PERFORMANCE EVALUATION OF DISTRIBUTED SYNCHRONOUS GREEDY GRAPH COLORING ALGORITHMS ON WIRELESS AD HOC AND SENSOR NETWORKS

Esra Ruzgar[1] and Orhan Dagdeviren[2]

[1]Software Engineering Department, Izmir University, Izmir, Turkey
esra.ruzgar@izmir.edu.tr
[2]International Computer Institute, Ege University, Izmir, Turkey
orhan.dagdeviren@ege.edu.tr

ABSTRACT

Graph coloring is a widely used technique for allocation of time and frequency slots to nodes, for forming clusters, for constructing independent sets and dominating sets on wireless ad hoc and sensor networks. A good coloring approach should produce low color count as possible. Besides, since the nodes of a wireless ad hoc and sensor network operate with limited bandwidth, energy and computing resources, the coloring should be computed with few message passing and computational steps. In this paper, we provide a performance evaluation of distributed synchronous greedy graph coloring algorithms on ad hoc and sensor networks. We provide both theoretical and practical evaluations of distributed largest first and the distributed version of Brelaz's algorithm. We showed that although distributed version of Brelaz's algorithm produces less color count, its resource consumption is worse than distributed largest first algorithm.

KEYWORDS

Mobile Ad hoc Networks, Distributed Graph Coloring, Distributed Largest First Algorithm, Brelaz's Algorithm.

1. INTRODUCTION

The availability of wireless devices have been increased rapidly in recent years and number of these devices reached to enormous amounts. So need for self-organizing networks not require pre-established infrastructure become inevitable. Ad-hoc networks consists of many autonomous wireless devices that communicates with each other via radio signals. Ad-hoc networks can be static or mobile. In static ad-hoc networks, positions of devices do not change after it joined the network, whereas, in mobile ad-hoc networks, devices can move arbitrarily.

One of most common problems for wireless ad-hoc networks is accessing the shared wireless medium. If two or more neighbor nodes in an ad hoc network transmit at the same time, they cause interference and receiver node hears only noise. Media access control (MAC) protocols for wireless ad hoc and sensor networks try to coordinate the access to the shared medium. MAC protocols use three ways, time (TDMA), frequency (FDMA), or code division multiple access (CDMA) schemes to divide the channel among the nodes. Channel assignment is an NP-Hard problem [1]. Many methods have been proposed to design efficient MAC protocols for wireless ad hoc and sensor networks [2-5]. TDMA, FDMA, and CDMA protocols are implemented by a

DOI : 10.5121/ijcnc.2013.5213

vertex coloring of the graph constructed according to interference relations [6]. Many MAC protocols use distributed graph coloring for coordinating access to the shared wireless medium [7-16].

Another application of distributed graph coloring is finding maximal independent sets and dominating sets in wireless networks. A maximal independent set (MIS) in an undirected graph $G = (V, E)$ is a maximal collection of vertices I ▫ V with the restriction that no pair of vertices in I are adjacent. A dominating set (DS) is a collection of vertices D ▫ V with the restriction that each vertex in V is either in D or adjacent to one vertex in D. These two problems are strongly related to graph coloring problem. In solution of optimal graph coloring problem, each set of vertices that share same color in a graph forms a maximal independent set [17].

In wireless ad hoc and sensor networks, partitioning the nodes into clusters is also very important. Clustering is a fundamental approach to manage the wireless ad-hoc networks. In clustered networks, nodes can be cluster members or cluster heads. A cluster member is an ordinary cluster node which only sends its request to its cluster head. A cluster head collects requests from cluster members and manage inter cluster operations [18]. The most important advantage of clustering is minimizing the amount of data to be exchanged between nodes. Clustering also enables systems to adapt dynamically with changing network configurations [19]. Most greedy clustering algorithms propose methods for finding independent or dominating sets of nodes which can act as set of clusterheads. Several papers use graph coloring for constructing independent or dominating sets in wireless networks [20-22].

Wireless ad hoc and sensor networks consist of devices with limited power and storage capabilities. Many wireless devices operate with only batteries so they have to keep computational time as short as possible. Distributed algorithms that operate on wireless devices communicate with each other with message passing and there is no central coordination. So, the nodes have to use algorithms which do not need high amount of message passing. These constraints make us choose distributed algorithms with as little computational time and message passing as possible. Selection of appropriate distributed graph coloring algorithm for wireless ad hoc networks is based on these constraints [11].

