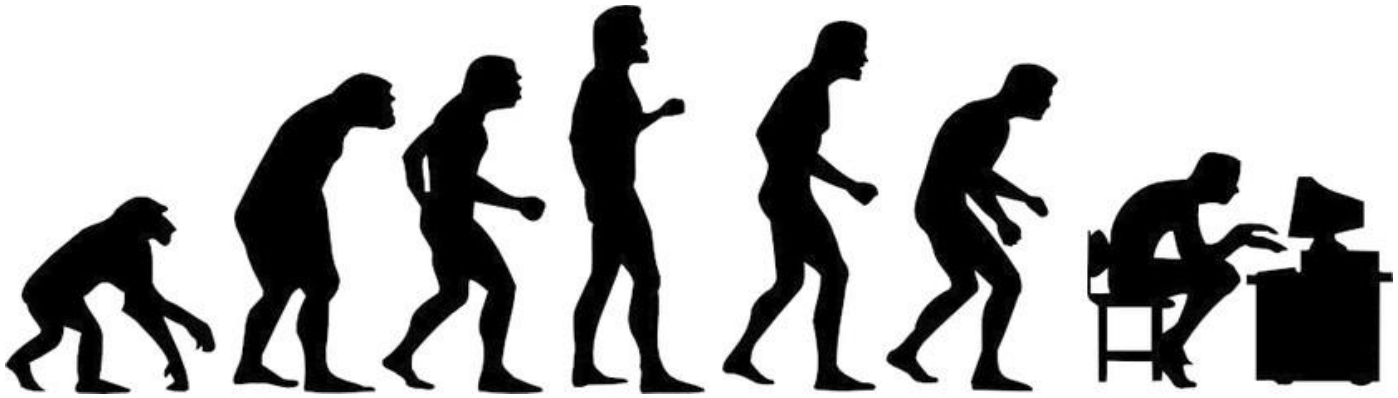
Genetic Algorithms applications to optimisation problems in Physics

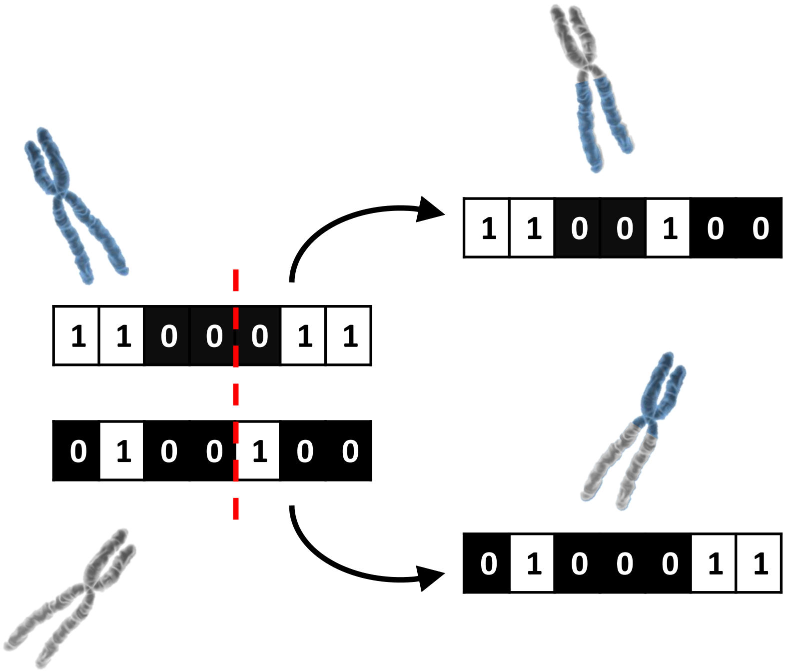
mirror coatings in advanced interferometric detectors of gravitational waves

FRANCESCO DI RENZO

FIRST YEAR SEMINAR
DOCTORAL SCHOOL OF PHYSICS - UNIVERSITY OF PISA
21 SEPTEMBER 2017

Genetic Algorithms





Evolutionism in programming

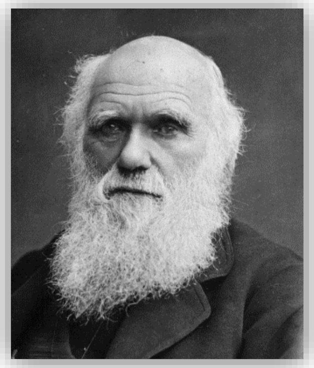
- **Genetic Algorithms** (Holland, 1975) are **adaptive heuristic search algorithms** that were invented to solve optimisation problems

- They attempt at making an **intelligent exploitation of a random search** mimicking some processes observed in Nature: natural selection, "survival of the fittest" (Darwin, 1837)

- Although randomised, they are **not random**: they exploit historical information to direct search into the region(s) of better performance(s) within the search space

- They are credited as among the most **robust and reliable** *non-derivative* global-optimisation tools for problems of moderate size (few tens of unknown)

John H. Holland, Adaptation in Natural and Artificial Systems (1975)



Charles R. Darwin, Tree of Life (1837)

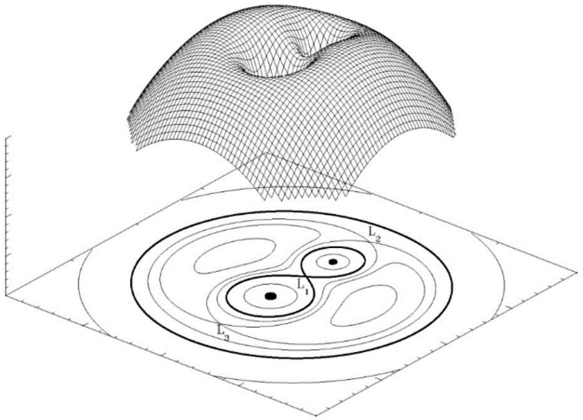
Optimisation problems

Simple case:

given a function

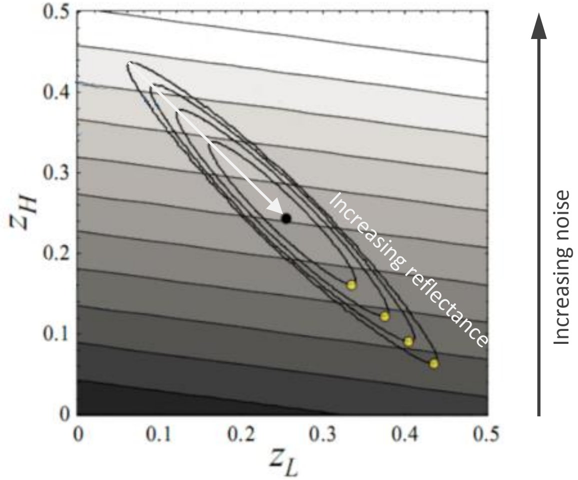
$$f(x_1, x_2, x_3, \dots)$$

find the set of variables (optimal solution) x_i, with $i = 1, 2, 3, \dots$, for which f takes the maximum value.



