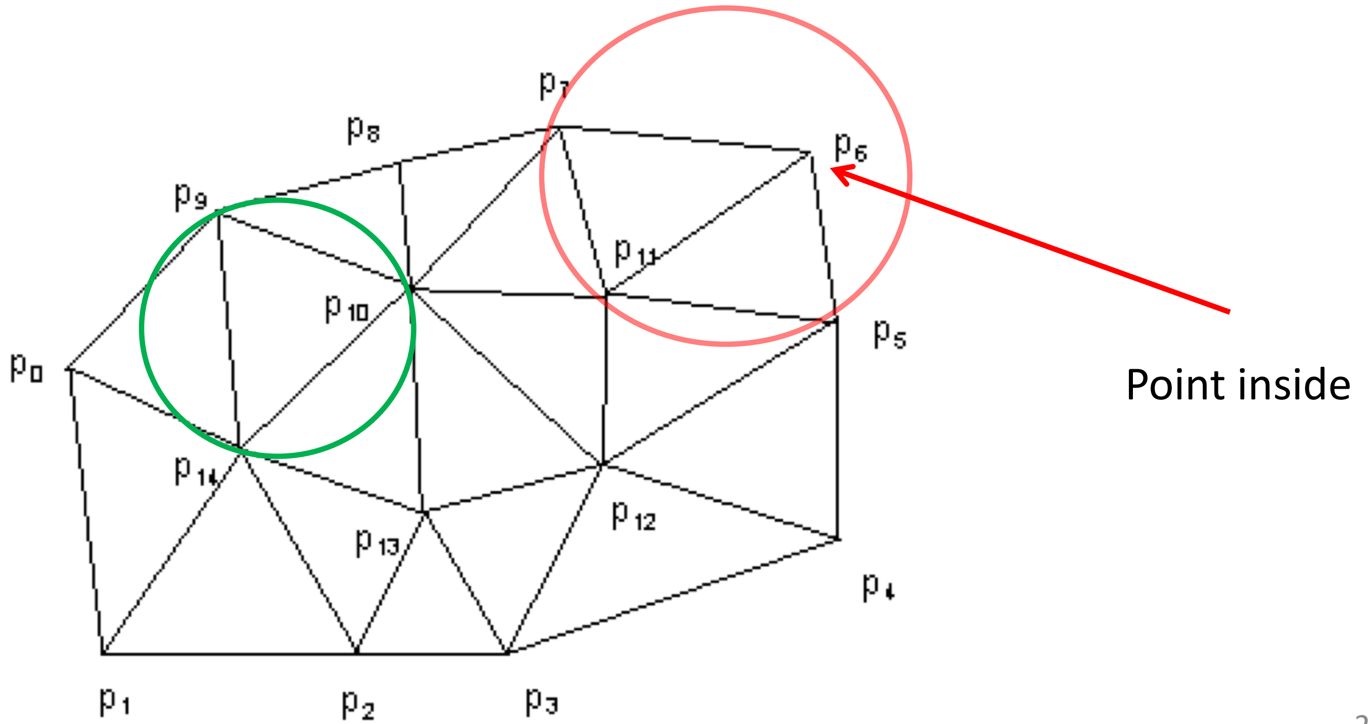
Parallel Incremental Delaunay Triangulation

Julian Shun

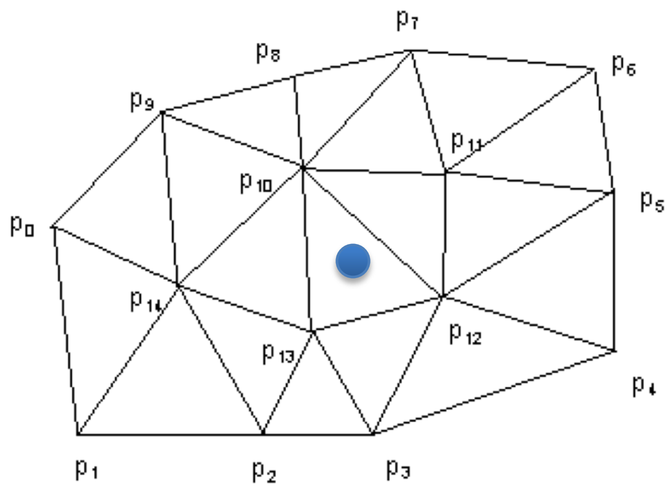
Delaunay Triangulation

- Problem: Given a set of points, create a triangle among 3 points if there is no other point inside the circumcircle of those 3 points



