Facility Location Selection Using Community-Based Single Swap: A Case Study

Rixin Xu[1], Zijian Zhang[1](✉), Jiamou Liu[2](✉), Nathan Situ[2], and Jun Ho Jin[2]

[1] Beijing Institute of Technology Beijing, Beijing, China
{xurixin,zhangzijian}@bit.edu.cn
[2] University of Auckland, Auckland, New Zealand
jiamou.liu@auckland.ac.nz

Abstract. This paper focuses on the uncapacitated k-median facility location problem, which asks to locate k facilities in a network that minimize the total routing time, taking into account the constraints of nodes that are able to serve as servers and clients, as well as the level of demand in each client node. This problem is important in a wide range of applications from operation research to mobile ad-hoc networks. Existing algorithms for this problem often lead to high computational costs when the underlying network is very large, or when the number k of required facilities is very large. We aim to improve existing algorithms by taking into considerations of the community structures of the underlying network. More specifically, we extend the strategy of *local search with single swap* with a community detection algorithm. As a real-world case study, we analyze in detail Auckland North Shore spatial networks with varying distance threshold and compare the algorithms on these networks. The results show that our algorithm significantly reduces running time while producing equally optimal results.

Keywords: Facility location · k-median problem
Community structures · Spatial networks · Auckland Open Data
Single swap algorithm

1 Introduction

Facility location problem concerns with the deployment of decentralized service across a network of interconnected nodes. The goal is to choose a set of nodes to host facilities which lead to a minimized overall cost. Consider, as an example, a *wireless sensor network* [20], which consists of self-organizing sensors that communicate through wireless transmission without a pre-designated infrastructure. It is often more cost-effective to designate nodes in the networks to host

Z. Zhang—This work is partially supported by China National Key Research and Development Program No. 2016YFB0800301.

servers (or "hubs") that collect and aggregate sensory data [14]. The costs associated with this scheme include resources consumed by the servers, as well as the communication costs between sensors to servers. To minimize costs, a challenge lies in the selection of server locations so that every sensor reaches a nearby server while keeping the number of servers reasonable. This would mean an even distribution of workload carried out by the servers and thus reliable network performance. One needs to deal with two central questions: how many servers should there be and on which nodes should servers be placed?

This paper focused on (uncapacitated) *k-median facility location problem.* Abstractly, complex networks such as communication, physical or social networks are characterized by interactions between its nodes; channels of interactions are represented as edges and are typically weighted to reflect distance or strength of the connection. Nodes are classified into ones that are resource-rich and resource-poor, and facilities (or "servers") may be placed on resource-rich nodes. When a facility is opened on a node, the node becomes a *server nodes* and it may provide services to others. A non-server node is also called a *client node*, and it is designated a particular server node to communicate with. As client nodes may have different levels of demand for service from their designated server, the communication cost for a client is captured by (1) the distance between the client and its nearest server; and (2) the demand level of this client. The problem takes a parameter k and seeks for k nodes to act as server nodes that minimize total communication costs among all client nodes.

The facility location problem has many potential applications in wireless communication. Apart from the application in sensor networks discussed above, consider, as a second application, an urban *cellular network.* An important problem of cellular networks is to reduce energy consumption through the use of base stations with low transceiver power budget. The main challenge here is to place such base stations in appropriate locations that meet the QoS requirement of every user while minimizing energy consumption. This challenge can be rephrased in terms of facility location problem, where the facilities correspond to base stations and the costs correspond to energy consumption [16]. A third potential application involves information-centric networking (ICN), which transmits data directly between devices without the need for a pre-existing infrastructure [3]. In particular, *opportunistic networks* present as an efficient, scalable and robust scheme to delivery contents. Here, communications are supported through mobile ad hoc networks (MANETs) and contacts are opportunistically established between devices [18]. To facilitate reliable delivery of contents, one may deploy a number of "hub storage" of user contents. These storages can be selected via a facility location problem: the underlying network in this scenario contains social contact patterns among users, i.e., the frequency of contacts between people implies the potential to deliver message between their devices. The facilities correspond to storage hubs, and the cost is associated with how easy or likely a node may reach a facility via multi-hop paths [28].

This paper's main aim is three-fold. Firstly, due to the high computational complexity of facility location problem, attention has mostly been concentrated on designing heuristics to approximate optimal solutions of the facility location problem. Classical solutions, such as the *reverse greedy algorithm* and the *single-swap algorithm* incur a large computation cost when applied to complex networks or when a lot of facilities are required to be placed on the network [5,10]. Thus the first goal of the paper is to design more efficient heuristic to solve the problem. Secondly, community structure is a prevalent property of complex networks in the real world and has been intensively studied in the context of large and complex networks. Intuition tells us that facilities should be placed to serve local communities. However, to the authors' knowledge, no work has so far emphasized on the potential that community structure may support the selection of facility locations within a network; The goal of the research is to explore this potential. Thirdly, by processing government open-access GIS dataset, we obtain geospatial networks representing Auckland's North Shore, a major urban region in New Zealand. Nodes in the network correspond to land zones and edges represent geodesic proximity. We investigate the facility location problem in the context of this region as a real-world case study. The findings would potentially provide us more insights on the urban topology of the city.

