous position.

$$\Delta\varphi_{t+1} = (\mathcal{s}\mathcal{J}_i + a\mathcal{T}_i + c\mathcal{C}_i + \mathcal{g}\mathcal{G}_i + e\mathcal{E}_i) + \mathcal{w}\Delta\varphi_{t+1} \tag{2.16}$$

$$\varphi_{t+1} = \varphi_t + \Delta\varphi_{t+1} \tag{2.17}$$

Where s describes the separation weight, $\mathcal{J}_i$ represents the separation of the i^{th} individual, a is the alignment weight, $\mathcal{T}_i$ is the alignment of the i^{th} individual, c is the cohesion weight, $\mathcal{C}_i$ is the cohesion of the i^{th} individual, g shows the food factor, $\mathcal{G}_i$ is the food source of the i^{th} individual, e indicates the enemy factor, $\mathcal{E}_i$ is the enemy position of the i^{th} individual, w is the inertia weight, and finally, t is the number of iterations. The swarming behavior and mathematical model of this algorithm are given as follows:

$$\mathcal{J}_i = \sum_{j=1}^{n} \varphi - \varphi_j \tag{2.18}$$

φ_j represents the j^{th} neighboring individual of the φ position, and n is the neighborhood size.

$$\mathcal{T}_i = \sum_{j=1}^{n} V_j / n \tag{2.19}$$

V_j represents the j^{th} neighboring individual velocity.

$$\mathcal{C}_i = \frac{\sum_{j=1}^{n} V_j}{n} - \varphi \tag{2.20}$$

$$\mathcal{G}_i = \varphi^+ - \varphi \tag{2.21}$$

$$\mathcal{E}_i = \varphi^- + \varphi \tag{2.22}$$

φ represents the current individual's position, φ^+ describes the position of the food source and φ^- represents the enemy positions.

In a binary search space, the following equations should be used to update the step vector:

$$\varphi_{t+1} = \begin{cases} \beta\varphi_t, & if\ x < T(\Delta\varphi_{t+1}) \\ \varphi_t, & if\ x \geq T(\Delta\varphi_{t+1}) \end{cases} \tag{2.23}$$

β indicates a random number in the range [0,1], and $T(\Delta\varphi_{t+1})$ is as follows:

$$T(\Delta\varphi) = \left| \frac{\Delta\varphi}{\sqrt{\Delta\varphi^2 + 1}} \right| \tag{2.24}$$

Chapter 3

Automatic Clustering Using AI-Based Optimization Techniques in a General Framework

This chapter provides an extensive evaluation and comprehensive discussion of the proposed clustering solution in a general framework. This chapter is important because it validates and facilitates the development of the other two clustering schemes presented as part of the placement phase of the proposed caching networks in the following chapters.

Cluster analysis using AI-based optimization techniques has earned increasing popularity due to the excellent performance of such solutions in finding high-quality clusters in complex problems. This chapter proposes a novel framework for automatic data clustering, creating clusters with approximately the same maximum distance using meta-heuristic AI-based optimization

techniques. Such techniques do not require offline training but could provide an optimal or near-optimal solution, considering the system objectives and constraints.

The inherent problem with clustering using such algorithms is having a huge search space. Therefore, we have also proposed a binary encoding scheme for the particle representation in the AI-optimizer module to alleviate this problem. The proposed clustering solution does not require prior knowledge of the number of clusters and evolves based on re-clustering, merging, and modifying the small clusters to compensate for the distortion deviation between groups with different sizes.

The performance of this framework has been evaluated over a wide range of synthetic, real-life, and higher dimensional datasets first by considering four different binary optimization algorithms in the AI-optimizer module. Furthermore, utilizing internal validity measures, it has also been evaluated in comparison with multiple classical and new clustering solutions as well as two other automatic clustering techniques in continuous search space. The experimental results show the proposed solution is highly efficient in creating well-separated and compact clusters with approximately the same distance in almost all datasets. Moreover, the application of the proposed framework to the correlated binary dataset is also investigated as a case study.

Besides having a dynamic number of clusters, simplicity, customizability, and flexibility in adding extra conditions to the proposed solution are the advantages of the proposed framework.

3.1. Introduction

Over the years, many clustering algorithms have been proposed for different data types, algorithms, and applications. In general, partitional and hierarchal clustering algorithms are the two main approaches for cluster analysis in the literature [38],[66],[94]-[99]. The partitional algorithms can be performed in two different modes: hard and fuzzy. In the hard-clustering algorithm, each pattern only belongs to one cluster, while in the fuzzy algorithm, different membership degrees are assigned to each pattern in a group. Partitional algorithms are often non-deterministic. These algorithms require a priori knowledge of the number of clusters [38][94].

The well-known K-means method is a famous example of this type that is initialized with a random solution and tries to partition a given dataset into a predefined number of clusters. Although it is an efficient and robust algorithm, the results strongly depend on the initial random guesses. Furthermore, this algorithm computes the local minimum and cannot guarantee the global optimum solution [95]-[97].

On the other hand, the hierarchical clustering algorithm develops a tree-based data structure to reach the exact number of clusters by splitting the tree at different levels. This algorithm creates a more deterministic and flexible mechanism than the partitional approach. However, the final grouping is static since each cluster's assigned objects cannot move to other groups. Besides, hierarchical clustering exhibits poor performance when the separation of overlapping clusters is carried out [38][96][98]. In this case, the fuzzy clustering algorithm can express the overlapping nature of clusters much better than the hierarchical model. The fuzzy C-mean (FCM) algorithm [99] is a widely used clustering algorithm that divides objects into a C number of clusters. This number is determined by trial and error in advance. Although FCM is a powerful method, it is highly dependent on initial guesses; thus, it can be easily entrapped within local optima [38][66][99].

Being sensitive to initial solutions, entrapping in local optima, and requiring a priori knowledge of the number of clusters in most classical clustering algorithms made it challenging to handle this task in some applications. On the other hand, most real-world clustering problems can be described as a typical optimization problem that tries to optimize a criterion and specify the clustering quality [9]. Therefore, meta-heuristic AI-based techniques have been proposed as superb solutions in automatic clustering to overcome the weaknesses of the classical approaches in recent years [48][68].

In a nutshell, metaheuristic algorithms can address the clustering problem via two main approaches. In one approach, the optimization algorithm tries to find the optimum centroids for the desired dataset. Then, it divides data points into predetermined clusters according to the identified centroids. This approach requires a priori knowledge of the exact number of clusters. In another approach, the optimization algorithm assigns each data point directly to a group and tries to reach the best possible solution over the course of iterations. The second approach is more convenient since the number of clusters is not required to be predefined, and a sufficient number

of clusters evolve during the entire clustering process. However, it suffers from a huge search space, which makes the overall solution extremely difficult without providing additional insights.

The advantage of not requiring predefined information in the second approach motivated us to develop our clustering problem based on this solution. In this regard, we have formulated the clustering problem as an AI-based optimization problem and adopted a dynamic range of clusters in accordance with the input data to improve the convergence speed.

3.2. Data Clustering Problem Formulation

Let $\boldsymbol{\mathcal{A}}_{m\times l}$ represents a set of m patterns, each having l features. The data clustering problem considers a given dataset $\boldsymbol{\mathcal{A}}_{m\times l}$ and attempts to partition it into K clusters $C = [c_1, c_2, \dots, c_K]$ such that $K \leq m$. In such partitioning, the following properties should be maintained:

- $\forall i \in \{1,2,3, \dots, K\}, c_i \neq \emptyset$
- $\bigcup_{i=1}^{K} c_i = \boldsymbol{\mathcal{A}}$
- $c_i \cap c_j = \emptyset, \forall i, j \in \{1,2, \dots, K\}$, and $i \neq j$

Given dataset $\boldsymbol{\mathcal{A}}_{m\times l}$, the fitness function $\boldsymbol{f}$ is defined as a partitioning adequacy measure to quantify the goodness of a partition based on the similarity of the patterns. Therefore, the clustering problem turns into finding optimal partitions among all other feasible solutions [94].

Euclidean distance is one of the most popular choices for distance measures to evaluate the similarity between data points in clustering problems. The Euclidean distance between any two l-dimensional data points is given by:

$$d(\mathcal{A}_l, \mathcal{A}'_l) = \sqrt{\sum_{i=1}^{l} (\mathcal{A}_l^i - \mathcal{A}_l'^i)^2} \tag{3.1}$$

3.3. Proposed Data Clustering Framework

The main idea of designing the proposed framework using AI-based optimization techniques is to reach a sufficient number of compact and well-separated clusters with approximately the same maximum distance within each group without requiring prior knowledge of the number of clusters.

To overcome the inconvenience of having a huge search space as well as not having a priori knowledge of the adequate number of clusters, we propose a binary encoding scheme for the particle representation of the optimization algorithm in a predetermined range $[K_{min}, K_{max}]$, where m is the total number of data points and $K_{min} = 1$, $K_{max} = \lfloor\sqrt{m}\rfloor$.

K_{max} is considered as $\lfloor\sqrt{m}\rfloor$ according to similar assumptions discussed in [100] for a clustering solution with the fuzzy C-mean model and [66] for automatic clustering using metaheuristic algorithms. They have mentioned that $K = \sqrt{m}$ usually provides a decent answer for such clustering solutions as a rule of thumb. Hence, we applied this assumption to our dynamic range of clusters in our proposed solution. We observed that considering this assumption along with the proposed binary encoding scheme provides excellent results in our proposed framework.

Overall, the proposed framework operates in two main steps: First, each data point is directly equipped with an initial cluster number. Consequently, a primary clustering solution is formed at this step. The obtained clusters will then be re-clustered, merged, and modified based on some conditions to compensate for the distortion deviation between groups with different sizes and improve the result according to the desired objectives.

Following is a detailed description of the AI-based optimizer module, the binary encoding scheme, and the objective function.

3.3.1. AI-Based Optimizer Module

The first step is to formulate the clustering problem in a way that is amenable to an AI optimizer module. The second step is utilizing AI-based optimization techniques in the optimizer module to find the near-optimum solution. In this way, the optimizer intelligently encompasses the process of finding a decent number of clusters with the same maximum distance in accordance with the dataset considering the defined constraint in the system.

The AI-based optimizer module stands at the highest level of the proposed approach and considers the clustering problem as a black box that needs to be optimized iteratively. In this regard, the automatic clustering problem is formulated as a single objective optimization function based on the desired goals and system constraints. The AI-based optimizer takes a binary vector

as input and calculates the corresponding output according to some merit factors while minimizing this function during the entire process. In other words, the optimizer module checks combinations of the input to determine which input vector will yield the minimum output of the function.

A wide range of binary optimization algorithms can be utilized for the AI-based optimizer module. This research considers the BPSO algorithm, the BBA algorithm, the BGA algorithm, and the BDA algorithm due to their excellent performance discussed in the literature.

3.3.2. Binary Encoding Scheme

Similar to other iterative metaheuristic algorithms [66], our approach requires a representation of a solution that is directly related to the objective function to be optimized. Therefore, a binary encoding scheme for particle representation has also been proposed in this research. The proposed binary encoding scheme assigns binary vectors with the length of $m \times L$ bits to each particle in the optimizer, where L is the number of required bits to address each cluster number and calculated as

$$L = \log_2 K_{max} \tag{3.2}$$

The suggested binary vectors are considered the candidate solution and evaluated by the optimizer module iterations. For this purpose, every L bit of this vector is converted to the decimal equivalent in sequential order. These decimal numbers are assigned to data points as the initial cluster numbers.

Example 3.1: A simple example of the proposed encoding scheme is shown in Figure 3.1 for $m = 16$ data points. In this case, $K_{max} = 4$, and $L = 2$; therefore, each particle is represented by a binary vector of length $m \times L = 32$ bits.

Then every 2-bit of the binary vector is converted to the decimal equivalent in sequential order. The decimal equivalents are considered the initial cluster numbers and will be assigned to the data points in the same order to form the initial clustering.

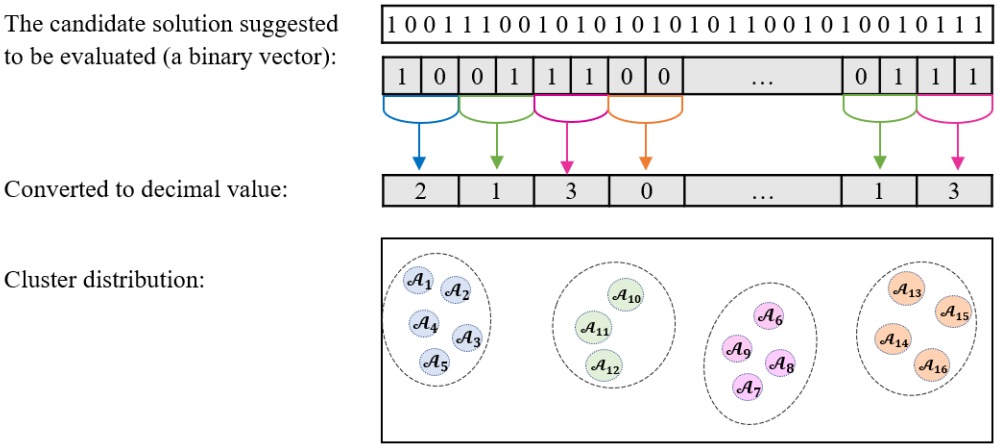


Figure 3.1: The steps of assigning a cluster number to each data point by the proposed binary encoding scheme in the general clustering framework

3.3.3. Objective Function

We formulate the data clustering problem as a single objective optimization problem with objective function $\boldsymbol{f}$ that needs to be minimized iteratively over the set of all feasible ways of clustering a given dataset.

The set of all feasible ways of clustering a given dataset is defined as $\psi = \{C^1, C^2, \dots, C^{S(m,K)}\}$. $S(m, K)$ is the total number of combinations in assigning m files into K clusters [66] obtained by the Stirling number of the second kind [118].

$$S(m, K) = \frac{1}{K!}\sum_{i=0}^{K} (-1)^i \binom{K}{i}(K - i)^m \tag{3.3}$$

Once the initial clustering solution is formed, a representative is selected for each cluster. The representative is defined as the nearest data point to the current centroid in each group, where the centroid is the mean of all data points in each cluster. Then, data points will be re-clustered according to the identified representatives of the clusters. Consequently, the representatives will also be updated according to the recent changes. Re-clustering and updating the representatives will be repeated several times until no further changes are observed.

Although the achieved clustering solution up to this stage consists of separated clusters, it is not well-organized yet and needs to be revised to satisfy our goals. To reach clusters with approximately the same maximum distance while keeping the number of clusters less than K_{max}, the distortion gap between the largest and the smallest clusters must be compensated. In this regard, the smallest clusters are identified and marked as defective, meaning they should be merged appropriately with adjacent clusters. To merge clusters, we compare the maximum distances of all clusters with the cluster with the largest maximum distance across all clusters according to inequality (3.4). Clusters that do not satisfy this inequality are considered defective and merged with adjacent clusters.

$$\delta^{c_i} \leq 0.9 \underset{C}{\text{Max}} \{\delta^{c_i}\} \tag{3.4}$$

where δ^{c_i} is the maximum distance within the cluster $c_i \in C$, for $i \in \{1, \dots, K\}$, and $C \in \psi$.

The process of re-clustering and updating representatives is repeated to stabilize the clustering in this step. Once no change happens in the clusters, that solution is considered the best result for the corresponding input vector, and the following merit factors are calculated by the optimizer:

- K_{max} as the maximum number of clusters
- δ^{c_i} as the maximum intra-cluster distance within each cluster $c_i \in C$
- Δ_C as the distortion deviation over all clusters in a clustering solution C
- $\bar{d}_C^{max}$ as the average of the maximum distance over all clusters in a clustering solution C
- $\bar{E}_C^{min}$ as the average of minimum inter-cluster distances in a clustering solution C

The definition of these parameters is given as follows.

Definition 1: Considering δ^{c_i} as the maximum distance in each cluster c_i, then $\bar{d}_C^{max}$ is defined as the average of the maximum distance of all clusters and is calculated as:

$$\bar{d}_C^{max} = \frac{1}{K} \sum_{i=1}^{K} \delta^{c_i} \tag{3.5}$$

where $i \in \{1,2, \dots, K\}$ and $C \in \psi$.

Definition 2: The distance deviation Δ_C is defined as the difference between the maximum and the minimum value of maximum distance over all clusters and is defined as:

$$\Delta_C = \max_{c \in C}\{\delta^{c_i}\} - \min_{c \in C}\{\delta^{c_j}\} \tag{3.6}$$

where $i, j \in \{1,2, \dots, K\}\ j \neq i$, and $C \in \psi$.

Definition 3: Let $E_{ij}^{\mathcal{A}}$ be the inter-cluster distance between data point $\mathcal{A}$ within the cluster i to the cluster-head j, where $\mathcal{A} \in c_i,\ i \in \{1,2, \dots, K\},\ j \in \{1,2, \dots, K-1\}$ and $i \neq j$.

$E_{ij}^{\mathcal{A}}$ is calculated for all data points in all clusters to determine the inter-cluster distances. The minimum of $E_{ij}^{\mathcal{A}}$ is determined for each cluster and denoted by $E_{c_i}^{min}$.

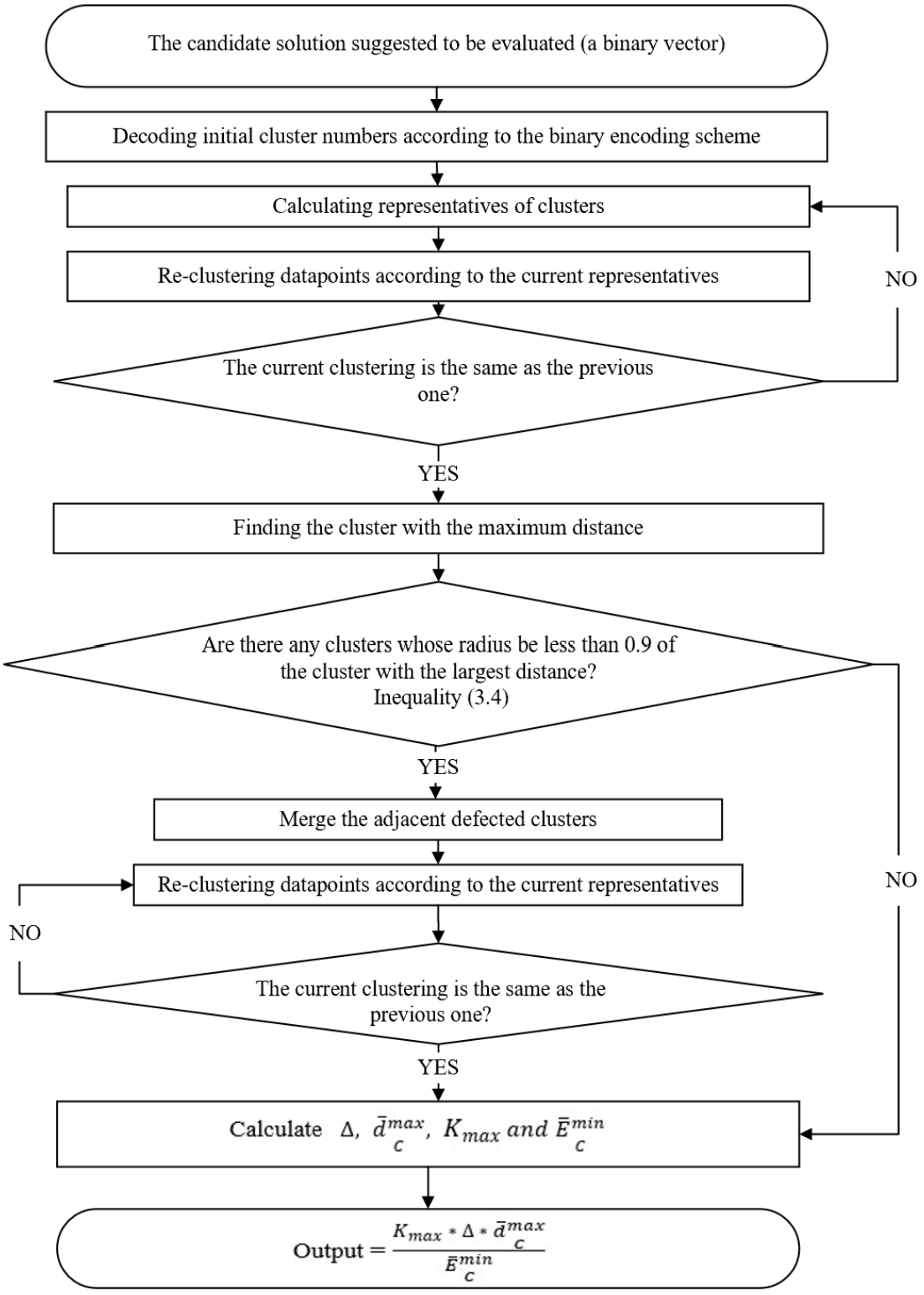
Then, $\bar{E}_{C}^{min}$ is calculated as the average of the minimum inter-cluster distances over all clusters given by:

$$\bar{E}_{C}^{min} = \frac{1}{K}\sum_{i=1}^{K} E_{c_i}^{min} \tag{3.7}$$

Finally, the output of the function $\boldsymbol{f}$ will be calculated according to the defined parameters, and the goal is to minimize it.

$$\boldsymbol{f} = \frac{K_{max}\ \Delta_C\ \bar{d}_{C}^{max}}{\bar{E}_{C}^{min}} \tag{3.8}$$

Flowchart 3.1 summarizes all the steps within the objective function.



Flowchart 3.1: The objective function of the proposed clustering framework

Also, the following example illustrates the process of evolving clusters and migration of data points to other groups during different stages.

Example 3.2. This evaluation has been performed on dataset R15 with $m = 320$ points, which is one of the standard datasets in UCI machine learning [101] illustrated in Figure (3.2-a). We have considered the BBA optimization algorithm with 100 agents and 200 iterations for this experiment. Figure (3.2-b) shows the result of initial clustering, where an initial cluster number is assigned to each data point. Figures (3.2-c) and (3.2-d) show the clustering result after two re-clustering levels. Although the clusters are distinguishable in this stage, the smallest and largest clusters still have a considerable size difference. During Stages (e) to (h), the process of merging and modifying the small clusters is performed, and the result is evaluated according to output. As can be seen, the proposed framework provides a sufficient number of compact and well-separated clusters with relatively close values for maximum distance.

Figure 3.3 illustrates the convergence curve of this example. The convergence curve is a very common tool to qualitatively present the results of a single-objective optimization algorithm.

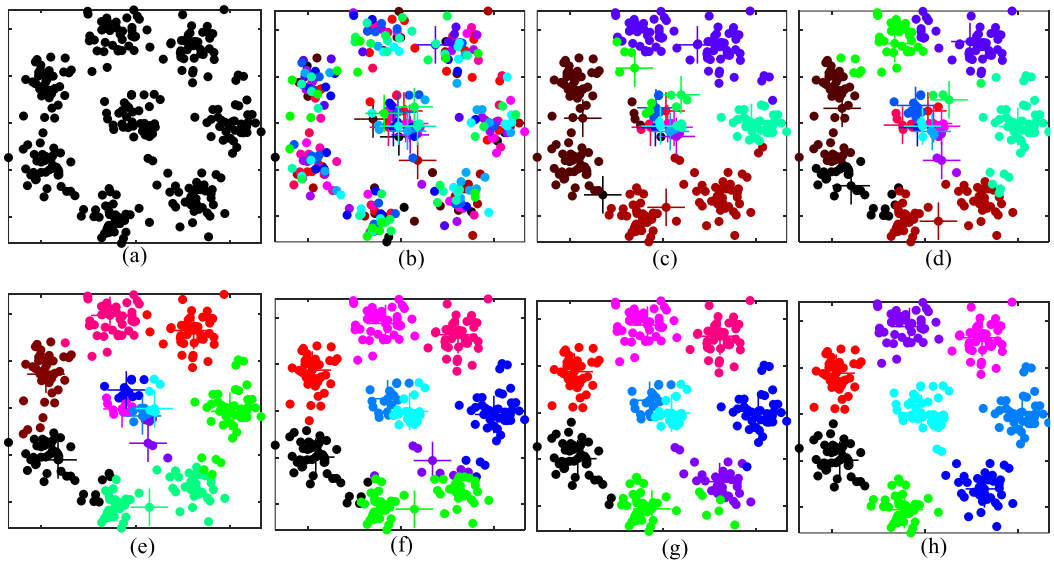


Figure 3.2: The process of evolving clusters and migration of data points to other clusters during different stages of the proposed clustering framework.

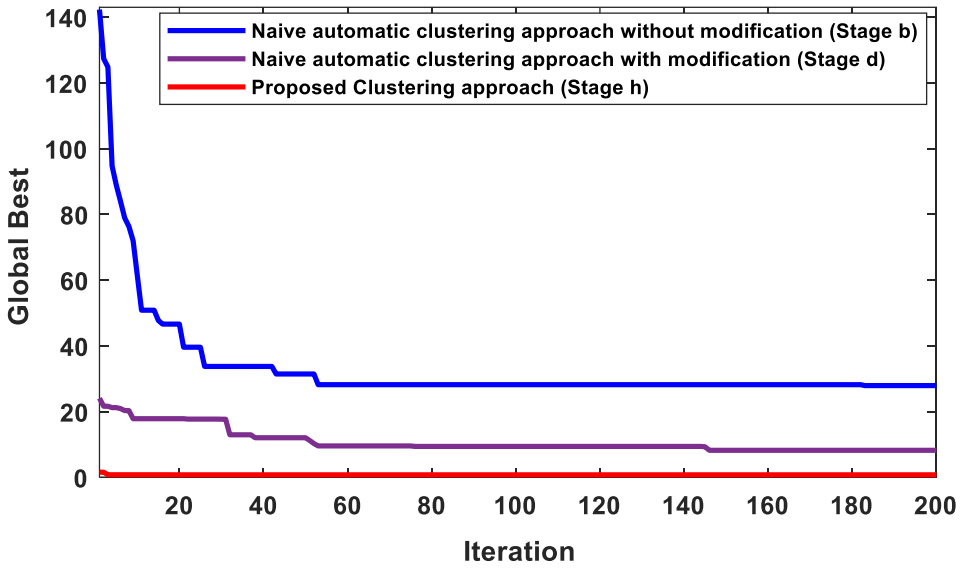


Figure 3.3: The convergence curve of Example 3.2 in three different stages

As shown, the proposed framework demonstrates high convergence speed in a small number of iterations when we perform the merging and modifying steps to compensate for the distortion gap.

3.4. Results and Discussion

In this section, the performance of the proposed automatic clustering framework is evaluated on twenty-four benchmark datasets. The details of the used datasets and the parameter settings for this performance evaluation are also presented here. Then, simulation results and comparative studies are discussed.

3.4.1. Dataset and Validity Measure

This experiment collects twenty-four different synthetic and real-world datasets from the UCI Machine learning [101] and KEEL repositories [102]. This collection includes datasets with different shapes, densities, and dimensions ranging from 2 to 512, with diverse data points from 106 to 3100.

Table 3.1 describes the summary of the datasets.

Table 3.1: Summary of used datasets with different features

Dataset		Number of features	Number of points	Number of Clusters
Shape dataset	Aggregation	2	788	6
	Compound	2	399	6
	D31	2	3100	31
	Flame	2	240	2
	Jain	2	373	2
	Pathbased	2	300	3
	R15	2	600	15
	Spiral	2	312	3
Real-World dataset	Appendicitis	7	106	2
	Dermatology	34	358	6
	Ecoli	7	336	8
	Glass	9	214	7
	Haberman	3	306	2
	Housevotes	16	232	2
	Ionosphere	33	351	2
	Iris	4	150	3
	Segment	19	2310	7
	Vehicle	18	846	4
	Wdbc	30	569	2
	Wine	13	178	3
Higher-dimensional dataset	Dime064	64	1024	16
	Dime128	128	1024	16
	Dime256	256	1024	16
	Dime512	512	1024	16

We have used internal validity measures to examine the quality of the proposed clustering scheme. In this regard, we have calculated the sum of intra-cluster and inter-cluster distance validity measures [38][94] to ensure the compactness and the separation of the clusters. We have also evaluated the proposed framework using the DB-index validity measure [49], which describes

a trade-off in maximizing intra-cluster compactness and inter-cluster separation. In the DB-index measure, the intra-cluster distances in the i^{th} cluster and the inter-cluster distances between cluster i^{th} and j^{th} are defined as follows:

$$\Upsilon_{i,\tau} = \left[\frac{1}{n_i} \sum_{\mathcal{A} \in c_i} \| \mathcal{A} - m_i \|_2^{\tau}\right]^{1/\tau} \tag{3.9}$$

where n_i is the number of elements in the i^{th} cluster denoted by c_i. $\overrightarrow{m_i}$ is the i^{th} cluster centroid and τ is an integer, and both $\tau, t \geq 1$.

$$\Lambda_{i,j,t} = \left\{\sum_{\ell=1}^{l} |m_{i,\ell} - m_{j,\ell}|^t\right\}^{1/t} = \| m_i - m_j \|_t \tag{3.10}$$

$$\Phi_{i,\tau,t} = \underset{j \in K, j \neq i}{\text{Max}} \left\{\frac{\Upsilon_{i,\tau} + \Upsilon_{j,\tau}}{\Lambda_{i,j,t}}\right\} \tag{3.11}$$

Finally, the DB-index measure is given by (3.12).

$$DB(K) = \frac{1}{K} \sum_{i=1}^{K} \Phi_{i,\tau,t} \tag{3.12}$$

The smaller the value of the DB-index measure, the better the compactness and the separation.

3.4.2. Experimental Results and Parameter Settings

The experiments have been carried out on a PC with Windows 10 Professional 64-bit operating system, an Intel(R) Core ™ i7-10700K processor, and 48 GB RAM using MATLAB software 2021 b. We have also used MATLAB packages[4], Yarpize packages[5], and NPIR source code[6] [63] with its required python packages for the comparative study during this experiment. Besides, the IBM SPSS Version 27 has been used for performing the statistical analysis test.

The comparison results are explained in two parts. The first set of results describes the effect of considering different binary optimization algorithms on the proposed framework. In this regard, we have evaluated the proposed framework performance using BBA, BPSO, BGA, and BDA algorithms in the optimizer module and presented the result for different cases.

[4] https ://www.mathworks.com/products/matlab.html.
[5] http://yarpiz.com/64/ypml101-evolutionary-clustering
[6] http://evo-ml.com/2019/10/28/npir/

The parameter settings for the used algorithms are represented in Table 3.2.

In order to evaluate each optimizer module, twenty independent trials are conducted on each dataset. Then the best and the worst cost, the average cost, and the standard deviation are reported. These comparison results are described in Tables 3.3, 3.4, and 3.5. The best entries are shown in boldface in all tables.

