Simulated Annealing Variants for Self-organized Resource Allocation in Small Cell Networks

Furqan Ahmed and Olav Tirkkonen

Aalto University

Abstract

This paper discusses the application of simulated annealing (SA) based metaheuristics to self-organized orthogonal resource allocation problems in small cell networks (SCN)s, for static and dynamic topologies. We consider the graph coloring formulation of the orthogonal resource allocation problem, where a planar graph is used to model interference relations in a SCN comprising of randomly deployed mutually interfering cells. The aim is to color the underlying conflict graph in a distributed way, for which different variants of SA such as SA with focusing heuristic (i.e., limiting the local moves only to the cells that are in conflict), and fixed temperature, are investigated.For static topologies, distributed algorithms are used, in which no dedicated message-passing is required between the cells, except for the symmetrization of conflict graph. To enable distributed SA in dynamic topologies, a distributed temperature control protocol based on message-passing is considered. Different aspects relevant to self-organizing cellular networks are analyzed using simulations. These include the number of cells with resource conflicts, number of resource reconfigurations required by the cells to resolve the conflicts, requirements on dedicated message-passing between the cells, and sensitivity to the temperature parameter that guides the stochastic search process. Simulation results indicate that the considered algorithms are inherently suitable for SCNs, thereby enabling efficient resource allocation in a self-organized way. Furthermore, the underlying concepts and the key conclusions are general, and relevant to other problems that can be solved by distributed graph coloring.

Keywords: Simulated annealing, self-organization, resource allocation, small cell networks, distributed graph coloring

1. Introduction

Self-organization in wireless networking entails functionalities that ensure ubiquitous network connectivity and scalability, whilst guaranteeing the desired quality of service to the served users [1]. In the context of contemporary cellular systems such as Long Term Evolution (LTE)/LTE-Advanced (LTE-A), the self-organizing networking (SON) paradigm encompasses mechanisms for self-configuration, self-healing, and self-optimization [2]. As the cellular networks are becoming increasingly complex due to massive deployments of small cells, a multitude of challenges related to network resource allocation, management, and operation have emerged. SON mechanisms address these challenges by enabling automated optimization of network parameters and reduction in capital/operational expenditure.

The importance of SON in future networks is underscored by the fact that the vision for 5G entails diverse use-cases, involving both cooperative and non-cooperative scenarios. For fully cooperative scenarios, virtualization of network resources is under consideration, which involves abstraction of multiple network management functions to network graphs for software-based control [3]. On the other hand, resource allocation problems in non-cooperative scenarios, such as multiple-operators sharing spectrum in authorized shared access and multiple technologies sharing unlicensed spectrum are inherently more challenging, and will require SON algorithms which do not involve any dedicated message-passing between the nodes. Moreover, in large-scale networks, computation of an optimal resource allocation is prohibitively complex, especially under dynamically changing topologies, as it requires the availability of complete information regarding the network state at every node. Local decisions based on limited information, computation capabilities, and inter-cell signaling are thus inevitable, and motivate the application of self-organizing algorithms.

A number of existing SON mechanisms in cellular networks involve self-organized allocation of orthogonal resources among cells. Notable examples include primary component carrier (PCC) selection [4, 5], physical cell ID (PCI) assignment [5, 6], and the classical frequency assignment problem [7].These can be modeled as graph coloring problems, where vertices represent the cells and colors are the available resources. The aim is to color the underlying interference graph such that no two adjacent cells use the same resource. Thus, resource allocation problems are of due importance for both contemporary and future cellular networks, and mandate the study of new

self-organizing algorithms, tailored to the requirements of SCNs. Generally, self-organized resource allocation in wireless networks involves computing a solution to an underlying network optimization problem, using distributed algorithms [8, 9]. For discrete problems such as orthogonal resource allocation on an interference graph, combinatorial optimization methods involving metaheuristics are an attractive option. Accordingly, distributed algorithms based on metaheuristics can pave the way for engineering self-organizing solutions to discrete resource allocation problems in SCNs.

In this paper, we study different variants of the Simulated Annealing (SA) metaheuristic for self-organized resource allocation in SCN, under static as well as dynamic topologies. To this end, different parameters and performance indicators important from a perspective of SCNs are taken into account, in the design and analysis of algorithms. It is assumed that the resources are orthogonal, which leads to a generic system model applicable to a number of SON problems. The algorithms investigated are SA with *focused search* enhancements [10], which focuses uphill and/or plateau moves only on the cells which are in conflict with their respective neighbors. Furthermore, the overall concept can be considered as a generic framework for distributed resource allocation under limited information and dynamic topology. In the remaining part of this section, we give a brief description of the state-of-the-art of SA based methods in wireless networks, followed by a summary of the contributions of this work.

1.1. Related work

The well known SA based optimization methods essentially balance *exploration* and *exploitation* to search the solution space in an efficient manner. In particular, the hill-climbing feature and the plateau moves play a key role in escaping from local minima and long plateaus, whereas downhill moves ensure attraction to the global optimum. Frequency of uphill moves is controlled via a temperature parameter which is set before-hand, and is reduced according to a cooling schedule as the algorithm progresses [11].

In wireless networking domain, SA has been discussed for distributed channel allocation in wireless local area networks [12], where each access point is assigned a sigmoidal utility function, parametrized by the interference level it experiences. A similar approach proposed in [13], minimizes the total interference in the network, and is shown to outperform the Leith and Clifford algorithm [14]. SA for uplink power control in LTE-A networks is discussed in [15], and for power optimization of pilot signals in [16, 17]. Other resource

allocation problems in cellular networks addressed using SA include downlink scheduling in LTE networks [18], handover parameter optimization in LTE networks [19], and antenna-tilt optimization [20].

To develop the SA based methods for the orthogonal resource allocation problem, we consider the graph coloring model, where the aim is to assign resources to cells in a non-conflicting way. The minimization of resource conflicts in the network translates to mitigation of cross-channel interference between the cells. This work can be considered as an extension of [12, 13], however, our focus is on the soft computing aspects and different variants of SA, which are pertinent to self-organization in SCNs.

1.2. Contributions

We seek to minimize the number of resource conflicts among the cells, as well as the number of resource reconfigurations in the SCN, for both static and dynamically changing topologies. It is important from the perspective of a SCN that both the cells with conflicting resources, and total reconfigurations incurred while resolving those conflicts, are minimized. The first SA variant, we discuss, comprises of SA metaheuristic combined with the *focused search* mechanism [21, 10], which allows uphill/plateau moves only to the cells that are in conflict. Existing works discuss SA with a cooling schedule, where temperature is considered as a network parameter. This introduces a centralized component in the procedure, and is therefore not suitable for SCNs which often have dynamic topologies. Thus, we introduce the fixed temperature alternative [22, 23], which is *fully distributed* as it does not require any cooperation between cells. It is observed that the fixed temperature variant performs better than the standard SA algorithm. The second variant is motivated by the practical issue of minimizing the reconfigurations of cells in a network, and it involves focusing plateau moves only on the cells that are in conflict. Similar variants that involve combined focusing of plateau moves and uphill moves are also considered. To determine the optimal noise strategy (temperature) for coloring problem, we compare fixed temperature SA variants to their cooling counterparts across a wide range of temperatures. Furthermore, we discuss distributed temperature control protocols for applying SA to dynamic networks. The discussed algorithms have different characteristics related to plateau moves, uphill moves and downhill moves, which results in varying performance in different scenarios, when applied to the graph coloring problem.

