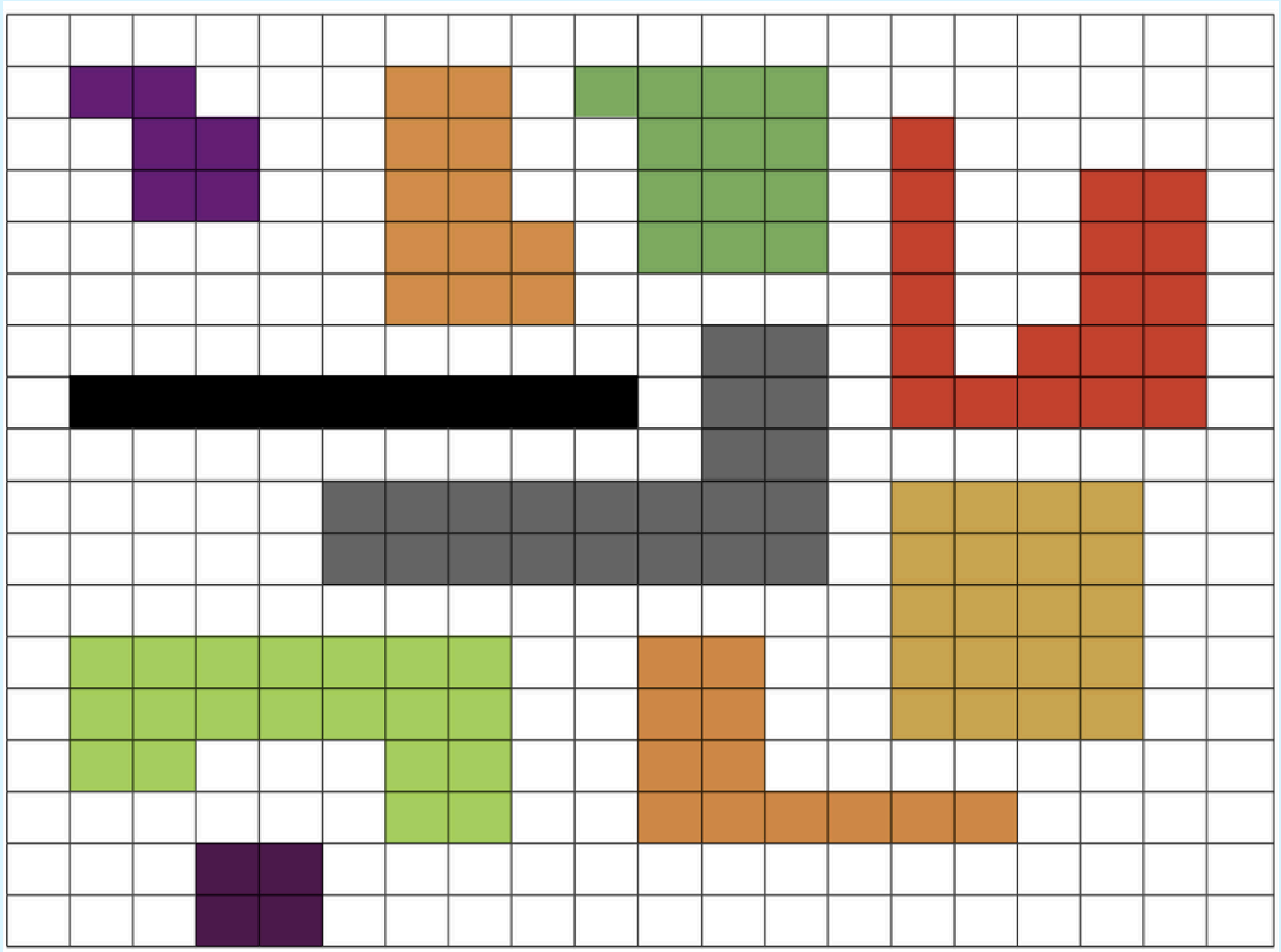
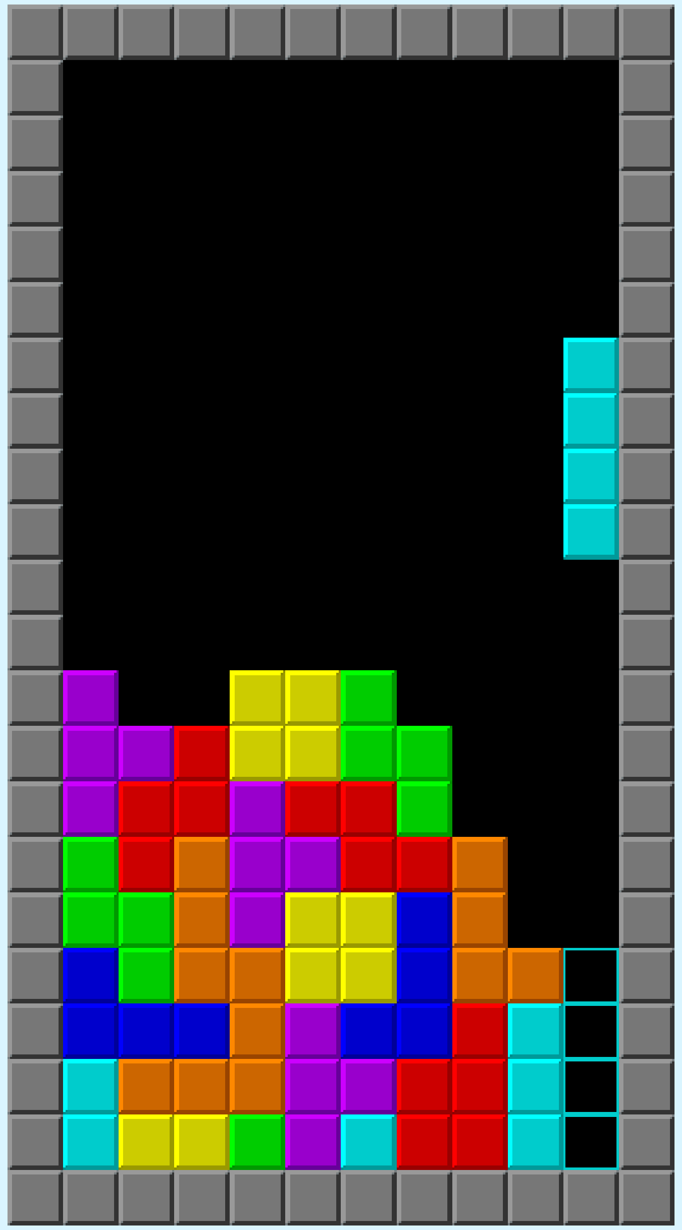
SQUARE PACKING

