Low Complexity Code Design for Lossless and Near-Lossless Side Information Source Codes*

Qian Zhao Michelle Effros

Abstract

We consider instantaneous side information source code (SISC) design. In the SISC configuration, the encoder describes source X to the decoder; the decoder uses this description and side information Y (which is not available at the encoder) to reconstruct X. Prior work on lossless and near-lossless SISC design demonstrates that globally optimal design is NP-hard. In this paper, we introduce a family of polynomial complexity code design algorithms that approximates the optimal solution for lossless and near-lossless SISCs. The algorithms may be used to design both Huffman and arithmetic SISCs for an arbitrary probability mass function $p(x, y)$. Experimental results comparing the resulting performances to each other and to the theoretical limit are included.

I Introduction

A side information source code (SISC) is a compression algorithm in which the decoder can use side information not available to the encoder. SISCs are useful, for example, in sensor networks, where one sensor can use its own sensory input in decoding the message from another sensor. An SISC comprises an encoder $\gamma : \mathcal{X} \to \{0,1\}^*$ and a decoder $\gamma^{-1} : \{0,1\}^* \times \mathcal{Y} \to \mathcal{X}$. Suppose that $\mathcal{X} \times \mathcal{Y}$ is finite and $(X, Y) \in \mathcal{X} \times \mathcal{Y}$ has joint, memoryless probability mass function (pmf) $p(x, y)$. Let (γ, γ^{-1}) be instantaneous; then γ^{-1} uses only the first $|\gamma(x_1)|$ bits of $\gamma(x_1)\gamma(x_2)\ldots$ to reconstruct $\hat{x}_1$. The probability of decoding error is $P_e = \Pr(\gamma^{-1}(\gamma(X), Y) \neq X)$. In lossless SISCs, $P_e \equiv 0$; in near-lossless SISCs, P_e is a tiny positive value.

Prior work on SISCs includes sub-optimal [1, 2, 3, 4, 5] and optimal [6, 7, 8] design algorithms. While the encoding and decoding complexities of optimal SISCs are low, optimal Huffman SISC design is NP-hard [9]. Optimal Huffman multiple access source code (MASC) design is also NP-hard by Lemma 1 in the Appendix. Optimal arithmetic SISC and MASC design seems to have similar complexity problems. We therefore turn our attention to low complexity code design. Building on [8], we investigate polynomial time algorithms for lossless and near-lossless SISC design.

*Q. Zhao (qianz@caltech.edu) and M. Effros (effros@caltech.edu) are with the Dept. of Electrical Engineering, MC 136-93, California Institute of Technology, Pasadena, CA 91125. This material is based upon work supported by NSF Award No. CCR-9909026 and CCR-0220039 and by Caltech's Lee Center for Advanced Networking.

Proceedings of the Data Compression Conference (DCC'03)
1068-0314/03 $17.00 © 2003 IEEE

Section II treats Alon and Orlitsky's elegant chromatic entropy bound on optimal SISC rates [2]; we show that the underlying code construction is NP-hard and therefore is not a viable alternative for sub-optimal code design. Section III introduces a constrained SISC design problem and its polynomial-time optimal solution. Section IV describes a family of search strategies for the unconstrained problem built from the design algorithm for the constrained problem. Section V contains experimental results comparing the achievable rates of different low complexity SISC design algorithms with the optimal solution. Section VI gives a brief summary of results.

II Obtaining Chromatic Entropy is NP-Hard

Since design of optimal lossless SISCs [9] and MASCs (Lemma 1 in the Appendix) is NP-hard, we consider sub-optimal alternatives. The design used to bound the optimal SISC rate in [2] seems an attractive alternative. We next consider its complexity.

Associate with each $p(x, y)$ on $\mathcal{X} \times \mathcal{Y}$ a graph $G = (\mathcal{X}, E_{\mathcal{X}})$. Distinct vertices $x, x' \in \mathcal{X}$ are connected if and only if $p(x, y)p(x', y) > 0$ for some $y \in \mathcal{Y}$. Code γ_X is valid if and only if for every edge $(x, x') \in E_{\mathcal{X}}$, $\{\gamma_X(x), \gamma_X(x')\}$ satisfies the prefix condition; a valid code is a lossless instantaneous SISC [8].

