Using Simulated Annealing to Design Good Codes

ABBAS A. EL GAMAL, SENIOR MEMBER, IEEE, LANE A. HEMACHANDRA, ITZHAK SHPERLING, AND VICTOR K. WEI, MEMBER, IEEE

Abstract—**Simulated annealing is a computational heuristic for obtaining approximate solutions to combinatorial optimization problems. It is used to construct good source codes, error-correcting codes, and spherical codes. For certain sets of parameters codes that are better than any other known in the literature are found.**

I. INTRODUCTION

ANNEAL, as defined in *Webster's New Collegiate Dictionary*, means: 1) to heat (as glass) in order to fix laid-on colors, and 2) to heat and then cool (as steel or glass) usually for softening and making less brittle.

The cooling stage of annealing is usually done slowly to reduce the number of defects in the crystal structure (as steel) and to minimize the potential energy stored in the molecular configuration (as glass). A fundamental question in statistical mechanics concerns whether the system solidifies in the limit of low temperature, and if it does, whether it forms a crystalline solid or a glass. Motivated by a subtlety in numerical integration, the Metropolis algorithm [1] has often been used to simulate numerically the annealing process to gain an understanding of the ground-state configuration.

A typical combinatorial optimization problem seeks the minimum of a given objective (or cost) function of many variables. The variables are subject to intertwining constraints, and they interact with each other in complicated ways not unlike the molecules in a physical system. By appropriately defining an effective temperature for the multivariable system and imitating the physical annealing process, researchers have sought to solve a diverse collection of problems, with varying degrees of success.

Kirkpatrick [2] first investigated the use of simulated annealing in connection with the physical design of computers. He found the technique useful for solving VLSI layout and partitioning problems. More recently, researchers have compared simulated annealing with other heuristic techniques for *NP*-complete problems [5]. At present, simulated annealing provides the best heuristic algorithm for the graph partitioning problem, but extensive fine tuning has failed to make it the method of choice for graph coloring or the traveling salesman problem [3]. Johnson *et al.* [4] have investigated the interplay among local search, neighborhood structure, and combinatorial optimization problems.

The successes of simulated annealing have resulted in a surge of interest in the method. Many evaluations of the technique and its application to diverse areas are in progress [6]–[8]. In this paper we use simulated annealing to construct good source codes, error-correcting codes, and spherical codes. We have rediscovered many optimal or near-optimal source codes. For several parameter sets we have found constant-weight error-correcting codes that are better than any previous such code. We have also obtained spherical codes better than those constructed by an *ad hoc* method analogous to apple peeling. If we conceptualize codewords as repelling molecules, then the analogy between a code with good distance properties and a molecule structure with low potential energy is apparent. This may account for our success here. A more sophisticated attempt to explain the success of simulated annealing in a diverse variety of problems can be found in [2].

The rest of the paper is organized into four sections. In Section II we describe the coding problems we investigate. In Section III we present our simulated annealing algorithm. In Section IV we summarize the results obtained from computer experiments with the algorithm, including a few record-breaking codes. In Section V we conclude the paper by remarking on the advantages and disadvantages of the algorithm.

A detailed explanation of the intuition behind the simulated annealing method can be found in [2] or in many recent papers. We shall not include one here.

II. THE DESIGN OF GOOD CODES

In a communication system (Fig. 1) the design of good codes is of fundamental importance. Source information, such as English text and audio and video data, is represented efficiently via a *source code* [10]. American Na-

Manuscript received October 9, 1985; revised January 1, 1986. This work was supported in part by the National Science Foundation under NSF Grant ECS-83-00988. This paper was presented in part at the Simulated Annealing Conference, Yorktown Heights, NY, 1984; at the 22nd Annual Allerton Conference on Communication, Control, and Computing, Monticello, IL, October 3–5, 1984; and at the International Symposium on Information Theory, Brighton, England, June 1985.

A. A. El Gamal and I. Shperling are with the Information Systems Laboratory, Department of Electrical Engineering, Stanford University, Stanford, CA 94305, USA.

L. A. Hemachandra was with Bell Communications Research, Morristown, NJ. He is now with the Department of Computer Science, Cornell University, Ithaca, NY 14853, USA.

V. K. Wei is with Bell Communications Research, 435 South Street, Morristown, NJ 07960, USA.

IEEE Log Number 8610106.

0018-9448/87/0100-0116$01.00 ©1987 IEEE

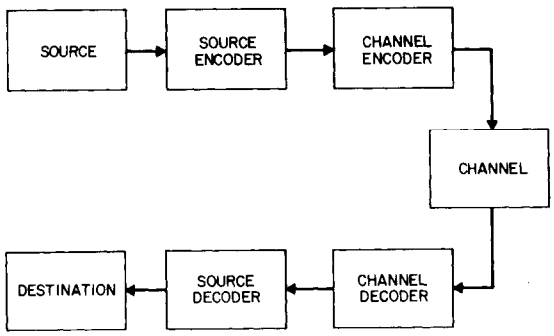

Fig. 1. Communication system.

tional Standard Code for Information Interchange (ASCII) codes, the Huffman code, Morse code, and various voice and picture encoding schemes fall into this category. The source-encoded information is then sent over a channel (transmission media), such as a telephone line, microwave link, or optical fiber, where the signals are often corrupted by noise. To ensure reliable transmission, the data are further encoded via a *channel code* (*error-correcting code*). This enables the detection of errors and their eradication. Prominent channel codes include Hamming codes, Reed–Solomon codes, convolutional codes, and trellis codes [11], [12], [26].

