A Simulated Annealing Algorithm for Router Nodes Placement Problem in Wireless Mesh Networks

Fatos Xhafa*[,a], Admir Barolli[b], Christian Sánchez[c], Leonard Barolli[d]

[a] *Department of Languages and Informatics Systems, Technical University of Catalonia, Campus Nord, Ed. Omega, C/Jordi Girona 1-3, 08034 Barcelona, Spain.*
[b] *Department of Computers and Information Science, Seikei University, 3-3-1 Kichijoji-Kitamachi, Musashino-Shi, Tokyo 180-8633, Japan.*
[c] *Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Campus Nord - Ed. Omega, C/Jordi Girona 1-3, 08034 Barcelona, Spain.*
[d] *Department of Information and Communication Engineering, Fukuoka Institute of Technology (FIT), 3-30-1 Wajiro-higashi, Higashi-ku, Fukuoka 811-0295, Japan.*

Abstract

Mesh router nodes placement is a central problem in Wireless Mesh Networks (WMNs). An efficient placement of mesh router nodes is indispensable for achieving network performance in terms of both network connectivity and user coverage. Unfortunately the problem is computationally hard to solve to optimality even for small deployment areas and a small number of mesh router nodes. As WMNs are becoming an important networking infrastructure for providing cost-efficient broadband wireless connectivity, researchers are paying attention to the resolution of the mesh router placement problem through heuristic approaches in order to achieve near optimal, yet high quality solutions in reasonable time. In this work we propose and evaluate a Simulated Annealing (SA) approach to placement of mesh router nodes in WMNs. The optimization model uses two maximization objectives, namely, the size of the giant component in the network and user coverage. Both objectives are important to deployment of WMNs; the former is crucial to

*Corresponding author. Address: Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Campus Nord - Ed. Omega, C/Jordi Girona Salgado 1-3, 08034 Barcelona, Spain. Voice: +34 93-413-7880 Fax: +34 93-413-7833

Email addresses: `fatos@lsi.upc.edu` (Fatos Xhafa), `admir.barolli@gmail.com` (Admir Barolli), `csanchez@lsi.upc.edu` (Christian Sánchez), `barolli@fit.ac.jp` (Leonard Barolli)

achieve network connectivity while the later is an indicator of the QoS in WMNs. The SA approach distinguishes for its simplicity yet its policy of neighborhood exploration allows to reach promising areas of the solution space where quality solutions could be found. We have experimentally evaluated the SA algorithm through a benchmark of generated instances, varying from small to large size, and capturing different characteristics of WMNs such as topological placements of mesh clients. The experimental results showed the efficiency of the annealing approach for the placement of mesh router nodes in WMNs.

Key words: Wireless Mesh Networks, Simulated Annealing, Size of Giant Component, User Coverage.

1. Introduction

Node placement problems are known for their capability to model many combinatorial optimization problems related to facility and location concepts. They are showing their usefulness also in modelling optimization problems from Wireless Mesh Networks (WMNs). In their general setting, node placement problems aim at finding optimal placement of facilities ("facilities provides some kind of service to clients") such that the system services, e.g. cost reduction, demand capture, equitable service supply, fast response time, etc. are optimized. In the case of mesh router nodes placement, facilities are mesh routers that provide connectivity to mesh client nodes. In fact, the node placement problem considered in this paper is even more challenging due to two additional characteristics: (a) locations of mesh router nodes are not pre-determined (any available position in the considered area can be used for deploying the mesh routers), and (b) routers are assumed to have different radio coverage area and thus some routers are more powerful than others. This later characteristic is important to explore in view of client density in the deployment area.

Wireless Mesh Networks (WMNs) [1, 12] are becoming an important networking infrastructure due to their low cost and increased high speed wireless Internet connectivity. In WMNs there are have two types of nodes: mesh routers and mesh clients. Mesh routers are similar to normal routers but incorporate also additional functions to support mesh networking, and are usually equipped with multiple interfaces to work with different wireless technologies. Another feature of this type of routers with respect to normal routers is their ability to provide the same coverage with much less transmitter power through multi-hop communications. Also, mesh routers can be installed on a dedicated machine or on a general purpose machine. With regard to mesh clients, they have the necessary functions for mesh networking and could also be able to act as routers but do not have the functionality of a gateway or bridge and their single wireless interface with the hardware and software platform is much simpler than in the case of mesh routers.

The placement of mesh nodes plays an important role in achieving important properties of WMNs such as reliability, robustness, and self-configuration. Indeed, the performance of WMNs is primarily affected by the location of mesh nodes, specifically, that of mesh router nodes of the WMN. However, in a real deployment of WMN the automatic or purely random node placements produce poor performance since the resulting placement could be far from optimal. Moreover, an efficient deployment of mesh router nodes in WMNs may require to take into account specific restrictions and characteristics of real geographic area and therefore one needs to explore different topologies for placing mesh routers.

