Under consideration for publication in Knowledge and Information
Systems

Efficient processing of optimal meeting point queries in Euclidean space and road networks

Da Yan, Zhou Zhao and Wilfred Ng

Department of Computer Science and Engineering,
Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong

Abstract. Finding an *Optimal Meeting Point* (OMP) for a group of people (or a set of objects) at different locations is an important problem in spatial query processing. There are many real life applications related to this problem, such as determining the location of a conference venue, deciding the pick-up location of a tourist bus, and planing tactics of artificial intelligence in real-time strategy games. Formally, given a set Q of query points in a spatial setting P, an OMP query fetches the point $o \in P$ that minimizes a cost function defined over the distances from o to all points in Q. Since there are infinitely many locations in a given space setting, it is infeasible to examine all of them to find the OMP and, thus, the problem is challenging.

In this paper, we study OMP queries in the following two spatial settings which are common in real life applications: Euclidean space and road networks. In the setting of Euclidean space, we propose a general framework for answering all OMP query variations, and also identify the best algorithms for particular types of OMP queries in the literature. In the setting of road networks, we study how to access only part of the road network and examine part of the candidates. Specifically, we explore two pruning techniques, namely *Euclidean distance bound* and *threshold algorithm*, which help improve the efficiency of OMP query processing. Extensive experiments are conducted to demonstrate the efficiency of our proposed approaches on both real and synthetic datasets.

Keywords: Optimal meeting point; Spatial query processing; Road network; Threshold algorithm

Received Mar 17, 2013
Revised Jun 09, 2013
Accepted Aug 02, 2013

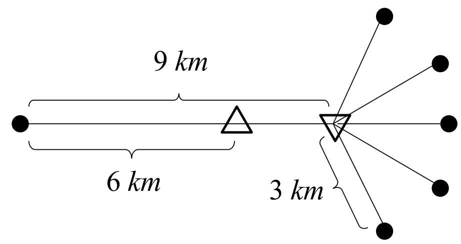


Fig. 1. Min-max (△) and Min-sum (▽) OMPs

1. Introduction

Optimal Meeting Point queries (or simply *OMP queries*) are useful in many real-world applications, ranging from location-based services to computer games. For example, a travel agency may issue an OMP query to decide the location for a tourist bus to pick up the tourists, so that the tourists can make the least effort to get to the meeting point. OMP queries are also important for merging military forces in a war field or finding a place that is convenient for military officers to hold a meeting. In strategy games like *Warcraft*[1], a computer player may need to find OMPs efficiently in order to decide the routes of its warriors.

Formally, given a set of query points $Q = \{q_1, q_2, \ldots, q_n\}$ in a spatial setting P, an OMP query finds the point $o \in P$ that minimizes a cost function defined over the distances from o to all points in Q. The spatial setting P can be the Euclidean space or a road network. Besides, there are two ways of defining the OMP o, which are based on the following two commonly used cost functions:

– **min-sum OMP**: $o = \arg\min_{p \in P} \sum_i d(q_i, p)$, and
– **min-max OMP**: $o = \arg\min_{p \in P} \max_i d(q_i, p)$,

where $d(p_i, p_j)$ denotes the distance between points p_i and p_j. The metric of distance can be the Euclidean distance (when it is defined in the Euclidean space setting) or the *network distance* (when it is defined in a road network setting). The network distance between two points in a road network is the length of the shortest path connecting them.

Figure 1 illustrates the concept of OMPs using a road network having six people (or query points in general) represented by the six black dots. Let us assume that those people want to meet together at some location in the road network. The upward (left) triangle in Figure 1 is the *min-max* OMP, and the downward (right) one is the *min-sum* OMP.

We can see from the above example that, a *min-sum* OMP minimizes the total travel distance of all the people, while a *min-max* OMP minimizes the elapsed travel time before all the people reach the meeting point. Referring to Figure 1 again, the person at the far left has to walk for 9 km to reach the *min-sum* OMP, and those on the right have to wait for him after they reach the meeting point. On the other hand, all the people would walk for 6 km to get to the *min-max* OMP, which is faster than the *min-sum* one.

The challenge of processing OMP queries is that, since there are infinitely many locations in P, it is infeasible to examine all of them to find the OMP.

[1] `http://us.blizzard.com/en-us/games/war3/`

To tackle this challenge, we propose two efficient algorithms for answering OMP queries, in two spatial settings that are fundamental in real-life applications: Euclidean space and road networks. Our algorithms well tackle the challenge of infinite search space.

In the setting of Euclidean space, we propose a general framework for answering all OMP query variations, and also identify the best algorithms for particular types of OMP queries in the literature.

In the setting of road networks, we study how to derive a finite number of OMP candidates, and how to access only part of the road network and examine part of the candidates, by exploring two pruning techniques that improve the efficiency of processing OMP queries.

The contributions of this paper are summarized as follows:

- We propose and define min-sum and min-max OMP queries, and study efficient algorithms for answering them in the settings of Euclidean space and road networks.
- We develop a gradient-descent based framework for answering all OMP query variations in Euclidean space in general, and also identify the best algorithms for particular types of OMP queries.
- For both min-sum and min-max OMP queries in road networks, we present an R-tree based branch-and-bound algorithm, which adopts a pruning technique called *Euclidean distance bound* to find an OMP by just accessing part of the road networks and examining part of the candidates. This algorithm can be applied when the Euclidean distance between any two locations in the road network lower bounds their network distance.
- We further propose another algorithm for finding OMPs in road networks. The algorithm is based on the *threshold algorithm* for top-k queries, and it is able to avoid deriving and examining all the candidates.
- We conduct extensive experiments to evaluate the efficiency of the proposed algorithms on seven real road networks of various sizes, as well as the synthetic datasets.

The rest of this paper is organized as follows: we review the related work in Section 2. Section 3 presents our gradient-descent based framework for answering all OMP query variations in Euclidean space, and reviews the relevant algorithms in the literature. In Section 4, we study how to derive a finite number of OMP candidates for min-sum and min-max OMP queries in road networks, and in Section 5, we describe our algorithms for OMP query processing. The proposed methods are empirically studied in Section 6. Finally, we conclude our paper in Section 7.

2. Related Work

In this section, we first give an overview of the related work for OMP queries in Euclidean space and in road networks. Then, we review the work related to two other spatial problems, namely *aggregate nearest neighbor queries* and the *facility location problem*, which also aim to determine an optimal location in a spatial setting, but the spatial contexts are different from our proposed OMP queries.

The Weber Problem. The studies of min-sum OMP queries in Euclidean space date back to the 60s–70s (Cooper, 1968; Ostresh, 1977; Chen, 1984a; Chen, 1984b). When the Euclidean distance is adopted as the metric of distance, the min-sum OMP query is called the *Weber problem* (Cooper, 1968), and the min-sum OMP is called the geometric median of the query point set Q.

Cooper (1968) extended the Weber problem by formalizing the problem of minimizing the weighted sum of powers of the Euclidean distances, which was further generalized to handle radial cost functions by Reuven Chen (Chen, 1984b). However, it is shown that no closed form formula exists for the Weber problem and its generalizations, and these problems are usually solved by gradient descent methods (Beck and Teboulle, 2009; Brimberg and Love, 1993; Wesolowsky, 1982).

OMP in Road Networks. As for OMP queries in road networks, Xu and Jacobsen (2010) studies them from the perspective of monitoring the proximity relations of a group of objects in a road network. The concept of an OMP query is first formalized in our prior work (Yan et al, 2011a), where we only study min-sum OMP queries. Given a query set Q in a road network $G = (V, E)$, Yan et al (2011a) proves that a min-sum OMP exists among the points in $V \cup Q$. Furthermore, two efficient search techniques are proposed in Yan et al (2011a), which are able to return high-quality meeting points. However, these two methods are not able to guarantee result optimality.

Compared with the preliminary work (Yan et al, 2011a), this work makes further contributions in the following issues. Firstly, we study min-sum and min-max OMP queries in Euclidean space. Secondly, we study min-max OMP queries in road networks. Last but not least, we improve the efficiency of OMP query processing in road networks, by using two pruning techniques: *Euclidean distance bound*, and *threshold algorithm*.

Aggregate Nearest Neighbor Queries. *Aggregate Nearest Neighbor queries* (or *ANN queries* in short) (Papadias et al, 2005; Yiu et al, 2005; Li et al, 2011a) are closely related to our OMP queries. However, the fundamental difference is that, for ANN queries, the result location is chosen among a finite data point set $P = \{p_1, p_2, \ldots, p_m\}$, while for OMP queries, the result location is chosen from a spatial setting P that contains infinite number of points. ANN queries can be applied, for instance, when n people at locations $\{q_1, \ldots, q_n\}$ want to choose a restaurant to have dinner together, among a set of restaurants at locations $\{p_1, \ldots, p_m\}$ in a city. However, when people are not able to fix a set of possible locations to meet at in advance, ANN queries are not applicable, while OMP queries are an appropriate choice. OMP queries are also more appropriate, when a school needs to decide the location for its school bus to pick up the students in some district, since the location can be anywhere in the road network.

A variation of ANN queries is studied in Li et al (2011b), where the cost function is defined over the distances from the target location o to any subset of $\varphi|Q|$ ($0 < \varphi \leq 1$) query points in Q. Recently, Ke Deng et al. propose the group nearest group query (Deng et al, 2012), which generalizes ANN queries by allowing multiple meeting points. Li et al (2013) studies ANN queries when query points are continuously moving, and Lian and Chen (2008) studies the processing of ANN queries when the data and query points are uncertain.

Facility Location Problem. *Facility Location Problem* (or FLP in short) is also related to OMP queries. Given a client point set C and a server point set S, FLP aims to find the location for a new server in a spatial setting, to minimize a cost function defined over the distance from each client to its nearest server. The problem is fundamentally different from finding an OMP, since FLP involves two sets C and S, while OMP queries only consider one query set Q. Yan et al (2011b), Du et al (2005) and Wong et al (2009) study this problem when P is the Euclidean space, while Xiao et al (2011) studies this problem when P is a finite point set in a road network.

3. Finding OMPs in Euclidean Space

In this section, we first define the notation and the concepts of min-sum and min-max OMPs in Euclidean space. Then, we present our gradient-descent based framework for answering all OMP query variations. Finally, we identify the best algorithms for particular types of OMP queries in the literature.

3.1. Problem Definition

Notation. We only focus on 2D Euclidean space for simplicity. Although it is straightforward to generalize the problem definition and gradient-descent solution to higher dimensional Euclidean spaces, they are less common as a practical spatial setting. In 2D Euclidean space, each point p is a location with coordinates (x, y), and the distance between two points $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ is actually the ℓ_2-norm of the vector $\overrightarrow{p_1p_2}$:

$$\|\overrightarrow{p_1p_2}\|_2 = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$

We now review the concept of ℓ_p-norm (Boyd and Vandenberghe, 2004):

Definition 1. The ℓ_p-norm ($p \geq 1$) of an n-dimensional vector $\overrightarrow{x} = (x_1, x_2, \ldots, x_n)$ is given by

$$\|\overrightarrow{x}\|_p = (|x_1|^p + |x_2|^p + \cdot + |x_n|^p)^{1/p}. \tag{1}$$

When $p \to \infty$, we obtain the *Chebyshev* or ℓ_∞-norm:

$$\|\overrightarrow{x}\|_\infty = \lim_{p\to\infty} \|x\|_p = max\{|x_1|, |x_2|, \ldots, |x_n|\}. \tag{2}$$

Problem Definition. The OMP queries in the Euclidean space are formally defined as follows:

Definition 2. Given a set of query points $Q = \{q_1, q_2, \ldots, q_n\}$ in 2D Euclidean space, where $q_i = (x_i, y_i)$, the min-sum OMP of Q is given by $\arg\min_{p\in\mathbb{R}^2} \sum_i \|\overrightarrow{q_ip}\|_2$, and the min-max OMP of Q is given by $\arg\min_{p\in\mathbb{R}^2} \max_i \|\overrightarrow{q_ip}\|_2$.

