Router Node Placement with Service Priority in Wireless Mesh Networks Using Simulated Annealing with Momentum Terms

Chun-Cheng Lin, *Member, IEEE,* Lei Shu, *Member, IEEE,* Der-Jiunn Deng, *Member, IEEE*

***Abstract*—In wireless mesh networks (WMNs), mesh clients communicate with each other via the gateway and bridging functions of mesh routers. The performance of a WMN is generally affected by its network connectivity and client coverage, both of which are determined by its router node placement (RNP) in the deployment area. For simplicity, previous works considered only the RNP where each mesh client is served as an equal. In practice, however, mesh clients should be served with different priorities owing to factors such as their importance and their different payments for the service access. To fulfil this requirement, by assuming that each mesh client is also associated with a service priority, this paper investigates a RNP problem with a service priority constraint in which the mesh clients with service priorities higher than a threshold must be served. Given that this problem inherited from the complexity of the original RNP problem is computationally intractable in general, this work also develops a novel simulated annealing (SA) approach that takes into account momentum terms to improve the efficiency and accuracy of annealing schedules and prevent fluctuations in values of the acceptance probability function. Additionally, the time complexity of the proposed SA algorithm is analyzed. Furthermore, evaluation of different-size instances under various parameters and annealing schedules demonstrates the superiority of the proposed approach.**

***Index Terms*—Simulated annealing, router node placement, wireless mesh network, annealing schedule.**

I. Introduction

Advances in multi-radio and multi-channel technologies have led to the emergence of wireless mesh networks (WMNs), capable of offering high-speed Internet access for mobile users without time or location constraints, and achieving a greater cost-efficient design and deployment than those of conventional networks based on wired connections. A detailed survey on WMNs can be found in [1]. Owing to the generally complex nature of problems associated with each layer of WMNs, most studies have focused on the simplified problems after some degree of abstraction and assumptions. To fulfil more practical requirements, WMNs face various challenges, including link scheduling [2], gateway selection [3], routing problems with different concerns [4], assignment problems for multi-channel WMNs [5], [6], [7], and security in WMNs [8].

WMNs are composed of mesh clients and mesh routers, in which mesh routers serve as the access points that allow mesh clients to access the Internet, as well as connect to other mesh routers through point-to-point wireless communication. Router node placement in WMNs (WMN-RNP for short) involves placing mesh routes in a geographical deployment area in order to optimize some performance measures of WMNs. The previous works for the WMN-RNP problem can generally be classified into two categories. The first category focuses on tailoring the problem model by adding different objectives and mesh nodes for fulfilling practical requirements, including throughput optimization for gateway placement [4], [9], bi-objective RNP for two-tier WMNs [10], and multi-objective WMN-RNP [11]. Exact optimal solutions for the WMN-RNP problem are generally unattainable in deterministic polynomial time. Therefore, the second category of previous works focuses on designing metaheuristic approaches, including local search algorithm [12], simulated annealing [13], genetic algorithm [14], tabu search [15], and particle swarm optimization [16].

To our knowledge, the previous works on the WMN-RNP problem assume that all mesh clients are treated as an equal when evaluating network performance. In practice, however, WMNs may differ in performance (e.g., network connectivity, stability, as well as quality of service) owing to factors such as the increasing demand for service, interference from other wireless devices, and surrounding environments. Therefore, some of the mesh clients may be willing to pay more for a higher-quality network connectivity and service. To fulfil this practical requirement, this work extends the original WMN-RNP problem to a more realistic one that incorporates the service priority constraint, in which each mesh client is also associated with a service priority; the mesh clients with higher priorities than a threshold must be served as well.

Assume that the location and service priority of each mesh client are predefined. As for the problem of the WMN-RNP with service priority constraint (WMN-RNPSP for short), of priority concern is how to find a placement of mesh routers in a rectangular deployment area in order to maximize two network performance measures: network connectivity (i.e., the size of the greatest component of the topology graph underlying the WMN) and client coverage (i.e., the number of clients within the radio coverage of mesh routers). Although this problem is a combinatorial optimization problem related to facility and location problems [6], the WMN-RNPSP

C.-C. Lin is with the Department of Industrial Engineering and Management, National Chiao Tung University, Hsinchu 300, Taiwan. E-mail: cclin321@nctu.edu.tw

L. Shu is with the Guangdong Provincial Key Laboratory of Petrochemical Equipment Fault Diagnosis, Guangdong University of Petrochemical Technology, Maoming 525000, China. E-mail: lei.shu@lab.gdupt.edu.cn

D.-J. Deng is with the Department of Computer Science and Information Engineering, National Changhua University of Education, Changhua 500, Taiwan. D.-J. Deng is the corresponding author of this paper. E-mail: djdeng@cc.ncue.edu.tw

problem is challenging owing to its following three features: the locations of mesh routers are not predetermined; each mesh router has a different-size radio coverage; each mesh client with a different service priority is of priority concern. Inherited from the original WMN-RNP problems [17], [18], [19], the WMN-RNPSP problem is computationally intractable in general. Therefore, this work presents a novel simulated annealing (SA) approach with momentum terms based on [20] to provide a highly promising solution to the above problem. The proposed SA approach increases the efficiency of the SA algorithm by additionally considering momentum terms, capable of improving the speed and accuracy of the original annealing schedules and preventing extreme changes in values of the acceptance probability function. Besides the original neighbor selection scheme, three neighbor selection schemes are compared with the proposed SA approach. Analysis results demonstrates the outperformance of our proposed SA approach not only by discussing in detail how different parameters affect the performance, but also comparing with the original SA algorithm.

This work contributes to optimizing the router node placement in WMNs in the following ways:

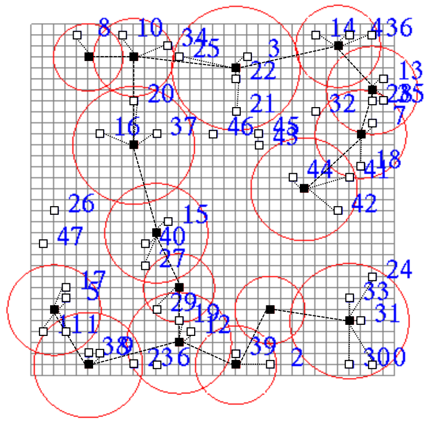
- The service priority constraint in the WMN-RNP problem is considered here, which is more realistic than that in previous works [12], [13], [15], [16];
- A novel SA approach with momentum terms for the WMN-RNPSP problem is developed, capable of achieving convergence of the algorithm more efficiently and finding better solutions;
- Time complexity of the proposed SA approach is analyzed theoretically; and
- Three neighbor selection schemes are considered in the proposed SA approach for comparison. Analysis results indicate that the proposed approach is highly effective in finding the optimal solutions.

II. Preliminaries

This section first introduces the basic settings of the original WMN-RNP problem and, then, formulates mathematically the WMN-RNPSP problem of concern in this work. The SA algorithm with momentum terms proposed in [20] is also reviewed.

