Genetic Algorithms for Wireless Mesh Network Planning

Rastin Pries, Barbara Staehle, Dirk Staehle
University of Würzburg
Institute of Computer Science
Würzburg, Germany
{pries|bstaehle|dstaehle}@informatik.uni-wuerzburg.de

Viktor Wendel
Multimedia Communications Lab - KOM
TU Darmstadt
Darmstadt, Germany
viktor.wendel@kom.tu-darmstadt.de

ABSTRACT

The complex multi-hop structure of WMNs requires a careful network planning. In this paper, we investigate the usability of Genetic Algorithms (GAs) for such a planning approach. The simplicity of GAs allow us to examine a large number of network configurations in order to optimize the network throughput and to fairly distribute the resources. This is achieved with a max-min fair share throughput distribution and by evaluating node positions, routing configurations, and channel assignments. We adapt standard genetic operators and evaluate the influence of the operators on the performance. The results show that GAs are well-suited for planning WMNs.

Categories and Subject Descriptors

C.2.1 [**COMPUTER-COMMUNICATION NETWORKS**]: Network Architecture and Design—*Wireless communication*
; G.1.6 [**NUMERICAL ANALYSIS**]: General—*Optimization*

General Terms

Algorithms, Performance

Keywords

Wireless Mesh Networks, Planning, Optimization, Routing, Genetic Algorithms

1. INTRODUCTION

Wireless Mesh Networks (WMNs) should have the so-called self*-properties, i.e. they are self-organizing, self-configuring, and self-healing and are thus gaining an increasingly important role in next generation wireless networks. However, the complex structure of WMNs and the huge parameter space require a careful planning of these networks. The planning process includes node placement, routing, channel allocation, and a fair resource sharing scheme for the users.

The problem to solve here is NP-hard, which exacerbates to find optimal solutions for networks consisting of more than a handful of nodes. Until today, many linear programming algorithms have been proposed addressing parts of this optimization problem, e.g. [1, 6]. Due to the high complexity of these algorithms, they cannot be applied to large-scale WMNs. Genetic Algorithms (GAs) are a good alternative because of their simplicity and ability to optimize large WMN scenarios.

GAs are based on the idea of natural evolution and are used to solve optimization problems by simulating the biological cross of genes. A randomly created population of individuals represents the set of candidate solutions for a specific problem. The GA evaluates the quality of each individual by applying a fitness function. The best individuals are selected for the new population. However, the selection without any other operation on the individuals would not lead to a genetic optimization. Therefore, two operators, crossover and mutation, are used to create additional new individuals.

In our previous works [9,10], we use genetic optimization to plan the WMN backbone without considering end user location and with fixed router and gateway positions. In this paper, we consider fixed end user locations and build the wireless mesh backbone based on the user distribution. Thus, the location and the number of gateways and routers is not fixed and part of the genetic optimization in addition to routing and channel assignment.

The remainder of this work is organized as follows. Section 2 introduces the WMN architecture, shows the challenges of wireless network planning, and presents the work related to WMN planning. This is followed by Section 3 describing our planning approach. In Section 4 the influence of the genetic operators on the performance of the WMN is evaluated and compared to the results of a greedy algorithm. Finally, Section 5 concludes this paper.

2. BACKGROUND & RELATED WORK

Before reviewing the work related to WMN planning, we take a look at the WMN architecture and show the challenges of wireless network planning.

2.1 WMN Architecture

A WMN is normally organized in a tree-like structure shown in Figure 1. At the root of the tree stands a Mesh Gateway (MGW). An MGW bridges traffic between different WMNs or connects the WMN to the Internet. Connected to the MGW are both, end users and Mesh Routers (MRs). An MR is responsible for mesh relaying, meaning that it is capable of forming an association with its neighbors and forwarding traffic on behalf of other MRs. An MR can be equipped with one or more wireless interfaces. Using several wireless interfaces, the interference in a WMN can be reduced but the complexity of the channel assignment for the interfaces increases. In general, there can be more than one MGW, whereby mesh routers connected to one gateway can be considered as a tree and the complete WMN as a forest.

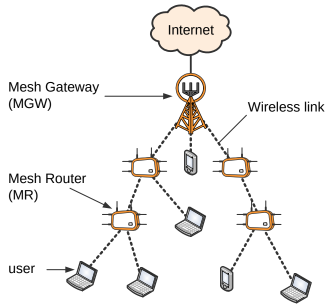


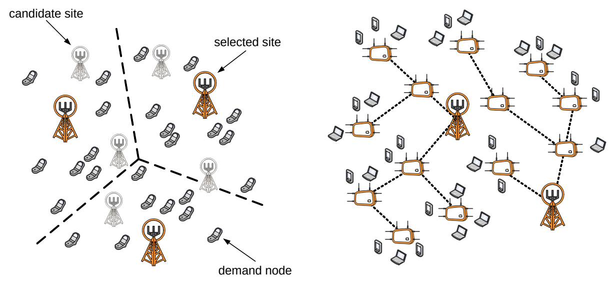
Figure 1: Wireless Mesh Network Tree.

2.2 Wireless Network Planning

The planning of wireless mesh networks can be applied to a variety of wireless networks, like WiMAX, WLAN, and sensor networks. Although the network technology changes, the planning challenges remain similar. In contrast to traditional cellular network planning, the planning and optimization of WMNs is much more complex. A widely used concept for cellular network planning is the demand node concept introduced by Tutschku [14] and illustrated in Figure 2(a).

The algorithm first looks for the demands of cellular services. Therefore, different demographic areas are taken into account. For example, more phone calls occur in urban areas than in rural areas. According to these demographic regions, a different number of demand nodes are set up like shown in Figure 2(a). In addition to the demand nodes, candidate sites for base stations are inserted into the optimization algorithm. As each base station is able to support a fixed amount of users in cellular systems of the second generation, candidate sites are selected for base station placement in such a way that all demand nodes can be served with a certain probability.

In contrast, the planning of WMNs is much more complex. Not only the covered area or the number of end users has to be considered, but also the capacity and the interference of the relaying links. The capacity of a link does not only depend on the distance between two mesh points, but also on the interference which in turn depends on the used channels. Looking again at Figure 2(a), we can see that the channel assignment has to be performed in such a way that neighboring base stations do not use the same channel. In WMNs, such as shown in Figure 2(b), each mesh point can be equipped with multiple interfaces which can be assigned one channel each.



