Geoinformatica
https://doi.org/10.1007/s10707-020-00416-9

Size constrained k simple polygons

KwangSoo Yang[1] · Kwang Woo Nam[2] · Ahmad Qutbuddin[3] · Aaron Reich[4] · Valmer Huhn[1]

Received: 5 September 2019 / Revised: 6 April 2020 / Accepted: 16 June 2020 /
Published online: 14 July 2020

Abstract
Given a geometric space and a set of weighted spatial points, the Size Constrained k Simple Polygons (SCkSP) problem identifies k simple polygons that maximize the total weights of the spatial points covered by the polygons and meet the polygon size constraint. The SCkSP problem is important for many societal applications including hotspot area detection and resource allocation. The problem is NP-hard; it is computationally challenging because of the large number of spatial points and the polygon size constraint. Our preliminary work introduced the Nearest Neighbor Triangulation and Merging (NNTM) algorithm for SCkSP to meet the size constraint while maximizing the total weights of the spatial points. However, we find that the performance of the NNTM algorithm is dependent on the t-nearest graph. In this paper, we extend our previous work and propose a novel approach that outperforms our prior work. Experiments using Chicago crime and U.S. Federal wildfire datasets demonstrate that the proposed algorithm significantly reduces the computational cost of our prior work and produces a better solution.

Keywords Spatial covering · Constrained optimization · Simple polygon

1 Introduction

Given a geometric space and a set of weighted spatial points, the Size Constrained k Simple Polygons (SCkSP) problem finds k simple polygons that honor the size constraint and maximize the total weights of the spatial points covered by the k simple polygons. Figure 1a shows an example input of SCkSP consisting of a geometric space with 12 spatial points (i.e., $A, B, \ldots, L$). Every spatial point is associated with a weight as indicated by the

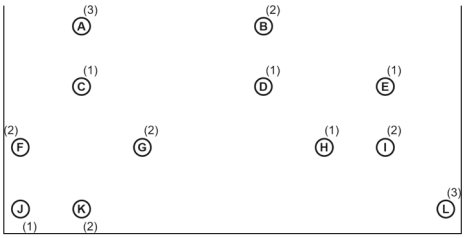
✉ KwangSoo Yang
yangk@fau.edu

[1] Computer Science, Florida Atlantic University, 777 Glades Road, Boca Raton, FL 33431-0991, USA

[2] School of Computer, Information, and Communications Engineering, Kunsan National University, 558 Daehak-ro, Gunsan, 55140, South Korea

[3] Computer Engineering, Florida Atlantic University, Boca Raton, FL 33431, USA

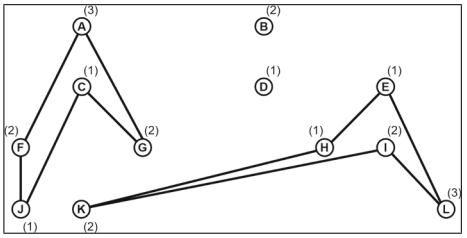
[4] Computer Science, Georgia Institute of Technology, Atlanta, GA 30332, USA



(a) Input

Node ID	x	y	weight	Node ID	x	y	weight
A	1	4	3	G	2	2	2
B	4	4	2	H	5	2	1
C	1	3	1	I	6	2	2
D	4	3	1	J	0	1	1
E	6	3	1	K	1	1	2
F	0	2	2	L	7	1	3

(b) Point Locations



(c) Output

Fig. 1 Example of the input and output of SCkSP

number displayed over the point. Figure 1b shows the location and weight of each spatial point. Let k be 2. Assume that the size of a polygon never exceeds 1.5. Figure 1c shows an example output of SCkSP where two simple polygons cover 10 spatial points. The total point-weights covered by the two simple polygons is 18 and the size of the polygons meets the size constraint (i.e.,1.5).

A solution to this problem would provide a quick way to identify local hotspots and provide insight into the optimal allocation of resources in order to address certain issues. This would have many important societal applications including hotspot detection and resource allocation for crime incidents and car accidents. The SCkSP problem is NP-hard (a proof is provided in Section 1.4). The problem is computationally challenging because of the large number of spatial points and the polygon size constraint.

1.1 Representative application domains

The SCkSP problem can be applied to many real-life scenarios including the following:

- Hotspot detection: The proposed method accurately finds the local highest concentration of car accidents or incidents of crime. The identification of car accident hotspots helps city or state governments determine roads for construction. The identification of crime hotspots allows for county, state, or federal governments to more precisely label particular areas of high criminal activity.
- Resource allocation management for local governments: Effective applications of SCkSP include the determination of the optimal allocation of construction funding for roads that are most in need of repair in order to improve safety conditions. Another application involves the identification of the appropriate allocation of law enforcement officials to best lower crime in vulnerable areas.
- Air pollution dispersion plume tracking: Another interesting application of SCkSP is the analysis of polluted air. SCkSP is better suited for mapping the type of shapes

incorporated in air pollution plumes than traditional clustering methods. This can have a large impact on the environment and economy.

The simple polygon constraint makes it possible to identify anisotropic or irregular shapes. In addition, it allows for the representation of the complex spatial area to be in a concise format (e.g., polygon chains). This enables valuable information to be clearly derived. The size constraint is important because it allows SCkSP to highlight the most critical areas for the distribution of limited resources. For the example of the allocation of law enforcement resources, the size constraint allows for the spatial coverage to be defined by law enforcement officials. Consequentially, resource allocation is optimized for that given size constraint. The objective of SCkSP is to maximize the total weights covered by the simple polygons. This allows for the optimal utilization of limited resources.

1.2 Our contribution

Our previous work proposed the Nearest Neighbor Triangulation and Merging (NNTM) algorithm to create an SCkSP which will be reviewed in Section 2 [31]. NNTM follows three main steps: 1) construction of the t-nearest graph and a set of triangles; 2) identification of seed triangles based on density; 3) merging a set of dense triangles while obeying the size constraint thereby constructing an SCkSP.

Our previous contribution was as follows:

- We introduced a new spatial covering problem, namely Size Constrained k Simple Polygons (SCkSP).
- We proved that the SCkSP problem is NP-hard.
- We proposed the Nearest Neighbor Triangulation and Merging (NNTM) algorithm that creates an SCkSP.
- We experimentally evaluated our proposed algorithm using a Chicago crime dataset.

The NNTM algorithm used the t-nearest graph for the construction of the building blocks allowing for the creation of an SCkSP [31]. However, we found that the performance of NNTM is highly dependent on the t-nearest graph. We propose a new approach in this paper called the Nearest Neighbor Point and Merging (NNPM) algorithm to remedy this issue. Our new contribution is as follows:

- We propose the Nearest Neighbor Point and Merging (NNPM) algorithm for creating an SCkSP.
- We theoretically evaluate all proposed algorithms using cost models and proofs of algorithmic properties.
- We experimentally evaluate all proposed algorithms using a Chicago crime dataset and a U.S. Federal wildfire dataset.

1.3 Problem definition

In our formulation of the SCkSP problem, a geometric space contains a set of weighted points. The $SCkSP(S, P, k, a)$ problem is defined as follows:
Input: A geometric space S with

- a set of weighted spatial points $p \in P$,
- the number of polygons k, and
- the polygon size constraint a

Output: Size Constrained k Simple Polygons (SCkSP)
Objective:

– Max-sum: Maximize the total weights of the spatial points covered by the k simple polygons.

Constraints:

– The simple polygon meets the size constraint.

1.4 Problem hardness

The NP-hardness of SCkSP follows from the result regarding the NP-hardness of the Grid-Empty Polygonalization (GEP) problem.

Theorem 1 *Pick's theorem* [8, 17]*: Let P be a simple polygon with integer vertices on integer grids. Let n_i be the number of grid points in the interior of P and let n_b be the number of grid points on the boundary of P. Then the size of P is $n_b/2 + n_i - 1$.*

Theorem 2 *The SCkSP problem is NP-hard.*

Proof The NP-hardness of SCkSP can be proved by reduction from the well known NP-complete problem, the Grid-Empty Polygonalization (GEP) problem [16]. Given n grid points in the geometric space, the GEP problem finds the simple polygon that covers n points and has a size of $n/2 - 1$. Let $A = (S, P)$ be an instance of GEP, where S is a geometric space and P is a set of grid points. Let $B(S, P, k, a)$ be an instance of the SCkSP problem, where S is a geometric space, P is a set of weighted grid points, k is the number of simple polygons, and a is the size constraint. Let $k = 1$ and $a = n/2 - 1$. Assume that every point has a unit weight. Then it is easy to show that the instance of GEP is a special case of SCkSP. Since A is constructed from B in polynomial-bounded time, the proof is complete. □

1.5 Related work

Many studies have been conducted in the context of Constraint Polygonization [4, 16, 18, 22, 26]. When regarding the special case of polygons (e.g., convex polygon, star-shaped polygon, square, pentagon, etc.), various Minimum Area Optimization techniques have been employed [1, 12, 20]. However, none of these approaches have focused on maximizing the total point-weights enclosed by a fixed number of polygons. The work most similar to ours is that of the Minimum Simple Polygonalization problem [28, 30, 33]. However, these approaches do not consider the use of weighted points and k polygons as the constraint. Instead of the identification of simple polygons, the minimum covering circle problems identify the k smallest circles that are able to maximize the coverage of the spatial points [2, 11, 19]. Since a circle maximizes the area for a given fixed perimeter length according to the isoperimetric inequality, their techniques are not applicable to our problem [3].

