Analysis on Rate-distortion Performance of Compressive Sensing for Binary Sparse Source

Feng Wu[2], Jingjing Fu[1], Zhouchen Lin[2], Bing Zeng[1]
1. Hong Kong University of Science and Technology, Hong Kong
2. Microsoft Research Asia, Beijing, China
Email:{fengwu, v-jifu , zhoulin}@microsoft.com, eezeng@ece.ust.hk

Abstract

This paper proposes to use a bipartite graph to represent compressive sensing (CS). The evolution of nodes and edges in the bipartite graph, which is equivalent to the decoding process of compressive sensing, is characterized by a set of differential equations. One of main contributions in this paper is that we derive the close-form formulation of the evolution in statistics, which enable us to more accurately analyze the performance of compressive sensing. Based on the formulation, the distortion of random sampling and the rate needed to code measurements are analyzed briefly. Finally, numerical experiments verify our formulation of the evolution and the rate-distortion curves of compressive sensing are drawn to be compared with entropy coding.

1. Introduction

The recent compressive sensing theory reveals that a sparse signal can be recovered by a small amount of measurements [1][2]. Considering a signal $x = \{x_1, x_2, \dots, x_N\} \in \mathbb{R}^N$ with $x = \psi u$ and u has only K non-zero elements, x is called as K-sparse with respect to transform ψ. Random measurements $y = \{y_1, y_2, \dots, y_M\} \in \mathbb{R}^M$ are generated by

$$y = \Phi x,\ \Phi \in \mathbb{R}^{M \times N}. \tag{1}$$

$\boldsymbol{\Phi}$ is a random basis. Obviously, it is an ill-posed problem recovering $\boldsymbol{x}$ from $\boldsymbol{y}$ because M is often much smaller than N. But, according to the compressive sensing theory, if $\boldsymbol{x}$ is K-sparse and the condition $M \sim 2Klog(N/K)$ is satisfied [3], the recovery can be achieved with probability close to one by solving the following convex optimization

$$\widehat{\boldsymbol{u}} = argmin\|\boldsymbol{u}'\|_p, \text{ subject to } \boldsymbol{y} = \boldsymbol{\Phi\psi u}'. \tag{2}$$

$\|\cdot\|_p$ is the p-th norm.

Because M is often much smaller than N, the recovery can be viewed as a sort of compression. It is natural to arise the question whether random measurements provide an efficient representation of sparse signal in an information-theoretic sense. Some papers [4]~[9] have analyzed the performance of compressive sensing from the Shannon's rate distortion theory. Both theoretical and experimental results show that encoding a sparse signal by scale quantization of random measurements results in a significant penalty [8][9].

In most of the above work, the optimization problem (2) is solved by $l1$-norm techniques [10][11]. Although they work well in practice, their performance is difficult to be analyzed quantitively. Thus, a solution to (2) is needed that is more suitable to performance analysis. Some studies exhibit that compressive sensing has a link with channel coding that provides such solution. If the random transform $\boldsymbol{\Phi}$ is set as one kind of channel codes, the recovery can be done like channel decoding. The work categorized into this line includes Reed-Solomon (RS) code [6], Low-Density-Parity-Check (LDPC) code [12], Sudocode [13] and expander graph [14][15].

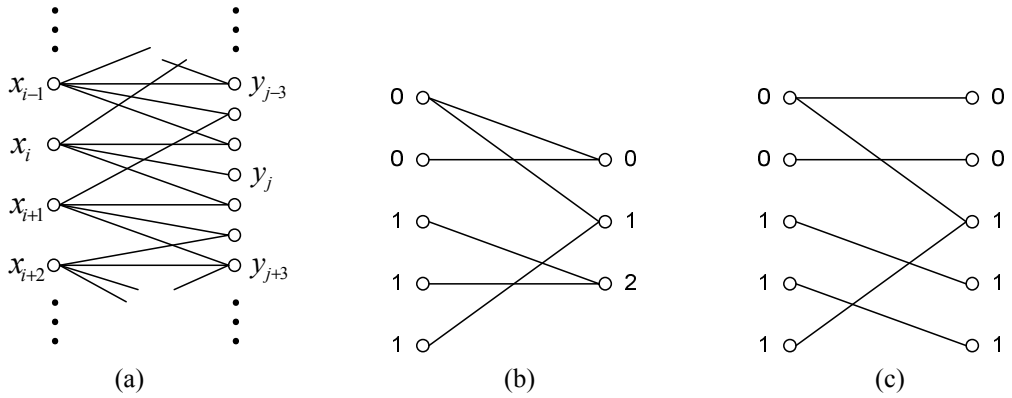
1068-0314/09 $25.00 © 2009 IEEE
DOI 10.1109/DCC.2009.24

This paper attempts to analyze compressive sensing for a binary random sparse source with independent and identical distribution. It is the most fundamental problem in information theory to evaluate the performance of one coding scheme. To achieve it, we have to get the distortion at a given number of measurements and the entropy of those measurements. However, both of them are very difficult to get in the $l1$-norm optimization. Therefore, we propose to use a bipartite graph to represent compressive sensing. The recovery is equal to the evolution of nodes and edges in the graph. They can be characterized by a set of differential equations. The most important contribution in this paper is that we derive the close-form formation of the evolution in statistics. Based on the formulation, we approximately analyze the distortion at a given number of measurements and the corresponding rate.

