A Comparison of Antenna Placement Algorithms

Abhinav Jauhri

April 6, 2015

Motivation

- Placing new antennas requires a long, manual effort to complete an antenna placement study
- Systems with multiple antennas offer interference, and thereby reduce each antenna's efficiency
- Parasitic effects due to fixed or mobile plaform
- Frequency bands change over time requiring new antennas, and therefore need to find new placements

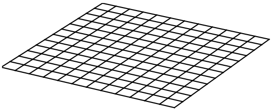
Outline of this talk

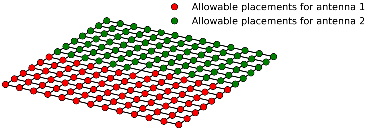
- Part 1: Introduction to the antenna placement problem
- Part 2: Description of stochastic algorithms, their properties and operators
- Part 3: Evaluation of test cases

Part 1: Introduction to the antenna placement problem

Antenna Placement Problem

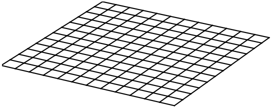
Given, platform $+$ allowable placements of antennas

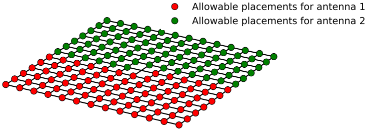




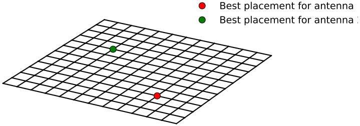
Antenna Placement Problem

Given, platform + allowable placements of antennas





Problem: Find best antenna placements to maximize gain and minimize coupling



Antenna Placement Problem

Given:

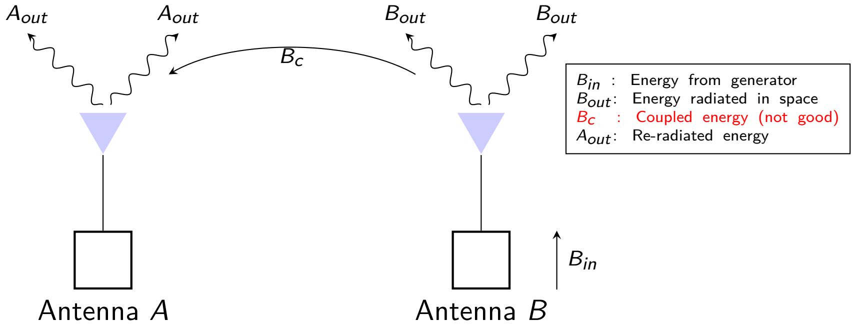
- platform P with its surface gridded such that end points represent possible antenna placements
- set of n antennas $A = A_1, A_2, \ldots, A_n$ such that $n > 1$
- for each A_i, L_i denote the set of allowable placements $\in \mathbb{R}^3$ such that $|L_i| = m_i$ and $\forall i, m_i > 1$; $L_i = \{(x_1, y_1, z_1) \ldots (x_{m_i}, y_{m_i}, z_{m_i})\}$

Problem: Find a set of n optimal antenna placements on P to maximize gain and minimize coupling

Question: How is a good antenna placement quantified in the context of platform and other antennas?

Mutual Coupling

When two antennas are in proximity, and one is transmitting, the second will receive some of the transmitted energy.



Minimize Mutual Coupling

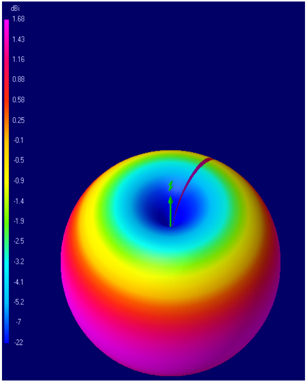
$$F_{MC} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} CP(A_i, A_j), \qquad (1)$$

where

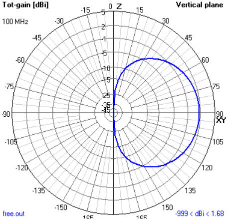
- $CP(\cdot,\cdot) \in \mathbb{R}$ is the coupling between two antennas, and computed using a simulator

