An Open-Source Toolbox for Computer-Aided Investigation on the Fundamental Limits of Information Systems, Version 0.1

Chao Tian, James S. Plank and Brent Hurst

https://github.com/ct2641/CAI/releases/tag/0.1

Citation Information - Plain text

Plain text:

```
author          C. Tian and J. S. Plank and B. Hurst
title           An Open-Source Toolbox for Computer-Aided Investigation
                on the Fundamental Limits of Information Systems, Version 0.1
howpublished    https://github.com/ct2641/CAI/releases/tag/0.1
month           October
year            2019
```

Citation Information - Bibtex

```
@MISC{tph:19:cai,
    author = "C. Tian and J. S. Plank and B. Hurst",
    title = "An Open-Source Toolbox for Computer-Aided Investigation
             on the Fundamental Limits of Information Systems, Version 0.1",
    howpublished = "https://github.com/ct2641/CAI/releases/tag/0.1",
    month = "October",
    year = "2019"
}
```

An Open-Source Toolbox for Computer-Aided Investigation on the Fundamental Limits of Information Systems, Version 0.1

Chao Tian, James S. Plank, and Brent Hurst

October 18, 2019

Abstract

We provide an open source toolbox at `https://github.com/ct2641/CAI/releases/tag/0.1` to conduct computer-aided investigation on the fundamental limits of information systems. The toolbox relies on either Gurobi or Cplex as the linear program solving engine. The program can read a problem description file, and then fulfill the following tasks: 1) compute a bound for a given linear combination of information measures; 2) efficiently compute a polytope tradeoff outer bound between two information quantities; and 3) produce a proof (as a weighted sum of known information inequalities). This technical report provides an overview of this toolbox, a detailed description of the syntax of the problem description file, and a few example use cases.

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 unweidly, 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 most well-known effort along this direction is perhaps the information theory inequality prover (ITIP) program [1, 2], which is designed to test whether a given information inequality is true or not. However, utilizing this generic inequality prover on any specific coding problem is a daunting task, and it also often fails to provide meaningful results due to the computation scale. Instead, a more desirable approach is to directly use a computer-aided approach on the specific problem of interest, whose inherent problem structure can help reduce the computation scale. With this approach, we can identify the fundamental limits and provide proofs directly from the problem description alone, without going through the additional step of first forming a conjecture on the information inequality to prove. This was indeed the approach taken in on several problems of recent interest in [3–7], and it turns out to be rather effective. These initial successes motivated our work on the toolbox.

The purpose of the toolbox is to help researchers in the field better utilize this computer-aided approach in their own research, and also to foster the discovery of more advanced computer-aided techniques. This toolbox is designed to provide a streamlined process of several already relatively mature methods. Generally speaking, if a problem can be represented by relations among the information measures, then the program can read a simple problem description file and produce the desired bounds, tradeoff region, or proofs, assuming

the scale of the problem does not exceed certain memory and word length restrictions. It is our hope that by opening up the source code, researchers will be able to find more creative techniques to further advance this computer-aided approach.

2 Entropy, Entropy LP Formulation, and Reductions

Although it is not strictly required to understand the mechanism behind the computer-aided approach to use this toolbox, such knowledge may become critical when a more efficient problem formulation is required or additional customized functionalities are needed in the toolbox. It also helps the readers understand the various sections in the problem description file we will discuss later. For this reason, we include a brief overview in this section; readers are referred to [8–10] for more details on information measures and the entropy linear programming (LP) problem.

2.1 Information Measures

The entropy of a random variable X distributed on a finite alphabet $\mathcal{X}$ is a mapping from a probability mass function to a real value. Denote the probability mass for any given letter $x \in \mathcal{X}$ as $p(x)$, then the entropy of the random variable X is denoted as

$$H(X) = -\sum_{x\in\mathcal{X}} p(x)\log p(x), \tag{1}$$

where we take the convention of defining $a\log a = 0$ when $a = 0$. Although we referred to $H(X)$ as the entropy of the random variable X, it is perhaps more accurate to refer to it as the entropy of a given probability mass function. The entropy of X is usually understood as a measure of the expected uncertainty in the random variable X. The joint entropy of n random variables $(X_1, X_2, \ldots, X_n)$ is defined similarly as

$$H(X_1, X_2, \ldots, X_n) = -\sum_{(x_1,x_2,\ldots,x_n)\in\prod_{i=1}^{n}\mathcal{X}_n} p(x_1, x_2, \ldots, x_n)\log p(x_1, x_2, \ldots, x_n). \tag{2}$$

The same way of generalization can be used for the information measures in the discussion that follows, and thus we only provide the basic definitions but not the generalized ones next.

The conditional entropy of two random variables X_1 given X_2 is defined as

$$H(X_1|X_2) = H(X_1, X_2) - H(X_2), \tag{3}$$

which is understood as a measure of the expected uncertainty in X_1 when X_2 is known. The mutual information between X_1 and X_2 is defined as

$$I(X_1; X_2) = I(X_2; X_1) = H(X_1) + H(X_2) - H(X_1, X_2), \tag{4}$$

which is understood as the information in X_1 about X_2, and vice versa. Similarly, the conditional mutual information is given as

$$\begin{aligned} I(X_1; X_2|X_3) &= H(X_1|X_3) + H(X_2|X_3) - H(X_1, X_2|X_3) \\ &= H(X_1, X_3) + H(X_2, X_3) - H(X_1, X_2, X_3) - H(X_3). \end{aligned} \tag{5}$$

For random variables with a continuous alphabet, the corresponding definitions are differential entropy, joint differential entropy, conditional differential entropy, and mutual information, however, in this work we will only work with random variables with finite and discrete alphabets.

2.2 Information Inequalities

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

$$H(X_1|X_2) \geq 0$$

$$I(X_1; X_2|X_3) \geq 0, \tag{6}$$

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 $X_1, X_2, \ldots, X_n$. These inequalities are usually referred to as Shannon-type inequalities [11]. Within the set of all information inequalities in the form shown in (6), 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 generate and utilize [12–15]. In practice, most bounds on the fundamental limits of information systems are derived using only Shannon-type inequalities, and we will mostly work only with them, though it is indeed possible to introduce non-Shannon-type inequalities into the problems within the toolbox.

