Solving Mesh Router Nodes Placement Problem in Wireless Mesh Networks by Tabu Search Algorithm

Fatos Xhafa[a,*], Christian Sánchez[a], Admir Barolli[b], Makoto Takizawa[b]

[a] *Technical University of Catalonia, Spain*
[b] *Hosei University, Japan*

Abstract

Wireless Mesh Networks (WMNs) are an important networking paradigm that offer cost effective Internet connectivity. The performance and operability of WMNs depend, among other factors, on the placement of network nodes in the area. Among the most important objectives in designing a WMN is the formation of a mesh backbone to achieve high user coverage. Given a number of router nodes to deploy, a deployment area and positions of client nodes in the area, an optimization problem can be formulated aiming to find the placement of router nodes so as to maximize network connectivity and user coverage. This optimization problem belongs to facility location problems, which are computationally hard to solve to optimality. In this paper we present the implementation and evaluation of Tabu Search (TS) for the problem of mesh router node placement in WMNs. The experimental evaluation showed the efficiency of TS in solving a benchmark of instances.

Keywords:
Tabu Search, Optimization, Mesh Router Node, Wireless Mesh Networks, Node Placement, Size of Giant Component, User Coverage, *QoS*.

1. Introduction

Wireless Mesh Network (WMN) [1] is an important paradigm in today's networking infrastructures. WMNs can be used either as stand alone net-

*Corresponding author
Email addresses: fatos@lsi.upc.edu (Fatos Xhafa), csanchez@lsi.upc.edu (Christian Sánchez), admir.barolli@gmail.com (Admir Barolli), makoto.takizawa@computer.org (Makoto Takizawa)

works or as extensions of wired networks to provide Internet connectivity to mesh client nodes through mesh router nodes and gateways. Their low cost of deployment and maintenance make WMNs very attractive technology for a large range of applications such as for urban areas, community networking, metropolitan area networks, municipal networks, corporative networks, medical systems, transport systems, surveillance systems, enterprise and campus applications, etc. [6]. In general, WMNs can provide Internet connectivity to users of a certain geographical area. The main feature of WMN is its mesh topology and multi-hop communication through a mesh topology, in which every node (e.g. mesh router) is connected to one or more nodes, enabling thus the information transmission in more than one path. The path redundancy is a robust feature of this kind of topology. These characteristics of networks with mesh topology make them very reliable and robust networks to potential node failure. However, as in other types of wireless networks, connectivity and stability are critical factors to the performance of WMNs.

Internet connectivity of mesh client nodes is achieved through mesh routers and gateways, therefore the placement of such nodes is crucial to achieve optimized performance of the network. Assuming that each mesh router can have its own radio connectivity coverage, a mesh router/client node is connected to a mesh router node if it is within its radio coverage. Therefore, the connectivity of the mesh router nodes and the user coverage depend on the placement of mesh router nodes. It is for this reason that one needs to formulate and solve some optimization problems to achieve desired performance in WMNs. The efficient resolution of such problems, especially those related to node placement, is useful during the design phase of the WMNs. On the one hand, one can ask how many mesh router nodes have to be deployed in the given geographical area and, on the other, where to place the mesh router nodes in the grid area. The former is directly related to minimizing the cost of the deployment of the WMN while the later aims at achieving maximized performance of WMN, for a given number of mesh router nodes.

The design of WMNs, however, is not an easy task and could be challenging if many objectives are to be met under several restrictions. Indeed, the network performance and operability depend, among other factors, on the placement of network nodes (mesh routers, gateways, ...) in the geographical area. Different kinds of restrictions could be formulated for the problem. For instance, there could be topological restrictions on the deployment area, client nodes could be distributed arbitrarily on the area without following any pre-established pattern, and router nodes can have different radio cover-

age. Additionally, there could be investment cost restrictions. Whatever the restrictions could be, the most important objective in designing a WMN is to establish a mesh backbone to achieve high user coverage. In this work, we consider WMNs in which mesh client nodes are fixed nodes whose positions are *a priori* known. The applicability of this kind of WMN arises in many real life scenarios such as in urban areas (neighbourhoods and communities), enterprise and campus areas, etc. Furthermore, we assume as input a given number of mesh router nodes (which could have their own radio coverage). It should be noted that a more generalized version of the problem would be to minimize the number of mesh router nodes to deploy to achieve full connectivity and full client coverage. Nevertheless, even if the network designer would know the number of mesh routers to deploy, that's not enough: the locations of routers to deploy should be computed as well. In fact, while not at optimality, the number of routers to deploy can be computed based on estimation. Therefore, the computation of the router locations is more important and challenging here.

It is commonplace to assist decision making at network design process through optimization. In the case of WMNs, given a number of router nodes to deploy, a 2D deployment area and positions of client nodes in the area, an optimization problem can be formulated aiming to find the placement of router nodes in the deployment area so as to maximize network connectivity and user coverage. The former can be measured by the size of the giant component (i.e. the largest connected component) in the mesh router nodes network while the user coverage accounts the number of client nodes that fall within the radio coverage of routers. Naturally, the largest the size of the giant component, the largest is the mesh backbone (mesh routers connected to mesh clients).

An optimization problem that can be taken as a reference here is that of facility locations, which in a general setting seeks an optimal placement of a number of facilities that give service to a certain number of clients. The similarity with mesh routers node placement in WMNs is straightforward by considering mesh routers as facilities, which give service, i.e. Internet connectivity, to mesh clients nodes. The problem of facility placements has been long investigated in optimization and logistics domains. Facility location problems are known to be computationally hard to solve to optimality in their general formulations and thus heuristic methods are the alternative to cope in practice with their resolution.