To be precise, the source code we study here is a set of M binary n-tuples

$$S = \{x_1, x_2, \cdots, x_M\} \subset \{0,1\}^n.$$

The rate of the code S is $R = (1/n)\log M$. (All logarithms in this paper are base two.) Let $B^n = \{0,1\}^n$ denote the set of all n-tuples. The *distortion* of a source code is the following quantity:

$$\delta(S) = \frac{1}{2^n} \sum_{y \in B^n} \min_{x \in S} d_H(x, y)$$

where the Hamming distance $d_H(x, y)$ between two binary n-tuples is the number of bit positions where they differ. A binary n-tuple is encoded to its nearest neighbor in S. The distortion is the average number of erroneous bits in encoding a random n-tuple. When S is used in a communication system, it can encode a random binary source with rate R and average distortion δ. A desirable property of a source code is a small distortion.

Given the length n and the size M, the construction of a source code with minimum distortion is a very difficult problem. The Lloyd–Max algorithm [25] can be used, but it often gets stuck in a local optimum and fails to find a minimum-distortion code. An exhaustive search algorithm would have to check all $\binom{2^n}{M}$ possible codes, a practical impossibility even for small values of n and M. For example, if $n = 7$ and $M = 16$, then about $\binom{128}{16} \approx 10^{20}$ codes would need to be checked. However, we shall see in the next section how simulated annealing can be used to find an optimum or near-optimum source code without doing an exhaustive search.

The channel code (error-correcting code) we study here is also a set of binary n-tuples

$$C = \{x_1, x_2, \cdots, x_M\}$$

with rate $R = (1/n)\log M$. Each member $x \in C$ is called a codeword. The minimum distance of C is

$$d_{\min} = \min_{\substack{x, y \in C \\ x \neq y}} d_H(x, y).$$

When the channel code is used in a communication system, up to $d_{\min}/2$ transmission errors can be eradicated. A desirable property of a channel code is a large minimum distance $d_{\min}$. The Hamming weight of a binary n-tuple x is the number of ones in x. In this paper we restrict ourselves to *constant-weight codes* where all codewords have the same Hamming weight. An important number in the theory of error-correcting codes is $A(n, d, w)$—the maximum size of a constant-weight code of length n, minimum distance d or more, and constant codeword weight w.

The constants $A(n, d, w)$ are extremely difficult to determine. Sophisticated mathematical theories, such as the Leech lattice [12], have been used to calculate these constants. Only a handful of values have been exactly determined [12], [13], notably $A(24, 8, 8) = 759$ and $A(24, 8, 12) = 2576$. For other values the gaps between upper and lower bounds are often sizable [12], [13]. One of the smallest undetermined values is $17 \leq A(23, 10, 7) \leq 23$. In the next section we shall see how simulated annealing is used to raise this lower bound to 18 and to help narrow other gaps.

Another channel code we study is the spherical code. A *spherical code* is a set of real vectors $\Phi = \{x_1, x_2, \cdots, x_m\}$ on the surface of a unit sphere in n-dimensional Euclidean space R^n. The minimum separating angle of a spherical code is

$$\theta_{\min} = \min_{\substack{x, y \in \Phi \\ x \neq y}} \cos^{-1}(x \cdot y)$$

where $x \cdot y$ is the inner (dot) product of x and y. Spherical codes have important applications to transmission over the Gaussian channel [15] and to many other areas [16], [17]. A desirable property of a spherical code is to have a large minimum separating angle. An important number is $M(n, \theta)$—the maximum number of vectors of a spherical code in n dimensions having a minimum separating angle greater than or equal to θ.

The case $\theta = \pi/3$ is particularly interesting. The number $M(n, \pi/3)$, called the *kissing number*, is the maximum number of nonoverlapping unit balls in n-dimensional space that can touch a given unit ball simultaneously. The determination of the kissing numbers has a long and fascinating history [18]–[20], culminating in the resolution of $M(8, \pi/3) = 240$ and $M(24, \pi/3) = 196\,560$. Other kissing numbers beyond three dimensions remain undecided.

Generally, $M(n, \theta)$ is very difficult to determine, save for a few trivial cases. In the next section we shall see how

simulated annealing can be used to produce interesting lower bounds for $M(n, \theta)$. These results may not improve upon existing lower bounds for particular values of n and θ. For example, simulated annealing has not yielded any improvement on existing lower bounds on the kissing numbers. However, the technique can be used to generate reasonable lower bounds for general values of n and θ rather quickly.

