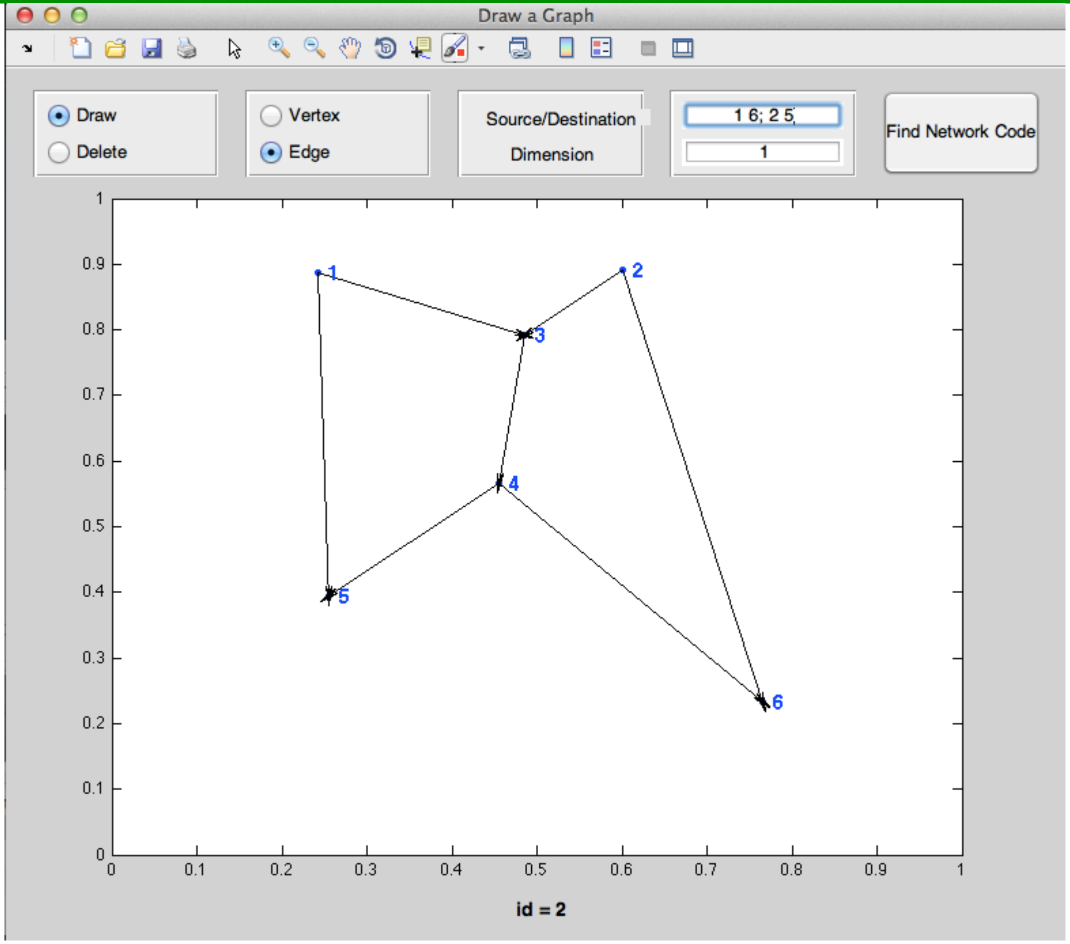
Index Coding Algorithms for Constructing Network Codes

Salim El Rouayheb

Illinois Institute of Technology, Chicago

Matlab Code with GUI for Constructing Linear Network Codes



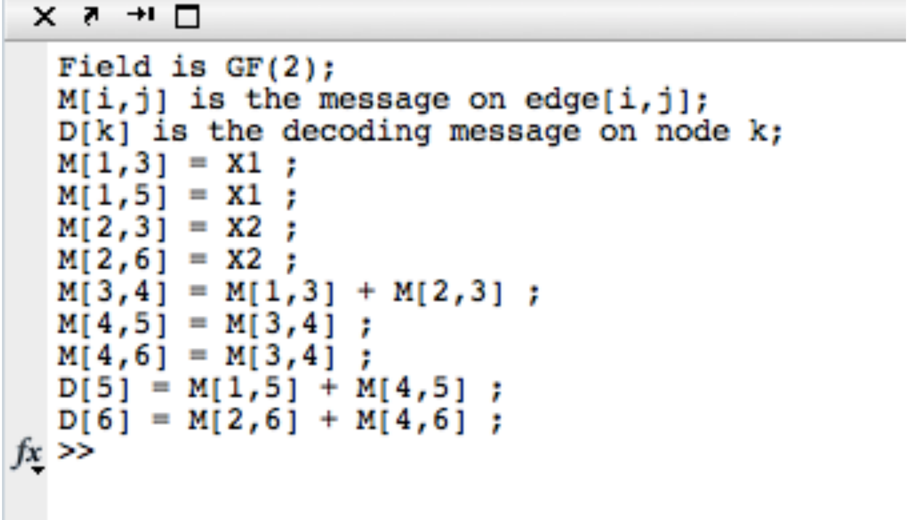
● Draw
○ Delete

○ Vertex
● Edge

Source/Destination: 1 6; 2 5,
Dimension: 1

GUI

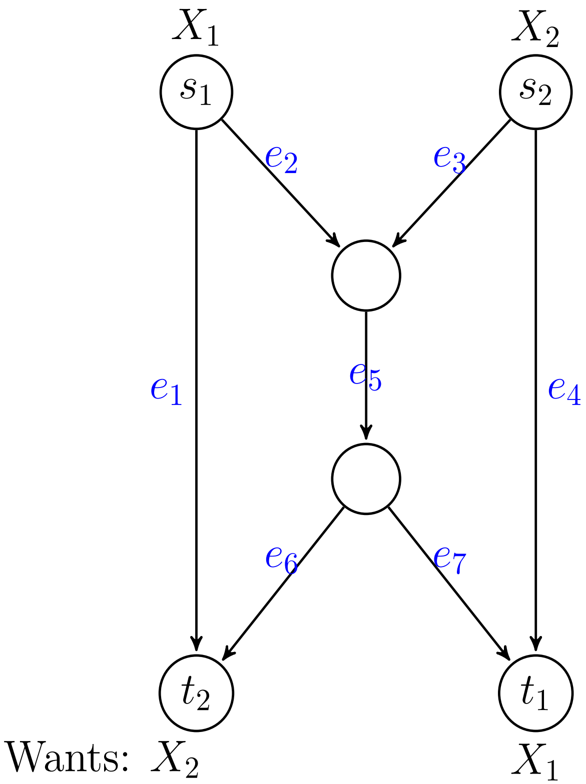
Matlab outputs network code



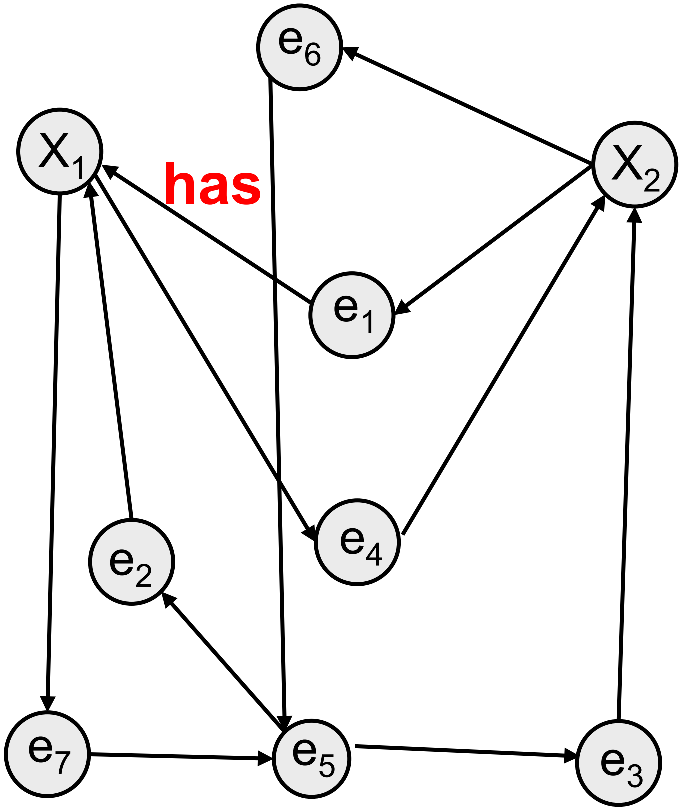
```
Field is GF(2);
M[i,j] is the message on edge[i,j];
D[k] is the decoding message on node k;
M[1,3] = X1 ;
M[1,5] = X1 ;
M[2,3] = X2 ;
M[2,6] = X2 ;
M[3,4] = M[1,3] + M[2,3] ;
M[4,5] = M[3,4] ;
M[4,6] = M[3,4] ;
D[5] = M[1,5] + M[4,5] ;
D[6] = M[2,6] + M[4,6] ;
>>
```

Download Matlab code here:

www.tinyurl.com/IndexCodingRocks

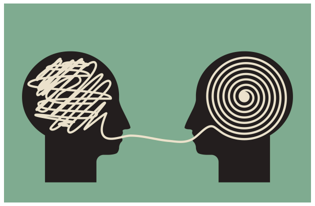


Network Coding Problem

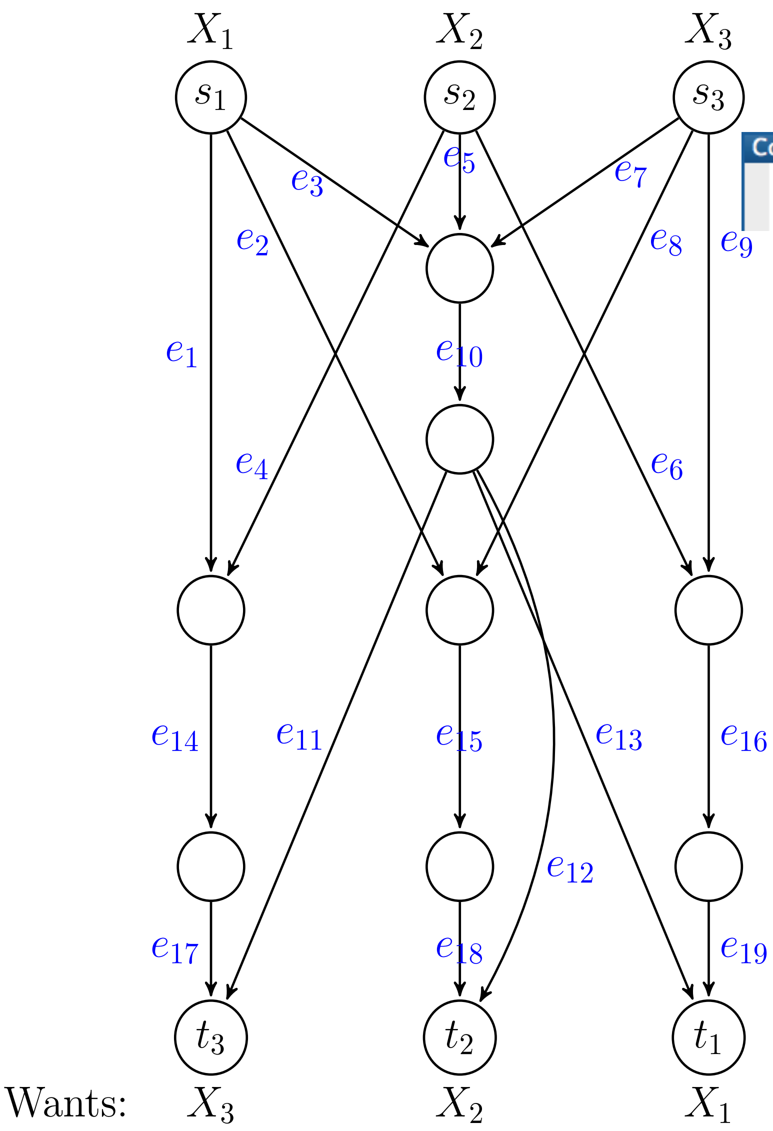


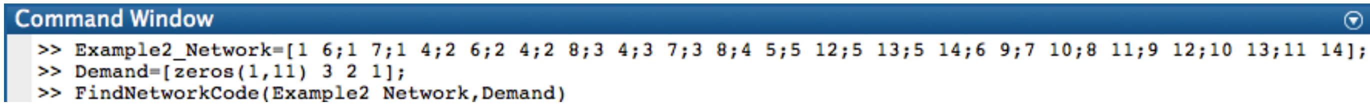
Index Coding Problem

Index Code

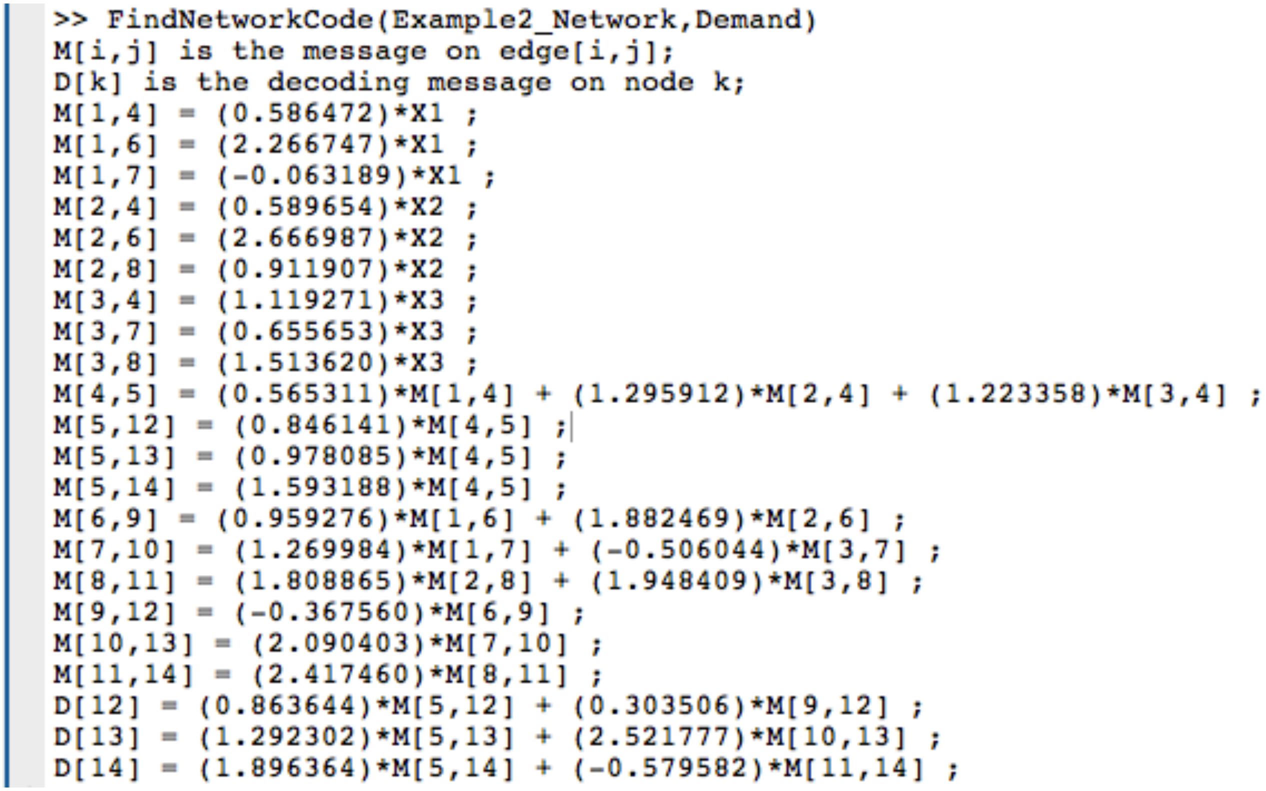


Example 2



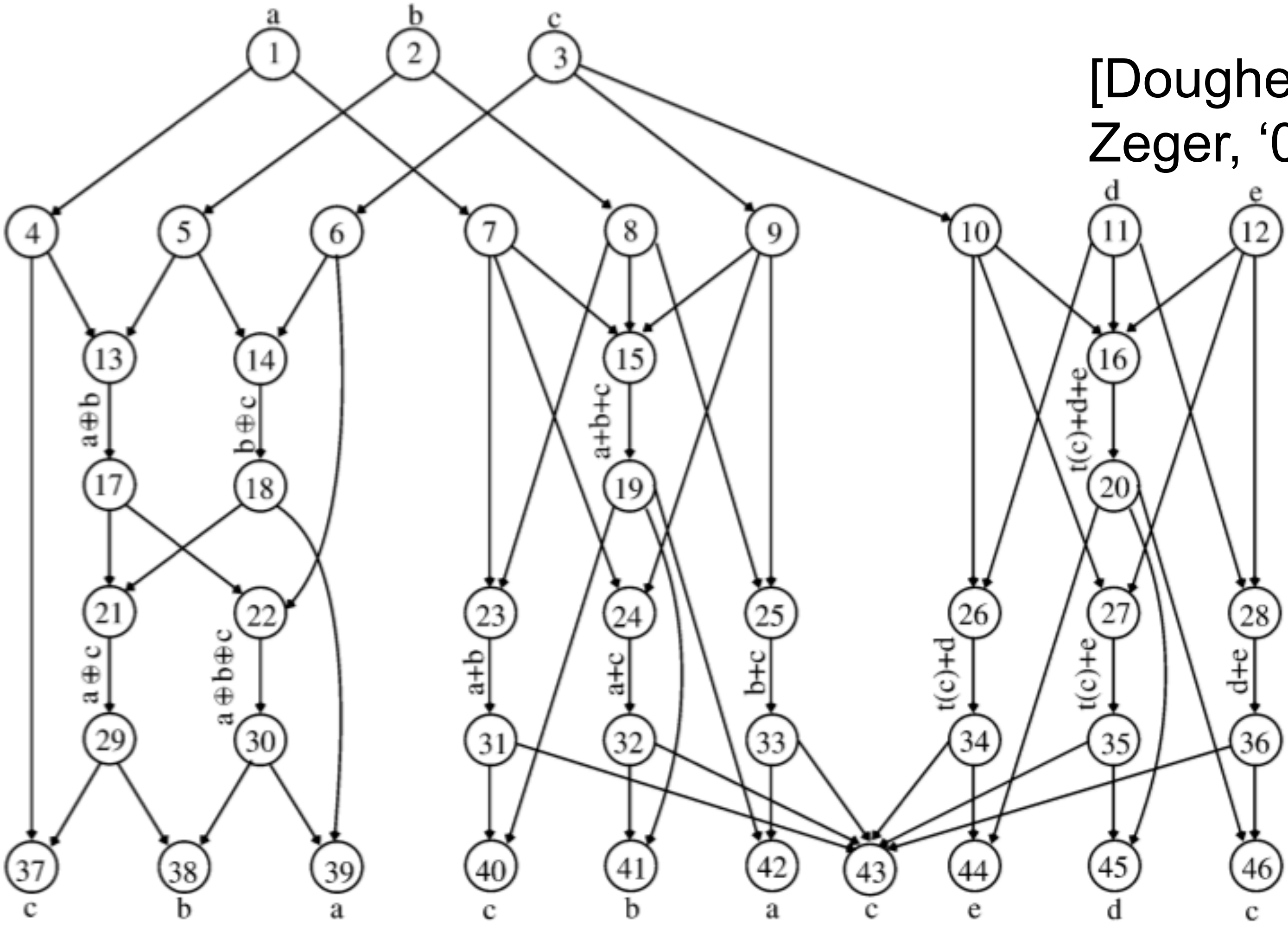


```
>> Example2_Network=[1 6;1 7;1 4;2 6;2 4;2 8;3 4;3 7;3 8;4 5;5 12;5 13;5 14;6 9;7 10;8 11;9 12;10 13;11 14];
>> Demand=[zeros(1,11) 3 2 1];
>> FindNetworkCode(Example2 Network,Demand)
```

