Florida International University Technical Report **TR-2013-02-25**

Geometric Avatar Problems

Mario E. Consuegra, Giri Narasimhan, Shin-ichi Tanigawa[†]
Florida International University
[†] Kyoto University

mcons004@fiu.edu giri@cs.fiu.edu tanigawa@kurims.kyoto-u.ac.jp

Abstract

We introduce the concept of *Avatar problems* that deal with situations where each entity has multiple copies or "avatars" and the solutions are constrained to use at most (or exactly) one of the avatars. The resulting set of problems show a surprising range of hardness characteristics and elicit a variety of algorithmic solutions. In particular, we show how to extend the concept of ϵ-kernels to find approximation algorithms for some avatar problems.

1 Introduction

We introduce a family of optimization problems which we call *Avatar problems*. The main feature of this family of problems is that their input entities have multiple replicas (or copies, or *avatars*), but their output is constrained to use exactly one of the copies. Avatar problems have many applications in resource allocation problems. For example, storage systems may maintain multiple copies of data items; a purchase for a single part can be made by picking one of several brands or from one of several retail outlets; a task can be assigned to any one of a few select specialists. These applications involve choosing from a limited set of choices in order to achieve efficient use of resources.

An example of a problem containing avatars is the *traveling salesman problem with neighborhoods*, where the challenge is to find a tour of minimum length that visits each of the neighborhoods (i.e., regions or sets of points). This problem is known to be NP-Hard even in Euclidean space [10, 14], and it is known to be $(2-\epsilon)$ inapproximable [18]. Approximation algorithms have been found for geometric instances [7]. This problem can be thought of as an *Avatar problem* since each neighborhood provides a set of choices, one of which needs to be visited.

Another related problem is the job interval selection problem (JISP) [6]. In this problem the input is a set of n jobs assigned to a worker. Each job is a set of one or more intervals on the real line, and we must select one interval for each job such that we schedule as many jobs as possible by picking non-overlapping intervals. In the k-avatar version of JISP each job is a set of at most k intervals. Avatar problems share some overlap with the area of parameterized complexity. The study of the complexity of a *k-avatar* problem as k goes from 1 to ∞ provides better understanding of the complexity landscape of the problem. A model that is similar to the *avatar* model is the indecisive (uncertain) points model, see [12]. There the input is a set of elements where each element can take a value over a set of candidate data points as governed by a probability distribution over candidate points.

In this paper we introduce and study the *avatar* versions of classical algorithmic problems and evaluate their hardness. Given any optimization (or decision) problem, its avatar version is required to achieve the same optimization (or decision) over all possible instances where each instance is created by assigning each element a_i to **exactly** one of k possible values. The main results are summarized below, and is indicative of how "choice" affects the complexity of these problems in different ways.

Results: We consider geoemtric avatar problems in Sections 2 through 4. For points on a line, in section 2, we design a $\mathcal{O}(n^2 \log n)$-time algorithm for the 2-avatar maximum minGap problem for inputs in $\mathcal{R}^d$, and a 2-approximation polynomial time algorithm for the k-avatar minimum maxGap problem for points on a line. In section 3 we design a polynomial-time $(1+\epsilon)$-approximation algorithm for the k-avatar convex hull problem in $\mathbb{R}^d$. More significantly, in the process, we extend the concept of ϵ-kernels to the avatar world and show how to compute it efficiently for a k-avatar point set in $\mathbb{R}^d$. The ϵ-kernel result was also used to design polynomial-time $(1+\epsilon)$-approximation algorithms for the avatar versions of the following geometric problems: smallest volume axis-aligned enclosing hyperbox; and in the Appendix, smallest axis-aligned perimeter and diameter. Finally, a significant result here is that the 2-avatar version of the geometric minimum spanning tree problem, even for points in the plane, is NP-Complete; see section 4. For unweighted graphs, we show that the k-avatar reachability problem is NP-Complete, and for weighted graphs, we show that the k-avatar shortest path problem is inapproximable to any constant factor unless $P = NP$. Additionally there are a couple of "warm up" problems in the Appendix (see sections 7.1 and 7.2).

We establish some basic notation for this paper. Let $L = \{a_1, a_2, \ldots, a_n\}$ be a set of n k-avatar entities. In other words, for each entity $a_i \in L$, one can assign a_i to one of the k avatar values from the set $Av(a_i) = \{v_i^{(1)}, v_i^{(2)}, \ldots, v_i^{(k)}\}$. An *avatar assignment* for entities in L, denoted by $A(\cdot)$, is an assignment of a single avatar value to each entity in L. Thus, $A(a_i) \in Av(a_i)$. Let $A(L)$ denote the set of values assigned to each element in L.

2 Avatar Minimum and Maximum Gaps

Given a set of points $\{x_1, \ldots, x_n\}$ on a line, we define the *minGap* (resp. *maxGap*) as the smallest (resp. largest) gap between consecutive items in the sorted order. The avatar version of the *maximum minGap* and *minimum maxGap* problems can be stated as follows: *Given a set of n k-avatar entities, find an avatar assignment that results in the maximum min-*

Gap (resp. minimum maxGap); these two problems will be tackled in the following subsections. More formally, assume that we are given a set of k-avatar entities $L = \{a_1, a_2, \ldots, a_n\}$, where each entity a_i can be assigned one values from the set $\{v_i^{(1)}, v_i^{(2)}, \ldots, v_i^{(k)}\}$.

Avatar Maximum minGap We present a polynomial time algorithm for the 2-avatar version of *maximum minGap* problem. It is clear that the minGap must be between a pair of points from the set of all avatar values,

$$\bigcup_{a_i \in L} Av(a_i) = \bigcup_{a_i \in L} \{v_i^{(1)}, \ldots, v_i^{(k)}\} \qquad (1)$$

We first solve the decision problem of determining if there exists an avatar assignment so that the minGap is at least B; this is achieved by giving a polynomial-time reduction to 2SAT. The construction creates two complementary boolean variables, x_i and $\neg x_i$, to represent the two avatars of entity a_i. For every pair of values that are not avatars of each other and that have a distance of at most B, a clause is created to ensure that the corresponding boolean variables are not simultaneously set to true; a conjunction of these clauses generates an instance of 2SAT. It is easily shown that the resulting 2SAT formula is satisfiable if and only if the original 2-Avatar Maximum minGap problem has a minGap that is no smaller than B. Given the linear time algorithm for 2SAT [2], it is not difficult to see that the above algorithm takes $\mathcal{O}(n^2)$ time, and that the maximum minGap can be found in $\mathcal{O}(n^2 \log n)$ time by doing a binary search on the sorted list of all interpoint distances.

The above reduction to 2SAT for the 2-avatar minGap problem readily generalizes to the case where the entity values are points in d-dimensional space. However, the k-avatar minGap problem is NP-complete, and can be proved by a trivial adaptation of the proof of NP-Completeness of the problem of finding a *System of q-Distant Representatives* proved by Fiala et al. [9].

Theorem 1 *The 2-avatar maximum minGap problem for n points in $\mathbb{R}^d$ can be solved in $\mathcal{O}(n^2 \log n)$ time. The corresponding k-avatar problem for $k > 2$ is NP-hard.*

Avatar Minimum maxGap The avatar minimum maxGap problem appears to be harder than the avatar maximum minGap problem. While an exact polynomial time algorithm for the minimum maxGap problem remains open, below we present an approximation algorithm for the k-avatar minimum maxGap problem for points on a line.

Let B^* be the length of the minimum MaxGap, where the minimum is over all possible avatar assignments. We will perform binary search on the sorted list of all interpoint distances in order to find good lower and upper bounds B_ℓ and B_u for B^* such that $B_\ell \leq B^* \leq B_u$. Establishing bounds for the ratio between the lower and upper bounds gives an approximation for B^*. A sorted list of interpoint distances can be computed in $O(n^2k^2 \log nk)$. For a given value of B during this binary search we need to solve the decision problem of determining if there exists an avatar assignment so that the maxGap is at most B. The algorithm described below will give an approximate solution to this decision problem in the following sense. If the algorithm says "NO", then maxGap is greater than B. If the algorithm says "YES", then the maxGap is at most $2B$.

Let V denote the set of kn avatar values mapped on to the real line. Any avatar assignment is a subset of n points from V. A partition of the line into infinite number of disjoint abutting cells each of size B (see Fig. 1) is called a *valid* partition if there exists an avatar assignment such that all the points in the assignment are contained in a sequence of consecutive non-empty cells. Therefore, it follows that if there exists an avatar assignment for L such that the resulting point set has maxGap at most B then every partition of the line into infinite cells of size B is *valid*. The consequence is that if there is any partition of the line into infinite cells of size B that is not valid, then we know for sure that the maxGap for every assignment is greater than B. The difficulty is that the converse need not be true. Even though the assigned values appear in a sequence of consecutive cells, the maxGap could be between two items in adjacent cells that are nearly $2B$ apart, a key observation that leads to a 2-approximate algorithm. For example, in Fig. 1, v_6^2 and v_5^2 are in adjacent cells (of the partition with vertical dotted lines) but are almost $2B$ apart.

Given B, a fixed infinite partition of the line into cells of size B, and a fixed sequence of consecutive cells, we check if that partition is valid for some avatar assignment of L by a reduction to Network Flow. Briefly, we construct a bipartite network where one partition P has vertices corresponding to entities $a_i \in L$ and the other parition Q has vertices corresponding to cells of the partition. There is an edge from

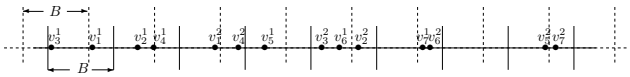


Figure 1: *Two different infinite partitions of the line are shown. The partition with dotted vertical lines is* valid*, while the partition with solid vertical lines is not* valid*. The valid partition is achieved by an avatar assignment that picks all choices with superscript 2.*

a vertex $p \in P$ to a vertex $q \in Q$ if the entity corresponding to p has an avatar in the cell corresponding to q. Finally, the reduction involves showing that the network has a flow of n if and only if the partition is valid. For lack of space, details of the algorithm are provided in the Appendix (Algorithm 2 in Section 7.3). As mentioned above, we perform binary search on the sorted list of interpoint distances until we find two adjacent gaps B_{i-1} and B_i in the list of gaps such that $B_{i-1} \leq B_i$, Algorithm 2 returns `NO` for all partitions into cells of length B_{i-1}, and returns `YES` for at least one partition into cells of length B_i. Thus, $B_{i-1} < B^*$. Since the smallest possible gap attainable that is larger than B_{i-1} is B_i, we have $B_i \leq B^*$. Also, since we have a partition into cells of length B_i for which we can find an entity-avatar assignment where all the chosen points are in a set of adjacent cells such that each cell in that cell contains a chosen point, we can use that avatar assignment to produce an assignment with a maximum gap no larger than $2 \cdot B_i$. Hence we have that $B_i \leq B^* \leq 2 \cdot B_i$. This gives us a polynomial-time 2-approximation algorithm for the 1D k-avatar minimum maxGap problem. The hardness of the avatar minimum maxGap for points in $\mathbb{R}^d$ remains open, even for $d = 1$.