Table 3.2: Parameter configuration of the utilized optimization algorithms in the proposed framework and two of the comparative studies

BBA		BPSO	
Parameter	Value	Parameter	Value
$Pop\ size$	100	$Pop\ size$	100
$Iter_{Max}$	200	$Iter_{Max}$	200
K_{Max}	$\sqrt{m}$	K_{Max}	$\sqrt{m}$
Fr_{min}, Fr_{max}	0.2	c_1, c_2	2
A	0.25	w	0.85
r	0.5		
BGA		BDA	
Parameter	Value	Parameter	Value
$Pop\ size$	$Pop\ size$	$Pop\ size$	100
$Iter_{Max}$	$Iter_{Max}$	$Iter_{Max}$	200
K_{Max}	K_{Max}	K_{Max}	$\sqrt{m}$
$Crossover$	0.8	w_{min}, w_{max}	0.2, 0.9
$Mutation$	0.001	α	0.01
		β	0.09
GCUK		DCPSO	
Parameter	Value	Parameter	Value
$Pop\ size$	100	$Pop\ size$	100
$Iter_{Max}$	200	$Iter_{Max}$	200
K_{Max}	$\sqrt{m}$	K_{Max}	$\sqrt{m}$
$Crossover$	0.8	c_1, c_2	1.494
$Mutation$	0.001	$P_{initialize}$	0.75

Table 3.3: Comparison results for the shape dataset with considering BBA, BPSO, BGA, and BDA in the optimizer module in the proposed method

Datasets	BBA				BPSO			
	Best	Worst	Mean	Std	Best	Worst	Mean	Std
Aggregation	5.62	6.76	**6.58**	0.25	6.76	8.16	6.92	0.32
Compound	12.69	22.47	18.70	2.91	16.63	24.00	20.56	2.33
D31	2.16	6.74	**4.87**	1.39	3.66	7.88	6.59	0.78
Flame	3.59	6.78	**5.32**	0.66	5.12	6.81	6.01	0.45
Jain	6.04	7.28	6.21	0.35	6.04	7.12	6.18	0.32
Pathbased	8.57	12.52	**11.34**	1.08	8.95	12.51	11.75	0.94
R15	3.00	3.83	3.53	0.21	2.63	3.81	3.55	0.28
Spiral	11.18	16.68	14.65	1.59	12.67	17.02	14.97	1.32
Datasets	BGA				BDA			
	Best	Worst	Mean	Std	Best	Worst	Mean	Std
Aggregation	6.47	7.25	6.68	0.16	6.47	7.66	6.77	0.22
Compound	12.08	21.86	**16.03**	2.56	12.44	24.05	16.95	2.83
D31	2.18	6.73	5.45	1.35	2.16	7.40	6.22	1.15
Flame	3.50	6.22	5.34	0.68	5.43	6.42	5.92	0.33
Jain	5.92	8.87	6.50	0.90	5.75	6.32	**6.04**	0.09
Pathbased	10.05	12.14	11.60	0.58	9.59	12.44	11.83	0.63
R15	1.54	3.80	**2.95**	0.79	1.54	3.81	3.45	0.51
Spiral	11.12	15.92	**14.16**	1.32	12.77	16.67	14.92	1.11

Table 3.4: Comparison results for the real-life dataset with considering BBA, and BPSO in the optimizer module in the proposed method

Datasets	BBA				BPSO			
	Best	Worst	Mean	Std	Best	Worst	Mean	Std
Appendicitis	1.66	2.28	1.95	0.13	2.23	2.23	2.08	0.11
Dermatology	26.69	41.53	36.35	4.16	43.59	43.59	39.49	2.80
Ecoli	367.51	484.57	446.22	29.70	501.28	501.28	466.99	27.12
Glass	7.78	8.14	8.12	0.08	8.42	8.42	8.18	0.10
Haberman	44.92	46.86	45.79	0.99	46.86	46.86	**45.40**	0.86
Housevotes	2.24	3.79	2.95	0.38	5.46	5.46	3.59	0.51
Ionosphere	34.98	48.35	41.42	3.10	50.72	50.72	45.40	3.38
Iris	2.01	3.20	2.68	0.40	3.33	3.33	2.97	0.22
Segment	145.10	148.73	145.29	0.81	166.36	166.36	146.71	4.81
Vehicle	140.68	141.73	141.48	0.41	226.51	226.51	151.81	23.68
Wdbc	1861.93	1869.16	**1864.61**	3.45	1858.82	1867.87	**1864.54**	2.398
Wine	139.27	296.37	210.93	47.56	296.37	296.37	244.10	32.78

Table 3.5: Comparison results for the Real-life dataset with considering BGA, and BDA in the optimizer module in the proposed method

Datasets	BGA				BDA			
	Best	Worst	Mean	Std	Best	Worst	Mean	Std
Appendicitis	1.66	2.24	2.03	0.14	1.66	2.28	2.00	0.17
Dermatology	35.45	44.13	38.01	2.03	31.16	44.04	38.07	3.29
Ecoli	388.90	487.49	447.70	26.49	404.82	524.54	462.88	28.41
Glass	8.14	8.42	8.17	0.08	8.14	8.420	8.20	0.11
Haberman	44.92	46.86	**45.40**	0.86	44.92	46.86	45.60	0.94
Housevotes	2.03	3.33	**2.81**	0.38	2.35	3.54	2.99	0.34
Ionosphere	36.05	47.56	42.15	3.58	36.80	48.65	43.04	3.41
Iris	2.19	3.20	2.79	0.30	2.24	3.52	2.83	0.38
Segment	145.10	168.93	150.73	9.61	145.10	168.93	150.55	9.68
Vehicle	140.68	141.73	**141.33**	0.48	140.68	226.84	148.11	21.23
Wdbc	1809.57	1928.53	1870.38	28.08	1857.94	1871.54	**1864.76**	3.451
Wine	139.27	296.37	215.24	52.86	116.02	296.37	**178.40**	37.98

Table 3.6: Comparison results for the higher-dimensional dataset with considering BBA, BPSO, BGA, and BDA in the optimizer module in the proposed Method

Datasets	BBA				BPSO			
	Best	Worst	Mean	Std	Best	Worst	Mean	Std
Dime064	13.06	13.07	**13.06**	0.00	13.10	13.18	13.14	0.01
Dime128	12.95	13.20	13.06	0.05	13.01	13.14	13.07	0.03
Dime256	3.56	5.15	**4.24**	0.40	3.84	4.84	4.36	0.29
Dime512	3.02	4.78	**3.95**	0.48	3.55	4.87	4.14	0.38
Datasets	BGA				BDA			
	Best	Worst	Mean	Std	Best	Worst	Mean	Std
Dime064	13.14	13.14	13.14	0.00	13.10	13.10	13.10	0.00
Dime128	12.93	13.12	**13.03**	0.04	13.68	13.81	13.74	0.03
Dime256	3.74	4.92	4.25	0.27	4.12	5.07	4.61	0.29
Dime512	3.28	5.14	4.26	0.50	3.05	5.54	4.27	0.62

From the results, we can see the four binary optimization algorithms have reached a very competitive result. However, the performance of the proposed framework by considering the BBA algorithm in the optimizer module is more significant in most datasets.

We have also provided the convergence curve of the shape and the real-world datasets by considering all four optimizer modules in Figures 3.4 to 3.10. The convergence curve is a useful tool to visualize how an algorithm improved the global best (*gbest)* as the first best path iteratively to reach the global optimum solution starting from a random solution. In our model, the global best represents the minimum of the objective function across all search agents in each iteration.

In all these curves, the optimizer that reaches the minimum cost after passing all iterations is suitable for that specific dataset and can provide the best solution.

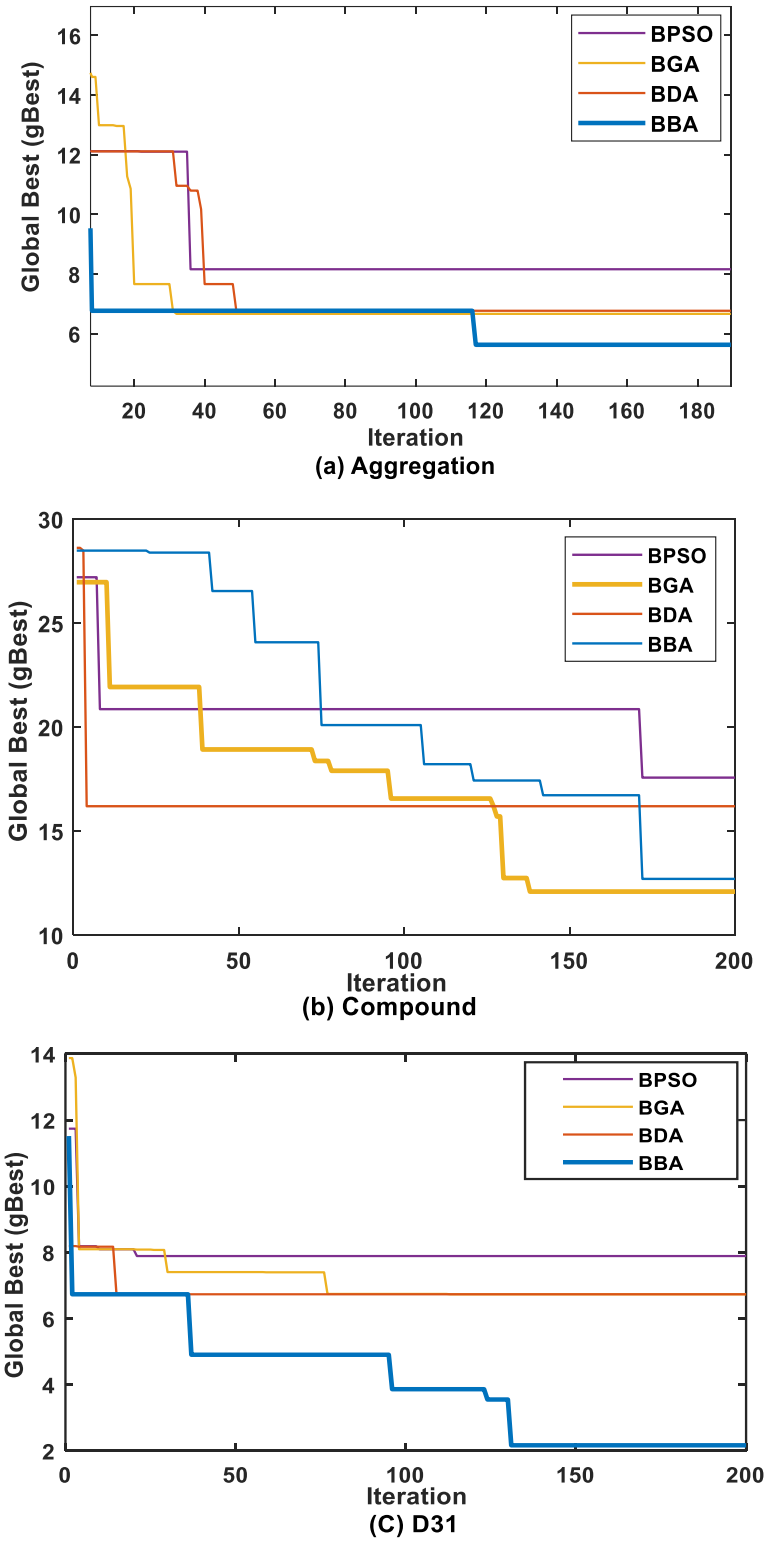


Figure 3.4: Convergence curve of (a) aggregation, (b) compound, and (c) D31 datasets considering BBA, BPSO, BGA, and BDA in the optimizer module.

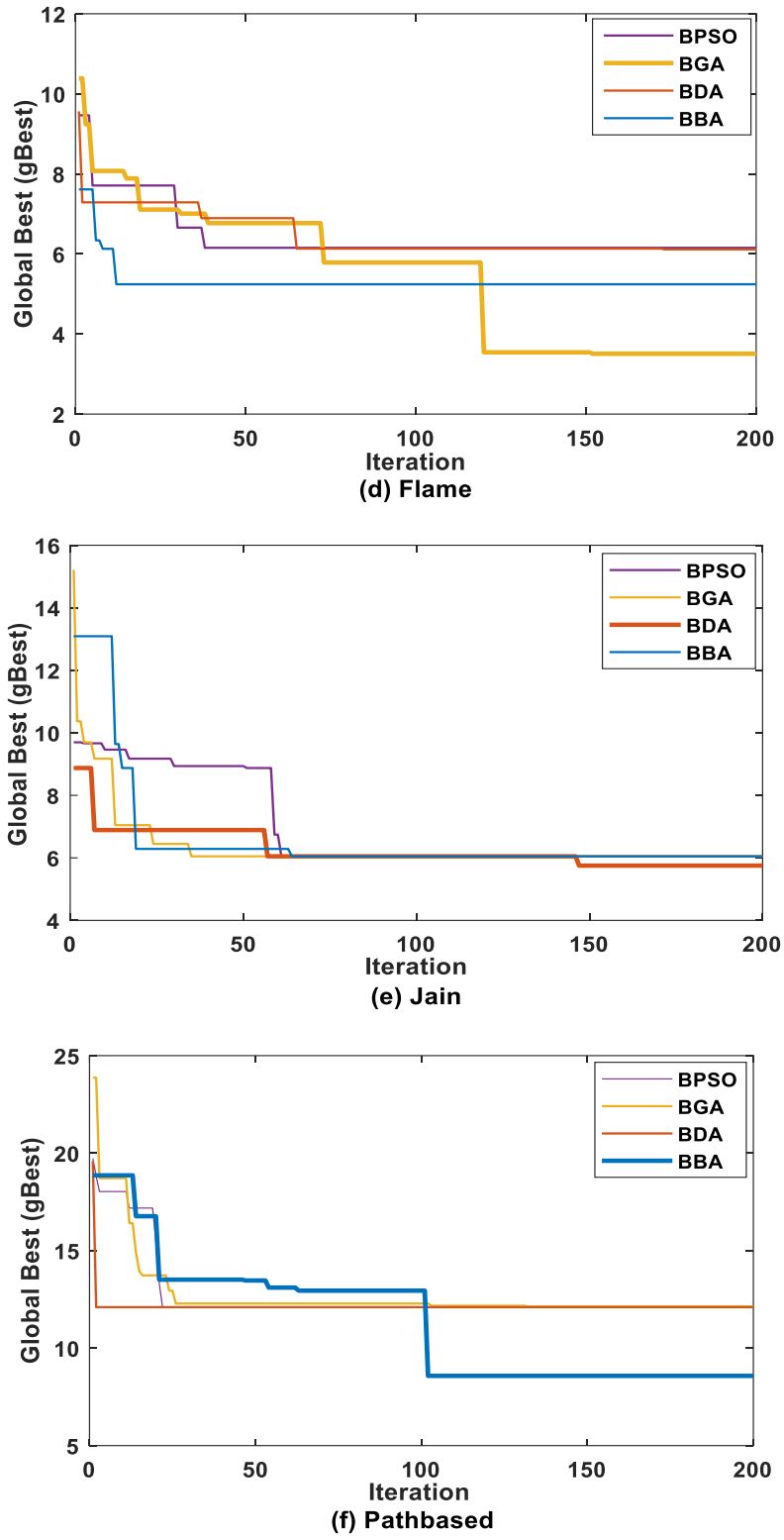


Figure 3.5: Convergence curve of (d) flame, (e) Jain, and (f) Pathbased datasets considering BBA, BPSO, BGA, and BDA in the optimizer module.

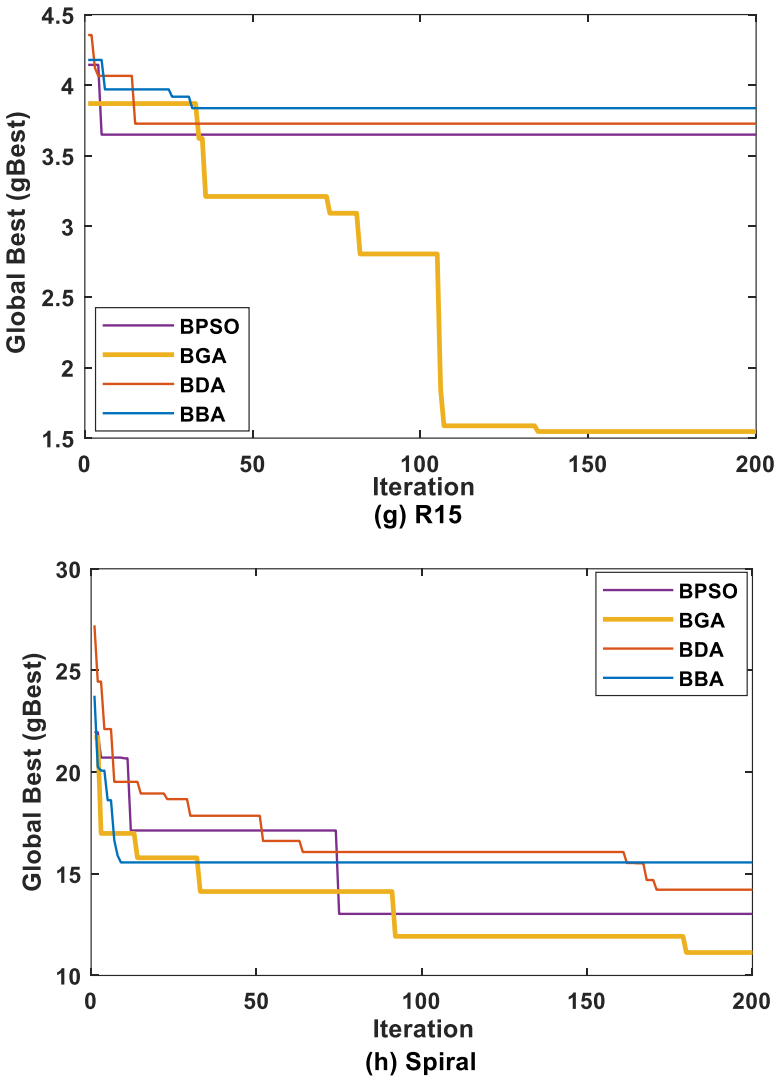


Figure 3.6: Convergence curve of (g) R15, (h) spiral datasets considering BBA, BPSO, BGA, and BDA in the optimizer module.

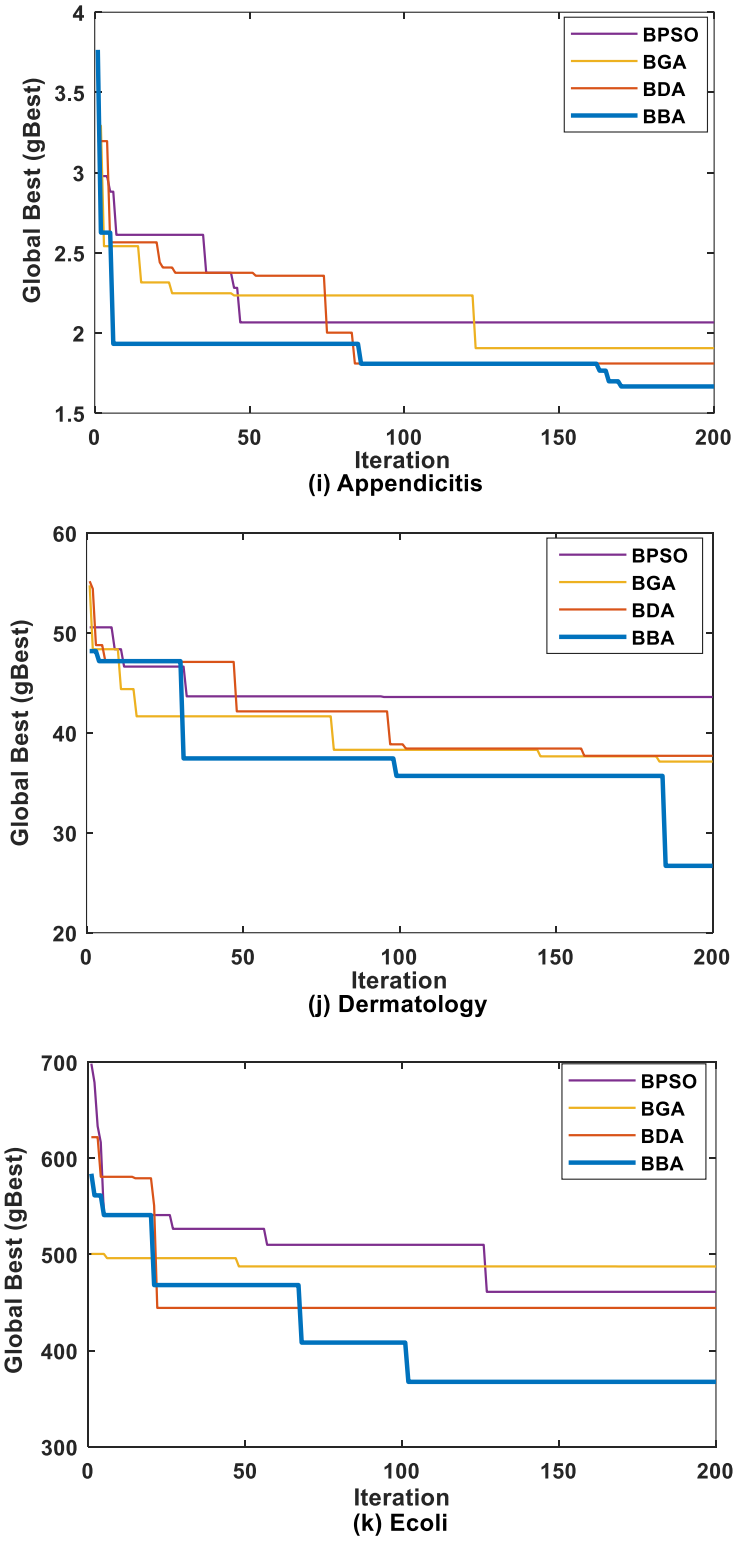


Figure 3.7: Convergence curve of (i) appendicitis, (j) dermatology, and (k)Ecoli datasets considering BBA, BPSO, BGA, and BDA in the optimizer module.

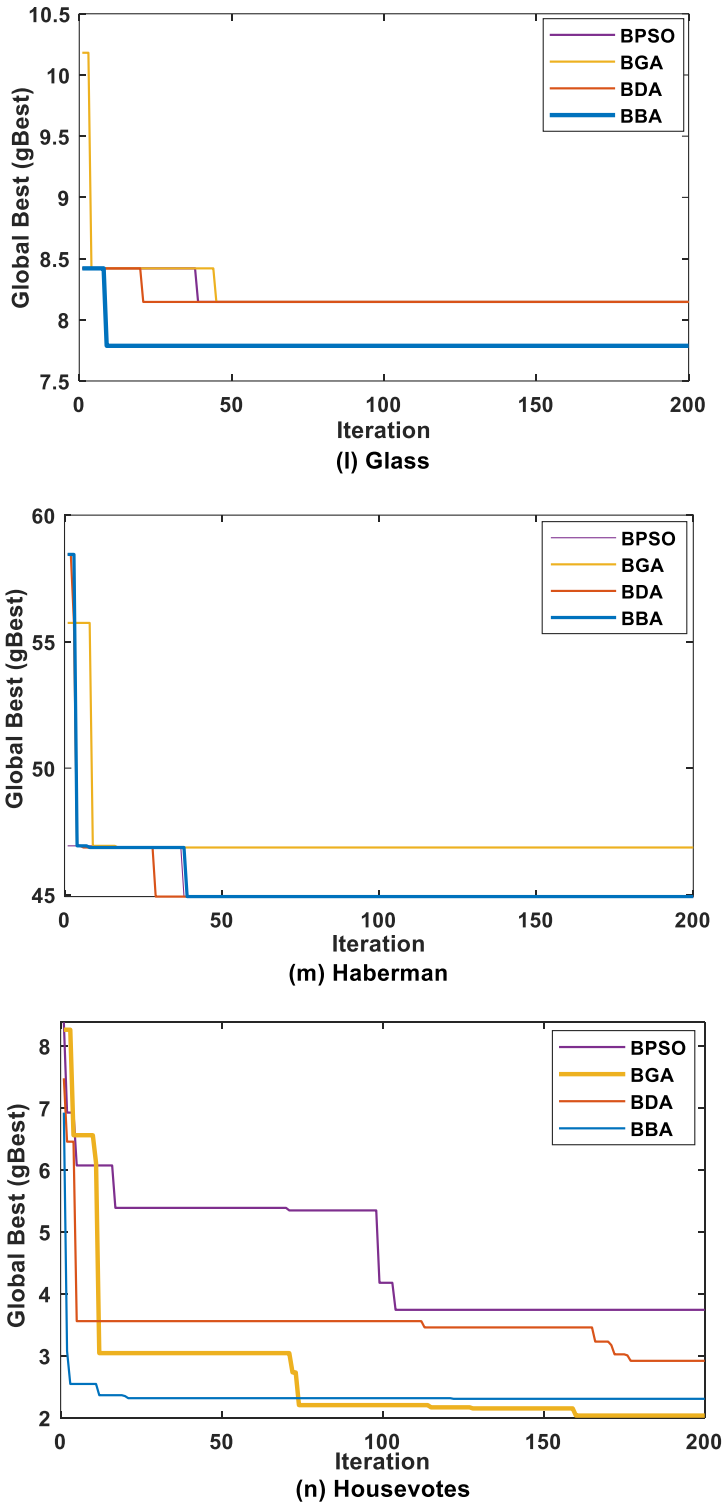


Figure 3.8: Convergence curve of (l) glass, (m) Haberman, and (n) housevotes datasets considering BBA, BPSO, BGA, and BDA in the optimizer module.

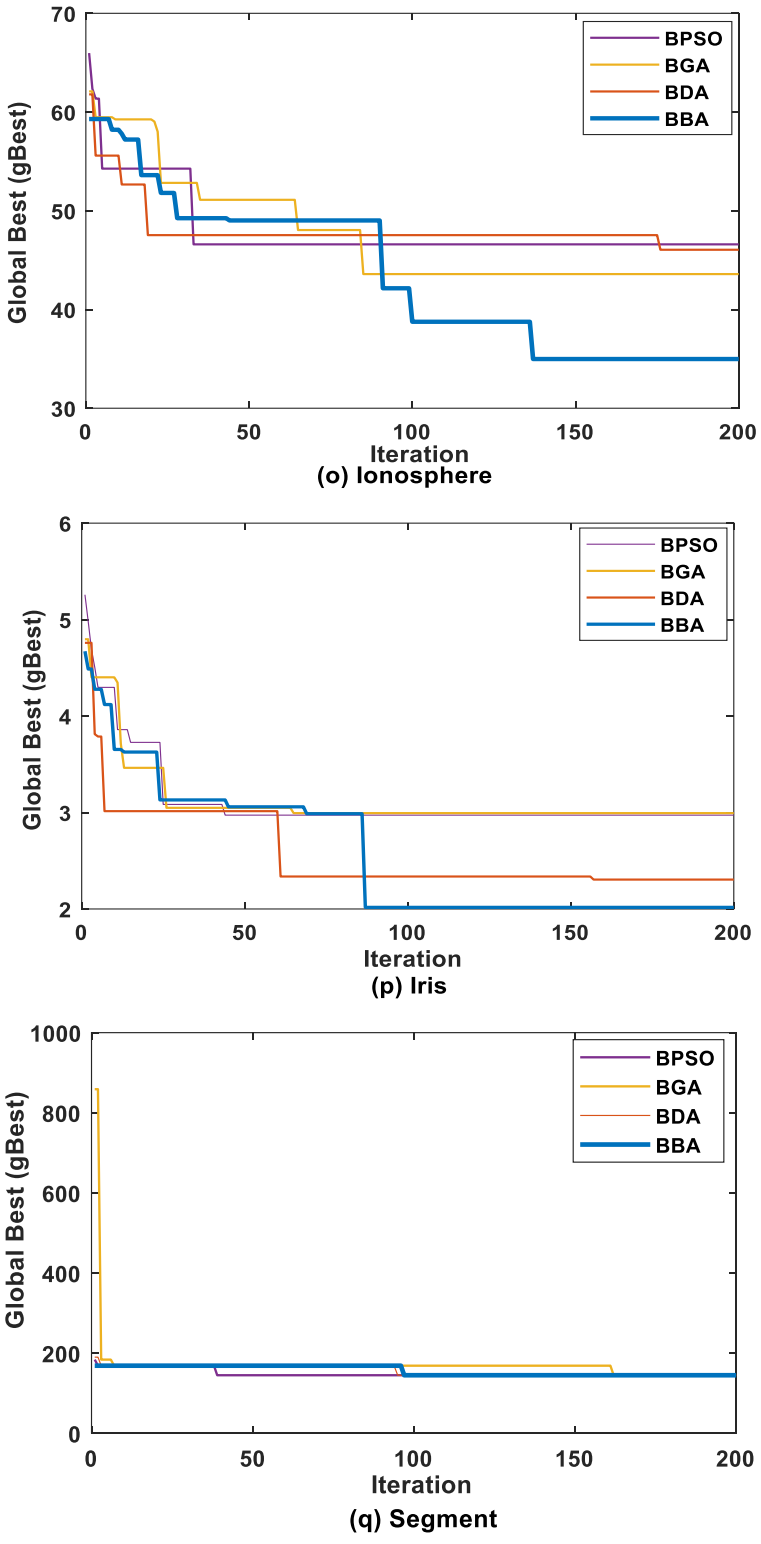


Figure 3.9: Convergence curve of (o) ionosphere, (p) iris, and (q) segment datasets considering BBA, BPSO, BGA, and BDA in the optimizer module.

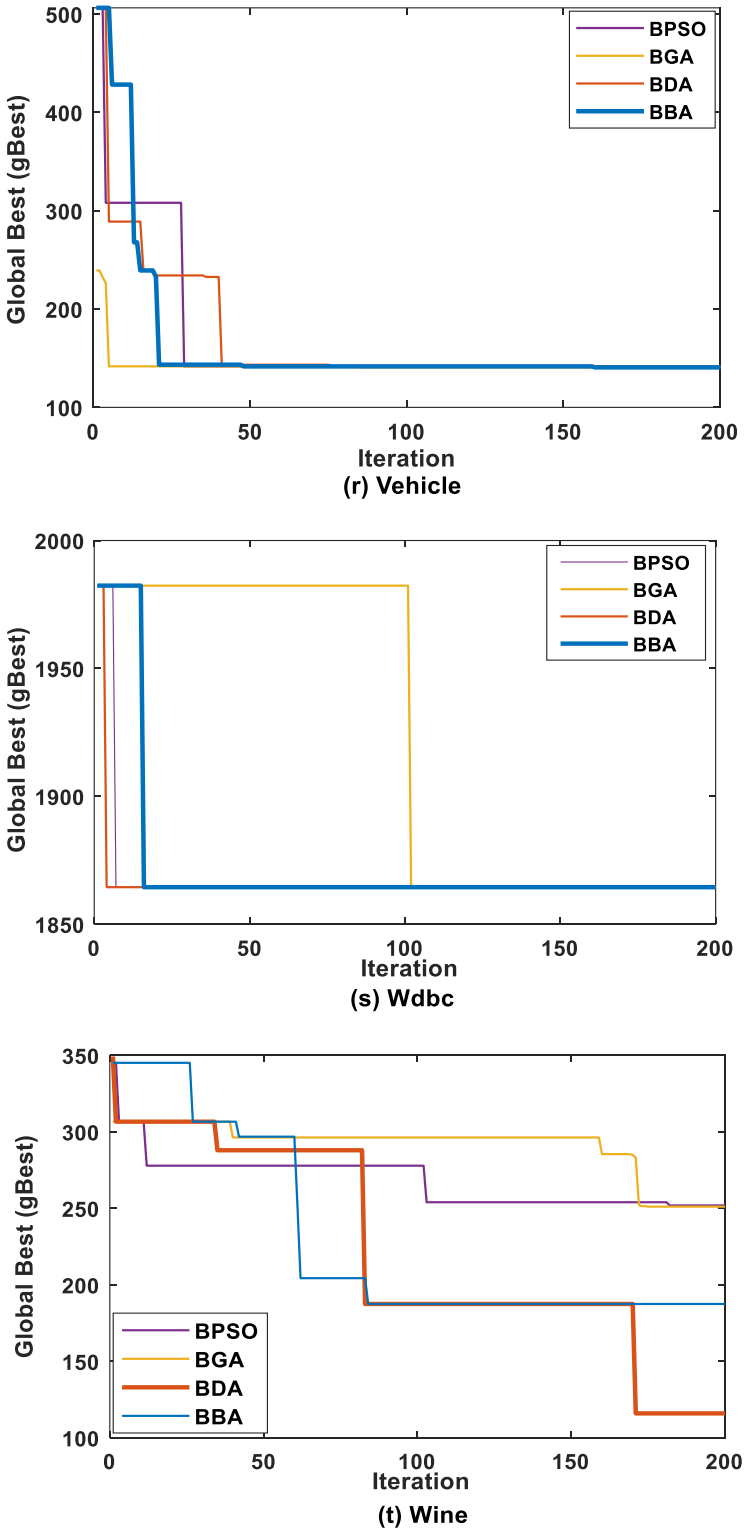


Figure 3.10: Convergence curve of (r) vehicle, (s) Wdbc, and (t) Wine datasets considering BBA, BPSO, BGA, and BDA in the optimizer module.