Given a point $p \in \mathbb{R}^2$ and a query point set Q, let us define $\overrightarrow{Qp} = (\|\overrightarrow{q_1p}\|_2, \|\overrightarrow{q_2p}\|_2, \ldots, \|\overrightarrow{q_np}\|_2)$. Then, the cost function for min-sum OMP is given by

$$f_{sum}(p) = \|\overrightarrow{Qp}\|_1 = \sum_{i=1}^{n} \|\overrightarrow{q_ip}\|_2, \tag{3}$$

and the cost function for min-max OMP is given by

$$f_{max}(p) = \|\overrightarrow{Qp}\|_\infty = \max_{i=1}^{n} \|\overrightarrow{q_ip}\|_2. \tag{4}$$

A generalization of the OMP query is the weighted OMP query, where each query point $q_i \in Q$ is associated with a weight w_i. In this case, $\overrightarrow{Qp} = (w_1 \cdot \|\overrightarrow{q_1p}\|_2, \ldots, w_n \cdot \|\overrightarrow{q_np}\|_2)$, and the cost functions become:

$$f_{sum}(p) = \|\overrightarrow{Qp}\|_1 = \sum_{i=1}^{n}\Big[w_i \cdot \|\overrightarrow{q_ip}\|_2\Big], \tag{5}$$

$$f_{max}(p) = \|\overrightarrow{Qp}\|_\infty = \max_{i=1}^{n}\Big[w_i \cdot \|\overrightarrow{q_ip}\|_2\Big]. \tag{6}$$

The intuition behind Equations (5) and (6) is that, a query point with a larger weight is more important and, thus, its travel cost is higher. For example, consider a group of people who want to find a meeting point. If one person is the boss and the rest are his/her employees, then the boss could be given a weight larger than his/her employees.

3.2. Gradient-Descent Framework

The problem of finding an OMP of a query point set Q can be regarded as an optimization problem: to find a point $p \in \mathbb{R}^2$ that minimizes the target function $f_{sum}(p)$ or $f_{max}(p)$.

Both $f_{sum}(p)$ and $f_{max}(p)$ are convex functions, since they are the composite of affine mapping and ℓ_p-norm operations that preserve convexity (Boyd and Vandenberghe, 2004). As a result, we can find the OMP using gradient descent: the convexity of $f_{sum}(p)$ and $f_{max}(p)$ guarantees that gradient descent is able to approach the global minimum without being stuck at local minimal values.

Gradient Evaluation. We now consider how to evaluate the gradient of functions $f_{sum}(p) = \|\overrightarrow{Qp}\|_1$ and $f_{max}(p) = \|\overrightarrow{Qp}\|_\infty$ at point $p = (x, y)$. Let us first compute the gradient of $\|\overrightarrow{Qp}\|_m$ at $p = (x, y)$ for arbitrary $m \geq 1$:

$$\|\overrightarrow{Qp}\|_m = \left[\sum_{i=1}^{n} w_i^m \cdot \|\overrightarrow{q_ip}\|_2^m\right]^{1/m} = \left\{\sum_{i=1}^{n} w_i^m \Big[(x_i - x)^2 + (y_i - y)^2\Big]^{m/2}\right\}^{1/m},$$

and therefore,

$$\begin{aligned}
\frac{\partial \|\overrightarrow{Qp}\|_m}{\partial x} &= \frac{1}{m}\left\{\sum_{i=1}^{n} w_i^m \Big[(x_i-x)^2+(y_i-y)^2\Big]^{m/2}\right\}^{\frac{1-m}{m}} \times \\
&\quad \sum_{i=1}^{n}\left\{w_i^m \cdot \frac{m}{2}\cdot \Big[(x_i-x)^2+(y_i-y)^2\Big]^{\frac{m-2}{2}} \times 2\cdot(x_i-x)\cdot(-1)\right\} \\
&= \left\{\sum_{i=1}^{n} w_i^m \Big[(x_i-x)^2+(y_i-y)^2\Big]^{m/2}\right\}^{\frac{1}{m}-1} \times \\
&\quad \sum_{i=1}^{n}\left\{w_i^m \Big[(x_i-x)^2+(y_i-y)^2\Big]^{\frac{m}{2}-1}\cdot(x-x_i)\right\} \qquad (7) \\
&\triangleq g_1(x,y)\times g_2(x,y). \qquad (8)
\end{aligned}$$

Due to the symmetry of x and y, $\frac{\partial \|\overrightarrow{Qp}\|_m}{\partial y}$ is similar to Equation (7), except that the last term becomes $(y-y_i)$ instead of $(x-x_i)$. For ease of presentation, let us define the following short-hand notations:

$$\Delta x_i = x - x_i, \qquad (9)$$

$$\Delta y_i = y - y_i. \qquad (10)$$

Since $f_{sum}(p)$ corresponds to the case when $m=1$, according to Equation (7), we have the following derivatives:

$$\frac{\partial f_{sum}(p)}{\partial x} = \sum_{i=1}^{n}\frac{w_i\cdot \Delta x_i}{\|\overrightarrow{q_ip}\|_2},\ \ \frac{\partial f_{sum}(p)}{\partial y} = \sum_{i=1}^{n}\frac{w_i\cdot \Delta y_i}{\|\overrightarrow{q_ip}\|_2}. \qquad (11)$$

As for $f_{max}(p)$, we need to set $m\to\infty$, which gives:

$$\lim_{m\to\infty} g_1(x,y) = \lim_{m\to\infty}\frac{\left[\sum_{i=1}^{n}(w_i\cdot\|\overrightarrow{q_ip}\|_2)^m\right]^{\frac{1}{m}}}{\sum_{i=1}^{n}(w_i\cdot\|\overrightarrow{q_ip}\|_2)^m} = \frac{\max_{i=1}^{n}(w_i\cdot\|\overrightarrow{q_ip}\|_2)}{\lim_{m\to\infty}\sum_{i=1}^{n}(w_i\cdot\|\overrightarrow{q_ip}\|_2)^m}, \qquad (12)$$

where the last step is obtained by using Equation (2).

Note that $g_2(x,y)$ can be reformulated as follows:

$$g_2(x,y) = \sum_{i=1}^{n}\frac{(w_i\cdot\|\overrightarrow{q_ip}\|_2)^m\cdot\Delta x_i}{\|\overrightarrow{q_ip}\|_2^2}. \qquad (13)$$

Therefore, according to Equations (8), (12) and (13), we have the following derivative:

$$\begin{aligned}
\frac{\partial f_{max}(p)}{\partial x} &= \lim_{m\to\infty} g_1(x,y)\times g_2(x,y) \\
&= \max_{i=1}^{n}(w_i\cdot\|\overrightarrow{q_ip}\|_2)\times\sum_{i=1}^{n}\left(\frac{\Delta x_i}{\|\overrightarrow{q_ip}\|_2^2}\cdot\lim_{m\to\infty}\frac{(w_i\cdot\|\overrightarrow{q_ip}\|_2)^m}{\sum_{i=1}^{n}(w_i\cdot\|\overrightarrow{q_ip}\|_2)^m}\right). \qquad (14)
\end{aligned}$$

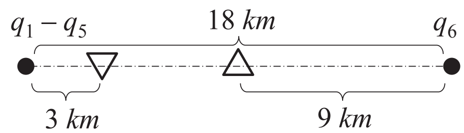


Fig. 2. Choice of the Starting Point for Finding Min-Max OMP

Let us define $i^* = \arg\max_i(w_i \cdot \|\overrightarrow{q_i p}\|_2)$, then we have

$$\lim_{m\to\infty} \frac{(w_i \cdot \|\overrightarrow{q_i p}\|_2)^m}{\sum_{i=1}^{n}(w_i \cdot \|\overrightarrow{q_i p}\|_2)^m} = \begin{cases} 1, & i = i^* \\ 0, & \text{otherwise} \end{cases} . \tag{15}$$

According to Equations (14) and (15), we obtain

$$\frac{\partial f_{max}(p)}{\partial x} = \frac{w_{i^*}\Delta x_{i^*}}{\|\overrightarrow{q_{i^*} p}\|_2}. \tag{16}$$

Finally, due to the symmetry of x and y, we also have

$$\frac{\partial f_{max}(p)}{\partial y} = \frac{w_{i^*}\Delta y_{i^*}}{\|\overrightarrow{q_{i^*} p}\|_2}. \tag{17}$$

Starting Point. We now consider how to choose the starting point for gradient descent. The starting point should be chosen to be close to the OMP, so that gradient descent requires fewer steps to reach the OMP. For a weighted min-sum OMP query, the starting point is usually chosen to be the center of gravity of the query point set Q, i.e. $(\frac{\sum_{i=1}^{n} w_i \cdot x_i}{\sum_{i=1}^{n} w_i}, \frac{\sum_{i=1}^{n} w_i \cdot y_i}{\sum_{i=1}^{n} w_i})$.

However, this is not a good choice for the min-max OMP query. Consider the example shown in Figure 2, where all the six query points q_1 to q_6 carry equal weight, and the query points q_1 to q_5 on the left are at the same location. It is straightforward to see that the min-max OMP is the upward triangle in Figure 2, but the center of gravity of the query points is the downward triangle, which is far from the min-max OMP.

Suppose that (q_a, q_b) is the farthest pair of points in the query point set Q. We propose to choose the midpoint of the line segment $\overline{q_a q_b}$ as the starting point of gradient descent for min-max OMP queries. Referring again to Figure 2, we can see that the starting point coincides with the min-max OMP.

In most cases, the midpoint of $\overline{q_a q_b}$, denoted as p_c, is a better starting point than the center of gravity of the query points for min-max OMP queries. Besides, p_c has the following validating property, which enables early termination:

Observation 1. For unweighted min-max OMP queries, if the query point farthest from p_c is q_a, then p_c is guaranteed to be the OMP.

Proof. Since p_c is the midpoint of $\overline{q_a q_b}$, $\|p_c q_a\|_2$ lower bounds $f_{max}(p)$, $\forall p \in R^2$. Furthermore, since the query point farthest from p_c is q_a, $f_{max}(p_c) = \|p_c q_a\|_2$. Therefore, p_c is the min-max OMP. $\square$

Note that the farthest pair (q_a, q_b) can be obtained in $O(n)$ time, by finding all the $O(n)$ antipodal pairs among Q, and then selecting the pair with the maximum separation. A detailed description of the algorithm can be found in Alsuwaiyel (1999)[1].

Gradient Descent Algorithm. Given the starting point and the gradient of the cost function, we can find the OMP of a query point set Q by gradient descent. We use the gradient descent algorithm of Boyd and Vandenberghe (2004) to find the OMP, where the step length is determined by backtracking line search that utilizes the target (cost) function, instead of being fixed as a small constant. Gradient descent stops when the movement of the current point (measured by Euclidean distance) is smaller than the tolerance parameter ϵ, which is usually set to a small number like 10^{-6}.

3.3. Algorithms for Specific OMP Query Types

While our gradient descent framework is able to answer arbitrary types of OMP queries, there exist even faster algorithms for particular types of OMP queries, which we now discuss next.

Faster Algorithms for Min-sum OMP Queries. The gradient descent method is only based on the first-order Taylor approximation of the target (cost) function. A more efficient approach for answering weighted min-sum OMP queries is to employ Newton's method (Boyd and Vandenberghe, 2004). Newton's method enables faster convergence, since it is based on the second-order Taylor approximation instead. As a result, besides the gradient of the target function, Newton's method also requires its Hessian matrix. Note that $f_{sum}(x)$ is second order derivable, and we derive its Hessian matrix from Equation (11) as follows:

$$\begin{aligned}\nabla^2 f_{sum}(p) \;&=\; \begin{bmatrix} \frac{\partial^2 f_{sum}(p)}{\partial x^2} & \frac{\partial^2 f_{sum}(p)}{\partial x \partial y} \\ \frac{\partial^2 f_{sum}(p)}{\partial y \partial x} & \frac{\partial^2 f_{sum}(p)}{\partial y^2} \end{bmatrix} \\ = \sum_{i=1}^{n} w_i &\begin{bmatrix} \frac{1}{\|\overrightarrow{q_i p}\|_2} - \frac{\Delta x_i^2}{\|\overrightarrow{q_i p}\|_2^3} & -\frac{\Delta x_i \Delta y_i}{\|\overrightarrow{q_i p}\|_2^3} \\ -\frac{\Delta x_i \Delta y_i}{\|\overrightarrow{q_i p}\|_2^3} & \frac{1}{\|\overrightarrow{q_i p}\|_2} - \frac{\Delta y_i^2}{\|\overrightarrow{q_i p}\|_2^3} \end{bmatrix}. \end{aligned} \tag{18}$$

We may also use Newton's method to find the weighted min-sum OMP. Our experiments show that Newton's method takes considerably less steps to reach the OMP, and is faster than gradient descent.

Another competitive method for solving the Weber problem is to use Weiszfeld's algorithm, a form of iteratively re-weighted least squares, where the current point p is updated by the operation $p \leftarrow \left(\sum_{i=1}^{n} \frac{q_i}{\|\overrightarrow{q_i p}\|_2}\right) / \left(\sum_{i=1}^{n} \frac{1}{\|\overrightarrow{q_i p}\|_2}\right)$. We find through experiments that Weiszfeld's algorithm is comparable to Newton's method in terms of efficiency although it takes more steps. This is because Newton's method requires evaluating the Hessian matrix, while the update operation of Weiszfeld's algorithm is much cheaper.

