IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 10 February 2025, accepted 10 March 2025, date of publication 12 March 2025, date of current version 25 March 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3550790

RESEARCH ARTICLE

Path-Based Clustering Approaches for a Hybrid Slepian-Wolf Compression and Coded Caching System

BENJAMIN ROSEN[1], ADNAN M. ABU-MAHFOUZ[2], (Senior Member, IEEE), AND LING CHENG[1], (Senior Member, IEEE)

[1]School of Electrical and Information Engineering, University of the Witwatersrand, Johannesburg 2000, South Africa
[2]Council for Scientific and Industrial Research, Pretoria 0001, South Africa

Corresponding author: Ling Cheng (ling.cheng@wits.ac.za)

This work was supported in part by the National Research Foundation of South Africa under Grant CHN22111571052.

ABSTRACT Combining caching with source coding, a hybrid content delivery system further facilitates the shift towards Information-Centric Networks. This is a promising technology heralded as the next phase in network design. However, finding the optimal balance between the source coding gains and the computational complexity is itself an NP-hard problem. By modelling the problem using a path-based approach, this paper outlines iterative algorithms that can be tuned to provide control over this trade-off. 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 Ant Colony Optimisation family of meta-heuristic algorithms is adapted to solve this problem, providing a benchmark that outperforms the Genetic Algorithm presented in prior work. 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.

INDEX TERMS Ant colony optimization, clustering algorithms, content distribution networks, heuristic algorithms, Information-Centric Networking, information entropy, information theory, iterative algorithms, mutual information, source coding.

I. INTRODUCTION

Latest-generation networks are struggling to keep up with the exponential increase in users' demands for content. Accordingly, research in network structure is shifting towards Information-Centric Networks (ICNs) [1]. This new regime requires a focus on distributed technique, which demands a high degree of flexibility in terms of routing structure [2], [3] while maintaining a strong level of security [4], [5].

An integral component of ICNs is caching, allowing data to be stored at the user end during hours when the network usage is low [6]. 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. As noted in [1] 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.

In [7], a Slepian-Wolf Coded Caching (SW/CC) hybrid system is presented as a means to combat this issue. The CC method allows end users to store a compressed version of the content locally [8]. SW coding further increases the coding gains when compressing the information, if they are correlated [9]. In that paper, an iterative and meta-heuristic approach are developed to reduce the complexity of the SW coding by clustering the files into compressed and uncompressed sub-libraries. However, it was found that, while less computationally complex, the iterative algorithm

The associate editor coordinating the review of this manuscript and approving it for publication was Qilian Liang.

produced worse performing results as compared to the meta-heuristic one.

As a result, this paper aims to improve on these results, showing how better-performing clusters can be obtained, albeit at the cost of increased time complexity, while still running in polynomial time.

A. BACKGROUND

Caching is a fundamental tool to improve the efficiency in networks. It aims to reduce the strain on networks by storing partial information closer to the users, when the demands on the networks are not as high. Then, when users request information, the amount of data required to fulfil their requests is reduced, as repeat requests by different users can be achieved by sending the previously stored information instead of fetching it from the origin server. 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 [10]. So too, emphasis is placed on distributed methods, in order to maintain flexibility and scalability when implementing caching and coding in real-world scenarios [11], [12].

Hassanzadeh et al. [13] first described a system that utilises two different types of compression paradigms for content delivery optimisation.

The first is Distributed Source Coding (DSC), which enables correlated information to be compressed. Slepian and Wolf, in their foundational paper [9], outline the feasibility of this type of source coding. Significantly, this can be achieved optimally without the need for communication between the distributed nodes. As a result, this type of coding is particularly well suited to Wireless Sensor Networks (WSNs) and is widely studied in literatures [14], [15], [16], and [17].

After SW compression, the second phase uses Coded Caching to store files at the user end. The goal is to minimise the amount of data sent when users request files during peak hours. Since the demands of the users are unknown when caching the files, it is necessary to intelligently distribute the data, such that any demand can be fulfilled with bounds on the rates. Maddah-Ali and Niesen first described such a method, which codes the files with each other by XOR'ing them together [8]. When requesting a specific file, the server supplies the missing information, which can be XOR'ed with the cached information to retrieve the requested data. This technique has recently found application in ICNs, proving useful in scenarios with both sparse [18] and dense [6] user distributions. So too, it can be set up in a centralised [19] or hierarchical [20] manner, allowing for scalability if required. This is especially so, since it has a low complexity (relying primarily on the XOR function) and is well defined for multiple files. As a result, it would be able to handle caching within a decentralised context, which is well suited to ICNs. Additionally, it provides for security and integrity of the cached information [21]. These features make it highly compatible with ICN requirements. More recently, this has been extended and analysed for co-operative implementations such as Device to Device [22] and vehicular [23] communications

When the two coding methods are combined, their effects are complementary, since their approaches are independent. However, there are limitations to this hybrid approach. In particular, SW coding becomes prohibitively complex with the increase in the number of files compressed at the same time [24].

To this end, the authors in [7] show that the SW complexity can be bounded by limiting the number of files compressed. Thus, the library of files is divided into two categories. The first is compressed using SW, while the other is left uncompressed. Since the two stages are independent, the subdivision can be performed without the knowledge of the CC phase. As a result, the two sub-libraries can then be combined before going through the CC phase, which treats them as files of differing lengths using a method such as that given in [25]. The resulting hybrid system is summarised in Figure 1.

In that paper, it was further shown that finding the optimal clustering is an NP-hard problem. Although clustering algorithms have been devised for the SW case [24], [26], [27], they rely on estimations and assumptions about the correlation, which do not take into account real-world scenarios.

Consequently, two sub-optimal algorithms are presented in [7]. One is a novel iterative and greedy algorithm called GISGEM, while the other is an adaptation of the Genetic Algorithm (GA). It was found that, although less complex in terms of time, the quality of the GISGEM results were not as good as those found by the GA. Furthermore, there is no method to test the potential optimality of a clustering solution.