Figure 3.11 shows the shape dataset in its original form without clustering. Figure 3.12 illustrates the results of applying the proposed framework to the shape dataset to visualize some results. As can be seen, the proposed approach can generate different well-separated clusters while maintaining a trade-off between the number of clusters and their shape.

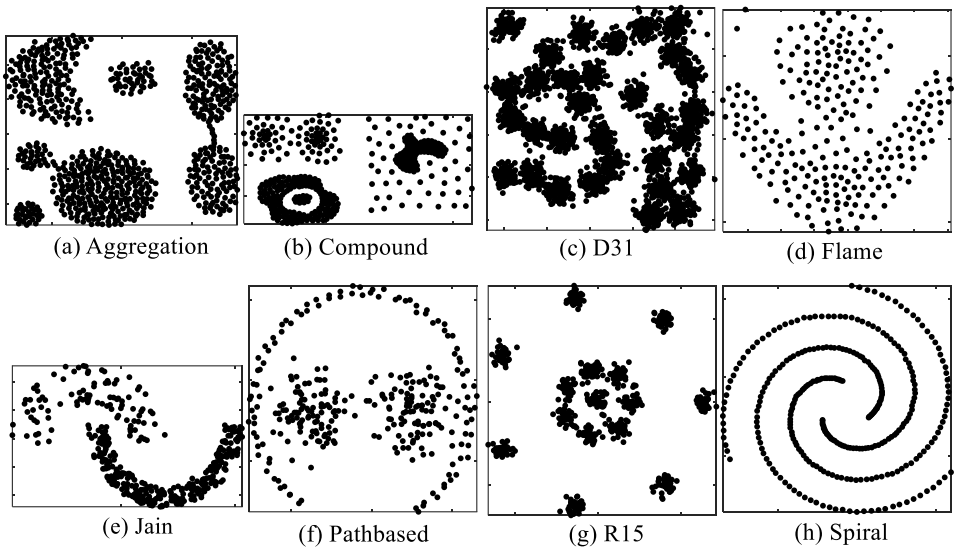


Figure 3.11: The shape dataset in its original form before performing the proposed clustering

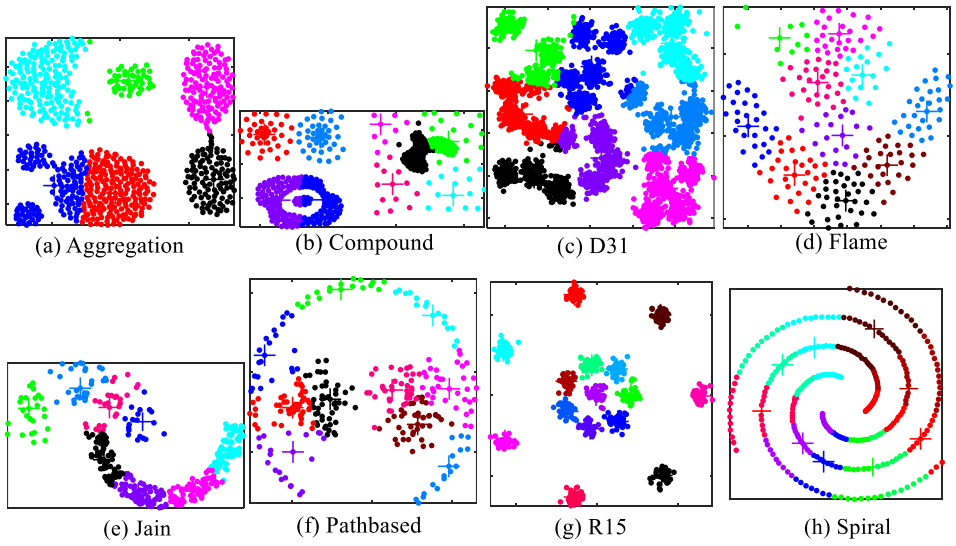


Figure 3.12: Visual results of performing the proposed clustering framework on the shape datasets.

In the second part of the comparison results, the performance of the proposed framework has been compared with other classical and new clustering algorithms in terms of internal validity measures.

We have considered the K-means ++ [103] as a representative of partitional clustering and the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm [104] as a representative of density-based models. Also, we have considered the Expectation-Maximization (EM) algorithm [105] and the Nearest point with indexing ratio (NPIR) algorithm [106]. EM is a famous example of distribution-based clustering that employs a fixed number of Gaussian distributions to reach the distribution of the objects. NPIR is one of the latest clustering algorithms and works based on finding the nearest neighbors. Moreover, we have considered two well-known optimization algorithms in the continuous search space. These two algorithms are known as GCUK [107], a genetic-based clustering with an unknown number of clusters, and DCPSO [108], the dynamic PSO.

In this comparison, we have evaluated the sum of intra-cluster distances, the sum of inter-cluster distances, and the DB-index for twenty independent trials with each approach. We have also calculated the distortion deviation in all clustering solutions and compared the result. The distortion deviation, calculated in Eq. (3.6), shows the difference in the size (radius) of clusters in a clustering approach. We wish to keep it minimized in our proposed method.

Our comparison study continues by utilizing the BBA in the optimizer module since it has provided excellent performance based on Tables 3.4 to 3.6; However, the other three algorithms have also shown competitive results, so they can also be considered in the optimizer module for the rest of the performance evaluation.

For the comparative study, preliminary experiments have been done to determine the best settings for the required parameters to calculate internal validity measures. For GCUK and DCPSO, the parameter settings are described in Table 3.2. The EM algorithm needs to know the number of clusters in advance.

The NPIR algorithm also requires prior knowledge of the number of clusters and the indexing ratio (IR) [106]. Therefore, we have performed multiple runs with various clusters in both

algorithms and also have considered different IR values suggested by the authors for the NPIR to find the most appropriate parameters for each dataset in these algorithms.

The DBSCAN algorithm forms clusters based on density-based connectivity, and its performance is affected by MinPts and eps parameters. The MinPts can be selected based on dimensionality, and the eps can be specified based on the elbow in the k-distance graph [109]. Authors in [110] suggest using larger MinPts for a noisy and large dataset. Also, depending on the aim of clustering, you can decrease eps to avoid large clusters or increase it to avoid noise. Hence, we have run the DBSCAN algorithm with different MinPts and eps values for each dataset to select the value leading to the best results in terms of the mentioned validity measures.

Table 3.7: Mean and standard deviation of the sum of intra/inter-cluster distances, DB-index and distortion deviation over 20 independent runs for the proposed framework, DCPSO, GCUK, K-MEANS++, DBSCAN, EM, and NPIR for the shape datasets

Algorithm	Measure \ Datasets	Aggregation	Compound	D31	Flame
Proposed	Intra-C Distances	3183 ± 125	1450 ± 458	9329 ± 186	318 ± 21
	Inter-C Distances	327 ± 8	484 ± 308	394 ± 90	279 ± 12
	DB Index	**0.57 ± 0.01**	**0.66 ± 0.07**	0.75 ± 0.03	0.76 ± 0.09
	Distortion Deviation	**1.69 ± 0.06**	**1.83 ± 0.04**	**0.37 + 0.01**	**0.48 ± 0.00**
DCPSO	Intra-C Distances	3232 ± 262	1507 ± 135	6453 ± 302	527 ± 143
	Inter-C Distances	313 ± 11	303 ± 42	2792 ± 339	158 ± 118
	DB Index	0.63 ± 0.05	0.72 ± 0.07	0.75 ± 0.02	**0.69 ± 0.05**
	Distortion Deviation	4.8 ± 1.58	3.54 ± 1.44	4.89 ± 0.72	2.14 ± 0.69
GCUK	Intra-C Distances	3795 ± 110	1491 ± 303	7334 ± 373	411 ± 67
	Inter-C Distances	375± 48	798 ± 453	1871 ± 953	275 ± 157
	DB Index	0.69 ± 0.06	0.74 ± 0.08	0.79 ± 0.00	0.76 ± 0.04
	Distortion Deviation	6.50 ± 1.45	2.46 ± 0.75	4.13 ± 0.83	2.16 ± 0.69
K-means++	Intra-C Distances	2937 ± 2	1159 ± 61	3169 ± 121	783 ± 6
	Inter-C Distances	271 ± 0.5	234 ±7	6170 ± 104	6 ± 0.04
	DB Index	0.64 ± 0.00	0.79 ± 0.06	0.65 ± 0.04	1.11 ± 0.00
	Distortion Deviation	1.72 ± 0.49	4.81 ± 0.86	1.81 ± 0.25	0.93 ± 0.22
DBSCAN	Intra-C Distances	3808 ± 537	871 ± 0.00	11202 ± 785	731 ± 30
	Inter-C Distances	207 ± 97	131 ± 0.00	365 ± 90	11 ± 1
	DB Index	0.65 ± 0.06	3.39 ± 0.00	0.76 ± 0.02	2.03 ± 1.64
	Distortion Deviation	10.77 ± 1.28	5.18 ± 0.00	7.54 ± 0.02	5.04 ± 1.44
EM	Intra-C Distances	3251 ± 286	1248 ± 80	3032 ± 200	819 ± 13
	Inter-C Distances	271 ± 13	222 ± 18	6371 ± 153	5.54 ± 0.05
	DB Index	0.65 ± 0.07	1.42 ± 0.74	1.14 ± 0.18	1.20 ± 0.02
	Distortion Deviation	7.06 ± 2.10	6.52 ± 1.22	4.13 ± 0.90	1.20 ± 0.60
NPIR	Intra-C Distances	3158 ± 142	1335 ± 42	2886 ± 6	766 ± 96
	Inter-C Distances	307 ± 53	180 ± 37	6085 ± 97	12 ± 1
	DB Index	0.85 ± 0.06	1.06 ± 0.13	**0.55 ± 0.01**	1.10 ± 0.15
	Distortion Deviation	7.73 ± 0.53	3.88 ± 0.14	1.76 ± 0.02	2.48 ± 1.25

Table 3.8: Mean and standard deviation of the sum of intra/inter-cluster distances, DB-index and distortion deviation over 20 independent runs for the proposed framework, DCPSO, GCUK, K-MEANS++, DBSCAN, EM, and NPIR for the shape datasets

Algorithm	Datasets / Measure	Jain	Pathbased	R15	Spiral
Proposed	Intra-C Distances	1007 ± 81	930 ± 65	501 ± 11	1782 ± 121
	Inter-C Distances	653 ± 34	665 ± 48	655 ± 59	310 ± 56
	DB Index	**0.64 ± 0.03**	**0.66 ± 0.00**	0.36 ± 0.07	**0.74 ±0.02**
	Distortion Deviation	**1.04 ± 0.001**	**0.57 ± 0.18**	**0.34 ± 00**	0.84 ± 0.21
DCPSO	Intra-C Distances	1819± 746	1267 ± 216	508 ± 41	1823 ± 253
	Inter-C Distances	269 ± 181	340 ± 205	622± 48	179 ± 79
	DB Index	**0.64 ± 0.03**	**0.66 ± 0.03**	0.41 ± 0.04	**0.74 ± 0.01**
	Distortion Deviation	4.03 ± 2.21	3.30 ± 1.54	0.78 ± 0.10	2.15 ± 0.98
GCUK	Intra-C Distances	1427 ± 407	975 ± 184	512 ± 49	1968±198
	Inter-C Distances	553 ± 374	908 ± 465	656 ± 63	123 ± 32
	DB Index	0.69 ± 0.09	0.73 ± 0.03	0.42 ± 0.06	**0.74 ± 0.00**
	Distortion Deviation	4.89 ± 2.30	4.35 ± 1.72	0.71 ± 0.00	1.98 ± 0.99
K-means++	Intra-C Distances	2623 ± 4	1435 ± 1	229± 12	1815 ± 0
	Inter-C Distances	18.00 ± 0.01	47 ± 0.07	644 ±11	40 ± 0.04
	DB Index	0.78 ± 0.00	**0.66 ± 0.00**	**0.33 ± 0.02**	0.87 ± 0.003
	Distortion Deviation	2.41 ± 0.13	1.20 ± 0.15	0.34 ± 0.18	**0.34 ± 0.09**
DBSCAN	Intra-C Distances	2809 ± 11	1528 ± 106	250 ±3	2908 ± 32
	Inter-C Distances	54 ± 0.09	47 ± 21	572 ± 0	9.94 ± 0.04
	DB Index	0.78 ± 0.00	1.67 ± 0.41	0.37 ±0.00	5.94 ± 0.07
	Distortion Deviation	53.97 ± 0.09	9.82 ± 0.70	0.84 ± 0.07	0.71 ± 0.08
EM	Intra-C Distances	2739 ± 0	1469 ± 0	273± 24	1829 ± 0
	Inter-C Distances	18 ± 4	48 ± 0	674 ± 25	36 ± 0
	DB Index	0.74 ± 1.39	0.68 ± 0.07	0.51 ± 0.09	0.99 ± 0.00
	Distortion Deviation	9.32 ± 0.00	7.15 ± 0.00	1.27 ± 0.27	0.83 ± 0.00
NPIR	Intra-C Distances	2647 ± 61	1699 ± 1	231 ± 7	1765 ± 70
	Inter-C Distances	18 ± 0.05	15.04 ± 0.01	640 ± 0	64 ± 19
	DB Index	0.76 ± 0.01	0.75 ± 0.00	**0.33 ± 0.02**	0.94 ± 0.02
	Distortion Deviation	6.99 ± 1.59	0.60 ± 0.27	0.57 ± 0.20	4.96 ± 2.64

Table 3.9: Mean and standard deviation of the sum of intra/inter-cluster distances, DB-index and distortion deviation over 20 independent runs for the proposed framework, DCPSO, GCUK, K-MEANS++, DBSCAN, EM, and NPIR for the real-life datasets

Algorithm	Measure \ Datasets	Appendicitis	Dermatology	Ecoli	Glass
Proposed	Intra-C Distances	66 ± 6	2977 ± 170	13904 ± 1764	376 ± 87
	Inter-C Distances	1.86 ± 0.11	1206 ± 546	156 ± 12	125 ± 24
	DB Index	**0.74 ± 0.02**	**1.01 ± 0.02**	**0.90 ± 0.01**	0.68 ± 0.18
	Distortion Deviation	0.20 ± 0.09	**3.53 ± 0.87**	**5.27 ± 3.44**	**3.30 ± 0.92**
DCPSO	Intra-C Distances	72 ± 6	2987 ± 272	15729 ± 2307	333 ± 26
	Inter-C Distances	1.58 ± 0.12	588 ± 130	1848 ± 1890	154 ± 32
	DB Index	0.89 ± 0.02	1.06 ± 0.03	0.91 ± 0.06	**0.60 ± 0.13**
	Distortion Deviation	0.59 ± 0.22	5.30 ± 0.78	34.38 ± 17.27	4.31 ± 1.08
GCUK	Intra-C Distances	65 ± 12	3173 ± 239	14829 ± 2254	499 ± 29
	Inter-C Distances	11.23 ± 9.46	451 ± 64	3037 ± 2337	158 ± 30
	DB Index	0.93 ± 0.10	1.02 ± 0.05	1.06 ± 0.06	0.88 ± 0.02
	Distortion Deviation	0.53 ± 0.26	4.05 ± 2.13	30.87 ± 14.87	4.85 ± 1.06
K-means++	Intra-C Distances	38 ± 0	2039 ± 39	10029 ± 178	243 ± 6
	Inter-C Distances	0.75 ± 0.01	397 ± 19	950 ± 77	135 ± 7
	DB Index	1 ± 0.03	**1.01 ± 0.02**	1.35 ± 0.11	0.72 ± 0.09
	Distortion Deviation	0.20 ± 0.10	4.02 ± 1.30	28.66 ± 5.73	4.31 ± 0.40
DBSCAN	Intra-C Distances	36 ± 0	2333 ± 115	10048 ± 191	188 ± 1
	Inter-C Distances	1.74 ± 0.01	772 ± 66	957 ± 71	20 ± 3
	DB Index	0.97 ± 0.01	1.41 ± 0.11	1.34 ± 0.02	0.83 ± 0.13
	Distortion Deviation	1.73 ± 0.00	19.27 ± 1.30	18.19 ± 0.33	4.33 ± 0.04
EM	Intra-C Distances	39 ± 0	3555 ± 304	11503 ± 809	233 ± 13
	Inter-C Distances	0.73 ± 0	292 ± 32	1793 ± 73	110 ± 10
	DB Index	1.06 ± 0.01	2.56 ± 0.59	2.20 ± 0.57	1.16 ± 0.22
	Distortion Deviation	**0.13 ± 0.01**	27.20 ± 4.27	44.08 ± 10.60	4.27±1.02
NPIR	Intra-C Distances	37 ± 2	2283 ± 106	12696 ± 358	226 ± 3
	Inter-C Distances	1.76 ± 0.77	262 ± 54	106 ± 58	120 ± 0
	DB Index	0.97 ± 0.30	1.67 ± 0.66	1.38 ± 0.19	1.50 ± 0.07
	Distortion Deviation	0.57 ± 0.08	12.02 ± 2.11	18.77 ± 5.01	3.78 ± 0.02

Table 3.10: Mean and standard deviation of the sum of intra/inter-cluster distances, DB-index and distortion deviation over 20 independent runs for the proposed framework, DCPSO, GCUK, K-MEANS++, DBSCAN, EM, and NPIR for the real-life datasets

Algorithm	Measure \ Datasets	Haberman	Housevotes	Ionosphere	Iris
Proposed	Intra-C Distances	2001 ± 125	341 ± 20	1210 ± 75	157 ± 8
	Inter-C Distances	152 ± 2	32 ± 0	78 ± 0	5 ± 0.16
	DB Index	**0.59 ± 0.00**	1.26 ± 0.06	1.24 ± 0.02	0.43 ± 0.00
	Distortion Deviation	7.49 ± 1.5	**0.07 ± 0.4**	**0.28 ± 0.14**	**0.20 ± 0.05**
DCPSO	Intra-C Distances	2964± 596	400 ± 21	1427 ± 67	132± 10
	Inter-C Distances	301 ± 281	35 ± 29	85 ± 56	11± 7
	DB Index	0.63 ± 0.05	1.44 ± 0.19	1.34 ± 0.07	0.55 ± 0.9
	Distortion Deviation	13.49 ± 4.58	0.29 ± 0.12	1.42 ± 0.31	0.60 ± 0.28
GCUK	Intra-C Distances	2600 ± 647	408 ± 20	1422 ± 52	139 ± 11
	Inter-C Distances	700 ± 664	102 ± 34	262 ± 132	8 ± 7
	DB Index	0.72 ± 0.07	1.97 ± 0.07	1.70 ± 0.04	0.44 ± 0.10
	Distortion Deviation	11.53 ± 4.57	0.39 ± 0.16	1.42 ± 0.47	0.31 ± 0.12
K-means++	Intra-C Distances	3054 ± 254	333 ± 0	831 ± 72	97 ± 0
	Inter-C Distances	17 ± 0	2.5 ± 0	3.7 ± 1.2	10 ± 0
	DB Index	1.22 ± 0.15	1.13 ± 0	1.30 ± 0.42	0.66 ± 0.00
	Distortion Deviation	**5.61 ± 3.65**	0.10 ± 0.01	0.74 ± 0.38	0.41 ± 0.00
DBSCAN	Intra-C Distances	247 ± 0	305 ± 0	479 ± 1	91 ± 12
	Inter-C Distances	11 ± 0	2.66 ± 0	2.40 ± 0	4.65 ± 1.85
	DB Index	1.13 ± 0.00	**1.04 ± 0.00**	1.21 ± 0.00	0.40 ± 0.15
	Distortion Deviation	21.18 ± 0.00	0.13 ± 0.09	3.45 ± 0.00	0.57 ± 0.19
EM	Intra-C Distances	3335 ± 0	333 ± 0	904 ± 0	104 ± 9
	Inter-C Distances	8.8 ± 0	2.54 ± 0	1.88 ± 0	9.30 ± 0.2
	DB Index	2.54 ± 0	1.13 ± 0.00	2.82 ± 0.01	0.77 ± 0.03
	Distortion Deviation	19.72 ± 0.00	0.12 ± 0.00	0.92 ± 0.08	0.89 ± 0.35
NPIR	Intra-C Distances	2716 ± 0	450 ± 0	845 ± 4	128 ± 0
	Inter-C Distances	18.55 ± 0	0.67 ± 0.09	11.08 ± 0.02	3.97 ± 0.01
	DB Index	0.91 ± 0.00	5.81 ± 0.91	**1.17 ± 0.01**	**0.38 ± 0.00**
	Distortion Deviation	22.48 ± 0.00	0.18 ± 0.02	2.16 ± 0.54	1.31 ± 0.00

Table 3.11: Mean and standard deviation of the sum of intra/inter-cluster distances, DB-index and distortion deviation over 20 independent runs for the proposed framework, DCPSO, GCUK, K-MEANS++, DBSCAN, EM, and NPIR for the real-life datasets

Algorithm	Datasets / Measure	Segment	Vehicle	Wdbc	Wine
Proposed	Intra-C Distances	246834 ± 54297	107956 ± 7782	262665 ± 9839	17221 ± 663
	Inter-C Distances	6083 ±2971	528 ± 42	3655 ± 260	7288 ± 21
	DB Index	**0.50 ± 0.09**	0.54 ± 0.00	0.51 ± 0.00	**0.47 ± 0.01**
	Distortion Deviation	**110 ± 45.87**	**22.40 ± 8.67**	**69.58 ± 35.89**	**16.96 ± 4.71**
DCPSO	Intra-C Distances	373634 ± 7257	73457 ± 6883	188642 ± 52069	17253 ± 895
	Inter-C Distances	15215 ± 2472	846 ± 125	10456 ± 9420	7065 ± 6736
	DB Index	0.54 ± 0.14	0.48 ± 0.03	0.52 ± 0.05	0.48 ± 0.01
	Distortion Deviation	361 ± 21	22.49 ± 9.90	528 ± 171	53.33 ± 16.22
GCUK	Intra-C Distances	477924 ± 38190	84066 ± 6907	221180 ± 53611	19242 ± 6757
	Inter-C Distances	25576 ± 8408	3894 ± 5314	12662 ± 16499	84142 ± 8529
	DB Index	0.78 ± 0.05	0.72 ± 0.22	0.59 ± 0.13	0.55 ± 0.04
	Distortion Deviation	325 ± 38	56.65 ± 13.90	438 ± 125	82.94 ± 37.62
K-means++	Intra-C Distances	270182 ± 24659	57195 ± 3886	152647 ± 0	17967 ± 836
	Inter-C Distances	17402± 86	2228 ± 160	1331 ± 0	1550 ± 45
	DB Index	0.69 ± 0.01	0.65 ± 0.017	0.50 ± 0.00	0.54 ± 0.006
	Distortion Deviation	122 ± 22	81.15 ± 41.96	2215 ± 0	205 ± 59
DBSCAN	Intra-C Distances	124129 ± 1294	16856 ± 56	93540 ± 1602	21726 ± 6405
	Inter-C Distances	3529 ± 375	1684 ± 6	786 ± 9	1173 ± 356
	DB Index	1.01 ± 0.02	0.63 ± 0.01	**0.40 ± 0.01**	0.59 ± 0.01
	Distortion Deviation	118 ± 3	46.02 ± 1.73	436 ± 22	184 ± 9
EM	Intra-C Distances	204704 ± 9713	55697 ± 1680	175896 ± 0	22874 ± 339
	Inter-C Distances	7185 ± 2785	1823 ± 345	1030 ± 0	1230 ± 17
	DB Index	2.97 ± 1.11	0.92 ± 0.09	0.70 ± 0.00	0.84 ± 0.02
	Distortion Deviation	791 ±2 71	167 ± 41	2558 ± 0	387 ± 147
NPIR	Intra-C Distances	160144 ± 6912	65104 ± 67	175454 ± 0	22535 ± 6782
	Inter-C Distances	2736 ± 656	359 ± 0	2113 ± 0	1023 ± 512
	DB Index	1.32 ± 0.09	**0.44 ± 0.00**	0.71 ± 0.00	0.58 ± 0.05
	Distortion Deviation	1331 ± 6	99 ± 0	3261 ± 0	424 ± 136

Table 3.12: Mean and standard deviation of the sum of intra/inter-cluster distances, DB-index and distortion deviation over 20 independent runs for the proposed framework, DCPSO, GCUK, K-MEANS++, DBSCAN, EM, and NPIR for the real-life datasets

Algorithm	Measure \ Datasets	Dime064	Dime128	Dime256	Dime512
Proposed	Intra-C Distances	122308 ± 137	138102 ± 227	14106 ± 221	184041 ± 399
	Inter-C Distances	143803 ± 1169	18489 ± 409	27070 ± 914	38553 ± 1364
	DB Index	**0.04 ± 0.00**	0.07 ± 0.00	0.04 ± 0.00	0.15 ± 0.05
	Distortion Deviation	**26.82 ± 2.88**	**30.55 ± 1.65**	**18.00 ± 3.64**	**16.78 ± 5.30**
DCPSO	Intra-C Distances	21901 ± 517	11852 ± 683	18457 ± 1463	190630 ± 13235
	Inter-C Distances	9475 ± 520	201642 ± 796	19413 ± 1565	384915 ± 17197
	DB Index	0.20 ± 0.04	1.17 ± 0.23	0.10 ± 0.03	1.01 ± 0.03
	Distortion Deviation	45.81 ± 5.06	78.37 ± 13.54	43.06 ± 7.39	78.61 ± 4.49
GCUK	Intra-C Distances	21455 ± 599	12731 ± 703	19928 ± 2129	198168 ± 14725
	Inter-C Distances	10331 ± 629	21486 ± 707	18547 ± 2181	395861 ± 13172
	DB Index	0.14 ± 0.04	1.42 ± 0.11	0.09 ± 0.04	2.16 ± 0.13
	Distortion Deviation	46.22 ± 4.42	71.26 ± 14.56	41.84 ± 8.51	75.12 ± 6.42
K-means++	Intra-C Distances	12149 ± 246	13691 ± 359	13975 ± 164	16022 ± 436
	Inter-C Distances	144819 ± 475	205577 ± 612	296376 ± 765	418384 ± 684
	DB Index	**0.04 ± 0.00**	**0.04 ± 0.00**	**0.02 ± 0.00**	**0.02 ± 0.01**
	Distortion Deviation	35.46 ± 1.81	46.91 ± 1.91	22.60 ± 2.67	27.71 ± 0.74
DBSCAN	Intra-C Distances	11958 ± 202	13182 ± 309	13400 ± 242	15106 ± 1798
	Inter-C Distances	144915 ± 484	205765 ± 973	296078 ± 1087	420203 ± 2817
	DB Index	0.08 ± 0.01	0.10 ± 0.01	0.07 ± 0.01	0.41 ± 0.07
	Distortion Deviation	36.82 ± 3.29	35.41 ± 4.16	24.83 ± 2.29	40.41 ± 3.42
EM	Intra-C Distances	12014 ± 277	13416 ± 467	13985 ± 226	152965 ± 939
	Inter-C Distances	143968 ± 499	205785 ± 739	286227 ± 1104	461964 ± 3321
	DB Index	0.06 ± 0.01	**0.04 ± 0.00**	0.04 ± 0.00	0.68 ± 0.07
	Distortion Deviation	39.61 ± 2.69	47.85 ± 2.79	26.62 ± 3.03	38.12 ± 5.97
NPIR	Intra-C Distances	11677 ± 933	12935 ± 1060	14203 ± 920	151784 ± 1176
	Inter-C Distances	144075 ± 2259	205647 ± 2011	276229 ± 1520	471126 ± 2768
	DB Index	0.09 ± 0.03	1.15 ± 0.55	0.03 ± 0.00	0.67 ± 0.09
	Distortion Deviation	40.44 ± 2.15	54.46 ± 6.07	25.72 ± 4.17	35.85 ± 4.28

3.4.3. Discussion

The numerical results over twenty-four datasets have been summarized in Tables 3.7 to 3.12. It can be seen that in comparison with k-needed clustering methods (NPIR, K-Means++, and EM), the proposed framework has shown an excellent performance in most of the datasets in terms of the DB-Index and the distortion deviation measures as the main focus in our clustering method. In some cases, such as D31 and R15 in the shape datasets and Ionosphere, Iris, and vehicle in the Real-world datasets, the NPIR algorithm has shown better performance in the DB-index measure. However, the proposed method has yielded a smaller distortion deviation in all these datasets.

The reason why the proposed framework has reached a bit higher DB-index measure compared to the other algorithms in some datasets can be explained as follows. The proposed framework focuses mainly on reaching clusters with approximately the same distance while satisfying the other designed merit factors. This goal has been achieved by dividing the dataset into more or fewer groups compared to other algorithms in some cases; for example, D31, which contains several spherical clusters with high overlap. The proposed framework has divided D31 into fewer clusters compared to others. Consequently, the sum of intra-cluster distances has significantly increased while inter-cluster distances have considerably decreased. As a result, we have reached a higher DB-index measure. Yet, we have demonstrated the best performance in terms of the distortion deviation in this dataset.

K-means++ has shown highly competitive performance in the Pathbased and dermatology datasets in terms of the DB-index. It has also been able to achieve better performance in distortion deviation on spiral and Haberman datasets. However, by increasing the dimension and the data points as shown in higher-dimensional datasets, the proposed framework outperforms the K-means++ in terms of the distortion deviation measure.

The proposed algorithm has achieved the best DB-index measure in the Appendicitis dataset, but the EM algorithm reached the best distortion deviation in this case.

