Using List Decoding to Improve the Finite-Length Performance of Sparse Regression Codes

Haiwen Cao and Pascal O. Vontobel

Department of Information Engineering
The Chinese University of Hong Kong

ch017@ie.cuhk.edu.hk, pascal.vontobel@ieee.org

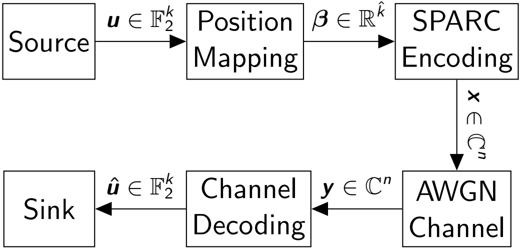
April 2021

Overview

Overview

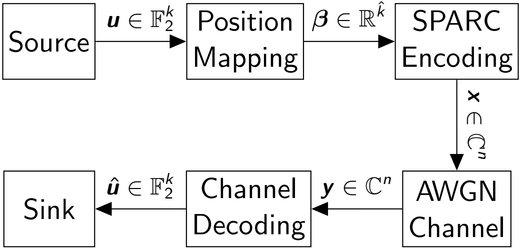
1. Sparse Regression Codes (SPARCs) and their Decoders
2. List Decoding for SPARCs concatenated with CRC codes
3. Simulation Results
4. Conclusion

Data Communication Model for SPARCs



Data Communication Model for SPARCs.

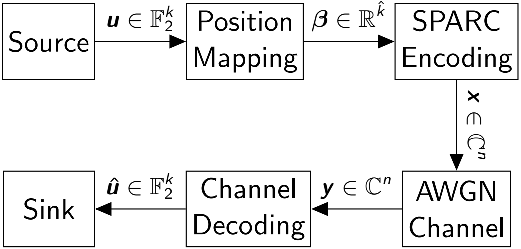
Data Communication Model for SPARCs



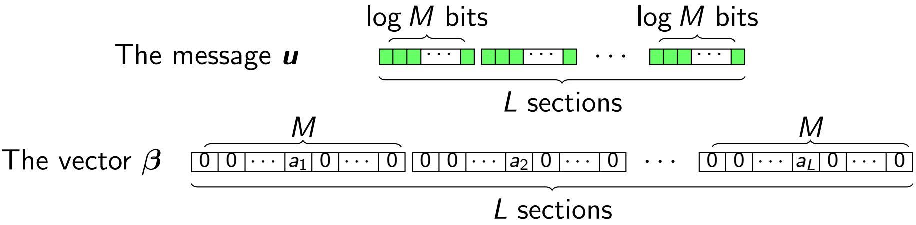
Data Communication Model for SPARCs.

Before we move into the SPARC encoding part, we need to convert the messages we want to transmit into a sparse structured vector $\boldsymbol{\beta}$ via "position mapping".

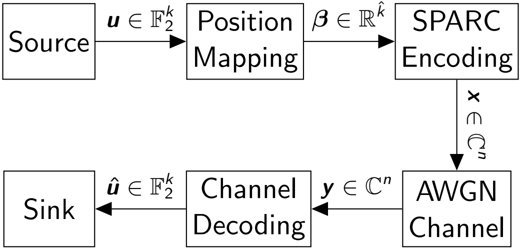
Data Communication Model for SPARCs



Data Communication Model for SPARCs.

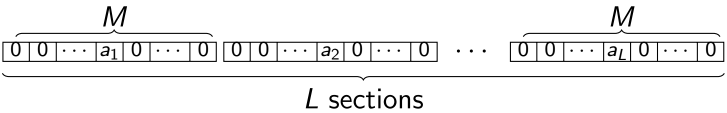


Data Communication Model for SPARCs

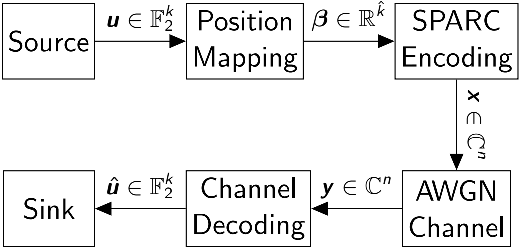


Data Communication Model for SPARCs.