In this paper we present the implementation and evaluation of Tabu

Search (TS) for solving the mesh router nodes placement considered as bi-objective optimization problem (maximizing the size of the giant component and maximizing the user coverage.) Unlike other local search methods, TS is known for its adaptive search mechanisms aiming to overcome getting stuck into local optima. We implemented several forms of adaptive memory (short and long term) as well as tabu solution tagging as effective mechanisms to avoid cycling among visited solutions in the search space. Also several types of neighbourhoods are considered by defining different movement types. We evaluated the TS method through a set of instances for the problem. Instances of small to large size were generated using different distributions of mesh client nodes (uniform, normal, exponential and Weibull).

The rest of the paper is organized as follows. In Section 2 we consider the node placement problems in a general setting and show the formulation of the mesh router nodes as a special case of node placement problem. In Section 3, we overview the main features of the TS method and present its instantiation for the case of mesh router nodes problem. Some implementation issues are also discussed. We present the experimental evaluation of TS method in Section 4 and end the paper in Section 5 with some conclusions and indications for future work.

2. Mesh Routers Placement in WMNs as a Node Placement Problem

In this section we recall some basic concepts from node/facility placement problems and formulate the mesh router nodes as node placement problem (see also [4]). Node placement problems arise in many applications such as facility location, logistics, services, etc. In such problems, we are given a number of potential facilities to serve to costumers connected to facilities. The aim is to find locations of facilities such that the cost of serving to all customers is optimized for some metric function(s). In traditional versions of the problem, facilities could be hospitals, polling centres, fire stations serving to a number of clients and aiming to optimize some distance function in a metric space between clients and facilities. One classical version of the problem is that of $p-$median problem, defined as follows: Given a set $\mathcal{F}$ of m potential facilities, a set $\mathcal{U}$ of n users, a distance function $d : \mathcal{U} \rightarrow \mathcal{F}$, and a constant $p \leq m$, determine which p facilities to open so as to minimize a metric, for instance the sum of the distances from each user to its closest open facility.

Facility location problems are showing their usefulness to communication networks, and more especially to WMNs [1, 12, 6] where facilities could be servers, routers, etc., offering connectivity services to clients. In WMNs mesh routers provide network connectivity services to mesh client nodes. In a general setting, location models in the literature have been defined as follows:

(a) a universe $\mathcal{U}$, from which a set $\mathcal{C}$ of client input positions is selected;

(b) an integer, $N \geq 1$, denoting the number of facilities to be deployed;

(c) one or more metrics $d : \mathcal{U} \times U \rightarrow R_+$, which measure the quality of the location; and,

(d) an optimization model.

The optimization model 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 metric(s).

It is possible to establish different models 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).

Node placement problems in their general formulations are shown to be computationally hard to solve to optimality [8, 10, 2, 18], and therefore heuristic and meta-heuristic approaches are useful resolution approaches for practical purposes.

2.1. Mesh router nodes placement

Mesh router nodes placement can be see as a family of multi-objective optimization problems. Different versions of the problem can be formulated based on the objectives to optimize and a set of constraints, such as topological restrictions, battery restrictions, *QoS* requirements, among others [18]. The presence of many objectives such as minimizing the number of mesh routers, maximizing network connectivity, maximizing user coverage, maximizing throughput, minimizing deployment cost, etc., is in fact a main challenge.

Informally, we are given a 2D area where to distribute a number of mesh router nodes and a number of *stationary* mesh client nodes (see Fig. 1). Positions of the mesh client nodes can follow any arbitrary distribution. The objective is to find a location assignment for the mesh routers that maximizes network connectivity (size of the giant component) and client coverage. An instance of the problem consists thus of the following, according to the location model (a)-(d) above:

(a) The set of positions of mesh clients is selected from a 2D area. Every position is specified by the coordinates in 2D plane.

(b) an integer N denoting the number mesh router nodes to be deployed, each having its own radio coverage. An area $W \times H$ where to distribute N mesh routers (positions of mesh routers are to be computed). The 2D area is divided in square cells of *a priori* fixed length and mesh router nodes are to be deployed in the cells of the grid area. An integer M denoting client mesh nodes already located in arbitrary cells of the considered grid area.

(c) The metrics to measure the quality of location are: (1) the size of the giant components (that is, the size of the largest connected component in the resulting graph) and (2) the number of mesh clients covered (that is, the number of mesh clients that are connected to some mesh router, i.e. fall within the radio of some mesh router).

(d) The optimization model is that of a bi-objective optimization model (see below).

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). Each mesh node in the graph is a triple $v =< x, y, r >$ representing the 2D location point and r is the radius of the transmission range. There is an arc between two nodes u and v, if v is within the radio coverage area of u. The deployment area is partitioned by grid cells, representing graph nodes, where we can locate mesh router nodes. In fact, in a cell, both a mesh router and a mesh client node can be placed (see also Fig. 2).

The objective is to place mesh router nodes in cells of considered area to maximize network connectivity and user coverage. Network connectivity

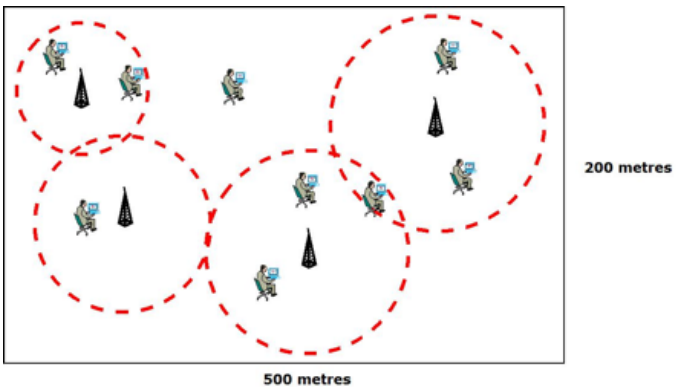

Figure 1: An example of deployment area of $500 \times 200m$ (four routers and nine clients).

and user coverage are among most important metrics in WMNs. The former measures the degree of connectivity of the mesh nodes while the later refers to the number of mesh client nodes connected to the WMN. Both objectives are crucial to WMN and directly affect the network performance; nonetheless, network connectivity is considered here as more important than user coverage.