B. CONTRIBUTIONS

As a result of the above, this paper has three main contributions:

1) We present a novel path-based iterative approach to solve the SW/CC clustering problem that allows for a trade-off between the quality of results and time complexity. The iterative nature of this approach makes the performance more predictable and has fewer parameters to tune than the meta-heuristic approaches, while providing comparable performance.
2) Ant Colony Optimisation (ACO) is applied to this problem as an additional benchmark. It is shown to find better quality results than the GA method at the expense of complexity.
3) A necessary (but not sufficient) condition is derived for optimality. This is used to further improve the performance of an existing solution obtained by other optimisation methods by exchanging files. Combining this method with the iterative approach listed above, it is found to outperform previous solutions to this optimisation problem, finding close-to-optimal clusterings.

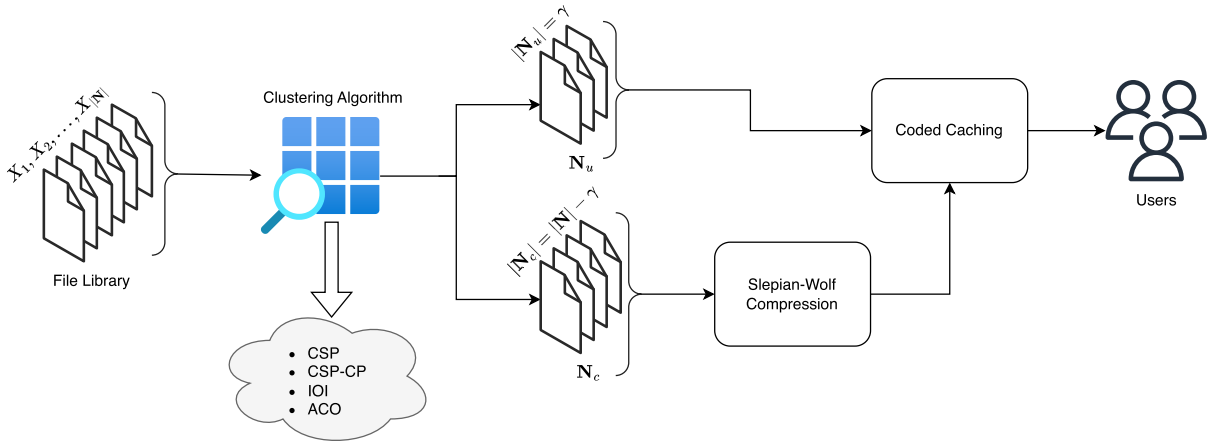


FIGURE 1. An overview of the hybrid SW/CC system. The proposed improvements are summarised in the clustering algorithm list.

C. LAYOUT

The rest of this paper is structured as follows: Section II gives the necessary background for the mathematical structure and modelling, while Section III uses these models to improve the greedy iterative algorithm and Section IV outlines the meta-heuristic approach. The theoretical complexity of the two approaches is given in Section V. In Section VI, the newly proposed iterative algorithms are demonstrated using a numerical example. Simulation results can be found in Section VII, with further discussion and future research directions in Section VIII. Finally, Section IX concludes this paper.

II. PRELIMINARIES

Let $\mathbf{N}$ be a set of files, modelled as information sources $\{X_1, X_2, \ldots, X_{|\mathbf{N}|}\}$. Without loss of generality, the sources produce binary information that are i.i.d and ergodic. The contents of each file are correlated with one another according to some distribution $p(x_1, x_2, \ldots, x_{|\mathbf{N}|})$.

As depicted in Figure 1, the main thrust of this paper is the clustering algorithm used to split the files into two disjoint sub-libraries. The first, denoted as $\mathbf{N}_c$, will be compressed using SW coding, while the other, $\mathbf{N}_u$, will be left uncoded. The sub-libraries are combined and passed into the CC block, which ensures the information is cached optimally. Let $|\mathbf{N}_u| = \gamma$ and $|\mathbf{N}_c| = |\mathbf{N}| - \gamma$. Consequently, it is required that $\mathbf{N}_c, \mathbf{N}_u \subset \mathbf{N}$ and $\mathbf{N}_c \cap \mathbf{N}_u = \emptyset$.

The main optimisation goal of the clustering algorithm is to find the best partitioning of the library, such that the coding gains are maximised for a fixed sub-library size. More formally:

$$\mathbf{N}_u^* = \arg\max_{\mathbf{N}_u} H(\mathbf{N}) - H(\mathbf{N}_u) \tag{1}$$

$$\text{s.t.} \quad \mathbf{N}_u \subset \mathbf{N}, |\mathbf{N}_u| = \gamma \tag{2}$$

where $H(\mathbf{N})$ and $H(\mathbf{N}_u)$ are the total entropies of the sources in the sets $\mathbf{N}$ and $\mathbf{N}_u$ respectively. This is an NP-hard problem, where the search space is $\binom{|\mathbf{N}|}{\gamma}$ and thus increases exponentially [7]. The GISGEM and GA algorithms are two close-to-optimal approaches, but the results can be improved.

In order to describe the improvements to the GISGEM algorithm accurately, a new model of the search algorithm is required. Consequently, the necessary mathematical preliminaries are now defined and explained.

Let $G = (V, A, t)$ be a directed graph representing a t-ary tree, where V is a set containing all of the 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 $i \in \{1, 2, \ldots, t^d\}$. In 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 $a \in A$, $d \in \{1, 2, \ldots, \gamma\}$, $i \in \{1, 2, \ldots, t^d\}$, $j \in \{t(i-1)+1, t(i-1)+2, \ldots, ti\}$. In this paper, 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 $\mathbf{C}_{v_i^d} := \{v_j^{d+1} \in V | v_i^d \hookrightarrow v_j^{d+1}\}$ 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 $c_l \to v_j^{d+1}$ and $c_m \to v_k^{d+1}$, $c_l, c_m \in \mathbf{C}_{v_i^d}$, 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 $|\mathbf{P}_{v_i^d}| = d$. 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 $X_k \in N$. 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 $\mathbf{P}_{v_i^d}$,

TABLE 1. A summary of the terms used in this paper with their definitions.