Real-world complications:

- ✓ Multi-objective optimization (conflicting criteria)
- ✓ Multiple constraints
- ✓ Non-differentiable functions
- ✓ Combination of continuous and discrete variables
- ✓ …



Other common optimisation methods

Calculus approach: find x_i such that $\nabla f(x_1, x_2, x_3, \dots) = \mathbf{0}$

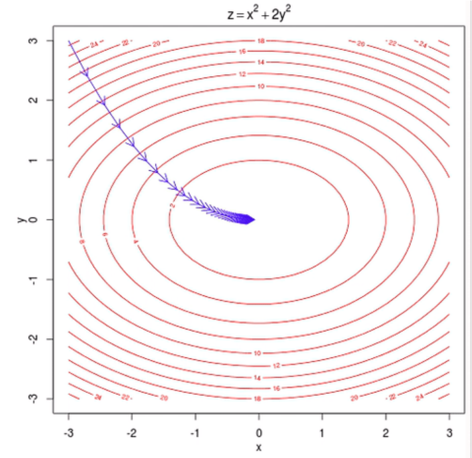
Random search: points are randomly selected and evaluated

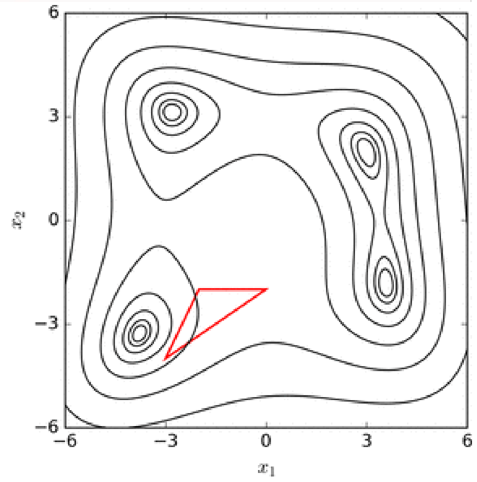
Gradient based methods

- **Classical "hill-climbing" method:** starting at a random location, moving in the direction of steepest ascent
- **Iterative hill-climbing:** the procedure is reiterated at different starting points
- **Simulated annealing:** up- and down-hill moves are weighted

Non-gradient search

- **Nelder-Mead's method:** movements of a "simplex" in the search space





Example: Fraunhofer single slit diffraction

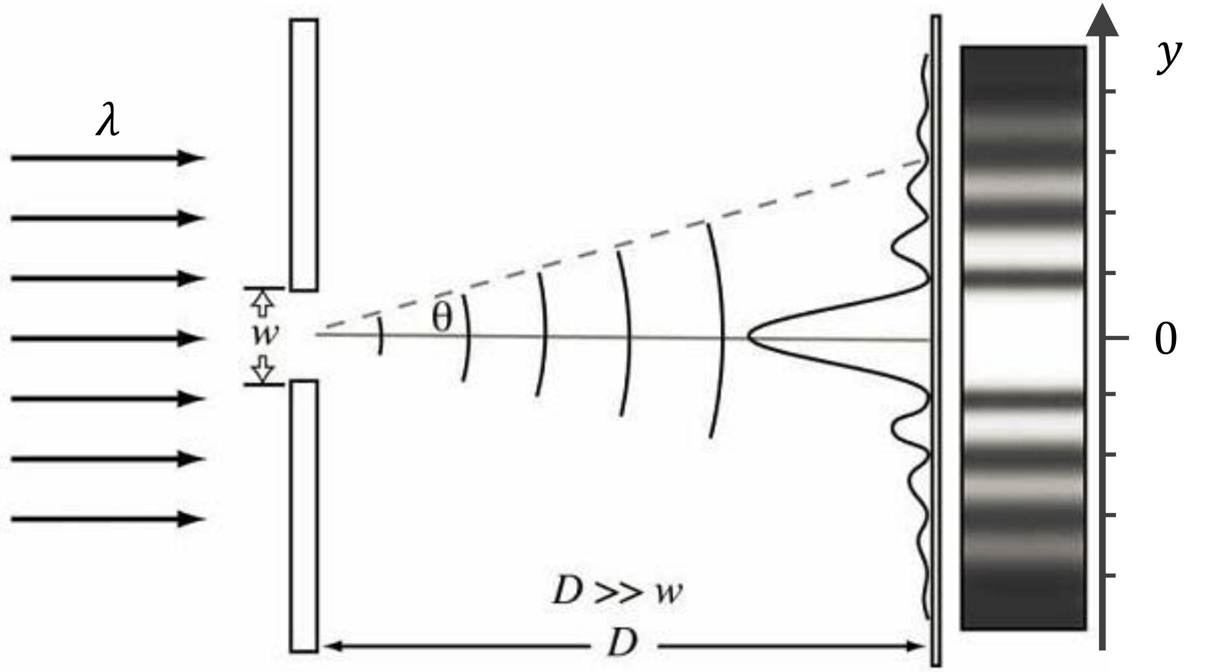
Problem: find the maxima of the intensity of the diffracted light from a single vertical slit:

$$I(x) = I_0 \left(\frac{\sin x}{x}\right)^2$$

where $x = \frac{\pi w\, y}{\lambda\, D}$.

Minima: $x_k = k\pi$, with $k = \pm 1, \pm 2, \dots$.

$\boldsymbol{f'(x)} = 2 \sin x \left(\frac{\cos x}{x^2} - \frac{\sin x}{x^3}\right)$



Problem definition: search space, individuals and genes

Each **candidate solution** to the problem, drawn from a suitable **search space**, is encoded in an **individual** (composed of one or more chromosomes): in our case, “any” $x \in \mathbb{R}$.

Conversion: in order to better model the genetic process of evolution, real numbers are usually converted into arrays of binaries.
Example:

integer and fractional part

$$\pi \simeq 3.141\ldots \rightarrow \underbrace{[\overbrace{\boxed{0},0,1,1}\,,\overbrace{0,0,1,0,0,1,0,0,0,0}]}_{\textbf{chromosome/individual}}$$

Gene: possible values (alleles) 0 or 1, encodes the genetic features of the individual.

```
# Genetic Algorithm solution to the
# single slit diffraction problem

>>> from random import randint,random
>>> from numpy import sin,arange,array


# Definition of an individual
# as an array of integers
>>> def individual(length, min, max):
        return [ randint(min,max) for \
        x in xrange(length)]


# Genes representation
>>> x = individual(12,0,1)

# Get the real value of the individual
>>> def xvalue(individual):
        return round(sum(individual[i]\
        *2**(3-i) for i in \
    xrange(len(individual))),3)
```

Creation of the initial population

A number (**size**, chosen by the use) of individuals is drawn from the search space:

- This is (usually) randomly generated, in order to guarantee **no initial bias** in the search space
- In some circumstances, **a priori knowledge** can be implemented producing an initial bias towards what we expect to be the **true solution**.

In our case the objective function is **even** and the candidate solutions are expected to be **close to zero** (in front of the slit).

```
# Creation of a collection of
# individuals, population

>>> def population(size,length,min,\
    max):
        return[individual(length,min,\
        max) for x in xrange(size)]


# Example: population of 5 individuals
# constituted by 12 random binary
# genes each

>>> Pop = population(5,12,0,1)
```

Fitness function

We need a way to judge **how effective** each candidate solution is, *i.e.* the **fitness** of each individual.

- The **fitness function** maps the chromosome representation into a scalar value: (usually) the higher, the better
- It has to contain ALL the objectives that need to be optimised

In our optimisation problem, the (non-negative) function that has to be maximised can be simply used as the fitness function as well

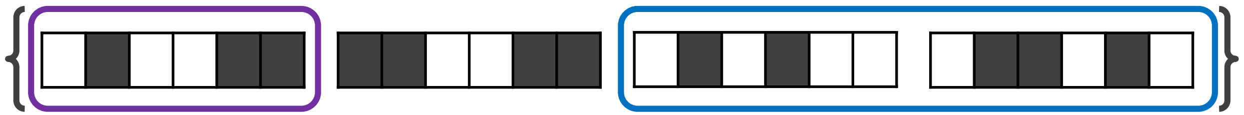
```
# Evaluate the objective function

>>> def fvalue(individual):
        xv = xvalue(individual)
        if xv == 0:
            return 1
        else:
            return (sin(xv)/xv)**2

# fitness function = objective f.
>>> def fitness(individual)
        return fvalue(individual)


# Define the average pop fitness
>>> def afitness(pop):
        return sum(fitness(\
        population[i]) for i in \
        xrange(len(population)) \
        /len(population)
```

Evolution of the population (1)

We want to improve our population in an iterative (evolutionary) process:

1. **Natural selection:** for each **generation** (iteration) we select a fraction of the "best performing" individuals, as judged by their fitness, to be the parents for the next one.