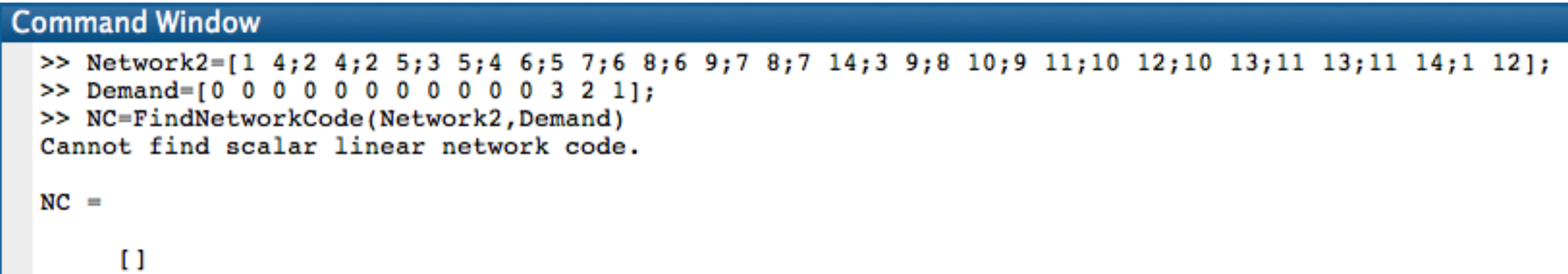


```
>> FindNetworkCode(Example2_Network,Demand)
M[i,j] is the message on edge[i,j];
D[k] is the decoding message on node k;
M[1,4] = (0.586472)*X1 ;
M[1,6] = (2.266747)*X1 ;
M[1,7] = (-0.063189)*X1 ;
M[2,4] = (0.589654)*X2 ;
M[2,6] = (2.666987)*X2 ;
M[2,8] = (0.911907)*X2 ;
M[3,4] = (1.119271)*X3 ;
M[3,7] = (0.655653)*X3 ;
M[3,8] = (1.513620)*X3 ;
M[4,5] = (0.565311)*M[1,4] + (1.295912)*M[2,4] + (1.223358)*M[3,4] ;
M[5,12] = (0.846141)*M[4,5] ;
M[5,13] = (0.978085)*M[4,5] ;
M[5,14] = (1.593188)*M[4,5] ;
M[6,9] = (0.959276)*M[1,6] + (1.882469)*M[2,6] ;
M[7,10] = (1.269984)*M[1,7] + (-0.506044)*M[3,7] ;
M[8,11] = (1.808865)*M[2,8] + (1.948409)*M[3,8] ;
M[9,12] = (-0.367560)*M[6,9] ;
M[10,13] = (2.090403)*M[7,10] ;
M[11,14] = (2.417460)*M[8,11] ;
D[12] = (0.863644)*M[5,12] + (0.303506)*M[9,12] ;
D[13] = (1.292302)*M[5,13] + (2.521777)*M[10,13] ;
D[14] = (1.896364)*M[5,14] + (-0.579582)*M[11,14] ;
```

Non-linear code

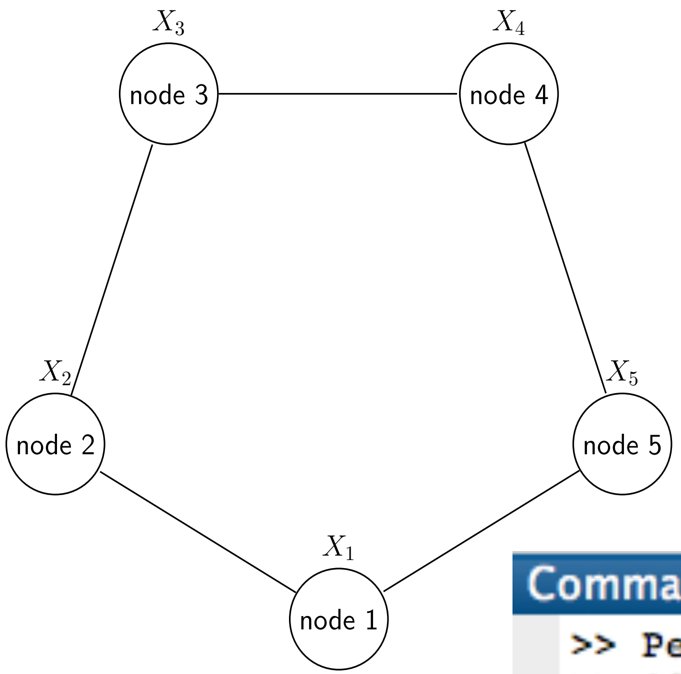


[Dougherty, Freiling, Zeger, '05]

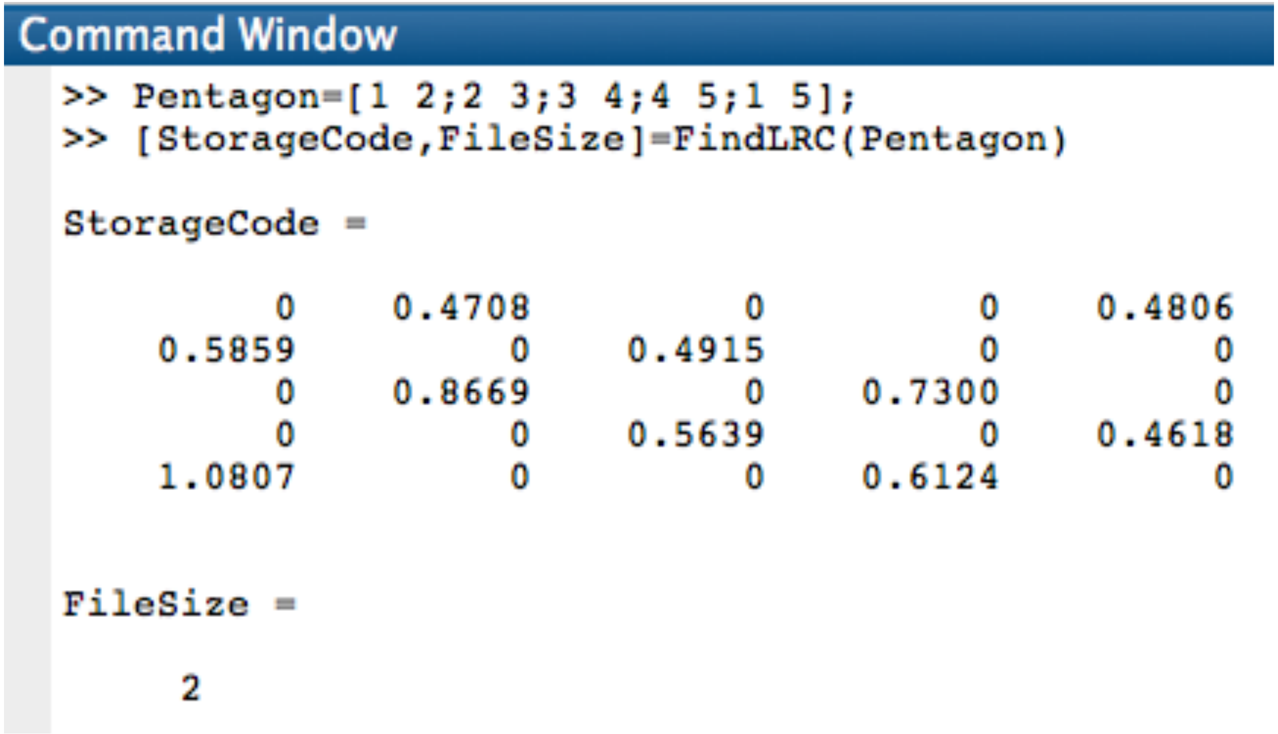


```
>> Network2=[1 4;2 4;2 5;3 5;4 6;5 7;6 8;6 9;7 8;7 14;3 9;8 10;9 11;10 12;10 13;11 13;11 14;1 12];
>> Demand=[0 0 0 0 0 0 0 0 0 0 0 3 2 1];
>> NC=FindNetworkCode(Network2,Demand)
Cannot find scalar linear network code.

NC =

     []
```

Locally Repairable Code

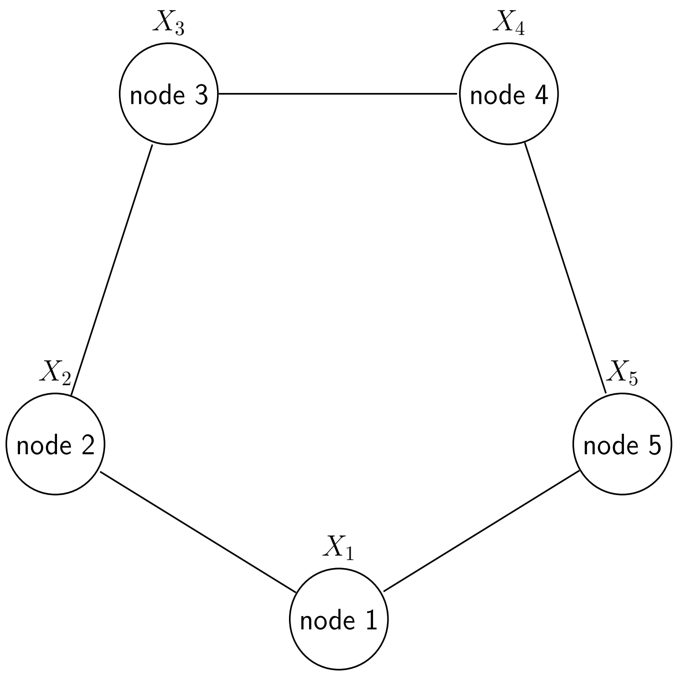


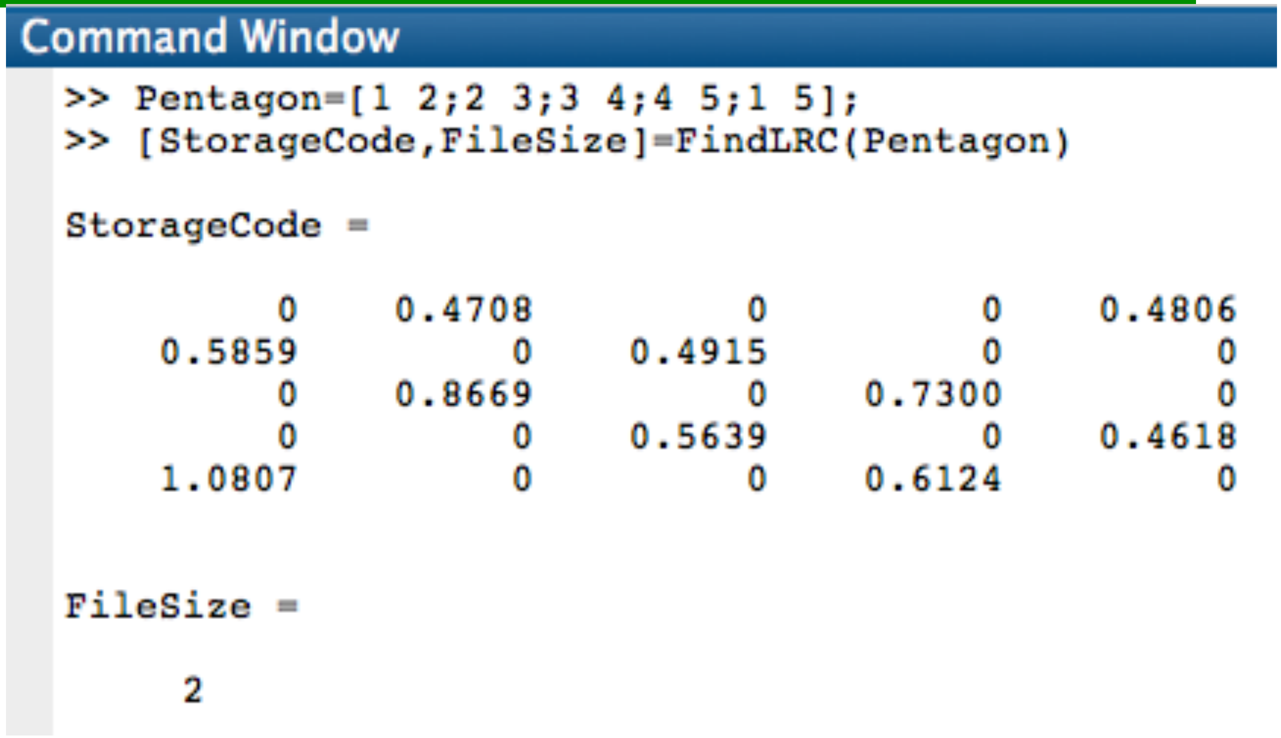
- Constructing Linear Repairable Codes* is equivalent to constructing linear index codes
- [Mazumdar '14],[Shanmugam, Dimakis'14]



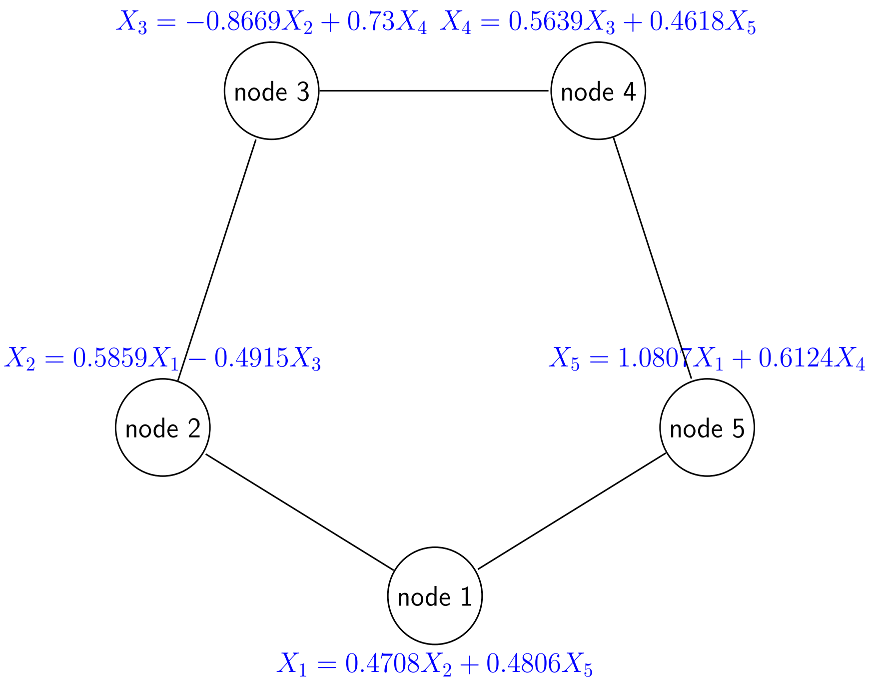
```
>> Pentagon=[1 2;2 3;3 4;4 5;1 5];
>> [StorageCode,FileSize]=FindLRC(Pentagon)

StorageCode =

         0    0.4708         0         0    0.4806
    0.5859         0    0.4915         0         0
         0    0.8669         0    0.7300         0
         0         0    0.5639         0    0.4618
    1.0807         0         0    0.6124         0


FileSize =

     2
```

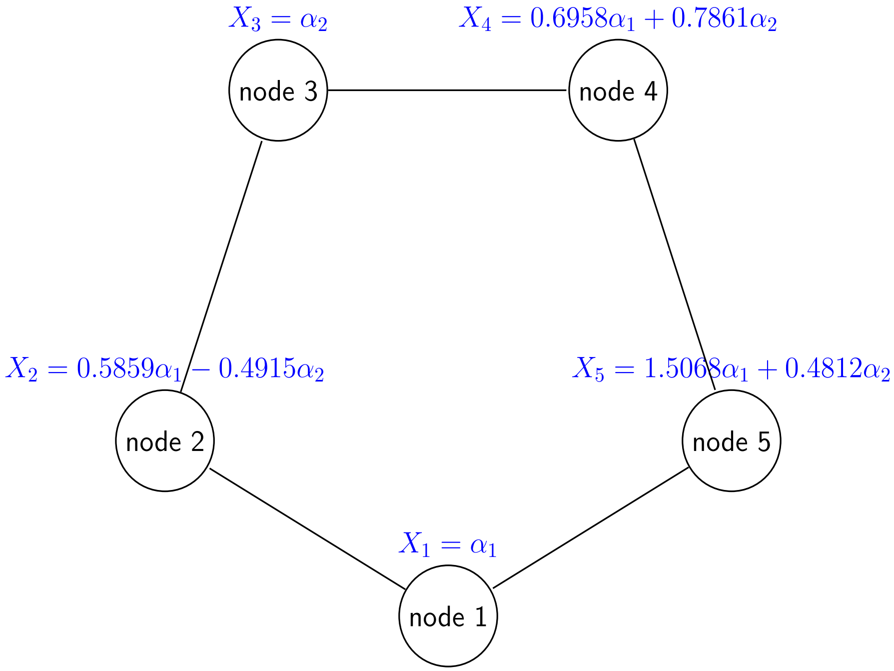
Locally Repairable Code



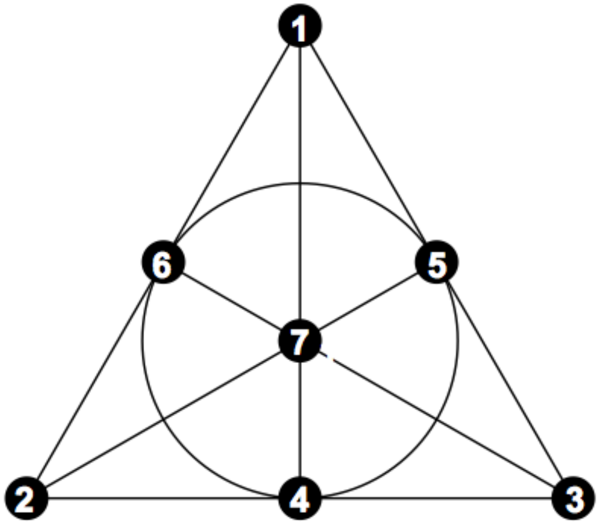


```
>> Pentagon=[1 2;2 3;3 4;4 5;1 5];
>> [StorageCode,FileSize]=FindLRC(Pentagon)

StorageCode =

        0    0.4708         0         0    0.4806
   0.5859         0    0.4915         0         0
        0    0.8669         0    0.7300         0
        0         0    0.5639         0    0.4618
   1.0807         0         0    0.6124         0


FileSize =

     2
```

