Algorithms and Complexity: Midterm

Due on April 5, 2019 at the start of class

Professor Bradford

Patrik Sokolowski

Problem 1

Recall our definition of expander graphs.

Assuming an undirected bipartite graph $G = (\mathcal{I} \cup \mathcal{O}, E)$ where the vertex partition $\mathcal{I}$ and $\mathcal{O}$ are so that

$$|\mathcal{I}| \quad = \quad |\mathcal{O}| \quad = \quad p^2$$

for some prime $p > 2$.

Now, take all inputs $\mathcal{I}$ as $\mathcal{I} = \bigcup_{j=0}^{p-1} I_j$ where each I_j is an input block so $|I_j| = p$. Likewise, for all outputs $\mathcal{O}$ it must be that $\mathcal{O} = \bigcup_{j=0}^{p-1} O_j$ where each O_j is an output block where $|O_j| = p$.

Define $\mathbb{Z}_p^+ = \{0, 1, \cdots, p-1\}$ where addition is computed mod p.

For any input node $(j, k) \in I_j$ is position k in block j where $j \in \mathbb{Z}_p^+$ and $k \in \mathbb{Z}_p^+$. The graph G_4 has the following sets of edges:

1. *Identity*: $\boldsymbol{id}(j,k) \longrightarrow (j,k)$.
2. *Local Shift*: $\boldsymbol{loc}(j,k) \longrightarrow (j,\ (j+k+1) \bmod p)$.
3. *Global Shift*: $\boldsymbol{g}(j,k) \longrightarrow ((j+k) \bmod p,\ k)$.

These edges are directed from the inputs $\mathcal{I}$ to the outputs $\mathcal{O}$. The adversary can only select inputs to prevent expansion among the outputs. In general, for block I_j, an adversary selects input elements $\widehat{I}_j \subseteq I_j$ and these selected inputs have output neighbors $\widehat{O} \subseteq \mathcal{O}$.

The global shift $\boldsymbol{g}$ has the inverse $\boldsymbol{g}^{-1}(j,k) = (j-k, k)$ since $\boldsymbol{g}(j+k-k, k) = (j,k)$ computed mod-p. Further, $\boldsymbol{g}_1(j,k), \boldsymbol{loc}_1(j,k), \boldsymbol{id}_1(j,k)$, denotes the first component of the pair, while $\boldsymbol{g}_2(j,k), \boldsymbol{loc}_2(j,k), \boldsymbol{id}_2(j,k)$.

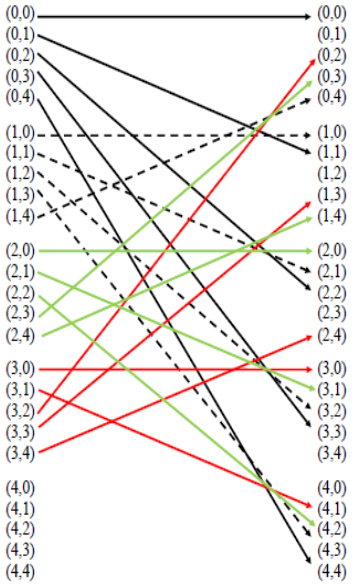

Figure 1: Selected edges of global permutation for $p = 5$

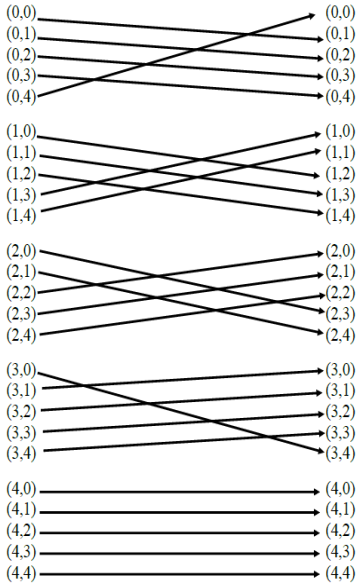

Figure 2: Local edges from ***loc*** for $p = 5$

Our focus is on showing expansion with exactly $p(p-1)/2$ inputs.

Question 1: Show that given the local and identity permutations 'most' of the inputs must follow each local pattern.

Question 2: Show that given the global permutations 'most' of the inputs must follow the global pattern.

Question 3: How may we combine the local, id and global permutations to show we have expansion?

Problem 2

P 246, # 4.

4. You've been working with some physicists who need to study, as part of their experimental design, the interactions among large numbers of very small charged particles. Basically, their setup works as follows. They have an inert lattice structure, and they use this for placing charged particles at regular spacing along a straight line. Thus we can model their structure as consisting of the points $\{1, 2, 3, \ldots, n\}$ on the real line; and at each of these points j, they have a particle with charge q_j. (Each charge can be either positive or negative.)

They want to study the total force on each particle, by measuring it and then comparing it to a computational prediction. This computational part is where they need your help. The total net force on particle j, by Coulomb's Law, is equal to

$$F_j = \sum_{i<j} \frac{Cq_iq_j}{(j-i)^2} - \sum_{i>j} \frac{Cq_iq_j}{(j-i)^2}$$

They've written the following simple program to compute F_j for all j:

```
For j = 1, 2, ..., n
  Initialize F_j to 0
  For i = 1, 2, ..., n
    If i < j then
      Add C q_i q_j / (j - i)^2 to F_j
    Else if i > j then
      Add -C q_i q_j / (j - i)^2 to F_j
    Endif
  Endfor
  Output F_j
Endfor
```

It's not hard to analyze the running time of this program: each invocation of the inner loop, over i, takes $O(n)$ time, and this inner loop is invoked $O(n)$ times total, so the overall running time is $O(n^2)$.

The trouble is, for the large values of n they're working with, the program takes several minutes to run. On the other hand, their experimental setup is optimized so that they can throw down n particles, perform the measurements, and be ready to handle n more particles within a few seconds. So they'd really like it if there were a way to compute all the forces F_j much more quickly, so as to keep up with the rate of the experiment.

Help them out by designing an algorithm that computes all the forces F_j in $O(n \log n)$ time.

We can use convolution.
If we have a vector set of $(q_1, q_2, ..., q_n)$ and a vector set of $(n^{-2}, (n-1)^{-2}, ..., 1/4, 0, -1, -1/4, ..., -n^{-2})$
For each j we can put the convolution of both vector sets and have the equivalent of the coulumb equation:

$\sum_{i<j} \frac{q_i}{(j-i)^2} + \sum_{i<j} \frac{-q_i}{(j-i)^2}$

We simply multiply every $q_i with C q_i$ in the end to get the full F_j.
This will get the faster running time they wanted.