Term	Description
Entropy	The randomness of an information source. Lower entropy implies that the same information can be transmitted with less physical data.
Mutual Information	The amount of shared information between two or more information sources. Larger areas of mutual information shows where information sources are highly correlated, implying that a reduction of total entropy is possible.
Information-Centric Networks	A new paradigm for content delivery. Instead of a client-server architecture, where the information is stored and retrieved from servers, ICNs store the data within the network itself, moving the data closer to end users.
Slepian-Wolf Coding	A source coding technique that leverages high correlation between information sources to reduce their total entropy.
Coded Caching	A low complexity and flexible caching technique that exploits global caching gains to dramatically reduce the rate required to transmit the data during the delivery phase.
Meta-Heuristic Optimisation	A family of optimisation techniques, inspired by real-world phenomena, that can be used to solve optimisation problems. Although they are performant for certain problems, they are non-deterministic and are not guaranteed to be optimal.

TABLE 2. A summary of the notations used in this paper with descriptions.

Notation	Description
$\mathbf{N}$	A set containing all of the files in the system.
X_i	A single file, modelled as an information source that produces binary information that is i.i.d and ergodic.
$\mathbf{N}_c$	A partition of $\mathbf{N}$, containing all of the files to be coded using SW coding.
$\mathbf{N}_u$	The files not included in $\mathbf{N}_c$, of size γ. The files in this set will not be encoded using SW coding.
v_i^d	A vertex in the tree representing the solutions found using the iterative algorithms, at depth d with index i. Each vertex corresponds to a single information source in $\mathbf{N}$. A vertex is unique, but not necessarily the source that it represents.
$\mathbf{C}_{v_i^d}$	A set containing all of the children vertices of vertex v_i^d.
$\mathbf{P}_{v_i^d}$	A set representing a path of vertices. It is constructed by following the graph from vertex v_i^d and moving to the parent of each successive vertex until the root node v_0.
$f^{\mathcal{X}}(\cdot)$	A function that maps an element (e.g. a vertex or path element) to the information source that it represents.
$\mathbf{S}_{v_i^d}$	A set that contains all legal information sources that can be associated with the children vertices of the reference vertex v_i^d.
$\mathbf{E}_{v_i^d}$	A set containing vertices that are excluded from the selection pool when using the CSP-CP variant of the CSP algorithm.
$\mathbf{M}_i$	A set containing the m best vertices in iteration i, based on the performance of the paths starting from these vertices.

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:

$$H(\mathbf{P}_{v_i^d}) = \sum_{n=1}^{d} H(v^n_{\lceil i/t^{d-n} \rceil} | v^{n-1}_{\lceil i/t^{d-n+1} \rceil}, v^{n-2}_{\lceil i/t^{d-n+2} \rceil}, \ldots, v^1_{\lceil i/t^{d-1} \rceil}) \tag{3}$$

$$= \sum_{n=1}^{d} H(p_n | p_{n-1}, p_{n-2}, \ldots, p_1) \tag{4}$$

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

III. 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.

A. CHAMPION SELECTION PROCEDURE (CSP)

The GISGEM algorithm in [7] selects a single node at each iteration by maximising the conditional entropy gain at that iteration. Using the definitions in Section II, 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 more optimal 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. 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 1 and summarised in Algorithm 1.

Procedure 1 (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:

$$H(v_i^1) \geq H(v_j^1), \quad \forall i, j \in \{1, 2, \ldots, t\}, i > j \tag{5}$$

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:

$$H(v_j^d|\mathbf{P}_{v_i^{d-1}}) \geq H(v_k^d|\mathbf{P}_{v_i^{d-1}}), \quad \forall j,k \in \{t(i-1) + 1, t(i-1)+2, \ldots, ti\}, j > k \tag{6}$$

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:

$$\mathbf{S}_{v_i^{d-1}} := \mathbf{N} \setminus f^{\mathcal{X}}(\mathbf{P}_{v_i^{d-1}}) \tag{7}$$

As defined above, $\mathbf{C}_{v_i^{d-1}}$ contains the children of v_i^{d-1}. Let $\mathbf{C} := \bigcup \mathbf{C}_{v_i^{d-1}}, \forall v_i^{d-1} \in \mathbf{M}_{d-1}$ 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) and (6), 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.

Algorithm 1 Champion Selection Procedure (CSP)

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 (6)
      $V \leftarrow V \cup \mathbf{C}_{v_i^{d-1}}$
      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

B. CSP VARIATIONS

The CSP approach broadens the search range of the GISGEM algorithm. Nevertheless, there 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 1 (Champion Selection Procedure with Ancestor Guarantee - CSP-AG:) Let the children set $\mathbf{C}$ be created according to Procedure 1. 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, $i \in \{1, 2, \ldots, \text{AG}\}$. The remaining $m - \text{AG}$ children to include are chosen as in Procedure 1. If this updated procedure is followed, then it will always produce a result greater or equal to the GISGEM procedure.

Proof: If AG $\geq$ 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 (6). 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 1, this means that the set contains vertices that evaluate to the same or higher entropy than v_i^d.

