Article

Computational Techniques for Investigating Information Theoretic Limits of Information Systems

Chao Tian[1], James Plank[2], Brent Hurst[3], and Ruida Zhou[4]

[1] Texas A&M University; chao.tian@tamu.edu
[2] University of Tennessee Knoxville; jplank@utk.edu
[3] University of Tennessee Knoxville; tmn678@vols.utk.edu
[4] Texas A&M University; ruida@tamu.edu

Version September 16, 2021 submitted to Information

Abstract: Computer-aided methods, based on the entropic linear program framework, have been shown to be effective in assisting the study of information theoretic fundamental limits of information systems. One key element that significantly impacts their computation efficiency and applicability is the reduction of variables, based on problem-specific symmetry and dependence relations. In this work, we propose using the disjoint-set data structure to algorithmically identify the reduction mapping, instead of relying on exhaustive enumeration in the equivalence classification. Based on this reduced linear program, we consider four techniques to investigate the fundamental limits of information systems: 1) computing an outer bound for a given linear combination of information measures and providing the values of information measures at the optimal solution; 2) efficiently computing a polytope tradeoff outer bound between two information quantities; 3) producing a proof (as a weighted sum of known information inequalities) for a computed outer bound; and 4) providing the range for information quantities between which the optimal value does not change, i.e., sensitivity analysis. A toolbox, with an efficient JSON format input frontend, and either Gurobi or Cplex as the linear program solving engine, is implemented and open-sourced.

Keywords: Capacity, converse bounds, computational methods.

---

1. Introduction

One of the most distinguishing features of information theory is its ability to provide fundamental limits to various communication and computation systems, which may be extremely difficult, if not impossible, to establish otherwise. There are a set of well-known information inequalities, such as the non-negativity of mutual information and conditional mutual information, which are guaranteed to hold simply due to the basic mathematical properties of the information measures such as entropy and conditional mutual information. Fundamental limits of various information systems can be obtained by combining these inequalities strategically. The universality of the information measures implies that fundamental limits of diverse information systems can be derived in a general manner.

Conventionally, the proofs for such fundamental limits are hand-crafted and written as a chain of inequalities, where each individual step is one of the afore-mentioned known information inequalities, or certain equality and inequalities implied by the specific problem settings. As information systems become more and more complex, such manual efforts have become increasingly unwieldy, and computer-aided approaches naturally emerge as possible alternatives. A computer-aided approach can be particularly attractive and productive during the stage of initial problem exploration and when the complexity of the system prevents an effective bound to be constructed manually.

The entropic linear programming (LP) framework [1] was the first major step toward this direction, however since the resultant LPs are usually very large, a direct adoption limits its applicability to simple problem settings, typically with no greater than ten random variables. In several recent works [2–7] which were led by the first author of the current work, it was shown that reductions based on problem-specific symmetry and dependence relations can be used to make the problems more manageable. In this work, we further develop this research direction. First, we adopt an efficient data structure, namely disjoint-set [8], to improve the efficiency of the afore-mentioned reduction. Then we consider and develop four techniques to investigate the fundamental limits of information systems: 1) computing a bound for a given linear combination of information measures and providing the value of information measures at the optimal solution; 2) efficiently computing a polytope tradeoff outer bound between two information quantities; 3) producing a proof (as a weighted sum of known information inequalities; and 4) providing the range for information quantities between which the optimal value does not change (sensitivity analysis). To improve the utility of the approach, an efficient JSON input format is provided, and a toolbox using either Cplex [9] or Gurobi [10] as the linear program solving engine, is implemented and open-sourced [11].

2. Literature Review

In a pioneer work, Yeung [1] pointed out and demonstrated how a linear programming (LP) framework can be used to computationally verify whether an information inequality involving Shannon's information measures is true or not, or more precisely, whether it can be proved using a general set of known information inequalities, which has since been known as Shannon-type inequalities. A Matlab implementation based on this connection, called the information theory inequality prover (ITIP) [12], was made available online at the same time. A subsequent effort by another group (XITIP [13]) replaced the Matlab LP solver with a more efficient open source LP solver and also introduced a more user-friendly interface. Later on, a new version of ITIP also adopted a more efficient LP solver to improve the computation efficiency. ITIP and XITIP played important roles in the study of non-Shannon-type inequalities and Markov random fields [14–16].

Despite its considerable impact, ITIP is a generic inequality prover, and utilizing it on any specific coding problem can be a daunting task. It can also fail to provide meaningful results due to the associated computation cost. Instead of using the LP to verify a hypothesized inequality, a more desirable approach is to use a computational approach on the specific problem of interest to directly find the fundamental limits, and moreover, to utilize the inherent problem structure in reducing the computation burden. This was the approach taken on several problems of recent interest, such as distributed storage, coded caching and private information retrieval [2–7], and it was shown to be rather effective.

One key difference in the above-mentioned line of work, compared to several other efforts in the literature, is the following. Since most information theoretic problems of practical relevance or current interests induce a quite large LP instance, considerable effort was given to reducing the number of LP variables and the number of LP constraints algorithmically, before the LP solver is even invoked. Particularly, problem-specific symmetry and dependence have been used explicitly for this purpose, instead of the standard approach of leaving them for the LP solver to eliminate. This approach allows the program to handle larger problems than ITIP can, which has yielded meaningful results on problems of current interest. Moreover, through LP duality, it has been demonstrated in [2] that human-readable proofs can be generated by taking advantage of the dual LP. This approach of generating proofs has been adopted and extended by several other works [17,18].

From more theoretical perspectives, a minimum set of LP constraints under problem-specific dependence was fully characterized in [19], and the problem of counting the number of LP variables and constraints after applying problem specific symmetry relations was considered in [20]. However, these results do not lead to any algorithmic advantage, since the former relies on a set of relationship tests which are algorithmically expensive to complete, and the latter provided a method of counting instead of enumerating these information inequalities.

Li et al. used a similar computational approach to tackle the multilevel diversity coding problem [21] and multi-source network coding problems with simple network topology [22] (see also [23]); however the main focus was to provide an efficient enumeration and classification of the large number of specific small instances (all instances considered require 7 or fewer random variables) where each instance itself poses little computation issue. Beyond computing outer bounds, the problem of computationally generating inner bounds was also explored [24,25].

Recently, Ho et al. [18] revisited the problem of using the LP framework for verifying the validity of information inequalities, and proposed a method to computationally disprove certain information inequalities. Moreover, it was shown that the alternating direction method of multipliers (ADMM) can be used to speed up the LP computation. In a different application of the LP framework [26], Gattegno et al. used it to improve the efficiency of the Fourier-Motzkin elimination procedure often encountered in information theoretic study of multiterminal coding problems. In another generalization of the approach, Gurpinar and Romashchenko used the computational approach in an extended probability space such that information inequalities beyond Shannon-types may become active [27].

3. Information Inequalities and Entropic LP

In this section, we provide the background and a brief review of the entropic linear program framework. Readers are referred to [28–30] for more details.

3.1. Information Inequalities

The most well-known information inequalities are based on the non-negativity of the conditional entropy and mutual information, which are

$$\begin{aligned} &H(X_1|X_2) \geq 0 \\ &I(X_1;X_2|X_3) \geq 0, \end{aligned} \tag{1}$$

where the single random variables X_1, X_2, and X_3 can be replaced by sets of random variables. A very large number of inequalities can be written this way, when the problem involves a total of n random variables $X_1, X_2, \ldots, X_n$. Within the set of all information inequalities in the form shown in (1), many are implied by others. There are also other information inequalities implied by the basic mathematical properties of the information measure but not in these forms or directly implied by them, which are usually referred to as non-Shannon-type inequalities. Non-Shannon-type inequalities are notoriously difficult to enumerate and utilize [31–34]. In practice, almost all bounds for the fundamental limits of information systems have been derived using only Shannon-type inequalities.

3.2. The Entropic LP Formulation

Suppose we express all the relevant quantities in a particular information system (a coding problem) as random variables $(X_1, X_2, \ldots, X_n)$, e.g., X_1 is an information source and X_3 is its encoded version at a given point in the system. In this case, the derivation of a fundamental limit in an information system or a communication system may be understood conceptually as the following optimization problem:

minimize: a weighted sum of certain joint entropies
subject to: (I) generic constraints that any information measures must satisfy
(II) problem specific constraints on the information measures,

where the variables in this optimization problem are all the **information measures** on the random variables $X_1, X_2, \ldots, X_n$ that we can write down in this problem. For example, if $H(X_2, X_3)$ is a certain quantity that we wish to minimize (e.g., as the total amount of the compressed information in the system), then the solution of the optimization problem with $H(X_2, X_3)$ being the objective function will provide the fundamental limit of this quantity (e.g., the lowest amount we can compress the information to).

