Distributed Code Assignment in Multihop Radio Networks: Object-Oriented Software Simulations

Roberto Battiti[*] Alan A. Bertossi[*†] Mauro Brunato[*†‡]

Abstract

Code Assignment is one of the central problems in the management of mobile wireless networks, where a very precious resource (radio bandwidth) needs to be carefully shared among network nodes. As this problem has been shown to be NP-hard in every real world instance, no practical exact solution can be provided, and heuristic algorithms are being developed whose actual performance can hardly be calculated by theorical means; hence the development of heuristic schemes must always be accompanied by appropriate simulation software.

In this paper we present a new distributed algorithm for code assignment in a multihop radio network. The algorithm is based on the Saturation-Degree coloring scheme proposed in [1], which has proved better than earlier attempts at solving the problem. A crucial parameter of the proposed algorithm is the depth of the neighborhood that must be considered when a local decision must be taken. By tuning this parameter one can obtain a tradeoff between the quality of the solution and the number of parallel passes and exchanged messages.

We analyze the results of our C++ simulation programs where the proposed algorithm is compared with its sequential version and with competitive schemes proposed in the literature. Furthermore, to help analyzing the behavior of our distributed technique, we provide a graphical simulation environment written in Java.

Keywords: Saturation Degree, Distributed Computing, Wireless Networks

1 Introduction

The Saturation Degree coloring method has proved to be effective to approximate the solutions of many graph coloring problems. In this paper we design and analyze a distributed version of the Saturation Degree heuristic. The problem considered is that of minimizing the codes needed to eliminate primary and/or hidden collisions in a Wireless Network with transmitter-oriented code assignment (TOCA) protocols [11, 8], where the transmitters are code-agile (i.e. they can modify their transmission code within a suitable range).

Previous work concerning the use of the Saturation Degree heuristic in Code Assignment problems can be found in [1], where it is applied in its original, sequential form; in Section sec:satdeg we give a short resume of this technique and show its good behavior when compared with other algorithms.

The paper is organized as follows. In Section 2 a brief summary of the problem and of previous works is reported, basic definitions are given. and the code assignment problem is formulated. In Section 3 a fast overview of the sequential Saturation Degree algorithm for graph coloring is given. Section 4 is the central topic of our paper, describing how the Saturation Degree algorithm can be implemented in a distributed environment, where every node is able to compute its own code. Finally, Section 5 reports some experimental results, showing that the performance does not deteriorate when the system is distributed.

[*]Università di Trento, Dipartimento di Matematica, Via Sommarive 14, I-38050 Pantè di Povo — Italy
[†]Email: `bertossi|battiti|brunato@science.unitn.it`
[‡]Corresponding author.

2 Multihop radio networks

In a wireless network, computers (“hosts”, “nodes”) are equipped with radio transceivers to broadcast outgoing packets and to listen to incoming ones. Radio transmitters have a limited range; we shall consider *multihop* networks, where the communication between two nodes happens by having packets received and later retransmitted by intermediate stations, if necessary, before reaching their final destination, with a “store and forward” scheme.

Unconstrained transmission in broadcast media may lead to collisions of two or more packet receptions, caused by frequency and time overlap, resulting in damaged useless packets at the destination and therefore in increased delays and bandwidth usage caused by retransmissions. Interferences and collisions can be *direct* (or primary), when they are due to the transmission of stations which can hear each other, and *hidden* (or secondary), when stations outside the hearing range of each other transmit to the same receiving stations.

Multiple access protocols can be used to totally avoid collisions [12, 13] by dividing the available radio spectrum into channels, i.e. frequency bands, time slices or orthogonal codes. These protocols require both transmitters and receivers to be *code-agile*, i.e. able to communicate over a multitude of channels. These channels share the fixed spectrum slice allocated to the network in the design stage. Thus, their number must not exceed a given bound, and their use has to be minimized.

2.1 The code assignment problem

A wireless network with n stations can be modeled as an undirected graph $G = (V, E)$, where the set of vertices (or nodes) $V = \{1, ..., n\}$ represents the set of stations, and the set of edges $E \subseteq V \times V$ the mutual hearing between pairs of stations. Thus, the graph G represents the physical network topology. A *path* between the vertices i and j is a sequence $i = v_1, v_2, ..., v_h = j$ of h vertices in V such that $(v_k, v_k + 1) \in E$ for $k = 1, 2, ..., h - 1$; its length is $h - 1$, namely the number of edges appearing in it. The *distance* d_{ij} between two vertices i and j of G is the length of the shortest path between i and j; it equals the minimum number of hops that a packet must undergo in a communication between stations i and j.

