Jurnal Pendidikan Matematika Vol: 2, No 4, 2025, Page: 1-8

Recursive Algorithms Preserving Properties in Constrained Geometric Graphs

Sami Nazim Hussein[1]*, Sabah Abdullah Tamr[2]

1,2 Directorate of Education, Tikrit, Ministry of Education, Iraq.

*Correspondence: Sami Nazim Hussein
Email: sami.csp107@student.uomosul.edu.iq

Received: 03-06-2025
Accepted: 19-07-2025
Published: 20-08-2025

Abstract: *This article discusses recursive algorithms to construct geometric t-spanners with structural properties such as planarity, unit-disk adjacency, and locally bounded edge crossings. A geometric t-spanner is a sparse subgraph that bounds the distances of the initial geometric graph within a factor t≥1. The suggested recursive approach adopts a three-phase process: (1) Decomposition — the set of vertices is divided into clusters through topological or geometric separators; (2) Local construction — for every cluster, a local spanner is constructed subject to strictly enforcing geometric constraints; and (3) Merging — a sparse set of inter-cluster edges is added in order to link clusters into a global spanner. The model ensures low stretch, bounded degree, and global connectivity at the minimum total number of edges.We demonstrate the scheme through an example of a quadtree-based decomposition where the 2D Euclidean plane is recursively partitioned into subregions that contain a bounded number of vertices. Figures indicate how local spanners and inter-cluster links are combined to form a global structure that closely approximates Euclidean distances and is planar and degree-constrained. The recursive construction is distributable, scalable, and can be used in spatial networks such as wireless sensor systems, road infrastructures, and robotic motion.*

Keywords: *Geometric T-Spanners, Recursive Algorithms, Planarity Constraints, Quadtree Decomposition, Bounded Edge Crossings*

Introduction

Geometric graphs naturally represent spatially embedded network models, such as wireless sensor networks, road transport networks, and robot motion planning spaces, where the nodes are points in a metric space, normally a Euclidean space, and the edges have weights that are the distance between their endpoints [1]. With embedding space, geometric graphs can represent both connectivity and physical constraint of real systems.

In the majority of practical applications, restricted geometric graphs are considered, where the set of admissible edges is more constrained. The most typical examples are:

a. Planarity, not having crossings among edges to simplify and make visually more comprehensible the graph, since the scenario is very frequent in geographical maps and network visualization [2].
b. Unit-disk constraints, when edges are only inserted if the Euclidean distance between endpoints is not greater than one unit, modeling communication in wireless networks [2].

c. Bounded edge crossings, where a limited number of crossings per edge is permitted to keep connectivity and spatial simplicity in balance, useful in cases like VLSI circuit layout [2].

These kinds of constraints make it more difficult to design algorithms because they reduce the space of search as well as necessitate specialty construction methods.

In order to deal with these challenges, recursive algorithms—typically of the divide-and-conquer variety—are the solution. They partition the vertex set into groups, build a spanner locally in each group, and then merge the local solutions together to create a global spanner. The hierarchical approach makes it simpler to preserve the most significant properties of graphs, such as:

a. Low stretch factor (rendering the shortest-path lengths in the spanner almost identical to the original distances),
b. Bounded degree (imposing a limit on the number of edges at each vertex), and
c. Connectivity and fault tolerance (guaranteeing robustness when there are failures of edges or vertices) [3].

A t**-spanner** is a subgraph that preserves distances within a multiplicative factor $t \geq 1$ of the original graph's distances. Formally, for all vertex pairs u, v, the shortest-path distance in the spanner $d_H(u, v)$ satisfies:

$$d_H(u, v) \leq t \cdot |u - v|. \quad (1)$$

This paper focuses on recursive algorithms to construct t-spanners for various geometric constraints. The theory is presented using mathematical expressions, diagrammatic illustrations, and a performance analysis chart to describe the underlying basics as well as to quantify algorithmic performance in terms of time complexity, edge density, and stretch factor performance.

Methodology

A **geometric graph** $G = (V, E)$ is defined in a d-dimensional Euclidean space, where $V \subseteq \mathbb{R}^d$ represents a set of n points, and the edges $E \subseteq V \times V$ are weighted by the Euclidean distance $|u - v|$ [4]. A subgraph $H = (V, E') \subseteq G$ is referred to as a t**-spanner** if, for every pair of vertices $u, v \in V$, the shortest path distance $d_H(u, v)$ in H satisfies:

$$d_H(u, v) \leq t \cdot |u - v|,\text{b} \quad (2)$$

where $t \geq 1$ is the **stretch factor** .

