An Empirical Study of Two MIS Algorithms

Tushar Bisht and Kishore Kothapalli
International Institute of Information Technology, Hyderabad
Hyderabad, Andhra Pradesh, India 500 032.
Email: tushar.bisht@research.iiit.ac.in, kkishore@iiit.ac.in

***Abstract*—Distributed algorithms for computing a maximal independent set of a graph have been very popular in the research community. Starting with the algorithm of Luby dating back to 25 years, several researchers have focussed on distributed MIS algorithms for general graphs with little success. Only recently, Métevier et al. have proposed an algorithm for MIS in the distributed setting with a round complexity that matches Luby's algorithm asymptotically.**

However, very little is known about the relative performance of the algorithm of Luby and Métevier in practice. Such a study can throw light on some aspects of the algorithms that are not brought out by a theoretical analysis. In this paper, we compare implementations of the algorithm of Luby and Métevier and study their behavior on a class of random graphs and bipartite graphs. Further, we study how varying the probability that a node wishes to join the MIS in the case of Luby's algorithm affects the number of rounds required.

I. Introduction

A Maximal Independent Set (MIS) in a graph $G = (V, E)$ is a subset $S \subseteq V$ such that S is maximal and nodes in S are mutual non-neighbors. Computing an MIS is one of the fundamental problems in distributed computing and has attracted the attention of a vast amount of research [3], [1], [4], [5]. The best known algorithm so far for general graphs are the algorithm of Luby [3] that runs in $O(\log n)$ probabilistic rounds where n is the number of nodes in the graph. Only recently, Métivier et al. [4] proposed another algorithm that too has an $O(\log n)$ round complexity.

Despite the immense research interest in distributed algorithms for MIS, very little is known on how these algorithms work in practice. For instance, the algorithms of both Luby [3] and Métivier et al. [4] have require $O(\log n)$ distributed rounds asymptotically. They are to date the fastest known algorithms for computing an MIS in a general graph. The approach of these two algorithms, though both are randomized, is slightly different. Métivier's algorithm considers a node to be in MIS if the random choice of this node is a local maxima. The algorithm of Luby however considers a node to be in MIS as follows. Each node opts to be in MIS with a probability inversely proportional to its degree. In case of ties, i.e., edges with both end points opting to be in the MIS, then the algorithm prefers higher degree vertices to be in the MIS. Both algorithms proceed in a similar way by removing nodes selected in one round along with their neighbors.

An experimental study of the relative speed of algorithms can offer several lessons. Especially in the context of graph algorithms, one can study the behaviour of the algorithms on various graph classes. It is possible that an algorithm is much faster on some graph classes compared to another algorithm. Similarly, it is sometimes the case that distributed graph algorithms can finish in fewer rounds than that given out by the analysis. Finally, in the case of MIS algorithms, maximality does not impose any constraints on the size of the MIS produced by a particular algorithm. An experimental study can highlight the difference in the sizes of the MIS produced by different algorithms.

In this paper, we focus on the algorithms of Luby [3] and the algorithm of Métivier et al. [4] to find an MIS in a general graph. We consider parameters such as the number of rounds needed by the algorithms and the size of the MIS produced. We also consider several graph classes ranging from random graphs, bipartite graphs, and regular graphs. Further, we study the analytical and experimental effect of modifying the probability a node chooses to join the MIS in Luby's algorithm [3]. Our study in general indicates that the algorithm of Métivier outperforms the algorithm of Luby by a factor of 2. It is surprising however that the size of the MIS produced does not differ significantly in both algorithms.

The rest of the paper is organized as follows. In Section II, we describe the algorithms of Luby [3] and Métivier et al. [4]. Section III discusses our experiments and their results. Section IV describes a change to Luby's algorithm and its effect on the runtime. The paper ends with some concluding remarks in Section V.

II. Preliminaries

Below we describe both the algorithms and give pseudo code for them.

A. 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 $d(v)$ denotes the current degree of v, that is the number of `active` neighbors v has. If two neighbors v and w get 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 impact is also studied. One such modification is to use a tie-breaking rule when two nodes are selected in the same round. We study two versions in this modification.

1) **LubyRandom:** In version LubyRandom, we use a random tie breaking rule.
2) **LubyHighID:** In version LubyHighID, we select the node with a higher identifier.

In a second modification we introduce a technique where nodes that are still `awake` choose with a certain probability to participate in a given round. This probability is varied from $1/2d(v)$ to $1/8d(v)$. This version is termed as **Luby-WakeSleep**.