On the other hand, the proposed framework has shown highly successful results in the automatic k-determination clustering methods (DBSCAN, DCPSO, and GUCK) in most datasets. DBSCAN algorithm has reached the best value of DB-index in the Housevotes and WDBC datasets.

Nevertheless, the proposed method has obtained the best distortion deviation among all other algorithms in both datasets. It should be mentioned that DBSCAN is fundamentally different from center-based clustering methods. Although the DB-index measure may not be considered a fair validity measure for this clustering method, we needed to evaluate the proposed solution in terms of the DB-index with other algorithms for the purpose of this research. Aside from this point, the algorithm has not performed well in reaching the minimum possible value of distortion deviation compared to the other algorithms.

In a few datasets (such as Jain, Pathbased, and Spiral), the DCPSO and the GCUK algorithms have reached the same value as the proposed solution in the DB index. Nevertheless, the distortion deviation, which is a primary goal in this research, still yields lower figures in these datasets for the proposed framework.

3.5. Statistical Analysis

To validate the above numerical results, we have performed the Friedman non-parametric statistical test [111][112]. This test, similar to the ANOVA [48], can point out significant differences between the behavior of two or more algorithms.

Table 3.13 describes the achieved ranks by the Friedman test in the proposed framework considering different optimizer modules. The ranks indicate that the optimizer module with all four optimization algorithms has shown excellent performance, especially among the BBA, BGA, and BPSO. However, BBA and BGA have been ranked better in most datasets with the proposed framework.

Then, we performed another statistical test called the Wilcoxon rank-sum test [113][114] to draw a more meaningful conclusion from the results. The Wilcoxon rank-sum test for equal medians establishes a proper pairwise comparison between the algorithms. It compares the null hypothesis that two values are samples from a continuous distribution with equal medians against the alternative that they are not.

For evaluating the first set of comparison results, the Wilcoxon test has been applied to the optimizer module considering the optimization algorithm with the best average performance against the rest of the algorithms, according to Tables 3.3 to 3.6.

The significance level is considered 0.05, which gives strong evidence against the null hypothesis. The achieved p-values by the Wilcoxon test for the first set of comparison results considering the optimizer module with different optimization algorithms have been reported in Table 3.14. As can be seen in some datasets (i.e., R15, Wine), the p-value by the optimizer module with one of the algorithms has a significant difference from others. On the other hand, the achieved p-values by the optimizer module with two or three other algorithms are not significantly different in some other datasets (i.e., Pathbased, Dermatology, Haberman, Housevotes, Wdbc, Dim256). It means the proposed framework utilizing all these optimizer modules exhibits similar performance for that dataset.

We have also applied these non-parametric statistical tests to the second set of comparison results to compare the proposed framework with other algorithms statistically. During this experiment, we implemented the proposed framework considering the BBA optimizer.

We have applied the statistical tests to the DB-index and distortion deviation measures achieved in the second set of comparison results. The achieved ranks by the Friedman test and the p-values by the Wilcoxon test for the second set of comparison results are reported in Tables 3.15 and 3.18. As shown, the proposed framework is able to reach the best or the second-best DB-index rank in multiple datasets. Besides, the proposed framework has achieved the best distortion deviation rank in almost all datasets, which is a great success.

The reason why we have not achieved the lowest DB-index rank in a few datasets lies in how the problem has been formulated in our model. In line with the initial motivation of this research, reaching the minimum distortion deviation has been prioritized in our proposed model. Therefore, there might be cases where the proposed framework achieves the minimum distortion by dividing the data points into more or fewer groups, affecting the DB-index measure.

Table 3.13: Achieved Ranks by the Friedman Test for the Proposed Framework Considering four Different Optimizer Modules

Datasets	BBA	BPSO	BGA	BDA
Aggregation	**1.65**	2.65	2.08	3.63
Compound	2.75	2.23	**1.58**	3.45
D31	**1.70**	2.70	2.13	3.48
Flame	2.10	2.78	**1.93**	3.20
Jain	2.45	**2.08**	2.75	2.73
Pathbased	**2.10**	2.60	2.25	3.05
R15	2.90	2.55	**1.68**	2.88
Spiral	2.45	2.65	**2.10**	2.80
Appendicitis	**2.00**	2.35	2.70	2.95
Dermatology	**1.98**	2.48	2.35	3.20
Ecoli	2.25	2.75	**2.10**	2.90
Glass	**2.20**	2.70	2.50	2.60
Haberman	2.75	2.55	**2.35**	2.35
Housevotes	2.15	2.40	**1.85**	3.60
Ionosphere	**2.05**	2.55	2.20	3.20
Iris	**2.30**	2.40	2.45	2.85
Segment	**2.20**	2.60	2.68	2.53
Vehicle	2.38	2.78	**1.83**	3.03
Wdbc	**2.20**	2.55	2.65	2.60
Wine	2.30	**1.63**	2.65	3.43
Dim64	**2.03**	2.43	2.73	2.83
Dim128	**2.08**	2.48	2.75	2.70
Dim256	**2.40**	2.50	2.50	2.60
Dime512	**2.30**	**2.40**	**2.55**	**2.75**

Table 3.14: Achieved P-values by the Wilcoxon Rank-Sum Test

Datasets	BBA	BPSO	BGA	BDA
Aggregation	1	0.0077	0.4849	0.0000
Compound	0.000	0.2235	1	0.0000
D31	1	0.0006	0.2439	0.0000
Flame	0.6073	0.0032	1	0.0005
Jain	0.2288	1	0.0711	0.0265
Pathbased	1	0.0496	0.6262	0.0149
R15	0.0023	0.0110	1	0.0009
Spiral	0.2733	0.0909	1	0.0468
Appendicitis	1	0.4092	0.0482	0.0058
Dermatology	1	0.2789	0.4986	0.0087
Ecoli	1	0.1636	0.8817	0.0133
Glass	1	0.0251	0.0914	0.0482
Haberman	0.1960	0.5065	1	1.0000
Housevotes	0.3104	0.1988	1.0000	0.0000
Ionosphere	1	0.1895	0.4249	0.0005
Iris	1	0.2972	0.5883	0.0138
Segment	1	0.0671	0.0335	0.1573
Vehicle	0.1410	0.0129	1	0.0029
Wdbc	1	0.9246	0.6750	0.7972
Wine	0.0441	1.0000	0.0088	0.0000
Dim64	1	0.1774	0.0094	0.0019
Dim128	1	0.1794	0.0097	0.0226
Dim256	1	0.3421	0.3421	0.1624
Dime512	1	0.2733	0.0565	0.0962

Table 3.15: Achieved Ranks by the Friedman Test on the DB-index for the Proposed Framework Compared to Other Algorithms

	DB-index ranks						
	Proposed	DCPSO	GCUK	K-means++	DBSCAN	EM	NPIR
Aggregation	**1.2**	3.6	4.8	4.4	3.95	3.2	6.85
Compound	**1.65**	2.5	2.6	3.4	7	5.3	5.55
D31	3.95	3.9	5.75	1.95	4.4	7	**1.05**
Flame	2.35	**1.5**	2.35	4.6	5.25	6.75	5.2
Jain	**1.6**	1.85	3.55	6.35	6.25	3.8	4.6
Pathbased	**1.9**	2.35	4.95	3.25	7	3	5.55
R15	3.05	5	5.35	2.05	3.7	6.65	**2.2**
Spiral	2.35	1.9	**1.75**	4	7	6	5
Appendicitis	**1.45**	2.8	3.75	4.9	4.45	6.35	4.3
Dermatology	2.35	3.45	2.35	**1.85**	5.65	6.65	5.7
Ecoli	**1.4**	1.65	2.95	5.05	5.1	7	4.85
Glass	2.55	**1.7**	4.55	2.65	3.65	6	6.9
Haberman	**1.15**	2	2.85	5.75	5.25	7	4
Housevotes	3.9	4.75	6	3	**1.05**	2.3	7
Ionosphere	3.2	3.95	6	4.2	2.45	7	**1.2**
Iris	3.45	4.65	3.3	5.7	1.85	7	**2.05**
Segment	**1.4**	1.85	3.9	2.85	5	7	6
Vehicle	3.25	2.1	4.95	5.3	4.5	6.8	**1.1**
Wdbc	3.65	3.5	4.55	2.6	**1.1**	5.8	6.8
Wine	**1.5**	1.7	3.9	3.875	5.35	7	4.675
Dim64	**1.45**	6.85	5.95	1.75	4.40	3.30	4.30
Dim128	3.85	6.20	6.80	2.30	4.95	**1.25**	2.65
Dim256	3.70	6.25	5.75	**1.00**	5.55	3.70	2.05
Dim512	2.00	6.30	6.70	**1.00**	3.00	4.50	4.50

Table 3.16: Achieved Ranks by the Friedman Test on Distortion Deviation Measures for the Proposed Framework Compared to Other Algorithms

	DB-index ranks						
	Proposed	DCPSO	GCUK	K-means++	DBSCAN	EM	NPIR
Aggregation	**1.45**	3.35	4.65	1.55	6.85	4.85	5.3
Compound	**1.4**	3.25	2.15	5.35	5.5	6.7	3.65
D31	**1**	6.65	6.05	3.45	2.1	5.3	3.45
Flame	**1.05**	4.95	4.55	2.55	6.85	2.8	5.25
Jain	**1.1**	3.35	3.55	2.25	7	5.9	4.85
Pathbased	**1.45**	4.25	4.75	3	6.95	5.95	1.65
R15	2.15	4.95	3.9	**1.4**	5.45	6.7	3.45
Spiral	3.5	5.45	5.3	**1.15**	2.85	3.65	6.1
Appendicitis	2.25	5.1	4.65	2.35	7	**1.8**	4.85
Dermatology	**1.95**	3.3	2.35	2.4	6.05	6.95	5
Ecoli	**1.05**	5.2	4.65	4.9	2.7	6.25	3.25
Glass	**2**	4.25	5.15	4.5	4.95	4.6	2.55
Haberman	**2.3**	3.7	2.95	1.3	5.9	4.85	7
Housevotes	**1.9**	5.85	6.4	2.25	3.5	3.25	4.85
Ionosphere	**1.2**	4.55	4.2	2.35	7	3	5.7
Iris	**1.5**	4.25	2.45	3.2	4.1	5.7	6.8
Segment	1.85	4.75	4.25	2.4	**1.75**	6	7
Vehicle	**1.3**	1.7	4	5.25	3.25	6.75	5.75
Wdbc	**1**	3.3	3.05	5	2.65	6	7
Wine	**1**	2.15	2.85	4.375	4.75	6.25	6.625
Dim64	**1.00**	6.00	6.40	2.60	2.70	4.30	5.00
Dim128	**1.10**	6.50	6.25	3.50	1.90	3.70	5.05
Dim256	**1.30**	6.50	6.40	2.55	3.35	4.20	3.70
Dim512	**1.00**	6.60	6.40	2.05	4.45	3.95	3.55

Table 3.17: Achieved P-values by the Wilcoxon Rank-Sum Test on the DB-index for the Proposed Framework Compared to Other Algorithms

	DB-index					
	DCPSO	GCUK	K-means++	DBSCAN	EM	NPIR
Aggregation	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000
Compound	0.0071	0.0026	0.0000	0.0000	0.0000	0.0000
D31	0.0000	0.0001	0.0000	0.0000	0.0960	0.0000
Flame	0.0060	0.9892	0.0000	0.0000	0.0000	0.0000
Jain	0.7353	0.0077	0.0000	0.0000	0.0000	0.0000
Pathbased	0.8392	0.0000	0.0574	0.0000	0.0000	0.0000
R15	0.0043	0.0040	0.0970	0.2067	0.0000	0.2493
Spiral	0.0294	0.0215	0.0000	0.0000	0.0000	0.0000
Appendicitis	0.0000	0.0000	0.0000	0.0000	0.0000	0.5953
Dermatology	0.0000	0.7972	0.2499	0.0000	0.0000	0.0000
Ecoli	0.2287	0.0000	0.0000	0.0000	0.0000	0.0000
Glass	0.1806	0.0001	0.2731	0.0175	0.0000	0.0000
Haberman	0.2534	0.0000	0.0000	0.0000	0.0000	0.0000
Housevotes	0.0020	0.0000	0.0000	0.0000	0.0000	0.0000
Ionosphere	0.0001	0.0000	0.0010	0.0001	0.0000	0.0000
Iris	0.0000	0.4570	0.0000	0.0002	0.0000	0.0000
Segment	0.6949	0.0000	0.0009	0.0000	0.0000	0.0000
Vehicle	0.0000	0.0040	0.0000	0.0000	0.0000	0.0000
Wdbc	0.4903	0.0083	0.0000	0.0000	0.0000	0.0000
Wine	0.2534	0.0000	0.0000	0.0000	0.0000	0.0000
Dim64	0.0000	0.0000	0.3548	0.0000	0.0000	0.0000
Dim128	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001
Dim256	0.0000	0.0002	0.0000	0.0000	0.3734	0.0000
Dim512	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table 3.18: Achieved P-values by the Wilcoxon Rank-Sum Test on Distortion Deviation Measures for the Proposed Framework Compared to Other Algorithms

	Distortion deviation					
	DCPSO	GCUK	K-means++	DBSCAN	EM	NPIR
Aggregation	0.0000	0.0000	0.5801	0.0000	0.0000	0.0000
Compound	0.0005	0.0003	0.0000	0.0000	0.0000	0.0000
D31	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Flame	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Jain	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Pathbased	0.0000	0.0000	0.0000	0.0000	0.0000	0.7338
R15	0.0000	0.0000	0.0000	0.0000	0.0000	0.0005
Spiral	0.0000	0.0001	0.0000	0.0148	0.8168	0.0001
Appendicitis	0.0000	0.0000	0.6488	0.0000	0.3304	0.0000
Dermatology	0.0000	0.6554	0.1070	0.0000	0.0000	0.0000
Ecoli	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Glass	0.0060	0.0000	0.0000	0.0002	0.0016	0.0010
Haberman	0.0000	0.0040	0.0000	0.0000	0.0000	0.0000
Housevotes	0.0000	0.0000	0.0127	0.0625	0.0001	0.0000
Ionosphere	0.0000	0.0000	0.0010	0.0000	0.0000	0.0000
Iris	0.0000	0.0006	0.0000	0.0000	0.0000	0.0000
Segment	0.0000	0.0000	0.1961	0.6749	0.0000	0.0000
Vehicle	0.6359	0.0000	0.0000	0.0000	0.0000	0.0000
Wdbc	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Wine	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Dim64	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Dim128	0.0000	0.0000	0.0000	0.0001	0.0000	0.0000
Dim256	0.0000	0.0000	0.0001	0.0000	0.0000	0.0000
Dim512	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

We have observed the performance of the proposed clustering framework in different datasets with varying numbers of data points and dimensions. According to our statistical analysis, the proposed clustering framework performs exceptionally well on datasets with small to mid-sized data points from different dimensions. From our perspective, having a dataset with high data points can slightly affect the performance of the clustering framework as an increase in data points directly affects the proposed binary encoding scheme and therefore increases the size of the search space in our proposed model. Meanwhile, increasing the dataset dimensions has no direct impact on the proposed approach and only affects the distance calculation process, as should be expected when working with higher-dimensional datasets. As also pointed out by [106], the Euclidean distance is not an appropriate distance measure for higher-dimensional datasets in general; Hence, considering an appropriate distance measure in such datasets should also enhance the results, which could be investigated in future studies.

3.6. Automatic Clustering for Binary Correlated Sources

The proposed clustering framework has been applied to a set of correlated binary datasets as a case study in this section.

Binary data is the simplest case of categorical data in which only two possible values describe discrete attributes and can be reflected as a special case of quantitative data. Binary data clustering is a challenging task due to its high dimensionality and sparsity [115]. The correlated binary clustering is beneficial in various disciplines such as medical sciences, machine learning, big data, pattern recognition, image analysis [38][94], and many other recent applications such as cache-aided networks and edge caching. In such cases, taking advantage of the similarity between the sample sets in the clustering solution can improve efficiency and reduce the delivery load. The presence of correlation in a binary dataset can be realized as the relevance of content files in the same category, such as the repeated measurements in remote sensing, the updated versions of dynamic content, augmented reality, news updates, etc. [38][17]. Moreover, correlated binary data clustering is widely used in medical studies, such as dental and radiologic studies. In such cases, the observations are taken from multiple representations of the same subject [116].

3.6.1. Methodology for Binary Case

In the binary case, each data point is a B-bit binary vector. Therefore, the B-dimensional binary dataset with m data points is indicated by $\mathcal{A}$.

$$\boldsymbol{\mathcal{A}}_{m\times B} = \begin{bmatrix} \mathcal{A}_1 \\ \vdots \\ \mathcal{A}_m \end{bmatrix} = \begin{bmatrix} a_{1,1}, a_{1,2} & \dots & , a_{1,B} \\ \vdots & \vdots & \vdots \\ a_{m,1}, a_{m,2} & \dots & , a_{m,B} \end{bmatrix}_{m\times B} \tag{3.12}$$

Since the proposed clustering, discussed in Chapter 3, is designed as a customizable general framework, it can also cater to binary datasets; therefore, all of the steps of problem formulation are the same as the general clustering framework. However, the definition of *distance measure* and *representative selection* has been tailored with binary space.

In the binary case, the distance measure is the Hamming distance [38], defined as follows:

$$d_{i,j}^{H} = \begin{cases} 1 & if\ a_{i,B} \neq a_{j,B}\ , \forall i \neq j \\ 0 & Otherwise \end{cases} \tag{3.13}$$

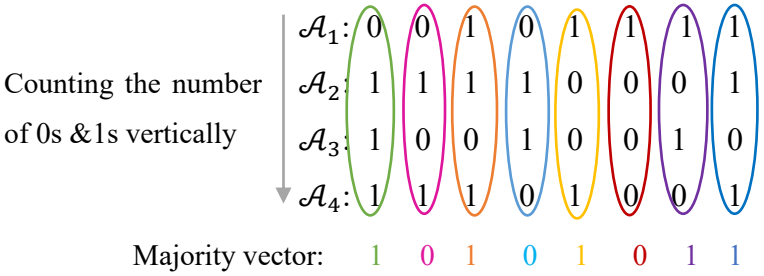
The maximum distance of each cluster is calculated according to the Hamming distance measure between each data point $\mathcal{A}_i$ and its representative $\hat{F}_i$ within each cluster c_i, where $c_i \in C$, and $i = [1,2, \dots, K]$.

$$\delta^{c_i} = \max_{c_i} d^{H}\left(\mathcal{A}_i, \hat{F}_i\right) \tag{3.14}$$

Selecting a cluster representative in the binary case is carried out in two consecutive steps; First, a centroid is identified within each cluster by performing the majority rule. Then, one actual point within the clusters with the least distance to the centroid is selected as the representative. According to the majority rule, a decision is made based on the majority of alternatives.

The following example shows how a centroid is determined based on the majority rule for a group of 4 binary data points.

Example 3.3:



The similarity among correlated vectors is extracted by a statistic measure known as *the simple matching coefficient (SMC)* [105][118], which is closely related to the definition of the hamming distance on bit strings.

Consider $\mathcal{A}_i$ and $\mathcal{A}_j$ as two different n-bit binary vectors in a cluster. Let B_{11} and B_{00} represent the number of bits that are 1 or 0 simultaneously among two vectors while B_{01} and B_{10} represent the number of bits that are not the same in each position. Then SMC is defined as follows.

$$SMC = \frac{B_{00}+B_{11}}{B_{11}+B_{00}+B_{01}+B_{10}} \tag{3.15}$$

Similar to the discussed experiment in the general framework, the goal is partitioning $\boldsymbol{\mathcal{A}}_{m\times B}$ into K number of compact and well-separated clusters with relatively close values for the maximum distance in each group, such that $K \leq m$.

3.6.2. Results and Discussion for Correlated Binary Case

In this section, the performance of the proposed correlation-aware clustering scheme has been analyzed on a set of correlated binary datasets. For this purpose, based on assumptions considered in [22] for generating correlated binary vectors, three synthetic binary datasets are generated with dimensions 128×100 bits and a similarity of 50%, 60%, and 70%.

Similar to the general framework, this scheme is also evaluated under the presence of the BBA, BPSO, BGA, and BDA in the optimizer module, and the convergence curve is illustrated in Figure 3.12. For each algorithm, twenty independent trials have been performed on each dataset. The best and worst cost, the average cost, and the standard deviation have been reported in Table 3.19.

The statistical analysis is described in Table 3.20, which shows the binary scheme has high capabilities in performing the clustering task considering all four optimization algorithms in the optimizer module. However, the optimizer module considering the BBA and BDA provides superb performance in solving binary clustering problems since the fastest convergence rate belongs to the BBA, followed by the BDA optimizer.

The BBA algorithm can take advantage of the loudness and pulse emission balance between the exploration and exploitation to accelerate the convergence rate toward the global optimum and not trap in local minima over the course of iterations. Besides, the V-shaped transfer function in the BBA algorithm helps the particles not go through the unpromising area of the search space, and therefore it contributes to having a fast convergence rate in this case. The BDA optimizer also inherits high exploration and exploitation from the DA algorithm and provides an excellent result.

Furthermore, the convergence curves of this experiment show that by increasing the similarity among data points, the convergence speed significantly increases. Consequently, reaching the minimum cost can be achievable in fewer iterations. The reason is that, as the correlation among datasets increases, the similarity between data points becomes very large. As a result, the maximum Hamming distance between data points decreases; therefore, the clustering problem becomes a much simpler problem that can even be solved in less than half of the iterations. In such cases, the maximum distance in each cluster will be decreased as well.

Table 3.19: Achieved P-values by the Wilcoxon Rank-Sum Test on Distortion Deviation Measures for the Proposed Framework Compared to Other Algorithms

Datasets	BBA				BPSO			
	Best	Worst	Mean	Std	Best	Worst	Mean	Std
Dataset 1	13.22	27.59	17.44	5.75	27.59	27.59	21.53	6.55
Dataset 2	13.23	13.98	**13.72**	0.22	14.13	14.13	13.93	0.12
Dataset 3	0.00	13.93	**12.16**	4.17	13.98	13.98	13.78	0.15
Datasets	BGA				BDA			
	Best	Worst	Mean	Std	Best	Worst	Mean	Std
Dataset 1	27.65	27.65	18.81	6.17	12.67	27.48	**17.01**	5.67
Dataset 2	14.26	14.26	13.91	0.18	13.25	14.02	13.79	0.17
Dataset 3	13.96	13.96	13.65	0.18	12.95	13.84	13.57	0.25

Table 3.20: Statistical Results of the Proposed Correlation-Aware Clustering Scheme Considering Four Different Optimizer Modules

Datasets	Friedman's Rank				P-Values			
	BBA	BPSO	BGA	BDA	BBA	BPSO	BGA	BDA
Dataset 1	2.35	3.25	2.7	**1.7**	0.13	0.0003	0.0071	1
Dataset 2	**2.05**	3.05	2.8	2.1	1	0.0026	0.0051	0.4249
Dataset 3	**1.85**	3.5	2.45	2.2	1	0.0016	0.2184	0.4093

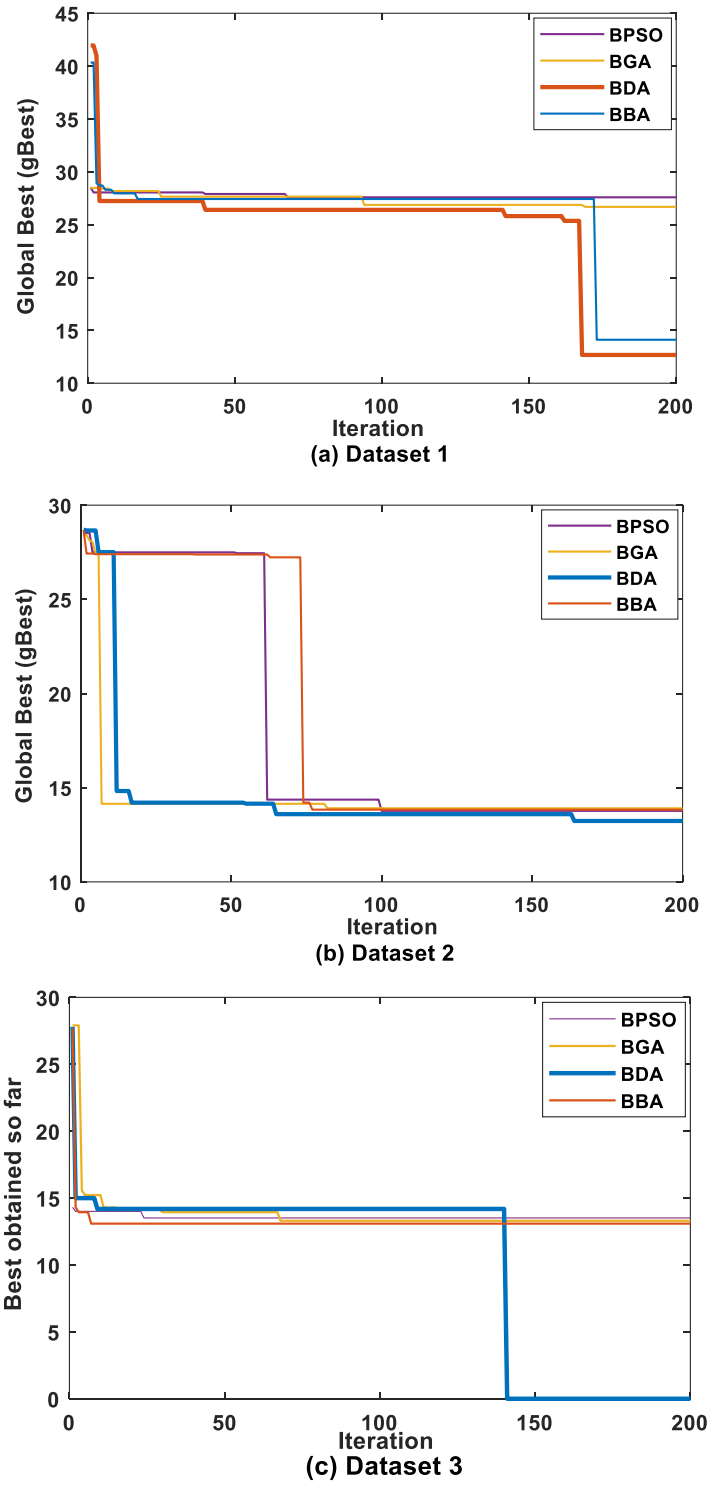


Figure 3.13: Convergence curve of (a) dataset 1, (b) dataset 2, and (c) dataset 3 considering BBA, BPSO, BGA, and BDA in the optimizer module.

3.7. Summary

Clustering algorithms are developed as a powerful tool to analyze the massive amount of data produced by cutting-edge technologies. Over the years, various meta-heuristic searching techniques have been proposed to achieve optimal or near-optimal solutions due to the challenges such as defining a suitable objective function and ambiguity in data clustering definition.

In this chapter, the clustering problem is formulated as an optimization problem with the motivation to reach well-separated clusters with approximately the same maximum distance. This framework utilized an AI-based optimizer module considering metaheuristics binary optimization algorithm. It adopts a dynamic range of clusters in accordance with the input data to tackle the problem of determining the exact number of clusters in advance. Hence, users do not need prior knowledge of the number of clusters. We have also proposed a binary encoding scheme for the particle representation in the proposed framework.

Furthermore, we examined the proposed clustering framework for correlated binary datasets as a case study. A wide range of practical applications can benefit from such datasets, including repeated measurements in remote sensing, medical studies, cache-aided networks with correlated content, and crowdsourced multi-view video uploading.

According to the results, we have successfully reached a fair number of well-separated clusters with approximately the same maximum distance for each cluster in most datasets. This chapter can be considered the opening for further research to improve the distortion deviation between clusters in other applications. Future studies can consider the proposed automatic clustering framework with approximately the same maximum distance in each cluster as a multi-objective optimization algorithm problem and consider the maximum distance of the clusters as an objective to possibly improve the distortion deviation gap.

Chapter 4

Content Delivery in a Network with a Single Shared Cache and Correlated Content

So far, we have comprehensively described and evaluated the proposed clustering scheme in a general framework. This framework is an essential part of our research as it will be used to develop two novel clustering schemes for the placement phase of the proposed caching networks in the current and following chapters.

Caching networks are typically analyzed under uniform or non-uniform popularity demand distributions. In this regard, this chapter considers a uniform demand distribution and studies the proposed network under lossy caching.

Developing the lossy caching scenario is highly beneficial in many practical applications, particularly involving multimedia content, as files can be downloaded at different quality levels depending on the channel and traffic conditions or the capabilities of the device. For example, the description of a file requested by a laptop user may require high quality, while a mobile user can be satisfied with a much lower-resolution description [119].

4.1. Introduction

In this chapter, we study a content delivery caching network considering a single shared cache. Connecting all receivers to a single shared cache is beneficial in many practical applications. Consider robots as laborers in a factory or drones as operators in deserted areas. In such settings, providing a shared cache in the access point to be filled with the most useful content during the placement phase leads to transmitting only small updates (e.g., recent maps and frequent updates of the locations under their coverage) during the delivery phase.

Networks with multiple caches can reduce delivery rates by taking advantage of multicast opportunities and global caching gains in addition to local gains; But networks with a single cache can not benefit from the global caching gain as only one local cache is available in the network. The optimal caching strategy for a single-cache network with independent library sources is the highest popularity first (HPF) scheme in which the highest popular files are placed into the cache [6]. In cases where files have no priority over each other and demand popularity is uniform, some files are randomly allocated to memory (random placement) according to the memory size.

In this chapter, we show that the gain of the conventional random placement in a single-cache network with independent sources does not carry over to a single-cache network with correlated content; thus, an efficient solution is still needed for such a setting.

We address the caching strategy and examine the trade-off between the delivery rate and the memory size from an information-theoretic perspective. As content placement is the key challenge in such networks, we first introduce a clustering scheme to extract the efficient side information for the entire library during the placement phase considering the similarity among content and the maximum distortion constraint in the system.

Then, we formulate the expected delivery rate by joint consideration of the rate-distortion function and caching strategy, where the limit for the maximum allowable distortion in the system is determined based on the Lagrange multipliers technique and reverse water-filling optimization. Our extensive simulations validate the proposed scheme, which provides a considerable boost in network efficiency compared to the legacy caching scheme.

In the remainder of this chapter, we first describe the system model and introduce the correlation-aware clustering scheme for the placement phase. Next, we describe the caching and delivery strategy and analyze the delivery rate of the system.

4.2. System Model

In this chapter, we study a centralized cache-aided delivery network consisting of a server, multiple users, and a single cache over a shared error-free broadcast link. The network model is illustrated in Figure 4.1.

All N users have access to a single shared cache of size $M = K_{\boldsymbol{C}}$ files, where $K_{\boldsymbol{C}}$ is the number of representatives determined by the clustering solution $\boldsymbol{C}$ upon performing the proposed clustering scheme in the placement phase.

The proposed caching strategy operates in two phases; in the placement phase, the clustering scheme is performed to identify the side information for placing into the shared cache. Then in the delivery phase, encoded messages are transmitted as refinement segments enabling users to reconstruct their requested files by jointly decoding the received message and the side information in the shared cache.