2.3 The Entropy 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 can conceptually be understood as the following optimization problem:

$$\begin{aligned} \text{minimize: } & \text{a weighted sum of certain joint entropies} \\ \text{subject to: } & \text{(I) generic constraints that any information measures must satisfy} \\ & \text{(II) problem specific constraints on the information measures,} \end{aligned}$$

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 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 above only need to involve all joint entropies but not others, i.e., $2^n - 1$ quantities in the form of $H(X_\mathcal{A})$ where $\mathcal{A} \subseteq \{1, 2, \ldots, n\}$ and $X_\mathcal{A} = \{X_i, i \in \mathcal{A}\}$. The reason that we do not need to include conditional entropy, mutual information, or conditional mutual information in the problem is simple: they can be written simply as a linear combination of the joint entropies as given in (3)-(5).

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. In a mathematical optimization problem, we wish to use a concisely represented constrained set, and an immediate question is which set is the most concise. Yeung identified such a minimal set of constraints which are called elemental inequalities [9,11]:

$$H(X_i|X_\mathcal{A}) \geq 0, \quad i \in \{1, 2, \ldots, n\}, \qquad \mathcal{A} \subseteq \{1, 2, \ldots, n\} \setminus \{i\} \tag{7}$$

$$I(X_i; X_j|X_\mathcal{A}) \geq 0, \quad i \neq j,\, i, j \in \{1, 2, \ldots, n\}, \qquad \mathcal{A} \subseteq \{1, 2, \ldots, n\} \setminus \{i, j\} \tag{8}$$

Note that both (7) and (8) can be written as a linear constraint 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 relation 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

$$H(X_4|X_1, X_2) = H(X_1, X_2, X_4) - H(X_1, X_2) = 0, \tag{9}$$

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

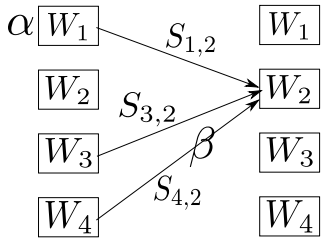

Figure 1: The regenerating code problem with $(n, k, d) = (4, 3, 3)$.

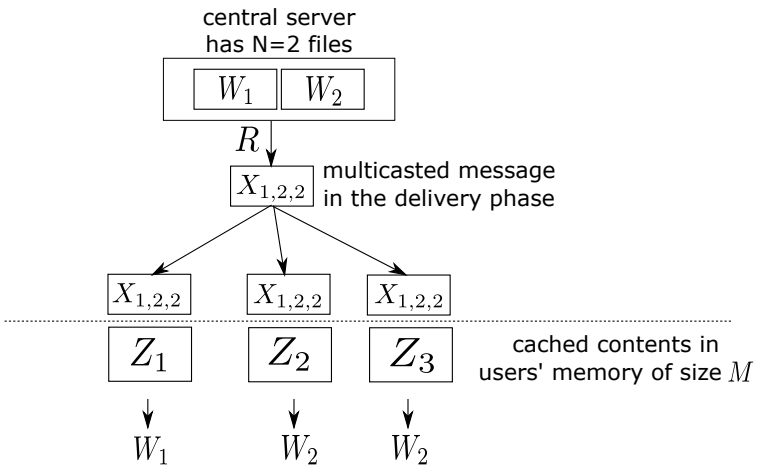

Figure 2: The caching problem with $(N, K) = (2, 3)$.

can be represented as

$$H(X_1, X_3, X_7) - H(X_1) - H(X_3) - H(X_7) = 0, \tag{10}$$

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. This optimization problem forms the basis of the functionalities of this toolbox.

3 Reduced Entropy LP

In this section, we discuss two techniques to reduce the complexity of the entropy LP, which are usually necessary if the problem is not extremely simple. Without using these techniques, many information system or coding problems of practical interest appear too complex to be solved directly in the entropy LP formulation. In order to be more specific, we first introduce two working examples that will be used throughout this document to illustrate the main idea.

3.1 Two Working Examples

The two working problems we shall use are the regenerating code problem and the coded caching problem:

- The (n, k, d) regenerating code problem [16, 17] considers the situation a message is stored in a distributed manner in n nodes, each having capacity α. Two coding requirements need to be satisfied: 1) the message can be recovered from any k nodes, and 2) any single node can be repaired by downloading β amount of information from any d remaining nodes each. The fundamental limit of interest is

the optimal tradeoff between the storage cost α and the download cost β; see Fig. 1. We will use the $(n, k, d) = (4, 3, 3)$ case as our working example. In this setting, the stored contents as 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 in this setting 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. Later on, we will discuss another layer of simplification when dealing with $(n, k, d) = (5, 4, 4)$.

- The (N, K) coded caching problem [18] considers the situation that a server, which holds a total of N mutually independent files of unit size each, serves a set of K users, each with a local cache size M. The users can prefetch some content, but when they reveal their requests, the server must multicast certain common information of size R. 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; see Fig. 2. 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 $(d_1, d_2, \ldots, d_K)$ are 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:

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

3.2 The Dependency Reduction

The dependency (or implication) relation, e.g., the one given in (9), can be included in the optimization problem in different ways. The first option, which is the simplest, is to include these equality constraints directly in the set of constraints of the LP. There is however, another indirect 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 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

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

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

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

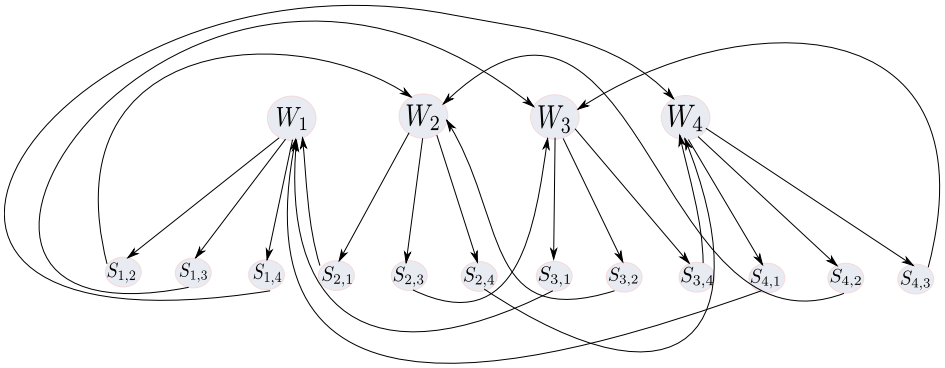

Figure 3: The dependence graph for the regenerating code problem with $(n, k, d) = (4, 3, 3)$.