Example: If $n = 3$, then $F_{MC} = CP(A_1, A_2) + CP(A_1, A_3) + CP(A_2, A_3)$

Radiation Pattern

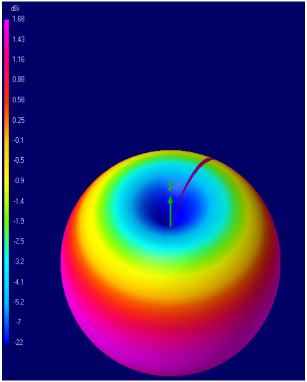


Free-space pattern without platform or other antennas

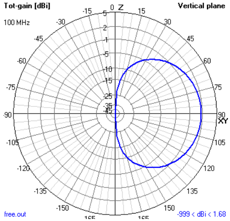


2D view of the free-space gain pattern

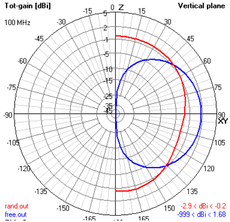
Radiation Pattern



Free-space pattern without platform or other antennas



2D view of the free-space gain pattern



In-situ pattern (in red) for random antenna placements

Minimize Difference in Radiation Pattern

$$F_{RP} = \sum_{i=1}^{n} \sum_{\theta=0}^{\frac{180^\circ}{S}} \sum_{\phi=0}^{\frac{360^\circ}{S}} \left(FSG_i(S\theta, S\phi) - ISG_i(S\theta, S\phi)\right)^2, \quad (2)$$

where

- S is the step size
- θ, ϕ spherical coordinates in degrees
- $FSG(\cdot,\cdot) \in \mathbb{R}$ is the free-space gain pattern computed by the simulator
- $ISG(\cdot,\cdot) \in \mathbb{R}$ is the in-situ gain pattern computed by the simulator

Fitness Evaluation

Find a placement such that F is minimal:

$$F = \alpha F_{MC} + \beta F_{RP}, \tag{3}$$

where α, β are adjustable weights for each of the objectives

Part 2: Stochastic Algorithms

Stochastic Algorithms

We will consider algorithms which are based on randomization principle.

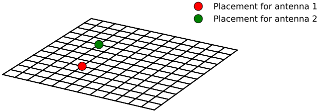
- Genetic Algorithm
- Evolutionary Strategy
- Simulated Annealing
- Hill Climbing

Each algorithm maintains a candidate solution or pool of candidate solutions called population

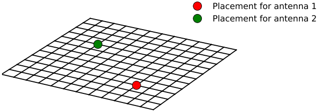
Stochastic Algorithms: Operand

Candidate solution or an **individual** is a member of a set of possible solutions.

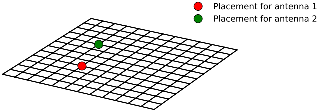
- Simulated Annealing and Hill Climbing maintain single individual



- Genetic Algorithm and Evolutionary Strategy maintain a population of individuals

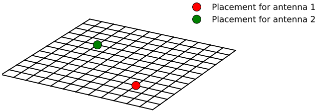


. . .

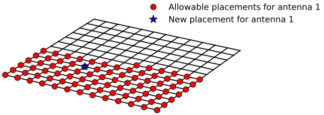


Stochastic Algorithms: Mutation Operator

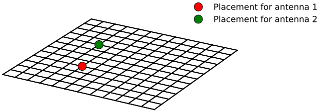
1. Given an individual, select an antenna uniformly at random, let's say antenna 1:



2. For antenna 1, select any other allowable placement:

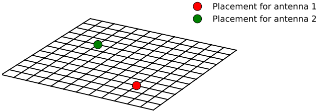


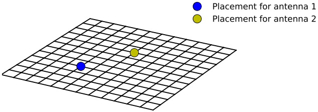
3. Change position for antenna 1 in individual:



Stochastic Algorithms: Crossover Operator

1. Select two individuals from population:

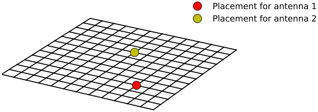


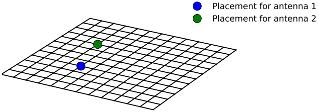


2. Select a crossover point, and swap placements prior to the point:

$$
\begin{array}{lc:c}
\text{ind}_1 : & {\color{red} A_1} & {\color{teal} A_2} \\
 & \downarrow \quad \uparrow & \\
\text{ind}_2 : & {\color{blue} A_1} & {\color{yellow} A_2}
\end{array}
$$

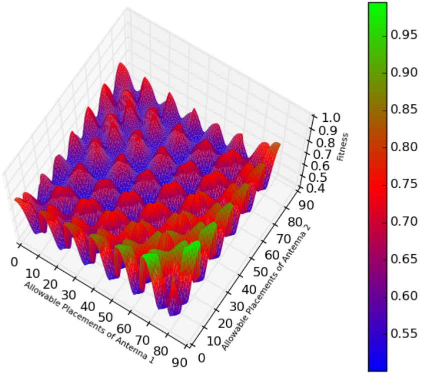
3. Two new offsprings created:





Question: Why use stochastic algorithms?