(a) Cellular network planning. (b) WMN planning.

Figure 2: Comparison of traditional cellular network planning and wireless mesh network planning.

In addition to the more complex channel assignment in WMNs compared to traditional cellular networks, also the routing has to be considered. In a fixed wireless mesh network where each mesh node is equipped with multiple interfaces, the Modulation and Coding Scheme (MCS), the interference from neighboring nodes, and the number of flows traversing a link have to be taken into account for the routing decision.

This complex planning has attracted the interest of various researchers and Internet providers. Hence, a number of papers have been published on the problem of planning WMNs and estimating their performance. We divide the related work into three parts. The first part shows general WMN planning approaches. In the second part, the work related to channel assignment and routing is presented. Finally, we present papers working with genetic algorithms for planning radio networks.

2.3 WMN Planning Using Optimization Techniques

Sen and Raman [12] introduce a variety of design considerations and a solution approach which breaks down the WMN planning problem into four tractable parts. These sub-problems are interdependent and are solved by heuristics in a definite, significant order. The evaluations of the presented algorithms show that they are able to generate long-distance WLAN deployments of up to 31 nodes in practical settings.

Other related works [1,6] deal with creating a wireless mesh network model, planning its parameters, and evaluating the solutions via linear programming. He et al. [6] propose mechanisms for optimizing the placement of integration points between the wireless and wired network. The developed algorithms provide best coverage by making informed placement decisions based on neighborhood layouts, user demands, and wireless link characteristics. Amaldi et al. [1] propose other planning and optimization models based on linear programming. The aim is to minimize the network installation costs by providing full coverage for wireless mesh clients. Thereby, traffic routing, interference, rate adaptation, and channel assignment are taken into account.

2.4 Routing and Channel Assignment

One of the first contributions on channel assignment is presented by Raniwala and Chiueh [11]. The channels are assigned according to the expected load evaluated for shortest path and randomized multi-path routing. It is shown that by using only two network interface cards per mesh point, the throughput increases up to eight times. In contrast to [11], Chen et al. [4] do not only consider the expected load for the channel assignment, but also consider the link capacities. Based on the link metrics, called expected-load and expected-capacity, the channel assignment is optimized using simulated annealing.

Raniwala and Chiueh [11] and Chen et al. [4] only consider non overlapping, orthogonal channels. Mohsenian Rad and Wong [7,8] instead also consider partially overlapping channels and propose a congestion-aware channel assignment algorithm. It is shown that the proposed algorithm increases the throughput by 9.8 % to 11.4 % and reduces the round trip time by 28.7 % to 35.5 % compared to the approach of Raniwala and Chiueh [11].

2.5 Genetic Algorithms for Radio Network Planning

Badia et al. [2] use genetic algorithms for joint routing and link scheduling for WMNs. The packet delivery ratio is optimized depending on the frame length. It is shown that genetic algorithms solve the studied problems reasonably well, and also scale, whereas exact optimization techniques are unable to find solutions for larger topologies. The performance of the GA is shown for a single-rate, single-channel, single-radio WMN.

Vanhatupa et al. [15] apply a genetic algorithm for the WMN channel assignment. Capacity, AP fairness, and coverage metrics are used with equal significance to optimize the network. The routing is fixed, using either shortest path routing or expected transmission times. Compared to manual tuning, the algorithm is able to create a network plan with 133 % capacity, 98 % coverage, and 93 % costs. Furthermore, the algorithm needs 15 minutes for the optimization whereas the manual network planning takes hours.

In contrast to the related work, we combine the different optimization approaches and optimize the node placement, the routing, and the channel allocation to increase the overall throughput while still maintaining a max-min fair throughput allocation between the nodes. This is done for a multi-channel, multi-radio, multi-rate WMN.

3. WMN PLANNING APPROACH

In this section, we show the parameters which we have to consider and to evaluate in order to achieve a near-optimal WMN solution, meaning that the throughput in the WMN is fairly shared among the mesh points.

3.1 Problem Formulation

We define a WMN as a set of $\mathcal{N}$ nodes $n_1, ..., n_N$, a set of $\mathcal{U}$ users $u_1, ..., u_U$, and a set of links $\mathcal{L}$ connecting the users to the nodes and the nodes among each other. A subset $G \subseteq \mathcal{N}$ contains the gateway nodes which are connected to the Internet. The remaining nodes $n_i \in \mathcal{N} \setminus G$ are either routers connecting the users via a fixed path to a gateway or disabled nodes if no user is connected to them. The path from user u_i to a gateway is denoted as $\mathcal{P}_i$ and consists of a set of links, $\mathcal{P}_i \subseteq \mathcal{L}$. Thus, the users and routers connected to one gateway can be considered as a tree and the complete WMN as a forest. This structure is used for the optimization with genetic algorithms.

3.2 Fairness and Capacity in Wireless Mesh Networks

To achieve a fair resource distribution among the mesh points, we use a max-min fair share approach introduced by Bertsekas and Gallager [3]. A solution is max-min fair if no rate can be increased without decreasing another rate to a smaller value. Max-min fairness is achieved by using an algorithm of progressive filling. Firstly, all data rates are set to zero. Then, the data rates of all flows are equally increased until one flow is constrained by the capacity set. This is the bottleneck flow and all other flows have to be faster than this one. Afterwards, the data rates of the remaining flows are increased equally until the next bottleneck is found. This procedure is repeated until all flows are assigned a data rate.

Before assigning the data rates to the flows, the capacity of the network has to be estimated. Therefore, we first have to estimate the link capacities. The capacity of a single link is determined by the pathloss and the Signal to Noise Ratio (SNR). For the pathloss calculation, we use a modified COST 231 Hata [5] pathloss model for carrier frequencies between 2 GHz and 6 GHz. The model is proposed by the IEEE 802.16 working group as the WiMAX urban macrocell model, but is also valid for WLAN mesh networks and is defined as

$$PL = 35.2 + 35 \cdot log_{10}(d(n_i, n_j)) + 26 \cdot log_{10}\left(\frac{f}{2}\right). \quad (1)$$

Here, f denotes the operating frequency and d denotes the euclidean distance between mesh points n_i and n_j. The pathloss model is used to calculate the SNR which is required to determine the maximum achievable throughput. The SNR is calculated as

$$\gamma_{n_i,n_j} = T_x - PL(n_i, n_j, f) - (N_0 + 10 \cdot log_{10}(W)), \quad (2)$$

where T_x is the transmit power, N_0 is the thermal noise spectral density (-174 dBm/Hz), and W is the system bandwidth. Now, the Modulation and Coding Scheme (MCS) mcs is selected with an SNR requirement γ^*_{mcs} that is smaller or equal to the link's SNR γ_{n_i,n_j}. The MCS is chosen in such a way that the frame error rate lies below 1 %. If the SNR requirement for the most robust MCS cannot be met, the two mesh points n_i and n_j are not within communication and interfering range.