III. The Simulated Annealing Algorithm

We use the simulated annealing algorithm outlined in Fig. 2 to find good source codes, constant-weight codes, and spherical codes in our experiments. Our algorithm is similar to those simulated annealing algorithms used in other applications, such as those in [2].

```
CHOOSE CODE C, TEMPERATURE T
DO
   {
    DO
       {
       CHOOSE C', A PERTURBATION OF C
       LET ΔE=ENERGY(C')-ENERGY(C)
       IF ΔE<0 THEN C←C'
       ELSE WITH PROBABILITY EXP(-ΔE/T) C←C'
       }
   UNTIL (SEVERAL ENERGY DROPS OR TOO MANY ITERATIONS)
   LOWER TEMPERATURE
   }
UNTIL(STABLE CODE CONFIGURATION)
```

Fig. 2. Simulated annealing algorithm.

Initially, we choose a random code and a sufficiently high temperature. The outer loop decreases the temperatures until a series of stable code configurations has been seen; the inner loop perturbs the code until a prescribed number of energy drops occurs (not necessarily in a row), or too many iterations. After each perturbation, if the energy is decreased, we adopt the new code; if the energy is increased, we adopt the new code with probability $\exp(-\Delta E/T)$.

The temperature T is a control parameter of the algorithm. At high temperatures, $\exp(-\Delta E/T)$ is invariably close to one for positive ΔE. Therefore, we almost always adopt the new code. At low temperatures $\exp(-\Delta E/T)$ is close to zero for large ΔE, so we are not likely to adopt a new code which greatly increases the energy function. The initial temperature is set sufficiently high. For source codes we used $T_{\text{initial}} = 10 \cdot \text{length}$ in our experiments. For constant-weight codes and spherical codes we use $T_{\text{initial}} = 1000$. The energy function E is chosen to steer the code transfiguration in the general direction of reduced objective function. For source codes the energy function E is set to be the distortion $\delta(S)$, the objective function. For constant-weight codes we choose

$$E = \sum_{\substack{x, y \in C \\ x \neq y}} [d_H(x, y)]^{-k}$$

where k is a constant. As the simulated annealing process proceeds, the codewords have a tendency to repel each other, thus increasing the minimum distance. We do not choose the minimum distance as the energy function because it depends on only a pair of codewords; changes not involving any minimum-distance codeword pair would not be reflected in the energy function. The choice of an energy function different from the objective function is not an uncommon practice in simulated annealing [2]. We will comment on this later. We often use $k = 2$ in the computation. Other values of k and even other energy functions have been examined experimentally without noticeable improvement.

To perturb a code, we randomly select a codeword and randomly "jiggle" it, i.e., change one or two bits. The initial code has M codewords, each of which is an all-zero n-tuple (a very bad code indeed). We have tried other random initial codes with similar results because the all-zero initial code gets randomized very quickly at high temperatures.

The temperature is lowered in geometric series, i.e., $T \leftarrow \alpha T$. Typically, we choose the constant $\alpha > 0.9$. For source codes the temperature is lowered when the number of acceptances or rejections exceeds code size $\cdot$ $(1 + 0.1 \cdot$ code size$/T)$. The initial temperature is set at around $10 \cdot$ length. The algorithm terminates when the temperature drops below $0.01 \cdot$ length, or when five consecutive temperature stages produce no change in the code. For constant-weight codes and spherical codes we decrease the temperature when several (three to five) energy drops have occurred, or several hundred iterations. The initial temperature is chosen to be 1000, and we terminate the algorithm manually when we see a stable code configuration.

If we had chosen the minimum distance as the energy function, perturbations not involving any minimum-distance codeword pair would have resulted in no change in the energy function and hence would be adopted with probability one. This would have resembled a random walk. We feel that distance variations between every codeword pair should be reflected in the energy function. Experimental success seems to vindicate our decision. We have also tried the energy function $E = \Sigma f[d_H(x, y)]$, where $f(\cdot)$ is a concave function with a single minimum at the desired minimum distance. We attempted this energy function because in the cases where $A(n, d, w)$ is determined, the distance profile of the optimum codes tends to be a convex function with a single maximum at the desired distance. However, this energy function converged more slowly than our original choice in the experiments we have performed.

For the spherical code the simulation algorithm is essentially identical. The energy function is chosen to be

$$E = \sum_{x \neq y \in C} [\cos^{-1}(x \cdot y)]^{-k}.$$

Note that $\cos^{-1}(x \cdot y)$ is the separating angle between the two vectors. For each perturbation we jiggle a random codeword on the spherical surface.

TABLE I
SOURCE CODES VIA SIMULATED ANNEALING

Length n	Rate R	Distortion by Simulated Annealing	Hamming Lower Bound on Distortion	Number of Codes Checked
6	1/2	1/6	1/6	300
7	4/7	1/8	1/8	1782
10	0.3322	0.2371	0.2359	1674
10	0.5	0.16396	0.16250	3000

IV. RESULTS

Preliminary simulations have been performed on source codes, constant-weight codes, and spherical codes. The results are summarized in the following. Due to the nature of the simulated annealing algorithm, the codes in each parameter set are not exhaustively searched. Only a small portion of all possible codes are examined. We have also limited the CPU time spent in a simulation run to a few hours. The full power of the algorithm is not explored. The results obtained here demonstrate the potential power, not the ultimate limit, of the simulated annealing algorithm.

