2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing

A fast heuristic for solving the D1EC coloring problem

Fabio Campoccia
DIEET, Università di Palermo
Palermo, Italy

Vincenzo Mancuso
Institute IMDEA Networks
Madrid, Spain

Abstract. **In this paper we propose an efficient heuristic for solving the Distance-1 Edge Coloring problem (D1EC) for the on-the-fly assignment of orthogonal wireless channels in wireless as soon as a topology change occurs. The coloring algorithm exploits the simulated annealing paradigm, i.e., a generalization of Monte Carlo methods for solving combinatorial problems. We show that the simulated annealing-based coloring converges fast to a suboptimal coloring scheme even for the case of dynamic channel allocation. However, a stateful implementation of the D1EC scheme is needed in order to speed-up the network coloring upon topology changes. In fact, a stateful D1EC reduces the algorithm's convergence time by more than 60% in comparison to stateless algorithms.**

Keywords: Channel assignment, Edge coloring, Simulated annealing.

I. Introduction

The widespread diffusion of 802.11 wireless networks in the last few years fosters the adoption of protocols for the optimization of the narrow radio spectrum utilization. In particular, the main problem that degrades network performance is the interference between concurrent transmissions, i.e., packet collisions. To avoid collisions, it is necessary to coordinate the transmitters within the network to access the wireless media through a multiple access scheme based on time, code or frequency. IEEE 802.11-like WLANs adopt single radio devices and time division multiple access, thus suffering for transmission collisions. In turn, collisions result in very low per-node throughput as soon as a few nodes are in transmission range with each other. However, thanks to the availability of multiple orthogonal channels (e.g., CDMA or OFDMA channels), it is possible to assign dedicated channels to node pairs that would otherwise interfere with each other, so that multiple simultaneous transmissions can successfully occur. Unfortunately, the number of available orthogonal channels is limited by the total available bandwidth and the specific transmission technique. Thereby, it is not always possible to assign a different channel to each different node pair, and often interfering node pairs might share the same channel, thus resulting in possible packet collisions.

In this paper we focus on the problem of assigning orthogonal channels in such a way that the collision probability is minimized. We assume that orthogonal channels are ideal, so that cross-channel interference can be neglected. Thus, we aim at minimizing the number of interfering nodes. To this purpose, we propose to adopt the *Distance-One Edge Coloring* (D1EC) [1].

Assume first that the distance in between two links is the minimum number of hops in between a node in the first link and a node in the other link. Then, given a physical graph G, a selected subgraph $A \subseteq G$, and a set of colors, the Distance-1 Edge Coloring problem consists in finding a mapping between colors and links in A, such that pairs of links in A that are at distance one—with respect to the topology of G—are assigned different colors [1]. We propose a heuristic for D1EC, which uses a centralized algorithm and exploits a global knowledge of the network topology and the node activity. In our work, it is assumed that (*i*) a wireless network can be represented by means of a graph, and (*ii*) the channel assignment problem can thus be formulated as a graph coloring problem [2]. The proposed channel assignment algorithm uses a combinatorial method called Simulated Annealing [3]. In particular, we propose a stateful approach to further improve the coloring algorithm performance by running the coloring scheme from an initial channel assignment whose cost is near to the minimum. We developed a Java simulator to validate our proposal. The major advantage with respect to other proposals is that we provide an easy and concrete approach for practical implementation of coloring algorithms.

The rest of the paper is organized as follows. Section II reviews the related work. Section III introduces network coloring and simulated annealing concepts. Section IV presents our novel heuristic for D1EC. Section V shows how to setup the parameters of the coloring algorithm. In Section VI we report simulation results that validate our heuristic and show its potentiality. Section VII concludes the paper.

II. Related work

The classical graph coloring methods suffer from three major problems. First, they need a hard decision that indicates whether the use of the same channel by two radio cells is allowed or not. This decision is based on interference level but it is questionable since it depends on a high amount of uncertainty (e.g., spatial distribution of the traffic) [4]. Second, if a channel assignment is not contention-free, then there is no way to define in algorithm which constraint is preferable to violate (e.g., based on some link priority) [5]. Third, the graph theoretic approach only aims to minimize the usage of spectrum, even if there are several ways to use the assigned bandwidth [6]. An interesting alternative solution is to define the channel assignment problem as a cost function optimization problem. This allows to use a generic discrete optimization method. The neural network approach of [7] and [8] was shown to be inappropriate for channel assignment [9], as it yields bad solutions even in

978-0-7695-4322-2/10 $26.00 © 2010 IEEE
DOI 10.1109/EUC.2010.71

simple cases, e.g., it can favor suboptimum channel assignments. Simulated annealing, instead, has been successfully applied to different practical planning situations yielding good results [10]. In [11], the authors formulate an integer linear programming problem to optimize the channel assignment for hot-spot service areas. However, linear programming is generally slower than Monte Carlo and simulated annealing methods, even though it permits to reach the optimum.

