Fast Ruling Sets

Thesis submitted in partial fulfillment
of the requirements for the degree of

MS By Research
in
Computer Science

by

Tushar Bisht
200802037
tushar.bishtug08@students.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
May 2016

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Fast Ruling Sets" by Tushar Bisht, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date	Adviser: Dr. Kishore Kothapalli

I would like to dedicate this thesis to my beloved parents and sister

Acknowledgments

I would like to thank my guide, Dr. Kishore Kothapalli, for the patient guidance, encouragement, and advice he has provided throughout my time as his student. I have been extremely lucky to have a supervisor who cared so much about my work, and who responded to my questions and queries so promptly. I am infinitely grateful to him for sharing with me his knowledge and wisdom, for his valuable advice, comments and suggestions, for many interesting discussions related to research and beyond, for his ability of being demanding and patient at the same time, for teaching me to follow high standards in research and never give up.

I am a very fortunate person to have truly good friends and I am enormously grateful to all of them for their support and love, for all moments of fun we had together. Many thanks to all of you Yash, Ajay, Saket, Sarvesh, Harshit, Akshay, Sandeep, Aman, Hardeep, Hemant, Arpit, Nachiket and more.

I would also like to express my gratitude to all the people in the C-STAR lab Nadeem, Jatin sir, Shashank, Aman, Teja, Dharmeet sir for being a great company in the lab.

Finally, I express my endless gratitude and love to my family for their infinite love and belief in me, for wise thoughts and advise, for huge support and encouragement in everything I do - for all this and much more I am forever grateful to them.

Abstract

Computing a maximal independent set (MIS) is a fundamental problem in distributed computing for its ability to nicely capture the essential challenge of symmetry breaking and also for its myriad applications to other problems. A relaxed version of computing an MIS is the problem of computing a t-ruling set. A t-ruling set for an integer $t \geq 1$ in a graph G is an independent subset $S \subseteq V$ such that every nodes $v \in V$ is either in S or has a neighbor $w \in S$ at a distance at most t from v. Computing a t-ruling set also involves symmetry breaking.

All the algorithms we present in this thesis are randomized and synchronous in nature. First, we present an algorithm which computes a 2-ruling set in any graph in $\frac{\log \Delta}{\log \log n} + e^{O(\sqrt{\log \log n})}$rounds with high probability, where n denotes the number of vertices in the graph and Δ denotes the maximum degree of any vertex in the graph. This result is an improvement over the $O\left(\frac{\log \Delta}{(\log n)^{\epsilon}} + (\log n)^{1/2+\epsilon}\right)$ round algorithm from [6] whenever $\Delta \in O(2^{\sqrt{\log n}})$. We then use it to present another algorithm which can compute a 2-ruling set for graphs of arboricity $a = 2^{O(\sqrt{\log n})}$ in $O(\sqrt{(\log n / \log \log n)}$ rounds with high probability.

We then extend the ideas from the above algorithm and show that a $(t+1)$-ruling set for any $t \geq 1$ in graphs with arboricity $a = O(\log \log n)$ can be computed in $O(t \log^{1/t} \Delta + t^2 (\log \log n)^{1/t} + (\log \log n)^2)$ rounds. Therefore, a $\log \log \Delta$-ruling set for bounded arboricity graphs can be computed in $O((\log \log n)^2)$ rounds. Our results therefore indicate that while computing an MIS in graphs of bounded arboricity in $O(\sqrt{\log n})$ rounds is still an open problem, ruling sets for small values of t can be found in much fewer rounds. Similarly, a $(t+1)$-ruling set for a general graph can be computed in $O(t \log^{1/t} \Delta + t^2 (\log \log n)^{1/t} + e^{O(\sqrt{\log \log n})})$ rounds. Also, a t-ruling set for graphs of girth > 6 can be computed in $O(t \log^{1/t} \Delta + t^2 (\log \log n)^{1/t} + e^{O(\sqrt{\log \log n})})$ rounds.

In a later section, we compare the run time of Luby's [13] and Métivier's [14] MIS algorithms, two fundamental algorithms for computing an MIS in general graphs. Also, we try to understand the effect of probability of a node getting selected by Luby's algorithm on the number of rounds required by algorithm to finish. We run these algorithms on different types of graphs, namely Random, Regular and Bipartite Graphs.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Graph theory deals with the representation of objects and their pairwise relationships. It is the study of graphs where vertices in the graph represents objects and the edges between vertices represents their relationship. The general notion of representing a graph is $G = (V, E)$, where V represents the vertex set and E the edge set. Distributed systems are modelled via graphs. The computing nodes comprises of the vertex set and the an edge between two node represents a direct communication channel between them. In a sequential system, the effort to solve a problem is measured in terms of the number of steps the system requires. The same is the concept with distributed computing. The general idea of distributed computing is that the nodes use the local information they have to compute something useful, then they share their findings with their neighbors, so every node gets some new useful information to process and then they repeat this cycle again until the desired output. Local computation is very fast compared to the communication, but the power communication provides cannot be substituted by fast computation. The overall effort required by a distributed algorithm is the number of rounds it requires to finish. In one round it computes something locally, sends it to all its neighbors thereby receiving data from its neighbors too.

Communication serves to exchange intermediate results but it also is required to establish coordination among nodes in the graph. Coordination can be achieved in many ways. The first one is a commanding node which controls the whole graph, the disadvantage with this is a single point of failure. Another way could be that a node gathers the whole information of the graph to process and distributes the result back. This again has the disadvantage of a single point of failure, also it basically removes the distributed computing advantage when nodes may not be individually capable to process the whole data. Also, gathering requires exponential flow of data which is not recommended. One alternative to above approaches is somehow having many commanding nodes governing their small neighborhood. Therefore, these nodes can gather neighboring data and process it locally. A fundamental constraint in distributed systems is the cost of sending messages to nodes that are very far. The time required to send a message between two nodes will be directly proportional to the distance between them. The maximum distance between two nodes is the diameter but there are problems which have far better run time than

the diameter of the graph suggesting the possibility to rely on the local information to get the output faster for various problems. This raises the question as to which things can be computed fast locally?

For most of distributed algorithms symmetry breaking is the key, which avoids conflicts and redundant actions among vertices. Symmetry breaking is of fundamental interest in distributed computing. A classic example involving symmetry breaking is the computation of a maximal independent set (MIS) in a graph $G = (V, E)$. A very close related problem is of graph coloring which also employs symmetry breaking at the heart of it. Both of them are highly studied problems of distributed computing with applications to basic problems like resource allocation and scheduling.

1.1 Definitions

Definition 1.1.1 *Independent Set : Given a graph $G = (V, E)$, a set $I \subseteq V$ is an independent set if no two vertices in I are neighbors, i.e. connected by an edge. Refer to Figure 1.2 where selected nodes are indicated by dark colored nodes.*

Definition 1.1.2 *Maximal Independent Set(MIS) : Given a graph $G(V, E)$, an MIS of G is an independent set that is maximal with respect to inclusion. Refer to Figure 1.1, the dark colored nodes represents the MIS.*

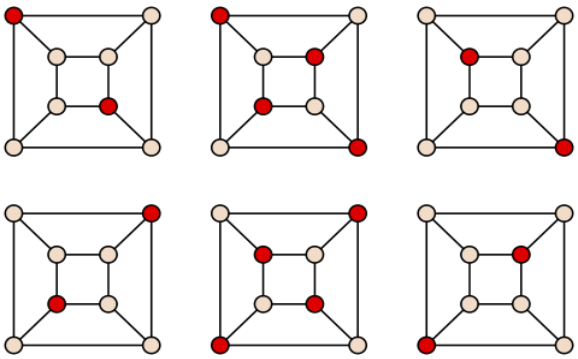

Figure 1.1 MIS

Definition 1.1.3 *Dominating Set : It is a subset $D \in V$, such that every vertex $v \in V$ of the graph $G = (V, E)$ either belongs to D or has a neighbor in D. Refer to Figure 1.2*

The difference between a dominating set and an MIS is that nodes may not be independent in a dominating set. An independent set I can only be a dominating set if it is a maximal independent set thereby making I to be a minimal dominating set. The minimal dominating set of the smallest size is called

minimum dominating set and the size of it is called the domination number of the graph. The problem of testing whether the domination number of a graph is less than some k is a NP-complete decision problem. Therefore it is believed that there is no efficient algorithm for finding the minimum dominating set of a graph.

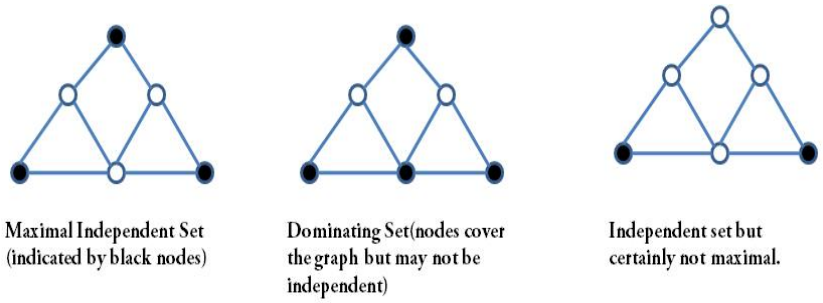

Figure 1.2 Independent Sets vs Dominating Sets

Definition 1.1.4 *t-Ruling Set : A t-ruling set RS_t of a graph $G(V, E)$ is defined as a subset of V such that no two vertices in RS_t are neighbors and any vertex $v \in V$ either belongs to RS_t or has a vertex $w \in RS_t$ such that $dist(v, w) \leq t$. Therefore a maximal independent set (MIS) is a 1-ruling set. $dist(v, w)$ refers to the minimum distance between v and w in the graph. Refer to Figure 1.3.*

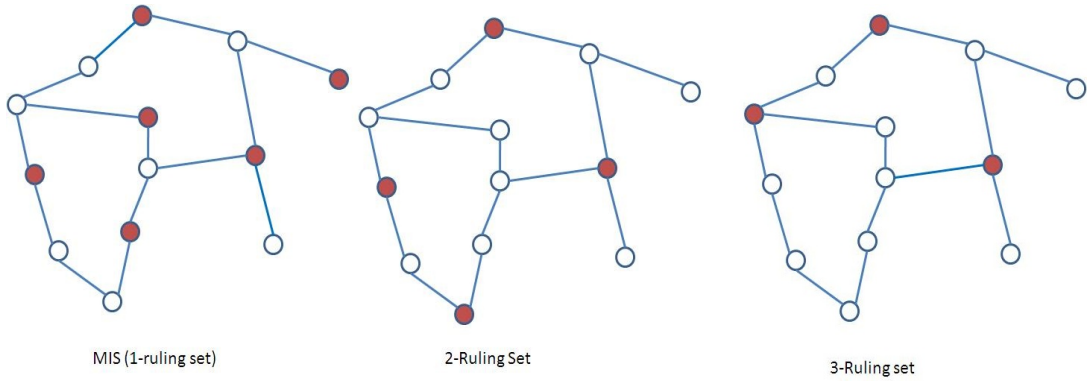

Figure 1.3 Ruling Sets

The colored nodes mark the ruling set in each case. We can observe from the above graph that the ruling set becomes sparser as we go from 1 to 3-ruling set.

Lemma 1.1.5 *Upper tail version of Chernoff bounds : Let $X := \sum_{i=1}^{n} X_i$ with $E[X_i] = p$ for each $1 \leq i \leq n$, where $X_i (1 \leq i \leq n)$ are independent random variables. Then, $\Pr[X \geq E[X] \cdot (1 + \epsilon)] \leq \exp(-E[X]\epsilon^2/3)$ for any $0 < \epsilon < 1$.*

Definition 1.1.6 *With high probability : Given a graph $G = (V, E)$ with $n = |V|$, if the probability of occurrence of an event is at least $1 - 1/n^c$ for some constant $c \geq 1$, the event is said to happen* with high probability.