The most important task in SCkSP is how to group a set of points and construct polygon shapes at specific spatial locations in order to maximize the total weights. The SCkSP problem can be solved as a clustering problem as well. Density-based methods may be used to maximize the total weights of each cluster [14, 21, 34]. The general idea is to measure the local density of each point using grid-cells or circles, and group the nearest dense points to

create a set of clusters. These methods can discover dense areas and create irregular-shaped clusters. However, there are no clear concepts regarding geometric optimization constraints (e.g., polygon shapes and size constraint) in these methods. Many variant approaches based on clustering methods have been proposed to enforce different types of optimization constraints. Graph-based clustering methods use the proximity graph to define a similarity measure between two points and partition a set of points into k groups based on graph-cut algorithms (e.g., min-cut) [13, 23, 35]. Network connectivity constraints were incorporated into the clustering methods to identify dense graph-paths or sub-networks [24, 29, 36]. Clustering methods that have to obey size constraints have been investigated [6, 25]. However, there has been no work to our knowledge on the identification of a set of simple polygons that can maximize the total weights of points under a size constraint.

1.6 Basic concepts

In geometry, a line segment is represented by two endpoints (p_1, p_2). Given n points, a polygon chain is a finite ordered sequence of line segments (p_1, p_2), (p_2, p_3), (p_3, p_4), $\ldots (p_{n-1}, p_n)$ joining adjacent pairs in a finite sequence of points $p_1, p_2, \ldots p_n$ in the plane. A polygon chain is closed if it has at least one line segment and its first and last points coincide (i.e., $p_1 = p_n$). A simple polygon is a closed polygonal chain of line segments that do not cross each other. The simple polygon is commonly used to describe the closed region bounded by a simple closed polygon chain. Polygons are basic building blocks in most geometric applications because they are flexible enough for the modeling of arbitrarily complex shapes [32]. In this work, we consider only one important special case: a simple polygon without holes.

A simple polygon can be represented by the orientation of directed line segments [27]. We restrict the polygon's vertex to a spatial point (see Lemma 1).

Lemma 1 *All weighted spatial points should be located on a vertex of the polygons in an SCkSP.*

Proof Assume that a spatial point is located inside the polygon. In this case, we can identify the nearest edge of the polygon from the point and construct a triangle. We can easily see that after removing the triangle the polygon still becomes a simple polygon and satisfies the size constraint. It also potentially includes other spatial points to be used to increase the total weights without violating the polygon size constraint. Therefore, all spatial points should be located on a vertex of the polygon. □

Let $S = p_1, p_2, \ldots, p_n$ denote a set of n spatial points. Every spatial point p_i has its x and y coordinates. Let P be an ordered list of spatial points on the polygon in counter clockwise (CCW) direction. Then the area of the polygon can be computed using the Shoelace algorithm [5]. The union operation of two simple polygons is an operation that finds the simple polygon containing the area inside either of the two simple polygons. In this paper we study k simple polygons that maximize the total weights of the spatial points while obeying the polygon size constraint.

1.7 Outline

The rest of the paper is organized as follows: Section 2 reviews the Nearest Neighbor Triangulation and Merging (NNTM) algorithm for SCkSP. Section 3 describes our proposed

approach. We provide correctness proofs for the proposed approach in Section 4. Section 5 describes the experimental design and presents the experimental observations and results. Section 6 concludes the paper.

2 Preliminary results

In this section, we first review our preliminary approach to the SCkSP problem. We then show a limitation of our preliminary work.

2.1 Nearest neighbor triangulation and merging (NNTM)

In this subsection, we describe the nearest neighbor triangulation and merging (NNTM) approach to the SCkSP problem [31]. NNTM follows three main steps: 1) construction of the t-nearest neighbor graph and a set of triangles; 2) identification of seed triangles based on density; 3) merging a set of dense triangles while obeying the size constraint thereby constructing an SCkSP.

A triangle is one of the basic shapes in geometry: a polygon with three vertices and three edges. It has been proven that a simple polygon can be decomposed into a set of non-overlapping triangles [9]. The core idea in NNTM is to use the t-nearest graph to create a set of triangles as building blocks for an SCkSP (see Lemma 2) while incrementally merging these triangles in order to maximize the total point-weights. This is all performed while ensuring that the size constraint and the simple polygon constraint are obeyed.

Lemma 2 *Given n spatial points, the n-nearest graph provides the building blocks to construct an SCkSP.*

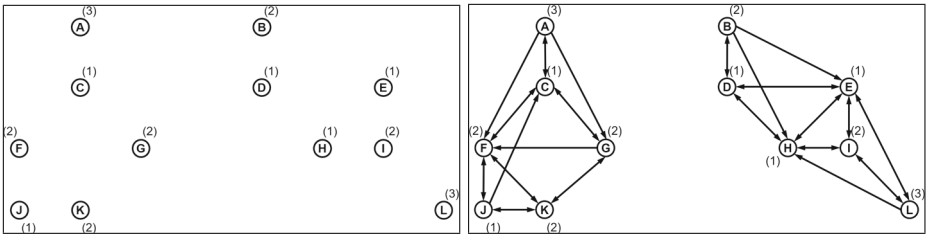
Proof Every polygon has a triangulation [10]. Let n be the number of spatial points. The n-nearest graph is a directed complete graph where each pair of spatial points are connected by a directed edge. Let $\triangle A$ be the set of all triangles produced by Polygon Triangulation and let $\triangle B$ be the set of triangles produced by the n-nearest graph. Since $\triangle A \subset \triangle B$, the n-nearest graph provides the building blocks to construct an SCkSP. □

In the remainder of this paper, we will use the following notation: (1) $\triangle ABC$ represents triangle ABC, (2) $\square ABCD$ represents quadrilateral $ABCD$, and (3) $\pentagon ABCDE$ represents pentagon $ABCDE$.

The Nearest Neighbor Triangulation and Merging (NNTM) algorithm starts with constructing the t-nearest neighbor graph and creates a set of overlapping triangles on the geometric space.

Definition 1 Given a set of points (P), the t-nearest neighbor graph NG_t is a directed graph where two points $a \in P$ and $b \in P$ are connected by a directed edge $\overrightarrow{ab}$ when the distance from a to b is among the t-th smallest distances from a to all other points in P.

Figure 2a shows the input example of SCkSP (reproduced from Fig. 1a). Every spatial point is associated with a weight, as indicated by the number displayed over it. Let the number of polygons be 2 (i.e., $k = 2$) and let the polygon size constraint be 1.5 (i.e.,



(a) Input (b) The 3-nearest neighbor graph (NG_3)

Fig. 2 Example of the t-nearest neighbor graph (NG_t)

$a = 1.5$). Assume that we use 3-nearest neighbors to construct the t-nearest graph (i.e., 3-nearest neighbor graph). Figure 2b shows the creation of the 3-nearest neighbor graph (NG_3) consisting of 12 nodes and 36 directed edges.

NNTM then constructs a set of overlapping triangles based on the t-nearest graph. A triangle is constructed using the following rule: triangle $\triangle ABC$ is defined if node A has two successors (i.e., B and C). Once a set of triangles has been constructed, the NNTM algorithm chooses one of the triangles as a seed and combines the nearest triangles to construct a simple polygon (see Lemma 3).

Lemma 3 *Given a triangle and a simple polygon consisting of n_s vertex points, NNTM merges them and creates a new simple polygon in worst case $O(n_s)$ time.*

Proof A triangle is a simple polygon. A simple polygon has no self-intersections. NNTM can therefore identify the edge on the adjacent side of two polygons in $O(n_s)$. After removing the edge on the adjacent side, it can construct a simple polygon oriented counterclockwise direction in $O(n_s)$. □

One naive approach is to enumerate all possible seed triangles with the goal of combining neighboring triangles that are able to maximize the total weights while at the same time meeting the polygon size constraint. However, the search space becomes too large to be able to examine all available polygons. To remedy this issue, NNTM identifies the densest triangle by finding the local densest area and iteratively increases the size of the polygon to maximize the total weights. The density of a triangle (or polygon) $\triangle t$ can be defined as:

$$density(\triangle t) = \sum_{i \in \triangle t} \frac{w(i)}{size(\triangle t)}, \tag{1}$$

where $w(i)$ is a weight for point i and $size(\triangle t)$ is the size of $\triangle t$.

NNTM incrementally merges local dense triangles to construct a simple polygon. This merging process involves multiple iterations. The key idea in this process is that NNTM maintains a set of seed triangles and constructs one simple polygon for each seed triangle. We refer to this polygon as a candidate for an SCkSP. NNTM constructs a set of candidates and chooses the densest k simple polygons among them in order to maximize the total weights under the size constraint. We define the number of candidates used during the NNTM algorithm as $c \cdot k$, where c is the parameter ratio and k is the number of polygons. We will use it as a stopping criterion for the NNTM algorithm (Lemma 7).

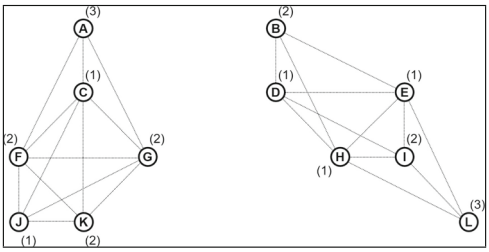
Algorithm 1 Nearest Neighbor Triangulation and Merging (NNTM) algorithm (pseudo-code).