Some works analyze algorithms or mathematical models in order to execute the channel assignment in a dynamic way [12], [13]. In those papers, it is fundamental the method that assigns the channels in order to minimize the contention in as short time as possible. In fact it is mandatory that the time to reconfigure all link channels is shorter than a critical threshold, in order not to decrease network performance. In particular, channel assignment can be executed locally at each node, based on the local interference experienced by the node. Thus, any transceiver can select its own channel by just listening to its air interface. This approach is called TBSC (Transceiver Based Channel Selection) [14]. Unfortunately, TBSC permits the optimization of single hop links only, while no optimal allocation is ensured for multihop topologies [1]. Furthermore the channel selection mechanism depends on the accuracy of the state of the available channels. Another way to execute a distributed channel assignment is the AMCP (Asynchronous Multichannel Coordination Protocol) [15], which uses a dedicated control channel on which nodes contend to reserve data channels by exchanging RTS/CTS packets according to 802.11 DCF. Nevertheless, corrupted and inaccurate channel availability state leads to frequent data packet collisions or unnecessary waiting on the control channel while data channels are free. Consequently, AMCP does not efficiently utilize the number of channels.

III. Basic techniques

In this section some basic definitions about graph coloring techniques are reported. The section also describes how these techniques can be applied for network coloring, e.g., for channel assignment in wireless networks. We will also introduce the concept of simulated annealing [3] that will be used to implement the selected coloring technique.

A. Network coloring definitions

In order to introduce the use of graph coloring schemes for network coloring purposes, we first formally define the network coloring and the distance between network links.

Definition 1: Let $G=(V, E)$ be a graph. A k-coloring for G is a function $f: E \rightarrow [k]$ such that:

$$f(e_1) \neq f(e_2), \forall (e_1, e_2) \in E \times E .$$

In other words, a k-coloring is an assignment of edges to k colors. We say that a graph G is k-colorable if there is a k-coloring for G. The chromatic number of G is the least k such that G is k-colorable. Given a k-colorable graph G, finding a k-coloring for G is solvable in polynomial time for $k = 2$, but NP-hard for $k \geq 3$ [16].

Because of the graph coloring problem in its classical formulation is NP-complete [17], the channel assignment problem is also NP-complete, and therefore an optimal assignment cannot be found in polynomial time.

Several approaches were developed in order to define a channel assignment problem as a graph coloring problem. The graph coloring approach was applied both to cellular radio networks [18], [19], and in IEEE 802.11-based mesh networks [20], [21].

Definition 2: Given a graph G, and two nodes u and v contained in G, a link $l_{u,v}$ is defined as the edge contained in G that connects node u to node v. Let the distance $d(u_1, u_2)$ between two nodes u_1 and u_2 in a graph G be the minimum number of hops in G from u_1 to u_2. Then let the distance between two links $l_{u1,u2}$ and $l_{v1,v2}$ be:

$$d(l_{u_1 u_2}, l_{v_1 v_2}) = \min_{i,j \in \{1,2\}} d(u_i, v_j)$$

Here, it is assumed that zero-distance between two links means that the two links share a common node; distance equal to one between two links means that there is at least one edge e in the graph G that connects one node of the first link to one node in the second link, hence simultaneous transmissions would interfere. Distances greater than one mean that no edge e in G exists that connects a node in the first link to a node in the second link. Hence, if the distance between two links is greater than one, then both links can transmit on the same channel simultaneously.

With the previous definition of link distance, only links that are at distance one should be assigned different channels in order to attain a contention-free channel assignment. In the following we review some notable network coloring schemes that use graph coloring techniques.

B. Channel assignment schemes

In [1] the following "distance one channel assignment" definition is proposed:

Definition 3: "Given a physical graph G and a selected subgraph $A \subseteq G$, the Distance-1 Edge Coloring (D1EC) problem seeks a mapping of colors to links in A such that any two links in A that are at distance one with respect to G are assigned different colors" [1].

If a valid D1EC is realized, any selected set of links in A that do not share a node in common can be active at the same time and hence, in terms of channel assignment, maximum throughput is achieved. However, due to the limited number of channels, it can happen that there is no channel assignment that realizes a valid distance one channel assignment. In this case, distance one links with same channel will interfere with each other. By means of the algorithm reported in [1], however, the amount of interference on each active link, and hence the load between channels, can be balanced by minimizing the number of interfering links with each active link. The approach described in [1] allows a suitable solution for the static channel assignment, i.e., the channel assignment is calculated at network start-up and it is kept unchanged.

C. Simulated annealing

Simulated annealing is a general method for the approximate solution of difficult combinatorial optimization problems. It was originally proposed by Kirkpatrick et al. [3] and Cerny [22]. The algorithm can be viewed as a simulation of the physical annealing processes found in nature.

Generally, a combinatorial optimization problem consists of a set S of configurations or solutions, also known as "conformations", and a cost function K which determines for each configuration s the cost $K(s)$. With a local search, it is necessary to know the neighbor s' of each solution s, in order to define a neighborhood structure N on S. $N(S)$ determines, for each solution s, a set of possible transitions which can be proposed when the system is in s. At each step, a neighbor s' of s is proposed at random. Conformation s is replaced by s' if its cost $K(s')$ is lower than $K(s)$. To avoid terminating the search in a local minimum, simulated annealing occasionally accepts a solution of higher cost according to the so-called Metropolis criterion [23]: if the cost difference $\Delta K = K(s)-K(s')$ is negative then the neighbor's conformation is accepted and it becomes the new conformation in the chain. If $\Delta K > 0$, however, a random number R between 0 and 1 is generated according to a uniform distribution, and the resulting conformation is accepted if $e^{-\Delta k/C} > R$. The parameter C is a constant called control parameter. It regulates the percentage of new configurations that are accepted in case ΔK is positive. A cooling process in simulated annealing applications is emulated by decreasing C progressively from C_0 to C_f through a cooling function $f(C) = u\ C$.