Faster Algorithms for Min-Max OMP Queries. While a min-sum

[1] Note that the pseudo-code on Page 478 of Alsuwaiyel (1999) is incorrect unless "$A \leftarrow A \cup \{(p_i, p_j)\}$" is added between Lines 13 and 14.

OMP can only be found by numerical methods, an unweighted min-max OMP can be computed exactly.

In Euclidean space, an unweighted min-max OMP query with query point set Q is equivalent to the smallest enclosing circle problem, which finds the smallest circle that contains all the query points in Q. Note that the center of that circle is exactly the min-max OMP. Using the terminology of facility location problem, the unweighted min-max OMP query is also known as the *1-center problem* (Drezner and Shelah, 1987).

Shamos and Hoey propose an $O(n \log n)$ algorithm (Shamos and Hoey, 1975) for tackling the smallest enclosing circle problem, which is based on the farthest Voronoi diagram of Q. The best time complexity is achieved by Megiddo's algorithm (Megiddo, 1982). Essentially, each iteration of Megiddo's algorithm prunes at least $\lfloor n/16 \rfloor$ points, and has time cost $O(n)$; thus, if we denote $T(n)$ to be the time cost of Megiddo's algorithm, we have $T(n) = O(n) + T(15n/16) = O(n + \frac{15}{16}n + (\frac{15}{16})^2 n + \cdots) = O(n)$.

Megiddo's algorithm is important in theory, since it shows that min-max OMP can be found in linear time. However, Welzl's randomized algorithm (De Berg et al, 2008) is much faster than Megiddo's algorithm in practice, although it only runs in expected $O(n)$ time. This is due to the simplicity of Welzl's randomized algorithm, compared with the complicated operations involved in Megiddo's algorithm. In this paper, we only focus on Welzl's randomized algorithm. Our experiments also show that, for unweighted min-max OMP queries, Welzl's algorithm is extremely efficient, which is much faster than our gradient descent method.

Role of Our Gradient-Descent Framework. Although our gradient-descent framework is not competitive with the alternative approaches mentioned above for specific types of OMP queries, it serves as a baseline for comparison. Furthermore, some types of OMP queries do not have an alternative approach and have to be solved by our gradient-descent framework, such as the weighted min-max OMP queries.

4. Deriving OMP Candidates in Road Networks

We now study OMP query processing in road networks, which is a more realistic spatial setting for location-based services, in cities with well-developed traffic networks. In this section, we first present the approaches of deriving a finite number of OMP candidates. The algorithmic details of finding the OMP will be discussed in Section 5.

Notation. Let us use $d_N(p_1, p_2)$ to denote the network distance between two locations p_1 and p_2 in a road network.

Given a set of query points $Q = \{q_1, q_2, \ldots, q_n\}$ in a road network $G = (V, E)$, the min-sum OMP of Q is given by $\arg\min_{p \in G} \sum_i d_N(q_i, p)$, and the min-max OMP of Q is given by $\arg\min_{p \in G} \max_i d_N(q_i, p)$, where $p \in G$ means that p is located on some edge of G.

When Q is clear from the context, we define $sd(p) = \sum_i d_N(q_i, p)$ and $md(p) = \max_i d_N(q_i, p)$. Furthermore, we denote by $\widetilde{p_i p_j}$ the shortest path between p_i and p_j, and if p_i and p_j are on edge $(u, v) \in E$, we denote by $|p_i p_j|$ the length of the part of the edge between p_i and p_j.

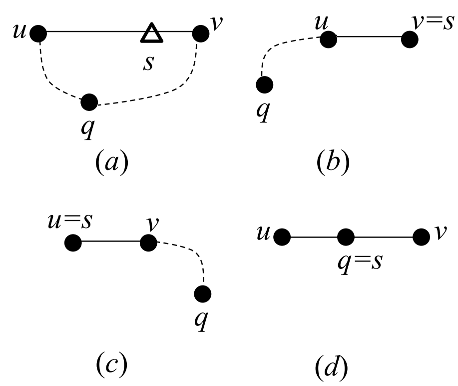


Fig. 3. Four Cases for Split Points

For a point p on edge $(u, v) \in E$, we represent p as a triplet (u, v, θ) with θ satisfying $\overrightarrow{up} = \theta \cdot \overrightarrow{uv}$. Note that $|up| = \theta|uv|$ and $|pv| = (1-\theta)|uv|$.

4.1. Split Points

Split points are an important concept for deriving OMP candidates in road networks. For a point q in a road network, its *split point* on edge (u, v) is defined as the point s such that

$$d_N(q, u) + |us| = d_N(q, v) + |vs|. \tag{19}$$

Figure 3(a) illustrates the concept of *split points*, where the dotted curves denote the shortest paths between the end points. The location marked by the triangle in Figure 3(a) is the split point s of q on edge (u, v). The shortest path from q to any point on the left (or right) of s on edge (u, v) passes through u (or v). The split point s exists because $d_N(q, u)+|uv| \geq d_N(q, v)$ and $d_N(q, v)+|uv| \geq d_N(q, u)$.

Note that the above definition of *split points* is only applicable when q is not on (u, v). This condition can be further divided into three cases illustrated by Figures 3(a)–(c):

- **Case 1: $\widetilde{qv}$ does not pass through u, and $\widetilde{qu}$ does not pass through v (see Figure 3(a)).** For an arbitrary point $p = (u, v, \theta)$ on edge (u, v), $d_N(q, p)$ can be represented as a piecewise linear function of θ delimited by s: if $\theta < |us|/|uv|$, $d_N(q, p) = d_N(q, u) + |up| = d_N(q, u) + \theta|uv|$; otherwise, $d_N(q, p) = d_N(q, v) + |vp| = d_N(q, v) + (1-\theta)|uv|$.
- **Case 2: $\widetilde{qv}$ passes through u (see Figure 3(b)).** In this case, the split point is vertex v, as can be verified by using Equation (19). For an arbitrary point $p = (u, v, \theta)$ on edge (u, v), $d_N(q, p) = d_N(q, u) + |up| = d_N(q, u) + \theta|uv|$, which is a linear function of θ.
- **Case 3: $\widetilde{qu}$ passes through v (see Figure 3(c)).** In this case, the split point is vertex u, as can be verified by using Equation (19). For an arbitrary point $p = (u, v, \theta)$ on edge (u, v), $d_N(q, p) = d_N(q, v) + |vp| = d_N(q, v) + (1-\theta)|uv|$, which is a linear function of θ.

In the above three cases, one can easily derive from Equation (19) that $s = (u, v, \theta_s)$, with $\theta_s = \frac{|uv|+|q_i v|-|q_i u|}{2|uv|}$.

When $q = (u, v, \theta_q)$ is on (u, v) (see Figure 3(d)), we define q to be the split point s, so that for an arbitrary point $p = (u, v, \theta_p)$ on edge (u, v), $d_N(q, p) = |\theta_p - \theta_q| \cdot |uv|$, which is still a piecewise linear function delimited by s.

Therefore, given a query point set Q, for each query point $q \in Q$ and a point p on edge (u, v), $d_N(q, p)$ is always a piecewise linear function delimited by the split point s. Since $sd(p)$ is the sum of piecewise linear functions, it achieves the minimum or the maximum at delimiting points. Thus, Xu and Jacobsen (2010) concludes that a min-sum OMP must exist among the split points. An algorithm is proposed in Xu and Jacobsen (2010) which checks the split point of each query point in Q on each edge in the road network $G = (V, E)$, and picks the split point p with the smallest value of $sd(p)$ as the min-sum OMP. As a result, the number of candidates to check is $|Q| \cdot |E|$. Although Xu and Jacobsen (2010) includes a pruning technique to skip some split points that are guaranteed not to be a min-sum OMP, the search space after pruning is still huge.

4.2. Deriving Min-sum OMP Candidates

We discover the following property of min-sum OMP queries in road networks, which significantly reduces the computational cost:

Theorem 1. Given a query point set $Q = \{q_1, q_2, \ldots, q_n\}$ in graph $G = (V, E)$, where each point q_i is associated with a weight w_i. If all the weights are integers or rational numbers, then $V \cup Q$ must contain a min-sum OMP.

Theorem 1 states that it suffices to check only the vertices in V and the points in Q for finding the min-sum OMP, which reduces the candidate space of min-sum OMP queries from $|Q| \cdot |E|$ into $(|V| + |Q|)$. Note that Theorem 1 is valid even for weighted min-sum OMP, since computers approximate irrational numbers with floating point numbers. Our previous experiments in Yan et al (2011a) show that the algorithm that is based on Theorem 1 is always an order of magnitude faster than the algorithm of Xu and Jacobsen (2010).

Before we present the complete proof of Theorem 1, we first prove that this theorem holds when query points are unweighted, as would be established by Lemma 2. The proof of this special case is based on Lemma 1 below. We will then use Lemma 2 to prove Theorem 1 for weighted query points.

Lemma 1. Given a query point set Q, let $sd(p)$ denote the sum of distances of point p to the points in Q. Suppose that no point in Q is on edge (u, v) except for the two end points u and v, then for any point x on edge (u, v), we have $sd(x) \geq \min\{sd(u), sd(v)\}$.

Proof. For a point x on edge (u, v), we denote Q_u as the set of query points whose shortest paths to x pass through u. Accordingly, $Q_v = Q - Q_u$ is the set of query points whose shortest paths to x pass through v. Without loss of generality, let us assume that $|Q_u| \geq |Q_v|$. Figure 4 illustrates this scenario, where the hollow points are the query points and the dotted lines are part of their shortest paths to x.

Now, consider the point x' on edge (u, v) which is δ closer to u than x. Let Q_{ab} $(a, b \in \{u, v\})$ denote the set of query points that belong to Q_a when the

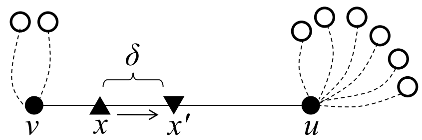


Fig. 4. Illustration of Lemma 1

meeting point is x and belong to Q_b when the meeting point is x'. Therefore, we can classify the points in Q into the four disjoint sets of Q_{uu}, Q_{vv}, Q_{uv} and Q_{vu}.

For these four point sets, we have the following properties:

1. $\forall p \in Q_{uu}$, $d_N(p, x') = d_N(p, x) - \delta$.
   This is because: $d_N(p, x') = d_N(p, u) + |ux'| = d_N(p, u) + (|ux| - \delta) = (d_N(p, u) + |ux|) - \delta = d_N(p, x) - \delta$.
2. $\forall p \in Q_{vv}$, $d_N(p, x') = d_N(p, x) + \delta$.
   This is because: $d_N(p, x') = d_N(p, v) + |vx'| = d_N(p, v) + (|vx| + \delta) = (d_N(p, v) + |vx|) + \delta = d_N(p, x) + \delta$.
3. $Q_{uv} = \emptyset$.
   This is because: for any $p \in Q_u$ when the meeting point is x, we have $d_N(p, v) + |vx'| = d_N(p, v) + (|vx| + \delta) > d_N(p, v) + |vx| \geq d_N(p, x) = d_N(p, u) + |ux| = d_N(p, u) + (|ux'| + \delta) > d_N(p, u) + |ux'|$, which implies that the shortest path from p to x' cannot pass through v (i.e. $p \notin Q_v$) when the meeting point is x'.
4. $\forall p \in Q_{vu}$, $d_N(p, x') \leq d_N(p, x) + \delta$.
   This is because: $d_N(p, x') \leq d_N(p, v) + |vx'| = d_N(p, v) + |vx| + \delta = d_N(p, x) + \delta$.

Therefore, we have

$$
\begin{aligned}
&\sum_{q \in Q} d_N(q, x) \\
&= \left(\sum_{q \in Q_{uu}} + \sum_{q \in Q_{vv}} + \sum_{q \in Q_{uv}} + \sum_{q \in Q_{vu}} \right) d_N(q, x) \\
&\geq \sum_{q \in Q_{uu}} [d_N(q, x') + \delta] + \\
&\quad \left(\sum_{q \in Q_{vv}} + \sum_{q \in Q_{vu}} \right) [d_N(q, x') - \delta] \\
&= \left(\sum_{q \in Q_{uu}} + \sum_{q \in Q_{vv}} + \sum_{q \in Q_{vu}} \right) d_N(q, x') \\
&\quad + \delta(|Q_{uu}| - |Q_{vv}| - |Q_{vu}|).
\end{aligned}
$$

As $Q_{uv} = \emptyset$, we have $\sum_{q \in Q_{uv}} d_N(q, x') = 0$. Besides, since $|Q_u| \geq |Q_v|$ when the meeting point is x, i.e. $|Q_{uu}| + |Q_{uv}| \geq |Q_{vu}| + |Q_{vv}|$, we have $|Q_{uu}| - |Q_{vv}| - |Q_{vu}| \geq -|Q_{uv}| = 0$.