Lemma 2 (Champion Selection Procedure with Convergence Prevention - CSP-CP:) Let the general selection procedure follow that outlined in Procedure 1. However, let

the construction of $\mathbf{S}_{v_i^{d-1}}$ be updated as follows:

$$\mathbf{S}_{v_i^{d-1}} := \mathbf{N} \setminus \{ \bigcup_{n=d-1}^{1} f^{\mathcal{X}}(\mathbf{E}_{v^n_{\lceil i/t^{d-1-n} \rceil}}) \} \tag{8}$$

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. $v_k^{d-1} \hookrightarrow v_i^d$, $v_k^{d-1} \hookrightarrow v_j^d$. 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 2, 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. □

C. 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 [7], a stopping condition was 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 3: Let $\mathbf{P}_{v_{\text{ref}}}$ be a path of length γ with known entropy H_{ref}. If the following condition is not satisfied for a given path $\mathbf{P}_{v_i^d}$, then the path cannot be optimal:

$$H_{\text{ref}} - H(\mathbf{P}_{v_i^d}) > (\gamma - d) H(v_i^d | \mathbf{P}_{v_i^d} \setminus v_i^d) \tag{9}$$

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

$$H(v_j^l | \mathbf{P}_{v_j^l} \setminus v_j^l) \leq H(v_i^d | \mathbf{P}_{v_i^d} \setminus v_i^d) \tag{10}$$

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 4: 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:

$$|\mathbf{S}_{v_i^{d-1}}| < \gamma - d \tag{11}$$

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. □

D. OPTIMALITY TEST

Despite the modifications to the GISGEM selection procedure outlined in Section III-A, as well as the variations in Section III-B, it is still possible that the most optimal result is not obtained. In the following Lemma, a necessary condition for optimality is derived.

Lemma 5 (optimality condition): A necessary condition of optimality for a given path $\mathbf{P}_{v_i^d}$ is that each $p_j \in \mathbf{P}_{v_i^d}$ is the solution to the optimisation condition:

$$p_k^* = \arg\max_{p_k} H(p_k | \mathbf{P}_{v_i^d} \setminus p_k) \tag{12}$$

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 (4) 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 6: The optimality test in Lemma 5 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:

$$H(X_i|X_j) = H(X_j|X_i) = 0 \quad \forall X_i \in \mathbf{N}_1, X_j \in \mathbf{N}_2 \tag{13}$$

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 5, sources from $\mathbf{N}_2$ will never be suggested, since their conditional entropy will always be 0 as shown in (13).□

Corollary 1 (Iterative Optimality Improvement - IOI): In the optimality test described in Lemma 5, if any $p_k \in$

$\mathbf{P}_{v_i^d}$ is not the solution to optimisation condition (12), 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:

$$H(\mathbf{P}^*_{v_i^d}) > H(\mathbf{P}_{v_i^d}) \tag{14}$$

Proof: The entropy term given in (12) is the last term in the total entropy calculation in (3) and (4). 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 (4) 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 (4) 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 5 can repeatedly be used to improve any grouping result. However, Lemma 6 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 5 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:

$$\mathbf{p}_k^* = \arg\max_{\mathbf{p}_k} H(\mathbf{p}_k|\mathbf{P}_{v_i^d} \setminus \mathbf{p}_k) \tag{15}$$

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 5, 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 $\mathbf{p}_k$. □

IV. PATH-BASED META-HEURISTIC APPROACH

Encouraged by the modifications made to the iterative search algorithm in [7] using a path-based mathematical model, the meta-heuristic approach is also updated using this outlook. The Ant Colony Optimisation (ACO) is a family of meta-heuristic algorithms whose efficacy in path-finding problems is well known [28], [29], [30]. 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 [31]. In this paper, 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 [32]. A summary of the algorithm is now given, together with the modifications that are made to tailor the algorithm to the considerations of this paper.

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 [31]. 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 maximum 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:

$$p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in \mathbf{N}_i^k} [\tau_{il}(t)]^\alpha [\eta_{ij}]^\beta} \tag{16}$$

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 (16). 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 (16).

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:

$$\tau_{ij}(t) = (1-\xi)\tau_{ij}(t) + \xi\tau_0 \tag{17}$$

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:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \rho\Delta\tau_{ij}^{gb}(t) \tag{18}$$

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 Stall Generations (SG). 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 2.

Algorithm 2 Ant Colony System Optimisation Approach

```
Initialise the distance and pheromone states:
for all η_ij ∈ D do                          ▷ Create distance matrix
    η_ij ← H(X_j|X_i)
end for
for all τ_ij ∈ P do                          ▷ Create pheromone matrix
    τ_ij ← |N|C_nn   ▷ Nearest Neighbour path performance
end for
while Stopping conditions have not been met do
    while Path length < γ do
        for each ant k do
            Construct attractiveness set according to (16)
            With probability q_0:
            NextNode ← max_{j∈N_i^k}(p_ij^k(t))
            Otherwise:
            NextNode ← random biased choice using (16)
            P_k ← P_k ∪ NextNode   ▷ Update path of ant k
            τ_ij(t) ← (1−ξ)τ_ij(t) + ξτ_0   ▷ Update arc's
                pheromone
        end for
    end while
    Evaluate performance of chosen paths
    Update global best
    for all arc ij ∈ global best path do
        τ_ij ← (1−ρ)τ_ij + ρΔτ_ij^gb
    end for
end while
```

V. COMPLEXITY

In [7], 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, 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 -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 [7], 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 5 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 I, then the number of times this algorithm calculates the entropy is approximately $\binom{\mathbf{N}}{2} + kI$, giving a complexity of $O(|\mathbf{N}|^2)$.

VI. NUMERICAL EXAMPLE

To demonstrate how the CSP algorithm and its variants function, the algorithm is applied to an example. Consider a system $|\mathbf{N}| = 5$, $\gamma = 3$, where the correlations between nodes $\{X_1, X_2, \ldots, X_5\}$ are represented as Mutual Information Areas (MIAs) as given in Table 6. 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-AG=2 (i.e. 2 guaranteed ancestors). The results are shown in Figures 2-4.

