Schedule of Talks

27th Annual Fall Workshop on Computational Geometry

November 3–4, 2017

Stony Brook University
Stony Brook, NY

Sponsored by the National Science Foundation and the Department of Applied Mathematics and Statistics and the Department of Computer Science, College of Engineering and Applied Sciences, Stony Brook University

Program Committee:

Greg Aloupis (Tufts)
William Randolph Franklin (RPI)
Mayank Goswami (Queens College)
Joseph S. B. Mitchell (co-chair, Stony Brook)
Jinhui Xu (University at Buffalo)
Peter Brass (CCNY)
Jie Gao (co-chair, Stony Brook)
Matthew P. Johnson (Lehman College)
Don Sheehy (University of Connecticut)

Friday, November 3, 2017: Computer Science (NCS), Room 120

9:20 Opening remarks

Contributed Session 1 Chair: Joe Mitchell

9:30–9:45 "Sampling Conditions for Clipping-free Voronoi Meshing by the VoroCrust Algorithm", Scott Mitchell, *Ahmed Abdelkader*, Ahmad Rushdi, Mohamed Ebeida, Ahmed Mahmoud, John Owens and Chandrajit Bajaj

9:45–10:00 "Quasi-centroidal Simplicial Meshes for Optimal Variational Methods", *Xiangmin Jiao*

10:00–10:15 "A Parallel Approach for Computing Discrete Gradients on Mulfiltrations", *Federico Iuricich*, Sara Scaramuccia, Claudia Landi and Leila De Floriani

10:15–10:30 "Parallel Intersection Detection in Massive Sets of Cubes", W. Randolph Franklin and Salles V.G. Magalhães

10:30–11:00 *Break.*

Contributed Session 2 Chair: Jie Gao

11:00–11:15 "Dynamic Orthogonal Range Searching on the RAM, Revisited", Timothy M. Chan and *Konstantinos Tsakalidis*

11:15–11:30 "Faster Algorithm for Truth Discovery via Range Cover", Ziyun Huang, Hu Ding and Jinhui Xu

11:30–11:45 "An Efficient Sum Query Algorithm for Distance-based Locally Dominating Functions", Ziyun Huang and Jinhui Xu

11:45–12:00 "Approximate Convex Hull of Data Streams", Avrim Blum, Vladimir Braverman, Ananya Kumar, Harry Lang and Lin Yang

12:00–1:00 Lunch (provided)

1:00–2:00 Invited Talk: "The geometry of data structures", John Iacono (NYU Tandon School of Engineering and UniversitÉLibre de Bruxelles)

Abstract: A technique has been developed to study and analyze data structures called the geometric view. Several examples of the use of this view will be presented including, rotation-based binary search tree data structure, cache-oblivious persistence, and disjoint sets. In all of these examples the main theme is that by moving to the geometric view, details of the data structure that seem cumbersome and hard to approach are abstracted in such a way that allows a simple, yet equivalent, view.

Contributed Session 3 Chair: Jinhui Xu

2:00–2:15 "Online Unit Covering in L_2", *Anirban Ghosh*

2:15–2:30 "An $O(n \log n)$-Time Algorithm for the k-Center Problem in Trees", Haitao Wang and Jingru Zhang

2:30–2:45 "Compact Data Structures for Abstract Order Types and Optimal Encodings for SUM Problems", Jean Cardinal, Timothy Chan, John Iacono, Stefan Langerman and *Aurélien Ooms*

2:45–3:00 "Lightweight Sketches for Mining Trajectory Data", Maria Astefanoaei, Panagiota Katsikouli, *Mayank Goswami* and Rik Sarkar

3:00–3:30 Break

Contributed Session 4 Chair: Don Sheehy

3:30–3:45 "Calculating the Dominant Guard Set of a Simple Polygon", Eyup Serdar Ayaz and Alper Ungor

3:45–4:00 "Perfect Polygons", Hugo A. Akitaya, Erik D. Demaine, Martin L. Demaine, Adam Hesterberg, Joseph S.B. Mitchell and *David Stalfa*

4:00–4:15 "Efficient Approximations for the Online Dispersion Problem", Jing Chen, Bo Li and Yingkai Li

4:15–4:30 "Edge Conflicts Can Increase Along Minimal Rotation-Distance Paths", *Sean Cleary* and Roland Maio

4:30–4:45 "Optimal Safety Patrol Scheduling Using Randomized Traveling Salesman Tour", *Hao-Tsung Yang*, Shih-Yu Tsai, Jie Gao and Shan Lin

5:00–6:00 Open Problem Session

Saturday, November 4, 2017: Computer Science (NCS), Room 120

Contributed Session 5 Chair: Jie Gao

9:30–9:45 "On the Density of Triangles with Periodic Billiard Paths", *Ramona Charlton*

9:45–10:00 "Nearest Neighbor Condensation with Guarantees", Alejandro Flores Velazco and David Mount

10:00–10:15 "On the Complexity of Random Semi Voronoi Diagrams", Chenglin Fan and Benjamin Raichel

10:15–10:30 "Efficient Algorithms for Computing a Minimal Homology Basis", Tamal K. Dey, *Tianqi Li* and Yusu Wang

10:30–11:00 Break

Contributed Session 6 Chair: Matt Johnson

11:00–11:15 "Cardiac Trabeculae Segmentation: an Application of Computational Topology", Chao Chen, Dimitris Metaxas, Yusu Wang, Pengxiang Wu and Changhe Yuan

11:15–11:30 "Topological and Geometric Reconstruction of Metric Graphs in R^n", Brittany Fasy, Rafal Komendarczyk, Sushovan Majhi and Carola Wenk

11:30–11:45 "Randomized Incremental Construction of Net-Trees", *Mahmoodreza Jahanseir* and Donald Sheehy

11:45–12:00 "On Computing a Timescale Invariant Bottleneck Distance", *Nicholas J. Cavanna*, Oliver Kisielius and Donald R. Sheehy

12:00–1:00 Lunch (provided)

1:00–2:00 Invited Talk: "Sublinear algorithms for outsourced data analysis", Suresh Venkatasubramanian (University of Utah)

Abstract: In the era of outsourcing, communication has replaced computation as an expensive resource. Researchers have proposed numerous models for communication-efficient computing that draw on classical ideas of interactive proofs, updated for our more "sublinear" world.

I'll talk about the model of streaming interactive proofs and how we can solve classic problems in data analysis like near neighbor search, minimum enclosing balls, and range searching in general with sublinear communication.

Contributed Session 7 Chair: Mayank Goswami

2:00–2:15 "Improved Results for Minimum Constraint Removal", Eduard Eiben, Jonathan Gemmell, Iyad Kanj and Andrew Youngdahl

2:15–2:30 "Algorithm for Optimal Chance Constrained Knapsack with Applications to Multi-robot Teaming", Fan Yang and Nilanjan Chakraborty

2:30–2:45 "Freeze Tag Awakening in 2D is NP-hard", *Hugo Akitaya*, Jingjin Yu and Zachary Abel

2:45–3:00 "Freeze Tag is Hard in 3D", Erik D. Demaine and Mikhail Rudoy; Matthew P. Johnson ("Easier Hardness for 3D Freeze-Tag")

3:00–3:30 Break

Contributed Session 8 Chair: W. Randolph Franklin

3:30–3:45 "Declarative vs Rule-based Control for Flocking Dynamics", Usama Mehmood, Nicola Paoletti, Dung Phan, Radu Grosu, Shan Lin, Scott Stoller, Ashish Tiwari, Junxing Yang and Scott Smolka

3:45–4:00 "Realizing Minimum Spanning Trees from Random Embeddings", Saad Quader, Alexander Russell and Ion Mandoiu

4:00–4:15 "The Minimum Road Trips Problem", *Samuel Micka* and Brendan Mumey

4:15–4:30 "Harder Hardness of Approximation for 2D r-Gather", Matthew P. Johnson

Sampling Conditions for Clipping-free Voronoi Meshing by the VoroCrust Algorithm

Scott Mitchell[1], Ahmed Abdelkader[2], Ahmad Rushdi[1,3], Mohamed Ebeida[1], Ahmed Mahmoud[3], John Owens[3] and Chandrajit Bajaj[4]
[1] Sandia National Laboratories
[2] University of Maryland, College Park
[3] University of California, Davis
[4] University of Texas, Austin

For presentation at the Fall Workshop on Computational Geometry (FWCG 2017)

Abstract

Decomposing the volume bounded by a closed surface using simple cells is a key step in many applications. Although tetrahedral meshing is well-established with many successful implementations, many research questions remain open for polyhedral meshing, which promises significant advantages in some contexts. Unfortunately, current approaches to polyhedral meshing often resort to clipping cells near the boundary, which results in certain defects. In this talk, we present an analysis of the VoroCrust algorithm, which leverages ideas from α-shapes and the power crust algorithm to produce conforming Voronoi cells on both sides of the surface. We derive sufficient conditions for a weighted sampling to produce a topologically-correct and geometrically-accurate reconstruction, using the ϵ-sampling paradigm with a standard sparsity condition. The resulting surface reconstruction consists of weighted Delaunay triangles, except inside tetrahedra with a negative weighted circumradius where a Steiner vertex is generated close to the surface.

Generating quality meshes is an important problem in computer graphics and geometric modeling. Polyhedral meshing offers higher degrees of freedom per element than tetrahedral or hex-dominant meshing, and is more efficient in filling a space, because it produces fewer elements for the same number of nodes. Within the class of polyhedral mesh elements, Voronoi cells are particularly useful in numerical simulations for their geometric properties, e.g., planar facets and positive Jacobians.

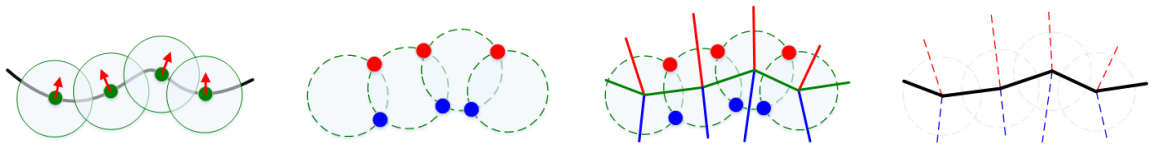
A conforming mesh exhibits two desirable properties *simultaneously*: 1) a decomposition of the enclosed volume, and 2) a reconstruction of the bounding surface. A common technique for producing boundary-conforming decomposition from Voronoi cells relies on *clipping*, i.e., intersecting and truncating, each cell by the bounding surface [3]. An alternative to clipping is to locally mirror the Voronoi generators on either side of the surface [2].

VoroCrust can be viewed as a principled mirroring technique. Similar to the power crust [1], the reconstruction is composed of the facets shared by cells on the inside and outside of the manifold. However, VoroCrust uses pairs of unweighted generators tightly hugging the surface, which allows further decomposition of the interior without disrupting the surface reconstruction. VoroCrust can also be viewed as a special case of the weighted α-shape [4]. A description of the abstract VoroCrust algorithm we analyze is provided next. Figure 1 illustrates the basic concepts in 2D.

The Abstract VoroCrust Algorithm

1. Take as input a weighted point sampling $\mathcal{P} = \{(p_i, w_i)\}$ of a closed 2-manifold $\mathcal{M}$.
2. Use weights to define a ball B_i of radius $r_i = \sqrt{w_i}$ centered at each sample p_i.
3. Find intersecting triplet of balls to obtain one corner point on either side of $\mathcal{M}$.
4. Collect the Voronoi generators $\mathcal{G}$ as the set of corner points outside all sample balls.
5. Optionally, include in $\mathcal{G}$ more generators far-inside $\mathcal{M}$ to further decompose its interior.
6. Output the Voronoi diagram of $\mathcal{G}$ as the desired decomposition, and the facets separating the inside and outside generators as the reconstructed surface $\hat{\mathcal{M}}$.

Problem Statement: We seek to characterize the locations and weights of the input samples in Step (1) to guarantee a topologically-correct and geometrically-accurate reconstruction.



(a) Weighted samples. (b) Intersection pairs. (c) Voronoi cells. (d) Reconstruction facets.

Figure 1: VoroCrust reconstruction, demonstrated on a planar curve. The weight of a point defines the radius of a ball around it. The reconstruction is the Voronoi facets separating the uncovered intersection pairs on opposite sides of the manifold.

Summary of Results: An ϵ-sampling $\mathcal{S}$ is *δ-weighted* if each sample is associated with a ball of radius $r_i = \delta\,\mathrm{lfs}(p_i)$, with $\delta \geq \epsilon$. In addition, $\mathcal{S}$ is *conflict-free* if $\forall j \neq i, \|p_i - p_j\| \geq \epsilon \cdot \min(\mathrm{lfs}(p_i), \mathrm{lfs}(p_j))$. For some constants ϵ, δ, c, we show that the reconstruction $\hat{\mathcal{M}}$ is geometrically-close and topologically-correct. Moreover, $\hat{\mathcal{M}} \to \mathcal{M}$ quadratically as $\epsilon \to 0$. Specifically, for every $p \in \mathcal{M}$ with closest point $q \in \hat{\mathcal{M}}$, and for every $q \in \hat{\mathcal{M}}$ with closest point $p \in \mathcal{M}$, we have $\|pq\| < c \cdot \delta^2 \mathrm{lfs}(p)$. The reconstruction $\hat{\mathcal{M}}$ contains every input sample as a vertex. Rather than filtering, the algorithm as outlined above naturally resolves slivers by introducing Steiner vertices.

References

[1] Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust, unions of balls, and the medial axis transform. *Computational Geometry*, 19(2):127–153, July 2001.

[2] Joseph E. Bishop. Simulating the pervasive fracture of materials and structures using randomly close packed Voronoi tessellations. *Computational Mechanics*, 44(4):455–471, September 2009.

[3] Mohamed S. Ebeida and Scott A. Mitchell. Uniform random Voronoi meshes. In *International Meshing Roundtable (IMR)*, pages 258–275, 2011.

[4] Herbert Edelsbrunner. *Weighted alpha shapes*. University of Illinois at Urbana-Champaign, Department of Computer Science, 1992.

Quasicentroidal Simplicial Meshes for Optimal Variational Methods

Xiangmin Jiao

***Abstract*—Mesh generation is an important problem in computational geometry with numerous applications. One of its most important applications is the variational methods for the numerical solutions of partial differential equations, which are essential not only for scientific discoveries and engineering innovations but also computer-aided design. Traditional requirements on mesh generation include *well-shapedness* in aspect ratios and *quasiuniformity* in element sizes, which are necessary conditions for the convergence of Galerkin finite elements. In this work, we introduce the concepts of *quasicentroidal meshes*, in which all the elements are nearly symmetric about their subelements. These meshes can enable superconvergence of some variational methods and hence optimal efficiency. In particular, quadratic finite elements on quasicentroidal meshes enjoy fourth-order convergence for L^2 projection and the Poisson equation, compared to the regular third-order convergence. We propose an approach to generate quasicentroidal simplicial meshes based on quasiregular honeycomb tessellations and pose some open problems in optimal mesh generation.**

I. Introduction

Mesh generation is an important problem at the intersection of computational geometry, applied mathematics, and applications. It has numerous applications in computer science and computer engineering, such as computer graphics, visualization, image processing, computer-aided design, which provide valuable tools for scientists and engineers. Meshes also provide geometric supports for defining well-behaved piecewise basis functions, which are the underpinnings of any variational method for solving partial differential equations (PDEs) that describe some physics phenomena. Therefore, the importance of mesh generation for science and engineering cannot be overstated. The requirements of mesh generation are typically motivated by the convergence properties of the variational methods. Therefore, rigorous convergence analysis, especially in terms of their requirements on meshes, is fundamentally important to the researchers in mesh generation, as well as to the practitioners who rely on variational methods in engineering, computer-aided design, etc.

Mesh generation has been investigated for a few decades. In this work, we focus on simplicial meshes due to their flexibility in dealing with complex geometries and mesh adaptation. We will review the traditional requirements of mesh generation, and then propose a new optimality condition called *quasicentroidality*. This condition is important for achieving *superconvergence* of variational methods using quadratic elements, and hence it can lead to better accuracy without extra computational cost. We propose a method to generate these meshes based on quasiregular honeycomb tessellations and pose some new open problems in optimal mesh generation.

Department of Applied Mathematics and Statistics, Stony Brook University, Stony Brook, NY 11794. xiangmin.jiao@stonybrook.edu.

II. Traditional Requirements on Meshes

To motivate the requirements on mesh generation, let us consider some example variational problems. One of the simplest examples is the Poisson equation

$$-\Delta u(\boldsymbol{x}) = f(\boldsymbol{x}) \quad \text{for } \boldsymbol{x} \in \Omega \subseteq \mathbb{R}^d \tag{1}$$

with some Dirichlet boundary condition $u(\boldsymbol{x}) = u_D(\boldsymbol{x})$ on its boundary Γ. In the context of CAD, the Laplace operator is often replaced by the surface Laplacian, a.k.a. the Laplace-Beltrami operator, which introduce nonlinearity. For simplicity, let us focus on the linear case by assuming the standard Laplace operator.

To solve a PDE using variational methods, we first introduce a set of *test functions* $\{\psi_1, \psi_2, \ldots, \psi_n\}$. Assume that the ψ_i are (weakly) differentiable and that they vanish along the Dirichlet boundary $\partial\Omega$. By taking the inner product of (1) with ψ_i and then integrating the second derivatives by parts, we can convert (1) into the *variational form* (or the *weak form*)

$$\int_\Omega \boldsymbol{\nabla} u \cdot \boldsymbol{\nabla} \psi \,\mathrm{d}\boldsymbol{x} = \int_\Omega f\psi \,\mathrm{d}\boldsymbol{x}. \tag{2}$$

We introduce a set of *basis functions* (or *trial functions*) $\boldsymbol{\phi} = [\phi_1, \phi_2, \ldots, \phi_n]^T$ to approximate u as $u \approx \boldsymbol{u}_h^T \boldsymbol{\phi}$, and then obtain an algebraic equation

$$\boldsymbol{K}\boldsymbol{u}_h = \boldsymbol{f}. \tag{3}$$

In the Galerkin methods, $\{\phi_i\} = \{\psi_i\}$. A closely related variational problem is the *projection*

$$\int_\Omega u\psi_i \,\mathrm{d}\boldsymbol{x} = \int_\Omega f\psi_i \,\mathrm{d}\boldsymbol{x}, \tag{4}$$

which is a zeroth-order elliptic problem known as L^2 *projection* in the context of Galerkin methods. See e.g. [3] for more detail.

The basis and trial functions are typically defined through the aid of a *mesh* that tessellates Ω into a set of elements τ_k, i.e., $\Omega = \bigcup_{k=1}^{m} \tau_k$. We assume the mesh is *conformal*, which means that any two adjacent elements must meet at a lower-dimensional subelement (a node, edge, face, etc.) shared by the two elements. For meshes with linear elements, two conditions are typically imposed on mesh generation for the Galerkin methods:

Quasiuniformity in element sizes: The ratio between the sizes of the largest and smallest elements should be bounded by a constant in some metric space.

Well-shapedness in aspect ratio. The maximum angle should be bounded away from $180°$, or in a simplified sense, the ratio between the circumradius to inradius (i.e., the *aspect ratio*) should be bounded by some constant.

Quasiuniformity and well-shapedness together are sufficient mesh conditions for the convergence of the variational formulation with consistent discretizations for well-posed PDEs. Quasiuniformity is a necessary condition for the well-conditioning of all variational methods, whereas well-shapedness is a necessary condition for second and higher-order PDEs. Besides their importance for linear elements, these conditions also apply to quadratic and higher-order elements. In addition, the following condition is often cited in mesh generation:

Voronoi/Delaunay property: The facets of the Voronoi cells pass through the mid-points of, and are perpendicular to, the edges in the Delaunay mesh.

The Voronoi/Delaunay property is typically used as a heuristic for achieving well-shapedness of elements. More importantly, the Voronoi property is useful in the context of finite volume methods, which compute fluxes along the facets of the Voronoi cells: Thanks to the symmetry about the facets, the leading-order error terms in the fluxes cancel out on Voronoi meshes similar to centered differences in 1-D [4]. In addition, the so-called *centroidal Voronoi tessellation* (CVT) has been shown to enable better accuracy for various applications such as image compression, cellular biology, etc. [1]. However, such a notion of "symmetry" or "centroidality" has not been investigated systematically in the context of finite elements. The goal of the remainder of this work is to address this fundamental question, with a focus on meshes with quadratic elements.

III. Centroidality and Quasicentroidality

We now introduce a new condition for mesh generation with some notion of "symmetry" of a mesh. From numerical-analysis point of view, we introduce the conditions of *centroidality* and *quasicentroidality*, which lead to error cancellations in variational methods, analogous to the error cancellation due to the Voronoi property for finite volumes.

First, we introduce the following notion of *weighted centroids*, which generalizes the notion of centorids.

Definition 1. Given a region $\tau \in \mathbb{R}^d$, its *weighted centroid* with a weighting function $\psi(\boldsymbol{x})$ is a point $\bar{\boldsymbol{x}} \in \mathbb{R}^d$ such that

$$\int_\tau (\boldsymbol{x} - \bar{\boldsymbol{x}})\psi(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = \boldsymbol{0}. \tag{5}$$

If $\psi \equiv 1$, the above definition reduces to the centroid (a.k.a. the center of mass). If $\psi(\boldsymbol{x})$ is a positive function, then $\bar{\boldsymbol{x}}$ lies within the convex hull of τ. In general, ψ is the Lagrange polynomial basis functions corresponding to the nodes of τ, and hence ψ may have negative values. Because ψ is in general nonlinear, the existence of the weighted centroid cannot be taken for granted. We note the following fact for elements with Lagrange polynomial basis functions.

Proposition 2. *Any element τ with stable degree-p Lagrange basis functions $\{\phi_i\}$ must have a weighted centroid with respect to each of its basis function ϕ_i.*

Here by stable basis functions, we mean that for each $\phi_i(\boldsymbol{x})$, $\int_\tau |\phi_i(\boldsymbol{x})| \, \mathrm{d}\boldsymbol{x} \le C \int_\tau \phi_i(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}$ for some small $C > 0$. The proof of Proposition 2 involves a geometric argument.

Proof: Let $\psi = \phi_i(\boldsymbol{x})$ in (5). Consider each nonlinear equation

$$g_j(\bar{\boldsymbol{x}}) = (x_j - \bar{x}_j)\psi(\boldsymbol{x}),$$

for $j = 1, 2, \dots d$. By assumption, $\int_\tau \psi(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} > 0$. Let $\boldsymbol{e}_j$ denote the vector along the jth coordinate axis. Since $g_j(\bar{\boldsymbol{x}})$ is a polynomial in $\bar{\boldsymbol{x}}$ and $G_j = \int_\tau g_j(\alpha \boldsymbol{e}_j) \, \mathrm{d}\boldsymbol{x}$ has opposite signs as $\alpha \to -\infty$ and $\alpha \to \infty$, there must exist two hyperplanes P_1 and P_2 perpendicular to $\boldsymbol{e}_j$ such that G_j has opposite signs for any pair of points from beyond P_1 and P_2, respectively. Therefore, the zero levelset of $G_j(\bar{\boldsymbol{x}})$ exists and is bounded between P_1 and P_2. The d zero-levelset hypersurfaces of (5) intersect pairwise within the hyperrectangle defined by the d pairs of bounding hyperplanes, so the intersection must exist within the hyperrectangle in $\mathbb{R}^d$. ∎

Now, we shall focus on quadratic simplicial meshes, of which the nodes include the vertices of the simplices, along with the centers of the edges, faces, etc. For these meshes, we define centroidality as follows.

Definition 3. A node $\boldsymbol{x}_i$ in a mesh is *centroidal* if the weighted centroids of its incident elements with respect to the basis function $\phi_i(\boldsymbol{x})$ are symmetric pairs about $\boldsymbol{x}_i$. Node $\boldsymbol{x}_i$ is *quasicentroidal* if it is within $\mathcal{O}(h^2)$ from the average of each symmetric pair of weighted centroids.

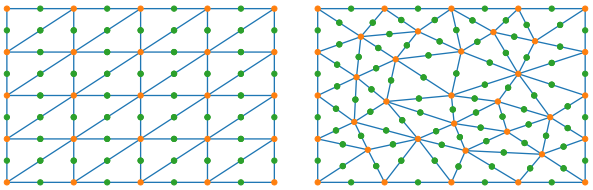


Figure 1. Example quadratic triangular meshes with centroidal (left) and non-centroidal nodes in the interior.

We say a node is *non-centroidal* if it is not quasicentroidal. Figure 1 shows two example quadratic meshes with centroidal and non-centroidal nodes, respectively. The quasicentroidality concept is fundamental from a numerical-analysis point of view, because it enables error cancellation in the variational forms. Let u_I denote the interpolation of a smooth function u with the basis functions ϕ_i. Let σ denote the collection of elements around a quasicentroidal node $\boldsymbol{x}_i$, i.e., $\sigma = \cup\{\tau_k \mid \boldsymbol{x}_i \in \tau_k\}$, the 1-ring neighborhood of $\boldsymbol{x}_i$. Let V_σ be the volume of σ, and h the longest edge length in σ. Then, it can be shown that [2]

$$\left| \int_\sigma (u - u_I)\, \phi_i \, \mathrm{d}\boldsymbol{x} \right| \leq V_\sigma \mathcal{O}(h^4). \tag{6}$$

As a corollary, L^2 projection with quadratic elements converges at the order of $\mathcal{O}(h^4)$, assuming the boundary values are approximated to fourth order accuracy. The quadratic Galerkin method for the Poisson equation also converges at the order of $\mathcal{O}(h^4)$ assuming Dirichlet boundaries [2].

From the point of view of mesh generation, we define the following concept about meshes.

Definition 4. A mesh is *centroidal* if the elements are symmetric about all of the sub-elements (nodes, edges, faces, etc.) in the interior of the mesh.

In a centroidal mesh, all the interior nodes (i.e., excluding the nodes on Γ) are *centroidal*. The symmetry requirement in Definition 4 does not allow mesh grading. One can define a centroidal mesh in a metric space and then map it onto the Euclidean space, which leads to a *quasicentroidal* mesh, of which the interior nodes are quasicentroidal. This allows quadratic meshes to have smooth mesh grading while still enjoying error cancellation for quadratic Galerkin methods.

In the preceding discussions, we have left out one fundamental question: *Do centroidal simplicial meshes exist in all dimensions?* For a hyperbox in $\mathbb{R}^d$, it is trivial to prove the existence of centroidal meshes: The hyperbox can be tessellated by congruent hyperrectangles, which form a centroidal mesh with tensor-product elements. We can subdivide the hyperrectangles into simplices in a locally symmetric fashion, which will result in centroidal simplicial meshes. For irregular domains, it is very difficult, if not impossible, to ensure quasicentroidality near boundaries. We will revisit this issue in Section V.

IV. Well-Shaped Centroidal Meshes

Quasicentroidal meshes have excellent numerical properties in terms of error cancellation. However, all quasicentroidal meshes are not equally good for Galerkin methods, because well-shapedness of elements must also be taken into account. The elements generated by subdividing hyperrectangles may not produce well-shaped simplex elements, so the solutions may not have optimal accuracy in terms of the constant factor of the leading error term.

In 2-D, regular triangles are space filling, and a mesh composed of regular triangles in the interior are well-shaped and centroidal. Such a mesh is optimal for isotropic problems. For anisotropic problems, we can introduce smoothly varying metric tensors to grade the mesh, which would result in quasicentroidal meshes.

In 3-D, regular tetrahedra are not space filling. However, the 3-D space can be tessellated by the so-called *quasiregular honeycomb*, such as the *tetrahedral-octahedral honeycomb*; see Figure 2 for an example. In this tessellation, each tetrahedron is regular with equal dihedral angles of $70.53°$, and each octahedron is also regular with equal dihedral angles of $109.42°$. All the faces are equilateral triangles. We can subdivide each octahedron into two pyramids, and then uniformly subdivide each pyramid into two or four congruent tetrahedra in a locally symmetric fashion. We refer to the resulting tetrahedral mesh as *TOH-based*. In this mesh, every face has two equal-sized minimum angles of $45°$ or $60°$. In each tetrahedron, there are three equal-sized minimum angles of $70.53°$ for the regular tetrahedra and of $54.71°$ for those from subdividing octahedra. The maximum angles are $90°$. These elements minimize the maximum errors in the derivative approximations within a factor of 2 among all tetrahedra, and hence they are nearly optimal in terms of well-shapedness for isotropic problems. This optimality is beneficial for not only quadratic elements but also linear elements. Similar to triangular meshes in 2-D, the meshes can be graded for anisotropic problems by introducing metric tensors.

Another desirable property of the TOH-based mesh is that each tetrahedra can be uniformly subdivided into four regular tetrahedra and one regular octahedron, and each octahedron can be uniformly subdivided into six regular octahedron and eight regular tetrahedra. This allows uniform mesh refinement to enable efficient multigrid methods, which have linear time complexity for solving linear systems from variational methods.

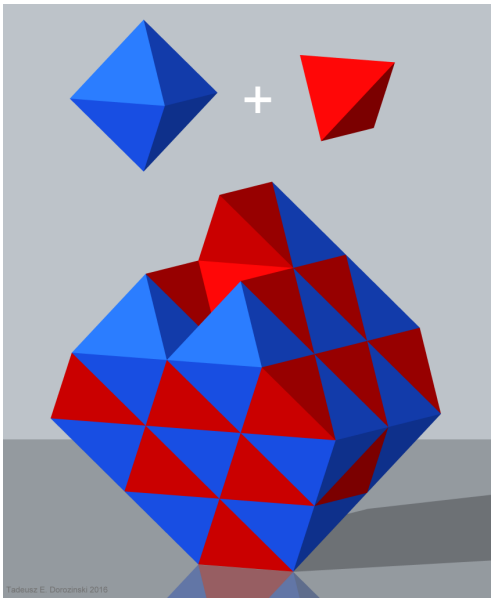


Figure 2. Quasiregular honeycomb with octahedra and tetrahedra [5].

V. Nearly Quasicentroidal Meshes

In general, quasicentroidality cannot be attained near the boundary of an irregular domain. From numerical point of view, this is not an issue, because boundary nodes are not quasicentroidal for quadratic elements, so special treatments are required for boundary nodes in order to achieve uniform fourth-order convergence in the presence of Neumann boundaries. These treatments would require degree-4 basis functions locally near the boundary, which are more expensive than using quadratic elements by a constant factor. For the nodes near boundaries, if quasicentroidality is lost locally, a similar treatment can be applied to these nodes. Due to the surface-to-volume ratio, the additional cost for these nodes is negligible, and hence optimal performance can still be achieved. More generally, if a mesh loses quasicentroidality for $\mathcal{O}(n^\alpha)$ nodes with $\alpha < 1$, we say the mesh is *nearly quasicentroidal*. From a practical point of view, near quasicentroidality can enable the same optimal performance as quasicentroidal meshes, as long as additional care is taken to ensure uniform convergence for the non-centroidal nodes.

For irregular domains, the TOH-based meshes are more adaptive to irregular domains than those obtained from subdividing cubes. This is because the outer surface of a TOH-based mesh M is composed of equilateral triangles. To generate meshes near boundary, we propose a cut-cell approach as follows. Assuming that the mesh is sufficiently fine near boundary, we compute the intersection of the edges with Γ, snap the nodes onto the surface if they are within $h/2$ to the boundary, and tetrahedralize the remainder of the cut-cells. This ensures that the elements next to the boundaries are quasiuniform and are reasonably well-shaped, with only small increase to the maximum angles.

VI. Conclusions and Discussions

In this work, we introduced the concepts of centroidal and quasicentroidal meshes, which enjoy superconvergence in the interior for quadratic Galerkin methods. These are defined based on the *weighted centroids* of the elements, where the weighting functions are the test functions in the variational methods. In practice, quasicentroidality may be lost near boundaries and at some isolated points. This leads to near quasicentroidal meshes, which still enjoy the same optimal efficiency as quasicentroidal meshes as long as special care is taken to ensure uniform convergence.

This work only addressed the theoretical issue mostly at a conceptual level. We outlined an algorithm for generating nearly quasicentroidal meshes based on tetrahedral-octahedral honeycomb in the interior and cut-cells near boundaries. As a future research direction, we plan to implement the method and compare it against other mesh generation techniques in terms of both the efficiency of mesh generation and the accuracy of variational methods. We expect the method also generalizes to 4-D based on quasiregular honeycombs in $\mathbb{R}^4$.

In terms of surfaces, there are some interesting open problems. For spheres, there exist quasiregular tessellation of spheres, such as the tetrakis hexahedron and disdyakis dodecahedron, which can be used to generate quasicentroidal meshes. However for genus-k surfaces with $k \geq 1$, it is unclear whether quasiregular tessellations exist to generate quasicentroidal meshes. If the answer to the question is negative, then what is the lowest bound of the number of non-centroidal points in a nearly quasicentroidal mesh on a genus-k surface?

References

[1] Q. Du, V. Faber, and M. Gunzburger. Centroidal Voronoi tessellations: Applications and algorithms. *SIAM Review*, 41(4):637–676, 1999.

[2] X. Jiao. Backward error analysis of variational methods I: High-order and super-convergence of Galerkin FEM. 2017. In preparation.

[3] M. G. Larson and F. Bengzon. *The Finite Element Method: Theory, Implementation, and Applications*, volume 10. Springer Science & Business Media, 2013.

[4] I. D. Mishev. Finite volume methods on Voronoi meshes. *Numerical Methods for Partial Differential Equations*, 14(2):193–212, 1998.

[5] Wikipedia. Honeycomb (geometry), 2017. Online; accessed 22-Oct-2017; https://en.wikipedia.org/wiki/Honeycomb_(geometry).

A parallel approach for computing discrete gradients on mulfiltrations

Federico Iuricich[1], Sara Scaramuccia[2], Claudia Landi[3] and Leila De Floriani[1]

[1]Department of Geographical Sciences, University of Maryland, College Park (MD), USA.
[2]Department of Computer Science, Bioengineering, Robotics, and Systems Engineering, University of Genova, Genova, Italy.
[3]Department of Sciences and Methods for Engineering DISMI, University of Modena and Reggio Emilia, Modena-Reggio Emilia, Italy

1 Introduction

Persistent homology is a powerful tool for analyzing shapes based on their homological features [4]. For computing persistence homology we need a cell complex Γ (our shape), and a filtering function $f : \Gamma \to \mathbb{R}$. It is common in applications, especially for the analysis of scalar fields, to having to deal with multiple filtrations (i.e., cell complexes with a vector-valued function defined on its vertices). For analyzing this kind of data Multidimensional Persistent Homology (MPH) [3] has been defined. Computing MPH is extremely challenging even with data of modest size and scalable algorithms are needed for extracting this information efficiently. Here, we consider using a discrete gradient field V [6] as an intermediate representation for computing MPH on Γ and f. From V we extract its associated chain complex, represented as a graph G, where: the nodes are the critical cells of V and, the arcs are obtained by following the gradient paths of V. By following our approach for computing V we guarantee that the corresponding chain complex G shares the same MPH of Γ, according to f, but it is also composed by fewer cells. Then, computing MPH on G is much faster than on Γ.

2 Background

We will discuss our work in terms of triangle meshes, although, all the results are valid for any kind of regular cell complex. A k-dimensional simplex, or k-simplex, σ is the convex hull of $k+1$ affinely independent points. A *face* τ of σ is the convex hull of any subset of $k-1$ points of σ (indicated $\tau < \sigma$), while σ is a *coface* of τ (indicated $\sigma > \tau$). A simplicial complex Σ is a collection of k-simplices such that every face of a simplex in Σ is also in Σ and the intersection of any two simplices of Σ is a face of both. Triangle meshes (or meshes for brevity) are examples of simplicial complexes: the vertices, edges, and faces correspond to 0-, 1- and 2-simplices, respectively. We will denote by Σ_k the set of k-simplices in Σ. Given a triangle mesh Σ, a filtration is a sequence of meshes $\Sigma^0 \subset \Sigma^1 \subset ... \subset \Sigma^n = \Sigma$. By defining a real-valued function on the vertices of Σ it is possible to induce a filtration by assigning, to each simplex of Σ, the maximum function value of its vertices. The filtration of Σ is then defined as the sequence of sub-level complexes $\Sigma^u = f^{-1}(-\infty, u]$. In this context, persistent homology [4] is used to study the homological changes of the sub-level sets $\Sigma^u = f^{-1}(-\infty, u]$.

The algorithm proposed in this paper retrieves an acyclic discrete vector field (called discrete gradient for brevity) over the domain Σ. The relevance of this output has to be seen within the framework of Forman's discrete Morse Theory [6]. In discrete Morse Theory, a *(discrete) vector* is a pair of simplices (σ, τ) such that $\sigma < \tau$. A *discrete vector field* V is any collection of vectors over a simplicial complex such that each simplex belongs to at most one vector. Given a discrete vector field V, if a simplex belongs to no vector, it is called *critical*. A V-path is a sequence of vectors (σ_i, τ_i) belonging to V , for $i = 1, \ldots, r$, such that, for all indexes $i \leqslant r-1$, $\sigma_{i+1} < \tau_i$ and $\sigma_i \neq \sigma_{i+1}$. A V-path might be *closed* if $\sigma_1 = \sigma_r$ and *trivial* if $r = 1$. A *discrete gradient* V is a discrete vector field whose closed V-path are all trivial. Forman [6] proves that the homology of Σ is always isomorphic to the homology of the chain complex G computed from V. The discrete gradient V can be adapted to preserve the sub-level structure with respect to a filtering function. In the univariate setting, Theorem 4.3 in [8] proves that the persistent homologies of V also coincides with those of G. For multidimensional persistent homology, this result is guaranteed by Corollary 3.12 in [1].

3 Computing a discrete vector field in parallel

For ease of exposition, in the reminder of this work we will focus on a specific class of cell complexes, the simplicial complexes. Let Σ be a d-dimensional simplicial complex and $f : \Sigma_0 \longrightarrow \mathbb{R}^n$ a vector-valued function defined on the 0-simplices (vertices) of Σ. We require

the function to be component-wise injective which is achieved by means of simulation of simplicity [5]. From the injective function, a new function $\tilde{f} = (\tilde{f}_0, \ldots, \tilde{f}_n)$ is defined on each simplex σ of Σ as, $\tilde{f}_i(\sigma) = \max_{v \in \sigma} f_i(v)$. Then, our approach is organized in three steps:

- an indexing schema is computed on the vertices of Σ and successively extended to all its simplices,
- the lower star of each vertex is split into independent sets based on such schema,
- gradient pairs and critical cells are identified in each independent set and combined to form the output discrete gradient.

Computing the Indexing schema. In the first step we compute a *well-extensible* indexing I by defining a total order over the vertices of Σ.

Definition 1 *An indexing schema I is well-extensible with respect to function $\tilde{f}$ if and only if, for every two simplices σ_1 and σ_2,*

$$\tilde{f}(\sigma_1) \preceq \tilde{f}(\sigma_2) \implies \tilde{I}(\sigma_1) \leqslant \tilde{I}(\sigma_2),$$

where $\tilde{I}(\sigma) := \max_{v \in \sigma} I(v)$.

In our approach we produce I by ordering all the vertices with respect to a single and fixed component $i \in \{1, \ldots, n\}$.

Proposition 1 *I is a well-extensible indexing.*

Proof. I provides a total order on the vertices so, $\tilde{f}(\sigma) \preceq \tilde{f}(\tau)$ implies that $\tilde{f}_i(\sigma) \leqslant \tilde{f}_i(\tau)$. That is, there exists a vertex $v \in \tau$ such that $\tilde{f}_i(v)$ is greater than or equal to all $\tilde{f}_i(w)$ with $w \in \sigma$. This implies that $I(w) < I(v)$ for every w in Σ, and hence $\tilde{I}(\sigma) \leqslant \tilde{I}(\tau)$. ▫

Extracting independent sets. From this point vertices are processed one by one (possibly in parallel) and, for each vertex $v \in \Sigma_0$, the *index-based lower star subdivision* $\mathrm{L}_I(v)$ is computed. Intuitively, $\mathrm{L}_I(v)$ is the set of simplices having v has vertex with maximum value of $I(\cdot)$,

$$\mathrm{L}_I(v) := \{\sigma \in \Sigma \mid \tilde{I}(\sigma) = I(v)\}.$$

Within a single index-based lower stars we may have simplices with different values of $\tilde{f}$. However, in order to preserve the persistence module, two simplices σ, τ should be paired if and only if they belong to the same level-set (i.e., $\tilde{f}(\sigma) = \tilde{f}(\tau)$). Thus, for each $\mathrm{L}_I(v)$ we subdivide the simplices according to $\tilde{f}$. More precisely, K_v is created as the quotient of $\mathrm{L}_I(v)$ obtained through the equivalence relation $\sigma \sim \tau$ if and only if $\tilde{f}(\sigma) = \tilde{f}(\tau)$. According to the definition of index-based lower star, we can easily see that each simplex of Σ belongs to exactly one $\mathrm{L}_I(v)$, i.e., the lower star of its maximum vertex according to I. Moreover we have cat state the following.

Proposition 2 *For any pair of simplices σ, τ, if $\tilde{f}(\sigma) = \tilde{f}(\tau)$ then there exist a unique vertex v such that $\{\sigma, \tau\} \in \mathrm{L}_I(v)$ (proof omitted for brevity).*

That is, if two simplices belong to the same level set, they will end up in the same indexed-based lower star.

Homotopy expansion. The last step consists in the actual pairing of cells, within each $\Lambda \in K_v$ performed by homotopy expansion.

Two simplices, say k-simplex σ and $(k+1)$-simplex τ, are paired via homotopy expansion when σ has no unpaired faces and τ has only one unpaired face (i.e. σ). The procedure `HomotopyExpansion` has no conceptual differences from the one described in [10] for the unidimensional case, except that we will work with the simplices in Λ and not on the full indexed-based lower star. Two priority queues `PQ0` and `PQ1` are used for selecting which simplex in Λ needs to be paired respectively with a higher or with a lower dimensional simplex. The priority queues are organized by listing, in lexicographic order, the tuples containing the values of I for the vertices of each simplex σ. This ensures that if $\sigma \subsetneq \tau$ in Λ, then σ takes priority over τ. While we refer to [10] for details on the algorithm, it is important to notice that also here each simplex enters a queue at most once. Thus the procedure is always linear in the number of simplices in Λ. Once collected all the gradient pairs computed on each Λ, we have that each simplex is either critical or paired with another simplex.

3.1 Analysis and complexity

The input complex is assumed to be a finite d-dimensional simplicial complex. In the reminder of this paper we indicate with $|\cdot|$ the cardinality of a set. Given a simplicial complex Σ and a simplex $\sigma \in \Sigma$, we assume that $\tilde{f}(\sigma)$ can be retrieved in linear time in the number of the vertices of σ. Moreover, we assume the boundary and the coboundary of a given simplex σ to be computed offline and stored so that it can be retrieved in constant time. We indicate with Ω the set of simplices incident into σ.

We recall that our algorithm performs three steps. During the computation of the indexing schema, the vertices are sorted according to a single component of the input function. Thus, this step takes $O(|\Sigma_0| \cdot \log |\Sigma_0|)$ operations.

To split each indexed lower star $L_I(v)$ we have to visit each simplex incident into the vertex ($O(|\Omega|)$).

In the last step `HomotopyExpansion` is called over each level set Λ. The maximal size of each queue is $|\Lambda|$,

which is bounded by $|\Omega|$. We are assuming an heap implementation for the queues so that pushing into, and removing an element from the queue, is logarithmic in the length of the queue and popping the first element out is a constant time operation. As already proved in [2, 10], each cell in Λ always enters a queue at most once. Hence, each call of `HomotopyExpansion` consists of a number of operations of order $O(|\Omega| \cdot \log |\Omega|)$.

Homotopy expansion is repeated for each level set Λ, then we can express the cost of the gradient construction as $O(|\,\Lambda\,| \cdot |\Omega| \log |\Omega|)$.

Notice that $|\Sigma_0| \leqslant |\,\Lambda\,| \leqslant |\Sigma|$ and $|\Omega|$ and $|\,\Lambda\,|$ depend on one another. Indeed, if we consider the extremal situation where each level set Λ consists of a single cell, then $|\,\Lambda\,| = |\Sigma|$. However, this would mean that Ω is empty or, in other words, there would be no actual contribution from `HomopotyExpansion`. On the other hand, if each Λ coincides with its own index-based lower star $\mathrm{L}_I(v)$, this means that $|\,\Lambda\,| = |\Sigma_0|$ and the contribution of Ω is relevant. If we consider multivariate data in real world applications where the dimension of the dataset is generally low (embedded in two or three dimensional Euclidean spaces), the number of simplices per level set (i.e., Ω) is always negligible, as well as the number of total level sets $|\,\Lambda\,|$. Thus, the complexity is dominated by the for-cycle (and thus it is linear in the number of vertices in Σ).

4 Experimental results

In this section we evaluate the advantages of our methods over the state of the art approach for computing the discrete gradient [2]. All our experiments have been performed on a dual Intel Xeon E5-2630 v4 CPU at 2.20Ghz with 64GB of RAM.

4.1 Efficiency of the index-based partition

In [2] a first algorithm has been proposed for computing a discrete gradient on multivariate data. A topological sorting is applied to the cells of Σ and pairings are created within the lower star of each cell in a sequence. We will refer to the latter as *global approach* since it is based on the construction of a global queue containing all the simplices of the dataset. This approach is inefficient since it requires all the simplices to be explicitly encoded. Moreover it does not lead to parallel implementations. The two algorithms are compared by using nine different triangle meshes having between 1.3 and 60.9 millions of simplices.

Both approaches have been implemented in C++ based on the MDG library [7]. The latter provides an efficient encoding for both a simplicial complex and the discrete gradient. We refer to [11] and to the library's documentation for further details. For the sake of this work, the use of the same data structure guarantees a fair comparison for the two approaches since the differences between them can be addressed to the algorithm solely.

Timings are shown in Figure 1(left). Our approach takes between 0.89 seconds and 4.8 minutes to finish depending on the dataset and it is generally 7 times faster than the global approach. We have also implemented a preliminary parallel version of our algorithm based on OpenMP. The latter is generally 2 to 3 times faster than the single thread approach making it, at least, 14 times faster than the global approach. Other than time efficiency, our divide-and-conquer strategy also provides a limited use of memory. The memory consumption is shown in Figure 1(right) where we are reporting the maximum peak of memory used by the two algorithms. Our approach uses at most 10 gigabytes of memory and it is twice more compact than the global approach.

4.2 Impact on the persistence module computation

In this subsection we are evaluating the impact of preprocessing a dataset by computing the discrete gradient for computing multidimensional persistent homology. By using different triangle meshes we compute the discrete vector field and the corresponding chain complex. Then we use the chain complex as input for the Topcat library [9]. Topcat is currently the only software available in public domain for computing multidimensional persistent homology on arbitrary filtrations.

Table 1 shows the results obtained by computing the persistence module of different simplicial complexes (described through their boundary matrices) or by using the boundary matrices of the chain complex G provided by the discrete gradient.

The actual limits of Topcat show up when working with relatively small simplicial complexes. The algorithm is affected by both the number of simplices and the number of steps in the filtration. We get that the program runs out of memory with filtrations having 2000 steps. The problem is drastically reduced when using G. All the runs reach the end without running out of memory. For those datasets where a comparison is possible we notice that by using G we are twice as fast and two to four times more compact.

5 Conclusions

We have presented a new approach for computing a discrete vector field V on a simplicial complex Γ and a multifiltration f, such that the chain complex computed from V shares the same MPH of Γ according to f. This is an important first step towards the development of new tools for computing MPH on real-world data. A first prototype of our tool is publicly available on GitHub (`https://github.com/IuricichF/NM_FormanMultifiltration`).

		Simpl. C.			Chain C.		
Name	$\lvert\Sigma\rvert$	Steps	Time	Mem.	Steps	Time	Mem.
Sphere	40	8x8	0.3	0.24	5x5	0.18	0.1
	244	42x42	4.4	0.86	10x10	0.28	0.2
	2884	482x482	-	-	92x89	24.3	1.5
Torus	96	16x16	0.5	0.1	9x9	0.25	0.2
	4608	768x768	-	-	65x66	7.96	2.4
	7200	1200x1200	-	-	70x80	12.05	3.0

Table 1: Timings and storage cost required for computing the persistence module on a simplicial complex (column *Simpl. C.*) and on the chain complex G computed on the corresponding discrete gradient V (column *Chain C.*)

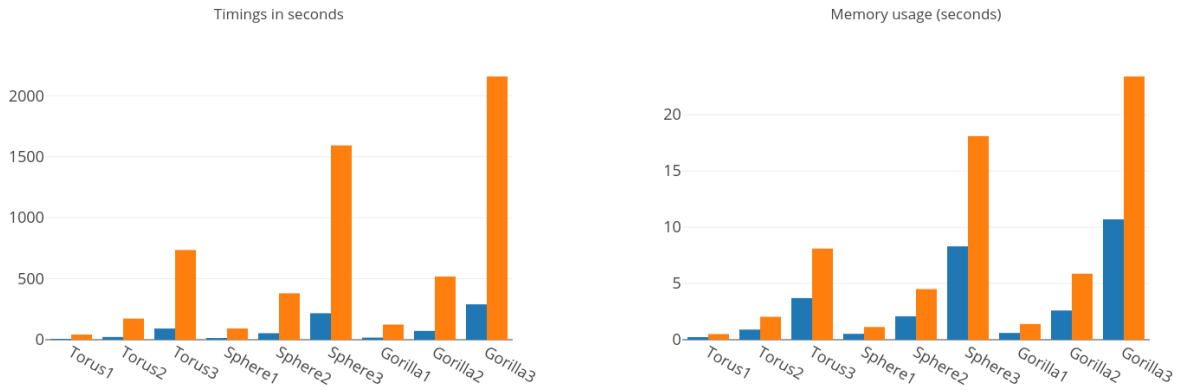


Figure 1: Comparison of timings and storage required by our algorithm (depicted in blue) and the global approach (depicted in orange)

References

[1] M. Allili, T. Kaczynski, and C. Landi. Reducing complexes in multidimensional persistent homology theory. *Journal of Symbolic Computation*, 78:61–75, 2017.

[2] M. Allili, T. Kaczynski, C. Landi, and F. Masoni. Algorithmic Construction of Acyclic Partial Matchings for Multidimensional Persistence. In W. G. Kropatsch, N. M. Artner, and I. Janusch, editors, *Discrete Geometry for Computer Imagery: 20th IAPR International Conference, DGCI 2017, Vienna, Austria, September 19 – 21, 2017, Proceedings*, pages 375–387. Springer International Publishing, Cham, 2017.

[3] G. Carlsson and A. Zomorodian. The theory of multidimensional persistence. *Discrete and Computational Geometry*, 42(1):71–93, 2009.

[4] H. Edelsbrunner and J. Harer. Persistent homology a survey. *Contemporary Mathematics*, 0000:1–26, 2008.

[5] H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics*, 9(1):66–104, 1 1990.

[6] R. Forman. Morse Theory for Cell Complexes. *Advances in Mathematics*, 134:90–145, 1998.

[7] F. Iuricich. MDG: a C++ library for computing the multidimensional discrete gradient.

[8] K. Mischaikow and V. Nanda. Morse theory for filtrations and efficient computation of persistent homology. *Discrete and Computational Geometry*, 50(2):330–353, 2013.

[9] Oliver Gäfvert. TopCat: a Java library for computing invariants on multidimensional persistence modules.

[10] V. Robins, P. J. Wood, and A. P. Sheppard. Theory and Algorithms for Constructing Discrete Morse Complexes from Grayscale Digital Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1646–1658, 2011.

[11] K. Weiss, F. Iuricich, R. Fellegara, and L. De Floriani. A primal/dual representation for discrete Morse complexes on tetrahedral meshes. *Computer Graphics Forum*, 32(3):361–370, 2013.

Parallel intersection detection in massive sets of cubes

W. Randolph Franklin
Rensselaer Polytechnic Institute
Troy, NY, USA
mail@wrfranklin.org

Salles V. G. Magalhães
Universidade Fed. de Viçosa
Viçosa, MG, Brazil
salles@ufv.br

ABSTRACT

We present ParCube, which finds the pairwise intersections in a set of millions of congruent cubes. This operation is required when computing boolean combinations of meshes or polyhedra in CAD/CAM and additive manufacturing, and in determining close points in a 3D set. ParCube is very compact because it is uses a uniform grid with a functional programming API. ParCube is very fast; even single threaded it usually beats CGAL's elapsed time, sometimes by a factor of 3. Also because it is FP, ParCube parallelizes very well. On an Nvidia GPU, processing $10M$ cubes to find $6M$ intersections, it took 0.33 elapsed seconds, beating CGAL by a factor of 131. ParCube is independent of the specific parallel architecture, whether shared memory multicore Intel Xeon using either OpenMP or TBB, or Nvidia GPUs with thousands of cores. We expect the principles used in ParCube to apply to other computational geometry problems. Efficiently finding all bipartite intersections would be an easy extension.

BigSpatial'17, November 7–10, 2017, Los Angeles Area, CA, USA

ACM ISBN 978-1-4503-5494-3/17/11…$15.00
https://doi.org/10.1145/3150919.3150921

CCS CONCEPTS

• **Theory of computation** → **Nearest neighbor algorithms**; *MapReduce algorithms*; *Computational geometry*; • **Computing methodologies** → *MapReduce algorithms*;

KEYWORDS

Parallel Programming, Computational Geometry, Intersection, Close Points, Near Points, Uniform Grid, Functional Programming, Thrust, Map-Reduce Algorithms

ACM Reference Format:
W. Randolph Franklin and Salles V. G. Magalhães. 2017. Parallel intersection detection in massive sets of cubes. In *Proceedings of BigSpatial'17:6th ACM SIGSPATIAL Workshop on Analytics for Big Geospatial Data , Los Angeles Area, CA, USA, November 7–10, 2017 (BigSpatial'17),* 1 pages.
https://doi.org/10.1145/3150919.3150921

NOTE

This paper will be presented at BIGSPATIAL. As allowed by ACM, a copy is available at https://wrf.ecse.rpi.edu/p/224-parcube-bigspatial-2017.pdf.

Dynamic Orthogonal Range Seaching on the RAM, Revisited

Timothy M. Chan[1] and Konstantinos Tsakalidis[2]

[1]Department of Computer Science, University of Illinois at Urbana-Champaign
[1]Department of Computer Science and Engineering, Tandon School of Engineering, New York University

Abstract

We study a longstanding problem in computational geometry: 2-d dynamic orthogonal range reporting. We present a new data structure achieving $O\left(\frac{\log n}{\log\log n} + k\right)$ optimal query time and $O\left(\log^{2/3+o(1)} n\right)$ update time (amortized) in the word RAM model, where n is the number of data points and k is the output size. This is the first improvement in over 10 years of Mortensen's previous result [*SIAM J. Comput.*, 2006], which has $O\left(\log^{7/8+\varepsilon} n\right)$ update time for an arbitrarily small constant ε.

In the case of 3-sided queries, our update time reduces to $O\left(\log^{1/2+\varepsilon} n\right)$, improving Wilkinson's previous bound [ESA 2014] of $O\left(\log^{2/3+\varepsilon} n\right)$.

This work appeared in the Proceedings of the 33rd International Symposium on Computational Geometry (SoCG 2017). Link to full version

Faster Algorithm for Truth Discovery via Range Cover*

Ziyun Huang[1] Hu Ding[2] Jinhui Xu[1]

[1] Department of Computer Science and Engineering
State University of New York at Buffalo
{ziyunhua, jinhui}@buffalo.edu
[2] Department of Computer Science and Engineering
Michigan State University
huding@msu.edu

1 Overview

Truth discovery is an important problem arising in data analytics, and has received a great deal of attentions in recent years in the fields of data mining, database, and big data [3, 6, 7, 4, 8–10]. Truth discovery seeks to find trustworthy information from a dataset acquired from a number of sources which may contain false or inaccurate information. There are numerous applications for this problem. For example, the latest search engines are able to answer user queries directly, instead of simply listing webpages that might be relevant to the query. This process involves retrieving answers from potentially a large number of related webpages. It is quite common that these webpages may provide inaccurate or inconsistent information. Thus a direct answer to the query needs the search engine to be able to extract the most trustworthy information from all these webpages, which is exactly the problem of truth discovery.

Truth discovery is an unsupervised learning problem. Besides the input data, no prior knowledge about the reliability of each data source is provided. In such settings, an intuitive approach is to view all data sources equally reliable and obtain the solution by averaging or majority rule. A major issue of this approach is that the yielded answer may be quite far away from the truth. This is because a small number of unreliable data sources could significantly deviate the final solution. To deal with this issue, truth discovery treats data sources differently by estimating the reliability for each of them. This greatly increases the level of challenge for the problem. Moreover, since the truth discovery problem often occurs in big data scenarios, the number of data sources could be quite large and the dimensionality of the data could be rather high, which brings another dimension of challenge to the problem.

A widely accepted geometric modeling of the truth discovery problem is the follows. Data from each source is formulated as a set of real number attributes, and thus can be viewed as a vector in $\mathbb{R}^d$, where d is the number of attributes.

* A preliminary version of this work has appeared in the 15th Algorithms and Data Structures Symposium, WADS 2017.

Each data source is associated with a positive variable (or weight) representing its reliability. Formally, the truth discovery problem can be defined as follows.

Definition 1. *(Truth Discovery [4, 7]). Let $P = \{p_1, p_2, \dots p_n\}$ be a set of points in $\mathbb{R}^d$ space, where each p_i represents the data acquired from the i-th source among a set of n sources. The truth discovery problem is to find the truth vector p^* and w_i (*i.e., *reliability) for each i-th source such that the following objective function is minimized.*

$$\min \Sigma_{i=1}^{n} w_i \|p_i - p^*\|^2, s.t.\ \Sigma_{i=1}^{n} e^{-w_i} = 1. \tag{1}$$

The meaning of the above truth discovery formulation was discussed in [1] from an information theory's point of view. It is shown that the constraint on w_i in Definition 1 ensures that the **entropy** is minimized when p^* approaches the truth vector. For this reason, the problem is also called *Entropy based Geometric Variance* problem [1].

Despite extensive studies on this problem, most of the existing techniques are of heuristic nature, and do not provide any guarantee on the quality of solution. It is not until very recently that the truth discovery problem has a theoretically guaranteed solution [1]. This result ensures that a $(1+\epsilon)$-approximation of the problem can be achieved in $O(dn^2 + (n\Delta)^\sigma nd)$ time, where n is the number of input points (*i.e.,* data sources), d is the dimensionality of the space, Δ is the spread ratio of the input points (*i.e.* the ratio of the largest distance between any two input points to the smallest distance), and σ is any fixed small positive number. The result is based on an elegant sampling technique called Simplex Lemma [2] which is capable of handling high dimensional data. A main issue of this method is that its running time depends on the spread ratio of the input points, and is polynomial only when the spread ratio is relatively small (*i.e.,* $\Delta = O(\sqrt{n})$). This could severely restrict its applicability.

To overcome this main issue, we present in this paper a faster algorithm for the truth discovery problem. With constant probability, our algorithm achieves a $(1+\epsilon)$-approximation in $O(dn^2(\log n + \log d))$ time, and is completely independent of the spread ratio. Our algorithm is also space efficient, using only near linear space, while the space complexity of [1] also depends on the spread ratio. Our algorithm relies on a new data structure called *range cover*, which is interesting in its own right. Roughly speaking, range cover is a data structure designed for a class of optimization problems (in high dimensional space) which are decomposable into a number of "easier" cases, where each case can be characterized by a parameterized assumption. For example, truth discovery can be formulated as a problem of finding a truth vector $p^* \in \mathbb{R}^d$ from a given set P of points in $\mathbb{R}^d$ so that a certain objective function (the exact formulation will be discussed later) is minimized. We are able to show that although directly optimizing the objective function is challenging, the problem is much easier to solve if some additional information (*e.g.,* the distance r between p^* and P) is known. Thus, by viewing the additional information as a parameterized assumption, we can solve the truth discovery problem by searching for the best assumption. The range cover data structure shows that even though the number of parameterized

assumptions could be very large (or even infinite), it is sufficient to sample only a small number of assumptions to ensure an approximate solution. This leads to a small-size data structure (*i.e.,* $O(n \log n)$ space) and a faster algorithm for truth discovery. Since the idea of decomposing problem into cases is not restricted only to the truth discovery problem, we expect that this data structure will provide new approaches to other problems.

2 Range Cover Data Structure

In this section, we describe our main technique. the range cover data structure.

Range cover is motivated by several high dimensional optimization problems (such as truth discovery). In these problems, an input point set P is given in $\mathbb{R}^d$ space, and the objective is to find a point p^* in $\mathbb{R}^d$ so that a certain objective function is optimized. We would like to characterize all possibilities of p^* into a small number of cases so that in each case p^* is associated with a certain parametrized assumption which could help solve the problem more efficiently. For instance, in some optimization problem, p^* could be much easier to obtain if we know in advance the nearest neighbor (say p) of p^* in P and its distance r to p^* (*i.e.,* $\|p - p^*\| = r$) for some parameter r. We expect that these parameterized assumptions form a space with much lower dimensionality than d, and thus the overall time complexity is low.

