**Assignment 3**

**By: Malka Abramovitch, Maor Betser**

**The Problem:**

Given quadratic family functions with mixed variables, we had to optimize the integer part of the solution matrix in the range of [−100,100][-100,100][−100,100], where the first 32 numbers are real numbers and the latter 32 numbers are integers.

**Path to Solution:**

At first, we tried implementing a Genetic Algorithm (GA) to optimize the integer part of the solution after running 1+1 ES on the real part. The idea was that since the real part was already optimized, we could focus entirely on refining the integer values.  
However, we ran into runtime issues—running **10⁶ iterations with 30 NRUNS** was costly. Our results were **~10 on HCigar and RotatedHEllipse**, while for **Hadamard**, we reached **~500**.

At that point, we felt stuck and started doubting our GA approach. Instead of forcing it, we decided to talk to classmates, compare ideas, and look for inspiration on a better way to tackle the problem. That brainstorming session helped us rethink our strategy and eventually led to our second approach.

For our second attempt, we wanted something simple, fast, and effective. Instead of overcomplicating things, we went back to the 1+1 ES from class to handle the real part of the vector. Once that was optimized, we focused on the integer part using mutation.

At first, we kept the mutation basic, randomly picking a step from [−1,0,1] to allow small jumps. Later, we improved it by switching to a **Laplace discrete mutation**, where:

* Small jumps happened more often.
* Larger jumps were rare but possible, helping the search escape local minima ("dips").

(This has reminded us a lot of the gaussian distribution but for discrete values)+-

We have also implemented a check that counts the number of iterations we are “stuck”  
and if it surpasses a certain threshold we roll a new candidate with a larger jump lamda,  
this helps us explore better. With all this implemented our results weren’t great yet

But for lack of time we had to submit, then we saw there was an extension in the deadline and that prompted us to get back to work.

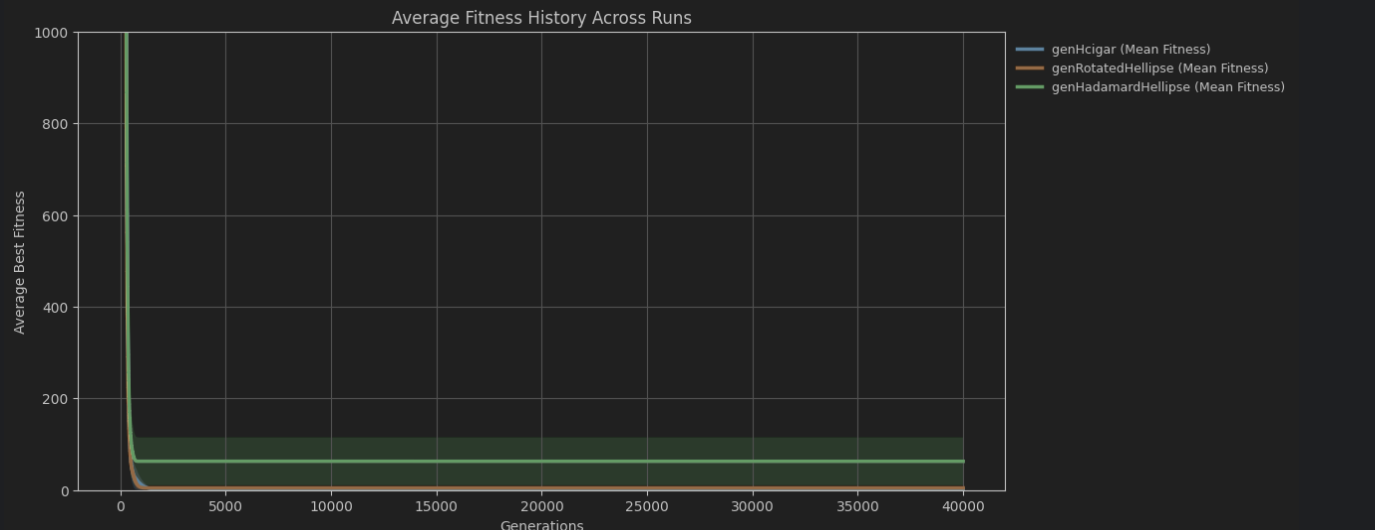
For our third and last attmept we chose to work on the 1+1ES and introduced spererate sigma for each coordinate allows for more control in the stepsize.

The success rate of each coordinate is calculated seperatly and if the success rate is larger than the target success rate we increase sigma otherwise we decrease it, this allows us to be more flexible and is better suited for a high dimensional problem like this but costing us more computational time.  
with that change we got great results reaching the minima for all three functions ☺

**Final Thoughts**

This assignment was a rollercoaster. We started with high expectations for GA but got humbled by runtime issues and ultimately found a faster, simpler approach through mutation. This path was also not easy not yielding results very fast but when we revised the 1+1ES, we saw our greatest improvements, together with the laplace distribution it is safe to say we are happy with our current results ☺   
  
**We Tested Different Distributions**

Our latest results with 10^6 iterations



**One of our better results**

genHadamardHellipse | Run: 1:

Best Combined Fitness: 4.275263850194872   
Best solution =   
[ 8.25593027 -8.72113803 6.14334257 -8.8354361 7.0275079 -8.8796796  
6.17061135 -9.63242912 7.80197315 -8.13407755 6.18133646 -8.22923161  
7.02131438 -8.82997032 6.74595644 -9.47398822 7.85191871 -8.72599718  
6.13317069 -8.83162378 7.00701431 -8.87815447 6.17058258 -9.59193762  
7.76148165 -8.13404875 6.1798113 -8.20873803 7.01750205 -8.81979844  
6.75081556 -9.0699767 8. -9. 6. -9.  
7. -9. 6. -9. 8. -8.  
6. -8. 7. -9. 7. -9.  
8. -9. 6. -9. 7. -9.  
6. -9. 8. -8. 6. -8.  
7. -9. 7. -9.      ]