⇒ We also randomly add some **other individuals** to promote genetic diversity: this decrease the risk of getting stuck at a local maximum.



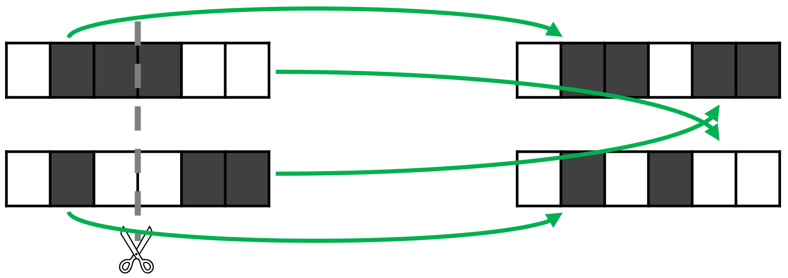
```
# Evolution of the popultion through
# generations

>>> def evolve(pop,retained_frac=.2,\
    random_prob=0.05, mutation_prob= \
    0.02):

    selected = [ (fitness(x),x) for x\
    in pop]
    selected = [ x[1] for x in \
    sorted(selected)]

    retained = int(len(selected)\
    *retained_frac)
    parents = selected[retained:]

    for individual in \
    selected[:retained]:
        if random_prob > random():
            parents.append(individual)
```

Evolution of the population (2)

We want to improve our population in an iterative (evolutionary) process:

2. Crossover: we breed together parents to give birth to children until the desired population size is restored.

It is ok to have one parent breed multiple times, but one parent should never be both the father and the mother of a child at the same time.



```
parents_number = len(parents)
desired_children = len(pop) - \
parents_number

children = []
while len(children) < \
desired_children:
  dad = randint(0,parents_number-1)
  mom = randint(0,parents_number-1)
  if dad != mom:
     dad = parents[dad]
     mom = parents[mom]
     half = len(dad)/2
     son = dad[:half] + mom[half:]
     children.append(son)

 parents.extend(children)
```

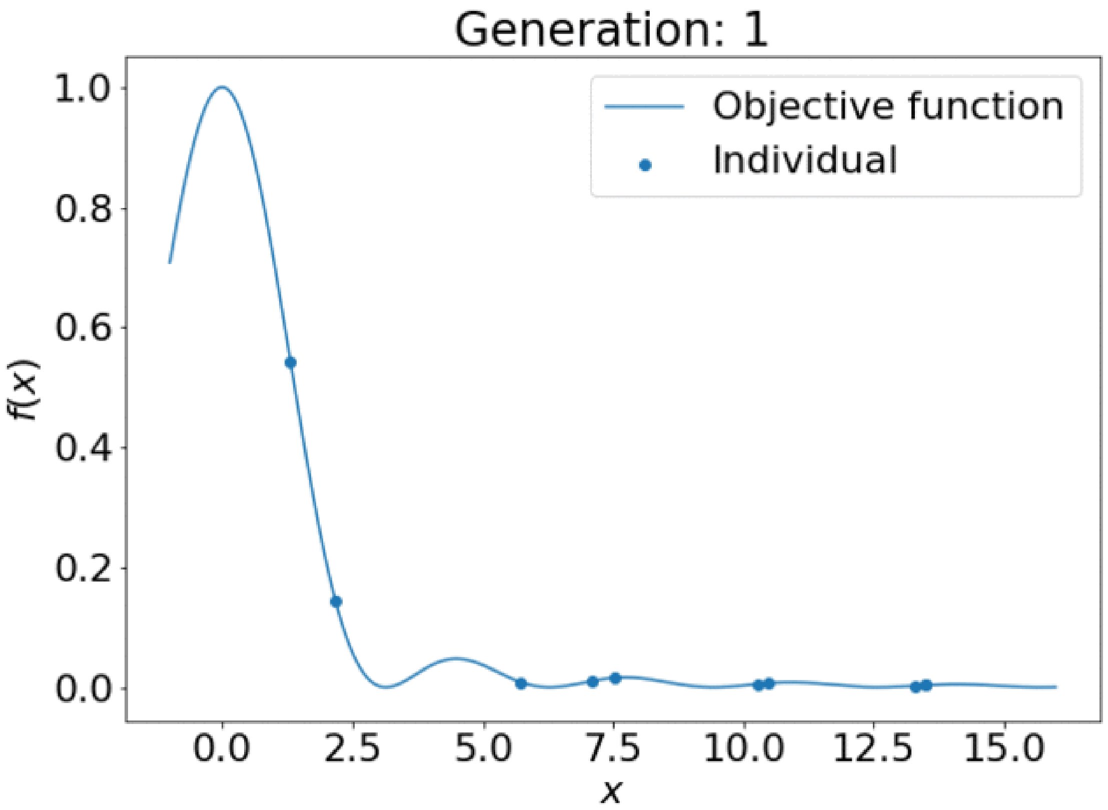
Evolution of the population (3)

We want to improve our population in an iterative (evolutionary) process:

3. Mutation: we randomly change the values of the genes in the individuals’ chromosomes, introducing new genetic material.

```
for individual in parents:
    if mutation_prob > random():
        gene_to_mutate = randint(0,\
        len(individual)-1)

        individual[gene_to_mutate]=\
        randint(min(individual),\
        max(individual))


return parents
```

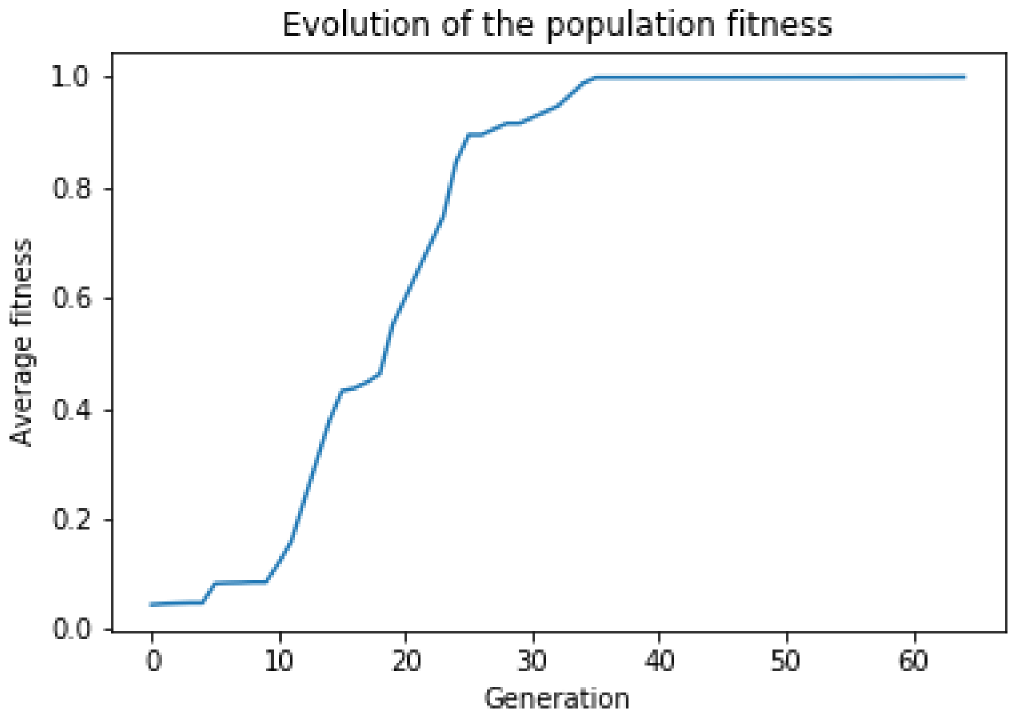
Test: generation of the individuals



Generations: **40**, Individuals in the population: **10**, Genes: **12**

```
>>> Pop = population(10,12,0,1)

>>> popval = []

>>> for i in xrange(len(Pop)):
        popval.append([xvalue(Pop[i]),\
        fvalue(Pop[i])])

>>> xval,yval = array(popval).T
>>> x = arange(-1,16,.01)

>>> plt.plot(x,(sin(x)/x)**2)
>>> plt.scatter(xval,yval)

>>> plt.xlabel('$x$')
>>> plt.ylabel('$f(x)$')
>>> plt.legend(['Objective function',\
    'Individual'])

>>> plt.show()
```

