

(Meta)Heuristics for Physics

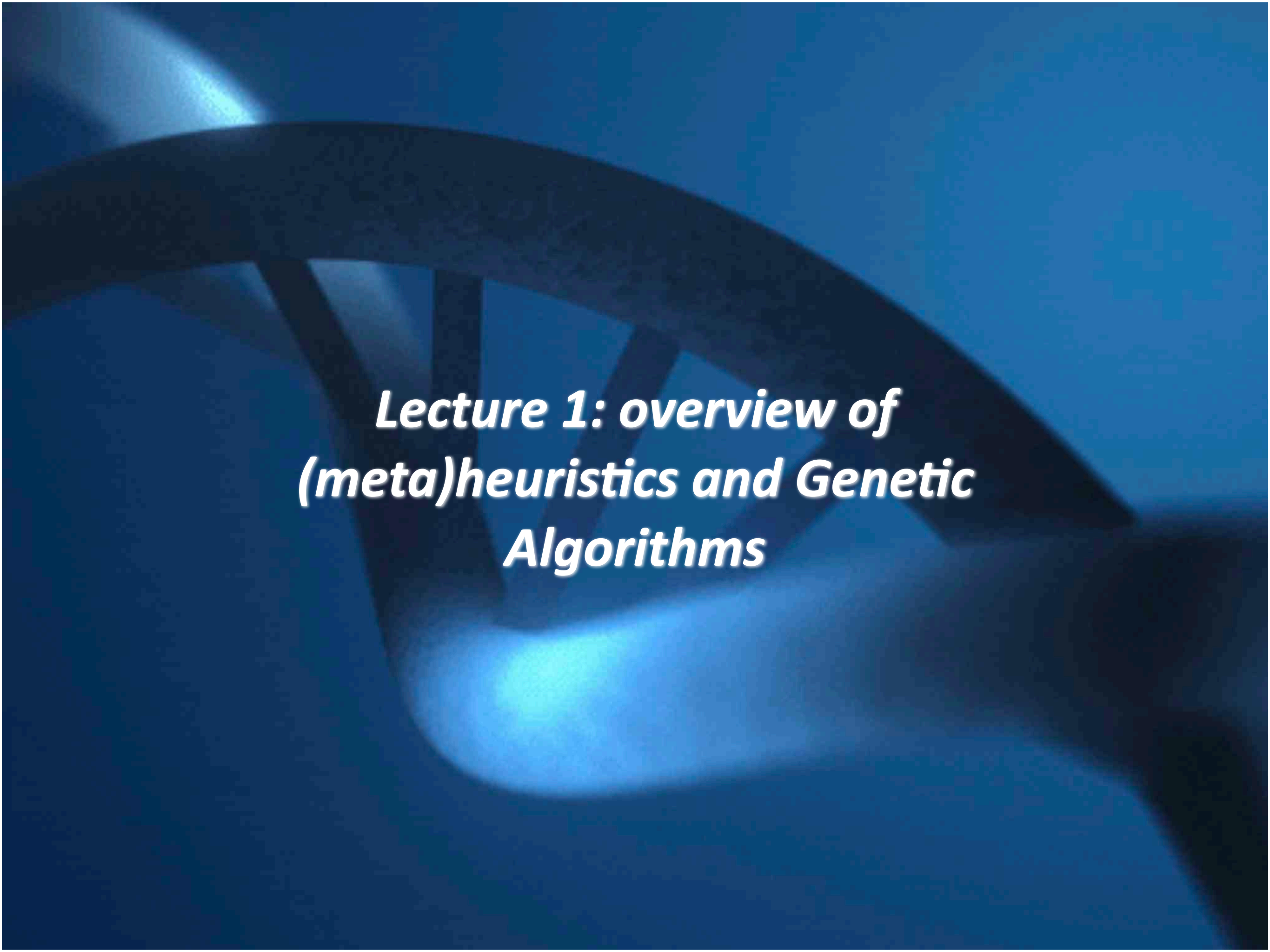
Steven Abel

(Durham Univ. IPPP+Maths)

ML in Mathematics and Theoretical Physics, Oxford 17-21 July

(Meta)Heuristics for Physics

- Lecture 1: Genetic Algorithms (GAs)
- Tutorial: Thomas Harvey
- Lecture 2: Quantum Annealing and AQC
- Tutorial: Luca Nutricati
- Lecture 3: Overview of applications



***Lecture 1: overview of
(meta)heuristics and Genetic
Algorithms***

Overview ...

- Motivation
- Metaheuristics overview
- Some GA introduction and background
 - largeness or otherwise of string (and other) landscapes
 - How do they work?
 - Why do they work?
- GA's on a toy problem: TaxiCab numbers
- GA in combination with other methods
- GA as search tools in continuous parameter spaces

Motivation ...

- String theories typically produce vast theory spaces. e.g 10^{500} or even $10^{(272000)}$
Denef, Douglas; Ashok, Douglas; Denef, Douglas, Greene, Zokowski; Taylor, Wang
- Finding the “Standard Model” typically requires solving a set of Diophantine conditions (e.g. setting some index to get 3 generations, anomaly cancellation, GSO conditions etc)
- Such tasks are typically computationally hard. e.g. for an NP complete problem difficulty (probably) can increase rapidly with the size of the search space (but the solution can be verified in polynomial time). *Halverson, Ruehle; Halverson, Plesser, Ruehle, Tian*
- Other problems in physics also present big data problems: e.g. finding a suitable set of parameters in some BSM model that satisfies all experimental constraints
- But there are some venerable techniques that remain powerful: e.g. **Genetic Algorithms** *Turing; Barricelli; Fraser, Burnell; Crosby; Bremermann; **Holland**; Goldberg; Jones*

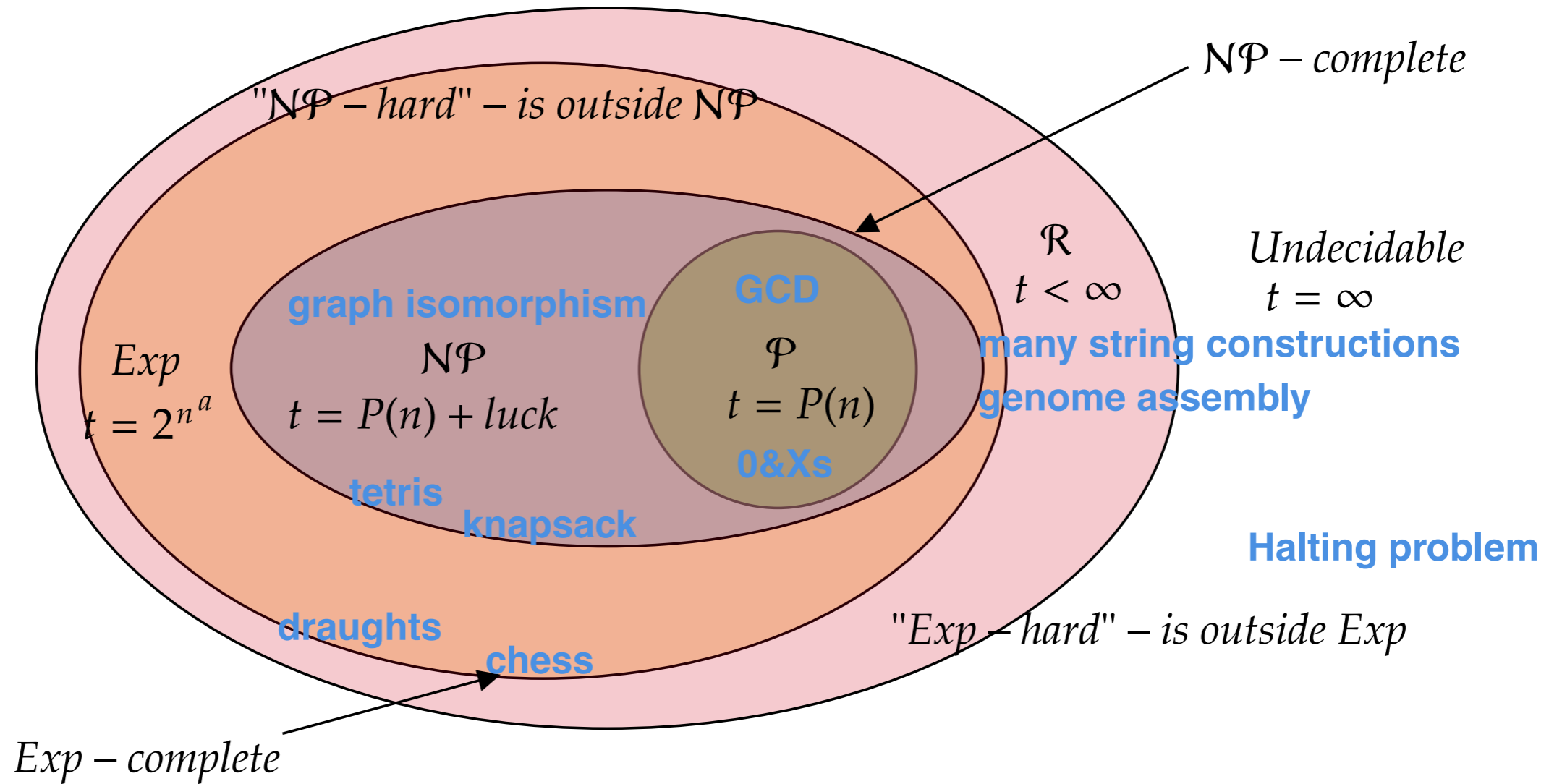


Metaheuristics overview

What is a Metaheuristic?

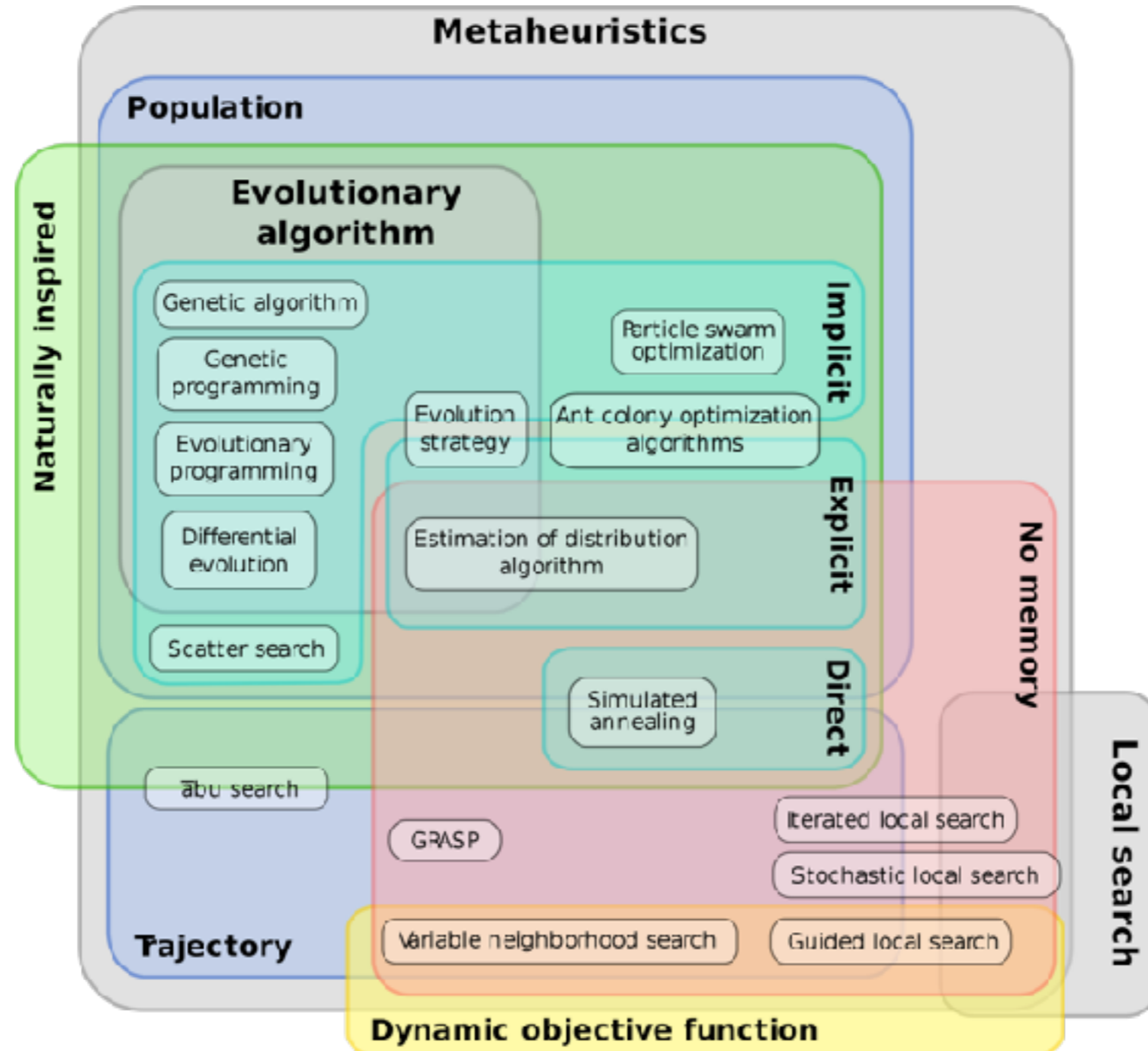
- Nature finds solutions that would be virtually impossible for us to solve: e.g.
 - a) Virtually all biological processes: e.g. find a molecule that can transport and release oxygen
 - b) Find a configuration of proteins that can disable a virus
 - b) Various other problems that look like they were solved by an omniscient being ...
- The main theme is that nature does not solve problems by blind searches, but by building complex solutions out of simpler parts in various ways.

What sort of problems are we thinking about? Assuming $P \neq NP$ then ...



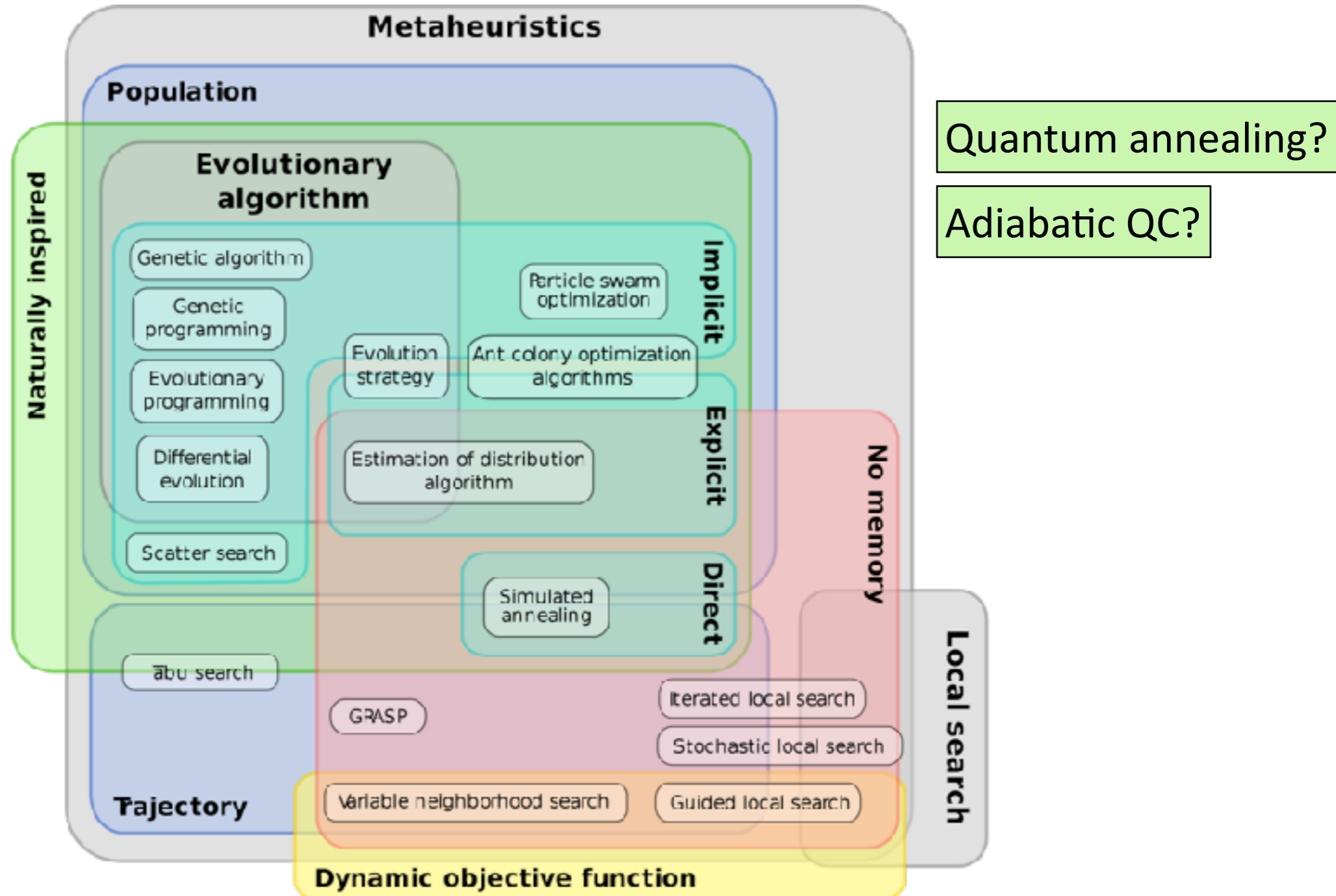
Many string constructions seem "reducible" to genome assembly!

