1 Recap

Let $G(L, R, E)$ be an $(n \times 3n)$ $d$-left regular Bipartite graph whose parameter is $(\frac{n}{d}, (1 - \epsilon)d)$-left expander. Left and right partition of $G$ consists of $L$ and $R$ vertex set respectively. We can treat the nodes on the left partition as the bits in the codework and then each node on the right hand side corresponds to a parity check. We proved that the minimum relative distance for the corresponding expander code is $\geq \frac{2\gamma(1-\epsilon)}{d}$.

2 Decoding Expander Codes

Let $y \in \mathbb{F}_2^L$ be the received message. We call a parity check node $v \in R$ satisfied if $\sum_{u \in \Gamma(v)} y_u = 0$

Algorithm for decoding: Informally, we look at each bit in position $i$ in $y$, $y_i$, and check whether the associated parity check on the right partition is satisfied or not. If more than half of the neighbours of $y_i$ is unsatisfied, we flip it and we keep on doing this till there is none to satisfy.

Formally: while $\exists u \in L$ s.t. if there are more unsatisfied parity check nodes than satisfied ones in neighbours of $u$ in $R$, denoted by $\Gamma(u)$, Flip $y_u$.

Claim 1. If $\# \text{ of errors in } y \in [1, \frac{n}{d}]$, then $u$ exists.

Proof. Let $T$ be the set of error positions in $y \subseteq L$. If $|T| \in [1, \frac{n}{d}] \implies |\Gamma(T)| > (1-\epsilon)d(T)$, by expander property. Let us denote by $u(T)$ the set of nodes in $\Gamma(T)$ with unique neighbours in $T$.

Intuitively, we fixed a set $T$ and we are looking at all of its neighbours, $\Gamma(T)$. All the neighbours in $\Gamma(T)$ might have different neighbours but $u(T)$ is the set of neighbours in $\Gamma(T)$ whose neighbour set has only one neighbour back in $T$. If $T$ is the error positions, then all those in $u(T)$ are not satisfied because there is exactly one error. So the nodes in $u(T)$ are not satisfied. Intuitively we would want to show that $u(T)$ is big.
Now,
\[ |\text{edges}(T, \Gamma(T))| \geq 2(\|\Gamma(T) - u(T)\|) + |u(T)| \]
\[ = 2(\|\Gamma(T)\|) - |u(T)| \]
\[ \geq 2(1 - \epsilon)d|T| - |u(T)| \]
Also
\[ |\text{edges}(T, \Gamma(T))| \leq d|T| \]

And
\[ d|T| \geq 2(1 - \epsilon)d|T| - |u(T)| \]
\[ \implies |u(T)| \geq (1 - 2\epsilon)d|T| \]
\[ \implies |u(T)| > \frac{1}{2}d|T|, \text{ by assumption that } \epsilon < 1/4 \]

So \( \exists u \in T, \text{ s.t. } |\Gamma(u) \cap u(T)| > \frac{1}{2}d \). And we are done. \( \square \)

Alternative proof sketch: Another way to prove the previous claim is to prove by contradiction. In this case the proof sketch is as follows: Let us assume \( \forall u \in T, \|\Gamma(u) \cap u(T)\| \leq \frac{1}{2}d \).

Now \( \sum_{u \in T} |\Gamma(u) \cap u(T)| \leq \frac{1}{2}d|T| \). However \( |\Gamma(T) \cap u(T)| \leq \sum_{u \in T} |\Gamma(u) \cap u(T)| \) and we reach a contradiction.

The next claim shows that the algorithm can correct errors as long as we start with an error message with \# errors no greater than \( \frac{\gamma(1 - 2\epsilon)n}{d} \).

Claim 2. If we start with \( y \) with \( \leq \frac{\gamma(1 - 2\epsilon)n}{d} \) errors, the algorithm will never reach a word with \( \frac{\gamma}{d}n \) errors.

Proof. Assume we could reach codeword \( y' \) with \( T \), set of error positions, where \( T = \frac{\gamma}{d}n \).

Then the \# unsatisfied nodes \( \geq u(T) \geq (1 - 2\epsilon)d|T| = (1 - 2\epsilon)\gamma n \).

However we started with \# unsatisfied nodes \( \leq d\frac{\gamma(1 - 2\epsilon)}{d}n = \gamma(1 - 2\epsilon)n \).

