Stochastic Local Search Based Channel Assignment in Wireless Mesh Networks

Stochastic Local Search based Channel Assignment in Wireless Mesh Networks

M.A.Hakim Newton[1], Duc Nghia Pham[1], Wee Lum Tan[2,3], Marius Portmann[2,3], and Abdul Sattar[1]

[1] Institute for Integrated and Intelligent Systems, Griffith University
[2] Queensland Research Lab, National ICT Australia (NICTA)
[3] School of ITEE, The University of Queensland

Abstract. In this paper, we consider the problem of channel assignment in multi-radio, multi-channel wireless mesh networks. We assume a binary interference model and represent the set of interfering links in a network topology as a conflict graph. We then develop a new centralised stochastic local search algorithm to find a channel assignment that minimises the network interference. Our algorithm assigns channels to communication links rather than radio interfaces. By doing so, our algorithm not only does preserve the network topology, but is also independent of the network routing layer. We compare the performance of our algorithm with that of a well-known tabu-based approach (by Subramanian et al.) on randomly generated sparse and dense network topologies. Using graph-theoretic evaluation and ns2 simulations (a widely used discrete event network simulator), we show that our algorithm consistently outperforms the tabu-based approach in terms of both the network interference and the throughput obtained under various offered loads. In particular, for a practical setting of 3 radio interfaces per mesh node in a dense network topology with 12 channels available, our approach achieves 70% lower network interference and thus 15 times higher average throughput than those achieved by the tabu-based approach.

1 Introduction

Wireless interference is one of the major factors that limits the performance of IEEE 802.11-based wireless mesh networks. There have been various approaches proposed to improve network performance by mitigating or taking into account the effects of interference in wireless networks. These approaches include optimising the transmission power used by the nodes in a wireless network [6], utilising network routing protocols that are interference-aware [8], and scheduling conflict-free transmissions that consider the physical interference in the wireless network [2]. In wireless mesh networks, a popular approach to minimise the effects of interference is to equip wireless mesh nodes with multiple radio interfaces, and assign non-overlapping channels to these interfaces. The key challenge in this case is to design a channel assignment algorithm that assigns the available channels to the radio interfaces in such a way that minimises the network interference

and thus maximises the network throughput, while at the same time preserving the network topology.

In this paper, we present a new centralised stochastic local search algorithm for channel assignment in multi-radio, multi-channel wireless mesh networks. The objective is to minimise the network interference while satisfying the interface constraint. We assume that the interference model is binary. Consequently, two communication links are said to interfere with each other if they are assigned the same channel and are within interference range of each other. The *network interference* is the number of interfering pairs of links in the channel assignment. There is also an *interface constraint* for each mesh node to ensure that the number of channels being used at that node does not exceed the number of radios available at the node. Our algorithm assigns channels to communication links rather than radio interfaces. By doing so, our algorithm not only does preserve the network topology, but is also independent of the network routing layer.

We developed our algorithm on top of Kangaroo, a constraint-based local search system [12]. We designed a new constraint to model the interface constraint at each mesh node. We then represent the problem as a constrained optimisation problem and use stochastic local search to find a solution (i.e. channel assignment). Our search algorithm starts from a randomly generated initial solution; which may be infeasible. It then iteratively improves the feasibility and optimality metrics of the solution. The search attempts to improve the feasibility metric when the current solution is far from being feasible; other times, it tries to improve the optimality metric. When there is no improvement within a number of iterations, it also restarts by randomly assigning values to a number of variables. For the selection of links that require changing its channel assignment, we use Novelty [11], a very well-known stochastic local search algorithm. For the selection of a channel to be assigned to a link, we pick the best possible channel.

We compare the performance of our algorithm with that of a well-known tabu-based approach by Subramanian et al. [18] on randomly generated sparse and dense network topologies. Using graph-theoretic evaluation and ns2 simulations (a widely used discrete event network simulator [1]), we empirically show that our algorithm consistently outperforms the tabu-based approach in terms of both the network interference and the throughput obtained under various offered loads. In particular, for a practical setting of 3 radio interfaces per mesh node in a dense network topology with 12 channels available, our approach achieves 70% lower network interference and thus 15 times higher average throughput than those achieved by the tabu-based approach.

The rest of the paper is organised as follows: Section II reviews related work; Section III describes the model and problem formulation; Section IV provides a brief review of the Tabu-based algorithm by Subramanian et al. [18]; Section V describes in detail our stochastic local search based approach; Section VI presents our experimental evaluations. Finally, Section VII summarises our conclusions and outlines the future work.

2 Related Work

The problem of channel assignment in wireless mesh networks has been a major research topic in the last few years. There has been various channel assignment algorithms that have been proposed in the literature. In this section, we briefly review a few relevant approaches. Interested readers can refer to a few good survey papers on the various channel assignment algorithms [4] [17].