Performance: average fitness of the population

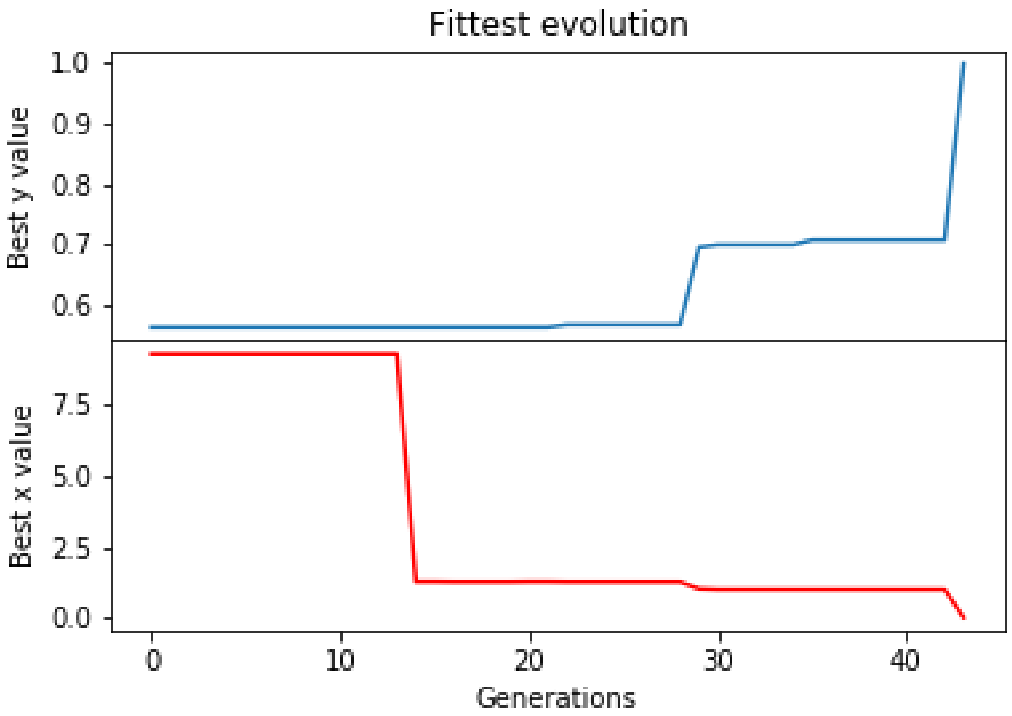


Individuals in the population: **10**, Genes: **12**.

Average fitness tends to **increase** with generations.

```
>>> Pop = population(10,12,0,1)

>>> fitness_history = [afitness(Pop),]

>>> while afitness(Pop) < 0.999:
        Pop = evolve(Pop)
        fitness_history.append(\
        afitness(Pop))

>>> for datum in fitness_history:
        print datum


>>> plt.plot(fitness_history)
>>> plt.title('Evolution of the \
    population fitness')
>>> plt.xlabel('Generation')
>>> plt.ylabel('Average fitness')
>>> plt.savefig('average.png')
>>> plt.show()
```

Performance: evolution of the fittest individual in the population



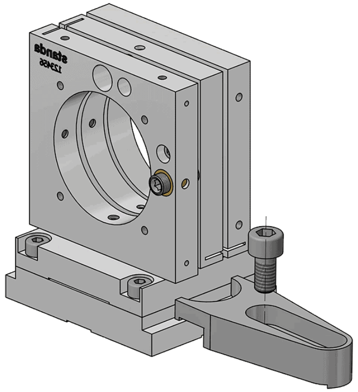
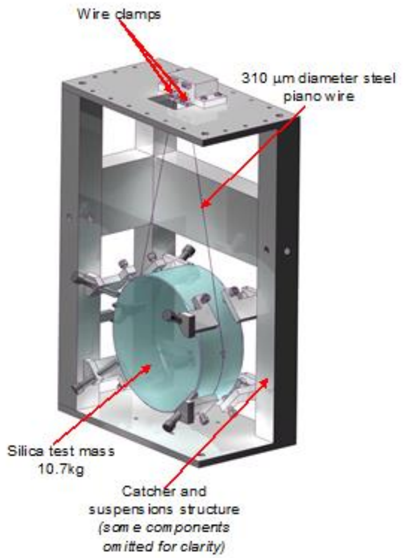
Individuals in the population: **10**, Genes: **12**.

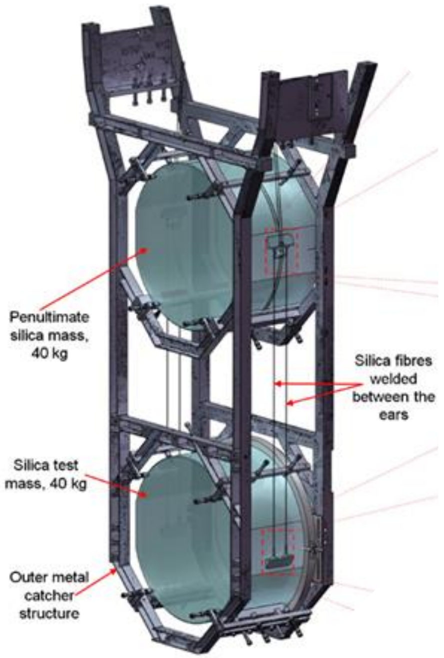
Jumps are due to **mutation** of the most significant genes.

```
>>> def find_best(pop):
        selected=[[fitness(x),x] for x\
        in pop]
        selected = sorted(selected)
        best = selected[-1]
        return best

>>> best_history = [find_best(Pop)]

>>> while find_best(Pop)[0] < 0.999:
        Pop = evolve(Pop)
        best_history.append(\
        find_best(Pop))

>>> best_array = []
>>> for datum in best_history:
        best_array.append(\
        [xvalue(datum[1]),datum[0]])
```

