Stable Roommates and Geometric Spanners

Paz Carmi* Lilach Chaitman†

Abstract

In this paper we devise a new geometric spanner based on a generalization of the known Stable Roommates algorithm. This spanner is on the "path" between the Yao graph and Yao-Yao graph, and as such it can be computed in a distributed manner and has a bounded degree as the Yao-Yao graph, while on the other hand, it has the same stretch factor bound as the Yao graph.

1 Introduction

The Stable Roommates problem is the problem of finding a stable matching for a set X – a matching in which there are no two elements $x, y \in X$ that prefer each other over their matches. More precisely, given a set X of $2n$ people, where each person has ranked all the others with a unique number between 1 and $n-1$ in order of preference, a matching is a set of n pairs $\{x, y\} \subset X$, such that there are no two persons who would both rather have each other over their current matched roommates. If there are no such persons, the matching is called stable.

Geometric representation of the problem has already been introduced in [1], where a participant is represented as a point in a metric space, and his preference list as a sorted list of the other participants by nondecreasing order of their distances from him. They showed how the stable roommates problem can be easily solved in $O(n \log n)$ time. Our geometric problem is a bit different. We approve a point to have more than one roommate as long as they come from different "regions" with respect to the point. Another way to look at it is as a different geometric representation of the generalization of the Stable Roommates problem, where the participants are cones around all the points and the matching is restricted to pairs that contain a common edge. Note there may be cones without a match. The principle guiding the preferences of each cone will be given in the next section.

In [7], Yao has defined the Yao graph Y_k as follows. At each point p, any k equally-separated rays with apex p define k cones of equal angles. In each cone, the shortest edge $\{p, q\}$ among all edges incident to point p (if there are any) is added to Y_k. The resulting graph is treated as an undirected graph and called Yao graph. It is known that the Yao graph for $k > 6$ is a t-spanner (recently shown for $k = 4$ by Bose et al. [3]). In [6], Wang and Li have defined the Yao-Yao graph $Y_{ao}Y_k$ as a subgraph of the directed Yao graph where the directed edges are pruned as follows. At each node p, in each cone only the shortest incoming edge among all the incoming edges is kept. The resulting graph is treated as an undirected graph. Various properties of $Y_{ao}Y_k$ have been established, but the question whether $Y_{ao}Y_k$ is a t-spanner remains open.

Here, we show a spanner that is on the "path" between the Yao graph and the Yao-Yao graph in the sense that it maintains the good properties of both graphs. More precisely, We give a new local algorithm, '*Stable roommates*', for computing a sparse sub-graph of a given unit disk graph. Given a unit disk graph, the Stable roommates algorithm is very simple, fully distributed, performed independently on each node of the unit disk graph and can be computed in $O(n/R * n \log n)$ time, where R is the number of processors in the distributed system. The resulting spanner obtained by this algorithm is a strong t-spanner, has a degree at most k as $Y_{ao}Y_k$, its stretch factor is at most Y_k and it contains the minimum weight spanning tree as its sub-graph. A spanner is strong when for any pair of points $p, q \in P$, the spanning path between p and q only contains edges whose length is at most $|pq|$.

Moreover, we take a step forward towards answering the open problem of constructing a t-spanner with angles greater than a fixed angle θ in $O(n \log n)$ time. With minor changes we achieve an additional property for our t-spanner of having only angles greater than $\theta = 2\pi/k$. The only t-spanner that possesses this property is the greedy t-spanner. However, while our spanner can be computed in a distributed manner in $O(n/R * n \log n)$ time, where R is the number of processors in the distributed system, the most efficient construction of the greedy t-spanner is sequential with time complexity $O(n^2 \log n)$ (see [2]) and is more complicated and less intuitive.

We hope that this result will help to shed some light on the open problem: Is the Yao-Yao graph a t-spanner for constant t?

We find this result quite surprising in the sense that the Stable Roommates idea admits a new t-spanner with

*Department of Computer Science, Ben-Gurion University, Beer-Sheva, Israel.

†Department of Computer Science, Ben-Gurion University, Beer-Sheva, Israel. Research is partially supported by Lynn and William Frankel Center for Computer Science.

such good properties as the Yao-Yao graph on the one hand and on the other hand its stretch factor is easily shown to match the stretch factor of the Yao graph.

