Information-Theoretic Clustering Techniques for Correlated Sources in Network Content Caching

Benjamin Rosen

A thesis submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Doctor of Philosophy.

Johannesburg, September 2025

Declaration

I declare that this thesis is my own, unaided work, except where otherwise acknowledged. It is being submitted for the degree of Doctor of Philosophy to the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination to any other university.

Signed this 4th day of September 2025

Benjamin Rosen

Abstract

Two powerful and complementary tools have been presented in literature to facilitate the paradigm shift planned for the Internet, where the focus is on storing information within the network infrastructure itself. The first involves Slepian-Wolf (SW) coding, in which the high degree of correlation between pieces of information are exploited to reduce the overall size of the data. Then, Coded Caching (CC) stores parts of the information at the user end during off-peak hours when demand is low. Although this hybrid system results in an impressive reduction of bandwidth usage during peak hours, the SW coding operation in particular becomes prohibitively complex to implement. Clustering has been proposed in other fields pertaining to SW coding to solve this issue, but the assumptions and simplifications used there are not applicable in this scenario.

This work solves the problem of clustering within a hybrid Slepian-Wolf Coded Caching (SW/CC) system by first looking at a simplified single-group model. To this end, two iterative algorithms are developed and compared to the Genetic Algorithm (GA) and Ant Colony Optimisation (ACO) meta-heuristic approaches. Then, the problem is extended to multiple groups, which presents a much greater challenge. The single group algorithms are expanded and improved to cater for the new paradigm, providing close-to-optimal approaches while still running in polynomial time. At the same time, necessary conditions for optimality are derived for both cases, allowing for solutions to be further improved.

The iterative approaches in particular, when used in conjunction with the optimality tests, perform comparably to the benchmark meta-heuristic ones, but with more predictable performance and much lower complexity in both the single and multi-group scenarios. Clustering the files in this way greatly reduces the complexity in this system, making it more feasible to implement in practical scenarios. More research can be done to improve these algorithms, generalise the problem to arbitrarily sized groups and allow for files to belong to more than one group.

For Hashem, an insignificant repayment for constantly sustaining me.

For my parents, without whose support and upbringing I never could have begun...

...and for Chana, Rivka and Avigail Tzerel, whom I never would have dreamed would be here when I started, but without whom I would never dream of finishing.

Finally, for those unborn (thoughts and otherwise), brimming with infinite potential and exciting possibilities.

Acknowledgements

I would like to thank the Council for Scientific and Industrial Research (CSIR) for funding three years of my Ph.D. studies. This financial peace of mind allowed me to focus solely on deepening my understanding of my topic and developing my contributions. I also thank my co-supervisor, Professor Adnan Abu-Mahfouz, for his assistance and comments on my papers, helping me to polish the content and presentation of an earlier paper attempt, which was eventually published in a reputable journal.

Next, the hard work of the support staff did not go unnoticed. Without their continued assistance and support, it would have been nearly impossible to manage the complex university systems. Thank you in particular, but in no particular order, to Jeannine, Jeanne, Moeniera, and Mumtaz! You were always in my corner since I started my studies all those years ago.

I want to formally acknowledge the support of my family, siblings and brothers-in-law. I especially appreciated the constant concern about whether the thesis was completed yet and when that would be the case.

Last but most importantly, I am eternally grateful to Professor Ling Cheng for his outstanding supervision, mentorship, and guidance. I enjoyed our "blackboard boxing", which sharpened my presentation and understanding of my concepts, and the "therapy sessions", where we delved deeply into my psyche and explored why I work the way I do. I trust that the time and effort you invested in me were properly reciprocated.

I hope to one day release the dumpling from the teapot...

List of Publications

This thesis is mainly comprised of the following papers which have been published in *IEEE Access*. The literature review and background of these papers have been combined to form Chapters 2 and 3.

[1] B. Rosen and L. Cheng, "Greedy iterative and meta-heuristic clustering with coded caching and Slepian-Wolf compression for correlated content," *IEEE Access*, vol. 12, pp. 12 323–12 334, Jan. 2024. The content of this paper is contained in Chapter 4.

[2] B. Rosen, A. M. Abu-Mahfouz, and L. Cheng, "Path-based clustering approaches for a hybrid Slepian-Wolf compression and coded caching system," *IEEE Access*, vol. 13, pp. 48 935–48 949, Mar. 2025. The content of this paper features primarily in Chapter 5.

Contents

List of Figures

List of Tables

List of Algorithms

Nomenclature

0-1K	0-1 Knapsack
ACO	Ant Colony Optimisation
ACS	Ant Colony System
AS	Ant System
BCH	Bose–Chaudhuri–Hocquenghem
BF	Brute Force
BtW	Best-to-Worst
CC	Coded Caching
CNA	Candidate Node Assignment
CNS	Candidate Node Selection
CSP	Champion Selection Procedure
CSP-AG	CSP with Ancestor Guarantee
CSP-CP	CSP with Convergence Prevention
DJSCC	Distributed Joint Source-Channel Coding
DSC	Distributed Source Coding
ECC	Error Correction Code
GA	Genetic Algorithm
GISGEM	Greedy Iterative Single Group Entropy Minimisation
HC	High Complexity
ICN	Information-Centric Network

i.i.d.	independent and identically distributed
IIOI-MG	Individual IOI: Multi-Group
INS	Initial Node Selection
IOI	Iterative Optimisation Improvement
IOI-MG	IOI: Multi-Group
IP	Internet Protocol
JSCC	Joint Source-Channel Coding
LC	Low Complexity
LDPC	Low-Density Parity-Check
MIA	Mutual Information Area
MNI	Maximum Number of Iterations
MP	Matrix Partitioning
MWSC	Minimum Weight Set Covering
SG	Stall Generations
SW	Slepian-Wolf
SW/CC	Slepian-Wolf Coded Caching
TreB-MAP	Trellis-Based Multi-group Allocation Procedure
UAV	Unmanned Aerial Vehicle
WSN	Wireless Sensor Network
WtB	Worst-to-Best
XOR	Exclusive Or

Chapter 1

Introduction

With the advent of the Internet and the advancement of high-speed network access, the demand for consumer content has increased exponentially. Although the size of the network bandwidth is consistently increased, it is not able to keep up with the demand increase. This is especially due to the asymmetric load that is put on the networks. Consumers will often download content during the same periods in the day, known as peak hours. This further compounds the issue, since the load is not balanced and the network bandwidth has to be designed to facilitate the highest load. Furthermore, upgrading the network capacity is an involved and expensive process.

Together with this challenge, a number of opportunities have also presented themselves. Firstly, the cost of computing has decreased rapidly with the consumerisation of personal computation devices. This enables distributed computing, as the tasks previously only possible on an expensive central computer can be allocated to many smaller devices that have the same capability but at a fraction of the cost.

So too, there have been advances in the storage capacity of these edge devices. For example, entry-level smartphones consistently come with over 64 GB of storage. Finally, the content that is consumed has a high degree of correlation. Consumers tend to view similar content (current news, viral videos, popular TV shows, etc.).

A new network architecture has been proposed to replace the current Internet Protocol (IP) model. The IP structure relies on a rigid client-server model, which greatly inhibits the ability of the network to continue to scale at its current rate. Called Information-Centric Networks (ICNs), the new approach allows information to reside at any level of the network in a distributed manner. However, the design of such a system is still in its infancy and requires advanced techniques to handle such a shift.

Distributed Source Coding (DSC) (also known as Slepian-Wolf (SW) coding) has been proposed to leverage the increase in edge computing power. First described by Slepian and Wolf [3], this provides a framework to compress correlated data in a distributed fashion. They showed that it is possible to achieve this without communication between the devices and with no loss in compression quality.

Caching is a technique that takes advantage of increased storage capacity at the user end. This involves storing pieces of information at end users during off-peak hours. Then, during peak hours, consumers only need to receive part of the data and combine it with their local cache to obtain the desired content. A highly effective approach to achieve this is the Coded Caching (CC) technique, which intelligently fills the cache storage so as to greatly reduce the peak hour bandwidth usage.

Combining the two in a hybrid Slepian-Wolf Coded Caching (SW/CC) system is studied relatively recently. An advantage of the hybrid approach is that the two techniques are independent, meaning that they can be combined without any negative interference. As a result, the caching aspect reduces the overall load on the network, smoothing out the demand. At the same time, DSC is able to reduce the size of the data stored on the end devices, effectively increasing the amount of information that can be cached, leading to a further reduction of the load on the network infrastructure. At the same time, these techniques are well-suited to distributed settings, making them highly compatible with the ICN architecture.

A major limitation of SW coding is that the information from all of the nodes are required for decoding, resulting in two issues. The one is that the decoding becomes expensive as more nodes are included. The other is that, if some of the information is lost, then none of the data can be recovered. As a result, this research aims to reduce the complexity and increase the robustness of the SW coding using clustering. The general idea is to partition the files into different groups, such that the complexity and robustness are bounded (based on the number of nodes in that group). The challenge is then to find the best clustering of the nodes, so that the SW compression is maximised. This also ensures that the SW/CC system as a whole is optimised in terms of time complexity.

1.1 Research Questions

The following research questions will be answered in the course of this research:

What approaches can be used to group nodes in a hybrid correlated source coding and content caching system that optimise the overall coding gains?

The sub-questions following from this are:

Can algorithms be constructed to solve the clustering problem, that run in polynomial time while still producing near optimal results?

Do iterative or meta-heuristic approaches find better clusters?

Can the optimality of a clustering be guaranteed?

1.2 Contributions

This research aims to comprehensively define the clustering problem within a hybrid SW/CC system. It will be shown how the problem has not been sufficiently addressed in literature to date and why the problem is worthwhile solving. The problem itself is solved in two different forms, the first of which is the case where files are divided into two groups. This is then extended to an arbitrary number of groups. Two novel algorithms are designed to solve the former, while another novel approach is created to solve the latter. In addition, two meta-heuristic algorithms are adapted to solve these problems, which are used as solutions in their own right and as benchmarks with which to evaluate the novel approaches.

1.3 Layout

The research is organised as follows: in Chapter 2, the necessary literature is examined to determine what has already been accomplished and where it is lacking. Chapter 3 then establishes the background and mathematical definitions used to model the system, the problem formulations, the methods with which to analyse it and the metrics with which to evaluate potential solutions.

The rest of the research looks at the problem from two different angles. In the first, a simplified version of this question is analysed, in which the library of files is partitioned into two different groups. The first of these is compressed using DSC

while the other is left uncoded. The goal is to find a single cluster that optimises the coding gains. This approach itself is divided into two. Chapter 4 solves the single coded group problem in a greedy manner, returning a single solution. The iterative approach is benchmarked using the Genetic Algorithm (GA) meta-heuristic. Chapter 5 extends this by examining multiple potential solutions and branching off accordingly, eventually choosing the most optimal one. It is compared to the Ant Colony Optimisation (ACO) meta-heuristic, as well as the results from Chapter 4.

Chapter 6 takes a second view of the problem, where the approach is generalised to multiple groups, all of which are coded. Here, the goal is to maximise the coding gains for the entire system. It is an extension of Chapter 5, but combines the different paths generated in that solution to produce a single grouping that conforms to the restrictions of the new problem formulation. The GA solution developed in Chapter 4 is found to be the most suitable for this new regime and is adapted accordingly to provide a benchmarking algorithm. Finally, Chapter 7 concludes this thesis, critically analysing the findings presented and identifying avenues for future work.

1.4 Conclusion

The hybrid SW/CC system has the potential to support the substantial change in network design. However, the complexity of such a system demands that a clustering approach is adopted, which itself is a complex and unsolved problem in literature. This research tackles this problem, with a framework for solving the simpler two-cluster problem before extending this to multiple groups. So too, the research aims to adapt existing meta-heuristic algorithms to solve these problems.

Chapter 2

Literature Review

The concepts used in this thesis are now explored in terms of the literature. These are examined from their seminal works and the progression of the salient points and innovations are given until the state-of-the-art. In doing so, it becomes readily apparent where the gaps in the literature are and how this work fits in with the existing knowledge.

2.1 Introduction

SW coding and CC are well-developed fields, but have largely been worked on independently, often with different application areas. Nevertheless, there have been some recent developments that take advantage of the overlap between these two techniques. This chapter aims to outline the progression of each of these fields and the nascent research in combining them, the general structure of which is shown in Figure 2.1. First, Section 2.2 looks at the literature pertaining to SW coding and its techniques and advancements before Section 2.3 addresses the research into CC. Section 2.4 shows how recent developments have combined the two techniques and also where the gaps currently lie in this line of research. Section 2.5 summarises the findings of this chapter.

2.2 Distributed Source Coding

Slepian and Wolf, in their foundational paper, outlined the potential to perform DSC, where the information from correlated sources are able to be compressed individually [3]. This was extended to multiple ergodic nodes by Cover, who also

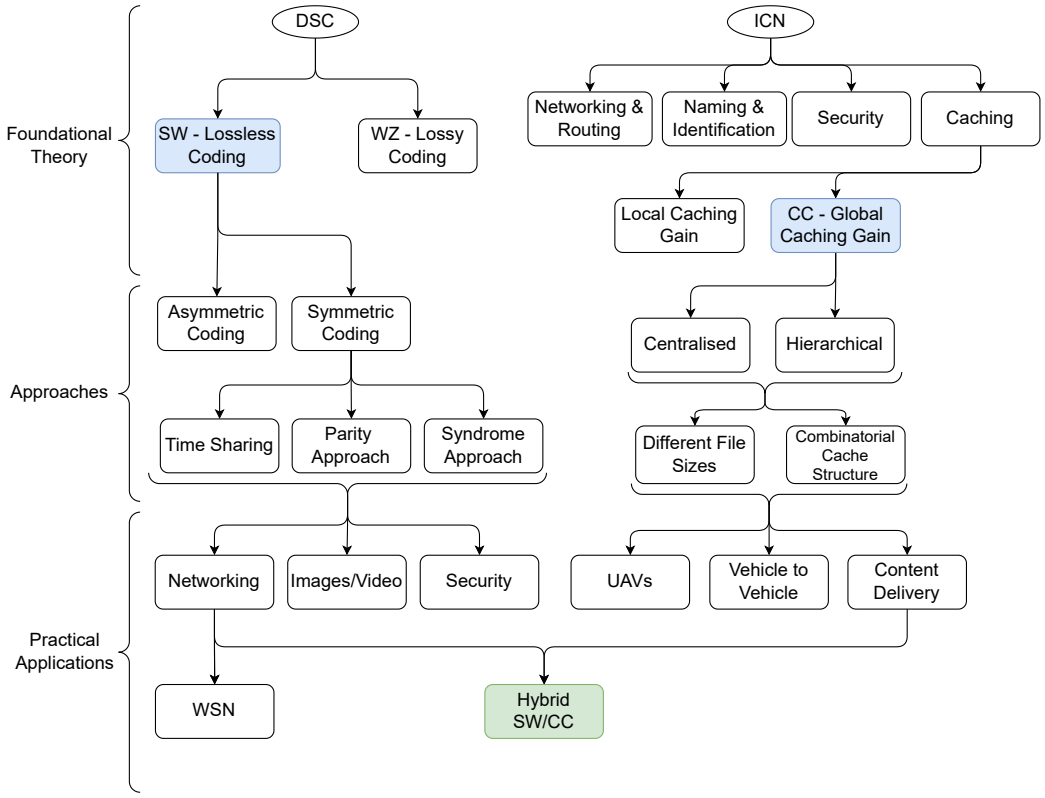


Figure 2.1: A graphical overview of the literature pertaining to a hybrid SW/CC system

simplified the proof of this approach [4]. This spawned an entire field dedicated to finding practical methods to realise this. SW coding is an example of lossless DSC, where the information is retrieved without error. Wyner and Ziv proposed a lossy version of this in [5].

In general, the SW approach is to treat the information from the sources as noisy versions of each other. Then, using Error Correction Codes (ECCs), the syndromes can be detected and corrected [6]. In particular, Pradhan and Ramchandran proposed asymmetric [7] and symmetric [8] approaches. In the asymmetric approach, one of the nodes transmits full, uncompressed information, while the other compresses its information using an ECC. In contrast to this, the symmetric approach uses Matrix Partitioning (MP) to divide the parity check matrix of the ECC among the different sources [9–11]. The number of rows assigned to each node decides the compression that it can achieve. In this way, the amount of information can be tuned to the channel bandwidth that each node possesses, such that the system as a whole is balanced.

A plethora of ECCs have been used to perform SW coding. Stanković *et al.* [12]

use systematic Hamming codes to realise their implementation, while others use Bose–Chaudhuri–Hocquenghem (BCH) codes [13, 14]. Based on their performant characteristics, Low-Density Parity-Check (LDPC) codes are popular in performing SW coding [15–17]. So too, turbo codes have been found to be feasible [18–20], where in this case it is necessary to feed the syndromes as extrinsic information to the decoders, which continuously perform the decoding until consensus is reached. Even rateless codes such as fountain codes have been transformed to implement SW coding, with promising results [21, 22].

SW coding has found particular success as a component in a Wireless Sensor Network (WSN) context, since the distributed nature of the problem is well suited to this type of compression [23]. Here, the issue is in matching the correct level of compression with the rates available to each node, such that the system's throughput is balanced while preserving the lifetime of the nodes [24–27]. Han first provided a theoretical framework with which to implement SW coding in a network context, where information is viewed in terms of 'flows' [28]. Roumy and Gesbert, in [29], use pairwise SW coding to achieve the rate minimisation. On the other hand, Li and Ramamoorthy [16] use a system of linear equations, which can be solved using their algorithm. Barros and Servetto [30] design a practical protocol stack to maximise the throughput of a network with many sensors and a single sink. Cristescu *et al.* show that, if scaled to an arbitrary size, the networked SW problem can become NP-complete [31].

When the correlation between nodes is unknown, Garcia-Frias [18] presents an iterative, turbo code implementation that estimates the correlation and does not show significant performance degradation. The iterative method involves using two turbo code decoders that share information at each step of decoding for a number of iterations. The iterations provide a trade-off between decoding speed and accuracy.

Another method to combat unknown correlation is presented by Toto-Zarasoa *et al.* in [15]. Their design uses the matrix partitioning method of Pradhan and Ramchandran (as discussed above) for LDPC codes, but puncture additional bits for increased compression. In the event that the decompression algorithm does not converge, more bits are successively requested from the nodes until the decompression succeeds. The number of bits needed becomes the new rate of transmission of the nodes, allowing for optimisation while the system is functioning. Forchhammer *et al.* [13] propose a similar method, but using BCH syndromes. After a certain number of decoding passes, the algorithm evaluates whether the error pattern is closer to being obtained. If not, the server requests a new syndrome. They also include a method to choose a BCH code with only an estimate as to the correlation between

nodes. In a more recent work, Mo *et al.* combine arithmetic coding with SW LDPC coding in a hybrid system, with the arithmetic source coding technique providing the extrinsic information so that the SW code can function without knowing the correlation beforehand [32].

A major limitation of SW coding is that it requires the information from all of the sources in order to decode correctly. This means that the joint decode becomes exponentially more expensive as more sources are added to the scheme. In addition, the likelihood of a source's information being lost increases, making the system more fragile. Added to this is that not all of the sources' information is strongly correlated. Thus, it is possible that the increase in decode complexity and fragility do not result in more overall compression.

Wang *et al.* [33] were the first to point this out. They provide a heuristic clustering algorithm to solve this problem. First, the nodes are organised into clusters using a distance metric. Then, the power set of each cluster is computed, resulting in various combinations of subsets. The algorithm greedily chooses the best performing subset (in terms of the total entropy of the subset) at each step, and removes the nodes that are selected as part of this subset. The algorithm continues until all nodes have been selected. They extended their work to increase the security [34] and overall compression [35] of their system. Zheng, one of the authors of the above papers, used the same approach and added robustness by electing backup nodes [36] and energy efficiency by correlating the chance of becoming a cluster head to the distance from the next hop in the network [37]. Clustering has also been performed in an SW context using traditional clustering methods such as the weighted pair group method with arithmetic mean [38].

Yang *et al.* [39, 40] use a similar approach to Wang *et al.*, which assumes that the correlation structure can be predicted and that nodes that are not in the immediate vicinity of one another are assumed to have weak correlation, meaning that these nodes are not to be put in the same cluster. This makes sense in a WSN environment, where nodes are correlated based on the observations that they make. Thus, those that are spatially close to one another are more likely to have a higher degree of correlation. Their novelty lies in their solution, in which Lagrangian multipliers are used and solved in layers, with the solution for inner layers being used to update the outer layers. More recently, Amutha *et al.* use a Sail Fish meta-heuristic approach to solve this problem [41].

2.3 Network Caching

Research in network structure is shifting towards ICNs [42, 43], a state-of-the-art networking paradigm shift in which the information is stored in the network components themselves, as opposed to the traditional IP client-server architecture. This new regime requires a focus on distributed techniques, which demands a high degree of flexibility in terms of routing structure [44, 45] while maintaining a strong level of security [46, 47].

An integral component of ICNs is caching, allowing data to be stored at the user end during hours when the network usage is low [48]. In turn, this reduces the load on the network infrastructure during peak hours, since the cached content can be partially delivered using the local cache contents. Extensive research has been done on the optimal methods for improved placement of the information, as well as the methods used to efficiently deliver the content when requested [49]. So too, emphasis is placed on distributed methods, in order to maintain flexibility and scalability when implementing caching and coding in real-world scenarios [50, 51].

As noted in [43] however, scalability remains a current issue. If ICNs are to replace the IP model, where information is stored in the network itself instead of clients directly communicating with servers, it is necessary to implement flexible caching methods that are efficient while keeping the complexity low. This would allow ICNs to handle the flow of large amounts of data currently supported by the IP model and allow it to continue to grow with the increasing bandwidth demands. Traditional caching methods, however, are not suited to ICNs, since they focus on maximising local coding gains.

Maddah-Ali and Niesen [52] were the first to devise a unique coding scheme for network content caching that maximises global coding gains in addition to the local coding gains. Named Coded Caching, the system operates by slicing the different pieces of information to be delivered into different subsets. These subsets are intelligently stored at the user end, such that each user has unique contents in its cache. When the users request the information, the server is able to satisfy any possible demand by determining which subsets are not stored at the user end for each file requested. The Exclusive Or (XOR) function is used to combine all of these together and the resulting codeword is broadcast to the users. At the user end, their requested information is obtained by XORing the other partial information that they have stored previously with the codeword. This provides a low complexity way to obtain the full information for all users, while minimising the number of bits sent

over the channel.

This in turn spawned a flurry of activity, with methods being devised that are able to handle different file sizes [53, 54]. So too, the approach has been made more scalable using centralised [55] and hierarchical [56] techniques. Sojdeh *et al.* [57] improved the security of the system by implementing shared keys in a multi-server scenario. This form of caching has been shown to be fundamental in the shift towards ICNs, and has found applications in co-operative environments such as device to device [58], Unmanned Aerial Vehicle (UAV) [59] and terrestrial vehicle [48, 60, 61] communications.

More recently, Brunero and Elia fundamentally analyse the generalisation of CC, in which users can be connected to multiple caches without restriction [62]. They show that the gains are even more impressive using this paradigm.

2.4 Hybrid Slepian-Wolf Coded Caching

Hassanzadeh *et al.* [63, 64] recognised the potential of combining DSC and CC together, which has a twofold effect: Firstly, after performing the source coding, less information needs to be physically transmitted over the network during off-peak periods, meaning that more information can be delivered during these times. Additionally, the partial codewords that are stored at the user end are reduced in size, resulting in more information stored per bit of physical storage space, further increasing the effectiveness of the system. In their work, they derive the upper bounds of such a system, and use a greedy colouring algorithm to distribute the files to be cached amongst the users. They find that this system outperforms ones that do not take correlation into account. By simplifying the model, a more optimal placement scheme was designed in [65]. This was made more systematic by incorporating Gray-Wyner coding into the compression design [66].

The trade-off of this approach is that the system requires two decoding phases, one for the CC and one for the SW coding. As mentioned, the CC decode is not complex, relying on simple XOR operations. However, a large grouping in the DSC phase will be prohibitively complex, as discussed with respect to WSNs. Indeed, Hassanzadeh *et al.* note in [66] that the problem becomes exponentially difficult to solve optimally. They note this with regards to Gray-Wyner coding, the chosen DSC approach used in that paper. Nevertheless, SW coding has a similar goal and structure to Gray-Wyner coding and suffers from the same exponential increase in complexity [67]. Owing to

the complexity, they only analyse cases with two files and multiple users, and three files and two users. Although Merikhi and Soleymani extend this to a scenario with multiple servers, their clustering only uses groups of two files in the SW clustering phase [68, 69].

It could be posited that the solutions to the clustering problem from WSNs discussed in Section 2.2 could be adapted to this scenario. However, it is now argued that the assumptions and simplifications from the papers described above do not apply here.

In the WSN context, the information comes from nodes' observations of an event. Thus, a correlation can be assumed to exist between nodes that are spatially close to one another. For the same reason, the underlying distribution of the correlation can also be assumed, based on the knowledge of the phenomenon being observed by the nodes (such as the propagation of the measurable data in the medium where the WSN is implemented). In the SW/CC scenario, however, the nodes are a collection of randomly correlated files instead, meaning that no knowledge of the underlying causes of the correlation between files is available before receiving them. Even though techniques exist to estimate the correlation, these require multiple transmissions in order to be effective, which is undesirable in this scenario, since the goal is to reduce the data transmitted over the network itself. Thus, a more specific correlation model must be developed, as well as a clustering algorithm that does not take into account the above assumptions and simplifications. This is not addressed in literature to date.

2.5 Conclusion

This chapter has outlined the knowledge surrounding the focus of this research. Robust literature exists for SW coding and CC independently and they have both been implemented in networking environments. However, the hybrid SW/CC system is a relatively new concept. More particularly, although the complexity of the SW coding component of the system has been addressed in a WSN environment in the form of clustering, these have not been examined in the hybrid environment when the focus is on content delivery. This chapter has made the case that the current techniques to date fall short in this regard, validating the focus topic of this thesis. As a result, novel techniques are required to be developed that do not make the simplifying assumptions used for SW coding in a WSN regime.

Chapter 3

Background and Preliminaries

An outline of the necessary background of the two key components of this research, DSC and CC, are presented. In addition, the novel mathematical framework and modelling used in this research are introduced, as well as the problem formulations for both the single and multiple group scenarios. This gives the reader the necessary tools to understand the contributions in the following chapters.

3.1 Introduction

The system consists of two primary components: CC and SW coding. The former ensures that the bandwidth of the server is minimised during peak hours, while the latter seeks to reduce the total amount of information needed to be sent by the server to the users by compressing the files beforehand. It is necessary to describe the background of each of these techniques, as it forms the backbone of the hybrid system. In doing so, it also becomes apparent where the focus of the research lies. Developing from this is the mathematical modelling and problem formulation of the system, which this research looks to solve. Some initial analysis of the problem reveals that it is complex and has not been addressed by literature to date. Section 3.2 outlines the necessary background knowledge for SW coding, while Section 3.3 does the same for CC. Then, Section 3.4 starts to develop the metrics for the hybrid SW/CC system. Sections 3.5 and 3.6 formulate the problem mathematically for the single group and multi-group cases respectively, and Section 3.7 concludes this chapter.

3.2 Distributed Source Coding

In their seminal paper, Slepian and Wolf [3] examined the efficacy of source coding, where the sources are correlated and independent and identically distributed (i.i.d.). Their most notable contribution is the proof that, for a system of two encoders and one decoder, the admissible rate regions are comparable, regardless of whether information sharing between encoders is allowed or not. When the encoders x and y are allowed to share information, the admissible rate region is bounded by $R_x + R_y \geq H(X, Y)$, where $H(\cdot)$ is the information entropy function. In this bound, R_i is the compression rate of encoder i in bits/symbol, meaning that a higher level of compression yields a lower compression rate. However, if the sharing of information is not allowed, Slepian and Wolf prove that the admissible rate region is given by:

$$R_x \geq H(X|Y) \tag{3.1}$$

$$R_y \geq H(Y|X) \tag{3.2}$$

$$R_x + R_y \geq H(X, Y) \tag{3.3}$$

Figures 3.1 and 3.2 show that, although the admissible rate region is greater in the former case, it only differs at the extremities. Surprisingly, the maximum achievable rates are the same. This paper has inspired an entire field dedicated to lossless DSC.

Cover [4] improved this contribution by first proving the SW rate region in a simpler manner. He then extends their work to a set of sources with an arbitrary size, defined in this work as $\mathbf{N} = \{X_1, X_2, \ldots, X_{|\mathbf{N}|}\}$, where each $X_i \in \mathbf{N}$ is an information source equivalent to the encoders x or y in [3]. For this expansion to $|\mathbf{N}|$ sources, it is necessary that each $X_i \in \mathbf{N}$ is ergodic. Of importance to this research is the fact that $|\mathbf{N}|$ sources corresponds to $2^{|\mathbf{N}|} - 1$ rate region equations, meaning the maximum admissible rate region can be accurately obtained for any number of sources. As an example, Cover gives the seven coding bounds for a system $\mathbf{N} = \{X_1, X_2, X_3\}$:

$$R_1 \geq H(X_1|X_2, X_3) \tag{3.4}$$
$$R_2 \geq H(X_2|X_1, X_3) \tag{3.5}$$
$$R_3 \geq H(X_3|X_1, X_2) \tag{3.6}$$
$$R_1 + R_2 \geq H(X_1, X_2|X_3) \tag{3.7}$$
$$R_2 + R_3 \geq H(X_2, X_3|X_1) \tag{3.8}$$
$$R_1 + R_3 \geq H(X_1, X_3|X_2) \tag{3.9}$$
$$R_1 + R_2 + R_3 \geq H(X_1, X_2, X_3) \tag{3.10}$$

Significantly, the bound on the total coding rate, and therefore the maximum compression of the system as a whole, is given as the joint entropy, which in general can be expanded to:

$$\sum_{i=1}^{|\mathbf{N}|} R_i \geq H(\mathbf{N}) = H(X_1, X_2, \ldots, X_{|\mathbf{N}|}) \tag{3.11}$$
$$= H(X_1) + \sum_{i=2}^{|\mathbf{N}|} H(X_i|X_{i-1}, X_{i-2}, \ldots, X_1) \tag{3.12}$$

Equation (3.12) is obtained by repeatedly using the chain rule for entropy.

This bound is only achievable if all sources are decoded jointly (although the coding is disjoint). Furthermore, the joint decode is computationally expensive and is governed by the number of sources involved in the coding scheme.

3.2.1 Symmetric and Asymmetric Rate Regions

Within the admissible rate region for SW coding, there are two distinct regions. Designing compression schemes for the corner points of the region (at points $[H(X|Y)$, $H(Y)]$ and $[H(X)$, $H(Y|X)]$ in Figure 3.2) correspond to the cases where one node sends full information and the other node sends fully compressed information. These schemes are known as asymmetric coding. They are the simplest schemes to implement, but result in an uneven compression distribution. Slepian and Wolf posited that symmetric coding (or coding for the entire region) could be achieved by time-sharing between the encoders. This is practically intractable, however, as

it requires synchronisation between nodes. Instead, literature has proposed two methods to perform systematic symmetric coding: syndrome and parity approaches.

Both approaches require the use of (n, k) ECCs, which are now briefly explained. Typically, an ECC maps a message of length k to a larger codeword of size n. In the case of a linear code, this is done by multiplying it by a specially constructed generator matrix $\mathbf{G}$, with dimensions $k \times n$. At the receiver, the codeword is converted to a syndrome of length $n - k$, which is used to detect and correct the errors. In the case of a linear ECC, the codeword is multiplied by a transposed parity check matrix $\mathbf{H}^T$, which is defined as the inverse of $\mathbf{G}$, to obtain the syndrome, which gives information as to whether an error occurred and where its location is. Thus, the function of an ECC is to increase the size of the message to increase the reliability of message transfer. The general approach to use an ECC as a compression function is to reverse this process. Although other ECCs can be used (such as LDPC codes), linear ECCs are easier to understand and explain.

In the syndrome approach, the encoding is performed on one of the sources by multiplying the original message (of size n) by the transposed parity check matrix ($\mathbf{H}^T$) to obtain the syndrome (of size $n - k$). The other source will send its full information (with length n). On the receiver's side, the syndrome is used to 'correct' the information from the source that was not compressed in order to decode the compressed message. Using this technique thus caters for the asymmetric part of the SW admissible rate region. This method is improved upon by Pradhan and Ramchandran in [8], by partitioning the matrix amongst the sources and adapting the decoder to perform joint decoding. This extends the technique to the symmetric region as well, providing the ability to balance the length of the codewords amongst the encoders.

In contrast to this, the parity approach takes a k-length message and multiplies it by the ECC to create an n-length codeword. Part of this codeword is then punctured, resulting in an $(n - k)$-length compressed message. The decoding is performed in a similar manner to the previous approach and also caters for the symmetric rate region.

Tan *et al.* [70] critically analyse and compare the above techniques. They find that the parity approach is more universal and is therefore simpler to implement. On the other hand, the syndrome method gives the greater compression that is closer to the theoretical bound. This comes at a cost of noise tolerance as the parity approach is able to losslessly decode in the presence of more noise than the syndrome approach.

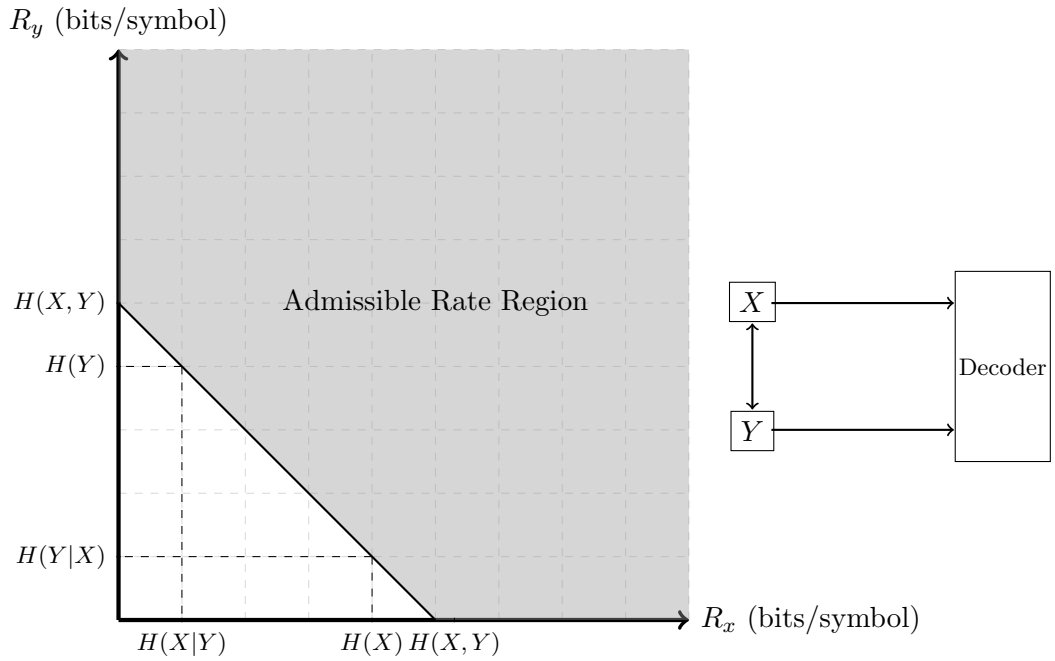


Figure 3.1: Slepian-Wolf admissible rate region for two correlated sources with information sharing [3]

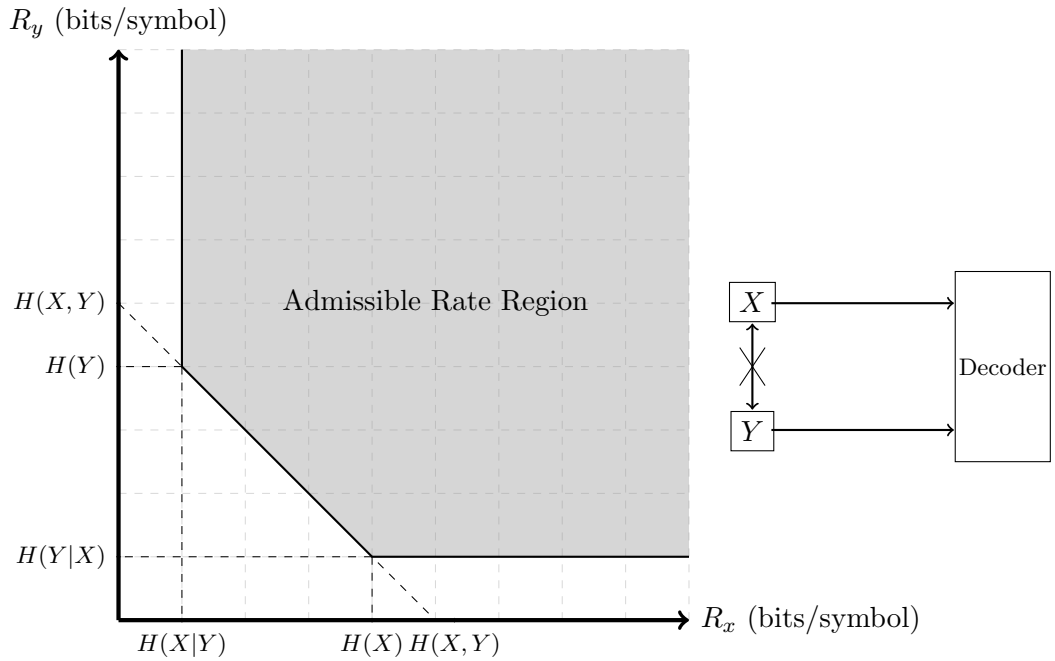


Figure 3.2: Slepian-Wolf admissible rate region for two correlated sources without information sharing [3]

3.2.2 Lossless and Lossy Coding

In traditional SW coding, and in both of the above approaches, there is no consideration of the rate distortion function. Similarly, the maximum compression is not given sufficient attention when taking into account the loss of information during transmission, meaning that the sources are expected to either be fully recovered without error or not at all. Wyner and Ziv [5] define $R^*(\mathcal{D})$ as the infimum of rates that allow for the average distortion level of the sources to not exceed the sum of the distortion ($\mathcal{D}$) and an arbitrarily small error (ϵ). Their results are used to reduce the theoretical bound by taking into account the effect of noise or quantisation error (both of which are used in place of distortion in literature). They thus modify SW coding to account for lossy decoding, which can be useful for certain data formats such as music.

