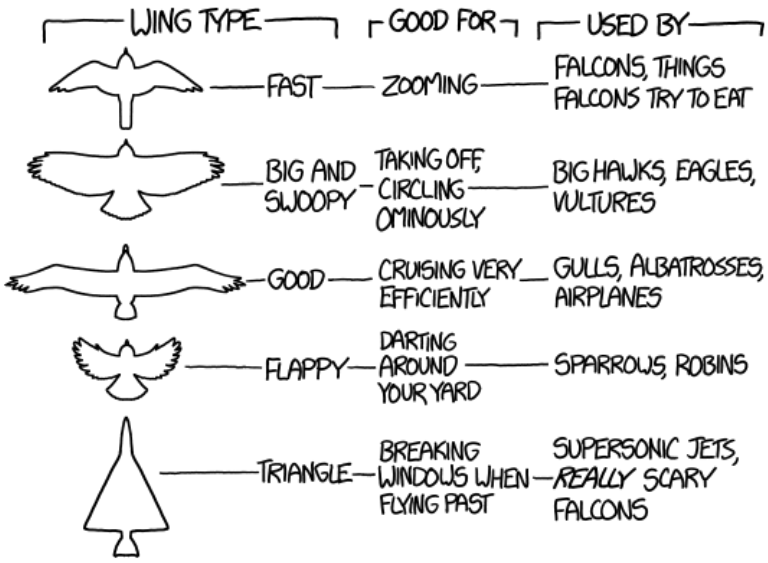
ECTA Homework 3
Shape Matching Problem

Arun Prabhu, arun.prabhu@smail.inf.h-brs.de
Dharmin Bakaraniya, dharmin.bakaraniya@smail.inf.h-brs.de

June 14, 2018



To optimize wings, it is useful to start with known designs and then adjust them to specific tasks. How to convert from one representation to another? In this assignment you will convert the 4-dimensional NACA wing specific representation into a general B-Spline representation with 32 control points. You will compare 2 evolutionary methods, Genetic Algorithms and Evolution Strategies.

1 Assignment Description

Shape Matching Problem

1. Write your own Genetic Algorithm to solve the shape matching problem
2. Write your own version of the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) to solve the shape problem.
3. Compare the performance of the two algorithms on three airfoils (NACA airfoil shapes: 0012, 5522, 9735). Is there a significant difference between a GA and an ES?

- Grading Scheme
  - ☐ Genetic Algorithm (20 pts)
    - ☐ Bitstring *or* Real-Valued (20 pts)
    - ☐ Bitstring *and* Real-Valued (+10pts)
  - ☐ Evolution Strategies (60 pts)
    - ☐ ES with 1/5 rule (30 pts)
    - ☐ CMA-ES with out evolution paths (30 pts)
    - ☐ CMA-ES with evolution paths (+10)
  - ☐ Comparisons (20 pts)
    - ☐ Big beautiful wall of data

2 Submission Instructions

To be perfectly clear we expect two submissions to LEA:

1. 1 PDF (report) – a modified version of your submission PDF, with your own code snippets, figures, and responses inserted
2. 1 ZIP (code and data) – a .zip file containing all code use to run experiments (.m files) *and* resulting data as a .mat file
3. Make sure to follow the naming scheme HW_NUMBER_LASTNAME1_LASTNAME2.suffix
4. → A valid name would be HW_02_Smith_Fernandez.pdf
5. Make sure both team members use the same filename!

3 The Assignment

3.1 Genetic Algorithm

- Genetic Algorithms are typically represented by a string, this string could take many forms, such as bitsrings or real-valued numbers. What are the advantages and disadvantages of each encoding in this application? How would you convert each into 32 real-valued numbers? How could you perform crossover and mutation in each? Which do you think would be best?