Paper organization. Section 2 discusses background and related works. Section 3 introduces the facility location problem, the reverse greedy and single swap algorithm. Section 4 presents our two community-based methods: the community select and community swap algorithms. Section 5 discusses our case study on North Shore dataset and experimental results obtained on the spatial networks. Section 6 concludes with future works.

2 Background and Related Works

2.1 Main Themes and Background

Selecting facility locations to effectively serve a region has been an important problem in operation research [4,11,17]. There are two main versions of the problem that are closely related: Firstly, the *classical facility location problem* seeks not only to decide on the location but also the number of facilities to be placed. Secondly, the *k-median facility location problem* assumes that the number k of facilities is given as an input, and lifts the restriction on the coverage of each server. The focus of this paper is on the k-median facility location problem.

Real-world networks are seldom uniformly distributed, but rather, exhibit distinguishing patterns such as scale-freeness (i.e., power-law degree distribution) and small-worldness (i.e., short average path length and high clustering coefficient) [6,25,29,30]. A real-world network is typically composed of a collection of densely connected regions, which are sparsely connected between themselves [23,26]. Detecting these dense regions allows us to develop a macro-level topology of the network, where each such dense region is called a *community.* Intuition tells us that it is reasonable to place servers at the center of communities so that they dedicate their services towards their own communities.

Spatial analysis studies geometric and topological properties of physical locations through geographical information systems [22,24,31]. The field has been widely applied to urban planning [32], transportation [7], and telecommunication engineering [19]. Auckland North Shore (formerly North Shore City) comprises of the 4th largest urban area in New Zealand and it has become an integrated part of Auckland since 2010. Auckland City Council has published data of North Shore which contains detailed accounts of geographic information. This allows us to extract distance-based networks of land areas in the region and perform analytics over these networks.

2.2 Related Works

Here we briefly survey important algorithms for solving the facility location problem. Chaudhuri et al. in [9] discussed the k-median facility location problem by exploiting the notion of *distance-d dominating sets* [12]. This approach ignores the varying service requirements from nodes, furthermore, the identification of dominating sets is, in general, a computationally hard problem.

Chrobak et al. in [10] focuses on greedy approaches to solve the k-median facility location problem. The naive greedy approach minimizes cost with each addition of server node but only produces solutions with $\Omega(n)$ approximation ratio. The *reverse greedy algorithm*, on the other hand, starts with all nodes being servers and iteratively removes nodes from the solution set. This method results in a much-improved approximation ratio of between $\Omega(\log n/\log\log n)$ and $O(\log n)$ when the distance is metric. In our experiments, we will use this method as a benchmark algorithm.

The local search with swap strategy was introduced in [5]. The algorithm works by choosing an initial set of k facilities. The algorithm then examines possible ways to swap a current server location with another client node for possible improvements over the current cost and executes the best swap. The procedure repeats until when no swap may reduce the cost. In the worst case, the solution produced will create a cost of $(3 + 2/p)$-times the optimal cost, where p is the number swaps that are being done at one time. In this paper, we aim to improve upon this algorithm by extending swap strategy above with community structure of the underlying graph.

Liao et al. in [21] explored clustering-based location-allocation methods. Their method utilizes Euclidean distances between physical locations. The main difference between this algorithm and our proposed algorithm is that we use pre-computed clusters and execute a local search to take place within computed communities, whereas their approach also identifies clusters along with the process. Therefore, our algorithm may utilize a wide range of community detection algorithms to provide community structures of different granularity.

3 Problem Formulation and Existing Solutions

We consider models of wireless networks that consist of undirected links between nodes. Formally, we define the model as follows:

Definition 1. *A* facility location (FL) network *is represented by a weighted graph $G = (V, E, V_f, \rho, w)$ where V is the set of* nodes *and E is a set of (undirected)* edges*; no multi-edge nor self-loop is allowed. The subset $V_f \subseteq V$ contains nodes that are candidate locations of servers while $V \setminus V_f$ contains client nodes. The function $\rho\colon V \setminus V_f \to \mathcal{R}$ is the* demand function *that assigns a* demand level *of service to each client node. The weight $w\colon E \to \mathcal{R}$ is a* distance function *measuring how close two neighboring nodes are.*

A *path* in the network is a sequence of edges $\{v_0, v_1\}, \ldots, \{v_{k-1}, v_k\}$ in E; k is the *length* of the path. The *distance* between two nodes u, v is the minimum length of a path connecting u and v and is denoted $\text{dist}(u, v)$. The dist function is a *metric* as for all nodes $u, v, w \in V$, $\text{dist}(u, w) \leq \text{dist}(u, v) + \text{dist}(v, w)$. We also assume that $\text{dist}(u, v) < \infty$ for any pairs of nodes u, v (i.e., G is connected).