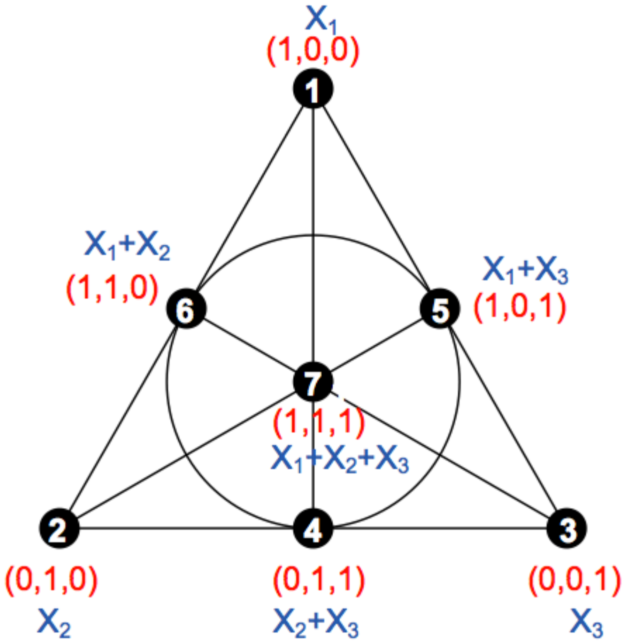




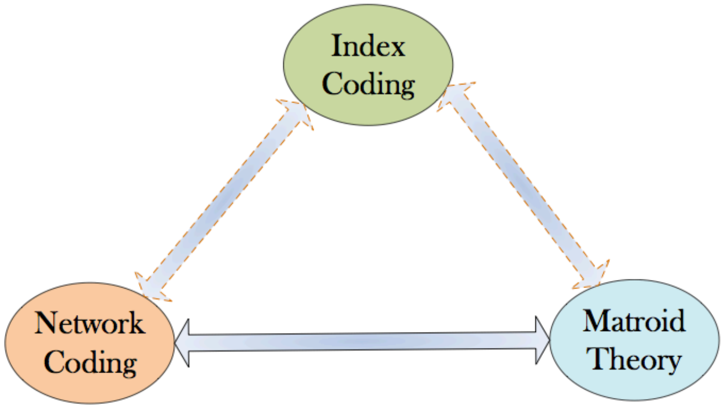
Matroids and Index Coding



Fano matroid



Linear representation over GF(2)
(does not exist over GF(3))

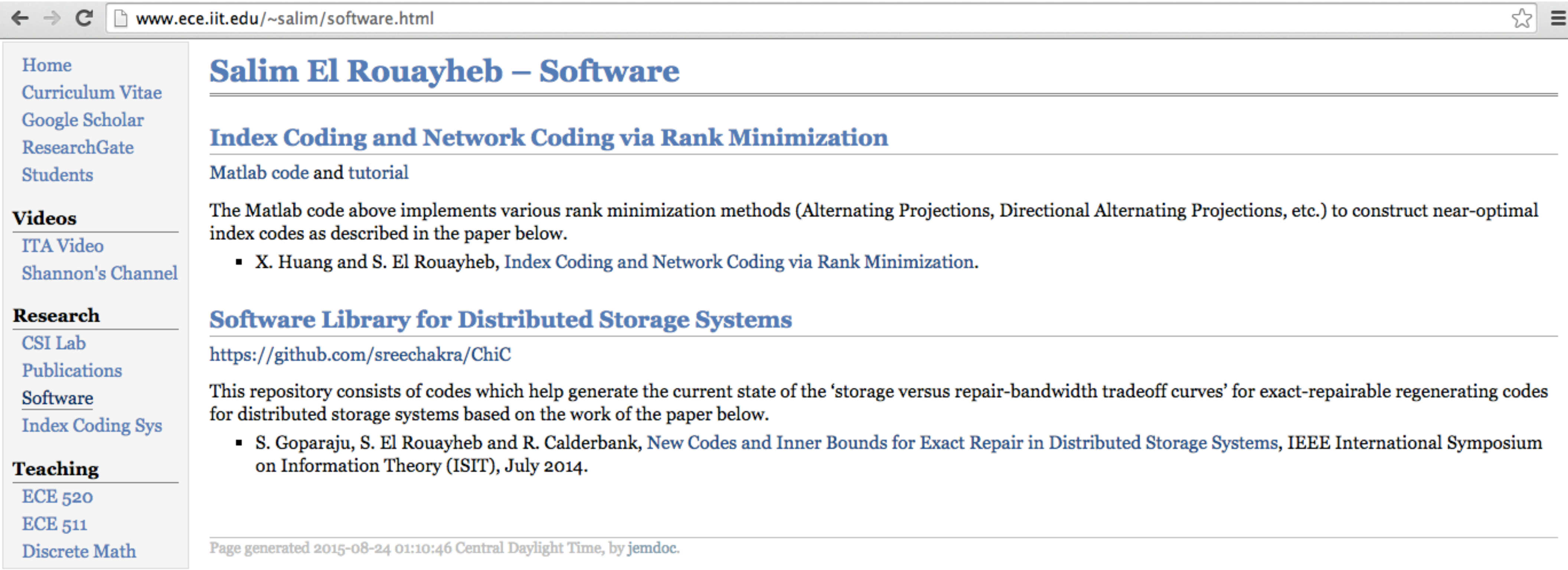


Theorem: [R,Sprintson, Georghiades ’09]

For any matroid M(E, r(.)), one can construct an index coding problem and an integer $c=|E|+r(E)$ such that there exists a linear index code of length c over F, iff the matroid M has a representation over F.

Matlab Code Available Online

www.tinyurl.com/IndexCodingRocks



Salim El Rouayheb – Software

Index Coding and Network Coding via Rank Minimization

Matlab code and tutorial

The Matlab code above implements various rank minimization methods (Alternating Projections, Directional Alternating Projections, etc.) to construct near-optimal index codes as described in the paper below.

- X. Huang and S. El Rouayheb, Index Coding and Network Coding via Rank Minimization.

Software Library for Distributed Storage Systems

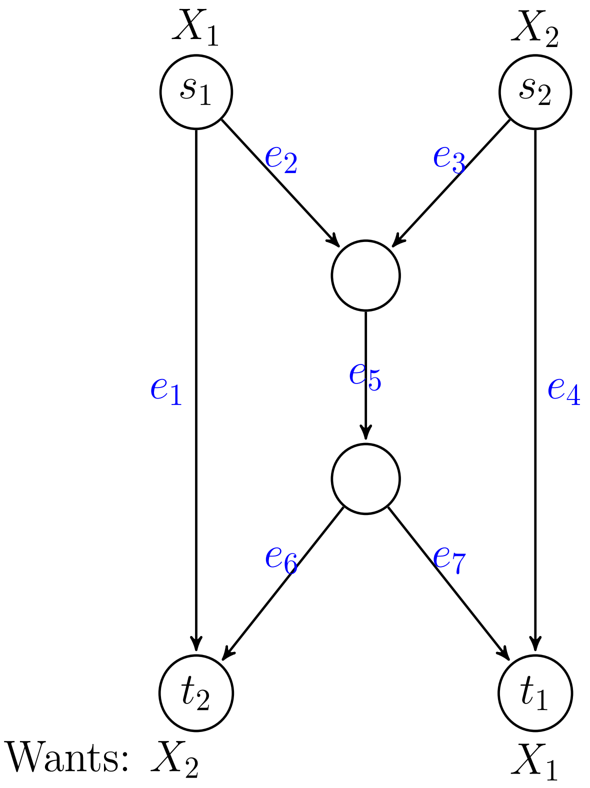
https://github.com/sreechakra/ChiC

This repository consists of codes which help generate the current state of the ‘storage versus repair-bandwidth tradeoff curves’ for exact-repairable regenerating codes for distributed storage systems based on the work of the paper below.

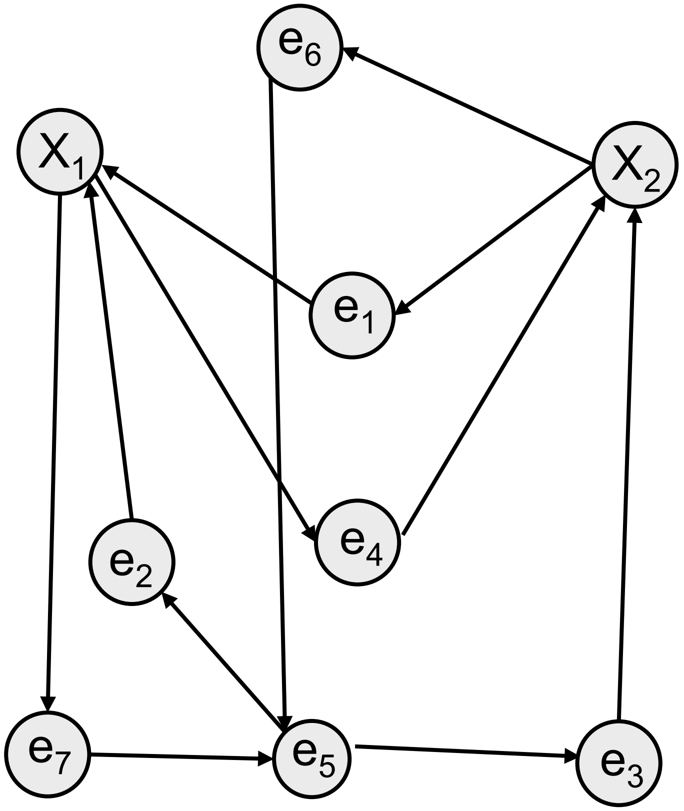
- S. Goparaju, S. El Rouayheb and R. Calderbank, New Codes and Inner Bounds for Exact Repair in Distributed Storage Systems, IEEE International Symposium on Information Theory (ISIT), July 2014.

Page generated 2015-08-24 01:10:46 Central Daylight Time, by jemdoc.

Part II



Network Coding Problem



Index Coding Problem

Index Code

Index Coding

Wants: X_1
Has: X_2 X_3

t_1

t_4

Wants: X_4
Has: X_1

X_1 X_2 X_3 X_4

t_2

Wants: X_2
Has: X_1 X_3

t_3

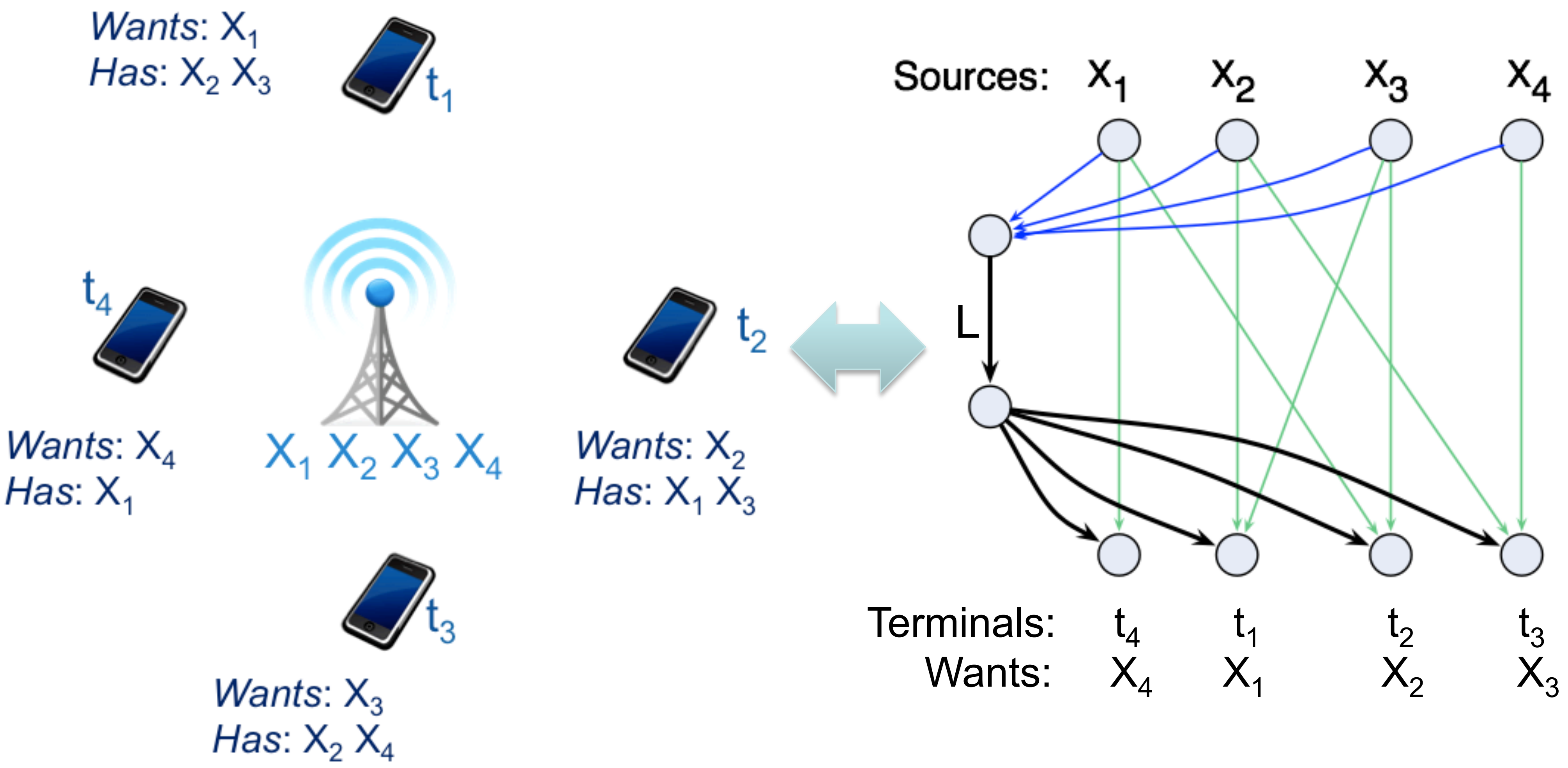
Wants: X_3
Has: X_2 X_4

Trans-mission #	Index code 1	Index code2
1	X_1	X_1+X_2
2	X_2	X_3
3	X_3	X_4
4	X_4	

$L=4$ $L=3$

Informed-source coding-on-demand [Birk & Kol infocom'98]

Equivalence to Network Coding

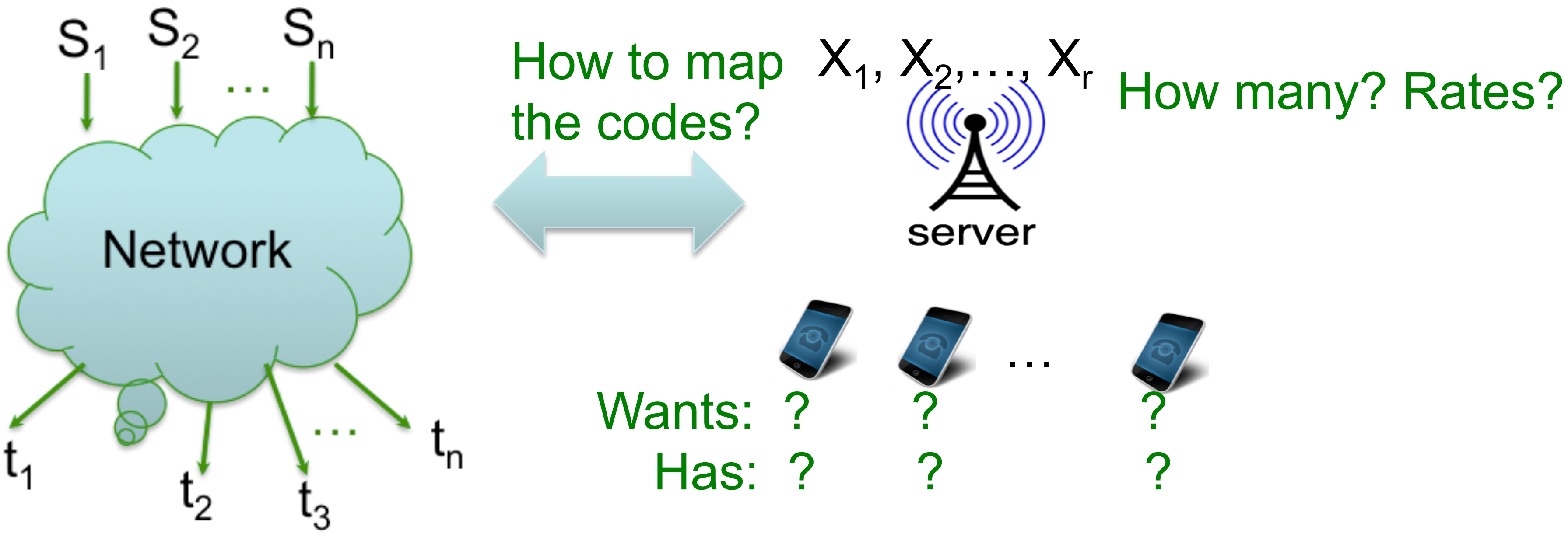


An index code of length L that satisfies all the users ↔ **A network code that satisfies all the terminals**

Stating the Equivalence

Theorem: [R,Sprintson, Georghiades'08] [Effros,R,Langberg ISIT'13]

For any network coding problem, one can construct an index coding problem and an integer L such that given any ~~linear~~ network code, one can efficiently construct a ~~linear~~ index code of length L, and vice versa. (same block length, same error probability).

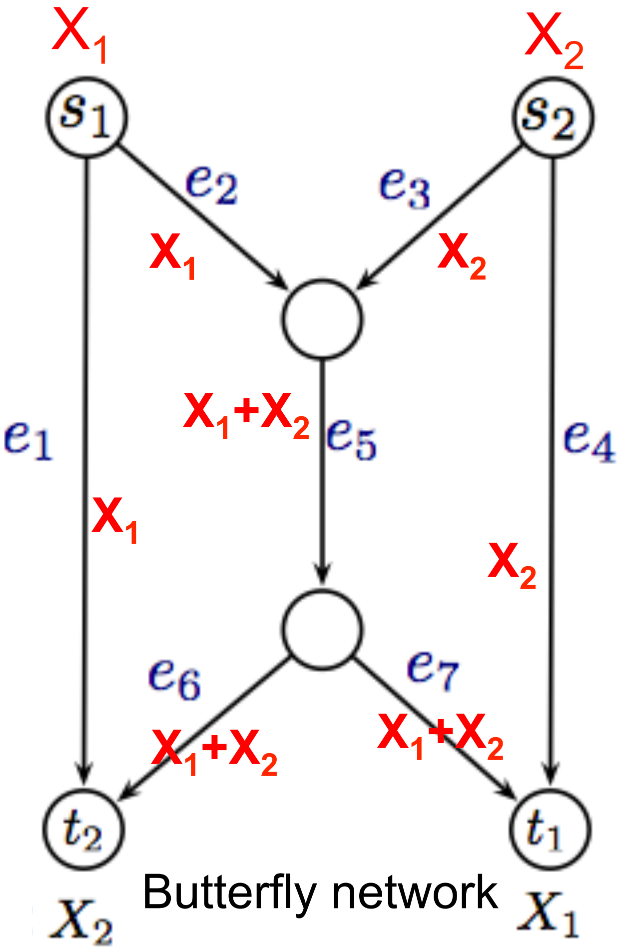


How to map the codes?

How many? Rates?

Wants: ? ? ?
Has: ? ? ?

