Geometric Avatar Problems

Mario E. Consuegra and Giri Narasimhan

1 School of Computing and Information Sciences, Florida International University, Miami, FL 33199, USA. \{mcons004,giri\}@fiu.edu

Every man has the power to choose, but no power to escape the necessity of choice. – Ayn Rand

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 exactly one of the avatars. The resulting set of problems show a surprising range of hardness characteristics and elicit a variety of algorithmic solutions. Many avatar problems are considered. In particular, we show how to extend the concept of \( \epsilon \)-kernels to find approximation algorithms for geometric avatar problems. Results for metric space graph avatar problems are also presented.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Avatar problems, choice

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 manifest themselves in many practical applications. For example, if disk storage systems have multiple copies of data items, then disk scheduling algorithms may process requests by visiting any one of the copies of each requested data to optimize the total access cost. 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. In this paper we investigate the complexity of avatar versions of classical algorithmic problems.

Avatar versions of NP-hard problems are easily shown to be NP-hard. However, designing good approximation algorithms often requires the solution of other avatar problems (e.g., avatar TSP can be well approximated by approximating avatar MST or avatar matching). This suggests the need for solving a family of avatar problems, over and beyond those with direct relevance to practical applications.

Related problems include generalized MST (MST spanning at least one vertex from each given set) [20], group TSP (minimum length tour visiting at least one vertex from each given group) [9], and TSP with neighborhoods (minimum length tour that visits each given neighborhood) [8,22], all of which are NP-hard, even in Euclidean space. A closely related model is the indecisive (uncertain) points model [16], where input points have spatial uncertainty, but their true locations are known to be from a set of (possibly infinite) possibilities. Jørgensen et al. [16] suggest many applications for their model; these are applicable to the avatar model as well. For example, privacy considerations may prevent a database from storing the precise location of a person with a certain illness, but may provide a zip code; sensors may have limited accuracy and may provide approximate location data. A major difference with the avatar model is that in the uncertainty model, the power of choice lies with an adversary. Also see [5,10–12,14,19,21].
The concept of switching graphs [15, 18] introduces another related model. In the area of scheduling, a related problem is the job interval selection problem (JISP) [7]. 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 \( \infty \) provides better understanding of the complexity landscape of the problem.

**Results**

The main results are summarized here, and is indicative of how “choice” affects the complexity of these problems in different ways.

1. In Section 2, we tackle two seemingly simple problems, \( \text{minGap} \) and \( \text{maxGap} \), where the avatar versions result in natural optimization problems. We design a \( \mathcal{O}(n^2 \log n) \)-time algorithm for the 2-avatar maximum \( \text{minGap} \) problem for inputs in \( \mathbb{R}^d \), and a 2-approximation \( \mathcal{O}(n^3 \cdot k^3 \log(nk)) \)-time algorithm for the \( k \)-avatar minimum \( \text{maxGap} \) problem for points on a line.

2. In Section 3, we extend the concept of \( \epsilon \)-kernels to the avatar world and show how to compute it efficiently for a \( k \)-avatar point set in \( \mathbb{R}^d \). This enabled us to design a polynomial-time algorithm for finding an \( \epsilon \)-approximate smallest convex hull for the \( k \)-avatar convex hull problem in \( \mathbb{R}^d \). The \( \epsilon \)-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.

3. Reachability is a fundamental problem with linear-time algorithms for non-avatar inputs. Surprisingly, we show in Section 4 that for unweighted graphs, the \( k \)-avatar reachability problem is NP-complete. For weighted graphs, we show that the \( k \)-avatar shortest path problem is inapproximable to any constant factor unless \( P = NP \).

