GAntenna Cover Algorithm

April 6, 2017

Problem Statement

Given the following:

- An unlimited supply of sensors, each of which can have multiple antennas, such that :
  - All antennas have the same aperture angle.
  - A sensor has at least one antenna.
  - A given sensor can have multiple antennas.
  - Each antenna belonging to a given sensor has the same sensitivity but each antenna assigned to a sensor may have a different roatation (azimuth) angle.
  - Different sensors may have antennas having different sensitivities.
  - Sensors can each have different numbers of antennas.
- A set of *possible_ locations* in 2-d space where sensors may be placed sorted in z-order (also known as Morton order). Call the lines joining these points in order the *Coastal Line* (CL).
- An set of *interference_ knot_ points* sorted in z-order. Call the lines joining these points the *Interference Contour* (IC).
- CL and IC do not intersect.

We call the simple polygon constructed by adding edges from the extremities of IC to CL, the *Coverage Region* (CR). Our goal is to cover CR such that:

- The the entire area of CR is covered to within a defined tolerance (i.e. the tolerance is some small number < 1 denoting the fraction of the area not covered by any antenna).
- The region outside CR covered by the antennas is minimized. We call this the *Excess Region* (ER).

- The total area of the cover (i.e. the sum of all the areas covered by each antenna) is minimized.
- No two sensors are closer to each other than some defined minimum distance *min_ distance*

Determine the following:

- Optimal placement of the sensors
- Orientation of the antennas (i.e. azimuth angle with respoect to the horizontal).
- Calibration of the antennas (i.e. "optimum" detection range of the antennas at each location where they are placed).

Algorithm High Level Description

Inputs:

- The sorted set of *possible_ centers*
- The sorted set of *interference_ knot_ points*
- The *min_ distance* specifying the minimum separation between two sensors.
- A family of concetric *detection_ coverage* curves definining the coverage regions for different antenna sensitivities all oriented in the same direction. These antenna curves are closed convex curves centered at *(0,0)* that, in general do not pack into a circle without overlap.

Algorithm

1. Construct the *coverage_ region* (CR):
   (a) Constuct a multi-line segment consisting of the points in *possible_ centers* joined in order.
   (b) Construct a multi-line segment consisting of the *interference_ knot_ points* joined in order.
   (c) Complete the simple polygon to generate CR.
2. Generate a rectangular grid of points such that each point of the grid is entirely within the region CR. (We start with some default grid spacing. This may be adjusted in subsequent steps.) We call the grid of points thus generated the *interference_ set*.

3. Generate a greedy circle cover (isotropic antenna) for *(possible_ centers, interference_ set, minum_ separation)* using algorithm *min_ area_ cover_ greedy*. This returns a set of centers chosen from *possible_ centers* and a set of radii of circles that cover the *interference_ set* such that antennas are spaced apart by at least *min_ distance* and the set of circles completely covers *interference_ set*

4. Place the antenna lobes (chosen from *detection_ coverage*) within the circles returned from step 3 such that the antenna lobes completely cover the intersection area between the circles and CR using algorithm *find_ antenna_ overlay_ for_ sector*.

5. Using simulated annealing with a cost function defined as the area of the convex hull of the cover, rotate the antenna cover lobes on their assigned sensors so that the cost function is minimized.

6. Eliminate redundant antenna lobes.

We now present the non-obvious steps in greater detail:

Algorithm 1 *find_max_min_circle*: Find the center and radius of a circle with center c and radius r for $c \in possible\_centers$ and $p \in interference\_set$ such that a circle centered at c covers p and has maximal radius among all such circles.

Parameters:

possible_centers: The possible centers in the 2-dimensional plane where sensors may be placed sorted in z-order.

interference_set: A set of points to be covered by cricles placed at *possible_centers*.

Output:

(c, r) where c is the center coordinate of the circle to be placed and r is the radius of a circle placed at c.

Procedure:

$closest\_centers :=$ **select** (**{** (c, p, r) **:** $c \in possible\_centers$
$\wedge\ p \in interference\_set$
$\wedge$ **r** $=$ **euclidean_distance(**c,p**)**
$\wedge$ **r is minimal }**)

max_min_circle **:=** **select** (**{** (c, r) **:**
$(c, p, r) \in closest\_centers$
$\wedge\ r$ **is maximal }**)

return max_min_circle

Algorithm 2 *min_area_circle_cover_greedy*: Find the greedy minimum area circle cover to cover a given *interference_set*.