Network Code → Index Code
The linear case first



Butterfly network

$Y_{e1}+X_1$

$Y_{e2}+X_1$

$Y_{e3}+X_2$

$Y_{e4}+X_2$

$\boxed{Y_{e5}}+X_1+X_2$

$Y_{e6}+X_1+X_2$

$Y_{e7}+X_1+X_2$

server

X_1, X_2

$Y_{e1}, Y_{e2}, \ldots, Y_{e7}$

$H(Y_{ei})=c(e_i)=1$

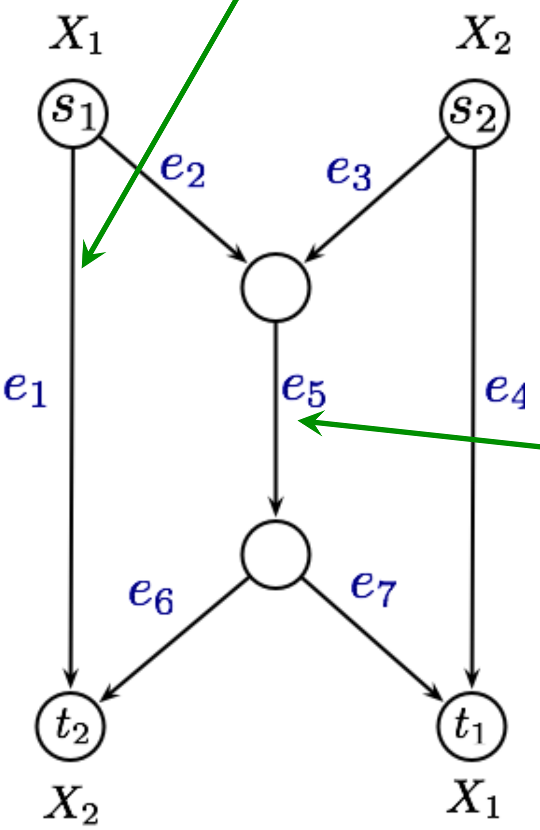
Terminal	Wants	Has
U_{e_1}	Y_{e_1}	X_1
U_{e_2}	Y_{e_2}	X_1
U_{e_3}	Y_{e_3}	X_2
U_{e_4}	Y_{e_4}	X_2
U_{e_5}	Y_{e_5}	Y_{e_2} Y_{e_3}

Terminal	Wants	Has
U_{e_6}	Y_{e_6}	Y_{e_5}
U_{e_7}	Y_{e_7}	Y_{e_5}

Equivalent index code

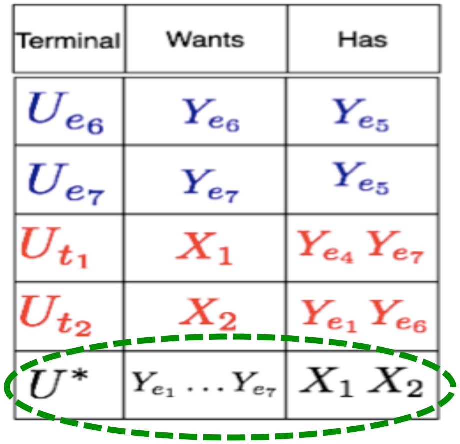
- All terminals in the index coding problem can decode
- Any linear network code gives an index code of length L=7

Index Code → Network Code



Butterfly network

$Y_{e1}+X_1$
$Y_{e2}+X_1$
$Y_{e3}+X_2$
$Y_{e4}+X_2$
$Y_{e5}+X_1+X_2$
$Y_{e6}+X_1+X_2$
$Y_{e7}+X_1+X_2$



Terminal	Wants	Has
U_{e_6}	Y_{e_6}	Y_{e_5}
U_{e_7}	Y_{e_7}	Y_{e_5}
U_{t_1}	X_1	$Y_{e_4}\ Y_{e_7}$
U_{t_2}	X_2	$Y_{e_1}\ Y_{e_6}$
U^*	$Y_{e_1} \ldots Y_{e_7}$	$X_1\ X_2$

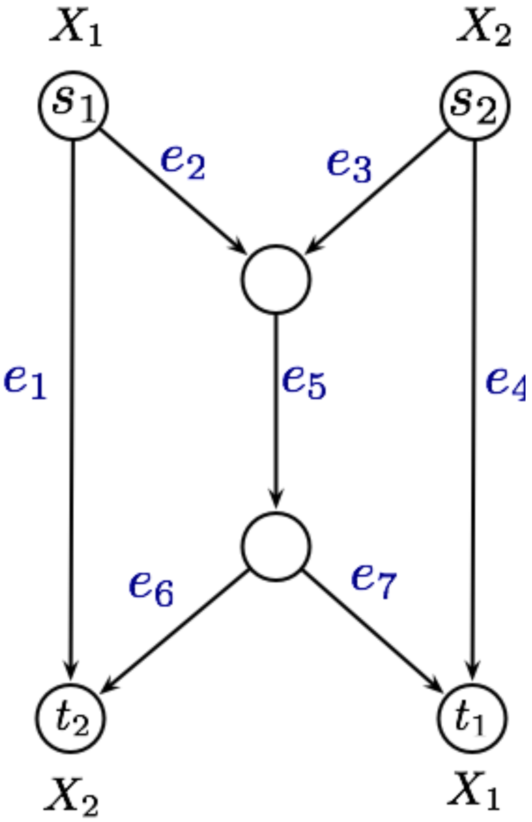
Given a linear index code

$Y_{e1}+Y_{e2}$
$Y_{e2}+X_1$
$Y_{e3}+X_2$
$Y_{e4}+X_2$
$Y_{e5}+Y_{e4}+X_1$
$Y_{e6}+X_1+X_2$
$Y_{e6}+Y_{e7}$

Can always diagonalize

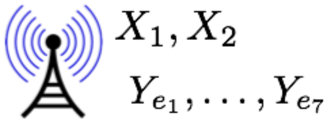
- Any linear index code of length L=7 can be mapped to a linear network code
- Works for scalar linear and vector linear

Non-Linear Network Code → Index Code EASY



Butterfly network

$$Y_{e_1} + f_{e_1}(X_1, X_2)$$
$$Y_{e_2} + f_{e_2}(X_1, X_2)$$
$$Y_{e_3} + f_{e_3}(X_1, X_2)$$
$$Y_{e_4} + f_{e_4}(X_1, X_2)$$
$$Y_{e_5} + f_{e_5}(X_1, X_2)$$
$$Y_{e_6} + f_{e_6}(X_1, X_2)$$
$$Y_{e_7} + f_{e_7}(X_1, X_2)$$



Terminal	Wants	Has
U_{e_1}	Y_{e_1}	X_1
U_{e_2}	Y_{e_2}	X_1
U_{e_3}	Y_{e_3}	X_2
U_{e_4}	Y_{e_4}	X_2
U_{e_5}	Y_{e_5}	Y_{e_2} Y_{e_3}

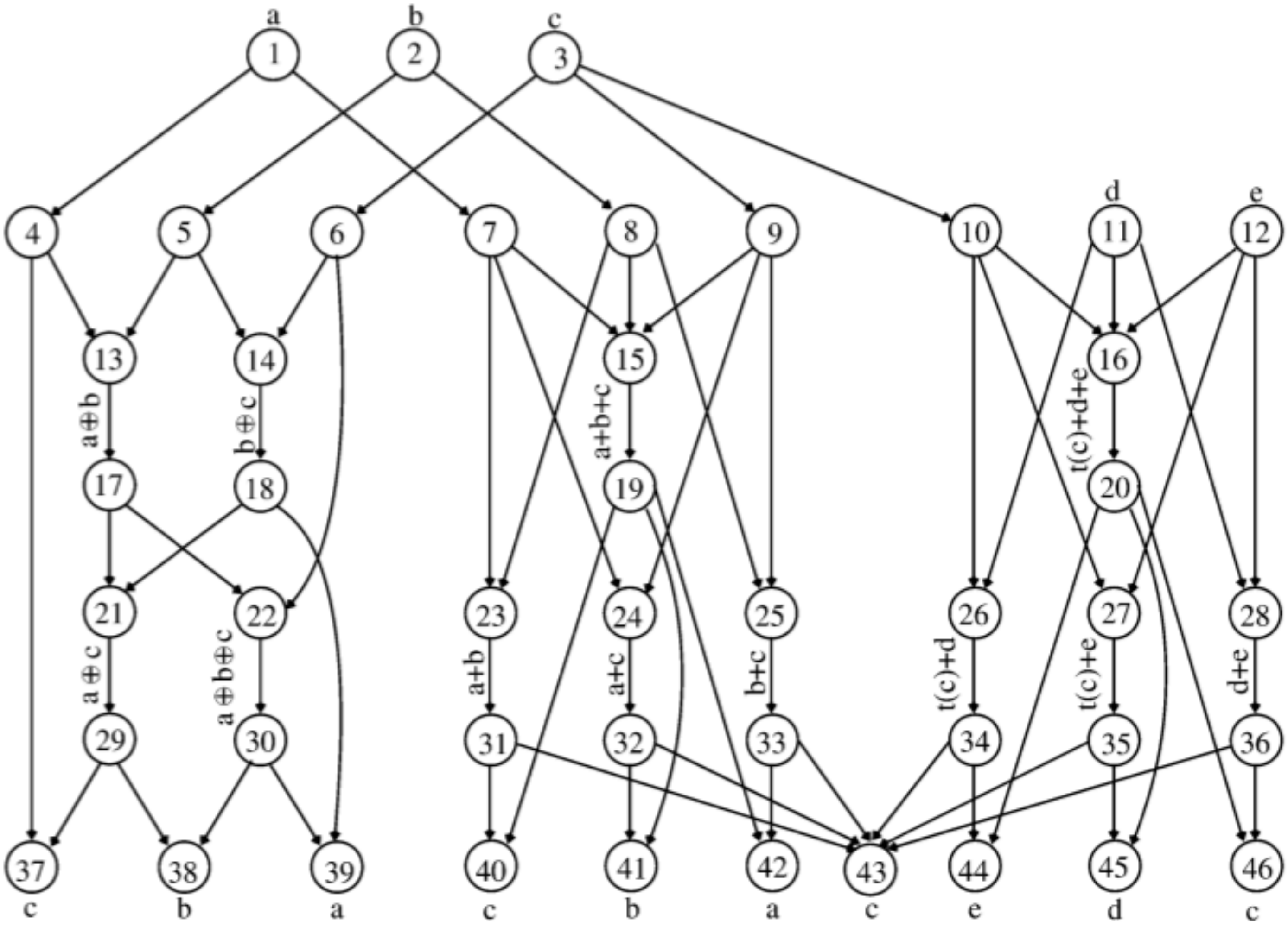
Terminal	Wants	Has
U_{e_6}	Y_{e_6}	Y_{e_5}
U_{e_7}	Y_{e_7}	Y_{e_5}
U_{t_1}	X_1	Y_{e_4} Y_{e_7}
U_{t_2}	X_2	Y_{e_1} Y_{e_6}
U^*	$Y_{e_1} \ldots Y_{e_7}$	X_1 X_2

Equivalent index code

$f_{e_i}(X_1, X_2)$: message on edge e_i

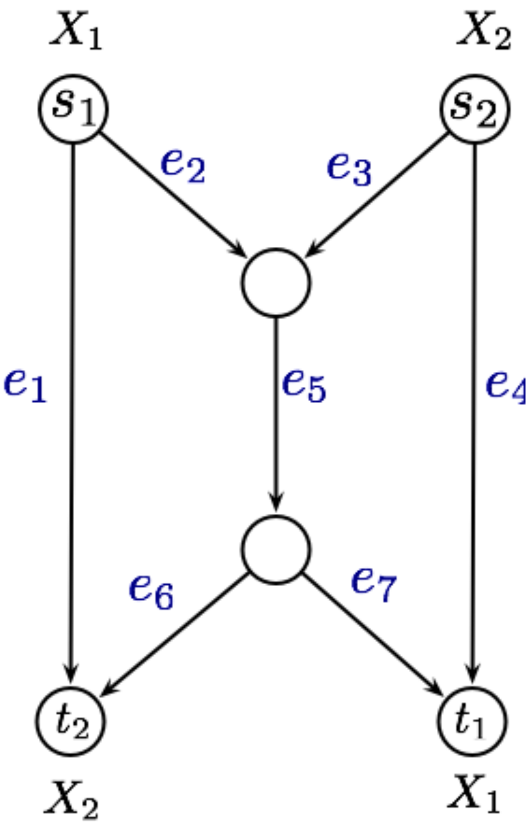
Implications: Scalar vs. Vector Linear

Scalar linear index codes are not optimal



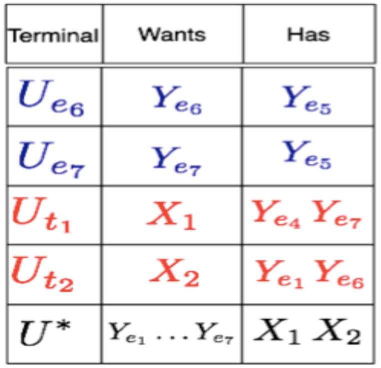
[Dougherty, Freiling, Zeger, '05]

Non-linear Index Codes -> Network Codes HARD!!!



Butterfly network

$$B'_1 = g'_1(Y_{e_1}, \bar{X})$$
$$B'_2 = g'_2(Y_{e_2}, \bar{X})$$
$$B'_3 = g'_3(Y_{e_3}, \bar{X})$$
$$B'_4 = g'_4(Y_{e_4}, \bar{X})$$
$$B'_5 = g'_5(Y_{e_5}, \bar{X})$$
$$B'_6 = g'_6(Y_{e_6}, \bar{X})$$
$$B'_7 = g'_7(Y_{e_7}, \bar{X})$$



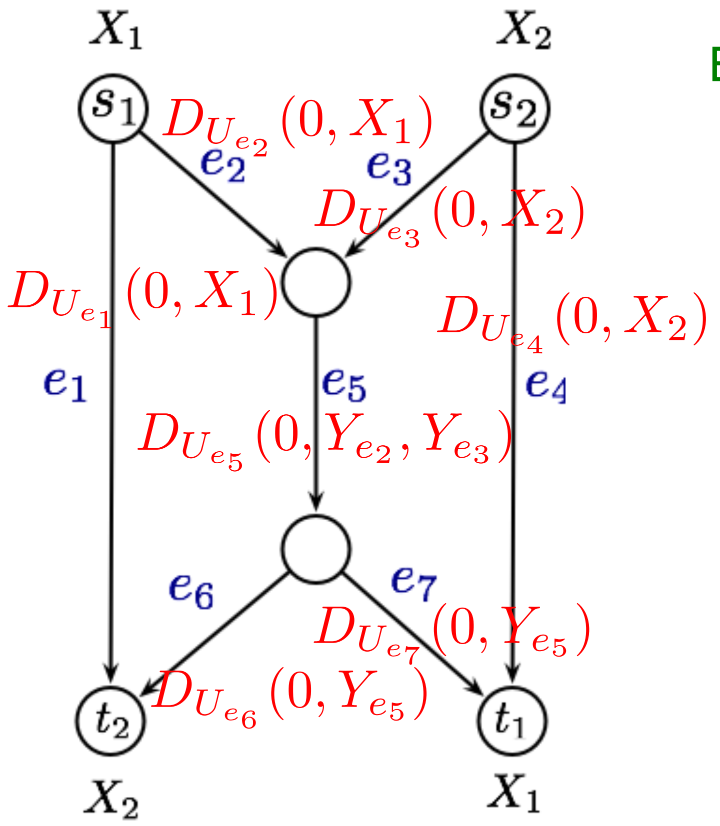
Terminal	Wants	Has
U_{e_6}	Y_{e_6}	Y_{e_5}
U_{e_7}	Y_{e_7}	Y_{e_5}
U_{t_1}	X_1	$Y_{e_4}\, Y_{e_7}$
U_{t_2}	X_2	$Y_{e_1}\, Y_{e_6}$
U^*	$Y_{e_1} \ldots Y_{e_7}$	$X_1\, X_2$

Given a non-linear index code

$$B_1 = g_1(\bar{Y}_e, \bar{X})$$
$$B_2 = g_2(\bar{Y}_e, \bar{X})$$
$$B_3 = g_3(\bar{Y}_e, \bar{X})$$
$$B_4 = g_4(\bar{Y}_e, \bar{X})$$
$$B_5 = g_5(\bar{Y}_e, \bar{X})$$
$$B_6 = g_6(\bar{Y}_e, \bar{X})$$
$$B_7 = g_7(\bar{Y}_e, \bar{X})$$

Cannot always Diagonalize

Non-linear Index Code → Network Code



Broadcast message

Decoding function

$$X1 = D_{U_{t1}}(B, Y_{e_4}, Y_{e_7})$$

$$Y_{e_4} = D_{U_{e_4}}(B, X_2)$$

$$Y_{e_7} = D_{U_{e_7}}(B, Y_{e_5})$$

Fix a value for B, say B=0

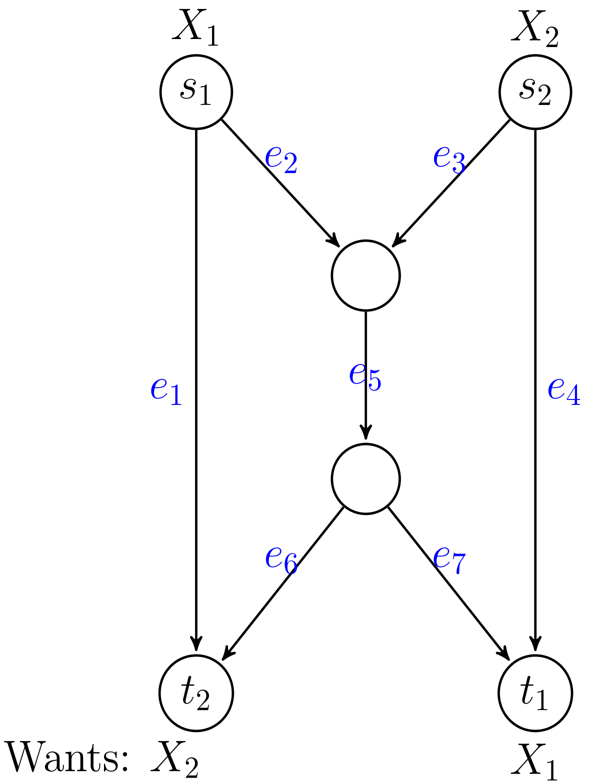
X_1, X_2

$Y_{e_1}, \ldots, Y_{e_7}$

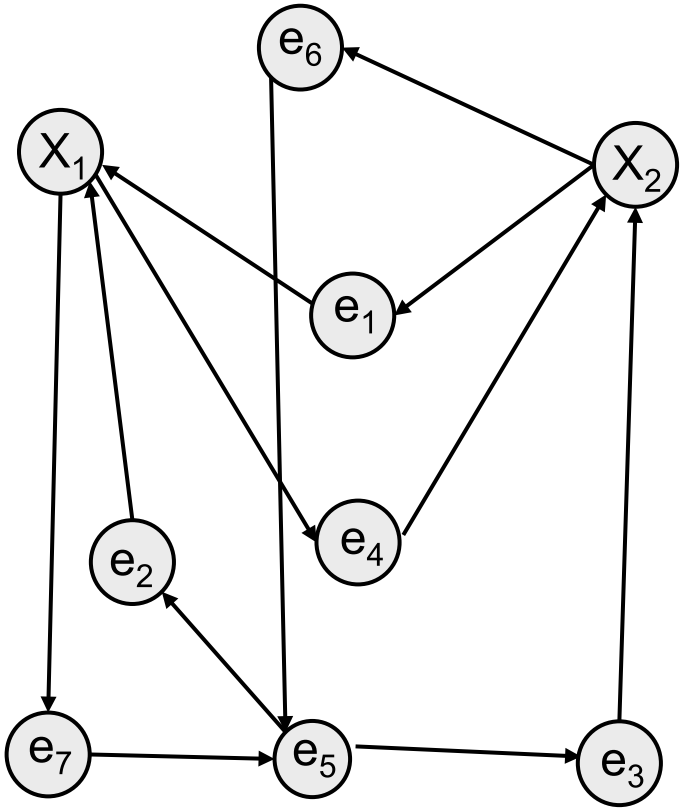
Terminal	Wants	Has
U_{e_1}	Y_{e_1}	X_1
U_{e_2}	Y_{e_2}	X_1
U_{e_3}	Y_{e_3}	X_2
U_{e_4}	Y_{e_4}	X_2
U_{e_5}	Y_{e_5}	$Y_{e_2}\ Y_{e_3}$
U_{e_6}	Y_{e_6}	Y_{e_5}
U_{e_7}	Y_{e_7}	Y_{e_5}
U_{t_1}	X_1	$Y_{e_4}\ Y_{e_7}$
U_{t_2}	X_2	$Y_{e_1}\ Y_{e_6}$
U^*	$Y_{e_1} \ldots Y_{e_7}$	$X_1\ X_2$

