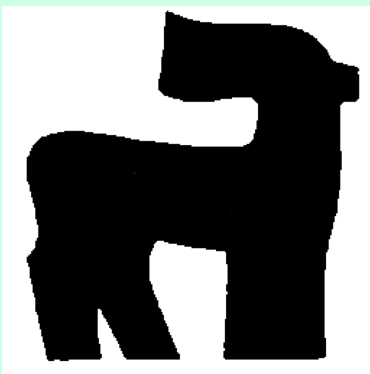
ACCORD: WITH APPROXIMATE COVERING OF CONVEX ORTHOGONAL DECOMPOSITION

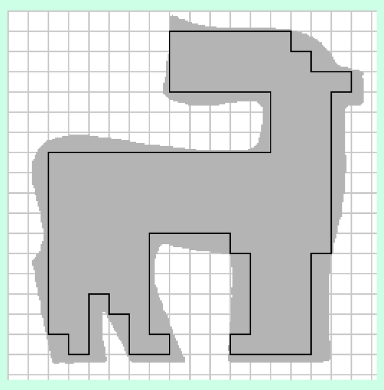
Mousumi Dutt, Arindam Biswas : Bengal Engineering and Science University, Shibpur, India
Partha Bhowmick : Indian Institute of Technology, Kharagpur, India
{duttmousumi,barindam}@gmail.com, bhowmick@gmail.com

OBJECTIVE

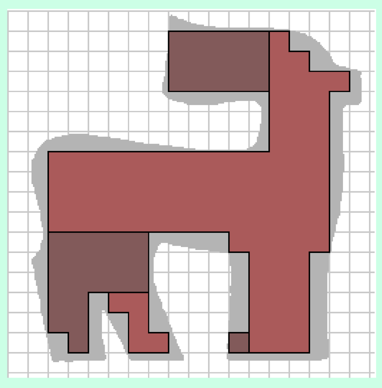
To decompose approximately a 2D digital object by partitioning the inner cover, A'_{in}, (the maximally inscribed orthogonal polygon) of the object into a set of orthogonally convex components.



Object A



Inner isothetic cover, A'_{in}



Decomposition of A'_{in}

Conditions:

Decomposition of hole-free polyomino (here, A'_{in}) into a sub-optimal[a] set of orthogonally convex components (OCC or, hv-convex polyominoes) such that

- each OCC is orthogonal with all its vertices as grid points
- no two OCC overlap each other except at their boundaries

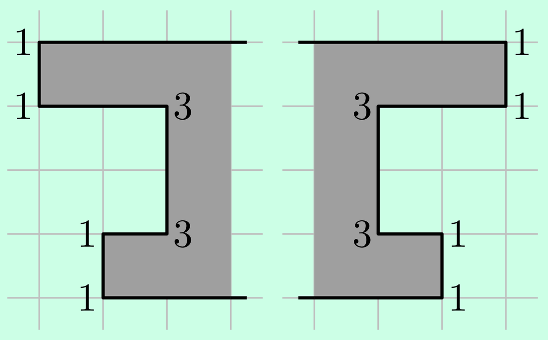
Open Problem:

To the best of our knowledge, there exists no proof till date to show whether partitioning an orthogonal polygon into a minimal set of OCCs can be done in polynomial time.

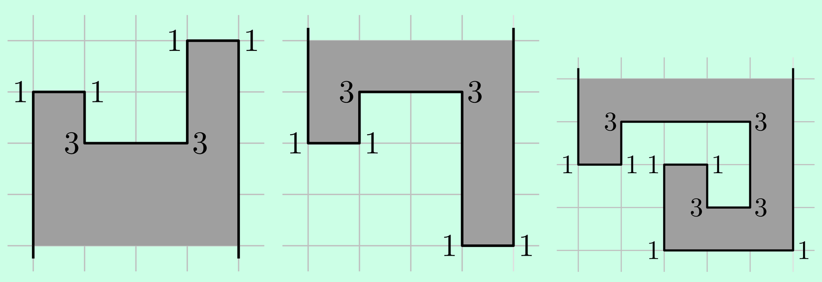
PRELIMINARIES

Types of Concavity:

- Four kinds of *simple concavities* ("1331" vertex pattern): Type L (left), Type R (right), Type T (top), Type B (bottom)
- Three or more consecutive Type 3 vertices form a *compound concavity*, stored as simple concavities in L_c



Type L   Type R

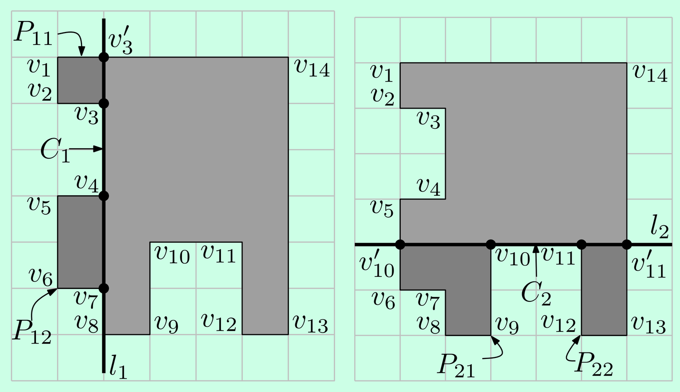


Type T   Type B   Compound Concavity

[a]Exhaustive experimentation shows that our algorithm frequently produces optimal solutions.

PROPOSED ALGORITHM

Sub-polygon of a Concavity:



- *Concavity line*, l_i divides the polygon into two sub-polygons lying one side of l_i and the main polygon on other side
- Each sub-polygon has at least two points on l_i, *start vertex* and *terminal vertex*
- Terminal vertices are determined using H_x and H_y

RULES FOR DECOMPOSITION

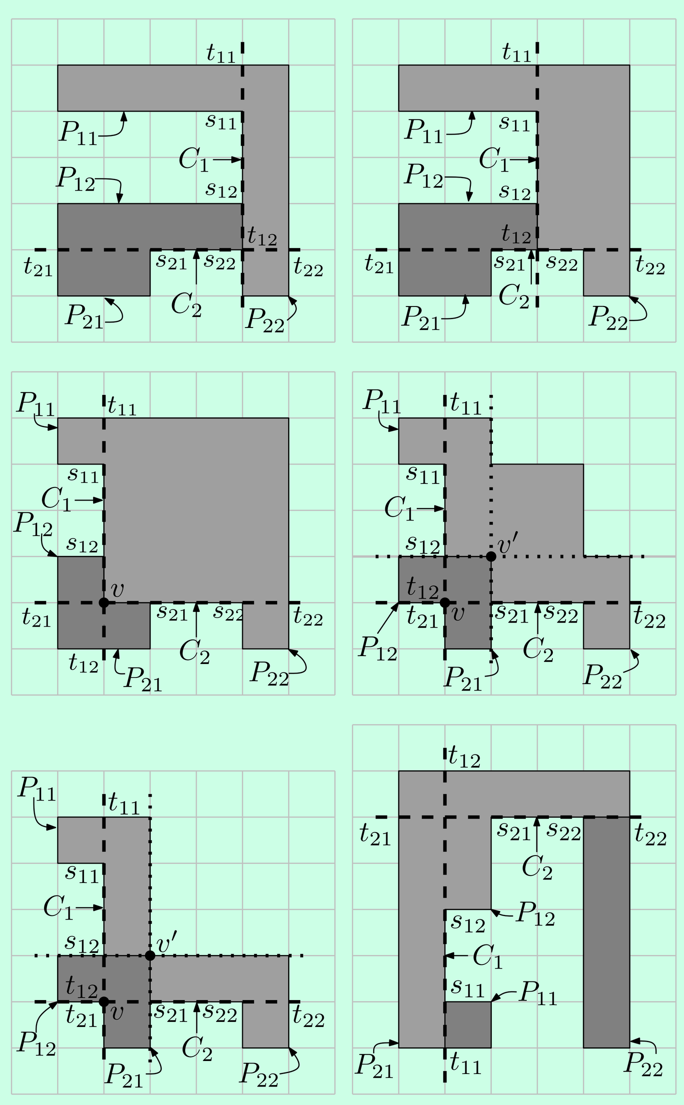
- Rules are applied to a pair of concavities at a time to obtain (sub-)optimality