Having computed the maximum data rate of each link according to the pathloss, we now have to calculate the capacity of each link taking interference from neighboring mesh points into account. In [9, 10], we use collision domains for calculating the link capacities. However, we showed in [13] that cliques can utilize the resource more efficiently while still achieving a max-min fair throughput allocation.

The proposed algorithm, called extended Effective Load Based Algorithm (extended ELBA), first calculates a contention graph for links, which are not allowed to be active simultaneously. Then it finds cliques in this graph and uses them to assign a max-min fair throughput allocation to the flows in the network. The algorithm uses end-to-end flows between users and gateways. A flow $f_i \in \mathcal{F}$ from a user u_i to a gateway g is always fixed to a path $\mathcal{P} \subseteq \mathcal{L}$, which are all links traversed from u to g. The algorithm assigns throughputs rather to flows than to users, but the throughput assigned to $flow_i$ is the same throughput which node u_i receives. Each link (i, j) has an individual link rate $r_{i,j}$ defined by the underlying MCS. A flow with rate b_k traversing this link would occupy $\frac{b_k}{r_{i,j}}$ of the link's bandwidth. Therefore, the link is active for $\frac{b_k}{r_{i,j}}$ of the time. Now, let $\mathcal{K}_{i,j}$ be the set of flows traversing link (i, j). Then the link is active for

$$\Theta_{i,j} = \sum_{k \in \mathcal{K}_{i,j}} \frac{b_k}{r_{i,j}} \quad (3)$$

percent of time. Note that $\sum_{k \in \mathcal{K}_{i,j}} b_k \leq r_{i,j}$ as the link can be active no more than 100 % of time. Using this, we can define the activity percentage of a clique $\mathcal{C}$ as

$$\Theta_{\mathcal{C}} = \sum_{(i,j) \in \mathcal{C}} \sum_{k \in \mathcal{K}_{i,j}} \frac{b_k}{r_{i,j}}. \quad (4)$$

During the process of fixing flows, i.e. assigning fixed throughputs to them, the maximum rate for unassigned flows in a clique has to be known. From the fact that a clique can be active no more than 100 % of time, it can be stated that the maximum rate available for unassigned flows $b_{\mathcal{C}}$ is

$$b_{\mathcal{C}} = \frac{1 - \sum_{(i,j) \in \mathcal{C}} \sum_{k \in \mathcal{K}^a_{i,j}} \frac{b_k}{r_{i,j}}}{\sum_{(i,j) \in \mathcal{C}} \sum_{k \in \mathcal{K}^u_{i,j}} \frac{b_k}{r_{i,j}}} \quad (5)$$

where $\mathcal{K}^a_{i,j}$ is the set of unassigned flows and $\mathcal{K}^u_{i,j}$ is the set of assigned flows. Thus, we can define the load of a clique as

$$m_{\mathcal{C}} = \sum_{k \in \mathcal{F}} \sum_{(i,j) \in \mathcal{P}_k \cap \mathcal{C}} \frac{1}{r_{i,j}} = \sum_{(i,j) \in \mathcal{C}} \frac{n_{i,j}}{r_{i,j}} \quad (6)$$

with $n_{i,j}$ as the number of flows traversing link (i, j).

Now the algorithm works in two steps. The first step is an initialization phase. During the initialization all cliques are computed, all flows are set as unassigned, the load of all cliques is calculated, and the capacity of each link is set to 1. The second phase iterates over the bottleneck cliques, one per step. In each step the bottleneck clique $\mathcal{C}^*$ is the clique $\mathcal{C}$ with the lowest throughput $b_{\mathcal{C}}$ per flow through $\mathcal{C}$. The rates of all flows in the bottleneck clique are then set to $b_{\mathcal{C}}$ and the flows marked as assigned (fixed). Next, the capacity and the load of all cliques containing at least one link traversed by at least one of the just assigned flows have to be recalculated. Finally, all cliques with a load of no more than 0 are removed from the clique corpus. Then, the throughputs per flow $b_{\mathcal{C}}$ for all cliques are recalculated and a new bottleneck clique is determined. This iteration stops as soon as the clique corpus is empty. The algorithm is shown in detail in Algorithm 1.

Algorithm 1 Extended ELBA

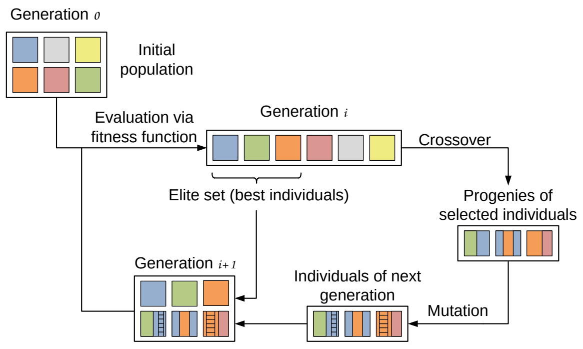
```
Input:    network solution $\mathcal{N}$
Output:   a throughput allocation for $\mathcal{N}$
Variables:
$\mathcal{F}$ – Set of all flows in $\mathcal{N}$
$\mathcal{O}$ – Set of all unassigned(open) flows in $\mathcal{N}$
$\Omega_C{}^*$ – The clique corpus, i.e. the set of all collision cliques in $\mathcal{N}$
$r_{i,j}$ – max. rate of link between node i and node j
$n_{i,j}$ – number of flows traversing the link between node i and node j
$m_C$ – load of clique $\mathcal{C}$
$p_C$ – capacity of clique $\mathcal{C}$
$b_C$ – throughput per flow through clique $\mathcal{C}$
$\mathcal{B}$ – set of bottleneck flows (all flows traversing the bottleneck clique)

1:  $\mathcal{O} = \mathcal{F}$ {all flows are unassigned}
2:  $\Omega_C{}^* = \Omega_C$ {the clique corpus}
3:  $m_c = \sum \frac{n_{i,j}}{r_{i,j}}, \mathcal{C} \in \Omega_C$
4:  $p_c = 1, \mathcal{C} \in \Omega_C$
5:  while $\Omega_C^* \neq \emptyset$ do
6:      $b_c = p_c / m_c$ for all $\mathcal{C} in \Omega_C$
7:      $C^* = \min_{C \in \Omega_C{}^*} b_C$
8:      $\mathcal{B} = \{k \in \mathcal{O} | P_k \cap C^* \neq \emptyset\}$
9:      $b_k = b_{C^*}$ for all $k \in \mathcal{B}$
10:     $\mathcal{O} = \mathcal{O} \setminus \mathcal{B}$
11:     $p_c = p_c - \sum_{k \in \mathcal{B}} \sum_{(i,j) \in \mathcal{P}_k \cap C} \frac{b_C^*}{r_{i,j}}$
12:     $m_c = m_c - \sum_{k \in \mathcal{B}} \sum_{(i,j) \in \mathcal{P}_k \cap C} \frac{1}{r_{i,j}}$
13:     $\Omega_C{}^* = \mathcal{C} \in \Omega_C{}^* | m_c > 0$
14: end while
```