Existing works on channel assignment algorithms can be divided into the *distributed* and *centralised* approaches. In distributed channel assignment approaches [7] [16] [15], individual network nodes compute its channel assignment based on locally gathered information about its network neighbourhood. Distributed channel assignment approaches are more suitable to be used once a network has been set up and is operationally running. This is because it is more adaptive to dynamic changes in local network topology (due to node failures or external interference) and any changes in the channel assignment can be confined to the local neighbourhood. In this paper, we are interested in the optimal channel assignment for mesh nodes in an initial network-wide deployment, which is better handled by a centralised channel assignment approach.

In centralised channel assignment approaches [18] [10] [14], a central entity computes the optimal channel assignment based on global information about the network topology, such as the interference relationship between the nodes or communication links in the network. Typically, the interference relationship is represented as a conflict graph. The central entity then disseminates the channel assignment information throughout the network to every node.

Subramanian et al. proposed a tabu-based approach in [18] in which the tabu search based technique [5] is used to find a channel assignment that minimises the network interference. We will describe the tabu-based approach in more detail in Section IV, and compare the performance of our proposed algorithm with this approach in Section VI. In [10], a greedy heuristic channel assignment algorithm called CLICA is proposed to find a connected and low interference network topology. Subramanian et al. compared the performance of their tabu-based approach with CLICA in [18] and showed that their approach performs better. In the BFS channel assignment algorithm in [14], each mesh node has one radio interface configured to a default common channel in order to maintain network-wide connectivity. With typical mesh nodes having at most two or three radio interfaces each, this can lead to inefficient utilisation of the available channels in the network and poor network performance. In contrast, our algorithm in this paper preserves network connectivity by assigning channels to communication links, instead of using a dedicated radio interface configured to a common channel.

3 Model and Problem Formulation

A typical architecture of a wireless mesh network has two tiers: *backbone tier* and *access tier*. The backbone tier consists of stationary *mesh nodes* (or *nodes*)

forming a wireless multihop backbone infrastructure, with one or more nodes also functioning as gateways to the Internet. The access tier sees the end-user client devices connecting to the mesh nodes in order to communicate with other client devices or to access the Internet. The 5GHz and 2.4GHz frequency bands are typically used for the communications in the backbone and access tiers respectively.

The mesh nodes usually have multiple radio interfaces, each of which might be configured with a channel $k \in \mathcal{K}$, where $\mathcal{K}$ is the set of available channels in the backbone tier. We assume all nodes to have the same number of r radio interfaces, each having omni-directional antennas with the same transmission range R_{tx}. Let D_{uv} denote the physical distance between two nodes u and v. There exists a *communication link* (or *link*) $l \equiv l_{uv}$ between nodes u and v, if $D_{uv} < R_{\text{tx}}$ and both nodes have a radio interface configured to a common channel. A given set of nodes V and the links E can be modelled as an undirected graph $G = (V, E)$. Let E_v denote the links incident on a node v.

In this paper, we are interested in the optimal assignment of channels to links in a multihop backbone infrastructure in order to minimise the interference between the links. Due to the inverse relationship between interference and network throughput, by minimising the interference in the network, we essentially maximise the network throughput. We model the interference between co-channel communication links with the range-based interference model. In this interference model, every node has an associated interference range, R_{int}, that is typically larger than the transmission range. A *unicast* transmission on a link l from node u to node v can be interfered with by another simultaneous transmission (on the same channel) from any node within the interference range of both nodes u and v.

We assume that the interference model is binary in the sense that the unicast transmission is successful if there is no interference present, and unsuccessful otherwise. We also assume that transmissions on different channels do not interfere. Note that our channel assignment algorithm does not depend on the choice of the interference model used in this paper, *i.e.* the range-based interference model. Our channel assignment algorithm will work with other interference models such as the hop-based and the protocol interference models. These interference models are called *pairwise* interference models after the fact that interference is defined on pairs of communication links. Interested readers can refer to [9] for a description of these and other interference models.

Given a pairwise interference model, we use a conflict graph G_c to model the set of interfering communication links in the wireless mesh network represented by a graph G. Each link l in G essentially becomes a vertex in the conflict graph G_c. An edge in G_c exists between a pair of vertices l and l', if the links l_{uv} and $l'_{u'v'}$ in the network are on the same channel and interfere with each other. With the range-based interference model that we adopt in this paper, an edge exists if any of the following is true: $D_{uu'} \leq R_{\text{int}}$, or $D_{uv'} \leq R_{\text{int}}$, or $D_{vu'} \leq R_{\text{int}}$, or $D_{vv'} \leq R_{\text{int}}$.

The objective of our channel assignment problem is to find a mapping ϕ that assigns unique channels to the links in such a way that minimises the network interference. Given a channel assignment ϕ, let $\phi(l)$ denote the channel assigned to a link l and $\sigma_\phi(v)$ the number of unique channels assigned to the links incident on a node v. The *network interference* $\eta(\phi)$ of a channel assignment ϕ is the number of edges in the conflict graph. A feasible channel assignment must satisfy all the *interface constraints* to ensure that for each node v, the number of unique channels assigned to all communication links incident on v is at most the number of radio interfaces available at v, *i.e.* $\sigma_\phi(v) \leq r$.