Multi-Modal Search Space



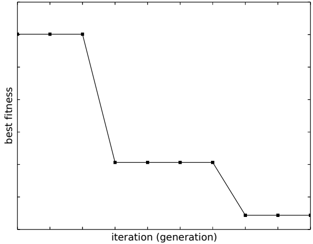
Search space for one of the test cases evaluated. There are multiple local minimas which makes convergence difficult. z-axis is the combined fitness F

Genetic Algorithm

P $\leftarrow$ generate p random individuals. Compute $fitness(\mathbf{ind}_i), i \in [1,p]$;
$i = 0$;
while $i < gen_{max}$ **do**
    Elitism: Select n_e fittest individuals to add to **P'** ;
    for $(p - n_e)/2$ *times* **do**
        `/* 'select' returns a pair of individuals */`
        $\mathbf{M} \leftarrow select(\mathbf{P}, 2)$;
        if $rand(0,1) < p_c$ **then**
            $\mathbf{O} \leftarrow crossover(\mathbf{M})$;
            Add O to P' ;
        else
            Add M to P' ;
    Uniformly select $p_m \cdot (p - n_e)$ individuals from P, and apply *mutation* operator to each ;
    Update $P \leftarrow P'$;
    Compute $fitness(\mathbf{ind}_i); i \in [1,p]$;
    Update $i \leftarrow i + 1$;

Genetic Algorithm

P $\leftarrow$ generate p random individuals. Compute $fitness(\mathbf{ind}_i), i \in [1,p]$;
$i = 0$;
while $i < gen_{max}$ **do**
    Elitism: Select n_e fittest individuals to add to **P'** ;
    for $(p - n_e)/2$ *times* **do**
        `/* 'select' returns a pair of individuals */`
        $\mathbf{M} \leftarrow select(\mathbf{P}, 2)$;
        if $rand(0,1) < p_c$ **then**
            $\mathbf{O} \leftarrow crossover(\mathbf{M})$;
            Add O to P' ;
        else
            Add M to P' ;
    Uniformly select $p_m \cdot (p - n_e)$ individuals from P, and apply *mutation* operator to each ;
    Update $P \leftarrow P'$;
    Compute $fitness(\mathbf{ind}_i); i \in [1,p]$;
    Update $i \leftarrow i + 1$;



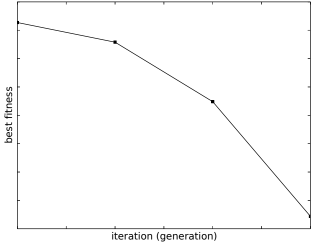
Plateaus suggesting stagnation of search

Evolutionary Strategy

```
P ← generate μ random individuals ;
i = 0 ;
while i < gen_max do
    Create λ/μ offsprings from each μ individuals by
    applying mutation operator;
    Add all offsprings to P ;
    Compute fitness(ind_i), i ∈ [1, λ + μ] ;
    Keep μ best individuals in P, and discard remaining
    λ − μ individuals ;
    Update i ← i + 1
```

Evolutionary Strategy

P ← generate μ random individuals ;
$i = 0$;
while $i < gen_{max}$ **do**
    Add all offsprings to **P** ;
    Create λ/μ offsprings from each μ individuals by applying *mutation* operator ;
    Compute $fitness(\mathbf{ind}_i), i \in [1, \lambda + \mu]$;
    Keep μ best individuals in **P**, and discard remaining $\lambda - \mu$ individuals ;
    Update $i \leftarrow i + 1$



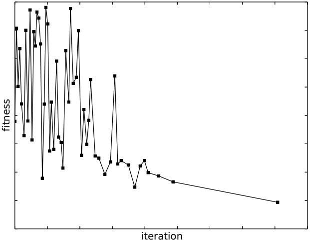
More mutations allow in-depth exploration of search space, and also makes rapid progress

Simulated Annealing

c $\leftarrow$ generate a random individual ;
$i = 0$;
while $i < i_max$ **do**
    n $\leftarrow$ $mutate(\mathbf{c})$;
    if $fitness(\boldsymbol{c}) < fitness(\boldsymbol{n})$ **then**
        if $rand(0,1) < e^{-\delta f / T}$ **then**
            `/* replace current individual by a higher fitness (less fitter) individual */`
            c $\leftarrow$ **n**
    else
        c $\leftarrow$ **n** ;
    $T \leftarrow T \cdot f_{cooling}$;
    $i \leftarrow i + 1$;