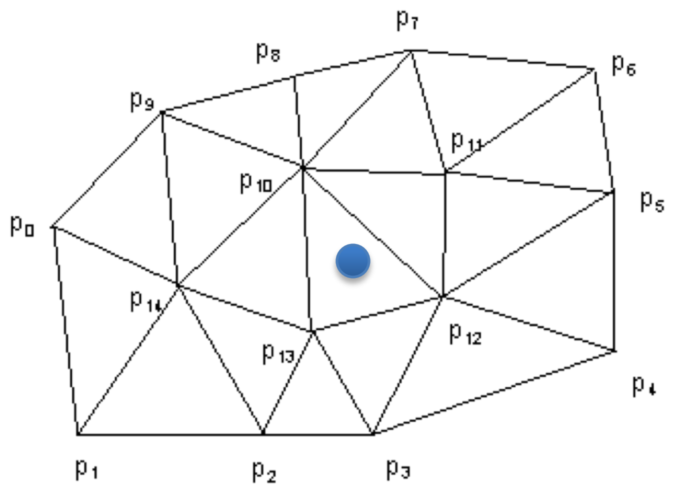
Delaunay Triangulation

- Serial incremental algorithm

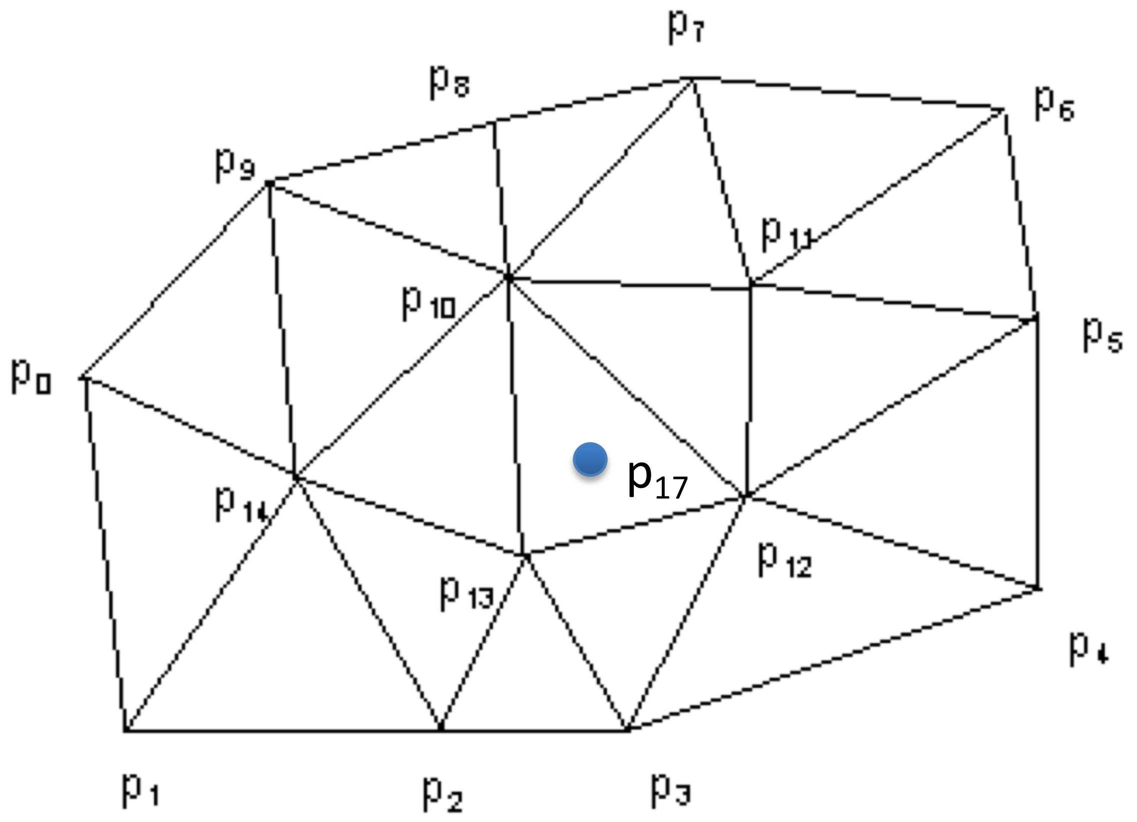


Delaunay Triangulation

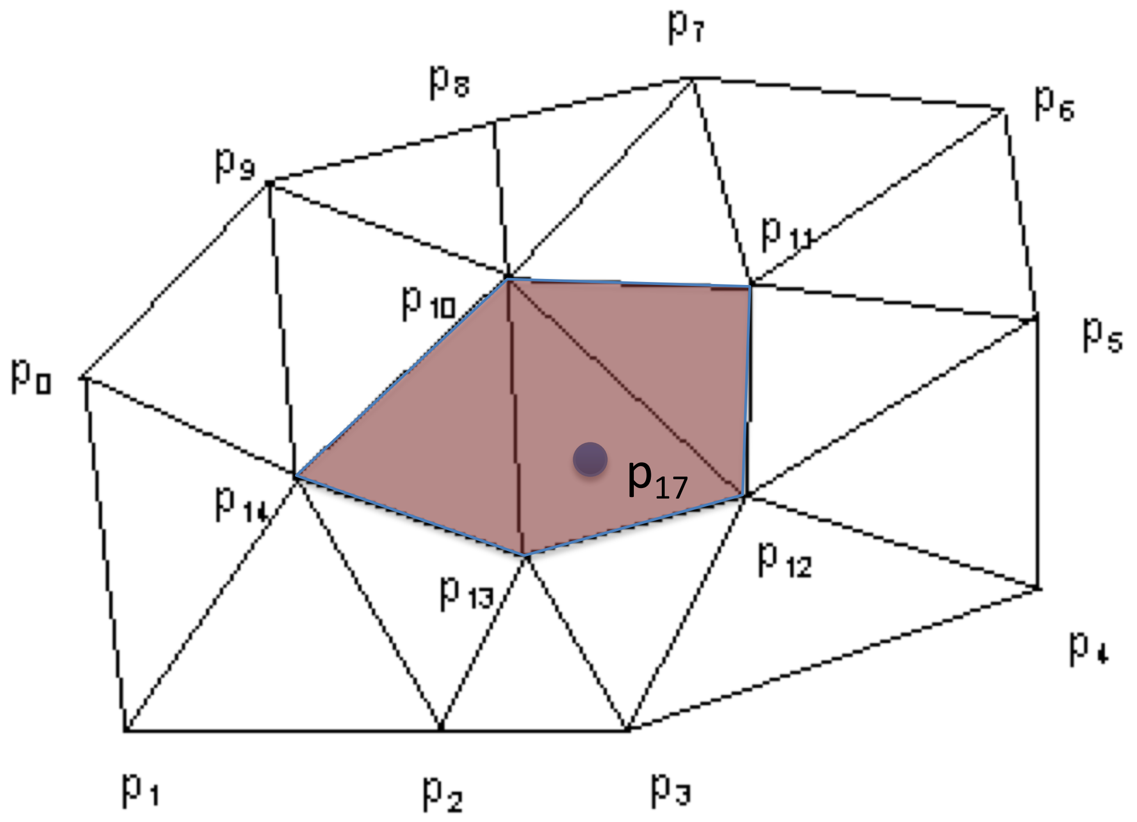
- Serial incremental algorithm adds one point at a time, but points can be added in parallel if they don't interact.
- How can we find a set of independent points to add in parallel?