2.2. Bi-objective optimization model

For optimization problems having two or more objective functions, two models are usually considered: the hierarchical and simultaneous optimization. In the former model, the objectives are classified according to their priority. Thus, for a problem having two objectives sorted as $f_1 \succ f_2$ means that f_1 is the primary objective and f_2 is secondary objective. The algorithm would first optimize according to f_1 until no further improvements are possible. Then, the algorithm optimizes according to f_2 subject to not worsening the value achieved for f_1.

This model is useful when for design or deployment needs, some parameters (objectives) are considered of more priority than others. For instance, network connectivity could be considered a primary objective and the user coverage a secondary one. It should however be noted that the final solution computed by the optimization procedure need not to be optimal and could be far from optimal for the less priority objectives.

In the simultaneous approach, all objectives are simultaneously optimized. Thus, for a problem having two objectives $f_1; f_2$, the optimization procedure tries to optimize at the same time both objectives, which actually

leads to computing the so called *Pareto* front of the optimal solutions. In some cases, it is possible to apply the sum model in which the two objectives are reduced to a single objective function $f = \lambda_1 f_1 + \lambda_2 f_2$, under the conditions $\lambda_1 + \lambda_2 = 1$, $\lambda_1 > 0, \lambda_2 > 0$.

It should be noted that in our hierarchical approach, the optimality is established only with regard to the first objective as we have not considered a proper Pareto front optimization. Due to the hierarchical approach, there is no guarantee on the quality of solutions obtained for the second objective, while the TS tries its optimization as much as possible without worsening the first objective.

An example of solution is graphically shown in Fig. 2. In the figure, router nodes are denoted by **R** and client nodes are denoted by **C**. As can be seen, different routers have different radio coverage and a certain number of clients is covered by each router. However, some clients might eventually not be covered due the problem is not solved to optimality (in the case of hierarchic approach the foremost objective is to establish router connectivity and in case of simultaneous approach there is a trade-off among two objectives). The placement of mesh router nodes (the "*mesh cloud*") is therefore crucial to both establishing full connectivity of mesh routers and full coverage of clients.

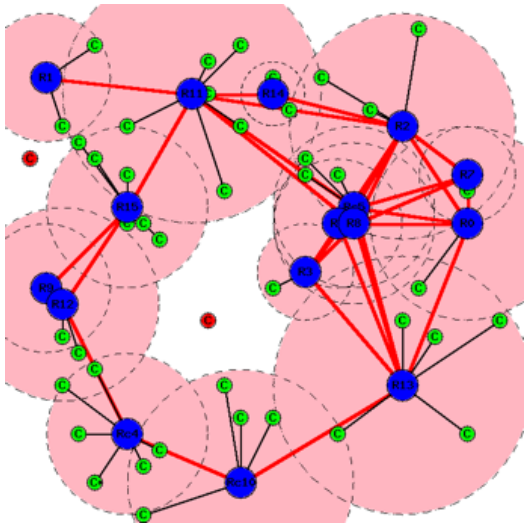

Figure 2: An example of solution: mesh routers network and client coverage.

2.3. Client mesh nodes distributions

An important issue when formulating mesh node placement is whether the client nodes are stationary or mobile nodes. WMN with stationary nodes arise in many real situations, for instance, in a neighbouring community. In stationary nodes case, the positions of the client nodes are *a priori* known although the mesh client nodes can be arbitrarily situated in the given area. In the later case of WMN with mobile nodes, the position of client nodes can change over time. It could as well be considered the case of a WMN where we have both stationary and mobile nodes, for instance, in a neighborhood users inside the homes are stationary and users along the roads are mobile. In both cases, it is interesting, however, to consider concrete distributions of client mesh nodes. The Uniform, Normal, Exponential and Weibull distributions for client mesh nodes can be considered (see Section 4). In particular, the Weibull distribution has shown useful for modelling clustering of mobile users in urban areas or university campuses.

While supporting user mobility is an important and attractive feature of wireless networking, it should be noted that from many empirical studies it has been shown that users do not necessarily move around all the time [5]. In some recent works [3], there has been defined a user's home location as the AP (Access Point) at which a user spends more than 50% of the total time on the network. From this definition, the empirical study in [5] showed that more than 95% of valid users had a home location. Therefore, the mobility feature in WMNs can be in fact approached by assuming different distributions of stationary mesh client nodes, in particular for evaluation purposes.

3. Tabu Search Method

Several heuristic approaches are found in the literature for node placement problems in WMNs [7, 11, 13, 15, 14, 17]. In this section we present an implementation of TS method for the problem of mesh routers nodes placements in WMNs. We start by describing the main features of TS and then their instantiation for the problem under study.

3.1. Main features of Tabu Search

TS method was introduced by Glover [9] as a high-level local search algorithm, specifically to the family of guided search techniques. TS method uses proper mechanisms to guide the search aiming to perform an intelligent exploration of the search space that would eventually allow to avoid getting

trapped into local optima. The objective is thus to remedy one of the main issues of local search methods, namely the useless search in neighbourhood of local optima without further improvements due to re-visiting solutions or paths of solutions already explored. This is achieved by giving the tabu status to solutions visited in the recent search. TS is also designed to be a flexible method, so that the tabu status of solutions can be waived, in case they have been prohibited for a long while or if they satisfy some aspiration criteria. The classification of some solutions as tabu is achieved through the intelligent use of adaptive memory, which evolves over time and eventually can change the status of tabu solutions.