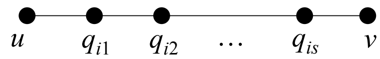


Fig. 5. Illustration of Lemma 2

According to the above analysis,

$$\begin{aligned}
&\sum_{q\in Q} d(\overline{q,x})\\
&\geq \left(\sum_{q\in Q_{uu}} + \sum_{q\in Q_{vv}} + \sum_{q\in Q_{uv}} + \sum_{q\in Q_{vu}}\right) d(\overline{q,x'})\\
&= \sum_{q\in Q} d(\overline{q,x'}).
\end{aligned}$$

Thus, we conclude that $sd(x') \leq sd(x)$ for arbitrary x, x' and δ. If we set x' to be u, we reach the conclusion that $\forall x$ on edge (u,v), $sd(u) \leq sd(x)$. Due to the symmetry of u and v, if $|Q_v| \geq |Q_u|$ we get: $\forall x$ on edge (u,v), $sd(v) \leq sd(x)$. To sum up, $\forall x$ on edge (u,v), $\min\{sd(u), sd(v)\} \leq sd(x)$. □

Intuitively, Lemma 1 shows that for any edge on the road network, one of the endpoints is at least as good as any other point on the edge in terms of the sum-of-distances value. Now, let us take into consideration the special case where there exist some query points on an edge, as illustrated by Figure 5. By using Lemma 1, we have the following lemma:

Lemma 2. Given an OMP query with query point set Q on a road network $G = (V,E)$, $V \cup Q$ contains an OMP.

Proof. For each edge (u,v) that contains some query points on it, but not at the end points u and v, let us denote these query points as $q_{i_1}, q_{i_2}, \ldots, q_{is}$, as illustrated in Figure 5. We introduce s dummy vertices $p_{i_1}, p_{i_2}, \ldots, p_{is}$ on the edge (u,v), where each dummy vertex $p_{ij},\ (j = 1,2,\ldots,s)$ is located at q_{ij}.

After the introduction of the dummy vertices for all the edges that contain some query points on it but not at its end points, we obtain another road network G' such that all the query points in Q are at its vertices. Since the vertex set of G' is $V \cup Q$, we can conclude that $V \cup Q$ contains an OMP according to Lemma 1. □

Using Lemma 2 we now prove Theorem 1, which is for the general case of weighted query points.

Proof. It is straightforward to convert the rational number weights into integer weights with the same weight distribution among all the points in Q. For example, suppose $Q = \{q_1, q_2, q_3\}$, $w_1 = 0.15$, $w_2 = 1.11$ and $w_3 = 0.8$, then we may re-assign the weights to be $w_1 = 15$, $w_2 = 111$ and $w_3 = 80$. Clearly, this transformation does not change the result point $\overline{x} = \arg\min_x \sum_i w_i \cdot d_N(q_i, x)$.

Now, let us assume that all the weights are integers. We replace each point q_i with w_i new points at the same location of q_i, each of which has weight 1. The resulting new query point set Q' can be treated as unweighted, and thus $Q' \cup V$ contains the OMP $\overline{x}$ according to Lemma 2. It is straightforward to see that the

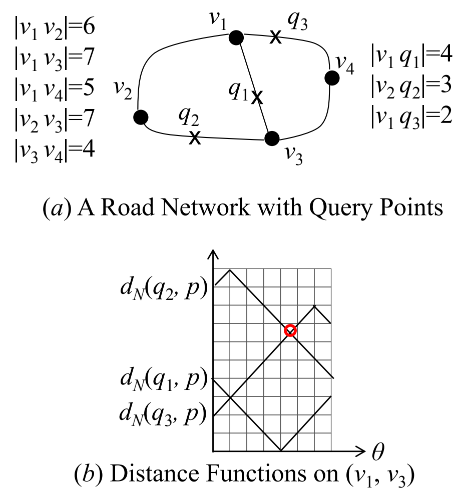


Fig. 6. Evaluation of Min-Max OMP Candidates

transformation from Q to Q' does not change the result point $\overline{x}$, and that the locations in Q' is exactly the locations in Q. □

4.3. Deriving Min-Max OMP Candidates

Unlike min-sum OMP queries, the min-max OMP may not coincide with any split points or edge endpoints. For each edge $(u, v) \in E$, we define its candidate for min-max OMP as the local optimal point $\arg\min_{p \in (u,v)} md(p)$. Figure 6 illustrates the process of computing the min-max OMP candidate for an edge. We show in Figure 6(a) an example of a road network with three query points q_1, q_2 and q_3, where the edge length and the positions of the query points are given. One can easily derive the distance functions $d_N(q_i, p)$ $(i = 1, 2, 3)$ for point $p = (v_1, v_3, \theta)$ on edge (v_1, v_3), which is shown in Figure 6(b), and $md(p)$ is the upper envelope of the distance functions $d_N(q_i, p)$. In Figure 6(b), $md(p) = d_N(q_2, p)$ when $0 \leq \theta \leq 4.5/7$, and $md(p) = d_N(q_3, p)$ when $4.5/7 < \theta \leq 1$, and the min-max OMP candidate is the lowest point on the upper envelope, i.e. $\theta = 4.5/7$ where $md(p) = 6.5$. Here, the split points on edge (v_1, v_3) are those points with $\theta = 1/7, 4/7, 6/7$, and the min-max OMP candidate is neither a split point, nor an edge endpoint.

We now present our approach of computing the min-max OMP candidate of an edge (u, v). The first step is to compute the split points of all query points q_i on (u, v), and divide the domain $[0, 1]$ of θ into several ranges using the split points. Referring to the example in Figure 6 again, we can obtain three ranges $[0, 1/7]$, $[1/7, 4/7]$ and $[4/7, 6/7]$ for θ. Note that in each range $[\theta_a, \theta_b]$, for any query point q_i, the portion of function $d_N(q_i, p)$, denoted as $d_N(q_i, p)|_{[\theta_a, \theta_b]}$, is a unique linear function.

After deriving the ranges, we compute the local optimal points for all ranges, and then choose the one with the minimum value of $md(p)$ as the min-max OMP candidate of edge (u, v).

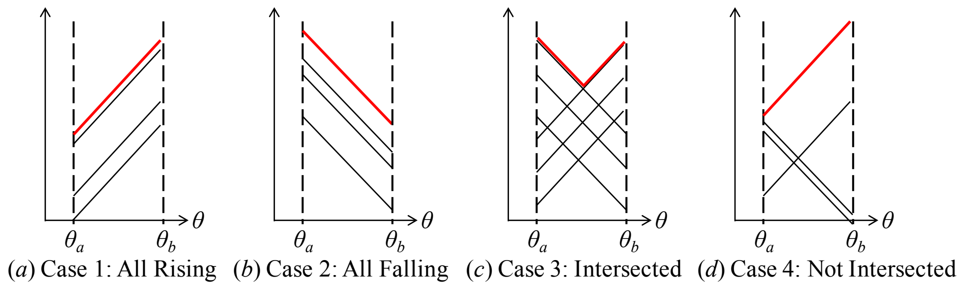


Fig. 7. Illustration of Observation 2

For each range $[\theta_a, \theta_b]$, we call a query point q_i as "rising" (or respectively "falling") if $d_N(q_i, p)|_{[\theta_a,\theta_b]}$ increases (or respectively decreases) as θ increases. Our algorithm for computing the local optimal point in each range is based on the following observation:

Observation 2. Within each range $[\theta_a, \theta_b]$ of an edge (u, v), the distance functions $d_N(q_i, p)|_{[\theta_a,\theta_b]}$ of all the rising (or respectively falling) query points q_i are linear functions of θ with the same slope $|uv|$ (or $-|uv|$).

Therefore, referring to Figure 7(a) (or (b)), if all the query points are rising (or falling), their distance functions $d_N(q_i, p)|_{[\theta_a,\theta_b]}$ are parallel line segments and the upper envelope is defined only by the highest one. Otherwise, the upper envelope is defined by both the highest distance function of the rising query point, and the highest distance function of the falling query point: if the two line segments intersect (see Figure 7(c)), the local optimal point is the intersection point of the two line segments; otherwise, the upper envelope is defined by the higher line segment (see Figure 7(d)).

Thus, a local optimal point of a range can be computed by picking the highest distance functions, which takes $O(|Q|)$ time. Since there are $O(|Q|)$ ranges for each edge, it takes $O(|Q|^2)$ time to find the min-max OMP candidate on each edge, which is rather expensive.

To cope with this problem, candidate evaluation should be avoided on those edges that do not contain the min-max OMP. We now formalize this idea by proposing a pruning rule, which is given in Theorem 2.

Lemma 3. For any two points p and p' in the road network, $md(p) \leq md(p') + d_N(p, p')$.

Proof. Let q_j be the query point farthest from p, and let q_k be the query point farthest from p'. Then, $md(p) = d_N(q_j, p) \leq d_N(q_j, p') + d_N(p', p) \leq d_N(q_k, p') + d_N(p, p') = md(p') + d_N(p, p')$. □

Theorem 2. For any point p on edge (u, v), $md(p) \geq (md(u) + md(v) - |uv|)/2$.

Proof. According to Lemma 3, we have $md(u) \leq md(p) + |up|$, and $md(v) \leq md(p) + |pv|$. The proof follows immediately by summing the above two inequalities. □

Theorem 2 gives the lower bound of the $md(p)$ for all points p on (u, v). Let p^* be the best min-max OMP candidate currently found. If the lower bound defined by Theorem 2 is larger than $md(p^*)$, then (u, v) can be pruned.

5. Algorithms for Answering OMP Queries in Road Networks

In this section, we present the algorithms for finding the OMP in a road network. In Section 5.1, we introduce how we organize the road network and compute the network distance. We present our baseline algorithm in Section 5.2, the Euclidean distance bound based algorithm in Section 5.3, and the threshold algorithm based algorithm in Section 5.4.

5.1. Road Network Organization

Road Network Representation. We use the adjacency list representation to organize road networks. Since our algorithms for processing OMP queries require the techniques of graph traversal, we hold the road network data in memory to avoid I/O operations. In fact, although several disk-based adjacency list representations have been proposed for road networks (Yiu et al, 2005; Shekhar and Liu, 1997; Papadias et al, 2003), these methods require one I/O operation to access an edge, and thus, they are very inefficient for graph traversal.

Most road network data fit well into the main memory of a modern computer, such as the road networks for cities from Li et al (2005). Even when the road network is too large to fit into the memory, such as a continental road network, one may partition the road network using existing road network partition techniques (Xu and Jacobsen, 2010; Yan et al, 2013), and only load the relevant partitions into the main memory. Since this issue is not in the scope of this work, we do not provide further discussion.

Network Distance Computation. The definition of OMP queries is based on the network distance from a meeting point p to all the query points in Q. Therefore, before presenting our algorithms for answering OMP queries, we first briefly describe our approach to computing the network distance.

Some previous studies about query processing in road networks materialize the network distances between all the $|V|(|V|-1)$ pairs of vertices (Yiu et al, 2005; Yan et al, 2011a), and store them on disk. However, the storage cost is prohibitive for large road networks, and this method requires one I/O operation to obtain the network distance between each pair of vertices during query processing.

In fact, for each OMP query with query set Q, it is sufficient to run $|Q|$ rounds of *Single-Source Shortest Path* (SSSP) computation online, each with a source $q_i \in Q$. On the other hand, our previous work on OMP queries (Yan et al, 2011a) chooses to materialize all pairwise network distances offline, and as a result, these algorithms become I/O bound.

We have done experiments to compare the algorithms in Yan et al (2011a) with those proposed in this paper, and the results show that the algorithms proposed in this work are much more efficient than the original algorithms, except for a gradient-descent-style greedy algorithm that does not guarantee result optimality.

Another choice for network distance computation is to use a shortest path index. A lot of shortest path indices have been proposed for road networks. For example, the indices of Samet et al (2008), Sankaranarayanan et al (2009), and Cohen et al (2003) still require to pre-compute all pairwise network distances offline, but they consume less space than $O(|V|^2)$ at the cost of spending longer query time than $O(1)$. On the other hand, the indices of Geisberger et al (2008) and Tao et al (2011) focus on accelerating online shortest path computation.

