Computer Communications 34 (2011) 1375–1388

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

S^2U: An efficient algorithm for optimal integrated points placement in hybrid optical-wireless access networks

Yu Liu *, Chi Zhou, Yu Cheng

Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, Illinois 60616, USA

ARTICLE INFO

Article history:
Received 13 August 2010
Received in revised form 12 January 2011
Accepted 12 February 2011
Available online 17 February 2011

Keywords:
Optical network
Wireless network
Integrated system
Clustering algorithm
Approximation ratio

ABSTRACT

Integration of optical and wireless networks is considered as one of the promising technologies for next generation Internet access. In this paper, we consider the integrated points placement problem in the hybrid optical-wireless system for optimal resource utilization under the given constraints including hop count, cluster size, and relay load. While the optimization formulation is an NP-hard problem in general, we propose a polynomial-time heuristic algorithm – S^2U algorithm to obtain the near-optimal solution that minimizes the number of integrated points required to support all wireless BSs residing in the wireless part of the integrated system. In contrast to the existing work, our S^2U algorithm forms the clusters starting from the network edge towards its center and the construction of clusters is not only based on the greedy idea but also considers load balancing. We present a theoretical analysis of the complexity of the proposed S^2U algorithm and its approximation ratio to the optimal solution. Furthermore, we present extensive numerical results to compare the proposed S^2U algorithm with the main existing methods. It is shown that S^2U can not only cover a network with a smaller number of integrated points, but also achieve better network performance in terms of the average transmission delay (average hop count) and load balance. In addition, we compare our results with the optimal solution obtained via CPLEX in terms of the minimum number of integrated points. The results show that the gap between the results obtained from our S^2U algorithm and the optimal results is within 5% in average.

1. Introduction

The hybrid optical and wireless networks have been proposed as a promising approach to meet the increasing demand for higher bandwidth requirement, provide broadband and ubiquitous high-speed internet access, and address the growing gap between core network and local area network (last mile problem) effectively [1–6]. This hybrid system consists of a wireless network at the front end, and it is supported by an optical network at the back end. As a promising wireline solution to broadband access, optical network provides much better reliable transmission and much higher bandwidth in Gbps-scale compared with wireless networks as well as covering long distance (around 20 km) from the telecom central office (CO) to end users. However, the fixed infrastructure not only limits its coverage in a high densely populated area due to the high cost of fiber layout and equipments, but also makes it difficult to deploy in certain rugged environment. On the other hand, wireless networks support mobility and provide ubiquitous access for end users in the metropolitan area or the local area. However, wireless networks provide limited bandwidth compared with the optical fiber networks. Thus, integrating the optical and the wireless networks together can utilize their complementary advantages to provide better service for end users and increase revenue for the service providers.

The optical part can choose passive optical network (PON) [5] technologies, such as EPON, GPON, and the wireless part usually choose WiFi or WiMAX. In this paper, we choose the EPON [7] standard IEEE 802.3ah as the optical part and consider the IEEE 802.16–2004 standard of WiMAX [8] technology as the wireless part, since EPON can utilize the existing Ethernet infrastructure providing low cost and simplicity deployment, and WiMAX covers longer distance and provides higher data rates up to 1 Gb/s compared with WiFi technology, and supports both point-to-multipoint mode and mesh mode [8]. Fig. 1 illustrates the architecture of hybrid optical-wireless system. EPON uses a tree topology where the optical line terminal (OLT) located at the telecom CO connects multiple optical network units (ONUs) through the passive splitter. In the wireless part, WiMAX base stations (BSs) are grouped into clusters. Within each cluster, one WiMAX BS is selected as the gateway to combine with one ONU as the integrated point of the hybrid network and the rest of WiMAX BSs in the cluster referred as relay stations form a multi-hop wireless mesh

* Corresponding author. Tel.: +1 312 567 3916x7996/7998.
E-mail addresses: yliu77@iit.edu (Y. Liu), zhou@iit.edu (C. Zhou), cheng@iit.edu (Y. Cheng).

0140-3664/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2011.02.005

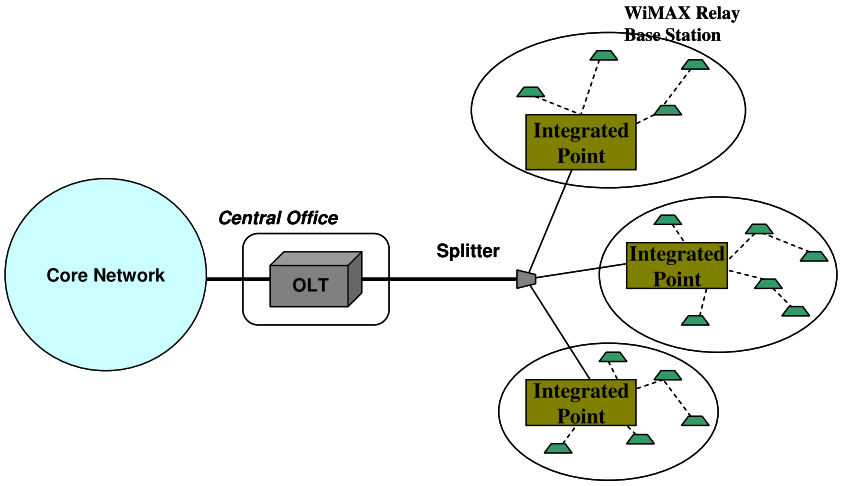


Fig. 1. Hybrid EPON-WiMAX network architecture.

network. These integrated points are the locations where the wireless part and optical part meet together. In data transmissions, an end user sends the packets to its closest relay station and this relay station forwards the packets to the integrated point through one hop or multi-hops in the cluster. Then integrated point will forward the packets to the OLT through the optical connection and finally to the core network. If the receiver resides in the hybrid optical-wireless network, the data flows in the reverse direction as described above.

When developing the hybrid optical-wireless network, we need to address several issues including placement of integrated points, resource allocation and scheduling, routing protocol design, etc., in order to make the whole system work efficiently with the minimum system cost. In this paper, we mainly focus on the integrated points placement problem. Given a wireless network topology, we aim to minimize the number of integrated points (or clusters that each cluster has one integrated point to support the rest of wireless relay stations within the cluster) to lower the fiber layout cost, equipments cost and installation cost, while still maintaining the network connectivity and satisfying the constraints including hop count, cluster size and relay load. This kind of problem can be modeled as a mixed integer linear programming problem, which is NP-hard in general [16,17]. Thus, we aim to develop an efficient heuristic algorithm in this paper to obtain the near-optimal solution.

Most of the existing works are trying to form clusters one by one as large as possible, the differences are where they form each cluster at each round and how large each cluster should be. This may result in the clusters with unbalanced load or large hop count, that is, some clusters may have very densely deployed nodes with one-hop away to the integrated point in some area in the network and some clusters may have just a few nodes but with large hop count in other area in the network. In [9], we propose a modified clustering algorithm (MCA) to achieve load balancing while minimizing the number of integrated points to cover all the wireless nodes. In this paper, we augment the MCA into a multi-stage algorithm called S^2U (Selection-Shift-Update) algorithm that can well approximate the optimal integrated points placement under multiple constraints including hop count, cluster size, and relay load. "Selection" is used to select the starting node and the corresponding integrated point for each cluster, "Shift" is used to reduce the number of clusters, and "Update" is used to update the integrated point location for each cluster to reduce the average hop count. In contrast to the existing work, our S^2U algorithm forms clusters starting from the network edge towards the center; constructs each cluster based on the considered constraints as well as a properly designed criterion for load balancing; and then performs a shift operation and updates the integrated point final location to further improve the performance. In general, our approach can minimize the number of clusters, reduce average transmission delay (average hop count), and balance load. Extensive numerical results verify that our proposed algorithm is better than others presented in the literature. Thereafter, we give a complexity analysis of the proposed S^2U algorithm and show that it is indeed a polynomial one. We also obtain the approximation ratio [22,23] of S^2U compared with the optimal results.

In summary, this paper has fourfold main contributions: (1) We design a novel polynomial algorithm, for optimal integrated points placement in a hybrid optical-wireless access network, while maintaining the considered three constraints: hop count, cluster size and relay load. (2) We derive the complexity analysis and obtain a derivation of the approximation ratio of our algorithm. (3) We analyze the impact of the considered constraints on the number of integrated points and system performance. (4) We present extensive numerical results to compare the proposed S^2U algorithm with the main existing methods, as well as the optimal result.

The rest of paper is organized as follows: Section 2 reviews related work. Section 3 describes the system model and formulates the placement problem as a linear programming optimization problem. In Section 4 the proposed S^2U algorithm is discussed in detail. Section 5 presents numerical results and analysis based on the obtained data. Finally we give the conclusions in Section 6.

2. Related work

ONU placement in hybrid optical-wireless broadband access networks has been studied in [6,14,15,21], using greedy algorithm, simulated annealing algorithm, and combined heuristic algorithm. The former two algorithms, given the number of ONUs and user locations, aim to find out the optimal ONU locations through minimizing some cost functions, which are usually formulated as the average distance between end users and ONUs to represent the fiber layout cost. A little different from the two former ones, the third algorithm first determines the number of Base Stations based on the co-channel interference threshold, then derives the optimal solution based on the greedy algorithm. In [10], the authors also study the ONU placement problem in fiber-wireless networks. They propose the tabu algorithm to minimize the total hop count when consider peer-to-peer communications in addition to the

traffic destined to the Internet. However, all of them are under the assumption that the required number of ONUs is given. Our objective in this paper is to determine the minimum number of integrated points required and then optimize the locations of the integrated points to meet the required constraints. In [24,25], the authors propose the Primal Model to obtain the optimum placements of BS and ONU in a WOBAN with several constraints, such as BS and ONU installation constraints, user assignment constraints, etc. However, the following questions need to be answered: (1) The number and the locations of BSs are not specified. (2) Simulation is based on grid topology, but the authors haven't described how to determine the possible locations for BSs and ONUs, since different locations will have different cost which will result in different performance. (3) The upper bound is obtained by the heuristic algorithm, subgradient method, but whether this is an upper bound has not been justified.