Theorem 2 *The k-avatar minimum maxGap problem for points on a line has a 2-approximate algorithm that runs in $O(n^3 \cdot k^3 \log(nk))$ time.*

2.1 Avatar Line Segment Intersection problem:

We mention briefly that the *2-avatar* version of the classical line segment intersection problem (*is there an avatar assignment that ensures that none of the line segments intersect)* can be solved in polynomial time ($\mathcal{O}(n^2)$) by a reduction to 2SAT. The reduction involves representing the avatars $s_i^{(1)}$ and $s_i^{(2)}$ of each entity a_i as complementary boolean variables x_i and $\neg x_i$ respectively, and representing a pair of intersecting line segments by a 2SAT clause that prevents both segments to be chosen. The same reduction, however, does not solve the optimization version of the 2-avatar line segment problem, where the goal is find the avatar assignment that produces the largest subset of non-intersecting line segments (or conversely the smallest number of intersecting line segments). However, the reduction does allow us to provide an approximation guarantee of 0.93 (based on the best approximation algorithms for MAX-2SAT [8]).

3 Avatar Convex Hulls

Let L be a set of k-avatar entities where each entity can be assigned one of k different points in d-dimensional space. The k-avatar convex hull of L is the convex set that contains at least one avatar for each entity $a \in L$ and that minimizes a specific measure (such as the perimeter, surface area, or volume). The computational complexity of the problem of computing the avatar minimum convex hull remains an open problem. Related work includes results on the minimum and maximum convex hull for a set of points with imprecise locations [19, 13].

Here we discuss two avatar convex hull problems. First, we consider a special case of the problem where all avatars of a single entity are collinear and lie on a line parallel to the line containing the avatars of any other entity. We refer to this problem as the avatar convex hull problem for *parallel entities*. Due to limited space, the dynamic programming based algorithm for this special case is relegated to the Appendix (see section 7.4). More importantly, we present an ϵ-approximation algorithm for the avatar convex hull problem.

3.1 Approximate Avatar Convex Hulls

A *smallest avatar convex hull* is a convex hull that has minimum perimeter over all possible avatar assignments. The results can be extended to minimum area/volume convex hulls. We present an algorithm that finds an ϵ-approximate smallest avatar convex hull for the k-avatar convex hull problem in $\mathbb{R}^d$.

For any point set $X \subset \mathbb{R}^d$, let $\omega(u, X)$ denote the *directional width* of X in direction u (see Fig. 2 (a)). A subset $Q \subseteq P$ is called an ϵ-approximation of P if for all directions $u \in S^{d-1}$ we have $(1 - \epsilon)\omega(u, P) \leq \omega(u, Q)$. Our proposed algorithm finds an ϵ-approximate smallest convex hull $\mathcal{CH}(Q)$ by returning a set of avatar points $Q \subseteq A'(L)$ for some avatar assignment $A'(L)$ such that $(1-\epsilon)\omega(u, \mathcal{CH}^*(L)) \leq$

$\omega(u, \mathcal{CH}(Q))$, where $\mathcal{CH}^*(L)$ is the minimum avatar convex hull of L. Using the terminology of Agarwal et al. [1], one can think of the set Q as the avatar equivalent of an ϵ-kernel. This is formalized in the following definition of an *avatar ϵ-kernel* whose width along any direction is within a $1-\epsilon$ factor of the width of the optimal hull along that direction.

Definition 1 *Given a set L of n k-avatar entities, we say that a point set Q is an* avatar ϵ-kernel *of L if and only if $(1-\epsilon)\omega(u, \mathcal{CH}^*(L)) \leq \omega(u, Q), \forall u \in \mathbb{S}^{d-1}$, where $\mathbb{S}^{d-1}$ is the unit hypersphere centered at the origin.*

The following procedure for finding a *diameter-oriented bounding box* $\mathcal{B}$ of a set S of points in R^d was described by Bareqet and Har-Peled [3]. Let $\mathcal{D}(\mathcal{S})$ be the diameter of S and let $s_1, t_1 \in S$ s.t. $|s_1t_1| = \mathcal{D}(S)$. Let H be a hyperplane perpendicular to s_1t_1 and let Q be the orthogonal projection of S onto H. We again compute two points $s_2, t_2 \in Q$ s.t. $|s_2t_2| = \mathcal{D}(Q)$. Once again we project Q onto a hyperplane H' perpendicular to s_1t_1 and s_2t_2 and determine the diameter $\mathcal{D}(Q')$ of the projection Q onto H' and select two more points $s_3, t_3 \in Q'$ s.t. $|s_3t_3| = \mathcal{D}(Q')$. After d iterations of this process we have a diameter-oriented bounding box $\mathcal{B}(S)$ of S with the diameter in each iteration determined by the direction from s_i to t_i, for $i = 1, 2, \ldots, d-1$.

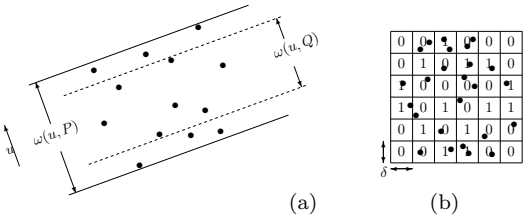


Figure 2: *(a) Directional Width (b) ϵ-grid Z*

Note that $\mathcal{CH}^*$ must cover a set S of $2 \cdot d$ avatar points of an avatar assignment such that the diameter-oriented bounding box $\mathcal{B}(\mathcal{S})$ is exactly the same as the diameter-oriented bounding box $\mathcal{B}(\mathcal{CH}^*)$. See Algorithm 1 for the pseudocode for the following procedure. We pick all possible subsets of $2 \cdot d$ avatar points of L, of which there are $\binom{n \cdot k}{2 \cdot d}$. For each subset S_i, first check that no two points in S_i are in the same avatar set, then find the diameter-oriented bounding box $B_i = \mathcal{B}(S_i)$. If every entity in L has an avatar point inside B_i then it is possible that $B_i = \mathcal{B}(\mathcal{CH}^*)$, otherwise we can discard B_i. We find an ϵ-approximate minimum avatar convex hull $\mathcal{CH}_i$ of all the points inside B_i and output the smallest one $\mathcal{CH}_{min}$. Since $\mathcal{B}(\mathcal{CH}^*) = B_i$ for some i, $\mathcal{CH}_{min}$ will ϵ-approximate $\mathcal{CH}^*$. The following lemma from [1] is useful for this proof. A point set is α-fat if its convex hull (a) is contained in a hypercube $\mathcal{H}$ and (b) contains a copy of $\mathcal{H}$ sharing the same center as $\mathcal{H}$, but shrunk by a factor $\alpha < 1$.

Algorithm 1 Computing ϵ-approximate min avatar convex hull

Require: L: set of n k-avatar entities; μ: a measure function of the size of a convex hull, $T(.)$ affine transform procedure
`let` $\mathcal{CH}_{min} =$ `null`
`let` S be the set of all possible sets of $2d$ avatar points of L.
for $S_i \in S$ **do**
  if no two points in S_i are the same avatar set **then**
    `let` $\mathcal{B}(S_i)$ be the *diameter oriented bounding box*
    `let` B_i be the set of all avatar points inside $\mathcal{B}(S_i)$
    `let` $\mathcal{CH}_i$ be the ϵ-approximate smallest avatar convex hull of B_i
    `find` $\mathcal{CH}_i$ with algorithm 3 for α-fat avatar point set $T(B_i)$
    $\mathcal{CH}_{min} =$ `Min`$(\mathcal{CH}_{min}, \mathcal{CH}_i)$
  end if
end for
return $\mathcal{CH}_{min}$

Lemma 1 [1] *For any point set P with non-zero volume in $\mathbb{R}^d$ there exists an affine transform M s.t. $M(P)$ is an α-fat point set where the hypercube $\mathbb{C} = [-1, +1]^d$ is the smallest enclosing box of $M(P)$ and s.t. a subset $Q \subseteq P$ is an ϵ-kernel of P iff $M(Q)$ is an ϵ-kernel of $M(P)$.*

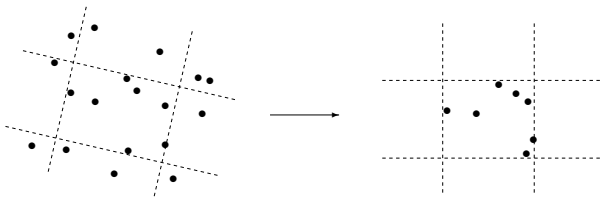


Figure 3: *Affine transform of space inside diameter-oriented bounding box of $2 \cdot d$ points.*

It is known that for a diameter oriented bounding box B with largest side $\mathcal{D}$, if we appropriately expand or contract the box along each direction until it becomes a hypercube of side $\mathcal{D}$ and scale it to the hypercube $\mathbb{C}$, then the transformed point set is an α-fat point set in $\mathbb{C}$ [3]. This transformation $T(B)$ of B as well as the transformed points can be computed in linear time, i.e., $O(n \cdot k)$ time. By Lemma 1, to compute an ϵ-approximate avatar convex hull of all the points in $\mathcal{B}$, we only need to compute an ϵ-approximate avatar convex hull of all the points in $\mathbb{C}$, which is computed as follows.

As in [3], let δ be the largest value such that $\delta \leq (\epsilon/\sqrt{d})\alpha$ and $\frac{1}{\delta}$ is an integer. We then partition the bounding hypercube into a uniform grid with cells of side length δ (see Fig. 2 (b)). However, applying the algorithm described in [3] does not help us compute ϵ-kernels in $\mathbb{C}$ because even though the point set may be α-fat, once we perform an avatar assignment, the resulting set may not be α-fat.

We need one other idea to compute ϵ-kernels in $\mathbb{C}$. The following procedure computes the ϵ-kernel in $T(B)$ (for details, see Algorithm 3 in the Appendix). Consider all possible assignments of binary values (0/1) to the cells in the grid (see Fig. 2 (b)). For the i^{th} binary assignment let Q_i be the set of cells that are assigned a value of 1. We call the set Q_i *legal* if each avatar entity has at least one element in at least one of the cells of Q_i, and it is possible to pick a representative point from each cell such that no two cells have representative points that are avatars of the same entity. Since there are $1/\delta^d$ cells, there are at most $2^{1/\delta^d}$ legal sets. In particular, if $A_{OPT}(\cdot)$ is the avatar assignment that leads to the optimal avatar convex hull, then it is easy to see that one of these legal sets must contain exactly the collection of cells with points from $A_{OPT}(\cdot)$.