B. 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 is either in the set or has a neighbor in the 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 $V' \setminus (S \cup N(S))$ ; $V' = V(G')$
6.    Output $I$
End
```

C. Networkx

We use *Networkx* [2] 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.

III. Experiments and Results

We implemented the algorithms of Luby and Métivier et al. in ANSI C. The program is written as a sequential program. The actions of each node is simulated by a sequential execution one after the another. This does not affect the behavior of the algorithms and since the measure of interest is the number of rounds, this does not affect the correctness of the experiments. For generating random numbers, we used the output of the SHA algorithm [1] as a random number between 0 and 1, both inclusive. The output of SHA is known to possess better uniformness properties compared to the standard `rand` function in ANSI C.

While presenting the results of the experiments we performed each experiment multiple times depending on the size of the instance and averaged the results. This gives us enough confidence on the validity of the empirical analysis. Since we can compare the number of rounds independent of the machine speed and characteristics, we do not report on the exact details of the machine.

A. Results

In our experiments, we use a variety of graph classes including random graphs, bipartite graphs, and regular graphs. The results of our study on these graphs are presented in the following subsections.

1) Random Graphs: We first show the number of rounds taken by the two algorithms on sparse random graphs generated with $p = 0.2$ in Figure 1. The plot labelled **LubyHighID** represents Luby's algorithm with a tie breaking rule prefers the larger vertex ID. As can be seen, this does not have much impact on the number of rounds compared to the **LubyRandom** version. It can also be noticed from Figure 1 that Métivier's algorithm runs faster than all variants of Luby's algorithm. The difference in the two algorithms is roughly a factor of 2.

[1]see http://csrc.nist.gov/groups/STM/cavp/documents/shs/sha256-384-512.pdf

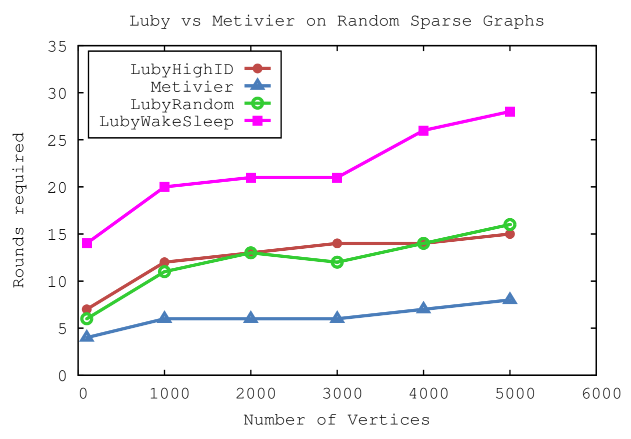


Fig. 1. Luby vs Métivier on Sparse Random graphs, $p = 0.2$.

A similar effect can be seen on random graphs generated with a slightly higher probability of 0.5. This result is shown in Figure 2. As can be seen, the version with a wakeup probability takes more rounds. It therefore seems that having nodes forgo a round probabilistically is not helping break symmetry.