2 Stable Roommates t-spanner

In this section we describe an algorithm that computes a bounded degree spanner based on a generalization of the Stable Roommates problem. Given a constant $k \geq 8$ and a set of fixed points P, for each $p \in P$, let $C_p = \{C_{p_0}, C_{p_1}, \ldots, C_{p_{k-1}}\}$ denote a set of k cones labeled in clockwise order, with apex p, and angle $\theta = \frac{2\pi}{k}$. We assume the cones are half open half closed.

The approach we take to build such a spanner is to observe all pairs of points in P and connect two points $p, q \in P$ with an edge iff the corresponding cones $C_{p_i} \in C_p$ and $C_{q_j} \in C_q$ match according to the stable matching definition. Meaning there are no two cones in $\bigcup_{p \in P} C_p$ that prefer each other over their matches. The preference list of each cone C_{p_i} is defined as follows. All cones C_{q_j} such that $\{p, q\} \notin C_{p_i} \cap C_{q_j}$ will not be ranked, since they are forbidden matches. As for the rest, a cone C_{q_j} will be preferred over the cone C_{r_k} if $|pr| > |pq|$, while ties are broken arbitrarily.

Given a constant $k \geq 8$, for each point p, we define a set of k cones with apex $p - C_p$. For achieving a spanner with stretch factor t, one should choose $k = 2\pi/\theta$ such that $t \geq 1/(\cos(\theta) - \sin(\theta))$. In the beginning of the Algorithm (Algorithm 1), all pairs $\{p, q\} \subset P$ are sorted in nondecreasing order of their distances. An edge $\{p, q\}$ is added to G if both p and q *agree* on it. A point p *agrees* on an edge $\{p, q\}$ if the cone $C_{p_i} \in C_p$ that contains $\{p, q\}$ is empty, i.e. $C_{p_i} \cap E = \emptyset$.

Algorithm 1 StableRoommates(P)

Input: A set P of points in the plane.
Output: A *Stable Roommates* t-spanner $G = (P, E)$.

1: Let L be a sorted list of all pairs $\{p, q\} \subset P$ by nondecreasing order of their distances.
2: $E \leftarrow \emptyset$
3: **for** each edge $\{p, q\} \in L$ (* in the sorted order *) **do**
4: Let $C_{p_i} \in C_p$ and $C_{q_j} \in C_q$ be the cones containing $\{p, q\}$.
5: **if** $C_{p_i} \cap E = \emptyset$ and $C_{q_j} \cap E = \emptyset$ **then**
6: $E \leftarrow E \cup \{\{p, q\}\}$

Lemma 1 *The matching implied by the set of edges E in the resulting graph $G = (P, E)$ of Algorithm 1 is a stable matching.*

Proof. Assume on the contrary there is a pair of cones C_{p_i} and C_{q_j} that prefer each other over their matches, C_{r_k} and C_{s_t} respectively. Meaning $\{p, q\} \in C_{p_i} \cap C_{q_j}$ is shorter than $\{p, r\} \in C_{p_i}$ and $\{q, s\} \in C_{q_j}$, however $\{p, r\}$ and $\{q, s\}$ are in E while $\{p, q\}$ is not in E. Since $\{p, r\}$ and $\{q, s\}$ were added to E, necessarily $C_{p_i} \cap E$ and $C_{q_j} \cap E$ were empty when the algorithm observed them. Since $\{p, q\}$ is shorter than $\{p, r\}$ and $\{q, s\}$, it was observed by the algorithm before $\{p, r\}$ and $\{q, s\}$, and therefore $C_{p_i} \cap E$ and $C_{q_j} \cap E$ were empty at that time and $\{p, q\}$ should have been added to E, in contradiction to the assumption. □

For a set of points P and two points $p, q \in P$, let D_p denote the disk centered at p and containing q on its boundary, and let D_q denote the disk centered at q and containing p on its boundary.