The vector β



Data Communication Model for SPARCs

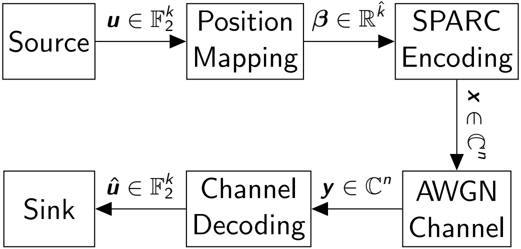


Data Communication Model for SPARCs.

The vector β $$\underbrace{\overbrace{[0 \mid 0 \mid \cdots \mid a_1 \mid 0 \mid \cdots \mid 0]}^{M}\ [0 \mid 0 \mid \cdots \mid a_2 \mid 0 \mid \cdots \mid 0]\ \cdots\ \overbrace{[0 \mid 0 \mid \cdots \mid a_L \mid 0 \mid \cdots \mid 0]}^{M}}_{L \text{ sections}}$$

The non-zero elements a_ℓ are chosen to be $\sqrt{nP_\ell}$, where $\sum_{\ell=1}^{L} P_\ell = P$.

Data Communication Model for SPARCs

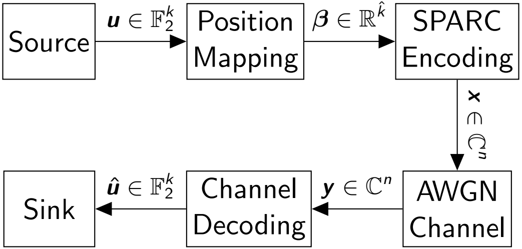


Data Communication Model for SPARCs.

The vector β $\underbrace{\overbrace{\boxed{0\,|\,0\,|\cdots|\,a_1\,|\,0\,|\cdots|\,0}}^{M}\ \boxed{0\,|\,0\,|\cdots|\,a_2\,|\,0\,|\cdots|\,0}\ \cdots\ \overbrace{\boxed{0\,|\,0\,|\cdots|\,a_L\,|\,0\,|\cdots|\,0}}^{M}}_{L \text{ sections}}$

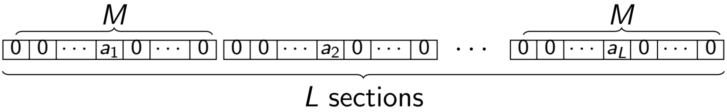
This guarantees that $\frac{1}{n}\sum_{i=1}^{n}|x_i|^2 \leq P$ with high probability.

Data Communication Model for SPARCs



Data Communication Model for SPARCs.

The vector β



The choice of $\{P_\ell\}$ affects the finite-length performance of SPARCs and the iterative power allocation scheme gives us the resulting $\{P_\ell\}$.

SPARC Encoding

- The codeword $\boldsymbol{x}$ of length n is given by the matrix-vector multiplication, i.e., $\boldsymbol{x} = \boldsymbol{A}\boldsymbol{\beta}$.

SPARC Encoding

- The codeword $\boldsymbol{x}$ of length n is given by the matrix-vector multiplication, i.e., $\boldsymbol{x} = \boldsymbol{A}\boldsymbol{\beta}$.

- The matrix $\boldsymbol{A}$ of size $n \times ML$ is the so-called design matrix and its entries are i.i.d. Gaussian $\sim \mathcal{CN}(0, 1/n)$.