The main features of the TS method are essentially the *adaptive memory* and *responsive exploration*. Again, the adaptive memory is the basis to guide the search in taking intelligent decisions. This gives the TS method advantages with regard to other memoryless methods, being these local search methods (Hill Climbing, Simulated Annealing, etc.) or population based methods (Genetic Algorithms, Memetic Algorithms, etc.). On the other hand, the responsive exploration enables the method to select some solutions which though not so good at the current search iteration could at long run lead to promising areas of good solutions in the search space.

Neighbourhood definition. The definition of the neighbourhood of solutions is fundamental piece of the method. Indeed, it is crucial for implementing the adaptive memory (newly explored neighbour solutions of a solution are given tabu status or are removed the tabu status if aspiration criteria are satisfied) and has an impact on the method's performance (large neighbourhoods would need long exploration time). In fact, the neighbourhood structure is decided by the definition of a "*movement*" that allows to generate a new solution s' from a solution s. Formally, for a solution s, the neighbourhood of s, denoted $\mathcal{N}(s)$, is defined as the set of feasible solutions reachable from s by applying a movement, as follows:

$$\mathcal{N}(s) = \{s' \mid s \xrightarrow{m} s', m \in \mathcal{M}(s), s' \in \mathcal{S}\} \tag{1}$$

where $\mathcal{S}$ is the solution space, $\mathcal{M}(s)$ is the set of movement that can be applied to s. Movements can be seen as *operators* and are usually defined in terms of attributes of solution, based on the combinatorial structure of the solution. The aim is to define operators that when applied to solutions make

small local perturbations yielding to a new solution (which differs little from the original one).

Tabu status. One difference of TS with other local search methods is the use of tabu status to solutions aiming to avoid visiting solutions already found during the search course. Keeping a kind of tabu list is thus useful to break cycling among previously visiting solutions. In fact, for efficiency reasons, instead of giving tabu status to solutions, it is given tabu status to movements that lead to tabu solutions. It should be noted however that the tabu status is a restrictive condition if kept unchanged over solutions for a long time. This could eventually prevent visiting good solutions during the search. It seems thus reasonable to waive the tabu status to movements if they satisfy some additional conditions known as *aspiration criteria*. Therefore, we can define the set of admissible solutions, which are solutions of the neighborhood by excluding tabu ones but including those that have enough level of aspiration:

$$Admissible(s) = \{(\mathcal{N}(s) - \mathcal{T}(s)) \cup Aspiration(s)\} \tag{2}$$

where $\mathcal{T}(s)$ is the set of tabu solutions reachable from s:

$$\mathcal{T}(s) = \{s' | s \xrightarrow{m} s', s \in \mathcal{S}, s' \in \mathcal{S}, m \in \mathcal{M}(s), is\_tabu(s', m) = true\} \tag{3}$$

and $Aspiration(s)$ is the set of tabu movements that satisfy aspiration criteria:

$$\begin{aligned} Aspiration(s) = \{s' | s \xrightarrow{m} s', s \in \mathcal{S}, s' \in \mathcal{S}, m \in \mathcal{M}(s), \\ is\_tabu(s', m) = true, aspirates(s', m) = true\} \end{aligned} \tag{4}$$

It should also be noted from the expression of $Admissible(s)$ (see Eq. (2)) that unlike other local search method, the neighbourhood structure in TS is dynamic, due the set of $Aspiration(s)$ can vary along the iterations of exploring the neighbourhood of s.

3.2. Pseudo-code of Tabu Search method

We have used the Alg. 1 for designing the TS algorithm for mesh router nodes placement in WMNs. As can be seen in the high level setting of Alg. 1, its components should be specified for designing a TS algorithm for a concrete problem under study.

Algorithm 1 Tabu Search Algorithm

begin
Compute an initial solution s;
let $\hat{s} \leftarrow s$;
Reset the tabu and aspiration conditions;
while not termination-condition **do**
    Generate a subset $N^*(s) \subseteq N(s)$ of solutions such that:
        (none of the tabu conditions is violated) or (the aspiration criteria hold)
    Choose the best $s' \in N^*(s)$ with respect to the cost function;
    $s \leftarrow s'$;
    if improvement$(s', \hat{s})$ **then**
        $\hat{s} \leftarrow s'$;
    end if
    Update the recency and frequency;
    if (intensification condition) **then**
        Perform intensification procedure;
    end if
    if (diversification condition) **then**
        Perform diversification procedures;
    end if
end while
return $\hat{s}$;
end;

3.3. Instantiation of TS to mesh router nodes placement in WMNs

We describe next how the main mechanisms of TS methods are instantiated for the problem of mesh router nodes placement.

3.3.1. Neighbourhood exploration

We have implemented the exploration method *steepest descent-mildest ascent*. According to this method, if there is an *admissible* solution in the neighbourhood with better fitness function than the current solution, it is accepted as next solution; otherwise, an admissible solution that yields the least worsening in the fitness value is accepted (see Eq. (2) for the definition of admissible solutions).

