Theoretical Computer Science 556 (2014) 34–44

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Following a curve with the discrete Fréchet distance

Tim Wylie [a,*], Binhai Zhu [b]

[a] Department of Computer Science, The University of Alberta, Edmonton, AB T6G-2E8, Canada
[b] Department of Computer Science, Montana State University, Bozeman, MT 59717-3880, USA

ARTICLE INFO

Article history:
Received 28 February 2014
Received in revised form 7 June 2014
Accepted 18 June 2014
Available online 25 June 2014

Keywords:
Fréchet distance
Set-chain matching
Curve/point-set matching
Polygonal curve
Curve similarity

ABSTRACT

Finding the similarity between curves is an important problem that comes up in many areas such as 3D modeling, GIS applications, ordering, and reachability. A related problem is to find one of the curves given a measure of similarity and another curve. Given a set of points S, a polygonal curve P, and an $\varepsilon > 0$, the discrete set-chain matching problem is to find another polygonal curve Q such that the nodes of Q are points in S and $d_F(P, Q) \leq \varepsilon$. Here, d_F is the discrete Fréchet distance between the two polygonal curves. For the first time we study the set-chain matching problem based on the discrete Fréchet distance rather than the continuous Fréchet distance. We further extend the problem based on unique or non-unique nodes and on limiting the number of points used. We prove that three of the variations of the set-chain matching problem are **NP**-complete. For the version of the problem that is polynomial, we give an $\mathcal{O}(|P||S|)$ time greedy solution.

1. Introduction

Matching geometric objects and finding curves through designated points are common problems in many areas of research such as pattern matching, computer vision, map routing, protein structure alignment, ordering, etc. Some of these path problems are fundamental, and are used to define complexity classes and completeness. A problem closely related to our study here is map matching where the goal is to find a path through an embedded graph that minimizes the distance from a given polygonal curve [1]. This has several useful applications, as mentioned by Alt et al., such as determining the path of a vehicle on a road network (graph) given noisy approximate GPS data (polygonal curve). For global map matching, the distance measure generally used is the Fréchet distance.

The Fréchet distance was originally defined by Maurice Fréchet in 1906 as a measure of similarity between two parametric curves [2]. In the early 1990s, the Fréchet distance between polygonal curves was studied by Alt and Godau [3] who presented efficient algorithms and time bounds of $\mathcal{O}(mn \log mn)$, where m, n are the number of vertices in the polygonal curves. Following in 1994, Eiter and Mannila [4] defined the discrete Fréchet distance as an approximate solution to the Fréchet distance based on polygonal curves where only the nodes are taken into consideration.

With the continuous Fréchet distance, the time complexity of map matching on a complete graph was further improved upon in [5] where a new problem was introduced, which we will call set-chain matching (it was unnamed in this work). Given a polygonal curve P, a set of points S, and a maximum distance $\varepsilon > 0$, the problem is to find another polygonal curve, Q, through the set of points such that the Fréchet distance between the new curve and the original is within an allowed distance, $d_{\mathcal{F}}(P, Q) \leq \varepsilon$. They further demonstrated some modifications that can be made to their algorithm for

* Corresponding author.
E-mail addresses: twylie@ualberta.ca (T. Wylie), bhz@cs.montana.edu (B. Zhu).

http://dx.doi.org/10.1016/j.tcs.2014.06.026
0304-3975/

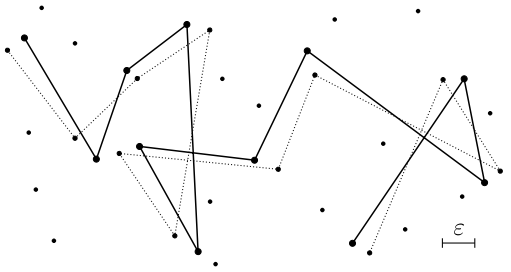


Fig. 1. An instance of the set-chain matching problem in 2D with one solution for a curve of size 11.

various other questions, but also leave open some important future work. We build upon their main problem by extending it and analyzing the problems for discrete cases based on the discrete Fréchet distance.

In this work, we investigate many variations of set-chain matching. We look at the complexity of set-chain matching based on the discrete Fréchet distance, and although the original definition allowed points in the set to be reused in the chain, we now consider both unique and non-unique points. We show that the unique point versions are **NP**-complete, and the non-unique point versions are **NP**-complete when restricting the size of the set of points used, but polynomial when limiting the size of the curve (number of nodes). Fig. 1 shows a simple instance of the set-chain matching problem, which is formally defined at the beginning of Section 3.

The variations of discrete set-chain matching have many applications. Suppose we have intermittent lossy GPS vehicle data where we cannot guarantee the path of the vehicle between our data points. We can find the shortest (and arguably the most plausible) path of the vehicle based on the discrete Fréchet distance. Set-chain matching only solves this problem for the complete graph instance, and requires a different approach for true map matching on a road network (modeled by an embedded planar graph rather than a set of points). The more applicable problem here is if the points in our set represent signal towers (cellular, radio, etc.), which generally have a spherical range. This input allows us to consider several coverage problems. Assuming we know the path of a vehicle (the polygonal curve), what is the minimum number of towers needed to ensure that the signal is not lost? Knowing whether the path is fully covered is important. Finally, being able to determine the best ordering of towers to use during the vehicle's trip may be a necessity. These types of problems are studied in many areas related to wireless sensor networks, graphics, scheduling, and ordering.

This paper is an extension of a previous conference publication [6] and short abstract [7]. Here, we have expanded our discussion and explanations considerably. This includes many new and updated illustrations, pseudocode for a greedy solution, and additional work related to USM. We first provide some background and related work in Section 2. We then cover the definitions and variations of the discrete set-chain matching problem in Section 3. Sections 4, 5, and 6 follow with the actual results of the problems. Finally, we conclude in Section 7 and give some future work related to this research.

2. Background

With respect to map matching, the problem of finding a path in a graph given a polygonal line was first posed by Alt et al. [1] as follows: Let $G = (V, E)$ be an undirected connected planar graph with a given straight-line embedding in $\mathbb{R}^2$ and a polygonal line P, find a path π in G which minimizes the Fréchet distance between P and π. They give an efficient algorithm which runs in $\mathcal{O}(pq \log q)$ time and $\mathcal{O}(pq)$ space where p is the number of line segments of P and q is the complexity of G, but it also allowed vertices and edges to be visited multiple times.