The first step, which is the same for all algorithms including the GISGEM algorithm, involves calculating $H(X_i)$, $i \in \{1, 2, \ldots, 5\}$. For example, the value of $H(X_1) = 169$ bits

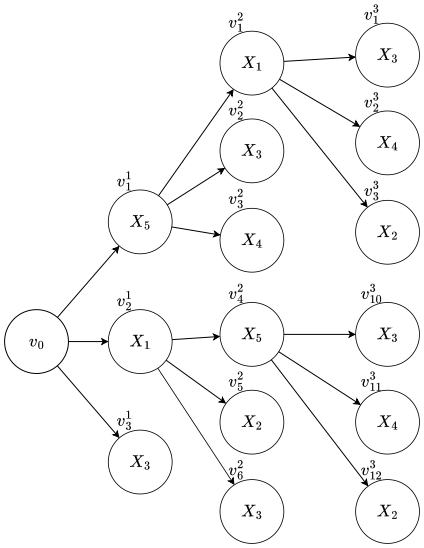


FIGURE 2. The resulting graph after applying the CSP algorithm to the numerical example.

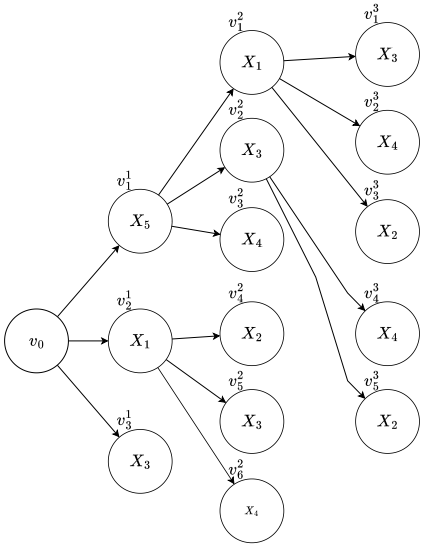


FIGURE 3. The resulting graph after applying the CSP-CP algorithm to the numerical example.

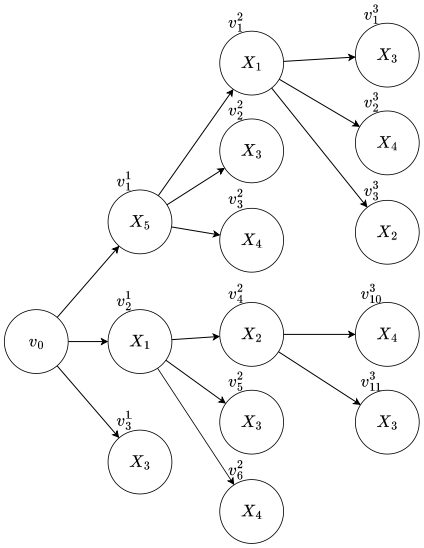


FIGURE 4. The resulting graph after applying the CSP-CP algorithm to the numerical example with 2 guaranteed ancestors.

is determined by summing the MIA values that have a binary representation `1XXXX`, while $H(X_1, X_2) = 250$ bits is obtained 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 -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 -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 have paths in the final iteration. Since the -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, X_3 are already selected and X_1 is excluded based on the -CP rule. So too, based on the branch and bound conditions described in Section III-C, 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 -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 to 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-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-AG=2 algorithms produced the same result, the CSP algorithm only produced 3 unique paths in the final selection, whereas the CSP-CP-AG=2 algorithm has 5. As proved in Lemma 1, the CSP-CP-AG=2 version guarantees that it will find a combination at least as good as the GISGEM approach. In contrast to this, although

the CSP-CP algorithm found a better result in this example, it is not always guaranteed to do so. 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-AG=2 variant and it was not able to find a better result than the plain CSP.

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.

VII. 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 benchmark algorithms from [7]. 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.

A. 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.

1) CSP RESULTS

In the CSP approach, m and t control the breadth of the search algorithm, the -AG variant prevents the algorithm from diversifying too much and the -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 while the time complexity results are shown in Figure 6. For both

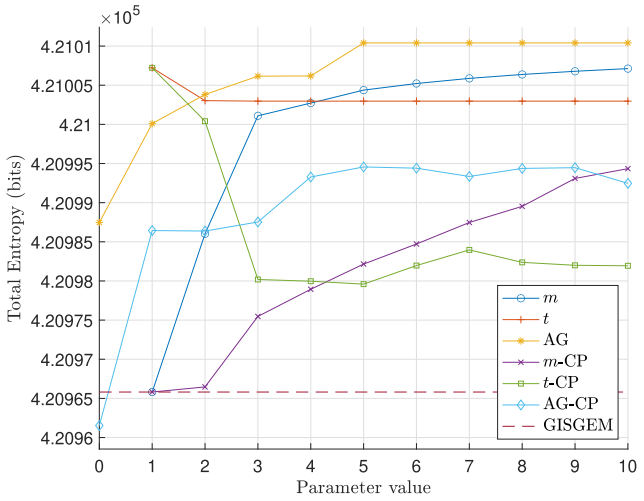


FIGURE 5. A comparison of the different variables and variants for the CSP algorithm and their effect on performance. $|\mathbf{N}| = 16$, $\gamma = 8$.

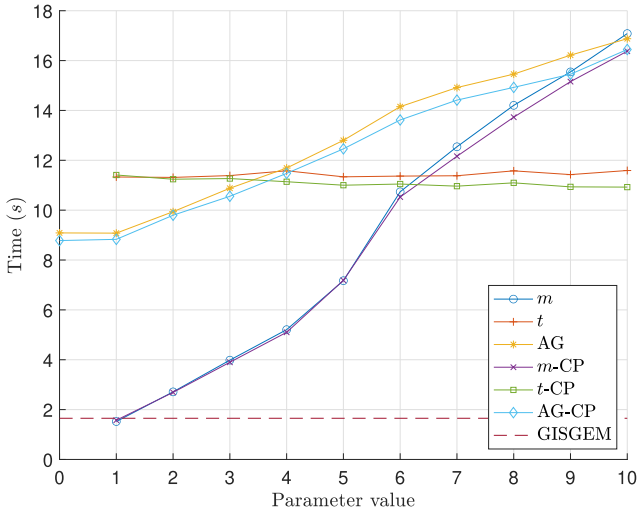


FIGURE 6. A comparison of the different variables and variants for the CSP algorithm and their effect on time complexity. $|\mathbf{N}| = 16$, $\gamma = 8$.