- Destinations can decode with no errors:
- Recall that $B=f(X_1,X_2, Y_{e1},\ldots,Y_{e7})$
- For a fixed B and given values of X_1 and X_2, there is a unique possible vector $(Y_{e1},\ldots,Y_{e7})$
- Otherwise, U* cannot decode correctly

Part III



Network Coding Problem



Index Coding Problem

Index Code

Index Coding

Wants: X_1
Has: X_2 X_3

Wants: X_4
Has: X_1

X_1 X_2 X_3 X_4

Wants: X_2
Has: X_1 X_3

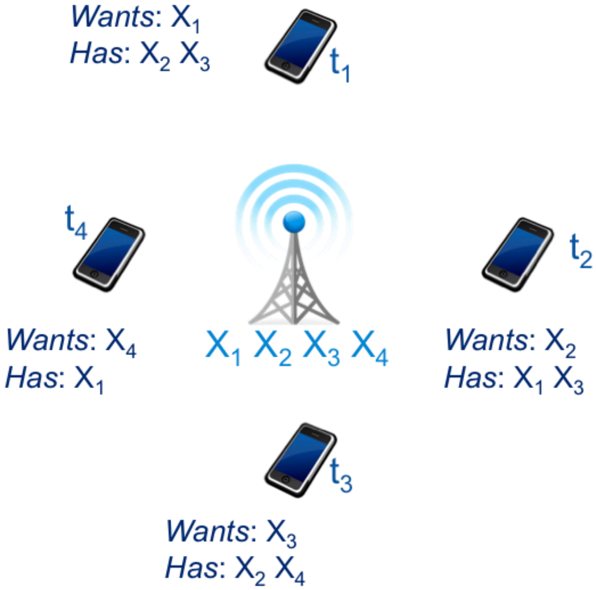
Wants: X_3
Has: X_2 X_4

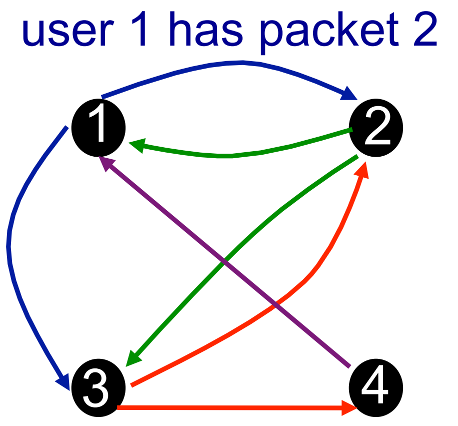
Trans-mission #	Index code 1	Index code2
1	X_1	X_1+X_2
2	X_2	X_3
3	X_3	X_4
4	X_4	

$L=4$ $L=3$

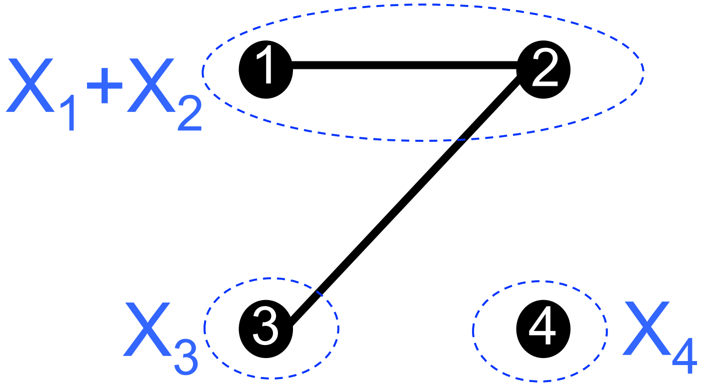
Informed-source coding-on-demand [Birk & Kol infocom’98]

Index Coding & Graph Coloring





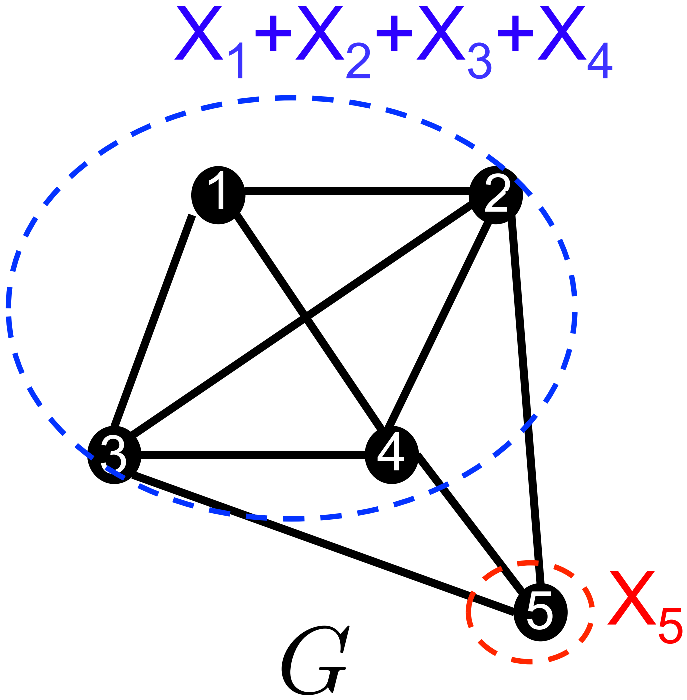
Side info graph G_d

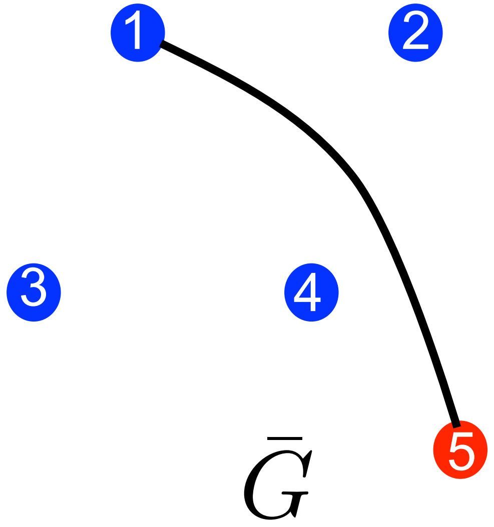


Clique cover of G

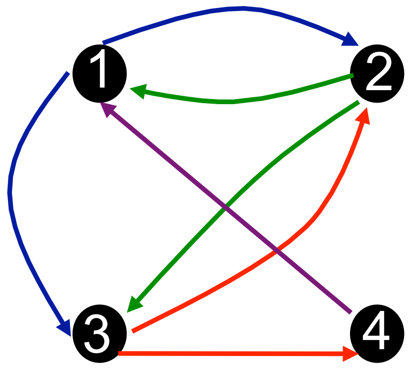
=

Chromatic nbr of $\bar{G}$

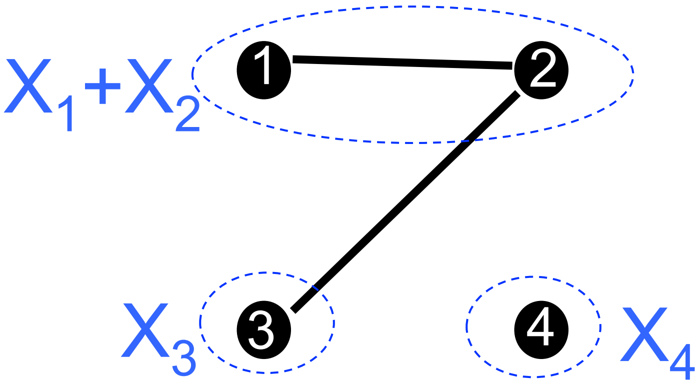




Index Coding & Graph Coloring



Side info graph G_d



Chromatic nbr of $\bar{G}$

Independence nbr

$$\alpha(G_d) \leq c(G_d) \leq L^*_{min} \leq \chi_f(\bar{G}) \leq \chi(\bar{G})$$

Shannon capacity
[Haemers '79]

Fractional Chromatic nbr
[Blasiak et al.'11]

$$\leq \chi_{f\ell}(G)$$

Fractional local chrom. nbr
[Shanmugan et al. '13]

[Alon et al., '08]

[Maleki, Cadambe, Jafar '12]

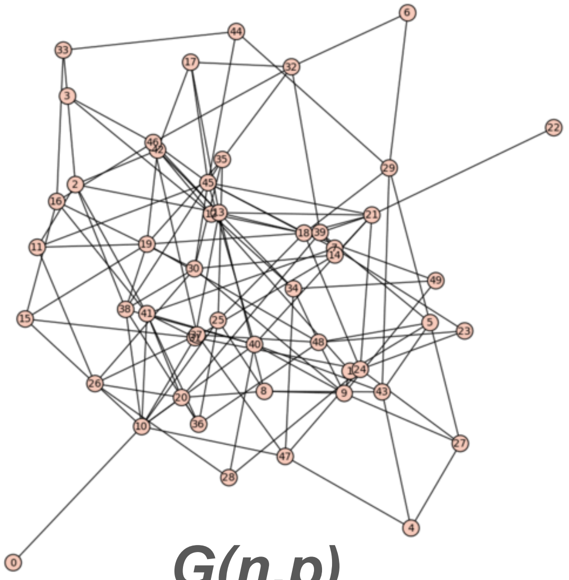
[Arbabjolfaei et al., '13]

…

Index Coding on Erdős-Rényi Graphs

Independence nbr Chromatic nbr

$$\alpha(G) \leq L^*_{min} \leq \chi(\bar{G})$$



G(n,p)

- When $n \to \infty$, we have with prob 1

$$\log n \leq L^*_{min} \leq \frac{n}{\log n}$$

- Can improve the lower bound [Haviv & Langberg '11]

$$c\sqrt{n} \leq L^*_{min} \leq \frac{n}{\log n}$$

- Coring is the best upper bound we know on random graphs. Is it tight? OPEN

Index Coding & Rank Minimization

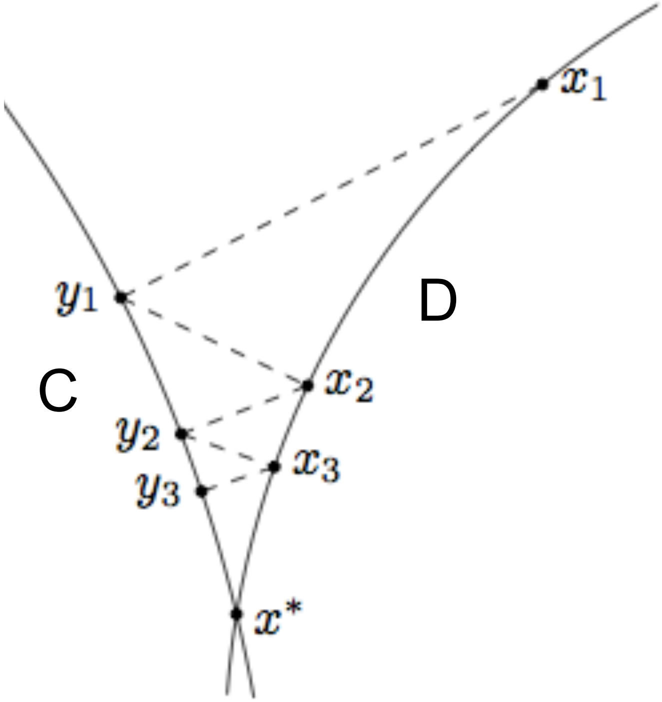
$$\begin{array}{c|c|c|c|c|}
 & X_1 & X_2 & X_3 & X_4 \\ \hline
t_1 & 1 & * & * & 0 \\ \hline
t_2 & * & 1 & * & 0 \\ \hline
t_3 & 0 & * & 1 & * \\ \hline
t_4 & * & 0 & 0 & 1 \\ \hline
\end{array}$$

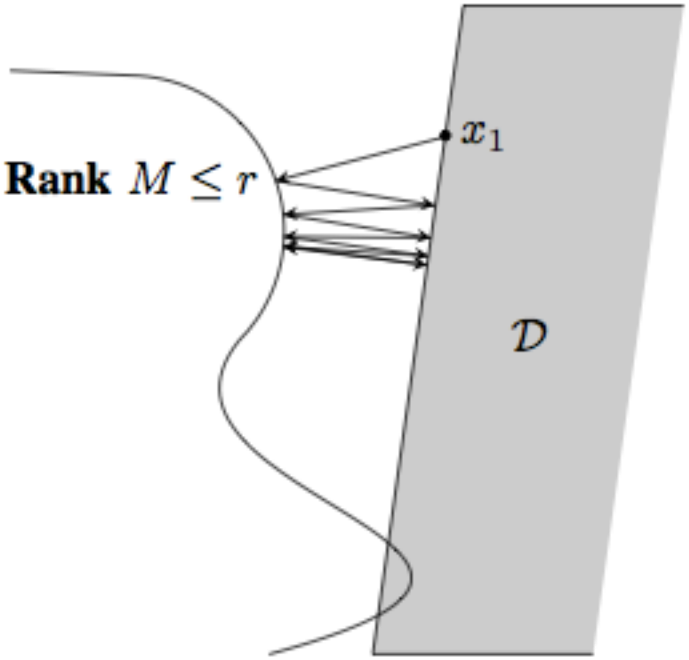
Matrix M

- Linear case: $L^*_{min} = \min rk(M)$ [Bar-Yossef et al. '06]
- Min rank introduced by Haemers in 79 to bound the Shannon graph capacity.
- Computing L^*_{min} is NP hard. [R. et al. '07] [Peeters '96]
- Recent work on matrix completion for index coding [Hassibi et al. '14]

Use Matrix Completion Methods to Construct Index Codes

- Min nuclear norm [Recht & Candes ‘09] does not work here
- Try alternative rank minimization methods [Fazel et al. 2001]





Index coding via AP

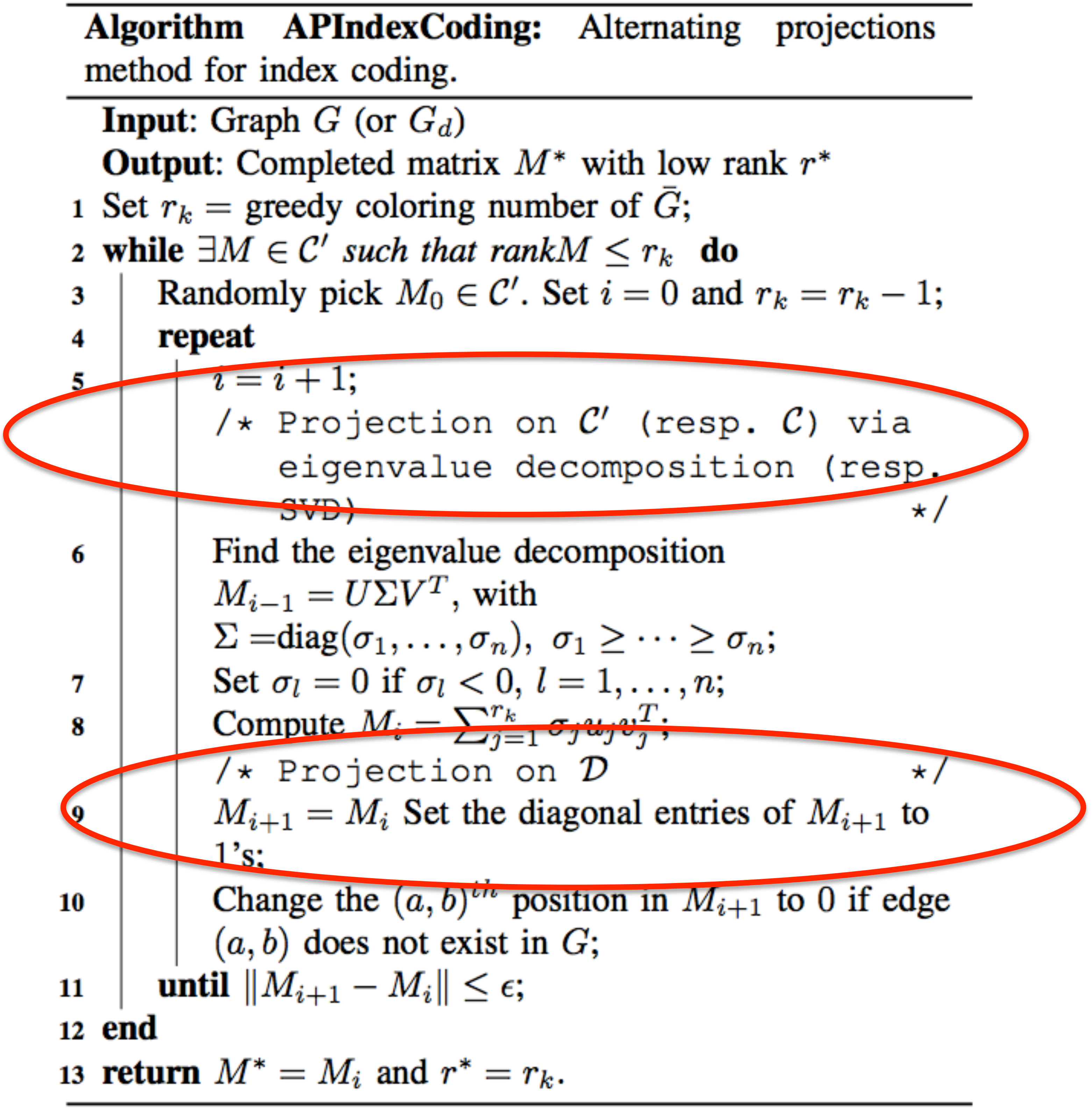
Two problems:

1) Regions not convex
2) Optimization over the reals

Network codes over the reals [Shwartz & Medard ’14], Jaggi et al. ‘08]

Theorem: [Alternating Projections (AP)]