Common meta heuristics :



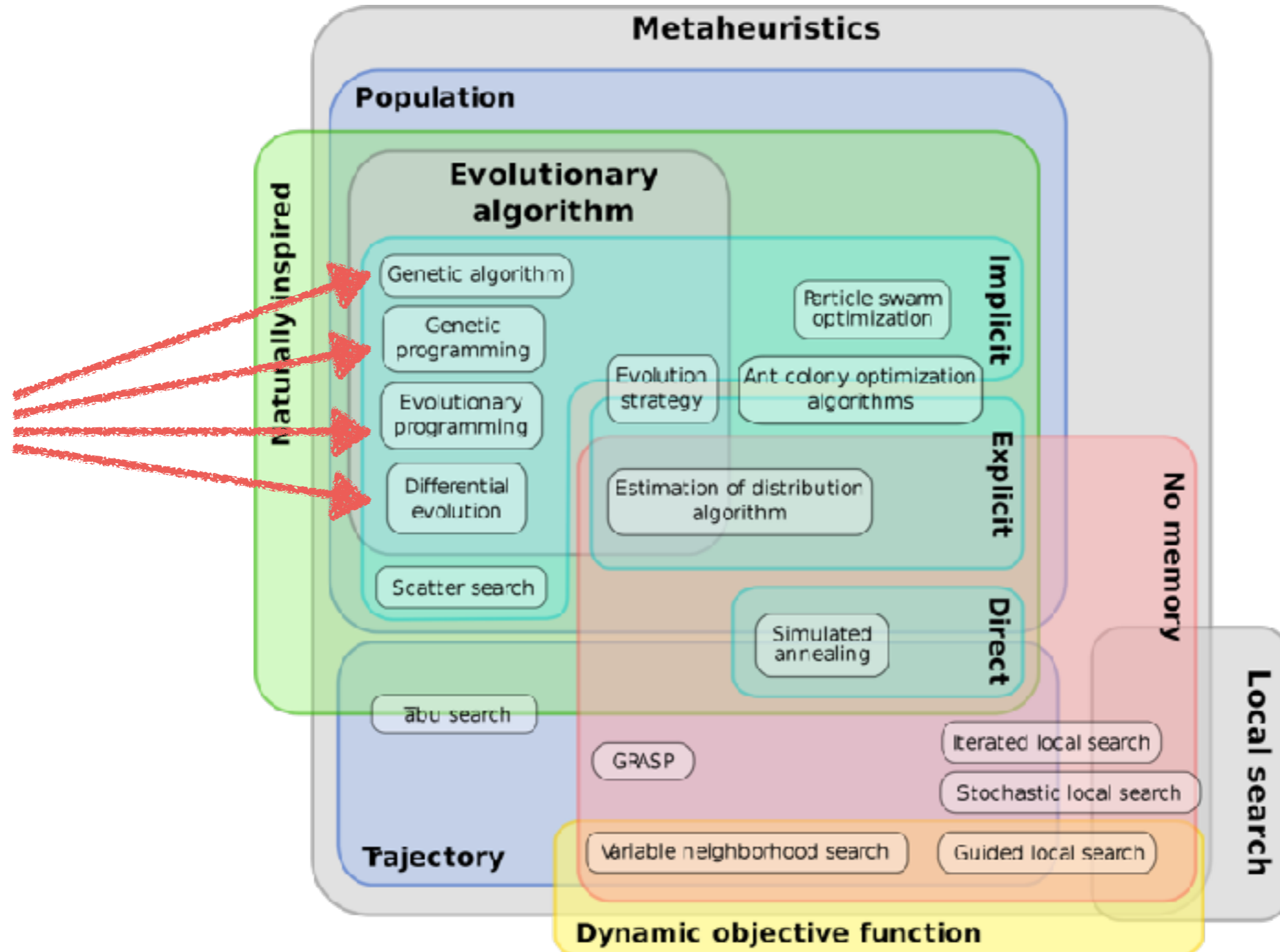
Common meta heuristics :

Not obvious where quantum annealing or adiabatic quantum computing belong here:



Genetic Algorithms

Basic idea: imitate evolutionary principles of fitness, selection, breeding ...

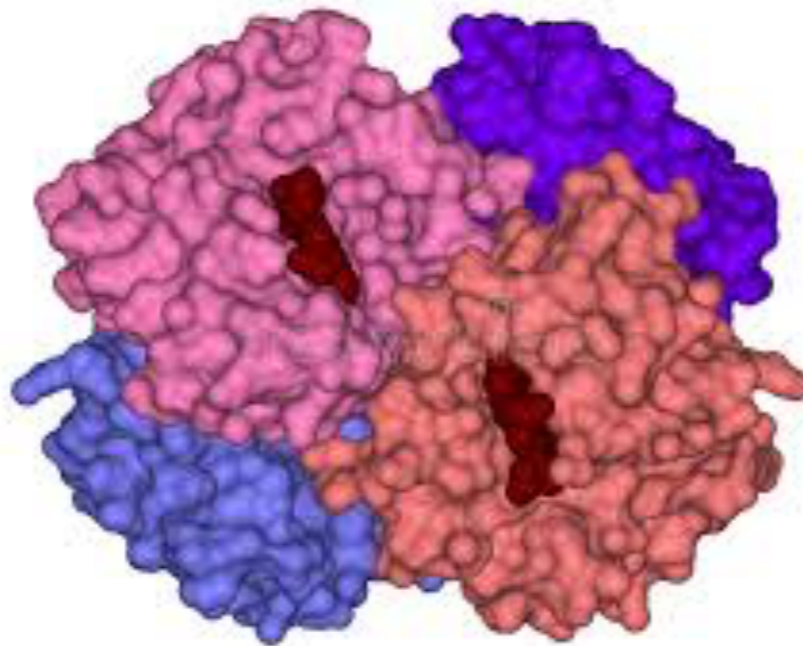
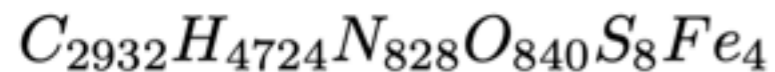




GA introduction and background

Why? Let's consider Nature's landscapes ...

- Consider biological landscapes: problems that were solved by evolution
e.g. Haemoglobin molecule.



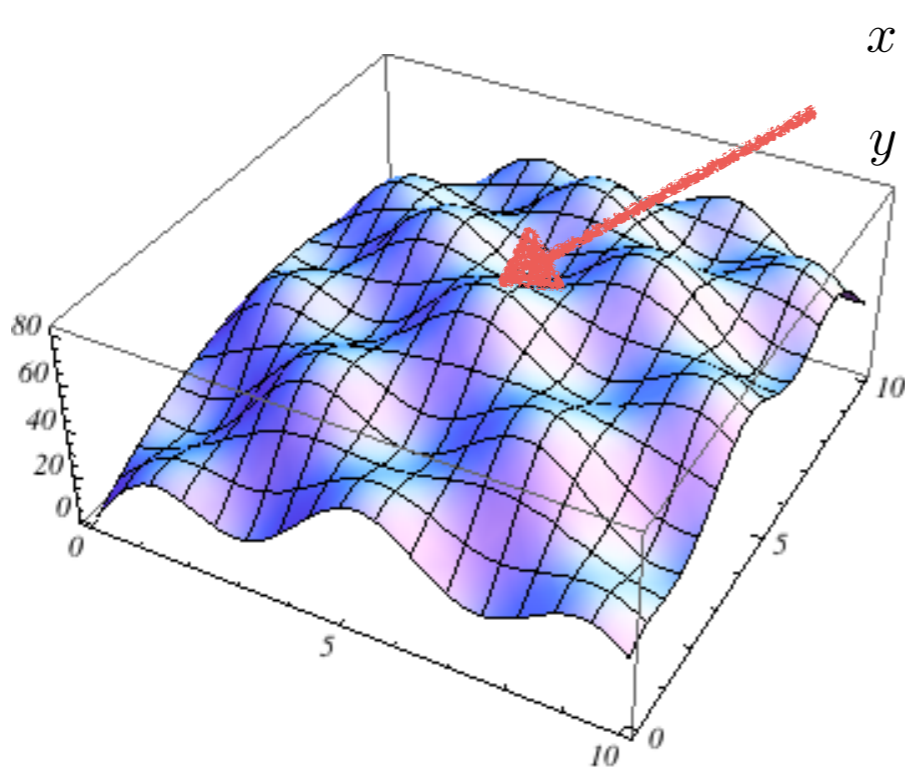
I am not the solution!!

- # Choices of C,H,...Fe from 92 elements ... 10^{18334} !!!
- Or should it be: 2 legs of 141 amino acids, plus 2 legs of 146.
20 amino acids means only ... 10^{747} ??
- Or maybe chances of making human DNA = $1 : 4^{3,000,000,000}$

GA's for searching a string sized landscape

- GA's (based on evolutionary dynamics) work most effectively when
 - many criteria being applied at the same time Holland; David; Reeves, Rowe;
 - correlation between "goodness of fit" and "closeness to maximum" Jones, Forrest
- Disadvantage: by their nature statistical information very hard/impossible to get

Example landscape task: find global maximum to 250 decimal places without using calculus ...



$$\begin{aligned} x &= a.bcdf... \\ y &= g.hijkl... \end{aligned} \implies 10^{500}$$

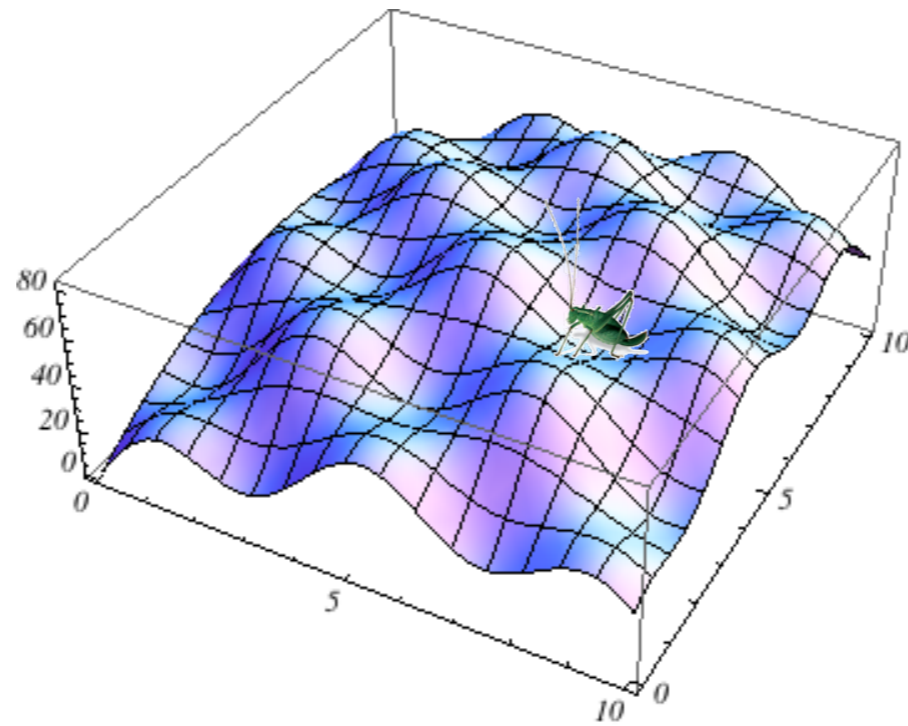
$$f(x, y) = 12 \left(\cos \frac{3y}{2} \sin \frac{3x}{2} + x + y \right) - x^2 - y^2.$$

Define a “creature” and write out its coordinates => genotype

Terminology: *Genotype* = data. *Phenotype* = $f(x,y)$.

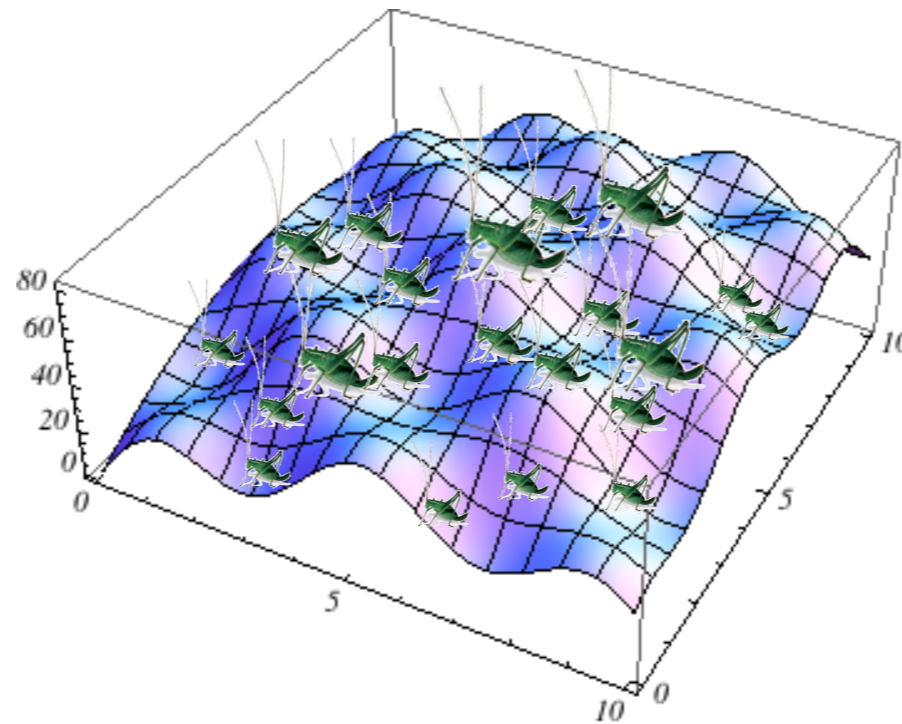
$$x = a.bcdf...$$

$$y = g.hijkl...$$



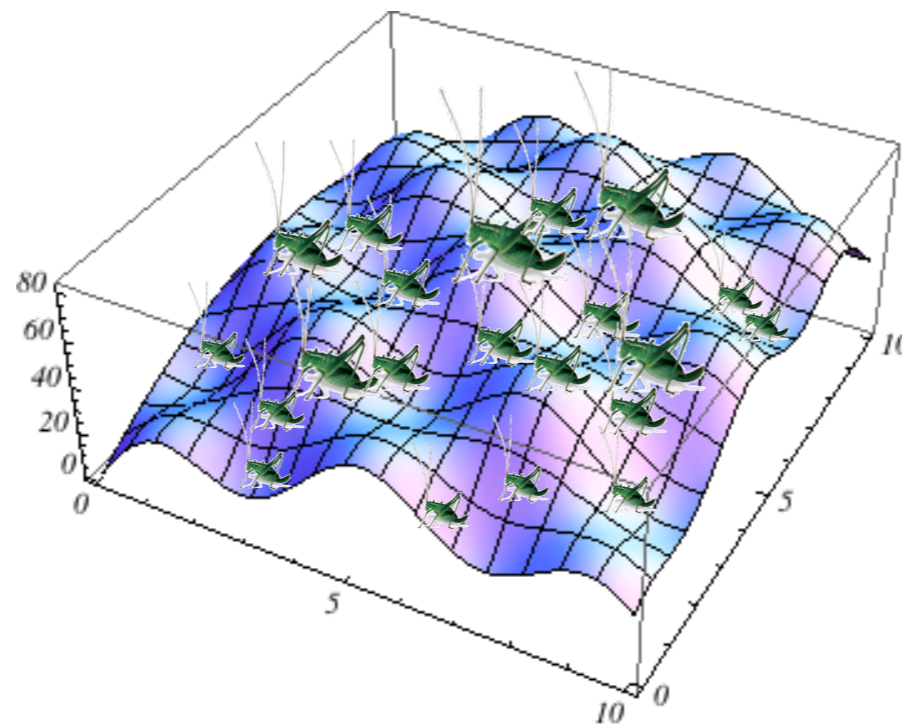
Work with a population of typically ~100 individuals initially sprinkled at random

Step 0: Define fitness function, and work out the fitness F of each individual (e.g. $F = f(x,y)$ in this case).

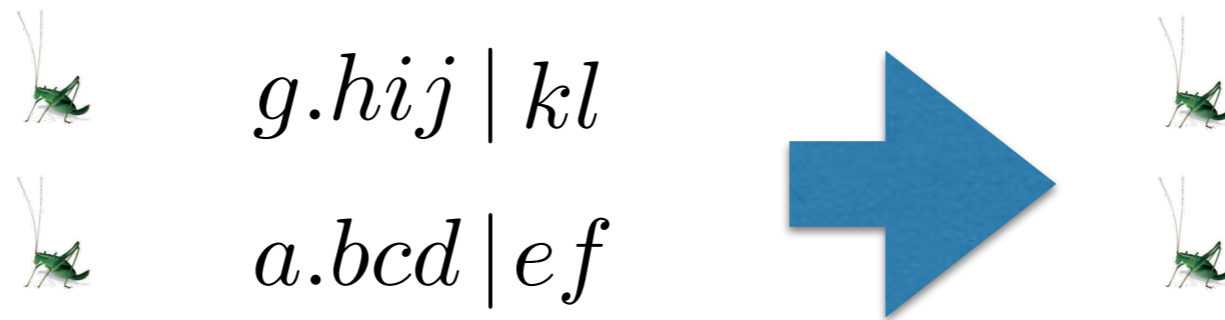


Step 1: Selection: Select pairs for breeding such that the most fit individuals can breed several times, while unfit ones might not breed at all: e.g. “roulette wheel” based on ranking k , with $P_1 = \alpha P_{N_{\text{pop}}}$. Alpha is similar to a “learning rate”

$$P_k = \frac{2}{(1 + \alpha)N_{\text{pop}}} \left(1 + \frac{N_{\text{pop}} - k}{N_{\text{pop}} - 1} (\alpha - 1) \right)$$



Step 2: breeding: cut and splice genotypes of breeding pairs somehow (not really crucial how) to make an entirely new population of the same size.