Parameters:

interference_set: Set of interference points that we want to cover
possible_centers: Set of possible centers.
min_distance: Minimum distance permissible between circle centers.

Output:

cover : A set of circles $\{(c, r)$: the circles completely cover *interference_set*.$\}$

Procedure:

```
circle_cover = ∅
while interference_set is not ∅ do
    (center, radius) := find_max_min_circle ( interference_set,
                            possible_centers )
    covered_points := { p : p ∈ interference_set ∧ p is inside circle(c,r) }
    // Check if this center already exists in our cover
    if ∃ (c, r_1) ∈ cover : c = center then
        circle_cover := circle_cover - {(c, r_1)} ∪ {(c, max(radius, r_1))}
    else
        circle_cover := circle_cover ∪ (center, radius)
    end if
    // remove centers that are closer to the chosen center than min_distance
    possible_centers := possible_centers - { c: c ∈ possible_centers
                            ∧ euclidean_distance(c, center) ≤ min_distance
                            }
    // Prune the interference set.
    interference_set := interference_set - covered_points
end while
return circle_cover
```

Algorithm 3 *find_antenna_overlay_for_sector* : Position the antenna lobes within a section bounded by a circle (which is part of the *circle_cover*) and *coverage_region*

Parameters:

points_to_cover : A set of grid points that define the sector to be covered.

center : center of the circle that covers the sector.

radius : Radius of the circle that covers the sector.

detection_coverage_lobe : Antenna lobe of size 1.2*radius oriented in the horizontal direction. 1.2 is the excess overlap factor. This allows the lobe to overlap the circle.

antenna_angle : Aperture angle for the given antenna pattern.

Output:

angles : A vector of angles giving the orientation of the lobes that covers the sector.

Procedure:

```
//The number of discrete angles to check.
// incuded_angle is the angle obtained by intersecting the
// detection_coverage_lobe with the circle((0,0), radius)
// find_slice_angle finds this angle (not presented here).
included_angle := find_slice_angle(radius, detection_coverage_lobe)
// We check for cover by rotating to a discrete set of points.
npatterns := int(2*π / included_angle)
delta_angle := 2*π / npatterns
// Rotate and translate the lobe to cover the sector
// increments of delta_angle
rotated_lobes = { (angle, lobe) :
                  angle = k*delta_angles
                  ∧ lobe = rotate_and_translate(detection_coverage_lobe,
                           center,k*delta_angle)
                  ∧ (0 ≤ k < npatterns) }

angles = ∅
while points_to_cover != ∅ do
   // points_covered_by (lobe) is the set of
   // points covered by a lobe.
   selected_pattern := select( { p ∈ rotated_lobes :
                    |points_covered_by(points_to_cover, p.lobe)|
                         is maximal } )
   lobe_cover := points_covered_by(points_to_cover, selected_pattern.lobe)
   angles := angles ∪ {selected_pattern.angle}
   points_to_cover := points_to_cover - lobe_cover
end while
// Return the orientation of the lobes.
// Note that the result will contain redundant lobes and overlap
// Which must be removed by simulated annealing at a later setp
return angles
```

Algorithm 4 *min_antenna_cover_greedy* : Find the antenna cover for an *interference_set* given a set of concentric antenna detection lobes, a set of *possible_centers* where sensors may be placed and a *minimum_distance* of separation between sensors.

Parameters:
interference_set: Set of interference points that we want to cover
possible_centers: Set of possible centers.
min_distance: Minimum distance permissible between circle centers.
detection_coverage: An array of non intersecting concentric detection coverage lobes polygon for the antenna.
Each polygon has the same aperture angle and is oriented in the horizontal direction.