Source Codes

Simulations were performed on source codes with lengths ranging from six to ten and sizes varying from eight to 32. The resulting distortion is equal to or near the Hamming lower bound (Table I). The total number of iterations run ranges from 300 to 3000 for each set of parameters. Thus simulated annealing proves useful in finding good source codes, at least for the ranges of parameters tested.

For critical comparison we ran a greedy version of the algorithm in which we start with a low initial temperature. This modification has the effect of taking only the code transfigurations which reduce distortion. For $n = 7$, $m = 16$, and about 200 iterations we got stuck with a code having a distortion 10–20 percent above minimum in eight out of ten runs.

Fig. 3 illustrates the history of a single run of our algorithm which successfully finds a source code of length seven, rate 4/7, and distortion 1/8 which meets the Hamming lower bound. (This gives the result indicated in the second row of Table I.) The horizontal axis corresponds to the temperature, and the vertical axis corresponds to the distortion per bit, i.e., $\delta(C)$/length. The figure should be read from right to left because we proceed from high temperatures to low temperatures. Although we started out with the code consisting of 16 all-zero codewords at temperature 10 · length = 70, only the part of the history corresponding to temperatures below ten is shown. We stay at each temperature stage for a total of 10*(1 + 2*code size/T) iterations. The temperature is lowered in a geometric series by a factor of 0.9 at each reduction. The total number of iterations in this run is 3206. The per-bit distortions of the source codes found by the algorithm at each temperature stage is shown in circles. More codes may be found than different values of distortion because different codes may have identical distortion. The complete figure shows that the distortion has a decreasing tendency, resulting in the minimum at low-enough temperatures. Notice that, occasionally, a code with larger distortion than any code found at the previous (higher) temperature may occur. This is the particular characteristic of the simulated annealing algorithm that enables it to avoid the trap of a local minimum in its search for the global minimum.

The histories of other runs of our algorithm which have generated the remaining codes listed in Table I are not included here. The figures for all histories exhibit a decreasing tendency for the distortion as the temperature drops, resulting in minimum or near-minimum distortions.

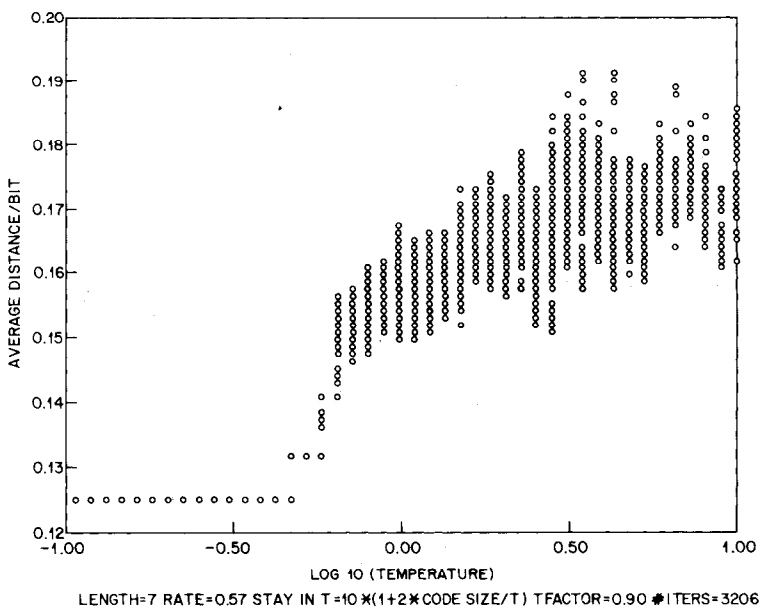

Fig. 3. History of simulated annealing run.

TABLE II
CONSTANT-WEIGHT CODES VIA SIMULATED ANNEALING

Length	Weight	Distance	Code Size	Upper Bound
21	9	10	22	41
22	9	10	23	57
23	7	10	18[a]	23
23	8	10	28[a]	50
23	9	10	24	87
23	10	10	39	117
23	11	10	39	135
24	8	10	33[a]	68
24	9	10	24	119
24	11	10	57	223
24	12	10	60	247

[a] Best-known result in the literature.

TABLE III
SIZE 18 CONSTANT-WEIGHT CODE FOR $A(23, 10, 7)$

1.	00000010010000011110001
2.	00000101011010010001000
3.	00001101000000100010101
4.	00010000101000111000100
5.	00010001000101010000011
6.	00011000000011100101000
7.	00101010101001000000001
8.	00110010000010000010110
9.	01000000110000000001111
10.	01001000001110001000010
11.	01010110010100100000000
12.	01100001000100000111000
13.	10000000110111000010000
14.	10000011100000100100010
15.	10001010000100010001100
16.	10111001010000001000000
17.	11000100000001001100100
18.	11100000000010110000001