The rest of the paper is organized as follows: Section II introduces the system model along with an overview of distributed graph coloring concept, and properties of planar graphs. Section III involves discussion on SA for orthogonal resource allocation, and related self-organizing algorithms for SCNs. In Sections IV and V, comparison of the considered algorithms is carried out by simulations, for static and dynamic networks, respectively. Finally, conclusions are given in Section VI.

2. System Model

The motivation for applying metaheuristics for self-organized allocation of orthogonal resources (e.g., channels, PCCs, PCIs) in cellular networks stems from the fact that in such cases graph coloring models can be readily applied. In fact, graph coloring is a well studied problem, and both centralized as well as distributed graph coloring approaches exist. These approaches may be applied to SCNs, provided that the assumptions on computation and inter-cell communication aspects of the system are reasonable. In this regard, the distributed graph coloring approaches have gained considerable popularity, because of their promising features well suited to the needs of SCNs.

2.1. Distributed Graph Coloring

Graph coloring is an NP-complete combinatorial optimization problem, with a wide range of applications, see for example [24, 25]. Centralized coloring approaches based on SA are discussed in [26, 27]. Another important metaheuristic is Tabu search [28], which avoids being trapped in local minima by maintaining a list of bad moves, and updating them during iterations. For large graphs, pure local search methods may not work efficiently. Such graphs can be colored by using stable set extraction as a first step, followed by using local search on residual graph. Another important approach is genetic optimization combined with local search, which leads to hybrid algorithms [29, 30]. A detailed survey of local search algorithms for graph coloring, along with classification of different local search strategies is given in [31]. The distributed approaches have seen a steady rise in popularity due to their practical applications in different areas, for details see [32], and references therein. For distributed coloring, local search methods as well as distributed constraint satisfaction algorithms are relevant [33, 34]. Fully distributed algorithms are particularly important for self-organized coloring. These algorithms work in a way that each node of the graph makes

the decisions regarding its color, on the basis of local information only. This motivates their application for self-organized resource allocation in the SCNs.

2.2. Network Model: Interference Graphs and Voronoi Tessellations

We consider a planar graph model for a SCN comprising of low power cells deployed randomly, in a given geographical area. A cell considers a neighboring cell to be an interferer if the interference received from it is greater than a given threshold. The threshold models the measurement and reporting capabilities of the users served by the cell. The neighbor relations among the cells are based on the mutual interference which is predominantly dependent on their spatial separation. Thus, an interference graph can be created via thresholding, where the cells are the vertices, and the edges represent the interference couplings between them. These interference couplings may be symmetrized in the cellular networks in which a backhaul connection exists between the cells (e.g., LTE/LTE-A). The resulting interference graph is undirected, so that a conflict can be seen by both cells. Alternatively, for SCNs deployed on a 2D plane, an effective approach is to use planar graph model [35, 36]. This model is particularly accurate under the assumptions that non-distance dependent propagation effects such as shadow fading are mild. Consequently, the interference couplings are considered only between the cells that are neighbors geographically, as they would be measured and reported as strong potential interferers by the users. In this case, each user in the network coverage area will connect to the closest base-station (cell), i.e., the base-station with the minimum Euclidean distance, whereas the points equidistant from multiple base-stations may pick their serving base-station randomly. This results in a Voronoi tessellation of the coverage area, which is planar by definition. In order to create such graph, we consider a square-shaped coverage area and drop N_v points $v \in \mathcal{V}$ at random in it, and compute the Voronoi tessellation corresponding to those points. The coverage area is split into Voronoi cells, where each cell constitutes the area consisting of the points that are closest to its base-station. The points $v \in \mathcal{V}$ are taken as the vertices of the created graph $G(\mathcal{V}, \mathcal{E})$, whereas edges $e \in \mathcal{E}$ connect the vertices of the cells that share the common boundary. Figure 1 shows an example instance of a planar graph generated by this method. It can be seen that no two edges cross, and the graph is colorable by four colors, which is a distinct characteristic of planar graphs [37].

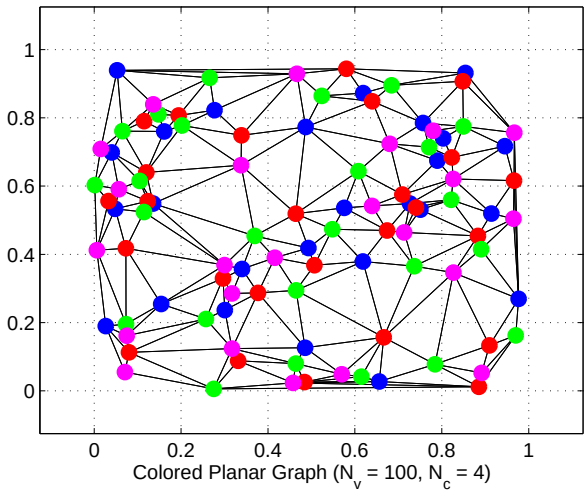

Figure 1: A planar graph with $N_v = 100$ vertices that represent small cells, colored with $N_c = 4$ colors, that can be regarded as orthogonal resources assigned to the cells in a non-conflicting way.

2.3. Problem Formulation

In graph coloring terminology, a $N_c-coloring$ for a given number of colors N_c, is defined as function $col : \mathcal{V} \rightarrow \{1, \ldots, N_c\}$. Two adjacent vertices x and y are said to be conflicting if they have the same color, i.e., $col(x) = col(y)$. A N_c-coloring is defined to be *legal* if the colors are assigned to vertices in a way that there are no conflicting edges. For the resource allocation problem under consideration, we are interested in legal N_c-coloring of G, where N_c represents the number of orthogonal resources, and G is planar and takes into account the interference couplings in a SCN.

3. Simulated Annealing Algorithms

3.1. Local Search: Downhill, Uphill and Plateau moves

We discuss distributed algorithms based on local search mechanism, which involves local changes by individual cells to move in the configuration space from one solution to another. The aim is to reach a conflict-free state, where no two neighboring cells are using the same resource. To this end, the search space is explored by making perturbations to the existing solution, known

as local moves. In this case, a local move in the configuration space is the change of a resource by one cell, and the neighbors in configuration space are two states that are connected by a local move. Thus, the configuration space is searched by a sequence of local moves taken by cells. A local move that reduces the conflicts is known as downhill move. The key feature of configuration space in coloring conflict graphs is the existence of plateaus, i.e., neighboring states with the same number of conflicts. A greedy local search in which each cell picks the best move gets trapped in a local minimum, which is often located at plateaus. A main feature of stochastic local search algorithms is that they can move on the plateaus, and avoid entrapment in local minima. In this context, a plateau move is a local move, in which the number of conflicts remain unchanged. Plateau moves are therefore important for escaping from local minima, in search of a global optimum. An effective strategy to further improve the performance of stochastic local search algorithms is to enable uphill moves occasionally, i.e., accepting local moves that increase the cost (number of conflicts). Moreover, the probability of uphill moves is controlled by a noise parameter, which is temperature for the SA algorithms we discuss here. The performance of algorithm is strongly dependent on the value chosen for the temperature. Hence, finding an optimal set of parameters is also an interesting problem [10, 38].