From a combinatorial optimization perspective, mesh node placement in WMNs is in fact a family of problems, actually, different versions of the problem can be obtained depending on the types of mesh nodes to deploy as well as the objectives to optimize. For instance, in [9, 11, 16, 13] there is considered the gateway placement aiming to optimize the throughput. In [5], the authors consider a bi-objective version of the problem for two-tier WMNs. Vanhatupa et al. [14] considered Genetic Algorithm approaches for optimizing node placement and configuration for WLAN planning. Chen et al. [3] considered the case of urban wireless mesh network planning for the case of directional antennas.

Unfortunately, node placement problems are shown to be computationally hard to solve to optimality [6, 10, 2, 17], and therefore heuristic and meta-heuristic approaches are used to cope with them in practice. Heuristic methods distinguish for achieving near-optimal solutions in reasonable time. It should be noted that even though such solutions could be local optima, they suffice for most practical situations when facility locations must be computed before deployment of the system in a real setting.

In this work we consider the version of the problem that given an area where to distribute a number of mesh router nodes and a number of mesh client nodes of fixed positions (of an arbitrary distribution), the objective is to find a location assignment for the mesh routers that maximizes both the network connectivity[1] and client coverage. These two objectives are among most important objectives in WMNs. Both of them are related to the performance of the network; the later can be also seen as a QoS in WMNs. We propose and evaluate a Simulated Annealing (SA) approach for solving mesh router node placement problem. The optimization approach follows a hierarchical setting in which the primary objective is that of maximizing the size of the giant component while user coverage is considered secondary one. In such setting, the SA algorithm tries to first maximize the size of the giant component and then tries to maximize the user coverage without worsening the size of the giant component.

SA is a local search based method that distinguishes for its efficiency in reaching faster near-optimal solutions. Different from simple local search methods such as Hill climbing, SA accepts neighboring solutions which could be worse than current solution, in an attempt to escape from local optima. Research in heuristic methods has shown that SA is more effective than simple local search and can find high quality solutions if an effective cooling strategy is employed. Moreover, SA has shown to be effective in finding the optimal solution for both discrete and continuous optimization problems.

We have experimentally evaluated the proposed SA algorithm through a benchmark of generated instances, consisting of 48 instances, ranging from small to large size in terms of mesh router nodes and the grid area. Moreover, instances are generated using different distributions of mesh clients (Uniform, Normal, Exponential and Weibull). In the experimental study we evaluated the effectiveness of different local movements, namely, *Random*, *Radius*, *Swap* and *Combination*, in terms of the maximization of the size of the giant component and user coverage.

The rest of the paper is organized as follows. In Section 2 we present the definition of the mesh router nodes placement problem in WMNs. The SA approach and its application to mesh router nodes placement problem is presented in Section 3. The experimental evaluation is given in Section 4. We end the paper in Section 5 with some conclusions.

2. Problem statement

In a general setting, location models in the literature (see e.g. [7]) have been defined as follows:

[1]Network connectivity is measured by the size of the giant component.

- a universe U, from which a set C of client input positions is selected;
- an integer, $N \geq 1$, denoting the number of facilities to be deployed;
- one or more metrics of the type $d : U \times U \rightarrow R_+$, which measure the quality of the location, and
- an optimization model that takes in input the universe where facilities are to be deployed, a set of client positions and returns a set of positions for facilities that optimize the considered metrics.

It should be noted that different models can be established depending on whether the universe is considered:

(a) *continuous* (universe is a region, where clients and facilities may be placed anywhere within the continuum leading to an uncountably infinite number of possible locations);

(b) *discrete* (universe is a discrete set of predefined positions); and,

(c) *network* (universe is given by an undirected weighted graph; in the graph, client positions are given by the vertices and facilities may be located anywhere on the graph).

We consider the version of the mesh node placement problem corresponding to the network space model above (see [4] –Chapter 3 in [7]). Thus, in this version, we are given a 2D area where to distribute a number of mesh router nodes and a number of mesh client nodes of fixed positions (of an arbitrary distribution) and finds a location assignment for the mesh routers that maximizes the network connectivity (size of the giant component) and client coverage. An instance of the problem consists of:

- N mesh router nodes, each having its own radio coverage.
- An area $W \times H$ where to distribute N mesh routers. Positions of mesh routers are not pre-determined. The area is divided in square cells of an *a priori* fixed length and mesh router nodes are to be deployed in the cells of the grid area.
- M client mesh nodes located in arbitrary cells of the considered grid area.