The server has access to a library of m uniformly popular content files $f_i \in \mathcal{F} = \{f_1, \dots, f_m\}$ with the same length. In line with studies on lossy caching scenarios [119][120], the library content files in this chapter are assumed to be zero-mean correlated gaussian sources. Thus, we have generated m correlated gaussian random variables $f_i \sim \mathcal{N}(0, \Sigma)$, where covariance matrix Σ is given by

$$\begin{pmatrix} \sigma_1^2 & \rho_{12}\sigma_1\sigma_2 & \cdots & \rho_{1m}\sigma_1\sigma_m \\ \rho_{12}\sigma_1\sigma_2 & \sigma_2^2 & \cdots & \rho_{2m}\sigma_2\sigma_m \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{1m}\sigma_1\sigma_m & \rho_{2m}\sigma_2\sigma_m & \cdots & \sigma_m^2 \end{pmatrix} \tag{4.1}$$

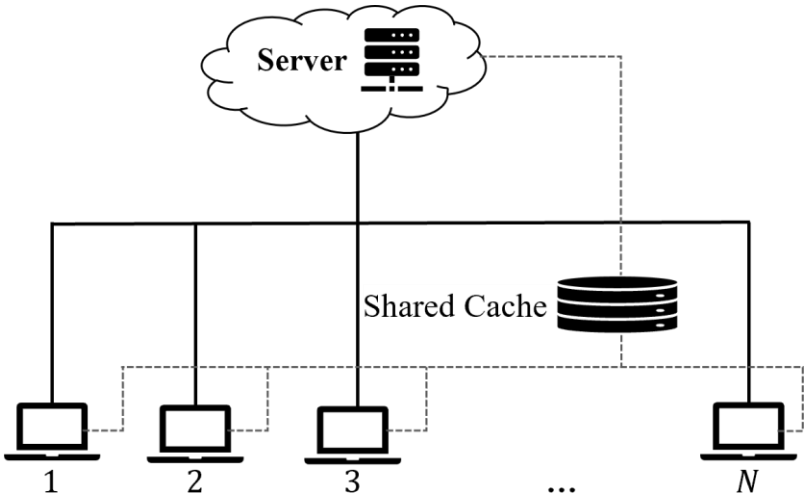


Figure 4.1: Cache-aided delivery network model with a single shared cache

4.3. Correlation-Aware Clustering Scheme (CACS)

The CACS aims to extract the most efficient side information for the entire library during the placement phase taking into account the similarity among sources and the maximum allowable distortion in the system. The application of this scheme is not limited to the correlated gaussian sources, and it can also work with correlated binary sources considering an appropriate distance measure.

The CACS is designed to provide a sufficient number of compact and well-separated clusters with approximately the same maximum distance per cluster without requiring prior knowledge of the exact number of clusters.

Achieving clusters with approximately the same maximum distance is useful to introduce the optimum allocation of the maximum allowable distortion to the files and reduce the transmission rate. Later in the delivery phase, we discuss this point more extensively.

4.3.1. CACS Methodology

The CACS operates in two general steps. First, the AI-based optimizer module assigns an initial cluster number to each data point to form a primary clustering. The primary clusters are then re-clustered, merged, and modified based on the designed condition described in Eq. (3.3) to compensate for the distortion deviation between clusters of different sizes and improve the result.

This modification leads clusters to gradually achieve the same maximum distance without increasing the number of clusters. Following that, the AI-based optimizer module optimizes the clusters in accordance with the objectives over iterations.

The cluster representative is selected in two consecutive steps; First, a centroid (approximated center) is identified within each cluster. Then, one actual point within the clusters with the least distance to the centroid is selected as the representative. In binary sources, a centroid is determined according to the majority rule, where a decision is made based on the majority of bits in each position [Chapter 3, section 3.6.1].

The CACS can adopt any appropriate distance measure based on the problem settings; In this regard, we consider the squared error measure for gaussian content files in this chapter and the hamming measure for binary content files in Chapters 5 and 6.

4.3.2. CACS Optimizer Module

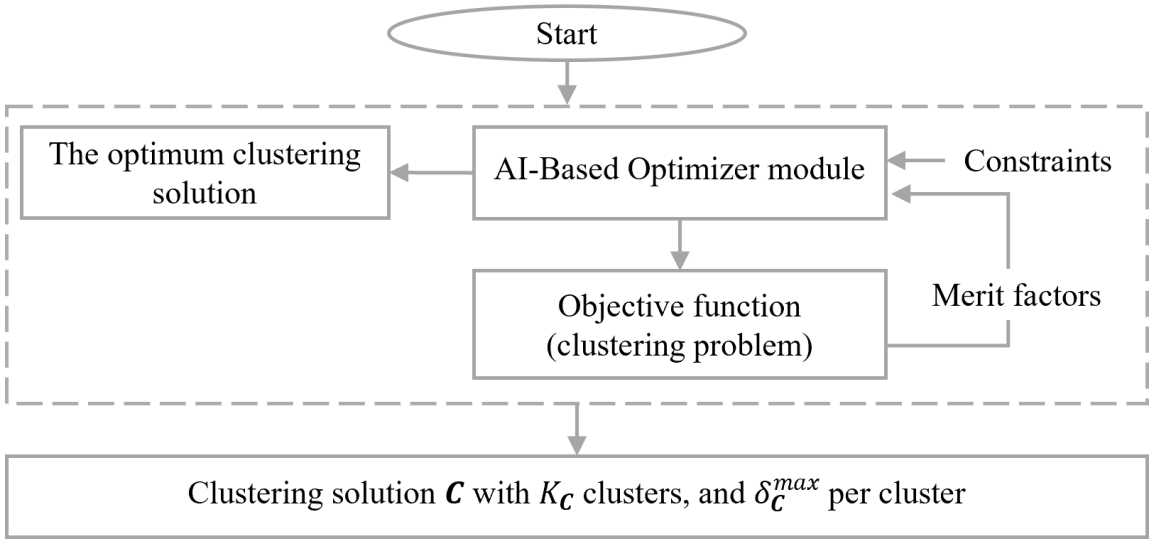
The AI-based optimizer module stands at the highest level of the proposed approach and considers clustering as a problem that must be minimized over the course of iterations.

The AI-based optimizer module checks combinations of the input to determine which candidate solution yields the minimum output of the objective function. The optimizer module creates binary vectors of length $m \times L$ bits as a candidate solution to be assigned to the particles in the utilized binary optimization algorithm. L is the number of bits that are required to define a cluster number as a binary address (BA) for the optimizer module and is calculated as $L = \log_2 K$, where K is the number of clusters. The candidate solution then converts to the decimal address equivalent (DA) to form the initial clustering.

The number of clusters is initialized by $K = \lfloor \sqrt{m} \rceil$. However, the value of K will be optimized as needed over the merging and modifying steps to comply with the maximum distortion constraint for the system while clusters obtain the same maximum distance. If the constraint still allows for more distance within each cluster, it means the number of clusters can be reduced to meet the condition. While if the maximum distance per cluster exceeds the distortion constraint, the number of clusters should be increased to meet the condition.

As shown in the general framework, the proposed clustering can be implemented using a variety of binary optimization algorithms in the AI module. Here we used the binary bat algorithm [83] in the AI module due to its excellent performance, which is comprehensively discussed and analyzed in Chapter 3.

Flowchart 4.1 shows how the AI module and objective function collaborated to solve the clustering problem in the proposed CACS.



Flowchart 4.1: How the optimizer module and objective function collaborated to solve the clustering problem in the proposed CACS considering the given constraints of the system

4.3.3. CACS Objective Function and Formulation

The CACS is formulated as a single objective AI-based optimization problem with objective function $\boldsymbol{f}$ that needs to be minimized iteratively over the set of all feasible clustering solutions, denoted by $\psi = \{C^1, C^2, \dots, C^{S(m,K)}\}$, where $S(m, K)$ is defined by Eq. (3.3).

Our goal is to find the clustering solution $\boldsymbol{C}$ with $K_{\boldsymbol{C}}$ clusters and maximum distance $\delta_{\boldsymbol{C}}^{max}$ within each cluster, where $\boldsymbol{f}(\boldsymbol{C}) = \text{Min}\{\boldsymbol{f}(C) \mid C \in \psi\}$. To this end, we define $G(C)$ and $E(C)$ as two functions reflecting the criteria to ensure the compactness and the separation of the clusters considering the given maximum distortion constraint $\boldsymbol{\delta}$ in the system.

The objective function $\boldsymbol{f}$ is defined as follows

$$\begin{aligned} &\boldsymbol{f}(C) = \underset{C\in\psi}{\text{Min}}\left(\frac{G(C)}{E(C)}\right) \\ &s.t.\ \delta_C^{max} \leq \boldsymbol{\delta} \end{aligned} \tag{4.2}$$

Such that

$$\underset{C\in\psi}{G(C)} = K_C\ .\ \bar{d}_C^{max}.\Delta_C \tag{4.3}$$

$$\underset{C\in\psi}{E(C)} = \frac{1}{K_{\boldsymbol{C}}}\sum_{i=1}^{K_{\boldsymbol{C}}} E_{C_i}^{min} \tag{4.4}$$

Where $\bar{d}_C^{max}$, Δ_C, and $E_{C_i}^{min}$ are defined by equations (3.5), (3.6), and (3.7), respectively, in the general clustering framework in Chapter 3.

Below are the summarized steps of the objective function algorithm followed by the binary optimization algorithm utilized in the optimizer module.

Objective Function Algorithm:

Input: *candidate solution*, $\mathcal{F}$, m , L , $\boldsymbol{\delta}$

1: Cluster initialization based on the DA assigned to each data point

2: Determining clusters' representatives based on the two-step rule and updating the clusters based on recent changes

3: **if** the current clustering solution is different from the previous one

4: *flag*=0

5: **while** *flag*=0 **do** (2)

6: **else**

7: find $\delta^{c_i}, \forall\, c_i \in C, \forall i \in \{1,2,\dots,K_C\}$
find $\underset{c_i \in C}{\text{Max}}\ \{\delta^{c_i}\}$

8: **end if**

9: **while** $\forall\, c_i \in C,\ \delta^{c_i} \leq 0.9\ \underset{c_i \in C}{\text{Max}}\ \{\delta^{c_i}\}$

10: Merge clusters w.r.t the maximum distortion $\boldsymbol{\delta}$ constraint

11: **do** (2) & (3)

12: **end while**

13: Calculate K_C, δ_C^{max}

14: Calculate $\bar{d}_C^{max}.\Delta_C$

15: Calculate $\underset{C\in\psi}{G(C)} = K_C\ .\ \bar{d}_C^{max}.\Delta_C$

16: Calculate $\underset{C\in\psi}{E(C)} = \frac{1}{K_C}\sum_{i=1}^{K_C} E_{C_i}^{min}$

17: Calculate $\boldsymbol{f}(C) = \frac{G(C)}{E(C)}$

Output over the course of iterations: $\boldsymbol{C}$, K_C, δ_C^{max}

Binary bat Algorithm:

Initialize: the bat population $\mathbb{X}_i = (1, \dots, n)$, $V_i = 0$

Define pulse frequency Fr_i

Initialize pulse rate r_i and the loudness A_i

while $(t < Max_{iterations})$

  Generate new solutions by adjusting frequency and updating velocities and positions

  if $(rand > r_i)$

    Select a solution *(gbest)* among the best solutions randomly

    Change the dimensions of the positions vector in line with the dimensions of *gbest*

  end if

  Generate a new solution by flying randomly

  if $(rand < A_i \& \boldsymbol{f}(x_i) < \boldsymbol{f}(gbest))$

    Accept the new solutions

    Increase r_i and reduce A_i

  end if

Rank the bat and find the current $Gbest$

end while

4.4. Proposed Caching and Delivery Scheme

Here we describe the caching strategy in two phases: placement and delivery.

4.4.1. Placement Phase

During the placement phase, the CACS is performed and the $K_{\boldsymbol{C}}$ representatives corresponding to the achieved clustering solution $\boldsymbol{C}$ are identified as the selected side information for the rest of the library content. The set of all representatives of the clustering solution $\boldsymbol{C}$ is denoted by $\boldsymbol{F} = \{\hat{F}_1, \dots, \hat{F}_{K_{\boldsymbol{C}}}\}$. The selected side information is then placed into a shared cache with size $M = K_{\boldsymbol{C}}$ to minimize peak delivery rate.

Figure 4.2 presents a simple illustration of the placement phase.

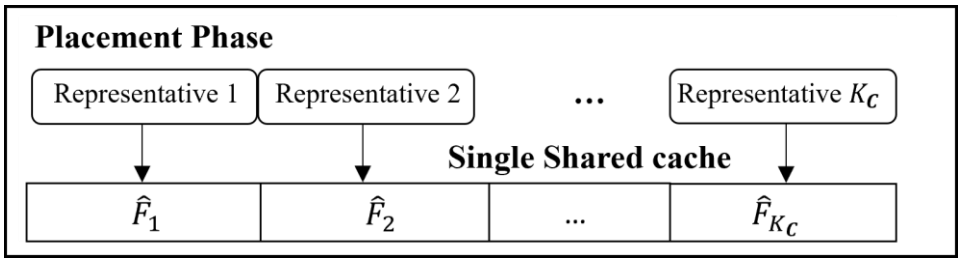


Figure 4.2: Placing the set of achieved representatives as the selected side information into the shared cache

4.4.2. Delivery Phase

In the delivery phase, demands are revealed to the server. Each user uniformly requests a file $f_i^j \in \mathcal{F}$ from the server, where $i \in \{1,2,\dots,m\}$ is the index of the requested file and $j \in N$ is the index of the requesting user.

Let $F_{k,i}$ be the notation of the file f_i after clustering, where $k \in K_C$ and shows the file f_i is located in which cluster. As the server has global knowledge of the populated caches and clustered files, such demands will be satisfied by transmitting an encoded message X_i to the user $j \in N$ in response to each requested file $F_{k,i}^j$. The size of encoded messages is $h\left(F_{k,i}^j|\hat{F}_k\right) \leq h\left(F_{k,i}\right)$, where $\hat{F}_k$ denotes the relevant representative, and $h(.|.)$ describes the conditional entropy.

These messages are generated in order to enable users to reconstruct their requested files by jointly decoding the received message and the side information available in the single shared cache.

The expected delivery rate of the system is formulated in Theorem 4.1 based on a joint consideration of the rate-distortion function and the caching strategy, where the limit for the maximum allowable distortion at the receivers is determined based on the Lagrange multipliers technique and reverse water-filling algorithm.

Theorem 4.1. The expected delivery rate of the proposed system considering a clustering solution $\boldsymbol{C}^S$, with $K_{C}s$ clusters and m_k files per cluster $k \in K_{C}s$, during the placement phase is as follows

$$\mathbb{E}\{R(D)\} = \frac{1}{K_{C^S}} \sum_{k=1}^{K_{C^S}} \frac{1}{m_k} \sum_{i=1}^{m_k} \frac{1}{2} log\left(\frac{\delta_{k,i}^2}{D_{k,i}}\right) \tag{4.5}$$

Where

$$D_{k,i} = \begin{cases} \lambda & if\ \lambda < \delta_{k,i}^2, \\ \delta_{k,i}^2 & if\ \lambda \geq \delta_{k,i}^2, \end{cases} \tag{4.6}$$

Where λ is chosen to minimize the total distortion as well as the expected delivery rate.

Proof. Consider the case of a clustering solution $\boldsymbol{C}^S$ with K_{C^S} clusters in the placement phase, i.e., $\boldsymbol{C}^S = \left[C_1^S, C_2^S, \dots, C_{K_{C^S}}^S\right]$, where each cluster $k \in K_{C^S}$, includes a subset of library content and a representative. Based on the optimizer module and the objective function of the clustering scheme in our system design, files within a cluster have a random distance with respect to the cluster representative and could be considered independent normal random variables $F_{k,1}, F_{k,2}, \dots, F_{k,m_k}$ with regard to the cluster representative, $\hat{F}_k$; where $F_{k,i} \sim \mathcal{N}\left(0, \delta_{k,i}^2\right)$, and $\delta_{k,i}^2$ is a function of δ_k^{max}, the maximum distance within a cluster k.

The rate-distortion function [81] for our clustering solution can be written as

$$R(D) = \min_{\mathbb{E}d(F_k, \hat{F}_k) \leq D} I\left(F_k; \hat{F}_k\right) \tag{4.7}$$

The distortion measure is the squared error distortion, and it is considered between the clustered files and the corresponding cluster representative.

Then, the mutual information function can be expanded as

$$I\left(F_k; \hat{F}_k\right) = h(F_k) - h\left(F_k | \hat{F}_k\right)$$

Considering m_k files within a cluster k

$$\geq \sum_{i=1}^{m_k} h\left(F_{k,i}\right) - \sum_{i=1}^{m_k} h\left(F_{k,i} | \hat{F}_k\right)$$

$$= \sum_{i=1}^{m_k} I\left(F_{k,i}; \hat{F}_k\right)$$

$$\geq \sum_{i=1}^{m_k} R(D_{k,i})$$

$$= \sum_{i=1}^{m_k} \frac{1}{2} log\left(\frac{\delta_{k,i}^2}{D_{k,i}}\right) \tag{4.8}$$

Where $D_{k,i} = E\left(F_{k,i} - \hat{F}_k\right)^2$. As for the number of files within each cluster, i.e., m_k, it is considered a random variable during the clustering phase. However, upon successful clustering, m_k can be considered as a constant parameter for each cluster k in the determined clustering solution $\boldsymbol{C}^S$.

Lagrange Multipliers Technique. To evaluate the optimal λ that achieves the objectives of minimizing the total distortion and the delivery rate, we have used the Lagrange multipliers technique. Considering Eq. (4.8), the function can be written as

$$J(D) = \sum_{i=1}^{M_k} \frac{1}{2} \log\left(\frac{\delta_{k,i}^2}{D_{k,i}}\right) + \lambda \sum_{i=1}^{M_k} D_{k,i}$$

Differentiating with respect to $D_{k,i}$ and setting equal to zero gives

$$\frac{\partial J}{\partial D_{k,i}} = -\frac{1}{2}\frac{1}{D_{k,i}} + \lambda = 0$$

This implies $D_{k,i} = \lambda'$ that is a very interesting finding, wherein the optimum allocation of delivery rate is achieved by considering equal distortion for all files. Based on Eq. (4.8), it is observed that the optimization is achieved if $D_{k,i} = \lambda' \leq \delta_{k,i}^2$ for all i. Alternatively, if the distortions $D_{k,i}$ increase so does λ' until it exceeds $\delta_{k,i}^2$, as in Eq. (4.6).

The concept can be interpreted according to the reverse water-filling, see Figure 4.3. This implies that the delivery rate is available only for the files with variances greater than constant λ. No delivery is expected for the files with a variance lower than λ. In other words, λ reflects the limit of the maximum allowable distortion introduced to the files. ■

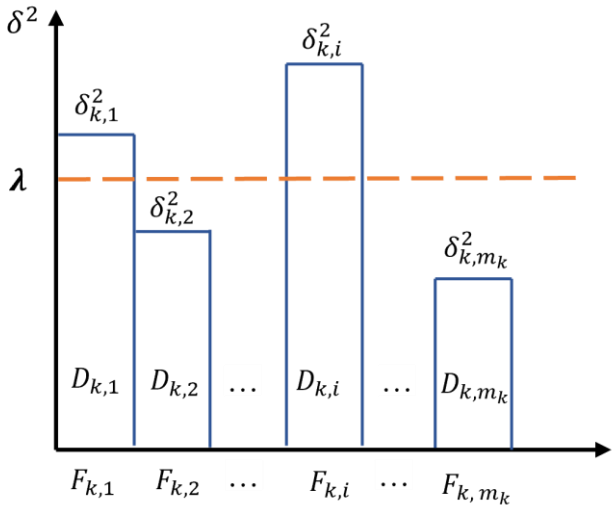


Figure 4.3: Reverse water-filling algorithm in delivery rate optimization for clustered files in a cluster k

4.5. Performance Analysis and Discussion

We have evaluated the performance of the proposed caching scheme in this section, where the conventional random placement strategy is used as the benchmark.

According to our problem settings, the library content files are considered zero-mean correlated gaussian random variables. In order to simulate the library content files, we have first generated a vector of uncorrelated gaussian random variable W and then multiplied it by a matrix φ, where $\varphi\varphi^T = \Sigma$ and Σ is the desired covariance matrix. It should be noted that φ can be created by using the Cholesky decomposition of Σ, or from the eigenvalues and eigenvectors of Σ.

We have used the MASS package library in the R programming language[7] for this experiment.

We have considered library content files with at least 0.75 similarities among library content for the following experiments.

Figure 4.4 compares the delivery rate-memory trade-off between the conventional random placement and the proposed caching schemes, considering different threshold values λ. The

[7] https ://www.geeksforgeeks.org/simulate-bivariate-and-multivariate-normal-distribution-in-r

memory size describes the total number of files placed in the cache during the placement phase, whereas the expected delivery rate represents the average delivery rate per requested file per user during the delivery phase.

It can be seen that the delivery rate of both the conventional approach and the proposed scheme decreases as the memory size increases. However, our proposed scheme requires much lower delivery rates for the same memory size to accomplish the delivery phase.

In Figure 4.4, we have also evaluated the effect of increasing the distortion threshold in the proposed scheme, which implies allowing some distortion levels in the received files on the user side. The threshold is considered based on the maximum proportion of the distortion introduced to the corresponding file. As the threshold increases, the delivery rate decreases, which is due to the allowable distortion that is reflected on the received files. The trade-off between the expected distortion and memory size is investigated as follows.

Figure 4.5 shows the trade-off between the users' expected distortion and the memory size in the proposed caching scheme and the conventional approach considering different fixed delivery rates. The expected distortion implies the average of the distortion introduced per requested file per user, and the memory size describes the total number of files placed in the cache during the placement phase.

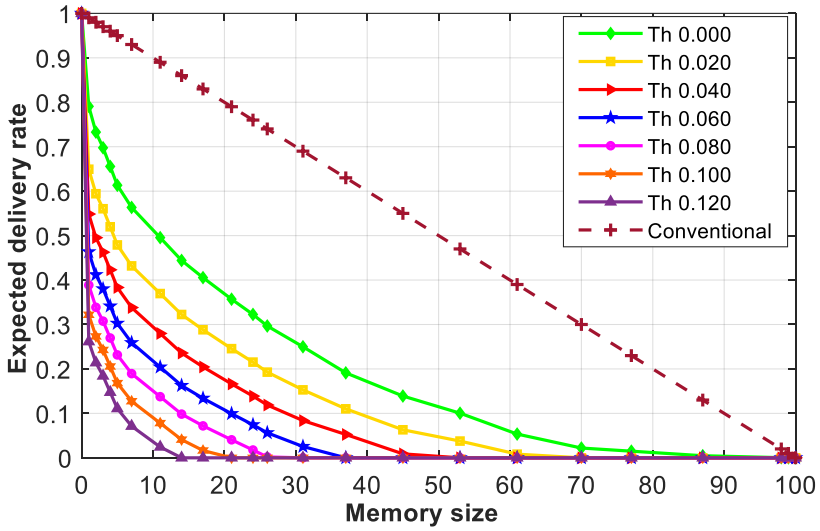


Figure 4.4: Delivery rate memory trade-off in the proposed scheme compared to the conventional caching for N=100 users, m=100 files

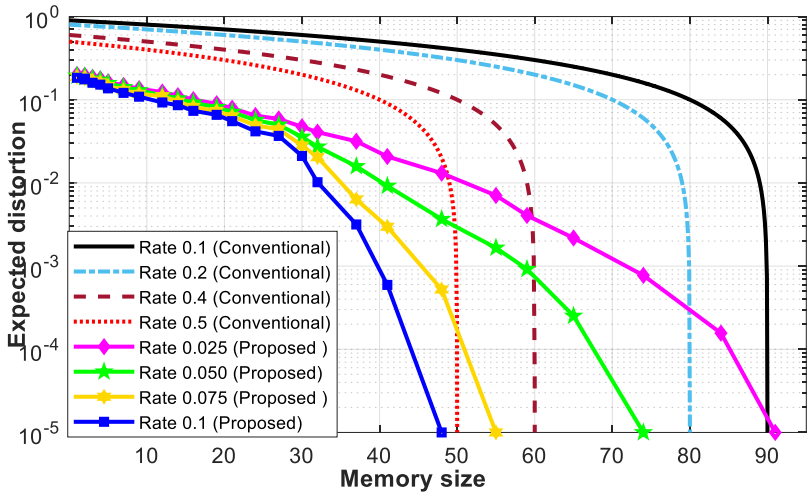


Figure 4.5: The expected distortion memory trade-off in the proposed scheme for different fixed delivery rates compared to the conventional approach for N=100 users, m=100 files

It can be seen that by increasing the memory size, the expected distortion drops in both approaches for a fixed delivery rate. However, our proposed scheme has yielded a significant reduction in the expected distortion for all different delivery rates compared to the conventional approach.

Another observation is that increasing the delivery rate leads to reducing the expected distortion for a fixed memory size in both schemes. Still, this reduction is considerably boosted in our proposed solution compared to the conventional scheme. For instance, for the expected delivery rate of 0.1, the expected distortion of our proposed scheme significantly drops to lower than 10^{-3} even for the memory size of 40 files, while the conventional scheme hardly reaches a distortion level 10^{-1}.

It is seen that the conventional approach has the highest expected distortion, even by increasing the delivery rate from 0.1 to 0.5. Meanwhile, the proposed approach reaches significantly better results by adopting a much lower delivery rate (e.g., 0.025 to 0.1).

4.6. Summary

This chapter investigates a cache-aided delivery network with correlated content and a single shared cache connected to multiple users to combat the high delivery data rates. The proposed placement scheme considers a correlation-aware clustering scheme that categorizes content files into clusters with approximately the same maximum distance by considering the similarity among the content files. During the delivery phase, encoded messages are delivered to users in order to meet all demands. We have formulated the expected delivery rate of the system by joint consideration of the rate-distortion function and caching strategy. The limit for the maximum allowable distortion of the system is determined based on the Lagrange multipliers technique and reverse water-filling optimization. Finally, we have analyzed the trade-off between the memory size, the delivery rate, and the users' expected distortion. Our simulation results show that the proposed caching scheme exhibits excellent performance in reducing the delivery rate and users' expected distortion compared to the conventional scheme.

Chapter 5

Content Delivery in a Network with Multiple Shared Caches and Correlated Content under Uniform Demand

5.1. Introduction

This chapter extends our previous study to a cache-aided network with multiple shared caches. Such systems are beneficial in current and next-generation wireless networks as they can be applied to small base station architecture. It allows users in an SBS to access the nearby caches and be served locally for some parts of demands to reduce the strain on the backhaul. Also, it is

useful in a HetNet environment or as an upper layer of hierarchical caching networks in IoT-based applications.

We examine the proposed network under a uniform popularity demand distribution in this chapter. To this end, the side information for the entire library is first extracted by using the correlation-aware clustering scheme discussed in Chapter 4. Next, we introduce the placement strategy based on the coded caching solution with uncoded placement. We describe the delivery phase by considering the transmission of the coded multicast messages and refinement segments. Following that, we discuss how increasing the number of users affects the peak delivery rate of our proposed system. Furthermore, we introduce the optimum library partitioning of the system formulated to minimize the peak delivery rate in the network.

In the remainder of this chapter, we first describe the system model and the caching and delivery strategy to analyze the delivery rate of the system. Then we introduce the optimum library partitioning of the system and present simulation results.

5.2. System Model

Consider a centralized cache-aided delivery network with a server, N total users, and Z SBSs over a shared error-free broadcast link. The network model is illustrated in Figure 5.1.

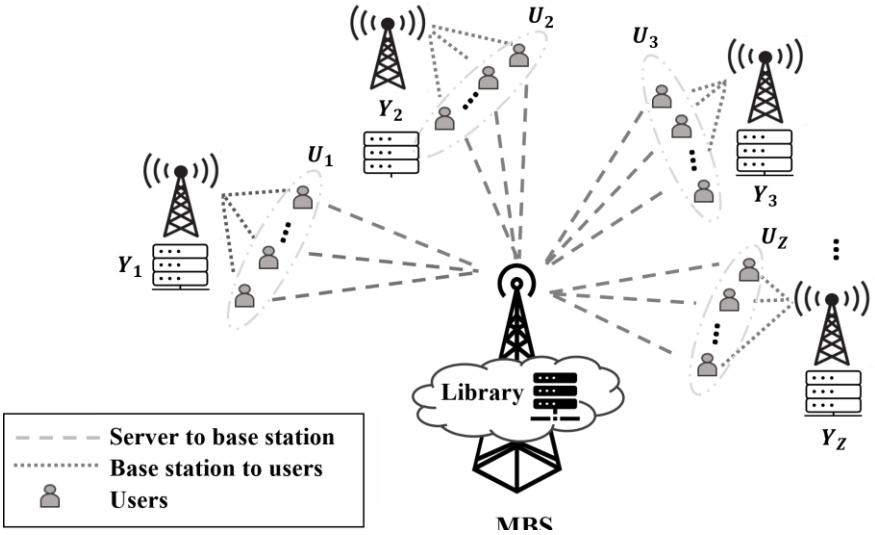


Figure 5.1: Cache-aided delivery network with multiple shared caches

Each SBS $i \in \{1, \dots, Z\}$ is equipped with a shared cache Y_i, which is connected to U_i number of users, where $U_i \in \{U_1, \dots, U_Z\}$ and $\sum_{i=1}^{Z} U_i = N$, and $Z \leq N$. It is assumed that each user is connected to only one SBS and can receive messages from its SBS as well as the server.

The proposed caching strategy operates in two phases; in the placement phase, the CACS is performed to identify the side information for placing into the shared caches according to a coded caching strategy with uncoded placement. Then in the delivery phase, multicast coded messages and refinement segments are transmitted to enable users to reconstruct their requested files by jointly decoding the received message and the side information in the shared cache.

The server has access to a library of m uniformly popular content files $\mathcal{F} = \{1, \dots, m\}$ with the same length. In line with studies on coded caching scenarios [5][6], the library content files in this chapter and the next chapter are assumed to be binary content files. In this regard, the library with correlated content could be modeled according to the classical binary symmetric channel (BSC) correlation model with crossover probability $\rho_0 \in [0,0.5]$.

Let $\tilde{n}$ be an i.i.d binary random variable generated according to the Bernoulli distribution $\tilde{n} \sim Bern\,(1/2)$ and $(f_1, f_2, \dots, f_m)$ be the output of a set of BSC with crossover probability ρ_0 fed by the same input $\tilde{n}$. Thus, based on the model for correlated binary content files [17][18], each content file is represented by a vector of i.i.d binary symbols with the same length. Since symbols of the content files are correlated according to a joint distribution $\mathcal{P}_{\mathcal{F}}$, for a block length of B bits $f_i \in \mathbb{F}_2^B$ and $H(f_i) = B$ bits, and $\forall f_i, f_j \in \mathcal{F}$, $H(f_i|f_j) \leq B$; therefore $H(f_1, \dots, f_m) \leq mB$ bits. $\mathbb{F}_2^B$ denotes the set of binary sequences of length B bits.