We focus on distributed graph coloring algorithms based on SA, where each cell performs an SA update to select a resource, while avoiding conflicts with the neighboring cells. In all cases, we consider asynchronous and periodic updates, where each cell updates on its turn. The standard SA algorithm has been discussed for channel selection problem in [12, 13]. Here, we consider several alternatives.

Consider cell x using resource c. The number of conflicts of cell x (number of neighbors which are using the same resource) is $\mathcal{F}(c)$. The cell may or may not be in conflict with neighbors while using resource c. On its turn to update, it picks a resource $c' \in \mathcal{C} \backslash c$ randomly and evaluates its cost in terms of conflicts, given by the cost function $\mathcal{F}(c')$. If $\Delta = \mathcal{F}(c') - \mathcal{F}(c) < 0$, cell x selects resource c', otherwise it selects resource c' with the probability $e^{-\Delta/T}$. Thus, the higher is the cost of taking the uphill move, the lower is the probability of it being accepted. The temperature parameter T is reduced according to some cooling schedule such as $T(t) = T_0/\log_2(2+t)$, where t is the update time (or iteration) and T_0 is the initial temperature. The algorithm is summarized as Algorithm 1. Note that the parameter T here is assumed to be same for all the cells $v \in \mathcal{V}$. For a fully distributed

ALGORITHM 1: SA

Cell v using resource c selects a resource $c' \in \mathcal{C} \backslash c$ and computes:
$\Delta = \mathcal{F}(c') - \mathcal{F}(c)$.
if $\{\Delta \leq 0\}$ **or** $\{rand(1) < e^{-\Delta/T}\}$
then
    $c \leftarrow c'$
else
    $c \leftarrow c$
end
Update the temperature parameter: $T(t) = \frac{T_0}{\log_2(2+t)}$.

ALGORITHM 2: SA with focused uphill (SAFU)

Cell v using resource c selects a resource $c' \in \mathcal{C} \backslash c$ and computes:
$\Delta = \mathcal{F}(c') - \mathcal{F}(c)$.
if $\{\Delta \leq 0\}$ **or** $\{\mathcal{F}(c) > 0$ **and** $rand(1) < e^{-\Delta/T}\}$
then
    $c \leftarrow c'$
else
    $c \leftarrow c$
end
Update the temperature parameter: $T(t) = \frac{T_0}{\log_2(2+t)}$.

implementation, it can be fixed to $T = T_0$. The key feature of this algorithm is that when a cell is in conflict, it always takes downhill and plateau moves. Moreover, it also takes uphill move with the probability $e^{-\Delta/T}$. On the other hand, when the cell is conflict-free, it again accepts all plateau moves and also the uphill moves with probability $e^{-\Delta/T}$.

3.2. Focused Uphill moves

In SA, we see that a cell may accept uphill moves from both conflict-, and conflict-free states. Following [21, 10], we introduce a focused search feature to make the search more effective by limiting the number of uphill moves accepted. Only the cells that are in conflict accept uphill moves with probability $e^{-\Delta/T}$, conflict-free cells accept plateau moves only. Thus, the focus is only on the cells that are in conflict, hence the name, *focused search*. An important feature induced by this search strategy is that the algorithm does not escape the conflict-free state, i.e., the global optimum, as no uphill moves are accepted by any of the cells. This property, where the global optimum

ALGORITHM 3: SA with focused plateau (SAFP)

Cell v using resource c selects a resource $c' \in \mathcal{C} \backslash c$ and computes:
$\Delta = \mathcal{F}(c') - \mathcal{F}(c)$.
if $\{\mathcal{F}(c) > 0$ **and** $\Delta \leq 0\}$**or** $\{rand(1) < e^{-\Delta/T}\}$
then
    $c \leftarrow c'$
else
    $c \leftarrow c$
end
Update the temperature parameter: $T(t) = \frac{T_0}{\log_2(2+t)}$.

ALGORITHM 4: SA with focused uphill and plateau (SAFUP)

Cell v using resource c selects a resource $c' \in \mathcal{C} \backslash c$ and computes:
$\Delta = \mathcal{F}(c') - \mathcal{F}(c)$.
if $\{\mathcal{F}(c) > 0$ **and** $\Delta \leq 0\}$**or**$\{\mathcal{F}(c) > 0$ **and** $rand(1) < e^{-\Delta/T}\}$
then
    $c \leftarrow c'$
else
    $c \leftarrow c$
end
Update the temperature parameter: $T(t) = \frac{T_0}{\log_2(2+t)}$.

becomes an absorbing state is often referred to as an *absorbing minimum*. The SA algorithm with focused search enhancement and cooling schedule is summarized as SAFU in Algorithm 2, whereas the fixed temperature version SAFU-Fix can be obtained by simply setting $T = T_0$. In SAFU algorithm metropolis dynamics have been considered, in conjunction with the cooling schedule. The SAFU-Fix is same as the FSAM algorithm presented in [36], albeit it is based on metropolis dynamics.

3.3. Focused Plateau moves

In the distributed graph coloring algorithms we consider here, one parameter that is particularly relevant to the resource allocation problems for SCNs is the total number of resource reconfigurations taken to reach the conflict-free state. From a practical standpoint, it is important to minimize these, as the reconfiguration essentially could mean rebooting of the cell. Moreover, change of resource in one cell could set off a chain of reconfigurations in the whole network. Therefore, we consider an alternative strategy

with reduced plateau moves, where a cell always accepts plateau moves, only if it is in conflict. Uphill moves are also accepted from conflict states with probability $e^{-\Delta/T}$. From the conflict-free state, uphill and plateau moves are both accepted with probability $e^{-\Delta/T}$. The price of reduction of plateau moves is an increase in the possibility of getting stuck in local minima. The SA algorithm with focused search enhancement for plateau moves is summarized as SAFP in Algorithm 3. Moreover, the focused search principle for uphill and plateau moves can be combined as SA with focused uphill and plateau (SAFUP) algorithm given as Algorithm 4. In this variant, only cells with conflicts will reconfigure their resource, thereby resulting in a significant reduction in total number of reconfigurations in the network.

3.4. Temperature: Fixed vs. Cooling

The probability of accepting an uphill move depends on the temperature parameter, and the cooling schedule. The initial value strongly influences the algorithm performance, and a suitable value can be found by trying different starting points. The systematic tuning of temperature parameter for stochastic local search algorithms is discussed in [10, 38]. As temperature T is a network parameter, with the cooling enabled, the algorithm is not fully distributed, as the cells would require a common reference to start their clocks. To make the algorithm *fully distributed* (i.e., with no inter-cell dedicated message-passing), it can be kept constant. The fixed temperature SA has been explored for other types of problems in [22, 23]. It is worth noting that for dynamic networks, the application of SA variants with cooling requires implementation of temperature control protocol. A cell joining the network may need to acquire the network timing or temperature from its neighboring cells by message-passing. Distributed temperature control can be enabled via message-passing, at an obvious cost of an information exchange overhead between the cells. However, such approaches are not fully distributed and thus, not suitable from a self-organization perspective. This point is further elaborated in Section V, where we develop temperature control protocols for SA in dynamic networks.