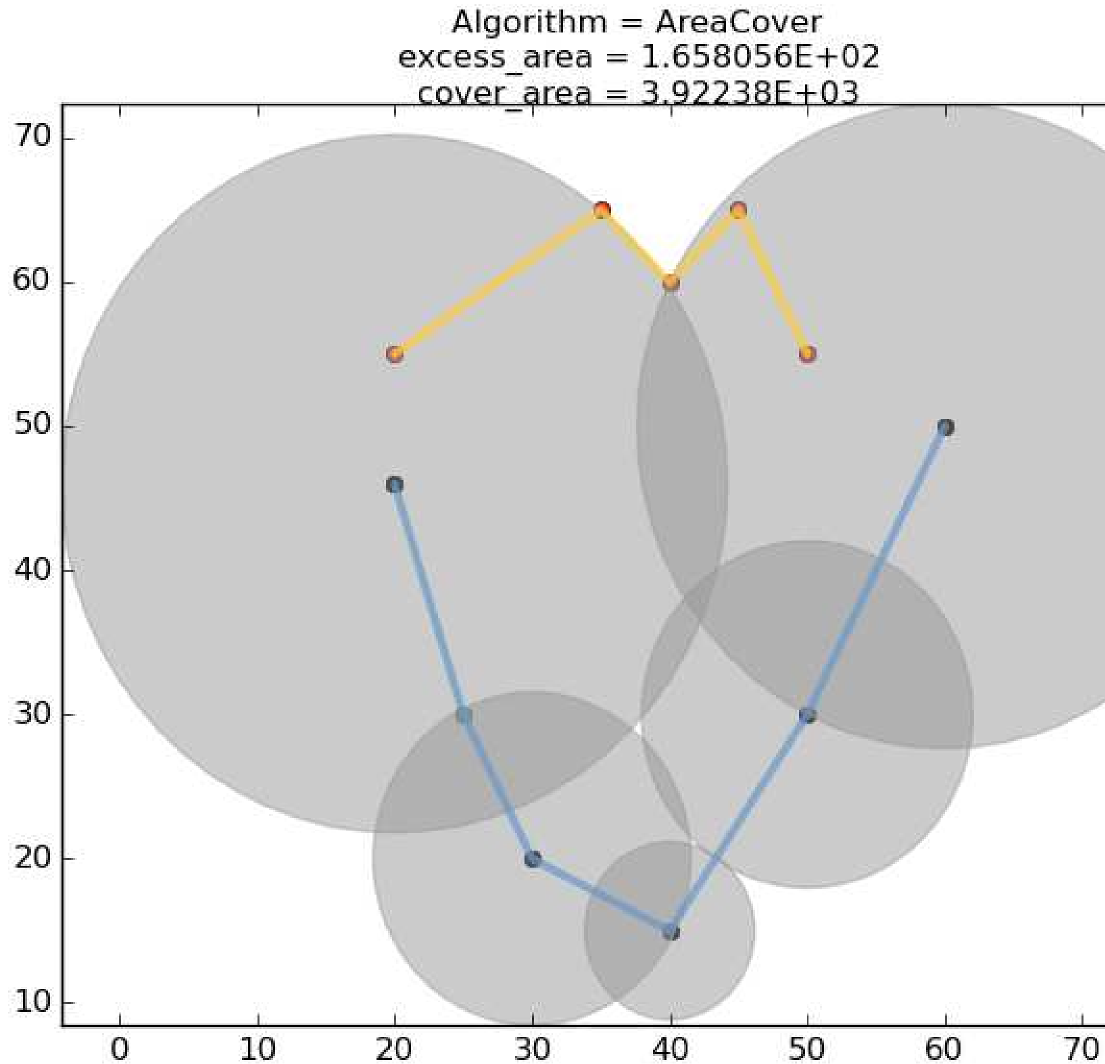
Output:
$\{(center, lobe, \{angles\})\}$: A set containing ($center$,$lobe$,$\{angles\}$) for $lobe \in detection\_coverage$ identifying the location, antenna lobe and azimuth angles of the lobes placed at $center$.