Definition 1.1.7 *Symmetry Breaking: It is the process of differentiating between nodes in a graph to speed up a distributed algorithm.*

Definition 1.1.8 *Girth of a graph : It is the length of the shortest cycle contained in the graph. If the graph does not contain any cycle, its girth is defined to be infinity.*

Definition 1.1.9 *Growth-Bounded Graph : A graph G is called f-growth-bounded, if there is a function $f(r)$ such that every r neighborhood in the graph contains at most $f(r)$ independent vertices. A r neighborhood of a vertex v is collection of all its neighbors within a distance r.*

Definition 1.1.10 *Arboricity $a(G)$: The arboricity of a graph G denoted by $a(G)$ is the minimum number of forests into which its edge set can be partitioned. Equivalently, it is the minimum number of spanning forests needed to cover all the edges of the graph. The arboricity is a measure of how dense the graph is. Nash Williams defines arboricity of a graph as the $\max\{m_s/n_s - 1\}$, where m_s and n_s denotes the number of edges and vertices in any subgraph S of G.*

Definition 1.1.11 *$poly(logn)$: Polynomial function of $logn$*

Therefore, a tree has an arboricity of 1, in fact any forest has an arboricity of 1. Any planar graph with n vertices has at most $3n - 6$ edges, therefore any planar graph has arboricity at most 3. The following figure shows the complete bipartite graph $K_{4,4}$ with colors indicating the edge distribution into forests. Also $K_{4,4}$ cannot be divided into any lesser number of forests because $K_{4,4}$ has 16 edges, and any forest can only cover at most 6 edges since it can contain at most 7 vertices. Therefore, at least 3 forests are needed to cover whole of $K_{4,4}$, hence an arboricity of 3. Refer to Figure 1.4.

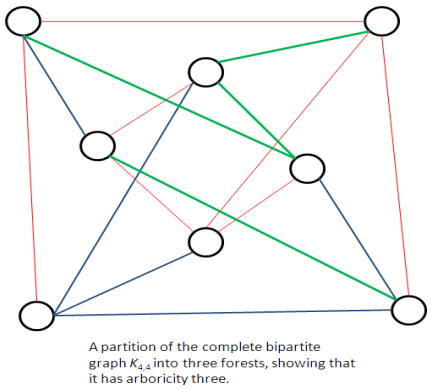

Figure 1.4 $K_{4,4}$ graph

1.2 Model and Distributed Computing

We consider distributed systems that can be modeled by a graph $G = (V, E)$ with the vertices representing the computational entities and the edges representing communication links between pairs of computational entities. We use the standard synchronous message passing model of communication in which each node, in each round, can send a possibly distinct message along each incident edge. Synchronous message passing can be viewed as availability of a global clock to all the nodes allowing communication to happen only at particular time instants. We use n to denote $|V|$ and Δ to denote the degree of the graph G. We denote by $N(v)$ the neighborhood of v and by $\deg_G(v)$ the quantity $|N(v)|$. We use $\Delta := \max_v \deg_G(v)$. Let $\text{dist}_G(u, v)$ refer to the shortest distance between any two vertices u and v in G. For a subset of vertices $V' \subseteq V$, let $G[V']$ be the subgraph induced by the subset V'. We define the arboricity of a graph as follows. Let the *density* of a graph $G = (V, E)$, $|V| \geq 2$, be the ratio $\lceil |E|/(|V| - 1) \rceil$. Let the density of a single-vertex graph be 1. The *arboricity* of a graph $G = (V, E)$, denoted $a(G)$, can be defined as $a(G) := \max\{density(G') \mid G' \text{ is a subgraph of } G\}$. By the celebrated Nash-Williams decomposition theorem [15], the arboricity of a graph is exactly equal to the minimum number of forests that its edge set can be decomposed into. For example, trees have arboricity one. The family of graphs with arboricity $a(G) = O(1)$ includes all planar graphs, graphs with treewidth bounded by a constant, graphs with genus bounded by a constant, and the family of graphs that exclude a fixed minor.

Our algorithms are randomized, which means they use uniformly random bits as an auxiliary input to guide its behavior and typically have several phases. Our algorithms involve nodes to choose uniform random numbers and require each node v to inform its neighbors of the random choice made by v. The random choice can be seen as a bit string of length $O(\log n)$ where $n := |V|$. Therefore, communication in our algorithm can be viewed as occurring in in the $\mathcal{CONGEST}$ model in which each node is only allowed to send a $O(\log n)$ sized message to each of its neighbors. However our algorithm uses MIS algorithms from Barenboim et al. [2],which are designed to run in the $\mathcal{LOCAL}$ model, which allows

each node to send a message of arbitrary size to each neighbor in each round. Thus, due to their dependency on the MIS algorithms of Barenboim et al. [2], the algorithms in this thesis also require the use of the $\mathcal{LOCAL}$ model.

1.3 Contribution of the Thesis

All the algorithms we present in this thesis are randomized and synchronous in nature. First we present an algorithm which computes a 2-ruling set in any graph in $\frac{\log \Delta}{\log \log n} + e^{O(\sqrt{\log \log n})}$rounds with high probability. We then use this to present another algorithm which can compute a 2-ruling set for graphs of arboricity $a = 2^{O(\sqrt{\log n})}$ in $O(\sqrt{(\log n / \log \log n)})$ rounds with high probability.

We then extend the ideas from the above algorithm and show that a $(t+1)$-ruling set for any $t \geq 1$ in graphs with arboricity $a = O(\log \log n)$ can be computed in $O(t \log^{1/t} \Delta + t^2 (\log \log n)^{1/t} + (\log \log n)^2)$ rounds. Therefore, a $\log \log \Delta$-ruling set for bounded arboricity graphs can be computed in $O((\log \log n)^2)$ rounds. Our results therefore indicate that while computing an MIS in graphs of bounded arboricity in $O(\sqrt{\log n})$ rounds is still an open problem, ruling sets for small values of t can be found in much fewer rounds. Similarly, a $(t+1)$-ruling set for a general graph can be computed in $O(t \log^{1/t} \Delta + t^2 (\log \log n)^{1/t} + e^{O(\sqrt{\log \log n})})$ rounds. Also, a t-ruling set for graphs of girth > 6 can be computed in $O(t \log^{1/t} \Delta + t^2 (\log \log n)^{1/t} + e^{O(\sqrt{\log \log n})})$ rounds.

In a later section, we compare the run time of Luby's [13] and Métivier's [14] MIS algorithms, two fundamental algorithms for computing an MIS in general graphs. We also try to understand the effect of probability of a node getting selected in Luby's algorithm on the number of rounds required by algorithm to finish. We run these algorithms on different types of graphs, namely Random, Regular and Bipartite Graphs.

Chapter 2

Background And Related Work

Symmetry breaking is of fundamental interest in distributed computing. A classical example involving symmetry breaking is the computation of a maximal independent set (MIS) in a graph $G = (V, E)$. An MIS of G is an independent set that is maximal with respect to inclusion. Independence above mean that no two nodes in MIS should be neighbors. The fastest algorithm for general graphs so far is still that of Luby [13] and runs in $O(\log n)$ rounds where $n = |V|$.

Computing a maximal independent set (MIS) is a fundamental problem in distributed computing for its ability to nicely capture the essential challenge of symmetry breaking and also for its myriad applications to other problems. It is not known if there exist any poly-logarithmic round deterministic algorithms for computing an MIS in the distributed setting. Randomization however has been quite helpful, and an $O(\log n)$ round randomized algorithm for computing an MIS in a general graph of n vertices has been known since the work of Luby [13] and Alon et al. [1]. Since the breakthrough result of [13, 1], no faster algorithm for general graphs is reported. Only recently, Barenboim et al. [2], have used techniques first designed in [7], to show that an MIS in a general graph can be computed in $O(\log \Delta.\sqrt{\log n})$ rounds, where Δ is the degree of the graph. The runtime of this algorithm is sub-logarithmic whenever $\Delta \in O(2^{\sqrt{\log n}})$. On the other hand, it is shown by Kuhn et al. [9], that $\Omega(\sqrt{\log n})$ rounds are required by *any* algorithm to compute an MIS in the distributed setting. A challenging open problem at present is to see if there exist sub-logarithmic algorithms for MIS given the gap between the lower bound from [9, 10] and the upper bound results of [1, 13, 2]. However, when one focuses on special graph classes, one finds that there are indeed several significant success stories. It is shown in [4] that an MIS in graphs of bounded degree can be found in $O(\log^* n)$ rounds, which also meets the lower bound results from [12]. Similarly, Wattenhofer and Schneider [17] show that a $O(\log^* n)$ round deterministic algorithm can be designed for also growth-bounded graphs. The work of Barenboim et al.[2], extends a key theorem from [11] to the case of graphs with girth greater than 6 and shows that also on such graphs an MIS can be computed in $\tilde{O}(\sqrt{\log n} + \exp \sqrt{\log \log n})$ rounds.

Graphs with bounded arboricity are an important class of graphs that is being considered extensively in distributed computing in recent years. Define the density of a graph to be the ratio of the number of edges in the graph to the number of vertices. The arboricity of a graph G denoted by $a(G)$ or simply a

can be defined as the maximum density of any subgraph of G. The graphs of bounded arboricity includes planar graphs, graphs of bounded genus, graphs of bounded treewidth, graphs obtained by excluding a fixed minor, bounded degree graphs, and the like. Lenzen and Wattenhofer [11] show that an MIS in a tree, which has arboricity 1 by definition, can be found in just $\tilde{O}(\sqrt{\log n})$ rounds.(We use $\tilde{O}(f(n))$ to indicate a runtime of $O(f(n)\log f(n))$.) A small technical error in their proof is corrected in [2]. The authors of [2] also show that an MIS in a graph of arboricity a can be found in $O(\min\{\log a\sqrt{\log n} + \log^{3/4} n, \log\Delta + a + \log\log n\})$ rounds for general values of arboricity a, and in $O(\log\Delta \cdot (\log\Delta + \frac{\log\log n}{\log\log\log n})$ rounds when $a \in O(\log^{1/2-\epsilon}\log n)$. Lenzen and Wattenhofer [11] leave it as an open problem if their analysis can be extended to also graphs of arboricity beyond $a = 1$.

We approach this problem from the direction of ruling sets. A relaxed version of computing an MIS is the problem of computing a t-ruling set. A t-ruling set for an integer $t \geq 1$ in a graph G is an independent subset $S \subseteq V$ such that every nodes $v \in V$ is either in S or has a neighbor $w \in S$ at a distance at most t from v. Computing a t-ruling set also involves symmetry breaking considerations.

A question of recent interest is to see whether it is possible to design algorithms for a t-ruling set for small values of t and outperform corresponding algorithms for an MIS. The recent work of Kothapalli and Pemmaraju [6] shows that a 3-ruling set in a graph of arboricity $a = O(1)$ can be computed in just $O((\log\log n)^2)$ rounds with high probability. They use a technique called rapid sparsification in which in the i^{th} round for $i \geq 1$, a subset of vertices with degree in $[\Delta^{1/2^{i-1}}, \Delta^{1/2^i}]$ is chosen probabilistically and removed along with its neighbors. This ensure that all the vertices with degree in $[\Delta^{1/2^{i-1}}, \Delta^{1/2^i}]$ are removed at the end of round i. They approached the problem of Maximal Independent set i.e. a 1-ruling set from the direction of computing a fast t-ruling set for a small t and posing an open question as to whether it can be extended fast enough to get an MIS. They first describe their 2-ruling set algorithm which requires $O(\frac{\log\Delta}{(\log n)^\epsilon} + (\log n)^{\frac{1}{2}+\epsilon})$ rounds with high probability to compute a 2-ruling set for any general graph. They also present faster algorithms for restricted classes of graph, high girth and bounded arboricity graphs including trees. For graphs G with girth at least 6 they present an algorithm which computes a 3-ruling set of G and is expected to finish in $O(\sqrt{\log\log n})$ rounds with high probability. Also for trees to compute a 3-ruling set it requires $O((\log\log n)^2\log\log\log n)$ rounds with high probability. For graphs G with arboricity $a \in (\log n)^k$ for a constant k, a 3-ruling set can be computed in $O(\sqrt{\log n}(\log\log n)^2 + \log^{3/4} n\log\log n)$ with high probability. Further if $a = O(1)$, then it can be done in $O((\log\log n)^3)$ with high probability.