Movement. By applying a movement to a solution, a local perturbation is done to the current solution yielding to a new solution in the neighbourhood. The entity movement manages the following information:

- `type`: indicates the type of the movement to apply. It should be noted that the movement type need not to be unique, several types of movements, and thus different neighbourhood structures, can be defined. In our case we have defined two types of movements: *moveToCell* and *swap*. The former selects a mesh router at random and moves it to another cell in the grid. The later selects two mesh routers and swaps their positions in the grid.
- `router_to_moveToCell`: router index to move to another cell.
- `posx_to_moveToCell` and `posy_to_moveToCell`: coordinates of the new position.
- `router1_to_swap` and `router2_to_swap`: the indices of the routers to swap.

3.3.2. Short and long term memory

The historical memory in TS is usually composed by a *short term memory* (or *recency*), with information on recently visited solutions, and a *long term memory* (or *frequency*), storing information gathered along the search.

Short term memory. The tabu list, denoted here TL is implemented as a three dimensional table (of dimensions `number_routers` $\times$ `size_grid_x` $\times$ `size_grid_y`) in which $TL[r][x][y]$ indicates the index of the last iteration in which the assignment of router r to cell $[x, y]$ was done tabu. Additionally, we employ a tabu hash, denoted TH here, which serves to efficiently identify those solutions already visited. The objective of TH is to further filter the already visited solutions and thus to break cycling. This second filtering is needful due the tabu status to movements is waived after a certain number of iterations and also because only some "features" of solutions are deemed tabu (not solutions as a whole). Thus, the mere mechanism of tagging movements as tabu might not suffice to discard already visited solutions. For a solution s, the expression

$$TH[(s.hash\_id) \texttt{ mod } tabu\_size]$$

would indicate if s has already been visited during search course.

Long term memory. On the other hand, for the long term memory, we record the following information:

- `frequency`: a three dimensional table in which `frequency[r][x][y]` records the number of times router r has been assigned to cell $[x, y]$ in the grid.
- `tfrequency`: a vector in which `tfrecuency[r]` indicates how many times the position of router r has been changed.
- `best_sols`: a vector storing the best solutions found so far (by using an `elite_size` parameter).

Tabu status to movements. For efficiency reasons, instead of tagging solutions as tabu, movements are given tabu status. Because movements are defined using attributes (or *features*) of solutions, a solution is tabu if it has some tabu features. In our case, when a router is moved from its original position, via either movement type, its original position is tagged as tabu. According to the movement type, the following update is done to TL:

1. Let m be of type *moveToCell*: $m = moveToCell(r_i, posx_j, posy_k)$ to be applied to solution:

$$s = ((r_0, posx_l, posy_m), \ldots (r_i, posx_n, posy_h), \ldots (r_{n-1}, posx_p, posx_q))$$

at iteration k. Assume that the application of m to s yields to the new solution:

$$s' = ((r_0, posx_l, posy_m), \ldots (r_i, posx_j, posy_k), \ldots (r_{n-1}, posx_p, posx_q))$$

Then, we update $TL[i][n][h] = k$.

2. Let m be a movement of type swap $m = swap(r_i, r_j)$ to be applied to s:

$$s = (\ldots (r_i, posx_a, posy_b) \ldots (r_j, posx_c, posy_d) \ldots)$$

at iteration k. Assume that the application of m to s yields to the new solution:

$$s' = (\ldots (r_i, posx_c, posy_d) \ldots (r_j, posx_a, posy_b) \ldots)$$

Then, we update $TL[i][a][b] = k$ and $TL[j][c][d] = k$.

Additionally, a hash function is used to check further for possible cycling to visited solutions:

$$hash = \sum_{router \in Routers} hashing[router][pos\_router(router)]$$

where $pos\_router(router)$ returns the position $[posx][posy]$ of a router and $hashing[r][posx][posy]$ is an integer from $[1, LARGE\_HASH]$ chosen at random. Then, if the application of m to s yields to a visited solution, the current solution goes back to previous solution, which should be registered in the hash table $TH[(s.hash\_id) \texttt{ mod } tabu\_size] = true$.

Aspiration criteria. The aspiration criteria aim to cancel the tabu status to solutions. In the simplest of cases, the tabu status can be cancelled when the solution has remained tabu for a long time (a sufficiently large number of iterations). The following aspiration criteria have been defined:

- Cancel the tabu status based on improved fitness: $A_1(s) = \{s' : is\_better\_than(s', s) = true\}$

- Cancel tabu status after a sufficient number of iterations: $A_2(s) = A_{moveToCell}(s) \cup A_{swap}(s)$, where:

$$A_{moveToCell}(s) = \{s' : s \xrightarrow{m_{moveToCell}(r_i, posx_a, posy_b)} s', \\ (TL[i][a][b] + aspiration\_value) \leq k\}$$

$$A_{swap}(s) = \{s' : s \xrightarrow{m_{swap}(r_i, r_j)} s', \\ (\max(TL[i][pos\_router(j)], TL[j][pos\_router(i)]) + aspiration\_value) \leq k\}$$

where *aspiration_value* is an input parameter and k the iteration number. Then, if solution s' obtained by applying m to s belongs to A_1 or A_2, its tabu status is cancelled.

3.3.3. Intensification and diversification procedures

These procedures are used for appropriately managing the exploration *vs.* exploitation tradeoff on the search space. They enable to either explore the solution space more thoroughly in the neighbourhood of a solution or move the search to other areas of the solution space. The former feature is known as *intensification* and the later *diversification*. In the case of the intensification, the aim is to deeply explore some promising area of the solution space. This is essentially done by rewarding solutions in a neighbourhood of a solution. While, the diversification tries to remedy one of the main issues in local search methods, namely, the "*locality*" of the search, that is, the search could not have the chance to explore certain areas of the solution space. The diversification is achieved by penalizing solutions (in fact, features of solutions) so that the search is moved away from the current area and is implemented in two forms: soft diversification, which moves to a new search area but close to the current search area, and strong diversification, which moves the search away from current search area.