IV. Channel assignment through efficient simulated annealing

Wireless network topologies can vary over time, so that a dynamic channel assignment might be needed as soon as a node joins or leaves the network. We define active network topology the subnet that only contains nodes that are sending and receiving packets, and all edges that connect these nodes. It is clear that if the active network changes, also interfering links change, and a new channel assignment might be needed. Furthermore, static coloring of the entire network graph might results in using an unnecessary high number of colors to optimize the channel assignment to edges that will not really contend for transmission most of the time. In practice, with a dynamic coloring scheme, channel assignment could be performed online, e.g., when the set of active nodes changes, or on a fixed schedule.

Therefore, our study is twofold. First, we develop a simulator that, given a network subgraph, finds the solution of the D1EC coloring problem by means of the simulated annealing method. We will demonstrate that simulated annealing converges to a solution faster than not-combinatorial methods. Results are compared with theoretic bounds reported in [1]. Second, we introduce a stateful dynamic channel assignment algorithm based on the assumption that the active topology changes slowly. Thereby, upon a topology change, the current channel assignment is likely to be close to the optimal channel assignment for the new topology.

A. D1EC coloring via simulated annealing

In what follows we show that dynamic channel assignment can be performed online, as soon as the traffic matrix changes. Furthermore, online coloring can be performed faster than off-line by initializing the algorithm, at each update, with the last utilized coloring assignment.

Simulation goal. D1EC does not specify the algorithm to be used. In [1], the authors propose a heuristic solution for distance one coloring problem (proving it is NP-complete). In [24] authors analyze an alternative algorithm which relies on the representation of each vertex of the graph by a vector of its coordinates in each dimension. The second method is more general (resolve k-coloring problem for all $k \geq 1$), while the first method uses a faster algorithm. In our implementation of D1EC, the color assignment is executed by means of simulated annealing. Adopting the simulated annealing simulation requires to accurately tuning the algorithm parameters, with the aim of minimizing the run time while achieving a contention degree close to the minimum. The contention degree represents the cost in the Metropolis criterion.

Simulator. We developed a Java simulator that can be used to evaluate the coloring scheme, as well as to design and test the simulated annealing parameters. Tests executed by means of the java simulator have been applied to a regular graph and to an irregular graph, extracted from the 802.11 network deployed in Chaska (Minnesota), one of the first U.S. cities to offer WiFi to almost all of its residents by implementing a civic 802.11 open network. The simulator code is based on four main java methods.The g*raph initialization* method builds the network graph. The *cost calculation* method calculates the graph cost in D1EC. The minimum cost depends on the number of colors (i.e., orthogonal channels) available, and it can be as low as zero. In particular, the *cost calculation* method sums, per each edge, the number of distance-one edges that have the same color. The g*raph perturbation* method: allows to introduce a random perturbation in the simulated annealing. Finally, the e*quilibrium condition* method tests, at each iteration, the ratio between the number of new accepted configurations and the refused ones. If the ratio is below a preconfigured threshold, a new equilibrium is reached. The following main java class utilizes the four method introduced above:

MAIN JAVA CLASS:

1. Graph initialization: same color is assigned to all edges;
2. Until $C > C_f$ repeat:
    a. Until (Equilibrium condition) repeat:
        i. Update color distribution: New colored graph configuration = Graph perturbation method (old configuration);
        ii. Update graph cost: New cost = Cost calculation method (new colored graph configuration);
        iii. Apply the Metropolis criterion to new cost, based on current control parameters;
        iv. Accept or refuse the new configuration.
    b. Update $C := u\ C$.

B. Dynamic D1EC

We define a study period T in which a given graph network is monitored. Not all network nodes transmit during the entire study period, so it is possible to select a subset of nodes that are exchanging data during a subinterval τ_i of T. These nodes are defined as "active nodes" for the i-th "check period" τ_i. The topology of active nodes is a function of time, but it can be seen as fixed within a check period. In a similar way, the list of active edges in a time interval is defined as the list of network edges that are expected to be used for one or more packets during that time interval. Also the subset of active edges changes as function of time, but it is fixed within a check period. Note that if a connection begins or terminate during the check period, we consider all nodes and edges involved in the connection as active for the entire duration of the check period. For each τ_i, the following definition applies:

Definition 4: Given a graph $G = (E,V)$, denoting by $e \in E$ the graph edges, and by $(v_1, v_2) \in V \times V$ the vertex pairs in which v_1 and v_2 are connected by e, the sub-graph $H\tau_i$ is defined as follows:

$$H_{\tau_i} = \left\{ \left((v_1, v_2), e \right) : e \text{ is active in } \tau_i \right\}.$$

$H\tau_i$ is a graph extracted from G, containing only active nodes and active edges for the check period τ_i. Based on the definition above, in the simulation section we show that: (*i*) D1EC applied to a smaller graph converges faster than if it is applied to a bigger graph; (*ii*) D1EC that starts from a sub-optimal solution converges faster than starting from a non-colored graph; (*iii*) consequently, if $H\tau_i$ is strictly contained in G, then D1EC operation applied to $H\tau_i$ converges faster than if it is applied to G.

Mandatory requirement for D1EC is the rapid convergence of the coloring algorithm. So, a "stateful Initial condition" feature is proposed in order to attain a high performance algorithm. The definition of such a feature will be detailed to next paragraph, but in few words it allows that: at every check period, the coloring algorithm begins its search for optimal assignment by using the same colors as in the previous check period for all edges that remained active.