Two vertices (stations) i and j can generate a *primary collision* if and only if $d_{ij} = 1$, a *hidden collision* if and only if they are two hops away, namely, when $d_{ij} = 2$.

Now, the collisions can be eliminated if i and j transmit on different channels, so interference problems can be mapped to coloring problems on the associated graphs obtained as follows:

- in problem `P-CA` (Primary Collision Avoidance) the associated graph is $G_P = G$;
- in problem `H-CA` (Hidden Collision Avoidance) the associated graph if $G_H = (V, E_H)$ where $E_H = \{(i, j) \in V \times V : d_{ij} = 2\}$;
- in problem `HP-CA` (Hidden and Primary Collision Avoidance) the associated graph is $G_{HP} = (V, E \cup E_H)$.

Complete definitions and discussions of these problems can be found in [1]; a general discussion of graph models of wireless networks can be found in [14], where a general description of constraints is shown and analyzed.

3 Saturation degree code assignment

The basic design principles of the Saturation Degree coloring heuristic proposed in [4] is that the first nodes to be colored are those that have more colors already assigned to nodes in the neighborhood. The motivation is that these nodes have a more constrained choice and therefore a higher risk that at a certain moment, having all colors been assigned to neighbors, a new color needs to be introduced, and a higher overall number of different colors will be necessary in future steps. The heuristic is used in [4] to choose the next branching node in the branch & bound algorithm DSATUR. Korman [10], and later Ramanathan [14], recommended choosing a node with highest degree in the uncolored subgraph (Progressive Minimum Neighborhood First, PMNF) and Kubale and Jackowski [9] validate the choice in their experiments.

```
procedure Saturation Degree
    Order nodes according to decreasing degree in associated graph
    $ToBeAssigned \leftarrow \{1, 2, ..., n\}$
    for $i \leftarrow 1$ to $n$
        $NUnassignedNeighbors[i] \leftarrow Order(i)$
        $NeighCodes[i] \leftarrow \emptyset$
    while $ToBeAssigned \neq \emptyset$
        $MaxNeighCodes \leftarrow -1$
        $MaxUnassignedNeigbors \leftarrow -1$
        for each $i \in ToBeAssigned$
            if $|NeighCodes[i]| > MaxNeighCodes$
                $MaxNeighCodes \leftarrow |NeighCodes[i]|$
                $MaxUnassignedNeigbors \leftarrow NUnAssignedNeighbors[i]$
                $v^* \leftarrow i$
            else if $|NeighCodes[i]| = MaxNeighCodes$
                if $NUnassignedNeighbors[i] > MaxUnassignedNeighbors$
                    $v^* \leftarrow i$
                    $MaxUnassignedNeighbors \leftarrow NUnassignedNeighbors[i]$
        $code[v^*] \leftarrow$ lowest code not in $NeighCodes[i]$
        $ToBeAssigned \leftarrow ToBeAssigned \setminus \{v^*\}$
        for each $j \in N(v^*) \cap ToBeAssigned$
            $NUnassignedNeighbors[j] \leftarrow NUnassignedNeighbors[j] - 1$
            $NeighCodes[j] \leftarrow NeighCodes[j] \cup code[v^*]$
```

Figure 1: The Saturation Degree algorithm

The Saturation Degree algorithm is illustrated in Fig 1. Let $code[i]$ be the code assigned to node i; the set $ToBeAssigned$ contains the yet-unassigned nodes. In addition, to each node i one associates a set $NeighCodes[i]$ containing the codes already assigned to nodes in the neighborhood, and an integer $NUnassignedNeighbors[i]$ given by the number of neighbors that haven't yet been colored. At each iteration, the node v^* that is assigned a code is one with the largest saturation degree (number of codes already assigned in the neighborhood, i.e. number of blocked channels, see lines 8–14). Ties between nodes are broken by preferring the nodes with the highest number of uncolored neighbors (lines 15–17); further ties are broken by node IDs.