We consider $\rho = 1 - \rho_0$ as the correlation parameter in this chapter. It is clear that ρ_0 close to zero indicates highly correlated sources while ρ_0 close to 0.5 indicates roughly no correlation between binary sources. This correlation model can describe different communication scenarios in which sources share common information, but each also has an individual component (e.g., A wireless network in which a set of nodes collect and transmit correlated data arising from the same physical phenomenon to a common sink) [121][122].

5.3. Proposed Caching and Delivery Scheme

We present the caching strategy in two following phases:

5.3.1. Placement Phase

First, upon performing the CACS, the K clusters' representatives are identified as the selected side information for the rest of the library. Then the placement phase is carried out according to the CC strategy with uncoded placement to take advantage of the multicast opportunities. Contrary to the CC scheme, we only store representatives in our model. However, the rest of the library (clustered files) will also be accessible at a low delivery rate by transmitting refinement segments to the requesting users since delivery of the clustered files has been formulated as distributed source coding with side information.

Let parameter T define integer values $T \in [0, Z]$. Then, the available memory size of each shared cache to store the obtained K representatives is defined as $M \in KT/Z$, i.e., $M \in \{0, K/Z, 2K/Z, \dots, K\}$.

Recall that $\boldsymbol{F} = \{\hat{F}_1, \dots, \hat{F}_K\}$ denotes the set of K representatives. We divide each $\hat{F}_k$, $k \in K$, into $\binom{Z}{T}$ non-overlapping chunks of size $1/\binom{Z}{T}$. The chunks of each $\hat{F}_k$ are labeled as follows

$$\hat{F}_k = (\hat{F}_{k,\tau} : \tau \subset [Z], |\tau| = T) \tag{5.1}$$

where $T = ZM/K$ and $[Z] \triangleq \{1, \dots, Z\}$

Then, each shared cache $z \in [1, Z]$ fills its caches as follows:

$$Y_z = (\hat{F}_{k,\tau} : k \in [\boldsymbol{F}], \tau \subset [Z], |\tau| = T, z \in \tau) \tag{5.2}$$

In other words, each cache Y_z stores all chunks $\hat{F}_{k,\tau}$ if $z \in \tau$.

In this way, each shared cache stores $K\binom{Z-1}{T-1}$ number of representative chunks in total. In this case $K\binom{Z-1}{T-1}\frac{1}{\binom{Z}{T}}$ memory size is required as the size of each chunk is $1/\binom{Z}{T}$ file, which results in $K\binom{Z-1}{T-1}\frac{1}{\binom{Z}{T}} = \frac{KT}{Z} = M$ files and satisfy the memory size constraint.

The following example shows how a set of side information is placed in the shared caches.

Example 5.1: Consider the proposed network with $Z = 4$ shared caches, each with size $M = 1$ file. Assume the library files are categorized into $K = 4$ clusters by performing the CACS scheme, which results in a representative set $\boldsymbol{F} = \{\hat{F}_1, \dots, \hat{F}_4\}$.

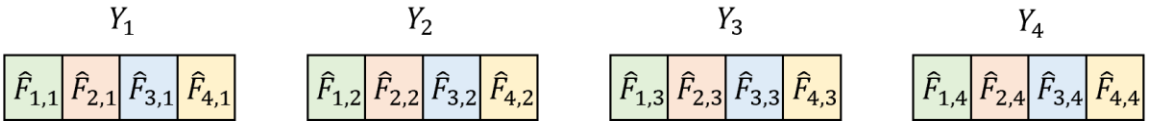
Since $T = M\,Z/K = 1$, each $\hat{F}_i$ is split into $\binom{4}{1} = 4$ chunks of size $1/4$ file. Hence, caches are filled as follows:

$Y_1 = \{\hat{F}_{1,1}, \hat{F}_{2,1}, \hat{F}_{3,1}, \hat{F}_{4,1}\}$

$Y_2 = \{\hat{F}_{1,2}, \hat{F}_{2,2}, \hat{F}_{3,2}, \hat{F}_{4,2}\}$

$Y_3 = \{\hat{F}_{1,3}, \hat{F}_{2,3}, \hat{F}_{3,3}, \hat{F}_{4,3}\}$

$Y_4 = \{\hat{F}_{1,4}, \hat{F}_{2,4}, \hat{F}_{3,4}, \hat{F}_{4,4}\}$



The next section explains how the delivery phase works according to this placement setting.

5.3.2. Delivery Phase

During the delivery phase, users reveal their requests to the SBSs. Depending on the cache size and requested files, some of the demands can be locally satisfied, while others need to be processed by the server. Demands received by the server are either associated with representatives or clustered files. In both cases, the relevant representative should be first constructed.

Let Q_z includes all the demands of the cache Y_z, $z \in Z$, to be processed by the server. Then, all files in a Q_z are mapped to the relevant representatives and create a vector of unique demanded representatives across all caches denoted by Q in the server.

All unique requested clustered files across all SBSs are also assigned to the demand vector Q'. Lastly, requests will be satisfied in one of the following ways:

1. Demands corresponding to the representatives (Demand vectors Q):

- When $M = K$: such demands are locally served as they are fully cached and accessible by all users.
- When $M < K$: Such demands are served by transmitting coded multicast messages in accordance with the cached content. Consider a case where one distinct representative is requested in each shared cache, therefore; request vector $Q = (q_{\hat{F}_1}, \dots, q_{\hat{F}_Z})$ is created, where $q_{\hat{F}_i}$ means the representative $\hat{F}_i$ is requested in cache $i \in Z$. Consider subset $\mathcal{S} \subset [Z]$ of cardinality $|T + 1|$ shared caches. In this case, every T cache in $\mathcal{S}$ shared a chunk in their memory that is needed at the rest of the shared caches in $\mathcal{S}$. Therefore, the server transmits the following coded message

$$\bigoplus_{z \in \mathcal{S}} q_{\hat{F}_S}, \mathcal{S}\backslash\{z\}, \forall \mathcal{S} \subseteq [\mathrm{Z}]\,, |\mathcal{S}| = \mathrm{T} + 1 \tag{5.3}$$

Where $\oplus$ indicates the bitwise XOR operation and $\mathcal{S}$ is the subset of shared caches that receives the coded message. The size of each coded message is $1/\binom{Z}{T}$ file.

Example 5.2 shows the transmitted coded multicast messages during the delivery phase.

2. Demands corresponding to the clustered files (Demand vector Q'):

All unique requested clustered files across all SBSs are assigned to the demand vector Q'. As the server has global knowledge of the populated caches and the clustered files, let such files in the Q' be described by their cluster numbers for convenience, i.e., $F_{k,i}$, where $i \in m_k$ and $k \in K$. Also, let $F_{k,i}^{j_z}$ describes the requested file $F_{k,i}$ by the user $j \in N$ connected to cache $z \in Z$.

Then, demands corresponding to the clustered files will be satisfied by transmitting a refinement segment in addition to the coded multicast messages. To this end, in response to each request in Q', the server creates an encoded message X_i of size $H\left(F_{k,i}^{j_z} | \hat{F}_k\right) \leq H\left(F_{k,i}\right)$ to be transmitted to the requesting user, where $H(.|.)$ denotes the conditional entropy, and $\hat{F}_k$ is the corresponding representative. As such, users can reconstruct the desired content by jointly decoding the refinement segment and the side information available in the cache.

It is clear that the refinement segment needed for the same file in different SBSs is transmitted just once due to the broadcast nature of the medium.

The following example shows how the delivery phase works in accordance with the populated caches in the network. We have considered a simple case in this example to focus on the delivery solution.

Example 5.2: Consider the proposed cache-aided network with $Z = 2$ shared caches Y_i, for $i = [1, 2]$, connected to $U_1 = 14$ and $U_2 = 16$ users, results in $N = \sum_{i=1}^{2} U_i = 30$ total users. Let the size of the cache be $M = 1$ file. Assume the server consists of $m = 30$ uniform popular files with size B bits that are categorized into $K = 2$ clusters by performing the CACS scheme results in a representative set $\boldsymbol{F} = \{\hat{F}_1, \hat{F}_2\}$ and maximum distance $\delta_{max} = 0.2$ per cluster.

Therefore, $T = M\,Z/K = 1$ and each $\hat{F}_i$ is split into $\binom{2}{1} = 2$ chunks of size $1/2$ file. Hence, the caches are filled as follows: $Y_1 = \{\hat{F}_{1,1}, \hat{F}_{2,1}\}$, and $Y_2 = \{\hat{F}_{1,2}, \hat{F}_{2,2}\}$

In this example, we investigate the delivery rate for the worst-case demand, which assumes all the files are requested in the network. Therefore, both representatives are needed in each cache, requiring the following coded multicast messages

Coded multicast messages (with size $B/2$)

$$\text{decode } \hat{F}_{12} \;\Leftarrow\; \hat{F}_{12} \oplus \hat{F}_{11} \;\Rightarrow\; \text{decode } \hat{F}_{11}$$

$$\text{decode } \hat{F}_{22} \;\Leftarrow\; \hat{F}_{21} \oplus \hat{F}_{22} \;\Rightarrow\; \text{decode } \hat{F}_{21}$$

Therefore, $R_{CM} = 2 * B/2 = B$ bits or 1 file.

Refinement segments (with size $B/5$):

$\left(F_{k,i} \middle| \hat{F}_k\right) = B/5$ for all the clustered files. Therefore, $R_{RS} = 28 * B/5 = 5.6$ files

Thus, $R = R_{CM} + R_{RS} = 6.6$ files are required to serve all 30 users with 30 unique requests in the proposed network.

5.4. Delivery Rate Analysis

The delivery rate of the proposed system includes two components; R_{CM} which is needed for constructing the representatives and the R_{RS} which is required for the refinement segments. Hence, the total delivery rate of the proposed system is

$$R = R_{CM} + R_{RS} \tag{5.4}$$

Coded multicast messages account for most of the delivery rate; therefore, the peak delivery rate occurs when

- firstly, requests from each SBS involve all clusters; thus, all representatives must be constructed in all SBSs via coded multicast messages.
- Secondly, all the clustered files are also requested in the network, for which we need to assume $N = m$, implying that the total number of users and files is the same.

In order to analyze the peak delivery rate, the upper bounds for R_{CM} and R_{RS} are described next.

Consider the case of reaching the clustering solution $\boldsymbol{C}^S$ with $K_{C}s$ clusters during the placement phase. Let ζ describe the maximum number of requested representatives per shared cache in the system. The coded multicast delivery rate R_{CM} in a network with Z shared caches, each having a memory size of $M = K_{\boldsymbol{C}}sT/Z$ files for $T \in [0: Z]$ to construct a set of $\zeta \in [1: K_{\boldsymbol{C}}s]$ representatives in all receivers is given by

$$R_{CM}(M) = Z\zeta\left(1 - \frac{M}{K_{\boldsymbol{C}}S}\right)\min\left(\frac{1}{1+ Z\, M/K_{\boldsymbol{C}}S}, \frac{K_{\boldsymbol{C}}S}{Z\zeta}\right) \tag{5.5}$$

In cases where $\zeta = K_{\boldsymbol{C}}s$, the upper bound for R_{CM} is achieved, representing the case when requests from each SBS involve all clusters, and therefore, all representatives must be constructed in all SBSs via coded multicast messages.

The coded multicast delivery rate R_{CM} can be achieved by treating each of the ζ sets of representative demands independently and then applying the coded multicast scheme proposed in

[5, Theorem 1] for each set of representative demands. The second term in the minimum function is considered for the cases when multicasting does not improve the unicast rate.

Since we are interested in the peak rate R_{CM}, we assume that all SBSs request all representatives, i.e., $\zeta = K_{\boldsymbol{C}}s$, which results in the peak rate R_{CM}^{P} as follows

$$R_{CM}^{P}(M) = ZK_{\boldsymbol{C}}s\left(1 - \frac{M}{K_{\boldsymbol{C}}s}\right)\min\left(\frac{1}{1+ Z\,M/K_{\boldsymbol{C}}s}, \frac{1}{Z}\right) \tag{5.6}$$

The delivery of the clustered content occurs by transmitting refinement segments according to the representatives since the caching problem is formulated as distributed source coding with side information at the decoder. Such messages describe the difference between the requested content and the cached representatives. Therefore, the peak delivery rate R_{RS}^{P} which upper bounds the delivery rate R_{RS} required for transmitting clustered content to the requesting users across all Z shared caches in the system is given by:

$$R_{RS} \leq R_{RS}^{P} = \sum_{k=1}^{K_{\boldsymbol{C}}s}\sum_{i=1}^{m_{\boldsymbol{k}}} H\left(F_{k,i}\middle|\hat{F}_{k}\right) \tag{5.7}$$

5.5. The Optimal Library Partitioning

The optimum library partitioning achieved by the CACS is formulated with the objective of minimizing the peak delivery rate R^{P} of the system considering constraint $\delta_{C}^{max} < \boldsymbol{\delta}$.

$$R^{P}(Z, M, m, \boldsymbol{F}, \boldsymbol{\delta}) = \min_{\delta_{C}^{max} < \boldsymbol{\delta}} \left\{ \left(ZK_{\boldsymbol{C}}s\left(1 - \frac{M}{K_{\boldsymbol{C}}s}\right)\min\left(\frac{1}{1+\ Z\,M/K_{\boldsymbol{C}}s}, \frac{1}{Z}\right)\right) + \sum_{k=1}^{\zeta}\sum_{i=1}^{m_{\boldsymbol{k}}} H(F_{k,i}|\hat{F}_{k}) \right\} \tag{5.8}$$

The first term of the min {.} function comes from the upper bound of the coded multicast messages for the worst-case demand $\zeta = K_{\boldsymbol{C}}s$. The second term represents the rate R_{RS}, required for the refinement segments under the worst-case demand assumption.

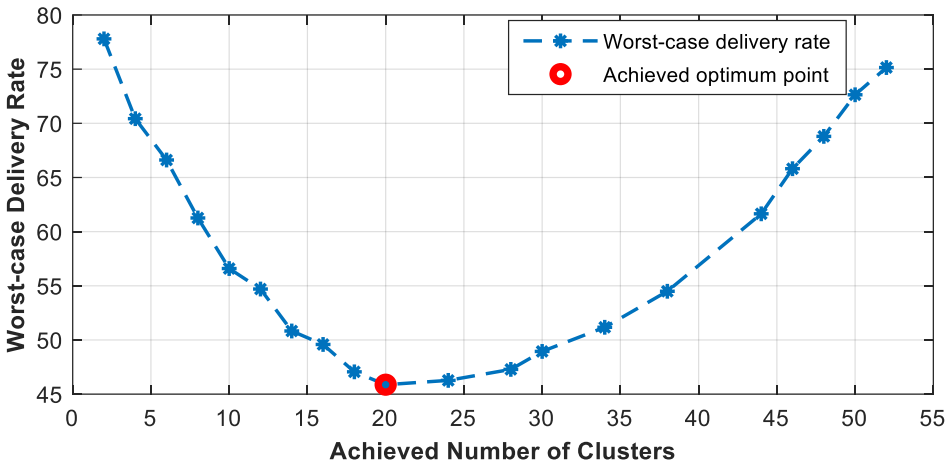


Figure 5.2: The minimum of the Peak delivery rate occurred in memory size $M = 20$ for a library of $m = 100$ files, categorized into $K_{CS} = [1{:}\,100]$ clusters with different $\delta_{max} \leq 0.231$

From this point of view, we are interested in finding the clustering solution C^* with parameters $K_{C^*}, \delta_{C^*}^{max}$ that minimize the R^P. Therefore, $M_{opt} = K_{C^*}$ is considered the optimum library partitioning for this setting.

It should be mentioned that the peak rate R^P is significantly impacted by the coded multicast messages in our model; hence, a balance should be maintained between the number of achieved representatives and the global cache size in the network during the placement phase in order to decrease the number of multicast messages in the delivery phase.

Figure 5.2 shows how the delivery rate decreases to a certain point by increasing memory size but rises again once it touches its minimum.

5.6. Performance Analysis and Discussion

The performance of the proposed caching and clustering schemes is evaluated here. The experimental results have been carried out on a PC with Windows 11 Professional 64-bit operating system, an Intel(R) Core ™ i7-10700K processor, and 48 GB RAM using MATLAB software 2021 b.

We start our evaluation with Figure 5.3, which illustrates the trade-off between the achieved number of clusters, the maximum distance within each cluster, and the similarity among the library sources in the correlation-aware clustering scheme. This scheme categorizes the library files into clusters with approximately the same maximum distance in all clusters.

This evaluation is performed on $m = 130$ files with different correlation level among the source files. This simulation is performed by considering the BBA algorithm in the optimizer module. The parameter setting for the BBA solution is described in Table 3.2.

It is observed that increasing the number of clusters reduces the maximum distance within the clusters. On the other hand, we can see the effect of having higher correlated sources in the library. Increasing the correlation among the library files results in clustering solutions with fewer groups and lower maximum distance in the clusters.

If the constraint of the system allows for more distance within each cluster, it means the number of clusters can be reduced to meet the condition. While if the maximum distance per cluster exceeds the system constraint, the number of clusters should be increased to meet the condition.

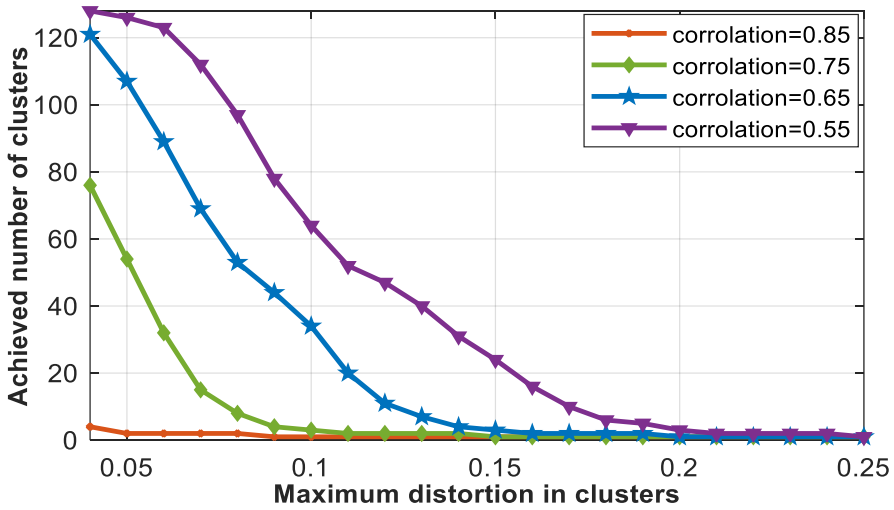


Figure 5.3: The trade-off between the achieved number of clusters and the maximum distance in the clusters for $m = 130$ files with different correlation level among sources

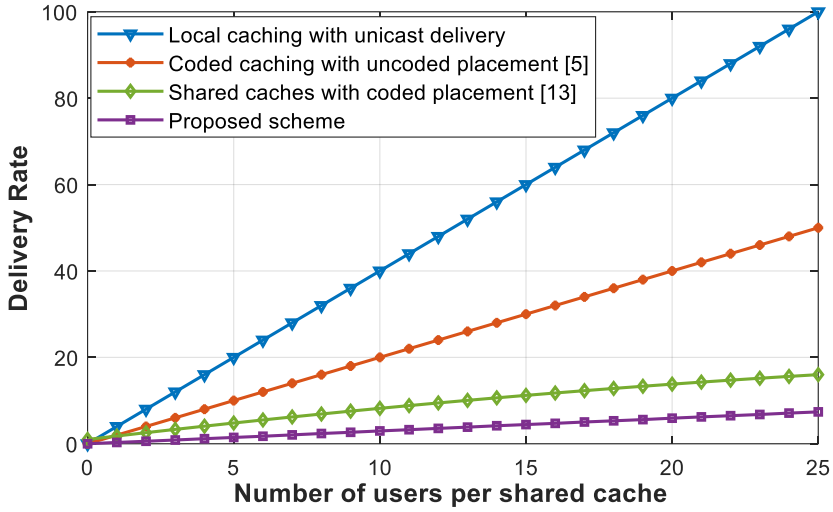


Figure 5.4: Delivery rate comparison in a network with $Z = 5$ shared caches of size $M = 20$

As shown in Figure 5.4, our next evaluation examines the effect of increasing the number of users connected to the shared caches in a delivery network with multiple caches under uniform popularity demand. This evaluation is performed considering $m = 100$ content files with 0.80 similarities among the library content in a network with $Z = 5$ shared caches of size $M = 20$. We have investigated the delivery rate of the system for the worst-case demand, which assumes a different content is requested by the users and results in the worst-case delivery rate of the system.

For this evaluation, we have considered the proposed scheme compared to other studies, including local caching with unicast delivery, coded caching strategy with uncoded placement [5], and shared cache solution with coded placement [13]. Local caching with unicast delivery considers the conventional local caching solution, in which M/m of each content is stored in all caches, followed by a unicast delivery to transmit the remaining portions $(1 - M/m)$ of the requested content to each user. Coded caching with uncoded placement is considered based on [5], which takes advantage of a global caching gain of $(\frac{1}{1+ZM/m})$ beside the local caching gain for each set of requests. Shared caches with coded placement [13] assume a network with Z shared caches and N users and divide users into Z groups. This scheme places both the coded and uncoded pieces of the content into the caches according to the connectivity pattern of each cache, where the optimal parameters for the caching scheme are obtained by solving a linear program.

We can see that although [13] reaches a lower delivery rate by increasing the number of users in each SBS compared to the coded caching strategy with uncoded placement, our approach achieves a higher gain due to the careful extraction of content for the placement phase. Our scheme places less load on the system as the number of users increases because we are required only to transmit extra refinement segments at low rates after a certain point.

5.7. Summary

As the demand for high delivery data rates continues to rise, content caching has become an important technique for reducing the delivery rate and improving the quality of service in current delivery networks. This chapter studies a cache-aided network with correlated sources and multiple shared caches, where each cache is connected to a group of users. The proposed approach considers the CACS for content placement that divides library content into clusters with approximately the same maximum distance by considering the similarity among the sources and the maximum allowable distortion in the network. Then, the representatives are used as the side information for the placement phase according to the coded caching strategy with uncoded placement. The delivery phase considers transmitting the coded multicast messages and refinement segments to serve all demands. We have formulated the peak delivery rate for the worst-case demand of the system by joint consideration of the delivery for the refinement segments and the caching strategy. We have also addressed the optimum partitioning with the objective of minimizing the peak rate. Our simulation results show that the proposed scheme exhibits excellent performance in reducing the peak delivery rates compared to others. Further studies can consider the proposed caching scheme in case of non-uniform demand. Also, considering a coded strategy in such a network can be studied as future work.

Chapter 6

Content Delivery in a Network with Multiple Shared Caches and Correlated Content under Non-Uniform Popularity Demand

Designing an effective placement scheme is crucial to maximizing caching gains and reducing the peak delivery load of cache-aided delivery networks. So far, we have addressed this challenge in joint consideration with the delivery phase of a cache-aided network under uniform demand and observed that a symmetric placement strategy for the extracted side information significantly reduces the delivery rate in such networks.

This chapter extends our previous study in Chapter 5 to account for heterogeneous user preferences as well. In this chapter, we study a more general case with a non-uniform popularity demand, resulting in a more complicated caching design and analysis since some files will be more popular than others.

6.1. Introduction

Conventionally, the HPF caching strategy is used in cache-aided networks with non-uniform demands; This strategy utilizes the popularity parameter to place the most popular files in all caches [5]. A few years ago, the coded caching strategy emerged as a major breakthrough in different types of caching networks to increase global gain and combat the high delivery data rates. A typical analysis of caching networks with coded caching strategy under heterogeneous user preferences is to design a placement strategy under this setting and evaluate the system's performance accordingly [123][124]. In this regard, content placement using file grouping is a common method to reduce the complexity of the problem. The authors in [6] propose grouping files on the basis of their popularity to allocate different chunks of caches to different groups. Still, they consider the same identical placement for the files within each group. Several other studies [7][10][124] proposed partitioning files in groups for caching networks with independent library content, demonstrating that file grouping in different ways is an effective method to cope with non-uniform popularity demand. Motivated by the above, we propose a content placement strategy based on joint consideration of the popularity and the similarity of library content in this chapter and then evaluate the peak delivery rate in the proposed network.

In the remainder of this chapter, we first describe the system model and introduce a clustering scheme based on the popularity and similarity of library content for the placement phase. Next, we describe the caching and delivery strategy and analyze the delivery rate of the system.

6.2. System Model

Consider a cache-aided delivery network with a server, N total users, and Z SBSs over a shared error-free broadcast link. The network model of the proposed system is illustrated in Figure 6.1.

Each SBS $i \in \{1, \dots, Z\}$ is equipped with a shared cache Y_i, which is connected to U_i number of users, where $U_i \in \{U_1, \dots, U_Z\}$ and $\sum_{i=1}^{Z} U_i = N$, and $Z \leq N$. We assume each user is connected to only one SBS and can receive messages from its SBS and the server.

The server has access to a library containing m content files $\mathcal{F} = \{1, \dots, m\}$ with the same size. The correlated library content files are considered based on the same model discussed in Chapter 5, in which each file is represented by a vector of i.i.d binary symbols of length B bits, $f_{\mathrm{i}} \in \mathbb{F}_2^B$; thus, for a block length of B, $H(f_{\mathrm{i}}) = B$ bits and $H(f_1, \dots, f_m) \leq mB$ bits.

We also consider a popularity parameter in this chapter to investigate the network under non-uniform popularity demand. The popularity of each content file $i \in \mathcal{F}$ is denoted by p_{i}, where $\sum_{i=1}^{m} p_{\mathrm{i}} = 1$. Without loss of generality, we assume that the file popularity is decreasing in the index, i.e., $p_j \geq p_i$ where $j \geq i$.

We also define P_k as an aggregate popularity parameter for each cluster $k \in K_{\boldsymbol{C}}$, which represents the sum of the popularity of files per cluster k, i.e., $P_k = \sum_{i=1}^{m_k} p_i$, and $\sum_{k=1}^{Kc} P_k = 1$ where m_k indicates the number of files within cluster k and could be different across clusters. We use this parameter later in designing the objective function.

We aim to optimize the content placement of the proposed network under non-uniform popularity demand distribution to reduce the delivery rate during peak hours. To address the placement challenge, we propose a *popularity-based correlation-aware clustering scheme* (PB-CACS) that considers both the popularity and similarity of library content, extracting two types of side information for the entire library known as;

- *Popular side information* (PSI)
- *Clusters' side information* (CSI)

Then, we consider a hybrid placement strategy in which PSIs are fully stored in all caches, while CSIs are divided into chunks and stored in different caches based on the coded caching solution.

Next, coded multicast messages and refinement segments are transmitted in the delivery phase to construct the requested CSIs and clustered files.

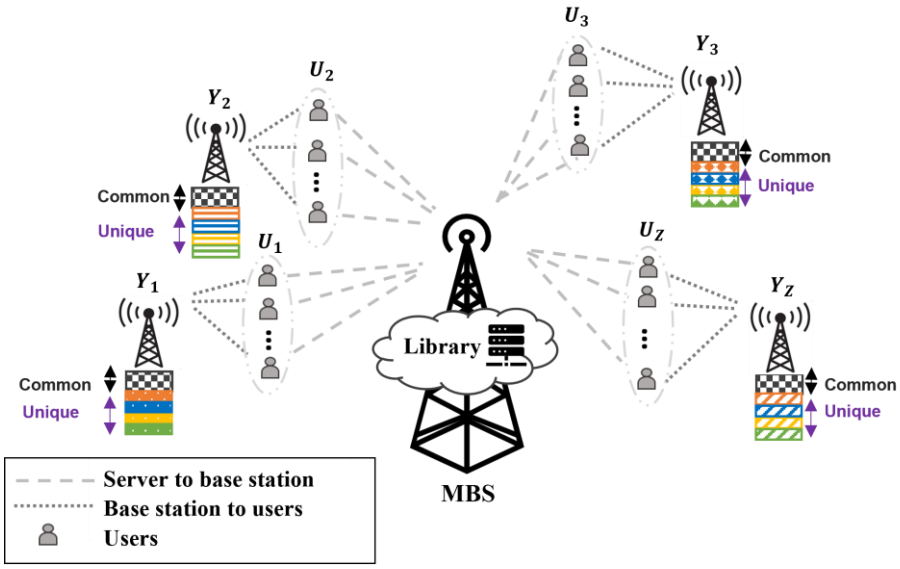


Figure 6.1: Content delivery network in a shared cache Framework with multiple caches under non-uniform popularity demand considering a hybrid placement strategy

6.3. Popularity-Based Correlation-Aware Clustering Scheme

The proposed PB-CACS identifies the side information for the entire library based on joint considerations of the similarity and popularity of library content files during the placement phase. The proposed clustering scheme categorizes the library content into a set of highly popular files along with a sufficient number of compact and well-separated clusters with approximately the *same aggregate popularity* and maximum distance per cluster.

It is worth mentioning that we are interested in reaching clusters with the same aggregate popularity to be able to formulate a non-uniform popularity case into a uniform popularity case. Therefore, we can easily maintain symmetry among representatives during the placement phase and create coded multicast messages efficiently.

6.3.1. PB-CACS Objective Function and Methodology

Similar to the proposed CACS in Chapter 4, the PB-CACS is formulated as an AI-based optimization problem with objective function $\boldsymbol{f}$ that needs to be minimized iteratively over the set

of all feasible clustering solutions defined by $\psi = \{C^1, C^2, \dots, C^{S(m,K)}\}$, where $S(m, K)$ is the set of all feasible clustering solutions defined by Eq. (3.3).

In addition to the similarity in the PB-CACS, a popularity parameter and a popularity-related constraint are also considered in the objective function of this scheme. In this regard, we define $\bar{P}_C$, as the average aggregate popularity over all clusters in the clustering solution C, given by

$$\bar{P}_C = \frac{1}{K_C} \sum_{k=1}^{K_C} P_k \tag{6.1}$$

where $P_k = \sum_{i=1}^{m_k} p_i$.

We also define P_C^{max} and P_C^{min} as the maximum aggregate popularity and minimum aggregate popularity parameters of all K_C clusters in the clustering solution C.

$$P_C^{max} = \max_{\forall k \in K_C} \{P_k\} \tag{6.2}$$

$$P_C^{min} = \min_{\forall k \in K_C} \{P_k\} \tag{6.3}$$

Moreover, for each clustering solution C, we define the deviation of aggregate popularity ΔP_C, given by:

$$\Delta P_C = P_C^{max} - P_C^{min} \tag{6.4}$$

Then, we define $J(C)$ as a function that reflects the compactness of the clusters, given by:

$$\underset{C \in \psi}{J(C)} = K_C \,.\, \bar{d}_C^{max} . \Delta_C \,.\, \Delta P_C \tag{6.5}$$

where K_C is the number of clusters obtained by the clustering solution C, and $\bar{d}_C^{max}$ and Δ_C are defined as described in chapter 3 by Eqs. (3.5) and (3.6).