Stateful initial condition. Given a network topology, it is possible to define a network state at each time instant t. The state contains the list of active edges and their colors at time t. At each check period, the list of active nodes and edges is updated and a new D1EC coloring is calculated. The D1EC coloring algorithm we have introduced so far is meant to run at the beginning of each new check period, and to use as initial coloring conditions the same color for all edges. Thereby, the described algorithm has to be considered stateless, because of initial condition is always same. We propose an algorithmic improvement: as a new check period begins, the algorithm is run by using as initial coloring the color distribution resulted from the previous check period. Additionally, new active nodes and edges are added with a same default color, and nodes and edges not anymore active are simply removed. We call this new approach stateful, since the initial condition is inherited from the previous check period. In the following we describe the procedures executed by our stateful D1EC algorithm when the system boots and at the beginning of each check period.

Start-up procedure. This procedure is executed at network booting in order to test the stateful algorithm, starting from an empty topology. All active node pairs are added step by step, as described in the following:

START UP PROCEDURE:

1. At initial time t_0, it is estimated which stations have to transmit during the first check period $[t_0; T_0]$, with $T_0 = (t_0+T)$, and which path in the graph has to be active (that is, the list of edges that will forward data):
   N_0 = {list of n_0 node pairs that will exchange data in T_0 };
   P_0 = {list of n_0 paths that link each node pair in N_0, calculated with the Dijkstra algorithm};
   H_k = subgraph containing the first k active paths. H_0 is set as empty graph.
2. In interval $[t_0; T_0]$ the following cycle is executed:
   a. for (k=1 to n_0) path k contained in P_0 is added to H_k;
   b. for each iteration, calculate D1EC for H_k considering as initial condition H_{k-1}.

Running procedure for the stateful algorithm**.** This procedure is executed at each check period using as initial state the distribution used in the previous check period.

RUNNING PROCEDURE:

1. At generic initial time t_i, it is estimated which stations have to transmit during interval $[t_i; T_i]$, with $T_i = (t_i+T)$, and which path in the graph has to be active (that is list of edges that will forward data):
   N_i= {list of n_i node pairs that will exchange data in T_i };
   P_i= {list of n_i paths that link every node pair contained in N_i, calculated with the Dijkstra algorithm};
   H_i = subgraph containing the active paths at time Ti.
2. Between time t_i *and* (t_i+T), the D1EC is calculated for H_i, considering as initial condition H_{i-1}.
3. Running procedure is executed every T seconds.

V. D1EC-SUITABLE SIMULATED ANNEALING

In this section we compare some perturbation methods that can be used by means of simulated annealing. We show how to setup simulator's parameters in order to obtain a fast convergence to a suboptimal coloring conformation which is close to the optimal one. Parameter tuning depends on the network topology, so in order to generalize our study we classify network topologies in two broad classes: regular graphs (grids with a defined multiplicity) and non-regular graphs (constituted by several nodes groups linked each other by some gateways); then, in our simulations, we consider two topologies representing the two classes. The first topology is an extraction from the topology of Chaska's 802.11 network (Figure 1), and contains 45 vertex and 75 edges. The second topology is a rectangular 5x10 grid (Figure 2).

Perturbation method. First, here we validate the need for a perturbation method and we test some alternative methods that could be used for simulated annealing.

A perturbation method has to be chosen in order to attain the fastest convergence in terms of number of iterations that allow approximating the minimum graph cost. As baseline, we consider an algorithm that does not utilize the Metropolis

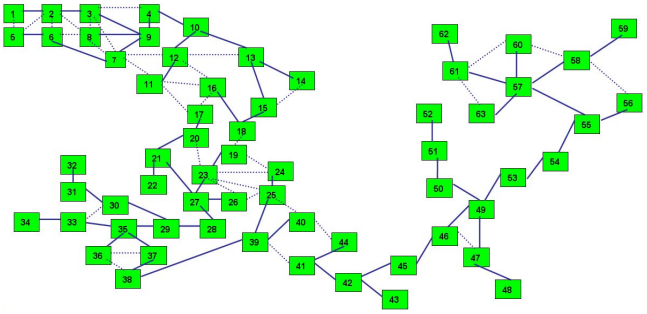

Figure 1. Graph obtained from the Chaska network.

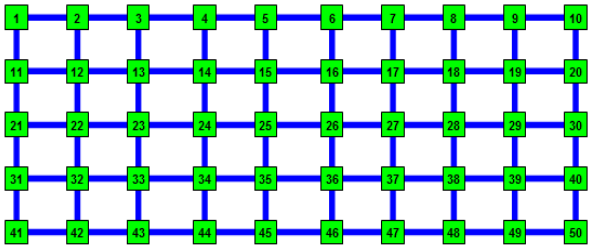

Figure 2. Rectangular grid topology.

criterion, and hence it does not perform a legacy simulated annealing. For this reason we call this algorithm "not simulated annealing" (NOT-SA). Then we compare the results obtained with NOT-SA with the results obtained with simulated annealing when using two different perturbation methods (we name the resulting algorithms as SA-A and SA-B, respectively). In particular, the three perturbation methods are as follows:

- *NOT-SA* perturbation method: (*i*) choose an edge at random, (*ii*) assign a color at random to that edge, and (*iii*) accept the new configuration only if the new cost is lower than the previous.
- *SA-A* perturbation method: (*i*) choose an edge at random, (*ii*) assign a color at random to the selected edge, and (*iii*) accept the new conformation only if the new cost satisfies the Metropolis criterion.
- *SA-B* perturbation method: (*i*) select all edge pairs that have the same color and whose distance is one, (*ii*) assign a color, chosen at random from the available color list, to each selected edge, and (*iii*) accept the new configuration only if the new cost satisfies the Metropolis criterion.