In this work, we develop an algorithm to generate a collection $\mathcal{A}$ of parameterized assumptions about the truth vector p^*, given input point set P. Each of the item from $\mathcal{A}$ is denoted in the form of either $\mathcal{NN}_{p^*}(v, r)$ or $\mathcal{DOM}_{p^*}(v)$, which has the following meaning:

Assumption 1 *$\mathcal{NN}_{p^*}(v, r)$: For a subset v of P, $\mathcal{NN}_{p^*}(v, r)$ is an assumption made about p^* which says: $D(v) \leq \lambda r$ for some small constant $\lambda > 0$, where $D(v)$ is the diameter of v, and $r \leq \|p' - p^*\| \leq (1 + \lambda)r$ holds for p' which denotes the nearest neighbor of p^* in v.*

Assumption 2 *$\mathcal{DOM}_{p^*}(v)$: For a subset v of P, $\mathcal{DOM}_{p^*}(v)$ is an assumption made about p^*, which says: there exists a point $p_v \in v$ such that $D(v) \leq \lambda \|p^* - p_v\|$ and $\|p_v - p^*\| \leq \xi \|p_{-v} - p^*\|$ for any point $p_{-v} \in P \setminus v$, where $D(v)$ is the diameter of v.*

Intuitively, $\mathcal{NN}_{p^*}(v, r)$ assumes that p^* has approximate nearest neighbor set v, and $\mathcal{DOM}_{p^*}(v)$ assume that p^* is very close to points in a subset v of P. The collection $\mathcal{A} = \{\mathcal{DOM}_{p^*}(v_1), \mathcal{DOM}_{p^*}(v_2), \ldots, \mathcal{DOM}_{p^*}(v_h), \mathcal{NN}_{p^*}(v'_1, r_1), \mathcal{NN}_{p^*}(v'_2, r_2) \ldots, \mathcal{NN}_{p^*}(v'_g, r_g)\}$ provide a complete coverage for all possible candidate $p^* \in \mathbb{R}$: for any $p^* \in \mathbb{R}$, at least one of the assumptions listed in $\mathcal{A}$ holds. Thus, if for every case in $\mathcal{A}$, we compute a candidate truth vector by solving the problem as if the case actually holds, and combine the results together by trying all the computed truth vectors and pick the one that minimize the objective function, we obtain the result for the whole problem. Also the size of $\mathcal{A}$ is only $O(n \log n)$, making our scheme efficient.

3 Solving All the Cases

We describe briefly how to solve the truth discovery problem under assumption $\mathcal{NN}_{p^*}(v, r)$ or $\mathcal{DOM}_{p^*}(v)$ for arbitrary v and r.

Solving $\mathcal{DOM}_{p^*}(v)$. If this assumption holds, it is shown that one of the input point in v will be the $(1+\epsilon)$-approximate solution for the truth discovery problem. Thus we may try all the input points as candidates of the truth vector candidate. All cases of the form $\mathcal{DOM}_{p^*}(v)$ in $\mathcal{A}$ are considered by doing this. It takes $O(dn^2)$ time to try all the input points and solves these cases.

Solving $\mathcal{NN}_{p^*}(v, r)$. We use v as the approximate location of p^*, and obtain the approximate optimal weights of input points. The approximated weights combined with the techniques used in [1] enable us to solve the problem efficiently. For each $\mathcal{NN}_{p^*}(v, r)$, it takes $O(nd)$ time to compute the candidate truth vector.

References

1. Ding, H., Gao, J., and Xu, J.: Finding Global Optimum for Truth Discovery: Entropy Based Geometric Variance. Leibniz International Proceedings in Informatics (LIPIcs), 32nd International Symposium on Computational Geometry (SoCG 2016), Vol. 51, 34:1-34:16(2016).
2. Ding, H. and Xu, J.: A Unified Framework for Clustering Constrained Data without Locality Property. Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA 2015), pp. 1471-1490, January 4-6, 2015, San Diego, California, USA.
3. Dong, X.L., Berti-Equille, L., Srivastava, D.: Integrating conflicting data: The role of source dependence. PVLDB, 2(1): 550-561(2009).
4. Li, Y., Gao, J., Meng, C., Li, Q., Su, L., Zhao, B., Fan, W., Han, J.: A Survey on Truth Discovery, CoRR abs/1505.02463(2015).
5. Har-Peled, S.: Geometric approximation algorithms. Vol. 173. Boston: American mathematical society(2011).
6. Li, H., Zhao, B., Fuxman, A.: The Wisdom of Minority: Discovering And Targeting The Right Group of Workers for Crowdsourcing. Proc. of the International Conference on World Wide Web (WWW'14), pp. 165-176(2014).
7. Li, Q., Li, Y., Gao, J., Zhao, B., Fan, W., Han, J.: Resolving Conflicts in Heterogeneous Data by Truth Discovery and Source Reliability Estimation. Proc. the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD'14), pp. 1187-1198(2014).
8. Pasternack, J., Roth, D.: Knowing what to believe (when you already know something). Proc. of the International Conference on Computational Linguistics (COLING'10), pp. 877-885(2010).
9. Whitehill, J., Ruvolo, P., Wu, T., Bergsma, J., Movellan, J.: Whose Vote Should Count More: Optimal Integration of Labelers of Unknown Expertise. Advances in Neural Information Processing Systems (NIPS'09), pp. 2035-2043(2009).
10. Yin, X., Han, J., and Yu, P.S.: Truth discovery with multiple conflicting information providers on the web: Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'07), pp. 1048-1052(2007).

An Efficient Sum Query Algorithm for Distance-based Locally Dominating Functions*

Ziyun Huang[1] Jinhui Xu[1]

Department of Computer Science and Engineering
State University of New York at Buffalo
{ziyunhua, jinhui}@buffalo.edu

1 Overview

In this paper, we consider the following *sum query* problem: Given a set P of points in $\mathbb{R}^d$ (where the dimensionality d could be very high) and a function $f(,)$, the sum query problem is to build a data structure for P so that the sum of $\sum_{p\in P} f(p,q)$ can be efficiently computed or approximated for any query point q in $\mathbb{R}^d$, where $f(p,q)$ is a non-negative *distance-based* function. We say that $f(p,q)$ is *distance-based* if the value of $f(p,q)$ depends only on the distance between p and q. In other words, $f(p,q)$ can be written as $F(\|p,q\|)$ for some non-negative real function $F(\cdot)$.

The distance-based sum query problem are frequently encountered in many applications. A good example is the well known 1-median problem: given a point set P in $\mathbb{R}^d$, find a point q such that the objective value $C(q) = \sum_{p\in P}\|q-p\|$ is minimized. $C(q)$ is clearly an example of the distance-based sum query problem (with respect to the to-be-determined median point q), where each term of the summation is trivially the Euclidean distance $\|q-p\|$ of p and q. The sum query problem also appears in many other real world applications. For example, the problem of computing the illumination intensity of a given point can be viewed as a sum query problem. In such an application, the intensity of the query point may jointly be determined by the total amount of light received from multiple light sources. The light contributed by each source is inversely proportional to its squared distance to the given point (*i.e.* obeying the inverse squared distance law in physics). Note that in this case the distance-based functions may be different for each light source, depending on its base intensity. However, if we view a light source with base intensity w as a collection of w light sources with "unit" intensity located at the same place, we may still treat the intensity as a purely distance-based function.

Several previous results are closely related to some versions of the problem considered in this paper. They are mainly based on some *core-set* techniques [2, 6, 9]. In the 1-median problem, for example, a core-set of a point set P in $\mathbb{R}^d$ is a small-size (weighted) subset of P such that for any $q \in \mathbb{R}^d$, the sum $\sum_{p\in P}\|p-q\|$ can be approximated by just inspecting the distances between q

* A preliminary version of this work has been accepted to the 28th International Symposium on Algorithm and Computation (ISAAC 2017).

and points in the core-set. In general, a core set of P with respect to a function $f(p,q)$ is a small subset of P such that for any q, $\sum_{p\in P} f(p,q)$ can be estimated by using only the information of the points in the core-set. For functions $f(p,q)$ satisfying certain properties, it is possible to construct a core-set for any point set P efficiently [7].

In this paper, we aim to develop an efficient algorithm for supporting distance-based functions that have *local domination property* [5], which means that $f(p,q)$ can be very large when $\|p-q\|$ is small. For example, a distance-based function obeying the inverse squared distance law (*i.e.* $f(p,q) = w/\|p-q\|^2$ for some constant w), is a function having such a property. While the aforementioned core-set method is useful for a large family of functions $f(p,q)$, it does not directly apply to functions which have local domination property. This is because the $\sum_{p\in P} f(p,q)$ could become infinitely large when q approaches any point in P, which means that any "traditional" core-set of P will fail if the core-set is a proper subset of P.

The local domination property imposes additional challenges to the sum query problem. Particularly, it requires the query algorithm to be able to detect points that are close to the query points. This means that the algorithm should have certain ability for proximity search. However, in high dimensional space, highly accurate nearest neighbor search cannot be done very efficiently. Well-known techniques for high dimensional nearest neighbor search, such as the Locality Sensitive Hashing (LSH) [8], require almost linear time to achieve a c-approximate nearest neighbor when c is close to 1 [3]. Thus, for the sum query problem, we are required to develop an estimation algorithm with high accuracy, but not allowed to use the high accuracy proximity search techniques.

To deal with the additional challenge caused by the local domination property, we first assume that the distance function F satisfies the following local domination implied properties.

1. $F(\cdot)$ is positive and $F(0)$ could be infinite.
2. $F(\cdot)$ is monotonously decreasing. [1]
3. For any constant $\lambda \geq 1$, there exists a constant $\Delta(\lambda) \geq 1$, such that $F(x) \leq \Delta(\lambda)F(x\lambda)$ for any $x \geq 0$.

It is worth noting that although our technique is designed for functions with local domination property, it actually works for any distance-based non-negative functions. Particularly, our approach is capable of solving the "inverse" version of the problem, where $F(\cdot)$ is a monotonically increasing function satisfying some accordingly changed conditions. Since other types of distance-based functions have already been studied in [7], we focus our investigation on locally dominating functions in this paper.

Our Result: Our main result for the sum query problem is a novel scheme based on some sampling and searching techniques, and is capable of reporting

[1] Indeed this restriction can be greatly soften. Our scheme applies as long as $F(\cdot)$ is "not increasing rapidly", *i.e.*, $F(x_1) \leq CF(x_2)$ for some constant C when $x_1 > x_2$. The listed restriction is mainly for ease of presentation.

a $(1+\epsilon)$-approximation for each sum query ($\sum_{p\in P} f(p,q) = F(\|p-q\|)$) in $\tilde{O}_{\epsilon,d}(n^{0.5+c})$ time with success probability at least $1-1/n$ for any $c>0$. The query algorithm makes use of a soft boundary range reporting data structure to determine a number of points that are among the closest to the query point q. The soft boundary range reporting data structure can be computed within $\tilde{O}_{\epsilon,d}(n^{1+c})$ time for any $c>0$. The hidden constants in the time complexities depend only polynomially on d and $1/\epsilon$. The error factor ϵ can be very small and is assumed to be within the range of $[8/\sqrt{n}, 1)$. One major advantage of our scheme is that the query algorithm runs much faster than the best existing $(1+\epsilon)$-approximate nearest neighbor search technique (which takes almost linear time) in high dimensional space for small enough ϵ .

1.1 Our Techniques

Our query algorithm consists of 2 main steps.

First step. We identify a number of points P_Ω that are among the closest to the query point q, and compute directly their contributions to the sum $\sum_{p\in P_\Omega} f(p,q)$. We use a soft boundary range reporting data structure to identify points that are among the closest to q. Our soft boundary range reporting scheme solves a different version of range reporting problem than traditional formulations: ideally we would like to report all the $O(\sqrt{n})$ closest points in P to q, but for performance consideration, we soften our requirement so that, we only need to guarantee the points that lies within distance $\Omega(r_\Omega)$ to q are reported, where r_Ω denotes the distance from q to its $O(\sqrt{n})$-th closest point in P.

To solve the range reporting problem, we reduce the reporting problem to a number of nearest neighbor search queries. Each nearest neighbor query is performed on a sampled subset of P, which is produced in a preprocessing procedure. With properly chosen parameters, we are able to show that it suffices to use a relatively low quality approximate range search procedure to obtain an accurate solution. The step of the query takes $\tilde{O}(dn^{1+\alpha})$ time for any $\alpha>0$. The size of the reported set P_Ω is at most $O(\sqrt{n})$. It takes $\tilde{O}(dn^{1+\alpha})$ time to preprocess P and build the data structure for this type of range reporting.

Second step. we sample, from the rest of the points in P, a small subset of points to estimate their contributions to the sum. Intuitively speaking, since we have already identified a number of points that have the largest contribution to the sum before sampling, the error incurred by sampling the rest of points is relatively small and thus controllable. The intuition can be proved by an involved argument with the use of some concentration inequalities. This step is also efficient since we only need to sample $\tilde{O}(\sqrt{n})$ points.

Finally, we combine the results from the first and the second steps to obtain an approximate final solution.

1.2 Related Works

As mentioned earlier, the sum query problems can be solved by using core-sets for distance functions satisfying some "nice" properties. Our work can be viewed

as a complement to those core-set results as it addresses a rather general case that is hard to solve by using core-sets.

Our scheme makes use of some ideas from range search and top-k indexing. There are a number of previous results on both problems [11, 10, 4, 1]. Many of them are not the best fit, especially in high dimensional space, as they cannot be directly applied to our problem. The special property of our problem enables us to develop a range search scheme with better performance.

References

1. Peyman Afshani and Timothy M. Chan. Optimal halfspace range reporting in three dimensions. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 180–186. SIAM, 2009.
2. Pankaj K. Agarwal, Sariel Har-peled, and Kasturi R. Varadarajan. Geometric approximation via coresets. In *Combinatorial and Computational Geometry, MSRI*, pages 1–30. University Press, 2005.
3. Alexandr Andoni, Piotr Indyk, Huy L Nguy˜ên, and Ilya Razenshteyn. Beyond locality-sensitive hashing. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1018–1028. SIAM, 2014.
4. Boris Aronov and Sariel Har-Peled. On approximating the depth and related problems. *SIAM Journal on Computing*, 38(3):899–921, 2008.
5. D. Z. Chen, Z. Huang, Y. Liu, and J. Xu. On clustering induced voronoi diagrams. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 390–399, 2013.
6. Ke Chen. On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications. *SIAM Journal on Computing*, 39(3):923–947, 2009.
7. Dan Feldman and Michael Langberg. A unified framework for approximating and clustering data. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 569–578. ACM, 2011.
8. Sariel Har-Peled, Piotr Indyk, and Rajeev Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory of computing*, 8(1):321–350, 2012.
9. Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 291–300. ACM, 2004.
10. Saladi Rahul and Yufei Tao. Efficient top-k indexing via general reductions. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 277–288. ACM, 2016.
11. Cheng Sheng and Yufei Tao. Dynamic top-k range reporting in external memory. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems*, pages 121–130. ACM, 2012.

Approximate Convex Hull of Data Streams

Avrim Blum*
TTI-Chicago
avrim@ttic.edu

Vladimir Braverman†
Johns Hopkins University
vova@cs.jhu.edu

Ananya Kumar‡
Carnegie Mellon University
skywalker94@gmail.com

Harry Lang§
Johns Hopkins University
hlang8@math.jhu.edu

Lin F. Yang¶
Princeton University
lin.yang@princeton.edu

October 27, 2017

Abstract

Given a finite set of points $P \subseteq \mathbb{R}^d$, we would like to find a finite subset $S \subseteq P$ such that the convex hull of S approximately covers the original set. More formally, every point in P is within distance ϵ from the convex hull of S. Such a subset is also called an ϵ-hull or an ϵ-generating points set for P. An ϵ-hull is called optimal if it has minimum cardinality. Computing such a point set gives a sparse dictionary for the original points and is an important problem in computational geometry, machine learning, and approximation algorithms.

In many real world applications, the set P is too large to fit in memory. Streaming algorithms that use sublinear memory would provide excellent solutions to such situations. Existing streaming algorithms for this problem give bounds that only depend on ϵ but ignore the structure of the data. A natural question is whether we can do better than state-of-the-art when the data is well-structured, in particular when the optimal generating set is small.

Unfortunately, our lower bounds for this problem show that it is still hard even if the data is well-structured. We then propose two interesting relaxations of the problem, in which tighter input-dependent bounds are possible. In the first relaxation, we consider data in $\mathbb{R}^2$ that is randomly permuted before the algorithm runs. We give an algorithm that stores only $O(\texttt{opt} \cdot \log n)$ points while maintaining an ϵ-hull, where opt is the size of an optimal one. In the second relaxation, our approximation only needs to be correct in most directions. We provide the first, nearly tight, input-dependent upper bound for this case even for $\mathbb{R}^d$ with arbitrary constant dimension d.

1 Introduction

Let P be a set of n points in the unit ball $\mathcal{S}^{d-1}$. Let $\mathcal{C}_P$ denote the convex hull of the point set P. An important question to ask is that whether it is possible to obtain a small subset $S \subset \mathbb{R}^d$ such that the Hausdorff distance between $\mathcal{C}_P$ and $\mathcal{C}_S$ is small, i.e., for every point $p \in P$, p is close to a point in $\mathcal{C}_S$ and vice versa. We call such an S an *ϵ-hull* to P (with respect to the convex hull). This question and its variants are fundamental in computational geometry, computer vision, data mining, and many more. Such a set S is called an ϵ-coreset or kernel for P, e.g., [AHPV04, AHPV05]. Since every point in P can be approximated by the convex combination of points in S, it is also called a generating set [BHR15]. Many different variants of this problem have been studied in the literature, e.g., one may require that any directional width (the diameter of S in a particular direction) of S is a $(1 \pm \epsilon)$ approximation to that of P. There are subtle differences between these variants, but one can usually change from one variant to another without much effort. For more details, please refer to [AHPV05].

The worst case lower bound for the size of S is $\Omega(\epsilon^{-(d-1)/2})$. Recently, it has been shown in [BHR15] that one can in fact do much better than the worst case bound if the size opt of the smallest generating set for P is small. In their paper, they show that one can efficiently obtain S of size nearly linear in opt and at most linear in the dimension d. This result shows the possibility of overcoming the curse of dimensionality in high-dimensional computational geometry problems.

*This work was supported in part by the National Science Foundation under grant CCF-1525971. Work was done while the author was at Carnegie Mellon University.

†This material is based upon work supported by NSF Grants IIS-1447639, EAGER CCF-1650041, and CAREER CCF-1652257.

‡Now at DeepMind.

§This research was supported by the Franco-American Fulbright Commission. The author thanks INRIA (l'Institut national de recherche en informatique et en automatique) for hosting him during the writing of this paper.

¶This material is based upon work supported by the NSF Grant IIS-1447639. Work was done while the author was at Johns Hopkins University.

One concern of the algorithms in [BHR15] is that they require storing all points of P in memory. The huge size of real-world datasets limits the applicability of these algorithms. It is thus a natural question to ask whether it is possible to obtain an ϵ-hull of P, when P is presented as a data stream, while using a small amount of memory. That is, is there an efficient streaming algorithm to obtain the similar guarantees as in [BHR15]?

In this paper, we provide both negative and positive results. First, we review previous results on this problem.

1.1 Related Work

Batch Algorithms for ϵ-kernels We use the term batch algorithm for an algorithm that stores the entire set of points in memory. In the batch setting, Bentley, Preparata, and Faust [BPF82] give a $O(1/\epsilon^{d-1})$ space algorithm for computing an ϵ-hull of a set of points. Agarwal, Har-Peled, and Varadarajan [AHPV04] use this to give a $O(1/\epsilon^{d-1})$ space algorithm for computing multiplicative approximation of convex hulls. This was improved to a $O(1/\epsilon^{\frac{d-1}{2}})$ space algorithm [YAPV04, Cha06] for computing an ϵ-hull. The time bounds on these algorithms were further improved [Cha17, SA17]. Recently, Blum, Har-Peled, and Raichel [BHPR16] give the only known batch algorithms for an ϵ-hull that are competitive with the optimal of the given point set.

Streaming Algorithms for ϵ-kernels with Worst Case Guarantees Hershberger and Suri [HS04] give a 2-D one-pass streaming algorithm for ϵ-hulls that uses $O(1/\sqrt{\epsilon})$ space. Agarwal, Har-Peled, and Varadarajan [AHPV04] give a one-pass streaming algorithm for ϵ-kernels (which is an multiplicative error version of the ϵ-hull) that uses $(1/\epsilon^{\frac{d-1}{2}})\log^d n$ space. Chan [Cha06] removes the dependency on n and gives a streaming algorithm for ϵ-kernels that uses $O((1/\epsilon^{d-3/2})\log^d 1/\epsilon)$ space. This was then improved to $O((1/\epsilon^{\frac{d-1}{2}})\log\frac{1}{\epsilon})$ [ZZ11] and the time complexity was further improved by Arya and Chan [AC14]. Chan [Cha16] also gives a dynamic streaming (allowing deletions) algorithm based on polynomial methods.

1.2 Our Contributions

In Section 3, we provide lower bounds to show that no streaming algorithm can achieve space bounds comparable to $\texttt{opt}$, the optimal size of an ϵ-hull. In particular, no streaming algorithm can have space complexity competitive with $f(\texttt{opt})$ for any fixed positive function f in 3 dimensions or higher. This strong negative result implies only relaxations of the problem in the streaming model are possible to achieve input-dependent bounds.

We devise and prove streaming algorithms for two relaxations of the problem. In Section 4, we show the first relaxation, in which the points are from $\mathbb{R}^2$ and come in a random order. In Section 5, we show the second relaxation, in which the points come in an arbitrary order and from d-dimensional space.

In the first relaxation, our algorithm maintains an initially empty point set S. When our algorithm sees a new point p, it adds p to S if p is at least distance ϵ away from the convex hull of S. Additionally, our algorithm keeps removing points $p \in S$ where p is contained inside the convex hull of $S \setminus \{p\}$, that is, removing p does not change the convex hull of S. Surprisingly, for any point stream P, with high probability this algorithm keeps $O(\texttt{opt} \cdot \log n)$ points, where n is the size of P.

In the second relaxation, we only need to be correct in "most" directions (all but a δ fraction of directions). Our algorithm picks $O(\frac{\texttt{opt}^2}{\delta^2}\log\frac{\texttt{opt}}{\delta})$ random unit vectors. For each of these vectors v, we keep the point in the stream that has maximal dot product with v. We give a proof based on VC-dimension to show that this algorithm achieves the desired bound. For the 2D case we achieve an even stronger bound.

Although our algorithms are simple, it is surprising that input-dependent bounds are achievable in these settings. To the best of our knowledge, this is the first work that gives streaming algorithms for ϵ-hulls with space complexity comparable to the optimal approximation.

2 Preliminaries

Definition 2.1. For any bounded set $C \subset \mathbb{R}^d$, we say a point q is ϵ-close to C if $\inf_{x\in C}\|q - x\| \le \epsilon$.

Definition 2.2. Given a set of points $P \subseteq \mathbb{R}^n$, $S \subseteq P$ is an *ϵ-hull* of P if for every $p \in P$, p is ϵ-close to the convex hull of S.

(a) Set of points (b) ϵ-hull in red

Figure 1: ϵ-hull of a set of points.

Definition 2.3. Let $\texttt{opt}(P, \epsilon)$ denote the size of a (not necessarily unique) smallest ϵ-hull of P.

2.1 Streaming Model

In our model, a streaming algorithm $\mathcal{A}$ is given $\epsilon \in (0, 1)$ in advance but not the size of the input point stream P. More formally, we denote the streaming point stream P as a sequence of points

$$P = (p_1, p_2, \ldots, p_t, \ldots)$$

where $p_t \in \mathbb{R}^d$ is the point coming at time t. Note that P may have duplicate points. Algorithm $\mathcal{A}$ is given the points in P sequentially, and is required to maintain a subset $S \subset P$ of the points. For each point $p \in P$, $\mathcal{A}$ can choose to add p to S (remembering p) or ignore p (therefore permanently forgetting p). $\mathcal{A}$ can also choose to delete points in S, in which case these points are permanently lost. After one-pass of the stream, we require S to be an ϵ-hull of the points set P. A trivial streaming algorithm could just keep all points it has seen. However, such an algorithm might not be feasible in the big data regime. Ideally, $\mathcal{A}$ should use space competitive with $\texttt{opt}(P, \epsilon)$.

3 Lower Bounds

3.1 Always-opt Lower Bounds

An always-opt algorithm uses space competitive with $f(\texttt{opt}(P, \epsilon))$ for some fixed positive function f at any time t. Note that this definition is rather permissive, since it allows an arbitrary function of opt, and allows the algorithm to maintain an $r\epsilon$ hull where $r > 1$.

Definition 3.1. For $r \in \mathbb{Z}^+$, we say a streaming algorithm $\mathcal{A}$ is *always-r-opt* if there exists a function $f : \mathbb{Z}^+ \to \mathbb{Z}^+$ such that: if $\mathcal{A}$ is run on an arbitrary point stream P, then after processing all points in P, $\mathcal{A}$ keeps an $r\epsilon$-hull of P with size at most $f(\texttt{opt}(P, \epsilon))$.

Theorem 3.1. *For all $r \in \mathbb{Z}^+$, there does not exist an always-r-opt streaming algorithm in $\mathbb{R}^n$ for $n \geq 3$.*

Proof. See appendix A. □

3.2 Sometimes-opt Lower Bounds

We can ask a slightly different question: what if an algorithm is given a number k in advance, and only needs to be competitive with opt when opt of the current substream falls below k? The algorithm we give for (ϵ, δ)-hulls in Section 5 is of this form.

Definition 3.2. A streaming algorithm $\mathcal{A}$ is *sometimes-opt* if there exists a function $f : \mathbb{Z}^+ \to \mathbb{Z}^+$ such that the following holds. Suppose $\mathcal{A}$ is given $k \in \mathbb{Z}^+$ in advance, and is run on an arbitrary point stream P with $\texttt{opt}(P, \epsilon) \leq k$. At any point, $\mathcal{A}$ is allowed to keep at most $f(k)$ points. After processing all points in P, $\mathcal{A}$ keeps an ϵ-hull of P.

Theorem 3.2. *There does not exist a sometimes-opt streaming algorithm in $\mathbb{R}^n$ for $n \geq 3$.*

Proof. See appendix A. □

These lower bounds rule out the possibility of obtaining ϵ-hulls in the streaming model while only using space comparable to the optimal approximation size. We thus seek to relax the problem under reasonable assumptions.

4 2D Random Order Algorithm

In many cases, data points are generated from independent identical distributions, for example mixture models or topic models. In this section we assume a more general setup: that the points come in a random order. More precisely, for all sets of points P, every permutation of P must have equal probability density. The case where the data points are generated i.i.d. (*making no assumptions about the distribution*) is a special case. We assume the points are in 2D.

Definition 4.1. We say a point p is interior to P if p is in the convex hull of $P \setminus \{p\}$.

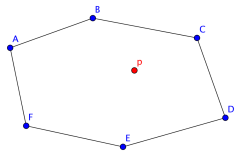
The algorithm begins by keeping a set $S = \{\}$. For each point $p \in P$ that the algorithm sees, if the distance from p to the convex hull of S is at most ϵ, we discard p. Otherwise, we add p to S (see figure 2).

The algorithm then repeatedly deletes interior points in S until S has no interior points (see figure 3). It is easy to show that S is always an ϵ-hull, but showing the space bound is more challenging.

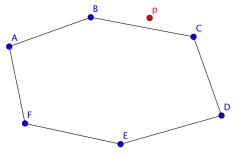
Theorem 4.1. *There exists a constant $c > 0$ and a deterministic one-pass streaming algorithm $\mathcal{A}$, such that for any random order input streams $P \subseteq \mathbb{R}^2$ containing at most n points, $\mathcal{A}$ maintains a subset $S \subseteq P$ which is an ϵ-hull of P at all times. Moreover, with probability at least $1 - 1/n^2$,*

$$|S| \leq c \cdot \textit{opt}(P, \epsilon) \cdot \log n.$$

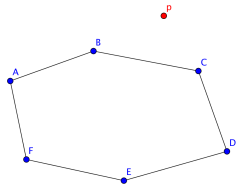
Since the algorithm is deterministic, this probability guarantee is over the randomness in the arrival order of P.



(a) Discard p inside hull

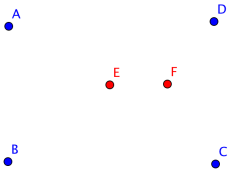


(b) Discard p near hull



(c) Keep p far from hull

Figure 2: We keep points p far from the current hull.



(a) Interior points in red

(b) No interior points

Figure 3: We iteratively remove interior points from S.

Proof. For the full proof, see Appendix B - here we give a sketch. Consider the optimal ϵ-hull $S_{\texttt{opt}}$. Since our algorithm deletes interior points, we can show it only keeps points close to the boundary of the convex hull of $S_{\texttt{opt}}$. We split the boundary into $\texttt{opt}$ sections, and show that with high probability our algorithm keeps $O(\log n)$ points in each section. Since the boundary is 1-dimensional, we effectively reduce proving the 2D result, to proving an analogous 1D result for our algorithm. □

A corollary of the high probability bound is that in expectation the algorithm keeps an ϵ-hull of size $O(\texttt{opt} \cdot \log n)$.

5 (ϵ, δ)-Hull

In this section we give an algorithm for a relaxation of ϵ-hulls, which we call (ϵ, δ)-hulls. Our results hold for arbitrary point sets $P \subseteq \mathbb{R}^d$. Intuitively, an (ϵ, δ)-hull of P is within distance ϵ from the boundary of the convex hull of P in at least $1 - \delta$ directions.

Definition 5.1. Given a vector v and a point set P, we define the **directional extent** as

$$\omega_v(P) = \max_{p \in P} p \cdot v$$

Definition 5.2. If p is a point we define $\omega_v(p) = p \cdot v = \omega_v(\{p\})$

Definition 5.3. We say a set S **maximizes** P in v if

$$\omega_v(P) = \omega_v(S)$$

Note that as per Definition 5.2, S can be either a single vector or a set of vectors.

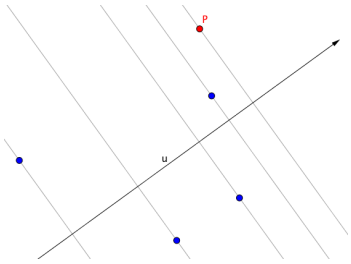


Figure 4: Point p maximizes the set of points in direction u because its projection onto u is the highest.

Definition 5.4. An (ϵ, δ)**-hull** is the minimum sized set $S \subseteq P$ such that if we pick a vector v uniformly at random from the surface of the unit sphere, $\mathcal{S}^{d-1}$, S ϵ-maximizes P in direction v with probability at least $1 - \delta$, that is,

$$\Pr(|\omega_v(P) - \omega_v(S)| > \epsilon) \leq \delta$$

There is a simple deterministic algorithm that stores $O(\frac{k}{\delta})$ points and gives us an (ϵ, δ)-hull of a point set P, where k is the batch optimal for the ϵ-hull of P. Here we focus on higher dimensions.

Suppose we fix the dimension d. We give a randomized algorithm that uses n points and with probability at least $1 - p$ gives us an (ϵ, δ)-hull of a point set P, where k is the batch optimal for the ϵ-hull of P, and n satisfies:

$$n = O\left(\frac{k^2}{\delta^2}\left(\log k + \log\frac{1}{\delta} + \log\frac{1}{p}\right)\right)$$

Note that the given complexity hides the dependency on d, the actual time complexity will be multiplied by some (exponential) function of d.

We sketch our algorithm as follows: Choose n random vectors on the unit sphere. For each chosen vector v we store the point p that maximizes P in direction v, that is, $p \cdot v = \omega_v(P)$. This can be done in streaming.

We reduce proving this algorithm in $\mathbb{R}^d$, to proving a related property in $\mathbb{R}^{d-1}$. We then use a VC-dimension based uniform bound to prove this property. A proof outline of this algorithm is in Appendix C.

References

[AC14] S. Arya and T. M. Chan. Better epsilon-dependencies for offline approximate nearest neighbor search, euclidean minimum spanning trees, and epsilon-kernels. In *Proceedings of the Thirtieth Annual Symposium on Computational Geometry*, SOCG'14, pages 416:416–416:425, New York, NY, USA, 2014. ACM. Available from: `http://doi.acm.org/10.1145/2582112.2582161`, `doi:10.1145/2582112.2582161`.

[AHPV04] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Approximating extent measures of points. *J. ACM*, 51(4):606–635, July 2004. Available from: `http://doi.acm.org/10.1145/1008731.1008736`, `doi:10.1145/1008731.1008736`.

[AHPV05] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Geometric approximation via coresets. In *COMBINATORIAL AND COMPUTATIONAL GEOMETRY, MSRI*, pages 1–30. University Press, 2005.

[BHPR16] A. Blum, S. Har-Peled, and B. Raichel. Sparse approximation via generating point sets. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 548–557, 2016.

[BHR15] A. Blum, S. Har-Peled, and B. Raichel. Sparse approximation via generating point sets. *CoRR*, abs/1507.02574, 2015. Available from: `http://arxiv.org/abs/1507.02574`.

[BPF82] J. L. Bentley, F. P. Preparata, and M. G. Faust. Approximation algorithms for convex hulls. *Commun. ACM*, 25(1):64–68, January 1982. Available from: `http://doi.acm.org/10.1145/358315.358392`, `doi:10.1145/358315.358392`.

[Cha06] T. M. Chan. Faster core-set constructions and data-stream algorithms in fixed dimensions. *Computational Geometry*, 35(1):20 – 35, 2006. `doi:http://dx.doi.org/10.1016/j.comgeo.2005.10.002`.

[Cha16] T. M. Chan. Dynamic streaming algorithms for -kernels. In *Proc. 32nd Annu. Sympos. Comput. Geom. (SoCG)*, 2016.

[Cha17] T. M. Chan. Applications of chebyshev polynomials to low-dimensional computational geometry. In *Proc. 33rd Annu. Sympos. Comput. Geom. (SoCG)*, 2017.

[HS04] J. Hershberger and S. Suri. Adaptive sampling for geometric problems over data streams. In *Proceedings of the Twenty-third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '04, pages 252–262, New York, NY, USA, 2004. ACM. Available from: `http://doi.acm.org/10.1145/1055558.1055595`, `doi:10.1145/1055558.1055595`.

[SA17] D. M. M. Sunil Arya, Guilherme D. da Fonseca. Near-optimal -kernel construction and related problems. In *Proc. 33rd Annu. Sympos. Comput. Geom. (SoCG)*, 2017.

[YAPV04] H. Yu, P. K. Agarwal, R. Poreddy, and K. R. Varadarajan. Practical methods for shape fitting and kinetic data structures using core sets. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, SCG '04, pages 263–272, New York, NY, USA, 2004. ACM. Available from: `http://doi.acm.org/10.1145/997817.997858`, `doi:10.1145/997817.997858`.

[ZZ11] H. Zarrabi-Zadeh. An almost space-optimal streaming algorithm for coresets in fixed dimensions. *Algorithmica*, 60(1):46–59, May 2011. Available from: `http://dx.doi.org/10.1007/s00453-010-9392-2`, `doi:10.1007/s00453-010-9392-2`.

Online Unit Covering in L_2

Anirban Ghosh
School of Computing
University of North Florida
Jacksonville, FL, USA
✉ anirban.ghosh@unf.edu

Abstract

Given a set of points P in the plane, the classic Unit Covering (UC) problem asks to compute the minimum number of unit disks (possibly intersecting) required to cover the points in P, along with a placement of the disks.

In this paper, we study an online version of UC in L_2, which we call as Online Unit Covering (OUC) in L_2, or, simply Online Unit Covering (OUC). In OUC, the points arrive online sequentially. Once a point p arrives, a decision needs to be taken - either assign p to a previously placed disk (if possible) or place a new disk to cover p since none of the previously placed disks can cover p. In this setup, once a disk is placed, its placement cannot be altered in future. We show that competitive ratio of an optimal deterministic online algorithm for OUC is at least 3 and at most 5.

Keywords: competitive ratio, geometric cover, online algorithm, unit disk.

1 Introduction

The classic problem of Unit Covering (UC) in L_2 metric is defined as follows. Given a set P of n points $\{p_1, \ldots, p_n\}$ in the Euclidean plane, compute the minimum number of unit disks[1] (possibly intersecting) required to cover the points in P, along with a placement of the disks. UC is a well studied problem in computational geometry. UC is also known by the name Unit Disk Cover. UC problems find their applications in wireless networking, facility location, robotics and other areas of research. Note that the algorithms for UC can be easily scaled for covering points using disks of any fixed radius $r > 0$. In this paper, we fix $r = 1$.

UC is known to be NP-hard in L_2 metric, refer to [15]. Naturally, several attempts have been made to design off-line[2] approximation algorithms. The first PTAS was given by Hochbaum and Maass in [21]. Gonzalez [20] presented two approximation algorithms in L_2 metric; a $2(1 + \frac{1}{l})$-approximation algorithm with runtime $\mathcal{O}(l^2 n^7)$, where l is an integer and another 8-approximation algorithm with runtime $\mathcal{O}(n \log s)$ where s is the number of disks in the optimal cover. Franceschetti et al. [16] presented an algorithm with approximation factor $3(1 + \frac{1}{l})$ and runtime $\mathcal{O}(Kn)$, where l is a integer and K is a constant which depends on l. A 2.8334-approximation algorithm was presented in [18] having runtime $\mathcal{O}(n(\log n \log\log n)^2)$. Chang et al. [7] devised a 5-approximation algorithm with runtime $\mathcal{O}(nm^2)$, where m is the size of a square enclosing the point set. Liu and Lu [26] designed a 25/4-approximation algorithm with runtime $\mathcal{O}(n \log n)$. Recently, Biniaz et al. [3] devised an algorithm with approximation factor 4 and runtime $\mathcal{O}(n \log n)$.

There are several variants of the UC problem which are well-studied in computational geometry. Refer to [1, 2, 5, 9–12, 17, 19, 22, 25] for problem definitions of some of the variants and related results.

Online algorithms have attracted great attention in computer science. Refer to [4, 23] for a discussion on online algorithms. Online problems have numerous areas of application, such as, robotics, operating systems, scheduling, networks, computational finance and many others. Online algorithms form an integral part of interactive computing where data arrives as a sequence of input portions. Once a data element arrives, an online algorithm must take a decision with an objective to construct a solution based on the input seen so far. Typically, when a new data element arrives the solution constructed so far is extended. A decision taken cannot be changed in future. Nothing is known in advance about the future of the incoming data elements. Clearly, the aim should be to always take decisions which prevents the algorithm from producing a bad quality solution in future. Since we can-

[1] In this paper, by unit disk we refer to a closed disk of radius 1.

[2] In an off-line setup, the point set P is known in advance as opposed to online setup.

not have the whole input in advance, it is frequently a challenge to design online algorithms which produce solutions not too bad in comparison to the ones produced by their off-line counterparts.

For an optimization problem, the *competitive ratio* of an online algorithm $\mathcal{A}$ is defined as the worst case ratio between the cost of the solution found by $\mathcal{A}$ and the cost of an optimal solution. It is analogous to *approximation factor* for off-line algorithms.

To the best of our knowledge, the online version of UC in L_2 has not been studied yet. This motivates us to introduce the problem; we call it Online Unit Covering (OUC) in L_2 or in order to shorten it, just Online Unit Covering (OUC). Here, points arrive online, one at a time. When a new point p arrives, we need to take one of the following two decisions. Either we assign p to one of the disks, placed previously, or, we place a new disk in order to cover p (since no other previously placed disk covers it). At the start, no disk is placed; the first disk is placed when the first point arrives. Note that OUC has been studied in L_∞ norm and different variants; refer to [6, 8, 13].

In this paper, we show that any deterministic online algorithm for OUC has competitive ratio at least 3 using a lower bound instance; see Section 2. Lower bound instance constructions have great importance in the research of online algorithms since they provide good estimates to algorithm designers as to what competitive ratio can be achieved in the best case. See [14, 24], for instance. On the other hand, we design a deterministic algorithm with competitive ratio 5. See Section 3.

Notations. Let D_i be a unit disk. Then, by c_i, we denote its center. For a point p, we denote its y-coordinate by $y(p)$.

2 A lower bound for the optimal competitive ratio

In this section, a lower bound for the optimal competitive ratio for OUC is presented. For the off-line version, the best approximation factor achieved so far is $2(1+\frac{1}{l})$, where l is an integer, refer to [20]. Observe that for large positive l, one can get an approximation factor close to 2. Naturally one expects that in the online version, the competitive ratio should be higher than 2, since, it is not possible to see the whole input in advance. In this regard, we show that the optimal competitive ratio is at least 3 in the following theorem.

Theorem 1. *The competitive ratio of any deterministic online algorithm for OUC is at least* 3.

Proof. Consider a deterministic online algorithm $\mathcal{A}$ for OUC. We present an input instance σ to $\mathcal{A}$ and show that the solution $\mathcal{A}(\sigma)$ constructed by it is at least 3 times worst the solution $\mathrm{OPT}(\sigma)$, constructed by an optimal off-line algorithm.

Our proof works like a two player game, played by Alice and Bob. Here, Bob is presenting points to Alice, one at a time. Alice takes the decision whether to place a new disk or not. If a new disk is required, Alice decides where to place it. Bob tries to force Alice to place as many new disks as possible by presenting the points in a smart way. Alice tries to place a new disk in a way such that the placed disk may cover other points presented by Bob in future, thereby reducing the need of placing new disks quite often.

Refer to the full version of this paper for the input instance and the analysis. □

3 An upper bound for the optimal competitive ratio

In this section, we present a deterministic algorithm for OUC having competitive ratio 5. This implies that 5 is also an upper bound of the optimal competitive ratio for any optimal deterministic algorithm for OUC. In the following, the algorithm is presented; we named it CENTERS-OUC.

Algorithm CENTERS-OUC: Let S be the set of unit disks already placed (initially, S is empty) and p be a point which has arrived. If p is covered by an unit disk already placed, then do not take any action. Otherwise, place a new disk centered at p.

We begin with some observations which will aid us to analyze the competitive ratio of the algorithm.

Observation 1. *Consider two unit disks D_1, D_2 centered at c_1, c_2, respectively, such that $c_2 \in D_1$. Let AB be the boundary arc of D_1 lying in D_2. Then, if $p \in AB$, every point in the line segment c_1p is covered by D_2. Furthermore, $|AB| \geq 2\pi/3$.*

Now, we extend our above observation for k unit disks $\{D_1, \ldots, D_k\}$, which collectively can cover a given unit disk D. Observation 2 gives us a condition under which a set of unit disks can cover D.

Observation 2. *Consider a set S of unit disks $\{D_1, \ldots, D_k\}$ such that every $c_i \in D$, where D is an unit disk. For every disk D_i, let A_iB_i be the boundary arc of D lying in D_i. If $\cup_{i=1}^{k} A_iB_i = \mathcal{C}$, where $\mathcal{C}$ is the boundary of D, then $D \subseteq \cup_{i=i}^{k} D_i$. In other words, the disks in S together cover D.*

Next, we present the following observation which will be used in our proof.

Observation 3. *Let D_1, D_2, D_3 be three unit disks such that $c_2, c_3 \in D_1$. If $|c_1c_2| \leq |c_1c_3|$, then $|D_1 \cap D_2| \geq |D_1 \cap D_3|$. In other words, D_2 covers a greater area of D_1 than D_3 does.*

Observation 3 also shows that an unit disk D_1 covers the least part of another unit disk D_2 when c_1 lies on the boundary of D_2.

Theorem 2. *The competitive ratio of any optimal deterministic online algorithm for OUC is at most* 5.

Proof. In this proof, we show that our online algorithm CENTERS-OUC has competitive ratio 5.

Consider an optimal unit disk cover X for the points arrived so far and let $D \in X$. Then we will show that CENTERS-OUC uses at most five unit disks to cover every input point in D. In doing so, we will assume that no other disk in X covers any input point in D. If it does, then lesser number of disks may be placed by our algorithm to cover every input point in D, but certainly never more.

By Observation 2, it follows that if every point on the boundary of D is covered by some unit disk centered in D, then the boundary of D is covered in full by these disks which also cover D. Hence, we focus on covering the boundary points of D using disks centered in D. Our objective is to investigate the worst case scenario and see how many disks can be placed in D to cover the input points in D.

By Observation 3, it is clearly evident that intersection of a disk D_i with D is least when D_i is centered on the boundary of D. So, from now on we will safely assume that incoming input points arrive on the boundary of D. If they arrive inside D, then lesser number of disks may be used by our algorithm, but never more. Next, we will try to accommodate as many unit disks we can centered on the boundary of D using our algorithm CENTERS-OUC.

In the following, we define *gap* to be a maximal contiguous portion of the boundary of D which is not yet covered by any of the placed disks.

Consider the first input point p_1 which arrives on the boundary on D. CENTERS-OUC will place a new disk D_1 centered at p_1. Observe that D_1 covers one-third of the boundary of D. In other words, by Observation 1, the length of the partial boundary of D in D_1 is $2\pi/3$. If no more points arrive in D, we are done.

Now, two more points p_2, p_3 arrive on the boundary of D such that they do not belong to D_1. Assume that for p_2, p_3, disks D_2, D_3 need to be placed, respectively.

There are following three cases based on the number of gaps, G, present after placing D_1, D_2, D_3. It is shown in the full version of this paper that for $G \in \{0, 1, 2\}$, we need at most five disks to cover the input points in D. Also, it is easy to check that $G \geq 3$ is not possible since each disk covers one-third of the boundary of D.

Hence, we see that in worst case, our algorithm places at most five disks in order to cover every input point in D. This concludes our proof. □

4 Open directions

We gave a lower bound of 3 for the optimal competitive ratio for OUC. It remains open to investigate if this lower bound can be improved. On the other hand, to investigate if there is an online algorithm with a competitive ratio less than 5 remains open.

References

[1] M. Basappa, R. Acharyya, G. K. Das, Unit disk cover problem in 2D, *J. Discrete Algorithms* **33** (2015), 193–201.

[2] M. D. Berg, S. Cabello, S. Har-Peled, Covering many or few points with unit disks, *Theory Comput. Syst.* **45** (2009), 446–469.

[3] A. Biniaz, P. Liu, A. Maheshwari, M. Smid, Approximation algorithms for the unit disk cover problem in 2D and 3D, *Comput. Geom.* **60** (2017), 8–18.

[4] A. Borodin, R. El-Yaniv, Online computation and competitive analysis, Cambridge University Press, 1998.

[5] H. Brönnimann, M. T. Goodrich, Almost optimal set covers in finite VC-dimension, *Discrete Comput. Geom.* **14(4)** (1995), 463–479.

[6] T. M. Chan and H. Zarrabi-Zadeh, A randomized algorithm for online unit clustering, *Theory Comput. Syst.* **45(3)** (2009), 486–496. A preliminary version in *Proc. 5th Workshop on Approximation and Online Algorithms (WAOA)*, LNCS 4368, Springer, 2006, pp. 121–131.

[7] C.-Y. Chang, C.-C. Chen, C.-C. Liu, A novel approximation algorithm for minimum geometric disk cover problem with hexagon tessellation, In *Proceedings of the International Computer Symposium*, Advances in Intelligent Systems and Applications - Vol. 1, Springer, (2013), 157–166.

[8] M. Charikar, C. Chekuri, T. Feder, and R. Motwani, Incremental clustering and dynamic information retrieval, *SIAM J. Comput.* **33(6)** (2004), 1417–1440.

[9] B. M. Chazelle, D. T. Lee, On a circle placement problem, *Computing* **36** (1986), 1–16.

[10] F. Claude, G. K. Das, R. Dorrigiv, S. Durocher, R. Fraser, A. López-Ortiz, B. G. Nickerson, A. Salinger, An improved line-separable algorithm for discrete unit disk cover, *Discrete Math. Algorithm. Appl.* **02** (2010), 77–87.

[11] M. De, G. K. Das, P. Carmi, S. C. Nandy, Approximation algorithms for a variant of discrete piercing set problem for unit disks, *Internat. J. Comput. Geom. Appl.* **23** (2013), 461–477.

[12] G. K. Das, R. Fraser, A. Lopez-Ortiz, B. G. Nickerson, On the discrete unit disk cover problem, *Internat. J. Comput. Geom. Appl.* **22** (2012), 407–419.

[13] A. Dumitrescu, C. D. Tóth, Online unit clustering in higher dimensions, `arXiv:1708.02662`, 2017.

[14] L. Epstein, R. van Stee, On the online unit clustering problem, *ACM Trans. Algorithms*, **7(1)**, (2010).

[15] R. J. Fowler, M. S. Paterson, S. L. Tanimoto, Optimal packing and covering in the plane are NP-complete, *Inform. Process. Lett.* **12(3)** (1981), 133–137.

[16] M. Franceschetti, M. Cook, J. Bruck, A geometric theorem for approximate disk covering algorithms, Technical report, 2001.

[17] R. Fraser, A. López-Ortiz, The within-strip discrete unit disk cover problem, *Theoret. Comput. Sci.* **674** (2017), 99–115.

[18] B. Fu, Z. Chen, M. Abdelguerfi, An almost linear time 2.8334-approximation algorithm for the disc covering problem, In *Proceedings of 3rd International Conference of Algorithmic Aspects in Information and Management*, Springer, 2007, pp. 317–326.

[19] H. Ghasemalizadeh, M. Razzazi, An improved approximation algorithm for the most points covering problem, *Theory Comput. Syst.* **50** (2012), 545–558.

[20] T. F. Gonzalez, Covering a set of points in multidimensional space, *Inf. Process. Lett.* **40 (4)** (1991), 181–188.

[21] D. S. Hochbaum, W. Maass, Approximation schemes for covering and packing problems in image processing and VLSI, *J. ACM* **32(1)** (1985), 130–136.

[22] H. Kaplan, M. J. Katz, G. Morgenstern, M. Sharir, Optimal cover of points by disks in a simple polygon, *SIAM J. Comput.* **40(6)** (2011), 1647–1661.

[23] R. M. Karp, On-line algorithms versus off-line algorithms: How much is it worth to know the future?, In *IFIP Congress* (1), 1992, (Vol. 12, pp. 416–429).

[24] J. Kawahara, K. M. Kobayashi, An improved lower bound for one-dimensional online unit clustering, *Theoret. Comput. Sci.* **600** (2015), 171–173.

[25] C. Liao, S. Hu, Polynomial time approximation schemes for minimum disk cover problems, *J. Comb. Optim.* **20** (2010), 399–412.

[26] P. Liu, D. Lu, A fast 25/6-approximation for the minimum unit disk cover problem, `arXiv:1406.3838`, 2014.

Online Unit Covering in L_2

Anirban Ghosh
School of Computing
University of North Florida
Jacksonville, FL, USA
✉ anirban.ghosh@unf.edu

Abstract

Given a set of points P in the plane, the classic Unit Covering (UC) problem asks to compute the minimum number of unit disks (possibly intersecting) required to cover the points in P, along with a placement of the disks.

In this paper, we study an online version of UC in L_2, which we call as Online Unit Covering (OUC) in L_2, or, simply Online Unit Covering (OUC). In OUC, the points arrive online sequentially. Once a point p arrives, a decision needs to be taken - either assign p to a previously placed disk (if possible) or place a new disk to cover p since none of the previously placed disks can cover p. In this setup, once a disk is placed, its placement cannot be altered in future. We show that competitive ratio of an optimal deterministic online algorithm for OUC is at least 3 and at most 5.

Keywords: competitive ratio, geometric cover, online algorithm, unit disk.

URL for the full version: https://arxiv.org/abs/1710.00954

An $O(n \log n)$-Time Algorithm for the k-Center Problem in Trees*

Haitao Wang[1] and Jingru Zhang[2]

[1] Department of Computer Science, Utah State University, Logan, UT 84322, USA,
haitao.wang@usu.edu
[2] Department of Computer Science, Marshall University, Huntington, WV 25755, USA,
jingru.zhang@marshall.edu

Abstract. We consider a classical k-center problem in trees. Let T be a tree of n vertices and every vertex has a nonnegative weight. The problem is to find k centers on the edges of T such that the maximum weighted distance from all vertices to their closest centers is minimized. Megiddo and Tamir (SIAM J. Comput., 1983) gave an algorithm that can solve the problem in $O(n \log^2 n)$ time by using Cole's parametric search. Since then it has been open for over three decades whether the problem can be solved in $O(n \log n)$ time. We present an $O(n \log n)$ time algorithm for the problem and thus settle the open problem affirmatively.

* The full paper of this abstract can be found at https://arxiv.org/abs/1705.02752.

Subquadratic-Space Query-Efficient Data Structures for Realizable Order Types

Jean Cardinal[*], Timothy M. Chan[†], Stefan Langerman[‡], Aurélien Ooms[§]

October 27, 2017

Abstract

Realizable order types and abstract order types are combinatorial analogues of line arrangements and pseudoline arrangements. They only store information about the relative orientation of triples of lines and pseudolines. We give an optimal encoding for abstract order types that allows efficient query of the orientation of any triple: the encoding uses $O(n^2)$ bits and an orientation query takes $O(\log n)$ time in the word-RAM model. We show how to shorten the encoding to $o(n^2)$ bits for realizable order types. We show how to attain $o(\log n)$ query time at the expense of an extra $O(\log^{\varepsilon} n)$ factor in the size of the encoding. For realizable order types, the encoding remains subquadratic in size.

1 Introduction

At SoCG'86, Chazelle asked [22]:

> "How many bits does it take to know an order type?".

This question is of importance in Computational Geometry for the following two reasons. First, in many algorithms dealing with sets of points in the plane, the only relevant information about the points is the orientation (clockwise or counterclockwise) of the triples of points in the set [13]. Second, computers as we know them can only handle numbers with finite description and we cannot assume that they are going to handle arbitrary real numbers without some sort of encoding. The study of *robust* algorithms is focused on ensuring the correct solution of problems on finite precision machines (see the Chapter on this issue in The Handbook [30]).

The orientation of an input triple $((x_1, y_1), (x_2, y_2), (x_3, y_3))$ corresponds to the sign of the determinant

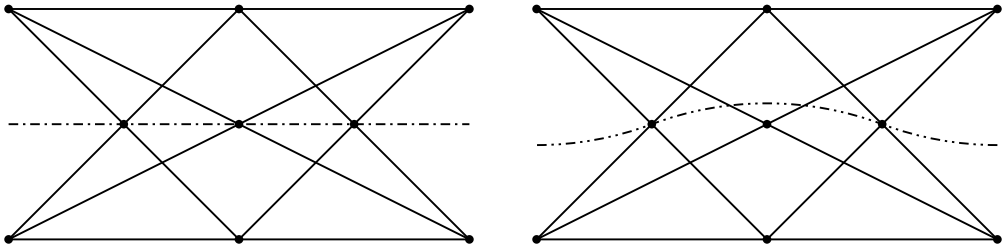
$$\begin{vmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{vmatrix}.$$

[*]jcardin@ulb.ac.be, Université libre de Bruxelles (ULB). Supported by the "Action de Recherche Concertée" (ARC) COPHYMA, convention number 4.110.H.000023.

[†]tmc@illinois.edu, University of Illinois at Urbana-Champaign.

[‡]slanger@ulb.ac.be, Université libre de Bruxelles (ULB). Directeur de recherches du Fonds de la Recherche Scientifique-FNRS.

[§]aureooms@ulb.ac.be, Université libre de Bruxelles (ULB). Supported by the Fund for Research Training in Industry and Agriculture (FRIA).



(a) Realizable order type. (b) Abstract order type.

Figure 1: Pappus's configuration

The set of all orientations is usually referred to as the *order type* of the point set. A great deal of the literature in computational geometry deals with this notion [1–12, 14–23, 25–29], and the list goes on. The order type of a point set has been further abstracted into combinatorial objects known as (rank-three) *oriented matroids* [16]. The *chirotope axioms* define consistent systems of signs of triples [10]. From the Topological Representation Theorem [11], all such *abstract* order types correspond to pseudoline arrangements, while, from the standard projective duality, order types of point sets correspond to straight line arrangements.

In this contribution, we are interested in *compact* data structures for order types. We consider both order types that are realizable as point sets, and abstract order types, that is, order types realizable as pseudoline arrangements. We wish to design data structures using as few bits as possible that can be used to quickly answer orientation queries.

Abstract order types are much more numerous than realizable order types: there are $2^{\Theta(n^2)}$ abstract order types [14] and only $2^{\Theta(n \log n)}$ realizable order types [9, 20]. Hence information theory tells us that we need quadratic space for abstract order types whereas we only need linearithmic space for realizable order types. This discrepancy stems from the algebraic nature of realizable order types. As an example, Pappus gives a configuration where eight triples of concurrent straight lines force a ninth, whereas the ninth triple cannot be enforced by pseudolines [26, 29] (see Figure 1).

An obvious idea for storing an order type of a set of points (aside from explicitly storing all $\binom{n}{3}$ orientations) is to store the coordinates of the points, and answer orientation queries in constant time by computing the determinant. While this should work in many practical settings, it cannot work for all point sets. Perles's configuration shows that some arrangement of points, containing collinear triples, forces at least one coordinate to be irrational [24] (see Figure 2). Even for points in general position, it is well known that some arrangements require doubly exponential coordinates, hence coordinates with exponential bitsizes [23].

Goodman and Pollack defined λ-matrices that can be used for encoding realizable order types using $O(n^2 \log n)$ bits [18]. They asked if the space requirements can be moved closer to the information theoretic lower bounds. Felsner and Valtr [14, 15] show how to encode abstract order types optimally via the wiring diagram of their corresponding allowable sequence (as defined in [17]). However, it is not known how to decode the orientation of one triple from any of those encodings in, say, sublinear time. Moreover, since the information theoretic lower bound for realizable order types is only $\Omega(n \log n)$, we must ask if this space bound is approachable for those order types if we wish simultaneously to keep orientations queries reasonably efficient.

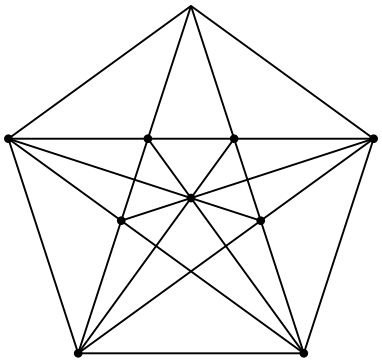


Figure 2: Perles's configuration

2 Our Results

We give the first optimal encoding for abstract order types that allows efficient query of the orientation of any triple: the encoding uses $O(n^2)$ bits and a query takes $O(\log n)$ time in the word-RAM model. Our data structure is far from being space-optimal for realizable order types and we can hope to save some space while keeping the queries fast. We show how to reduce the size of our data structure to subquadratic for realizable order types. We further refine our data structure so as to reduce the query time to sublogarithmic. This improvement is applicable for both abstract and realizable order types at the cost of an extra $O(\log^{\varepsilon} n)$ factor in the space complexity. For realizable order types, the encoding remains subquadratic in size.

Our data structure is the first subquadratic encoding of realizable order types that allows efficient query of the orientation of any triple. We know of no subquadratic constant-degree algebraic decision tree for the related problem of deciding whether a point set contains a collinear triple. Any such decision tree would yield another subquadratic encoding for realizable order types. We see our results as a stepping stone towards subquadratic nonuniform algorithms for this related problem.

References

[1] Oswin Aichholzer, Franz Aurenhammer, and Hannes Krasser. Enumerating order types for small point sets with applications. *Order*, 19(3):265–281, 2002.

[2] Oswin Aichholzer, Franz Aurenhammer, and Hannes Krasser. On the crossing number of complete graphs. In *Proceedings of the eighteenth annual symposium on Computational geometry*, pages 19–24. ACM, 2002.

[3] Oswin Aichholzer, Jean Cardinal, Vincent Kusters, Stefan Langerman, and Pavel Valtr. Reconstructing point set order types from radial orderings. *International Journal of Computational Geometry & Applications*, 26(03n04):167–184, 2016.

[4] Oswin Aichholzer, Matias Korman, Alexander Pilz, and Birgit Vogtenhuber. Geodesic order types. *Algorithmica*, 70(1):112–128, 2014.

[5] Oswin Aichholzer and Hannes Krasser. The point set order type data base: A collection of applications and results. In *CCCG*, volume 1, pages 17–20, 2001.

[6] Oswin Aichholzer and Hannes Krasser. Abstract order type extension and new results on the rectilinear crossing number. In *Proceedings of the twenty-first annual symposium on Computational geometry*, pages 91–98. ACM, 2005.

[7] Oswin Aichholzer, Vincent Kusters, Wolfgang Mulzer, Alexander Pilz, and Manuel Wettstein. An optimal algorithm for reconstructing point set order types from radial orderings. In *International Symposium on Algorithms and Computation*, pages 505–516. Springer, 2015.

[8] Oswin Aichholzer, Tillmann Miltzow, and Alexander Pilz. Extreme point and halving edge search in abstract order types. *Computational Geometry*, 46(8):970–978, 2013.

[9] Noga Alon. The number of polytopes configurations and real matroids. *Mathematika*, 33(1):62–71, 1986.

[10] Anders Björner, Michel Las Vergnas, Bernd Sturmfels, Neil White, and Günter M Ziegler. Oriented matroids, encyclopedia of mathematics, vol. 46, 1993.

[11] Jürgen Bokowski, Susanne Mock, and Ileana Streinu. On the Folkman-Lawrence topological representation theorem for oriented matroids of rank 3. *Eur. J. Comb.*, 22(5):601–615, 2001.

[12] Jürgen Bokowski, Jürgen Richter-Gebert, and Werner Schindler. On the distribution of order types. *Computational Geometry*, 1(3):127–142, 1992.

[13] Herbert Edelsbrunner. *Algorithms in combinatorial geometry*, volume 10. Springer Science & Business Media, 2012.

[14] Stefan Felsner. On the number of arrangements of pseudolines. In *Proceedings of the twelfth annual symposium on Computational geometry*, pages 30–37. ACM, 1996.

[15] Stefan Felsner and Pavel Valtr. Coding and counting arrangements of pseudolines. *Discrete & Computational Geometry*, 46(3):405–416, 2011.

[16] Jon Folkman and Jim Lawrence. Oriented matroids. *Journal of Combinatorial Theory, Series B*, 25(2):199–236, 1978.

[17] Jacob E Goodman. Proof of a conjecture of Burr, Grünbaum, and Sloane. *Discrete Mathematics*, 32(1):27–35, 1980.

[18] Jacob E Goodman and Richard Pollack. Multidimensional sorting. *SIAM Journal on Computing*, 12(3):484–507, 1983.