A legitimate coloring c colors the vertices of G so that no two connected vertices have the same color. The entropy of c is $H[c(X)] = -\sum_\beta P[c^{-1}(\beta)] \log P[c^{-1}(\beta)]$, where $c^{-1}(\beta)$ describes all symbols with color β and $P[A] = \sum_{(x,y)\in A\times\mathcal{Y}} p(x, y)$. The chromatic entropy is $H_\chi(G, P) = \min H[c(X)]$ (the minimum is taken over legitimate colorings). By [2, Theorem 2], the optimal rate $\bar{L}$ for an SISC on $p(x, y)$ satisfies

$$H_\chi(G, P) - \log[H_\chi(G, P) + 1] - \log e \leq \bar{L} \leq H_\chi(G, P) + 1.$$

We prove that calculating $H_\chi(G, P)$ is NP-hard by showing that the H_χ decision problem is NP-complete. The H_χ decision problem inputs graph $G(V, E)$ and marginal pmf P and outputs the answer to "Is $H_\chi(G, P) \leq \log_2 3$?" The proof of Theorem 1 appears in the Appendix.

Theorem 1 *H_χ is NP-complete.*

III Constrained SISC Design Algorithms

By [6, 7, 8], optimal SISC design is equivalent to optimal partition design followed by optimal matched code design. A partition $\mathcal{P}$ is a tree structure describing prefix relationships for an SISC's descriptions of all $x \in \mathcal{X}$. Thus x and x' sit at the same node in $\mathcal{P}$ if they have identical descriptions, and the node for x is an ancestor of the node for x' in $\mathcal{P}$ if the description of x is a proper prefix of that of x'. A matched code for $\mathcal{P}$ is an SISC with the prefix relationships described by $\mathcal{P}$. Optimal matched code design for $\mathcal{P}$ involves designing optimal entropy codes (e.g., Huffman or arithmetic codes) on the children of each internal node in $\mathcal{P}$. We use $R(\mathcal{P})$ to denote the expected rate achieved by the optimal matched code for $\mathcal{P}$. Optimal matched code design requires only polynomial complexity, so optimal partition design is NP-hard [8].

Partition $\mathcal{P}$ yields an instantaneous SISC if and only if $p(x,y)p(x',y) > 0$ for some $x, x' \in \mathcal{X}$ and $y \in \mathcal{Y}$ implies the description of x is not a proper prefix of the description of x'. The above instantaneous SISC is lossless if and only if $p(x,y)p(x',y) > 0$ for some $x, x' \in \mathcal{X}$ and $y \in \mathcal{Y}$ also implies that the descriptions of x and x' are not identical. Given a partition $\mathcal{P}$ satisfying the first property but not the second, the optimal decoder is the MAP decoder $\gamma^{-1}(\gamma(x), y) = \arg\max_{x' \in \mathcal{G}(x)} p(x', y)$, where $\mathcal{G}(x)$ is the node of $\mathcal{P}$ containing x. The resulting error probability is $P_e(\mathcal{P}) = \sum_{\mathcal{G} \in \mathcal{P}} P_e(\mathcal{G})$, where for any node $\mathcal{G} \in \mathcal{P}$, $P_e(\mathcal{G}) = \sum_{y \in \mathcal{Y}} \left[\sum_{x \in \mathcal{G}} p(x,y) - \max_{x \in \mathcal{G}} p(x,y)\right]$.

Given the high complexity of optimal partition design, we temporarily restrict our attention to defining and solving a constrained partition design problem.

Order alphabet $\mathcal{X}$ as $\mathcal{O} = \{x_1, \ldots, x_N\}$, where $N = |\mathcal{X}|$ and $i < j$ implies x_i precedes x_j in the chosen order. An *order-constrained partition for* $\mathcal{O}$ is any $\mathcal{P}$ such that a depth-first search [10] of $\mathcal{P}$ can describe $\mathcal{X}$ in order $\mathcal{O}$. Since partitions don't specify the order of symbols in a node or siblings in the tree and since a depth-first search doesn't preserve the structure of $\mathcal{P}$, many partitions can give the same order and many orders can come from the same partition. The optimal order-constrained partition for order $\mathcal{O}$ and constant $\lambda \geq 0$ is $\mathcal{P}(\mathcal{O}, \lambda) = \arg\min_{\mathcal{P}} J_\lambda(\mathcal{P})$, where $J_\lambda(\mathcal{P}) = R(\mathcal{P}) + \lambda P_e(\mathcal{P})$ and the minimum is taken over partitions with order $\mathcal{O}$. The λ-optimal order is $\mathcal{O}^\star(\lambda) = \arg\min_{\mathcal{O}} J_\lambda(\mathcal{P}(\mathcal{O}, \lambda))$. We control P_e by varying λ.

We focus on near-lossless SISC design in the following descriptions. Since lossless and near-lossless SISC design differ only in the prefix property, extension to lossless SISC design is straightforward.

Theorem 2 *The worst case complexity in constructing the optimal order-constrained partition and matched code for a given order* $\{x_1, \ldots, x_N\}$ *is* $O(N^4)$.