3.3 Genetic Algorithm Model

The throughputs per flow are needed to estimate the goodness of a WMN solution found by the genetic algorithm. In the following, we explain the steps of the genetic algorithm for the WMN optimization.

The complete procedure of our genetic algorithm is shown in Figure 3. Firstly, a random population is created which contains a predefined number of individuals. The fitness of each individual is evaluated using the fitness function and the individuals are ordered according to the fitness value. The best individuals, the elite set, are kept for the new population. Afterwards, the crossover and mutation operator are used to create the remaining number of individuals for the new population. The procedure is repeated until a satisfying solution is achieved. In the next sections, we explain the steps of our WMN optimization approach in more detail.



Generation 0
Initial population
Evaluation via fitness function
Generation i
Crossover
Elite set (best individuals)
Progenies of selected individuals
Generation i+1
Individuals of next generation
Mutation

Figure 3: Basic functionality of our genetic algorithm.

3.3.1 *Individual Encoding*

Each individual of a population represents a WMN deployment. In order to perform the crossover and mutation operations, the WMN has to be encoded in such a way that a simple modification of genes is possible. We use a two-dimensional string of integers, where the node locations are ordered along the first dimension, whereas the second dimension represents a location's properties, the node ID, the ID of the next hop towards the gateway, the channel of the link to the next hop, and the type of the location.

If the ID of the next hop is set to -1, either the node is a gateway or the node is not connected to any other node. The type can be either "user", represented by -1, "unused candidate location", represented by 0, "gateway", represented by 1, or "router", represented by 2. Table 1 shows an example of the second dimension of an individual coding.

Table 1: Individual Encoding example.

	gateway	router	unused	conn. user	disconn. user
node-ID	2	12	13	45	52
next hop	-1	2	-1	12	-1
channel	-1	1	-1	2	-1
type	1	2	0	-1	-1

3.3.2 *Creation of Initial Population*

Most commonly, the initial population is created randomly and the optimization is left to the GA. The random creation of an individual has four steps: Firstly, the gateways are placed. For this purpose, a random number of gateways is set up among randomly chosen candidate locations. In the second step, the routers are connected to the gateways. Afterwards, the users are connected to the routers or gateways. Finally, all unnecessary routers are disabled. In Section 3.5 we describe another, more intelligent possibility for creating the initial population.

3.3.3 *Evaluation via Fitness Function*

The fitness of an individual is estimated using the allocated end user throughputs obtained as shown in Section 3.2. A simple fitness function which would lead to a max-min fair throughput allocation could be

$$f_1(I) \quad = \quad min(\tau(u_i) | u_i \in \mathcal{U}(I)), \qquad (7)$$

where $\tau(u_i)$ is the throughput of user u_i of individual I. This fitness function however, would only take the worst throughput in the network solution into account, without regarding all other users. Therefore, the mean network throughput should also be considered. This leads to the following fitness function

$$f_2(I) \quad = \quad \frac{\sum_{u_i \in \mathcal{U}(I)^*} \tau(u_i) \cdot i^\alpha}{\sum_{u_i \in \mathcal{U}(I)} i^\alpha}, \qquad (8)$$

where u_i is the i-th user in a list of all users of $\mathcal{U}(I)^*$ ordered by their throughput in a descending order.

This fitness function considers all throughputs received by the users, but weights them according to α. For $\alpha = 0$ all throughputs are weighted equally. For $\alpha \to \infty$ only the lowest throughput will be relevant which is equivalent to f_1.

3.3.4 Selection Principle

After the evaluation of a population, we select a set of individuals, which have the highest fitness of all and keep them unchanged in the new generation. This set is called the elite set. To get to the same population size, the remaining individuals are created from selected members of the previous generation.

Let the population size be p, the elite set size e, then the e best individuals from generation i are transfered to generation $i + 1$. Additionally, $p - e$ progenies are created from individuals of generation i. The e members of the elite set and the $p - e$ progenies form the new population at generation $i + 1$, again of size p. The parents from which the progenies are created are selected the following way. One parent is chosen with a probability proportional to its relative fitness among all individuals of its generation. The other parent is chosen with a uniformly distributed probability among all individuals. This method ensures that "good" individuals are chosen as parents more often. However, choosing the second parent with a uniform probability ensures that not only the best solutions are taken to create new individuals. This is done in order to prevent the algorithm from running into local optima.

3.3.5 Crossover

To create the progenies, different crossover operators are now applied to the selected number of individuals. For the cross of genes, we introduce three different mesh-specific crossover variants, the *Area Crossover*, the *Cell Crossover*, and the *Subtree Crossover* because we have already shown in [9] that the standard *Two-Point Crossover* does not perform well for WMNs. In contrast to [9], only one progeny is created by the crossover variants.

Area Crossover
The Area Crossover selects a random area in the topology and copies the complete structure in one individual, i.e. the configuration of all nodes contained in the area as well as all links between all nodes in the area. Links crossing the border of the area have to be regarded separately because they can either lead to candidate locations which are unused or create circles in the connection graph. Therefore, a repair function is designed. Its purpose is to bring the network back into a consistent state with only minimal changes.