We now simulate the behavior of the perturbation methods under the following parameter settings (we will explain the parameter choice in the next subsection): C_0 is set to 4 for the Chaska topology, and 16 for grid topology; C_f is 0.1, and the number of available colors is three. We first define the maximum graph cost (100%) as the cost of a graph in which all edges use the same color. In Figures 3 and 4 we normalize the graph cost to the maximum, and report the trend for the relative graph cost as a function of the number of iterations for both the Chaska topology and the regular grid.

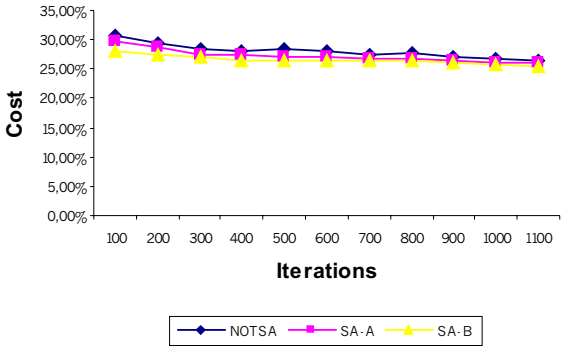

Figure 3. Cost vs. iterations (Chaska topology).

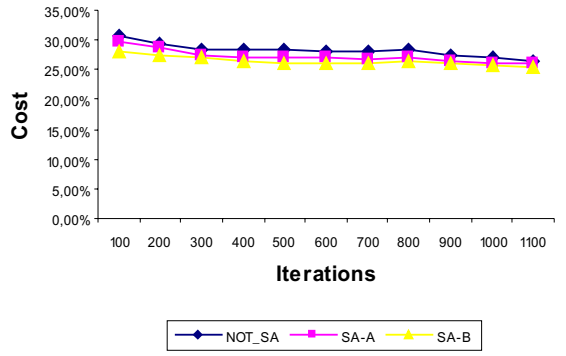

Figure 4. Cost vs. iterations (Regular grid).

As illustrated in [1], three colors are not enough to color a graph with more than three edges and reach zero cost. However, the results depicted in Figures 3 and 4 show that the simulated annealing mechanism is able to reach a significant cost reduction within few iteration. The impact of the perturbation algorithm is not dramatic, even when the Metropolis criterion is not adopted. However, in the experiments of Figures 3 and 4, as well as in others not reported here, SA-B performs slightly better than the other perturbation methods. Thereby, we decided to adopt SA-B for the evaluation tests reported in this paper.

Control parameters selection. The control parameter C regulates the performance of Metropolis criterion-based perturbation methods. Simulated annealing algorithms allows C to be updated at each iteration according to a function $f(C) = uC$, *i.e.,* C evolves at each step as $C_n = uC_{n-1}$. The algorithm starts from $C = C_0$ and ends when $C \leq C_f$. In this way, the control parameter C regulates the transition rate towards higher cost conformations. In particular, C_0 is critical in order to engage the simulating annealing process, while C_f allows to define a final condition for the algorithm. The following empiric rule has been proposed for selecting C_0: choose a small random positive value and run the simulated annealing, then duplicate C_0 and restart the simulation. Continue doubling C_0 until the number of accepted transitions decreases to less than 80% [3]. In the case of the Chaska graph and the regular grid of Figures 1 and 2, we found that the accepted transition rate reaches

80% as soon as $C_0 = 4$ (Chaska) and $C_0 = 16$ (regular grid). Hence we use these values in the rest of the paper. In order to find the correct value for C_f we set the color number to 17 or more for the Chaska topology, and to 15 or more for regular grid. With this numbers, a zero cost conformation exists for both the regular and irregular graphs under evaluation. Thus we study the convergence of the algorithm to zero cost, estimate the number of iterations required for convergence, and verify whether C_f allows to effectively converging. In fact, if control parameter C would reach C_f in too little iteration, algorithm might not converge to the minimum cost [3]. On the other side, if control parameter C needs too much iteration to reach C_f, it is possible that the cost reaches a steady state in which it oscillates [3]. Then, several further iterations would be of no use. It is important to set the C_f value to limit both events.

Figures 5 and 6 report the number of convergence iterations as a function of the number of available colors for the Chaska topology and the regular grid, respectively. Different values were tested for C_f values, ranging from 0.001 to 1. Note that C_f=1 did not allow converging to zero cost. Thus, the line for C_f=1 in Figures 5 and 6 express the convergence time to a suboptimal cost. Instead, using C_f <1, the algorithm converges to zero cost. In particular, the case C_f = 0.1 converges faster than C_f = 0.01 and C_f = 0.001, so we select C_f = 0.1 for the experiments reported in the evaluation section. Finally, as for the evolution function $f(C) = uC$, the authors of [3] suggest to use a coefficient u smaller than 1 but very close to 1. We set u = 0.95.