The recent work by Maheshwari et al. improved the running time for the case of a complete graph [5]. The original algorithm decides the map matching problem in $\mathcal{O}(pk^2 \log k)$ where k is the number of vertices in the graph, and the new algorithm solves it in $\mathcal{O}(pk^2)$. Although they do not specify the name for the problem, we refer to it as set-chain matching to avoid confusion with other matching problems. Formally, the set-chain matching problem is defined as: Given a point set S and a polygonal curve P in $\mathbb{R}^d$ ($d \geq 2$), find a polygonal curve Q with its vertices chosen from S, which has a minimum Fréchet distance to P. They decide this problem in $\mathcal{O}(pk^2)$, and also give an algorithm to find the minimal Fréchet distance in $\mathcal{O}(pk^2 \log pk)$.

We originally noted the complexity of discrete set-chain matching with unique nodes, without the actual proof, in [7]. We not only prove it here, but we also show that the continuous version of the problem with unique points is **NP**-complete. This paper is a continuation of our earlier work [6,7,20], but the result based on the continuous Fréchet distance was also independently proven by Accisano and Üngör [8]. They refer to the problem as Curve/Point-Set Matching (CPSM). Shahbaz et al. also proved it was **NP**-complete for non-unique points when trying to visit all nodes under the Fréchet distance [9,10].

The discrete Fréchet distance was originally defined by Eiter and Mannila [4] in 1994, and was further expanded on theoretically by Mosig et al. in 2005 [11]. Given two polygonal curves, we define the discrete Fréchet distance in Definition 1. We use $d(a, b)$ to represent the Euclidean distance between two points a and b, but it could be replaced with other distance measures depending on the application. Since the discrete Fréchet distance looks at all discrete monotonic parameterizations over the nodes of the two polygonal curves, each curve must be reparameterized from the number of monotonic combinations $(m+n)$ to the number of nodes in its curve (m or n). The function takes the minimum, among all reparameterizations

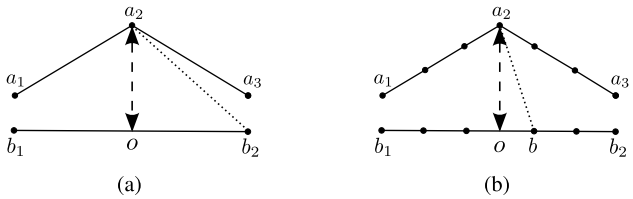


Fig. 2. The relationship between the discrete and continuous Fréchet distance where o is the continuous and the dotted line between nodes is the discrete. (a) shows a case where the curves have fewer nodes and a larger discrete Fréchet distance, while (b) is the same basic curve with more nodes, and thus provides a better approximation of the Fréchet distance.

of the curves (σ and β) from the maximum distance between the nodes $\sigma(s)$ of f and $\beta(s)$ of g where s is the parameter in $[1:m+n]$.

Definition 1. The discrete Fréchet distance d_F between two polygonal curves $f:[0,m]\to\mathbb{R}^k$ and $g:[0,n]\to\mathbb{R}^k$ is defined as:

$$d_F(f,g)=\min_{\sigma:[1:m+n]\to[0:m],\beta:[1:m+n]\to[0:n]}\max_{s\in[1:m+n]}\{d(f(\sigma(s)),\ g(\beta(s)))\}$$

where σ and β range over all discrete non-decreasing onto mappings of the form $\sigma:[1:m+n]\to[0:m]$, $\beta:[1:m+n]\to[0:n]$.

The continuous Fréchet distance is typically explained as the relationship between a person and a dog connected by a leash walking along the two curves and trying to keep the leash as short as possible. However, for the discrete case, we only consider the nodes of these curves, and thus the man and dog must "hop" along the nodes of the curves.

Since the moves taken along the chains are discrete, finding the best walk between the two chains is relatively straightforward. By giving a dynamic programming solution for finding the discrete Fréchet distance between two polygonal curves, in [4] Eiter and Mannila proved that it could be easily solved in $\mathcal{O}(mn)$ time. Recently, Agarwal et al. provided the first subquadratic algorithm for the discrete Fréchet distance giving the following theorem.

Theorem 1. *The discrete Fréchet distance between two polygonal curves, with m and n vertices respectively, can be computed in* $\mathcal{O}(\frac{mn\log\log n}{\log n})$ *time [12].*

Fig. 2 shows the relationship between the discrete and continuous Fréchet distances. In Fig. 2(a), we have two polygonal curves (or chains) $\langle a_1,a_2,a_3\rangle$ and $\langle b_1,b_2\rangle$, the continuous Fréchet distance between the two is the distance from a_2 to segment $\overline{b_1b_2}$, i.e., $d(a_2,o)$. The discrete Fréchet distance is $d(a_2,b_2)$. The discrete Fréchet distance can be much larger than the continuous distance. On the other hand, with enough sample points on the two curves, the resulting discrete Fréchet distance, i.e., $d(a_2,b)$ in Fig. 2(b), closely approximates $d(a_2,o)$.

3. Discrete set-chain matching

We begin with the formal definitions of the problem and the variations as well as some terminology. It is important to note that, as in the continuous version, we make no requirements that P or Q be planar. For discussion, we will refer to the number of nodes in a polygonal curve as the "size" of the curve and it will be denoted as $|A|$ for a polygonal curve A.

Definition 2 *(The discrete set-chain matching problem).*
Instance: Given a point set S, a polygonal curve P in $\mathbb{R}^d$ $(d\geq 2)$, an integer $K\in\mathbb{Z}^+$, and an $\varepsilon>0$.
Problem: Does there exist a polygonal curve Q with vertices chosen from S' where $S'\subseteq S$, such that $T\leq K$ and $d_F(P,Q)\leq\varepsilon$?

T is defined in two ways. When limiting the number of nodes in the curve, $T=|Q|$, and if restricting the number of points used then $T=|S'|$. Fig. 3 shows an example demonstrating the difference between minimizing $|Q|$ or $|S'|$. Here, minimizing $|Q|$ will always yield $|Q|=3$ regardless of the points chosen. However, minimizing $|S'|$ will return $|S'|=2$ and $|Q|=3$, which is the only set of points that is minimal.

We look at three variations of discrete set-chain matching. They vary whether there is a uniqueness constraint on $s\in S$ being used as a node in Q (if points may be used more than once), and whether our goal is to limit the size of the curve Q or the set S'. We distinguish the problems as Unique/Non-unique(U/N) Set-Chain(S) Matching(M) with Subset/Curve(S/C) of size K. When looking at unique nodes, limiting $|Q|$ is equivalent to limiting the set of points used, $|S'|$, since they can only be used once, so we do not separate the cases. The variants are as follows:

- NSMC: Non-unique Set-Chain Matching with $T = |Q|$ where we want to find a curve Q with up to K nodes. This is covered is Section 4.
- NSMS: Non-unique Set-Chain Matching with $T = |S'|$ where we want to find a curve Q that uses up to K points from S. This is covered is Section 5.
- USM: Unique Set-chain Matching where we want to find a curve with up to K nodes, but the points from S can only be used once. This is covered is Section 6.