As soon as a code is assigned to node v^*, all uncolored neighbors j must decrease by one the number of uncolored neighbors $NUnassignedNeighbors[j]$ (line 22) and update the set of neighboring codes $NeighCodes[j]$ (line 23).

4 The distributed algorithm

So far, the Saturation Degree heuristic has been implemented as a sequential algorithm [1]. In fact, at each step only the node having the globally maximum number of channels blocked is allowed to proceed; if we want to introduce a distributed technique based on Saturation Degree we need to cope with the fact that the knowledge of a node does not extend to the whole network, but is limited to the status of its neighbors, eventually up to a certain depth.

In order to be enabled to choose its own color, a node must ensure that it satisfies the saturation degree condition up to a certain depth of neighbors. We refer to the maximum neighbor depth as the "depth" of the distributed algorithm, and we identify it by the positive integer parameter ρ.

When a node considers itself a candidate for coloring, it sends out a message to ask its neighbors for permission. Every neighbor, up to depth ρ, answers by communicating its own saturation degree and other tie-breaking data. If the candidate node has the highest saturation degree (or, in case of tie, a higher order with respect to its direct contendants), it colors itself, based on the fresh information received by its neighbors. Then it communicates its new status to its neighbors up to depth ρ, which consequently update their saturation degree. If a node increases its saturation degree over the highest known value, it considers itself a new candidate for coloring and

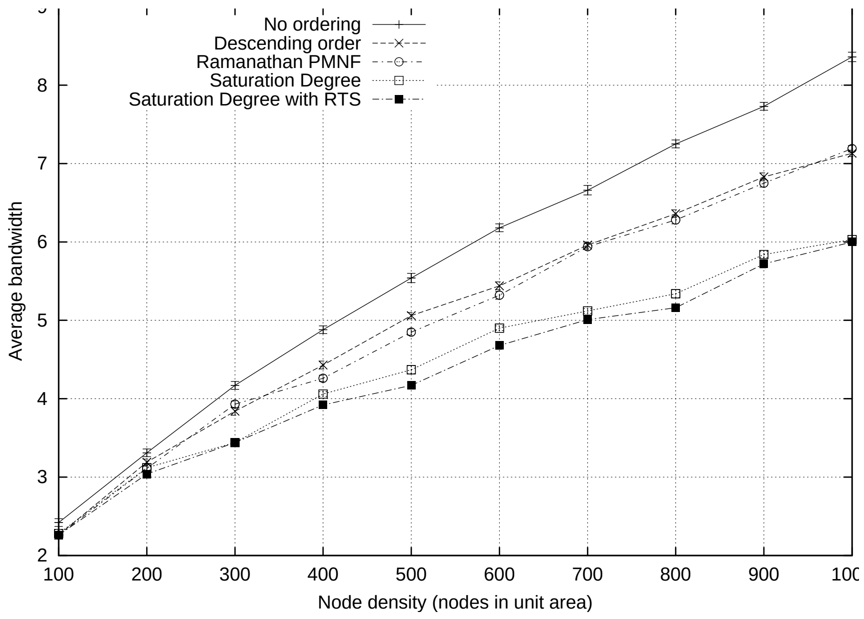

Figure 2: Bandwidth vs Graph Density: comparison among sequential algorithms

starts gathering information from its neighbors.

At the beginning of the coloring procedure, every node considers itself a candidate for coloring.

The performance of the distributed saturation-degree heuristic is expected to improve as ρ grows. In fact, if the graph is fully connected the distributed algorithm is almost equivalent to its sequential version when ρ is large enough.

5 Experimental results

Simulations were executed by uniformly scattering random points on a unit square. Each point, representing a mobile host, is able to broadcast a radio signal within a circle of radius 0.05 units. As all hosts have the same power, the Euclidean communication graph (a vertex for every host, a link between points if they are nearer than 0.05 units) is undirected.

The problem taken into account in the experimental tests is the hidden interference problem, where two nodes are connected by an arc if and only if they are second-order neighbors in the primary Euclidean graph (i.e. iff they are not primary neighbors and have a common primary neighbor). Evidence is given in [1] that, of the three possible problems (primary interference, hidden interference or both), the hidden interference is the most difficult to treat.