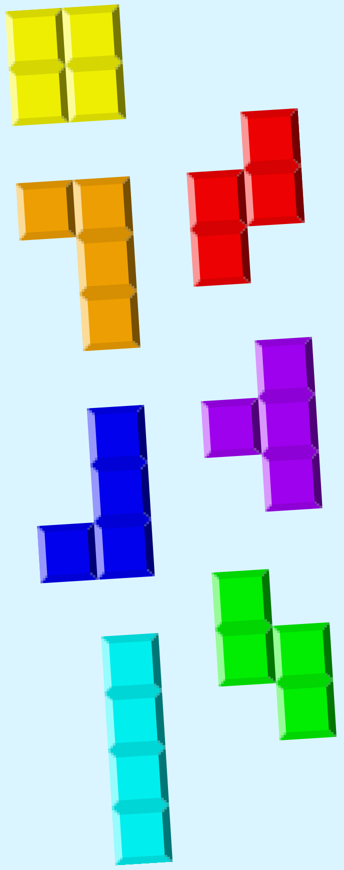
ARTUR SAHAKYAN

PROBLEM

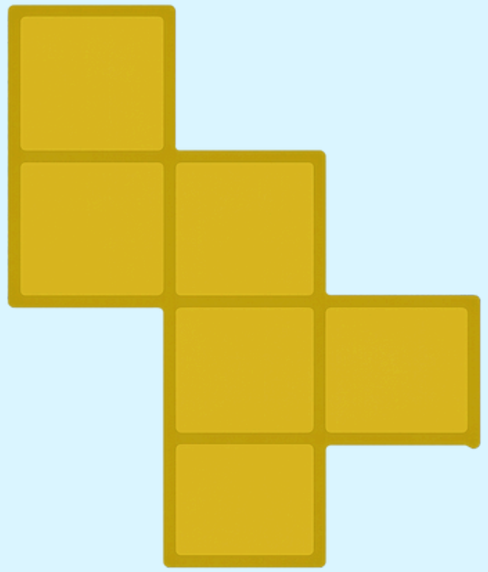


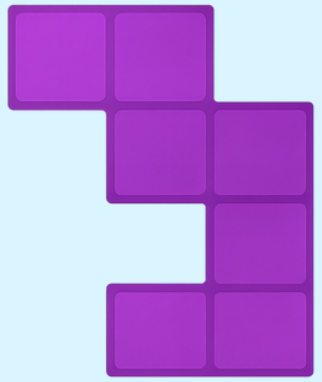


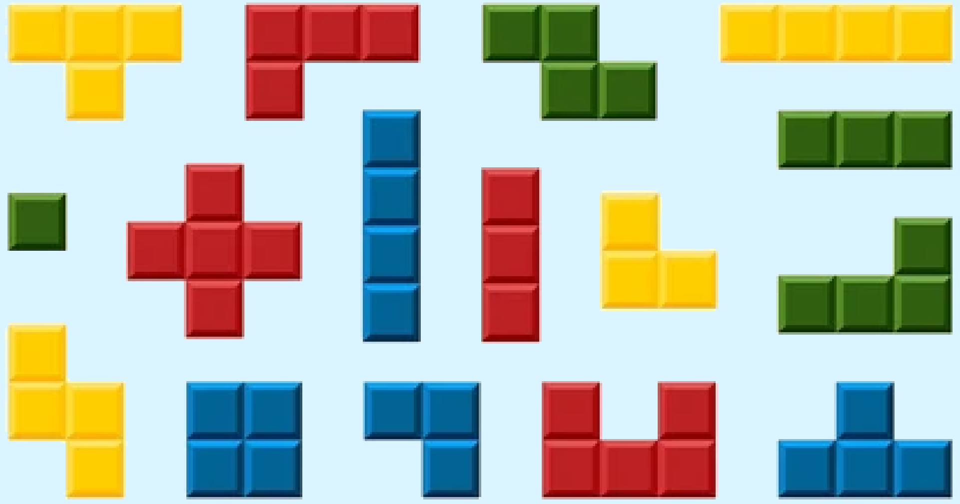
LOGICAL TO REPRESENT AS "LABELED" MATRIX



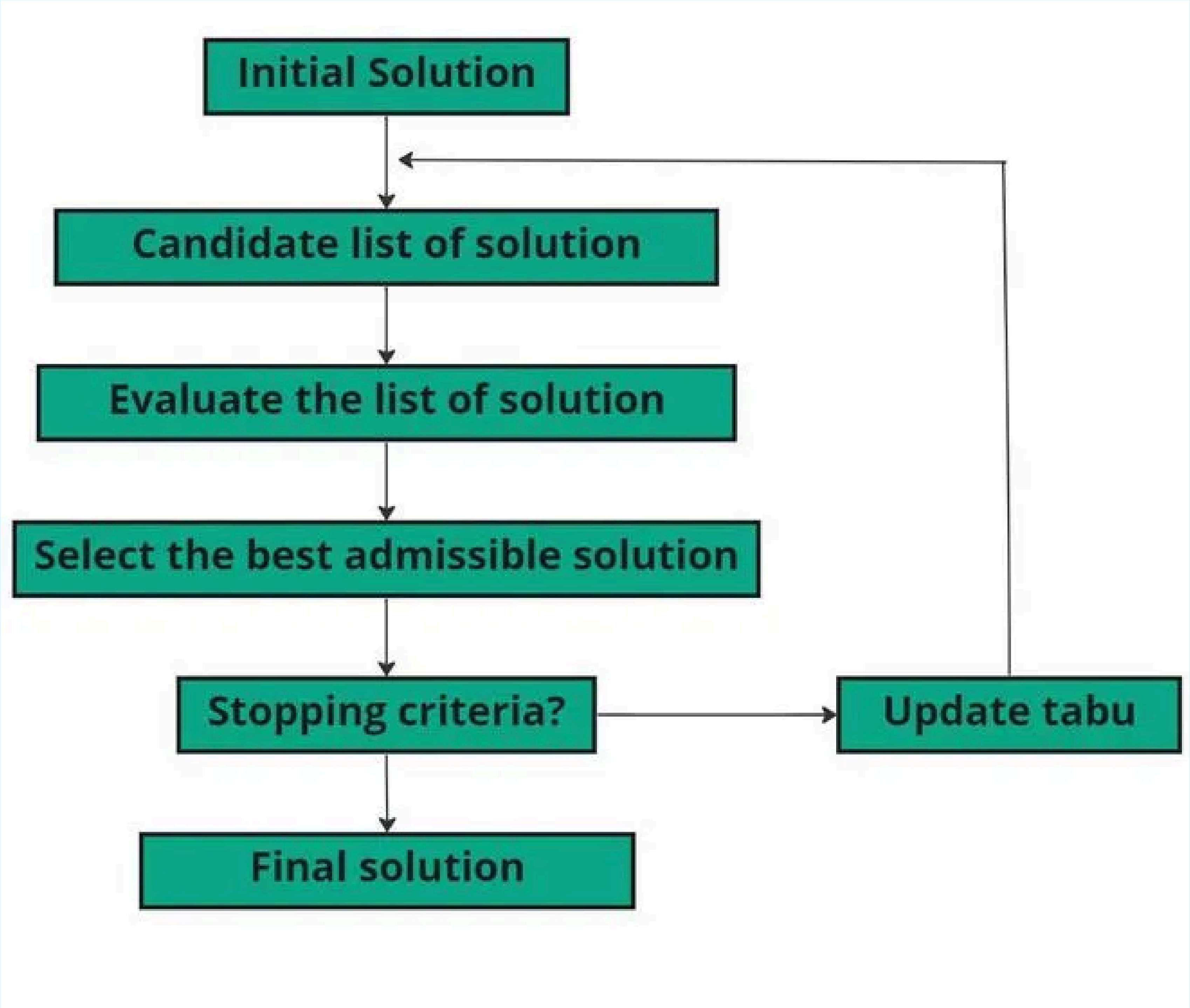
```
matrix = [
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,1,1,0,0,0,2,2,0,3,3,3,3,0,0,0,0,0,0,0],
    [0,0,1,1,0,0,2,2,0,0,3,3,3,0,4,0,0,0,0,0],
    [0,0,1,1,0,0,2,2,0,0,3,3,3,0,4,0,0,4,4,0],
    [0,0,0,0,0,0,2,2,2,0,3,3,3,0,4,0,0,4,4,0],
    [0,0,0,0,0,0,2,2,2,0,0,0,0,0,4,0,0,4,4,0],
    [0,0,0,0,0,0,0,0,0,0,0,5,5,0,4,0,4,4,4,0],
    [0,6,6,6,6,6,6,6,6,6,0,5,5,0,4,4,4,4,4,0],
    [0,0,0,0,0,0,0,0,0,0,0,5,5,0,0,0,0,0,0,0],
    [0,0,0,0,0,5,5,5,5,5,5,5,5,0,7,7,7,7,0,0],
    [0,0,0,0,0,5,5,5,5,5,5,5,5,0,7,7,7,7,0,0],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,7,7,7,7,0,0],
    [0,9,9,9,9,9,9,9,0,0,8,8,0,0,7,7,7,7,0,0],
    [0,9,9,9,9,9,9,9,0,0,8,8,0,0,7,7,7,7,0,0],
    [0,9,9,0,0,0,9,9,0,0,8,8,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,9,9,0,0,8,8,8,8,8,8,0,0,0,0],
    [0,0,0,10,10,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,10,10,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
]
```