Procedure:

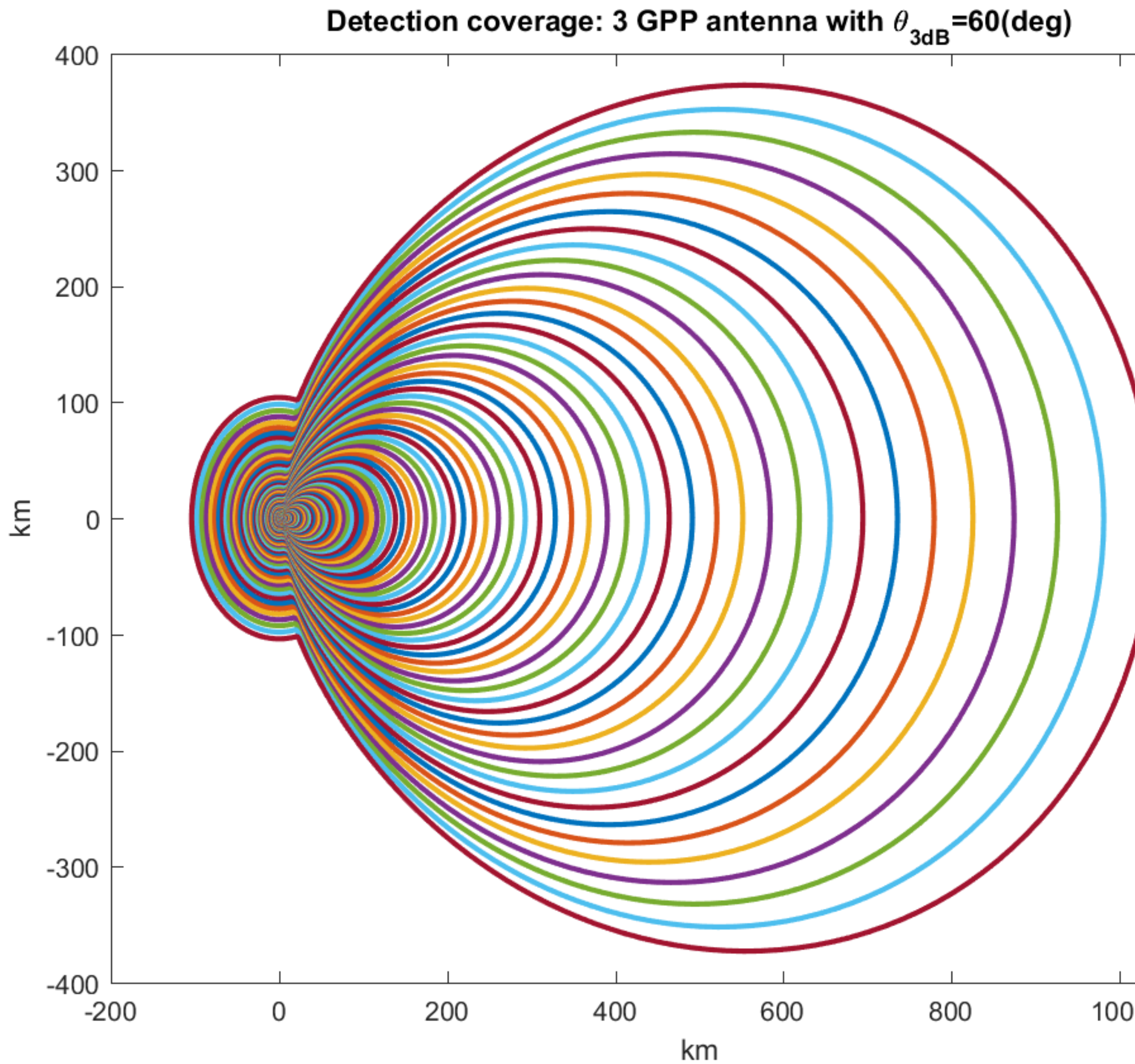
// cover is a set of circles that covers the *interference_set*
$cover := min\_area\_circle\_cover\_greedy(interference\_set,$
$possible\_centers, min\_distance)$

$antenna\_cover := \emptyset$
for C $\in cover$ **do**
  $lobe :=$ **select**(**{** $lobe$ **:** $lobe \in detection\_coverage$
   $\wedge\ lobe.radius \geq 1.2 * C.radius$
   $\wedge\ lobe.radius$ **is minimal }**)
  // **p is a point in 2-d space. The inside predicate is a point**
  // **in polygon test.**
  $points\_to\_cover =$ **{ p : p** $\in interference\_set \wedge$ **inside(C,p) }**
  $sector\_antenna\_cover$ **:=** $find\_antenna\_overlay\_for\_sector(points\_to\_cover,$
   $C.center, C.radius, lobe)$
  $antenna\_cover = antenna\_cover \cup$
   {$(C.center, lobe, sector\_antenna\_cover)$**}**
end for
return *antenna_cover*