Nevertheless, this research focuses on content delivery where data integrity is paramount. This makes the lossless case more suitable. In addition, the scheme involves multiple nodes, where it is beneficial to balance the coding amongst them. Thus, the syndrome approach is more appropriate.

3.3 Network Caching

In general, caching is a method used to reduce the strain on a network during peak hours by storing partial information at the user end. As in the SW case, the library of files is represented by the set $\mathbf{N}$. Each element of $\mathbf{N}$ is modelled as an information source X_i, $i \in \{1, 2, \ldots, |\mathbf{N}|\}$. Without loss of generality, each file X_i produces F binary symbols which are i.i.d. and ergodic. Accordingly, each file has an entropy of $H(X_i) = F$ bits, $\forall X_i \in \mathbf{N}$. Furthermore, this system comprises Z users, with each user having at its disposal a local cache of size M files, or MF bits. The contents of user i's cache is denoted as $\mathbf{Z}_i$.

The caching approach is divided into two phases: In the *placement* phase, the network fills the users' caches with files from its library during off-peak hours. Thus, the transmission rate is not constrained in this phase, only the size of memory available at the user end. This is usually done by storing content that the user regularly demands during peak hours, but it can also be done intelligently, by predicting the users' future demands. At the end of this phase, $\mathbf{Z}_i \subset \mathbf{N}$, $\forall i \in \{1, 2, \ldots, Z\}$.

In the *delivery* phase, the users reveal their demands, modelled here as a vector

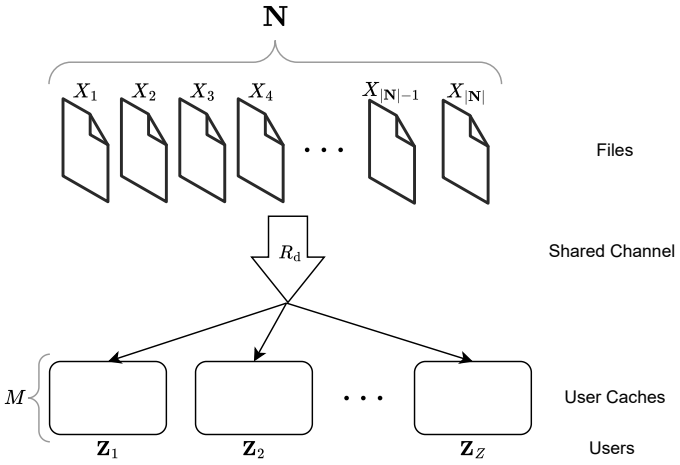


Figure 3.3: System model for the CC part of the hybrid content delivery system.

$\mathbf{d} := \{d_1, d_2, \ldots, d_Z\}$, where each d_i is an index corresponding to a request from user i for file X_{d_i}. The base station attempts to fulfil the file requests of the users by sending the missing information to each user, based on the placement in the caches in the previous phase. The challenge is to minimise the required delivery rate (R_{d}) for a given $\{\mathbf{N}, M, Z\}$ system, as shown in Figure 3.3.

Maddah-Ali and Niesen, in [52], were the first to discuss a novel caching method that aims to significantly increase the potency of caching. They achieve this by coding the data with respect to the data stored in the other caches. They show that, for a regular caching system, the system parameters $\{\mathbf{N}, M, Z\}$ result in a required delivery rate of:

$$R_{\mathrm{d}} = Z\left(1 - \frac{M}{|\mathbf{N}|}\right) \tag{3.13}$$

Their technique begins by defining the variable $q = {}^{MZ}/_{|\mathbf{N}|}$. Then, each of the files is split into $\binom{Z}{q}$ non-overlapping sub-files. Each user receives a specific arrangement of these sub-files, such that their cache consists of $|\mathbf{N}|\binom{Z-1}{q-1} = FM$ bits, fulfilling the maximum cache size constraint. During the delivery phase, the server checks the demand vector and XORs the missing pieces of information for each of the user's requests into a single codeword. The scheme is designed such that every user has the necessary sub-files to XOR with the sent codeword in order to recover the information they have not stored previously. In this way, the server can broadcast the missing information in a single transmission and deliver the required information to all users. Using this technique, they prove that the scheme can fulfil any demand vector, and the required rate is reduced to:

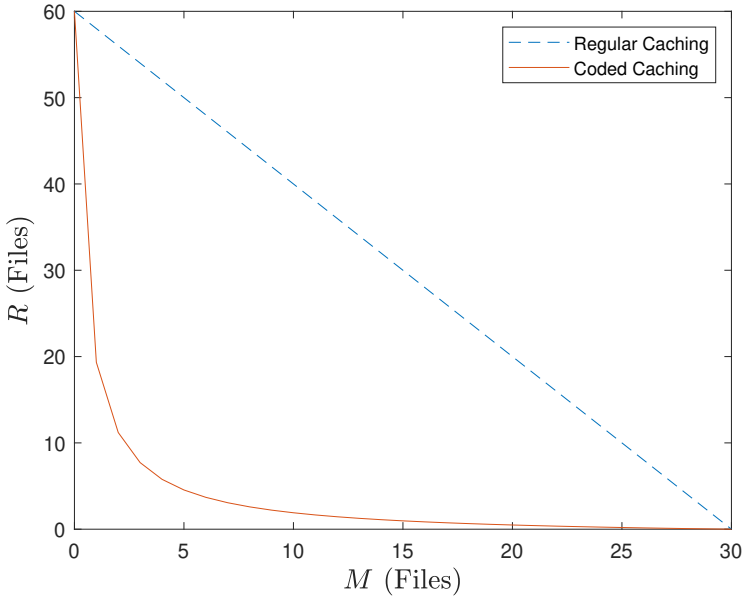


Figure 3.4: A comparison of the achievable rate for different cache sizes when using regular caching and CC for $Z = 60$ users, $|\mathbf{N}| = 30$ files

$$R_{\mathrm{d}} = \frac{Z(1 - \frac{M}{|\mathbf{N}|})}{1 + \frac{ZM}{|\mathbf{N}|}} \tag{3.14}$$

This is borne out in Figure 3.4, where the required rate (calculated in terms of the number of files) in the CC method decreases exponentially compared to the regular caching approach. This means that, even for small cache sizes, the CC method is able to dramatically reduce the required delivery rate.

3.4 Hybrid Slepian-Wolf Coded Caching System Model

Hassanzadeh *et al.* [63, 64] consider a system as described in Section 3.3. However, they now assume that the files are correlated to one another according to the distribution $p(x_1, x_2, \ldots, x_{|\mathbf{N}|})$. This allows for a hybrid system, where the files are first compressed using DSC coding. This becomes the library that is coded using CC and delivered to the user caches.

In this research, the correlation between the sources is modelled in terms of the Mutual Information Area (MIA) for each unique subset of $\mathbf{N}$. Each MIA is evaluated as the number of bits needed to represent the mutual information of the files involved, denoted by the function $I(\cdot)$. Thus, an MIA is related to the entropies of the different

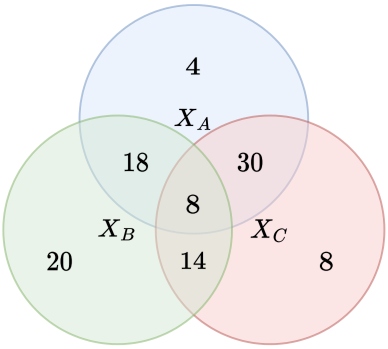


Figure 3.5: Example of an MIA correlation diagram for three sources

sources. There are a total of $2^{|\mathbf{N}|} - 1$ MIAs for the $|\mathbf{N}|$ files. Naturally, the entropy of all the files is given as $H(\mathbf{N}) \leq |\mathbf{N}|F$ bits. Nevertheless, the correlation values are merely statistical and do not necessarily describe the actual correlation between the contents of the files. For example, Figure 3.5 shows a configuration for three nodes, consisting of seven individual MIAs. To illustrate, the segment equal to 18 bits in Figure 3.5 corresponds to the MIA of $I(X_A; X_B|X_C)$. This means that 18 bits are necessary to represent the information common to both X_A and X_B when X_C is already known. So too, the entropy $H(X_i)$ can be derived by summing the MIAs associated with that source, namely the values contained within each circle in Figure 3.5. To show the potential coding gains, although the entropy of each source is $H(X_i) = 60$ bits, the correlation between them means that the number of bits required to represent all of them is given by the total entropy $H(X_A, X_B, X_C) = 102$ bits instead of 180.

In [66], the objective of the caching scheme is evaluated according to the minimum multicast rate necessary to fulfil the worst demand:

$$R_{d_{\min}} = \max_{\mathbf{d} \in \mathbf{D}} \frac{\mathbb{E}[\ell(Y_{\mathbf{d}})]}{F}, \tag{3.15}$$

where $\ell(\cdot)$ is the length of the broadcast codeword Y for demand $\mathbf{d}$, $\mathbf{D}$ is the set of all possible demand vectors and $\mathbb{E}[\cdot]$ is the expected value. Another evaluation metric is the average multicast rate, defined as:

$$\bar{R}_d = \frac{\mathbb{E}[\ell(Y_{\mathbf{D}})]}{F} \tag{3.16}$$

It is based on the average broadcast codeword length over all demands.

Since the files are correlated, the server is able to use SW coding to compress the files that are stored in the caches, although the exact contents of the files are unknown. This has the effect of storing more content from the files in the users' caches, meaning that $\ell(Y_{\mathbf{d}})$ is reduced when compared to the DSC and CC approaches individually. They note, however, that the complexity of the DSC becomes prohibitive with an increase in the number of files. As a result, they only consider a maximum of three files in their paper.

Thus, the general optimisation problem considered in this work is to decrease the complexity by clustering sources in the scheme while minimising the impact on the total achievable compression. This is considered using two different approaches, which are discussed in the next two sections.

3.5 Single Group Clustering Model

The first system that this research considers is shown in Figure 3.6, which is adapted from [66]. In that paper, all the files are compressed before they are transmitted to the caches. The goal of our proposed system is to partition the library into two groups, one of which will be compressed according to a DSC method such as SW coding with MP [9], denoted as $\mathbf{N}_c$. The other group will remain uncoded and is represented by $\mathbf{N}_u$. The sets are disjoint, meaning that $\mathbf{N}_c \cup \mathbf{N}_u = \mathbf{N}$ and $\mathbf{N}_c \cap \mathbf{N}_u = \emptyset$. Consequently, if $|\mathbf{N}_u| = \gamma$ files, then $|\mathbf{N}_c| = |\mathbf{N}| - \gamma$ files. This new hybrid library $\{\mathbf{N}_c \cup \mathbf{N}_u\}$ can then be optimally distributed amongst the Z users using a CC encoder optimised for files of unequal lengths (such as [53]), since the files in $\mathbf{N}_c$ have their size reduced from the DSC step. Unlike [65], which considers that files are divided into a finite number of blocks, this system allows for compressed files of any size. As depicted in Figure 3.6, the main thrust of this work is finding the clustering algorithm used to split the files into two disjoint sub-libraries, which simply requires choosing a single group, as the other follows as a matter of course. The optimisation problem is now defined mathematically.

The main goal of the hybrid system is to reduce the complexity of the encoding and decoding of the compression scheme without sacrificing too much of the compression rate for the system as a whole. However, there are two approaches to achieve this. In the first, the complexity is fixed by setting the maximum number of nodes to compress. Hence, let $\gamma = |\mathbf{N}_u|$ be the minimum number of nodes that should not be compressed, at which the number of compressed nodes $|\mathbf{N}_c| = |\mathbf{N}| - \gamma$ achieves a reasonable decoding complexity. Then, the objective is to choose a subset $\mathbf{N}_u$ from

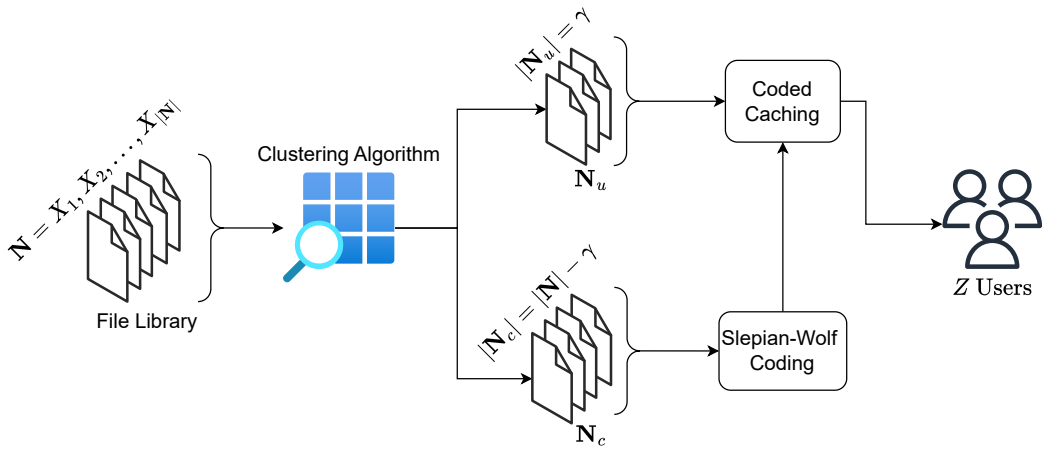


Figure 3.6: System model of the single group clustering objective

$\mathbf{N}$ that maximises the reduction in information for the coded sources $\mathbf{N} \setminus \mathbf{N}_u = \mathbf{N}_c$. This is formulated as follows:

$$\mathbf{N}_u^* = \arg \min_{\mathbf{N}_u \subset \mathbf{N}} H(\mathbf{N}) - H(\mathbf{N}_u) \tag{3.17}$$

$$\text{s.t.} \quad |\mathbf{N}_u| = \gamma \tag{3.18}$$

$H(\mathbf{N})$ and $H(\mathbf{N}_u)$ are the total entropies of the sources in the sets $\mathbf{N}$ and $\mathbf{N}_u$ respectively.

Another approach is to bound the entropy of the compressed group of sources, effectively setting the compression performance in this group. Then, the objective is to find the maximum number of sources to put in $\mathbf{N}_c$ without exceeding the entropy bound. As a result, let ζ be an entropy value in the range $0 < \zeta < H(\mathbf{N})$. The goal is to minimise the number of sources in $\mathbf{N}_u$ while keeping the entropy of the compressed group to $H(\mathbf{N}_c) \leq \zeta$. In this instance, the objective function is defined as:

$$\mathbf{N}_u^* = \arg \min_{\mathbf{N}_u \subset \mathbf{N}} |\mathbf{N}_u| \tag{3.19}$$

$$\text{s.t.} \quad H(\mathbf{N}_c) \leq \zeta \tag{3.20}$$

The focus of the optimisation problems is on $\mathbf{N}_u$ instead of $\mathbf{N}_c$. This is so, since a fundamental principle of entropy is that conditioning can never increase it. Thus,

approaching the problem by maximising the entropy of $\mathbf{N}_u$ allows the algorithms to focus on the upper bound. As will become apparent in later chapters, this allows for better characterisation of the entropy calculations. So too, maximising the entropy of $\mathbf{N}_u$ will automatically minimise the entropy of $\mathbf{N}_c$, based on their definitions above. Consequently, this approach still looks at solving the general problem of maintaining performance while reducing complexity in the hybrid SW/CC system. Clearly, the number of possible combinations in these optimisation problems is $\binom{|\mathbf{N}|}{\gamma}$, which has a complexity in the order of $O(2^{|\mathbf{N}|})$.

Traditional clustering algorithms, such as hierarchical clustering and the k-means algorithm, would not be effective solutions for this clustering problem, since they are not adapted for entropy calculations. This is because the relative entropy changes depending on the sources that it is conditioned on, meaning that the entropy continuously changes, and depends on the sources placed within a group.

Instead, the optimisation problems in (3.17) and (3.19) are similar to the Minimum Weight Set Covering (MWSC) problem, where each subset has a weight attached to it and the goal is to choose the fewest subsets that covers all members of the set while minimising the total weight. This problem is known to be NP-hard [71].

Another similar optimisation problem is the 0-1 Knapsack (0-1K) problem, in which a set of members each contain a weight and value. The objective is to choose the best performing subset of members that maximises the total value of the members while not exceeding a certain total weight. This problem too is NP-hard [72].

In the MWSC and 0-1K problems, the number of valid combinations in the search space for a set of size $|\mathbf{N}|$ is $2^{|\mathbf{N}|}$, whereas in the single group clustering problem it is $\binom{|\mathbf{N}|}{\gamma}$. Nevertheless, the weights (entropies) of each subset are not known beforehand and must be calculated, unlike the MWSC and 0-1K problems. Furthermore, the weight calculation must be summed over $2^{\gamma} - 1$ MIAs. As a result, when γ is in the region of $|\mathbf{N}|/2$, or if γ is large, the optimisation problems in (3.17) and (3.19) become NP-hard as well.

3.6 Multi-Group Clustering Model

The second approach generalises the single group clustering problem to t groups, each of size γ. Although it could be argued that the next progression from the single group case would be to have a mixture of coded and uncoded groups, this would

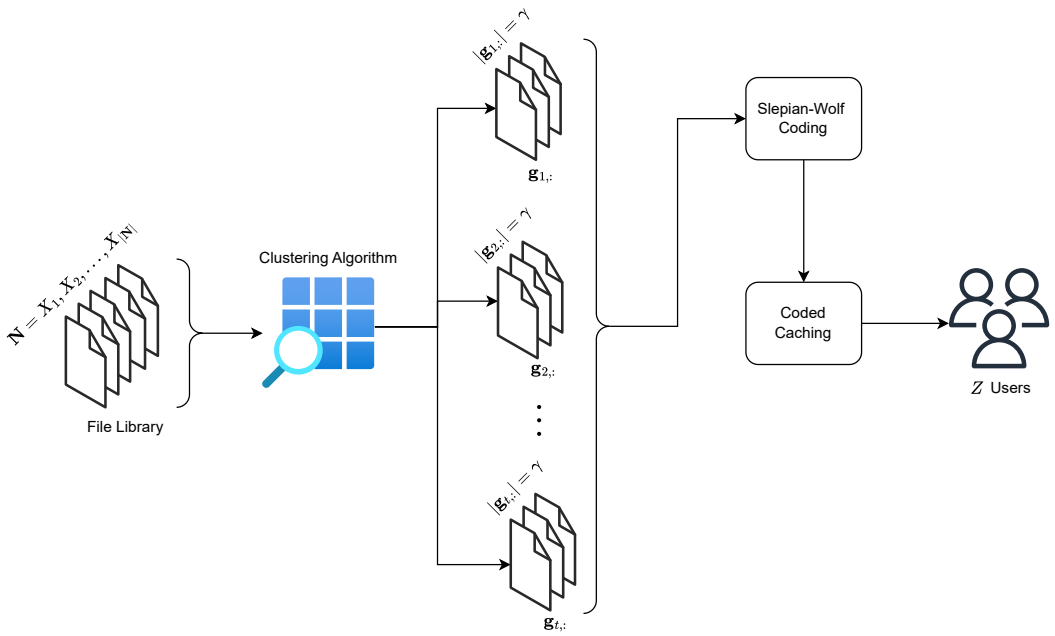


Figure 3.7: System model of the multi-group clustering objective

mean that the SW coding is not utilised to the fullest extent, since an assumption in this research is that the files in the system have a strong correlation between them. So too, the group size could be chosen so as to limit the complexity of the SW coding, meaning that using SW coding for all of the groups would not significantly increase the complexity of the system as a whole. As a result, each of these groups in this problem will be coded individually using SW coding, as shown in Figure 3.7.

Without loss of generality, assume that γ divides $|\mathbf{N}|$ exactly and that $t = {}^{|\mathbf{N}|}/_{\gamma}$. Let $\mathcal{G}$ be a $t \times \gamma$ matrix that represents a single clustering of files, with each element $g_{i,j} \in \mathcal{G}$ representing a single file in $\mathbf{N}$. Let $f^{\mathcal{X}} : g_{i,j} \to X_k$ be a function that maps a group element $g_{i,j}$ to a file $X_k \in \mathbf{N}$. A grouping $\mathcal{G}$ is structured such that files in the same group appear in the same row. It is thus of the form:

$$\mathcal{G} = \begin{pmatrix} g_{1,1} & \cdots & g_{1,\gamma} \\ \vdots & \ddots & \vdots \\ g_{t,1} & \cdots & g_{t,\gamma} \end{pmatrix} \tag{3.21}$$

For a single grouping $\mathcal{G}$, it is required that every group element is mapped to a valid file in the system, meaning that $\forall g_{i,j} \in \mathcal{G},\ f^{\mathcal{X}}(g_{i,j}) \in \mathbf{N}$. So too, the mapping of all the group elements $f^{\mathcal{X}}(\mathcal{G})$ is equivalent to the set $\mathbf{N}$.

To simplify the indexing of a group, let $\mathbf{g}_{i,:} \subset \mathcal{G}$ be equivalent to the ith row in $\mathcal{G}$,

representing the nodes in the same group. It follows that $\mathbf{g}_{i,:} \coloneqq \{g_{i,1}, g_{i,2}, \ldots, g_{i,\gamma}\}$, with $|\mathbf{g}_{i,:}| = \gamma$ and $\mathcal{G} = \bigcup_{i=1}^{t} \mathbf{g}_{i,:}$. So too, define the ith column set of grouping $\mathcal{G}$ as $\mathbf{g}_{:,i} \coloneqq \{g_{1,i}, g_{2,i}, \ldots, g_{t,i}\}$, meaning that $\mathbf{g}_{:,i} \subset \mathcal{G}$ and $|\mathbf{g}_{:,i}| = t$. In this case, each element in the column $\mathbf{g}_{:,i}$ is from a different group, and $\mathcal{G} = \bigcup_{i=1}^{\gamma} \mathbf{g}_{:,i}$.

It is required that all rows of $\mathcal{G}$ are disjoint, implying that $\mathbf{g}_{i,:} \cap \mathbf{g}_{j,:} = \emptyset$ when the group rows are different, i.e. $\forall \mathbf{g}_{i,:}, \mathbf{g}_{j,:} \in \mathcal{G},\ i \neq j$. The same can be said about the columns, namely that $\mathbf{g}_{:,i} \cap \mathbf{g}_{:,j} = \emptyset$ when $\forall \mathbf{g}_{:,i}, \mathbf{g}_{:,j} \in \mathcal{G},\ i \neq j$. In addition, let $\mathbf{P}_{g_{i,j}} \coloneqq \{g_{i,j}, g_{i,j-1}, \ldots, g_{i,1}\}$ be a path, defined using the given group element as the starting point, and terminating at the first group element of that row.

The multi-group clustering problem looks at dividing the $|\mathbf{N}|$ nodes into t groups, each of size γ (where the value of γ is chosen to minimise the complexity of the SW coding). The goal of the optimisation problem is to ensure that the total entropy is minimised. Given a grouping $\mathcal{G}$ as defined above, the entropy is determined as:

$$H(\mathcal{G}) = \sum_{i=1}^{t} H(\mathbf{g}_{i,:}), \qquad \forall \mathbf{g}_{i,:} \in \mathcal{G} \tag{3.22}$$

$$= \sum_{i=1}^{t} \sum_{j=1}^{\gamma} H(g_{i,j} | \mathbf{P}_{g_{i,j-1}}), \qquad \forall g_{i,j} \in \mathcal{G} \tag{3.23}$$

Let $\mathcal{C}_t^{|\mathbf{N}|}$ be a set of all unique and valid groupings for the set $\mathbf{N}$ with t groups. Then, the objective for this optimisation is to find the grouping $\mathcal{G} \in \mathcal{C}_t^{|\mathbf{N}|}$ that minimises the total entropy. More formally:

$$\mathcal{G}^* = \arg \min_{\mathcal{G} \in \mathcal{C}_t^{|\mathbf{N}|}} H(\mathcal{G}) \tag{3.24}$$

Unlike the single group clustering problem, the number of unique and valid possibilities is not readily apparent and is determined in the following lemma.

Lemma 1. Let $\mathcal{C}_t^{|\mathbf{N}|}$ be a set of all possible unique combinations of valid groupings $\mathcal{G}_i \in \mathcal{C}_t^{|\mathbf{N}|}$. Then, the number of unique combinations for a system with $|\mathbf{N}|$ files clustered into t groups is given by:

$$|\mathcal{C}_t^{|\mathbf{N}|}| = \frac{|\mathbf{N}|!}{t!\,(\gamma!)^t} \tag{3.25}$$

Proof. The following properties are observed based on the problem constraints: every one of the t groups requires that we draw γ nodes. Every successive group draw reduces the total number of nodes by γ. Thus, the total number of combinations is given by:

$$|\mathcal{C}_t^{|\mathbf{N}|}| = \frac{1}{t!} \underbrace{\binom{|\mathbf{N}|}{\gamma}\binom{|\mathbf{N}|-\gamma}{\gamma}\binom{|\mathbf{N}|-2\gamma}{\gamma}\cdots\binom{|\mathbf{N}|-(t-2)\gamma}{\gamma}\binom{|\mathbf{N}|-(t-1)\gamma}{\gamma}}_{t\,\text{terms}} \tag{3.26}$$

$$= \frac{1}{t!}\frac{|\mathbf{N}|!}{\gamma!\,\cancel{(|\mathbf{N}|-\gamma)!}}\frac{\cancel{(|\mathbf{N}|-\gamma)!}}{\gamma!\,\cancel{(|\mathbf{N}|-2\gamma)!}}\frac{\cancel{(|\mathbf{N}|-2\gamma)!}}{\gamma!\,\cancel{(|\mathbf{N}|-3\gamma)!}}\cdots\frac{\cancel{2\gamma!}}{\gamma!\,\cancel{(\gamma)!}}\frac{\cancel{\gamma!}}{\gamma!\,\underbrace{(\gamma-\gamma)!}_{=1}} \tag{3.27}$$

$$= \frac{|\mathbf{N}|!}{t!\,(\gamma!)^t} \tag{3.28}$$

□

Although the last term in (3.26) is equal to 1, it is needed to cancel out a γ in the previous term. The reason for the term $1/t!$ is to take the combinations of the groups into account. For example, without this term, the groupings $\mathcal{G}_i = \{\{X_1, X_2, X_3\}, \{X_4, X_5, X_6\}, \{X_7, X_8, X_9\}\}$ and $\mathcal{G}_j = \{\{X_4, X_5, X_6\}, \{X_7, X_8, X_9\}, \{X_1, X_2, X_3\}\}$ are treated as unique.

It should be noted that the same result is achieved by reducing:

$$\begin{aligned}|\mathcal{C}_t^{|\mathbf{N}|}| = &\binom{|\mathbf{N}|-1}{\gamma-1}\binom{|\mathbf{N}|-\gamma-1}{\gamma-1}\binom{|\mathbf{N}|-2\gamma-1}{\gamma-1}\cdots \\ &\binom{|\mathbf{N}|-(t-2)\gamma-1}{\gamma-1}\binom{|\mathbf{N}|-(t-1)\gamma-1}{\gamma-1}\end{aligned} \tag{3.29}$$

Equation (3.25) is difficult to analyse, especially compared to the single group clustering problem. Nevertheless, it is clear that $|\mathcal{C}_t^{|\mathbf{N}|}| = 1$ when $t = 1$ or $t = |\mathbf{N}|$. Additionally, the comparison of the number of combinations in the multi-group clustering case with the single group case for the same values of t and γ is determined by the following equation:

$$\frac{|\mathcal{C}_t^{|\mathbf{N}|}|}{\binom{|\mathbf{N}|}{\gamma}} = \frac{|\mathbf{N}|!}{t!\,(\gamma!)^t}\frac{\gamma!(|\mathbf{N}|-\gamma)!}{|\mathbf{N}|!} \tag{3.30}$$

$$= \frac{(|\mathbf{N}|-\gamma)!}{t!(\gamma!)^{t-1}} \tag{3.31}$$

Interestingly, if $\gamma = |\mathbf{N}|/2 \implies t = 2$ then the above equation reduces to:

Table 3.1: Comparing the number of possible valid groupings for the single group and multi-group clustering cases for $|\mathbf{N}| = 48$ files.

t	γ	$\binom{\lvert\mathbf{N}\rvert}{\gamma}$	$\lvert\mathcal{C}_t^{\lvert\mathbf{N}\rvert}\rvert$
1	48	1	1
2	24	3.2248×10^{13}	1.6124×10^{13}
3	16	2.2548×10^{12}	2.2589×10^{20}
4	12	6.9669×10^{10}	9.8254×10^{24}
6	8	377348994	4.0129×10^{30}
8	6	12271512	4.2631×10^{33}
12	4	194580	7.0964×10^{35}
16	3	17296	2.1031×10^{35}
24	2	1128	1.1926×10^{30}
48	1	48	1

$$\begin{aligned}\frac{|\mathcal{C}_t^{|\mathbf{N}|}|}{\binom{|\mathbf{N}|}{\gamma}} &= \frac{\gamma!}{t!(\gamma!)^{t-1}} \\ &= \frac{1}{t!(\gamma!)^{t-2}} \\ &= \frac{1}{2(\gamma!)^{2-2}} = 0.5 \end{aligned} \tag{3.32}$$

This means that the single group problem has double the number of possible solutions for this problem setup. It is also the setup in which the single group problem has the most number of combinations, since $\gamma = {}^{|\mathbf{N}|}/_2$ maximises $\binom{|\mathbf{N}|}{\gamma}$. It has been empirically determined, however, that the multi-group problem is generally more complex. Table 3.1 shows the number of possible combinations for different t and γ combinations, where $|\mathbf{N}| = 48$ files. The case of $\gamma = 24$ maximises the number of possible combinations for the single group case and is double the number of possible combinations in the multi-group case, as predicted by (3.32). However, in every other problem setup except for $t = 1$ and $t = |\mathbf{N}|$, the multi-group problem is more complex than the single group case by many orders of magnitude. So too, the $\gamma = 24$ case is the one with the lowest number of possible combinations in the multi-group scenario.

3.7 Conclusion

The hybrid SW/CC system leverages two key information-theoretic concepts involving source and network coding. SW coding is able to reduce the number of bits needed to represent sources based on their inherent correlation and the technique has been refined in literature to allow for flexible schemes in terms of rates and correlation structures, as well as the ECC method used. So too, CC allows for impressive rate reductions in delivering content to users by intelligently and programmatically storing data at the user end during off-peak hours. A strong theoretical framework and evaluation metrics have been developed to incorporate the two techniques. However, the complexity of the system has not been adequately dealt with by literature.

Consequently, this chapter has presented two optimisation problems that address these issues. The first involves choosing a subset of files to encode while the other is left intact. The second looks at dividing the files into a number of different groups of the same size. The mathematical representations and structures for these problems have been thoroughly formulated. So too, the complexity of the problems are defined and explained with rigorous mathematical derivations. It is shown that the single group case in general has fewer possible combinations for a given problem setup and is thus a more suitable problem to tackle first, while the multi-group case has more potential for realising the gains of the hybrid SW/CC approach.

Chapter 4

Single Group Entropy Maximisation Approach

An initial attempt at solving the single group clustering problem for a hybrid SW/CC system is presented. Two sub-optimal yet well-performing approaches are proposed. The first is an iterative, greedy method while the other is a meta-heuristic version based on the GA. When compared to an exhaustive search using a system of 18 files, the proposed solutions find clusters that fall above the 90th percentile of all possible solutions on average with significantly lower time complexity. Of note is that the iterative method produces results within one percentile of the meta-heuristic approach yet it finds a solution 2.31 times faster. The iterative approach has an additional benefit, in that it is able to predict the relative gains when adding more files to a compression group. It is thus able to terminate prematurely if the estimated gains are less than a chosen threshold.

4.1 Introduction

The previous chapter has shown the single group clustering problem to be NP-hard, meaning it is difficult to solve in an iteration-free manner. So too, this means that it is highly unlikely to find an algorithm that runs in polynomial time while guaranteeing optimality. Consequently, it is necessary to break the problem down using an information-theoretic view in order to develop an algorithm that is performant in this problem setup.

As shown in Figure 3.6, the goal of the algorithm is to divide the files into two groups,

$\mathbf{N}_c$ and $\mathbf{N}_u$, where one is to be coded while the other is left uncoded. Although, at first glance, this appears to be dividing the files into two groups, only one group needs to be created, since the other is defined as a matter of course. As a result, the algorithms attempting to solve this problem only need to find a single grouping. Additionally, the files can already be viewed as being in a single group $\mathbf{N}$ at the start. Thus, all that is required is to remove the correct files from this group in order to arrive at a solution. Following from all of the above, this chapter mainly focuses on how to find the most suitable files to remove, such that the resulting division of files yields the best performance. As a benchmark, the GA is proposed as a potential solution, but it needs to be adapted so that it satisfies the specific constraints of the single group optimisation problem. So too, a demonstrative example is developed to show how the concepts in Chapter 3 are combined and how the clustering achieves its goal of improving the performance of the hybrid SW/CC system while reducing the complexity.

This chapter is organised as follows: Sections 4.2 and 4.3 outline the two approaches and are the main contributions of this chapter. Section 4.4 analyses the complexity and optimality of the various solutions and Section 4.5 presents and compares the simulation results. Finally, Section 4.6 concludes this chapter.

4.2 Greedy Iterative Selection Procedure

The basic approach to solving the optimisation problem in (3.17) is to iteratively select the most suitable sources until a subset of size γ is reached. Further examination of the expansion of the total entropy in (3.12) reveals that the entropy of the set can be expressed in individual terms, where each term refers to only one source conditioned on other sources. This means that choosing the source X_i for each term, such that $H(X_i|X_{i-1}, X_{i-2}, \ldots, X_1) \geq H(X_j|X_{i-1}, X_{i-2}, \ldots, X_1)$, $\forall j \in \{i+1, \ldots, |\mathbf{N}|\}$ should guarantee a maximisation of the entropy expression at that point. The only exception is the first term, since, as mentioned in Section 3.3, the entropies of all sources are set to be the same. In this case, it is necessary to choose the source based on a different criterion. Notice that the final term in (3.12) is $H(X_{|\mathbf{N}|}|X_{|\mathbf{N}|-1}, X_{|\mathbf{N}|-2}, \ldots, X_1)$, which is equivalent to the MIA $I(X_{|\mathbf{N}|}|X_{|\mathbf{N}|-1}; X_{|\mathbf{N}|-2}; \cdots; X_1)$. Thus, choosing the source that maximises the entropy term is equivalent to selecting the node that is least correlated with the other sources.

These observations imply that, by continually selecting the largest term from the available set of files to add to $\mathbf{N}_u$, the entropy of the set $\mathbf{N}_c = \mathbf{N} \setminus \mathbf{N}_u$ is minimised.

As a result, the following selection procedure is proposed:

Procedure 1 (source selection). Let the sources in $\mathbf{N}$ be drawn according to the following conventions: Choose X_1 such that $H(X_1|\mathbf{N} \setminus X_1) \geq H(X_i|\mathbf{N} \setminus X_i)\ \forall X_i \in \mathbf{N} \setminus X_1$. Then, for $k \in \{2, \ldots, |\mathbf{N}|\}$, choose source X_k such that

$$H(X_k|X_{k-1}, X_{k-2}, \ldots, X_1) \geq H(X_i|X_{k-1}, X_{k-2}, \ldots, X_1) \tag{4.1}$$

$$\forall X_i \in \mathbf{N} \setminus \{X_1, X_2, \ldots, X_k\} \tag{4.2}$$

It is possible for there to be multiple options for X_k, in which case X_k should be chosen arbitrarily.