Mirrors optical coatings in advanced gravitational wave detectors



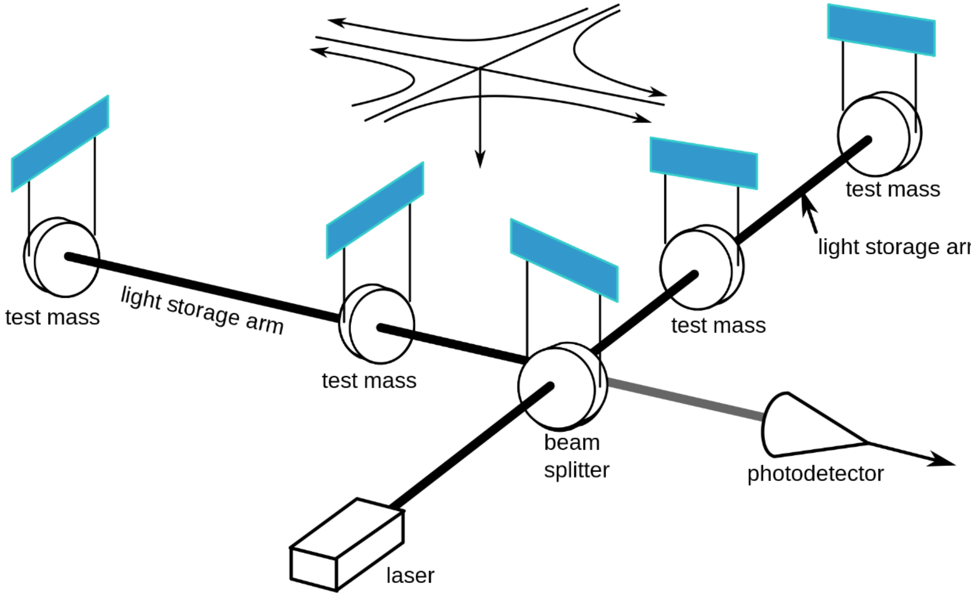





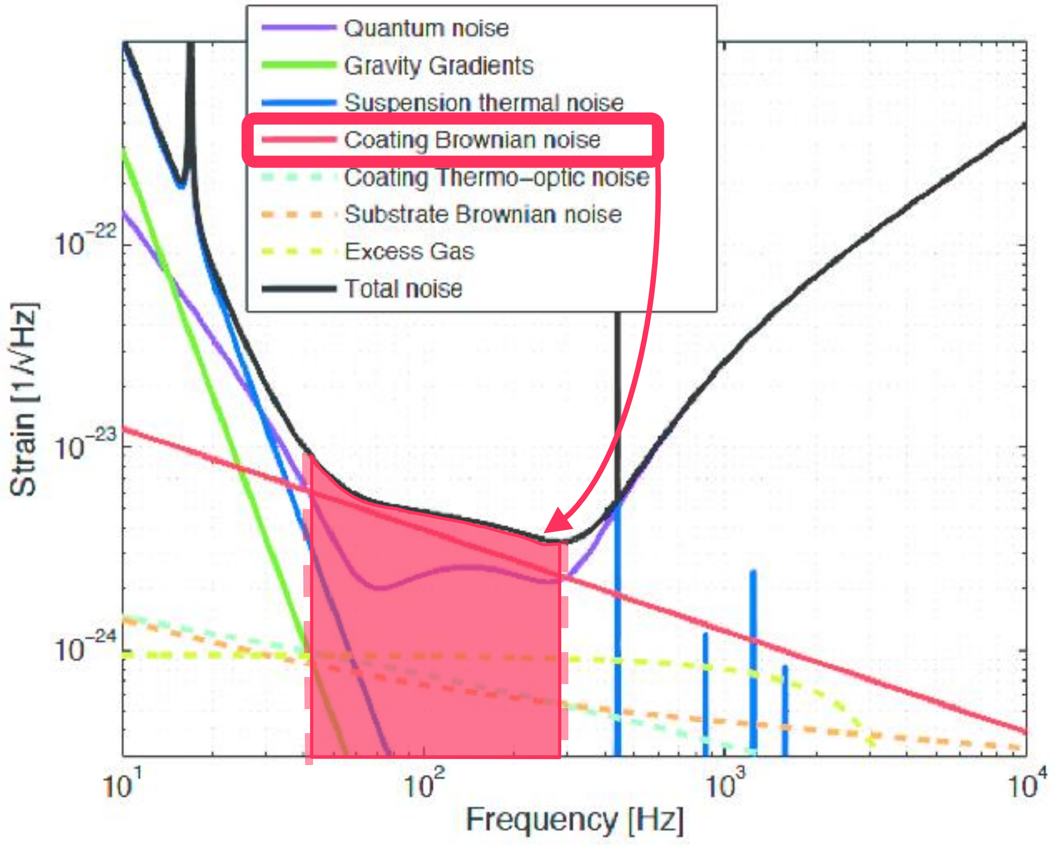
(Pictures not to scale)

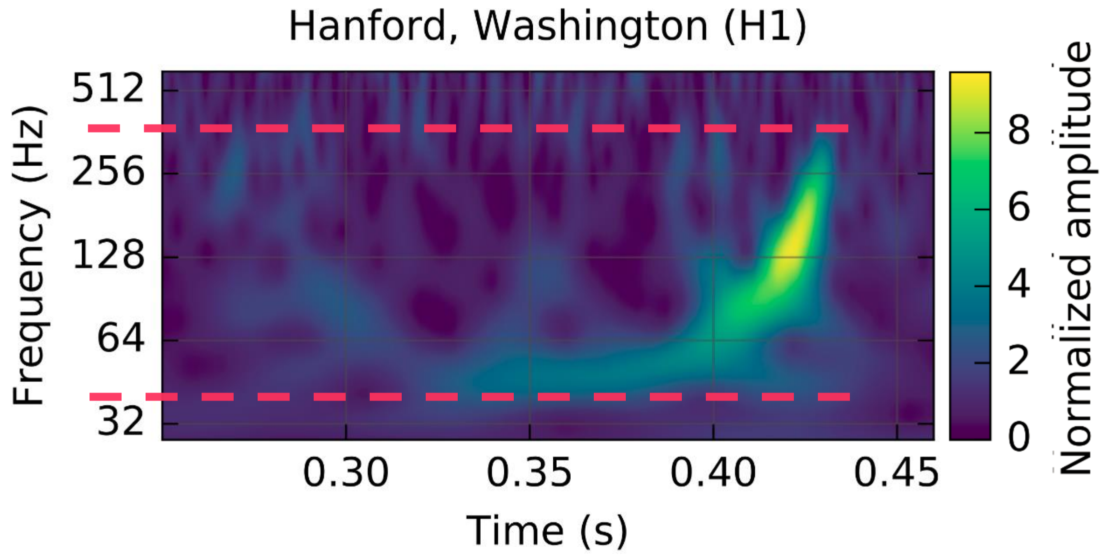
Interferometric gravitational wave detectors

- Modern Gravitational Waves (GWs) detectors are modified Michelson interferometers
- GWs act as quadrupolar tidal forces on the suspended mirrors (test masses)
- The **goal** is to **measure these forces** through the difference of phase of the recombined light at the output of the two interferometer arms
- Many **noise sources** can mimic this effect and limit the detector sensitivity



Detector sensitivity to gravitational waves





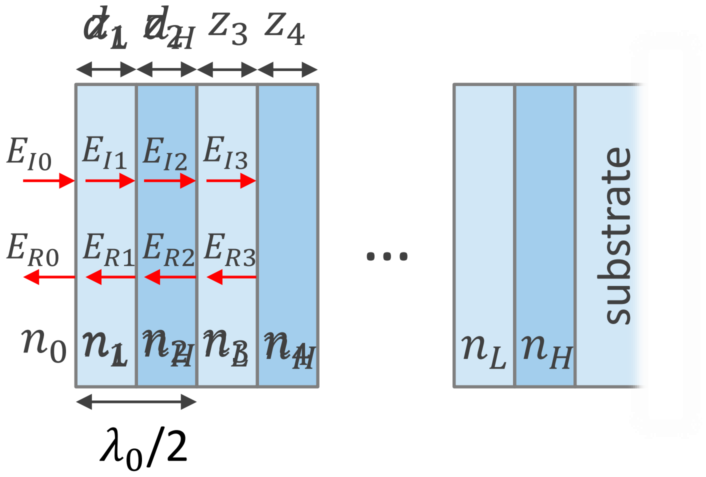
Left: detector strain sensitivity (black line), together with the corresponding noise budget (coloured lines).
Top: spectrogram of GW150914 at LIGO Hanford.