An instance of the problem can be formalized by an adjacency matrix of the WMN graph, whose nodes are of two types: router nodes and client nodes and whose edges are links in the mesh network (there is a link between a mesh router and mesh client if the client is within radio coverage of the router). It should be noticed that here the deployment area is arranged in grid cells, representing graph nodes, where we can locate mesh nodes. In fact, in a cell, both a mesh and a client node can be placed.

2.1. Optimization setting

The objective is to place mesh router nodes in cells of considered area to maximize network connectivity and user coverage. For optimization problems having two or more objective functions, two settings are usually considered: the hierarchical and simultaneous optimization. In the former, the objectives are classified (sorted) according to their priority. Thus, for the two objective cases, one of the objectives, say f_1, is considered as primary objective and the other, say f_2, as secondary one. In the former, we try to optimize f_1, and then, we try to optimize f_2 without worsening the best value of $f1$. In the later approach, both objectives are optimized simultaneously.

In this work we have considered the hierarchical approach[2] in which the size of the giant component is a primary objective and the user coverage is a secondary one. Thus, the local search algorithm will first maximize the size of the giant component through local perturbations; next, when no further improvements are possible, the algorithm will try to maximize the user coverage without worsening the size of the giant component.

2.2. Client mesh nodes distributions

It should be noted from the above problem description that mesh client nodes can be arbitrarily situated in the given area. For evaluation purposes, it is interesting, however, to consider concrete distributions of clients. For instance, it has been shown from studies in real urban areas or university campuses that users (client mesh nodes) tend to cluster to hotspots. Therefore, different client mesh nodes distributions should be considered, for instance Weibull distribution, in evaluating WMN metrics. We have considered Uniform, Normal, Exponential and Weibull distributions for client mesh nodes in the experimental evaluation (see Section 4).

[2]In the bi-objective case, this model is also known as bi-level optimization approach.

3. Simulated Annealing

Simulated Annealing (SA) algorithm (proposed by S. Kirkpatrick et al. [8]) is inspired by the cooling process of metals by which a material is heated and then cooled in a controlled way to increase the size of its crystals and reduce their defects. The heat causes the atoms to leave their initial positions (a local minimum of energy) and move randomly; the slow cooling gives them more likelihood to find configurations with lower energy than the previous one. In each iteration, it considers some neighbors of the current state s, and probabilistically decides between changing the system to the state s' or staying in the state s. The probabilities are chosen so that the system converges towards lower energy states. Typically this step is repeated until the system reaches a state good enough for the application or when a certain number of iterations is performed. The probability of making the transition to the new state s' is a function $P(\delta E, T)$ of the energy difference $\delta E = E(s') - E(s)$ between the two states, and the variable T, called temperature.

An important characteristics of the SA algorithm is that the transition probability P is always non-zero, even when δE is positive, i.e., the system can move to a higher energy state (worse solution) than the current state. This fact allows the method to overcome local optima. So, when the temperature tends to a minimum, the probability tends to zero asymptotically. Thus, every time the algorithm accepts fewer moves to increase the system's energy. If δE is negative, i.e., the transition energy decreases, the movement is accepted with probability $P = 1$. The idea is that as the algorithm progresses through the search space, the temperature decreases according to a particular function (which is usually an exponential function).

3.1. Pseudo-code of basic SA algorithm

SA algorithm is a generalization of the Hill Climbing (HC) heuristic. Indeed, SA consists of a sequence of executions of HC with a progressive decrement of the temperature starting from a an "high" temperature, where almost any move is accepted, to a low temperature, where the search resembles HC. In fact, it can be seen as a hill-climber with an internal mechanism to escape local optima (see pseudo-code in Alg. 1). In SA, the solution s' is accepted as the new current solution if $\delta \leq 0$ holds, where $\delta = f(s') - f(s)$. Additionally, to allow escaping from a local optimum, moves that increase the energy function are accepted with a decreasing probability $\exp(-\delta/T)$ if $\delta > 0$, where T is the temperature parameter (see function *Accept* in

Alg. 1). The decreasing values of T are controlled by a *cooling schedule*, which specifies the temperature values at each stage of the algorithm, what represents an important decision for its application (a typical option is to use a proportional method, like $T_k = \alpha \cdot T_{k-1}$). SA usually gives better results in practice, but tends to be rather slow. The most striking difficulty in applying SA is to choose and tune its parameters such as initial and final temperature, decrement of the temperature (cooling schedule), equilibrium detection, etc.

Algorithm 1 : Pseudo-code of Simulated Annealing.

```
$t$ := 0
Initialize $T$
$s0$ := Initial_Solution()
$v0$ := Evaluate($s0$)
while (stopping condition not met) do
    while $t$ mod MarkovChainLen = 0 do
        $t$ := $t$+1
        $s1$ := Generate($s0$,$T$)    //Move
        $v1$ := Evaluate($s1$)
        if Accept($v0$,$v1$,$T$) then
            $s0$ := $s1$
            $v0$ := $v1$
        end if
    end while
    $T$ := Update($T$)
end while
return $s0$
```

3.2. Particularization of SA for mesh router node placement

We present here the particularization of the SA algorithm for the case of mesh router node placement problem in WMNs.