TABLE IV
SIZE 28 CONSTANT-WEIGHT CODE FOR $A(23, 10, 8)$

1.	00000101000110100001110
2.	00100101100000010100011
3.	10000100110100010011000
4.	10001110000100001100010
5.	00001001101111010000000
6.	01100000101010100010100
7.	11000011000101100010000
8.	01001101010010000110000
9.	11001100100001000000101
10.	11010010100010000001010
11.	10011000000000110010110
12.	00100100011010001001001
13.	00010001011000010101100
14.	10000000001001000111011
15.	00000011100000001011101
16.	00101011010001000001010
17.	00010100100001101101000
18.	00110110000011010010000
19.	10100010001100010000101
20.	01000110010000111000100
21.	10000001111000101000010
22.	01010100011101000000010
23.	00111000110100001000100
24.	00011111001000100000001
25.	01101000000100110101000
26.	10010000010110100100001
27.	01000000000110011010011
28.	10100001000011001100100

TABLE V
SIZE 33 CONSTANT-WEIGHT CODE FOR $A(24, 10, 8)$

1.	000000000101100011101010
2.	000000100100111001010100
3.	000000110110000100101100
4.	000001000011111000001001
5.	000001010100010110000011
6.	000010001110011010100000
7.	000010101000000001100111
8.	000100001010010000011110
9.	000100110001011000100010
10.	000101010001000001010101
11.	001000000011001110000110
12.	001001101100001000001010
13.	001011000010000011011000
14.	001011011000100100000100
15.	001100101000010111000000
16.	001110000100100000010011
17.	010000000010010101110001
18.	010001110010100000010010
19.	010100001100001100000101
20.	010110010100010001001000
21.	010111001001000010000010
22.	011000001001101000110000
23.	100000101111000000010001
24.	100001100000001110110000
25.	100010001001001101001000
26.	100010010001110010010000
27.	100100100000100010001101
28.	100101000110100101000000
29.	101000010100001001100001
30.	110000011010000011000100
31.	110001000101010000100100
32.	111000000000110100001010
33.	111110100010000000100000

Constant Weight Codes

Using the simulated annealing algorithm, we have discovered many good constant-weight codes whose parameters are listed in Table II. They provide improvements to the previous results in Graham and Sloane [13]. During the course of this research, the authors became aware of the newer results of Conway and Sloane [14] which supersede our findings. After more computational effort we were able to find three codes that are better than those reported in [14]. They are a $(23, 10, 7)$ code with 18 codewords, a $(23, 10, 8)$ code with 28 codewords, and a $(24, 10, 8)$ code with 33 codewords. The numbers in the parentheses are the length, the minimum distance, and the codeword weight, respectively. The codewords of these codes are listed in Tables III–V. The upper bounds in Table II are taken from Graham and Sloane [13].

Spherical Codes

We have also found many good spherical codes. They provide lower bounds on the constants $M(n, \theta)$. Using a method analogous to apple peeling, we have constructed a class of spherical codes. The details of our construction are contained in the Appendix. The results of the apple-peeling construction and simulated annealing in three-dimensional space are compared in Table VI. Simulated annealing found better codes than the apple-peeling construction. One of these codes is listed in Table VII. It is possible that bounds exist rivaling or even superseding the apple-peeling bound in the literature. The comparison made in Table VI is preliminary.

The fourth column in Table VI contains the Wyner lower bound [27], which is similar to the one derived by Shannon [28] but slightly sharper. The bound is

$$M(n, \theta) \geq \frac{n\sqrt{\pi}\,\Gamma\left(\frac{n+1}{2}\right)}{(n-1)\Gamma\left(\frac{n+2}{2}\right)}\left[\int_0^{\theta} \sin^{n-2}\phi\, d\phi\right]^{-1}$$

where $\Gamma(\cdot)$ is the Gamma function. In three dimensions it

TABLE VI
SPHERICAL CODES VIA SIMULATED ANNEALING

Dimension	Angle $\cos\theta$	Angle θ	Lower Bound for $M(3,\theta)$ Wyner	Lattice	Apple Peel	Annealing	Upper Bound Rankin
3	~ 2/3	0.841	6		14	18	24
3	13/16	~ 0.62	10	12	30	31	42
3	5/6	~ 0.586	12	24	34	35	48
3		0.524	14		40	45	59
3	~ 9/10	0.45	20		56	59	80
3	13/14	~ 0.38	28	48	82	83	112
3		$\pi/9$	32		98	99	132

TABLE VII
SIZE 35 SPHERICAL CODE FOR $M(3, \arccos(5/6))$

0.	−0.183291	0.022164	0.982809
1.	0.100781	−0.959698	−0.262341
2.	0.790523	−0.462605	0.401335
3.	−0.810027	−0.387179	−0.440396
4.	0.765679	0.579538	0.279056
5.	−0.269579	0.954104	0.130435
6.	0.302644	−0.846476	0.438047
7.	0.623330	−0.776523	−0.092038
8.	−0.066041	0.897189	−0.436681
9.	−0.208099	0.471576	−0.856919
10.	−0.129810	−0.588486	0.798018
11.	−0.255299	0.666100	0.700809
12.	0.761180	0.103834	0.640175
13.	−0.973947	−0.001453	0.226773
14.	0.874298	−0.322242	−0.362993
15.	0.258973	0.858889	0.441863
16.	−0.815145	0.670210	−0.415228
17.	−0.803612	−0.577832	0.142539
18.	−0.233786	−0.571646	−0.786489
19.	−0.652256	−0.345887	0.674481
20.	0.840418	0.417846	−0.345113
21.	0.291185	0.399403	0.869304
22.	0.620219	0.023453	−0.784078
23.	0.397097	−0.587405	−0.705173
24.	0.995169	0.034284	0.092000
25.	−0.878789	0.259884	0.686809
26.	−0.924050	0.182779	−0.335743
27.	−0.775339	0.603816	0.185083
28.	−0.294027	−0.908184	0.297910
29.	0.378628	0.576793	−0.723845
30.	0.447649	0.881887	−0.147941
31.	0.384192	−0.297955	0.873853
32.	0.069441	−0.068884	−0.995205
33.	−0.555841	−0.007377	−0.831256
34.	−0.466423	−0.836125	−0.288695