  1. Bitstring
     - **Advantages**
       - Crossover and mutation is easy perform.
       - Genome manipulation is faster.
     - **Disadvantages**
       - Difficult to represent solutions of real valued problems.
     - **Crossover**: One point or N point crossover.
     - **Mutation**: A bit is flipped randomly with `mutProb` probability.
  2. Real-valued
     - **Advantages**
       - Easy to represent solutions to real valued problem.
     - **Disadvantages**
       - Slower performance compared to bit string representation.
       - Difficult to perform genome manipulations.
     - **Crossover**: Take mean of the parents.
     - **Mutation**: We replace a gene with a random float with `mutProb` probability.

```
function num = bit2float(bitString, nBit)
[nRows, nCols] = size(bitString);
nFloat = nCols/nBit;
num = zeros(nRows, nFloat);
for r = 1:nRows
    for c = 1:nFloat
        bs = ...
            bitString(r,((c-1)*nBit+1):((c-1)*nBit+1)+nBit-1);
        number = bin2dec(num2str(bs));
        number = (number/(2^nBit)) - 0.5;
        num(r,c) = number;
    end
end
end
```

- Each genome is made of `nFloat` bitstrings.
- Each bitstring contains `nBit` bits.
- We convert it to decimal number. That decimal is divided by the range on the bitstring space(2^{nBit}).
- This float value will be in range of 0 to 1. Now, we subtract 0.5 to get float values in our desired range.

There is no best way to solve GA problems. The representation of genome depends on the problem statement. For example, knapsack will highly benefit by the use of bitstring, but here float representation is advantageous.

- Implement the encoding you think will perform best and plots its median performance over 20 iterations, using 20,000 evaluations. Figure 1 & 2

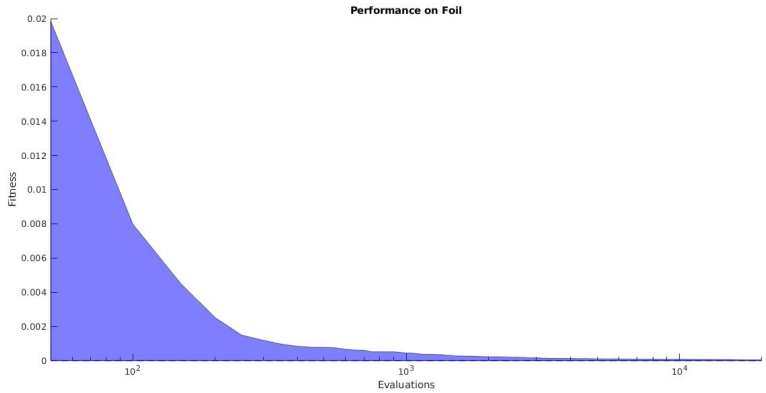


Figure 1: GA 20 experiments single foil (logarithmic X scale)

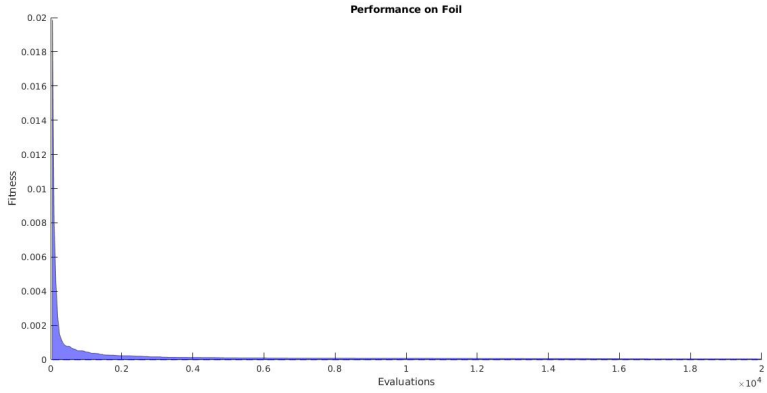


Figure 2: GA 20 experiments single foil (Linear scale)

- ****Extra Credit**** Implement both encodings and compare them on matching task for <u>one</u> of the shapes. Is one significantly better? Can you explain why?
  - As seen from Figure 3 and 4, GA with float genes performed better than GA with bitstring.
  - GA with float as genes converged in less evaluations compared to GA with bitstring.
  - The reason could be that GA with bitstring uses one point crossover which might combine unrelated genes. The conversion from bitstring to float is a slow process which made it slower.