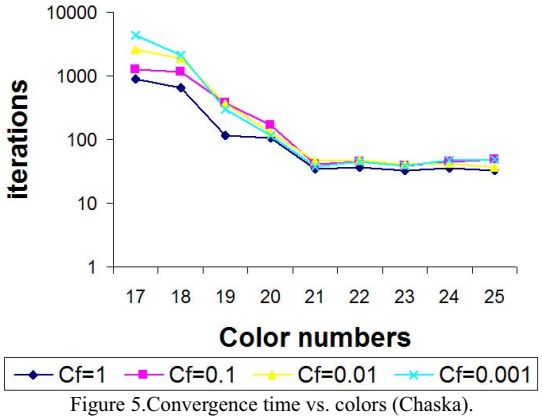

Figure 5.Convergence time vs. colors (Chaska).

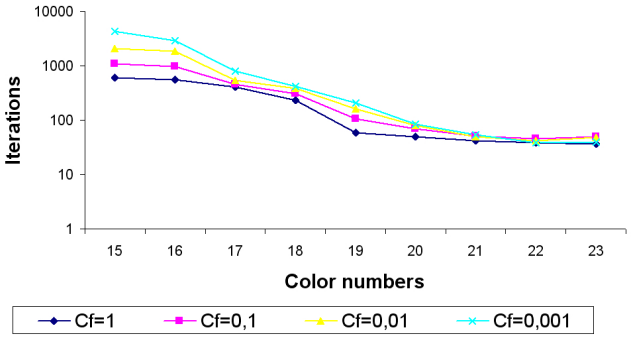

Figure 6. Convergence time vs. colors (Regular grid).

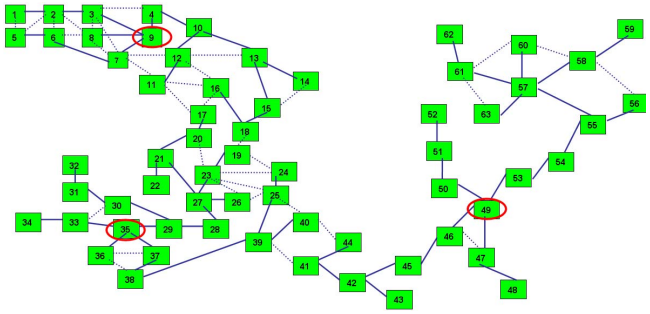
Figure 7. Chaska Topology with gateways.

VI. Evaluation

In this section we evaluate by simulation the dynamic D1EC described in section III. All tests reported here were repeated for 10 times, and results are shown in terms of average statistics. The target graph A was an extraction of the Chaska topology (as shown in Figure 7), obtained by (*i*) selecting a few gateway nodes (*GW*s, circled nodes in Figures 7, 9, 10, and 11), and (*ii*) assuming that the traffic between two nodes is allowed only if at least one of the nodes is a gateway. These assumptions are realistic in normal network implementations. In fact, adopting a limited number of gateways to interconnect the mesh network to the external network, allows to separate the entire network in sub-areas, each managed by a single gateway, and so it is easier to control the traffic load, to manage fault and eventually redirect data, to execute the billing and rearrange all the topology when necessary by changing the gateway distribution. With the assumption reported above, the potentially active links in the Chaska topology are individuated by means of the following procedure: first, three gateways are chosen, and for each node n that is not a *GW*: (*a*) the shortest path between n and the nearest *GW* is calculated, and (*b*) all the edges contained in that shortest path between n and its *GW* are marked as potentially active. The graph reported in Figure 7 shows the potentially active links with solid lines, and the inactive links with dashed lines. The nodes inside red circles are the gateways.

For the evaluation, we first consider a stateless version of the algorithm, and then we show that the stateful algorithm is twice as fast as the stateless one. In order to compare stateful and stateless algorithm, it is preferable to define a common convergence condition that it is always reached. In the following tests, the convergence target is zero cost with three colors available.

Evaluation of the stateful algorithm. For the evaluation of the stateful algorithm, the initial graph is composed by only the three gateways and no edges. Then, one after the other, all nodes are added in a random order, together with the links belonging to the shortest path between the node and one of the available gateways. We compute the new D1EC coloring every time a node is added, using as initial coloring the color assignment computed at the previous step. At each step, we count the number of iterations that allow convergence for the new graph coloring.

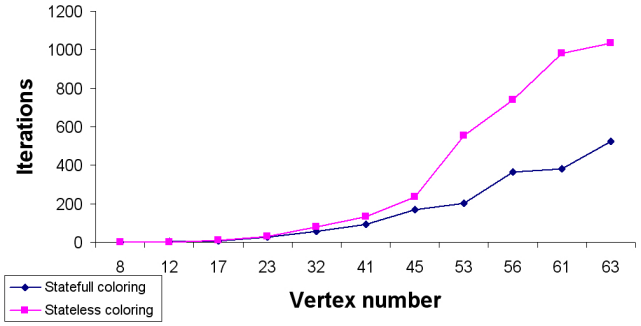

Figure 8. Comparison between stateful and stateless procedures.

