A Comparison of Antenna Placement Algorithms

Abhinav Jauhri

March 30, 2015

Contributions

- Formulation of the antenna placement problem
- Evaluation of standard stochastic algorithms on a real-world problem
- Able to achieve global optimum with as low as **25% evaluations** of search space

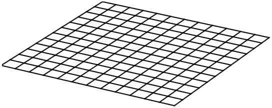
Outline of this talk

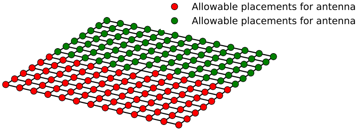
- Part 1: Introduction to the antenna placement problem
- Part 2: Description of stochastic algorithms, and formulation of an instance of antenna placment problem
- Part 3: Results of our experiments

Part 1: Introduction to the antenna placement problem

Antenna Placement Problem

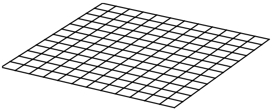
Given, platform + allowable placements of antennas

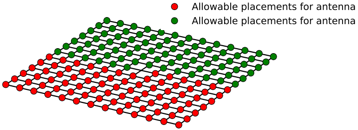




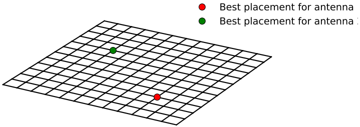
Antenna Placement Problem

Given, platform + allowable placements of antennas





Problem: find best antenna placements



Antenna Placement Problem

Given:

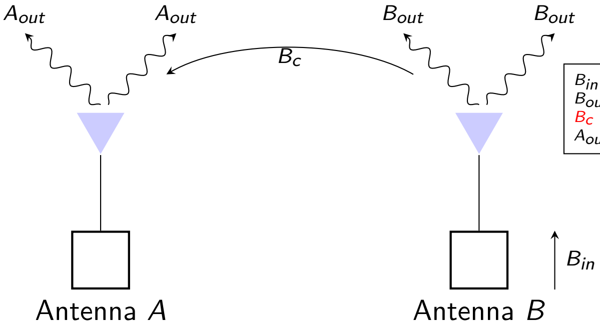
- platform P with its surface gridded such that end points represent possible antenna placements
- set of m antennas $A = A_1, A_2, \ldots, A_m$ such that $m > 1$
- for each A_i, L_i denote the set of allowable placements $\in \mathbb{R}^3$ such that $|L_i| = n_i$ and $\forall i, n_i > 1$;
  $L_i = \{(x_1,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

Problem: Find a set of n optimal antenna placements on P

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.



B_{in} :	Energy from generator
B_{out}:	Energy radiated in space
B_c :	Coupled energy (not good)
A_{out}:	Re-radiated energy

Minimize Mutual Coupling

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

where

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

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

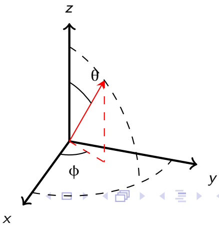
Minimize Difference in Radiation Pattern

Pattern defines the ratio of energy radiated and input energy in a particular direction. For each antenna A_i:

$$F_{RP} = \sum_{i}^{n} \sum_{\theta} \sum_{\phi} (FSG_i(\theta,\phi) - ISG_i(\theta,\phi))^2, \qquad (2)$$

where

- θ, ϕ spherical coordinates
- $FSG(\cdot)$ returns free-space gain pattern
- $ISG(\cdot)$ returns in-situ gain pattern



Objective Function

Find a placement such that F is minimal:

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

where α, β are weights for each of the objectives

Part 2: Stochastic Algorithms

Stochastic Algorithms

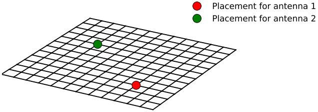
We will consider algorithms which rely on *randomization* principle.