But as the algorithm progresses, the number of unsatisfied nodes strictly decreases. So we reach a contradiction. \( \square \)

The algorithm can correct errors in \( O(n^2) \) time.
3 Linear Programming (LP)

Definition 1. Linear programming (LP) is defined as follows: given a set of constraints $K \subseteq \mathbb{R}^n$ where

$$K = \begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \leq b_2 \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \leq b_m \end{cases}$$

Where $a_{ij}, b_i \in \mathbb{Q}, \forall i \in [m], j \in [n]$

Goal:

1. Decide whether $K \neq \emptyset$
2. Maximize $c_1x_1 + \cdots + c_nx_n$ s.t. $x \in K$

Remark 1. 1. Could allow “≤”, “≥” and “=”.
2. Don’t allow “<” and “>”

Theorem 2. [Kha79] LP is solvable in polynomial time (Polynomial in terms of $m, n$ and description of $a_i$s)

3.1 Application

3.1.1 MaxFlow Problem

Input: Directed graph $G = (V, E)$ with source $s$, sink $t \in V$, capacity $C_{uv}$ s.t. $\forall (u, v) \in E, C_{uv} \in \mathbb{Q}^{\geq 0}$.

Goal: Finding maximum possible flow from $s$ to $t$.

LP formulation:

$$\max \sum_{u:s \rightarrow u} f_{su} - \sum_{u:s \rightarrow s} f_{us}$$

subject to the following constraints:

$$\begin{cases} \forall u \notin \{s, t\}, \sum_{v:u \rightarrow v} f_{uv} - \sum_{v:u \rightarrow u} f_{vu} = 0 \text{ (called flow conservation)} \\ \forall (u, v) \in E, f_{uv} \geq 0, f_{uv} \leq C_{uv} \end{cases}$$
3.2 Duality

Next we discuss about how to formulate the dual problem of an LP form.

The previous LP formulation can be generalized as follows:

\[
\text{maximize } C^T x \\
\text{subject to } Ax \leq b.
\]

For every \( x_i \), introduce \( x_i^+, x_i^- \) s.t. \( x_i = x_i^+ - x_i^- \), \( x_i^+, x_i^- \geq 0 \), so that we can convert LP into standard form or primal form: maximize \( C^T x \) subject to \( Ax \leq b, x \geq 0 \).

3.2.1 A Toy Example (TP)

Consider the following LP:

\[
\text{Maximize } 4x_1 + 7x_2 \\
\text{subject to }
\begin{align*}
    x_1 + 3x_2 &\leq 10 \\
    5x_1 + 2x_2 &\leq 18 \\
    3x_1 + x_2 &\leq 12 \\
    x_1, x_1 &\geq 0
\end{align*}
\]

This is in primal form.

Observation regarding TP: upper bounded by the value 38 (because \( 2 \times (3) + (4) \implies 7x_1 + 8x_2 \leq 38 \)) We also know \( 7x_1 + 8x_2 \geq 4x_1 + 7x_2 \)

Observation 2: \( 2 \times (3) + (5) \implies 5x_1 + 7x_2 \leq 32. \)

In general: \( y_1 \times (3) + y_2 \times (4) + y_3 \times (5) \implies (y_1 + 5y_2 + 3y_3)x_1 + (3y_1 + 2y_2 + y_3)x_2 \leq 10y_1 + 18y_2 + 12y_3. \)

So we want to minimize \( 10y_1 + 18y_2 + 12y_3 \) subject to

\[
\begin{align*}
    y_1 + 5y_2 + 3y_3 &\geq 4 \\
    3y_1 + 2y_2 + y_3 &\geq 7 \\
    y_1, y_2, y_3 &\geq 0
\end{align*}
\]

This is the dual of TP (lets call it TD).

**Fact:** TD \( \geq \) TP
3.2.2 Dual Formulation

The dual formulation of an LPP is as follows:

\[
\text{minimize } b^T y \text{ subject to } y \geq 0, A^T y \geq c
\]

**Theorem 3.** (Weak duality theorem) Dual (D) is always upper bound of the primal (P) i.e. \( (D) \geq (P) \).

**Proof.** \( C^T y \leq y^T Ax \leq y^T b \)

**Theorem 4.** (Strong Duality theorem) If either (P) or (D) is feasible, then \( (P) = (D) \).

**Fact:** Dual of dual is the primal itself.

3.3 Dual of the MaxFlow Problem

The **primal** form of the maxFlow problem:

\[
\max f \sum_{u:s \rightarrow u} f_{su} - \sum_{u:s \rightarrow s} f_{us}
\]

subject to

\[
\begin{align*}
\forall u \neq s, t & \quad \sum_{v:u \rightarrow v} f_{uv} - \sum_{v:u \rightarrow u} f_{vu} \leq 0 \quad (10) \\
\forall (u, v) \in E & \quad \sum_{v:u \rightarrow v} f_{vu} - \sum_{v:u \rightarrow u} f_{uv} \leq 0 \quad (11) \\
f_{uv} & \leq C_{uv} \quad (12) \\
f_{uv} & \geq 0 \quad (13)
\end{align*}
\]

Let \( y^{(1)}_v, y^{(2)}_u \) and \( y_{uv} \) denote the corresponding variable in the constraints of the dual formulation for Equations 10, 11 and 12 respectively.