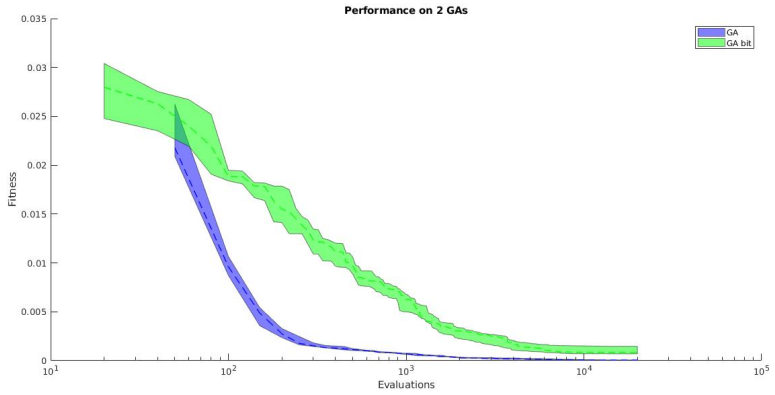


Figure 3: GA vs. GA bitstring (20 experiments all foil (logarithmic X scale))

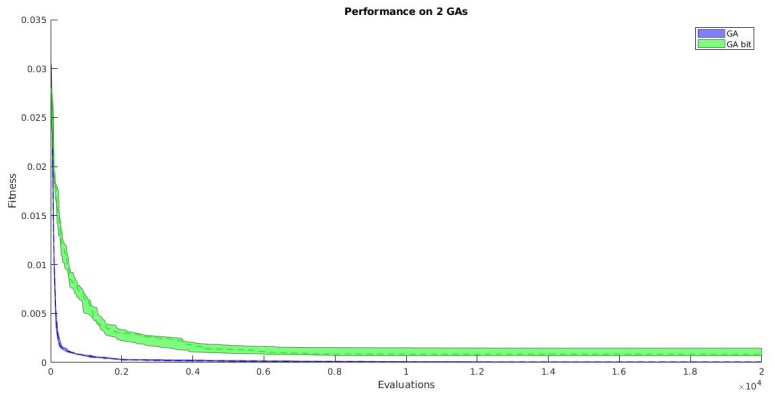


Figure 4: GA vs. GA bitstring (20 experiments all foil (Linear scale))

3.2 Evolution Strategies

- CMA-ES is an advanced version of ES. As a first step implement a simple ES first.
  - In this ES mutation of all parameters should have equal strength which is adjusted by the 1/5th rule.
    (*every N generations change mutation strength: if* $> 1/5th$ *of mutations resulted in an better fitness (i.e. a best solution) increase mutation strength, otherwise decrease mutation strength*). Test your implementation on <u>one</u> of the shapes.
  - Compare to your best GA. Is one significantly better than the other?
    - ES uses (1+1) strategy with $1/5^{th}$ rule so it has 20000 generations while GA only has 400 generations as it has `popSize=50`.
    - As seen in Figure 5, ES is performing significantly better.

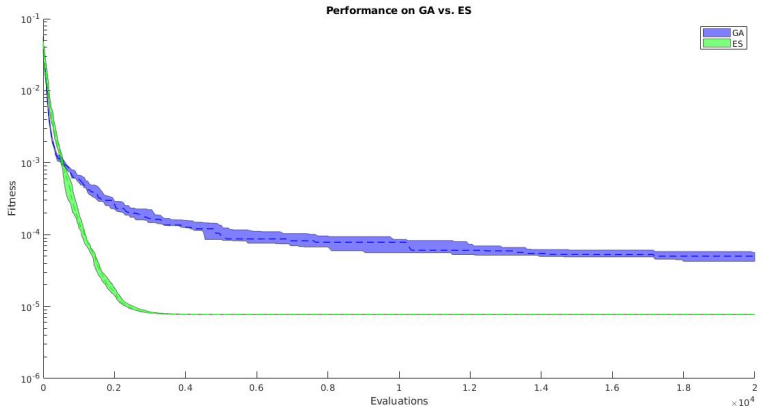


Figure 5: GA vs. ES (20 experiments all foil (logarithmic Y scale))

- Now program CMA-ES without evolution paths.
  - Compare to your ES results. Is a there a significant improvement?
    - As seen in Figure 6 and 7, ES is performing better but it is not performing significantly better. ES converges a little faster in our implementation.
    - ES uses (1+1) strategy with $1/5^{th}$ rule while CMA-ES has $\lambda = 10$ along with $1/5^{th}$ rule.