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

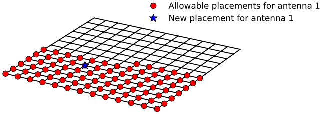
Each algorithm maintains a candidate solution or pool of candidate solutions called population

Stochastic Algorithms: Mutation Operator

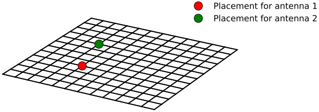
1. Given an individual, select an antenna uniformly at random, let's say antenna 1:



2. For antenna 1, select any other placement:

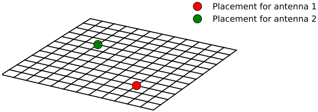


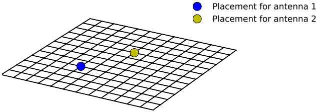
3. Change position for antenna 1 in individual:



Stochastic Algorithms: Crossover Operator

1. Select two individuals from population:

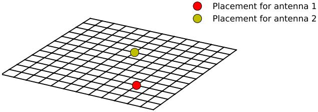


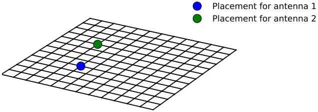


2. Select a crossover point, and swap placements prior to the point:

$$
\begin{array}{ll|l}
\text{ind}_1 : & A_1 & A_2 \\
 & \downarrow \;\; \uparrow & \\
\text{ind}_2 : & A_1 & A_2
\end{array}
$$

3. Two new offsprings created:



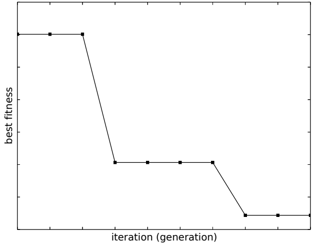


Genetic Algorithm

P ← generate p random individuals. Compute $fitness(ind_i), i \in [1,p]$;
$i = 0$;
while $i < gen_{max}$ **do**
  Elitism: Select n_e fittest individuals to add to **P'** ;
  for $(p - n_e)/2$ *times* **do**
    `/* select returns pair of individuals */`
    $\mathbf{M} \leftarrow select(\mathbf{P}, 2)$;
    if $rand(0,1) < p_c$ **then**
      Apply $crossover(M)$ to get two offsprings **O** ;
      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 $fitness(ind_i); i \in [1,p]$;
  Update $i \leftarrow i + 1$;

Genetic Algorithm

P $\leftarrow$ generate p random individuals. Compute $fitness(ind_i), i \in [1,p]$;
$i = 0$;
while $i < gen_{max}$ **do**
    ;
    Elitism: Select n_e fittest individuals to add to **P'** ;
    for $(p - n_e)/2$ *times* **do**
        /* select returns pair of individuals */
        M $\leftarrow select(\mathbf{P}, 2)$;
        if $rand(0,1) < p_c$ **then**
            Apply $crossover(M)$ to get two offsprings **O** ;
            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 $fitness(ind_i); i \in [1,p]$;
    Update $i \leftarrow i + 1$;



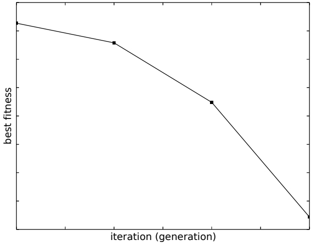
Plateaus suggesting stagnation of search

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, and add all offsprings to **P**;
    Compute $fitness(ind_i), i \in [1, \lambda]$;
    Keep μ best individuals in **P**, and discard remaining $\lambda - \mu$ individuals ;
    Update $i \leftarrow i + 1$

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, and add all offsprings to **P** ;
    Compute $fitness(ind_i), i \in [1, \lambda]$;
    Keep μ best individuals in **P**, and discard remaining $\lambda - \mu$ individuals ;
    Update $i \leftarrow i + 1$



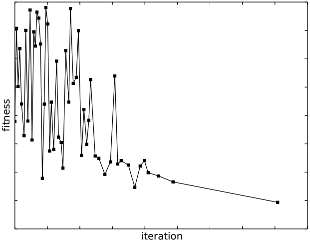
No stagnation in exploration of search space