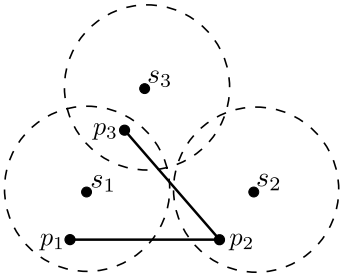


Fig. 3. The difference between minimizing $|Q|$ and $|S'|$. Minimizing $|S'|$ gives $Q = \langle s_1, s_2, s_1 \rangle$ where $|S'| = 2$ and $|Q| = 3$, but minimizing $|Q|$ will yield $|Q| = 3$ whether it uses the sequence $\langle s_1, s_2, s_1 \rangle$ or $\langle s_1, s_2, s_3 \rangle$.

4. Set-chain matching with $T = |Q|$ (NSMC)

The original set-chain matching work dealt with the continuous version of NSMC. The discrete version is decidable by first solving the optimization version with a straightforward greedy solution. We overview the greedy implementation and the complexity of NSMC, and then give the pseudocode.

Fig. 3 demonstrates that we must find at least one point $s_i \in S$ for every $p_j \in P$. First find the $s \in S$ that covers the longest prefix of P, i.e., the longest consecutive subchain starting with p_1. Then add s to the optimal set and remove all the nodes of the prefix of P that it covers. Since the nodes in Q do not have to be unique, we keep s in our set of possible new nodes. Then repeat using the remaining subset of S and subchain of P. Algorithm 1 gives the pseudocode for a straightforward implementation of the greedy method. This solution only uses $\mathcal{O}(|S|)$ space, and has a running time of $\mathcal{O}(|P||S|)$.

Theorem 2. *The discrete non-unique set-chain matching problem where $T = |Q|$ is polynomial, i.e., NSMC $\in$ **P**.*

Proof. Since we can solve the optimization version of NSMC, given P, S, and K, we can find an optimal K', and decide NSMC by comparing whether $K \leq K'$. □

Algorithm 1 solves the optimization version of NSMC with a greedy approach. As mentioned, we want to cover the largest prefix possible at each step. Thus, 'NumCovered' is an array that stores the current prefix length that each point covers. 'PCovered' keeps track of how many nodes have already been covered in P. Thus, if 'PCovered' has the value 5, we

Algorithm 1 GREEDY-NSMC → Return the minimal size of $|Q|$ using a greedy approach.

Input: P is the polygonal curve, S is the set of points, and ε is our distance threshold.
Output: The minimum $|Q|$ for a curve through the points in S such that $d_F(P, Q) \leq \varepsilon$.

```
1:  function GREEDY-NSMC(P, S, ε)
2:      NumCovered ← |S|                          ▷ The prefix covered by each s ∈ S
3:      PCovered ← 0                              ▷ The number of nodes of P already covered
4:      Total ← 0                                 ▷ The number of nodes of Q
5:      while PCovered < |P| do                   ▷ If there are nodes left to cover
6:          for i ← 1, |S| do                     ▷ Loop over S
7:              NumCovered[i] ← 0
8:              j ← PCovered +1
9:              while d(s_i, p_j) ≤ ε do          ▷ Count the number covered by s_i
10:                 NumCovered[i] ← NumCovered[i] + 1
11:                 j ← j + 1
12:         if max(NumCovered) > 0 then           ▷ If the prefix covered is positive
13:             PCovered ← PCovered + max(NumCovered)   ▷ Get the biggest prefix
14:             Total ← Total +1
15:         else                                  ▷ If the current p_j is not covered by any s ∈ S
16:             PCovered ← |P|
17:             Total ← 0                         ▷ Return 0 since there is no feasible solution
18:     return Total
```

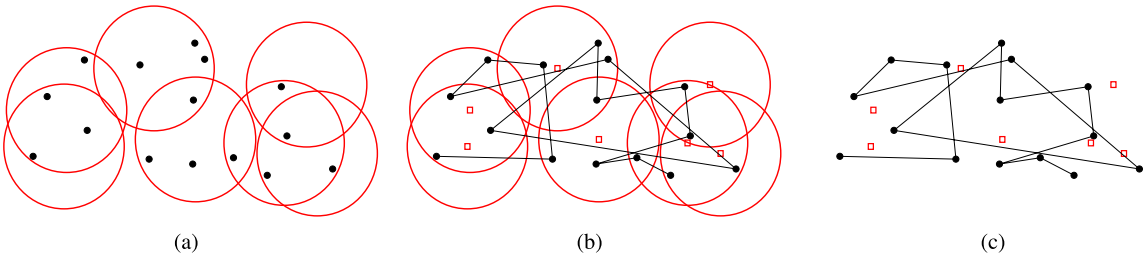


Fig. 4. An example of the steps in the reduction from the discrete unit disk cover problem. (a) An instance of DUDC. (b) Note the centers of the disks and connect the points as a polygonal line. (c) The final instance of NSMS.

are only concerned with finding a prefix beginning at p_6. Finally, 'Total' tracks how many points from S that have been used, which will become our solution length when finished.

This problem can also be solved less efficiently with dynamic programming, and the recurrence relation is shown in Eq. (1). It assumes a 2D array, M, of size $|S| \times |P|$ where the columns represent the nodes in the polygonal curve P and the rows represent points in the set S. The initial condition assumes a column zero populated with 0's in every row. The recurrence can then be processed column by column until finished. The final optimal value will be $Opt = \min_{k=1}^{|S|}(M[k, |P|])$. Using the recurrence naïvely solves the optimization problem in $\mathcal{O}(|P||S|^2)$ time. However, the minimum value at the previous node of P can be saved, the algorithm time can be reduced to $\mathcal{O}(|P||S|)$ time.

$$M[i, j] = \min \begin{cases} M[i, j-1], & \text{if } d(s_i, p_j) \leq \varepsilon, M[i, j-1] \neq 0 \\ \min_{k=1}^{|S|}(M[k, j-1]) + 1, & \text{if } d(s_i, p_j) \leq \varepsilon, M[i, j-1] = 0 \\ 0, & \text{if } d(s_i, p_j) > \varepsilon \end{cases} \tag{1}$$

5. Set-chain matching with $T = |S'|$ (NSMS)

The discrete non-unique set-chain matching problem where we limit the number of points from S used as nodes in Q turns the problem into a coverage issue. This variation of the discrete set-chain matching problem is related to the discrete unit disk cover (DUDC) problem when limiting the number of points from S used. The DUDC problem is known to be **NP**-Hard, and is also difficult to approximate with the most recent results being an 18-approximation algorithm [13], a 15-approximation algorithm [14], and a $(9 + \varepsilon)$-approximation algorithm [15]. Nearly all of the constant factor approximations have been within the last decade. The problem does admit a PTAS [16], but this is infeasible for most instances of the problem with more than two disks. DUDC does not admit a Fully Polynomial Time Approximation Scheme (FPTAS) unless $\mathbf{P} = \mathbf{NP}$.

