Time and Space Optimal Massively Parallel Algorithm for the 2-Ruling Set Problem

Mélanie Cambus
Aalto University, Finland

Fabian Kuhn
University of Freiburg, Germany

Shreyas Pai
Aalto University, Finland

Jara Uitto
Aalto University, Finland

Abstract

In this work, we present a constant-round algorithm for the 2-ruling set problem in the Congested Clique model. As a direct consequence, we obtain a constant round algorithm in the MPC model with linear space-per-machine and optimal total space. Our results improve on the $O(\log\log\log n)$-round algorithm by [HPS, DISC'14] and the $O(\log\log\Delta)$-round algorithm by [GGKMR, PODC'18]. Our techniques can also be applied to the semi-streaming model to obtain an $O(1)$-pass algorithm.

Our main technical contribution is a novel sampling procedure that returns a small subgraph such that *almost* all nodes in the input graph are adjacent to the sampled subgraph. An MIS on the sampled subgraph provides a 2-ruling set for a large fraction of the input graph. As a technical challenge, we must handle the remaining part of the graph, which might still be relatively large. We overcome this challenge by showing useful structural properties of the remaining graph and show that running our process twice yields a 2-ruling set of the original input graph with high probability.

2012 ACM Subject Classification Theory of computation → Massively parallel algorithms; Theory of computation → Streaming models

Keywords and phrases Ruling Sets, Parallel Algorithms, Congested Clique, Massively Parallel Computing, Semi-Streaming

Digital Object Identifier 10.4230/LIPIcs.DISC.2023.11

Related Version *Full Version*: `https://arxiv.org/abs/2306.00432`

Funding *Mélanie Cambus*: Academy of Finland, Grant 334238.
Shreyas Pai: Academy of Finland, Grant 334238.

1 Introduction

In this paper, we design and analyze a parallel algorithm for finding ruling sets. For a graph $G = (V, E)$ (with $|V| = n$ and $|E| = m$) and integer $\beta \geq 1$, a β-ruling set $S \subseteq V$ is a set of non-adjacent nodes such that for each node $u \in V$, there is a ruling set node $v \in S$ within β hops. This is a natural generalization of one of the most fundamental problems in parallel and distributed graph algorithms: Maximal Independent Set (MIS), which corresponds to a 1-ruling set. Ruling sets are closely related to clustering problems like Metric Facility Location, as fast algorithms for β-ruling sets imply fast algorithms for $O(\beta)$-approximate metric facility location [7, 18].

Our main contribution is an $O(1)$ round algorithm for 2-ruling sets in Congested Clique, improving on the $O(\log\log\log n)$ time algorithm by [16] and $O(\log\log\Delta)$ algorithm by [14], where Δ denotes the maximum degree of the input graph. While problems like minimum spanning tree [27] and $(\Delta+1)$-coloring [9] surprisingly admit constant round solutions in the Congested Clique model, the MIS problem and even β-ruling set for $\beta = O(1)$ have resisted attempts to obtain constant round algorithms. We make significant progress in this direction by giving the first $O(1)$ round algorithm for 2-ruling sets in Congested Clique.

The Congested Clique model [22], is a distributed synchronous message-passing model where each node is given its incident edges as an input and the nodes perform all-to-all communication in synchronous rounds. The crucial limitation is that the size of the messages sent between any pair of nodes, in a single round, is limited to $O(\log n)$ bits, where n is the number of nodes in the input graph. The goal is to minimize the number of required synchronous communication rounds.

As a consequence of our Congested Clique algorithm, we obtain a constant round algorithm for 2-ruling sets in the Linear Memory MPC model [19, 15, 5] and a constant pass algorithm in the semi-streaming model [10, 11, 26]. The Congested Clique and MPC implementations are asymptotically optimal in both time and space, as they use constant rounds and $O(n+m)$ total memory. In the semi-streaming model, one typically aims for very few passes, ideally just one. While we show that constant passes are enough to solve 2-ruling sets with $O(n)$ space, discovering the precise number of passes required is an interesting open question. The implementation details and definitions of the different models of computation can be found in Section 3. Our main results are captured in the following theorem.

▶ **Theorem** (Main Theorem). *There is a randomized parallel algorithm to the* 2*-ruling set problem that can be implemented in* $O(1)$ *rounds/passes (1) in the Congested Clique model, (2) in the MPC model with* $O(n)$ *words of local memory and* $O(n+m)$ *words of total memory, and (3) in the semi-streaming model with* $O(n)$ *words of space. The running time guarantee of the algorithm holds with high probability (w.h.p.)*[1]*.*

1.1 Previous Works on Ruling Sets and MIS

Recall that a 2-ruling set S is a set of non-adjacent nodes, such that each node in the input graph has a node from S within its 2-hop neighborhood. This is a strictly looser requirement than the one of an MIS and hence, an MIS algorithm directly implies a 2-ruling set algorithm. The classic algorithms by Luby [23] and Alon, Babai, and Itai [2] yield $O(\log n)$ time algorithms for ruling sets. In Congested Clique and MPC, this was later improved to $\widetilde{O}(\sqrt{\log \Delta})$ [13] and the current state-of-the-art for MIS is $O(\log\log \Delta)$ rounds [14].

When focusing on the 2-ruling set problem, roughly a decade ago, [7] gave an expected $O(\log\log n)$ time algorithm and [16] gave an $O(\log\log n)$ time algorithm w.h.p. in the Congested Clique model. Combining the result of [16] with the $O(\log\log \Delta)$ algorithm for MIS, this was improved to $O(\log\log\log n)$ rounds (w.h.p.). The 2-ruling set problem can be solved *deterministically* in $O(\log\log n)$ rounds in the Congested Clique model [28]. On the other hand, [4] shows that $\Omega(\log\log n)$ rounds are required to compute an MIS in the Broadcast Congested Clique model.