Note that by assigning channels to communication links rather than to radio interfaces, our channel assignment algorithm maintains the same network topology as in the case when a single channel is used for all communication links in the network. By doing so, our channel assignment algorithm is independent of the network routing layer.

4 Tabu-Based Algorithm

The tabu-based algorithm by Subramanian et al. [18] comprises two phases, which can be viewed as optimisation and satisfaction phases. First, in the *optimisation phase*, the network interference is iteratively minimised without considering the interface constraints. Therefore, the best solution found in the first phase normally violates the interface constraints. Next, in the *satisfaction phase*, these violated interface constraints are heuristically fixed to obtain a feasible solution. Note that while fixing those violated constraints, the network interference may increase. However, the tabu-based algorithm returns its one and only solution at the end of the second phase without attempting to improve the network interference further.

4.1 Optimisation Phase

In this phase, the tabu-based algorithm starts with a random initial solution ϕ_0 wherein each link l is assigned to a random channel $k \in \mathcal{K}$ *i.e.* $\phi_0(l) = k$. Given a solution ϕ_i, it then obtains the next solution ϕ_{i+1} by selecting the best candidate (having the lowest network interference) from a number of randomly generated neighbouring solutions. A neighbouring solution ϕ_i' of a solution ϕ_i is generated by randomly selecting a link l and then assigning a random channel $k \in \mathcal{K}$ to the link. The neighbour generation process ensures that $k \neq \phi_i(l)$ and the pair (l, k) does not appear in a maintained tabu list τ, which is a first-in-first-out queue of a given length. When ϕ_{i+1} is obtained from ϕ_i, the modified link l and the new channel k assigned to the link are pushed into the queue; before that, one queue element is popped out, if the queue is full. The first phase of the algorithm terminates if there is no improvement in the network interference after $|E|$ iterations, where E is the set of links in the network.

4.2 Satisfaction Phase

In this phase, the tabu-based algorithm mainly attempts to satisfy the violated interface constraints in the solution ϕ returned by the first phase. For this, it picks a node v that has the most violations in its interface constraint. The number of violations of the interface constraint at a given node v is $\mathsf{max}(\sigma_\phi(v) - r, 0)$. The algorithm then performs a merge operation that chooses two channels k and k', and makes a replacement $\phi(l) = k'$ for each incident link l to v such that $\phi(l) = k$. The replacement process has a recursive cascading effect on each node v' that l is also incident on. While satisfying the interface constraints, the merge operation may increase the network interference $\eta(\phi)$. Therefore, the choice of k and k' is greedily made so that the increase in $\eta(\phi)$ is minimised.

5 Stochastic Local Search Based Algorithm

We developed our stochastic local search based (or SLS-based) algorithm on top of Kangaroo, a constraint-based local search system [12]. We define the variables and functions such as constraints and objectives in the Kangaroo system. Kangaroo then efficiently propagates changes from variables to the dependant functions. In each iteration when the variables are assigned with new values, Kangaroo performs *execution* by updating the functions' values. It also helps to efficiently explore potential neighbouring solutions by performing *simulation*: computing the feasibility and optimality metrics temporarily due to the potential changes in the variables.

5.1 Constraint and Objective Functions

We designed a new constraint function AtMostCount in Kangaroo to model the interface constraints. The AtMostCount constraint maintains the degree of constraint violation and penalises the links for causing the violation. We also use NotEqual constraints to represent 'no conflict' between two given links, and Sum function to accumulate the constraints to form the top-level objective functions.

Each function $f(p_1, \cdots, p_n)$ has the parameters p_js that are either variables or other functions. A function f depends on a variable x, denoted by $f \rightarrow x$, if x is itself a parameter of f or f has a parameter $p \rightarrow x$. Each function f has a non-negative metric f^{m} denoting its evaluation. For each $x \leftarrow f$, it also has a non-negative hint $f^{\mathrm{h}}(x)$ denoting the preference of changing x's value to improve f^{m}. A constraint f is satisfied when $f^{\mathrm{m}} = 0$ and in that case $f^{\mathrm{h}}(x) = 0$ for any x. This means a constraint's metric improves when it is minimised.

AtMostCount The number of unique values occurred in the given n variables $x_1, \cdots, x_n$ must not exceed a specified limit m. This constraint, denoted by $C(x_1, \cdots, x_n)$, has its $C^{\mathrm{m}} = \mathsf{max}(c-m, 0)$ and for each j, $C^{\mathrm{h}}(x_j) = n-c_j$, where c is the number of unique values used in the variables while c_j is the number of times x_j's value has occurred. Notice that C^{m} is the number of additional values

used beyond the limit. Also, notice that $C^{\mathrm{h}}(x_j)$ captures the heuristic stated as 'the fewer the value of a variable occurs, the more the preference of the variable's value to be modified'.

NotEqual The values of two given variables must not be equal. This constraint, denoted by $\bar{Q}(x, x')$, has its $\bar{Q}^{\mathrm{m}} = 1$ if x equals x', else 0. Besides, $\bar{Q}^{\mathrm{h}}(x) = \bar{Q}^{\mathrm{h}}(x') = \bar{Q}^{\mathrm{m}}$. When two variables have the same value changing either one's value could make them unequal and thus improve the metric.