METAHEURISTIC APPROACH WITH TABU SEARCH



TABU SEARCH APPLIED ON SQUARE PACKING

Step 1: Random Valid Placement

We randomly place each shape in the grid within the board bounds, even if it overlaps.

Step 2: Define the Fitness Function

We want to minimize the number of illegal cells, i.e., cells that are:Out of bounds,or overlapping with other shapes

fitness = 1000 + number_of_conflict_cells

(The 1000 ensures all imperfect packings are worse than any perfect one.)

Step 3: Explore the Neighborhood

We generate neighbors by:

Choosing one shape at random, moving it to a different random legal position on the board. This gives a new potential solution.

Step 4: Tabu List

To avoid cycling back to recent placements:

We store recent moves as (shape_index, old_position)

These moves are tabu (forbidden) for TABU_TENURE steps

Aspiration rule: we allow a tabu move if it improves the best solution ever found

This helps the algorithm:

- Escape local optima
- Avoid infinite loops
- Explore the search space better
-

Step 5: Update Best Solution

If a neighbor is better (lower fitness), it becomes the new current solution.

If it’s the best so far (even if TABU), we remember it.

Step 6: Repeat

We do this for a fixed number of iterations (say 500), and restart the process if needed.

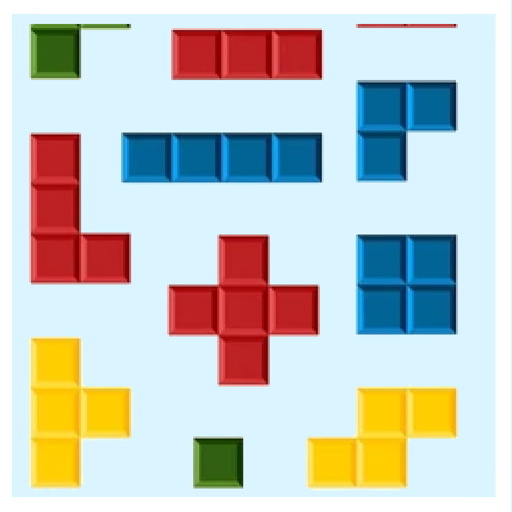
OVERLAPS, VIOLATIONS

WE CHECK FOR COLLISIONS, THERE ARE MANY METHODS TO DO SO

Search space has lower bound, [length of side of smallest square for packing, we can increment it in case of failure to pack]