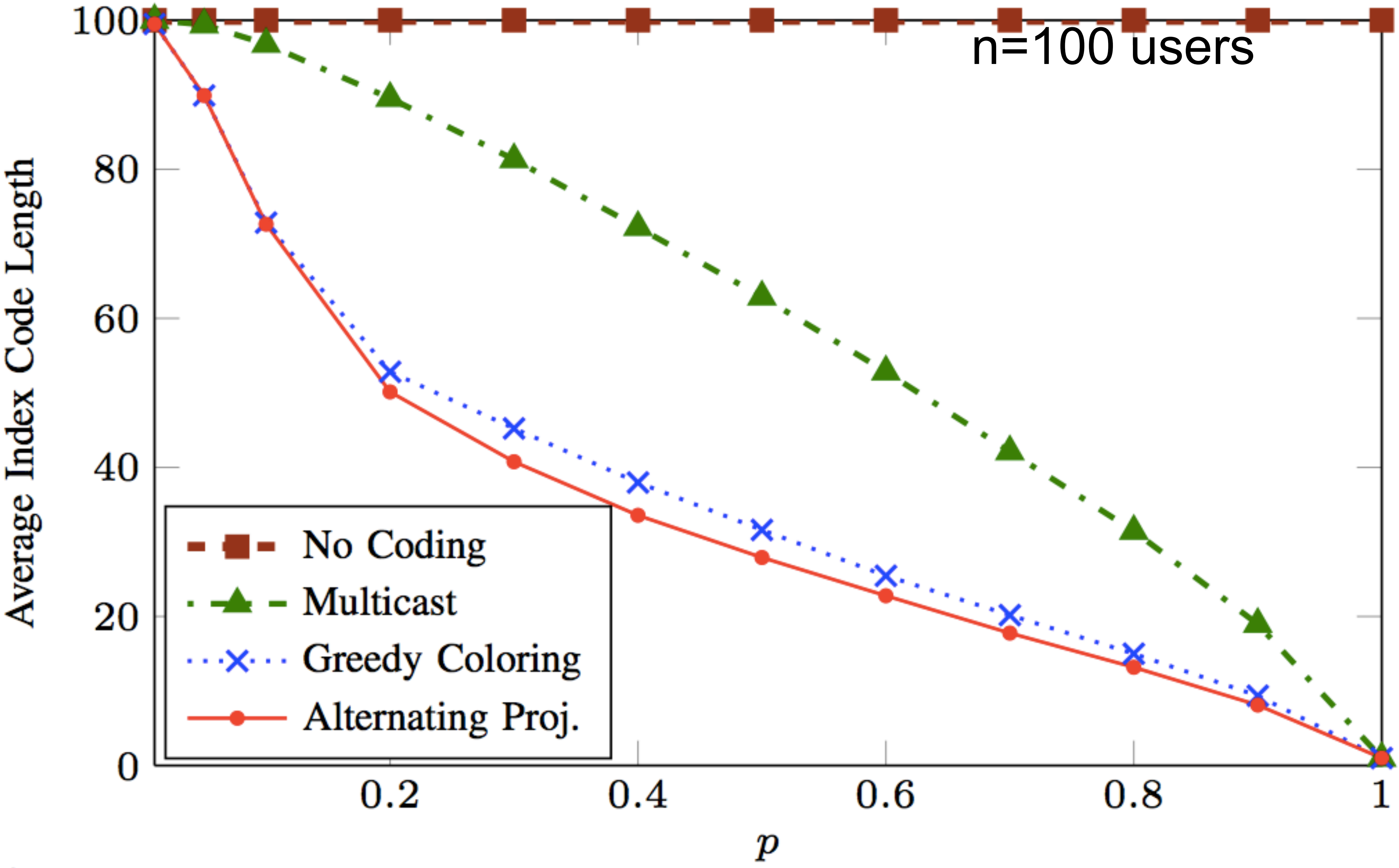
If C and D are convex, then an alternating projection sequence between these 2 regions converges to a point in their intersection.



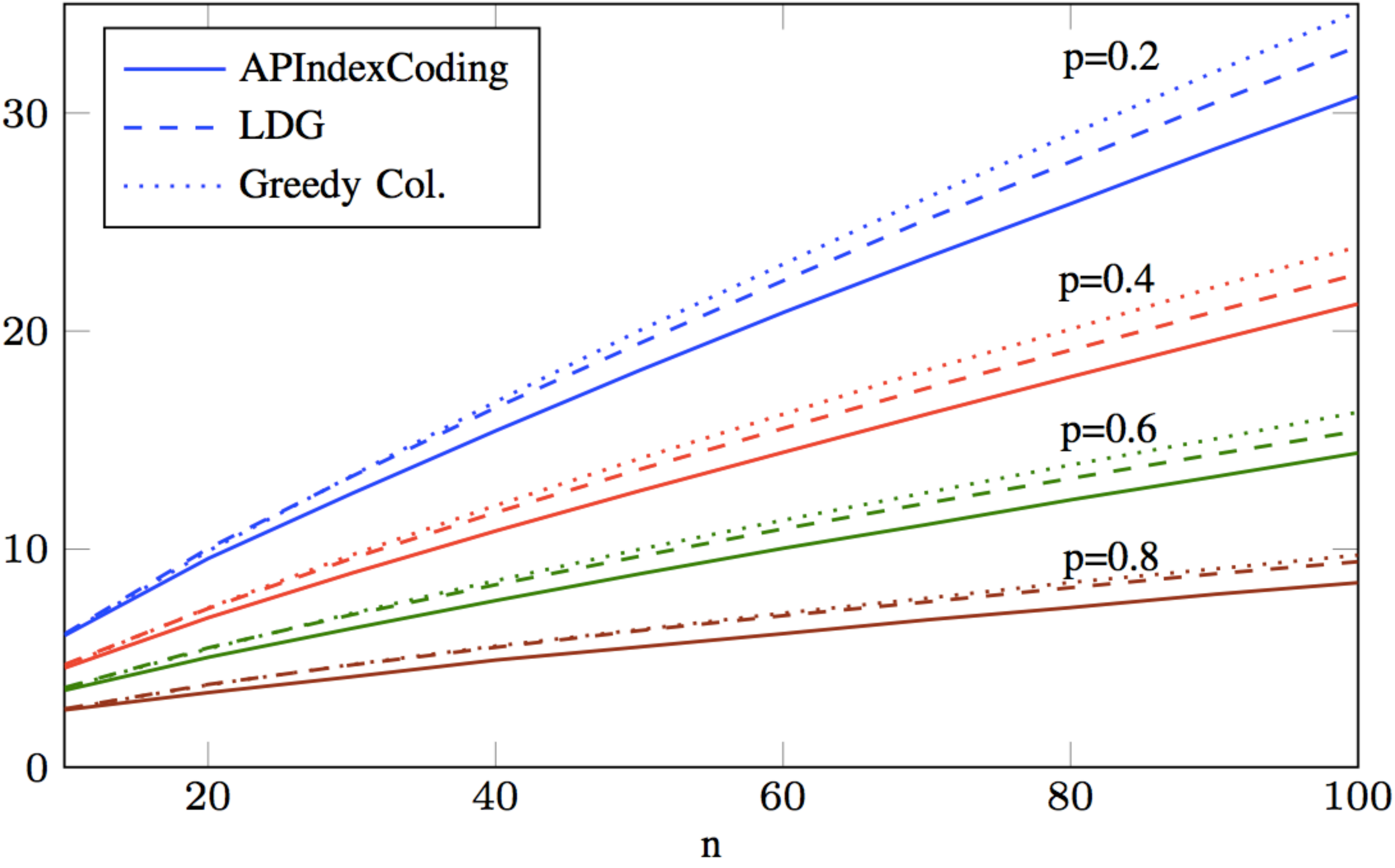
Algorithm APIndexCoding: Alternating projections method for index coding.

```
Input: Graph $G$ (or $G_d$)
Output: Completed matrix $M^*$ with low rank $r^*$
Set $r_k$ = greedy coloring number of $\bar{G}$;
while $\exists M \in \mathcal{C}'$ such that $\mathrm{rank} M \leq r_k$ do
    Randomly pick $M_0 \in \mathcal{C}'$. Set $i = 0$ and $r_k = r_k - 1$;
    repeat
        $i = i + 1$;
        /* Projection on $\mathcal{C}'$ (resp. $\mathcal{C}$) via
           eigenvalue decomposition (resp. SVD)                          */
        Find the eigenvalue decomposition
        $M_{i-1} = U \Sigma V^T$, with
        $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_n),\ \sigma_1 \geq \cdots \geq \sigma_n$;
        Set $\sigma_l = 0$ if $\sigma_l < 0$, $l = 1, \ldots, n$;
        Compute $M_i = \sum_{j=1}^{r_k} \sigma_j u_j v_j^T$;
        /* Projection on $\mathcal{D}$                                   */
        $M_{i+1} = M_i$ Set the diagonal entries of $M_{i+1}$ to
        1's;
        Change the $(a, b)^{th}$ position in $M_{i+1}$ to 0 if edge
        $(a, b)$ does not exist in $G$;
    until $\|M_{i+1} - M_i\| \leq \epsilon$;
end
return $M^* = M_i$ and $r^* = r_k$.
```

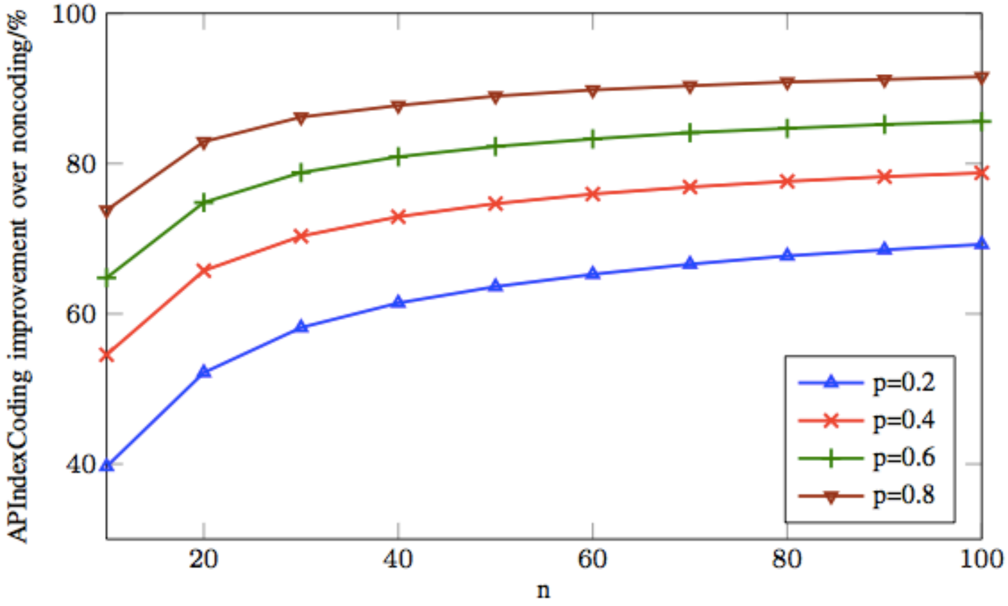
Index Coding via Alternating Proj on Random Undirected Graphs

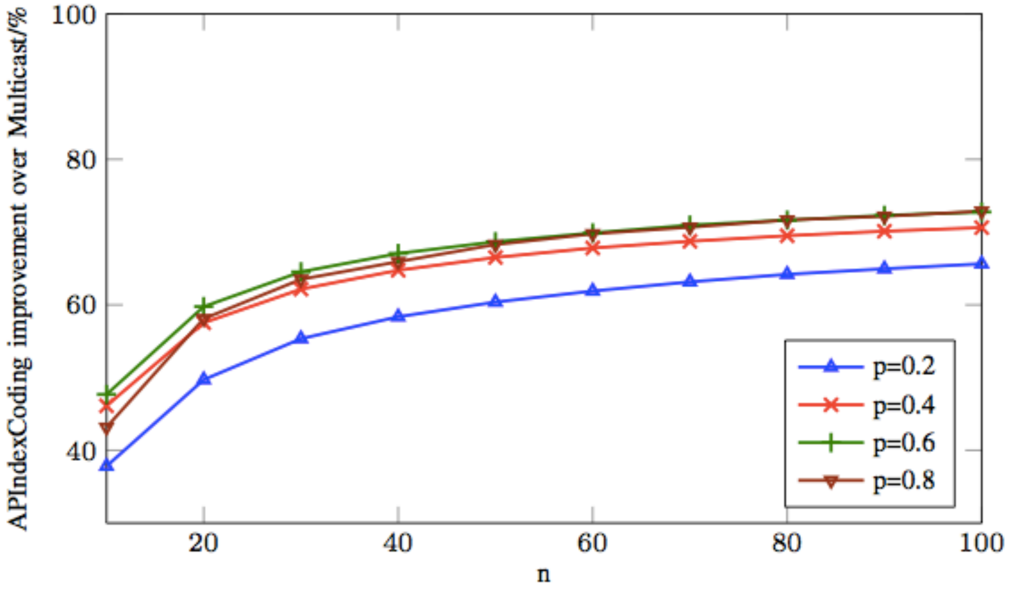


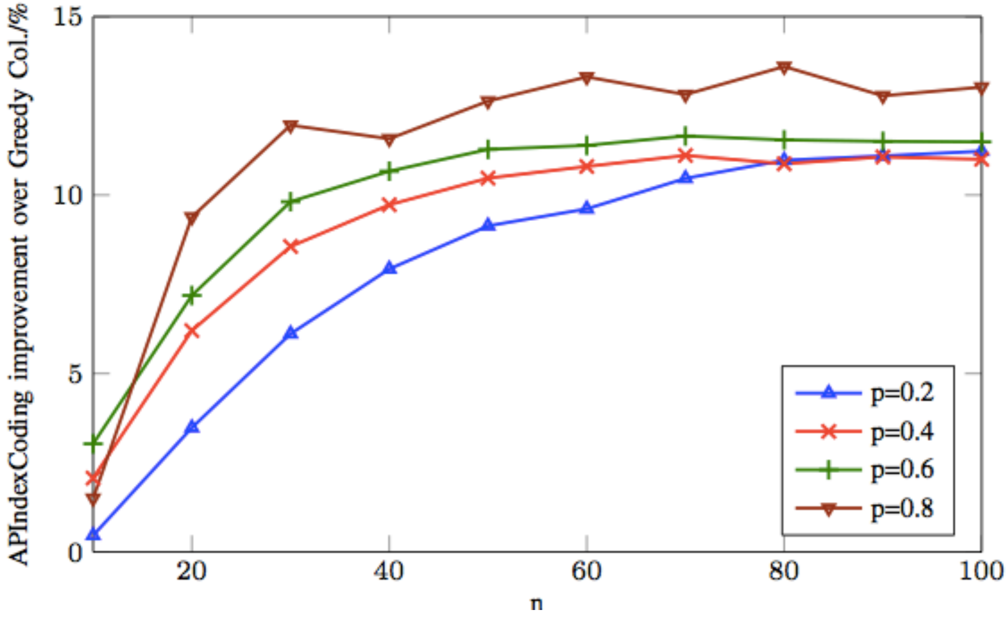
Performance with Increasing Number of Users