Sum Sum up a given number of functions. This function, denoted by $S(f_1, \cdots, f_n)$, has its $S^{\mathrm{m}} = \sum f_j^{\mathrm{m}}$ and for each $x \leftarrow S$, $S^{\mathrm{h}}(x) = \sum_{x \leftarrow f_j} f_j^{\mathrm{h}}(x)$, where $1 \leq j \leq n$.

5.2 Constrained Optimisation Model

The constrained optimisation model used by our algorithm is defined in Procedure 1 ConstructModel. We first create a variable x_l for each link l in the network (Lines 1–2). The domain of each variable is $\mathcal{K}$, the set of available channels. For each node v, we then create an AtMostCount constraint C_v with the links incident on v (Lines 3–5). The limit given to each C_v is r which is the number of radio interfaces available at a node. The Sum function S_{sat} accumulates all the AtMostCount constraints (Line 6). The accumulated metric of S_{sat} must be 0 in order to get a feasible channel assignment for the network.

Procedure 1: ConstructModel

```
1  foreach link l ∈ E do
2      Create a variable x_l with domain K;
3  foreach node v ∈ V do
4      Let X be the set of variables for the links E_v;
5      Create a constraint C_v with X and the limit r;
6  Create a function S_sat with all C_vs created above;
7  foreach potential edge e in G_c do
8      Let l and l' be the vertices which e incidents on;
9      Create a constraint Q̄_ll' with variables x_l and x_l';
10 Create a function S_opt with all Q̄s created above;
11 Create a function S_comb with S_sat and S_opt;
```

For each *potential* edge in the conflict graph G_c, we create a NotEqual constraint $\bar{Q}_{ll'}$ (Lines 7–9) with the variables x_l and $x_{l'}$ respectively associated with the two links l and l'; note that these two links cause interference if they are assigned with the same channel. The Sum function S_{opt} accumulates all the NotEqual constraints (Line 10) and thus represents the network interference. The metric of this Sum function needs to be as small as possible. Finally, we have another Sum function S_{comb} that combines S_{sat} and S_{opt} (Line 11) to give an overall evaluation of the candidate channel assignment.

Procedure 2: PerformSearch

```
1  BestSoln = ∅;
2  MinConflict = ∞;
3  Proximity = 1;
4  IterSinceRestart = 0;
5  RestartPeriod = |E| × 10;
6  initialiseRandomly (CurrSoln);
7  while S^m_comb > 0 ∧ ¬TimeOut do
8      if S^m_sat = 0 ∧ S^m_opt < MinConflict then
9          MinConflict = S^m_opt;
10         BestSoln = CurrSoln;
11         IterSinceRestart = 0;
12     if ++IterSinceRestart > RestartPeriod then
13         IterSinceRestart = 0;
14         X = Select |E|/8, 2|E|/8 or 3|E|/8 variables randomly;
15         Assign random values to the variables X;
16     else if S^m_sat < Proximity then
17         Select C randomly from all C_v s;
18         (x, k) = selectVarValue(C, S_opt, S_comb);
19         Assign the value k to the variable x;
20     else // S^m_sat ≥ Proximity
21         Select C randomly from C_v s such that C^m_v > 0;
22         (x, k) = selectVarValue(C, S_sat, S_sat);
23         Assign the value k to the variable x;
24 return BestSoln;
```

5.3 Local Search Method

Our search algorithm, shown in Procedure 2 PerformSearch, starts from a randomly generated initial solution; which may be infeasible (Line 6). It then iteratively improves the feasibility and optimality metrics of the solution. In each iteration, our algorithm chooses one of the three options: *i)* satisfy: minimise the violation of interface constraints (Lines 20–23) when the current assignment is far from being feasible; *ii)* optimise: minimise the network interference (Lines 16–19) if the current assignment is already feasible or very close to being feasible; or *iii)* restart: randomly change a large portion of the current assignment to escape stagnation (Lines 12–15).

The first option (satisfy) randomly selects a violated AtMostCount C with a view to improving the feasibility metric $S^{\mathrm{m}}_{\mathrm{sat}}$ by changing the value of a variable $x \leftarrow C$. Function 3 selectVarValue selects such a variable by using the hints of S_{sat}. The first option then selects a value k for x such that the metric of S_{sat} is minimised. The second option (optimise) randomly selects an AtMostCount C. It then uses Function 3 to select a variable $x \leftarrow C$ by using the hints of S_{opt}. Moreover, it selects a value k for x such that the metric of S_{comb} is minimised. The selections made by this option are explained later. The third option (restart)

randomly assigns values to a number of variables, if there is no improvement after a given number of iterations.

Function 3 selectVarValue selects a variable from a subset of variables, and lastly its value. The subset of variables to consider comes from the variables that VarFunc parameter depends on. With a small probability, the function selects a random variable in the subset (Line 4). Otherwise for most of the time, the function selects a variable in the subset that has the maximum hint in VarSum, breaking ties on being assigned earlier (Lines 5–11). However, when that variable is the most recently assigned one in this subset of variables, the function with a given small noise probability may also select the variable that has the second maximum hint (Novelty heuristic in Line 10).

