Algorithms for Self-Organized Resource Allocation in Wireless Networks

Matti Peltomäki, Juha-Matti Koljonen, Olav Tirkkonen *Member, IEEE*, and Mikko Alava

***Abstract*—We consider the problem of allocating orthogonal resources in a self-organized manner on conflict graphs that describe discretized interference couplings of wireless networks. The target is to find a global optimum, i.e., a conflict-free allocation. We consider both random planar and nonplanar graphs that result from a path-loss model in a wireless network. Two algorithms for the self-organized coloring of arbitrary graphs are devised, and their performances are compared with each other and a rule-based reasoning algorithm that is known from the literature. The semigreedy distributed local search (SDLS) algorithm, which is a particularly simple algorithm that is proposed here, is shown to outperform other algorithms in several cases. In a cellular system setting, we consider a negotiated worst coupling procedure to symmetrize the interference coupling between cells, targeting an improvement of the channel quality of cell-edge users. We compare this approach with an interference coupling based on the path loss that was experienced between base stations and see significant gains in cell-edge performance. In addition, we compare the channel quality that was experienced by users in a cellular network that employs SDLS with the channel quality in a network that employs an interference-reducing network algorithm which finds a local optimum in terms of real-valued interference couplings. In some cases, attempting global optimization based on a conflict graph interpretation outperforms local real-valued optimization.**

***Index Terms*—Cognitive networks, frequency assignment, distributed algorithms, self-organization, trees (graphs)**

I. Introduction

The allocation of orthogonal resources, such as channels for Frequency Division Multiple Access (FDMA) or time slots for Time Division Multiple Access (TDMA), is an essential component of wireless systems. In second generation (2G) cellular systems such as GSM, FDMA and TDMA are used to separate users in a cell, and Inter-cell Radio Resource Management (RRM) is based on orthogonal channel allocations with static reuse planning [1]. The simplest of the classical frequency assignment problems considers the allocation of one channel to each cell, with interference between the cells modeled by a conflict graph [2], [3]. This is essentially a graph coloring problem [4], where the colors are the available channels. A more involved problem is multicoloring where the number of channels is larger, and the cells may have different load and accordingly need different numbers of channels. For evolved 2G systems [5], [6], dynamic frequency assignment problems, reacting to fluctuations in load, were investigated in the field of Dynamic Channel Allocation (DCA). In 3G and evolving 4G systems, such as UTRA Long Term Evolution (LTE) [7], physical layers with higher tolerance for intermittent interference have lead to the prevalence of randomized inter-cell RRM [8], [9] with no need for orthogonalization between cells. These systems perform best under “reuse 1” in time and frequency.

Recently, channel allocation has earned renewed interest due to the active research on cognitive radios and networks [10], [11], and the related spectrum sharing problem. Another important contemporary trend in wireless networking is self-organization [12], [13], where the network nodes perform network operations autonomously in a bottom-up manner, without centralized control. Such functionalities are essential in situations where centralized network elements are impossible, e.g. in sensor and ad hoc mesh networks, or where they are not desired, which is one alternative in the Self-Organized Network (SON) study performed in the LTE context [14]. Self-organized dynamic spectrum sharing, generalizing the principles of dynamic channel allocation, lies at the intersection of these developments, and can be seen as a fundamental cognitive networking function.

In the context of next generation cellular networks, self-organizing channel allocation has become relevant in local area scenarios. With uncoordinated deployments of low-power “femto” base stations, interference becomes worse than in coordinated deployments, and inter-cell RRM with orthogonal channel allocation becomes a viable option also for LTE-based systems [15].

Self-organized channel allocation was first investigated in the DCA literature [16], [17]. The distributed multicoloring algorithm based on carrier sensing in [16] is of particular interest, as the underlying principles generalize to any cellular layout.

In cognitive radio studies, three flavors of distributed sharing of spectrum between peers have been addressed. A first classification is whether power is continuously allocated over multiple channels [18], [19], [20] or whether each peer selects one channel (or more), but power control as such is not considered. In this paper we concentrate on the non-power-controlled case, which may be subdivided into two cases, corresponding to whether the decisions made by the peers are based on a continuous or binary function of the interference couplings between the peers. The case with continuous interference couplings, when the decision process may be modeled on an interference graph, has been addressed in [21], [22], [23]. All of these consider the selection of one resource, in an 802.11 -type scenario. Using the terminology of [21], the algorithms proposed in these papers may be called Interference

This work is funded by Nokia.

M. Peltomäki, J.-M. Koljonen, and M. Alava are with the Department of Applied Physics, P.O. Box 14100, Aalto University, FIN-00076 Aalto, Finland.

O. Tirkkonen is with the Department of Communications and Networking, P.O. Box 13000, Aalto University, FIN-00076 Aalto, Finland. Emails: firstname.lastname@aalto.fi

Reducing Network (IRN) algorithms—in each step, the node deciding on the channel to use, either changes to a channel providing smaller interference at the node, or does not change the channel. In [21], it was observed that for convergence, it is crucial that the interference experienced between the peers is symmetric, which was guaranteed by considering an interference coupling based on path loss measurements performed by the Access Points (APs) on other APs, and by requiring all transmit powers of all APs to be the same. With symmetric interference couplings, greedy IRN converges to what was observed to be a Nash Equilibrium of an exact potential game [21].

In this paper we investigate the use of distributed graph coloring algorithms, acting on a conflict graph, for spectrum sharing between peers. Conflict graphs with binary interference couplings can be related to interference graphs with real-valued couplings by thresholding. In the FDMA setting, algorithms based on local information exchange and rule-based reasoning and bargaining have been recently investigated in [24], [25], [26], [27], [28], with the TDMA setting addressed in [29]. Of these, [24], [26] consider a multicoloring problem, the others the selection of a single resource per node.

When distributed channel allocation approaches have been compared to centralized ones, performance gaps have been observed. In [26], the multicoloring problem on a conflict graph is addressed with heuristic rules based either on knowledge of the neighbors' channel use or on contention detection with backoff, both of which perform worse than a centralized assignment. In [23], an IRN algorithm is contrasted with a centralized frequency reuse plan, based on the same interference information, and it is observed that the distributed algorithm reaches some 90% of the capacity of the centralized solution. The reason for this is that the distributed algorithm gets trapped in a local optimum. This is the case for all IRNs that have been shown to converge in [21], [22], [23]. The requirement of reducing the interference at each step, together with the continuity of the interference couplings, drives the system to a nearby local optimum, which is rarely a global optimum.

In this paper we address this gap. We consider the cognitive networking function of finding a network-level optimum in a distributed manner. In a coloring of a conflict graph, our target is to converge to a perfect, conflict-free coloring. This indicates that one has found a *global optimum* of the channel allocation problem subject to thresholded interference couplings.

Two novel algorithms are proposed in this paper, with different convergence properties and need for message passing between neighbors. Semigreedy Distributed Local Search (SDLS) is an extremely simple algorithm, based on minimum information and no negotiation. The Self-Organizing Search Tree (SOST), in contrast, is a complete algorithm that finds a conflict free state if one exists. It is based on an elaborate negotiation protocol, implementing a distributed version of a classical graph coloring algorithm. The speed of convergence and reliability of these are studied via simulations and compared to the most recent negotiation-based algorithm from the literature [27]. For SOST we prove a proposition about the convergence time.

Next, we investigate different measurement, reporting and negotiation principles for generating interference and conflict graphs for self-organizing cellular networks, and evaluate distributed channel allocation algorithms in cellular settings. We discuss a Negotiated Worst Couplings (NWC) procedure, which guarantees symmetric couplings and is designed to favor cell edge users. In a cellular system setting we observe significant gains for cell edge users from using NWC, instead of using direct BS-BS couplings proposed in [21]. Also, we compare the interference situation experienced by individual users in networks employing SDLS and the IRN [21]. In some networking scenarios, when the interference coupling describes the interference experienced by a group of users, SDLS, attempting global optimization with a binary quantization of interference, may outperform IRN which finds a local optimum with real-valued interference.

The paper is organized as follows. First, Sec. II introduces the resource allocation problem on graphs, discusses the connection to graph coloring and the connection to spectrum allocation in wireless networks, and the example network topologies considered here. Sec. III discussed the algorithms introduced here and the emergence engineering algorithm to which the performance is compared. Convergence results on static networks are discussed in Sec. IV, as well as issues for future algorithm studies. In Sec. V, we consider a cellular system setting, propose the Negotiated Worst Coupling procedure, and measure the interference levels experienced in networks employing different principles to symmetrize interference couplings, and different distributed algorithms to find the resource allocation. Sec. VI wraps up the paper.

II. System Model

The main target of this paper is to analyze convergence properties towards conflict-free colorings on conflict graphs describing interference couplings between network nodes, and to evaluate their performance on conflict graphs that describe cellular networks.

A. Distributed Graph Coloring Problem

The resource allocation problem addressed in most of this paper can be formulated as follows. Form a conflict graph $G = G(V, E)$ of vertices $v \in V$ and edges $e \in E$ connecting two vertices. The vertices consist of collections of wireless network elements which are able to determine their resource usage on a much shorter time scale than the time scale of inter-vertex communication, and may be interchangeably called nodes, or cells. An edge connects two vertices if there is an conflict between the vertices when they use the same resource; the edge $(v_1, v_2) \in E$ if communication *within the vertices* v_1 and v_2 would interfere with each other if they use the same channel. Note that the channels are considered indistinguishable. The objective of resource allocation is to find an allocation so that no pairs of nodes connected by an edge share the same channel. The problem in itself is equivalent to the graph coloring problem [4], [30], in which one wants to find such an assignment of a finite set of colors to the nodes of the graph that no connected pair of vertices shares the same color.