The rest of this paper is organized as follows. In Section 2, compressive sensing is converted to a bipartite graph. Section 3 derives the close-form formulation of the node and edge evolutions. Section 4 analyzes the rate-distortion performance of compressive sensing. In Section 5, the numerical results verify our formation and the performance of CS is compared with that of entropy coding. Finally, Section 6 concludes this paper.

2. Problem formulation

In this paper, we only consider $\boldsymbol{x}$ as a sparse binary source with independent and identical distribution P_x. In other words, the transform $\boldsymbol{\psi}$ is not necessary or it is the identity matrix. For a binary source, it is reasonable to assume the random matrix $\boldsymbol{\Phi}$ as binary too, namely, the element in $\boldsymbol{\Phi}$ is one or zero. We set the probability $P_x\{x_i = 1\} = p_w$ and $P_x\{x_i = 0\} = p_b$. Because $\boldsymbol{x}$ is a sparse source, p_w should be much smaller than p_b.



(a) (b) (c)

Figure 1: The bipartite graph generated by compressive sensing and the splitting process on the right side.

The random sampling in (1) to generate measurements constitutes a bipartite graph as depicted in Figure 1 (a). The left side is N nodes corresponding to binary elements in $\boldsymbol{x}$, and the right is M nodes to measurements in $\boldsymbol{y}$. Random sampling $\boldsymbol{\Phi}$ specifies the edges between left and right nodes. According to (1), the value of one measurement y_j is equal to the sum of binary elements x_i with ϕ_{ji} equal to one. For simplification, we assume one measurement always samples S elements in $\boldsymbol{x}$. Thus, the value of one measurement is from zero to S.

The degree of a node is defined as the number of edges connected to the node. The recovery from measurements is converted to the evolution of the bipartite graph. The process is equivalent to finding a node with degree one on the right side, and removing it, its connected left node and all edges connected to the left node. The process is repeated until there are no nodes of degree one on the right side. The recovery is successful if all nodes on the left side or all edges are removed.

The node of degree one on the right side is usually generated by only sampling one element in $\boldsymbol{x}$. However, such sampling is very inefficient. To recover $\boldsymbol{x}$, more measurements are needed if including single sampling. Because of the sparsity of $\boldsymbol{x}$, even if we sample multiple elements in $\boldsymbol{x}$, many of the measurement values may be zero. Therefore, a splitting process is proposed before the graph evolution. It is depicted in Figure 1 (b), where one measurement randomly samples two elements in $\boldsymbol{x}$. If the measurement value is zero or two, every sampled x_i is known. Thus the corresponding right node is split into two nodes of degree one. After this process, the graph will have many right nodes of degree one.

3. Recovery processing

This section will analyze the bipartite graph evolution and derive the close-form formulation of the evolution on left nodes and right edges.

3.1 Node evolution on the left side

The analysis on the left side is done by modeling the evolution of nodes. It is a random process along with time t_l. The unit of t_l is defined as the time needed for removing one node on the left side, namely, $\Delta t_l = 1/E_l$, where E_l is the total number of nodes that will be removed on the left side. According to the graph generated by compressive sensing, the expected total number of left nodes that are sampled is

$$E_l = N(1 - (1 - p_S)^M),\ E_l \leq N. \tag{3}$$

$p_S = S/N$ is the probability that one element in $\boldsymbol{x}$ is sampled by one measurement. The rest $N - E_l$ elements in $\boldsymbol{x}$ that are not sampled by any measurement will be discussed in the next section.

In the left side, the number of nodes of degree i at time t_l is defined as $L_i(t_l)$. Thus, the corresponding fraction of nodes of degree i is $l_i(t_l) = L_i(t_l)/E_l$. The fraction of left nodes remained at time t_l is $e_l(t_l) = \sum_i l_i(t_l)$. When t_l is equal to zero, the initial fraction λ_i of left nodes with degree i is

$$\lambda_i = N p_s^i (1 - p_s)^{M-i} \binom{M}{i} / E_l,\ i = 1,2, \dots, M. \tag{4}$$

The maximum degree is equal to M, that is to say, one node is sampled by all measurements.