A. The WMN-RNP Problem

The WMN consists of mesh routers and mesh clients, in which each mesh client can only communicate with the node within the same radio coverage or any node that can be accessed via multi-hop router communications. Restated, a mesh client cannot communicate with other nodes in the network if it is not located within the radio coverage of a mesh router. For instance, Figure 1(a) displays a WMN deployed in a grid area in which each mesh client (white node) is located at some grid point; in addition, each mesh router (black node) has a different-size radio coverage (represented by a circle centered at the mesh router). Notably, all nodes must be placed only at grid points of the deployment area. The WMN-RNP problem [13], [15] focuses on how to find a placement of


(a)

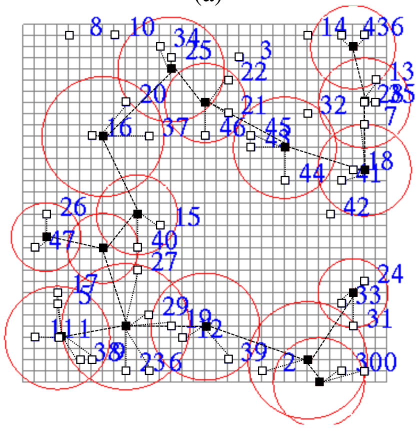

(b)

Fig. 1. Comparison of the experimental results between (a) our concerned WMN-RNPSP problem and (b) the previous WMN-RNP problem.

mesh routers in the grid deployment area in order to ensure that both *network connectivity* and *client coverage* are as large as possible.

To evaluate the above network performance measures, this work establishes a network topology graph $G = (V, E)$ based on the deployed locations of mesh routers (Figure 1(a)), in which V denotes the set of all mesh routers and mesh clients, and E represents the set of edges, which include the following two connections. First, for overlapped radio coverages of two mesh routers, there is an edge between the two mesh routers. Second, for a mesh client located within the radio coverage of a mesh router, there is an edge between the mesh client and the mesh router. *Network connectivity* refers to the size of the greatest graph component of graph G, while *client coverage* refers to the number of covered mesh clients. With the above definitions, *network connectivity* and *client coverage* for the instance in Figure 1(a) are 16 and 43, respectively.

B. The WMN-RNP Problem with Service Priority Constraint

To satisfy the practical requirement of mesh clients in terms of nonuniformity, assume that each mesh client is associated with a service priority. Additionally, consider the service priority constraint in which the mesh client with a higher priority than a threshold must be served. A mathematical model for WMN-RNP with service priority constraint (WMN-RNPSP for short) is established as follows.

Consider a WMN with n mesh routers and m mesh clients deployed in a two-dimensional grid deployment area, in which the location of each mesh client is predefined. Meanwhile, the location of each mesh router must be determined. The ability to determine a placement X of mesh routers of a WMN allows us to establish a graph $G_X = (V, E_X)$ where

- $V = \{r_1, r_2, \cdots, r_n, c_1, c_2, \cdots, c_m\}$ where each r_i denotes a mesh router for $i \in \{1, 2, \cdots, n\}$, and each c_j denotes a mesh client for $j \in \{1, 2, \cdots, m\}$; let $R = \{r_1, r_2, \cdots, r_n\}$ and $C = \{c_1, c_2, \cdots, c_m\}$;
- each mesh router $r_i \in R$ has a radio coverage Υ_i, which can be viewed as a circle centered at node r_i with a different radius γ_i;
- each mesh client $c_j \in C$ has a service priority ω_j;
- according to placement X, if $\Upsilon_i \cap \Upsilon_k \neq \emptyset$ for any two mesh routers $r_i, r_k \in R$, then edge $(r_i, r_k) \in E_X$; and
- according to placement X, if $c_j \in \Upsilon_i$ for any mesh client $c_j \in C$ and any mesh router $r_i \in R$, then edge $(c_j, r_i) \in E_X$.

Notably, for similarity, assume that two mesh routers can communicate with each other only if their radio coverages are overlapped. This assumption is acceptable because two mesh routers with overlaped radio coverage imply that their close geographical proximity so that they can communicate with each other.

With respect to placement X, assume that there are h subgraph components $G_X^1, G_X^2, \cdots, G_X^h$ in G_X, i.e., $G_X = G_X^1 \cup G_X^2 \cup \cdots \cup G_X^h$, and $G_X^a \cap G_X^b = \emptyset$ for $a, b \in \{1, \cdots, h\}$. The network connectivity is calculated based on the size of the greatest subgraph component, which is expressed as follows:

$$\phi(G_X) = \max_{a \in \{1, \cdots, h\}} \{|G_X^a|\}$$

The client coverage can be expressed as follows:

$$\psi(G_X) = |\{j; d_X(c_j) > 0 \text{ for } j \in \{1, \cdots, m\}\}|$$

where $d_X(c_j)$ denotes the degree of node c_j in graph G_X. This work attempts to maximize the two network performance measures simultaneously as follows:

$$f(G_X) = \lambda \cdot \phi(G_X)/(n+m) + (1-\lambda) \cdot \psi(G_X)/m \quad (1)$$

where λ represents the weighting scale in the range $[0, 1]$, and it controls the balance between the two terms of the objectives. Notably, the denominator of each term of the equation is used for normalization.

With the above notations, the WMN-RNPSP problem can be stated as follows:

ROUTER NODE PLACEMENT WITH SERVICE PRIORITY CONSTRAINT IN WMNS (WMN-RNPSP)
INSTANCE: a number κ, and a WMN deployed in a two-dimensional $W \times H$ grid area where each mesh client is located at a particular grid point.
QUESTION: find a placement X of mesh routers of the WMN problem such that $f(G_X)$ for the underlying graph G_X is maximized. Meanwhile, the mesh clients with the top κ service priorities must be served, and each mesh router must be placed at a grid point.

Figure 1 compares the experimental results of the WMN-RNPSP problems of concern in this work with the previous WMN-RNP for a WMN with 16 mesh routers (black nodes) and 48 mesh clients (white nodes) labeled by their service priority ordering, in which a smaller label number represents a higher priority. Let $\kappa = m/3$, i.e., the mesh clients with the top $48/3 = 16$ service priorities (labeled by 0 to 15) must be served. Note that κ can be any number, and we set it to be $m/3$ in this example. Analysis results for the previous WMN-RNP problem in Figure 1(b) indicate that the mesh clients labeled by 3, 8, 10 and 14 are not serviced even if they have higher priorities. As for the experimental results for the WMN-RNPSP problem of concern in this work (Figure 1(a)), all the mesh clients labeled by 0 to 15 are serviced, although some mesh clients with lower priorities are still not served.