In addition to the formulation above, we want to perform the allocation of the resources in a self-organized i.e. distributed manner, so that every node makes the decision of which resource to pick based on local information only. In general, these decisions take place inside a *routine* of which each node executes independently an identical copy. This routine has access to local information including the resources used by the neighboring nodes together with any data the nodes choose to broadcast to their neighbors. More precisely, the nodes execute the routine asynchronously, i.e. one at a time, and the order in which they do this is random but fixed. In addition, any information sent by a node to its neighbors is assumed to be immediately available to the neighboring nodes, i.e. no transmission delays are modeled.

B. Connection to Wireless Network

The conflict graphs addressed in this paper can be related to snapshots of networks of mobile radio Transmitter-Receiver elements (TRx's). In a snapshot, each TRx is located at a specific location $\mathbf{x}_i$. The transmitters (Tx) transmit, the receivers (Rx) receive, and due to the broadcast nature of the radio channel, each Rx receives an attenuated version of each signal transmitted. Due to the existence of thermal noise, weak signals can be neglected. As a consequence, the snapshot of the network is a weighted directional graph, where there are edges starting from the Tx's and ending at the Rx's. These edges are weighted by the interference powers I_{jk} between a Tx j and a Rx k. These are real numbers usually modeled as having a deterministic, distance-dependent component, and a random component due to shadowing and fading, e.g.

$$I_{jk} = P_j ||\mathbf{x}_j - \mathbf{x}_k||^{-\alpha} \xi_{jk} \ , \tag{1}$$

where P_j is the transmission power of j, α is the path loss exponent and ξ_{jk} is the random component of the path loss [31].

An *Interference graph* is a weighted graph with vertices describing entities where a radio resource is used, and edges decorated with real-valued interference couplings, is a typical abstraction of a Network graph used in frequency assignment problems [3]. To construct an interference graph from a network graph, we need to cluster a number of TRx's together. There are multiple candidates for such clusters.

- a transmitter-receiver pair in a cognitive network (considered in [22])
- a Base Station (BS) in a cellular network, together with the mobile stations it is serving, or a WLAN Access Point, together with its clients (e.g. [15], [21])
- a cluster in an ad hoc or sensor network (e.g. [23]).

There is an established intra-cluster method to share radio resources. In distributed inter-cluster RRM, the cluster is the entity that decides which resource to use. We consider all clusters to be peers, there are no preferred primary users. We shall not distinguish between Base Stations, Access Points or cluster heads, nor between cells and clusters. We use "BS" to refer to this category of network elements throughout, with "cell" denoting the surrounding cluster.

The interference graph is a tool to model the decision making process. Each cluster is a vertex in this graph, and the interference couplings between two vertices are functions of the interference power couplings of the TRx's belonging to these vertices.

Several alternatives for the *interference coupling* between a vertex consisting of a set $\mathcal{A}$ and a vertex consisting of a set $\mathcal{B}$ have been considered in the literature. In [21], the interference power of the transmission of the Base Station in $\mathcal{B}$, as measured at the BS in $\mathcal{A}$ is used. In [22], there is only one transmitter in $\mathcal{B}$ and one receiver in $\mathcal{A}$. In [23] the interference coupling is approximated by the path loss related to the distance of the centers of the clusters. In [15], a cellular network scenario is considered. The Mobile Stations in $\mathcal{A}$ measure DL interference powers created by transmissions of $\text{BS}_{\mathcal{B}}$, and signal them to $\text{BS}_{\mathcal{A}}$, who collects statistics and decides the interference coupling to be a predetermined percentile of this statistics, corresponding to a predefined outage probability (e.g. 5% outage).

Once an interference graph is constructed, it is straight forward to construct a conflict graph by *thresholding* [3]. All interference couplings weaker than a threshold H are put to zero, and all couplings stronger than the threshold are put to one. The number of neighbors of a vertex and correspondingly the colorability of the graph with a given number of colors depends crucially on H. It is worth noting that the conflict graph is not a model of the network as such, but a model of the decision-making process at the vertices. Modeling a network algorithm on a conflict graph implies that the information about the network interference that the vertices use to make decisions has been quantized to a binary on-off value.

In [21], [22], [23] asynchronous distributed channel allocation algorithms on interference graphs were considered. Convergence to a local optimum was proven for *symmetric interference couplings* between the vertices. This is also a crucial assumption for the convergence towards a global optimum considered here: the conflict graphs should be bi-directional. There are a few ways to guarantee this. For example, the interference couplings considered in [21], [23] are automatically symmetric if the transmit powers P_j are all same. In many cases, however, the only way to guarantee symmetry of the interference or constraint couplings is symmetrization based on inter-vertex signaling.

If *inter-vertex communication* is required, for symmetrization purposes, or for the operation of the algorithm itself, a signaling channel between the neighboring vertices is needed. For example, in addition to the channels used for communication within the vertices, there may be a channel reserved for inter-vertex communication. In cellular settings, mobile stations at cell edges may be used to relay information from one BS to another. In addition, in LTE systems, the network of BSs is meshed by so-called X2 interfaces over the core network, see e.g. [15]. This interface provides a natural channel for signaling between two BSs.

C. Example Network Graphs

In most cases we consider the simplifying assumption of having bi-directional graphs, i.e. that both vertices connected by an edge observe the conflict. We consider two types of graphs—random planar graphs based on Voronoi tesselations, and 3D graphs generated according to cellular system modeling principles.

1) Random Networks with Voronoi Tessellations: When the location of the Tx-Rx pairs in a vertex are randomly distributed in the two-dimensional plane, and the interference couplings are considered only between geographically nearest neighbors, the result is an irregular planar conflict graph. We consider a class of irregular planar graphs generated by a Voronoi tessellation [32] as follows. Take a square-shaped 2D box of size $L \times L$. First, draw N random points uniformly inside this box. Then compute the Voronoi tessellation of the box corresponding to the points [33]. The result is a division of the plane into cells surrounding each point such that a location $\vec{x}$ in the box belongs to the cell corresponding to the point that is the closest to $\vec{x}$. From the division into cells, a graph $G = G(V, E)$ is created. The drawn random points are taken as the vertices $v \in V$. Two vertices are connected by an edge $e \in E$ if and only if the corresponding cells are separated by a one-dimensional, i.e. not point like, boundary. By construction, the graphs generated in this way are planar. In other words, they can be drawn in the plane such that no two edges overlap, and they can be colored with four colors [4].

2) Three-dimensional Graphs Modeling Cellular Networks: An example of typical system simulation prescriptions for wide area cellular network evaluation can be found in [7]. A single-slope path loss model (1) with path loss exponent $\alpha = 3.76$ is assumed. Shadow fading is correlated between cells with a correlation coefficient 0.5, and the shadow fading correlation distance is 50 m. The base stations are placed on vertices of a regular triangular lattice.

A conflict graph for a wide area network generated according to the path-loss and shadow fading models of [7], is depicted in Fig. 1. We have dropped mobile stations on the sides of the regular hexagons surrounding the cells, separated by twice the correlation distance, i.e. 10 on each side. The shadow fades experienced at these mobile station locations can be considered uncorrelated. Cell selection is performed for all mobile stations; the serving cell is selected based on the smallest total path loss (including the deterministic distance dependent path loss, and shadow fading). Two cells are considered to suffer from an interference coupling if there is at least one mobile station for which one cell is the serving cell, and the path loss difference between the serving cell and the other cell is less than a threshold value, here 6 dB.

In the family of conflict graphs created according to these principles, the average number of neighbors of a node is 10.33, with a standard deviation of 2.25. The effect of correlated shadow fading giving rise to long-distance connections is clearly visible in Fig. 1. These connections make the graph non-planar (i.e. 3D).

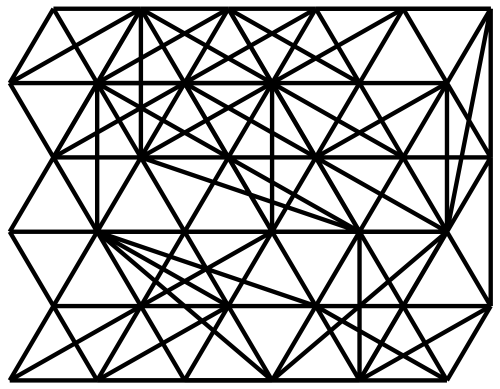


Fig. 1. An example graph generated based on path-loss modeling with distance-dependent path loss and shadow fading that is correlated between cells. Interference coupling threshold 6 dB.