Simulated Annealing

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

Simulated Annealing

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



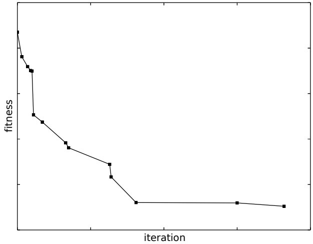
Fluctuation in fitness gradually reduces due to cooling

Hill Climbing

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

Hill Climbing

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



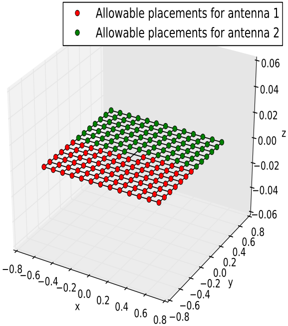
Only accepts fitter individual

Part 3: Results of our experiments

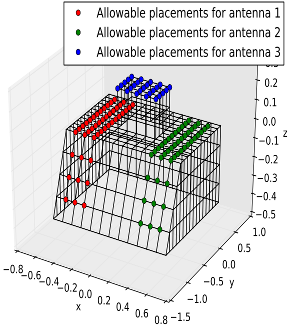
Experimental Setup

1. All test cases describe platforms which are replicas of real-world use cases like mobile devices, tanks, and cars
2. Run all individuals through *NEC* 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