[19] Jacob E Goodman and Richard Pollack. Semispaces of configurations, cell complexes of arrangements. *Journal of Combinatorial Theory, Series A*, 37(3):257–293, 1984.

[20] Jacob E. Goodman and Richard Pollack. Upper bounds for configurations and polytopes in $\mathbb{R}^d$. *Discrete & Computational Geometry*, 1:219–227, 1986.

[21] Jacob E Goodman and Richard Pollack. The complexity of point configurations. *Discrete Applied Mathematics*, 31(2):167–180, 1991.

[22] Jacob E Goodman and Richard Pollack. Allowable sequences and order types in discrete and computational geometry. In *New trends in discrete and computational geometry*, pages 103–134. Springer, 1993.

[23] Jacob E. Goodman, Richard Pollack, and Bernd Sturmfels. Coordinate representation of order types requires exponential storage. In *STOC*, pages 405–410. ACM, 1989.

[24] Branko Grünbaum. *Convex Polytopes*. Springer, 2005.

[25] Alfredo Hubard, Luis Montejano, Emiliano Mora, and Andrew Suk. Order types of convex bodies. *Order*, 28(1):121–130, 2011.

[26] Friedrich Levi. Die teilung der projektiven ebene durch gerade oder pseudogerade. *Ber. Math.-Phys. Kl. Sächs. Akad. Wiss*, 78:256–267, 1926.

[27] Yoshitake Matsumoto, Sonoko Moriyama, Hiroshi Imai, and David Bremner. Matroid enumeration for incidence geometry. *Discrete & Computational Geometry*, 47(1):17–43, 2012.

[28] Jaroslav Nešetřil and Pavel Valtr. A ramsey property of order types. *Journal of Combinatorial Theory, Series A*, 81(1):88–107, 1998.

[29] Gerhard Ringel. Teilungen der ebene durch geraden oder topologische geraden. *Mathematische Zeitschrift*, 64(1):79–102, 1956.

[30] Chee K. Yap. Robust geometric computation. In *Handbook of Discrete and Computational Geometry, 2nd Ed.*, pages 927–952. Chapman and Hall/CRC, 2004.

Lightweight Sketches for Mining Trajectory Data

Maria Astefanoaei* Panagiota Katsikouli* Mayank Goswami** Rik Sarkar*
*University of Edinburgh
**City University of New York
m.s.astefanoaei@sms.ed.ac.uk, p.katsikouli@sms.ed.ac.uk, mayank.goswami@qc.cuny.ny, rsarkar@inf.ed.ac.uk

Abstract

Mining trajectory data can be expensive in large datasets, due to the cost of computing distances between trajectories. We develop two complementary lightweight sketches or summaries for trajectories. The first sketch is geometry-based and takes into account the embedding of the trajectory, whereas the other sketch is purely shape-based.

Our first sketch consists of results of intersecting a trajectory with a random set of disks in the plane. This simple mechanism yields a short bit vector that can be used to practically estimate more traditional measures like Hausdorff and Frechet distance between trajectories. On real trajectory data, using bit vectors of only 60 bits, we find that the Hamming distance between the sketches shows strong correlation with the Hausdorff and Frechet distances.

Our second sketch is independent of scaling, translations and rotations. It consists of measuring the turns made by the tangent vector of a trajectory. After finding the optimal rotation, we compare two trajectories based on this turn-angle metric. We show how to use dimension-reduction methods to reduce the space usage of this sketch.

The sketches can be further used to apply locality sensitive hashing on the trajectories, and achieve high pruning ratio for nearest neighbour searches. The sketch mechanism has additional desirable qualities such as ease of implementation in resource constrained environments, and in streaming data. Its performance degrades gracefully with memory constraints, sensing errors and missing data.

1 Introduction

The proliferation of mobile and GPS enabled devices have given rise to large databases containing trajectories of individuals. Collective data from many users can provide fine grained information about social behavior and interactions, traffic, infrastructure services etc.

Mining trajectory data requires efficient algorithms for basic operations such as distance computation and nearest neighbor queries. Commonly used distance measures like Hausdorff and Fréchet [2] distances are expensive compute in large datasets. In this work, we develop a simple sketching mechanism to easily compare trajectories and speed up the essential operations of distance estimation and near neighbor search.

In related works, trajectory processing has traditionally been treated from the *simplification* perspective, where the goal is to construct an approximate curve with fewer vertices, such as in the Douglas-Peucker Algorithm [8], and other methods with stronger compression guarantees [9, 1]. More recent methods have generalized the simplification problem to preserve speed, time, direction and other motion related information [13, 12], or focused on computing simplifications online [11].

The purpose of these methods is to store enough information to reconstruct the orginal trajectory to a high accuracy. In contrast, our objective is to store enough information to simply determine if two trajectories are likely to be similar. We show that on real trajectories, a simple hashing strategy suffices to detect closeness.

Motivations. There are many domains where trajectory mining techniques can be applied. Transportation and traffic engineering are natural applications. Trajectory data can be used to understand typical routes taken by users, and thus improve public transport infrastructure [14]. In geographic information systems, they can be used to automatically infer road segments and crossings [5]. They can be used to optimize bus routes [14] and taxi deployments [15] and infrastructure for electrical vehicles [7]. sIn a long term perspective, as ride-sharing and self driving vehicles become common, both efficiency and benefits of these systems can be improved through better understanding of travel patterns of users.

Our contributions. We develop two different sketching methods that reflect the **proximity and shapes** of trajectories trajectories respectively.

1. Proximity Sketching If two trajectories are generally close, then in a randomly chosen neighborhood, the presence of one trajectory will likely imply presence of the other. A neighborhood that intersects one trajectory but not the other will be relatively rare. On the other hand, for two trajectories that are generally far from each other, common neighborhoods will be rare. This intuition is shown in Figure 1, where we use circular

disks to represent randomly chosen neighborhoods. A disk intersecting a trajectory but missing a nearby one is relatively rare. This is a form of Locality Sensitive Hashing for trajectories, where each disk intersection acts as a hash function.

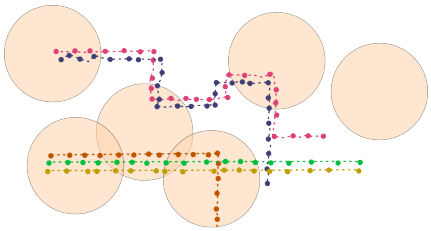


Figure 1. An example of random disk neighborhoods intersecting trajectories. Similar trajectories usually either both intersect a particular disk or both do not.

We show that under some realistic assumptions that model trajectories with clear start and destinations, this property can be used to provably distinguish between similar and dissimilar trajectories with high probability. We define a measure of distance based on the disk intersections and experiments on real data show that a relatively small number of random disks (about 60 disks) suffice to obtain a fair estimate of the distance between two trajectories in a city; it shows high correlation with the Hausdorff and Fréchet distance measures.

In nearest neighbour queries the sketches can be used to effectively prune the data and reduce the cost of searching, without any substantial loss in accuracy. The method is also amenable to use in practical distributed settings such as sensor and mobile computing.

2. Shape sketching In many applications, we only care about the "shape" of the trajectory, and not about how it is embedded in the ambient space. In such shape-based recognition or searching applications, if a trajectory T_2 is a translated and/or rotated and/or scaled version of trajectory T_1, we ideally want to consider them to be the same. Notice that the previous metrics such as the Hausdorff metric or the Frechet distance are not invariant under such changes. Motivated by [3], we consider a metric that arises from the "turns" made by a trajectory. We define this metric in Section 3, and then we show how to compute lightweight sketches using dimension-reduction methods and build the database for comparing trajectories.

2 Disk Intersection Distance

We define a trajectory T as a sequence of embedded vertices $T = \{v_0, v_1, \ldots v_n\}$. In the most general case they can be quite arbitrary, but practical trajectories are often more well behaved. Consider, for example, trajectories corresponding to trips taken by passengers in taxi cabs or buses. Trajectories of such trips have well defined start and end points and they make clear progress in a direction; they do not backtrack long sections. Our interest therefore, is in trajectories that consistently achieve an efficient displacement for the user compared to the local length.

We define the R-offset of a trajectory T, denoted by $A_{\text{off}}(T, R)$, as the area of the region within Euclidean distance at most R form T. For a trajectory of length l the area of the offset changes depending on the angle α between successive straight segments. As α decreases, the total area of the offset decreases, meaning that sharp turns reduce the area of the offset. In particular, sharp turns at a scale comparable to R will cause a greater reduction of the area, while small noisy fluctuations, even if sharp, will not affect a large change.

Based on this idea, we define the class of trajectories that we consider in our problem:

Definition 2.1 (Progressive trajectory). *We say that a trajectory T of length l is ε-progressive if the area of its $R-$offset is*

$$(1-\varepsilon)(2Rl + \pi R^2) \leq A_{\text{off}}(T, R) \leq (1+\varepsilon)(2Rl + \pi R^2),$$

that is, at most an ε factor away from the area of the R-offset of the straight line of length l.

2.1 Disk Intersection Sketch and Trajectory Distances

We define our *sketch* of a trajectory as the record of its intersection with D random discs on a map:

Definition 2.2 ((D, R)-Disk Intersection Sketch). *The Disc Intersection Sketch (DIS) of a trajectory T is the binary vector $S(T) = d_0 d_1 \ldots d_{D-1}$ of length D, defined in terms of a set of D random but fixed disks. The vector element $d_i = 1$ if the disk i intersects trajectory T, else it is 0.*

Correspondingly, we define the distance between trajectories as the hamming distance between their sketches:

Definition 2.3 ((D, R)-Disk Intersection Distance $d_{D,R}$). *The Disk Intersection Distance (DID) between two trajectories T_1, T_2 is the Hamming distance between their (D, R) Disk Intersection Sketches. That is, the number of indices with different entries: $|\{i : d_{1,i} \neq d_{2,i}\}|$.*

Suppose we are mining trajectories in a region of area A, and for two trajectories $T_1 = \{v_0, v_1, \ldots v_p\}$ and $T_2 = \{u_0, u_1, \ldots u_q\}$, $\ell = p + q$ is the total size of the two trajectories. For simplicity, we are assuming that the boundaries of the region are at a distance of at least R from any vertex of T_1 or T_2. Then We can show that:

Theorem 2.4. *For Hausdorff distance $H(T_1, T_2) < 2R$, the (D, R) Disk Intersection Distance satisfies*

$$E[d_{D,R}(T_1, T_2)] \leq \frac{4\pi D \ell R}{A} H(T_1, T_2).$$

The implication of the theorem is that, when the Hausdorff distance is small, then beyond factors of constants and known parameters, the DID is bounded by the Hausdorff distance.

On the other hand, we are also interested in cases where at least significant portions of the trajectories are far from each other. Which leads us to the property:

Theorem 2.5. *If in ε-progressive trajectories T_1 and T_2 there exist subtrajectories of total length at least l where each vertex is at a distance of least $2R$ from the other trajectory, then:*

$$E[d_{D,R}(T_1, T_2)] \geq D\frac{2(1-\varepsilon)(2Rl + \pi R^2)}{A}.$$

DID is therefore a robust measure – it relies on large portions of the trajectories being well separated to assign a large distance between them and is not sensitive to a few outliers.

3 Turn-based metric

In order to define the metric, we first need to talk about how the trajectories are represented. Scale invariance demands that it should be possible for two trajectories of different lengths to be distance zero in our metric, thus requiring some kind of normalization. We normalize all our trajectories to be length one. In what follows, a trajectory T denotes a piecewise linear[1] curve $T : [0,1] \to \mathbb{R}^2$.

Given a trajectory T, we define a function $\Theta_T(s)$ (supported on $[0,1]$) that measures the turn-angle as in [3]. $\Theta_T(0)$ is the angle that the forward tangent at the starting point $T(0)$ makes with the x-axis (or any fixed orientation). $\Theta_T(s)$ similarly denotes the angle at the point $T(s)$. For piece-wise linear trajectories, the function Θ will have jumps at the vertices (or turning points) of the trajectory. It decreases with left-hand turns and increases with right-hand turns[2].

Because we will only be using the function $\Theta_T(s)$ to compare a trajectory T to other trajectories, it is clear that our metric will be translation-invariant. Next, we notice that rotating a trajectory T amounts to a shift of $\Theta_T(s)$. To take rotations into account, we need to match the trajectories upto such shifts. Given two trajectories T_1 and T_2, we define their distance[3] to be $d(T_1, T_2) := \min_\theta E(T_1, T_2, \theta)$, where $E(T_1, T_2, \theta) = \left(\int_0^1 |\Theta_{T_1}(s) - \Theta_{T_2}(s) + \theta|^2\right)^{1/2}$.

[1]The definition given makes sense for smooth trajectories too. However, our data structure only works on piece-wise linear trajectories.

[2]Note that for simple closed trajectories (or simple polygons), $\Theta(s+1) = \Theta(s) + 2\pi$.

[3]The fact that d is a metric follows from Minkowski inequality ($||f+g|| \leq ||f|| + ||g||$) applied to the L_2 norm, and we omit the proof here.

3.1 The Sketching algorithm

We discretize $[0,1]$ into a set S of D points, for some large D. We will restrict the preimage of any vertex of a trajectory $T : [0,1] \to \mathbb{R}^2$ to lie in S. In other words, our trajectories are only allowed[4] to "turn" at points in S.

For each trajectory T we compute the function $\Theta_T(.)$ as a D-dimensional vector by computing it on every $s \in S$. We also compute the value $V(T) := \int_0^1 \Theta_T(s)ds$. Simple calculus reveals that argmin $E(T_1, T_2, \theta)$ is $\theta^* = V(T_1) - V(T_2)$. Thus, given the vectors $\Theta_{T_i}(.)$ and the values $V(T_i)$ for $i = 1, 2$, we can calculate the distance.

We now show how to represent sketches in a much more compact way, while still being able to compute a good approximation of the distance between two trajectories. Since storing vectors of length D may be prohibitive, we use the Johnson-Lindenstrauss lemma [6]. We denote by $w(T_i)$ the D-dimensional vector representing T_i. By the Johnson-Lindenstrauss lemma [6], there exists a linear map $f : \mathbb{R}^D \to \mathbb{R}^{D'}$, where $D' = O(\frac{\log n}{\epsilon^2})$, such that for any two trajectories T_i and T_j ($1 \leq i, j \leq n$), $||f(w(T_i)) - f(w(T_i))||^2$ is within $1 \pm \epsilon$ times $||w(T_i) - w(T_i)||^2$.

However, because we want our metric to be rotation-invariant, what we want is to approximate $||w(T_i) - w(T_j) + \theta^*.\mathbf{1}||^2$, where $\theta^* = V(T_i) - V(T_j)$ and $\mathbf{1}$ is the vector of ones. We note that we have stored the values $V(T_i)$ and $V(T_j)$, and we also have the linear map f. By linearity, $f(w(T_j) + \theta^*.\mathbf{1}) = f(w(T_j)) + f(\theta^*.\mathbf{1})$, which in turn equals $f(w(T_j)) + \theta^* f(\mathbf{1})$ and so we can just store $f(\mathbf{1})$ and the reduced sketches after applying the JL transform. Thus we have proved:

Theorem 3.1. *Given n trajectories $T_i : [0,1] \to \mathbb{R}^2$ that only turn on a discrete set of points $S \subset [0,1]$ and an $\epsilon > 0$, one can store sketches of the n trajectories using $O(\frac{n \log n}{\epsilon^2})$ space in total. Given a pair (i,j), one can compute from these sketches in $O(\frac{\log n}{\epsilon^2})$ time a $1 \pm \epsilon$ approximation of the turn-angle-distance $d(T_i, T_j)$.*

We remark that if one wants $\epsilon = 0$, the space usage will be $n|S|$, as dimension reduction cannot be used.

The above data structure returns the distance between any two input trajectories. In many applications one may be interested in nearest neighbor queries, i.e., given as query a trajectory T, return the nearest (among $T_1, \cdots, T_n$) trajectory to T, where nearest is w.r.t. to the turn-angle metric. Because this metric is translation, rotation and scale invariant, this is essentially a database of "trajectory shapes".

The exact nearest neighbor problem does not allow efficient (sublinear in n) query time, and so here we look at the c-approximate nearest neighbor problem, where

[4]Although this sounds like a limitation of our method, a dense-enough discretization S does not alter the distance computation drastically, and is akin to the snap-to-grid simplification used often in computational geometry and vision.

the algorithm must return a trajectory whose distance to the query trajectory T is at most c times the distance of T to its nearest trajectory. Because of the way we design our sketches, we can use an LSH-based approximate nearest neighbor data structure ([10]), giving rise to the following theorem.

Theorem 3.2. *Given n trajectories $T_i : [0,1] \to \mathbb{R}^2$ that only turn on a discrete set of points $S \subset [0,1]$, a $c > 1$, and an $\epsilon > 0$, one can construct a data structure requiring $O(n^{1+1/c^2} + \frac{n \log n}{\epsilon^2})$ space. Given a query trajectory T, the data structure returns in $O(n^{1/c^2})$ time a trajectory T_i such that $d(T, T_i)$ is at most $c(1+\epsilon)$ times $\min_j d(T, T_j)$.*

4 Experiments

We tested the algorithm on commonly used public datasets [4, 16]. On real data, the disk intersection distance shows high correlation with Hausdorff and Frechet distances (Fig. 2).

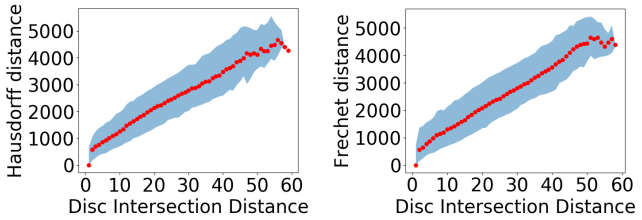


Figure 2. Correlation with DID. The red dots are the mean for each bucket, shaded area shows 5 to 95 percentile. (a) Hausdorff distance. (b) Fréchet distance.

In other experiments, we found that DID is an effective way of pruning data for near neighbor search. When pruning away 95% of all trajectories from the search space, it preserves the nearest neighbor in 60% of cases and one of the two nearest neighbors in 90% of cases. Thus, a more accurate algorithm can then be applied to a small dataset to find near neighbors in most cases. In experiments on computational time, this approach naturally outperforms using simple Hausdorff or Frechet distance based nearest neighbor search by a large margin.

5 Conclusion

We have presented two complementary approaches to trajectory sketching. The natural next step is to develop a combined approach that can simultaneously incorporate shape and proximity of trajectories to produce more accurate and faster processing.

References

[1] P. K. Agarwal, S. Har-Peled, N. H. Mustafa, and Y. Wang. Near-linear time approximation algorithms for curve simplification. In *Proceedings of the 10th Annual European Symposium on Algorithms*, ESA '02, pages 29–41, London, UK, UK, 2002. Springer-Verlag.

[2] H. Alt and M. Godau. Computing the fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5(01n02):75–91, 1995.

[3] E. M. Arkin, L. P. Chew, D. P. Huttenlocher, K. Kedem, and J. S. Mitchell. An efficiently computable metric for comparing polygonal shapes. Technical report, CORNELL UNIV ITHACA NY, 1991.

[4] L. Bracciale, M. Bonola, P. Loreti, G. Bianchi, R. Amici, and A. Rabuffi. CRAWDAD dataset roma/taxi (v. 2014-07-17). Downloaded from `http://crawdad.org/roma/taxi/20140717`, July 2014.

[5] Y. Chen and J. Krumm. Probabilistic modeling of traffic lanes from gps traces. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 81–88. ACM, 2010.

[6] S. Dasgupta and A. Gupta. An elementary proof of a theorem of johnson and lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003.

[7] J. Dong, C. Liu, and Z. Lin. Charging infrastructure planning for promoting battery electric vehicles: An activity-based approach using multiday travel data. *Transportation Research Part C: Emerging Technologies*, 38:44–55, 2014.

[8] D. Douglas and T. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 11(2):112–122, 1973.

[9] H.Imai and M. Iri. Polygonal approximations of a curve-formulations and algorithms. *Computational morphology*, 1988.

[10] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.

[11] P. Katsikouli, R. Sarkar, and J. Gao. Persistence based online signal and trajectory simplification for mobile devices. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 371–380. ACM, 2014.

[12] C. Long, R. C.-W. Wong, and H. V. Jagadish. Direction-preserving trajectory simplification. *Proc. VLDB Endow.*, 6(10):949–960, Aug. 2013.

[13] N. Meratnia and A. Rolf. Spatiotemporal compression techniques for moving point objects. In *Advances in Database Technology-EDBT 2004*, pages 765–782. Springer, 2004.

[14] S. Ngamchai and D. J. Lovell. Optimal time transfer in bus transit route network design using a genetic algorithm. *Journal of Transportation Engineering*, 129(5):510–521, 2003.

[15] H. Tang, M. Kerber, Q. Huang, and L. Guibas. Locating lucrative passengers for taxicab drivers. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 504–507. ACM, 2013.

[16] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma. Mining interesting locations and travel sequences from gps trajectories. In *Proceedings of the 18th international conference on World wide web*, pages 791–800. ACM, 2009.

Calculating the Dominant Guard Set of a Simple Polygon

Eyüp Serdar Ayaz* Alper Üngör*

Abstract

Within a simple polygon, a point p is said to dominate another point q, if the visibility polygon of q is a subset of the visibility polygon of p. A point that is not dominated by others is called a dominant point. An optimal solution for an art gallery problem can be chosen among dominant points only. In this paper, we present an algorithm that finds all dominant points in a simple polygon. Our algorithm works in $O(n^4 \log n)$ time which is an improvement over the previous algorithm that works in $O(n^5)$, where n is the number of vertices of the input polygon.

1 Introduction

A relation that is reflexive, transitive and anti-symmetric is called an inclusion relation in set theory. Domination relation between two points in the concept of visibility is the inclusion relation with respect to the visibility polygons of those points. For a set of points that guards a polygon, we can replace a guard g with another guard g' if g' dominates g. Therefore, we can replace a possible guard set G with another guard set G' that only includes the dominant points.

Domination concept for visibility polygons have been studied on art gallery problems on discrete domains [2,7] (as dominant regions and light atomic visibility polygons respectively) and continuous domains [1] as well as on watchman tours [3, 5]. For a given polygon, an $O(n^5)$ time algorithm to find the set of dominant points is given in [1] . In the same paper, this algorithm is used in an iterative scheme to refine the locations of possible guards to approach the art gallery problem. Here, we propose a new algorithm to calculate the set of all dominant points in $O(n^4 \log n)$ time.

2 Preliminaries

The input for dominant guard set problem is a simple polygon $P \subset \mathbb{R}^2$ with n vertices. Two points $p, q \in P$ *see* each other if the line segment $\overline{pq}$ does not intersect with the outside of P. If a point p in P sees a reflex vertex v of P and the tangential ray $\overrightarrow{pv}$ continues in P after hitting v then p *sees past* v. If the outer region is on the left (right) of the tangent, then v *sees past left (right)* v. For an edge $\overrightarrow{e}$ of P in the counter-clockwise orientation, the closure of the half-plane on the left of $\overrightarrow{e}$ is denoted as $l^c(\overrightarrow{e})$. For two points $p, v \in P$ such that p sees past left (right) v, the closure of the half-plane on the right (left) of $\overrightarrow{pv}$ is denoted as $l^c(p, v)$.

The set of points in P that can be seen from a point $p \in P$ is called the *visibility polygon* of p, denoted as $\mathcal{V}(p)$. The set of points that sees all the points in a polygon is called the *kernel* of that polygon. The kernel of $\mathcal{V}(p)$ is called the *visibility kernel* of p, denoted as $\mathcal{VK}(p)$ (See Figure 1). The importance of the visibility kernel is that if a guards sees p, then it is guaranteed to see all point in $\mathcal{VK}(p)$, and there are no other points in P that satisfies this guarantee.

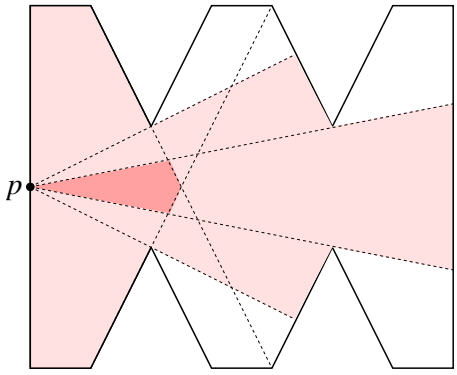


Figure 1: $\mathcal{V}(p)$ is shown as the union of shaded areas. $\mathcal{VK}(p)$ is the dark shaded area. Point p sees past all reflex vertices, two of them on the left, two of them on the right.

Let p be a point on P. $E(p)$ denotes the set of edges of P of which p sees at least one interior point. $R(p)$ denotes the set of reflex vertices of P that are seen past from p. Then $\mathcal{VK}(p)$ is calculated as the intersection of $O(n)$ half-planes [6] :

$$\mathcal{VK}(p) = \bigcap_{v \in R(p)} l^c(p, v) \cap \bigcap_{e \in E(p)} l^c(\overrightarrow{e}) \qquad (1)$$

2.1 Dominant guard set

A point $p \in P$ is called a *dominant point*, if $\nexists q \in P$ such that $\mathcal{V}(p) \subset \mathcal{V}(q)$. The *dominant guard set* for P, denoted as $\mathcal{DG}(P)$, is the set of all dominant points in P (See Figure 2).

*CISE Department, University of Florida, Gainesville [ayaz,ungor]@cise.ufl.edu

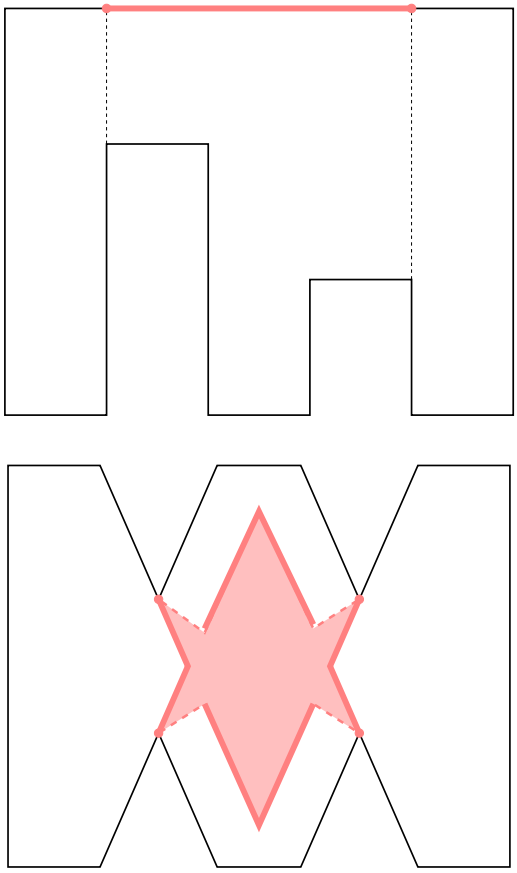


Figure 2: Thick lines and shaded areas indicate $\mathcal{DG}(P)$

For star-shaped polygons, dominant guard set consists of the points in the kernel of the polygon. Since all of the kernel points have the same visibility polygon, which is P itself, a point in the polygon is dominant only if that point is in the kernel. We can calculate the kernel of a star-shaped polygon with an $O(n)$-time algorithm [9]. Therefore, we consider the case where P is not star-shaped as an input of our algorithm.

We rely on the visibility kernel concept to determine whether a point is a dominant point. If a point $q \in P$ is in the visibility kernel of another point $p \in P$, then any point that sees p also sees q. In other words, if $q \in \mathcal{VK}(p)$, then $\mathcal{V}(p) \subseteq \mathcal{V}(q)$. Then, we have the following lemma:

Lemma 1 **[1]** *A point $p \in P$ is a dominant point if and only if $\forall q \in \mathcal{VK}(p)$ we have $\mathcal{V}(p) = \mathcal{V}(q)$.*

Consider the following arrangement, denoted as $\mathcal{A}(P)$, consisting of three types of line segments:

1. Extensions of the edges at reflex vertices.

2. Extensions of line segments between two reflex vertices that see each other.

3. The line segments between two reflex vertices that see past left each other or see past right each other.

Maximal contiguous regions that do not include line segments of the subdivision and the boundary of P are called the *cells of* $\mathcal{A}(P)$. The intersection points of the line segments among themselves and with ∂P are called the *vertices of* $\mathcal{A}(P)$ and the open line segments between two vertices of $\mathcal{A}(P)$ are called the *edges of* $\mathcal{A}(P)$. (See Figure 3). The complexity of the vertices, edges and cells of $\mathcal{A}(P)$ is at most $O(n^4)$.

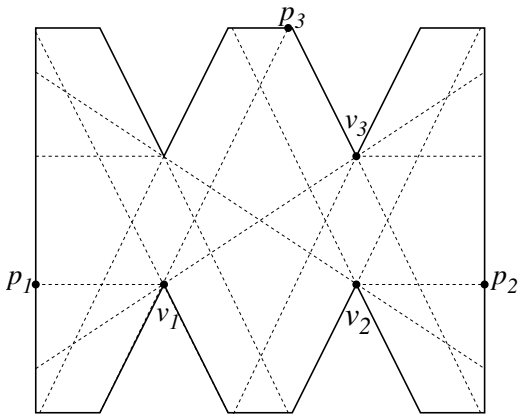


Figure 3: Subdivision of a polygon P, $\mathcal{A}(P)$. $\overline{v_1p_3}$ is of Type 1, $\overline{v_1p_1}$ and $\overline{v_2p_2}$ are of Type 2, $\overline{v_1v_3}$ is of Type 3.

The cells and the edges have the same dominancy properties, which means if a point in a cell (or on an edge) of $\mathcal{A}(P)$ is a dominant point, then all points in that cell (on that edge) are also dominant points [1]. Our algorithm relies on the arrangement construction as well as the following lemma.

Lemma 2 **[1]** *Let p be a dominant point in a polygon P. Then, at least one of the following statements is correct:*
(i) $\mathcal{VK}(p) = \{p\}$.
(ii) *p is in the kernel of P.*
(iii) *p is on a Type 3 line segment $\overline{r_1r_2}$ and $\mathcal{VK}(p) \cap \{r_1, r_2\} = \emptyset$.*

3 The algorithm

We first review the algorithm presented in [1]:
1. Calculate $\mathcal{A}(P)$ in $O(n^4)$ time using the algorithm described in [4].
2. For each cell of $\mathcal{A}(P)$, choose an arbitrary point p in the cell. Then:
2.1 Calculate $\mathcal{VK}(p)$ by calculating $\mathcal{V}(p)$ [8] and the kernel of $\mathcal{V}(p)$ [9] in linear time.
2.2 If $\mathcal{VK}(p) = \{p\}$, then the whole cell is in $\mathcal{DG}(P)$. Otherwise, none of the points in the cell is in $\mathcal{DG}(P)$.
3. Use step 2 for the Type 1 and 2 line segments of

$\mathcal{A}(P)$.
4. Choose an arbitrary point p on each arrangement edge on a Type 3 line segment $\overline{r_1 r_2}$. Then:
4.1 Calculate $\mathcal{VK}(p)$ in linear time.
4.2 If $\mathcal{VK}(p) = \{p\}$, then all points on the edge are in $\mathcal{DG}(P)$.
4.3 If $\mathcal{VK}(p) \subset \overline{r_1 r_2}$ and $r_1, r_2 \notin \mathcal{VK}(p)$, then all the points on $\mathcal{VK}(p)$ are dominant points (with identical visibility polygons, see Figure 4).

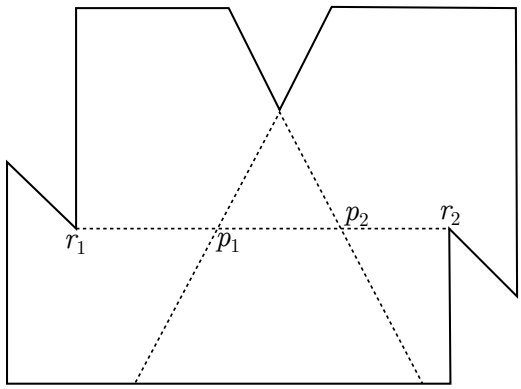


Figure 4: All points on the line segment $\overline{p_1 p_2}$ are dominant. Their visibility kernels are $\overline{p_1 p_2}$ and their visibility polygons are identical.

To improve the running time of the algorithm, we use the similarity of the properties of neighboring cells to calculate the dominant guard set. Below we explain these properties.

For calculating whether a point p in a cell that has the property $\mathcal{VK}(p) = \{p\}$, we do not need to calculate the visibility kernel directly. We only need to find a constant number of the reflex vertices p sees past and calculate the intersection of the half planes induced by these reflex vertices according to the equation (1).

Another observation we use is that two points in the neighbor cells see past the same reflex vertices except one. The counter-clockwise order of the reflex vertices two points in the neighbor cells sees past left (right) is also the same. A Type 1 line segment seperates two neighbor cells by seeing past property. Points on one side of the line segment see past the reflex vertex that induces the Type 1 line segment and the points on the other side only see it. A Type 2 line segment separates two neighbor cells by the visibility. Points on one side see both the reflex vertices that induce the Type 2 line segment and points on the other side see only one of those reflex vertices. A Type 3 line segment may change the orientation of the visibility kernel with respect to two points on neighbor cells. If the largest angle α between a point p and two consecutive reflex vertices p sees pass is less than π, then $\mathcal{VK}(p) = \{p\}$, i.e., it is zero dimensional. If α is greater than π, $\mathcal{VK}(p)$ can be two dimensional. If $\alpha = \pi$, then $\mathcal{VK}(p)$ can be at most one dimensional which should be taken care of seperately.

We use four balanced binary search trees (BBST) to record the reflex vertices a point p in a cell sees past: $UpperLeft$ records the reflex vertices with higher x coordinates p sees past left. $LowerLeft$ similarly records the reflex vertices with lower x coordinates p sees past left. $UpperRight$ and $LowerRight$ are the symmetric versions of the previous two BBSTs. The nodes of each BBST are in the counter-clockwise order with respect to the point they are seen by. For each BBST, extracting minimal and maximal elements take $O(\log n)$ time as well as inserting, removing and searching a reflex vertex.

For a point p in a cell, $\mathcal{VK}(p) = \{p\}$ if and only if the intersection of the half planes induced by the minimal and maximal elements of each BBST is also equal to p.

To make our arrangement suitable for the BBSTs, we insert horizontal line segments next to each reflex vertex until they hit the boundary on the left and on the right and call this Type 4 line segments. These line segments are in the order of n which do not change the complexity of the arrangement which is $O(n^4)$.

Then, starting from an arbitrary cell, preferably a boundary cell, we fill the four BBSTs for that cell. Then, we use breath first search (BFS) from that cell to spread to the neigboring cells (the cells that have a common arrangement edge). We manipulate the BBSTs while traversing the cells according to the type of the arrangement edge between the cells. Here are the actions for each type:

- Type 1 & 2: If the reflex vertex the newly visited cell starts (stops) to see past is on the $UpperLeft$, insert (remove) that reflex vertex to (from) $UpperLeft$. Do the same for the other three BBSTs.
- Type 3: No update is needed on BBSTs.
- Type 4: Remove the reflex vertex that induced by the line segment from $UpperLeft$ ($LowerLeft$) BBST and insert it to the $LowerLeft$ ($UpperLeft$) BBST. Do the same for the symmetric cases, $LowerRight$ and $UpperRight$.

After each action, we need to recalculate the intersection of (at most) eight half planes. If the intersection consists of only one point, then the whole cell is in Dominating Guards Set. The boundary of the cells are also included except the $\mathcal{A}(P)$ edges on Type 3 line segments on the boundary of the cell.

We do the same analysis on the $\mathcal{A}(P)$ edges on the Type 3 line segments according to Lemma 2. The number of $\mathcal{A}(P)$ edges is also $O(n^4)$, so the complexity does not change.

Each action takes $O(\log n)$ time. Since there are $O(n^4)$ cells, the total time complexity is $O(n^4 \log n)$.

References

[1] E. S. Ayaz and A. Üngör. An iterative refinement scheme of dominating guards and witnesses for art gallery problems. In *Canadian Conference on Computational Geometry (CCCG), Book of Abstracts*, pages 168–174, 2016.

[2] A. Bottino and A. Laurentini. A nearly optimal sensor placement algorithm for boundary coverage. *Pattern Recognition*, 41(11):3343–3355, 2008.

[3] S. Carlsson, H. Jonsson, and B. J. Nilsson. Finding the shortest watchman route in a simple polygon. *Proc. Algorithms and Computation: 4th International Symposium (ISAAC)*, pages 58–67, 1993.

[4] B. Chazelle and H. Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *J. ACM*, 39(1):1–54, Jan. 1992.

[5] W.-P. Chin and S. Ntafos. Shortest watchman routes in simple polygons. *Discrete & Computational Geometry*, 6(1):9–31, 1991.

[6] K. Chwa, B. Jo, C. Knauer, E. Moet, R. van Oostrum, and C. Shin. Guarding art galleries by guarding witnesses. *Int. J. Comput. Geometry Appl.*, 16(2-3):205–226, 2006.

[7] P. J. de Rezende, C. C. de Souza, S. Friedrichs, M. Hemmer, A. Kröller, and D. C. Tozoni. *Engineering Art Galleries*, pages 379–417. Springer International Publishing, Cham, 2016.

[8] B. Joe and R. B. Simpson. Corrections to lee's visibility polygon algorithm. *BIT Numerical Mathematics*, 27(4):458–473, 1987.

[9] D. T. Lee and F. P. Preparata. An optimal algorithm for finding the kernel of a polygon. *J. ACM*, 26(3):415–421, July 1979.

Perfect Polygons*

Hugo A. Akitaya† Erik D. Demaine‡ Martin L. Demaine‡ Adam Hesterberg‡
Joseph S.B. Mitchell§ David Stalfa¶

For a given polygon P (possibly with holes), we let $g(P)$ denote the minimum number of guards (points within P) in a visibility cover of P, so that every point of P is seen (via a straight line segment within P) by one of the $g(P)$ guards. The Art Gallery Problem is to compute $g(P)$; this is a known NP-hard problem, even for a simple polygon P (with no holes). The Art Gallery Theorem gives a worst-case tight upper bound on $g(P)$ in terms of n, the number of vertices of P, and h, the number of holes of P. For simple polygons ($h = 0$), the classic result is that $g(P) \leq \lfloor n/3 \rfloor$, while for a polygon with h holes, the extended result is that $g(P) \leq \lfloor (n+h)/3 \rfloor$; in both cases, the bounds are tight, as examples exist that achieve the upper bounds.

The *witness number*, $w(P)$, of P is the maximum number of *independent* witness points that can be packed into P, where two points $p \in P$ and $q \in P$ are independent if their visibility polygons, $VP(p)$ and $VP(q)$, are disjoint (so that no single guard in P can see both p and q). The use of $w(P)$ was introduced in [1], where it was used for finding lower bounds on $g(P)$, since clearly $g(P) \geq w(P)$, which can help in evaluating heuristic methods for computing small guard sets. We say that a polygon P is *perfect* if $w(P) = g(P)$. (There are examples of simple polygons having $n = 6$ vertices that are not perfect, with $g(P) = 2$, $w(P) = 1$. In general, there can be a large gap, with $w(P) = 1$ and $g(P) = \Omega(n)$, for simple polygons P.)

In this paper, we study the complexity of the problem of determining if a given polygon P is perfect. We study the problem in a very special kind of polygon, namely a connected union of non-overlapping line segments. We call such polygons *thin* (see Fig. 1). We show that the problem is in P if P is the union of line segments of 2 different orientations. We also show that the problem already becomes NP-hard if P is the union of line segments of 3 different orientations.

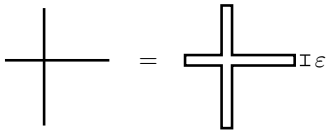


Figure 1: Thin polygon whose interior is the union of line segments in the limit when $\varepsilon \to 0$.

Theorem 1. *Deciding whether a given thin polygon P whose interior is the connected union of n line segments of 2 different orientations can be done in $O(n^{2.5})$ time.*

Proof. We show that we can compute $w(P)$ and $g(P)$ in polynomial time by reducing to problems in bipartite graphs. Create a graph G in which vertices are line segments that form P, and two vertices are connected by an edge if the corresponding line segments intersect. Clearly, G is bipartite, because there are only two orientations of line segments and a segment can only intersect segment in a different direction.

The optimal witness set contain points whose visibility is a single line segment, i.e., non-intersection points, or else we could move a witness point at a intersection without affecting the optimality of the solution. By choosing to place a witness in a line segment we create a constrain enforcing that intersecting line segments should not contain another witness point. Hence, finding $w(P)$ corresponds to finding the maximum independent set in G, which can be done in $O(n^{2.5})$ time [2].

Similarly, we can assume that, in an optimal guard set, every guard is placed at intersection points, or else we can move a guard at a non-intersection point and increase its visibility region. The visibility region of a guard is then the union of line segments that contain the intersection point on which that guard lies.

*Research on this paper was supported in part by the NSF awards CCF-1422311 and CCF-1423615, and the Science Without Borders scholarship program.

†Department of Computer Science, Tufts University, Medford, MA

‡CSAIL, MIT, Cambridge, MA

§Department of Applied Mathematics and Statistics, State University of New York at Stony Brook, Stony Brook, NY

¶Department of Computer Science, Northeastern University, Boston, MA

Hence, finding $g(P)$ corresponds to finding the minimum edge cover of G, which can be done in $O(n^2)$ time by greedily expanding a maximal matching. □

Theorem 2. *Deciding whether a given polygon P is perfect is NP-hard.*

Proof sketch. We reduce from 3SAT-3 which is NP-hard [3] An instance of 3SAT-3 consists of a set $V = \{x_1, \ldots, x_n\}$ of n variables and a boolean formula ϕ in 3CNF with m clauses, each of the form $(l_i \vee l_j \vee l_k)$ where each *literal* l_w, $w \in \{i, j, k\}$ is a copy or a negated copy of x_w. Additionally, a variable appears in at most 3 3SAT asks whether there exist an assignment from V to $\{\texttt{true}, \texttt{false}\}$ so that ϕ evaluates to `true`. Every 3SAT instance can be represented as a bipartite graph between V and the set of clauses. We transform a drawing of such graph into a thin polygon P that is perfect if and only if the 3SAT instance admits a positive solution.

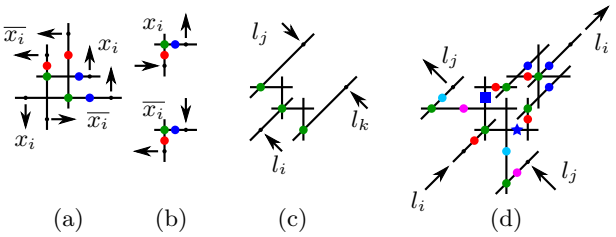


Figure 2: (a) Variable, (b) wire, (c) clause, and (d) crossover gadgets. The arrows indicate the direction from a variable to an adjacent clause. Black dots indicate the intersection point between gadgets.

Consider the planarization of such drawing by replacing every edge crossing by a vertex. Refer to Fig. 2. We replace every variable vertex, clause vertex, and crossing vertex by a variable, clause, and crossover gadgets respectively. Clauses incident to only two variables are obtained by omitting the line segment labeled l_i in Fig. 2(c). The drawing must be scaled so that no two gadgets intersect. In Fig. 2, green dots represent an optimal guard set while blue/cyan or red/magenta dots represent possible choices for a locally optimal witness set. Small black dots represent where the corresponding gadget intersect a wire gadget and vice versa. Then, connect two gadgets whose corresponding vertices are adjacent using a chain of one or more copies of the wire gadget, so that gadgets only intersect at the relevant black dots. Each wire gadget can be scaled and/or reflected in axis-aligned directions. Note that the wire gadget is the same for positive and negative literals, but using a different orientation as shown in Fig. 2(b). Fig. 2(a) shows the black dots that can intersect wires carrying positive or negative literals. Since every variable vertex is max-degree-3, we can always assign degree(x_i) black dots where the wire gadgets will intersect the variable gadget to a correspondent edge in the drawing incident to x_i so that the wires occur in the same circular order around the variable gadget as the edges occur around x_i. An example of a complete reduction is shown in Fig. 3.

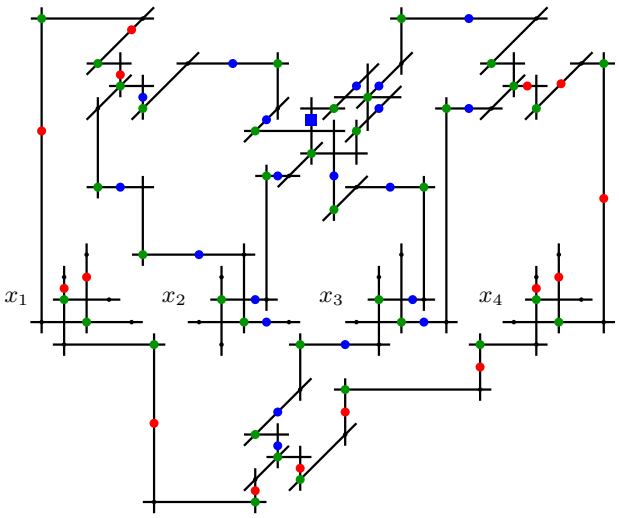


Figure 3: Reduction from the instance $\phi = (x_1 \vee \overline{x_2} \vee x_3) \wedge (x_2 \vee x_3 \vee x_4) \wedge (\overline{x_1} \vee \overline{x_3} \vee \overline{x_4})$.

Notice that the green point set is the global optimal for the guard set, because each gadget individually requires at least the same amount of guards as the number of green dots on them, and no such required guard can be shared with an adjacent gadget, except for the wire gadget. However, since a chain of wire gadgets connect two other gadgets, sharing guards between different adjacent wire gadgets is never beneficial. The local choice in the witness set encode the boolean value of variables. Next, we show that the produced instance is a perfect polygon if and only if the 3SAT-3 instance is satisfiable.

First assume that P is a perfect polygon. Notice that no line segment is completely visible to more than one guard. Then, every guard can see at most one witness since the witness set is independent. Since P is perfect, every guard sees exactly one witness. For each variable gadget, if a guard sees a witness in a horizontal line segment, then the other guard in the gadget should also see a witness in a horizontal segment, or else the witness set is not independent. The same argument applies for a witness in a vertical line segment. The wire gadgets incident

to viable sets should then contain a witness in a horizontal or vertical segment according to the variable gadget. If a wire corresponds to a positive (resp., negative) literal, then its witness appear before its guard using the direction from variables to clauses if the adjacent variable has its witnesses in vertical (resp., horizontal) line segments. We say that such wires carry a `true` value. For a crossover gadget shown in Fig. 2(d), if the wire attached to the upper left black dot carries a `true` value, then the leftmost guard sees a witness in the horizontal segment and, therefore, the bottommost guard sees a witness in the segment at 45°. Then, the wire attached to the bottom right black dot also carries a `true` value. Similarly, we show that if the wire attached to the upper right carries a `true` value, then so does the wire attached to the lower left. If the above case happens, then the guard at the three-way intersection must see a witness in either a horizontal or vertical segment. Then there must exist a witness in each of the three segments containing a red dot and, hence, the lower left wire also carries a `true` value. Using similar arguments, for every clause gadget, the guard at the three-way intersection must see a witness and then there must exist at least one wire incident to the gadget that carries a `true` value. Then, we can use the witness set of P to obtain a truth assignment that satisfies the 3SAT-3 instance.

Now, assume that the 3SAT-3 instance is satisfiable. For every variable x_i assigned `true` (resp., `false`) place a witness at every red (resp., blue) point in Fig. 2(a). Do the same for every wire gadget carrying a literal of such variable. For every crossover gadget involving literals of x_i and x_j as in Fig. 2(d), place a witness at every magenta point if the literal l_j evaluates `true`. Otherwise, place a witness at every cyan point. If l_i evaluates `true`, place a witness at every red point. Else, if l_j evaluates `true` place a witness at every point marked with a blue star and dots. Else, place a witness at every point marked with a blue square and dots. For every clause gadget, choose an incident literal that evaluates to `true`. Place a witness in the line segment that intersects the wire gadget of the corresponding literal. If the chosen literal is l_j (resp., l_k), place a witness in the vertical (resp., horizontal) line segment containing the three-way intersection. For each of the two guards placed at two-way intersections in the clause gadget, place a witness in one of the incident line segments so that the visibility polygons of witness are disjoint. By construction the set of witness is independent and every guard sees exactly one witness. Hence, P is perfect. □

References

[1] Amit, Yoav, Joseph SB Mitchell, and Eli Packer. Locating guards for visibility coverage of polygons. *International Journal of Computational Geometry & Applications* 20(05), pp.601-630.

[2] Clark, Brent N., Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discrete mathematics* 86(1-3), pp.165-177.

[3] Papadimitriou, Christos, and Mihalis Yannakakis. Optimization, approximation, and complexity classes. In *Proceedings of the twentieth annual ACM symposium on Theory of computing.* (pp. 229-234). ACM.

Efficient Approximations for the Online Dispersion Problem *

Jing Chen† Bo Li† Yingkai Li†
†Department of Computer Science, Stony Brook University
Stony Brook, NY 11794, USA
{jingchen, boli2, yingkli}@cs.stonybrook.edu

Abstract

The *dispersion* problem has been widely studied in computational geometry and facility location, and is closely related to the packing problem. The goal is to locate n points (e.g., facilities or persons) in a k-dimensional polytope, so that *they are far away from each other and from the boundary of the polytope.* In many real-world scenarios however, the points arrive and depart at different times, and decisions must be made without knowing future events. Therefore we study, for the first time in the literature, the *online dispersion* problem in Euclidean space.

There are two natural objectives when time is involved: the *all-time worst-case* (ATWC) problem tries to maximize the minimum distance that ever appears at any time; and the *cumulative distance* (CD) problem tries to maximize the integral of the minimum distance throughout the whole time interval. Interestingly, the online problems are highly non-trivial even on a segment. For cumulative distance, this remains the case even when the problem is time-dependent but offline, with all the arriving and departure times given in advance.

For the online ATWC problem on a segment, we construct a deterministic polynomial-time algorithm which is $(2\ln 2+\epsilon)$-competitive, where $\epsilon > 0$ can be arbitrarily small and the algorithm's running time is polynomial in $\frac{1}{\epsilon}$. We show this algorithm is actually *optimal*. For the same problem in a square, we provide a 1.591-competitive algorithm and a 1.183 lower-bound. Furthermore, for arbitrary k-dimensional polytopes with $k \geq 2$, we provide a $\frac{2}{1-\epsilon}$-competitive algorithm and a $\frac{7}{6}$ lower-bound. All our lower-bounds come from the structure of the online problems and hold even when computational complexity is not a concern. Interestingly, for the offline CD problem in arbitrary k-dimensional polytopes, we provide a polynomial-time black-box reduction to the online ATWC problem, and the resulting competitive ratio increases by a factor of at most 2. Our techniques also apply to online dispersion problems with different boundary conditions.

*An extended abstract of this paper appeared at the 44th International Colloquium on Automata, Languages, and Programming (ICALP'17). We thank Joseph Mitchell for motivating us to study the online dispersion problem. We thank Esther Arkin, Michael Bender, Rezaul A. Chowdhury, Jie Gao, Joseph Mitchell, Jelani Nelson, and the participants of the Algorithms Reading Group at Stony Brook for helpful discussions, and several anonymous reviewers for helpful comments. A full version is available at http://arxiv.org/abs/1704.06823.

Edge conflicts can increase along minimal rotation-distance paths

Sean Cleary* Roland Maio†

Abstract

There are no known efficient algorithms to calculate distance in the one-skeleta of associahedra, a problem that is equivalent to finding rotation distance between rooted binary trees or the flip distance between polygonal triangulations. One measure of the difference between trees is the number of conflicting edge pairs, and a natural way of trying to find short paths is to minimize successively this number of conflicting edge pairs using flip operations in the corresponding triangulations. Though it is quite common for the number of conflicts to decrease along a geodesic (a minimal length path) connecting two triangulations, we describe examples that show that the number of such conflicts does not always decrease along geodesics. Thus, a greedy algorithm that always chooses a transformation that reduces conflicts will not produce a geodesic in all cases. Further, there are examples of increasing size showing that the number of conflicts can increase by any specified amount.

*Dept. of Math, City College of New York, City University of New York, cleary@sci.ccny.cuny.edu

†Dept. of Computer Science, City College of New York, City University of New York, rolandmaio38@gmail.com

1 Introduction

Rooted binary trees arise across a range of areas, from phylogenetic trees representing genetic relationships to efficient organizational structures in large datasets. In the setting of rooted binary trees with a natural right-to-left order, such as those corresponding to binary search trees, a widely-considered distance is rotation distance. The *rotation distance* $d(S,T)$ between two trees S and T is the minimum number of rotations needed to transform the tree S to the tree T. There is a natural bijection between rooted binary trees with n internal nodes (or $n+1$ leaves) and triangulations of a marked regular polygon with $n+2$ sides. The operation that corresponds to rotation at a node for binary trees corresponds to an edge-flip between triangulations. Thus, rotation distance between two trees is exactly equivalent to a corresponding edge-flip distance between triangulations. The associahedron of size n is a combinatorial object capturing many aspects of Catalan objects, including triangulations, trees, and bracketed expressions. Here, we work entirely in the one-dimensional skeleton and neglect the higher dimensional faces in the face lattices of associahedra.

There are no known polynomial-time algorithms to find shortest paths in associahedra graphs or to find distances between vertices in those graphs. There are a number of approx-

imation algorithms [1, 4] for rotation distance. Two edges are said to be *conflicting* if it is not possible for them to be part of the same triangulation, due to the two edges crossing.

Here, we investigate connections between edge conflicts and finding geodesics in the setting of ordered trees, or equivalently of triangulations of polygons. We describe example triangulation pairs of size 8 and larger where there is no geodesic along which the total number of conflicts between the triangulations uniformly decreases or even remains the same, ruling out the success of greedy algorithms. Further, we show that the required increase of conflicts along a geodesic can grow to any specified increase k for triangulation pairs of size $k + 7$ and larger.

2 Background and definitions

We consider the marked regular $(n + 2)$-gon P with edges labelled as R for a marked root edge and then consecutively counter-clockwise from 0 to n. A *triangulation* T of P is a collection of $n - 1$ non-crossing edges from vertices of P that separate P into n triangles. An *edge flip on* T is the process of taking two triangles that share an interior edge, thus forming a quadrilateral Q, and replacing the interior diagonal of Q that lies in T with the other diagonal of Q to form a new triangulation T'. For each n, the relevant *associahedron graph* is the graph whose vertices are triangulations of the regular $(n+2)$-gon and whose edges connect vertices that differ by a single edge flip. The *edge flip distance* from a triangulation S to a triangulation T of the same size is the minimal length path in the associahedron graph. Alternatively, we can regard the vertices of the size n associahedron graph as rooted binary trees of size n interior nodes with $n+1$ leaves numbered from 0 to n, with the edges connecting vertices whose trees differ by a single rotation (left or right) at an interior node. To any triangulation, there is a well-known natural dual construction giving rise to the tree description.

Remarkable work of Sleator, Tarjan and Thurston [6] showed that the upper bound for distance between vertices for $n \geq 11$ is $2n - 6$, and furthermore that that upper bound is realized for all n larger than some very large N. Recent work of Pournin [5] showed that in fact the upper bound is realized for all $n \geq 11$. There are no known polynomial-time algorithms to compute rotation distance, although rotation distance has been shown to be fixed parameter tractable by Cleary and St. John [3] and there are a number of approximation algorithms.

Two edges s and t are said to be *conflicting* if it is not possible for them to be present in the same triangulation; equivalently, if the edges cross as chords connecting vertices in the relevant polygon. For example, a chord from vertex 5 to 8 is in conflict with a chord from vertex 3 to 6, which can be seen as one of the intersections of the red chords and blue chords in Figure 1 if we number the vertices as the counterclockwise ends of the numbered edges. The number of conflicts for a pair of triangulations S and T is the sum of the conflicts between the pairs of edges, which gives a semi-metric on the set of triangulations.

3 Conflicts along geodesics

There are a range of conflict-based greedy algorithms to reduce conflicts to try to find a geodesic from a given triangulation S to a given

target triangulation T.

Such algorithms will always give a path from S to T, and this results in an upper bound for the edge-flip distance. However, this path is not necessarily a geodesic path so the distance from S to T may be less than various greedy conflict-based algorithms may find. There are multiple ways in which such a greedy algorithm to reduce conflicts can fail to find a geodesic. It may be that there are several choices among the neighbors of S_i with the same number of conflicts, and at least one of them does not begin a geodesic path from S_i to T. Or it may be that none of the neighbors with minimal conflicts begins a geodesic path. Broadening the notion to allow choices where the number of conflicts is not necessarily minimal but merely equal or smaller than the current number of conflicts gives many more possibilities. But even algorithms that consider reducing conflicts (not necessarily minimally) or keeping conflicts constant will not always find a geodesic path because of examples of the following type:

Theorem 1 *For any n at least 8, there are examples of triangulation pairs (S,T) of size n where every geodesic γ from S to T with $\gamma = \{S = S_0, S_1, S_2, \ldots, S_k = T\}$ has the property that S_1 has more conflicts with T than S has with T.*

Thus, any algorithm that attempts to find geodesics by either minimizing conflicts, at least reducing conflicts, or keeping conflicts at least non-increasing will not find any geodesic from S to T.

This particular example is not symmetric, in that if we proceed instead from T toward S reducing conflicts, we do find a geodesic. However, we can construct triangulation pair examples that do have this property from both ends.

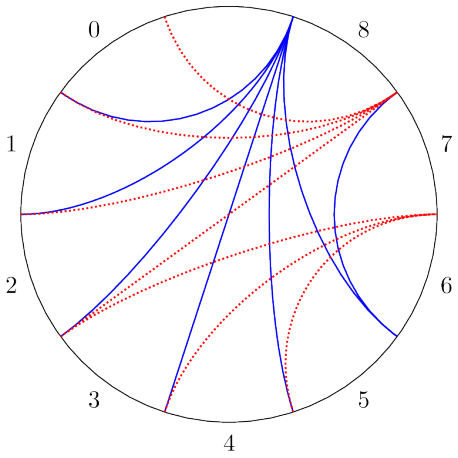


Figure 1: The triangulations used to prove Theorem 1 superimposed, with S in blue and T in dotted red. There are 27 conflicts in the pair (S,T) which can be seen as the 27 intersections between the triangulations.

Theorem 2 *There are examples of triangulation pairs (U,V) where for all geodesic paths γ from U to V or from V to U there is a step where the number of conflicts increases along γ.*

The particular triangulation proving Theorem 1 shows that conflicts can rise along geodesics by one. In fact, for increasingly large triangulation pairs, conflicts along geodesics can rise by an arbitrary amount. We use methods similar to analyses in Cleary, Rechnitzer, and Wong [2] and Cleary and St. John [4] to bound distances sharply. A *one-off* edge in S in a triangulation pair (S,T) is an edge that is not common to S and T but which flips directly to an edge in T. By [6], for any one-off edge, there is a geodesic from S to T that begins with that edge flip. A triangulation pair of size n (with $n-1$ edges)

with no common edges and no one-off edges must have distance at least n as each edge flip can create at most one new common edge, and if there are no one-off edges present then it will take at least n steps to transform S to T. With this we can show that conflicts can rise along a geodesic by any specified amount.

Theorem 3 *For any positive k, there are examples of triangulation pairs (S,T) of size $n \geq k+7$ where every geodesic γ from S to T with $\gamma = \{S = S_0, S_1, S_2, \ldots, S_k = T\}$ has the property that S_1 has k more conflicts with T than S has with T.*

Asymptotically, the fact that the number of conflicts can rise by $n-7$ is remarkable in light of the fact that the greatest number of conflicts a single edge can have is $n-1$ if it crosses all other edges in the other triangulation.

Again, we can concatenate two triangulations to create larger more symmetric examples where the number of conflicts must increase in either direction by any specified amount.

Theorem 4 *For any positive k and any $n \geq 2k+14$, there are examples of triangulation pairs (S,T) of size n where every geodesic γ from S to T or from T to S has a step along the geodesic where the number of conflicts increases by k.*

Thus, there are large pairs where the number of conflicts along geodesics must rise in a single step an amount proportional to the size for geodesics in either direction.

We do note that this kind of behavior where every geodesic has an increase in conflicts along the path somewhere are somewhat rare, with these 28 equivalence classes of conflict-increasing examples of size 8 occurring from among the 117,260 equivalence classes of edge-flip distance problems of size 8, for example.

References

[1] Jean-Luc Baril and Jean-Marcel Pallo. Efficient lower and upper bounds of the diagonal-flip distance between triangulations. *Information Processing Letters*, 100(4):131–136, 2006.

[2] Sean Cleary, Andrew Rechnitzer, and Thomas Wong. Common edges in rooted trees and polygonal triangulations. *Electron. J. Combin.*, 20(1):Paper 39, 22, 2013.

[3] Sean Cleary and Katherine St. John. Rotation distance is fixed-parameter tractable. *Inform. Process. Lett.*, 109(16):918–922, 2009.

[4] Sean Cleary and Katherine St. John. A linear-time approximation for rotation distance. *J. Graph Algorithms Appl.*, 14(2):385–390, 2010.

[5] Lionel Pournin. The diameter of associahedra. *Adv. Math.*, 259:13–42, 2014.

[6] Daniel D. Sleator, Robert E. Tarjan, and William P. Thurston. Rotation distance, triangulations, and hyperbolic geometry. *J. Amer. Math. Soc.*, 1(3):647–681, 1988.