Proof: Fix λ, and let $\mathcal{G}[i,k] = \mathcal{P}(\{x_i, \ldots, x_k\}, \lambda)$, $r[i,k] = R(\mathcal{G}[i,k])$, and $e[i,k] = P_e(\mathcal{G}[i,k])$. We use dynamic programming to find $\mathcal{G}[i,k]$ for all $1 \leq k - i \leq N - i$. (Note that $\mathcal{G}[i,i] = (x_i)$ and $r[i,i] = e[i,i] = 0$ for all i.)

Let $r_m[i,j,k]$ and $e_m[i,j,k]$ denote the rate and error probability resulting from combining $\mathcal{G}[i,j]$ with $\mathcal{G}[j+1,k]$ into $\mathcal{G}[i,k]$ using a type-m combination, $m = 1, 2$. When $m = 1$, $\mathcal{G}[i,k]$ comprises an empty root with children $\mathcal{G}[i,j]$ and $\mathcal{G}[j+1,k]$; thus $e_1[i,j,k] = e[i,j] + e[j+1,k]$, and

$$r_1[i,j,k] = \begin{cases} r[i,j] + r[j+1,k] + P[i,k] & \text{in Huffman coding,} \\ r[i,j] + r[j+1,k] + P[i,k]H(P[i,j]/P[i,k]) & \text{in arithmetic coding.} \end{cases}$$

where $P[i,j] = \sum_{\ell=i}^{j} p_X(x_\ell)$ and $H(p) = -p\log p - (1-p)\log(1-p)$. When $m = 2$, we build $\mathcal{G}[i,k]$ by combining one of $\mathcal{G}[i,j]$ and $\mathcal{G}[j+1,k]$ with the root of the other. Let $\mathcal{R}(\mathcal{G})$ be the root of $\mathcal{G}$, $\mathcal{C}(\mathcal{G})$ be the children of $\mathcal{R}(\mathcal{G})$, and define $s[i,j,k]$ as

$$s[i,j,k] = \begin{cases} 0 & \text{if } r[i,j] = 0, r[j+1,k] = 0 \text{ and } (\mathcal{G}[i,j] \cup \mathcal{G}[j+1,k]) = \mathcal{R}(\mathcal{G}[i,k]) \\ 1 & \text{if } r[i,j] = 0, r[j+1,k] > 0 \text{ and } \mathcal{G}[i,j] \text{ can join } \mathcal{R}(\mathcal{G}[j+1,k]) \\ 2 & \text{if } r[i,j] > 0, r[j+1,k] = 0 \text{ and } \mathcal{G}[j+1,k] \text{ can join } \mathcal{R}(\mathcal{G}[i,j]) \\ 3 & \mathcal{G}[i,j] \text{ and } \mathcal{G}[j+1,k] \text{ can't be joined by type-2 combination,} \end{cases}$$

where $\mathcal{G}, \mathcal{G}'$ can be joined if the resulting partition is valid. Then

$$(r_2[i,j,k], \quad e_2[i,j,k])$$
$$= \begin{cases} (0, \quad P_e(\mathcal{G}[i,j] \cup \mathcal{G}[j+1,k])) & \text{if } s[i,j,k] = 0 \\ (r[j+1,k], \quad P_e(\mathcal{G}[i,j] \cup \mathcal{R}(\mathcal{G}[j+1,k])) + P_e(\mathcal{C}(\mathcal{G}[j+1,k]))) & \text{if } s[i,j,k] = 1 \\ (r[i,j], \quad P_e(\mathcal{R}(\mathcal{G}[i,j]) \cup \mathcal{G}[j+1,k]) + P_e(\mathcal{C}(\mathcal{G}[i,j]))) & \text{if } s[i,j,k] = 2 \\ (\infty, \quad \infty) & \text{if } s[i,j,k] = 3. \end{cases}$$

Let $m^\star = m^\star[i,j,k] = \arg\min_{m\in\{1,2\}}\{r_m[i,j,k] + \lambda e_m[i,j,k]\}$ be the optimizing method and $j^\star = j^\star[i,k] = \arg\min_{j\in\{i,i+1,\ldots,k-1\}}\{r_{m^\star}[i,j,k] + \lambda e_{m^\star}[i,j,k]\}$ the optimizing index. Then $r[i,k] = r_{m^\star}[i,j^\star,k]$, $e[i,k] = e_{m^\star}[i,j^\star,k]$, and $\mathcal{G}[i,k]$ is the group described by the type-$m^\star$ combination of $\mathcal{G}[i,j^\star]$ and $\mathcal{G}[j^\star+1,k]$.