The $O(\log\log \Delta)$ algorithm by [14] carries over to the semi-streaming model. We note that there is an older variant of this algorithm that also yields an $O(\log\log \Delta)$-pass randomized *greedy* MIS, which is given by picking a random permutation over the nodes [1]. Then the permutation is iterated over and, whenever possible, the current node is added to the MIS. However, this approach requires a polylogarithmic overhead in space and hence, does

[1] Following the standard, we say that an event holds with high probability if it holds with probability $1 - 1/n^c$ for a constant $c \geq 1$ we can choose.

not work in the Congested Clique model. While it is known that computing an MIS in a *single-pass* of the stream requires $\Omega(n^2)$ memory [8], there has been progress towards designing single-pass semi-streaming algorithms for 2-ruling sets, but the current approaches require significantly larger than $O(n)$ memory: [20] shows that it can be done in $O(n\sqrt{n})$ memory and this was improved to $O(n^{4/3})$ by [3]. The $\Omega(n^2)$-space lower bound for MIS was generalized to $(\alpha, \alpha-1)$-ruling sets by [3]. An (α, β)-ruling set is a set S of nodes such that nodes within S are distance at least α apart and every node in $V \setminus S$ has a node in S within distance β. An MIS is a $(2,1)$-ruling set, and a 2-ruling set is a $(2,2)$-ruling set, in this paper we drop the α parameter for sake of convenience as it is always 2. A single-pass semi-streaming algorithm for 2-ruling set has not been ruled out.

1.2 A High-Level Technical Overview of Our Algorithm

Our strategy is to compute an MIS iteratively on subgraphs of size $O(n)$ until all nodes are covered. We begin with a sampling process where each node u is sampled independently with probability $1/\sqrt{\deg(u)}$. Call the set of sampled vertices V_{samp}. The intuition behind this sampling probability is to maximize the probability that each node has a sampled neighbor while ensuring $O(n)$ edges in the sampled subgraph $G[V_{\text{samp}}]$. To see why the sampled subgraph is small, assume we have a d-regular graph. Hence, each node is independently sampled with probability $1/\sqrt{d}$. For an edge $\{u, v\}$ to be sampled, both u and v must be sampled together. Therefore, an edge exists in the sampled graph with probability $1/d$, and we can say that there are at most $O(n)$ edges in the sampled graph in expectation (since the total number of edges is nd).

We show that the same expectation holds for general graphs by orienting the edges based on the degree of their end points and counting the number of outgoing sampled edges per node. In order to show that $G[V_{\text{samp}}]$ has $O(n)$ edges w.h.p., we face a technical challenge that the random choices for all edges are not independent. The dependencies disallow the use of standard Chernoff bounds, but we overcome the challenge by using the method of bounded differences to show concentration. For convenience, we state the concentration bounds in Section 4.

We classify nodes in the graph as either *good* or *bad*, based on how their initial degree compares with the sum of sampling probabilities of their neighbors:

▶ **Definition 1** (Good and Bad Nodes). *A node u is good if $\sum_{v \in N(u)} 1/\sqrt{\deg(v)} \geq \gamma \log \deg(u)$ (for a constant γ). If a node is not good, it is bad.*

Note that this definition is based entirely on the graph structure (i.e., degrees of all nodes). Furthermore, it is straightforward to detect which nodes are good and bad in $O(1)$ rounds.

By definition, good nodes are expected to have many sampled neighbors, so they have a relatively high probability of being covered by a node in V_{samp}. So we put the good nodes that do not have any sampled neighbors into a set V^* and process it later. We call V^* the nodes that are *set aside*. Therefore, the remaining graph of uncovered nodes only consists of bad nodes. In addition to all the good nodes without a sampled neighbor, we also add some bad nodes that are very likely to be covered (but weren't) into V^*. We use the fact that all nodes in V^* have a good chance of being covered to show that the graph $G[V^*]$ has $O(n)$ edges w.h.p. The main difficulty in proving this claim is that nodes are not put independently in V^*, and only nodes that are a certain distance apart are independent of each other. Moreover, the probability of a single node in V^* with degree d being covered is only $1 - 1/\text{poly}(d)$ which is not enough to do a simple union bound over all nodes. We overcome this challenge by showing that V^* can be partitioned into large sets of far apart nodes, which allows us to (1) boost the probability of each set being covered, and (2) apply a union bound over all sets.

Finally, we do an intricate counting argument that bounds the number of uncovered bad nodes. We do this by counting the number of edges that can exist between bad nodes and higher degree nodes, and show that the number of bad nodes with degree d is roughly bounded by $n/\sqrt{d}$. Note that this bound is not enough to show that the remaining graph is small. So we run the entire sampling and setting nodes aside process again to get the bound to be roughly n/d, which makes it straightforward to show that the remaining graph has $O(n)$ edges.

2 Parallel 2-Ruling-Set

In this section, we present a parallel algorithm for finding a 2-ruling set. For technical convenience, we assume that the maximum degree of the input graph is bounded by n^α for some constant $\alpha < 1/8$, and in Section 2.3 we show how to remove this assumption. When computing an MIS on the sampled vertices, we use a version of the Luby's algorithm that works as follows: in an iteration, each node picks a real number independently and uniformly at random in $[0, 1]$, local minima join the MIS, and these two steps are repeated until all nodes are either in the MIS or have a neighbor in the MIS. It is well known that this algorithm terminates in $O(\log n)$ iterations w.h.p. (see for example [12]).

Recalling the definition of good and bad nodes from Definition 1, let B_d be the set of bad nodes in G with (initial) degree in the range $[d, 2d)$. We describe the Parallel 2-Ruling-Set algorithm in Algorithm 1.