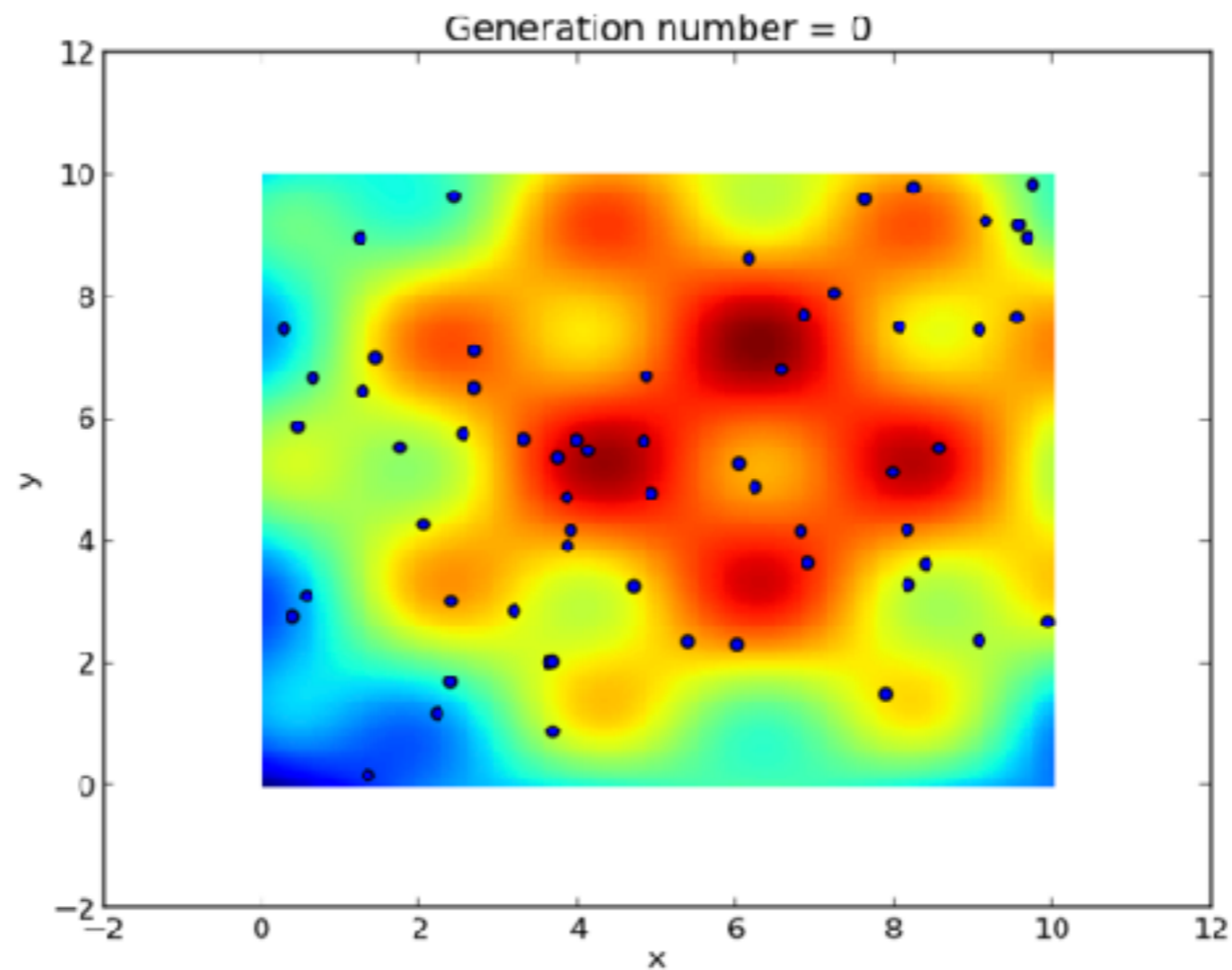
Step 3: Mutation of a randomly chosen small percentage of digits (alleles).



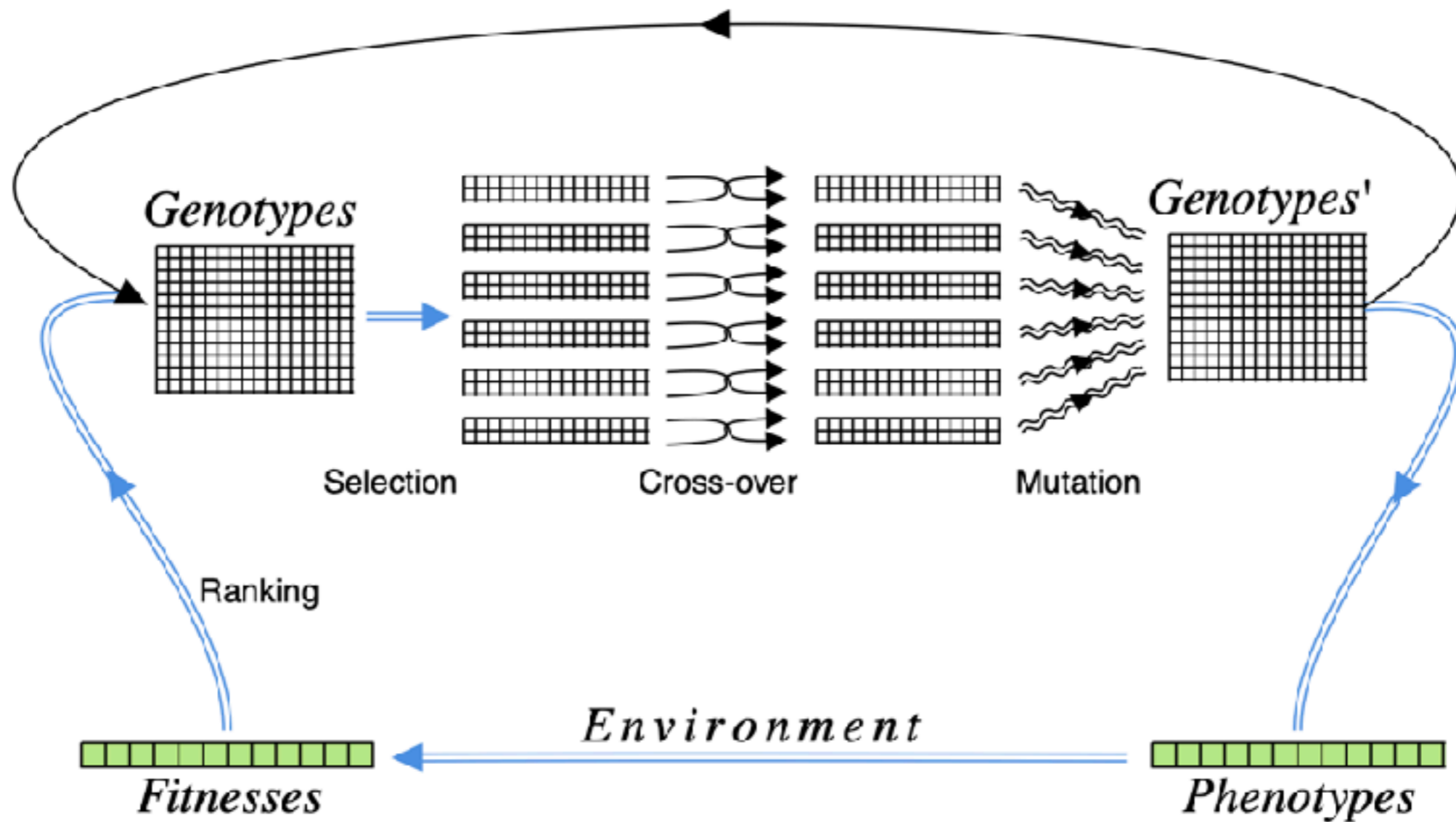
$a.bcde f'ghi'j.j\dots$

Steps 4 ... infinity: rinse and repeat. The population should converge round solutions.

Summary — three crucial ingredients: **Selection** (favours the optimisation); **Breeding/crossover** (propagates favourable properties); **Mutation** (prevents stagnation: evolution proceeds by punctuated equilibria)



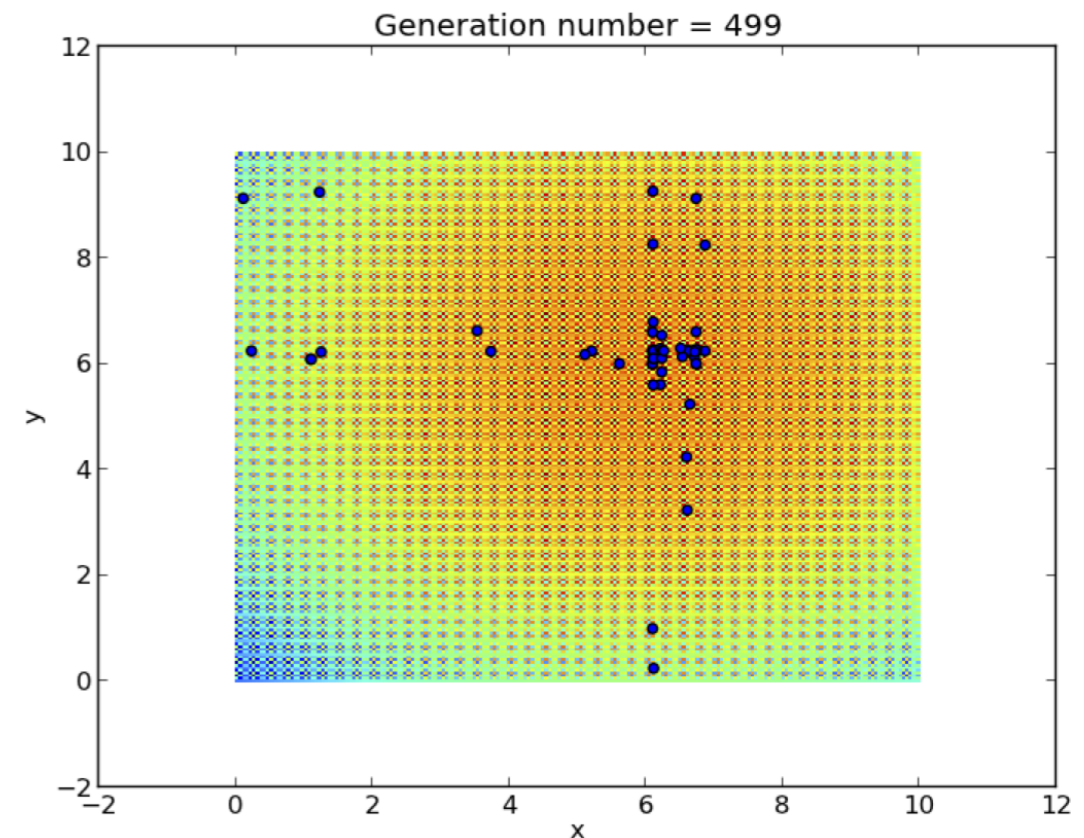
Summary — three crucial ingredients: **Selection** (favours the optimisation); **Breeding/crossover** (propagates favourable properties); **Mutation** (prevents stagnation: evolution proceeds by punctuated equilibria)



Why do they work?

- Holland proposed a probabilistic explanation for the efficiency of genetic algorithms: based on growth rate of “good” schema S , e.g. here $S = 61***62***$
- Holland argues that initial growth of a good schema in the population is exponential
- Selection pushes towards convergence
- Mutation pushes system away from convergence
- Some controversy in 1990s, rehabilitated somewhat by Poli. (D. White, “An Overview of Schema Theory”, 1401.2651)
- Fitness/distance correlation seems to be important
Jones+Forrest; Collard, Gaspar, Clergue, Escazu

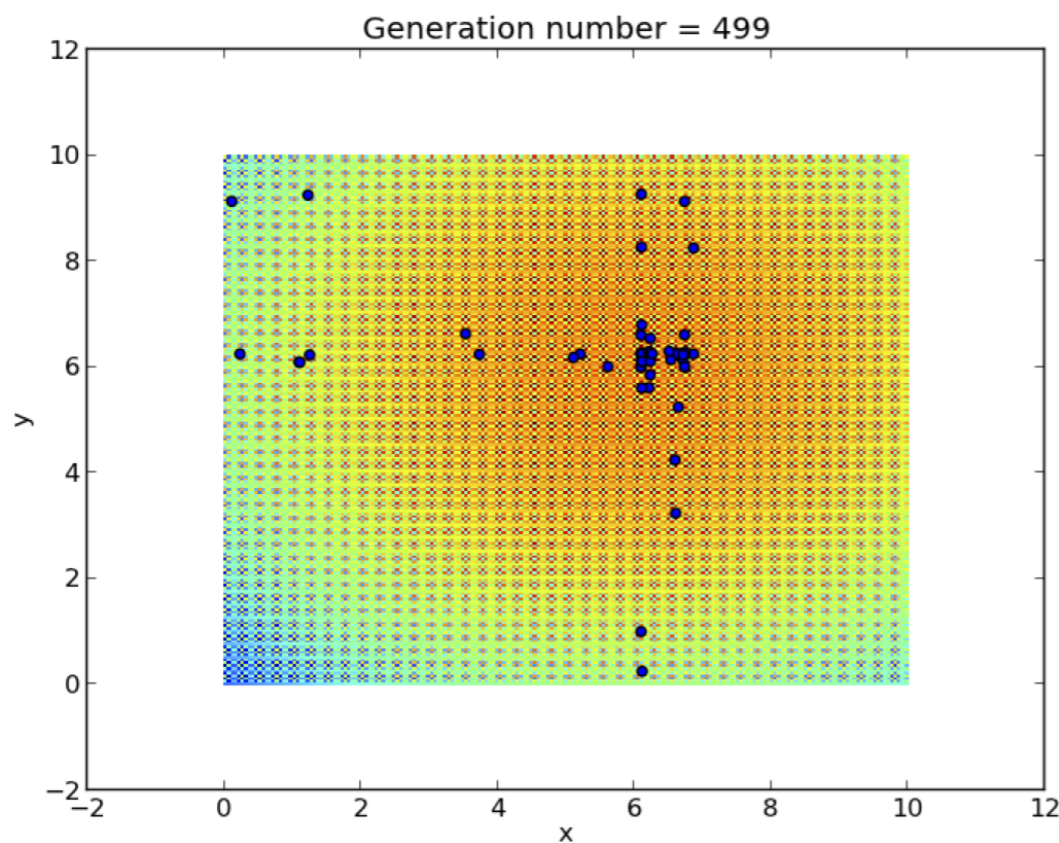
In this example the leading digits of x and y are schema and get propagated throughout the population



Why do they work?

- In detail suppose we have $n(S,t)$ members of population with schema S
- Can use simple probabilistic arguments to incorporate the effect of a single-point crossover destroying S , and mutations at a rate p_m per allele to find a lower bound

$$n(S, t + 1) \geq n(S, t) \frac{f_S(t)}{\bar{f}} \left(1 - \frac{d(S)}{l - 1} \right) (1 - p_m)^{o(S)}$$



↑ Avg fitness of members with S
↑ In this example order $o=4$
↑ defining length $d=7$

$S = 61 * * * 62 * * *$

Why do they work?

- Initial growth of $n(S,t)$ is exponential
- At late times find equilibrium for average fitness determined by p_m
- *i.e. Selection pushes towards convergence and mutation pushes system away*

$$n(S, t + 1) \geq n(S, t) \frac{f_S(t)}{\bar{f}} \left(1 - \frac{d(S)}{l - 1} \right) (1 - p_m)^{o(S)}$$

Optimisation:

Like any machine learning technique you can run into problems unless you optimise ...



- Fitness — rank selection often works best to overcome flat maxima
- Selection — Elitist selection (copy fittest individual into new population and kill weakest). Also various kinds of selection: tournament selection, roulette wheel, etc
- Breeding — two or more point cross-over to avoid edge effects
- Optimise mutation rate (See later)
- Creep mutation (on phenotype) to overcome “Hamming walls” e.g. 0.999... ~ 1.0000...

Y. Akrami, P. Scott, J. Edsjo, J. Conrad and L. Bergstrom (2009)



GA toy example: TaxiCab numbers

Let's see these principles at work in an example: taxicab numbers ...



$$\begin{aligned}1729 &= 1^3 + 12^3 = 9^3 + 10^3, \\4104 &= 9^3 + 15^3 = 16^3 + 2^3, \\20683 &= 24^3 + 19^3 = 10^3 + 27^3, \\32832 &= 32^3 + 4^3 = 18^3 + 30^3, \\&\dots\end{aligned}$$

More general problem becomes very “anomaly cancellation like”: numbers that are expressible as the sum of n numbers to the k 'th power or m numbers to the k 'th power:

$$\sum_i^m x_i^k = \sum_j^n y_j^k = T(k, m, n)$$

So the smallest $T(3,2,2)$ number is 1729. Pythagorean triples are $T(2,1,2)$ numbers. Fermat's last theorem is there are no $T(k,1,2)$ numbers when $k > 2$. etc. And anomaly matching says that there is a $T(3,m,n)$ number for every global symmetry!

How would we set up a problem like this?

How are we going to map the integers? Binary encoding works best. Let each number be written

$$\mathbf{w} = [x_1, x_2, y_1, y_2]$$

$$w_i = \sum_{l=0}^{\beta-1} 2^l \tau_l^i \quad \tau_l^i \in \{0, 1\}$$

Then the a given trial set of numbers is a binary number with 4β alleles ...



$$\tau = [0, 1, 0, \dots, 1, 1, 0, \dots, 0, 1, 1, 0, \dots, 1, 0, 1, 1, \dots, 0]$$

The search space is $2^{4\beta}$

Aside for later (relevant to annealers): if we define

$$w_0^3 + w_1^3 - w_2^3 - w_3^3 \equiv \sum_i a_i w_i^3$$

then optimising the fitness of the equation part is isomorphic to minimising a sextic spin Hamiltonian:

$$\begin{aligned} f &= -(w_0^3 + w_1^3 - w_2^3 - w_3^3)^2 \\ &= - \sum_{i,j_1,j_2,j_3,k,l_1,l_2,l_3} a_i a_k 2^{j_1+j_2+j_3+l_1+l_2+l_3} \tau_{i,j_1} \tau_{i,j_2} \tau_{i,j_3} \tau_{k,l_1} \tau_{k,l_2} \tau_{k,l_3} \end{aligned}$$

which can in turn be reduced to an Ising model.