We can determine if a given set of grid cells Q_i is legal by solving a network flow problem as follows. Create a set of vertices T such that each vertex in T represents a different cell in Q_i. Create a source vertex s with directed edges to each vertex in T. Create a set of vertices T' such that each vertex in T' represents a distinct point in some cell in Q_i. Add an edge from $u \in T$ to $u' \in T'$ if the corresponding cell in Q_i contains the corresponding point. Create another set of vertices T'' such that each vertex in T'' corresponds to an avatar entity. Add an edge from $u' \in T'$ to $u'' \in T''$ if u' is a possible assignment for the avatar entity u''. Finally add a sink vertex t and connect all vertices in T'' to t by an edge. All edges have capacity 1. A maximum flow of size $|T|$ from s to t will identify a representative point in each cell such that no two points are avatars of the same entity. It is easy to see that such a flow exists if and only if the corresponding set of cells Q_i is legal. The following theorem formalizes the result.

Theorem 3 *There is an algorithm that finds an ϵ-approximate smallest k-avatar convex hull in time $O((nk)^{(2d+3)} \cdot \frac{n}{\delta^d} \cdot (2d)^2 \cdot 2^{\frac{1}{\delta^d}(\frac{2}{\delta^{d-1}})^{\lfloor \frac{d}{2} \rfloor}})$, by finding an avatar ϵ-kernel Q of L, which by Definition 1 satisfies:*

$$(1-\epsilon)\omega(u, \mathcal{CH}^*(L)) \leq \omega(u, \mathcal{CH}(Q)), \quad \forall u \in S^{d-1} \tag{2}$$

The proof is sketched as follows. Given a legal set, Q_i, let $Q_i' \subseteq Q_i$ be the collection of highest and lowest cells in every hypercolumn containing at least one cell of Q_i. Let Q (resp., Q') be the set of representative points of cells in Q_i (resp., Q_i'). It is easy to see that Q is an ϵ-kernel of Q'. We argue that $A_{OPT}(\cdot)$, the avatar assignment that leads to the optimal avatar convex hull, occupies a collection of cells (call this set of cells Q_{OPT}), which would have been considered by our algorithm. While the algorithm may not have picked the points in the optimal avatar assignment, it is sure to pick one representative point from each of the cells in Q_{OPT}. Since for each point in there is at least one representative point that is within distance $\epsilon \cdot \alpha$ for every point in the optimal avatar assignment, we immediately have an avatar ϵ-kernel of the original input. The algorithm is fleshed out in some detail in the Appendix (see Section 7.5).

Approximate Smallest Volume Axis-Aligned Enclosing Hyperbox We can compute a $(1+\epsilon)$-approximate smallest volume axis-aligned enclosing hyperbox $B(L)$ containing an avatar of each entity in L after finding an ϵ'-kernel of L, for some constant ϵ'. Let $\mathcal{CH}(L)$ be the smallest avatar convex hull of a set L of k-avatar points. If Q is a k-avatar ϵ'-kernel of L such that $Q \subset \mathcal{CH}(L)$, then we have:

$$\begin{aligned}
(1-\epsilon') \cdot \omega(u, L) &\leq \omega(u, Q), \quad \forall u \in S^{d-1} \\
(1-\epsilon') \cdot \omega(u, L) &\leq \omega(u, Q), \quad \forall u \in [d] = \{e_1, e_2, \ldots, e_d\} \\
(1-\epsilon')^d \prod_{u \in [d]} \omega(u, L) &\leq \prod_{u \in [d]} \omega(u, Q)
\end{aligned}$$

There exists a constant c (function of ϵ' and d), such that $(1-c\epsilon') \leq (1-\epsilon')^d$, thus implying the following:

$$\begin{aligned}(1-c\epsilon') \prod_{u\in[d]} \omega(u,L) &\leq \prod_{u\in[d]} \omega(u,Q) \\ (1-c\epsilon') \cdot Volume(B(L)) &\leq Volume(B(Q))\end{aligned}$$

Thus by choosing $\epsilon = \frac{1}{1-c\epsilon'}$, we obtain a $(1+\epsilon)$-approximation of the smallest volume axis-aligned enclosing rectangle, since

$$1 \leq \frac{(1+\epsilon)\cdot Volume(B(Q))}{Volume(B(L))} \leq (1+\epsilon)$$

Theorem 4 *Given an exact algorithm for finding the smallest volume axis-aligned enclosing hyperbox that runs in time $O(n^a)$, there exists an algorithm that finds a $(1+\epsilon)$-approximate smallest volume axis-aligned avatar enclosing hyperbox in time $O((nk)^{(2d+3)} \cdot \frac{n}{\delta^d} \cdot (2d)^2 \cdot 2^{\frac{1}{\delta^d}} (\frac{2}{\delta^{d-1}})^{\lfloor \frac{d}{2} \rfloor} + (\frac{2}{\delta^{d-1}})^a)$.*

Similar results can be achieved for an approximate smallest avatar diameter and minimum perimeter axis-aligned avatar enclosing box, and is given in the Appendix (see sections 7.6 and 7.7).

4 Avatar Euclidean Minimum Spanning Tree

Let L be a set of k-avatar entities where each entity can be assigned one of k avatar points in the plane. We define an *avatar tree* to be a tree whose vertex set is a subset of all avatar points, but is constrained to include at most one avatar for each entity $a_j \in L$. An avatar tree is said to be *spanning* if its vertex set includes exactly one avatar point for each entity $a_j \in L$. An *avatar minimum spanning tree (MST)* is an avatar spanning tree of minimum weight. The problem of computing the *k-avatar Euclidean Minimum Spanning Tree (EMST)* for points in the plane is a natural and interesting problem. Chambers et al. [4] show that the related problem *best case connectivity under uncertainty* (BCU) is NP-Hard.

Theorem 5 .
The k-avatar EMST problem is NP-Complete.

It is sufficient to consider the general 2-avatar case, where each entity has one or two avatars. To prove the hardness of the problem we provide a reduction from Hamiltonian Path Problem in Cubic Directed Planar Graph (DHP). The reduction makes use of the slightly modified version of DHP (denoted by DHP_{st}) in which G has one vertex s with out-degree of one (and no incoming edges) and one vertex t with in-degree of one (and no outgoing edges). It can be shown that DHP_{st} is NP-complete.

Given an instance I of the DHP_{st} problem, we construct an instance of the avatar EMST problem by first forming a rectilinear planar layout of I with integer coordinates. This step is somewhat involved and technical and we have provided a sketch of this step in the Appendix (Section 7.8). An example of a rectilinear layout can be found in the left figure of Fig. 4.

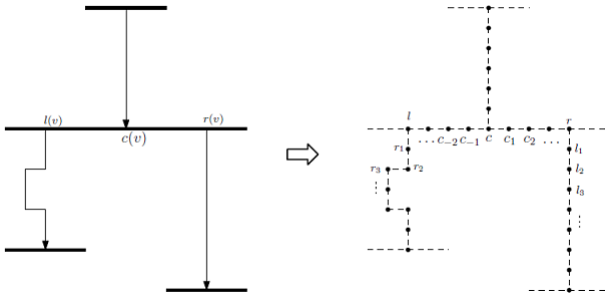


Figure 4: *Locating points and avatar pairs along edges of the planar rectilinear layout*

Next we construct an instance of the avatar EMST problem by placing a series of closely located points on each line segment in the layout. Then, we identify the avatar pairs. Finally, we show the equivalence of the two instances.

Informally, the distance between adjacent points placed on a line segment is 1 and the distance between any other pair is strictly greater than 1. It is important that we do not place points on "overhangs" (i.e., the left and right overhang of each horizontal line) (see Fig. 4). For vertex $v \in V$ with in-degree 1 and out-degree 2 in G, consider its corresponding horizontal segment in the layout. If $c(v)$ denotes the intersection point of the horizontal segment corresponding to v and the vertical incoming line incident on this segment, then we make two points horizontally next to $c(v)$ (denoted by $c_{-1}(v)$ and $c_1(v)$), as an avatar pair, ensuring that both branches of the fork at vertex v will not be simultaneously part of the EMST. We continue to pair the vertices on the horizontal segment and also continue along the two vertical edges leaving. As shown in Fig. 4, every vertex c_{-i} is paired with c_i, and continuing on, vertex r_j is paired with l_j.

If $v \in V$ is a vertex with in-degree 1 and out-degree 2 in G, then only pair the points on the horizontal segment. The result of this construction is a set of points forming the instance of the avatar EMST.

The critical proof that the EMST instance has an avatar spanning tree of a certain weight if and only if G has a Hamiltonian path from s to t is provided in the Appendix (see Theorem 9 in Section 7.9).

5 Avatar Problems in Graphs and Metric Spaces

In this section we consider the hardness of the avatar versions of vertex reachability and shortest paths in unweighted graphs. The results easily generalize to weighted graphs and metric spaces. Vertex reachability has strong ties to *rainbow connectivity* problems from the graph theory literature [5]. As before, in order to set the stage, we provide some formal definitions.

Avatar graph reachability A k-avatar graph $G(V, E, L, \mathcal{A})$ (or simply an "avatar" graph) consists of the following: a set of vertices V; a set of edges E connecting pairs of vertices in V; a set of entities $L = \{a_1, \ldots, a_m\}$; and a collection of disjoint avatar sets $\mathcal{A} = \{A_1, \ldots, A_m\}$ such that $\forall i$, $A_i \subseteq V$ is the avatar set for entity i, $|A_i| \leq k$, and $A_i \bigcap A_j = \phi$, if $i \neq j$. As with the definition of an avatar tree, an *avatar path* in G is a path p such that no two vertices on the path p are avatars of the same entity.

The *k-avatar reachability problem* is stated as follows: *Given an avatar graph G and two vertices s and t in G determine if there is an avatar path P from s to t.* Reachability is a fundamental graph problem and can be solved in linear time using simple techniques like DFS or BFS. Surprisingly enough, in the avatar setting it turns out to be NP-Complete, even for $k = 2$. The proof of the following theorem can be found in the Appendix (Section 7.10).