Lemma 2. If the sources are organised as outlined in Procedure 1, then it is guaranteed that

$$H(X_k|X_{k-1}, X_{k-2}, \ldots, X_1) \geq H(X_{k+1}|X_k, X_{k-1}, \ldots, X_1) \tag{4.3}$$

Proof. From (4.1) it is known that

$$H(X_k|X_{k-1}, X_{k-2}, \ldots, X_1) \geq H(X_{k+1}|X_{k-1}, X_{k-2}, \ldots, X_1) \tag{4.4}$$

$$\geq H(X_{k+1}|X_k, X_{k-1}, \ldots, X_1) \tag{4.5}$$

This is based on the fact that conditioning reduces entropy. □

Using this selection procedure thus ensures that the terms in the expansion are in descending order.

Theorem 1. Let the sources be indexed according to Procedure 1. If, at any point in the selection process, $H(X_k|X_{k-1}, X_{k-2}, \ldots, X_1) \geq \zeta \geq H(X_{k+1}|X_k, X_{k-1}, \ldots, X_1)$ for some threshold ζ, then

$$H(\mathbf{N}) - H(X_1, X_2, \ldots, X_k) \leq (|\mathbf{N}| - k)\zeta \tag{4.6}$$

Proof. Since $\zeta \geq H(X_{k+1}|X_k, X_{k-1}, \ldots, X_1)$, then from Lemma 2

$$\zeta \geq H(X_{k+1}|X_k, X_{k-1}, \ldots, X_1) \tag{4.7}$$

$$\geq H(X_{k+2}|X_{k+1}, X_k, \ldots, X_1) \tag{4.8}$$

$$\vdots \tag{4.9}$$

$$\geq H(X_{|\mathbf{N}|}|X_{|\mathbf{N}|-1}, X_{|\mathbf{N}|-2}, \ldots, X_1) \tag{4.10}$$

Together with this, we have

$$H(\mathbf{N}) - H(X_1, X_2, \ldots, X_k) = \sum_{i=1}^{|\mathbf{N}|-k} H(X_{k+i}|X_{k+i-1}, X_{k+i-2}, \ldots, X_1) \tag{4.11}$$

Finally,

$$(|\mathbf{N}| - k)\zeta \geq \sum_{i=1}^{|\mathbf{N}|-k} H(X_{k+i}|X_{k+i-1}, X_{k+i-2}, \ldots, X_1) \tag{4.12}$$

□

Theorem 1 can thus be used as a stopping condition to satisfy (3.19). If, while choosing the sources, the entropy of the selected source is less than ζ, then the rest of the sources' entropy contribution can be bounded and the selection process will terminate with k sources.

The above derivations are combined to produce the Greedy Iterative Single Group Entropy Minimisation (GISGEM) selection procedure, outlined in Algorithm 1. It is able to generate a potential solution to the optimisation problems in (3.17) and (3.19) by maximising the entropy of the source removed at each step.

Algorithm 1 Greedy Iterative Single Group Entropy Minimisation Procedure

$\mathbf{N}_c \leftarrow \mathbf{N}$
$\mathbf{N}_u \leftarrow \emptyset$
$i \leftarrow 1$
sort $\mathbf{N}_c$ s.t. $H(X_i|\mathbf{N} \setminus X_i) \geq H(X_{i+1}|\mathbf{N} \setminus X_{i+1}) \geq \cdots \geq H(X_{|\mathbf{N}_c|}|\mathbf{N} \setminus X_{|\mathbf{N}_c|})$
$\mathbf{N}_u \leftarrow \mathbf{N}_u \cup X_i$
$\mathbf{N}_c \leftarrow \mathbf{N}_c \setminus X_i$
while $|\mathbf{N}_u| \leq \gamma$ **do**
    $i \leftarrow i + 1$
    sort $\mathbf{N}_c$ s.t. $H(X_i|\mathbf{N}_u) \geq H(X_{i+1}|\mathbf{N}_u) \geq \cdots \geq H(X_{|\mathbf{N}_c|}|\mathbf{N}_u)$
    if $H(X_i|\mathbf{N}_u) < \zeta$ **then**
        Break
    end if
    $\mathbf{N}_u \leftarrow \mathbf{N}_u \cup X_i$
    $\mathbf{N}_c \leftarrow \mathbf{N}_c \setminus X_i$
end while
return $\mathbf{N}_c$, $\mathbf{N}_u$

4.3 Combinatorial Meta-Heuristic Approach

The GA is a meta-heuristic algorithm whose efficacy in combinatorial problems is well known [73]. Every valid combination is represented by a genome sequence,

which sets the variables in the optimisation problem. For the single group entropy minimisation problem, the genome structure is updated as follows: the position in the genome represents the source in the set $\mathbf{N}$ and can take on the Boolean values 0 and 1 depending on whether the source is in the set $\mathbf{N}_u$ or not. This means that the genomes are of length $|\mathbf{N}|$ with the constraint that the Hamming weight must be equal to γ.

A population of random genomes is created, with rules defined for populating the next generation based on genome crossover and mutation. The crossover function selects features from both parents, based on a random crossover point. In our approach, the crossover functions need to produce child genomes that conform to the weight constraint mentioned above. As a result, the crossover function is changed to the following: given two genome sets $\mathbf{G}_1$ and $\mathbf{G}_2$ (defined as the sources in $\mathbf{N}_u$ shown by genome sequences $\mathbf{g}_1$ and $\mathbf{g}_2$), the child genome set $\mathbf{G}_3$ is constructed by randomly choosing γ sources from $\mathbf{G}_1 \cup \mathbf{G}_2$.

The mutation function randomly flips a bit in the genome, ensuring that the genome pool does not become too small. Here, the mutation is updated to swap a random number of value pairs in the genome, since simply flipping a single bit could violate the Hamming weight constraint. The rest of the algorithm, such as parent selection, does not require any changes. A summary of this approach can be found in Algorithm 2.

4.4 Theoretical Analysis

The methods outlined above are now analysed in terms of the algorithm complexity and whether they produce objectively optimal solutions as compared to a Brute Force (BF) search.

4.4.1 Complexity

The BF approach to solving the entropy minimisation optimisation requires that every combination of $\mathbf{N}_u$ is determined and evaluated and the grouping that provides the least impact to the compression gains is chosen. This means that the complexity is in the order of $O(2^{|\mathbf{N}|})$. In contrast to this, the GISGEM selection procedure has a much lower complexity. In the ith step, there are $|\mathbf{N}| - i - 1$ terms that are calculated. Thus, the total number of calculations required for γ steps is:

Algorithm 2 Genetic Algorithm Approach

```
P ← random population of genomes
Calculate and rank the performance of each genome in P
while not converged do
    Choose α_c|P| random pairs of parent genomes for crossover
    for each parent genome pair {g_1, g_2} do
        G_child ← γ random sources from G_1 ∪ G_2          ▷ Crossover
    end for
    Choose α_m|P| random genomes for mutation
    for each genome g_i do
        g_child ← g_i
        swap random number of pairs in g_child             ▷ Mutation
    end for
    P_new ← |P|(1 − α_c − α_m) best genomes from P
    P_new∪ crossover children
    P_new∪ mutation children
    P ← P_new
    Calculate and rank each genome in P
end while
```

$$\gamma|\mathbf{N}| - \sum_{i=1}^{\gamma-1} i = \gamma(|\mathbf{N}| - 1/2(\gamma - 1)) \tag{4.13}$$

There is the additional complexity of ranking the terms in each step, however this complexity is negligible when compared to calculating the entropy. As a result, the most complex operation of calculating the entropy is performed close to $\gamma|\mathbf{N}|$ times, resulting in a complexity in the order of $O(|\mathbf{N}|)$, meaning it runs in polynomial time. The complexity of the GA varies, and is dependent on the size of the population, as well as the number of generations required until the algorithm converges.

In literature, the authors in [33] propose a greedy algorithm that requires ranking the power set of $\mathbf{N}$. However, their algorithm finds the minimum entropy disjoint grouping for *all* sources. Nevertheless, when determining the best grouping of size γ, this method will have a complexity in the same order as the BF method. Thus, the GISGEM selection procedure dramatically reduces the complexity as compared to the brute-force and literature methods when γ is in the region of $|\mathbf{N}|/2$.

4.4.2 Optimality

Although less complex than the BF approach, the following lemmas show that using GISGEM does not provide an optimal result with regards to the optimisations in (3.17) and (3.19) respectively.

Lemma 3. The GISGEM selection procedure is not optimal in terms of the optimisation in (3.17).

Proof. Since the GISGEM selection procedure is iterative, it begins with $\mathbf{N}_u = \emptyset$ and adds a single source at a time. Without any stopping conditions, this will result in $\mathbf{N}_u = \mathbf{N}$. However, every possible order of choosing sources to put into $\mathbf{N}_u$ must follow these same start and end states. Thus, it is impossible that a single selection order will produce an optimal result for any arbitrary $0 \leq \gamma \leq |\mathbf{N}|$, since it cannot guarantee that it will be optimal at every point. □

Lemma 4. GISGEM is not optimal in terms of the optimisation in (3.19).

Proof. Theorem 1 provides a stopping condition for the GISGEM algorithm. Nevertheless, as shown in Lemma 3, this result is not necessarily optimal. Thus, it is possible that another selection of sources has a more optimal $\mathbf{N}_u$ and, even after removing one or more of the worst-performing sources, would still perform better than GISGEM. □

Thus, theoretically, the GISGEM algorithm should produce sub-optimal results but with less time complexity than the BF and GA methods. The following section details the numerical results obtained for the different approaches.

4.5 Practical Testing and Results

In the following subsection, an illustrative example is presented, comparing the performance of the different coding schemes, namely the CC [52] and hybrid SW/CC [66] approaches from literature, and the hybrid SW/CC method with reduced complexity that is proposed in this work (depicted in Figure 3.6). Then, the algorithms used to reduce the complexity in the SW/CC system are compared in the following subsection.

4.5.1 Numerical Example

Consider three files with a correlation model shown in Figure 3.5, where each file has an entropy of 60 bits. Furthermore, let the number of users $Z = 2$, each with cache size $MF = 90$ bits. For the sake of clarity, we denote $X_A^{i:j}$ as representing a sub-file of X_A, consisting of bits i to j inclusive, where $i < j$ and files begin with bit 1. As a result, the length of this file is determined as $\ell(X_A^{i:j}) = j - i + 1$ bits. In addition, let $\{X_A^{i:j}; X_B^{k:l}\}$ (with a semi-colon) be the concatenation of sub-files from X_A and X_B.

In the classic CC method, where correlation is ignored, the ideal placement of files is to divide each file in half and place each half at a different user. When the users' demands are revealed, the server XORs the files not currently stored by that user together and transmits the joint codeword. For instance, let the demand $\mathbf{d} = \{A, B\}$. Since user Z_1 already stored $X_A^{1:30}$ in its cache $\mathbf{Z}_1$ and user Z_2 has $X_B^{31:60}$ in $\mathbf{Z}_2$, the codeword transmitted by the server is $Y_\mathbf{d} = X_A^{31:60} \oplus X_B^{1:30}$. At the receiving end, the users XOR $Y_\mathbf{d}$ with the cached half of the file not requested to decode the rest of its requested file (e.g. Z_1 calculates $Y_\mathbf{d} \oplus X_B^{1:30} = X_A^{31:60}$). Under these conditions, it is readily verifiable that the delivery rate is $R_d = \bar{R}_m = 30$ bits.

The second approach, SW/CC without complexity reduction, uses SW coding before doing the CC. To begin, it is necessary to calculate the bounds on the coding rate when compressing the files. In this example, there are 7 conditions to be met:

$$R_A \geq 4 \tag{4.14}$$

$$R_B \geq 20 \tag{4.15}$$

$$R_C \geq 8 \tag{4.16}$$

$$R_A + R_B \geq 42 \tag{4.17}$$

$$R_A + R_C \geq 42 \tag{4.18}$$

$$R_B + R_C \geq 42 \tag{4.19}$$

$$R_A + R_B + R_C \geq 102 \tag{4.20}$$

The most restrictive of these is (4.20). Thus, to satisfy all conditions, it is sufficient to set all coding rates to $102/3 = 34$ bits. Let the files compressed using these compression rates be $\hat{X}_A$, $\hat{X}_B$ and $\hat{X}_C$ (this can be achieved using the MP method for SW coding [9]). Using the same cache size as above, we are now able to store 30 bits from each of these files, corresponding to 88% of the file as opposed to 50% in the CC method. In this case, both caches receive $\hat{X}_A^{1:30}$, $\hat{X}_B^{1:30}$, $\hat{X}_C^{1:30}$. The

Table 4.1: The entropies of the sources needed to index the files using the GISGEM algorithm

Source	$H(X_i\|\mathbf{N} \setminus X_i)$	$H(X_i\|X_1)$	$H(X_i\|X_2, X_1)$
X_B	20		
X_C	8	38	
X_A	4	34	4

remaining 12 bits (4 from each file) are transmitted regardless of the demand vector [1]. On the receiver's end, the new information is used to perform a joint decode to losslessly obtain X_A, X_B and X_C. As a result, the delivery rate is $R_d = \bar{R}_m = 12$ bits. However, this comes at an increased complexity cost, owing to the number of constraints that need to be met, as well as the joint decode complexity.

Finally, the SW/CC method is demonstrated with the complexity reduction approach presented in this chapter. Before performing the SW coding, the GISGEM algorithm is applied to the files to determine which ones to remove. Table 4.1 lists all the iterations used to index the files. In the first step, the excess entropies for each file (i.e. the entropy of each file conditioned on all other files) is calculated and the maximum is chosen to be X_1. The next iterations exclude all previously chosen sources and the conditional entropies are calculated, with the maximum chosen in each successive stage. In this example, the selection procedure is always optimal, since the first and last steps are guaranteed to be optimal and there are only 3 iterations.

If we set $\gamma = 1$ for example, the algorithm will suggest that X_B is removed. Thus, the library is partitioned into two subsets, $\mathbf{N}_c = \{X_A, X_C\}$ and $\mathbf{N}_u = \{X_B\}$. Consequently, the SW coding method from the previous SW/CC example will only be applied to the two files in $\mathbf{N}_c$. These files have 3 conditions for lossless SW coding:

$$R_A \geq 4 \tag{4.21}$$

$$R_C \geq 8 \tag{4.22}$$

$$R_A + R_C \geq H(X_A, X_C|X_B) = H(X_A, X_C) = 82 \tag{4.23}$$

Notice that the bound (4.23) is different to (4.18), since X_B is treated as independent of X_A and X_C in this case. Setting $R_A = R_C = 41$ bits satisfies all requirements and

[1] In this specific example, there is no need for the CC method, since the cache sizes are large enough to contain sufficient information to negate the extra coding.

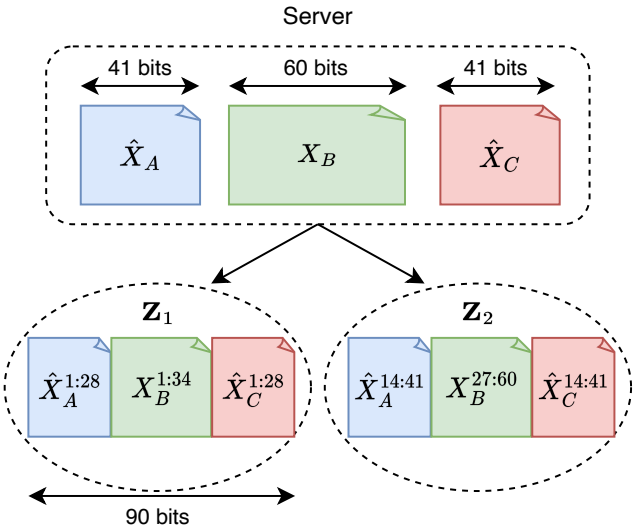


Figure 4.1: Cache storage contents for $|\mathbf{N}| = 3$ files, $Z = 2$ users and $MF = 90$ bits using the SW/CC method with reduced complexity

the files are compressed accordingly. Thus, the library now consists of $\{\hat{X}_A, X_B, \hat{X}_C\}$. In the next phase, the cache encoder stores 28 bits from the two compressed files and 34 bits from the uncompressed file. The caches are filled in this way so as to balance out the peak multicast rate. The results of this phase are shown in Figure 4.1, and the cache contents of each user is $\mathbf{Z}_1 = \{\hat{X}_A^{1:28}, X_B^{1:34}, \hat{X}_C^{1:28}\}$ and $\mathbf{Z}_2 = \{\hat{X}_A^{14:41}, X_B^{27:60}, \hat{X}_C^{14:41}\}$. Using this placement, all files can be decoded at the user end regardless of $\mathbf{d}$, as shown in Table 4.2. In terms of performance, Table 4.2 also shows that $R_d = \bar{R}_m = 26$ bits.

In summary, Table 4.3 shows what is stored at each cache for the different approaches. Although the SW/CC method has the lowest average and peak rates, it comes at the cost of higher complexity in designing the coding scheme as well as the decoding algorithm. On the other hand, the SW/CC method with GISGEM sacrifices some of the compression gains to achieve reduced complexity in terms of the number of equations, the decoding algorithm and the number of demand permutations in which SW decoding is needed at each cache (6 out of 12 permutations, compared to 12 for the SW/CC method). Nevertheless, the average rate is still reduced as compared to the regular CC approach.

Table 4.2: Encoding and decoding procedure for the hybrid SW/CC system with reduced complexity using GISGEM

d	Encoding ($Y_{\mathbf{d}}$)	$\ell(Y_{\mathbf{d}})$	Decoding for Z_1	Decoding for Z_2
$\{A,A\}$ $\{A,C\}$ $\{C,A\}$ $\{C,C\}$	$\{\hat{X}_A^{29:41}; \hat{X}_C^{29:41}\}$ $\oplus$ $\{\hat{X}_A^{1:13}; \hat{X}_C^{1:13}\}$	26	$Y_{\mathbf{d}} \oplus \{\hat{X}_A^{1:13}; \hat{X}_C^{1:13}\}$ + SW decoding	$Y_{\mathbf{d}} \oplus \{\hat{X}_A^{29:41}; \hat{X}_C^{29:41}\}$ + SW decoding
$\{A,B\}$ $\{C,B\}$	$\{\hat{X}_A^{29:41}; \hat{X}_C^{29:41}\} \oplus X_B^{1:26}$	26	$Y_{\mathbf{d}} \oplus X_B^{1:26}$ + SW decoding	$Y_{\mathbf{d}} \oplus \{\hat{X}_A^{29:41}; \hat{X}_C^{29:41}\}$
$\{B,A\}$ $\{B,C\}$	$X_B^{35:60} \oplus \{\hat{X}_A^{1:13}; \hat{X}_C^{1:13}\}$	26	$Y_{\mathbf{d}} \oplus \{\hat{X}_A^{1:13}; \hat{X}_C^{1:13}\}$	$Y_{\mathbf{d}} \oplus X_B^{35:60}$ + SW decoding
$\{B,B\}$	$X_B^{35:60} \oplus X_B^{1:26}$	26	$Y_{\mathbf{d}} \oplus X_B^{1:26}$	$Y_{\mathbf{d}} \oplus X_B^{35:60}$

Table 4.3: Cache contents for the different coding methods

	$\mathbf{Z}_1$	$\mathbf{Z}_2$
CC	$X_A^{1:30}$, $X_B^{1:30}$, $X_C^{1:30}$	$X_A^{31:60}$, $X_B^{31:60}$, $X_C^{31:60}$
SW/CC	$\hat{X}_A^{1:30}$, $\hat{X}_B^{1:30}$, $\hat{X}_C^{1:30}$	$\hat{X}_A^{1:30}$, $\hat{X}_B^{1:30}$, $\hat{X}_C^{1:30}$
SW/CC with GISGEM	$\hat{X}_A^{1:28}$, $X_B^{1:34}$, $\hat{X}_C^{1:28}$	$\hat{X}_A^{14:41}$, $X_B^{27:60}$, $\hat{X}_C^{14:41}$

4.5.2 Simulation Results

Three methods have been outlined in this chapter to determine the sources to remove in the SW/CC approach with reduced complexity. These are the BF, GA and GISGEM methods. The literature does not deal directly with the optimisation problem presented in Chapter 3, and the current approaches (when ignoring their simplifications) are equivalent to the BF method.

4.5.2.1 Genetic Algorithm Parameter Tuning

The GA has different parameters that can be tuned to obtain better results. As discussed in [74], these variables are critical to changing the diversification (breadth of search area) and intensification (refining the current results) of the algorithm. They also found that there is not necessarily a single parameter that controls each type of behaviour of the algorithm. In order to examine the effect of each parameter in our scenario, the GA was run with parameters in the ranges shown in Table 4.4. To

Table 4.4: Different parameter settings for GA testing and their impact on the quality of results and complexity

Parameter	Symbol	Range	Performance Impact	Time Impact	Iteration Impact
Stall Generations	SG	5, 10, 15, 20	Large	Large	Large
Population Size	$\|\mathbf{P}\|$	10, 15, 20	Medium	Small	Minimal
Elite Percentage	α_{e}	10%, 20%, 30%	Minimal	Small	Small
Crossover Percentage	α_{c}	20%, 40%, 60%, 80%	Small	Minimal	Minimal

explain the parameters: Stall Generations (SG) refers to the number of iterations in which the elite value obtained remains the same, after which the algorithm converges. The Population Size is the number of configurations tested in each generation. The value α_{e} is the number of members of $\mathbf{P}$ that survives to the next generation, expressed in terms of the percentage of $|\mathbf{P}|$. The Crossover Percentage is the number of children produced using the gene crossover method. It is expressed as the percentage of $|\mathbf{P}| - \alpha_{\mathrm{e}}$. The number of mutation children is not shown here, as it is automatically set as the inverse of α_{c}.

The GA was run for $|\mathbf{N}| = 18$ files and $\gamma \in \{7, 8, \ldots, 11\}$ for every combination of parameter. Figures 4.2-4.4 show the resulting effects, on average, of each parameter on the best result obtained, the time taken to converge and the number of iterations. These results are also summarised in Table 4.4. It is clear from the results that the SG has the biggest impact on both the quality of results and the time complexity. The trade-off between the two is found to be directly proportional. Consequently, an SG of 15 is chosen to slightly favour the quality of results over time complexity. Population size has a medium impact on quality but small impact on time complexity, while the number of iterations is barely affected. This is because increasing the population size increases the number of configurations tested per generation. For these reasons, the maximum population size of $|\mathbf{P}| = 20$ is chosen. The Elite Percentage has a minimal impact on quality of results but a small impact on time complexity. However, the results are found to not be linear, with $\alpha_{\mathrm{e}} = 20\%$ producing the best time relative to the quality of results. Thus, this value is chosen for the final testing. Finally, the balance between α_{c} and α_{m} corresponds to a small impact in quality but minimal impact on time. It is found that $\alpha_{\mathrm{c}} = 60\%$ maximises the best results obtained, so this is the value selected.

One interesting result is that a change in γ had a small effect on the time and minimal effect on the number of iterations taken to converge. This is owing to the fact that the GA operates on a combinatorial basis, meaning that the entire genome, and thus all files, is considered at once. This is unlike the GISGEM approach, which

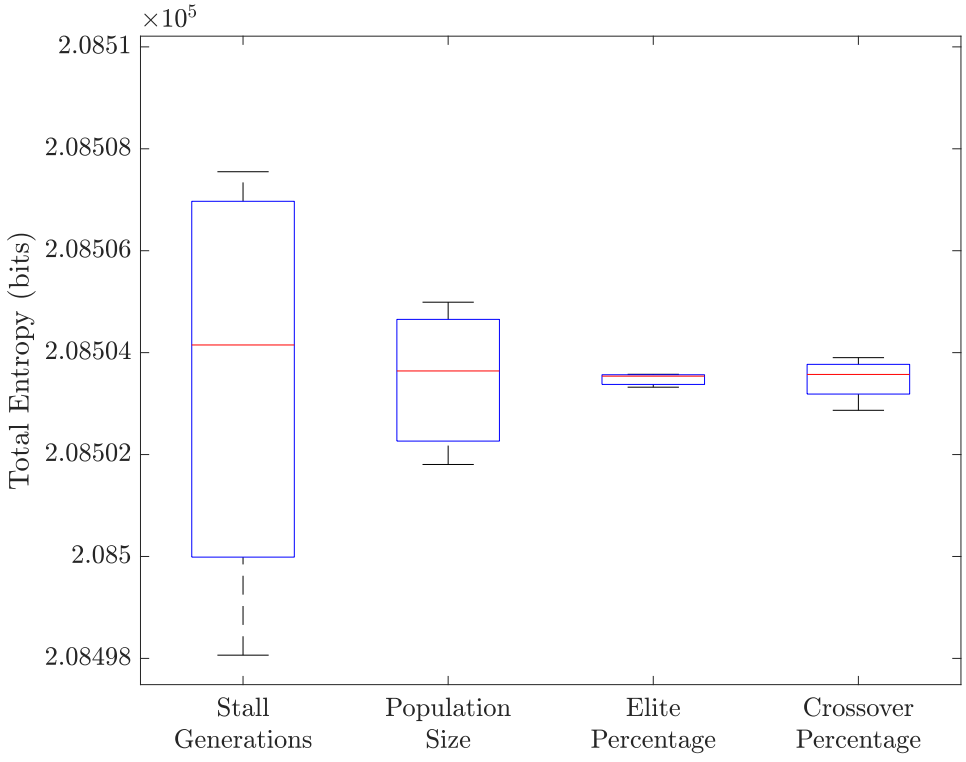


Figure 4.2: The effect of different GA configurations on performance

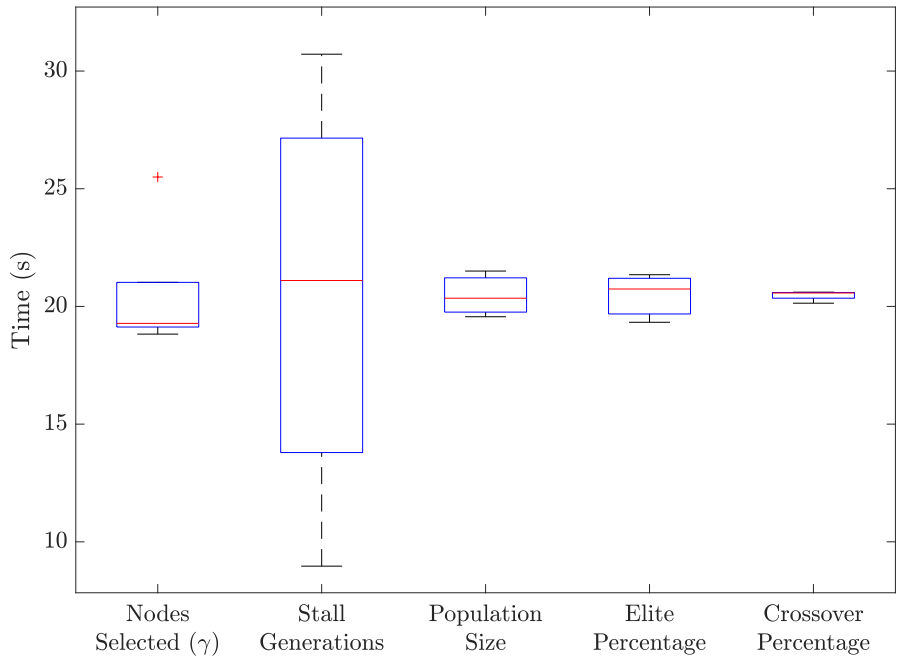


Figure 4.3: The effect of different GA configurations on convergence in terms of time

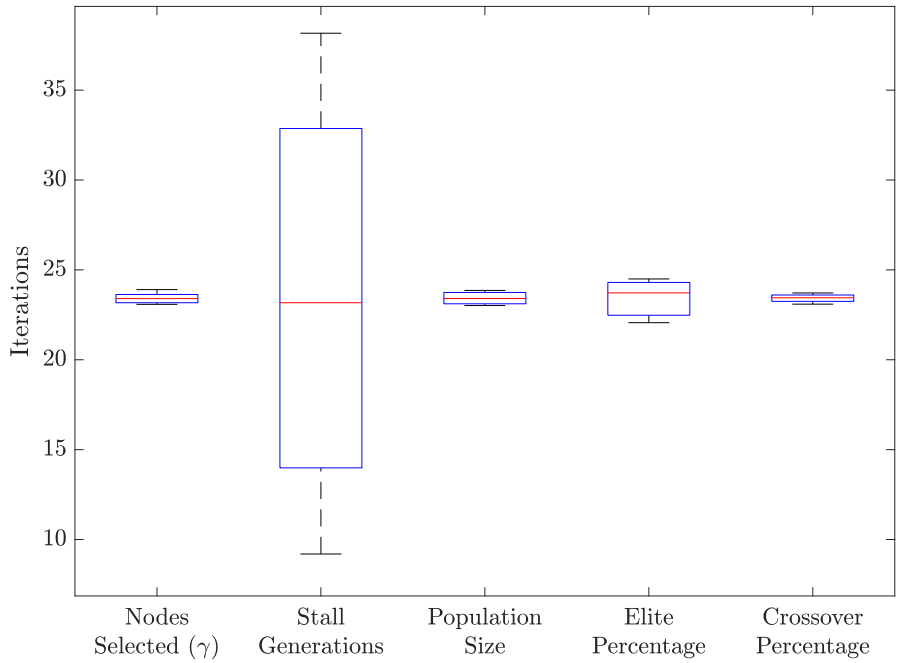


Figure 4.4: The effect of different GA configurations on convergence in terms of the number of iterations

is iterative and increases in complexity as γ increases.

4.5.2.2 GISGEM and GA Algorithm Comparison

To perform the comparison, 18 sources were used, where the optimal grouping was found for an increasing number of sources in $\mathbf{N}_u$. Testing was performed using Matlab on a MacBook Air (2013 model) with 1.3 GHz Dual-Core Intel Core i5 processor and 4 GB of 1600 MHz DDR3 RAM. Figure 4.5 shows the entropy obtained for each method when increasing the size of the selected group. Each of the methods are compared to the worst performing configuration (found using the BF method). The results confirm the sub-optimal performance predicted in Lemma 3, as it is found that, in the range of best to worst performing results, the GISGEM's results fall in the 91st percentile on average, with a minimum value falling in the 82nd percentile. The GA's results are in the 92nd percentile on average with the lowest value in the 76th percentile. As compared to the BF method, both the GISGEM and GA are able to find results close to the optimal one. However, since the GA is not constrained to choosing the same grouping as the previous iteration, it is sometimes able to find a better result than GISGEM. Figure 4.5 also shows the correctness of Theorem 1,

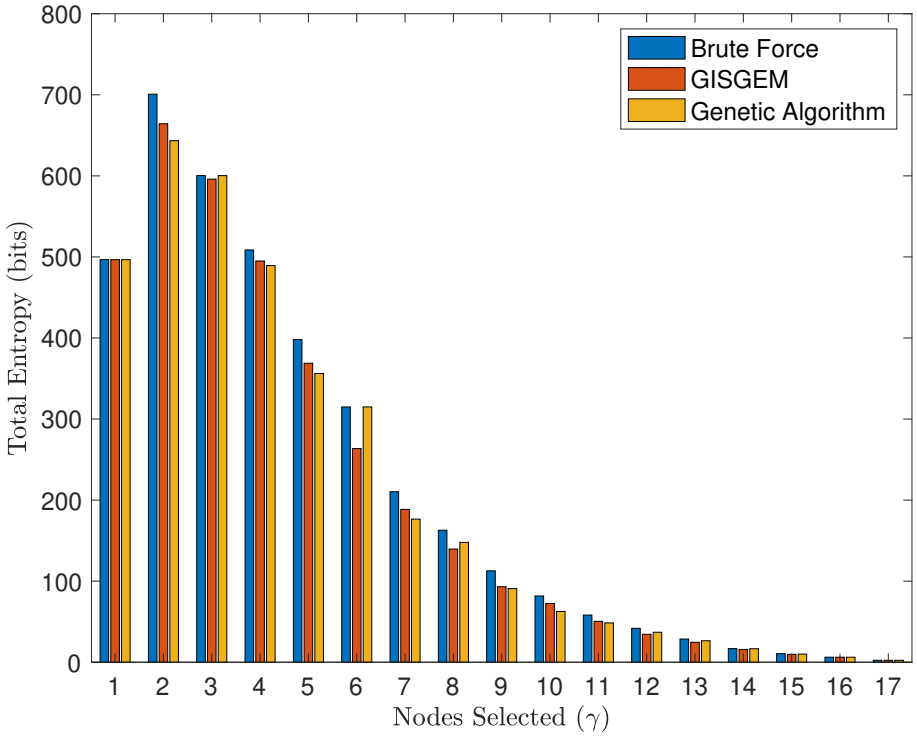


Figure 4.5: The performance of each selection method as compared to the least optimal combination for $|\mathbf{N}| = 18$ files

since the distance between the most and least optimal group selections decreases as more sources are selected. This highlights another advantage of GISGEM, as it is able to terminate earlier if it detects that the current number of sources is sufficient to achieve the objective in (3.17).

The time taken to find the optimal result for each method is given in Figure 4.6. The GA is, on average, 4.42 times faster than the BF exhaustive search, while GISGEM is 10.20 times faster and increases linearly with respect to γ. These practical results conform well to the theoretical complexity predictions in Section 4.4.1.

In another simulation, the GISGEM and GA methods were tested, this time for a system with $|\mathbf{N}| = 22$ files. At this number of nodes, the BF method's complexity becomes prohibitively large. In Figure 4.7, the difference between the total entropies of the uncoded groups $\mathbf{N}_u$ for the GISGEM and GA algorithms are plotted. It shows that, in this run, the GISGEM method outperforms the GA approach for small γ. For larger values of γ the two approaches are closer, with the GA approach sometimes besting GISGEM. Nevertheless, as depicted in Figure 4.8, the time complexity of GISGEM is consistently lower than the moving average of the GA.