Two Simple, Orthogonal, Consecutive Concavities:

- l_1 and l_2 corresponding to C_1 and C_2 are orthogonal and intersect at v
- no sub-polygon of any concavity contains the other concavity in full
- Extraction of one sub-polygon (by traversing from s_{12} to $v = t_{12}$) resolves both C_1 and C_2
- **Combined type of C_1 and C_2**: LB, BR, RT, TL

Cases:

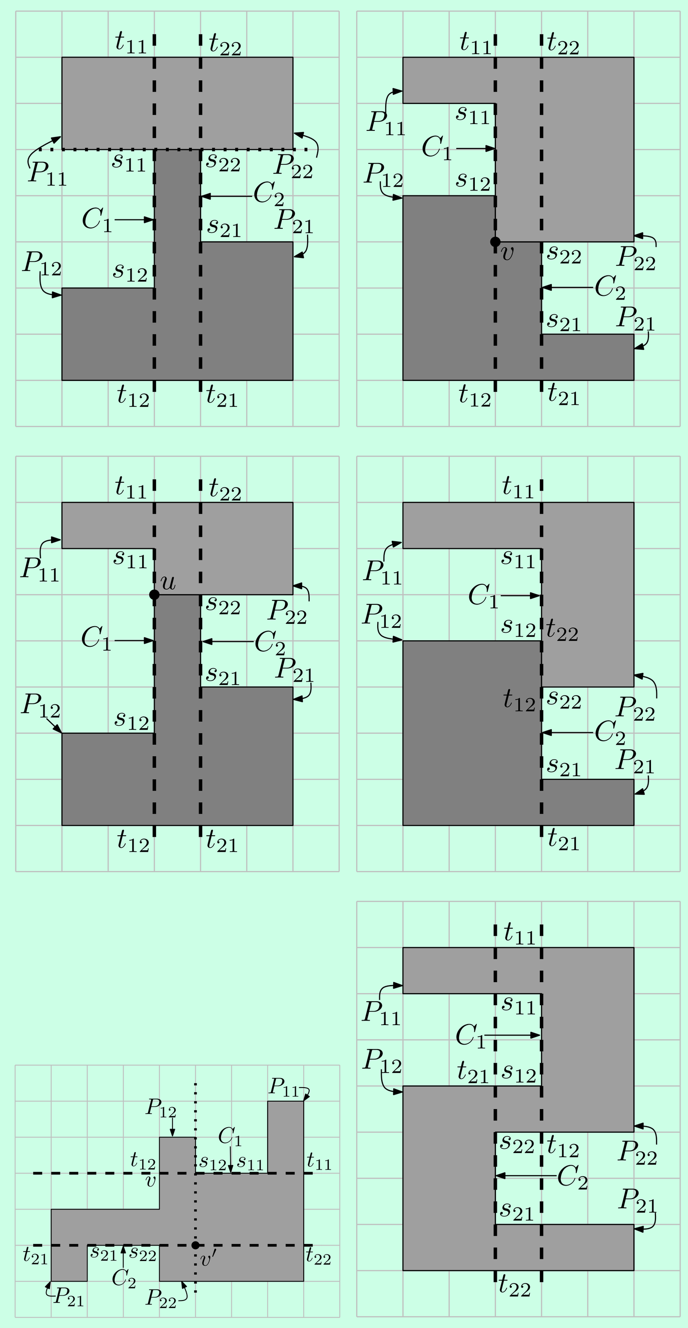
1. $v \in \{s_{11}, s_{12}, s_{21}, s_{22}\}$: v is present in L
2. v lies on the edge (s_{11}, s_{12}) or (s_{21}, s_{22}): v is inserted in L (using H_x and H_y)
3. v lies not on the boundary but inside A'_{in}: v is inserted in L as above
4. v lies on (the boundary of) or outside A'_{in}: Find v'. If it is inside A'_{in}, then one component is extracted, otherwise both P_{11} and P_{22} are extracted
5. If C_1 (C_2) lies entirely in one sub-polygon, say P_{21}, corresponding to C_2 (C_1), then both P_{11} and P_{22} are extracted