We are interested in ways to place servers on nodes in G. Since we are going to focus on uncapacitated version of the facility location problem, each client node will implicitly connect to the server node that has the least distance.

Definition 2. *Given FL network $G = (V, E, V_f, \rho, w)$, and $k \in \mathbb{N}$, a k*-facility location (FL) instance *on G is a set $S \subseteq V_f$ containing k* server nodes*. The* cost *of a k-FL instance S is* $\text{cost}(S) = \sum_{v \in V} \min\{\text{dist}(v, u) \mid u \in S\} \cdot \rho(v)$.

The *k-means facility location problem* seeks a k-FL instance with minimum cost. Hence the problem is formally defined as

INPUT An FL network $G = (V, E, V_f, \rho, w)$, $k \in \mathbb{N}$.
OUTPUT A k-FL instance with minimum cost.

The problem has long been known to be NP-hard [10] through a reduction from dominating set problem. Next, we review two important approximation algorithms with known approximation ratios.

Reverse Greedy Algorithm [10]. Reverse greedy algorithm is a simple greedy algorithm that starts with setting the solution set $S = V_f$ and iteratively reduces the set by removing server nodes that leads to the least cost. The procedure repeats until $|S| = k$. More precisely, the algorithm is described in Algorithm 1. The algorithm repeats $n - k$ iterations where each iteration i computes costs for $n - i$ sets S', each taking $O(n(n - i))$ in a naive implementation, where $n = |V|$. Assuming that computing all-pair shortest path distance takes time $O(g(n))$. The total running time of the algorithm is thus $O\left(g(n) + (n - k)\sum_{j=k}^{n} j^2\right) = O(g(n) + (n - k)n^3)$.

Single-Swap Algorithm [5]. The algorithm starts with a set of k randomly selected facility locations. It then loops over pairs of nodes (u, v) where u is a currently selected location and v is not, and compares the costs of the k-FL instances before and after when u is swapped with v. After identifying the pair (u, v) which will result in maximum improvement in the cost, the algorithm performs the swap. This action will guarantee that the cost goes down with each iteration. The algorithm terminates when no pair (u, v) is found that reduces

Algorithm 1. RevGreedy(G, k)

```
[INPUT] FL network $G = (V, E, V_f, \rho, w)$, integer $k$
[OUTPUT] FL instance $S \subseteq V_f$
$S \leftarrow V_f$
**while** $|S| > k$ **do**
    $m \leftarrow \infty$
    **for** $u \in S$ **do**
        $S' \leftarrow S \setminus \{u\}$
        **if** cost$(S') < m$ **then**
            $v \leftarrow u$
    $S \leftarrow S \setminus \{v\}$
**return** $S$
```

the cost any further, at which point we are sure to reach a local optimum. The algorithm is named *single-swap algorithm*. Note that as each facility can only be swapped-in at most once, the algorithm will always terminate. Each iteration of the algorithm runs in time $O(k(n-k)n)$ and there may be at most $O(n)$ iterations. Assuming an $O(g(n))$ algorithm to compute all-pair shortest path, the algorithm runs in time $O(g(n) + kn^3)$.

Algorithm 2. SingleSwap(G, k)

```
[INPUT] FL network $G = (V, E, V_f, \rho, w)$, integer $k$
[OUTPUT] FL instance $S \subseteq V_f$
$S \leftarrow$ Initialize$(G, k)$; $c \leftarrow$ cost$(S)$; swap $\leftarrow$ true
**while** swap **do**
    $T \leftarrow S$
    **for** every pair $(u, v) \in V_f \times (V \setminus V_f)$ **do**
        $S' \leftarrow (S \setminus \{u\}) \cup \{v\}$; swap $\leftarrow$ false
        **if** cost$(S') < c$ **then**
            $T \leftarrow S'$; swap $\leftarrow$ true; $c \leftarrow$ cost$(S')$
    $S \leftarrow T$
**return** $S$
```

4 Community-Based Algorithms

Community structure refers to a notable property of a network where nodes are typically clustered into several subgraphs, i.e., *communities*, which are densely connected on the inside, and sparsely connected on the outside [15]. The property naturally arises from small-world networks and gives rise to a modular view of the overall network structure that enabled a wide range of applications. Over the last 10–15 years, a vast literature has been devoted to the description and detection of communities in a network [13,27], which has lead to a number of well-established methods. In particular, *modularity maximization* has been a widely-used approach that maximizes the concentration of edges within communities compared with a random null model. In particular, given a partition of nodes into clusters $C_1, C_2, \ldots, C_k$, the *modularity* is defined as

$$Q = \frac{1}{2m} \sum_{i,j \in V} \left[A_{i,j} - \frac{d_i d_j}{2m} \right] \delta(i,j)$$

where m is the sum of edge weights, $A_{i,j}$ is the edge weight between nodes i and j, d_x is the sum of edge weights adjacent to node $x \in \{i,j\}$, and $\delta(i,j)$ is 1 if i,j are in the same cluster and is 0 otherwise. An ideal community structure is a partition that maximizes Q. Finding a partition with maximum modularity is NP-hard in general. In our experiments, we use Louvain method that uses an agglomerative greedy heuristic to approximate communities [8].