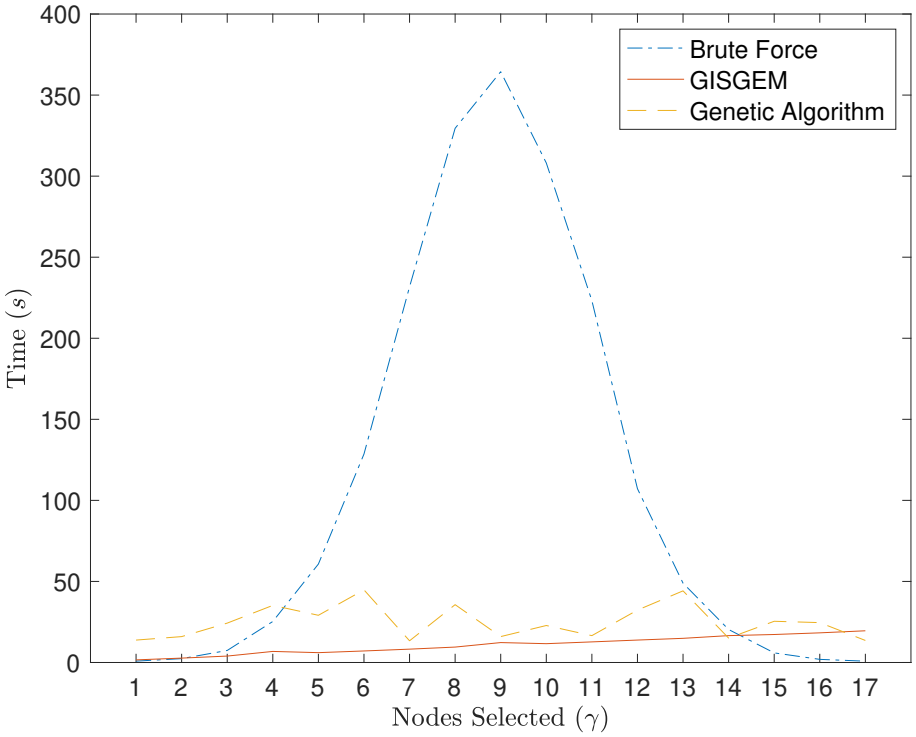


Figure 4.6: The time complexity of each selection method for $|\mathbf{N}| = 18$ files

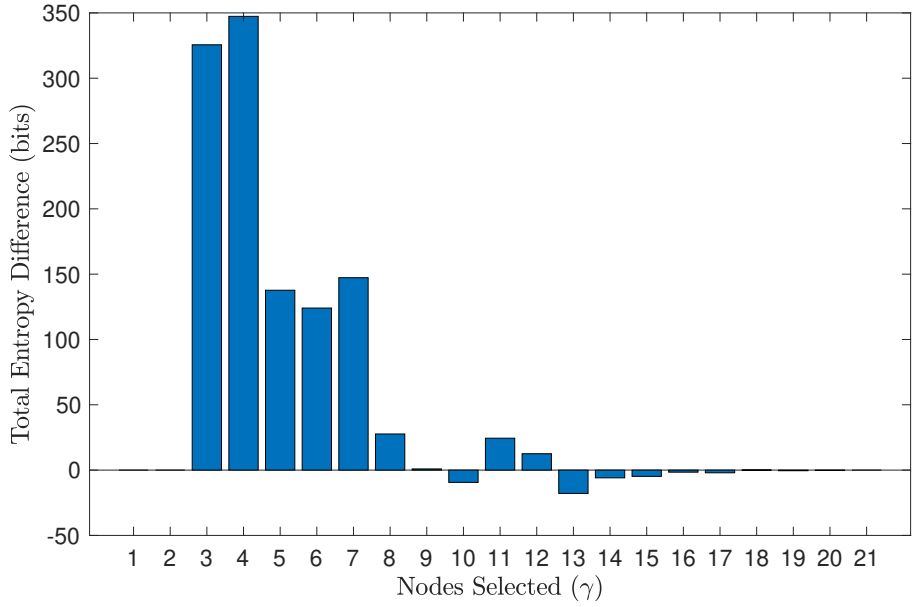


Figure 4.7: The difference between total entropies of $\mathbf{N}_u$ produced by the GISGEM and GA algorithms for $|\mathbf{N}| = 22$ files. A value above 0 indicates that the GISGEM result performs better than the GA

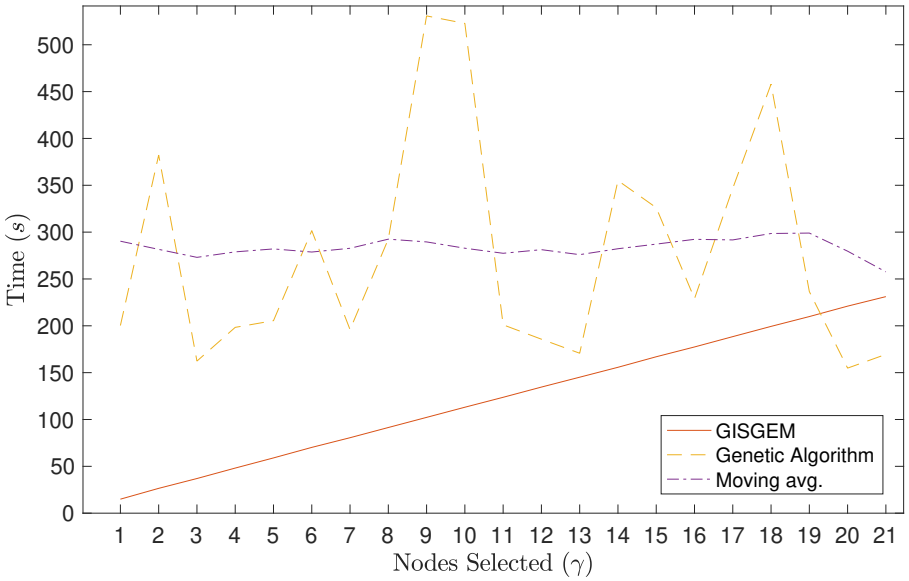


Figure 4.8: The time complexity of the GISGEM and GA selection methods for $|\mathbf{N}| = 22$ files

4.6 Conclusion

A greedy iterative selection procedure and meta-heuristic approach are proposed as potential solutions to the clustered problem in the context of caching correlated information. Files are grouped together, such that the overall decoding complexity is reduced with minimal impact on compression gains. The iterative method is based on the inherent properties of entropy and it is able to find close-to-optimal results with less time complexity as compared to the BF and meta-heuristic approaches. There is the additional benefit that the algorithm can bound the entropy gains at each iteration and can compare that to the relative complexity of the system, allowing it to terminate prematurely if necessary. It is demonstrated, using a numerical example, that these methods are able to successfully reduce the complexity of an SW/CC system while not sacrificing too much of the compression gains.

Chapter 5

Path-Based Single Group Optimisation

The algorithm presented in the previous chapter is expanded using a path-based approach. So too, a necessary condition of optimality is derived. This condition can be applied repeatedly to improve the performance of the results from the iterative algorithms. The ACO family of meta-heuristic algorithms is adapted to solve this problem, providing a benchmark that outperforms the GA presented in the previous chapter. The iterative algorithms have a larger time complexity than other solutions, but still converge in polynomial time. When combined with the optimality condition, they outperform all of the currently proposed algorithms that solve this problem to date. More specifically, this approach produces results that are found to fall in the 99.97th percentile on average.

5.1 Introduction

Thus far, a potential approach for solving the single group clustering problem has been presented as the GISGEM algorithm. It has a low time complexity and performs relatively well. However, it has the potential to be improved, by trading off the time complexity to improve the quality of the results. The previous chapter also showed the power of tuning variables, where the GA has its performance optimised by analysing the effect of different input variables. This could also be beneficial in improving the GISGEM approach. So too, without an optimality condition, it is difficult to know how well a cluster found by the algorithm actually performs, unless it is compared to a benchmark one. This is the main focus of this chapter.

The GA meta-heuristic used thus far is based on a combinatorial approach. The

improvements shown in this chapter now assume a path-based approach in order to improve the GISGEM approach. Consequently, another benchmark introduced is the ACO meta-heuristic, which is path-based in nature and could potentially outperform the GA.

The outline of this chapter is as follows: Section 5.2 gives the mathematical background needed to understand the path-based approach. Section 5.3 uses these concepts to develop the improved GISGEM algorithm and Section 5.4 describes the path-based meta-heuristic approach. The theoretical analysis is performed in Section 5.5 and an example of how the new algorithm works is given in Section 5.6. The algorithms presented in this chapter are benchmarked and compared to the previous chapter in Section 5.7. The results are discussed in more detail in Section 5.8. Section 5.9 finishes off this chapter.

5.2 Path-Based Modelling

To improve the GISGEM algorithm, a path-based approach is adopted. The paths are stored in a tree structure, the mathematical modelling of which is now outlined. Let $G = (V, A, t)$ be a directed graph representing a t-ary tree, where V is a set containing all of the vertices and A is a set of the connections between vertices. The vertices are indexed as v_i^d, where d is the depth of the vertex in the tree, $d \in \{1, 2, \ldots, \gamma\}$ and i is the unique index of the vertex within that depth. A special vertex v_0 is created at the root of the tree, which is used as a dummy variable to connect the tree at its base.

The tree is constrained by t, which determines the maximum number of children that any vertex can have. Thus, for any vertex v_i^d the index i is limited by the depth of that vertex and t, such that it falls in the range $i \in \{1, 2, \ldots, t^d\}$.

In the graph G, A is a set of ordered pairs $a = \{v_i^d, v_j^{d+1}\}$, where each a is the arc from parent vertex v_i^d to child vertex v_j^{d+1}. Necessarily, for all arcs $a \in A$ and vertices having a depth value in the range $d \in \{1, 2, \ldots, \gamma\}$, the indices of the parent and child vertices are in the ranges of $i \in \{1, 2, \ldots, t^d\}$ and $j \in \{t(i-1)+1, t(i-1)+2, \ldots, ti\}$ respectively. In this work, we denote parent-child relationships as $v_i^d \hookrightarrow v_j^{d+1}$. The kth parent of v_i^d is given by $v_{\lceil i/t^k \rceil}^{d-k}$, where $\lceil \cdot \rceil$ is the ceiling function.

Let $\mathbf{C}_{v_i^d} \coloneqq \{v_j^{d+1} \in V | v_i^d \hookrightarrow v_j^{d+1}\}$ be an ordered set of all the children of vertex v_i^d that exist in the graph. The set is ordered by the indices of the vertices in ascending

order. This implies that if a mapping of $c_l \to v_j^{d+1}$ and $c_m \to v_k^{d+1}$ exists for any two valid child elements $c_l, c_m \in \mathbf{C}_{v_i^d}$, then $l > m$ if $j > k$.

A path $\mathbf{P}_{v_i^d}$ is defined as an ordered set containing all parent vertices from the vertex v_i^d up to, but not including, the root vertex v_0. Taking the above constraints into account, the path is defined as $\mathbf{P}_{v_i^d} := \{v_i^d, v_{\lceil i/t^1 \rceil}^{d-1}, v_{\lceil i/t^2 \rceil}^{d-2}, \ldots, v_{\lceil i/t^{d-1} \rceil}^{1}\}$ and contains $|\mathbf{P}_{v_i^d}| = d$ elements. Furthermore, let the path element $p_j \in \mathbf{P}_{v_i^d}$, $j \in \{1, 2, \ldots, d\}$ correspond to the vertex $v_{\lceil i/t^{j-1} \rceil}^{d-j+1}$.

Each vertex v_i^d is associated with a single entropy source $X_k \in \mathbf{N}$. Let $f^{\mathcal{X}} : v_i^d \to X_k$ be a function that maps a vertex to an entropy source. We require that, within a given path $\mathbf{P}_{v_i^d}$, all path elements are mapped in a one-to-one fashion with entropy sources, i.e. $f^{\mathcal{X}}(p_j) \neq f^{\mathcal{X}}(p_k)$, $\forall p_j, p_k \in \mathbf{P}_{v_i^d}$, $p_j \neq p_k$.

Using the above terminology, the entropy of path $H(\mathbf{P}_{v_i^d})$ can be calculated as follows:

$$H(\mathbf{P}_{v_i^d}) = \sum_{n=1}^{d} H(v_{\lceil i/t^{d-n} \rceil}^{n} | v_{\lceil i/t^{d-n+1} \rceil}^{n-1}, v_{\lceil i/t^{d-n+2} \rceil}^{n-2}, \ldots, v_{\lceil i/t^{d-1} \rceil}^{1}) \tag{5.1}$$

$$= \sum_{n=1}^{d} H(p_n | p_{n-1}, p_{n-2}, \ldots, p_1) \tag{5.2}$$

For brevity, $H(v_i^d)$ and $H(p_1)$ are equivalent to $H(f^{\mathcal{X}}(v_i^d))$.

5.3 Path-Based Greedy Iterative Selection Procedure

The general approach to improving the GISGEM algorithm is to broaden the search range of the algorithm, which allows it to overcome the greedy nature of the algorithm somewhat and gives it a better chance at finding the optimal result. The following subsections describe the main algorithm together with its variants and considerations.

5.3.1 Champion Selection Procedure

The GISGEM algorithm in Chapter 4 selects a single node at each iteration by maximising the conditional entropy gain at that iteration. Using the definitions in Section 5.2, this can be viewed as an algorithm that generates a single path, with $t = 1$. A major shortcoming of this approach is that it is possible for a better

performing solution to exist that does not contain nodes that fit these rigid criteria. As a result, the general improvement is to allow for a greater number of paths to be considered, increasing the value of t. However, this would also mean that the number of paths would increase exponentially, generating t^γ paths. To limit this, we introduce the champion condition, which limits the total number of paths to a different value, denoted as $m < t$.

The general outline of the algorithm is as follows. In the beginning, there is only the root node, which produces t vertex children that are assigned sources sorted in descending order according to their entropy. Of those, m will survive to the next iteration and produce children. In the next iteration, only m of those children which have the highest path entropy will survive to the next iteration, and so on. At each stage, the new child vertices and parent-child arc connections are added to the graph G. This continues until a path of size γ is produced. The algorithm is described more formally in Procedure 2 and summarised in Algorithm 3.

Procedure 2 (Champion Selection Procedure (CSP)). Given a set of entropy sources $\mathbf{N}$, with restrictions t and m, let the best path of length γ be chosen as follows:

In the first iteration, t vertices are created with parent v_0. The vertices are assigned entropy sources such that:

$$H(v_i^1) \geq H(v_j^1), \quad \forall i, j \in \{1, 2, \ldots, t\}, i < j \tag{5.3}$$

Of these, only the m best vertices are selected. Let $\mathbf{M}_i$ be the set of the m best performing paths in iteration i, with $|\mathbf{M}_i| = m$. In the subsequent iterations $d \in \{2, 3, \ldots, \gamma\}$, all of the vertices in $\mathbf{M}_{d-1}$ produce t children. Thus, there are tm children at each iteration $d > 1$. For each of the selected vertices $v_i^{d-1} \in \mathbf{M}_{d-1}$, the children are assigned entropy sources such that:

$$H(v_j^d|\mathbf{P}_{v_i^{d-1}}) \geq H(v_k^d|\mathbf{P}_{v_i^{d-1}}), \quad \forall j, k \in \{t(i-1)+1, t(i-1)+2, \ldots, ti\}, j < k \tag{5.4}$$

In order to maintain the unique sources property for all paths, let the potential sources to choose from for each parent vertex v_i^{d-1} be limited to the set:

$$\mathbf{S}_{v_i^{d-1}} \coloneqq \mathbf{N} \setminus f^{\mathcal{X}}(\mathbf{P}_{v_i^{d-1}}) \tag{5.5}$$

As defined above, $\mathbf{C}_{v_i^{d-1}}$ contains the children of v_i^{d-1}. Let $\mathbf{C} \coloneqq \bigcup \mathbf{C}_{v_i^{d-1}}$, $\forall v_i^{d-1} \in \mathbf{M}_{d-1}$ be a set containing all of the children vertices created in this step.

Next, the number of paths is restricted by m. The m best performing paths are chosen by evaluating $H(\mathbf{P}_{v_i^d})$, $\forall v_i^d \in \mathbf{C}$ and sorting them in descending order. These selected vertices are stored in $\mathbf{M}_d$ and will produce children in the next iteration.

It is possible for there to be multiple valid possibilities when assigning entropy sources according to (5.3) and (5.4), and when selecting the best m performers. This can happen if the entropies are equal, in which case the order of the sources should be chosen arbitrarily.

The algorithm continues until $d = \gamma$, at which point the best-performing path is chosen as the final solution.

5.3.2 Champion Selection Procedure Variations

The CSP approach broadens the search range of the GISGEM algorithm. Nevertheless, there are a few issues that could arise when using the algorithm without alterations. This subsection looks at various improvements to the algorithm that attempt to solve or minimise these issues.

One major limitation of the CSP method is that, for a given iteration, it is possible that all t candidate vertex choices for a given parent vertex v_i^{d-1} are not strong enough to be selected. Thus, the entire potential path $\mathbf{P}_{v_i^{d-1}}$ will be discarded at this step, even though it is possible that the path will produce a better result in a later iteration. This means that it is possible for the CSP to find a worse result than the standard GISGEM approach, if the first path is discarded. To avoid this, a modified selection procedure is defined in the following lemma, in which a fixed number of starting vertices are guaranteed to have at least one path up for consideration in the final path selection.

Lemma 5 (CSP with Ancestor Guarantee (CSP-AG)). Let the children set $\mathbf{C}$ be created according to Procedure 2. Then, when selecting the m children that will produce children in the next iteration, first include the best performing child from each path that contains v_i^1, $i \in \{1, 2, \ldots, \mathrm{AG}\}$, where $\mathrm{AG} \leq m$. The remaining

Algorithm 3 Champion Selection Procedure

Require: $t \leq |\mathbf{N}|$, $m \leq t$

$V \leftarrow \{v_0\}$

$A \leftarrow \emptyset$

$G \leftarrow (V, A, t)$

$\mathbf{C} \leftarrow \{v_1^1, v_2^1, \ldots, v_t^1\}$ s.t. $H(v_1^1) \geq H(v_2^1) \geq \cdots \geq H(v_t^1)$

$V \leftarrow V \cup \mathbf{C}$

$\mathbf{M}_1 \leftarrow \{v_1^1, v_2^1, \ldots, v_m^1\}$

for all $v_i^1 \in \mathbf{C}$ **do**

$A \leftarrow A \cup \{v_0, v_i^1\}$

end for

for $d \in \{2, 3, \ldots, \gamma\}$ **do**

for all $v_i^{d-1} \in \mathbf{M}_{d-1}$ **do**

$\mathbf{C}_{v_i^{d-1}} \leftarrow t$ best $X_j \in \mathbf{S}_{v_i^{d-1}}$, sorted according to (5.4)

$V \leftarrow V \cup \mathbf{C}_{v_i^{d-1}}$

for all $v_j^d \in \mathbf{C}_{v_i^{d-1}}$ **do**

$A \leftarrow A \cup \{v_i^{d-1}, v_j^d\}$

end for

end for

$\mathbf{C} \leftarrow \bigcup \mathbf{C}_{v_i^{d-1}}$, $\forall v_i^{d-1} \in \mathbf{M}_{d-1}$

sort $\mathbf{C}$ s.t. $H(\mathbf{P}_{c_1}) \geq H(\mathbf{P}_{c_2}), \cdots \geq H(\mathbf{P}_{c_{|\mathbf{C}|}})$

$\mathbf{M}_d \leftarrow \{c_1, c_2, \ldots, c_m\} \subset \mathbf{C}$

end for

$m -$ AG children to include are chosen as in Procedure 2. If this updated procedure is followed, then it will always produce a result greater or equal to the GISGEM procedure.

Proof. If AG ≥ 1, then a path with vertex v_1^1 will always exist, as per the updated selection procedure. In the first iteration, the first vertex is assigned an entropy source such that $v_1^1 = \max_{X_i \in \mathbf{N}} H(X_i)$, which is identical to the choice of the GISGEM approach. In subsequent steps, the best vertex that has v_1^1 in its path will be v_1^d, since this fulfils the condition in (5.4). Consequently, this updated selection procedure guarantees that the path $\mathbf{P}_{v_1^\gamma}$ is considered in the final step, which is identical to the path chosen by GISGEM. As a result, this path will be selected, unless there is a better performing path. □

Another problem that could arise with the plain CSP approach is that the paths

could converge with one another, since, in subsequent steps, they could choose nodes that causes the path to be permutations of one another. Since the entropy function is order-agnostic, their total entropy will be the same at this point. Thereafter, their paths will not diverge again and will terminate with the same solution. This has the effect of reducing the number of potential paths searched, negating the purpose of increasing the breadth in the first place. The next lemma provides a means to prevent paths from converging, but requires definitions of some further terms.

Let $\mathbf{E}_{v_i^d} \subset \mathbf{C}_{v_j^{d-1}}$, $v_j^{d-1} \hookrightarrow v_i^d$ be a set of vertices taken from the children set of the parent of v_i^d, such that $\forall v_k^d \in \mathbf{E}_{v_i^d}$, $k \leq i$. In terms of the drawing procedure outlined in Procedure 2, this means that the set contains vertices that evaluate to the same or higher entropy than v_i^d.

Lemma 6 (CSP with Convergence Prevention (CSP-CP)). Let the general selection procedure follow that outlined in Procedure 2. However, let the construction of $\mathbf{S}_{v_i^{d-1}}$ be updated as follows:

$$\mathbf{S}_{v_i^{d-1}} \coloneqq \mathbf{N} \setminus \{ \bigcup_{n=d-1}^{1} f^{\mathcal{X}}(\mathbf{E}_{v^n_{\lceil i/t^{d-1-n} \rceil}}) \} \tag{5.6}$$

In essence, this means that for each vertex in the path $\mathbf{P}_{v_i^{d-1}}$, the entropy source for that vertex and the other children with a lower index are excluded from the pool used to select the next generation's sources. If this updated drawing procedure is used, any two paths in G will be different by at least one entropy source.

Proof. Two paths $\mathbf{P}_{v_i^d}$ and $\mathbf{P}_{v_j^d}$ are said to have converged if $f^{\mathcal{X}}(\mathbf{P}_{v_i^d}) = f^{\mathcal{X}}(\mathbf{P}_{v_j^d})$. Before the next iteration in the selection procedure, the most similar paths are ones where the two vertices share a parent, i.e. $v_k^{d-1} \hookrightarrow v_i^d$ and $v_k^{d-1} \hookrightarrow v_j^d$. At this point, the difference between the two paths $f^{\mathcal{X}}(\mathbf{P}_{v_i^d})$ and $f^{\mathcal{X}}(\mathbf{P}_{v_j^d})$ is $\{f^{\mathcal{X}}(v_i^d), f^{\mathcal{X}}(v_j^d)\}$, since they share a common ancestor. Without loss of generality, assume $i > j$. According to Lemma 6, vertex v_j^d will not include $f^{\mathcal{X}}(v_i^d)$ as part of its entropy source pool in subsequent iterations. Thus, the continuations of the paths stemming from these nodes will always differ by at least $f^{\mathcal{X}}(v_i^d)$. Since the graph is connected, it will always be possible to find a common ancestor for any two paths and the paths will differ by at least one of the children of that ancestor. □

5.3.3 Branch and Bound Considerations

The branch and bound method involves terminating a branch of the graph prematurely if certain conditions (usually a lower or upper bound) are determined to be impossible to satisfy if this branch continues to be used. In Chapter 4, a stopping condition is given, which can be used in the CSP approach as well. In addition, it is possible to use a pre-existing path's performance and find the difference between it and a given path to determine the entropy gap. For example, the GISGEM approach can be used to find a single path. Then, when evaluating the other paths, the following stopping condition can be used:

Lemma 7. Let $\mathbf{P}_{v_{\text{ref}}}$ be a path of length γ with known entropy H_{ref}. A given path $\mathbf{P}_{v_i^d}$ cannot be optimal if the following condition is not satisfied:

$$H_{\text{ref}} - H(\mathbf{P}_{v_i^d}) > (\gamma - d)H(v_i^d|\mathbf{P}_{v_i^d} \setminus v_i^d) \tag{5.7}$$

Proof. Since conditioning reduces entropy, any v_j^l, a descendant of v_i^d ($d > l$, $j \geq i$) will have an entropy bounded by:

$$H(v_j^l|\mathbf{P}_{v_j^l} \setminus v_j^l) \leq H(v_i^d|\mathbf{P}_{v_i^d} \setminus v_i^d) \tag{5.8}$$

As there are only $\gamma - d$ nodes left to choose from at iteration d, the most gain in entropy is if $H(v_j^l|\mathbf{P}_{v_j^l} \setminus v_j^l) = H(v_i^d|\mathbf{P}_{v_i^d} \setminus v_i^d)$ for all subsequent descendants. If this is less than the difference between H_{ref} and the current entropy of the path, then it will be impossible for this path to exceed the reference entropy and thus cannot be optimal. □

If GISGEM is used to determine $\mathbf{P}_{v_{\text{ref}}}$, then this is equivalent to $\mathbf{P}_{v_1^\gamma}$.

There is a further optimisation for the CSP-CP. Since the selection pool is limited, this means that it can prematurely terminate a path if not enough nodes remain for it to complete it.

Lemma 8. If, while using the CSP-CP algorithm at iteration d, a parent v_i^{d-1} has a node pool that satisfies the following condition:

$$|\mathbf{S}_{v_i^{d-1}}| < \gamma - d \tag{5.9}$$

then there are not enough nodes remaining for path $\mathbf{P}_{v_i^{d-1}}$ to grow to size γ.

Proof. The selection pool $\mathbf{S}_{v_i^{d-1}} \subset \mathbf{N}$ determines which nodes can be chosen from. This selection pool decreases by at least one element at each iteration, since the selected child is removed from $\mathbf{S}_{v_i^{d-1}}$ in the subsequent step. At iteration d, there are $\gamma - d$ nodes left to choose for that path. Thus, if the selection pool is smaller than this, it will be impossible for the path length to reach γ. □

5.3.4 Optimality Tests

Despite the modifications to the GISGEM selection procedure outlined in Section 5.3.1, as well as the variations in Section 5.3.2, it is still possible that the optimal result is not obtained. In the following lemma, a necessary condition for optimality is derived.

Lemma 9 (optimality condition). A necessary condition of optimality for a given path $\mathbf{P}_{v_i^d}$ is that each path element $p_j \in \mathbf{P}_{v_i^d}$ is the solution to the optimisation condition:

$$p_k^* = \arg\max_{p_k} H(p_k | \mathbf{P}_{v_i^d} \setminus p_k) \tag{5.10}$$

where $p_k \in \mathbf{N} \setminus \{\mathbf{P}_{v_i^d} \setminus p_k\}$.

Proof. A path is a set of vertices, ordered by the iteration it was chosen in. The CSP procedure makes the optimal choice for a single iteration, since it finds the node that provides the greatest increase in entropy. However, the order of calculating the entropy of a path in (5.2) does not depend on order. Thus, for a path to be optimal, any node removed from the path should be the solution to the CSP selection requirement, if that node were to be one up for selection in the final iteration. □

Lemma 10. The optimality test in Lemma 9 is not a sufficient condition of optimality.

Proof. Let $\mathbf{N}$ be composed of two independent and disjoint sets $\mathbf{N}_1$ and $\mathbf{N}_2$. As a result:

$$H(X_i|X_j) = H(X_j|X_i) = 0 \tag{5.11}$$
$$\forall X_i \in \mathbf{N}_1, X_j \in \mathbf{N}_2$$

Furthermore, let the optimal combination of nodes be contained in $\mathbf{N}_2$. Consequently, any path $\mathbf{P}_{v_i^\gamma} \subset \mathbf{N}_1$ will be sub-optimal. If $\mathbf{P}_{v_i^\gamma}$ is tested using the optimality condition in Lemma 9, sources from $\mathbf{N}_2$ will never be suggested, since their conditional entropy will always be 0 as shown in (5.11). □

Corollary 1 (Iterative Optimisation Improvement (IOI)). In the optimality test described in Lemma 9, if any $p_k \in \mathbf{P}_{v_i^d}$ is not the solution to optimisation condition (5.10), then continuously replacing p_k with the actual solution p_k^* will only improve the overall result. Let $\mathbf{P}^*_{v_i^d}$ be the path obtained using this approach. Then it is guaranteed that:

$$H(\mathbf{P}^*_{v_i^d}) > H(\mathbf{P}_{v_i^d}) \tag{5.12}$$

Proof. The entropy term given in (5.10) is the last term in the total entropy calculation in (5.1) and (5.2). Although, in those equations, the ordering is important, nevertheless, the actual entropy function is order-agnostic. As a result, let p_k be the last source in $\mathbf{P}_{v_i^d}$, implying that $k = d$.

It is clear from (5.2) that the last source does not appear in any of the conditional terms for the other sources, since for any term in the summation in (5.2) for path element p_i, the conditional term indices are in the set $\mathbf{I} = \{i-1, i-2, \ldots, 1\}$ and if $k \geq i$, $k \notin \mathbf{I}$. Thus, replacing p_k will only change the value of the final term in the calculation. As a result, if $H(p_k^*|\mathbf{P}_{v_i^d} \setminus p_k) > H(p_k|\mathbf{P}_{v_i^d} \setminus p_k)$, then replacing p_k with p_k^* will increase the value of the total entropy. □

Consequently, Corollary 1 shows that the optimality test in Lemma 9 can repeatedly be used to improve any grouping result. However, Lemma 10 shows that it is possible for this approach to find a solution that is a local maximum instead of a global one.

In the next result, we extend Lemma 9 to include checking the optimality of a path with multiple nodes.

Corollary 2 (extension of the optimality test). Let $\mathbf{P}^{(c)}_{v_i^d} \subset \mathbf{P}_{v_i^d}$ be a set of all combinations of nodes with length c. Thus, $|\mathbf{p}_j| = c$, $\forall \mathbf{p}_j \in \mathbf{P}^{(c)}_{v_i^d}$ and $|\mathbf{P}^{(c)}_{v_i^d}| = \binom{d}{c}$. A necessary condition of optimality for the path $\mathbf{P}_{v_i^d}$ is that each $\mathbf{p}_j \in \mathbf{P}^{(c)}_{v_i^d}$ satisfies the following optimisation condition:

$$\mathbf{p}_k^* = \arg\max_{\mathbf{p}_k} H(\mathbf{p}_k|\mathbf{P}_{v_i^d} \setminus \mathbf{p}_k) \tag{5.13}$$

where $\mathbf{p}_k \subset \mathbf{N} \setminus \{\mathbf{P}_{v_i^d} \setminus \mathbf{p}_k\}$.

Proof. The proof follows a similar line to the proof of Lemma 9, replacing a single source with a set of sources. Instead of setting the final source in $\mathbf{P}_{v_i^d}$, the final c sources in $\mathbf{P}_{v_i^d}$ are set to $\mathbf{p}_k$. □

5.4 Path-Based Meta-Heuristic Approach

Encouraged by the modifications made to the iterative search algorithm in Chapter 4 using a path-based mathematical model, the meta-heuristic approach is also updated using this outlook. The ACO is a family of meta-heuristic algorithms whose efficacy in path-finding problems is well known [75–77]. The algorithms are carried out by agents called ants, and each finds a potentially optimal path based on their starting node and a combination of heuristic and deterministic approaches.

Different variants of the approach have been proposed in literature [78]. For this work, we select the Ant Colony System (ACS) variant, since it performs well in the travelling salesman optimisation problem and is an improvement to the original Ant System (AS) algorithm [79]. A summary of the algorithm is now given, together with the modifications that are made to tailor the algorithm to the considerations of this work.

The algorithm is initialised by calculating the distances between all nodes in $\mathbf{N}$. In our adaptation, the distance between node X_i and X_j is calculated as the conditional entropy $\eta_{ij} = H(X_j|X_i)$. This format is chosen, since it represents the increase in entropy when including X_j in the solution. However, this is not totally accurate, since the entropy should also take into account the other nodes that exist in the path at that point. Nevertheless, it was decided to use this metric, since determining the distance is an expensive operation and would have to be done for every unique combination of nodes. So too, this value would change depending on how many nodes had previously been chosen in the path. In contrast to this, the distance values can be calculated once at the beginning of the algorithm and used throughout its duration.

The pheromone values between the nodes are also calculated at this stage. The suggested method of determining the starting pheromone values is $1/|\mathbf{N}|C_{nn}$, where C_{nn} is the performance of a path selected using the Nearest Neighbour algorithm [78]. Instead, the inverse of that given in literature is used, since the objective is to maximise the distance. Thus, the starting values are given by $|\mathbf{N}|C_{nn}$ in the adapted version.

The next phase of the algorithm involves finding the best paths using the ant agents and updating the pheromone values. This phase is looped until a stopping condition is met, which could be a predetermined number of iterations or dependent on the change in the global best result between iterations. Each ant constructs its path of length γ iteratively, with the starting node chosen randomly. At each subsequent iteration, the ant first constructs a set of attractiveness for the neighbouring nodes that have not already been selected. The attractiveness set is calculated as:

$$p_{ij}^{k}(t) = \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in \mathbf{N}_i^k}[\tau_{il}(t)]^{\alpha}[\eta_{ij}]^{\beta}} \tag{5.14}$$

where $p_{ij}^{k}(t)$ is the attractiveness of moving from node X_i to X_j for ant k at iteration t and $\tau_{ij}(t)$ is the pheromone strength of this arc. The value is normalised over the attractiveness of all possible moves available to ant k, the set of which is denoted as $\mathbf{N}_i^k$. The two constants, α and β, control the influence of the two metrics.

Next, the ant chooses whether to use a deterministic or probabilistic method to choose the next node to move to. With probability q_0, the ant will choose the arc that maximises (5.14). Conversely, with probability $1 - q_0$, ant k will randomly select the arc, but the chance of choosing an arc is directly related to its attractiveness as calculated using (5.14).

Immediately after an ant selects an arc, it will reduce the amount of pheromone that is deposited there in order to prevent the other ants from selecting similar paths. The pheromone reduction is performed according to the following calculation:

$$\tau_{ij}(t) = (1 - \xi)\tau_{ij}(t) + \xi\tau_0 \tag{5.15}$$

where ξ and τ_0 are predetermined constants. This procedure is followed for all ants until they have chosen their respective paths. The paths are evaluated as their joint entropy and are ranked. The algorithm first checks if a new global best has been found and updates it accordingly. Then, only the arcs that are in the global best path have their pheromone levels updated according to the following formula:

$$\tau_{ij}(t+1) = (1 - \rho)\tau_{ij}(t) + \rho\Delta\tau_{ij}^{gb}(t) \tag{5.16}$$

where all ij are arcs in the global best path, ρ is a constant that determines the pheromone evaporation and $\Delta\tau_{ij}^{gb}(t)$ is the joint entropy of the global best path.

The algorithm has two stopping conditions: the Maximum Number of Iterations (MNI) and number of iterations in which a new best path is not found, also known as the number of SGs. If either of these two conditions are met, the algorithm will terminate and the global best path is returned. The algorithm is summarised in Algorithm 4.

Algorithm 4 Ant Colony Optimisation Approach

Initialise the distance and pheromone states:
for all $\eta_{ij} \in D$ **do** ▷ Create distance matrix
  $\eta_{ij} \leftarrow H(X_j|X_i)$
end for
for all $\tau_{ij} \in P$ **do** ▷ Create pheromone matrix
  $\tau_{ij} \leftarrow |\mathbf{N}|C_{nn}$ ▷ Best path performance using the Nearest Neighbour algorithm
end for
while Stopping conditions have not been met **do**
  while Path length $< \gamma$ **do**
    for each ant k **do**
      Construct attractiveness set according to (5.14)
      With probability q_0:
      NextNode $\leftarrow \max_{j \in \mathbf{N}_i^k}(p_{ij}^k(t))$
      Otherwise:
      NextNode $\leftarrow$ random biased choice using (5.14)
      $\mathbf{P}_k \leftarrow \mathbf{P}_k \cup$ NextNode ▷ Update path of ant k
      $\tau_{ij}(t) \leftarrow (1-\xi)\tau_{ij}(t) + \xi\tau_0$ ▷ Update arc's pheromone
    end for
  end while
  Evaluate performance of chosen paths
  Update global best
  for all arc $ij \in$ global best path **do**
    $\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij}^{gb}$
  end for
end while

5.5 Complexity Analysis

In Chapter 4, it is shown that the GISGEM algorithm needs to perform the entropy calculation (the most computer intensive operation) $\gamma|\mathbf{N}|$ times, giving a complexity of $O(|\mathbf{N}|)$. The CSP is an extension of the GISGEM algorithm, that allows for a maximum of m different GISGEM paths to be traversed. However, the first step in the CSP and GISGEM approaches are identical and the CSP algorithm is m times more complex for subsequent steps. As a result, the CSP approach performs the entropy calculation $|\mathbf{N}|(1 + m(\gamma - 1))$ times and thus also has a complexity of order $O(|\mathbf{N}|)$. Although the CSP-CP variant is able to terminate paths prematurely if it is not able complete a path, nevertheless, it does not have a significant effect on the order of the time complexity.

It is interesting to note that the value of t does not significantly affect the complexity, since its role is to include more paths in the selection stage. As previously discussed in Chapter 4, the complexity of calculating the entropy of the paths is significantly more than the ranking complexity. Furthermore, the GISGEM approach requires all potential nodes' entropies to be calculated in any event. The only role of t is how many of those nodes that are not the maximum are included for potential selection in the next step.

The optimality test proposed in Lemma 9 requires that each node in the path be removed and tested again as if it were the selection stage of the GISGEM algorithm. As a result, the complexity of running the optimality test is the same as creating a path of the same length. Thus, it requires $\gamma|\mathbf{N}|$ entropy calculations as well, with order $O(|\mathbf{N}|)$. However, the extension of the optimality test requires testing all combinations of nodes in the path. This has the effect of increasing the complexity exponentially.

The complexity of the ACS is dependent on the number of iterations and ants that are in the system. In order to calculate the distances in the beginning, it is necessary to compute the conditional entropy for all $\binom{\mathbf{N}}{2} = |\mathbf{N}|(|\mathbf{N}| - 1)/2$ combinations. So too, it is necessary to evaluate the path of the Nearest Neighbour algorithm once to calculate the initial value for the pheromones. Thereafter, building the paths requires comparatively less complexity, as well as the pheromone calculations. However, all k paths need to be evaluated at the end of each iteration in order to determine the global best path, which requires more complexity. If the number of iterations required is Γ, then the number of times this algorithm calculates the entropy is approximately $\binom{\mathbf{N}}{2} + k\Gamma$, giving a complexity of $O(|\mathbf{N}|^2)$.

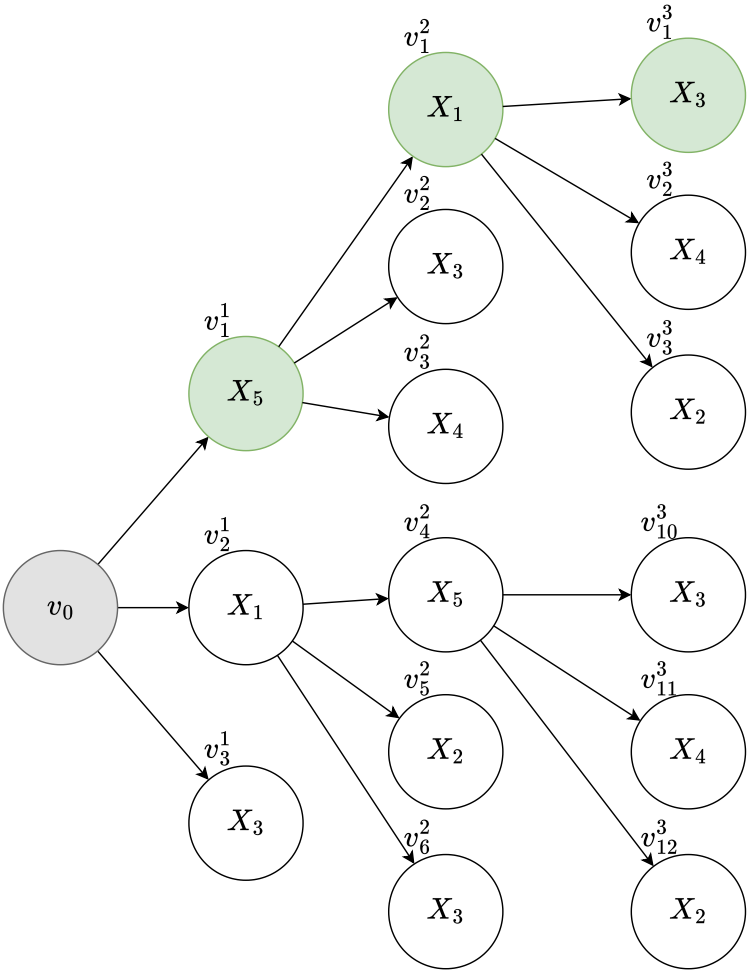


Figure 5.1: The resulting graph after applying the CSP algorithm to the numerical example. The selected path is shown in green

5.6 Numerical Example

To demonstrate how the CSP algorithm and its variants function, the algorithm is applied to an example. Consider a system $|\mathbf{N}| = 5$ files, $\gamma = 3$, where the correlations between nodes $\{X_1, X_2, \ldots, X_5\}$ are represented as MIAs as given in Table 5.1. The CSP algorithm is applied to the above example, with $t = 3$ and $m = 2$. Two variants are also applied, the CSP-CP and CSP-CP with AG=2 (i.e. 2 guaranteed ancestors). The results are shown in Figures 5.1-5.3.