Refer to the selections made in the second option (optimise) in Procedure PerformSearch. We could use a violated NotEqual constraint $\bar{Q}$ to obtain a subset of variables to be used in Function selectVarValue. However, a $\bar{Q}$ depends only on two variables, and thus restricts the choice particularly in a stagnant situation. Notice that most of the time the variable selection is greedy and the value selection is also greedy with no tabu; which quickly leads to a local optimal and thus to a stagnant situation which must be escaped for any further improvement. The Novelty algorithm and a random selection are used to escape this stagnation. The use of a random C gives more choices to select a variable from. While selecting the variable, we use the hints of S_{opt} as we want to improve its metric. However, we use the metric of S_{comb} to select a value; this is to allow moving to an assignment that might converge in the optimality metric but could even diverge in the feasibility metric.

Function 3: selectVarValue(VarFunc, VarSum, ValSum)

```
1  Noise = 0.01;
2  X = {x : x ← VarFunc};
3  Select a variable x ∈ X in the following way:
4    if bernoulii (Noise) then randomly;
5    else // use Novelty on VarSum
6         Assume x1 ∈ X and x2 ∈ X have the first
7              and second maximum hint in VarSum
8              breaking tie on being assigned earlier;
9         if x1 is the last assigned variable then
10             x= if bernoulii (Noise) then x2 else x1
11        else x = x1
12 By simulation, select a value k for x
13   such that the metric of ValSum is minimised;
14 return (x, k);
```

6 Performance Evaluations

For performance comparison, we implemented the tabu-based algorithm as specified in [18]. We compared the performance of the tabu-based algorithm and our SLS-based algorithm using both graph-theoretic evaluations and ns2 simulations [1]. In our evaluations, we used two different types of random network topologies: dense and sparse. We generated these networks by randomly placing 50 nodes respectively in 500×500 and 800×800 square meter areas. Using the default ns2 simulation settings for IEEE 802.11a, the transmission range R_{tx} is set to 163 meters, while the interference range R_{int} is set to 410 meters. With these settings, the average node degree is around 11 in the dense network while in the sparse network, the average node degree is around 5.

6.1 Graph-Theoretic Evaluations

For both of the channel assignment algorithms, we used a computer equipped with Intel(R) Xeon(R) 64bit quad-core CPU X3470 @2.93GHz with 8MB L2 Cache and 8GB RAM running Ubuntu Linux version 12.04. We ran both the algorithms 25 times on each benchmark topology with a realistic 30 seconds cutoff time. For each run, we collected the best solution found within the cutoff time. We then used the median quality solution out of those 25 best solutions to run the ns2 simulations and to present our results and provide analysis based on them.

Note that the tabu-based algorithm is a *one-off* algorithm meaning it produces only one solution as its output and then terminates. This is obvious from the two phases of the algorithm. The first phase iteratively minimises only the network interference while ignoring the interface constraints. Therefore, a feasible solution can only be found at the end of the second phase which fixes those constraint violations. Nevertheless, in most cases, the tabu-based algorithm was able to return a solution within the stipulated cutoff time.

In contrast, our SLS-based algorithm is an *any-time* algorithm [20], meaning that one can terminate the algorithm at any time (after the initial time to find the first solution) and still has a valid solution. In other words, our algorithm is capable to quickly find a feasible solution. Once such a solution is found, our algorithm still keeps running in the quest of further improving solutions (in terms of the network interference) as long as the cutoff time permits (Procedure 2 Lines 7–10). Trading off with the time available, one can at any time take the best solution found so far.

We plot the typical convergence pattern of our algorithm in Figure 1 on a dense network with 3 radio interfaces at each node and 12 available channels. Notice that by the time the tabu-based algorithm found its solution, our algorithm has already found a number of solutions that are much better. Our algorithm also continues to improve the quality of the solution. If one were to terminate our algorithm early, still a sufficiently good solution would be returned.

For further comparison, we graphically show in Figure 2 the fractional network interferences of the solutions produced by the SLS-based and Tabu-based

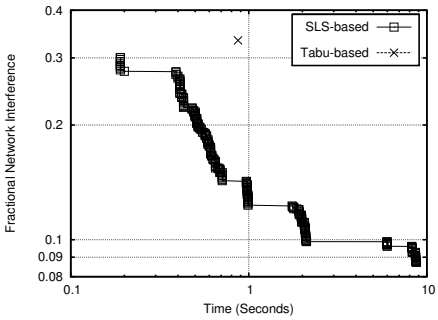
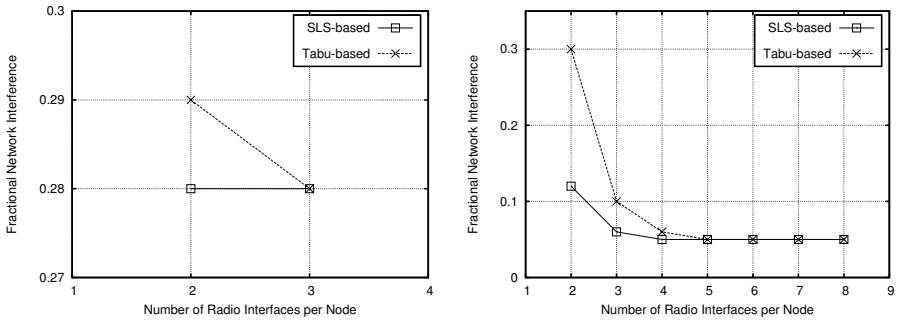


Fig. 1: Convergence of fractional network interference w.r.t. time for SLS-based and Tabu-based algorithms in the dense network with 12 channels and 3 radio interfaces per node.