In figure 2 we show a comparison among some sequential greedy techniques: unordered and descending order [3], PMNF (see Section 3) [14], the sequential Saturation Degree heuristic described in Section 3 [1] and the same heuristic followed by two iterations of Reactive Tabu Search [2]. Every algorithm has been tested for 10 different node densities (from 100 to 1000 nodes); every plotted point is the average of 100 simulations; error bars represent the 95% confidence interval for the true average value; due to the high number of runs, the bar size is barely visible. The saturation degree heuristic (either in its pure and postprocessed form) clearly outperforms all other considered algorithms, in particular it improves the PMNF technique by more than 16%. This obvious advantage motivated our choice of the algorithm to distribute. In fact, a small degradation of result quality could be introduced by relaxing the complete-knowledge requirements of any centralized algorithm; while this degradation is acceptable when other advantages are considered, it is our purpose to minimize it.

Figure 3 plots the resulting average bandwidth (number of different channels required to avoid hidden interference) versus the number of nodes placed in the square. The different lines show the behavior of the sequential Saturation Degree algorithm and of the distributed version when every node compares its saturation and its order

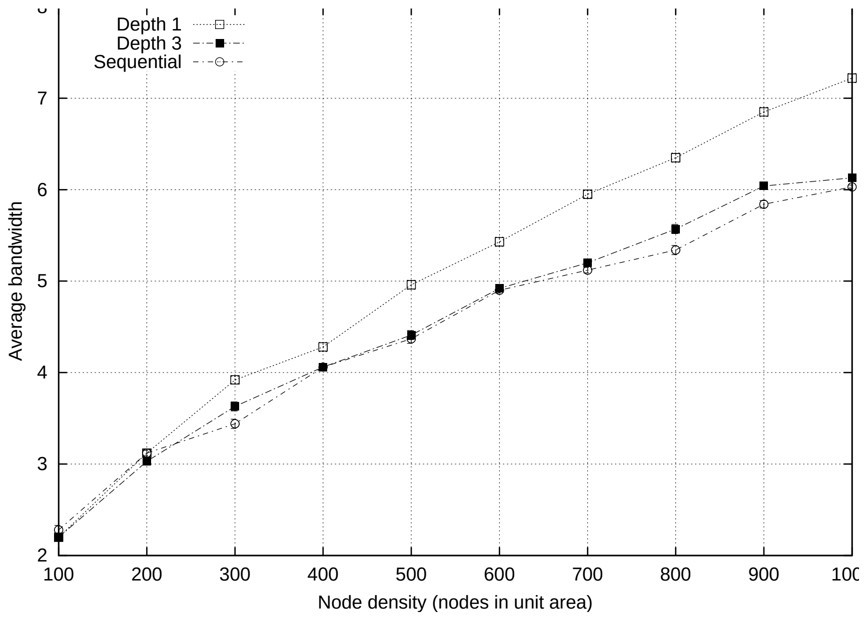

Figure 3: Bandwidth vs Graph Density: comparison among different depths of the distributed algorithm

with neighbors of depth 1 and 3. Here, too, points represent the average and the confidence interval after 100 simulations.

Figure 3 shows that taking into account the status of third-order neighbors ($\rho = 3$) improves the algorithm's behavior at such point to make it almost indistinguishable from the sequential ($\rho \to \infty$) version. Figure 4 further stresses this point by showing how the depth of the distributed algorithm affects its performance. We can see that no improvements can be achieved with more than three or four orders of neighbors. The graph has been obtained with 10 simulations per point.

The simulation algorithm operates in *steps*: at every step, all nodes satisfying the saturation degree condition up to the given depth ρ of neighbors are selected and colored. The total number of passes required to color the whole graph (which in a truly distributed setting, together with ρ, would determine the overall coloring time) is plotted in figure 5. Each point is the average number of passes in 10 runs of the coloring algorithm.

Figure 6 shows the behavior of the distributed saturation degree algorithm[1] during various passes. At the beginning (a), uncolored nodes are placed randomly on the 1×1 square. After pass 1 (b), some sparse nodes are colored (those who prevail within their neighborhood up to depth 3). These nodes act as clustering kernels. After some more passes, clusters of colored nodes have grown (c) and begin to merge (d).

A case where many nodes get colored in a single pass is, of course, when nodes are too sparse to be connected (Figure 7): this is the reason why the total number of passes shown in Figure 5 remains so low on a 100-nodes network.