III. Algorithms

In the literature, several algorithms solving the distributed graph coloring problem [34], [35], [24], [26], [27], [28], or almost equivalent problems on interference graphs [16], [21], [22], [23], [29], [15] have been proposed. The algorithms used can be classified into two classes. Greedy local search algorithms have been used in [16], [21], [22], [23], and rule-based reasoning and negotiation algorithms have been discussed in [24], [26], [27], [28], [29]. In [36], iterative local search algorithms were discussed for a generalized graph coloring problem. The algorithms in [21], [23] are distributed versions of such algorithms. In [21] a candidate color is selected randomly to be compared to the present color, whereas in [23], a multiple-try algorithm is used, where in each iteration, a node selects the best available color.

Here, we discuss two new classes of algorithms for distributed graph coloring. The first class is exemplified by a very simple Semigreedy Distributed Local Search algorithm, based on limited information exchange and no negotiations between the neighbors. It is an autonomous algorithm closely related to a conflict graph version of the greedy local search algorithms of [21], [23], but with an innocent-looking twist allowed by the conflict graph interpretation of the network, which makes it possible to random walk towards a global optimum. The second class consists of classical deterministic graph coloring algorithms that are deformed into distributed (but not autonomous) algorithms by reinterpreting the routines.

A. Emergence engineering algorithm

To compare the algorithms introduced in this contribution to existing ones, we use the state-of-the-art local-negotiation-based algorithm by Anthony [27], called here the emergence engineering algorithm (EEA, see also [25] for a predecessor). The algorithm is not limited to planar graphs or to any other particular setting. For convenience, we describe the algorithm briefly below.

The colors are ordered according to a global order of preference. At any given time, each node is in one of four possible states: isolated, legal, illegal, or no-color. Initially, all nodes are in the isolated state, and assigned the first color according to the order of preference. The algorithm works by messages sent by nodes to their neighbors. The messages come in two flavors: status messages, and force messages. Status messages are almost periodic communications of the present state, color, and accumulated information from the neighbors of the sending node. When a node cannot assign an immediately satisfying color assignment, it chooses the color least disturbing to the neighbors according to an heuristic, forces some neighbors to change their color by a force message. To break ties, the algorithms uses random weights assigned initially to the nodes uniformly from the interval $[0, 1]$. These weights are then carried in the status messages, with nodes with larger weights dominating in otherwise symmetric conflict situations. See Sec. IV-C for a discussion of the consequences of this feature.

To measure time in EEA in a manner directly comparable to the algorithms introduced in this work we proceed as follows. An event is either the scheduled broadcasting of status messages by a node, or receiving a message and performing the related tasks. An iteration is a sequence if N events, where N is the number of nodes, and the unit of time used in this work is one iteration.

B. Semigreedy distributed local search

The Semigreedy Distributed Local Search (SDLS) algorithm is a distributed algorithm embodiment of the principles underlying the zero-temperature Markov Chain Monte Carlo algorithm in statistical physics [37]. It works on any graph, and in it, each node executes periodically a routine which works as follows.

- Compute the number of neighbors N_{old} using the same resource.
- Pick a resource not currently used uniformly at random.
- Compute the number of neighbors N_{new} using this newly picked resource.
- If $N_{\text{new}} \leq N_{\text{old}}$, then switch to use the randomly picked resource. Otherwise do nothing.

A flow chart of the algorithm can be found in Fig. 2. In the last rule, there is an innocent-looking equality in addition to $<$. Transitions which do not change the number of conflicts are allowed and do take place if the solution is degenerate enough. This partially prevents stalling to local optima, since the non-converged cases are free to move randomly inside the subset of configurations with equal (or smaller) number of conflicts, giving a chance to stumble on a global optimum. This is a *significant difference from an IRN* operating with real-valued interference couplings. In an IRN, all local optima (up to a zero-measure set) are isolated, and a greedy algorithm never gets away from a local optimum.

Again, one sweep in which each node executes the routine once constitutes an iteration used in measuring the time. For SDLS, the only information required by a node is the resource occupancy of the neighbors, and no negotiations are needed. For this, the node needs to be able to distinguish its neighbors. The algorithm can start from any configuration, and when a properly colored end state is found no color changes occur anymore regardless of the nodes still running their routines. In this respect, SDLS and EEA are directly comparable. For the self-organizing search tree these statements do not hold.

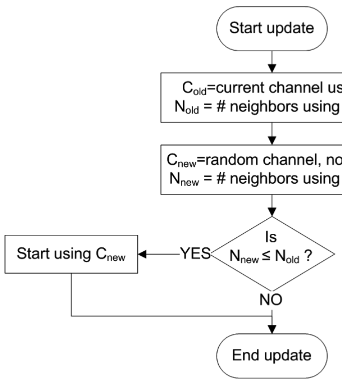

Start update
C_{old}=current channel used
N_{old} = # neighbors using C_{old}
C_{new}=random channel, not C_{old}
N_{new} = # neighbors using C_{new}
Is $N_{\text{new}} \leq N_{\text{old}}$?
YES
NO
Start using C_{new}
End update

Fig. 2. Flow chart of the SDLS algorithm.

C. Self-organizing search tree

In addition to the EEA and SDLS algorithms, based on more or less complex metaheuristics, it is essential to compare to a complete algorithm, which is able to find a conflict-free coloring if it exists. For this, we introduce a self-organizing search tree (SOST) algorithm. It is the well-known classical naïve search tree [38], [39] for vertex coloring, cast in a different notation in which each node executes independently a routine. Symmetry breaking related to self-organized starting of the execution is solved by using global identifiers. In this setting, all the heuristics and tricks related to the non-distributed counterparts are also usable in SOST.

Algorithms implementing a distributed version of a search tree are known in the literature [34], [35]. However, in the distributed algorithm of [34], Asynchronous Back-Tracking (ABT), there is a procedure for adding links between nodes that originally do not share a constraint. In the wireless resource allocation problem addressed here, nodes not originally sharing a constraint are not radio neighbors. To establish a link between two such nodes would require establishing a relaying protocol, where intermediate nodes relay the information passed over the added link. To avoid such complexity, it is worthwhile to reconsider the task of distributing a search tree, without using a link addition procedure. For this purpose, we propose SOST.

To understand the algorithm, a brief review of the underlying classical search tree [38], [39] is in order. In it, in addition to its color, each node carries a Boolean flag indicating whether the node has already been colored, and a list of the root nodes of the subtrees colored under the node. Initially, all the Boolean flags are false, and the lists are empty. The algorithm starts by calling the coloring procedure for a single node, which can be chosen arbitrarily in the case of connected graphs. It calls recursively itself, and an auxiliary uncoloring procedure. The algorithm finishes when the first call to the coloring procedure returns. Then, either a coloring has been found, or the return value signals that no coloring exists.

In the distributed version, SOST, the nodes are equipped with global unique identifiers[1]. Using them, each node can spontaneously decide to act as the initial node, with a predefined probability p_1. The branch started from this node carries the global identifier of the initiating node, and this identifier is then used to arbitrate between two branches whenever they collide. The global domain identifiers thus play here a role similar to the random weights in EEA. This is enough to finally select one point source to initiate a coloring. The calls of the coloring and uncoloring procedures and the associated returns from a node to another are implemented by a series of messages passed between the nodes, indicating the procedure to be called and its parameters. A detailed description of SOST can be found in Appendix A.

Note that SOST makes the assumption together with EEA and SDLS, that all nodes are aware of their neighbors. This is enough to build the tree one link at a time by using these neighborhood relations. The algorithm is implemented as a routine that all nodes execute independently an identical copy of. We consider two convergence improving heuristics - a neighbor selection heuristic, and a bound-and-restart protocol, see Appendix A. Similarly to SDLS, one sweep in which each node executes this algorithm once forms an iteration, the unit of time in measuring the speed of convergence.

To summarize with short descriptions the ideas of the algorithms: in SOST, implement the classical search tree for graph coloring in a self-organized manner, by turning the function calls and associated returns into communication between the nodes. In SDLS, a descent towards a minimum number of conflicts is combined with moves that do not change the total number of conflicts.

Now that SOST has been explained, a detailed comparison with the algorithms in [34], [35], is in order. Yokoo and coworkers discuss two implementations of a search tree for constraint satisfaction, a classical non-distributed one called synchronous backtracking (SBT), and a distributed one called asynchronous backtracking (ABT). While SBT is extremely close to the classical search tree, and ABT bears some resemblance to SOST, these algorithms are materially different. The crucial differences of ABT and SOST are the link addition procedure of ABT, and the utilization of the global identifiers of the nodes as inherited identifiers or the domains in SOST. First, ABT has a procedure which deduces new constraints based on the existing ones and incorporates these into the problem structure, and adds new links between at least a subset of the nodes involved in the new constraints. In the language of the graph coloring problem, this means deducing rules stating that two nodes not originally connected cannot share the same color since such a configuration would imply an unresolvable conflict in another part of the graph. SOST has no such feature, and is accordingly better suited to a wireless environment. Second, SOST uses global identifiers so that in addition to belonging to individual nodes, the identifiers of the nodes that have started the branches are also properties of the branches, i.e. domains. These can then be used to arbitrate between different domains to resolve conflicts through symmetry breaking. On the other hand, ABT has no such feature: the symmetry breaking happens through node-specific identifiers only. The effects of these differences in the running time are *a priori* unknown.

[1]The requirement of uniqueness is lifted in Appendix A.

IV. Convergence Results

In this section, we study convergence towards a conflict-free channel allocation, which is a global optimum of the channel allocation problem when interference has been thresholded to a conflict graph.