Past works which bring network clustering and facility location problem together normally select server nodes in hope to find communities in networks. The logic flow goes as follows: An algorithm picks a set of server nodes in the network and at the same time, decides on which client nodes would be assigned to a server. Thus the process identifies clusters of nodes that are within close proximity to each server node, giving rise to a community structure. In this work, we adopt a different perspective: Assuming an extraneous mechanism truthfully identifies communities in the network. The ideal selection of server nodes in the network would then rely on this identified community structure. Namely, when picking server nodes, it would be sufficient to identify one or a few server nodes within single communities, hence more efficiently identify k-FL instances with low costs. In this line of thoughts, we propose two algorithms: *community select* and *community swap*.

4.1 Community Select Algorithm

The community select algorithm takes the identified community structure and selects a server node in each community that leads to minimum cost within this community. See Algorithm 3. The algorithm is simply implemented with running time $O(f(n) + g(\tilde{n}) + k\tilde{n}^2)$ where $f(n)$ is the running time of the extraneous community detection algorithm, $\tilde{n}$ is the largest size of a community, $g(\tilde{n})$ is the running time to compute all-pair shortest path on a graph with $\tilde{n}$ nodes.

Algorithm 3. CommunitySelect(G, k)

```
[INPUT] FL network $G = (V, E, V_f, \rho, w)$, integer $k$
[OUTPUT] FL instance $S \subseteq V_f$
Set $\mathcal{C} = (C_1, C_2, \ldots, C_k) \leftarrow$ CommunityDetect$(G, k)$
                                    ▷ Apply a community detection algorithm on $G$
for $C_i \in \mathcal{C}$ do
    $V_{f,i} = C_i \cap V_f$
    $v_i \leftarrow \arg\min_u \left\{ \sum_{v \in C_i \setminus V_f} \mathrm{dist}(v,u) \cdot \rho(v) \mid u \in V_{f,i} \right\}$
Return $S \leftarrow \{v_1, v_2, \ldots, v_k\}$
```

4.2 Community Swap Algorithm

The community swap algorithm extends from the single swap algorithm by incorporating the community structure. As opposed to the community select algorithm, it does not just pick the single optimal server node in each community, but rather, it applies local search and swaps to reach optimize costs. The difference between the classical single swap algorithm is that, when identifying possible pairs (u, v) to swap, the algorithm only looks at pairs (u, v) where u and v belong to the same community. In this way, the algorithm drastically reduces the running time. See Algorithm 4 for a detailed description.

Algorithm 4. CommunitySwap(G, k)

```
[INPUT] FL network $G = (V, E, V_f, \rho, w)$, integer $k$
[OUTPUT] FL instance $S \subseteq V_f$
Set $\mathcal{C} = (C_1, C_2, \ldots, C_k) \leftarrow$ CommunityDetect$(G, k)$
                                        ▷ Apply a community detection algorithm on $G$
for $1 \le i \le k$ do
    $v_i \leftarrow$ Select$(C_i, V_f)$              ▷ Initialize a server node in $V_f$
    $S \leftarrow S \cup \{v\}$
$c \leftarrow$ cost$(S)$
for $1 \le i \le k$ do
    swap $\leftarrow$ true
    while swap do
        $T \leftarrow S$
        for every $u \in (C_i \setminus V_f)$ do
            $S' \leftarrow (S \setminus \{v_i\}) \cup \{u\}$; swap $\leftarrow$ false
            if cost$(S') < c$ then
                $T \leftarrow S'$; swap $\leftarrow$ true; $c \leftarrow$ cost$(S')$
        $S \leftarrow T$
return $S$
```

This algorithm repeats the swapping process for each community $C_1, \ldots, C_k$; every community C_i may run in $O\left(\tilde{n}^2 n\right)$ where $\tilde{n}$ is the size of the largest community. Thus the total running time of the swapping process is $O\left(k\tilde{n}^2 n\right)$. The algorithm runs in time $O\left(f(n) + g(n) + k\tilde{n}^2 n\right)$ where $g(n)$ is the time of computing all-pair distance and $f(n)$ is the time of community detection.