6 Conclusions

The results shown in Section 5 justify the choice of the Saturation Degree heuristic for approximating the optimal solution to the Channel Assignment problem.

Our results also show that, although the distributed version of the Saturation Degree algorithm that considers only first neighbors deteriorates the algorithm performance, a variation where neighbors up to a given distance are considered before taking a decision achieves results that are comparable to those of other preprocessed greedy techniques. Moreover, a tradeoff between parallelism and performance can be obtained by tuning the

[1]Figures 6 and 7 have been grabbed from a visual simulation written as a java applet and available on `http://rtm.science.unitn.it/~brunato/dsd/`. All other simulations have been written in `C++`.

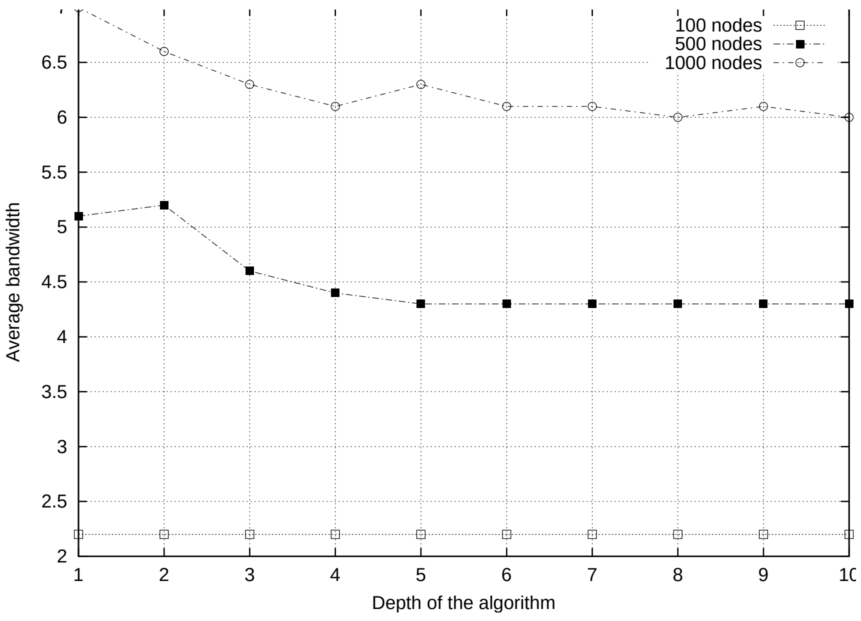

Figure 4: Bandwidth vs depth of the algorithm

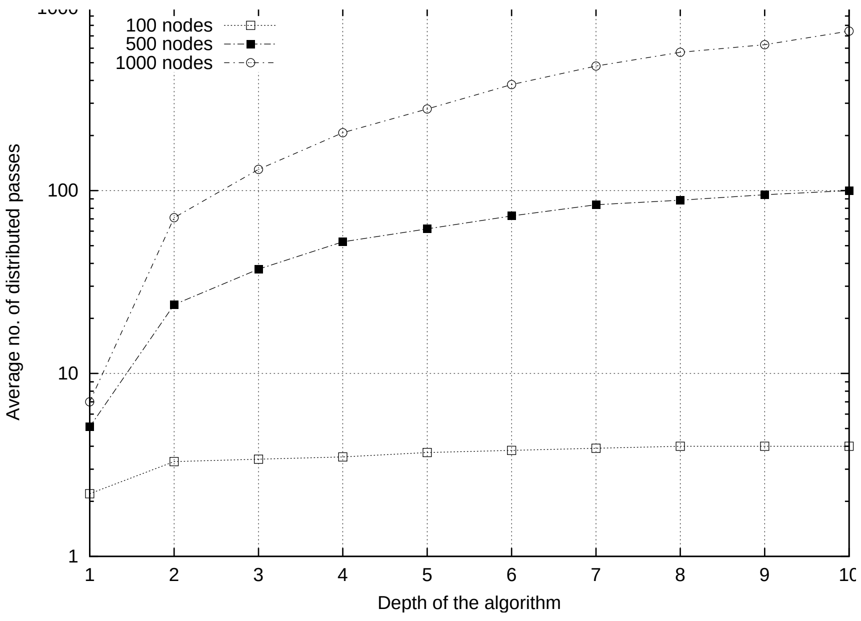

Figure 5: Number of distributed passes vs depth of the algorithm

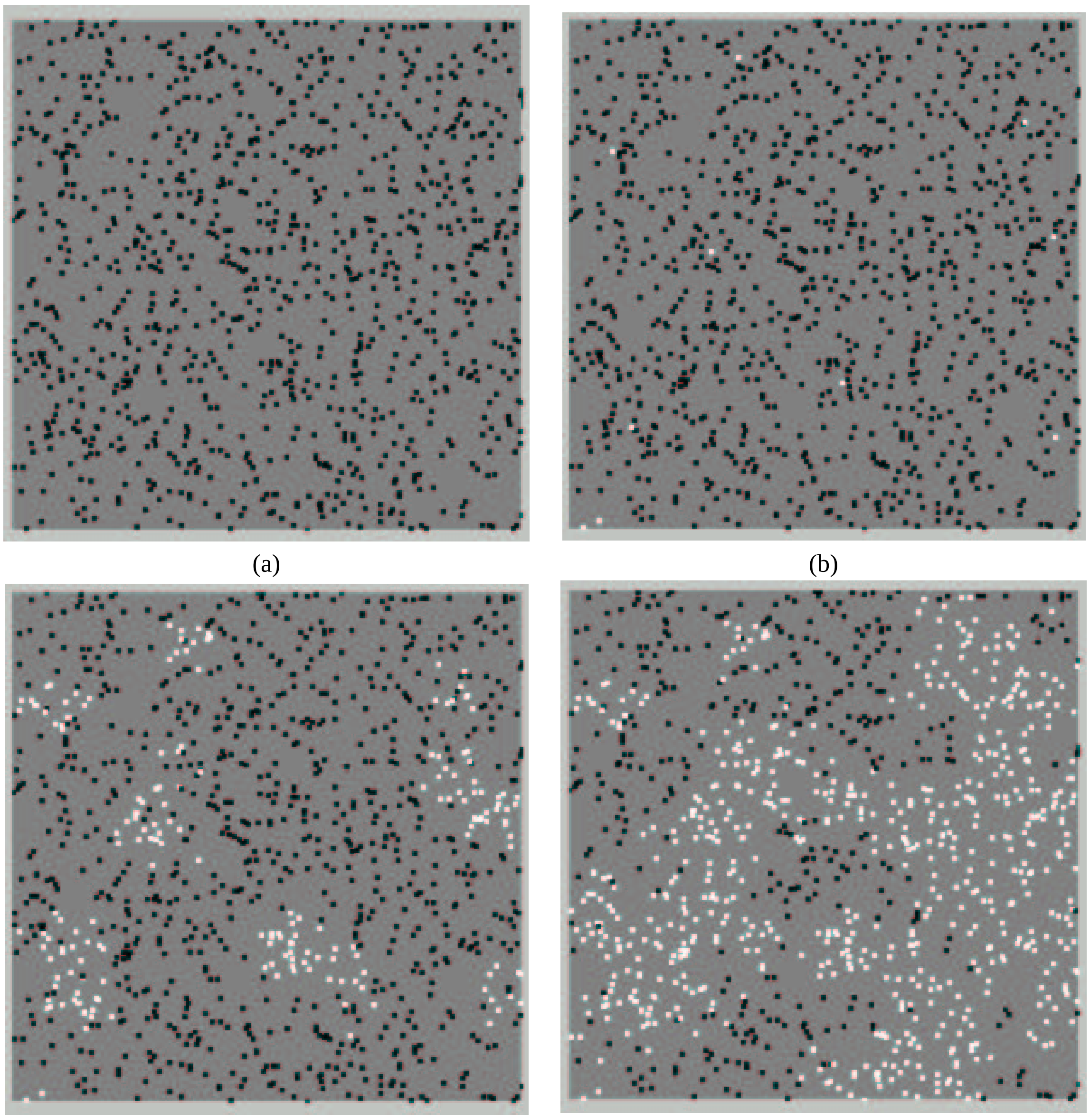
(a)
(b)
(c)
(d)

Figure 6: The coloring process (1000 nodes, radius $.05$, $\rho = 3$) after 0 (a), 1 (b), 37 (c), 170 (d) passes; white nodes are colored.

