AMS 545 Project

Sophia Nolas

5/2/23

Packing Anchored Rectangles

Packing Anchored Rectangles

Article in Combinatorica - July 2011

Adrian Dumitrescu (University of Wisconsin, Milwaukee)
Csaba D Toth (California State University, Northbridge)

Rectangle Packing Problem

Problem expressed in the form of a game, popularized by Peter Winkler:

- Alice chooses n points in the unit square (including the origin)
- Bill chooses an axis-parallel rectangle anchored at each point s $\in$ S; that is, the lowest left point of the rectangle is the point s

Rectangle Packing Problem

Conjecture

For any finite set S in the planar unit square, Bill can choose rectangles that cover, all together, half of the unit square (net area $= 0.5$).

It has not been proven that Bill can always score a positive constant area coverage!

Rectangle Packing Problem

Example

If Alice chooses the points equally spaced along the diagonal, Bill can cover at most $\frac{1}{2} + \frac{1}{2n}$; so if $\epsilon > 0$ is fixed, the constant $\frac{1}{2} + \epsilon$ cannot guaranteed to be covered - choose n large enough that $\frac{1}{2n} < \epsilon$.

Rectangle Packing Problem: The Algorithms

This paper proposes two algorithms that give Bill a guaranteed score of 0.09121 area covered.

1. GREEDYPACKING
2. TILEPACKING

In this project, I implemented the Greedy Packing algorithm, which always yields greater covered area than Tile Packing.

1. GREEDYPACKING

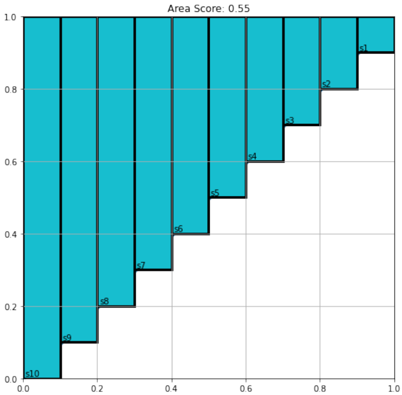
- Sort the points in descending order from (1,1) to (0,0) such that for points s_i, s_j, if $i < j$, then $x_i + y_i > x_j + y_j$ (and note that $s_n = (0, 0)$)
- Starting with the point furthest from (0,0), s_1, choose the largest rectangle so that s_1 is the lower left vertex
- For each s_i in increasing order, choose the largest anchored rectangle that does not overlap any previous rectangle.

1. GREEDYPACKING

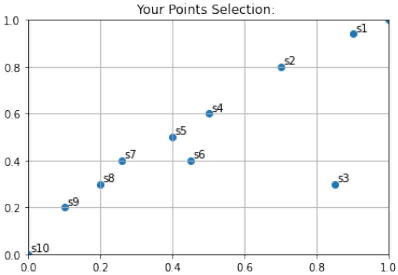
I chose the largest rectangle at each set by simply naively testing every possible rectangle, that does not overlap a previous one. Choose a height from the current point to the height of any previous point, and draw a rectangle rightward until it hits the x value of another rectangle that is lower than that height.

1. GREEDYPACKING

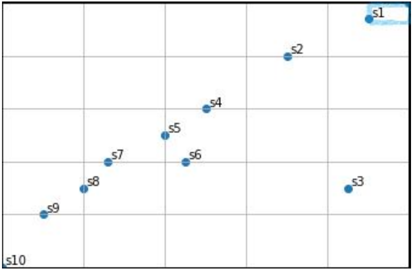
Greedy Packing Rectangles:



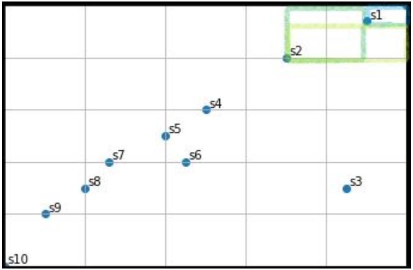
1. GREEDYPACKING



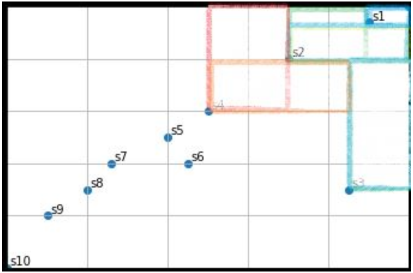
1. GREEDYPACKING



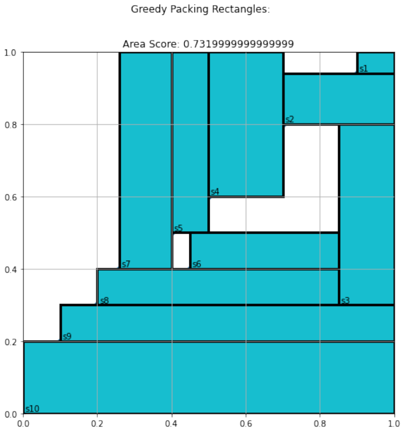
1. GREEDYPACKING



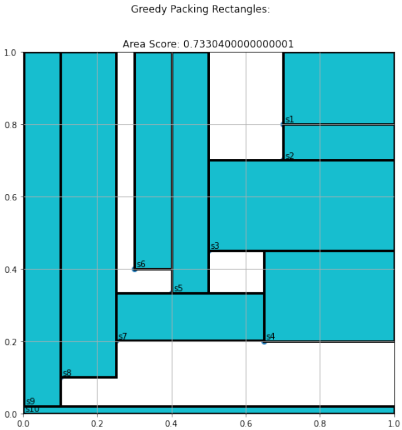
1. GREEDYPACKING



1. GREEDYPACKING



1. GREEDYPACKING



Runtime Analysis

Time and Space implementation

Tile Packing: O(nlogn) time, O(n) space
Greedy Packing: O(n^2) time, O(n) space

1. Time
   1. Greedy Packing: sort points in order of distance from origin; for each s_i of n points, compare $n - i - 1$ rectangles choose largest; O(n(n-1)/2)
   2. Tile Packing: sort points in order; compute dominant rectangles; then search through the binary tree holding the dominant rectangles, and choose the largest rectangle in each
2. Space: both only need to store the n points, x and y values