Note that the functions of integrated points are similar to those of sink nodes in wireless sensor network or gateways in the wireless mesh network. Thus, the integrated points placement problem has a similar essence to that of sink node deployment in wireless sensor networks [11–13], which adopt popular algorithms such as integer linear programming (ILP), genetic algorithm or k-mean clustering algorithm respectively, to find out the optimal locations of the sink nodes. Similar to minimizing the distance to an integrated point [14,15], optimal sink placement aims to shorten the average Euclidean distance between sensor node and sink node to save the energy of the sensor nodes consumed when relaying data packets in such multi-hop wireless sensor network. Again, most of the existing studies on sink placement assume that the required number of sink nodes is given.

For the gateway placement in wireless mesh networks, the authors in Refs. [16–20] have studied how to minimize the number of gateways given the network topology while taking into account several constraints, e.g., hop count, cluster size, etc. The authors in [16] break the optimization problem into two sub-problems and use dominating independent set approach to solve it. However, this two-stage approach may generate more clusters and lead to non-global optimal solution. In [17], the authors formulate three different link models and propose a greedy algorithm to form clusters iteratively to maximize the traffic demand, with a trade-off of degraded delay performance. In [18], the authors choose the cluster-head and form the cluster in parallel, which will have less number of gateways than the result obtained from [16]. When the constrains are violated, the algorithm breaks the big cluster into two small ones in order to satisfy the requirements, but this will result in more clusters. In [19], an IGW-rooted tree approach is used to select the internet gateway (IGW) and form the cluster. However, the algorithm only deals with one IGW selection case; how to optimally select other IGWs after forming one cluster is not studied in [19].

The most related work to ours is [20]. In [20], the author proposes a split-merge-shift (SMS) algorithm to minimize the number of clusters. This algorithm forms one-hop cluster first at the selected node with the maximum node degree. Then it merges neighboring small-size clusters. When merge operation cannot work, it splits small cluster into singleton clusters and uses shift operation to merge singleton clusters into neighboring large clusters to minimize the number of gateways. To the best of our knowledge, [20] is the most efficient work to get the minimum number of gateways in wireless mesh network. However, our S^2U algorithm can further improve the performance, compared to SMS, mainly in two aspects: (1) Our algorithm will update the integrated point position during the shift operation in order to reduce the number of integrated points. (2) We properly select the node to be shifted when multiple exchanged paths are available in the shift operation for better load balancing.

3. System model and formulations

3.1. System model

Our system model is shown in Fig. 2. We consider the integrated points placement problem for hybrid optical-wireless system where the wireless BSs form multi-hop wireless mesh network (WMN). We model an n-node WMN as an undirected connected graph $G(V,E)$, where V is the vertex set representing the set of nodes including all the BSs and E is the set of all the edges representing the communication link between two neighboring nodes. All the relay stations are considered as the candidate gateway that will be integrated with ONU. Only some of the relay stations are selected as gateways based on our algorithm, and each cluster is constructed around a selected integrated point. Two nodes are called neighbors if and only if their Euclidean distance $d(u, v)$ is less than or equal to the node transmission range t_r. The neighbors of a node v, denoted by N_v, is the set of nodes located within node v's transmission range. The number of neighbors of a node v is called the degree of v, denoted by δ. The residue node degree δ' is defined as the number of neighbors of a node excluding those nodes that have already been added into the clusters. The hop count between any two nodes u and v, denoted by $h(u, v)$, is the minimum number of hops between them. $h(u, v)$ is known when the network topology is given.

We assume the system has uniformly distributed traffic on each node (wireless BS) and every end-to-end communication needs to pass through a selected integrated point. Each node has the same transmission range. We only consider the uplink transmission since the three general constraints considered in this paper only relates to the uplink transmission.

3.2. Major constraints

In our system, we consider three major constraints, including hop count, cluster size and relay load. These constraints are similar to the ones in [20] and are specified as follows:

- Maximum hop count (R_h): Since all the traffic generated by the nodes in a cluster will be passed through the corresponding integrated point, the less the hop count, the smaller the transmission delay. Thus, setting an upper bound on the hop count is equivalent to allowing certain maximum transmission delay. The hop count constraint is specified as:

$$h(g_i, v) \leqslant R_h, \forall v \in G_i \text{ and } v \neq g_i \quad (1)$$

where G_i is cluster i, g_i is the integrated point in G_i, and $h(g_i, v)$ is the hop count between node v and integrated point g_i.

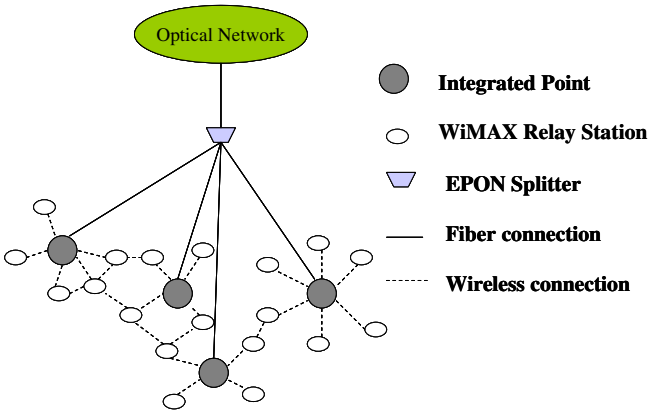


Fig. 2. System model.

- Maximum cluster size (C_{max}): In each cluster, the total traffic demand should not exceed the integrated point maximum capacity. Since we assume uniform traffic distribution on each node (relay station), the number of nodes in each cluster indicates the total traffic going through the integrated point. Thus, this constraint is formulated as:

$$C_i \leqslant C_{max},\ \forall G_i \in \mathcal{G} \tag{2}$$

where C_i is the number of nodes for cluster i and $\mathcal{G}$ is the union of all the clusters that covers the whole system.

- Maximum relay load (L_{max}): The total relay traffic passing through each node should not exceed L_{max}, and this constraint is formulated as:

$$L(u) = \Sigma_{u' \in T_u} l(u') \leqslant L_{max},\ u' \neq u \tag{3}$$

where L_U is the total traffic passing through node u, T_u is a sub-tree (rooted at node u) of the breadth-first-spanning (BFS) tree T_j rooted at integrated point j, and $l(u')$ is the traffic load generated by node u'. The BFS tree is constructed only when we check whether the relay load constraint is satisfied or not. The relay load is also reflected as the maximum number of descendents of this node along the spanning tree rooted at this node, since we assume uniform traffic distribution.

3.3. Integer linear program formulation/optimization model

Let $N = |V|$ be the number of wireless relay stations. We introduce a binary integer y_i to indicate whether node i has been selected as an integrated point in G, and $y_i = 1$ means that node i is selected as an integrated point. $\tau_{i,j}^k = 1$ means that node k is the parent node of node j along the Spanning tree rooted at node i. In order to represent the relationship between a relay station and an integrated point, we define another binary variable $x_{i,j}$. If this variable equals to 1, this means that the relay station $j \in N$ is assigned to integrated point i; otherwise, $x_{i,j}$ equal to 0. Thus, we can formulate the integrated points placement problem subject to the considered three constraints as an optimization problem as follows:

$$\min \sum_{i \in N} y_i \tag{4}$$

subject to

$$\forall j \in N : \sum_{i \in N} x_{i,j} = 1 \tag{5}$$

$$\forall i,j \in N : y_i \geqslant x_{i,j} \tag{6}$$

$$\forall i,j \in N : h(i,j) \cdot x_{i,j} \leqslant R_h \tag{7}$$

$$\forall i \in N : \sum_{j \in N} x_{i,j} \leqslant C_{max} \tag{8}$$

$$\forall i,k \in N : \sum_{j \in N} \tau_{i,j}^k \leqslant L_{max}, \quad k \neq i \tag{9}$$

$$\forall i,j,t \in N : x_{i,j} + x_{j,t} \leqslant 1, \quad i \neq j \tag{10}$$

$$\forall i,j,k \in N : x_{i,j} \leqslant \sum_{k \in N} x_{i,k} \cdot \tau_{i,j}^k \tag{11}$$

$$\forall i \in N : y_i \in \{0,1\} \tag{12}$$

$$\forall i,j \in N : x_{i,j} \in \{0,1\} \tag{13}$$

$$\forall i,j,k \in N : \tau_{i,j}^k \in \{0,1\} \tag{14}$$

Eq. (5) denotes that each relay station is assigned to one and only one integrated point, so this guarantees the whole network coverage and non-overlap clusters. Eq. (6) means that a relay station should be assigned to an integrated point after that integrated point has been set up. Eqs. (7)–(9) specify the three constraints in terms of the hop count, cluster size and relay load. Eq. (10) ensures that if a relay station has been assigned to an integrated point, then this relay station cannot be used as a candidate of integrated point. Eq. (11) ensures that a relay station can be assigned to an integrated point if at least one of the parents of this relay station in the shortest path tree rooted at this integrated point has already been assigned to this integrated point. Eqs. (12)–(14) means that the variables here only can take binary value.

The optimization problem can be solved via Integer Linear Programming (ILP), which, however, is proved to be NP-hard [16,17]. In practice, a linear programming software such as CPLEX or Matlab can be used to solve the optimization problem only for the network with small network size. It's hard to solve the ILP for large-scale network due to the complexity and memory limitation. Thus, we propose a heuristic algorithm to obtain the near-optimal result, which is described next in detail.