4. Performance in Static Networks

4.1. Simulation Scenario

In order to analyze the performance of algorithms in a static scenario, the first step is to generate a planar graph according to principles enunci-

ated in Section II. As the network is static, the adjacencies of graph remain fixed, while the cells run the algorithms aimed at avoiding resource conflicts with their respective neighbors. The algorithms are run till convergence to a conflict-free (i.e., colored) state, or for a maximum number of iterations MaxIters. A range of temperatures is considered to determine the optimal performance, and its sensitivity to the temperature settings. The starting point of all algorithms is a randomly generated uncolored state, which is the same for all algorithms, across the whole range of temperatures. The parameters evaluated after each run, are the average number of conflicts per cell per iteration, average number of resource reconfigurations per cell per iteration, and convergence probability. In all simulations, planar graphs of size $N_v = 100$ cells are used, whereas the number of resources used to color the graph is $N_c = 4$. The maximum number of iterations is MaxIters = 1000, and the statistics are averaged over 250 randomly generated planar graph instances. In total, there are eight algorithms that we compare here. These include fixed temperature ($T = T_0$) and cooling ($T(t) = T_0/\log_2(2+t)$) variants of SA, SAFU, SAFP, and SAFUP, as summarized in Algorithms 1 through 4. The cooling schedule used for temperature reduction is illustrated in Fig. 2 for the whole range of temperature considered.

4.2. Simulation Results

Each cell carries out an update on its turn, where the updates are asynchronous and periodic, i.e., the cells update their channels in a sequential manner. Figure 3 shows the probability of reaching a conflict-free state against a given range of temperatures. The corresponding average number of conflicts per cell per iteration are illustrated in Fig. 4. Performance depends strongly on the selection of initial temperature. The cooling variants, which involve temperature reduction, are effective in the high temperature range, when compared to their fixed temperature counterparts. Thus, reaching conflict-free state is possible with both fixed temperature and cooling. For the fixed temperature variants, the algorithms perform well in temperature range in which the number of uphill and plateau moves is reasonable enough to escape local minima. Increase of temperature beyond that results in too many conflicts due to a large acceptance probability of uphill moves, which are impossible to resolve. On the other hand, in cooling variants, the temperature falls and the uphill moves are reduced gradually, and consequently, the algorithm converges. Nevertheless, with too high temperature, an algorithm may fail to converge, if the number of iterations are not sufficient.

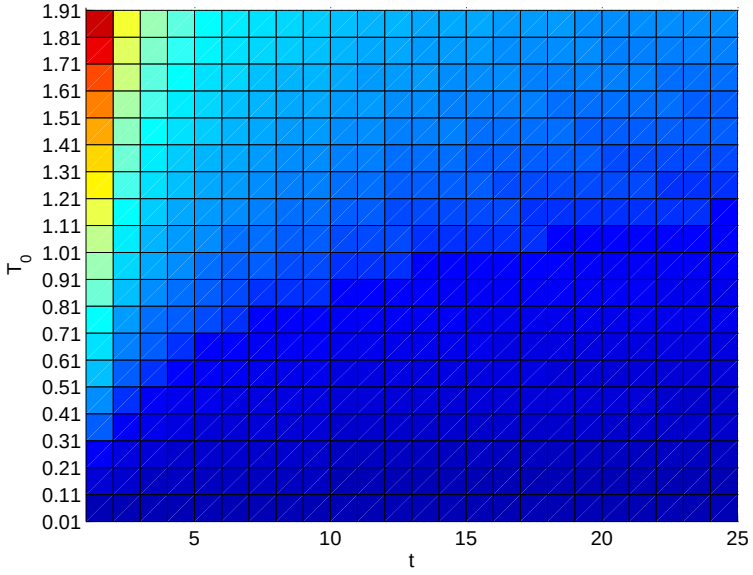

Figure 2: SA involves gradual reduction of temperature according to the cooling schedule $T = \frac{T_0}{\log_2(2+t)}$ shown here, where t is the iteration index and T_0 is the initial temperature.

If we compare SA-fix with SA-cool, we observe that SA-fix is very sensitive to the temperature as its convergence probability plummets to ≈ 0.15 whereas SA-cool degrades gracefully. As shown in Fig.4, SA-fix at temperature 0.21 (temperature at which convergence probability is 1) results in lower number of conflicts than SA-cool at temperature 1.21, because SA-cool requires a high initial temperature which results in a large number of uphill moves in the initial iterations leading to a surge in number of conflicts that dies down gradually and results in a conflict-free state. However, the number of conflicts are still large when compared to SA-fix. A similar trend can be observed in comparison of SAFU-fix and SAFU-cool as well, i.e., SA variants with fixed temperature outperform their cooling counterparts at their respective optimal temperatures.

Moreover, the SAFU-fix outperforms both SA-fix and SA-cool significantly. For example, the optimal temperature of SA-fix is 0.21, and the corresponding values of conflicts and reconfigurations are 0.00815 and 0.00726 respectively. Compare this to SAFU-Fix which has an optimal tempera-

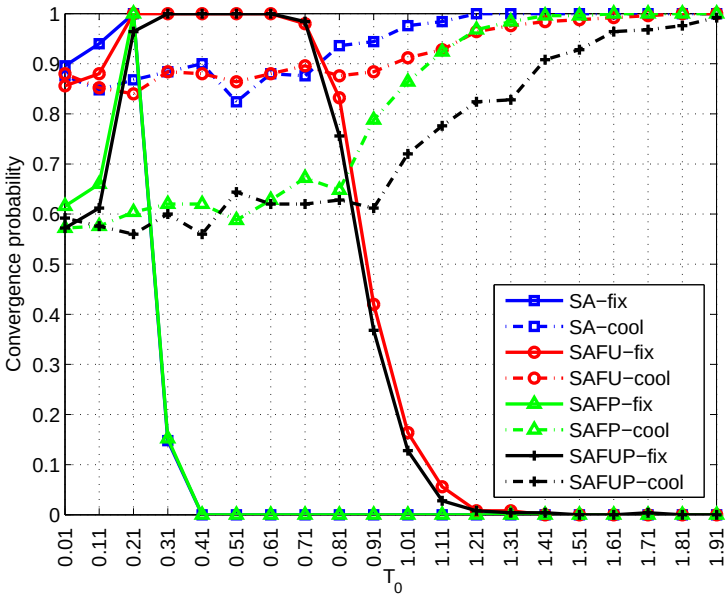

Figure 3: Comparison of SA variants in terms of convergence (to conflict-free state) probabilities against a range of initial temperatures, in a static SCN.

ture of 0.31, and the corresponding conflicts and reconfigurations equal to 0.00552 and 0.00474 respectively. Thus, we see that the number of conflicts and reconfigurations for SA-fix are 50% higher than that of SAFU-fix. The gains yielded by focused search enhancement in SAFU-fix are even more pronounced against SA-cool. Moreover, it is evident that SAFU-fix is less sensitive against temperature settings as compared to SA-fix, and is therefore easy to configure and more suitable for practical implementations.