1 **Inputs:**
2 – A geometric space S,
3 – a set of weighted points $p \in P$,
4 – the number of polygons k,
5 – the polygon size constraint a,
6 – the number of nearest neighbors t, and
7 – the parameter for constructing a set of candidates c
8 **Outputs:** A Size Constrained k Simple Polygons ($SCkSP$)
9 **Steps:**
1: Construct the t-nearest neighbor graph (NG_t)
2: Construct a set of overlapping triangles ($\triangle T$)
3: Construct a set of seed triangles ($\triangle S$) from $\triangle T$
4: $PY \leftarrow \emptyset$
5: **while** $\triangle S$ not empty **do**
6: Choose a seed triangle ($\triangle s \in \triangle S$).
7: Start with the seed triangle ($\triangle s$) and merge neighboring triangles ($\triangle t \in \triangle T$) to create polygon py that can maximize the total point-weights under the size constraint.
8: $\triangle S \leftarrow \triangle S \setminus \{\triangle s\}$, $PY \leftarrow PY \cup \{py\}$
9: **end while**
10: Identify k polygons from PY that can maximize the total point-weights of the polygons.
11: **return** $SCkSP$

Algorithm 1 presents the pseudo-code for a generalized version of NNTM. First, NNTM constructs the t-nearest neighbor graph (NG_t) and a set of overlapping triangles ($\triangle T$) (Lines 1–2). It then creates a set of seed triangles ($\triangle S$) from $\triangle T$ (Line 3). The number of seed triangles is defined as $c \cdot k$. It then chooses the densest triangle from a set of seed triangles ($\triangle s \in \triangle S$) (Line 6) and iteratively merges neighboring triangles with the seed triangle to create a simple polygon that is able to maximize the total point-weights under the size constraint (Line 7). NNTM then stores both the seed triangle and simple polygon into the candidate list (Line 8) and repeats the process until no available seed triangle exists (Line 5). Finally, NNTM identifies a set of polygons (i.e., k simple polygons) from the candidate list that can maximize the total weights under the size constraint and returns an SCkSP (Lines 10-11).

Consider a set of overlapping triangles constructed from the 3-nearest neighbor graph (see Fig. 3a). Figure 3b shows a list of triangles sorted by their density. In this example, $\triangle EIL$ is a triangle with the highest density. Let this triangle be a seed triangle in the first iteration (see Fig. 4a). NNTM starts with $\triangle EIL$ and incrementally merges the seed triangle with its neighboring triangles to maximize the density of the polygons. Since $\triangle HLI$ is the neighboring triangle with the highest density, NNTM combines $\triangle EIL$ and $\triangle HLI$ to construct a single simple polygon (i.e., $\square EIHL$), allowing for the density of the weights to be maximized (see Fig. 4b).

Given $\square EIHL$, $\triangle DHI$ can then be merged with $\square EIHL$ to construct a simple polygon $\pentagon EIDHL$ (see Fig. 5a). Since $\pentagon EIDHL$ has reached the maximum size according to the size constraint (i.e., $a = 1.5$), NNTM stores $\pentagon EIDHL$ in the candidate list for an SCkSP



(a) Overlapping Triangles

Triangle	weight	size	density	Triangle	weight	size	density
△EIL	6	0.5	12.0	△FKG	6	1.0	6.0
△AFC	6	0.5	12.0	△CFG	5	1.0	5.0
△ACG	6	0.5	12.0	△CFK	5	1.0	5.0
△HLI	6	0.5	12.0	△CKG	5	1.0	5.0
△FJK	5	0.5	10.0	△BDE	4	1.0	4.0
△GJK	5	0.5	10.0	△DIE	4	1.0	4.0
△BDH	4	0.5	8.0	△CJK	4	1.0	4.0
△EHI	4	0.5	8.0	△EHL	5	1.5	3.333
△CFJ	4	0.5	8.0	△DHE	3	1.0	3.0
△DHI	4	0.5	8.0	△BHE	4	1.5	2.666

(b) Densities of Triangles

Fig. 3 NNTM algorithm: example of seed triangles

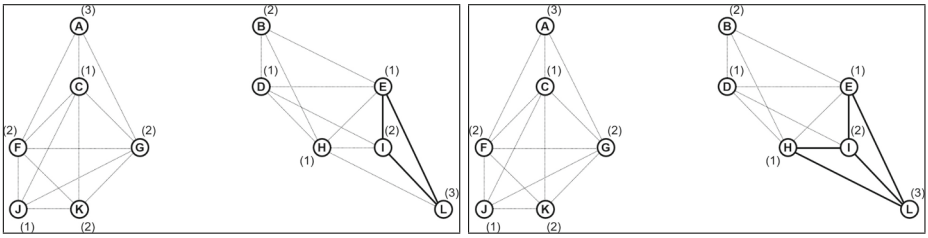
and chooses the next seed triangle to continue the process. In this example, the triangle with the highest density is $\triangle AFC$ (see Fig. 3b). Therefore NNTM sets $\triangle AFC$ to be the next seed triangle (Fig. 5b) and continues this merging process; $\triangle AFC$ can be merged with $\triangle ACG$ and $\triangle CFJ$ to construct $\pentagon AFJCG$ (see Fig. 6a to b). Since $\pentagon AFJCG$ has reached the size constraint, NNTM stores $\pentagon AFJCG$ in the candidate list.

After the construction of two polygons, NNTM identifies the next seed triangle and merges the neighboring triangles to create new candidates. In this example, the other seed triangles cannot produce a better solution, so NNTM creates a solution of SCkSP with two polygons (i.e, $\pentagon EIDHL$ and $\pentagon AFJCG$) that are able to maximize the total weights under the size constraint. As stated earlier, it is time-consuming to examine all of the seed triangles that are able to be used when producing simple polygons. In our approach, we construct $c \cdot k$ candidates and choose the best polygons among them to construct a solution of SCkSP.

2.2 Limitation of NNTM

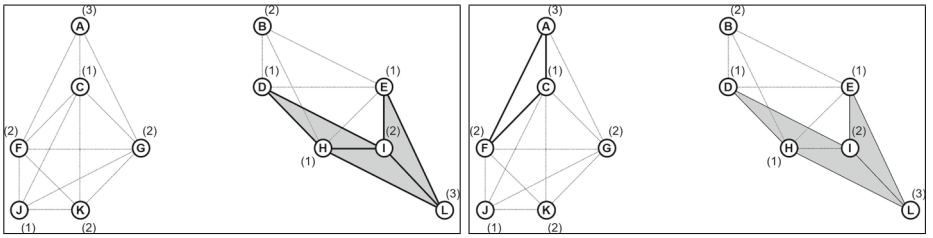
In this subsection, we demonstrate that the bottleneck of NNTM is the construction of the t-nearest neighbor graphs. In our small example, we can see that the output of NNTM (Fig. 6b) cannot produce the best solution for SCkSP as shown in Fig. 1c. This is because 3-nearest neighbors are not sufficient for the construction of the building blocks (i.e., triangles) needed to construct an SCkSP. One solution to handle this issue is to use a higher value of t to create more triangles as building blocks. However, the time complexity for creating the t-nearest graph is $O(n \cdot t^2)$, which cannot handle a large amount of spatial points (Lemma 4).

Lemma 4 *The cost for constructing the t-nearest graph is* $O(n \cdot t^2)$.



(a) 1st iteration: step 1 (b) 1st iteration: step 2

Fig. 4 NNTM algorithm: 1st iteration merging steps 1–2



(a) 1st iteration: step 3 (b) 2nd iteration: step 1

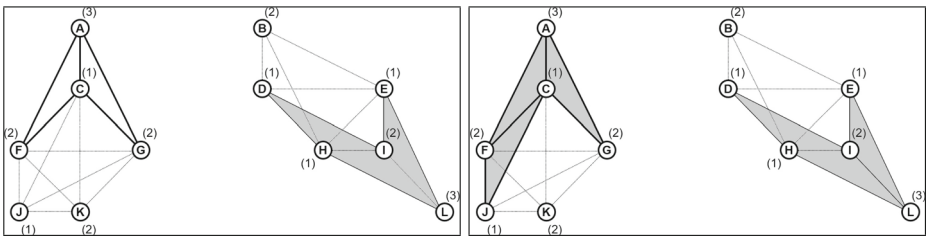
Fig. 5 NNTM algorithm: 1st iteration merging step 3 and 2nd iteration merging step 1

Proof Let n be the number of points. For each point, we choose two nearest points (i.e., successors) among the t nearest spatial points to create the directed edges. Therefore, the time complexity for constructing the t-nearest graph is $O(n \cdot t^2)$. □

In order to understand the bottleneck of NNTM, we used real-world crime datasets regarding 'THEFT' in 2017 that consisted of 35,960 spatial points in Chicago [7]. First, we randomly chose 6,000 spatial points and created 10 test cases. Each spatial point is weighted based on the number of accidents in the same location. We then fixed the size constraint to 3,000 m^2 and varied the number of nearest neighbors (i.e., t). We set c to 100 to construct a set of candidates. We averaged the execution times over 10 test runs.