5 Case Study: Auckland North Shore Networks

5.1 Data Set and Network Definition

With an area of 130 km^2 and 141 km coastline, the former North Shore City was the fourth largest urban area in New Zealand prior to its merge with other local councils to form the current Auckland City Council. We take Auckland Open Data offered by Auckland City Council with geographical information of

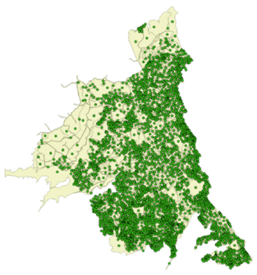
North Shore zoning [2]. The data set (Auckland Council District Plan Operative North Shore Section 2002) contains maps and descriptions of all land and sea zones in the region, each labeled by length, area, and zone types. Zone types include residential (7 classes), recreational (4 classes), rural (4 classes), business (12 classes), sea, road, and other. For clarification, zone types specify how the zone can be used and possible developments. For example, only a business can be placed in a business zone, and housing can only be placed in a residential zone. The zone class determines further restrictions on how buildings in that zone can be placed. For example, a Residential 6 zone is classified as an intensive housing zone where high-density housing is placed near specific commercial sites. The dataset can be processed and visualized using GIS applications such as ArcGIS[1], from which we are able to extract location (vectorization) of each zone, and the centroid of each zone. Further details and usage of the data set can be found in [1].

Our goal is to draw up simulated wireless mesh networks of North Shore given the zoning data set. Here, each land zone is going to be a node in our network (e.g., assume a device is placed at the centroid of the zone). The data set contains 3986 land zones. We calculate distances between centroids of each pair of zones, which become the edge weight. Thus the data set would result in a complete graph. To further reveal topological structures, we set a *threshold* θ so that the network only contains edges $\{u, v\}$ when the distance between u and v is no more than θ meters. Such edges represent simulated wireless connections based on proximity. We set all business zones on this network as potential facility locations. From Auckland City Council, we also obtain policies on allowable housing densities on residential zones of different classes. By combining density with zone area, we obtain estimates on the population of each residential zone. This allows us to assign a service demand to each residential zone. Here is the formal definition of the FL network $\text{NS}\theta = (V, E_\theta, w, V_f, \rho)$:

- V contains all land zones from North Shore.
- The set E_θ of edges contains pairs of nodes $\{u, v\}$ whose are within θ meters.
- $w(u, v) \in [0, \theta]$ represents the distance between u and v.
- The set V_f consists of all business zones.
- The service demand $\rho(v)$ of a node $v \in V \setminus V_f$ equals to the estimated population of v if v is residential, and 0 if v is of other types.

For our experiment, we set $\theta \in \{200, 300, 400\}$; the corresponding networks are called NS200, NS300, and NS400 networks, respectively. Figure 1 illustrates a map of North Shore with all land zones and illustrates the Fruchterman-Reingold visualization of the three networks. The network structures all exhibit strong community structures. Table 1 lists various properties of the networks including average clustering coefficient (ACC), density and average degree. It is clear that the networks are all sparse networks with very low density, however, with high clustering coefficients, meaning that they exhibit small-world property.

[1] www.arcgis.com.



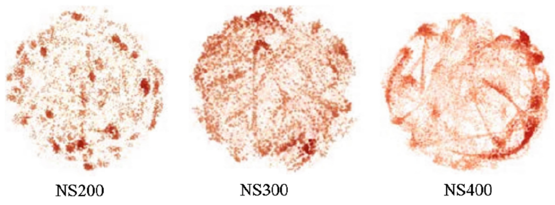


Fig. 1. Above: Auckland North Shore map and land zones. Below: Fruchtermar-Reingold layout of the networks NS200, NS300, and NS400.

Table 1. Properties of the graphs used

Graph	#Node	#Edge	ACC	Density	Avg deg
NS200	3986	12001	0.507	0.00151	6.02
NS300	3986	25544	0.593	0.00322	12.8
NS400	3986	42936	0.625	0.00541	21.5

Applying Louvain method, we identify the following community structure of the NS400 network. The NS400 network exhibits 18 non-trivial communities as shown in Fig. 2(a). The result is remarkably consistent with the real-world administrative divisions. For example, community 0 (in red) aligns very well with the suburb of Northcote, while community 14 (in violet) and 17 (in blue) align closely to the suburbs of Albany and Devonport, resp. Also noticeable is that the communities are clearly divided by State Highway 1 which divides North Shore vertically into eastern and western regions.

It is important to point out that, due to stochastic nature of the Louvain method, different runs of the algorithm will result in different communities. In our experiments, we need to set a parameter k indicating the number of communities identified from the data set. This results in different communities being found. In particular, we choose k largest communities in the process whenever these communities cover an area that is at least 75% of the overall area.