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

3.3 The Symmetry Reduction

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.

- 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 their other 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:

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

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

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 $2, 9, 15$ columns, then we have that

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

There are a total of $2^{17} - 1$ subsets of columns, and they each will induce a set of equality relations, but some of them may be equivalent. For a more rigorous discussion of this symmetry relation, the readers can refer to [3, 19].

- Similarly, in the caching problem, there are two types of symmetry relations. The first is to exchange the coding functions for each user, 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:

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

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

For a more detailed discussion on this symmetry relation, the readers can refer to [5, 19].

Three cautions should be given at this point when working with symmetry relation:

- The symmetry reduction is valid in the entropy LP, when it is without loss of optimality to consider only the corresponding symmetry codes. Such assertion is usually not too difficult to make with a concrete problem in mind, but it can be less clear when the problem is abstract.
- The precise permutations can be subtle. For example, observe in the caching example, the difference between the first row and the eleventh row is induced by the permutation of the user contents into (Z_3, Z_1, Z_2), and as a result, the indices of the $X_{i,j,k}$ variables will change. However, the precise way it is changed to is less intuitive and needs to be carefully worked out. In fact, when there are more than two messages, the permutation is even more complicated; readers are referred to [5] for a thorough discussion.
- 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 only 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.

4 Three Streamlined Computer-Aided Techniques

There are three computer-aided techniques that we have implemented in this toolbox and are ready for use. In this section, we provide an overview of the usages.

4.1 Find a Bounding Plane of Fixed Direction

This is the case when a bound of a specific combination needs to be found. Fig. 4 illustrates this specific application, 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 sum of the storage cost and repair cost for 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 $H(X_{1,2,2})$, 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

  $$H(X_{1,1,1}) \le R$$
  $$H(X_{1,2,2}) \le R,$$

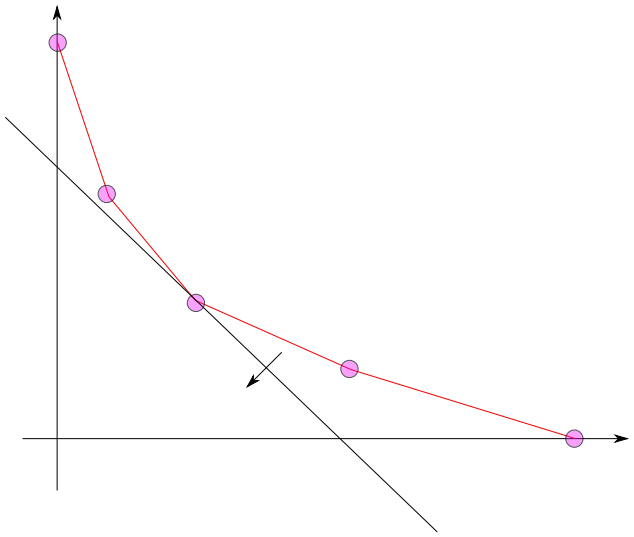

Figure 4: A fixed direction bounding plane and the tradeoff region computation.

and then set the objective function to be

$$H(Z_1) + R,$$

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

4.2 Tradeoff and Convex Hull Computation

In many cases, instead of bounding a fix rate combination, we are interested in the tradeoff of several quantities, in fact, most time 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 very 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 in fact provides 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 [20], and the specialization used in the program can be found in [5].

4.3 Duality and Computer-Generated Proof

After identifying a valid outer bound, we sometimes wish to find a proof for this bound. For example, in the regenerating code problem, we have

$$H(W_1) + H(S_{1,2}) \geq \frac{5}{8}. \tag{11}$$

How can we prove this inequalities? 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 [3, 5, 21, 22].

In this toolbox, this task of finding a proof is implemented, which will print out a computed weight sum. Note, however, such a proof is not unique, and the proof such found may lack a structure desirable for further generalization.

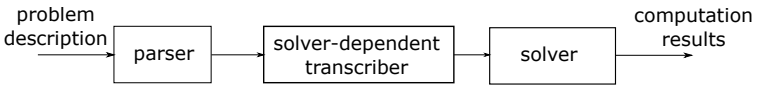

Figure 5: The computation pipeline.

5 Software Dependence and Computation Pipeline

The toolbox needs a linear programming solver as the backend. There are two options currently allowed in the toolbox: Cplex [23] or Gurobi [24]. Both solvers are known to perform extremely well for large scale linear programs, and offer free license for academic use (but not commercial use). There is no need to have both solvers, and installing one package will be sufficient. A make file is given in the package, and a visual C++ solution configuration (VC2017) is also given. The only place that we need to pay attention is to link the solver library correctly. The toolbox otherwise has no dependence on third party packages.

The toolbox has a relatively sophisticated front-end parser, and this may be one of the most notable features different from existing efforts [1, 2]. The parser will parse a problem description file into an inherent data structure, after certain error and syntax checking. Before the solver can be called, the problem must also be transcribed into the linear program using the reduction technique we previous discussed. The parser is not solver dependent, but the transcriber is solver dependent, as the transcribing process can be done more efficiently by utilizing the solver APIs. After the problem is transcribed to an LP suitable for the chosen solver, the solver can then be called to solve the program. Depending on the mode of computation, the results are then translated and print to output. The computation pipeline is illustrated in Fig. 5.

6 Syntax for Problem Description Files

The main computations are implemented such that the program will read a problem description file (a plain text file), and the desired computed bounds or proof will be produced without further user intervention. In this section, we provide the details of the syntax of the problem description file.

6.1 Key Phrases

There are a total of 9 key phrases allowed in the description file, but in future release we may include more. The key phrases are:

random variables, additional LP variables, objective, dependency, conditional independence, constant bounds, symmetry, bounds to prove, end

Each key phrase must be on a dedicated line and any content after the key phrase will be ignored on that line. The description file must end with the keyword **end**. Each keyword will start a new section. The section corresponding to the key phrase **random variables** cannot be empty but all other sections are optional, depending on the problem and the specific computation to be performed. The key phrases are not case sensitive.

The program will assume the problem to be minimization, i.e., to find a lower bound for certain information quantity. If instead we need to derive an upper bound, then the objective function will need to be written in its negated form. Comments in the problem description file are allowed, which must appear as a new line starting with "//".