Similar to SAFU, the SAFP-fix and SAFP-cool apply focusing principle on the plateau moves, i.e., only the cells with conflicts are allowed plateau moves. As illustrated in Fig. 5, it is a viable option for reducing the reconfigurations in the network. This reduction comes at a cost of increase in number of conflicts. With a suitable temperature, the convergence behavior of SAFP-fix mimics SA-fix for obvious reasons—both work in the similar way, except for the fact that disadvantage of not having plateau moves by conflict-free cells results in delayed convergence for SAFP-fix, due to which the number of conflicts rise. The characteristics of SAFU and SAFP are merged in SAFUP,

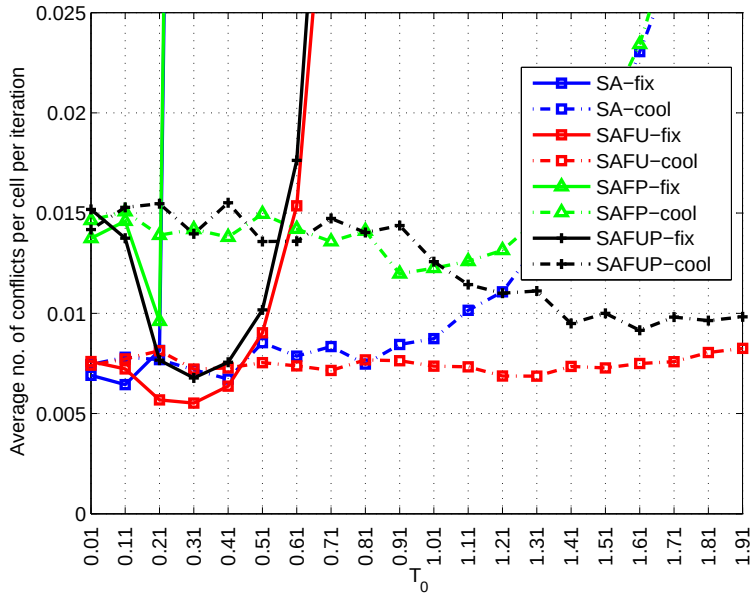

Figure 4: Comparison of SA variants in terms of number of conflicts against a range of initial temperatures, in a static SCN.

which for SAFUP-fix results in almost same number of conflicts as SA-fix, but the number of reconfigurations is significantly lower. Thus, SAFUP-fix outperforms SA-fix significantly in reducing reconfigurations, and convergence properties with a slight increase in number of conflicts. Therefore, it can be concluded that for the static networks, the fixed temperature and the focusing variants perform substantially better than the standard SA.

We observe that despite SA-fix and SAFP-fix exploring a larger fraction of the configuration space than their cooling and focused uphill versions, the probability to find the optimum, even with centralized termination (stopping the algorithm when a solution is found), is worse. The reason is that the cooling and focused uphill versions concentrate the search in configuration space to the vicinity of the optimum. Finally, it should be noted that the focused uphill moves act effectively as a distributed way of algorithm termination.

4.3. Cooling Schedules and Rates

The motivation behind using the standard logarithmic schedule [39], is the fact that it performs well across a broad class of problems. However, it

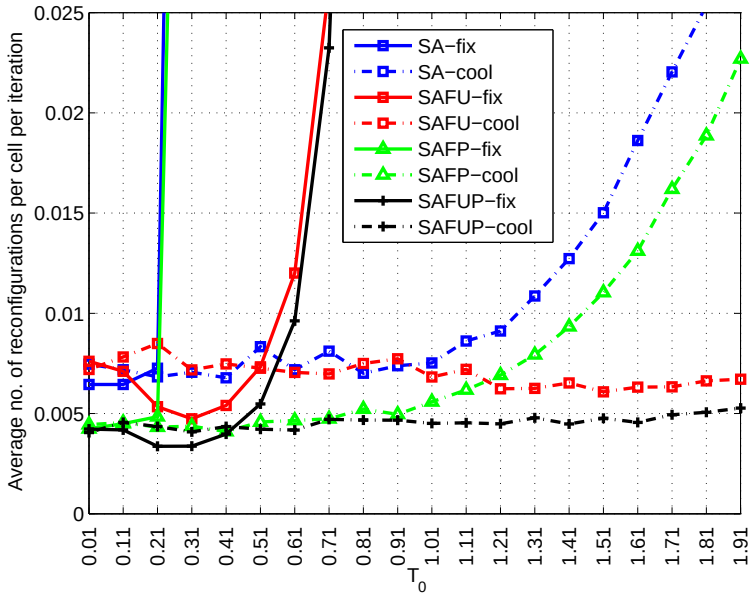

Figure 5: Comparison of SA variants in terms of number of reconfigurations (or cell reboots) against a range of initial temperatures, in a static SCN.

seems worthwhile to compare it to linear and exponential cooling schedules discussed in [40, 41]. For an initial temperature T_0, the linear schedule is given by $T(t) = T_0 - \eta t$, and the exponential schedule is defined as $T(t) = T_0 \alpha^t$, with $0 \leq \alpha \leq 1$. It is worth mentioning that the rate of temperature decrease may also have an impact on the performance of SA algorithms. Therefore, different cooling rates are considered for each of the three schedules. The parameters η and α can be used to control the rate of temperature change in the linear and exponential cases, respectively. For logarithmic schedule, we introduce a rate control parameter β such that $T(t) = \frac{T_0}{\log_2(2+\beta t)}$. The linear schedule is considered with $\eta = 0.05$ and $\eta = 0.015$, exponential schedule with $\alpha = 0.95$ and $\alpha = 0.99$, and logarithmic schedule with $\beta = 1$ and $\beta = 4$. The performance of SAFU-cool variant is analyzed against the number of iterations, as it is a fully distributed algorithm, where we do not need any centralized view to characterize converged states. Results for different cooling rates for each of the three cooling schedules are shown in Fig.6. It can be seen that the logarithmic schedule outperforms linear and exponential cooling

schedules, and converges to an optimal solution within the fixed number of iterations. Moreover, both exponential and linear cooling schedules results in higher number of conflicts in the beginning as well, resulting in higher cumulative number of conflicts.

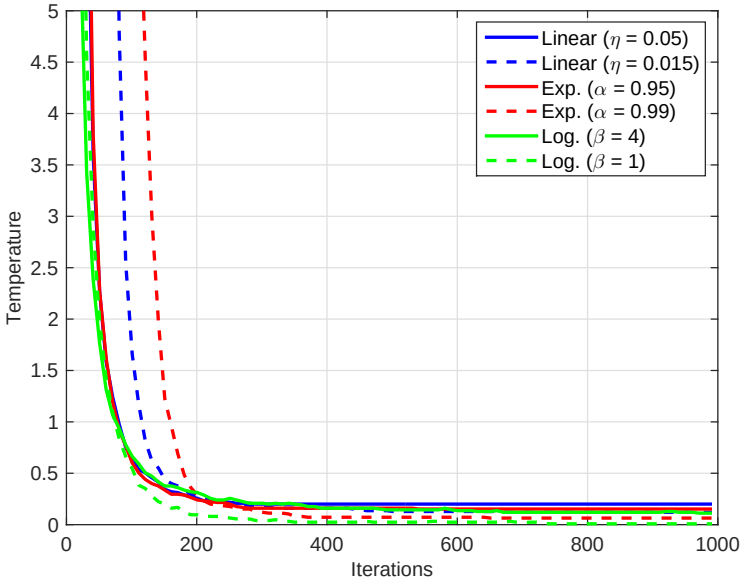

Figure 6: Performance comparison of linear, exponential, and logarithmic cooling schedules for SAFU-cool.