Initial solution. The algorithms starts by generating an initial solution. Either random or *ad hoc* methods can be used for this purpose (see [15] for implementation of *ad hoc* methods for the problem).

Evaluation of fitness function. An important aspect is the determination of an appropriate adaptive function or objective function and its encoding.

Ideally, we would construct objective functions with certain regularities, i.e. objective functions for which it holds that for any two solutions that are close in the search space, their respective fitness values in the objective functions are similar. One issue to consider here is that if the fitness function has not been correctly coded, there can appear many local optima in search space, which could prevent the algorithm from progressing towards desired solutions.

As stated earlier, we tackle a bi-criteria optimization problem in which the fitness function follows a hierarchical approach. The main objective is to maximize the size of giant component in WMN while the number of covered users is considered a secondary objective. In this way, we prioritize the connection between the routers in order to ensure network connectivity and at the same time trying to achieve the largest number of users covered.

Neighbor selection and movement types. The neighborhood structure is essential in the design of SA. On the one hand, the neighborhood structure, and therefore the movement type from one solution to a neighboring solution, determines the time efficiency of the algorithm as a considerable part of the running time of SA is spent to explore the neighborhood. In this sense, it is desirable to define several movement types and study the performance of SA using them. On the other hand, the movement type is based on the combinatorial structure of the solution and therefore some movements could yield to better quality solutions due to the ability to efficiently explore the solution space.

Given a solution s, its neighborhood $N(s)$ consists of all solutions in the search space that are reachable by a local move from the current solution s. In the implementation of SA, we have considered four different types of movements. The first, called *Random*, consists in choosing a router at random in the grid area and placing it in a new position at random. The second move, called *Radius*, chooses the router of the largest radio and places it at the center of the most densely populated part (in terms of client mesh nodes) of the deployment area. The third move, called *Swap*, consists in swapping two routers: the one of the lowest radio router in the most densely populated area with that of largest radio in the sparsest populated area. The idea is that largest radio routers should serve to more clients and thus should be placed in more dense areas. Finally, we also considered the possibility to combine the above movements in sequences of movements. The idea is to see if the combination of these movements offers some improvement over stand alone movements. We called this type of movement *Combination.*

Acceptability criteria. The acceptability criteria for newly generated solutions is based on the definition of a threshold value (accepting threshold) as follows. We consider a succession t_k such that $t_k > t_{k+1}$, $t_k > 0$ and $t_k \to 0$ as $k \to \infty$. Then, for any two solutions s_i and s_j, if $fitness(s_j) - fitness(s_i) < t_k$, then solution s_j is accepted. In other terms, all solutions that reduce the cost of the current solution are accepted; those that would increase the cost are accepted on a limited basis. With increasing values of k (as the algorithm progresses) only small increments are accepted, until eventually only accepted improvements occur.

For the SA, t_k values are taken as accepting threshold but the criterion for acceptance is probabilistic:

- If $fitness(s_j) - fitness(s_i) \leq 0$ then s_j is accepted.
- If $fitness(s_j) - fitness(s_i) > 0$ then s_j is accepted with probability $\exp[(fitness(s_j) - fitness(s_i))/t_k]$ (at iteration k the algorithm generates a random number $R \in (0,1)$ and s_j is accepted if $R < \exp[(fitness(s_j) - fitness(s_i))/t_k]$).

It should be noted that each neighbor has a positive probability of replacing the current solution. The t_k values are chosen in way that solutions that most worsen the cost of the solutions are less likely to be accepted (but there is still a positive probability of accepting them).

4. Experimental study

4.1. Parameter set up

One of the main issues in using heuristic approaches is to adequately set up the parameter values, which have a direct impact on the performance of the algorithm. In our case, there are two groups of parameters to set up: those related to SA algorithm itself, such as the temperature, and those corresponding to the problem under study, such as number of routers to deploy, number of costumers to cover, grid area sizes, etc.

Further, the values of these parameters should be set up in a way that no biased results will be obtained when empirically evaluating the implementation. To this end, randomly generated instances of three different grid area sizes (32×32, 64×64, 128×128, respectively) are used. Then, the resulting setting of parameter is used for obtaining computational results for a benchmark of instances.

One important aspect of the tuning process is to study the performance of SA under different movements. As stated earlier (see Section 3), the movement type has a considerable effect on the performance of SA. It is to expect that there is no movement type that would work best for all instance sizes, therefore, it is necessary to identify which movement type works best for which instance size.