Optimal Safety Patrol Scheduling Using Randomized Traveling Salesman Tour

Hao-Tsung Yang, Shih-Yu Tsai, Kin Sum Liu, Shan Lin, and Jie Gao

Stony Brook University

***Abstract*—In this paper, we consider the problem of designing schedules for a patroller to guard a given set of n sites in a strategic setting modeled as the attacker-defender game. It belongs to the general framework of security games and the main challenge here is to explicitly model the time spent moving between the sites and the scalability issue, as the strategy space contains exponentially many schedules even with a finite time horizon. The traveling salesman tour visits all sites with the highest frequency but is deterministic and could be exploited by the attacker. Random tours are most unpredictable but could be substantially inefficient in visiting the sites. Instead, we formulate the randomized TSP problem and provide solutions that achieve a nice tradeoff between frequency in visiting the sites and unpredictability for the strategic setting. We provide a rigorous analysis of the randomized TSP and show how to solve for the best defender strategies under this family of tours. Evaluations demonstrate the effectiveness of our solutions compared to other alternatives using both artificial data and a real-world crime dataset.**

INTRODUCTION

Public safety is crucial to everyday life. When responding to events of a criminal nature, it is necessary to consider game theoretic models and strategic behaviors. Modeling crimes by a game theory model is rooted at Nobel Prize laureate Gary Becker's theory [1]. The choice to commit a crime depends on the expected utility of the crime relative to the utility possibly obtained by using the time and resources at other activities. A perceived increase in the likelihood of being caught will deter certain offenders from engaging in criminal activities. This theory led to the adoption of policing plans that emphasize on preventative actions such as increased police presence, patrolling hotspots, among others [2].

Incorporating strategic behaviors in fighting against crimes is first done in the paradigm-shifting research area of security games (see [3] for a survey of the recent work). In this setting the security problem is modeled as a Stackelberg game, consisting of a defender with a limited set of resources to protect a set of targets and an attacker who casts attacks after learning the defender's strategy and conducting careful planning. A pure strategy for the defender is a deterministic schedule of the resources. A pure strategy of an attacker is a specific attack on one target. It is clear that a pure strategy for a defender is not optimal as the attacker can exploit the strategy and launch an attack at the weakest link. Therefore a better strategy for the defender is to launch a *mixed strategy* which is a probabilistic distribution on pure strategies. In this setting, the goal is to compute a *Stackelberg equilibrium*, a mixed strategy for the defender that maximizes the defender's utilities, under the condition that the attacker knows the defender's strategy and chooses the best response strategy. The framework has been enriched by considering various extensions such as uncertainties and bounded rationality of adversaries, and has rendered a number of effective schedules in real-world settings for protecting airport, ports, harbors, and national forests. Despite the great success, in [3] three challenges were mentioned for further advancing security game, the first one being the issue of *scalability*.

Most of the settings in the studied security games are combinatorial in nature and the space of pure strategies for the defender is naturally all possible ways of allocating resources to the locations to be guarded. For example in one of the first application scenarios at the LAX airport, the strategy space is composed of combinatorial decisions of placing guards at checkpoints. The time needed for a guard to move between these checkpoints is neglected. In some other scenarios, the target is mobile but follows a fixed trajectory (along with a straight line) with a fixed schedule. The defender adopts mobile patrollers but the defender's strategy space is discretized and represented by a path in a graph [4]. In these work, the overhead of switching between these pure strategies (i.e., the police officer moves from one target location to another target location) is often ignored.

There were good reasons to ignore the overhead of switching between the pure strategies. Consider the airport guarding scenario. If the domain is small the time needed to move from one checkpoint to the other is indeed negligible compared to the time spent for guarding. Moreover,

When mobility is explicitly modeled the strategy space may be large. To handle the scalability issue most methods use heuristic functions with some reduction techniques which are largely dependent on the specific scenarios [7], [8], [6], [9]. For example, pure strategies were generated with columns independently to avoid the combinatorial explosion of the search space [10], [8]. But here the assumption is that the attacker has no ability to observe the defender's position during the patrol phase.

Our Contribution In this paper, we take a look at the family of patrol scheduling problems. Consider a campus setting when a police officer's task is to tour around campus parking lots and check for illegal parking. Many campuses, including ours, ask the police officer to follow a fixed schedule. And it is not surprising that users gradually observe the movement patterns and start to exploit that which hurts the effectiveness

of patrolling.

In our setting, we assume that the time spent moving is non-negligible. This is indeed the case with campus patrolling. We also assume that the attacker can observe both the strategies of the defender as well as current defender location during the patrol, as is used in [11], [12], [13]. Additionally, the attacker is flexible to decide when and where to launch the attack, for how long (i.e., the period of illegal parking) to maximize the attacker's utility.

When motion is explicitly modeled, finding the patrol routes is immediately related to the classical traveling salesman problem. On one hand, the shortest traveling salesman tour is the most efficient route – it visits each location most frequently, without considering the strategic aspect. But such a route is unique (when there is no degeneracy) and thus is totally predictable. Users can park for long hours right after the police officer leaves the location. On the other hand, we can imagine stochastic tours that are less predictable, but unless designed carefully, a random tour may be much longer than the optimal TSP tour. This leads to less frequent visits to each location and could hurt the system performance as well. Furthermore, the total number of possible schedules can be infinite if we consider an infinitely long time horizon. Even if we limit ourselves in periodic schedules, the total number of possible schedules can be exponential in the number of parking lots, which makes solving the Stackelberg game difficult. Therefore, an interesting problem is to consider how to balance the two aspects – that each location shall be visited sufficiently frequently, while the next location to visit by the police officer is maximally unpredictable. We call this problem the *randomized TSP* problem.

In this paper, we provide a few solutions for the randomized TSP and apply them to the safety patrol problem. For the randomized TSP, we consider two settings, when the locations have uniform weights or non-uniform weights. Here the weights are used to model the importance of a location or different safety requirements. In one of our evaluations, we use the crime rate at Denver, CO to generate the weights for each district. Regions of a higher crime rate are visited more frequently. This is realized by minimizing the weighted length of the interval between two consecutive visits to the same site as one of the evaluation metrics. The second evaluation metrics measures how predictable is the next site to visit (quantified by the entropy of the distribution of the next site to visit).

When the weights on the sites are the same, we modify the TSP tour and introduce randomness – by touring around TSP with a probability $1-p$ we may skip the next site. By varying p we show that there is a tradeoff between how frequent the sites are visited versus how predictable is the next site to visit, from the attacker's perspective. When the weights are not the same, we will show how to generate a tour that adjusts to the different weights. The final solution is a collection of subtours, where each tour is a TSP tour on sites of similar weights. The final patrol schedule is produced by first randomizing on the different TSP tours and on each subtour use random skipping to further reduce the chance to predict the next site. In this paper, we provide rigorous analysis on the two evaluation metrics for the family of tours we generate, with different p.

One of the nice things about the randomized TSP solutions is the compact representation. We run the security game optimization on this family of tours, which is parameterized by mainly a single parameter p. Depending on the payoff parameters of the security game (e.g., how much could one benefit from illegal parking for a unit of time, versus how much penalty one received with a parking ticket), the best strategies can be computed by choosing the best value p for this family. Intuitively, if the penalty of a ticket is higher relative to the benefit of illegal parking, the adversary will tend not to take chances, in order to maximize his/her expected payoff. This leads to the defender's strategy of taking a smaller value of p. That is, the probability of skipping the next city on the TSP tour is higher, which makes the expected time till the next visit to be longer but less predictable. If we consider the distribution of the time till the next visit, it will be fatter but lower. This will be more effective into 'scaring' the attacker away. Simulations show that there are no scalability issues in our algorithms and our solution is adaptive to various payoff functions using both artificial and real-world datasets. We also show that our solutions are much better (achieving reduced expected payoff for the attacker) than with other baselines such as the deterministic TSP solution, random walk or random permutations.

REFERENCES

[1] R. A. Posner, "The law and economics movement: from bentham to becker," *The Origins of Law and Economics: Essays by the Founding Fathers*, 2005.

[2] N. Billante, "The beat goes on: Policing for crime prevention," http://www.cis.org.au/IssueAnalysis/ia38/ia38.htm, 2003.

[3] T. H. Nguyen, D. Kar, M. Brown, A. Sinha, A. X. Jiang, and M. Tambe, "Towards a science of security games," in *New Frontiers of Multidisciplinary Research in STEAM-H*, B. Toni, Ed., 2016.

[4] F. Fang, A. X. Jiang, and M. Tambe, "Optimal patrol strategy for protecting moving targets with multiple mobile resources," in *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, ser. AAMAS '13. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2013, pp. 957–964. [Online]. Available: http://dl.acm.org/citation.cfm?id=2484920.2485072

[5] J. Pita, M. Jain, J. Marecki, F. Ordóñez, C. Portway, M. Tambe, C. Western, P. Paruchuri, and S. Kraus, "Deployed armor protection: the application of a game theoretic model for security at the los angeles international airport," in *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems: industrial track*. International Foundation for Autonomous Agents and Multiagent Systems, 2008, pp. 125–132.

[6] E. Shieh, M. Jain, A. X. Jiang, and M. Tambe, "Efficiently solving joint activity based security games," in *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*. AAAI Press, 2013, pp. 346–352.

[7] E. Shieh, B. An, R. Yang, M. Tambe, C. Baldwin, J. DiRenzo, B. Maule, and G. Meyer, "Protect: A deployed game theoretic system to protect the ports of the united states," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 2012, pp. 13–20.

[8] M. Jain, D. Korzhyk, O. Vaněk, V. Conitzer, M. Pěchouček, and M. Tambe, "A double oracle algorithm for zero-sum security games on graphs," in *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 2011, pp. 327–334.

[9] N. Basilico, N. Gatti, and F. Amigoni, "Patrolling security games: Definition and algorithms for solving large instances with single patroller and single intruder," *Artificial Intelligence*, vol. 184, pp. 78–123, 2012.
[10] M. Jain, E. Kardes, C. Kiekintveld, F. Ordónez, and M. Tambe, "Security games with arbitrary schedules: A branch and price approach." in *AAAI*, 2010.
[11] N. Basilico, N. Gatti, and F. Amigoni, "Leader-follower strategies for robotic patrolling in environments with arbitrary topologies," in *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 2009, pp. 57–64.
[12] B. Bošanskỳ, V. Lisỳ, M. Jakob, and M. Pěchouček, "Computing time-dependent policies for patrolling games with mobile targets," in *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*. International Foundation for Autonomous Agents and Multiagent Systems, 2011, pp. 989–996.
[13] Y. Vorobeychik, B. An, M. Tambe, and S. P. Singh, "Computing solutions in infinite-horizon discounted adversarial patrolling games." in *ICAPS*, 2014.
[14] K. S. Liu, T. Mayer, H. T. Yang, E. Arkin, J. Gao, M. Goswami, M. P. JohnsonS, N. KumarP, and S. Lin, "Joint sensing duty cycle scheduling for heterogeneous coverage guarantee."
[15] K. S. Liu, J. Gao, S. Lin, H. Huang, and B. Schiller, "Joint sensor duty cycle scheduling with coverage guarantee." in *MobiHoc*, 2016, pp. 11–20.
[16] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001.
[17] N. Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem," Graduate School of Industrial Administration, Carnegie Mellon University, Technical Report 388, 1976.
[18] J. Min and T. Radzik, "Bamboo garden trimming problem," in *SOFSEM 2017: Theory and Practice of Computer Science: 43rd International Conference on Current Trends in Theory and Practice of Computer Science, Limerick, Ireland, January 16-20, 2017, Proceedings*, vol. 10139. Springer, 2017, p. 229.

On the Density of Triangles with Periodic Billiard Paths

Ramona Fae Charlton
fae.charlton@gmail.com

Abstract

We consider the classic open problem of whether every triangle has a periodic billiard path. While this has been shown for rational-angled triangles [Masur86], the irrational case remains open. Finding periodic billiard paths is an easy exercise for acute triangles, a long computer-aided case analysis when the maximum angle is at most 100° [Schwartz08], and beyond that very little is known.

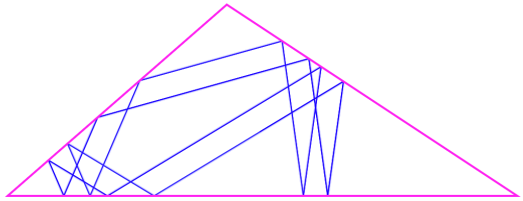
We examine the inequalities characterizing the set of triangles over which a given billiard path is periodic. We prove polynomial upper bounds on their rate of change, and use these bounds to derive positive radii within which a periodic billiard path must remain valid. We perform a computer search for periodic billiard paths on randomly selected triangles over a fixed rational grid, and use the radius bounds to show (under a Bayesian model with constant prior, assuming the uniformity of the selected grid points but otherwise unconditionally) that the likelihood that $\geq 98\%$ of all obtuse triangles admit a periodic billiard path is > 0.99999.

Background

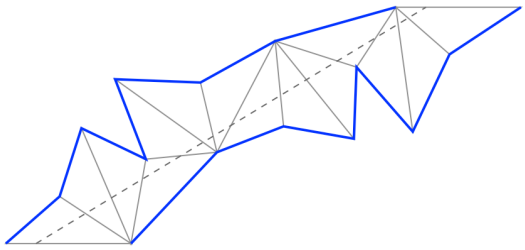
A natural way to model a billiard path on a triangle is as a straight line in the plane, along which the triangle is repeatedly reflected. A billiard path is then periodic if it ever reaches the same relative point on a triangle with the same orientation (see Figure 1). This is equivalent to the usual model, but is easier to visualize and compute with, so we will use it exclusively.

The basis on which the problem can be systematically analyzed is this: each reflection of the triangle through its edges changes the angle of the edges by integer multiples of the triangle's vertex angles. Inductively, any sequence of reflections will yield a triangle whose orientation differs by an integer linear combination of the vertex angles. If those angles are irrational, then the only way such a combination can equal zero – that is, the only way for two triangles to have the same orientation – is if the coefficients in the linear combination are also zero. That means that if a sequence of reflections produces a final triangle with the same orientation as the first, then the same will be true for *all* initial triangles, regardless of geometry. Since a periodic billiard path must pass through the interior of the reflection edges, and the vertices arising from reflection are continuous functions of the original triangle coordinates, any such path remains valid for all triangles in an open neighborhood.

This is a constructive argument: given a periodic billiard path on a particular triangle, one can derive the algebraic constraints on the triangle vertices such that a trajectory with the same combinatorial type is still a periodic path. [Schwartz08] explored this approach, exhibiting several paths and path families which remain periodic over a relatively large range, the union of which covered all triangles with maximum angle less than 100°.



(a) *A periodic billiard path as a ray reflected off the triangle's boundary*



(b) *The same path visualized by reflecting triangles along a straight line*

Figure 1

Unfortunately, this approach doesn't scale well: the combinatorial length of a periodic path increases with the maximum angle of the triangle. This increases the computation cost for analysis and, more importantly, dramatically decreases the radius within which the path is periodic, meaning the number of different cases required to substantially increase the current 100° bound is (probably) intractable.

Our Results

We derive new bounds on the derivatives of the constraint functions for periodic paths. Our bounds depend

polynomially on the combinatorial length of the path (i.e. the number of reflections), but not at all on the path itself. This means that given any periodic path on a triangle, and without any further analysis beyond a single evaluation of the constraint functions, we immediately have an explicit radius within which every triangle is guaranteed to have a periodic billiard path.

We apply these bounds experimentally to derive formal lower bounds on the density of triangles with periodic billiard paths, by showing that for a large fraction of triangles we can exhibit an explicit path.

Constraint bounds

An *edge path* E is a sequence $E_{1\ldots n}$ of n edge indices $\{1,2,3\}$ through which a triangle T is to be reflected. $\mathrm{Tri}_E(T,i)$ is the ith triangle in the reflected sequence when starting from T, defined inductively as $\mathrm{Tri}_E(T,0) = T$ and $\mathrm{Tri}_E(T,i) =$ the reflection of $\mathrm{Tri}_E(T,i-1)$ through edge E_i.

An edge path is *closed* if its final triangle $\mathrm{Tri}_E(T,n)$ is generically (for all T) of the same orientation as the first $\mathrm{Tri}_E(T,0)$. The *offset* $\overrightarrow{\mathrm{Off}}_E(T)$ of a closed edge path E on triangle T is the vector offset from the first triangle $\mathrm{Tri}_E(T,0)$ to the last one $\mathrm{Tri}_E(T,n)$.

Given a closed edge path, the question of whether it admits a periodic billiard path on a particular triangle can be answered by linear inequalities on the vertices of the reflected triangles: does any path parallel to $\overrightarrow{\mathrm{Off}}_E(T)$ remain strictly inside the union of the reflected triangles $\cup_i \mathrm{Tri}_E(T,i)$ (as in Figure 1b)? We choose to think of edge paths as starting from the base edge of an initial oriented triangle and proceeding upward, so we call the ordered vertices bounding the reflections the *left boundary* and *right boundary*, and denote the sequences by (Left_i) and (Right_i). While their precise coordinates depend on T, combinatorially they are a function only of the edge path itself. In particular, the length of each sequence depends only on E.

The *constraint functions* $\psi_{i,j}$ for a closed edge path E on a triangle T are polynomials computing (a positive multiple of) the margin between the left and right boundary vertices when moving along the path's offset:

$$\psi_{i,j}(E,T) = \langle \overrightarrow{\mathrm{Off}}_E(T)^{\perp}, \mathrm{Left}_i - \mathrm{Right}_j \rangle \qquad (1)$$

where i and j range over the lengths of the left and right boundary sets, and $\overrightarrow{\mathrm{Off}}_E(T)^{\perp}$ is the rotation of $\overrightarrow{\mathrm{Off}}_E(T)$ by $\pi/2$. There is a periodic billiard path on T through E if and only if $\psi_{i,j}(E,T) > 0$ for all i and j.

Given E and a T with rational vertex coordinates, the constraint functions can be evaluated explicitly to determine if E yields a periodic billiard path on T. Given only E, variables can be substituted for T's coordinates to yield generic polynomial constraints that are satisfied only on the set of triangle coordinates for which E gives a periodic billiard path. This is, in outline, the method underlying the results of [Schwartz08].

To simplify formulas in what follows we without loss of generality consider triangles whose longest edge (the base) lies on a fixed unit interval, so that any two triangles differ only in the coordinates of their apex. We then have:

Theorem 1. *Given triangles T_1, T_2 with apexes a_1, a_2 and a closed edge path E of length n, let ℓ_{min} be the length of the shortest edge of T_1 and define $\psi_{i,j}$ as in Equation 1. Then*

$$\left|\psi_{i,j}(E,T_1) - \psi_{i,j}(E,T_2)\right| \leq \frac{(n+2)^3 \|a_1 - a_2\|}{8\ell_{min}} \qquad (2)$$

for all i and j.

Corollary 1. *Given triangle T_1 with apex a_1 having a periodic billiard path along a closed edge path E of length n, let ℓ_{min} be the length of the shortest edge of T_1, and set $\psi_{min} = min_{i,j}[\psi_{i,j}(E,T)]$. Then every triangle T_2 with apex a_2 such that*

$$\|a_1 - a_2\| \leq \frac{8\ell_{min}\psi_{min}}{(n+2)^3} \qquad (3)$$

also has a periodic billiard path along E.

Given a periodic billard path on a triangle with rational vertices, Corollary 1 gives an explicit radius within which the apex of the triangle can be perturbed while preserving the billiard path.

Experimental Results

We now wish to obtain a lower bound on the fraction of obtuse triangles that admit periodic billiard paths. As above, we set the longest edge to be a fixed unit-length base. By symmetry, we also assume the shortest edge is the one lying clockwise of the base. Our problem space is thus the set of possible apexes within a radius-1/2 quarter-circle. With this parameterization, the "fraction" of obtuse triangles with periodic paths is taken to mean the relative measure within that quarter-circle of the set of apexes admitting a periodic path.

Our approach is to fix a rational grid covering the problem space and select apexes uniformly at random from the grid, then search for periodic billiard paths for the chosen apexes. If a path is found, and Corollary 1 shows that it remains valid up to a radius larger than the grid spacing, then all the triangles nearest to that grid point are guaranteed to have periodic billiard paths, and we call that apex a success. A high rate of success gives us a Bayesian lower bound on the probability that in fact a high proportion of all obtuse triangles admit periodic billiard paths.

Grid spacing	2×10^{-14}
Apex count	2500
Success count	2479

Path length	
Min	14
Max	28572
Avg	288.1

Path validity radius	
Min	1.45×10^{-14}
Max	1.3×10^{-3}
Mult. Avg	8.6×10^{-7}

Table 1: *Experimental results*

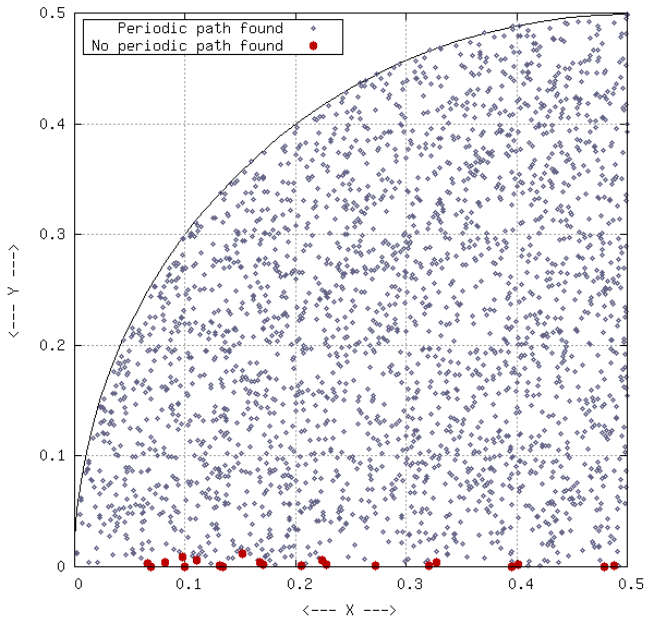


Figure 2: *A scatter plot of the trial outcomes. Successful points denote a cyclic billiard path that is valid up to a positive radius greater than the spacing of the apex grid.*

Once a periodic path is found, it is easy to independently verify its correctness and radius, so the probabilities we derive are conditioned only on the source of randomness used to generate the apex set. We used `libbsd`'s `arc4random` running on Ubuntu Linux 17.04, which we consider robust enough for problems of this nature, but it would be straightforward to reproduce the experiment using any desired source.

Our numeric results are in Table 1, with a scatter plot in Figure 2. Given the preceding, we consider this to be strong empirical evidence for the assertion that most obtuse triangles admit periodic billiard paths. Specifically, if we assume the uniformity of our test coordinates and compute the conditional probability of the success rate we have:

Confidence Bound 1. *Assuming a constant prior on the fraction of obtuse triangles that admit a periodic billiard path, and given an experimental outcome at least as good as the one in Table 1, the likelihood that* $\geq 98\%$ *of obtuse triangles admit a periodic billiard path is* > 0.99999.

Both the likelihood and the proportion of triangles may be readily increased by repeating the experiment with a higher trial count and search depth.

Proof Sketch

We now outline the proof of Theorem 1. Set $T = T_1$, $a = a_1$. As above, let (Left_i) and (Right_i) denote the left and right boundaries of T's reflections along E.

If there is a periodic billiard path, it must lie parallel to the edge path's offset $\overrightarrow{\text{Off}}_E(T) = \text{Left}_{\text{final}} - \text{Left}_0$. This means it lies perpendicular to $\overrightarrow{\text{Off}}_E(T)^\perp$, so given points p_1 and p_2 we can measure their mutual distance orthogonal to the offset via the dot product $\langle \overrightarrow{\text{Off}}_E(T)^\perp, p_2 - p_1 \rangle$, which is positive if and only if p_2 lies to the left of p_1 when traveling in the direction of $\overrightarrow{\text{Off}}_E(t)$. From this we get the constraint functions $\psi_{i,j}$ of Equation 1.

Let us telescope the boundary representation of the offset:

$$\begin{aligned} \overrightarrow{\text{Off}}_E(T) &= \text{Left}_{\text{final}} - \text{Left}_0 \\ &= \sum_{i=1}^{\text{final}} (\text{Left}_i - \text{Left}_{i-1}) \end{aligned} \tag{4}$$

The elements of this sum are edge vectors along the boundary. Focusing on these edges, we define:

$$\begin{aligned} \overrightarrow{\text{Left}}_i &= \text{Left}_i - \text{Left}_{i-1} \\ \overrightarrow{\text{Right}}_i &= \text{Right}_i - \text{Right}_{i-1} \end{aligned} \tag{5}$$

Every boundary vertex can then be expressed as a sum of edge vectors from the first points:

$$\begin{aligned} \text{Left}_i &= \text{Left}_0 + \sum_{j \leq i} \overrightarrow{\text{Left}}_j \\ \text{Right}_i &= \text{Right}_0 + \sum_{j \leq i} \overrightarrow{\text{Right}}_j \end{aligned} \tag{6}$$

Since the operation $\perp$ is linear, it commutes with this form, so we also have:

$$\overrightarrow{\text{Off}}_E(T)^\perp = \sum_i \overrightarrow{\text{Left}}_i^\perp = \sum_i \overrightarrow{\text{Right}}_i^\perp \tag{7}$$

Combining these gives us the following expression for the constraint functions:

$$\begin{aligned}\psi_{i,j}(E,T) = &\left\langle \sum_k \overrightarrow{\text{Left}}_k^{\perp}, \sum_{k\le i} \overrightarrow{\text{Left}}_k \right\rangle \\ &+ \left\langle \sum_k \overrightarrow{\text{Right}}_k^{\perp}, \sum_{k\le j} \overrightarrow{\text{Right}}_k \right\rangle\end{aligned} \tag{8}$$

This suggests many simplifications using the fact that the inner product of a vector with its orthogonal is zero. All we will use for now is that (since both i and j are bounded by the number of reflections n) we can expand both sides of the inner products by distributivity into a sum

$$\psi_{i,j}(E,T) = \sum_k^N \langle v_k^{\perp}, w_k \rangle \tag{9}$$

where the v_k, w_k are elements of $\overrightarrow{\text{Left}}$ or $\overrightarrow{\text{Right}}$ and N is at most $(n+2)^2/4$. This is not completely trivial: an $n^2/2$ bound is immediate, but the extra $1/2$ factor is obtained by more careful analysis of path length, which we omit here.

For the rest, we can rewrite Equation 9 in angle form:

$$\psi_{i,j}(E,T) = \sum_k^N \|v_k\|\|w_k\| \cos(\theta_k) \tag{10}$$

where θ_k is the angle between $v_k^{\perp}$ and w_k. Recall that the angle of edges produced by repeated reflection varies by integer combinations of the triangle's vertex angles $\alpha_{\{1,2,3\}}$, so $\theta_k = \pi/2 + \sum_i c_{k,i}\alpha_i$ for integers $(c_{k,i})$. Induction shows the difference in any two edge angles is at most $n/2$ individual rotations, so we may choose $\sum_i |c_{k,i}| \le n/2$.

Now vary the apex a by an infinitesimal offset $\overrightarrow{d(a)}$, and let $d(r) = \left\|\overrightarrow{d(a)}\right\|$ be the (positive) infinitesimal change in distance from a. Let $d(\alpha_i)$ be the change in angle i of T, and let $d(\alpha)_{\max}$ be the largest-magnitude change in any angle of T. Then the change of angle θ_k can also be bounded:

$$\left|d(\theta_k)\right| \le \frac{n\left|d(\alpha)_{\max}\right|}{2} \tag{11}$$

by the coefficient bound above.

Either of the base angles have their variation $\left|d(\alpha_i)\right|$ bounded by $\frac{d(r)}{\ell_{\min}}$, so the change in the apex angle $\left|d(\alpha_3)\right|$ is at most $\frac{2d(r)}{\ell_{\min}}$. Substituting this for $d(\alpha)_{\max}$ gives:

$$\left|d(\theta_k)\right| \le \frac{nd(r)}{\ell_{\min}} \tag{12}$$

We now differentiate Equation 10 and substitute these inequalities, obtaining:

$$\begin{aligned} d(\psi_{i,j}) &\le d(r)\left(\frac{n+2}{\ell_{\min}}\right)\left(\frac{(n+2)^2}{4}\right) \\ &= \frac{d(r)(n+2)^3}{4\ell_{\min}}\end{aligned} \tag{13}$$

Again there is a missing factor of $1/2$ in this argument. The 8 in the denominator in the full theorem comes from rewriting the preceding expressions to omit the apex angle α_3. This can be done without breaking any of the bounds, but it requires lengthier bookkeeping, so we again omit it here.

Conclusion

Our results give the first quantitative empirical evidence that a high proportion of obtuse triangles have periodic billiard paths. Though our methods can't extend to the full conjecture that *all* triangles do, some improvements and extensions still suggest themselves:

- Our experiments have yielded a larger and more diverse set of example billiard paths than was previously available. Examination of these paths reveal common structures that in some cases are amenable to analysis; exploring these might allow better heuristics for finding periodic paths, or (if one is optimistic) even unconditional algorithms for some classes of triangles.
- Observations during testing lead us to believe that the remaining failure cases in our dataset are limited by the grid density we chose: most of them have periodic billiard paths whose radius is too low. A denser grid, or one that grew progressively denser as it approached the base, would probably have a greater success rate for very narrow triangles.
- Our heuristic for finding periodic billiard paths was relatively naive, essentially just testing random vectors from a promising distribution. While this was enough to succeed on most inputs, it seems likely that more principled selection would do even better.

References

[Masur86] Masur, Howard. Closed Trajectories for Quadratic Differentials with an Application to Billiards. *Duke Math. J.* 53 (1986), 307–313.

[Schwartz08] Schwartz, Richard Evan. Obtuse Triangular Billiards II: 100 Degrees Worth of Periodic Trajectories. *Experiment. Math.* 18 (2009), no. 2, 137–171.

Nearest Neighbor Condensation with Guarantees

Alejandro Flores V.* David M. Mount†

Abstract

The problem of nearest-neighbor (NN) condensation deals with reducing the training-set P for the NN-rule classifier. We propose a new NN condensation algorithm called RSS, and prove that selects a subset of size at most $\mathcal{O}(k \log \Delta)$, where k is the number of points in the decision borders of P, and Δ its spread. Similarly, we show that a state-of-the-art algorithm called MSS, selects a subset of size at most $\mathcal{O}(k)$. To the best of our knowledge, these are the first bounds on the sizes of point sets generated by NN condensation algorithms. Additionally, we proof the probability of correctly classifying a query point with ϵ-ANN using RSS, grows exponentially with respect to the size of the RSS set.

1 Introduction

Consider a training-set $P \subset \mathbb{R}^d$ of n points, where each point $p \in P$ belongs to one of a set of discrete classes, denoted $l(p)$. The goal of nonparametric classification techniques, is to accurately predict the class of new points. Among the most well-known techniques is the *nearest-neighbor* (NN) *rule*, which given a query point $q \in \mathbb{R}^d$, assigns it the class of its nearest neighbor in P, denoted $\mathrm{NN}(q)$.

Despite its simplicity, the NN rule exhibits good classification accuracy. Theoretical results show that its probability of error is bounded by twice the Bayes probability of error (the minimum of any decision rule). However, the NN rule is often criticized on the basis of its memory requirements, as P must be stored to answer queries. For this reason, we consider the problem of *Nearest Neighbor Condensation*: selecting a subset of P that maintains its classification performance.

2 Related work

Consider the *Delaunay triangulation* of P. Points with at least one neighbor of a different class are called *border points*, while others are called *internal points*. One approach for NN condensation is to select the set of *border points* of P. This is called *Voronoi condensation* [7]. Unfortunately, a straightforward algorithm is impractical in high-dimensional spaces. For the planar case, an output-sensitive algorithm was proposed [3], which runs in $\mathcal{O}(n \log k)$ time when k is the number of *border points* in P. Yet, it remains an open problem whether this is possible for higher dimensions.

There are other properties that can be used to condense P. First, let's introduce the concept of *enemy*; an enemy of a point $p \in P$ is any point in P of different class, and the *nearest enemy* of p is denoted as $\mathrm{NE}(p)$. Now, let $R \subseteq P$ we say that:

- R is a *consistent* subset of P *iff* for every point $p \in P$, its NN in R is of the same class as p (i.e., p is correctly classified by R).
- R is a *selective* subset of P *iff* for every point $p \in P$, its NN in R is closer to p than is NE in P. Clearly, selectiveness implies consistency.

It has been shown that both problems of finding minimum-size *consistent* and *selective* subsets are NP-complete [8]. Therefore, many heuristics have been proposed to find subsets with such properties (for a comprehensive survey see [6]). Among them, CNN [4] was the first algorithm proposed for computing *consistent* subsets. It has been widely used, despite its worst-case cubic running time, and being order-dependent[1]. Recent efforts resulted in FCNN [1] and MSS [2], which produce *consistent* and *selective* subsets respectively. Both algorithms run in $\mathcal{O}(n^2)$ time, and are order-independent. Unfortunately, to the best of our knowledge, no bounds are known for the size of the subsets generated by any of these algorithms.

Moreover, condensation can introduce problems during classification. In general, these algorithms focus on selecting border points, as they are key in maintaining the classification accuracy on exact NN queries. However, we argue that keeping internal points can be beneficial when performing *approximate* NN queries, increasing the chances of correct classification, and reducing the query time [5].

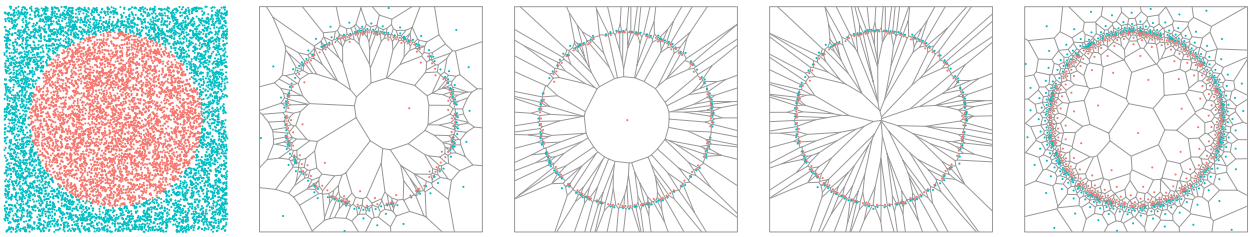
3 Relaxed Selective Subset

We propose an algorithm for NN condensation called RSS, or *Relaxed Selective Subset* (See Algorithm 1). First let's introduce some extra notation. Let $d(p, q)$

*Department of Computer Science, University of Maryland, College Park, MD afloresv@cs.umd.edu

†Department of Computer Science, University of Maryland, College Park, MD mount@cs.umd.edu

[1] Order-dependence means the resulting subset is determined by the order in which points are considered by the algorithm.



(a) Dataset (10^4 pts) (b) CNN (281 pts) (c) FCNN (222 pts) (d) MSS (233 pts) (e) RSS (627 pts)

Figure 1: An illustrative example of the subsets selected by CNN, FCNN, MSS, and RSS.

be the distance between any two points $p, q \in \mathbb{R}^d$, and $d_{\text{NE}}(p)$ be the *nearest enemy* distance of any point $p \in P$, defined as $d_{\text{NE}}(p) = d(p, \text{NE}(p))$.

Algorithm 1: Relaxed Selective Subset

Input: Initial point set P
Output: Condensed point set RSS $\subseteq P$
1 Let $\{p_i\}_{i=1}^{n}$ be the points of P sorted in increasing order of NE distance $d_{\text{NE}}(p_i)$
2 RSS $\leftarrow \phi$
3 **foreach** $p_i \in P$, *where* $i = 1 \dots n$ **do**
4 **if** $\neg\exists\ r \in \text{RSS}$ *such that* $d(p_i, r) < d_{\text{NE}}(r)$ **then**
5 RSS $\leftarrow$ RSS $\cup \{p_i\}$
6 **return** RSS

RSS runs in $\mathcal{O}(n^2)$ time, and it is order-independent as points are sorted before being considered. Moreover:

Theorem 1 RSS *is a selective subset of* P.

This places RSS among state-of-the-art algorithms for NN condensation. However, how do these algorithms compare in terms of the size of their selected subsets? While other algorithms tend to select border points (or points close to the decision borders), the idea behind RSS is to also select internal points following a particular strategy. Figure 1 illustrates the way RSS selects points in comparison with other NN condensation algorithms. In the following theorems, we formalize the bounds on the size of RSS and MSS.

Theorem 2 *Let* k *be the number of border points of* P, *and* Δ *the spread of* P. *Then,* $|\text{RSS}| \leq \mathcal{O}(k \log \Delta)$.

Theorem 3 *Let* k *be the number of border points of* P. *Then,* $|\text{MSS}| \leq \mathcal{O}(k)$.

While RSS can select more points than MSS, we argue that these extra points are beneficial during classification. These internal points will increase the probability of correct classification when using ϵ-approximate NN queries. This intuition is formalized as follows:

Theorem 4 *Let point* $q \in \mathbb{R}^d$ *be drawn uniformly at random from the minimum enclosing ball of* P, *where* P *has* k *border points and spread* Δ. *Then, the probability of equally classifying* q *with* RSS, *using both exact and* ϵ*-approximate* NN *queries (for* $\epsilon \leq 2$*), is bounded by:*

$$Pr\Big[l(\text{NN}_{\text{RSS}}(q)) = l(\text{ANN}_{\text{RSS}}(\epsilon, q))\Big] \geq \frac{k 2^{\Omega(\frac{|\text{RSS}|}{k})}}{\Delta^d}$$

References

[1] F. Angiulli. Fast nearest neighbor condensation for large data sets classification. *Knowledge and Data Engineering, IEEE Transactions on*, 19(11):1450–1464, 2007.

[2] R. Barandela, F. J. Ferri, and J. S. Sánchez. Decision boundary preserving prototype selection for nearest neighbor classification. *International Journal of Pattern Recognition and Artificial Intelligence*, 19(06):787–806, 2005.

[3] D. Bremner, E. Demaine, J. Erickson, J. Iacono, S. Langerman, P. Morin, and G. Toussaint. Output-sensitive algorithms for computing nearest-neighbour decision boundaries. In F. Dehne, J.-R. Sack, and M. Smid, editors, *Algorithms and Data Structures: 8th International Workshop, WADS 2003, Ottawa, Ontario, Canada, July 30 - August 1, 2003. Proceedings*, pages 451–461, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[4] P. Hart. The condensed nearest neighbor rule (corresp.). *IEEE Trans. Inf. Theor.*, 14(3):515–516, Sept. 1968.

[5] D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. Chromatic nearest neighbor searching: A query sensitive approach. *Computational Geometry*, 17(3):97 – 119, 2000.

[6] G. Toussaint. Open problems in geometric methods for instance-based learning. In J. Akiyama and M. Kano, editors, *JCDCG*, volume 2866 of *Lecture Notes in Computer Science*, pages 273–283. Springer, 2002.

[7] G. T. Toussaint, B. K. Bhattacharya, and R. S. Poulsen. The application of voronoi diagrams to non-parametric decision rules. *Proc. 16th Symposium on Computer Science and Statistics: The Interface*, pages 97–108, 1984.

[8] A. V. Zukhba. NP-completeness of the problem of prototype selection in the nearest neighbor method. *Pattern Recognit. Image Anal.*, 20(4):484–494, Dec. 2010.

On the Complexity of Random Semi Voronoi Diagrams

Chenglin Fan[1] and Benjamin Raichel[1]

[1]Department of Computer Science, University of Texas at Dallas
cxf160130@utdallas.edu, benjamin.raichel@utdallas.edu

Abstract

In the standard Voronoi diagram each site is visible to all the points in the plane. Here we consider the Semi Voronoi Diagram [CLX10], where instead each site is visible to some given half plane whose boundary passes through the site. The worst case complexity of such diagrams is $\Theta(n^2)$ [FLWZ14]. Here we show that when the direction of the half plane for each site is random, then with high probability the Semi Voronoi Diagram has near linear complexity, and in the full version of the paper we later show the expected complexity is $\Theta(n)$.

1 Introduction

The Voronoi diagram is a fundamental structure in Computational Geometry, with applications throughout the sciences. Given an input consisting of a set of n points in the plane, called sites, the standard Voronoi diagram partitions the plane into cells, where each cell is the subset of points in the plane sharing the same closest site under standard Euclidean distance. Many variations of this basic structure have been considered, including (i) sites consisting of other geometric objects, (ii) weighted or other distance function, (iii) higher dimensions and surfaces, (iv) sites other than the nearest one, and many and more. The topic of Voronoi diagrams if far to broad to survey here, and so we refer the reader to the book of Aurenhammer *et al.* [AKL13].

Here we consider a variant of the Voronoi diagram where the visibility of each site is constrained. Many previous visibility restrictions have been considered, for example, Aurenhammer and Stöckl [AS91] studied the so called Peeper's Voronoi diagram, where sites are only visible through a given segment lying outside the convex hull of the sites. In this paper we consider the so called Visibility Restricted Voronoi Diagram (VRVD) [FLWZ14, ASXZ14], where each site is only visible about a given angle around that site (i.e. a region bounded by two rays emanating from the site). These diagrams, for example, model scenarios where the site has a restricted field of view, such as may be the case with various optical sensors or human vision. When the visible region for each site is a half plane whose boundary passes through the site (i.e. a visible angle of $\pi/2$), such diagrams were originally studied under the name Semi Voronoi Diagrams [CLX10].

The worst case complexity of the standard Voronoi diagram is linear in the number of sites, and a number of generalizations of the Voronoi diagram retain this desirable property. In particular, Klein [Kle89] introduced the notion of abstract Voronoi diagrams, outlining a simple set of general axioms, capturing a large class of Voronoi diagrams with linear worst case complexity. Unfortunately, there are also many natural generalizations of Voronoi diagrams for which the worst case complexity grows significantly. To combat this many previous works have shown, that by assuming some form of randomness in the input, the expected complexity of some of these diagrams remains linear. While the worst case complexity of higher dimensional Voronoi diagrams is $\Theta(n^{\lceil d/2 \rceil})$, Dwyer [Dwy89] showed if the sites locations are sampled uniformly from a d dimensional ball, the expected complexity is $\Theta(n)$. For geodesic Voronoi diagrams on so called Realistic Terrains, introduced by Aronov *et al.* [AdBT08], the worst case complexity is $\Theta(m+n\sqrt{m})$ (where m is the terrain complexity), however, Driemel *et al.* [DHR16] showed if the site locations are sampled uniformly from the terrain then the expected complexity is $\Theta(n+m)$. For multiplicative weighted Voronoi diagrams, the worst case complexity is $\Theta(n^2)$ [AE84], however, an expected $O(n\log^2(n))$ bound was shown if the weights were randomly permuted amongst the sites [HR15, CHR16]. (A similar result for the union complexity of randomly expanded segments was shown by [AHKS14].)

In this paper we continue this line of research for the Semi Voronoi diagram. Such diagrams have $\Theta(n^2)$ worst case complexity [FLWZ14]. (Actually they show this tight bound for the more general VRVD.) Here we show, however, if the visibility half plane for each site is randomly sampled then the complexity is near linear with high probability, and moreover is linear in expectation. (The linear expectation bound is left to the full version of the paper.) Our proof can easily be extended to show that the more general VRVD has expected linear complexity (assuming the angle for each site is at least a constant), although we omit the details here.

2 Preliminaries

The standard Voronoi diagram: Let $S = \{s_1, s_2, ..., s_n\} \subset \mathbb{R}^2$ be a set of n point *sites* in the plane. Let $\|x - y\|$ denote the Euclidean distance from x to y, and for two sites $p, q \in S$ let $\beta(p, q)$ denote their bisector, that is the set of points x in plane such that $\|p - x\| = \|q - x\|$. Any site $p \in S$ induces a distance function $f_p(x) = \|p - x\|$ defined for any point x in the plane. For any subset $X \subset S$, the *Voronoi cell* of p with respect to X, $\mathcal{V}_{\text{cell}}(p, X) = \{x \in \mathbb{R}^2 \mid \forall q \in X \quad f_p(x) \leq f_q(x)\}$, is the subset of points in the plane sharing p as their closest site in X. We define the *Voronoi diagram* of X, denoted $\mathcal{V}(X)$, as the partition of the plane into Voronoi cells induced by the minimization diagram of the set of distance functions $\{f_p \mid p \in X\}$.

One can view the union, U, of the boundaries of the cells in the Voronoi diagram as a planar graph. Specifically, define a *Voronoi vertex* as any point in U which is equidistant to three sites in S (which happens at the intersection of bisectors). For simplicity, we make the general position assumption that no point is equidistant to four or more sites in the plane. Furthermore, define a *Voronoi edge* as any maximal connected subset of U which does not contain a Voronoi vertex. (Note that in order for each edge to have two endpoints we must include the "point" at infinity, i.e. the graph is defined on the stereographic projection of the plane onto the sphere.) The complexity of the Voronoi diagram is then defined as the total number of Voronoi edges, vertices, and cells. As the cells in the standard Voronoi diagram are simply connected sets, which are the faces of a straight line planar graph where every vertex has degree three, the overall complexity is $\Theta(n)$. (It is also known that diagrams allowing much more general bisectors, called abstract Voronoi diagrams [Kle89], also have linear complexity.)

kth order Voronoi diagram: In our analysis we will make use of the kth order Voronoi diagram. Let S be a set of n point sites in the plane. Then the ***kth order Voronoi diagram*** of S is the partition of the plane into cells such that each cell is the locus of points which have the same set of k nearest sites of S (the ordering of these k sites by distance to the query point can vary within the cell). It is not hard to see that this again defines a straight line partition of the plane into cells where the edges on the boundary of a cell are composed of bisector pieces. It is well known that the worst case complexity of this diagram is $\Theta(k(n-k))$ (see [AKL13, Section 6.5]).

Semi Voronoi Diagram: In this paper we investigate the expected complexity of the Semi Voronoi diagram. As before let $S = \{s_1, s_2, ..., s_n\}$ be a set of n point sites in the plane. Additionally, let $\mathsf{H}(s_i)$ be some closed half plane associated with s_i, whose boundary passes through s_i. We use $L(s_i)$ to denote the bounding line of the half plane $\mathsf{H}(s_i)$. For any point x in the plane and any site $s_i \in S$, we say that x and s_i are visible to each other precisely when $x \in \mathsf{H}(s_i)$. Given a point x in the plane, let $S(x) = \{s_i \in S \mid x \in \mathsf{H}(s_i)\}$ denote the set of sites which are visible to x.

For any subset $X \subset S$, define the *Semi Voronoi cell* of p with respect to X as, $\mathcal{SV}_{\text{cell}}(p, X) = \{x \in \mathbb{R}^2 \mid \forall q \in S(x) \quad \|x - p\| \leq \|x - q\|\}$. Note it is possible there exists points in the plane which are not visible by any site in S, however, this technicality can be avoided by adding a pair of sites far away from S which combined can see the entire plane. As before the Semi Voronoi cells define a straight line partition of the plane, where now the edges on the boundary of a cell are either portions of a bisector or of a half plane boundary. In the worst case, the Semi Voronoi diagram of such a set of sites can have quadratic complexity [FLWZ14].

Probabilistic Model: We consider Semi Voronoi diagrams where the set of sites $S = \{s_1, \ldots, s_n\}$ is allowed to be any fixed set of n points in general position. For each site s_i, the line bounding the half plane of s_i, $L(s_i)$, is allowed to be any fixed line in $\mathbb{R}^2$ passing through s_i. Such a line defines two possible visible closed half spaces. We assume that independently for each site s_i, one of these two spaces is sampled uniformly at random (i.e. each has $1/2$ probability).

Observation 2.1. *An alternative natural input assumption would be to assume that the normal of the half plane for each site is sampled uniformly at random from $[0, 2\pi)$. Note that our model is strictly stronger, that is any bound we prove for our model will also imply the same bound for this alternative formulation. This is because one can think of sampling normals from $[0, 2\pi)$, as instead first sampling directions for the bounding lines from $[0, \pi)$, and then next sampling one of the two sides of each line for the normal.*

3 The Expected Complexity of Random Semi Voronoi Diagrams

3.1 The probability of covering the plane

As it will be used in our later calculations, we first bound the probability that for a given subset X of k sites of S that there exists a point in the plane which is not visibile to any site in X. Namely,

$$\mathbf{Pr}\left[\left(\bigcup_{x_i \in X} \mathcal{SV}_{\text{cell}}(x_i, X)\right) \neq \mathbb{R}^2\right]$$

Lemma 3.1. *For any set $X = \{x_1, \ldots, x_k\}$ of k sites $\mathbf{Pr}\left[\left(\bigcup_{x_i \in X} \mathcal{SV}_{\text{cell}}(x_i, X)\right) \neq \mathbb{R}^2\right] = O(k^2/2^k)$.*

Proof: Consider the arrangement of the k bounding lines $L(x_1), \ldots, L(x_k)$. Let $\mathcal{F}$ denote the set of faces in this arrangement (i.e. the connected components of the complement of the union of lines), and note that $|\mathcal{F}| = O(k^2)$. Observe that for any face $f \in \mathcal{F}$ and any fixed site $x_i \in X$, either every point in f is visible by x_i or no point in f is visible by x_i. Moreover, the probability that face f is not visible by site x_i is $\mathbf{Pr}[\mathsf{H}(x_i) \cap f = \emptyset] \leq 1/2$. Hence the probability that a face f is not visible by any of the k sites is

$$\mathbf{Pr}\left[\bigcup_{x_i \in X} \mathsf{H}(x_i) \cap f = \emptyset\right] \leq 1/2^k.$$

Hence by the union bound the probability that at least one face in $\mathcal{F}$ is is not visible by any sites in X is

$$\mathbf{Pr}\left[\left(\bigcup_{x_i \in X} \mathcal{SV}_{\text{cell}}(x_i, X)\right) \neq \mathbb{R}^2\right]$$
$$\leq \sum_{f \in \mathcal{F}} \mathbf{Pr}\left[\bigcup_{x_i \in X} \mathsf{H}(x_i) \cap f = \emptyset\right] = O\left(\frac{k^2}{2^k}\right) \qquad \blacksquare$$

3.2 A Simple Near Linear Bound

Ultimately we can show that the expected complexity of a random Semi Voronoi diagram is linear, however, here we show that Lemma 3.1 implies a near linear bound which also holds with high probability. Specifically, we say that a quantity is bounded by $O(f(n))$ with high probability, if for any constant α there exists a constant β, depending on α, such that the quantity is at most $\beta \cdot f(n)$ with probability at least $1 - 1/n^\alpha$.

Lemma 3.2. *Let $S = \{s_1, \ldots, s_n\} \subset \mathbb{R}^2$ be a set of n sites, where each site has a corresponding line $L(s_i)$ passing through s_i. For each s_i, sample a half plane $\mathsf{H}(s_i)$ uniformly from the set of two half planes whose boundary is $L(s_i)$ (i.e. each has $1/2$ probability). Then the expected complexity of the Semi Voronoi diagram on S is $O(n \log^3 n)$, and moreover this bound holds with high probability.*

Proof: Let $k = c \log n$, for some constant c. Consider the kth order Voronoi diagram of S. We first triangulate this diagram so that the boundary of each cell has constant complexity (Note triangulating does not asymptotically change the number of cells.) Fix any cell Δ in this triangulation, which in turn fixes some (unordered) set X of k-nearest sites. By Lemma 3.1,

$$\mathbf{Pr}\left[\left(\bigcup_{x_i \in X} \mathcal{SV}_{\text{cell}}(x_i, X)\right) \neq \mathbb{R}^2\right] = O(k^2/2^k)$$
$$= O((c \log n)^2 / 2^{c \log n}) = O(1/n^{c-\varepsilon'}),$$

for any arbitrarily small value $\varepsilon' > 0$. Thus with polynomially high probability for every point in Δ its closest visible site will be in X. Now let T be the set of all $O(k(n-k)) = O(n \log n)$ triangles in the triangulation of the kth order diagram. Observe that the above high probability bound applies to any triangle $\Delta \in T$. Thus taking the union bound we have that with probability at least $1 - 1/n^{c-(1+\varepsilon)}$ (where $\varepsilon > \varepsilon' > 0$ is an arbitrarily small value), simultaneously for every triangle Δ, every point in Δ will be visible by one of its k closest sites. Let e denote this event happening (and $\bar{e}$ denote it not happening). Conditioning on e happening, there are only $O(k) = O(\log n)$ relevant sites which contribute to the Semi Voronoi diagram in any cell Δ. Thus the total complexity of the Semi Voronoi diagram restricted to any cell Δ is at most $O(\log^2 n)$, as the Semi Voronoi diagram has worst case quadratic complexity [FLWZ14]. (Note that as Δ is a triangle, we

can ignore added complexity due to clipping the Semi Voronoi diagram of these sites to Δ.) On the other hand, if $\overline{e}$ happens, then the worst case complexity of the Semi Voronoi diagram is still $O(n^2)$.

Now the complexity of the Semi Voronoi diagram is bounded by the sum over the cells in the triangulation of the complexity of the diagram restricted to each cell. Thus the above already implies that with high probability the complexity of the Semi Voronoi diagram is $O\left(\sum_{\Delta \in T} \log^2 n\right) = O(n \log^3 n)$. As for the expected value, by choosing c sufficiently large we have,

$$\begin{aligned}
&\mathbf{E}[|\mathcal{SV}(S)|] \\
=&\mathbf{E}\left[|\mathcal{SV}(S)| \;\middle|\; e\right] \mathbf{Pr}[e] + \mathbf{E}\left[|\mathcal{SV}(S)| \;\middle|\; \overline{e}\right] \mathbf{Pr}[\overline{e}] \\
=&O\left(\sum_{\Delta \in T} \log^2 n\right) \mathbf{Pr}[e] + O(n^2)\, \mathbf{Pr}[\overline{e}] \\
=&O(n \log^3 n)\, \mathbf{Pr}[e] + O(n^2)\, \mathbf{Pr}[\overline{e}] \\
=&O(n \log^3 n)\, \mathbf{Pr}[e] + O(n^2) \cdot (1/n^{c-(1+\varepsilon)}) \\
=&O(n \log^3 n) + O(1/n^{c-(3+\varepsilon)}) = O(n \log^3 n). \qquad \blacksquare
\end{aligned}$$

The above proof worked by partitioning the plane based on the kth order Voronoi diagram, for $k = c \log n$, and then arguing that for all cells simultaneously one of the k nearest neighbors will be visible. Rather than using a fixed kth order diagram, if one is more careful, and allows k to vary then one can argue that the expected complexity of the Semi Voronoi diagram is in fact linear. The details are omitted here due to space, however, they will be posted to the arXiv at a later date.

Theorem 3.3. *Let $S = \{s_1, \ldots, s_n\} \subset \mathbb{R}^2$ be a set of n sites, where each site has a corresponding line $L(s_i)$ passing through s_i. For each s_i, sample a half plane $\mathsf{H}(s_i)$ uniformly from the set of two half planes whose boundary is $L(s_i)$ (i.e. each has $1/2$ probability). Then the expected complexity of the Semi Voronoi diagram on S is $\Theta(n)$.*

References

[AdBT08] B. Aronov, M. de Berg, and S. Thite. The complexity of bisectors and Voronoi diagrams on realistic terrains. In *16th Annual European Symposium on Algorithm (ESA)*, pages 100–111, 2008.

[AE84] F. Aurenhammer and H. Edelsbrunner. An optimal algorithm for constructing the weighted voronoi diagram in the plane. *Pattern Recognition*, 17(2):251–257, 1984.

[AHKS14] P. Agarwal, S. Har-Peled, H. Kaplan, and M. Sharir. Union of random minkowski sums and network vulnerability analysis. *Discrete & Comput. Geometry*, 52(3):551–582, 2014.

[AKL13] F. Aurenhammer, R. Klein, and D.T. Lee. *Voronoi Diagrams and Delaunay Triangulations*. World Scientific, 2013.

[AS91] F. Aurenhammer and G. Stöckl. On the peeper's voronoi diagram. *SIGACT News*, 22(4):50–59, September 1991.

[ASXZ14] F. Aurenhammer, B. Su, Y. Xu, and B. Zhu. A note on visibility-constrained voronoi diagrams. *Discrete Applied Mathematics*, 174:52–56, 2014.

[CHR16] H. Chang, S. Har-Peled, and B. Raichel. From proximity to utility: A Voronoi partition of pareto optima. *Discrete & Computational Geometry*, 56(3):631–656, 2016.

[CLX10] Y. Cheng, B. Li, and Y. Xu. Semi Voronoi diagrams. In *Computational Geometry, Graphs and Applications (CGGA)*, pages 19–26, 2010.

[DHR16] A. Driemel, S. Har-Peled, and B. Raichel. On the expected complexity of Voronoi diagrams on terrains. *ACM Trans. Algorithms*, 12(3):37:1–37:20, 2016.

[Dwy89] R. Dwyer. Higher-dimensional Voronoi diagrams in linear expected time. In *Proc. 5th Annual Symposium Computational Geometry (SOCG)*, pages 326–333, 1989.

[FLWZ14] C. Fan, J. Luo, W. Wang, and B. Zhu. Voronoi diagram with visual restriction. *Theor. Comput. Sci.*, 532:31–39, 2014.

[HR15] S. Har-Peled and B. Raichel. On the complexity of randomly weighted multiplicative Voronoi diagrams. *Discrete & Computational Geometry*, 53(3):547–568, 2015.

[Kle89] R. Klein. Combinatorial properties of abstract Voronoi diagrams. In *Int. Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, pages 356–369, 1989.

Efficient algorithms for computing a minimal homology basis

Tamal K. Dey[1†], Tianqi Li[1‡], and Yusu Wang[1§]

Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, 43214

1 Introduction

Many applications in science and engineering require computing some "features" such as "holes" and "tunnels" in a shape that is finitely represented by a simplicial complex. A concise definition of these otherwise vague notions can be obtained by considering homology groups and their representative cycles. In particular, a one-dimensional homology basis, that is, a maximally independent set of cycles in the 1-skeleton of the input simplicial complex can be taken as a representative of the "holes" and "tunnels" present in the shape. However, instead of any basis, one would like to have a homology basis whose representative cycles are small under some suitable metric, thus bringing the "geometry" into picture along with topology.

When the input complex is a graph with n vertices and m edges, a number of efficient algorithms have been designed to compute a minimal cycle basis [1, 4, 7, 9, 10]. The best known algorithm for this case runs in $O(m^2n/\log\, n + n^2m)$ [10]. For the special case of a combinatorial 2-manifold with weights on the edges, Erickson and Whittlesey [6] gave an $O(n^2\log n + gn^2 + g^3n)$-time algorithm to compute a minimal homology basis where n is the total number of simplices and g is the rank of the first homology group. Dey et al. [5] and Chen and Friedman [3] generalized the results above to arbitrary simplicial complexes. Busaryev et al. [2] improved the running time of this generalization from $O(n^4)$ [5] to $O(n^\omega + n^2g^{\omega-1})$ where $\omega < 2.3728639$ [**?**] is the matrix multiplication exponent. This gives the best known $O(n^{1+\omega})$ worst-case time algorithm when $g = \Theta(n)$. In Section 3, combining the divide and conquer approach of [9] with annotations [2], we develop an improved algorithm to compute a minimal 1-dimensional homology basis for an arbitrary simplicial complex in only $O(n^2g + n^\omega)$ time. Considering $g = \Theta(n)$, this gives the first $O(n^3)$ worst-case time algorithm for the problem.

We can further improve the time complexity if we allow for approximations. In Section 4, we present two algorithms to compute an approximate minimal homology basis: one is a $(2k-1)$-approximation in time $O(kn^{1+1/k}g \text{ polylog } n + n^\omega)$ where k is a positive integer, and the other is a 2-approximation algorithm running in $O(n^\omega\sqrt{n\log n})$ expected time.

2 Background and notations

In this paper, we are interested in computing a minimal basis for the one dimensional homology group of a simplicial complex over the field $\mathbb{Z}_2$.

Homology We are concerned with the homology bases of H_d and particularly in H_1 (more formally below). We use g to denote the 1-*st Betti number* of $\mathcal{K}$, which is the dimension of vector space H_1.

- A set of cycles $C_1, \cdots, C_L$, with $L = rank(\mathsf{Z}_1)$, that generates the cycle space Z_1 is called its *cycle basis*.
- For any 1-cycle c, let $[c]$ denote its homology class. A set of homology classes $\{[C_1], ..., [C_g]\}$ that constitutes a basis of H_1 is called a *homology basis*. For simplicity, we also say a set of cycles $\{C_1, C_2, \cdots, C_g\}$ is a homology basis if their corresponding homology classes $\{[C_1], [C_2], \cdots, [C_g]\}$ form a basis for $\mathsf{H}_1(\mathcal{K})$.
- Let $\mu : \mathsf{Z}_1 \to \mathbb{R}^+ \cup 0$ be a size function that assigns a non-negative weight to each cycle $C \in \mathsf{Z}_1$. A cycle or homology basis $C_1, \cdots, C_g$ is called *minimal* if $\sum_{i=1}^{g} \mu(C_i)$ is minimal among all basis of Z_1 or $\mathsf{H}_1(\mathcal{K})$ respectively.

Annotation. To compute a minimal homology basis of a simplicial complex $\mathcal{K}$, it is necessary to give a way to represent and distinguish homology classes of cycles. Here we use annotation based on [2]. An annotation for a d-simplex is a g-bit binary vector, where $g = rank(\mathsf{H}_d(\mathcal{K}))$. The annotation of a cycle z, which is the sum of annotations of all simplices in z, provides the coordinate vector of the homology class of z in a pre-determined homology basis.

† tamaldey@cse.ohio-state.edu

‡ li.6108@osu.edu

§ yusu@cse.ohio-state.edu

Definition 2.1. (Annotations). [2] *Let $\mathcal{K}$ be a simplicial complex and $\mathcal{K}_d$ be the set of d-simplices in $\mathcal{K}$. An annotation for d-simplices is a function $a : \mathcal{K}_d \to (\mathbb{Z}_2)^g$ with the following property: any two d-cycles z and z' are homologous if and only if*

$$\sum_{\sigma \in z} a(\sigma) = \sum_{\sigma \in z'} a(\sigma)$$

Given an annotation a, the annotation of any d-cycle z is defined by $a(z) = \sum_{\sigma \in z} a(\sigma)$.

