CS 671: Graph Streaming Algorithms and Lower Bounds **Rutgers: Fall 2020**

Problem set 4

Due: 11:59PM, October 6, 2020

Problem 1. Consider a *hypergraph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of rank r s.t. each *hyperedge* $e = (u_1, \ldots, u_r) \in \mathcal{E}$ connects exactly r vertices $u_1, \ldots, u_r \in \mathcal{V}$ together (a hypergraph of rank 2 is a simple graph). A *hypermatching* $\mathcal{M}$ in $\mathcal{G}$ is a collection of hyperedges that do not share any vertices. In the semi-streaming setting for hypergraphs, we again assume $\mathcal{V} := [n]$ and each hyperedge e in $\mathcal{E}$ arrives in the stream; we require the algorithm to use space $O(n \cdot \text{polylog}(n))$ as before—note that this space is in particular enough to write down all edges of a hypermatching as its size can only be $O(n/r)$ and each can be represented in $O(r)$ space.

(*i*) Design a semi-streaming r-approximation algorithm for the problem of finding a *maximum cardinality* hypermatching.

(*ii*) Design a semi-streaming $O(r^2)$-approximation algorithm for the problem of finding a *maximum weight* hypermatching.

(*iii*) Design a semi-streaming $(1+\varepsilon)r$-approximation algorithm for the problem of finding a *maximum weight* hypermatching.

Note: If you solved the last part (even if it is an $O(r)$-approximation algorithm) you do not need to solve either of the previous two parts. Also, you may assume that the weights of edges are $\text{poly}(n)$-bounded.