Theorem 6 *The k-avatar reachability problem is NP-Complete.*

5.1 Avatar Maximum Matching

Based on classical work of Edmonds in 1965, we know that non-bipartite weighted graph matching is solvable in polynomial time ($O(|V|^4)$, later improved using matrix multiplication); an excellent exposition can be found in Papadimitriou and Steiglitz [15]. We start with some definitions in order to consider the avatar version of the problem. Given a k-avatar graph $G(V, E, M, A, \alpha)$, an *avatar matching* is a matching of a set of vertices $V' \subset V$ such that for any $u, v \in V', \alpha(u) \neq \alpha(v)$. A *maximum avatar matching* of a k-avatar graph is an avatar matching of maximum cardinality. Given a weight function ℓ on the edges of the graph G, a *minimum weight maximum avatar matching* is a maximum avatar matching of minimum weight. We show that the avatar version of the weighted matching problem can be reduced to non-bipartite weighted graph matching.

The **k-avatar minimum weight maximum matching problem** is stated as follows: *Given a k-avatar graph $G(V, E, A, \alpha)$ with edge-weight function $\ell : E \rightarrow R$, find a maximum avatar matching of minimum weight.*

The reduction is intuitively straightforward. If an entity has k avatars, then it involves introducing $k - 1$ new vertices for each entity and connecting each of them by an edge of very small weight (say, 0) to each of its k avatar vertices. A minimum weight graph matching would then force the use of as many of these 0 weight edges as possible, guaranteeing the match of all but one avatar vertex for that entity to the newly introduced entity vertices (using 0-weight edges). The vertex that is not matched with a 0-weight edge will, in turn, be matched to some other vertex in G in a way as to achieve a minimum weight matching.

6 Open Problems

Several open problems still remain. The most notable ones that result from this paper include determining the time complexity of the k-avatar versions of (a) minimum MaxGap problem, and (b) convex hull.

Acknowledgments

The work of GN and MEC was partly supported by NSF Grant # 1018262. MEC was also partly supported by NSF Graduate Research Fellowship DGE-1038321. The authors thank Joshua Kirstein for useful discussions.

References

[1] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Approximating extent measures of

points. *J. ACM*, 51(4):606–635, July 2004.

[2] B. Aspvall, M. F. Plass, and R. E. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8(3):121–123, March 1979.

[3] G. Barequet and S. Har-Peled. Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. *Journal of Algorithms*, 38(1):91 – 109, 2001.

[4] E. W. Chambers, A. Erickson, S. P. Fekete, J. Lenchner, J. Sember, V. Srinivasan, U. Stege, S. Stolpner, C. Weibel, and S. Whitesides. Connectivity graphs of uncertainty regions. In *Proceedings of the 21st International Symposium, ISAAC 2010*, LNCS 6507, pages 434–445, 2010.

[5] G. Chartrand, G. L. Johns, K. A. McKeon, and P. Zhang. The rainbow connectivity of a graph. *Networks*, 54(2):75–81, 2009.

[6] J. Chuzhoy, R. Ostrovsky, and Y. Rabani. Approximation algorithms for the job interval selection problem and related scheduling problems. *Mathematics of Operations Research*, 31(4):730–738, 2006.

[7] A. Dumitrescu and J. S. B. Mitchell. Approximation algorithms for TSP with neighborhoods in the plane. *Journal of Algorithms*, 48(1):135 – 159, 2003. Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms.

[8] U. Feige and M. Goemans. Approximating the value of two prover proof systems, with applications to MAX 2SAT and MAX DICUT. In *Proceedings of the 3rd Israel symposium on Theory of Computing an Systems*, pages 182–189, 1995.

[9] J. Fiala, J. Kratochvíl, and A. Proskurowski. Systems of distant representatives. *Discrete Appl. Math.*, 145(2):306–316, January 2005.

[10] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[11] M. R. Garey, D. S. Johnson, and R. E. Tarjan. The planar hamiltonian circuit problem is np-complete. *SIAM J. Comput.*, 5(4):704–714, 1976.

[12] A. Jørgensen, M. Löffler, and J. M. Phillips. Geometric computations on indecisive and uncertain points. In *Proceedings of the 12th International Symposium, WADS 2011*, LNCS 6844, pages 536–547, 2011.

[13] W. Ju and J. Luo. New algorithms for computing maximum perimeter and maximum area of the convex hull of imprecise inputs based on the parallel line segment model. In *Proceedings of the CCCG*, 2009.

[14] C. H. Papadimitriou. The euclidean travelling salesman problem is NP-complete. *Theoretical Computer Science*, 4(3):237 – 244, 1977.

[15] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1982.

[16] J. Plesnk. The np-completeness of the hamiltonian cycle problem in planar digraphs with degree bound two. *Inf. Process. Lett.*, 8(4):199–201, 1979.

[17] P. Rosenstiehl and R. E. Tarjan. Rectilinear planar layouts and bipolar orientations of planar graphs. *Discrete and Computational Geometry*, 1:343–353, 1986.

[18] S. Safra and O. Schwartz. On the complexity of approximating TSP with neighborhoods and related problems. *Comput. Complex.*, 14:281–307, March 2006.

[19] M. van Kreveld and M. Löffler. Largest and smallest convex hulls for imprecise points. *Algorithmica*, 56(2):235–269, February 2010.

7 Appendix

7.1 Avatar Sorting

In this section, we present some simple "warm-up" problem that provide some intuition for tackling avatar problems. We start with *sorting*, which is a fundamental and well-studied algorithmic problem. Given a permutation Π, an avatar assignment $A(\cdot)$ for entities in L *achieves* Π if the sorted order of the (multi)set $A(a_i), a_i \in L$ is the same as Π. We look at the following problem: *find the permutation* Π *that is achieved by the largest number of sorted distinct avatar assignments*. We solve it by by reducing it to topological sort in an acyclic directed graph. Details can be found in the Appendix (Section 7.1.1).

7.1.1 Avatar most stable sorting

Here we present a solution for the problem: *find the permutation* Π *that is achieved by the largest number of distinct avatar assignments.* There exists an efficient algorithm that involves topological sort in a directed acyclic graph. The following algorithm solves the above problem. Given $L = \{a_1, \ldots, a_n\}$ and the avatars for each entity a_i, $Av(a_i) = \{v_i^{(1)}, \ldots, v_j^{(k)}\}$. For any pair of entities a_i and a_j there are k^2 possible avatar assignments for that pair of values. These k^2 arrangements result in two kinds of permutations, ones in which i appears before j and the remaining with j appearing before i.

We construct a directed acyclic graph (DAG) $G(V, E)$ as follows: For each entity a_i add a vertex v_i to V. We add a directed edge (v_i, v_j) in G if there are (strictly) more avatar assignments that put i before j than those that put j before i.

If there are an equal number of the two kinds of order-preserving permutations, then no edge connects the two corresponding vertices. Obviously, at most one of the two edges, (v_i, v_j) and (v_j, v_i), will be created, and the graph G has no cycles and is therefore a DAG. We state without proof that a permutation defined by any *topological sort* of the graph G gives us a permutation that is achieved by the largest number of avatar assignments $A(\cdot)$.

7.2 Avatar Maximum Matching

Based on classical work of Edmonds in 1965, we know that non-bipartite weighted graph matching is solvable in polynomial time ($O(|V|^4)$, later improved using matrix multiplication); an excellent exposition can be found in Papadimitriou and Steiglitz [15]. We start with some definitions in order to consider the avatar version of the problem. Given a k-avatar graph $G(V, E, M, A, \alpha)$, an *avatar matching* is a matching of a set of vertices $V' \subset V$ such that for any $u, v \in V', \alpha(u) \neq \alpha(v)$. A *maximum avatar matching* of a k-avatar graph is an avatar matching of maximum cardinality. Given a weight function ℓ on the edges of the graph G, a *minimum weight maximum avatar matching* is a maximum avatar matching of minimum weight. We show that the avatar version of the weighted matching problem can be reduced to non-bipartite weighted graph matching.

The **k-avatar minimum weight maximum matching problem** is stated as follows: *Given a k-avatar graph $G(V, E, A, \alpha)$ with edge-weight function $\ell : E \to R$, find a maximum avatar matching of minimum weight.*

The reduction is intuitively straightforward. If an entity has k avatars, then it involves introducing $k-1$ new vertices for each entity and connecting each of them by an edge of very small weight (say, 0) to each of its k avatar vertices. A minimum weight graph matching would then force the use of as many of these 0 weight edges as possible, guaranteeing the match of all but one avatar vertex for that entity to the newly introduced entity vertices (using 0-weight edges). The vertex that is not matched with a 0-weight edge will, in turn, be matched to some other vertex in G in a way as to achieve a minimum weight matching.

The following is a sketch of the proof for $k = 2$. Construct a graph $G'(V', E')$ with edge-weight function w' as follows. For each entity $a_i \in A$, create a new vertex in V'. Also, add each vertex $u \in V$ to V' as u'. Hence $V' = V \bigcup \{a_i | 1 \leq i \leq n\}$. Then we have $E' = E \bigcup \{(u, v) | u \in (V' - V), v \in (V' \bigcap V, \alpha(v) = u\}$. The weight $w'(e)$ of each edge $e \in E'$ is defined as follows:.

$$w'(e) = \begin{cases} w(e) & \text{if } e \in E, \\ 0 & \text{if } e \in E' - E, \end{cases} \tag{3}$$

Then we simply proceed to find a minimum weight maximum matching in G' solving it as a standard non-bipartite matching problem. We can use Edmonds's algorithm to solve a weighted maximum matching problem in any graph. Let $M' = \{u'_1, u'_2, \ldots, u'_{m'}\}$ be a minimum weight maximum matching in G' with $u'_i \in V'$ and m' even, then we say that $M = \{u_1, u_2, \ldots, u_m\}$ is a minimum weight maximum avatar matching of G.

We now prove that M is a minimum weight maximum matching of G.

Proof *If $M' = \{u'_1, u'_2, \ldots, u'_{m'}\}$ is a minimum weight maximum matching in G' with $u'_i \in V'$, then all of the n vertices in $(V' - V)$ are contained in that matching. See that $\forall a \in (V' - V)$ we have that $w'(a, v) = \epsilon \ \forall (a, v) \in E'$, and that $v \in (V' \bigcap V)$ by construction of G'. If there was a vertex $a \in (V' - V)$ such that $a \notin M'$, then there are two vertices $u \in M'$ and $v \in M'$ in $V' \bigcap V$ that are connected to a. Let x and y be matched to u and v respectively in M'. Then by construction we know that $x \notin (V' - V)$ and $y \notin (V' - V)$, which implies $w'(u, x) > w'(a, u)$, $w'(v, y) > w'(a, v)$. Then removing (u, x) from M' and adding (a, u) produces a maximum matching of lower weight than the initial matching M'. Then M' was not a minimum weight maximum matching, and this is a contradiction.*

Having shown that M' contains all of the n vertices in $(V' - V)$, we proceed to show that $M = M' - \{(a_i, v) \in M' | 1 \leq i \leq n, v \in (V' \bigcap V)\}$ is an avatar maximum matching of minimum weight. Since all the vertices in $\{a_i | 1 \leq i \leq n\}$ are matched to vertices in $V' \bigcap V$, then M contains only one avatar per entity since there are only two avatars per entity and each vertex $v \in M'$ is either matched to an entity vertex a_i or to a vertex that is an avatar of another entity a_j. Hence M is an avatar matching of G. Furthermore, M is a minimum weight maximum avatar matching of G. If M were not a minimum weight maximum avatar matching of G and there were another maximum avatar matching M_2 of G then there would be a maximum matching M'_2 with lower weight than M', which would be a contradiction. Therefore M is a minimum weight maximum avatar matching of G. ■

As mentioned in the text, generalizing the above proof for $k > 2$ is straightforward. The detailed proof will be provided in a full version of the paper.