An instance of the DUDC problem is a set of points in the plane and a set of unit disks that are also in the plane. The problem is to find the minimum number of disks that cover all of the points. Note that the disks and the points are stationary and given as input. Fig. 4(a) shows a simple instance of DUDC with seven unit disks to cover fifteen points. The optimal solution for this example is four of the disks.

Fig. 4 shows a simple example walk-through of how the reduction is constructed. The problems are similar enough that the reduction is fairly straightforward. Basically, we can treat the centers of the unit disks as the points in our set and turn all the points from DUDC into the polygonal chain for NSMS. In Fig. 4(b), the centers of the disks have been marked and we have connected all the points as a polygonal chain in a random order. Fig. 4(c) shows the final instance of NSMS that was constructed.

Theorem 3. *The discrete non-unique set-chain matching (NSMS) problem where $T = |S'|$ is **NP**-complete.*

Proof. This can be shown via a straightforward reduction from the discrete unit disk cover (DUDC) problem which is **NP**-Hard [13]. Formally, we are given a set of points P and a set of disks $D = \{D_1, D_2, ..., D_N\}$ with centers $C = \{c_1, c_2, ..., c_N\}$ with all disks of radius r.

Now, let P' be a polygonal curve made of all points in P in any order. Let $S = C$ and $\varepsilon = r$. Now, $\exists$ a minimum-cardinality subset $D' \subseteq D$ with centers C' such that $\forall p \in P$, $\exists$ a $D_i \in D'$ that contains p if and only if $\exists$ a polygonal curve Q where the vertices are from points in $S' \subseteq S$ such that $|S'| = |D'|$ and $d_F(P', Q) \leq \varepsilon$.

We first prove the forward direction. Given an instance $I \subseteq D$ that is a minimum covering for all points in P. We construct P' by connecting all points in P in any order. Making a polygonal curve Q with the set of centers (C_I) of I is straightforward. We construct Q by finding the disk (D_i) that covers $p_1 \in P'$, and we set $q_1 = c_i$ where c_i is the center of disk D_i. Similarly, we walk through each $p_i \in P'$ and set the center of the disk $D_j \in I$ covering point p_i as $q_i = c_j$. Every ordered node in P' is now still within ε of a node in Q, thus $d_F(P', Q) \leq \varepsilon$, and the set of nodes used, $|S'|$, is equal to $|I|$.

In the other direction, if we have a polygonal curve $Q = \{q_1, q_2, \ldots, q_N\}$ such that the number of unique locations used for vertices is of minimum cardinality and $d_F(P', Q) \leq \varepsilon$. Suppose the set of unique locations S' that Q is made of is

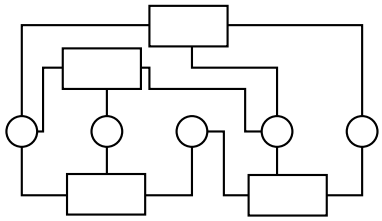


Fig. 5. A simple planar 3-SAT example where clauses are represented by rectangles and variables by circles. Here, there are four clauses and five variables.

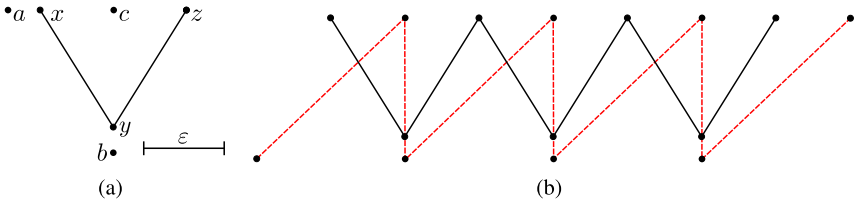


Fig. 6. (a) A choice gadget. (b) A chain with a **false** connection.

not a minimal disk cover of all the vertices of P' viewed as points in a set P. This implies there exists at least one q_i that is unnecessary for a covering by C, and there is a point p_j that can be covered by another c_k. Let C' be this smaller covering. Using the same construction as above we can build a P'' and Q'. This would mean $|C'| < |S'|$ which contradicts our assumption that S' in minimal. Thus, every node $p_i \in P'$ is within ε of at least one node $q_j \in Q$, and S' is a minimum cover.

Finally, we show the problem is in **NP**. Given an instance I we can check whether $d_F(P, I) \leq \varepsilon$ in polynomial time via Theorem 1. □

Since NSMS is so closely related to DUDC, any of the approximations for DUDC can be adapted. The basic greedy heuristic will only give an $\mathcal{O}(\log n)$ approximation. This method selects the point $s \in S$ that covers the most nodes of P, and then removes those nodes since they are covered. The method is repeated until all nodes are covered and the set of points chosen from S can be ordered in any way to be the polygonal curve Q.

6. Unique set-chain matching (USM)

We now address unique set-chain matching where any point from the set can be used at most once, and show that this problem is **NP**-complete via a reduction from planar 3-SAT [17]. Planar 3-SAT is any 3-SAT formula that can be drawn as a planar graph with vertices representing clauses and variables. This is a convenient form of 3-SAT for geometric reductions since a crossover gadget is unnecessary [21,22]. Planar 3-SAT is still **NP**-complete even when all variables are aligned along a single line and all variables and clauses can be contained within a given rectilinear embedding [18]. The input is a planar graph with vertex labels to indicate which vertices are clauses and which are variables. Thus, for our reduction we must handle the standard 3-SAT inputs (the vertices), and the additional elements of a graph, i.e., the edges which imply inclusion.

Fig. 5 shows a simple planar 3-SAT example with five variables and four clauses. The clauses are represented by rectangles and the variables by circles. Note that the edge between a variable and a clause defines inclusion, but does not immediately show whether the variable or its complement is included. This can be represented by various methods such as multiple vertices or "splitting" the vertex in half.