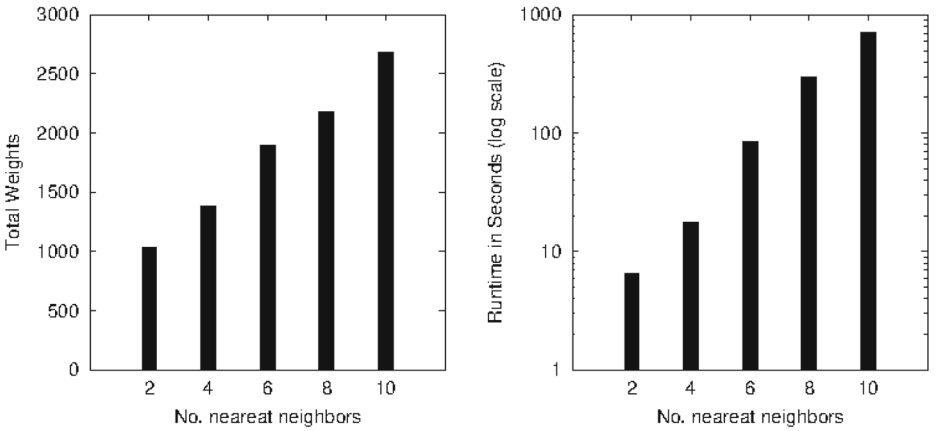
Figure 7 shows the results of the bottleneck analysis. As the number of nearest neighbors increases, the sum of the weights increases (Fig. 7a). As expected, NNTM with a greater number of nearest neighbors significantly improves the solution quality because of the large number of nearest neighbors (i.e., t) which provides better building blocks (i.e., triangles) for the construction of an SCkSP. This consequentially produces a better solution. However, Fig. 7b shows that the construction of the t-nearest graph is the main bottleneck in NNTM in terms of computational cost. As the number of nearest neighbors increases, the run-time significantly increases.

To verify this interpretation, we measured the run-times of two components in NNTM: (1) construction of the t-nearest graph and (2) merging of the triangles. First, we fixed t to 10, the number of spatial points to 6,000, and the size constraint to 6,000 m^2. We incrementally increased the number of polygons from 2 to 8. Figure 8a shows that the construction of the t-neighbor graph is the main bottleneck. The run-time of the merging process increases



(a) 2nd iteration: step 2 (b) 2nd iteration: step 3

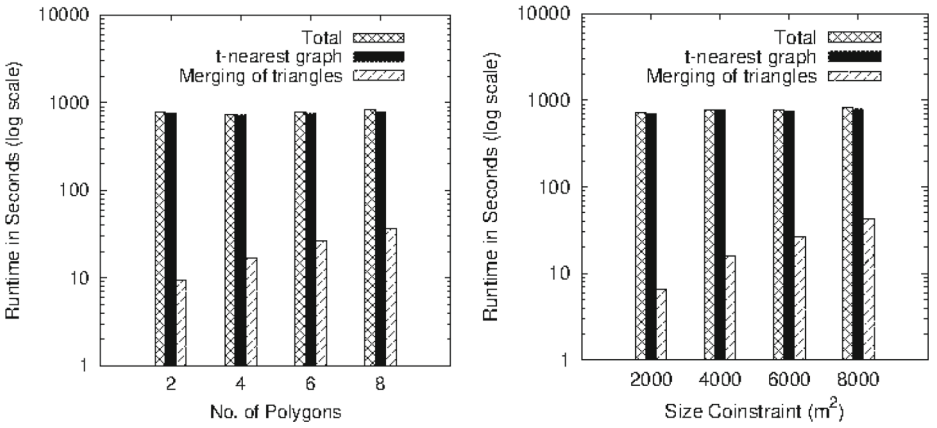
Fig. 6 NNTM algorithm: 2nd iteration merging steps 2–3



(a) Comparison of solution quality (b) Run-time bottleneck analysis

Fig. 7 Effect of the number of nearest neighbors

as the number of polygons increases. However, the increased cost is small and therefore negligible. We then fixed t to 10, the number of spatial points to 6,000, and the number of polygons to 6. We increased the size constraint from 2,000 m^2 to 8,000 m^2. In Fig. 8b, we can see that the main bottleneck of NNTM is the construction of the t-neighbor graph. As the size constraint increases, the run-time of the merging process increases. The computational cost of NNTM slightly increases as the size constraint increases. However, since the cost of the merging process is very small, no significant effect of the size constraint on the performance of NNTM can be observed. Therefore, we conclude that the main bottleneck of NNTM is the construction of the t-nearest neighbor graph.



(a) n=6,000, a=6,000m^2 (b) n=6,000, k=6

Fig. 8 Bottleneck analysis of NNTM (t = 10)

3 Proposed approach

In this section, we introduce a new approach, namely Nearest Neighbor Point and Merging (NNPM), to reduce the computation cost for constructing an SCkSP and explain key elements involved in the designing of the algorithm in detail.

3.1 The nearest neighbor point and merging algorithm

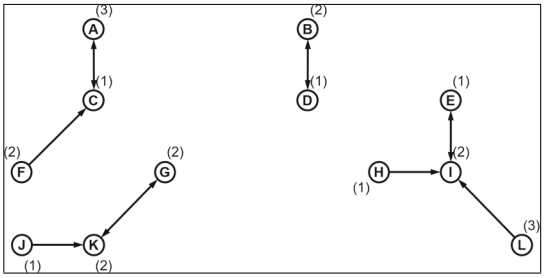
In this subsection, we introduce a novel method in order to reduce the computational cost and produce a better solution for the SCkSP problem. In this work, we represent the polygon as an ordered list of points in counter-clockwise order. We first point out that a solution of NNTM is highly dependent on the t-nearest neighbor graph. To remedy this issue, we propose an incremental algorithm that adds a spatial point one after another to the polygon while maximizing the density of the polygon.

The first core idea in NNPM is to use the nearest neighbor graph for constructing a set of directed seed edges.

Definition 2 Given a set of points (P), the nearest neighbor graph NG is a directed graph where two points $a \in P$ and $b \in P$ are connected by a directed edge $\overrightarrow{ab}$ when the distance from a to b is the smallest of the distances from a to any other point in P.

NNPM starts by constructing the nearest neighbor graph (NG). It then sorts all directed edges by weight in descending order. Consider the input example of SCkSP in Fig. 2a. Figure 9a shows a set of directed edges on the nearest neighbor graph and Fig. 9b shows a list of edges ordered by the total weights of the two end-points. NNPM chooses the densest edges among them and constructs a set of seed edges.

The second core idea in NNPM is to represent the polygon as a directed cycle graph with all of the edges being oriented counterclockwise. The directed cycle graph representation makes it easier to incrementally add new edges and grow the size of the polygon as well as maximize the density of the point-weights. The counterclockwise orientation can produce a positive value to be used in the computing of the area of the simple polygon without the use of the absolute value signs in the Shoelace formula [5]. Since a simple polygon can be decomposed into a set of non-overlapping triangles, the directed cycle graph can be decomposed into a set of directed cycle graphs (Lemma 5). Based on this concept, NNPM combines two directed cycle graphs to construct a new directed cycle graph (i.e., simple polygon).



(a) The directed nearest neighbor graph (NG)

Edge	weight	Edge	weight
LI	5	HI	3
AC	4	IE	3
CA	4	JK	3
BD	3	FC	2
DB	3	GK	2
EI	3	KG	2

(b) Directed seed edges

Fig. 9 Example of the nearest neighbor graph and directed seed edges

Lemma 5 *A directed cycle graph can be decomposed into a set of directed cycle graphs.*

Proof Let a and b be the two nodes that are not adjacent in a directed cycle graph. After adding two edges $\overrightarrow{ab}$ and $\overrightarrow{ba}$, we can decompose the directed cycle graph into two directed cycle graphs. Therefore, a directed cycle graph can be decomposed into a set of directed cycle graphs by repeating this process in a hierarchical fashion. □

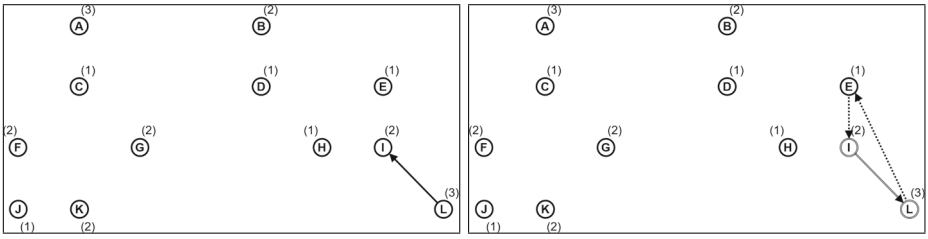
The third core idea in NNPM is to start with the directed seed edge and add spatial points one by one while updating the simple polygon. This process requires three main steps: 1) reversing the direction of one of the edges in the directed cycle graph, 2) scanning all of the spatial points located on the left-hand side of the directed edge, and 3) identifying the spatial point that can maximize the density of the directed cycle (i.e., simple polygon) (see Eq. 1) and creating a new simple polygon using a graph traversal algorithm.