A. Bounds on Convergence Times for SOST

To compare different algorithms, bounds on convergence times are of importance. Here, we prove bounds for the convergence times of SOST under certain conditions, or if the graph is uncolorable, the time for SOST to find this. We also discuss why similar bounds on other algorithms and other cases are not readily obtainable.

Proposition 1. In a network with N nodes, the self-organizing search tree (SOST) without the bound-and-restart protocol converges to a colored state, or stops execution if the graph is uncolorable, in

$$n \leq n_{\text{init}} + n_{\text{prop}}$$

iterations, where

$$n_{\text{prop}} \leq 2(q+1)^N$$

with q the number of available colors, and the probability

$$P\big(n_{\text{init}} \leq k\big) \;=\; p(k;p_1) \;\geq\; [1-(1-p_1)^k]^N \quad \forall\, k\ .$$

Here p_1 is the parameter of the algorithm determining the probability for a node to decide to act as an initial node for a coloring branch.

Proof. All contributions to the convergence times come from two processes, n_{init} and n_{prop}, separately. The first contribution n_{init} is stochastic, and related to the self-organizing spontaneous starting of coloring branches. The system ends in a state in which no new branches start anymore if all nodes have decided to act as the initial node. Therefore, the probability that the any given individual node has decided to act as an initial node within k iterations is $1-(1-p_1)^k$, and consequently the probability that all nodes have decided to act as initial nodes within k iterations is $[1-(1-p_1)^k]^N$, since the nodes make the decisions independently. Thus $p(k;p_1) \geq [1-(1-p_1)^k]^N$.

The second contribution, n_{prop}, is deterministic, and related to the convergence of the search tree on an individual coloring branch. With q colors there are $(q+1)^N$ possible states, where the factor $q+1$ originates in the q available colors and the uncolored state. In the progress of the algorithm, each state is visited at most two times, and each visit takes at most one iteration. Therefore, the deterministic contribution to the number of iterations is limited from above, $n_{\text{prop}} \leq 2(q+1)^N$. □

It is worth noting that $p(k;p_1) \to 1$ as $k \to \infty$ for all $p_1 > 0$. Tuning the parameter p_1 in SOST can be used to improve the speed of convergence. However, this improvement can be seen only in the initial non-propagating phase of the algorithm, which severely limits the amount of improvements that can be gained. This limitation is most visible when the convergence time is dominated by the actual search tree, and as shown below in Sec. IV-B1 grows at least exponentially.

In the special case $p_1 = 1$, an initial branch is always generated within the first iteration, and the upper bound for the convergence time is then $n \leq 1+2(q+1)^N$. This eliminates the stochastic contribution, and turns the estimated convergence time into a strict upper limit, but does not deal with the root cause of slowness explained above.

Similar propositions for SOST with bound-and-restart or for SDLS seem not to be easily obtainable, and therefore we consider pursuing any such outside the scope of this work. It is likely that for both cases it is possible to construct pathological scenarios that can be shown to never converge. Especially for non-colorable graphs, SDLS may end up not converging. As an example, consider a conflict ring of three nodes, and two colors. In this scenario, SDLS will continue flipping colors forever. Such pathological loops may be cut by introducing tabu-lists, see e.g. [40]. Note, however, that the examples considered below in Sec. IV-B constitute a proof that both converge with a strictly positive probability.

B. Numerical results

1) On Voronoi tessellations: On the planar graphs built via the Voronoi tessellations (Sec. II-C1) the SOST and SDLS algorithms are usable and can be compared to EEA [27].

We have measured the median running time for SOST, SDLS, and EEA on Voronoi-tessellated graphs for four, five, and six resources. The results can be found in Fig. 3. For four resources and graphs with less than 250 nodes, SDLS converges faster than the other algorithms. Above this limit, it does not converge at all since it is trapped in a local minimum of the number of conflicts. Above this limit, EEA still converges, and for the whole size region considered, its convergence time scales as the logarithm of the network size. SOST behaves clearly worse, even though for less than approximately 60 nodes, it outperforms EEA. See Sec. IV-C for discussion on how to overcome the poor performance of SDLS for large networks.

For five and six resources, the convergence time of both EEA and SDLS scale as the logarithm of the number of nodes, and clearly outperform SOST. In addition, SDLS converges faster than EEA for all considered network sizes. For six resources, the convergence time of SOST scales linearly in the number of nodes. Based on this observation, the amount of backtracking performed is negligible. Note that only networks with fixed node densities are considered, when the node number is varied. The case with three or less resources is not worth considering in such arbitrary planar graphs as those created through the Voronoi tessellation since in general they are not colorable with three colors.

2) On three-dimensional graphs: For the three-dimensional case, (Sec. II-C2) we discuss both the EEA and SDLS algorithms and leave SOST aside since it is expected to be outperformed both by EEA and SDLS. The median convergence time for six and seven resources is plotted in Fig. 4 as a function of the number of nodes. There are two immediate conclusions. First, the convergence time behaves roughly polynomially as a function of the number of nodes but no clear functional dependence can be fitted. Second, SDLS converges faster than EEA for all network sizes considered, and for small networks (25 nodes for the data shown in the figure) it is even ten times faster than EEA. For seven resources, SDLS outperforms EEA by a factor of approximately ten, and both algorithms converge in time polynomial in network size. The three-dimensional network topology considered here contains 6-clicks, i.e. completely connected subgraphs with six nodes. Thus the graphs are not colorable with less than six colors.

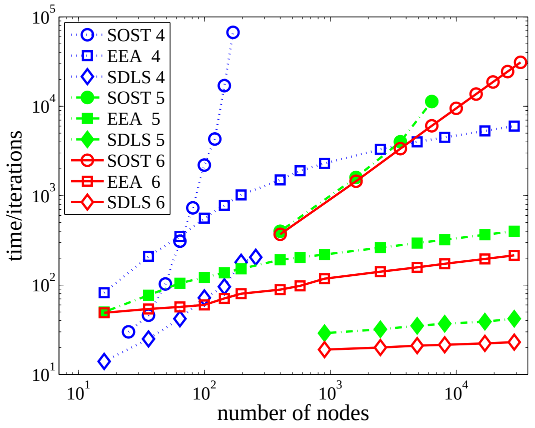


Fig. 3. The median convergence time as a function of the number of nodes for SOST, SDLS, and EEA for four, five and six resources on Voronoi-tessellated graphs. The number of resources indicated in legend after name of algorithm.

C. Discussion

In the light of the results obtained here, the self-organizing search tree, despite being a complete algorithm, appears somewhat useless. It is outperformed by both Semigreedy Distributed Local Search, and the Emergence-Engineering Algorithm [27], both of which allow for true parallelization of the coloring problem. This may not be the whole truth, as SOST is open to extensions. Namely, the search tree is node-recursive in the sense that it has the following properties:

- The nodes have a coloring procedure and possibly other (auxiliary) procedures.
- These procedures work by accessing internal variables of the nodes, calling the same procedures for neighboring nodes, and processing their return values.
- The algorithm can be started by calling the coloring procedure of any node, and it ends when the call terminates.

Using the same reasoning as in Sec. III-C and in Appendix A, any graph coloring algorithm with these properties can be turned into a self-organizing one without increasing the computational complexity. At least some of the existing graph coloring algorithms can most likely be cast in a node-recursive form. Reasonable candidates for these include the polynomial-time four- and five-colorings for arbitrary planar graphs introduced in [30].

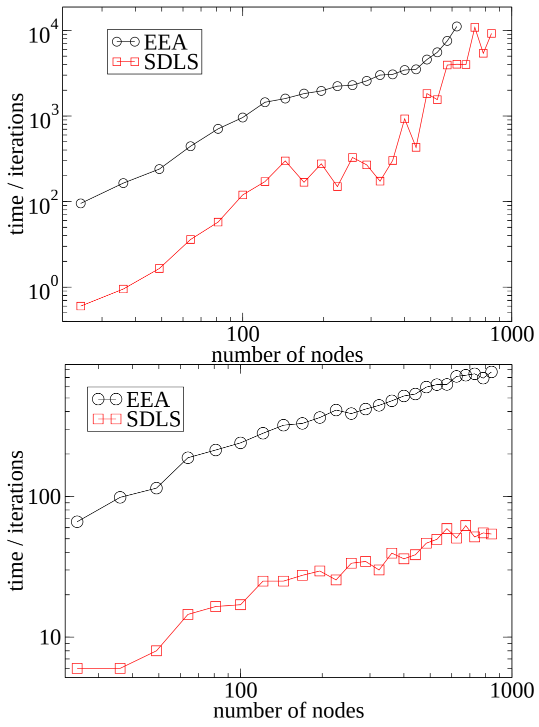


Fig. 4. The median convergence time as a function of the number of nodes for EEA and SDLS on the three-dimensional graph topology. Up: six resources. Down: seven resources.

Some of the algorithms discussed here require identifiers for the nodes with variable level of uniqueness. The self-organizing search tree requires locally unique identifiers to resolve conflicts. The requirement of global uniqueness has been lifted as a side effect of the bound-and-restart protocol, but local uniqueness is still needed for the procedure calls and returns between nodes to route properly. The emergence-engineering algorithm has a similar requirement in terms of the random weights which are used to break symmetry between neighboring nodes. If the weights are equal for two adjacent nodes, which might happen for example when they are drawn from a finite set, the symmetry cannot be broken, and converge is either never achieved or it slows down. On the other hand, SDLS does not have any such requirement.