Algorithm 1 Parallel 2-Ruling-Set.

```
   Input: Graph $G = (V, E)$, with $\Delta \le n^\alpha$ ($\alpha < 1/8$). Each node $v \in V$ knows its degree
   $\deg(v)$ in $G$.
1: Each node $v \in G$ is independently sampled into $V_{\text{samp}}$ with probability $1/\sqrt{\deg(v)}$.
2: We put good nodes that do not have any neighbors in $V_{\text{samp}}$ in the set $V^*$.
3: Compute an MIS $I$ on $G_{\text{samp}} = G[V_{\text{samp}}]$ using Luby's algorithm first on the bad nodes
   and then on the rest of the nodes.
4: If any uncovered $u \in B_d$ has a neighbor $v$ that has at least $c\sqrt{d}\log^5(d)$ neighbors in $B_d$,
   we put $u$ in $V^*$.          ▷ $B_d$ is the set of bad nodes in $G$ with (initial) degree in $[d, 2d)$
5: Compute an MIS on $G[V^*]$.
6: Run Lines 1 to 5 on $G' = G[V \setminus (\mathcal{I} \cup N(\mathcal{I}) \cup N(N(\mathcal{I})))]$ where $\mathcal{I}$ is the set of MIS nodes.
7: Compute an MIS on the graph induced by the uncovered nodes.
   Output: The set of nodes that joined an MIS during the algorithm.
```

A node is considered covered if and only if it is at most 2-hops away from a node in the MIS, and two adjacent nodes can never join the MIS. Since the last step of the algorithm computes an MIS on the uncovered nodes, it is guaranteed to output a valid 2-ruling set, hence proving the following theorem.

▶ **Theorem 2.** *The Parallel 2-Ruling-Set algorithm (Algorithm 1) outputs a valid 2-ruling set.*

2.1 Structural Properties of the Subgraphs

We will now prove key structural properties of the different subgraphs on which we compute an MIS in Algorithm 1. This will allow for fast implementation of this algorithm in linear memory MPC, Congested Clique, and semi-streaming models (see Section 3). We first prove the fact that the sampled graph has a linear number of edges w.h.p.

▶ **Lemma 3.** *The sampled graph G_{samp} has $O(n)$ edges w.h.p.*

Proof. Let X be the random variable denoting the number of edges in G_{samp}. Let X_u be the indicator random variable for the event that u is sampled in V_{samp} and let Y_e be the indicator random variable for the event that edge e belongs to G_{samp}. We orient all edges from the end point with lower initial degree to higher initial degree. By the degree sum lemma, we have $X = \sum_{u \in V} \sum_{e \in \text{Out}(u)} Y_e$, where $\text{Out}(u)$ is defined as the set of outgoing edges of u.

Consider an oriented edge $e = (u, v)$ with $\deg(u) \leq \deg(v)$. We have that both u and v are sampled with probability at most $1/\sqrt{\deg(u)}$, so the probability that e is in G_{samp} is at most $1/\deg(u)$. Therefore, $\mathbb{E}[Y_e] \leq 1/\deg(u)$, and $\mathbb{E}[X] \leq n$.

We can interpret X as a function of the random variables $X_u, u \in V$, where changing one coordinate changes X by at most $\Delta = n^\alpha$. Therefore, X follows the bounded differences property with bounds $c_u = n^\alpha$ for all $u \in V$. Since the X_u's are independent of each other, we can use Lemma 18 with $\mu = t = n$ to say that: $\Pr[X > 2n] \leq 2\exp(n^2/n^{1+2\alpha}) \leq 2\exp(n^{1-2\alpha}) \leq 1/\text{poly}(n)$. ◀

As we observed earlier, good nodes expect to see a lot of sampled neighbors, therefore it is very unlikely that a good node has no sampled neighbors. The following lemma formalizes this intuition.

▶ **Lemma 4.** *In Line 2 of Algorithm 1, a good node u with $\deg(u) = d$ is added to V^* with probability $1/\text{poly}(d)$. This event is independent of the randomness (for sampling into V_{samp}) of nodes more than distance 1 from u in G.*

Proof. Since u is good, the sum of sampling probabilities of its neighbors is at least $\gamma \cdot \log d$. So we can bound the expected number of neighbors in V_{samp} as $\mathbb{E}\left[|N(u) \cap V_{\text{samp}}|\right] \geq \gamma \cdot \log d$. Since the sampling is done independently for each node, we can use Chernoff bound (Lemma 16) to compute the probability that no neighbor of u is in V_{samp}. We get that $\Pr\left[|N(u) \cap V_{\text{samp}}| = 0\right] < 1/d^{\gamma/2}$. Hence, u is added to V^* with probability at most $1/\text{poly}(d)$. This event only depends on the randomness of u and its neighbors in G, therefore it is independent of the randomness of nodes at distance more than 1 from u. ◀

On the other hand, bad nodes expect to have few sampled neighbors. So a sampled bad node will have a good probability of being a local minimum in the first iteration of Luby's algorithm. Therefore, nodes having many bad neighbors are very likely to have one such bad neighbor join the MIS, and hence all such bad neighbors are 2-hop covered.

▶ **Lemma 5.** *In Line 4 of Algorithm 1, each node $u \in B_d$ is added to V^* with probability at most $1/\text{poly}(d)$. This happens independently of the randomness (for sampling into V_{samp} and for the first Luby round when computing I) of nodes more than distance 3 from u in G.*

Proof. Recall that u is added to V^* if there is a node $v \in N(u)$ such that v has more than $c\sqrt{d} \cdot \log^5 d$ neighbors in B_d. Let v be an arbitrary such node and let A_u be a subset of $N(v) \cap B_d$ such that $|A_u| = c\sqrt{d}\log^5 d$ and $u \in A_u$. We will show that at least one node of A_u joins the MIS in Line 3 of Algorithm 1 in the first Luby round with probability at least $1 - 1/\text{poly}(d)$.