Fitness function? Has to include the equations and the constraints — here are chunks of my code:

```
def neg_f(w):                                # negative fitness -- this is problem specific -
    ans = 0
    if w[0] == w[1]: ans += lamb
    if w[0] == w[2]: ans += lamb
    if w[0] == w[3]: ans += lamb
    if w[1] == w[2]: ans += lamb
    if w[1] == w[3]: ans += lamb
    if w[2] == w[3]: ans += lamb
    if w[0] == 0: ans += lamb
    if w[1] == 0: ans += lamb
    if w[2] == 0: ans += lamb
    if w[3] == 0: ans += lamb

    ans += abs(w[0]**3+w[1]**3-w[2]**3-w[3]**3)
    return ans
```

(lamb = 500 is the constraint penalty. Note we have to hammer all the trivial solutions.)

A solution to the problem has $f = 0$. Non-solutions have $f < 0$.

Here we see one of the advantages of GAs. The procedure is very flexible. The GA part of the code operates on $\tau = [0, 1, 0, ..1, 1, 1, 0, ..0, 1, 1, 0, ..1, 0, 1, 1, ..0]$ and we just need to figure out the fitness f for each binary code. i.e. the binary code could be defining anything - string theories - whatever, but everything from this point onwards looks the same.

Selection. The easiest way to operate is to use ranking and simply keep the population in an array and reorder it by fitness. Then the likelihood of breeding is a constant given by the position in the array:

```
def reorder(): # reorders creatures with least fit first and fittest last
    global creatures
    fit = fitness(creatures)
    newi = np.argsort(fit) # returns indices for re_ordering creature with fittest goes last
    oldcreatures = creatures[:] # copy the np.array
    creatures = oldcreatures[newi]

def breeding_pair(): # Generates a set of breeding pairs randomly based on fitness
    global likelihood
    pair=[]
    for k in range(0,2):
        accept = 0
        while accept == 0:
            candidate = np.random.randint(pop)
            ok = np.random.random()/pop
            if ok < likelihood[candidate]:
                accept = 1
                pair.append(candidate)
    return pair
```

This is the equivalent of “rank” selection which is a special version of roulette-wheel sel’n.

Types of selection ...

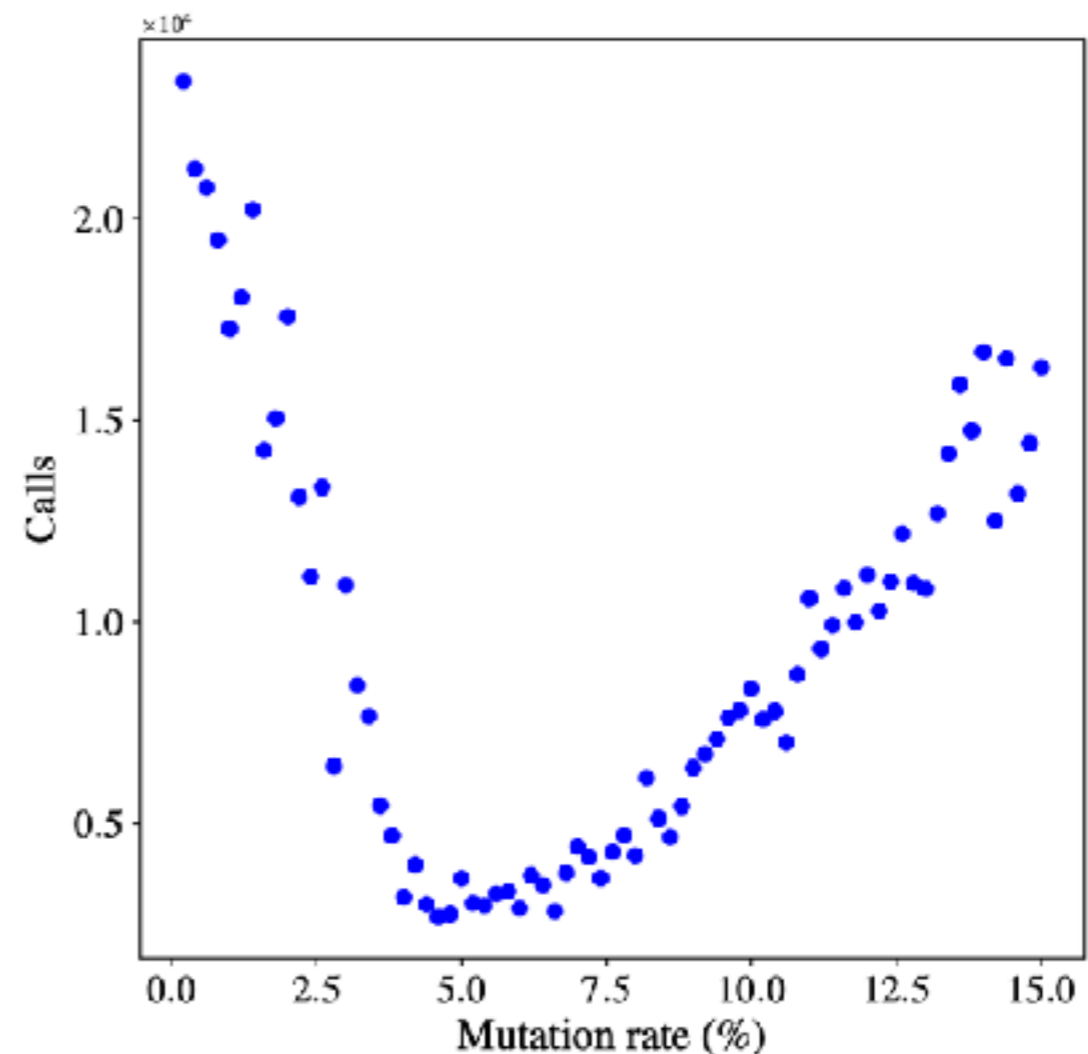
- Roulette wheel selection: (likelihood function of fitness)
- Rank selection (likelihood based purely on rank)
- Steady state (no generations - just replace a few at a time)
- Stochastic selection (similar to rank selection)
- Tournament selection (winner of tournaments selected for breeding)
- Elitist selection
- Boltzmann selection (similar to Metropolis approach with changing temp.)
- Truncation selection

Next mutation:

```
def mutate(creatures2): # introduce mutations everywhere except the best creature
    mutation_number=int(mutate_rate*ch_length*pop)
    for l in range(0,mutation_number):
        victim_num = np.random.randint(0,pop-1)
        entry = np.random.randint(0,ch_length)
        allele = np.random.randint(0,2)
        creatures2[victim_num][entry]=allele
    return creatures2
```

Elitism — we don't mutate in the last entry. The last entry is before reordering a copy of the fittest individual of the previous generation.

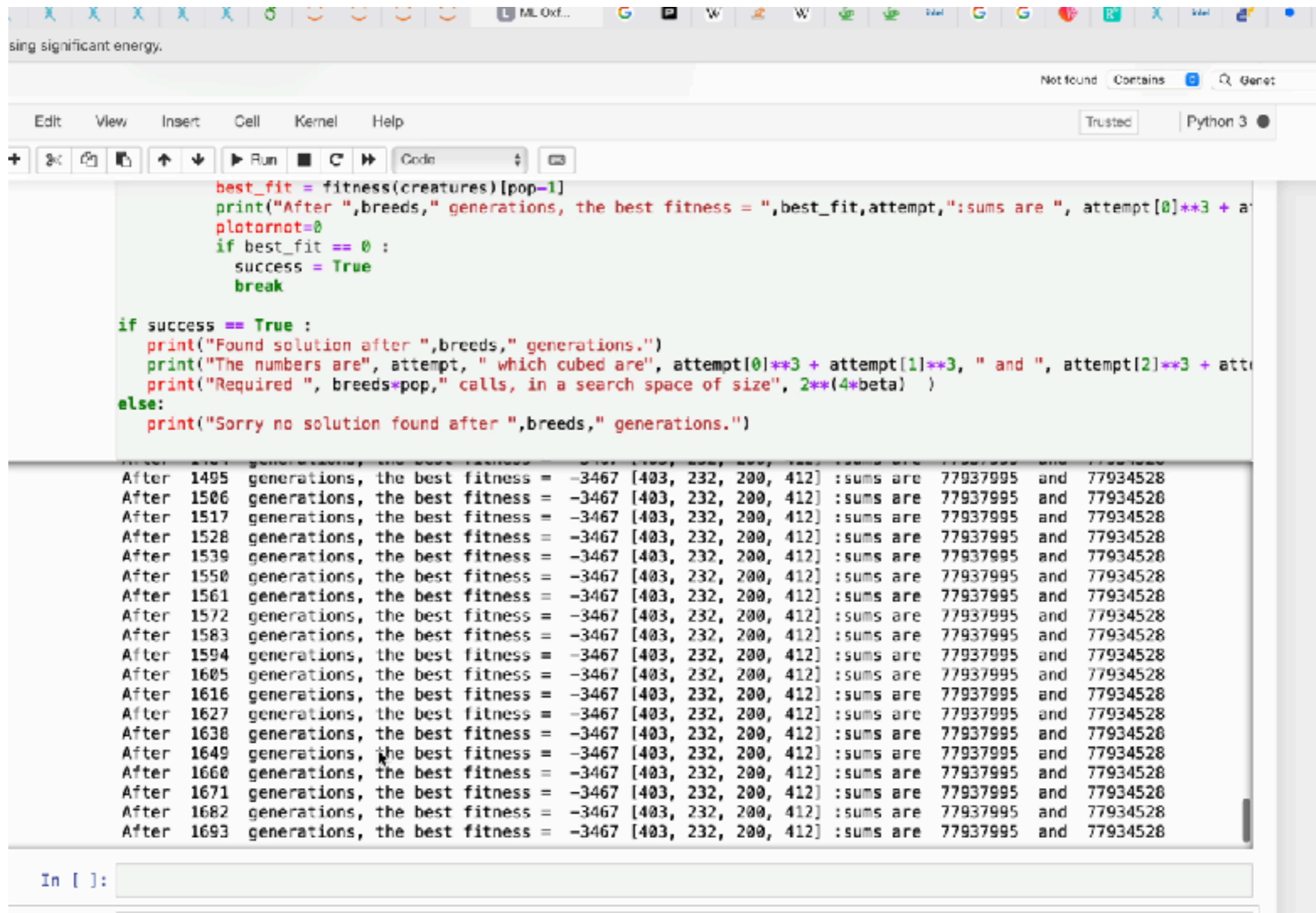
Optimise mutation rate: this is a way to see if it is genuinely working as a GA: [SAA Rizos](#); [SAA, Nutricati, Spannowsky](#)



For 5% mutation rate and a population of 150 we get ...

```
File Edit View Insert Cell Kernel Help Trusted Python 3
+ ↻ ↺ ↻ ↻ ↻ ↻ ↻ Run Code
In [77]: plotornot=0
creatures = build()
success = False
for breeds in range(0,meteor_after):
    global creatures
    reorder() # order in terms of fitness ranking with fittest last
    creaturestmp=creatures[pop-1][:]
    for i in range(0,pop): breed(creatures)
    creatures=mutate(creatures)
    creatures[0]=creaturestmp[:]# copies the previous fittest individual over the least fit
    plotornot += 1
    reorder()
    if (plotornot > 10):
        attempt = pheno(creatures[pop-1])
        best_fit = fitness(creatures)[pop-1]
        print("After ",breeds," generations, the best fitness = ",best_fit,attempt,":sums are ", attempt[0]**3 + a
        plotornot=0
        if best_fit == 0 :
            success = True
            break
if success == True :
    print("Found solution after ",breeds," generations.")
    print("The numbers are", attempt, " which cubed are", attempt[0]**3 + attempt[1]**3, " and ", attempt[2]**3 + att
    print("Required ", breeds*pop," calls, in a search space of size", 2**(4*beta) )
else:
    print("Sorry no solution found after ",breeds," generations.")
<ipython-input-76-850ea09191f4>:32: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (wh
ich is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant
to do this, you must specify 'dtype=object' when creating the ndarray.
vv= np.array([yy[0:splice],zz[splice:]])
After 10 generations, the best fitness = -24339 [98, 149, 162, 28] :sums are 4249141 and 4273480
After 21 generations, the best fitness = -3122 [73, 153, 47, 157] :sums are 3970594 and 3973716
After 32 generations, the best fitness = -111 [119, 131, 145, 96] :sums are 3933250 and 3933361
After 43 generations, the best fitness = -88 [106, 148, 164, 28] :sums are 4432808 and 4432896
After 54 generations, the best fitness = -88 [106, 148, 164, 28] :sums are 4432808 and 4432896
After 65 generations, the best fitness = -88 [106, 148, 164, 28] :sums are 4432808 and 4432896
After 76 generations, the best fitness = -88 [106, 148, 164, 28] :sums are 4432808 and 4432896
```

whereas for 90% mutation rate and a population of 150 (a.k.a. a random scan) we get ...



```
best_fit = fitness(creatures)[pop-1]
print("After ",breeds," generations, the best fitness = ",best_fit,attempt,":sums are ", attempt[0]**3 + attempt[1]**3 + attempt[2]**3)
plotornot=0
if best_fit == 0 :
    success = True
    break

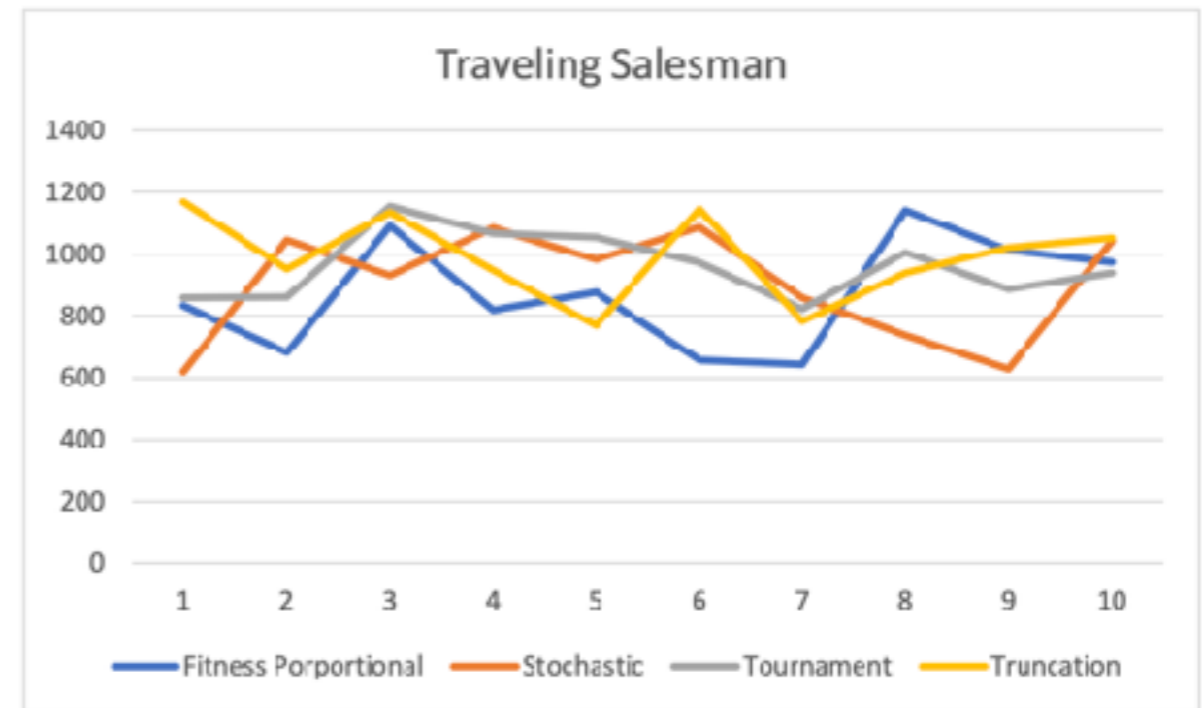
if success == True :
    print("Found solution after ",breeds," generations.")
    print("The numbers are", attempt, " which cubed are", attempt[0]**3 + attempt[1]**3, " and ", attempt[2]**3 + attempt[3]**3)
    print("Required ", breeds*pop," calls, in a search space of size", 2**(4*beta) )
else:
    print("Sorry no solution found after ",breeds," generations.")
```

After 1495 generations, the best fitness = -3467 [403, 232, 200, 412] :sums are 77937995 and 77934528
After 1506 generations, the best fitness = -3467 [403, 232, 200, 412] :sums are 77937995 and 77934528
After 1517 generations, the best fitness = -3467 [403, 232, 200, 412] :sums are 77937995 and 77934528
After 1528 generations, the best fitness = -3467 [403, 232, 200, 412] :sums are 77937995 and 77934528
After 1539 generations, the best fitness = -3467 [403, 232, 200, 412] :sums are 77937995 and 77934528
After 1550 generations, the best fitness = -3467 [403, 232, 200, 412] :sums are 77937995 and 77934528
After 1561 generations, the best fitness = -3467 [403, 232, 200, 412] :sums are 77937995 and 77934528
After 1572 generations, the best fitness = -3467 [403, 232, 200, 412] :sums are 77937995 and 77934528
After 1583 generations, the best fitness = -3467 [403, 232, 200, 412] :sums are 77937995 and 77934528
After 1594 generations, the best fitness = -3467 [403, 232, 200, 412] :sums are 77937995 and 77934528
After 1605 generations, the best fitness = -3467 [403, 232, 200, 412] :sums are 77937995 and 77934528
After 1616 generations, the best fitness = -3467 [403, 232, 200, 412] :sums are 77937995 and 77934528
After 1627 generations, the best fitness = -3467 [403, 232, 200, 412] :sums are 77937995 and 77934528
After 1638 generations, the best fitness = -3467 [403, 232, 200, 412] :sums are 77937995 and 77934528
After 1649 generations, the best fitness = -3467 [403, 232, 200, 412] :sums are 77937995 and 77934528
After 1660 generations, the best fitness = -3467 [403, 232, 200, 412] :sums are 77937995 and 77934528
After 1671 generations, the best fitness = -3467 [403, 232, 200, 412] :sums are 77937995 and 77934528
After 1682 generations, the best fitness = -3467 [403, 232, 200, 412] :sums are 77937995 and 77934528
After 1693 generations, the best fitness = -3467 [403, 232, 200, 412] :sums are 77937995 and 77934528

In []:

The importance of rank selection

Often doesn't make a difference Champlin



but here it is crucial: because of degeneracy of solutions and discrete nature of the problem there is not a good alternative selection probability based on the fitness function ... e.g. is this solution

After 76 generations, the best fitness = -35 [14, 22, 21, 16] :sums are 13392 and 13357

35 times worse than this one (which comes up a lot) ?