Here we briefly describe the 2-ruling algorithm for general graphs [6]. The algorithm proceeds in stages. They use a variable f to balance the number of stages required with the time required to compute an MIS in selected subgraph. At the end of stage i the maximum degree in the graph will be Δ/f^i with high probability. So it requires $\log_f\Delta$ rounds. The total time required will be $\log_f\Delta$ plus the time required to compute an MIS in a graph of maximum degree $\Delta = f\log n$. In each stage i, they select a subgraph M_i of low degree and then remove M_i along with its neighbors denoted by W_i from the graph. They later compute an MIS on $H = \cup M_i \forall i$ and since they removed the neighbors of M_i earlier, the MIS of H turns out to be a 2-ruling set for the entire graph. Also the removal of neighbors of M_i

at each stage ensure all the M_i are mutually independent and hence the union of them is also of low degree. M_i denotes the set of vertices selected randomly in round i and W_i denotes the set of vertices which are neighbor of vertices in M_i. This is depicted in the figure below.

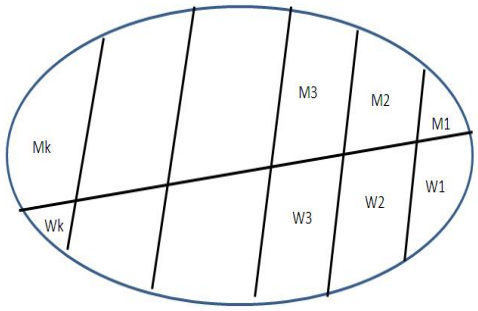

Figure 2.1 RAPID SPARSIFICATION

In the paper [6], the authors develop a novel technique to compute fast 3-ruling sets for restricted families of graphs. They make use of the special properties of graph which is lots of independent nodes in a neighborhood in case of high girth graphs or bounded arboricity graphs. The general idea they follow is similar to the 2-ruling set but this time they do it in less number of steps. Earlier they used degree range $[\Delta/f, \Delta]$, $[\Delta/f, \Delta/f^2]$ but for 3-ruling set they use $[\Delta^{1/2}, \Delta]$, $[\Delta^{1/4,\Delta^{1/2}}]$ therefore now they only require $O(\log\log n)$ steps to go from Δ as the maximum degree to a constant.

Now we describe their first phase which reduces the maximum degree from Δ to $\Delta^{1/2}$. Let us call the vertices in the range $[\Delta^{1/2}, \Delta]$ as *high degree vertices*. Now all the *high degree vertices* selects themselves with probability $6.\log n/\Delta^{\frac{1}{2^i-1}}$ and all the remaining vertices with degree less than $\Delta^{1/2}$ called *low degree vertices* selects themselves with probability $6.\log n/\Delta^{\frac{1}{2^i}}$. This slight increase in probability for the *low degree vertices* ensures that all the *high degree vertices* have at least a neighbor selected in its 2-distance neighborhood. Next they compute an MIS in the subset of selected nodes and remove the 2-distance neighborhood of selected nodes, therefore any node can be at most at a distance of 3 from a node in MIS, hence a 3-ruling set. The selection probability ensures that no selected vertex have too many selected neighbors. MIS of selected nodes is computed in the following way, first the MIS of *high degree* selected nodes is computed and its neighbors are removed from the graph. Then the same is done for *low degree vertices* and the union of the MIS is returned. The probability used in the algorithm ensures that the degree of the graph of the selected vertices in either *high degree vertices* or *low degree vertices* is low enough to compute a fast MIS, where we make use of the fast MIS algorithm for poly logarithmic degree [16]. The algorithm is further explained with the help of a figure below. The colored nodes in the figure represents the vertices which are selected in this particular round randomly. As we see from the figure every *high degree vertices* is at a distance of at most 2 from a selected node. Now we compute a MIS of all the selected nodes, therefore every *high degree vertex* is at most at a distance of 3 from some node in MIS, hence a 3-ruling set.

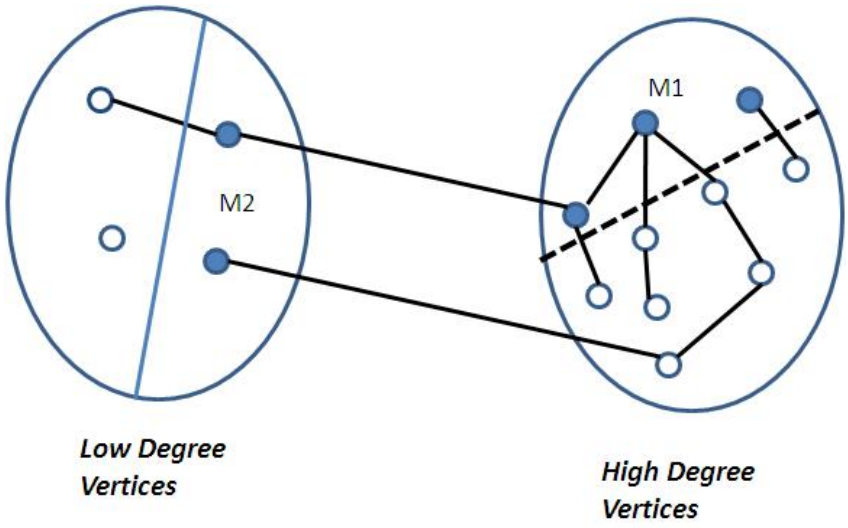

Figure 2.2 Stage i of 3-Ruling Set algorithm

Another interesting recent work on computing ruling sets is that of Gfeller and Vicari [3]. In [3], the authors show that a *not-necessarily independent* subset $S \subseteq V$ can be computed with the property that every node $v \in V$ has some node $w \in S$ at a distance at most $O(\log\log\Delta)$. This algorithm also runs in $O(\log\log n)$ rounds with high probability. While the result of [3] is applicable for general graphs, our result indicates that an *independent* subset with the above property can be computed in just $O((\log\log n)^2)$ rounds for graphs with arboricity $a = O(\log\log n)$. Notice that the subset produced by [3] might not be independent and converting it to an independent subset will require $O(e^{\sqrt{\log\log n}})$ rounds for a general graph, hence our result for graphs of arboricity $a = O(\log\log n)$ is better. A motivation for working with ruling sets is somehow extending it into Maximal Independent sets fast. A favorable result for the above is by Gfeller and Vicari [3] in case of growth bounded graphs. A graph G is called f-growth-bounded, if there is a function $f(r)$ such that every r neighborhood in the graph contains at most $f(r)$ independent vertices. They present an algorithm which can compute a MIS in $O(\log\log n \log^* n)$ rounds for growth bounded graphs. Their algorithm consists of three fundamental parts. First they compute a $O(\log\log n)$ ruling set(not independent), the induced subgraph of which has a maximum degree of $O(\log^5 n)$. Then they convert this into an $O(\log\log n)$ independent ruling set in phase 2. In the last phase they extend the $O(\log\log n)$ independent ruling set into a Maximal Independent Set. Their first phase works for any graph, so even for a general graph they can compute a $O(\log\log\Delta)$-ruling set(not independent) with maximum induced degree $O(\log^5 n)$ in $O(\log\log\Delta)$ rounds with high probability. Phase 2 and 3 uses technique from [8] which requires special property of growth bounded graph.

Graph Class	**MIS**	**Source**
General	$O(\log n)$	[13, 1]
General	$O(\log \Delta \cdot \sqrt{\log n})$	[2]
Trees	$\tilde{O}(\sqrt{\log n})$	[11, 2]
Girth ≥ 6	$O(\log \Delta \log\log n + e^{O(\sqrt{\log\log n})})$	[2]
Growth Bounded	$O(\log^* n)$ Deterministic	[17]
Arboricity $a \in O(\log^{1/2-\epsilon} \log n)$	$O(\log \Delta(\log \Delta + \frac{\log\log n}{\log\log\log n}))$	[2]
Arboricity a	$O(\min\{\log a\sqrt{\log n} + \log^{3/4} n, \log \Delta + a + \log\log n\})$	[2]

Table 2.1: Runtime of distributed algorithms for MIS.

Graph Class	**2-rs**	**Source**
General	$O(\log \Delta/(\log n)^{\epsilon} + (\log n)^{\epsilon+1/2})$	[6]
General	$O(\log \Delta/\log\log n)$ + $e^{O(\sqrt{\log\log n})}$	This Thesis
Girth ≥ 6	$O(\sqrt{\log \Delta \log\log n} + e^{O(\sqrt{\log\log n})})$	This Thesis
Growth Bounded	$O(\log^* n)$ Deterministic	[17]
Arboricity $a \in O(\log\log n)$ and $\Delta > 2^{(\log\log n)^3}$	$O((\log \Delta)^{2/3} + (\log\log n)^2)$	This Thesis
Arboricity $a \in O(2^{\sqrt{\log n \log\log n}})$	$O(\sqrt{\log n/\log\log n})$	This Thesis

Table 2.2: Runtime of distributed algorithms for 2-ruling sets

Chapter 3

2-Ruling Sets

Computing a 2-ruling set requires symmetry breaking at its core. A 2-ruling set RS_2 of a graph $G(V, E)$ is defined as a subset of V such that no two vertices in RS_2 are neighbors and any vertex $v \in V$ either belongs to RS_2 or has a vertex $w \in RS_2$ such that $dist(v, w) \leq 2$. In this chapter we present our 2-ruling set algorithms for general and bounded arboricity graphs.

3.1 2-ruling set in general graph

Given a graph $G = (V, E)$, we present our algorithm which computes a 2-ruling set of G in $\frac{\log \Delta}{\log \log n} + e^{O(\sqrt{\log \log n})}$rounds where Δ is the maximum degree of the graph at the starting of the algorithm. Here we describe the algorithm, the reader is encouraged to read the pseudo code of the algorithm. The algorithm proceeds in phases. In phase i, all the vertices still active select themselves with probability $\frac{6.\log^{i+1} n}{\Delta}$, we call this subset of vertices as M_i. We then remove M_i and all the neighbors of M_i from the graph thereby making them inactive for further rounds. We also see that the maximum degree of a node in subgraph $H = G(M_i)$ is bounded by $12 \log^2 n$ and so an MIS can be computed very fast on H. Using Srinivasan and Panconesi's algorithm as shown by Barenboim and Elkin, an MIS of a graph with maximum degree $\Delta = poly(\log n)$ can be computed in $e^{O(\sqrt{\log \log n})}$rounds, which we use in our algorithm. Also notice since we remove the neighbors of M_i in each phase, M_i for all i are mutually independent, therefore we can compute an MIS of each M_i in parallel which we do at the end of all phases, thereby saving some time. The maximum degree of the graph is reduced to $\Delta/(\log^i n)$ after the end of phase i as we show later. Therefore we only require $\log \Delta / \log \log n$ phases. Below we present the pseudo code of our algorithm.