When the procedure is complete, $\mathcal{G}[1,N]$ is the optimal order-constrained partition on order $\{x_1,\ldots,x_N\}$; $r[1,N]$ and $e[1,N]$ are its expected rate and error probability.

The number of operations required to calculate $r_m[i,j,k]$ and $e_m[i,j,k]$ dominates the complexity of this algorithm. In calculating $s[i,j,k]$, we need to check whether subsets of $\mathcal{G}[i,j]$ and $\mathcal{G}[j+1,k]$ can be joined, which requires at most $\min\{j-i+1, k-j\} \le (k-i)/2$ operations (previous join-ability results are stored). Thus the worst case complexity is $\sum_{i=1}^{N-1}\sum_{j=i}^{N-1}\sum_{k=j+1}^{N}(k-i)/2 = O(N^4)$. □

IV Fast SISC Design

Since order-constrained partition optimization requires only polynomial time, optimal order design is NP-hard. We tackle this combinatorial optimization problem using simulated annealing (SA) and iterative descent techniques. In the discussion that follows, we use $R(\mathcal{O}) = R(\mathcal{P}(\mathcal{O},\lambda))$ for some fixed $\lambda > 0$.

Simulated Annealing (SA)

SA [11, 12] attempts to find an optimal solution while avoiding local optima. Applying SA to optimal order design, gives the following algorithm.

1. Initialize order $\mathcal{O}$ and temperature T.
2. While the *outer loop stopping criterion* is not satisfied, do the following.
   a) While the *inner loop stopping criterion* is not satisfied, do the following.
      i) Choose a random neighbor $\mathcal{O}'$ of $\mathcal{O}$.
      ii) If $R(\mathcal{O}') \le R(\mathcal{O})$, set $\mathcal{O} = \mathcal{O}'$.
      iii) If $R(\mathcal{O}') > R(\mathcal{O})$, set $\mathcal{O} = \mathcal{O}'$ with probability $e^{-(R(\mathcal{O}')-R(\mathcal{O}))/T}$.
   b) Set $T = \rho T$ ($0 < \rho < 1$ to reduce the temperature).
3. Return the best order.

The choices of the initial order and temperature, neighbor definitions, stopping criteria and parameter ρ all affect the speed of convergence and final solution of the SA algorithm. We choose the initial order at random and set the initial temperature

T_o to make the initial distribution on orders uniform. We set $\rho \in [0.9, 0.99]$ and use the symmetric neighbor relation (switching the positions of two randomly chosen elements in $\mathcal{O}$) that achieves the best performance in our tests. We stop the inner loop if the rate has not decreased for L_{in} orders in this inner loop and stop the outer loop if the rate has not decreased for L_{out} outer loop iterations.

Descent Neighbor (DN)
Iterative descent algorithms are a degenerate special case of SA algorithms. In this case, we define an asymmetrical neighbor (*descent neighbor*) relationship that guarantees $R(\mathcal{O}') \leq R(\mathcal{O})$ for all neighbors $\mathcal{O}'$ of $\mathcal{O}$. We guarantee this property by defining the neighbors of $\mathcal{O}$ to be all orders $\mathcal{O}'$ for which $\mathcal{P}(\mathcal{O}, \lambda)$ (the optimal partition for $\mathcal{O}$) is a legitimate order-constrained partition on $\mathcal{O}'$. The basic DN algorithm is:

1. Choose an initial order $\mathcal{O}$ at random.
2. While no more than L_{DN} orders with identical rates have been chosen, do:
   i) Choose a random descent neighbor $\mathcal{O}'$ of $\mathcal{O}$.
   ii) If $R(\mathcal{O}') < R(\mathcal{O})$, set $\mathcal{O} = \mathcal{O}'$.
   iii) If $R(\mathcal{O}') = R(\mathcal{O})$, set $\mathcal{O} = \mathcal{O}'$ with probability p_a.
   iv) Set $p_a = \rho_a p_a$ $(0 \leq \rho_a \leq 1)$.
3. Return $R(\mathcal{O})$.

Given a complexity budget equivalent to testing C orders, we can combat local minimality problems by running the DN algorithm k times with k distinct randomly chosen initial orders. We stop the algorithm once C orders have been tested and output the best order observed over all of the experiments.

Mixing SA with DN
Various mixtures of SA with DN are also possible. For example:

1. **SA + DN**: Each time we reach step b) of SA, we run DN starting from $\mathcal{O}$. (Note: In the inner loop, we stop the SA chain if the rate has not decreased for L_{SA} orders in that SA chain and stop the DN chain if the rate has not decreased for L_{DN} orders in that DN chain. We stop the outer loop if the rate has not decreased over L_{out} outer loops.)
2. **DN-merged-in-SA**: Change step i) of SA to: choose a random symmetric neighbor of $\mathcal{O}$ with probability P_s, choose a random descent neighbor of $\mathcal{O}$ with probability $1 - P_s$.

