Streaming and Massively Parallel Algorithms for Euclidean Max-Cut

Nicolas Menand [1] Erik Waingarten [1]

[1]University of Pennsylvania

We devise algorithms for Euclidean Max-Cut in the streaming and massively parallel computation (MPC) settings, which can provide oracle access to an approximately optimal cut. The core procedure in our algorithms is a "parallel" and "subsampled" greedy decision, based on a greedy algorithm for max-cut in dense graphs [MS08].

Parallel and Subsampled Greedy Assignment

Consider a fixed dataset $X = \{x_1, \ldots, x_n\}$ of n points in Euclidean space. works as follows:

- A discrete time axis begins at $t = 0$ toward an end time t_e. Each point x_i generates a timeline $\mathbf{A}_i \in \{0,1\}^{t_e}$: at each time $t = 1, \ldots, t_e$, it "activates" by sampling $\mathbf{A}_{i,t} \sim \mathrm{Ber}(\boldsymbol{w}_i^t)$ with probability proportional to it's weight and $1/t$.
- Alongside a timeline, each x_i samples a mask $\mathbf{K}_i \in \{0,1\}^{t_e}$: at each time t, x_i is "kept" by sampling $\mathbf{K}_{i,t} \sim \mathrm{Ber}(\gamma_t)$, where γ_t is 1 at $t \le t_0$ and $1/t$ for $t > t_0$.

A point assigns itself to inside/outside the cut the moment it is first activated.

- Points activated by time t_0 try all cut assignments.
- Points activated after t_0 are assigned greedily by maximizing a weighted contribution to already-assigned points which were simultaneously "activated" and "kept" before time t.

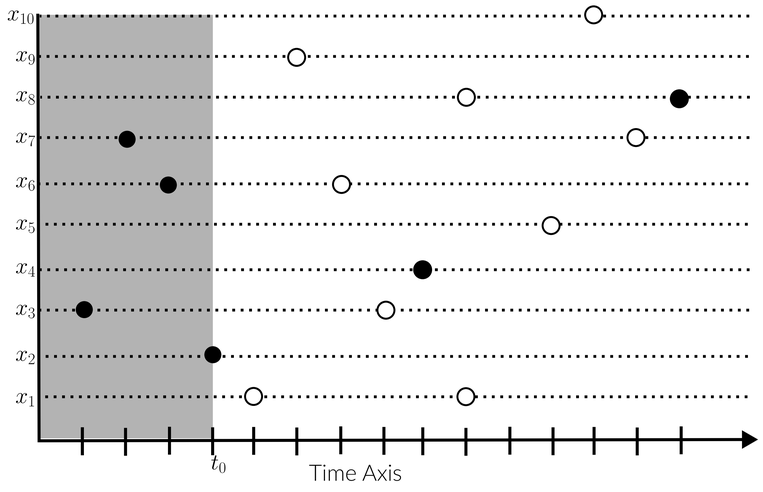

Figure 1. Representation of the timelines for each point. Each horizontal dotted line corresponds to the timeline of each point $x_1, \ldots, x_{10}$. Time t_0 appears in the timeline with times up to t_0 in gray. For a point x_i, a dot on the timeline corresponds to an activation, and a solid dot to an activation which is also kept.

Theorem 1: MPC

Theorem 1: *There is a $O(1)$-round fully-scalable MPC algorithm which outputs a $(1+\varepsilon)$-approximate Euclidean max-cut using $O(nd) + n \cdot \mathrm{poly}(\log n/\varepsilon)$ total space.*

- Each point computes its own weight in $O(1)$ rounds using cascaded $\ell_1(\ell_p)$-sketches.
- Since each point x_i computes its own weight, it can generate a timeline $\mathbf{A}_i$ and mask $\mathbf{K}_i$.
- If there exists a time $t \le t_e$ where a point is simultaneously activated and kept, it is communicated across all machines along with its weight $\boldsymbol{w}_i$ and activation time t.
- A root machine finds the best assignment of the points activated by t_0 (using a few additional samples) and shares this assignment to all machines.
- The remaining points assign themselves by maximizing a weighted contribution to the activated and kept points.