Algorithm 2RULINGSET($G = (V, E)$)
Begin
1. $I \leftarrow \phi; H \leftarrow \phi$
2. for $i = 1, 2, \cdots, \log \Delta / \log \log n$ do
3. $M_i \leftarrow \phi; W_i \leftarrow \phi$

4. for each $v \in V$ in parallel do
5. $M_i \leftarrow M_i \cup v$ with probability $\frac{6.\log^{i+1} n}{\Delta}$
6. for each v in parallel do
7. if $v \in N(M_i) \setminus M_i$, then $W_i \leftarrow W_i \cup v$
8. $V \leftarrow V \setminus (M_i \cup W_i)$
9. $H \leftarrow H \cup M_i$
10. end-for(i)
11. return $I \cup \text{MIS}(H \cup G)$

End

Lemma 3.1.1 *The maximum degree of the subgraph induced by any $M_i(1 \leq i \leq \log\Delta/\log\log n)$ is $12(\log n)^2$. Also since $M_i(1 \leq i \leq \log\Delta/\log\log n)$ are mutually independent, therefore the maximum degree in $H := G[\cup_i M_i]$ is also $12(\log n)^2$ with high probability.*

Proof. Any vertex in H can have at most $\frac{\Delta}{\log^{i-1} n}$ neighbors at the starting of round $i-1$. Thus,

$$E[deg_H(v)] \leq \frac{\Delta}{\log^{i-1} n} \cdot \frac{6\log^2 n}{\Delta/\log^{i-1} n} = 6.\log^2 n.$$

Since vertices join H independently, using Chernoff bounds we conclude that $Pr[deg_H(v) > 12.\log^2 n] \leq 1/n^2$. Therefore with probability at least $1-1/n$ all such subgraphs will have maximum degree bounded by $12\log^2 n$. □

Lemma 3.1.2 *The degree of any vertex in G after i rounds($1 \leq i \leq \log\Delta/\log\log n$), is at most $\Delta/(\log n)^i$ with high probability.*

Proof. Consider a vertex $v \in V$ at the start of iteration i that has degree greater than $\frac{\Delta}{\log^i n}$. Therefore v has at least $\frac{\Delta}{\log^i n}$ neighbors trying to join set H in this round. Hence,

$$\Pr[\text{at least some neighbor of } v \text{ joins } H] \geq 1 - (1 - \frac{6.\log^2 n}{\Delta/\log^{i-1} n})^{\frac{\Delta}{\log^i n}}. \geq 1 - e^{-6.\log n} = 1 - \frac{1}{n^6}.$$

Therefore, v will be removed in this round with probability at least $1 - \frac{1}{n^5}$. □

Theorem 3.1.3 *A 2-ruling set in a general graph can be computed in $O(\frac{\log\Delta}{\log\log n} + e^{O(\sqrt{\log\log n})})$ rounds. This result is an improvement over the $O\left(\frac{\log\Delta}{(\log n)^\epsilon} + (\log n)^{1/2+\epsilon}\right)$ round algorithm from [6] whenever $\Delta \in O(2^{\sqrt{\log n}})$.*

Proof. We first prove that Phase II computes a 2-ruling set in a general graph in $O(\log \Delta / \log\log n + e^{O(\sqrt{\log\log n})})$ rounds, where Δ is the maximum degree of the graph. We make use of Lemma 3.1.1 and 3.1.2 to prove this. First we prove it finishes in $O(\log \Delta / \log\log n + e^{O(\sqrt{\log\log n})})$ rounds. Lemma 3.1.2 state that at the end of i^{th} round, the maximum degree of the graph is reduced to $\Delta/(\log n)^i$. Therefore, to reduce the maximum degree of the graph to $12(\log n)^2$, $\log \Delta / \log\log n$ rounds suffice. The rest is computing a MIS in H, which has a maximum degree of $12(\log n)^2$ with high probability according to Lemma 3.1.1. To compute a MIS in H, $e^{O(\sqrt{\log\log n})}$ rounds suffice using Panconesi-Srinivasan algorithm [16]. Now we need to prove the result is correctly a 2-ruling set of G. Let V_{rem} denote the set of vertices which doesn't join M_i or W_i for any i, $1 \leq i \leq \log \Delta / \log\log n$. Notice, every vertex v either joins H or W_i for some i or is in V_{rem}. Since we compute a MIS in $H \cup V_{rem}$, it is definitely a 2-ruling set for vertices in $H \cup V_{rem}$. Now every vertex $w \in W_i$ for some i, has a neighbor $v \in M_i$ and every such v either is in MIS or has a neighbor in MIS, therefore w is at most a distance of 2 from some vertex in MIS. Therefore it is a 2-ruling set of G. □

Remark 3.1.4 *For graphs with arboricity $a = O(\log\log n)$, a 2-ruling set can be computed by Phase II in $O(\log \Delta / \log\log n + (\log\log n)^2)$ rounds using Theorem 6.4 [2].*

3.2 2-ruling set in graphs of bounded arboricity

In this section, we design an algorithm that can compute a 2-ruling set in a graph of arboricity $a \in 2^{O(\sqrt{\log n})}$ in $O(\sqrt{\log n / \log\log n})$ rounds. Our algorithm acts in two phases. Line 2 of the algorithm 2RULINGSET-BoundedArboricity$(G = (V, E))$ corresponds to the Phase I and lines 3 to 11 to Phase II. In Phase I, which runs for $\log_t n$ rounds for some parameter $t \geq max(a^8, (c \log n)^7)$, we use Métivier's algorithm [14] to get an independent set and reduce the maximum degree of the graph fast. Phase II as we proved earlier computes a 2-ruling set in a general graph in $O(\frac{\log \Delta}{\log\log n} + e^{O(\sqrt{\log\log n})})$ rounds, where Δ is the maximum degree of the graph. Let $r := c_1 \sqrt{\frac{\log n}{\log\log n}}$ be number of rounds required by Phase I. Also, $\Delta = a \cdot 2^{O(\sqrt{\log n \log\log n})}$ for phase II of our algorithm.

Algorithm 2RULINGSET-BoundedArboricity$(G = (V, E))$
Begin
1. $I \leftarrow \phi; H \leftarrow \phi$
2. Run Métivier's algorithm [14] for r rounds and let I be the nodes in the MIS.
3. for $i = 1, 2, \cdots, \log \Delta / \log\log n$ do
4. $M_i \leftarrow \phi; W_i \leftarrow \phi$
5. for each $v \in V$ in parallel do
6. $M_i \leftarrow M_i \cup v$ with probability $\frac{6.\log^{i+1} n}{\Delta}$
7. for each v in parallel do
7. if $v \in N(M_i) \setminus M_i$, then $W_i \leftarrow W_i \cup v$

8. $V \leftarrow V \setminus (M_i \cup W_i)$
9. $H \leftarrow H \cup M_i$
10. end-for(i)
11. return $I \cup \text{MIS}(H \cup G)$

End

3.2.1 Analysis

The progress during Phase I of our algorithm is captured by the following theorem from [2].

Theorem 3.2.1 (BEPS12) *Let G be a graph of arboricity $a := a(G)$. After $\log_t n$ rounds of Métivier's algorithm, it holds that the degree of the graph induced by nodes still remaining is bounded by $O(a \cdot t)$ with high probability, given $t \geq max(a^8, (c \log n)^7)$.*

The following lemmata concern phase II of our algorithm and helps in proving Theorem 3.2.4. The maximum degree of the graph at the start of phase II is $\Delta = a \cdot 2^{O(\sqrt{\log n \log\log n})}$.

Lemma 3.2.2 *The maximum degree of the subgraph induced by any $M_i (1 \leq i \leq \log \Delta / \log\log n)$ is $12(\log n)^2$. Also since $M_i (1 \leq i \leq \log \Delta / \log\log n)$ are mutually independent, therefore the maximum degree in $H := G[\cup_i M_i]$ is also $12(\log n)^2$ with high probability.*

Lemma 3.2.3 *The degree of any vertex in G after i rounds($1 \leq i \leq \log \Delta / \log\log n$), is at most $\Delta/(\log n)^i$ with high probability.*

Theorem 3.2.4 *A 2-ruling set in a graph of arboricity $a \in 2^{O(\sqrt{\log n})}$ can be computed in $O(\sqrt{\log n / \log\log n})$ rounds.*

Proof. Let us choose $t = 2^{O(\sqrt{\log n \log\log n})}$. Then, the time taken by Phase I of our algorithm is $O(\sqrt{\frac{\log n}{\log\log n}})$. At the end of Phase I, the maximum degree Δ of the remaining graph is bounded by $a \cdot t = a \cdot 2^{O(\sqrt{\log n \log\log n})}$. If $a = 2^{O(\sqrt{\log n \log\log n})}$, phase II requires $O(\sqrt{\frac{\log n}{\log\log n}} + e^{\sqrt{\log\log n}})$ rounds. Therefore the total runtime of our algorithm is $O(\sqrt{\frac{\log n}{\log\log n}})$ rounds. This finishes our proof. □

3.3 2-Ruling Set in Graphs of High Girth

In this section, we present our result of computing a 2-ruling set in graphs with girth > 6 in $O(\sqrt{\log \Delta \log\log n}) + e^{O(\sqrt{\log\log n})}$rounds.

High girth graph provides us with an advantage as we have faster MIS algorithm due to Elkin et al. [2] as described by the theorem below. We use the following result combined with the algorithm to compute a 2-ruling in general graph [6] to get the stated result.

Theorem 3.3.1 *Given a graph G of girth greater than* 6*, a MIS of G can be computed in* $O(\log \Delta \log\log n+)$ $e^{O(\sqrt{\log\log n})}$ *rounds with high probability.*

As we have already described earlier, the algorithm to compute a 2-ruling in a general graph [6] works as follows. It has a variable f, the algorithm has two phases; first one runs for $O(\log_f \Delta)$ rounds, the second phase comprises of computing a MIS in a subgraph of maximum degree $\Delta = O(f \log n)$. So, we try to balance the time required for the two phases as depicted by following equations.

$$\text{Time required} = \log_f \Delta + \log(f \log n) \log\log n + e^{\sqrt{\log\log n}} \tag{3.1}$$

Therefore to minimize the time we equate below which leads us to our result

$$\log_f \Delta = \log(f \log n) \log\log n \tag{3.2}$$

We now show our main theorem.

Theorem 3.3.2 *Given a graph G with* $girth > 6$*, a 2-ruling set of G can be found in* $O(\sqrt{\log \Delta \log\log n})+$ $e^{O(\sqrt{\log\log n})}$ *rounds.*

Proof. Using equation 3.2 we get $\log f = \sqrt{\frac{\log \Delta}{\log\log n}}$. Using this value of f we get our desired result. □

Comparing the run time for computing a MIS and 2-ruling set for graphs of girth > 6, we can see that it is quite an improvement for a slightly relaxed problem.

3.4 2-Ruling Set in Graphs of arboricity $a = O(\log\log n)$

In this section, we present our result of computing a 2-ruling set in a graph with arboricity $a = O(\log\log n)$
and $\Delta > 2^{(\log\log n)^3}$ in $O(\log \Delta)^{2/3} + O((\log\log n)^2)$ rounds.

Similar to the case of high girth graph we have faster MIS algorithm for bounded arboricity graphs too [2], which we present in the following theorem.

Theorem 3.4.1 *For more general values of a, Bareboim et al. [2] show that an MIS in a graph of arboricity a can be computed in* $min\{\log a\sqrt{\log n} + \log^{3/4} n, \log^2 \Delta + a \log \Delta + a^\epsilon \log\log n, \log^2 \Delta + a^{1+\epsilon} \log \Delta + \log a \log\log n\}$ *rounds with high probability, for a constant* $\epsilon > 0$.

We follow the same technique used in 2-ruling in high girth graphs.

$$\text{Time required } = \frac{\log \Delta}{\log f} + \log^2(f \log n) + a \log(f \log n) + a^{\epsilon} \log \log n \tag{3.3}$$

$$\text{If } a = O(\log \log n) \text{ and } f > \log n;\ \log \Delta = \log^3 f \tag{3.4}$$

We now present our main theorem.

