Signal Recovery Under Perturbations

Eeshan Malhotra, 14305R001

Under the guidance of

Prof. Ajit Rajwade
Department of Computer Science and Engineering, IIT Bombay

Dr. Karthik Gurumoorthy
Amazon Development Centre & ICTS Bangalore

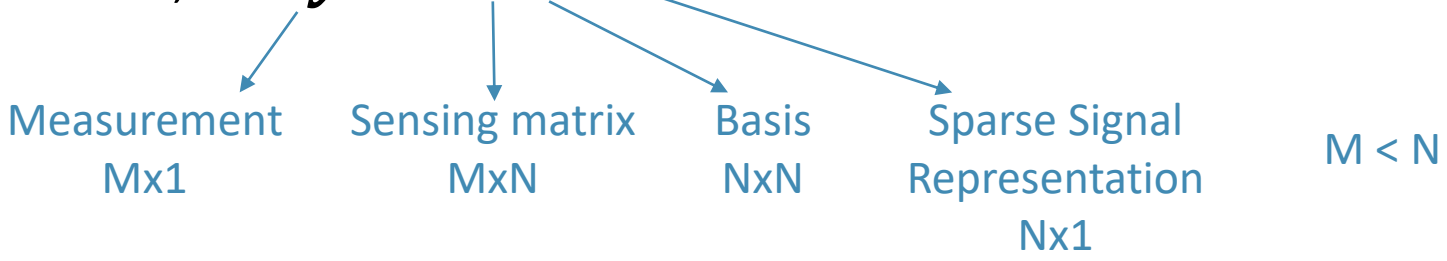
Outline

- Introduction
- Perturbation in sensing matrix
  - Problem definition, algorithm and experimental results
- Tomographic reconstruction with unknown angles
  - Problem definition, algorithm and experimental results
- Interference among two sparse signals
  - Framework for improving recovery guarantees

Introduction: Compressed Sensing Framework

$$\boldsymbol{y} = \boldsymbol{\Phi x}$$

or, $$\boldsymbol{y} = \boldsymbol{\Phi \Psi \theta}$$



Typical scenario: $\boldsymbol{y}$, $\boldsymbol{\Phi}$, $\boldsymbol{\Psi}$ are known. Recover $\boldsymbol{\theta}$ (or $\boldsymbol{x}$).

$$\min_{\boldsymbol{x}} ||\boldsymbol{y} - \boldsymbol{\Phi x}||_2^2 + \lambda ||\boldsymbol{x}||_1$$

In practice: Any of $\boldsymbol{y}$, $\boldsymbol{\Phi}$, $\boldsymbol{\Psi}$ may have inaccuracies, or perturbations, or even simply structural properties known

Introduction: Compressed Sensing Framework

$$\boldsymbol{y} = \boldsymbol{\Phi x}$$

$$\min_{\boldsymbol{x}} ||\boldsymbol{y} - \boldsymbol{\Phi x}||_2^2 + \lambda ||\boldsymbol{x}||_1$$

In practice: Any of $\boldsymbol{y}$, $\boldsymbol{\Phi}$, $\boldsymbol{\Psi}$ may have inaccuracies, or perturbations, or even simply structural properties known

We will discuss and tackle some of these perturbations, and propose empirical algorithms and/or theoretical bounds for recovery in each case. The two broad classes of problems include

1. Perturbation in sensing matrix
2. Interference among two sparse signals

Section 1

Perturbed Sensing Matrix

Perturbed Sensing Matrix

- Sensing matrix Φ may have inaccuracies
- Sources: Calibration error, relative motion between instrument and sample
- E.g.: Frequencies of Fourier sensing matrix in MRI being off by a small amount, tomography projection angles being imperfectly known, etc.
- Because of its widespread use, we consider the Fourier sensing matrix for measurement
- The framework developed is more general in applicability than the specific cases mentioned

Perturbed Sensing Matrix

Consider the case of a Fourier sensing matrix with erroneously specified frequencies

Instead of $\boldsymbol{y_{true}} = \boldsymbol{Fx} + \boldsymbol{\varepsilon}$

We measure $\boldsymbol{y} = \boldsymbol{\widetilde{F}x} + \boldsymbol{\varepsilon}$

Where

$\boldsymbol{F}$: Fourier sensing matrix at presumed frequencies $\boldsymbol{u}$

$\boldsymbol{\widetilde{F}}$: Fourier sensing matrix at the perturbed frequencies $\boldsymbol{\widetilde{u}}$

$\boldsymbol{\widetilde{u}} = \boldsymbol{u} + \boldsymbol{\delta u}$, $\boldsymbol{\delta u}$ is small.

Perturbed Sensing Matrix: Framework

Using a first order Taylor series approximation

$$\widetilde{\boldsymbol{F}} = \boldsymbol{F} + \Delta\frac{\boldsymbol{\partial F}}{\boldsymbol{\partial u}} + \boldsymbol{R}$$

where Δ is a diagonal matrix, with $\Delta_{i,i} = \boldsymbol{\delta u}_i$,

$\boldsymbol{R}$ is the second order remainder in the Taylor approximation

It can be verified that

$$\frac{\boldsymbol{\partial F}}{\boldsymbol{\partial u}} = \boldsymbol{FX}$$

where $\boldsymbol{X}$ is a diagonal matrix, with $\boldsymbol{X}_{i,i} = \left(-2\pi j\ \frac{i}{N}\right)$

Therefore, $$\boldsymbol{y} = (\boldsymbol{F} + \Delta\boldsymbol{FX})\boldsymbol{x}$$

The algorithm is designed for a general acquisition model of the form $\boldsymbol{y} = (\boldsymbol{A} + \Delta\boldsymbol{B})\boldsymbol{x}$

Perturbed Sensing Matrix: Algorithm

$$\boldsymbol{y} = (\boldsymbol{F} + \Delta \boldsymbol{F}\boldsymbol{X})\boldsymbol{x}$$

We use a two-step, alternating algorithm, where we iterate multiple times over two optimization problems – each one solved using well-known convex optimization methods

- **Step 1**: Solve for best $\boldsymbol{x}$

$$\boldsymbol{x} = \min_{\hat{x}} ||\boldsymbol{y} - (\boldsymbol{F} + \Delta \boldsymbol{F}\boldsymbol{X})\hat{\boldsymbol{x}}||_2^2 + \lambda ||\hat{\boldsymbol{x}}||_1$$

- **Step 2**: Solve for best diagonal Δ

$$\Delta = \min_{\hat{\Delta}} ||\boldsymbol{y} - \left(\boldsymbol{F} + \hat{\Delta}\boldsymbol{F}\boldsymbol{X}\right)\boldsymbol{x}||_2^2$$
$$s.t. \qquad (\boldsymbol{\delta u})^2 \leq \boldsymbol{E}^2$$

Where $\boldsymbol{E}^2$ is an upper bound on the magnitude of the perturbations (a constant vector, containing the same value E_{max} in each element)

Perturbed Sensing Matrix: Results

- In experiments with simulated data, results are much superior than a naïve approach of ignoring perturbation
- Error measured as:

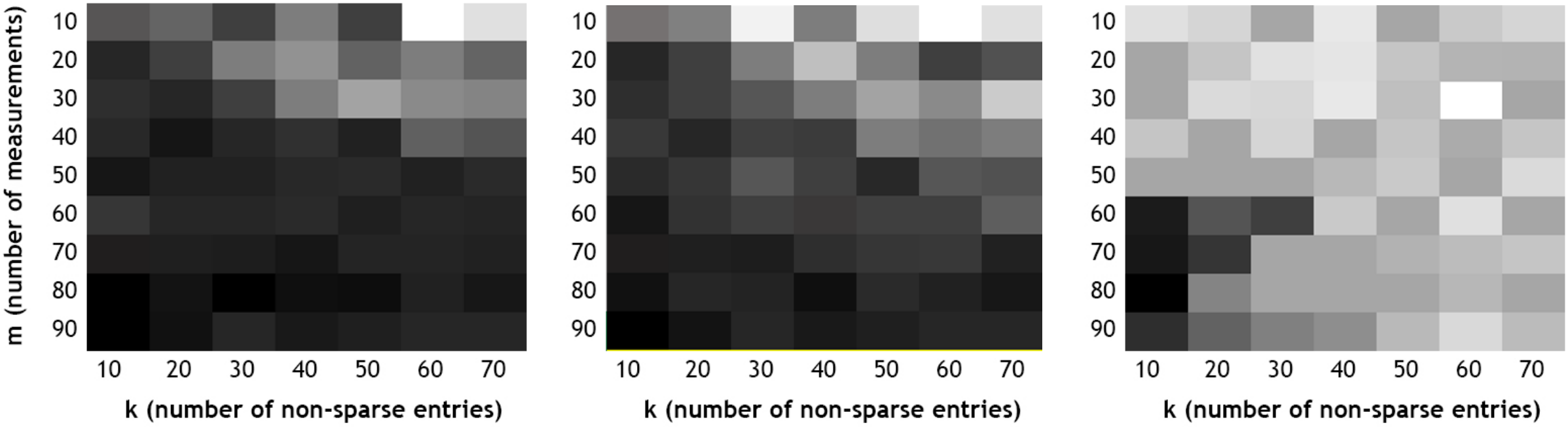
$$relative\_error = \frac{||x_{true} - x_{recovered}||_2}{||x_{true}||_2}$$

- With $E_{max} \approx 0.1$,
  - Alternating algorithm: error of ≤5% consistently
  - Ignoring perturbations: often >20%
- Not much impact of varying length, l-2 norm
- Results significantly vary by changing number of measurements, sparsity of signal, E_{max}

Perturbed Sensing Matrix: Results

Relative error in signal, x

Darker cells denote lower fractional error.
Min error: 2.5%; Max error: 95%



$E_{max} = 0.01$ $E_{max} = 0.1$ $E_{max} = 0.5$

Perturbed Sensing Matrix: 2D recovery

- As such, extending to 2D images can be achieved naturally
- Special case arises when errors in a radial acquisition scheme lie only in angle calibration
- $\tilde{\theta}$ is the set of perturbed angles, given by $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta} + \boldsymbol{\delta\theta}$.
- The degrees of freedom in this formulation are considerably fewer: Only T angle values are unknown, compared to N (values of r) * T (angle values) in the basic case