4. Heuristic integrated points placement algorithm

4.1. S^2U algorithm

The Greedy algorithm in [16] forms clusters iteratively and always picks a cluster that has the largest cluster size in the current stage. The SMS algorithm in [20] forms clusters from the node with the maximum node degree and then performs the split-merge-shift operations iteratively without relocating the gateway location. These approaches generally result in the clusters with unbalanced load or large hop count, that is, some clusters may have very densely deployed nodes one-hop away from the gateway in some area in the network and some clusters may have just a few nodes but with large hop count in other area in the network. To address this issue, our proposed algorithm consists of three phases: starting node and integrated point *selection*, *shift* operation and integrated point final location *update*. The essence of our S^2U algorithm is that it covers all the nodes cluster by cluster. In each cluster it first selects the starting node and the corresponding integrated point, and then forms this cluster based on some criterion which will be described later with that integrated point at the center. Clusters are formed from the network edge towards its center. After all the nodes have been covered, S^2U performs the shift operation to reduce the number of clusters and integrated points. Finally, it updates the integrated point location to better balance the load and reduce the hop count among clusters.

Algorithm 1. S^2U Algorithm

1: $S_1 = V = \{1,\ldots,N\}$, $I_1 = 1$;
2: **while** $S_1 \neq \emptyset$ **do**
3: **if** there is a node w_j with $\delta_{w_j} = 1$ & this node has not been added to any cluster **then**
4: $v = w_j$
5: **else if** I_1==1 **then**
6: $v = w_i$, where w_i has the largest vertical coordinate (tallest node) among all the nodes
7: $I_1 = 0$
8: **else**
9: $v = w_k$, where $h(w_k, w_{k'}) = 1, w_k \in S_1, w_{k'} \notin S_1, w_k = \arg\max_i\{\delta_i\}, \delta_i = \max\{\delta_j\}, j \in N_{w_k}^{R_h-1}$
10: **end if**
11: $u = i$, where $\delta_i = \max\{\delta_j\}, j \in N_v^{R_h-1}$
12: **while** Constraints C_{max}, R_h,L_{max} & Criterion 1 satisfied **do**
13: Add node v and nodes $v_i \in SP(v,u)$ to G_u
14: Add node v_j to G_u from 1-hop to R_h-hop away form node u

15: **end while**
16: Remove all the nodes added to G_u from S_1
17: **end while**
18: **if** $\sum_{i=1}^{m} C_i \leqslant (m-1)\cdot C_{max}$, where $C_i < C_{max}$ **then**
19: Calculate $M_2 = \lceil \sum_{i=1}^{m} \frac{C_i}{C_{max}} \rceil$, $M = M_1 + M_2$, $\overline{C_2} = \frac{1}{M_2}\cdot \sum_{i=1}^{m} C_i$
20: **while** $\sum_{i=1}^{m} C_i \leqslant (m-1)\cdot C_{max}$ & constraints C_{max}, R_h and L_{max} satisfied **do**
21: Find shift path rooted at the cluster with smallest cluster size
22: Shift the nodes in the cluster based on Criterion 2
23: Update cluster sizes and m
24: **end while**
25: **end if**
26: Update the integrated point for each cluster G: relocate integrated point u_i to location k based on $\forall k \in G_i : H(k) = arg\min\left\{\sum_{j\in G_i} h(k,j), k \neq j\right\}$, as long as the constraints are satisfied

We first define two criteria which will be used in the S^2U algorithm.

Criterion 1. A node v_j, which is more than one hop away from an integrated point u under consideration, can be added into the cluster associated with the integrated point only if the residue node degree δ' of node v_j satisfies $\delta' \leqslant C_{max} - C_i - 1$, except the case that $h(v_j, u) = 1$. C_i is defined as the number of nodes in cluster i.

The idea behind Criterion 1 is that we aim to avoid adding a node into the current cluster when this node may have a chance to be selected as the future integrated point, since a node will be added to the current integrated point only if all of its neighbors can be accommodated in this cluster. If a node cannot be added, this means that this node may have large residue node degree and will have a chance to be selected as the future integrated point.

Criterion 2. For nodes in each not-full-sized cluster, shift node with the largest hop count away from integrated point first; if two or more nodes have the same hop count, shift node with more outgoing-links connecting to the nodes in neighboring clusters along shift path first; if two or more nodes have the same number of outgoing-links, shift node with fewer incoming-links inside this not-full-sized cluster first.

The idea behind Criterion 2 is that by shifting the node with the largest hop count, we reduce the average hop count of current cluster. Additionally, we shift a node with better connection with target cluster, indicated by the number of outgoing-links and incoming-links.

The proposed S^2U approach is listed in Algorithm 1. The notations used in Algorithm 1 are listed in Table 1.

Table 1
Notation used in Algorithm 1.

C_i	Size of cluster i
G_i	Cluster i
I_1	Program indicator
m	Number of not-full-sized clusters
$h(w_k, w_{k'})$	Hop count between $w_k, w_{k'}$
$N_v^{R_h-1}$	Node v's $R_h - 1$ hop neighbors
$SP(v,u)$	Shortest path from node v to node u
M_1	Number of full-sized clusters
M_2	Minimum number of not-full-sized clusters
M	Minimum required number of clusters
$\overline{C_2}$	Average cluster size for not-full-sized clusters
$H(k)$	Total hop count with integrated point k in G_k

I_1 is an indicator that is used to make sure choosing the node with the largest vertical coordinate as the starting point only once after all 1-degree nodes have been selected or there are no 1-degree nodes in the network originally. Lines 3–10 of Algorithm 1 perform the starting node selection operation. If there is one or more unassigned nodes whose node degree equal to 1, select any one of them as the starting node v. If there is no 1-degree node or all of the 1-degree nodes have already been selected to form cluster, select the node at network edge, e.g., select the node with the largest vertical coordinate in the network as the starting node. Further, if the node with largest vertical coordinate has been selected as a starting node, we then select the node which is closest to the previous formed clusters, where closest means 1 hop count. When multiple closest nodes exist, we select the one with the maximum value of the maximum node degree among each closest node's $(R_h - 1)$ hop count neighbors.

Lines 11–15 perform the cluster construction operation. Select node u as the current cluster's integrated point where u has the maximum node degree among the nodes within starting node v's $(R_h - 1)$ hop count. Then select all the nodes along the shortest path (minimum hop count path) between the starting node and the corresponding integrated point first to join the cluster. New node with the smallest residue node degree is preferred to be added into current cluster first as long as Criterion 1 and all the constraints are satisfied. Nodes will be added continuously from 1-hop to R_h-hop away from integrated point u until the constraints are violated. Note that all the nodes should be checked from 1-hop to R_h-hop away from integrated point u unless the current cluster reach the maximum cluster size.

Lines 2–17 will be iteratively executed until all the nodes have been assigned to an integrated point, then the initial clusters have been constructed.

Lines 18–25 perform the shift operation. If the condition in line 18 is satisfied, it means that it's possible to reduce the current number of clusters at least by one, but whether the number of clusters can be reduced or not also depends on whether the constraints can be satisfied or not. Line 19 computes $\overline{C_2}$ to have an estimate of the number of nodes to be included in each cluster on average. We start from the cluster of the smallest size to perform the shift operation. Then construct shortest path tree rooted at that cluster representing the inter-domain connection among clusters and select one path which ends with a not-full-sized cluster. Nodes are shifted based on Criterion 2 and then updates the current existing cluster sizes and m. After this, choose another smallest-sized cluster and repeat the shift operation continuously until $\sum_{i=1}^{m} C_i \leqslant (m-1)\cdot C_{max}$ or the constraints are violated.

Line 26 perform integrated point location update operation. Integrated point in each cluster has been relocated to the position k where the total hop count in the corresponding cluster is minimized.

4.2. Algorithm illustration

We use a 40-node network to illustrate the S^2U approach step by step. The initial network topology is randomly generated, shown in Fig. 3(a). In this example we set $R_h = 3$, $C_{max} = 8$ and $L_{max} = 5$, L_{max} is relaxed to simplify the illustration.

The first step is to pick a starting point. According to lines 3–4, Algorithm 1, node 1 with node degree equal to one is first selected as the starting point to form the first cluster. Then node 2 is selected as the initial integrated point for the first cluster since it has the maximum node degree $\delta = 6$ among node 1's 2-hop neighbors.

When forming the cluster, node 1 has been first added into the cluster, and the three constraints are satisfied, according to line 13, Algorithm 1. Then node 2's neighboring node 7 has been added in