SPARC Encoding

- The codeword $\boldsymbol{x}$ of length n is given by the matrix-vector multiplication, i.e., $\boldsymbol{x} = \boldsymbol{A}\boldsymbol{\beta}$.
- The matrix $\boldsymbol{A}$ of size $n \times ML$ is the so-called design matrix and its entries are i.i.d. Gaussian $\sim \mathcal{CN}(0, 1/n)$.
- In actual implementation, we usually use the suitably sub-sampled discrete Fourier transform (DFT) matrix as the design matrix $\boldsymbol{A}$ instead of the original design matrix in order to reduce the encoding complexity.

Various Decoders for SPARCs (over real-valued AWGNs)

At the receiver side,

- the received vector $\boldsymbol{y}$ can be expressed as $\boldsymbol{A}\boldsymbol{\beta} + \boldsymbol{w}$.

- The additive noise vector $\boldsymbol{w} = (w_i)_{i \in [n]}$ and w_i are i.i.d. $\mathcal{CN}(0, \sigma^2)$ for all $i \in [n]$.

- Our goal for decoding is to estimate $\boldsymbol{\beta}$ based on $\boldsymbol{y}$, the design matrix $\boldsymbol{A}$, and the structure of $\boldsymbol{\beta}$.

Various Decoders for SPARCs (over real-valued AWGNs)

There are a few previous results on decoding of SPARCs (over real-valued AWGNs) listed chronologically.

- SPARCs were first introduced by Joseph and Barron (2012) and the optimal decoder (i.e., the maximum likelihood decoder) was proposed accordingly.
- Joseph and Barron (2014) introduced an efficient decoding algorithm called "adaptive successive decoding".
- An adaptive soft-decision successive decoder was proposed by Barron and Cho (2012).
- The **approximate massage passing (AMP)** decoder was first proposed by Barbier and Krzakala (2014), and then it was rigorously proven to be asymptotically capacity-achieving by Rush *et al.* (2017).

AMP decoding for SPARCs over complex-valued AWGNs

Initialize $\boldsymbol{\beta}^0 := 0$. For $t = 0, 1, 2, \ldots$, compute

$$\boldsymbol{z}^t := \boldsymbol{y} - \boldsymbol{A}\boldsymbol{\beta}^t + \frac{\boldsymbol{z}^{t-1}}{\tau_{t-1}^2}\left(P - \frac{\|\boldsymbol{\beta}^t\|^2}{n}\right),$$

$$\beta_i^{t+1} := \eta_i^t\left(\boldsymbol{\beta}^t + \boldsymbol{A}^*\boldsymbol{z}^t\right), \; i = 1, \ldots, ML,$$

AMP decoding for SPARCs over complex-valued AWGNs

Initialize $\beta^0 := 0$. For $t = 0, 1, 2, \ldots$, compute

$$\boldsymbol{z}^t := \boldsymbol{y} - \boldsymbol{A}\beta^t + \underbrace{\frac{\boldsymbol{z}^{t-1}}{\tau_{t-1}^2}\left(P - \frac{\|\boldsymbol{\beta}^t\|^2}{n}\right)}_{\text{“Onsager term”}},$$

$$\beta_i^{t+1} := \eta_i^t\left(\boldsymbol{\beta}^t + \boldsymbol{A}^*\boldsymbol{z}^t\right), \ i = 1, \ldots, ML,$$

AMP decoding for SPARCs over complex-valued AWGNs

Initialize $\beta^0 := 0$. For $t = 0, 1, 2, \ldots$, compute

$$\boldsymbol{z}^t := \boldsymbol{y} - \boldsymbol{A}\boldsymbol{\beta}^t + \underbrace{\frac{\boldsymbol{z}^{t-1}}{\tau_{t-1}^2}\left(P - \frac{\|\boldsymbol{\beta}^t\|^2}{n}\right)}_{\text{“Onsager term”}},$$

$$\beta_i^{t+1} := \eta_i^t \underbrace{\left(\boldsymbol{\beta}^t + \boldsymbol{A}^* \boldsymbol{z}^t\right)}_{\approx \boldsymbol{\beta} + \tau^t \boldsymbol{u}}, \; i = 1, \ldots, ML.$$