Simulated Annealing

c ← generate a random individual ;
$i = 0$;
while $i < i_m ax$ **do**
  n ← $mutate(\mathbf{c})$;
  if $fitness(\mathbf{c}) < fitness(\mathbf{n})$ **then**
    if $rand(0,1) < e^{-\delta f / T}$ **then**
      /* replace current individual by a higher fitness (less fitter) individual */
      c ← **n**
  else
    c ← **n** ;
  $T \leftarrow T \cdot f_{cooling}$;
  $i \leftarrow i + 1$;



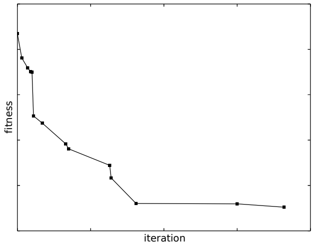
Fluctuation in fitness gradually reduces due to cooling

Hill Climbing

```
Initialize c ← generate a random inidividual ;
Compute fitness(c) ;
i = 0 ;
while i < i_max do
    n ← mutate(c) ;
    if fitness(n) < fitness(c) then
        c ← n
    i ← i + 1
```

Hill Climbing

Initialize **c** $\leftarrow$ generate a random inidividual ;
Compute *fitness*(**c**) ;
$i = 0$;
while $i < i_{max}$ **do**
    n $\leftarrow$ *mutate*(**c**) ;
    if $fitness(n) < fitness(c)$ **then**
        $\mathbf{c} \leftarrow \mathbf{n}$
    $i \leftarrow i + 1$



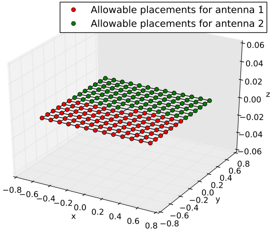
Greedy approach to accept only fitter(low fitness) individuals

Part 3: Evaluation of test cases

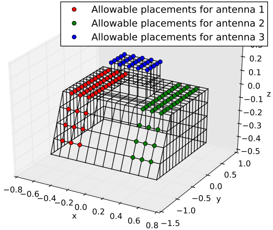
Experimental Setup

1. All test cases describe platforms which are replicas of real-world use cases like mobile devices, tanks, and cars
2. We use a popular *NEC2* simulator [1] to get fitness parameters
3. Evaluate the entire search space using an exhaustive algorithm to find the optimal antenna locations

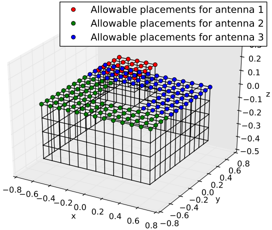
Experiments: Test Cases



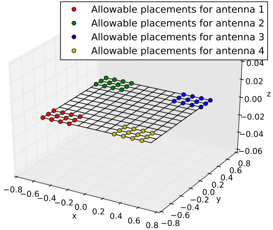
Test Case #1 with 7056 (84x84) allowable placements



Test Case #2 with 50625 (45x45x25) allowable placements

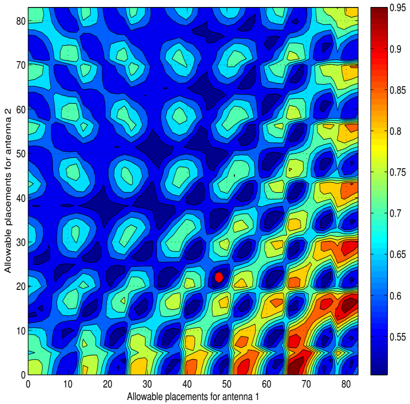


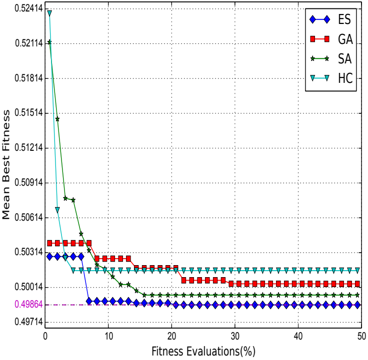
Test Case #3 with 126025 (71x71x25) allowable placements