A node $w \in A_u$ joins I in the first Luby round iff w is sampled and if w has the smallest random number (in the first Luby round) among all its sampled neighbors. Note that if one node of A_u joins the MIS only depends on the randomness of the bad nodes in $A_u \cup N(A_u)$, which are a subset of the 2-hop neighborhood of v and thus of the 3-hop neighborhood of u.

First, note that the number of nodes in $A_u \cup N(A_u)$ is at most $O(d^{3/2}\log^5(d))$ because $A_u \subseteq B_d$, so every node has at most $2d$ neighbors in $N(A_u)$.

Let S_u be the set of sampled nodes in A_u. Every node in $N(A_u)$ with at most $O(\sqrt{d}\log^2 d)$ neighbors in A_u has at most $O(\log^2 d)$ neighbors in S_u with probability at least $1 - 1/poly(d)$ (by using Chernoff bound Lemma 16 and then union bound over all such nodes in $N(A_u)$). On the other hand, every node $w \in N(A_u)$ with $\Omega(\sqrt{d}\log^2 d)$ neighbors in A_u has $\Omega(\log^2 d)$ neighbors in S_u in expectation. If w is a good node, it does not participate in the first Luby round carried out by the nodes in A_u, and if w is a bad node, the expected number of sampled neighbors of w is at most $\gamma\log(\deg(w))$ and we therefore have $\log(\deg(w)) = \Omega(\log^2 d)$ and thus $\deg(w) = \exp(\Omega(\log^2 d))$. The probability that w is sampled is therefore $\ll 1/poly(d)$.

With probability $1 - 1/\text{poly}(d)$, all the sampled bad nodes in $N(A_u)$ therefore have at most $O(\log^2 d)$ sampled neighbors in A_u. Moreover, all the sampled nodes in A_u have at most $O(\log d)$ overall sampled neighbors, since A_u is a subset of B_d, it has at most $\gamma\log 2d$ sampled neighbors in expectation. Using a Chernoff bound (Lemma 16) gives us that each node in A_u has $O(\log d)$ sampled neighbors with probability $1 - 1/poly(d)$. In the following, we condition on this event happening.

Consider the graph G_S induced by the sampled nodes in $A_u \cup N(A_u)$. For any two nodes $x, y \in S_u$ that are at distance at least 3 in G_S, the events that x and y join the MIS I in the first Luby step are independent. For every node $x \in S_u$, there are at most $O(\log^3 d)$ other nodes in S_u at distance at most 2 in G_S (at most $O(\log d)$ direct neighbors and because the direct neighbors can be in $N(A_u)$ at most $O(\log^3 d)$ 2-hop neighbors). By greedily picking nodes in S_u, we can therefore find a set of size $\Omega(|S_u|/\log^3 d)$ of nodes in S_u that independently join the MIS I in the first Luby step. Because with probability $1 - 1/\text{poly}(d)$, $|S_u| = \Omega(\log^5 d)$, we have $\Omega(|S_u|/\log^3 d) = \Omega(\log^2 d)$.

Each of those nodes independently joins I with probability at least $1/O(\log d)$ and therefore, one of those nodes joins I with probability at least $1 - 1/\text{poly}(d)$. ◀

We use the fact that nodes are added mostly independently and with low probability to V^* in order to show that the graph induced by these nodes cannot have many edges.

▶ **Lemma 6.** *The induced subgraph $G[V^*]$ has $O(n)$ edges w.h.p.*

Proof. A node v is placed in V^* if either (1) v is a good node with no neighbors in V_{samp}, or (2) v is an uncovered bad node in B_d and has a neighbor v that has at least $c \cdot \sqrt{d} \cdot \log^5(d)$ neighbors in B_d. By Lemmas 4 and 5, each such node v is put in V^* with probability at most $1/\text{poly}(\deg(v))$, and this happens independently of the randomness (for sampling into V_{samp} and for the first Luby round when computing I) of nodes at distance more than 3 from v. The exponent of the polynomial depends on c and γ.

Nodes with constant degree can be ignored, as they will contribute at most $O(n)$ edges. Therefore, we can assume that each node is put in V^* with probability at most $1/2$.

For the sake of analysis, we compute a greedy coloring of $G^7[V^*]$. To get $G^7[V^*]$, we first build the graph G^7 which is the graph where we add an edge between any pair of nodes that are at distance at most 7 in G, and then we take the induced subgraph of G^7 on V^*.

For each color class that has at least $n^{1-8\alpha}$ nodes, all of which are at distance more than 7 from each other in G, they join V^* independently of each other. Therefore, the probability that all nodes in a single color class belongs to V^* is at most $(1/2)^{n^{1-8\alpha}} \ll 1/\mathrm{poly}(n)$. By union bounding over the color classes, we get that with probability $1 - 1/\mathrm{poly}(n)$, the size of each color class is less than $n^{1-8\alpha}$.

Each node in $G^7[V^*]$ has degree at most $n^{7\alpha}$ and hence there are $n^{7\alpha} + 1$ color classes. Recall that $\Delta \leq n^{\alpha}$ in G, so if a color class $C \subseteq V^*$ has less than $n^{1-8\alpha}$ nodes, the number of edges in G that are incident on C is at most $n^{1-7\alpha}$. Therefore, all color classes with less than $n^{1-8\alpha}$ nodes can only add at most $O(n)$ edges to $G[V^*]$.

The lemma follows since we already showed that w.h.p., the size of each color class is less than $n^{1-8\alpha}$. ◀

Recall that B_d is the set of bad nodes in G with (initial) degree in the range $[d, 2d)$. Let B_d^* be the nodes in B_d that have a neighbor v with more than $c\sqrt{d}\log^5 d$ neighbors in B_d. If a node in B_d^* is not covered by the MIS I on V_{samp}, then it is put into V^*. Let $\overline{B}_d = B_d \setminus B_d^*$. Define $B = \cup_{i=0}^{\log n} B_{2^i}$ and $\overline{B} = \cup_{i=0}^{\log n} \overline{B}_{2^i}$.