The rest of this paper is organized as follows. In Section 2, the background information for ad-hoc network modeling and distributed graph coloring problem are given. In Section 3, the related work about distributed graph coloring algorithms for ad-hoc networks are surveyed. The implemented algorithms and performance evaluations are described in detail in Section 4 and Section 5 respectively. Lastly, conclusions are given in Section 6.

2. BACKGROUND

2.1 Network Model

A wireless ad hoc or sensor network assumed to have following properties:

- Each node in the network has a unique id.
- There are directional links between nodes. This means that if there is a link from node u to node v, there is also a link from node v to node u.
- Each node knows ids of its neighbors in its transmission range.

With considering these properties, the set of nodes in an ad-hoc or sensor network can be modeled as an undirected graph $G = (V, E)$. In graph G, the nodes are vertices and links are edges that existing between two vertices if they are close enough to communicate directly with each other. We denote the number of nodes in the network by $n = |V|$ and the maximum degree of G by Δ . In Figure 1, in the left side a sample ad-hoc network with 4 nodes and their transmission ranges is given and in the right side its corresponding undirected graph is given.

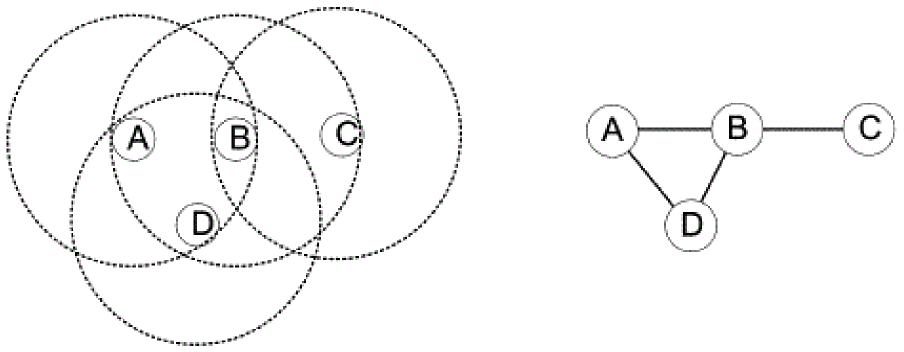

Figure 1. A sample ad-hoc network and its graph representation

2.2. Graph Coloring

Graph coloring is defined as coloring the elements of a graph with using the minimum number of colors with the restriction that any two adjacent element of the graph cannot have the same color. Graph coloring has two common forms: vertex coloring, edge coloring. The most common form is vertex coloring and it is coloring vertices of a graph that any adjacent vertices have the same color. Very similar to vertex coloring, edge coloring aims to color the edges of a graph that any adjacent edges have the same color. In this paper, we refer graph coloring as vertex coloring. The coloring of a graph $G = (V, E)$ is a mapping $c: v \longrightarrow s$, where s is a set of colors, such that if $vw \in E$ then $c(v) \neq c(w)$. This means that any adjacent vertices are not colored with the same color. The chromatic number $\chi(G)$ is the minimum number of colors needed for a coloring of a graph G. A graph G is k-chromatic, if $\chi(G) = k$, and G is k-colorable, if $\chi(G) \leq k$ [23].

3. RELATED WORK

Many methods have been proposed to solve graph coloring problem. It is an easy case to decide if a given graph can be colored with 1 color or 2 colors. But it is NP-complete to decide if a given graph has a k-coloring for $k \geq 3$. It is also NP-hard to find the chromatic number $\chi(G)$ of a given graph. It is also difficult to find an approximation ratio for chromatic number, it is NP-hard to approximate the chromatic number within $n^{1-\varepsilon}$, where $\varepsilon > 0$.

Chromatic number of a graph can be found in polynomial time for some special graphs. To find that a given graph can be colored with 2 colors is equal to find that it is a bipartite graph or not. Testing for bipartiteness can be done using depth-first search in linear time. For other special classes of graphs such as chordal graphs, cycles, wheels, ladders, trapezoid graphs chromatic numbers can be computed in polynomial time [25, 26].

Some exponential-time exact algorithms have been proposed to find k-coloring of a graph. The easiest exact approach is making a brute-force search for a k-coloring. In the brute force search we should consider each of the k^n assignments of k colors to n vertices and check for each that any adjacent vertex do not share the same color. To find the chromatic number, brute force is done for every $k = 1, ..., n-1, n$ and the minimum value of k with a legal coloring is chromatic number. This procedure can be only used for small graphs. Other exact algorithm for finding chromatic number in $\mathrm{O}(2.415^{n})$ time uses a dynamic programming approach and puts a bound on the number of maximal independent sets [27]. Yate's algorithm uses the principle of inclusion–exclusion and find that if a graph is k-colorable or not in $\mathrm{O}(2^{n}\mathrm{n})$ time for any *k* [28]. For finding 3- and 4-colorability of a graph $\mathrm{O}(1.3289^{n})$ time and $\mathrm{O}(1.7504^{n})$ time algorithms have been proposed recently [29, 30].