Rules of decomposition for $l_1 \perp l_2$

Two Simple, Parallel, Consecutive Concavities:

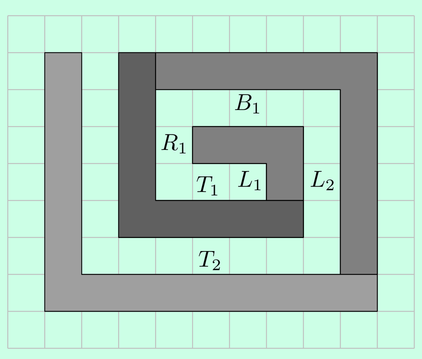
- If the projection of the edge (s_{11}, s_{12}) on l_2 (or the edge (s_{21}, s_{22}) on l_1) lies on or inside A'_{in}, then extraction of one sub-polygon resolves both C_1 and C_2
- Otherwise, P_{11} and P_{21} are extracted to resolve C_1 and C_2



Rules of decomposition for $l_1 \parallel l_2$

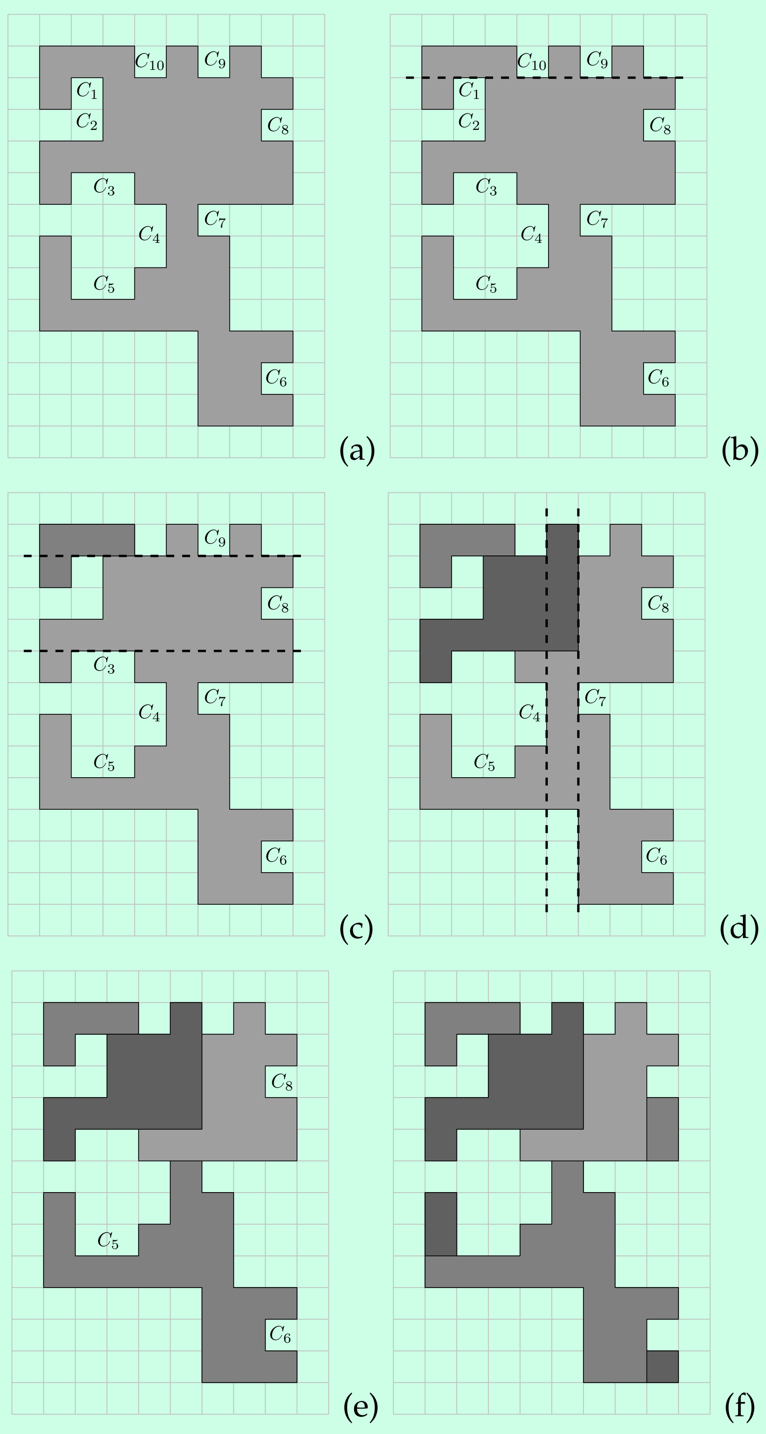
Compound Concavities:

- $t(>2)$ consecutive Type 3 vertices broken into $t-1$ simpler concavities, each consisting of two consecutive Type 3 vertices



Decomposition of compound concavity (three pairs solved in three steps)

DEMONSTRATION



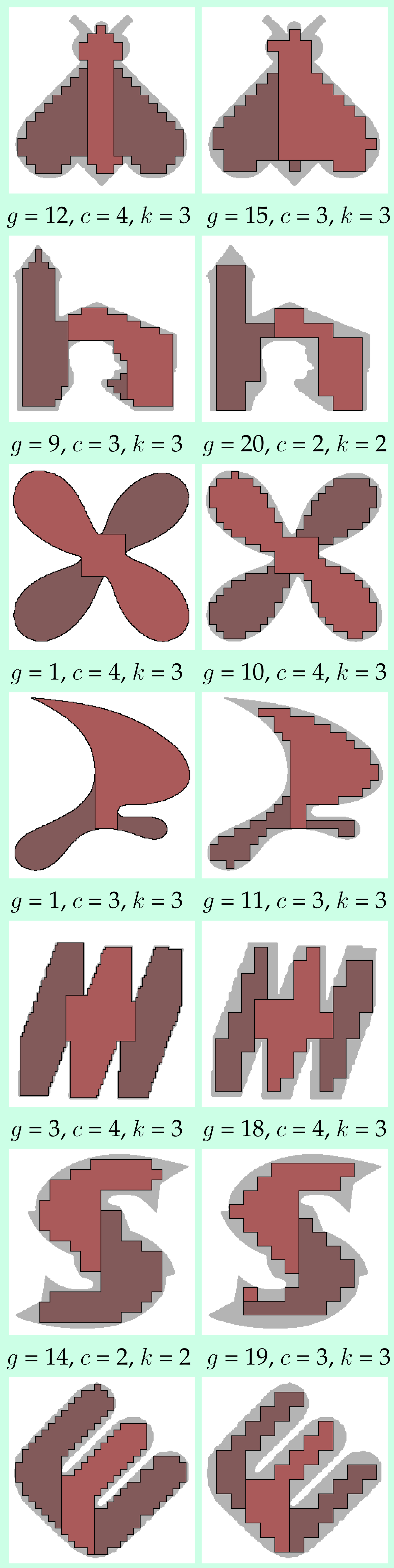
(a) (b) (c) (d) (e) (f)

TIME COMPLEXITY

- Stage 1: Construction of A'_{in}, L, L_c, H_x, H_y takes $O(n \log n)$ time
- Stage 2: Rules are applied which requires $O(n \log n)$ time
- Overall Complexity: $O(n \log n)$

EXPERIMENTAL RESULTS

- The count, k, for OCC depends on the grid size, number of concavities, and their orientation



$g = 12, c = 4, k = 3$ $g = 15, c = 3, k = 3$
$g = 9, c = 3, k = 3$ $g = 20, c = 2, k = 2$
$g = 1, c = 4, k = 3$ $g = 10, c = 4, k = 3$
$g = 1, c = 3, k = 3$ $g = 11, c = 3, k = 3$
$g = 3, c = 4, k = 3$ $g = 18, c = 4, k = 3$
$g = 14, c = 2, k = 2$ $g = 19, c = 3, k = 3$
$g = 9, c = 4, k = 3$ $g = 20, c = 4, k = 3$

CONCLUSION

- Efficient and robust algorithm
- Results shown are mostly optimal
- Application: shape analysis