NNPM chooses one of the directed edges from the directed cycle graph and identifies one spatial point that can maximize the density. It takes linear time to examine all points for one directed edge. Since the number of edges increases as the size of the polygon increases, the algorithm may scan all spatial points for each directed edge in the directed cycle graph. However, it is important to note that one directed edge and one spatial point determine a unique triangle. NNPM uses this condition for a unique triangle and materializes the scanned information for the next iteration. Therefore, NNPM scans the spatial points for only two directed edges for each iteration.

Consider the example of the directed edges in Fig. 9b. Edge $\overrightarrow{LI}$ has the largest value in terms of weight. In the first iteration, NNPM starts with setting edge $\overrightarrow{LI}$ as a seed edge. First, NNPM reverses the direction of edge $\overrightarrow{LI}$, which becomes edge $\overrightarrow{IL}$ (see Fig. 10a). Second, it scans all spatial points located on the left-hand side of edge $\overrightarrow{IL}$. Third, NNPM identifies the spatial point that can maximize the density of the spatial points on the polygon. Since point E is only one candidate in this example, NNPM chooses point E to create a new directed cycle graph (i.e., simple polygon). Finally, it uses graph traversal search (e.g., depth-first search) to construct $\triangle LEI$ (see Fig. 10b).

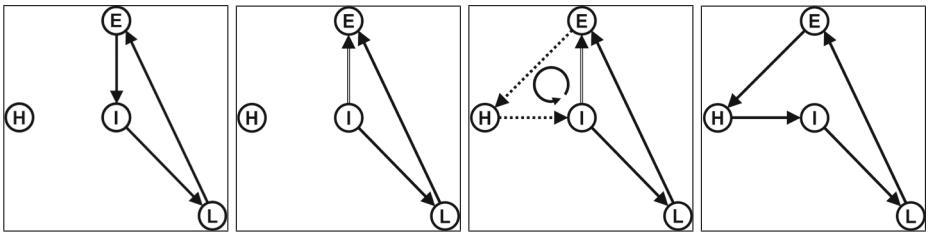
$\triangle LEI$ consists of three directed edges (i.e., $\overrightarrow{LE}$, $\overrightarrow{EI}$, and $\overrightarrow{IL}$) (see Fig.11a). NNPM reverses one of these edges and identifies the spatial point that can maximize the density of the polygon. In this example, NNPM reverses edge $\overrightarrow{EI}$ which becomes edge $\overrightarrow{IE}$ (see Fig. 11b), and it then chooses point H located on the left-hand side of edge $\overrightarrow{IE}$ in order to maximize the density of the polygon (see Fig. 11c). NNPM uses the counter-clockwise graph traversal search to construct $\square EHIL$ (see Figs. 11d and 12a).

After the construction of $\square EHIL$, NNPM reverses edge $\overrightarrow{HI}$ and selects point K in order to construct $\pentagon ILEHK$ (see Fig. 12b). Since $\pentagon ILEHK$ reaches the size constraint,



(a) 1st iteration: step 1 (b) 1st iteration: step 2

Fig. 10 NNPM algorithm: 1st iteration merging steps 1–2



(a) Polygon $\triangle LEI$ (b) Edge Reversal (c) Graph Traversal Search (d) Polygon $\square EHIL$

Fig. 11 Merging spatial point H with $\triangle LEI$

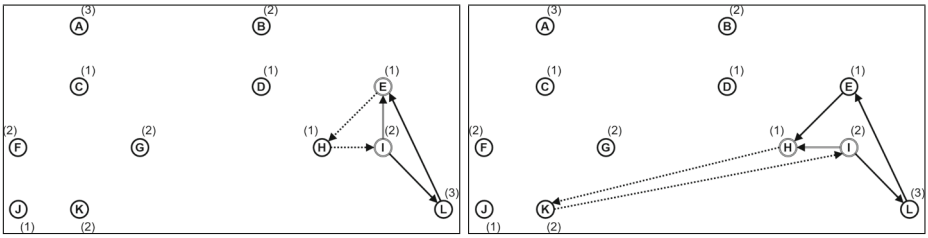
NNPM stores this polygon in the candidate list for an SCkSP and repeats this process using a new seed edge.

For the second iteration, edge $\overrightarrow{AC}$ has the largest weight (see Fig. 9b). NNPM sets $\overrightarrow{AC}$ as a seed edge and reverses the direction of the edge (i.e., $\overrightarrow{CA}$). It then merges the spatial point F located on the left-hand side of edge $\overrightarrow{CA}$ and creates $\triangle AFC$ (see Fig. 13b).

NNPM then reverses edge $\overrightarrow{CA}$ and merges point G located on the left-hand side of edge $\overrightarrow{AC}$ to construct $\square CGAF$ (see Fig. 14a). Finally, NNPM reverses edge $\overrightarrow{FC}$ and merges point J located on the left-hand side of edge $\overrightarrow{CF}$ to construct $\pentagon FJCGA$ (see Fig. 14b). This completes the second iteration. After two iterations, NNPM cannot produce a better solution than the two polygons. Therefore, it creates a solution of SCkSP with two polygons (i.e., $\pentagon ILEHK$ and $\pentagon FJCGA$). Since the number of directed seed edges is large, NNPM constructs $c \cdot k$ candidates and chooses the best polygons among them to construct an SCkSP.

Algorithm 2 Nearest Neighbor Point and Merging (NNPM) algorithm (pseudo-code).

Inputs:
– A geometric space S,
– a set of weighted points $p \in P$,
– the number of polygons k,
– the polygon size constraint a, and
– the parameter for constructing a set of candidates c
Outputs: A Size Constrained k Simple Polygons ($SCkSP$)
Steps:
1: Construct the nearest neighbor graph (NG)
2: Construct a set of seed edges ($\overrightarrow{S}$)
3: $PY \leftarrow \emptyset$
4: **while** S not empty **do**
5: Choose a seed edge ($\overrightarrow{s} \in \overrightarrow{S}$).
6: Start with the seed edge and merge neighboring points ($p \in P$) to create a polygon py that is able to maximize the total point-weights under the size constraint.
7: $\overrightarrow{S} \leftarrow \overrightarrow{S} \setminus \{\overrightarrow{s}\}$, $PY \leftarrow PY \cup \{py\}$
8: **end while**
9: Identify k polygons from PY that can maximize the total point-weights.
10: **return** $SCkSP$



(a) 1st iteration: step 3 (b) 1st iteration: step 4

Fig. 12 NNPM algorithm: 1st iteration merging steps 3–4

Algorithm 2 presents the pseudo-code for the NNPM algorithm. First, NNPM constructs the nearest neighbor graph (Line 1). It then constructs a set of seed edges and sorts them by weight (Line 2). After that, it chooses the densest seed edge and merges spatial points in order to maximize the total point-weights in the polygon (Line 5–6). If the size of the polygon reaches the size constraint, it stores the simple polygon in the candidate list (Line 7) and continues this process (Line 4). The number of seed edges is defined as $c \cdot k$. Therefore, after $c \cdot k$ iterations, NNPM selects the most weighted polygons from the candidate list and returns an SCkSP (Line 9–10).

4 Analysis on the quality of the proposed approaches

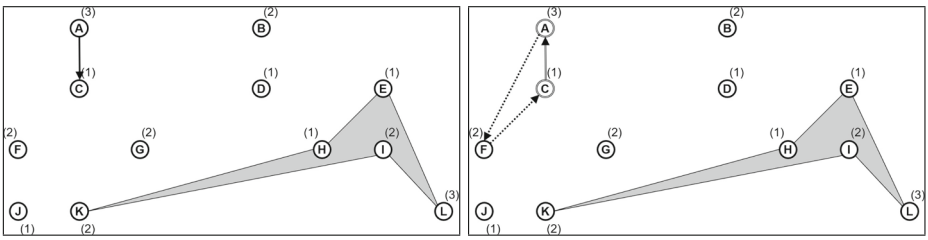
In this section, we prove that the proposed approaches create an SCkSP.

4.1 Analysis of the proposed approaches

Lemma 6 *NNPM algorithm is correct and creates an SCkSP that meets the size constraint and the simple polygon constraint.*

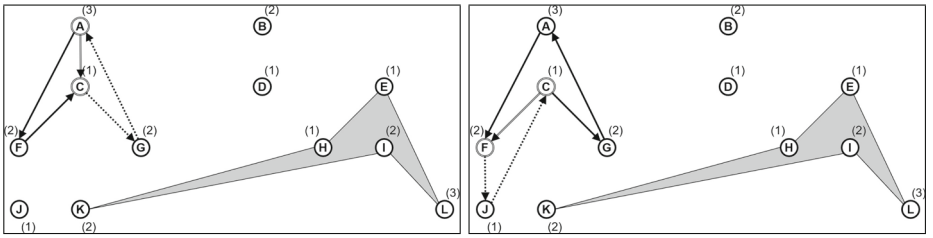
Proof Given a directed cyclic graph, NNPM selects one point located on the outside of the directed cyclic graph and creates a new one. Since NNPM does not create a self-intersecting polygon, it obeys the simple polygon constraint during the addition of each spatial point. It also does not violate the size constraint for creating a polygon. □

Lemma 7 *Both NNPM and NNTM algorithms terminate after $c \cdot k$ iterations.*



(a) 2nd iteration: step 1 (b) 2nd iteration: step 2

Fig. 13 NNPM algorithm: 2nd iteration merging steps 1–2



(a) 2nd iteration: step 3 (b) 2nd iteration: step 4

Fig. 14 NNPM algorithm: 2nd iteration merging steps 3–4

Proof Both NNPM and NNTM create $c \cdot k$ candidates. Each iteration constructs one simple polygon; therefore the algorithms terminate after $c \cdot k$ iterations. □

4.2 Algebraic cost model for the proposed approaches

We developed a cost model for the proposed approaches for SCkSP. Let n be the number of weighted points, let k be the number of polygons, let t be the number of nearest neighbors, and let i be the number of iterations to terminate the algorithm. Since the proposed algorithm is output-sensitive, we assume that the number of points in SCkSP is n_s. The size constraint can be approximately defined by n_s. The proposed approaches compute the distance between two spatial points at a cost of $O(n^2)$.

