The Kirkpatrick-Seidel Marriage Before Conquest Algorithm

1 Purpose

For most commonly used 2-D convex hull algorithms, the asymptotic complexity is $O(n \log(n))$, where n is the size of the point set P. This bound holds regardless of how the input set of points P is arranged. Can this bound be improved for certain arrangements of points?

The set P_1 in Figure 1 has a very small number of points on its convex hull in relation to the size of the whole set. In set P_2, *all* points in the set are on the convex hull. Is there an algorithm which is *output sensitive* in relation to the number of points on the convex hull? It turns out there is such an algorithm - Kirkpatrick-Seidel Marriage Before Conquest. Its complexity, as we shall see, is $O(n \log(h))$, where n is the size of the point set P and h is the number of points on the convex hull of P.

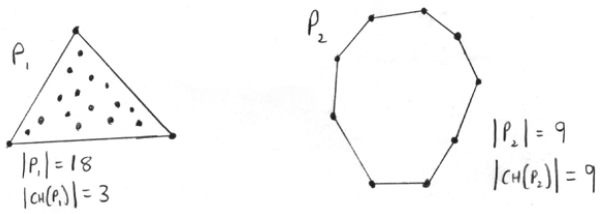

Figure 1: Size of hull vs. size of set

2 Algorithm Description

Like the standard Divide & Conquer for 2D Convex Hull, Marriage-Before-Conquest (MBC) utilizes the strategy of finding *bridges* on the convex hull. A bridge is simply a single edge of the convex hull. In the case of Divide & Conquer, finding bridges constitute the "merge" step of the algorithm. In MBC, finding bridges *are* the primary algorithmic step. The technique used to find a bridge is commonly known as *Prune and Search*. Once such a bridge

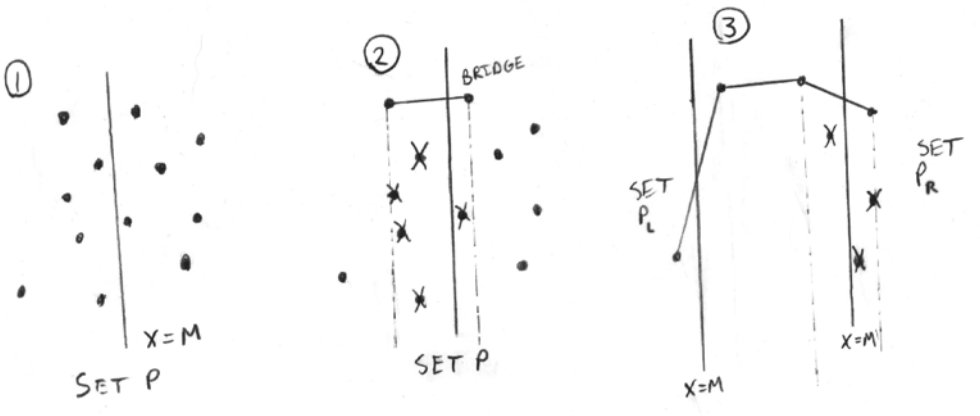

Figure 2: MBC on Fast-Forward

is found, certain subsets of points in P may be eliminated from consideration in the convex hull of P. This algorithm, as we shall see, is output sensitive because points may be removed from consideration in this way.

MBC is comprised of the following major steps, each of which will be described in more detail in later paragraphs:

1. Calculate the median x-coordinate M of the points in P.
2. Find the bridge segment that crosses the line $x = M$ using the Prune & Search technique
3. Trim the set P based on the characteristics of the bridge segment
4. Split remainder of set P into two new sets P_L and P_R, and recursively perform the steps above on each subset to find all remaining bridge segments
5. The above steps must be run twice - once for the upper-hull, and once again for the lower-hull. The merged upper & lower hulls comprise the complete convex hull.

Figure 2 shows a simplified version of the algorithm. Note the presence of the lines $x = M$ in each step and the bridge segments which cross them.

Note also the dotted lines which show the Prune & Search area; points in this region are crossed out and removed from further processing.

3 Algorithm Details

The following sections apply to computation of the upper convex hull of the set P. The lower-hull computation steps are nearly identical; for the sake of brevity, those steps will not be outlined here. We can safely assume that lower-hull may be computed with the same algorithmic complexity as upper-hull and that merging the two hulls may be performed in constant time.

Initial Steps

Before processing begins, determine the x-coordinate boundaries of the set P. Let $p_j = max_x(P)$ and $p_k = min_x(P)$. In the event that there are any duplicates in x-coordinate, the points with the larger y-coordinates are chosen. MBC assumes that the set P is not ordered. Therefore, $O(n)$ time must be spent selecting p_j and p_k from the set P.