Figure 4 shows an Area Crossover, where the marked area of individual I_2 is copied into I_1. The black squares are the gateways, the gray ones the enabled routers, and the users are shown as a circle. The lower left figure shows the resulting progeny after the crossover. User u_{12} is not connected to any router. In order to connect u_{12} to a gateway, a repair function is applied which activates the router r_5 and connects user u_{12} over the router to gateway g_3.

Cell and Subtree Crossover
The Cell Crossover selects a random gateway from individual I_2 and copies its whole cell, i.e. all nodes connected to the gateway into individual I_1. The Subtree Crossover instead only copies a subtree and not the complete cell into another individual, i.e. copies the type (router) of the node as well as the types of its children (router, user). Thus, a Subtree Crossover with a selected gateway is the same as the Cell Crossover.

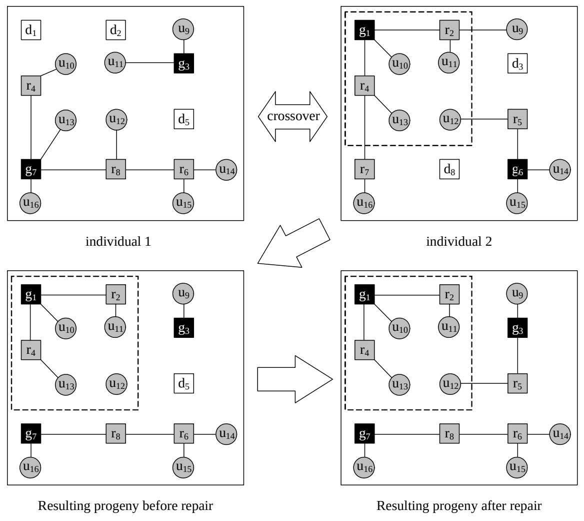


Figure 4: Area Crossover from I_2 into I_1.

An example of the Subtree Crossover is shown in Figure 5. Router r_2 is chosen for the crossover. All routers and users whose path to the gateway cross r_2 are copied to the resulting progeny. User u_6 which was connected to node 2 in individual I_1 is still connected to node 2 similar to user u_{11} who is connected to node 4. As router r_1 is not relaying any traffic in the resulting progeny, it is disabled. Finally, a randomly selected router, in our case router r_2, is converted to a gateway.

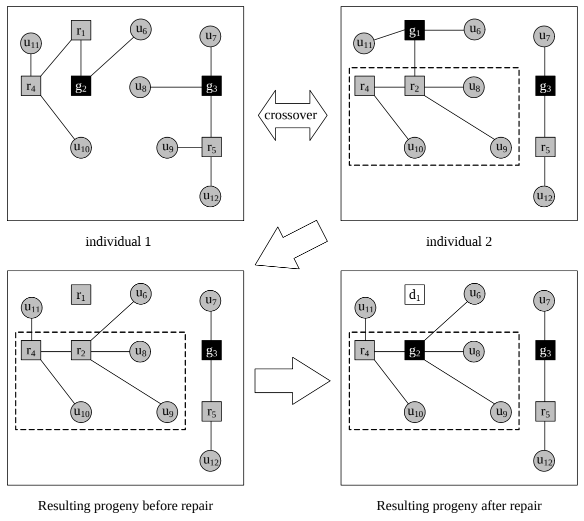


Figure 5: Subtree Crossover from I_2 into I_1 with r_2 as root.

3.3.6 Mutation

In contrast to crossover operations, mutations only slightly modify the individuals and do not create completely new ones. The following six mutations have been created.

Change Channel
The most simple mutation operation is Change Channel. As the name implies, it changes the channel used by a randomly selected link. The new channel is chosen randomly from a set of available channels with the only restriction that the previously used channel is not allowed.

Change Parent
This mutation operation changes the parent, i.e. the next hop of a randomly chosen active non-gateway node. “Active” means that the node can be either a user or a router, but not a disabled candidate location. The new next hop is chosen randomly from the set of potential neighbors of the node. To avoid circles, these nodes must not part of the node's subtree.

Router to Gateway
This mutation randomly chooses a router and changes its type to gateway. Its complete subtree remains unchanged and only the links between the new gateway and its parent are deleted. A repair function is applied in order to achieve a good balance in terms of number of nodes between old and new gateway.

Gateway to Router
This operation is the counterpart to the previous operation. A randomly selected gateway is mutated to a router. A parent is chosen to attach the complete subtree to another gateway with performing as few changes as possible.

Move Gateway
This operation randomly chooses a gateway from all gateways of the individual and generates a list of possible new locations. Possible new locations are all direct neighbors which are routers of the gateway's subtree. One of them is randomly chosen and its type changed to gateway. The old gateway's type is changed to router.

Disable Router
One method to improve the performance of the network is to disable unnecessary routers. This operation disables one randomly chosen router and deletes the link from the node to its parent. Afterwards, its whole subtree is disabled. Finally, the repair operation tries to find a new routing for those nodes.

The description of the mutation operators closes the circle of the genetic optimization. In order to compare the results gathered by the genetic algorithm, we developed a greedy algorithm which is explained in detail in the next subsection.

3.4 Greedy WMN Planning

Our greedy algorithm works in four steps. In step one the gateways are placed, in step two all routers are connected to the gateways so that every router has the shortest possible path to one gateway. In the next step, the users are connected to the network and finally, all unused routers are disabled.

3.4.1 Gateway Placement

The gateways are placed according to a user-density function. For each candidate location c_i, a value $g(c_i)$ is calculated indicating its qualification to become a gateway. The value represents how many user locations u_j are close to the candidate location. Therefore, a maximum distance $limit$ is defined. All users within this distance are used to calculate $g(c_i)$, all other users are not taken into account. $Limit$ depends on the length l_{city} and width w_{city} of the underlying city and number of gateways to be placed n_g. It is calculated as

$$limit = \sqrt{(l_{city}/n_g + 1)^2 + (w_{city}/n_g + 1)^2}. \qquad (9)$$

Let us now denote the relevant set of user locations with a distance to candidate location c_i smaller than $limit$ as Rel_{c_i} and the euclidean distance between any two locations l_1 and l_2, as $dist(l_1, l_2)$. $g(c_i)$ for a candidate location c_i is then

$$(c_i) = \sum_{u_j \in Rel_{c_i}} \frac{limit - dist(c_i, u_j)}{limit}. \qquad (10)$$

This works well for just one gateway, but if more gateways have to be placed, this function would always select the same candidate location. Even if that location would be removed from the set of possible candidate locations after placing a gateway, the second gateway would probably be placed very close to the first one. To prevent this, the set of relevant users of already existing gateways must be removed from the set of relevant users when searching for the next gateway candidate location. This leads to an equal distribution of gateways within the topology if the users are uniformly distributed. Otherwise, the gateways are located in areas with higher user density.

