A Comparison of Antenna Placement Algorithms

Abhinav Jauhri

April 23, 2015

Backup Slides

Exhaustive Algorithm

Pseudo code:

```
def exhaustive_search::initialize:
    makeConfigurations(new antenna_configuration,0)

def make_configurations(configuration, count):
    if configuration.length == selected_antennas.length:
        population.push_back(configuration)
        return

    for i in range(0,selected_antennas[count].points.size()):
        if not selected_antennas[count].points.at(i) in configuration:
            configuration.push_back(selected_antennas[count].points.at(i))
            make_configurations(configuration,count+1)
            configuration.pop_back();
```

Parameters - GA and ES

Genetic Algorithm

Test Case	Population	Generations	Mutation Prob.	Crossover Prob.	Elitism	Tournament Size
tc1	500	10	0.1	0.6	50	50
tc2	3600	10	0.1	0.6	360	360
tc3	8500	10	0.1	0.6	850	850
tc4	1500	10	0.1	0.6	150	150

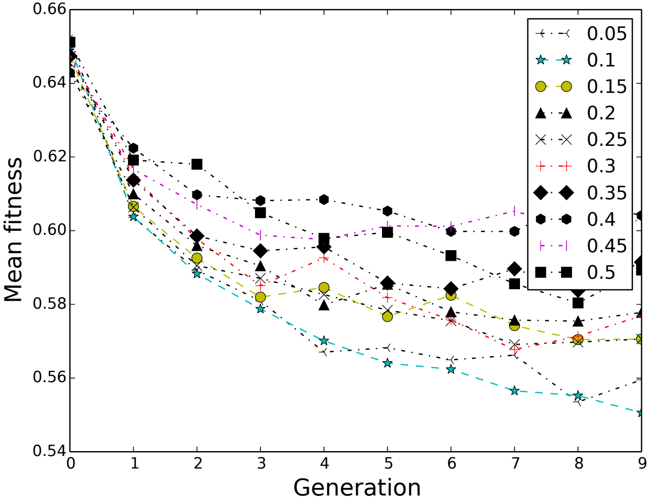
Evolutionary Strategy

Test Case	μ	λ	Generations
tc1	70	490	10
tc2	550	3850	10
tc3	1200	8400	10
tc4	220	1540	10

1/7 ratio[1] between μ and λ. Higher ratios led to higher evaluations per run to reach optimal.

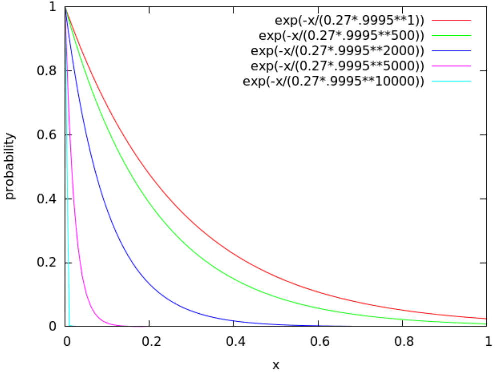
[1] Eiben, A. E., & Smith, J. E. (2003). Introduction to evolutionary computing.

Parameter selection - Mutation Prob. (GA)



Parameters - SA

1. Initial temperature $\in [0.23, 0.27]$
2. Cooling Schedule: Geometric cooling $T_{i+1} = \tau T_i$ $(\alpha < 1)$
   where $\tau \in [0.99, 1)$ such that $T_i <= 10^{-4}$ at 50% iterations



Parameter Selection - Temperature (SA)

Initial temperature selected using technique mentioned by *Dowsland et al.*. It is important to note that acceptance rate drops monotonically with temperature.

Step 1: Set a large initial temperature

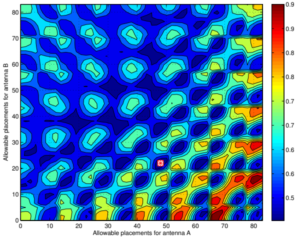
Step 2: Sample some neighbourhood moves

Step 3: If the targeted acceptance ratio is not reached, then modify temperature

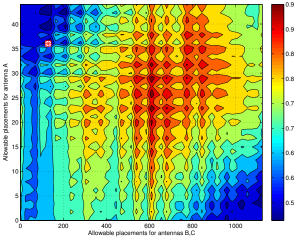
Step 5: Repeat steps 2 and 3 till predefined acceptance ratio is reached

Dowsland, K. A., & Thompson, J. M. (2012). Simulated annealing. In Handbook of Natural Computing (pp. 1623-1655). Springer Berlin Heidelberg.