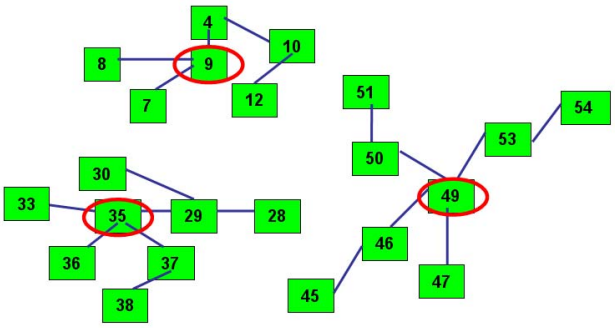

Figure 9. Graph constituted by 30% of nodes.

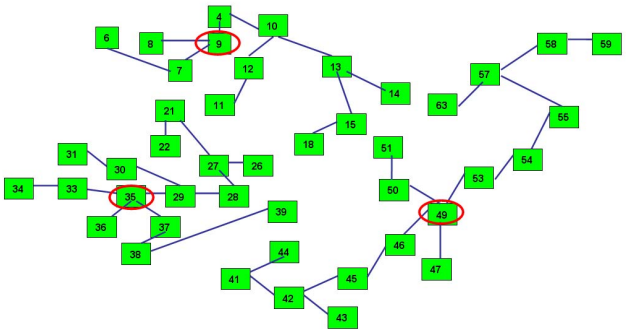
Figure 10. Graph constituted by 70% of nodes.

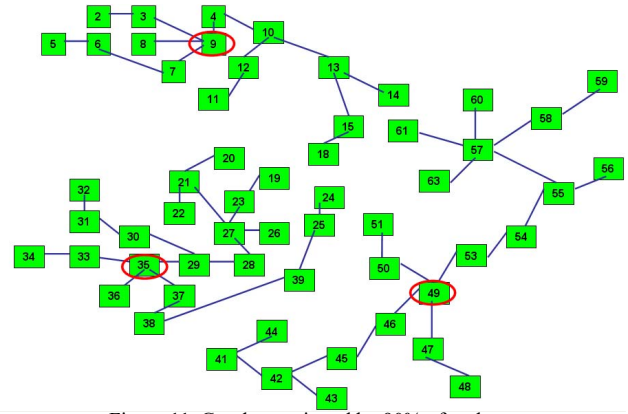
Figure 11. Graph constituted by 90% of nodes.

Evaluation of the stateless algorithm. As for the evaluation of the stateless algorithm, the initial graph is composed by only the three gateways and no edges as well. Then the non-gateway nodes are added in random order, and every time a node is added, the shortest path to its nearest gateway is added too (including links and intermediate nodes). Upon adding a node and the path to its gateway, the resultant graph is colored from scratch, i.e., starting with all links having the same color. At each step, the number of iterations that bring to convergence is calculated.

Algorithm comparison. In Figure 8 we report the comparison between the performance of the stateless and stateful algorithms for the network of Figure 7. Note that the stateful algorithm converges to zero cost in almost half time with respect to the stateless one. This means that keeping memory of the last coloring schema allows halving the re-coloring time of the network after a new path is added to the active network topology.

In a second group of experiments, we repeat our algorithm comparison by limiting the total number of nodes in the active subnetwork to about 30%, 70%, and 90% of the total nodes in the original Chaska topology. However, in these experiments, the chosen nodes are uniformly spread around the three gateways. In Figures 9 to 11 we show the network topologies used here. Tables I to III show the number of iterations needed for solving the D1EC coloring problem when we add the last-but-one and the last node of the series. This situation reflects the common case in which, during the real network operation, a new single transceiver joins a fully operative network. It is interesting to observe that also in this case stateful algorithm converges in half the number of iterations needed by the stateless algorithm.

TABLE I. SINGLE NODE ACTIVATION WITH 30% OF NODES

Statefull algorithm		**Stateless algorithm**	
Nodes Number	***Iteration num***	***Nodes Number***	***Iteration num***
19	11	19	13
20	19	20	26

TABLE II. SINGLE NODE ACTIVATION WITH 70% OF NODES

Statefull algorithm		**Stateless algorithm**	
Nodes Number	***Iteration num***	***Nodes Number***	***Iteration num***
44	98	44	195
45	87	45	218

TABLE III. SINGLE NODE ACTIVATION WITH 90% OF NODES

Statefull algorithm		**Stateless algorithm**	
Nodes Number	***Iteration num***	***Nodes Number***	***Iteration num***
57	283	57	792
58	301	58	815

The gain offered by the stateful algorithm is particularly evident in the case of dense networks. In fact, to add the 20th node to the network of Figure 9, with only a mere 30% of nodes active in the overall network, the stateful algorithm reaches zero cost in 19 iteration while the stateless algorithm needs 26 iterations, thus yielding a time saving of more than 25%. Adding the node number 45 in the network of Figure 10 (with a total of 70% nodes out of the overall network population), the stateful algorithm reduces the number of iteration from 218 to 87, i.e., 60% less iterations. Finally, for the network of Figure 11 (90% of the total number of nodes), the last node can be added and the network colored with 301 iterations using the stateful algorithm, and 815 using the stateless. The saving is then equal to 63%.

VII. Conclusions

In this work we proposed a channel assignment algorithm based on a graph coloring mechanism. Channel assignment, e.g., frequency or code assignment, can be operated by means of simulated annealing. Besides, we adopted as performance metric the number of iterations needed for the algorithm to reach a coloring conformation that is close enough to the optimal (e.g., no more than a few percents higher that optimal). We demonstrated that, if correctly tuned, simulated annealing allows a solution which is suboptimal with respect to non-combinatorial algorithms proposed in the literature, but more practical to implement and running much faster. Besides, we introduce a state in the algorithm, which allows running the coloring scheme starting from an initial channel assignment whose cost is near to the minimum possible cost. We showed that, by using the stateful algorithm in dynamically changing topologies, coloring dense networks takes up to 63% less time than by using stateless algorithms.