After 164 generations, the best fitness = -1 [14, 30, 23, 26] :sums are 29744 and 29743

when a nearby solution is ... [24, 2, 20, 18] which cubed are 13832 and 13832

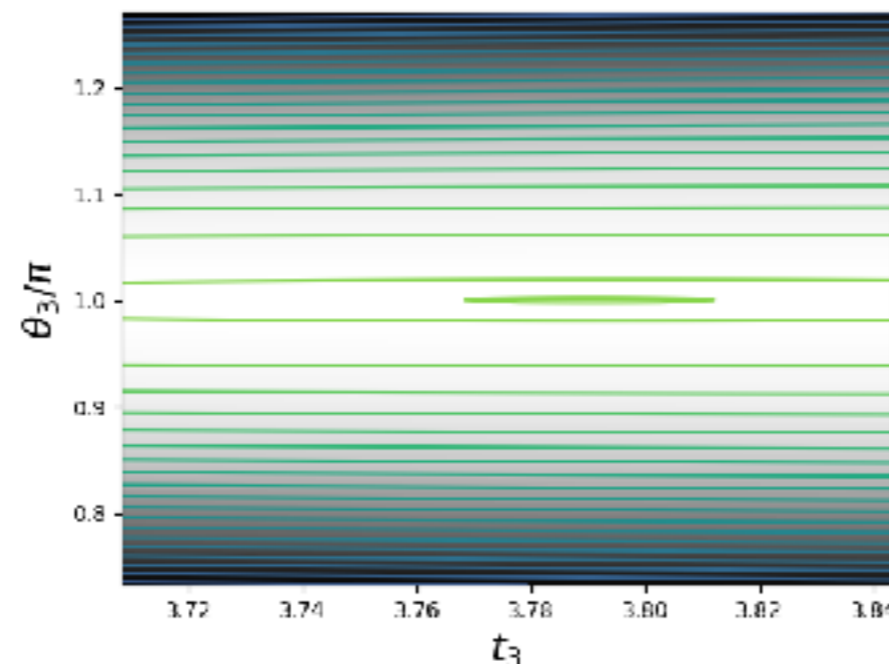


Combining GA with other methods

Why combining?

- GAs are very good at exploring parameters spaces with arbitrarily complicated problems
- Their initial convergence rate is very fast
- The final convergence rate is slow
- This is the effect of mutation continually pushing the population out of the “wells”

Because of this in many cases, especially in physics, it makes sense to combine GAs with clustering and “gradient descent” type techniques. e.g. **GA+Kmeans+Nelder-Mead**. This method turns out to be very useful for searching for KKLT like and LVS like metastable string vacua where the potentials are very flat in some directions.



Abdussalam, SAA, Cicoli, Quevedo, Shukla

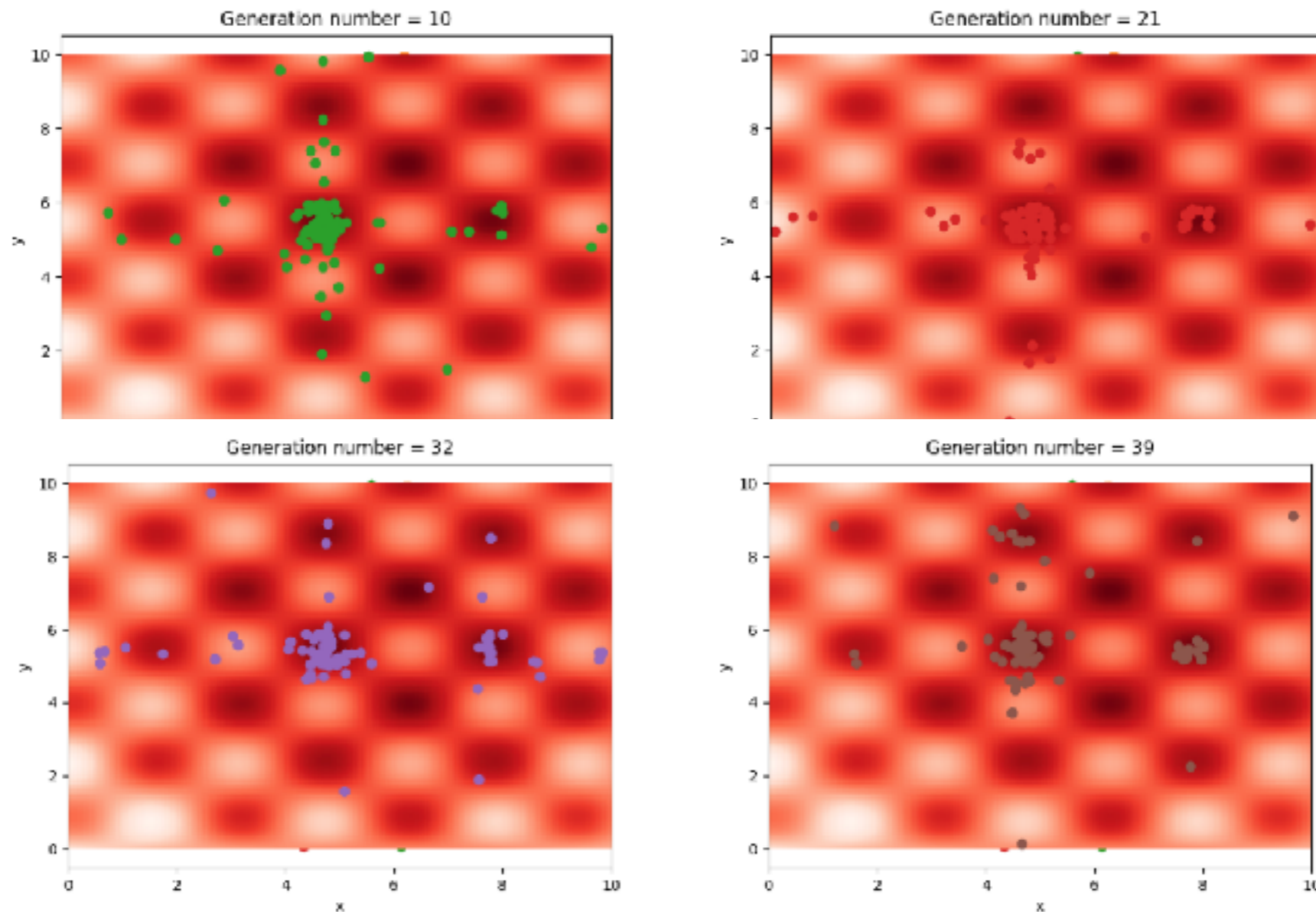
Example

Abdussalam, SAA, Cicoli, Quevedo, Shukla

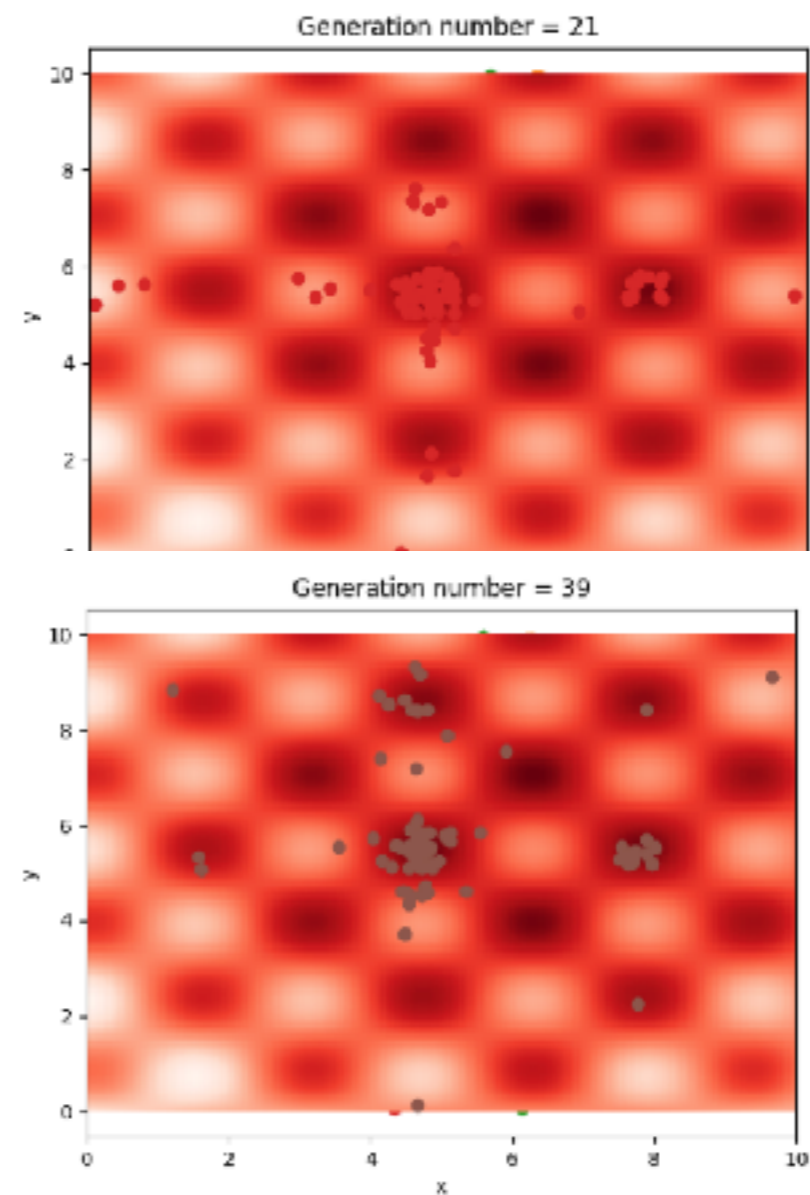
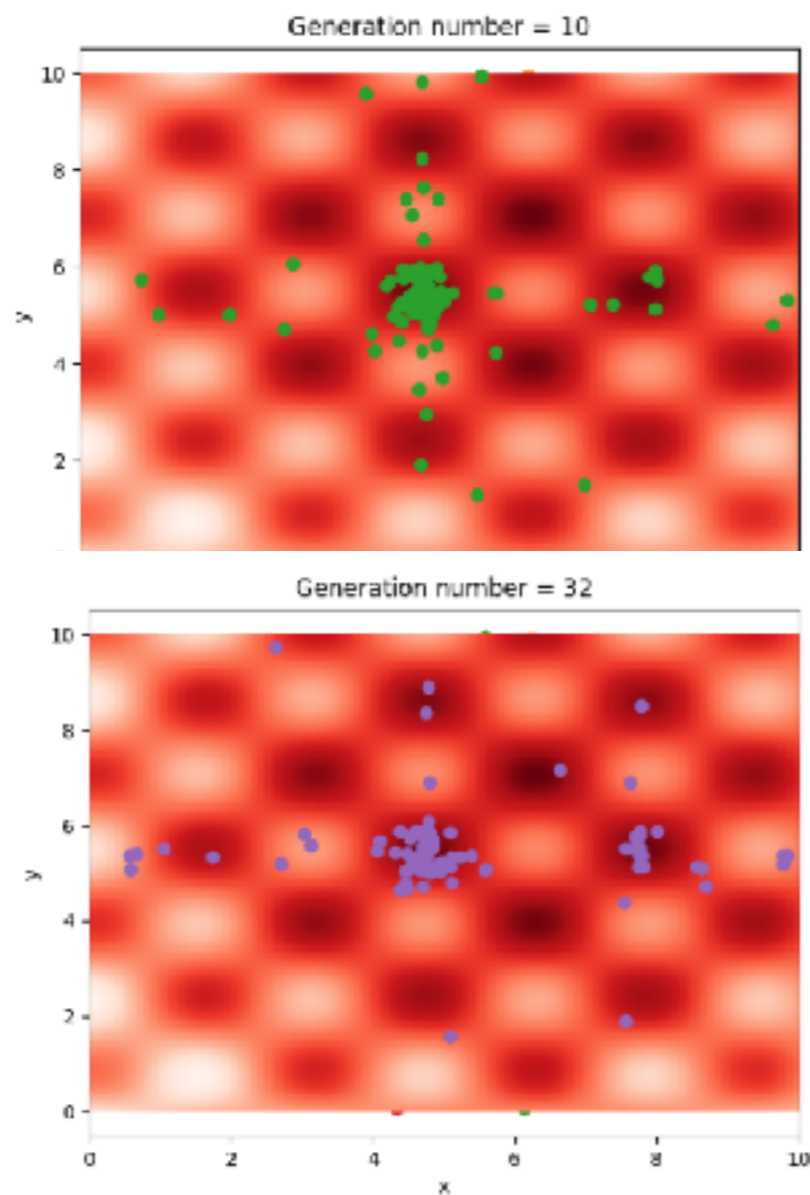
- Find all the *local* minima of this function:

$$V(x, y) = -(36 \sin(2y) \cos(2x) + 12(x + y) - x^2 - y^2)$$

- If we construct and run a GA over this landscape for a while it does this:

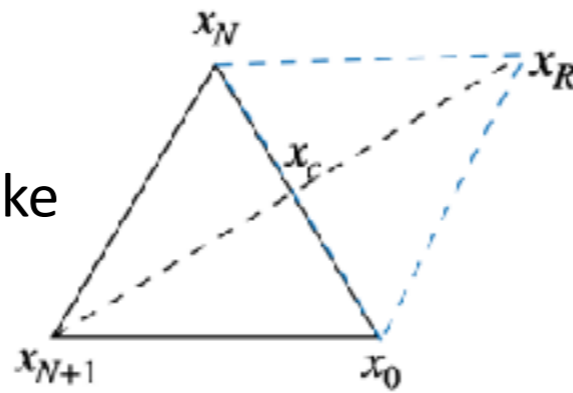


- As we suspected the convergence to the final minima stops due to the mutation. For this particular kind of problem (where we are not looking for a discrete solution in a huge parameter space) the GA stops being very powerful after about 10 generations.
- However it clearly gives useful information in the early stages.



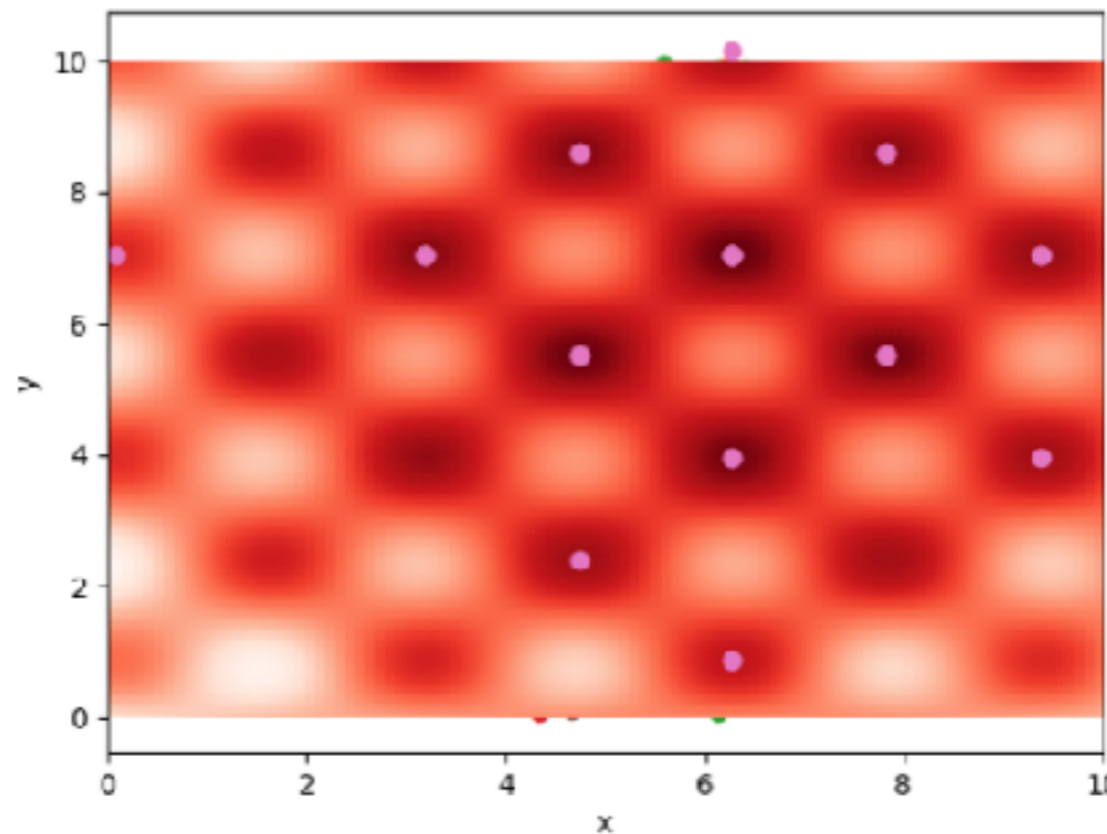
- Therefore run the GA until we find clusters, then use a clustering algorithm (K-means clustering) which groups the creatures into clusters from which we make simplexes.
- Finally use a Nelder-Mead descent algorithm. (This is a heuristic method btw)

Nelder-Mead: simplex moves like



where x_{N+1} is the “worst” point

Repeat several times ...