The greedy algorithms for graph coloring problem considers the vertices in a specific order and assigns to the smallest available color not used by any adjacent vertices. The fastest and easiest heuristic is First Fit (FF) algorithm. FF sequentially assigns a vertex with the lowest possible color and its time complexity is $O(n)$. In other important heuristics, vertices are ordered according to their degrees. If the vertex with maximum degree is selected to be colored, then this heuristic is called Largest First (LF) algorithm [31]. Time complexity of LF is $O(n^{2})$. Another heuristic due to Brélaz establishes the ordering dynamically while the algorithm proceeds, choosing next the vertex adjacent to the largest number of different colors. The number of different colors used by neighbors of a node is called saturation degree of that node, so Brelaz's algorithm is also called Dsatur algorithm [32]. Dsatur's time complexity is $O(n^{3})$.

Many distributed algorithms have been proposed for graph coloring problem. The most important issue in distributed approach is breaking the symmetry between nodes. It is because deterministic algorithms cannot find a legal coloring of a symmetric graph. Randomized algorithms are used to overcome this problem. The state-of-art randomized algorithms for graph coloring are faster than deterministic algorithms. An $O(\Delta)$-coloring can be computed in a randomized way in $O(\sqrt{\log n})$ time where Δ is the maximum degree of the graph. [33]. Using randomization, a $(\Delta+1)$-coloring of a graph can be found with an algorithm based on finding maximal independent set of the graph [20] and it is an $O(\mathrm{logn})$ time algorithm. The fastest randomized algorithm uses the multi-trials technique of Schneider and Wattenhoffer. By this technique, $(\Delta+1)$-coloring takes $O(\log\Delta+\sqrt{\log n})$ time and to find an $O(\Delta+\log^{1+1/\log^{*} n} n)$ coloring takes $\mathrm{O}(\log *\mathrm{n})$ time [34].

Linial studied the lower bound of time complexity of distributed graph coloring in [35]. The main result of [35] is $\Omega(\log^{*} n)$ time is lower bound for coloring a ring with a constant number of colors. Some of best deterministic algorithms to compute a $(\Delta+1)$-coloring have time complexities of $O(2^{\sqrt{\log n}})$ and $\mathrm{O}(\Delta\log\Delta+\log *\mathrm{n})$ [36, 37]. The fastest deterministic algorithm for $(\Delta+1)$-coloring for small Δ runs in time $\mathrm{O}(\Delta)+\log^{*}(n)/2$. This result is is optimal in terms of *n* because according to Linial's lower bound the constant factor 1/2 cannot be improved [38]. For special graph classes, there are more efficient deterministic algorithms. For rings and bounded degree graphs, a $(\Delta+1)$-coloring can be computed in $\mathrm{O}(\log *\mathrm{n})$ time [39, 40]. This time complexity is also applied to much larger class of graphs with bounded local independent sets [41]. In particular, these graph classes contains most of the graph classes that are used to model wireless ad hoc and sensor networks [11].

Greedy approach is also used to compute distributed graph coloring problem. The Largest First and Brelaz's heuristics are applied to graphs in a distributed setting; nodes compute minimal colorings according to local and neighbors' data. Distributed version of LF algorithm is called Distributed Largest First (DLF) and studied in [42]. Brelaz's saturation degree heuristic was implemented for parallel computers in [43], but it has not been implemented for a distributed system. Due to their implementation simplicity, they are appropriate for many applications.

Details of implemented greedy algorithms are explained in the next sections. Performance evaluation of distributed greedy graph coloring algorithms in many aspects is also provided.

4. ALGORITHMS

Greedy distributed graph coloring algorithms are simpler than the fast deterministic and randomized algorithms mentioned before. They are also easier to implement on large graphs, they compute optimal colorings for most general graphs and optimum results for bipartite graphs. For many applications, the number of used colors and total messages sent between nodes are optimal.