6.2 Examples

Below are two example problem description files for our two working examples.

```
//PDRG4x3x3.txt: problem description file for the (4,3,3) regenerating code problem.

Random variables:
W1,W2,W3,W4,S12,S13,S14,S21,S23,S24,S31,S32,S34,S41,S42,S43

Additional LP variables:
A,B

Objective:
A+B

Dependency:
S12,S13,S14:W1
S21,S23,S24:W2
S31,S32,S34:W3
S41,S42,S43:W4
W1:S21,S31,S41
W2:S12,S32,S42
W3:S13,S23,S43
W4:S14,S24,S34

Constant bounds:
H(W1)-A<=0
H(S12)-B<=0
H(W1,W2,W3,W4)>=1

Symmetry:
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

Bounds to prove:
8A+12B>=6

end
```

```
// problem description file for the (2,3) caching problem.
Random variables:
W1, W2, Z1, Z2, Z3, X111,X112,X121,X122,X211,X212,X221,X222

Additional LP variables:
M,R

Objective:
M+2R

Dependency:
Z1,Z2,Z3,X111,X112,X121,X122,X211,X212,X221,X222: W1, W2
W1: Z1, X111
W1: Z2, X111
W1: Z3, X111
W1: Z1, X112
W1: Z2, X112
W2: Z3, X112
W1: Z1, X121
W2: Z2, X121
W1: Z3, X121
W1: Z1, X122
W2: Z2, X122
W2: Z3, X122
W2: Z1, X211
W1: Z2, X211
W1: Z3, X211
W2: Z1, X212
W1: Z2, X212
W2: Z3, X212
W2: Z1, X221
W2: Z2, X221
W1: Z3, X221
W2: Z1, X222
W2: Z2, X222
W2: Z3, X222

Conditional independence:
W1;W2

Constant bounds:
H(W1) = 1
H(X111) -R<= 0
H(X112) -R<= 0
H(Z1) -M<= 0

Symmetry:
W1, W2, Z1, Z2, Z3, X111,X112,X121,X122,X211,X212,X221,X222
W2, W1, Z1, Z2, Z3, X222,X221,X212,X211,X122,X121,X112,X111
W1, W2, Z2, Z1, Z3, X111,X112,X211,X212,X121,X122,X221,X222
W2, W1, Z2, Z1, Z3, X222,X221,X122,X121,X212,X211,X112,X111
W1, W2, Z3, Z2, Z1, X111,X211,X121,X221,X112,X212,X122,X222
W2, W1, Z3, Z2, Z1, X222,X122,X212,X112,X221,X121,X211,X111
W1, W2, Z1, Z3, Z2, X111,X121,X112,X122,X211,X221,X212,X222
W2, W1, Z1, Z3, Z2, X222,X212,X221,X211,X122,X112,X121,X111
W1, W2, Z2, Z3, Z1, X111,X211,X112,X212,X121,X221,X122,X222
W2, W1, Z2, Z3, Z1, X222,X122,X221,X121,X212,X112,X211,X111
W1, W2, Z3, Z1, Z2, X111,X121,X211,X221,X112,X122,X212,X222
W2, W1, Z3, Z1, Z2, X222,X212,X122,X112,X221,X211,X121,X111

end
```

6.3 Syntax for the Key Phrase Sections

We next specify the syntax for each key phrase section.

1. Random variables: this is the complete list of all the random variables in the problem; each random variable should be less than 32 characters; only letters a-z, A-Z and 0-9 are allowed in the random variable name; a random variable name is required to start with a letter; random variables are separated using comma or semi-colon in this section; this section should always be the first section in the problem description file.

2. Additional LP variables: as mentioned early, sometimes we have to introduce additional auxiliary LP variables to bound multiple quantities at the same values; these additional LP variables follow the same naming convention as the random variables; this section, if exists, should follow immediately the "random variables" section.

3. Objective: the combination of information measures to be minimized; written on a single line.

4. Dependency: dependency relation of (X, Y) are functions of (Z, W) is written as $X, Y : Z, W$, i.e., $H(X, Y|Z, W) = 0$; one dependency relation per line.

5. Conditional independence: the form $X; Y; Z|T$ means X, Y, Z are mutually independent given T; one conditional independence per line.

6. Constant bounds: equality or inequality bounds for mutual information or conditional entropy; one bound per line; the left hand side can be combinations of information measures, but the right hand side must be a constant.

7. Symmetry: symmetry relation is represented as a permutation group; one permutation per line; commas are used to separate random variables; the first row must be identical to the list of random variables.

8. Bounds to prove: the bounds to be proved if computer-generated proofs are sought after; one bound per line.

9. End: to indicate the end of the description file.

Given these specifications, the two example problem description files are mostly self-explanatory. The only thing to note is perhaps for both problems, we introduced additional LP variables, some of which are not strictly necessary. However, having these LP variables helps conceptual understanding and leads to more readable problem descriptions.

6.4 Command Line Input

The problem description file can include both an "objective" function section and a "bounds to prove" section. The precise computation is then determined by the command line input. The main computation commands are CplexCompute and GurobiCompute, and the command line follows the same format.

```
CplexCompute foo.txt [options]
```

or

```
GurobiCompute foo.txt [options]
```

The command line argument follows a simple rule:

- Here "foo.txt" stands for the problem description file. If no other option given, the program will assume an objective function is included in the problem description file, and simply minimize that objective function. The bound value will be shown, together with the corresponding values for the quantities in the objective function.

- The options can be either "hull" or "prove". For the hull computation, the objective function is analyzed and the two quantities in the objective function (must be two joint entropies or additional LP variables) are viewed as the two quantities whose tradeoff are being sought after; then the tradeoff hull is computed and shown as the corner points.
- For the "prove" option, the bounds given in the "bounds to prove" section are proven one by one, and the weighted combinations are shown. This process is in fact done twice, the first as a real-valued LP, and the second as an integer program. The two solvers may produce different solutions, and particularly, the integer program solver may produce a sparser and more desirable solution.