When one right node of degree one is randomly selected, the probability that one left node of degree i is connected to the right node is $l_i(t_l)/e_l(t_l)$. The node is then removed and the corresponding fraction of node of degree i decreases. According to the martingale and large deviation in probability theory [16], the random process can be formulated by a set of differential equations as

$$dl_i(t_l)/dt_l = -l_i(t_l)/e_l(t_l),\ i = 1,2, \dots, M. \tag{5}$$

For different i, $dl_i(t_l)/l_i(t_l)$ should be equivalent from (5). If we set $dl_i(t_l)/l_i(t_l)$ for all i equal to df/f that f is a function, (5) can be rewritten as

$$dl_i(t_l)/l_i(t_l) = df/f = -dt_l/e_l(t_l). \tag{6}$$

The first equation of (6) can be solved directly. Thus, $l_i(t_l)$ can be represented by f as

$$l_i(t_l) = c_i f. \tag{7}$$

c_i is a parameter to be determined by boundary conditions given in (4). Because $e_l(t_l)$ is the sum of all $l_i(t_l)$, combining (7) with the second equation of (6), we can solve f as

$$f = -t_l / \sum_i c_i + c. \tag{8}$$

c is another parameter to be determined by boundary conditions. Combining (8) and (4) with (7), $l_i(t_l)$ can be finally described by

$$l_i(t_l) = \lambda_i(1 - t_l). \tag{9}$$

It is not difficult to explain the close-form formulation given in (9). The fraction of nodes of degree i is decreasing linearly along with t_l because the right node of degree one randomly connects to the left node of degree i. Furthermore, $e_l(t_l)$ is a linear function of t_l too because one left node is removed in each unit time.

3.2 Edge evolution in the right side

Different from the analysis in the left side, the analysis in the right side is to model the evolution process of edges instead of nodes. The unit of time t_r is defined as the time to remove an edge, namely, $\Delta t_r = 1/E$, where E is the total number of edges in the graph. According to the graph generation, the total number of edges in the graph is $E = SM$.

The edge of degree is defined as the degree of right node that it connects to. We denote the number of edges of degree j at time t_r as $R_j(t_r)$. The corresponding fraction of edges of degree j is $r_j(t_r) = R_j(t_r)/E$. The fraction of edges remained at time t_r is $e(t_r) = \sum_j r_j(t_r)$. Following up the result in left side, $e(t_r)$ should be equal to $1 - t_r$ because one edge is removed in each unit time. After the splitting process, the initial fraction ρ_j of edges with degree j is

$$\rho_j = \begin{cases} p_w^S + p_b^S & j = 1 \\ 0 & j = 2, \dots, S-1. \\ 1 - p_w^S - p_b^S & j = S \end{cases} \tag{10}$$

It is non-zero only when j is equal to one or S. When a left node is removed, the expected number of edges connected to the left node is $\sum_i i l_i(t_l)/e_l(t_l)$. Considering $l_i(t_l)$ given in (9), the expected number of edges removed subsequently is

$$\frac{\sum_i i l_i(t_l)}{e_l(t_l)} = \frac{\sum_i i\lambda_i}{\sum_i \lambda_i} = \sum_i i\lambda_i \coloneqq \alpha. \tag{11}$$

Interestingly, α is a constant independent on time and is determined only by the distribution of λ_i. It is worthy to mention that it will take $\alpha \Delta t_r$ to remove the expected number of edges.

There is one edge of degree one in α edges. We assume that the rest $\alpha - 1$ edges are randomly connected to right nodes. The probability that one edge is connected to one right node of degree j is $r_j(t_r)/e(t_r)$. Once this edge is removed, the degree of rest edges $j - 1$ in the right node becomes $j - 1$. The evolution process of edge in the right side can be formulated as a set of differential equations too

$$\frac{dr_j(t_r)}{dt_r} = \begin{cases} \frac{(\alpha-1)\left(r_2(t_r) - r_1(t_r)\right)}{\alpha e(t_r)} - \frac{1}{\alpha}, & j = 1 \\ \frac{j(\alpha-1)\left(r_{j+1}(t_r) - r_j(t_r)\right)}{\alpha e(t_r)}, & j = 2, \dots, S-1 \cdot \\ -\frac{j(\alpha-1) r_S(t_r)}{\alpha e(t_r)}, & j = S \end{cases} \tag{12}$$

When j is less than S, two kinds of edge removing will change $r_j(t_r)$. The first is, if a edge is removed in one right node of degree $j + 1$, the rest edges j of degree $j + 1$ become the edges of degree j. The second is, if a edge is removed in one right node of degree j, the rest edges $j - 1$ of degree j become the edges of degree $j - 1$. When j is equal to S and one edge is re-

moved, the rest edges of degree S become the edges of degree $S-1$. We first try to solve the differential equations (12) when j is more than one. They can be written as the matrix form

$$\begin{bmatrix} dr_2(t_r) \\ dr_3(t_r) \\ \dots \\ dr_S(t_r) \end{bmatrix} = \boldsymbol{A} \begin{bmatrix} r_2(t_r) \\ r_3(t_r) \\ \dots \\ r_S(t_r) \end{bmatrix} \frac{(\alpha-1)dt_r}{\alpha e(t_r)}, \text{ where } \boldsymbol{A} = \begin{bmatrix} -2 & 2 & 0 & \dots & 0 \\ 0 & -3 & 3 & \dots & 0 \\ \dots & \dots & \dots & \ddots & \dots \\ 0 & 0 & 0 & \dots & -S \end{bmatrix}. \tag{13}$$