4.1. DLF

Distributed Largest First (DLF) algorithm [42] is based on sequential LF algorithm. The main assumptions of the algorithm are there is no shared memory and each processor knows only its own links and its unique identifier. They aim these units to compute a coloring of the associated graph without any other information about the structure of G. Another assumption is that the system is synchronized in rounds. The number of rounds will be measure of efficiency. In each round, the nodes with largest degrees are selected to be colored. In [42] they claim that,

1. The algorithm does not use colors more than $(\Delta + 1)$ colors.
2. At each round at least one node gets colored.
3. In each round, in whole graph at most one new color can be assigned.

In DLF algorithm, each vertex has three parameters: degree: *deg(v)*, random value: *rndvalue(v)*, palette of forbidden colors, which were used by its neighbors: *usedcolor(v)* (initially empty). To find the priority between two node *u* and *v*, we follow the following rules:

deg(v) > deg(u)

or

(deg(v) = deg(u)) AND (rndvalue(v) > rndvalue(u))

In the implementation of DLF, each node sends four types of messages: *DATA* messages are for informing neighbors about its own *degree*, *rndvalue* and *legal color*; *COLOR* messages are for informing neighbors the color it used, so neighbors can update their *usedcolor* list. Legal color means the lowest numbered color that a node wants to use if it has the highest priority; *START* messages are send by a system node to start new round and *END* messages are sent by the system node to end application.

The pseudocode of DLF is as follows. During each round every uncolored vertex *v* executes the following five steps:

1. Choose parameter *rndvalue(v).*
2. Send to all neighbors the following parameters: *deg(v)*, *rndvalue(v)*, and the first legal color (not on the list of forbidden colors).
3. Compare its own parameters with these received from the neighbors and check which vertex has the highest priority.
4. If vertex *v* has the highest priority, keep the proposed color and stop.
5. If not, update list *usedcolor(v).*

In Figure 2, a sample execution of DLF algorithm is shown. DLF algorithm colors the graph with 3 colors in 3 rounds. In first round, the node with maximum degree is colored with first legal color, in the next rounds the uncolored nodes with maximum degrees in their neighborhood are colored with their first legal color.

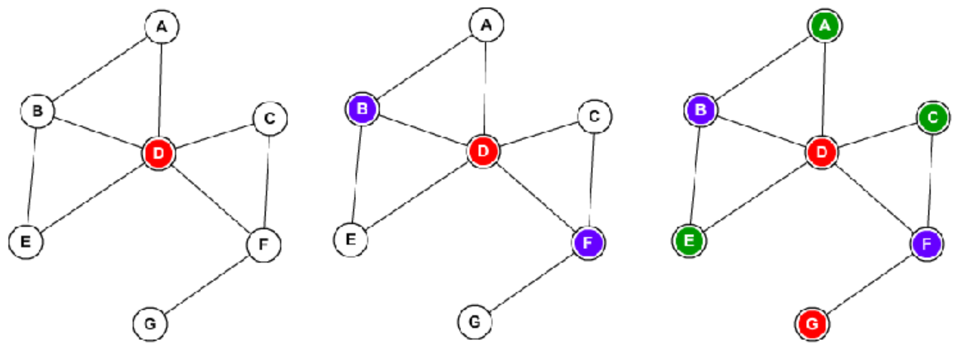

Figure 2. A sample execution of DLF algorithm

4.2 D_Dsatur

The sequential Dsatur algorithm [32] of Brelaz is a sequential coloring algorithm with a dynamically established order of the vertices. The degree of saturation of a vertex *x*, *degs(x)*, is the number of different colors at the vertices adjacent to *x*. Dsatur starts by assigning lowest numbered color to a vertex of maximal degree. The vertex to be colored next is a vertex *x* with maximal *degs(x).* The distributed version of Brelaz's Dsatur algorithm is called D_Dsatur in this paper.

The implementation of D_Dsatur is same as DLF. Each node keeps same data structures as in DLF, but each node also keeps its saturation degree. The message types are also same as DLF. The pseudocode of D_Dsatur is as follows. During each round every uncolored vertex *v* executes the following five steps:

```
1. Choose parameter rndvalue(v).
2. If it is first round then send to all neighbors the following parameters: deg(v), rndvalue(v),
and the first legal color,
    else send to all neighbors the following parameters: sat_deg(v), rndvalue(v), and the first
legal color.
3. Compare its own parameters with these received from the neighbors and check which
vertex has the highest priority.
4. If vertex v has the highest priority, keep the proposed color and stop.
5. If not, update list usedcolor(v).
```

In Figure 3, a sample execution of D_Dsatur algorithm is shown. D_Dsatur algorithm also colors the graph with 3 colors in 3 rounds. In first round, the node with maximum degree is colored with first legal color, in the next rounds the uncolored nodes with maximum saturation degrees in their neighborhood are colored with their first legal color. If two nodes have same saturation degrees, such as A, B and E in round two, the nodes with greater random numbers color themselves.