Polar coordinate system is a natural choice for this system

Thus, for image $\boldsymbol{I}$ $(N \times N)$, in polar coordinates:

$$\tilde{F}(r, \tilde{\theta})\, I = \sum_{x,y} \left(e^{-2\pi j r \left(\frac{x \cos\tilde{\theta}}{N} + \frac{y \sin\tilde{\theta}}{N} \right)} \times I(x, y) \right)$$

Perturbed Sensing Matrix: 2D recovery

Again, using a Taylor's series approximation,

$$\tilde{\boldsymbol{F}}(r,\tilde{\theta})\boldsymbol{I} \approx \sum_{x,y}\left[\left(e^{-2\pi jr\left(\frac{x\cos\theta}{N}+\frac{y\sin\theta}{N}\right)}+\right.\right.$$
$$\left.\left.\Delta\times e^{-2\pi jr\left(\frac{x\cos\theta}{N}+\frac{y\sin\theta}{N}\right)}\times 2\pi jr\left(\frac{x\sin\theta}{N}-\frac{y\cos\theta}{N}\right)\right)\times I(x,y)\right]$$

Which is also gives an acquisition model of the form $\boldsymbol{y} \approx (\boldsymbol{A}+\Delta\boldsymbol{B})\boldsymbol{x}$

- The remainder term in this formulation is proportional to $\Delta \times r^2$
- At large values of r, the error in measurement is considerably high even though (per assumption) the error in θ is small
- Experiments verified that this is indeed an issue, and recovery error was large. However, excellent recovery was achieved when the modelling error was artificially reduced to zero

Perturbed Sensing Matrix: Future Work

Proposed Solutions (Yet to be investigated)

- Only use measurements in the low r range in the alternating algorithm. Once the angle estimates converge, utilize all available measurements to re-estimate the signal
- Pooling method: Use multiple signals measured with the same angle perturbations. This increases the number of available data points (measurements), while increasing the number of unknowns by a smaller amount (sparse signals to be recovered)

Other avenues of future work

- Theoretically prove the uniqueness and/or bound the error in the solution obtained by the presumed model, possibly using a lifting scheme

Section 2

Tomographic Recovery with Unknown View Angles

Unknown View Angles: Motivation

- In computed tomography, an extreme case of sensing matrix being erroneously known is when view-angles for projections are completely unknown
- Eg: Cryo-electron microscopy, and to a lesser degree, patient motion in medical CT

- However it has been shown by Basu & Bresler (2000), that under modest assumptions, the image is still uniquely determined by the set of projections (up to a global rotation and reflection ambiguity)
- Existing work (Coifman et al (2008), Fang et al (2010), Fang et al (2011)) typically -
  a. Produces an ordering for projections (often using dimensionality reduction), and assumes a distribution to determine values of angles
  b. Requires an extremely large number of angles for the recovery to be stable

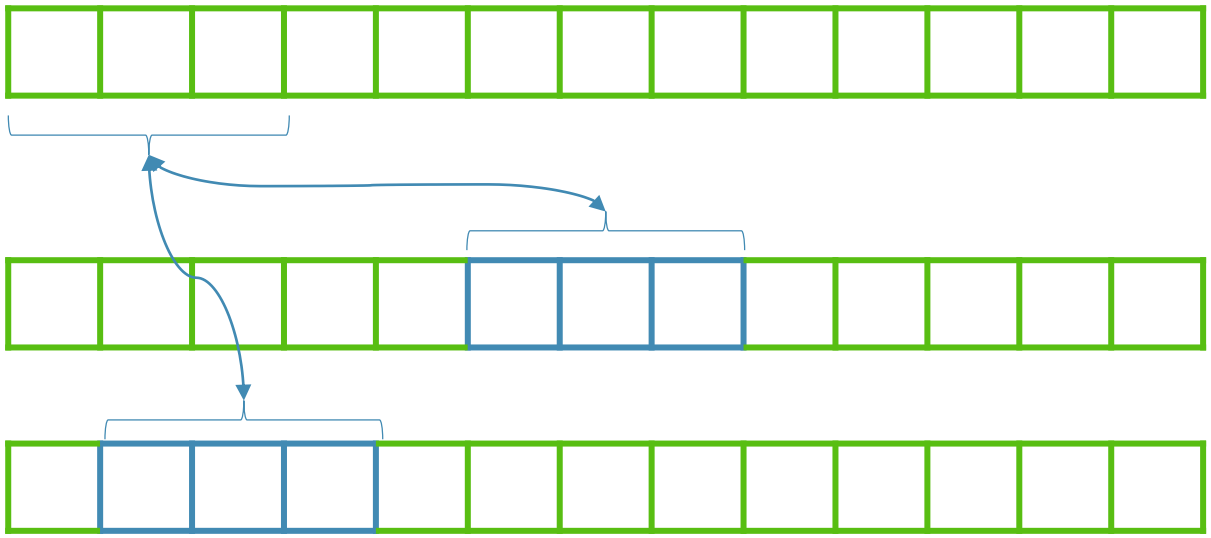
Unknown View Angles: Problem Statement

- Consider
  - A set of angles: $\boldsymbol{\theta} \triangleq \{\theta_1, \theta_2, \theta_3, \ldots \theta_p\}$
  - Corresponding projections of a 2-D image: $\boldsymbol{P_\theta} \triangleq \{P_{\theta_i}\}_{i=1}^{p}$

    Where P_{θ_i} is a vector of line integrals of the image, as viewed from angle θ_i
- Given $\boldsymbol{P_\theta}$, our aim is to recover $\boldsymbol{\theta}$.
- Knowing the angles $\boldsymbol{\theta}$ and the projection $\boldsymbol{P_\theta}$, it is possible to reconstruct the original image using Filtered Back Projection (FBP)
- Other methods for reconstruction are available, and often perform better. FBP is used for its simplicity and robustness to small amounts of noise in angles.

Unknown View Angles: Algorithm (Denoising)

- The first step in the algorithm involves denoising the projections, since noisy projections may significantly hamper the efficacy of the following step
- Patch-based PCA denoising method



For each d-length patch:

- Find L most similar patches
- Perform PCA on this set of L vectors to produce eigencoefficients
- Denoise with Wiener-like update:

$$\hat{x}_{il} \leftarrow y_{il}\left(\frac{\sigma_l^2}{\sigma_l^2 + \sigma^2}\right)$$

where

$\hat{x}_{il}$ is an estimate of the l^{th} denoised coefficient for patch i

y_{il} is the corresponding coefficient in the noisy patch

σ_l^2 is the mean-normalized squared value of the l^{th} coefficient across all L patches

Similar patches can be found in the same projection as well

Unknown View Angles: Algorithm (Coordinate Descent)

- This algorithm utilizes the Helgasson-Ludwig Consistency Conditions (HLCC)
- HLCC describes a relationship between the nth order projection moments and the nth order image moments

Given p projections, the p *nth-order* projection moments are

$$m_{\theta_i}^{(n)} = \int_{-\infty}^{\infty} P_{\theta_i}(s) s^n ds$$

Given the underlying image, the (n+1) *nth-order* image moments are

$$\upsilon_{p,q} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y) x^p y^q dx dy$$

- HLCC states:

$$m_{\theta_i}^{(n)} = \sum_{j=0}^{n} \binom{n}{j} (\cos\theta_i)^{n-j} (\sin\theta_i)^j \upsilon_{n-j,j}$$

- Or in matrix form:

$$m^{(n)} = A^{(n)} \upsilon^{(n)}$$

Unknown View Angles: Algorithm (Coordinate Descent)

- In our problem setting, neither image moments, nor view angles are known. Given *estimates* of each we can define our cost function as the HLCC residual

$$E(\boldsymbol{\theta}, \boldsymbol{v}) = \sum_{n=0}^{k} \sum_{i=1}^{p} \left(m_{\theta_i}^{(n)} - \sum_{j=0}^{n} \boldsymbol{A}_{i,j}^{(n)} \boldsymbol{v}_{n-j,j} \right)^2$$

- This is the key quantity in the algorithm. We use a coordinate descent strategy, with multiple passes over the data.
- At each step, *one* angle estimate, θ_i is improved, using a 1-D brute-force search, and the image moments recalculated
- $E(\theta, \upsilon)$ is a non-negative quantity that decreases at each step, and is therefore guaranteed to converge
- The algorithm is sensitive to initialization values, and to reduce the impact of this, a multi-start strategy is used

Unknown View Angles: Results

- Simulated experiments gave quite good recovery, across multiple images, even with very few angles
- Most estimates were off by ≤ 3°, with rare exceptions going above 5°