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 \( \text{Av}(a_i) = \{v^{(1)}_i, v^{(2)}_i, \ldots, v^{(k)}_i\} \). 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, \( A(a_i) \in \text{Av}(a_i) \). 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 \( \text{minGap} \) (resp. \( \text{maxGap} \)) as the smallest (resp. largest) gap between consecutive items in the sorted order. The avatar version of the maximum \( \text{minGap} \) and minimum \( \text{maxGap} \) problems is: Given a set of \( n \) \( k \)-avatar entities, find an avatar assignment that results in the maximum \( \text{minGap} \) (resp. minimum \( \text{maxGap} \)). More formally, 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 of \( k \) values from the set \( \{v^{(1)}_i, v^{(2)}_i, \ldots, v^{(k)}_i\} \). We first solve the decision problem of determining if there exists an avatar assignment so that the \( \text{minGap} \) 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 \([3]\), it is not difficult to see that the above algorithm takes \( O(n^2) \) time, and that the maximum minGap can be found in \( O(n^2 \log n) \) 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. \([13]\).

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

### 2.2 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 \( B_{\text{low}} \) and \( B_{\text{upp}} \) for \( B^* \) such that \( B_{\text{low}} \leq B^* \leq B_{\text{upp}} \). 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 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, \( v_2^7 \) and \( v_2^5 \) 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 partition $Q$ has vertices corresponding to cells of the partition. There is an edge from 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 omitted from this draft. 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

1. $B_{i-1} \leq B_i$,
2. the algorithm returns NO for all partitions into cells of length $B_{i-1}$, and
3. 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 avatar assignment where all the chosen points are in a set of adjacent cells such that each cell in that set contains a chosen point, we can use that avatar assignment to produce an assignment with a maximum gap that is 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.

### 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. A $k$-avatar convex hull of $L$ is a minimal convex set that contains at least one avatar for each entity $a \in L$. The aim is to minimize 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 [17,23], and a recent paper by Abdullah et al. [1] for the model of uncertain points.

A smallest avatar convex hull is a convex hull that has minimum perimeter over all possible avatar assignments. Using the concept of $\epsilon$-kernels, we present an algorithm that finds an $\epsilon$-approximate smallest avatar convex hull for the $k$-avatar convex hull problem in $\mathbb{R}^d$. The results can be extended to minimum area/volume convex hulls.
3.1 Approximate Avatar Convex Hulls

For any point set \( X \subseteq \mathbb{R}^d \), 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 \( \epsilon \)-approximation of \( P \) if for any direction \( u \in S^d \) we have \( (1 - \epsilon)\omega(u, P) \leq \omega(u, Q) \leq (1 + \epsilon)\omega(u, P) \). Our proposed algorithm finds an \( \epsilon \)-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 \((1 - \epsilon)\omega(u, \mathcal{CH}^*(L)) \leq \omega(u, \mathcal{CH}(Q)) \leq (1 + \epsilon)\omega(u, \mathcal{CH}^*(L)) \), where \( \mathcal{CH}^*(L) \) is the minimum avatar convex hull of \( L \). Using the terminology of Agarwal et al. [2], one can think of the set \( Q \) as the avatar equivalent of an \( \epsilon \)-kernel. This is formalized in the following definition of an avatar \( \epsilon \)-kernel whose width along any direction is within a \( 1 - \epsilon \) factor of the width of the optimal hull along that direction.

**Definition 3.** Given a set \( L \) of \( n \) \( k \)-avatar entities in \( \mathbb{R}^d \), we say that a point set \( Q \) is an avatar \( \epsilon \)-kernel of \( L \) if and only if \((1 - \epsilon)\omega(u, \mathcal{CH}^*(L)) \leq \omega(u, Q) \leq (1 + \epsilon)\omega(u, \mathcal{CH}^*(L)), \forall u \in S^d \), where \( S^d \) is the unit hypersphere centered at the origin.

The following procedure for finding a diameter-oriented bounding box \( B \) of a set \( S \) of points in \( \mathbb{R}^d \) was described by Barequet and Har-Peled [4]. Let \( \mathcal{D}(S) \) 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 \( B(S) \) of \( S \) with the diameter in each iteration determined by the direction from \( s_i \) to \( t_i \), for \( i = 1, 2, \ldots, d - 1 \).