Below we illustrate our facility location algorithms when applied on the NS300 network. Fig. 2(b) shows the result of the community select algorithm with $k = 4$. The four identified communities roughly overlap with the four district boards: Devenport-Takapuna (green), Kaipatiki (brown), Upper Harbour (purple), and Hibiscus And Bays (blue).

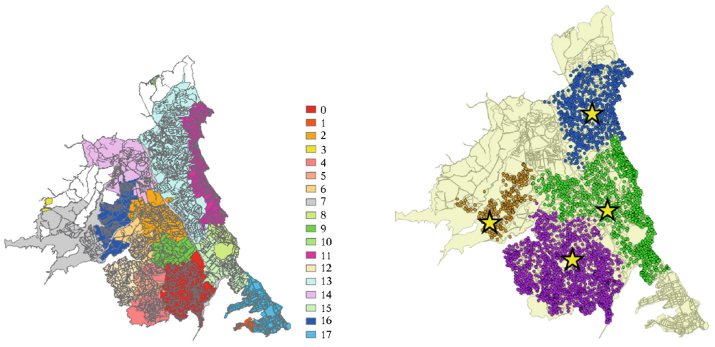


Fig. 2. Left (a): The 18 found communities of North Shore; Right (b): The result of CommunitySelect with $k = 4$ on NS300. Stars indicate the locations of the selected server nodes. (Color figure online)

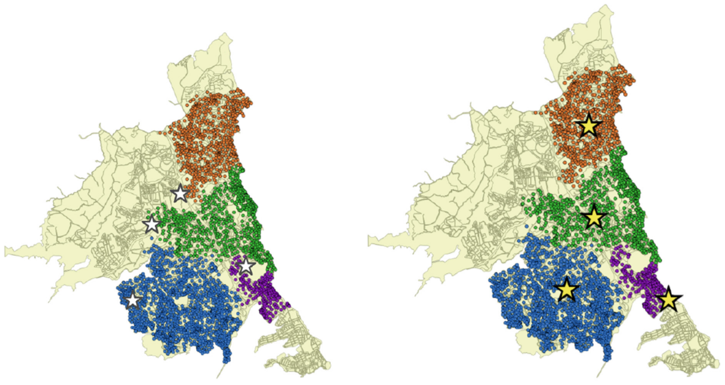


Fig. 3. The result of CommunitySwap with $k = 4$ on NS300. Left: the selected server nodes before swapping. Right: server nodes after swapping.

Figure 3 shows the result of the community swap algorithm. The algorithm identifies four different communities and initialized a random location in each community, which are visibly not optimal. Through local search and swapping, the algorithm is able to adjust the server nodes so that eventually produce a reasonable 4-FL instance.

5.2 Experiments

Our experiments aim to compare the performance of our community-based algorithm against the reverse greedy and single swap algorithms. The metrics that we use in our comparison include the cost of the resulting FL instance computed by each algorithm as well as the running time. The running time of the algorithms take into account also the time for computing shortest path distance between nodes as well as the time for detecting communities. To measure the performance of algorithms as k changes, we set $k \in \{1, 3, 5, 7, 9, 11\}$. In implementing the community select and community swap algorithms, we used Louvain method to compute k communities. Due to the inherent randomness of the Louvain method, for each network and each value of k, we run each algorithm 5 times and calculate the average outcome.

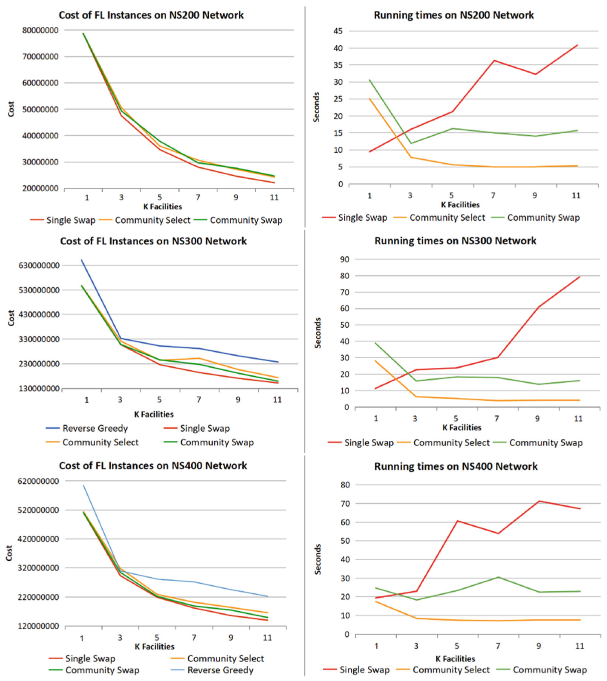


Fig. 4. Costs and running time of different algorithms ran on NS200 (top), NS300 (middle), NS400 (bottom) Network.

