Charles Zheng CME 323 HW 3

**1.** a. Consider a "broom" graph with nodes $1, \cdots, n$ with edges $(1,2), \ldots, (n_1 - 1, n_1)$ and edges $(n_1, i)$ for all $n_1 < i < n$, and finally an edge $(n-1, n)$. Then the worst-case $k$ is $k = n_1 + 1$ and the worst-case IDs for this case are where $ID(i) = n - i + 1$ for $i = 1, \ldots, n$. Then what will happen is that in the first $k$ iterations, each node except for node $n$ will carry the ID of node 1, and all nodes $i \geq k$ (other than node $n$) will be of distance exactly $k$ from node 1. Hence $n - k$ nodes will be added to the graph. The algorithm terminates on the $(k+1)$st iteration when the message from node $(n-1)$ carrying the ID of node 1 reaches node $n$.

b. For large (and even) $n_1$, the best ID distribution would be to have $ID(n_1/2) = n$, and the other ids so that $ID(i) > ID(j)$ for all $i$ satisfying $i = n_1/2 + 2\ell k^*$ for integer $\ell$ and $i \leq n_1$ and all $j$ not satisfying those conditions, where $k^* = \sqrt{\frac{3}{2} n_1}$. Here the optimal value of $k$ is also $k^*$. In that case the total number of unique IDs at step $k = k^*$ is $\sqrt{\frac{2}{3} n_1}$, and the corresponding partitions are arranged along the line $1, 2, \ldots, n_1, n-1, n$. In the $k+1$th iterate, the maximal ID travels a distance of 1 to reach the edge of another partition, in the $k+2$nd it travels a distance of $k^*$ to reach the node with the original ID of that partition, and in the $k+3$rd the maximal ID has completely replaced the ID of that partition. This happens in both left and right directions. In general, it takes 3 iterations for the maximal ID to spread a distance of $2k^* + 1$ in either direction. The maximum distance of the central node $n_1/2$ any other node is $n_1/2$. Hence it takes about $k^* + n_1/4k^* = n_1(\sqrt{3/2} + \sqrt{1/6})$ iterations.

c. Initialize each node with two different unique random IDs. In the first phase (first $k$ iterations), use the first ID to implement the original algorithm. After the $k$th iteration, propagate the both pairs of IDs, along with a distance counter initialized to 1. In this phase, nodes ignore all messages where the first ID does not match the max ID from the first phase. If a message with the correct first ID has a higher second ID than the previous highest, the node will transmit a copy of that message to all its neighbors in the next round, while also incrementing the distance counter in the message by 1. This continues for $k$ iterations. In the third phase, any node whose max ID matches their original ID links to the max ID from the second phase if and only if the distance counter for that message is greater than $\alpha k$. In the fourth phase, the algorithm continues as it did in the first phase.

d. Call all of the nodes whose original IDs are still stored on some node

in the on the $k$th iteration as 'centroids'. The distance between neighboring centroids is between $k+1$ and $2k+1$. We can conclude further that the distribution is uniform since e.g. the neighoring centroid on the left of any centroid is the node with the max ID within that window of nodes between $k+1$ and $2k+1$ to the left of the centroid, and all of those nodes within the window have IDs which are independent of the centroid. Therefore the distance between centroids is on average $3k/2$. As a result, the average number of such centroids is $2n/3k$. The distribution of distances from the centroid to nodes with matching max IDs is a mixture of uniform distributions. Hence the probability of making a link is roughly proportional to $k(1-\alpha)$. If we sum $k, k-1, \ldots, 1$, we get $k(k-1)/2$. Hence the probability of making a link is $2(1-\alpha)/(k-1)$. The speedup of making such links is the time saved per link, which is $\alpha k$, times the number of such links made, which is the probability of making a link times the number of centroids, $4n(1-\alpha)/(3(k-1))$. Hence the benefit of making such links is $4n(1-\alpha)\alpha k/(3(k-1))$ iterations. However, there is also a cost associated with adding $4n(1-\alpha)/(3(k-1))$ edges to the graph, so we have to compare the cost of one iteration of message passing to the cost of adding edges. The number of messages passed in each iteration is roughly two times the number of unique IDs remaining, and hence is initially $2n/3k$ and decreases exponentially from there. Overall this means that the cost of adding edges is comparable to 2 or more iterations. But with $n$ and $k$ growing, the savings in iterations overpowers the cost of adding edges. Therefore, neglecting the cost of adding edges, the optimal $\alpha = 1/2$. The optimal $k$ is trickier to calculate since we had to assume growing $k$ to justify ignoring the cost of adding edges. On the other hand, the equation resulting from that assumption favors small $k$. In reality it would be advantageous to choose $k$ just large enough to control the number of added edges, and then repeat the procedure of adding edges periodically, until it became too costly to justify repeating the procedure in part c.

