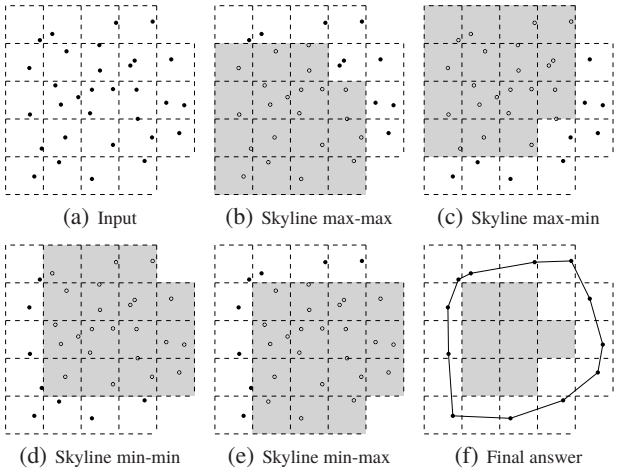


**Figure 5: Convex hull in SpatialHadoop**

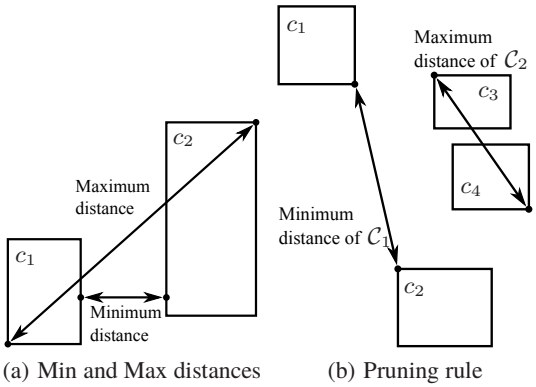


**Figure 6: Farthest pair algorithm in SpatialHadoop**

tinues by computing the lower chain in a similar way by checking all points of $P$ from right to left and doing the same check. Using PostGIS [32], the convex hull can be computed by a single SQL query using the function `ST_ConvexHull`. Since this function takes one record as argument, points have to be first combined in one line string using the function `ST_Makeline`.

```
SELECT ST_ConvexHull(ST_Makeline(points.coord))
FROM points;
```

In this section, we introduce two convex hull algorithms for Hadoop and SpatialHadoop, while using input dataset in Figure 1(c) as an illustrative example.

## 5.1 Convex Hull in Hadoop

Our Hadoop convex hull algorithm is very similar to our Hadoop skyline algorithm, where we start by a *partitioning* phase to distribute the input data into small chunks such that each chunk fits in memory. Then, the *local convex hull* of each subset is calculated using the traditional in-memory algorithm [3] and only those points forming the convex hull are retained. The points from all convex hulls in all machines are combined in a single machine that computes the *global convex hull*, which forms the final answer, using the traditional in-memory convex hull algorithm. Similar to skyline, the number of points on the convex hull is expected to be $O(\log n)$ [10] for uniform data, making this algorithm very efficient in pruning most of the points when computing the local hull and allowing the global hull to be computed in one node.

## 5.2 Convex Hull in SpatialHadoop

The convex hull algorithm in Hadoop processes more file partitions than necessary. Intuitively, the parts of the file that are towards the center do not contribute to the answer. In SpatialHadoop, we improve the convex hull algorithm by early pruning those partitions that do not contribute to answer. The key idea is that any point on the convex hull must be part of at least one of the four skylines of the dataset (max-max, min-max, max-min, and min-min) [33]. A max/min-max/min skyline considers that maximum/minimum points are preferred in $x$-$y$ dimensions. This property allows us to reuse the skyline *filtering* step in Section 4.2. As given in Figure 5, we apply the skyline algorithm four times to select the partitions needed for the four skylines and take the union of all these partitions as the ones to process. Clearly, a partition that does not contribute to any of the four skylines will never contribute to the final convex hull. Once the partitions to be processed are selected, the algorithm works similar to the Hadoop algorithm in Section 5.1 by computing the local convex hull of each partition and then combining the local hulls in one machine, which computes the global convex hull. The gain in the SpatialHadoop algorithm comes from the spatially-aware partitioning scheme that allows for the pruning in the *filtering* step, and hence the cost saving in both local and global convex hull computations. For interested readers, the pseudocode of the SpatialHadoop convex hull algorithm is in Appendix A.3.

# 6. FARTHEST PAIR

A nice property of the farthest pair (shown in Figure 1(e)) is that the two points forming the pair must lie on the convex hull of all points [34]. This property is used to speed up the processing of the farthest pair operation by first computing the convex hull, then finding the farthest pair of points by scanning around the convex hull using the rotating calipers method [33]. In this section, we introduce our farthest pair algorithms for Hadoop and SpatialHadoop.

## 6.1 Farthest Pair in Hadoop

A Hadoop algorithm for the rotating calipers method [33] would complete the convex hull first as discussed in Section 6.1. Then, a single-machine will need to scan all the points in the convex hull, which may be a bottleneck based on the number of points in the convex hull. In that case, it may be better to develop a Hadoop algorithm based on parallelizing the brute-force approach of farthest pair algorithm, which calculates the pairwise distances between every possible pair of points and select the maximum. The brute force approach will be expensive for very large input files, yet it may be used if it is not feasible for one machine to calculate the farthest pair from the points in the convex hull as in the rotating calipers method. Overall, both the brute force and rotating calipers methods have their own drawbacks when realized in Hadoop.

## 6.2 Farthest Pair in SpatialHadoop

Our SpatialHadoop algorithm works similar to our skyline and convex hull algorithms as we have four steps, *partitioning*, *filtering*, *local farthest pair*, and *global farthest pair*. In the *partitioning* step, we mainly use the SpatialHadoop partitioning scheme. In the *filtering* step, we apply a specialized filtering rule for the *farthest pair* operation. The main idea is explained in Figure 6. For each pair of cells, $c_i$ and $c_j$, we compute the *minimum* (*maximum*) distance between $c_i$ and $c_j$ as the minimum (maximum) possible distance between any two points $p_i \in c_i$ and $p_j \in c_j$ (Figure 6(a)). Then, given two pairs of cells $\mathcal{C}_1 = \langle c_1, c_2 \rangle$ and $\mathcal{C}_2 = \langle c_3, c_4 \rangle$, we