▶ **Lemma 7.** *The graph $G' = G[V \setminus (\mathcal{I} \cup N(\mathcal{I}) \cup N(N(\mathcal{I})))]$ of nodes that are uncovered before Line 6 contains only nodes in $\overline{B}$.*

Proof. Nodes not in $\overline{B}$ are the good nodes and the bad nodes in B_d^* for all d. We show that all these nodes are covered before Line 6 and hence cannot belong to G'. Nodes that are either in V_{samp}, or have a neighbor in V_{samp} are covered by the MIS I computed in Line 3. Good nodes that are not covered by I are put in V^* in Line 2. Similarly, nodes in B_d^* that are not covered by I are put in V^* in Line 4. All nodes in V^* are covered because we compute an MIS on $G[V^*]$ in Line 5. Therefore, all nodes not in $\overline{B}$ are covered before Line 6. ◀

2.2 Counting the Bad Nodes

Let $V_{\geq d}$ be the set of all nodes in G of (initial) degree at least d. Intuitively, we now want to say: (1) for each bad node in $\overline{B}_d$, there are at least $d/2$ edges to higher degree nodes and (2) from the higher degree nodes, only roughly $\sqrt{d}$ edges to $\overline{B}_d$. Hence, we can conclude that $d \cdot |\overline{B}_d| \leq |V_{\geq d^2}| \cdot \sqrt{d}$ which further implies that $|\overline{B}_d| \leq |V_{\geq d^2}|/\sqrt{d}$.

▶ **Lemma 8.** *Consider a bad node u with $\deg(u) = d$. Then for at least $d/2$ nodes $v \in N(u)$ it holds that $\deg(v) \geq d^2/(2\gamma^2 \log^2 d)$.*

Proof. Otherwise, more than half of the neighbors have degree less than $d^2/(2\gamma^2 \log^2 d)$. Hence,

$$\sum_{v \in N(u)} \frac{1}{\sqrt{\deg(v)}} \geq \frac{d}{2} \cdot \frac{\sqrt{2\gamma^2 \log^2 d}}{\sqrt{d^2}} = \frac{d}{2} \cdot \frac{2\gamma \log d}{d} = \gamma \log d \ ,$$

which is a contradiction with u being bad. ◀

▶ **Lemma 9.** *For any d, we have that $|\overline{B}_d| \leq 2|V_{\geq d^2/(2\gamma^2 \log^2 d)}| \cdot \log^5 d/\sqrt{d} \leq 2n \log^5 d/\sqrt{d}$.*

Proof. Let $d' = d^2/(2\gamma^2 \log^2 d)$. From Lemma 8, we know that for any $u \in \overline{B}_d$, at least $d/2$ edges go to $V_{\geq d'}$. Furthermore, we have that any $v \in V_{\geq d'}$ has at most $c\sqrt{d}\log^5 d$ edges to $\overline{B}_d$, since otherwise, none of v's neighbors are in $\overline{B}_d$. Hence, we can conclude that $\frac{d}{2} \cdot |\overline{B}_d| \leq |V_{\geq d'}| \cdot \sqrt{d}\log^5 d$ which proves the lemma. ◀

Now we have that just before Line 6, $|\overline{B}_d| \approx n/\sqrt{d}$, and we now run the entire algorithm again on the uncovered graph G'. For G', we define the sets $V'_{\geq d}$, B'_d, $\overline{B'}_d$, B', and $\overline{B'}$ similarly as we did for G. Now we would expect that $|\overline{B'}_d| \approx n/d$, which is good because all nodes have degree at most $2d$.

▶ **Lemma 10.** *The graph induced by the uncovered nodes before Line 7 has $O(n)$ edges.*

Proof. Again let $d' = d^2/(2\gamma^2 \log^2 d)$. By a similar argument as Lemma 7, the uncovered nodes are a subset of $\overline{B'}$. We can assume that d is at least some large enough constant, as the nodes in all B'_d for constant d have at most $O(n)$ edges. By Lemma 9, we have that $|\overline{B}_d| \leq 2|V_{\geq d'}|\log^5 d/\sqrt{d}$ for any d, and we can similarly argue $|\overline{B'}_d| \leq 2|V'_{\geq d'}|\log^5 d/\sqrt{d}$ for any d.

Now $V'_{\geq d'} \subseteq \overline{B}_{\geq d'}$ where $\overline{B}_{\geq d'} = \cup_{i=\log d'}^{\log n} \overline{B}_{2^i}$. Therefore,

$$|V'_{\geq d'}| \leq \sum_{i=\log d'}^{\log n} \frac{2n\log^5 2^i}{\sqrt{2^i}} = \sum_{i=\log d'}^{\log n} \frac{2ni^5}{2^{i/2}} \leq \sum_{i=\log d'}^{\log n} \frac{2n}{2^{i/3}} \leq O\left(\frac{n\log^{2/3} d}{d^{2/3}}\right)$$

Where the last inequality follows because the sum on the left is a geometric sequence with rate $2^{-1/3}$ and it is well known that if the rate is between 0 and 1, the sum is asymptotically dominated by the first term. Therefore, $|\overline{B'}_d| \leq O(n \cdot \text{poly}(\log d)/d^{7/6}) \leq O(n/d)$. Since each node in $\overline{B'}_d$ has degree at most $2d$, and each node belongs to exactly one set, the graph induced by nodes in $\overline{B'}$ has $O(n)$ edges. ◀

2.3 Degree Reduction

We define the degree reduction process similar to [14] and for sake of completeness we provide a self-contained explanation here. Our goal is to lower the maximum degree of the input graph G such that, after a constant number i of steps, the maximum degree is strictly less than n^α for some fixed constant $\alpha < 1/8$. Let $G_1 = G$, and $\Delta_j = \Delta^{(3/4)^j}$. Let i be a value such that for all $1 \leq j < i$, $\Delta_j > n^{\alpha/2}$ and $\Delta_i \leq n^{\alpha/2}$. Since $\Delta \leq n$, we can say that the largest value i can take is $\lceil \log_{4/3}(2/\alpha) \rceil = O(1)$.