Finally, considering $E(C)$ described in Eq. (4.4) as a function that reflects the separation of the clusters in the clustering solution C, we introduce the objective function as follows

$$\begin{aligned} \boldsymbol{f}(C) &= \underset{C \in \psi}{\text{Min}}\left(\frac{J(C)}{E(C)}\right) \\ s.t. \; & \delta_C^{max} \leq \boldsymbol{\delta} \\ & P_C^{max} \leq \bar{P}_C + P_C^{min} \end{aligned} \tag{6.6}$$

We iteratively minimize $\boldsymbol{f}(C)$ subject to two constraints; one is with respect to the given distortion of the system, and the other is to ensure reaching clusters with the same aggregate popularity.

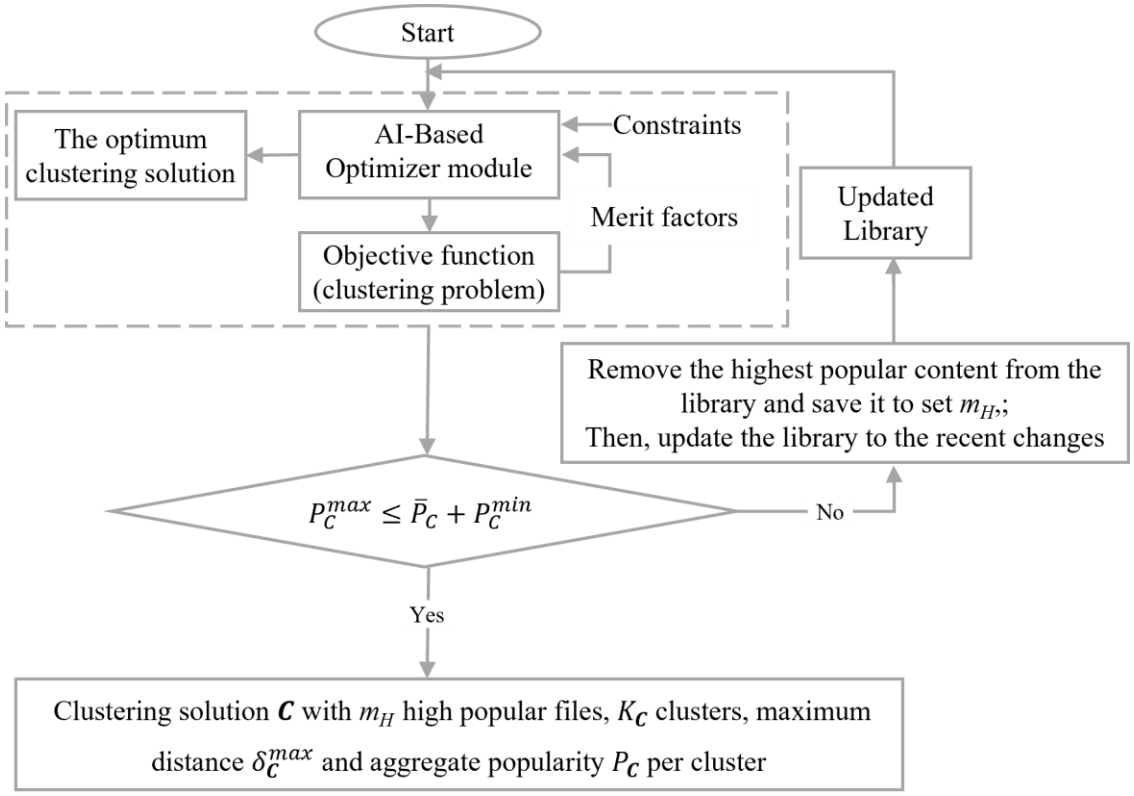
However, the PB-CACS goes through an extra step to meet the aggregate popularity constraint. The PB-CACS consists of three main steps.

1. **Cluster Initialization step:** Initially, each content file is equipped with a cluster number in order to form a primary clustering solution; then, the primary solution will be repeatedly re-clustered and updated according to the representatives until no change is seen in the clusters.
2. **Merging and Modifying step:** Following the achievement of the clustering solution in step 1, the achieved clusters will be re-clustered, merged, and modified based on a designed condition, Eq. (3.4), to compensate for the distortion deviation between clusters of different sizes and improve the result. This condition leads clusters to gradually achieve the same maximum distance without increasing the number of clusters. The AI-optimizer module then optimizes the achieved clusters over the course of iterations based on the problem objectives.
3. **Aggregate Uniformity step:** The final step involves optimizing clusters based on a uniformity condition to reach the same aggregate popularity per cluster $c_i \in C$, according to the aggregate popularity constraint for all clusters defined as:

$$P_C^{max} \leq \bar{P}_C + P_C^{min} \tag{6.7}$$

As long as the uniformity condition is not met, the highest popular content file in the library will be removed from the library and stored in the set m_H; then, the library will be updated based on recent changes, and the clustering scheme will proceed according to the updated library. As a result of performing all the above steps, the clustering solution $\boldsymbol{C}$ is obtained, consisting of m_H number of highly popular content and $K_{\boldsymbol{C}}$ representatives, while achieved clusters have approximately the same maximum distance $\delta_{\boldsymbol{C}}^{max}$ and aggregate popularity $P_{\boldsymbol{C}}$.

Flowchart 6.1 explains the above steps and illustrates how the optimizer module and objective function collaborated to solve the clustering problem in the proposed PB-CACS, considering the given constraints of the system.



Flowchart 6.1: How the optimizer module and objective function collaborated to solve the clustering problem in the proposed PB-CACS considering system constraints

6.4. Proposed Hybrid Caching and Delivery Strategy

The proposed network operates in two phases: The cache placement and the delivery phase.

6.4.1. Hybrid Cache Placement Strategy

Let $\boldsymbol{C}$ represents the achieved clustering solution by the PB-CACS during the placement phase. Then, library files fall into one of the following groups based on the achieved $\boldsymbol{C}$:

- m_H content files with very high popularity, known as *popular side information (PSIs)*
- $K_{\boldsymbol{C}}$ cluster representatives, known as *clusters' side information (CSIs)*
- $m_C = (m - m_H - K_{\boldsymbol{C}})$ files, known as *Clustered files*

We consider a hybrid strategy for the placement phase, in which PSIs are fully stored in all shared caches. At the same time, distinct portions of CSIs are placed in different caches based on the coded caching strategy with uncoded placement, which is introduced in section 5.3.1 of Chapter 5.

In contrast with the coded caching strategy with non-uniform demands [6], our model stores only PSIs instead of all files in the same group. Still, the rest of the library would be accessible at a low rate to the requesting users by sending refinement segments in our model. The reason is that delivery of the clustered files has been formulated as distributed source coding with side information at the decoder.

It is worth recalling that we have partitioned the library files into clusters with the same aggregate popularity in our scheme. Therefore, different clusters do not have any priority over each other; Hence, we allocate the same memory size to each CSI and preserve the symmetry across different groups.

Recall that the parameter T defined as integer values $T \in [0, Z]$. Then, the available memory size to store CSIs per shared cache is defined as $M_R \in K_C T/Z$, i.e., $M_R \in \{0, K_C/Z\ , 2K_C/Z\ , \dots , K_C\}$, where K_C is the number of clusters achieved by the clustering scheme and Z is the number of shared caches in the network.

The process of dividing CSIs into chunks and placing chunks in different caches is similar to what we have discussed in section 5.3.1 of Chapter 5. In this regard, Eqs (5.1) and (5.2) indicate how CSIs could be divided into chunks, labeled, and then placed in different caches.

This solution requires a maximum memory size of $M = m_H + M_R$ files per shared cache.

6.4.2. Delivery Phase

During the delivery phase, users reveal their demands to the SBSs. Demands corresponding to the PSIs will be locally served as these files are fully stored in all SBSs, while the rest of the demands should be processed by the server.

Let Q_z represents all the distinct requests of cache Y_z, $\forall z \in Z$, to be processed by the server. Demands received by the server are either associated with representatives or clustered files. In

both cases, the relevant representative should also be constructed. Therefore, the server maps all requested files in a demand vector Q_z, to the relevant distinct representatives. We assume the mapped vector is sorted ascending by the index of the requested representatives and denoted by $y_z \subseteq \{K_1, \dots, K_C\}$, for each demand vector Q_z , $\forall z \in Z$.

Similar to what we have discussed in section 5.3.2 of Chapter 5, the rest of the demands will be satisfied in one of the two following ways:

1. ***Demands corresponding to representatives (CSIs)*:**
   - When $M_R = K_C$: These demands are locally served since such content files have been fully cached and are accessible to all users.
   - When $M_R < K_C$: Such demands are served by receiving coded multicast messages in accordance with the cached content. The server sends the multicast coded messages to $T + 1$ caches for each subset of the requested chunks according to Eq (5.3).

In recent years, a few efficient solutions have been proposed to optimize the caching strategy by focusing on highly popular demands and the coded multicast messages in caching networks with non-uniform demands. However, it is still a challenge to serve files from the less popular groups at a low rate; Such requests could cause an unexpected spike in the system load if they are not considered in the placement phase of the caching strategy. A key part of our problem formulation involves targeting this group to satisfy such demands at a low rate as follows:

2. ***Demands corresponding to the clustered files:***

Similar to the uniform case, all unique requested clustered files across all SBSs are assigned to a demand vector Q' in the server. Such demands will be served by transmitting a refinement segment in addition to the coded multicast messages required for constructing the CSIs. In this regard, the server transmits encoded messages X_i of size $H\left(F_{k,i}^{j_z} \middle| \hat{F}_k\right) \leq H\left(F_{k,i}\right)$ to the requesting user j as a refinement segment so that the user, where $H(.|.)$ describes the conditional entropy, and $\hat{F}_k$ is the corresponding representative. As a result, users can reconstruct the clustered file $F_{k,i}$ by jointly decoding the received message and the available side information in the cache.

The refinement segments needed for the same file in different SBSs are transmitted just once due to the broadcast nature of the medium.

6.5. Delivery Rate Analysis

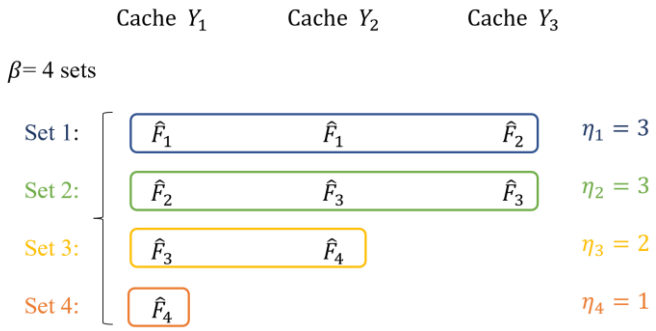
Let β denote the maximum number of requested representatives in a mapped vector y_z among all Z caches. We assume the server creates β sets of demand and then serves the demands from each set together; In order to create each set, the first element with the lowest index is picked from each y_z vector. Since the number of requested representatives is not necessarily the same in all y_z, the number of elements in each set can be different, reflecting the number of caches with a request in this set. Then, we denote the number of caches with a request from the server in each set by η. Example 6.1 illustrates this step in a simple way.

Example 6.1: Assume the library content is categorized into $K = 4$ clusters with representatives $\boldsymbol{F} = \{\hat{F}_1, \dots, \hat{F}_4\}$ by the PB-CACS scheme in the proposed network with $Z = 3$ shared caches. Then, the mapped vector y_z for each SBS is as follows

$y_1 = \{\hat{F}_1, \hat{F}_2, \hat{F}_3, \hat{F}_4\} \rightarrow y_1$ has the maximum number of requested representatives = 4

$y_2 = \{\hat{F}_1, \hat{F}_3, \hat{F}_4\}$

$y_3 = \{\hat{F}_2, \hat{F}_3, \hat{F}_4\}$



Therefore, $\beta = 4$ sets and $\eta_1 = 3, \eta_2 = 3, \eta_3 = 2$, and $\eta_4 = 1$

According to Eq. (5.4), the total delivery rate of this proposed system also includes two components; R_{CM} which is needed for constructing CSIs and the R_{RS} which is required for the refinement segments; hence $R = R_{CM} + R_{RS}$.

In this regard, the delivery rate for transmitting the refinement segments is calculated based on Eq. (5.7), while the delivery rate for the coded multicast messages is calculated as follows.

Consider the case of reaching the clustering solution $\boldsymbol{C}^S$ with $K_{C}s$ clusters during the placement phase. The coded multicast delivery rate in a cache-aided network with Z shared caches, each having a demand vector Q_z, $\forall z \in Z$, and $M_R = K_{\boldsymbol{C}}{}^s T/Z$ cached files is given by

$$R_{CM}(M_R) = \sum_{i=1}^{\beta} \text{Min}\left(\frac{\binom{Z}{T+1}-\binom{Z-\eta_i}{T+1}}{\binom{Z}{T}}, \eta_i\left(1-\frac{M_R}{K_{C}S}\right)\right) \tag{6.8}$$

where $T = M_R\, Z/K_{\boldsymbol{C}}s$, $T \in [0:Z]$, β describes the maximum number of requested representatives (CSIs) across all mapped vectors, and η describes the number of caches with a request from the server in each set of i.

The coded multicast delivery rate R_{CM} can be achieved by treating each set $i \in \beta$ representative demands independently and then applying the coded delivery scheme proposed in [5, Theorem 1] for each set of β demands.

In this case, for each set of β, the server sends the XOR of the $T+1$ requested segments to $T+1$ caches, as each cache has stored T/Z of all segments of each representative. Thus, each SBS can reconstruct one requested representative after $\binom{Z}{T+1}$ coded multicast transmissions, where the size of the coded messages is $1/\binom{Z}{T}$.

Accordingly, $\binom{Z}{T+1}$ transmissions are needed to serve all caches with one set of demands. However, there might be some cases where all demands of an SBS be associated with only one cluster or belong to only PSIs. In such cases, the demand vector Q_z and consequently, the mapped vector y_z can be empty or have fewer requests than the rest of the caches; hence, for some sets of β, they have no demand for the server to meet. In this regard, we assume η_i as the maximum number of caches that contributes to the multicast transmission in set $i \in \beta$; thus, the number of unnecessary transmissions is $\binom{Z-\eta_i}{T+1}$. As a result, the number of coded transmissions is $\binom{Z}{T+1} - \binom{Z-\eta_i}{T+1}$ for each set of β requested representatives, where the size of each coded message is $1/\binom{Z}{T}$.

The second term in the min (.) function is derived from the unicast rate and considered for the cases when multicasting does not improve the rate; therefore, the minimum of the function is transmitted.

If demands involve all clusters in all SBSs, then all CSIs must be constructed in all SBSs; that is the worst-case demand for the multicast rate in our model and leads to the coded multicast peak rate. In such a case $\beta = K_{\mathcal{C}^S}$ and $\eta = Z$ in all sets; thus, the upper bound (normalized by the file size B) will be reached as follows: $T = M_R\, Z/K_{\mathcal{C}^S}$

$$\begin{aligned} R_{CM}^{P} &= K_{\mathcal{C}^S} \frac{\binom{Z}{T+1}}{\binom{Z}{T}} \\ &= K_{\mathcal{C}^S} \frac{Z-T}{T+1} \\ &= K_{\mathcal{C}^S} \frac{Z(1-M_R/K_{\mathcal{C}^S})}{1+\mathrm{Z}M_R/K_{\mathcal{C}^S}} \end{aligned} \tag{6.9}$$

Therefore, the peak delivery rate of the system is given by:

$$R^{P}(M_R) = K_{\mathcal{C}^S} \frac{Z(1-M_R/K_{\mathcal{C}^S})}{1+\mathrm{Z}M_R/K_{\mathcal{C}^S}} + \sum_{k=1}^{K_{\mathcal{C}^S}} \sum_{i=1}^{Q_k'} H\left(F_{k,i} \middle| \hat{F}_k\right) \tag{6.10}$$

Where $Q_k' \subseteq m_k$ is the number of unique requested clustered files of cluster $k \in K_{\mathcal{C}^S}$ in all SBSs.

6.6. Results and Discussion

The performance of the proposed clustering solution and the caching strategy is evaluated in this section. In line with existing studies, we consider the Zipf popularity distribution q, with parameter ξ to describe the non-uniform popularity demand given by Eq. (2.2).

The evaluations are performed by considering the BBA algorithm in the AI-optimizer module with 100 bat populations over 200 iterations. The parameter setting for the BBA solution is described in Table 3.2.

Figure 6.2 illustrates the trade-off between the achieved number of clusters, the maximum distance within each cluster, and the similarity among the library content in the proposed PB-CACS. As can be seen, the achieved number of clusters reduces by increasing the similarity among sources for a fixed maximum distance value. On the other hand, if the system specification allows for more distance within each cluster, the number of clusters can be reduced.

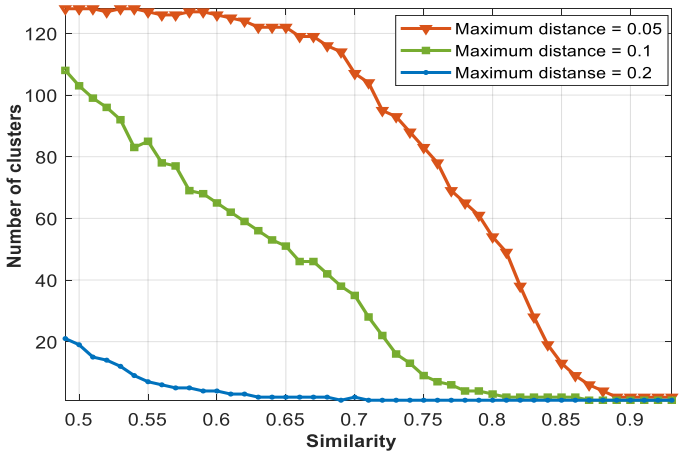


Figure 6.2: The trade-off between the achieved number of clusters and the similarity among content files with $m = 130$ in the proposed clustering solution

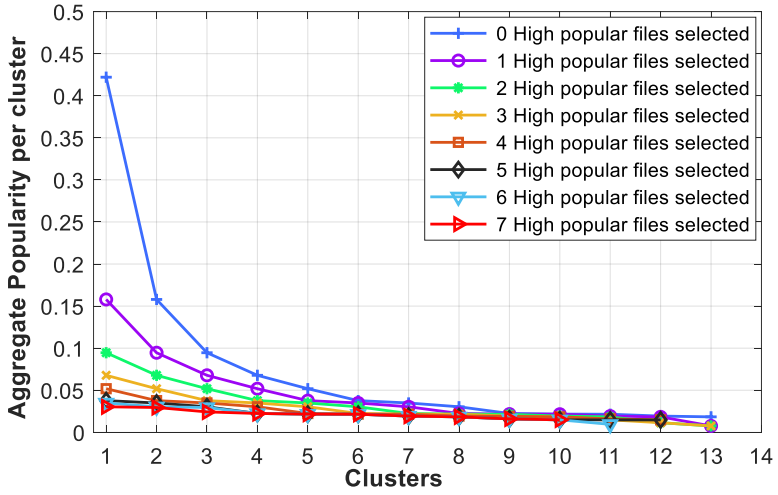


Figure 6.3: Uniformity of the aggregate popularity in the process of clustering by selecting $m_H = 7$ high popular files and $K_C = 10$ CSIs in the PB-CACS considering Zipf parameter $\xi = 1.4$

Figure 6.3 exhibits the process of partitioning library content into a set of highly popular side information and several clusters with approximately the same aggregate popularity in each cluster. As can be seen, desired clustering solution has been achieved by selecting $m_H = 7$ highly popular files and partitioning the rest of the library into 10 clusters. In that case, the achieved clusters reach approximately the same aggregate popularity shown by the red line.

Figure 6.4 illustrates the performance of the proposed scheme compared to other studies for a random demand request, including the famous HPF solution with an uncoded prefetching strategy in addition to the state-of-the-art coded-uncoded placement strategy for shared caches with independent sources [11]. We have assumed the same number of highly popular files fully placed in all shared caches in both our solution and the coded-uncoded strategy to have a fair comparison.

We can see that all three solutions reduce at almost the same rate for the smaller memory size, while our solution and the coded-uncoded strategy enjoy a higher reduction by increasing the memory size. Although the coded-uncoded strategy exhibits a great performance, our proposed strategy yields a higher gain after a certain point due to the careful extraction of content as side information considering the similarity and popularity of sources for the placement phase. Additionally, our problem formulation allows us to serve less popular requests at a low rate, preventing unexpected spikes for such requests in the system load. In fact, with this comparison, we show how leveraging similarities among sources, besides the popularity of demands, results in significant reductions in the delivery rates in such networks.

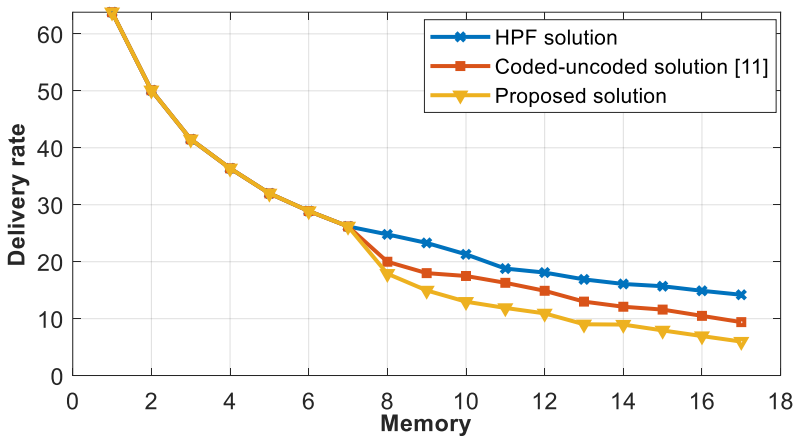


Figure 6.4: Delivery rate comparison in a network with $m=100$ content files and $Z=10$ shared cache each serving U_i=10 users

6.7. Summary

This chapter proposes a content delivery network in a shared cache framework with correlated content under non-uniform demand popularity distribution, where each cache is responsible for

serving a group of users. Our goal is to optimize the content placement in order to reduce the peak delivery rate of the system. To this end, we formulate the caching problem as a source coding with side information at the decoder scenario. To address the content placement challenge, we propose a popularity-based correlation-aware clustering scheme to extract the most efficient side information for the entire library with the joint consideration of the popularity and similarity among library files considering a given distortion in the system. Then, a hybrid placement strategy is used, which consists of storing highly popular content in all caches and caching the cluster representatives according to the coded caching strategy. The delivery phase includes transmitting coded multicast messages and refinement segments to construct the representatives and clustered files, respectively. We have studied the peak delivery rate consisting of the rate for refinement segments and the coded multicast. Further studies can be conducted on the proposed network under heterogeneous user preferences in each SBS. Furthermore, adopting coded prefetching can be considered a future work in networks with correlated sources and shared caches.

Chapter 7

Conclusions and Future Research Directions

The exponential growth of content-related data traffic has posed a serious challenge for current delivery networks, requiring them to come up with innovative solutions beyond traditional CDNs to combat this challenge. Therefore, investigating novel techniques besides infrastructural development is crucial to be able to reduce the delivery rate while providing high-quality services for end users. Content caching has proved to be an efficient technique in reducing the delivery rate by storing some content close to the end users.

This research considers a shared cache framework architecture, allowing users in SBSs to access the deployed cache in the cell. As such, we are able to efficiently overcome the imbalanced network load and high data rate challenge while improving user experience. This setting is highly

beneficial in next-generation wireless networks, HetNet environment, or as an upper layer of hierarchical caching networks in IoT-based applications.

7.1. Conclusions

In this dissertation, we have examined the use of shared caches for users within an SBS in a network with correlated content under different settings. We have addressed the content placement and delivery phase in such settings and calculated the rate memory trade-off, the peak rate, and the expected delivery rates of the system.

This dissertation concludes with the following remarks:

- Caching networks are directly affected by cached content; therefore, designing an efficient placement solution along with a practical delivery strategy is crucial to developing efficient cache-aided content delivery networks. Meanwhile, exploiting the similarity among library content plays a crucial role in reaching a more efficient content placement and delivery strategy in cache-aided networks with correlated content and shared caches.
- We have formulated the caching problem as a source coding with side information at the decoder; therefore, the first challenge is identifying the most efficient side information for the entire library. Our solution to this challenge is to develop two clustering schemes based on AI-based optimization techniques to categorize library content into clusters with the same maximum distance according to the similarity of the content and the popularity of demands, considering the distortion constraint of the system. The representative of the clusters is then selected as the side information for the placement phase.
- In Chapter 3, we comprehensively examined the proposed clustering in a general framework and investigated its performance considering a wide range of datasets and different algorithms in the AI-optimizer module. Our model has demonstrated excellent performance against comparative studies based on extensive simulation results and statistical analysis. This clustering framework is essential to our research as it facilitated the development of two new clustering schemes, the CACS and the PB-CACS, as part

of the placement phase of the proposed caching networks, taking into account both similarity and popularity of library sources.

- We have discussed that achieving clusters with approximately the same maximum distance in both proposed schemes (CACS and PB-CACS) is useful in introducing the optimum allocation of the delivery rate and maximum allowable distortion to the files.
- We have observed a trade-off between the achieved number of clusters, the maximum distance within each cluster, and the similarity among content in the CACS solution; We have shown that the maximum distance within the clusters can be reduced by increasing the number of clusters in the clustering solution. At the same time, increasing the correlation among the library content results in clustering solutions with fewer groups and lower maximum distance within the clusters.
- We have also stated that we are interested in clusters with the same aggregate popularity in the PB-CACS; The reason is that we have found that in this way, we can easily maintain symmetry among representatives during the placement phase and create coded multicast messages efficiently.

▪ In Chapter 4, we discussed a delivery network with a single shared cache and correlated content under lossy caching.

- We have shown that the efficiency of the current shared cache networks with independent sources does not carry over to the same caching network with correlated content, which implies why an efficient solution is still required in such systems.
- We exploited the correlation among library content in the placement phase by utilizing the proposed CACS. We have also optimized the expected delivery rate and the users' expected distortion in the delivery phase by joint consideration of the rate-distortion function and our proposed caching strategy, which resulted in a significant reduction in the delivery rate.
- A key challenge in optimizing the expected delivery rate in our model is the limit for the maximum allowable distortion in the system. We have addressed it based

on the Lagrange multipliers technique and reverse water-filling optimization algorithm. In this regard, we found that the optimal allocation of delivery rate is achieved by considering equal distortion for all files, which validates our intention to create clusters with approximately the same maximum distance in the first place.

- We have discussed the performance of the proposed scheme, and we have demonstrated that our proposed scheme requires much lower delivery rates for the same memory size to accomplish the delivery phase. Also, we have illustrated that the proposed scheme has yielded a significant reduction in the expected distortion for different delivery rates compared to the comparative study.

- In Chapter 5, we discussed a delivery network with multiple shared caches and correlated content under uniform popularity demand.
  - First, we extracted the side information for this network according to the CACS. We then carried out the placement phase based on the coded caching strategy to take advantage of the multicast opportunities and global caching gain. Yet contrary to the CC scheme, we only store representatives in our model, while the rest of the library is accessible at a low rate by sending refinement segments to the requesting users in this model. The delivery phase involves constructing the clusters' representatives and clustered files by transmitting coded multicast messages and refinement segments, respectively.
  - By formulating the problem based on the DSC solution and ensuring that content is carefully extracted for the placement, we have demonstrated a higher gain in this system. Our study has examined the impact of increasing the number of users connected to each cache. We have observed that our scheme puts less load on the system as users increase because we need to transmit only extra refinement segments after a certain point. While with a pure coded caching strategy, we must transmit more coded multicast messages as users increase, which adversely affects delivery rates.

- Our analysis on increasing the number of users also implies why a shared cache framework should be designed differently than a network with an individual cache for each user. In a network with individual caches, increasing users contribute to the global memory size and, therefore, the global gain. However, a shared cache framework requires a different strategy since increasing the number of users connected to each cache only increases delivery rates as more demands are placed on the server.
- We have also discussed the optimum library partitioning formulated to minimize the worst-case delivery rate in the system. We have demonstrated that the worst-case demand is significantly impacted by the coded multicast messages in our model; hence, a balance should be maintained between the number of achieved representatives and the global cache size in the network during the placement phase in order to decrease the number of multicast messages in the delivery phase. We have illustrated how the delivery rate decreases to a certain point by increasing memory size but rises again once it reaches its minimum.

▪ In Chapter 6, we discussed a delivery network with multiple shared caches and correlated content under non-uniform popularity demand.

- In this system, we extract two sets of side information upon performing the PB-CACS solution in the placement phase; the popular side information and the clusters' side information. As such, we introduce a hybrid placement strategy in which the popular side information is stored completely in all caches, and the clusters' side information is split among different caches. In the delivery phase, coded multicast messages and refinement segments are transmitted to construct the requested cluster representatives and clustered files.
- We have shown in this network that our proposed strategy yields a higher gain after a certain point due to the careful problem formulation and caching strategy considering both the popularity and similarity of library content in the placement phase.
- Additionally, our problem formulation allows us to serve less popular requests at a low rate, preventing unexpected spikes for such requests in the system load.

We have illustrated how leveraging similarities among content, in addition to the popularity of demands, results in significant reductions in the delivery rates in such networks.

7.2. Future Directions

This research has been conducted on formulating the caching network as a DSC problem and optimizing the content placement of a shared cache framework to maintain a balanced load while improving the quality of experience for users. There are promising research directions arising from this framework, which can be investigated further as outlined below:

- *Investigating the studied framework under non-ideal channel conditions:* It is possible to extend the proposed framework with shared caches to networks with non-ideal conditions where the broadcast link is not error-free. The challenge with noisy channels is not only determining what should be cached during the placement phase or what should be transmitted in the delivery phase but also addressing how to transmit the required content is very important. In such a scenario, a joint-source channel coding technique can be used as an efficient solution to reduce the delivery rate and the expected distortion in receivers.
- *Investigating the studied framework under heterogeneous cache size:* another line of research can focus on having shared caches of different sizes in the network, posing exciting challenges to the coding design. In a conventional coded caching scheme with homogenous cache sizes, the placement procedure results in caching segments of equal sizes. Therefore, a maximal clique of different segments is formed that can be utilized to create efficient coding opportunities in the delivery phase. However, a network with heterogeneous cache sizes would prevent us from maintaining symmetry and diversity as described in the conventional coded caching strategy since the size of the transmitted segment would also be heterogeneous. One possible solution is to perform zero-padding so that all segments can be aligned for coding. Grouping caches with approximately the same size could also be considered a potential solution to this challenge which should be investigated considering the system model and constraints.

- *Investigating a case for a network with a dense deployment of SBS:* Having a dense deployment of SBS increases the possibility of some users accessing more than one cache simultaneously. In such a scenario, the placement strategy should be designed so that such users can take advantage of diverse cache content simultaneously to increase the global caching gain of the network.

7.3. Publications

The following is the list of publications throughout this research.

[126] B. Merikhi, M. Soleymani, "Cache-Aided Networks with Shared Caches and Correlated Content under Non-Uniform Demands," 2023 20th *IEEE Consumer Communications & Networking Conference (CCNC)*, Jan. 8-11, 2023.

[127] B. Merikhi, M. Soleymani, "Cache-Aided Delivery Network in a Shared Cache Framework with Correlated Sources," in *Proceedings of the 18th ACM International Symposium on QoS and Security for Wireless and Mobile Networks (ACMQ2SWinet)*, New York, NY, USA, Oct. 2022, pp. 121–129.