Instances of 32×32 grid area size. For the evaluation of performance of these instances, we obtained the following setting of parameters: $temperature = 3$, $nb\_independent\_runs = 15$ and $nb\_iterations\_per\_phase = 60$. In the instances, the client positions were generated following a normal distribution $N(\mu = 16, \sigma = 32/10)$, and 16 routers were to be placed in the 32×32 grid area to cover 48 clients. It should be noted that we conducted 15 independent runs in order to avoid fortuitous results; averaged results are then used.

The averaged results of 15 independent runs showed that the movement *Radius* achieved the best improvements in the size of giant component. We also observed that the *Combination* of movements showed a good performance. The *Random* move showed the worst performance, actually it gets stagnated around 16th phase of the search process. Regarding the *Swap* movement, it could be the case that this movement is too computationally expensive for small size instances and doesn't outperform other movements. It should be noted however that for small size instances, all movements (except *Random*) performed well after a considerable number of search phases (about 30 phases). We show graphically the performance of the local movements in Fig. 1.

Instances of 64×64 grid area size. The parameter values $temperature = 2$, $nb\_independent\_runs = 15$ and $nb\_iterations\_per\_phase = 150$ were obtained. In the instances, the client positions were generated following a normal distribution $N(\mu = 32, \sigma = 64/10)$, and 32 routers were to be placed in the 64×64 grid area to cover 96 clients.

As can be seen from Fig. 2, at the beginning, *Radius* showed a very good performance (up to phase 20) but with the increasing number of phases, *Swap* is able to achieve better quality solutions (about from phase 25 onwards).

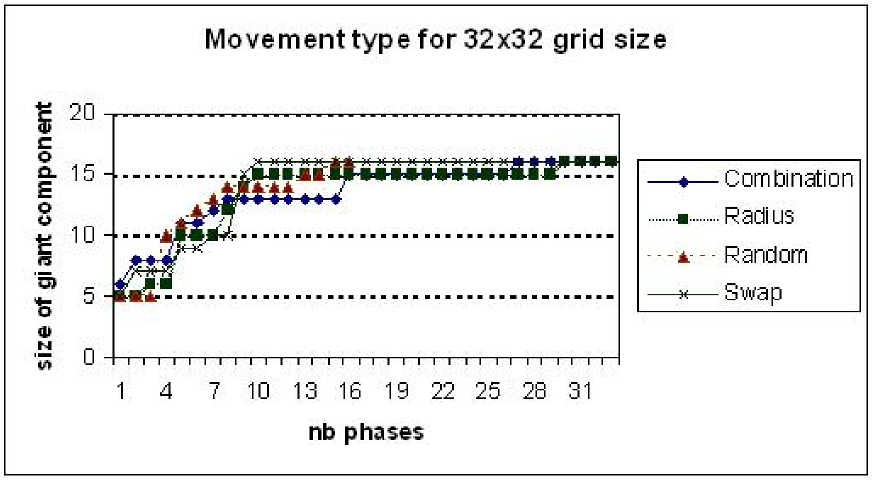

Figure 1: Performance of local movements in SA algorithm for 32×32 grid area where 16 routers were to be placed and give coverage to 48 clients.

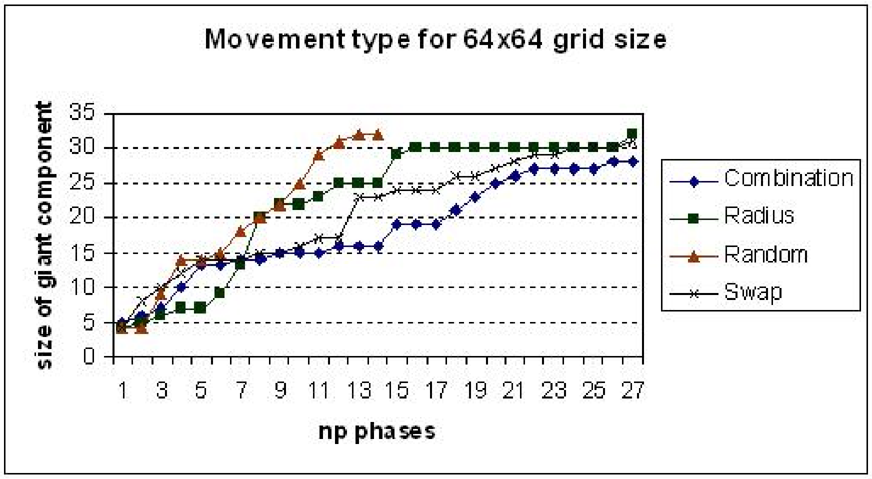

Figure 2: Performance of local movements in SA algorithm for 64×64 grid area where 32 routers were to be placed and give coverage to 96 clients.