V Experimental Results

We test both lossless and near-lossless SISCs' performance using the dynamic programming algorithm from Section IV. In this paper, we only present lossless SISC's results due to space constraint.

For lossless SISCs, we use the following sources. Let $G_{N,q} = (\mathcal{X}, E_{\mathcal{X}})$ be a random graph with N vertices; each pair of vertices is connected with probability q (independent of all other pairs). We choose a distribution $P[\cdot]$ on the underlying size-N alphabet by drawing $u_1, \ldots, u_N$ uniformly on $(0, 1)$ and then normalizing.[1]

We test the fast algorithms on $G_{N,q}$ and P using $N \in \{8, 16, 32, 64, 128, 256\}$ and $q \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$. We show the performance of our algorithm as a function of the number C of orders tested. Since the algorithm involves random order choices, we run each experiment K times. We measure the performance of the algorithm both by the fraction of trials in which the algorithm achieves the target rate (the globally optimal rate when N is small or the best rate from N^4 randomly chosen orders when N is large) and by the average (over trials) of the code's rate at the end of a trial.

In Figure 1, we present the experimental results as a function of C for $N \in \{16, 64\}$ and $q \in \{0.3, 0.7\}$ when a set of generally good parameters is used for each of the algorithms. Table 1 compares how close the fast algorithm's results come to the optimum. Table 2 gives the best rates the fast algorithms obtained for $N = 256$ and $q \in \{0.3, 0.5, 0.7\}$. Table 3 summarizes the parameters used in Figure 1 and Table 2.

From these figures and tables, we have the following observations:

1. SA is the most successful at avoiding local minima, but its rate of convergence is slow. When C is N^3, the average rate is close to the target rate, but the probability of hitting the target rate is still very low for $N \geq 16$.

2. The rate of DN decreases much faster than that of SA when the complexity is low. If the DN algorithm is run only once, it usually gets stuck with a local minimum. By running DN multiple times, we avoid this problem in all of our experiments and achieve a much better performance than SA. At $C = N^3$, multiple-run DN achieves performances very close to the optimum. For example, for $N = 16$, at $C = N^3$, the probability of hitting the optimal solution is at least 0.97 and the average rate differs from the optimal rate by at most 0.02%. Even at $C = N^2$ and $C = 2N^2$, the average rates differ from the optimal rate by at most 2% and 1% respectively.

3. SA+DN and DN-merged-in-SA perform slightly better than SA but worse than DN for $N = 16$ and $q = 0.3, 0.5$; they perform better than DN for $N = 16$ and $q = 0.7$. For $N = 64$ and $C \leq 2N^2$, DN-merged-in-SA achieves performance very close to that of DN and both are much better than SA + DN's; but as the complexity further increases, DN-merged-in-SA outperforms the others, especially in terms of the probability of hitting the target rate for $q = 0.5, 0.7$.

VI Summary

We treat lossless and near-lossless SISC design for arbitrary source pmf $p(x, y)$. We present several algorithms for low complexity sub-optimal design. In our experi-

[1] By [2], if pmfs p and p' satisfy (1) $p(x, y) = 0$ if and only if $p'(x, y) = 0$ and (2) $\sum_y p(x, y) = \sum_y p'(x, y)$ for all x, then the optimal lossless SISCs and expected performances for the two pmfs are identical. Therefore, we specify only $P[x]$ for all x and not $p(x, y)$ for all (x, y).

Table 1: Closeness to the optimal solution. $Ratio = R_{fast}/R_{opt}$, R_{fast} = average rate of the fast algorithm over K trials, $Prob.$ = probability of hitting the target rate.

$N = 16$		$q = 0.3$		$q = 0.5$		$q = 0.7$	
		Prob.	*Ratio*	*Prob.*	*Ratio*	*Prob.*	*Ratio*
SA	$C = N^2$	0.11	1.0701	0.02	1.0874	0.03	1.0488
	$C = 2N^2$	0.16	1.0455	0.03	1.0610	0.06	1.0311
	$C = N^3$	0.79	1.0013	0.30	1.0147	0.58	1.0038
DN	$C = N^2$	0.54	1.0063	0.20	1.0222	0.19	1.0094
	$C = 2N^2$	0.71	1.0022	0.36	1.0121	0.40	1.0041
	$C = N^3$	1	1	0.97	1.0002	0.99	1.000
SA + DN	$C = N^2$	0.31	1.0261	0.12	1.0433	0.19	1.0173
	$C = 2N^2$	0.57	1.0084	0.21	1.0248	0.46	1.0055
	$C = N^3$	0.98	1	0.83	1.0015	1	1
DN-merged-in-SA	$C = N^2$	0.27	1.0394	0.06	1.0460	0.31	1.0154
	$C = 2N^2$	0.37	1.0213	0.11	1.0351	0.53	1.0050
	$C = N^3$	0.98	1	0.67	1.0050	0.99	1

Table 2: Achievable rates for $N = 256$, complexity limit $C = N^2 = 65536$.