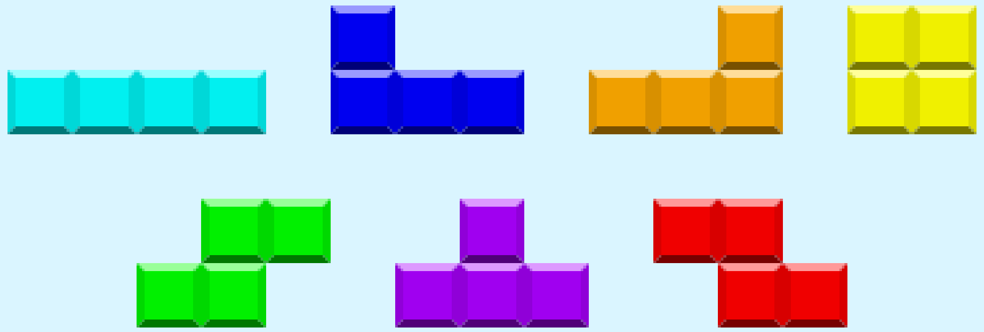
$$S(A) = \sqrt{A}$$

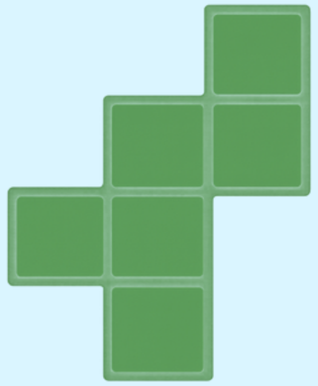
$$S(A) = \sqrt{A}$$



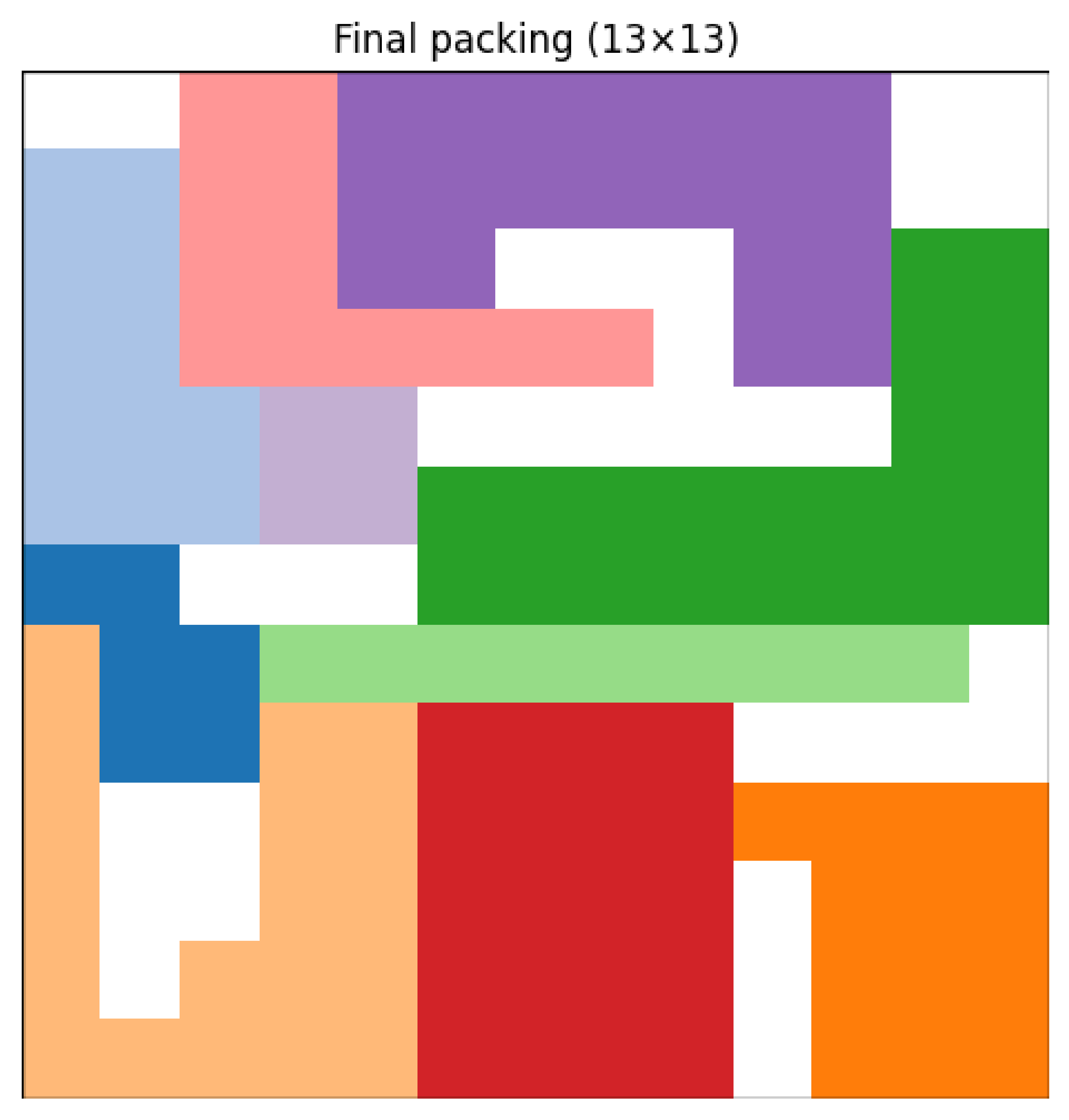

```python
def build_board(anchors, shape_list, S):
    ## blank board
    board = np.zeros((S, S), int)
    bad   = 0
    for (r0, c0), (k, sh) in zip(anchors, shape_list):
        for dr, dc in sh["cells"]:
            r, c = r0 + dr, c0 + dc

            ## cell out of bounds ??
            if r < 0 or r >= S or c < 0 or c >= S:
                bad += 1

            ## cell already occupied ??
            elif board[r, c]:
                bad += 1
            else:
                board[r, c] = k
    return board, bad
```

```python
def fitness(anchors, shape_list, S):
    _, bad = build_board(anchors, shape_list, S)
    if bad == 0:
        return 0.0
    return 1000 + bad
```