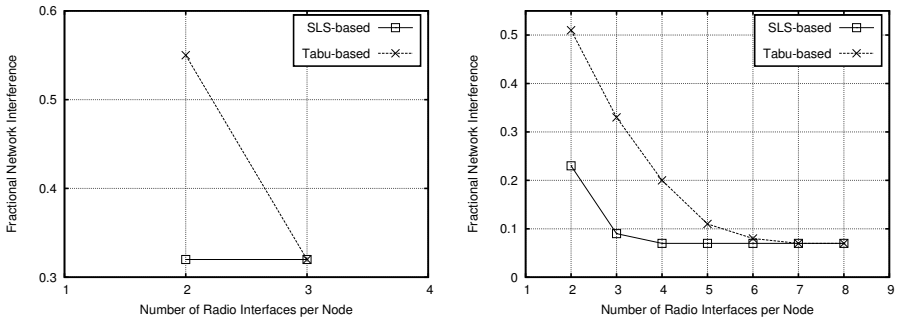


(a) 3 channels (b) 12 channels

Fig. 2: Fractional network interference of the solutions produced by the SLS-based and Tabu-based algorithms in the sparse network with (a) 3 channels and (b) 12 channels

algorithms for the sparse network. These results were obtained by using 3 and 12 channels with the number of radio interfaces varied from 2 to 8. We show the same for the dense network in Figure 3. The *fractional network interference* is defined as the ratio of the number of edges in the conflict graph produced by a given channel assignment and the total number of conflicts in the single-channel network. From the results in the figures, it is clear that our algorithm obtained significantly better fractional network interference than the tabu-based algorithm.

We sought a reasonable explanation for these performances. Note that our algorithm switches between the satisfaction and optimisation phases in an interleaving fashion. Moreover, both phases respect the interface constraints and the network interference in a separate or combined way. On the other hand, the tabu-based algorithm has two separate phases where the satisfaction phase



(a) 3 channels (b) 12 channels

Fig. 3: Fractional network interference of the solutions produced by the SLS-based and Tabu-based algorithms in the dense network with (a) 3 channels and (b) 12 channels

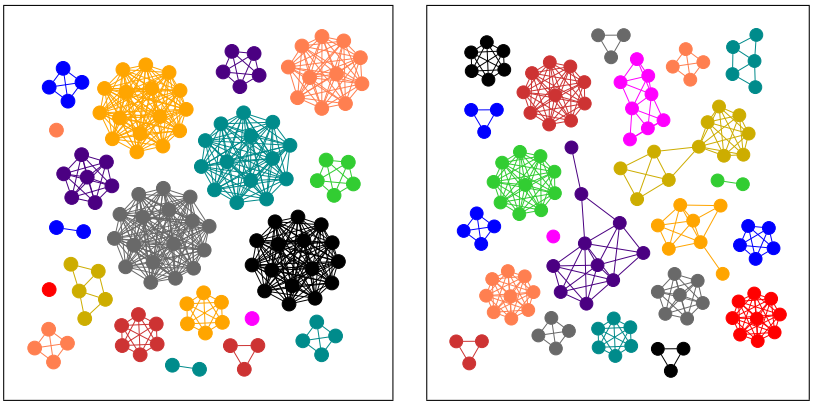
follows the optimisation phase. While the latter phase only minimises the network interference, the former phase just satisfies the interface constraints; no interaction between these two criteria is considered.

Our further observations suggest the solution quality of the tabu-based algorithm greatly reduces by the fact that fewer than the total number of available channels are used by both of its phases. In particular, the satisfaction phase while addressing the violations of the interface constraints replaces one channel with another; which greatly reduces the number of channels being used in the end. As a result, the conflict graphs contain larger clusters and a higher number of edges for the solutions produced by the tabu-based algorithm, compared to those produced by the SLS-based algorithm. This can be observed in Figure 4 where we show the conflict graphs produced by the SLS-based and Tabu-based algorithms in the sparse network with 12 channels and 3 radio interfaces per node.

In Figures 2 and 3, notice that the fractional network interference produced by our SLS-based algorithm reaches the minimum after 2 radios for the 3-channels case and 4 radios for the 12-channels case. Further increase in the number of radio interfaces does not make any significant difference in the network interference. This means the number of radio interfaces at each node could effectively be reduced to these levels without increasing the network interference; which would result in lower hardware cost for the mesh nodes.