Mirrors structure in Virgo and LIGO detectors

Mirrors are obtained by means of N_d stacked doublets of low- and high-refraction index materials. *Typically*:

High: Ta_2O_5, tantalum pentoxide (**tantala**), $n_H \gtrsim 2$
Low: SiO_2, silicon dioxide (**silica**), $n_L \simeq 1.45$

with: $z_{L,H} =$ thickness in units of local wl. $\equiv d_{L,H}\left(\frac{n_{L,H}}{\lambda_0}\right) = 1/4$



The mirror **reflectance** is given by: [P. Beyersdorf, 2016]

$$R = |E_{R0}/E_{I0}|^2, \quad \text{where:} \quad \begin{pmatrix} E_{I0} \\ E_{R0} \\ E_{I1} \\ E_{R1} \\ \vdots \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots \\ r_{01} & 0 & 0 & t_{01}e^{-2\pi i z_1} & \cdots \\ t_{01}e^{-2\pi i z_1} & 0 & 0 & 0 & \cdots \\ 0 & 0 & r_{12} & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} E_{I0} \\ E_{R0} \\ E_{I1} \\ E_{R1} \\ \vdots \end{pmatrix},$$

$$r_{01} = \frac{n_1 - n_0}{n_1 + n_0} \qquad t_{01} = \frac{2n_0}{n_1 + n_0}$$

Coating Thermal Noise: how to reduce

Fluctuation-dissipation theorem: internal friction in coating layers produces a **Brownian noise** whose power spectral density is: [I.M. Pinto, 2009]

Reduce temperature T
(KAGRA, 3rd gen. Detectors)

$$S_B(f) = \frac{2k_B \boxed{T}}{\pi^{3/2} f} \frac{(1 - \boxed{\sigma}^2)}{\boxed{w}\, Y_s} \phi_c$$

Broaden the beam

$$\phi_c = b_L \boxed{d_L} + b_H \boxed{d_H},$$

Change the properties of the materials: Y, n, ϕ.

$$b_{L,H} = \frac{\lambda_0}{\sqrt{\pi}} \frac{\boxed{\phi_{L,H}}}{\boxed{n_{L,H}}} \left(\boxed{Y_{L,H}}/Y_s + Y_s/\boxed{Y_{L,H}}\right)$$

Reduce the loss angle changing the thickness of the dielectrics

High-refraction layers, Ta_2O_5, have the highest dispersions

Genetic Algorithm optimization of the mirror coatings

Objective: reduce the **thermal noise** (increase the sensitivity), reduce the Ta_2O_5 thickness

Constraints: fixed mirrors **reflectance**, maximum total thickness (others)

Individuals: $\underbrace{\{z_1, z_2, z_3, \dots, z_{N_d}\}}_{\text{Continuous variables: } z_i \in [0, 1/2)}$

Discrete variable: chromosomes are allowed to *change their length*

Fitness function: largely arbitrary, one possibility:

$$f(R, d_{\mathrm{Ta_2O_5}}) = \left(\frac{1-R}{15\ \mathrm{ppm}}\right)^2 + \left(\frac{d_{\mathrm{Ta_2O_5}}}{5000\ \mathrm{nm}}\right)^2 \longleftarrow \text{minimise}$$

Results: single doublet optimisation

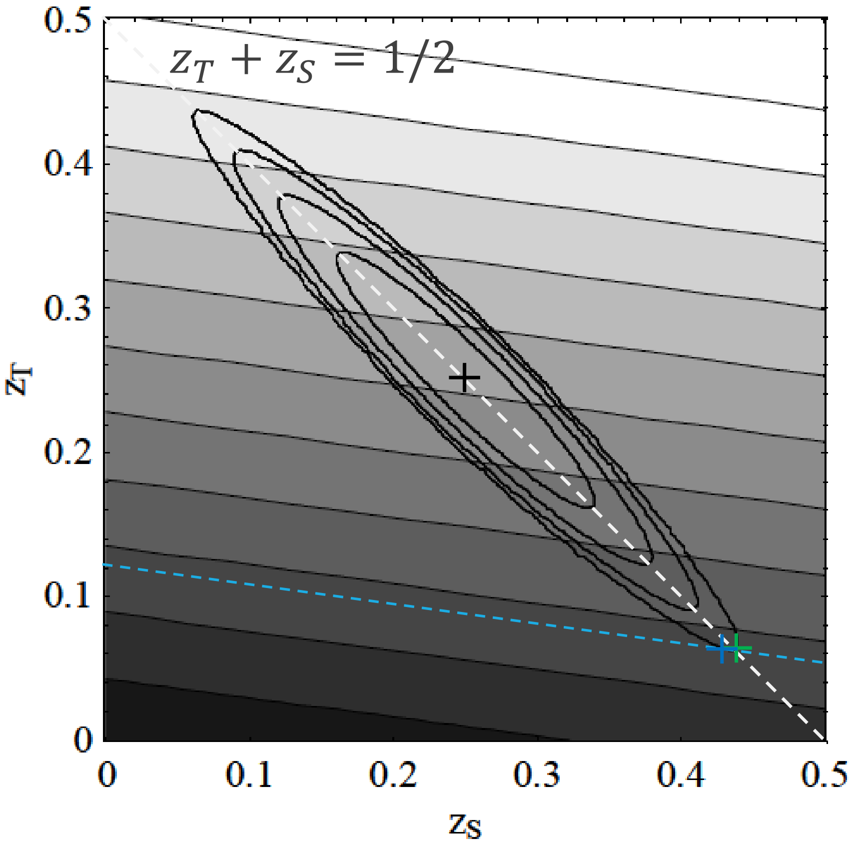


Figure: constant reflectance contours (ellipses) and constant Thermal Noise (straight lines) vs. optical thickness in units of local wavelength for a single $SiO_2 - Ta_2O_5$ doublet. [J. Agresti, 2006]

Standard layout: QWL layers

- $z_S = z_T = 1/4$
- maximum reflectance
- high Thermal Noise

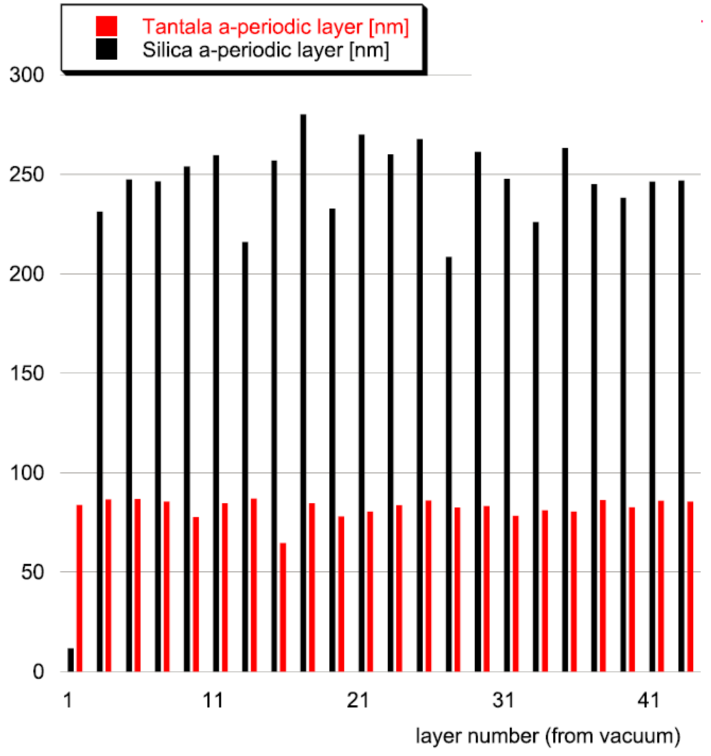
Approximate optimisation:

Half-wavelength doublets, $z_S = 3/8, z_T = 1/8.$

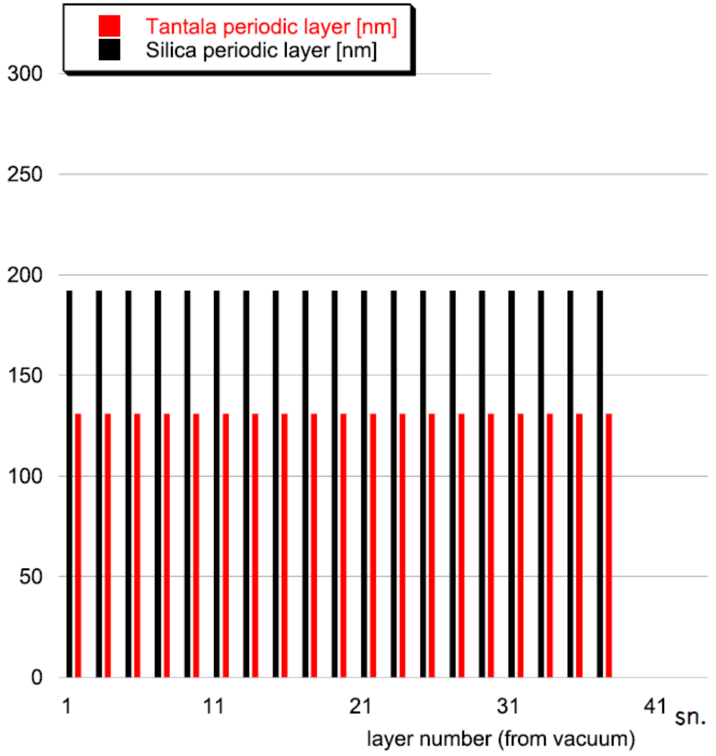
Exact optimisation:

- minimum Thermal Noise for fixed reflectance
- non-periodic layers

Results, comparison: 15 ppm loss $(1 - R)$

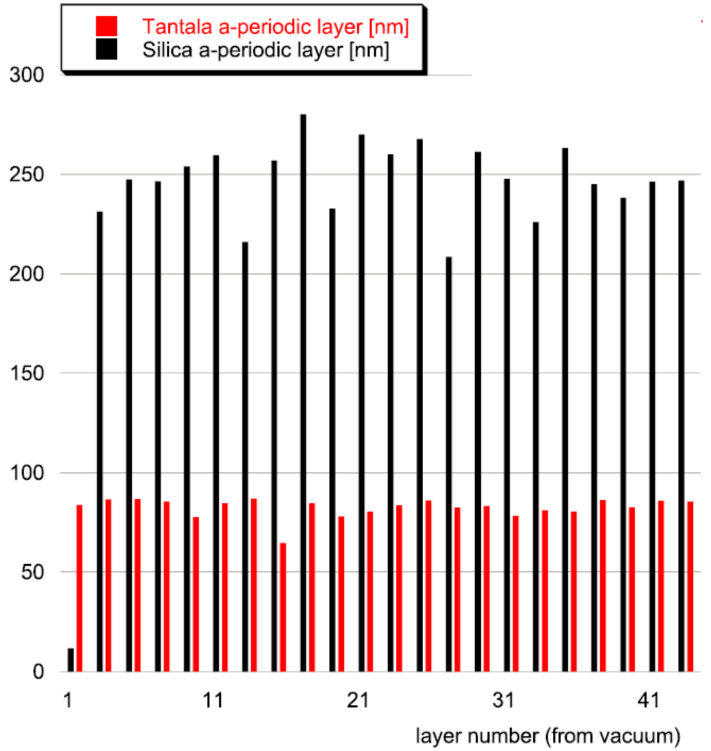


Non-periodic, 44 layers,
7033 nm thickness, **1816 nm Ta_2O_5**

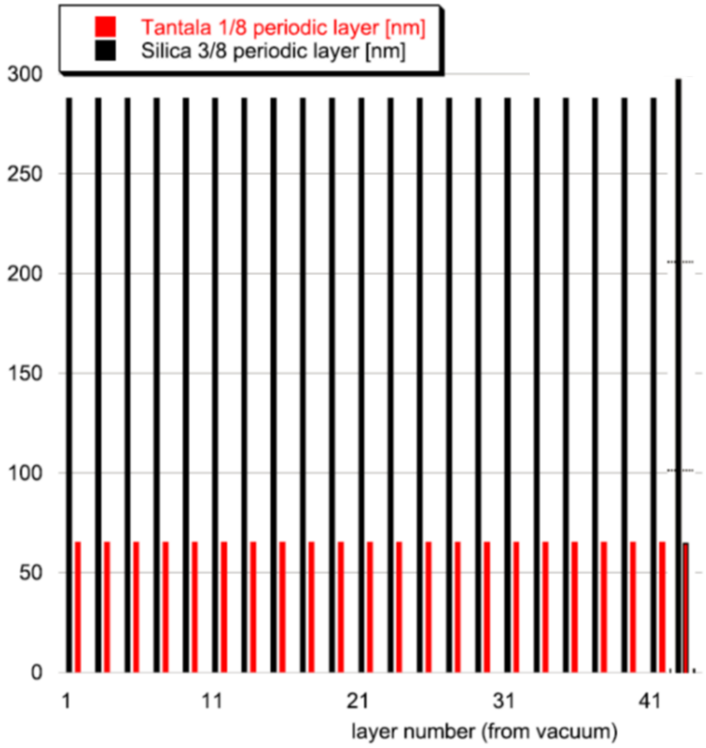


(Standard) Periodic $\lambda/4 + \lambda/4$, 38 layers,
6153 nm thickness, 3663 nm Ta_2O_5

Results, comparison: 44 layers



Non-periodic, **15 ppm loss**,
7033 nm thickness, 1816 nm Ta_2O_5



Periodic $3\lambda/8 + \lambda/8$, 60 ppm loss,
7766 nm thickness, **1430 nm Ta_2O_5**

Results: noise optimisation

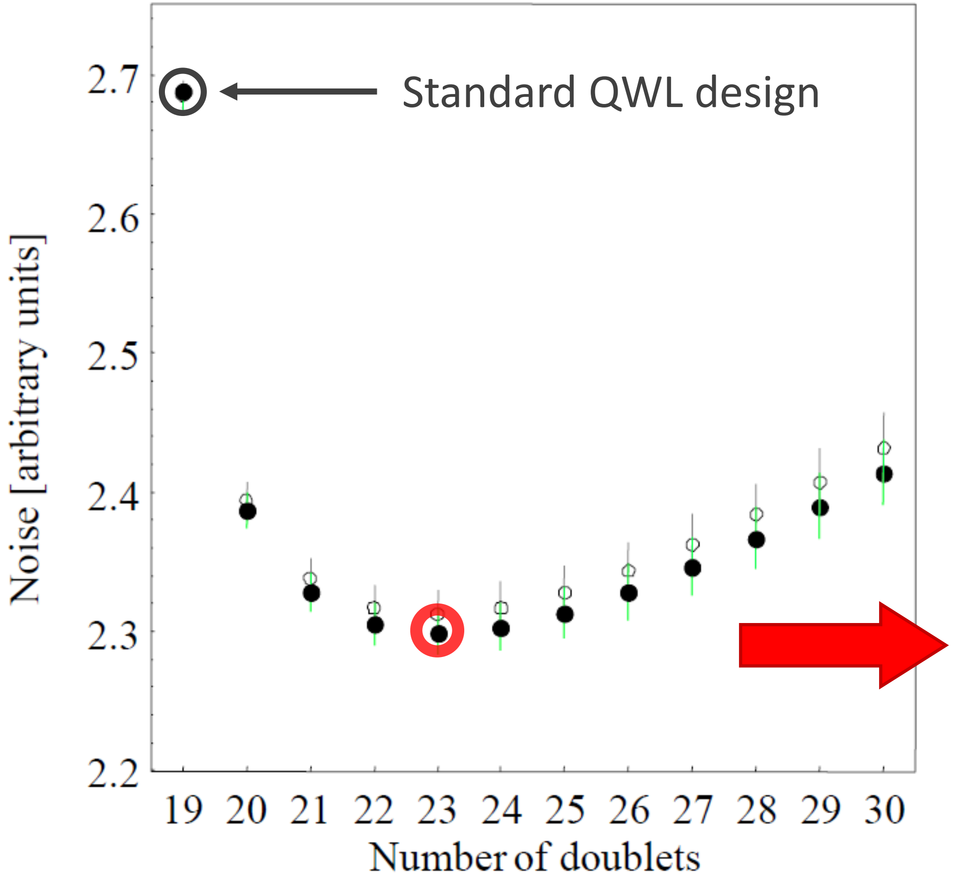


Figure: coating Thermal Noise at $1 - R = 8.3$ ppm as a function of the number of doublets N_d.