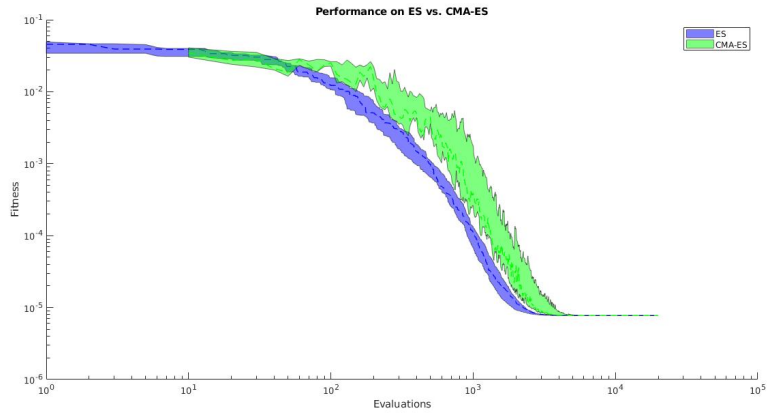


Figure 6: ES vs. CMA-ES (20 experiments all foil (logarithmic X & Y scale))

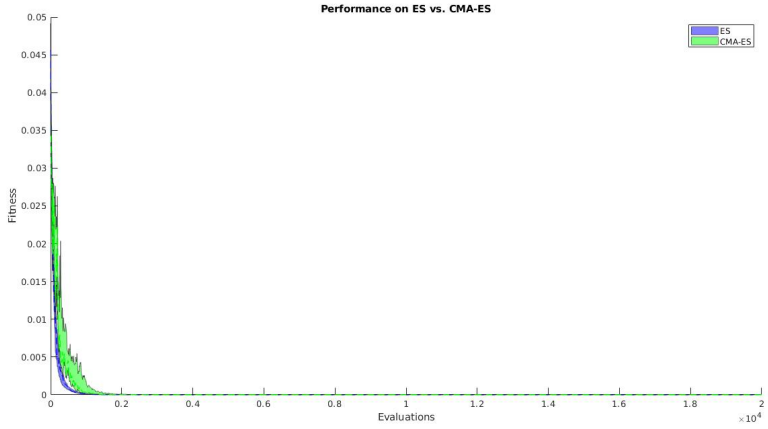


Figure 7: ES vs. CMA-ES (20 experiments all foil (Linear scale))

- ***Extra Credit*** Now program CMA-ES with evolution paths.
  - Compare to your previous CMA-ES results. Is a there a significant improvement?
    - As seen in Figure 8 and 9, CMA-ES is performing better in our implementation.
    - CMA-ES has $\lambda = 10$ along with $1/5^{th}$ rule while CMA-ES EP has evolutionary path for updating covariance matrix and cumulative step length adaptation to update sigma.

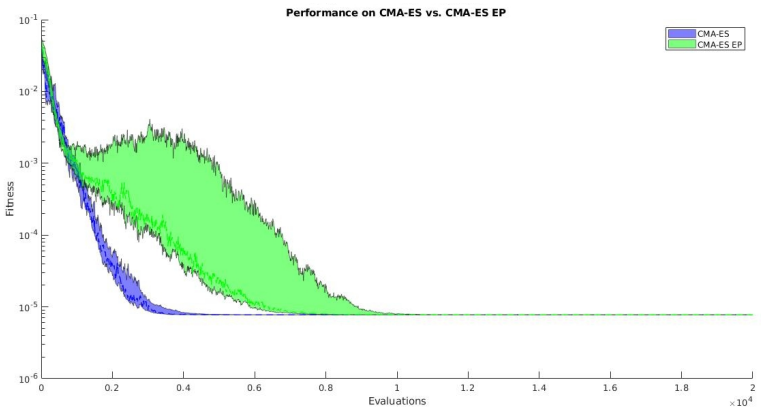


Figure 8: CMA-ES vs. CMA-ES EP (20 experiments all foil (logarithmic Y scale))