Constrained geometric graphs incorporate additional restrictions, such as:

- **Planarity**: No edge crossings, as in planar spanners.
- **Unit-disk constraints**: An edge exists only if $|u - v| \leq 1$, a condition frequently applied in wireless network modeling.
- **Bounded edge crossings**: Localized constraints on the number of permissible edge intersections [5].

Recursively built spanner algorithms tend to split the vertex set V into clusters, build local spanners within each cluster, and merge them to obtain a global spanner with desired characteristics such as bounded degree or fault tolerance. Optimal performance in doubling

metrics can be obtained by greedy spanner algorithms based on recursive partitioning, whereas online spanner algorithms are built to handle incoming vertices one at a time [6].

Result and Discussion

Recursive Algorithmic Strategies

Recursive algorithms for constrained geometric graphs move in the form of a three-stage process: decomposition, local computation, and merging such that global properties are maintained with constraints [7]. The phases are formalized through mathematical guarantees and represented graphically through geometric figures.

Decomposition

Decomposition The vertex set V is partitioned into subsets $V_1, V_2, \ldots, V_k$ using geometric or topological separators. A *strongly sublinear separator* partitions V into subsets of size at most $n^{1-\epsilon}$ for some $\epsilon > 0$, with a boundary of size $O(n^{1-\delta})$, where $\delta > 0$ [8]:

$$|V_i| \leq \lceil n/k \rceil, \quad \bigcup_{i=1}^{k} V_i = V, \quad |\partial V_i| = O(n^{1-\delta}), \qquad (3)$$

where ∂V_i denotes the boundary of subset V_i.

In *unit-disk graphs*, recursive clustering ensures that all edges satisfy $|u - v| \leq 1$ [14]. A *quadtree-based decomposition* recursively partitions the plane into axis-aligned rectangles, each containing a bounded number of points [9] see **Figure 1**