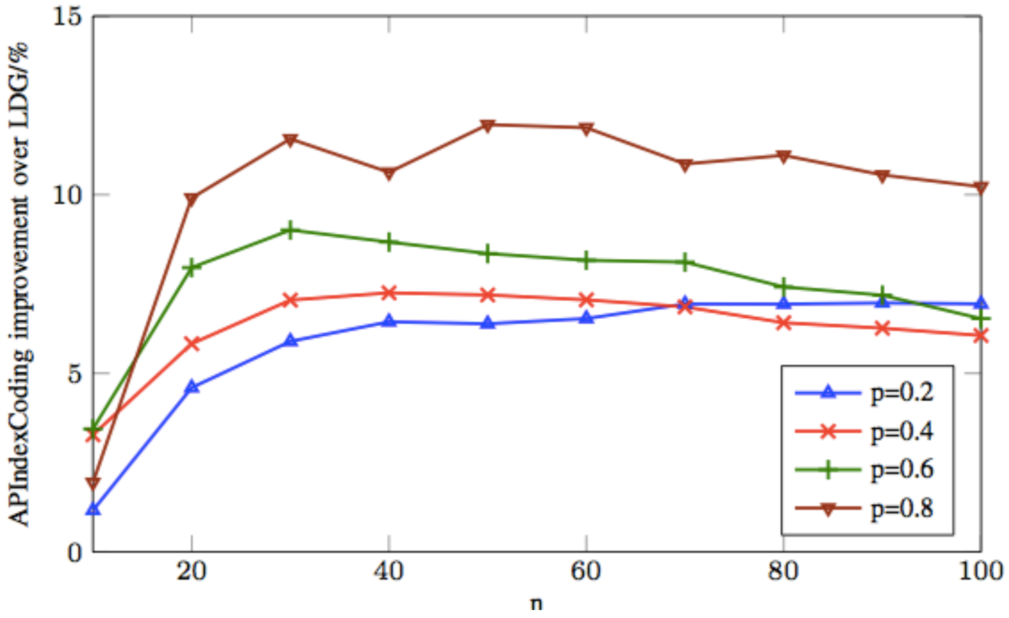


Improvement in Percentage

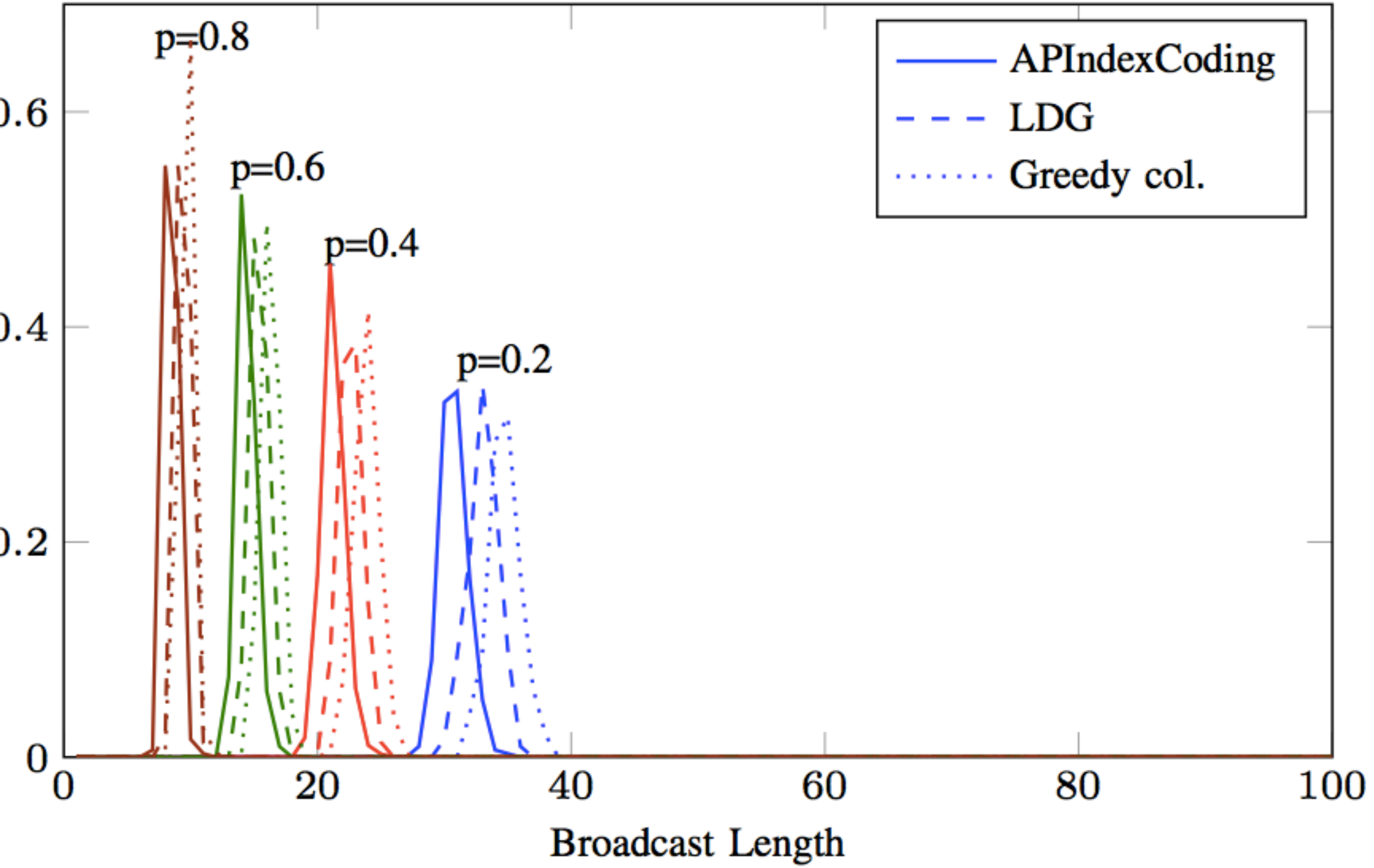




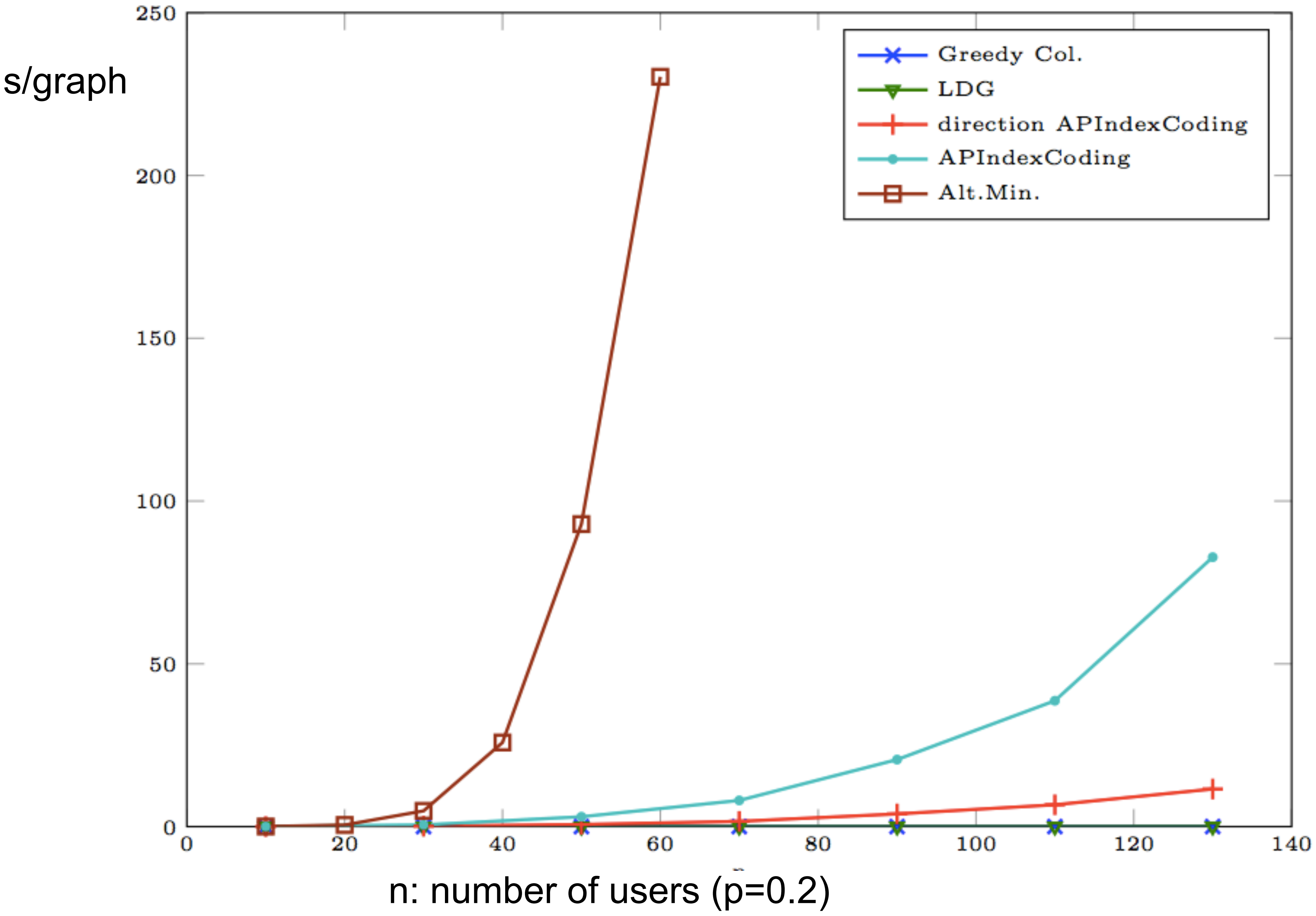




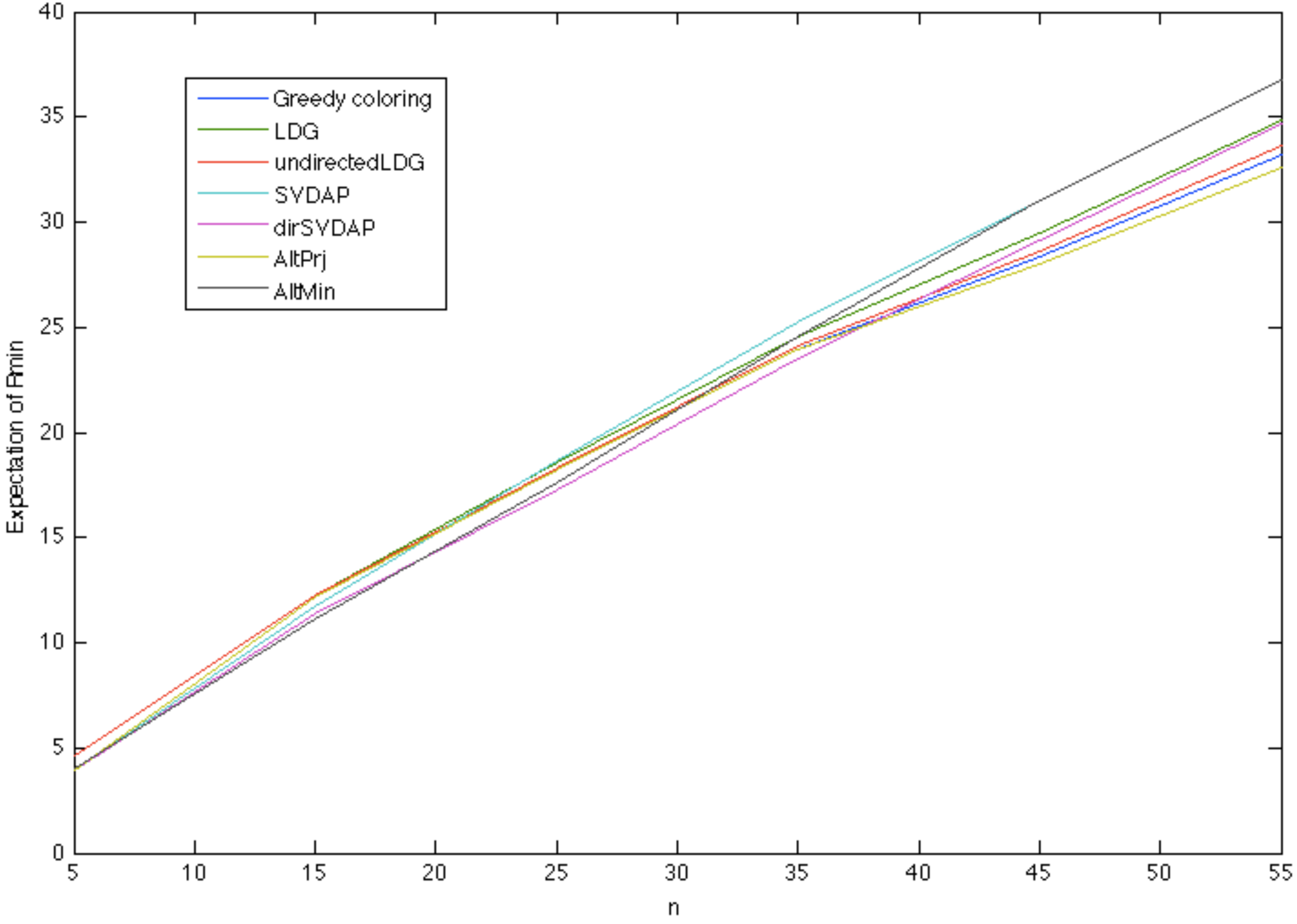
Concentration around the Average



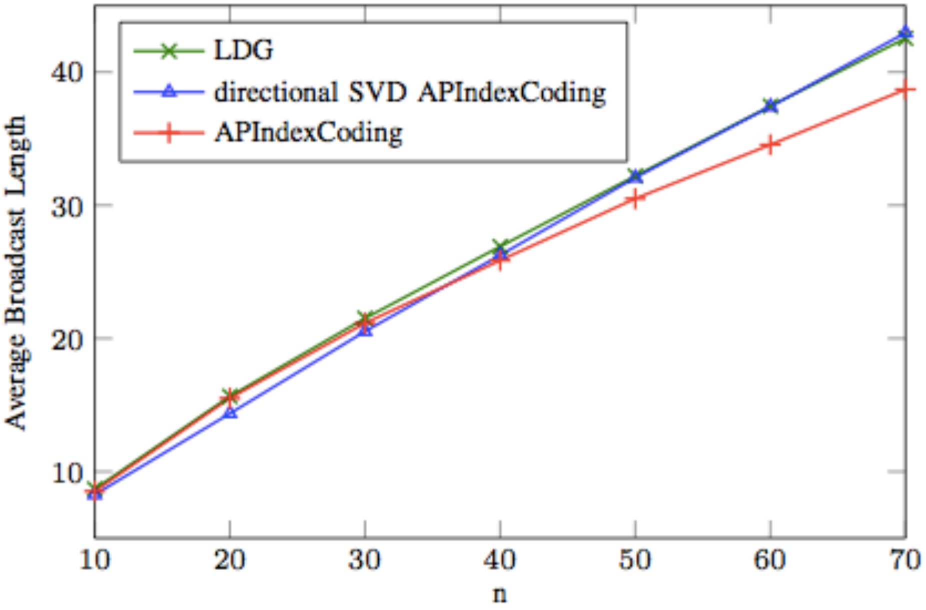
Running Time

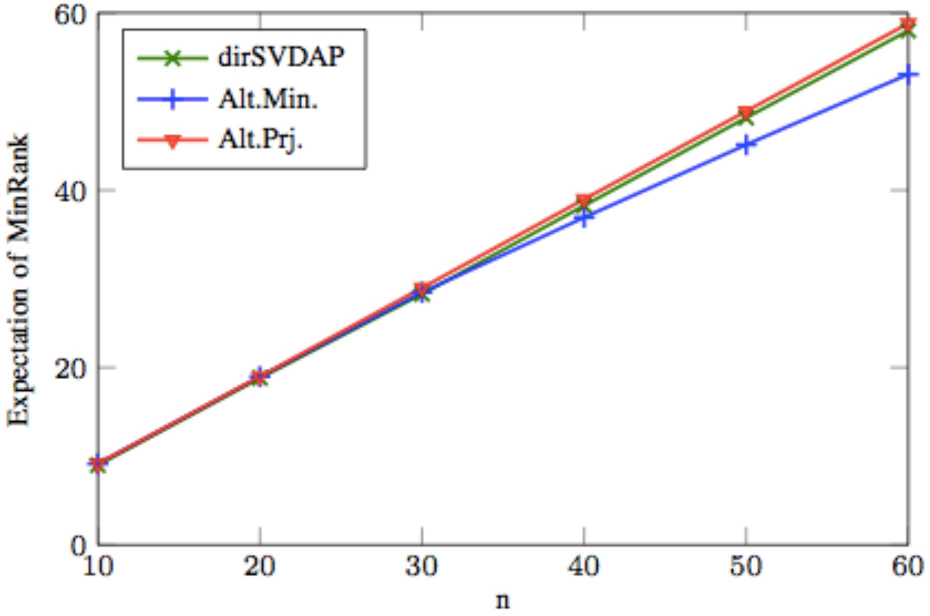


Random Directed Graphs



Which Method to use for Directed Graphs?





How close are these heuristics to the actual minimum

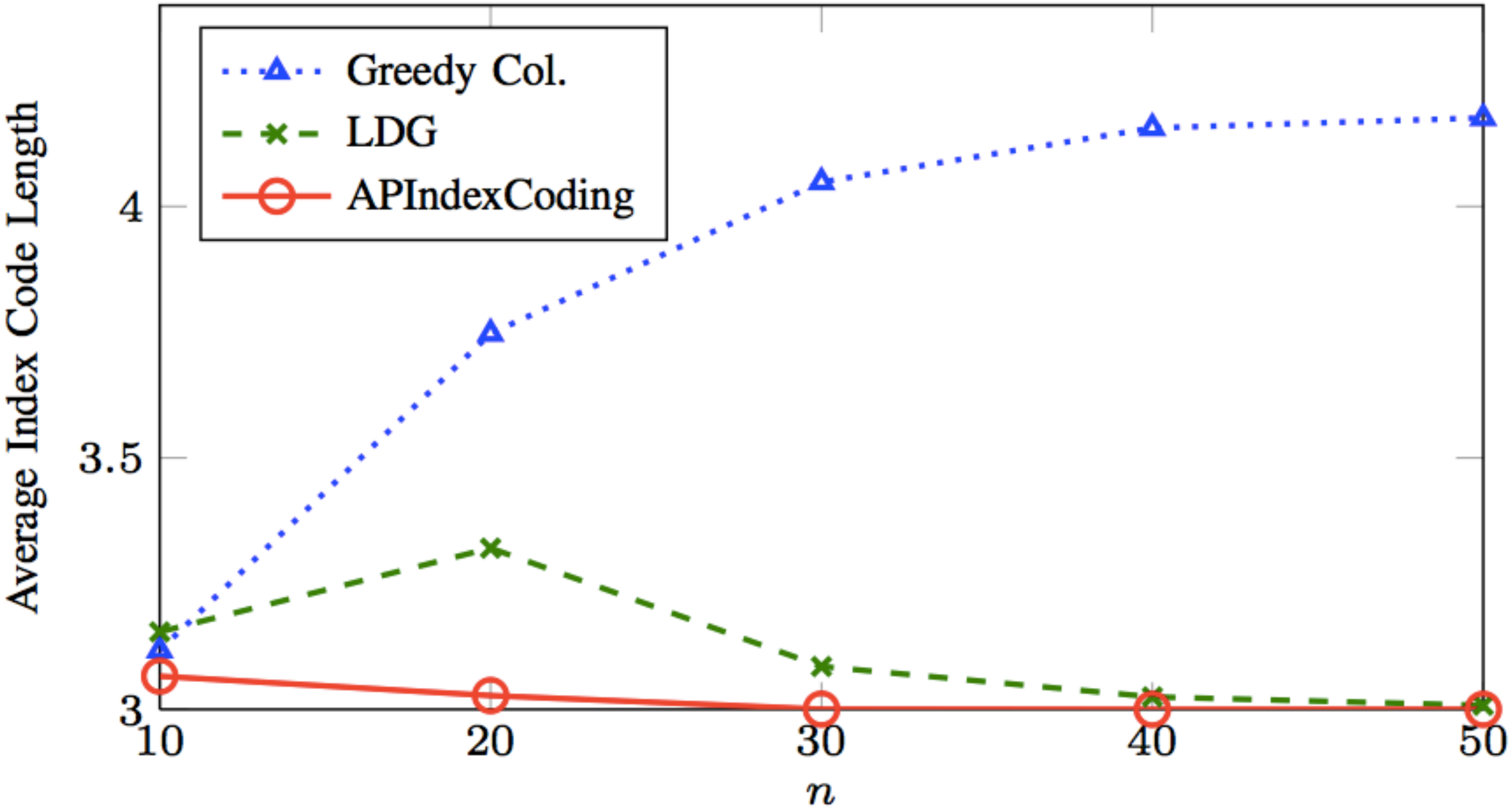


Fig. 9: Average index code length obtained by using Greedy Coloring, LDG and APIndexCoding for random 3-colorable graphs when $p = 0.5$.

- For n≤5, linear index coding achieve capacity [Ong,'14]. Online list of optimal index coding rates [kim]
- APindex coding was able to achieve all these rates whenever they are integers

Concluding Remarks

- Index coding is NP hard. But, this is not the end of the story.
- Proposed the use of different rank minimizations methods for constructing index codes
- Index coding is connected to many other interesting topic in the literature
- Many theoretical open questions: From reals to finite fields? theoretical guarantees? Index coding on random graphs? Need a stronger equivalence for equivalence of capacity regions….
- To do list for the matlab library
  - Vector linear network codes
  - Now only multiple unicast
  - Include more methods (heuristics) for index coding: Now only LDG and minrank
  - Network desgin…

QUESTIONS?

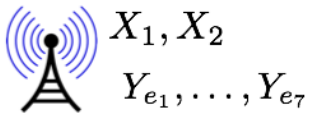
www.tinyurl.com/IndexCodingRocks

Dealing with Errors

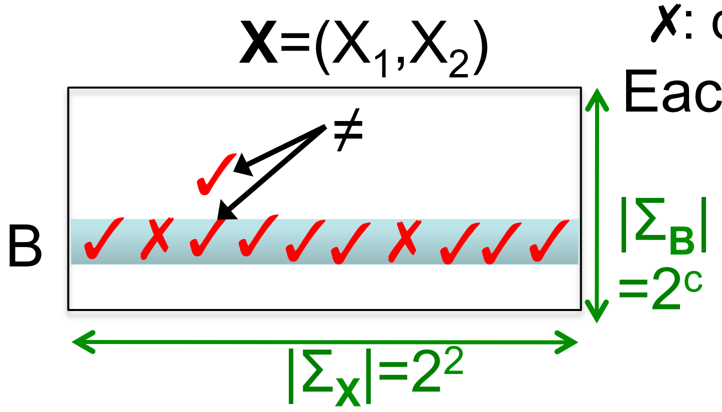
- Consider an index code where decoding errors only happen when the broadcast message B=0
- ε: Prob of error in the index code $=1/2^c=1/2^7=0.0078$
- Prob of error in the network code =1 (bad).

Claim: There exists σ, such that for B=σ, in the previous construction, the network code will have a prob of error at most ε (ε=error prob of the index code).

- Intuition: if for every value of B, the resulting network code will have a prob of error>ε, this implies that the prob of error in the index code >ε. A contradiction.