In the algorithms discussed in this article, the nodes update their status in a random fixed order. This differs from the approach taken e.g. in EEA [27]. There the nodes respond to status messages they receive, and these are sent in intervals consisting of a deterministic part, which is the same for all nodes, and a random part. As a result, after a couple of iterations, the nodes act completely asynchronously and the ordering of their updates cannot be predicted from the order a couple of iterations earlier. Since this amounts to additional symmetry breaking between nodes on top of all other random components of the algorithms, a question arises whether this has an effect on the speed of convergence. To answer this question, we have run simulations using both a random fixed sequence, and sequences randomized separately during each iteration. The simulations were performed using a planar random graph (Sec. II-C1) with $N = 10000$ nodes, the SDLS

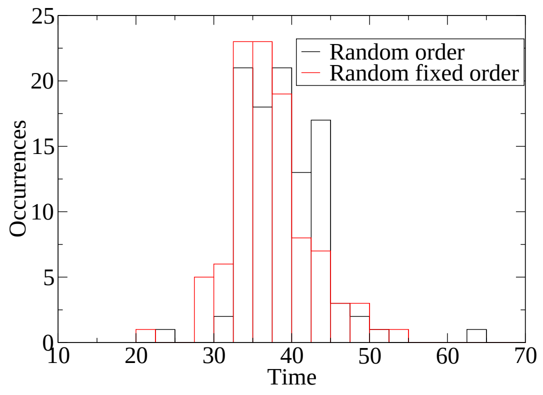


Fig. 5. Histograms of the convergence time of SDLS algorithm with fixed random order (red) and non-fixed random order (black) for the updates.

algorithm, and 5 colors, and repeated for both update-order schemes 100 times. The histograms of the convergence time in both cases are shown in Fig. 5. No difference is observable in the convergence times.

The results here have shown that Semigreedy Distributed Local Search is the fastest of the algorithms studied here in some cases. This includes every other combination of the parameters except the case with four resources and large networks. Note however that the networks studied in this work are built such that the node density does not increase nor decrease as a function of the network size. If this were the case, the results might be altered significantly, and cases in which EEA outperforms SDLS even though the latter converges might be revealed.

Simple examples above have highlighted that the failure of SDLS to converge on large graphs with four colors is caused by trapping in local minima, and therefore a question of how to overcome this barrier while still keeping the properties of SDLS making it converge fast is of great interest. Since SDLS is a distributed algorithm equivalent of the zero-temperature Markov Chain Monte Carlo in statistical physics, applying its finite-temperature counterpart seems appealing. In particular, then transitions increasing the number of conflicts become possible, and trapping to local minima no longer takes place. However, in this case the full solution loses an important property it has in all cases considered so far: it is no longer an absorbing state of the algorithm. In other words, the algorithm might first find the full solution, but then because of the nonzero temperature abandon it. This behavior is not desired, and more elaborate schemes are needed to resolve the issue. Work in this direction is in progress.

V. System Performance Evaluation

In this section, we compare the system performance of cognitive radio networks employing different distributed resource allocation methods. We discuss a symmetrization procedure targeting a maximum improvement in cell-edge performance, and compare between Interference Reducing Networks (IRN) of the kind suggested in [21], [23], with real-valued interference couplings, and SDLS with thresholded interference couplings. More exactly, we consider the IRN algorithm

TABLE I
SIMULATION PARAMETERS

Shadowing std	3 dB
Shadowing correlation	0.5
BS selection criterion	Euclidean distance
Distance dependent path loss	$L = 122.1 + 37.6 \log_{10}(d[\mathrm{km}])$
Inter-BS distance	50 m
Transmit power P	20 dBm
Antenna patterns	omni directional

proposed in [21], where each base station tests a resource not used by itself randomly, and selects the new resource if its interference is smaller.

A. System Model

For evaluation, snapshots of a radio network are created as discussed in Section II-C2. One hundred base stations are located at centers of 10×10 hexagonal cells. A number of users are randomly dropped in each cell. The BSs select one out of $N = 4$ orthogonal channels based on running a routine of a self-organized resource allocation algorithm. The simulation parameters are summarized in Table I. The path loss model is of the type (1) with log-normal shadow fading and parameters according to [7] at carrier frequency 2 GHz. For simplicity it is assumed that all traffic is in the downlink direction.

The distributed resource allocation algorithms are run until they converge (to a local optimum for IRN, or to a global optimum for SDLS), or until 100 iterations have been done. After that, statistics of the experienced Carrier-to-Interference plus noise (C/I) ratios experienced by the users in the system are gathered. Based on these statistics, metrics for comparing system performance can be evaluated. In contemporary cellular standardization studies [7], two numbers are used to describe the efficiency of RRM; the expected rate experienced by an average user and the rate experienced by a user in poor radio conditions, which is defined as a user located at the 5% point $\gamma_{0.05}$ of the Cumulative Distribution Function (CDF) of C/I. As we shall not perform proper intra-cellular RRM, we shall be content with investigating system performance on the level of the C/I-statistics, and occasionally consider the 5% point of the CDF.

B. Interference Couplings

Three alternative interference couplings between cells $\mathcal{A}$ and $\mathcal{B}$ have been studied:

- *BS-BS couplings*—interference power measured at $\mathrm{BS}_\mathcal{A}$ of transmission of $\mathrm{BS}_\mathcal{B}$ [21]
- *Cellular Worst Couplings* (CWC)—worst relative interference from $\mathrm{BS}_\mathcal{B}$ experienced by users in $\mathcal{A}$
- *Negotiated Worst Couplings* (NWC)—CWC partially symmetrized by inter-cell negotiations.

CWC and NWC are designed to maximally improve the experience of the worst positioned users in the system. With CWC, this is done inside each cell, whereas with NWC, inter-cell cooperation is used in addition. Concentrating on the worst users is in accordance with the fact that in a cellular network, the role of Inter-cell interference coordination, is to improve the channels experienced by cell edge users [9].

BS-BS couplings are symmetric if all BSs transmit with the same power [21]. CWC is by definition not symmetric, only when the number of users per cell approaches infinity, it would become symmetric. NWC cannot be symmetrized over all pairs of BSs—symmetrization only makes sense if the interference is above a threshold. Denoting by $C_{k\mathcal{B}}$ the ratio of the signal powers received from $\mathrm{BS}_\mathcal{B}$ and $\mathrm{BS}_\mathcal{A}$ by user k, the CWC interference coupling decided by $\mathrm{BS}_\mathcal{A}$ based on the information it has collected from the served users is

$$C^{\mathrm{C}}_{\mathcal{AB}} = \max_{k \in \mathcal{A}} \ C_{k\mathcal{B}} \ . \tag{2}$$

In [15], instead of the worst interference, the fifth percentile of the CDF of the interferences caused by $\mathrm{BS}_\mathcal{B}$ to the users in $\mathcal{A}$ was used as the interference coupling. Here we consider at most ten users per cell, so that CWC corresponds in essence to the interference coupling of [15].

To construct a NWC from CWC, $\mathrm{BS}_\mathcal{A}$ informs $\mathrm{BS}_\mathcal{B}$, if this coupling is larger than a threshold H. Then both $\mathrm{BS}_\mathcal{A}$ and $\mathrm{BS}_\mathcal{B}$ use the larger of their CWCs as the NWC,

$$C^{\mathrm{N}}_{\mathcal{AB}} = \begin{cases} \max\{C^{\mathrm{C}}_{\mathcal{AB}},\ C^{\mathrm{C}}_{\mathcal{BA}}\} & \text{if } C^{\mathrm{C}}_{\mathcal{AB}} > H \ \text{ or } C^{\mathrm{C}}_{\mathcal{BA}} > H \\ C^{\mathrm{C}}_{\mathcal{AB}} & \text{otherwise} \end{cases} \tag{3}$$

The conflict graph representation of the interference graphs, used by SDLS, is a thresholded version of the interference graphs. With BS-BS couplings, there is a conflict between $\mathcal{A}$ and $\mathcal{B}$ if the interference coupling is larger than an absolute value threshold H_{abs}. With CWC and NWC, a relative threshold is used, as in (3), with the resulting adjacency matrix element

$$A^{\mathrm{N}}_{\mathcal{AB}} = \begin{cases} 1 & \text{if } C^{\mathrm{C}}_{\mathcal{AB}} > H \ \text{ or } C^{\mathrm{C}}_{\mathcal{BA}} > H \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The interference coupling alternatives have different signaling requirements. For CWC and NWC, intra-cell signaling is required to collect the measurement results to the BS. For NWC, additional inter-cell signaling is required for partial symmetrization.

SDLS and IRN, have rather similar signaling and measurement requirements. Typically, SDLS could use less bits for both intra- and inter-cell signaling, as the interference information is binary. The measurement requirements of the algorithms are rather similar as well. To perform the measurements required for BS-BS couplings, BSs employing SDLS need to be able to distinguish between transmissions from different BSs, whereas for IRN, it is sufficient to identify which transmissions are from BSs in the first place. To construct Worst Couplings (CWC or NWC), users need to distinguish between transmissions from different base stations, irrespectively of the algorithm used, with the exception of CWC when there is only one user in the cell (“CWC/1u”). In this case, studied in [22], the user only has to measure the total interference power on a channel, and the decision whether a channel is better than the one used can be taken at the measuring device. The measurement and signaling requirements are summarized in Table II.