AMP decoding for SPARCs over complex-valued AWGNs

Initialize $\boldsymbol{\beta}^0 := 0$. For $t = 0, 1, 2, \ldots$, compute

$$\boldsymbol{z}^t := \boldsymbol{y} - \boldsymbol{A}\boldsymbol{\beta}^t + \underbrace{\frac{\boldsymbol{z}^{t-1}}{\tau_{t-1}^2}\left(P - \frac{\|\boldsymbol{\beta}^t\|^2}{n}\right)}_{\text{“Onsager term”}},$$

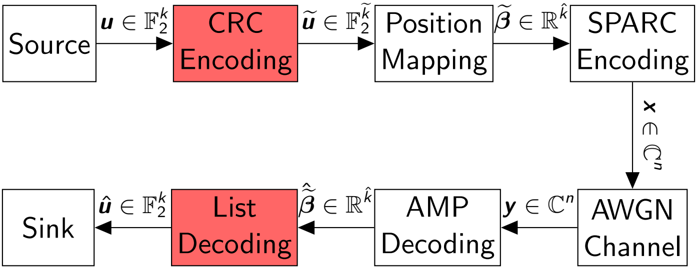
$$\beta_i^{t+1} := \eta_i^t \underbrace{\left(\boldsymbol{\beta}^t + \boldsymbol{A}^* \boldsymbol{z}^t\right)}_{\approx \boldsymbol{\beta} + \tau^t \boldsymbol{u}}, \ \ i = 1, \ldots, ML.$$

- The additive Gaussian noise vector $\boldsymbol{u}$ has i.i.d. $\mathcal{CN}(0, 1)$ entries and is independent with $\boldsymbol{\beta}$.
- The constants $\{\tau_t\}$ can be determined via the state evolution.
- In actual implementation, we use an online estimate $\hat{\tau}_t^2 = \frac{\|\boldsymbol{z}^t\|^2}{n}$.
- the denoiser functions $\eta_i^t(\cdot)$ are the Bayes-optimal estimators.

Overview

1. Sparse Regression Codes (SPARCs) and their Decoders
2. List Decoding for SPARCs concatenated with CRC codes
3. Simulation Results
4. Conclusion

Block diagram for SPARCs concatenated with CRC codes



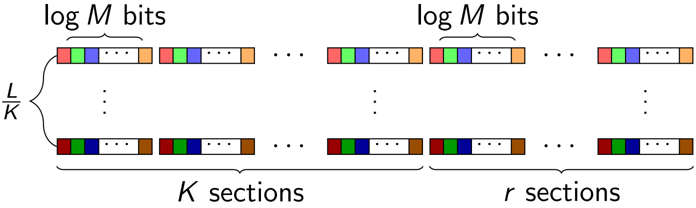
Block diagram of a communication system employing SPARCs combined with CRC codes as outer-error detection codes.

CRC Encoding

There are two ways to employ CRC codes:

- an "inter-section-based approach" that encodes the original message to generate extra check sections bit by bit when we focus on the bit error rate (BER) performance;
- an "intra-section-based approach" that encodes them section by section when we focus on the section error rate (SecER) performance.

The inter-section-based approach is illustrated via the following figure.



List Decoding

1. Perform T iterations of AMP decoding; the resulting estimate of $\widetilde{\beta}$ is called $\widetilde{\beta}^{(T)}$.

List Decoding

1. Perform T iterations of AMP decoding; the resulting estimate of $\widetilde{\boldsymbol{\beta}}$ is called $\widetilde{\boldsymbol{\beta}}^{(T)}$.
2. For each section $\ell \in [\widetilde{L}]$, normalizing $\widetilde{\boldsymbol{\beta}}_{\ell}^{(T)}$ gives the a posterior distribution estimate of the location of the non-zero entry of $\widetilde{\boldsymbol{\beta}}_{\ell}$, denoted by $\hat{\widetilde{\boldsymbol{\beta}}}_{\ell}^{(T)}$.