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 non -CP variant across the different tuning variables. This is a somewhat surprising result, since the -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 -CP variant is better overall in terms of time complexity. This is owing to the ability for the -CP variant to terminate paths early based on the branch and bound considerations discussed in Section III-C.

Another notable result is that all variants performed equal to or better than the GISGEM algorithm, with the exception of the -CP variant when the number of guaranteed ancestors is 0. As mentioned in Section III-B, the -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 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 non -CP variant 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 -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 non -CP variant, 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 7, the results are broken down in terms of different combinations of AG and m. For readability, the minimum value (at $\{AG, m\} = \{0, 2\}$) 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 $AG = m$ 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 $m/2 < AG < m$, although there are some outliers (such as $AG = 1$ 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 to keep the time complexity low. The value of t should be kept small to avoid providing too much diversity, while the -CP variant performs worse but has a guarantee on the diversity of the results.

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 it was experimentally determined to not have a major influence on the results. Owing 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

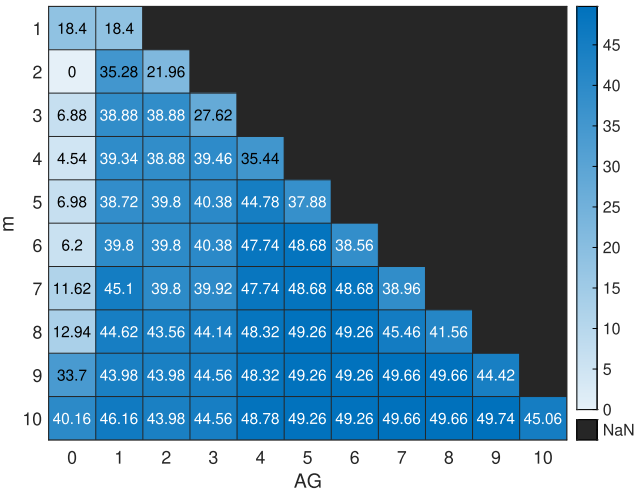


FIGURE 7. A combination-wise comparison of AG and m on the performance of the CSP algorithm.

TABLE 3. 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

experiment values to $|\mathbf{N}| = 14$ and $\gamma = 7$, giving a total of 3432 possible combinations.

The results of the performance and time complexity are depicted in Figures 8 and 9 respectively. The ranges of the tested variables and a description of their effect on performance and time complexity are shown in Table 3. 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 V, 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 $\{\alpha, \beta\} = \{1, 0.6\}$ featured most regularly in the best performing configurations.

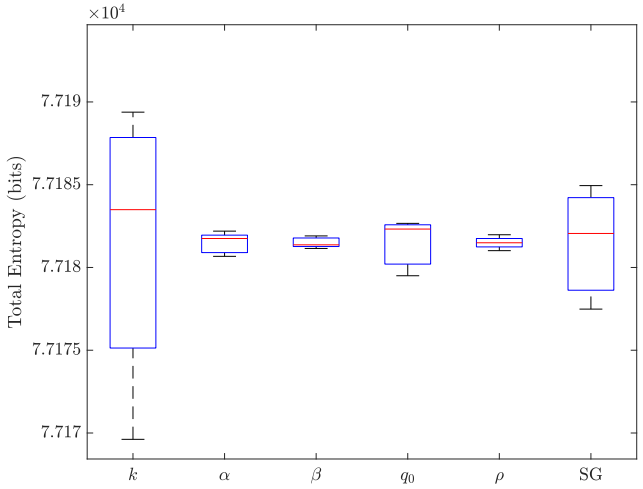


FIGURE 8. A comparison of the different variables for the ACO approach and their effect on performance. $|\mathbf{N}| = 14$, $\gamma = 7$.

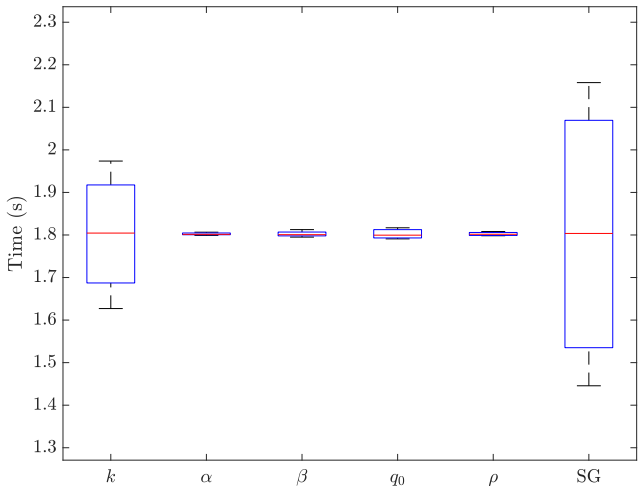


FIGURE 9. A comparison of the different variables for the ACO approach and their effect on time complexity. $|\mathbf{N}| = 14$, $\gamma = 7$.

B. ALGORITHM COMPARISON RESULTS

The different algorithms are compared with one another using the best values determined in Section VII-A. In this experiment, $|\mathbf{N}| = 18$ and $\gamma = \{1, 2, \ldots, 17\}$. The GA approach from [7] 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. This allows the datasets to mimic real-world correlation scenarios. Table 4 lists the algorithms tested as well as the variable values used for each one.

Figure 10 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.

TABLE 4. The algorithms used in the algorithm comparison test, together with their associated tuned variables.