Instances of 128×128 grid area size. Values of parameters $temperature = 2$ $nb\_independent\_runs = 15$ and $nb\_iterations\_per\_phase = 300$ were obtained. In the instances, the client positions were generated following a normal distribution $N(\mu = 64, \sigma = 128/10)$, and 64 routers were to be placed in the 128×128 grid area to cover 192 clients.

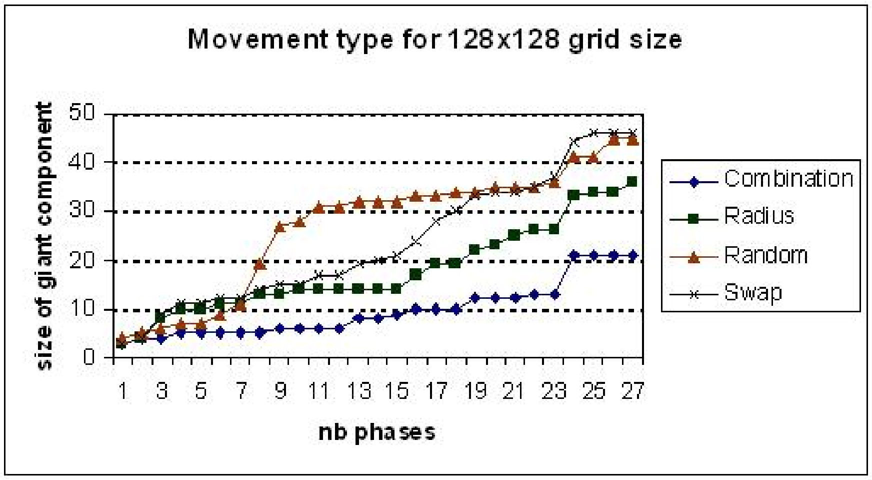

Figure 3: Performance of local movements in SA algorithm for 128×128 grid area where 64 routers were to be placed and give coverage to 192 clients.

For this size, as can be seen from Fig. 3, the experimental results confirm our expectations that *Swap* is more effective than the rest of movements (roughly from phase 23 onward). Surprisingly, *Random* movement also offers good results but has a slower convergence. Note that for instances of this size it doesn't seem worth to use *Radius* due to the diverse density of client mesh nodes in the area.

4.2. Benchmark of instances

We have generated a benchmark consisting of 48 instances, having different sizes of grid area and using four probability distributions for the positions of mesh client nodes in the grid area. These instances aim to represent realistic-size instances[3].

[3]In the literature, instances having up to 60 mesh devices are considered realistic-size instances.

Instances are arranged in three groups, each having 16 instances and are labelled $I_{x\times x\_D\_k}$, where:

- x stands for the height and width of the grid area, that is, the number of cells of arbitrary edge length; it takes values 32, 64 and 128.
- D stands for the distribution of the client mesh routers in the grid area; four distributions are considered: Uniform (U), Normal (N), Exponential (E) and Weibull (W).
- k is the index of the instance.

Thus, we have 16 instances for each grid size (32, 64 and 128, respectively) and within each group we have 4 instances for each distribution (Uniform, Normal, Exponential and Weibull, resp). For instance, in this notation, $I_{64\times 64\_N\_3}$ denotes the third instance of a 64×64 grid area, with mesh clients nodes positions generated using Normal distribution. Regarding the number of mesh routers to deploy and number of mesh clients to cover, instances of 32×32 grid area consist of 16 mesh routers nodes and 48 client mesh nodes; instances of 64×64 grid area consist of 32 mesh routers nodes and 96 client mesh nodes; and, instances of 128×128 grid area consist of 64 mesh routers nodes and 192 client mesh nodes.

4.3. Simulated Annealing results for the benchmark

Once the fine tuning of parameters was done, we measured the performance of the SA algorithm for instances of the benchmark.

Computational results for instances of size 32×32 grid area. We give in Table 1 computational results for instances of benchmark of 32×32 grid area using movement type *Radius*, which showed to performed better for this size of instances. As can be seen from Table 1, the SA algorithm achieved to establish a network of all routers connected and almost all clients are covered for all but uniform distribution. Moreover the small deviation showed the robustness of the implementation, in that, the algorithm is always able to deliver good solutions.

Table 1: Size of giant component and user coverage for 32×32 grid size instances, 16 routers nodes and 48 clients.