C. Simulated Annealing with Momentum Terms

Simulated annealing (SA) is a metaheuristic algorithm like genetic algorithm (GA), which can solve various combinatorial optimization problems. As GA has been applied to various applications (e.g., [21], [22], [23]), some recent applications of SA in a variety of fields include [24], [25], [26]. SA simulates the cooling process of metals by heating and cooling the metals to increase the size of crystals and reduce defects. Initially, an encoding method must be designed to represent the solution of the concerned problem as a *state* of the metals in SA, as well as establish the relationship between the objective function of the concerned problem and the *energy* model of the metals. Consequently the optimal solution corresponds to a state with the lowest energy. In SA, the *annealing schedule* and the *Metropolis rule* significantly influence the performance of SA. Let T_k denote the temperature of the k-th iteration. Three conventionally adopted annealing schedules [20] that update the temperature per iteration are stated as follows:

- *Geometric*: $T_{k+1} = \alpha \cdot T_k$ where α denotes a cooling parameter ranging from 0.9 to 0.999;
- *Logarithmic*: $T_{k+1} = c/\log(b_0 + k)$ where c represents a value determined by many experimental trials and depends on the concerned problem; b_0 is a base; and
- *Boltzmann*: $T_{k+1} = T_0/\log(1+k)$ where T_0 denotes the initial temperature, i.e., $T_0 = T_h$.

The SA algorithm is also characterized by the *Metropolis rule*, which considers an acceptance probability function that overcomes the local optimal problem and leads the system towards the global optimal solution. To allow to escape from the local optimal solution, the rule allows for acceptance of an undesired state with the following Boltzmann acceptance probability: $P(\Delta\mathcal{E}) = e^{-\Delta\mathcal{E}/(B \cdot T_k)}$ where B denotes the Boltzmann constant and is generally set to one; $\Delta\mathcal{E} = \mathcal{E}'_k - \mathcal{E}_k$ for the minimization problem (in which $\mathcal{E}_k$ and $\mathcal{E}'_k$ represent the energy values of the current state and neighboring state, respectively) and $\Delta\mathcal{E} = \mathcal{E}_k - \mathcal{E}'_k$ for the maximization problem.

A previous work [20] suggested adding momentum terms to SA to improve cooling rate and prevent extreme changes in values on the acceptance probability function, allowing for a more efficient solution of the problem and a higher accuracy of the solution. For an annealing schedule, to avoid extreme cooling and increase the efficiency of deriving better solutions, three annealing schedules were proposed by adding momentum terms to the above three annealing schedules, respectively, as follows:

- *Hybrid*: $T_{k+1} = T_k - \alpha \cdot T_k - k \cdot (T_k - T_{k-1})/e^k$.
- *Extended Logarithmic*: $T_{k+1} = c/\log(T_0 + k) - k/e^k - \sqrt{\log(k)}$.
- *Extended Boltzmann*: $T_{k+1} = T_0/\log(1+k) - \log(1+k)$.

Further details of those designs can be found in [20].

As for the Metropolis rule, the extended Boltzmann acceptance probability was proposed based on the notion of adding momentum terms to the Boltzmann acceptance probability as follows:

$$P(\Delta\mathcal{E}) = e^{-\Delta\mathcal{E}/(B \cdot T_k)} \tag{2}$$

where $\Delta\mathcal{E} = (\mathcal{E}'_k - \mathcal{E}_k) - \beta \cdot B \cdot T_k \sqrt{\mathcal{E}'_k - \mathcal{E}_k}$ for minimization problem, or $\Delta\mathcal{E} = (\mathcal{E}_k - \mathcal{E}'_k) - \beta \cdot B \cdot T_k \sqrt{\mathcal{E}_k - \mathcal{E}'_k}$ for maximization problem, where we recall that $\mathcal{E}_k$ and $\mathcal{E}'_k$ are the energy values of current state and neighboring state, respectively; β is a running time parameter that depends on the annealing schedule and initial temperature. Notably, $\Delta\mathcal{E}$ is always no less than zero because the neighboring state is accepted if $\Delta\mathcal{E} < 0$.

III. Our Simulated Annealing Approach with Momentum Terms for the WMN-RNPSP Problem

This section describes in detail the proposed SA approach with momentum terms for the WMN-RNPSP problem. Basic components of the SA approach (i.e., solution representation, fitness function, neighbor selection scheme, repairing function, and acceptance criterion) are introduced. Details of the SA algorithm are provided as well.

A. Solution Representation

The solution to the WMN-RNPSP problem of concern in this work is a placement of m mesh routers in a two-dimensional $W \times H$ grid deployment area, whose lower-left corner is place at the origin of an $x \times y$ plane. Restated, the (x, y)-coordinates of the m mesh routers should be determined for each candidate solution. Therefore, a state (solution) of SA for the WMN-RNPSP problem can be expressed by $X = (x_1, x_2, \cdots, x_{2n})$ where (x_{2i-1}, x_{2i}) denotes the (x, y)-coordinate of mesh router r_i for $i = 1, 2, \cdots, n$.

Additionally, each iteration of SA maintains the optimum state found so far, and, hence, records it in the vector $P = (p_1, p_2, \cdots, p_{2n})$ and also records its energy (objective) value $f(P)$. Notably, each node is restricted to be placed at a grid point, and, hence, x_i and p_i are integers for any $i \in \{1, \cdots, n\}$.

Since all of the mesh routers are placed within a $W \times H$ area, the following constraint is obtained: $\forall i \in \{1, 2, \cdots, n\}$,

$$0 \le x_{2i-1} \le W, \text{ and } 0 \le x_{2i} \le H. \tag{3}$$

B. Fitness Function

In the setting of the original SA, a state of metals has an energy value corresponding to the objective value of the solution that the state represents. Although the original SA attempts to find the state with the minimal energy value, the WMN-RNPSP problem of concern in this work focuses on obtaining a solution with the maximal objective value. For clarity, by use of the terminology of evolutionary computation, the proposed SA approach assumes that each state has a *fitness* value rather than an energy value. Therefore, this work attempts to find a state with the fittest value, i.e., the maximal fitness value. Doing so makes us to simply allow the fitness function to be the objective function as follows. Consider a state that represents a candidate placement X of mesh routers. A topology graph G_X underlying the WMN deployment can be established, as described in Subsection II-B. As mentioned earlier, the problem of concern in this work attempts to maximize the size of the greatest subgraph component $\phi(G_X)$ and the client coverage $\psi(G_X)$ simultaneously. Hence, SA searches for a state that maximizes the fitness function $f(G_X)$ for G_X in Equation (1).

C. Neighbor Selection

Neighbor selection plays an important role in finding the optimal state of the metals. The neighboring state is generally transformed from the current state, and differs only slightly from the current state. For experimental diversity, consider the following three neighbor selection schemes:

- *Standard*: Randomly select a mesh router in the current state, and then place it at an arbitrary grid point (e.g., see Figure 2(a));
- *Local*: Select a mesh router randomly, and then place it at an arbitrary grid point within its local area (e.g., see Figure 2(b)); and
- *Random*: All of the mesh routers are relocated randomly (e.g., see Figure 2(c)).

Notably, more than just a slight modification, the Random neighbor selection scheme changes the locations of all of the mesh routers. Hence, comparing this scheme with the other two schemes with small-scale modification is of relevant interest.