The first observation is that the variables in the optimization problem may be restricted to all possible joint entropies. In other words, there are $2^n - 1$ variables of the form of $H(X_{\mathcal{A}})$ where $\mathcal{A} \subseteq \{1, 2, \ldots, n\}$. We do not need to include conditional entropy, mutual information, or conditional mutual information, becuase they may be written simply as linear combinations of the joint entropies.

Next let us focus on the two classes of constraints. To obtain a good (hopefully tight) bound, we wish to include all the Shannon-type inequalities as generic constraints in the first group of constraints. However, enumerating all of them is not the best approach, as we have mentioned earlier that there are redundant inequalities that are implied by others. Yeung identified a minimal set of constraints which are called elemental inequalities [1,29]:

$$H(X_i|X_{\mathcal{A}}) \geq 0, \quad i \in \{1, 2, \ldots, n\}, \qquad \mathcal{A} \subseteq \{1, 2, \ldots, n\} \setminus \{i\} \tag{2}$$

$$I(X_i; X_j|X_{\mathcal{A}}) \geq 0, \quad i \neq j,\ i, j \in \{1, 2, \ldots, n\}, \qquad \mathcal{A} \subseteq \{1, 2, \ldots, n\} \setminus \{i, j\}. \tag{3}$$

Note that both (2) and (3) can be written as linear constraints in terms of joint entropies. It is straightforward to see that there are $n + \binom{n}{2} 2^{n-2}$ elemental inequalities. These are the generic constraints that we will use in group (I).

The second group of constraints are the problem specific constraints. These are usually the implication relations required by the system or the specific coding requirements. For example, if X_4 is a coded representation of X_1 and X_2, then this relation can be represented as

$$H(X_4|X_1, X_2) = H(X_1, X_2, X_4) - H(X_1, X_2) = 0, \tag{4}$$

which is a linear constraint. This group of constraints may also include independence and conditional independence relations. For example, if X_1, X_3, X_7 are three mutually independent sources, then this relation can be represented as

$$H(X_1, X_3, X_7) - H(X_1) - H(X_3) - H(X_7) = 0, \tag{5}$$

which is also a linear constraint. In the examples in later sections, we will provide these constraints more specifically.

The two groups of constraints are both linear in terms of the optimization problem variables, i.e., the $2^n - 1$ joint entropies (defined on the n random variables), and thus we have a linear program (LP) at hand.

4. Symmetry and Dependence Relations

In this section, we discuss two relations that can help reduce the complexity of the entropic LP, without which many information system or coding problems of practical interest appear too complex to be solved in the entropic LP formulation. To be more specific, we first introduce two working examples that will be used throughout this paper to illustrate the main idea.

4.1. Two Examples

The two example problems are the regenerating code problem and the coded caching problem:

- The (n, k, d) regenerating code problem [35,36] is depicted in Figure 1. It considers the situation that a message is stored in a distributed manner in n nodes, each having capacity α (Figure 1(a)). Two coding requirements need to be satisfied: 1) the message can be recovered from any k nodes (Figure 1(b)), and 2) any single node can be repaired by downloading β amount of information from any d of the other nodes (Figure 1(c)). The fundamental limit of interest is the optimal tradeoff between the storage cost α and the download cost β. We will use the $(n, k, d) = (4, 3, 3)$

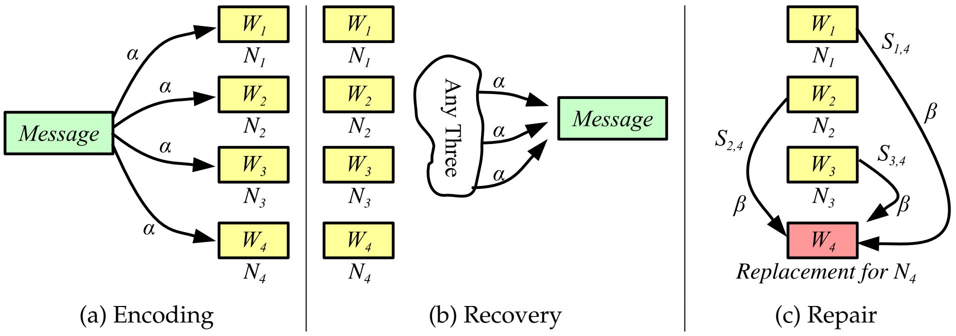

Figure 1. The regenerating code problem with $(n, k, d) = (4, 3, 3)$. .

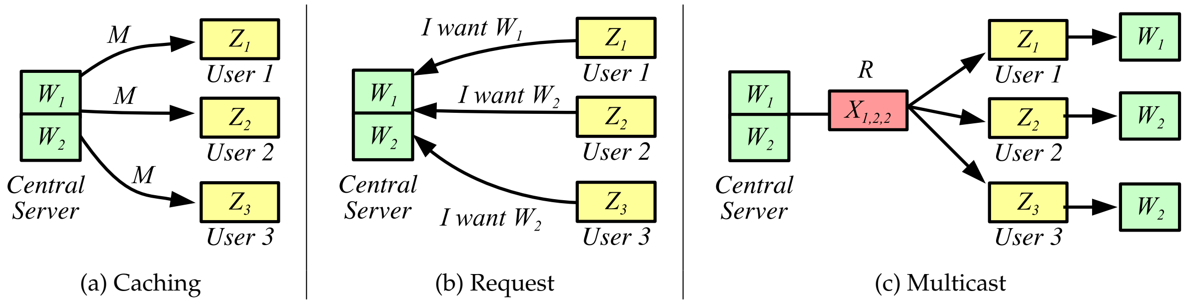

Figure 2. The caching problem with $(N, K) = (2, 3)$.

case as our working example. In this setting, the stored contents are W_1, W_2, W_3, W_4, and the repair message sent from node i to repair j is denoted as $S_{i,j}$. In this case, the set of the random variables in the problem are

$$W_1, W_2, W_3, W_4, S_{1,2}, S_{1,3}, S_{1,4}, S_{2,1}, S_{2,3}, S_{2,4}, S_{3,1}, S_{3,2}, S_{3,4}, S_{4,1}, S_{4,2}, S_{4,3}.$$

Some readers may notice that we do not include a random variable to represent the original message stored in the system. This is because it can be equivalently viewed as the collection of (W_1, W_2, W_3, W_4) and can thus be omitted in this formulation.

- The (N, K) coded caching problem [37] considers the situation that a server, which holds a total N mutually independent files of unit size each, serves a set of K users, each with a local cache of size M. The users can prefetch some content (Figure 2(a)), but when they reveal their requests (Figure 2(b)), the server must calculate and multicast a common message of size R (Figure 2(c)). The requests are not revealed to the server beforehand, and the prefetching must be designed to handle all cases. The fundamental limit of interest is the optimal tradeoff between the cache capacity M and the transmission size R. In this setting, the messages are denoted as $(W_1, W_2, \ldots, W_N)$, the prefetched contents as $(Z_1, Z_2, \ldots, Z_K)$, and the transmission when the users requests $(d_1, d_2, \ldots, d_K)$ is written as $X_{d_1, d_2, \ldots, d_K}$. We will use the case $(N, K) = (2, 3)$ as our second running example in the sequel, and in this case the random variables in the problem are

$$W_1, W_2, Z_1, Z_2, Z_3, X_{1,1,1}, X_{1,1,2}, X_{1,2,1}, X_{1,2,2}, X_{2,1,1}, X_{2,1,2}, X_{2,2,1}, X_{2,2,2}.$$

4.2. The Dependency Relation

The dependency (or implication) relation, e.g., the one given in (4), can be included in the optimization problem in different ways. The first option, which is the simplest, is to include these equality constraints directly as constraints of the LP. There is however another method. Observe that since the two entropy values are equal, we can simply represent them using the same LP variable, instead of generating two different LP variables then insisting that they are of the same value. This

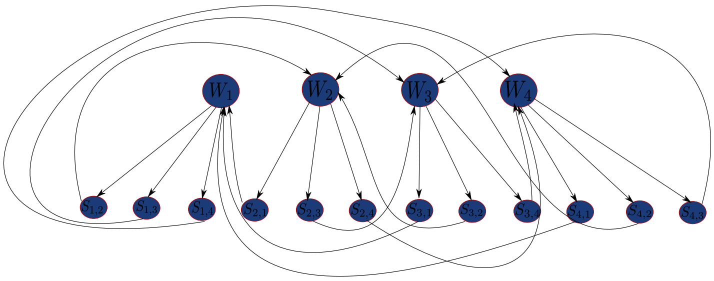

Figure 3. A graph representation for dependence relation in the regenerating code problem with $(n, k, d) = (4, 3, 3)$.

helps reduce the number of LP variables in the problem. In our two working examples, the dependence relations are as follows.