TABLE II
SIGNALING AND MEASUREMENT REQUIREMENTS OF INTERFERENCE COUPLING AND ALGORITHM ALTERNATIVES.

Interf. coupling	Algo.	Measurement capability	Signaling type	Signaling intra-cell	Signaling inter-cell
BS-BS	IRN	distinguish BSs from other Txs	N/A	no	no
	SDLS	distinguish each BS	N/A	no	no
CWC/1u	IRN	measure interf. power	bin	user→BS	no
	SDLS	distinguish each BS	bin	user→BS	no
CWC	IRN	-”-	real	users→BS	no
	SDLS	-”-	bin	users→BS	no
NWC	IRN	-”-	real	users→BS	yes
	SDLS	-”-	bin	users→BS	yes

C. Simulation Results

The performance of SDLS depends crucially on the threshold used. In the left half of Fig. 6, IRN is contrasted with SDLS with three different threshold values. The number of snapshots is 1000, and there is one user per cell. Neighboring BSs with received powers larger than -68, -64 and -60 dBm are considered as conflicting BSs, respectively. The corresponding convergence probabilities for SDLS are 0%, 85% and 100%, respectively. With a properly selected threshold, $H_{\rm abs} = 64$dBm, SDLS outperforms IRN, and global optimization based on binary couplings outperforms local optimization based on real couplings. With the optimum threshold, the resulting graph is barely colorable. With $H_{\rm abs} = 60$dBm, less conflicts are considered, resulting in more interference left outside of the scope of SDLS, and with $H_{\rm abs} = 68$dBm, there are too many conflicts for SDLS to color the graph. As there is no ordering among the conflicts, less interference is removed by the unsuccessful coloring than with the barely successful coloring.

The lower end of the C/I distribution for Negotiated Worst Couplings with one user per cell is depicted in the right half of Fig. 6. SDLS with thresholds H =-15, -17 and -19dB are called “SDLS NWC” with $\text{abs}(H)$ appended, and the corresponding convergence probabilities are 89%, 28% and 0.3%, respectively. In order to treat IRN fairly, we have considered IRN with symmetrization of interference couplings above $H = -19$dB, which has the legend “IRN NWC”. With this threshold IRN NWC generates as much traffic as the most traffic generating of the considered SDLS NWC algorithms. Now, when the interference couplings are directly connected to the interference experienced at Rx, IRN performs well. Also, the picture related to the best value of H for SDLS is richer. With $H = -15$dB, SDLS is barely capable of coloring the network, and with $H = -17$dB, some conflicts are typically left. As opposed to the case with BS-BS interference couplings, these conflicts lead directly to interference at some Rx’s, and there is no control over the level of interference. Accordingly, an outage gap starts to develop for SDLS with too high threshold. However, with respect to all metrics considered, $H = -17$dB is the best threshold, despite the outage gap (and the corresponding non-convergence). Except a small region for users with C/I close to the 5% point, IRN

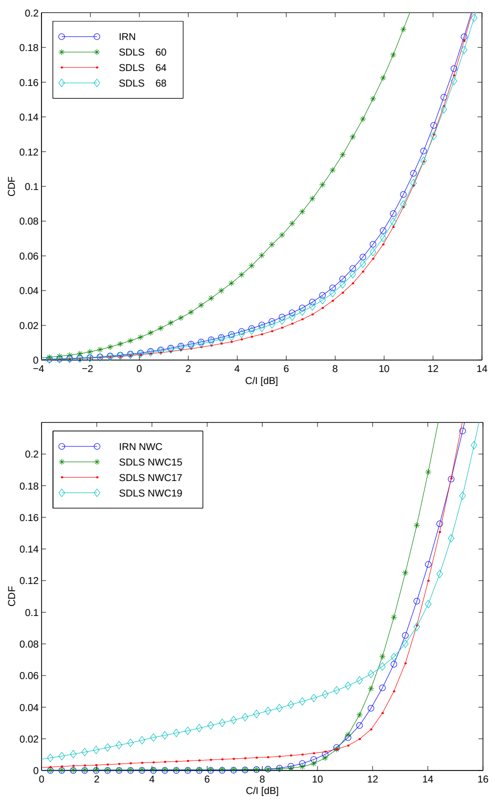


Fig. 6. Lower end of C/I CDFs, four orthogonal resources. SDLS with three thresholds. Up: BS-BS interference couplings. Down: Negotiated Worst Couplings, one user per cell.

outperforms SDLS in this scenario.

In these simulations the combination of thresholds and parameters considered is judiciously selected off-line, so that the cross-over behavior demonstrated here is visible. Indeed, as demonstrated in Figure 6, the dependence of performance on the threshold is very delicate. A mechanism for self-organizing threshold selection has been discussed in [41].

A comparison between IRN and threshold-optimized SDLS with both BS-BS, CWC, and NWC couplings is depicted in Fig. 7, for one and 10 users per cell. For the one user case, most of the range of the CDF is shown, as there is cross-over behavior. For ten users, the plot is zoomed to the low end, where the differences are seen. The results can be summarized as follows

- Despite the couplings being non-symmetric, so that convergence cannot be proved, IRN with Cellular Worst Couplings always converges in this scenario. On the contrary, SDLS with CWC never converges, and performs uniformly poorly.
- Negotiated Worst Coupling symmetrization outperforms BS-BS coupling except for the high C/I-users when there is one user per cell—symmetrization explicitly discriminates these users.
- IRN with CWC performs uniformly slightly *better* than IRN NWC when there are ten users per cell. When there is one user per cell, IRN CWC (which then becomes the

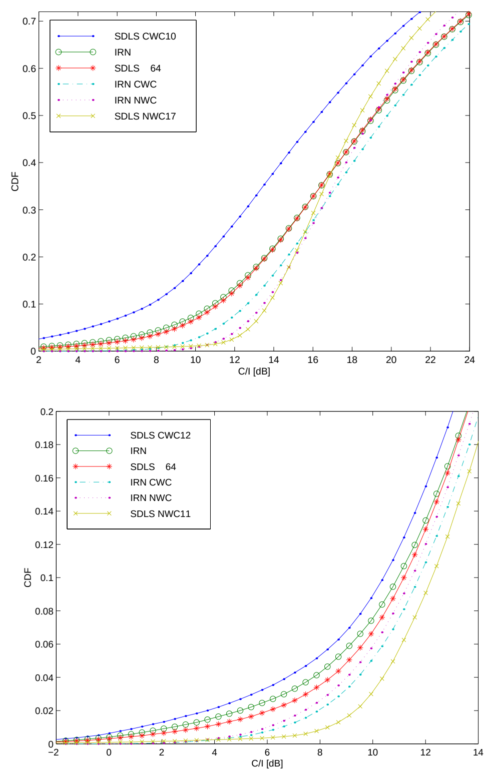


Fig. 7. C/I CDFs, four orthogonal resources, different interference couplings. Up: one user per cell, Down: 10 users per cell.

algorithm of [22] with utility U1) uniformly outperforms IRN, which uses BS-BS couplings. Accordingly, when the number of users per cell increases, there seems to be an unexplored cross-over in IRN where inter-cell selfishness becomes favorable over inter-cell coordination.

- When the interference couplings are only statistically related to the average user's C/I situation (i.e. with BS-BS couplings, or with NWC with 10 users), using SDLS to attempt global optimization based on binary interference outperforms local IRN-optimization based on real-valued interference .

It is interesting to note that CWC sometimes outperforms NWC. Convergence of IRN algorithms to a local optimum can only be proven for symmetric couplings [21], [23]. For asymmetric couplings it is easy to construct pathological non-converging examples. The fact that CWC nevertheless sometimes outperforms NWC indicates that from the perspective of network operation, convergence to a global or local optimum per se may not be all that important.

VI. Conclusion

In this work, we have considered the problem of conflict-free self-organizing resource allocation for wireless networks—a problem equivalent to the distributed graph coloring problem. Two algorithms were proposed for this, a simple Semigreedy Distributed Local Search, and a complete algorithm, the Self-Organizing Search Tree. These have been compared to the Emergence-Engineering Algorithm [27], based on heuristic rule-based reasoning. For coloring planar graphs, it was observed that the fully parallelized EEA and SDLS outperformed the complete SOST with a wide margin, and that SDLS provided the fastest convergence always when it was not trapped in a local minimum.

In a wireless networking scenario, we have evaluated the resulting interference levels experienced by users in a self-organizing cellular system, contrasting the graph-coloring based approach (on a conflict graph) with Interference Reducing Networks (on a weighted interference graph) from the literature [21], [23]. Circumstances leading to less interference when graph coloring is used have been pointed out. Such situations occur when the interference levels experienced by the average users are only statistically linked to the interference couplings between base stations.

Using graph coloring requires the knowledge of an optimized threshold for constructing the constraint graph from the interference graph. Finding such thresholds is an interesting self-organization problem in its own right. First steps in understanding this problem have been taken in [41], where instead of striving for consensus about a global threshold, each node keeps its own threshold, which is updated according to the progress of the coloring.