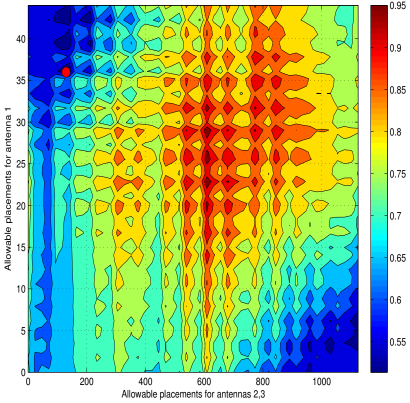
Test Case #4 with 20736 (12x12x12x12) allowable placements

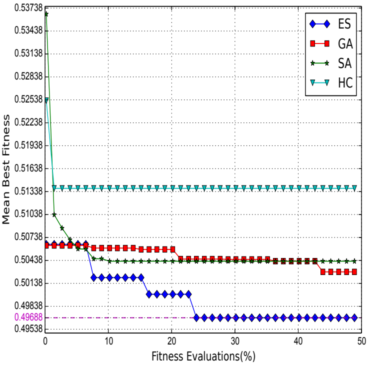
Results - Test Case 1



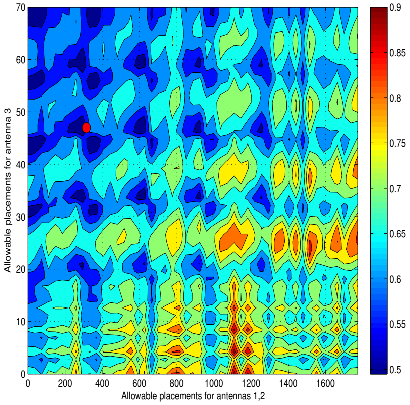


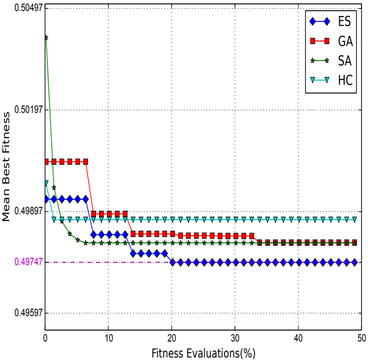
Results - Test Case 2



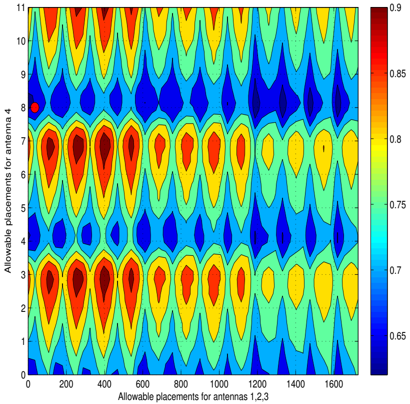


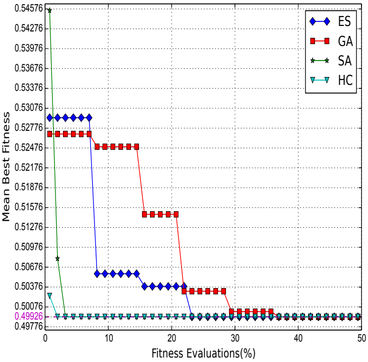
Results - Test Case 3



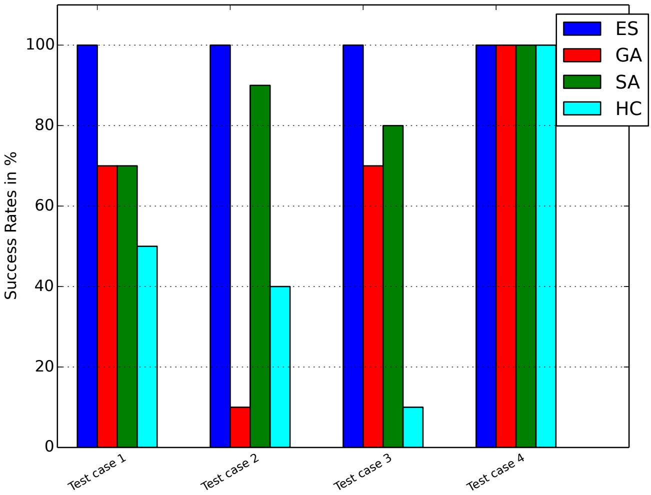


Results - Test Case 4





Results - Success Rates



Conclusion

- Formulation of the antenna placement problem
- Generic problem formulation to accommodate multiple antennas and platforms
- Optimal placements found using Evolutionary Strategy with at most 25% evaluations of search space