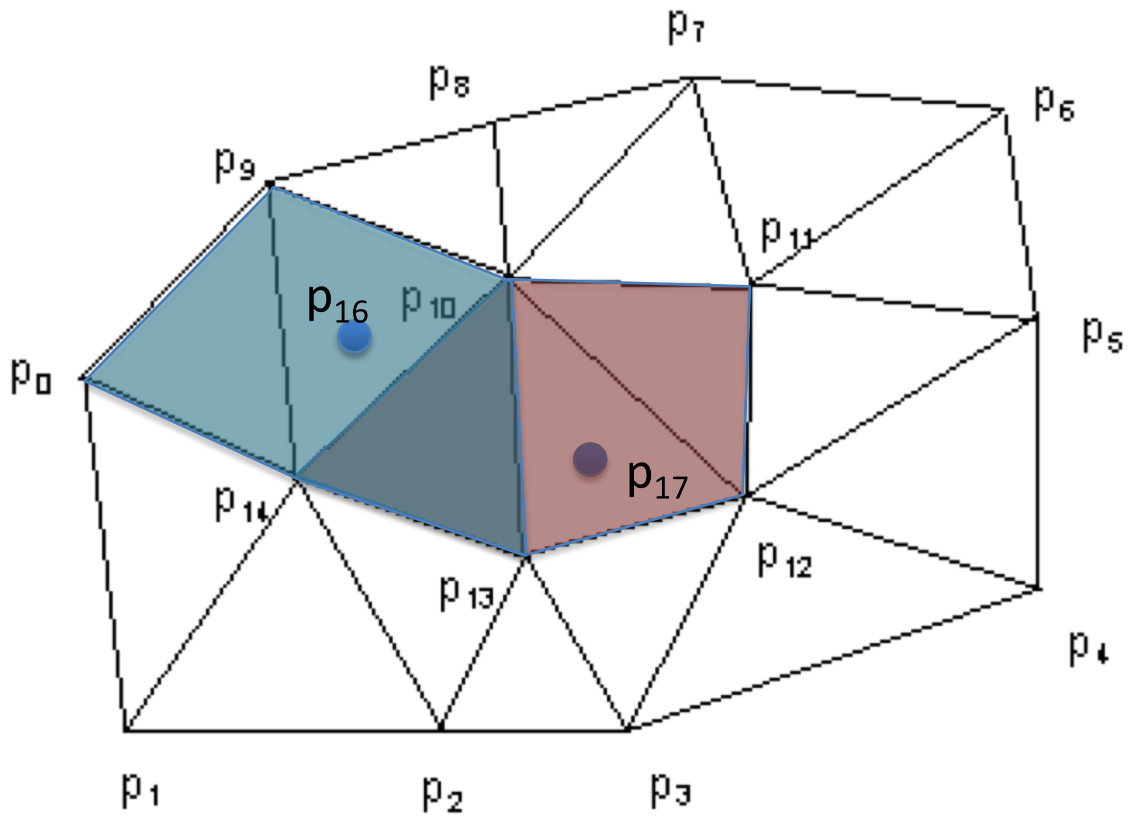
Delaunay Triangulation



Delaunay Triangulation

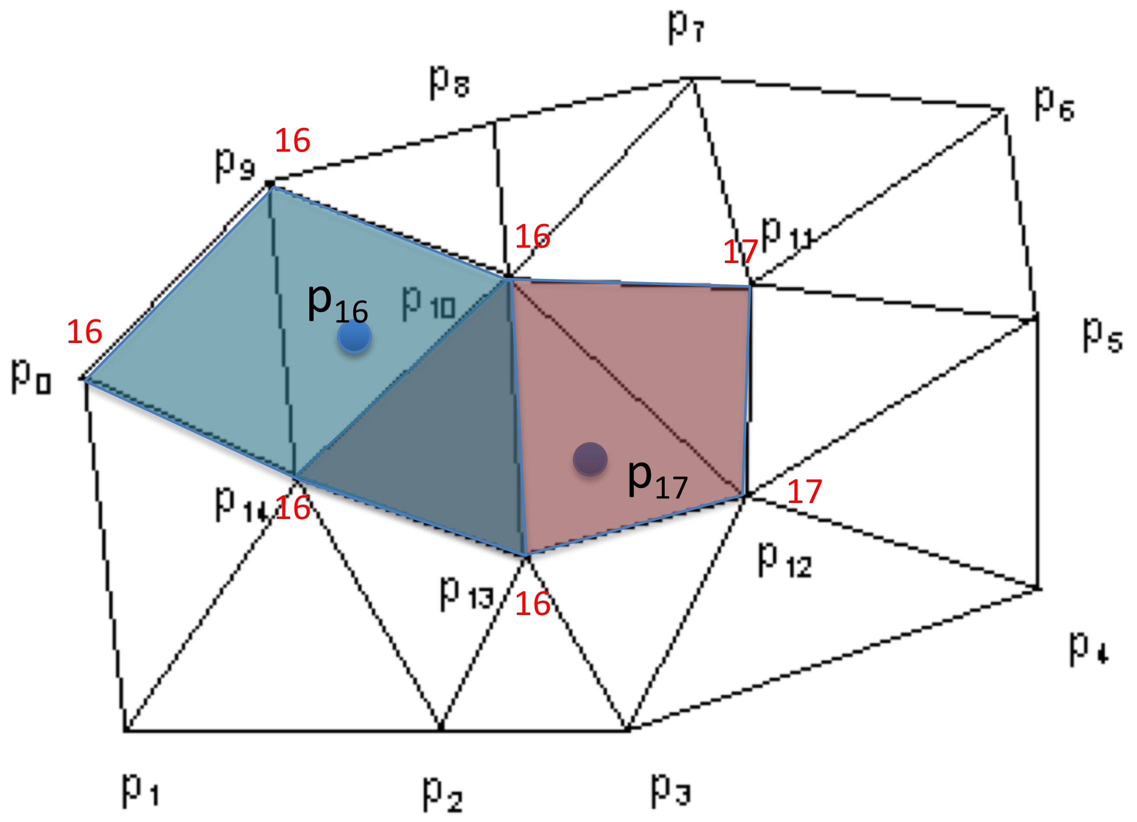


Delaunay Triangulation



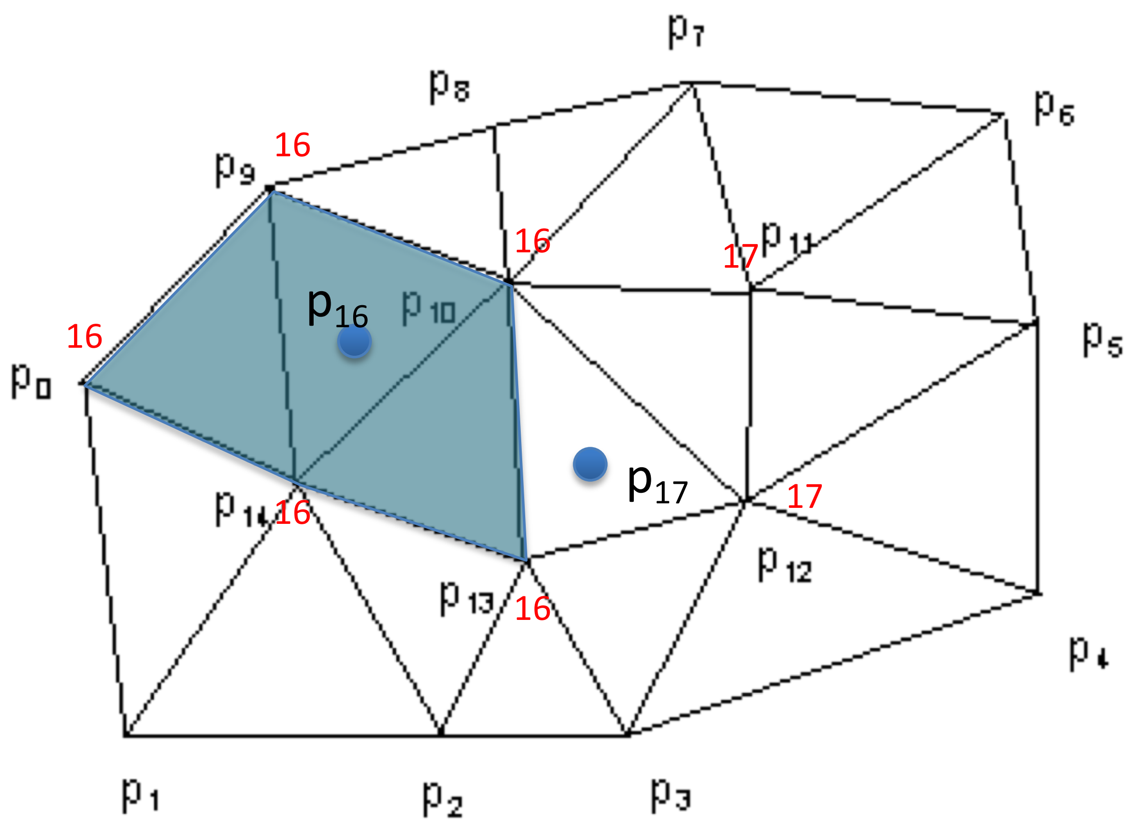
Delaunay Triangulation

- Use a write-min with ID to endpoints of affected triangles



Delaunay Triangulation

- A point that has its ID on all of its affected triangles can insert itself



Deterministic Reservations

<u>Generic framework</u>

```
elements = [1,…,n];
while(elements remain){

    Phase 1: in parallel, all i in elements