For self-organizing resource allocation in wireless networks, we considered a Negotiated Worst Coupling procedure to partially symmetrize the interference couplings of a network, so that cell edge users are favored. Irrespective of the algorithm used to color the corresponding graph (IRN or SDLS), this procedure guaranteed significantly better cell-edge performance than using direct BS-BS couplings. Also, we observed that with real-valued IRN algorithms, asymmetric interference couplings sometimes lead to better system performance than symmetrized couplings, despite the fact that convergence cannot be proven with asymmetric couplings.

Appendix A
Self-Organizing Search Tree

In the self-organizing search tree, each node executes independently and asynchronously an update routine consisting of the rules explained below. At any time there may be many coloring branches growing on the graph. A given node, however, is part of at most one branch at a given time. Collisions of branches are resolved by a branch ID. A successful coloring finds a spanning tree for the network in which all nodes belong to the successful branch, and each node except the root of the tree has a well-defined parent. To realize this, each node i has a node-specific identifier $r_i > 0$, and node i broadcasts (directly or indirectly) the following flags to its conflict graph neighbors:

- A_i, the identifier of the coloring branch node i considers itself to belong to. If $A_i = 0$, node i does not consider itself colored.
- c_i, the color being used by node i (note that a node may use a color, even if it does not consider itself colored).
- n_i, the number of colored neighbors of node i.

In addition to these flags, the nodes maintain variables storing the IDs of the current parent node m_i and current child node d_i, and two lists, one of usable colors and the other of successful children. Initially $A_i = n_i = m_i = d_i = 0$, the two lists are empty, and c_i is randomly chosen.

Five types of messages are passed between the nodes: `color`, `all colored`, `backtrack`, `uncolor`, and `uncoloring complete`. These implement calls and returns to the routines of the distributed coloring algorithm. The nodes maintain an inbox where they store incoming messages until it is time to run an update step.

Within a branch, at each instant of time, the execution is at one node only. This node processes its inbox, treating its possible incoming messages, and possibly generates a new call. Message `color` is transmitted from a parent to a new child, with the return `all colored` or `backtrack`. If a node receives `backtrack` from its current child, it has to send `uncolor` to all its previously colored children. The return of this call is `uncolor complete`.

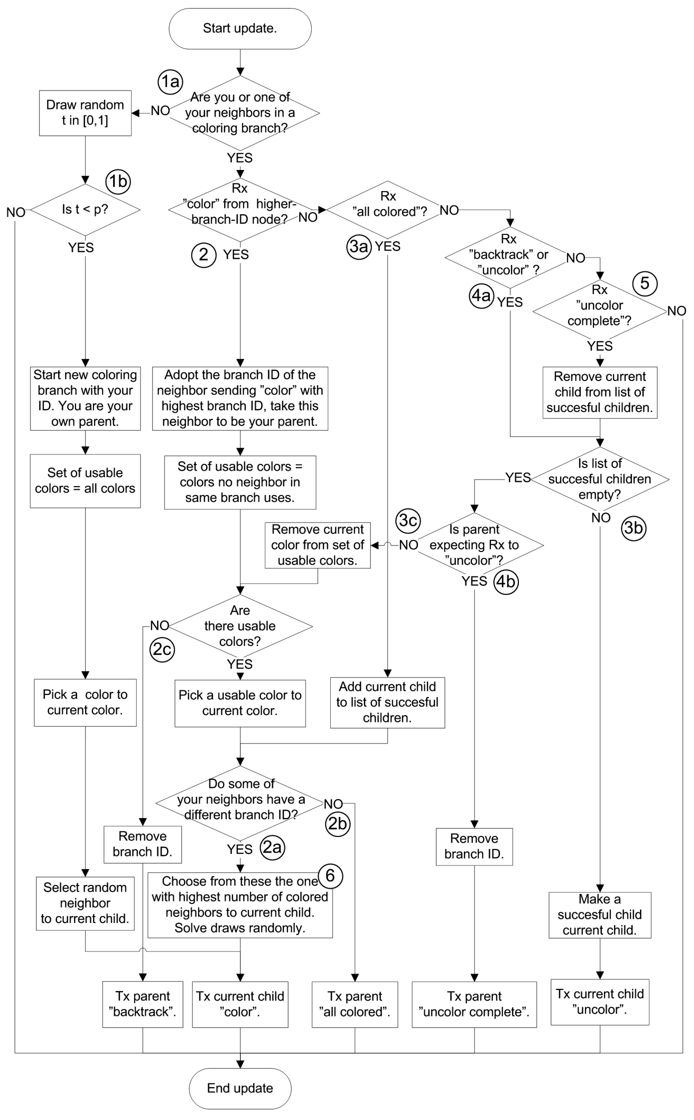
A flow diagram depicting one implementation of SOST is depicted in Fig. 8. In this implementation, the nodes need in addition a Boolean variable indicating whether the current parent expects a reply to an `uncolor` message.

Some explanations are in order to clarify the structure of the algorithm. First, at points (1a) and (1b), nodes may spontaneously decide to act as seeds of a branch with given probability p. To avoid excess communication, this happens only if there are no developing pre-existing branches in the immediate neighborhood.

At point (2) a neighboring node calls the coloring procedure of a given node. Depending on the situation, the node can either call recursively the coloring procedure of another node (case 2a), signal that everything reachable has been colored (case 2b), or signal that it cannot be colored with any color (case 2c). Returns from the calls to the coloring procedure are taken care of at (3a), (3b) and (3c). The return value can either indicate that everything has been colored (3a) or, if the subtree is uncolorable, `backtrack` is received. In the former case, another emanating subtree can be colored (case 2a), or if uncolored subtrees do not exist, the procedure can return indicating that all reachable nodes have been colored (case 2b). In the latter case, if previously colored subtrees exist, they have to be uncolored (case 3b), and if such do not exist (case 3c), the color used by the node can be either changed and the coloring reattempted (cases 2a and 2b) or uncolorability can be signaled (case 2c).

An `uncolor` message may be received if a node first has received a call to the coloring procedure, has colored itself and all emanating subtrees, returned a return value signaling that everything has been colored, and then the parent needs to backtrack. In such a situation, possible colored subtrees have to be uncolored (case 4a to 3b), or if there are no such subtrees, a value signaling that everything has been uncolored has to be returned (case 4b).

Situations in which a node has called the `uncolor` procedure of a child to uncolor the corresponding subtree, and receives an `uncolor complete` return from this call, are handled in point (5). Either another emanating colored subtree to uncolor might exist (case 3b), or the node selects action depending on whether the `uncolor` call was caused by a `backtrack` command of a child (case 3c) or a `uncolor` command of a parent (case 4b).



Start update.
1a
Are you or one of your neighbors in a coloring branch?
NO
Draw random t in [0,1]
1b
Is t < p?
NO
YES
YES
Rx "color" from higher-branch-ID node?
NO
2
YES
Rx "all colored"?
NO
3a
YES
Rx "backtrack" or "uncolor" ?
NO
4a
YES
5
Rx "uncolor complete"?
NO
YES
Start new coloring branch with your ID. You are your own parent.
Adopt the branch ID of the neighbor sending "color" with highest branch ID, take this neighbor to be your parent.
Remove current child from list of succesful children.
Set of usable colors = all colors
Set of usable colors = colors no neighbor in same branch uses.
Is list of succesful children empty?
YES
NO
3b
3c
Remove current color from set of usable colors.
NO
Is parent expecting Rx to "uncolor"?
YES
4b
Are there usable colors?
NO
2c
YES
Pick a color to current color.
Pick a usable color to current color.
Add current child to list of succesful children.
Do some of your neighbors have a different branch ID?
NO
2b
YES
2a
Remove branch ID.
Remove branch ID.
6
Choose from these the one with highest number of colored neighbors to current child. Solve draws randomly.
Select random neighbor to current child.
Make a succesful child current child.
Tx parent "backtrack".
Tx current child "color".
Tx parent "all colored".
Tx parent "uncolor complete".
Tx current child "uncolor".
End update

Fig. 8. Flow chart of the SOST algorithm.

For the algorithm to be complete, the node starting a branch considers itself its own parent. If a node sends itself `all colored`, the whole graph is colored. If it sends itself `backtrack`, the graph is uncolorable.

The search tree can be amended with heuristics to speed up the algorithm. Here, we implement two such heuristics. First, at point (6) in the procedure, the selection of a node from the list of uncolored neighbors is performed with a heuristic that is likely to speed up convergence—the uncolored neighbor which has the largest number of uncolored neighbors itself is selected for coloring. Possible ties are solved at random. This is supposed to make the convergence faster since it makes the colored regions more compact and less dendritic, reducing the number of hard-to-resolve conflicts arising.

Second, a bound-and-restart protocol (not depicted in Fig. 8) re-initializes the algorithm when the number of conflict resolutions exceeds a predefined threshold. For it, we need two variables, R_i, the counter for the number of backtrack steps performed, and f_i, the largest branch ID of a branch that node i has considered itself to belong to. Both variables are initially set to zero. The counter R_i is passed along with any `color` call. Its value is increased by one every time the return value of the coloring procedure signals uncolorability. When R reaches $R_{\max}$, the algorithm is restarted using the node at which R reached its maximum as the new starting point. The largest branch ID used, f_i, governs the restarting of the coloring procedure. The node observing $R = R_{\max}$ having node identifier r_j, starts a new branch with the branch identifier $\max\{f_i + 1, r_i\}$. This way, the newly started branch will always have a larger identifier than the interrupted branch.