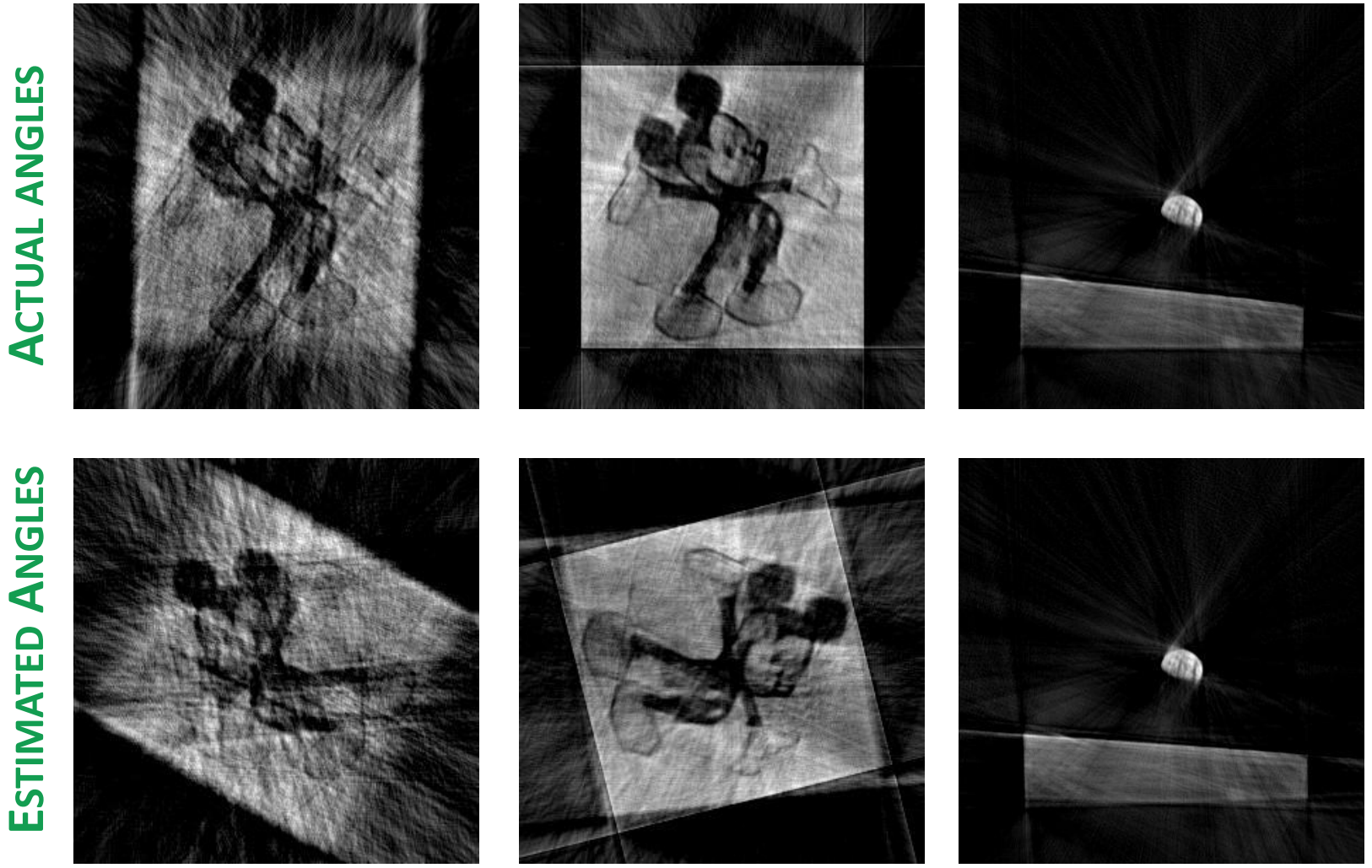
Error	Earthrise 30 angles	Earthrise 100 angles	Mickey 30 angles	Mickey 100 angles
$\leq 1°$	13	94	20	66
$\leq 3°$	29	99	29	96
$\leq 5°$	30	100	29	100
$> 5°$	0	0	1	0

(Rotational ambiguity was eliminated manually for clearer representation)

Unknown View Angles: Results

Reconstruction results using FBP

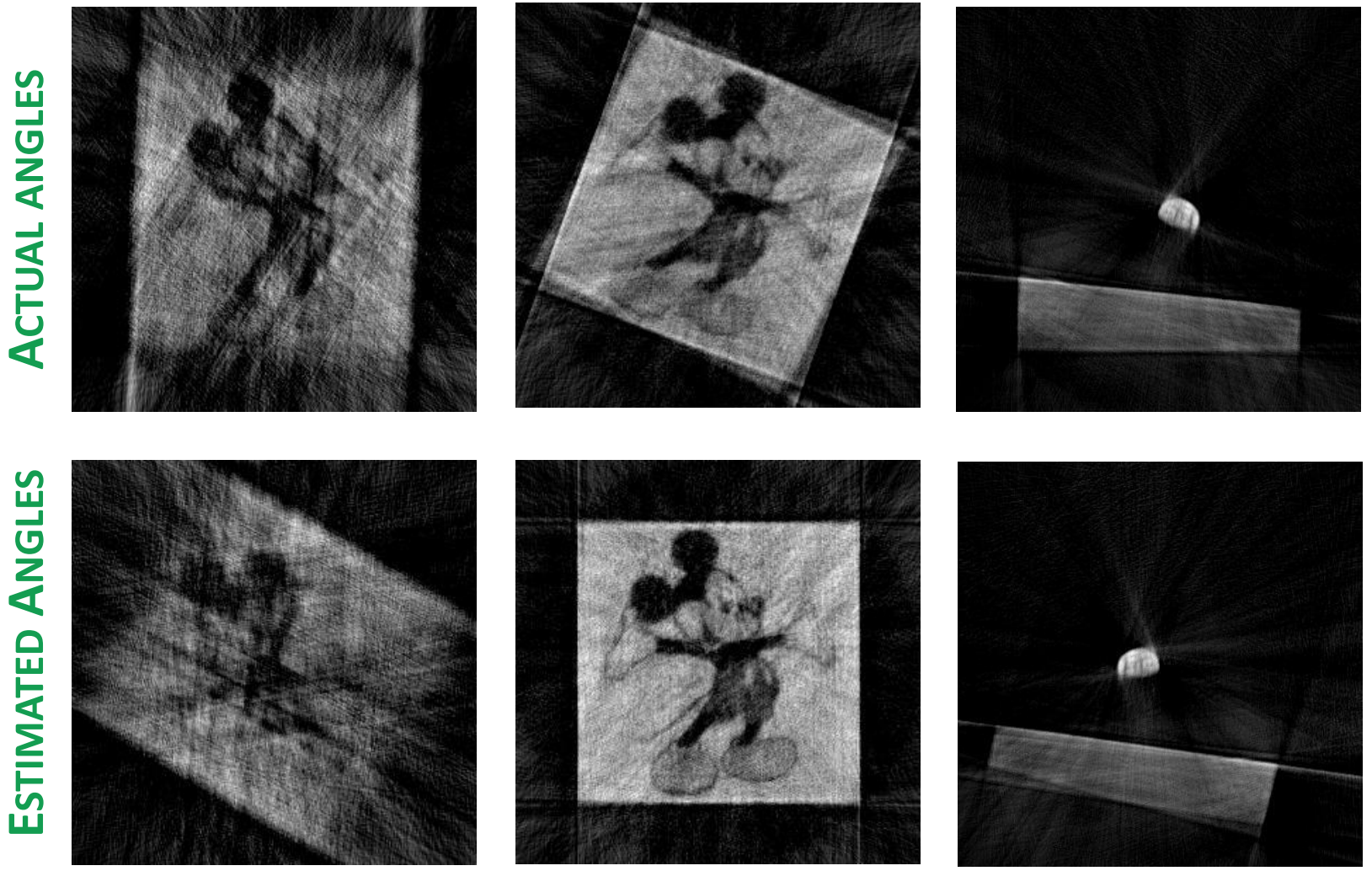
5% noise



Unknown View Angles: Results

Reconstruction results using FBP

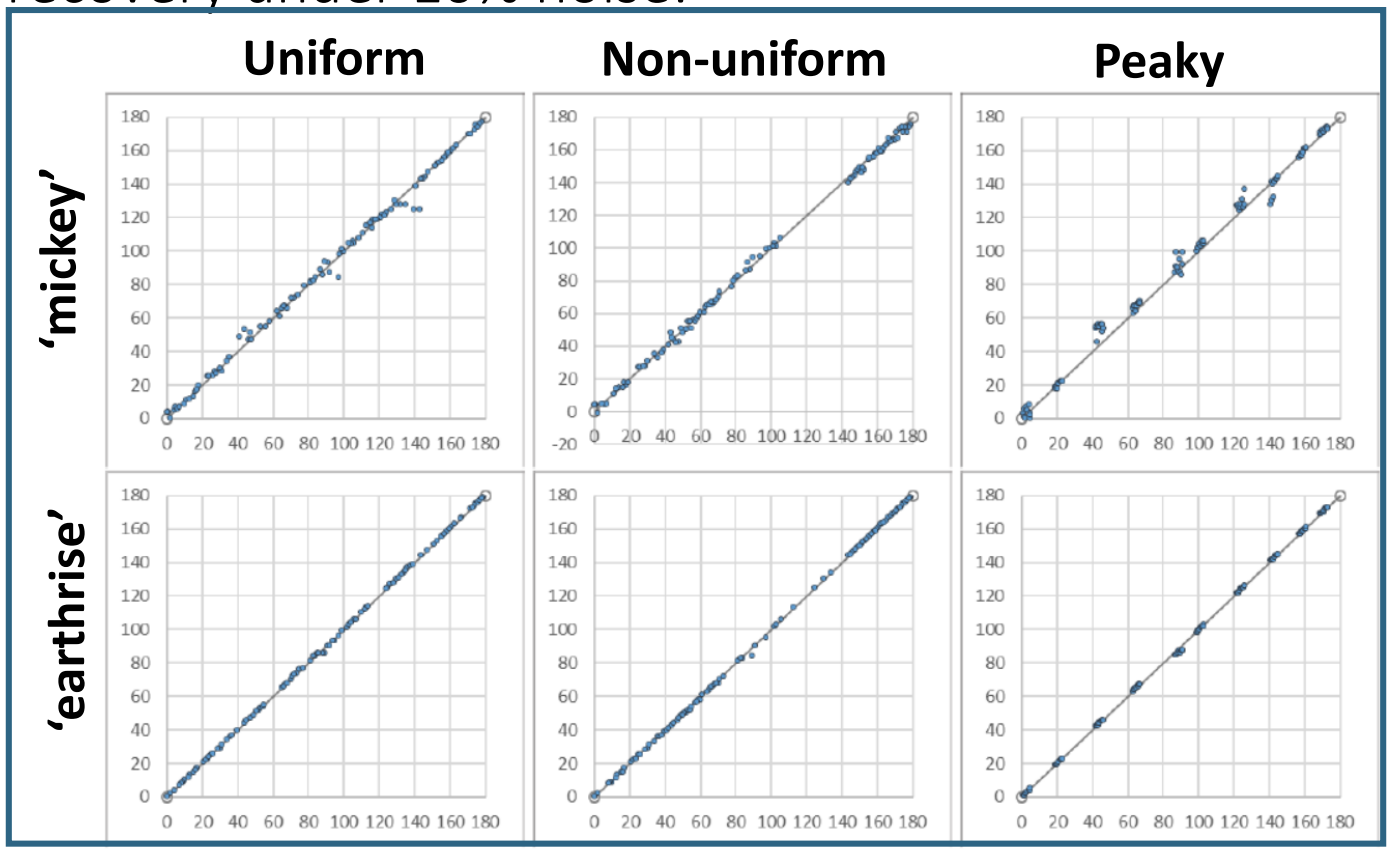
10% noise



Unknown View Angles: Results

- The algorithm also proved to be extremely robust to the distribution from which the angles were taken. Angle recovery under 10% noise:

X: True angle value
Y: Estimated angle value



(Rotational ambiguity was eliminated manually for clearer representation)

Unknown View Angles: Conclusions and Future Work

- This algorithm extends the angle recovery to scenarios where the projection angles are completely unknown
- The recovery is excellent, considering there is zero available knowledge about view angles, and the algorithm is robust. However, the angle errors are not small enough for applications requiring precise measurements
- For such scenarios, we propose a pipeline using the Coordinate descent algorithm for obtaining approximate estimates of the angles, followed by fine-tuning using the alternating algorithm described in the first section

Section 3

Interference Among Sparse Signals

Interference: Background

- Consider one more time, the forward model

$$\boldsymbol{y} = \boldsymbol{D}\boldsymbol{x} + \boldsymbol{\eta}$$

- Sparse signal recovery from the compressed measurement using basis pursuit has well-established bounds based on the coherence of the dictionary $\boldsymbol{D}$
- We look at the case where the k-sparse signal $\boldsymbol{x}$ is actually a composite signal of two individually sparse signals $\boldsymbol{x_a}$ and $\boldsymbol{x_b}$, i.e.:

$$x^T = \left[x_a^T \;\; x_b^T\right]; \quad D = [A\ B]$$

- $\boldsymbol{x_a}$ and $\boldsymbol{x_b}$ can be recovered using the same technique. However, the guarantees can be improved by utilising this structure in the signal

Interference: Background

- Studer & Baraniuk (2014) showed coherence based results in such a scenario, which are reproduced here
- Coherence of the individual dictionaries $\boldsymbol{A}$ and $\boldsymbol{B}$ with unit-normalized columns is defined as:

$$\mu_a = \max_{i,j,i\neq j} \left|A_i^T A_j\right| \quad \text{and} \quad \mu_b = \max_{i,j,i\neq j} \left|B_i^T B_j\right|$$

- The mutual coherence is defined as

$$\mu_m = \max_{i,j} \left|A_i^T B_j\right|$$

- Consequently, the coherence of the dictionary $\boldsymbol{D}$

$$\mu_d = \max_{i,j,i\neq j} \left|D_i^T D_j\right| = \max\{\mu_a, \mu_b, \mu_m\}$$

Interference: Background

Studer & Baraniuk show that assuming $\mu_a \geq \mu_b$

- 1. If

$$k < \max\left\{\frac{2(1+\mu_a)}{\mu_a + 2\mu_d + \sqrt{\mu_a^2 + \mu_m^2}}, \frac{1+\mu_d}{2\mu_d}\right\}$$

then the solution $\hat{x}$ of the basis pursuit problem

$$\text{minimize } ||\tilde{x}||_1$$
$$\text{subject to } ||y - D\tilde{x}||_2 \leq \eta$$

$$\text{satisfies } ||x - \hat{x}||_2 \leq C(\epsilon + \eta) + C'(||x - x_s||_1)$$

- 2.

$$(1-\hat{\delta})||x||_2^2 \leq ||Dx||_2^2 \leq (1+\hat{\delta})||x||_2^2$$

where

$$\hat{\delta} = \min\left\{\frac{1}{2}\left(\mu_a(k-2) + k\sqrt{\mu_a^2 + \mu_m^2}\right), \mu_d(k-1)\right\}$$

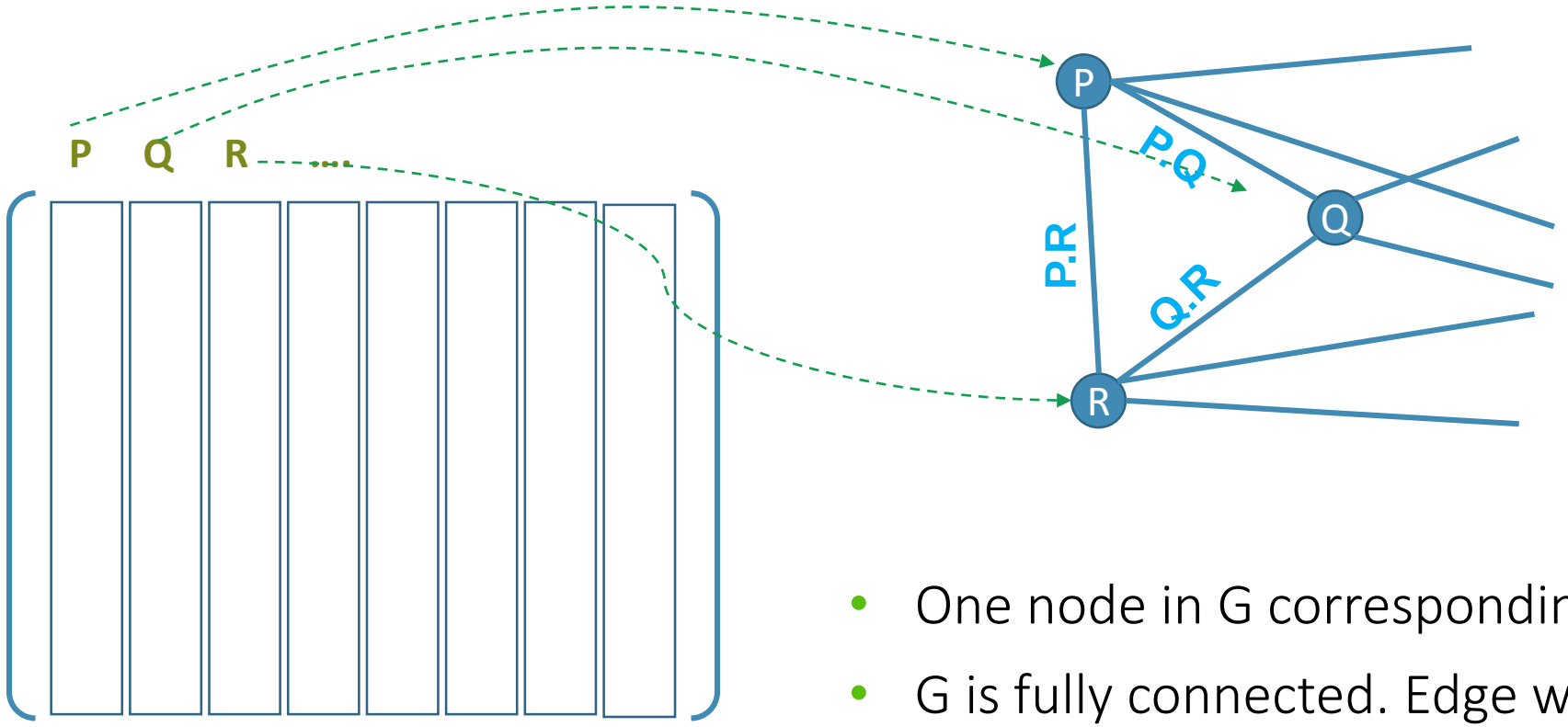
Interference: Problem Statement

- The previous result assumes there is an obvious structure apparent in the dictionary
- The result itself is applicable to any arbitrary split in the dictionary $\boldsymbol{D}$
- This is exactly what we do – we propose an algorithm to *induce* a split in a given dictionary $\boldsymbol{D}$ that optimizes the bounds!
- Specifically, we solve:
- Given a dictionary $\boldsymbol{D}$ devise matrices $\boldsymbol{A}$, $\boldsymbol{B}$, such that $\boldsymbol{D}' = [\boldsymbol{A}\ \boldsymbol{B}]$, and the columns of $\boldsymbol{D}'$ can be permuted to give $\boldsymbol{D}$, so as to maximize F :

$$F = \frac{2(1+\mu_a)}{\mu_a + 2\mu_d + \sqrt{\mu_a^2 + \mu_m^2}}$$

Interference: Algorithm

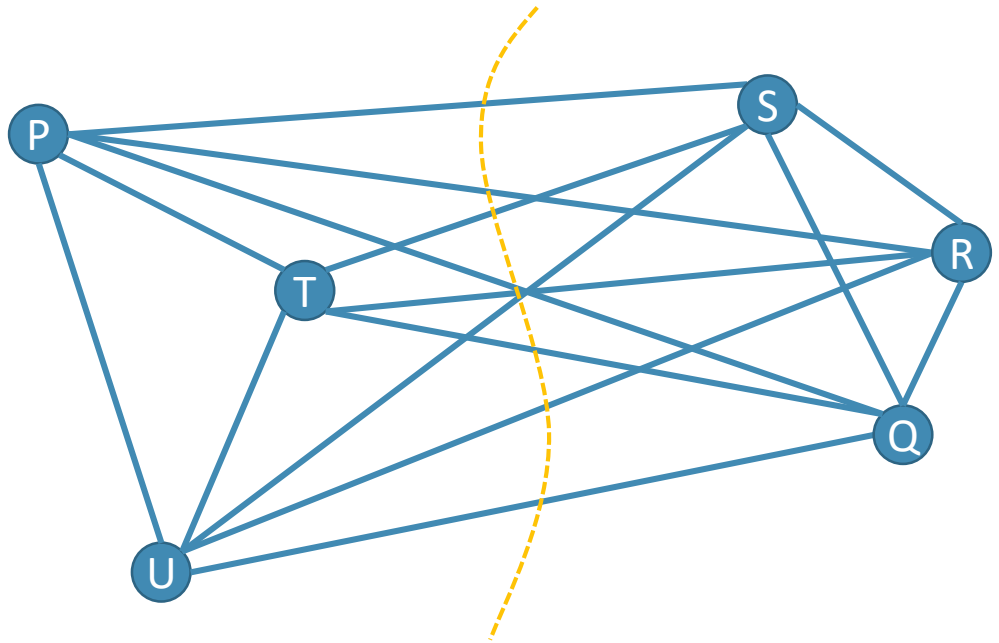
- Equivalent graph problem:



- One node in G corresponding to each column in D
- G is fully connected. Edge weight = dot product of the columns connected

Interference: Algorithm

- Equivalent graph problem:



- Define a *cut* through the graph, separating it into components $\boldsymbol{A}(\boldsymbol{V_a}, \boldsymbol{E_a})$ and $\boldsymbol{B}(\boldsymbol{V_b}, \boldsymbol{E_b})$
- $\boldsymbol{E_x}$ be the edges in the cut
- $\boldsymbol{\mu_a}, \boldsymbol{\mu_b}$: Heaviest edges in components $\boldsymbol{A}$ and $\boldsymbol{B}$ respectively
- $\boldsymbol{\mu_m}$: Heaviest edge crossing the cut