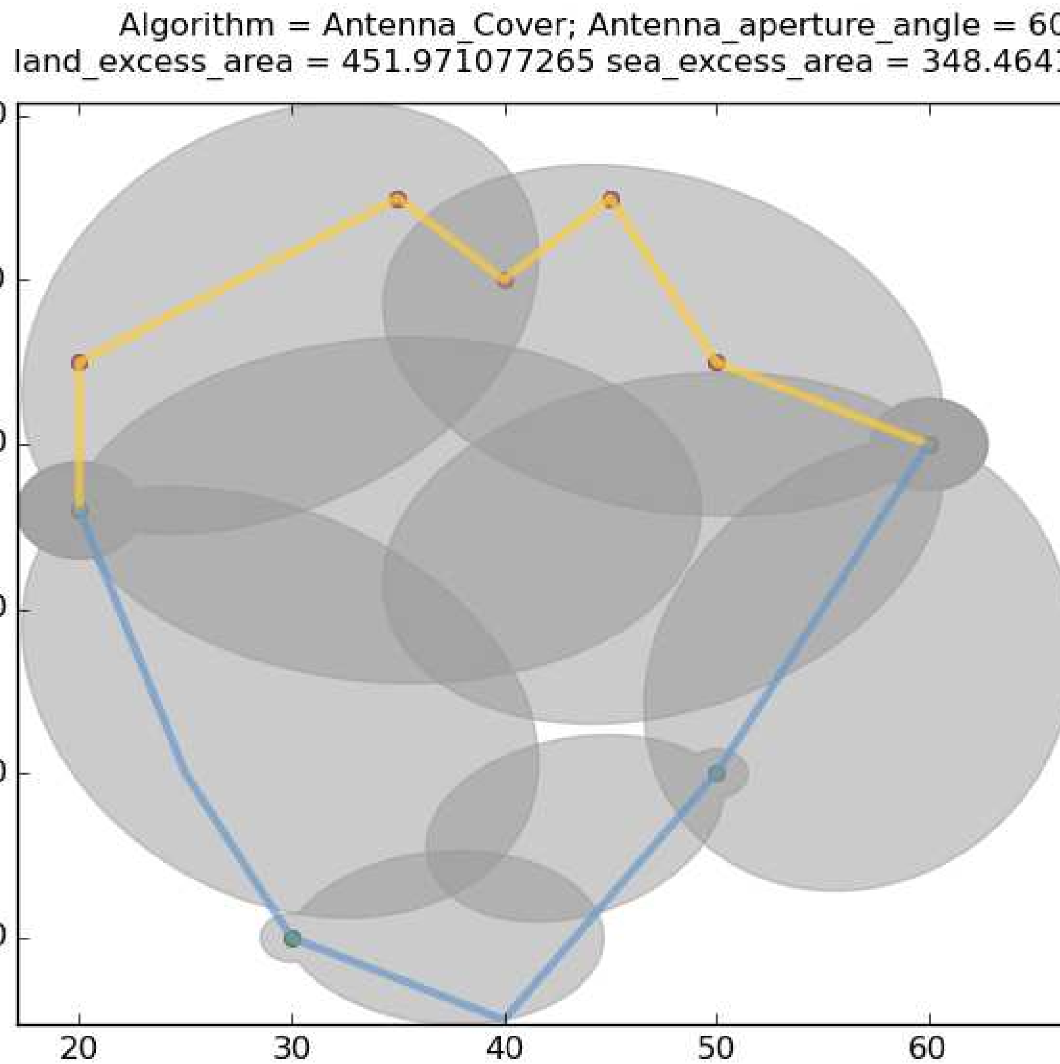
Pictorially, after step 3, we achieve the following circle cover.