By standard convention, we first introduce several planar "gadgets" that we then arrange in our reduction. We will build up the gadgets in a piecewise manner, and then show how they are connected to form a single polygonal curve. Due to the length of this section, we cover the gadgets and then formally do the reduction with the assumption of their correctness.

Let φ be the 3-SAT formula represented by the input instance of planar 3-SAT with N variables and M clauses. Given an $\varepsilon > 0$, we construct a point set S and a polygonal curve P and let $K = |S_\varepsilon| = |S|$ requiring all points to be used. Here, $S_\varepsilon = \{s \in S \mid p \in P \text{ and } d(p, s) \leq \varepsilon\}$ and referred to as the set of reachable points. We show that φ is satisfiable if and only if with our construction there exists a polygonal curve Q with unique nodes from the set S such that $d_F(P, Q) \leq \varepsilon$, i.e. $|Q| = |S| \leq K$.

6.1. Choices and chains

We first look at the main building block for our gadgets in this reduction, which is the choice gadget shown in Fig. 6(a). There are two ways for a new curve to be constructed starting at a and using the points $\{a, b, c\}$ in order to "cover" the nodes of the curve $\langle x, y, z\rangle$. We label the curve $\langle a, b, c\rangle$ as **true**, and the curve $\langle a, c, b\rangle$ as **false**. This is because the second curve violates our ε constraint since $d(b, z) > \varepsilon$.

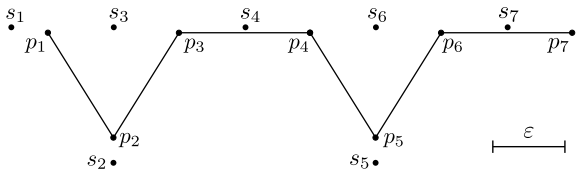


Fig. 7. The base of the variable gadget.

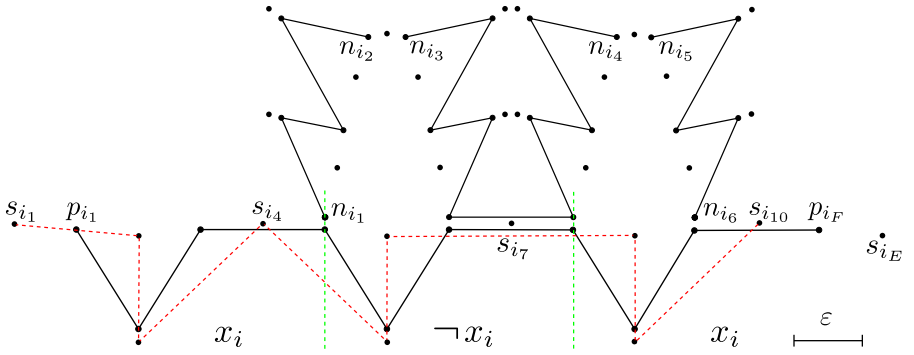


Fig. 8. Variable gadgets linked together for variable x_i where x_i is set to **false** (s_{i_4} is used) and thus $\neg x_i$ is **true** (s_{i_7} is free).

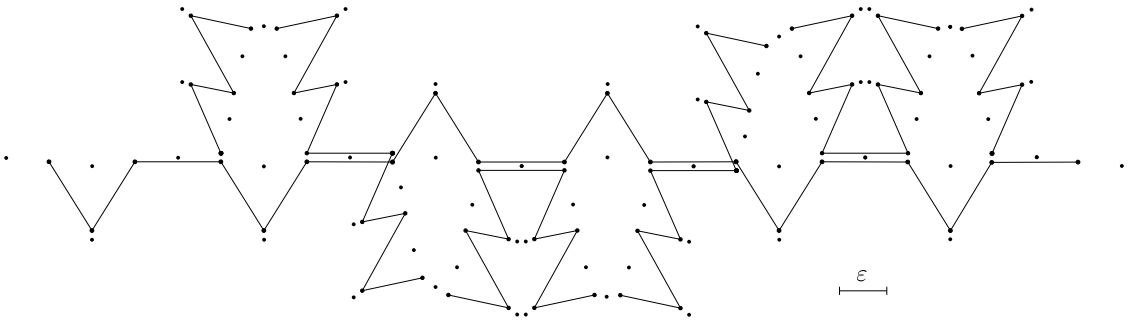


Fig. 9. A variable gadget showing how a chain for a clause gadget can attach on the bottom of the variable gadget. The variable and switch gadgets can "flip" any number of times as needed.

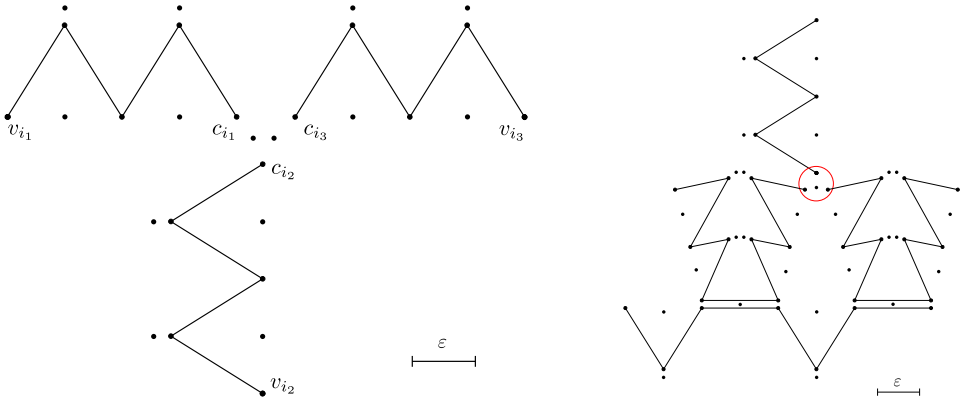
Choice gadgets are linked together to make a chain. Chain gadgets are important because they force a new curve to stay in a **true** or **false** orientation, and therefore transfer information. An example of a chain with a **false** curve is shown in Fig. 6(b).

6.2. The variable gadget

The base of the variable gadget is shown in Fig. 7. A **true** setting begins the new chain as $\langle s_1, s_2, s_3, s_6\rangle$ while a **false** setting begins $\langle s_1, s_3, s_2, s_4, s_5\rangle$. The different settings change whether s_4 is needed to keep $d_F(P, Q) \leq \varepsilon$. A **true** setting does not need the extra node while the **false** does. This free node is what is propagated to the clause gadget. Fig. 8 shows the full variable gadget. As is standard in many reductions, each variable is repeated some finite length while alternating between x and $\neg x$ based on what is needed in the equation.