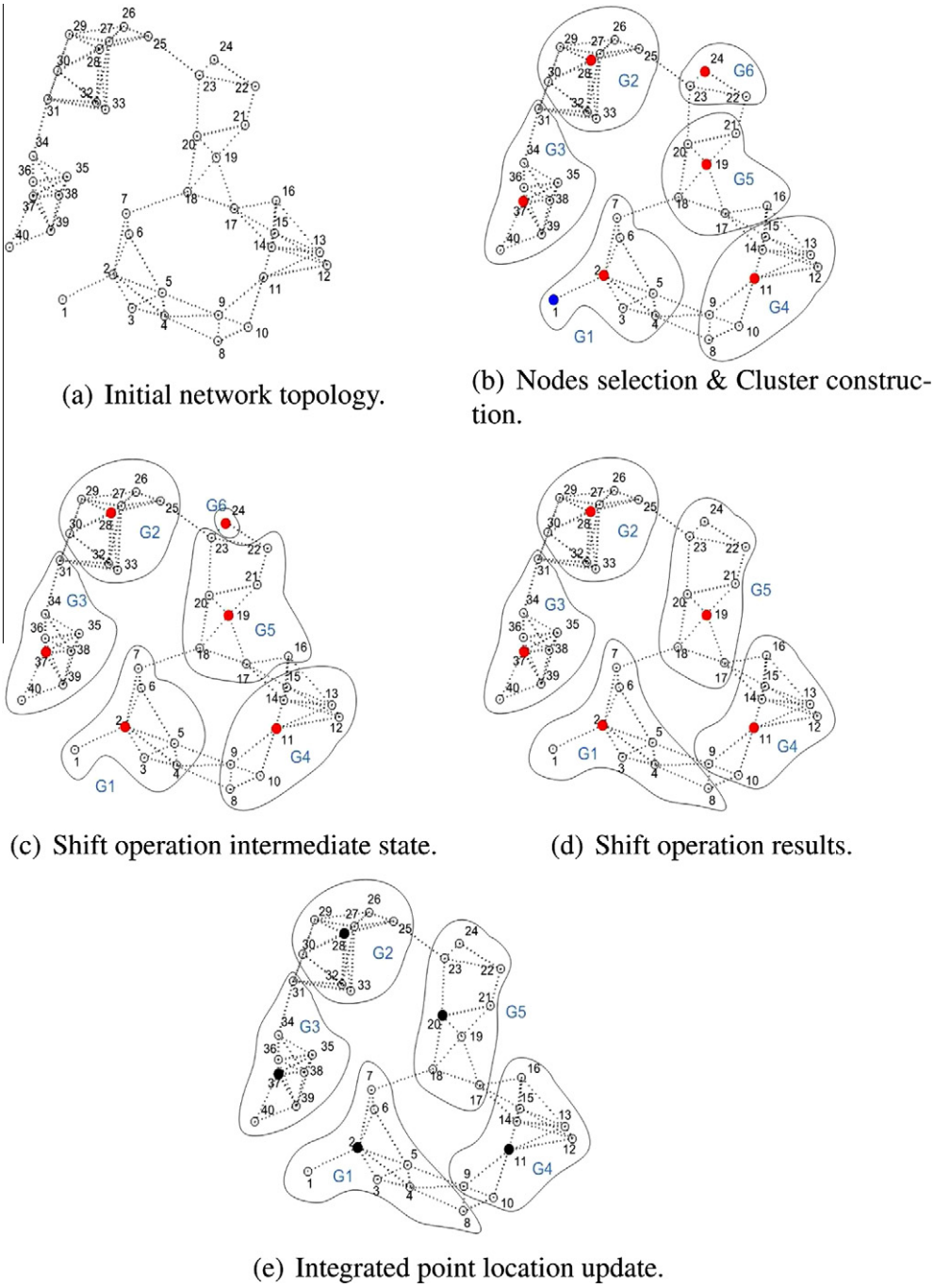


Fig. 3. Algorithm illustration.

since it has the smallest residue node degree $\delta' = 2$ and it can maintain all the constraints. Next, node 6 has been added since it is the node currently with the smallest residue node degree $\delta' = 1$, and satisfies Criterion 1 and all the constraints; then node 5, 4, and 3 have been added one by one because of the same reason. But two-hop nodes 8, 9 and 18 cannot be added into this cluster since Criterion 1 is violated for these nodes. Thus, the cluster construction has been stopped and the first cluster has been formed. Then we check whether all the nodes have been formed into clusters. If not, we repeat from the starting node selection operation again. In this round, we select the highest node 26 as the starting point according to lines 5–6, and form the next cluster based on lines 11–15. The algorithm continues to perform these two operations until all the nodes have been formed into clusters and assigned to one and only one integrated point. Eventually, 6 clusters are formed with the cluster size $C_1 = 7$, $C_2 = 8$, $C_3 = 8$, $C_4 = 8$, $C_5 = 6$, and $C_6 = 3$. Among those clusters, cluster 2, 3, and 4 are full-sized and 1, 5 and 6 are not-full-sized. This result is shown in Fig. 3(b).

Since $C_1 + C_5 + C_6 \leqslant (3-1) \cdot C_{max} = 16$, we move to the shift operation to reduce the number of clusters, specified in lines 18–25. Based on $\overline{C_2} = \frac{1}{M_2} \cdot \sum_{i=1}^{m} C_i$ from line 19, Algorithm 1, we find that each cluster should have the same number of nodes which equal to 8. Thus, we pick the smallest-sized cluster $G6$ to shift first. Based on Criterion 2, we find that node 22 and 23 have the same hop count away from their corresponding integrated point 24, have the same number of outgoing-links and the same number of

incoming-links, thus, these two nodes have been shifted along the selected shift path ($G6 \rightarrow G5$) to cluster 5 since $C_5 < C_{max}$. The result is shown in Fig. 3(c). Now node 24 in cluster 6 needs to be shifted towards cluster 1 to reduce the number of clusters and we choose the shift path as ($G6 \rightarrow G5 \rightarrow G4 \rightarrow G1$). According to Criterion 2, we shift two-hop node 8 from cluster 4 to cluster 1 rather than shift one-hop nodes 9 and 18 to reduce the average transmission delay. Following the same criterion, node 16 is shifted from cluster 5 into cluster 4 and node 24 in cluster 6 is shifted into cluster 5. Then we needs 5 clusters, which is minimum, to cover all the nodes. The result is shown in Fig. 3(d).

Finally, we update the integrated point location in each cluster to minimize the total hop count in each cluster according to line 26 in Algorithm 1, and this final result is shown in Fig. 3(e).

4.3. Algorithm complexity analysis

The S^2U algorithm begins with the starting node and its corresponding integrated point selections, which need $O(N)$ to check all the nodes for the starting nodes inside the network and $O(N)$ to identify the corresponding integrated point as the worst case running time, respectively. So the total running time is bounded by $O(N^2)$ for this operation. For each cluster construction, given the shortest path between the starting node and the integrated point, the worst case running time is $O(N)$, and $O(N^3)$ is the worst case running time for constructing one cluster including the node selection phase. Therefore, to cover all the nodes inside the network before the shift operation, the worst case running time is bounded by $O(N^4)$.

In the shift operation, for each cluster that needs to be shifted, it will take $O(N^2)$ to construct the spanning tree or shift path, and need $O(C_{max})$ to identify which node should be shifted along the shift path between the neighboring clusters; each cluster may have at most C_{max} nodes to shift, and there are totally at most N clusters, so the shift operation is bounded by $O(C_{max}^2 \cdot N^3)$. Consider we may perform integrated point location update during the shift operation, the worst case running time of this action will take $O(C_{max}^2)$. Therefore, the worst case of the shift operation is bounded by $O(C_{max}^4 \cdot N^3)$. Given C_{max}, that is a constant, $O(C_{max}^4 \cdot N^3)$ becomes to $O(N^3)$.

At the final phase, the integrated point location update is bounded by $O(N)$.

Thus, the overall upper bound of our S^2U algorithm is $O(N^4)$.

4.4. Algorithm approximation ratio

In this section, we first obtain the approximation ratio for the worst case, then consider a general case given average node degree for each node. The approximation ratio is derived based on the node cost. For the simplicity, we only analyze the node selection and cluster construction phase and omit the shift and update phase since the shift and update operations will not degrade the performance.

Suppose there is a connected network with N nodes, each node i, $i \in \{1,\ldots,N\}$ is considered as a candidate of integrated point and can form different clusters with respect to the three constraints centered at this node. In the whole network, we have a solution set S that has m clusters, $S = \{G_1, G_2, \ldots, G_m\}$ to cover all the nodes. Assign unit cost to the possible clusters in S, to find the clusters with minimum cost covering all the nodes is equivalent to find the minimum number of integrated points to cover all the nodes. We further define node cost as $n_{cj} = \frac{1}{C_i(j)}, j \in \{1,\ldots,N\}$, $C_i(j)$ denotes the cluster size of cluster i to which node j belongs. Suppose the optimal solution set is S_{opt} that has m_{opt} clusters, $S_{opt} = \{G_1^{opt}, G_2^{opt}, \ldots, G_{m_{opt}}^{opt}\}$, G_i^{opt} means cluster i in optimal solution. The optimal cost can then be equivalently represented as the summation of optimal node costs over all the nodes, that is,

$$m_{opt} = \sum_{k=1}^{m_{opt}} 1 = \sum_{k=1}^{m_{opt}} \frac{1}{C_k} \cdot C_k = \sum_{k=1}^{m_{opt}} \sum_{i \in G_k^{opt}} \frac{1}{C_k} = \sum_{j=1}^{N} n_{cj}^{opt} = \sum_{j=1}^{N} \frac{1}{C_i(j)^{opt}}, \quad j \in G_i^{opt},\ G_i^{opt} \in S_{opt} \tag{15}$$

Similarly, based on our S^2U algorithm, the solution set is S_{S^2U} with m_{S^2U} clusters. $S_{S^2U} = \{G_1^{S^2U}, G_2^{S^2U}, \ldots, G_{m_{S^2U}}^{S^2U}\}$, $G_i^{S^2U}$ means cluster i in S^2U solution. The total cost of our S^2U algorithm can be expressed as

$$m_{S^2U} = \sum_{k=1}^{m_{S^2U}} 1 = \sum_{k=1}^{m_{S^2U}} \frac{1}{C_k} \cdot C_k = \sum_{k=1}^{m_{S^2U}} \sum_{i \in G_k^{S^2U}} \frac{1}{C_k} = \sum_{j=1}^{N} n_{cj}^{S^2U} = \sum_{j=1}^{N} \frac{1}{C_i(j)^{S^2U}}, \quad j \in G_i^{S^2U},\ G_i^{S^2U} \in S_{S^2U}. \tag{16}$$

Based on the node cost, it is obtained that the lower bound of the optimal cost in cluster formation is $\frac{N}{C_{max}}$, since

$$m_{opt} \geqslant \sum_{j=1}^{N} \frac{1}{C_{max}} = \frac{N}{C_{max}}. \tag{17}$$

And the upper bound of the solution is $\frac{N}{C_{min}}$, since

$$m_{S^2U} \leqslant \sum_{j=1}^{N} \frac{1}{C_{min}} = \frac{N}{C_{min}}. \tag{18}$$

Thus the approximation ratio σ for our S^2U algorithm can be expressed as

$$\sigma \leqslant \frac{m_{S^2U}}{m_{opt}} \leqslant \frac{\frac{N}{C_{min}}}{\frac{N}{C_{max}}} = \frac{C_{max}}{C_{min}}. \tag{19}$$

Now our goal is to figure out C_{min} according to our S^2U algorithm. Imagine that we have a star topology of $C_{max} + 1$ nodes, where $C_{max} - 1$ nodes connect to the node in the center, we may have $C_{min} = 1$. That is, one integrated point covers $C_{max} - 1$ nodes forming one cluster, and the leftover one form another cluster. Thus, the approximation ratio for our algorithm becomes to $\sigma = C_{max}$.