Proposition 2.1 ([2]) *There is an algorithm that annotates the d-simplices in a simplicial complex with n simplices in $O(n^\omega)$ time.*

3 An efficient algorithm for minimal homology basis

In this section, we briefly describe an algorithm to compute a minimal homology basis using edge annotation [2] and divide-and-conquer technique [9]. More details will appear in the full version of the paper

Let $\mathcal{K}$ be a simplicial complex with n number of total simplices.[1] Assume that the edges in $\mathcal{K}$ are weighted with non-negative weights. Given any homology basis $\{C_1, \ldots, C_g\}$ with $g = rank(\mathsf{H}_1(\mathcal{K}))$, the size $\mu(C)$ of a cycle $C \in Z_1(\mathcal{K})$ is defined to be the total weights of its edges. Thus the problem of computing a minimal homology basis of H_1 is to find a basis $\mathcal{C} = \{C_1, C_2, \cdots, C_g\}$ such that the sum of $\sum_{i=1}^{g} \mu(C_i)$ is the smallest.

We describe the algorithm to compute a minimal homology basis. First we need to annotate all edges using the algorithm mentioned in [2]. Then we compute a candidate set $\mathcal{G}$ of cycles that includes a minimal homology basis. We use the shortest path tree approach proposed in [5, 6].

Proposition 3.1 [5,6] The candidate set $\mathcal{G}$ has $O(n^2)$ cycles and admits a minimal homology basis.

We compute a minimal homology basis from the candidate set $\mathcal{G}$. We modify the divide and conquer approach of [9] which computes a minimal cycle basis of a graph with non-negative weights. The algorithm uses an auxiliary set $\mathcal{S}$ of g-bit support vectors [9] to select a minimal homology basis from the candidate set $\mathcal{G}$.

Before describing the algorithm, we define the function $m : \mathcal{S} \times \mathcal{G} \to \{0, 1\}$ with $m(S, C) = \langle S, a(C) \rangle$ where $\langle \cdot, \cdot \rangle$ is the inner product over $\mathbb{Z}_2$. We say a cycle C is *orthogonal* to a support vector S_i if $m(S_i, C) = 0$ and is *non-orthogonal* if $m(S_i, C) = 1$.

Now we describe the algorithm choosing a minimal homology basis from $\mathcal{G}$, namely, selecting C_1, $\cdots$, C_g iteratively from $\mathcal{G}$ and adding to the minimal homology basis. During the procedure, the algorithm always finds cycles C_i where $1 \leq i \leq g$ and maintains a set of support vectors $S_1, S_2, \cdots, S_g$ with the following properties:

(1). $S_1, S_2, \cdots, S_g$ forms a basis of $\{0, 1\}^g$.
(2). When $C_1, C_2, \cdots, C_{i-1}$ have already been computed, $m(S_i, C_j) = 0$, $j < i$.
(3). When $C_1, C_2, \cdots, C_{i-1}$ have already been computed, C_i is chosen so that C_i is the shortest cycle with $m(S_i, C_i) = 1$.

We have the following theorem to support our algorithm (proof will appear in the full version of the paper).

Theorem 3.1. *The set $\{C_1, C_2, \cdots, C_g\}$ computed by maintaining properties (1), (2) and (3) is a minimal homology basis.*

In order to maintain property 2, we need to update the support vectors $S_i, \cdots, S_g$ after computing C_i. We initialize each support vector S_i so that only the ith bit is set to 1. Following from [9], we have two procedures, ExtendBasis(i, k) which extends the current partial basis $\{C_1, \cdots, C_{i-1}\}$ by adding k new cycles, and Update(i, k) to maintain property (2). To compute a minimal homology basis, we run ExtendBasis(1, g).

The procedure ExtendBasis(i, k) is implemented as follows. If $k > 1$, it first recursively calls ExtendBasis(i,$\lfloor k/2 \rfloor$) to add $\lfloor k/2 \rfloor$ cycles to the current partial basis. Then it calls the procedure Update(i, k) which uses the already updated support vectors $S_i, \cdots, S_{i+\lfloor k/2 \rfloor - 1}$ to update S_l so that $m(S_l, C_j) = 0, \forall j < i + \lfloor k/2 \rfloor, i + \lfloor k/2 \rfloor \leq l \leq i + k - 1$. Finally, it calls ExtendBasis($i + \lfloor k/2 \rfloor$, $\lceil k/2 \rceil$) to add the remaining $\lceil k/2 \rceil$ cycles to the basis. If $k = 1$(base case), it calls ShortestCycle(S_i) to choose the shortest cycle satisfying property (3).

We now describe Update(i, k). It updates S_j to $\hat{S}_j = S_j + \sum_{t=0}^{\lfloor k/2 \rfloor - 1} \alpha_{jt} S_{i+t}$ where $i + \lfloor k/2 \rfloor \leq j \leq i + k - 1$ aiming to guarantee property (1) and (2). We just need to determine the coefficients $\alpha_{j1}, ..., \alpha_{j\lfloor k/2 \rfloor}$ so that

[1] We are only interested in 1-dimensional homology basis, so we assume $\mathcal{K}$ contains simplices of dimension at most 2.

$m(\hat{S}_j, C_t) = 0, t = i, ..., i + \lfloor k/2 \rfloor - 1$. Modified from Kavitha *et al.* [9], we can compute those coefficients and update the vectors $\hat{S}_j$ in time $O(nk^{\omega-1})$(The details will appear in the full version of the paper).

We also compute $m(S_j, e)$ to $m(\hat{S}_j, e)$ for $i + \lfloor k/2 \rfloor \leq j \leq i + k - 1$ and every edge e which helps in procedure ShortestCycle$(\cdot)$. From the analysis above, it follows that for every edge e, $m(\hat{S}_j, e) = m(S_j + \sum_{t=0}^{\lfloor k/2 \rfloor - 1} \alpha_{jt} S_{i+t}, e) = m(S_j, e) + \sum_{t=0}^{\lfloor k/2 \rfloor - 1} \alpha_{jt} m(S_{i+t}, e)$, $i + \lfloor k/2 \rfloor \leq j \leq i + k - 1$. Using matrix multiplication, $m(\hat{S}_j, e)$ can be updated in time $O(nk^{\omega-1})$ where $i + \lfloor k/2 \rfloor \leq j \leq i + k - 1$.

Base case for selecting a shortest cycle. We implement the procedure ShortestCycle(S_i) to compute the shortest cycle C_i non-orthogonal to S_i, i.e. the shortest cycle C_i satisfying $m(S_i, C_i) = 1$. Instead of computing the annotation of each cycle directly which increases the time complexity, we use the label technique [10]. Let $\Pi_p(u)$ for any vertex $u \in vert(\mathcal{K})$ denote the unique tree path in T_p from p to u. Then we prove that $m(S_i, C(p, e)) = m(S_i, \Pi_p(u)) + m(S_i, \Pi_p(v))) + m(S_i, e)$. For a fixed $p \in vert(\mathcal{K})$, we can compute the label $m(S_i, \Pi_p(u))$ for every vertex u in $O(n)$ time. It follows that in $O(n^2)$ time one can compute $m(S_i, C)$ for all cycles $C \in \mathcal{G}$ and select the shortest cycle.

We summarize our result in the following theorem.

Theorem 3.2. *For a simplicial complex $\mathcal{K}$, there exists a minimal homology basis that can be computed in time $O(n^2 g + n^\omega)$.*

4 An approximate minimal homology basis of $\mathsf{H}_1(\mathcal{K})$

In this section, we briefly present two algorithms for approximating a minimal 1-dimensional homology basis. The details will appear in the full version of the paper. Here the approximation is defined as follows.

Definition 4.1 (Approximate minimal homology basis). *Suppose $\mathcal{C}^*$ is a minimal homology basis for $\mathsf{H}_1(\mathcal{K})$, and let $\ell_1^* \leq \ell_2^* \leq \cdots \leq \ell_g^*$ denote the sequence of sizes of cycles in $\mathcal{C}^*$ sorted in non-decreasing order. A set of g cycles $\mathcal{C}'$ is a* t-approximate minimal homology basis for $\mathsf{H}_1(\mathcal{K})$ *if (i) $\{[C], C \in \mathcal{C}'\}$ form a basis for $\mathsf{H}_1(\mathcal{K})$; and (ii) let $\ell_1, \ldots, \ell_g$ denote the sequence of sizes of cycles in $\mathcal{C}'$ in non-decreasing order; then for any $i \in [1, g]$, $\ell_i^* \leq \ell_i \leq t \cdot \ell_i^*$.*

$(2k-1)$-Approximation algorithm. The high-level framework of this approximation algorithm is the same as the algorithm described in Section 3. The only change is that in the approximation algorithm, we need to replace the base case ShortestCycle(i) which computes a shortest cycle C_i after a partial basis $\{C_1, \ldots, C_{i-1}\}$ with AproximateCycle(i) where we will compute a cycle A_i which only approximates the size of C_i.

We briefly describe the procedure ApproximateCycle(i) modified from [9, Section 2]. For every i, we construct an auxiliary graph G_i. For each vertex v in $\mathcal{K}$, we make two copies of it in the graph G_i, v^+ and v^-. For every edge $e = (u, v)$ in $\mathcal{K}^{(1)}$, if $m(S_i, e) = 1$ then we add edges (u^+, v^-) and (u^-, v^+) to G_i; otherwise we add edges (u^+, v^+) and (u^-, v^-) to G_i. A shortest cycle C with $m(S_i, C) = 1$ corresponds to a shortest path between two copies of some vertex v in G_i minimizing over all v. Instead computing the shortest path connecting some v^+ and v^-, we now compute the $(2k-1)$-approximate shortest path using the algorithm from [11] for each vertex v and take the minimum as the cycle A_i.

Theorem 4.1. *Using the above algorithm to replace* CycleBasis$(\cdot)$ *in Section 3, we obtain an algorithm that outputs a $(2k-1)$-approximate minimal homology basis in time $O(kn^{1+1/k} g \text{ polylog } n + n^\omega)$, where $k \geq 1$ is an integer.*

2-Approximation algorithm. We now present an algorithm to compute a 2-approximate minimal homology basis. In the high level, we will first compute a set of candidate set $\mathcal{G}'$ of cycles that guarantees to contain a 2-approximate minimal homology basis. We then extract the 2-approximate basis from the candidate set.

To construct the candidate set $\mathcal{G}'$, we first apply the algorithm by Kavitha et al. [8] which can compute a smaller candidate set $\mathcal{G}'$ of $O(n\sqrt{n \log n})$ cycles guaranteed to contain a 2-approximate minimal *cycle basis* (not homology basis) for graph $\mathcal{K}^{(1)}$ (i.e, 1-skeleton of the complex $\mathcal{K}$) in $O(n\sqrt{n} \log^{3/2} n)$ *expected time*. We then prove that such a candidate set $\mathcal{G}'$ also contains a 2-approximate minimal homology basis. Now what remains is to describe how to compute such an approximate basis from the candidate set $\mathcal{G}'$. We use a matrix M to represent the annotations of all cycles in $\mathcal{G}'$ where each column represents a cycle sorted in non-decreasing order. We prove that such a matrix M can be computed in time $O(n^\omega \sqrt{n \log n})$. Then a 2-approximate minimal homology basis can be extracted from M using Gaussian elimination. Therefore the following theorem holds.

Theorem 4.2. *The algorithm above computes a 2-approximate minimal homology basis of the 1-dimensional homology group $\mathsf{H}_1(\mathcal{K})$ of a simplicial complex with non-negative weights in $O(n^\omega \sqrt{n \log n})$ expected time.*

References

1. Glencora Borradaile, Erin Wolf Chambers, Kyle Fox, and Amir Nayyeri. Minimum cycle and homology bases of surface-embedded graphs. *Journal of Computational Geometry*, 8(2):58–79, 2017.
2. Oleksiy Busaryev, Sergio Cabello, Chao Chen, Tamal K Dey, and Yusu Wang. Annotating simplices with a homology basis and its applications. In *Scandinavian Workshop on Algorithm Theory*, pages 189–200. Springer, 2012.
3. Chao Chen and Daniel Freedman. Measuring and computing natural generators for homology groups. *Computational Geometry*, 43(2):169–181, 2010.
4. José Coelho de Pina. *Applications of shortest path methods.* Ph.D. thesis, University of Amsterdam, 1995.
5. Tamal K Dey, Jian Sun, and Yusu Wang. Approximating loops in a shortest homology basis from point data. In *Proceedings of the twenty-sixth annual symposium on Computational geometry*, pages 166–175. ACM, 2010.
6. Jeff Erickson and Kim Whittlesey. Greedy optimal homotopy and homology generators. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1038–1046. Society for Industrial and Applied Mathematics, 2005.
7. Joseph Douglas Horton. A polynomial-time algorithm to find the shortest cycle basis of a graph. *SIAM Journal on Computing*, 16(2):358–366, 1987.
8. Telikepalli Kavitha, Kurt Mehlhorn, and Dimitrios Michail. New approximation algorithms for minimum cycle bases of graphs. *STACS 2007*, pages 512–523, 2007.
9. Telikepalli Kavitha, Kurt Mehlhorn, Dimitrios Michail, and Katarzyna Paluch. A faster algorithm for minimum cycle basis of graphs. In *International Colloquium on Automata, Languages, and Programming*, pages 846–857. Springer, 2004.
10. Kurt Mehlhorn and Dimitrios Michail. Minimum cycle bases: Faster and simpler. *ACM Transactions on Algorithms (TALG)*, 6(1):8, 2009.
11. Liam Roditty, Mikkel Thorup, and Uri Zwick. Deterministic constructions of approximate distance oracles and spanners. In *International Colloquium on Automata, Languages, and Programming*, pages 261–272. Springer, 2005.

Cardiac Trabeculae Segmentation: an Application of Computational Topology

Chao Chen[1], Dimitris Metaxas[2], Yusu Wang[3], Pengxiang Wu[2], and Changhe Yuan[1]

[1]CUNY Queens College and Graduate Center, Flushing, NY, United States,
{chao.chen;changhe.yuan}@qc.cuny.edu
[2]Rutgers University, New Brunswick, NJ, United States, {dnm;pw241}@rutgers.edu
[3]Ohio State University, Columbus, OH, United States, yusu@cse.ohio-state.edu

Abstract. We present a topological approach for cardiac trabeculae segmentation. Trabeculae are fine muscle columns within human ventricles whose both ends are attached to the wall. Extracting these structures are very challenging even with state-of-the-art image segmentation techniques. We observe that these structures form natural topological handles and thus employ a topological approach for it. The contribution includes extracting salient topological handles based on persistent homology, and improving the quality of the extracted handles using optimal persistent cycles. The results have been published in [6, 9].

1 Problem

The interior of a human cardiac ventricle is filled with fine structures including the papillary muscles and the *trabeculae*, i.e., muscle columns of various width whose both ends are attached to the ventricular wall (Figure 1). Accurately capturing these fine structures is very important in understanding the functionality of human heart and in the diagnostic of cardiac diseases. These structures compose 23% of left ventricle (LV) end-diastolic volume in average and thus is critical in accurately estimating any volume-based metrics, e.g., ejection fraction (EF) and myocardial mass; these measures are critical in most cardiac disease diagnostics. A detailed interior surface model will also be the basis of a high quality ventricular flow simulation [8], which reveals deeper insight into the cardiac functionality of patients with diseases like hypokinesis and dyssynchrony.

With modern advanced imaging techniques, e.g., Computed Tomography (CT), we can capture details within cardiac ventricles (Fig. 1(left)). However, most state-of-the-art cardiac analysis methods [11, 10], although very efficient, can not accurately capture these complex structures. The challenge is twofold. First, large variation of geometry and intensity of trabeculae makes it difficult to distinguish them from noise. Second, most segmentation models, e.g., region competition [12] and Markov random field [1], employ global priors, which tend to work against fine structures; the smoothness prior tends to simplify the model and thus remove any fine structures. The shape prior, e.g., the active shape model (ASM) [3], tends to use an average shape and thus remove most fine-scale geometric details.

2 Method

Extraction of trabeculae using persistent homology. We exploit novel global information which is more suitable for the extraction of trabeculae, namely, the *topological prior*. A trabeculae is naturally a *topological handle*; both of its ends are attached to the wall, while the intermediate section is freely mobile. We propose a topological method that explicitly computes topological handles which are salient compared with their surrounding regions. The saliency is measured based on the theory of *persistent*

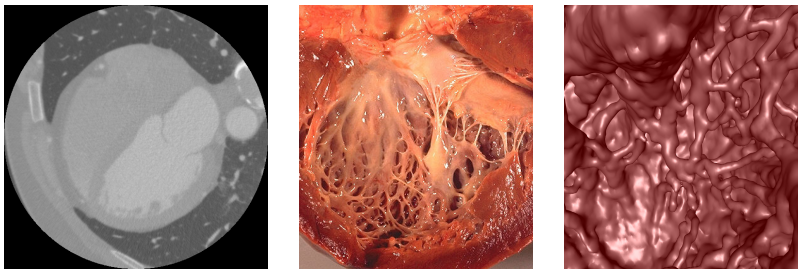


Figure 1: Left: our input CT image. Middle: interior of LV [5]. Right: our result (a 3D triangle mesh) successfully captures the trabeculae (viewed from the valve).

homology [4] and can be computed efficiently. Our algorithm first computes persistence homology using the intensity function of the image. Next, persistent dots on the diagram with high persistence (based on hand-selected threshold) are chosen as hypothetical trabeculae structures. We extract cycles representing these topological structures. Finally, we filter these structures using a classifier trained on the geometric features. The remaining structures are considered the true signal and are included in the final segmentation. The method can be demonstrated in Fig. 2.

Computing the optimal generators for persistence dots. To include the structures in our final segmentation, we need cycles to represent each selected dot in the persistent diagram. However, simply using cycles fails to provide an ideal description of each detected handle. The generated non-optimal descriptions carry noisy geometric information and will hurt the performance of the classifier and segmentation module down the pipeline. To address this issue, we propose a new method that not only detects salient topological handles, but also finds the *ideal description* of each handle. Observe that at the end-diastole state, the heart is maximally relaxed and the trabeculae are maximally stretched out. Therefore, we argue that an ideal description of a topological handle should be geometrically concise; being generally straight rather than wiggling freely. In fact, there are exponentially many cycles that can represent a same persistence dot (Figure 3(middle-left)). We propose to compute the shortest one as it gives us the most concise description. In real data (Fig. 3(right)), we observe that computing optimal loops generates straight trabeculae as desired, while previous topological method using non-optimal loops generates wiggling trabeculae (Fig. 3(middle-right)). Our method is closely related to existing methods for homology localization, but with a more efficient algorithm based on A* search [7].

For a fixed persistence dot of interest, the computation of the optimal cycle can be reduced into a homology localization problem, namely, computing the shortest representative cycle of a given homology class. Existing algorithm [2] solves the problem by finding the shortest path between a source and a target within a *covering graph*. The size of the covering graph is $O(N2^g)$, in which N is the size of the original simplicial complex (in our case, the number of voxels in a given 3D image) and g is the Betti number of dimension one. Running an exhaustive search on such graph is prohibitive as the Betti number g can be up to hundreds in practice. Instead, we propose an A* search algorithm to find the shortest path. The proposed search algorithm relies on a heuristic function which can be computed in linear time $O(gN)$. See Figure 4 for the performance comparison of different optimization methods on synthetic examples with increasing Betti number.

The overall pipeline of our system is shown in figure 5. We validate our method on a real patient dataset consisting of six cardiac CT images at the end-diastolic state (512512320 voxels with spacial resolution from 0.3mm to 0.5mm). For each image, we compute the persistence diagram and then compute the optimal cycle for each salient persistence dot (persistence 80). The running time is about seven minutes and the memory is about six to ten GB. This is about the same expense as the topological method without optimal cycle computation, thanks to the high efficiency of the A* search strategy.

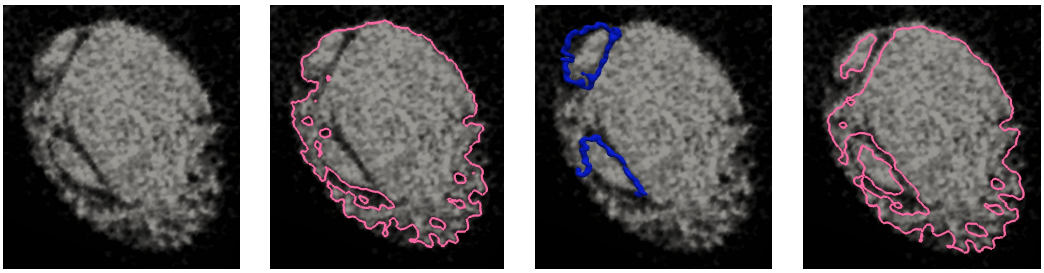


Figure 2: Left: a 2D slice of the intensity function, shaded bridges through the white regions are trabeculae. Middle-left: existing methods will miss the trabeculae completely. Middle-right: our method recovers missed trabeculae using persistent homology. Right: including these trabeculae in the final segmentation gives a better quality segmentation.

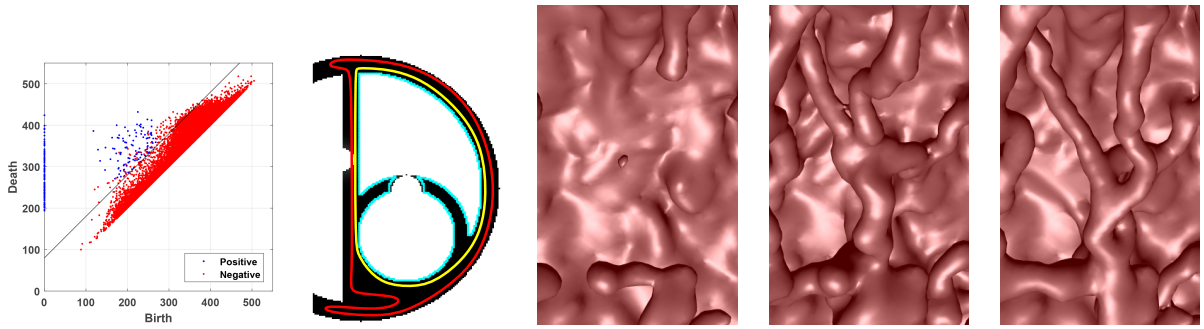


Figure 3: Left: the persistence diagram of a cardiac image function. Middle-Left: a topological handle and its representative cycles. The yellow one is the shortest and best describes its geometry. Middle-right: result with topological prior but without optimal cycles. The reconstructed trabeculae wiggle. Right: our result, using both topological prior and optimal cycles, generates straight trabeculae.

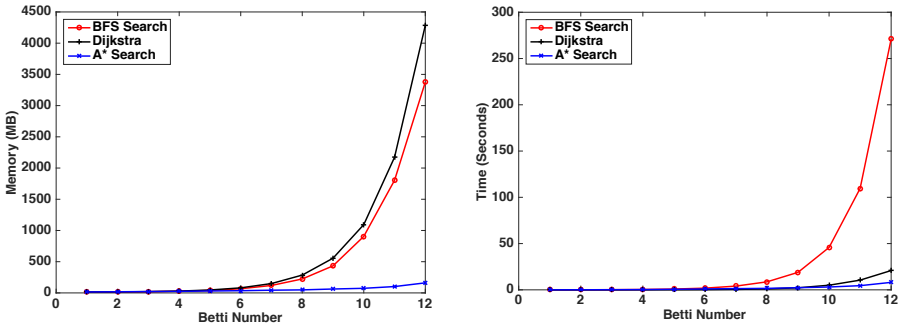


Figure 4: Performance of different search algorithms on the covering graph. Left: memory consumption; Right: running time. We compare breadth-first search (BFS), exhaustive search (Dijkstra's algorithm), and A$*$ search.

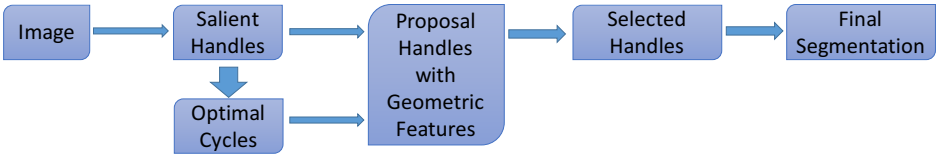


Figure 5: The pipeline of our method.

Optimal cycles for topology data analysis. We believe our formulation and computation of optimal cycles of persistent homology classes are also novel and important to the topology data analysis community. While many algorithms have been proposed to compute persistent homology, the computation of optimal cycles representing them have never been tackled. Our methodology will have broad applications in various persistent-homology-based data analytics. A general purpose software has been made available. [1]

References

[1] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence*, 23(11):1222–1239, 2001.

[2] O. Busaryev, S. Cabello, C. Chen, T. K. Dey, and Y. Wang. Annotating simplices with a homology basis and its applications. In *Scandinavian Workshop on Algorithm Theory*, pages 189–200. Springer Berlin Heidelberg, 2012.

[3] T. F. Cootes, C. J. Taylor, D. H. Cooper, J. Graham, et al. Active shape models-their training and application. *Computer vision and image understanding*, 61(1):38–59, 1995.

[4] H. Edelsbrunner and J. Harer. *Computational topology: an introduction*. Amer Mathematical Society, 2010.

[5] J. Edwin P. Ewing. Gross pathology of idiopathic cardiomyopathy — Wikipedia, the free encyclopedia, 2016. [Online; accessed 09-December-2016].

[1] `https://github.com/astrolagrange/Persistent-Homology-Localization-Algorithms`

[6] M. Gao, C. Chen, S. Zhang, Z. Qian, D. Metaxas, and L. Axel. Segmenting the papillary muscles and the trabeculae from high resolution cardiac ct through restoration of topological handles. In *Information Processing in Medical Imaging (IPMI)*, 2013.

[7] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[8] S. Kulp, M. Gao, S. Zhang, Z. Qian, S. Voros, D. Metaxas, and L. Axel. Using high resolution cardiac CT data to model and visualize patient-specific interactions between trabeculae and blood flow. In *MICCAI*, LNCS, pages 468–475. 2011.

[9] P. Wu, C. Chen, Y. Wang, S. Zhang, C. Yuan, Z. Qian, D. Metaxas, and L. Axel. Optimal topological cycles and their application in cardiac trabeculae restoration. In *Information Processing in Medical Imaging (IPMI)*, 2017.

[10] X. Zhen, H. Zhang, A. Islam, M. Bhaduri, I. Chan, and S. Li. Direct and simultaneous estimation of cardiac four chamber volumes by multioutput sparse regression. *Medical Image Analysis*, 2016.

[11] Y. Zheng, A. Barbu, B. Georgescu, M. Scheuering, and D. Comaniciu. Four-chamber heart modeling and automatic segmentation for 3D cardiac CT volumes using marginal space learning and steerable features. *TMI*, 27(11):1668 –1681, nov. 2008.

[12] S. Zhu, T. Lee, and A. Yuille. Region competition: unifying snakes, region growing, energy/Bayes/MDL for multi-band image segmentation. In *ICCV*, pages 416 –423, June 1995.

Topological and Geometric Reconstruction of Metric Graphs in $\mathbb{R}^n$

Brittany Terese Fasy[1,2], Rafal Komendarczyk[3], Sushovan Majhi[3], and Carola Wenk[4]

[1]School of Computing, Montana State University
[2]Department of Mathematical Sciences, Montana State University
[3]Department of Mathematics, Tulane University
[4]Department of Computer Science, Tulane University

1 Introduction

In the last decade, estimation of topological and geometric features of an unknown underlying space from a finite sample has received an increasing attention in the field of computational topology and geometry. For example in [9] the authors provide a reconstruction guarantee for the topology of an embedded smooth n-manifold from a finite cover by balls of sufficiently small radius around a dense enough finite sample. Random sampling is also considered in [9], and estimates for the probability of reconstructing M from a sample are obtained. These estimates imply that with increasing sample size, the probability of reconstructing M tends to 1, thus we can recover M almost surely as the sample size increases to infinity. Also, curve and surface reconstruction algorithms are discussed in [5]

In practice, not all manifolds are smoothly embedded, nor all spaces of interest are topological manifolds. In [4], the authors show that the homologies of a compact set K of $\mathbb{R}^n$ can be obtained by considering only the relevant homological features in the nerve of the ε radius balls around a sample that is ε close to K in the Hausdorff metric. An upper bound for the required ε estimate is expressed in terms of *weak feature size* of K (defined below). The result of [4] works for non-manifolds, but it is limited to spaces that have a positive weak feature size.

1.1 Background and related work

First, we outline a general approach to estimation of topology and geometry of Euclidean compact sets from the union of balls centered around a finite set of points densely sampled from the underlying space K. Let S be a sample that satisfies some density constraints, and let ε be a radius that depends on K. Then our goal lies to estimate the topology and geometry of K from S^ε. Here, S^ε is the union of Euclidean balls of radius ε around S. Roughly speaking, one can expect to capture the topological features of K if ε is chosen proportional to the size of the features of K and proportional to $d_H(S, K)$, the Hausdorff distance between S an K. If ε is too small or too large, then S^ε may fail to capture the topological features of K.

This suggests the following generic scheme:

1. The underlying space K should be a well-behaved space that would allow us to choose an appropriate "feature size" τ, that would restrict the radius ε not to be too big to capture even the smallest topological feature of K.
2. Having τ, for any $\varepsilon < \tau$ one chooses a sample S that approximates our underlying space K very closely, i.e., $d_H(S, K) < \varepsilon$.
3. One then expects to estimate the topology and geometry of K from S^ε.

In [9] the authors show that for a smooth manifold M embedded in $\mathbb{R}^n$, τ_n can be chosen to be the maximal radius of the embedded normal disk bundle of M. In Theorem 1.1, the authors show that $\sqrt{\frac{3}{5}}\tau_n$ can be chosen to be a threshold for ε to compute the homologies of M from the union of balls around an $\frac{\varepsilon}{2}$-dense sample S.

Theorem 1.1 (Deformation Retraction [9])**.** *Let M be a manifold with injectivity radius τ_n. Let $\overline{x}$ be any finite collection of points $x_1, ..., x_n \in \mathbb{R}^N$ such that it is $\frac{\varepsilon}{2}$ dense in M, where $\varepsilon < \sqrt{\frac{3}{5}}\tau_n$. Then, the union $U = \cup_i B(x_i, \varepsilon)$ deformation retracts onto M. Consequently, the homology of U equals homology of M.*

We also mention the reconstruction results of [4] for compact sets K in $\mathbb{R}^n$. If K admits a positive weak feature size (wfs) and $\varepsilon < \frac{1}{4}\texttt{wfs}$, then a densely sampled set of points can give us the right topology of K.

Theorem 1.2. *Let K and S be two compact sets of $\mathbb{R}^n$ such that $0 < \varepsilon < \frac{1}{4}\mathtt{wfs}(K)$ and $d_H(K,S) < \varepsilon$. Then,*

$$H_k(K) \backsimeq Image(i),$$

where H_k denotes the k-th homology group and i is the inclusion of S^ε in $S^{3\varepsilon}$.

1.2 Summary of results

The current paper is motivated by the following questions:

1. Theorem 1.1 provides a reconstruction result when $\varepsilon < \sqrt{\frac{3}{5}}\tau_n$. Does the result fail to hold when $\sqrt{\frac{3}{5}}\tau_n < \varepsilon < \tau_n$?
2. The result of [4] works for a compact space K having a positive `wfs`. One can easily find very simple 1-dimensional complexes, e.g. trees, that have zero `wfs`. These spaces are often of interest in applications, for instance in road network reconstruction problems [2].
3. The feature size τ is defined for smooth manifolds. How can we define an appropriate feature size when the space is not a smooth manifold, e.g., an embedded simplicial complex.

We answer the first question positively in Theorem 2.1, addressing the case of smooth curves in $\mathbb{R}^2$. Theorem 2.1 shows that $\varepsilon < \tau_n$ is sufficient for the reconstruction. The second and third questions are considered in the setting where K is a metric graph (later denoted by G) embedded in $\mathbb{R}^n$. In this setting, unlike in the manifold case, it is generally not possible to choose a threshold τ for the sampling parameter ε so that S^ε has the same homotopy type as K, even if S is an arbitrary dense sample, since S^ε will contain unnecessary "small" features (noise) that are not present in K. In order to address this issue, we propose a different notion of a feature size that we call *geodesic feature size* (denoted by τ_G). This new definition of feature size allows us to threshold the sampling parameter ε, in the case of a metric graph G, and leads to the reconstruction algorithm shown in Algorithm 1. In particular, we obtain a simplicial complex K^ε in $\mathbb{R}^2$, which is ε-close to G (in the sense of the Hausdorff distance) and which deformation retracts onto G.

2 Reconstruction Results

2.1 Smooth Curve Reconstruction

Theorem 2.1 (Smooth Curve Reconstruction)**.** *Let M be a smooth curve in $\mathbb{R}^2$ without boundary and let τ be the injectivity radius. Let $\varepsilon \in (0, \tau]$ and let S be a finite subset of M such that $M \subseteq S^\varepsilon$. Then, the medial axis of S^ε is homeomorphic to M.*

From the result of [8], which shows that any bounded open subset of Euclidean space is homotopy equivalent to its medial axis, we conclude that S^ε and M are homotopy equivalent.

Remarks 2.2. A deformation retraction constructed in [9], collapses S^ε along the normal lines M. The collapse is not well defined if the intersection of S^ε with a normal line has more than one connected component. The condition $\varepsilon^2 < \frac{3}{5}\tau_n^2$ (for a sample that is $\frac{\varepsilon}{2}$-dense in M) guarantees that such intersections do not happen and the deformation retraction is well-defined.

Sketch of proof. Let S satisfy the assumptions of Theorem 2.1. For brevity of exposition, we assume that M has one path-connected component. Then, we know that M is, in fact, the image of an injective, smooth map $\gamma\colon [0,1] \to \mathbb{R}^2$ with $\gamma(0) = \gamma(1)$. Let us denote the ε-tubular neighborhood of M by M^ε.

Without loss of generality, assume that the sample points of $S = \{x_1, x_2, ..., x_k\}$ are enumerated with increasing preimages: $\gamma^{-1}(x_i) < \gamma^{-1}(x_{i+1})$ for all $i \in \{1, 2, \dots k-1\}$. Then, these samples introduce a partition $\{M_i\}_{i=1}^k$ of the manifold, where $M_i = \gamma([t_i, t_{i+1}])$ and $M_k = \gamma([t_k, 1]) \cup \gamma([0, t_1])$. Let $\widehat{M}$ be the piecewise linear curve obtained by connecting x_i's in the respective order and let $\widehat{M}_i = \overline{x_i x_{i+1}}$.

Since $\varepsilon < \tau$, each ball $B_\varepsilon(x_i)$ intersects the tubular neighborhood M^ε at exactly at two points, say at u_i and l_i; see Figure 1. Let N_i denote the normal $\overline{u_i l_i}$ passing through x_i. Notice that these line segments $\overline{u_i l_i}$ do not intersect. In fact, these normal lines partition the tubular neighborhood into k regions, where the i-th region, denoted M_i^ε, is the one containing M_i.

We will show that $\widehat{M}$ is homeomorphic to M. Observe that $\widehat{M}$ is also the medial axis of S^ε. Restricting our attention to M_i^ε, we define a homeomorphism between $\widehat{M}_i$ and M_i, and extend it globally so that they retain continuity since they agree on each N_i by the pasting lemma.

We define a homeomorphism $\phi_i : \widehat{M}_i \to M_i$ for each M_i^ε in the following way. If we draw a perpendicular L at any point z on $\widehat{M}_i$, we show that L intersects M_i at exactly one point y and define $\phi_i(x) := y$.

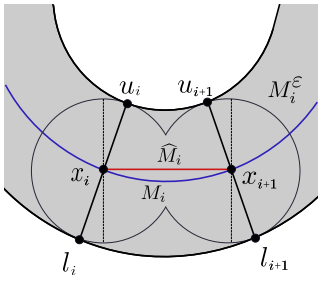


Figure 1: The medial axis and region M_i^ε defined by consecutive sample points x_i and x_{i+1}. The blue smooth curve is a portion of our manifold M. The gray region is the tubular neighborhood M^ε.

As a consequence, M_i is a continuous graph on $\widehat{M}_i$, hence a homeomorphism.

On the contrary, let's assume that there exists a point x on M_i whose normal L intersects M_i at at least two points z_1 and z_2. We arrive at a contradiction by showing that there is a point z on M_i such that the normal T_z at z is parallel to $\widehat{M}_i$.

Without loss of generality, we assume that L cuts the manifold at both z_1 and z_2. Note that z_1 and z_2 are points on the manifold and tangents T_{z_1} and T_{z_2} are not parallel to L. By continuity of the tangents of M, we conclude that there exists a point z on M_i such that T_z is parallel to L. Consequently, the normal N_z at z is parallel to $\widehat{M}_i$.

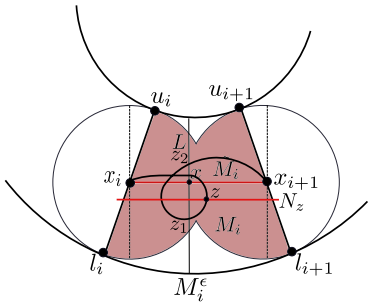


Figure 2: Contradiction in the proof of Theorem 2.1.

Now, we arrive at a contradiction in either of the following cases; see Figure 2.1 for an illustration.

Case 1: If $||x_{i+1} - x_i|| \leq \varepsilon$, then the ε-radius normal, $N_z \cap M^\varepsilon$, at z intersects either N_i or N_{i+1}. This contradicts the fact that τ is the injectivity radius.

Case 2: If $||x_{i+1} - x_i|| > \varepsilon$, then the ε-radius normal, $N_z \cap M^\varepsilon$, at z lies completely in the interior of M_i^ε, which is a contradiction because the boundary of each ε-radius normal lies on the boundary of the tubular neighborhood M^ε of the manifold.

Therefore, the function ϕ_i is a well-defined, invertible, continuous map on a compact domain, hence a homeomorphism.

Since the ϕ_i's agree on the boundary of each $\widehat{M}_i$, we glue them to obtain a global homeomorphism $\phi : \widehat{M} \to M$. This completes the proof. □

2.2 Metric Graph Reconstruction

A weighted graph $G = (V, E)$ is said to be a *metric graph* if the edge-weights are all positive. Then, we can interpret the weights as lengths, and thus each point $e \in E$ has a well-defined distance to the endpoints of E. We define the length of a given continuous path in G to be the total length of all edges and partial edges in the path. Then, the distance function $d_G \colon G \times G \to \mathbb{R}$ is defined to be the length of the shortest path connecting two points; in words, d_G is the geodesic distance in G. Metric graphs were first introduced in [7], and have recently been studied in [1, 6].

Below we list our assumptions about the underlying graph G that we aim to reconstruct.

Assumption 2.3. G is an embedded metric graph with straight line edges. $V = \{v_1, v_2, ..., v_n\}$ is the vertex set and $E = \{e_1, e_2, ..., e_m\}$ is the edge set.

Assumption 2.4. The length of the smallest edge of G is l.

Definition 2.5 (Nerve of a Cover)**.** Suppose $\mathcal{U} = \{U_\alpha\}_{\alpha \in \Lambda}$ is a cover of a topological space X. We take Λ to be the vertex set and form an abstract simplicial complex $\mathcal{K}$ in the following way: if a k-way intersection $U_{\alpha_1} \cap U_{\alpha_2} \cap ... \cap U_{\alpha_k}$ is non-empty, then $\{\alpha_1, \alpha_2, ..., \alpha_k\} \in \mathcal{K}$.

$\mathcal{K}$ is then called the *nerve* of the cover $\mathcal{U}$ and is denoted by $\mathcal{N}(\mathcal{U})$.

Lemma 2.6 (Nerve Lemma [3])**.** *Suppose $\mathcal{U} = \{U_\alpha\}_{\alpha \in \Lambda}$ is a "good" covering of X, i.e., every $U_\alpha \in \mathcal{U}$ is contractible along with all non-empty finite intersections of elements of $\mathcal{U}$. For such a good covering X has the same homotopy type as $\mathcal{N}(\mathcal{U})$.*

We now propose our feature size that we call Geodesic Feature Size (`gfs`).

Definition 2.7 (Geodesic Feature Size)**.** Let G be an embedded metric graph. We define the Geodesic Feature Size (`gfs`) $\boldsymbol{\tau_G}$ of G to be the supremum of all $r > 0$ having the following property: for any x, $y \in G$, if $||x - y|| < 2r$ then $d_G(x, y) < l$, where l is the length of the smallest edge of G.

To motivate the above definition of `gfs`, we take a finite sample $S = \{x_1, x_2, \ldots, x_k\}$ from G. Let $\{\mathbb{B}_\varepsilon(x)\}_{x \in S}$ be a cover of G and let $\mathcal{K}_1 = \mathcal{N}(S, \varepsilon)$ be

its nerve, where $\mathbb{B}_\varepsilon(x)$ is the Euclidean ε-ball centered at x. An edge e, between two vertices x_i and x_j in $\mathcal{K}_1$, is called *transverse* if x_i and x_j belong to two different edges of G. If $\varepsilon < \tau_G$ and e is transverse, then the geodesic distance $d_G(x_i, x_j) < l$ and the geodesic on G is unique. This implies that there is at most one vertex v of G lying on this geodesic. We call this geodesic the geodesic shadow of the edge e. The threshold τ_G for ε forces any transverse edge e to be within $\mathbb{B}_{\xi\epsilon}(v)$ around that vertex v, where $\xi = \max \frac{1}{\sin(\alpha/2)}$, the maximum is taken over all acute angles α between any pair of edges of G. The idea behind this definition of `gfs` comes from our goal to estimate the diameter of non-trivial 1-cycles of $\mathcal{K}_1$ that are not present in G. These *noisy* one-cycles in $\mathcal{K}_1$ are formed by some of the transverse edges. We also mention here without a proof that `gfs` is positive for the type of metric graphs we are considering. In fact, we can show that $0 < \tau_G \leq l/2$, where l is the length of the shortest edge in G.

We now state our main reconstruction theorem for embedded metric graphs. This theorem proves the correctness of Algorithm 1 for computing the 1-dimensional Betti number of G.

Theorem 2.8. *Let $\varepsilon < \frac{1}{\xi}\mathit{gfs}(G)$ and S be a finite sample from G such that $S^\varepsilon \supseteq G$. Then, $H_1(G) = i_*(H_1(S^\varepsilon))$, where i is the inclusion map from $S^\varepsilon \to S^{\xi\varepsilon}$, $H_1(\cdot)$ denotes the first homology group in $\mathbb{Z}$ coefficients and ξ is as defined above.*

Sketch of proof. Let $\mathcal{K}_1 = \mathcal{N}(S, \varepsilon)$ and $\mathcal{K}_2 = \mathcal{N}(S, \xi\varepsilon)$. As $\xi > 1$, it follows that $\varepsilon < \mathtt{gfs}(G)$. An application of the nerve lemma implies that there is an injective homomorphism ϕ from $H_1(G)$ to $H_1(\mathcal{K}_1)$. In other words, $\mathcal{K}_1$ contains all the non-trivial 1-cycles of G. Similarly, we can show that there also exists an injective homomorphism from $H_1(G)$ to $H_1(\mathcal{K}_2)$. We then consider the induced homomorphism $i_* : H_1(K_1) \to H_1(K_2)$, where $i : \mathcal{K}_1 \to \mathcal{K}_2$. Finally, we show that $H_1(\mathcal{K}_1) = \phi(H_1(G)) \oplus Ker\ i_*$. Therefore, $H_1(G) \simeq i_*(H_1(S^\varepsilon))$. □

We believe that it should be possible to find a simplicial complex with the same homotopy type as G. We formulate this stronger result as follows:

Conjecture 2.9. *Let $\varepsilon < \frac{1}{2(2+\xi)}\mathit{gfs}(G)$ and S be a finite sample from G such that each edge of G can be covered by the union of ε-balls centered at the sample points on the same edge. Then the Vietoris-Rips complex $VR(S, d_\varepsilon, 2(1+\xi)\varepsilon)$, computed on S w.r.t. the geodesic metric on the 1-skeleton of $\mathcal{K}_1$ at a scale of $2(1+\xi)$, has the same homotopy type as G.*

Data: The finite sample S from the unknown metric graph G and $\varepsilon > 0$
Result: one dimensional Betti number of G

```
Compute K_1 Compute K_2
for non-trivial 1-cycle η in K_1 do
    if η is trivial in K_2 then
        Collapse η in K_1
    end
end
Compute remaining non-trivial cycles in K_1;
```

Algorithm 1: Metric graph reconstruction from a finite sample.

Remarks 2.10. The idea of collapsing the "small" 1-cycles in Algorithm 1 motivates us to add a full simplex around each vertex whenever a subset of S has a diameter smaller than the estimated scale. That is precisely what the Vietoris-Rips complex does on a finite metric space.

2.3 Discussion

To further extend our result, we also consider a probabilistic reconstruction, as considered by the authors of [9]. Given a $(1-\delta)$ chance of correct reconstruction, one can find the smallest sample size to guarantee the given chance of recovery. Also, we can extend our definition of `gfs` to metric graphs and obtain similar reconstruction results. Lastly, we also consider the reconstruction question when samples that are drawn not exactly from our underlying space, but from a close vicinity of it.

3 Acknowledgments

The first, third, and fourth authors would like to acknowledge the generous support of the National Science Foundation under grants CCF-1618469 and CCF-1618605.

References

[1] Aanjaneya, M., Chazal, F., Chen, D., Glisse, M., Guibas, L., and Morozov, D. Metric graph reconstruction from noisy data. *International Journal of Computational Geometry & Applications 22*, 04 (2012), 305–325.

[2] Ahmed, M., Karagiorgou, S., Pfoser, D., and Wenk, C. *Map Construction Algorithms*. Springer, 2015.

[3] Alexandroff, P. Über den allgemeinen dimensionsbegriff und seine beziehungen zur elementaren geometrischen anschauung. *Mathematische Annalen 98*, 1 (March 1928), 617–635.

[4] Chazal, F., and Lieutier, A. Stability and computation of topological invariants of solids in $\mathbb{R}^n$. *Discrete & Computational Geometry 37*, 4 (2007), 601–617.

[5] Dey, T. K. *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis (Cambridge Monographs on Applied and Computational Mathematics)*. Cambridge University Press, New York, NY, USA, 2006.

[6] Gasparovic, E., Gommel, M., Purvine, E., Sazdanovic, R., Wang, B., Wang, Y., and Ziegelmeier, L. A complete characterization of the 1-dimensional intrinsic cech persistence diagrams for metric graphs. *Research in Computational Topology*. To appear; arXiv preprint arXiv:1702.07379.

[7] Kuchment, P. Quantum graphs: I. some basic structures. *Waves in Random Media 14*, 1 (2004), S107–128.

[8] Lieutier, A. Any open bounded subset of $\mathbb{R}^n$ has the same homotopy type as its medial axis. *Computer-Aided Design 36*, 11 (2004), 1029–1046.

[9] Niyogi, P., Smale, S., and Weinberger, S. Finding the homology of submanifolds with high confidence from random samples. *Discrete And Computational Geometry 39. 1-3* (2008), 419–441.

Randomized Incremental Construction of Net-Trees

Mahmoodreza Jahanseir* Donald R. Sheehy†

1 Introduction

Har-Peled & Mendel introduced the net-tree as a linear-size data structure that efficiently solves a variety of (geo)metric problems such as approximate nearest neighbor search, well-separated pair decomposition, spanner construction, and others [6]. Net-trees are similar to several other data structures that store points in hierarchies of metric nets (subsets satisfying some packing and covering constraints) arranged into a tree or DAG. Examples include navigating nets [7], cover trees [1], dynamic hierarchical spanners [3, 5], and deformable spanners [4]. In Euclidean spaces, the construction times and size depend on the dimension. Similar bounds hold for doubling metrics. There are two known algorithms for building a net-tree [6] or a closely related structure [3] in $O(n \log n)$ time for doubling metrics. Both are quite complex and are primarily of theoretical interest. A much simpler algorithm due to Clarkson [2] can be combined with an algorithm of Har-Peled and Mendel [6] to run in $O(n \log \Delta)$ time, where Δ is the the *spread* of the input, i.e. the ratio of the largest to smallest pairwise distances. Most of the complications of the theoretical algorithm are to eliminate this dependence on the spread.

The goal of this paper is to combine the conceptual simplicity of Clarkson's idea with a simple randomized incremental algorithm to achieve the same $O(n \log n)$ running time of the best theoretical algorithms. The main improvement over the related data structures [3, 6] that can be computed in $O(n \log n)$ time is the increased simplicity.

2 Preliminaries

Doubling Metrics and Packing The input is a set of n points P in a metric space. The *closed metric ball* centered at p with radius r is denoted $\mathbf{B}(p, r) := \{q \in P \mid \mathbf{d}(p, q) \leq r\}$. The *doubling constant* ρ of P is the minimum $\rho \in \mathbb{R}$ such that every ball $\mathbf{B}(p, r)$ can be covered by ρ balls of radius $r/2$. We assume ρ is constant. The *doubling dimension* is defined as $\lg \rho$. A metric space with a constant doubling dimension is called a *doubling metric*. Throughout, we assume that the input metric is doubling.

The following easy to prove lemma is at the heart of all the packing arguments in this paper.

Lemma 1 (Packing Lemma)**.** *If $X \subseteq \mathbf{B}(p, r)$ and for every two distinct points $x, y \in X$, $\mathbf{d}(x, y) > r'$, where $r > r'$, then $|X| \leq \rho^{\lfloor \lg r/r' \rfloor + 1}$.*

The distance from a point p to a compact set Q is defined as $\mathbf{d}(p, Q) := \min_{q \in Q} \mathbf{d}(p, q)$. The *Hausdorff distance* between two sets P and Q is $\mathbf{d}_H(P, Q) = \max\{\max_{p \in P} \mathbf{d}(p, Q), \max_{q \in Q} \mathbf{d}(q, P)\}$.

Net-trees A net-tree is a tree in which each level represents the metric space at some scale. In net-trees, points are leaves in level $-\infty$ and each point can be associated with many internal nodes. Each node is uniquely identified by its associated point and an integer called its *level*. The node in level ℓ associated with a point p is denoted p^ℓ. We assume that the root is in level $+\infty$. For a node $p^\ell \in T$, we define $\text{par}(p^\ell)$ and $\text{ch}(p^\ell)$ to be the parent and the set of children of that node, respectively. Let P_{p^ℓ} denote leaves of the subtree rooted at p^ℓ. For each node p^ℓ in a net-tree, the following properties hold: (Packing) $\mathbf{B}(p, c_p \tau^\ell) \bigcap P \subseteq P_{p^\ell}$, (Covering) $P_{p^\ell} \subset \mathbf{B}(p, c_c \tau^\ell)$, (Nesting) if $\ell > -\infty$, then p^ℓ has a child with the same associated point p.

The constant $\tau > 1$, called the *scale factor*, determines the change in scale between levels. We call c_p and c_c the *packing constant* and the *covering constant*, respectively, and $c_c \geq c_p > 0$. We represent all net-trees with the same scale factor, packing constant, and covering constant with $\text{NT}(\tau, c_p, c_c)$.

There are two different representations for net-trees. In the *uncompressed* representation, every root to leaf path has a node in every level. The size complexity of this representation is $O(n \log \Delta)$, because there are $O(\log \Delta)$ explicit levels between $-\infty$ and $+\infty$. The *compressed* representation is obtained from the uncompressed one by removing the nodes that are the only child of their parents and they have only one child and merging the two adjacent edges as a long edge. We call such long edges *jumps*. One can show that this representation has size of $O(n)$.

A net-tree can be augmented to maintain a list of nearby nodes called relatives. We define relatives of a node p^ℓ to be $\text{Rel}(p^\ell) = \{x^f \in T \text{ with } y^g = \text{par}(x^f) \mid f \leq \ell < g, \text{ and } \mathbf{d}(p, x) \leq c_r \tau^\ell\}$. We call c_r the *relative constant*, and it is a function of the other parameters of a net-tree. In this paper, we assume that net-trees are always equipped with relatives. Note that we defined ch(), par(), and Rel() for a node of a tree; however, we will abuse notation by applying

*University of Connecticut reza@engr.uconn.edu
†University of Connecticut don.r.sheehy@gmail.com

these to sets of nodes. In such cases, the result will be the union of output for each node.

As shown in the following lemma, a compressed net-tree on a doubling metric has $\rho^{O(1)}n$ size.

Lemma 2. *For each node p^ℓ in $T \in \mathrm{NT}(\tau, c_p, c_c)$, $|\mathrm{ch}(p^\ell)| \leq \rho^{\lfloor \lg c_c\tau/c_p \rfloor+1}$ and $|\mathrm{Rel}(p^\ell)| \leq \rho^{\lfloor \lg c_r/c_p \rfloor+1}$.*

3 Net-tree Variants

In this section, we introduce two natural modifications that simplify both construction and analysis.

Local Net-trees Here, we define a local version of net-trees and show that for some appropriate parameters, a local net-tree is a net-tree. The "nets" in a net-tree are the subsets $N_\ell = \{p \in P \mid p^m \in T \text{ for some } m \geq \ell\}$. A *local net-tree* $T \in \mathrm{LNT}(\tau, c_p, c_c)$ satisfies the nesting property and the following invariants: (Local Packing) for distinct $p, q \in N_\ell$, $\mathbf{d}(p, q) > c_p\tau^\ell$, (Local Covering) if $p^\ell = \mathrm{par}(q^m)$, then $\mathbf{d}(p, q) \leq c_c\tau^\ell$, (Local Parent) if $p^\ell = \mathrm{par}(q^m)$, then $\mathbf{d}(p, q) = \mathbf{d}(p, N_{m+1})$.

The difference between the local net-tree invariants and the net-tree invariants given previously, is that there is no requirement that the packing or covering respect the tree structure. We are interested in local packing and covering properties because they are easier invariants to maintain after each tree update.

The switch to local net-trees comes at the cost of having slightly different constants. Theorem 3 gives the precise relationship and we omit its proof due to space constraints.

Theorem 3. *For $\tau > \frac{2c_c}{c_p} + 1$ and $0 < c_p \leq c_c < \frac{c_p(\tau-1)}{2}$, if $T \in \mathrm{LNT}(\tau, c_p, c_c)$, then $T \in \mathrm{NT}(\tau, \frac{c_p(\tau-1)-2c_c}{2(\tau-1)}, \frac{c_c\tau}{\tau-1})$.*

In the rest of this paper, we focus on local net-trees.

Semi-Compressed Net-trees This intermediate structure between uncompressed and compressed net-trees has linear size, and produces a neighborhood graph that is easier to work with because edges are undirected and stay on the same level of the tree. In semi-compressed net-trees, we do not remove a node if it has any relatives other than itself. The following theorem shows that the semi-compressed representation has linear size.

Theorem 4. *Given n points P in a doubling metric with doubling constant ρ. The size of a semi-compressed net-tree on P is $O(\rho^{O(1)}n)$.*

In semi-compressed net-trees, the relative relation is symmetric. From now on, we use $\sim$ to denote the symmetric relative relation between pairs of nodes in semi-compressed net-trees.

4 Approximate Voronoi Diagrams from Net-Trees

Given a set of points P and a query q, the *nearest neighbor* of q in P is the point $p \in P$ such that for all $p' \in P$, we have $\mathbf{d}(q, p) \leq \mathbf{d}(q, p')$. Relaxing this notion, p is a *c-approximate nearest neighbor* (or c-ANN) of q if for all $p' \in P$, we have $\mathbf{d}(q, p) \leq c\mathbf{d}(q, p')$.

The Voronoi diagram of a set of points P is a decomposition of space into cells, one per point $p \in P$ containing the so-called reverse-nearest neighbors of P, those points for which p is the nearest neighbor. The nearest neighbor search problem can be viewed as point location in a Voronoi diagram. Voronoi diagrams have a natural graph structure induced by the incidence between neighboring cells. To limit the complexity of this graph, c-approximate Voronoi diagrams have been studied for which a point x belongs to the cell of p implies that p is a c-ANN of x.

It is straightforward to extract an approximate Voronoi diagram from a net-tree. We start with a slightly more useful decomposition that associates points in the metric space M with nodes in the tree T. Define $f : M \times T \to \mathbb{R} \times \mathbb{Z}$ as

$$f(x, p^\ell) := \begin{cases} (\mathbf{d}(x, p), \ell) & \text{if } \mathbf{d}(x, p) \leq c_r\tau^\ell \\ (\infty, \infty) & \text{otherwise} \end{cases}$$

The Voronoi cell of a node p^ℓ is defined as $\mathrm{Vor}(p^\ell) := \{x \in M \mid f(x, p^\ell) \leq f(x, q^m) \text{ for all } q^m \in T\}$. The ordering on pairs is lexicographical. For a point $q \notin P$, the **center** for q in T, denoted $\mathcal{C}(q)$, is the node $p^\ell \in T$ such that $q \in \mathrm{Vor}(p^\ell)$. As we will show later, finding the center of a point is the basic point location operation required to insert it into the net-tree. Figure 1 illustrates the construction.

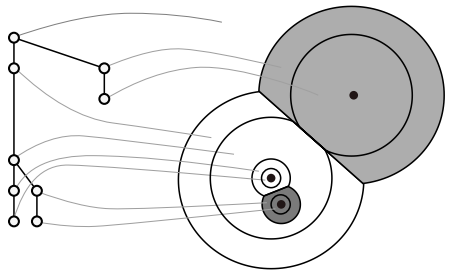


Figure 1: The net-tree on the left induces the approximate Voronoi diagram on the right.

The union of Voronoi cells p^ℓ for all ℓ gives an approximate Voronoi cell for the point p. The following lemma makes this precise.

Lemma 5. *Let T be a net-tree with $c_r > c_c\tau/(\tau-1)$ on a point set P. For any point q, if $\mathcal{C}(q) = p^\ell$, then p is a $(\frac{c_r\tau(\tau-1)}{c_r(\tau-1)-c_c\tau})$-ANN of q in P.*

Proof. Let $m = \lceil \log_\tau \mathbf{d}(p,q)/c_r \rceil$. Clearly, we have $m \le \ell$ and $c_r\tau^{m-1} < \mathbf{d}(p,q) \le c_r\tau^m$. Since $p \in N_m$, $\mathbf{d}(q, N_m) > c_r\tau^{m-1}$. Furthermore, $\mathbf{d}(q, N_{m-1}) > c_r\tau^{m-1}$, because otherwise $\mathcal{C}(q)$ should be a node other than p^ℓ and in level $m-1$. Also note that each node associated to a point in $P \setminus N_{m-1}$ has an ancestor in a level at least $m-1$. Therefore, $\mathbf{d}_H(N_{m-1}, P) \le c_c\tau^m/(\tau-1)$. Now, using the triangle inequality,

$$\begin{aligned}\mathbf{d}(q,P) &\ge \mathbf{d}(q,N_{m-1}) - \mathbf{d}_H(N_{m-1},P)\\ &> c_r\tau^{m-1} - \frac{c_c}{\tau-1}\tau^m\\ &> (\frac{c_r(\tau-1)-c_c\tau}{c_r\tau(\tau-1)})\mathbf{d}(p,q).\end{aligned}$$ □

5 Bottom-up Insertion into a Net-tree

A net-tree can be constructed from the bottom up, adding one point at a time. There are two steps, point location (PL) and propagation. In the PL step, the center of a new point p is computed and p is inserted to the tree based on this information. Note that the newly-inserted node may violate the local covering property if it is too far from every point in the level above. In such cases, the bottom-up propagation algorithm restores the local covering property by repeatedly making new nodes associated to p at each level up the tree until the covering property is restored.

We only require the PL step finds the center for the next point. Once the center is found, p is added to the tree as follows. Let $q^\ell = \mathcal{C}(p)$. If q^ℓ is the top of a jump, then p is inserted as a child of q at level $h = \lceil \log_\tau \mathbf{d}(q,p)/c_r \rceil$ if $\mathbf{d}(p,q) > c_p\tau^h$, or at level $h-1$ otherwise. If q^ℓ is not the top of a jump, then p is inserted at level ℓ and as a child of the closest node to p among the nodes in $\mathrm{Rel}(\mathrm{par}(q^\ell))$. When a new node of p is added to the tree, say p^h, we need to find its relatives and children. We find relatives and children of p^h from $\mathrm{Rel}(\mathrm{par}(p^h))$ and $\mathrm{ch}(\mathrm{Rel}(p^h))$, respectively. If the previous parent of a child of p^h has only one child, then we check the semi-compressed condition for that node to see whether the node should be removed from tree or not.

Theorem 6. *Given a semi-compressed tree $T \in \mathrm{LNT}(\tau, c_p, c_c)$ with $c_r \ge \frac{2c_c\tau}{\tau-2}$ and an uninserted point q with $q^\ell = \mathcal{C}(p)$. The insertion of p into T results a semi-compressed tree $T' \in \mathrm{LNT}(\tau, c_p, c_c + \frac{c_r}{\tau})$.*

If the insertion of a new point p violates the local covering property, we promote the new node p^ℓ to higher levels of the tree as follows. Let $q^{\ell+1} = \mathrm{par}(p^\ell)$. First, we create node $p^{\ell+1}$ and make it as the parent of p^ℓ. Then, we make the closest node among $\mathrm{Rel}(\mathrm{par}(q^{\ell+1}))$ to p as the parent of $p^{\ell+1}$. Finally, we find relatives and children of $p^{\ell+1}$ in a way similar to the insertion method. If node $p^{\ell+1}$ still violates the covering property, we use the same procedure to promote it to higher levels. Here, we use iteration i to indicate promotion of point p to level $\ell+i$.

