On $k$-Enclosing Objects in a Coloured Point Set

Luis Barba
Stephane Durocher
Robert Fraser
Ferran Hurtado
Saeed Mehrabi
Debajyoti Mondal
Jason Morrison
Matthew Skala
Mohammad Abdul Wahid
Problem Definition

• Given a coloured point set, where colours are in \( \{1,\ldots,t\} \)
• Given a query \( c = (c_1,\ldots,c_t) \) (and define \( k = \sum_i c_{-i} \))
• Does there exist an axis aligned rectangle containing a set of points satisfying the query exactly?

Say colours are \((r,b,g)\)
\(c = (1,1,3)\)
How about \(c = (0,1,3)\)?
Motivation

• Inspired by problems in string processing (jumbled pattern matching).
Motivation

• Inspired by problems in string processing (jumbled pattern matching).

• Given a string of characters from some set \{1,\ldots,t\} and a query \(c=(c_1,\ldots,c_t)\), does there exist any contiguous substring that is a permutation of the query?

• String: 0111001111011110

• Query: (1,6)
Let’s generalize this to 2D!

- Most natural (CG) generalization may be to have coloured points enclosed by a rectangle.

Wait, wait, wait!
Surely this has been done!

What has been done?
Related Problems

• **Jumbled Pattern Matching** - Find a permutation of a query in a string (the 1D version of our problem).

• **k-Enclosing Objects** - Find the smallest $k$-enclosing rectangle/square/disc.
  – Points are *uncoloured*.

• **Smallest Colour-Spanning Object** - Smallest object containing at least one point of each colour.

• **Subarray Sum** - Given a 2D array and a value $v$, find a subarray whose sum is $v$. 
Results

- Axis parallel rectangles in $O(n^2 k)$ time.
  - This is the main result, and certainly the most technical.
- Discs in $O(KVD(n, k))$ time.
- Squares are similar.
- Two-sided dominating regions in $O(n \log n)$ time.
- Various extensions:
  - Improvements on running time for rectangles
  - Smallest $k$-enclosing objects
  - Higher dimensions
The Rectangle Problem

• Consider a brute force approach...
• Any combination of 4 points may determine the sides of the rectangle.
• $O(n^4)$ possible solutions...
1st Insight: Break into Strips!

- Rather than looking at every combination brute force, bound the top and bottom and look at the 1D problem in the strip!
Strips are good

• The benefit of using strips:
  – Suppose you checked a given strip and found no solution.
  – Move the top of the strip up to include the next point encountered.
  – This leads to an $O(n^3)$ time solution: spend linear time checking for a solution in each strip.
  – If a solution exists in this expanded strip, it must lie within $k = \sum_i c_i$ points of the new top point.
How to do this quickly?

• Have the points sorted by y-coordinate, and maintain a linked list sorted by x-coordinate.
• As the height of the strip is grown, insert the point at the appropriate place in the linked list.
  – Want constant time update!
• Check the \( \pm k \) neighbourhood of the inserted point for a solution \( \rightarrow O(n^2k) \) time.
Improvements

• Only need to check $O(n - k)$ possible tops for the same number of bottoms, since a solution must contain at least $k$ points.
  – Leads to $O(n \log n + (n - k)^2 k)$ time.
  – For context, consider the best known algorithm for the uncoloured version of the problem on rectangles: $O(nk^2 + n \log n)$ time (Eppstein & Erickson, 94).

• We can remove $\max_i c_i$ from the running time!
  – $O(n \log n + (n - k)^2 (k - \max_i c_i))$ time.

• Generalizes to higher dimensions in $O(n^{2(d-1)}kd)$ time.
**$k$-Enclosing Discs**

- Compute the $k$th order Voronoi diagram.
- Recall that a cell in the diagram corresponds to the set of points with the same $k$ closest points.
$k$th Order Voronoi Diagram

$k$-Enclosing Discs

- Compute the $k$th order Voronoi diagram.
- Recall that a cell in the diagram corresponds to the set of points with the same $k$ closest points.
- The challenge is to check each cell of the diagram in amortized $O(1)$ time.
- Possible by being smart during a traversal of the cells.
Results

• Axis parallel rectangles in $O(n^2k)$ time.
  – Various improvements to running time.
• Discs in $O(KVD(n, k))$ time, e.g. $O(nk^2 \log n)$.
• Squares are similar.
• Two-sided dominating regions in $O(n \log n)$ time.
• Smallest $k$-enclosing objects
• Higher dimensions

Thanks!
**k-Enclosing Objects**

- Find the smallest $k$-enclosing rectangle/square/disc
  - Points are *uncoloured*.
- 1991 (Aggarwal et al.) – smallest rectangle or square can be found in $O(nk^2 \log n)$ time.

...  
- 1994 (Eppstein & Erickson) – rectangles in $O(nk^2 + n \log n)$ time.
- 1994 (Eppstein & Erickson) – discs in $O(nk \log k + n \log n)$ time.
- 1995 (Datta et al.) – squares in $O(n \log^2 k + n \log n)$ time.
- 1999 (Chan) – smallest square in $O(n \log n)$ expected time.
Smallest Colour-Spanning Object

• Smallest object containing at least one point of each colour.

• 1993 (Huttenlocher et al.) – discs in $O(nt \log n)$ time.

• 2009 (Das et al.) – rectangles in $O(n(n - t) \log t)$ time.
Subarray Sum

• Given a matrix and a value $v$, find a subarray whose sum is $v$.

• Could be used to solve our rectangle query problem in $O(n^3 \log n)$ time.
  – We solve it in $O(n^2 k)$ time.
Tweaking the running time

• We can remove $\max_i c_i$ from the running time!
  - $O(n \log n + (n - k)^2 (k - \max_i c_i))$ time.

• Idea:
  – Call the dominant colour $m$.
  – Count the number of $m$ coloured points between each other point in the data set.
  – If you have a match for the rest of the query, check to see if a match exists for these too.
Counting the $m$'s

- Consider in the strip again. Suppose the query is $(1,2,2,5)$ for $(r,g,b,m)$.
- The numbers between the points are the number of $m$ coloured points.
- Need the query to be satisfied for all other colours, and $c_m$ is within bounds.
$k$-Enclosing Squares

- Again, use $k$th order Voronoi diagram, but under the $L_\infty$ distance.
Two-Sided Dominating Regions

- Compute $c_i$-levels for each colour in $O(n \log n)$ time.
- Merge in $O(n \log t)$ time.
Smallest Exact Solution

• For rectangles, no extra time is required. Simply compute the area of the rectangle for any solution found.

• Discs and squares may be solved with an at most $O(k)$ increase in running time.