7.3 Avatar Maximum Gap Reduction to Network Flow

Using a reduction to network flow, it can be shown that for a given value of B, a fixed infinite partition of the line into cells of size B, and a fixed sequence of consecutive cells, there exists a polynomial-time algorithm to determine if that partition is valid for some avatar assignment of L.

Below we formalize the ideas expressed above. For a given value of B, a fixed infinite partition of the line into cells of size B, and a fixed sequence of consecutive cells, we first show a polynomial-time algorithm to determine if that partition is valid for some avatar assignment of L. Consider the following reduction to network flow. Construct a bipartite directed graph with two sets P and Q such that each vertex $p_i \in P$ corresponds to entity $a_i \in A$, and each vertex $q_j \in Q$ corresponds to cell C_j in the infinite partition. Create a directed edge (p_i, q_j) with capacity 1 and minimum flow 0, if $\exists u \in C_j$ which is an avatar of a_i. Add a vertex s and add a directed edge $(s, p_i), \forall p_i$ with capacity 1 and minimum flow 0. Also we add a vertex t. Add a directed edge $(q_j, t), \forall q_j$ with capacity n and minimum flow 1. After this construction, we find the maximum $s - t$ flow of G'. If this flow is of value n, then there is an avatar assignment such that each cell in the partition contains one of the points used in the avatar assignment. If these cells are chosen to be contiguous, then we have effectively tested if the partition is valid for the specific cell size and choice of cells. In Algorithm 2, the call to procedure `makeNetwork` constructs the instance of the network flow problem mentioned above. This procedure takes as input the set of entities, the set of points, and a group of cells. In order to try every possible set of consecutive cells, the algorithm has two for-loops.

Algorithm 2 Determine if there is an avatar assignment such that all the cells that contain assigned points are adjacent to each other

Require: A: list of n entities
Require: V: list of points on the line
Require: α: avatar to entity relation
Require: B_r: cell length
  `let` $C_1, C_2, \ldots, C_m$ the ordered set of cells that contain points from V.
  for $(i = m - 1$ `down to` $i = 1)$ **do**
    for $(j = 1$ `to` $j = m - i)$ **do**
      $G' =$ `makeNetwork`$(A, V, (C_{j,\ldots,C_{j+i}}))$
      if $(n ==$ `maxFlow`$(G'))$ **then**
        return `YES`
      end if
    end for
  end for
  return `NO`

We need to try every possible "partition". Two partitions are different if the set of points in at least one cell are different in the partitions. We argue that there are at most kn different "partitions" and that we can efficiently generate every possible partition

with cells of size B. For each point v on the line we divide the line into cells such that v is at the leftmost end of the cell that contains it. These partitions are sufficient because otherwise, we can slide the "partitions", making sure that during the slide that no changes occur to the set of points contained, until some point is hit on the boundary. If at least one point has to be on a partition boundary, clearly there are at most kn partitions. Thus the algorithm ends up solving a series of network flow problems as shown in Algorithm 2.

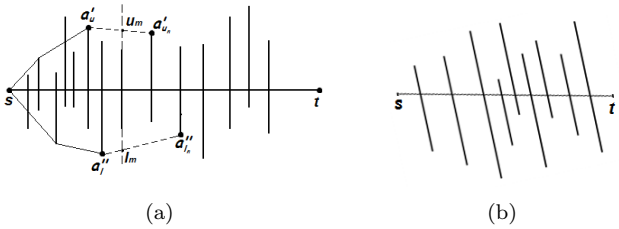
7.4 Avatar Convex Hull Problem for Parallel Entities

We first consider the case when $k = 2$ and later show how to generalize the solution to $k > 2$. Assume that entity $a_i \in L$ has two avatars $\{a'_i, a''_i\}$, both points in $\mathcal{R}^2$. Let s_i be the line segment joining the two points and t_i be the infinite line containing s_i. We now show a polynomial-time algorithm to solve the 2-*avatar convex hull for parallel entities* problem using dynamic programming. Our presentation here will assume that each t_i is vertical. However, the algorithm can be extended to the case when they are not vertical (see Fig. 5(b)). Let us assume without loss of generality (wlog) that $\forall a_i \in L$, a'_i has higher y-coordinate than a''_i. Also assume wlog that a'_i has smaller x-coordinate than a'_j for $i < j$. To solve the 2-*avatar convex hull problem for parallel entities* we first divide the initial problem into 4 subproblems. The final solution may have a'_1 (resp. a'_n) or a''_1 (resp. a''_n) as the leftmost (resp. rightmost) point on the hull. Since we do not know which one is correct, we try all four possible subproblems and pick the best of the solutions. Thus in each subproblem, we have only one leftmost point and only one rightmost point, both of which will be on the final convex hull. First, we label the leftmost and rightmost points as s and t (see Fig. 5(b)). The algorithm attempts to simultaneously extend a partial upper hull and a partial lower hull from s to t. Note that the line segment st will be contained in the convex hull. Also note that if both avatars of any entity a_i lie on the same side of the line st then we can discard the one that is farther from the line and instead make two copies of the closer one.

Next we present the details of the dynamic programming algorithm for the avatar convex hull problem for parallel entities. Define:

(i) $C[u, u_n, l, l_n, m]$ to be the smallest length polygonal chain consisting of a concatenation of the partial upper hull from s to a'_u ($s \rightsquigarrow a'_u$) and the partial lower hull from s to a''_l ($s \rightsquigarrow a''_l$), such that (a) $u \leq m < u_n$, (b) $l \leq m < l_n$, (c) the region enclosed between it and the polygonal chain $a'_u - u_m - l_m - a''_l$ is a convex region covering at least one avatar for each entity in $a_1, \ldots, a_m$. As shown in Fig. 5(a), u_m (resp. l_m) is the point of intersection between the line t_m (line containing all avatars of entity a_m) and the segment $a'_u a'_{u_n}$ (resp. segment $a''_l a''_{l_n}$). If any of the three conditions stated above (a)-(c) are not satisfied, then $C[u, u_n, l, l_n, m]$ is undefined.

(ii) $L[u, u_n, l, l_n, m]$ = Length of the polygonal chain $C[u, u_n, l, l_n, m]$.
If $C[u, u_n, l, l_n, m]$ is undefined, then $L[u, u_n, l, l_n, m] = \infty$.



(a) (b)

Figure 5: *(a) Figure for Avatar convex hull problem with parallel entities. (b) Case when the lines containing all avatars of an entity are parallel, but not necessarily vertical.*

The length function defined above satisfies the following recurrence:

$$L[u, u_n, l, l_n, m] = \begin{cases} L[u, u_n, l, l_n, m-1] & \text{if } (m < u) \text{ and } (m < l) \\ \min_{i<l} L[u, u_n, i, m, m-1] + |a''_i a''_m| & \text{if } l = m, \\ \min_{i<l} L[i, m, l, l_n, m-1] + |a'_i a'_m| & \text{if } u = m, \\ \infty & \text{otherwise} \end{cases} \quad (4)$$

Finally by setting $L[s, i, s, j, s] = 0$ as the base case and running the recurrence relation in Eq (4) we can get the length of the optimal convex hull by calculating L_{opt} as follows::

$$L_{opt} = \min_{u,l \in [n-1]} \{L[u, n, l, n, n-1] + |a'_u t| + |a''_l t|\} \quad (5)$$

To understand the correctness of the recurrence in Eq (4) we observe that the optimal solution $C[u, u_n, l, l_n, m]$ is determined by the following cases (assume wlog that $u < l$):

- First, note that $L[u, u_n, l, l_n, m-1] \leq L[u, u_n, l, l_n, m]$. If a_m does not have any avatar as part of the polygonal chain $C[u, u_n, l, l_n, m]$ and $u < l < m$ then at least one avatar of a_m is between the segments uu_n and ll_n. However, since $u < l \leq m-1$, one of the two avatars of a_{m-1} is also between the segments uu_n and ll_n. Then the claim is that $C[u, u_n, l, l_n, m] = C[u, u_n, l, l_n, m-1]$ and $L[u, u_n, l, l_n, m] = L[u, u_n, l, l_n, m-1]$, since otherwise, we can replace the polygonal chain $C[u, u_n, l, l_n, m]$ with the shorter polygonal chain $C[u, u_n, l, l_n, m-1]$ and one of the avatars of a_m would get covered since it lies in between the segments uu_n and ll_n.

- If one of the avatars of a_m is part of the polygonal chain $C[u, u_n, l, l_n, m]$ (assume wlog that $u < l = m$), then there must exist a point a''_i in the lower hull of the polygonal chain $C[u, u_n, l, l_n, m]$ such that the polygonal chain $C[u, u_n, i, l, m-1]$ concatenated with the polygonal chain $a''_i - a''_l - a'_{u_m} - a'_u$ encloses a convex region that contains at least one avatar from the entity set $[m]$. Since the recurrence relation tries out every possible i, the solution would have considered $C[u, u_n, i, l, m-1]$ concatenated with the segment ll_n and found it.

7.5 Algorithm for finding ϵ-approximate smallest avatar convex hull

We calculate an ϵ-approximate minimum convex hull for a set of k-avatar entities L in R^d by finding an ϵ-kernel Q of L. We call it ϵ-approximate because the Hausdorff distance between $\mathcal{CH}^*(L)$ and $\mathcal{CH}(Q)$ would be at most $O(\epsilon)$. The approximation is achieved by computing a convex region whose width along every direction is within a factor of at least $1-\epsilon$ of the width of the optimal hull along that direction. See Algorithm 1.