reduces to

$$M(3,\theta) \geq \frac{2}{1-\cos\theta}.$$

The second lower bound (fifth column of Table VI) is obtained from well-known lattice structures as described in Sloane [19]. The upper bound (column eight of Table VI) is due to Rankin [29] as follows:

$$M(n,\theta) \leq \frac{\sqrt{\pi}\,\Gamma\left(\frac{n-1}{2}\right)\sin\psi\tan\psi}{2\Gamma\left(\frac{n}{2}\right)\int_0^{\psi}(\sin\phi)^{n-2}(\cos\phi-\cos\psi)\,d\phi}$$

where $\psi = \sin^{-1}[\sqrt{2}\sin(\theta/2)]$. This bound is tighter than the standard volume bound. In three dimensions it reduces to

$$M(3,\theta) \leq \frac{\sin\psi\tan\psi}{(1-\cos 2\psi)/4 - \cos\psi(1-\cos\psi)}.$$

Among the lower bounds included in Table VI, simulated annealing provided the best results.[1] The preliminary investigations in four dimensions also look promising. Simulated annealing shows that $M(4, \cos^{-1}(5/6)) \geq 97$, which betters the lattice bound of 96, and $M(4, \cos^{-1}(9/10)) \geq 145$, which betters the lattice bound of 144.

Implementation Issues

The simulation programs for constant-weight codes and spherical codes are implemented on a VAX 780 with a floating-point accelerator. In each application the *C* program consists of less than a thousand statements. Most of the statements are for bookkeeping, debugging, and input/output. The core of the program, which executes the annealing process, is less than two hundred statements long.

Each of the three constant-weight codes listed in Tables III–V are found after a few runs of the algorithm. The CPU time spent on a typical run ranges from 15 min for the code in Table III to a maximum of about 2 h for the codes in Tables IV and V. Unsuccessful runs are terminated manually after about 1–2 h. Once a code has been found, we have been able to reproduce the success in almost every repeat run using the same cooling rate, although the code found each time is different.

The simulation runs for three-dimensional spherical codes typically take less than 30 min. The test runs for higher dimensions are allowed to consume up to several hours of CPU time.

We have used the basic algorithm as shown in Fig. 2 throughout our experiments. No major speed-up modification has been attempted. If large-scale experiments are to be embarked upon, methods such as described in [21] can be incorporated to reduce the running time. Jiggling several codewords simultaneously in each iteration of the algorithm in Fig. 2 may also help.

V. CONCLUDING REMARK

Simulated annealing promises to be a useful tool in designing good codes. We have found new codes better than any code previously known in the literature for many

[1] During the writing of this paper, the excellent constructions of three-dimensional spherical codes for several specific minimum separating angles [30] have been brought to the authors' attention. It appears unlikely that the simulated annealing algorithm can better these codes. However, the algorithm remains useful in the automatic generation of codes with general parameters.

sets of parameters. The method of simulated annealing has already proven useful in many combinatorial optimization problems. Our experiment here indicates that simulated annealing is also useful in providing lower bounds to combinatorial constants, such as the coding theoretical constants $A(n, d, w)$ and $M(n, \theta)$. Improvement to other constants, such as the Ramsey numbers, may be obtainable. Several related source-coding problems, such as vector quantization [32] and discrete Markov source codes [10], are also subject to analysis by simulated annealing. Essentially, the same algorithm can be used with very little modification.

The algorithm itself seems robust with respect to modifications of the choice of energy function, perturbation pattern, and annealing schedule (consisting of the initial temperature, the temperature decrement, and the termination conditions) as long as they stay reasonable. Though some interest exists in the fine tuning of annealing schedules [9], we have adhered to one kind of schedule in most of our experiments. In our experience simulated annealing appears robust for reasonable changes in the schedule. Better results are usually produced by the devotion of more CPU time. An important issue in simulated annealing is the convergence of the algorithm. We have largely ignored this issue in our studies. However, in all our experiments the results are converging. For more details on this issue see [33] and [34]. The method is easily applicable to new areas, and the results may be exciting.