6.5 Example Command Lines and Computation Results

Consider the command line

```
CplexCompute PDRG4x3x3.txt
```

where the problem description file is for our working example of the regenerating code problem, then the program shows the following output.

```
****************************************************************
-The following 16 random variables were found:
W1 W2 W3 W4 S12 S13 S14 S21 S23 S24 S31 S32 S34 S41 S42 S43
---The problem has 2 additional LP variables.
---The objective function has 2 non-zero terms.
---The problem has 8 dependency relations.
---The problem has 3 constant value bounds.
---Permutations in the symmetry relation =  24.
---Number of bounds to prove =  1.
****************************************************************
Total number of elements to reduce: 65536
CPXPARAM_Read_DataCheck                          1
Tried aggregator 1 time.
DUAL formed by presolve
LP Presolve eliminated 38643 rows and 3 columns.
Reduced LP has 177 rows, 5084 columns, and 17831 nonzeros.
Presolve time = 0.05 sec. (24.27 ticks)
Parallel mode: using up to 20 threads for barrier.
Number of nonzeros in lower triangle of A*A' = 5960
Using Approximate Minimum Degree ordering
Total time for automatic ordering = 0.00 sec. (0.44 ticks)
Summary statistics for Cholesky factor:
  Threads                  = 20
  Rows in Factor           = 177
  Integer space required   = 923
  Total non-zeros in factor = 12808
  Total FP ops to factor   = 1232248
 Itn      Primal Obj        Dual Obj  Prim Inf Upper Inf  Dual Inf Inf Ratio
   0   1.0000000e+01   0.0000000e+00  3.77e+04  0.00e+00  5.26e+03  1.00e+00
   1   6.1373374e+00   1.5078468e+00  2.60e+04  0.00e+00  3.34e+03  2.44e+00
   2   2.2445860e+00   1.4338756e+00  1.07e+04  0.00e+00  1.06e+03  8.08e+04
   3   1.5228039e+00   9.6993458e-01  4.73e+03  0.00e+00  1.81e+02  9.34e+01
   4   1.0830951e+00   9.9552024e-01  7.79e+02  0.00e+00  1.94e+01  1.01e+03
   5   1.0262480e+00   9.8688399e-01  3.25e+02  0.00e+00  2.65e+00  7.06e+03
   6   1.0237260e+00   9.7472519e-01  3.14e+02  0.00e+00  1.78e+00  6.55e+03
   7   9.5978567e-01   9.5337070e-01  4.00e+01  0.00e+00  2.14e-01  5.25e+04
   8   9.0444925e-01   8.9317974e-01  2.54e+01  0.00e+00  1.26e-01  6.17e+04
   9   8.0010814e-01   8.3512233e-01  7.88e+00  0.00e+00  7.35e-02  8.41e+04
  10   7.6631321e-01   7.7048934e-01  5.74e+00  0.00e+00  4.38e-02  1.33e+05
  11   7.0912848e-01   7.0989975e-01  2.21e+00  0.00e+00  1.64e-02  3.38e+05
  12   6.5432023e-01   6.5701740e-01  6.27e-01  0.00e+00  5.20e-03  9.72e+05
  13   6.2554618e-01   6.2728184e-01  1.45e-02  0.00e+00  3.91e-04  1.33e+07
  14   6.2499995e-01   6.2500055e-01  1.44e-06  0.00e+00  1.13e-07  5.06e+10
  15   6.2500000e-01   6.2500000e-01  2.11e-10  0.00e+00  1.58e-11  4.65e+14
Barrier time = 0.11 sec. (33.54 ticks)

Total time on 20 threads = 0.11 sec. (33.54 ticks)

****************************************************************
Optimal value is 0.625000.
Values achieving the optimal solution:
 0.375000   0.250000
****************************************************************
```

The first part in the star-separated segment is essentially what the parser reads from the problem description file. If there are warnings or errors in the problem description file, they will be shown in this segment, and the problem may either terminate directly, or continue to run. The warning should not be ignored in most cases, since the computation results may not be correct if a warning persists. The next part of the output content is the computation progress given the chosen solver. The program can be configured not to output such information, but for large problems, such information can be useful and thus we recommend to

keep them. The last star-separated segment means that we have a bound

$$\alpha + \beta \geq 0.625,$$

and the value for the two quantities (α, β) in the LP optimal solution are $(0.375, 0.25)$.

If instead we run the following command line

```
CplexCompute PDRG4x3x3.txt hull
```

the result shows:

```
*****************************************************************
-The following 16 random variables were found:
W1 W2 W3 W4 S12 S13 S14 S21 S23 S24 S31 S32 S34 S41 S42 S43
---The problem has 2 additional LP variables.
---The objective function has 2 non-zero terms.
---The problem has 8 dependency relations.
---The problem has 3 constant value bounds.
---Permutations in the symmetry relation =  24.
---Number of bounds to prove =  1.
*****************************************************************
Total number of elements to reduce: 65536
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.
```

In this case, the solver output is suppressed, and it is seen that the three (α, β) pairs are the corner points of the lower convex hull of the tradeoff. Occasionally, certain non-corner points on the boundary will also be included in the list, but this does not invalidate the hull. In other words, every corner points of the convex hull will be in this list, but not all the points in this list will be a corner point. The non-corner points can be addressed easily in subsequent processing.

Finally if we run the following command:

```
CplexCompute PDRG4x3x3.txt prove
```