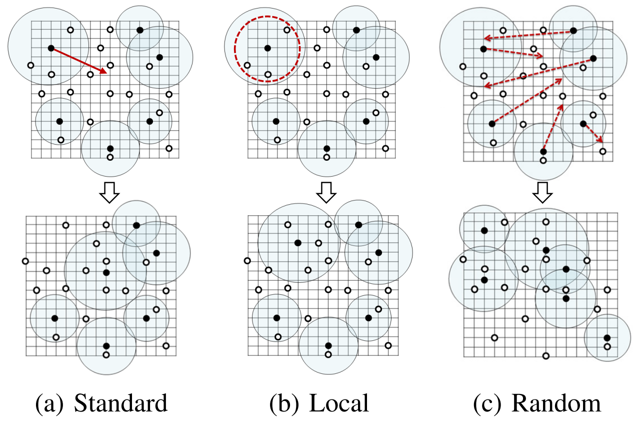


Fig. 2. Illustration of the proposed three neighbor selection schemes.

D. Repairing Function

Any of the three neighbor selection schemes described above may destroy the solution feasibility. Restated, the service priority constraint is violated as some of the mesh clients with the top κ service priorities are not served. Maintaining the solution feasibility in each iteration of the SA algorithm depends on our ability to design how to rectify this infeasibility. As mentioned earlier, each mesh client c_j is associated with a service priority ω_j, and a smaller ω_j value implies a higher priority. This work attempts to repair the operator by moving the mesh routers that cover low-priority mesh clients to serve those that must be served but are not served yet.

Our algorithm for the repairing operator is stated in detail in Algorithm 1, which is explained in the following. Line 1 considers each mesh client c_j that must be served but is not yet served. Line 2 then allows S to be the set of mesh clients which are covered but do not belong to those with the top κ service priorities, and are sorted in a decreasing ordering of service priorities (i.e., lowest priority to highest priority). To determine whether the job of covering the concerned mesh client c_j is complete, Line 3 uses a flag variable called $isDone$, which is set to false initially.

Algorithm 1 REPAIRING

```
1:  for each mesh client $c_j$ that must be served but is not yet do
2:      let $S$ denote a list of mesh clients in which they are covered but do not belong to those with the top $\kappa$ service priorities; they are sorted in a decreasing ordering of service priorities (i.e., lowest priority to highest priority)
3:      $isDone \leftarrow false$
4:      for each mesh client $c_t$ in list $S$ do
5:          let $R_\ell$ denote the set of the mesh routers that cover mesh client $c_t$
6:          for each mesh router $r_i$ in $R_\ell$ do
7:              find an available grid point $g_j$ that is the closest to $c_j$
8:              if the placement of $r_i$ at grid point $g_j$ does not violate the service priority constraint then
9:                  place mesh router $r_i$ at grid point $g_j$ to serve mesh client $c_j$
10:                 $isDone \leftarrow true$
11:                 break the for loop
12:             end if
13:         end for
14:         if $isDone = true$ then break the for loop
15:     end for
16:     if $isDone = false$ then discard the solution
17:     update the mesh clients that must be served but is not yet
18:     if all the mesh clients are served then return the solution
19: end for
```

We repeat the for loop of Lines 4 – 15 until each mesh client c_t in S is considered. Notably, since S is a list of mesh clients from the lowest priority to the highest priority, c_t is a mesh client in S with the lowest priority that is not yet considered. Line 6 finds the mesh routers R_ℓ that cover mesh client c_t. Then, the for loop of Lines 6 – 13 is repeated until each mesh router in R_ℓ is considered. In each iteration of the loop, Lines 7 and 8 consider whether to move mesh router r_i to serve mesh client c_j (by placing r_i at an available grid point that is the closest to c_j). The move is accomplished if it does not lead to any new violation of the service priority constraint (i.e., the mesh clients that were served by mesh router r_i originally are still served after the move). Since the move is executed in the most inner loop, the flag variable $isDone$ is allowed to be true in Line 11.

In Line 14, if $isDone = true$ (i.e., the job of serving mesh client c_j is done), break up the for loop of Lines 4 – 15 until R_ℓ is empty. In Line 16, if $isDone = false$ (i.e., mesh client c_j is still not served), meaning that the feasibility of the solution cannot be solved, then the solution is discarded. Lines 17 and 18 update the solution and return it as the final solution if there is no constraint violation.

Notably, Algorithm 1 may lead to a situation in which although some low-priority mesh clients are not covered, it can still guarantee the solution feasibility.

Consider Figure 1(a) as an example, in which mesh clients are labeled by 0 – 47; in addition, the mesh clients labeled by 26, 43, 45, 46, 47 are not served. By assuming that the mesh clients with the top 26 service priorities must be served, mesh client #26 (i.e., c_j in Line 1 of Algorithm 1) must be served. Line 2 creates a list S that includes mesh clients #44, #42, #41, $\cdots$, #27, #28, i.e., all of the mesh clients exclusive of #26, #43, #45, #46, #47. Line 4 finds mesh #44 with the lowest priority. Line 5 indicates that R_ℓ includes the mesh router between mesh clients #44 and #42 in Figure 1(a). Lines 7 – 12 move the mesh router to serve mesh client #26 by placing it at one of the four grid points near mesh client #26. The algorithm is completed since it serves mesh client #26 successfully and does not induce any new violation.

The time complexity of Algorithm 1 is analyzed as follows:

Lemma 1. *The worst-case upper bound of the time complexity of Algorithm 1 is $O(\kappa \cdot (m-\kappa) \cdot (m+n) \cdot n)$.*

Proof. Line 1 in Algorithm 1 considers at most κ mesh clients, because an attempt is made to serve at most κ mesh clients. Hence, the main loop has at most κ iterations. Relatively, in Line 2, S may include the remaining $m-\kappa$ mesh clients at most, and hence, $|S| \leq m-\kappa$. Notably, despite an attempt to sort S in Line 2, it does not need to be sorted because the data structure containing all mesh clients in a decreasing ordering of service priorities was known originally. Restated, Line 2 can be completed in $O(m-\kappa)$ time. Next, Line 3 is completed in $O(1)$ time.

Line 4 considers each mesh client in S, i.e., there are $O(m-\kappa)$ iterations. Next, in Lines 5 and 6, R_ℓ contains n mesh routers in the worst case, though the worst case does not occur often in practice. Line 7 computes the available closest grid point to c_j in $O(m+n)$ time, because the maximum number of the neighboring grid points of c_j that are occupied by other mesh routers and clients is bounded by $O(m+n)$.

Lines 8 – 12 and Line 14 are completed in $O(1)$ time. Lines 16 – 18 can be completed in $O(m)$ time.

In light of the above discussion, the time complexity of Algorithm 1 can be calculated as follows: $O(\kappa \cdot ((m-\kappa) + (m-\kappa)\cdot n \cdot (m+n) + m)) = O(\kappa \cdot (m-\kappa)\cdot(m+n)\cdot n)$. $\square$

E. Acceptance Criterion

After neighbor selection and repairing, the acceptance criterion for the neighboring state is based on the difference of the fitness values between the current state X and the neighboring state X' as follows:

- If $f(X') - f(X) \geq 0$, then X' is accepted; and
- If $f(X') - f(X) < 0$, then X' is accepted based on the Extended Boltzmann acceptance probability function in Equation (2), where $\Delta\mathcal{E}$ can be rewritten as $(f(X) - f(X')) - \beta \cdot B \cdot T_k\sqrt{f(X) - f(X')}$.