$N = 256$	q	SA	DN	SA + DN	DN-merged-in-SA
Huffman Rate	0.3	5.00656	4.43684	4.38234	4.30231
= 7.75968	0.5	5.57978	5.09944	5.06005	4.96012
	0.7	6.16025	5.61802	5.59836	5.53641

ments, the multiple-run DN, SA+DN, and DN-merged-in-SA algorithms yield good performance. The latter two are better suited for large values of N and C.

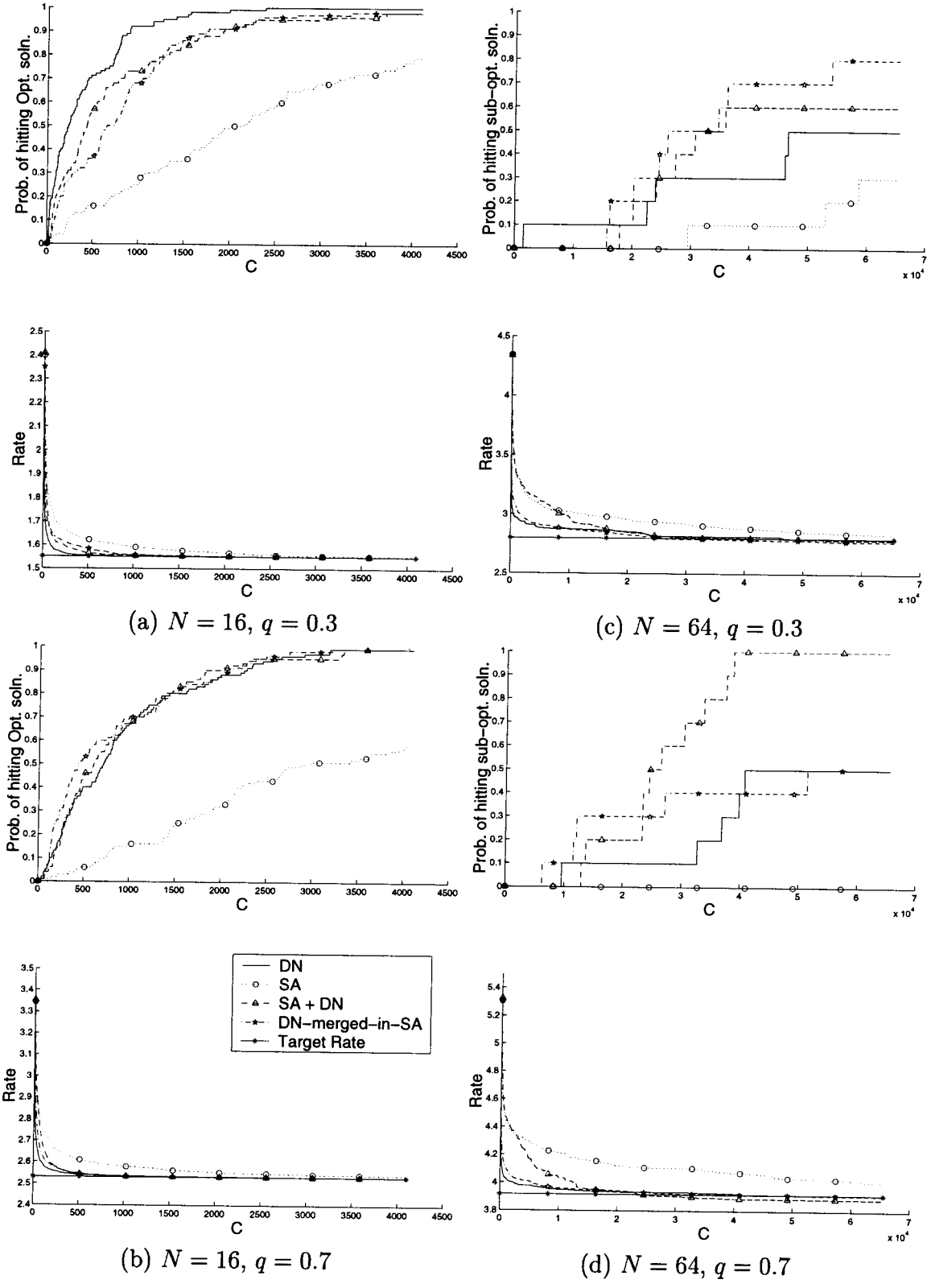
VII Appendix

Optimal MASC design is NP-hard

In an MASC, X and Y are independently described by two encoders; the decoder reconstructs X and Y from the pair of descriptions.