This bound is derived from an extreme case. Now we consider a more general scenario, where we assume that the N-node connected network has average node degree d_{avg} for each node and $d_{avg} \leqslant \frac{C_{max}}{2}$. Fig. 4 shows one possible network connection which is the worst case for C_{min}. First we explain why this is the worst case for C_{min}. The integrated point u has exactly d_{avg} nodes at layer 1 (within its one hop) according to the above assumption that each node has node degree d_{avg}. There must be at least one node at each layer because of the network connectivity. Since the cluster is constructed layer by layer and centered at integrated point u (layer 0), the more nodes at each layer, the larger cluster size for the cluster. Layer i, $i > 1$ can have more nodes compared with Fig. 4, but we want to figure out the least number of nodes for each layer in order to obtain the minimum value of C_{min} that gives the upper bound of σ. Layer 2 can have only one node with $d_{avg} - 1$ links connecting with layer 1 nodes and one link connecting with layer 3 node. Suppose there is one node at layer 3 with one link connecting with layer 2 node and $d_{avg} - 1$ connected nodes may locate at layer 2, 3 and 4. But we can treat these $d_{avg} - 1$ nodes locating at layer 4. Thus, layer 3 can have only one node and layer 4 can have only $d_{avg} - 1$ nodes. One may challenge that if we put 2 nodes at layer 3, then we can only put 2 nodes at layer 4, which is less that 3 nodes at layer 4. For this case, we obtain that the extra node in layer 3 comes from one of the nodes in layer 4, making layer 4 reducing 1 node. But the total number of nodes from layer 3 and 4 are the same. Similarly, layer 5 has one node, layer 6 has one

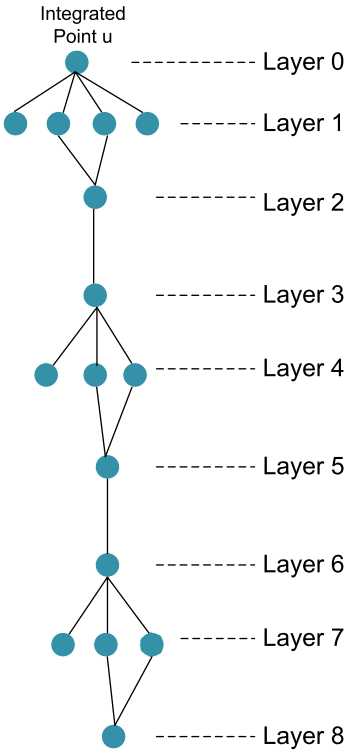


Fig. 4. Example of network topology.

node, and layer 7 has $d_{avg}-1$ nodes, etc. These are the minimum number of nodes for each layer if we denote the integrated point as layer 0. When we have this structured topology, we analyze C_{min} for different cases.

- The hop count constraint R_h is large enough. In this case, the construction of each cluster will stop when Criterion 1 is violated. This means that $C_i = C_{max} - \delta'$ when cluster construction stops. Since $\delta' \leqslant d_{avg} - 1$, we have $C_i \geqslant C_{max} - d_{avg} + 1$. So we only need to consider $C_i = C_{max} - d_{avg} + 1$. We relax the relay load constraints L_{max} first and consider the following case:
  1. Cluster construction stop at layer 1 $C_i = 1 + d_{avg} = C_{max} - d_{avg} + 1$, we have $d_{avg} = \frac{C_{max}}{2}$, then $C_i = \frac{C_{max}}{2} + 1$.
  2. Cluster construction stop at layer 2 $C_i = 1 + d_{avg} + 1 = C_{max} - d_{avg} + 1$, we have $d_{avg} = \frac{C_{max}-1}{2}$, then $C_i = \frac{C_{max}}{2} + \frac{3}{2}$.
  3. Cluster construction stop at layer r, $r \geqslant 3$ $C_i = \frac{r+1}{3}(d_{avg}+1) + 1 = C_{max} - d_{avg} + 1$, we have $d_{avg} = \frac{3}{r+4}(C_{max} - \frac{r+1}{3})$, then $C_i = \frac{r+1}{r+4}(C_{max} - \frac{r+1}{3}) + \frac{r+4}{3}$.

From the above cases we conclude that, the more the layers have been covered, the smaller the average degree d_{avg} is, and the larger the C_i will be. Thus, when $r = R_h$, we have the largest C_i where $Ci = \frac{R_h+1}{R_h+4}(C_{max} - \frac{R_h+1}{3}) + \frac{R_h+4}{3}$. Note that, we assume the cluster includes all the nodes at layer r when the construction of cluster stops in the above cases. If the cluster only has some of the nodes at layer i, it will induce smaller d_{avg}, and consequently larger C_i. Therefore, the assumption is reasonable when we analyze the smallest C_i for each case and the minimum value for C_{min} is $\frac{C_{max}}{2} + 1$ when the hop count constraints R_h is large enough.

Now we consider the relay load constraints L_{max}. We know that if the one-hop nodes satisfy the relay load constraints, then all the nodes in the cluster will satisfy it. So according to Fig. 4, C_i becomes to.

1. $C_i = \frac{C_{max}}{2} + 1$ for $r = 1$
2. $C_i = \frac{C_{max}}{2} + \frac{3}{2}$ for $r = 2$
3. $C_i = d_{avg} + 1 + L_{max}$ for $r \geqslant 3$

since in Case (1) and (2), the relay load constraints are always satisfied. We know that the larger the hop count is, the smaller the d_{avg} will be, so $C_{min} = \frac{3}{R_h+4}(C_{max} - \frac{R_h+1}{3}) + 1 + L_{max}$ when $d_{avg} = \frac{3}{R_h+4}(C_{max} - \frac{R_h+1}{3})$.

Therefore, for R_h large enough, $C_{min} = \min\{\frac{C_{max}}{2} + 1, \frac{3}{R_h+4}(C_{max} - \frac{R_h+1}{3}) + 1 + L_{max}$, and $\sigma = \frac{C_{max}}{\min\{\frac{C_{max}}{2}+1, \frac{3}{R_h+4}(C_{max}-\frac{R_h+1}{3})+1+L_{max}}$. For example, if we set $C_{max} = 30$, $R_h = 10$ and $L_{max} = 5$, then $\sigma = 2.577$.

- The hop count constraint R_h is quite small. In this case, the cluster construction will be stopped when the hop count constraint R_h is violated. Thus, $C_i = \frac{R_h+1}{3}(d_{avg}+1) + 1$. When we consider the relay load L_{max} constraint, C_{min} becomes to $C_{min} = d_{avg}^{min} + 1 + L_{max}$, and $\sigma = \frac{C_{max}}{d_{avg}+1+L_{max}}$. If we set $C_{max} = 30$, $R_h = 10$, $L_{max} = 5$ and $d_{avg} = 10$, then $\sigma = 1.875$. From this result we can see that the smaller the d_{avg} is, the larger the gap between the S^2U algorithm and optimal results will be. But, given a network, the d_{avg} is determined and will not be quite small. So the approximation ratio will not be quite large.

Now we summarize our analysis of the approximation ratio below:

- The upper bound of the approximation ratio is $\sigma = C_{max}$ for all the possible situations.
- When we assume a network has average node degree d_{avg} and $d_{avg} \leqslant \frac{C_{max}}{2}$, the approximation ratio becomes:
  1. When R_h is large enough, $\sigma = \frac{C_{max}}{\min\{\frac{C_{max}}{2}+1, \frac{3}{R_h+4}(C_{max}-\frac{R_h+1}{3})+1+L_{max}\}}$.
  2. When R_h is quite small, $\sigma = \frac{C_{max}}{d_{avg}^{min}+1+L_{max}}$.

We observe that when we have known all the possible combinations of clusters, our placement problem is equivalent to the well-known *Set Cover* problem. In [22,23], the Greedy algorithm is used to solve the same *Set Cover* problem and the approximation ratio of the Greedy algorithm is upper bounded by $1 + \log(C_{max})$ where C_{max} is largest cluster size for a given network. It seems that the Greedy algorithm looks better than our S^2U algorithm in terms of the worst case approximation ratio, but the shift operation of our S^2U algorithm can further improve our S^2U algorithm performance, though it's difficult to analyze the shift operation. Furthermore, the extensive simulation results show that the S^2U algorithm always generates less number of clusters compared with the Greedy algorithm.