● : "exact" non-periodic optimisation
○ : approximate half-wavelength solution

Thermal Noise is reduced by 14%!!!
Event rate boosted by 25%!!!

$r_{\max} \propto 1/h_{\min}$, $\quad h_{min} \propto S_{\text{floor}}^{1/2}$, $\quad \text{Rate} \propto r_{\max}^3 \propto S_{\text{floor}}^{-3/2}$

Results summary for the coating optimisation

- Multilayer mirror coating of lowest Thermal Noise at a prescribed reflectivity can be designed by Genetic Algorithms;

- Non-periodic optimal solution can be found as well as periodic non-QWL approximate solutions;

- 14% reduction of the Thermal Noise and **25% boost of the event rate** for isotropic sources of Gravitational Waves.

Extra slides

GAs applications in High Energy Physics

Experimental HEP: [L. Teodorescu, 2007]

- Event selection for Higgs search at the LHC [K. Cranmer, arXiv:physics/0402030v1]
- Trigger optimisation [L1 and L2 CMS SUSY trigger – NIM A502 (2003) 693]
- Neural-netwok optimisation for Higgs search [F. Hakl et.al., talk at STAT2002]

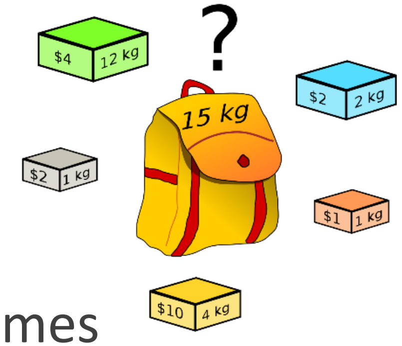
Theoretical/phenomenological HEP:

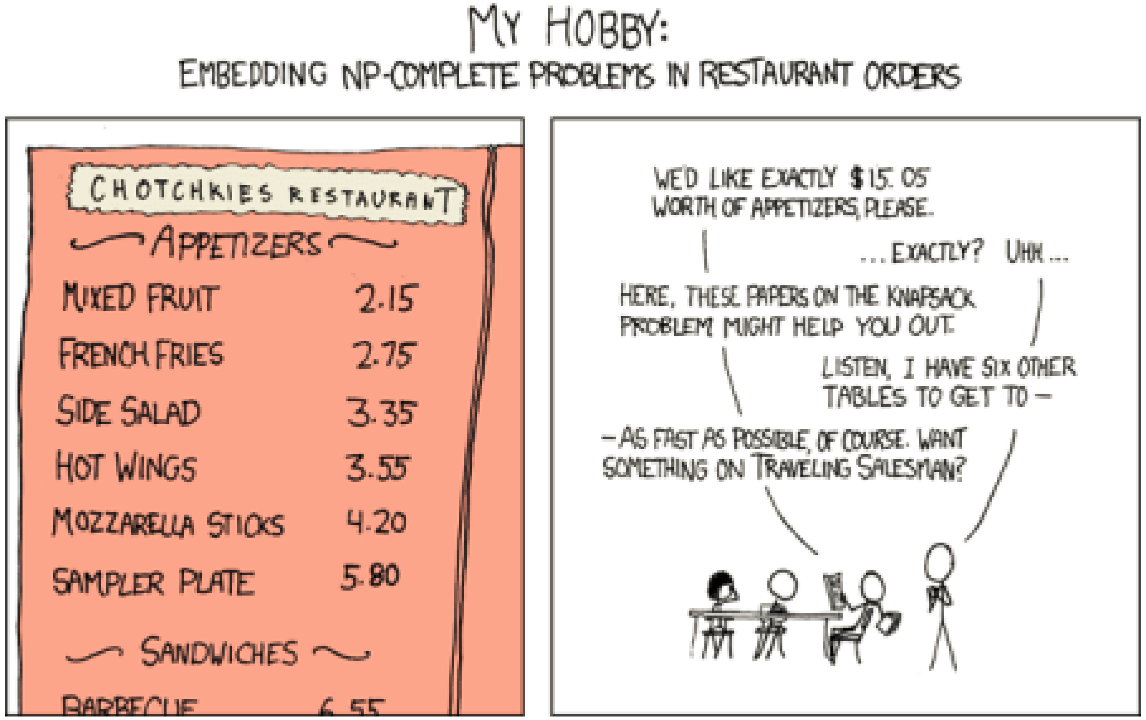
- Fitting isobar models to data for $p(\gamma, K^+)\Lambda$ [D.G. Ireland, arXiv:nucl-th/0312103v3]
- Discrimination of SUSY models [B.C. Allanach, arXiv:hep-ph/0406277]
- String Theory [F. Ruehle, JHEP08(2017)038]

Other applications of GAs: the knapsack problem

Simple problem in **combinatorial optimisation**:

"Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible." (Dantzig, 1930)



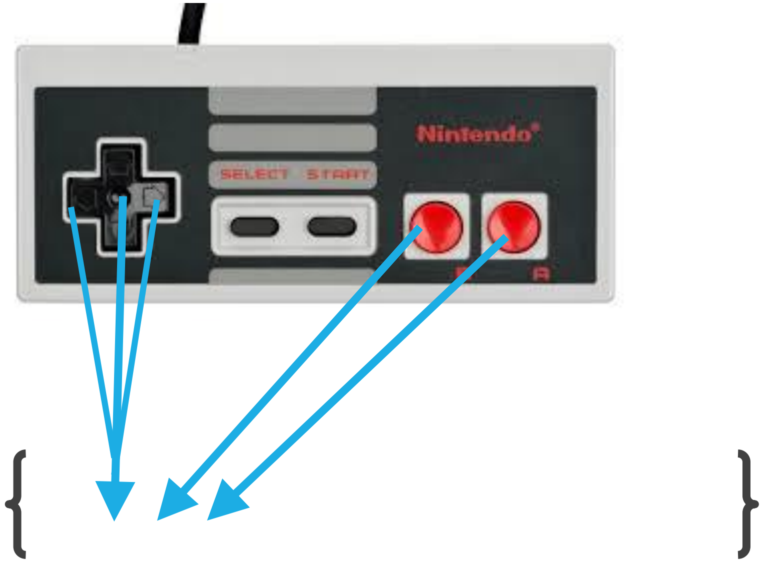


GAs solution:

- One gene for each object
- Variable length chromosomes
- Fitness function is the value of the knapsack

Having fun with GAs: beat Super Mario

https://www.youtube.com/watch?v=PHwMH28wFuM



{ }

- One gene per every quarter second
- Each gene is one of: ←,↑,→,↓,A,B or a combination of two of them
- The fitness function is the distance Mario travels combined with a built-in score function

Bibliography

Genetic Algorithms

T. Pang, “Introduction to computational Physics,” Cambridge University Press, New York (2006).

K.R. Williams, “Applications of Genetic Algorithms to a Variety of Problems in Physics and Astronomy,”

Master's Thesis, University of Tennessee, 2005.

Mirror Coatings and miscellanea about GW detectors

J. Agresti, et al., “Optimized multilayer dielectric mirror coatings for gravitational wave interferometers,” Proc. SPIE 6286, 628608 (2006).

R. Flaminio, et al., “A study of coating mechanical and optical losses in view of reducing mirror thermal noise in gravitational wave detectors,” Class. Quantum Grav. 27, 084030 (2010).

A. E. Villar, et al., “Measurement of thermal noise in multilayer coatings with optimized layer thickness,” Phys. Rev. D 81, 122001 (2010).

B. P. Abbott et al. (LIGO Scientific Collaboration and Virgo Collaboration), “Observation of Gravitational Waves from a Binary Black Hole Merger,” Phys. Rev. Lett. 116, 061102.