6.2 ns2 Simulations

In this evaluation, we investigate the network throughput performance of both our SLS-based algorithm and the tabu-based algorithm. We use the IEEE 802.11a implementation in ns2 with its default settings, and added support for multiple interfaces and multiple channels [3]. Using the same two network topologies as in



(a) Tabu-based (b) SLS-based

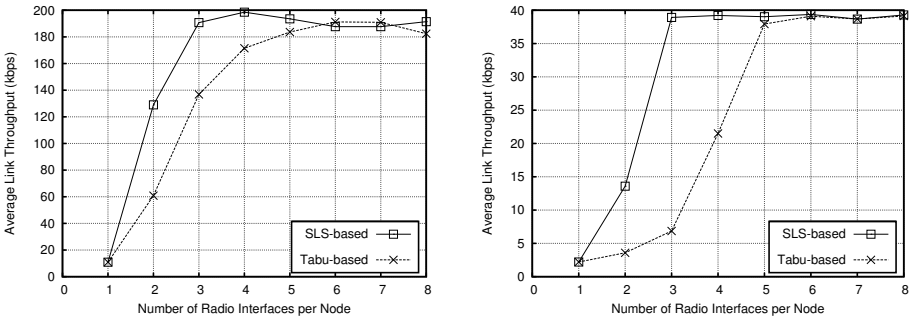
Fig. 4: Conflict graphs for the solutions produced by the SLS-based and Tabu-based algorithms in the sparse network with 12 channels and 3 radio interfaces per node

Section VI.A, we employ CBR traffic on every single-hop communication link in the network. We use this single-hop traffic model in order to evaluate the performance when all communication links in the network carry the same traffic load. We measure the throughput on every link as we slowly increase the offered traffic load on each link until the achievable throughput does not increase anymore. All transmissions are unicast transmissions with a packet size of 1000 bytes, and are using the IEEE 802.11a MAC protocol with the RTS/CTS feature turned off. The transmission bit rate is set to a fixed rate of 6Mbps, and each simulation run is 60 seconds long.

Figure 5 shows the average link throughput achieved in the sparse and dense networks with 12 available channels, for different offered traffic loads as we vary the number of radio interfaces per node from 1 to 8. We observe that the average link throughput achieved by our SLS-based algorithm reaches a maximum after 3 or 4 radios. This is consistent with the graph-theoretic results shown earlier in Figures 2b and 3b. We also see that our SLS-based algorithm consistently outperforms the tabu-based algorithm in terms of the achievable throughput obtained under various offered traffic loads. In particular, we see that for a practical network setting of 2 or 3 radio interfaces per node, our SLS-based algorithm achieves an average throughput that is markedly higher by as much as 2 times in the sparse network and 15 times in the dense network, compared to the tabu-based algorithm.

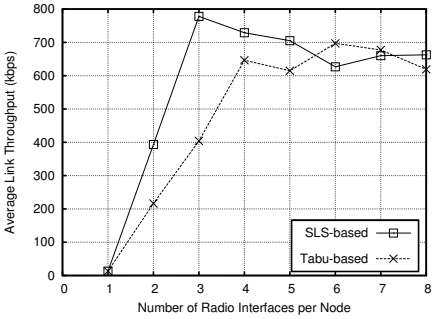
7 Conclusion

In this paper, we have presented a new centralised stochastic local search algorithm to find a channel assignment that minimises the network interference in a

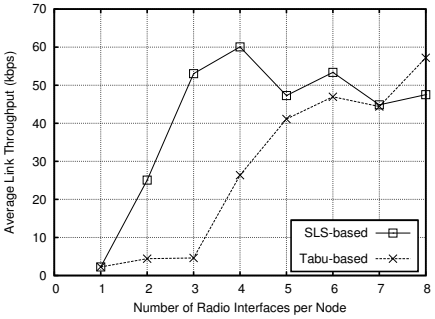


(a) sparse, low (200kbps)

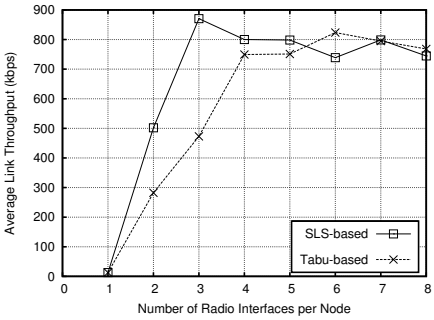
(b) dense, low (40kbps)



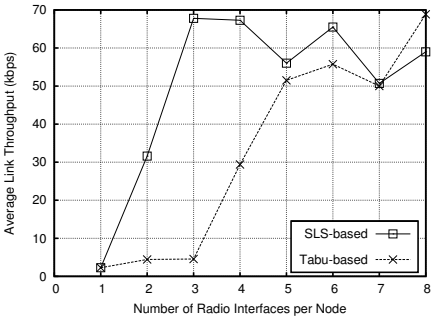
(c) sparse, medium (2000kbps)



(d) dense, medium (400kbps)



(e) sparse, high (4000kbps)



(f) dense, high (800kbps)

Fig. 5: Average link throughput in a sparse/dense network with 12 channels, and low/medium/high offered traffic load.

multi-radio and multi-channel wireless mesh network. Using a binary interference model, we represent the interfering links in the network with a conflict graph. Our SLS-based algorithm preserves the network topology and is independent of the routing layer. We compared the performance of our SLS-based algorithm

with the tabu-based approach [18] on randomly generated sparse and dense network topologies by using graph-theoretic evaluation and ns2 simulations.