**Previous results that led us to revisit our code and improve.**

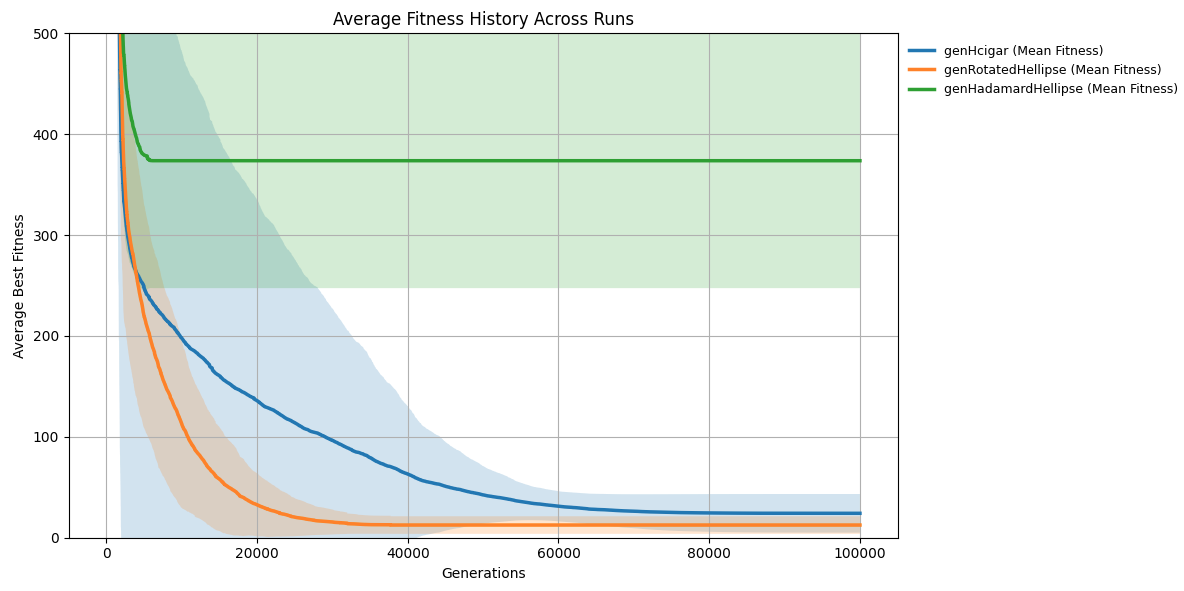
**Outputs taken from ipynb files:**

**030.ipynb  
genHcigar** | Run 10: Best Combined Fitness = 9.164359451072592e-10 | Best solution = [ 7.00029887 -6.99999996 6.9999993 -7.00000022 7.00000052 -7.00000018  
 7.00000049 -7.00000136 7.0000004 -7.00000042 6.99999877 -7.00000145  
 6.9999995 -6.9999988 7.00000115 -6.99999975 6.99999846 -6.99999821  
 6.99999862 -6.99999907 6.99999979 -7.00000048 6.99999943 -6.99999943  
 6.99999918 -7.00000016 7.00000051 -6.99999936 7.00000095 -6.99999968  
 6.99999882 -7.00000042 7. -7. 7. -7.  
 7. -7. 7. -7. 7. -7.  
 7. -7. 7. -7. 7. -7.  
 7. -7. 7. -7. 7. -7.  
 7. -7. 7. -7. 7. -7.  
 7. -7. 7. -7. ]

**030.ipynb  
genRotatedHellipse** | Run 29: Best Combined Fitness = 9.917684940356375e-07 | Best solution = [ 6.99234341 -6.99969282 7.00141066 -7.00020068 7.00062212 -7.00010197  
 7.00001591 -6.99999754 7.00027108 -7.00030247 7.00009227 -7.00020855  
 7.00020467 -7.00015642 7.00036259 -7.00017031 7.00029702 -6.999943  
 7.0002107 -7.00013039 7.00036073 -7.00005952 7.00002376 -7.00008649  
 7.00025615 -7.00017909 6.99996681 -7.0001505 7.00010171 -7.00020887  
 6.99998808 -7.00002225 7. -7. 7. -7.  
 7. -7. 7. -7. 7. -7.  
 7. -7. 7. -7. 7. -7.  
 7. -7. 7. -7. 7. -7.  
 7. -7. 7. -7. 7. -7.  
 7. -7. 7. -7. ]

**031.ipynb  
genHadamardHellipse** | Run 13: Best Combined Fitness = 35.96924336817989 | Best solution = [   
-0.57723131 -6.32582277 -0.33073493 -8.33854945 0.39354542  
 -8.31936819 2.7482502 -10.14164616 0.07752319 -7.34328953  
 -0.17975415 -8.56656369 0.33327156 -9.57054144 3.20034984  
 -11.44033878 -1.4510248 -7.14941702 -0.76426736 -8.43442183  
 0.30317312 -8.3776269 2.67519636 -10.63467302 -0.43368136  
 -7.4449882 -0.26289842 -9.02195256 -0.11108997 -9.60876257  
 2.768308 -12.31578248 0. -7. -1.  
 -8. 0. -8. 2. -10.  
 1. -7. 0. -9. -1.  
 -10. 3. -12. -1. -7.  
 0. -9. 0. -9. 2.  
 -10. 0. -8. 0. -10.  
 0. -10. 3. -12. ]

**First run with Uniform Distribution (100,000 iterations)**



**Run with Gaussian Distribution (50,000 iterations)** Testing the gaussian distribution and rounding down.