As shortest path computation is not our main focus, we simply compute network distance online by Dijkstra's algorithm. For each query point $q_i \in Q$, we run Dijkstra's algorithm with source q_i in a pay-as-you-go fashion: whenever the query evaluation requires a network distance $dist(p, q_i)$, we check whether it is already computed by the SSSP computation with source q_i; if $dist(p, q_i)$ is already computed, we use it directly; otherwise, we continue to run the SSSP computation until $dist(p, q_i)$ is computed.

Our pay-as-you-go SSSP computation only visits the vertices that have to be visited, and each vertex is visited at most once from a source q_i. The method is particularly effective when the query points in Q is clustered in a small region of the whole road network.

5.2. Baseline Algorithms

Recall that the min-sum OMP candidates consist of all the vertices in V and all the query points in Q. The baseline algorithm for min-sum OMP queries, denoted as BL_{sum}, evaluates $sd(v)$ for all $v \in V$ and evaluates $sd(q)$ for all $q \in Q$. It then returns the candidate point with the minimum sum-of-distances value as the min-sum OMP.

For min-max OMP queries, an OMP candidate is derived for each edge of a road network $G = (V, E)$. The baseline algorithm for min-max OMP queries, denoted as BL_{max}, checks each edge $(u, v) \in E$: if $(md(u) + md(v) - |uv|)/2 > md(p^*)$ where p^* is the best candidate currently found, then (u, v) is pruned according to Theorem 2; otherwise, the OMP candidate is computed and compared with p^*, and then p^* gets updated if necessary.

5.3. Euclidean Distance Bound Based Approach

Our first set of algorithms are designed to answer OMP queries in road networks whose edge length corresponds to the physical distance. The algorithms are based on *Best-First Search* (BFS) technique over an R-tree index. We now briefly review the concepts of BFS and Euclidean distance bound.

Best-First Search. BFS is an effective search technique for optimization problems over discrete data domains, and has been applied in the evaluation of various spatial queries.

Given a discrete data domain $O = \{o_1, \ldots, o_n\}$, suppose that we want to find a data object $o \in O$ such that o minimizes a target function $f(o)$. The BFS framework requires that, for each data object o, a tight lower bound of $f(o)$, denoted as $LB(o)$, can be efficiently computed. The data objects are then checked in non-decreasing order of $LB(o)$, since objects with smaller $LB(o)$ have a higher chance of being optimal. Meanwhile, the data point o^* with the minimum target function value currently found is maintained. The search stops as long as a data object o is checked to have $LB(o) \geq f(o^*)$, since all the non-checked data objects o' have target function values $f(o') \geq LB(o') \geq LB(o) \geq f(o^*)$.

The BFS framework has two benefits: (1) only a portion of the data objects are checked, and (2) the threshold value $f(o^*)$ decreases after checking each object, which increases the chance of pruning.

Although other search techniques are applicable for OMP query processing in road networks, they cannot outperform BFS. For example, a novel road network partitioning scheme is proposed in Xu and Jacobsen (2010), by which an OMP query only needs to access the graph fragments that collectively enclose all the points in Q. However, this method does not enjoy the second benefit of BFS mentioned above, and as is observed by Yan et al (2013), it is only correct in planar road networks without flyovers and tunnels. Our prior work (Yan et al, 2011a) also proposes two efficient search techniques to find high-quality meeting points but they do not guarantee result optimality. On the other hand, the algorithms proposed in this paper find the exact OMPs.

Euclidean Distance Bound. When applying our branch-and-bound algorithms, we require that the following network distance lower bound holds for any two points p_i and p_j in a road network,

$$d_N(p_i, p_j) \geq \|p_i p_j\|_2. \tag{20}$$

For any point p in a road network and a query point set Q, Equation (20) implies the following Euclidean distance lower bounds of $sd(p)$ and $md(p)$:

$$sd(p) = \sum_{i=1}^{n} d_N(q_i, p) \geq \sum_{i=1}^{n} \|q_i p\|_2. \tag{21}$$

$$md(p) = \max{}_{i=1}^{n} d_N(q_i, p) \geq \max{}_{i=1}^{n} \|q_i p\|_2. \tag{22}$$

Let e be an entry of an R-tree node, and let $e.B$ be the *Minimum Bounding Rectangle* (MBR) of e. Note that any object indexed under e is contained within $e.B$. Furthermore, let $mindist(e.B, q_i)$ be the minimum Euclidean distance between $e.B$ and q_i, which can be easily computed (Yan et al, 2011b). Then, we have the following Euclidean distance lower bounds for any point p contained within $e.B$:

$$sd(p) = \textstyle\sum_{i=1}^{n} d_N(q_i, p) \geq \sum_{i=1}^{n} mindist(e.B, q_i) \triangleq LB_{sum}(e, Q). \tag{23}$$

$$md(p) = \max_{i=1}^{n} d_N(q_i, p) \geq \max_{i=1}^{n} mindist(e.B, q_i) \triangleq LB_{max}(e, Q). \tag{24}$$

We use lighter notations, $LB_{sum}(e)$ and $LB_{max}(e)$, to denote the lower bounds on the RHS (right-hand side) of Equations (23) and (24), whenever Q is clear from the context. We also denote the lower bounds on the RHS of Equations (21) and (22) by $LB_{sum}(p)$ and $LB_{max}(p)$.

Branch-and-Bound Algorithms for Finding OMP. We first consider the algorithm for finding a min-sum OMP. Since the vertices in V are the candidates of the min-sum OMP, we first bulk-load an R-tree index T over all the vertices in V, using the sort-tile-recursive algorithm (Leutenegger et al, 1997). Note that each vertex is just a 2D point.

Our branch-and-bound algorithm for answering min-sum OMP queries, denoted as BB_{sum}, is given in Algorithm 1. We check the vertex candidates in Lines 5–14, following the BFS framework: a priority queue H is maintained during R-tree traversal, whose elements are given by (key, val), where $val = e$ is either an R-tree node or a vertex, and key is the BFS lower bound $LB_{sum}(e)$.

Algorithm 1 The Branch-and-Bound Algorithm (BB_{sum})

Input: a query set Q, a road network $G = (V, E)$,
an R-tree index T built on the vertices in V
Output: an OMP p^*

$p^* \leftarrow NULL$; $best \leftarrow \infty$
$H \leftarrow$ empty priority queue with elements of format (key, val)
for each entry e in root(T) **do**
  $H.enqueue(LB_{sum}(e), e)$
while H is not empty **do**
  $(LB_{sum}(e), e) \leftarrow H.dequeue()$
  if $LB_{sum}(e) \geq best$ **then**
    break
  if e is a vertex **then**
    Compute $sd(e)$
    Update p^* and $best$ if $sd(e) < best$
  else
    for each entry e' in the R-tree node that e points to **do**
      $H.enqueue(LB_{sum}(e'), e')$
for each query point $q \in Q$ **do**
  if $LB_{sum}(q) < best$ **then**
    Compute $sd(q)$
    Update p^* and $best$ if $sd(q) < best$
return p^*

Algorithm 2 The Branch-and-Bound Algorithm (BB_{max})

Input: a query set Q, a road network $G = (V, E)$,
an R-tree index T built on the edges in E
Output: an OMP p^*

$p^* \leftarrow NULL$; $best \leftarrow \infty$
$H \leftarrow$ empty priority queue with elements of format (key, val)
for each entry e in root(T) **do**
  $H.enqueue(LB_{max}(e), e)$
while H is not empty **do**
  $(LB_{max}(e), e) \leftarrow H.dequeue()$
  if $LB_{max}(e) \geq best$ **then**
    return p^*
  if e is an edge (u, v) **then**
    if $(md(u) + md(v) - |uv|)/2 < best$ **then**
      Compute OMP candidate p^c of (u, v)
      Compute $md(p^c)$
      Update p^* and $best$ if $md(p^c) < best$
  else
    for each entry e' in the R-tree node that e points to **do**
      $H.enqueue(LB_{max}(e'), e')$
return p^*

Elements with smaller $LB_{sum}(e)$ are processed first. We maintain the best meeting point p^* currently found, as well as the sum-of-distances value $best = sd(p^*)$, during the checking, until the BFS stopping condition is satisfied (Lines 7–8). After checking all the vertex candidates, we already have a tight threshold *best*, which is then used for "Euclidean distance bound" pruning when we check all the query point candidates (Lines 15–18).

We now consider the branch-and-bound algorithm for answering min-max OMP queries, denoted as BB_{max}, which is given in Algorithm 2. The algorithm is similar to BB_{sum}, except that the OMP candidates are computed from each edge (Line 11), rather than directly available. Since each edge contains an OMP candidate, we first build an R-tree T over all the edges in E, and during query processing, we check the edges using BFS over the R-tree T. Note that each edge is just a line segment, since in real road network datasets, a non-straight edge is usually modeled by a polyline, which explains why many vertices have degree 2.

We will demonstrate that the "Euclidean distance bound" technique is very effective in Section 6.2. Compared with the baseline algorithms, the branch-and-bound algorithms considerably improve the performance of answering both min-sum and min-max OMP queries. However, BB_{sum} and BB_{max} can only be applied when the network distance is lower bounded by Euclidean distance, as defined in Equation (20).

5.4. Threshold Algorithm Based Approaches

The branch-and-bound algorithms are only applicable when the edge length of a road network corresponds to the physical distance. However, this assumption may not always hold. For example, the edge length may refer to travel delay.

Therefore, we develop our second set of algorithms to work on arbitrary road networks, based on the *Threshold Algorithm* (or TA) for top-k queries. We first review Fagin's TA (Fagin et al, 2003).

Threshold Algorithm. We are given a relational table with schema $(A_1, A_2, \ldots, A_n)$ along with n lists of all the tuples in the table, where each list L_i sorts the tuples in non-decreasing order of the value of attribute A_i. Fagin's TA picks the top-1 tuple t with the smallest score $\sum_{i=1}^{n} A_i(t)$ (or $\max_{i=1}^{n} A_i(t)$), by accessing the next tuple of the lists L_i in a round-robin fashion, until the score lower bound of all the unchecked tuples becomes larger than the best score currently found. Suppose that the last element accessed in L_i is t_i, then the score lower bound is computed as $\sum_{i=1}^{n} A_i(t_i)$ (or $\max_{i=1}^{n} A_i(t_i)$).

Application of TA for Finding OMPs. Our algorithms concurrently and incrementally expand the network around each $q_i \in Q$ using Dijkstra's algorithm. For each q_i, the vertices are visited in non-decreasing order of $d_N(q_i, v)$ in the expansion. Here, a vertex v is analogous to a tuple in TA, $d_N(q_i, v)$ is analogous to the attribute value $A_i(v)$, and the list L_i in TA corresponds to the sequence of vertices v with non-decreasing $d_N(q_i, v)$. In contrast to TA, we do not check the lists in a round-robin fashion. Let n_i be the next vertex to visit in the network expansion of q_i, then we pick the vertex n_j to check in each iteration, where $j = \min_i\{d_N(n_i, q_i)\}$.

To realize this traversal order, for each query point $q_i \in Q$, we maintain a *shortest path wrapper* w_i for incremental SSSP computation with source q_i. The wrapper w_i supports two operations. First, $w_i.top()$ returns the next vertex

n_i whose network distance $d_N(q_i, n_i)$ is to be computed, and returns $NULL$ when the network distances to all the vertices in V are computed. Second, $w_i.forward()$ computes $d_N(q_i, n_i)$ and updates the distance estimations for all the vertices adjacent to n_i, which corresponds to one round of Dijkstra's algorithm.

Unlike our baseline algorithms and branch-and-bound algorithms presented in Sections 5.2 and 5.3, which use pay-as-you-go SSSP computation implicitly when computing $sd(v)$ (or $md(v)$) for some OMP candidate v, our TA-based algorithms use the shortest path wrapper explicitly for traversal, rather than for network distance computation.

Note that each vertex v is visited for at most $|Q|$ times, upon which time $d_N(q_i, v)$ is available for all q_i, and $sd(v)$ (or $md(v)$) is computed.

If a vertex v is an OMP candidate, we add v to a candidate set S when v is visited for the first time, and the evaluation of $sd(v)$ (or $md(v)$) is delayed until v is visited for the $|Q|$-th time. We maintain a lower bound for vertex v which is initialized as the Euclidean distance bound (cf. Equation (21) or (22)) when v is visited for the first time, and the bound is tightened by replacing $\|q_i v\|_2$ with $d_N(q_i, v)$ whenever q_i expands to v. Let p^* be the best meeting point currently found, then v is removed from S if the tightened lower bound is larger than $sd(p^*)$ (or $md(p^*)$). We say that v is pruned in this case.