In each step $j = 1 \ldots i$, we sample nodes S_j in G_j with probability $1/\Delta_j$, and then compute an MIS on the subgraph induced by the sampled nodes $G_j[S_j]$. The residual graph G_{j+1} is obtained by removing all the neighbors of the sampled nodes. For each of these steps to be possible, the graph induced by the sampled nodes must have $O(n)$ edges w.h.p. In order to guarantee the feasibility of step j, the maximum degree in the residual graph after step $j-1$ (i.e. G_j) has to be sufficiently small, which we show in the following lemma.

▶ **Lemma 11.** *If we process a graph G_j induced by nodes picked uniformly at random with probability $1/\Delta_j$, the maximum degree in the residual graph G_{j+1} is $O(\Delta_j \log n)$ w.h.p.*

Proof. Consider a node v in the residual graph such that $\deg(v) > d$. The probability that a neighbor of v is sampled is at least $1/\Delta_j$. Therefore, the probability that no neighbor of v is sampled is at most $\left(1 - \frac{1}{\Delta_j}\right)^d \leq \exp\left(-d/\Delta_j\right)$.

Denote $c > 1$ an arbitrary constant, and suppose that $d = c\Delta_j \log n$. Then, the probability that $\deg(v) > d$ is at most $\exp\left(-c\log n\right) = n^{-c}$. Hence, $\deg(v) = O(\Delta_j \log n)$ w.h.p. We conclude the lemma by union bounding over all nodes of the residual graph. ◀

Therefore, the residual graph G_{i+1} has maximum degree $O(\Delta_i \log n) \leq O(n^{\alpha/2} \log n)$ $\leq n^\alpha$ w.h.p., which is our assumption in Algorithm 1. We now finish by showing that each induced subgraph has linear size.

▶ **Lemma 12.** *For all $1 \leq j \leq i$, the graph induced by sampled nodes in step j, $G_j[S_j]$, has $O(n)$ edges w.h.p.*

Proof. First, consider a node $v \in S_j$, and $u \in N(v)$. The probability that $u \in S_j$ is $1/\Delta_j$. By Lemma 11, we condition on the high probability event that $O(\Delta_{j-1} \log n)$ is the maximum degree of G_j. Note that this conditioning only affects the randomness used for sampling in iterations $1, \ldots, j-1$. In particular, this implies that the conditioning does not affect the randomness used for sampling in iteration j. Therefore, the expected degree of v in S_j is at most $\mu = O(\log n \cdot \Delta_{j-1}/\Delta_j)$. Using Lemma 16, $\Pr[\deg(v) \geq (1+c)\mu] \leq \exp(-c^2\mu/(2+c)) \leq n^{-c}$, since $\mu \geq \log n$. By union bounding over all nodes of S_j, the maximum degree in $G_j[S_j]$ is $O(\log n \cdot \Delta_{j-1}/\Delta_j)$ w.h.p.

Second, the expected number of nodes in S_j is at most $\mu' = n/\Delta_j$. Using Lemma 16, $\Pr[|S_j| \geq (1 + c\log n)\mu'] \leq \exp(-c^2 \log^2 n\mu'/(2 + c\log n)) \leq n^{-c}$, since $\mu' \geq 1$. Therefore, $|S_j| = O(n \log n/\Delta_j)$ w.h.p.

Now we can upper bound the number of edges in $G_j[S_j]$ by $|S_j| \cdot \Delta(G_j[S_j])$. Therefore, we can say w.h.p. that

$$|S_j| \cdot \Delta(G_j[S_j]) = O\left(\frac{n\log n}{\Delta_j} \cdot \frac{\Delta_{j-1}\log n}{\Delta_j}\right) = O\left(n\log^2 n \cdot \frac{\Delta_{j-1}}{\Delta_j^2}\right)$$

Moreover, since $\Delta_j = \Delta^{(3/4)^j}$, we have that $\Delta_{j-1}/\Delta_j^2 = \Delta^{(3/4)^{j-1}-2(3/4)^j} = \Delta^{-(1/2)(3/4)^{j-1}} = 1/\sqrt{\Delta_{j-1}}$. This term cancels the $\log^2 n$ term since $\Delta_{j-1} > n^{\alpha/2} \gg \log^4 n$. Hence, the number of edges in $G_j[S_j]$ is $O(n)$ w.h.p. ◀

3 Implementation of Parallel 2-Ruling-Set

In this section, we show how to implement the algorithm Parallel 2-Ruling-Set in several models of parallel computation. In each case of the implementations, we only need to bound the runtime and memory usage of the algorithm in the corresponding model. Since we faithfully execute the Parallel 2-Ruling-Set algorithm, the solutions computed are guaranteed to be correct.

3.1 Congested Clique

In the Congested Clique model [22], we have n machines, where each machine is identified with a single node in the input graph. The communication network is a clique, that is, the machines are connected in all-to-all fashion, and the input graph is considered to be a subgraph of the network. Machines can send unique messages to all other machines via the edges in the clique, and the bandwidth of each edge is limited to $O(\log n)$ bits. Congested Clique and MPC are closely related to each other, for example [17, 6] show how to implement any Congested Clique algorithm in the MPC model.