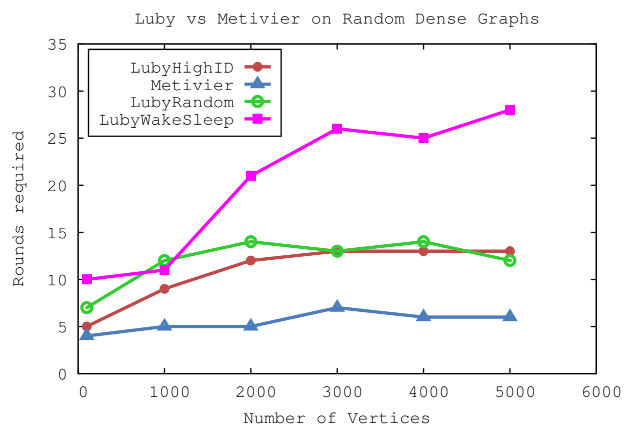


Fig. 2. Luby vs Métivier on Dense Random graphs $p = 0.5$.

2) Regular Graphs: Another class of graphs that we experimented with is random graphs with a regular degree. This helps us study if all nodes having the same probability of being in an MIS has any effect on the number of rounds required by the algorithms of Luby and Métivier. We experimented with both sparse and dense random regular graphs. The results of our experiments are shown in Figure 3 for sparse random regular graphs and in Figure 4 for dense random regular graphs. As can be seen in this case too, a random tie breaking rule works better for implementing Luby's algorithm. Further, having a wakeup probability is not helpful. Finally, the algorithm of Métivier is faster than Luby's algorithm by a factor of 2. Moreover, sparse graphs are easy to break symmetry in compared to dense graphs.

3) Bipartite Graphs: Another class of graphs that we used in our experiments are bipartite graphs. We generated bipartite graphs with equi-sized partitions and also with unequal-sized partitions. The results of this experiment are shown in Figure 5. It is noticed that both algorithms behave in a near-similar

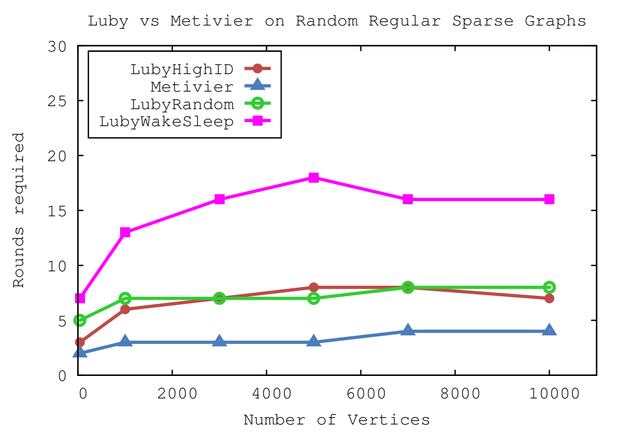


Fig. 3. Luby vs Métivier on Sparse Random Regular graphs with degree 3.

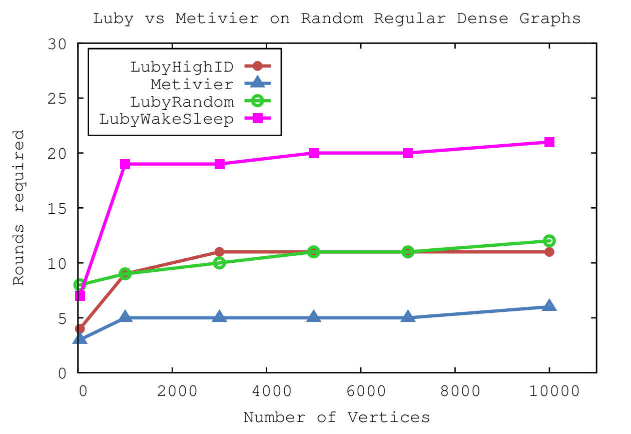


Fig. 4. Luby vs Métivier Dense Random Regular graphs with degree 30.

fashion in the case of bipartite graphs. Perhaps, this can be explained by the fact that bipartite graphs have a large MIS in general and this is helping break symmetry quickly.

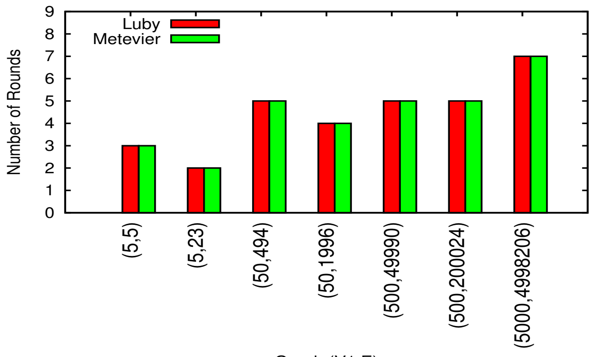
IV. Changing the Selection Probability in Luby's Algorithm

Let the probability for a node v with degree $d(v)$ to get selected in a round be $1/cd(v)$ for a constant c. 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.

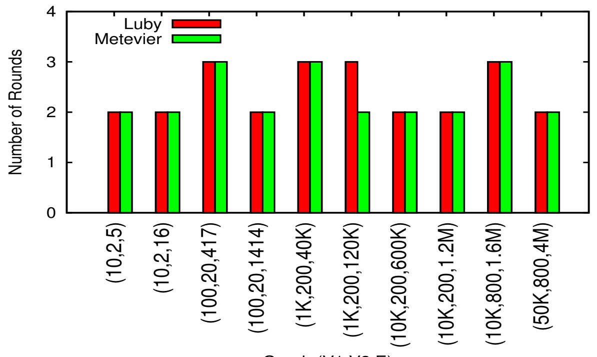
Every good vertex has a lot of low degree vertices whose degree is bounded by $d(v)$. So with good probability at least one of these vertices will make it to the independent Set thereby deleting good vertex and therefore good edges.

Lemma 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) $= 1/cd(w) \geq 1/cd(v)$



(a) Bipartite Graphs with Equi-sized Partitions



(b) Bipartite Graphs with Unequal-sized Partitions