For the implementation of intensification and diversification, the method uses long term memory to know those solution features that have most frequently appeared in solutions along the search process. In the case of intensification, most frequent features are rewarded while in case of diversification, the less frequent features are promoted (the most frequent ones are penalized) in a temporarily modified fitness function.

Intensification. The TS works with a set of *elite solutions*, consisting of top best so far solutions. Usually the number of elite solutions is kept small (typically 10-20 solutions). Based on the elite solutions found along the search process, a data structure $freqs\_best$ stores the best features of the solutions, namely, for any router r and position $(posx, posy)$ in the grid area, the $freqs\_best[r][posx][posy]$ indicates the frequency of placing router r to the position $(posx, posy)$. Then, based on the frequency information a roulette-wheel criteria is used to assign router r to cell $(posx, posy)$ of the grid according to probability p_{rxy}:

$$p_{rxy} = \frac{freqs\_best[r][x][y]}{\sum_{x=0,y=0}^{x=size\_grid\_x-1,y=size\_grid\_y-1} freqs\_best[r][x][y]}.$$

Soft diversification. The implementation of the soft diversification is done by penalizing those movements that have been most occurring at solutions found along the search process and promote those that have been less occurring. Basically, a vector $tfrequency$ stores the frequency with which routers have changed positions ($tfrequency[r]$ indicates the frequency that router r has changed position in the grid area). The frequency vector is decreasingly sorted and a number of routers (between 5%-10%) of smallest frequencies is selected and are moved to most frequent positions according to $freqs\_best$. In other words, routers that have changed less frequently their positions are now "forced" to change their positions.

Strong diversification. Unlike soft diversification, which essentially makes small perturbations to current solution, the strong diversification implements an "escape" mechanism to restart the search at a different search area. Obviously, the simplest way would be to generate a random solution and re-start the search from there, having thus the drawback of starting the search from scratch. We have implemented it in a way that a large perturbation to current solution is done (about 25% of routers are changed their positions). Thus, the newly generated solution is considerably different from the current one yet it inherits some of the features found so far along the search process.

4. Experimental evaluation

4.1. Problem instances

In order to evaluate the performance of the TS method, we have developed a simple simulator, which can generate instances of the problem using

different distributions of client mesh nodes (Uniform, Normal, Exponential and Weibull). We have generated a total 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. Instances are arranged in three groups, each having 16 instances and are labelled $I_{x\times x\_D\_k}$, where: (a) 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; (b) D stands for the distribution of the client mesh nodes in the grid area; four distributions are considered: Uniform (U), Normal (N), Exponential (E) and Weibull (W); and, (c) k is the index of the instance.

Thus, we have 16 instances for each grid size (32, 64 and 128, resp.) and within each group we have 4 instances for each distribution (Uniform, Normal, Exponential and Weibull, resp). For instance, in this notation, $I_{64\times64\_N\_3}$ denotes the third instance of a 64×64 grid area, with mesh clients nodes positions generated using Normal distribution. Finally, the 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.

4.2. Parameter setting

The values of the main parameters of TS method have been set as follows, based on a previous parameter tuning using instances other than those used for the computational results. Instances different from those used for computational results were generated for tuning purpose. The number of iterations for the TS was set according to the instance size, namely, $size\_grid\_x \cdot size\_grid\_y \cdot k$, where k is a small constant multiplying factor. In this way, the number of iterations is increased with the increase in instance size.

start_choice *parameter.* This parameter indicates to the TS which adhoc method to select to compute the initial solution. We have used the `StartNear` adhoc method for instances generated using the normal distribution, while we have used `StartHotSpot` for the other types of instances, which distinguishes for fast covering of mesh clients following the hots spots idea.

use_delta_function *parameter.* This parameter tells to the algorithm whether the objective function should be evaluated anew for any solution or it can use

information on previous fitness function to compute the fitness of the newly explored solution. It should be noted that in our case, the fitness function computation is complex due to any router change requires re-computing the networking connections. Due to this, we have used the the `use_delta_function` parameter to improve the computation time needed for the fitness function evaluation.

`tabu_size` parameter. This parameter indicates the size of the tabu hash table (TH) used to further filter already visited solutions. Its value has been selected to be a sufficiently large prime number, more precisely it has been set to 51113.

`max_tabu_status` parameter. This is the maximum number of iterations that a movement could remain tabu, unless it satisfies the aspiration criteria, has been set to `number_routers/2`.

`elite_size` parameter. The `elite_size` parameter used for the `best_sols` has been chosen in the range [10,20].

`aspiration_value` parameter. This parameter indicates the "tabu life" condition of a solution, after which the tabu status can be removed. It has been set to `(max_tabu_status/2)`-$\log_2$`max_tabu_status`.

Finally, the maximum number of iterations per one intensification / diversification phase is set to $\log_2(\max(\texttt{grid\_size\_x}, \texttt{grid\_size\_y}))$.

4.3. Computational results

The method was run at our department's HPC cluster[1] 15 times for the same instance using the same configuration to avoid biased results. Average values of the size of the giant components and of the number of user coverage are reported, for the hierarchic optimization model (see Subsect. 2.2). The search effort of the method is referred to in terms of the *search iterations*, which are not just simple local movements, but search phases of the Tabu Search, i.e. the exploration of a neighborhood of a solutions (see lines 6-8 in Alg. 1). Additionally, the intensification and diversification processes *per se* consists of a number of simple local search iterations (in our case the maximum number of iterations per one intensification / diversification phase is set to $\log_2(\max(\texttt{grid\_size\_x}, \texttt{grid\_size\_y})$, see Subsect. 4.2).