It is clear that for any box B_i, with largest side of length $\mathcal{D}_i$, if we expand the box along each direction until it becomes a hypercube of side $\mathcal{D}_i$ and scale it to the unit hypercube $\mathbb{C}$, we are left with an α-fat point set in $\mathbb{C}$ since $\mathcal{CH}(B_i)$ must cover all the points in S_i and hence it must touch each face of $\mathbb{C}$. This transformation $T(B_i)$ of B_i can be found in time

Algorithm 3 ϵ-APPROXIMATION OF α-FAT AVATAR POINT SET MINIMUM AVATAR CONVEX HULL

Require: P: an α-fat set of k-avatar points inside the unit hypercube $\mathbb{C}$ where there is at least a 1-avatar point on each face of $\mathbb{C}$

```
let Z be a d-dimensional grid of cell size δ where δ is the largest integer s.t. δ ≤ (ε/√d)α
for each assignment of binary values (0/1) to the cells in the grid do
    let Q_i be the set of cells assigned with a 1 in the i^th binary assignment
    if Q_i is legal then
        let Q'_i ⊆ Q_i be the collection of highest and lowest cells in every hypercolumn containing at least one cell of Q_i
        let Q (resp., Q') be the set of representative points of cells in Q_i (resp., Q'_i)
        //It is easy to see that Q' is an ε-kernel of Q
        let CH_i = CH(Q')
        if μ(CH_i) < μ(CH_min) then
            CH_min = CH_i
        end if
    end if
end for
return CH_min
```

linear in the number of points in B_i, which is equal to $O(nk)$. By Lemma 1, we know that finding an ϵ-approximate avatar convex hull of all the points in $\mathbb{C}$ gives us directly an ϵ-approximate avatar convex hull of all the points in $\mathcal{B}_i$.

We can determine if a given set of grid cells Q_i is legal by solving a network flow problem as follows. Create a set of vertices T such that each vertex in T represents a different cell in Q_i. Create a source vertex s with directed edges to each vertex in T. Create a set of vertices T' such that each vertex in T' represents a distinct point in some cell in Q_i. Add an edge from $u \in T$ to $u' \in T'$ if the corresponding cell in Q_i contains the corresponding point. Create another set of vertices T'' such that each vertex in T'' corresponds to an avatar entity. Add an edge from $u' \in T'$ to $u'' \in T''$ if the corresponding point is a possible value for the corresponding avatar entity. Finally add a sink vertex t and connect all vertices in T'' to t by an edge. All edges have capacity 1. A maximum flow of size $|T|$ from s to t will identify a representative point in each cell such that no two points are avatars of the same entity. It is easy to see that such a flow exists if and only if the corresponding set of cells Q_i is legal.

7.6 $(1+\epsilon)$-Approximate Avatar Diameter

Definition 2 *Define the minimum* avatar *diameter $diam(L)$ of a set L of avatar points as the diameter of the* avatar assignment *$A(L)$ with the smallest diameter.*

We can use the algorithm described above to find an *avatar* ϵ'-kernel Q and ϵ'-approximate smallest convex hull $\mathcal{CH}(Q)$ of L. Our measure function is $\mu(.) = diam(.)$. We have that this measure function $diam(.)$ has the property: $diam(L) = diam(\mathcal{CH}^*(L))$. Let $\bar{u} \in S^{d-1}$ be the direction of $diam(L)$. Then we have that:

$$\begin{aligned}(1-\epsilon')\omega(u,L) &\leq \omega(u,Q), \quad \forall u \in S^{d-1}\\ (1-\epsilon')\cdot diam(L) &\leq \omega(\bar{u},Q)\\ &\leq diam(Q)\\ diam(L) &\leq (1+\epsilon)\cdot diam(Q),\\ &\qquad \text{where } \epsilon = \frac{1}{1-\epsilon'}\end{aligned}$$

Thus we can just return $(1+\epsilon)\cdot diam(Q)$, which gives us an $(1+\epsilon)$-approximate minimum avatar diameter knowing that:

$$\begin{aligned}diam(Q) \leq diam(L) &\leq (1+\epsilon)\cdot diam(Q)\\ 1 \leq \frac{(1+\epsilon)\cdot diam(Q)}{diam(L)} &\leq (1+\epsilon)\end{aligned}$$

Theorem 7 *Given an exact algorithm for finding the diameter of a convex hull that runs in time $O(n^a)$, there exists an algorithm that computes a $(1+\epsilon)$-approximate smallest k-avatar diameter in time $O((nk)^{(2d+3)}\cdot \frac{n}{\delta^d}\cdot(2d)^2\cdot 2^{\frac{1}{\delta^d}(\frac{2}{\delta^{d-1}})^{\lfloor\frac{d}{2}\rfloor}+(\frac{2}{\delta^{d-1}})^a})$.*

7.7 $(1+\epsilon)$-Approximate Smallest Perimeter Axis-Aligned Enclosing Hyperbox

For instance we would find $(1+\epsilon)$-approximate smallest perimeter axis-aligned avatar enclosing hyperbox $B(L)$ containing an avatar of each entity in L.

Let $\mathcal{CH}(L)$ be the smallest avatar convex hull of a set L of k-avatar points. Then if Q is a k-avatar ϵ'-kernel of L such that $Q \subset \mathcal{CH}(L)$. Let $\mu(.)$ be the measure function measuring the perimeter of a hyperbox. Then we have:

$$\begin{aligned}(1-\epsilon')\cdot\omega(u,L) &\leq \omega(u,Q), \quad \forall u \in S^{d-1}\\ (1-\epsilon')\cdot\omega(u,L) &\leq \omega(u,Q), \quad \forall u \in [d] = \{e_1, e_2, \ldots, e_d\}\\ (1-\epsilon')\sum_{u\in[d]}\omega(u,L) &\leq \sum_{u\in[d]}\omega(u,Q)\\ (1-\epsilon')\cdot\mu(B(L)) &\leq \mu(B(Q))\\ \mu(B(L)) &\leq (1+\epsilon)\cdot\mu(B(Q)), \quad \text{where } \epsilon = \frac{1}{1-\epsilon'}\end{aligned}$$

Thus once again we obtain a $(1+\epsilon)$-approximation by returning $(1+\epsilon)\cdot\mu(B(Q))$. knowing that:

$$1 \leq \frac{(1+\epsilon)\cdot\mu(B(Q))}{\mu(B(L))} \leq (1+\epsilon)$$

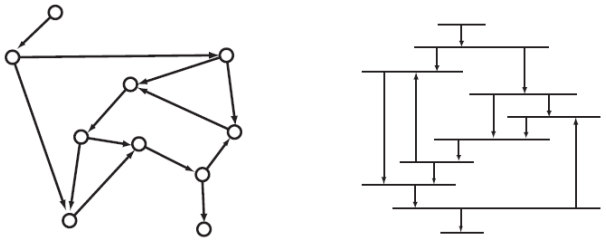
Theorem 8 *Given an exact algorithm for finding the diameter of a smallest perimeter axis-aligned enclosing hyperbox that runs in time $O(n^a)$, there is an algorithm that finds a $(1+\epsilon)$-approximate smallest perimeter axis-aligned avatar enclosing hyperbox in time $O((nk)^{(2d+3)}\cdot\frac{n}{\delta^d}\cdot(2d)^2\cdot 2^{\frac{1}{\delta^d}(\frac{2}{\delta^{d-1}})^{\lfloor\frac{d}{2}\rfloor}+(\frac{2}{\delta^{d-1}})^a})$.*

7.8 Rectilinear planar layout of Cubic Directed Planar Graph

A graph is a cubic graph if all vertices have degree three. The following problem is known to be NP-complete [11, 16]: given a cubic directed planar graph

G (in which we can assume that each vertex has in-degree two and out-degree one, or in-degree one and out-degree two), determine whether there exists a directed Hamiltonian path in G?

A rectilinear planar layout of a planar graph $G = (V, E)$ is a mapping such that each vertex in V is mapped to a horizontal line segment and each edge in E is mapped to a vertical line segment. Two horizontal line segments are connected by the vertical line segment if and only if their corresponding vertices in G are connected with an edge of G (see Fig. 6). Rosenstiehl and Tarjan [17] showed that a rectilinear layout of G can be computed in $O(n)$ time such that all endpoints of segments have integer coordinates and the height and the width of the layout are both $O(n)$.



(a) (b)

Figure 6: *Rectilinear planar layout.*

By introducing bending edges, we can redraw the rectilinear planar layout satisfying the following property:

(i) For each horizontal line corresponding to $v \in V$ with in-degree 1 and out-degree 2 (resp. in-degree 2 and out-degree 1), the entering (leaving, reps.) vertical segment lies between the two leaving (entering) vertical segments as shown in Fig. 7(a).

For each $v \in V$, let us denote the intersection points of the horizontal line corresponding to v and the three incident vertical lines from the left to right by $l(v)$, $c(v)$ and $r(v)$ (see Fig. 4). Then, by bending vertical edges, we further impose the following properties for the rectilinear drawing:

(i) For each $v \in V$, $|l(v) - c(v)| = |r(v) - c(v)|$. (Fig. 7(b))

(i) For each horizontal line corresponding to $v \in V$ with out-degree 2 in G, the leaving horizontal lines have the same length. (Fig. 7(c))

If the parity of the y-coordinates of all horizontal lines coincide, then the above redrawing is always possible. (We can adjust the y-coordinate of each horizontal line at the beginning. So, this redrawing process is always possible.) Also, by making the grid finer, we may assume that all segments are drawn in the grid and parallel lines are at least 4 unit-length apart from each other.

7.9 Reduction from DHP$_{st}$ to 2-Avatar EMST

Here we show a polynomial-time reduction from DHP$_{st}$ to 2-Avatar EMST. Let n be the sum of the number of pairs and the number of unpaired points. In this setting, we observe the following.