F. The proposed SA Algorithm

The proposed SA algorithm works with the current state X, which is a $2n$-length vector that represents a candidate solution of the WMN-RNPSP problem. First, each pair (x_{2i}, x_{2i+1}) in X is initialized to be a random position in the grid deployment area. The proposed SA algorithm then finds a neighboring state X' of the current state X and probabilistically determines whether the system moves to the neighboring state X' or remains in the current state X. In doing so, the system can move to a state of lower energy. This step is repeated until the system reaches a state that is acceptable for the problem of concern in this work, or until the maximum number of iterations is achieved.

Algorithm 2 states the proposed SA algorithm. The key steps of Algorithm 2 are explained as follows. Lines 1–4 are the initialization of current state X, best state P obtained so far, their fitness values, highest temperatures T_h, lowest temperature T_ℓ, initial temperature T_0, and initial iteration number k. Line 1 initializes the current state X randomly under Constraint (3), and calculates its fitness value by Equation (1). Line 2 assigns the current state X and its fitness value $f(X)$ to the best state P found so far and its corresponding fitness value $f(P)$, respectively. Lines 3 initializes temperatures T_h and T_ℓ, and assigns T_h to the initial temperature T_0.

Lines 5–23 are the main loop (outer loop) of the SA algorithm, which is repeated until the number of iterations k is greater than the maximum iteration number η or the current temperature T_k drops to the lowest temperature T_ℓ. At a fixed temperature T_k, the inner loop in Lines 6–20 determines whether to accept the neighboring state until the *equilibrium detection condition* is achieved, i.e., there have been consecutive τ times of rejecting the neighboring state (i.e., the solution has not been modified for a number of τ times) or the maximum number of the inner loops (Line 20) has been achieved. Line 7 generates the neighboring state X' by a neighbor selection scheme described in Subsection III-C. Line 8 checks whether the neighboring state violates the service priority constraint. If true, the repairing operation described in Subsection III-D is applied. Next, Line 9 calculates the fitness value $f(X')$ of X'. Once a better state is found, the current state is replaced by the neighboring state X'; in addition, the best state P found so far and its fitness value $f(P)$ are updated in Lines 10–14. Otherwise (i.e., the neighboring state X' is worse), a nonzero Expanded Botzmann acceptance probability is considered to accept the worse neighboring state in Lines 15–19. Line 21 updates the temperature according to an annealing schedule described in Subsection II-C. Finally, Line 24 outputs the final solution represented by P upon completion of the SA algorithm.

Algorithm 2 OUR SA WITH MOMENTUM TERMS

```
1: initialize the current state $X = (x_1, x_2, \cdots, x_n)$ randomly
   where $x_{2i-1}$ is an integer in the range $[0, W]$ and $x_{2i}$ is
   an integer in the range $[0, H]$ for each $i \in \{1, 2, \cdots, n\}$,
   and calculate its fitness value $f(X)$
2: $P \leftarrow X$ and $f(P) \leftarrow f(X)$
3: initialize temperatures $T_h$ and $T_\ell$, and let temperature
   $T_0 \leftarrow T_h$
4: $k \leftarrow 0$
5: while $T_k > T_\ell$ and $k < \eta$ where $\eta$ is the maximal iteration
   number do
6:   repeat
7:     generate the neighboring state $X'$ by a neighbor
       selection scheme (which may be one of the Standard, Local and Random neighbor selection schemes
       described in Subsection III-C)
8:     if $X'$ violates the service priority constraint, then repair its infeasibility as described in Subsection III-D
9:     calculate the fitness value $f(X')$ of neighboring state
       $X'$
10:    if $f(X') \geq f(X)$ then
11:      $X \leftarrow X'$ and $f(X) \leftarrow f(X')$
12:      if $f(X) > f(P)$ then
13:        $P \leftarrow X$ and $f(P) \leftarrow f(X)$
14:      end if
15:    else
16:      if $rand() < \exp(-\Delta\mathcal{E}/(B \cdot T_k))$ then
17:        $X \leftarrow X'$ and $f(X) \leftarrow f(X')$
18:      end if
19:    end if
20:  until there have been consecutive $\tau$ times of rejecting
     the neighboring state or the maximal iteration of the
     inner loop is achieved
21:  update temperature according a annealing schedule
     (which may be one of the Hybrid, Extended Logarithmic, Extended Boltzmann annealing schedules described in Subsection II-C)
22:  $k \leftarrow k + 1$
23: end while
24: output $P$ as the solution
```

Before the time complexity of Algorithm 2 is analyzed, the following lemma is obtained:

Lemma 2 *Given a placement X of mesh routers, the fitness value $f(G_X)$ with respect to X can be computed in $O(m \cdot n)$ time.*

Proof. The fitness value is calculated by first establishing the topology graph G_X underlying the concerned WMN instance in $O(m \cdot n)$ time. To do so, whether each of the m mesh clients is located within the radio coverage of each of the n mesh routers is verified. Once graph G_X is established, the size of the greatest component $\phi(G_X)$ is obtained in $O(1)$ time. Next, $O(m)$ time is taken to verify whether each of the m mesh clients is covered; hence, the client coverage $\psi(G_X)$ is obtained in $O(m)$ time. Therefore, $f(G_X)$ can be computed in $O(m \cdot n + m) = O(m \cdot n)$ time. □

Time complexity of the main loop (i.e., Lines 6 – 20) in Algorithm 2 is analyzed as follows:

Theorem 1 *Each iteration of the loop of Lines 6 – 20 in Algorithm 2 can be computed in $O(\kappa \cdot (m - \kappa) \cdot (m + n) \cdot n)$ time.*

Proof. First, the Standard, Local and Random neighbor selection schemes in Line 7 can obviously be computed in $O(1)$, $O(1)$, and $O(n)$ times, respectively. By Lemma 1, Line 8 is completed in $O(\kappa \cdot (m - \kappa) \cdot (m + n) \cdot n)$ time. By lemma 2, Line 9 is completed in $O(m \cdot n)$ time. Each of the other lines in the concerned loop is an assignment or comparative instruction, and hence can be done in $O(1)$ time. □

IV. Implementation and Analysis Results

The section implements the proposed SA approach to the WMN-RNPSP problem. The experimental data and environment are described first, and, then, a comprehensive experimental analysis is conducted under different neighbor selection schemes, annealing schedules and acceptance probability functions, as well as distributions of mesh clients.