Instance	Size of giant component			#Users covered		
	best	avg	dev	best	avg	dev
I32x32_U_1	13	10	0.3	23	22	0.1
I32x32_U_2	13	8	0.5	24	23	0.1
I32x32_U_3	12	8	0.4	23	22	0.1
I32x32_U_4	11	8	0.3	26	24	0.2
I32x32_N_1	16	15	0.1	43	40	0.3
I32x32_N_2	16	15	0.1	42	39	0.3
I32x32_N_3	16	15	0.1	45	40	0.5
I32x32_N_4	16	15	0.1	43	40	0.3
I32x32_E_1	16	13	0.3	45	36	0.9
I32x32_E_2	16	14	0.2	46	39	0.7
I32x32_E_3	16	11	0.5	45	36	0.9
I32x32_E_4	16	13	0.3	45	41	0.4
I32x32_W_1	16	11	0.5	32	30	0.2
I32x32_W_2	16	14	0.2	43	36	0.7
I32x32_W_3	16	12	0.4	44	34	1
I32x32_W_4	16	13	0.3	43	35	0.8

Computational results for instances of size 64×64 grid area. We give in Table 2 computational results for instances of benchmark of 64×64 grid area using movement type *Swap*, which showed to performed better for this size of instances. As can be seen from Table 2, the SA algorithm followed the same trend of results; it was possible to establish a network of almost all routers connected and almost all clients covered for all but uniform distribution.

Table 2: Size of giant component and user coverage for 64×64 grid size instances, 32 routers nodes and 96 clients.

Instance	Size of giant component			#Users covered		
	best	avg	dev	best	avg	dev
I64x64_U_1	15	9	0.6	46	43	0.3
I64x64_U_2	12	8	0.4	41	41	0
I64x64_U_3	14	8	0.6	45	39	0.6
I64x64_U_4	13	9	0.4	44	40	0.4
I64x64_N_1	32	28	0.4	71	69	0.2
I64x64_N_2	32	26	0.6	70	65	0.5
I64x64_N_3	32	26	0.6	74	69	0.5
I64x64_N_4	31	28	0.3	74	73	0.1
I64x64_E_1	28	20	0.8	72	69	0.3
I64x64_E_2	26	19	0.7	88	74	1.4
I64x64_E_3	25	20	0.5	64	54	1
I64x64_E_4	24	19	0.5	69	67	0.2
I64x64_W_1	27	22	0.5	82	70	1.2
I64x64_W_2	30	22	0.8	65	64	0.1
I64x64_W_3	29	22	0.7	65	50	1.5
I64x64_W_4	26	21	0.5	89	67	2.2

Computational results for instances of size 128×128 grid area. We give in Table 3 computational results for instances of benchmark of 128×128 grid area using movement type *Swap*, which showed to performed better for this size of instances. As can be seen from Table 3, the SA algorithm performed very well for all but uniform distribution of clients in the grid area.

4.4. Analysis of the results

The SA algorithm showed a good performance for all instances of the benchmark. The algorithm performed better the Hill Climbing algorithm

Table 3: Size of giant component and user coverage for 128×128 grid size instances, 64 routers nodes and 192 clients.

Instance	Size of giant component			#Users covered		
	best	avg	dev	best	avg	dev
I128x128_U_1	10	7	0.3	75	75	0
I128x128_U_2	13	7	0.6	77	71	0.6
I128x128_U_3	16	8	0.8	72	69	0.3
I128x128_U_4	11	7	0.4	73	73	0
I128x128_N_1	44	27	1.7	125	115	1
I128x128_N_2	41	25	1.6	121	115	0.6
I128x128_N_3	43	28	1.5	122	120	0.2
I128x128_N_4	46	26	2	119	116	0.3
I128x128_E_1	30	21	0.9	137	127	1
I128x128_E_2	32	20	1.2	144	133	1.1
I128x128_E_3	38	26	1.2	134	125	0.9
I128x128_E_4	32	25	0.7	162	144	1.8
I128x128_W_1	40	30	1	138	133	0.5
I128x128_W_2	41	26	1.5	138	137	0.1
I128x128_W_3	41	28	1.3	135	125	1
I128x128_W_4	41	32	0.9	131	118	1.3

for the problem [15], which is due to its mechanism of accepting with certain probability also worse solutions than current solution in an attempt to escape from local optima. The results are more striking in the case of large size 128×128 grid area instances, for which SA clearly outperformed the Hill Climbing algorithm.

From the results we could conclude that there were no a single neighborhood structure that performed best in all cases; rather, for small size instances *Radius* movement was adequate for exploring the neighborhood while for medium and large size it was the *Swap* movement. In addition, the results showed a very good performance of *Combination* movement (which is the sequential combination of the other three movements) for instances of small size.

Finally, the experimental study revealed also that it is necessary to evaluate the performance of the placement node algorithms, such as SA algorithm considered in this work, against different distributions of clients in the grid area. In particular, the Weibull distribution which is based on the idea of hot spots, seems interesting to further explore for distributions of client nodes in the grid area.