5. Performance evaluation and comparison

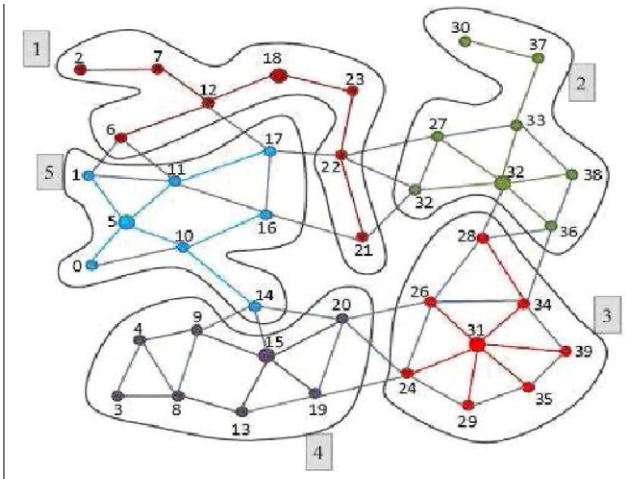
In this section, we evaluate the performance of our S^2U algorithm, provide extensively numerical results to show the effective and efficiency of our propose algorithm.

we use the following three measurements to evaluate our S^2U algorithm:

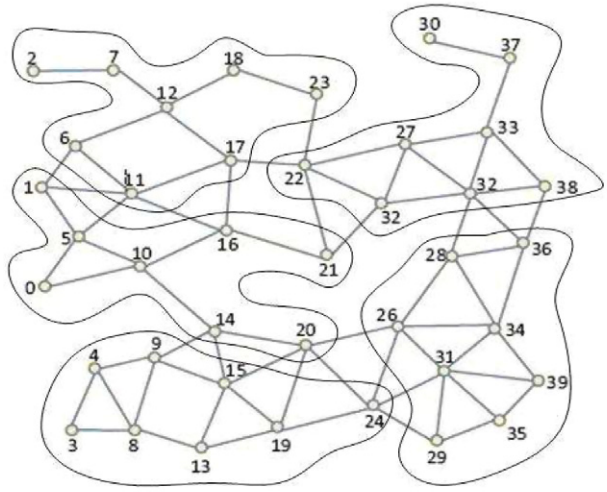
1. Total number of clusters M.
2. Average hop count $\bar{h}$ for a node, where $\bar{h} = \frac{H}{N-M}$. H is the summation of hop count between every node and its corresponding integrated point over all of the clusters.
3. Load variance $var(C)$ among all the clusters, where $var(C) = \frac{1}{M} \cdot \sum_{i=1}^{M}(C_i - \overline{C})^2$ and $\overline{C} = \frac{N}{M}$.

5.1. Comparison with SMS based on a concrete example

First, we use the example in Drabu's work [20] (SMS) to compare the results between ours and theirs with $R_h = 3$, $C_{max} = 8$ and $L_{max} = 5$. Fig. 5(a) and (b) show the results using SMS algorithm and our S^2U algorithm, respectively. The comparison in terms of



(a) SMS



(b) S^2U

Fig. 5. Comparison with SMS.

three measurements obtained from Fig. 5 are listed in Table 2. From this table we can see that compared with SMS, our approach reduces the average hop count, though the number of clusters and load variance are the same. The load variances are zero since all the 40 nodes in the network are connected and the constructed clusters are all full-sized reaching the cluster capacity. This example is just an concrete example used in [20], we will illustrate more numerical results in the following subsection.

5.2. Comparison with greedy and SMS

Since the Greedy algorithm [16] is the classic algorithm and the SMS algorithm is shown to be the most effective algorithm in the related work, now we compare our S^2U algorithm with Greedy and SMS in general case.

Table 2
Measurements from Fig. 5.

	M	$\bar{h}$	$Var(C)$
SMS	5	1.51	0
S^2U	5	1.43	0

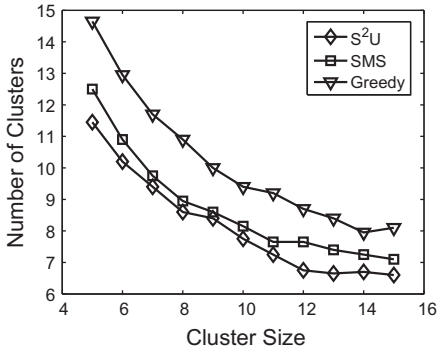
We first implement the three algorithms for a 50-node connected network with 10 different network topologies randomly generated. We set $C_{max} = 10$, $R_h = 3$ and $L_{max} = 5$. The results are shown in Table 3. From this table we can see that our algorithm always generates the minimum number of clusters. In addition, it performs better in terms of the average hop count and load variance than the other two algorithms when they generate the same number of clusters.

Then we investigate how each constraint would affect the performance. For each constraint, we generate 20 different network topologies for a 50-node connected network randomly, and the results presented are the average.

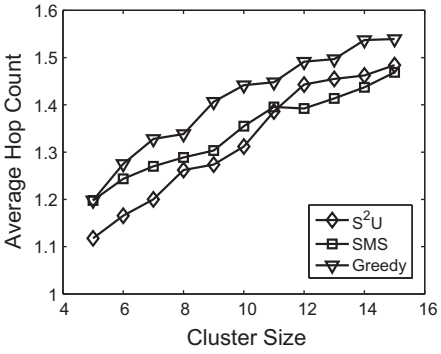
First we fix the two constraints hop count $R_h = 3$, relay load $L_{max} = 5$, and vary the cluster size from 5 to 15. From Fig. 6(a) we can see that, as the cluster size becomes larger, the total number of clusters generated is reduced, and our algorithm performs the best. Fig. 6(b) shows the average hop count increases as the cluster size increases. The three algorithms achieve similar average hop count. Fig. 6(c) shows the cluster load variance based on the cluster size variation. When the cluster size is small, our algorithm does not have the best result. The reason is that our algorithm has fewer chance to get more different sets of nodes because of the limited cluster size. As we can see, when the cluster size becomes larger,

Table 3
Measurements for the three algorithms with $N = 50$, $R_h = 3$, $C_{max} = 10$, $L_{max} = 5$.

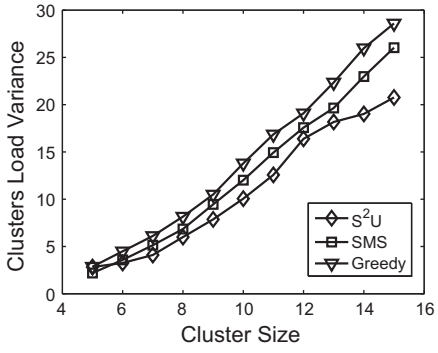
	S^2U	SMS	Greedy
Topology 1			
M	7	7	9
$\bar{h}$	1.30	1.34	1.39
$var(C)$	7.84	9.27	16.25
Topology 2			
M	7	7	7
$\bar{h}$	1.26	1.26	1.49
$var(C)$	9.55	12.12	10.69
Topology 3			
M	9	9	10
$\bar{h}$	1.35	1.44	1.43
$var(C)$	10.91	13.36	14.20
Topology 4			
M	7	7	8
$\bar{h}$	1.24	1.34	1.40
$var(C)$	6.12	10.98	14.19
Topology 5			
M	7	7	10
$\bar{h}$	1.24	1.24	1.40
$var(C)$	8.98	11.21	16.80
Topology 6			
M	9	9	10
$\bar{h}$	1.21	1.28	1.52
$var(C)$	10.91	13.36	14.20
Topology 7			
M	8	8	9
$\bar{h}$	1.31	1.38	1.51
$var(C)$	11.69	13.94	16.69
Topology 8			
M	9	9	11
$\bar{h}$	1.34	1.44	1.56
$var(C)$	10.69	14.02	12.79
Topology 9			
M	8	8	9
$\bar{h}$	1.24	1.50	1.39
$var(C)$	13.69	17.44	16.69
Topology 10			
M	8	9	8
$\bar{h}$	1.40	1.41	1.50
$var(C)$	12.44	10.47	12.94



(a) Total number of clusters.

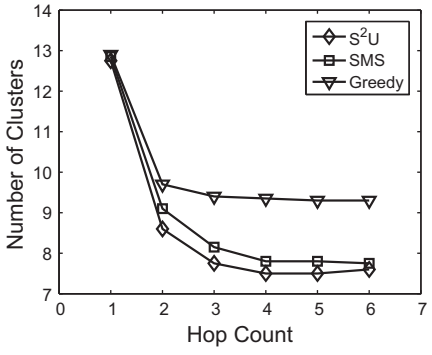


(b) Average hop count.

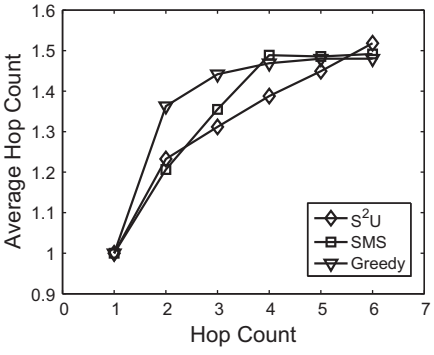


(c) Load variance.

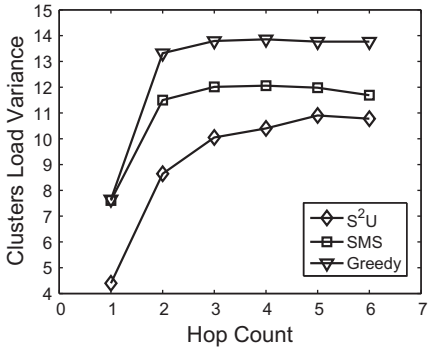
Fig. 6. $N = 50$, $R_h = 3$, $L_{max} = 5$, cluster size C_{max} varies from 5 to 15.



(a) Total number of clusters.



(b) Average hop count.



(c) Load variance.

Fig. 7. $N = 50$, $C_{max} = 10$, $L_{max} = 5$, hop count R_h varies from 1 to 6.

our algorithm always performs better than the other two algorithms in terms of the load variance.

In Fig. 7 we fix C_{max} = 10, L_{max} = 5, and vary the hop count from 1 to 6. Fig. 7(a) shows that our algorithm always performs the best in terms of the total number of clusters. As the hop count increases, the total number of clusters generated by the three different algorithms decrease at the same time, and will level at some value since the cluster size will become to the limiting parameter when the hop count is large enough. Fig. 7(b) shows the average hop count based on the hop count variation. The three algorithms perform very closely. Fig. 7(c) shows the cluster load variance based on the hop count variation. We can see that our algorithm always performs the best in terms of the load variance whatever the hop count is large or small, as long as the cluster size is large enough.