The first step, which is the same for all algorithms including the GISGEM algorithm, involves calculating the individual entropy as $H(X_i)$ for $i \in \{1, 2, \ldots, 5\}$. For example, the value of $H(X_1) = 169$ bits is determined by summing the MIA values that have a binary representation `1XXXX`, while $H(X_1, X_2) = 250$ bits is obtained

Table 5.1: An example of MIA values for five entropy sources

Index	Binary Representation	MIA	Value
1	00001	$I(X_E\|X_A;X_B;X_C;X_D)$	11
2	00010	$I(X_D\|X_A;X_B;X_C;X_E)$	14
3	00011	$I(X_D;X_E\|X_A;X_B;X_C)$	18
4	00100	$I(X_C\|X_A;X_B;X_D;X_E)$	20
5	00101	$I(X_C;X_E\|X_A;X_B;X_D)$	11
6	00110	$I(X_C;X_D\|X_A;X_B;X_E)$	3
7	00111	$I(X_C;X_D;X_E\|X_A;X_B)$	3
8	01000	$I(X_B\|X_A;X_C;X_D;X_E)$	6
9	01001	$I(X_B;X_E\|X_A;X_C;X_D)$	17
10	01010	$I(X_B;X_D\|X_A;X_C;X_E)$	6
11	01011	$I(X_B;X_D;X_E\|X_A;X_C)$	17
12	01100	$I(X_B;X_C\|X_A;X_D;X_E)$	5
13	01101	$I(X_B;X_C;X_E\|X_A;X_D)$	19
14	01110	$I(X_B;X_C;X_D\|X_A;X_E)$	7
15	01111	$I(X_B;X_C;X_D;X_E\|X_A)$	4
16	10000	$I(X_A\|X_B;X_C;X_D;X_E)$	6
17	10001	$I(X_A;X_E\|X_B;X_C;X_D)$	13
18	10010	$I(X_A;X_D\|X_B;X_C;X_E)$	10
19	10011	$I(X_A;X_D;X_E\|X_B;X_C)$	8
20	10100	$I(X_A;X_C\|X_B;X_D;X_E)$	17
21	10101	$I(X_A;X_C;X_E\|X_B;X_D)$	12
22	10110	$I(X_A;X_C;X_D\|X_B;X_E)$	11
23	10111	$I(X_A;X_C;X_D;X_E\|X_B)$	19
24	11000	$I(X_A;X_B\|X_C;X_D;X_E)$	6
25	11001	$I(X_A;X_B;X_E\|X_C;X_D)$	16
26	11010	$I(X_A;X_B;X_D\|X_C;X_E)$	16
27	11011	$I(X_A;X_B;X_D;X_E\|X_C)$	8
28	11100	$I(X_A;X_B;X_C\|X_D;X_E)$	12
29	11101	$I(X_A;X_B;X_C;X_E\|X_D)$	2
30	11110	$I(X_A;X_B;X_C;X_D\|X_E)$	2
31	11111	$I(X_A;X_B;X_C;X_D;X_E)$	11

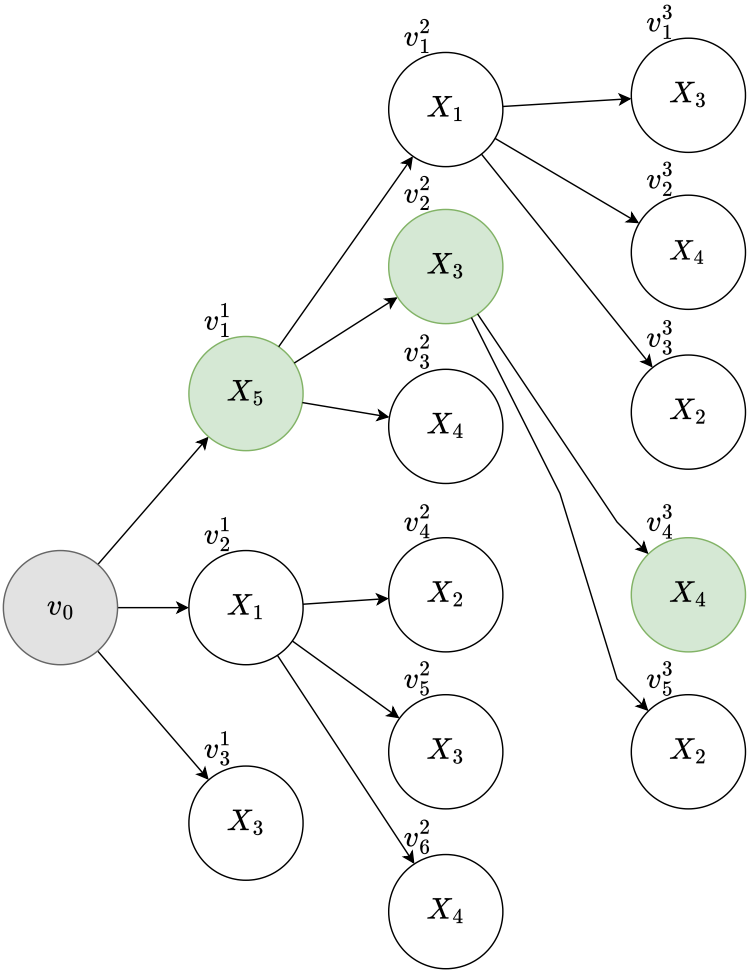


Figure 5.2: The resulting graph after applying the CSP-CP algorithm to the numerical example. The selected path is shown in green

by summing MIA values with binary matching `11XXX`, `10XXX` and `01XXX` (where the value of `X` can be a `0` or `1`). Since $m = 2$, instead of only X_5 being chosen in the regular GISGEM approach, X_1 is also selected by all the algorithms. The second iteration is where the algorithms start to diverge from one another. First, as opposed to the standard GISGEM procedure, two different calculations are carried out: $H(X_i|X_5)$ and $H(X_j|X_1)$. In the standard CSP approach, $X_i \in \{X_1, X_2, X_3, X_4\}$ and $X_j \in \{X_2, X_3, X_4, X_5\}$. However, the CSP-CP variant reduces the choices in the second entropy calculation to prevent X_5 from being chosen, meaning that $X_j \in \{X_2, X_3, X_4\}$. In the plain CSP version, vertices v_1^2 and v_4^2 are chosen. However, it is clear that $\mathbf{P}_{v_1^2}$ and $\mathbf{P}_{v_4^2}$ are identical, only the order is different. Although the CSP-CP variant does not have this issue, it chooses vertices v_1^2 and v_2^2, which means that the paths stemming from v_2^1 are lost in subsequent iterations. To remedy this issue, the AG=2 addition guarantees that the two initial ancestors (v_1^1 and v_2^1) both

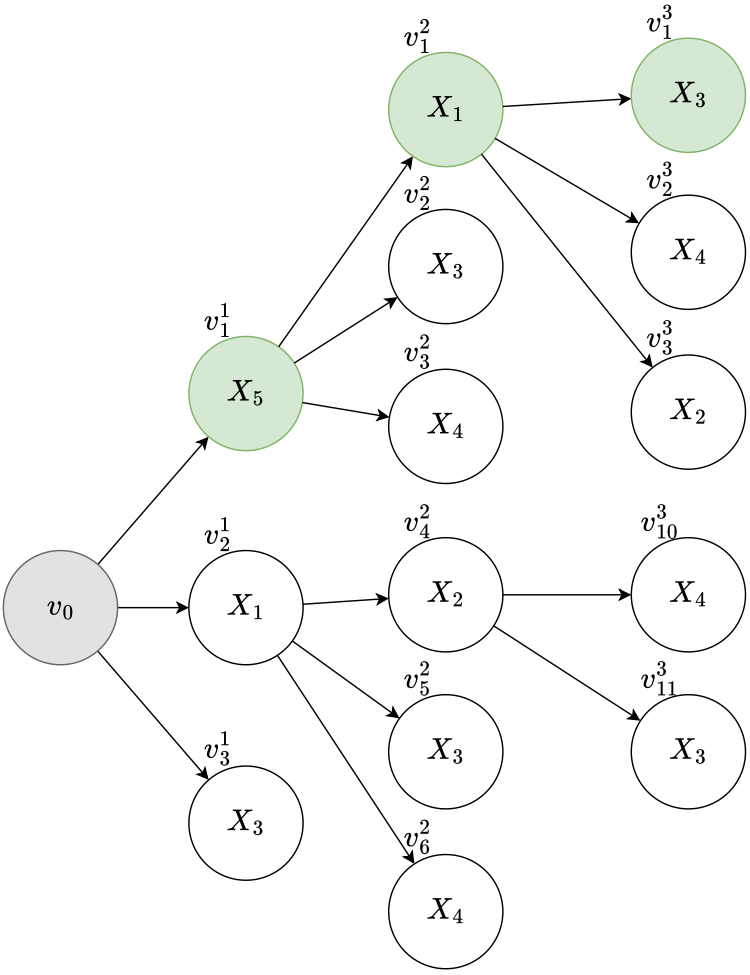


Figure 5.3: The resulting graph after applying the CSP-CP algorithm to the numerical example with AG=2. The selected path is shown in green

have paths in the final iteration. Since the CSP-CP variant restricts the number of nodes available for selection, only two vertices are available to choose from the children of v_2^2, since X_5 and X_3 are already selected, and X_1 is excluded based on the CSP-CP rule. So too, based on the branch and bound conditions described in Section 5.3.3, v_6^2 is automatically pruned from the tree without considering it since, from the initial set $\mathbf{N}$, X_1 was chosen in the previous iteration and $\{X_5, X_2, X_3\}$ are excluded from the set based on the CSP-CP rule. Thus, after selecting X_4 for v_6^2, there are no more nodes available to choose from. So too, if v_3^1 was chosen in the first iteration, it would automatically terminate with the path $\{X_3, X_2, X_4\}$.

After the algorithms are applied, the CSP algorithm recommends the path $f^{\mathcal{X}}(\mathbf{P}_{v_1^3}) = \{X_5, X_1, X_3\}$, the CSP-CP algorithm yields $f^{\mathcal{X}}(\mathbf{P}_{v_4^3}) = \{X_5, X_3, X_4\}$ and the CSP-CP with AG=2 gives the same result as the CSP algorithm. If these combinations are evaluated, $H(\mathbf{P}_{v_1^3}) = 304$ bits and $H(\mathbf{P}_{v_4^3}) = 312$ bits, meaning that the CSP-CP

algorithm found a better result in this example. It should be noted that the GISGEM algorithm also suggests the set $\{X_5, X_1, X_3\}$. However, it did not consider as many options as the other algorithms. So too, although the CSP and CSP-CP with AG=2 algorithms produced the same result, the CSP algorithm only produced 3 unique paths in the final selection, whereas the CSP-CP with AG=2 algorithm has 5. As proved in Lemma 5, the CSP-CP with AG=2 version guarantees that it will find a combination at least as good as the GISGEM approach. The reason for this is that, since two ancestors' paths were guaranteed, and only two paths are chosen at each step, the breadth of the search space was constrained for the CSP-CP with AG=2 variant and it was not able to find a better result than the plain CSP. In contrast to this, although the CSP-CP algorithm without ancestor guarantee found a better result in this example, it is not always guaranteed to do so.

To show how the optimality of a result is tested, the result of the CSP algorithm is used. Obviously, node X_3 maximises $H(X_i|X_5, X_1)$, since it was chosen last in the path $\mathbf{P}_{v_1^3}$ and the algorithm is always optimal in a single iteration. So too, X_5 is found to maximise $H(X_i|X_1, X_3)$. However, when determining the maximum for $H(X_i|X_5, X_3)$, X_4 is suggested. Swapping X_1 and X_4 gives the same result as the CSP-CP variant. This was determined to be optimal for this example through an exhaustive search. In fact, for this example, the set $\{X_3, X_4, X_5\}$ is the only set that passes the optimality test, meaning that the iterative optimality test will always move towards the optimal result regardless of the initial set given to it. However, this is not always the case for larger $|\mathbf{N}|$, since a local maximum could be found instead.

5.7 Simulation Results

This section outlines the practical experiments performed on the different approaches. First, the new algorithms are tested to find the optimal configurations of the different input variables. Then, the tuned algorithms are compared to one another and the algorithms from Chapter 4. All experiments were performed using Matlab on a MacBook Air (2013 model) with 1.3 GHz Dual-Core Intel Core i5 processor and 4 GB of 1600 MHz DDR3 RAM.

5.7.1 Variable Tuning

The different approaches have various variables that affect the quality of the results produced as well as the time complexity. The algorithms were implemented and tested

in Matlab and were configured with all combinations of the tuning variables within certain ranges. To ensure consistency in the results, five unique MIA configurations were tested for each combination of parameter. The results were then averaged to determine their individual effect on the performance and time complexity of the algorithms.

5.7.1.1 CSP Results

In the CSP approach, m and t control the breadth of the search algorithm, the CSP-AG variant prevents the algorithm from diversifying too much and the CSP-CP variant controls how many unique results are compared. To test the effect of each of these parameters, $|\mathbf{N}|$ and γ were chosen to be 16 and 8 respectively, as this provides the greatest number of possible combinations ($\binom{16}{8} = 12870$). The values of m and t were in the range of 1 to 10, while the number of guaranteed ancestors was varied between 0 and $\min(10, m)$. The reason for this is that the number of guaranteed ancestors reserves some the available m paths. Thus, it cannot be greater than m.

The results for the performance are shown in Figure 5.4 while the time complexity results are shown in Figure 5.5. For both figures, the GISGEM performance is shown as a benchmark. In terms of performance, it is clear that the CSP-CP variant performed worse than the plain CSP algorithm across the different tuning variables. This is a somewhat surprising result, since the CSP-CP variant is supposed to prevent paths from converging. It may be argued that, by restricting the pool of nodes to choose for the other paths, those paths are not able to find a better solution and are not able to survive long enough to improve their position. At the same time, the CSP-CP variant is better overall in terms of time complexity. This is owing to the ability of the CSP-CP variant to terminate paths early based on the branch and bound considerations discussed in Section 5.3.3.

Another notable result is that all variants performed equal to or better than the GISGEM algorithm, with the exception of the CSP-CP variant when the number of guaranteed ancestors is 0. As mentioned in Section 5.3.2, the CSP-AG variant is specifically introduced to ensure that the algorithm will perform at least as well as the GISGEM algorithm. This result confirms the hypothesis. So too, on average, the improved algorithms have a greater time complexity than the GISGEM approach as expected.

Comparing the trade-off between quality of results and time complexity, it is clear to

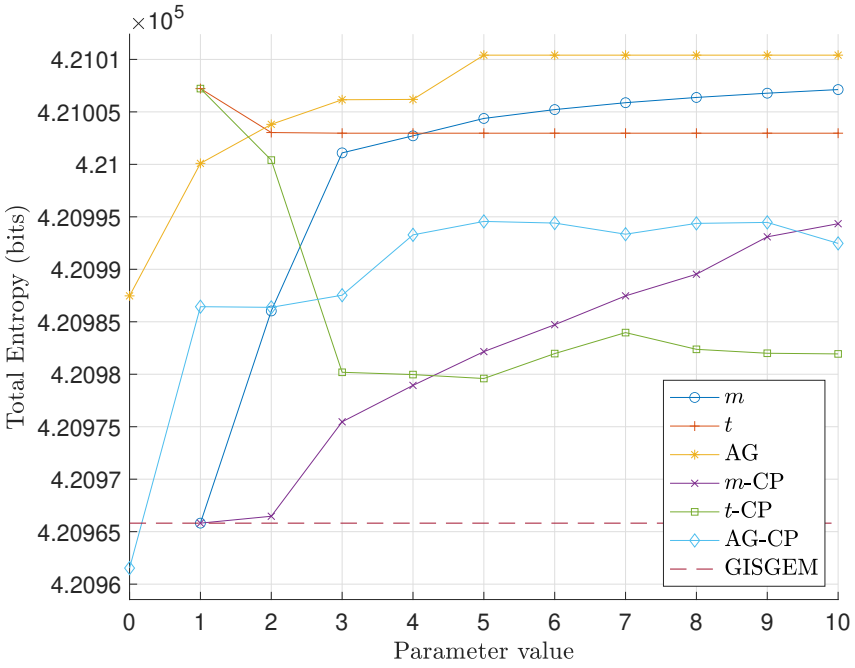


Figure 5.4: A comparison of the different variables and variants for the CSP algorithm and their effect on performance. $|\mathbf{N}| = 16$ files, $\gamma = 8$

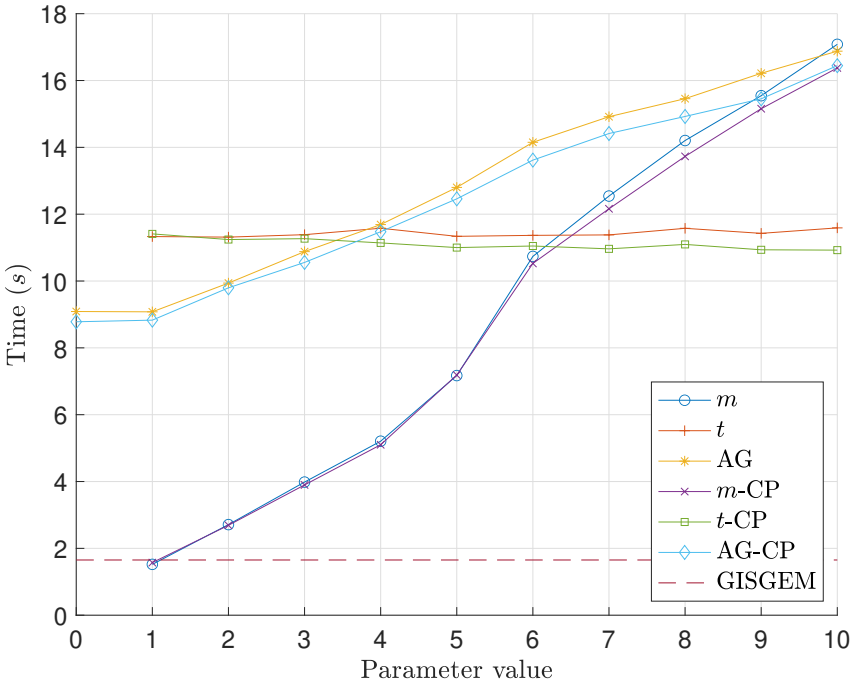


Figure 5.5: A comparison of the different variables and variants for the CSP algorithm and their effect on time complexity. $|\mathbf{N}| = 16$ files, $\gamma = 8$

see that an increase in m has a positive effect on the performance but a negative effect on time complexity. So too, the increase in m improves the result dramatically for the plain CSP algorithm until $m = 3$, after which the improvements are not as marked. Nonetheless, the effect of m on the time complexity is linear. Interestingly, changing the size of t has a negative effect on performance, especially for the CSP-CP variant. At the same time, the effect on time complexity is negligible. A possible reason for this is that increasing the value of t favours a single path that is performing well until this point. At the selection phase, a path that performed well in the previous iteration is more likely to have its children perform better than the other paths. Thus, increasing t means that the algorithm favours a single path instead of diversifying. Finally, increasing the number of guaranteed ancestors has a significant effect on the quality of results. For the plain CSP algorithm, increasing the number of guaranteed ancestors produced the best results performance-wise. So too, although it had a negative effect on the time complexity, it did not have as much of an effect as m.

In Figure 5.6, the results are broken down in terms of different combinations of AG and m. For readability, the minimum value (at $\{\text{AG}, m\} = \{0, 2\}$) is subtracted from the rest of the results. As expected, higher values of AG and m perform better. However, it is interesting to note that setting $\text{AG} = m$ made the average performance go down. This is because, by reserving all of the available paths using AG, the diversity of the search algorithm is limited and so it is not able to find a good result. The best performance is obtained when the value of AG is in the general range of $m/2 < \text{AG} < m$, although there are some outliers (such as $\text{AG} = 1$ for all values of m).

To summarise, increasing AG is the best way to balance performance and time complexity, as long as some of the paths are not reserved. So too, increasing m is favourable for the quality of results, but it should not be increased too much, so as to keep the time complexity low. The value of t should be kept small to avoid providing too much diversity, while the CSP-CP variant performs worse but has a guarantee on the diversity of the results.

5.7.1.2 ACO Results

In the ACO approach, there are a number of variables that could be tuned. For our experiments, this list was limited to the number of ant agents (k), the values of α, β, q_0, ρ and SG. The MNI, τ_0 and ξ were fixed at 200, 10 and 0.1 respectively, as they were experimentally determined to not have a major influence on the results. Owing

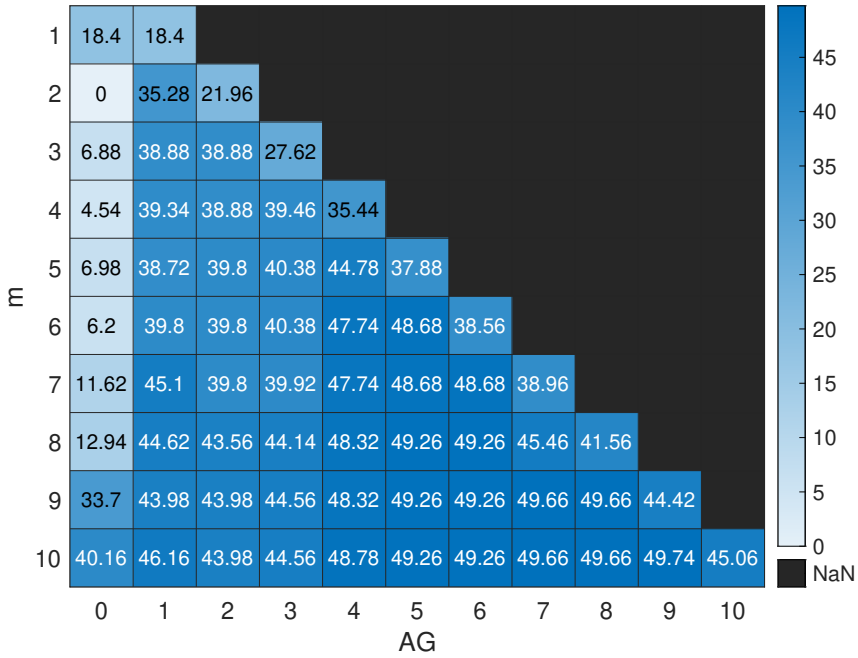


Figure 5.6: A combination-wise comparison of AG and m on the performance of the CSP algorithm

to the large number of tuning variables that were selected for tuning, as well as the time complexity of the ACO approach, it was decided to limit the experiment values to $|\mathbf{N}| = 14$ files and $\gamma = 7$, giving a total of 3432 possible combinations.

The results of the performance and time complexity are depicted in Figures 5.7 and 5.8 respectively. The ranges of the tested variables and a description of their effect on performance and time complexity are shown in Table 5.2. Since the ranges are vastly different, the performance and time complexity effects are shown in a box-and-whisker plot. It was found that the values of k and SG have a linear and proportional effect on performance and time, meaning a higher value resulted in better performance and increased time. The other variables did not exhibit a specific trend, but rather had value combinations that performed better than others. The results support the complexity predicted in Section 5.5, since increasing k negatively affects the time complexity. The SG does not linearly affect the complexity, since it only controls how many iterations the algorithm should wait for if the result is not improved. Accordingly, increasing this value has more of an effect on the time complexity.

In terms of the most optimal values for the different parameters, it was experimentally determined that $q_0 = 0.1$ and $\rho = 0.2$ consistently produced the best results. However, the values of α and β did not show specific trends. Nevertheless, the combination

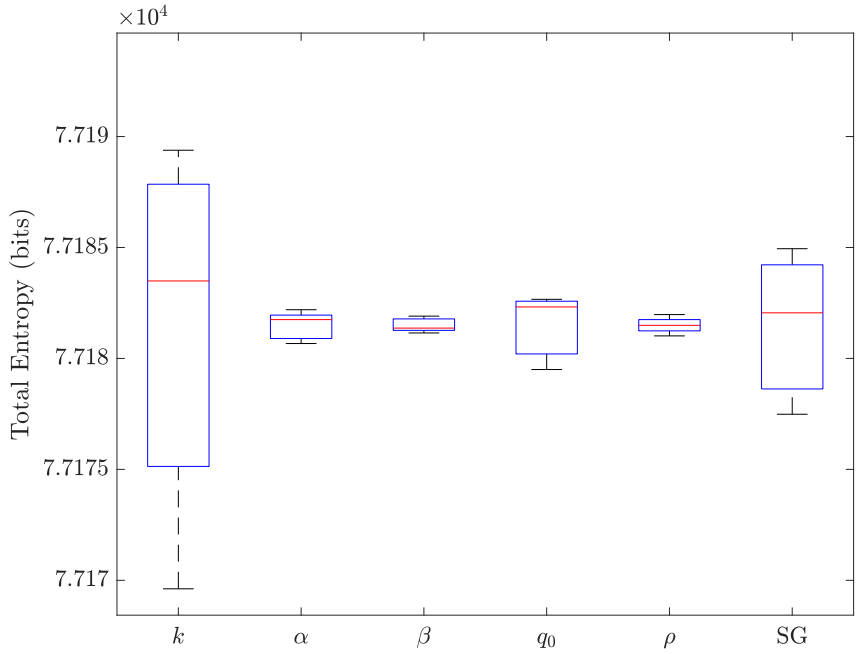


Figure 5.7: A comparison of the different variables for the ACO approach and their effect on performance. $|\mathbf{N}| = 14$ files, $\gamma = 7$

Table 5.2: The range of values tested for the tuning variables in the ACO approach and their effect on performance and time complexity

Variable	Values Tested	Performance Impact	Time Impact
k	5, 10, 15, 20	Major	Moderate
α	0.3, 0.6, 1, 3, 5	Minor	Minimal
β	0.3, 0.6, 1, 3, 5	Minor	Minimal
q_0	0.1, 0.5, 0.9	Moderate	Minimal
ρ	0.2, 0.4, 0.6, 0.8	Minor	Minimal
SG	10, 15, 20	Moderate	Major

$\{\alpha, \beta\} = \{1, 0.6\}$ featured most regularly in the best performing configurations.

5.7.2 CSP and ACO Comparison Results

The different algorithms were compared with one another using the best values determined in Section 5.7.1. In this experiment, $|\mathbf{N}| = 18$ files and $\gamma = \{1, 2, \ldots, 17\}$. The GA approach from Chapter 4 was used as the benchmark and 10 different MIA configurations were used. The datasets were set up to be distinct from one another by randomising the values of the MIAs and changing the percentage of MIAs that are set to 0, effectively controlling the density of the correlation structure. Nevertheless,

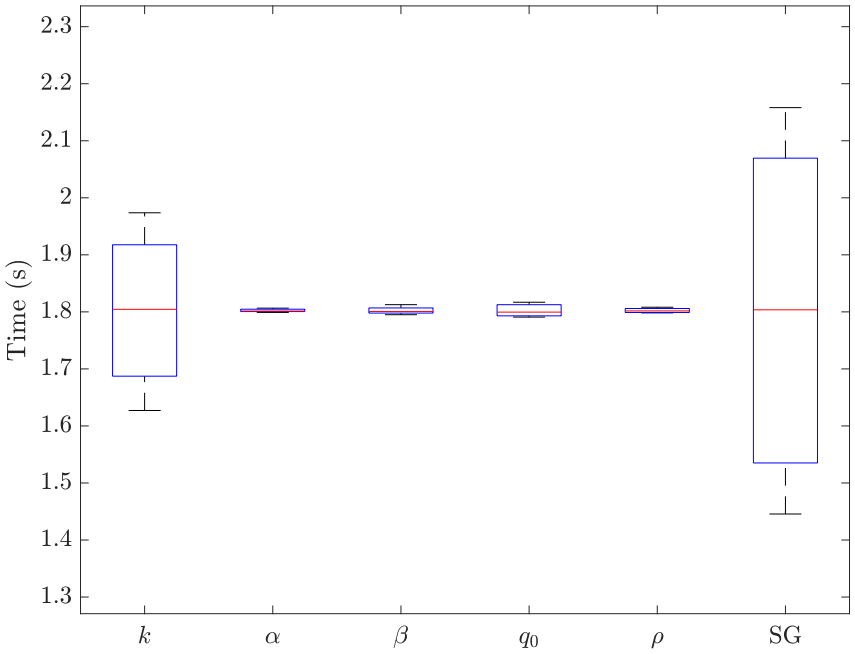


Figure 5.8: A comparison of the different variables for the ACO approach and their effect on time complexity. $|\mathbf{N}| = 14$ files, $\gamma = 7$

Table 5.3: The algorithms used in the algorithm comparison test, together with their associated tuned variables

Algorithm	Variable Values
GISGEM	$t = 1$, $m = 1$, AG $= 0$
CSP	$t = 5$, $m = 5$, AG $= 3$
CSP-CP	$t = 5$, $m = 5$, AG $= 3$
ACO	$k = 20$, $\alpha = 1$, $\beta = 0.6$, $q_0 = 0.1$, $\rho = 0.2$, $\tau_0 = 0$, $\xi = 0.1$, SG $= 20$, MNI $= 200$
GA	$\|\mathbf{P}\| = 20$, $\alpha_e = 0.2$, $\alpha_c = 0.6$, SG $= 15$, MNI $= 200$

the diversity of the MIAs was set to be diffuse, meaning that an optimal solution is not clear or distinct from the other solutions. This was done in order to test the algorithms in a more challenging environment and more clearly demonstrate their performance relative to one another. In contrast to this, a correlation structure with distinct clusterings would mean that all of the algorithms would be likely to converge on the optimal result. Table 5.3 lists the algorithms tested as well as the variable values used for each one.

Figure 5.9 shows the performance of each of the algorithms with respect to the GA results. For $\gamma > 10$, the differences between the algorithms becomes less significant. Surprisingly, the CSP-CP algorithm performs stronger than expected, with much higher performance for the clusters that it found as compared to the other algorithms.

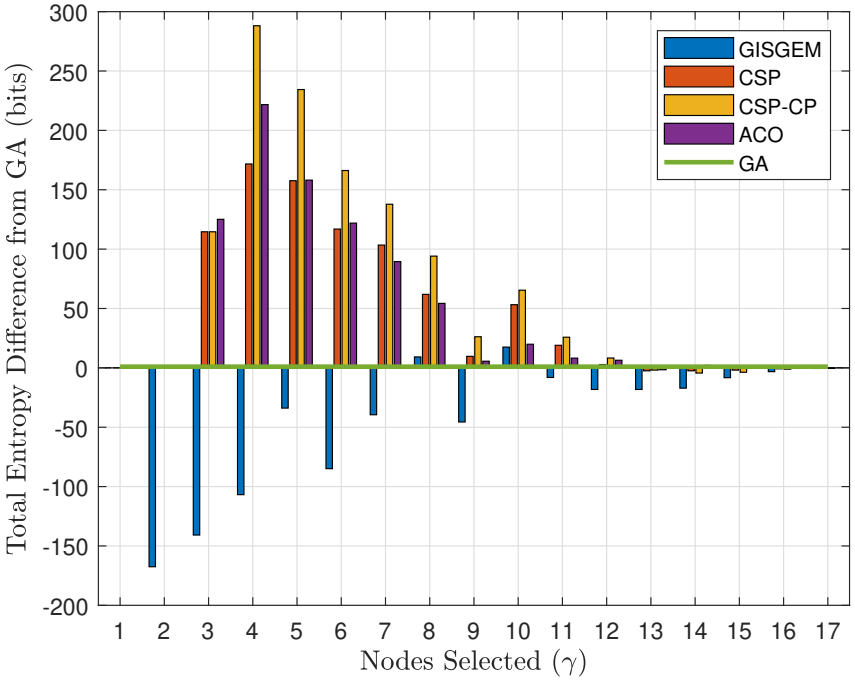


Figure 5.9: Performance results for each of the algorithms, as compared to the GA. Higher values correspond to better performance

Figure 5.10 reveals that the new algorithms come with an associated increase in time complexity. The ACO approach is 1.8 times slower on average than the GA, while the CSP and CSP-CP are 1.55 and 1.22 times slower on average. When comparing the times of the CSP and GISGEM approaches, it is found that the CSP is 4.17 times slower than GISGEM on average. According to Section 5.5, $O_{\text{CSP}}/O_{\text{GISGEM}} = 4.19$ on average, for the same range of γ (when considering the total number of iterations). This confirms that the complexity prediction is correct.

For the range $\gamma < 9$, in which the difference between the algorithms' results are significant, the CSP and CSP-CP algorithms are 2.5 times faster than the ACO approach on average and 1.5 times faster than the GA approach. One interesting result is that the CSP-CP algorithm's time complexity decreases for $\gamma > 13$, since it is able to use the branch and bound condition in Lemma 8 to terminate the search paths prematurely. Consequently, it has a better time complexity than the plain CSP. For $\gamma \leq 13$ however, its time performance is comparable.

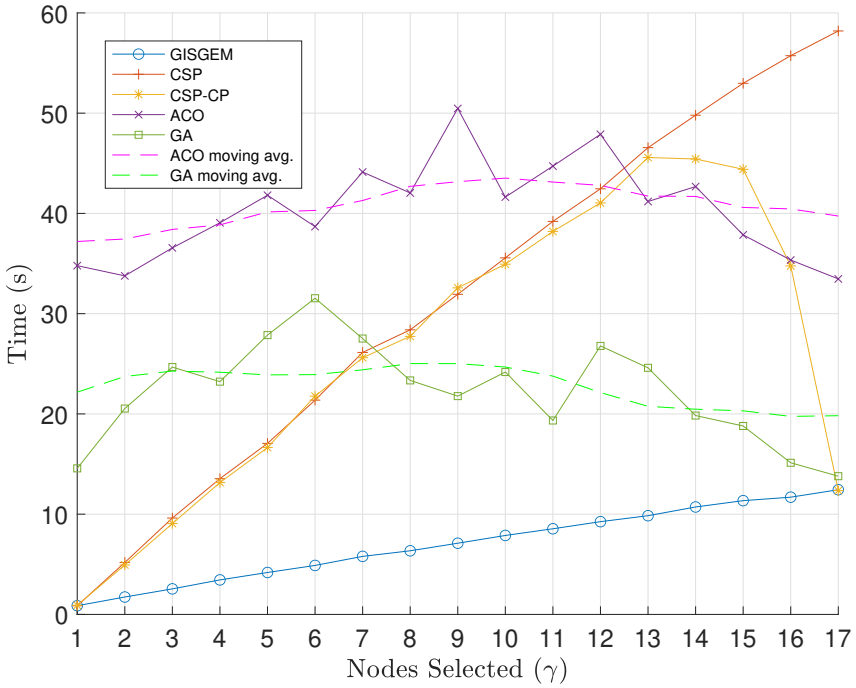


Figure 5.10: Time complexity for each of the algorithms

5.8 Discussion

All of the algorithms are found to converge, based on the empirical evidence. The two meta-heuristic approaches have an SG variable, which determines how many iterations to wait, in which the result does not change, before terminating. This value was tuned so that the algorithms converge in a reasonable time, as shown in Figure 5.10. The iterative algorithms are guaranteed to converge, since there are a predetermined number of steps to perform, based directly on the value of γ. The same applies to the IOI approach, which is shown to converge based on (5.12) in Corollary 1. To further compare the performance of the different algorithms, an exhaustive search was performed for all values of γ and the 10 different datasets to obtain the performance of every possible combination of clustering. The performance of each cluster found by the algorithms was then compared to this list, allowing for each algorithm to be ranked according to the percentile in which it fell. In addition, three applications of the IOI approach are included in this experiment. In the first, a random cluster is chosen, which is iteratively improved by the IOI algorithm until it converges. The other two approaches involve improving the clusters found using the CSP and CSP-CP algorithms with IOI respectively. These results are summarised in Table 5.4, together with the average time and the average of the performance values from Figure 5.9.