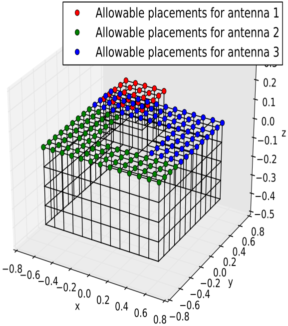


Test Case #1 with 7056(84x84) allowable placements

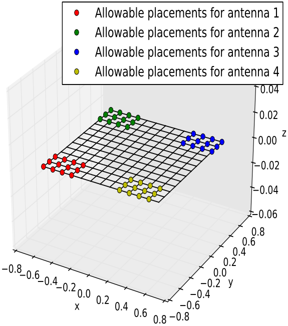


Test Case #2 with 50625(45x45x45) allowable placements

Experiments: Test Cases

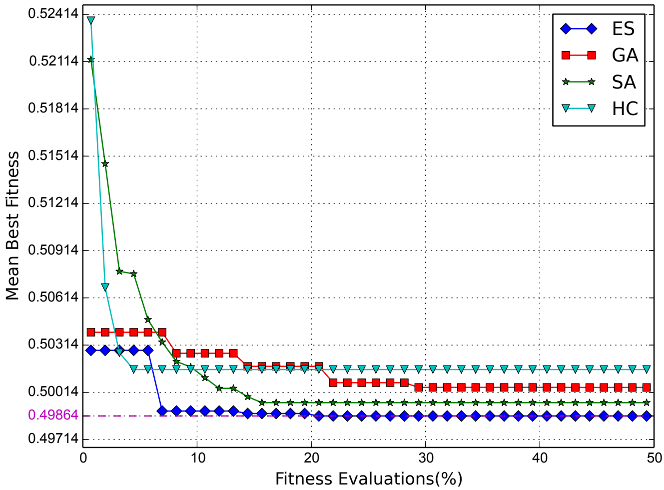


Test Case #3 with 126025(71x71x25) allowable placements

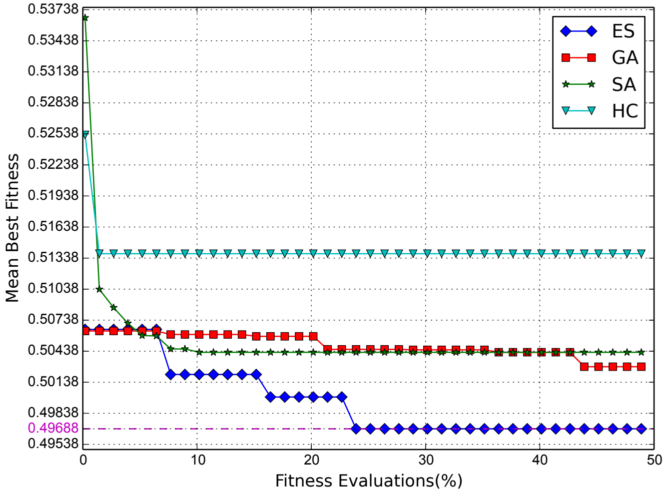


Test Case #4 with 20736(12x12x12x12) allowable placements

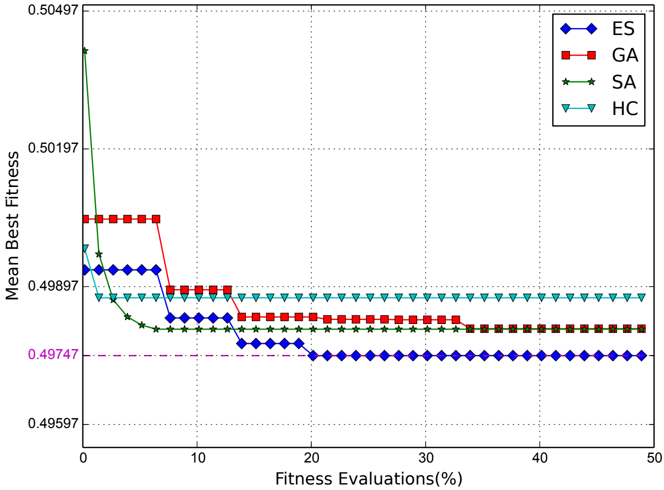
Results - Test Case 1



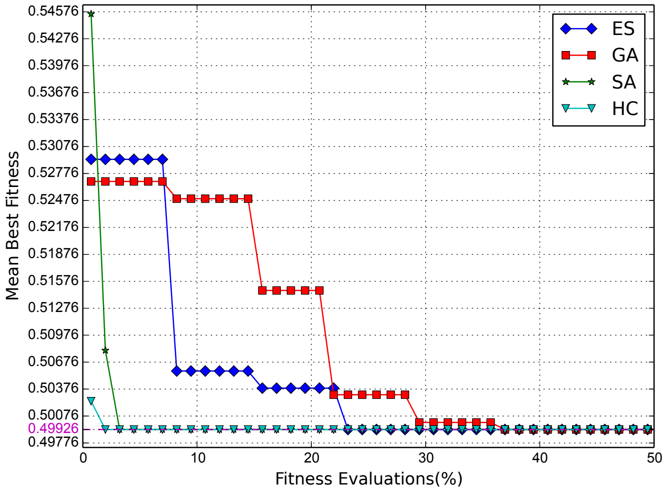
Results - Test Case 2



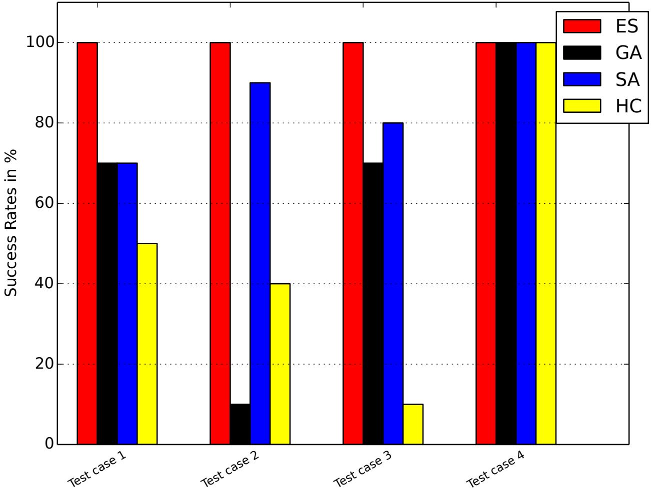
Results - Test Case 3



Results - Test Case 4



Success Rates



Conclusion

- Formulation of the antenna placement problem
- Generic problem formulation to accommodate multiple antennas and platforms
- Optimal placements found using stochastic algorithms