Lemma 1 *For a fixed partition $\mathcal{P}_Y$ on $\mathcal{Y}$, finding the partition $\mathcal{P}_X$ on $\mathcal{X}$ that optimizes the MASC given $\mathcal{P}_Y$ is NP-hard.*

Proof: Given a partition $\mathcal{P}_Y$ on $\mathcal{Y}$, partition $\mathcal{P}_X$ yields a lossless, instantaneous MASC if and only if for any $y, y' \in \mathcal{Y}$ such that $\gamma_Y(y)$ is a prefix of $\gamma_Y(y')$, the set $\{\gamma(x) : x \in \mathcal{X} \wedge (p(x,y) > 0 \vee p(x,y') > 0)\}$ is prefix-free [6, 8]. Thus given $\mathcal{P}_Y$, we connect vertices x, x' in $G(\mathcal{X}, E_{\mathcal{X}})$ if and only if $x, x' \in \{x \in \mathcal{X} : p(x,y) > 0$ or $p(x,y') > 0\}$ for some $y, y' \in \mathcal{Y}$ such that $\gamma_Y(y)$ is a prefix of $\gamma_Y(y')$. Thus optimal MASC design



(a) $N = 16$, $q = 0.3$

(c) $N = 64$, $q = 0.3$

(b) $N = 16$, $q = 0.7$

(d) $N = 64$, $q = 0.7$

Figure 1: Performance of four fast SISC design algorithms.

Table 3: Good parameters

	SA	DN	SA + DN	DN-merged-in-SA
$N = 16$ $(K = 100)$ $C = 4096$	$L_{in} = 4$ $L_{out} = 16$ $T_o = 0.01$	$L_{DN} = 16$ $p_a = 0$	$L_{SA} = 4$ $L_{DN} = 16$ $L_{out} = 4$ $T_o = 0.01$	$L_{in} = 4$ $L_{out} = 16$ $P_s = 0.7$ $T_o = 0.01$
$N = 64$ $(K = 10)$ $N = 256$ $(K = 1)$ $C = 65536$	$L_{in} = 256$ $L_{out} = 256$ $T_o = 0.01$	$L_{DN} = 1024$ $p_a = 0$	$L_{SA} = 256$ $L_{DN} = 1024$ $L_{out} = 256$ $T_o = 0.01$	$L_{in} = 256$ $L_{out} = 256$ $P_s = 0.7$ $T_o = 0.01$

given $\mathcal{P}_Y$ is the same as optimal SISC design for graph $G(\mathcal{X}, E_{\mathcal{X}})$, which is NP-hard. □

Proof of Theorem 1: H_χ is NP-Complete.

Proof: Given any coloring on G as a guess, we can always verify in polynomial time if this coloring has an entropy $\leq \log_2 3$. Thus $H_\chi \in NP$.

We next show that there exists a polynomial reduction of $3C$ to H_χ. (Here $3C$ denotes the problem "is G colorable with 3 colors?" and is NP-complete [10].) The input I of $3C$ is graph $G'(V', E')$; we give a polynomial algorithm to construct input $f(I)$ of H_χ from I; then we show that G' is 3-colorable if and only if $H_\chi(G, P) \leq \log_2 3$.

Construct $G(V, E)$ as: $V = V' \cup \{v_1, v_2, v_3\} = \{v'_1, \ldots, v'_M, v_1, v_2, v_3\}$ $(M = |V'|)$, $E = E' \cup \{(v_1, v_2), (v_1, v_3), (v_2, v_3)\}$, i.e. v_1, v_2, v_3 form a triangle. Construct P as: $P(v_1) = 1/3$, $P(v_2) = 1/3$, $P(v_3) = 1/3 - 1/M$, and $P(v'_j) = 1/M^2$ for $j = 1, \ldots, M$.

Assume G' is 3-colorable, then G is also 3-colorable. Thus $H_\chi(G, P) \leq \log_2 3$.