Theorem 9 *The avatar MST problem constructed in section 4 has the optimal cost of $(n-1)$ if and only if G has a Hamiltonian path from s to t.*

Proof *Since any two points have a distance of at least 1 unit-length, any avatar MST hast the cost at least $n-1$ unit-length. It is clear that there is an avatar MST with the cost $n-1$ if G has a Hamiltonian path. Observe that the distance of two points of P is equal to 1 iff they are next to each other on a line segment in the drawing. Hence, if we consider the graph on P consisting of unit length edges, then only $c(v)$ $(v \in V)$ can have the degree three. However, since $c_{-1}(v)$ and $c_1(v)$ are paired, any avatar forest consisting of only unit-length edges has no vertex with degree more than two. So, if G has no Hamilton path, then any avatar forest consisting of only unit-length edges is disconnected, and any avatar MST contains an edge whose length is more than unit-length.* ■

7.10 NP-Completeness Proof of Avatar Reachability

Theorem 10 *The k-avatar reachability problem is NP-Complete.*

Proof *The reduction is from the CLIQUE problem. Let graph $G_C(V, E)$ and integer k denote an instance of the CLIQUE problem. We construct graph $G_A(V', E')$ as follows: create $k+2$ layers of vertex sets, $V'_0, V'_1, ..., V'_{k+1}$. Let $V'_0 = \{s\}$ and $V'_{k+1} = \{t\}$. For $i = 1, \ldots, k$, let $V'_l = \{v'_{l,i,j} : 1 \leq i \leq |V|, 1 \leq j \leq k\}$. Let vertices $v'_{x,i,y}$ and $v'_{y,i,x}$ be avatars of the same entity. Add edges $(v'_{l,i,j}, v'_{l,i,j+1})$ for all l, i, and j; for $0 \leq l < k$, add edges $(v'_{l,i,k}, v'_{l+1,j,1})$ for all i, j. Note*

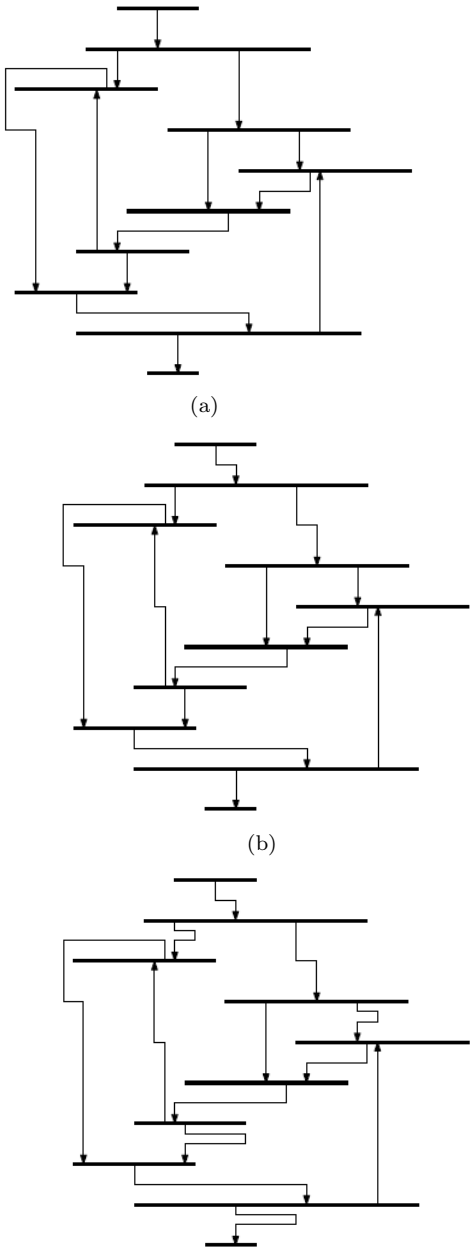


Figure 7:

that the vertices $v'_{l,i,1}, ..., v'_{l,i,k}$ in layer l form n connected subpaths. Denote the subpath from $v'_{l,i,1}$ to $v'_{l,i,k}$ by $S_{l,i}$. It is important to note that for each vertex $v_i \in V$, there is exactly one corresponding subpath in each layer V'_l. See figure 8 in the appendix (see section 7.12).

Now for each pair of non-adjacent vertices $v_i, v_j \in V$ from G_C, add vertices $u'_{l,i,x}$ and $u'_{l,j,x}, 1 \leq x \leq k$ to G_A. Add edges $(u'_{l,i,x}, u'_{l,i,x+1})$ and $(u'_{l,j,x}, u'_{l,j,x+1})$, $1 \leq x < k$. Update subpaths $S_{l,i}$ by adding an edge $(v_{last}, u'_{l,i,1})$ to its last vertex; and take each outgoing edge from it and make it outgoing from $u'_{l,i,k}$ instead. Finally, let each pair of vertices $u'_{x,i,y}, u'_{y,j,x}$ be avatars of each other, for all $1 \leq x, y \leq k$.

It is not hard to see that the size of graph G_A is polynomial in n. More importantly, we claim that if the set $\{v_{i_1}, ..., v_{i_k}\}$ is a clique in G_C, then a 2-avatar path can be found from s to t in G_A by starting at s (layer V'_0), moving from each layer to the next, selecting in each layer a subpath corresponding to a distinct vertex from the clique. Intuitively, if the path in level l goes through vertices of the form $v'_{l,i,j}$, then vertex i is chosen as the l-th vertex in the clique. Furthermore, the vertices of the form $u'_{l,i,j}$ that are required to be visited by the path (and its avatars) ensure that the other vertices picked for the clique are indeed adjacent to i. The converse is proved by starting from a 2-avatar path and selecting the clique vertices based on the subpaths traversed in each layer.

Hence there is a clique of size k in G_C if and only if there is a 2-avatar $s \rightsquigarrow t$ path in G_A. It is readily shown that 2-avatar reachability is in NP, thus completing the proof that it is NP-Complete. ■

7.11 Gap-preserving reduction from max-clique to 2-avatar shortest path

Lemma 2 *There is a gap-preserving reduction from max-clique to 2-avatar shortest path that transforms a graph $G_{clique}(V, E)$ to a graph $G_{avatar}(V', E')$ such that:*

(i) if $OPT_{clique}(G) \geq k \cdot |V|$, $OPT_{avatar}(s—t) \leq m$

(ii) if $OPT_{clique}(G) < \alpha \cdot (k \cdot |V|)$, $OPT_{avatar}(s - t) > (2 - \alpha) \cdot m$

where $m = (k^2 \cdot |V|^3) + 1$, and α and k are any constants such that $0 \leq \alpha, k \leq 1$. Here $OPT_{clique}(G)$ is the size of the maximum clique in $G_{clique}(V, E)$, and $OPT_{avatar}(s-t)$ is the length of the shortest 2-avatar path from a vertex s to a vertex t in $G_{avatar}(V', E')$.

Proof *Let $G_{clique}(V, E)$ represent an arbitrary instance of a max-clique problem, and $0 < k \leq 1$. We first create a graph $G_{avatar}(V', E')$ as follows:*

Assume that $k \cdot |V|$ is an integer, if it is not then just round it to an integer. Create $k \cdot |V| + 2$ layers $V'_0, V'_1, ..., V'_{k\cdot|V|+1}$. Add a new vertex s in layer V'_0. Add a new vertex t in layer $V'_{k\cdot|V|+1}$. In each layer $V'_l \in \{V'_1, ..., V'_{k\cdot|V|}\}$, $1 \leq l \leq k \cdot |V|$, add $k \cdot |V|^2$ new vertices, $V'_l = \{v'_{l,i,j} : 1 \leq i \leq |V|, 1 \leq j \leq k\cdot|V|\}$. Then make each pair of vertices $(v'_{x,i,y}, v'_{y,i,x})$ be avatars of each other, and add an edge from $v'_{l,i,j}$ to $v'_{l,i,j+1}$ in each layer V'_l for all vertices $v'_{l,i,j}$ for $1 \leq l \leq k \cdot |V|$.

Note that in each layer V'_l, there exists a connected subpath from $v'_{l,i,1}$ to $v'_{l,i,k\cdot|V|}$, for each $1 \leq i \leq |V|$. In other words, each level consists of $|V|$ subpaths each of length $k \cdot |V|$ with no links between the subpaths. Denote the subpath of vertices $\{v'_{l,i,1}, ..., v'_{l,i,k\cdot|V|}\}$ as $S_{l,i}$. It represents the subpath corresponding to vertex $v_i \in V$ in layer V'_l. Notice that for each vertex $v_i \in V$ there is one, and only one, subpath in each layer V'_l that corresponds to it.

Now append more vertices to each subpath in the following way: For each pair of non-adjacent vertices $v_i, v_j \in V$ from G_{clique}, add vertices $u'_{l,i,j,x}$ to $S_{l,i}$ and $u'_{l,j,i,x}$ to $S_{l,j}$ for all x, $1 \leq x \leq k \cdot |V|$ in each layer V'_l in the graph G_{avatar}. Add an edge from $u'_{l,i,j,x}$ to $u'_{l,i,j,x+1}$ and an edge from $u'_{l,j,i,x}$ to $u'_{l,j,i,x+1}$ for all $1 \leq x < k \cdot |V|$. Then update the subpath $S_{l,i}$ corresponding to vertex $v_i \in V$ in layer V'_l by adding an edge from its last vertex to $u'_{l,i,j,1}$. Update subpaths $S_{l,j}$ accordingly for all layers too. Then for all the new vertices added in all layers, make each pair of vertices $u'_{x,i,j,y}$,$u'_{y,j,i,x}$ be avatars of each other, for all $1 \leq x, y \leq k \cdot |V|$.

Now, for each layer V'_l add two slack subpaths C'_l and D'_l by adding new vertices $C'_l = \{c'_{l,1}, \ldots, c'_{l,k\cdot|V|^2+\frac{m}{k\cdot|V|}}\}$ and $D'_l = \{d'_{l,1}, d'_{l,2}, ..., d'_{l,k\cdot|V|^2+\frac{m}{k\cdot|V|}}\}$. Make each pair of vertices $c'_{l,x}$,$d'_{l,x}$ be avatars of each other, for all $1 \leq x \leq k \cdot |V|^2 + \frac{m}{k\cdot|V|}$. Also add an edge from $c'_{l,x}$ to $c'_{l,x+1}$ and an edge from $d'_{l,x}$ to $d'_{l,x+1}$ for all $1 \leq x < k \cdot |V|^2 + \frac{m}{k\cdot|V|}$. Add an edge from $c'_{l,k\cdot|V|^2+\frac{m}{k\cdot|V|}}$ to the first vertex of each subpath in layer V'_{l+1}, and do the same for $d'_{l,k\cdot|V|^2+\frac{m}{k\cdot|V|}}$. Then in each layer V'_l, add a shortcut edge from the last vertex of subpath $S_{l,i}$ to every vertex that receives an edge incoming from $c'_{l,\frac{m}{k\cdot|V|}+|S_{l,i}|}$. The effect of adding these shortcut edges is to ensure that the length of any maximal in level l is $k\cdot|V|^2$ or $k\cdot|V|^2+\frac{m}{k\cdot|V|}$; the length is $k\cdot|V|^2+\frac{m}{k\cdot|V|}$ if the path utilizes only vertices from C'_l or D'_l, and $k\cdot|V|^2$ if the path uses vertices from any subpath $S_{l,i}$. Add an edge from $c'_{l,k\cdot|V|^2+\frac{m}{k\cdot|V|}}$ to $c'_{l+1,1}$, and an edge from $d'_{l,k\cdot|V|^2+\frac{m}{k\cdot|V|}}$ to $d'_{l+1,1}$. Finally add edges from s to $c'_{1,1}$, to $d'_{1,1}$, and to the first vertex of $S_{1,i}$ for $0 \leq i \leq |V|$; from $c'_{k\cdot|V|,k\cdot|V|^2+\frac{m}{k\cdot|V|}}$ to t; and from $d'_{k\cdot|V|,k\cdot|V|^2+\frac{m}{k\cdot|V|}}$ to t.