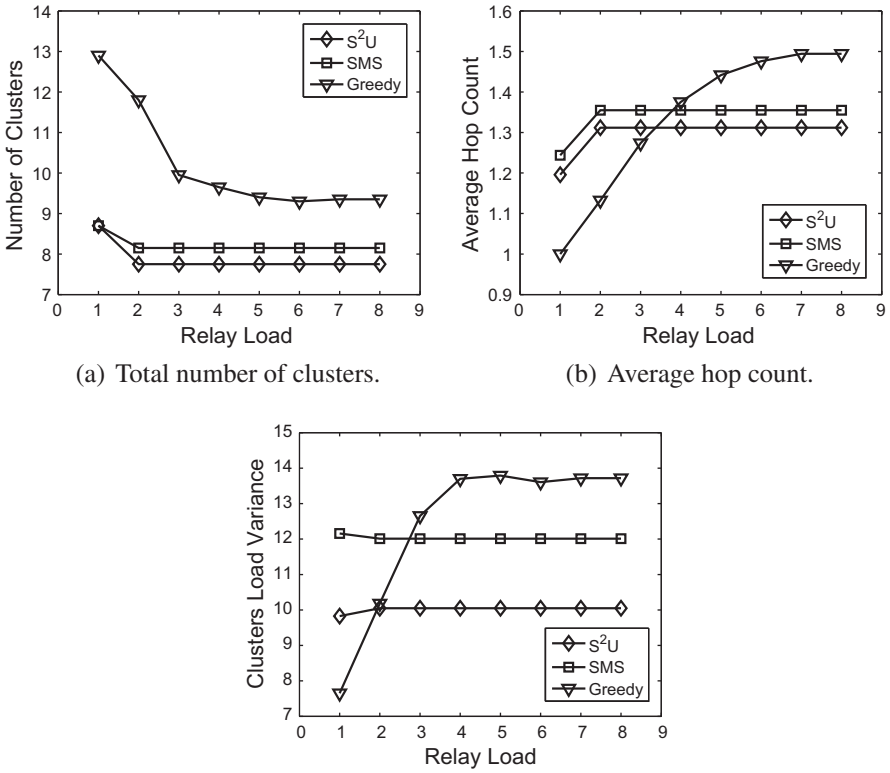
In Fig. 8 we fix C_{max} = 10, R_h = 3, and vary the relay load from 1 to 8. From Fig. 8(a) we can see that as the relay load increases, the total number of clusters generated by the three different algorithms decrease at the same time, and will level at some value since the cluster size will become to the limiting parameter when the relay load is large enough. Also, Fig. 8(a) shows that our algorithm always performs the best in terms of the total number of clusters when the relay load is larger than 1. When relay load is equal to 1, SMS algorithm performs a little bit better than ours since the relay load is too small, our algorithm has smaller chance to get more different sets of nodes, and this induces to more clusters compared with SMS. Fig. 8(b) shows the average hop count based on the relay load variation. Our S^2U algorithm performs quite close to SMS, but a little different compared with greedy algorithm. In Fig. 8(c), when the relay load is small, Greedy algorithm performs the best. As the relay load becomes larger and larger, our algorithm performs the best, following by the SMS algorithm, and the Greedy algorithm performs the worst. This is because, the Greedy algorithm always picks the largest node set, which leads to bigger variance. On the other hand, our algorithm will have more chance to get more different sets of nodes when the relay load becomes larger, resulting in smaller variance.

Based on the results obtained from Figs. 6–8, we summarize here: our algorithm is designed to minimize the number of total clusters as well as balancing the load among the clusters. Thus, for the cases studied, our algorithm performs the best in terms of the total number of clusters and the load variance. Since the design does not consider the hop count that much, our algorithm usually does not perform the best in terms of the average hop count, but our results are comparable to Greedy and SMS results. The total number of clusters is mainly determined by the cluster size when the hop count and relay load is large enough, and the relay load has more effect on the load variance than the hop count. The reasons that our algorithm always performs better than the other two algorithms are: (1) we form clusters starting from the edge of the network and then go to the network center rather than start from the node with the maximum degree used in SMS; (2) we construct and shift the cluster based on Criterion 1 and Criterion 2, and do not always add neighbor nodes to reach the maximum cluster size; (3) we update the integrated point location at the final stage to minimize the average hop count and balance the load.

We only consider 50 nodes for the network size, but we vary the cluster size, hop count and relay load in the simulation. Thus, if we consider different network size, we will obtain similar results because of the C_{max}, R_h and L_{max} considerations.

5.3. Comparison with optimal results

Now we compare our algorithm with the optimal results for a 50-node connected network with 10 different network topologies randomly generated and the results shown are the average. The optimal results are obtained using the CPLEX. We fix two of the



(a) Total number of clusters.

(b) Average hop count.

(c) Load variance.

Fig. 8. N = 50, C_{max} = 10, R_h = 3, relay load L_{max} varies from 1 to 8.

three constraints and vary the other one. From Figs. 9–11 we can see that our algorithm performs a little bit worse than the optimal results. The maximum differences are 11.1%, 1.4%, 6.6% and the average differences are 4.6%, 0.8%, 2.6% in terms of the number of clusters when vary hop count, relay load and cluster size, respectively. After this, we run our S^2U algorithm in matlab and the *ILP* in CPLEX to see the difference of the running time. It takes 2.03 s, 3.05 s, 6.88 s and 13.68 s in matlab for 50 nodes, 75 nodes, 100 nodes and 125 nodes, respectively, with considered constraints randomly selected; but it takes 63.31 s, 153.49 s, 3.27 h and 78.04 h in CPLEX for 50 nodes, 75 nodes, 100 nodes and 125 nodes, respectively. We can see that the running time grows polynomially when the network size increases for our S^2U algorithm, but for the optimal results the running time grows exponentially when the network size increases.

Note that the size of the gap distance between our S^2U algorithm and the optimal results is actually determined by the design of the shift operation in S^2U. The more criteria we include in the shift operation, the smaller gap we will obtain. Theoretically we can design a perfect shift operation that makes our S^2U algorithm reaching to the optimum, but the price is the algorithm complexity.

5.4. More considerations

In this subsection, we investigate our S^2U algorithm in terms of the total hop count when consider peer-to-peer communications,

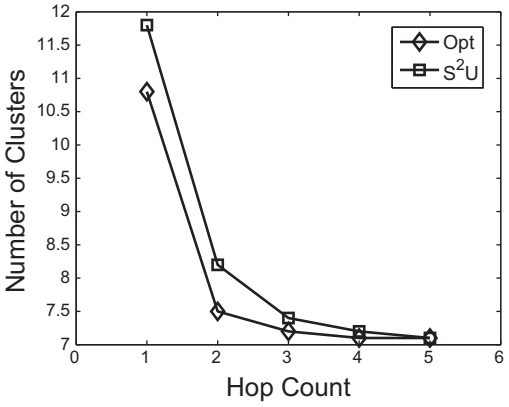


Fig. 9. Total number of clusters with $N = 50$, $C_{max} = 10$, $L_{max} = 5$, hop count R_h varied from 1 to 5.

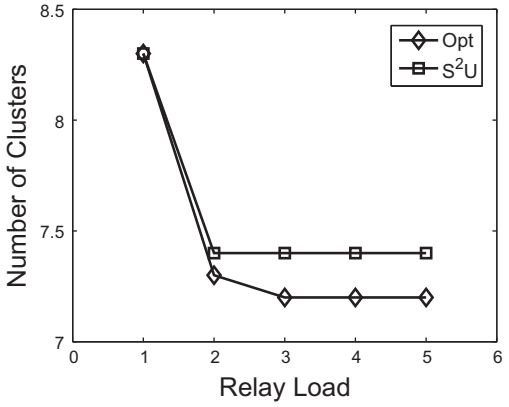


Fig. 10. Total number of clusters with $N = 50$, $C_{max} = 10$, $R_h = 3$, relay load L_{max} varied from 1 to 5.

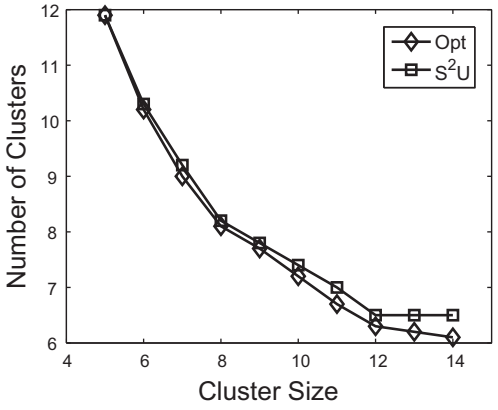


Fig. 11. Total number of clusters with $N = 50$, $R_h = 3$, $L_{max} = 5$, cluster size C_{max} varied from 5 to 14.

and evaluate our algorithm under more realistic factors, such as non-uniform channel capacities and non-uniform traffic pattern.

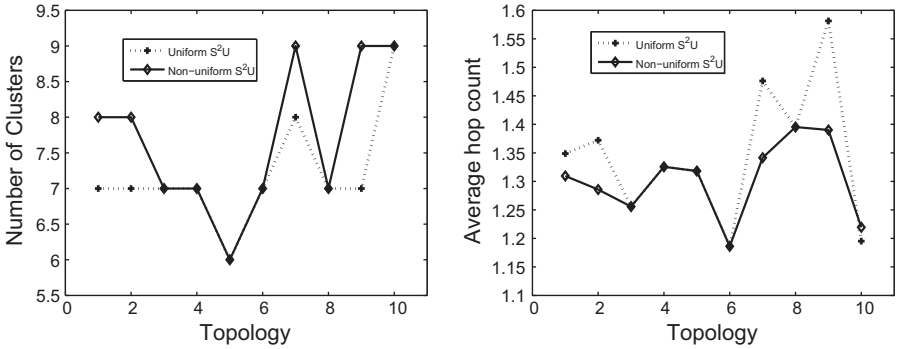
First, we compare our S^2U algorithm with the Tabu algorithm [10] based on 120 nodes. The total hop counts based on these two algorithms are shown in Table 4. From this table we can see that using the same number of integrated points, the total hop counts based on our algorithm is smaller than the tabu algorithm, which means our algorithm has better performance in terms of the system throughput based on the assumptions in [10]. From another point of view, we can see that when the two algorithms result in the similar total number of hop counts, our algorithm has smaller number of integrated points, which means our algorithm has smaller cost when set up the infrastructure (integrated point deployment).