Lemma 7. *Given $c_r \ge c_c\tau/(\tau-2)$, in the i-th iteration of the bottom-up propagation, $\mathbf{d}(p^{\ell+i}, \mathrm{par}(p^{\ell+i})) \le (c_c + c_r/\tau)\tau^{\ell+i+1} \le c_r\tau^{\ell+i+1}$ and $p^{\ell+i} \sim q^{\ell+i}$.*

Theorem 8. *Given a semi-compressed tree $T \in \mathrm{LNT}(\tau, c_p, c_c + c_r/\tau)$ with $c_r \ge 2c_c\tau/(\tau-2)$ and only one node p^ℓ having $c_c\tau^{\ell+1} < \mathbf{d}(p^\ell, \mathrm{par}(p^\ell)) \le (c_c + c_r/\tau)\tau^{\ell+1}$. The bottom-up propagation method results a semi-compressed tree $T' \in \mathrm{LNT}(\tau, c_p, c_c)$.*

Theorem 9. *Not counting the PL step, the bottom-up construction runs in $O(\rho^{O(1)}n)$ time.*

Proof. In the promotion phase, Lemma 7 implies that every node of $p^{\ell+i}$ has at least one relative besides itself, namely $q^{\ell+i}$. So, we can make $q^{\ell+i}$ responsible to pay the cost of iteration i for p. Note that a node $q^{\ell+i}$ will not be removed by any other points because $p^{\ell+i} \sim q^{\ell+i}$ satisfies the semi-compressed condition. Therefore, to pay the cost of all promotions for all n points, each node in the output requires $\rho^{O(1)}$ charge for each of its relatives. By Theorem 8, the output is semi-compressed and Theorem 4 implies that it has $O(\rho^{O(1)}n)$ size. Thus, using Lemma 2, the total cost of all promotions for all n points is $O(\rho^{O(1)}n)$. □

6 Randomized Incremental Construction

In Section 5, we observed that the efficiency of the bottom-up construction algorithm depends on the PL step. In this section, we show how to eagerly compute the centers of all uninserted points. The centers are updated each time either a new node is added or an existing node is deleted by dong a local search among parents, children, and relatives of the node. We show that the following invariant is satisfied after each insertion or deletion.

Invariant. *The centers of all uninserted points are correctly maintained.*

6.1 The Point Location Algorithm

We describe a simple eager point location algorithm referred to as *the PL algorithm* from here on. The idea is the precompute the center for each uninserted point. We also store the inverse information by also keeping a list of uninserted points for each node in

the tree. The *cluster* of a node p^ℓ, denoted $\mathcal{S}(p^\ell, T)$, is the list of uninserted points in $\mathrm{Vor}(p^\ell)$. We divide each cluster into two sub-clusters $\mathcal{S}_{in}(p^\ell, T) = \{q \in \mathcal{S}(p^\ell, T) \mid \mathbf{d}(p, q) \leq c_p \tau^{\ell-1}/2\}$ and $\mathcal{S}_{out}(p^\ell, T) = \{q \in \mathcal{S}(p^\ell, T) \mid c_p \tau^{\ell-1}/2 < \mathbf{d}(p, q) \leq c_r \tau^\ell\}$.

When a node of p checks an uninserted point q to see if q belongs to its cluster, we say p *touches* q. A touch only happens if a node is either added to or deleted from the tree. Note that a deletion occurs if a node does not satisfy the semi-compressed condition. In other words, we delete a node if it has only one child, one relative (itself) and it is the only child of its parent. When a node p^ℓ is removed, the PL algorithm moves all uninserted points in its inner cluster to the inner cluster of its parent. Then, it checks the uninserted points in the outer cluster of p^ℓ to see if a point belongs to the inner or the outer cluster of $\mathrm{par}(p^\ell)$. Therefore, p touches all uninserted points in the outer cluster of p^ℓ.

In case of insertion, the PL algorithm only requires to check the set of *nearby uninserted points* and update their centers, if necessary. There are two different ways that a new node is created, either it splits a jump or it is inserted as a child of an existing node in the tree. The following cases specify the set of nearby uninserted points. Let T and T' be the trees before and after insertion or deletion of a node, respectively.

(a) A jump from p^h to p^g is split at level ℓ, where $g < \ell < h$: $\mathcal{S}(p^\ell, T') \subseteq \mathcal{S}(p^h, T)$.

(b) p^ℓ is inserted as a child of $s^{\ell+1}$: $\mathcal{S}(p^\ell, T') \subseteq \{\mathcal{S}_{out}(x^h, T) \mid x^h \in \mathrm{Rel}(s^{\ell+1}) \cup \mathrm{ch}(\mathrm{Rel}(s^{\ell+1})) \cup \mathrm{ch}(\mathrm{Rel}(p^\ell))\}$.

Lemma 10. *The PL algorithm correctly maintains the invariant after insertion or deletion of a node.*

6.2 Analysis of the PL Algorithm

To analyze the point location algorithm, we should count the total number of touches, because each touch corresponds to a distance computation. We classify the touches into three groups of *basic touches*, *split touches*, and *merge touches*. If p_i is touched by a new point p_j, then we say a *basic touch* has happened. If p_i is touched by the point of $\mathcal{C}(p_i)$ after the insertion of p_j, then a *split touch* has happened. Intuitively, a split touch in the tree occurs when $\mathcal{C}(p_i)$ is the top of a jump and insertion of p_j results that jump to be split at a lower level. Similarly, If p_i is touched by the point of $\mathcal{C}(p_i)$ after the deletion of $\mathcal{C}(p_i)$ triggered by the insertion of p_j, then a *merge touch* has happened. It is not hard to see that the number of jump touches is bounded by the number of split touches.

We use a backwards analysis to bound the expected number of basic and split touches. The standard approach of using backwards analysis for randomized incremental constructions will not work directly for the tree construction, because the structure of the tree is highly dependent on the order the points were added. Instead, we define random events that can happen for each point p_i of a permutation $\langle p_1, \ldots, p_n \rangle$ at time j, where time j indicates the moment after insertion of the first j points into the tree and $j < i$.

To bound the expected number of split touches, we use the notion of *bunches* near a point p_i, which are nothing more than far enough disjoint groups of points around p_i. Formally, for $j < i$, $\mathcal{B} \subseteq P_j$ is a *bunch* near p_i, if there exists a center $x \in \mathcal{B}$ such that: $\mathcal{B} = \mathbf{B}(x, \alpha\mathbf{d}(p_i, x)) \cap P_j$, $\big[\mathbf{B}(x, \beta\mathbf{d}(p_i, x)) \setminus \mathbf{B}(x, \alpha\mathbf{d}(p_i, x))\big] \cap P_j = \emptyset$, $\mathbf{d}(p_i, x) \leq \frac{2\tau(\tau-1)}{2\tau-3}\mathbf{d}(p_i, P_j)$, where $0 < \alpha \leq 0.5$ and $\beta \geq 2\alpha$. It is not hard to show that there are $\rho^{O(1)}$ number of bunches near each point p_i, for some constants α and β.

We define the events as follows. A basic event $\psi_{i,j}$ happens if p_j is the unique nearest neighbor of p_i among the first j points. It is not hard to see that the expected number of basic events for each point p_i is $O(\log n)$. A split event $\phi_{i,j}$ occurs if there is a bunch $\mathcal{B}$ near p_i, for some constant values of α and β, and p_j is either the unique farthest point in $\mathcal{B}$ or the unique closest point not in $\mathcal{B}$ to the first point in $\mathcal{B}$. The following lemma bounds the expected number of basic and split events.

Lemma 11. *The expected number of basic and split events for a point p_i in a permutation is $O(\rho^{O(1)} \log n)$.*

The expected number of basic and split touches can be counted by the number of basic and split events. We eliminate the proof due to lack of space.

Lemma 12. *The expected number of basic and split touches for a point p_i in a permutation is $O(\rho^{O(1)} \log n)$.*

Therefore, the PL algorithm runs in $O(\rho^{O(1)} n \log n)$ time in expectation for all insertions.

References

[1] A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 97–104, 2006.

[2] K. L. Clarkson. Nearest neighbor searching in metric spaces: Experimental results for sb(s). Avalable from `http://kenclarkson.org/Msb/white_paper.pdf`, 2002.

[3] R. Cole and L.-A. Gottlieb. Searching dynamic point sets in spaces with bounded doubling dimension. In *Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing*, pages 574–583, 2006.

[4] J. Gao, L. J. Guibas, and A. Nguyen. Deformable spanners and applications. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, SCG '04, pages 190–199, New York, NY, USA, 2004. ACM.

[5] L.-A. Gottlieb and L. Roditty. An optimal dynamic spanner for doubling metric spaces. In *Proceedings of the 16th annual European symposium on Algorithms*, pages 468–489, 2008.

[6] S. Har-Peled and M. Mendel. Fast construction of nets in low dimensional metrics, and their applications. *SIAM Journal on Computing*, 35(5):1148–1184, 2006.

[7] R. Krauthgamer and J. R. Lee. Navigating nets: Simple algorithms for proximity search. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 798–807, 2004.

On Computing a Timescale Invariant Bottleneck Distance

Nicholas J. Cavanna, Oliver Kisielius, and Donald R. Sheehy
University of Connecticut

Introduction

Persistent homology is the computation of the topological changes in geometric data over a sequence of scales or times. This is a central tool in the field of topological data analysis. It proves useful when examining data that is noisy or parameter-selection dependent, e.g. data resulting from image processing, biology, or materials sciences [2]. The persistent homology of a growing sequence of spaces can be represented both algebraically and geometrically. We will focus on its geometric representation called a persistence diagram.

A persistence diagram is a multi-set in $\mathbb{R}^2$ constructed from the persistent topological features, where each topological feature is mapped to a point such that its x and y-coordinates are the feature's birth and death time respectively. Given sufficient conditions the persistent homology and persistence diagram can be computed using standard matrix reduction algorithms. Refer to [3] for more details on persistent homology and its computation.

Persistence diagrams are traditionally compared by their bottleneck distance which is equivalent to the ∞-Wasserstein distance, and is closely related to the Earth-mover's distance between sets. Unfortunately, when persistence diagrams are the result of independent data, a choice of at what time parameter the two shall be compared has been made. This can result in arbitrarily large bottleneck distances for similar data. Our goal is to find the optimal translation and the minimal bottleneck distance resulting from linearly shifting the scale parameters and thus eliminating this decision. This can be viewed as a restriction of the problem of finding the matching/bijection between two multi-sets in the plane over all transformations that minimizes the bottleneck cost, a problem motivated by pattern matching and recognition.

Our approach borrows heavily from Efrat et al.'s [4] work on geometric matching under translations in $\mathbb{R}^n$, and the preceding work by Alt et al. [1], where the authors consider geometric matchings under all rigid transformations in $\mathbb{R}^n$. We use the constructions outlined in Kerber and Morozov's work on computing bottleneck distance [5] in which the authors equate computing the bottleneck distance between persistence diagrams and finding a perfect matching with minimal bottleneck cost on a complete bipartite graph.

Efrat et al. proved that one can decide if the minimal bottleneck cost of two multi-sets in the plane is less than some parameter r in $O(n^{1.5}\log n)$ time and that the minimal cost itself can be found in the same time. They also prove that one can compute the minimal bottleneck cost in $O(n^5\log^2 n)$ time, using an oracle that decides if a translation exists that makes the bottleneck cost less than some r in $O(n^5\log n)$ time. They improve upon Alt et al.'s computation and decision oracle complexities of $O(n^6\log n)$ and $O(n^6)$ respectively.

In this work, we prove that the minimum bottleneck distance over all diagonal translations is a pseudo-metric that can be computed in $O(n^3\log^2 n)$ time with a decision oracle running in $O(n^3\log n)$ time.

Background

Given a topological space X, e.g. a simplicial complex or a metric space, consider a real-valued function $f : X \to \mathbb{R}$. Persistent homology computes the topological changes and thus the persistent topological features of the sub-level sets, $X^\alpha := f^{-1}(-\infty, \alpha]$, as α extends to infinity. The growing sequence of spaces $(X^\alpha)_\alpha$ is called a filtration. More specifically, it computes the persistent homology groups which are the images of the maps induced by inclusion $h_*^{\alpha,\beta} : H_*(X^\alpha) \to H_*(X^\beta)$.

An example of a function on a simplicial complex is the function induced by assigning values to each vertex, so that each simplex in X has a real-value associated to it that is the maximum over all values of its vertices. Another standard example is if one has a finite point sample P of a metric space X, consider the distance-to-set function $f_P(x) = \min_{p \in P} \mathrm{d}(x, p)$. The sub-level sets $f_P^{-1}(-\infty, \alpha]$ are the points $x \in X$ that are within α of some point $p \in P$.

Each of the independent features that appear have a scale at which they are born, α, and at which they die, β, i.e. they are homologically trivial. These persistent features can be represented by a point (α, β) in the plane, so the vertical height from the diagonal to a point in the persistence diagram represents its lifespan. These points along with each point (α, α) on the diagonal counted with infinite multiplicity constitute the persistent diagram $D(f)$, where f is the defining function.

Two persistence diagrams A and B are traditionally compared via the bottleneck distance. We call a finite persistence diagram one in which there are only finitely many off-diagonal points. The bottleneck distance between A and B is

$$\mathrm{d}_B(A, B) := \min_{\phi} \max_{a \in A} \|a - \phi(a)\|_\infty,$$

where the minimum is taken over all bijections $\phi : A \to B$.

A complete bipartite graph is an undirected graph $G = (U \sqcup V, U \times V)$, i.e. there are two disjoint vertex sets U and V and there is an edge between vertices u and v if and only if $u \in U$ and $v \in V$. We can construct a complete weighted bipartite graph from two persistence diagrams A and B as follows. See [5] and [3] for more details. Denote A_0 and B_0 as the off-diagonal points of A and B respectively and A_0' and B_0' as the orthogonal projection of the points in A_0 and B_0 respectively onto the diagonal. Define the two disjoint vertex sets as $A_0 \cup B_0'$ and $B_0 \cup A_0'$. The weight of an edge between two vertices $u \in A_0 \cup B_0'$ and $v \in B_0 \cup A_0'$ is $\|u - v\|_\infty$ if $u \in A_0$ or $v \in B_0$, and 0 otherwise. This implies that all edges between diagonal elements have distance 0. A matching M of a bipartite graph is a subset $M \subseteq U \times V$ such that for each vertex $w \in U \sqcup V$, $w \in e$ for at most one $e \in M$. A perfect matching is a matching where the edges correspond to a bijection between U and V. The bottleneck cost of a complete bipartite graph is the minimum over all perfect matchings of the maximum over all of its edge weights.

The following lemma, due to Edelsbrunner and Harer, is called the Reduction Lemma, which implies one can look at matchings in a bipartite graph to compute the bottleneck distance between persistence diagrams, This implies that bottleneck distance between persistence diagrams can be computed in $O(n^{1.5} \log n)$ time by the algorithm in Efrat et al.

Lemma 1. *Given finite persistence diagrams A and B, the bottleneck distance $d_B(A, B)$ is equal to the bottleneck cost of the complete bipartite graph associated with A and B.*

Results

Given a finite persistence diagram A, define the t-translation of A as $A_t := \{(a_x + t, a_y + t) \mid (a_x, a_y) \in A\}$ for $t \in \mathbb{R}$. Denote $a + t := a + t\vec{1}$. Note that this morphism takes persistence

diagrams to persistence diagrams, as it preserves the infinite diagonal. Define the function $\mathrm{d}_B^{\min} := \min_{t\in\mathbb{R}} \mathrm{d}_B(A_t, B)$.

Lemma 2. *$\mathrm{d}_B^{\min}$ is a pseudo-metric on the space of finite persistence diagrams.*

Computing this pseudo-metric efficiently is our goal, as it represents the similarity of the two persistence diagrams regardless of the time parameter.

Recall that we denote the multiset of off-diagonal points of a persistence diagram A by A_0 and the multi-set of orthogonal projections of the points in A_0 by A_0'. We will now introduce a decision oracle to narrow down the potential values of $\mathrm{d}_B^{\min}$.

Lemma 3. *Given finite persistence diagrams A and B, where $|A_0| = |B_0| = n$ and a distance parameter $r > 0$, one can decide if there exists a translation $t \in \mathbb{R}$ such that $\mathrm{d}_B(A_t, B) \leq r$ in $O(n^3 \log n)$ time.*

Proof. The oracle will decide the equivalent problem of whether there exists a translation $t \in \mathbb{R}$ such that there is a perfect matching in the graph $G_t[r]$ – the complete bipartite graph corresponding to the diagrams A_t and B, restricted to edge lengths $\leq r$.

A lemma due to Edelsbrunner and Harer [3] states that there exists an optimal matching where no off-diagonal points are matched to points on the diagonal that are not their orthogonal projection. We can ignore these edges in the perfect matching oracle algorithm in order to translate an off-diagonal point $a \in A_0$ while not adding new edges between the image of another off-diagonal point and $a' \in A_0'$. We will simply store the distance from a to the diagonal and allow the edge between $a + t$ and its new orthogonal projection $a_0 + t$ to be added to a matching.

For each pair of points $a \in A_0$, $b \in B_0$, we compute all the translations of A that result in a being r away from b. These are the critical translations t where $(a + t, b)$ is an edge in $G_t[r]$. The images of a under these translations are the intersections of the line of slope 1 through a and the metric ball $\mathrm{ball}_\infty(b, r)$. There are at most two critical translations per pair, one "lower translation" and one "upper" translation, each computable in constant time. For each such translation, if one considers the translation ε less or more respectively, there is no longer an edge between b and the image of a in $G_t[r]$, so these are exactly the translations under which the graph changes.

Now consider all the translations corresponding to each pair, $O(n^2)$ in total, sort them by their values, measured by the length of the vectors corresponding to the translations, in $O(n^2 \log n)$. Between each of these sorted translations, there is one edge lost or gained in the bipartite graph. Initialize with the least translation t_0 and decide whether there is a perfect matching in $G_{t_0}[r]$ in $O(n^{1.5} \log n)$ time. If there is, then there is a translation such that $\mathrm{d}_B(A_t, B) \leq r$, namely t_0. Otherwise, for $i \geq 1$, consider the matching resulting from t_i, and compute whether there is an augmenting path in $G_{t_{i+1}}[r]$ in $O(n \log n)$ time, a result of Efrat et al. If the resulting matching is a perfect matching, then the oracle outputs yes. If none of the critical translations result in a perfect matching, then the oracle outputs no. This procedure takes $O(n^3 \log n)$ time. □

Theorem 1. *Given two finite persistence diagrams A and B, where $|A_0| = |B_0| = n$, $\mathrm{d}_B^{\min}(A, B)$ can be computed in $O(n^3 \log^2 n)$ time.*

Proof. We first compute a multi-set of critical distances that are the minimal distances between each pair of points $a \in A_0$, $b \in B_0$ achieved by some translation $t \in \mathbb{R}$. These correspond to the minimal r such that there exists a translation t where $(a + t, b)$ is an edge of $G_t[r]$.

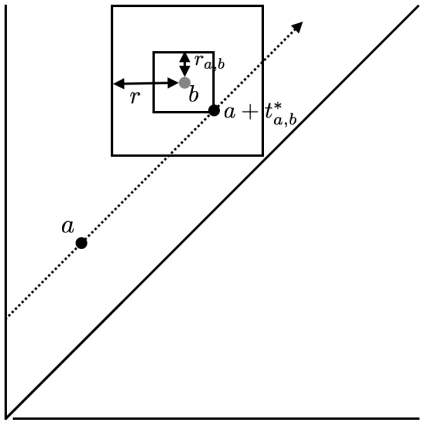


Figure 1: A point a and its optimal translation with respect to b, $a + t^*_{a,b}$, achieving distance $r_{a,b}$. Any decrease in this distance would result in the edge being removed from the bipartite graph.

For each pair a, b, we can compute the distance $r_{a,b}$ in constant time as it is the distance from b to the intersection of the line orthogonal to the diagonal and the line of slope 1 through a. For each pair there is one critical shift corresponding to the distance from a to its translation, so there are n^2 in total. In addition the distances from each point to its orthogonal projection to the diagonal can be bottleneck edges, yielding $n^2 + 2n$ in total. We can sort these in $O(n^2 \log n)$ time. Then for each distance $r_{a,b}$, we can use the oracle from Lemma 3 to decide whether there exists a translation t such that $\mathrm{d}(B_t, A) \leq r_{a,b}$ in $O(n^3 \log n)$ time. As the persistence diagrams are finite, there exists some pair of points a, b such that $\mathrm{d}_B^{\min}(A, B) = \|(a + t^*_{a,b}) - b\|_\infty$ for some critical translation $t^*_{a,b}$.

By performing binary search on the sorted multi-set $\{r_{a,b}\}_{a \in A_0, b \in B_0}$, and looking at lower distances or higher distances depending on a yes or no answer to the oracle, we can compute the minimal distance r^* such that there exists a translation t^* such that $\mathrm{d}_B(A_{t^*}, B) \leq r^*$ in $O(n^3 \log^2 n)$ time. This implies that $\mathrm{d}_B^{\min}(A, B) = r^*$ as it must be achieved by some pair. This procedure also computes one of the optimal translations, as it is one of the translations $t^*_{a,b}$ where $r^* = r_{a,b}$.

□

References

[1] Helmut Alt, Kurt Mehlhorn, Hubert Wagener, and Emo Welzl. Congruence, similarity and symmetries of geometric objects. *Discrete and Computational Geometry*, 3:237–256, 1988.

[2] Herbert Edelsbrunner, , and Dmitriy Morozov. Persistent homology: theory and practice. *Tech. report, Ernest Orlando Lawrence Berkley National Laboratory, Berkely, CA (US)*, 2012.

[3] Herbert Edelsbrunner and John Harer. *Computational Topology. An Introduction.* American Mathematical Society, 2010.

[4] Alon Efrat, Alon Itai, and Matthew J. Katz. Geometry helps in bottleneck matching and related problems. *Algorithmica*, 31(1):1–28, 2001.

[5] Michael Kerber, Dmitriy Morozov, and Arnur Nigmetov. Geometry helps to compare persistence diagrams. *J. Exp. Algorithmics*, 22(1):1.4:1–1.4:20, September 2017.

Improved Results for MINIMUM CONSTRAINT REMOVAL (Extended Abstract)

Eduard Eiben[*] and **Jonathan Gemmell**[†] and **Iyad Kanj**[†] and **Andrew Youngdahl**[†]

A fundamental problem in robot motion planning is to move a robot from a starting position to a final position while avoiding collision with a given set of obstacles. This problem is generally referred to as the *piano-mover*'s problem. If a collision-free path does not exist, which is in general computationally-feasible to decide, one naturally seeks a path that collides with the minimum number of obstacles.

We study a variant of the piano mover's problem, referred to as the MINIMUM CONSTRAINT REMOVAL problem. We are given a set I of polygonal obstacles in the plane and two designated points s and t. We need to compute a minimum subset of obstacles in I whose removal results in an obstacle-free path between s and t. In addition to its applications in robotics, the problem had been studied extensively, motivated by applications in wireless computing, under the name BARRIER COVERAGE or BARRIER RESILIENCE. In such applications, we are given a range field covered by sensors (usually assumed to be simple geometric shapes), and the goal is to compute a minimum set of sensors that need to fail before an entity can move undetected between two given sites (Alt et al. 2011; Tseng and Kirkpatrick 2012; Chan and Kirkpatrick 2014; Kumar, Lai, and Arora 2005; Yang 2012).

The MINIMUM CONSTRAINT REMOVAL problem was also formulated as a graph problem (Chan and Kirkpatrick 2014; Hauser 2014). For an instance I of the problem, the auxiliary graph of I, G_I, is defined. Consider the plane subdivision whose regions are determined by the intersections of the obstacles in I. For each region[1], associate a vertex in G_I representing the set of obstacles intersecting at that region if any, and an empty set otherwise; add an edge between two vertices in G_I iff the corresponding regions share an edge. See Figure 1. Clearly, G_I is a plane graph since it is the dual graph of a plane subdivision. The problem then reduces to computing a path in G_I between the vertices corresponding to s and t, such that the total number of obstacles represented by the vertices on this path is minimum.

[*]Vienna University of Technology, Vienna, Austria, eiben@ac.tuwien.ac.at

[†]School of Computing, DePaul University, Chicago, USA, {jgemmell,ikanj,ayoungda}@cdm.depaul.edu

[1]We do not require the obstacles to contain their interiors. If the intersection of two obstacles is not a 2-D region, we can thicken the borders of the obstacles without changing the sets of obstacles they intersect, so that their intersection becomes a 2-D region.

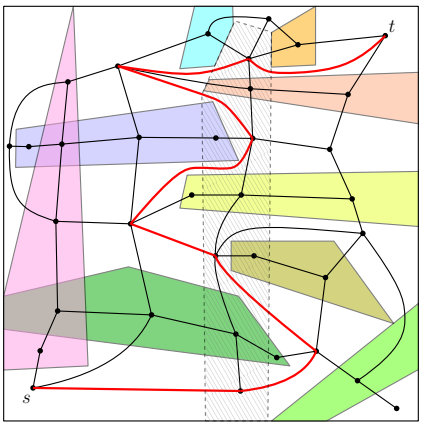


Figure 1: Illustration of the regions determined by a set of obstacles (placed within a bounding box) and its auxiliary graph, with a highlighted optimal path crossing one obstacle.

MINIMUM CONSTRAINT REMOVAL was studied by many researchers in Wireless Computing, AI, and Computational Geometry (Alt et al. 2011; Tseng and Kirkpatrick 2012; Chan and Kirkpatrick 2014; Kumar, Lai, and Arora 2005; Yang 2012; Erickson and LaValle 2013; Hauser 2014). Alt *et al.* (Alt et al. 2011) showed that the problem is NP-hard when the obstacles are line segments such that no three intersect at the same point. Independently, Yang (Yang 2012), in his Ph.D. dissertation, showed the NP-hardness of the problem when the obstacles are line segments. This result was refined independently by Tseng and Kirkpatrick (Tseng and Kirkpatrick 2012) who showed that the problem is NP-hard even when the obstacles are line segments of unit length. The more general graph problem was considered by several researchers, including (Chan and Kirkpatrick 2014; Hauser 2014), and it is well known to be NP-hard (for instance, see (Hauser 2014) for a proof).

In this paper, we continue the study of the MINIMUM CONSTRAINT REMOVAL problem. We first consider the complexity of the problem, and show the following:

(1) MINIMUM CONSTRAINT REMOVAL is NP-hard even if all the obstacles are axes-parallel rectangles.
(2) MINIMUM CONSTRAINT REMOVAL is NP-hard even if all the obstacles are line segments such that no three intersect at the same point.

The results in (1) and (2) refine and improve the earlier work on the problem. More specifically, the result in (1)

answers an open question posed in (Erickson and LaValle 2013). Even though the result in (2) was obtained earlier by Alt *et al.* (Alt et al. 2011), which was also proved by Yang (Yang 2012) and Chan and Kirkpatrick (Tseng and Kirkpatrick 2012) but without the assumption that no three segments intersect at a point, the NP-hardness reduction we use to prove (2) is more refined than the reductions used in the other papers. In particular, the reduction we use to prove (2) implies the ETH results in (3) and (5) below, and those cannot follow from the reductions in the earlier results. As a byproduct of the reductions used to derive the above NP-hardness results, we obtain the following:

(3) Unless the Exponential-Time Hypothesis (ETH) fails, MINIMUM CONSTRAINT REMOVAL cannot be solved in subexponential time $2^{o(n)}$, where n is the number of obstacles in the instance.

The result in (3) shows that significant improvement on the $2^{\mathcal{O}(n)}$-time brute-force algorithm is unlikely, as ETH is a standard hypothesis for proving lower bounds (Lokshtanov, Marx, and Saurabh 2011), which states that the satisfiability of k-CNF formulas (for $k \geq 3$) is not solvable in subexponential-time $2^{o(n)}$, where n is the number of variables in the formula.

We then design algorithms for the NP-hard restriction of CONSTRAINT REMOVAL to instances in which no more than a constant number b of obstacles overlap at the same point, denoted b-OVERLAP MINIMUM CONSTRAINT REMOVAL, for any integer-constant $b \geq 2$. We show that:

(4) There is a subexponential-time algorithm for b-OVERLAP MINIMUM CONSTRAINT REMOVAL that runs in time $2^{\mathcal{O}(\sqrt{N})}$, where N is the number of the vertices in the auxiliary graph associated with the instance; and

(5) unless ETH fails, b-OVERLAP MINIMUM CONSTRAINT REMOVAL cannot be solved in time $2^{o(\sqrt{N})}$.

The result in (4) gives a subexponential-time algorithm for b-OVERLAP MINIMUM CONSTRAINT REMOVAL w.r.t. the number of obstacles n, for instances in which the number of regions N, equivalently vertices in G_I, is $o(n^2)$.

Hardness Results

Consider the decision version of MINIMUM CONSTRAINT REMOVAL, denoted CONSTRAINT REMOVAL, in which we are given a set I of obstacles, two points s and t, and $k \in \mathbb{N}$, and we need to decide if there is an s-t path that intersects at most k obstacles in I. We first show that CONSTRAINT REMOVAL remains NP-hard even when the obstacles are axes-parallel rectangles. We do so via a reduction from the NP-hard problem MAXIMUM NEGATIVE 2-SATISFIABILITY that yields instances of CONSTRAINT REMOVAL in which the obstacles are axes-parallel rectangles. The reduction works for both cases when the interior of the rectangle is considered as part of the obstacle and when it is not. Second, we show how our reduction can be modified to yield a refined reduction in which the obstacles are line segments such that no three of them intersect at the same point.

To obtain the hardness results, we reduce from an NP-hard restriction of the MAXIMUM NEGATIVE 2-SATISFIABILITY problem. An instance of MAXIMUM NEGATIVE 2-SATISFIABILITY is given as a pair (F, m'), where $m' \in \mathbb{N}$ and F is a Boolean formula on n variables and m clauses. The question is to decide whether F has a satisfying assignment that satisfies at least m' clauses. The NP-hard restriction of MAXIMUM NEGATIVE 2-SATISFIABILITY we use, denoted R-MN2Sat, satisfies the following properties: (1) Each clause in the formula F is either a unit clause containing a positive literal $\{x_i\}$, or a binary clause containing two negative literals; (2) the unit clauses in F are precisely the clauses $\{x_i\}$, for each variable x_i in F; and (3) the number of both positive and negative occurrences of each variable is at most 4. A consequence of (3) is that the number of clauses m in F is at most $n + 3n/2 \leq 3n$. It can be easily shown that R-MN2Sat is NP-hard via a straightforward reduction from INDEPENDENT SET on graphs of maximum degree at most 3. It follows from (Johnson and Szegedy 1999), that unless ETH fails, INDEPENDENT SET on graphs of maximum degree at most 3 is not solvable in time $2^{o(n)}$, where n is the number of vertices in the graph. It follows from the reduction from INDEPENDENT SET to R-MN2Sat that, unless ETH fails, R-MN2SAT is not solvable in time $2^{o(n)}$. This result is used to derive lower-bound results on the subexponential-time complexity of MINIMUM CONSTRAINT REMOVAL.

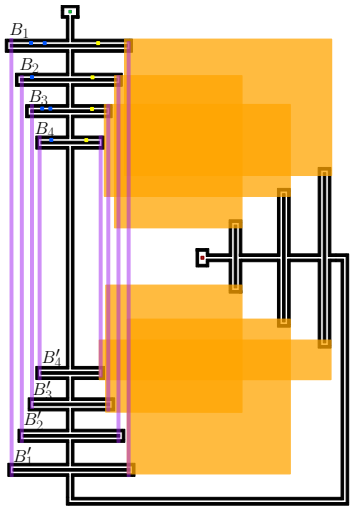


Figure 2: Illustration for the proof of Theorem 1 for $F = x_1 \wedge x_2 \wedge x_3 \wedge x_4 \wedge (\bar{x}_1 \vee \bar{x}_4) \wedge (\bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_3 \vee \bar{x}_1)$.

Axes-parallel Rectangles: Let RECTANGLE-CONSTRAINT REMOVAL be the restriction of CONSTRAINT REMOVAL to instances in which each obstacle is an axes-parallel rectangle.

Theorem 1. RECTANGLE-CONSTRAINT REMOVAL *is* NP*-hard.*

Proof. (Sketch) We give a polynomial-time reduction from R-MN2Sat to the decision version of RECTANGLE-CONSTRAINT REMOVAL. See Figure 2 for illustration.

Let (F, m') be an instance of R-MN2Sat, where F has n variables $x_1, \ldots, x_n$ and m clauses $C_1, \ldots, C_m$. Let $m_1 = n$ be the number of unit clauses in F, and $m_2 = m - n$ be the

number of remaining (binary) clauses. To construct the instance (I, k) of RECTANGLE-CONSTRAINT REMOVAL, we start by enforcing a schema that any valid path must be confined within. We first outline a rectilinear region $\mathcal{R}$, whose boundary is shown in black in Figure 2. This region consists of an open rectangular region containing the starting point s (green); a rectilinear corridor $\mathcal{C}$ that runs vertically and then makes three left turns; a set of rectangular-like "boxes" intersecting $\mathcal{C}$ in their middle, each enclosing a black rectangle; and an open rectangular region containing the destination point t (burgundy).

For each variable x_i, we associate two boxes, B_i and B'_i, of the same shape and size, laid out horizontally and symmetrically on opposite ends of the vertical part of corridor $\mathcal{C}$ (before $\mathcal{C}$ turns). We call the B_i's and B'_i's *variable-boxes*. The B_i's appear on the top and run in decreasing horizontal dimension, and the B'_i's appear on the bottom and run in increasing horizontal dimension. The order in which the B_i's appear is the opposite order of the B'_i's. Each B_i (resp. B'_i) contains a black rectangle inside, subdividing it into an upper and a lower part, and creating a left passage and a right passage from its upper to its lower part. For each binary clause C_j, we associate a box B_{C_j} that we call a *clause-box*; the clause-boxes are placed towards the end of $\mathcal{C}$, just before the destination point t, and are laid out vertically.

To place the obstacles of I in $\mathcal{R}$, we start by fixing two integer-constants: $c_1 > 4$ and $c_2 > c_1 \cdot n + 3m$. To simplify, we assign integer weights to certain obstacles to indicate an overlaying of the weight-many distinct identical-shape obstacles. Intuitively, the obstacles will be placed so that the portion of the path traversing the variable-boxes corresponds to an assignment of the variables in F, where traversing the left (resp. right) side of box B_i corresponds to assigning x_i to TRUE (resp. FALSE); the other portion of the path, traversing the clause-boxes, can be done in such a way that if a literal has been assigned TRUE by the first portion of the path, the clauses containing the literal can be traversed at no additional cost. To confine the path to the interior of region $\mathcal{R}$, we form the boundary of $\mathcal{R}$ by placing axes-parallel rectangular obstacles (shown in the same black color as the boundary), each of weight c_2, along this boundary so that they only (pairwise) overlap (in small squares) to form the corners of $\mathcal{R}$; this ensures that the cost of crossing the boundary of $\mathcal{R}$ exceeds the required budget k, which will be specified shortly. Similarly, each black rectangle outlined inside a box in $\mathcal{R}$, whose role is to block the direct passage of a path from one box to the next without setting the truth assignment of the variable associated with the first box, is formed using an axes-parallel rectangular obstacle of weight c_2; this way, the desired path cannot intersect any of these internal rectangles. We refer to the obstacles of weight c_2 as *heavy* obstacles.

For each binary clause C_j in F, we arbitrarily order the two (negative) literals in C_j as first and second. For each literal $\overline{x_i}$ and clause C_j such that $\overline{x_i}$ is the first (resp. second) literal in C_j, we create an axes-parallel rectangular obstacle of weight 1 (orange) that intersects the right side of box B_i (resp. B'_i) including the internal rectangle, *without* intersecting any other variable-boxes, and intersects the top (resp. bottom) of the clause-box corresponding to C_j (including the internal rectangle) *without* intersecting any other clause-boxes (see Figure 2). These obstacles ensure that a path that sets a literal $\overline{x_i}$ to TRUE can traverse C_j at no additional cost. For each positive clause $\{x_i\}$, we place an axes-parallel rectangular obstacle of weight 1 (yellow) in the right side of B_i so that any path setting x_i to false intersects this obstacle. We call all these (orange plus yellow) obstacles *incidency* obstacles. For each variable x_i, we add two axes-parallel rectangular obstacles (purple), each of weight c_1. The first obstacle intersects the left sides of B_i and B'_i *without* intersecting any other variable-boxes, and the second intersects the right sides of these boxes; these obstacles, referred to as *consistency obstacles*, are used to ensure that we do not set both a variable and its negation to TRUE.

Finally, for each x_i, let p_i be the number of occurrences of $\overline{x_i}$ in F; we place p_i many weight-1 axes-parallel rectangular obstacles (blue), referred to as *balancing* obstacles, in the left side of B_i. Let $k = c_1 \cdot n + 2m_2 + m_1 - (m' - m_2)$, and note that $k < c_2$. This completes the construction of the instance (I, k) of RECTANGLE-CONSTRAINT REMOVAL.

It can be shown that (F, m') is a yes-instance of R-MN2SAT iff (I, k) is a yes-instance of RECTANGLE-CONSTRAINT REMOVAL. □

Straight-line Segments: For an instance I of CONSTRAINT REMOVAL, define the *overlap number* of I to be the maximum number of obstacles whose intersection is nonempty. Let LINE-CONSTRAINT REMOVAL be the restriction of CONSTRAINT REMOVAL to instances in which each obstacle is a line segment.

We can prove the following theorem again via a reduction from R-MN2Sat. The reduction is similar to that in Theorem 1, except for the shapes and layout of the obstacles, as we no longer can overlay obstacles since we need to keep the overlap number at most 2. For that, we use line-segments to mimic the rectangles used in the proof of Theorem 1, for each obstacle-type used in that proof.

Theorem 2. LINE-CONSTRAINT REMOVAL*, restricted to instance whose overlap number is at most 2, is* NP*-hard.*

The following corollaries are direct consequences of the reduction used to prove Theorem 2:

Corollary 3. *Unless ETH fails,* LINE-CONSTRAINT REMOVAL *restricted to instance whose overlap number is at most 2 cannot be solved in time $2^{o(\sqrt{N})}$, where N is the number of regions in the input instance.*

Corollary 4. *Unless ETH fails,* CONSTRAINT REMOVAL *restricted to instance whose overlap number is at most 2 cannot be solved in time $2^{o(n)}$, where n is the number of obstacles in the input instance.*

Subexponential-time Algorithm

For an integer $b \geq 2$, define b-OVERLAP MINIMUM CONSTRAINT REMOVAL to be the restriction of MINIMUM CONSTRAINT REMOVAL to instances whose overlap number is at most b. (For $b \geq 2$, b-OVERLAP MINIMUM CONSTRAINT REMOVAL is NP-hard by Theorem 2.) We

present a divide-and-conquer algorithm, based on a variant of the well-known balanced separator theorem for planar graphs (Miller 1986), that solves an instance I of b-OVERLAP MINIMUM CONSTRAINT REMOVAL in time $2^{\mathcal{O}(\sqrt{N})}$, where N is the number of vertices in the auxiliary graph G_I. This variant theorem states that the vertex-set of a triangulated plane graph on N vertices can be partitioned into three parts A, B, S such that: (1) S is a cycle separating A from B (*i.e.*, no edge exists between A and B) and $|S| \leq \sqrt{8N}$; (2) $|A| \leq 2N/3$ and $|B| \leq 2N/3$; and (3) A is interior to S and B is exterior to S (w.r.t. the plane embedding). Our algorithm follows the approach in Woeginger et al. (Deineko, Klinz, and Woeginger 2006), for computing a Hamiltonian path in a planar graph on N vertices in time $2^{\mathcal{O}(\sqrt{N})}$. There are complications, however, that are particular to b-OVERLAP MINIMUM CONSTRAINT REMOVAL. We describe below how to deal with these complications.

Consider an instance I of b-OVERLAP MINIMUM CONSTRAINT REMOVAL on n obstacles, and let G_I be its auxiliary graph. We assign each obstacle in I a distinct representative color and assume that each vertex v in G_I is colored by the color-set representing the obstacles forming the region of v. As in (Deineko, Klinz, and Woeginger 2006), we add edges to G_I so that the resulting graph is a triangulation, and then apply the cycle separator theorem (Miller 1986) to partition the vertex-set of the resulting graph into A, B, S; the added edges are removed afterwards, and play no role other than determining A, B, S.

The algorithm maintains a configuration, which is a tuple, and an auxiliary graph stipulating a partial ordering that the current enumeration dictates on the path vertices. We skip these details since they are very similar to those in (Deineko, Klinz, and Woeginger 2006), and highlight those that are particular to b-OVERLAP MINIMUM CONSTRAINT REMOVAL. After computing A, B, S, we enumerate every subset of S, as the subset of vertices that are contained in the path, P, we seek. For each enumerated subset F, we enumerate every subset of colors C that appear both on vertices in $S \setminus F$ and on P. We then remove all colors in S from G_I, and mark every vertex containing a color that is in $S \setminus F$ but not in C as "forbidden". Afterwards, the color-set appearing on vertices in A is disjoint from that appearing on vertices in B, because the colors that appear in both A and B must appear in S (the vertices on which the same color appears induce a connected subgraph of G_I), and those colors have been removed. The number of enumerations so far is at most $2^{\mathcal{O}(\sqrt{8N})} \cdot 2^{\mathcal{O}(b \cdot \sqrt{8N})} = 2^{\mathcal{O}(\sqrt{N})}$.

Fix such an enumeration. Next, we need to enumerate the order in which P traverses the vertices in F. Enumerating all permutations of the vertices in F will not result in a $2^{\mathcal{O}(\sqrt{N})}$-time algorithm. Instead, we adopt a similar enumeration method to the one in Woeginger et al. (Deineko, Klinz, and Woeginger 2006), which is based on the following observation. Suppose for now that the order in which P visits the vertices in F has been revealed. For any two nonadjacent vertices u, v in $F \cap V(P)$, say that u and v are *A-consecutive* (resp. *B-consecutive*) on P if the subpath of P between u and v, excluding u and v, is contained in A (resp. in B). Let $E_A \subseteq F \times F$ (resp. $E_B \subseteq F \times F$) be the set of edges between A-consecutive (resp. B-consecutive) vertices (these edges are not in G_I). The algorithm makes two recursive calls, one on $G_I[A \cup S] + E_B$ after modifying the auxiliary structure so that to enforce the order imposed by E_A and E_B, and the other on $G_I[B \cup S] + E_A$ after modifying the auxiliary structure so that to enforce the order imposed by E_A and E_B. The algorithm returns an s-t path that is the concatenation of a path having the minimum number of colors resulting from the recursive call on $G_I[A \cup S] + E_B$, with a path having the minimum number of colors resulting from the recursive call on $G_I[B \cup S] + E_A$.

It can be shown that we can enumerate E_A and E_B efficiently without enumerating all permutations of F in time $2^{\mathcal{O}(\sqrt{N})}$, which leads to the same time upper bound for the whole algorithm. We conclude with:

Theorem 5. *b-OVERLAP MINIMUM CONSTRAINT REMOVAL can be solved in time $2^{\mathcal{O}(\sqrt{N})}$, and unless ETH fails, it cannot be solved in time $2^{o(\sqrt{N})}$, even when the obstacles are line segments.*

References

Alt, H.; Cabello, S.; Giannopoulos, P.; and Knauer, C. 2011. On some connection problems in straight-line segment arrangements. In *the 24th European Workshop on Computational Geometry*, 27–30.

Chan, D., and Kirkpatrick, D. 2014. Multi-path algorithms for minimum-colour path problems with applications to approximating barrier resilience. *Theoretical Computer Science* 553:74–90.

Deineko, V.; Klinz, B.; and Woeginger, G. 2006. Exact algorithms for the hamiltonian cycle problem in planar graphs. *Operations Research Letters* 34(3):269–274.

Erickson, L., and LaValle, S. 2013. A simple, but NP-hard, motion planning problem. In *Proceedings of AAAI*. AAAI Press.

Hauser, K. 2014. The minimum constraint removal problem with three robotics applications. *International Journal of Robotics Research* 33(1):5–17.

Johnson, D., and Szegedy, M. 1999. What are the least tractable instances of Max independent set? In *Proceedings of SODA*, 927–928. ACM/SIAM.

Kumar, S.; Lai, T.; and Arora, A. 2005. Barrier coverage with wireless sensors. In *Proceedings of the 11th Annual International Conference on Mobile Computing and Networking MOBICOM*, 284–298. ACM.

Lokshtanov, D.; Marx, D.; and Saurabh, S. 2011. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS* 105:41–72.

Miller, G. 1986. Finding small simple cycle separators for 2-connected planar graphs. *Journal of Computer and System Sciences* 32(3):265–279.

Tseng, K., and Kirkpatrick, D. 2012. On barrier resilience of sensor networks. In *International Symposium on Algorithms for Sensor Systems, Wireless Ad Hoc Networks and Autonomous Mobile Entities Algosensors*, volume 7111 of *Lecture Notes in Computer Science*, 130–144. Springer.

Yang, S. 2012. *Some Path Planning Algorithms in Computational Geometry and Air Traffic Management*. Ph.D. Dissertation, University of New Yort at Stony Brook. Available at: https://dspace.sunyconnect.suny.edu/handle/1951/59927.

How to navigate a robot through obstacles?

Eduard Eiben* Iyad Kanj†

1 Problem Definition and Motivation

Motion planning is an important subject with applications in Robotics, Computational Geometry, Graphics, and Gaming, among others. The goal in motion planning problems is generally to move a robot from a starting position to a final position, while avoiding collision with a set of obstacles. This is usually referred to as the *piano-mover*'s problem.

This work is concerned with a variant of the *piano-mover*'s problem, where the obstacles are in the Euclidean plane and the robot is represented as a point. Since determining if there is an obstacle-free path for the robot in this case is solvable in polynomial time, if no such path exists, it is natural to seek a path that intersects as few obstacles as possible. More formally, in this setting, we are given a set of obstacles in the plane and $k \in \mathbb{N}$, and we need to determine if there is a path for the robot between two given points that intersects at most k obstacles; equivalently, we need to determine if we can remove at most k obstacles so that there is an obstacle-free path for the robot. This problem has also been studied extensively, motivated by applications in wireless computing, under the name Barrier Coverage or Barrier Resilience. In such applications, we are given a range field covered by sensors (usually assumed to be simple geometric shapes), and the goal is to compute a minimum set of sensors that need to fail before an entity can move undetected between two given sites [1, 2, 6, 7, 8, 9].

The problem was formulated and generalized into the following graph problem, by considering the auxiliary plane graph that is the dual of the plane subdivision determined by the obstacles. Given a plane graph G, each of whose vertices is colored by a (possibly empty) color set, two designated vertices $s, t \in V(G)$, and $k \in \mathbb{N}$, decide if there is an s-t path in G that uses at most k colors. See Figure 1 for illustrations.

This problem was studied by several research communities, including Computational Geometry, AI, and Wireless Computing, albeit under different names sometimes [1, 2, 3, 4, 5, 6, 7, 8, 9]. The problem is NP-hard even when the obstacles are simple geometric shapes, such as line segments or axes-parallel rectangles [1, 3, 4, 5, 8, 9]. The general graph problem was known to be NP-hard even earlier. The case when the obstacles are unit disk has received significant attention [2, 6, 7, 8, 9]; it is complexity remains open, but it was shown to be FPT by [6], who also extended the results to when the obstacles are fat regions.

2 Results and Techniques

We study the parameterized complexity of the general graph problem w.r.t. the auxiliary plane graph that models the geometric instances of the problem. We refer to this problem henceforth as the Obstacle Removal problem. As we show and discuss later, Obstacle Removal turns

*Algorithms and Complexity Group, TU Wien, Austria. Email: eiben@ac.tuwien.ac.at

†School of Computing, DePaul University, Chicago, USA. Email: ikanj@cs.depaul.edu

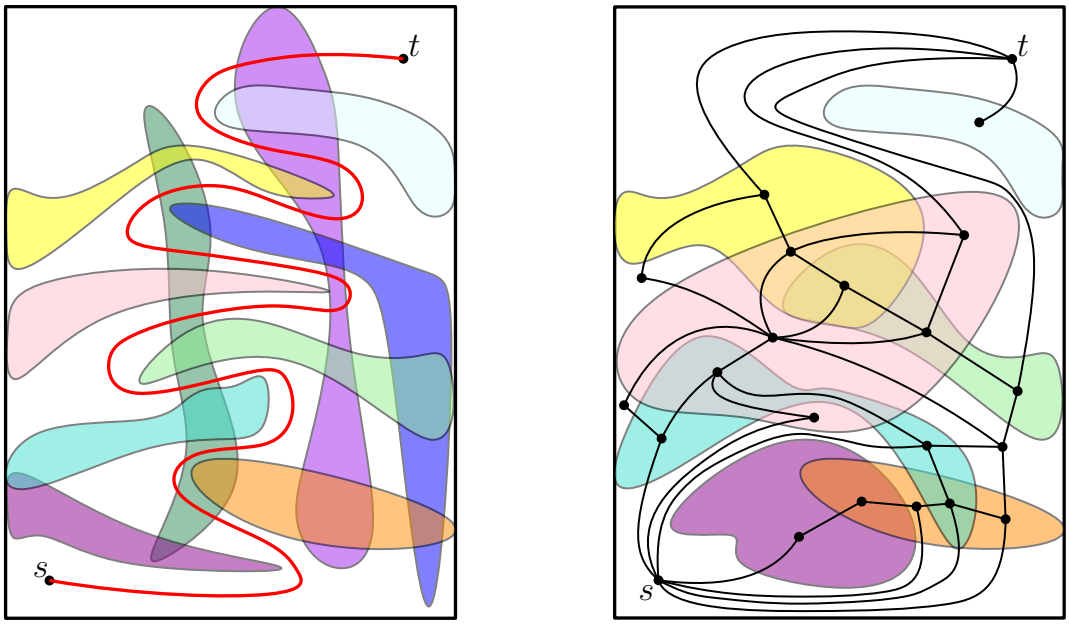


Figure 1: Illustration of instances of the problem drawn within a bounding box. The figure on the left shows an instance in which the optimal path crosses two obstacles, zigzagging between the other obstacles. The figure on the right shows an instance and its auxiliary plane graph.

out to be intractable w.r.t. the framework of parameterized complexity if some of the obstacles are not connected regions. As a result, we focus our study on a restriction of Obstacle Removal to instances satisfying that, for every color in the graph, the set of vertices on which this color appears induces a connected subgraph; we refer to this restriction as Connected Obstacle Removal. Clearly, Obstacle Removal and Connected Obstacle Removal model and generalize their geometric counterparts, referred to as Geomertic Obstacle Removal and Geomertic Connected Obstacle Removal, respectively, that are differentiated based on whether or not the obstacles are connected regions of the plane.

2.1 Hardness Results

Our first hardness result shows that both Obstacle Removal and Connected Obstacle Removal are NP-hard, even when the auxiliary graph has small outerplanarity and pathwidth:

Theorem 2.1. Obstacle Removal *is* NP*-complete, even for outerplanar graphs of pathwidth at most 2 and in which every vertex contains at most one color.*

Theorem 2.2. Connected Obstacle Removal *is* NP*-complete even for 2-outerplanar graphs of pathwidth at most 3.*

The combinatorial instances produced by the hardness results above can be realized as geometric instances of Geomertic Obstacle Removal and Geomertic Connected Obstacle Removal, thus showing that the above hardness results extend to restrictions of Geomertic Obstacle Removal and Geomertic Connected Obstacle Removal.

We then study the parameterized complexity of Obstacle Removal and Connected Obstacle Removal. Our first set of results shows that the color-connectivity property, and hence the connectivity of each obstacle, is crucial for any hope for an FPT-algorithm w.r.t. k, as we show that the combined parameterizations of Obstacle Removal are W[1]-complete:

Theorem 2.3. Obstacle Removal*, restricted to instances of pathwidth at most 4, and in which each vertex contains at most one color and each color appears on at most 2 vertices, is* W[1]-complete parameterized by k.

Theorem 2.4. Obstacle Removal*, parameterized by both k and the length of the sought path ℓ, is* W[1]-complete.

Without any restrictions, Obstacle Removal sits high in the parameterized hierarchy:

Theorem 2.5. Obstacle Removal*, parameterized by k, is* W[SAT]*-hard and is in* W[P].

By producing a generic construction that can be used to realize any combinatorial instance of Obstacle Removal as a geometric instance of Geomertic Obstacle Removal, the above results about Obstacle Removal extend to restrictions of Geomertic Obstacle Removal.

2.2 FPT Results and Applications

After establishing the aforementioned hardness results, we focus our attention on Connected Obstacle Removal. We show the following result:

Theorem 2.6. Connected Obstacle Removal*, parameterized by both k and the treewidth of the input graph, is* FPT.

The folklore dynamic programming approach based on tree decomposition, used for the Hamiltonian Path/Cycle problems, does not work for Connected Obstacle Removal. As opposed to the Hamiltonian Path/Cycle problems, where it is sufficient to keep track of how the path/cycle interacts with each bag in the tree decomposition, this is not sufficient in the case of Connected Obstacle Removal because we also need to keep track of which color sets are used on both sides of the bag. Although (by color-connectivity) any subset of colors appearing on both sides of the bag must appear on vertices in the bag as well, there can be too many such subsets, and certainly we cannot afford to enumerate all of them if we seek an FPT algorithm. To overcome this issue, we prove structural results that exploit the planarity of the graph and the connectivity of the colors to show the following. For any vertex $w \in V(G)$, and for any pair of vertices $u, v \in V(G)$, the set of (valid) u-v paths in $G - w$ that use colors appearing on vertices in the face of $G - w$ containing w can be "represented" by a minimal set of paths $\mathcal{P}$ whose cardinality is a function of k. To derive such an upper bound on the cardinality of $\mathcal{P}$, we select a maximal set $\mathcal{M}$ of color-disjoint paths in $\mathcal{P}$, and show that the cardinality of $\mathcal{P}$ is upper bounded by that of $\mathcal{M}$ multiplied by some function of k. The problem then reduces to upper bounding $|\mathcal{M}|$. To do so, we use an inductive proof whose main ingredient is showing that the subgraph induced by the paths in $\mathcal{M}$ has a u-v vertex-separator of cardinality $O(k)$. We then upper bound $|\mathcal{M}|$ by bounding the number of different traces of the paths of $\mathcal{M}$ on this small separator, and inducting on both sides of the separator.

We extend the notion of a minimal set of paths w.r.t. a single vertex to a "representative set" of paths w.r.t. a specific bag, and a specific enumerated configuration for the bag, in a tree decomposition of the input graph. This enables us to use the upper bound on the cardinality of a minimal set of paths to upper bound the size of a representative set of paths w.r.t. a bag and a configuration. This, in turn, yields an upper bound on the size of the table stored at a bag, in the dynamic programming algorithm by a function of both k and the treewidth of the input graph, thus yielding the desired result.

We extend the FPT results for Connected Obstacle Removal w.r.t. the combined parameters k and the treewidth of the auxiliary graph, to show that Connected Obstacle Removal parameterized by both k and the length ℓ of the sought path is FPT. This is shown by first showing that we can upper bound the treewidth of the graph and then using Theorem 2.6:

Theorem 2.7. Connected Obstacle Removal, *parameterized by both k and the length of the path is* FPT.

We show several applications of the above result. First, we can show that it directly implies the FPT results by Korman*et al.* [6] for the case when the obstacles are unit disks or fat regions. Second, we show that the above result answers an open question posed in [4] as follows.

We define the *intersection number* of the auxiliary graph to be the maximum number (over all colors) of vertices on which the same color appears. Auxiliary graphs with bounded intersection number model the case in which the obstacles are arbitrary connected convex regions satisfying that the number of regions intersected by any region is bounded, as it is easy to see that the intersection number of the auxiliary graph of such instances will be bounded. Note that convexity is essential here, as otherwise, the intersection number of the auxiliary graph may be unbounded.

Theorem 2.8. *Let h be a computable function. The restriction of* Geomertic Connected Obstacle Removal *to any set of connected convex obstacles in the plane satisfying that each obstacle intersects at most $h(k)$ other obstacles, is* FPT *parameterized by k.*

Whereas the complexity of the problem in Theorem 2.8 is open, the theorem settles its parameterized complexity by showing it to be in FPT. We finally mention that it remains open whether (Geometric) Connected Obstacle Removal is FPT parameterized by k only.

References

[1] H. Alt, S. Cabello, P. Giannopoulos, and C. Knauer. On some connection problems in straight-line segment arrangements. In *Proceedings of EuroCG*, pages 27–30, 2011.

[2] D. Chan and D. Kirkpatrick. Multi-path algorithms for minimum-colour path problems with applications to approximating barrier resilience. *Theoretical Computer Science*, 553:74–90, 2014.

[3] E. Eiben, J. Gemmell, I. Kanj, and A. Youngdahl. Improved results for minimum constraint removal, 2017. Under submission to a double-blind reviewed conference.

[4] L. Erickson and S. LaValle. A simple, but NP-hard, motion planning problem. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[5] K. Hauser. The minimum constraint removal problem with three robotics applications. *International Journal of Robotics Research*, 33(1):5–17, 2014.

[6] M. Korman, M. Löffler, R. Silveira, and D. Strash. On the complexity of barrier resilience for fat regions. In *Proceedins of ALGOSENSORS*, pages 201–216, 2014.

[7] S. Kumar, T. Lai, and A. Arora. Barrier coverage with wireless sensors. In *Proceedings of MOBICOM*, pages 284–298. ACM, 2005.

[8] K. Tseng and D. Kirkpatrick. On barrier resilience of sensor networks. In *Proceedings of ALGOSENSORS*, pages 130–144, 2012.

[9] S. Yang. *Some Path Planning Algorithms in Computational Geometry and Air Traffic Management.* PhD thesis, University of New Yort at Stony Brook. Available at: https://dspace.sunyconnect.suny.edu/handle/1951/59927, 2012.

Algorithm for Optimal Chance Constrained Knapsack with Applications to Multi-robot Teaming

Fan Yang[1] and Nilanjan Chakraborty[2]

***Abstract*— Motivated by applications in multirobot team selection, in this paper, we present a novel algorithm for solving chance-constrained 0-1 knapsack problem, where the objective function is deterministic but the weights of the items are stochastic and therefore the knapsack constraint is stochastic. We convert the chance-constrained knapsack problem to a two-dimensional discrete optimization problem on the variance-mean plane, where each point on the plane can be identified with an assignment of items to the knapsack. By exploiting the geometry of the non-convex feasible region of the chance-constrained knapsack problem in the variance-mean plane, we present a novel deterministic technique to find an optimal solution by solving a sequence of deterministic knapsack problems (called risk-averse knapsack problem). We apply our algorithm to a multirobot team selection problem to cover a given route, where the length of the route is much larger than the length each individual robot can fly and the length that an individual robot can fly is a random variable (with known mean and variance). We present simulation results on randomly generated data to demonstrate that our approach is scalable with both the number of robots and increasing uncertainty of the distance an individual robot can travel.**

I. Introduction

The knapsack problem is a fundamental problem in combinatorial optimization that has multiple applications in task allocation and team formation in multi-robot systems. For example, in algorithms to solve the generalized assignment problem for multiple robots, the knapsack problem is a subproblem that needs to be solved multiple times [6]. In this paper, we consider a multirobot team formation problem, where we consider a group of heterogeneous robots that has to cover a given route with known length. Each robot has a limited battery life and therefore there is a upper limit on the distance that the robot can travel. Furthermore, the travel distances are uncertain because they depend on uncertain environmental variables like wind speed. We assume that the lengths that robots can travel are independent Gaussian random variables with known means and variances. There is operating cost for each robot. The total cost of covering the route is a sum of individual costs of robots. Our goal is to find a team of robots with the minimum total cost that covers the route with high probability (specified *a priori*).

The deterministic version of our problem where the travel distances are known constants can be formulated as a 0-1 knapsack problem. There are many methods to solve the knapsack problem such as dynamic programming [12], branch and bound method [9] and other methods that combine both methods [10], [7], [8]. Although solving knapsack problem is NP-hard, there is a fully polynomial time approximation scheme [12]. There are different stochastic variations of the classical 0-1 knapsack problem that have been studied in the extant literature. In [1], [4], [11], the authors have studied the stochastic knapsack problem with deterministic weights and random costs whereas in our problem we have deterministic costs and random weights. In [2], the authors compute a solution policy that optimize the expected total values. Optimizing expected values provide no performance guarantees on a particular realization of the random variables. We want to develop methods that ensures the constraints are satisfied with a high probability irrespective of the realization of the random weights. An algorithm is designed to obtain good solutions to the chance-constrained problem in [5], by running a sequence of robust problems. The algorithm provides an optimal solution when the costs are identical or the uncertain weights present all the same characteristic. In this paper, our method computes the optimal solution in more general situation. In [3], the authors consider a stochastic knapsack problem similar to our setting and provide a polynomial time approximation scheme (PTAS) by using a parametric linear programming reformulation. Our solution to the chance-constrained problem is based on a geometric interpretation of the problem on variance-mean plane. Our method finds the optimal solution of chance-constrained problem by solving a sequence of a deterministic knapsack problems called the risk-averse knapsack problems.

Contributions: In this paper, we present a novel algorithm that solves 0-1 knapsack problem with chance constraint. By analyzing the feasible region of both chance-constrained and risk-averse knapsack problems on variance-mean plane, we prove that there exists a risk-averse knapsack problem such that the optimal solution of chance constrained knapsack problem is also the optimal solution of risk-averse knapsack problem. We use this insight to develop an iterative algorithm where we solve the chance constrained problem by repeatedly solving a sequence of risk-averse knapsack problem. The key aspect of our algorithm is that we maintain a probabilistic guarantee irrespective of the realization of the random variables (the lengths the robots could move). We present simulation results on randomly generated data which show that our algorithm works efficiently. An extended version of this paper with the proofs and more elaborate simulation results is under review at *IEEE International Conference on Robotics and Automation*, 2018.