we will have the following result:

```
****************************************************************
~The following 16 random variables were found:
W1 W2 W3 W4 S12 S13 S14 S21 S23 S24 S31 S32 S34 S41 S42 S43
---The problem has 2 additional LP variables.
---The objective function has 2 non-zero terms.
---The problem has 8 dependency relations.
---The problem has 3 constant value bounds.
---Permutations in the symmetry relation =  24.
---Number of bounds to prove =  1.
****************************************************************
Total number of elements to reduce: 65536
CPXPARAM_Read_DataCheck                         1
Parallel mode: deterministic, using up to 20 threads for concurrent optimization.
Tried aggregator 1 time.
LP Presolve eliminated 4 rows and 38641 columns.
Reduced LP has 177 rows, 5085 columns, and 17832 nonzeros.
Presolve time = 0.03 sec. (18.50 ticks)

Iteration log . . .
Iteration:     1   Dual objective     =            32.000000
Iteration:    62   Dual objective     =            48.000000
Iteration:   124   Dual objective     =            48.000000
Iteration:   186   Dual objective     =            48.000000
Perturbation started.
Iteration:   202   Dual objective     =            48.000000
Iteration:   264   Dual objective     =            52.096570
Iteration:   326   Dual objective     =            53.448280
Iteration:   388   Dual objective     =            54.075768
Iteration:   450   Dual objective     =            54.292218
Iteration:   512   Dual objective     =            54.400013
Iteration:   574   Dual objective     =            54.949171
Iteration:   636   Dual objective     =            55.361112
Iteration:   698   Dual objective     =            56.000004
Iteration:   760   Dual objective     =            56.631583
Iteration:   822   Dual objective     =            57.454543
Iteration:   884   Dual objective     =            57.999997
Iteration:   946   Dual objective     =            58.000003
Removing perturbation.

Dual simplex solved model.

****************************************************************
LP dual value 58.000000
Proved 0-th inequality.
001-th inequality: weight = 2.000000     -H(W1,S21)     H(W1,S21,S31)     H(W1,S23,S31)    -H(W1,S21,S23,S31)>=0
002-th inequality: weight = 2.000000     -H(W1,S23,S31)     H(W1,S23,S31,S41)     H(W2,S12,S13,S14,S31,S41)    -H(W2,S12,S13,S14,S31,S32,S41)>=0
003-th inequality: weight = 6.000000     2.0H(S12)    -H(S13,S23)>=0
004-th inequality: weight = 8.000000      H(S12)     H(W1)    -H(W1,S23)>=0
005-th inequality: weight = 6.000000     -H(S12)     H(S13,S23)     H(W1,S23)    -H(W2,S12,S13,S14,S31,S41)>=0
006-th inequality: weight = 2.000000     -H(W1)     H(W1,S23)     H(W1,W2)    -H(S12,S13,S14,S21,S23,S24,S31,S32,S34,S41,S42,S43)>=0
007-th inequality: weight = 2.000000     -H(W1,S21,S31)     H(W1,S21,S23,S31)     H(W2,S12,S13,S14,S31,S41)    -H(S12,S13,S14,S21,S23,S24,S31,S32,S34,S41,S42,S43)>=0
008-th inequality: weight = 2.000000     -H(W1,S23,S31,S41)     H(W2,S12,S13,S14,S31,S41)     H(W2,S12,S13,S14,S31,S32,S41)    -H(S12,S13,S14,S21,S23,S24,S31,S32,S34,S41,S42,S43)>=0
009-th inequality: weight = 2.000000     -H(S12)     H(W1)     H(W1,S21)    -H(W1,W2)>=0
010-th inequality: weight = 8.000000     -H(W1)       A>=0
011-th inequality: weight = 12.000000     -H(S12)       B>=0
012-th inequality: weight = 6.000000      H(S12,S13,S14,S21,S23,S24,S31,S32,S34,S41,S42,S43)    -1.0*>=0
****************************************************************
CPXPARAM_Read_DataCheck                         1
Tried aggregator 1 time.
MIP Presolve eliminated 4 rows and 38641 columns.
Reduced MIP has 177 rows, 5085 columns, and 17832 nonzeros.
Reduced MIP has 0 binaries, 5085 generals, 0 SOSs, and 0 indicators.
Presolve time = 0.05 sec. (25.25 ticks)
Tried aggregator 1 time.
Reduced MIP has 177 rows, 5085 columns, and 17832 nonzeros.
Reduced MIP has 0 binaries, 5085 generals, 0 SOSs, and 0 indicators.
Presolve time = 0.00 sec. (5.63 ticks)
MIP emphasis: balance optimality and feasibility.
MIP search method: dynamic search.
Parallel mode: deterministic, using up to 20 threads.
Root relaxation solution time = 0.09 sec. (106.53 ticks)

        Nodes                                         Cuts/
   Node  Left     Objective  IInf  Best Integer    Best Bound    ItCnt     Gap

*     0     0      integral     0       58.0000       58.0000      990    0.00%
Elapsed time = 0.25 sec. (239.18 ticks, tree = 0.00 MB, solutions = 1)

Root node processing (before b&c):
  Real time             =    0.25 sec. (240.36 ticks)
Parallel b&c, 20 threads:
  Real time             =    0.00 sec. (0.00 ticks)
  Sync time (average)   =    0.00 sec.
  Wait time (average)   =    0.00 sec.
                          ------------
Total (root+branch&cut) =    0.25 sec. (240.36 ticks)
****************************************************************
MIP dual value 58.000000
Proved 0-th inequality using integer values.
001-th inequality: weight = 2.000000     -H(W1,S23,S31)     H(W1,S23,S31,S41)     H(W2,S12,S13,S14,S31,S41)    -H(W2,S12,S13,S14,S31,S32,S41)>=0
002-th inequality: weight = 2.000000     -H(W1,S21)     H(W1,S21,S31)     H(W1,S23,S31)    -H(W1,S21,S23,S31)>=0
003-th inequality: weight = 6.000000     -H(W1)    2.0H(W1,S23)    -H(W2,S12,S13,S14,S31,S41)>=0
004-th inequality: weight = 14.000000      H(S12)     H(W1)    -H(W1,S23)>=0
005-th inequality: weight = 2.000000     -H(S12)     H(W1,S21)     H(S14,S23)    -H(W1,S24,S31)>=0
006-th inequality: weight = 2.000000     -H(W1,S21,S31)     H(W1,S21,S23,S31)     H(W2,S12,S13,S14,S31,S41)    -H(S12,S13,S14,S21,S23,S24,S31,S32,S34,S41,S42,S43)>=0
007-th inequality: weight = 2.000000     -H(W1,S23,S31,S41)     H(W2,S12,S13,S14,S31,S41)     H(W2,S12,S13,S14,S31,S32,S41)    -H(S12,S13,S14,S21,S23,S24,S31,S32,S34,S41,S42,S43)>=0
008-th inequality: weight = 2.000000     -H(S14,S23)     H(W1,S23)     H(W1,S24,S31)    -H(S12,S13,S14,S21,S23,S24,S31,S32,S34,S41,S42,S43)>=0
009-th inequality: weight = 8.000000     -H(W1)       A>=0
010-th inequality: weight = 12.000000     -H(S12)       B>=0
011-th inequality: weight = 6.000000      H(S12,S13,S14,S21,S23,S24,S31,S32,S34,S41,S42,S43)    -1.0*>=0
****************************************************************
```

The result can be interpreted as follows. The bound

$$8A + 12B \geq 6$$

in the problem description file can be proved by adding the 12 inequalities shown at the end, with the weights given for each one. A constant value is marked using the "*".

7 Errors, Warnings, and Auxiliary Functions

7.1 Errors and Warnings

Most of the error or warning messages will be encountered due to problem description syntax errors. For example, when the same random variable appears in the same row of the permutation twice, or a given line has a character not allowed in that section, e.g., "|" in the dependency section.