The worst performing algorithm is the GISGEM approach, which still produces clusters that perform higher than the 98th percentile. However, it is clear that the new approaches discussed in this chapter outperform the GISGEM approach, with a greater time complexity. It is interesting to note that, although the CSP-CP approach had a higher average bit performance than the GA, the average percentile that its results fell in were lower. On closer inspection, it is discovered that this is due to strong performance for lower values of γ, which are offset by comparatively worse performance for higher values. This is directly affected by the CSP-CP rule, which removes nodes from selection, potentially stymieing the algorithm from finding better results. Additionally, the bit difference between clusters arises from the difference in entropy, which is linked to the values in the MIAs for the nodes selected. It is possible therefore that a single change of node could result in a more substantial change in entropy if the MIAs associated with the exchanged nodes have a large differential. This shows the limitations of using the average bit values alone when determining relative performance. However, with larger values of $|\mathbf{N}|$, the exhaustive search becomes prohibitively complex, requiring a greater reliance on relative bit performance to determine the performance of a clustering algorithm.

The ACO meta-heuristic performs strongly, producing the best performing clusters on average than all of the algorithms tested without the addition of IOI. It is almost twice as slow as the GA meta-heuristic, yet it appears to be more suited to this optimisation problem than the GA in terms of quality of results.

The CSP-CP limited version in particular is able to reduce its complexity to close to the GA, yet the quality of its results are only slightly worse.

In addition, the CSP approach and its variants provide various parameters that are able to tune the trade-off between complexity and quality in a more controlled and predictable way than the meta-heuristic approaches. Thus, although the performance is comparable or slightly worse than the meta-heuristic approaches, the advantages listed above make them more attractive when solving this particular optimisation problem. Furthermore, from the more specific results discussed in Section 5.7.2, the algorithms have specific regimes in which they perform better than others.

Applying the IOI approach to a random initial clustering produced results with slightly better performance than the GA (the small difference in average bit performance does result in a small difference in percentile averages, but they are within rounding distance for the given precision in Table 5.4) with approximately the same time complexity. However, this negates the predictable nature of the iterative approaches,

Table 5.4: A summary of the performance of the different algorithms (algorithms presented in this chapter in bold)

Algorithm	Avg. Performance vs. GA (bits)	Avg. Percentile	Avg. Time (s)
GISGEM	−39.17	98.11	6.98
CSP	**47.24**	**99.75**	**31.45**
CSP-CP	**67.61**	**99.54**	**26.42**
ACO	**47.71**	**99.92**	**40.36**
GA	N/A	99.81	22.20
IOI	**17.67**	**99.81**	**21.43**
CSP with IOI	**67.47**	**99.94**	**40.93**
CSP-CP with IOI	**81.21**	**99.97**	**37.08**

since the performance is governed by the initial random selection of a cluster.

Significantly, applying the IOI to CSP and CSP-CP provides a considerable improvement to the results, yielding the best performance of all of the tested algorithms. In particular, when IOI is applied to CSP-CP, the resulting clusters are very close to optimal. This comes with a commensurate increase in time complexity of 9.48 and 10.66 seconds respectively. Nevertheless, the time complexity remains close to that of the ACO, with much improved results. The IOI can be applied to any result, which makes it an effective way to further refine and improve the results from any method used. It appears from the data that the CSP-CP approach finds clusters that are more amenable to improvement using the IOI as compared to the plain CSP approach. With the improved time complexity, this means that the CSP-CP with IOI approach is a very suitable method for solving the single group clustering optimisation problem in a hybrid SW/CC system.

5.9 Conclusion

Two new algorithms are presented in this chapter, that increase the quality of the results when partitioning a library of files to be used in a hybrid SW/CC content delivery system. This is achieved by creating a tree-based model and adopting a path-based view of finding solutions in the search space, resulting in the CSP algorithm. A trade-off between quality and time complexity is enabled using this approach, giving more flexibility than previous iterative and meta-heuristic algorithms used to solve this problem. Additionally, a necessary but not sufficient condition is derived for the optimality of a given result. This is used to further refine and improve the quality

of the results from any of the different approaches. The algorithms perform well when compared to benchmark meta-heuristic approaches. This chapter builds and improves on the algorithm in Chapter 4, but more research is required to extend and expand the algorithms to multiple groupings in order to further improve the performance of the SW/CC system.

Chapter 6

Trellis-Based Multi-Group Optimisation

The work presented thus far is now extended to the multi-group scenario. The complexity of this problem, when the objectives are updated, have been shown to be an increase compared to the single group case. As such, the previous methods used in this research are not sufficient to solve for this new problem. A novel trellis-based approach is designed and employed instead, which is based on the understanding from the single group clustering problem. The optimality checks and other theorems from the previous chapters are adapted as well. The GA is more optimally suited to this type of problem than the ACO and is used as a benchmark, with results falling in the 99th percentile. The proposed methods create groupings that fall between the 97th and 99th percentile, but are able to achieve this with significantly lower time complexity than the GA.

6.1 Introduction

Until now, this research has examined selecting a single subset of nodes to optimise a hybrid SW/CC system. Specifically, it has examined dividing the nodes into two groups, one coded and the other uncoded. It has been shown that a greedy and iterative approach is suitable to this type of problem. Now, the multi-group case is considered, which represents a much more real-world scenario than the single group optimisation. Nevertheless, this brings with it its own set of challenges. The existing approaches and theorems need to be re-examined and adapted or redesigned according to the new challenges of this regime.

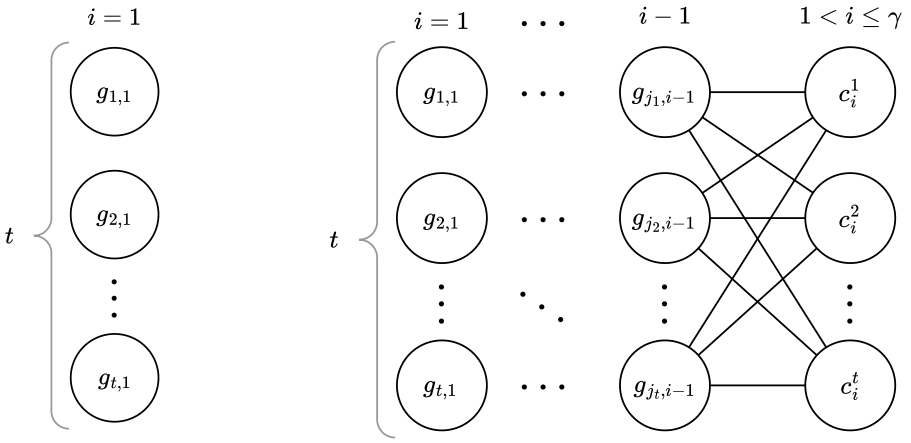
The new approach is derived from the algorithms in Chapters 4 and 5. It is noted that the CSP algorithm is able to find multiple paths that potentially maximise the total entropy by following two main steps. First, it identifies candidate nodes that can be added to the tree based on certain criteria based on the entropy performance. Then, only a subset of the nodes are assigned, with the size of the subset determined beforehand. The CSP-CP approach limits the pool of candidate nodes in order to prevent convergence between the paths, while the CSP-AG approach guarantees that a certain number of starting nodes will have a path in the final step.

Based on these observations, the method to extend this approach to the case at hand becomes apparent. First, since the optimisation problem for the multi-group consideration is to minimise the total entropy as opposed to the maximisation problem in the single group case, all of the operations used to determine the best nodes in the GISGEM and CSP algorithms should be changed to minimise instead of maximise. Next, it is clear to see that if the pool of nodes is limited as in the CSP-CP algorithm, such that paths can only choose from a pool of nodes that excludes *all* nodes selected thus far (including nodes from other paths), then it is guaranteed that all paths will be unique. In addition, adding the requirement of the CSP-AG approach, it can be guaranteed that every starting node has a path. Putting these ideas together, it can thus be guaranteed that t unique paths can be generated using this approach. However, it is still unclear as to how to adapt the two stage process of the CSP approaches to the multi-group scenario.

The remainder of this chapter is arranged as follows: Section 6.2 outlines the new algorithm to solve the multi-group clustering problem and updates the optimisation criteria for this new regime, while Section 6.3 modifies the GA algorithm so that it can solve the multi-group version. Section 6.4 analyses the approaches in terms of complexity and Section 6.5 explains the new iterative algorithm step-by-step by means of a numerical example. The testing and results of the proposed algorithms are given in Section 6.6 and are discussed in more depth in Section 6.7. Section 6.8 provides concluding thoughts.

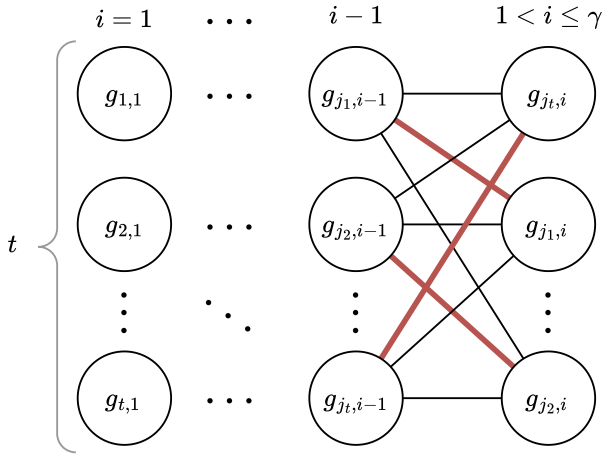
6.2 Trellis-Based Algorithm

The following subsections formalise the approach to solve the multi-group clustering problem using a trellis-based approach and provide more insight into how the concepts presented thus far are adapted. This algorithm is named the Trellis-Based Multi-group Allocation Procedure (TreB-MAP) algorithm.



(a) The INS Phase

(b) The CNS Phase



(c) The CNA Phase

Figure 6.1: A summary of the three phases in the TreB-MAP algorithm

The algorithm consists of three stages. First is the Initial Node Selection (INS), which chooses the first nodes, one to go into each of the t groups. Then, the algorithm alternates between Candidate Node Selection (CNS) and Candidate Node Assignment (CNA) for the rest of the $\gamma - 1$ iterations, until all of the nodes have been assigned a group. The algorithm is summarised in Figure 6.1 and the different techniques for each of the phases are outlined in the following subsections.

6.2.1 Initial Node Selection

The first stage of the trellis-based selection involves choosing the first t nodes, which will be added to different groups. In this step, the first column of group elements $\mathbf{g}_{:,1} = \{g_{1,1}, g_{2,1}, \ldots, g_{t,1}\}$ are assigned, which form the start of the t different groups.

The goal of this step is to set up the groups with files, such that the rest of the files can be assigned to produce a well-performing clustering. In the single group problem, the algorithm simply chooses the optimal file (in the case of the GISGEM algorithm) or the m best files (for the CSP algorithm) to add to the group in the first step. However, the multi-group problem requires assigning t files in the first step for a single grouping, meaning that this stage has more importance than the single group problem. The general approach of this phase is shown in Figure 6.1(a), and four different procedures are now presented for this stage.

Procedure 3 (Entropy Minimisation INS). Assign the group elements in $\mathbf{g}_{:,1}$, such that:

$$H(g_{i,1}) \leq H(g_{j,1}), \quad \forall i, j \in \{1, 2, \ldots, t\}, i < j \tag{6.1}$$

Since the entropy is calculated for individual entropy sources, entropies at this stage are not dependent on any other nodes. As a result, using this procedure guarantees that the sum of entropies $\sum_{i=1}^{t} H(g_{i,1})$ is minimised. This is a similar procedure to the first assignment in the CSP algorithm, given in (5.3), with the comparison operator reversed to account for the new optimisation objective.

Procedure 4 (Entropy Maximisation INS). Assign the group elements in $\mathbf{g}_{:,1}$, such that:

$$H(g_{i,1}) \geq H(g_{j,1}), \quad \forall i, j \in \{1, 2, \ldots, t\}, i < j \tag{6.2}$$

Although this finds the maximum entropy sum in this iteration, meaning it is the worst performing configuration at this point, the intention is to find entropy sources that are least likely to be put in the same group for the optimal grouping, since this would lead to a potentially unbalanced group.

Procedures 3 and 4 focus on finding the nodes based on their individual performance. In the numerical example in Section 4.5.1, all files are assumed to be equal in size. Consequently, using the procedures above would not differentiate between the nodes and the initial assignment would become a random one. Another approach is to look at the interaction between the files in terms of correlation. Files that have less

correlation between them are more likely to be in different groups in the optimal clustering, since the goal is to minimise the total entropy. Ideally, the entropy of the files would be evaluated conditioned on every combination of the other files to determine which are the most correlated, but this becomes impractical with larger values of $|\mathbf{N}|$. Thus, the following two procedures are presented to approximate this approach.

Procedure 5 (Final Entropy Maximisation). Assign the group elements in $\mathbf{g}_{:,1}$, such that:

$$H(g_{i,1}|\mathbf{N} \setminus g_{i,1}) \geq H(g_{j,1}|\mathbf{N} \setminus g_{j,1}), \quad \forall i, j \in \{1, 2, \ldots, t\}, i < j \tag{6.3}$$

This procedure is not guaranteed to be optimal in the first step. However, it tries to choose files that are least likely to be optimal if placed in the same group, since their co-operative entropies (the entropy of the file conditioned on all other files) are the smallest.

Procedure 6 (GISGEM INS). In this procedure, the GISGEM algorithm is used to determine $\mathbf{N}_u$ with $|\mathbf{N}_u| = t$. The group elements $\mathbf{g}_{:,1} = \{g_{1,1}, g_{2,1}, \ldots, g_{t,1}\}$ are then assigned the entropy sources in $\mathbf{N}_u$.

As with Procedure 5, the intention is to estimate the co-operative entropy and choose nodes that are least likely to produce an optimal result if placed in the same group. The advantage of this procedure is that it uses the GISGEM algorithm to find these nodes, which finds close to optimal results for a set of nodes which maximise the entropy (implying that they have the lowest co-operative entropy). One disadvantage of this approach is that it is more complex to implement, since the GISGEM algorithm runs in a non-trivial amount of time as compared to the other procedures. Although the CSP algorithms could also be used, this further increases the time complexity.

6.2.2 Candidate Node Selection

This stage is an adaptation of the path-based approach, which determines the pool of nodes to choose from at each iteration. Extending this approach, t nodes are selected as candidates, of which each needs to be assigned to a different group. This phase is depicted in Figure 6.1(b), where t candidate nodes are created that will be assigned,

in the next stage, to the t paths determined in the previous iterations. The candidate nodes are added in a trellis form, meaning that the parent nodes share the same pool of candidates and are connected to all of them in this phase of the algorithm. Stated more formally, the aim of this step is to create a set of candidate nodes $\mathbf{C}_i$ for iteration $1 < i \leq \gamma$, where $\mathbf{C}_i \coloneqq \{c_1^i, c_2^i, \ldots, c_l^i\}$ and $|\mathbf{C}_i| = t$. To prevent adding the same file to multiple groups, all previously chosen files are excluded, meaning that $\mathbf{C}_i \subset \mathbf{N} \setminus \bigcup_{k=1}^{i-1} \mathbf{g}_{:,k}$.

Six different methods are proposed for this stage, divided into two subsets. The first involves a turn-based assignment, in which a single path is selected at each step. This path chooses the best node (i.e. one that minimises the total entropy when added to this path). If the node has already been selected, it will choose the next best node, if available. Otherwise, it goes through its list until an unselected node is found. The next path then gets a turn to add its best unselected node. This approach is called Turn-Based Selection.

The second general approach is to select a single path. This path chooses all the t best candidate nodes from its pool alone. This has the advantage of less time complexity, since only one path needs to calculate the entropy gains for all nodes, whereas the other paths need only calculate the entropy for the t selected nodes instead of all of them. This approach is called the Individual-Based Selection.

Within these two subsets, three methods are identified to select the path. For the Turn-Based Selection, this involves determining the order, whereas the Individual-Based Selection involves choosing a single node. It can be done randomly, from Worst-to-Best (WtB) (in terms of current path performance) or Best-to-Worst (BtW). The different selection approaches are now formalised.

Procedure 7 (Turn-Based CNS). At iteration $i \in \{2, 3, \ldots, \gamma\}$, each of the parent group elements $g_{j,i-1} \in \mathbf{g}_{:,i-1}$ gets a turn to choose a candidate entropy source to add to the children group using a specific order. On its turn, the selected parent determines its candidate children nodes as the set $\mathbf{C}_{g_{j,i-1}} \coloneqq \mathbf{N} \setminus \bigcup_{k=1}^{i-1} \mathbf{g}_{:,k}$, ordered such that:

$$H(c_k|\mathbf{P}_{g_{j,i-1}}) \leq H(c_l|\mathbf{P}_{g_{j,i-1}}), \quad \forall c_k, c_l \in \mathbf{C}_{g_{j,i-1}}, k < l \tag{6.4}$$

Then, the parent iterates through its children set in ascending order and selects the first child $c_k \in \mathbf{C}_{g_{j,i-1}}$, such that:

$$c_k{}^* = \arg \min_{k \in \{1,2,\ldots,|\mathbf{C}_{g_{j,i-1}}|\}} f^{\mathcal{X}}(c_k) \notin \mathbf{C}_i, \tag{6.5}$$
$$c_k \in \mathbf{C}_{g_{j,i-1}}$$

Procedure 8 (Individual CNS). At iteration $i \in \{2, 3, \ldots, \gamma\}$, a single parent group element $g_{j,i-1} \in \mathbf{g}_{:,i-1}$ is selected. In turn, this parent determines its candidate children nodes as the set $\mathbf{C}_{g_{j,i-1}} \coloneqq \mathbf{N} \setminus \bigcup_{k=1}^{i-1} \mathbf{g}_{:,k}$, ordered such that:

$$H(c_k|\mathbf{P}_{g_{j,i-1}}) \leq H(c_l|\mathbf{P}_{g_{j,i-1}}), \tag{6.6}$$
$$\forall c_k, c_l \in \mathbf{C}_{g_{j,i-1}}, k < l$$

Then, the candidate node set is assigned as the best t nodes from this parent's child set, i.e. $\mathbf{C}_i = f^{\mathcal{X}}(\{c_1, c_2, \ldots, c_t\})$, $c_i \in \mathbf{C}_{g_{j,i-1}}$.

Thus, both procedures involve setting the candidate nodes in $\mathbf{C}_i$ to the children nodes of group elements. It is important to note that when the child element $c_k \in \mathbf{C}_{g_{j,i-1}}$ is assigned to the candidate node set $\mathbf{C}_i$ it is denoted as $c_i^l = f^{\mathcal{X}}(c_k)$. Procedure 7 ensures that each parent has a turn to select its best option that has not been selected yet, while Procedure 8 lets one parent group element choose all of the nodes. The latter is less complex since the node ordering only has to be done once, while the other requires t orderings. For both procedures, the order of choosing a group element can be determined using BtW, WtB or random ordering. The BtW ordering is based on the entropy of the group until this iteration, namely that each parent group's entropy $H(\mathbf{P}_{g_{j,i-1}})$, with $g_{j,i-1} \in \mathbf{g}_{:,i-1}$, is ordered from lowest to highest, whereas the WtB ordering is from highest to lowest. If using Procedure 8, only the highest or lowest performing parent group is chosen, respectively.

6.2.3 Candidate Node Assignment

After selecting the candidates, it is necessary to place the nodes such that the grouping is optimal (as in the path-based method). First, it is necessary to define the cost matrix for the selected vertices. Let $\mathcal{S}_i$ be the cost matrix for the candidate node set $\mathbf{C}_i$. In this representation, $\mathcal{S}_i$ is a $t \times t$ matrix, where every row $j \in \{1, 2, \ldots, t\}$ represents a parent path and the columns $k \in \{1, 2, \ldots, t\}$ represent the candidate nodes. Thus, the matrix element $s_{j,k}^i \in \mathcal{S}_i$ represents the cost of the selected path with the chosen candidate node. The cost matrix can thus be formally defined as:

$$s^i_{j,k} = H(c^i_k \cup \mathbf{P}_{g_{j,i-1}}), \qquad (6.7)$$
$$1 < i \leq \gamma, \quad c^i_k \in \mathbf{C}_i, \quad \forall j,k \in \{1,2,\ldots,t\}$$

In the GISGEM and CSP approaches, the candidate assignment problem can be solved optimally for this iteration. For the TreB-MAP approach, the problem involves finding the best node to add to each group, such that the overall entropy sum of the groups is minimised. More formally, the problem is to create a binary $t \times t$ assignment matrix $\mathbf{A}_i$, in which the sum of every row and column is equal to 1. An assignment value is defined as $a^i_{j,k} \in \mathbf{A}_i$, where j and k are the row and column indices in $\mathbf{A}_i$ respectively. A value $a^i_{j,k} = 1$ indicates that the candidate node c^i_k is being assigned to the group $\mathbf{g}_{j,:}$. Figure 6.1(c) shows a valid assignment of the candidate nodes defined in Figure 6.1(b) to the t paths found in previous iterations. The red lines show the assignment choices. As shown, this step involves creating and assigning the group elements for this iteration.

The problem outlined above is identical to the balanced assignment problem, where t workers need to be assigned to t tasks and each worker has an associated cost when assigned to a specific task. The goal is to minimise the total cost when assigning exactly one worker to each task. As a result, this can be solved optimally using the Hungarian method [80], outlined in brief in the following procedure.

Procedure 9 (Hungarian method CNA). The algorithm attempts to find the optimal result in one round. Otherwise, it performs further steps repeatedly until a solution is found.

1. Find the minimum element in each row and subtract it from every element in that row.
2. Find the minimum element in each column and subtract it from every element in each column.
3. Find rows with exactly one unmarked zero. Assign these zeros by circling them. Cross out all other zeros in the columns of the assigned zeros.
4. Find columns with exactly one unmarked zero. Assign these zeros by circling them. Cross out all other zeros in the rows of the assigned zeros.
5. Repeat steps 3 and 4 until all zeros are assigned or crossed out. It is possible that rows/columns still contain more than one unmarked or crossed out zeros. In this case, they should be assigned arbitrarily (multiple optimal results exist).

6. If the number of assigned cells equals the number of rows, then an optimal assignment has been found.
7. Otherwise, mark all the rows which have no assigned zero.
8. For each of the rows marked in the previous step, mark the column of each zero (assigned or unassigned) that appears in that row.
9. For each of the columns marked in the previous step, find all the zeros that are assigned already. Mark these rows.
10. Repeat steps 8 and 9 until no more rows/columns can be marked. Draw a line through every unmarked row or marked column. This ensures that all zeros are covered by the lines drawn.
11. Find the minimum element from the cells that are not covered by a line. Subtract this value from each element not covered by a line. Add this value to elements at the intersection of two lines.
12. Repeat steps 3 to 11 until an optimal assignment has been found.

The optimal assignment matrix returned by this algorithm is $\mathbf{A}_i$. It is then used to modify the cost matrix $\mathcal{S}_i$, in which the location of the assigned zeros show which candidate node to assign to each of the existing paths.

The Hungarian algorithm comes with associated complexity. As a result, a simpler approximation of this method is proposed, based on the Turn-Based CNS in Procedure 7.

Procedure 10 (Turn-Based CNA). Each path $\mathbf{P}_{g_{j,i-1}}$ beginning from each parent element $g_{j,i-1} \in \mathbf{g}_{:,i-1}$ has a turn to select a single candidate node $c_k^i \in \mathbf{C}_i$, such that:

$$c_k^{i\,*} = \arg\min_{c_k^i \in \mathbf{C}_i} H(c_k^i \cup \mathbf{P}_{g_{j,i-1}}) \tag{6.8}$$

This is equivalent to the path $\mathbf{P}_{g_{j,i-1}}$ finding the smallest available cost value in $\mathcal{S}_i$ corresponding to its row (j). The column of this value is then reserved from being chosen by other paths by marking it off.

The CNS and CNA procedures are run for iterations $i \in \{2, 3, \ldots, \gamma\}$. The algorithm is thus guaranteed to converge after γ iterations, since at this point there are t paths, each of size $|\mathbf{P}_{g_{j,\gamma}}| = \gamma$ when beginning with the group elements $g_{j,\gamma} \in \mathbf{g}_{:,\gamma}$. So too,

all files from $\mathbf{N}$ have been used, since the children are only chosen from files that have not been selected by any other path at that point. The groups are also disjoint, because the CNA phase only assigns each candidate node to a unique group. The algorithm as a whole is summarised in Algorithm 5.

Algorithm 5 Trellis-Based Multi-group Allocation Procedure

$\mathbf{g}_{:,1} \leftarrow t$ files $\in \mathbf{N}$ using one INS approach from Procedures 3-6
for $i \in \{2, 3, \ldots, \gamma\}$ **do**
  $\mathbf{C}_i \leftarrow t$ files $\in \mathbf{N} \setminus \bigcup_{j=1}^{i-1} \mathbf{g}_{:,j}$ using one CNS procedure from Procedures 7-8
  $\mathbf{g}_{:,i} \leftarrow t$ files $\in \mathbf{C}_i$, ordered using one CNA approach from Procedures 9-10
end for

6.2.4 Multi-Group Optimality Tests

In Chapter 5, a necessary yet not sufficient test is derived, which allows for a basic check to see if a path is potentially optimal. More potently, it is used to further improve the performance of an existing path by exchanging entropy sources. Thus, it is useful to derive optimality tests for the multi-group case as well. One way to extend this is to use a similar approach to the single group regime. Lemma 9 tests optimality by changing the order in which the entropy sources are selected and seeing if any other sources are suggested instead. The TreB-MAP approach selects t nodes at every iteration and can assign them optimally using the Hungarian method. Thus, the extension is to change the order in which the entropy sources are added to the grouping. This is now more formally defined.

Lemma 11 (multi-group optimality condition). Let $\mathbf{g}_{:,i}^{(p)}$ be a set that contains a set of every permutation of $\mathbf{g}_{:,i}$, implying that $|\mathbf{g}_{:,i}^{(p)}| = t!$. For a given grouping $\mathcal{G}$ to be optimal, it is necessary that, for each column $\mathbf{g}_{:,i} \subset \mathcal{G}$, $i \in \{1, 2, \ldots, \gamma\}$, the given column is the solution to the following optimisation condition:

$$\mathbf{g}_{:,i}^{*} = \arg \min_{\mathbf{g}_{:,i} \in \mathbf{g}_{:,i}^{(p)}} \sum_{j=1}^{t} H(\mathbf{g}_{j,:} \setminus g_{i,j}), \qquad \forall i \in \{1, 2, \ldots, \gamma\} \tag{6.9}$$

Proof. The test in (6.9) is equivalent to the CNA stage of the TreB-MAP algorithm, where a single column of $\mathcal{G}$ is removed and then re-assigned as if it were the final iteration $i = \gamma$. It has been shown that the CNA step of the TreB-MAP algorithm

is optimal when looking at a single iteration, since the Hungarian algorithm is guaranteed to find the most suitable assignment. However, the algorithm is not guaranteed to be optimal over multiple iterations, since this relies on the combinatorial nature of the problem and the TreB-MAP algorithm uses a greedy approach. Thus, if a grouping $\mathcal{G}$ is optimal, every CNA step should be the optimal one if it is selected last. □

Additionally, the extra dimension in this problem, provided by $t > 2$ groups, allows for a further optimality test, based on comparing the individual nodes within the grouping.

Lemma 12 (multi-group individual optimality condition). A grouping $\mathcal{G}$ is potentially optimal if exchanging the entropy sources in any group element with another in a different group does not produce a better result. More formally:

$$H(g_{m,n}|\mathbf{g}_{m,:} \setminus g_{m,n}) + H(g_{x,y}|\mathbf{g}_{x,:} \setminus g_{x,y}) < H(g_{m,n}|\mathbf{g}_{x,:} \setminus g_{x,y}) + H(g_{x,y}|\mathbf{g}_{m,:} \setminus g_{m,n}) \tag{6.10}$$

$$\forall g_{m,n}, g_{x,y} \in \mathcal{G}, \quad m \neq x$$

Proof. Consider two group elements $g_{m,n}$ and $g_{x,y}$ in a grouping $\mathcal{G}$ that are not in the same group, implying that $m \neq x$. Equation (3.23) can be rewritten as:

$$H(\mathcal{G}) = \sum_{i=1}^{t} \sum_{j=1}^{\gamma} H(g_{i,j}|\mathbf{P}_{g_{i,j-1}}) \tag{6.11}$$

Since the order of the terms when calculating the joint entropy is not important, let $g_{m,n}$ and $g_{x,y}$ be the final term in their respective groups. Thus their individual contributions of conditional entropy to their groups can be written as:

$$H(g_{m,n}|\mathbf{P}_{g_{m,n-1}}) = H(g_{m,n}|\mathbf{g}_{m,:} \setminus g_{m,n}) \tag{6.12}$$

$$H(g_{x,y}|\mathbf{P}_{g_{x,y-1}}) = H(g_{x,y}|\mathbf{g}_{x,:} \setminus g_{x,y}) \tag{6.13}$$

As a result, if the nodes for $g_{m,n}$ and $g_{x,y}$ exchanged groups, the only changes in terms of the total entropy of this new grouping $\mathcal{G}'$ would be the values of (6.12) and (6.13). This is because they are the final values in their respective groups, which means that they do not feature in any other terms in the summation of the group. Taking into account the new group assignments, these entropy values are now:

$$H(g_{m,n}|\mathbf{P}_{g_{x,y-1}}) = H(g_{m,n}|\mathbf{g}_{x,:} \setminus g_{x,y}) \tag{6.14}$$

$$H(g_{x,y}|\mathbf{P}_{g_{m,n-1}}) = H(g_{x,y}|\mathbf{g}_{m,:} \setminus g_{m,n}) \tag{6.15}$$

As a result, $\mathcal{G}$ can only be optimal if $H(\mathcal{G}) < H(\mathcal{G}')$. This formulation can be expanded to derive (6.10). □

Adding the column restriction of $n = y$ in (6.10) produces the same test as (6.9).

As with the IOI approach in Chapter 5, the optimality tests in Lemmas 11 and 12 can be continuously applied to a grouping, making changes according to better groupings found until the modified grouping passes the tests. These approaches are called the IOI: Multi-Group (IOI-MG) and Individual IOI: Multi-Group (IIOI-MG) algorithms respectively.

6.3 Updated Genetic Algorithm Approach

In Chapters 4 and 5, two different meta-heuristic algorithms were adapted to solve the optimisation problem in the single-group case. It was found that the ACO approach, which fitted well with the path-based view of the problem, produced better quality results in that regime than the combinatorial-based GA. As discussed in Section 6.2, modifications are needed to adapt the path-based view to the multi-group optimisation problem. Thus, it would seem that the ACO would be better suited to this problem as well. However, it is now shown that the GA method of solving this problem is actually the better one. This is because adapting the CSP approach to the multi-group problem involves generating multiple paths with restrictions on the entropy sources that can be chosen. The ACO algorithm does not allow for this, since the ant agents find a best path individually and do not share information about entropy sources that have already been selected. The only interaction between the ant agents is in influencing other's path-making decisions by increasing the attractiveness of sub-paths by depositing pheromones.

In contrast to this, the GA is more suited to being adapted to solve the multi-group problem in this chapter. This is because the approach of the GA is to look at the interactions between the elements in the genome alone.

The first change to the GA is to adapt the genome structure. Whereas the single-group optimisation used a binary genome, here it is extended to a t-ary genome. At the same time, the length of the genome is still equal to $|\mathbf{N}|$ and each position in the genome represents a single entropy source. The difference is that the value of each position represents the group that it belongs to. Thus, there are $|\mathbf{N}|$ genome positions of value $i \in \{1, 2, \ldots, t\}$. So too, the weight constraints from the single-group case are applicable here. Instead of a simple Hamming weight, it is required that the number of genome positions of a given value is equal to γ for all valid values of i.

The functions for crossover and mutation likewise need to be adapted to fit into the new paradigm. In the crossover function, two genomes are selected from the current population as parent genome sequences $\mathbf{g}_1$ and $\mathbf{g}_2$. The resulting child $\mathbf{g}_3$ should share some of the traits from each of the parents. As in the regular crossover function, a random crossover point is chosen between 1 and $|\mathbf{N}|$. The child inherits all of the values of parent $\mathbf{g}_1$. Then, iterating from the crossover point until the end of the genome, the following procedure is used:

1. If the value of both parents at the current position are the same, then no further action is required and the child inherits this value directly.
2. If the values differ, then it is necessary to inherit the new characteristic from parent $\mathbf{g}_2$. However, it is likely that this would cause the genome to become unbalanced and not conform to the weighting restriction. This is avoided by using the method below:
   (a) Find all locations in the child genome $\mathbf{g}_3$ in which the value is the same as parent $\mathbf{g}_2$ at the current position.
   (b) Choose one location randomly.
   (c) Swap the value at the currently selected point with the chosen location. This ensures that the currently selected position has the value of parent $\mathbf{g}_2$, while also making sure that all groups have the same number of elements.

In the same vein, mutations of a genome are implemented by swapping the genome values. The number of swaps is randomly determined, existing as pairs of values $\{i, j\}$, with the restriction that $i, j \in \{1, 2, \ldots, |\mathbf{N}|\}$ and $i \neq j$. Then, the values at position i are exchanged with those at position j. Using the swap method thus guarantees that the balanced weighting of the genome is maintained.

6.4 Complexity Analysis

As discussed in Section 6.2, the general structure of the TreB-MAP algorithm is based on the CSP path-based algorithm, with $m = t$. As a result, the complexity of this algorithm is similar to the CSP approach, which is determined to be in the order of $O(|\mathbf{N}|)$ in Section 5.5. The different procedures for the INS, CNS and CNA will have some effect on this complexity. In particular, Procedure 6 uses the GISGEM algorithm to select the initial nodes, which also has a complexity in the order of $O(|\mathbf{N}|)$ as calculated in Section 4.4.1. Using this procedure thus causes the TreB-MAP algorithm to run twice as slow. Nevertheless, the algorithm still runs in polynomial time.

Although Procedure 9 uses the Hungarian algorithm, which is known to run in $O(t^2)$ time at best and $O(t^3)$ at worst, the algorithm runs using pre-determined entropy calculations and does not need to recalculate the entropies. As mentioned in Section 4.4.1, the complexity incurred by sorting or otherwise manipulating entropy values is much lower than the complexity of calculating the entropy itself, as this requires summing over the necessary MIA values. Thus, the additional complexity in using the Hungarian algorithm (and any of the other procedures which require sorting or turn-based approaches) do not meaningfully add to the complexity of the algorithm as a whole. This also means that the complexity of the TreB-MAP algorithm only depends on $|\mathbf{N}|$ and not the values of t or γ. Interestingly, this means that the TreB-MAP algorithm's performance relative to an exhaustive search will change, since the latter's performance depends on the value of $\mathcal{C}_t^{|\mathbf{N}|}$ which in turn depends on the values of t and γ relative to $|\mathbf{N}|$ as shown in (3.25) and demonstrated in Table 3.1.

The optimality test in Lemma 11 performs the CNA step of the TreB-MAP algorithm on each column of a given grouping $\mathcal{G}$. Consequently, it is necessary to run the complex entropy calculation step γ times for each of the t groups, leading to a complexity of $O(|\mathbf{N}|)$ as well. As mentioned above, although the optimality check will make use of the Hungarian algorithm to guarantee the optimality of the node assignment, the complexity of this is much less than the entropy calculation and can be ignored. In contrast to this, the individual optimality test in Lemma 12 needs to test all unique pairs of nodes that are not in the same group. This results in $\gamma^2\, t(t-1)/2$ comparisons. So too, every test requires an entropy calculation. As a result, this version of the optimality test has a complexity of $O(|\mathbf{N}|^2)$.

The GA approach is structurally the same as the one used in Chapter 4 and thus

will have a similar complexity, which is proportional to the size of the population and the number of SGs. However, since the optimisation problem is more complex in terms of the number of combinations, it is likely to take longer to converge.

6.5 Numerical Example

Consider a system with $|\mathbf{N}| = 12$ files, $\{t, \gamma\} = \{4, 3\}$. Using (3.25), the number of unique combinations is calculated as $|\mathcal{C}_4^{12}| = 15400$. Unlike in previous numerical examples, the total number of MIAs is $2^{|\mathbf{N}|} - 1 = 4095$, which is too large to tabulate. Nevertheless, the final entropy calculations will be shown at each step when necessary, in order to demonstrate how the algorithm functions.

In the first step, there are four possible procedures that could be used to determine $\mathbf{g}_{:,1}$. Table 6.1 shows the entropy values for the individual entropy sources. Clearly, using Procedure 3, $\mathbf{g}_{:,1} = \{X_2, X_{11}, X_9, X_5\}$, while Procedure 4 selects $\mathbf{g}_{:,1} = \{X_3, X_8, X_1, X_4\}$ by sorting the second column in Table 6.1 in ascending and descending order respectively. Procedure 5 only looks at maximising the final entropy values, shown in the third column in Table 6.1. This performs the selection as $\mathbf{g}_{:,1} = \{X_2, X_5, X_7, X_{11}\}$. Finally, Procedure 6 requires using the GISGEM algorithm from Chapter 4, which yields $\mathbf{g}_{:,1} = \{X_3, X_8, X_{11}, X_{10}\}$. The different procedures thus produce vastly different results in the first step.