- The regenerating code problem: the relations are the following

$$
\begin{aligned}
&H(S_{1,2}, S_{1,3}, S_{1,4}|W_1) = 0, \quad H(S_{2,1}, S_{2,3}, S_{2,4}|W_2) = 0, \quad H(S_{3,1}, S_{3,2}, S_{3,4}|W_3) = 0,\\
&H(S_{4,1}, S_{4,2}, S_{4,3}|W_4) = 0, \quad H(W_1|S_{2,1}, S_{3,1}, S_{4,1}) = 0, \quad H(W_2|S_{1,2}, S_{3,2}, S_{4,2}) = 0,\\
&H(W_3|S_{1,3}, S_{2,3}, S_{4,3}) = 0, \quad H(W_4|S_{1,4}, S_{2,4}, S_{3,4}) = 0.
\end{aligned} \tag{6}
$$

  The first equality implies that

$$H(S_{1,2}, S_{1,3}, S_{1,4}, W_1) = H(W_1), \tag{7}$$

  and we can alternatively write

$$W_1 \rightarrow \{S_{1,2}, S_{1,3}, S_{1,4}\}. \tag{8}$$

  Other dependence relations can be converted similarly. This dependence structure can also be represented as a graph shown in Fig. 3. In this graph, a given node (random variable) is a function of others random variables with an incoming edge.
- The caching problem: the relations are the following

$$
\begin{aligned}
&H(Z_1, Z_2, Z_3, X_{1,1,1}, X_{1,1,2}, X_{1,2,1}, X_{1,2,2}, X_{2,1,1}, X_{2,1,2}, X_{2,2,1}, X_{2,2,2}|W_1, W_2) = 0,\\
&H(W_1|Z_1, X_{1,1,1}) = 0, \quad H(W_1|Z_2, X_{1,1,1}) = 0, \quad H(W_1|Z_3, X_{1,1,1}) = 0, \quad H(W_1|Z_1, X_{1,1,2}) = 0,\\
&H(W_1|Z_2, X_{1,1,2}) = 0, \quad H(W_2|Z_3, X_{1,1,2}) = 0, \quad H(W_1|Z_1, X_{1,2,1}) = 0, \quad H(W_2|Z_2, X_{1,2,1}) = 0,\\
&H(W_1|Z_3, X_{1,2,1}) = 0, \quad H(W_1|Z_1, X_{1,2,2}) = 0, \quad H(W_2|Z_2, X_{1,2,2}) = 0, \quad H(W_2|Z_3, X_{1,2,2}) = 0,\\
&H(W_2|Z_1, X_{2,1,1}) = 0, \quad H(W_1|Z_2, X_{2,1,1}) = 0, \quad H(W_1|Z_3, X_{2,1,1}) = 0, \quad H(W_2|Z_1, X_{2,1,2}) = 0,\\
&H(W_1|Z_2, X_{2,1,2}) = 0, \quad H(W_2|Z_3, X_{2,1,2}) = 0, \quad H(W_2|Z_1, X_{2,2,1}) = 0, \quad H(W_2|Z_2, X_{2,2,1}) = 0,\\
&H(W_1|Z_3, X_{2,2,1}) = 0, \quad H(W_2|Z_1, X_{2,2,2}) = 0, \quad H(W_2|Z_2, X_{2,2,2}) = 0, \quad H(W_2|Z_3, X_{2,2,2}) = 0.
\end{aligned}
$$

4.3. The Symmetry Relation

In many problems, there are certain symmetry relations present. Such symmetry relations are usually a direct consequence of the structure of the information systems. Often it is **without loss of optimality** to consider only codes with a specific symmetric structure. In our two working examples, the symmetry relations are as follows.

- The regenerating code problem: exchanging the coding functions for different storage nodes. For example, if we simply let node 2 store the content for node 1, and also exchange other coding functions, the result is another code that can fulfill the same task as before this exchange.

Mathematically, we can represent the symmetry relation using permutations of all the random variables, where each row indicates a permutation as follows:

$$\begin{gathered}
W_1, W_2, W_3, W_4, S_{1,2}, S_{1,3}, S_{1,4}, S_{2,1}, S_{2,3}, S_{2,4}, S_{3,1}, S_{3,2}, S_{3,4}, S_{4,1}, S_{4,2}, S_{4,3} \\
W_1, W_2, W_4, W_3, S_{1,2}, S_{1,4}, S_{1,3}, S_{2,1}, S_{2,4}, S_{2,3}, S_{4,1}, S_{4,2}, S_{4,3}, S_{3,1}, S_{3,2}, S_{3,4} \\
W_1, W_3, W_2, W_4, S_{1,3}, S_{1,2}, S_{1,4}, S_{3,1}, S_{3,2}, S_{3,4}, S_{2,1}, S_{2,3}, S_{2,4}, S_{4,1}, S_{4,3}, S_{4,2} \\
W_1, W_4, W_3, W_2, S_{1,4}, S_{1,3}, S_{1,2}, S_{4,1}, S_{4,3}, S_{4,2}, S_{3,1}, S_{3,4}, S_{3,2}, S_{2,1}, S_{2,4}, S_{2,3} \\
W_1, W_3, W_4, W_2, S_{1,3}, S_{1,4}, S_{1,2}, S_{3,1}, S_{3,4}, S_{3,2}, S_{4,1}, S_{4,3}, S_{4,2}, S_{2,1}, S_{2,3}, S_{2,4} \\
W_1, W_4, W_2, W_3, S_{1,4}, S_{1,2}, S_{1,3}, S_{4,1}, S_{4,2}, S_{4,3}, S_{2,1}, S_{2,4}, S_{2,3}, S_{3,1}, S_{3,4}, S_{3,2} \\
W_2, W_1, W_3, W_4, S_{2,1}, S_{2,3}, S_{2,4}, S_{1,2}, S_{1,3}, S_{1,4}, S_{3,2}, S_{3,1}, S_{3,4}, S_{4,2}, S_{4,1}, S_{4,3} \\
W_2, W_4, W_3, W_1, S_{2,4}, S_{2,3}, S_{2,1}, S_{4,2}, S_{4,3}, S_{4,1}, S_{3,2}, S_{3,4}, S_{3,1}, S_{1,2}, S_{1,4}, S_{1,3} \\
W_2, W_1, W_4, W_3, S_{2,1}, S_{2,4}, S_{2,3}, S_{1,2}, S_{1,4}, S_{1,3}, S_{4,2}, S_{4,1}, S_{4,3}, S_{3,2}, S_{3,1}, S_{3,4} \\
W_2, W_4, W_1, W_3, S_{2,4}, S_{2,1}, S_{2,3}, S_{4,2}, S_{4,1}, S_{4,3}, S_{1,2}, S_{1,4}, S_{1,3}, S_{3,2}, S_{3,4}, S_{3,1} \\
W_2, W_3, W_1, W_4, S_{2,3}, S_{2,1}, S_{2,4}, S_{3,2}, S_{3,1}, S_{3,4}, S_{1,2}, S_{1,3}, S_{1,4}, S_{4,2}, S_{4,3}, S_{4,1} \\
W_2, W_3, W_4, W_1, S_{2,3}, S_{2,4}, S_{2,1}, S_{3,2}, S_{3,4}, S_{3,1}, S_{4,2}, S_{4,3}, S_{4,1}, S_{1,2}, S_{1,3}, S_{1,4} \\
W_3, W_2, W_1, W_4, S_{3,2}, S_{3,1}, S_{3,4}, S_{2,3}, S_{2,1}, S_{2,4}, S_{1,3}, S_{1,2}, S_{1,4}, S_{4,3}, S_{4,2}, S_{4,1} \\
W_3, W_2, W_4, W_1, S_{3,2}, S_{3,4}, S_{3,1}, S_{2,3}, S_{2,4}, S_{2,1}, S_{4,3}, S_{4,2}, S_{4,1}, S_{1,3}, S_{1,2}, S_{1,4} \\
W_3, W_1, W_2, W_4, S_{3,1}, S_{3,2}, S_{3,4}, S_{1,3}, S_{1,2}, S_{1,4}, S_{2,3}, S_{2,1}, S_{2,4}, S_{4,3}, S_{4,1}, S_{4,2} \\
W_3, W_1, W_4, W_2, S_{3,1}, S_{3,4}, S_{3,2}, S_{1,3}, S_{1,4}, S_{1,2}, S_{4,3}, S_{4,1}, S_{4,2}, S_{2,3}, S_{2,1}, S_{2,4} \\
W_3, W_4, W_1, W_2, S_{3,4}, S_{3,1}, S_{3,2}, S_{4,3}, S_{4,1}, S_{4,2}, S_{1,3}, S_{1,4}, S_{1,2}, S_{2,3}, S_{2,4}, S_{2,1} \\
W_3, W_4, W_2, W_1, S_{3,4}, S_{3,2}, S_{3,1}, S_{4,3}, S_{4,2}, S_{4,1}, S_{2,3}, S_{2,4}, S_{2,1}, S_{1,3}, S_{1,4}, S_{1,2} \\
W_4, W_2, W_3, W_1, S_{4,2}, S_{4,3}, S_{4,1}, S_{2,4}, S_{2,3}, S_{2,1}, S_{3,4}, S_{3,2}, S_{3,1}, S_{1,4}, S_{1,2}, S_{1,3} \\
W_4, W_2, W_1, W_3, S_{4,2}, S_{4,1}, S_{4,3}, S_{2,4}, S_{2,1}, S_{2,3}, S_{1,4}, S_{1,2}, S_{1,3}, S_{3,4}, S_{3,2}, S_{3,1} \\
W_4, W_1, W_3, W_2, S_{4,1}, S_{4,3}, S_{4,2}, S_{1,4}, S_{1,3}, S_{1,2}, S_{3,4}, S_{3,1}, S_{3,2}, S_{2,4}, S_{2,1}, S_{2,3} \\
W_4, W_1, W_2, W_3, S_{4,1}, S_{4,2}, S_{4,3}, S_{1,4}, S_{1,2}, S_{1,3}, S_{2,4}, S_{2,1}, S_{2,3}, S_{3,4}, S_{3,1}, S_{3,2} \\
W_4, W_3, W_1, W_2, S_{4,3}, S_{4,1}, S_{4,2}, S_{3,4}, S_{3,1}, S_{3,2}, S_{1,4}, S_{1,3}, S_{1,2}, S_{2,4}, S_{2,3}, S_{2,1} \\
W_4, W_3, W_2, W_1, S_{4,3}, S_{4,2}, S_{4,1}, S_{3,4}, S_{3,2}, S_{3,1}, S_{2,4}, S_{2,3}, S_{2,1}, S_{1,4}, S_{1,3}, S_{1,2}
\end{gathered}$$