4.2.1 NNTM

NNTM starts with constructing the t-nearest neighbor graph and creates a set of overlapping triangles. It takes $O(n \cdot t^2)$ (Lemma 4). At each iteration of NNTM, the algorithm selects a seed triangle and incrementally merges the neighboring triangles.

The merging process involves multiple steps. Assume that the number of points in SCkSP is bounded by $O(n_s)$. Then the number of merging steps is bounded by $O(n_s)$. At each merging step, NNTM identifies a neighboring triangle that maximizes the density of the polygon. The number of neighboring triangles is bounded by $O(t \cdot n_s)$ (see Lemma 8). NNTM sorts neighboring triangles according to the density of the polygon. This takes $O(t \cdot n_s \cdot \log(t \cdot n_s))$. NNTM also tests if the output polygon is simple. This takes $O(t \cdot n_s^2)$. The creation of the new polygon takes $O(n_s)$. Therefore each merging step takes $O(t \cdot n_s \cdot (\log t + n_s))$. Since the number of merging steps is bounded by $O(n_s)$, the merging process to create one candidate takes $O(t \cdot n_s^2 \cdot (\log t + n_s))$. Since the number of iterations is bounded by $O(i)$, the complexity of NNTM is $O(n \cdot t^2 + i \cdot t \cdot n_s^2 \cdot (\log t + n_s))$. Assume that $n >> n_s$. Then we can see that the construction of the t-nearest graph is the main bottleneck of NNTM.

Lemma 8 *Given a simple polygon consisting of n_s vertex points and a set of triangles ($\triangle T$) based on the t-nearest neighbor graph, the number of neighboring triangles of the simple polygon is bounded by $O(t \cdot n_s)$.*

Proof Each point in the polygon is connected to t other points. Each edge in the polygon has at most t neighboring triangles. Since the polygon consists of n_s edges, the number of neighboring triangles is bounded by $O(t \cdot n_s)$. □

4.2.2 NNPM

NNPM starts with constructing the nearest neighbor graph and creates a set of directed seed edges. It takes $O(n)$. At each iteration of NNPM, the algorithm selects a directed seed edge and incrementally merges the neighboring spatial points.

The merging process involves multiple steps. Assume that the number of points in SCkSP is bounded by $O(n_s)$. Then the number of merging steps is $O(n_s)$. At each merging step, NNPM reverses one of the edges in the directed cyclic graph and finds the spatial point located on the left-hand side of the edge. The main goal of the merging process is to find a directed cyclic graph that can maximize the density. To achieve this goal, NNPM sorts all spatial points located on the left side of the directed edge according to the density of the polygon. This takes $O(n \cdot \log n)$. NNPM also tests if the polygon is simple at a cost of $O(n \cdot n_s)$. The creation of a new polygon takes $O(n_s)$ using a counter-clockwise graph traversal search. Therefore each merging step takes $O(n \cdot (\log n + n_s))$. Since the number of merging steps is bounded by $O(n_s)$, the merging process to create one candidate takes $O(n_s \cdot n \cdot (\log n + n_s))$. The number of iterations is bounded by $O(i)$; therefore the complexity of NNPM is $O(i \cdot n_s \cdot n \cdot (\log n + n_s))$.

5 Experimental evaluation

In this section, we present the experimental design and an analysis of the experimental results.

5.1 Experiment layout

Figure 15 shows our experimental setup. The overall goal of the experiments is to show the performance improvements for creating an SCkSP that can be achieved by the NNPM algorithm. The metric for comparison in our experiments was the run-time of the algorithm and the total weights covered by an SCkSP. We wanted to answer four questions: (1) What is the effect of the number of spatial points? (2) What is the effect of the number of polygons? (3) What is the effect of the size constraint? and (4) What is the effect of the number of candidates? We used two real-world datasets: 1) a Chicago crime dataset consisting of 35,960

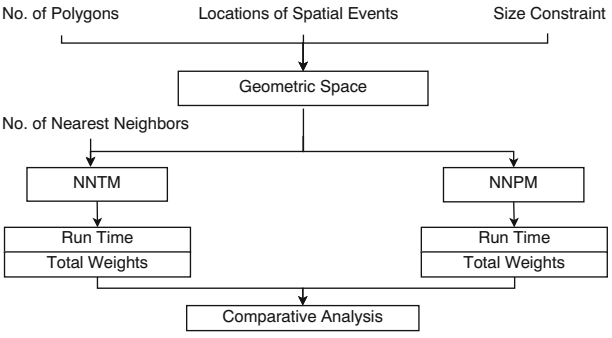


Fig. 15 Experiment layout

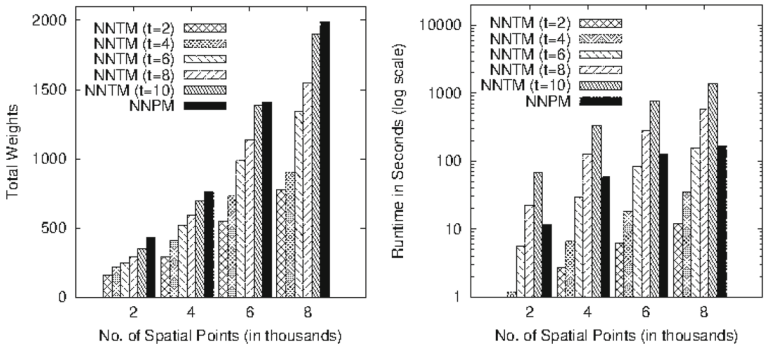
spatial points and 2) a U.S. Federal wildfire dataset consisting of 266,887 spatial points [7, 15]. Each spatial point is weighted by the number of accidents in the same location. The algorithms were implemented in Java 1.8 with a 32 GB memory run-time environment. All experiments were performed on an Intel(R) Core(TM) i5-6500 CPU @ 3.20GHz machine running Ubuntu 16.04.6 LTS with 32GB of RAM.

5.2 Experiment results and analysis

We experimentally evaluated two different approaches (i.e., NNTM and NNPM) by comparing the impact on the performance of 1) the number of weighted nodes (i.e., n), 2) the number of polygons (i.e., p), 3) the size constraint (i.e., a), and 4) the number of nearest neighbors (i.e., t). Since the performance of NNTM is dependent on the t-nearest graph, we varied t from 2 to 10. We set c to 100 to construct a set of candidates. Performance measurements were execution time and the total weights covered by the polygons. The input areas were randomly chosen from the two datasets. Execution times and total weights were averaged over 20 test runs for each input area.

5.2.1 Effect of the number of spatial points

The aim of the first set of experiments was to demonstrate the performance improvements when creating an SCkSP that can be obtained by the NNPM algorithm. We fixed the number of polygons to 6 and the size constraint to 6,000 m^2. We incrementally increased the number of spatial points from 2,000 to 8,000. Figure 16a shows that NNPM outperforms the NNTM approaches in terms of total weights covered by the simple polygons. As the number of spatial points increases, the sum of the weights increases. This is because the larger area provides a better chance to identify polygons with greater weights. We can also see that as the value of t increases, the sum of the weights created by NNPM increases. Figure 16b shows that NNPM significantly outperforms the NNTM approaches in terms of run-time when the value of t is greater than 6. Note that the NNTM approach with smaller t shows poor solution quality. The results of the experiments show that NNPM performs much better than NNTM.



(a) Comparison of the Sum of the Weights

(b) Run-time Comparison

Fig. 16 Effect of the number of spatial points ($k = 6, a = 6,000$ m^2, c = 100)

5.2.2 Effect of the number of polygons

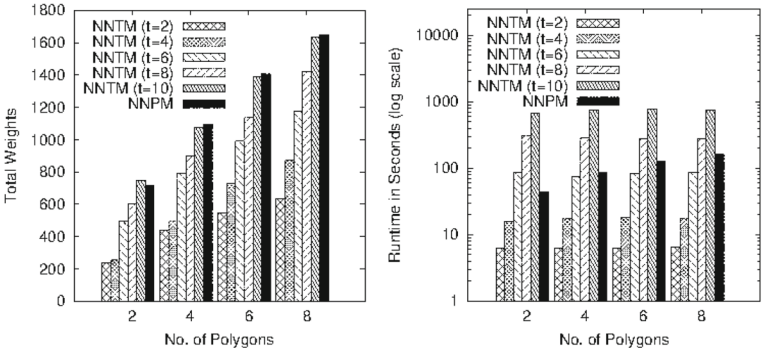
The second experiment evaluated the effect of the number of polygons. We fixed the number of spatial points to 6,000 and the size constraint to 6,000 m^2. We incrementally increased the number of polygons from 2 to 8. As shown in Fig. 17a, we observed that NNPM outperforms the NNTM approaches in terms of total weights covered. As the number of polygons increases, the sum of the weights increases. Figure 17b shows that NNPM performs faster than the NNTM approaches when the value of t is greater than 6. The run-time of NNTM does not significantly increase as the number of polygons increases. This is because the main bottleneck of NNTM is the construction of the t-nearest graph. The run-time of NNPM slightly increases as the number of polygons increases. The results of the experiments show that NNPM produces a better solution and is faster than NNTM when the value of t is high.