For an OMP candidate p on edge (u, v), we can only compute $sd(p)$ or $md(p)$ when both u and v are visited for Q times. To realize this operation, whenever a vertex is visited for the first time, for any edge e adjacent to it, we put e into S if e may contain an OMP candidate.

Let n_j (from w_j) be the next vertex to check, then we have the following stop condition:

Theorem 3. If $S = \emptyset$ and $d_N(n_j, q_j) \geq sd(p^*)/|Q|$ (or $d_N(n_j, q_j) \geq md(p^*)$), then p^* is the OMP.

Proof. First, $S = \emptyset$ implies that all the checked candidates are pruned.

For any non-visited candidate p on an edge (u, v), it holds that neither u nor v is visited, since edge (u, v) would be added into S whenever there exists an OMP candidate on (u, v) and u or v is checked.

Furthermore, such an edge (u, v) does not contain any query point q_i, or otherwise, p^* is not assigned a value yet since no vertex has ever been fetched from wrapper w_i.

Thus, for any query point q_i, we have $d_N(p, q_i) \geq \min\{d_N(u, q_i), d_N(v, q_i)\} \geq d_N(n_j, q_j)$, which implies $sd(p) \geq sd(p^*)$ when $d_N(n_j, q_j) \geq sd(p^*)/|Q|$ (or, $md(p) \geq md(p^*)$ when $d_N(n_j, q_j) \geq md(p^*)$).

It follows the proof, since p is an arbitrary non-visited candidate. □

TA-Based Algorithms for Finding OMP. Algorithm 3 shows our TA-based algorithm for min-sum OMP queries, denoted as TA_{sum}. A priority queue H is used to maintain the traversal order, which contains the next vertex to visit, n_j, for each query point q_j. H and the shortest path wrappers w_i are initialized in Lines 3–7, and whenever a vertex n_j is processed, H gets the next vertex to visit from w_j in Lines 38–40. Each min-sum OMP candidate v (note that v is either a vertex or a query point q_i) is associated with a counter $counter(v)$ to record the number of times it is visited, $bound(v)$ to record the sum of the non-updated Euclidean distance bounds, and $sd(v)$ to record the sum of the actual network distances already obtained. Lines 9–22 process the current vertex n_j, and Lines

Algorithm 3 The TA-Based Algorithm (TA_{sum})

Input: a query set Q, a road network $G = (V, E)$,
Output: an OMP p^*

$p^* \leftarrow NULL$; $best \leftarrow \infty$; $S \leftarrow \emptyset$
$H \leftarrow$ empty priority queue with elements of format (key, val)
for each query point $q_i \in Q$ **do**
  Initialize shortest path wrapper w_i with source q_i
  $n_i \leftarrow w_i.top()$
  if $n_i \neq NULL$ **then**
    $H.enqueue(d_N(n_i, q_i), (n_i, q_i))$; $w_i.forward()$
while H is not empty **do**
  $(n_j, q_j) \leftarrow H.dequeue()$
  if $S = \emptyset \wedge d_N(n_j, q_j) \geq best/|Q|$ **then**
    return p^*
  if $counter(n_j) = 0$ **then**
    $bound(n_j) \leftarrow \sum_{i=1}^{n} \|q_i n_j\|_2$; $sd(n_j) \leftarrow 0$
    $S \leftarrow S \cup n_j$ if $bound(n_j) < best$
  if $n_j \in S$ **then**
    $bound(n_j) \leftarrow bound(n_j) - \|q_j n_j\|_2$
    $sd(n_j) \leftarrow sd(n_j) + d_N(n_j, q_j)$
    $S \leftarrow S - n_j$ if $sd(n_j) + bound(n_j) \geq best$
  $counter(n_j) \leftarrow counter(n_j) + 1$
  if $n_j \in S \wedge counter(n_j) = |Q|$ **then**
    Update p^* and $best$ if $sd(n_j) < best$
    $S \leftarrow S - n_j$
  for each edge (u, v) adjacent to n_j containing a query point q_i **do**
    if both $d_N(u, q_i)$ and $d_N(v, q_i)$ are computed by w_i **then**
      if $q_i \in S$ **then**
        Evaluate $d_N(q_i, q_j)$
        $bound(q_i) \leftarrow bound(q_i) - \|q_i q_j\|_2$
        $sd(q_i) \leftarrow sd(q_i) + d_N(q_i, q_j)$
        $S \leftarrow S - q_i$ if $sd(q_i) + bound(q_i) \geq best$
      $counter(q_i) \leftarrow counter(q_i) + 1$
      if $q_i \in S \wedge counter(q_i) = |Q|$ **then**
        Update p^* and $best$ if $sd(q_i) < best$
        $S \leftarrow S - q_i$
    else
      if $counter(q_i) = 0$ **then**
        $bound(q_i) \leftarrow \sum_{k=1}^{n} \|q_k q_i\|_2$; $sd(q_i) \leftarrow 0$
        $S \leftarrow S \cup q_i$ if $bound(q_i) < best$
  $n_j \leftarrow w_j.top()$
  if $n_j \neq NULL$ **then**
    $H.enqueue(d_N(n_j, q_j), (n_j, q_j))$; $w_j.forward()$
return p^*

Algorithm 4 The TA-Based Algorithm (TA_{max})

Input: a query set Q, a road network $G = (V, E)$
Output: an OMP p^*

1: $p^* \leftarrow NULL$; $best \leftarrow \infty$; $S \leftarrow \emptyset$
2: $H \leftarrow$ empty priority queue with elements of format (key, val)
3: **for each** query point $q_i \in Q$ **do**
4:     Initialize shortest path wrapper w_i with source q_i
5:     $n_i \leftarrow w_i.top()$
6:     **if** $n_i \neq NULL$ **then**
7:         $H.enqueue(d_N(n_i, q_i),\ (n_i, q_i))$; $w_i.forward()$
8: **while** H is not empty **do**
9:     $(n_j, q_j) \leftarrow H.dequeue()$
10:     **if** $S = \emptyset \wedge d_N(n_j, q_j) \geq best$ **then**
11:         **return** p^*
12:     **if** $counter(n_j) = 0$ **then**
13:         $md(n_j) \leftarrow \max_{i=1}^{n} \|q_i n_j\|_2$;
14:         **for each** edge e adjacent to n_j **do**
15:             {Let v be the other endpoint of e}
16:             **if** $\frac{md(n_j)+md(v)-|n_j v|}{2} <= best$ **then**
17:                 $S \leftarrow S \cup e$
18:     **else**
19:         $md(n_j) \leftarrow d_N(n_j, p_j)$ if $d_N(n_j, p_j) > md(n_j)$
20:         **for each** edge e adjacent to n_j **do**
21:             {Let v be the other endpoint of e}
22:             **if** $e \in S \wedge \frac{md(n_j)+md(v)-|n_j v|}{2} > best$ **then**
23:                 $S \leftarrow S - e$
24:     $counter(n_j) \leftarrow counter(n_j) + 1$
25:     **if** $counter(n_j) = |Q|$ **then**
26:         **for each** edge e adjacent to n_j **do**
27:             {Let v be the other endpoint of e}
28:             **if** $e \in S \wedge counter(v) = |Q|$ **then**
29:                 Evaluate the OMP candidate on e
30:                 Update p^* and $best$ if necessary
31:                 $S \leftarrow S - e$
32:     $n_j \leftarrow w_j.top()$
33:     **if** $n_j \neq NULL$ **then**
34:         $H.enqueue(d_N(n_j, q_j),\ (n_j, q_j))$; $w_j.forward()$
35: **return** p^*

23–37 process the query points q_i on edges (u, v) adjacent to n_j, among which Lines 24–33 correspond to the case where both endpoints of (u, v) are visited by q_j, and Lines 35–37 correspond to the case where only one endpoint is visited by q_j.

Algorithm 4 shows our TA-based algorithm for min-max OMP queries, denoted TA_{max}. The algorithm is similar to Algorithm 3, except that the candidate set S is now a set of edges that contain OMP candidates rather than the OMP candidates themselves. Also, note that $md(v)$ is just the lower bound for vertex v that gets initialized in Line 13 and updated in Line 19 (according to Equation (22)), and it is needed only for pruning in Lines 16 and 22 (according to Theorem 2).

It is worth noting that the TA-based algorithms are proposed to study the potential of using the TA technique to improve the performance of answering OMP queries. However, there is no guarantee that the TA technique is always effective. In fact, our experiments in Section 6.2 show that, TA_{max} is much faster than BL_{max}, but TA_{sum} is not effective.

6. Experiments

In this section, we evaluate the performance of our algorithms using both real and synthetic datasets. We find that the weights of query points do not significantly influence the performance of OMP query processing and, thus, we only report the experiments on unweighted OMP queries.

We randomly generate query points in a spatial setting. Since the experimental results on query sets generated with biased distribution are similar to those on uniform query sets, we only report the experiments with uniformly generated query sets. For each experimental setting, the reported results are averaged over 100 randomly generated queries.

All the experiments were done on a computer with 3Hz Intel CPU and 3GB memory. All our programs were written in JAVA, and run on CentOS 5.7.

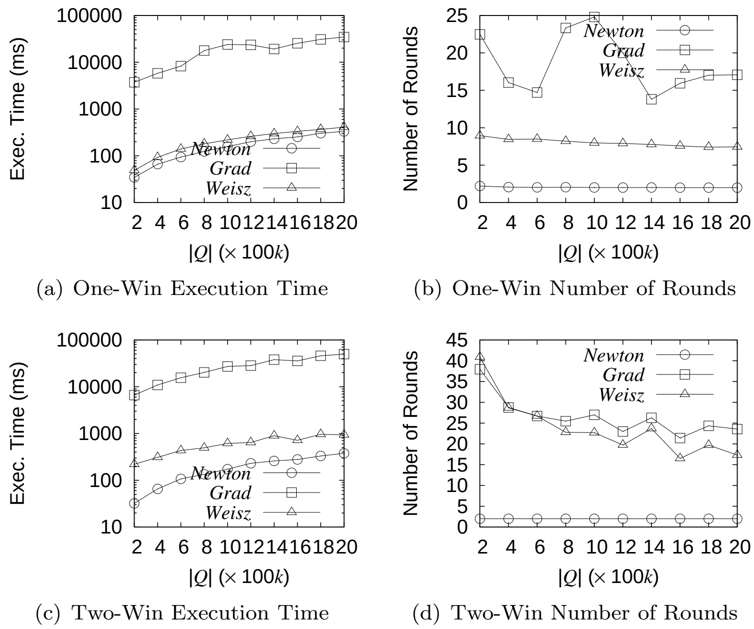
6.1. Performance of Answering OMP Queries in Euclidean Space

Query Generator. We generate two kinds of query point sets. In the first setting, $|Q|$ query points are randomly generated in the domain $[0, 1] \times [0, 1]$. In the second setting, we generate k groups of clustered points. Specifically, k square windows with side length δ $(0 < \delta < 1)$ are generated within the domain $[0, 1] \times [0, 1]$, and then $|Q|/k$ query points are randomly generated in each window. In our experiments, we set $\delta = 20\%$ and $k = 2$, and fix the tolerance parameter ϵ of our gradient descent framework to be 10^{-6}.

Results of Min-sum OMP Queries. From now on, we denote our gradient descent method for min-sum OMP queries by *Grad*, Newton's method by *Newton*, and Weiszfeld's algorithm by *Weisz*.

Figure 8 shows the experimental results of our algorithms for min-sum OMP queries, when the query set size $|Q|$ varies. In this set of experiments, we select $|Q|$ from $\{200k, 400k, \ldots, 2000k\}$. Figures 8(a) and (b) show the running time and number of rounds of the three algorithms when the query points are randomly generated in a $[0, 1] \times [0, 1]$ window. We can see that *Newton* is faster than *Weisz*, and both algorithms are two orders of magnitude faster than the gradient descent method; furthermore, *Newton* always stops in 2–3 rounds with the help of the Hessian matrix.