A. Data and Environment

Let $U(a, b)$ denote the uniform distribution over $[a, b]$. By analogy with [13], the following three cases are considered (as also used in our previous work in [16]):

- **Case 1:** There are 16 mesh routers ($m = 16$) and 48 mesh clients ($n = 48$) on a 32×32 grid area ($W = H = 32$). Each mesh router is associated with a circular radio coverage with radius following a uniform distribution $U(3, 6)$. Meanwhile, the position of each mesh client is generated following a uniform distribution $U(0, 32)$ or a normal distribution $N(16, 16/3)$. Temperatures $T_h = 100$ and $T_\ell = 1$.
- **Case 2:** There are 32 mesh routers ($m = 32$) and 96 mesh clients ($n = 96$) on a 64×64 grid area ($W = H = 64$). Each mesh router is associated with a circular radio coverage with radius following a uniform distribution $U(4\sqrt{2} - 2, 8\sqrt{2} - 2)$. Meanwhile, the position of each mesh client is generated following a uniform distribution $U(0, 64)$ or a normal distribution $N(32, 32/3)$. Temperatures $T_h = 50$ and $T_\ell = 1$.
- **Case 3:** There are 64 mesh routers ($m = 64$) and 192 mesh clients ($n = 192$) on a 128×128 grid area ($W = H = 128$). Each mesh router is associated with circular radio coverage with radius following a uniform distribution $U(7, 14)$. Meanwhile, the position if each mesh client is generated following a uniform distribution $U(0, 128)$ or a normal distribution $N(64, 64/3)$. Temperatures $T_h = 50$ and $T_\ell = 1$.

Notably, the deployment area is a grid, implying that each coordinate is an integer. For each case, 5 instances are generated, in which mesh clients are deployed in the area according to a uniform or normal distribution. Our experiments apply the following parameter settings for the SA: $\lambda = 0.3$ (i.e., weighting parameter of objective function), $\eta = 200$ (number of iterations), and $\tau = 20$ (i.e., allowed number of consecutive times of rejecting the neighboring state). Additionally, $\rho = 50$ independent runs are performed to avoid fortuitous results, explaining why averaged results are used. The proposed SA algorithm is tested on a PC with an Intel Core i7-3770 CPU and 16 GB memory. Under such a setting, the average running times for executing an instance of 32×32, 64×64, and 128×128 grid cases are 0.646, 12.864, and 56.003 seconds, respectively, implying that the proposed SA approach can cope with the concerned problem efficiently.

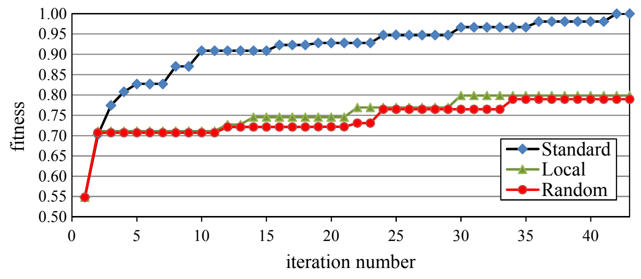
B. Experimental Results under Different Neighbor Selection Schemes

Performance of the proposed algorithm under different neighbor selection schemes (Subsection III-C) is evaluated by running 50 times of SA with or without momentum terms for each problem case under different neighbor selection schemes. The average of all the outputted fitness values in Table I is recorded as well. Notably, unless stated otherwise, the following basic settings of the SA components are as follows: the SA algorithm without momentum terms uses the Geometric annealing schedule and the Boltzmann acceptance probability. Meanwhile, the SA algorithm with momentum terms uses the Hybrid annealing schedule and the Expanded Botzmann acceptance probability. Table I reveals that the cases using the Standard neighbor selection scheme always generate better solutions than those in the other cases.

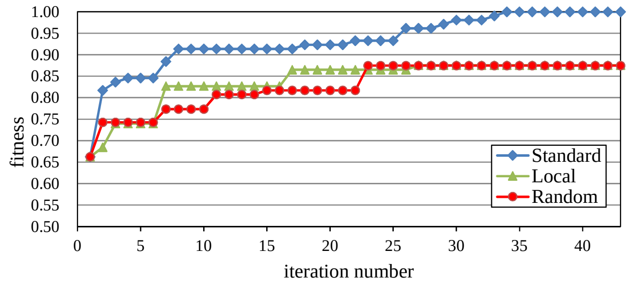
TABLE I
Performance of using different neighbor selection schemes on the original SA and our proposed SA with momentum terms for 32×32, 64×64 and 128×128 grid cases.

Case	Momentum terms	Standard	Local	Random
32×32 grid	without	0.955385	0.743010	0.744469
	with	0.982656	0.788635	0.783781
64×64 grid	without	0.923776	0.873375	0.876479
	with	0.999229	0.879516	0.871833
128×128 grid	without	0.884500	0.859797	0.860487
	with	0.981529	0.866550	0.868177

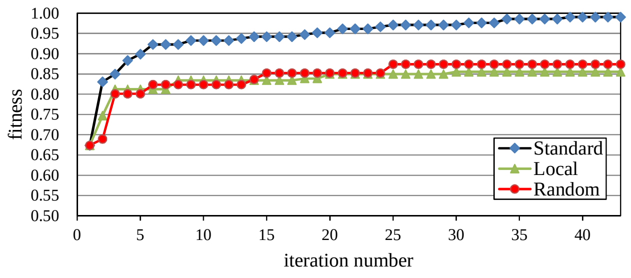
Next, solution convergence of the proposed SA algorithm is observed by plotting the outputted fitness values versus the number of iterations of the main loop of our proposed SA algorithm for an instance of each grid case in Figure 3. Figure 3(a) shows the convergence trend for the 32×32 grid case indicating that the Standard neighbor selection scheme converges very close to one (i.e., the best fitness value) at



(a)



(b)



(c)

Fig. 3. Convergence trend of our proposed SA under different neighbor selection schemes for (a) 32×32, (b) 64×64, and (c) 128×128 grid cases.

the 40th iteration and demonstrates a very good convergence performance. Meanwhile, the Local and Random neighbor selection schemes improve the solutions gradually after the first iteration, finally obtaining relatively worse solutions.

Figure 3(b) shows the convergence trend for the 64×64 grid case. From this, we infer that the Standard neighbor selection scheme performs better than the other two schemes. A similar inference can also be made for the case for the 128×128 grid case in Figure 3(c). However, the gap between the Standard and the other two neighbor selection schemes in Figure 3(b) (resp., Figure 3(c)) becomes narrower than that in Figure 3(a). This finding suggests that different neighbor selection schemes do not significantly influence larger grid cases.

C. *Experimental Results with Different Annealing Schedules and Acceptance Probability Functions*

This work also attempts to understand how using different annealing schedules and acceptance probability functions influences the resulting fitness values. A thorough analysis is performed for all possible combinations of annealing schedules and acceptance probability functions in Table II. This table summarizes the experimental results of the proposed SA algorithm using Boltzmann and extended Boltzmann acceptance probability functions, respectively, under six annealing schedules (mentioned in Section II) for each grid case. Notably, each entry in Table II is computed by the average of running 5 instances with uniformly-distributed mesh clients for the proposed SA approach.

TABLE II
RESULTANT FITNESS VALUES OF OUR PROPOSED SA USING BOLTZMANN AND EXPANDED BOLTZMANN ACCEPTANCE PROBABILITY FUNCTIONS, RESPECTIVELY, UNDER DIFFERENT ANNEALING SCHEDULES FOR THREE GRID CASES.