As a consequence of the restarts, the identifiers are no longer unique. This does not render the bound-and-restart protocol unusable, however, but provides a way to get rid of the requirement of uniqueness of the identifiers. Consider a case in which there are two growing branches with the same identifier. Two alternative scenarios may happen. First, the algorithm can by chance still converge. Then, no problem exists. Second, if the algorithm does not converge, either one of the branches will hit the bound-and-restart counter, and start a new branch overriding both earlier conflicting ones. Also, if a node receives `color` commands from two nodes with the same branch ID in the same iteration, this indicates conflicting branches, and is resolved by an immediate restart.

References

[1] W. Hale, "Frequency assignment: Theory and applications," *Proc. IEEE*, vol. 68, no. 12, pp. 1497–1514, 1980.

[2] A. Gamst, "Some lower bounds for a class of frequency assignment problems," *IEEE Trans. Veh. Tech.*, vol. 35, pp. 8–14, 1986.

[3] A. Eisenblätter, M. Grötschel, and A. M. Koster, "Frequency planning and ramifications of coloring," *Discussiones Mathematicae Graph Theory*, vol. 22, no. 1, pp. 51–88, 2002.

[4] K. Appel and W. Haken, *Every planar map is four colorable*, American Mathematical Society, 1989.

[5] I. Katzela and M. Naghshineh, "Channel assignment schemes for cellular mobile telecommunication systems: a comprehensive survey," *IEEE Personal Commun.*, vol. 3, no. 3, pp. 10–31, Jun 1996.

[6] J. Janssen, K. Kilakos, and O. Marcotte, "Fixed preference channel assignment for cellular telephone systems," *IEEE Trans. Veh. Tech.*, vol. 48, no. 2, pp. 533–541, 1999.

[7] 3rd Generation Partnership Project; TSG RAN, "Physical layer aspects for evolved Universal Terrestrial Radio Access (UTRA)," Tech. Rep. TR 25.814, Sept. 2006.

[8] J. Zander, "Radio resource management in future wireless networks: requirements and limitations," *IEEE Comm. Mag.*, vol. 35, no. 8, pp. 30–36, 1997.

[9] A. Simonsson, "Frequency reuse and intercell interference co-ordination in E-UTRA," in *Proc. IEEE VTC* Spring, 2007, pp. 3091–3095.

[10] I. F. Akyildiz, W. Y. Lee, M. C. Vuran, and S. Mohanty, "Next generation / dynamic spectrum access / cognitive radio wireless networks: a survey," *Computer Networks Journal*, vol. 50, pp. 2127–2159, Sept. 2006.

[11] R. W. Thomas, D. H. Friend, L. A. DaSilva, and A. B. MacKenzie, "Cognitive networks: Adaptation and learning to achieve end-to-end performance objectives," *IEEE Comm. Mag.*, pp. 51–57, Dec. 2006.

[12] F. Dressler, *Self-organization in sensor and actor networks*, Chichester: John Wiley and Sons, 2007.

[13] C. Prehofer and C. Bettstetter, "Self-organization in communication networks: Principles and design paradigms," *IEEE Comm. Mag.*, vol. 43, no. 7, pp. 78–85, July 2005.

[14] 3GPP, "Self-configuring and self-optimizing network use cases and solutions," Tech. Rep. TR 36.902, 2008, available: http://www.3gpp.org.

[15] L. G. U. Garcia, K. I. Pedersen, and P. E. Mogensen, "Autonomous component carrier selection: Interference management in local area environments for LTE-Advanced," *IEEE Comm. Mag.*, vol. 47, no. 9, pp. 110–116, Sept. 2009.

[16] Y. Akaiwa and H. Andoh, "Channel segregation-a self-organized dynamic channel allocation method: Application to TDMA/FDMA microcellular system," *IEEE J. Sel. Areas Comm.*, vol. 11, no. 6, Aug. 1993.

[17] J. Janssen, D. Krizanc, L. Narayanan, and S. Shende, "Distributed online frequency assignment in cellular networks," in *Proc. STACS 98: 15th Ann. Symp. on Theor. Aspects of Computer Science*. Springer Verlag, Feb. 1998.

[18] W. Yu, G. Ginis, and J. M. Cioffi, "Distributed multiuser power control for digitial subscriber lines," *IEEE J. Sel. Areas Comm.*, vol. 20, no. 5, pp. 1105–1115, June 2002.

[19] J. Huang, R. A. Berry, and M. L. Honig, "Distributed interference compensation for wireless networks," *IEEE J. Sel. Areas Comm.*, vol. 24, no. 5, pp. 1074–1084, May 2006.

[20] R. Etkin, A. Parekh, and D. Tse, "Spectrum sharing for unlicensed bands," *IEEE J. Sel. Areas Comm.*, vol. 25, no. 3, pp. 517–528, Apr. 2007.

[21] J. O. Neel and J. H. Reed, "Performance of distributed dynamic frequency selection schemes for interference reducing networks," in *Proc. IEEE MILCOM*, Oct. 2006.

[22] N. Nie and C. Comaniciu, "Adaptive channel allocation spectrum etiquette for cognitive radio networks," *Mob. Netw. and Appl.*, vol. 11, pp. 779–797, 2006.

[23] B. Babadi and V. Tarokh, "A distributed asynchronous algorithm for spectrum sharing in wireless ad hoc networks," in *Proc. Conf. Inf. Sci. Sys.*, Mar. 2008, pp. 831–835.

[24] C. Peng, H. Zheng, and B. Y. Zhao, "Utilization and fairness in spectrum assignment for opportunistic spectrum access," *Mobile Netw. Appl.*, vol. 11, no. 4, pp. 555–576, Aug. 2006.

[25] R. Anthony, "Emergent graph colouring," in *Engineering Emergence for Autonomic Systems (EEAS), First Annual International Workshop, at the 3rd Intl. Conf. on Autonomic Computing (ICAC), Dublin, Ireland*, June 2006, pp. 4–13.

[26] L. Cao and H. Zheng, "Distributed rule-regulated spectrum sharing," *IEEE J. Sel. Areas Comm.*, vol. 26, no. 1, pp. 130–145, Jan. 2008.

[27] R. Anthony, "Scalable and efficient graph colouring in 3 dimensions using emergence engineering principles," in *Proc. IEEE Conf. Self-adapt Self-org. Syst.*, Oct. 2008, pp. 370–379.

[28] J.-M. Koljonen, M. Peltomäki, M. Alava, and O. Tirkkonen, "Inertia-based distributed channel allocation," in *Proc. Internat. Wireless Commun. & Mob. Comp. Conf. (IWCMC)*, June 2009, pp. 126–131.

[29] F. Yu, T. Wu, and S. Biswas, "Toward in-band self-organization in energy-efficient MAC protocols for sensor networks," *IEEE Trans. Mob. Comp.*, vol. 7, no. 2, pp. 156–170, Feb. 2008.

[30] N. Robertson, D. Sanders, P. Seymour, and R. Thomas, "Efficiently four-coloring planar graphs," in *Proc. 28th annual ACM symp. on Theory of computing*. ACM New York, NY, USA, 1996, pp. 571–575.

[31] G. Stüber, *Principles of Mobile Communications*, Norwell, MA: Kluwer Academic Publishers, 1996.

[32] G. Voronoi, "Nouvelles applications des parametres continus a la theorie des formes quadratiques," *J. Reine Angew. Math*, vol. 134, pp. 198–287, 1908.

[33] S. Fortune, "A sweepline algorithm for Voronoi diagrams," *Algorithmica*, vol. 2, no. 1, pp. 153–174, 1987.

[34] M. Yokoo, E. Durfee, T. Ishida, and K. Kuwabara, "The distributed constraint satisfaction problem: Formalization and algorithms," *IEEE Transactions on Knowledge and Data Engineering*, vol. 10, no. 5, pp. 673–685, 1998.

[35] M. Yokoo and K. Hirayama, "Algorithms for distributed constraint satisfaction: a review," *Auton. Agents and Multi-Agent systems*, vol. 3, no. 2, pp. 185–270, 2000.

[36] T. Vredeveld and J. K. Lenstra, "On local search for the generalized graph coloring problem," *Oper. Res. Lett.*, vol. 31, pp. 28–34, 2003.

[37] M. Kardar, *Statistical physics of fields*, Cambridge University Press, 2007.

[38] D. Jungnickel, *Graphs, networks and algorithms*, Springer Verlag, 2007.

[39] R. Ahuja, T. Magnanti, and J. Orlin, *Network flows: theory, algorithms, and applications*, Prentice-Hall, New Jersey, 1993.

[40] P. Galinier and A. Hertz, "A survey on local search methods for graph coloring," *Computers & Operations Research*, vol. 33, no. 9, pp. 2547–2562, 2006.

[41] J.-M. Koljonen, M. Alava, M. Peltomäki, and O. Tirkkonen, "Distributed generalized graph coloring," in *Proc. IEEE Conf. Self-adapt Self-org. Syst.*, Sept. 2010, pp. 1–10.