The **dual** form of the maxflow can then be stated as follows:

\[
\min \sum_{(u,v) \in E} C_{uv} - y_{uv}
\]

subject to:

\[
\begin{align*}
\forall (s, v) \in E & \quad y^{(1)}_v - y^{(2)}_v + y_{sv} \geq 1 \quad (14) \\
\forall (v, s) \in E & \quad -y^{(1)}_v + y^{(2)}_v + y_{vs} \geq -1 \quad (15) \\
\forall (t, v) \in E & \quad y^{(1)}_v - y^{(2)}_v + y_{tv} \geq 0 \quad (16) \\
\forall (v, t) \in E & \quad -y^{(1)}_v + y^{(2)}_v + y_{vt} \geq 0 \quad (17) \\
\forall (u,v) \in E, u \neq s, t & \quad y^{(1)}_v - y^{(2)}_v - y^{(1)}_u + y^{(2)}_u + y_{uv} \geq 0 \quad (18)
\end{align*}
\]
Let $y_v = y_v^{(1)} - y_v^{(2)} \in \mathbb{R}$

So the constraints become:
\[
\begin{align*}
    y_v + y_{sv} &\geq 1 \implies y_v + y_{sv} - 1 \geq 0 \\
    -y_v + y_{vs} &\geq -1 \implies -y_v + y_{vs} + 1 \geq 0 \\
    y_v + y_{tv} &\geq 0 \\
    -y_v + y_{vt} &\geq 0 \\
    y_v + y_u + y_{uv} &\geq 0
\end{align*}
\]

Let $y_s = 1, y_t = 0$. Then Equation 19 $\implies y_v - y_s + y_{sv} \geq 0$.

Equation 20 $\implies y_s - y_v + y_{vs} \geq 0$

Equation 21 $\implies y_v - y_t + y_{tv} = 0$

Equation 22 $\implies y_t - y_v + y_{vt} \geq 0$

So $\forall (u, v) \in E, y_v - y_u + y_{uv} \geq 0, y_s = 1, y_t = 0, y_{uv} \geq 0$

### 3.4 The Mincut Problem

Let us partition a graph in two sides $S$ and $T$. $y_v \in \{0, 1\}, y_s = 1$, $y_t = 0$, $y_{uv}$ is an indicator variable to indicate whether edge $(u, v)$ is on the cut.

\[y_{uv} \geq y_u - y_v \geq 0\]

**Definition 5.** The mincut problem can be formulated as follows:

\[
\text{minimize} \sum_{(u,v) \in E} C_{uv} y_{uv}
\]

Note: Mincut is integer programming problem but dual of maxflow is LPP. Integer programming is NP. So we use relaxation.

**Claim 3.** Dual of maxflow \(\leq\) mincut

**Claim 4.** mincut \(\leq\) Dual of maxflow

**Proof.** We proof by first following a rounding procedure to obtain integer solutions as follows:

Let us suppose that we solve the mincut problem and have a bunch of solutions. We can put all these solutions on an axis line and all these solutions lie within the range of $[0, 1]$. Of course these are fractional numbers but we would like to make them 0 or 1 and thus
called rounding procedure. The procedure we follow is a randomization procedure: choose a threshold value \( \theta \in [0, 1] \) uniformly. Let

\[
y^* = \begin{cases} 1 & \text{if } y_i \leq \theta \\ 0 & \text{if } y_i > \theta \end{cases}
\]

Now we analyze it. \( \forall (u,v) \in E \), \((u,v)\) in “cut” edge if \( y^*_u = 1 \), \( y^*_v = 0 \)

So,

\[
E[\text{cut}] = E \sum_{(u,v) \in E} C_{uv} 1[ y^*_u = 1, y^*_v = 0 ] \\
= \sum_{(u,v) \in E} C_{uv} Pr [ y^*_u = 1, y^*_v = 0 ] \\
\leq \sum_{(u,v) \in E} C_{uv} \max \{ y_u - y_v, 0 \} \\
\leq \sum_{(u,v) \in E} C_{uv} y_{uv}
\]

Which is the dual.

So our rounding procedure results in the fact that the expected cut is the dual. \( \square \)

As we see there must exists a cut. However the dual is the lower bound. This means the dual of maxflow and the mincut problem are the same.

**References**