                    call reserve(i);

    Phase 2: in parallel, all i in elements
                    call commit(i);

    Remove successfully committed i's from
          elements;
}
```

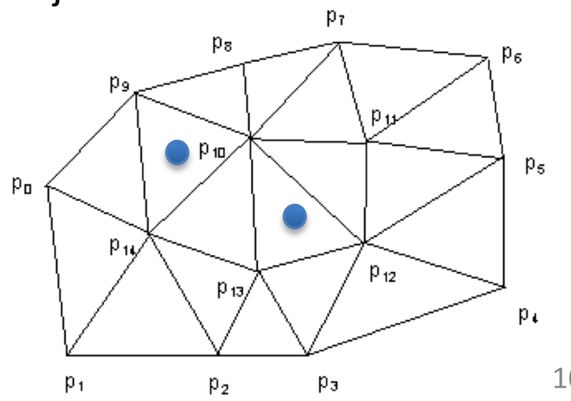
- Note: Which elements successfully commit is deterministic.

<u>Delaunay triangulation</u>

elements: points to be added

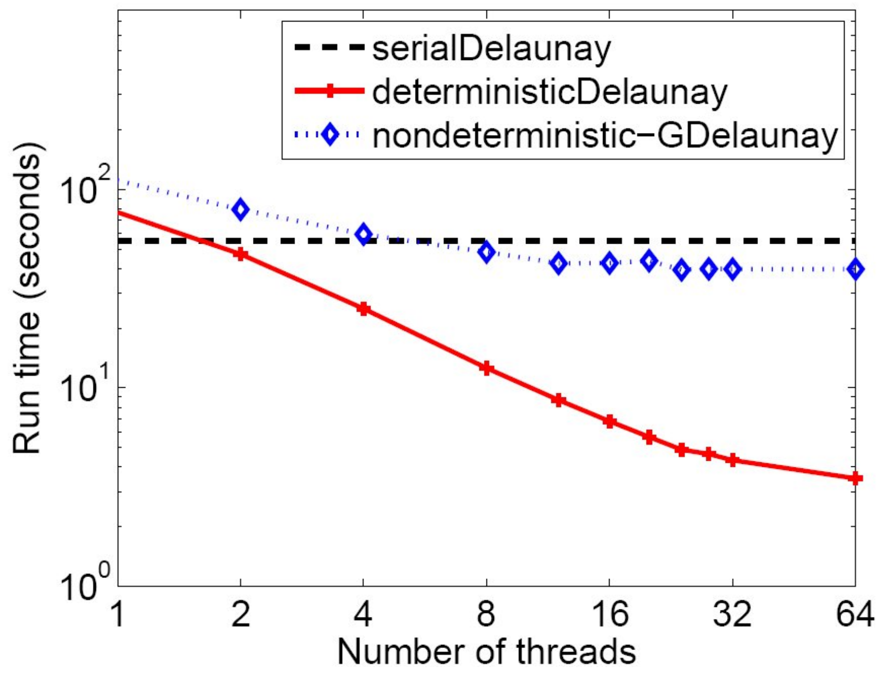
```
reserve(i){
    find affected region;
    reserve points in region;
}

commit(i){
    check reservations;
    if(all reservations successful){
        add point and triangulate;
    }
}
```



Experimental Results

Delaunay Triangulation



- 21x speedup on 32 cores
- On 1 thread, 1.4x slower than serial

Theoretical Bounds

- The presented algorithm takes could take a linear number of steps due to conflicts, giving $\Omega(n)$ span
- If we randomize point order and insert triangles of points that do not conflict while delaying the rest, we get $O(n \log n)$ expected work and $O(\log^2 n)$ span with high probability