[1] http://rdlab.lsi.upc.edu/index.php/en/services/cluster.html

Table 1: TS results for 32×32 grid size instances (16 routers and 48 clients).

Instance	size of giant component		Users covered	
	best	avg	best	avg
I32x32_U_1	16	16	10	9
I32x32_U_2	16	16	7	7
I32x32_U_3	16	16	9	9
I32x32_U_4	16	16	8	7
I32x32_N_1	16	16	27	26
I32x32_N_2	16	16	24	23
I32x32_N_3	16	16	26	26
I32x32_N_4	16	16	24	23
I32x32_E_1	16	16	14	14
I32x32_E_2	16	16	22	22
I32x32_E_3	16	16	22	22
I32x32_E_4	16	16	27	27
I32x32_W_1	16	16	21	19
I32x32_W_2	16	16	23	22
I32x32_W_3	16	16	26	25
I32x32_W_4	16	16	31	30

4.3.1. Small size instances (32×32 grid size)

We present in Table 1 results obtained with TS for small size instances.

As can be seen from the results in Table 1, full network connectivity of mesh router nodes is achieved (all 16 routers are connected in one giant component). Good results were obtained also for the user coverage, although due to the hierarchic approach the user coverage was not optimal. It is interesting to see how many iterations needed the method to achieve this. We show in Fig. 3 the graphical representation of evolution of size of giant component (averaged) for TS for the four distributions used for mesh client nodes. As can be seen from the figure, for all instances TS achieved full connectivity in about 40-50 search phases of Tabu Search. It should be noted that a phase of the tabu search consists essentially of searching a neighborhood of current solution.

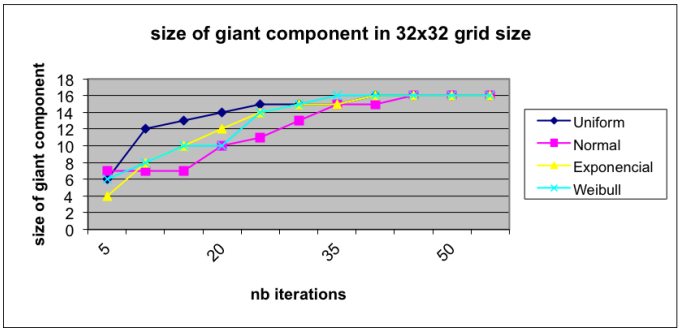

Figure 3: Evolution of size of the giant component for 32×32 grid size computed by TS.

4.3.2. Medium size instances (64×64 grid size)

We give in Table 2 computational results for instances of benchmark of 64×64 grid area. Again, the TS method achieved full network connectivity of the mesh routers nodes (all 32 routers are connected in one giant component).

The evolution of the size of the giant component obtained by TS for 64×64 grid area size is shown in Fig. 4, where we can also see that the optimal solution was found in about 80-100 iterations.

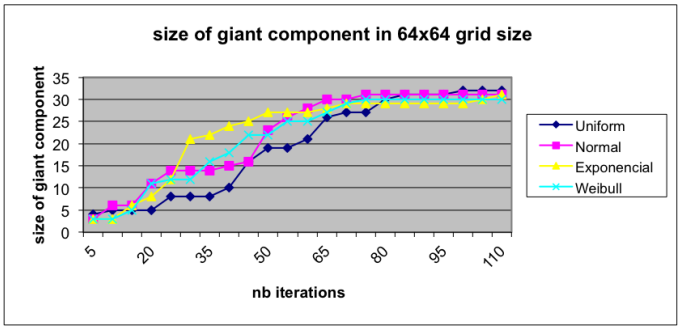

Figure 4: Evolution of size of the giant component for 64×64 grid size computed by TS.

4.3.3. Large size instances (128×128 grid size)

We give in Table 3 computational results for instances of benchmark of 128x128 grid area. For such larger size instances, TS achieved to compute a giant component of almost all of 64 routers. A graphical representation of the evolution of size of giant component is shown in Fig. 5. As can be seen from the figure, the largest giant component is achieved around 900 iterations for all but Normal distribution.

Table 2: TS results for 64×64 grid size instances (32 routers nodes and 96 clients.)

Instance	size of giant component		Users covered	
	best	avg	best	avg
I64x64_U_1	32	32	8	8
I64x64_U_2	32	32	7	7
I64x64_U_3	32	32	9	8
I64x64_U_4	32	32	7	5
I64x64_N_1	32	32	30	29
I64x64_N_2	32	32	38	37
I64x64_N_3	32	32	35	34
I64x64_N_4	32	32	38	36
I64x64_E_1	32	32	43	43
I64x64_E_2	32	32	44	40
I64x64_E_3	32	32	39	39
I64x64_E_4	32	32	40	40
I64x64_W_1	32	32	40	40
I64x64_W_2	32	32	43	43
I64x64_W_3	32	32	44	44
I64x64_W_4	32	32	38	38

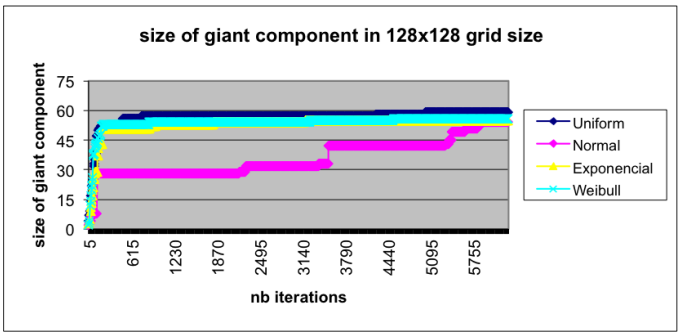

Figure 5: Evolution of size of the giant component for 128×128 grid size computed by TS.