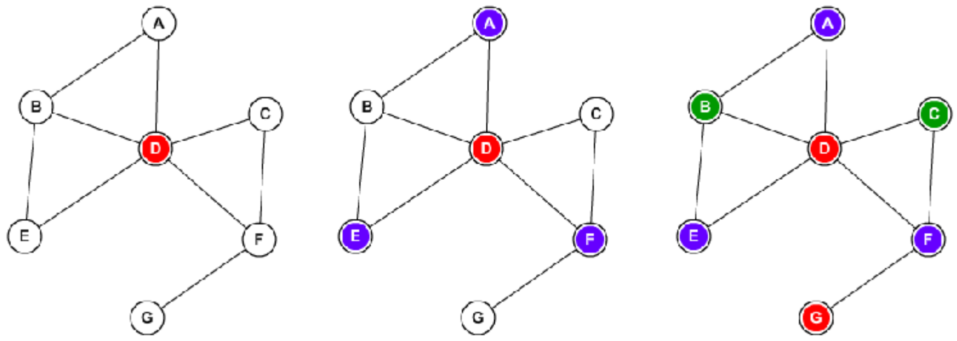

Figure 3. A sample execution of D_Dsatur algorithm

5. PERFORMANCE EVALUATION

DLF and D_Dsatur algorithms are also tested for large graph with varying node count and densities to compare them in terms of number of colors used, message counts and round counts. The algorithms are implemented in C language. The distributed environment is provided by using a synchronous thread based simulator. The simulator provides a library [44] for message passing between threads. Random graphs are generated and nodes communicate with each other by using the functions in the library. The simulator with a sample application can be downloaded from [45].

5.1 Theoretical Evaluation

When a distributed graph coloring algorithm is evaluated theoretically, there are two important measures to consider. The first important measure is time complexity of the distributed algorithm. The second important thing is message complexity of the distributed algorithm.

Both DLF and D_Dsatur algorithms use at most $(\Delta+1)$ colors, and they are not effective if $\Delta=\Theta(n)$. But, in practice graphs are sparse and numbers of used colors are nearly optimal [31]. Both algorithms produce optimal colorings for complete k-partite graphs, crown graphs, wheels.

Hansen et al. claims that DLF algorithm runs in $O(\Delta^2 \log n)$ rounds for random graphs, where n is number of vertices and Δ denotes the largest vertex degree. The detailed proof of time complexity of DLF is given in [42], but message complexity is not given. For our implementation of DLF, in each round there are at most $O(n^2)$ messages are sent, each node sends either DATA or COLOR message in each round and each node has at most n-1 neighbors, so message complexity is computed as $O(n^2\Delta^2 \log n)$.

To the best of our knowledge, there is not a detailed research about time complexity of distributed Dsatur algorithm. In [44], it is claimed that any distributed implementation of saturation based coloring algorithms require $\Omega(n)$ rounds, so it is not suitable for distributed applications. In practice, at each round, at least one node gets colored and D_Dsatur algorithm needs $O(n)$ rounds to complete coloring. The message complexity can be calculated as $O(n^3)$, where at each round at most $O(n^2)$ messages are sent such as DLF algorithm. Detailed analysis is required on the time complexity of D_Dsatur algorithm.

5.2 Practical Evaluation

DLF and D_Dsatur algorithms should also be compared experimentally in terms of number of colors used, number of total messages sent and execution time. It is not appropriate to use running time of algorithm for execution time comparison. It is because running time of the algorithm not only depends on the size of input, but also it depends on other running processes on the same processor. So number of rounds required to complete coloring is used for execution time comparison.

Experimental comparison of DLF and D_Dsatur algorithms are given in Table 1. Node count varies from 50 to 200, for each node count 3 different graphs are generated with increasing edge densities. Both algorithms are evaluated in terms of number of colors used, number of total messages sent and number of total round for each graph.

Experimental results show that DLF algorithm shows poorer performance than D_Dsatur algorithm in terms of total number of colors used. For the graphs of same size, D_Dsatur uses less different colors than DLF algorithm. When node count is constant and edge density of the graph increases, the number for colors used also increases for both algorithms. These results are very close to the practical comparisons of sequential LF and Dsatur algorithms such as in [27, 47].