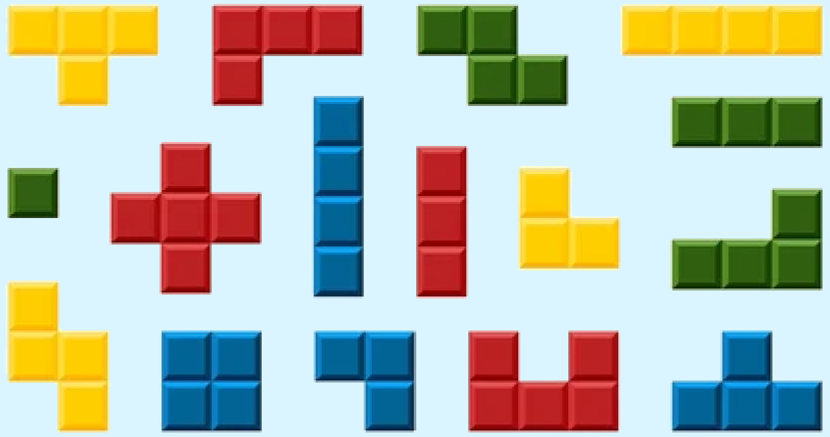


RESULT WITH LOW-COST PARAMETERS FOR TABU SEARCH



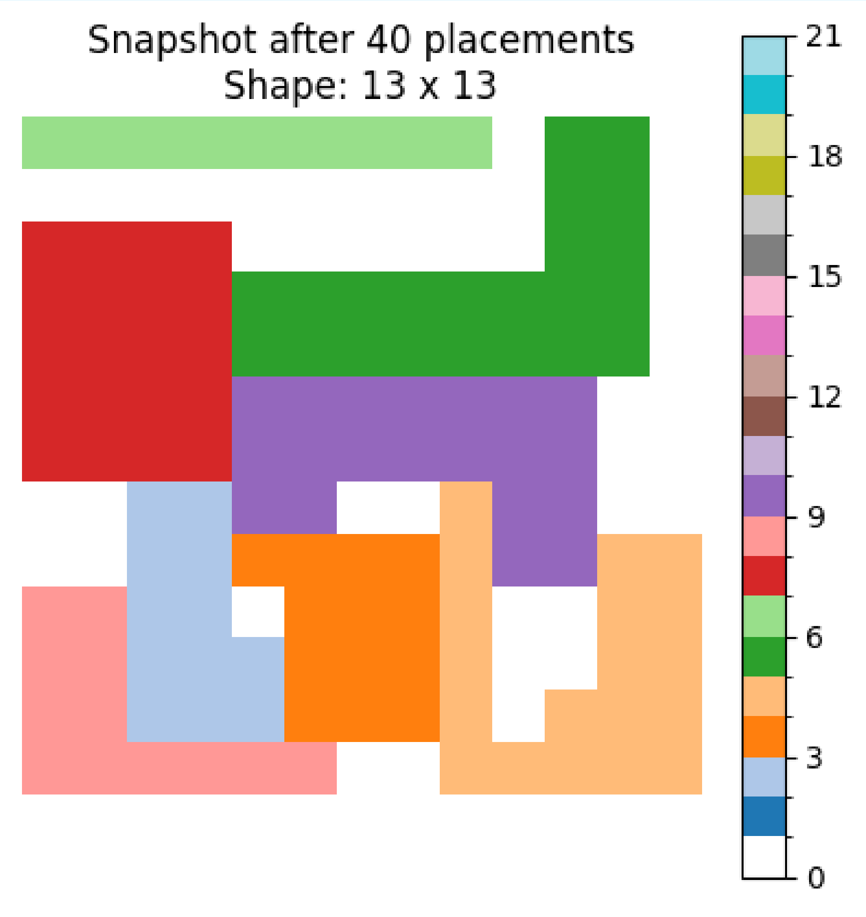
⚠ THIS IS HEURISTIC — WE FIND A LIKELY MINIMAL PACKING, NOT A PROVABLY MINIMAL ONE.