Theorem 3.4.2 *Given a graph $G = (V, E)$ with $a = O(\log \log n)$ and $\Delta > 2^{(\log \log n)^3}$ we can compute a 2-ruling set in $O(\log \Delta)^{2/3} + O((\log \log n)^2)$ rounds.*

Proof. Using equation 3.4 we get $\log \Delta = \log^3 f$. Using this value of f we get our desired result. □

Chapter 4

t-Ruling Sets

4.1 t-Ruling Sets in graphs of bounded Arboricity

A t-ruling set RS_t of a graph $G(V, E)$ is defined as a subset of V such that no two vertices in RS_t are neighbors and any vertex $v \in V$ either belongs to RS_t or has a vertex $w \in RS_t$ such that $dist(v, w) \leq t$.

In this section we will describe the algorithm RULINGSET(G, t, Δ), which computes a t-ruling set for graphs $G = (V, E)$ of maximum degree Δ and arboricity $a = O(\log \log n)$ for any $t > 1$. Our algorithm is a variant of the algorithm used to obtain a 2-ruling set in general graphs in [6]. Our algorithm consists of two phases. In phase I, we process the original graph to get a subgraph H with a smaller degree. Phase II computes a $(t-1)$-ruling set of H which also is a t-ruling set of G. Phase I of the algorithm proceeds in rounds. We define f to be a variable depending on t and Δ. In round i, for $i = 1, 2, \cdots$, we extract a subset M_i of vertices of degree in the range $[\Delta/f^{i-1}, \Delta/f^i]$ and make vertices in M_i and its neighbors inactive for further rounds. Phase I proceeds until the maximum degree of the graph reduces to $f \log n$. Now in Phase II, a $(t-1)$-ruling set of what remains of G along with $H := G[\cup_i M_i]$ is computed in a similar way, which is actually a t-ruling set of G. As Lemma 4.1.1 shows, the maximum degree of H will be $O(f \log n)$ with high probability. This helps in computing a $(t-1)$-ruling set in H faster.

We can apply our algorithm recursively. To compute a $(t-1)$-ruling set of a graph G, we in turn compute a $(t-2)$-ruling set of a subgraph H of G. Recursion stops at $t = 2$, where we use the result from Remark 3.1.4.

The number of rounds required for both phases depends on f. As we increase f, the number of rounds needed for phase I decreases but we then have to handle a large degree subgraph in phase II. Alternatively, if we keep f too small, phase I requires too many rounds. Therefore we select f such that number of rounds required by both the phases are nearly the same.

Algorithm RULINGSET($G = (V, E), t, \Delta$)
Begin
1. If $t = 2$
2. return a 2-ruling set of G computed using our 2-ruling algorithm for bounded graphs

that require $O(\log \Delta + (\log \log n)^2)$ rounds with high probability.

3. Else
4. $f \leftarrow 2^{\log^{\frac{t-2}{t-1}} \Delta}$; $H \leftarrow \phi$
5. for $i \leftarrow 1, 2, ... \log_f \Delta$
6. $M_i \leftarrow \phi$; $W_i \leftarrow \phi$
7. for each $v \in V$ parallel, do
8. $M_i \leftarrow M_i \cup v$ with probability $\frac{6 \log n f^i}{\Delta}$
9. for each v in parallel do,
10. if $v \in N(M_i) \setminus M_i$, then $W_i \leftarrow W_i \cup v$
11. $V \leftarrow V \setminus (M_i \cup W_i)$
12. $H \leftarrow H \cup M_i$
13. end-for(i)
14. return RULINGSET$(H \cup G, t-1, 12 \cdot f \log n)$

End

The following lemmata help in proving Theorem 4.1.3.

Lemma 4.1.1 *The maximum degree of the subgraph induced by any $M_i (1 \leq i \leq \log_f \Delta)$ is $12f \log n$. Also since $M_i (1 \leq i \leq \log_f \Delta)$ are mutually independent, therefore the maximum degree in $H := G[\cup_i M_i]$ is also $12f \log n$ with high probability.*

Proof. Any vertex in H can have at most $\frac{\Delta}{f^{i-1}}$ neighbors at the starting of round $i-1$. Thus,

$$E[deg_H(v)] \leq \frac{\Delta}{f^{i-1}} \cdot \frac{6 f^i \log n}{\Delta} = 6.f \log n.$$

Since vertices join H independently, using Chernoff bounds we conclude that $\Pr[deg_H(v) > 12.f \log n] \leq 1/n^2$. Therefore with probability at least $1 - 1/n$ all such subgraphs will have maximum degree bounded by $12f \log n$. □

Lemma 4.1.2 *The degree of any vertex in G after i rounds($1 \leq i \leq \log_f \Delta$), is at most Δ / f^i with high probability.*

Proof. Consider a vertex $v \in V$ at the start of iteration i that has degree greater than $\frac{\Delta}{f^i}$. Therefore v has at least $\frac{\Delta}{f^i}$ neighbors trying to join set H in this round. Hence,

$$\Pr[\text{at least some neighbor of } v \text{ joins } H] \geq 1 - (1 - \frac{6.f^i \log n}{\Delta})^{\frac{\Delta}{f^i}}. \geq 1 - e^{-6. \log n} = 1 - \frac{1}{n^6}.$$

Therefore, v will be removed in this round with probability at least $1 - \frac{1}{n^5}$. □

Theorem 4.1.3 *A $t+1$-ruling set for a graph with arboricity $a = O(\log\log n)$ can be computed in $O(t\log^{1/t}\Delta + t^2(\log\log n)^{1/t} + (\log\log n)^2)$ rounds.*

Proof. Let $N(t,\Delta)$ denote the number of rounds required to compute a t-ruling set in a graph of arboricity $a = O(\log\log n)$ and maximum degree Δ. The recurrence relation for $N(t,\Delta)$ is, $N(t,\Delta) = log_f\Delta + N(t-1, 12\cdot f\log n)$ and the base case will be $N(2,\Delta) = \frac{\log\Delta}{\log\log n} + (\log\log n)^2$. Keeping $f = 2^{(\log\Delta)^\epsilon}$ with $\epsilon = \frac{t-2}{t-1}$ balances both the terms in the recurrence. On solving the recurrence we get, $N(t,\Delta)$ = $(t\log^{1/t}\Delta + (t-1)(\log\log n)^{1/t-1} + (t-2)(\log\log n)^{1/t-2} + ...\log\log n + (\log\log n)^2)$. The right hand side can be simplified to $O(t\log^{1/t}\Delta + t^2(\log\log n)^{1/t} + (\log\log n)^2)$ rounds. □

Corollary 4.1.4 *A 3-ruling set, 4-ruling set, and a $\log\log\Delta$-ruling set can be computed for graphs with $a = O(\log\log n)$ in $O(\sqrt{\log\Delta} + (\log\log n)^2)$, $(O(\log\Delta)^{1/3} + (\log\log n)^2)$, $O(\log\log n)^2$ rounds, respectively for graph with maximum degree Δ.*

Corollary 4.1.5 *A $(t+1)$-ruling set for a general graph can be computed in $O(t\log^{1/t}\Delta + t^2(\log\log n)^{1/t} + e^{O(\sqrt{\log\log n})})$ rounds. Also, a t-ruling set for graphs of girth > 6 can be computed in $O(t\log^{1/t}\Delta + t^2(\log\log n)^{1/t} + e^{O(\sqrt{\log\log n})})$ rounds.*

Chapter 5

Comparison between Luby's and Métevier's MIS

5.1 Luby's vs Métivier's MIS

Both Luby and Métivier have an asymptotic run time of $O(\log n)$ distributed rounds. Since both the algorithms are randomized in nature, it is assumed that every vertex has infinite source of random numbers which it uses during the algorithm. They are to date the fastest known algorithms for computing an MIS in a general graph. In this chapter we compare the number of rounds required by both the algorithms on different type of graphs and see who performs better in practice. Below we describe both the algorithms and give pseudo code for them.

5.1.1 Luby's MIS

Using Luby's algorithm, a maximal independent set I of vertices V of a graph G is computed in an incremental fashion as follows. In each round every currently active vertex selects itself with probability $1/2d(v)$ where $d(v)$ denote the current degree of v, that is the number of active neighbors v has. If two neighbors v and w gets selected in the same round, then the node with smaller degree discards its earlier selection, ties are broken arbitrarily. Now it is ensured that all the vertices still selected are independent, that is no two of them are neighbors, so we can include them in MIS and remove them and their neighbors from the graph, thereby making them inactive and excluding them from consideration in remaining rounds. This goes on until all the vertices have been removed from the graph. It can be proved that expected number of rounds required to get an MIS(when all the vertices have been removed from the graph) is $O(\log n)$, n being the number of vertices in the original graph. Below we give the pseudo algorithm for Luby's MIS.

Algorithm MIS-Luby($G = (V, E)$)
Begin
1. $I \leftarrow \phi; G' \leftarrow G$
2. while (G' is not empty) do
3. Choose a random set of vertices $S \in G'$ by selecting each active vertex v

independently with probability $1/2d(v)$

4. For every edge$(u, v) \in E(G')$ if both endpoints are in S;
   then remove the vertex with lower degree from S(Break ties arbitrarily)
5. $I \leftarrow I \cup S$, $G' \leftarrow G' \setminus (S \cup N(S))$ i.e. G' is the induced subgraph
   on $V' \setminus (S \cup N(S))$ where $V' = V(G')$
6. Output I

End

There are two modifications to Luby's algorithm whose results are also computed. First is wake/sleep technique in which active vertices at the start of every round decides whether they will take part or not in this particular round. If not, every such vertex will not participate in this round hence will be surely not selected. The second modification is breaking the ties when two neighbors have both been selected in the same round. In the general version we break ties arbitrarily but in this case we break ties depending on the id of the vertices. The vertex with bigger id remains selected while the one with smaller id discards its earlier selection. Since id's for every vertex is unique this method is sure to break the tie. We also study the variation of number of rounds required by Luby's MIS with respect to the probability of a node getting selected which by default is $1/2d(v)$. We vary the probability from $1/4d(v)$ to $8/d(v)$.

Algorithm MIS-Luby-WakeorSleep$(G = (V, E))$

Begin

1. $I \leftarrow \phi; G' \leftarrow G$
2. while (G' is not empty) do
3. Every vertex first decides whether they will participate in this round or not,
   based on a coin toss probability.
4. Choose a random set of vertices $S \in G'$ by selecting each vertex v active
   in this round, independently with probability $1/2d(v)$
5. For every edge$(u, v) \in E(G')$ if both endpoints are in S;
   then remove the vertex with lower degree from S(Break ties arbitrarily)
6. $I \leftarrow I \cup S$, $G' \leftarrow G' \setminus (S \cup N(S))$ i.e. G' is the induced
   subgraph on $V' \setminus (S \cup N(S))$ where $V' = V(G')$
7. Output I

End

Algorithm MIS-Luby-ID$(G = (V, E))$

Begin

1. $I \leftarrow \phi; G' \leftarrow G$
2. while (G' is not empty) do
3. Choose a random set of vertices $S \in G'$ by selecting each active vertex v
   independently with probability $1/2d(v)$
4. For every edge$(u, v) \in E(G')$ if both endpoints are in S; then remove the vertex

with lower degree from S (If degree is equal, vertex with lower id is removed)

5. $I \leftarrow I \cup S, G' \leftarrow G' \setminus (S \cup N(S))$ i.e. G' is the induced subgraph on $V' \setminus (S \cup N(S))$ where $V' = V(G')$

6. Output I

End

5.1.2 Métivier's MIS

Métivier's MIS algorithm has the same bound on number of rounds required as of Luby's. In each round of Métivier's algorithm, the active vertices choose a random number. Subsequently, every vertex sends and receives the random number chosen by its neighbors in this round. Only the vertices which have the maximum number in the neighborhood remains selected. It is trivial to see no two vertices which are neighbors can be selected together in one round. Hence the set of nodes selected in any round will be independent. This set is included in the MIS and these vertices along with its neighbors are removed from the graph, hence are not considered in the remaining rounds. This goes on until there are no more vertices in the graph. The Independent set selected will be maximal, since every node in the graph is removed hence is either in the set or has a neighbor in the set in which case it cannot be included in the set, therefore the set will be a Maximal Independent set. It can be proved that Métivier's MIS is also expected to finish in $O(\log n)$ rounds. Below we present the pseudo code of the algorithm.