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

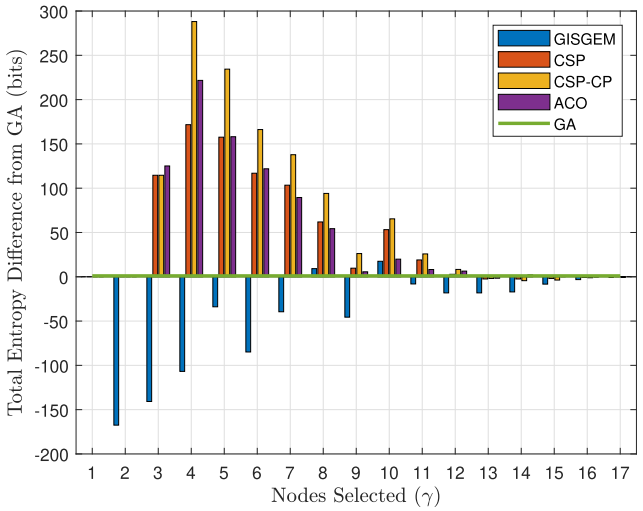


FIGURE 10. Performance results for each of the algorithms, as compared to the GA.

Figure 11 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 V, $O_{\text{CSP}}/O_{\text{GISGEM}} = 4.19$ 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 4 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.

VIII. DISCUSSION AND FUTURE WORK

All of the algorithms are found to converge, based on the empirical evidence. The two meta-heuristic approaches have a Stall Generations variable, which determines how many iterations to wait, in which the result does not change, before

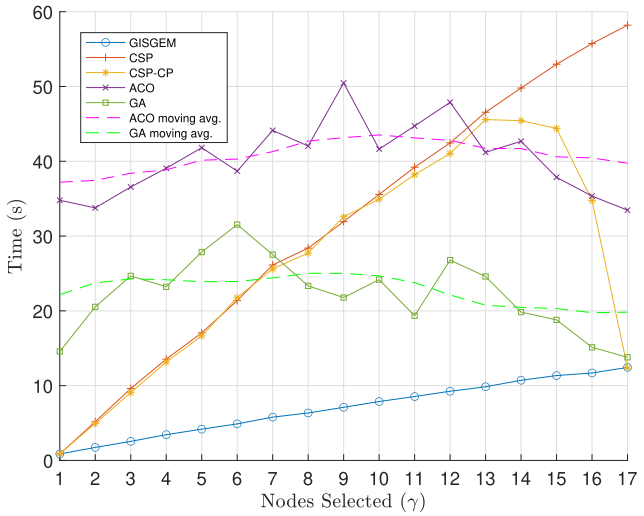


FIGURE 11. Time complexity for each of the algorithms.

terminating. This value was tuned so that the algorithms converge in a reasonable time, as shown in Figure 11. 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 (14) 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, together with the average time and the average of the performance values from Figure 10.

The worst performing algorithm is the GISGEM approach from [7], which still produces clusters that perform higher than the 98th percentile. However, it is clear that the new approaches discussed in this paper 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 -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 **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 -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 VII-B, the algorithms have specific regimes in which they perform better than others.

Applying the IOI approach to a random genome produced 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) with approximately the same time complexity. However, this negates the predictable nature of the iterative approaches, 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 results 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 clustering optimisation problem in a hybrid SW/CC system.

At present, 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

TABLE 5. A summary of the performance of the different algorithms (newly proposed algorithms in bold).

Algorithm	Avg. Performance vs. GA (bits)	Average 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**

TABLE 6. An Example of mutual information area 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

current algorithms. This is expected to further improve the quality of the results produced and may positively affect the convergence rate.

The CSP and ACO approaches still only look at partitioning the library into two sub-libraries. However, this work should be extended to multiple sub-libraries. Each one could be compressed using SW, such that the overall complexity is kept low while maximising the compression gains. More work needs to be done to quantify the size of the search space, update the objective functions and determine the approach to implementing the CSP in this new scenario.

IX. CONCLUSION

Two new algorithms are presented in this paper, 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 taking a tree-based model and path-based view of finding solutions in the search space. In particular, the CSP approach enables a trade-off between quality and time complexity, 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. These algorithms perform well when compared to benchmark meta-heuristic approaches. This paper builds and improves on previous work, 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.

APPENDIX
MUTUAL INFORMATION AREAS USED IN THE NUMERICAL EXAMPLE

See Table 6.

REFERENCES

[1] 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. 122431–122446, 2022.

[2] 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 Trans. Intell. Transp. Syst.*, vol. 24, no. 2, pp. 2447–2460, Feb. 2023.

[3] M. S. 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 J. Netw.*, vol. 2024, pp. 78–87, Jun. 2024. [Online]. Available: https://mesopotamian.press/journals/index.php/BJN/article/view/427

[4] 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 Trans. Ind. Informat.*, vol. 20, no. 3, pp. 3316–3326, Mar. 2024.

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

[6] 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. 150213–150228, 2024.

[7] 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. 12323–12334, 2024.

[8] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.

[9] D. Slepian and J. Wolf, "Noiseless coding of correlated information sources," *IEEE Trans. Inf. Theory*, vol. IT-19, no. 4, pp. 471–480, Jul. 1973.

[10] 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, p. 5033, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/15/5033

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

[12] S. Kruglik and A. Frolov, "An information-theoretic approach for reliable distributed storage systems," *J. Commun. Technol. Electron.*, vol. 65, no. 12, pp. 1505–1516, Dec. 2020.

[13] P. Hassanzadeh, A. M. Tulino, J. Llorca, and E. Erkip, "Rate-memory trade-off for caching and delivery of correlated sources," *IEEE Trans. Inf. Theory*, vol. 66, no. 4, pp. 2219–2251, Apr. 2020.