As illustrated in Fig. 4, results on all three networks are consistent: all four algorithm result in expected downwards trend in the cost as k increases. Despite its logarithmic theoretical approximation ratio, the reverse greedy produces significantly worse results than the other algorithms in terms of both cost and running speed by several magnitudes. We omit it in the NS200 network. In particular, the running time of the reverse greedy algorithm is about 30–100 times longer than community select algorithm (e.g. choosing 3 server nodes in NS300 network using community swap takes roughly 30 s, while using reverse greedy takes more than 30 min), so including it in the plot will trivialize the running time of all other algorithms. Thus we omit reverse greedy algorithm in all running time plots. On the other hand, the single swap algorithm in general produces FL instances with the lowest cost, however, in all three networks, the costs resulted from the community select and community swap algorithms are very close to single swap. Moreover, single swap algorithm results in much longer running time as compared with the community-based algorithm as $k > 3$.

Another remarkable point of the experimental result is regarding the running time comparisons between single swap with the community-based algorithms. While the single swap algorithm takes longer running time as k increases (this is consistent with the theoretical worst-case running time analysis in this paper), both community-based algorithms exhibit a flat or even downward trend in running time as k increases. This may be due to the fact that with a higher value of k, we divide the region into a larger number of communities, each having a smaller size. As the crucial factor in the running time depends on the largest size $\tilde{n}$ of a community, the running time of the community-based algorithm does not increase, resulting in almost constant-time algorithms.

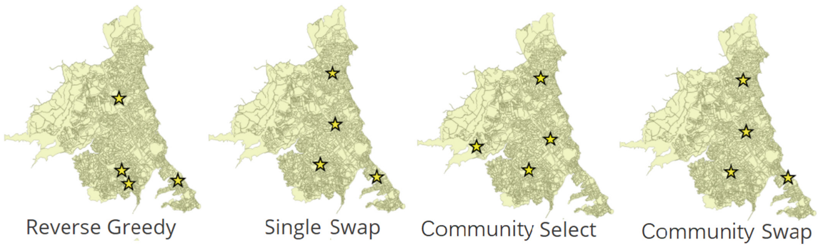


Fig. 5. Results from all four algorithms on NS300 Network where $k = 4$

The maps in Fig. 5 contain four server nodes picked from the NS300 network. The maps are to give an indication of why the cost of each algorithm within the network is displayed in that order, i.e. reverse greedy having the highest cost, community select having a somewhat evenly spaced facility locations and community swap and single swap having very similar results. In particular, when comparing with the real-world situation, the results produced by single swap and community swap are the most intuitive as all four facilities fall into well-recognized industrial areas that are also close to dense residential areas.

6 Conclusion and Future Work

This work focuses on the k-median facility location problem and proposes algorithms that incorporate a pre-determined community structure of the network. We analyze the performance of the algorithms on a real-world case study, i.e., spatial networks generated from Auckland North Shore. The experimental results reveal that the community swap algorithm produces outputs that are very close to the single swap algorithm while achieving almost constant time due to small search space. As a result, the community-based algorithms demonstrate a high potential to be used in real-world network data to solve this problem.

Our future work will be focused on adding more parameters to community swap so that it gives results that are useful for real-life networks. For example, it would be interesting to investigate the performance of the algorithm on road networks of land adjacency networks. Another future work concerns with more sophisticated application scenarios of the facility location problem. E.g., introducing competing and existing facilities in the problem domain. When competing facilities are considered we must take into account whether the algorithm is trying to take away as many customers away from the existing facilities or whether we are trying to avoid the existing algorithms so that it is connected to as customers as possible. A third direction to extend this work is to introduce capacity to server nodes. Our work only deals with uncapacitated facility location algorithms and therefore the next goal is to extend all our implemented facility location algorithms so that facilities have capacities in them.