Terminal	Wants	Has
U_{e_1}	Y_{e_1}	X_1
U_{e_2}	Y_{e_2}	X_1
U_{e_3}	Y_{e_3}	X_2
U_{e_4}	Y_{e_4}	X_2
U_{e_5}	Y_{e_5}	$Y_{e_2}\ Y_{e_3}$
U_{e_6}	Y_{e_6}	Y_{e_5}
U_{e_7}	Y_{e_7}	Y_{e_5}
U_{t_1}	X_1	$Y_{e_4}\ Y_{e_7}$
U_{t_2}	X_2	$Y_{e_1}\ Y_{e_6}$
U^*	$Y_{e_1} \ldots Y_{e_7}$	$X_1\ X_2$



Ye=$D_{U*}(B, X_1, X_2)$

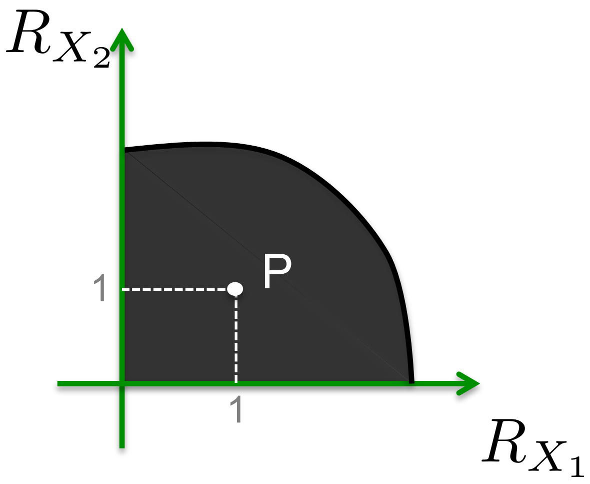
✗: decoding error

Each ✓ corresponds to a different “good” value of (**X**,**Ye**)

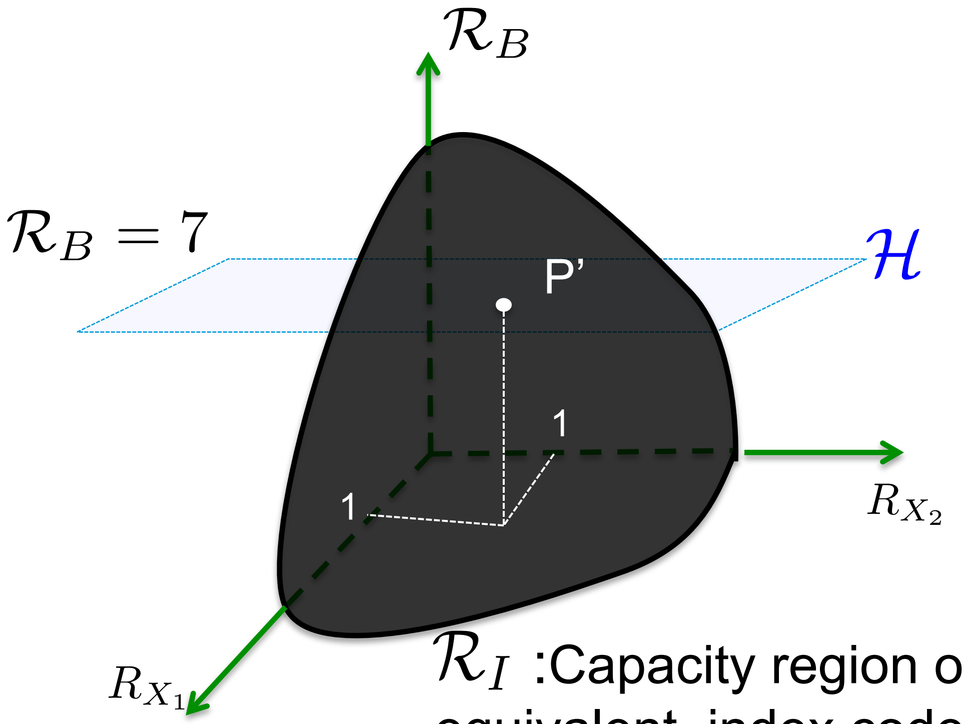
Total # of ✓$<(1-\varepsilon)|\Sigma_B|.|\Sigma_X|$
But $|\Sigma_B|=|\Sigma_{Ye}|$
➔Total # of “good” values$<(1-\varepsilon)|\Sigma_{Ye}|.|\Sigma_X|$

contradiction

Capacity Regions



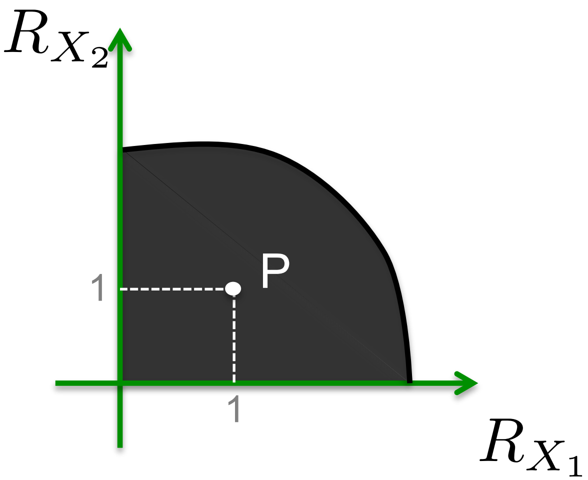
$\mathcal{R}_N$:Capacity region of a network



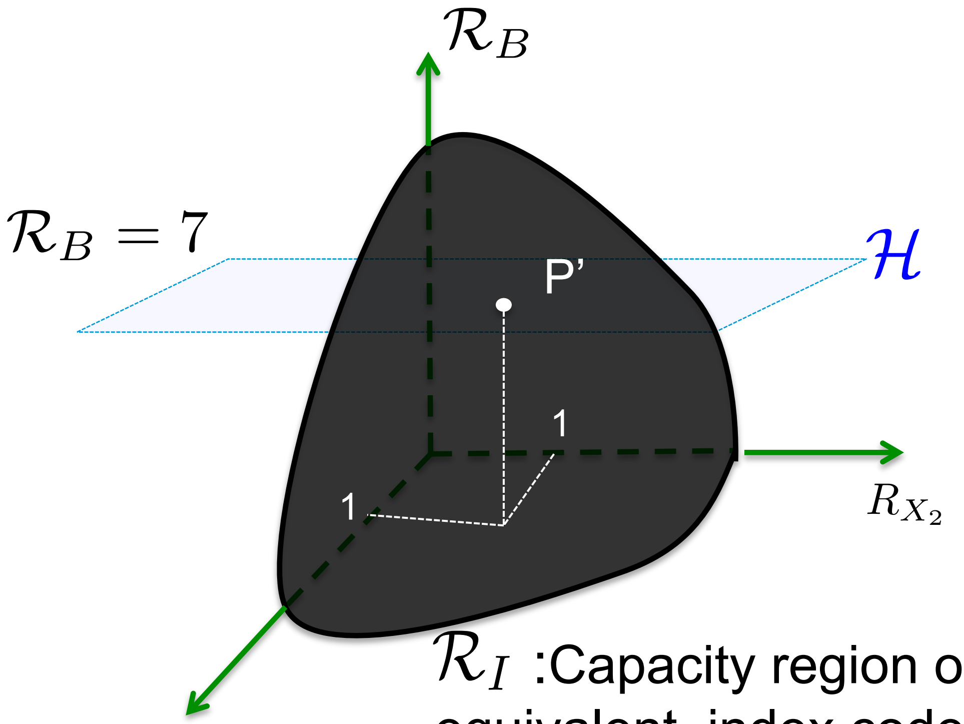
$\mathcal{R}_I$:Capacity region of the equivalent index code

- If there is a code that achieves P “exactly”, then P’ is in $\mathcal{R}_\mathcal{I} \cap \mathcal{H}$, and vice versa.
- What if a sequence of points (not necessarily in $\mathcal{H}$) converges to P. Does this mean that P is in $\mathcal{R}_N$?
- If true this will solve a long-standing open problem: Is zero-error capacity= ε-error capacity of networks?
- True for index coding problems [Langberg, Effros ‘11]

The Case of Co-located Sources



$\mathcal{R}_N$:Capacity region of a network



$\mathcal{R}_I$:Capacity region of the equivalent index code

Theorem: For any network $\mathcal{N}$ with co-located sources one can efficiently construct an index coding problem $\mathcal{I}$ and an integer L such that **R** is in the capacity region of $\mathcal{N}$ iff **R**’ is in the capacity region of $\mathcal{I}$ with broadcast length L.

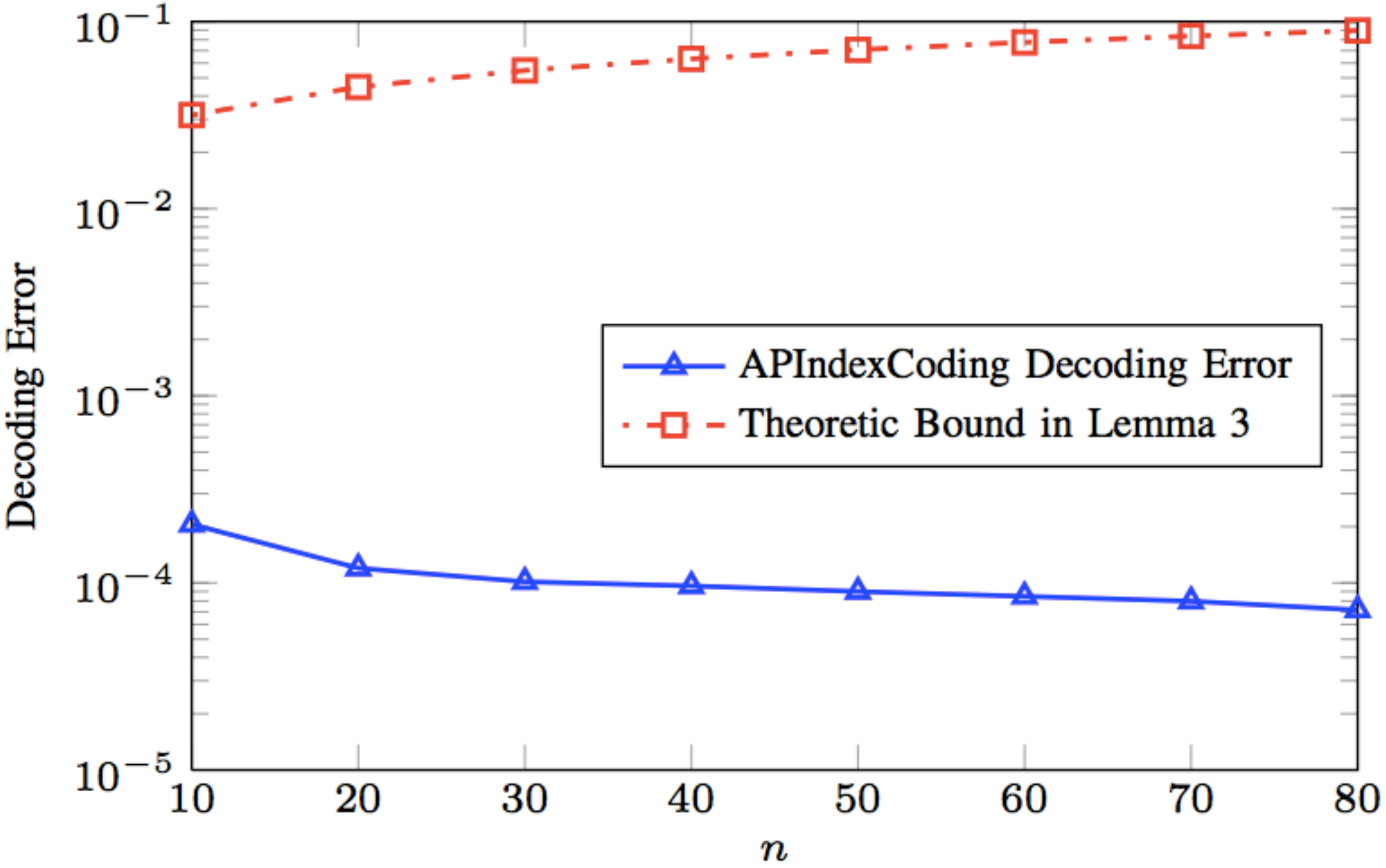
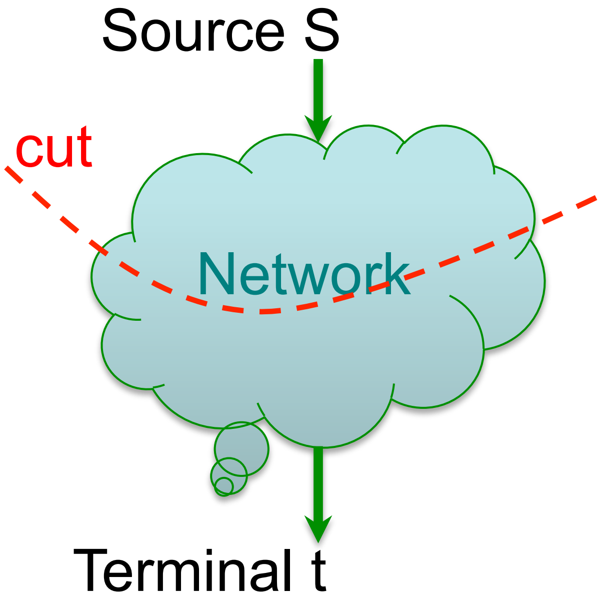


Fig. 12: Average decoding error $\|\mathbf{X} - \hat{\mathbf{X}}\|$ in APIndexcoding on random undirected graphs when $p = 0.2$, $\epsilon = 0.001$ and $X_i \in [-10, 10]$ ($X_{\max} = 10$).

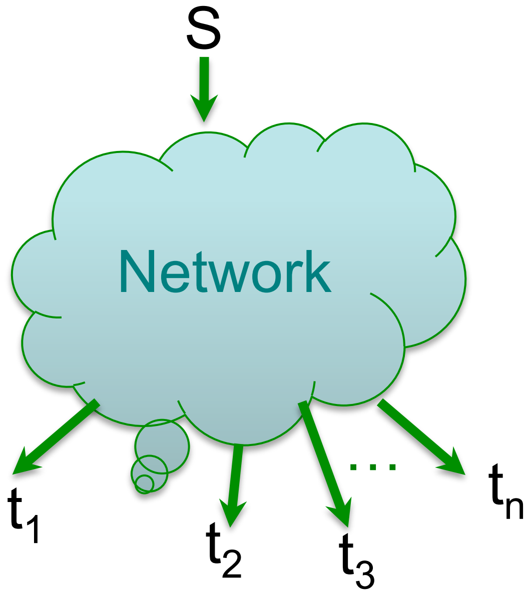
Information Flows in Wireline Networks:
What do we know in one slide



Unicast networks

Max Flow Min Cut theorem

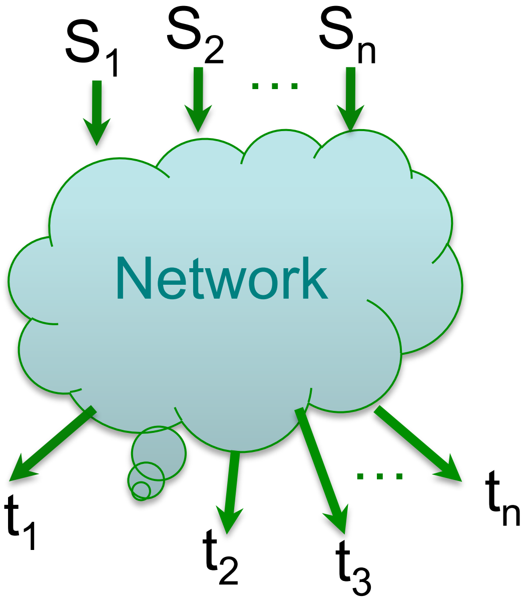
[Ford & Fulkerson, ‘56]
[Elias, Feinstein, Shannon ’56]



Multicast networks

Network coding can achieve min mincut

[Ahlswede et al. ‘00]



General Demands

Open!

Non-linear codes, Non-Shannon inequalities
[Zeger et al. ‘06]

Two-unicast is as hard.
[Kamath, Tse, Wang ‘14]

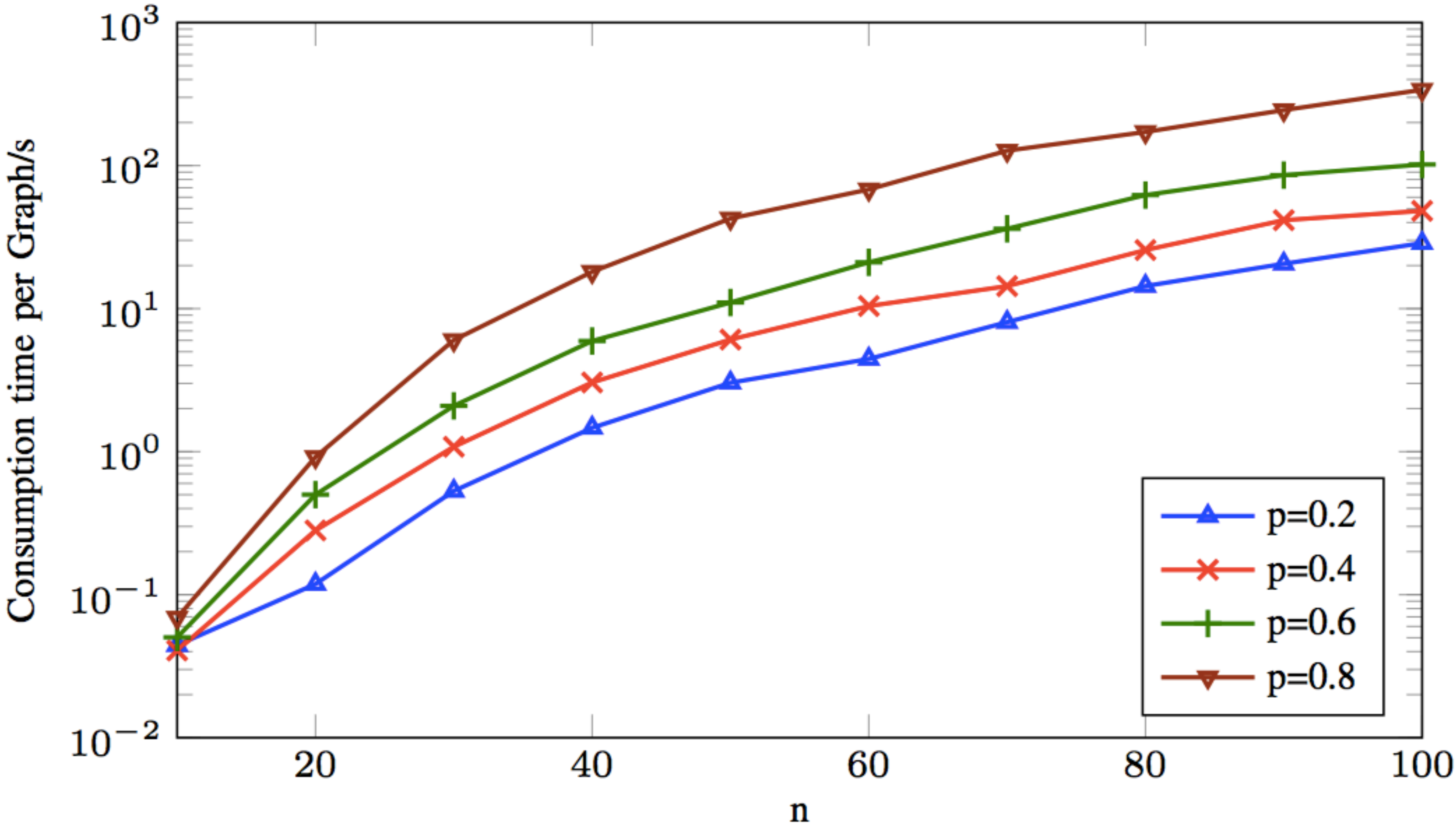


Fig. 9: Time consumption of APIndexCoding on random undirected graphs