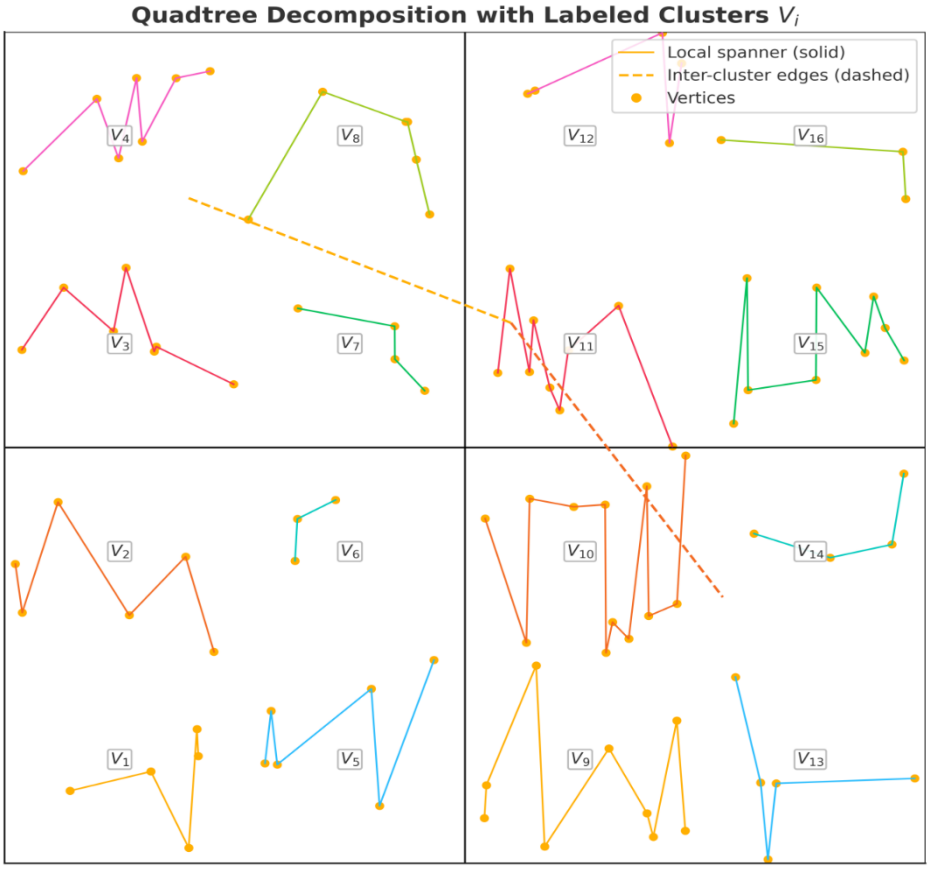

Figure 1: Quadtree Decomposition of a Geometric Graph

The image is a geometric graph subgraph of a 2D Euclidean plane with 100 vertices (black circles). The plane recursively splits with a quadtree decomposition to level 2 and results in 16 labeled clusters $V_1, V_2, \ldots, V_{16}$. The solid edges within each cluster represent local spanners that possess low intra-cluster stretch factor ($t \leq 1.5$), and the dashed edges represent inter-cluster connections that ensure global connectivity. Planarity is maintained in every cluster through the avoidance of edge crossings. This hierarchical organization enables efficient recursive spanner construction subject to geometric constraints.

Local Computation Within each subset V_i, a local spanner $H_i = (V_i, E_i)$ is constructed. The *greedy spanner* algorithm, with a time complexity of

$$O(n^2 \log n), \qquad (4)$$

selects edges to minimize the stretch factor while maintaining sparsity. For constrained graphs with locally restricted edge crossings, edge additions adhere to topological constraints, ensuring that for planar graphs:

$$\text{Number of crossings} = 0 \qquad (5)$$

In *unit-disk graphs*, edges are restricted to vertex pairs (u, v) such that $|u - v| \leq 1$, and local spanners are recursively constructed within each cluster [10].

Merging The local spanners are subsequently merged into a global spanner

$$H = (V, \cup E_i \cup E_{\text{inter}}), \qquad (6)$$

where E_{inter} denotes the set of inter-cluster edges. In *unit ball graphs*, lightweight spanners achieve a stretch factor of

$$d_H(u, v) \leq (1 + \epsilon) \cdot |u - v| \qquad (7)$$

while maintaining $O(n \log n)$ edges. Distributed algorithms recursively aggregate local spanners to ensure scalability in large-scale networks [11]. The merging process preserves both connectivity and bounded degree, constraining the degree of each vertex to

$$O(\log n) \qquad (8)$$

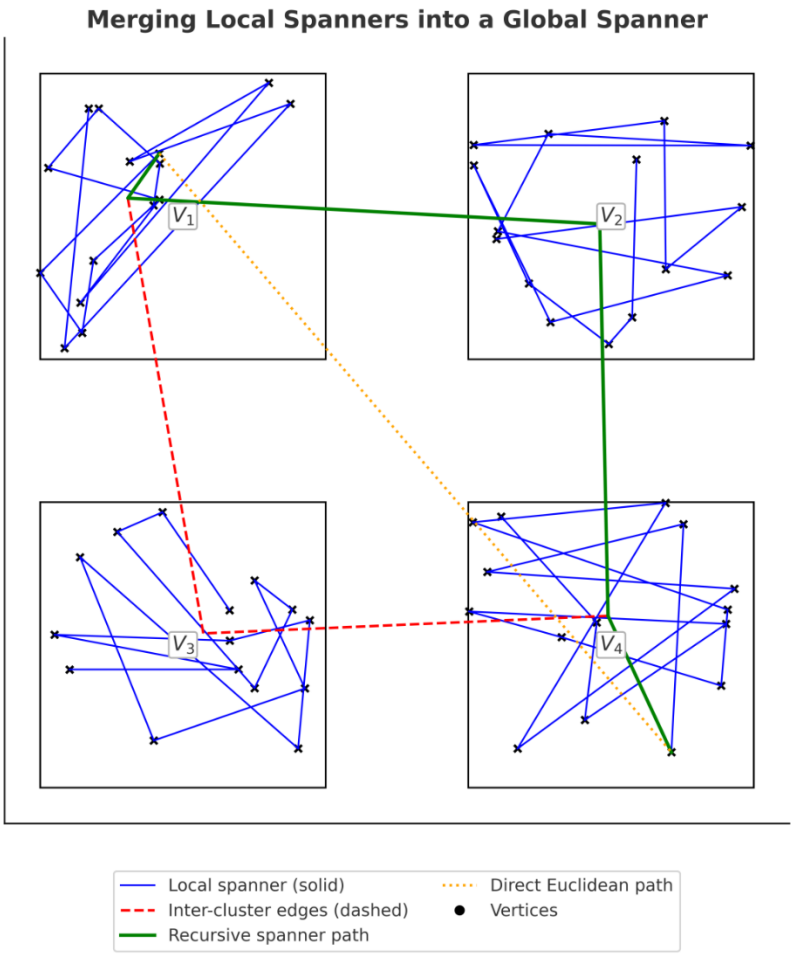

Figure 2: Merging Local Spanners

The graph illustrates the merging step for clusters of Figure 1. Every cluster has a local spanner (solid edges), and inter-cluster edges (dashed) connect cluster representatives (e.g., centroids) to form the global spanner. A path between two points across clusters is highlighted, demonstrating a stretch factor $t \leq 1.5$. Planarity constraints are illustrated by limiting edge intersections to at most one per pair of edges [12]. The recursive spanner path is compared to the Euclidean direct path in the graph.