Table 1. Experimental comparison of DLF and D_Dsatur algorithms

No.of Vertices	**No.of Edges**	**No. of Colors Used**		**No. of Total Rounds**		**No. of Total Messages**	
		DLF	**D_Dsatur**	**DLF**	**D_Dsatur**	**DLF**	**D_Dsatur**
50	125	4	3	8	12	704	860
	625	11	10	22	34	10369	15109
	1250	15	14	25	36	15023	20091
100	500	7	6	13	20	5675	7730
	2500	21	20	43	68	77411	134612
	5000	26	23	52	73	125438	171920
200	2000	9	8	22	33	34739	49644
	10000	36	32	94	136	623642	1024991
	20000	44	42	110	143	979443	1429561

According to Table 1, DLF algorithm generally uses one or two more colors than D_Dsatur for each node count and edge density. For graphs with 50 nodes, number of colors used for sparse graphs is 7%, for mid-sparse graphs it is 21% and for dense graphs it is 29% of node count in average of two algorithms. When number of nodes increase to 100, the average values of colors used are 6.5%, 20.5% and 24.5% of node count. For graphs of size 200, the average values of colors used are 4%, 17% and 21.5% of node count. When graph keeps same node count and gets denser, the number of colors used increases significantly. But when the node count of graph increases the percentage of number of colors used decreases.

DLF algorithm shows better performance than D_Dsatur algorithm in terms of execution time. It is because DLF needs fewer rounds for coloring than D_Dsatur. For graphs with 50 nodes, number of rounds for sparse graphs is 32%, for mid-sparse graphs it is 13% and for dense graphs it is 11% of node count for DLF algorithm. For each graph, D_Dsatur uses rounds more than approximately 52% of rounds that DLF algorithm uses. For large graphs it is more practical to choose DLF when number of colors used is not very critical.

The numbers of total messages sent are also larger for D_Dsatur, it is because D_Dsatur needs more rounds to finish coloring. In Table 1, for message counts only DATA and COLOR messages are considered, other synchronization messages are not considered. For each graph D_Dsatur sends messages more than approximately 50% of messages that DLF algorithm sends. Experimental results are strongly related to theoretical time and message complexities.

6. CONCLUSION

Distributed graph coloring has many applications in wireless ad hoc and sensor networks. The most common applications are assigning frequency, time and code slots to nodes in a shared wireless medium. Other important applications include finding independent sets, dominating sets and forming clusters of nodes in wireless networks.

Finding optimal coloring of a graph is NP-complete so we need some heuristics to solve this problem in a sub-optimal way. Many heuristics and algorithms have been proposed in this field and each of them tries to improve the existing algorithms in terms of execution time or number of colors used. Greedy algorithms are efficient approaches among the other algorithms to solve the graph coloring problem.

In this paper, two greedy distributed graph coloring algorithms DLF and D_Dsatur are examined in detail. They are implemented and tested for different graphs; their performances are compared in terms of number of colors used, total rounds and total messages sent. Experimental results show that DLF algorithm shows poorer performance than D_Dsatur algorithm in terms of total number of colors used whereas DLF algorithm shows better performance than D_Dsatur algorithm in terms of execution time and total messages sent. The results show us that DLF is suitable for energy constrained sensor networks, on the other hand D_Dsatur can be a good choice for IEEE 802.11 based ad hoc networks.

7. REFERENCES

[1] X. Wu, A. Jaekel & A. Bari, (2011) “Optimal Channel Allocation with Dynamic Power Control in Cellular Networks”, *International Journal of Computer Networks & Communications (IJCNC)*, Vol.3, pp. 83-93.

[2] M. Ramakrishnan & P. V. Ranjan, (2009) “Multi Channel MAC for Wireless Sensor Networks”, *International Journal of Computer Networks & Communications (IJCNC)*, Vol.1, pp. 47-54.

[3] G. Ma & D. Qiu, (2009) “An Efficient MAC Protocol Based On Hybrid Superframe For Wireless Sensor Networks”, *International Journal of Computer Networks & Communications (IJCNC)*, Vol.1, pp. 75-82.

[4] B. Dezfouli, M. Radi & S. A. Razak, (2010) “A Cross-Layer Approach for Minimizing Interference and Latency of Medium Access in Wireless Sensor Networks”, *International Journal of Computer Networks & Communications (IJCNC)*, Vol.2, pp. 126-142.

[5] S. S. Bamber, & A. K. Sharma, (2010) “Performance Trade Off With Modulation in 802.15.4 Wpan for Wireless Sensor Networks”, *International Journal of Computer Networks & Communications (IJCNC)*, Vol.2, pp. 77-87.