[128] B. Merikhi and M. R. Soleymani, "Automatic Data Clustering Framework Using Nature-Inspired Binary Optimization Algorithms," *IEEE Access*, vol. 9, pp. 93703–93722, 2021.

[129] B. Merikhi, S. M. Mirjalili, M. Zoghi, and S. Mirjalili, "Radiation pattern design of photonic crystal LED optimized by using multi-objective grey wolf optimizer," *Photonic Network Communications*, vol. 38, no. 1, pp. 167–176, Aug. 2019.

References

[1] "Cisco Annual Internet Report - Cisco Annual Internet Report (2018–2023) White Paper," *Cisco*. https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html (accessed Nov. 04, 2021).

[2] "Global IP Networks Focus on Sharpening Their Edge," *Cisco Blogs*, Dec. 04, 2018. https://blogs.cisco.com/sp/global-ip-networks-focus-on-sharpening-their-edge (accessed Nov. 04, 2021).

[3] E. Ghabashneh and S. Rao, "Exploring the interplay between CDN caching and video streaming performance," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, Jul. 2020, pp. 516–525. doi: 10.1109/INFOCOM41043.2020.9155338.

[4] G. S. Paschos, G. Iosifidis, M. Tao, D. Towsley, and G. Caire, "The Role of Caching in Future Communication Systems and Networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1111–1125, Jun. 2018, doi: 10.1109/JSAC.2018.2844939.

[5] M. A. Maddah-Ali and U. Niesen, "Fundamental Limits of Caching," *IEEE Transactions on Information Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014, doi: 10.1109/TIT.2014.2306938.

[6] U. Niesen and M. A. Maddah-Ali, "Coded Caching With Nonuniform Demands," *IEEE Transactions on Information Theory*, vol. 63, no. 2, pp. 1146–1158, Feb. 2017, doi: 10.1109/TIT.2016.2639522.

[7] M. Ji, A. M. Tulino, J. Llorca, and G. Caire, "Order-Optimal Rate of Caching and Coded Multicasting With Random Demands," *IEEE Transactions on Information Theory*, vol. 63, no. 6, pp. 3923–3949, Jun. 2017, doi: 10.1109/TIT.2017.2695611.

[8] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "The Exact Rate-Memory Trade-off for Caching With Uncoded Prefetching," *IEEE Transactions on Information Theory*, vol. 64, no. 2, pp. 1281–1296, Feb. 2018, doi: 10.1109/TIT.2017.2785237.

[9] M. A. Maddah-Ali and U. Niesen, "Decentralized Coded Caching Attains Order-Optimal Memory-Rate Tradeoff," *IEEE/ACM Transactions on Networking*, vol. 23, no. 4, pp. 1029–1040, Aug. 2015, doi: 10.1109/TNET.2014.2317316.

[10] J. Zhang, X. Lin, and X. Wang, "Coded Caching Under Arbitrary Popularity Distributions," IEEE Transactions on Information Theory, vol. 64, no. 1, pp. 349–366, Jan. 2018, doi: 10.1109/TIT.2017.2768517.

[11] A. G. Sheshjavani, A. Khonsari, S. P. Shariatpanahi, M. Moradian, and A. Dadlani, "Coded Caching Under Non-Uniform Content Popularity Distributions with Multiple Requests," in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, May 2020, pp. 1–6. doi: 10.1109/WCNC45663.2020.9120820.

[12] A. Ghaffari Sheshjavani, A. Khonsari, S. P. Shariatpanahi, and M. Moradian, "Content caching for shared medium networks under heterogeneous users' behaviors," *Computer Networks*, vol. 199, p. 108454, Nov. 2021, doi: 10.1016/j.comnet.2021.108454.

[13] A. M. Ibrahim, A. A. Zewail, and A. Yener, "Coded Placement for Systems with Shared Caches," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, May 2019, pp. 1–6. doi: 10.1109/ICC.2019.8761409.

[14] E. Parrinello, A. Ünsal, and P. Elia, "Coded Caching with Shared Caches: Fundamental Limits with Uncoded Prefetching." arXiv, Oct. 16, 2018. doi: 10.48550/arXiv.1809.09422.

[15] J. Hachem, N. Karamchandani, and S. Diggavi, "Content caching and delivery over heterogeneous wireless networks," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, Apr. 2015, pp. 756–764. doi: 10.1109/INFOCOM.2015.7218445.

[16] R. W. L. Coutinho and A. Boukerche, "Modeling and Analysis of a Shared Edge Caching System for Connected Cars and Industrial IoT-Based Applications," in *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 2003-2012, March 2020, doi: 10.1109/TII.2019.2938529.

[17] P. Hassanzadeh, A. Tulino, J. Llorca, and E. Erkip, "Cache-aided coded multicast for correlated sources," in *2016 9th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, Sep. 2016, pp. 360–364. doi: 10.1109/ISTC.2016.7593137.

[18] Hassanzadeh, A. M. Tulino, J. Llorca, and E. Erkip, "Rate-Memory Trade-Off for Caching and Delivery of Correlated Sources," *IEEE Transactions on Information Theory*, vol. 66, no. 4, pp. 2219–2251, Apr. 2020, doi: 10.1109/TIT.2020.2969918.

[19] K. Wan, D. Tuninetti, M. Ji, and G. Caire, "On Coded Caching with Correlated Files," in *2019 IEEE International Symposium on Information Theory (ISIT)*, Jul. 2019, pp. 692–696. doi: 10.1109/ISIT.2019.8849314.

[20] T. T. Nu, T. Fujihashi, and T. Watanabe, "Content-aware Efficient Video Uploading for Crowdsourced Multi-view Video Streaming," in *2018 International Conference on Computing, Networking and Communications (ICNC)*, Mar. 2018, pp. 98–104. doi: 10.1109/ICCNC.2018.8390288.

[21] S. Borst, V. Gupta, and A. Walid, "Distributed Caching Algorithms for Content Distribution Networks," in *2010 Proceedings IEEE INFOCOM*, Mar. 2010, pp. 1–9. doi: 10.1109/INFCOM.2010.5461964.

[22] Z. Bar-Yossef, Y. Birk, T. S. Jayram, and T. Kol, "Index Coding With Side Information," *IEEE Transactions on Information Theory*, vol. 57, no. 3, pp. 1479–1494, Mar. 2011, doi: 10.1109/TIT.2010.2103753.

[23] K. C. Almeroth and M. H. Ammar, "The use of multicast delivery to provide a scalable and interactive video-on-demand service," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 6, pp. 1110–1122, Aug. 1996, doi: 10.1109/49.508282.

[24] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman, "Placement Algorithms for Hierarchical Cooperative Caching," *Journal of Algorithms*, vol. 38, no. 1, pp. 260–302, Jan. 2001, doi: 10.1006/jagm.2000.1129.

[25] I. Baev, R. Rajaraman, and C. Swamy, "Approximation Algorithms for Data Placement Problems," *SIAM J. Comput.*, vol. 38, no. 4, pp. 1411–1429, Aug. 2008, doi: 10.1137/080715421.

[26] M. M. Amiri, Q. Yang, and D. Gündüz, "Coded caching for a large number of users," in *2016 IEEE Information Theory Workshop (ITW)*, Sep. 2016, pp. 171–175. doi: 10.1109/ITW.2016.7606818.

[27] C. Zhang and B. Peleato, "On the Average Rate for Coded Caching with Heterogeneous User Profiles," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, Jun. 2020, pp. 1–6. doi: 10.1109/ICC40277.2020.9148779.

[28] S. Wang and B. Peleato, "Coded Caching with Heterogeneous User Profiles," in *2019 IEEE International Symposium on Information Theory (ISIT)*, Jul. 2019, pp. 2619–2623. doi: 10.1109/ISIT.2019.8849537.

[29] C.-Y. Wang, S. H. Lim, and M. Gastpar, "Information-Theoretic Caching: Sequential Coding for Computing," *IEEE Transactions on Information Theory*, vol. 62, no. 11, pp. 6393–6406, Nov. 2016, doi: 10.1109/TIT.2016.2604851.

[30] C.-Y. Wang, S. H. Lim, and M. Gastpar, "Information-theoretic caching," in *2015 IEEE International Symposium on Information Theory (ISIT)*, Jun. 2015, pp. 1776–1780. doi: 10.1109/ISIT.2015.7282761.

[31] P. Hassanzadeh, A. Tulino, J. Llorca, and E. Erkip, "Correlation-aware distributed caching and coded delivery," in *2016 IEEE Information Theory Workshop (ITW)*, Sep. 2016, pp. 166–170. doi: 10.1109/ITW.2016.7606817.

[32] P. Hassanzadeh, A. M. Tulino, J. Llorca, and E. Erkip, "Broadcast caching networks with two receivers and multiple correlated sources," in *2017 51st Asilomar Conference on Signals, Systems, and Computers*, Oct. 2017, pp. 1449–1454. doi: 10.1109/ACSSC.2017.8335595.

[33] Q. Yang, P. Hassanzadeh, D. Gündüz, and E. Erkip, "Centralized Caching and Delivery of Correlated Contents Over Gaussian Broadcast Channels," *IEEE Transactions on Communications*, vol. 68, no. 1, pp. 122–136, Jan. 2020, doi: 10.1109/TCOMM.2019.2950930.

[34] P. Hassanzadeh, A. M. Tulino, J. Llorca, and E. Erkip, "On Coding for Cache-Aided Delivery of Dynamic Correlated Content," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 8, pp. 1666–1681, Aug. 2018, doi: 10.1109/JSAC.2018.2844579.

[35] R. Timo, S. Saeedi Bidokhti, M. Wigger, and B. C. Geiger, "A Rate-Distortion Approach to Caching," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1957–1976, Mar. 2018, doi: 10.1109/TIT.2017.2768058.

[36] Sharma and J. K. Chhabra, "Sustainable automatic data clustering using hybrid PSO algorithm with mutation," *Sustainable Computing: Informatics and Systems*, vol. 23, pp. 144–157, 2019, doi: https://doi.org/10.1016/j.suscom.2019.07.009.

[37] Cao, L. Liu, Y. Cheng, and X. Shen, "Towards Energy-Efficient Wireless Networking in the Big Data Era: A Survey," *IEEE Commun. Surv. Tutorials*, vol. 20, no. 1, pp. 303–332, 2018, doi: 10.1109/COMST.2017.2771534.

[38] C. Aggarwal and C. K. Reddy, "*Data clustering : algorithms and applications*". CRC Press, 2015.

[39] S. Wang, Y. Fang, and S. Cheng, "*Distributed Source Coding: Theory and Practice*". John Wiley & Sons, 2017.

[40] Z. Xiong, A. D. Liveris, and S. Cheng, "Distributed source coding for sensor networks," *IEEE Signal Processing Magazine*, vol. 21, no. 5, pp. 80–94, Sep. 2004, doi: 10.1109/MSP.2004.1328091.

[41] K. S. Al-Sultan, "A tabu search approach to the clustering problem," *Pattern recognition*, vol. 28, no. 9, pp. 1443–1451, 1995.

[42] S. Z. Selim and K. Alsultan, "A simulated annealing algorithm for the clustering problem," *Pattern recognition*, vol. 24, no. 10, pp. 1003–1008, 1991.

[43] R. Poll, J. Kennedy, and T. Blackwell, "Particle swarm optimization: an overview," *Swarm intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[44] D. W. Van der Merwe and A. P. Engelbrecht, "Data clustering using particle swarm optimization," in *The 2003 Congress on Evolutionary Computation, 2003. CEC'03.*, 2003, vol. 1, pp. 215–220.

[45] A. Abraham, S. Das, and S. Roy, "Swarm intelligence algorithms for data clustering," in *Soft computing for knowledge discovery and data mining*, Springer, 2008, pp. 279–313.

[46] J. Handl and J. Knowles, "An Evolutionary Approach to Multiobjective Clustering," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 1, pp. 56–76, Feb. 2007, doi: 10.1109/TEVC.2006.877146.

[47] E. Falkenauer, "Genetic algorithms and grouping problems." John Wiley & Sons, Inc., 1998.

[48] M. B. Agbaje, A. E. Ezugwu, and R. Els, "Automatic data clustering using hybrid firefly particle swarm optimization algorithm," *IEEE Access*, vol. 7, pp. 184963–184984, 2019.

[49] D. L. Davies and D. W. Bouldin, "A Cluster Separation Measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 224–227, Apr. 1979, doi: 10.1109/TPAMI.1979.4766909.

[50] C.-H. Chou, M.-C. Su, and E. Lai, "A new cluster validity measure and its application to image compression," *Pattern Anal Applic*, vol. 7, no. 2, pp. 205–220, Jul. 2004, doi: 10.1007/s10044-004-0218-1

[51] R. Kuo and F. E. Zulvia, "Automatic clustering using an improved particle swarm optimization," *J. Ind. Intell. Inf.*, vol. 1, no. 1, pp. 46_51, 2013.

[52] A. Abraham, S. Das, and A. Konar, "Kernel based automatic clustering using modified particle swarm optimization algorithm," in *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, New York, NY, USA, Jul. 2007, pp. 2–9, doi: 10.1145/1276958.1276960.

[53] S. J. Nanda and G. Panda, "Automatic clustering algorithm based on multi-objective Immunized PSO to classify actions of 3D human models," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 5, pp. 1429–1441, May 2013, doi: 10.1016/j.engappai.2012.11.008.

[54] Y. Liu, X. Wu, and Y. Shen, "Automatic clustering using genetic algorithms," *Applied Mathematics and Computation*, vol. 218, no. 4, pp. 1267–1279, Oct. 2011, doi: 10.1016/j.amc.2011.06.007.

[55] H. He and Y. Tan, "A two-stage genetic algorithm for automatic clustering," *Neurocomputing*, vol. 81, pp. 49–59, Apr. 2012, doi: 10.1016/j.neucom.2011.11.001.

[56] S. Das, A. Abraham, and A. Konar, "Automatic clustering using an improved differential evolution algorithm," *IEEE Transactions on systems, man, and cybernetics-Part A: Systems and Humans*, vol. 38, no. 1, pp. 218–237, 2007.

[57] I. Saha, U. Maulik, and S. Bandyopadhyay, "A new Differential Evolution based Fuzzy Clustering for Automatic Cluster Evolution," in *2009 IEEE International Advance Computing Conference*, Mar. 2009, pp. 706–711, doi: 10.1109/IADCC.2009.4809099.

[58] W. Lee and S. Chen, "Automatic Clustering with Differential Evolution Using Cluster Number Oscillation Method," in *2010 2nd International Workshop on Intelligent Systems and Applications*, May 2010, pp. 1–4, doi: 10.1109/IWISA.2010.5473289.

[59] R. J. Kuo, Y. D. Huang, C.-C. Lin, Y.-H. Wu, and F. E. Zulvia, "Automatic kernel clustering with bee colony optimization algorithm," *Inf. Sci.*, vol. 283, pp. 107_122, Nov. 2014.

[60] R. J. Kuo and F. E. Zulvia, "Automatic clustering using an improved artificial bee colony optimization for customer segmentation," *Knowl Inf Syst*, vol. 57, no. 2, pp. 331–357, Nov. 2018, doi: 10.1007/s10115-018-1162-5.

[61] V. Kumar, J. K. Chhabra, and D. Kumar, "Automatic Data Clustering Using Parameter Adaptive Harmony Search Algorithm and Its Application to Image Segmentation," *Journal of Intelligent Systems*, vol. 25, no. 4, pp. 595–610, Oct. 2016, doi: 10.1515/jisys-2015-0004.

[62] A. Chowdhury, S. Bose, and S. Das, "Automatic Clustering Based on Invasive Weed Optimization Algorithm," in *Swarm, Evolutionary, and Memetic Computing*, Berlin, Heidelberg, 2011, pp. 105–112, doi: 10.1007/978-3-642-27242-4_13.

[63] R. Qaddoura, H. Faris, I. Aljarah, and P. A. Castillo, "EvoCluster: An Open-Source Nature-Inspired Optimization Clustering Framework in Python," in *Applications of Evolutionary Computation*, Cham, 2020, pp. 20–36, doi: 10.1007/978-3-030-43722-0_2.

[64] S. J. Nanda and G. Panda, "A survey on nature inspired metaheuristic algorithms for partitional clustering," *Swarm and Evolutionary computation*, vol. 16, pp. 1–18, 2014.

[65] S. Das, A. Abraham, and A. Konar, "Metaheuristic clustering," vol. 178. Springer, 2009.

[66] A. José-García and W. Gómez-Flores, "Automatic clustering using nature-inspired metaheuristics: A survey," *Applied Soft Computing*, vol. 41, pp. 192–213, Apr. 2016, doi: 10.1016/j.asoc.2015.12.001.

[67] C.-W. Tsai, S.-J. Liu, and Y.-C. Wang, "A parallel metaheuristic data clustering framework for cloud," *Journal of Parallel and Distributed Computing*, vol. 116, pp. 39–49, Jun. 2018, doi: 10.1016/j.jpdc.2017.10.020.

[68] A. E. Ezugwu, "Nature-inspired metaheuristic techniques for automatic clustering: a survey and performance study," *SN Applied Sciences*, vol. 2, no. 2, p. 273, 2020.

[69] P. Gançarski, B. Crémilleux, G. Forestier, and T. Lampert, "Constrained Clustering: Current and New Trends," in *A Guided Tour of Artificial Intelligence Research*, Springer, 2020, pp. 447–484.

[70] D. Dinler and M. K. Tural, "A survey of constrained clustering," in *Unsupervised learning algorithms*, Springer, 2016, pp. 207–235.

[71] Z. Drezner, "The p-centre problem—heuristic and optimal algorithms," *Journal of the Operational Research Society*, vol. 35, no. 8, pp. 741–748, 1984.

[72] I. Davidson and S. S. Ravi, "Agglomerative hierarchical clustering with constraints: Theoretical and empirical results," in *European Conference on Principles of Data Mining and Knowledge Discovery*, 2005, pp. 59–70.

[73] S. Ghiasi, A. Srivastava, X. Yang, and M. Sarrafzadeh, "Optimal Energy Aware Clustering in Sensor Networks," *Sensors*, vol. 2, no. 7, pp. 258–269, Jul. 2002, doi: 10.3390/s20700258.

[74] L. Li, G. Zhao, and R. S. Blum, "A Survey of Caching Techniques in Cellular Networks: Research Issues and Challenges in Content Placement and Delivery Strategies," *IEEE Communications Surveys Tutorials*, vol. 20, no. 3, pp. 1710–1732, 2018, doi: 10.1109/COMST.2018.2820021.

[75] N. Dimokas, D. Katsaros, and Y. Manolopoulos, "Cooperative Caching in Wireless Multimedia Sensor Networks," *Mobile Netw Appl*, vol. 13, no. 3, pp. 337–356, Aug. 2008, doi: 10.1007/s11036-008-0063-3.

[76] M. Arlitt, L. Cherkasova, J. Dilley, R. Friedrich, and T. Jin, "Evaluating content management techniques for Web proxy caches," *SIGMETRICS Perform. Eval. Rev.*, vol. 27, no. 4, pp. 3–11, Mar. 2000, doi: 10.1145/346000.346003.

[77] K. Wan, D. Tuninetti, and P. Piantanida, "A novel index coding scheme and its application to coded caching," in *2017 Information Theory and Applications Workshop (ITA)*, Feb. 2017, pp. 1–6. doi: 10.1109/ITA.2017.8023478. Index coding with side information

[78] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000, doi: 10.1109/18.850663.

[79] M. Effros, S. El Rouayheb, and M. Langberg, "An Equivalence Between Network Coding and Index Coding," *IEEE Transactions on Information Theory*, vol. 61, no. 5, pp. 2478–2487, May 2015, doi: 10.1109/TIT.2015.2414926.

[80] Y. Birk and T. Kol, "Coding on demand by an informed source (ISCOD) for efficient broadcast of different supplemental data to caching clients," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2825–2830, Jun. 2006, doi: 10.1109/TIT.2006.874540.

[81] T. M. Cover and J. A. Thomas, Elements of Information Theory. John Wiley & Sons, 2012.

[82] S. Wang, Y. Fang, and S. Cheng, *Distributed Source Coding: Theory and Practice*. John Wiley & Sons, 2017.

[83] S. Mirjalili, S. M. Mirjalili, and X.-S. Yang, "Binary bat algorithm," *Neural Comput & Applic*, vol. 25, no. 3–4, pp. 663–681, Sep. 2014, doi: 10.1007/s00521-013-1525-5.

[84] S. Mirjalili and A. Lewis, "S-shaped versus V-shaped transfer functions for binary particle swarm optimization," *Swarm and Evolutionary Computation*, vol. 9, pp. 1–14, 2013.

[85] S. Mirjalili, "Genetic algorithm," in *Evolutionary algorithms and neural networks*, Springer, 2019, pp. 43–55.

[86] S. Mirjalili, "Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems," *Neural Comput & Applic*, vol. 27, no. 4, pp. 1053–1073, May 2016, doi: 10.1007/s00521-015-1920-1.

[87] X.-S. Yang, "A New Metaheuristic Bat-Inspired Algorithm," in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, J. R. González, D. A. Pelta, C. Cruz, G. Terrazas, and N. Krasnogor, Eds. Berlin, Heidelberg: Springer, 2010, pp. 65–74. doi: 10.1007/978-3-642-12538-6_6.

[88] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Computational Cybernetics and Simulation 1997 IEEE International Conference on Systems, Man, and Cybernetics*, Oct. 1997, vol. 5, pp. 4104–4108 vol.5. doi: 10.1109/ICSMC.1997.637339.

[89] G.-C. Luh, C.-Y. Lin, and Y.-S. Lin, "A binary particle swarm optimization for continuum structural topology optimization," *Applied Soft Computing*, vol. 11, no. 2, pp. 2833–2844, Mar. 2011, doi: 10.1016/j.asoc.2010.11.013.

[90] P.-Y. Yin, "A discrete particle swarm algorithm for optimal polygonal approximation of digital curves," *Journal of Visual Communication and Image Representation*, vol. 15, no. 2, pp. 241–260, Jun. 2004, doi: 10.1016/j.jvcir.2003.12.001.

[91] Q. Shen, J.-H. Jiang, C.-X. Jiao, G. Shen, and R.-Q. Yu, “Modified particle swarm optimization algorithm for variable selection in MLR and PLS modeling: QSAR studies of antagonism of angiotensin II antagonists,” *European Journal of Pharmaceutical Sciences*, vol. 22, no. 2, pp. 145–152, Jun. 2004, doi: 10.1016/j.ejps.2004.03.002.

[92] “The Binary Genetic Algorithm,” in *Practical Genetic Algorithms*, John Wiley & Sons, Ltd, 2003, pp. 27–50. doi: 10.1002/0471671746.ch2.

[93] Mishra, “Binary Genetic Algorithm,” *Medium*, Feb. 26, 2021. https://prateek-mishra.medium.com/binary-genetic-algorithm-19936e755271 (accessed Sep. 06, 2022).

[94] R. Xu and D. WunschII, “Survey of Clustering Algorithms,” *IEEE Trans. Neural Netw.*, vol. 16, no. 3, pp. 645–678, May 2005, doi: 10.1109/TNN.2005.845141.

[95] G. Gan, C. Ma, and J. Wu, “*Data clustering: theory, algorithms, and applications*.” SIAM, Society for Industrial and Applied Mathematics, 2007.

[96] C. K. Reddy and B. Vinzamuri, "A survey of partitional and hierarchical clustering algorithms", *Proc. Data Clustering Algorithms Appl.*, pp. 87-110, 2013.

[97] C. Boutsidis, P. Drineas, and M. W. Mahoney, “Unsupervised feature selection for the k-means clustering problem,” in *Advances in Neural Information Processing Systems*, 2009, pp. 153–161.

[98] C. Ding and X. He, “Cluster merging and splitting in hierarchical clustering algorithms,” in *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, 2002, pp. 139–146.

[99] Z. Huang and M. K. Ng, “A fuzzy k-modes algorithm for clustering categorical data,” *IEEE transactions on Fuzzy Systems*, vol. 7, no. 4, pp. 446–452, 1999.

[100] N. R. Pal and J. C. Bezdek, “On cluster validity for the fuzzy c-means model,” *IEEE Transactions on Fuzzy Systems*, vol. 3, no. 3, pp. 370–379, Aug. 1995, doi: 10.1109/91.413225.

[101] K. Bache and M. Lichman. (2013). UCI Machine Learning Repository. University of California, School of Information and Computer Science, Irvine, CA, USA. [Online]. Available: http://archive.ics.uci.edu/ml/

[102] J. Alcalá-Fdez *et al.*, “Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework.,” *Journal of Multiple-Valued Logic & Soft Computing*, vol. 17, 2011.

[103] D. Arthur and S. Vassilvitskii, "K-means+ +: The advantages of careful seeding", pp. 1027-1035, Jan. 2007.

[104] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise.,” in *Kdd*, 1996, vol. 96, no. 34, pp. 226–231.

[105] J. Han, M. Kamber, and J. Pei, “Data Mining: Concepts and Techniques - 3rd Edition.” The Morgan Kaufmann Series in Data Management Systems, 2011.

[106] R. Qaddoura, H. Faris, and I. Aljarah, “An efficient clustering algorithm based on the k-nearest neighbors with an indexing ratio,” *Int. J. Mach. Learn. & Cyber.*, vol. 11, no. 3, pp. 675–714, Mar. 2020, doi: 10.1007/s13042-019-01027-z.

[107] S. Bandyopadhyay and U. Maulik, “Genetic clustering for automatic evolution of clusters and application to image classification,” *Pattern recognition*, vol. 35, no. 6, pp. 1197–1208, 2002.

[108] M. G. H. Omran, A. Salman, and A. P. Engelbrecht, “Dynamic clustering using particle swarm optimization with application in image segmentation,” *Pattern Anal Applic*, vol. 8, no. 4, p. 332, Nov. 2005, doi: 10.1007/s10044-005-0015-5.

[109] E. Martin, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise.,” in *Kdd*, 1996, vol. 96, no. 34, pp. 226–231.

[110] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, “DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN,” *ACM Trans. Database Syst.*, vol. 42, no. 3, p. 19:1-19:21, Jul. 2017, doi: 10.1145/3068335.

[111] M. Friedman, “A Comparison of Alternative Tests of Significance for the Problem of m Rankings,” *The Annals of Mathematical Statistics*, vol. 11, no. 1, pp. 86–92, 1940.

[112] M. Friedman, “The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance,” *Journal of the American Statistical Association*, vol. 32, no. 200, pp. 675–701, 1937.

[113] J. Derrac, S. García, D. Molina, and F. Herrera, “A practical tutorial on the use of non-parametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms,” *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.

[114] S. García, D. Molina, M. Lozano, and F. Herrera, “A study on the use of non-parametric tests for analyzing the evolutionary algorithms’ behaviour: a case study on the CEC’2005 Special Session on Real Parameter Optimization,” *J Heuristics*, vol. 15, no. 6, p. 617, 2008.

[115] S. J. Nanda, R. Raman, S. Vijay, and A. Bhardwaj, “A new density based clustering algorithm for Binary data sets,” in *2014 International Conference on High Performance Computing and Applications (ICHPCA)*, 2014, pp. 1–6.

[116] C. Ahn, F. Hu, and W. R. Schucany, “Sample Size Calculation for Clustered Binary Data with Sign Tests Using Different Weighting Schemes,” *Statistics in Biopharmaceutical Research*, vol. 3, no. 1, pp. 65–72, Feb. 2011.

[117] V. Verma, and R. K. Aggarwal, “A New Similarity Measure Based on Simple Matching Coefficient for Improving the Accuracy of Collaborative Recommendations,” *IJITCS*, vol. 11, no. 6, pp. 37–49, Jun. 2019, doi: 10.5815/ijitcs.2019.06.05.

[118] E. W. Weisstein, “Stirling Number of the Second Kind.” https://mathworld.wolfram.com/StirlingNumberoftheSecondKind.html (accessed Oct. 07, 2021).

[119] Q. Yang and D. Gündüz, “Centralized coded caching for heterogeneous lossy requests,” in *2016 IEEE International Symposium on Information Theory (ISIT)*, Jul. 2016, pp. 405–409. doi: 10.1109/ISIT.2016.7541330.

[120] G. J. O. ’t Veld and M. Gastpar, “Caching (Bivariate) Gaussians,” *IEEE Transactions on Information Theory*, vol. 66, no. 10, pp. 6150–6168, Oct. 2020, doi: 10.1109/TIT.2020.3001176.

[121] A. Abrardo, G. Ferrari, M. Martalò, M. Franceschini, and R. Raheli, “Orthogonal Multiple Access With Correlated Sources: Achievable Region and Pragmatic Schemes,” *IEEE Transactions on Communications*, vol. 62, no. 7, pp. 2531–2543, Jul. 2014, doi: 10.1109/TCOMM.2014.2325039.

[122] X. Zhou, X. He, M. Juntti, and T. Matsumoto, “Outage probability of correlated binary source transmission over fading multiple access channels,” in *2015 IEEE 16th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Jun. 2015, pp. 96–100. doi: 10.1109/SPAWC.2015.7227007.

[123] G. J. Op ’t Veld and M. C. Gastpar, “Caching Gaussians: Minimizing total correlation on the Gray-Wyner network,” in *2016 Annual Conference on Information Science and Systems (CISS)*, Mar. 2016, pp. 478–483. doi: 10.1109/CISS.2016.7460549.

[124] Y. Deng and M. Dong, “Fundamental Structure of Optimal Cache Placement for Coded Caching With Nonuniform Demands,” *IEEE Transactions on Information Theory*, vol. 68, no. 10, pp. 6528–6547, Oct. 2022, doi: 10.1109/TIT.2022.3179266.

[125] J. Hachem, N. Karamchandani, and S. N. Diggavi, “Coded Caching for Multi-level Popularity and Access,” *IEEE Transactions on Information Theory*, vol. 63, no. 5, pp. 3108–3141, May 2017, doi: 10.1109/TIT.2017.2664817.

[126] B. Merikhi, M. Soleymani, “Cache-Aided Networks with Shared Caches and Correlated Content under Non-Uniform Demands,” 2023 20th IEEE Consumer Communications & Networking Conference (CCNC), Jan. 8-11, 2023.

[127] B. Merikhi and M. R. Soleymani, “Cache-Aided Delivery Network in a Shared Cache Framework with Correlated Sources,” in Proceedings of the 18th ACM International Symposium on QoS and Security for Wireless and Mobile Networks on 18th ACM International Symposium on QoS and Security for Wireless and Mobile Networks, New York, NY, USA, Oct. 2022, pp. 121–129. doi: 10.1145/3551661.3561372.

[128] B. Merikhi and M. R. Soleymani, “Automatic Data Clustering Framework Using Nature-Inspired Binary Optimization Algorithms,” IEEE Access, vol. 9, pp. 93703–93722, 2021, doi: 10.1109/ACCESS.2021.3091397.

[129] B. Merikhi, S. M. Mirjalili, M. Zoghi, S. Z. Mirjalili, and S. Mirjalili, “Radiation pattern design of photonic crystal LED optimized by using multi-objective grey wolf optimizer,” Photon Netw Commun, vol. 38, no. 1, pp. 167–176, Aug. 2019, doi: 10.1007/s11107-019-00843-1.