3.4.2 Connecting the Routers

Once the gateways are placed in the city, the routers have to be connected to them. This is done in a shortest-path manner, i.e. each router is connected to exactly one gateway in a way that the number of hops from router to gateway is minimal.

In the first step, all routers directly connectible to a gateway are connected to the closest gateway. Afterwards, the routers which can be connected on a two-hop path to the gateway are attached to the tree. This is repeated until all routers which can be connected to a gateway are added. The resulting backbone network is only temporary, as some routers may be disabled later if no user is connected to them after the next step.

3.4.3 Connecting the Users

For connecting the users, all possibilities are tested and evaluated and the best one is chosen. All possibilities means all possible channel configurations for all possible next hops of the users. The possible next hops are those candidate locations in the user's neighborhood, which are gateways or routers connected to gateways. This search is repeated until all connectible user are attached.

3.4.4 Channel Allocation

Whenever a link from a router or user is created, the greedy algorithm has to choose a channel for this link. We designed two ways of determining the channel. The first one always chooses the least used channel. The second way includes the search for a channel in the evaluation. This is only possible when creating links between users and routers. Using this method means, not only all users and all possibilities of next hops for each user are tested and evaluated, but also all possible channels for the links. This, of course multiplies the number of solutions which have to be tested and thus, the runtime of the algorithm.

3.5 Greedy Creation of Initial Population

Besides the comparison of the genetic algorithm with a greedy algorithm, we implemented a greedy algorithm for the creation of the initial population. Using the randomly creation of the population, very different individuals are created and the population contains several individuals with low fitness. To start with an initial population with higher fitness compared to the randomly created population, we use a randomized greedy algorithm. The creation works as follows.

3.5.1 *Gateway Placement*

The gateway placement works similar to the greedy WMN planning. However, for every gateway to place, the randomized algorithm does not only seek the best candidate location. Instead, it finds the five best locations and one of them is selected randomly to place a gateway. Fewer than five locations would result in too similar solutions, more than five would result in a higher probability of too bad solutions which would be contradictory to the idea of creating individuals with the rather time-consuming greedy algorithm.

3.5.2 *Connecting the Routers*

This step works exactly like the greedy WMN planning.

3.5.3 *Connecting the users*

In contrast to the greedy WMN planning, the randomized algorithm does not evaluate every unfixed user location for its next hop, but it chooses a user node randomly and evaluates all possible ways of linking it with the network. The least used channel at that moment is chosen for that link. This leads of course to worse individuals but it is a lot faster than searching for the best user in every step. A comparison of a randomly created initial population with an initial population created with the randomized greedy algorithm can be found in the next section.

4. PERFORMANCE EVALUATION

The performance of a genetic algorithm is an umbrella term for the efficiency of the genetic operators and the influence of the fitness function on the resulting solution. In this section, we evaluate the influence of all genetic operators and compare our optimized GA with the greedy algorithm.

4.1 Simulation Settings

Before showing the influence of the GA parameters on the fitness value, we list the parameters of the underlying physical characteristics. These parameters only affect the characteristics of the network connections and not the performance of the GA. Therefore, we do not consider their impact on the resulting solutions. The parameters are listed in Table 2 and denote the used carrier frequency, the channel bandwidth, the available channels, and the antenna power.

The functionality of the genetic algorithm is evaluated for different topologies. The user and candidate locations are created randomly on a grid, meaning that the locations are distributed along the area of the topology according to a uniform distribution. The five different topologies used for the evaluation are shown in Table 3. We limit the maximum number of allowed gateways in dependence of the topology to reduce the costs for the provider.

Table 2: Fixed parameters for all simulations of the GA.

Parameter	Value
carrier frequency	3500 MHz
channel bandwidth	7 MHz
available channels	3500 MHz; 3510 MHz; 3520 MHz; 3530 MHz
antenna power	25 dBm

Table 3: Testing topologies.

Name	Dimension	User locations	Candidate locations	Max. gateways
$city\_25\_25$	250 m x 250 m	25	25	2
$city\_25\_40$	400 m x 400 m	25	40	2
$city\_42\_37$	425 m x 250 m	42	37	2
$city\_60\_80$	1000 m x 480 m	60	80	3
$city\_100\_100$	1200 m x 800 m	100	100	5

4.2 Influence of Crossover and Mutation Rates

Firstly, we want to find out the main effects of the crossover and mutation operation using fitness function f_1 and the $city\_42\_37$ topology. The main effects are shown in Figure 6. For the crossover operations a 0 means that this specific crossover is not used and 1 means that it is performed. From the figure we can see that a higher rate of Area Crossover has a negative influence on the genetic algorithm performance. In contrast, the Subtree Crossover and the Cell Crossover have a positive effect. The Change Parent Mutation and the Router Off Mutation have a significantly negative effect on the genetic algorithm if the higher rates are chosen. The Router Off Mutation of course influences the fitness of the mutated individual negatively for the applied fitness function, because fewer routers mean more disconnected users and fewer routing paths and the hardware costs are not included in the fitness function. The negative influence of the Change Parent Mutation can be explained with the fact that this mutation significantly changes the structure of the network which should be avoided for later generations.

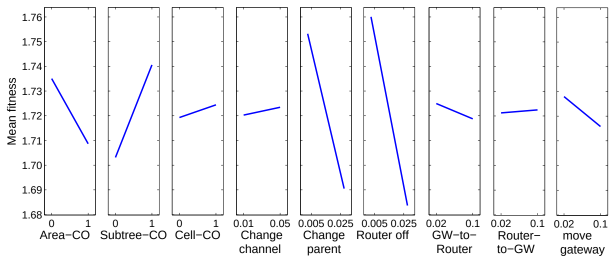


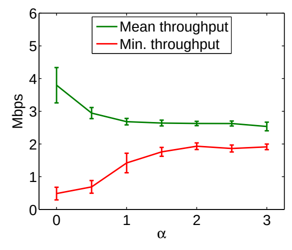
Figure 6: Influence of crossover and mutation operations on the fitness value.