Fig. 5. Number of rounds taken by the algorithms of Luby and Métevier on bipartite graphs. In the figure on the left, the X-axis represents the size of each partition and the total number of edges in the graph. In the figure on the right, the X-axis represents the sizes of the partitions and the total number of edges in the graph.

Pr(no neighbor of v is marked) $\leq (1 - 1/cd(v))^{d(v)/3} = e^{\frac{-1}{3c}}$. ■

Lemma 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)
$= 1 - Pr(\exists u \in N(w), d(u) \geq d(w), u\, is\, marked)$
$= 1 - |u \in N(w), d(u) \geq d(w)| * 1/cd(w)$
$\geq 1 - |u \in N(w)| \cdot 1/cd(w) = 1 - (d(w) \cdot 1/cd(w)) = 1/c$ ■

Lemma 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 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: 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 Luby's 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 variable c. Notice as we decrease c thereby

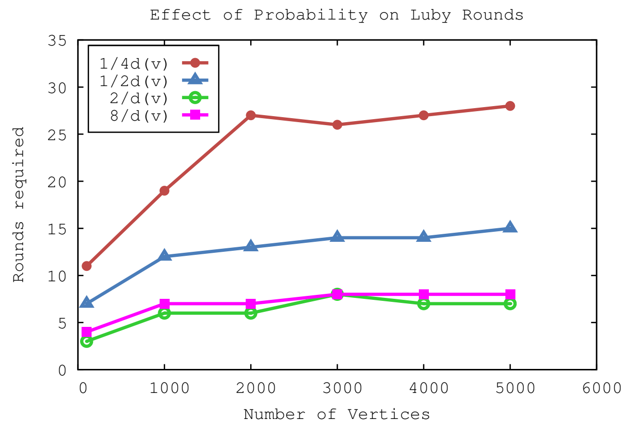


Fig. 6. Luby with variable probability on Sparse Random graphs, $p = 0.2$.

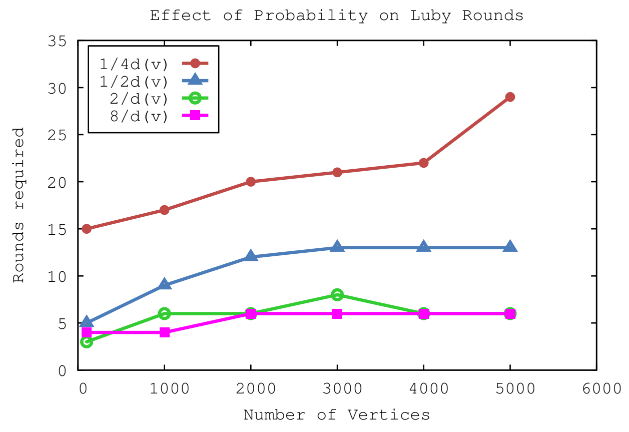


Fig. 7. Luby with variable probability on Dense Random graphs, $p = 0.5$.

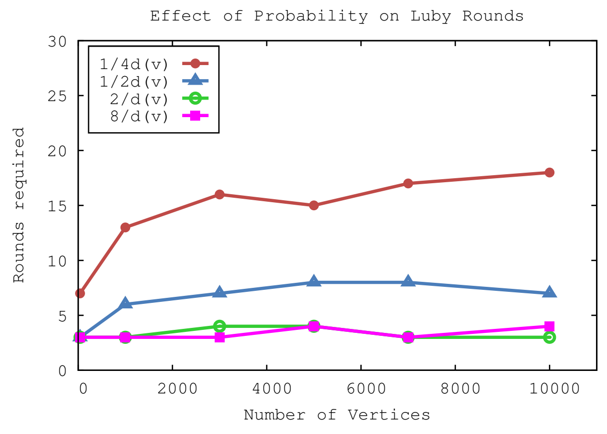


Fig. 8. Luby with variable probability on Sparse Random Regular graphs of degree 3.

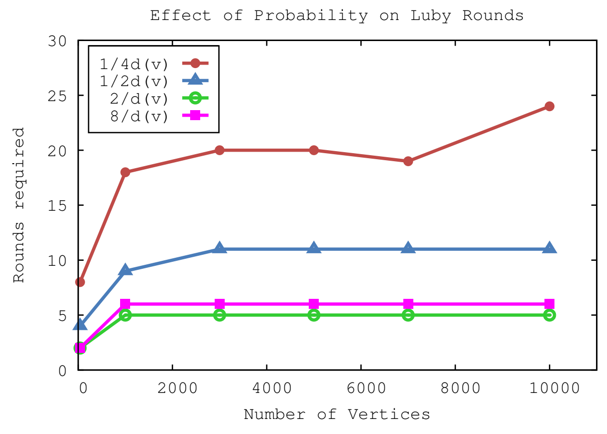


Fig. 9. Luby with variable probability on Dense Random Regular graphs of degree 30.

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 Figure 6, Figure 7, Figure 8, and Figure 9. 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.

V. Conclusions

Our experiments show that Métivier's MIS algorithm performs better than Luby's for almost every graph. We also noticed in our experiments that the size of MIS produced is nearly the same for both the algorithms. 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 our intuition, the number of rounds first decreases and then increases as we increase the probability thereby suggesting that too many nodes getting selected hinder the decrease in number of rounds required by the algorithm to finish.

References

[1] L. Barenboim and M. Elkin. Sublogarithmic distributed MIS algorithm for sparse graphs using nash-williams decomposition. pages 25–34, 2008.

[2] Aric A. Hagberg, Daniel A. Schult, and Pieter 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, August 2008.

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

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

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