Our graph-theoretic results show that our approach significantly outperforms the tabu-based approach in the network interference of the channel assignments produced. Our approach produces solutions with smaller clusters in the conflict graphs compared to those produced by the tabu-based approach. Furthermore, our approach usually finds a number of better solutions even before the tabu-based approach produces its only solution. Our approach still continues to improve the solution quality. Thus ours is an any-time algorithm meaning one could stop it at any time and still get a reasonably good solution.

The ns2 simulation results show that our SLS-based algorithm consistently outperforms the tabu-based approach in terms of the average network throughput obtained under various offered traffic loads. In particular, for a practical setting of 3 radio interfaces per mesh node in a dense network topology with 12 channels available, our approach achieves 70% lower network interference and 15 times higher average throughput than those achieved by the tabu-based approach.

In terms of future work, we plan to explore the use of more advanced stochastic local search algorithms (e.g. gNovelty$^+$ [13]) in our channel assignment approach. We also plan to use other interference models, e.g. measurement-based interference models [19], in our channel assignment algorithm in order to capture more realistic network interference scenarios.

References

1. ns-2, http://www.isi.edu/nsnam/ns/
2. Brar, G., Blough, D.M., Santi, P.: Computationally efficient scheduling with the physical interference model for throughput improvement in wireless mesh networks. In: Proc. ACM MobiCom (2006)
3. Calvo, R., Campo, J.: Adding multiple interface support in ns-2 (2007), http://personales.unican.es/aguerocr/files/ucmultiifacessupport.pdf
4. Crichigno, J., Wu, M., Shu, W.: Protocols and architectures for channel assignment in wireless mesh networks. Ad Hoc Networks 6(7), 1051–1077 (2008)
5. Hertz, A., Werra, D.: Using tabu search techniques for graph coloring. Computing 39(4), 345–351 (1987)
6. Kawadia, V., Kumar, P.R.: Principles and protocols for power control in wireless ad hoc networks. IEEE Journal on Selected Areas in Communications 23(1), 76–88 (2005)
7. Ko, B., Misra, V., Padhye, J., Rubenstein, D.: Distributed channel assignment in multi-radio 802.11 mesh networks. In: Proc. IEEE WCNC (2007)
8. Liu, T., Liao, W.: Interference-aware QoS routing for multi-rate multi-radio multi-channel ieee 802.11 wireless mesh networks. Trans. Wireless. Comm. 8(1), 166–175 (Jan 2009)
9. Maheshwari, R., Jain, S., Das, S.R.: A measurement study of interference modeling and scheduling in low-power wireless networks. In: Proc. ACM SenSys (2008)
10. Marina, M., Das, S., Subramanian, A.: A topology control approach for utilizing multiple channels in multi-radio wireless mesh networks. Computer networks 54(2), 241–256 (2010)

11. McAllester, D.A., Selman, B., Kautz, H.A.: Evidence for invariants in local search. In: AAAI/IAAI. pp. 321–326 (1997)
12. Newton, M.A.H., Pham, D.N., Sattar, A., Maher, M.: Kangaroo: An efficient constraint-based local search system using lazy propagation. In: Proc. CP (2011)
13. Pham, D.N., Thornton, J., Gretton, C., Sattar, A.: Combining adaptive and dynamic local search for satisfiability. JSAT 4(2-4), 149–172 (2008)
14. Ramachandran, K., Belding, E., Almeroth, K., Buddhikot, M.: Interference-aware channel assignment in multi-radio wireless mesh networks. In: Proc. IEEE INFOCOM (2006)
15. Raniwala, A., Chiueh, T.: Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network. In: Proc. IEEE INFOCOM (2005)
16. Shin, M., Lee, S., Kim, Y.: Distributed channel assignment for multi-radio wireless networks. In: Proc. IEEE MASS (2006)
17. Si, W., Selvakennedy, S., Zomaya, A.: An overview of channel assignment methods for multi-radio multi-channel wireless mesh networks. Journal of Parallel and Distributed Computing 70(5), 505–524 (2010)
18. Subramanian, A., Gupta, H., Das, S.R., Cao, J.: Minimum interference channel assignment in multiradio wireless mesh networks. IEEE Transactions on Mobile Computing 7(11) (2008)
19. Tan, W., Hu, P., Portmann, M.: Experimental evaluation of measurement-based SINR interference models. In: Proc. IEEE WoWMoM (2012)
20. Zilberstein, S.: Using anytime algorithms in intelligent systems. AI Magazine 17(3), 73–83 (1996), http://rbr.cs.umass.edu/shlomo/papers/Zaimag96.html