Note that the size of any subpath $S_{l,i}$ is at most $k\cdot|V|+k\cdot|V|\cdot(|V|-1)$, which is $k\cdot|V|^2$. Observe that the shortest path from the first vertex of any subpath $S_{l,i}$ to the first vertex of any subpath $S_{l+1,j}$ in the following layer is of length $k \cdot |V|^2$ by construction.

Suppose there is a clique of size $(k \cdot |V|)$ in graph G_{clique}. Let $C_{k\cdot|V|} = \{v_{i_1}, v_{i_2}, ..., v_{i_{(k\cdot|V|)}}\}$ be the vertices in the clique. Then it is possible to form a 2-avatar path p from s to t by starting at s and adding subpath S_{l,i_l} to the path p for each layer V'_l where $1 \leq l \leq k\cdot|V|$ (see that subpath S_{l,i_l} is the subpath corresponding to vertex $v_{i_l} \in C_{k\cdot|V|}$ in layer V_l). Then path p is a 2-avatar path since it cannot have any pair of vertices that are avatars of each other. Path p cannot have any pair of vertices that are avatars of each other because if there were any such pair then by construction it would mean that there is a pair of non-adjacent vertices in the given clique $C_{k\cdot|V|}$, which is a contradiction. This path p is of length $m = k^2 \cdot |V|^3 + 1$ since it moves from layer to layer by selecting subpaths corresponding to vertices in the clique from graph G_{clique}. The distance from the first vertex of a non-slack subpath to the first vertex of a subpath in the next layer is $k \cdot |V|^2$ for all layers V'_l with $1 \leq l \leq k \cdot |V|$. Since $k \cdot |V|$ non-slack subpaths are selected by path p going from s to t then p has length $m = k^2 \cdot |V|^3 + 1$. This path is also a 2-avatar path of shortest length connecting s to t since it moves from layer to layer by selecting the subpaths corresponding to the vertices in the clique.

If the max-clique is of size larger than or equal to $k \cdot |V|$ then the shortest 2-avatar path is of length exactly m after seeing that it is possible to find a 2-avatar path of size m from s to t in G_{avatar} given a

clique of size $(k \cdot |V|)$ *in* G_{clique}*. Then* (i) *follows.*

If the max-clique in G_{clique} *is of size smaller than* $\alpha \cdot (k \cdot |V|)$*, for some constant* $\alpha \leq 1$*, meaning:*

$$OPT_{clique}(G) < \alpha \cdot (k \cdot |V|),$$

Then any path from s *to* t *must use only vertices from the slack subpaths* C'_l *or* D'_l *in at least one layer* V_l*. If it were possible to gather* $(k \cdot |V|)$ *non-slack subpaths (one per layer) that make a 2-avatar path* p *from* s *to* t *then it would be possible to find a clique of size* $(k \cdot |V|)$ *in* G_{clique} *by selecting the vertices in* G_{clique} *that correspond to the subpaths used in* p*. Then, if the max-clique is of size less than* $\alpha \cdot (k \cdot |V|)$*, there is no 2-avatar path from* s *to* t *that uses more than* $\alpha \cdot (k \cdot |V|)$ *of the subpaths* $S_{l,i}$*. In that case at least* $(1-\alpha) \cdot (k \cdot |V|)$ *slack subpaths must be used in order to form a 2-avatar path from* s *to* t*. So the size of the shortest path must be larger than* $(2-\alpha) \cdot m$*. See that:*

$$\begin{aligned}
OPT_{avatar}(G) &> ((k \cdot |V|) \cdot (k \cdot |V|^2) + 1) \\
&\quad + \frac{m}{k \cdot |V|} \cdot (1-\alpha) \cdot (k \cdot |V|) \\
OPT_{avatar}(G) &> (k^2 \cdot |V|^3 + 1) + m \cdot (1-\alpha) \\
OPT_{avatar}(G) &> m + m \cdot (1-\alpha) \\
OPT_{avatar}(G) &> (2-\alpha) \cdot m
\end{aligned}$$

and (ii) *follows.*

7.11.1 Inapproximability of 2-avatar shortest path to within any constant factor

The 2-avatar shortest path problem cannot be approximated to within any constant factor in polynomial time because any polynomial-time constant-factor approximation algorithm for 2-avatar shortest path problem can be used to provide an exact polynomial time solution to the decision version of the clique problem. A polynomial-time solution to clique would imply that P=NP, hence a constant-factor polynomial-time approximation algorithm for 2-avatar shortest path cannot exist unless P=NP. We will show a gap-introducing reduction from clique to 2-avatar shortest path using a construction as the one shown above in section 7.11.

Theorem 11 *There is no polynomial-time* c-approximation *algorithm for 2-avatar shortest path problem for any constant* $c > 1$*.*

Proof *Proof by contradiction. Let* ALG_{avatar} *be a* c*-approximation algorithm for 2-avatar shortest path. We show that if there is such an approximation algorithm for 2-avatar shortest path then we can solve clique in polynomial time by introducing a gap. To show this, we describe how an instance* $I_{clique}(G, \kappa)$ *of clique can be transformed into an instance* I_{avatar} *of 2-avatar shortest path such that approximating* I_{avatar} *to within a constant factor* c *of its optimal solution gives us a solution to* $I_{clique}(G, \kappa)$*. We use a transformation almost identical to the transformation used in the proof of theorem 2. Let an instance* $I_{clique}(G, \kappa)$ *of clique be defined with a graph* $G_{clique} = (V, E)$ *as: Is there a clique of size at least* κ *in* G*? As in section 7.11, we create an instance* I_{avatar} *by creating a graph* $G_{avatar} = (V', E')$ *exactly as shown in section 7.11, letting* $k = \frac{\kappa}{|V|}$*. Here we are going to have the slack subpaths have length*

$$k \cdot |V|^2 + (c-1) \cdot m + 1$$

instead of $k \cdot |V|^2 + \frac{m}{k \cdot |V|}$ *as we did in section 7.11. This will give us the following relation:*

- *(i) if* $OPT_{clique}(G) \geq k \cdot |V|$*,* $OPT_{avatar}(s—t) = m$
- *(ii) if* $OPT_{clique}(G) < k \cdot |V|$*,* $OPT_{avatar}(s-t) > c \cdot m$

If $OPT_{clique}(G) \geq k \cdot |V|$ *that means that there is a clique of size at least* κ *in* G*. If* $OPT_{clique}(G) < k \cdot |V|$ *then there is no clique of size at least* κ *in* G*.*

In this construction of graph G_{avatar} *we have that in each layer, for each subpath that goes through vertices in non-slack subpaths* $S_{l,i}$*, for* $1 \leq l \leq k \cdot |V|$ *and* $1 \leq i \leq |V|$*, its length is going to be* $k \cdot |V|^2$ *as in the construction in section 7.11; and for each subpath that goes only through vertices in slack subpaths, its length is going to be* $k \cdot |V|^2 + (c-1) \cdot m + 1$*. Notice that the shortest possible path from* s *to* t *has length*

$$m \quad = \quad k^2 \cdot |V|^3 + 1$$

This path could be obtained by taking vertices from a subpath $S_{l,i}$ *in each layer* V_l*, for* $1 \leq l \leq k \cdot |V|$*. Any path from* s *to* t *of length larger than* m *must take only vertices from a slack subpath in at least one layer. In that case the length of the path must be at*

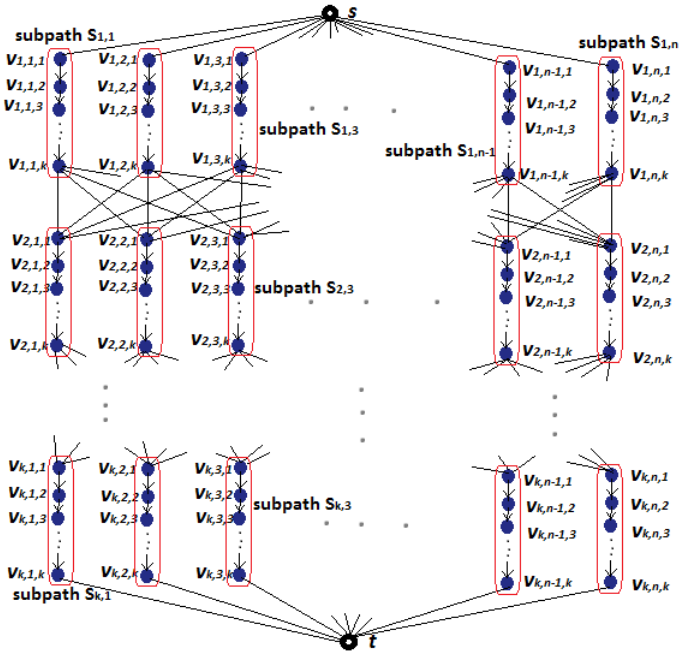
least $c \cdot m + 1$. If the path takes vertices from a slack subpath only in at least one layer then the length of the path would be:

$$\begin{aligned} OPT_{avatar}(s-t) &\geq ((k \cdot |V|) \cdot (k \cdot |V|^2) + 1) \\ &\quad +(c-1) \cdot m + 1 \\ OPT_{avatar}(s-t) &\geq m + c \cdot m - m + 1 \\ OPT_{avatar}(s-t) &\geq c \cdot m + 1 \\ OPT_{avatar}(s-t) &> c \cdot m \end{aligned}$$

So any algorithm ALG_{avatar} that approximates 2-avatar shortest path to within a factor of c will have to produce a path of length m. Since a path p of length m must go through non-slack subpaths in each layer then we can gather $k \cdot |V|$ vertices from the original graph G by taking the vertices corresponding to the non-slack subpaths used by p, this will give us κ vertices that form a clique in G. If ALG_{avatar} produces a path of length larger than $c \cdot m$ then we know that there is no clique of size at least $\kappa = k \cdot |V|$ in G.

Therefore if there were a c-approximation polynomial time algorithm for 2-avatar shortest path then we could decide if there is a clique of size at least κ in a given graph G by using the gap-introducing reduction shown above. This would imply that P=NP. Hence, there is no polynomial-time c-approximation algorithm for 2-avatar shortest path for any constant c unless P=NP.

7.12 Figures



(a)

Figure 8: *Graph G_V after the first set of constructions.*