List Decoding

1. Perform T iterations of AMP decoding; the resulting estimate of $\widetilde{\boldsymbol{\beta}}$ is called $\widetilde{\boldsymbol{\beta}}^{(T)}$.
2. For each section $\ell \in [\widetilde{L}]$, normalizing $\widetilde{\boldsymbol{\beta}}_{\ell}^{(T)}$ gives the a posterior distribution estimate of the location of the non-zero entry of $\widetilde{\boldsymbol{\beta}}_{\ell}$, denoted by $\hat{\widetilde{\boldsymbol{\beta}}}_{\ell}^{(T)}$.
3. For each section $\ell \in [\widetilde{L}]$, convert the posterior distribution estimate $\hat{\widetilde{\boldsymbol{\beta}}}_{\ell}^{(T)}$ into $\log_2 M$ bit-wise posterior distribution estimates.

List Decoding

1. Perform T iterations of AMP decoding; the resulting estimate of $\widetilde{\beta}$ is called $\widetilde{\beta}^{(T)}$.
2. For each section $\ell \in [\widetilde{L}]$, normalizing $\widetilde{\beta}_\ell^{(T)}$ gives the a posterior distribution estimate of the location of the non-zero entry of $\widetilde{\beta}_\ell$, denoted by $\hat{\widetilde{\beta}}_\ell^{(T)}$.
3. For each section $\ell \in [\widetilde{L}]$, convert the posterior distribution estimate $\hat{\widetilde{\beta}}_\ell^{(T)}$ into $\log_2 M$ bit-wise posterior distribution estimates.
4. For each codeword C_i, we establish a binary tree of depth $K + r$, where, starting at the root, at each layer, we keep at most S branches, which are the most likely ones.

List Decoding

1. Perform T iterations of AMP decoding; the resulting estimate of $\widetilde{\boldsymbol{\beta}}$ is called $\widetilde{\boldsymbol{\beta}}^{(T)}$.
2. For each section $\ell \in [\widetilde{L}]$, normalizing $\widetilde{\boldsymbol{\beta}}_\ell^{(T)}$ gives the a posterior distribution estimate of the location of the non-zero entry of $\widetilde{\boldsymbol{\beta}}_\ell$, denoted by $\hat{\widetilde{\boldsymbol{\beta}}}_\ell^{(T)}$.
3. For each section $\ell \in [\widetilde{L}]$, convert the posterior distribution estimate $\hat{\widetilde{\boldsymbol{\beta}}}_\ell^{(T)}$ into $\log_2 M$ bit-wise posterior distribution estimates.
4. For each codeword C_i, we establish a binary tree of depth $K + r$, where, starting at the root, at each layer, we keep at most S branches, which are the most likely ones.
5. For each codeword C_i, once we have established such a binary tree, list decoding will give us S ordered candidates corresponding to the remaining S paths from the root to the leaves.

AMP again

Besides the regular list decoding assisted with CRC codes, running AMP decoding and list decoding again for wrongly decoded sections might further improve the performance. More specifically, we can apply the following procedure:

1. Run AMP decoding as before, except that at each iteration, fix the "correctly decoded" parts of the message and only estimate the other sections. When the maximum number of iteration T is reached or some halting condition is satisfied, the algorithm outputs $\widetilde{\boldsymbol{\beta}}^*$.

2. Take only the wrongly decoded sections of $\widetilde{\boldsymbol{\beta}}^*$, denoted by $\widetilde{\boldsymbol{\beta}}^*_{\mathrm{WD}}$, and apply the above list decoding procedure to $\widetilde{\boldsymbol{\beta}}^*_{\mathrm{WD}}$, which gives the decoded message.

Overview

Simulation Results

We consider the following setups for our simulation results.