Theorem 2: Dynamic Streams

Theorem 2: *There is a dynamic streaming algorithm using $\mathrm{poly}(d \log \Delta/\varepsilon)$ space which provides oracle access to a $(1+\varepsilon)$-approximate Euclidean max-cut.*

In a dynamic stream, we need to use a geometric sampling sketches [CJK23] to produce samples $\boldsymbol{x} \sim X$ with sampling probability proportional to it's weight. This causes the following changes:

- We sample one mask $\mathbf{K} \in \{0,1\}^{t_e}$ to determine which times will have points simultaneously activated and kept (instead of independent masks for each point).
- If $\mathbf{K}_t = 0$, no point is activated and kept at that time. If $\mathbf{K}_t = 1$, we use a geometric sampling sketch to sample $\boldsymbol{x}_i \sim X$ and activate it by setting $\mathbf{A}_{i,t} = 1$.
- On a query x_j, we use the sketch to estimate the weight $\boldsymbol{w}_j$ and generate the remainder of the timeline $\mathbf{A}_j$; this specifies the activation time $\boldsymbol{t}_j$, which is used to simulate the "parallel" and "subsampled" greedy assignment.

Space

We expect one point to activate at each time t, and γ_t points (which is less than 1) to both be "activated and kept" at time t. Importantly, we only need to store points which are simultaneously activated and kept. By setting $t_0 = O(\log(n)/\varepsilon^2)$ and $t_e = O(n/\varepsilon)$, we expect $O(\log(n)/\varepsilon^2)$ many points to be stored while recovering the $(1+\varepsilon)$ approximation guarantee. In the streaming setting, $X \subset [\Delta]^d$, giving $\mathrm{poly}(d \log(\Delta)/\varepsilon)$ total space.

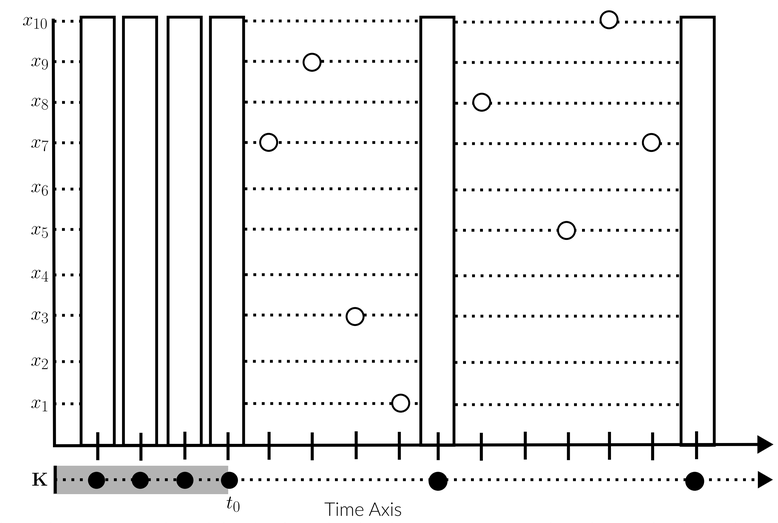

Figure 2. Representation of the timelines for the dynamic streaming algorithm. Right above the time axis on the bottom, we sample one mask $\mathbf{K}$ which determines which times contain points which are activated and kept. Whenever $\mathbf{K}_t = 1$, the rectangle above it represents a geometric sampling sketch used to determine which point is activated and kept. After geometric sampling sketches are generated, the horizontal dotted lines represents the remaining timelines to be generated for each point.

References

[CJK23] Xiaoyu Chen, Shaofeng H.-C. Jiang, and Robert Krauthgamer. Streaming euclidean max-cut: Dimension vs data reduction. In *Proceedings of the 55th ACM Symposium on the Theory of Computing (STOC '2023)*, pages 170–182, 2023.

[MS08] Claire Mathieu and Warren Schudy. Yet another algorithm for dense max cut: go greedy. In *Proceedings of the 19th ACM-SIAM Symposium on Discrete Algorithms (SODA '2008)*, 2008.