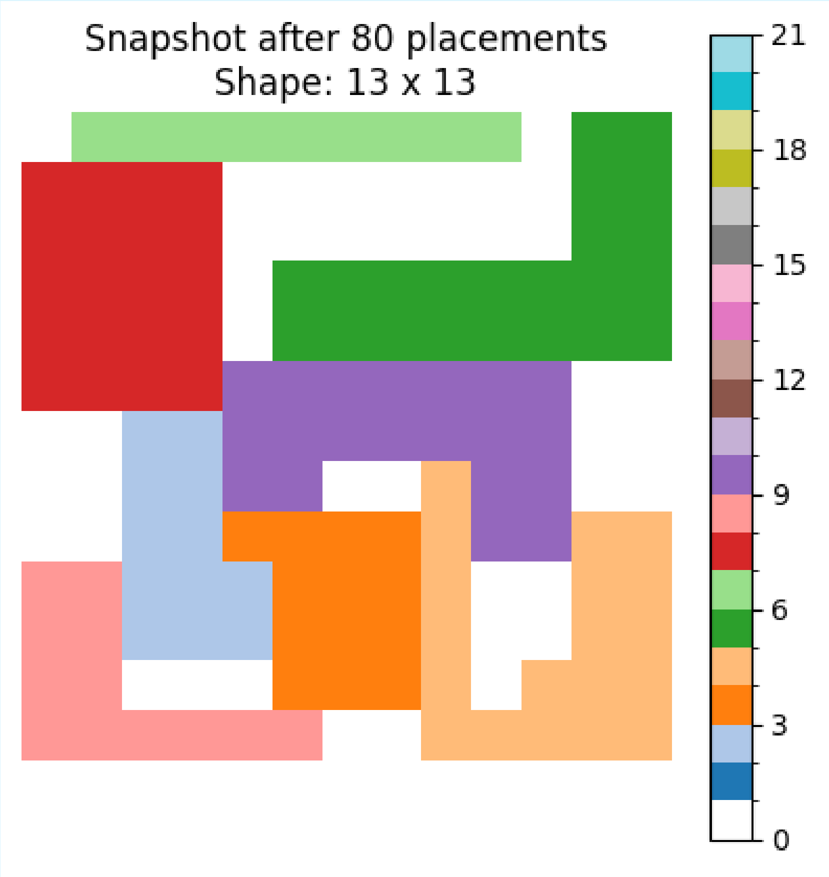
TABU SEARCH DOES USE THE SAME LOWER-BOUND STRATEGY TO START ITS SEARCH, BUT UNLIKE BACKTRACKING, IT CAN'T GUARANTEE OPTIMALITY — IT JUST TRIES TO FIND THE SMALLEST S WHERE A FEASIBLE PACKING CAN BE FOUND HEURISTICALLY.