Probing continuous parameter spaces

pMSSM; GAs as a tool in parameter spaces

SAA w/ Cerdano & Robles

Interesting feature of GA's is the fitness distance correlation, and how it affects the behaviour of the population as it evolves. (Checked with MultiNest — Bayesian Inference — GA 10-100 x faster for CMSSM)

For this study use pMSSM, 23 parameters:

(Berger, Gainer, Hewett, Rizzo; Abdussalam, Allanach, Quevedo, Feroz, Hobson; Cahill-Rowley, Hewett, Ismail, Rizzo)

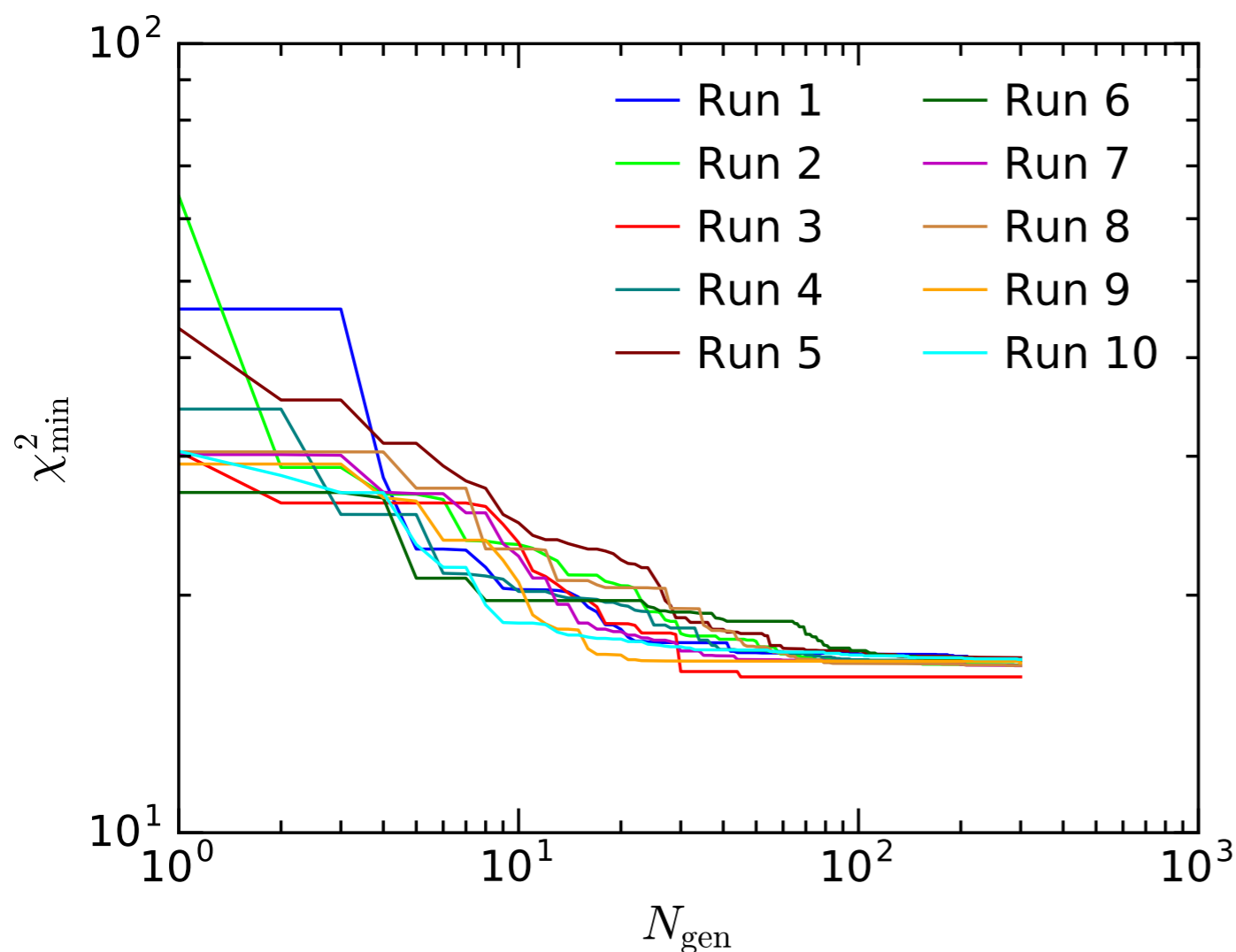
Observable	Value
$[\alpha_{EM}(M_Z)^{\overline{MS}}]^{-1}$	127.950 ± 0.017
$\alpha_S(M_Z)^{\overline{MS}}$	0.1185 ± 0.0006
$m_b(\text{GeV})$	4.78 ± 0.06
$m_t(\text{GeV})$	173.1 ± 0.6

Parameter	Range
SM	
$[\alpha_{EM}(M_Z)^{\overline{MS}}]^{-1}$	[127.882, 128.018]
$\alpha_S(M_Z)^{\overline{MS}}$	[0.1161, 0.1209]
$m_b(\text{GeV})$	[4.54, 5.02]
$m_t(\text{GeV})$	[170.1, 175.5]
pMSSM (GUT scale)	
$M_1, M_2, M_3(\text{GeV})$	[50,10000]
$m_{H_u}, m_{H_d}(\text{GeV})$	[50,10000]
$m_{\tilde{Q}_{1,2}}, m_{\tilde{Q}_3}(\text{GeV})$	[50,10000]
$m_{\tilde{U}_{1,2}}, m_{\tilde{U}_3}(\text{GeV})$	[50,10000]
$m_{\tilde{D}_{1,2}}, m_{\tilde{D}_3}(\text{GeV})$	[50,10000]
$m_{\tilde{L}_{1,2}}, m_{\tilde{L}_3}(\text{GeV})$	[50,10000]
$m_{\tilde{E}_{1,2}}, m_{\tilde{E}_3}(\text{GeV})$	[50,10000]
$A_t, A_b, A_\tau(\text{TeV})$	[-10,10]
$\tan \beta$	[2,62]

Fitness function is simply 1/likelihood derived from all experimental constraints: it singles out (g-2) of the muon as the offending observable.

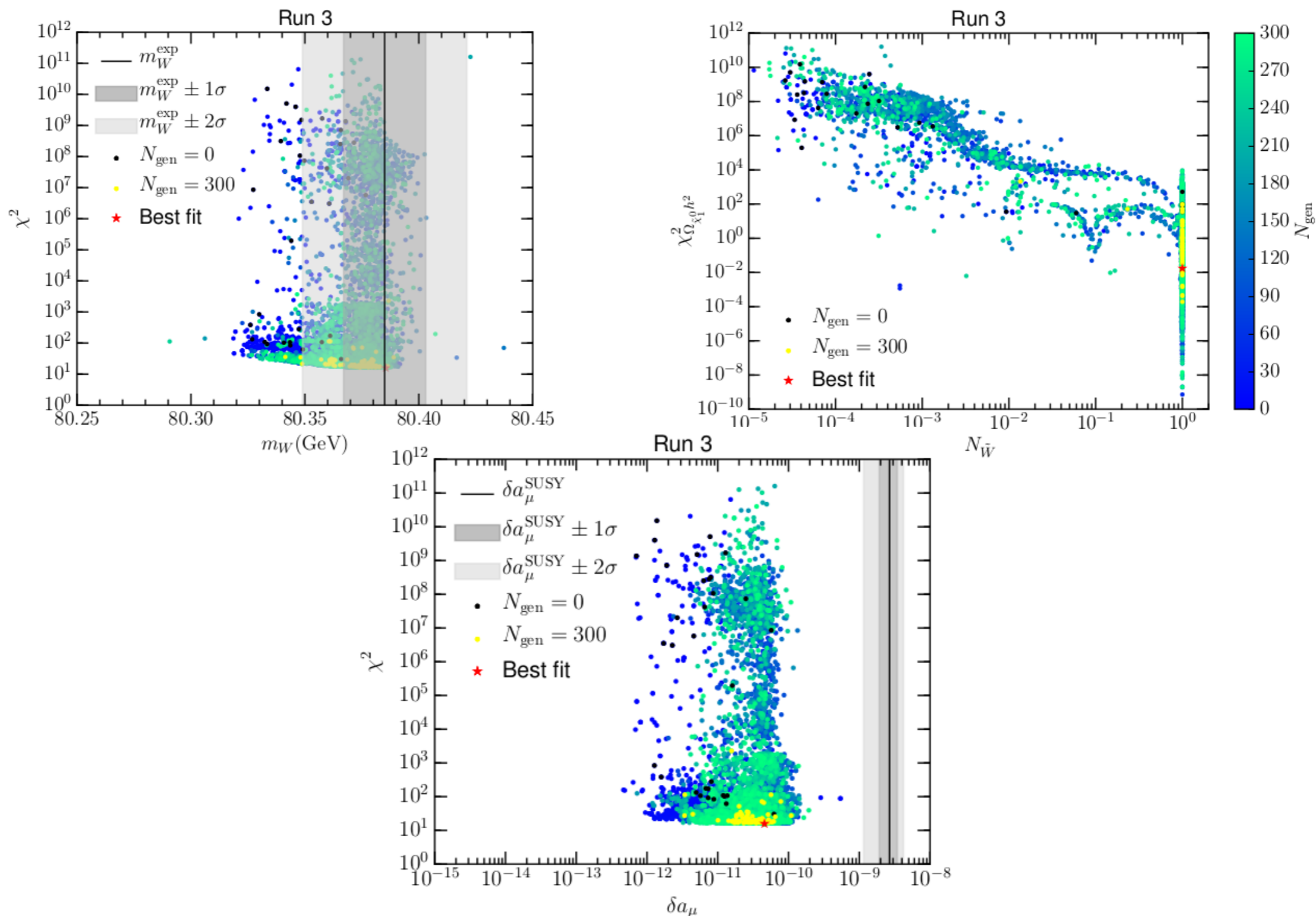
$$\ln \mathcal{L}_{\text{Joint}} = \ln \mathcal{L}_{\text{EWPO}} + \ln \mathcal{L}_B + \ln \mathcal{L}_{\text{Higgs}} + \ln \mathcal{L}_{\text{LEP}} + \ln \mathcal{L}_{\text{LHC}} + \ln \mathcal{L}_{\Omega_{\text{DM}} h^2} + \ln \mathcal{L}_{\delta a_{\mu}^{\text{SUSY}}}$$

Used the following: PIKAIA2.0 (Metcalf+Charbonneau), SoftSUSY, FeynHiggs, ZFITTER, MicrOMEGAS, HiggsSignals, PYTHIA, SModelS, NLL-Fast, Fastlim.

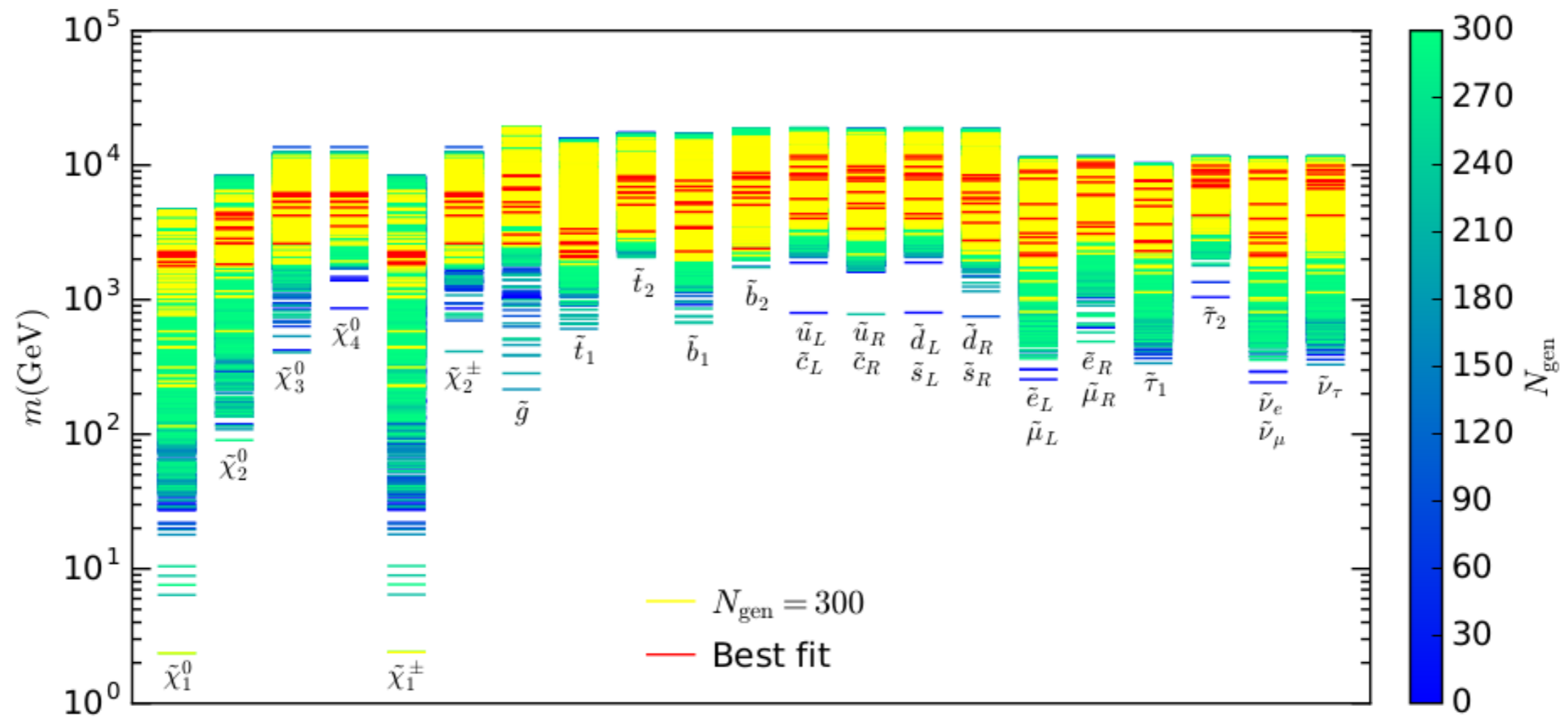


	Run 1
$\chi_{\Omega_{\tilde{\chi}_1^0} h^2}^2$	0.0067
$\chi_{\text{HiggsSignals}}^2$	1.2950
$\chi_{m_{h^0}}^2$	0.1125
$\chi_{M_W}^2$	0.1190
$\chi_{\sin^2 \theta_{\text{eff}}^{\text{lept}}}^2$	0.1538
$\chi_{\Gamma_Z}^2$	0.0332
$\chi_{\Gamma_Z^{\text{inv}}}^2$	2.3054
$\chi_{BR(B \rightarrow X_s \gamma)}^2$	0.0664
$\chi_{BR(B_s^0 \rightarrow \mu^+ \mu^-)}^2$	0.1647
$\chi_{\frac{BR(B_u \rightarrow \tau \nu)}{BR(B_u \rightarrow \tau \nu)_{\text{SM}}}}^2$	0.0140
χ_{LEP}^2	0.0000
χ_{LHC}^2	0.0000
$\chi_{\delta a_{\mu}^{\text{SUSY}}}^2$	12.2691
χ_{tot}^2	16.5398

Information about the structure can be inferred from the “flow” (assuming fitness distance correlation). e.g. the W mass is easy to fit and not constraining, DM is hard and constraining, g-2 is impossible.