The second iteration is composed of two parts: CNS and CNA. For this, only the INS that minimises entropy (Procedure 3) from the first step will be demonstrated, since there would be too many permutations otherwise. Table 6.2 shows the results of the necessary entropy calculations, using the initial group of $\{X_2, X_{11}, X_9, X_5\}$. As discussed in Section 6.2.2, both procedures outlined there require an ordering, based on the current performance of a given group. For this example, the BtW ordering will be used, which requires the set to be sorted in ascending order. As the INS procedure used in the first step already implemented this ordering, the set is left unchanged.

First X_2 chooses its best candidate node (i.e. the lowest value in the second column of Table 6.2), which is X_{12}. So too, X_{11} can choose its best candidate node as X_7. On the turn of X_9, it would like to choose X_{12} as a candidate node, but it has already been selected. Thus, it selects its next best choice of X_8. If this were to also have been selected, it would repeat this until a free node is found. Finally, X_5 chooses X_4, and the full candidate set is thus $\mathbf{C}_2 = \{X_{12}, X_7, X_8, X_4\}$.

Table 6.1: Entropy values of the sources used in the first iteration of the numerical example

Entropy Source	$H(X_i)$ (bits)	$H(X_i \mid \mathbf{N} \setminus X_i)$ (bits)
X_1	21577	4
X_2	21175	16
X_3	21667	8
X_4	21513	3
X_5	21366	14
X_6	21511	7
X_7	21427	13
X_8	21620	8
X_9	21329	7
X_{10}	21495	8
X_{11}	21297	12
X_{12}	21400	8

To demonstrate with another procedure, if the individual CNS is used with the WtB ordering, then X_2 chooses t nodes from its column in ascending order, resulting in $\mathbf{C}_2 = \{X_{12}, X_{10}, X_3, X_7\}$.

The second part of the second iteration is to perform the CNA. The cost matrix $\mathcal{S}_2$ is given here for the first candidate set $\mathbf{C}_2 = \{X_{12}, X_7, X_8, X_4\}$:

$$\mathcal{S}_2 = \begin{array}{c} \\ \{X_2\} \\ \{X_{11}\} \\ \{X_9\} \\ \{X_5\} \end{array} \begin{array}{c} \begin{array}{cccc} X_{12} & X_7 & X_8 & X_4 \end{array} \\ \begin{pmatrix} 31823 & 32018 & 32075 & 32183 \\ 32019 & 31979 & 32097 & 31994 \\ 32029 & 32097 & 32034 & 32118 \\ 32169 & 32056 & 31989 & 31915 \end{pmatrix} \end{array} \tag{6.16}$$

It is derived by taking the corresponding entropy values in Table 6.2, with each column representing a candidate node's conditional entropy, given the path in the corresponding row. Each row is then increased by the entropy of the current path, in order to get the total entropy of the path together with the candidate node. Since this is only the second step, in which a single node is assigned to each path, the corresponding values from Table 6.1 can be used as the total entropy for each path. For example, the cost element $s^2_{1,1} = H(c^2_1 \cup \mathbf{P}_{g_{1,1}})$ is calculated as $H(X_{12}, X_2) = H(X_2) + H(X_{12}|X_2)$, which can be found by summing $21175 + 10648$

Table 6.2: Entropy values of the sources used in the second iteration of the numerical example

X_i	$H(X_i\|X_2)$ (bits)	$H(X_i\|X_{11})$ (bits)	$H(X_i\|X_9)$ (bits)	$H(X_i\|X_5)$ (bits)
X_1	10901	10693	10796	10795
X_3	10732	10957	10709	10880
X_4	11008	10697	10789	10549
X_6	10911	10771	10777	10821
X_7	10843	10682	10768	10690
X_8	10900	10800	10705	10623
X_{10}	10718	10820	10774	10578
X_{12}	10648	10722	10700	10803

obtained from Table 6.1 and Table 6.2 respectively.

The final step in this iteration is to do the CNA. The Hungarian algorithm in Procedure 9 effectively optimises the cost matrix by moving the rows until the sum of its trace is minimised. In this example, applying the Hungarian algorithm to the cost matrix in (6.16) reveals that it is already optimised. As a result, the group paths after assignment are:

$$\begin{aligned} \mathbf{P}_{g_{1,2}} &= \{X_{12}, X_2\} \\ \mathbf{P}_{g_{2,2}} &= \{X_7, X_{11}\} \\ \mathbf{P}_{g_{3,2}} &= \{X_8, X_9\} \\ \mathbf{P}_{g_{4,2}} &= \{X_4, X_5\} \end{aligned} \tag{6.17}$$

Alternatively, the turn-based assignment (with WtB ordering) in Procedure 10 can be applied. Here, X_5 chooses X_4, X_9 selects X_{12} and X_{11} chooses X_7, which are the lowest values in the fourth, third and second rows in the cost matrix in (6.16) respectively. There are no conflicts, so each of these nodes can assign its first choice. However, X_2 has to assign X_8, since it is the only possible option, which happens to be its second worst choice in terms of entropy.

The latter two steps are repeated for the last iteration, producing a grouping shown as a trellis diagram in Figure 6.2. For comparison, another two groupings are obtained for a different permutation of procedures for the INS and CNS stages of the algorithm, shown in Figure 6.3. The final entropy values of the groupings are given in Table 6.3, together with the ranking of the result when compared to a list of all possible combinations of groupings. It is clear that the Hungarian algorithm assignment

Table 6.3: Total entropy values and rankings for the groupings found in the numerical example

INS	CNS	CNA	Total Entropy (bits)	Ranking
Procedure 3	Procedure 7	Procedure 10	149308	609
		Procedure 9	148925	2
Procedure 5	Procedure 8	Procedure 10	149309	619
		Procedure 9	149222	189

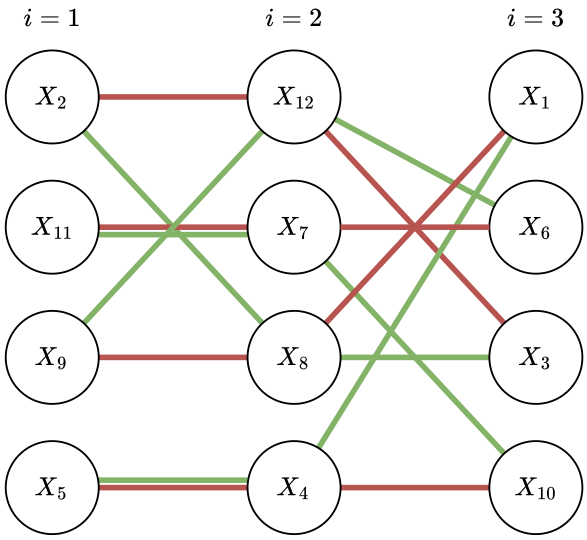


Figure 6.2: Trellis diagram for the numerical example using the TreB-MAP algorithm with Procedure 3 for INS and Procedure 7 for CNS with BtW ordering. For the CNA, the assignments for Procedure 9 are shown in red, while Procedure 10 (with WtB ordering) are green.

method for the first permutation of the algorithm discussed above performs the best, finding the second best possible grouping. The other algorithms still perform strongly, with the worst grouping ranking in the top 5%.

6.6 Simulation Results

Multiple options for the different phases of the TreB-MAP algorithm have been outlined in the previous sections. However, it is unknown what effect the different options have on the performance of the algorithm. So too, the different parameters of the GA have not been tuned for the multi-group scenario. Thus, this section first analyses these algorithms in order to look for the most impactful changes that can be

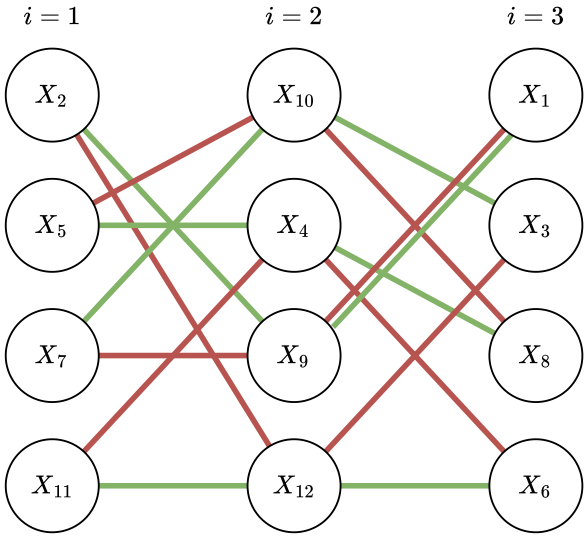


Figure 6.3: Trellis diagram for the numerical example using the TreB-MAP algorithm with Procedure 5 for INS and Procedure 8 for CNS with WtB ordering. For the CNA, the assignments for Procedure 9 are shown in red, while Procedure 10 (with WtB ordering) are green.

made to improve the performance and determine their effect on the time complexity. Then, the best algorithms found using this analysis are tested against each other for various problem setups.

6.6.1 Trellis-Based Algorithm Variable Tuning

There are three phases in the TreB-MAP algorithm. In the INS, there are four procedures to choose from. The CNS has two procedures, which also have possibilities for the ordering. Finally, the CNA stage has two procedures, one of which allows for different orderings. The orderings tested are BtW and WtB (where worst means highest total entropy). In order to test whether the different procedures have any tangible effects, they will also be compared to a random procedure. For the INS step, a random procedure is introduced that chooses all of the starting nodes, while a second randomises the first node and then performs the GISGEM algorithm to find the rest. For the CNS step, a random procedure is created for each of the two selection methods (turn-based and individual) that determines the order in which the nodes are chosen, whereas the CNA stage has a random procedure that just assigns the nodes randomly. All possible combinations of the different procedures are simulated for a setup of $\{|\mathbf{N}|, t\} = \{16, 4\}$, meaning that the total number of possible groupings is $|\mathcal{C}_4^{16}| = 2627625$. The simulation is run using Matlab for 20 different

MIA configurations, on an ASUS VivoBook, with an 11th Generation 2.40 GHz Intel Core i5-1135G7 processor and 8 GB of RAM.

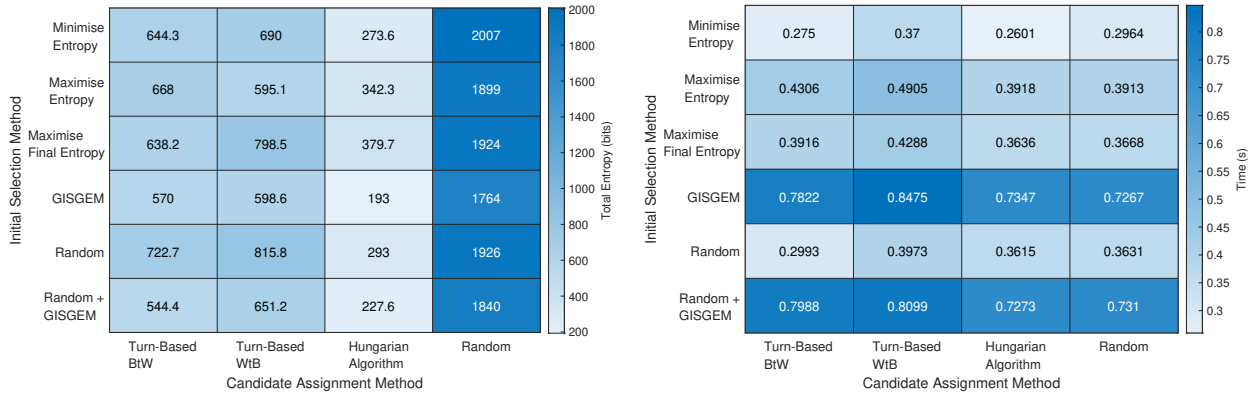
The results for every combination of algorithm stage are shown in Figures 6.4-6.6, for the performance of the groups created and the time taken to run the algorithm. Figures 6.4(a) and 6.6(a) show that the Hungarian algorithm method of assigning nodes is clearly the best in terms of performance. At the same time, the two turn-based methods also perform better than the random version, but not to the same degree. Interestingly, Figures 6.4(b) and 6.6(b) show that the Hungarian algorithm consistently outperforms the turn-based ones in terms of time. As discussed in Section 6.4, the assignment is performed on pre-calculated entropy values, which means that the algorithm will not have a large effect on the performance of the algorithm as a whole. In addition, the Hungarian algorithm implementation used is a highly optimised one with very good performance [81]. Thus, the choice of CNA method is important for the overall performance of the TreB-MAP algorithm.

When looking at the impact of the CNS stage, Figure 6.5(a) shows that there is a marked difference between the turn-based and individual selection methods, regardless of the ordering used. This is also shown in Figure 6.6(a), but the difference is less pronounced. However, Figures 6.5(b) and 6.6(b) show that this comes with a significant increase in time complexity. So too, the ordering does not seem to affect the algorithm's performance in a substantial way, with the random ordering sometimes outperforming the BtW and WtB ones for both CNS procedures.

Finally, the performance results for the INS stage are shown in Figures 6.4(a) and 6.5(a). Here too, the results are mixed, with only the GISGEM method consistently and significantly outperforming the random order. Nevertheless, randomising the first node before performing the GISGEM algorithm does not affect the algorithm to a significant degree. Figures 6.4(b) and 6.5(b) both show a marked increase in time complexity for the GISGEM approach, as predicted in Section 6.4.

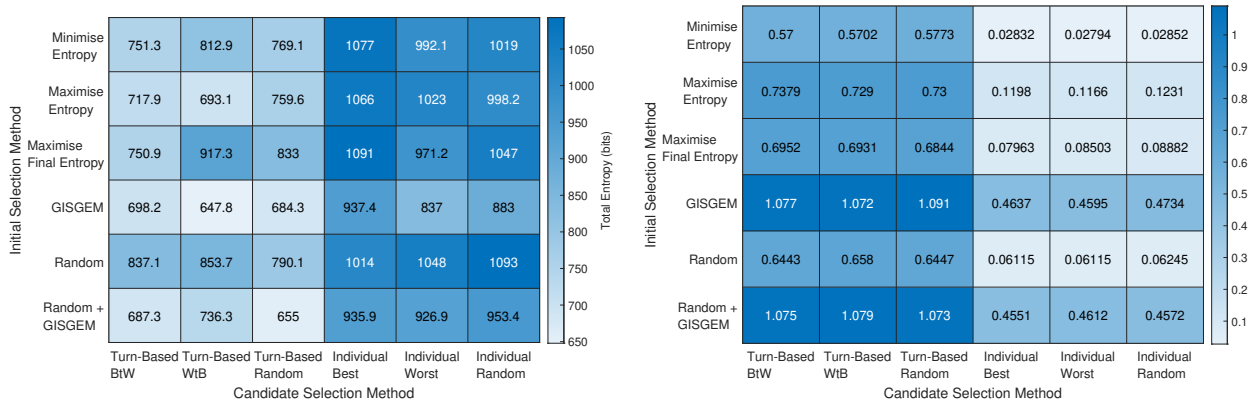
6.6.2 Genetic Algorithm Variable Tuning

As in Section 4.5.2.1, the various parameters for the GA are now tuned. Although this was done previously, the new optimisation problem is considered unique enough to warrant testing again. Table 6.4 shows the different parameters tested, as well as the ranges used. The ranges for SG, $|\mathbf{P}|$ and α_e are larger than in Section 4.5.2.1, in order to get a better understanding of their effect. The range for α_c already covers



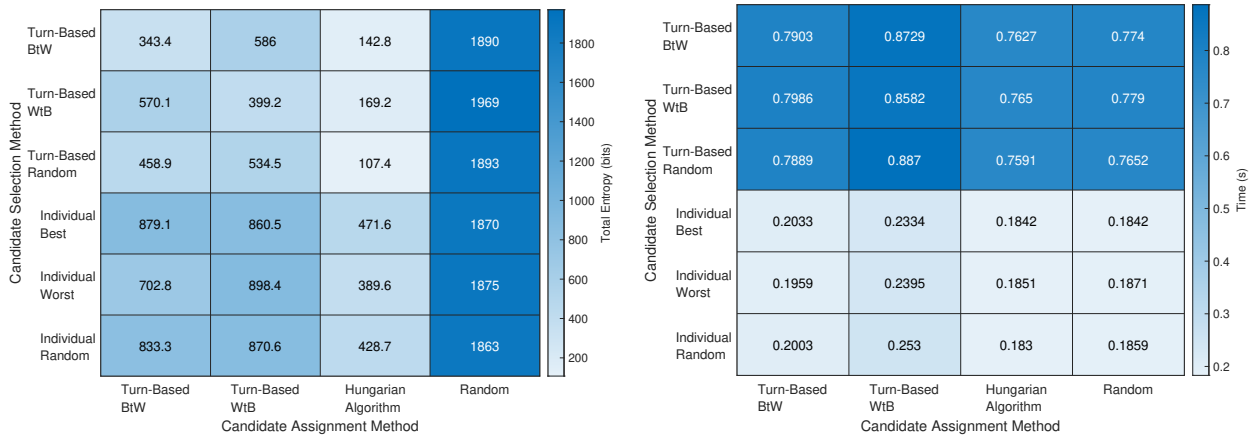
(a) Grouping entropy performance (b) Time performance

Figure 6.4: A heatmap showing the performance of the TreB-MAP algorithm for different INS and CNA methods. $|\mathbf{N}| = 16$ files, $t = 4$ groups



(a) Grouping entropy performance (b) Time performance

Figure 6.5: A heatmap showing the performance of the TreB-MAP algorithm for different INS and CNS methods. $|\mathbf{N}| = 16$ files, $t = 4$ groups



(a) Grouping entropy performance (b) Time performance

Figure 6.6: A heatmap showing the performance of the TreB-MAP algorithm for different CNS and CNA methods. $|\mathbf{N}| = 16$ files, $t = 4$ groups

Table 6.4: Different parameter settings for GA testing and their impact on the quality of results, complexity and number of iterations

Parameter	Symbol	Range	Performance Impact	Time Impact	Iteration Impact
Stall Generations	SG	5, 10, 15, 20, 25, 30	Large	Large	Large
Population Size	$\|\mathbf{P}\|$	5, 10, 15, 20, 25, 30	Large	Medium	Small
Elite Percentage	α_e	10%, 20%, 30%, 40%	Medium	Minimal	Minimal
Crossover Percentage	α_c	20%, 40%, 60%, 80%	Small	Minimal	Minimal

most of the possible values and so is kept the same. The simulations are run in the same environment as in Section 6.6.1, with the same device and problem setup of $\{|\mathbf{N}|, t\} = \{16, 4\}$. However, the GA is only run for 10 different MIA configurations in order to reduce the complexity of the simulation, as the GA has more parameters and ranges to test than in the previous variable tuning. Every possible combination of parameter value over the associated ranges was tested.

Figure 6.7 describes the effect of each of the parameters on the performance of the algorithm. Of note is that the size of the population has more of an effect on the performance than in Section 4.5.2.1, as well as the percentage of elite members. It was experimentally determined that the relationships between SG and $|\mathbf{P}|$ and performance are exponential, with small increases in these parameters resulting in much better performance. For the time complexity, the results in Figure 6.8 show that the increased effect of population size comes with a greater impact on the time taken for the algorithm to converge, whereas the two percentage parameters do not have a marked effect. Here, the relationships between SG and $|\mathbf{P}|$ and time complexity are linear (where better performance comes at the expense of time complexity). Finally, Figure 6.9 plots the effect of the parameters on the number of iterations taken until convergence is reached, with SG being the only parameter to have enough of a significance. It is surmised that this contributes to the greater effect that SG has on the time complexity as compared to $|\mathbf{P}|$, since the simulation can better optimise calculations performed in a single iteration as opposed to doing it over multiple iterations.

6.6.3 TreB-MAP and GA Comparison Results

In this chapter, two algorithms have been presented that attempt to solve the multi-group optimisation problem. Additionally, two different optimisation techniques have been outlined, which continuously improve an existing result. This section tests all of these approaches in order to determine their performance.

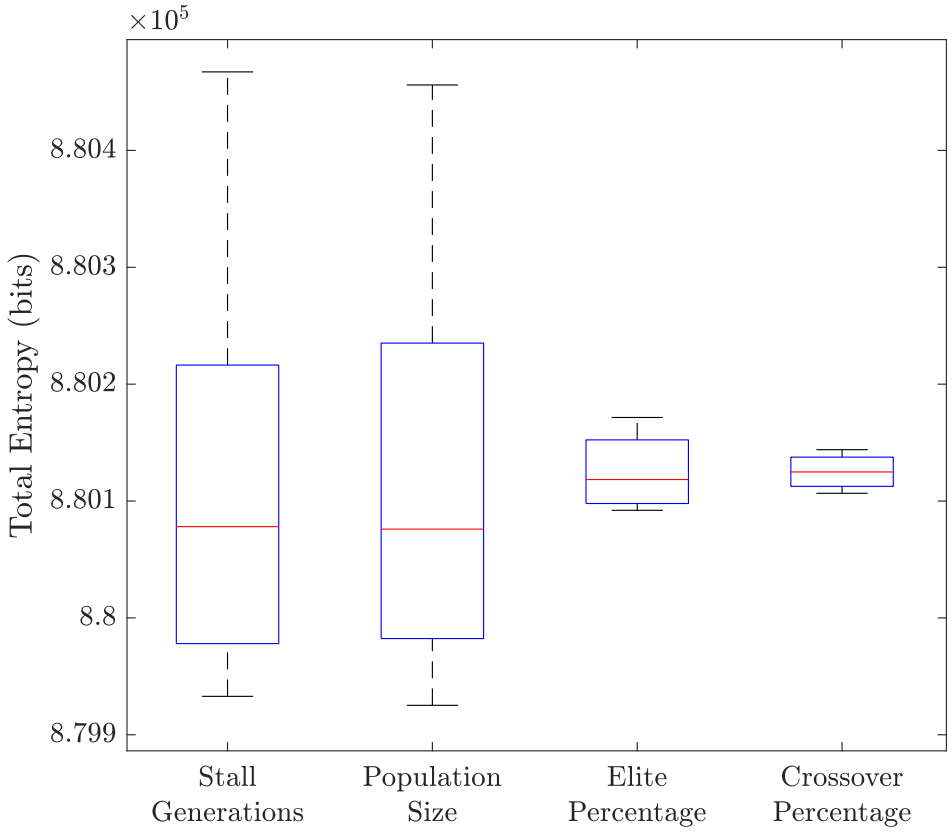


Figure 6.7: The effect of the different GA tuning variables on the performance of the resulting groupings. $|\mathbf{N}| = 16$ files, $t = 4$ groups

Based on a comparison of the different phases of the TreB-MAP algorithm in Section 6.6.1, Procedure 6 and 7 are the most effective at improving the performance in the INS and CNS stages respectively, with associated increases in time complexity. Nevertheless, the performance improvement is significant enough to justify this. The ordering for Procedure 7 does not affect performance enough to justify a particular order. As a result, a random order is used, as it simplifies the implementation somewhat. Procedure 9 is clearly the best method for the CNA stage.

For the GA approach, it is shown in Section 6.6.2 that the value of α_e showed better performance results when kept low without affecting time complexity, so 10% is chosen. The crossover percentage did not show a specific trend, but it consistently performed better when set to 60% and so this is chosen. There is a trade-off between time complexity and performance with respect to the SG and size of the population. It is thus decided that two versions of the GA will be tested. The first is a Low Complexity (LC) version, which is designed to lower the time complexity without sacrificing too much performance, while the other is a High Complexity (HC) version,

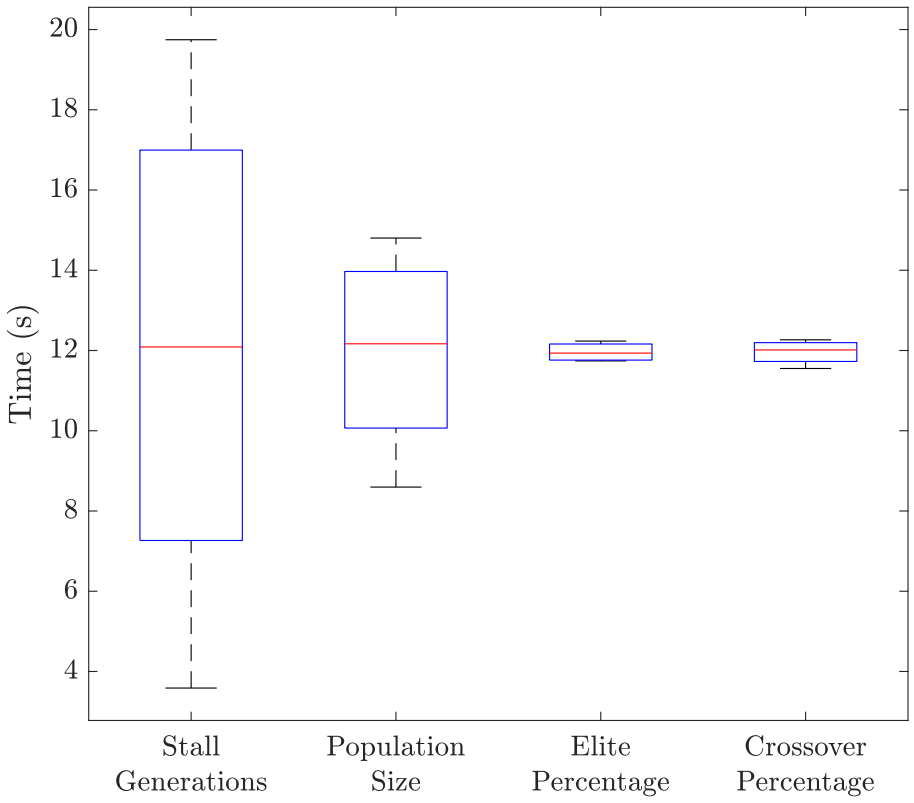


Figure 6.8: The effect of the different GA tuning variables on the time taken until convergence is reached. $|\mathbf{N}| = 16$ files, $t = 4$ groups

which aims to yield improved results without incurring too much time complexity. Table 6.5 summarises the setup variables for these two versions, as well as the TreB-MAP algorithm.

The algorithms are also tested in terms of how well they set up the IOI-MG and IIOI-MG optimisation algorithms, as a more suitable grouping would allow for better optimisation. As with the variable tuning, the algorithms will be tested against a randomly chosen grouping, to which the optimisation algorithms are also applied.

The simulation is run for 10 different MIA configurations on the same device as the TreB-MAP and GA tuning simulations. The values of $\{t, \gamma\}$ are adjusted such that the number of combinations $|\mathcal{C}_t^{|\mathbf{N}|}|$ is maximised for each value of $|\mathbf{N}|$, while the values of $|\mathbf{N}|$ are chosen to be composite numbers in the range of 4 to 20. All combinations of algorithms and optimisation approaches are found to converge within 300 iterations, after combining the iterations used for the algorithm and optimisations. So too, the two GA variants did not exceed their MNI.

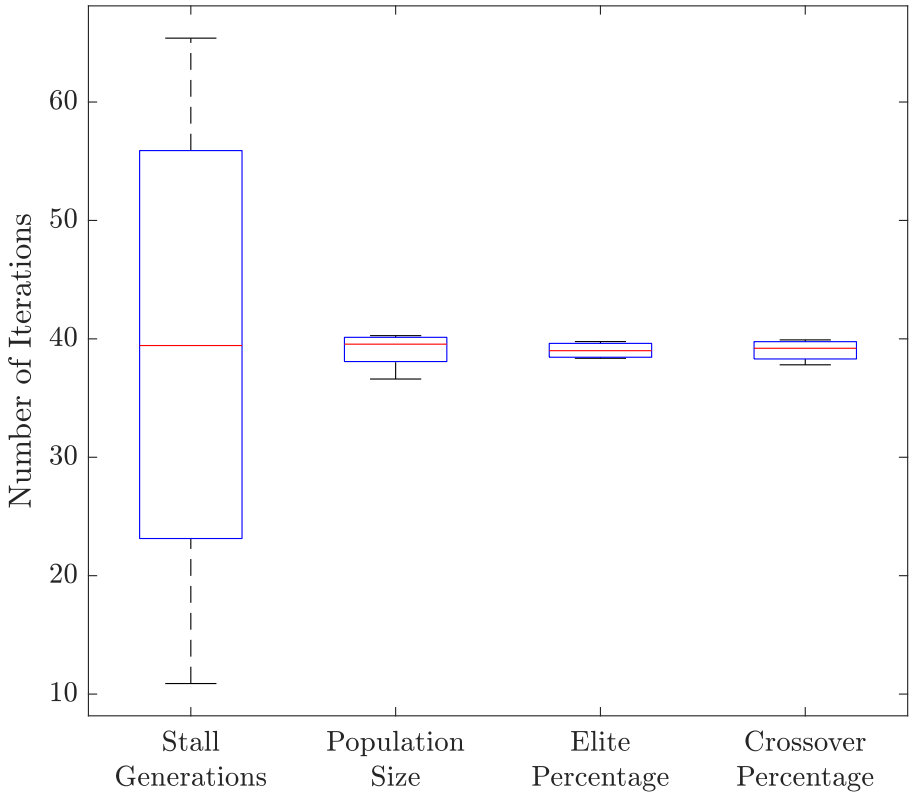


Figure 6.9: The effect of the different GA tuning variables on the number of iterations until convergence is reached. $|\mathbf{N}| = 16$ files, $t = 4$ groups

Table 6.5: The algorithms used in the algorithm comparison test, together with their associated tuned variables

Algorithm	Variable Values
TreB-MAP	INS = Procedure 6, CNS = Procedure 7 (random ordering), CNA = Procedure 9
GA (LC)	$\|\mathbf{P}\| = 15$, $\alpha_e = 0.1$, $\alpha_c = 0.6$, SG = 10, MNI = 200
GA (HC)	$\|\mathbf{P}\| = 30$, $\alpha_e = 0.1$, $\alpha_c = 0.6$, SG = 20, MNI = 200

As discovered in Section 5.8, the raw performance numbers are not a good enough indication of the real performance of the algorithm. Consequently, the groupings found by the different algorithms are ranked against every possible combination, found using a BF method, in order to determine the percentile in which they fall. For a configuration greater than $\{|\mathbf{N}|, t\} = \{14, 7\}$, the complexity of finding the performance of every group becomes prohibitive. As a result, Table 6.7 only shows percentile values up to that configuration. The raw performance results for the configuration with the highest number of possible groupings ($\{20, 5\}$) are shown in Figure 6.10. For better readability, the performance of a randomly found grouping is

Table 6.6: The $\{|\mathbf{N}|, t, \gamma\}$ values used in the simulation comparing the different algorithms and the number of possible groupings.

$\lvert\mathbf{N}\rvert$	t	γ	$\lvert\mathcal{C}_t^{\lvert\mathbf{N}\rvert}\rvert$
4	2	2	3
6	3	2	15
8	4	2	105
9	3	3	280
10	5	2	945
12	4	3	15400
14	7	2	135135
15	5	3	1401400
16	4	4	2627625
18	6	3	190590400
20	5	4	2546168625

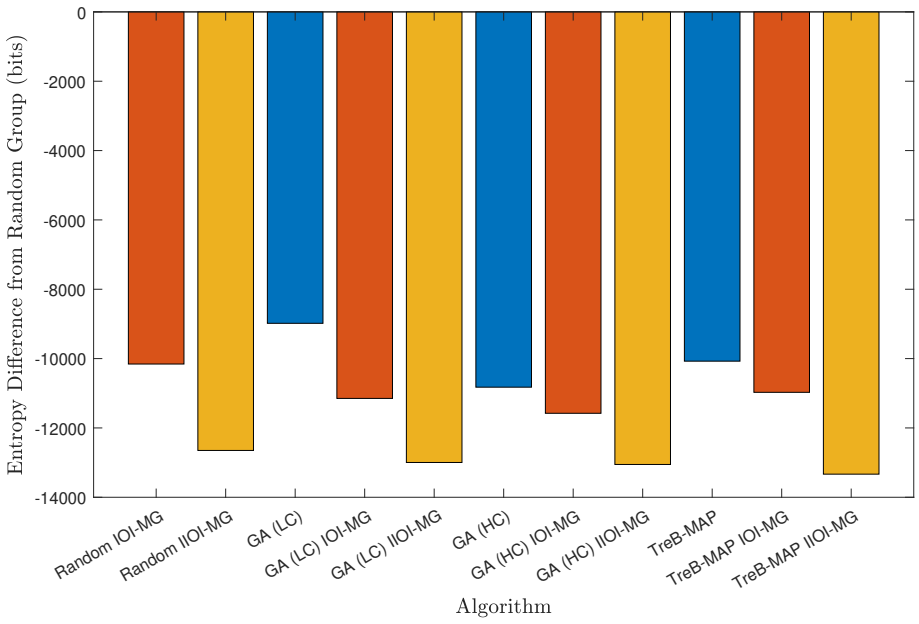


Figure 6.10: Performance of the various proposed algorithms, relative to the performance of the randomly chosen group. $|\mathbf{N}| = 20$ files, $t = 5$ groups

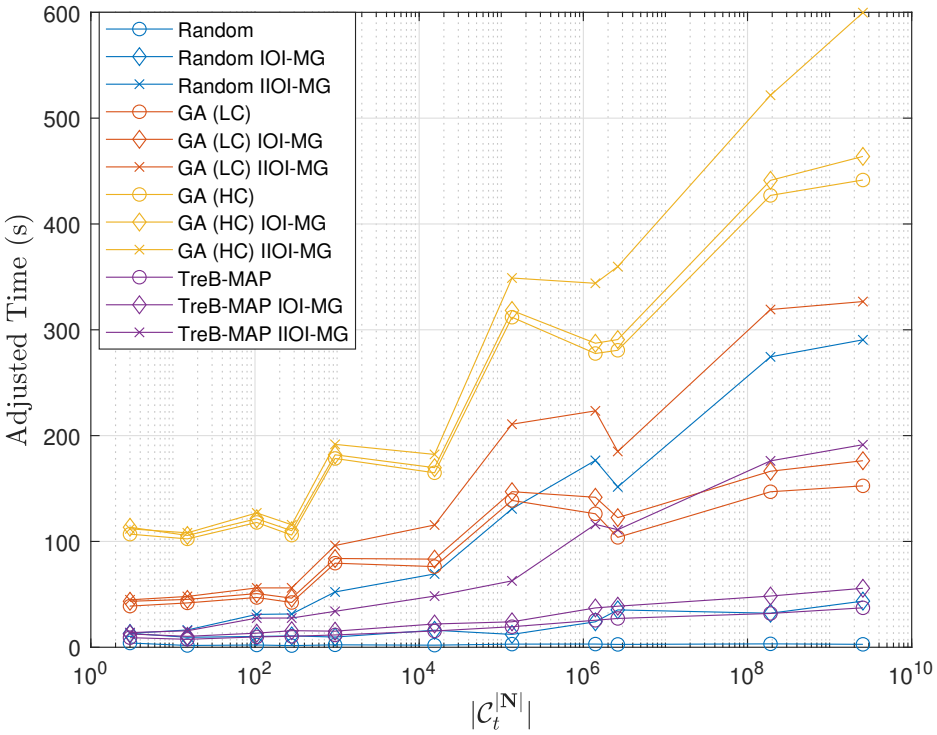


Figure 6.11: The time complexity of the proposed algorithms and optimisations for all the $\{|\mathbf{N}|, t\}$ configurations, adjusted for the complexity of calculating the entropy.

subtracted from each of the algorithm's ones. Thus, a lower value is better.

Plotting the time complexity values without further processing would result in an unfair comparison over the different $\{|\mathbf{N}|, t\}$ configurations. This is because, in addition to increasing the number of possible groupings, increasing $|\mathbf{N}|$ also increases the number of MIAs. Since calculating the total entropy of a grouping involves summing over the MIAs, this results in a greater time complexity that is unrelated to the algorithms themselves. Thus, the time results are adjusted by dividing the total time taken to run the algorithm/optimisation method by the time taken to evaluate a grouping's entropy, averaged over 50 repetitions. The results for all of the algorithms and optimisation method combinations over the different $\{|\mathbf{N}|, t\}$ configurations are shown in Figure 6.11, where the x axis is given as a log plot, owing to the exponential increase in the number of possible grouping combinations.