Unfortunately, the variable gadget alone will not ensure that the new curve alternates between **true** and **false** configurations, which we need for a variable and its complement. Therefore, the variable gadget has a "switch" component, which makes the free point necessary at every other variable gadget, and thus alternates Q between **true** and **false** paths. It is important to note that these switch segments will not be connected to the variable gadgets within ε. Note in Fig. 8 that the first and last instance of the variable gadget do not have the full switch component.

For our planar 3-SAT instance, there may be edges which need to connect from the top and the bottom of the variable gadget. The gadget must be able to handle any planar edges. Looking at Fig. 8, imagine everything is rotated in the gadget from s_{i_7} to s_{i_E} around that vector. This flips the variable and half of the switch component without changing the reduction, which allow attaching chains onto the other side of the variable gadget. The following switch component would also have to be below and then flip back up. Fig. 9 shows this rotation and the flip back to the standard layout. This can be done as often as needed for the connections to the variable.



(a) Clause Gadget (b) Variable Connection

Fig. 10. (a) The clause gadget. (b) The connection point between a variable gadget and a chain to the clause gadget.

6.3. The clause gadget

A clause gadget is straightforward. As shown in Fig. 10(a), three chains meet within ε of each other $(c_{i_1}, c_{i_2}, c_{i_3})$, and there are only two points between them. Each chain is connected at the other end $(v_{i_1}, v_{i_2}, v_{i_3})$ to variable gadgets. The **true** or **false** setting from the variable is propagated up to the clause gadget and at least one of the chains must have the new curve in a **true** position. Only two of the chains can have a **false** setting or else one of the end nodes (C_{k_i}) in the clause gadget will not be within ε of any available point, which is equivalent to the clause being **false** in 3-SAT. Also note that in the clause gadget, if either point is not needed, they can be used by a **true** chain so that all points are used.

The chains from the clause gadgets are attached to the variable gadgets in the highlighted area of Fig. 10(b). There is one point between the ends of the three chains. A segment is added from the clause endpoint v_{k_y} (for clause c_k where $1 \leq y \leq 3$) to the opposite side of the switch component of the variable (or complement) desired, e.g., if x_1 is the third variable in the clause c_k and the connection point is $n_{1_i}(x_1)$ or $n_{1_j}(\neg x_1)$, then a segment is placed connecting the chain v_{k_3} to $n_{1_j}(\neg x_1)$.

6.4. Connecting the gadgets

Although the polygonal curve P does not have to be planar, it must be a single continuous curve. Here, we will show that all the gadgets and segments can be connected to form P. The non-planarity allows us to focus on a single clause gadget to show one way in which everything can be connected. We have to be careful that we do not connect two nodes that would change the reduction such as connecting two end nodes at a clause – $c_{k_1}, c_{k_2}, c_{k_3}$ for clause C_k. For simplicity, we can connect all variables together and all the beginning and end switch points. Let $q_1 = p_{1_1}$ and then connect the variable gadgets by adding in the edge $\overline{p_{k_F} p_{k+1_1}}$ for all variables $1 \leq k \leq N-1$, and the last variable node p_{N_F} connects to a vertex in C_1.

We show a simple example of three variables and a clause in Fig. 11 without the connecting segments between gadgets. Let this be clause C_k, and the connected variables be x_1, x_2, x_3, at nodes n_{t_i} or n_{t_j} where $1 \leq t \leq 3$ and let n_{t_j} be the end node of the curve beginning with n_{t_i} (this will be either $n_{t_{i-1}}$ or $n_{t_{i+1}}$). We are only concerned about the end nodes of curves connected to the clause gadget. The other chains will be taken care of separately, including those which we will ignore for now (the switch component chains n_{1_3} to n_{1_4} and n_{3_3} to n_{3_4} in our example).

The end nodes of the curves that need to be connected are c_{k_1}, c_{k_2}, c_{k_3}, $n_{1_j}(n_{1_1})$, $n_{2_i}(n_{2_4})$, $n_{3_j}(n_{3_1})$. These can be connected as a single chain, with every edge longer than ε, by creating the segments $\overline{n_{1_j} c_{k_2}}, \overline{n_{2_i} n_{3_j}}$, and then c_{k_1} and c_{k_3} are the end nodes of the new curve. If we do this for all clauses, then we can connect the clauses with the segments $\overline{c_{k_3} c_{k+1_1}}$ for $1 \leq k \leq M-1$.

The only remaining unconnected curves are the switch components that are not tied to a clause gadget. These can be connected in any order provided the end nodes are not within ε, and we do not introduce a loop. This is straightforward by connecting every other switch component curve (never creating the segments $\overline{n_{t_{i-1}} n_{t_i}}$ or $\overline{n_{t_i} n_{t_{i+1}}}$ for $1 \leq t \leq N$), and then connecting all the skipped curves.

Fig. 12 shows the example from Fig. 11 with the connections and assignments $x_1 = 1$, $x_2 = 0$, and $x_3 = 0$. Our assignments allow the clause to be satisfiable. The example is one continuous polygonal curve P from the start of x_1 to the end at the clause gadget from the x_3 chain (the node c_{k_3}). If we had set $x_3 = 1$ notice that our Q would not have had a point to use within ε of c_{k_3}.

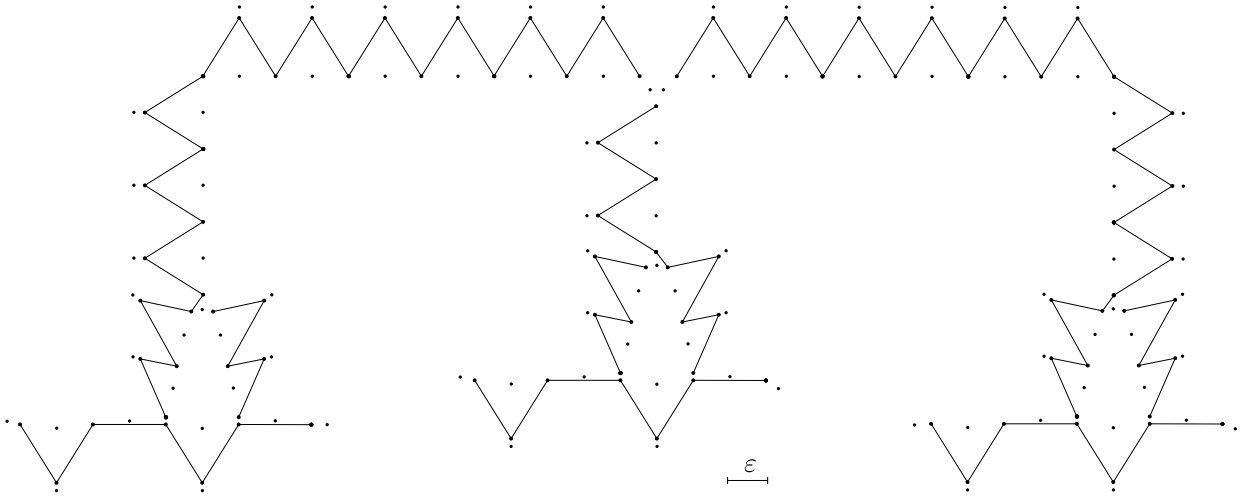


Fig. 11. Example USM clause with three variables $C_k = (\neg x_1 \vee x_2 \vee \neg x_3)$ without the connections shown between gadgets.