By the definition of Parallel 2-Ruling-Set, we compute an MIS sequentially on several subgraphs: (1) the sampled graphs $G_j[S_j]$ $(1 \leq j \leq i = O(1))$ for reducing the degree, (2) the graphs induced by V_{samp} and V^* on G during the first run and on G' during the second run, and finally (3) the graph induced by uncovered nodes in the last step. Creating a subgraph takes a constant number of rounds, since nodes just need to know the random choices and aggregate information of their 1-hop neighbors.

By the lemmas in Section 2.1 and Section 2.3, all these subgraphs have $O(n)$ edges w.h.p. Therefore, we can use Lenzen's routing protocol [21] to gather all the subgraphs one after the other at a single machine in $O(1)$ rounds. This machine computes the MIS according to Algorithm 1, and informs the rest of the nodes whether they joined the MIS or not, which allows us to identify the next subgraph. Therefore, we get the following result.

▶ **Theorem 13.** *There is a Congested Clique algorithm to find a* 2*-ruling set. The algorithm runs in* $O(1)$ *rounds w.h.p.*

3.2 Linear Memory MPC Model

In the MPC model [19, 15, 5], we have M machines with S words of memory each, where each word corresponds to $O(\log n)$ bits. Notice that an identifier of an edge or a node requires one word to store. The machines communicate in an all-to-all fashion. The input graph is divided among the machines and for simplicity and without loss of generality, we assume that the edges of each node are placed on the same machine. In the *linear-space* MPC model, we set $S = \Theta(n)$. Furthermore, the *total space* is defined as $M \cdot S$ and in our case, we have $M \cdot S = \Theta(m)$. Notice that $M \cdot S = \Omega(m)$, for the number of edges m in the input graph, simply to store the input.

The implementation follows as a direct consequence of the Congested Clique model. Since the local memory is $\Theta(n)$, we can send each of the $O(n)$ sized subgraphs one by one to a single machine in $O(1)$ rounds and use this machine to compute the MIS as described in the algorithm. We obtain the following theorem.

▶ **Theorem 14.** *There is an MPC algorithm with* $O(n)$ *words of local and* $O(m)$ *total memory to find a* 2*-ruling set. The algorithm runs in* $O(1)$ *rounds w.h.p.*

3.3 Semi-streaming

Typically, in the distributed and parallel settings, the input graph is too large to fit a single computer. Hence, it is divided among several computers (in one way or another) and the computers need to communicate with each other to solve a problem. Another angle at tackling large datasets and graphs is through the *graph streaming models* [10, 11, 26]. In these models, the graph is not stored centrally, but an algorithm has access to the edges one by one in an input stream, chosen randomly or by an adversary (the choice sometimes makes a difference). We assume that each edge is processed before the next pass starts. In the semi-streaming setting, the algorithm has $\widetilde{O}(n)$ working space, that it can use to store its state. The goal is to make as few *passes* over the edge-stream as possible, ideally just a small constant amount. Notice that in the case of many problems, such as matching approximation or correlation clustering, simply storing the output might demand $\Omega(n)$ words.

In the semi-streaming model, we use one pass to process one subgraph of size $O(n)$, by storing it in memory and computing an MIS as described in the algorithm. Since there are $O(1)$ such subgraphs, we require $O(1)$ passes. This leads to the following theorem.

▶ **Theorem 15.** *There is an* $O(1)$*-pass semi-streaming algorithm with* $O(n)$ *words of space to find a* 2*-ruling set w.h.p.*

4 Concentration Inequalities

▶ **Lemma 16** (Chernoff Bounds). *Let* $X_1, \ldots, X_k$ *be independent* $\{0,1\}$ *random variables. Let* X *denote the sum of the random variables,* μ *the sum's expected value. Then,*

1. *For* $0 \le \delta \le 1$, $\Pr[X \le (1-\delta)\mu] \le \exp(-\delta^2\mu/2)$ *and* $\Pr[X \ge (1+\delta)\mu] \le \exp(-\delta^2\mu/3)$,
2. *For* $\delta \ge 1$, $\Pr[X \ge (1+\delta)\mu] \le \exp(-\delta^2\mu/(2+\delta))$.

▶ **Definition 17** (Bounded Differences Property). *A function* $f : \mathcal{X} \to \mathbb{R}$ *for* $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \cdots \times \mathcal{X}_n$ *is said to satisfy the bounded differences property with bounds* $c_1, c_2, \ldots, c_n \in \mathbb{R}^+$ *if for all* $\overline{x} = (x_1, x_2, \ldots, x_n) \in \mathcal{X}$ *and all integers* $k \in [1, n]$ *we have*

$$\sup_{x'_k \in X_k} |f(\overline{x}) - f(x_1, x_2, \ldots, x_{i-1}, x'_k, \ldots, x_n)| \le c_k$$

▶ **Lemma 18** (Bounded Differences Inequality [25, 24]). *Let $f : \mathcal{X} \to \mathbb{R}$ satisfy the bounded differences property with bounds $c_1, c_2, \ldots, c_n$. Consider independent random variables $X_1, X_2, \ldots, X_n$ where $X_k \in \mathcal{X}_k$ for all integers $k \in [1, n]$. Let $\overline{X} = (X_1, X_2, \ldots, X_n)$ and $\mu = \mathbb{E}[f(\overline{X})]$. Then for any $t > 0$ we have:*

$$\Pr[|f(\overline{X}) - \mu| \geq t] \leq 2 \exp\left(\frac{-t^2}{\sum_{k=1}^{n} c_k^2}\right)$$

References

1 Kook Jin Ahn, Graham Cormode, Sudipto Guha, Andrew McGregor, and Anthony Wirth. Correlation Clustering in Data Streams. In *International Conference on Machine Learning (ICML)*, pages 2237–2246, 2015.