4.3 Fitness Function Analysis

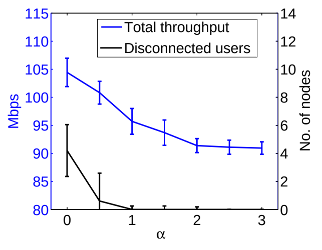
Now, we want to take a closer look at parameterization of fitness function f_2 which tries to maximize the minimal throughput. The goal of the following simulations is to evaluate how the GA reacts on different input parameters, i.e. how it optimizes the searched network for different requirements.

The main parameter of fitness function f_2 is α which indicates how strong smaller throughputs are weighted over higher throughputs. Therefore, it seems reasonable to vary α and to observe the minimal throughput a user receives as well as the mean throughput and the total throughput in the network. We are also interested in knowing how many users could not be connected. Each simulation was performed for two different topologies, $city\_25\_40$ and $city\_60\_80$ and repeated 10 times. In Figure 7 the results are shown for values of α between 0 and 3 in steps of 0.5.

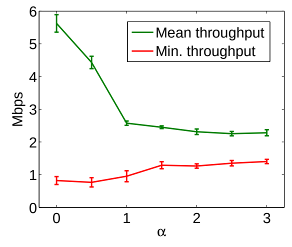
As we can see from the figures, with an increasing α, the minimal throughput in the network increases at the cost of a decrease of both the mean and the total throughput. Furthermore, the num-



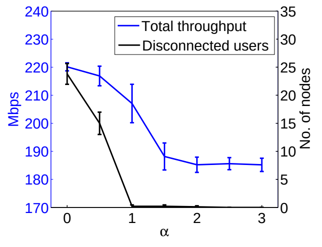
(a) $city\_25\_40$.



(b) $city\_25\_40$.



(c) $city\_60\_80$.



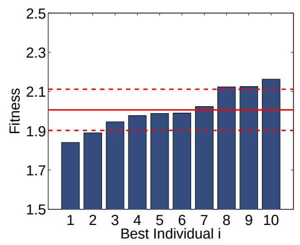
(d) $city\_60\_80$.

Figure 7: Influence of α on various network parameters.

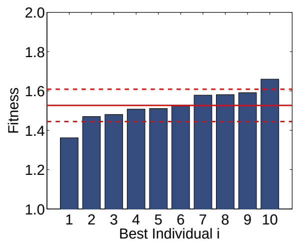
ber of disconnected user nodes is 0 for $\alpha > 1$ for both topologies. This means that the genetic algorithm tries to connect every user because a disconnected user would have the worst impact on the fitness function as it would be counted as a throughput of 0 Mbps received at the node. Looking at the minimal throughput, the best results are achieved with a value of α between 2 and 3. From the small confidence intervals for $\alpha = 3$ we might conclude that the resulting networks do not vary much which means that only a few optimization runs have to be performed. To underline this, we take a closer look at the variance of the best network solutions in the next subsection.

4.4 Variance

When the fitness of the best individuals strongly varies, it is necessary to have a number of runs to find the best result with the applied parameters, otherwise one run would be enough. For this reason, we performed test runs with fitness functions f_2 for topology $city\_25\_40$ and $city\_60\_80$, setting α to 3. The run was repeated 10 times. In Figure 8 the best results from each of the 10 runs are compared for both of the aforementioned topologies. Additionally, the mean of all 10 samples as well as the standard deviation are plotted.



(a) $city\_25\_40$.



(b) $city\_60\_80$.

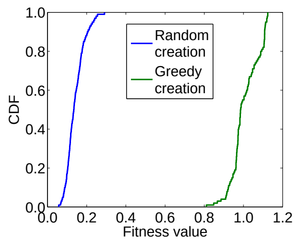
Figure 8: Fitness variance of the best solution found for fitness function f_2.

We can see that the variance of the fitness of the best individual is not depending on the topology. In both figures, the best result is about 20 % better than the worst one. Thus, as the results found by the different runs differ up to 20 %, the statement from the previous subsection that a few runs are sufficient is disproved. It is required to perform several runs of the GA for one problem to retrieve a good network solution.

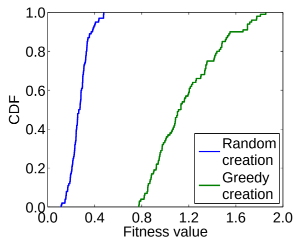
4.5 Individual Creation

The variance of the resulting network solutions also strongly depends on the initial population. As described in the previous section, the easiest way is to create the initial population in a completely random manner. This means that the number of gateways is chosen randomly between one and a predefined maximum number. The positions of the gateways are chosen randomly, too. Then, the routers are added to a random next hop, already connected to a gateway. Finally, the users are added in a random manner. Those individuals are very different from each other and such a population contains several solutions with low fitness. To start with an initial population with higher fitness compared to the completely random created population, we use a randomized greedy algorithm. By this, a better initial population is created whose members are still sufficiently different from each other. Alone by calculating the five best gateway positions per step and choosing one of them randomly, the emerged individuals have completely different gateway positions and subsequently a complete different routing. This is sufficient to satisfy the demand for a diverse initial population.

In Figure 9, the Cumulative Distribution Function (CDF) of the fitness of the randomly created individuals and the individuals created with the greedy algorithm, is shown. For evaluation, fitness function f_2 was used. The GA is applied for two different topologies, one with 42 possible router/gateway positions and 37 users and the other topology with 25 router/gateway positions and 25 users.



(a) $city\_42\_37$.



(b) $city\_25\_25$.

Figure 9: Comparison of the CDF of the fitness values of a randomly created starting population to the fitness values of a greedily created starting population.

The results show that the initial fitness is much higher for both scenarios. In the $city\_25\_25$ scenario, the fitness of the randomly created individuals varies between 0.15 and 0.5, whereas the individuals created with the greedy algorithm varies between 0.8 and 1.8. In addition to the higher fitness, the large variation of the fitness values indicate that the created individuals satisfy the demand for a diverse initial population.

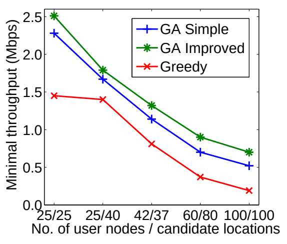
4.6 Performance Comparison with Greedy Algorithm

In the previous subsection, we have seen that an initial population created using a greedy algorithm has a higher average fitness.