5.2.3 Effect of the size constraint

The third experiment evaluated the effect of the size constraint. We fixed the number of spatial points to 6,000 and the number of polygons to 6. We increased the size constraint from 2,000 m^2 to 8,000 m^2. Figure 18a shows that NNPM exhibits better solution quality than NNTM. As the size constraint increases, the sum of the weights increases. Figure 18b shows that NNPM performs faster than the NNTM approaches when the value of t is greater than 6. As can be observed here, the size constraint does not significantly affect the performance of NNTM because the main bottleneck of NNTM is to construct the t-nearest graph. The run-time of NNPM slightly increases as the size of the polygon increases. It is shown that NNPM outperforms NNTM in terms of computational cost and solution quality.

5.2.4 Effect of the number of candidates

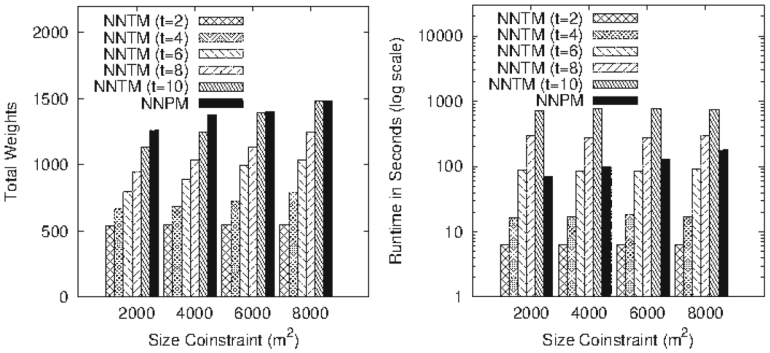
The fourth experiment evaluated the effect of the number of candidates. We fixed the number of spatial points to 6,000, the number of polygons to 6, and the size constraint to 6,000 m^2. We varied the number of candidates from 25 to 200. Figure 19a shows that NNPM produces a better solution than NNTM. As the number of candidates increases, the sum of the weights increases. This is because both NNPM and NNTM can choose polygons with



(a) Comparison of the Sum of the Weights

(b) Run-time Comparison

Fig. 17 Effect of the number of polygons ($n = 6,000$, $a = 6,000m^2$, $c = 100$)



(a) Comparison of the Sum of the Weights

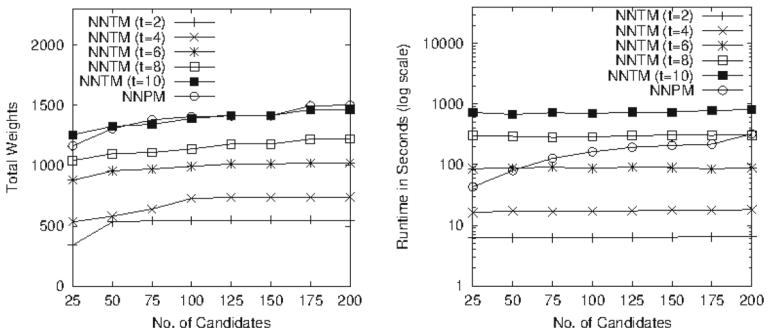
(b) Run-time Comparison

Fig. 18 Effect of the size constraint ($n = 6,000$, $k = 6$, $c = 100$)

greater weight as the number of candidates increases. Figure 19b shows that NNPM is faster than NNTM when the value of k is greater than 6.

5.2.5 Comparison with other potential solutions

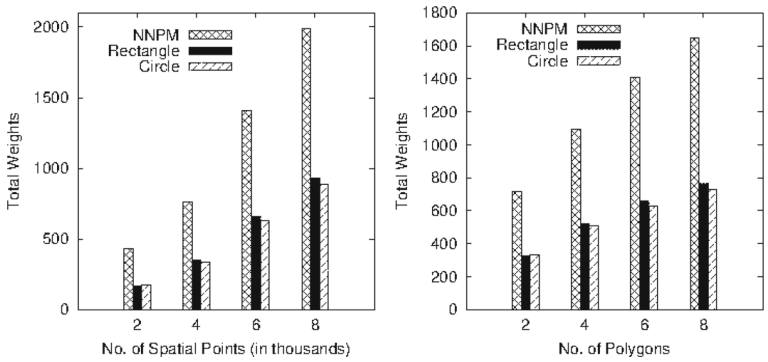
The fifth experiment compared the solution quality of NNPM with other potential solutions. The general idea idea that is found in the related work is the identification of dense areas based on grid-cells or circles. A circle is not a polygon, but it can be approximately represented by a simple convex polygon with multiple line segments. We used a rectangle or a circle to extract the top k dense areas and create simple polygons. First, we fixed the number of polygons to 6 and the size constraint to 6,000 m^2. We incrementally increased the number of spatial points from 2,000 to 8,000. Figure 20a shows that NNPM outperforms other potential solutions. This is because NNPM creates non-convex polygons to maximize the sum of the weights. As the number of spatial points increases, so does the performance gap. Next, we fixed the number of spatial points to 6,000 and the size constraint to 6,000 m^2. We



(a) Comparison of the Sum of the Weights

(b) Run-time comparison

Fig. 19 Effect of the number of candidates ($n = 6,000$, $k = 6$, $a = 6,000$ m^2)

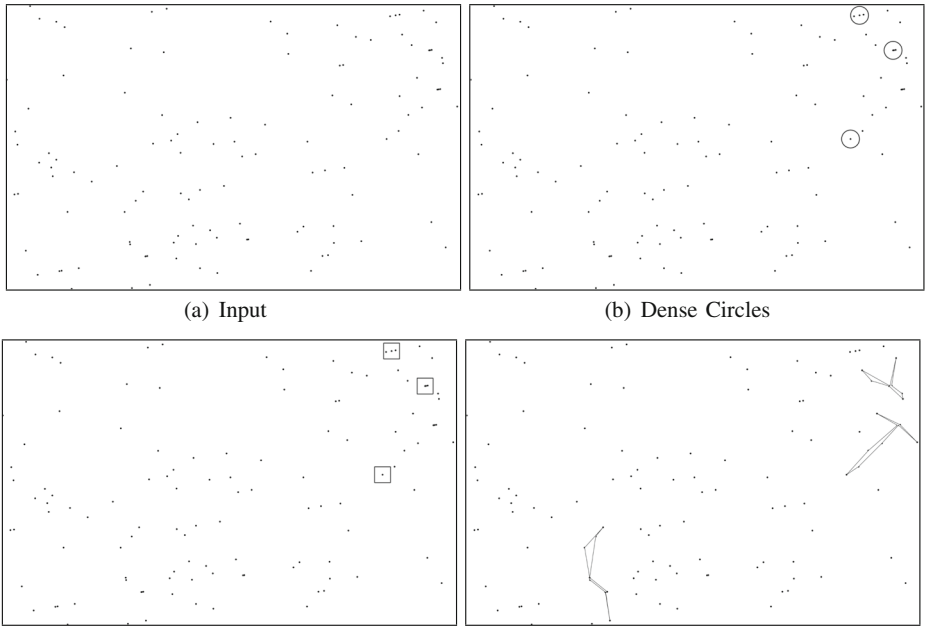


(a) $k = 6$, $a = 6,000\,m^2$ (b) $n = 6,000$, $a = 6,000\,m^2$

Fig. 20 Comparisons of the sum of the weights with other solutions

incrementally increased the number of polygons from 2 to 8. Figure 20b shows that the solution quality of NNPM is better than other solutions. As the number of polygons increases, the performance gap also increases.

We also conducted a qualitative evaluation of NNPM comparing its output with the output of other solutions. We used a motor vehicle theft dataset from Chicago in 2017 [7]. The study area is defined by a rectangle where latitude ranges between -87.66 and -87.6336 degrees and longitude ranges between 41.885 and 41.9 degrees. The total number of spatial



(a) Input (b) Dense Circles

(c) Dense Rectangles (d) NNPM

Fig. 21 Case study using a motor vehicle theft dataset from Chicago in 2017 ($n = 157, k = 3, a = 5,000\ m^2$)

points is 157 and each spatial point is weighted by the number of accidents in the same location (see Fig. 21a). We fixed the number of polygons to 3 and the size constraint to 5,000 m^2. Figure 21b and c show the output of the dense circles and the dense rectangles. The number of accidents covered by circles (or rectangles) is 12. Figure 21d shows the output of NNPM. The number of accidents covered by NNPM is 51. Assume that we have limited resources (e.g., sensors, cameras, policemen, etc.) that are able to be allocated for the prevention of criminal activities. Given the size (or area) constraint, NNPM can optimize the usage of these limited resources within the most critical areas.