2 Noga Alon, Lásló Babai, and Alon Itai. A Fast and Simple Randomized Parallel Algorithm for the Maximal Independent Set Problem. *Journal of Algorithms*, 7(4):567–583, 1986.

3 Sepehr Assadi and Aditi Dudeja. Ruling Sets in Random Order and Adversarial Streams. In Seth Gilbert, editor, *35th International Symposium on Distributed Computing (DISC 2021)*, volume 209 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:18, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.DISC.2021.6`.

4 Sepehr Assadi, Gillat Kol, and Zhijun Zhang. Rounds vs communication tradeoffs for maximal independent sets. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1193–1204. IEEE, 2022.

5 Paul Beame, Paraschos Koutris, and Dan Suciu. Communication Steps for Parallel Query Processing. *J. ACM*, 64(6), 2017. `doi:10.1145/3125644`.

6 Soheil Behnezhad, Mahsa Derakhshan, and MohammadTaghi Hajiaghayi. Semi-mapreduce meets congested clique, 2018. `doi:10.48550/ARXIV.1802.10297`.

7 Andrew Berns, James Hegeman, and Sriram V. Pemmaraju. Super-Fast Distributed Algorithms for Metric Facility Location. In *Proceedings of the International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 428–439, 2012.

8 Graham Cormode, Jacques Dark, and Christian Konrad. Independent Sets in Vertex-Arrival Streams. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 45:1–45:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. `doi:10.4230/LIPIcs.ICALP.2019.45`.

9 Artur Czumaj, Peter Davies, and Merav Parter. Simple, deterministic, constant-round coloring in the congested clique. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, PODC '20, pages 309–318, New York, NY, USA, 2020. Association for Computing Machinery. `doi:10.1145/3382734.3405751`.

10 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. Graph Distances in the Streaming Model: The Value of Space. In *the Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 745–754, 2005.

11 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On Graph Problems in a Semi-Streaming Model. *Theor. Comput. Sci.*, 348(2):207–216, 2005. `doi:10.1016/j.tcs.2005.09.013`.

12 Mohsen Ghaffari. An improved distributed algorithm for maximal independent set. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 270–277. SIAM, 2016.

13 Mohsen Ghaffari. Distributed MIS via All-to-All Communication. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 141–149, 2017. `doi:10.1145/3087801.3087830`.

14 Mohsen Ghaffari, Themis Gouleakis, Christian Konrad, Slobodan Mitrović, and Ronitt Rubinfeld. Improved Massively Parallel Computation Algorithms for MIS, Matching, and Vertex Cover. In *the Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 129–138, 2018.

15 Michael T. Goodrich, Nodari Sitchinava, and Qin Zhang. Sorting, Searching, and Simulation in the Mapreduce Framework. In *Proceedings of the International Symposium on Algorithms and Computation (ISAAC)*, pages 374–383, 2011. doi:10.1007/978-3-642-25591-5_39.

16 James Hegeman, Sriram Pemmaraju, and Vivek Sardeshmukh. Near-Constant-Time Distributed Algorithms on a Congested Clique. In *Proceedings of the International Symposium on Distributed Computing (DISC)*, pages 514–530, 2014. doi:10.1007/978-3-662-45174-8_35.

17 James W. Hegeman and Sriram V. Pemmaraju. Lessons from the congested clique applied to mapreduce. *Theoretical Computer Science*, 608:268–281, 2015. Structural Information and Communication Complexity. doi:10.1016/j.tcs.2015.09.029.

18 Tanmay Inamdar, Shreyas Pai, and Sriram V. Pemmaraju. Large-Scale Distributed Algorithms for Facility Location with Outliers. In Jiannong Cao, Faith Ellen, Luis Rodrigues, and Bernardo Ferreira, editors, *22nd International Conference on Principles of Distributed Systems (OPODIS 2018)*, volume 125 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:16, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.OPODIS.2018.5.

19 Howard Karloff, Siddharth Suri, and Sergei Vassilvitskii. A Model of Computation for MapReduce. In *the Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 938–948, 2010.

20 Christian Konrad, Sriram V Pemmaraju, Talal Riaz, and Peter Robinson. The complexity of symmetry breaking in massive graphs. In *33rd International Symposium on Distributed Computing (DISC 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

21 Christoph Lenzen. Optimal deterministic routing and sorting on the congested clique. In *Proceedings of the 2013 ACM Symposium on Principles of Distributed Computing*, pages 42–50. Association for Computing Machinery, 2013. doi:10.1145/2484239.2501983.

22 Zvi Lotker, Elan Pavlov, Boaz Patt-Shamir, and David Peleg. Mst construction in o(log log n) communication rounds. In *Proceedings of the Fifteenth Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 94–100. Association for Computing Machinery, 2003. doi:10.1145/777412.777428.

23 M. Luby. A Simple Parallel Algorithm for the Maximal Independent Set Problem. *SIAM Journal on Computing*, 15:1036–1053, 1986.

24 Colin McDiarmid. *Concentration*, pages 195–248. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998. doi:10.1007/978-3-662-12788-9_6.

25 Colin McDiarmid et al. On the method of bounded differences. *Surveys in combinatorics*, 141(1):148–188, 1989.

26 S. Muthukrishnan. Data Streams: Algorithms and Applications. *Theoretical Computer Science*, 1(2):117–236, 2005. doi:10.1561/0400000002.

27 Krzysztof Nowicki. A deterministic algorithm for the mst problem in constant rounds of congested clique. In *Proceedings of the ACM Symposium on Theory of Computing (STOC)*, STOC 2021, pages 1154–1165, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3406325.3451136.

28 Shreyas Pai and Sriram V. Pemmaraju. Brief Announcement: Deterministic Massively Parallel Algorithms for Ruling Sets. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 366–368, 2022. doi:10.1145/3519270.3538472.