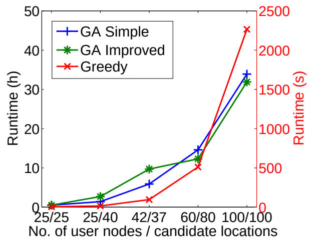
In this subsection, we want to see if the resulting network solutions whose initial population is created with the greedy algorithm, also have a higher fitness. Therefore, we use two variants of the genetic algorithm, the first variant of the GA (GA Simple) and a GA where the initial population is created using a simplified form of the greedy algorithm (GA Improved).

To have an indication for the overall performance of the genetic algorithm, we compare both mechanisms with a greedy algorithm introduced in Subsection 3.4. The optimal network solution cannot be shown because of the large number of nodes and the mutual influences of the different parameters.

The greedy algorithm was applied $maxGW$ times, with $maxGW$ as the maximum number of allowed gateways, cf. Table 3. For each number of gateways between 1 and $maxGW$, one run was performed. The best result of all those runs was taken. In Figure 10(a) the results found by the two GA variants and the greedy algorithm are compared for the 5 different topologies. In Figure 10(b) the runtimes for finding those solutions are shown.



(a) Minimum Throughput.



(b) Runtime.

Figure 10: Comparison of greedy algorithm and GA.

Looking at the figure, we can see that GA Improved is better than GA Simple. Depending on the topology, it outperforms GA Simple by 10-20 %. For the topologies shown in Figure 10, the total throughput of the network solutions is also about 10-20 % better, depending on the topology.

Both variants are visibly better than the solution found by the greedy algorithm. Depending on the topology, the difference lies between 15 % and 200 %. However, the greedy algorithm is a lot faster than the genetic algorithm. While the genetic algorithm ran for several hours, depending on the topology, the greedy algorithm ran only for a few seconds or minutes. Finally, we can state that the GA is visibly better than the greedy algorithm, altough it needs more computation time. However, as establishing a WMN requires some planning and has no real-time requirements, a computation time of about 30 hours for a large network, with 100 user nodes and 100 candidate locations, is acceptable.

5. CONCLUSION

In this paper, we introduced a genetic algorithm for planning and optimizing WMNs. The GA is used for the location planning of WMN gateways and routers as well as for the optimization of the routing and channel allocation. We introduced different crossover and mutation variants and evaluated their influence on the resulting network solution. In addition, we saw that the fitness values of the best individual varies for each evaluation run and thus recommend to perform several runs of the GA.

In order to compare the resulting network solutions, we introduced a greedy algorithm. The results show that the GA outperforms the greedy algorithm by 10-20 % using the normal GA, and about 15-200 % when using an improved GA, whose initial population is created with a greedy algorithm.

In future work, we want to include the monetary costs for gateways and routers in the fitness function and try to find a trade-off between throughput maximization and cost reduction.

6. REFERENCES

[1] E. Amaldi, A. Capone, M. Cesana, I. Filippini, and F. Malucelli. Optimization Models and Methods for Planning Wireless Mesh Networks. *Computer Networks*, 52(11):2159–2171, August 2008.

[2] L. Badia, A. Botta, and L. Lenzini. A Genetic Approach to Joint Routing and Link Scheduling for Wireless Mesh Networks. *Elsevier Ad Hoc Networks Journal*, Special issue on Bio-Inspired Computing:11, April 2008.

[3] D. P. Bertsekas and R. G. Gallager. *Data Networks*. Prentice-Hall, 1987.

[4] Y.-Y. Chen, S.-C. Liu, and C. Chen. Channel Assignment and Routing for Multi-Channel Wireless Mesh Networks Using Simulated Annealing. In *IEEE Globecom 2006*, San Francisco, CA, USA, November/December 2006.

[5] E. Damosso and L. M. Correia. *Digital Mobile Radio Towards Future Generation Systems, COST 231 Final Report*. European Commission, 1999.

[6] B. He, B. Xie, and D. P. Agrawal. Optimizing Deployment of Internet Gateway in Wireless Mesh Networks. *Computer Communications*, 31(7):1259–1275, 2008.

[7] A. H. Mohsenian Rad and V. W. S. Wong. Joint Channel Allocation, Interface Assignment and MAC Design for Multi-Channel Wireless Mesh Networks. In *IEEE Infocom 2007*, pages 1469–1477, Anchorage, AK, USA, May 2007.

[8] A. H. Mohsenian Rad and V. W. S. Wong. Congestion-Aware Channel Assignment for Multi-Channel Wireless Mesh Networks. *Computer Networks*, 53(14):2502–2516, September 2009.

[9] R. Pries, D. Staehle, M. Stoykova, B. Staehle, and P. Tran-Gia. A Genetic Approach for Wireless Mesh Network Planning and Optimization. In *PlanNet2009 workshop in conjuction with the IWCMC*, Leipzig, Germany, June 2009.

[10] R. Pries, D. Staehle, M. Stoykova, B. Staehle, and P. Tran-Gia. Wireless Mesh Network Planning and Optimization through Genetic Algorithms. In *The Second International Conference on Advances in Mesh Networks (MESH)*, Glyfada, Greece, June 2009.

[11] A. Raniwala and T. Chiueh. Architecture and Algorithms for an IEEE 802.11-Based Multi-Channel Wireless Mesh Network. In *IEEE Infocom 2005*, pages 2223–2234, Miami, FL, USA, March 2005.

[12] S. Sen and B. Raman. Long Distance Wireless Mesh Network Planning: Problem Formulation and Solution. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 893–902, New York, NY, USA, 2007.

[13] D. Staehle, B. Staehle, and R. Pries. Max-Min Fair Throughput in Multi-Gateway Multi-Rate Mesh Networks. In *IEEE VTC Spring 10*, Taipei, Taiwan, May 2010.

[14] K. Tutschku. Demand-Based Radio Network Planning of Cellular Communication Systems. In *IEEE Infocom 1998*, San Francisco, CA, USA, March 1998.

[15] T. Vanhatupa, M. Hännikäinen, and T. D. Hämäläinen. Performance Model for IEEE 802.11s Wireless Mesh Network Deployment Design. *Journal of Parallel and Distributed Computing*, 68(3):291–305, March 2008.