**2.** Let $w^*$ be an optimal weight vector with $||w^*|| = 1$ and $\min |x^T w^*| \geq \delta$. Let $w^{(0)} = 0$ and $w^{(k)}$ denote $w$ after the $k$th mistake, and $x^{(k)}$ be the feature of the $k$th mistake, so that

$$\text{sign}(w^{(k)} x^{(k)}) = -\text{sign}(w^* x^{(k)})$$

and

$$w^{(k+1)} = w^{(k)} - \text{sign}(w^{(k)} x^{(k)}) w^{(k)}$$

Then we have

$$||w^{(k+1)}||^2 = ||w^{(k)} - \text{sign}(w^{(k)} x^{(k)}) x^{(k)}||^2 = ||w^{(k)}||^2 + ||x^{(k)}||^2 - |w^{(k)} x^{(k)}| \leq ||w^{(k)}|| + R^2$$

and also

$$\langle w^{(k+1)}, w^* \rangle = \langle w^{(k)} + \text{sign}(w^* x^{(k)}) x^{(k)}, w^* \rangle = \langle w^{(k)}, w^* \rangle + |\langle x^{(k)}, w^* \rangle| \geq \delta$$

These two facts imply by induction that

$$||w^{(k)}|| \leq R\sqrt{k}$$

and

$$\langle w^{(k)}, w^* \rangle \geq \delta k$$

But now observe that

$$1 \geq \cos(w^{(k)}, w^*) = \frac{\langle w^{(k)}, w^* \rangle}{||w^*||||w||} \geq \frac{\delta k}{R\sqrt{k}} = \frac{\delta\sqrt{k}}{R}$$

Hence we get

$$k \leq \frac{R^2}{\delta^2}$$

This means that the number of mistakes is bounded by $\frac{R^2}{\delta^2}$, since otherwise we would arrive at a contradiction.

**3.** With probability $\min(1, \gamma / \min(||c_j||^2, ||c_k||^2))$, emit $(j,k) \to a_{ik}a_{jk}$.

a.

$$\begin{aligned} \text{shuffle} &\leq \sum_{i=1}^{n} \sum_{j=i+1}^{n} \gamma \frac{\#(c_i, c_j)}{\min(\#c_i, \#c_j)} \\ &\leq \sum_{i=1}^{n} \sum_{j=i+1}^{n} \gamma \#(c_i, c_j) \left( \frac{1}{\#c_i} + \frac{1}{\#c_j} \right) \end{aligned}$$

and the rest of Reza's proof goes through as before.

b. If combiners are used, shuffle size is

$$\min(Bm^2, nL\gamma)$$

since the output is bounded in size by $m^2$ per mapper. But if the output is so sparse that the post-combine sparsity is still less than $m^2$, the original DIMSUM result still applies.

**4.** a. The communication cost of the shuffle is $O(n \log n)$ and the communication cost of the reduce is $O(k)$.

b. The total communication cost is $O(Tk)$ and the total time it takes to reduce + broadcast is $O(T \log k)$.