Then, we compare the network performance between uniform and non-uniform channel capacities (relay load) among node pairs based on our S^2U algorithm under 10 random generated network topologies and the results are shown in Fig. 12(a), (b), and (c). From these figures we can see that our S^2U algorithm generates smaller or the same number of integrated points when consider uniform channel capacity. This is because nodes with smaller channel capacity or relay load in the non-uniform case can accommodate fewer number of descendant nodes, and for some topologies, some integrated points support fewer nodes and resulting in more clusters (integrated points). For the average hop counts and load variance, the results from the two different scenarios have similar performance and they are determined by the network topology and the distribution of channel capacity.

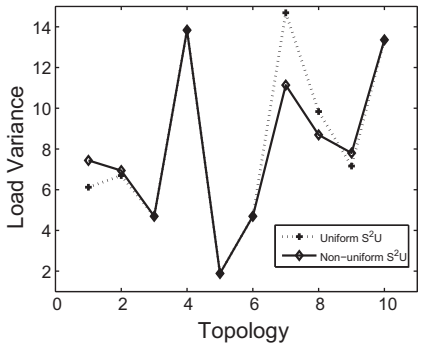
At last, we compare the network performance under uniform traffic pattern and non-uniform traffic pattern based on our S^2U algorithm with 10 random generated network topologies. The results are shown in Fig. 13(a), (b), and (c). We can see that the network performance in terms of the number of integrated points, average hop count, and load variance in the two scenarios are quite similar. In some cases the uniform traffic scenario has better performance, and in some cases the non-uniform traffic scenario has better performance, since different network topology and traffic pattern affect the network performance.

Table 4
Compare S^2U with Tabu under 120 nodes.

	6 ONUs	7 ONUs	8 ONUs
Tabu	4.87×10^4	4.46×10^4	4.11×10^4
S^2U	4.49×10^4	4.12×10^4	3.85×10^4

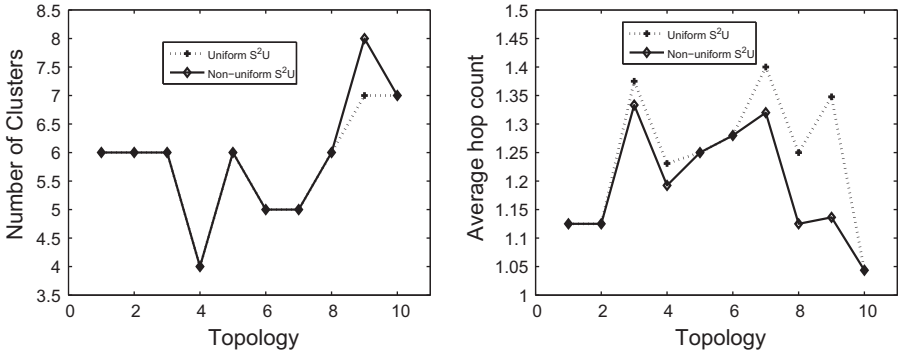


(a) Total number of clusters. (b) Average hop count.

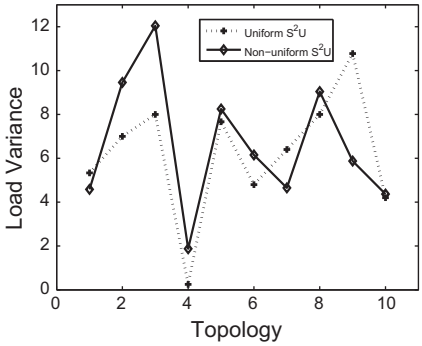


(c) Load variance.

Fig. 12. $N = 50$, $R_h = 3$, $C_{max} = 10$, relay load L_{max} set to 1 or 4.



(a) Total number of clusters. (b) Average hop count.



(c) Load variance.

Fig. 13. $N = 30$, $R_h = 3$, $C_{max} = 8$, traffic load for a node set to 0.5, 1, or 1.5.

6. Conclusion

In this paper, we solve the integrated points placement problem in the hybrid optical-wireless access networks while meeting several major constraints. Based on our system model, we first formulate this problem as an optimization problem, and this problem can be solved by the ILP. However, this is not suitable when the network size becomes large because of the scalability problem. Thus, we propose our S^2U (Selection-Shift-Update) algorithm that can achieve the near-optimal solution, with the number of clusters needed minimized. Furthermore, we give a detailed analysis about the algorithm complexity of the proposed S^2U algorithm, and obtain an upper bound of the approximation ratio of our algorithm compared with the optimal results. The numerical results have been compared with the current literature, and verify that our proposed algorithm generates fewer numbers of clusters in average while reducing average transmission delay (average hop count), balancing load for each cluster, and consequently saving the network deployment cost and improving the total network performance.

Future work include the investigation of integrated points placement problem while taking into account more considerations, such as channel interference among end users, design of distributed scheduling algorithm, and efficient resource allocation schemes for the hybrid system.

Acknowledgment

This work was partly supported by NSF under grant CNS 0619693.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at doi:10.1016/j.comcom.2011.02.005.

References

[1] W-T. Shaw, S-W. Wong, N. Cheng, Hybrid architecture and integrated routing in a scalable optical-wireless access network, J. Lightwave Technol. 25 (2007) 3443–3451.
[2] L.G. Kazovsky, N. Cheng, W-T. Shaw, CROWN-converged optical and wireless networks: network architecture and routing algorithms, ICTON 1 (2007) 266–269.
[3] S. A'issa, M. Maier, Towards seamless fiber-wireless (FiWi) access networks: convergence and challenges, ICTON (2007) 1–6.
[4] G. Shen, R.S. Tucker, C-J. Chae, Fixed mobile convergence architectures for broadband access: integration of EPON and WiMAX, IEEE Commun. Mag. 45 (8) (2007) 44–50.
[5] N. Ghazisaidi, M. Maier, C. Assi, Fiber-wireless (FiWi) access networks: a survey, IEEE Commun. Mag. 47 (2) (2009) 160–167.
[6] S. Sarkar, S. Dixit, B. Mukherjee, Hybrid wireless-optical broadband-access network (WOBAN): a review of relevant challenges, J. Lightwave Technol. 25 (11) (2007) 3329–3340.
[7] J. Jiang, X. Zhang, Research on EPON of broadband access technology and broadband network deployment, ICACTE 3 (2010). V3-148-V3-152.
[8] B. Jung, J. Choi, Y.-T. Han, M.-G. Kim, M. Kang, Centralized scheduling mechanism for enhanced end-to-end delay and QoS support in integrated architecture of EPON and WiMAX, J. Lightwave Technol. 28 (16) (2010) 2277–2288.
[9] Y. Liu, C. Zhou, Y. Cheng, Integrated BS/ONU placement in hybrid EPON-WiMAX access networks, in: Proceedings of IEEE GLOBECOM, 2009, pp. 1–6.
[10] Z. Zheng, J. Wang, X. Wang, ONU placement in fiber-wireless (FiWi) networks considering peer-to-peer communications, in: Procedings of IEEE GLOBECOM, 2009, pp. 1–7.
[11] Z. Vincze, R. Vida, A. Vidacs, Deploying multiple sinks in multi-hop wireless sensor networks, IEEE ICPS (2007) 55–63.
[12] E.I. Oyman, C. Ersoy, Multiple sink network design problem in large scale wireless sensor networks, IEEE ICC 6 (2004) 3663–3667.
[13] L. Yang, Determining sink node locations in wireless sensor networks, IEEE SMC 4 (2006) 3400–3404.
[14] S. Sarkar, B. Mukherjee, S. Dixit, Optimum placement of multiple optical network units (ONUs) in optical-wireless hybrid access networks, OFC/NFOEC, Mar 2006.
[15] S. Sarkar, B. Mukherjee, S. Dixit, Towards global optimization of multiple ONU placement in hybrid optical-wireless broadband access networks, COIN-NGNCON (2006) 65–67.
[16] Y. Bejerano, Efficient integration of multi-hop wireless and wired networks with QoS constraints, IEEE/ACM Trans. Networks 12 (6) (2004) 1064–1078.
[17] R. Chandra, L. Qiu, K. Jain, M. Mahdian, Optimizing the placement of integration points in multi-hop wireless networks, in: Proceedings of IEEE ICNP, Oct 2004.
[18] B. Aoun, R. Boutaba, Y. Iraqi, G. Kenward, Gateway placement optimization in wireless mesh networks with QoS constraints, IEEE J. Select. Areas Commun. 24 (11) (2006) 2127–2136.
[19] B. He, B. Xie, D.P. Agrawal, Optimizing the internet gateway deployment in a wireless mesh network, IEEE MASS (2007) 1–9.
[20] Y. Drabu, H. Peyravi, Gateway placement with QoS constraints in wireless mesh networks, IIEEE ICN (2008) 46–51.
[21] S. Sarkar, H-H. Yen, S. Dixit, B. Mukherjee, Hybrid wireless-optical broadband access network (WOBAN): network planning and setup, IEEE J. Select. Areas Commun. 26 (6) (2008) 12–21.
[22] V. Chvatal, A greedy heuristic for the set-covering problem, Mathematics of Operations Research 4 (3) (1979) 233–235.
[23] D.S. Hochba, Approximation Algorithms for NP-Hard Problems, PWS Publishing Company, 1997.
[24] S. Sarkar, H-H. Yen, S. Dixit, B. Mukherjee, A mixed integer programming model for optimum placement of base stations and optical network units in a hybrid wireless-optical broadband access network (WOBAN), IEEE WCNC (2007) 3907–3911.
[25] S. Sarkar, H-H. Yen, S. Dixit, B. Mukherjee, Hybrid wireless-optical broadband access network (WOBAN): network planning using Lagrangean relaxation, IEEE/ACM Trans. Networking 17 (4) (2009) 1094–1105.