One particular error checking we perform is to make sure that the input permutations form a valid permutation group. If they do not form a permutation group, then the reduction used in the program may not be valid, and as a consequence, the computed result may not be a valid bound. This type of error will trigger a message

```
---Permutations appear not to form a valid group.
```

If this message appears, the reader is advised to carefully examine the permutation section to eliminate any input error.

7.2 Auxiliary Functions on Symmetry Relations

The symmetry structure permutations become tedious to write manually for complex problems. One can use any scripting language, such as Matlab or Python, to produce the this section of the problem description file. One such set of Matlab scripts to generate the permutation group is included in the package which is used to generate the problem descriptions for the regenerating code problem. The readers may also find the GAP package [25] useful for this purpose. In future releases we plan to include more complex symmetry structures.

8 A Remark on Problem Formulation

Despite the various reductions we employ in the toolbox, the computation scale is still usually quite large. As a rule of thumb, the toolbox can usually handle problems with less than or equal to 20 random variables handily, which in many cases has already provided surprisingly powerful insights [3–5]. Depending on the problem structure, particularly the symmetry structure, it might be possible to produce meaningful results for problems with 25$\sim$27 random variables. Problems at an even large scale are likely to trigger memory issue or cause the overflow due to the word length restriction.

Problem formulation plays an extreme important role in utilizing this toolbox. Let us reconsider the regenerating code problem with parameter $(n, k, d) = (5, 4, 4)$. A simple extension of the $(4, 3, 3)$ formulation we discuss above will lead to a total of 25 random variables in the problem, which, though possible, poses a challenge in both memory usage and computation time. However, notice that we can in fact eliminate the random variables $(W_1, W_2, \ldots, W_5)$ altogether, by simply viewing W_i as the collection of $\{(S_{i,j}, j \neq i\}$. This helps to reduce the number of variables to 20, and the computation can be done without much difficulty in the toolbox. The first five sections of the problem description file are as follows in this case, and the we omitted the symmetry portion here for better formatting.

```
Random variables:
S12,S13,S14,S15,S21,S23,S24,S25,S31,S32,S34,S35,S41,S42,S43,S45,S51,S52,S53,S54

Additional LP variables:
A,B

Objective:
A+B

Dependency:
S12,S13,S14,S15:S21,S31,S41,S51
S21,S23,S24,S25:S12,S32,S42,S52
S31,S32,S34,S35:S13,S23,S43,S53
S41,S42,S43,S45:S14,S24,S34,S54
S51,S52,S53,S54:S15,S25,S35,S45

Constant bounds:
H(S12,S13,S14,S15)-A<=0
H(S12)-B<=0
H(S12,S13,S14,S15,S21,S23,S24,S25,S31,S32,S34,S35,S41,S42,S43,S45,S51,S52,S53,S54)>=1
```

The computation for the simple bounding will then produce the following result in about 21 seconds:

```
****************************************************************
-The following 20 random variables were found:
S12 S13 S14 S15 S21 S23 S24 S25 S31 S32 S34 S35 S41 S42 S43 S45 S51 S52 S53 S54
---The problem has 2 additional LP variables.
---The objective function has 2 non-zero terms.
---The problem has 5 dependency relations.
---The problem has 3 constant value bounds.
---Permutations in the symmetry relation =  120.
****************************************************************
Total number of elements to reduce: 1048576
1048576 constraints.
CPXPARAM_Read_DataCheck                          1
Tried aggregator 1 time.
DUAL formed by presolve
LP Presolve eliminated 1107201 rows and 3 columns.
Reduced LP has 7008 rows, 339847 columns, and 1316929 nonzeros.
Presolve time = 3.56 sec. (1707.11 ticks)
Parallel mode: using up to 20 threads for barrier.
Number of nonzeros in lower triangle of A*A' = 648673
Using Nested Dissection ordering
Total time for automatic ordering = 2.56 sec. (2818.41 ticks)
Summary statistics for Cholesky factor:
  Threads                  = 20
  Rows in Factor           = 7008
  Integer space required   = 100794
  Total non-zeros in factor = 16175104
  Total FP ops to factor   = 46968958308
 Itn      Primal Obj        Dual Obj  Prim Inf Upper Inf  Dual Inf Inf Ratio
   0   1.0000000e+01   0.0000000e+00  8.80e+05  0.00e+00  3.47e+05  1.00e+00
   1   2.0701651e+01   8.4192498e+00  6.94e+05  0.00e+00  2.74e+05  8.14e-01
   2   1.8767298e+01   4.2175046e+00  6.48e+05  0.00e+00  1.19e+05  1.72e+04
   3   5.1351847e+00   1.5379600e+00  1.61e+05  0.00e+00  3.55e+04  6.96e+01
   4   2.4336848e+00   8.6068979e-01  6.89e+04  0.00e+00  1.51e+04  1.08e+02
   5   1.1867538e+00   8.8624625e-01  1.31e+04  0.00e+00  2.05e+03  6.10e+02
   6   9.5421612e-01   8.0889031e-01  5.67e+03  0.00e+00  6.13e+02  1.61e+03
   7   8.0302220e-01   7.9448091e-01  3.73e+02  0.00e+00  7.50e+01  1.51e+04
   8   7.6424567e-01   7.6226224e-01  8.06e+01  0.00e+00  1.58e+01  1.08e+05
   9   6.9692539e-01   7.1327378e-01  3.13e+01  0.00e+00  8.66e+00  2.42e+05
  10   6.8938900e-01   7.1174542e-01  2.76e+01  0.00e+00  8.50e+00  2.48e+05
  11   6.7307981e-01   6.7140492e-01  2.28e+01  0.00e+00  5.54e+00  4.39e+05
  12   6.3338705e-01   6.5034196e-01  1.28e+01  0.00e+00  4.24e+00  6.14e+05
  13   6.1294376e-01   6.3985114e-01  9.84e+00  0.00e+00  3.81e+00  7.01e+05
  14   5.9689679e-01   6.1129887e-01  7.80e+00  0.00e+00  2.77e+00  1.07e+06
  15   5.7525705e-01   5.8530365e-01  5.37e+00  0.00e+00  1.91e+00  1.73e+06
  16   5.6002555e-01   5.6882731e-01  4.15e+00  0.00e+00  1.50e+00  2.34e+06
  17   5.4082887e-01   5.4964423e-01  2.87e+00  0.00e+00  1.10e+00  3.48e+06
  18   5.2644370e-01   5.3383287e-01  2.15e+00  0.00e+00  8.34e-01  4.91e+06
  19   5.1415031e-01   5.1691945e-01  1.65e+00  0.00e+00  5.99e-01  7.44e+06
  20   4.9550420e-01   5.0003185e-01  9.98e-01  0.00e+00  3.97e-01  1.24e+07
  21   4.9236974e-01   4.9058261e-01  9.14e-01  0.00e+00  3.07e-01  1.70e+07
  22   4.8443269e-01   4.8213260e-01  6.97e-01  0.00e+00  2.25e-01  2.46e+07
  23   4.7918387e-01   4.7472618e-01  5.66e-01  0.00e+00  1.60e-01  3.67e+07
  24   4.6915900e-01   4.6677646e-01  3.32e-01  0.00e+00  9.57e-02  6.69e+07
  25   4.6089628e-01   4.5912636e-01  1.72e-01  0.00e+00  4.60e-02  1.57e+08
  26   4.5151414e-01   4.5150408e-01  1.84e-02  0.00e+00  5.87e-03  1.50e+09
  27   4.5000015e-01   4.5000117e-01  3.42e-06  0.00e+00  5.09e-06  1.94e+12
  28   4.5000000e-01   4.5000000e-01  1.28e-09  0.00e+00  1.75e-09  6.03e+15
Barrier time = 21.08 sec. (17867.86 ticks)

Total time on 20 threads = 21.08 sec. (17867.86 ticks)

****************************************************************
Optimal value is 0.450000.
Values achieving the optimal solution:
 0.300000   0.150000
****************************************************************
```