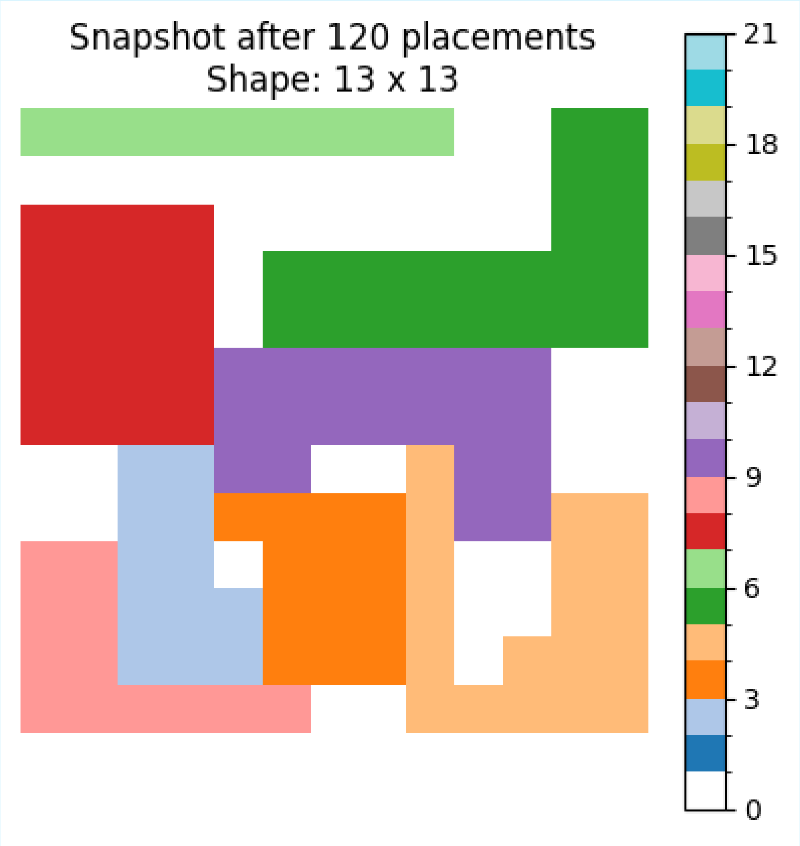


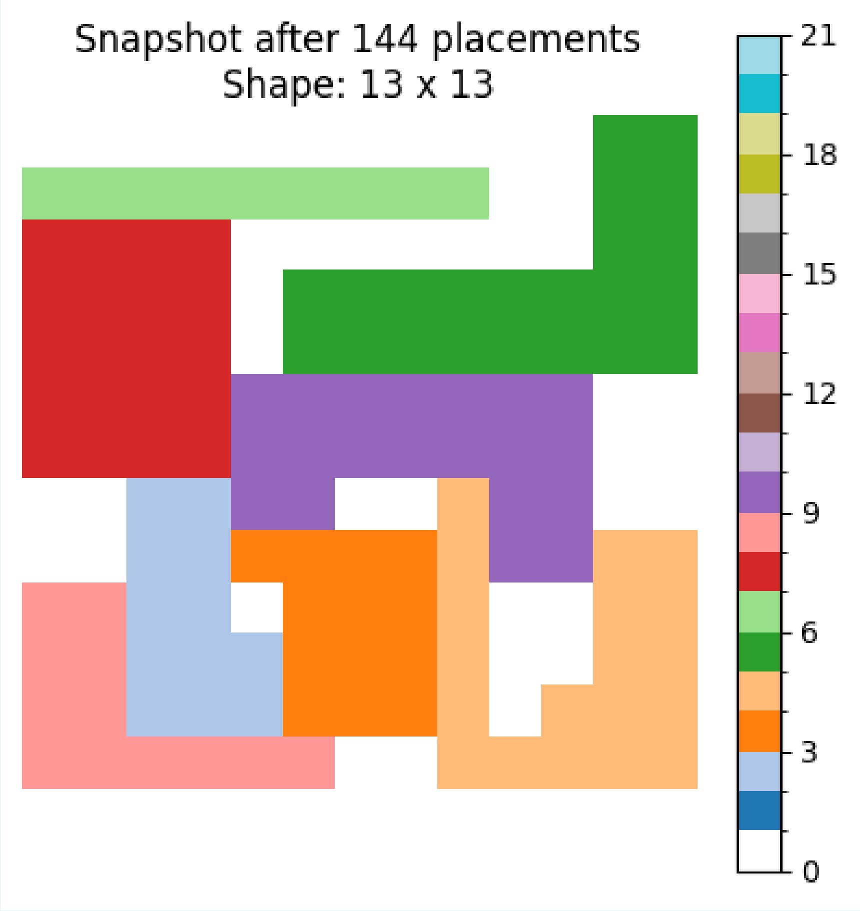
DETERMINISTIC SOLUTION

- SORT SHAPES
- START WITH A LOWER BOUND OF S = SQRT(A) AND INCREMENT IF NEEDED
- DO A DEPTH-FIRST SEARCH, PLACE LARGER SHAPES FIRST
- BACKTRACK IN CASE OF FAILING TO PACK THE NEXT SHAPE
- AFTER SOME N PLACEMENTS WE SEE NO IMPROVEMENT (NO REDUCTION IN SMALLEST SQUARE AREA)


Snapshot after 40 placements
Shape: 13 x 13
21
18
15
12
9
6
3
0


Snapshot after 80 placements
Shape: 13 x 13
21
18
15
12
9
6
3
0


Snapshot after 120 placements
Shape: 13 x 13
21
18
15
12
9
6
3
0


Snapshot after 144 placements
Shape: 13 x 13
21
18
15
12
9
6
3
0

COMPARISON

- ⏱ DETERMINISTIC SOLUTION COMPLETED IN 7.021 SECONDS
- ⏱ TABU SEARCH COMPLETED IN 4.631 SECONDS

THANK YOU