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 \( B(S) \) is exactly the same as the diameter-oriented bounding box \( B(\mathcal{CH}^*) \). 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 \( \binom{n}{2d} \). For each such subset \( S_i \), we first check that no two points in \( S_i \) are in the same avatar set, then we find the diameter-oriented bounding box \( B_i = B(S_i) \). If every entity in \( L \) has an avatar point inside \( B_i \) then it is possible that \( B_i = B(\mathcal{CH}^*) \), otherwise we can discard \( B_i \). We find an \( \epsilon \)-approximate minimum avatar convex hull \( \mathcal{CH}_i \) of all the points inside \( B_i \) and output the smallest one, which we refer to as \( \mathcal{CH}_{\text{min}} \). Since \( B(\mathcal{CH}^*) = B_i \) for some \( i \), \( \mathcal{CH}_{\text{min}} \) will \( \epsilon \)-approximate \( \mathcal{CH}^* \). The following lemma from [2] is useful for this proof. We say that a
point set is \( \alpha\)-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 \( \alpha < 1 \).

▶ Lemma 4. [2] 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 \( \alpha \)-fat point set (for some \( \alpha < 1 \)) where the hypercube \( C = [-1, +1]^d \) is the smallest enclosing box of \( M(P) \) and s.t. a subset \( Q \subseteq P \) is an \( \epsilon \)-kernel of \( P \) iff \( M(Q) \) is an \( \epsilon \)-kernel of \( M(P) \).

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

Algorithm 1 Computing \( \epsilon \)-approximate smallest avatar convex hull

Require: \( L \): set of \( n \) \( k \)-avatar entities; \( \mu \): a measure function of the size of the perimeter of a convex hull, \( T(\cdot) \) affine transform procedure

let \( \mathcal{C} \mathcal{H}_{min} = \text{null} \)

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 avatars of each other then

let \( B(S_i) \) be the diameter-oriented bounding box

let \( B_i \) be the set of all avatar points inside \( B(S_i) \) such that every entity is represented in \( B_i \)

let \( \mathcal{C} \mathcal{H}_i \) be the \( \epsilon \)-approximate smallest avatar convex hull of \( T(B_i) \) computed by Algorithm 2

\( \mathcal{C} \mathcal{H}_{min} = \text{Min}(\mathcal{C} \mathcal{H}_{min}, \mathcal{C} \mathcal{H}_i) \)

end if

end for

return \( \mathcal{C} \mathcal{H}_{min} \)

It is known that for a diameter-oriented bounding box \( B \) with largest side \( D \), if we appropriately expand or contract the box along each direction until it becomes a hypercube of side \( D \) and scale it to the hypercube \( C \), then the transformed point set is an \( \alpha \)-fat point set (for some \( \alpha < 1 \)) in \( C \) [4]. This is illustrated by an example in Fig. 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 4, to compute an \( \epsilon \)-approximate avatar convex hull of all the points in \( B \), we only need to compute an \( \epsilon \)-approximate avatar convex hull of all the points in \( C \), which is computed as follows.

As in [4], let \( \delta \) be the largest value such that \( \delta \leq (\epsilon / \sqrt{d}) \alpha \) and \( 1 / \delta \) is an integer. We then partition the bounding hypercube into a uniform grid with cells of side length \( \delta \) (see Fig. 2 (b)). However, applying the algorithm of Barequet and Har-Peled [4] does not help us to compute \( \epsilon \)-kernels in \( C \) because now we must make sure not to pick two or more avatars of the same entity.
We need one other idea to compute \( \epsilon \)-kernels in \( C \). The following procedure computes the \( \epsilon \)-kernel in \( T(B) \) (see Algorithm 2). 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 \( A_{OPT}(\cdot) \).