Interference: Algorithm

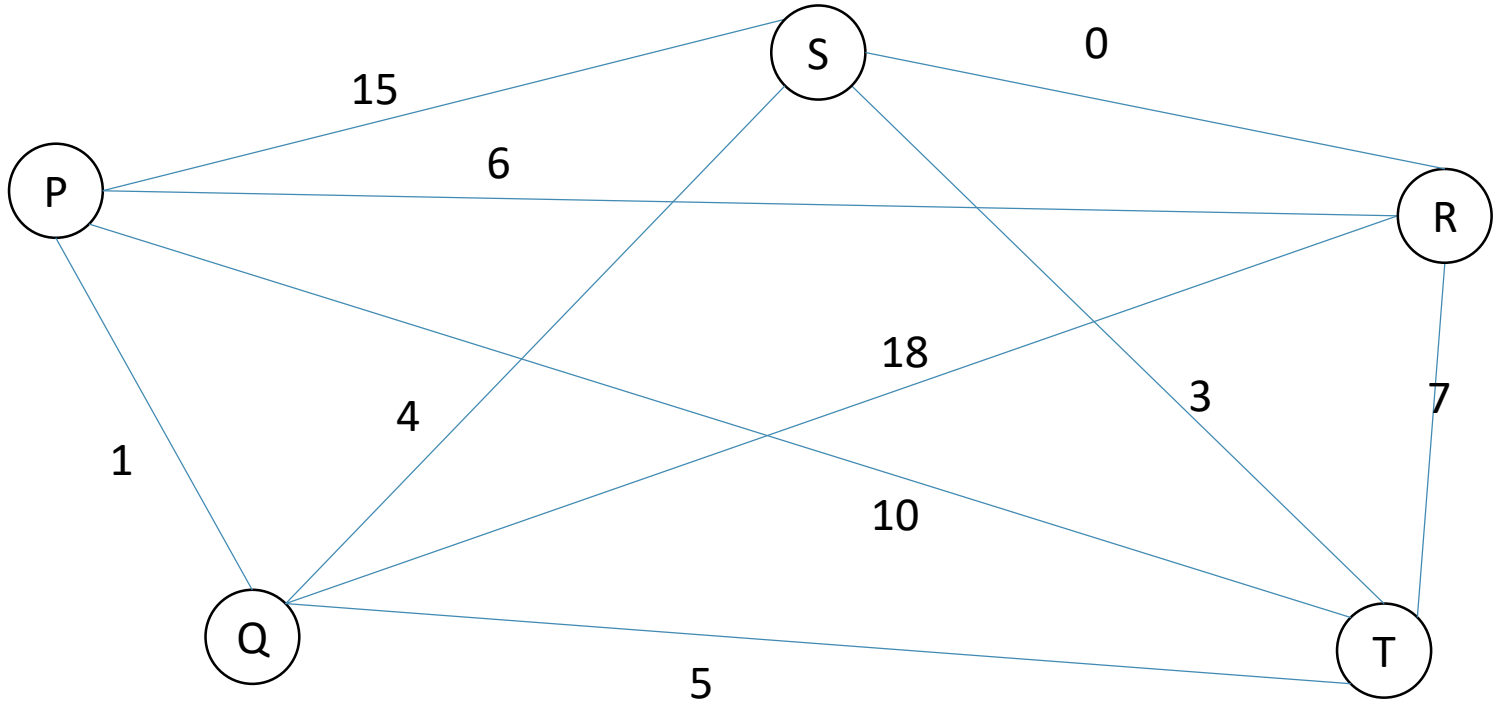
Algorithm intuition:

This is essentially a greedy algorithm

- Start with the highest weight edge in the cut, assigning one node each to parts **A** and **B**.
- While not all vertices are assigned:
  - Candidate edges = edges with exactly one end-point assigned.
  - Pick max weight edge from candidates (say *uv*, with *u* assigned).
  - Assign *v* to the component that *u* is not in.