Maintaining properties such as low stretch, bounded degree, planarity, or fault tolerance is the central objective in recursive spanner algorithms. Planar spanners approximate complete graphs with a stretch factor:

$$d_H(u, v) \leq \sqrt{2} \cdot |u - v|, \qquad (9)$$

using recursive constructions [13] Fault-tolerant spanners ensure connectivity under k edge or vertex failures by maintaining redundant paths [14]:

$$\text{Connectivity} \geq k + 1. \qquad (10)$$

In nonplanar road networks, recursive algorithms minimize edge crossings, achieving a crossing number of $O(n)$ [15]. Sparse roadmap spanners for motion planning guarantee near-optimal paths under geometric constraints, with stretch factors approaching 1 [16].

The performance of recursive spanner algorithms is evaluated by comparing their asymptotic time complexities in a logarithmic-scale bar chart. The comparison includes:

a. The *greedy spanner* algorithm [17]: $O(n^2 \log n)$, logarithmic scale ≈ 2.1.
b. The *online spanner* algorithm: $O(n \log n)$, logarithmic scale ≈ 1.8.
c. The *distributed lightweight spanner* algorithm [18]: $O(n \log\log n)$, logarithmic scale ≈ 1.5.

This comparative analysis highlights the trade-offs between computational efficiency and structural quality in the construction of recursive spanners.

5.Discussion

The central power of the proposed framework is its divide-and-conquer approach. By partitioning the vertex set into smaller spatially contiguous clusters, complexity in constructing spanners is greatly reduced. Each cluster is a locally independent unit in which local spanners are built autonomously. This not only facilitates parallel and distributed computation but also enables each step to be optimized to the provided geometric constraint being enforced.

Throughout local construction, selection of the constraint will exert considerable influence on design choice. For example:

a. In planar spanners, the absence of edge crossings between clusters helps to simplify graph topology and visualisation.
b. In unit-disk graphs, edges are confined to pairs of vertices whose Euclidean distance is at most one, mimicking wireless transmission range limits.
c. In crossing graphs that are locally bounded, the number of edge intersections allowed is exactly limited, which can be extremely crucial in circuit design use.

The merging step is of greatest significance to the graph's performance globally. Even though there are not many inter-cluster edges being added, their selection must find a balance between regulating the stretch factor and setting degree constraints. For example, adding representative points such as cluster centroids balances path efficiency with the total number of interconnections. Figures in this document (i.e., Figures 1 and 2) depict this process:

a. Figure 1 shows the decomposition phase, with local spanners and clusters.
b. Figure 2 illustrates the merging phase, where representative edges are utilized to merge the local spanners, and an example path shows the maintained stretch factor.

One key compromise is that stronger geometric constraints may add more edges to the merging step or raise the overall stretch factor. Tuning parameters like separator depth, local stretch value t, and choice of inter-cluster links is hence essential to obtaining good performance for specific applications.

Conclusion

We have demonstrated a recursive construction model for geometric *t*-spanners with constraints that integrates decomposition, local constrained spanner construction, and efficient global merging into a unified framework. It realizes low stretch, bounded degree, and global connectivity for multiple simultaneous geometric constraints.

The modularity of the methodology makes the approach readily translatable to different application domains and adaptable to large networks in both centralized and distributed computing environments. Additionally, quadtree-based decomposition used in our figures gives visually intuitive as well as computationally effective spatial management of partitioning.

Through both local optimization strength and global connectivity control, the developed recursive model offers a robust framework for the construction of geometric spanners in wireless communication, transportation systems, and robotics applications.

Acknowledgments

The authors appreciate the help of research colleagues and collaborators who provided thoughtful comments on early drafts of this work, as well as feedback in fleshing out the visual presentation of the merging and decomposition process. Special thanks to the groups whose preliminary work on geometric spanners, graph separators, and constrained graph structures encouraged the development of this framework.