It is clear that for any box \( B_i \), with largest side of length \( D_i \), if we expand the box along each direction until it becomes a hypercube of side \( D_i \) and scale it to the unit hypercube \( C \), we are left with an \( \alpha \)-fat point set in \( C \) (for some \( \alpha < \)) since \( CH(B_i) \) must cover all the points in \( S_i \) and hence it must touch each face of \( C \). This transformation \( T(B_i) \) of \( B_i \) can be found in time linear in the number of points in \( B_i \), which is equal to \( O(nk) \). By Lemma 4, we know that finding an \( \epsilon \)-approximate avatar convex hull of all the points in \( C \) gives us directly an \( \epsilon \)-approximate avatar convex hull of all the points in \( B_i \).

![Figure 4](image-url) Reduction to network flow used to determine if a set of cells is legal.

We can determine if a given set of grid cells \( Q_i \) is legal by solving a network flow problem as follows. (See Fig. 4.) 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 cell in \( Q_i \) corresponding to \( u \) contains the point corresponding to \( u' \). 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 5.** There is an algorithm that finds an \( \epsilon \)-approximate min-perimeter \( k \)-avatar convex hull in time \( O(nk(2d+3) \cdot \frac{1}{\delta^2} \cdot (2d)^2 \cdot 2^{\frac{d}{\delta^d}}(\frac{\epsilon}{\delta^d})^{\frac{1}{3}}) \) by finding an avatar \( \epsilon \)-kernel \( Q \).
Algorithm 2 $\epsilon$-APPROXIMATION OF SMALLEST AVATAR CONVEX HULL FOR $\alpha$-FAT AVATAR POINT SET

**Require:** $P$: an $\alpha$-fat set (for some $\alpha < 1$) of $k$-avatar points inside the unit hypercube $C$

$\mu$: measure function of the size of the perimeter

let $\delta$ be the largest integer s.t. $\delta \leq (\epsilon/\sqrt{d})\alpha$

let $Z$ be a $d$-dimensional grid of cell size $\delta$

for each assignment of binary values (0/1) to the cells in the grid $Z$
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 $Q'_i \subseteq Q_i$ be the collection of highest and lowest cells in every hypercolumn containing at least one cell of $Q_i$

let $Q'$ be the set of representative points of cells in $Q'_i$

let $\mathcal{CH}_i = \mathcal{CH}(Q')$

if $\mu(\mathcal{CH}_i) < \mu(\mathcal{CH}_{\text{min}})$ then

$\mathcal{CH}_{\text{min}} = \mathcal{CH}_i$

end if

end if

end for

return $\mathcal{CH}_{\text{min}}$

of $L$, which by Definition 3 satisfies $(1 - \epsilon)\omega(u,\mathcal{CH}^*(L)) \leq \omega(u,\mathcal{CH}(Q)), \forall u \in S^{d-1}$. Note that the choice of constant $\delta$ depends on $k, \epsilon$, and $\alpha$.

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 $\epsilon$-kernel of $Q'$. We argue that $A_{\text{OPT}}(\cdot)$, the avatar assignment that leads to the optimal avatar convex hull, occupies a collection of cells (call this set of cells $Q_{\text{OPT}}$), 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 $Q_{\text{OPT}}$. Since for each point in the legal set, 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 $\epsilon$-kernel of the original input.

3.2 Approximate Smallest Volume Enclosing Hyperbox

Using $\epsilon$-kernels we prove the following theorem.

**Theorem 6.** Given an exact algorithm for finding the min-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 $O((nk)^{(2d+3)} \cdot \frac{n}{\delta^a} \cdot (2d)^2 \cdot 2^{\frac{d}{2}} (\frac{2}{\sqrt{2}}) \cdot \frac{1}{2} + (\frac{2}{\sqrt{2}})^a)$. Note that the choice of constant $\delta$ depends on $k, \epsilon$, and $\alpha$.