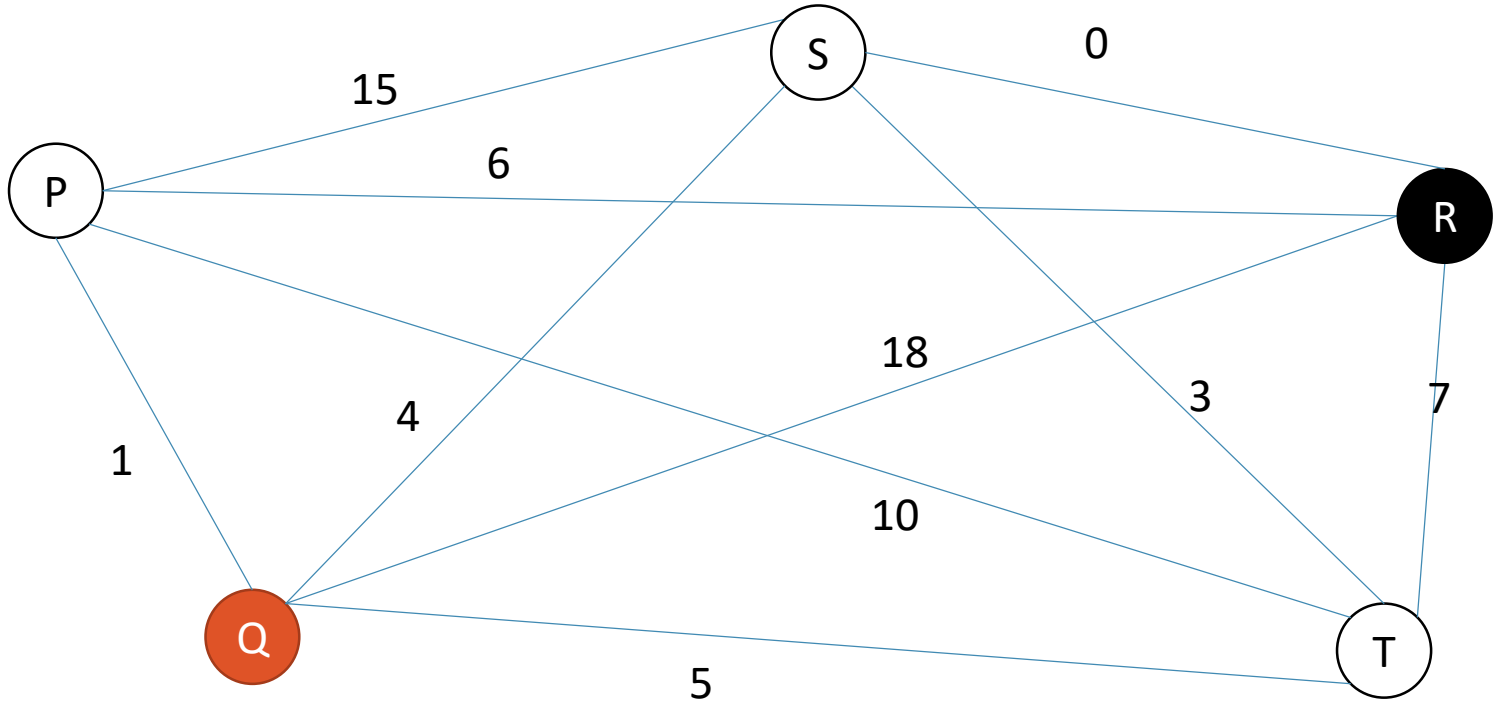
Interference: Algorithm

- Example run



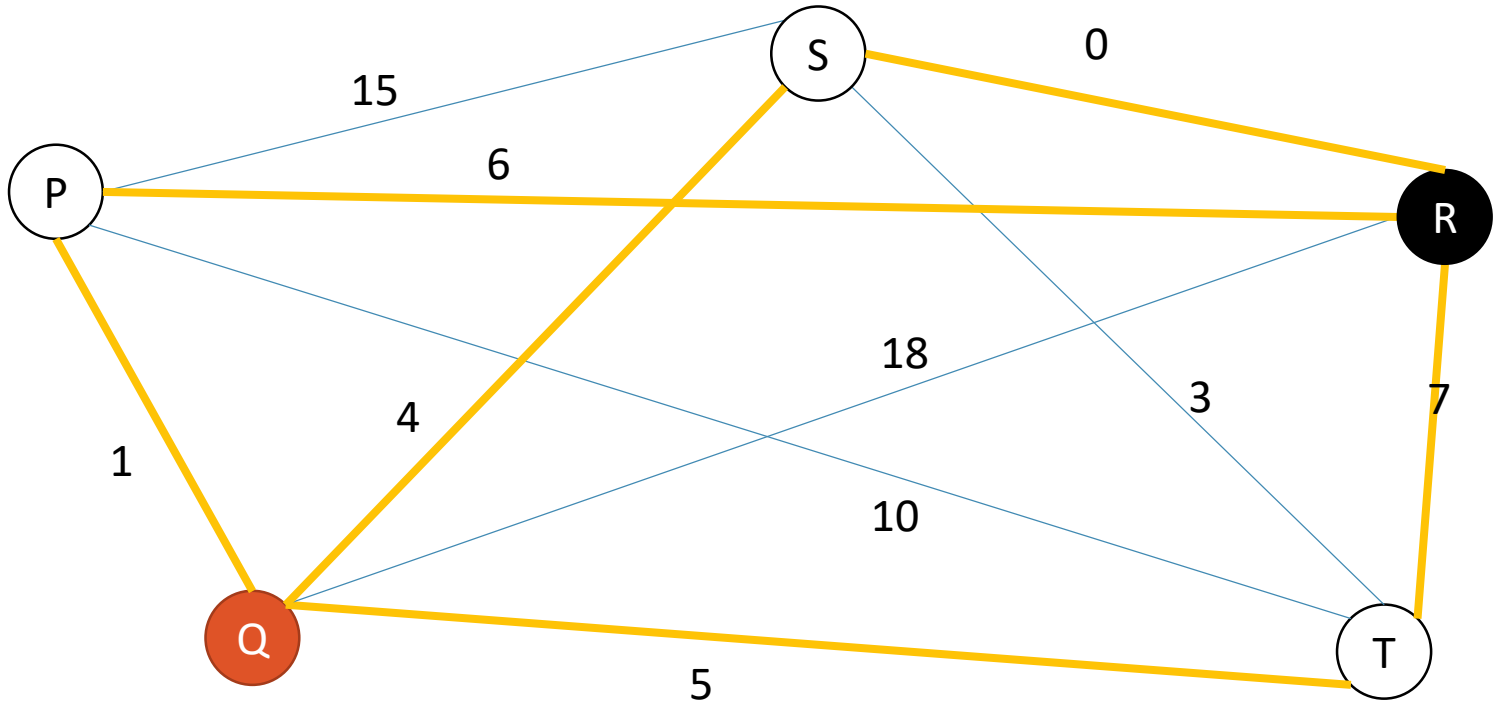
Interference: Algorithm

- Example run



Interference: Algorithm

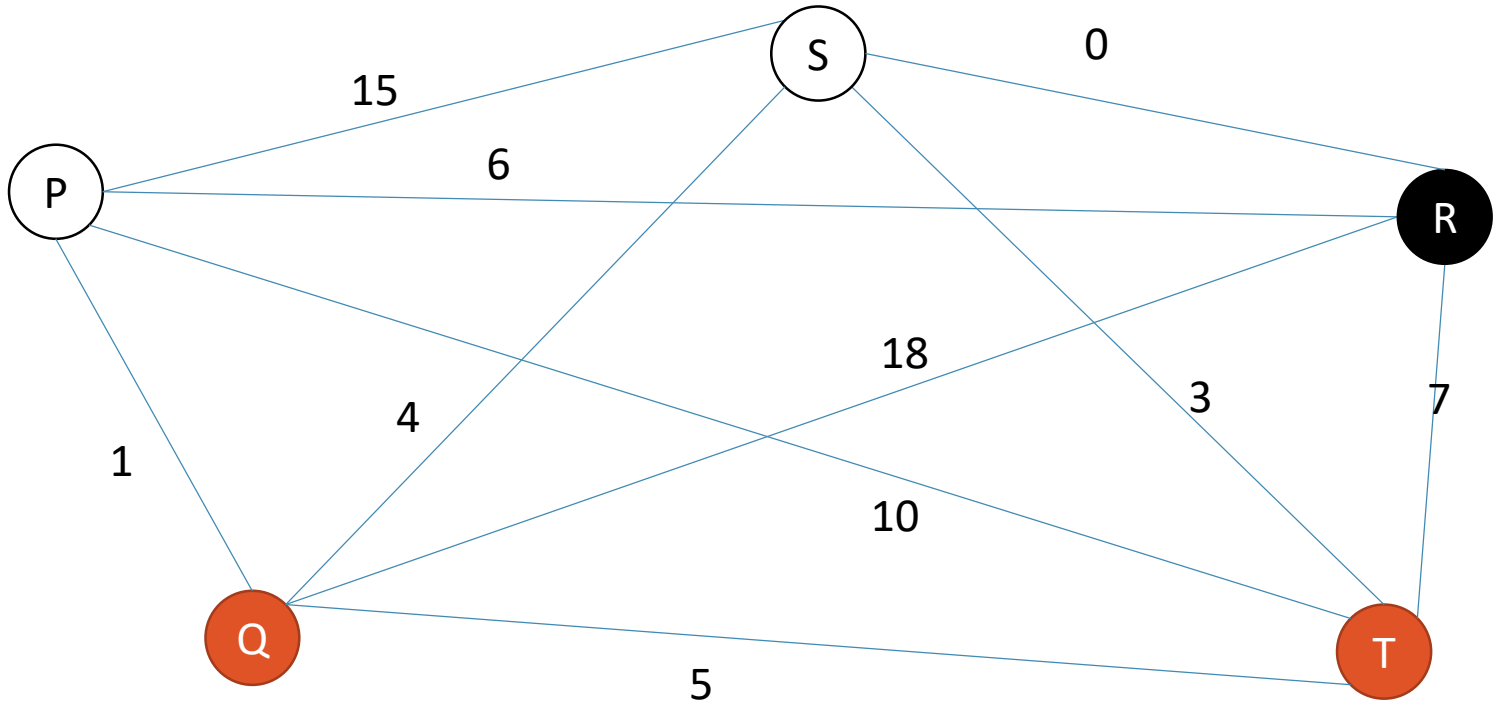
- Example run



Candidate Edges

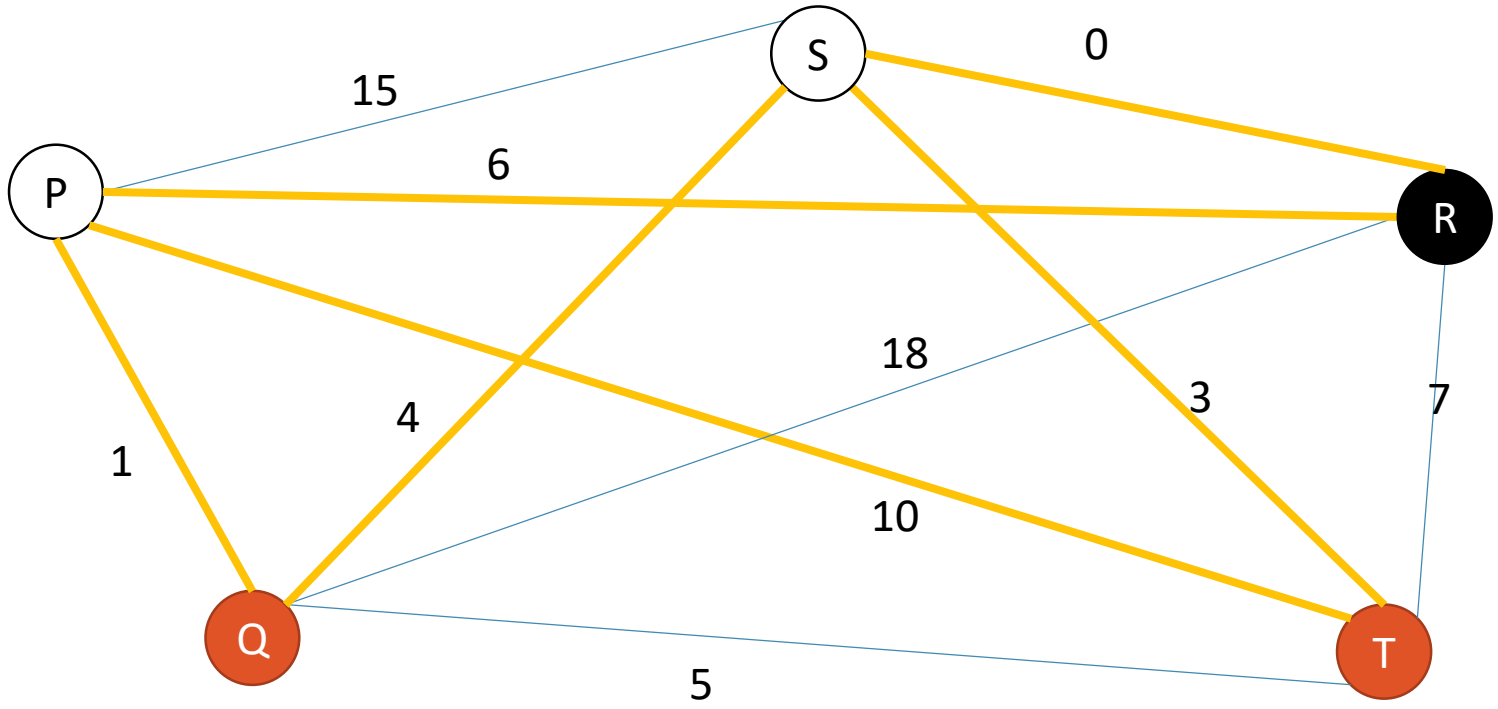
Interference: Algorithm

- Example run



Interference: Algorithm

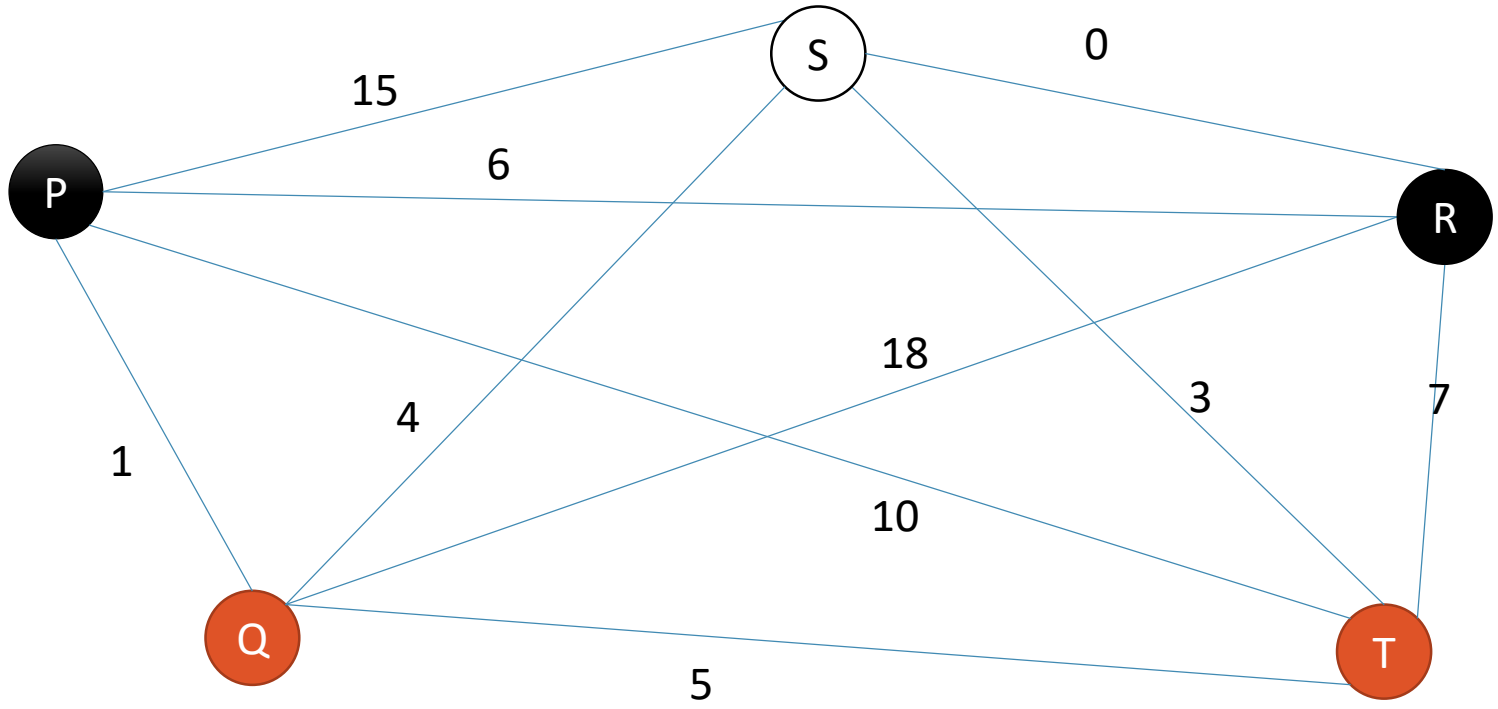
- Example run



Candidate Edges

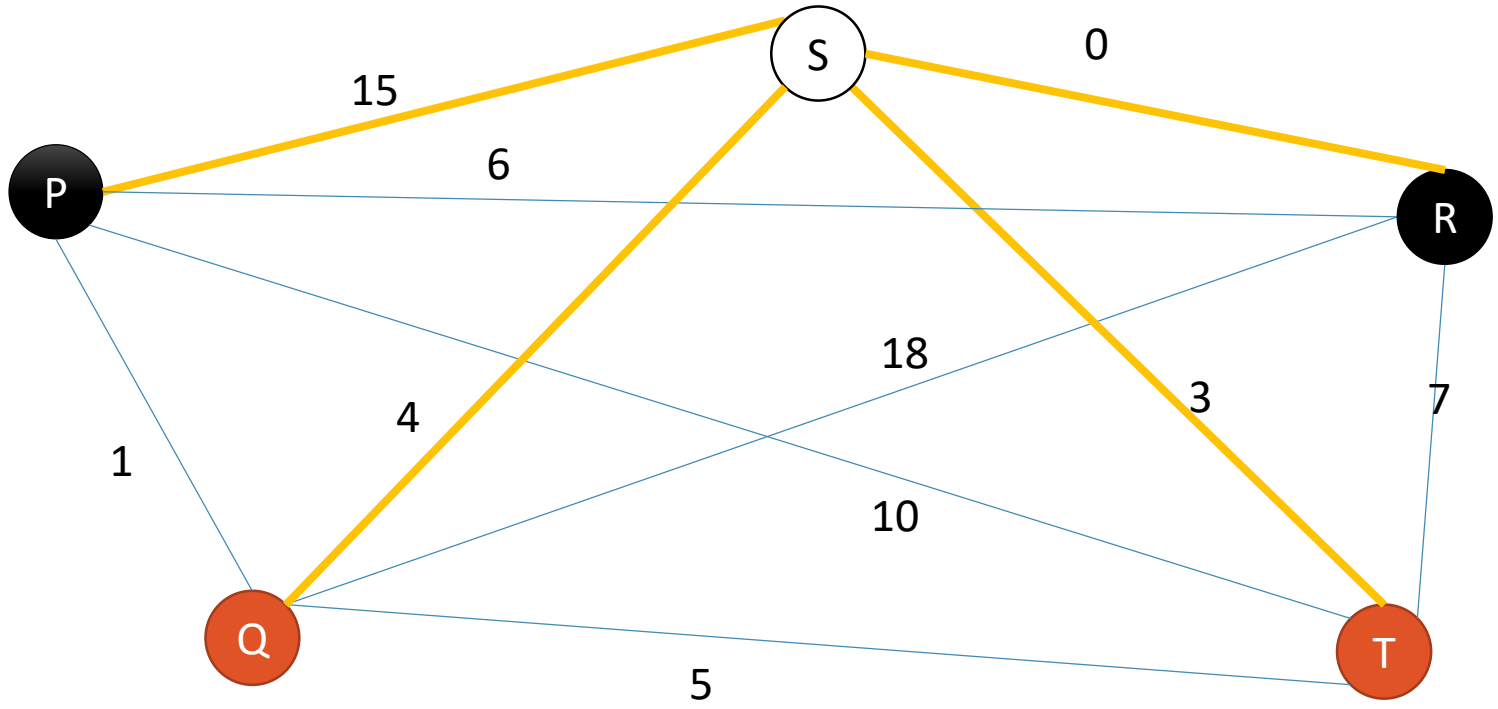
Interference: Algorithm

- Example run



Interference: Algorithm

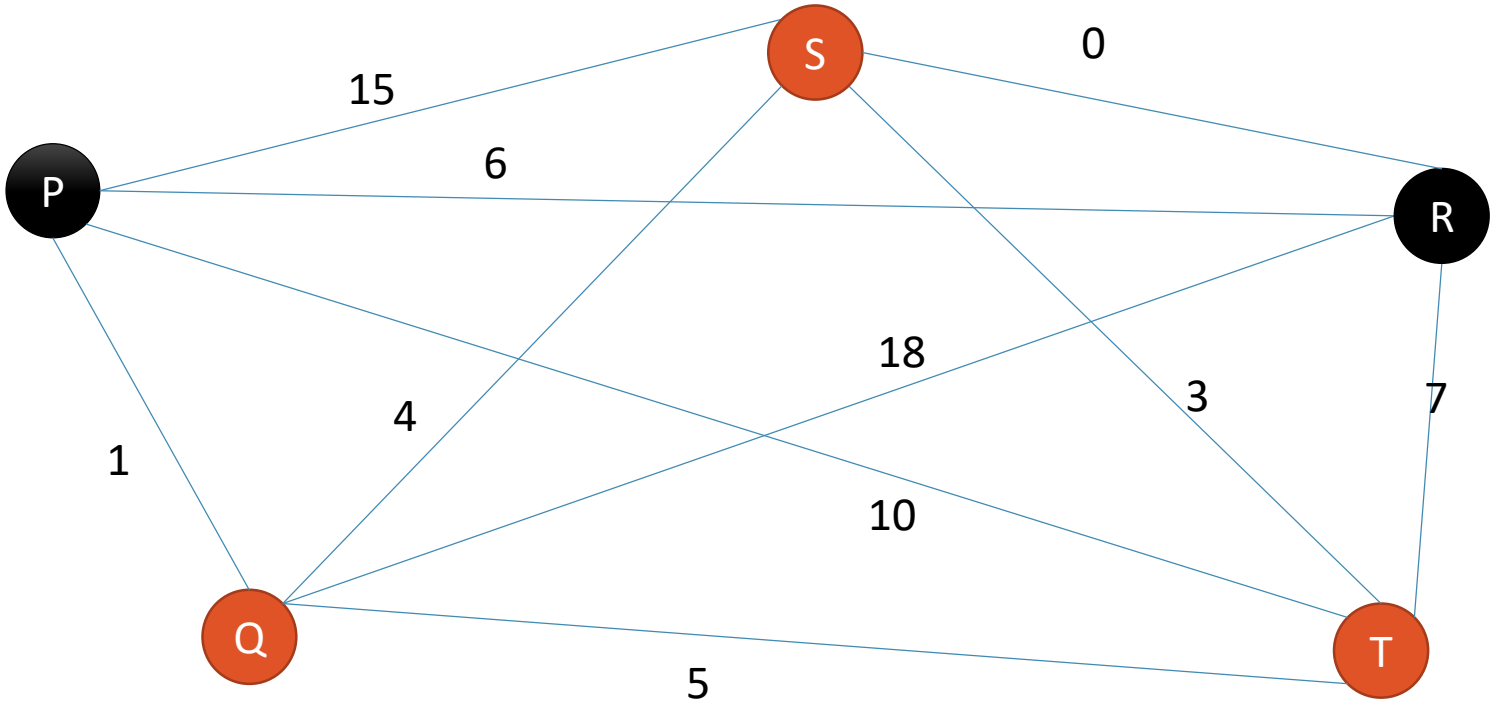
- Example run



Candidate Edges

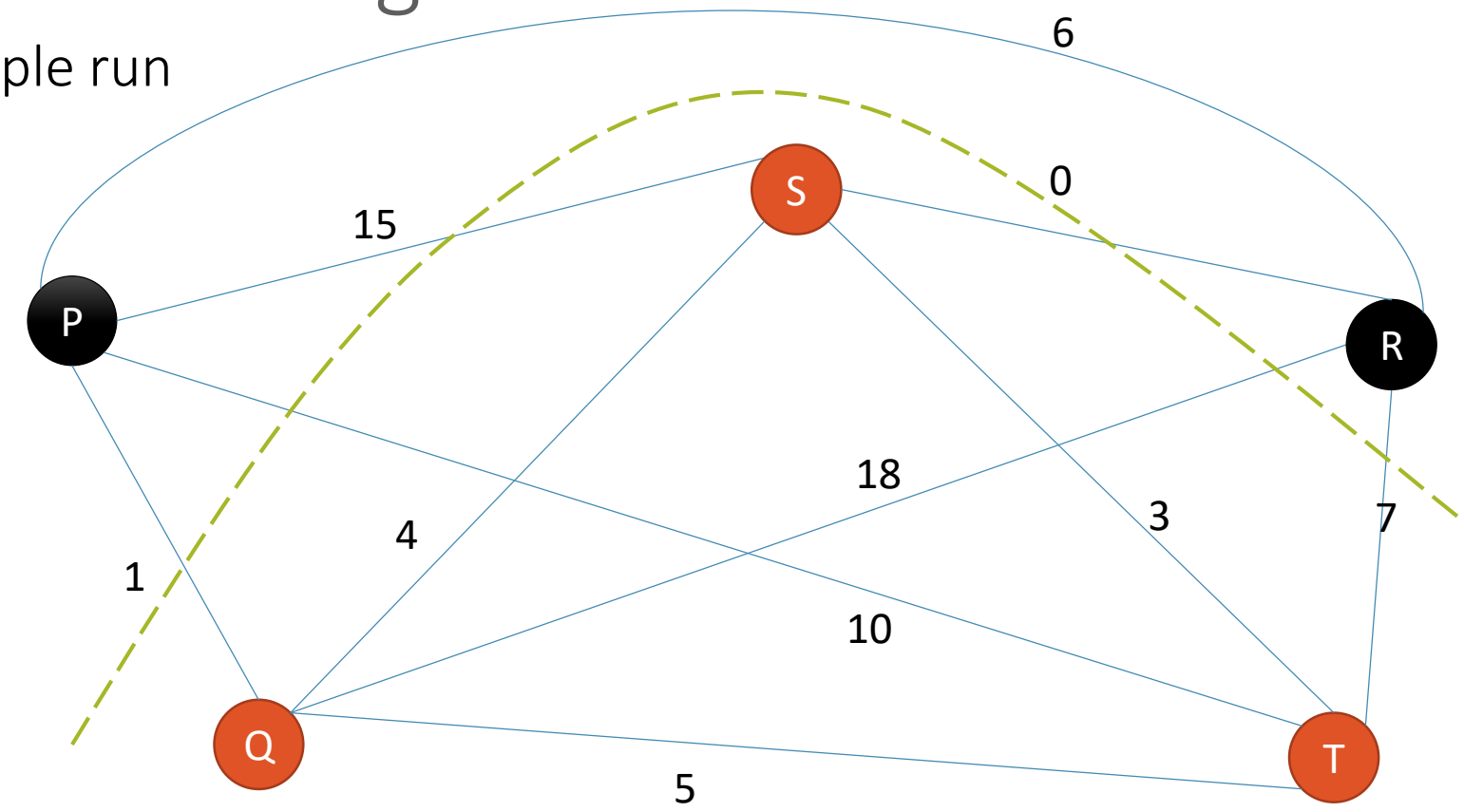
Interference: Algorithm

- Example run



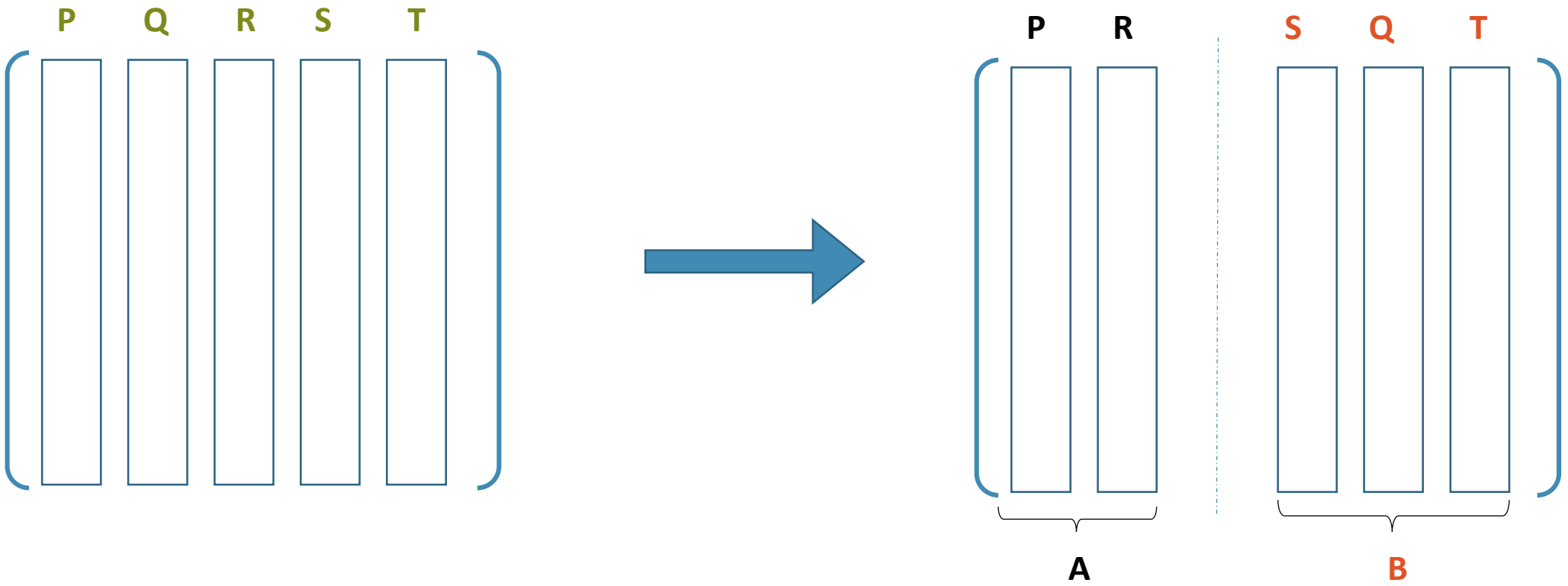
Interference: Algorithm

- Example run



Interference: Algorithm

Corresponding dictionary split



Interference: Algorithm

Why does this work?

- Given a dictionary μ_d is unalterable. The two column with the highest coherence can either contribute to μ_m or to $max\{\mu_a\ \mu_b\}$
- We saw that $\hat{\delta} = \min\left\{\frac{1}{2}\left(\mu_a(k-2) + k\sqrt{\mu_a^2 + \mu_m^2}\right),\ \mu_d(k-1)\right\}$

  Comparing the two quantities, the *effective* μ_d is $\left(\frac{\mu_a + \sqrt{\mu_a^2 + \mu_m^2}}{2}\right)$
- So, it's always beneficial to have μ_a as small as possible, or equivalently, μ_m as large as possible
- The greedy algorithm ensures exactly this – that heavier edges get put into the cut rather than in one of the components

(Full proof of optimality in appendix)

Interference: Higher order recursive algorithm

- The split induced by the previous algorithm is the *optimal* two way split – in the sense of minimizing the effective coherence of the dictionary
- The internal coherences of components A and B feature directly in the calculation of effective μ_d
- But we could optimize these too! The same algorithm could be used to potentially reduce the effective coherence of matrices $\boldsymbol{A}$ and $\boldsymbol{B}$