Observation 1 *If an edge $\{p, q\}$ is contained in the minimum spanning tree of the complete euclidean graph over P, then $D_p \cap D_q$ is empty of points.*

Lemma 2 *The resulting graph of Algorithm 1 G contains the minimum weight spanning tree as its subgraph.*

Proof. Let $\{p, q\}$ be an edge in the minimum spanning tree. We will now show it is also contained in G. Consider the cones $C_{p_i} \in C_p$ and $C_{q_j} \in C_q$ containing $\{p, q\}$. Let p' and q' be the two intersection points of the boundaries of D_p and D_q. Since the angle between $\{p, q\}$, and each one of $\{\{p, p'\}, \{p, q'\}, \{q, p'\}, \{q, q'\}\}$ is $\pi/3 > \theta$, if there was a point $r \in C_{p_i}$ such that $|rp| < |pq|$ then $r \in D_p \cap D_q$, and the same holds for C_{p_i}. However, by Observation 1, since $\{p, q\}$ is in the minimum spanning tree, $D_p \cap D_q$ is empty of points. Therefore p and q must have "agreed" on adding $\{p, q\}$ to G. □

2.1 Bounded degree

Lemma 3 *The degree of the graph $G = (P, E)$ constructed by Algorithm 1 is bounded above by k.*

Proof. k cones C_p are defined for each point $p \in P$. Consider the edges E_p incident to p that are added to E during Algorithm 1. Each edge $e \in E_p$ is added to E only if the cone in C_p containing e is empty. Thus $|E_p| \leq |C_p| = k$. □

2.2 Spanning ratio

In this section we show that the spanning ratio of the resulting subgraph is bounded. We use Lemma 6.4.1 from [5] by Giri Narasimhan and Michiel Smid that argues the following: Let t, θ, and w be real numbers, such that $0 < \theta < \pi/4$, $0 \leq w < (\cos(\theta) - \sin(\theta))/2$ and $t \geq 1/(\cos(\theta) - \sin(\theta) - 2w)$. Let p, q, r, and s be points in $\Re^d$, such that:

1. $p \neq q$, $r \neq s$,
2. $\angle(pq, rs) \leq \theta$,

3. $|rs| \leq |pq|/(\cos(\theta))$, and
4. $|pr| \leq w|rs|$.

Then $|pr| < |pq|$, $|sq| < |pq|$, and $t|pr| + |rs| + t|sq| \leq t|pq|$.

Lemma 4 *The resulting graph of Algorithm 1 is a strong t-spanner with $t = \frac{1}{\cos(\theta)-\sin(\theta)}$.*

Proof. Let $G = (P, E)$ be the output graph of Algorithm 1. Let $\delta_G(p, q)$ denote the length of the shortest path between p and q in G. To prove the Lemma we show that for every pair $\{s, p\} \in P$, $\delta_G(s, p) \leq t|sp|$, and the t-path connecting them contains only shorter edges than $|sp|$. We prove the above by induction on the rank of the distance $|\{s, p\}|$, i.e., the place of $\{s, p\}$ in a nondecreasing distances order of all pairs of points in P.
Base case: Let s, p be the closest points in P. Then the edge $\{s, p\}$ has been added to E during the first iteration of the loop in step 3, and therefore $\delta_G(s, p) = |sp|$ and clearly this path does not contain edges longer than $|sp|$.
Induction hypothesis: Assume for every pair $\{r, q\} \in P$ with distance shorter than $|sp|$, the Lemma holds.
The inductive step: If $\{s, p\} \in E$, we are done. Otherwise, w.l.o.g. assume $\{s, p\} \in C_{s_i}$ and $\{s, p\} \in C_{p_j}$; then, there exists either an edge $\{s, r\} \in C_{s_i} \cap E$, such that $|sr| \leq |sp|$, or an edge $\{p, r\} \in C_{p_j} \cap E$, such that $|pr| \leq |sp|$. Assume w.l.o.g. that there exists an edge $\{s, r\} \in C_{s_i} \cap E$, such that $|sr| \leq |sp|$. Since $\angle(sr, sp) \leq \theta \leq \pi/4$, $|rp| \leq |sp|$. By the induction hypothesis we get $\delta_G(r, p) \leq t|rp|$, and the t-path connecting r and p contains only edges shorter than $|rp|$. Applying Lemma 6.4.1 with $w = 0$, we get $\delta_G(s, p) \leq t|sp|$ and the corresponding t-path (goes through $\{s, r\}$) contains only edges shorter than $|sp|$. □