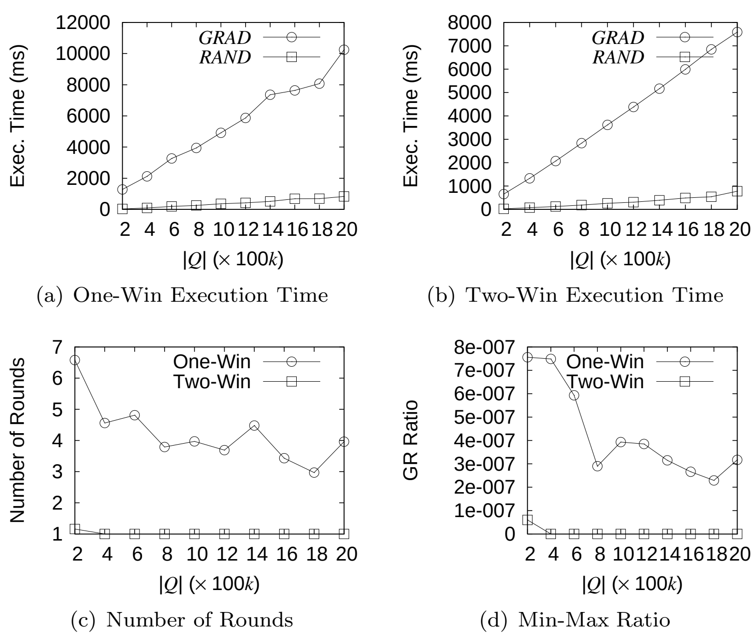
Figures 8(c) and (d) show the running time and number of rounds of the three algorithms when the query points are generated in two windows, each with $|Q|/2$ query points. The results are similar to those of Figures 8(a) and (b), except that the benefit of *Newton* is more prominent: *Newton* is an order of magnitude faster



(a) One-Win Execution Time (b) One-Win Number of Rounds

(c) Two-Win Execution Time (d) Two-Win Number of Rounds

Fig. 8. Min-sum OMP Query Results in Euclidean Space



(a) One-Win Execution Time (b) Two-Win Execution Time

(c) Number of Rounds (d) Min-Max Ratio

Fig. 9. Min-Max OMP Query Results in Euclidean Space

than *Weisz*. This observation verifies that *Newton* is the best choice for min-sum OMP, and performs extremely well with biased query point distribution.

To sum up, we recommend to use *Newton* in applications that require finding min-sum OMP (weighted or unweighted). For example, the SPM algorithm proposed in Papadias et al (2004) and Papadias et al (2005) for answering ANN queries, requires to find the min-sum OMP first, for the purpose of search space pruning. The gradient descent method was originally used in Papadias et al (2004) and Papadias et al (2005) to find the min-sum OMP, while Newton's method is actually a more efficient method.

Results of Min-Max OMP Queries. From now on, we denote our gradient descent method for min-max OMP queries by *GRAD*, and denote Welzl's randomized algorithm by *RAND*. Furthermore, given the meeting point p returned by *GRAD* and the exact OMP p^* computed by *RAND*, we define the *GRAD*:*RAND* ratio (or simply the *GR ratio*) to evaluate the quality of p, and the ratio is given by:

$$\frac{\max_{i=1}^{n} \|q_i, p\|_2}{\max_{i=1}^{n} \|q_i, p^*\|_2} - 1. \tag{25}$$

The quality of the meeting point p is higher when the GR ratio is closer to 0.

Figure 9 shows the experimental results of our algorithms for min-max OMP queries, when the query set size $|Q|$ varies. In this set of experiments, we select $|Q|$ from $\{200k, 400k, \ldots, 2000k\}$. Figure 9(a) shows the running time of both algorithms when the query points are randomly generated in one window of $[0,1] \times [0,1]$, where we can see that *RAND* is over an order of magnitude faster than *GRAD*. Figure 9(b) shows the running time of both algorithms when the query points are generated in two windows, each of which has $|Q|/2$ query points, where we obtain the same observation as in Figure 9(a). The results are within the expectation, since the expected time complexity of *RAND* is $O(|Q|)$.

Figure 9(c) shows the number of rounds of *GRAD* when the query points are generated in one window and two windows: *GRAD* usually stops after one round in the two-window case, due to our choice of starting point and the application of Observation 1 for early termination (i.e. in many cases the starting point is already the exact OMP).

Figure 9(d) shows the GR ratio when the query points are generated in one window and two windows: the ratio value is very small, and thus the result OMPs of *GRAD* are of high quality. Note that the GR ratio is smaller in the two-window case, due to our method of picking starting point and the application of Observation 1.

To sum up, we recommend to use *RAND* for finding unweighted min-max OMP, but since *RAND* is not applicable for weighted min-max OMP queries, we have to use our gradient descent method to find the weighted min-max OMP.

6.2. Performance of Answering OMP Queries in Road Networks

Real Road Network Datasets. We evaluate the performance of our algorithms for the OMP queries in road networks, using the five road network datasets from Li et al (2005), and the railroad and highway networks from CTA Trans-

Table 1. Real Road Network Datasets

Name	$\lvert V\rvert$	$\lvert E\rvert$
California (CA)	21048	21693
North America (NA)	175813	179179
Oldenburg (OL)	6105	7035
San Francisco (SF)	174956	223001
San Joaquin County (TG)	18263	23874
RailRoad Network (RWay)	25785	32249
Highway Network (HWay)	74028	111936

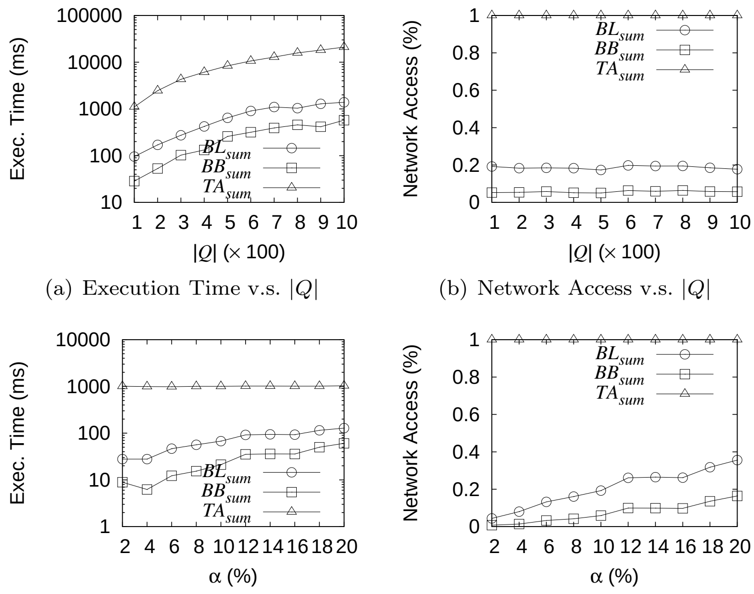
portation Networks [1]. Table 1 summarizes the seven datasets, which include the road networks of a continent (NA), a US state (CA), a US city (SF), a US county (TG), a European city (OL), and other types of networks (RWay and HWay). Although we use different kinds of network datasets, the experimental results are quite consistent over all kinds of network datasets. Therefore, we only show our experimental results for the CA dataset. The complete results are given in `http://www.cse.ust.hk/~yanda/datasets/summary.pdf`.

Query Generator. For each dataset, we generate query point sets by randomly generating a square window on the road network, and then randomly generate $|Q|$ query points on the part of the road network in the window. Let W denote the difference between the x-coordinates of the leftmost vertex and the rightmost vertex in the road network, and let H denote the difference between the y-coordinates of the highest vertex and the lowest vertex in the road network. Then, given a query window parameter α, the size of the query window is set to $\alpha W \times \alpha H$, and it is randomly generated within the minimum bounding box of the road network.

Measures. We define *network access* as the proportion of vertices in a road network that are visited during OMP query evaluation (i.e. by the SSSP computation with sources q_i). Note that if the road network is stored on a disk as in Yiu et al (2005), Shekhar and Liu (1997) and Papadias et al (2003), *network access* can be directly translated into the number of I/O operations, where each I/O operation accesses the adjacency list of a visited vertex.

Results of Min-sum OMP Queries. Figures 10(a) and (b) show the running time and network access of min-sum OMP queries when the window parameter $\alpha = 10\%$, and $|Q|$ varies from 100 to 1000. Figure 10(a) shows that the running time of all three algorithms increase as $|Q|$ increases. From Figure 10(a), we can see that TA_{sum} is over one order of magnitude slower than the other two algorithms, which is because of the loose lower bound in the stopping criteria. Besides, BB_{sum} is twice as fast as the baseline BL_{sum}. Figure 10(b) shows that the network access is insensitive to $|Q|$, and that TA_{sum} in most cases accesses the whole road network, while the other algorithms access just a small fraction of the network. Note that even the baseline BL_{sum} accesses just 20% (rather than

[1] `http://cta.ornl.gov/transnet/`



(a) Execution Time v.s. $|Q|$ (b) Network Access v.s. $|Q|$

(c) Execution Time v.s. α (d) Network Access v.s. α

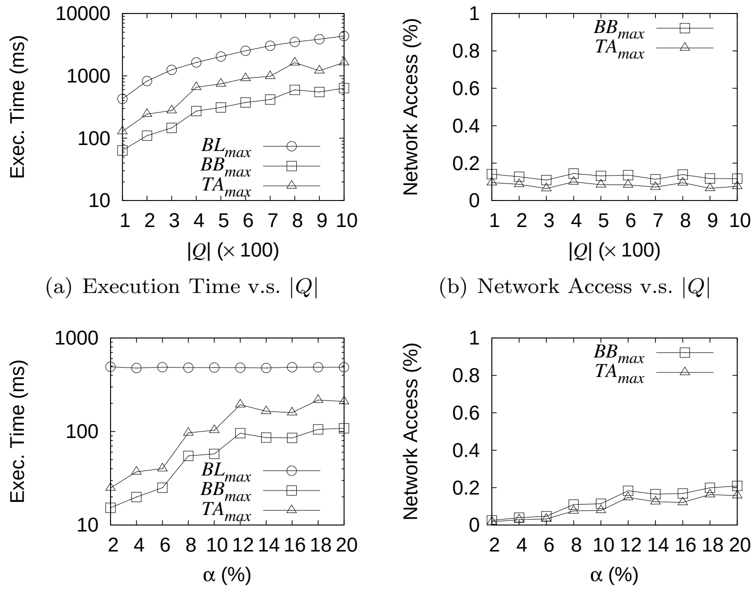
Fig. 10. Min-sum OMP Query Results for the CA dataset

100%) of the road network. This advantage is attributed to our pruning method in the evaluation of $sd(v)$: when checking vertex v, if we find that the partial summation $\sum_{i=1}^{k} d_N(q_i, v)$ $(k < n)$ is not smaller than the current optimal $sd(p^*)$, v is pruned without evaluating $d_N(q_i, v)$ $(k < i < n)$.

Figures 10(c) and (d) show the running time and network access of min-sum OMP queries when $|Q|$ is fixed to 100, and the window parameter α varies from 2% to 20%. Figure 10(c) shows that the running time of TA_{sum} is insensitive to α, which is because it always access the whole network as shown in Figure 10(d). On the other hand, Figure 10(d) shows that BL_{sum} and BB_{sum} access a larger fraction of the road network as α increases, which is because the query points spread out over a larger portion of the network, which has to be accessed by BL_{sum} and BB_{sum}. As a result of accessing a larger fraction of the network, the running time of BL_{sum} and BB_{sum} also increases as α increases.

Since TA_{sum} is always slower than the baseline BL_{sum}, it should not be used in min-sum OMP queries. The bad performance of TA_{sum} is due to its loose stopping threshold, and is also observed in Yiu et al (2005) for ANN queries. On the other hand, BB_{sum} always performs the best, and should be used whenever the edge length of a road network is based on the physical distance. Otherwise, BL_{sum} is the proper choice.

Results of Min-Max OMP Queries. Figures 11(a) and (b) show the running time and network access of min-max OMP queries when $\alpha = 10\%$, and $|Q|$ varies from 100 to 1000. Since BL_{max} checks all the edges $e \in E$ using Theorem 2, $md(v)$ has to be evaluated for all $v \in V$. As a result, the network access of BL_{max} is always 100% and, thus, we do not show it in Figure 11(b). Unlike TA_{sum}, TA_{max} is faster than the baseline BL_{max}, since its lower bound



(a) Execution Time v.s. $|Q|$ (b) Network Access v.s. $|Q|$

(c) Execution Time v.s. α (d) Network Access v.s. α

Fig. 11. Min-max OMP Query Results for the CA dataset

in the stopping criteria is much tighter: an edge is inserted and maintained in candidate set S, only if points on the edge is closer than the current optimal $md(p^*)$ from **all** the query points (checked by Theorem 2). Figures 11(a) and (b) show that BB_{max} always performs the best, while TA_{max} is better than BL_{max} in terms of both running time and network access. Figures 11(c) and (d) show the running time and network access of min-max OMP queries when $|Q| = 100$, and α varies from 2% to 20%, where we obtain similar results.

To sum up, BB_{max} is desirable for processing min-max OMP queries whenever the edge length of a road network is based on the physical distance. Otherwise, TA_{max} is the proper choice.

6.3. Impact of Outliers to the OMP Algorithms for Road Networks

In this subsection, we study how sensitive our pruning techniques are to outlier(s) in a query group.