- In fact, we could recursively apply this idea multiple times
- The multi-way split algorithm executed to depth l, achieves a 2^l-way split

Interference: Higher order recursive algorithm

To calculate effective coherence of $\boldsymbol{D}$ with a depth-l split:

EffectiveCoherence($\boldsymbol{D}$, l)

- Split $\boldsymbol{D}$ into $\boldsymbol{A}$ and $\boldsymbol{B}$ using the previous algorithm
- $\mu_a =$ EffectiveCoherence($\boldsymbol{A}$, $l-1$)
- $\mu_b =$ EffectiveCoherence($\boldsymbol{B}$, $l-1$)
- μ_m, μ_d are calculated as always
- If l == 1: return μ_d
- Else: return $\mathbf{min}\left\{\mu_d\,,\ \frac{\mu_a+\sqrt{\mu_a^2+\mu_m^2}}{2}\right\}$

Interference: Results

- Artificial sensing matrix D was constructed by shuffling together the columns of two orthogonal matrices – there *is* inherent structure in D, but it is not apparent
- The splitting algorithm finds the perfect split, separating D into the constituent matrices we started with
- In the process, the bounds for recovery were improved significantly

Unsplit dictionary			
N	200	500	1000
μ_d	0.141	0.0894	0.0632
Upper bound on k	4.036	6.090	8.406
$\hat{\delta}$	6.930	31.216	44.209
Split dictionary			
μ_a	0	0	0
μ_b	0	0	0
μ_m	0.141	0.0894	0.0632
μ_d	0.141	0.0894	0.0632
Effective coherence $\tilde{\mu_d}$	0.720	0.0448	0.0317
Upper bound on k	7.521	11.648	16.289
$\hat{\delta}$	3.536	15.652	22.136

Interference: Conclusions and Future Work

- The two-way split algorithm finds the optimal induced split in the dictionary.
- Unfortunately, this is not true of the recursive algorithm. However, it is guaranteed to be at least as good as the two-way split algorithm, since that case is subsumed here.
- Note that in either case, the recovery algorithm is unchanged. But we have improved the bounds on recovery for a given dictionary
- Future avenues of work include developing a multi-way splitting algorithm that *is* optimal, in terms of recovery bounds