As an optional step, the algorithm may decide to trim the set P using the line segment $p_j p_k$ as shown in Figure 3. All points on or above the line are candidates for upper/lower convex hull. All those below cannot be. Note the choice of wording in the previous sentences; the discarded points may very well be part of the *overall* convex hull, but they are not part of the half-hull currently under consideration. The set $P\prime$ denotes the set P with the aforementioned points removed. For the remainder of the discussion, we will continue to refer to the input set as P regardless of whether or not this step was performed.

Bridging (Prune & Search)

Calculate the median x-coordinate M of P. Let $x = M$ define a vertical line which divides the set P in half. The correct bridge across this line will be a piece of the final convex hull. Finding the bridge via Prune & Search consists of the following steps:

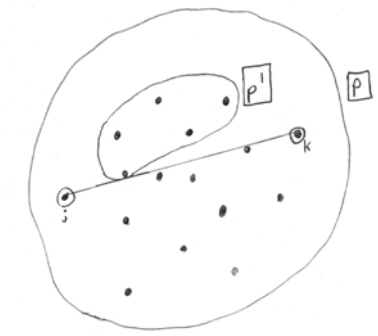

Figure 3: Trimming the set P

1. In linear time, randomly pair the points of the input set P into $\lfloor \frac{n}{2} \rfloor$ distinct line segments. Call this set Q, and name each of the $\frac{n}{2}$ line segments q_i. Name each endpoint of q_i according to the following convention: ql_i x-coordinate $\leq qr_i$ x-coordinate. That is to say, endpoints are named in ascending order from left to right. If the parity of n is odd, there may be one point which does not have a partner. This is acceptable.

2. In linear time, determine the median slope m of all of the line segments $\in Q$. For singleton points, consider slope to be zero. If the parity of Q is even, m assumes the greater valued slope between two segments.

3. Construct a line L whose equation is $y = mx + b$. This line is called the *sweep line*.

4. Using L, find a point p_t such that L is a supporting line for P at p_t. We will call the point p_t the *top point*. If the slope m of the sweep line L is such that two points $\in P$ lie on L, we choose p_t to be the point with the larger x-coordinate.

5. To prune the set of bridge point candidates, do the following:

   - If x-coordinate of $p_t \geq M$, for every line segment $q_i \in Q$ whose slope is $< m$, remove qr_i
   - If x-coordinate of $p_t < M$, for every line segment $q_i \in Q$ whose slope is $> m$, remove ql_i