Acceptance probability	Annealing schedule	32×32	64×64	128×128
Botzmann	Geometric	0.949562	0.928870	0.882378
	Logarithmic	0.769073	0.892849	0.870396
	Boltzmann	0.766104	0.895083	0.866521
	Hybrid	0.870500	0.852052	0.831609
	Extended Logarithmic	0.765312	0.896604	0.866997
	Extended Boltzmann	0.763875	0.895531	0.867477
Expanded Botzmann	Geometric	0.982292	0.962144	0.986125
	Logarithmic	0.981375	0.967908	0.985766
	Boltzmann	0.985792	0.977908	0.962916
	Hybrid	0.978687	0.945529	0.958355
	Extended Logarithmic	0.983083	0.953463	0.986221
	Extended Boltzmann	0.982406	0.964149	0.962278

Experimental results of the Boltzmann probability acceptance function in Table II indicate that the Geometric annealing schedule performs better than the others for 32×32 and 64×64 grid cases, especially for the 32×32 grid case. Meanwhile, the extended Logarithmic annealing schedule performs better for the 128×128 grid case. Experimental results under extended Boltzmann probability acceptance function in Table II indicate that almost all annealing schedules can perform better than those under the Boltzmann probability acceptance function. We thus conclude that the extended Boltzmann acceptance probability function of the novel SA performs better.

D. *Experimental Results of Different Distributions of Mesh Clients*

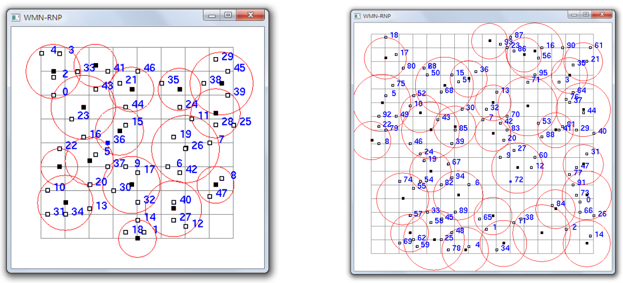
This work also evaluates the performance of the proposed SA algorithm for different problem instances. As mentioned earlier, mesh clients are distributed in the deployment area based on a uniform or normal distribution function. Tables III summarize all of the problem instances for all cases, with each one having four columns that store the best fitness value, average fitness value, the worst fitness value, and the standard deviation of fitness values of each instance; ten rows contain 5 instances with uniform distribution and 5 instance with normal distribution. The average of each column is stored at the bottom row of Table III, indicating that the proposed SA approach performs excellently for various grid cases. Especially for the 32×32 and 64×64 grid cases, the best fitness values for nearly all of the instances are optimal. Restated, the proposed SA approach for small-size and middle-size cases had a high probability of achieving the optimal solutions for both network connectivity and client coverage.

TABLE III
THE STATISTICS FOR THE THREE GRID CASES.

32×32 grid case				
instance	Best	Mean	Worst	SD*
Uniform_inst 1	1.000000	0.970719	0.932812	0.017754
Uniform_inst 2	1.000000	0.990750	0.961458	0.011835
Uniform_inst 3	1.000000	0.978708	0.942188	0.017179
Uniform_inst 4	1.000000	0.987958	0.942708	0.013422
Uniform_inst 5	0.985937	0.958458	0.922917	0.014834
Normal_inst 1	1.000000	0.996344	0.817188	0.025854
Normal_inst 2	1.000000	0.995781	0.827083	0.024931
Normal_inst 3	1.000000	0.978031	0.922917	0.020199
Normal_inst 4	1.000000	0.975292	0.922917	0.023592
Normal_inst 5	1.000000	0.972958	0.922917	0.025047
average	0.998594	0.9805	0.9115105	0.0194646
64×64 grid case				
instance	Best	Mean	Worst	SD*
Uniform_ins 1	1.000000	0.962948	0.901562	0.030137
Uniform_inst 2	1.000000	0.969005	0.874740	0.029385
Uniform_inst 3	0.990365	0.951599	0.911719	0.022636
Uniform_inst 4	0.983333	0.931099	0.865104	0.032537
Uniform_inst 5	0.954427	0.876341	0.807552	0.033011
Normal_inst 1	1.000000	0.998073	0.913281	0.012312
Normal_inst 2	1.000000	0.998849	0.944792	0.007808
Normal_inst 3	1.000000	0.989469	0.738542	0.041472
Normal_inst 4	1.000000	0.989016	0.882552	0.026658
Normal_inst 5	1.000000	0.989802	0.877344	0.028532
average	0.991215	0.965541	0.868067	0.025713
128×128 grid case				
instance	Best	Mean	Worst	SD*
Uniform_inst 1	0.990365	0.960885	0.879688	0.015012
Uniform_inst 2	0.980729	0.960461	0.937370	0.010884
Uniform_inst 3	0.985547	0.953370	0.908594	0.015991
Uniform_inst 4	0.977214	0.932828	0.869922	0.020921
Uniform_inst 5	0.995182	0.979461	0.959115	0.008762
Normal_inst 1	1.000000	0.961464	0.879688	0.015941
Normal_inst 2	0.985547	0.969656	0.942318	0.010350
Normal_inst 3	0.980729	0.958208	0.892969	0.017861
Normal_inst 4	1.000000	0.981003	0.900130	0.020739
Normal_inst 5	0.969922	0.944716	0.791927	0.024531
average	0.986524	0.960205	0.896172	0.016099

* SD denotes standard deviation.

To visualize the placement results, this work implements a visualization user interface to show the solution placement. For instance, if the effectiveness of the proposed SA algorithm in achieving the optimality for the 32×32 and 64×64 grid cases is of interest, the validity of the solutions can be verified by visualizing their solution placements in Figures 4(a) and 4(b), respectively. In these figures, each white node is a mesh client and is labeled by its priority ordering; each black node is a mesh router, and the circle centered at each mesh router is its radio coverage. Figure 4 reveals that each white mesh client is covered, and the underlying topology graph due to overlaped radio coverages is connected. Hence, the solution correctness is verified via visualization.



(a) (b)

Fig. 4. Visualization of the WMNs for (a) 32×32 and (b) 64×64 grid cases, where it should be noticed that each gray grid represents a 4×4 grid.

V. CONCLUSIONS AND FUTURE WORK

This work has presented a simulated annealing (SA) approach with momentum terms to optimize the placement of mesh routers with service priority constraint in wireless mesh networks. This work significantly contributes to addressing a more realistic problem, in which the service priority constraint is added and an SA approach with momentum terms is provided to increase the speed and accuracy of the original SA. Additionally, the theoretical time complexity of the proposed SA approach is also provided. Analysis results demonstrate that the proposed SA approach performs well. For different grid cases, the most effective strategies of the annealing schedule and acceptance probability function are identified. Analysis results further demonstrate that the Boltzmann schedule with extended Boltzmann probability function is the most effective strategy for 32×32 and 128×128 grid cases. Meanwhile, the Geometric model with the extended Boltzmann probability function is the most effective strategy for 64×64 grid case. Moreover, the proposed SA algorithm is an effective method for the concerned problem, owing to its ability to achieve the connectivity of the entire network and cover nearly all mesh clients in various grid cases.