Note that when we permute the storage contents (W_1, W_2, W_3, W_4) the corresponding repair information needs to be permuted accordingly. The 24 permutations clearly form a permutation group. With this representation, we can take any subset of the columns, and the collections of the random variables in each row in these columns will have the same entropy in the corresponding symmetric code. For example, if we take columns $2, 9, 15$, then we have that

$$H(W_1, S_{2,1}, S_{4,1}) = H(W_1, S_{2,1}, S_{3,1}) = H(W_1, S_{3,1}, S_{4,1}) = \ldots$$

There are a total of $2^{16} - 1$ subset of columns, and they each will induce a set of equality relations. For a more rigorous discussion of this symmetry relation, the readers can refer to [2,20].

- Similarly, in the caching problem, there are two types of symmetry relations. The first is to exchange the coding functions for each users, and the second is to exchange the operation on different files. Intuitively, the first one is due to a permutation of the users, and the second due to the permutation of the files. As a consequence, we have the following permutations that form a group:

$$\begin{gathered}
W_1, W_2, Z_1, Z_2, Z_3, X_{1,1,1}, X_{1,1,2}, X_{1,2,1}, X_{1,2,2}, X_{2,1,1}, X_{2,1,2}, X_{2,2,1}, X_{2,2,2} \\
W_2, W_1, Z_1, Z_2, Z_3, X_{2,2,2}, X_{2,2,1}, X_{2,1,2}, X_{2,1,1}, X_{1,2,2}, X_{1,2,1}, X_{1,1,2}, X_{1,1,1}
\end{gathered}$$

$$
\begin{aligned}
&W_1, W_2, Z_2, Z_1, Z_3, X_{1,1,1}, X_{1,1,2}, X_{2,1,1}, X_{2,1,2}, X_{1,2,1}, X_{1,2,2}, X_{2,2,1}, X_{2,2,2}\\
&W_2, W_1, Z_2, Z_1, Z_3, X_{2,2,2}, X_{2,2,1}, X_{1,2,2}, X_{1,2,1}, X_{2,1,2}, X_{2,1,1}, X_{1,1,2}, X_{1,1,1}\\
&W_1, W_2, Z_3, Z_2, Z_1, X_{1,1,1}, X_{2,1,1}, X_{1,2,1}, X_{2,2,1}, X_{1,1,2}, X_{2,1,2}, X_{1,2,2}, X_{2,2,2}\\
&W_2, W_1, Z_3, Z_2, Z_1, X_{2,2,2}, X_{1,2,2}, X_{2,1,2}, X_{1,1,2}, X_{2,2,1}, X_{1,2,1}, X_{2,1,1}, X_{1,1,1}\\
&W_1, W_2, Z_1, Z_3, Z_2, X_{1,1,1}, X_{1,2,1}, X_{1,1,2}, X_{1,2,2}, X_{2,1,1}, X_{2,2,1}, X_{2,1,2}, X_{2,2,2}\\
&W_2, W_1, Z_1, Z_3, Z_2, X_{2,2,2}, X_{2,1,2}, X_{2,2,1}, X_{2,1,1}, X_{1,2,2}, X_{1,1,2}, X_{1,2,1}, X_{1,1,1}\\
&W_1, W_2, Z_2, Z_3, Z_1, X_{1,1,1}, X_{2,1,1}, X_{1,1,2}, X_{2,1,2}, X_{1,2,1}, X_{2,2,1}, X_{1,2,2}, X_{2,2,2}\\
&W_2, W_1, Z_2, Z_3, Z_1, X_{2,2,2}, X_{1,2,2}, X_{2,2,1}, X_{1,2,1}, X_{2,1,2}, X_{1,1,2}, X_{2,1,1}, X_{1,1,1}\\
&W_1, W_2, Z_3, Z_1, Z_2, X_{1,1,1}, X_{1,2,1}, X_{2,1,1}, X_{2,2,1}, X_{1,1,2}, X_{1,2,2}, X_{2,1,2}, X_{2,2,2}\\
&W_2, W_1, Z_3, Z_1, Z_2, X_{2,2,2}, X_{2,1,2}, X_{1,2,2}, X_{1,1,2}, X_{2,2,1}, X_{2,1,1}, X_{1,2,1}, X_{1,1,1}
\end{aligned}
$$

For a more detailed discussion on this symmetry relation, the readers can refer to [4,20].

Two remarks are now in order:

- For the purpose of deriving outer bounds, it is valid to ignore the symmetry relation altogether, or consider only part of the symmetry relation, as long as the remaining permutations still form a group. For example, in the caching problem if we only consider the symmetry induced by exchanging the two messages, then we have the first 2 rows instead of the full 12 rows of permutations. Omitting some permutations means less reduction in the LP scale, but does not invalidate the computed bounds.
- Admittedly, representing the symmetry relation using the above permutation representation is not the most concise approach, and there exists mathematically precise and concise language to specify such structure. We choose this permutation approach because of its simplicity and universality, and perhaps more importantly, due to its suitability for software implementation.

5. Reducing the Problem Algorithmically via the Disjoint-Set Data Structure

In this section, we first introduce the equivalence relation and classification of joint entropies, and then introduce the disjoint-set data structure to identify the classification in an algorithmic manner.