Next, assume G' is K-colorable with $K > 3$ but not 3-colorable. Then $M \geq 4$ and G is also K-colorable but not 3-colorable. Let $k_j, j = 1, 2, \ldots, K$ be the number of vertices in G' that are colored color c_j, and without loss of generality assume $k_1 \geq k_2 \geq \ldots \geq k_K$. Then a simple minimization gives:

$$H_\chi(G, P) = f\left(\frac{k_1}{M^2} + \frac{1}{3}\right) + f\left(\frac{k_2}{M^2} + \frac{1}{3}\right) + f\left(\frac{k_3}{M^2} + \frac{1}{3} - \frac{1}{M}\right) + \sum_{j=4}^{K} f\left(\frac{k_j}{M^2}\right),$$

where $f(p) = -p \log p$. The minimal value of $H_\chi(G, P)$ is achieved when the k_j's take their boundary values: $k_1 = M - (K - 1)$, $k_j = 1 (2 \leq j \leq K)$. Thus we have

$$\min_{(G,P)\ :\ K\text{-colorable}} H_\chi(G, P)$$
$$= f\left(\frac{M - K + 1}{M^2} + \frac{1}{3}\right) + f\left(\frac{1}{M^2} + \frac{1}{3}\right) + f\left(\frac{1}{M^2} + \frac{1}{3} - \frac{1}{M}\right) + (K - 3) f\left(\frac{1}{M^2}\right)$$

Since $(a + b)\log(a + b) \geq a \log a + b \log b$, we have

$$\min_{K \geq 4} \min_{(G,P)\ :\ K\text{-colorable}} H_\chi(G, P) = \min_{(G,P)\ :\ 4\text{-colorable}} H_\chi(G, P)$$

$$= f\left(\frac{M-3}{M^2}+\frac{1}{3}\right)+f\left(\frac{1}{M^2}+\frac{1}{3}\right)+f\left(\frac{1}{M^2}+\frac{1}{3}-\frac{1}{M}\right)+f\left(\frac{1}{M^2}\right) \stackrel{\text{def}}{=} H_4(M)$$

It can be shown that $\frac{\mathrm{d}^3 H_4(M)}{\mathrm{d}M^3} > 0$ and $\lim_{M\to\infty}\frac{\mathrm{d}^2 H_4(M)}{\mathrm{d}M^2} = 0$, thus $\frac{\mathrm{d}^2 H_4(M)}{\mathrm{d}M^2} < 0$. Since $\frac{\mathrm{d}H_4(M)}{\mathrm{d}M}|_{M=4} < 0$, we have $\frac{\mathrm{d}H_4(M)}{\mathrm{d}M} < 0$ for $M \geq 4$. It can be verified that $\lim_{M\to\infty} H_4(M) = \log_2 3$, giving $H_4(M) > \log_2 3$ for $M > 3$.

Thus G' is 3-colorable if and only if $H_\chi(G,P) \leq \log_2 3$. □

References

[1] H. S. Witsenhausen. The zero-error side information problem and chromatic numbers. *IEEE Transactions on Information Theory*, 22:592–593, 1976.

[2] N. Alon and A. Orlitsky. Source coding and graph entropies. *IEEE Transactions on Information Theory*, 42(5):1329–1339, September 1996.

[3] A. Kh. Al Jabri and S. Al-Issa. Zero-error codes for correlated information sources. In *Proceedings of Cryptography*, pages 17–22, Cirencester,UK, December 1997.

[4] S. S. Pradhan and K. Ramchandran. Distributed source coding using syndromes (DISCUS) design and construction. In *Proceedings of the Data Compression Conference*, pages 158–167, Snowbird, UT, March 1999.

[5] Y. Yan and T. Berger. On instantaneous codes for zero-error coding of two correlated sources. In *Proceedings of the IEEE International Symposium on Information Theory*, page 344, Sorrento, Italy, June 2000.

[6] Q. Zhao and M. Effros. Optimal code design for lossless and near-lossless source coding in multiple access networks. In *Proceedings of the Data Compression Conference*, pages 263–272, Snowbird, UT, March 2001. IEEE.

[7] Q. Zhao and M. Effros. Lossless source coding for multiple access networks. In *Proceedings of the IEEE International Symposium on Information Theory*, page 285, Washington, DC, June 2001.

[8] Q. Zhao and M. Effros. Lossless and near lossless source coding for multiple access networks. *IEEE Transactions on Information Theory*, January 2003.

[9] P. Koulgi, E. Tuncel, S. Regunathan, and K. Rose. Minimum redundancy zero-error source coding with side information. In *Proceedings of the IEEE International Symposium on Information Theory*, page 282, Washington DC, USA, June 2001.

[10] S. Even. *Graph Algorithms.* Computer Science Press, Potomac, Maryland, 1979.

[11] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, May 1983.

[12] J. Vaisey and A. Gersho. Simulated annealing and codebook design. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 1176–1179, April 1988.