[1]Fan Yang and [2]Nilanjan Chakraborty are with the Mechanical Engineering Department at SUNY, Stony Brook. Email: fan.yang.3@stonybrook.edu, nilanjan.chakraborty@stonybrook.edu

II. Chance Constrained Knapsack Problem

Let L be the length of the closed curve (or a route) that a team of robots have to cover. We have a collection of heterogeneous robots that have different battery life and they can fly for different lengths. Let ℓ_i be the distance that robot i can fly. Each robot has a different operating and maintenance cost denoted by c_i. The variable ℓ_i is assumed to be a Gaussian random variable with mean μ_i and variance σ_i^2, i.e., $\ell_i \sim \mathcal{N}(\mu_i, \sigma_i^2), i = 1, ..., n$. Our goal is to find a set of robots from the collection of n robots that can cover the total length L with probability p (where $0 \leq p \leq 1$) while minimizing the total cost. Let f_i be an integer variable that takes the value 1 if robot i is part of the team and 0 otherwise. The integer program formulation of our problem is:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} c_i f_i \\ \text{s.t.} \quad & \mathbb{P}\left(\sum_{i=1}^{n} \ell_i f_i \geq L\right) \geq p \\ & f_i \in \{0,1\}, \quad \forall i = 1, \ldots, n \end{aligned} \tag{1}$$

If we relax f_i, the problem in (1) is a second order cone program with integrality gap $\Omega(\sqrt{n})$ [3].

Lemma 1: The CC-KAP problem in (1) with a given probability p is equivalent to the following formulation

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} c_i f_i \\ \text{s.t.} \quad & \sum_{i=1}^{n} \mu_i f_i - C\sqrt{\sum_{i=1}^{n} \sigma_i^2 f_i} \geq L \\ & f_i \in \{0,1\}, \quad \forall i = 1, \ldots, n \end{aligned} \tag{2}$$

where $C = \Phi^{-1}(p)$ is a constant.

In [3], the authors converted the problem in Equation (2) to a parametric linear program and presented an algorithm that for $\epsilon > 0$ gives a $1 - 3\epsilon$ approximate solution with running time $O\left(\frac{1}{\epsilon^2} n^{\frac{1}{\epsilon}}\right)$. We present an alternate parametric formulation, where different choices of the parameter leads to different knapsack problems. In the discussion below we will refer to both (1) and (2) as chance constrained knapsack problem (CC-KAP), which is a chance constrained integer optimization problem and is hard to solve in general. In this paper, instead of solving CC-KAP directly, we show that the solution to CC-KAP can be obtained by solving a number of deterministic knapsack problems (given below), which we call risk-averse knapsack problem (RA-KAP)

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} c_i f_i \\ \text{s.t.} \quad & \sum_{i=1}^{n} \mu_i f_i - \lambda \sum_{i=1}^{n} \sigma_i^2 f_i \geq L' \\ & f_i \in \{0,1\}, \quad \forall i = 1, \ldots, n \end{aligned} \tag{3}$$

Here λ is the risk-averse parameter that performs a weighted combination of the mean and variance of the travel lengths of each robot. The parameter L' is the constraint for the total length in RA-KAP.

III. Geometric Interpretation

In this section, we present a geometric interpretation of the CC-KAP on the variance-mean plane in which the horizontal axis is the variance and the vertical axis is the mean (see Figure 1). The CC-KAP is an integer optimization problems in which any solution is a vector of binary decision variables f_i. Given any particular solution $s = [f_1, ..., f_n]$, we can identify this solution with a point on the variance-mean plane. The y-coordinate of this point is the sum of means for all travel distance of robots chosen in the solution, $\sum_i^n \mu_i f_i$, and the x-coordinate is the sum of variances, $\sum_i^n \sigma_i^2 f_i$. The coordinate of this point related to solution s is denoted by $(\sigma^2(s),\ \mu(s))$. Thus the space of all possible robot teams can be identified with points in the variance-mean plane (however, we do not construct this explicitly because the number of such points will be exponential in the number of robots). Moreover, we can find the feasible region of solution for the CC-KAP based on the chance constraint in Formulation 2. As it is shown in Figure 1, the feasible region for CC-KAP, denoted by $\mathcal{C}$, is the space above the parabola in the first quadrant on variance-mean plane. Since the constraint in the RA-KAP is a linear inequality of σ^2 and μ, the feasible region for RA-KAP, denoted by $\mathcal{R}$, is the space above the line whose slope is equal to risk-averse parameter λ and y-intercept is equal to the length of the route L'. Based on this geometric viewpoint, we present the following lemmas (without proof).

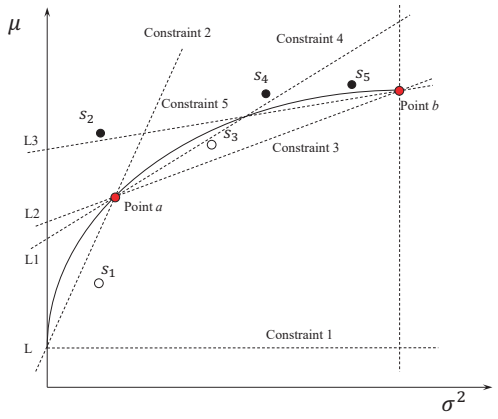


Fig. 1. The geometric interpretation of chance-constrained knapsack problem. Any solution is related to a point on variance-mean plane. The feasible region for CC-KAP is the space above the parabola while the feasible region for RA-KAP with given λ and L is the space above the line whose slope is equal to λ and y-intercept is equal to L.

Lemma 2: The optimal solution of RA-KAP that satisfies the chance constraint provides an upper bound of optimal solution of CC-KAP.

Lemma 3: There exists a RA-KAP, with some choice of λ and L' such that the optimal solution of CC-KAP is also the optimal solution of the RA-KAP.

IV. Algorithm Description

Let the intersection of the feasible region of CC-KAP and the risk-averse problem, RA-KAP, in the first quadrant be $\mathcal{I}$ ($\mathcal{I} = \mathcal{C} \cap \mathcal{R}$). Lemma 2 implies that the optimal solution of RA-KAP that satisfies the chance constraints is also optimal for CC-KAP over the feasible region of CC-KAP restricted to $\mathcal{I}$. In this paper we say that the region $\mathcal{I}$ is "explored". To explore the whole feasible region of CC-KAP, namely $\mathcal{C}$, we solve multiple RA-KAPs, whose optimal solution satisfy the chance constraint and corresponding feasible regions ($\mathcal{R}_j$) cover the feasible region of chance-constrained problem, i.e., $\mathcal{C} \subset \bigcup_{j \in S} \mathcal{R}_j$ where S is the index set of RA-KAPs whose optimal solutions satisfy the chance constraint.

The first step starts with solving RA-KAP with $\lambda = 0$ and $L' = L$. If the optimal solution satisfies the chance constraint, as shown in Figure 1 s_1 is above the parabola. The algorithm terminates because $\mathcal{C} \subset \mathcal{R}$. Otherwise, the algorithm computes the constraint of the RA-KAP for the next iteration by equation $\lambda' = C/\sigma$ where $\sigma = \sqrt{\sum_{i=1}^{n} \sigma_i^2 f_i}$. On the variance-mean plane, we can treat the updating procedure as the constraint line rotating clockwise about y-intercept $(0,\ L)$. The new constraint line will be guaranteed to be located above the previous point. The procedure continues until we obtain a feasible solution of chance-constrained problem or the RA-KAP is not feasible, i.e., $\sum_{i=1}^{n} \mu_i - \lambda \sum_{i=1}^{n} \sigma_i^2 < L$. Now we can conclude that the subset of feasible region $\mathcal{C}$, denoted by $\mathcal{I}_I$ is explored although the risk-averse problem might not be a feasible problem since there is no solution in $\mathcal{R}_I$ and $\mathcal{I}_I \subset \mathcal{R}_I$. Note that all solutions are located to the left of the vertical line $\sigma^2 = \sum_{i=1}^{n} \sigma_i^2$ (the line going through Point b in Figure 1) because $\sum_{i=1}^{n} \sigma_i^2 f_i \leq \sum_{i=1}^{n} \sigma_i^2$ where $f_i \in \{0, 1\}\ \forall i$. Let the subset of feasible region of CC-KAP that is on the right hand side of vertical line be $\mathcal{C}_l$. The remaining feasible region is $\mathcal{C} \backslash (\mathcal{I}_I \cup \mathcal{C}_l)$ denoted by $\mathcal{C}'$.

In the second step, the algorithm computes the intersection of parabola and the constraint line of the last RA-KAP in the first step (Point a in Figure 1) and intersection of parabola and the vertical line (Point b in Figure 1). Then we solve the RA-KAP with new constraint obtained by connecting those two points. Since $\mathcal{C}' \subset \mathcal{I}$, the algorithm terminates if the optimal solution of RA-KAP satisfies the chance constraint, e.g., point s_2 or s_4 in Figure 1. Otherwise, we compute two new constraints for two new RA-KAPs. The new constraints should be selected so that the solutions of RA-KAP is different from previous RA-KAP solutions that do not satisfy chance constraints. Moreover, the feasible regions of the new RA-KAPs, say $\mathcal{I}_a$ and $\mathcal{I}_b$, should cover the feasible regions of the current chance-constrained problem, i.e., $\mathcal{I} \subset (\mathcal{I}_a \cup \mathcal{I}_b)$. For example, in Figure 1, s_3, the solution of RA-KAP with constraint 3 is not feasible to CC-KAP. Therefore we compute new constraints 4 and 5. Our procedure guarantees that $\mathcal{I}_3 \subset (\mathcal{I}_4 \cup \mathcal{I}_5)$ and s_3 is not the solution of RA-KAPs with constraint 4 and constraint 5. If the solution of RA-KAP with any generated constraint j does not satisfy the chance constraint, new constraints will be generated based on constraint j. We then solve the RA-KAP with these constraints. If the solution of RA-KAP with a constraint j satisfies the chance constraint, we obtain the optimal solution in $\mathcal{I}_j$ and therefore there is no need to generate new constraints from j. If the RA-KAP with constraint j is not a feasible problem, we do not need to generate new constraints since there is no solution in $\mathcal{I}_j$. The second step terminates when there is no new constraint generated. Thus, we explore the whole feasible region of CC-KAP because $\mathcal{C}' \subset \bigcup_{j \in S} \mathcal{R}_j$ and $\mathcal{C} = \mathcal{C}_I \cup \mathcal{C}_l \cup \mathcal{C}'$. The optimal solution is the one with the smallest objective value among all feasible solutions of chance-constrained problem computed by solving risk-averse problems.

We claim that our algorithm stops in finite number of iterations. For the first step, λ increases till the solution of RA-KAP satisfy the chance constraint. In the worst case, λ will keep increasing until it exceeds a finite bound of λ for which the RA-KAP is not feasible. For the second step, the new constraints will be generated if the solution of RA-KAP is not feasible to chance constraint. The number of solutions is finite and the algorithm prevents obtaining the previous solutions that are not feasible to chance constraint. Therefore in the worst case, the algorithm finds all solutions that are not feasible with a finite number of iterations. In the next section, we present empirical evidence that the above algorithm terminates in a constant number of iterations irrespective of the number of robots.

V. Simulation Results

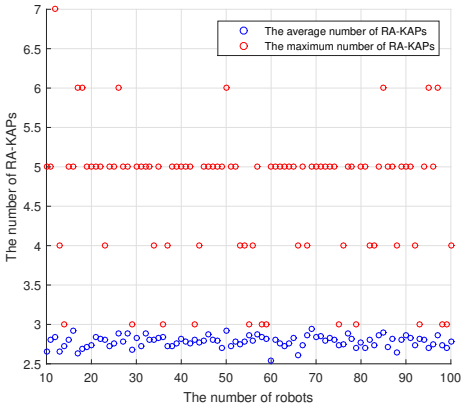


Fig. 2. The total number of RA-KAPs required for solving CC-KAP. The length of the route is 10000 meters and the number of robots vary from 10 to 100. Each data point is obtained from 100 simulations with randomly generated mean and variance of travel distance of each robot

We present simulation studies to understand the scalability of our algorithm as the number of robots increase and as the knowledge about the distance the robots can travel become more uncertain (i.e., the variance increases). To understand the effects of parameters such as the number of robots and the variance of travel distance of robots, we generated different scenarios based on randomly generated parameter values. We first present results for simulations in which the mean and variances of travel distance of robots are randomly generated

and the number of robots is varied methodically. Figure 2 and Figure 3 show the performance of our algorithm with the different number of robots and uncertainty in the travel length of robots. The results indicate that the number of robots does not have a significant influence on the speed of our algorithm and the number of calls to RA-KAPs is nearly a constant. In

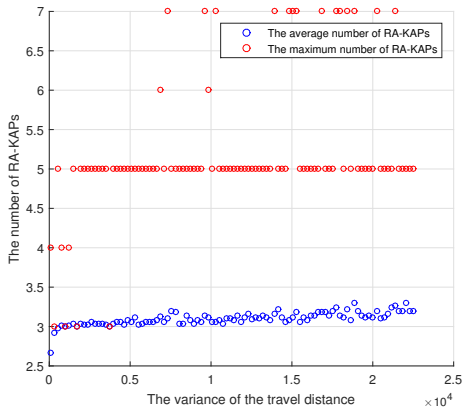


Fig. 3. The total number of RA-KAPs that are used for solving CC-KAP. The number of robots is 100, the length of the route is 50000 meters, and the variance of travel distance vary from 100 to 22500. Each data point is obtained from 100 simulations with randomly generated mean of travel distance of each robot.

the first simulation, we test the effect of number of robots on the number of RA-KAP to be solved, which influences the algorithm performance. The algorithm to solve the RA-KAP is dynamic programming that solves the knapsack problem optimally in pseudo-polynomial time $\mathcal{O}(n^2P)$ where n refers to the number of robots and P refers to the largest cost among all robots [12]. The means and variances of the travel lengths for each robot are generated independently from a uniform distribution $\mu_i \sim \mathcal{U}(1000, 3000)$ and $\sigma_i^2 \sim \mathcal{U}(10000, 12500)$. The length of the curve, L, is 10000 meters and $p = 0.99$. The operation and maintenance cost for each robot is distributed randomly in uniform distribution from 50 to 150, i.e., $c_i \sim \mathcal{U}(50, 150)$. Set $\epsilon = 1 \times 10^{-7}$. We count the number of RA-KAPs for solving CC-KAP when number of robots is equal to $10, 11, \ldots, 100$. For each case with a given number of robots, we generate the means and variances randomly for 100 times.

Figure 2 shows the performance of our algorithm with different number of robots. The results show that the number of calls to RA-KAPs is nearly a constant irrespective of the number of robots. The blue dots represent the average number of RA-KAPs required to solve CC-KAP while the red dots represent the maximum number of calls to RA-KAPs from 100 simulations. The average numbers of RA-KAP solved is almost constant (between 2.5 to 3) irrespective of the number of robots. In Figure 2, the maximum numbers of RA-KAPs solved are between 3 and 7. We observe that maximum number of deterministic knapsack problems solved is 7 which is a small value for application in practice.

In the second simulation, we obtain the effect of the uncertainty of travel distance of robot on the performance of our algorithm by counting the number of calls to RA-KAPs and the actual running time for our algorithm solving CC-KAP with variance equal to $100, 324, 548, \ldots, 22500$. For each case, we generate mean of travel distance of robot randomly based on $\mathcal{U}(1000, 3000)$ for 100 times. The number of robots is 100 and the length of the route is 50000 meters for all scenarios in this simulation. The other parameters such as the cost, probability and ϵ are same as the parameters in the first simulation. Figure 3 shows the average number of calls to RA-KAPs is practically constant as the variance of travel distance increases. The maximum numbers of calls are within the range from 3 to 7.

VI. Conclusion

We presented a novel deterministic algorithm for chance-constrained knapsack problem with application in multi-robot routing with the uncertain travel distances (weights). The key idea in our approach is to convert CC-KAP to a deterministic discrete optimization problem on the variance-mean plane, where each point on the plane can be identified with an assignment of items to the knapsack. By exploiting the geometry of the non-convex feasible region of the CC-KAP in the variance-mean plane, we showed that CC-KAP can be solved optimally by solving a sequence of deterministic knapsack problems (called risk-averse knapsack problem). We demonstrated empirically that our algorithm is quite efficient in practice. Future work includes theoretical complexity bounds of our algorithm.

References

[1] R. L. Carraway, R. L. Schmidt, and L. R. Weatherford. An algorithm for maximizing target achievement in the stochastic knapsack problem with normal returns. *Naval Research Logistics (NRL)*, 40(2):161–173, 1993.

[2] B. C. Dean, M. X. Goemans, and J. Vondrk. Approximating the stochastic knapsack problem: The benefit of adaptivity. *Mathematics of Operations Research*, 33(4):945–964, 2008.

[3] V. Goyal and R. Ravi. A ptas for the chance-constrained knapsack problem with random item sizes. *Operations Research Letters*, 38(3):161 – 164, 2010.

[4] M. I. Henig. Risk criteria in a stochastic knapsack problem. *Operations Research*, 38(5):820–825, 1990.

[5] O. Klopfenstein and D. Nace. A robust approach to the chance-constrained knapsack problem. *Operations Research Letters*, 36(5):628 – 632, 2008.

[6] L. Luo, N. Chakraborty, and K. P. Sycara. Distributed algorithm design for multi-robot generalized task assignment problem. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, November 3-7, 2013*, pages 4765–4771, 2013.

[7] S. Martello, D. Pisinger, and P. Toth. Dynamic programming and strong bounds for the 0-1 knapsack problem. *Manage. Sci.*, 45(3):414–424, Mar. 1999.

[8] S. Martello and P. Toth. A mixture of dynamic programming and branch-and-bound for the subset-sum problem. *Management Science*, 30(6):765–771, 1984.

[9] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, Inc., New York, NY, USA, 1990.

[10] V. Poirriez, N. Yanev, and R. Andonov. A hybrid algorithm for the unbounded knapsack problem. *Discrete Optimization*, 6(1):110 – 124, 2009.

[11] M. Sniedovich. Preference order stochastic knapsack problems: Methodological issues. *The Journal of the Operational Research Society*, 31(11):1025–1032, 1980.

[12] V. V. Vazirani. *Approximation Algorithms*. Springer-Verlag New York, Inc., New York, NY, USA, 2001.

Freeze Tag Awakening in 2D is NP-hard*

Zachary Abel† Hugo A. Akitaya‡ Jingjin Yu§

Arkin et al. [2] proposed a scheduling problem called *freeze tag awakening* motivated by problems involving robotic swarms. The input consists of n mobile point robots in Euclidean space labeled *awake* or *frozen*. An awakened robot may travel at unit speed in any direction. A frozen robot is awakened if touched by an awakened robot. Initially a single robot is awake and all other robots are frozen. The freeze tag awakening problem asks for *a minimum time in which all robots can be woken up*. If no two robots initially lie on the same place, the problem can also be phrased as obtaining the binary tree rooted at a specified point that minimizes the longest path. Although a PTAS for 2D is shown in [2], the computational complexity of this problem remained open as outlined in [2] and appears as problem 35 on The Open Problem Project[1]. The decision version of the problem may be stated as follows.

Problem 1 (FreezeTag)**.** *Given n mobile robots in the Euclidean d-space with exactly one robot initially awake, is there an awakening schedule such that all robots can be awakened in time no more than some $T > 0$?*

A paper in Fall Workshop 2016 [1] showed that FreezeTag in 1D has a trivial algorithm and claimed that the problem is NP-hard in 2D. However, their proof has a mistake as shown next. They reduce from Monotone-3SAT which is NP-complete [3]. An instance of this problem is given by a set of boolean *variables* $\{x_1, \ldots, x_n\}$ and *clauses* $\{c_1, \ldots, c_m\}$ in which each clause is formed by three literals and each literal is a copy or a negated copy of the boolean value of one of the variables. Clauses are either *positive* when all its literals are not negated or *negative* when all are negated. Monotone-3SAT asks if there is an assignment from $\{\texttt{true}, \texttt{false}\}$ to the variables such that every clause contains at least one `true` literal. Figure 1 shows their variable gadget where dots represent the initial position of a unique robot except for s_1 that contains two robots, one of which is awake. The L_1-distance between the points $g_i^+, i \in \{1, \ldots, n\}$ and s_1 is the same. Their variable gadget places robots on the line segments $s_i^+ g_i^+$. Their proof claims that the only robot that could reach g_i^+ in a positive solution is s_i^+, however this is not true as s_i^+ could switch roles with a robot on $s_j^+ g_j^+$ for some $j < i$ possibly resulting in a better solution.

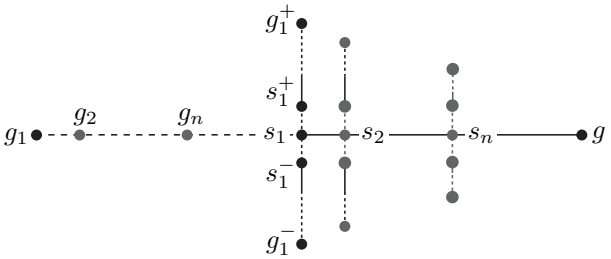


Figure 1: The variable gadget for all variables of the reduction in [1].

We show that the problem is indeed NP-hard in 2D Euclidean space by reducing from Monotone-3SAT building on ideas in [1]. We leave open whether FreezeTag is also NP-hard in 2D for other L_p spaces. In the proof, we use $||.||$ to denote Euclidean lengths.

*Research on this paper was supported in part by the NSF awards CCF-1422311 and CCF-1423615, and the Science Without Borders scholarship program.

†Department of Computer Science, MIT, Cambridge, MA

‡Department of Computer Science, Tufts University, Medford, MA

§Department of Computer Science, Rutgers University, Piscataway, NJ

[1] `http://cs.smith.edu/~orourke/TOPP/P35.html`

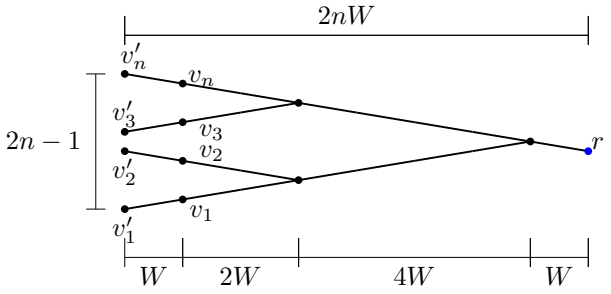


Figure 2: The setup gadget for an instance with four variables.

Theorem 1. FREEZETAG *in 2D is NP-complete.*

Proof. As shown in in [1], the problem is in NP. We reduce from MONOTONE-3SAT as in [1]. We assume that n is a power of 2, or else add variables that are not used in any clause until n is a power of 2. This at most doubles the number of variables. Let $T = 8n^2(m+1)$ and $\varepsilon = \sqrt{T^2+4} - T$. We build an instance of FREEZETAG using the following gadgets. To aid in the description of the gadgets, we also define $T' = 4n(m+1)$ and $W = \sqrt{(T')^2 - 1/4}$.

Setup gadget. The initially awake robot is placed at $r = (W(2n-1), n+\frac{1}{2})$ (shown as a blue dot in Fig 2). For each variable x_i, create a robot at $v_i = (0, 2(i-1))$. Create a balanced binary tree rooted at $(2W(n-1), n-1)$ and whose leaves are the points v_i. Place each internal node of such tree so that the vector from it to its right (resp., left) child forms an angle of $\pi - \arcsin(\frac{1}{2T})$ (resp., $-\pi + \arcsin(\frac{1}{2T})$) with vector $(1,0)$. Place a robot at each node of the tree. Create a robot at $v'_i = (-W, 2(i-1) - \frac{1}{2})$ (resp., $(-W, 2(i-1) + \frac{1}{2})$) for odd (resp., even) i.

Variable gadget. For each x_i, we place 5 robots at points v_i^+, v_i^-, q_i^+, q_i^-, and p_i as shown in Fig. 3. The first four are respectively located $\frac{\varepsilon}{2}$ to the right, $\frac{\varepsilon}{2}$ to the left, $T' - \varepsilon$ to the right, and $T' - \varepsilon$ to the left of v_i. The point p_i is located above v_i so that the distance between it and v_i^+ (or v_i^-) is $T' - \frac{3}{2}\varepsilon$. Intuitively, the order in which v_i^+ and v_i^- are awakened encodes the truth assignment of the 3SAT instance. They will then respectively target q_i^+ and q_i^- while the robot initially in v_i targets p_i.

Clause gadget. Fig. 4 shows the gadget for when the ℓ-th clause $(x_i \vee x_j \vee x_k)$ is positive. For negative clauses, reflect the construction through the y-axis. Assume that $i < j < k$, placing robots at $s_{i,\ell}$, $s_{j,\ell}$, and $s_{k,\ell}$ as indicated in Fig. 4. The x-coordinate of $s_{i,\ell}$, $s_{j,\ell}$ and $s_{k,\ell}$ are respectively $4n\ell + k - j$, $4n\ell$ and $4n\ell + 2(k-j)$. They are respectively placed in the line segments $v_i q_i^+$, $v_j q_j^+$ and $v_k q_k^+$. Robots at $s'_{w,\ell}$ are above $s_{w,\ell}$ and $||s_{w,\ell} s'_{w,\ell}|| = ||s_{w,\ell} q_w^+||$ for $w \in \{i,j,k\}$. Place a robot at c_ℓ located ε above $s'_{i,\ell}$. Place a robot at $s''_{w,\ell}$ under $s'_{w,\ell}$ so that $||s''_{w,\ell}(s'_{w,\ell} + (0,\varepsilon))|| = ||s''_{w,\ell} c_\ell||$, for $w \in \{j,k\}$. Intuitively, $s_{i,\ell}$, $s_{j,\ell}$ and $s_{k,\ell}$ will respectively target $s'_{i,\ell}$, $s'_{j,\ell}$ and $s'_{k,\ell}$ and c_ℓ can only be awaken at time T if a robot in $\{v_i^+, v_j^+, v_k^+\}$ was awakened before its reflected counterpart.

Correctness. Assume that the 3SAT instance has a positive solution. We show that all robots can be awakened within time T. We say that a robot *targets* a point if it is scheduled to visit the point. Send the initially awake robot at r to v'_n. Whenever a robot at a node of the tree in the setup gadget is awakened, send it to the unique v'_i not yet targeted located $T - t$ away, where t is the current time. At time $T - T'$ all v_i will be awakened and every robot in the setup gadget is either awake or targeted. If x_i is assigned `true`, send v_i to v_i^+, v_i^-, and then to p_i. Reverse the order between v_i^+ and v_i^- otherwise. By construction, p_i will be awakened exactly at time T. Send each v_i^+ (resp., v_i^-) to q_i^+ (resp., q_i^-) as soon as they are awake. As soon as each $s_{i,\ell}$ is awake, send it to $s'_{i,\ell}$. For each

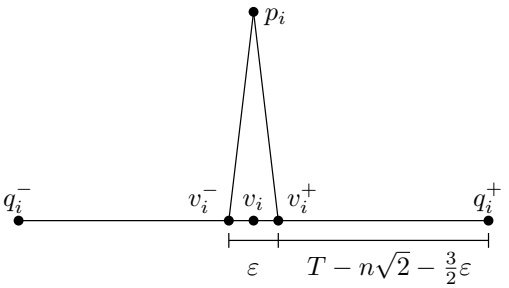


Figure 3: The variable gadget for the i-th variable.

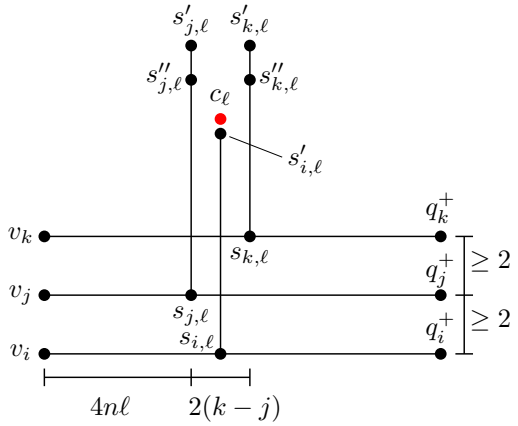


Figure 4: The clause gadget for the ℓ-th clause.

positive clause $(x_i \vee x_j \vee x_k)$, there must exist at least one robot in $\{v_i^+, v_j^+, v_k^+\}$ awakened at time $T - T' + \frac{\varepsilon}{2}$. If such robot v_w^+ is in $\{v_j^+, v_k^+\}$, send $s''_{w,\ell}$ to c_ℓ as soon as its awake. Else, send $s'_{i,\ell}$ to c_ℓ. Similar arguments apply to negative clauses. By construction, all robots will be awake at time T.

Now assume that the produced FreezeTag instance is positive. Since $||v'_n r|| = T$, the robot at r must move on a straight line towards v'_n for the whole duration of the awakening schedule. A similar argument can be used to match each node in the binary tree of the setup gadget to a v'_i. Then, every v_i will be awakened at time $T - T'$ and the robots targeting v'_i cannot interact with any other robot. At time $T - T'$ there are exactly four robots in the ε neighborhood of v_n, one of which is targeting v'_n. The three remaining robots are the only ones that can wake up p_n, q_n^- and q_n^+. Hence, the robot in v_n must wake up robots in v_n^+ and v_n^- in some order. Assume that the one at v_n^+ is awakened first. Then v_n^+ must target q_n^+, because by the time the robot in v_n^- is awakened, the robots at that position cannot reach q_n^+ in time. Then, at time $T - T' + \frac{3}{2}\varepsilon$, the two robots at v_n^- must move on straight lines to q_n^- and p_n for the remaining time. Notice that the robot initially at v_n^+ has ε spare time while targeting q_n^+ and potentially could reach another robot in the ellipse with foci at v_n^+ and q_n^+ and major axis $||v_n^+ q_n^+|| + \varepsilon$. By construction, every such robot lies in the line segment $v_n^+ q_n^+$ and, therefore, there exist a solution where the robot at v_n^+ goes to q_n^+ on a straight line as soon as it's awakened. The symmetric argument applies when v_n^- is awakened first. Recursively applying the same argument for the robots in the ε-neighborhood of $v_{n-1}, v_{n-2}, \ldots$ at time $T - T'$ we can determine the behavior of all such robots in the solution. Consider the first clause $(x_i \vee x_j \vee x_k)$, $i < j < k$, that is positive without loss of generality. The argument above implies that the robot at $s_{j,1}$ will be awakened at time $T - T' + 4n$ or $T - T' + 4n + \varepsilon$. The robot at $s'_{j,1}$ can only be reached in time by the robot awakened at $s_{j,1}$. Even if such robot awakens at time $T - T' + 4n$, the only other robot that it can reach is $s''_{j,1}$, since it must target $s'_{j,1}$. Then, there is a solution where it goes to $s'_{j,1}$ on a straight line as soon as it is awakened. The robot at $s''_{j,1}$ is awakened at time $T - ||s'_{j,1} s''_{j,1}||$ or $T - ||s'_{j,1} s''_{j,1}|| - \varepsilon$. In both cases, it cannot reach $s'_{i,1}$, $s'_{k,1}$ by construction, or any other robot in a different clause gadget because $||s'_{j,1} s''_{j,1}|| < 2n$. Then, by a similar argument, the robot at $s_{k,1}$ and $s_{i,1}$ must respectively move in a straight line to $s'_{k,1}$ and $s'_{i,1}$. Hence, this will be the behavior of all robots at positions of the form $s_{i,\ell}$. Assume for contradiction that, for the ℓ-th (without loss of generality positive) clause $(x_i \vee x_j \vee x_k)$, v_i^+, v_j^+, v_k^+ are awakened at time $T - T' + \frac{3}{2}\varepsilon$. Then no robot can reach c_ℓ within time T. Therefore, we can use the order of awakening between v_i^+ and v_i^-, $i \in \{1, \ldots, n\}$ to obtain a truth assignment that satisfies the original 3SAT instance. □

References

[1] Hugo A Akitaya and Jingjin Yu. Freeze tag awakening in euclidean spaces. In *Abstracts of the 26th Fall Workshop on Computational Geometry*, 2016.

[2] Esther M Arkin, Michael A Bender, Sándor P Fekete, Joseph SB Mitchell, and Martin Skutella. The freeze-tag problem: how to wake up a swarm of robots. *Algorithmica*, 46(2):193–221, 2006.

[3] E. M. Gold. Complexity of automaton identification from given data. *Information and control*, 37(3):302–320, 1978.

Freeze Tag is Hard in 3D

Erik D. Demaine* Mikhail Rudoy*†

Abstract

In the freeze tag problem, we start with n robots at specified locations, only one of which is initially "active". All active robots can move at unit speed, and upon reaching another robot's location, can activate that robot. The goal is to activate all robots in the minimum possible time. This problem was introduced by Arkin et al. in 2002, who developed approximation algorithms, but left hardness as an open problem. At FWCG 2016, Akitaya and Yu proved NP-hardness in Euclidean 3-space. Here we give a very simple proof of NP-hardness in Euclidean ℓ_p space for any $p > 1$.

1 Introduction

The freeze tag problem was introduced by Arkin et al. [2] at SODA 2002 to model automatic awakening of a swarm of robots by manually turning on just one robot. The input consists of a list of robot locations, exactly one of which is indicated to be initially "active" or "on". Phrased as a decision problem, we are also given a time limit. Robots that are active can move at unit speed; inactive robots cannot move, but are activated when they are in the same location as an active robot. The goal is to decide whether all of the robots can be activated within the given time limit.

The freeze tag problem has many variants depending on the space of possible robot locations and on the choice of metric on this space (defining "unit speed"). For example, Arkin et al. [2] prove the problem NP-hard and give an $O(1)$-approximation algorithm for star graphs, and more generally, for graphs of bounded degree. They also give an efficient PTAS for any ℓ_p space of fixed dimension, but leave open NP-hardness. This problem was tackled by Akitaya and Yu [1] at FWCG 2016, who claimed strong NP-hardness for Euclidean d-space for any $d \geq 2$. Unfortunately, it was later discovered that their proof is incorrect for $d = 2$.[1]

In this paper, we prove that the freeze tag problem is strongly NP-hard in 3-dimensional ℓ_p space, for any $p > 1$. (Recall that the ℓ_p norm defines the distance between two points (x_1, y_1, z_1) and (x_2, y_2, z_2) as $\sqrt[p]{|x_1 - x_2|^p + |y_1 - y_2|^p + |z_1 - z_2|^p}$.) Our result generalizes Akitaya and Yu's previous result for $p = 2$. Our proof is also very simple.

2 Reduction

To prove NP-hardness of freeze tag in 3-dimensional ℓ_p space for any $p > 1$, we reduce from dominating set in square grid graphs, which was shown NP-hard in [3]. Recall that a *square grid graph* is a finite induced subgraph of the integer lattice, and that the decision form of dominating set asks whether there is a choice of k or fewer vertices such that every vertex is either chosen or adjacent to a chosen vertex. We assume without loss of generality that the input graph is connected.

Given a connected square grid graph G and integer k, we can embed G in the plane such that (1) some vertex is at the origin, (2) each vertex has integer coordinates, and (3) two vertices are adjacent if and only if they have unit distance.

*MIT Computer Science and Artificial Intelligence Laboratory, 32 Vassar St., Cambridge, MA 02139, USA, edemaine@mit.edu, mrudoy@gmail.com

†Now at Google Inc.

[1] Personal communication with Hugo Akitaya, Sept. 2017.

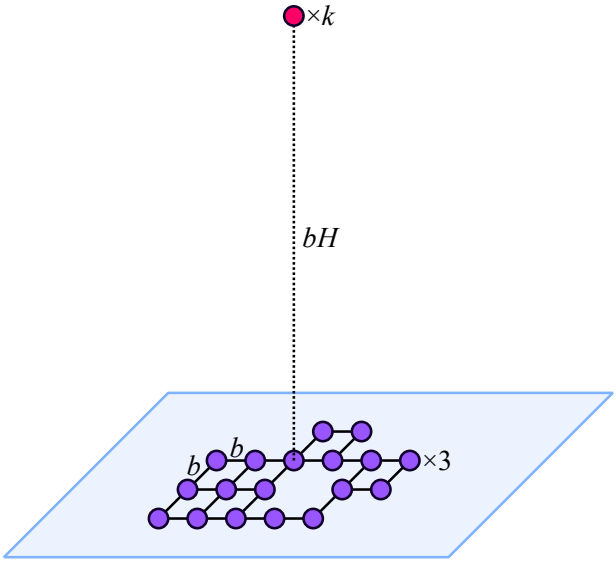


Figure 1: Reduction from dominating set in grid graphs: k active robots bH above 3 robots at each vertex of the grid graph scaled by b.

We refer to each vertex by its coordinates in this embedding. Let n be the number of vertices in G.

We place $3n+k$ robots as follows; see Figure 1. For every vertex $(x, y) \in G$, we place three inactive robots at location $(bx, by, 0)$ for a scale factor b defined below. We also add k robots, one active and $k-1$ inactive, at location $(0, 0, bH)$, where H is a large number defined below. All k of these robots will be active at time 0. Our choice of bH will result in the distance between these k robots and any other robot being at least bH and at most $bH+1$. The time limit T which we give for the freeze tag instance is $b(H+1)+1$.

Choice of b. For $p > 1$, we can define positive integers b and c (based only on p) such that $0 < \frac{1}{b} < 2^{1/p}-1$ and $1+\frac{1}{c} \le p$. Also define $p' = 1+\frac{1}{c}$.

Choice of H. No matter how we choose H, the distance between location $(0, 0, bH)$ and location $(bx, by, 0)$ (for $(x, y) \in G$) is at least bH. Next we describe how to choose H so that these distances are also at most $bH+1$. Specifically, we show that $H = n(bn)^c$ achieves this result.

The distance $d_{x,y}$ between location $(0, 0, bH)$ and location $(bx, by, 0)$ (for $(x, y) \in G$) is exactly $b\sqrt[p]{H^p + |x|^p + |y|^p}$. Because there are n vertices in G, G is connected, and $(0, 0) \in G$, we know that $|x| + |y| \le n$. By our assumption that $p > 1$, $|x|^p + |y|^p \le (|x| + |y|)^p \le n^p$. Thus $d_{x,y} \le b\sqrt[p]{H^p + n^p}$. It is well known that the norm of a fixed vector in ℓ_i space does not grow as i grows; since by definition $p' < p$, we then have that $\sqrt[p]{H^p + n^p} \le \sqrt[p']{H^{p'} + n^{p'}}$ and therefore that $d_{x,y} \le b\sqrt[p]{H^p + n^p} \le b\sqrt[p']{H^{p'} + n^{p'}}$. We can rewrite the latter as $bH\sqrt[p']{1 + (\frac{n}{H})^{p'}}$, or equivalently $bH + bH\left[\sqrt[p']{1 + (\frac{n}{H})^{p'}} - 1\right]$. But by Bernoulli's inequality, $\sqrt[p']{1 + (\frac{n}{H})^{p'}} \le 1 + \frac{1}{p'}(\frac{n}{H})^{p'}$, so $d_{x,y}$ is at most $bH + bH\left[1 + \frac{1}{p'}(\frac{n}{H})^{p'} - 1\right] = bH + \frac{bH}{p'}(\frac{n}{H})^{p'} = bH + \frac{bn}{p'}(\frac{n}{H})^{p'-1}$. Because $p' > 1$, this is less than $bH + bn(\frac{n}{H})^{p'-1}$. Plugging in our definition of $p' = 1 + \frac{1}{c}$, we obtain $bH + bn(\frac{n}{H})^{1/c}$. Then plugging in our definition of $H = n(bn)^c$, we obtain $bH + bn(\frac{n}{H})^{1/c} = bH + bn(\frac{n}{n(bn)^c})^{1/c} = bH + bn(\frac{1}{(bn)^c})^{1/c} = bH + bn(\frac{1}{bn}) = bH + 1$. Therefore $d_{x,y} \le bH + 1$ as desired.

3 Correctness

The reduction runs in polynomial time because b and c are constant and $k \le n$. Furthermore, every number produced by the reduction is either 0, bH, $b(H+1)+1$, bx, or by (where x and y are coordinates of vertices in the input). These numbers have magnitude polynomial in the size of the input, so the reduction suffices for strong NP-hardness. It remains to show that the reduction is answer preserving.

Dominating set $\implies$ freeze tag. Suppose that the input grid graph G has a dominating set S with $|S| \le k$. Without loss of generality, assume $|S| = k$ (as we can always add vertices to a dominating set). From this, we can construct a solution to the freeze tag instance produced by the reduction. This solution proceeds in two stages.

The first stage starts at time zero and lasts for $bH + 1$ units of time. At time zero, the k

robots at position $(0, 0, bH)$ are active; we send one robot to each of the locations corresponding to the vertices in S. That is, if $(x, y) \in S$, then one of the initially active robots will move from its initial location of $(0, 0, bH)$ to location $(bx, by, 0)$. This distance is at least bH and at most $bH + 1$ by our choice of H, so the robots can achieve this within the time allocated for the first stage.

The second stage lasts for b units of time. After the first stage, every robot is at a location corresponding to a vertex of G. In particular, each location corresponding to a vertex in S contains four active robots (the one initially active robot that got there from $(0, 0, bH)$ and the three initially inactive robots that started at the location) while each location corresponding to a vertex not in S contains three inactive robots. From each location $(bx, by, 0)$ corresponding to vertex $(x, y) \in S$, we send the four active robots at that location in the $+x$, $+y$, $-x$, and $-y$ directions for the duration of the second stage. As a result, these four robots reach locations $(b(x \pm 1), by, 0)$ and $(bx, b(y \pm 1), 0)$. These four final locations correspond to vertices $(x \pm 1, y)$ and $(x, y \pm 1)$ if they exist. In other words, these locations are exactly the four locations that can be associated with neighbors of (x, y). Thus, the robots in the locations associated with neighbors of (x, y) are activated by the end of the second stage. Since each vertex of G is either in S or adjacent to a vertex in S, all of the initially inactive robots are activated by the end of the second stage. Including the first stage, this solution takes time $bH + 1 + b$, which is exactly the time limit T.

Freeze tag $\implies$ dominating set. Suppose, on the other hand, that the freeze tag instance produced by the reduction is solvable: there exists a scheduling of destinations such that every robot is activated in at most $bH+1+b$ time. Consider the k robots at location $(x, y, z) = (0, 0, h)$. For each of these k robots, we can identify the first location with other robots that is visited by this robot. This location will be of the form $(bx, by, 0)$ where (x, y) is a vertex in G. Reaching this location requires at least bH time by our choice of H.

The remaining time for the robots at $(bx, by, 0)$ is then at most $1 + b$. Reaching location $(b(x + \Delta_x), b(y + \Delta_y), 0)$ associated with some other vertex $(x + \Delta_x, y + \Delta_y)$ requires time $b(|\Delta_x|^p + |\Delta_y|^p)^{1/p}$. Unless (Δ_x, Δ_y) is $(1, 0)$, $(0, 1)$, $(-1, 0)$, or $(0, -1)$, this value is at least $b2^{1/p}$. By definition of b, we know that $\frac{1}{b} < 2^{1/p} - 1$, or in other words that $b2^{1/p} > 1 + b$. In other words, the location $(b(x+\Delta_x), b(y+\Delta_y), 0)$ cannot be reached in the remaining time. Thus, the only locations reachable from $(bx, by, 0)$ are the locations associated to vertices adjacent to vertex (x, y).

We have shown that each of the k initially active robots, together with all the robots it activates, can reach only the locations corresponding to a single vertex and its neighbors. Since every robot is activated within the time limit, the location corresponding to every vertex must be reached by some robot, and so there exists a choice of k vertices (the first vertices visited by the k initially active robots) such that every vertex is either chosen or adjacent to a chosen vertex. In other words, there exists a dominating set of size k in G.

4 Open Problems

The obvious remaining open question is whether freeze tag is also NP-hard in the 2-dimensional ℓ_p metric for any p.

References

[1] Hugo A. Akitaya and Jingjin Yu. Freeze tag awakening in Euclidean spaces. In *Abstracts from the 26th Fall Workshop on Computational Geometry*, October 2016. http://matthewpjohnson.org/fwcg2016/FWCG_2016_paper_6.pdf.

[2] Esther M. Arkin, Michael A. Bender, Sándor P. Fekete, Joseph S. B. Mitchell, and Martin Skutella. The freeze-tag problem: how to wake up a swarm of robots. *Algorithmica*, 46(2):193–221, 2006.

[3] Brent N. Clark, Charles J. Colbourn, and David S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86(1):165–177, 1990.

Easier Hardness for 3D Freeze-Tag

Matthew P. Johnson*

1. INTRODUCTION

In the Freeze-Tag problem [2], we are given a set of n robots lying at points in a space, all but one of which is initially *frozen*. When a robot is not frozen, or *awake*, it can move within the space at constant speed, and when it encounters a frozen robot, it *unfreezes* it. The task to design a schedule for the robots to move about, unfreezing one another, with the objective of minimizing the makespan, i.e., the time at which the last robot is unfrozen. A proof of Freeze-Tag's NP-hardness in 3D Euclidean space was implicit in [1], via a relatively intricate reduction from Monotone 3SAT.

In this note, we give what we feel is a much simpler proof of this result, reducing from Hamiltonian Path in grid graphs, which is known to be NP-Complete, even in the special case where the graph has exactly two degree-1 nodes [3].

A grid graph is specified by a finite set of nodes lying at integer coordinates in the plane. The graph is induced by these nodes in the sense that any two nodes u, v whose distance equals 1 (i.e., either their x coordinates *or* their y coordinates differ by 1) will have an edge. Without loss of generality, the n points will lie within $\{1, ..., n\} \times \{1, ..., n\}$.

2. CONSTRUCTION

Given a grid graph, we construct the 3D Freeze-Tag instance as follows (see Fig. 1(a)). We embed the grid graph in the $z = 0$ plane in $\mathbf{R}^3$, placing a robot at each of its nodes, which we call *ground bots*. The location of the initially awake bot is one of the two degree-1 nodes, chosen arbitrarily, say g_0.

Then we create an additional $n-1$ robots, called *sky bots*, located at heights far above the $z = 0$ plane. For each $t = 1, ..., n-1$, sky bot s_t is at height $h_t = T - \epsilon - t$. The sky bots' integer xy coordinates are all distinct, chosen arbitrarily within $\{1, ..., n\} \times \{1, ..., n\}$. We set the deadline $T = 3n^4 + \epsilon$, where $\epsilon = \frac{1}{4n^2}$. Then it can be calculated that:

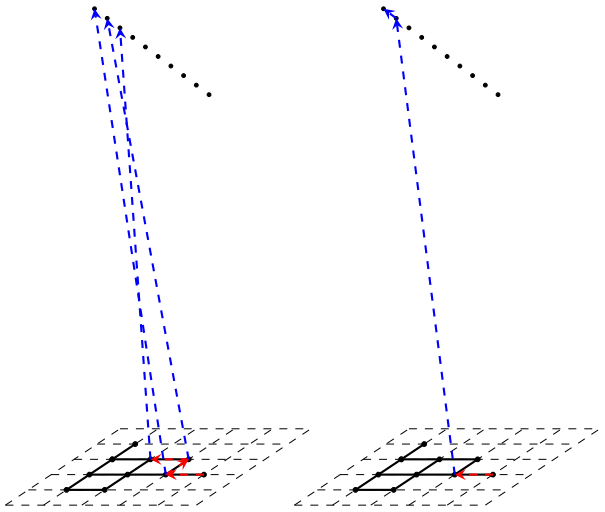
CLAIM 1. *The distance $d(g_i, s_j)$ from any ground bot g_i to any sky bot s_j of height h_j satisfies $h_j \leq d(g_i, s_j) < h_j + \epsilon$.*

The strategy of the design is to ensure that in a feasible schedule (i.e., one meeting deadline T), each initially frozen ground bot will awaken exactly one other robot, a sky bot.

3. PROOF

Fix a schedule, and rename the initially frozen ground bots $g_1, ..., g_{n-1}$ based on the order in which they are awoken.

If the schedule is feasible, clearly the first bot awoken by g_0 must be another ground bot (g_1), and $d(g_0, g_1) \geq 1$. But by time 1, we have only $T - 1 = h_1 + \epsilon$ time left to send



(a) Feasible schedule. (b) Infeasible schedule.

Figure 1: In (a), $g_0 \to g_1 \to g_2$ shown in red and $g_1 \to s_1$, $g_2 \to s_2$, and $g_3 \to s_3$ in blue; in (b), $g_0 \to g_1$ shown in red and $g_1 \to s_2 \to s_1$ shown in blue.

one of the two awake bots (g_1, wlog) to reach sky bot s_1. Because $h_1 \leq d(g_1, s_1) < h_1 + \epsilon$, and because every frozen ground bot is distance at least 1 away from g_1, g_1 will not have time to awaken any other ground bots. Moreover, g_1 must be a neighbor of g_0 because g_0's distance to any non-neighbor would be at least $\sqrt{2} - 1$ greater. Now, imagine g_1 trying to awaken another sky bot, say s_i, before awakening s_1, and consider g_1's journey $s_i \to s_1$ (see Fig. 1(b)). Their xy coordinates differ by at least 1, and the difference Δ_z between their heights is at most $n-2$, so it can be calculated that $d(s_i, s_1) > \Delta_z + \frac{1}{3n}$. This implies that the journey $g_1 \to s_i \to s_1$ would take time at least $h_1 + \frac{1}{3n} > h_1 + \epsilon$.

We have shown that in a feasible schedule, g_0 will first wake up one of its neighbors g_1, and then g_1 will travel to awaken s_1 (which then does not have time to wake any others up). By repeated application of this argument, it follows that g_0 must traverse a total of $n-1$ edges of the grid graph, each time visiting a new node and awakening another ground bot (which must leave immediately to awaken a single sky bot), i.e., g_0 must traverse a Hamiltonian path.

Conversely, it is clear that if the grid graph admits a Hamiltonian path, then there exists a feasible schedule.

Hence we conclude:

THEOREM 3.1. *Freeze-Tag in 3D is NP-hard.*

4. REFERENCES

[1] H. Akitaya and J. Yu. Freeze tag awakening in euclidean spaces. In *FWCG*, 2016.
[2] E. M. Arkin, M. A. Bender, S. P. Fekete, J. S. Mitchell, and M. Skutella. The freeze-tag problem: how to wake up a swarm of robots. *Algorithmica*, 46(2):193–221, 2006.
[3] A. Itai, C. H. Papadimitriou, and J. L. Szwarcfiter. Hamilton paths in grid graphs. *SIAM Journal on Computing*, 11(4):676–686, 1982.

*Lehman College and The Graduate Center, CUNY. This work was supported by NSF-INSPIRE award #1547205 and a CUNY Junior Faculty Research Award (J-FRASE) funded by the Sloan Foundation.

Declarative vs Rule-based Control for Flocking Dynamics

Usama Mehmood
Department of Computer Science, Stony Brook University, USA

Nicola Paoletti
Department of Computer Science, Stony Brook University, USA

Dung Phan
Department of Computer Science, Stony Brook University, USA

Radu Grosu
Cyber-Physical Systems Group, Technische Universitat Wien, Austria

Shan Lin
Department of Electrical and Computer Engineering, Stony Brook University, USA

Scott D. Stoller
Department of Computer Science, Stony Brook University, USA

Ashish Tiwari
SRI International, USA

Junxing Yang
Department of Computer Science, Stony Brook University, USA

Scott A. Smolka
Department of Computer Science, Stony Brook University, USA

ABSTRACT

The popularity of rule-based flocking models, such as Reynolds' classic flocking model, raises the question of whether more declarative flocking models are possible. This question is motivated by the observation that declarative models are generally simpler and easier to design, understand, and analyze than operational models. We introduce a very simple control law for flocking based on a cost function capturing cohesion (agents want to stay together) and separation (agents do not want to get too close). We refer to it as *d*eclarative flocking (DF). We use model-predictive control (MPC) to define controllers for DF in centralized and distributed settings. A thorough performance comparison of our declarative flocking with Reynolds' model, and with more recent flocking models that use MPC with a cost function based on lattice structures, demonstrate that DF-MPC yields the best cohesion and least fragmentation, and maintains a surprisingly good level of geometric regularity while still producing natural flock shapes similar to those produced by Reynolds' model. We also show that DF-MPC has high resilience to sensor noise.

1 INTRODUCTION

Flocking is a collective behavior exhibited by a large number of interacting agents possessing a common group objective [4]. The term is most commonly associated with birds, and more recently, drones. Examples include foraging for food, executing a predator-avoidance maneuver, and engaging in migratory behavior.

With the introduction of Reynolds' model [7, 8], *rule-based control* became the norm in the flocking community. Specifically, in this model, at each time-step, each agent executes a control law given in terms of the weighted sum of three competing forces to determine its next acceleration. Each of these forces has its own rule: *separation* (keep a safe distance away from your neighbors), *cohesion* (move towards the centroid of your neighbors), and *alignment* (steer toward the average heading of your neighbors). As the descriptions suggest, these rules are executed by each agent in a distributed environment with limited-range sensing and no communication.

The popularity of Reynolds' model and its many variants raises the question: Is there a more abstract *declarative* form of control for flocking? This question is important because declarative models are generally simpler and easier to design, understand, and analyze than operational models. This is analogous to declarative programs (e.g., functional programs and logic programs) being easier to write and verify than imperative programs.

We show that the answer to this question is indeed positive by providing a very simple control law for flocking based on a *cost function* comprising two main terms: *cohesion* (the average squared distance between all pairs of agents) and *separation* (a sum of inverse squared distances, except this time between pairs of agents within each other's sensing range). That is it. For example, no term representing velocity alignment is needed. The cost function specifies what we want as the goal, and is hence declarative. In contrast, the update rules in Reynolds' model aim to achieve an implicit goal and hence are operational. Executing declarative control amounts to finding the right balance between attracting and repelling forces between agents. We refer to this approach as *Declarative Flocking* (DF). We use MPC (model-predictive control) to define controllers for DF, and refer to this approach as *DF-MPC*. We define a centralized version of DF-MPC, which requires communication, and a distributed version, which does not.

Previous MPCs for flocking exist, e.g., [11–13]. Most of these MPCs are designed to conform to the α-lattice model of flocking proposed in [4]. α-lattices impose a highly regular structure on flocks: all neighboring agents are distance d apart, for a specified constant d. This kind of structure is seen in some settings, such as beehives, but is not expected in many other natural and engineered settings, and it is not imposed by Reynolds' model.

In this paper, we show, via a thorough performance evaluation, how centralized and distributed DF-MPC compare to Reynolds' rule-based approach [7, 8], Olfati-Saber's potential-based approach [4], a variant of Zhan and Li's centralized lattice-based MPC approach [10, 11], and Zhang *et al.*'s distributed lattice-based MPC approach [12]. We consider performance measures that capture multiple dimensions of flocking behavior: number of sub-flocks (flock fragmentation), maximum sub-flock diameter (cohesion), velocity convergence, and a new parameter-free measure of the geometric regularity of the formation.

Our experimental results demonstrate that DF-MPC yields the best cohesion and least fragmentation, and produces natural flock shapes like those produced by Reynolds' model. Also, distributed

DF-MPC maintains a surprisingly good level of geometric regularity. We also analyze the resiliency of DF-MPC and the lattice-based MPC approaches by considering the impact of sensor noise. Our results demonstrate a remarkably high level of resiliency on the part of DF-MPC in comparison with these other approaches.

2 MODELS OF FLOCKING BEHAVIOR

We consider a set of dynamic *agents* $\mathcal{B} = \{1, \ldots, n\}$ that move according to the following discrete-time equation of motion:

$$x_i(k+1) = x_i(k) + dT \cdot v_i(k),\ v_i(k) \in V \tag{1}$$

$$v_i(k+1) = v_i(k) + dT \cdot a_i(k),\ a_i(k) \in A, \tag{2}$$

where $x_i(k), v_i(k), a_i(k) \in \mathbb{R}^m$ are respectively position, velocity and acceleration of agent $i \in \mathcal{B}$ in the m-dimensional space at step k, and $dT \in \mathbb{R}^+$ is the time step. We consider physical constraints on velocities and accelerations, described by the sets V and A, respectively, which are defined by $V = \{v \mid |v| \leq \bar{v}\}$ and $A = \{a \mid |a| \leq \bar{a}\}$, where $\bar{v}$ and $\bar{a}$ limit the allowed magnitude of the velocity and acceleration vectors, respectively.

The *configuration* of all agents is described by the vector $\mathbf{x}(k) = [x_1^T(k)\ \ldots\ x_n^T(k)]^T \in \mathbb{R}^{m \cdot n}$. Let $\mathbf{v}(k) = [v_1^T(k)\ \ldots\ v_n^T(k)]^T \in \mathbb{R}^{m \cdot n}$, and $\mathbf{a}(k) = [a_1^T(k)\ \ldots\ a_n^T(k)]^T \in \mathbb{R}^{m \cdot n}$. Then the equation of motion for all agents can be expressed as

$$\mathbf{x}(k+1) = \mathbf{x}(k) + dT \cdot \mathbf{v}(k), \tag{3}$$

$$\mathbf{v}(k+1) = \mathbf{v}(k) + dT \cdot \mathbf{a}(k), \tag{4}$$

The local neighborhood of agent i is defined by the set of other agents, called *neighbors*, within a given distance from i, mimicking the agent's visibility sphere. For an *interaction radius* $r > 0$ and configuration $\mathbf{x}$, the set of *spatial neighbors* of agent i, $N_i(\mathbf{x}) \subseteq \mathcal{B}$, is given by:

$$N_i(\mathbf{x}) = \left\{ j \in \mathcal{B} \mid j \neq i \wedge \|x_i - x_j\| < r \right\}, \tag{5}$$

where $\|\cdot\|$ denotes the Euclidean norm.

For configuration $\mathbf{x} \in \mathbb{R}^{m \cdot n}$, we define the associated *proximity net* $G(\mathbf{x}) = (\mathcal{B}, \mathcal{E}(\mathbf{x}))$ as the graph that connects agents within their interaction radius:

$$\mathcal{E}(\mathbf{x}) = \left\{ (i,j) \in \mathcal{B} \times \mathcal{B} \mid \|x_i - x_j\| < r, i \neq j \right\}, \tag{6}$$

Definition 2.1 (α-lattice [4]). A configuration $\mathbf{x} \in \mathbb{R}^{m \cdot n}$ is called *α-lattice* if for all $i \in \mathcal{B}$ and all $j \in N_i(\mathbf{x})$, $\|x_i - x_j\| = d$, where $d \in \mathbb{R}^+$ is the scale of the α-lattice. For tolerance $\delta \in \mathbb{R}^+$, a configuration $\mathbf{x} \in \mathbb{R}^{m \cdot n}$ is called a *quasi α-lattice* if for all $i \in \mathcal{B}$ and all $j \in N_i(\mathbf{x})$, $|\|x_i - x_j\| - d| \leq \delta$.

2.1 Reynolds' rule-based model

In Reynolds' rule-based distributed model [7, 8], each agent $i \in \mathcal{B}$ updates its acceleration $a_i(k)$ at step k by considering the following three components :

(1) *Alignment*: agents match their velocities with the average velocity of nearby agents.
(2) *Cohesion*: agents move towards the centroid of the agents in the local neighborhood.
(3) *Separation*: agents move away from nearby neighbors.

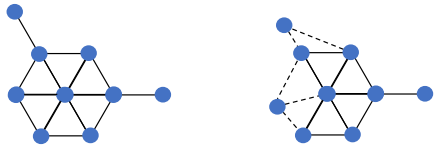


Figure 1: Examples of α-lattice a) and quasi α-lattice b). Solid lines connect agents in the same neighborhood that have distance d. Dashed lines connect those with have distance $d \pm \epsilon$ for $\epsilon \leq \delta$ (the tolerance).

2.2 Olfati-Saber's potential-based model

In potential-based flocking models, the interaction between a pair of agents is modeled by a potential field. It is assumed that an agent is a point source, and it has a potential field around it, which exerts a force, equal to its gradient, on other agents in its range of influence.In the work of Olfati-Saber [4], the potential function ψ_α for a pair of agents has its minimum at the desired inter-agent distance d of the desired α-lattice. Outside the interaction radius r, the potential function is constant, so the potential field exerts no force.

2.3 MPC-based models

Model predictive control (MPC) [1] is a well-established control technique that works as follows: at each time step k, it computes the optimal control sequence (agents' accelerations in our case) that minimizes a given cost function with respect to a predictive model of the controlled system and a finite prediction horizon of length T, i.e., from step $k+1$ to $k+T$. Then, the first control input of the optimal sequence is applied (the remainder of the sequence is unused), and the algorithm proceeds with a new iteration.

Two main kinds of MPC-based flocking models exist, *centralized* and *distributed*. Please refer to the full version of this paper on arxiv.org for further details.

3 DECLARATIVE FLOCKING

This section introduces centralized and distributed versions of our Declarative Flocking (DF) model, and presents a flocking algorithm based on MPC. Our formulation is declarative in that it consists of just two simple terms: (1) a cohesion term based on the average squared distance between pairs of agents, to keep the flock together, and (2) a separation term based on the inverse squared distances between pairs of agents, to avoid crowding. These two terms represent opposing forces on agents, causing agents to move towards positions in which these forces are balanced.

3.1 Centralized DF model

The cost function J for our centralized DF model contains the two terms described above, with the cohesion term considering all pairs of agents, and the separation term considering only pairs of agents that are neighbors. The weight ω of the separation term provides control over the density of the flock.

$$J^{\mathrm{C}}(\mathbf{x}) = \frac{2}{|\mathcal{B}| \cdot (|\mathcal{B}| - 1)} \cdot \sum_{i \in \mathcal{B}} \sum_{j \in \mathcal{B}, i < j} \|x_{ij}\|^2 + \omega \cdot \sum_{(i,j) \in \mathcal{E}(\mathbf{x})} \frac{1}{\|x_{ij}\|^2}$$

The control law is Eq. (??) with $J(k)$ equal to $\sum_{t=1}^{T} J^{\mathrm{C}}(\mathbf{x}(k+t \mid k))$.