5. Performance in Dynamic Networks

5.1. Simulation Scenario

Next, we consider a dynamic scenario where the cells can join/leave the network at random. In this case, it is assumed that the network is operational and cell appears at a random location in the coverage area of network. The new cell determines its neighbor relations on the basis of measurements done by the served users, and establishes backhaul connection with them. Then, it starts running the resource allocation protocol to avoid selecting the resource that is being used by any of its neighbors. The planar graph for this case is generated in the same way as in the static network case, i.e., by creating a

Voronoi tesselation of coverage area. However, the starting point is a colored state, as it assumed that the network is initially operational and there are no conflicts among neighbors. When a new cell appears, conflicts may arise due to the creation of new adjacency relations. Similarly, the departure of a cell results in a change in adjacency relations and may result in appearance of new conflicts, eventhough it reduces the total number of cells in the network. The cells arrive and depart from the network according to a Poisson birth/death process. The birth rate is λ, and the death rate is $\mu = (N_t\lambda)/N_u$, where N_t is the currently existing number of cells in the network. This keeps the total number of cells in the network close to the initial number of cells N_u.

For a given temperature, the algorithms are run for a certain number of iterations $M \leq \text{MaxIters}$, which depends on the birth/death rate parameters, and thus it models the time after which the network is changed. After M iterations, a new cell appears or a randomly chosen cell vanishes which is reflected by a change in the adjacency matrix, a new M is determined, and algorithms are run again with updated parameters. The process is repeated until $M = \text{MaxIters}$, after which the statistics related to conflicts and reconfigurations over iterations are collected for all algorithms. Note that in this case the network is randomly changing, and the network may or may not be in a converged state when the maximum number of iterations is reached. Thus, we will only evaluate the average number of conflicts and reconfigurations, per cell per iteration. The idea here is to capture the performance of an online algorithm, in a network snapshot with infinite time. Thus, SA with cooling as such does not make sense, as the temperature would always be approaching 0. We address SA with distributed temperature control, so that the temperature is raised whenever the network topology changes.

5.2. Distributed Temperature Control

The SA algorithms that involve cooling are inherently complicated to implement in dynamic settings, especially in the distributed networks where there is limited cooperation among network elements. Ideally, all cells $v \in \mathcal{V}$ in the network must have the same temperature at all times, i.e., $T_v = T$. However, in a dynamic network, when a cell joins or leaves the network, the graph essentially changes, and re-coloring will be required to resolve the conflicts that might appear. Re-coloring in this case should constitute re-run of the algorithms with temperature reset to the initial temperature for all cells $T_v = T_0$. Moreover, from a SCN perspective, it is important to achieve this in a distributed way, with limited or no cooperation between the

cells. For enabling temperature dependent distributed temperature control in the network, we discuss the following cooperative and non-cooperative approaches.

5.2.1. Cooperative global reset message-passing (CMP)

The CMP is based on local cooperation between the neighboring cells. When a new cell joins the network, it starts with the initial temperature of T_0, regardless of the current network temperature T. If a cell i detects a change in its neighbor relations, it re-initializes its temperature to $T_i = T_0$, and sends a temperature reset message to all its neighbors. A cell j, upon receiving temperature reset message originated by cell i, resets its temperature to the current temperature of T_i, and sends a temperature reset message to its neighbors. This eventually leads to a consensus of all cells on a network temperature $T_v \to T$, which keeps on decreasing until the network changes again. It should be noted that the price for achieving consensus on temperature in the whole network is the message-passing overhead. Nevertheless, the protocol enables the extension of SA algorithm to dynamic and distributed networks. The execution of CMP leading to the convergence to a common temperature in a dynamic network is illustrated in Fig. 7. First the network changes at $t = 19$, which generates messages which lead to network consensus on temperature. The same procedure is carried out for subsequent addition and deletion, which occurs at $t = 29$ and $t = 48$, respectively. It can be seen that in all cases, the cells in the network acquire the same temperature after exhanging messages for some iterations.

5.2.2. Non-cooperative local reset (NCR)

To complement the message-passing protocol CMP, we consider an alternative NCR with no messaging. There is no explicit cooperation between the cells in NCR, and no messages are generated. However, cells are capable of discovering new neighbors. A cell i re-initializes its temperature to $T_i = T_0$ only if it detects a change in its neighborhood. The underlying idea is to re-color only those parts of the graph which have changed. Because of the colored initial state, most of the graph would still be conflict-free. Therefore, the propagation of messages in the network to achieve a common temperature does not occur this case, and cells may have different temperatures. Consequently, the conflict-free cells that do not detect any neighborhood change have zero temperature, and the reconfigurations due to the uphill

moves do not spread in the whole network. The simulation results presented next provide a comparison of these approaches.

5.3. Simulation Results

Initially, the number of cells in the graph is $N_v = 100$. The key parameters such as the temperatures, and maximum number of iterations are the same as the ones that were used for the static case. Results are averaged over 250 randomly generated network instances. The performance of SA-cool and

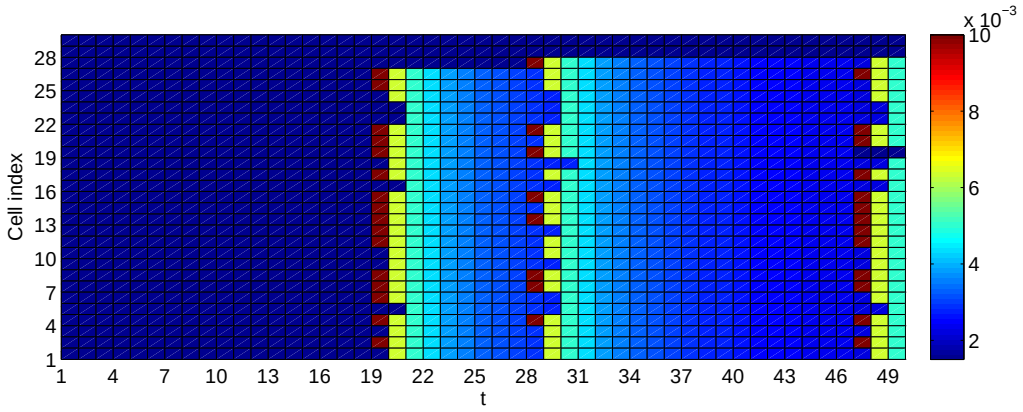

Figure 7: Distributed temperature control based on cooperative global message-passing (CMP) for enabling SA in dynamic SCNs.

SAFU-cool, in terms of number of conflicts is given in Fig. 8, for both CMP and NCR. Here, $\lambda = 0.1$ corresponds to fast changes in network topology compared to $\lambda = 0.01$, and therefore, results in an overall higher number of conflicts. First, we consider standard SA, which is effectively represented by SA-cool(CMP), for the dynamic topologies, as it combines SA-cool with a message-passing based temperature control which ensures the same temperature in the whole network. On the other hand, in NCR, only cells that detect a change in network topology raise their temperature. It can be seen that CMP results in performance deterioration, when compared to NCR, as the temperature rise throughout the network leads to higher probability of uphill move at every cell. A direct consequence of this is that cells with conflicts allow uphill moves with higher probability, whenever network topology changes. This is counter-productive, as the graph is mostly in a colored state already, and raising temperature means enabling unnecessary uphill moves.