Table 3: TS results for 128×128 grid size instances (64 routers nodes and 192 clients).

Instance	size of giant component		Users covered	
	best	avg	best	avg
I128x128_U_1	60	57	15	11
I128x128_U_2	57	53	12	9
I128x128_U_3	58	56	15	12
I128x128_U_4	57	51	11	7
I128x128_N_1	54	51	14	10
I128x128_N_2	56	52	11	7
I128x128_N_3	58	55	14	10
I128x128_N_4	55	52	12	9
I128x128_E_1	55	53	89	84
I128x128_E_2	56	53	176	172
I128x128_E_3	57	55	167	163
I128x128_E_4	56	54	181	176
I128x128_W_1	57	57	67	67
I128x128_W_2	57	57	67	67
I128x128_W_3	57	57	67	67
I128x128_W_4	57	56	67	66

4.4. Convergence to optimal solutions and evaluation

The TS implementation found an optimal placement of router nodes in the grid area for all small and medium size instances, and an almost optimal placement for large size instances. The TS showed thus its efficiency in solving the router placement under the hierarchical optimization problem in which the size of the giant component is primary objective. TS method showed robustness at computing the size of giant component while for the user coverage showed more variability depending on the mesh routers node distribution. The performance of the algorithm, nevertheless, was rather poor for the user coverage, especially for larger the instances, which is due the limitation of the hierarchical optimization model.

From the monitoring of execution of the algorithm, we observed that in average the TS converged to optimal solution in about 40-50 iterations for small instances, which is doubled for medium size instances (about 90-100 iterations). For large instances the algorithm needed a considerable number of iterations to converge (about 900 iterations in average).

4.5. Comparison to Simulated Annealing results

We studied the performance of the Simulated Annealing method for the problem in [16]. The TS outperformed the SA algorithm for all instances and notably for medium and large size instances.

5. Conclusions and future work

In this work we have presented the implementation and evaluation of Tabu Search method for solving the problem of mesh router nodes placement in Wireless Mesh Networks (WMNs). This problem falls in the family of node/facility location problems. The placement of mesh routers has a direct impact on the performance and connectivity of the WMN. We have considered the kind of WMN in which client nodes are stationary and of *a priori* known positions. While the router nodes placement problem can be seen as a family of multi-objective optimization problems, we have considered the bi-objective case which, given a number of routers to deploy in a 2D area and the positions of a number of client nodes, aims to compute the placement of mesh routers so as to maximize the size of the giant component in the mesh routers network (for measuring network connectivity) and maximizing the user coverage. For this version of the problem, we have considered the hierarchical optimization problem under which maximizing the

size of the giant component is considered primary objective. The TS implementation has been evaluated using a set of instances varying from small to large size generated from different distributions of mesh node clients (Uniform, Normal, Exponential and Weibull). The computational results showed the effectiveness and good convergence of TS method in computing the mesh router nodes placement. The method achieved to establish network connectivity of mesh router nodes in rather few iterations, although the number of iterations increases with the increase in the instance size. However, the algorithm performed rather poor for user coverage. In this regard, the computational results prompt to either increase the number of routers to deploy or use more advanced Pareto-like optimization approaches. In fact, for arbitrary values of the number of mesh routers and number of mesh clients, the optimal solutions need not to provide full network connectivity and full user coverage, therefore some trade-off is needed at design phase of the WMN.

In our future work we would like to investigate the resolution of the problem under simultaneous bi-objective optimization approach (maximizing at the same time the size of the giant component and user coverage) either through a weighted sum or a more general Pareto-like approach.

Acknowledgment

This work was partially supported by TIN2013-46181-C2- 1-R COMMAS Project Computational Models and Methods for Massive Structured Data.

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] M. Balazinska and P. Castro. Characterizing Mobility and Network Usage in a Corporate Wireless Local-Area Network. In *Proceedings of MobiSys*, 303-316, San Francisco, CA, 2003.

[4] A. Barolli, F. Xhafa, M. Takizawa Optimization Problems and Resolution Methods for Mesh Placement Nodes in Wireless Mesh Networks: A survey. In *Proceedings of NBiS-2011 Conference*, IEEE CPS, 7-9 September 2011.

[5] D.P. Blinn, T. Henderson, D. Kotz. Analysis of a Wi-Fi Hotspot Network. In *WitMeMo05: International Workshop on Wireless Traffic Measurements and Modeling*. 1-6, USENIX Association, 2005.

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

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

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

[9] F. Glover. Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers and Op. Res.*, **5** (1986) 533–549.

[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) 1-89.

[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*, 612-616, 2007.

[15] F. Xhafa, Ch. Sanchez, L. Barolli, E. Spaho. Evaluation of Genetic Algorithms for Router Nodes Placement in Wireless Mesh Networks. *Journal of Ambient Intelligence and Humanized Computing*, **1** (2010) 271-282.

[16] F. Xhafa, A. Barolli, Ch. Sanchez, L. Barolli. A simulated annealing algorithm for router nodes placement problem in Wireless Mesh Networks. *Simulation Modelling Practice and Theory* **19**:2276-2284, 2011.

[17] J. Wang, B. Xie, K. Cai, D.P. Agrawal. Efficient Mesh Router Placement in Wireless Mesh Networks, *Proceedings of IEEE MASS'07*, 1-9, 2007.

[18] J. Wong, R. Jafari, M. Potkonjak, Gateway placement for latency and energy efficient data aggregation, in *Proc. IEEE LCN*, 490-497, 2004.