Query Generator. Let the data domain be a $W \times H$ rectangle. We define five zones in the data domain as illustrated in Figure 12, where each zone is a $20\%W \times 20\%H$ rectangular window. For each query Q, we generate $|Q| - 1$ query points randomly in Zone 1 and the last query point (as an outlier) in Zone i for $i \in \{1, 2, \ldots, 5\}$. The larger i is, the farther the outlier is to the other query points. Note that the data domain is no longer the MBR of all the vertex points, since some zone can be empty. For example, in HWay and RWay, many regions are oceans without network coverage. We thus choose the data domain a smaller

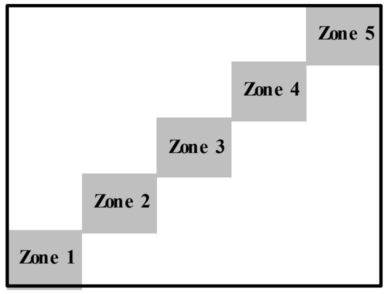
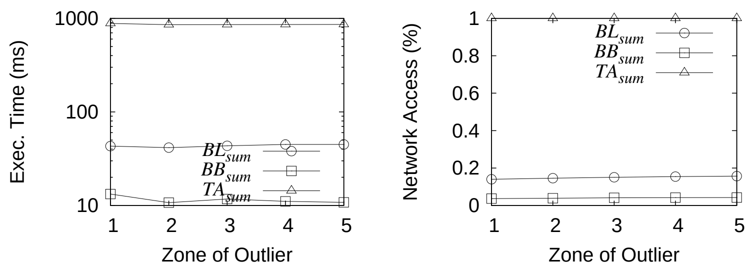


Fig. 12. Zones Used in the Query Generator



(a) Exec. Time (b) Network Access (Min-sum)

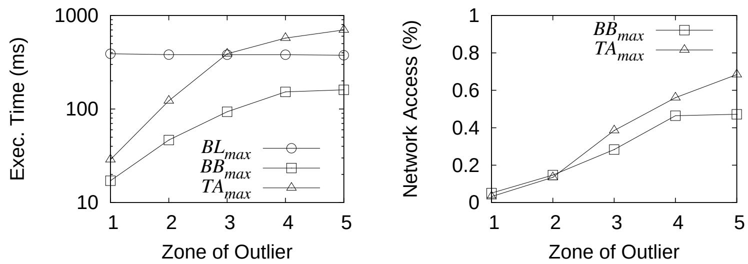
Fig. 13. Effect of Outlier on Min-sum OMP Algorithms on CA

dense region so that all zones are not empty. We just generate query points in the dense region, and the actual OMP is still allowed to be located in a sparse region.

Results of Min-sum OMP Queries. Figures 13(a) and (b) show the running time and network access of min-sum OMP queries when the outlier lies in different zones. For CA pick the data domain for query generation as $x = [-124, -120]$ and $y = [39, 42]$. We can see that for all the algorithms, the running time and network access is not significantly affected by the outlier location. In general, the running time and network access slightly increase when the outlier is farther way from the other query points.

Results of Min-max OMP Queries. Intuitively, the location of an outlier has a greater impact on the location of the min-max OMP than the min-sum OMP. Therefore, it is more important to study how sensitive the performance of min-max OMP algorithms is to the outlier location.

Figures 14(a) and (b) show the running time and network access of min-sum OMP queries when the outlier lies in different zones. As shown in Figure 14(a), except BL_{max} which already performs exhaustive search, the other two algorithms take significantly longer time when the outlier is farther away from the other query points. While BB_{max} is always faster than BL_{max}, TA_{max} is slower than BL_{max} when the outlier is very far away due to the additional overhead caused by Fagin's TA. This shows that the pruning technique of Fagin's TA is not effective when the query points are scattered over a large region. However, when the query points are not too scattered, TA_{max} can be over an order of



(a) Exec. Time (b) Network Access

Fig. 14. Effect of Outlier on Min-max OMP Algorithms on CA

magnitude faster than BL_{max}, and thus, it is thus preferred when BB_{max} is not applicable.

7. Conclusions

In this paper, we present a comprehensive study of OMP query processing in two spatial settings, Euclidean space and road networks. We utilize two new pruning techniques, Euclidean distance bound and threshold algorithm, to develop efficient algorithms for finding OMPs in road networks. The algorithms are efficient, since they only access part of the road networks and examine part of the candidates. We also propose a gradient-descent framework for answering weighted OMP queries in Euclidean space in general, and review the literature to identify the best algorithm for particular types of OMP queries. Finally, we find the best choice of the algorithms for each type of OMP query through extensive experiments on both real and synthetic datasets.

As a future work, we plan to study the performance of our proposed algorithms when the road network dataset is stored on a disk, especially focusing on how the I/O overhead impacts the overall cost of query processing.

Acknowledgements. This work is partially supported by GRF under grant numbers HKUST 617610.

References

Alsuwaiyel MH (1999) Algorithms: Design techniques and analysis. World Scientific Pub Co Inc.

Beck A, Teboulle M (2009) Gradient-based algorithms with applications to signal recovery. Convex Optimization in Signal Processing and Communications.

Boyd S, Vandenberghe L (2004) Convex optimization. Cambridge University Press.

Brimberg J, Love RF (1993) Global convergence of a generalized iterative procedure for the minisum location problem with lp distances. Operations Research, 41(6): 1153–1163.

Chen R (1984a) Location problems with costs being sums of powers of Euclidean distances. Computers & operations research, 11(3): 285–294.

Chen R (1984b) Solution of location problems with radial cost functions. Computers & Mathematics with Applications, 10(1): 87–94.

Cohen E, Halperin E, Kaplan H, Zwick U (2003) Reachability and distance queries via 2-hop labels. SIAM Journal on Computing, 32(5): 1338–1355.

Cooper L (1968) An extension of the generalized weber problem. Journal of Regional Science, 8(2): 181–197.

De Berg M, Cheong O, van Kreveld M, Overmars M (2008) Computational geometry. Springer Berlin Heidelberg.

Deng K, Sadiq S, Zhou X, Xu H, Fung GPC, Lu Y (2012) On group nearest group query processing. IEEE Transactions on Knowledge and Data Engineering, 24(2): 295–308.

Drezner Z, Shelah S (1987) On the complexity of the Elzinga-Hearn algorithm for the 1-center problem. Mathematics of Operations Research, 12(2): 255–261.

Du Y, Zhang D, Xia T (2005) The optimal-location query. Advances in Spatial and Temporal Databases, pp 163–180. Springer Berlin Heidelberg.

Fagin R, Lotem A, Naor M (2003) Optimal aggregation algorithms for middleware. Journal of Computer and System Sciences, 66(4): 614–656.

Geisberger R, Sanders P, Schultes D, Delling D (2008) Contraction hierarchies: Faster and simpler hierarchical routing in road networks. Experimental Algorithms, pp 319–333. Springer Berlin Heidelberg.

Kellaris G, Mouratidis K (2010) Shortest path computation on air indexes. Proceedings of the VLDB Endowment, 3(1-2): 747-757.

Leutenegger ST, Lopez MA, Edgington J (1997). STR: A simple and efficient algorithm for R-tree packing. IEEE 13th international conference on data engineering (ICDE), April 1997, pp 497–506.

Li F, Cheng D, Hadjieleftheriou M, Kollios G, Teng SH (2005) On trip planning queries in spatial databases. Advances in spatial and temporal databases, pp 273–290. Springer Berlin Heidelberg.

Li F, Yao B, Kumar P (2011a) Group enclosing queries. IEEE Transactions on Knowledge and Data Engineering, 23(10): 1526-1540.

Li Y, Li F, Yi K, Yao B, Wang M (2011b) Flexible aggregate similarity search. Proceedings of the 2011 ACM SIGMOD international conference on management of data, June 2011, pp 1009–1020).

Li J, Yiu ML, Mamoulis N (2013) Efficient notification of meeting points for moving groups via independent safe regions. IEEE 29th international conference on data engineering (ICDE), 2013, pp 422–433.

Lian X, Chen L (2008) Probabilistic group nearest neighbor queries in uncertain databases. IEEE Transactions on Knowledge and Data Engineering, 20(6): 809–824.

Megiddo N (1982) Linear-time algorithms for linear programming in R3 and related problems". Proceedings of 23rd annual IEEE symposium on foundations of computer science, November 1982, pp 329–338.

Ostresh LM (1977) The mulitfacility location problem: applications and descent theorems. Journal of Regional Science, 17(3): 409–419.

D. Papadias, Q. Shen, Y. Tao and K. Mouratidis. "Group nearest neighbor queries". In *ICDE*, 2004. Papadias D, Shen Q, Tao Y, Mouratidis K 2004 Group nearest neighbor queries. 20th international conference on data engineering, March 2004, pp 301–312.

Papadias D, Tao Y, Mouratidis K, Hui CK (2005) Aggregate nearest neighbor queries in spatial databases. ACM Transactions on Database Systems (TODS), 30(2): 529–576.

Papadias D, Zhang J, Mamoulis N, Tao Y (2003) Query processing in spatial network databases. Proceedings of 29th international conference on very large data bases, Volume 29, VLDB Endowment, September 2003, pp 802–813.

Samet H, Sankaranarayanan J, Alborzi H (2008) Scalable network distance browsing in spatial databases. Proceedings of the 2008 ACM SIGMOD international conference on management of data, June 2008, pp 43–54.

Sankaranarayanan J, Samet H, Alborzi H (2009) Path oracles for spatial networks. Proceedings of the VLDB Endowment, 2(1): 1210–1221.

Shamos MI, Hoey D (1975) Closest point problems. Proceedings of 16th annual IEEE symposium on foundations of computer science, October 1975, pp 151–162.

Shekhar S, Liu DR (1997) CCAM: A connectivity-clustered access method for networks and network computations. IEEE Transactions on Knowledge and Data Engineering, 9(1): 102–119.

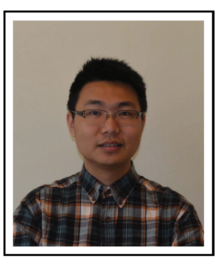
Tao Y, Sheng C, Pei J (2011) On k-skip shortest paths. Proceedings of the 2011 ACM SIGMOD international conference on management of data, June 2011, pp 421–432.

Wesolowsky GO (1982) Location problems on a sphere. Regional Science and Urban Economics, 12(4): 495–508.

Wong RCW, Özsu MT, Yu PS, Fu AWC, Liu L (2009) Efficient method for maximizing bichromatic reverse nearest neighbor. Proceedings of the VLDB Endowment, 2(1): 1126–1137.
Xiao X, Yao B, Li F (2011) Optimal location queries in road network databases. IEEE 27th international conference on data engineering (ICDE), April 2011, pp 804–815.
Xu Z, Jacobsen HA (2010). Processing proximity relations in road networks. Proceedings of the 2010 ACM SIGMOD international conference on management of data, June 2010, pp 243–254.
Yan D, Zhao Z, Ng W (2011a) Efficient algorithms for finding optimal meeting point on road networks. Proceedings of the VLDB Endowment, 4(11).
Yan D, Wong RCW, Ng W (2011b). Efficient methods for finding influential locations with adaptive grids. Proceedings of the 20th ACM international conference on Information and knowledge management, October 2011, pp 1475–1484.
Yan D, Cheng J, Ng W, Liu S (2013) Finding distance-preserving subgraphs in large road networks. IEEE 29th international conference on data engineering (ICDE), 2013, pp 625–636.
Yiu ML, Mamoulis N, Papadias D (2005) Aggregate nearest neighbor queries in road networks. IEEE Transactions on Knowledge and Data Engineering, 17(6): 820–833.

Author Biographies

Da Yan received his B.S. degree in Computer Science from Fudan University, Shanghai, in 2009. He is currently a Ph.D. student in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. His research interests include big data, spatial data management, uncertain data management and data mining.



Zhou Zhao received his B.S. degree in Computer Science from the Hong Kong University of Science and Technology (HKUST), in 2010. He is currently a Ph.D. student in the Department of Computer Science and Engineering, HKUST. His research interests include data cleansing and data mining.

Wilfred Ng received his MS.c. (Distinction) and Ph.D. in Computer Science from the University of London. Currently he is an Associate Professor of Computer Science and Engineering at the Hong Kong University of Science and Technology, where he is a member of the database research group. His research interests are in the areas of databases, data mining and information Systems, which include Web data management and XML searching. Further Information can be found at the following URL: `http://www.cs.ust.hk/faculty/wilfred/index.html`.

Correspondence and offprint requests to: Da Yan, Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong. Email: yanda@cse.ust.hk