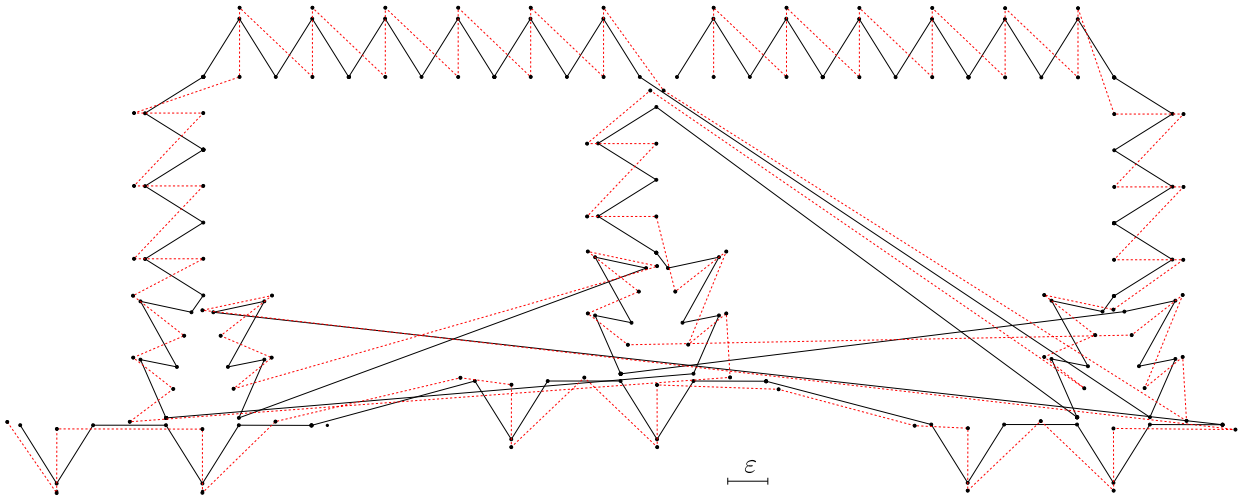


Fig. 12. Example USM clause with three variables $C_k = (\neg x_1 \vee x_2 \vee \neg x_3)$ with assignments $x_1 = 1$, $x_2 = 0$, and $x_3 = 0$.

6.5. The reduction

Now that we have covered the construction in detail and how it retains the properties of a planar 3-SAT instance, we can prove the complexity of USM. Following, we also look at the complexity based on the continuous Fréchet distance.

Theorem 4. *The discrete unique set-chain matching (USM) problem is **NP**-complete.*

Proof. We are given a planar 3-SAT instance $G_\varphi = \{V, E\}$ with vertices $V = X \cup C$ such that the vertices represent variables $X = \{x_1, x_2, \ldots, x_N\}$ and clauses $C = \{C_1, C_2, ..., C_M\}$, and the edges $E = \{e_1, e_2, \ldots, e_Z\}$ connect variables to clauses with the degree of each $C_i \in C$ being three. Given the planar 3-SAT instance G_φ, we construct a polygonal curve P and a point set S using an $\varepsilon > 0$ based on the method described. This construction takes $\mathcal{O}(|C| + |X| + |E|)$ for constructing P and S and is thus polynomial. The sizes of P and S are dependent on ε and the metric space. In general, for any edge $e_i \in E$ in the space, where $\|e_i\|$ is the length of the edge, there are $\lceil \|e_i\|/\varepsilon \rceil$ points in S and nodes of P used to transfer information along that edge.

We also refer to the 3-SAT equation φ derived from G_φ for the satisfiability of G_φ. The planar 3-SAT equation φ derived from G_φ is satisfiable if and only if there exists a polygonal curve Q with nodes from the set S such that $d_F(P, Q) \leq \varepsilon$ and each point represents a unique node in Q.

In the forward direction, we look at the value of φ. First, we assume φ is satisfiable. For every clause, there is at least one variable which has a **true** value. In our construction this means at least one chain does not need a point from the center of the clause gadget, and thus we can easily find a Q such that $d_F(P, Q) \leq \varepsilon$.

If φ is unsatisfiable, then there is at least one clause where all three variables have a **false** value. This means there is a clause gadget in our construction where all three chains are in a **false** setting, and all need a point in the clause gadget center (Fig. 10(a)). However, since there are only two points within ε of the clause gadget chains (the points $c_{i_1}, c_{i_2}, c_{i_3}$ for clause gadget C_i), one chain must use a point outside the clause gadget. This causes $d_F(P, Q) > \varepsilon$.

In the other direction, assume there exists a curve Q through $S' \subset S$ such that $d_F(P, Q) \leq \varepsilon$. There must be at least one **true** chain at each clause gadget, and since the three chains propagate this setting from the variable, we know at least one variable (or complement) was **true**. Thus, for every variable attached to a clause, it has the correct **true** or **false** setting. Therefore, if $d_F(P, Q) \leq \varepsilon$, then the current assignment of each variable also satisfies φ.

If no curve Q exists such that $d_F(P, Q) \leq \varepsilon$, then there is at least one clause gadget where all three chains had **false** settings and needed an extra point for Q within the clause gadget. Since the variable gadgets and switch components always have a curve within ε, the problem must occur in a clause. Again, this only happens if all three chains have a **false** setting, and similarly to the previous example, these propagated along the chains from the attachments to the variable gadgets. Thus, there must also exist a clause in φ where all three variables are **false**.

Last, we know the problem is in **NP**. Given an instance I we can check whether $d_F(P, I) \leq \varepsilon$ in polynomial time via Theorem 1. □

Our reduction is based on the discrete Fréchet distance, but our construction also ensures that any resulting curve Q is within ε of P along the edges as well. Thus, our reduction can be adapted to prove that USM is also **NP**-complete for the continuous Fréchet distance. This result was also recently proven independently and with a unique reduction in [8].