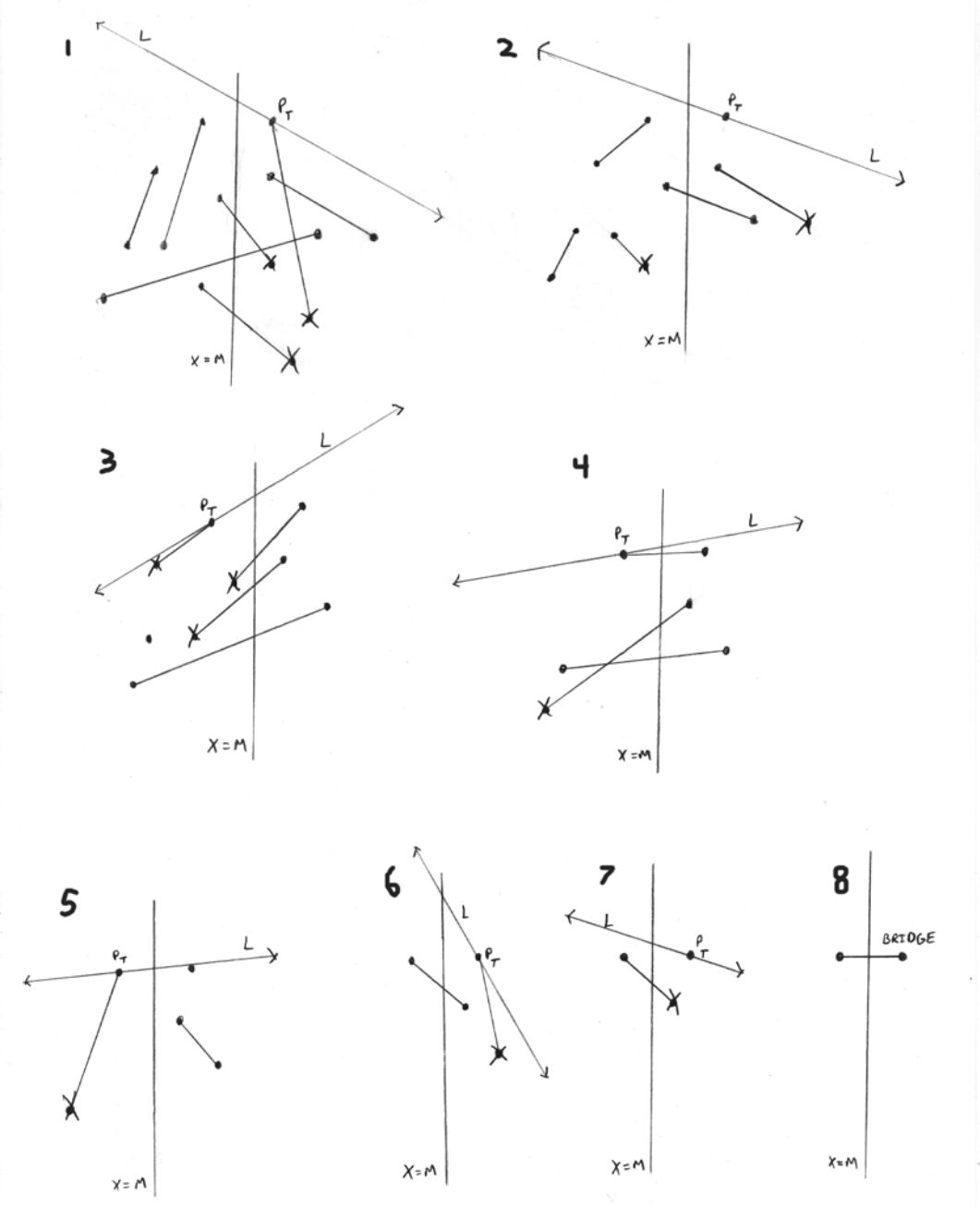

Figure 4: Prune & Search on 14 Point Set

Because L was chosen to have the median slope of all the segments $\in Q$, then we know that half of all q_i have greater slope and half have lesser slope. Therefore, on every iteration, $\frac{1}{4}$ of the remaining points are removed from consideration as bridge endpoints.

Subdivision

After finding the bridge over $x = M$ such that p_r and p_l are the bridge endpoints, perform the following:

1. Discard elements of P which cannot possibly be on the convex hull of P. Two acceptable methods (both $O(n)$) for doing this:

   - Discard all points with x coordinates between those of p_l and p_r. Visually, this is seen as discarding points "under" the bridge.
   - Discard all points in the quadrangle formed by p_j, p_l, p_r, and p_k.

2. Split the remaining points of P into two subsets, P_L and P_R. P_L contains points from p_j to p_l and P_R contains points from p_r to p_k.

3. Call MBC recursively on (P_L, p_j, p_l) and (P_R, p_r, p_k). When the recursive calls terminate, the result is a complete half-hull of P.

4 Algorithm Analysis

Theorem 4.1 *$MBC(n, h) = O(n \log(h))$, where n is the size of a set of points P in the two-dimensional Euclidean plane and h is the number of points on the convex hull of P*

Proof 4.2 $T_{bridge}(n) \leq T_{bridge}(n - \lfloor \frac{n}{4} \rfloor) + O(n) \rightarrow T_{bridge}(n) = O(n)$

$T(n, h)$ describes the general recurrence for finding a half-hull of P; i.e. the recurrences for $T_{lower}(n, h)$ and $T_{upper}(n, h)$ are identical.

$$T = \begin{cases} T(n,h) & \leq & T(\frac{n}{2}, h_l) + T(\frac{n}{2}, h_r) + T_{bridge}(n) \\ T(n,2) & = & O(n) \\ T(n,1) & = & 0 \end{cases}$$

where $h_l + h_r = h; h_r, h_l \geq 1$

$T(n,h) \leq cn\log(h)$
Proof by induction on h
Basis: $h = 2, T(n,2) = c_1 n \leq cn\log(2) \quad (Pick\ c \geq c_1)$
Inductive Hypothesis:

$$\begin{aligned} T(n,h) &\leq c_1 n + \frac{cn}{2}\log(h_l) + \frac{cn}{2}\log(h_r) && (1) \\ &\leq c_1 n + \frac{cn}{2}(\log(h_l) + \log(h_r)) && (2) \\ &= c_1 n + \frac{cn}{2}(\log(h_l h_r)) && (3) \\ &= c_1 n + \frac{cn}{2}(\log((h - h_l)h_l)) && (4) \\ &\leq c_1 n + \frac{cn}{2}(\log(\frac{h^2}{2^2})) && (5) \\ &= c_1 n + cn(\log(\frac{h}{2})) && (6) \\ &\leq cn\log(2) + cn(\log(\frac{h}{2})) && (7) \\ &= cn\log(h) && (8) \end{aligned}$$

To complete the overall convex hull, one must execute T(n, h) twice, once for the upper-hull and once again for the lower. This increases the bound by a constant factor, thus $MBC(n,h) = O(n\log(h))$.