5.1. Equivalence Relation and Reduction

The two reductions mentioned in the previous section essentially provide an equivalent relation and a classification of the joint entropies of subsets of the random variables. To see this, let us consider the regenerating code problem. Due to the dependence structure, the following entropies are equal:

$$
\begin{aligned}
&H(W_1, S_{2,1}) = H(W_1, S_{1,2}, S_{2,1}) = H(W_1, S_{1,3}, S_{2,1}) = H(W_1, S_{1,4}, S_{2,1}) = H(W_1, S_{1,2}, S_{1,3}, S_{2,1})\\
&= H(W_1, S_{1,2}, S_{1,4}, S_{2,1}) = H(W_1, S_{1,3}, S_{1,4}, S_{2,1}) = H(W_1, S_{1,2}, S_{1,3}, S_{1,4}, S_{2,1}),
\end{aligned}
\tag{9}
$$

and the following subsets are thus equivalent in this setting:

$$
\begin{aligned}
&\{W_1, S_{2,1}\} \equiv \{W_1, S_{1,2}, S_{2,1}\} \equiv \{W_1, S_{1,3}, S_{2,1}\} \equiv \{W_1, S_{1,4}, S_{2,1}\} \equiv \{W_1, S_{1,2}, S_{1,3}, S_{2,1}\}\\
&\equiv \{W_1, S_{1,2}, S_{1,4}, S_{2,1}\} \equiv \{W_1, S_{1,3}, S_{1,4}, S_{2,1}\} \equiv \{W_1, S_{1,2}, S_{1,3}, S_{1,4}, S_{2,1}\}.
\end{aligned}
\tag{10}
$$

Orthogonal to the dependence relation, the symmetry provides further opportunity to build equivalence. For example, for the first item $H(W_1, S_{2,1})$ above, we have by symmetry

$$
\begin{aligned}
&H(W_1, S_{2,1}) = H(W_1, S_{3,1}) = H(W_1, S_{4,1}) = H(W_2, S_{1,2}) = H(W_2, S_{4,2}) = H(W_2, S_{3,2})\\
&= H(W_3, S_{1,3}) = H(W_3, S_{2,3}) = H(W_3, S_{4,3}) = H(W_4, S_{1,4}) = H(W_4, S_{2,4}) = H(W_4, S_{3,4}),
\end{aligned}
\tag{11}
$$

and for the second term $H(W_1, S_{1,2}, S_{2,1})$ we have

$$\begin{aligned} & H(W_1, S_{1,2}, S_{2,1}) = H(W_1, S_{1,3}, S_{3,1}) = H(W_1, S_{1,4}, S_{4,1}) \\ & = H(W_2, S_{2,3}, S_{3,2}) = H(W_2, S_{1,2}, S_{2,1}) = H(W_2, S_{2,4}, S_{4,2}) \\ & = H(W_3, S_{1,3}, S_{3,1}) = H(W_3, S_{2,3}, S_{3,2}) = H(W_3, S_{3,4}, S_{4,3}) \\ & = H(W_4, S_{1,4}, S_{4,1}) = H(W_4, S_{2,4}, S_{4,2}) = H(W_4, S_{3,4}, S_{4,3}). \end{aligned} \tag{12}$$

Such symmetry-induced equivalence relation holds similarly for every item in (9). All joint entropies such related have the same value.

Mathematically, the dependence and the symmetry jointly induce an equivalence relation, and we wish to identify the classification based on this equivalence relation. The key to efficiently form the reduced LP is to identify the mapping from any subset of random variables to an index of the equivalence class it belongs to, i.e.,

$$f : 2^{\{1,2,\ldots,n\}} \to \{1, 2, \ldots, N^*\}, \tag{13}$$

where N^* is the total number of equivalence classes such induced. In terms of software implementation, the mapping f assigns any subset of the n random variables in the problem to an index, which also serves as the index of the variable in the linear program. More precisely, this mapping provides the fundamental reduction mechanism in the LP formulation, where an elemental constraint of the form

$$H(X_i|X_{\mathcal{A}}) \geq 0 \tag{14}$$

becomes the inequality in the resultant LP

$$Y_{f(\{i\}\cup\mathcal{A})} - Y_{f(\mathcal{A})} \geq 0, \tag{15}$$

where the Y's are the variables in the LP; similarly, the elemental constraint

$$I(X_i; X_j|X_{\mathcal{A}}) \geq 0 \tag{16}$$

becomes

$$Y_{f(\{i\}\cup\mathcal{A})} + Y_{f(\{j\}\cup\mathcal{A})} - Y_{f(\mathcal{A})} - Y_{f(\{i,j\}\cup\mathcal{A})} \geq 0. \tag{17}$$

5.2. Difficulty in Identifying the Reduction

Following the discussion above, each given subset $\mathcal{A} \subseteq \{1, 2, \ldots, n\}$ belongs to an equivalent class of subsets, and an arbitrary element in the equivalent class can be designated (and fixed) as the leader of this equivalent class. To efficiently complete the classification task, we need to be able to find for each given subset $\mathcal{A}$ the leader of the equivalent class this subset belongs to. In the example given above, this step is reasonably straightforward. Complications arise when multiple reduction steps are required. To see this, let us consider the set $\{S_{1,2}, S_{2,3}, S_{4,3}, S_{2,1}, S_{4,1}\}$. By the dependence relation $\{S_{1,3}, S_{2,3}, S_{4,3}\} \to W_3$, we know

$$H(S_{1,3}, S_{2,3}, S_{4,3}, S_{2,1}, S_{4,1}) = H(W_3, S_{1,3}, S_{2,3}, S_{4,3}, S_{2,1}, S_{4,1}). \tag{18}$$

However, we also have

$$H(W_3, S_{1,3}, S_{2,3}, S_{4,3}, S_{2,1}, S_{4,1}) = H(W_3, S_{1,3}, S_{2,3}, S_{4,3}, S_{2,1}, S_{3,1}, S_{4,1}, S_{3,2}, S_{3,4}), \tag{19}$$

because of the dependence relation $W_3 \to \{S_{3,1}, S_{3,2}, S_{3,4}\}$, from which we can further derive

$$H(W_3, S_{1,3}, S_{2,3}, S_{4,3}, S_{2,1}, S_{3,1}, S_{4,1}, S_{3,2}, S_{3,4}) = H(W_1, W_3, S_{1,3}, S_{2,3}, S_{4,3}, S_{2,1}, S_{3,1}, S_{4,1}, S_{3,2}, S_{3,4}), \quad (20)$$

due to the dependence relation $\{S_{2,1}, S_{3,1}, S_{4,1}\} \to W_1$. In this process, we have applied three different dependence relations in the particular order. In a computer program, this implies that we need to iterate over all the dependence relations in the problem to apply the appropriate one, and then repeat the process until no further dependence relation can be applied. To make things worse, the symmetry relation would need to be taken into account: for example, we will also need to consider how to recognize one subset to be a permuted version of another subset, and whether to do so before or after applying the dependence relation. A naive implementation to find the mapping function f will be highly inefficient.

5.3. Disjoint-Set Data Structure and Algorithmic Reduction

The afore-mentioned difficulty can be resolved using a suitable data structure, namely disjoint-set [8]. A disjoint-set data structure is also called a union-find structure, and as its name suggests, it stores a collection of disjoint sets. The most well known method to accomplish this task is through a disjoint-set forest [8], which can perform the union operation in constant time, and the find operation (find for an element the index, or the leading element, of the set that it belongs to) in near constant amortized time.

Roughly speaking, the disjoint-set forest in our setting starts with each subset of random variables $\mathcal{A} \subseteq \{1, 2, \ldots, n\}$ viewed as its own disjoint set and assigned an index; clearly we will have a total $2^n - 1$ singleton sets at initialization. We iterate through each symmetry permutation and dependence relation as follows:

- Symmetry step: For each singleton set (which corresponds to a subset $\mathcal{A} \subseteq \{1, 2, \ldots, n\}$) in the disjoint-set structure, consider each permutation in the symmetry relation: if the permutation maps $\mathcal{A}$ into another element (which corresponds to another subset of random variables $\mathcal{A}' \subseteq \{1, 2, \ldots, n\}$) not already in the same set in the disjoint-set structure, then we combine the two sets by forming their union.
- Dependence step: For each existing set in the disjoint-set structure, consider each dependence relation: if the set leader (which corresponds to a subset $\mathcal{A} \subseteq \{1, 2, \ldots, n\}$) is equivalent to another element due to the given dependence (which corresponds to another subset of random variables $\mathcal{A}' \subseteq \{1, 2, \ldots, n\}$) not already in the same set, then we combine the two sets by forming their union.