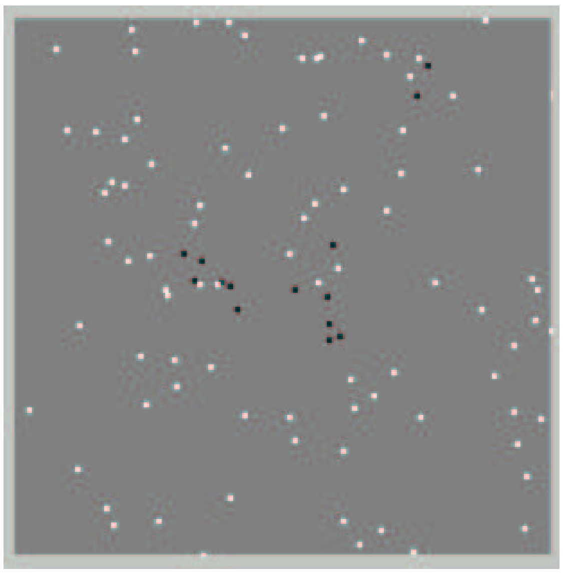

Figure 7: Extreme paralellism (100 nodes): after 1 pass most nodes are colored.

parameter ρ, which indicates how "deep" a message must travel in the neighborhood. A good performance is obtained by just setting ρ to 3.

The qualitative behavior of the distributed algorithm has also been exploited by means of a graphical simulation program, which has been made publicly available.

Because of space limits, all details and the analysis of the distributed algorithm will be presented in an extended version of this paper.

References

[1] R. Battiti, A. Bertossi and M. Bonuccelli. Assigning Codes in Wireless Networks: Bounds and Scaling Properties. *Wireless Networks* **5** (1999) 195–209.

[2] R. Battiti and M. Protasi. Reactive local search for the maximum clique problem. *Technical Report* TR-95-052, ICSI, 1947 Center St.- Suite 600 - Berkeley, California (1995).

[3] A. A. Bertossi and M. A. Bonuccelli. Code Assignment for Hidden Terminal Interference Avoidance in Multihop Packet Radio Networks. *IEEE Trans. on Networking* **3**(4) (1995) 441–449.

[4] D. Brélaz. New methods to color the vertices of a graph. *Communications of the ACM* **22** (1979) 251–256.

[5] I. Chlamtac and S. Kutten. On broadcasting in radio networks – Problem analysis and protocol design, *IEEE Trans. Commun.* (Dec. 1985).

[6] I. Chlamtac and S. Kutten. A spatial reuse TDMA/FDMA for mobile multi-hop radio networks. In: *Proc. IEEE INFOCOM* (Mar. 1985).

[7] I. Chlamtac and S. S. Pinter. Distributed Nodes Organization Algorithm for Channel Access in a Multihop Dynamic Radio Network. *IEEE Trans. Computers* **36** (1987) 728-737.

[8] L. Hu. Distributed Code Assignments for CDMA Packet Radio Networks. *IEEE Trans. on Networking* **1**(6) (1993) 668–677.

[9] M. Kubale and B. Jackowski. A generalized implicit enumeration algorithm for graph coloring. *Communications of the ACM* **28** (1985) 412–418.

[10] S. M. Korman. The graph-coloring problem. In: N. Christophides, P. Toth, and C. Sandi, editors, *Combinatorial Optimization*, pp. 211-235 (Wiley, New York, 1979).

[11] T. Makansi. Transmitted-oriented code assignment for multihop packet radio. *IEEE Trans. Communications* **35** (1987) 1379-1382.

[12] R. Nelson and L. Kleinrock. Spatial TDMA: a collision-free multihop channel access protocol. *IEEE Trans. Communications* **33** (1985) 934–944.

[13] M.J. Post, A.S. Kershenbaum, and P.E. Sarachik. Scheduling multihop CDMA networks in the presence of secondary conflicts. *Algorithmica* **4** (1989) 365-394.

[14] S. Ramanathan. A Unified Framework and Algorithm for Channel Assignment in Wireless Networks. *Wireless Networks* **5** (1999) 81–94.

[15] E. Lloyd and S. Ramanathan. Efficient Distributed Algorithms for Channel Assignment in Multihop Radio Networks. *Journal of High Speed Networks* **2** (1993), 405–428.