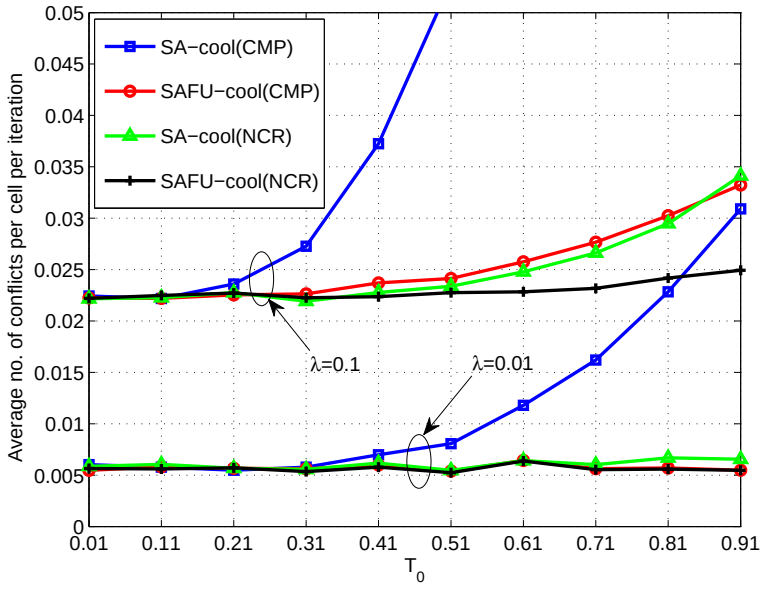

Figure 8: Comparison of SA variants in terms of number of conflicts against a range of initial temperatures, in a dynamic SCN.

Temperature control by NCR provides an effective alternative, where a limited number of cells raise their temperature (i.e., reset to T_0) to enable uphill moves, and this happens in the region where a change in network topology occurs. The key idea here is to re-color only the sub-graph that has changed, due to the appearance or vanishing of a cell. In the colored parts of the graph, the temperature remains zero, and there are no uphill moves.

An alternative strategy for reducing the unnecessary uphill moves is through the focused search variants. For example, in SAFU-cool (CMP), the temperature is raised but the uphill moves will not be enabled, if the cell is not in conflict. In addition, the SAFU-cool (NCR) combines focused search with the NCR, and is the most conservative in allowing the uphill moves. Hence, focused search variants, SAFU-cool (CMP) and SAFU-cool (NCR), perform better than their respective counterparts, for both CMP and NCR, as expected. This trend is more pronounced for $\lambda = 0.1$, as more conflicts arise in a fast changing network leading to an overall high network temperature, which results in more uphill moves. For $\lambda = 0.1$, the SAFU-cool(CMP)

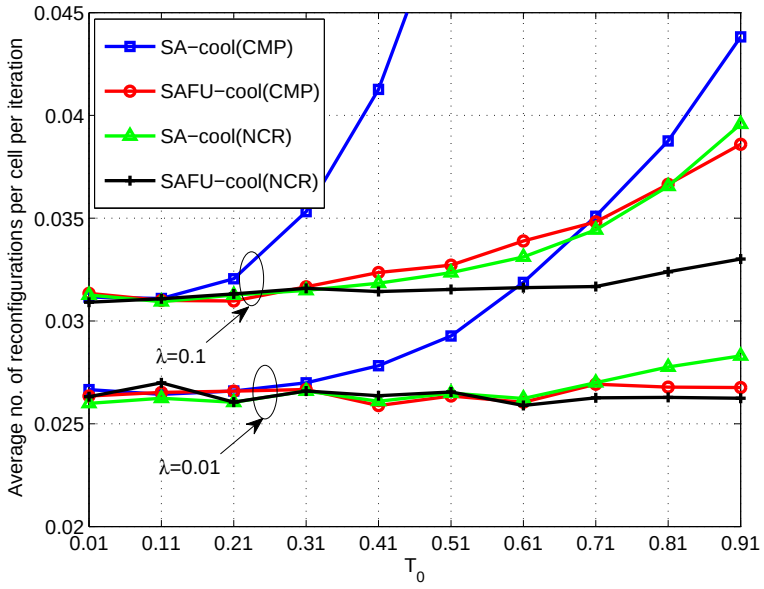

Figure 9: Comparison of SA variants in terms of number of conflicts against a range of initial temperatures, in a dynamic SCN.

gradually gets surpassed by SA-cool(NCR), which does not involve focused search and its number of conflicts eventually rise, as the temperature is increased. Both standard SA type network wide temperature raise, and lack of focused search leads to a significant deterioration in performance as the temperature is increased. The effect is more pronounced when the network topology is changing rapidly. Nevertheless, the focused search variants are robust compared to standard SA, as they are less impacted by increase in temperature and frequency at which network changes. Similar trends can be observed in the comparison of average number of reconfigurations per cell per node, shown in Fig. 9. The SAFU-cool(NCR) consistently performs better, especially, in the higher temperature range.

6. Conclusions

We investigate different variants of simulated annealing (SA) algorithm for graph coloring based self-organized resource allocation, in static and dynamic small cell networks (SCN)s. These include focused search enhance-

ment for SA, which involves constraining uphill/plateau moves only to the cells that are in conflict with the neighbors. This appears to be a more effective approach than standard SA, for minimizing the number of conflicts, as well as the resource reconfigurations, which is of key importance for an efficient operation of SCNs. It is also less sensitive to variation in temperature parameter. To find the optimal parameters for resource allocation, different strategies such as fixed temperature and cooling have been studied. The results suggest that for static networks, SA with an optimal fixed temperature outperforms its cooling counterpart. A similar trend ensues for other variants as well. In SCNs with dynamic topologies, standard SA with cooling can be enabled via distributed temperature control protocols. However, for dynamic networks with partially colored conflict graphs, raising the temperature of the whole network by standard SA type mechanisms is counter-productive, and leads to large numbers of unresolved conflicts. An effective way to mitigate such conflicts is to use the focused search variants, and the temperature control protocols which limit the uphill moves to a sub-graph comprising of only those cells that detect a change in network topology.

Acknowledgment

This work was supported in part by the Academy of Finland, grant 284725, and the European Commission in the framework of the FP7 project ITC-317669 METIS.

7. References

[1] C. Prehofer, C. Bettstetter, Self-Organization in Communication Networks: Principles and Design Paradigms, IEEE Comm. Mag. 43 (7) (2005) 78–85.

[2] 3GPP, Self-configuring and self-optimizing network use cases and solutions, Tech. Rep. TR 36.902, available: http://www.3gpp.org (2008).

[3] T. Chen, H. Zhang, X. Chen, O. Tirkkonen, SoftMobile: Control Evolution for Future Heterogeneous Mobile Networks, IEEE Comm. Mag. 21 (6) (2014) 70–78.

[4] L. Garcia, K. Pedersen, P. Mogensen, Autonomous Component Carrier Selection: Interference Management in Local Area Environments for LTE-Advanced, IEEE Comm. Mag. 47 (9) (2009) 110–116.

[5] F. Ahmed, O. Tirkkonen, M. Peltomäki, J.-M. Koljonen, C.-H. Yu, M. Alava, Distributed Graph Coloring for Self-Organization in LTE Networks, J. Elec. Computer Engineering (2010) 1–10.

[6] F. Ahmed, O. Tirkkonen, Self Organized Physical Cell ID Assignment in Multi-Operator Heterogeneous Networks, in: Proc. IEEE Veh. Tech. Conf. Spring, 2015, pp. 1–5.