Observation 2 *Algorithm 1 can be applied on* UDG(P)*, while observing the edges of* UDG(P) *instead of all pairs in P, resulting in a t-spanner of the* UDG(P) *with the same properties.*

Proof. By Lemma 4 the resulting t-spanner of Algorithm 1 is strong t-spanner. Since the existence of an edge $\{s, p\}$ in a UDG(P) implies the existence of an edge between every pair $\{r, q\}$ s.t $|rq| \leq |sp|$, the t-paths connecting edges of UDG(P) constructed by Algorithm 1 applied on P exist also in the resulting graph of Algorithm 1 applied on UDG(P). □

3 Distributed algorithm for Stable Roommates t-spanner

Consider a distributed system with R processors. In this section we show a distributed algorithm with $O(n/R * \log n)$ running time that, given a set of points P and a constant $k \geq 8$, computes the same t-spanner as Algorithm 1 using a localized construction method.

The distributed algorithm chooses the edges according to the same principle as in Algorithm 1, the shortest edge that both cones (corresponding to different endpoints) agree on will be added to E. However, instead of scanning all pairs sequentially, each point will be responsible for the edges incident to it. Every point p will hold a list of all the other points sorted in nondecreasing order of their distances from it $\{q_1, q_2, ..., q_{n-1}\}$ and attempt to add edges $\{\{p, q_1\}, \{p, q_2\}, ..., \{p, q_{n-1}\}\}$ one after another. An attempt to add an edge $\{p, q\} \in C_{p_i} \cap C_{q_j}$ will succeed if all the previous attempts of the other endpoint q to add edges in C_{q_j} failed and it has reached p in its list.

For simplicity of presentation we assumed that all edges in a cone are of different lengths; however, even if this is not the case, we can solve it using a known method in distributed computing such as *wait and notify*. Notice that Step 7 in Algorithm 4 terminates, since we are in the distributed setting.

Algorithm 2 StableRoomates(P)

Input: A set of points P.
Output: A *Stable Roommates* t-spanner $G = (P, E')$.

1: $E' \leftarrow \emptyset$
2: **for** each p in P (* in distributed behavior *) **do**
3: DistributedInit(p)
4: **for** each p in P (* in distributed behavior *) **do**
5: DistributedEdgeSelction(p)

Algorithm 3 DistributedInit(p)

Input: A point $p \in P$.
Output: A sorted list of $P\backslash\{p\}$, $List(p)$.

1: $List(p) \leftarrow$ Sort $P\backslash\{p\}$ by nondecreasing order of their distances from p.

Lemma 5 *The resulting graph $G = (P, E)$ of Algorithm 1 and the resulting graph $G' = (P, E')$ of Algorithm 2 are identical.*

Proof. Similarly to Algorithm 1, Algorithm 2 chooses at most one edge from every cone to be added to E'. Moreover, the edges are chosen by the same principle – the shortest edge whose cones on both sides agree" on is added to E. Therefore, $E = E'$ and $G = G'$. □

Observation 3 *The running time of Algorithm 2 is $O(n/R * n \log n)$.*

Proof. Each $p \in P$ sorts all the other $n - 1$ points in $O(n \log n)$ time. Since it is done on each processor in parallel we have $O(n/R * n \log n)$ sorting time

Algorithm 4 DistributedEdgeSelction(p)

Input: A point $p \in P$.
Output: The spanner edges in E' contributed by p.