**Proof.** 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 $\epsilon'$-kernel of $L$, for some constant $\epsilon'$. Let $\mathcal{CH}(L)$ be the smallest avatar convex hull of a set $L$ of $k$-avatar points. If
Q is a $k$-avatar $\epsilon'$-kernel of $L$ such that $Q \subseteq CH(L)$, then we have:

$$(1 - \epsilon') \cdot \omega(u, L) \leq \omega(u, Q), \quad \forall u \in S^{d-1}$$

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

$$(1 - \epsilon')^d \prod_{u \in [d]} \omega(u, L) \leq \prod_{u \in [d]} \omega(u, Q)$$

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

$$(1 - c\epsilon') \prod_{u \in [d]} \omega(u, L) \leq \prod_{u \in [d]} \omega(u, Q)$$

$$(1 - c\epsilon') \cdot Volume(B(L)) \leq Volume(B(Q))$$

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

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

Similar results can be achieved for an approximate min-diameter (see Section 3.3) and min-perimeter axis-aligned avatar enclosing box.

### 3.3 $(1 + \epsilon)$-Approximate Avatar Diameter

This section gives yet another result using $\epsilon$-kernels.

**Definition 7.** Define the minimum avatar diameter $\text{diam}(L)$ of a set $L$ of avatar points as the diameter of the avatar assignment $A(L)$ with the smallest diameter.

**Theorem 8.** 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 $O((nk)^{(2d+3)} \cdot \frac{n}{\epsilon^d} \cdot (2d)^2 \cdot 2^{\frac{d}{\epsilon^2}} (\frac{2}{\epsilon^2})^{\frac{d}{2}} + \frac{2}{\epsilon^2})^a)$. Note that the choice of constant $\delta$ depends on $k$, $\epsilon$, and $\alpha$.

**Proof.** We can use the procedure described in Section 3 to find an avatar $\epsilon'$-kernel $Q$. (We find $Q$ by finding an $\epsilon'$-approximate smallest convex hull $CH(Q)$ of $L$.) Our measure function is $\mu(.) = \text{diam}(.)$. This measure function $\text{diam}(.)$ has the property that $\text{diam}(L) = \text{diam}(CH')(L)$.

Let $\bar{u} \in S^{d-1}$ be the direction of $\text{diam}(L)$. Then we have that:

$$(1 - \epsilon')\omega(u, L) \leq \omega(u, Q), \quad \forall u \in S^{d-1}$$

$$(1 - \epsilon') \cdot \text{diam}(L) \leq \omega(\bar{u}, Q) \leq \text{diam}(Q)$$

$$\text{diam}(L) \leq (1 + \epsilon) \cdot \text{diam}(Q), \quad \text{where} \quad \epsilon = \frac{1}{1 - \epsilon'}$$

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

$$\text{diam}(Q) \leq \text{diam}(L) \leq (1 + \epsilon) \cdot \text{diam}(Q)$$

$$1 \leq \frac{(1 + \epsilon) \cdot \text{diam}(Q)}{\text{diam}(L)} \leq (1 + \epsilon)$$
4 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 ties to rainbow connectivity problems from the graph theory literature [6]. As before, in order to set the stage, we provide some formal definitions.

4.1 Avatar graph reachability

A $k$-avatar graph $G(V, E, L, A)$ (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 $A = \{A_1, \ldots, A_n\}$ such that $\forall i$, $A_i \subseteq V$ is the avatar set for entity $i$, $|A_i| \leq k$, and $A_i \cap A_j = \emptyset$ if $i \neq j$. 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 whether 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 such as DFS or BFS. Surprisingly enough, in the avatar setting it turns out to be NP-complete, even for $k = 2$.