[7] A. Eisenblätter, M. Grötschel, A. M. Koster, Frequency Planning and Ramifications of Coloring, Discussiones Mathematicae Graph Theory 22 (1) (2002) 51–88.

[8] F. Ahmed, O. Tirkkonen, Local optimum based power allocation approach for spectrum sharing in unlicensed bands, in: T. Spyropoulos, K. Hummel (Eds.), Self-Organizing Systems, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2009, pp. 238–243.

[9] F. Ahmed, O. Tirkkonen, Topological aspects of greedy self-organization, in: Proc. IEEE Int. Conf. Self-Adaptive and Self-Organizing Systems, 2014, pp. 31–39.

[10] S. Seitz, M. Alava, P. Orponen, Threshold behaviour of WalkSAT and focused metropolis search on random 3-satisfiability, in: Proc. Int. Conf. on Th. and Applications of Satisfiability Testing, Springer-Verlag, Berlin, Heidelberg, 2005, pp. 475–481.

[11] S. Kirkpatrick, C. D. Gelatt, Jr., M. P. Vecchi, Optimization by Simulated Annealing, Science 220 (4598) (1983) 671–680.

[12] F. Novillo, R. Ferrus, Distributed Channel Assignment Algorithm based on Simulated Annealing for Uncoordinated OSA-Enabled WLANs, in: Proc. IEEE Cognitive Radio Oriented Wireless Networks and Comm., 2011, pp. 81–85.

[13] J. Chen, S. Olafsson, X. Gu, Y. Yang, A Fast Channel Allocation Scheme Using Simulated Annealing in Scalable WLANs, in: Proc. IEEE Broadband Comm., Networks, and Systems, 2008, pp. 205–211.

[14] D. Leith, P. Clifford, A Self-Managed Distributed Channel Selection Algorithm for WLANs, in: Proc. Int. Symp. Model. and Opt. Mobile, Ad Hoc and Wireless Networks, 2006, pp. 1–9.

[15] Ö. Bulakci, A. Awada, A. B. Saleh, S. Redana, J. Hämäläinen, Automated uplink power control optimization in LTE-Advanced relay networks, EURASIP J. Wireless Comm. Networking 2013 (1) (2013) 1–19.

[16] M. Garcia-Lozano, S. Ruiz, J. Olmos, CPICH power optimisation by means of simulated annealing in an UTRA-FDD environment, IEEE Elect. Letters 39 (23) (2003) 1676–7–.

[17] Y. Li, Z. Feng, D. Xu, Q. Zhang, H. Tian, Automated Optimal Configuring of Femtocell Base Stations' Parameters in Enterprise Femtocell Network, in: Proc. IEEE Global Telecommun. Conf., 2011, pp. 1–5.

[18] M. Aydin, R. Kwan, J. Wu, J. Zhang, Multiuser Scheduling on the LTE Downlink with Simulated Annealing, in: Proc. IEEE Veh. Tech. Conf. Spring, 2011, pp. 1–5.

[19] A. Awada, B. Wegmann, I. Viering, A. Klein, A SON-Based Algorithm for the Optimization of Inter-RAT Handover Parameters, IEEE Trans. Veh. Tech. 62 (5) (2013) 1906–1923.

[20] A. Temesvary, Self-Configuration of Antenna Tilt and Power for Plug & Play Deployed Cellular Networks, in: Proc. IEEE Wireless Commun. Networking Conf., 2009, pp. 1–6.

[21] S. Seitz, M. Alava, P. Orponen, Focused local search for random 3-satisfiability, J. Stat. Mech. (2005) P06006.

[22] M. Fielding, Simulated Annealing With An Optimal Fixed Temperature, SIAM J. Optimization 11 (2) (2000) 289–307.

[23] M. H. Alrefaei, S. Andradttir, A Simulated Annealing Algorithm with Constant Temperature for Discrete Stochastic Optimization, J. Management Science 45 (5) (1999) pp. 748–764.

[24] D. Marx, D. A. Marx, Graph Coloring Problems and Their Applications in Scheduling, in: Proc. John von Neumann PhD Students Conference, 2004, pp. 1–2.

[25] M. Garey, D. Johnson, H. So, An application of graph coloring to printed circuit testing, IEEE Trans. Circuits Systems 23 (10) (1976) 591 – 599.

[26] M. Chams, A. Hertz, D. de Werra, Some experiments with simulated annealing for coloring graphs, European J. Oper. Res. 32 (2) (1987) 260–266.

[27] D. S. Johnson, C. R. Aragon, L. A. McGeoch, C. Schevon, Optimization by Simulated Annealing: An Experimental Evaluation; Part II, Graph Coloring and Number Partitioning, J. Oper. Res. 39 (3) (1991) pp. 378–406.

[28] A. Hertz, D. Werra, Using Tabu Search Techniques for Graph Coloring, J. Computing 39 (1987) 345–351.

[29] D. Costa, A. Hertz, C. Dubuis, Embedding a Sequential Procedure Within an Evolutionary Algorithm for Coloring Problems in Graphs, J. Heuristics 1 (1995) 105–128.

[30] C. Fleurent, J. Ferland, Genetic and hybrid algorithms for graph coloring, Annals of Oper. Res. 63 (1996) 437–461.

[31] P. Galinier, A. Hertz, A survey of local search methods for graph coloring, J. Comp. Oper. Res. 33 (2006) 2547–2562.

[32] L. Barenboim, M. Elkin, Distributed Graph Coloring: Fundamentals and Recent Developments, Synthesis Lect. Distributed Computing Th. 4 (1) (2013) 1–171.

[33] M. Yokoo, E. Durfee, T. Ishida, K. Kuwabara, The Distributed Constraint Satisfaction Problem: Formalization and Algorithms, IEEE Trans. on Knowledge and Data Eng. 10 (5) (1998) 673 –685.

[34] K. Hirayama, M. Yokoo, The distributed breakout algorithms, J. Artif. Intell. 161 (2005) 89–115.

[35] M. Peltomaki, J. Koljonen, O. Tirkkonen, M. Alava, Algorithms for Self-Organized Resource Allocation in Wireless Networks, IEEE Trans. Veh. Tech. 61 (1) (2012) 346–359.

[36] J.-M. Koljonen, M. Alava, M. Peltomaki, O. Tirkkonen, Distributed Generalized Graph Coloring, in: Proc. IEEE Self-Adaptive Self-Organizing Systems, 2010, pp. 174–183.

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

[38] D. McAllester, B. Selman, H. Kautz, Evidence for Invariants in Local Search, in: Proc. National Conf. Artif. Intell., AAAI Press, 1997, pp. 321–326.

[39] B. Hajek, Cooling Schedules for Optimal Annealing, Math. Oper. Res. 13 (2) (1988) 311–329.

[40] G. Mirkin, K. Vasudevan, F. A. Cook, W. G. Laidlaw, W. G. Wilson, A Comparison of Several Cooling Schedules for Simulated Annealing Implemented on a Residual Statics Problem, Geophysical Research Letters 20 (1) (1993) 77–80.

[41] Y. Nourani, B. Andresen, A comparison of simulated annealing cooling strategies, J. of Physics A: Mathematical and General 31 (41) (1998) 8373.