Algorithm MIS-Metivier($G = (V, E)$)

Begin

1. $I \leftarrow \phi; G' \leftarrow G$
2. while (G' is not empty) do
3. $S \leftarrow \phi$
3. Every vertex $v \in G'$ selects a random number r_v
4. If v has a local maximum, add v to S.
5. $I \leftarrow I \cup S, G' \leftarrow G' \setminus (S \cup N(S))$ i.e. G' is the induced subgraph on $V' \setminus (S \cup N(S))$ where $V' = V(G')$
6. Output I

End

5.1.3 Types of graphs

We run above algorithms on different types of graphs which we describe below.

Random graph : A random graph $G(n, p)$ is a graph formed by connecting nodes randomly. Each edge is included with a probability p independent of other edges.

Random regular graph : A random regular graph $G(d, n)$ is formed by connecting nodes randomly with the constraint that every node has exactly d neighbors directly connected by an edge.

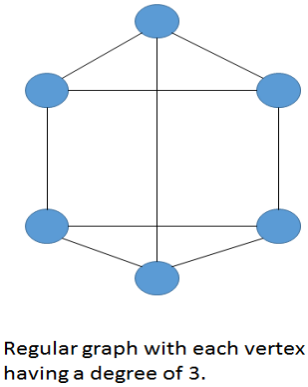

Figure 5.1 A regular graph with each vertex having a degree of 3

Bipartite graph : A bipartite graph (or bigraph) is a graph whose vertices can be divided into two disjoint sets U and V such that every edge connects a vertex in U to one in V; that is, U and V are each independent sets. Equivalently, a bipartite graph is a graph that does not contain any odd-length cycles

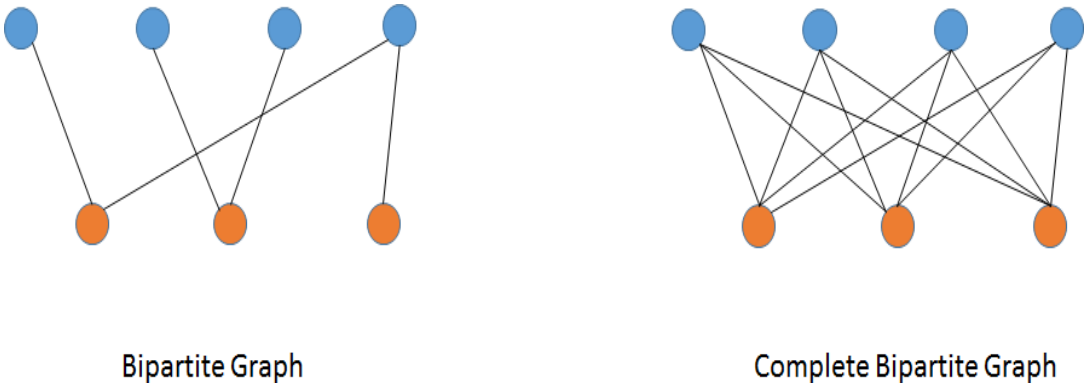

Figure 5.2 A bipartite graph with vertex sets denoted by red/blue colors

5.1.4 Networkx

We use *Networkx* [5] to generate graphs. NetworkX is a Python language software package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks. We compare Luby's and Métivier's algorithm on random, random regular as well as bipartite graphs of various sizes. Below we show the prototypes of functions of *Networkx* used to compute required graphs.

gnp_random_graph(n, p, seed=None, directed=False) : Returns a random graph $G(n, p)$ with n vertices. It chooses each of the possible edges with a probability p in the graph.

random_regular_graph(d, n, seed=None) : Returns a random regular graph $G(d, n)$ with n vertices each having d degree.

bipartite_random_graph(n, m, p, seed, directed) : Returns a random bipartite graph $G(n, m)$ with n nodes in its first bipartite set, m in the second set and p being the probability of creating an edge between two nodes in different set.

5.1.5 Results

Following plots and tables show the comparison between number of rounds required by Luby's, its variations' and Métivier's MIS.

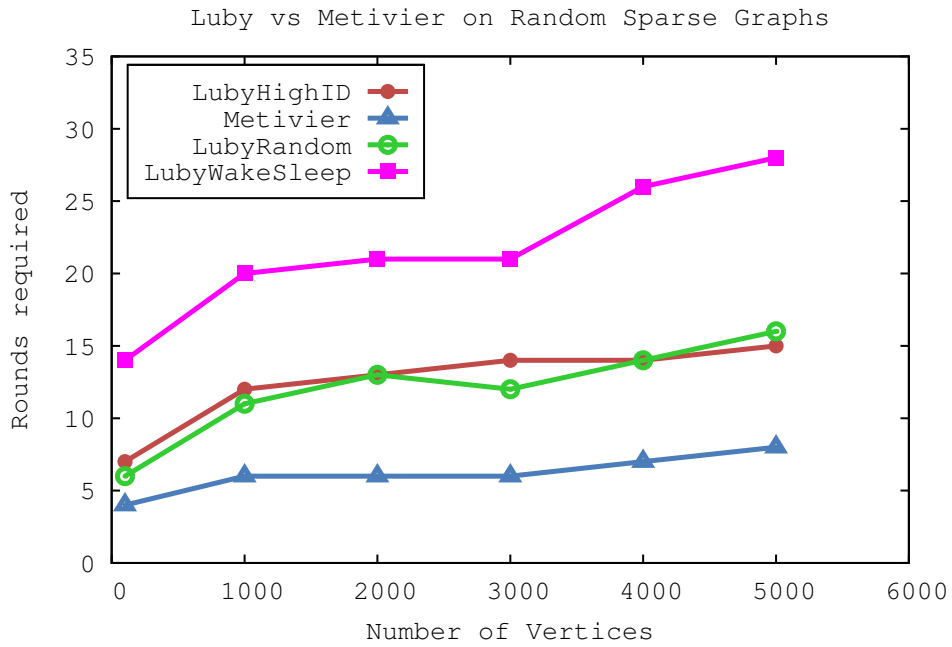

Figure 5.3 Luby vs Métivier on Sparse Random graphs, $p = 0.2$

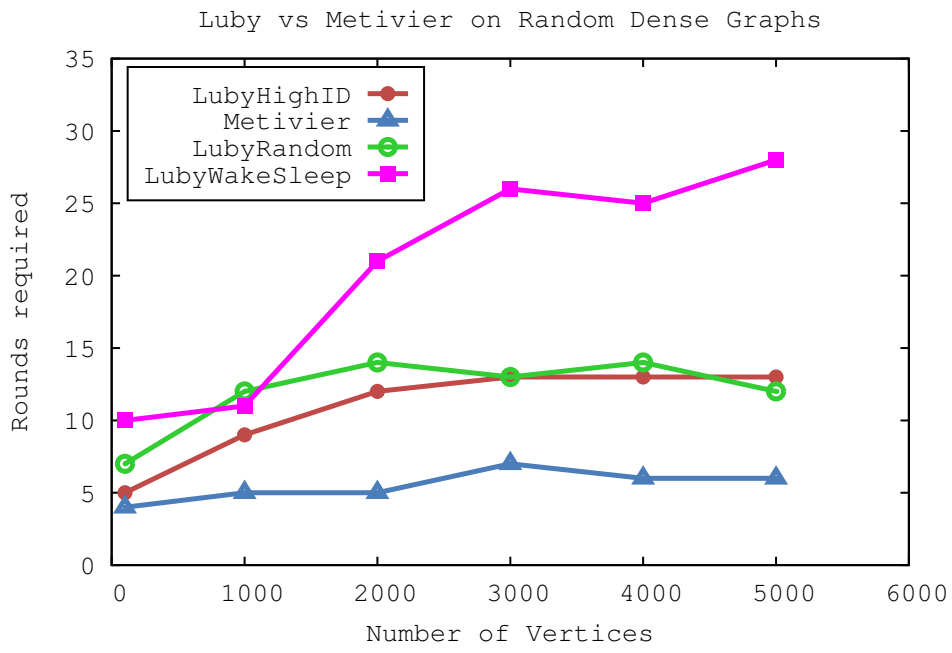

Figure 5.4 Luby vs Métivier on Dense Random graphs $p = 0.5$

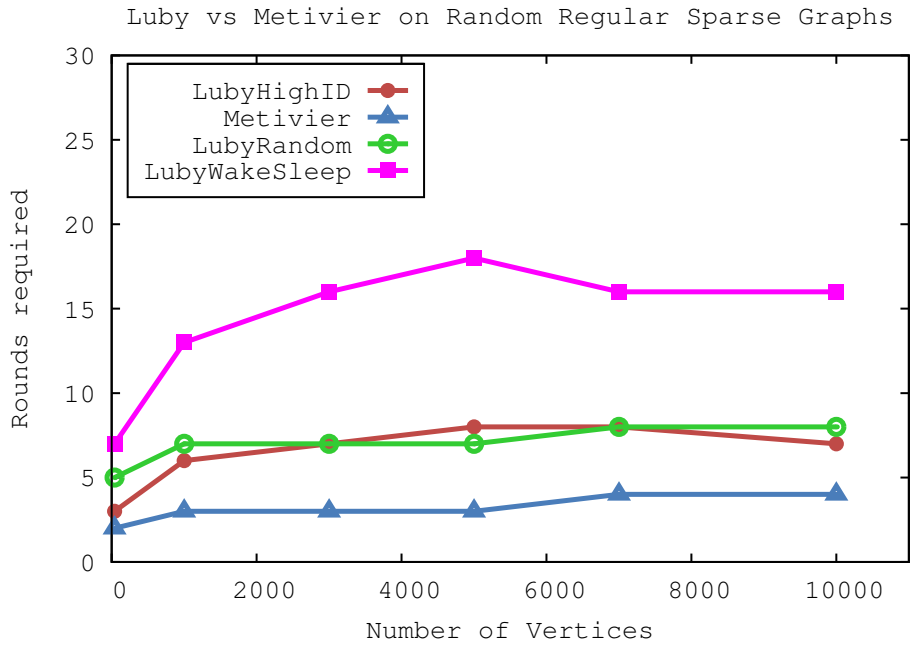

Figure 5.5 Luby vs Métivier on Sparse Random Regular graphs with $D = 3$

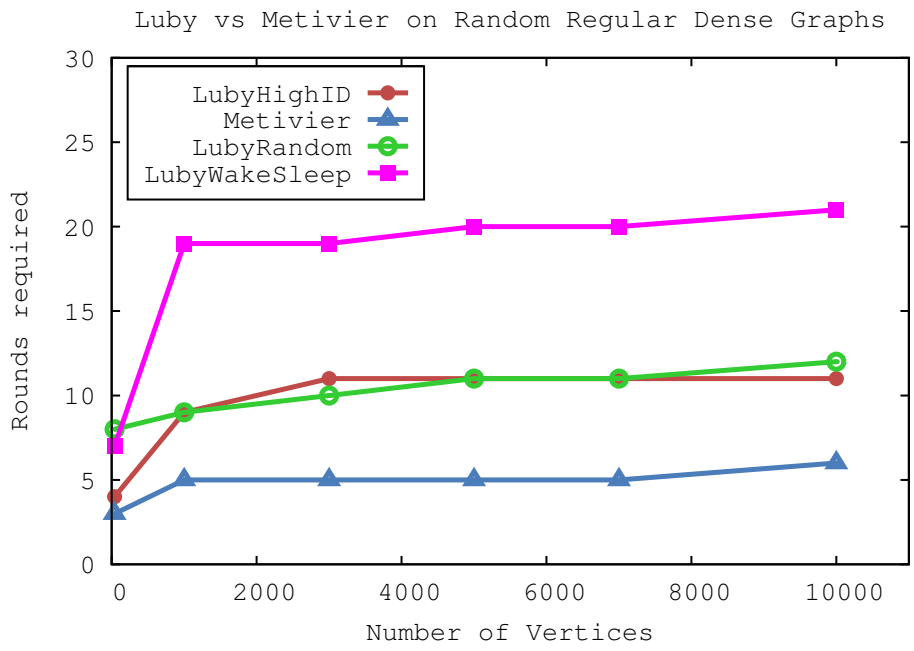

Figure 5.6 Luby vs Métivier Dense Random Regular graphs with $D = 30$

V1	V2	E	Rounds Luby	Luby's MIS size	Rounds Métivier	Métivier's MIS size
5	5	5	3	7	1	7
5	5	23	2	5	2	5
50	50	494	5	32	3	34
50	50	1996	4	50	3	50
500	500	49990	5	350	4	185
500	500	200024	5	250	3	500
5000	5000	4998206	7	2000	4	1500