Taking $\boldsymbol{A} = \boldsymbol{P^{-1}\Lambda P}$ presented in Appendix into account, multiplying $\boldsymbol{P}$ on the both side of (13), it can be described as

$$\boldsymbol{P} \begin{bmatrix} dr_2(t_r) \\ dr_3(t_r) \\ \dots \\ dr_S(t_r) \end{bmatrix} = \boldsymbol{\Lambda P} \begin{bmatrix} r_2(t_r) \\ r_3(t_r) \\ \dots \\ r_S(t_r) \end{bmatrix} \frac{(\alpha-1)dt_r}{\alpha e(t_r)}. \tag{14}$$

If $\tilde{\boldsymbol{r}} = [\tilde{r}_2(t_r), \dots, \tilde{r}_S(t_r)]^T$ is defined as $\boldsymbol{Pr} = \boldsymbol{P}[r_2(t_r), \dots, r_S(t_r)]^T$, then we have

$$\begin{bmatrix} d\tilde{r}_2(t_r) \\ d\tilde{r}_2(t_r) \\ \dots \\ d\tilde{r}_S(t_r) \end{bmatrix} = \boldsymbol{\Lambda} \begin{bmatrix} \tilde{r}_2(t_r) \\ \tilde{r}_2(t_r) \\ \dots \\ \tilde{r}_S(t_r) \end{bmatrix} \frac{(\alpha-1)dt_r}{\alpha e(t_r)}. \tag{15}$$

Because of $e(t_r) = 1 - t_r$ and (15), we have

$$d\tilde{r}_j(t_r)/\tilde{r}_j(t_r) = -j(\alpha-1)dt_r/\alpha(1-t_r),\ j = 2, \dots, S. \tag{16}$$

Considering the boundary conditions given in (10), the differential equations (16) can be readily solved

$$\tilde{r}_j(t_r) = \tilde{\rho}_j(1-t_r)^{j(\alpha-1)/\alpha},\ j = 2, \dots, S. \tag{17}$$

$\tilde{\rho}_j$ is the initial value of $\tilde{r}_j$ at time zero. Because $\boldsymbol{P}$ is an upper triangle matrix and $\tilde{\boldsymbol{r}} = \boldsymbol{Pr}$, $\tilde{\rho}_j$ can be is calculated by

$$\tilde{\rho}_j = \sum_{l=j}^{S} p_{jl}\rho_l,\ j = 2, \dots, S. \tag{18}$$

Similarly, because $\boldsymbol{P^{-1}}$ is a upper triangle matrix too and $\boldsymbol{r} = \boldsymbol{P^{-1}}\tilde{\boldsymbol{r}}$, the final $r_j(t_r)$ is described by

$$r_j(t_r) = \sum_{k=j}^{S}\left(p_{jk}^{-1}\left(\sum_{l=k}^{S} p_{kl}\rho_l\right)(1-t_r)^{k(\alpha-1)/\alpha}\right),\ j = 2, \dots, S. \tag{19}$$

p_{jk} and p_{jk}^{-1} are the elements of row j and column k in $\boldsymbol{P}$ and $\boldsymbol{P}^{-1}$, respectively. ρ_l is equal to zero except for one and S. By defining $\omega = (1-t_r)^{(\alpha-1)/\alpha}$, we have the final simplified $r_j(t_r)$ as

$$r_j(t_r) = \rho_S \binom{S-1}{S-j} \omega^j (1-\omega)^{S-j},\ j = 2, \dots, S. \tag{20}$$

$r_1(t_r)$ is difficult to get the close-form formulation from (12) even $r_2(t_r)$ is available in (20). Fortunately, because one edge is removed in every unit time, we have

$$\sum_{j=1}^{S} dr_j(t_r)/dt_r = -1, j = 1, \dots, S. \tag{21}$$

Combining (12) and (21), we can get the following equation

$$\frac{\alpha-1}{\alpha(1-t_r)}\sum_{j=1}^{S} r_j(t_r) + \frac{1}{\alpha} = 1, j = 2, \dots, S. \tag{22}$$

Incorporating (20) into (22), the final $r_1(t_r)$ is described as

$$r_1(t_r) = (1-t_r) - \rho_S\omega(1-(1-\omega)^{S-1}). \tag{23}$$

The bipartite graph and differential equations are also applied to analyze the performance of erasure correcting codes in [17]. To the authors' best knowledge, this paper is the first one that gives the close-form results as (20) and (23). In [17], the distributions of λ_i and ρ_j are vital to successfully decode because they have to make the number of right nodes of degree one more than zero before t_r is equal to 1. But in the compressive sensing, although every measurement samples same number of elements in $\boldsymbol{x}$, the right nodes of degree one can be guaranteed by the inherent sparseness of $\boldsymbol{x}$. From (23), $r_1(t_r)$ is decided by S, p_w and M, which becomes much simple. The detail conditions to make $r_1(t_r)$ will be discussed in our future paper.