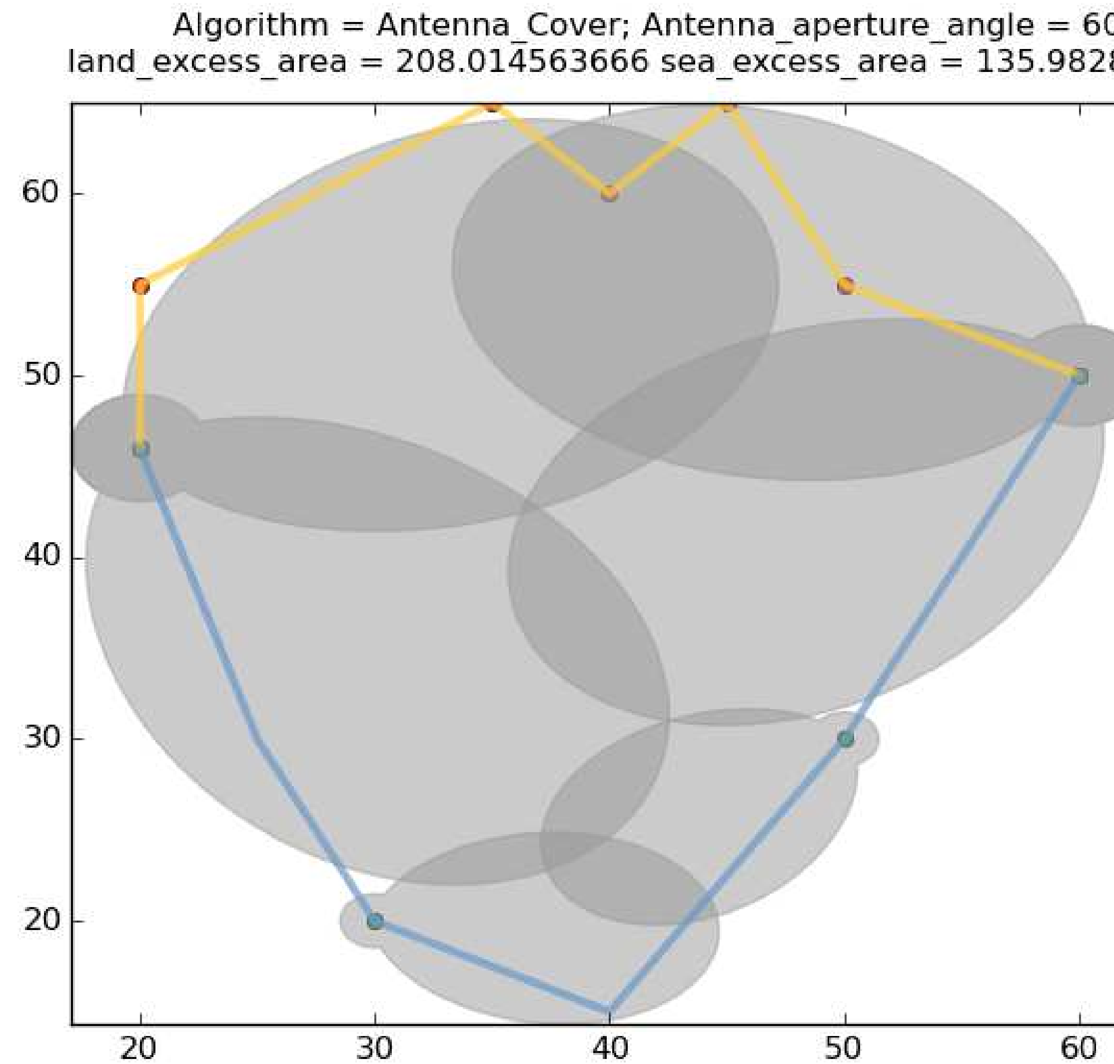
The detection_coverage curves are depicted pictorially as follows:



After overlaying the detection coverage curves on the circle_cover and overlaying, we get the following result. Note that small regions are uncovered because we eliminate lobes that cover less than threshold points on our interference_set :



After applying simulated annealing on the figure above, we get:



Algorithm = Antenna_Cover; Antenna_aperture_angle = 60
land_excess_area = 208.014563666 sea_excess_area = 135.9828
60
50
40
30
20
20
30
40
50
60