Table 5.1: Luby vs Métivier on Bipartite Graphs with equal size vertex sets

V1	V2	E	Rounds Luby	Luby's MIS size	Rounds Métivier	Métivier's MIS size
10	2	5	2	8	2	9
10	2	16	2	9	2	10
100	20	417	3	100	2	80
100	20	1414	2	100	2	100
1000	200	40057	3	900	3	800
1000	200	119998	3	1000	2	1000
10000	200	600715	2	10000	2	10000
10000	200	1200343	2	3950	2	10000
10000	800	1601395	3	10000	2	8055
50000	800	4001437	2	50000	2	48084

Table 5.2: Luby vs Métivier on Bipartite Graphs with unequal number of vertices in vertex set

V1	V2	E	Rounds Random-Luby	Size Random-Luby	Rounds Wake/Sleep-Luby	Size Wake/Sleep-Luby
5	5	5	3	7	3	7
5	5	23	3	5	4	5
50	50	494	4	34	8	40
50	50	1996	3	50	5	50
500	500	49990	5	330	9	240
500	500	200024	4	400	5	500
5000	5000	4998206	6	1900	9	2200

Table 5.3: Luby variations(Random, WakeorSleep) on Bipartite Graphs of equal size vertex sets

V1	V2	E	Rounds Random-Luby	Size Random-Luby	Rounds Wake/Sleep-Luby	Size Wake/Sleep-Luby
10	2	5	2	9	4	10
10	2	16	2	10	2	10
100	20	417	3	95	3	100
100	20	1414	3	90	3	100
1000	200	40057	3	1000	3	850
1000	200	119998	3	1000	3	1000
10000	200	600715	2	10000	2	10000
10000	200	1200343	2	10000	2	10000
10000	800	1601395	2	10000	3	10000
50000	800	4001437	2	50000	2	50000

Table 5.4: Luby variations(Random, WakeorSleep) on Bipartite Graphs of unequal size vertex sets

Below we present results from varying the probability of a node getting selected in each round in Luby's MIS algorithm. We vary the probability from $1/4d(v)$ to $8/d(v)$.

V1	V2	E	1/4d(v)	1/3d(v)	1/2d(v)	2/3d(v)	1/d(v)	2/d(v)	4/d(v)	8/d(v)
5	5	5	3	4	3	2	2	2	2	2
5	5	23	4	4	3	3	2	2	2	2
50	50	494	9	6	6	5	3	3	3	3
50	50	1996	5	7	4	3	2	3	2	2
500	500	49990	11	8	8	7	5	4	4	3
500	500	200024	7	4	4	4	4	3	2	3
5000	5000	4998206	7	7	9	6	4	4	3	3

Table 5.5: Luby with different probabilities on Bipartite Graphs of equal size vertex sets

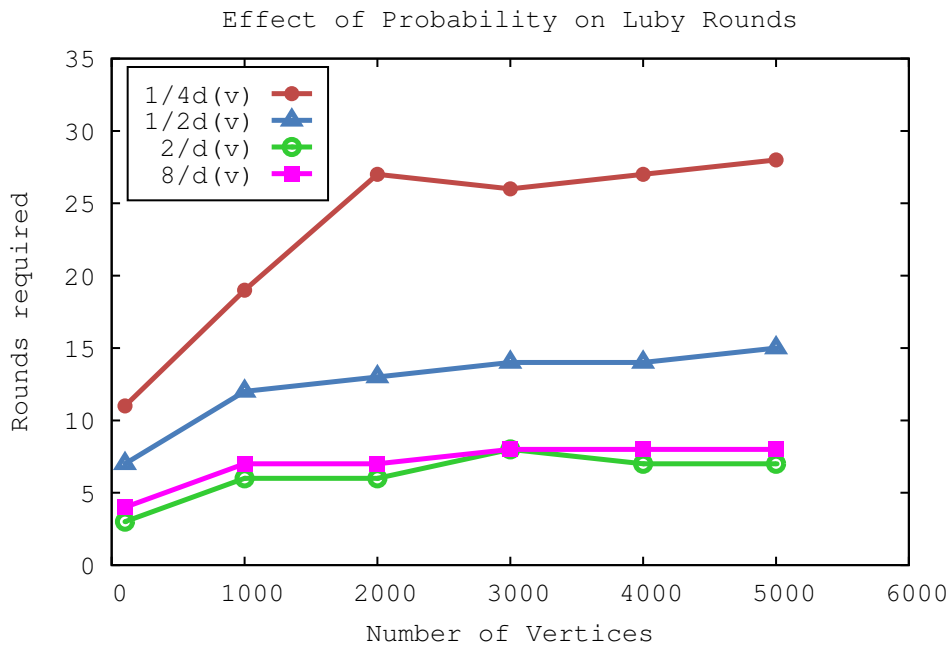

Figure 5.7 Luby with variable probability on Sparse Random graphs, $p = 0.2$

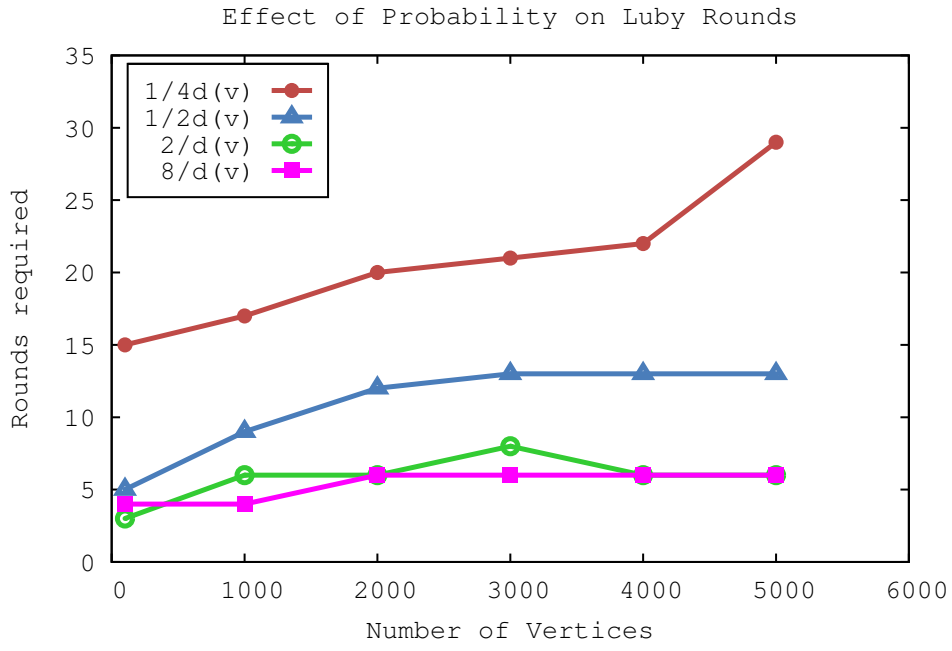

Figure 5.8 Luby with variabe probability on Dense Random graphs, $p = 0.5$

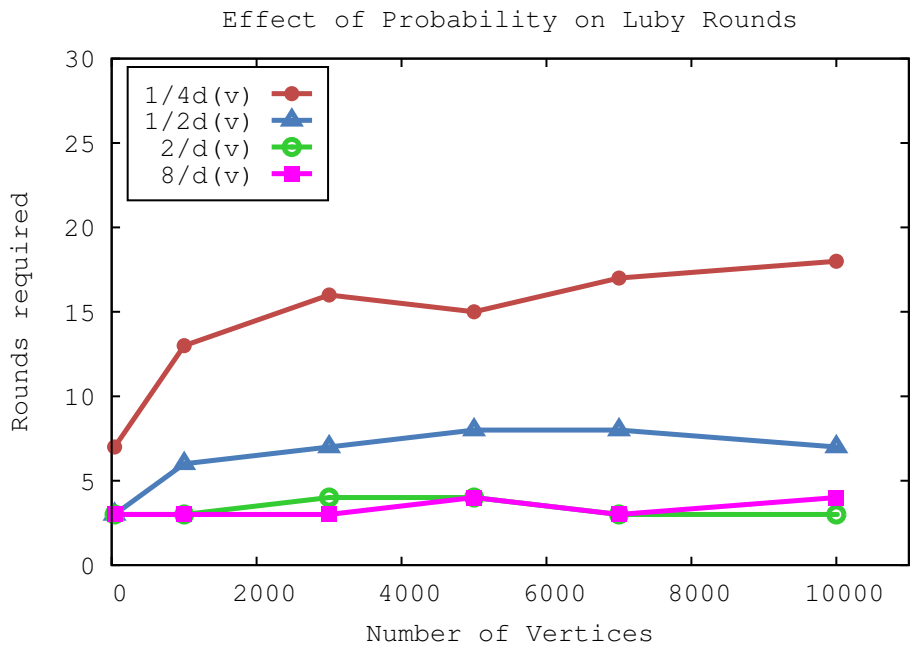

Figure 5.9 Luby with variabe probability on Sparse Random Regular graphs, $D = 3$

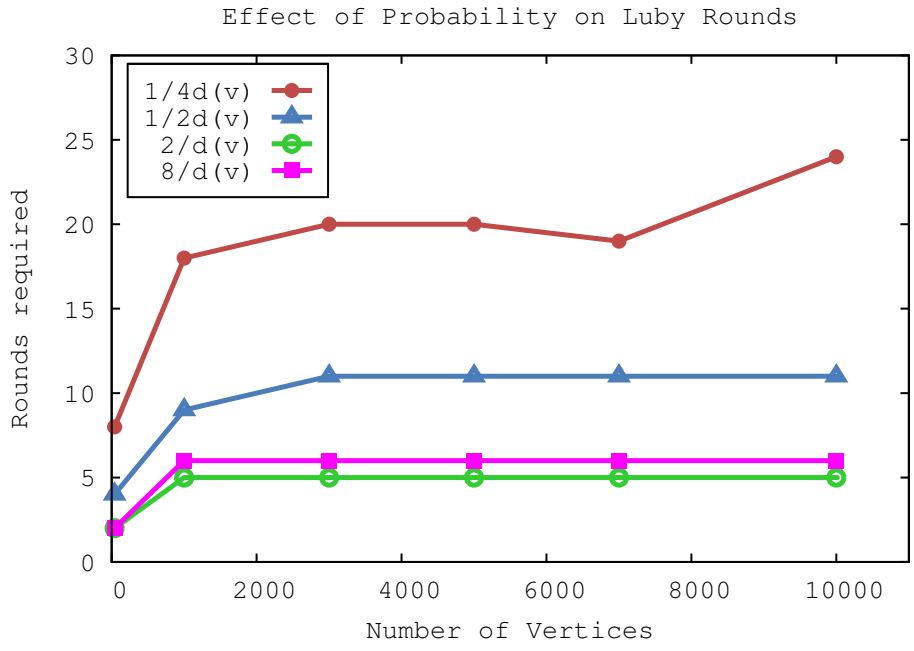

Figure 5.10 Luby with variabe probability on Dense Random Regular graphs, $D = 30$

V1	V2	E	1/4d(v)	1/3d(v)	1/2d(v)	2/3d(v)	1/d(v)	2/d(v)	4/d(v)	8/d(v)
10	2	5	5	4	2	3	1	2	2	2
10	2	16	3	2	2	2	2	1	1	1
100	20	417	4	3	3	3	2	2	1	2
100	20	1414	4	3	3	2	2	2	2	2
1000	200	40057	4	3	3	3	2	2	2	2
1000	200	119998	4	3	3	3	2	2	2	2
10000	200	600715	2	2	2	2	2	2	2	2
10000	200	1200343	2	2	2	2	2	2	2	2
10000	800	1601395	3	3	3	3	2	2	2	2
50000	800	4001437	2	2	2	2	2	2	2	2

Table 5.6: Luby with different probabilities on Bipartite Graphs of equal size vertex sets

5.1.6 Analysis of Luby's MIS

Let the probability be $1/cd(v)$ for a node v with degree $d(v)$ to get selected in a round. Let us define a vertex v to be a *good* vertex if at least $1/3^{rd}$ of its neighbors have degree less than $d(v)$. We also define an edge to be a *good* edge if at least one endpoint of it is a good vertex.