4. Rate-distortion performance

In the bipartite graph, random edges can be generated by random seeds. Although only degrees of measurements are used in the evolution, all values of measurements are needed to recover the value of every element in $\boldsymbol{x}$. Thus, the rate of measurements is calculated by

$$R = -M \sum_{i=0}^{S} p_y(i) \log\left(p_y(i)\right), \tag{24}$$

$$\text{with } p_y(i) = \binom{S}{i} p_w^i (1-p_w)^{S-i}.$$

$p_y(i)$ is the probability of one measurement equal to i. For constant $M \times S$, the rate R will decrease with S increasing. If the distortion is measured by the number of unrecovered element in $\boldsymbol{x}$, the total distortion of compressive sensing can be calculated by

$$D = D_1 + D_2, \text{ with } D_1 = N(1 - S/N)^M. \tag{25}$$

D_1 is the distortion caused by un-sampled elements in $\boldsymbol{x}$. For given p_w and N, D_1 is decided by the total edges $M \times S$. D_2 is the distortion caused by elements that are sampled but cannot be recovered in the graph. The fail recovery is caused by two reasons. The first one is that $r_1(t_r)$ is equal to zero before $t_r = 1$. From our close-form result (23), when the signal is sparser and S is not big, the condition can be readily satisfied. Thus in this paper we assume $r_1(t_r) > 0$ when $t_r \in [0,1]$. The second reason is that there exist fault structures in the graph, which cannot provide enough information to recover the left nodes in the structures.

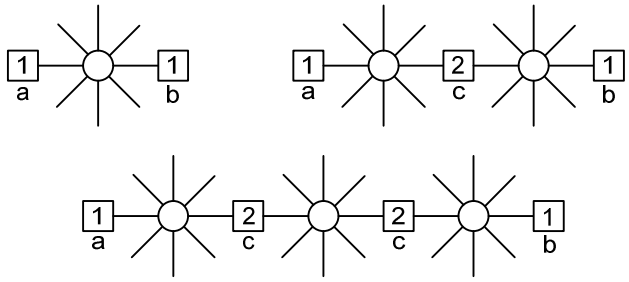


Figure 2: The exemplified sub-graphs that make left node unrecovered.

After our careful studies on un-recovered left nodes, we have found that the fault structures present a common feature as shown in Figure 2, where the squares indicate left nodes and the circles right nodes. The number in the square is the degree of left node. The lines in the circle are the edges to connect left nodes. The common feature in fault structures is that the structure always starts from one left node of degree one indicated by "a" in Figure 2 and ends at one left node of degree one indicated by "b". In general, the probability of left node "a" equal to left node "b" is very small. Thus, unlike the cycle problem in LDPC, the fault structures in our graph are open and line-type.

It is not difficult to explain why the left nodes in these structures cannot be recovered. Let us take the left-upper case in Figure 2 as an example. In the structure, there are two left nodes of degree one and one right node. No matter what other left nodes connected to the right node, it is impossible to recover two bits by only one bit. Most of un-recovered left nodes are on the structures depicted in Figure 2. Although the nodes "c" can be those of degree more than two, the probability is quite small when S is not big. Because of the limitation on pages, by only taking three structures in Figure 2 into account, we will approximate D_2 as

$$D_2 = N(p_1 + p_2) + 2(Mp_3 + Np_2)\frac{\binom{S-1}{1}\alpha_1(1-\alpha_1)^{S-2}}{(1-(1-\alpha_1)^{S-1})} + 2Mp_3. \quad (26)$$

p_1, p_2 and p_3 are probability of the three structures and α_1 is equal to $\lambda_1/\sum_i i\lambda_i$.

5. Numerical results

The first numerical experiment is designed to verify the close-form formulation of the edge evolution in the bipartite graph. The probability p_w is set as 0.05. There are 10000 nodes in the left side of the graph and 5000 nodes in the right side. Each measurement samples six left nodes. We randomly generate 5 different graphs. The evolutions of all $r_j(t_r)$ are depicted in Figure 2, where x coordinate is time and y coordinate is the number of left nodes at different degrees. The symbol "A" indicates the analyzed results and "R" the actual evolutions. One can observe that the analyzed evolutions have a good match to the actual ones. It is worthy to notice that the evolution of $r_2(t_r)$ cannot often be reduced to zero, which results in the distortion D_2.