\textbf{Theorem 9.} The $k$-avatar reachability problem is NP-complete, for $k \geq 2$.

\begin{figure}[h]
\centering
\includegraphics[width=0.7\textwidth]{figure5.png}
\caption{Sketch of reduction from CLIQUE to avatar vertex reachability.}
\end{figure}

\textbf{Proof.} The reduction is from the CLIQUE problem. Let graph $G_C(V, E)$ and integer $k$ denote an instance of the CLIQUE problem. $(G_C, k)$ is a YES instance if and only if $G_C$ contains a clique of size $k$. We construct graph $G_A(V', E')$ as follows (see Fig. 5): create $k + 2$ layers of vertex sets, $V'_0, V'_1, \ldots, V'_{k+1}$. Let $V'_0 = \{s\}$ and $V'_{k+1} = \{t\}$. For $l = 1, \ldots, k$, let $V'_l = \{v'_{i,j} : 1 \leq i \leq |V|, 1 \leq j \leq k\}$. Vertices $v'_{x,i,j}$ and $v'_{y,i,j}$ correspond to avatars
of the same entity $A_1$ (prevents picking same vertex from 2 different layers). Add edges $(v_{l,i,j}', v_{l,i,j}'+1)$ for all $l$, $i$, and $j$; for $0 \leq l < k$, add edges $(v_{l,i,k}', v_{l+1,i,j,1}')$ for all $i,j$. Note that the vertices $v_{l,i,1}',...,v_{l,i,k}'$ in layer $l$ form $k$ 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 $V_{l,i}$. Now for each pair of non-adjacent vertices $v_i, v_j \in V$ from $G'_C$, add vertices $u_{l,i,j,x}'$ and $u_{l,i,j,x}', 1 \leq x < k$ and $1 \leq l \leq k$ to $G_A$. Add edges $(u_{l,i,j,x}'u_{l,i,j,x}'+1)$ and $(u_{l,i,j,x}'u_{l,i,j,x}'+2)$, $1 \leq x < k$. Update subpaths $S_{l,i}$ by taking each outgoing edge from it and make it outgoing from $u_{l,i,j,k}'$ instead, and then adding an edge from it to $u_{l,i,j,1}'$ the new last vertex of subpath $S_{l,i}$. Finally, let each pair of vertices $u_{x,y}', u_{y,x}'$ be avatars of each other, for all $1 \leq x, y \leq k$.

It is not hard to see that the size of the graph $G_A$ is polynomial in $n$. More importantly, we claim that if the set $\{v_{i_1}, \ldots, v_{i_k}\}$ 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 $u_{l,i,j,x}'$ 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 $s \sim t$ path in $G_A$. It is readily shown that 2-avatar reachability is in NP, thus completing the proof that it is NP-complete.

### 4.2 Avatar Shortest Paths

Given an unweighted (or unit-weighted) $k$-avatar graph and two vertices $s$ and $t$ in the graph, find the shortest length avatar path from $s$ to $t$. Given that reachability is hard in the avatar setting, the shortest path would be expected to be at least as hard. The following theorem highlights its inapproximability.

▶ Theorem 10. The $k$-avatar shortest path problem is APX-Hard for $k \geq 2$.

The key to the proof is a gap-preserving reduction from the maximum clique problem, which is known to be APX-hard, implying immediately that it cannot be approximated to within any constant factor in polynomial time. The proof is omitted here because of limited space.

▶ Lemma 11. There is a gap-preserving reduction from the max-clique problem to the 2-avatar shortest path problem that transforms a graph $G_{\text{clique}}(V, E)$ to a graph $G_{\text{avatar}}(V', E')$ such that:

1. if $OPT_{\text{clique}}(G) \geq k \cdot |V|$, $OPT_{\text{avatar}}(s-t) \leq m$, and
2. if $OPT_{\text{clique}}(G) < k \cdot \alpha \cdot (k \cdot |V|)$, $OPT_{\text{avatar}}(s-t) > (2-\alpha) \cdot m$,

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

Open Problems: Open problems from this work include determining the time complexity of the $k$-avatar versions of minimum MaxGap problem and convex hull.

Acknowledgments: This work was partly supported by NSF Grant (CNS-1018262) and the NSF Graduate Research Fellowship (DGE-1038321). The authors thank Shin-ichi Tanigawa, Daniel Rodriguez, Joshua Kirstein, and Ning Xie for useful discussions on earlier drafts. The detailed reviews by anonymous FSTTCS reviewers is gratefully acknowledged.
References