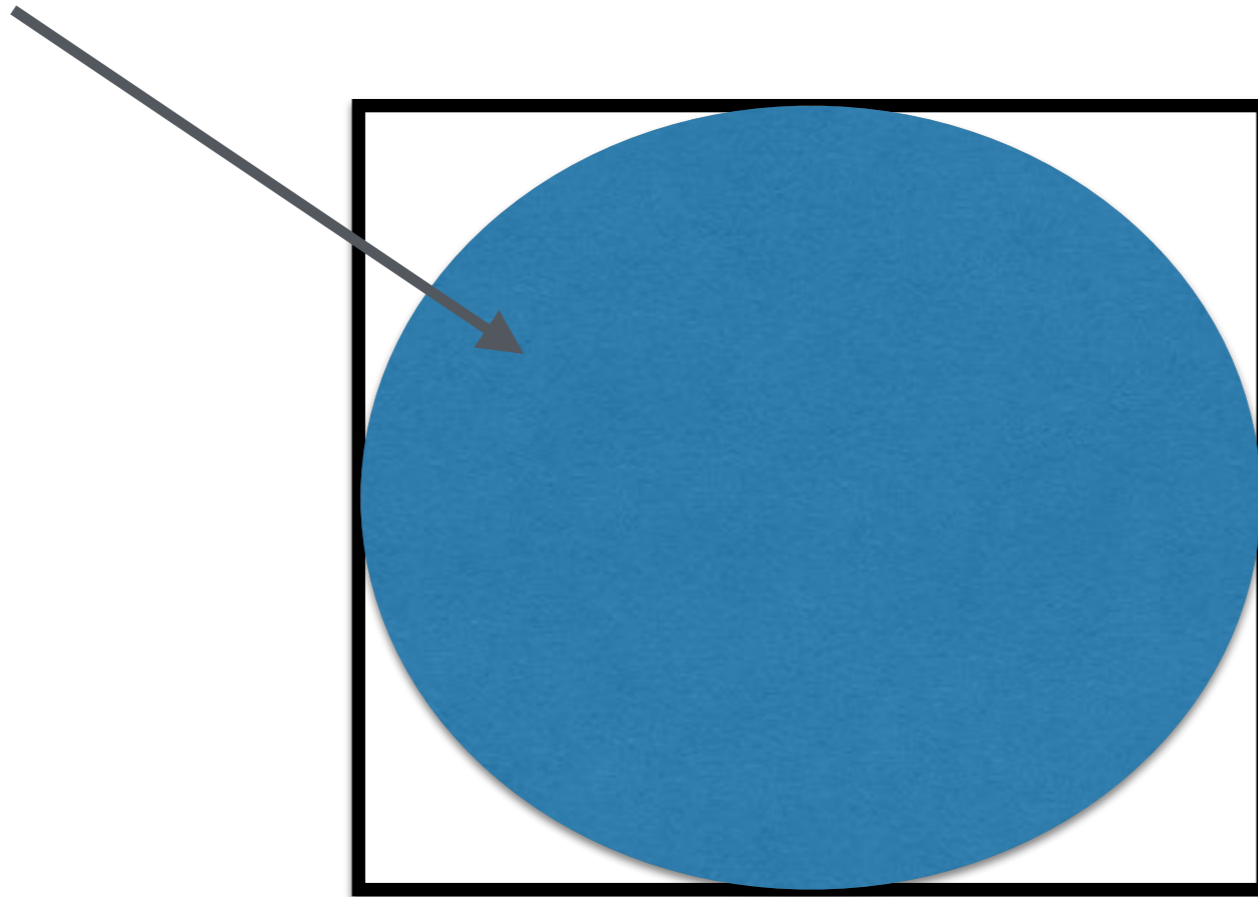


You can get “predictions” from the final generations. e.g. in this case the spectrum:

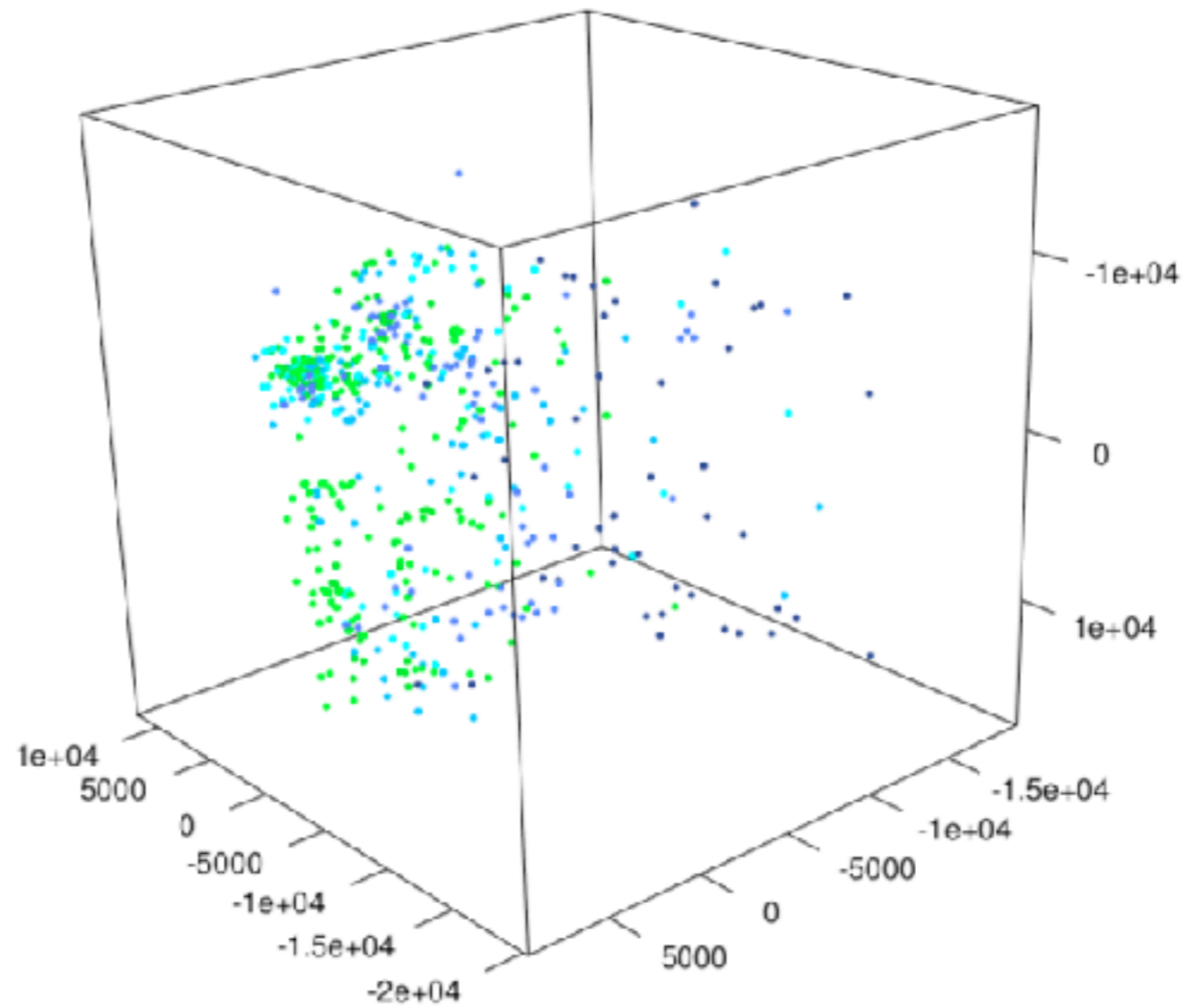


Note the “large dimensionality problem”: in 19 dimensions, slices give a misleading representation of the structure

In 19D this ball occupies only 10^{-7} of the volume of the cube!



Slices give a good idea of the *flow*, but non-linear (Sammon) mapping gives a better image of the *clustering*:



Summary

- GAs are one of the most effective (meta)heuristic search methods
- They excel at problems in difficult search spaces where there are many local optima
- They have three components: selection, breeding, mutation
- Their final convergence tends to slow down — in combination with other heuristic methods they can be even more effective.
- They have probably still untapped potential in BSM physics

First GA tutorial problem:

Tutor: Thomas Harvey

- **Minimising a continuous function**
- **Knapsack problem**
- ...

We have provided an example Jupyter notebook showcasing the application of GA to taxicab numbers. You can access the notebook at <https://www.tinyurl.com/ga-ox-taxi>. Requires Jupyter-notebook and python. Please refer to this notebook for the problem-solving exercises.



*Lecture 2: (Quantum) Annealing and
Quantum Adiabatic Computing*

Overview

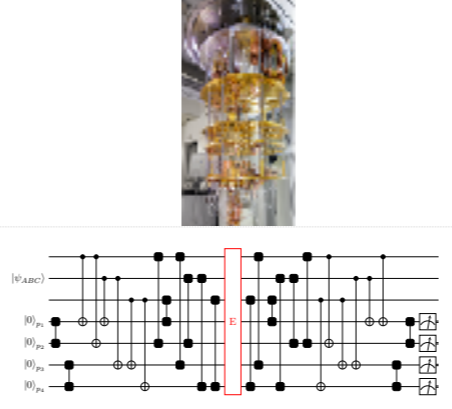
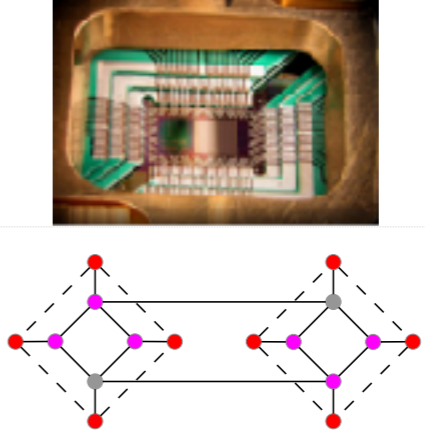


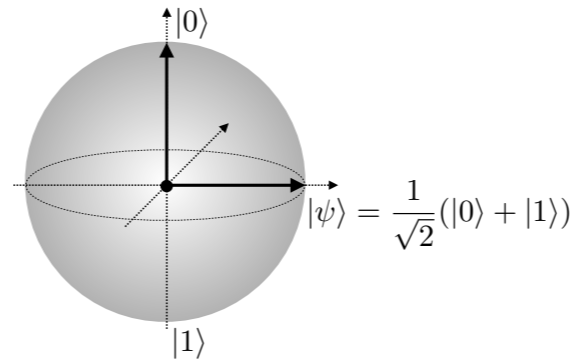
1. Quantum annealers background
2. Reduction
3. Application: Completely Quantum Neural Networks
4. Continuous parameters: simulating QFT
5. Adiabatic Quantum Computing Versus Quantum Annealers and QIBO

Background



Background: Quantum computing has a long and distinguished history but is only now becoming practicable. (Feynman '81, Zalka '96, Jordan, Lee, Preskill ... see Preskill 1811.10085 for review). Two types of Quantum Computer:

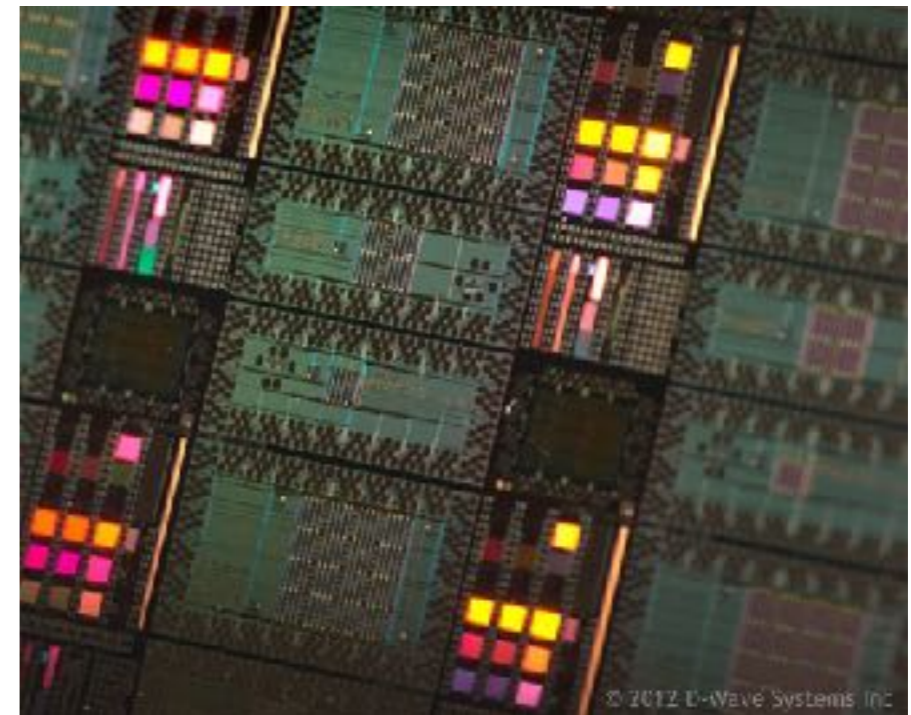
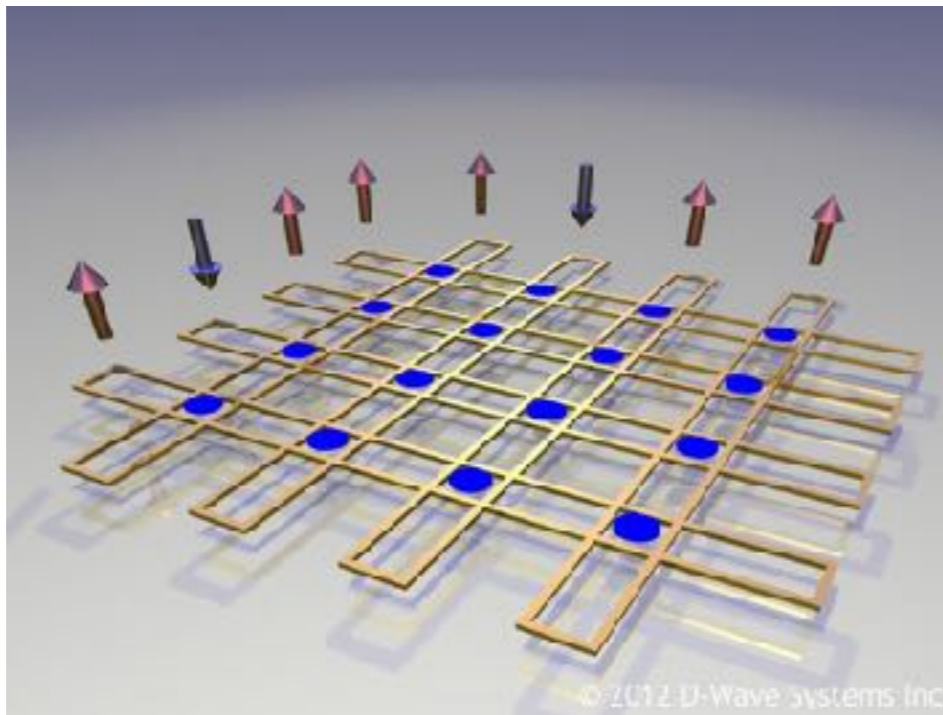
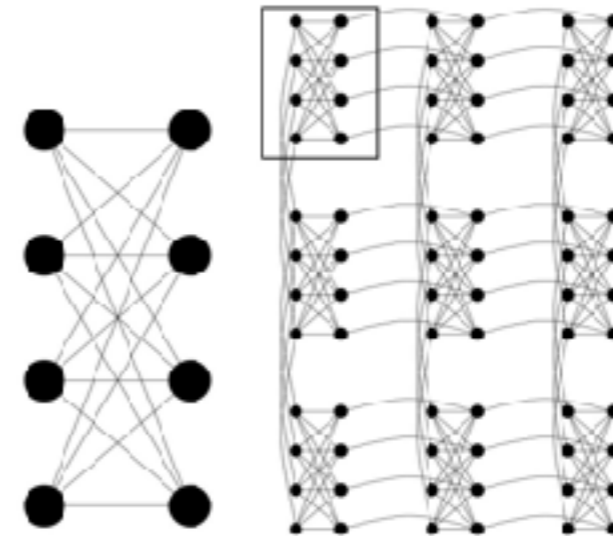
Type	Discrete Gate	Quantum Annealer
Property	Universal (any quantum algorithm can be expressed)	Not universal — certain quantum systems
How?	IBM - Qiskit ~50 Qubits	DWave - LEAP ~5000 Qubits
What?		



- Both types operate on the Bloch sphere: basically measuring $\sigma_i^Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ where $(\sigma^Z|0\rangle = |0\rangle, \sigma^Z|1\rangle = -|1\rangle)$ are the possible eigenvector eqns
- Each i represents a single qubit
- A discrete quantum gate system is good for looking at things like entanglement, Bell's inequality etc. Also discrete problems, cryptographical problems, Shor's, Grover's algorithms, etc.
- A quantum annealer is good for looking at network problems but from our perspective it is also a more natural tool for thinking about field theory. It is based on the general transverse field Ising model (Kadowaki, Nishimori):

$$\mathcal{H}_{\text{QA}}(t) = \sum_i \sum_j J_{ij} \sigma_i^Z \sigma_j^Z + \sum_i h_i \sigma_i^Z + \Delta(t) \sum_i \sigma_i^X$$

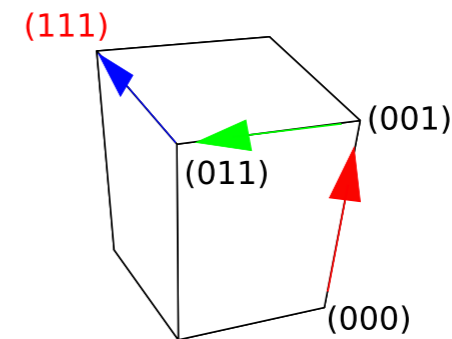
- It looks like this with (restricted couplings - DWave 2000Q)...