The second numerical experiment is designed to verify the formulation of rate and distortion in (24) ~ (26). We use the same graph parameters as those in the first numerical experiment. The experimental curves are depicted in Figure 4, where x coordinate is the bits for coding measurements and y coordinate is the number of unrecovered elements. S is six in left side and twelve in right side. The rate and distortion vary with different number of measurement. For each number of measurements, we decode 20 different random graphs. The corresponding pairs of rate and distortion are drawn as points in Figure 4. The rate and distortion calculated by (24) ~ (26) are drawn as curves. One can observe that the analyzed results have a good match with actual ones when S is six. There is a bit mismatch at high rates when S is twelve because of our approximation in D_2.

Finally, we will compare CS and entropy coding in term of rate-distortion performance. In entropy coding, some tail elements in $\boldsymbol{x}$ may be dropped so as to meet the bit budget. It makes entropy coding to have the similar distortion behavior as CS. The results of CS are calculated from (24) ~ (26). Two different p_w are adopted in this experiment. For each S, we calculate rate and distortion at different numbers of measurements. The rate distortion curves are depicted in Figure 5, where x coordinate is rate and y coordinate is distortion. One can observe, when the distortion is big, CS has the similar performance as entropy coding. The performance gap between CS and entropy coding is increasing with rate increasing. One can also observe, when S is increased, the performance gap can be reduced. One interesting question is arising here if the performance of CS can approach to that of entropy coding when S is larger than a threshold. If it likes as the conclusion drawn by [8][9], what is the theoretic performance gap between CS and entropy coding? Since we do not derive the accurate formulation of D_2 in this paper, this question cannot be answered yet.

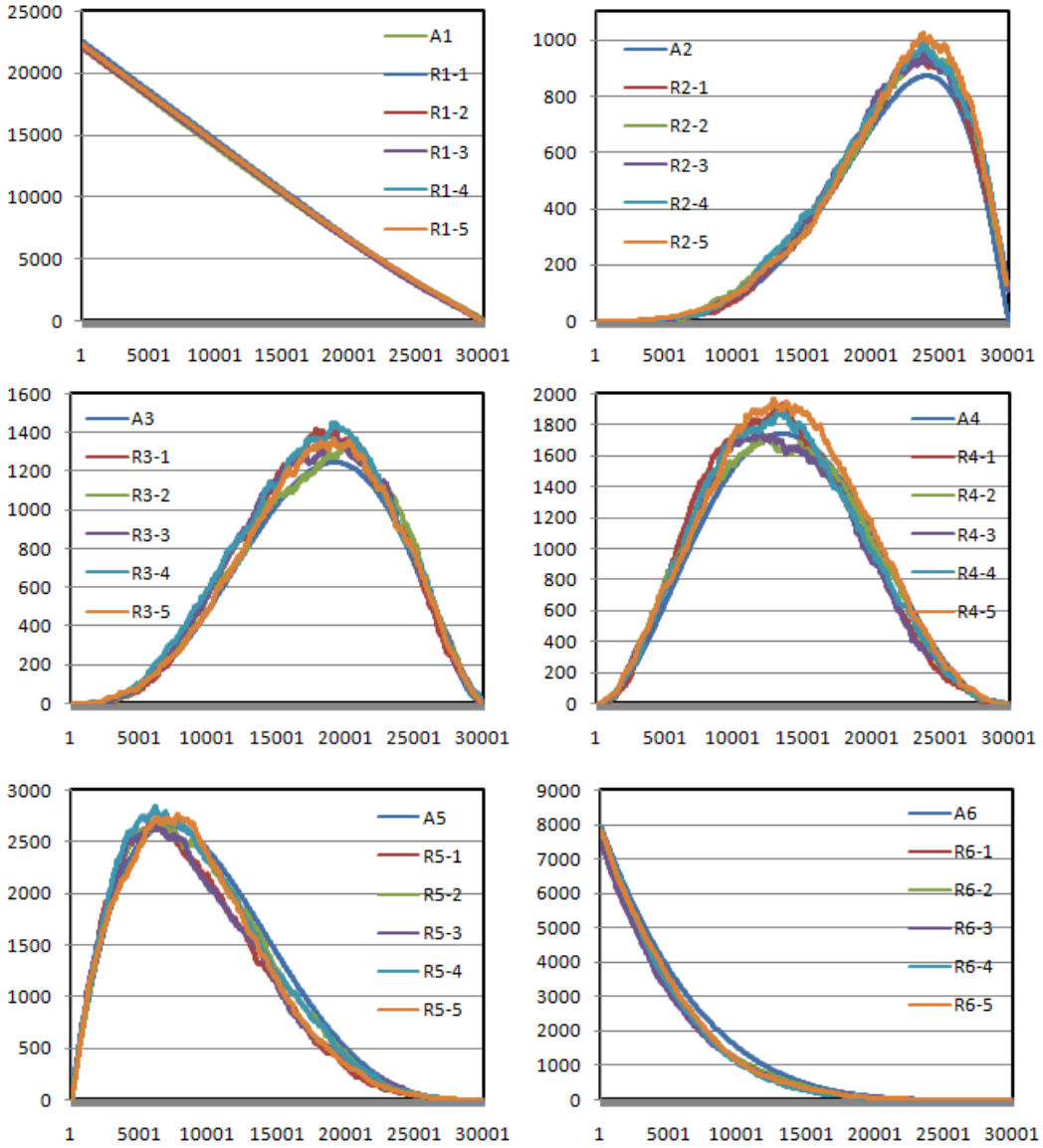


Figure 3: The theoretical and actual evolutions of edges of different degrees.

