Well-Separated Pair Decomposition and t-Spanner Construction using Quadtrees

Raja Gopal, Praneeth
210050160, 210050094

April 20, 2025

Problem Statement

Given a set P of n points in $\mathbb{R}^2$, we aim to:

1. Compute a Well-Separated Pair Decomposition of P using a separation constant $s > 0$.
2. Use the WSPD to compute a geometric t-spanner of P with stretch factor $t = (1 + \varepsilon)$.

Complexities:

- WSPD: $\mathcal{O}(n \log n)$
- t-Spanner: $\mathcal{O}(n \log n)$ (for constant ε)

Overview of the Approach

We use a **quadtree** data structure to recursively divide the 2D space containing the point set P. The WSPD is then computed using a recursive method that checks the well-separatedness of node pairs in the quadtree. For the spanner construction, we utilize the WSPD to add edges between representative points of the pairs.

Implementation Details

Building the Quadtree

Each node of the quadtree stores:

- The center and half-side length
- A pointer to the representative point
- Leaf or internal status

Listing 1: Quadtree Construction

```
void quadTree::build_tree(Node* node, vector<point> p, point center, double
    half_side){
    ...
    for(int i = 0; i < 4; i++){
        point new_center = {center.first + (dirs[i].first*half_side),
                            center.second + (dirs[i].second*half_side)};
        node->child[i] = new Node(new_center, half_side);
        build_tree(node->child[i], new_points[i], new_center, half_side);
    }
}
```

WSPD Generation

Two nodes u and v are considered well-separated if:

$$\text{distance}(u, v) \geq s \cdot \max(\text{radius}(u), \text{radius}(v))$$

Listing 2: Well-Separated Pair Decomposition

```
bool wellSeparated(Node* u, Node* v, double s) {
    double max_radius = max(radius(u), radius(v));
    double d = dist(u->center, v->center);
    return d >= s * max_radius;
}

void quadTree::get_wspd(Node* u, Node* v) {
    if (wellSeparated(u, v, s)) {
        wspd_pairs.push_back({u_leaves, v_leaves});
        edges.push_back({*u->rep, *v->rep});
    } else {
        ... // recursively check children
    }
}
```

t-Spanner Construction

Using the WSPD pairs, we create a t-spanner by connecting the representative points of each pair:

$$s = 4 * \frac{t+1}{t-1}$$

Listing 3: t-Spanner Generation

```
void quadTree::get_spanner(double eps){
    s = 4*(eps+2)/(eps);
    get_wspd(root, root);
    plot();
}
```

Plotting and Visualization

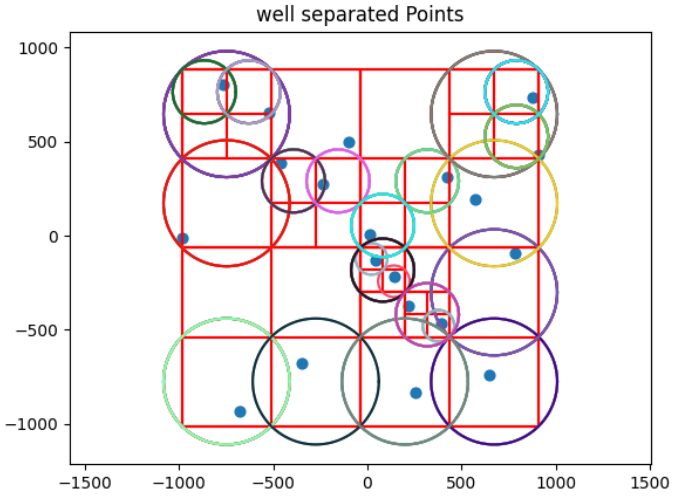
The plot function uses Matplotlib (through C++ bindings) to visualize the t-spanner/ quadtree/ wspd **(same color circle represents a pair)**:

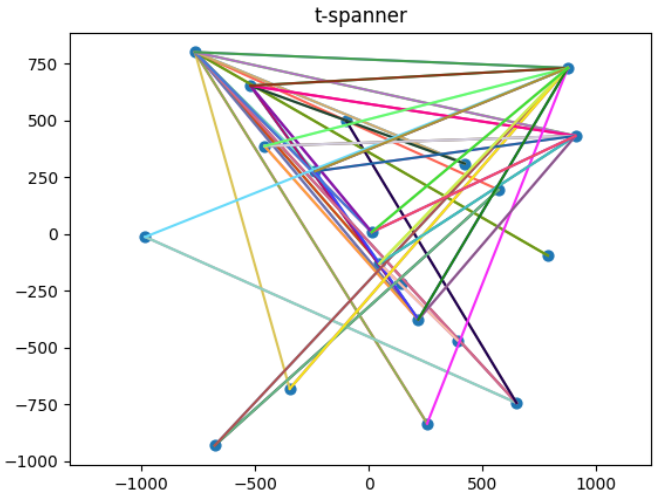
Listing 4: Plotting Spanner

```
void quadTree::plot(){
    scatter_points(points);
    plot_spanner(edges);
    plot_quadtree(centers, hs);
    plot_circles(centers, r);
    plt::save("spanner.png");
    plt::show();
}
```

Results

- Number of points used: 20
- Number of **WSPD** pairs: 73





Conclusion

We successfully implemented WSPD and used it to construct a geometric t-spanner in $\mathcal{O}(n \log n)$ time using a quadtree. The resulting spanner preserves approximate distances and can be visualized clearly using the plotted edges.