9 Conclusion and Future Work

We provide an open source toolbox for computer-aided investigation of the fundamental limits of information systems. The given program is designed to read a problem description file, convert it to the corresponding LP, and then produce meaningful bounds directly without much user intervention.

There are several places we plan to further improve in the toolbox in future releases. Particularly, several advanced techniques to improve the computation efficiency used in [5] were not included in this version for ease of use and simpler problem description syntax, and we plan to incorporate them in the future. Several functions are implemented using naive data structures which hinder the performance when the problem is

large, and we plan to replace them with more suitable data structures. Another area to improve is result data interpretation and visualization, the latter of which is not included, but is extremely useful in the first author's experience.

Acknowledgment

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

References

[1] *Information Theoretic Inequality Prover (ITIP)*, 1998-2008. [Online]. Available: http://user-www.ie.cuhk.edu.hk/~ITIP/

[2] *Xitip: Information Theoretic Inequalities Prover*, 2007. [Online]. Available: http://xitip.epfl.ch/

[3] C. Tian, "Characterizing the rate region of the (4, 3, 3) exact-repair regenerating codes," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 5, pp. 967–975, May 2014.

[4] C. Tian and T. Liu, "Multilevel diversity coding with regeneration," *IEEE Transactions on Information Theory*, vol. 62, no. 9, pp. 4833–4847, Sep. 2016.

[5] C. Tian, "Symmetry, outer bounds, and code constructions: A computer-aided investigation on the fundamental limits of caching," *Entropy*, vol. 20, no. 8, pp. 603.1–43, Aug. 2018.

[6] C. Tian and J. Chen, "Caching and delivery via interference elimination," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1548–1560, 2018.

[7] C. Tian, H. Sun, and J. Chen, "A Shannon-theoretic approach to the storage-retrieval tradeoff in PIR systems," in *2018 Proceedings of IEEE International Symposium on Information Theory (ISIT)*, Jun. 2018, pp. 1904–1908.

[8] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 1st ed. New York: Wiley, 1991.

[9] R. Yeung, *A First Course in Information Theory*. New York: Kluwer Academic Publishers, 2002.

[10] R. W. Yeung, *Information theory and network coding*. Springer Science & Business Media, 2008.

[11] ——, "A framework for linear information inequalities," *IEEE Transactions on Information Theory*, vol. 43, no. 6, pp. 1924–1934, Nov. 1997.

[12] Z. Zhang and R. W. Yeung, "A non-Shannon-type conditional inequality of information quantities," *IEEE Transactions on Information Theory*, vol. 43, no. 6, pp. 1982–1986, Nov. 1997.

[13] F. Matus, "Infinitely many information inequalities," in *2007 IEEE International Symposium on Information Theory*. IEEE, 2007, pp. 41–44.

[14] R. Dougherty, C. Freiling, and K. Zeger, "Insufficiency of linear coding in network information flow," *IEEE Transactions on Information Theory*, vol. 51, no. 8, pp. 2745–2759, Aug. 2005.

[15] ——, "Networks, matroids, and non-shannon information inequalities," *IEEE Transactions on Information Theory*, vol. 53, no. 6, pp. 1949–1969, 2007.

[16] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Transactions on Information Theory*, 2010.

[17] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, "A survey on network codes for distributed storage," *Proceedings of the IEEE*, vol. 99, no. 3, March 2011.

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

[19] K. Zhang and C. Tian, "On the symmetry reduction of information inequalities," *IEEE Transactions on Communications*, vol. 66, no. 6, pp. 2396–2408, 2017.

[20] C. Lassez and J.-L. Lassez, "Quantifier elimination for conjunctions of linear constraints via a convex hull algorithm," in *Symbolic and numerical computation for artificial intelligence*, B. R. Donald, D. Kapur, and J. L. Mundy, Eds. San Diego, CA: Academic Press, 1992, ch. 4, pp. 103–1199.

[21] S.-W. Ho, C. W. Tan, and R. W. Yeung, "Proving and disproving information inequalities," in *2014 IEEE International Symposium on Information Theory*. IEEE, 2014, pp. 2814–2818.

[22] C. Li, S. Weber, and J. M. Walsh, "Multilevel diversity coding systems: Rate regions, codes, computation, & forbidden minors," *IEEE Transactions on Information Theory*, vol. 63, no. 1, pp. 230–251, 2016.

[23] *IBM ILOG CPLEX Optimizer*, IBM, 2009-2019. [Online]. Available: http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/

[24] *Gurobi Optimizer*, Gurobi, 2008-2019. [Online]. Available: http:/www.gurobi.com/

[25] *GAP-Groups, algorithms, programming-A system for computational discrete algebra.* [Online]. Available: http://www.gap-system.org/