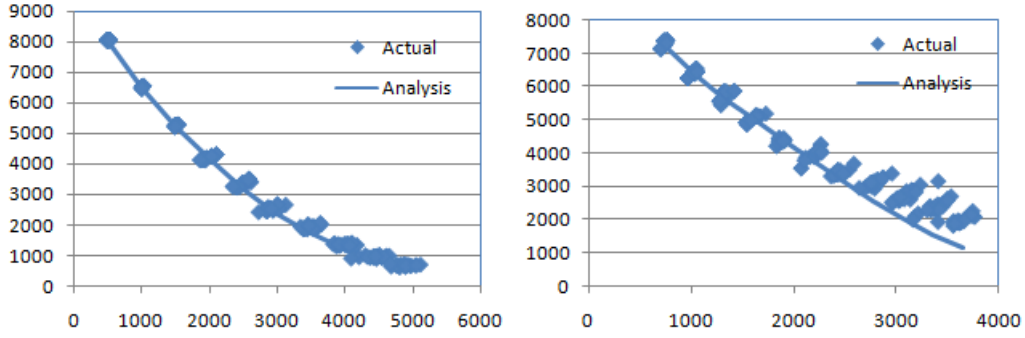


Figure 4: The actual and analyzed distortion vs. rate curves of $S = 6$ and $S = 12$.

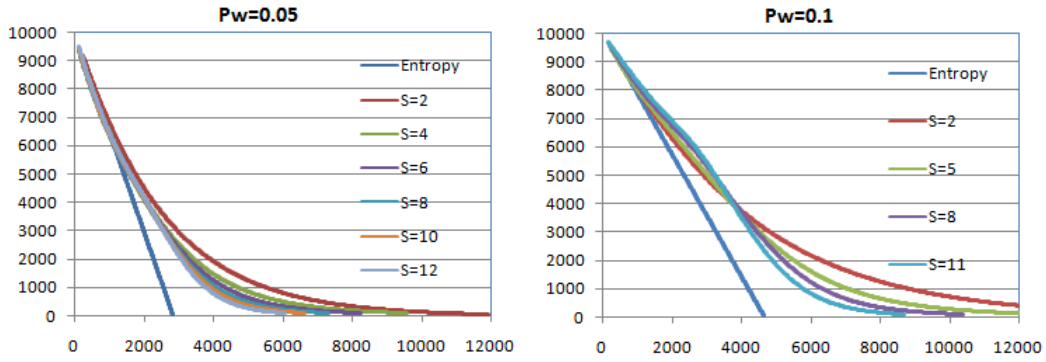


Figure 5: The rate-distortion curves of CS and entropy coding.

6. Conclusions

This paper coverts the decoding process of compressive sensing to the evolution of a bipartite graph and derives the close-form formulation of the evolution. Based on the formulation, we further analyze the performance of CS approximately and compare it with that of entropy coding. In future, we will derive the accurate distortion caused by unrecovered elements, which is able to conclude the performances between CS and entropy coding.

Acknowledgement

The authors are grateful to Peiwen Yu for introducing us the work in [17] and having some valuable discussions.

Appendix

The matrix $\boldsymbol{A}$ in (13) can be decomposed to $\boldsymbol{A} = \boldsymbol{P^{-1}\Lambda P}$, where $\boldsymbol{\Lambda}$ is a diagonal matrix and is defined as

$$\boldsymbol{\Lambda} = \begin{bmatrix} \theta_2 & 0 & \dots & 0 \\ 0 & \theta_3 & \dots & 0 \\ \dots & \dots & \ddots & \dots \\ 0 & 0 & \dots & \theta_S \end{bmatrix}. \tag{A-1}$$

Since $\boldsymbol{A}$ and the diagonal matrix $\boldsymbol{\Lambda}$ are similar matrix, $\theta_2, \dots, \theta_S$ should be eigenvalues of $\boldsymbol{A}$. Both sides of $\boldsymbol{A} = \boldsymbol{P^{-1}\,\Lambda P}$ are multiplied by $\boldsymbol{P^{-1}}$ on the right side and we can get $\boldsymbol{AP^{-1}} = \boldsymbol{P^{-1}\Lambda}$. We set $\boldsymbol{P^{-1}} = \boldsymbol{Q}$. In terms of sub-matrix multiplication, we have

$$\boldsymbol{AQ}_i = \theta_i \boldsymbol{Q}_i, i = 2, \dots, S. \tag{A-2}$$

It is easy to prove $\theta_i = -i$. The column vector $\boldsymbol{Q_i}$ is the eigenvector of the eigenvalue θ_i and can be obtained by solving