[6] M. Szegedy & S. Vishwanathan, (1993) “Locality based graph coloring”, *In Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pp. 201-207.

[7] R. Kawano & T. Miyazaki, (2007) “Distributed coloring algorithm for wireless sensor networks and its applications”, *In Proc. of Computer and Information Technology*, pp. 997-1002.

[8] S. Ramanathan, (1999) “A unified framework and algorithm for channel assignment in wireless networks”, *Wireless Networks*, Vol.5, pp. 81-94.

[9] W. K. Hale, (1980) “Frequency assignment: Theory and applications”, *In Proceedings of the IEEE*, Vol. 68, pp. 1497-1514.

[10] C. McDiarmid & B. Reed, (2000) “Channel assignment and weighted coloring”, *Networks*, Vol. 36, pp. 114-117.

[11] F. Kuhn, (2009) “Local Multicoloring Algorithms: Computing A Nearly-Optimal TDMA Schedule In Constant Time”, *In Proc. of Symposium on Theoretical Aspects of Computer Science*, pp. 613–624.

[12] T. Herman & S. Tixeuil, (2004) “A distributed TDMA slot assignment algorithm for wireless sensor Networks”, *Algorithmic Aspects of Wireless Sensor Networks*, Vol. 3121, pp. 45-58.

[13] L. Bao & J.J. Garcia-Luna-Aceves, (2001) “A new approach to channel access scheduling for ad hoc networks”, *In Proceedings of the 7th ACM annual international conference on Mobile computing and networking*, pp. 210-221.

[14] A. Mishra, V. Shrivastava, D. Agrawal, S. Banerjee & S. Ganguly, (2006) “Distributed channel management in uncoordinated wireless environments”, *In Proceedings of the 12th annual international conference on Mobile computing and networking*, pp. 170-181.

[15] I. Rhee, A. Warrier, J. Min & L. Xu, (2006) “DRAND: distributed randomized TDMA scheduling for wireless ad-hoc networks”, *In Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, pp. 190-201.

[16] C. Guo, L.C. Zhong & J.M. Rabaey, (2001) “Low power distributed MAC for ad hoc sensor radio networks” *In Proc. of IEEE Global Telecommunications Conference*, pp. 2944-2948.

[17] D. Eppstein, (2001) “Small maximal independent sets and faster exact graph coloring”, *J. Graph Algorithms Appl.*, Vol. 7, pp. 131-140.

[18] K. Erciyes & O. Dagdeviren, (2012) “A Distributed Mutual Exclusion Algorithm for Mobile Ad Hoc Networks”, *International Journal of Computer Networks & Communications (IJCNC)* Vol.4, pp. 129-148.

[19] S. Basagni, (2001) “Finding a maximal weighted independent set in wireless networks”, *Telecommunication Systems*, Vol. 18, pp. 155-168.

[20] M. Luby, (1986) “A simple parallel algorithm for the maximal independent set problem”, *SIAM journal on computing*, Vol. 15, pp. 1036-1053.

[21] D. Mahjoub & D. Matula, (2009) “Experimental Study of Independent and Dominating Sets in Wireless Sensor Networks Using Graph Coloring Algorithms”, *Wireless Algorithms, Systems, and Applications*, pp. 32-42.

[22] S. Parthasarathy & R. Gandhi, (2005) “Distributed algorithms for coloring and domination in wireless ad hoc networks”, *In Proc. of Foundations of Software Technology and Theoretical Computer Science*, pp. 447-459.

[23] C. Gavoille, R. Klasing, A. Kosowski, Ł. Kuszner & A. Navarra, (2009) “On the complexity of distributed graph coloring with local minimality constraints”, *Networks*, Vol. 54, pp. 12-19.

[24] D. P. Dailey, (1980) “Uniqueness of colorability and colorability of planar 4-regular graphs are NP-complete”, *Discrete Mathematics*, Vol. 30, pp. 289–293.

[25] M.E. Dyer & A.M. Frieze, (1989) “The solution of some random NP-hard problems in polynomial expected time”, *Journal of Algorithms*, Vol. 10, pp. 451–489.

[26] J. S. Turner, (1988) “Almost all k-colorable graphs are easy to color”, *Journal of Algorithms*, Vol. 9, pp. 63–82.

[27] D. Eppstein, (2003) “Small maximal independent sets and faster exact graph coloring”, *J. Graph Algorithms Appl.*, Vol.7, pp. 131-140.