Efforts are underway in our laboratory to solve the dynamic version of the concerned problem and consider the optimization of other objectives simultaneously (e.g., minimum number of mesh routers and minimum traffic) in order that the problem is more realistic. We recommend that future research is to develop other metaheuristic approaches or an SA algorithm integrated with other hybrid schemes [27] to improve the computational performance. Parametric analysis of the proposed approach should also be performed.

ACKNOWLEDGEMENTS

The authors would like to thank Yi-Ling Lin for improving the quality of this paper. This work has been supported in part by NSC 101-2628-E-009-025-MY3 and NSC 102-2221-E018-012-MY3, Taiwan, as well as the Guangdong University of Petrochemical Technology's Internal Project No. 2012RC0106 and Educational Commission of Guangdong Province, China Project No. 2013KJCX0131.

REFERENCES

[1] I. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: A survey," *Computer Networks*, vol. 47, no. 4, pp. 445–487, 2005.

[2] K. Papadaki and V. Friderikos, "Approximate dynamic programming for link scheduling in wireless mesh networks," *Computers & Operations Research*, vol. 35, no. 12, pp. 3848–3859, 2008.

[3] K. Gokbayrak and E. A. Yıldırım, "Joint gateway selection, transmission slot assignment, routing and power control for wireless mesh networks," *Computers & Operations Research.*, vol. 40, no. 7, pp. 1671–1679, 2013.

[4] B. Aoun, G. Kenward, R. Boutaba, and Y. Iraqi, "Gateway placement optimization in wireless mesh networks with QoS constraints," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 11, pp. 2127–2136, 2006.

[5] J. Crichigno, M.-Y. Wu, and W. Shu, "Protocols and architectures for channel assignment in wireless mesh networks," *Ad Hoc Networks*, vol. 6, pp. 1051–1077, 2008.

[6] J. Current, M. Daskin, and D. Schilling, *Discrete network location models*, ser. Facility Location: Applications and Theory. Springer Berlin Heidelberg, 2001, ch. 3.

[7] V. Ramamurthi, A. Reaz, D. Ghosal, S. Dixit, and B. Mukherjee, "Channel, capacity, and flow assignment in wireless mesh networks," *Computer Networks*, vol. 55, pp. 2241–2258, 2011.

[8] O. E. Muogilim, K.-K. Loo, and R. Comley, "Wireless mesh network security: A traffic engineering management approach," *Journal of Network and Computer Applications*, vol. 34, pp. 478–491, 2011.

[9] M. Tang, "Gateways placement in backbone wireless mesh networks," *International Journal of Communications, Network and System Sciences*, vol. 1, pp. 44–50, 2009.

[10] A. Franklin and C. S. R. Murthy, "Node placement algorithm for deployment of two-tier wireless mesh networks," in *Proceedings of 50th Annual IEEE Global Communications Conference (GLOBECOM 2007).* IEEE Press, 2007, pp. 4823–4827.

[11] A. Hafid, D. Benyamina, and M. Gendreau, "Wireless mesh network planning: A multi-objective optimization approach," in *Proceedings of 5th International Conference on Broadband Communications, Networks and Systems (BROADNETS 2008).* IEEE Press, 2008, pp. 602–609.

[12] C. Sánchez, F. Xhafa, and L. Barolli, "Locals search algorithms for efficient router nodes placement in wireless mesh networks," in *Proceedings of International Conference on Network-Based Information Systems (NBIS 2009).* IEEE Press, 2009, pp. 572–579.

[13] F. Xhafa, A. Barolli, C. Sánchez, and L. Barolli, "A simulated annealing algorithm for router nodes placement problem in wireless mesh networks," *Simulation Modelling Practice and Theory*, vol. 19, no. 10, pp. 2276–2284, 2011.

[14] F. Xhafa, C. Sánchez, and L. Barolli, "Genetic algorithms for efficient placement of router nodes in wireless mesh networks," in *Proceedings of 24th IEEE International Conference on Advanced Information Networking and Applications (AINA 2010).* IEEE Press, 2010, pp. 465–472.

[15] A. Barolli, C. Sánchez, F. Xhafa, and M. Takizawa, "A tabu search algorithm for efficient node placement in wireless mesh networks," in *Proceedings of 3rd International Conference on Intelligent Networking and Collaborative Systems (INCoS 2011).* IEEE Press, 2011, pp. 53–59.

[16] C.-C. Lin, "Dynamic router node placement in wireless mesh networks: A PSO approach with constriction coefficient and its convergence analysis," *Information Sciences.*, vol. 232, pp. 294–308, 2013.

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

[18] L. Rabiner, "Combinatorial optimization: Algorithms and complexity," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 32, no. 6, pp. 1258–1259, 1984.

[19] J. Wang, B. Xie, K. Cai, and D. Agrawal, "Efficient mesh router placement in wireless mesh networks," in *Proceedings of IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2007).* IEEE Press, 2007, pp. 1–9.

[20] M. M. Keikha, "Improved simulated annealing using momentum terms," in *Proceedings of 2nd International Conference on Intelligent System, Modeling and Simulation (ISMS 2011)*, 2011, pp. 44–48.

[21] F. Gonzalez-Longatt, P. Wall, P. Regulski, and V. V. Terzija, "Optimal electric network design for a large offshore wind farm based on a modified genetic algorithm approach," *IEEE Systems Journal*, vol. 6, no. 1, pp. 164–172, 2012.

[22] W. H. Ip, D. Wang, and V. Cho, "Aircraft ground service scheduling problems and their genetic algorithm with hybrid assignment and sequence encoding scheme," *IEEE Systems Journal*, vol. 7, no. 4, pp. 649–657, 2013.

[23] S. Saha, "GARO framework: A genetic algorithm based resource optimization for organizational efficiency," *IEEE Systems Journal*, vol. 7, no. 4, pp. 889–895, 2013.

[24] C.-C. Lin, Y.-Y. Lee, and H.-C. Yen, "Mental map preserving graph drawing using simulated annealing," *Information Sciences*, vol. 181, no. 19, pp. 4253–4272, 2011.

[25] J. Chen and W. Z. . M. Ali, "A hybrid simulated annealing algorithm for nonslicing VLSI floorplanning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, vol. 41, no. 4, pp. 544–553, 2013.

[26] J. Trovao, V. Santos, P. Pereirinha, H. Jorge, and C. Antunes, "A simulated annealing approach for optimal power source management in a small EV," *IEEE Transactions on Sustainable Energy*, vol. 4, no. 4, pp. 867–876, 2013.

[27] F. J. Rodriguez, C. Garcia-Martinez, and M. Lozano, "Hybrid metaheuristics based on evolutionary algorithms and simulated annealing: Taxonomy, comparison, and synergy test," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 6, pp. 787–800, 2012.