References

[1] E. Aryafar, O. Gurewitz, E. W. Knightly, "Distance-1 Constrained Channel Assignment in Single Radio Wireless Mesh Networks." In Proc. of IEEE INFOCOM 2008, Phoenix, AZ, April 2008.

[2] W. K. Hale, "Frequency assignment: Theory and applications." In Proc. IEEE, vol. 68, pp. 1497–1514, Dec. 1980.

[3] S. Kirkpatrick, C. D. Gelatt Jr., M. P. Vecchi, "Optimization by simulated annealing," Science, vol. 220, no. 4598, pp. 671-680, May 13, 1983.

[4] A. Gamst, W. Rave, "On frequency assignment in mobile automatic telephone systems." In Proc. GLOBECOM, November-December 1982, pp. 309-315.

[5] A. Gamst, "A resource allocation technique for FDMA systems." Alta Frequenza, vol. LVII, pp. 89-96, Feb.-Mar. 1988.

[6] W. K. Hale, "Frequency assignment: theory and applications." In Proc. IEEE, vol. 68, pp. 1497-1514, 1980.

[7] D. Kunz, "Channel assignment for cellular radio using neural networks." IEEE Transactions on Vehicular Technology, vol. 40, pp. 188-193, Feb. 1991.

[8] J. J. Hopfield and D. W. Tank, "'Neural' computation of decisions in optimization problems." Biol. Cybern., vol. 52, pp. 141 -152, 1985.

[9] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon, "Optimization by simulated annealing: an experimental evaluation, part II (graph coloring and number partitioning)." Operations research, Vol. 39, No. 3, May-June 1991, pp. 378-406.

[10] M. Duque-Anton, D. Kunz, and B. Ruber, "Channel assignment for cellular radio using simulated annealing." IEEE Trans. Vehicular Technology, vol. 42, no. 1, February, 1993.

[11] Y. Lee, K. Kim, and Y. Choi, "Optimization of AP placement and channel assignment in wireless LANs," LCN 2002. 27th Annual IEEE Conference on Local Computer Networks, pp. 831–836, November 2002.

[12] R. Akl and A. Arepally, "Dynamic Channel Assignment in IEEE 802.11 Networks." In Proc. of IEEE Portable 2007: International Conference on Portable Information Devices, March 2007.

[13] E. Garcia Villegas, R. Vidal Ferro, J. Paradells Aspas. "Implementation of a distributed dynamic channel assignment mechanism for IEEE 802.11 networks," In proceedings of the 16th International Symposium on Personal, Indoor and Mobile Radio Communications, PIRMC 2005, pp. 1458-1462, Sept. 11-14, 2005.

[14] J. So, N. Vaidya, "Multi-Channel Mac for Ad Hoc Networks: Handling Multi-Channel Hidden Terminals Using a Single Transceiver." In Proc. of ACM MobiHoc, 2004, pp 222-233, Tokyo, Japan, 2004.

[15] J. Shi, T. Salonidis, and E. W. Knightly, "Starvation Mitigation Through Multi-Channel Coordination in CSMA Multi-hop Wireles Networks." In Proc. of ACM MobiHoc, 2006. Florence, Italy, May 22-25, 2006.

[16] C. C. Hsu, P. Liu, D. W. Wang, J. J. Wu, "Generalized Edge Coloring for Channel Assignment in Wireless Networks." In Proc. of the International Conference on Parallel Processing ICPP, pp. 82 – 92, 2006.

[17] M. R. Garey and D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness." San Francisco, CA: W.H. Freeman, 1979.

[18] R. Battiti, A. Bertossi, D. Cavallaio, "A Randomized Saturation Degree Heuristic for Channel Assignment in Cellular Radio Networks." IEEE Transactions on Vehicular Technology, Vol. 50, issue 2, 364-374, 2001.

[19] M. I. Islam, A. B. M. Siddique Hossain, "Channel Allocation of Mobile Cellular Network Based on Graph Theory." In Proc. of IEEE TENCON 2004, Chiang Mai, Thailand, 21-24 November 2004.

[20] J. Riihijarvi, M. Petrova, P. Mahonen, "Frequency Allocation for WLANs Using Graph Colouring Techniques." In Proceedings of the Second Annual Conference on Wireless On-demand Network Systems and Services, pp. 216-222, January 19-21, 2005

[21] C. L. Barrett, V. S. Anil Kumar, M. V. Marathe, S. Thite, G. Istrate, "Strong Edge Coloring for Channel Assignment in Wireless Radio Networks." In Proc. of the 4th annual IEEE international conference on Pervasive Computing and Communications Workshops, PERCOMW, page 106, 2006.

[22] V. Cerny, "Thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm." J. Opt. Theory Appl., vol. 45, pp. 41-51, 1985.

[23] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, "Equation of state calculations by fast computing machines." Journal of Chemical Physics, vol. 21, pp. 1087-1092, 1953.

[24] G. Fertin, E. Godard, A. Raspaud, "Acyclic and k-distance coloring of the grid." In Information Processing Letters, vol. 87, pp. 51-58, 2003.