3.2 Distributed DF model

The cost function J for our distributed DF model is similar to the centralized one, except that both terms are limited to consider pairs of agents that are neighbors.

$$J_i^{\mathrm{D}}(\mathbf{x}) = \frac{1}{|N_i(k)|} \cdot \sum_{j \in N_i(k)} \|x_{ij}\|^2 + \omega \cdot \sum_{j \in N_i(k)} \frac{1}{\|x_{ij}\|^2} \tag{7}$$

The control law for agent i is Eq. (??) with $J_i(k)$ equal to $\sum_{t=1}^{T} J_i^{\mathrm{D}}(\mathbf{x}(k+t \mid k))$.

4 MEASURES OF FLOCKING PERFORMANCE

We introduce four key measures of flocking performance. A single measure is insufficient, because flocking is indeed characterized by multiple desirable properties, such as aligned velocities and cohesion. Olfati-Saber introduces four main properties for flocking [4], informally described as:

(1) the group of agents stays *connected* in a unique flock, i.e., no sub-flocks and fragmentation should emerge;
(2) the group remains *cohesive*, in a close-knit formation;
(3) the group moves in a coherent way as if it was a unique body, i.e., agents' velocities are aligned; and
(4) the group maintains a regular geometry (in the α-lattice sense).

We introduce the following four measures to capture these four requirements. An important concept in these definitions is a *sub-flock*, which is a set of interacting agents that is too far apart from other agents to interact with them. Formally, a sub-flock in a configuration $\mathbf{x}$ corresponds to a connected component of the proximity net $G(\mathbf{x})$. Let $CC(\mathbf{x}) \subseteq 2^{\mathcal{B}}$ be the set of connected components of the proximity net $G(\mathbf{x})$.

(1) The *number of connected components* of the proximity net quantifies connectedness—or, equivalently, fragmentation—of the flock. There is no fragmentation when $|CC(\mathbf{x})| = 1$. Fragmentation exists when $|CC(\mathbf{x})| > 1$. Fragmentation may be temporary or, if sub-flocks move in different directions, permanent.

(2) The *maximum component diameter*, denoted $D(\mathbf{x})$, quantifies cohesion. It is defined by

$$D(\mathbf{x}) = \max_{\mathcal{B}' \in CC(\mathbf{x})} D(\mathbf{x}, \mathcal{B}') \tag{8}$$

where $D(\mathbf{x}, \mathcal{B}')$ is the diameter of connected component $\mathcal{B}'$:

$$D(\mathbf{x}, \mathcal{B}') = \max_{\substack{(i,j) \in \mathcal{B}' \times \mathcal{B}' \\ i \neq j}} \|\mathbf{x}_{ij}\|. \tag{9}$$

(3) The *velocity convergence* measure, adopted from [12], quantifies the average discrepancy between each agent's velocity and the average velocity of the flock. In particular, we extend the measure of [12] to average velocity convergence values across sub-flocks:

$$VC(\mathbf{x}, \mathbf{v}) = \frac{\sum_{\mathcal{B}' \in CC(\mathbf{x})} \left\| \sum_{i \in \mathcal{B}'} v_i - \left(\frac{\sum_{j \in \mathcal{B}'} v_j}{|\mathcal{B}'|} \right) \right\|^2 \Big/ |\mathcal{B}'|}{|CC(\mathbf{x})|} \tag{10}$$

(4) To measure the regularity of the geometric structure of a flock, as reflected in the inter-agent spacing, we introduce a parameter-free and model-independent *irregularity* measure $I(\mathbf{x})$. For a connected component (sub-flock) $\mathcal{B}'$, it is defined as the sample standard deviation of the distances between each agent in $\mathcal{B}'$ and its closest neighbor. Thus, the measure penalizes configurations where there is dispersion in inter-agent distances, while not imposing any fixed distance between them (unlike α-lattices).

Let $CC'(\mathbf{x}) = CC(\mathbf{x}) \setminus \bigcup_{i \in \mathcal{B}} \{\{i\}\}$ be the set of connected components where isolated agents are excluded. For $|CC'(\mathbf{x})| = 0$ (or equivalently, $|CC(\mathbf{x})| = |\mathcal{B}|$), i.e., all agents are isolated, we set the irregularity $I(\mathbf{x}) = 0$, which is the optimal value. This reflects the fact that a single point is a regular structure on its own. Moreover, such a configuration is already highly penalized by $|CC(\mathbf{x})|$ and $VC(\mathbf{v})$. For $|CC'(\mathbf{x})| > 0$, the measure is defined by:

$$I(\mathbf{x}) = \frac{\sum_{\mathcal{B}' \in CC'} \sigma\left(\biguplus_{i \in \mathcal{B}'} \min_{j \neq i} \|x_{ij}\| \right)}{|CC'|}. \tag{11}$$

where $\sigma(S)$ is the standard deviation of the multiset of samples S and $\biguplus$ is the sum operator (or disjoint union) for multisets.

An α-lattice (see Def. 2.1) has the optimal value of $I(\mathbf{x})$, i.e., $I(\mathbf{x}) = 0$, since all neighboring agents are located at the same distance d from each other, leading to zero standard deviation for the term $\sigma(\{d, d, \ldots, d\})$. This shows that $I(\mathbf{x})$ captures the regularity underlying the concept of α-lattice.

We introduce this measure because previous measures of regularity or irregularity, such as those in [4, 11, 12], measure deviations from an α-lattice with a specified inter-agent distance d and are therefore inapplicable to flocking models, such as Reynolds' model and our DF models, that are not based on α-lattices and do not have a specified target inter-agent distance. Also, our irregularity measure is more flexible than those based on α-lattices, because it gives an optimal score to some configurations that are geometrically regular but not α-lattices.

5 PERFORMANCE EVALUATION

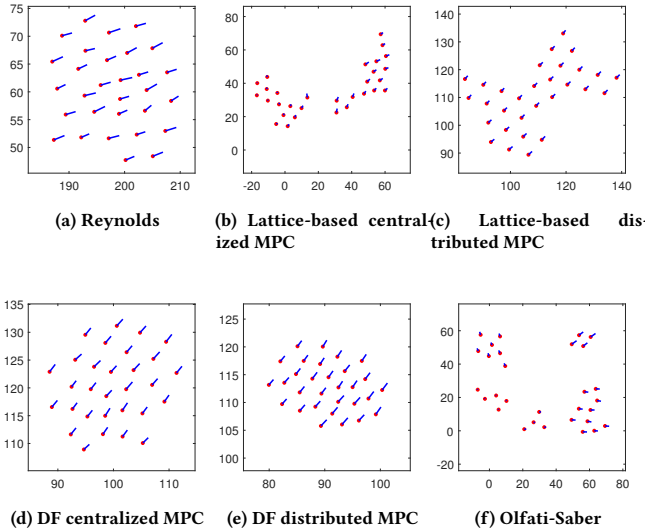
We compare the performance of the models of Section 2 with the newly introduced DF flocking models in the 2-dimensional setting. In the first set of experiments (Section 5.1), we evaluate the performance measures illustrated in Section 4. In the second set of experiments (Section 5.2), we analyze the resilience of the algorithms to sensor noise.

Unless otherwise specified, the population size is $n = 30$, the simulation length is 100, $dT = 0.3$, $\bar{v} = 8$, $\bar{a} = 1$, $r = 8.4$, $d = 7$, $T = 3$, and $\lambda = 1$. For further details about experimental settings please refer to the full version on arxiv.org.

5.1 Performance Comparison of Flocking Algorithms

Fig. 2 shows examples of final formations for all flocking models.

In Fig. 3, we compare the performance measures averaged over 100 runs for each flocking model. Regarding the number of connected components (sub-flocks), our centralized DF-MPC registers the best behavior, rapidly stabilizing to an average of 1 component (see plot a). Our distributed DF-MPC and Reynolds' model have



(a) Reynolds (b) Lattice-based centralized MPC (c) Lattice-based distributed MPC

(d) DF centralized MPC (e) DF distributed MPC (f) Olfati-Saber

Figure 2: Examples of final formations for different flocking models. The red dots are the agent positions. The blue lines denote the agent velocity vectors

comparable performance, reaching an average number of sub-flocks below 1.4. The lattice-based MPCs and Olfati-Saber instead lead to constant fragmentation, with more than 2 sub-flocks for the distributed lattice-based MPC, 6 for the centralized lattice-based MPC, and more than 8 for Olfati-Saber's model.

This ranking is confirmed by the diameter measure (plot b), where our centralized and distributed DF-MPC and Reynolds' model show the best cohesion, outperforming the lattice-based approaches. Recall that this measure indicates the maximum diameter over all sub-flocks, not the diameter of the entire population. As a consequence, fragmentation tends to improve diameter values since it produces sub-flocks with fewer individuals. This explains why our distributed DF-MPC performs better on this measure than the centralized version, and similarly why Olfati-Saber's model has smaller diameter measure than centralized lattice-based MPC, which in turn has smaller diameter measure than the distributed variant.

As expected, Olfati-Saber's model and the lattice-based MPCs have very good performance for irregularity (plot c), since they are designed to achieve the regular geometric formation of α-lattice. Surprisingly, our distributed DF-MPC performs almost as well as them on this measure. Centralized DF-MPC and Reynolds' model have the least regular formations.

For velocity convergence (plot d), we find that all models perform comparably well and are able to achieve flocks with consistent velocities fairly quickly after an initial spike.

5.2 Robustness to Sensing Noise

To evaluate the resiliency of the models to sensor noise, we performed 20 runs for each model at 10 noise levels. The noise levels are numbered from 1 to 10, and noise level i has $\sigma_x = 0.2i$ and $\sigma_v = 0.1i$. For each performance metric, we averaged its final values over 20 runs for each noise level. The results are plotted in Fig. 4.

Of the six models, Olfati-Saber's model is the most vulnerable to sensing noise: the number of sub-flocks $|CC|$ in Olfati-Saber's model quickly increases to nearly 30, rendering other metrics irrelevant. The lattice-based MPC models also exhibit high fragmentation, leading to nominally good but largely irrelevant values for the other performance metrics. Our distributed DF-MPC and Reynolds' model have the best resiliency to sensing noise, with both models exhibiting similar profiles in all metrics. While the irregularity and velocity convergence measures increase with noise level, as expected, both models remarkably maintain almost a single connected component with a nearly constant component diameter for all 10 noise levels, with DF-MPC achieving a smaller diameter than Reynolds' model.

6 RELATED WORK

Reynolds [7] introduced the first rule-based approach for simulation of flocking behavior. With three simple rules, his model is able to capture complex flocking behaviors of animals. Similar rule-base flocking models are also proposed by Pearce *et al.* [5] and Cucker and Dong [2].

Artificial potential fields have also been used extensively in flocking models. For example, Tanner *et al.* [9]. Ogren et.al. [6] use the motion of the leader to guide the motion of the flock; the leader's motion is independent.

La and Sheng [3] propose an extension of Olfati-Saber's model designed for noisy environments. In addition to the terms found in Olfati-Saber's model, their control law contains feedback terms for position and velocity, to make agents tend to stay close to the centroid of their neighborhood and minimizing the velocity mismatch with their neighbors. For further details regarding the related works please refer the full version of this paper on arxiv.org.

7 CONCLUSIONS

This paper presents an abstract declarative form of control for flocking behavior and the results of a thorough comparison of centralized and distributed versions of our MPC-based declarative flocking with four other flocking models. Our simulation results demonstrate that DF-MPC yields the best cohesion and least fragmentation, and produces natural flock shapes like those produced by Reynolds' rule-based model. Our resiliency analysis shows that the distributed version of our DF-MPC is highly robust to sensor noise.

As future work, we plan to study resilience of the flocking models with respect to additional noisy scenarios such as actuation noise (i.e., noise affecting acceleration) and faulty agents with deviant behavior. We also plan to investigate smoothing techniques to increase resilience to sensor noise.

REFERENCES

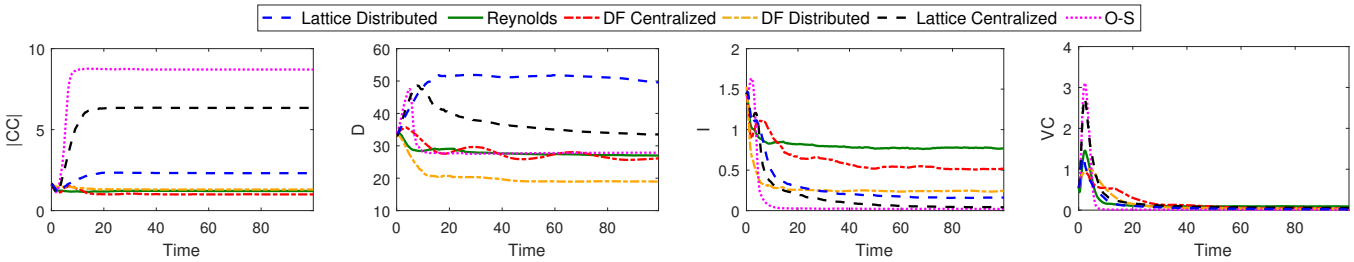
[1] E.F Camacho and C. Bordons. 2007. *Model predictive control.* Springer.

[2] Felipe Cucker and Jiu-Gang Dong. 2011. A general collision-avoiding flocking framework. *IEEE Trans. Automat. Control* 56, 5 (2011), 1124–1129.

[3] H. M. La and W. Sheng. 2010. Flocking control of multiple agents in noisy environments. In *2010 IEEE International Conference on Robotics and Automation.* 4964–4969. https://doi.org/10.1109/ROBOT.2010.5509668

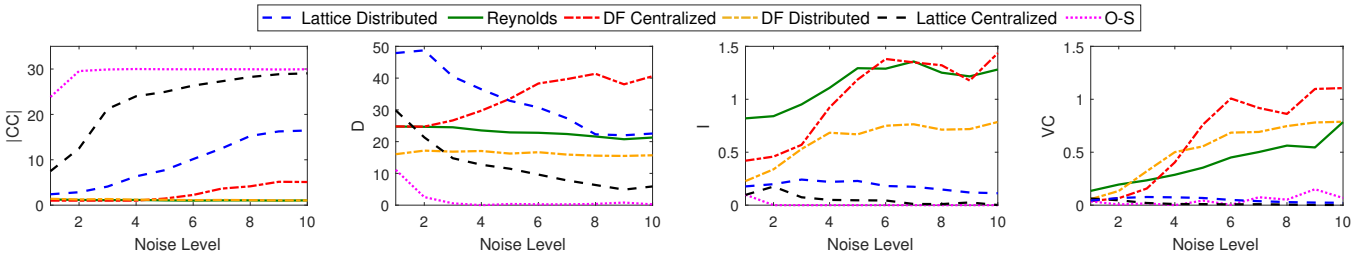
[4] Reza Olfati-Saber. 2006. Flocking for multi-agent dynamic systems: Algorithms and theory. *IEEE Transactions on automatic control* 51, 3 (2006), 401–420.

[5] Daniel J. G. Pearce, Adam M. Miller, George Rowlands, and Matthew S. Turner. 2014. Role of projection in the control of bird flocks. *Proceedings of the National*



(a) Number of connected components $|CC|$ **(b) Max component diameter D** **(c) Irregularity I** **(d) Velocity convergence VC**

Figure 3: Comparison of performance measures obtained with 100 runs for each flocking algorithm.



(a) Number of connected components $|CC|$ **(b) Max component diameter D** **(c) Irregularity I** **(d) Velocity convergence VC**

Figure 4: Comparison of the final values of the performance measures obtained with 20 runs for each flocking algorithm and for each noise level.

Academy of Sciences 111, 29 (2014), 10422–10426. https://doi.org/10.1073/pnas.1402202111 arXiv:http://www.pnas.org/content/111/29/10422.full.pdf

[6] Naomi Ehrich Leonard Peter Ogren. 2004. Cooperative control of mobile sensor networks:Adaptive gradient climbing in a distributed environment. *IEEE transactions on Automatic Control* 49, 8 (2004).

[7] Craig W. Reynolds. 1987. Flocks, Herds and Schools: A Distributed Behavioral Model. *SIGGRAPH Comput. Graph.* 21, 4 (Aug. 1987), 25–34. https://doi.org/10.1145/37402.37406

[8] Craig W. Reynolds. 1999. Steering Behaviors For Autonomous Characters. In *Proceedings of Game Developers Conference 1999*. 763–782.

[9] H. G. Tanner, A. Jadbabaie, and G. J. Pappas. 2003. Stable flocking of mobile agents part I: dynamic topology. In *42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)*, Vol. 2. 2016–2021 Vol.2.

[10] Jingyuan Zhan and Xiang Li. 2011. Flocking of discrete-time multi-agent systems with predictive mechanisms. *IFAC Proceedings Volumes* 44, 1 (2011), 5669–5674.

[11] Jingyuan Zhan and Xiang Li. 2013. Flocking of multi-agent systems via model predictive control based on position-only measurements. *IEEE Transactions on Industrial Informatics* 9, 1 (2013), 377–385.

[12] Hai-Tao Zhang, Zhaomeng Cheng, Guanrong Chen, and Chunguang Li. 2015. Model predictive flocking control for second-order multi-agent systems with input constraints. *IEEE Transactions on Circuits and Systems I: Regular Papers* 62, 6 (2015), 1599–1606.

[13] Lifeng Zhou and Shaoyuan Li. 2017. Distributed model predictive control for multi-agent flocking via neighbor screening optimization. *International Journal of Robust and Nonlinear Control* 27, 9 (2017), 1690–1705.

Realizing Minimum Spanning Trees from Random Embeddings

Ion Mandoiu [*], Saad Quader [†], and Alexander Russell [‡]

Department of Computer Science & Engineering, University of Connecticut, USA.

November 1, 2017

Abstract

Let $T = (V, E)$ be an undirected tree with n vertices. For any arbitrary $x, y \in \mathbb{R}$, let $f : V \to \{x, y\}^d$ be a random embedding of the tree-vertices where each $f(v)$ is selected independently and uniformly at random. We study the event that there exist nonnegative weights $w_1, \ldots, w_d$ so that T is "realized" by this embedding as the unique minimum spanning tree of the points $f(V)$ under the scaled ℓ_2 metric $\|x\|^2 = \sum w_i x_i^2$. The realization occurs in the following sense: under this metric, the distance between two embedded vertices will be smaller than a threshold if and only if these vertices are neighbors in T. We wish to bound the dimensionality d for which it is possible to realize T with high probability.

We show that any tree can be realized with high probability when $d = \Omega(n \log n)$. The proof gives rise to a simple algorithm that needs only select $w_i \in \{0, 1\}$ and works for both ℓ_2 and ℓ_1 metrics. We additionally study the case for general undirected graphs. We show two sufficient conditions in this case: we show that $d = \Omega(na^2 \log n)$ is sufficient to realize any graph with high probability where a is the arboricity of that graph, and that $d = \Omega(nr^{-2} \log n)$ is also sufficient where r is the smallest effective resistance of the edges in the graph. The former bound becomes $d = \Omega(n|E| \log n)$ in the worst case. We also show that $d = \Omega(n^2)$ and $d = \Omega(n)$ are necessary to realize an Erdős-Rényi random graph and a random n-vertex tree, respectively. We develop a probabilistic analog of Radon's theorem on convex sets, which may be of independent interest.

Variants of this natural "realizability problem" play a basic role in statistical inference of gene expression data, where the existence of such a scaled metric is taken as evidence for the relevance of the expression data to the biological dynamics modeled by the tree.

Link to full version. https://www.dropbox.com/s/61mulylcs0kbx0o/Realizing-Trees.pdf?dl=1

[*]Email: ion@engr.uconn.edu

[†]Corresponding author. Email: saad.quader@uconn.edu

[‡]Email: acr@cse.uconn.edu

The Minimum Road Trips Problem

Samuel Micka and Brendan Mumey
Gianforte School of Computing, Montana State University
Bozeman, Montana, USA
samuel.micka@msu.montana.edu and brendan.mumey@montana.edu

***Abstract*—Road networks can be represented as partially directed graphs; directed edges are one-way road segments and undirected edges can be traversed in either direction. Vehicle trips are simply paths from some starting vertex to some ending vertex that must agree with the direction of any directed edge taken. A *loop detector* is a device that counts the number of vehicles that cross an edge during some time period. Loop detectors are typically present on only a subset of the edges in the network. The basic problem we are interested in is to determine the minimum number of trips (simple paths) needed to explain all of the loop detector count measurements. We also consider a dynamic version of the problem in which time is discretized and vehicles move one edge per time step. In this case, the loop detectors provide traffic counts for each time step and the goal is again to determine the fewest number of trips needed to explain the data. Trips are now specified by a path and a starting time.**

I. Introduction

Networks arise everywhere in the modern world, from roadways to the Internet, networks are responsible for delivering us, and our ideas, to everyone else. Restricting our attention to road networks, we can represent the roadways as edges in a graph and intersections as vertices. Furthermore, the flow in this graph is representative of vehicles traveling from one location to another. We refer to individual vehicle paths as trips that describe the vehicular flow. Determining detailed trip information about vehicles is a difficult task without monitoring each individual vehicle. However, due the prevelance of loop detector data, it is not difficult to represent the network as a graph with a vehicle count on various edges. In this work we consider the problem of finding the smallest number of trips that could be responsible for the known edge volumes in the given graph.

We consider a network with a volume associated with some edges in the graph. A volume represents the number of vehicles traversing that edge. We discuss various problem formulations, each of which consider different types of graphs or trip decompositions. We cover complexity results and preliminary ideas for algorithmic solutions for inferring trips from these graphs.

Providing the underlying trips responsible for flow volumes in a network offers insight into the structure of vehicular flows. This structure can help provide input to planning and routing algorithms to make vehicles travel more efficiently, ultimately reducing travel times and congestion in various network settings.

In previous work, the problem of flow decomposition for individual commodities has been considered [1], [2], [3]. In the research done by Vatinlen *et al.*, the authors introduce two greedy heuristics for extracting the smallest number of paths from a s-t multipath flow [1]. The work done by Hartman *et al.* extend the work done by Vatinlen *et al.* by comparing their heuristics against new methods [2]. Specifically, the authors introduce a new approximation algorithm that decomposes the flow into no more than $(1/\epsilon^2)$ times the optimal number of paths. Two versions of their approximation algorithm are compared against the heuristics introduced by [1] and the width-based decomposition algorithm performs almost as well as the original width-based heuristic. Mumey *et al.* consider the same problem but offer improvements to the approximation bound [3]. The new algorithms are logarithmically bounded by the length of the longest path in the flow and the largest flow volume on any particular edge. We divert from previous research by considering graphs that do not have pre-defined sources and destinations. This formulation more accurately represents instantaneous edge volume data in a generic network, such as a roadway. Furthermore, we are interested in finding plausible trips in order to extract more precise information about individual agents in the graph. This type of information can provide insight into the current status of road networks, such as a lower bound on the number of vehicles on a road system at a given time.

II. Problem Formulation

To better understand graph flows in the context of road networks and vehicles, an accurate model of the roadway must be considered. To represent the roadway, we consider a mixed-graph defined as $G = \langle V, E, A \rangle$ where V is the set of vertices (intersections), E is the set of undirected edges, and A is the set of directed edges, or arcs. We define a partial volume function $vol : E \cup A \to \mathbb{Z}_+$. In other words, edges in some subset of $E \cup A$ have positive integer volumes associated with them, while some edges may remain unlabeled. Undirected edges are representative of roadways that can be traversed in either direction. Then, with vehicles, we are interested in the problem of identifying individual paths which can be used to identify large trends in movement data that might, otherwise, go unnoticed. Specifically, we want our decomposition to be composed of the fewest number of vehicle trip paths, hereon referred to as trips, that could explain all edge counts in the graph. Formally, *trips* are walks through the graph that do not contain repeated edges or vertices, i.e. simple paths. We restrict our attention to trips to avoid single vehicles explaining all of the traffic around cycles in the graph. When a trip traverses an edge, it accounts for a single unit of volume. We refer to this formulation as the Minimum Road Trips (MRT) problem.

Lemma 1. *MRT is NP-hard.*

Proof. We show that the problem is NP-hard with a reduction from 3-SAT. The 3-SAT problem asks whether there exists a truth assignment to variables $x_1, \ldots, x_n$ that will satisfy the boolean formula with m clauses which each contain three variables: $(x_i \vee x_j \vee x_k) \wedge \ldots \wedge (x_x \vee x_y \vee x_z)$. For an instance of 3-SAT with n variables, we create a corresponding instance of MRT such that there is an solution of the MRT instance with exactly n trips if and only if the 3-SAT instance is satisfiable. The construction is as follows: We begin by creating an undirected edge for each variable x_i for $i \in \{1, \ldots, n\}$, denoted as e_{x_i}. Set $vol(e_{x_i}) = 1$ for all variable edges. The left endpoint of a variable edge is associate with $\neg x_i$ and the right endpoint with x_i. Next, we create clause gadgets, each clause gadget is a directed cycle consisting of six edges, three of which with unit volume. Arrange the clause gadgets vertically as shown in Fig. 1. Beginning with the top clause, we create an edge to each literal in the clause. This edge originates at either the associated variable edge, if that literal has not appeared before in a clause, or from the last clause it was used in. We also label this newly created edge with the literal. Note that edges between clauses are directed downwards (towards higher number clauses). Note, that a variable traversing a clause does not result in a cycle because the variable can exit the clause one edge before repeating a vertex while simultaneously covering all unit edges in the clause.

Suppose the 3-SAT instance does have a satisfying assignment. We can create a trip solution for the MRT instance by creating a trip that originates at each variable edge; if x_i is true in the assignment then this trip originates at $\neg x_i$ and leaves the edge from x_i. If the trip enters a clause gadget, and the edges of the clause gadget have not yet been traversed, then the trip makes a traverse around the clause gadget prior to continuing if there is an outgoing edge with that literal label. Since each clause is satisfied by the truth assignment, some trip will reach each clause. Thus, the n trips created will provide a solution to the MRT instance.

Conversely, if there is an n trip solution to the MRT instance, the 3-SAT instance will be satisfiable. To see this, we observe that each trip must originate at a unique variable edge (exactly one trip must traverse each variable edge and there are n variable edges and n trips). Next, we argue that we can modify the trip solution so that all trips agree with the edge labels created. Observe that if one trip visits a clause gadget, then it must circle the gadget and return to the literal that entered on. If the trip continues it must do so on the correctly-labeled exiting edge. If two trips enter the gadget, then it is possible that the trips exit on the wrong label. However, we can do a path swap to modify these trips so that they continue on the correct labels, as shown in Fig. 2. Similarly, if three trips enter a clause gadge, it is easy to see that they can be path-swapped if needed so that all exiting trips do so on the correct labels. This process of path-swapping is continued until all trips agree with all edge labels. We set the truth value of each variable x_i according to which endpoint the trip leaves e_{x_i}. Since each clause gadget is now circled by a single literal (assigned true), this truth assignment satisfies all clauses. □

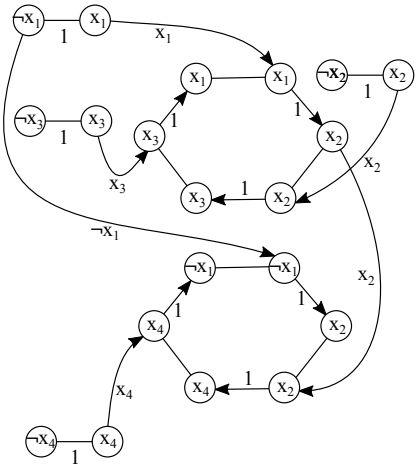
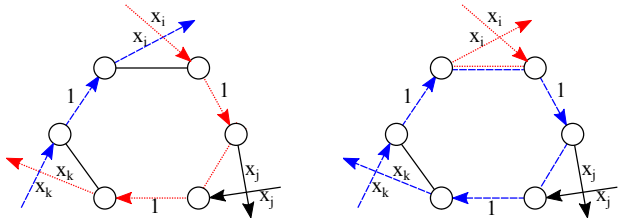


Fig. 1: The MRT instance corresponding to the 3-SAT instance $(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_4)$.



(a) Before (b) After

Fig. 2: By path swapping, all trips passing through a clause gadget can be routed to agree with their edge labels.

III. A Heuristic MRT Algorithm

In this section, we introduce a heuristic for the MRT Problem. The algorithm works by finding a *maximal* trip in the network, adding it to the set of trips τ, reducing the remaining volume of the volume-labeled edges along the path by 1 and repeating this process until all the volumes are accounted for. We refer to this algorithm as the Greedy Trips (GT) algorithm. A maximal trip p is constructed by starting with some edge that has positive volume and greedily extending the path in both directions until further expansion is not possible without adding an already-used vertex. The trip p is then trimmed so that it starts and ends with edges with positive volume, and added to the MRT trips solution. Volume labels along p are decremented by 1 and the GT algorithm continues until no edges with positive volume remain. Pseudocode can be found in Algorithm 1.

The trips returned by the GT heuristic will be a valid MRT solution as it accounts for all edge volumes in G. However, the solution is not necessarily minimal and we have not yet determined any performance guarantees for this heuristic on the general MRT problem.

IV. Variations on MRT

A. Directed Acyclic Graphs with No Missing Measurements

Under certain circumstances, we may have a graph that does not contain any undirected edges or cycles and has fully known

edge volumes. When the graph contains no undirected edges, no cycles, and there are no unknown edge volumes, i.e. we have a Directed Acyclic Graph (DAG), the problem becomes solvable in polynomial time with the heuristic described in Section III.

Lemma 2. *For the MRT problem where the graph is a DAG with no unknown edge counts, the GT algorithm produces (in polynomial time) an optimal trip solution.*

Proof. Let τ be the solution returned by the GT algorithm and let τ^* be an optimal solution. We will show that $|\tau| = |\tau^*|$ by demonstrating that every trip in τ is either already in τ^*, or that the trip can exist in τ^*, without producing any additional trips, through path swapping. Let p be a trip returned in τ, if $p \in \tau^*$, then p is compatible with the optimal solution and we are done. However, if p is not in τ^*, then p must be covered by a set of trips $[t_1, \ldots, t_i] \in \tau^*$. We order these trips in τ^* as they appear when we follow p from the source to the sink. Note that, since p is maximal, the first trip t_1 must start at p's source vertex and the final trip t_i covering p must end at p's sink vertex. Then, we can transform the trips that cover p in τ^* into p without adding any extra trips, demonstrating that p is compatible with the optimal solution. We start at p and t_1's source and follow p until t_1 diverges, at this divergence, we note that t_2 must enter the trail that p follows in order to cover the volume that t_1 misses by diverging. Then, we perform a path swap with t_1 and t_2 so that t_2 continues where t_1 diverged and t_1 continues to follow p. We continue to perform path swaps on t_1 and t_k, for $1 < k < i$, until we reach trip t_i, swapping t_1 and t_i allows us to set the sink of t_1 to be equal to the sink of p, making $p = t_1$ and proving that p compatible with τ^* without adding any additional trips. The described procedure can be repeated on all trips in τ to show that $\tau = \tau^*$ through a series of path swaps.

Finally, we must show that the algorithm runs in polynomial time. First, we consider time to find a maximal trail; in the worst case, we have to traverse every edge in the graph, so the time is bounded by $O(|E|)$. The number of iterations of the loop in the GT algorithm is bounded by $S = \sum_{e \in E \cup A} vol(e)$, since each maximal trail found decreases the volume remaining on at least one edge. Each loop iteration takes at most $O(|E|)$ time. Thus, the running time of the GT algorithm is $O(S|E|)$.

□

Algorithm 1 Greedy Trips

$\tau = \emptyset$
while $\exists\ e \in G$ with $vol(e) > 0$ **do**
  Find a maximal trip $p \in G$ as described
  $\tau = \tau \cup \{p\}$
  Decrement volume-labeled edges in p by 1
end while
return τ

B. Dynamic Minimum Road Trips Problem

In this section, we consider a mixed dynamic graph with edge volumes known for a subset of the edges. In essence, this version of the problem is a direct extension of the formulation discussed in Section II, but with a dynamic graph. We refer to the problem as the Dynamic Minimum Road Trips (DMRT) Problem and ask for the smallest set of trips that explain the edge volumes in the dynamic, time-varying, road network. The network is represented as a mixed graph $G = \langle V, E, A \rangle$ where V is the vertices and E and A represent the undirected and directed edges respectively. We denote the lifetime of G to be T, where T is a positive integer value (i.e. $lt(G) = T$). Edges $e \in E$ and $a \in A$ may have a volume $vol(e,t)$ or $vol(a,t)$ at each discrete time $t \in [1, \ldots, T]$ over the lifetime of G representative of the number of trips using that edge during that time. The volume of some edge or arc is either a positive integer, or unknown. Specifically, we have a partial volume function $vol : (E \cup A, t) \to \mathbb{Z}_+$. The solution to the DMRT Problem is the smallest set of trips that explain the edge volumes in G. We assume that it takes each flow one time unit to traverse any edge in the graph.

DMRT with No Undirected Edges

In this section we describe a reduction from the special case of DMRT where $E = \emptyset$, i.e. all edges in the graph are directed, to the original MRT problem. Given a DMRT instance $G = \langle V, E, A \rangle$, recall that the lifetime of the network $lt(G) = T$, then we have T discrete time intervals from $[1, T]$. For each $t \in [1, k]$ we create a new copy of G at time t and refer to it as $G_t =< V_t, E_t, A_t >$. We duplicate each $v_t \in V_t$ into v_t^{in} and v_t^{out}. We set any existing arc $(v_i, v_j) = (v_i^{in}, v_j^{out})$. This modification to the edges and vertices ensures that a trip can not travel more than one arc during a single time epoch. Then, for each modified arc $a_t \in A_t$ set $vol(a_t) = vol(a,t)$ where $vol(a,t)$ is specified in the original network G at time t. The construction of these graphs gives us T new copies of G, each with edge weights equal to those of G at a particular time interval.

Next, we connect these graphs using directed edges to make one, large, graph. Let t be a time value in the discrete interval $[1, T-1]$, then we connect G_t to G_{t+1} by adding an arc a from each $v_t^{out} \in V_t$ to each corresponding $v_{t+1}^{in} \in V_{t+1}$. We specify the $vol(a)$ to be undefined so that any number of trips can travel on these edges. However, a minimal trip solution will not generate extra trips that are not necessary to explain all labeled volumes in the graph. This ensures that extra trips will not be introduced on these directed edges connecting the graphs.

See Figure 3 for an example of the reduction, shown in the gray boxes. The original dynamic graph, shown at the bottom of the figure and labeled G, has only four vertices and a lifetime of three. The edge volumes at each time interval are clarified in the reduction, but not shown the original topology. For each time unit, a copy of the graph is stacked vertically above the last and, for clarity, each copy is encased by a gray rectangle. The dotted edges, with unspecified volume, connect

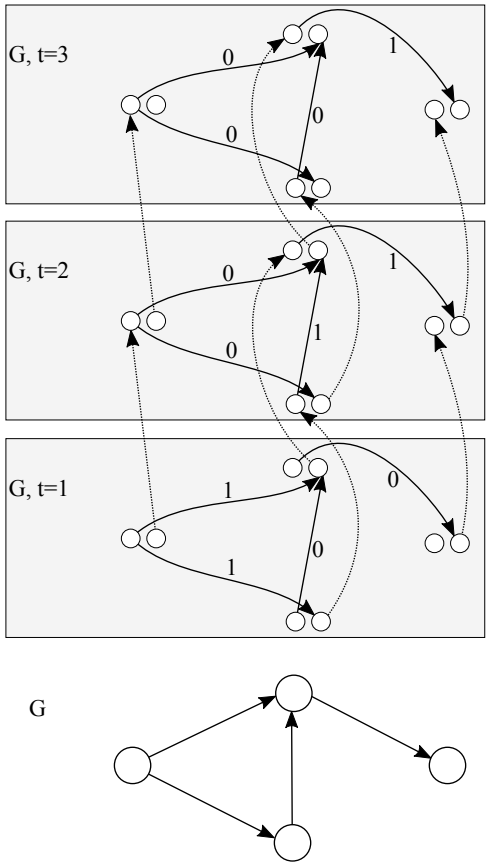


Fig. 3: Full reduction of the DMRT problem to the MRT problem. The graph at the bottom of the figure is the original topology of G. The reduction shows each graph at some discrete time enclosed by a gray box the dotted, directed, edges connect the different copies of the graph.

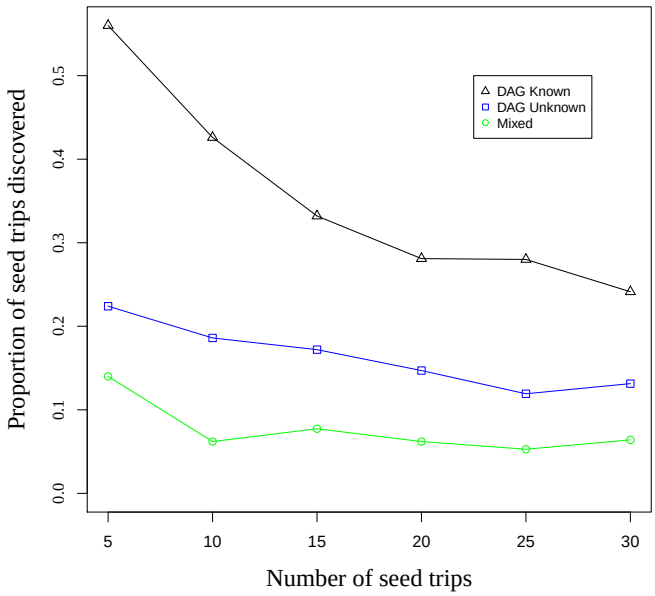


Fig. 4: The x-axis shows the number of trips generated on each graph. The y-axis shows the average proportion of correctly discovered trips (per 50 graphs). The three types of graphs were all 10x10 grids. The first, DAG Known, was a DAG with a loop detector on every edge (i.e. each edge has a known volume). The second type of graph, DAG Unknown, is a DAG with each edge having a 80% probability of having a loop detector. The final type of graph, Mixed, is a graph with both directed and undirected edges with each edge having an 80% chance of having a loop detector.

the copies of the graphs to unite them into one, large, static graph.

V. Discussion and Future Work

In this research, we have discussed various formulations related to the determining trips from traffic volume information in road networks. Solutions from different formulations can be used to answer different types of questions. For example, a solution to the MRT problem can provide a lower bound for the number of drivers on a road system at a given time. Alternatively, a solution to the DMRT problem can provide a set of potential vehicle routes over a specific time interval. In some preliminary simulations we have highlighted the proportion of correctly discovered trips using the heuristic described in the paper on three different types of synthetically generated 10x10 grid graphs and trips (i.e. simple random walks). Specifically, for each number of trips 50 *dag known* graphs (DAG with all edge weights known), 50 *dag unknown* graphs (DAG with some unknown edge weights), and 50 *mixed graphs* (undirected and directed edges with some missing edge weights) were generated. Then, the heuristic was used to discover trips in each graph, Figure 4 shows the average proportion of correctly identified trips on each type of graph for different numbers of synthetically generated trips. We can see that fewer walks leads to less ambiguity, presumably because there will be less overlap between the trips and therefore, less of an opportunity to merge them or swap their paths. The figure demonstrates that accuracy decreases as edge volumes are removed and, then again, as undirected edges are introduced.

Future work includes the development of additional heuristics and approximation algorithms for the these formulations as well as generalizing the DMRT reduction to work for undirected edges. We would like to introduce other techniques to help improve the trip extraction process to more accurately represent real paths taken by drivers. Future simulations will use real loop detector data sets from actual road networks.

References

[1] B. Vatinlen, F. Chauvet, P. Chrétienne, and P. Mahey, "Simple bounds and greedy algorithms for decomposing a flow into a minimal set of paths," *European Journal of Operational Research*, vol. 185, no. 3, pp. 1390–1401, 2008.

[2] T. Hartman, A. Hassidim, H. Kaplan, D. Raz, and M. Segalov, "How to split a flow?," in *INFOCOM, 2012 Proceedings IEEE*, pp. 828–836, IEEE, 2012.

[3] B. Mumey, S. Shahmohammadi, K. McManus, and S. Yaw, "Parity balancing path flow decomposition and routing," in *Globecom Workshops (GC Wkshps), 2015 IEEE*, pp. 1–6, IEEE, 2015.

Harder Hardness of Approximation for 2D r-Gather

Matthew P. Johnson*

1. INTRODUCTION

In the r-Gather problem [4], we are given a set of n points in a metric space and a parameter r. The task is to partition the points into subsets, or *clusters*, with each having cardinality at least r. The objective is to minimize the maximum diameter of the clusters, i.e., the diameter of the minimum enclosing disk surrounding the cluster's points. The problem is solvable in polynomial time when $r = 2$. When $r \geq 3$, the problem is NP-hard, it admits a 2-approximation algorithm, and is known to be NP-hard to approximate with any factor better than 2.

In the special case where the points lie in the plane and distance is Euclidean, the known hardness of approximation lower bounds are weaker, but there is no known approximation algorithm with factor better than 2. Here the problem is known to be NP-hard to approximate better than $\sqrt{13}/2 \approx 1.802$ when $r = 3$ and $\frac{\sqrt{35}+\sqrt{3}}{4}$ when $r \geq 5$ [4].

Results. We improve the hardness of approximation lower bounds for the cases of $r = 4$ and $r \geq 5$, to approximately 1.938443 and 1.938727, respectively.

2. OVERVIEW OF CONSTRUCTION

We reduce from the NP-hard variant of Planar 3SAT *restricted to instances where each variable appears in at most three constraints*[1] [2]. Given a SAT formula, each variable is modeled by a collection of points in a configuration that can be visualized as a circular chain of unit disks, containing an even number of disks. Each of the points corresponding to a variable lies at the intersection of two adjacent disks. Except in locations where *connection gadgets* are attached and at clause gadgets, each point location contains either one point or $r - 1$ points, alternating around the variable cycle.

A clause gadget consists of an equilateral triangle with side length $\sqrt{3}$ having r points located at each of its vertices and 1 point at its center (see Fig. 1(b)). This permits any one of its vertices' r points to feasibly cluster with its center point. Each vertex has r rather than $r - 1$ points because if more than one of its vertices wish to cluster with the center point, this also permits each vertices' points to cluster by themselves. If none of the vertices wishes to cluster with the center, i.e., if $r - 1$ of each of them is clustered with other points, then the center point is forced to cluster with diameter at least 2 (with the three vertices' three leftover points, in the case of $r = 4$, or else with one of the vertices' neighbors).

We now state some definitions.

DEFINITION 1. *An r-gather solution is* consistent *if every non-clause-adjacent cluster corresponds to all points lying within a unit disk and each clause point lies in a cluster containing* one *of its three* (r) *neighbors.*

DEFINITION 2. *A unit disk is* on *if all its points are clustered together (with no other points), and* off *if each of its points are clustered in the disk's neighboring disks' clusters.*

Observe that a consistent solution induces a parity on the collection of unit disks, where all the disks in a chain alternate between on and off, and thus all cluster diameters are 1 (except for clause-adjacent clusters of diameter 0).

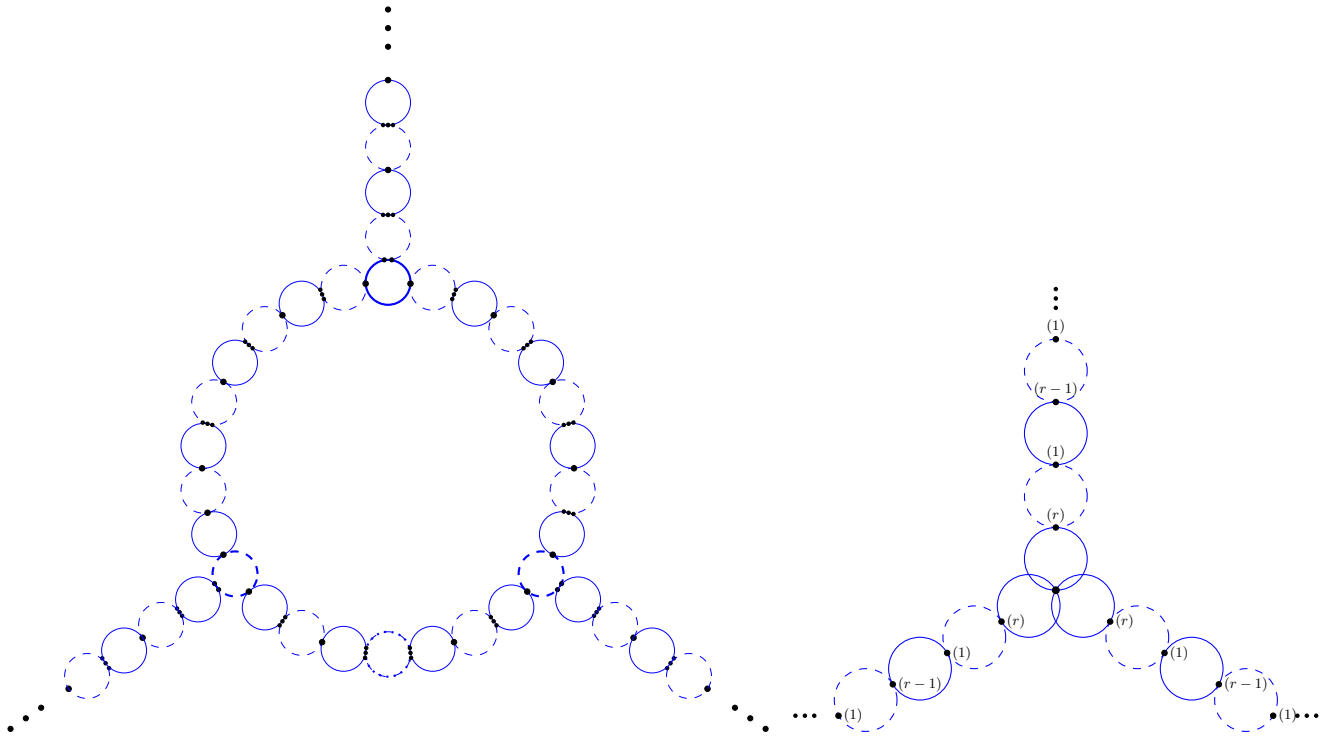
Restricting to solutions of diameter 1 will force a variable cycle to be in one of two possible alternating states, corresponding to the variable's truth value. The number of circles can be chosen sufficiently large so that any other possible solution, i.e., one involving points from *three separate* locations would, absent connection gadgets, have cost arbitrarily to 2. The same applies to the chains connecting variable gadgets to clause gadgets. Also, if none of a clause's three incoming chains is in the on state, correcting this at the clause gadget would require using a cluster of diameter 2. Most of our attention, therefore, will be focused on designing connection gadgets so that the minimum cost of diverging from the a consistent state in or around a connection gadget will be as close to 2 as possible.

The locations within the variable gadget where chains attach are chosen based on the numbers of appearances the variable has of each type, i.e., appearing in a clause negated or non-negated. Without loss of generality, a variable appears in either two or three clauses, including both negated and non-negated appearances, i.e., either one of each type or one of one type and two of the other. In the case of two appearances, the variable cycle is constructed using an even number of circles, with the chains attaching at circles of opposite parities (shown as solid and dashed in Fig. 1(a)), with the result that for each connection, the connection circle (shown bold in Fig. 1(a)) shares a single point each with its left and right neighbor circles, each of which shares $r - 1$ points with their other neighbor circles, and so on. That is, the alternation pattern of single-point locations and $r - 1$-point locations on one side of a connection is the mirror image of the opposite side's pattern.

In the case of three appearances, the variable gadget contains an odd number of circles, with (say) positive appearance(s) attaching at solid circle(s) and negative appearance(s) attaching at dashed circle(s). In order to permit the sym-

*Lehman College and The Graduate Center, City University of New York. This work was supported by NSF-INSPIRE award #1547205 and a CUNY Junior Faculty Research Award (J-FRASE) funded by the Sloan Foundation.

[1]Note that 3SAT here is intended to mean that each clause contains *at most* three distinct literals, not *exactly* three distinct literals. In the latter case, the restriction to three appearances is no longer NP-hard [2].



(a) Variable gadget. (b) Clause gadget.

Figure 1: Detailed description of variable gadget (a) for a variable appearing in three clauses: The two chains at the bottom correspond to two appearances of the same type (i.e., negated or not), and the chain at the top represents an appearance of the opposite type. Points shown in triplicate represent $r-1$ points; points shown in duplicate represent $r-2$ points. At each of the three connection points, the bold circle corresponds to C_1 from the connection gadget. The bold dashed/dotted circle at the bottom center is the location where the 1 versus $r-1$ parity flips (not needed for variables with only two appearances).

metric alternating patterns on the both sides of the two same-type connections, we modify one circle lying between those two connections to have $r-1$ points on both sides (shown in dashed/dotted bold at the bottom of Fig. 1(a)). Note that this $(2r-2)$-point circle does not provide any opportunity to have an inconsistency of cost less than 2.

If the underlying SAT formula is unsatisfiable, then any solution to the resulting r-gather instance must contain an inconsistency. That is, there must come a point where a cluster contains points drawn from multiple different unit disks. Therefore the only other possible location where an inconsistency could occur incurring cost less than 2 would be at a connection gadget. Thus we may assume without loss of generality that each the three chains meeting at a connection (two segments within the variable gadget and one going to the clause) are in consistent state, with the only possible inconsistency occurring within the vicinity of the connection gadget.

3. THE CONNECTION GADGET

To frame the successful operation of the connection gadget in terms of satisfying a constraint, let α_i be a boolean representing the truth value of the *literal* (v or $\bar{v}$) for a variable v's ith clause appearance, or equivalently a boolean that is true if the corresponding chain is in the on state (corresponding, in the top chain in Fig. 1(a), to the solid circles in that chain being on). Let β_i^L, β_i^R be booleans corresponding to the left and right segments of v's gadget, respectively, on either side of the connection gadget for v's ith appearance. Note that β_i^R and β_{i+1}^L refer to the same segment of the variable gadget, but their truth values have different meanings: β_i^R being true means that that segment's state is consistent with v's ith chain being on, whereas β_{i+1}^L being true means that that segment's state is consistent with v's $i+1$st chain being on. Chains i and $i+1$ could correspond to opposite kinds of literals.

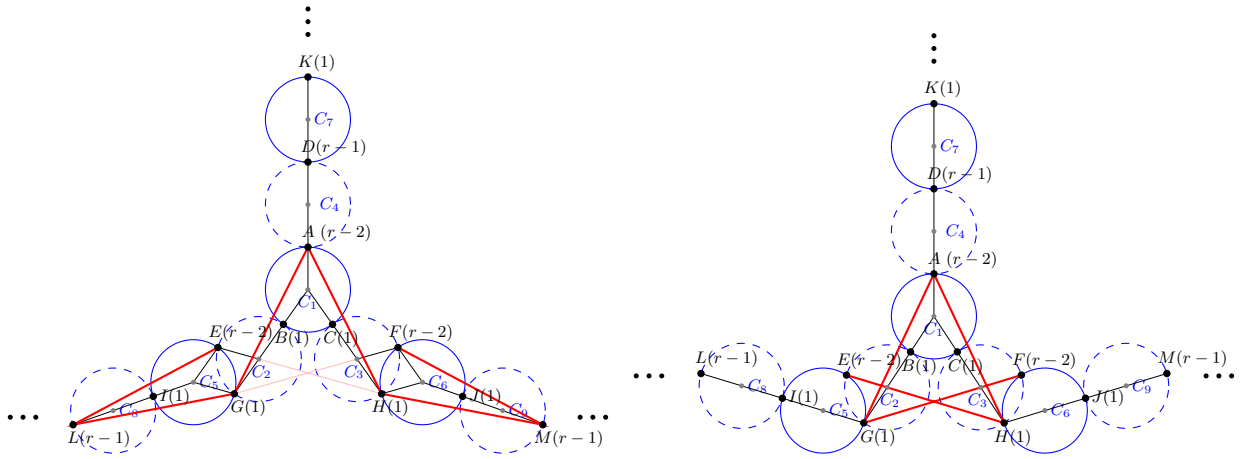
Intuitively we expect that in a consistent solution we will always satisfy $\alpha_i = \beta_i^L = \beta_i^R$. We now argue that it will suffice to analyze the cost of violating a weaker constraint than this, specifically $\alpha_i \to (\beta_i^L \wedge \beta_i^R)$.[2]

LEMMA 1. *If $\alpha_i \to (\beta_i^L \wedge \beta_i^R)$ is satisfied at every connection i (of each variable gadget) in a solution of cost strictly less than 2, then the formula is satisfiable.*

PROOF. Suppose there is a solution of cost strictly less than 2. We will argue that if the formula is not satisfiable, then the specified implication must be violated at some connection.

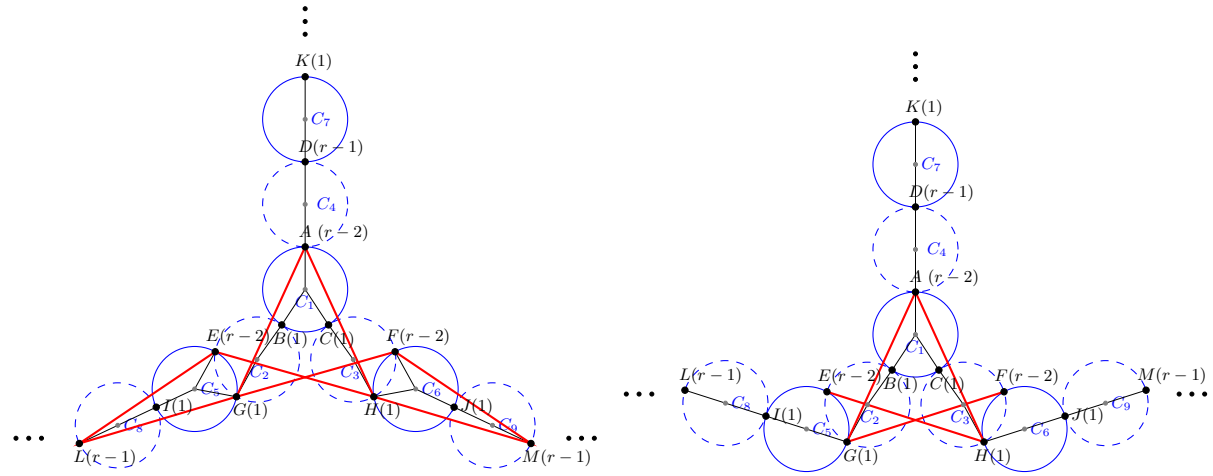
A feasible r-gather solution must cluster all points. Since the only possible clusters of cost less than 2 that can cover

[2]This constraint can be interpreted as being equivalent to the "SPLIT vertex" of Constraint Logic [1].



(a) $r = 4$. (b) $r \geq 5$.

Figure 2: Connection gadgets.



(a) $r = 4$. (b) $r \geq 5$.

Figure 3: Refined connection gadgets.

a clause point are those of its chains, at least one of those chains must at least start out in the on state. And since an inconsistency within a chain would cost 2, that chain must continue in the on state until it reaches a variable connection point.

For each such chain, interpret the fact that it is clustering a clause point as a message claiming that the corresponding literal is true. The formula being unsatisfiable means that there must be some variable v about which two clause points are receiving contradictory messages.

First, suppose v has two conflicting connection chains turned on, say, positive c_1 and negative c_2. Then if c_1's implication is satisfied, that causes v's two segments to both be in a state conflicting with c_2's implication.

Second, suppose v has three chains with at least two of them on and conflicting, say, one positive chain c_1 and two negative chains c_2, c_3, where c_1 and c_3 are on. With c_2 off there is no necessity that β_2^L and β_2^R be the same, permitting c_1's right segment to be compatible with it, and c_3's left segment to be compatible with it. But if c_1's implication is satisfied, then v's third segment, i.e., the one between c_3 and c_1, will be in a state conflicting with c_3's implication. □

Lemma 2. *Suppose that, for the r-gather instance constructed in this reduction, a violation of the implication $\alpha_i \to (\beta_i^L \wedge \beta_i^R)$ at any connection gadget would imply that a solution cost of at least c for some c. Then this restriction of r-gather is NP-hard to approximate with factor better than $\min\{c, 2\}$.*

Proof. One direction of the reduction is clear: if the formula is satisfiable, then the resulting r-gather instance has cost 1.

Conversely, suppose the formula is not satisfiable. Then the previous lemma tells us that any solution to the resulting r-gather instance must either have cost at least 2 or must have some $\alpha_i \to (\beta_i^L \wedge \beta_i^R)$ violation. □

We now analyze the cheapest possible way for $\alpha_i \to (\beta_i^L \wedge \beta_i^R)$ to be violated.

The first pair of approximation lower bounds we obtain involve the *Chebyshev polynomial of the first kind* $T_n(z)$, which can be defined by the identity $T_n(z) = \cos(n \operatorname{acos}(z))$ [3].

THEOREM 3.1. *r-gather with $r = 4$ is NP-hard to approximate with factor better than* $\sqrt{2.5 + 1.5 \cdot T_{\frac{1}{2}}(\frac{1}{3})} \approx 1.9299$.

PROOF. The construction in Fig. 2(a) is drawn in such a way that the angles BC_1C, EC_2G, and FC_6H all equal $\operatorname{acos}(\frac{1}{3}) \approx 70.529$. This results in all the line segments drawn in thick red having the same length: $\sqrt{2.5 + 1.5 \cdot \cos(\frac{1}{2}\operatorname{acos}(\frac{1}{3}))} \approx 1.9299$. The line segments GF, EH, AI and AJ are all *longer* than this.

Assume C_7 is on, and assume we never pay for a red line.

Then point A cannot be clustered with either G or H.

Since G cannot be clustered with F, H cannot be clustered with E, and E cannot be clustered with G, the locations A, G, H, E, F must be appear in at least three different clusters. (In fact G and H must appear in different clusters.)

L and M are both at least red distance away from each of A, G, H, E, and F.

Therefore there *at most* $3r$ points available (i.e., if none of C_5's points is clustered with C_8, and none of C_6's with C_9) to be clustered into at least 3 clusters, which requires that we turn C_5 and C_6 (and C_1) on, i.e., set the variable to the state consistent with satisfying the clause C_7 leads to. □

THEOREM 3.2. *r-gather with $r \geq 5$ is NP-hard to approximate with factor better than* $\sqrt{2.75 + \frac{\sqrt{33}}{4} \cdot T_{\frac{1}{3}}(\frac{-139}{33^{1.5}})} \approx 1.9372$.

PROOF. The construction in Fig. 2 is drawn with angle BC_1C equal to twice the value of $\theta = \operatorname{acos}\left(\frac{\sqrt{33}}{6} T_{\frac{1}{3}}(\frac{-139}{33^{1.5}}) + \frac{1}{6}\right) \approx 33.3721$. This angle results in AG's length $\sqrt{2.5 + 1.5\cos\theta}$ and GF's length $\sqrt{3.25 - 3\cos(2\theta)}$ (and all other red lines) being equal: $\sqrt{2.5 + 1.5\cos\theta} = \sqrt{2.75 + \frac{\sqrt{33}}{4} \cdot T_{\frac{1}{3}}(\frac{-139}{33^{1.5}})} \approx 1.9372$. Line segment EF is *longer* than this.

Assume C_7 is on, and assume we never pay for a red line.

Then point A cannot be clustered with either G or H.

There are only two points, B and C, that are in less than red distance from both G and H. Therefore G and H must appear in different clusters as well.

L and M are both greater than red distance away from each of A, G, and H.

Therefore there *at most* $3r$ points available (i.e., if none of C_5's points is clustered with C_8, and none of C_6's with C_9) to be clustered into at least 3 clusters, which requires that we turn C_5 and C_6 (and C_1) on, i.e., set the variable to the state consistent with satisfying the clause C_7 leads to. □

Now we strengthen the approximation lower bounds further, by considering a slightly more general class of drawings of the connection gadget. For both the $r = 4$ and $r \geq 5$ constructions, we relax the assumption that some points in the drawing are collinear. The angles chosen in the following two constructions were found by computer search.

THEOREM 3.3. *r-gather with $r = 4$ is NP-hard to approximate with factor better than* ≈ 1.938443.

PROOF. The argument is the same, about a slightly modified version of the configuration from Fig. 2(a). In Fig. 3(a), points B, C_2, and G are not collinear but bend slightly. (L, G, F are *not* collinear.) This permits the angle BC_1C to be slightly larger while maintaining equal lengths among all red lines. The following angles permitting this were found numerically: $BC_1C \approx 68.9248$ and $GC_2B \approx 174.4204$. □

THEOREM 3.4. *r-gather with $r \geq 5$ is NP-hard to approximate with factor better than* ≈ 1.938727.

PROOF. The argument is again the same, about a slightly modified version of the configuration from Fig. 2. In Fig. 3, points B, C_2, and G are not collinear but bend slightly. This permits the four red lines from Fig. 2 to grow slightly longer, with the two new red lines also now achieving the same length. The following angles permitting this were found numerically: $BC_1C \approx 68.7549$ and $GC_2B \approx 174.4582$. □

4. DISCUSSION

As noted above, the hardness of approximation factors given for $r = 4$ and $r \geq 5$ were found via computer search for the angle in the construction that would maximize the distances that the factors result from. Given this, it seems especially unlikely that these are the true final values. Intriguingly, however, the best hardness of approximation known for the $r = 3$ case remains $\sqrt{13}/2 \approx 1.802$. (That reduction is via Planar Circuit SAT rather than Planar 3SAT, because "cheating" at the clause costs only $\sqrt{3}$ when $r = 3$ [4].) The problem of finding a better than 2-approximation, even in the case of $r = 3$, remains open.

5. REFERENCES

[1] R. A. Hearn and E. D. Demaine. *Games, Puzzles, and Computation*. CRC Press, 2009.

[2] S. Tippenhauer. On planar 3-SAT and its variants. Master's thesis, Freien Universität Berlin, 2016.

[3] E. W. Weisstein. Chebyshev polynomial of the first kind. *MathWorld–A Wolfram Web Resource*. http://mathworld.wolfram.com/ChebyshevPolynomialoftheFirstKind.html.

[4] J. Zeng, G. Telang, M. P. Johnson, R. Sarkar, J. Gao, E. M. Arkin, and J. S. Mitchell. Mobile r-gather: Distributed and geographic clustering for location anonymity. In *ACM MOBIHOC*, 2017.