[28] A. Björklund, T. Husfeldt & M. Koivisto, (2009) “Set partitioning via inclusion–exclusion”, *SIAM Journal on Computing*, Vol. 39, pp. 546–563.

[29] R. Beigel & D. Eppstein, (2005) “3-coloring in time O(1.3289n)”, *Journal of Algorithms*, Vol. 54, pp. 168–204.

[30] J.M. Byskov, (2004) "Enumerating maximal independent sets with applications to graph coloring", *Operations Research Letters*, Vol. 32, pp. 547–556.
[31] D. J. A. Welsh & M. B. Powell, (1967) "An upper bound for the chromatic number of a graph and its application to timetabling problems", *The Computer Journal*, Vol. 10, pp. 85–86.
[32] D. Brélaz, (1979) "New methods to color the vertices of a graph", *Communications of the ACM*, Vol. 22, pp. 251-256.
[33] K. Kothapalli, C. Scheideler, M. Onus & C. Schindelhauer, (2006) "Distributed coloring in O(log$^{1/2}$ n) bit rounds", *In Proc. of International Parallel & Distributed Processing Symposium*.
[34] J. Schneider & R. Wattenhofer, (2010) "A new technique for distributed symmetry breaking", *In Proceedings of the Symposium on Principles of Distributed Computing.*
[35] N. Linial, (1992) "Locality in distributed graph algorithms", *SIAM Journal on Computing*, Vol. 21, pp. 193–201.
[36] A. Panconesi & A. Srinivasan, (1995) "On the complexity of distributed network decomposition" *Journal of Algorithms*, Vol. 20, pp. 581–592.
[37] F. Kuhn & R. Wattenhofer, (2006) "On the complexity of distributed graph coloring", *In Proc. of 25th ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 7–15.
[38] L. Barenboim & M. Elkin, (2009) "Distributed (Δ+1)-coloring in linear (in Δ) time". *In Proceedings of the 41st Symposium on Theory of Computing*, pp. 111–120.
[39] R. Cole & U. Vishkin, (1986) "Deterministic coin tossing with applications to optimal parallel list ranking", *Information and Control*, Vol.70, pp. 32–53.
[40] A. Goldberg, S. Plotkin, G. Shannon, (1988) "Parallel symmetry-breaking in sparse graphs", *SIAM Journal on Discrete Mathematics*, Vol. 1, pp. 434–446.
[41] J. Schneider & R. Wattenhofer, (2008) "A log-star distributed maximal independent set algorithm for growth-bounded graphs", *In Proc. of 27th ACM Symposium on Principles of Distributed Computing (PODC).*
[42] J. Hansen, M. Kubale, Ł. Kuszner & A. Nadolski, (2004) "Distributed largest-first algorithm for graph coloring", *LNCS*, Vol. 3149, pp.804–811.
[43] M. T.Jones & P. E. Plassmann, (1993) "A parallel graph coloring heuristic", *SIAM Journal on Scientific Computing*, Vol. 14, pp. 654-669.
[44] K. Erciyes, (2007) "A Formal and Practical Method to Develop Distributed and Critical Software (in Turkish)", *In Proc. of National Software Engineering Symposium*.
[45] Synchronous Thread Simulator, http://www.ube.ege.edu.tr/~dagdeviren/source/courses/2012Fall/Distributed_Algorithms/sync_thread_sim.tar.gz.
[46] A. Kosowski & Ł. Kuszner, (2006) "On greedy graph coloring in the distributed model", *In Proceedings of Euro-Par 2006 Parallel Processing*, pp. 592-601.
[47] W. Klotz, (2002) "Graph coloring algorithms", *Mathematics Report*, Technical University Clausthal, pp. 1-9.

Authors

Esra Ruzgar

Esra Ruzgar received the BSc. degree in Computer Eng. from Izmir Institute of Technology and MSc. degree from Ege University, International Computing Institute. She is currently a Ph.D. student in joint Computer Science Ph.D. program between Ege and Izmir University. She is a research assistant in Software Engineering Department of Izmir University. Her research interests are bioinformatics and distributed systems.

Orhan Dagdeviren

Orhan Dagdeviren received the BSc. degree in Computer Eng. and MSc. degree in Computer Eng. from Izmir Institute of Technology. He received Ph.D. degree from Ege University, International Computing Institute. He is an assistant professor in International Computing Institute in Ege University. His interests lie in the computer networking and distributed systems areas. His recent focus is on graph theoric middleware protocol design for wireless sensor networks, mobile ad hoc networks and grid computing.