Corollary 1. *The unique set-chain matching (USM) problem based on the continuous Fréchet distance is **NP**-complete.*

Proof. This can be proven based on the polygonal curves P and Q being constructed of straight line segments. Given two line segments $a = \langle p_1, p_2 \rangle$ and $b = \langle p'_1, p'_2 \rangle$, it is straightforward to see that if $d(p_1, p'_1) \leq \varepsilon$ and $d(p_2, p'_2) \leq \varepsilon$, then under the continuous Fréchet distance $d_{\mathcal{F}}(a, b) \leq \varepsilon$. This does not account for instances where multiple nodes are matched with a single node of the other curve, but it gives an intuitive idea for why it is true.

Formally, it is known that for any two polygonal curves P, Q, the relationship $d_{\mathcal{F}}(P, Q) \leq d_F(P, Q)$ holds, where $d_{\mathcal{F}}$ is the Fréchet distance and d_F is the discrete Fréchet distance [4]. Thus, if both P and Q are polygonal curves and the problem is **NP**-complete for the discrete Fréchet distance within ε, it also holds for the continuous Fréchet distance within an $\varepsilon' \leq \varepsilon$. Finally, for polygonal curves in the plane, the continuous Fréchet distance can be calculated in $\mathcal{O}(mn \log mn)$ [3]. Thus, given an instance I based on a set S and polygonal curve P, we can verify the answer in polynomial time of $\mathcal{O}(|I||P| \log |I||P|)$. □

7. Conclusion

In this paper we have outlined and extended the discrete set-chain matching problem and other variations based on restricting our selection to unique nodes, the number of nodes allowed in the curve, or the number of points to choose from. We proved that two variations are **NP**-complete, and the unique point variation is still **NP**-complete when based on the continuous Fréchet distance. We proved that the other variation is polynomial, and gave the algorithm and pseudocode for solving the optimization version of the problem. We conclude with some open problems and further research directions for this work.

(1) What are the complexities based on maximizing the number of vertices in Q?

(2) We can also reverse the problem – if we are given a set size for Q, can we minimize the discrete Fréchet distance between P, Q, i.e., $d_F(P, Q) \leq \varepsilon$?

(3) What are the complexities with imprecise input? How difficult is it to find the minimum and maximum length Q while respecting the discrete Fréchet distance? This builds off computing the discrete Fréchet distance with imprecise input in general [19].

Acknowledgements

We would like to acknowledge all of the reviewers who have helped make this research better. We thank Paul Accisano for some early reviews related to the reductions. We are also grateful to Guohui Lin for his support during the preparation of this work.

References

[1] H. Alt, A. Efrat, G. Rote, C. Wenk, Matching planar maps, J. Algorithms 49 (2) (2003) 262–283.
[2] M. Fréchet, Sur quelques points du calcul fonctionnel, Rend. Circ. Mat. Palermo (1884–1940) 22 (1) (1906) 1–72, http://dx.doi.org/10.1007/BF03018603.
[3] H. Alt, M. Godau, Computing the Fréchet distance between two polygonal curves, Internat. J. Comput. Geom. Appl. 5 (1995) 75–91.
[4] T. Eiter, H. Mannila, Computing discrete Fréchet distance, Technical report CD-TR 94/64, Information Systems Department, Technical University of Vienna, 1994.

[5] A. Maheshwari, J.-R. Sack, K. Shahbaz, H. Zarrabi-Zadeh, Staying close to a curve, in: Proceedings of the 23rd Canadian Conference on Computational Geometry, CCCG'11, 2011, August 10–12, 2011.
[6] T. Wylie, Discretely following a curve, in: The 7th Int. Conf. on Combinatorial Optimization and Applications, COCOA'13, in: Lecture Notes in Comput. Sci., vol. 8287, 2013, pp. 13–24.
[7] T. Wylie, B. Zhu, Discretely following a curve (short abstract), in: Computational Geometry: Young Researchers Forum, CG:YRF'12, 2012, pp. 33–34.
[8] P. Accisano, A. Üngör, Hardness results on curve/point set matching with Fréchet distance, in: Proc. of the 29th European Workshop on Computational Geometry, EuroCG'13, 2013, pp. 51–54.
[9] K. Shahbaz, Applied similarity problems using Fréchet distance, Ph.D. thesis, Carleton University, 2013.
[10] A. Maheshwari, J.-R. Sack, K. Shahbaz, Visiting all sites with your dog, CoRR, abs/1211.4559.
[11] A. Mosig, M. Clausen, Approximately matching polygonal curves with respect to the Fréchet distance, Comput. Geom.: Theory Appl. 30 (2) (2005) 113–127.
[12] P.K. Agarwal, R.B. Avraham, H. Kaplan, M. Sharir, Computing the discrete Fréchet distance in subquadratic time, in: Proc. of the 24th Annual ACM–SIAM Sym. on Discrete Algorithms, SODA'13, SIAM, 2013, pp. 156–167.
[13] G.K. Das, R. Fraser, A. Lòpez-Ortiz, B.G. Nickerson, On the discrete unit disk cover problem, in: Proc. of the 5th Int. Conf. on WALCOM: Algorithms and Computation, WALCOM'11, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 146–157.
[14] R. Fraser, A. Lòpez-Ortiz, The within-strip discrete unit disk cover problem, in: Proc. of the 24th Canadian Conf. on Computational Geometry, CCCG'12, 2012, pp. 53–58.
[15] R. Acharyya, M. B., G.K. Das, Unit disk cover problem, CoRR, abs/1209.2951.
[16] N.H. Mustafa, S. Ray, Improved results on geometric hitting set problems, Discrete Comput. Geom. 44 (4) (2010) 883–895, http://dx.doi.org/10.1007/s00454-010-9285-9.
[17] D. Lichtenstein, Planar formulae and their uses, SIAM J. Comput. 11 (2) (1982) 329–343.
[18] D.E. Knuth, A. Raghunathan, The problem of compatible representatives, SIAM J. Discrete Math. 5 (3) (1992) 422–427.
[19] H.-K. Ahn, C. Knauer, M. Scherfenberg, L. Schlipf, A. Vigneron, Computing the discrete Fréchet distance with imprecise input, Internat. J. Comput. Geom. Appl. 22 (1) (2012) 27–44.
[20] T. Wylie, The discrete Fréchet distance with applications, Ph.D. thesis, Montana State University, 2013.
[21] A. Wolff, A simple proof for the NP-hardness of edge labeling, Technical report W-SPNPH-00, Institut für Mathematik und Informatik, Universität Greifswald, 2000.
[22] M. Jiang, Map labeling with circles, Ph.D. thesis, Montana State University, 2005.