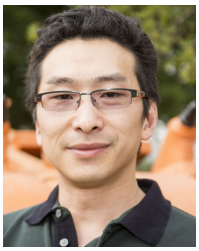
[14] D. Schonberg, K. Ramchandran, and S. S. Pradhan, "Distributed code constructions for the entire slepian–wolf rate region for arbitrarily correlated sources," in *Proc. Data Compress. Conf. (DCC)*, Mar. 2004, pp. 292–301.
[15] A. Roumy and D. Gesbert, "Optimal matching in wireless sensor networks," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun. 2007, pp. 2116–2120.
[16] K. Yuen, B. Liang, and L. Baochun, "A distributed framework for correlated data gathering in sensor networks," *IEEE Trans. Veh. Technol.*, vol. 57, no. 1, pp. 578–593, Jan. 2008.
[17] H. Mo, J. Chen, X. Lang, and J. Li, "Distributed source coding for utilization of inter/intra source correlation," *Signal Process., Image Commun.*, vol. 105, Jul. 2022, Art. no. 116698. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092359652200042X
[18] 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 Trans. Cognit. Commun. Netw.*, vol. 10, no. 5, pp. 1663–1677, Oct. 2024.
[19] M. Cheng, L. Liu, J. Wang, and Q. Deng, "A novel centralized coded caching scheme for edge caching basestation," *J. Syst. Archit.*, vol. 128, Jul. 2022, Art. no. 102556. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1383762122001096
[20] J. E. Espozo-Espinoza, M. Fernández-Veiga, and F. Troncoso-Pastoriza, "Generalized hierarchical coded caching," *J. Netw. Comput. Appl.*, vol. 232, Dec. 2024, Art. no. 104027. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804524002042
[21] M. J. Sojdeh, M. Letafati, S. P. Shariatpanahi, and B. H. Khalaj, "Secure multi-server coded caching," *Comput. Netw.*, vol. 253, Nov. 2024, Art. no. 110715. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128624005474
[22] W. Guan, K. Huang, X. Xie, J. Zhang, K. Cai, and X. Huang, "New results on coded caching in partially cooperative D2D networks," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2024, pp. 1556–1561.
[23] Y. Lin and C. Li, "Cooperative coded caching in Internet of Vehicles based on coded prefetching," in *Proc. IoT Service*, X. Chen, X. Wang, S. Lin, and J. Liu, Eds., Cham, Switzerland: Springer, Oct. 2024, pp. 396–413.
[24] Y. Yang, S. Guo, G. Liu, and Q. Wang, "Joint source coding rate allocation and flow scheduling for data aggregation in collaborative sensing networks," *Comput. Netw.*, vol. 175, Jul. 2020, Art. no. 107269.
[25] H. Cheng, C. Li, H. Xiong, and P. Frossard, "Optimal decentralized coded caching for heterogeneous files," in *Proc. 25th Eur. Signal Process. Conf. (EUSIPCO)*, Aug. 2017, pp. 2531–2535.
[26] P. Wang, C. Li, and J. Zheng, "Distributed data aggregation using clustered slepian–wolf coding in wireless sensor networks," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2007, pp. 3616–3622.
[27] P. Wang, C. Li, and J. Zheng, "Combined data aggregation and encryption using clustered slepian–wolf coding for wireless sensor networks," in *Proc. IEEE Global Telecommun. Conf.*, Nov. 2007, pp. 920–925.
[28] 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. 59093–59107, 2021.
[29] 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. 117261–117272, 2022.
[30] 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, 2023.
[31] M. Dorigo and T. Stützle, *The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances*. Boston, MA, USA: Springer, 2003, pp. 250–285.
[32] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, Apr. 1997.

BENJAMIN ROSEN received the B.Eng.Sc. degree in digital arts and the B.Sc.Eng. degree in information engineering from the University of the Witwatersrand, Johannesburg, South Africa, in 2017 and 2019, respectively, where he is currently pursuing the Ph.D. degree.

His research interests include information theory, the IoT, wireless sensor networks, and source coding.

ADNAN M. ABU-MAHFOUZ (Senior Member, IEEE) received the M.Eng. and Ph.D. degrees in computer engineering from the University of Pretoria, Pretoria, South Africa, in 2005 and 2011, respectively.

He is currently a Chief Researcher and the Centre Manager of the Emerging Digital Technologies for 4IR (EDT4IR) Research Centre, Council for Scientific and Industrial Research; an Extraordinary Professor with the University of Pretoria; and a Professor Extraordinaire with Tshwane University of Technology, Pretoria. His research interests include wireless sensor and actuator networks, low power wide area networks, software-defined wireless sensor networks, cognitive radio, network security, network management, and sensor/actuator node development. He is a member of many IEEE technical communities. He is the Section Editor-in-Chief of the *Journal of Sensor and Actuator Networks*, an Associate Editor of IEEE INTERNET OF THINGS JOURNAL, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE TRANSACTIONS ON CYBERNETICS, and IEEE ACCESS.



LING CHENG (Senior Member, IEEE) received the B.Eng. degree (cum laude) in electronics and information from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 1995, the M.Ing. degree (cum laude) in electrical and electronics, in 2005, and the D.Ing. degree in electrical and electronics from the University of Johannesburg (UJ), Johannesburg, South Africa, in 2011.

In 2010, he joined the University of the Witwatersrand, Johannesburg, where he was promoted to a Full Professor, in 2019. He has been a Visiting Professor with five universities and the Principal Advisor for more than 50 full research postgraduate students (11 Ph.D.s graduated). He has authored or co-authored more than 150 research papers in journals and conference proceedings. His research interests include telecommunications and artificial intelligence.

Dr. Cheng was a recipient of the Chancellor's Medals, in 2005 and 2019, and the National Research Foundation ratings, in 2014 and 2020. The IEEE ISPLC 2015 Best Student Paper Award was made to his Ph.D. student in Austin. He is the Vice-Chair of the IEEE South African Information Theory Chapter. He is an associate editor of three journal articles.