One disadvantage of the simulated annealing algorithm is its liberal consumption of CPU time. In our experiments the required computation time goes up very rapidly with increasing parameter values. The choice of simulated annealing for a given problem must be weighed against other computational methods. For certain parameter values the simulated annealing algorithm finds it hard to contend with tight code constructions which make ingenuous use of advanced mathematical structures such as those contained in [12], [19], and [30]. The usefulness of the algorithm lies in its ability to generate reasonably good results automatically. The algorithm is relatively simple to implement and to adapt to particular problem constraints. In places where computation power is abundant and where specific mathematical constructions are lacking or unsatisfactory, the simulated annealing algorithm can be used effectively to generate reasonably good results mechanically. The codes generated in this paper are all nonsystematic codes, which could limit their applicability in practical situations.

Acknowledgment

Fruitful discussions with N. J. A. Sloane and D. S. Johnson are heartily acknowledged.

Appendix

In this Appendix we prove the apple-peeling lower bound on the numbers $M(n, \theta)$. We use a method of code construction which is similar to apple peeling in three dimensions, thus the name. We also present another construction method based on certain lattices in multiple dimensions.

Theorem (The Apple-Peeling Bound): For $0 < \theta \le \pi$, $n \ge 3$, we have

$$M(n, \theta) \ge 2 \cdot \sum_{i=0}^{k} M\left(n-1, \arccos^*\left[\frac{\cos\theta - \sin^2((i+1/2)\theta)}{\cos^2((i+1/2)\theta)} \right] \right)$$

where $k = \lceil \pi/2\theta - 1/2 \rceil$ and

$$\arccos^*(x) = \begin{cases} \arccos(x), & -1 \le x < 1 \\ 2\pi, & x < -1. \end{cases}$$

Remark: In any dimension $n \ge 1$, $M(n, \pi + \epsilon) = 1$ for $0 < \epsilon \le \pi$. In particular, $M(n, 2\pi) = 1$.

Proof: Let $x = (\sin\alpha, z_1 \cos\alpha, z_2 \cos\alpha, \cdots, z_{n-1} \cos\alpha)$ and $y = (\sin\alpha, w_1 \cos\alpha, w_2 \cos\alpha, \cdots, w_{n-1} \cos\alpha)$ be two points on the surface of the unit sphere with identical first coordinates. The two points $z = (z_1, z_2, \cdots, z_{n-1})$ and $w = (w_1, w_2, \cdots, w_{n-1})$ are on the surface of the unit sphere in $n-1$ dimensions. The points x and y are separated by an angle $\cos^{-1}(x \cdot y) = \cos^{-1}(\sin^2\alpha + z \cdot w \cos^{22}\alpha)$, while z and w are separated by $\cos^{-1}(z \cdot w)$.

For each i and $\alpha = \pm(i + 1/2)\theta$ we have $M(n-1, \arccos^*(\cos\theta - \sin^2\alpha)/\cos^2\alpha))$ code points in n dimensions with first coordinates equal to $\cos\alpha$ and a minimum separating angle $\ge \theta$. Note that when $(\cos\theta - \sin^2\alpha)/\cos^2\alpha < -1$, we have precisely one point with a first coordinate equal to $\sin\alpha$ because $M(n-1, 2\pi) = 1$. Two code points x and y with different first coordinates have their inner product $x \cdot y \le \sin\alpha \sin\beta + \cos\alpha \cos\beta = \cos(\alpha - \beta) \le \cos\theta$. Hence $\cos^{-1}(x \cdot y) \ge \theta$. By summing over all legal values of i, we obtain the apple-peeling bound.

In two dimensions it is easy to show that $M(2, \theta) = \lceil 2\pi/\theta \rceil$ for $0 < \theta \le 2\pi$. This, together with the apple-peeling theorem, gives us the sixth column of Table VI. Note that when $\cos((i + 1/2)\theta) = 0$, then $\cos\theta - \sin^2((i + 1/2)\theta) < 0$, and we take $\arccos^*[(\cos\theta - \sin^2((i + 1/2)\theta))/\cos^2((i + 1/2)\theta)] = 2\pi$.

The apple-peeling construction provides large codes in many dimensions. Although lattices provide the best-known lower bounds for kissing numbers ($\theta = \pi/3$) and excellent lower bounds in eight and 24 dimensions [16], [19], [20], [21], [31], the apple-peeling bound works well in other cases [32]. An additional advantage of the apple-peeling bound is its ease of implementation in any dimension. Equivalent or superior bounds may exist in the literature. The studies made here are preliminary.

We have also investigated another construction method based on certain lattices in n-dimensional space. First, choose k real numbers $x_1, x_2, \cdots, x_k$ and k integers $n_1, n_2, \cdots, n_k$ which add up to n. The collection of points with n_i coordinates being x_i and with the n_i satisfying certain constraints form a spherical code. The minimum separating angle and the size of the code can be calculated for specific choices of the x_i and the constraints on the n_i. For example, the set of n-dimensional vectors with coordinates taken from $\{0, 1/\sqrt{m}, -1/\sqrt{m}\}$ and with exactly m nonzero coordinates forms a spherical code with $2^m\binom{n}{m}$ code points and a minimum separating angle $= \cos^{-1}(1 - (1/m))$. The lattice construction presented earlier can be used in combination with the apple-peeling bound to produce good lower bounds to the size of spherical codes.

REFERENCES

[1] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fasting computing machines," *J. Chem. Phys.*, vol. 21, June 1953.