The key for the efficiency of this data structure is that the union operation is done through pointers, instead of physical memory copy. Moreover, inherent in the data structure is a tree representation of each set, and thus finding the leader index is equivalent to finding the tree root, which is much more efficient than a linear search. The data structure is maintained dynamically during union and find operations, and the height of a tree will be reduced (compressed) when a find operation is performed or when the tree becomes too high.

Clearly, due to the usage of this data structure, the dependence relation does not need to be exhaustively listed, because the permuted version of the dependence relation is accounted for automatically. For example, in the regenerating code problem, including only two dependence relations will suffice, when used jointly with the symmetry relations:

$$W_1 \to \{S_{1,2}, S_{1,3}, S_{1,4}\}, \quad \{S_{2,1}, S_{3,1}, S_{4,1}\} \to W_1. \quad (21)$$

This replaces the 8 dependence relations given in (6).

In the context of our setting, after the disjoint-set forest is found after both the symmetry step and the dependence step, another enumeration step is performed to generate the mapping function $f(\cdot)$,

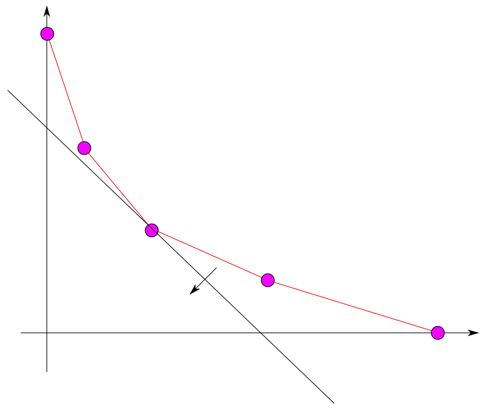

Figure 4. A fixed direction bounding plane and the tradeoff region computation.

which can be done in time 2^n. In practice, we observe this data structure is able to provide considerable speedup (sometimes up to 50 fold), though the precise speedup factor depends on the problem-specific dependence and symmetry relations case by case.

6. Four Investigative Techniques

In this section, we introduce four investigative techniques to study fundamental limits of information systems. With the efficient reduction discussed above, these methods are rather powerful tools in such information theoretic studies.

6.1. Bounding Plane Optimization and Queries

In this case, the objective function is fixed, and the optimal solution gives an outer bound of a specific linear combination of several information measures or relevant quantities. Fig. 4 illustrates the method, where we wish to find a lower bound of the given direction for the given optimal tradeoff shown in red. Let us again consider the two working examples.

- If the simple sum of the storage cost α and repair cost β, i.e., $\alpha + \beta$, needs to be lower-bounded in the regenerating code problem, we can let the objective function be given as

$$H(W_1) + H(S_{1,2})$$

  and then minimize it. The optimal value will be a lower bound, which in this case turns out to be $5/8$. Note that by taking advantage of the symmetry, the objective function set up above indeed specifies the sum rate of any storage and repair transmission.
- If we wish to lower bound the simple sum of memory and rate in the coded caching problem, the situation is somewhat sutble. Note that the rate R is a lower bound on the entropy $H(X_{1,1,1})$ and $H(1,2,2)$; however, the symmetry relation does not imply that $H(X_{1,1,1}) = H(X_{1,2,2})$. For this case, we can introduce an additional LP variable R, and add the constraints that

$$H(X_{1,1,1}) \leq R, \quad H(X_{1,2,2}) \leq R.$$

  We then set the objective function to be

$$H(Z_1) + R,$$

  from which the minimum value is a lower bound on the simple sum of memory and rate in this setting.

In addition to simply computing the supporting hyperplane, it is important to extract useful information from the optimal solution. Particularly, we may wish to probe for the values of certain information measures in the optimal solution. For example, in the case above for coded caching, we may be interested in the value of $I(Z_1; W_1)$, which essentially reveals the amount of information regarding W_1 that is stored in Z_1 in an uncoded form.

6.2. Tradeoff and Convex Hull Computation

In many cases, instead of bounding a fixed rate combination, we are interested in the tradeoff of several quantities, most frequently the optimal tradeoff between two quantities; see Fig. 4 again for an illustration. The two working examples both belong to this case.

Since the constrained set in the LP is a polytope, the resulting outer bound to the optimal tradeoff will be a piece-wise linear bound. A naive strategy is to trace the boundary by sampling points on a sufficiently dense grid. However, this approach is time consuming and not accurate. Instead the calculation of this piece-wise linear outer bound is equivalent to computing the projection of a convex polytope, for which Lassez's algorithm is in fact a method to complete the task efficiently. We implemented Lassez's algorithm for the projection on to two-dimensional space in this toolbox. A more detailed description of this algorithm can be found in [38], and the specialization used in the program can be found in [4].

6.3. Duality and Computer-generated Proof

After identifying a valid outer bound, we sometimes wish to find a proof for this bound. In fact, even if the bound is not optimal, or it is a only hypothesized bound, we may still attempt to prove it. For example, in the regenerating code problem, we have

$$H(W_1) + H(S_{1,2}) \geq \frac{5}{8}. \tag{22}$$

How can we prove this inequality? It is clear from the LP duality that this inequality is a weighted sum of the individual constraints in the LP. Thus as long as we find one such weighted sum, we can then write down a chain of inequalities directly by combining these inequalities one by one; for a more detailed discussion, see [2,4,17,18].

6.4. Sensitivity Analysis

At the computed optimal value, we can probe for the range of certain information measures such that forcing them to be in these ranges does not change the value of the optimal solution. Consider the quantity $I(Z_1; W_1)$ in the caching problem. It may be possible for it to take values between $[0.2, 0.4]$ without changing the optimal value of the original optimization problem. On the other hand, if it can only take value 0.2, then this suggests if a code to achieve this optimal value indeed exists, it must have this amount of uncoded information regarding W_1 stored in Z_1. This information can be valuable in reverse-engineering optimal codes; see [4] for discussion of such usage.

7. JSON Problem Descriptions

In the implemented toolbox, the program can read a problem description file (a plain text file), and the desired computed bounds or proof will be produced without further user intervention. In our work, significant effort has been invested in designing an efficient input format, and after a few iterations, a JSON based format was selected which considerably improves the usability and extendibility of the toolbox. In this section, we provide an example problem description, from which the syntax is mostly self-evident. More details can be found in the documentation accompanying the software [11]. The input problem description files must include the characters `PD` (which stand for "problem description"), followed by a JSON detailing the problem description.

7.1. Keys in `PD` JSON

The program solves a minimization problem, i.e., to find a lower bound for certain information quantity. There are a total of 12 JSON keys allowed in the problem description:

`RV`, `AL`, `O`, `D`, `I`, `S`, `BC`, `BP`, `QU`, `SE`, `CMD`, and `OPT`.

These stand for random variables, additional LP variables, objective function, dependence, independence, bound-constant, bound-to-prove, query, sensitivity, command, and options, respectively. For the precise syntax, the readers are referred to the toolbox user manual. We next provide a simple example to illustrate the usage of this toolbox, from which these keywords are self-evident.