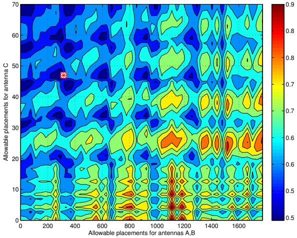
Test Cases - Contour Plots



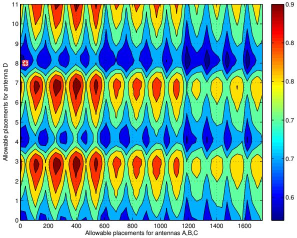
Test Case #1



Test Case #2

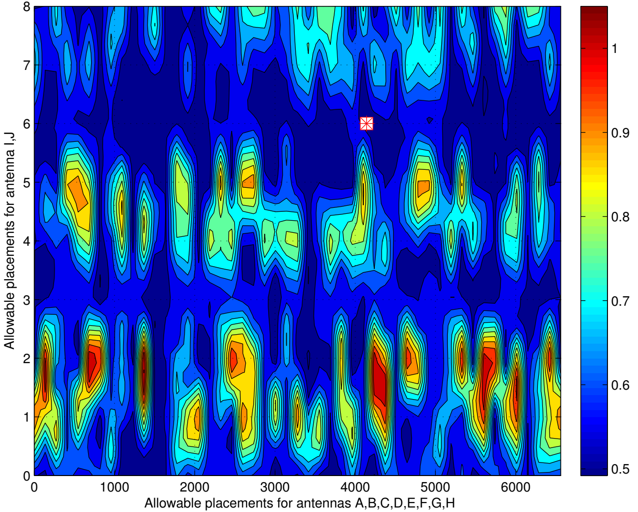


Test Case #3



Test Case #4

Contour plot for 10 antenna problem

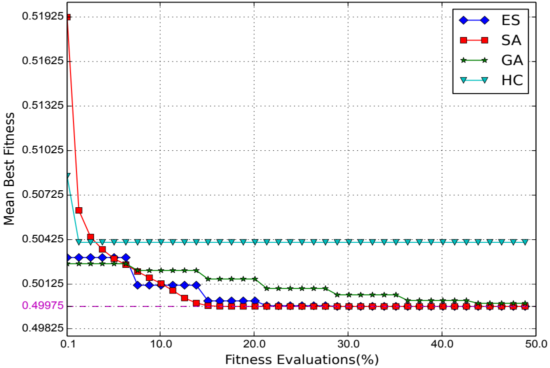


Search space for 10 antenna problem resembles similarity to search space seen in our experiments. Search space size = 59049

Results - Test Case 5

Sample size = 1000

Algorithm	%Evaluations vs. Exhaustive		Best fitness	
	Mean	Std. Dev.	Mean	Std. Dev.
ES	15.11	7.10	0.49975	0.00000
SA	11.58	3.50	0.49975	0.00000
GA	34.08	15.57	0.49977	0.00012
HC	0.13	0.08	0.50407	0.00761



Equivalence of fitness to efficiency

For a particular test case, fitness change of 0.001 is equivalent to either the corresponding value under expected gain ($\mathbb{E}_{\Delta g}$) column, or difference in coupling (Δ_c).

Test Case#	$\mathbb{E}_{\Delta g}$ (dB)	Δ_c (dB)
1	9.34	0.055
2	9.28	0.13
3	9.28	0.15
4	9.33	0.057

Future Work

Experiments on numerous optimization problems[2] have shown Differential Evolution and Particle Swarm Optimization to have convergence with high success rate. Another paper showed advantages of DE over GA[3]

[2] Vesterstrom, Jakob, and Rene Thomsen. "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems." Evolutionary Computation, 2004. CEC2004

[3] Hegerty et al. "A Comparative Study on Differential Evolution and Genetic Algorithms for Some Combinatorial Problems"

Differential Evolution

Step 1: Randomly initialize a population

Step 2: Mutation: For each target $x_i^g, i \in \{1,2,3\ldots,NP\}$, a mutant vector is formed for the subsequent generation using:

$$v_i^g = x_{r_1}^g + F \cdot (x_{r_2}^g - x_{r_3}^g),$$

where $F \in [0,2]$ and r_1, r_2, r_3 are mutually different and also $\neq i$

Step 3: Recombination: Formulate a trial vector as:

$$u_i^{g+1} = \begin{cases} v_{ij}^g, & \text{if rand}() \leq CR \text{ or } j = rnbr(i) \\ x_{ij}^g, & \text{if rand}() > CR \text{ and } j \neq rnbr(i) \end{cases}$$

Step 4: Selection: Compare trial vector u_i^{g+1} and target vector x_i^g, and select the vector which yields a smaller cost function.

Step 5: Termination check

Particle Swarm Optimization

Step 1: Randomly initialize velocity and position of all particles

Step 2: At each iteration, updated velocity as follows:

$$v_i = wv_i + c_1 R_1 (p_{i,best} - p_i) + c_2 R_2 (g_{best} - p_i),$$

where $p_{i,best}, g_{best}$ are positions with best objective value found so far by particle and entire population respectively, c_1, c_2 are weighting factors, $R_1, R_2 \sim \mathbb{U}(0,1)$, w is parameter cooling

Step 3: Position updating: $p_i = p_i + v_i$

Step 4: Memory updating: Update $p_{i,best}$ and g_{best}

Step 5: Termination check