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

[3] D. S. Johnson, L. McGeoch, C. Rodriguez, and C. Schevon, "Optimization by simulated annealing: An experimental evaluation," presented at the *Workshop on Statistical Physics in Engineering and Biology*, Apr. 1984.

[4] D. S. Johnson, C. H. Papadimitriou, and M. Yannakakis, "How easy is local search?" in *Proc. 26th Symp. Foundations of Computer Science*, Oct 1985, pp. 39–42.

[5] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: Freeman, 1979.

[6] F. Romeo, C. Sechen, and A. Sangiovanni-Vincentelli, "Simulated annealing research at Berkeley," in *Proc. IEEE Int. Conf. Computer Design*, 1984, pp. 652–657.

[7] M. P. Vecchi and S. Kirkpatrick, "Global wiring by simulated annealing," *IEEE Trans. Comput. Aided Des. Integrated Circuits Syst.*, vol. CAD-2, pp. 215–222, Oct. 1983.

[8] S. White, "Concepts of scale in simulated annealing," in *Proc. IEEE Int. Conf. Computer Design*, 1984, pp. 646–651.

[9] M. Lundy "Applications of the annealing algorithm to combinatorial problems in statistics," *Biometrika*, vol. 72, pp. 191–198, 1985.

[10] R. G. Gallager, *Information Theory and Reliable Communication*. New York: Wiley, 1968.

[11] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1983.

[12] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. New York: North-Holland, 1977.

[13] R. L. Graham and N. J. A. Sloane, "Lower bounds for constant weight codes," *IEEE Trans. Inform. Theory*, vol IT-26, pp. 31–43, Jan. 1980.

[14] J. H. Conway and N. J. A. Sloane, "Lexicographic codes: Error-correcting codes from game theory," to appear.

[15] I. F. Blake, "The Leech lattice as a code for the Gaussian channel," *Inform. Contr.*, vol. 19, pp. 66–74, 1971.

[16] A. Gersho, "Asymptotically optimal block quantization," *IEEE Trans. Inform. Theory*, vol. IT-25, pp. 373–380, 1979.

[17] L. Shepp, "Computerized tomography and nuclear magnetic resonance," *J. Comput. Assisted Tomog.*, vol. 4, pp. 94–107, Feb. 1980.

[18] N. J. A. Sloane, "Tables of sphere packings and spherical codes," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 327–338, May 1981.

[19] ——, "Recent bounds for codes, sphere packings, and related problems obtained by linear programming and other methods," to appear.

[20] H. S. M. Coxeter, "The problem of packing a number of equal nonoverlapping circles on a sphere," *Trans. N.Y. Acad. Sci.*, series II, vol. 24, pp. 320–331, Jan. 1962.

[21] J. W. Greene and K. J. Supowit, "Simulated annealing without rejected moves," in *Proc. IEEE Int. Conf. Computer Design*, 1984, pp. 658–664.

[22] J. L. Lutton and E. Bonomi, "An efficient non-deterministic heuristic for the minimum weighted perfect Euclidean matching problem: The Metropolis algorithm," preprint.

[23] A. El Gamal and I. Shperling, "Design of good codes via simulated annealing," presented at the Simulated Annealing Conf., Yorktown Heights, NY, 1984.

[24] L. A. Hemachandra and V. K. Wei, "Using simulated annealing to calculate combinatorial constants," in *Proc. 22nd Annual Allerton Conf. Communication, Control, and Computing*, Oct. 3–5, 1984, pp. 545–552.

[25] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 129–137, 1982.

[26] A. R. Calderbank, J. E. Mazo, and V. K. Wei, "Asymptotic upper bounds on the minimum distance of trellis codes," *IEEE Trans. Commun.*, vol. COM-33, pp. 305–309, Apr. 1985.

[27] A. D. Wyner, "Capabilities of bounded discrepancy decoding," *Bell Syst. Tech. J.*, vol. 44, pp. 1061–1122, July/Aug. 1965.

[28] C. E. Shannon, "Probability of error for optimal codes in a Gaussian channel," *Bell Syst. Tech. J.*, vol. 38, pp. 611–656, May 1959.

[29] R. A. Rankin, "The closest packing of spherical caps in n-dimensions," *Proc. Glasgow Math. Ass.*, vol. 2, pp. 139–144, 1955.

[30] T. Tarnai and Z. S. Gáspár, "Improved packing of equal circles on a sphere and rigidity of its graph," *Math. Proc. Cambridge Phil. Soc.*, vol. 93, part 2, pp. 191–218, Mar. 1983.

[31] L. Hemachandra and V. K. Wei, "A note on the size of spherical codes," Bell Commun. Res., Tech. Memo., 1984.

[32] A. Gersho, "Quantization," *IEEE Commun. Soc. Mag.*, vol. COM-15, pp. 16–29, Sept. 1977.

[33] S. Romeo and S. Sangiovanni-Vincentelli, "Probabilistic hill-climbing algorithms: Properties and applications," in *Proc. Chapel Hill Conf. VLSI*, 1985, pp. 393–418.

[34] B. Hajek, private communication.