7.2. An Example Problem Description File

Below is a sample PD file for the regenerating code problem we previously discussed.

```
# problem description file for (4,3,3) regenerating codes
PD
{
        "OPT": ["CS"] ,
        "RV" : ["W1","W2","W3","W4","S12","S13","S14","S21","S23","S24","S31","S32","S34","S41","S42","S43"] ,
        "AL" : ["A","B"] ,
        "O"  : "A+B" ,
        "D"  : [
                {"dependent" : ["S12","S13","S14"] , "given" : ["W1"]} ,
                {"dependent" : ["W1"] , "given" : ["S21","S31","S41"]}
                ],
        "BC" : [
                "H(W1)-A<=0" ,
                "H(S12)-B<=0" ,
                "H(W1,W2,W3)>=1"
                ],
        "SE": ["A", "B", "2I(S12;S21|S32)+H(S21|S31)+A"],
        "QU": ["A", "B", "2H(S12|S13)","-2I(S12;S21|S32)"],
        "S"  : [
                ["W1","W2","W3","W4","S12","S13","S14","S21","S23","S24","S31","S32","S34","S41","S42","S43"] ,
                ["W1","W2","W4","W3","S12","S14","S13","S21","S24","S23","S41","S42","S43","S31","S32","S34"] ,
                ["W1","W3","W2","W4","S13","S12","S14","S31","S32","S34","S21","S23","S24","S41","S43","S42"] ,
                ["W1","W4","W3","W2","S14","S13","S12","S41","S43","S42","S31","S34","S32","S21","S24","S23"] ,
                ["W1","W3","W4","W2","S13","S14","S12","S31","S34","S32","S41","S43","S42","S21","S23","S24"] ,
                ["W1","W4","W2","W3","S14","S12","S13","S41","S42","S43","S21","S24","S23","S31","S34","S32"] ,
                ["W2","W1","W3","W4","S21","S23","S24","S12","S13","S14","S32","S31","S34","S42","S41","S43"] ,
                ["W2","W4","W3","W1","S24","S23","S21","S42","S43","S41","S32","S34","S31","S12","S14","S13"] ,
                ["W2","W1","W4","W3","S21","S24","S23","S12","S14","S13","S42","S41","S43","S32","S31","S34"] ,
                ["W2","W4","W1","W3","S24","S21","S23","S42","S41","S43","S12","S14","S13","S32","S34","S31"] ,
                ["W2","W3","W1","W4","S23","S21","S24","S32","S31","S34","S12","S13","S14","S42","S43","S41"] ,
                ["W2","W3","W4","W1","S23","S24","S21","S32","S34","S31","S42","S43","S41","S12","S13","S14"] ,
                ["W3","W2","W1","W4","S32","S31","S34","S23","S21","S24","S13","S12","S14","S43","S42","S41"] ,
                ["W3","W2","W4","W1","S32","S34","S31","S23","S24","S21","S43","S42","S41","S13","S12","S14"] ,
                ["W3","W1","W2","W4","S31","S32","S34","S13","S12","S14","S23","S21","S24","S43","S41","S42"] ,
                ["W3","W1","W4","W2","S31","S34","S32","S13","S14","S12","S43","S41","S42","S23","S21","S24"] ,
                ["W3","W4","W1","W2","S34","S31","S32","S43","S41","S42","S13","S14","S12","S23","S24","S21"] ,
                ["W3","W4","W2","W1","S34","S32","S31","S43","S42","S41","S23","S24","S21","S13","S14","S12"] ,
                ["W4","W2","W3","W1","S42","S43","S41","S24","S23","S21","S34","S32","S31","S14","S12","S13"] ,
                ["W4","W2","W1","W3","S42","S41","S43","S24","S21","S23","S14","S12","S13","S34","S32","S31"] ,
                ["W4","W1","W3","W2","S41","S43","S42","S14","S13","S12","S34","S31","S32","S24","S21","S23"] ,
                ["W4","W1","W2","W3","S41","S42","S43","S14","S12","S13","S24","S21","S23","S34","S31","S32"] ,
                ["W4","W3","W1","W2","S43","S41","S42","S34","S31","S32","S14","S13","S12","S24","S23","S21"] ,
                ["W4","W3","W2","W1","S43","S42","S41","S34","S32","S31","S24","S23","S21","S14","S13","S12"]
                ],
        "BP" : [ "4A+6B>=3" ]
}
```

In this setting, we introduce two additional LP variables `A` and `B` to represent the storage rate α and the repair rate β, respectively. The objective function chosen is the sum rate `A+B`, i.e. $\alpha + \beta$. The option `CS` means that we are running a validity check on the symmetry relation. The sensitivity analysis is performed on `A`, `B`, and a third expression `2I(S12;S21|S32)+H(S21|S31)+A`. We can also query the four quantities given in the `QU` section. A bound that we may attempt to prove is `4A + 6B >= 3`. By choosing the right computation functionality in the toolbox, we can let the program perform direct bounding, convex hull computation, generating a proof, or sensitivity analysis using this problem description file.

7.3. Example Computation Result

The result to bound the sum rate $\alpha + \beta$ is given as follows:

```
Symmetries have been successfully checked.
Total number of elements before reduction: 65536
Total number of elements after reduction: 179
Total number of constraints given to Cplex: 40862
******************************************************************
Optimal value for A + B = 0.625000.
Queried values:
A                          = 0.37500
B                          = 0.25000
2H(S12|S13)                = 0.25000
-2I(S12;S21|S32)          = -0.25000
******************************************************************
```

The first part of the output is various information about the problem and the corresponding check and verification result. The last star-separated segment means that the program found a bound

$$\alpha + \beta \geq 0.625.$$

The queried quantities are also shown in this part, and it can be seen (α, β) in the LP optimal solution are $(0.375, 0.25)$, together with the values of two other information measures.

The toolbox can also identify the tradeoff between α and β, for which the output is as follows.

```
Total number of elements before reduction: 65536
Total number of elements after reduction: 179
Total number of constraints given to Cplex: 40862
New point (0.333333, 0.333333).
New point (0.500000, 0.166667).
New point (0.375000, 0.250000).

List of found points on the hull:
(0.333333, 0.333333).
(0.375000, 0.250000).
(0.500000, 0.166667).
End of list of found points.
```

Here the three (α, β) pairs are the corner points of the lower convex hull of the tradeoff.

To prove this inequality, the toolbox gives

```
Total number of elements before reduction: 65536
Total number of elements after reduction: 179
Total number of constraints given to Cplex: 40862
******************************************************************
LP dual value 29.000000
Proved 2-th inequality: 4A + 6B >= 3.
001-th inequality: weight = 1.000000     H(W1,W3,S21,S41)    -H(W1,W3,S21,S23,S41)     H(W1,S24,S31,S41)    -H(W1,S24,S41)>=0
002-th inequality: weight = 3.000000    -H(W1,W3,S21,S41)    2.0H(W1,S24)    -H(W2)>=0
003-th inequality: weight = 7.000000    -H(W1,S24)     H(S13)     H(W2)>=0
004-th inequality: weight = 1.000000    -H(S13)     H(W1,S31)     H(S13,S24)    -H(W1,S23,S41)>=0
005-th inequality: weight = 1.000000    -H(W1,W2,W3,W4)    -H(W1,S31)     H(W1,W4,S21)     H(W1,S24,S41)>=0
006-th inequality: weight = 1.000000     H(W1,W3,S21,S41)    -H(W1,W4,S21)>=0
007-th inequality: weight = 1.000000    -H(W1,W2,W3,W4)     H(W1,W3,S21,S41)     H(W1,W3,S21,S23,S41)    -H(W1,S24,S31,S41)>=0
008-th inequality: weight = 1.000000    -H(W1,W2,W3,W4)     H(W1,S24)    -H(S13,S24)     H(W1,S23,S41)>=0
009-th inequality: weight = 4.000000    -H(W2)       A>=0
010-th inequality: weight = 6.000000    -H(S13)       B>=0
011-th inequality: weight = 3.000000     H(W1,W2,W3,W4)    -1.0*>=0
******************************************************************
******************************************************************
MIP dual value 29.000000
Proved 2-th inequality using integer values: 4A + 6B >= 3.
001-th inequality: weight = 1.000000     H(W1,W3,S21,S41)    -H(W1,W3,S21,S23,S41)     H(W1,S24,S31,S41)    -H(W1,S24,S41)>=0
002-th inequality: weight = 3.000000    -H(W1,W3,S21,S41)    2.0H(W1,S24)    -H(W2)>=0
003-th inequality: weight = 1.000000     H(W1,W3,S21,S41)    -H(W1,W4,S21)>=0
004-th inequality: weight = 7.000000    -H(W1,S24)     H(S13)     H(W2)>=0
005-th inequality: weight = 1.000000    -H(S13)     H(W1,S31)     H(S13,S24)    -H(W1,S23,S41)>=0
006-th inequality: weight = 1.000000    -H(W1,W2,W3,W4)    -H(W1,S31)     H(W1,W4,S21)     H(W1,S24,S41)>=0
007-th inequality: weight = 1.000000    -H(W1,W2,W3,W4)     H(W1,W3,S21,S41)     H(W1,W3,S21,S23,S41)    -H(W1,S24,S31,S41)>=0
008-th inequality: weight = 1.000000    -H(W1,W2,W3,W4)     H(W1,S24)    -H(S13,S24)     H(W1,S23,S41)>=0
```

```
009-th inequality: weight = 4.000000     -H(W2)       A>=0
010-th inequality: weight = 6.000000     -H(S13)       B>=0
011-th inequality: weight = 3.000000      H(W1,W2,W3,W4)    -1.0*>=0
*****************************************************************
```

The result can be interpreted as follows. The bound

$$4A + 6B \geq 3$$

in the problem description file can be proved by adding the 11 inequalities shown, with the weights given for each one. A constant value is marked using the "*". Note that the proof is solved twice, one using floating point values, and the other as an integer program; the latter sometimes yields a more concise proof, though not in this case. In practice, it may be preferable to perform one of them to reduce the overall computation.