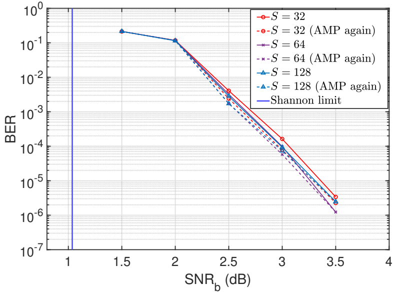
- we choose the number of information sections L to be 1000,
- we choose the size of each section M to be 512,
- we choose the number of CRC code information bits K to be 100,
- we use the CRC code with 8 redundant bits and its generator polynomial is 0x97$= x^8 + x^5 + x^3 + x^2 + x + 1$.

Simulation Results

We consider SPARCs with overall rate $R = 0.8$ bits/(channel use)/dimension, in which case $R_{\mathrm{PA}} = 0$ and the iterative power allocation scheme gives a flat allocation.

Simulation Results

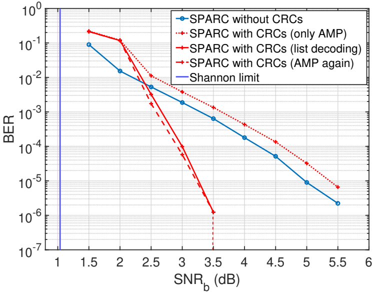
We consider SPARCs with overall rate $R = 0.8$ bits/(channel use)/dimension, in which case $R_{\mathrm{PA}} = 0$ and the iterative power allocation scheme gives a flat allocation.



- The figure shows the BER performance comparison of SPARCs with CRC codes for different list sizes.
- From this figure we deduce that $S = 64$ is the best choice for the considered setup.

Simulation Results

We consider SPARCs with overall rate $R = 0.8$ bits/(channel use)/dimension, in which case $R_{\mathrm{PA}} = 0$ and the iterative power allocation scheme gives a flat allocation.



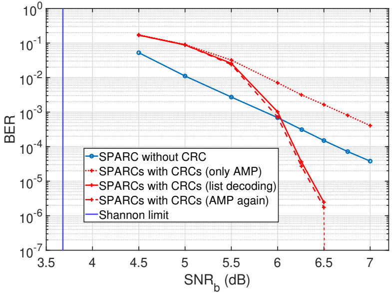
- F igure shows the BER performance comparison of low-rate SPARCs with CRC codes using list decoding and original SPARCs without CRC codes using only AMP.
- The figure shows that SPARCs concatenated with CRC codes can provide a steep waterfall-like behavior above a threshold of $\mathrm{SNR_b} = 3.5$ dB.

Simulation Results

We consider SPARCs with overall rate $R = 1.5$ bits/(channel use)/dimension, in which case $R_{\mathrm{PA}} \approx 3$ for the iterative power allocation scheme.

Simulation Results

We consider SPARCs with overall rate $R = 1.5$ bits/(channel use)/dimension, in which case $R_{\mathrm{PA}} \approx 3$ for the iterative power allocation scheme.



- Figure shows the BER performance comparison of high-rate SPARCs with CRC codes using list decoding and original SPARCs without CRC codes using only AMP.
- The figure shows that SPARCs concatenated with CRC codes can provide a steep waterfall-like behavior above a threshold of $\mathrm{SNR_b} = 6.5$ dB.

Overview

- We introduced AMP decoding for SPARCs over complex-valued AWGN channels.

- We introduced AMP decoding for SPARCs over complex-valued AWGN channels.

- We proposed a concatenated coding scheme that uses SPARCs concatenated with CRC codes on the encoding side and uses list decoding on the decoding side.

- We introduced AMP decoding for SPARCs over complex-valued AWGN channels.
- We proposed a concatenated coding scheme that uses SPARCs concatenated with CRC codes on the encoding side and uses list decoding on the decoding side.
- Simulation results showed that the finite-length performance is significantly improved compared with the original SPARCs.

Thanks!