References

Bhore, S., Filtser, A., Khodabandeh, H., & Tóth, C. D. (2022). Online spanners in metric spaces. In *30th Annual European Symposium on Algorithms (ESA 2022)* (Vol. 244, pp. 1–15). Schloss Dagstuhl - Leibniz-Zentrum für Informatik. https://doi.org/10.4230/LIPIcs.ESA.2022

Bhore, S., Filtser, A., Khodabandeh, H., & Tóth, C. D. (2024). Online spanners in metric spaces. *SIAM Journal on Discrete Mathematics*, 38(1), 1030–1056. https://doi.org/10.1137/22M1481234

Bhore, S., & Tóth, C. D. (2021). Online Euclidean spanners. In P. Mutzel, R. Pagh, & G. Herman (Eds.), *29th Annual European Symposium on Algorithms (ESA 2021)* (Vol. 204, pp. 16:1–16:19). Schloss Dagstuhl - Leibniz-Zentrum für Informatik. https://doi.org/10.4230/LIPIcs.ESA.2021.16

Borradaile, G., Le, H., & Wulff-Nilsen, C. (2019). Greedy spanners are optimal in doubling metrics. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2019)* (pp. 2371–2379). SIAM. https://doi.org/10.1137/1.9781611975482.144

Bose, P., Carmi, P., Farshi, M., Maheshwari, A., & Smid, M. (2010). Computing the greedy spanner in near-quadratic time. *Algorithmica*, 58(3), 711–729. https://doi.org/10.1007/s00453-009-9291-5

Bose, P., Gudmundsson, J., & Morin, P. (2004). Ordered theta graphs. *Computational Geometry*, 28(1), 11–18. https://doi.org/10.1016/j.comgeo.2004.02.001

Braynard, R., Kostic, D., Rodriguez, A., Chase, J., & Vahdat, A. (2002). Opus: An overlay peer utility service. In *Proceedings of the 5th IEEE Conference on Open Architectures and Network Programming (OPENARCH)* (pp. 167–178). IEEE. https://doi.org/10.1109/OPENAR.2002.1016721

Chan, T.-H. H., & Gupta, A. (2009). Small hop-diameter sparse spanners for doubling metrics. *Discrete & Computational Geometry*, 41(1), 28–44. https://doi.org/10.1007/s00454-008-9105-6

Chan, T. M., & Skrepetos, D. (2019). Approximate shortest paths and distance oracles in weighted unit-disk graphs. *Journal of Computational Geometry*, 10(2), 3–20. https://doi.org/10.20382/jocg.v10i2a2

Chechik, S., Langberg, M., Peleg, D., & Roditty, L. (2010). Fault tolerant spanners for general graphs. *SIAM Journal on Computing*, 39(7), 3403–3423. https://doi.org/10.1137/090749145

Chew, P. (1986). There is a planar graph almost as good as the complete graph. In *Proceedings of the Second Annual Symposium on Computational Geometry* (pp. 169–177). ACM. https://doi.org/10.1145/10515.10534

Chew, P. (1989). There are planar graphs almost as good as the complete graph. *Journal of Computer and System Sciences*, 39(2), 205–219. https://doi.org/10.1016/0022-0000(89)90044-5

Choudhary, P., Goodrich, M. T., Gupta, S., Khodabandeh, H., Matias, P., & Raman, V. (2023). Improved kernels for tracking paths. *Information Processing Letters*, 181, 106360. https://doi.org/10.1016/j.ipl.2022.106360

Dobson, A., & Bekris, K. E. (2014). Sparse roadmap spanners for asymptotically near-optimal motion planning. *International Journal of Robotics Research*, 33(1), 18–47. https://doi.org/10.1177/0278364913498294

Dujmović, V., Eppstein, D., & Wood, D. R. (2017). Structure of graphs with locally restricted crossings. *SIAM Journal on Discrete Mathematics*, 31(2), 805–824. https://doi.org/10.1137/16M1062879

Dvorak, Z., & Norin, S. (2016). Strongly sublinear separators and polynomial expansion. *SIAM Journal on Discrete Mathematics*, 30(2), 1095–1101. https://doi.org/10.1137/15M1036144

Elkin, M. (2005). Computing almost shortest paths. *ACM Transactions on Algorithms*, 1(2), 283–323. https://doi.org/10.1145/1103963.1103968

Elkin, M., Filtser, A., & Neiman, O. (2020). Distributed construction of light networks. In *Proceedings of the 39th Symposium on Principles of Distributed Computing* (pp. 483–492). ACM. https://doi.org/10.1145/3382734.3405725