1: **while** $(List(p) \neq \emptyset)$ **do**
2: $\{p,q\} \leftarrow TOP(List(p))$
/* $TOP(List(p))$ is the first element of $List(p)$ */
3: Remove $\{p,q\}$ from $List(p)$
4: Let $C_{p_i} \in C_p$ and $C_{q_j} \in C_q$ be the cones containing $\{p,q\}$.
5: **if** $C_{p_i} \cap E' \neq \emptyset$ **then**
6: Go back to 1
7: **while** $(|TOP(List(q))| < |\{p,q\}|)$ **do** (nothing)
/* distributed behavior ensures while-loop termination */
8: **if** $C_{q_j} \cap E' = \emptyset$ **then**
9: $E' \leftarrow E' \cup \{\{p,q\}\}$

of all points. For each $p \in P$ the running time of *DistributedEdgeSelction* method (Algorithm 4) is $O(n)$ since each point waits at most $O(n)$ iterations in the while loop. Thus we get the overall running time is $O(n/R * n \log n)$. □

Theorem 6 *Given a set P of points in the plane, or alternatively a unit disk graph, and a constant $k \geq 8$, a t-spanner based on a generalization of the Stable Roommates problem with the following properties, can be computed in $O(n/R * n \log n)$ time:*

1. *Stretch factor $t = \frac{1}{\cos(\theta)-\sin(\theta)}$, where $\theta = 2\pi/k$.*
2. *Strong t-spanner.*
3. *Contains the minimum spanning tree as its subgraph.*
4. *Bounded degree k.*

3.1 Angles greater than θ

By making minor changes we can achieve a t-spanner whose angles are all greater than θ and still has all the above properties. The only change needed is using *dynamic* cones instead of predefined ones. The boundaries of a *dynamic* cone are defined by the first edge e added to it. Then it is initialized to be a union of two cones, each one on a different side of e with angle θ, meaning every cone has an angle 2θ. Note two cones may intersect and therefore the conditions which refer to the two cones containing an edge $\{p,q\}$, each one corresponds to different endpoint, may now refer to at most four cones, at most two cones correspond to each endpoint. Before a cone is initialized it is defined to be empty from edges.

The only t-spanner that is known to have the above property is the greedy t-spanner. However, the most efficient way known to compute the greedy t-spanner is a sequential construction in $O(n^2 \log n)$ time and it

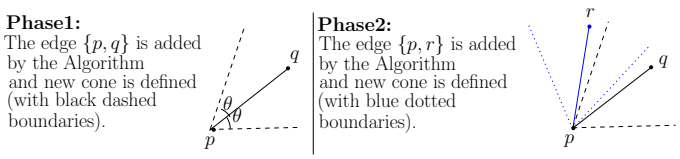

Figure 1: Illustrating the change described in Subsection 3.1.

is very complicated (see [2]), as opposed to the simple and intuitive local algorithm we suggested here that can be computed by R-processors distributed system in $O(n/R * n \log n)$ time. Note although there is also an approximate greedy Algorithm with $O(n \log^2 n)$ running time by Das and Narasimhan, great angles are not guaranteed in that construction (see [4]).

Theorem 7 *Given a set P of points in the plane or alternatively a unit disk graph and a constant $k \geq 8$, a t-spanner with all the properties of the* Stable roommates *t-spanner (Theorem 6) and additional property of having only angles grater than θ, where $\theta = 2\pi/k$, can be computed in $O(n/R * n \log n)$ time.*

References

[1] E. M. Arkin, S. W. Bae, A. Efrat, K. Okamoto, J. S. B. Mitchell, and V. Polishchuk. Geometric stable roommates. *Inf. Process. Lett.*, 109(4):219–224, 2009.

[2] P. Bose, P. Carmi, M. Farshi, A. Maheshwari, and M. H. M. Smid. Computing the greedy spanner in near-quadratic time. In *SWAT*, pages 390–401, 2008.

[3] P. Bose, M. Damian, K. Douïeb, J. O'Rourke, B. Seamone, M. H. M. Smid, and S. Wuhrer. Pi/2-angle yao graphs are spanners. *CoRR*, abs/1001.2913, 2010.

[4] G. Das and G. Narasimhan. A fast algorithm for constructing sparse Euclidean spanners. *Int. Journal on Computational Geometry and Applications*, 7(4):297–315, 1997.

[5] G. Narasimhan and M. Smid. *Geometric Spanner Networks*. Cambridge University Press, New York, NY, USA, 2007.

[6] Y. Wang and X.-Y. Li. Distributed spanner with bounded degree for wireless ad hoc networks. In *IPDPS '02: Proc. of the 16th International Parallel and Distributed Processing Symposium*, page 120, 2002.

[7] A. C.-C. Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM J. Comput.*, 11(4):721–736, 1982.