Table 6.7: Average percentile values for the proposed algorithms

Configuration	Random			GA (LC)			GA (HC)			TreB-MAP		
$(\{\|\mathbf{N}\|, t\})$	Original	IOI-MG	IIOI-MG	Original	IOI-MG	IIOI-MG	Original	IOI-MG	IIOI-MG	Original	IOI-MG	IIOI-MG
$\{4, 2\}$	63.33	100	100	100	100	100	100	100	100	96.67	96.67	100
$\{6, 3\}$	48.00	90.67	100	100	100	100	100	100	100	93.33	93.33	100
$\{8, 4\}$	53.05	90.00	99.14	100	100	100	100	100	100	96.48	96.48	99.81
$\{9, 3\}$	62.46	97.43	99.89	99.39	99.39	100	100	100	100	95.93	99.29	99.89
$\{10, 5\}$	67.51	98.11	99.95	99.84	99.92	99.95	99.97	99.97	99.98	98.33	98.33	99.96
$\{12, 4\}$	39.41	98.45	99.96	99.71	99.90	99.99	99.95	99.96	99.98	99.41	99.66	99.97
$\{14, 7\}$	47.55	99.68	100	99.87	99.99	99.99	99.99	100	100	99.91	99.91	99.99
mean	54.47	96.33	99.85	99.83	99.89	99.99	99.99	99.99	99.99	97.15	97.67	99.95

6.7 Discussion

From the percentile results in Table 6.7, it is clear that all of the proposed algorithms and optimisations have good performance, outperforming the random method and finding groupings that are above the 90th percentile. For the algorithms run without the optimisation methods, the GA (HC) consistently performs better than the LC version as expected, while the TreB-MAP algorithm performs slightly worse. At the same time, the performance in the most complex problem configuration shown in Figure 6.10 reveals that the TreB-MAP algorithm actually performs better than the GA (LC) variant, although this could not be confirmed by a percentile comparison. Nevertheless, the TreB-MAP algorithm runs with a significantly lower time complexity gradient over the different problem configurations, as shown in Figure 6.11.

Comparing the optimisation methods, the IOI-MG approach always improves an existing result, although this improvement is a modest one. The one exception is when the initial grouping used was chosen at random. In this case, the results' average rankings increased from the 54th to the 96th percentile, or close to the performance of the TreB-MAP approach without optimisation, as shown in Table 6.7. So too, using this optimisation technique comes with an increase in time complexity, although this is shown to be relatively small. The IIOI-MG optimisation method also improves the performance of existing groupings. In the case of the randomly found group and the group found using the TreB-MAP algorithm, the improvement is substantial, while the GA (LC) approach has a modest improvement and the GA (HC) does not see much of an improvement. However, this can be attributed to the fact that these algorithms already find close-to-optimal results. At the same time, this comes with a steep increase in the time complexity as evidenced by Figure 6.11, especially for the randomly selected group.

Overall, the TreB-MAP algorithm has an impressive performance compared to

its time complexity, performing better than the random approach with IOI-MG optimisation in a similar time. In addition, applying the IIOI-MG optimisation puts its time performance near the GA (LC) with IOI-MG optimisation, with better performance. It has the added benefit of being iterative, meaning that the time performance of the algorithm is predictable and repeatable, making it a suitable candidate for solving the optimisation problem in (3.24).

6.8 Conclusion

The TreB-MAP algorithm is proposed to solve the multi-group clustering problem for a hybrid SW/CC system. It is an evolution of the GISGEM and CSP algorithms, but uses a trellis-based approach. It was found that using the GISGEM algorithm to choose the starting nodes in the INS phase of the algorithm yields the best results. Moreover, using the Hungarian algorithm optimally solves the assignment problem in the CNA phase of the TreB-MAP algorithm, giving the best results. However, the different approaches in the CNS phase did not have as marked an effect on the final results.

Additionally, the optimisation tests are updated to account for the multi-group scenario, which allows for two different types of tests to be performed. The GA is more amenable to adapting to this scenario than the ACO, and the tuning variables are updated to take account of the new optimisation problem. Although the TreB-MAP algorithm found results in the 97th percentile compared to the 99th percentile for the GA in both HC and LC formats, it does so with a greatly reduced time complexity, even when applying the IOI-MG approach on top of the algorithm. As with the single group solutions, the proposed approaches perform at close-to-optimal levels for the multi-group clustering problem.

Chapter 7

Conclusion

7.1 Contributions

This research has examined the challenges in realising a hybrid Slepian-Wolf Coded Caching (SW/CC) system, in which the distributed Slepian-Wolf (SW) coding improves the already impressive Coded Caching (CC) used to reduce the information needed to be transmitted in a content delivery system. The first challenge is in the increased complexity in using these techniques, since SW coding requires a joint decode of all of the information at the user end. The second challenge is improving the fragility of the system, since a lossless decode is not guaranteed if some of the files used in the SW/CC system are not received. To solve this, this research introduced clustering of the files into different groups. This allows the system to be designed with a specific time complexity in mind, since the SW decoding complexity is well-defined, based on the number of files used. So too, the system is made more stable, since only files in the same group are required to recover the information. However, the clustering was shown to be an NP-hard problem in its own right. Additionally, it has been demonstrated that the existing techniques in literature are not sufficient solutions.

Thus, two optimisation problems were formulated. The first was a simplified one, which looks at grouping the files into two clusters. One of these is coded using SW while the other is left uncoded. The Greedy Iterative Single Group Entropy Minimisation (GISGEM) algorithm is proposed to solve this problem. An iterative algorithm, it chooses a single node to add to the uncoded group at each step until the desired size is reached. It attempts to maximise the entropy at each step and is shown to be a low-cost yet effective approach. In testing, GISGEM finds clusters with

an average percentile value of 98.11, when ranking the clusters using an exhaustive search. Although this is outperformed by the Genetic Algorithm (GA) meta-heuristic benchmark, which finds clusters with an average percentile value of 99.81, it enjoys a time complexity that is two times less.

GISGEM is further improved by allowing it to search for the best clustering using multiple paths, resulting in the Champion Selection Procedure (CSP) algorithm and its variants. They are able to produce clusterings that are significantly better performing than the GISGEM clusterings and are near-to-optimal, finding clusters with performance falling in the 99th percentile and having a comparable time complexity to the GA. Another meta-heuristic approach, the Ant Colony Optimisation (ACO), has an even better performance (with clusters having an average percentile value of 99.92), but with double the time complexity.

A necessary, but not sufficient, condition of optimality is derived, where the order of the grouping is changed and the GISGEM considerations are reapplied to highlight problematic sources. The optimality condition is integrated into an improvement approach called Iterative Optimisation Improvement (IOI) by repeatedly testing the optimality of a cluster and replacing the suggested sources. This approach outperforms all of the other proposed algorithms in the single-group case (with an average cluster percentile value of 99.97), while having a similar time complexity as the ACO.

In the second, more complex, problem, the files are clustered into t groups, all of which are coded. Here, the GA meta-heuristic shows impressive performance, whereas the ACO method is not able to be adapted to this new paradigm. The CSP approach is adapted to a trellis-based one by restricting the multiple paths and merging them into one solution. The resulting Trellis-Based Multi-group Allocation Procedure (TreB-MAP) algorithm produces results slightly lower than the GA (97th and 99th percentiles respectively), but with greatly improved time complexity. So too, the optimality condition is re-derived for the multi-group case, and the IOI technique is adapted to the new model, resulting in two optimisation techniques: the IOI: Multi-Group (IOI-MG) and the Individual IOI: Multi-Group (IIOI-MG). The former is less complex than the latter, but with worse performance. The TreB-MAP results in particular are greatly improved when using these techniques.

It is proved theoretically in the single-group case that an iterative approach is unable to guarantee an optimal solution, even when augmented with the continual optimisation approaches. This is backed up by strong empirical evidence for smaller

group sizes by comparing it to an exhaustive search and larger group sizes, where exhaustive searches are intractable, by considering that the iterative optimisation methods were able to improve the result consistently during testing. The same empirical evidence is obtained for the multi-group case, although this is not proved theoretically.

In terms of scalability, it is shown that the base algorithms have a linear time complexity that scales with the number of files in the system ($O(|\mathbf{N}|)$). This means that the algorithms are well suited to larger systems, making them attractive solutions for Information-Centric Networks (ICNs) as the field moves towards more practical applications. ICNs are a major departure from the current Internet Protocol (IP) models that make up the Internet today. As such, they are still in the applied research stage, meaning that there are currently no real-world applications. So too, CC requires a closed system in order to function correctly, where the end user caches are integral to the system as a whole. This makes it difficult to implement a SW/CC system with the technology currently available.

Despite this, one scenario where the hybrid SW/CC system could be applied is in the context of video surveillance. The positioning of various monitoring cameras means that the videos produced will naturally be highly correlated to one another. The files are usually downloaded by different parties continuously and on-demand (such as security guards, offshore monitoring facilities or automated feature detection algorithms), which could be constrained by the available bandwidth. The hybrid SW/CC system would alleviate the bandwidth used in the cases above by storing parts of the compressed video files at the end users during off-peak hours. The algorithms developed in this research could be used to group the files into clusters, such that the decoding complexity is reduced while maintaining the high compression gains and bandwidth availability.

7.2 Future Work

In creating the algorithms to solve the clustering problem throughout the course of this research, future possible improvements were uncovered. The following subsections look at the different ways to further develop the ideas introduced thus far. For each, it is shown in a general way how the current algorithms could be adapted to the new views of the problem. However, it is not guaranteed that these algorithms are amenable to be adapted. Thus, more theoretical research is needed to determine the dynamics of the new paradigms, with the possibility that entirely new algorithms

would need to be developed instead.

7.2.1 Algorithm Improvements

The progression of this research has been to create a single novel view of creating clusters, by adding a single file to a potential cluster at each step in a greedy manner. This is called the GISGEM approach. The extension of this is to have multiple paths in which the same approach is used. This is the framework for the CSP algorithms. The additional paths are able to compensate for the greediness of the GISGEM approach and the extra complexity can be traded off with the quality of the results. To solve the multi-group problem, the TreB-MAP algorithm is itself an adaptation of the CSP method, where the multiple paths are limited so that they are disjoint, thus producing multiple groups instead.

With regards to the GISGEM approach, the current design uses a 'no regret' scheme, where sources selected previously are not able to be removed in a future iteration. However, a better performing result could be achieved by using a 'look ahead' approach and is a potential avenue for future work. So too, the CSP chooses from a larger pool than GISGEM and so is able to find better results, but it is not able to go back a step. It would be interesting to see how the optimality condition could be used to add 'regret' to the current algorithms. This is expected to further improve the quality of the results produced and may positively affect the convergence rate.

For the TreB-MAP algorithm, the natural extension of this is to increase the number of disjoint paths created, similar to how the CSP extends the GISGEM algorithm. In order to achieve this, it would be necessary to alter the Candidate Node Selection (CNS) step of the TreB-MAP method. Instead of choosing t nodes in this step, a new multiplier variable can be introduced such as q, resulting in qt nodes selected in this phase. Every grouping of t nodes would be chosen using the rules outlined in Chapter 6.2.2 so that the paths remain disjoint. However, each grouping of t nodes could be allowed to overlap. The Candidate Node Assignment (CNA) phase would be performed for every t nodes as usual, resulting in q potential groupings. Much like the CSP approach, the variable $m < q$ could then limit the number of groupings that survive to the next iteration, ensuring that the complexity is bounded. This approach would allow for more groupings to be created, with the expectation that the quality of results would increase together with the time complexity.

7.2.2 Expanding the Problem Statement

Another avenue of future work is in the problem statement itself. The goal of the clustering is to reduce the complexity of the Distributed Source Coding (DSC) decoding and increase the robustness of the system as a whole. In order to simplify the problem, the optimisation included the condition that the groups are disjoint. However, in a more practical scenario, loosening the constraints of the groups could produce better results. The promise of this approach is now demonstrated in two different ways.

7.2.2.1 Arbitrary Group Sizes

In the research presented, the groups are always fixed according to the variable γ. This ensured that the problem analysis is simplified and the algorithms are straightforward to implement. The size of the group directly affects the fragility and complexity of the SW coding, since it is required that the information from all of the nodes are present and the joint decode increases in complexity depending on the number of nodes involved. Thus, in a more practical scenario, it would be beneficial to relax the rigidity of the group sizes, depending on the correlation structure. Files that have less correlation on average to other files would benefit from smaller group sizes. For example, if a file X_i has an entropy $H(X_i, X_j) \approx H(X_i, \mathbf{N} \setminus X_i)$, then including any other files besides for X_j can only increase the complexity and fragility of the system without tangible increases to the efficacy of the coding. The optimisation problem now becomes finding the best clustering for an arbitrary number of groups and group sizes, within limits, that minimises the entropy while maintaining a reasonable time complexity of the SW decoding operation. If the number of groups is bounded as $t_{\min} \leq t \leq t_{\max}$ and the group size for group i is given by $\gamma_{\min} \leq \gamma_i \leq \gamma_{\max}$, then the total number of possible clusterings can be determined using the Stirling number of the second kind:

$$\sum_{k=t_{\min}}^{t_{\max}} \left\{ {|\mathbf{N}| \atop k} \right\} = \sum_{k=t_{\min}}^{t_{\max}} \frac{1}{k!} \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} j^n \tag{7.1}$$

The number of possible clusterings is thus greater than in the multi-group optimisation problem presented in this research. In order to solve this updated optimisation problem, it would be necessary to develop a measure of the SW decoding complexity. Then, the current algorithms could be adapted by allowing a group to terminate

on any iteration greater than $\gamma_{\min}$ if the coding gains are less than the consequent increase in coding complexity, according to some equivalence evaluation between the two metrics.

7.2.2.2 Multi-Group Assignment

Another approach in expanding the problem statement is by increasing the number of groups that files can belong to. Currently, it is required that clusters are disjoint, as in Figure 7.1, allowing for easier characterisation of the problem. However, it is possible, for example, for a file X_i to be strongly correlated to X_j and X_k, while X_j and X_k themselves are weakly correlated. Thus, assigning X_i to the same group as X_j and X_k would not be as effective as assigning X_i to two different groups that contain X_j and X_k respectively. This presents an unconstrained assignment problem, shown in Figure 7.2. As above, using this problem statement would also require groups of arbitrary size, although this could be limited to $\gamma_i \leq \gamma_{\max}$, since the groups would be at least of size γ. So too, limitations on the maximum number of groups that a file can belong to and the number of files that can belong to more than one group could be implemented in order to reduce the complexity of the problem. The number of possible clusterings using this approach would need to be derived. A possible method of adapting the current algorithms is to remove the restrictions of the CSP and TreB-MAP approaches. When selecting and assigning files to groups, the limitations of the pool of files to choose from could be relaxed, allowing files to be chosen again in a future iteration. This would come with the proviso that this file cannot be added to the same group again.

7.2.3 Joint Source-Channel Coding Considerations

The hybrid SW/CC system considered in this research implements source coding (SW) and channel coding (CC) independently. It has been shown that a Joint Source-Channel Coding (JSCC) in this setting (called the Distributed Joint Source-Channel Coding (DJSCC) to take into account the DSC aspect) could yield better results than considering them independently, [82–85]. This is especially so in this case, since the CC part of the system could also benefit from clustering. For example, it has been shown recently in [62] that clustering nodes in a CC system greatly improves its performance. To accommodate this, the clustering algorithms could also take into account the effect of these clusters on the CC performance. An example of

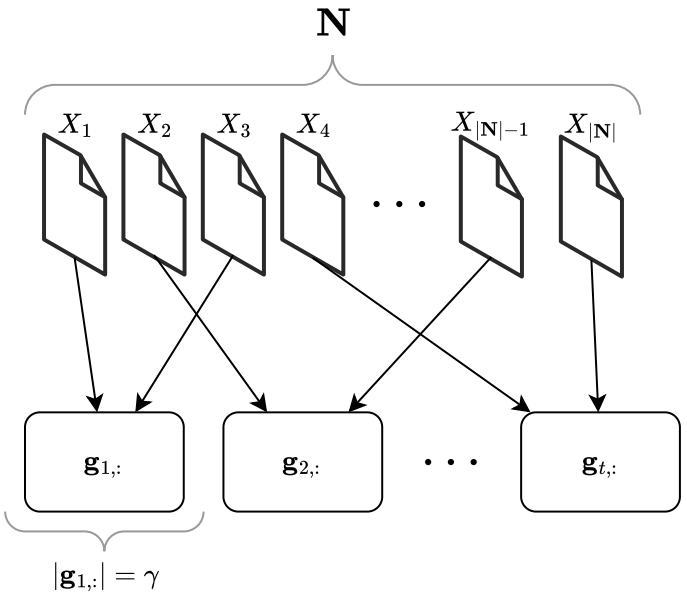


Figure 7.1: An example of a constrained assignment of files according to the optimisation problem presented in this research

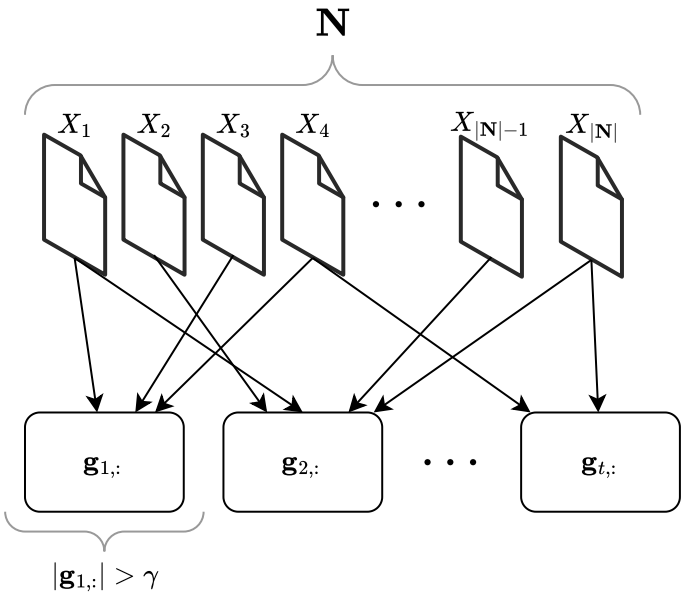


Figure 7.2: An example of an unconstrained assignment of files using the proposed multi-group assignment optimisation problem

the clustering is shown in Section 4.5.1, but this could be made more rigorous and incorporated into the SW clustering itself.

7.3 Conclusion

This research has shown that algorithms exist that are able to solve the clustering problem in a hybrid SW/CC system, for both the simpler single group problem and the more general multi-group problem, while running in polynomial time. For the single group problem, greedy iterative selection and tree-based approaches are able to solve this problem, while a trellis-based approach is more suitable for solving the multi-group problem. All approaches are shown to find near optimal results in their respective applications. Although necessary conditions for optimality are derived for both problems, these conditions are not sufficient, meaning that optimality cannot be guaranteed. However, the conditions are useful in further improving the results generated by the algorithms developed in this research.

It is found that using the raw performance of a clustering in terms of total entropy is not a good indicator of the performance of a prospective algorithm, since ranking the clusters in an exhaustive list of all potential combinations of clusters shows that the distance between them in terms of entropy values can vary greatly.

In general, the meta-heuristic approaches perform better than the iterative ones, but are less predictable in terms of the time complexity compared to the quality of the results. All of the algorithms proposed in this research are able to find clusters without making any assumptions about the underlying correlation structure, making them suitable for more real-world applications. So too, they represent novel solutions to problems that are identified, but not dealt with in literature to date.

Bibliography

[1] B. Rosen and L. Cheng, "Greedy iterative and meta-heuristic clustering with coded caching and Slepian-Wolf compression for correlated content," *IEEE Access*, vol. 12, pp. 12 323–12 334, Jan. 2024.

[2] B. Rosen, A. M. Abu-Mahfouz, and L. Cheng, "Path-based clustering approaches for a hybrid Slepian-Wolf compression and coded caching system," *IEEE Access*, vol. 13, pp. 48 935–48 949, Mar. 2025.

[3] D. Slepian and J. Wolf, "Noiseless coding of correlated information sources," *IEEE Transactions on Information Theory*, vol. 19, no. 4, pp. 471–480, Jul. 1973.

[4] T. Cover, "A proof of the data compression theorem of Slepian and Wolf for ergodic sources (corresp.)," *IEEE Transactions on Information Theory*, vol. 21, no. 2, pp. 226–228, Mar. 1975.

[5] A. Wyner and J. Ziv, "The rate-distortion function for source coding with side information at the decoder," *IEEE Transactions on Information Theory*, vol. 22, no. 1, pp. 1–10, Jan. 1976.

[6] 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.

[7] S. S. Pradhan and K. Ramchandran, "Distributed source coding using syndromes (DISCUS): design and construction," in *Data Compression Conference, 1999. Proceedings. DCC '99*, Mar. 1999, pp. 158–167.

[8] ——, "Distributed source coding: symmetric rates and applications to sensor networks," in *Proceedings DCC 2000. Data Compression Conference*, Mar. 2000, pp. 363–372.

[9] D. Schonberg, K. Ramchandran, and S. S. Pradhan, "Distributed code constructions for the entire Slepian-Wolf rate region for arbitrarily correlated sources,"

in *Data Compression Conference, 2004. Proceedings. DCC 2004*, Mar. 2004, pp. 292–301.

[10] F. Chen and S. Cheng, "An efficient decoding algorithm of matrix partition codes," *IEEE Signal Processing Letters*, vol. 21, no. 4, pp. 414–417, Apr. 2014.

[11] Y. Yang, V. Stankovic, Z. Xiong, and W. Zhao, "On multiterminal source code design," *IEEE Transactions on Information Theory*, vol. 54, no. 5, pp. 2278–2302, May 2008.

[12] V. Stankovic, A. D. Liveris, Z. Xiong, and C. N. Georghiades, "Design of Slepian-Wolf codes by channel code partitioning," in *Data Compression Conference, 2004. Proceedings. DCC 2004*, Mar. 2004, pp. 302–311.

[13] S. Forchhammer, M. Salmistraro, K. J. Larsen, X. Huang, and H. V. Luong, "Rate-adaptive BCH coding for Slepian-Wolf coding of highly correlated sources," in *2012 Data Compression Conference*, Apr. 2012, pp. 237–246.

[14] M. Vaezi and F. Labeau, "Distributed source-channel coding based on real-field BCH codes," *IEEE Transactions on Signal Processing*, vol. 62, no. 5, pp. 1171–1184, Mar. 2014.

[15] V. Toto-Zarasoa, A. Roumy, and C. Guillemot, "Rate-adaptive codes for the entire Slepian-Wolf region and arbitrarily correlated sources," in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, Mar. 2008, pp. 2965–2968.

[16] S. Li and A. Ramamoorthy, "Multiple-source Slepian-Wolf coding under a linear equation correlation model," *IEEE Transactions on Communications*, vol. 60, no. 9, pp. 2402–2407, Sep. 2012.

[17] S. Eleruja, U. Abdu-Aguye, M. Ambroze, M. Tomlinson, and M. Zak, "Design of binary LDPC codes for Slepian-Wolf coding of correlated information sources," in *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Nov. 2017, pp. 1120–1124.

[18] J. Garcia-Frias, "Compression of correlated binary sources using turbo codes," *IEEE Communications Letters*, vol. 5, no. 10, pp. 417–419, Oct. 2001.

[19] A. D. Liveris, Z. Xiong, and C. N. Georghiades, "A distributed source coding technique for correlated images using turbo-codes," *IEEE Communications Letters*, vol. 6, no. 9, pp. 379–381, Sep. 2002.

[20] K. Lajnef, C. Guillemot, and P. Siohan, "Distributed coding of three sources using punctured turbo codes," in *IEEE 6th Workshop on Multimedia Signal Processing, 2004.*, Sep. 2004, pp. 307–310.

[21] A. W. Eckford and Wei Yu, "Rateless Slepian-Wolf codes," in *Conference Record of the Thirty-Ninth Asilomar Conference onSignals, Systems and Computers, 2005.*, Oct. 2005, pp. 1757–1761.

[22] D. Sejdinovic, R. J. Piechocki, A. Doufexi, and M. Ismail, "Fountain code design for data multicast with side information," *IEEE Transactions on Wireless Communications*, vol. 8, no. 10, pp. 5155–5165, Oct. 2009.

[23] T. Wang, W. Heinzelman, A. Seyedi, and A. Vosoughi, "Maximizing the lifetime of clusters with Slepian-Wolf coding," in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, Mar. 2010, pp. 2922–2925.

[24] T. Wang, A. Vosoughi, W. Heinzelman, and A. Seyedi, "Maximizing gathered samples in wireless sensor networks with Slepian-Wolf coding," *IEEE Transactions on Wireless Communications*, vol. 11, no. 2, pp. 751–761, Feb. 2012.

[25] C.-H. Chang, W.-C. Liao, H.-Y. Hsieh, and H.-J. Su, "Not every bit counts: Shifting the focus from machine to data for machine-to-machine communications," in *2012 Conference Record of the Forty Sixth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, Nov. 2012, pp. 581–585.

[26] W. Wang, J. Zhu, S. Zhang, and W. Zhou, "Tradeoff between efficiency and delay of distributed source coding for uplink transmissions in machine type communications," in *2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*, Oct. 2017, pp. 1–6.

[27] K. Yuen, B. Liang, and B. Li, "A distributed framework for correlated data gathering in sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 57, no. 1, pp. 578–593, Jan. 2008.

[28] T. S. Han, "Slepian-Wolf-Cover theorem for networks of channels," *Information and Control*, vol. 47, no. 1, pp. 67–83, Oct. 1980. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0019995880902843

[29] A. Roumy and D. Gesbert, "Optimal matching in wireless sensor networks," in *2007 IEEE International Symposium on Information Theory*, Jun. 2007, pp. 2116–2120.

[30] J. Barros and S. D. Servetto, "Network information flow with correlated sources," *IEEE Transactions on Information Theory*, vol. 52, no. 1, pp. 155–170, Jan. 2006.

[31] R. Cristescu, B. Beferull-Lozano, and M. Vetterli, "Networked Slepian-Wolf: theory, algorithms, and scaling laws," *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4057–4073, Dec. 2005.

[32] H. Mo, J. Chen, X. Lang, and J. Li, "Distributed source coding for utilization of inter/intra source correlation," *Signal Processing: Image Communication*, vol. 105, p. 116698, Jul. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092359652200042X

[33] P. Wang, C. Li, and J. Zheng, "Distributed data aggregation using clustered Slepian-Wolf coding in wireless sensor networks," in *2007 IEEE International Conference on Communications*, Jun. 2007, pp. 3616–3622.

[34] ——, "Combined data aggregation and encryption using clustered Slepian-Wolf coding for wireless sensor networks," in *IEEE GLOBECOM 2007 - IEEE Global Telecommunications Conference*, Nov. 2007, pp. 920–925.

[35] J. Zheng, P. Wang, and C. Li, "Distributed data aggregation using Slepian–Wolf coding in cluster-based wireless sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 5, pp. 2564–2574, Jun. 2010.

[36] Q. Shu, Q. Hu, J. Zheng, and N. Mitton, "A dependable Slepian-Wolf coding based clustering algorithm for data aggregation in wireless sensor networks," in *2013 International Conference on Wireless Communications and Signal Processing*, Oct. 2013, pp. 1–6.

[37] Z. Huang and J. Zheng, "A Slepian-Wolf coding based energy-efficient clustering algorithm for data aggregation in wireless sensor networks," in *2012 IEEE International Conference on Communications (ICC)*, Jun. 2012, pp. 198–202.

[38] W. Wang, J. Zhu, S. Zhang, and W. Zhou, "Tradeoff between compression ratio and decoding delay of distributed source coding for uplink transmissions in machine-type communication," *International Journal of Distributed Sensor Networks*, vol. 14, no. 7, p. 1550147718787109, Jul. 2018. [Online]. Available: https://doi.org/10.1177/1550147718787109

[39] Y. Yang, S. Guo, G. Liu, and Q. Wang, "Joint source coding rate allocation and flow scheduling for data aggregation in collaborative sensing networks," *Computer Networks*, vol. 175, p. 107269, Jul. 2020.

[40] Y. Yang, S. Guo, and Y. Yang, "Distributed optimal source coding rate allocation for data aggregation in wireless sensor networks," in *2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS)*, Dec. 2016, pp. 32–39.

[41] R. Amutha, G. G. Sivasankari, and K. R. Venugopal, "Node clustering and data aggregation in wireless sensor network using sailfish optimization," *Multimedia Tools and Applications*, Apr. 2023. [Online]. Available: https://doi.org/10.1007/s11042-023-15225-z

[42] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 26–36, Jul. 2012.

[43] L. C. M. Hurali and A. P. Patil, "Application areas of information-centric networking: State-of-the-art and challenges," *IEEE Access*, vol. 10, pp. 122 431–122 446, Nov. 2022.

[44] J. Guo, H. Gao, Z. Liu, F. Huang, J. Zhang, X. Li, and J. Ma, "ICRA: An intelligent clustering routing approach for UAV ad hoc networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 2, pp. 2447–2460, Feb. 2023.

[45] M. Sheela, R. Suganthi, S. Gopalakrishnan, T. Karthikeyan, K. J. Jyothi, and K. Ramamoorthy, "Secure routing and reliable packets transmission in MANET using fast recursive transfer algorithm," *Babylonian Journal of Networking*, vol. 2024, p. 78–87, Jun. 2024. [Online]. Available: https://mesopotamian.press/journals/index.php/BJN/article/view/427

[46] Y. Zhang, Y. Miao, X. Li, L. Wei, Z. Liu, K.-K. R. Choo, and R. H. Deng, "Efficient privacy-preserving federated learning with improved compressed sensing," *IEEE Transactions on Industrial Informatics*, vol. 20, no. 3, pp. 3316–3326, Mar. 2024.

[47] I. I. Al Barazanchi and W. Hashim, "Enhancing IoT device security through blockchain technology: A decentralized approach," *SHIFRA*, vol. 2023, p. 10–16, Feb. 2023. [Online]. Available: https://peninsula-press.ae/Journals/index.php/SHIFRA/article/view/14

[48] I. Ullah, G. Musa Raza, and B.-S. Kim, "Smart proactive caching: Paving the way for efficient caching in vehicular ICN," *IEEE Access*, vol. 12, pp. 150 213–150 228, Oct. 2024.

[49] H. Wu, Y. Fan, Y. Wang, H. Ma, and L. Xing, "A comprehensive review on edge caching from the perspective of total process: Placement, policy and delivery," *Sensors*, vol. 21, no. 15, Jul. 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/15/5033

[50] Z. Jia, Q. Liu, J. Zhou, X. Gu, Y. Zhang, B. Li, and W. Wang, "Optimal caching for partial-observation regime and beyond," *IEEE Transactions on Services Computing*, vol. 17, no. 6, pp. 4041–4054, Nov. 2024.

[51] S. Kruglik and A. Frolov, "An information-theoretic approach for reliable distributed storage systems," *Journal of Communications Technology and Electronics*, vol. 65, no. 12, pp. 1505–1516, Dec. 2020.

[52] 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.

[53] H. Cheng, C. Li, H. Xiong, and P. Frossard, "Optimal decentralized coded caching for heterogeneous files," in *2017 25th European Signal Processing Conference (EUSIPCO)*, Aug. 2017, pp. 2531–2535.

[54] Y. Deng and M. Dong, "Decentralized caching under nonuniform file popularity and size: Memory-rate tradeoff characterization," *IEEE/ACM Transactions on Networking*, vol. 32, no. 1, pp. 175–190, Feb. 2024.

[55] M. Cheng, L. Liu, J. Wang, and Q. Deng, "A novel centralized coded caching scheme for edge caching basestation," *Journal of Systems Architecture*, vol. 128, p. 102556, Jul. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1383762122001096

[56] J. E. Espozo-Espinoza, M. Fernández-Veiga, and F. Troncoso-Pastoriza, "Generalized hierarchical coded caching," *Journal of Network and Computer Applications*, vol. 232, p. 104027, Dec. 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804524002042

[57] M. J. Sojdeh, M. Letafati, S. P. Shariatpanahi, and B. H. Khalaj, "Secure multi-server coded caching," *Computer Networks*, vol. 253, p. 110715, Jul. 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128624005474

[58] W. Guan, K. Huang, X. Xie, J. Zhang, K. Cai, and X. Huang, "New results on coded caching in partially cooperative D2D networks," in *2024 IEEE International Symposium on Information Theory (ISIT)*, Jul. 2024, pp. 1556–1561.

[59] Z. Tian, L. Wang, L. Xu, C. Xu, and A. Fei, "Coded caching for reliable map dissemination in symbiotic communication aided emergency UAV systems," *IEEE Transactions on Cognitive Communications and Networking*, vol. 10, no. 5, pp. 1663–1677, Aug. 2024.

[60] K. Zheng, Y. Cui, X. Liu, X. Wang, X. Jiang, and J. Tian, "Asymptotic analysis of inhomogeneous information-centric wireless networks with infrastructure support," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 6, pp. 5245–5259, Feb. 2018.

[61] Y. Lin and C. Li, "Cooperative coded caching in internet of vehicles based on coded prefetching," in *IoT as a Service*, X. Chen, X. Wang, S. Lin, and J. Liu, Eds. Cham: Springer Nature Switzerland, Oct. 2024, pp. 396–413.

[62] F. Brunero and P. Elia, "Fundamental limits of combinatorial multi-access caching," *IEEE Transactions on Information Theory*, vol. 69, no. 2, pp. 1037–1056, Feb. 2023.

[63] 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.

[64] ——, "Correlation-aware distributed caching and coded delivery," in *2016 IEEE Information Theory Workshop (ITW)*, Sep. 2016, pp. 166–170.

[65] 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.

[66] P. 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.

[67] R. Graczyk and A. Lapidoth, "Gray-Wyner and Slepian-Wolf guessing," in *2020 IEEE International Symposium on Information Theory (ISIT)*, Aug. 2020, pp. 2189–2193.

[68] 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*, ser. Q2SWinet '22. New York, NY, USA: Association for Computing Machinery, Oct. 2022, p. 121–129. [Online]. Available: https://doi.org/10.1145/3551661.3561372

[69] ——, "Cache-aided networks with shared caches and correlated content under non-uniform demands," in *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*, Mar. 2023, pp. 301–304.

[70] P. Tan, K. Xie, and J. Li, "Slepian-Wolf coding using parity approach and syndrome approach," in *2007 41st Annual Conference on Information Sciences and Systems*, Mar. 2007, pp. 708–713.

[71] Q. Xie, Y. Li, S. Hu, Y. Zhu, and H. Wang, "Two heuristic algorithms for the minimum weighted connected vertex cover problem under greedy strategy," *IEEE Access*, vol. 10, pp. 116 467–116 472, Nov. 2022.

[72] Z. Luo, X. Guo, B. Wen, and J. Cao, "An evolution algorithm based on cloud model for 0-1 knapsack problem," in *2022 15th International Symposium on Computational Intelligence and Design (ISCID)*, Dec. 2022, pp. 147–150.

[73] S. Forrest, "Genetic algorithms," *ACM Computing Surveys*, vol. 28, no. 1, pp. 77–80, Mar. 1996. [Online]. Available: https://doi.org/10.1145/234313.234350

[74] A. Scheibenpflug and S. Wagner, "An analysis of the intensification and diversification behavior of different operators for genetic algorithms," in *Computer Aided Systems Theory - EUROCAST 2013*, R. Moreno-Díaz, F. Pichler, and A. Quesada-Arencibia, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 364–371.

[75] N. T. Son, J. Jaafar, I. A. Aziz, and B. N. Anh, "Meta-heuristic algorithms for learning path recommender at MOOC," *IEEE Access*, vol. 9, pp. 59 093–59 107, Apr. 2021.

[76] K. Tomitagawa, A. Anuntachai, S. Chotipant, O. Wongwirat, and S. Kuchii, "Performance measurement of energy optimal path finding for waste collection robot using ACO algorithm," *IEEE Access*, vol. 10, pp. 117 261–117 272, Nov. 2022.

[77] E. Alhenawi, R. A. Khurma, A. A. Sharieh, O. Al-Adwan, A. A. Shorman, and F. Shannaq, "Parallel ant colony optimization algorithm for finding the shortest path for mountain climbing," *IEEE Access*, vol. 11, pp. 6185–6196, Jan. 2023.

[78] M. Dorigo and T. Stützle, *The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances.* Boston, MA: Springer US, 2003, pp. 250–285.

[79] M. Dorigo and L. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, Apr. 1997.

[80] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, Mar. 1955. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800020109

[81] Y. Cao, "Hungarian algorithm for linear assignment problems (v2.3)," Sep. 2011. [Online]. Available: (https://www.mathworks.com/matlabcentral/fileexchange/20652-hungarian-algorithm-for-linear-assignment-problems-v2-3

[82] X. Zhu, L. Zhang, and Y. Liu, "A distributed joint source-channel coding scheme for multiple correlated sources," in *2009 Fourth International Conference on Communications and Networking in China*, Aug. 2009, pp. 1–6.

[83] S. Gao, "Joint distributed source and network coding for correlated information multicasting," in *2011 6th International ICST Conference on Communications and Networking in China (CHINACOM)*, Aug. 2011, pp. 698–702.

[84] F. Pujaico and J. Portugheis, "Optimal rate for joint source-channel coding of correlated sources over orthogonal channels," *IEEE Communications Letters*, vol. 19, no. 1, pp. 22–25, Jan. 2015.

[85] B. Lu and J. Garcia-Frias, "Analog joint source-channel coding for transmission of correlated senders over separated noisy channels," in *2015 49th Annual Conference on Information Sciences and Systems (CISS)*, Mar. 2015, pp. 1–5.