Every good vertex has a lot of low degree vertices whose degree is bounded by $d(v)$. So with good probability at least one of these vertices will make it to the Independent Set thereby deleting good vertex and therefore good edges.

Lemma 5.1.1 *For every good vertex v with $d(v) > 0$, the probability that some neighbor w of v gets marked is at least $1 - e^{\frac{-1}{3c}}$.*

Proof. Consider one such neighbor w. Probability of w getting marked in current round = $1/cd(w)$. Since v is a good vertex, at least $d(v)/3$ neighbors have degree at most $d(v)$. Let w be such a neighbor.
Pr(w is marked) $= 1/cd(w) \geq 1/cd(v)$
Pr(no neighbor of v is marked) $\leq (1 - 1/cd(v))^{d(v)/3} = e^{\frac{-1}{3c}}$. □

Lemma 5.1.2 *If a vertex w is marked, then Pr(w is in MIS) $\geq 1/c$.*

Proof. If w is marked, then it is not in MIS only if some high degree neighbor of w is also marked. Each such high degree neighbors of w is marked with probability at most $1/2d(w)$.

Pr(w is in MIS) $= 1-$Pr(w is not in MIS)
$= 1 - Pr(\exists u \in N(w), d(u) \geq d(w), u\, is\, marked)$
$= 1 - |u \in N(w), d(u) \geq d(w)| * 1/cd(w)$
$\geq 1 - |u \in N(w)| * 1/cd(w) = 1 - (d(w) * 1/cd(w)) = 1/c$

□

Lemma 5.1.3 *If v is a good vertex, then Pr(v is deleted) $\geq (1 - e^{\frac{-1}{3c}})/c$.*

Proof. Combining above results proves this. □

Lemma 5.1.4 *At least half the edges are good.*

Proof. For every bad edge e, associate a pair of edges via a function $f : E_B \rightarrow \binom{E}{2}$ such that for any two distinct bad edges e_1 and e_2, $f(e_1) \cap f(e_2) = \phi$. This completes the proof since only $|E|/2$ such pairs exist. The function f is defined as below.

For each edge$(u, v) \in E$,orient it towards the vertex of higher degree. Consider a bad edge $e(u, v)$ oriented towards v. Since v is bad, the out-degree of v is at least twice its in-degree. So, there is a way to pick a pair of edges for every bad edge. □

Lemma 5.1.5 *In each iteration on an expectation a constant fraction of edges are deleted.*

Proof. Half the edges are good as proved before, and a good edge is deleted with probability at least $(1 - e^{\frac{-1}{3c}})/c$. So, on expectation $O(\log m) = O(\log n)$ rounds suffice for the luby algorithm to finish. This can be extended to show that this happens with high probability. □

Now we try to understand the variation of number of rounds required to the variabe c. Notice as we decrease c thereby increasing the probability of a vertex to get selected in a round, the minimum probability of a good edge getting deleted increases. This intuition supports the decrease in number of rounds required by Luby's MIS algorithm with the decrease in c as shown in Table 5.5, Table 5.6, Figure 5.7, Figure 5.8, Figure 5.9 and Figure 5.10. This happens pretty regularly till we increase the probability to $2/d(v)$, then it starts to show a constant or rather in some cases opposite behavior that is the number of rounds starts increasing with the decrease in c.

5.1.7 Conclusions

The calculations show that Métivier's MIS algorithm performs better than Luby's for almost every graph. Though the size of MIS is nearly the same for both. For most of the graph Métivier performs nearly twice better in terms of number of rounds required to compute the MIS. Also the more sparse the graph the better Métivier performs over Luby.

It can also be observed that both the Luby with random tie breaking and Luby with high vertex id tie breaking performs similar for all graphs and considerably better than Luby with wake/sleep technique.

We can also notice that the number of rounds required by Luby's MIS varies largely with the probability used. It gradually decreases with increasing probability for many cases. In some cases though, supporting intuition the number of rounds first decreases and then increases as we increase the probability thereby suggesting that too many nodes getting selected hindering the overall decrease in number of rounds required by the algorithm to finish.

Chapter 6

Conclusions

In this thesis, we presented new faster algorithms for computing ruling sets in distributed synchronous message passing model for general as well as specific graphs like high girth and bounded arboricity graphs. We first presented our algorithm to compute a 2-ruling set for general graphs which requires $\frac{\log \Delta}{\log \log n} + e^{O(\sqrt{\log \log n})}$round with high probability. This result is an improvement over the $O\left(\frac{\log \Delta}{(\log n)^{\epsilon}} + (\log n)^{1/2+\epsilon}\right)$ round algorithm from [6] whenever $\Delta \in O(2^{\sqrt{\log n}})$. We then extend this technique to compute a t-ruling set for both general graphs as well as bounded arboricity graphs. Our algorithm can compute a $(t+1)$-ruling set for any $t \geq 1$ in graphs with arboricity $a = O(\log \log n)$ can be computed in $O(t \log^{1/t} \Delta + t^2 (\log \log n)^{1/t} + (\log \log n)^2)$ rounds. Therefore, a $\log \log \Delta$-ruling set for bounded arboricity graphs can be computed in $O((\log \log n)^2)$ rounds. Similarly, a $(t+1)$-ruling set for a general graph can be computed in $O(t \log^{1/t} \Delta + t^2 (\log \log n)^{1/t} + e^{O(\sqrt{\log \log n})})$ rounds. Also, a t-ruling set for graphs of girth > 6 can be computed in $O(t \log^{1/t} \Delta + t^2 (\log \log n)^{1/t} + e^{O(\sqrt{\log \log n})})$ rounds. This shows that computing a t-ruling set for a small t can be comparably significantly faster than computing a MIS in the same graph, this encourages the approach for computing a MIS by first finding a t-ruling set for some t and then extending it to cover the complete graph.

We then combine our algorithm to compute a 2-ruling set in general graph with Theorem 6.2 [2] to present our algorithm which can compute a 2-ruling set in $O(\sqrt{\frac{\log n}{\log \log n}})$ rounds with high probability in any graph with arboricity $a = 2^{O(\sqrt{\log n})}$. This result is significantly faster than any of the MIS algorithms for such range of arboricity.

At last we compare the run time of Luby's [13] and Métivier's [14] MIS algorithms, two fundamental algorithms for computing an MIS in general graphs The calculations show that Métivier's MIS algorithm performs better than Luby's for almost every graph. Though the size of MIS is nearly the same for both. For most of the graph Metevier performs nearly twice better in terms of number of rounds required to compute the MIS. Also the more sparse the graph the better Metevier performs over Luby.

It can also be observed that both the Luby with random tie breaking and Luby with high vertex id tie breaking performs similar for all graphs and considerably better than Luby with wake/sleep technique.

We can also notice that the number of rounds required by Luby's MIS varies largely with the probability used. It gradually decreases with increasing probability for many cases. In some cases though,

supporting intuition the number of rounds first decreases and then increases as we increase the probability thereby suggesting that too many nodes getting selected hindering the overall decrease in number of rounds required by the algorithm to finish.

Related Publications

Bisht Tushar, Kishore Kothapalli, and Sriram Pemmaraju. ”Brief announcement: Super-fast t-ruling sets.” Proceedings of the 2014 ACM symposium on Principles of distributed computing. ACM, 2014.

Bisht Tushar, and Kishore Kothapalli. ”An empirical study of two MIS algorithms.” Advanced Computing, Networking and Security (ADCONS), 2013 2nd International Conference on. IEEE, 2013.

Bibliography

[1] N. Alon, L. Babai, and A. Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *J. Algorithms*, 7(4):567–583, 1986.

[2] L. Barenboim, M. Elkin, S. Pettie, and J. Schneider. Fast distributed algorithms for maximal matching and maximal independent set. In *Proc. of IEEE FOCS*, 2012.

[3] B. Gfeller and E. Vicari. A randomized distributed algorithm for the maximal independent set problem in growth-bounded graphs. In *Proc. of ACM PODC*, pages 53–60, 2007.

[4] A. Goldberg, S. Plotkin, and G. Shannon. Parallel symmetry breaking in sparse graphs. *SIAM J. Discrete Math.*, 1:434–446, 1989.

[5] A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pages 11–15, Pasadena, CA USA, Aug. 2008.

[6] K. Kothapalli and S. V. Pemmaraju. Super-fast 3-ruling sets. In *Proc. of FSTTCS*, 2012.

[7] K. Kothapalli, C. Scheideler, M. Onus, and C. Schindelhauer. Distributed coloring in $O(\sqrt{\log n})$ bit rounds. In *In International Parallel and Distributed Processing Symposium, (IPDPS)*, 2006.

[8] F. Kuhn, T. Moscibroda, T. Nieberg, and R. Wattenhofer. Fast deterministic distributed maximal independent set computation on growth-bounded graphs. In P. Fraigniaud, editor, *Distributed Computing*, volume 3724 of *Lecture Notes in Computer Science*, pages 273–283, Berlin, Germany, November 2005. Springer-Verlag.

[9] F. Kuhn, T. Moscibroda, and R. Wattenhofer. What cannot be computed locally! In *Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, PODC '04, pages 300–309, New York, NY, USA, 2004. ACM.

[10] F. Kuhn, T. Moscibroda, and R. Wattenhofer. Local computation: Lower and upper bounds. *CoRR*, abs/1011.5470, 2010.

[11] C. Lenzen and R. Wattenhofer. Minimum dominating set approximation in graphs of bounded arboricity. In *International Symposium on Distributed Computing (DISC)*, pages 510–524, 2010.

[12] N. Linial. Locality in distributed graph algorithms. *SIAM Journal of Computing*, 21:193–201, 1992.

[13] M. Luby. A simple parallel algorithm for the maximal independent set. *SIAM Journal on Computing*, 15:1036–1053, 1986.

[14] Y. Métivier, J. Robson, N. Saheb-Djahromi, and A. Zemmari. An optimal bit complexity randomised distributed mis algorithm. In *Proc. SIROCCO*, pages 323–337, 2009.

[15] C. Nash-Williams. Decompositions of finite graphs into forests. *J. London Math*, 39(12), 1964.

[16] A. Panconesi and A. Srinivasan. On the complexity of distributed network decomposition. *J. Algorithms*, 20(2):356–374, 1996.

[17] J. Schneider and R. Wattenhofer. A log-star distributed maximal independent set algorithm for growth-bounded graphs. In *PODC*, pages 35–44, 2008.