The sensitivity analysis gives

```
Total number of elements before reduction: 65536
Total number of elements after reduction: 179
Total number of constraints given to Cplex: 40862
*****************************************************************
Optimal value for A + B = 0.625000.
Sensitivity results:
Sensitivity A                         = [0.37500, 0.37500]
Sensitivity B                         = [0.25000, 0.25000]
Sensitivity 2I(S12;S21|S32) + H(S21|S31) + A= [0.87500, 0.87500]
*****************************************************************
```

In this case, there does not exist any slack at this optimal value in these quantities.

8. Conclusion

In this work, we considered computational techniques to investigate fundamental limits of information systems. The disjoint-set data structure was adopted to identify the equivalence class mapping in an algorithmic manner, which is much more efficient than a naive linear enumeration. We provide an open source toolbox for four computational techniques. A JSON format frontend allows the toolbox to read a problem description file, convert it to the corresponding LP, and then produce meaningful bounds and other results directly without user intervention.

Acknowledgment

This work is supported by the National Science Foundation through grants CCF-18-16518 and CCF-18-16546.

1. Yeung, R.W. A framework for linear information inequalities. *IEEE Transactions on Information Theory* **1997**, *43*, 1924–1934.
2. Tian, C. Characterizing the rate region of the (4, 3, 3) exact-repair regenerating codes. *IEEE Journal on Selected Areas in Communications* **2014**, *32*, 967–975.
3. Tian, C.; Liu, T. Multilevel diversity coding with regeneration. *IEEE Transactions on Information Theory* **2016**, *62*, 4833–4847.
4. Tian, C. Symmetry, outer bounds, and code constructions: A computer-aided investigation on the fundamental limits of caching. *Entropy* **2018**, *20*, 603.1–43.
5. Tian, C.; Chen, J. Caching and delivery via interference elimination. *IEEE Transactions on Information Theory* **2018**, *64*, 1548–1560.
6. Tian, C.; Sun, H.; Chen, J. A Shannon-theoretic approach to the storage-retrieval tradeoff in PIR systems. 2018 Proceedings of IEEE International Symposium on Information Theory (ISIT), 2018, pp. 1904–1908.
7. Tian, C. On the storage cost of private information retrieval. *IEEE Transactions on Information Theory* **2020**, *66*, 7539–7549.
8. Galler, B.A.; Fisher, M.J. An improved equivalence algorithm. *Communications of the ACM* **1964**, *7*, 301–303.

9. IBM. *IBM ILOG CPLEX Optimizer*, 2009-2019.
10. Gurobi. *Gurobi Optimizer*, 2008-2019.
11. An Open-Source Toolbox for Computer-Aided Investigation on the Fundamental Limits of Information Systems. https://github.com/ct2641/CAI, 2020. Accessed: 2021-01.
12. Information Theoretic Inequality Prover (ITIP). http://user-www.ie.cuhk.edu.hk/~ITIP/, 1998-2008. Accessed: 2020-12.
13. Xitip: Information Theoretic Inequalities Prover. http://xitip.epfl.ch/, 2007. Accessed: 2020-12.
14. Yeung, R.W.; Lee, T.T.; Ye, Z. Information-theoretic characterizations of conditional mutual independence and Markov random fields. *IEEE Transactions on Information Theory* **2002**, *48*, 1996–2011.
15. Dougherty, R.; Freiling, C.; Zeger, K. Six new non-Shannon information inequalities. 2006 IEEE International Symposium on Information Theory. IEEE, 2006, pp. 233–236.
16. Dougherty, R.; Freiling, C.; Zeger, K. Non-Shannon information inequalities in four random variables. *arXiv preprint arXiv:1104.3602* **2011**.
17. Li, C.; Weber, S.; Walsh, J.M. Multilevel diversity coding systems: Rate regions, codes, computation, & forbidden minors. *IEEE Transactions on Information Theory* **2016**, *63*, 230–251.
18. Ho, S.W.; Ling, L.; Tan, C.W.; Yeung, R.W. Proving and disproving information inequalities: Theory and scalable algorithms. *IEEE Transactions on Information Theory* **2020**, *66*, 5522–5536.
19. Chan, T.; Thakor, S.; Grant, A. Minimal characterization of Shannon-type inequalities under functional dependence and full conditional independence structures. *IEEE Transactions on Information Theory* **2019**, *65*, 4041–4051.
20. Zhang, K.; Tian, C. On the symmetry reduction of information inequalities. *IEEE Transactions on Communications* **2018**, *66*, 2396–2408.
21. Li, C.; Weber, S.; Walsh, J.M. Multilevel diversity coding systems: Rate regions, codes, computation, & forbidden minors. *IEEE Transactions on Information Theory* **2017**, *63*, 230–251.
22. Li, C.; Weber, S.; Walsh, J.M. On multi-source networks: Enumeration, rate region computation, and hierarchy. *IEEE Transactions on Information Theory* **2017**, *63*, 7283–7303.
23. Apte, J.; Walsh, J.M. Exploiting symmetry in computing polyhedral bounds on network coding rate regions. 2015 International symposium on network coding (NetCod), 2015, pp. 76–80.
24. Li, C.; Apte, J.; Walsh, J.M.; Weber, S. A new computational approach for determining rate regions and optimal codes for coded networks. 2013 International Symposium on Network Coding (NetCod), 2013, pp. 1–6.
25. Apte, J.; Li, C.; Walsh, J.M. Algorithms for computing network coding rate regions via single element extensions of matroids. 2014 IEEE International Symposium on Information Theory, 2014, pp. 2306–2310.
26. Gattegno, I.B.; Goldfeld, Z.; Permuter, H.H. Fourier-Motzkin elimination software for information theoretic inequalities. *IEEE Information Theory Newsletter* **2015**, *65*, 25–29.
27. Gürpınar, E.; Romashchenko, A. How to use undiscovered information inequalities: Direct applications of the copy lemma. 2019 IEEE International Symposium on Information Theory (ISIT), 2019, pp. 1377–1381.
28. Cover, T.M.; Thomas, J.A. *Elements of Information Theory*, 1 ed.; New York: Wiley, 1991.
29. Yeung, R. *A First Course in Information Theory*; Kluwer Academic Publishers: New York, 2002.
30. Yeung, R.W. *Information theory and network coding*; Springer Science & Business Media, 2008.
31. Zhang, Z.; Yeung, R.W. A non-Shannon-type conditional inequality of information quantities. *IEEE Transactions on Information Theory* **1997**, *43*, 1982–1986.
32. Matus, F. Infinitely many information inequalities. 2007 IEEE International Symposium on Information Theory. IEEE, 2007, pp. 41–44.
33. Dougherty, R.; Freiling, C.; Zeger, K. Insufficiency of linear coding in network information flow. *IEEE Transactions on Information Theory* **2005**, *51*, 2745–2759.
34. Dougherty, R.; Freiling, C.; Zeger, K. Networks, matroids, and non-Shannon information inequalities. *IEEE Transactions on Information Theory* **2007**, *53*, 1949–1969.
35. Dimakis, A.G.; Godfrey, P.B.; Wu, Y.; Wainwright, M.J.; Ramchandran, K. Network coding for distributed storage systems. *IEEE transactions on information theory* **2010**, *56*, 4539–4551.
36. Dimakis, A.G.; Ramchandran, K.; Wu, Y.; Suh, C. A survey on network codes for distributed storage. *Proceedings of the IEEE* **2011**, *99*, 476–489.

37. Maddah-Ali, M.A.; Niesen, U. Fundamental limits of caching. *IEEE Transactions on Information Theory* **2014**, *60*, 2856–2867.
38. Lassez, C.; Lassez, J.L. Quantifier elimination for conjunctions of linear constraints via a convex hull algorithm. In *Symbolic and numerical computation for artificial intelligence*; Donald, B.R.; Kapur, D.; Mundy, J.L., Eds.; Academic Press: San Diego, CA, 1992; chapter 4, pp. 103–1199.