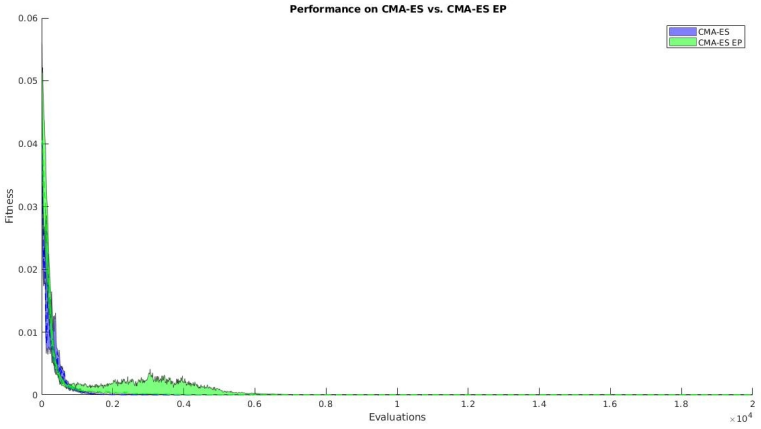


Figure 9: CMA-ES vs. CMA-ES EP (20 experiments all foil (Linear scale))

3.3 Comparisons

Produce one plot which shows the performance of each algorithm. Run each algorthm 20 times on each shape (NACA airfoil shapes: 0012, 5522, 9735), with a budget of 20,000 function evaluations. For each run record the best ever found individual at each evaluation. For each algorithm plot the median fitness of this best ever individual. This may take some time, write a script to run and collect this data. The following should be on this plot (leaving out any algorithms you chose not to implement):

1. Binary GA
2. Real-Valued GA
3. ES
4. CMA-ES (without evolution paths)
5. CMA-ES (with evolution paths)

Figure 10 & 11

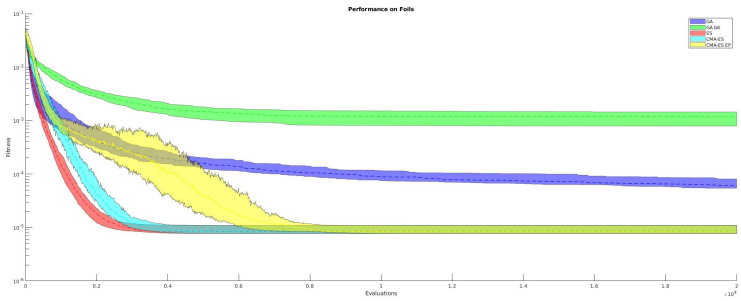


Figure 10: All 5 algorithms (20 experiments all foil (logarithmic Y scale))

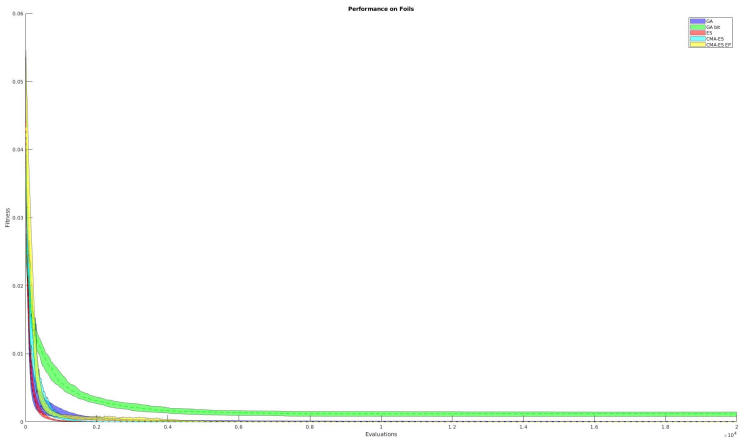


Figure 11: CMA-ES vs. CMA-ES EP (20 experiments all foil (Linear scale))