5.3 Discussion

NNPM achieved a significant computational performance gain over our previous approach. This improvement was obtained by eliminating the construction of the t-nearest graph. The key component of NNPM is to use a directed cyclic graph to represent a simple polygon and incrementally add spatial points for the construction of a new directed cycle graph. This novel approach can maximize the total weights of the simple polygon without constructing the t-nearest graph. The experimental results show that NNPM produces a better solution and significantly reduces the computational cost required when creating an SCkSP.

6 Conclusion and future work

We presented the problem of creating Size Constrained k Simple Polygons (SCkSP). Important potential applications of SCkSP include hotspot detection and resource allocation for car accidents, crime incidents, and air pollution dispersion plume tracking. The problem is computationally challenging because of the large number of spatial points and the size constraint. In this paper, we proposed our novel NNPM algorithm for identifying k simple polygons that can maximize the total weights covered by the polygons while obeying the polygon size constraint. Our experimental results demonstrated a significant reduction in computational cost illustrating why the NNPM algorithm is a better solution for creating an SCkSP.

In the future, we would like to explore the NNPM algorithm on big data processing platforms which are able to handle much larger datasets. We plan to identify independent components of both NNPM and NNTM and develop a parallel algorithm. We will also study the side length constraint for the simple polygon. The problem can be related to the well-known Traveling Salesman Problem, which is NP-hard. Due to the isoperimetric inequality [3], we may need to investigate special polygon substructures to maximize the total weights. Additionally, we would like to explore simple polygons with holes (or weakly simple polygons) to be able to identify the hotspot areas for resource allocation.

Acknowledgements We would like to thank the National Science Foundation CAREER under Grant No. 1844565.

References

1. Arkin EM, Chiang YJ, Held M, Mitchell JSB, Sacristan V, Skiena S, Yang TC (1998) On minimum-area hulls. Algorithmica 21(1):119–136

2. Bereg S, Daescu O, Zivanic M, Rozario T (2015) Smallest maximum-weight circle for weighted points in the plane. In: International conference on computational science and its applications. Springer, pp 244–253
3. Blåsjö V (2005) The isoperimetric problem. Am Math Mon 112(6):526–566
4. Boyce JE, Dobkin DP, Drysdale RLS III, Guibas LJ (1982) Finding extremal polygons. In: Proceedings of the fourteenth annual ACM symposium on theory of computing. ACM, pp 282–289
5. Braden B (1986) The surveyor's area formula. Coll Math J 17(4):326–337
6. Bradley P, Bennett K, Demiriz A (2000) Constrained k-means clustering. Microsoft research, Redmond, pp 1–8
7. City of Chicago Data Potal (2019) Crimes—2001 to present, https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2. Retrieved Feb. 2019
8. Coxeter HSM (1989) Introduction to geometry. John Wiley & Sons
9. De Berg M, Cheong O, Van Kreveld M, Overmars M (2008) Computational geometry: introduction. In: Computational geometry: algorithms and applications, pp 1–17
10. Devadoss SL, O'Rourke J (2011) Discrete and computational geometry. Princeton University Press, Princeton
11. Elzinga DJ, Hearn DW (1972) The minimum covering sphere problem. Manag Sci 19(1):96–104
12. Eppstein D, Overmars M, Rote G, Woeginger G (1992) Finding minimum areak-gons. Discrete Comput Geom 7(1):45–58
13. Ertoz L, Steinbach M, Kumar V (2002) A new shared nearest neighbor clustering algorithm and its applications. In: Workshop on clustering high dimensional data and its applications at 2nd SIAM international conference on data mining, pp 105–115
14. Ester M, Kriegel HP, Sander J, Xu X et al (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: Kdd, vol 96, pp 226–231
15. Federal Wildland Fire Occurrence Data, https://wildfire.cr.usgs.gov/firehistory/data.html/, Retrieved Feb. 2019
16. Fekete SP (2000) On simple polygonalizations with optimal area. Discrete Comput Geom 23(1):73–110
17. Fekete SP, Pulleyblank WR (1993) Area optimization of simple polygons. In: Proceedings of the ninth annual symposium on computational geometry, pp 173–182
18. Fulek R, Keszegh B, Morić F, Uljarević I (2013) On polygons excluding point sets. Graphs Comb 29(6):1741–1753
19. Hearn DW, Vijay J (1982) Efficient algorithms for the (weighted) minimum circle problem. Oper Res 30(4):777–795
20. Hêche JF, Liebling TM (1997) Finding minimum area simple pentagons. Oper Res Lett 21(5):229–233
21. Hinneburg A, Gabriel HH (2007) Denclue 2.0: fast clustering based on kernel density estimation. In: International symposium on intelligent data analysis. Springer, pp 70–80
22. Jiang M (2012) On covering points with minimum turns. In: Frontiers in algorithmics and algorithmic aspects in information and management. Springer, pp 58–69
23. Karypis G, Han EH, Kumar V (1999) Chameleon: hierarchical clustering using dynamic modeling. Computer 32(8):68–75
24. Li X, Han J, Lee JG, Gonzalez H (2007) Traffic density-based discovery of hot routes in road networks. In: International symposium on spatial and temporal databases. Springer, pp 441–459
25. Malinen MI, Fränti P (2014) Balanced k-means for clustering. In: Joint IAPR international workshops on statistical techniques in pattern recognition (SPR) and structural and syntactic pattern recognition (SSPR). Springer, pp 32–41
26. Mitchell JS, Polishchuk V (2008) Minimum-perimeter enclosures. Inf Process Lett 107(3-4):120–124
27. Munkres JR (2000) Topology. Prentice Hall, Upper Saddle River
28. Muravitskiy V, Tereshchenko V (2011) Generating a simple polygonalizations. In: 2011 15th international conference on information visualisation (IV). IEEE, pp 502–506
29. Oliver D, Shekhar S, Kang JM, Laubscher R, Carlan V, Bannur A (2014) A k-main routes approach to spatial network activity summarization. IEEE Trans Knowl Data Eng 26(6):1464–1478
30. Peethambaran J, Parakkat AD, Muthuganapathy R (2016) An empirical study on randomized optimal area polygonization of planar point sets. J Exp Algorithmics (JEA) 21:1–10
31. Reich A, Ohriniuc R, Yang K (2018) Size constrained k simple polygons. In: Proceedings of the 26th ACM SIGSPATIAL international conference on advances in geographic information systems. ACM, pp 500–503
32. Samet H (2006) Foundations of multidimensional and metric data structures. Morgan Kaufmann
33. Taranilla MT, Gagliardi EO, Hernández Peñalver G (2011) Approaching minimum area polygonization. In: XVII Congreso Argentino de Ciencias de la Computación

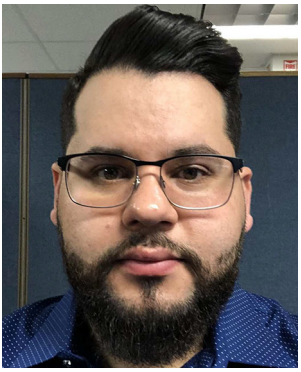
34. Wang W, Yang J, Muntz R et al (1997) Sting: a statistical information grid approach to spatial data mining. In: VLDB, vol 97, pp 186–195
35. Xu X, Yuruk N, Feng Z, Schweiger TA (2007) Scan: a structural clustering algorithm for networks. In: Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 824–833
36. Yang K (2016) Distance-constrained k spatial sub-networks: a summary of results International conference on geographic information science. Springer, pp 68–84

KwangSoo Yang is an assistant professor in the Department of Computer and Electrical Engineering and Computer Science at Florida Atlantic University. He received the BS degree in electrical engineering from Yonsei University in 1998, the master's degree in computer science from the University of Minnesota in 2010, and the PhD degree in computer science from the University of Minnesota in 2015. He was a software engineer for LG CNS in Seoul, Korea, from 2001 to 2008. He has received an NSF CAREER Award in 2019. His research interests include spatio-temporal network databases, spatio-temporal networks, and evacuation routing problems.

Kwang Woo Nam is a professor in the School of Computer, Information, and Communications Engineering at Kunsan National University in the Republic of Korea. He received the PhD degree in computer science from Chungbuk National University. After PhD, he studied location-based services and telematics in Electronics and Telecommunications Research Institute of Korea. His research interests includes spatial and moving objects database, spatial big data, and GeoAI.

Ahmad Qutbuddin received his BS degree in Computer Engineering from King Fahd University of Petroleum and Minerals in 2010. He worked as a Lecturer for Umm Al-Qura University in Makkah, Saudi Arabia from 2010 to 2014. He received a master's degree in Computer Engineering from the University of Central Florida in 2017. He is currently working toward his PhD degree in Computer Engineering at Florida Atlantic University. His research interests are spatial network databases and Network Voronoi Diagram.

Aaron Reich received a Bachelor of Science in Computer Science from Florida Atlantic University in 2017. He is currently working towards the Master of Science in Computer Science with a Specialization in Machine Learning at the Georgia Institute of Technology. He works as a Software Engineer for Pionetechs. His research interests include spatio-temporal networks, evacuation routing problems, data mining, and machine learning.



Valmer Huhn is currently working toward a dual BS degree in Computer Science and Computer Engineering from Florida Atlantic University. Upon completion of his BS degrees he will pursue a MS in Computer Science from Florida Atlantic University in 2020. His research and development interests include spatial networks and data mining.