References

1. Auckland council district plan operative north shore section 2002. Accessed 30 Sept 2017. http://temp.aucklandcouncil.govt.nz/EN/planspoliciesprojects/plansstrategies/DistrictRegionalPlans/northshorecitydistrictplan/Pages/home.aspx
2. Auckland open data. Accessed 30 Sept 2017. https://aucklandopendata-aucklandcouncil.opendata.arcgis.com/datasets
3. Ahlgren, B., Dannewitz, C., Imbrenda, C., Kutscher, D., Ohlman, B.: A survey of information-centric networking. IEEE Commun. Mag. **50**(7), 26–36 (2012)
4. Aikens, C.H.: Facility location models for distribution planning. Eur. J. Oper. Res. **22**(3), 263–279 (1985)
5. Arya, V., Garg, N., Khandekar, R., Meyerson, A., Munagala, K., Pandit, V.: Local search heuristics for k-median and facility location problems. SIAM J. Comput. **33**(3), 544–562 (2004)
6. Barabási, A.-L., Albert, R.: Emergence of scaling in random networks. Science **286**(5439), 509–512 (1999)
7. Bell, M.G., Iida, Y., et al.: Transportation Network Analysis. Wiley, New York (1997)
8. Blondel, V.D., Guillaume, J.-L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. J. Stat. Mech. Theor. Exp. **2008**(10), P10008 (2008)
9. Chaudhuri, S., Garg, N., Ravi, R.: The p-neighbor k-center problem. Inf. Process. Lett. **65**(3), 131–134 (1998)
10. Chrobak, M., Kenyon, C., Young, N.E.: The reverse greedy algorithm for the metric K-median problem. In: Wang, L. (ed.) COCOON 2005. LNCS, vol. 3595, pp. 654–660. Springer, Heidelberg (2005). https://doi.org/10.1007/11533719_66
11. Church, R., Velle, C.R.: The maximal covering location problem. Pap. Reg. Sci. **32**(1), 101–118 (1974)
12. Dai, F., Wu, J.: On constructing k-connected k-dominating set in wireless networks. In: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, 10 pp. IEEE (2005)
13. Fortunato, S.: Community detection in graphs. Phys. Rep. **486**(3), 75–174 (2010)
14. Frank, C., Römer, K.: Distributed facility location algorithms for flexible configuration of wireless sensor networks. In: Aspnes, J., Scheideler, C., Arora, A., Madden, S. (eds.) DCOSS 2007. LNCS, vol. 4549, pp. 124–141. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73090-3_9
15. Girvan, M., Newman, M.E.: Community structure in social and biological networks. Proc. Natl. Acad. Sci. **99**(12), 7821–7826 (2002)

16. González-Brevis, P., Gondzio, J., Fan, Y., Poor, H.V., Thompson, J., Krikidis, I., Chung, P.-J.: Base station location optimization for minimal energy consumption in wireless networks. In: IEEE 73rd Vehicular Technology Conference (VTC Spring), pp. 1–5. IEEE (2011)
17. Guha, S., Khuller, S.: Greedy strikes back: improved facility location algorithms. J. Algorithms **31**(1), 228–248 (1999)
18. Huang, C.-M., Lan, K.-C., Tsai, C.-Z.: A survey of opportunistic networks. In: 22nd International Conference on Advanced Information Networking and Applications-Workshops, AINAW 2008, pp. 1672–1677. IEEE (2008)
19. Lambiotte, R., Blondel, V.D., De Kerchove, C., Huens, E., Prieur, C., Smoreda, Z., Van Dooren, P.: Geographical dispersal of mobile communication networks. Phys. A Stat. Mech. Appl. **387**(21), 5317–5325 (2008)
20. Lewis, F.L., et al.: Wireless sensor networks. In: Smart Environments: Technologies, Protocols, and Applications, pp. 11–46 (2004)
21. Liao, K., Guo, D.: A clustering-based approach to the capacitated facility location problem. Trans. GIS **12**(3), 323–339 (2008)
22. Liu, J., Minnes, M.: Deciding the isomorphism problem in classes of unary automatic structures. Theor. Comput. Sci. **412**(18), 1705–1717 (2011)
23. Liu, J., Wei, Z.: Community detection based on graph dynamical systems with asynchronous runs. In: Second International Symposium on Computing and Networking (CANDAR), pp. 463–469. IEEE (2014)
24. Miller, H.J., Han, J.: Geographic Data Mining and Knowledge Discovery. CRC Press (2009)
25. Moskvina, A., Liu, J.: How to build your network? A structural analysis. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, pp. 2597–2603. AAAI Press (2016)
26. Moskvina, A., Liu, J.: Togetherness: an algorithmic approach to network integration. In: IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 223–230. IEEE (2016)
27. Newman, M.E.: Modularity and community structure in networks. Proc. Natl. Acad. Sci. **103**(23), 8577–8582 (2006)
28. Pantazopoulos, P., Stavrakakis, I., Passarella, A., Conti, M.: Efficient social-aware content placement in opportunistic networks. In: Seventh International Conference on Wireless On-Demand Network Systems and Services (WONS), pp. 17–24. IEEE (2010)
29. Watts, D.J., Strogatz, S.H.: Collective dynamics of 'small-world' networks. Nature **393**(6684), 440 (1998)
30. Yan, B., Chen, Y., Liu, J.: Dynamic relationship building: exploitation versus exploration on a social network. In: Bouguettaya, A., et al. (eds.) WISE 2017. LNCS, vol. 10569, pp. 75–90. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68783-4_6
31. Yang, K.-Q., Yang, L., Gong, B.-H., Lin, Z.-C., He, H.-S., Huang, L.: Geographical networks: geographical effects on network properties. Front. Phys. China **3**(1), 105–111 (2008)
32. Zhong, C., Arisona, S.M., Huang, X., Batty, M., Schmitt, G.: Detecting the dynamics of urban structure through spatial network analysis. Int. J. Geogr. Inf. Sci. **28**(11), 2178–2199 (2014)