5. Conclusions

In this work we have presented an implementation of Simulated Annealing (SA) algorithm for near-optimally solving the problem of placement of mesh router nodes in Wireless Mesh Networks (WMNs). In this version of the problem, a number of client mesh nodes are *a priori* distributed in a grid area, arranged in small cells, and a number of mesh router nodes are to be deployed in the area. We have considered the optimization model in which the objective is two-fold: to maximize the network connectivity (through the maximization of the size of the giant component) and user coverage. In this model, the former objective is considered as primary while the later is considered secondary, that is, the algorithm tries to optimize first the size of giant component and then tries to maximize the number of clients covered without worsening the size of the giant component.

The results of experimental study showed that an efficient implementation of SA requires the definition of different neighborhood structures in order to find the appropriate structure according to different instance size as well as to different distributions of clients in the grid area. The results confirmed that SA is an effective resolution method for the problem as it achieved to establish network connectivity of almost all mesh router nodes and covered almost all client mesh nodes for all considered benchmark instances.

Acknowledgment

Fatos Xhafa's research work partially done at Birkbeck, University of London, on Leave from Technical University of Catalonia (Barcelona, Spain). His research is supported by a grant from the General Secretariat of Universities of the Ministry of Education, Spain.

References

[1] I. F. Akyildiz, X. Wang, and W. Wang. Wireless mesh networks: a survey. *Computer Networks* **47**(4) (2005) 445-487.

[2] E. Amaldi, A. Capone, M. Cesana, I. Filippini, F. Malucelli. Optimization models and methods for planning wireless mesh networks. *Computer Networks* **52** (2008) 2159-2171.

[3] Ch. Chen and Ch. Chekuri. Urban Wireless Mesh Network Planning: The Case of Directional Antennas. *Tech Report* No. UIUCDCS-R-2007-2874, Department of Computer Science, University of Illinois at Urbana-Champaign (2007).

[4] J. Current, M. Daskin, and D. Schilling. Discrete Network Location Models. Chapter 3 in H.W. Hamacher, Z. Drezner. *Facility Location: Applications and Theory.* Springer Berlin Heidelberg, 2001.

[5] A. Franklin and C. Siva Ram Murthy. Node Placement Algorithm for Deployment of Two-Tier Wireless Mesh Networks. In *Proceedings of IEEE GLOBECOM 2007, IEEE Global Communications Conference* (2007) 4823-4827.

[6] M.R. Garey and D.S. Johnson. *Computers and Intractability –A Guide to the Theory of NP-Completeness.* Freeman, San Francisco (1979).

[7] H.W. Hamacher, Z. Drezner (Eds). *Facility Location: Applications and Theory.* Springer Berlin Heidelberg, 2001

[8] S. Kirkpatrik and C. D. Gelatt and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, **220** (1983) 671-680.

[9] F. Li, Y. Wang, X. Li, A. Nusairat, Y. Wu. Gateway Placement for Throughput Optimization in Wireless Mesh Networks. *Mobile Netw Appl* **13** (2008) 198-211.

[10] A. Lim, B. Rodrigues, F. Wang and Zh. Xua. $k-$Center problems with minimum coverage. *Theoretical Computer Science* **332** (2005) 1-17.

[11] S. N. Muthaiah and C. Rosenberg. Single Gateway Placement in Wireless Mesh Networks. In *Proceedings of 8th International IEEE Symposium on Computer Networks*, Turkey (2008).

[12] N. Nandiraju, D. Nandiraju, L. Santhanama, B. He, J. Wang, and D. Agrawal. Wireless mesh networks: Current challenges and future direction of web-in-the-sky. *IEEE Wireless Communications* (2007) 79-89.

[13] M. Tang. Gateways Placement in Backbone Wireless Mesh Networks. *International Journal of Communications, Network and System Sciences*, **1** (2009) 44-50.

[14] T. Vanhatupa, M. Hännikäinen and T.D. Hämäläinen. Genetic Algorithm to Optimize Node Placement and Configuration for WLAN Planning. In *Proceedings of 4th International Symposium on Wireless Communication Systems* (2007) 612-616.

[15] F. Xhafa, Ch. Sanchez, L. Barolli. Ad Hoc and Neighborhood Search Methods for Placement of Mesh Routers in Wireless Mesh Networks. *The 11-th International Workshop on Multimedia Network Systems and Applications (MNSA-2009)*, held in conjunction with the IEEE 29-th International Conference on Distributed Computing Systems (ICDCS-2009), in June 22-26, 2009, Montreal, Quebec, Canada. In Proceedings of ICDCS Workshops (2009) 400-405.

[16] P. Zhou, B. S. Manoj, and R. Rao. A gateway placement algorithm in wireless mesh networks. In *Proceedings of the 3rd International Conference on Wireless Internet* (2007) 1-9.

[17] J. Wang, B. Xie, K. Cai and D.P. Agrawal. Efficient Mesh Router Placement in Wireless Mesh Networks, MASS, *IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems* (2007) 1-9.