Thank You

Appendix

Complete Dictionary Splitting Algorithm

1: **for** $v \in V$ **do**
2: $colour(v) \leftarrow 0$
3: $L(v) \leftarrow$ edges from v, in decreasing order of weight
4: **end for**
5: E_x = cut edges $\leftarrow \phi$
6: Q = list of edges to process $\leftarrow \phi$
7: $processed \leftarrow \phi$

8: $e = uv \leftarrow$ heaviest edge in G
9: $colour(u) \leftarrow 1$
10: $L(u) \leftarrow L(u) \setminus \{e\}$
11: $Q \leftarrow L(u)$
12: $processed \leftarrow \{u\}$

13: **while** size$(processed) < n$ **do**
14: **if** one of $\{u, v\}$ (say v) is s.t. $colour(v) == 0$ **then**
15: **if** $colour(u) == 1$ **then** $colour(v) \leftarrow 2$
16: **if** $colour(u) == 2$ **then** $colour(v) \leftarrow 1$
17: $L(v) \leftarrow L(v) \setminus \{e\}$
18: Merge $L(v)$ into Q
19: $processed \leftarrow processed \cup \{v\}$
20: **end if**
21: **if** $colour(u)! = colour(v)$ **then** $E_x \leftarrow E_x \cup \{e\}$
22: $e = uv \leftarrow$ Pop heaviest edge from Q
23: **end while**

24: $\mathbf{V}_a \leftarrow \{v \mid colour(v) == 1\}$
25: $\mathbf{V}_b \leftarrow \{v \mid colour(v) == 2\}$
26: Output $\mathbf{V}_a$, $\mathbf{V}_b$

Optimality of Two-way Splitting Algorithm

- Consider the heaviest edge in G, say e* = uv. At the end of any possible assignment, either
  $max\{\mu_a\ \mu_b\} = w(e)$, or $\mu_m = w(e)$
- Using the 'effective μ_d reasoning, at the latter case is always advantageous
- The same reasoning can be applied to the next heaviest edge, the ones after, and so on, until it is impossible to add an edge to the cut - because doing so will create an inconsistent assignment of the vertices.
- In other words, adding edge e to the cut will create an odd length cycle, which cannot possibly occur in a graph cut
- But our candidate edge list only includes edges with *exactly* one end point assigned
- It can also be verified by an exchange argument that the edge e that is not included in the cut must be the lightest edge in the cycle.