$$(\boldsymbol{A}+i\boldsymbol{I})\boldsymbol{Q}_i = 0. \tag{A-3}$$

Considering $\boldsymbol{A}$ in (13), (16) can be written in the following way

$$(\boldsymbol{A}+i\boldsymbol{I})\boldsymbol{Q_i} = \begin{bmatrix} i-2 & 2 & 0 & \dots & 0 & 0 \\ 0 & i-3 & 3 & \dots & 0 & 0 \\ \dots & \dots & \dots & \ddots & \dots & \dots \\ 0 & 0 & 0 & \dots & -S+1+i & S-1 \\ 0 & 0 & 0 & \dots & 0 & -S+i \end{bmatrix} \begin{bmatrix} q_{2i} \\ q_{3i} \\ \dots \\ q_{(S-1)i} \\ q_S \end{bmatrix} = 0. \tag{A-4}$$

In the matrix of $\boldsymbol{A} + i\boldsymbol{I}$, there exists only one non-zero element in the i-th row and it locates at the $(i+1)$-th column because the diagonal element is equal to zero in the row. Thus, q_{ji} for

$j > i$ must be equal to zero. In other words, $\boldsymbol{Q}$ is an upper triangle matrix. By further solving (A-4), $\boldsymbol{Q_i}$ (or $\boldsymbol{P^{-1}}$) is represented by

$$p_{ji}^{-1} = q_{ji} = \begin{cases} 1 & j = i \\ (-1)^{i-j} \prod_{k=j}^{i-1} \frac{k}{i-k} & 1 < j \leq i. \\ 0 & j > i \end{cases} \quad \text{(A-5)}$$

Similarly, $\boldsymbol{P}$ can be calculated by

$$p_{ij} = \begin{cases} 1 & i = j \\ (-1)^{j-i} \prod_{k=i}^{j-1} \frac{k}{i-k-1} & j < i \leq S. \\ 0 & i < j \end{cases} \quad \text{(A-6)}$$

References

[1] E. Candes and T. Tao, "Near optimal signal recovery from random projects: Universal encoding strategies?", IEEE trans. Information Theory, vol. 52, no. 12, pp5406-5425, 2006.

[2] D. L. Donoho, "Compressive sensing", IEEE trans. Information Theory, vol. 52, no 4, pp. 1289-1306, 2006.

[3] D. L. Donoho and J. Tanner, "Counting faces of randomly-projected polytopes when the projection radically lowers dimension," submitted for publication.

[4] S. Arvotham, D. Baron, and R. G. Baraniuk, "Measurements vs. bits: compressed sensing meets information theory," Proc. 44th Ann. Allerton Conf. on Comm. , Control and Comp., Monticello, IL, 2006.

[5] A. K. Fletcher, S. Rangan and V. K. Goyal, "Rate-distortion bounds for sparse approximation", IEEE/SP workshop on Statistical Signal Processing, pp 254-258, 2007.

[6] M. Akcakaya and V. Tarokh, "A frame construction and a universal distortion bound for sparse representations", IEEE trans. on Signal Processing, vol. 56, no. 6, pp 2443-2450, 2008.

[7] M. Wainwright, "Information-theoretic bounds on sparsity recovery in the high-dimensional and noisy setting", International Symposium on Information Theory, France, 2007.

[8] A. K. Fletcher, S. Rangan and V. K. Goyal, "On the rate-distortion performance of compressed sensing", IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 3, pp 885-888, 2007.

[9] V. K. Goyal, A. K. Fletcher and S. Rangan, "Compressive sampling and lossy compression", IEEE Signal Processing Magazine, pp.48-52, 2008.

[10] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," IEEE trans. Signal Processing, vol. 41, no. 12, pp. 3397-3415, 1993.

[11] S. S. Chen, D. L. Donoho and M. A. Saunders, "Atomic decomposition by basis pursuit," SIAM Review, vol. 43, no. 1, pp. 129-159, 2001.

[12] S. Sarvotham, D. Baron and R. G. Baraniuk, "Compressed sensing reconstruction via belief propagation," preprint.

[13] S. Sarvotham, D. Baron and R. G. Baraniuk, "Sudocodes – fast measurement and reconstruction of sparse signals", Proc International Symposium Information Theory, 2006.

[14] W. Xu and B. Hassibi, "Efficient compressive sensing with deterministic guarantees using expander graphs," IEEE workshop on Information theory, pp. 414-419, 2007.

[15] S. Jafarpour, W. Xu, B. Hassibi and R. Calderbank, "Efficient and robust compressed sensing using high-quality expander graphs," preprint.

[16] N. C. Wormald, "Differential equations for random processes and random graphs", Ann. Appl. Probability, vol. 5 , pp. 1217-1235, 1995.

[17] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, "Efficient erasure correcting codes," IEEE trans. on Information Theory, vol. 47, no. 2, 2001.