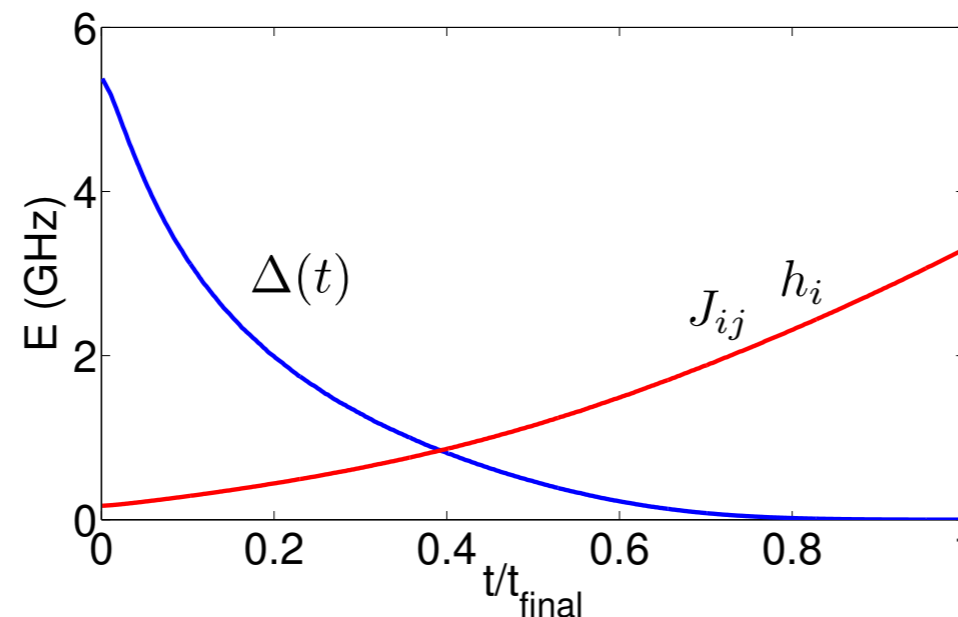
- What does the “anneal” mean?

$$\mathcal{H}_{QA}(t) = \sum_i \sum_j J_{ij} \sigma_i^Z \sigma_j^Z + \sum_i h_i \sigma_i^Z + \Delta(t) \sum_i \sigma_i^X$$

$\Delta(t)$ induces bit-hopping in the Hamming/Hilbert space



The idea is to dial this parameter to land in the global minimum (i.e. the solution) of some “problem space” described by J, h :

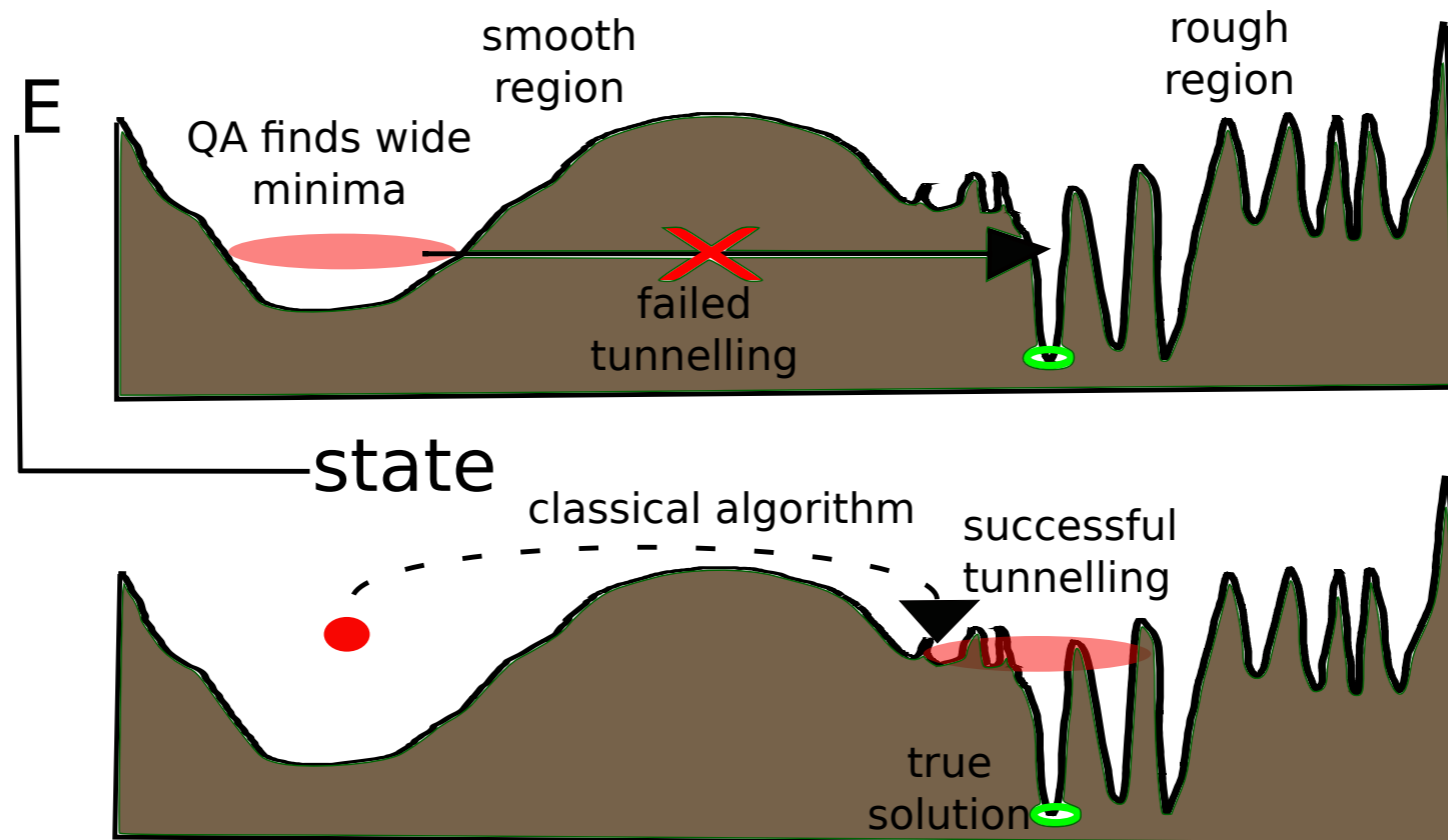


Thermal (classical) and Quantum Annealing are complementary:

More specifically: thermal annealing uses Metropolis algorithm: accept random σ_i^Z flips with probability

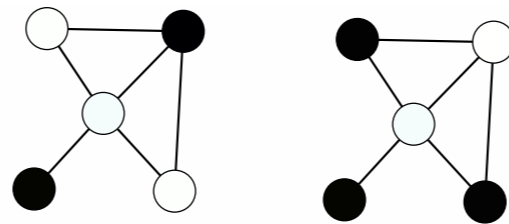
$$P = \begin{cases} 1 & \Delta H \leq 0 \\ e^{-\Delta H/KT} & \Delta H > 0 \end{cases}$$

Quantum tunnelling in QFT happens with probability $P \sim e^{-w\sqrt{2m\Delta H}/\hbar}$ so by contrast it can be operative for tall barriers if they are made thin



Encoding network problems in a general Ising model

- Example 1: how many vertices on a graph can we colour so that none touch? NP-hard problem.



- Let non-coloured vertices have $\sigma_i^Z = -1$ and coloured ones have $\sigma_i^Z = +1$.
- Add a reward for every coloured vertex, and for each link between vertices i, j we add a penalty if there are two +1 eigenvalues:

$$\mathcal{H} = -\Lambda \sum_i \sigma_i^Z + \sum_{\text{linked pairs } \{i,j\}} [\sigma_i^Z + \sigma_j^Z + \sigma_i^Z \sigma_j^Z]$$

- Example 2: N^2 students are to sit an exam in a square room with $N \times N$ desks 1.5m apart. half the students (A) have a virus while half of them (B) do not. How can they be arranged to minimise the number of ill students that are less than 2m from healthy students?
- Call the eigenvalue of A == +1 and that of B == -1. That is if I measure σ^Z at a point to have value +1 then I conclude that I should put an ill person there, and vice-versa.
- There are N^2 spins $\sigma_{\ell N+j}^Z$ arranged in rows and columns. I do not care if $A \geq \langle A \rangle$ or $B \geq \langle B \rangle$, but if $A \geq \langle B \rangle$ then I put a penalty of +2 on the Hamiltonian (ferromagnetic coupling). So ...

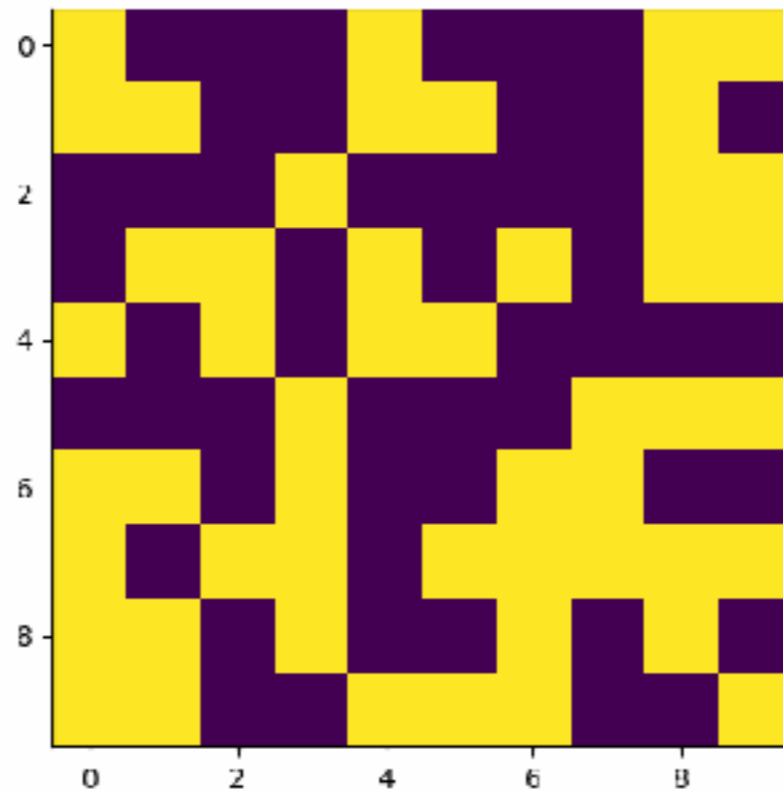
$$\mathcal{H} = \sum_{\ell m=1}^N \sum_{ij=1}^N (\delta_{\ell m} (\delta_{(i+1)j} + \delta_{(i-1)j}) + \delta_{ij} (\delta_{(\ell+1)m} + \delta_{(\ell-1)m})) [1 - \sigma_{\ell N+i}^Z \sigma_{m N+j}^Z]$$

- Finally I need to apply the constraint that #A = #B: $\mathcal{H}^{(\text{constr})} = \Lambda (\#A - \#B)^2$
- $$= \Lambda \left(\sum_{\ell, i}^N \sigma_{\ell N+i}^Z \right)^2$$
- $$= \Lambda \sum_{\ell m=1}^N \sum_{ij=1}^N \sigma_{\ell N+i}^Z \sigma_{m N+j}^Z$$

- Example 2 done with classical thermal annealing using the Metropolis algorithm. Note this represents a search over ${}_{100}C_{50} \sim 2^{100}$ configurations:

***System is
ferromagnetic with
constraint ...***

***$2^{100} \sim 10^{30}$
possibilities!***



- Importantly the constraint hamiltonian cannot be too big otherwise the hills are too high and it freezes too early. This makes the process require a (polynomial sized) bit of “thermal tuning”.

- In the metropolis algorithm the cooling time can increase exponentially with the size
- This can be done in microseconds on a quantum annealer
- To do this we simply fill h and J and call the quantum annealer from python
- However the architecture (connectivity of J,h) is limited, so it needs to be “embedded”
- “answer” is a list of [+1,-1,+1,+1] spins ordered by energy:

```

target_graph = DWaveSampler(solver={'qpu': True}).edgelist
Q= dimod.BinaryQuadraticModel.from_ising(h, J)
h0 = Q.to_ising()[0]
J0 = Q.to_ising()[1]

Q.to_ising()
embedding = find_embedding(Q.to_ising()[1], target_graph)

sampler_embedded = FixedEmbeddingComposite(DWaveSampler(solver=dict(qpu=True)), embedding)

#embed()
#sys.exit(1) # put this wherever you want to jump out to command line and test functions

try:
    answer = sampler_embedded.sample_ising(h0, J0, chain_strength=4, num_reads = nos_reads, **anneal_params)

except:
    looper = looper - 1
    continue

# inspector.show(answer) # to inspect the embedding

set_solutions, set_energies = answer.record.sample, answer.record.energy
print("Lowest energy found:", answer.record.energy[0])
print("Average energy is {:.2f} with standard deviation {:.2f}".format(set_energies.mean(), set_energies.std()))

```

TaxiCabs and reduction



Encoding the problem:

Suppose we want to solve the TaxiCab problem — we could contemplate doing

$$\tilde{H} = H_D + H_C$$

where $H_D = (a^3 + b^3 - c^3 - d^3)^2$

and each integer is binary encoded as e.g. $a = \sum_{\ell=0}^{\beta} 2^{\ell} \tau_{\ell}^a$

where $\tau_{\ell} = \frac{1}{2}(\sigma_{\ell} + 1)$ are the binaries from each spin (called the QUBO Hamiltonian).

The numbers should of course not repeat so $H_C \equiv H_{\delta}(a, c) + H_{\delta}(a, d)$

$$\begin{aligned} &\equiv \prod_{k=0}^{\beta-1} (1 - (\tau_{a,k} - \tau_{c,k})^2) \\ &\quad + \prod_{k=0}^{\beta-1} (1 - (\tau_{a,k} - \tau_{d,k})^2) \end{aligned}$$

Reduction:

But imagine substituting all these integers into the H in the loss function, we get a gigantic sextic in sigma's, and the Ising model can only be quadratic. We can get around this by adding to the Hamiltonian auxiliary (ancilla) spins that represent pairs of spins.

Thus consider the Hamiltonian $Q(\tau_{12}; \tau_1, \tau_2) = \Lambda(\tau_1\tau_2 - 2\tau_{12}(\tau_1 + \tau_2) + 3\tau_{12})$

$$\tau_1 = 0, \tau_2 = 0, \implies Q = 3\Lambda\tau_{12} \implies \tau_{12} = 0$$

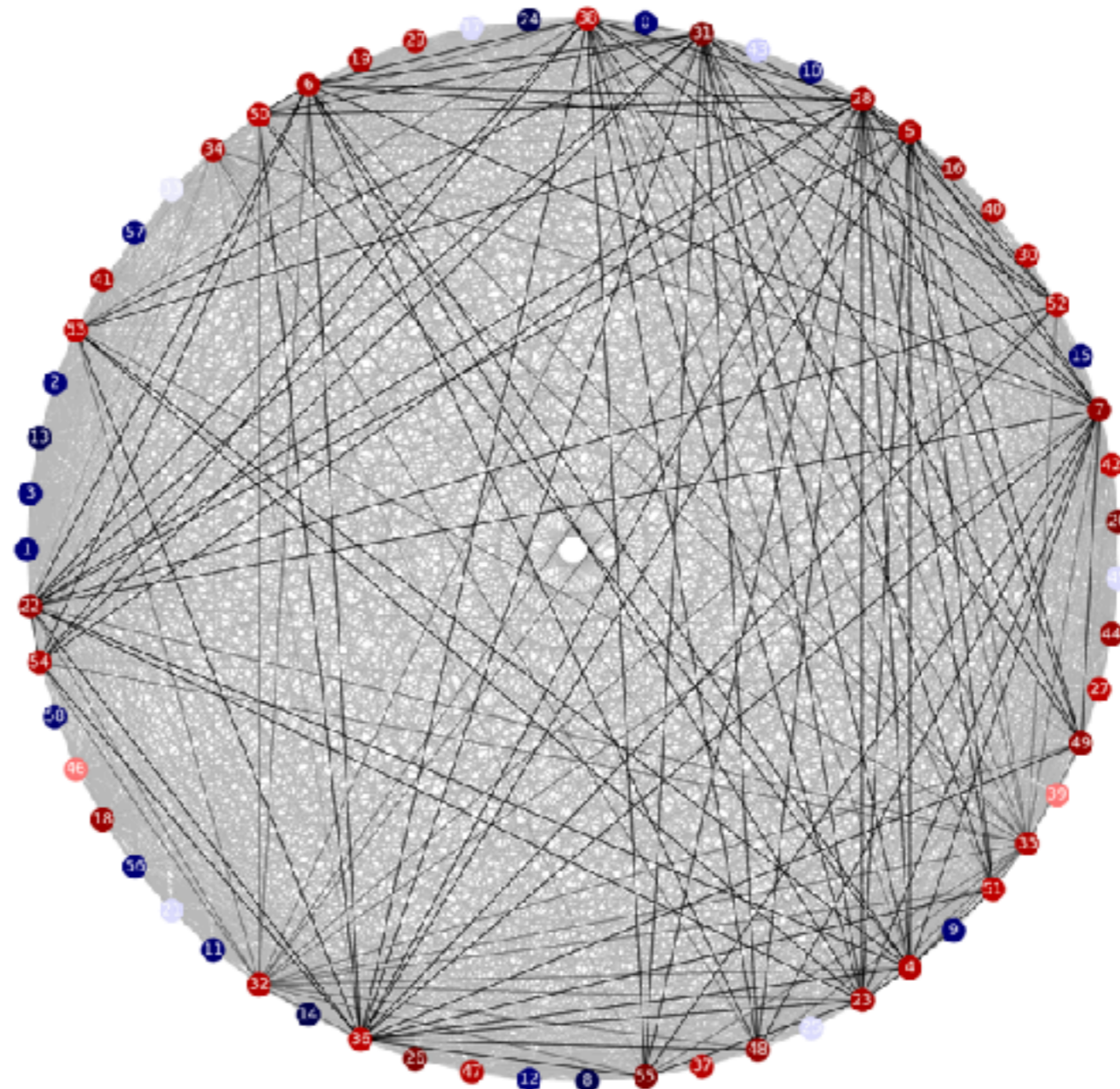
$$\tau_1 = 1, \tau_2 = 0, \implies Q = \Lambda\tau_{12} \implies \tau_{12} = 0$$

$$\tau_1 = 0, \tau_2 = 1, \implies Q = \Lambda\tau_{12} \implies \tau_{12} = 0$$

$$\tau_1 = 1, \tau_2 = 1, \implies Q = \Lambda(1 - \tau_{12}) \implies \tau_{12} = 1$$

It has a minimum at $Q=0$ where $\tau_{12} = \tau_1\tau_2$, so we can replace the pair $\tau_1\tau_2$ with a single qubit τ_{12} everywhere in the rest of the Hamiltonian, reducing the degree by one, as long as we add Q to H , the H 's are guaranteed to have the same vacuum.

- Thus we go through reducing the sextic down to a quartic and adding a Q for every pair: repeat eventually reducing to a quadratic (happily this can be automated):
- The Ising model looks kind of ugly ...



- Thus we go through reducing the sextic down to a quartic and adding a Q for every pair: repeat eventually reducing to a quadratic (happily this can be automated):

Nevertheless — Results:

- Find the early TaxiCab numbers. Plus e.g. equal sums of 3 quartics ...

$(4, 3, 3)$	a	b	c	d	e	f
2673	3	6	6	7	2	4
16562	9	1	10	11	6	5
28593	2	13	2	9	6	12
35378	13	4	9	11	12	1
43218	11	13	2	14	7	7
54977	4	8	15	9	14	10
195122	21	5	2	9	13	20
324818	14	9	23	21	2	19
619337	28	8	5	7	26	20
847602	1	25	26	29	19	10
1071713	12	32	7	28	26	3
1178898	29	11	26	1	32	19
1328498	29	9	28	23	32	3

- Thus we go through reducing the sextic down to a quartic and adding a Q for every pair: repeat eventually reducing to a quadratic (happily this can be automated):

Nevertheless — Results:

- Find the early TaxiCab numbers. Equal sums of 8 cubics ...

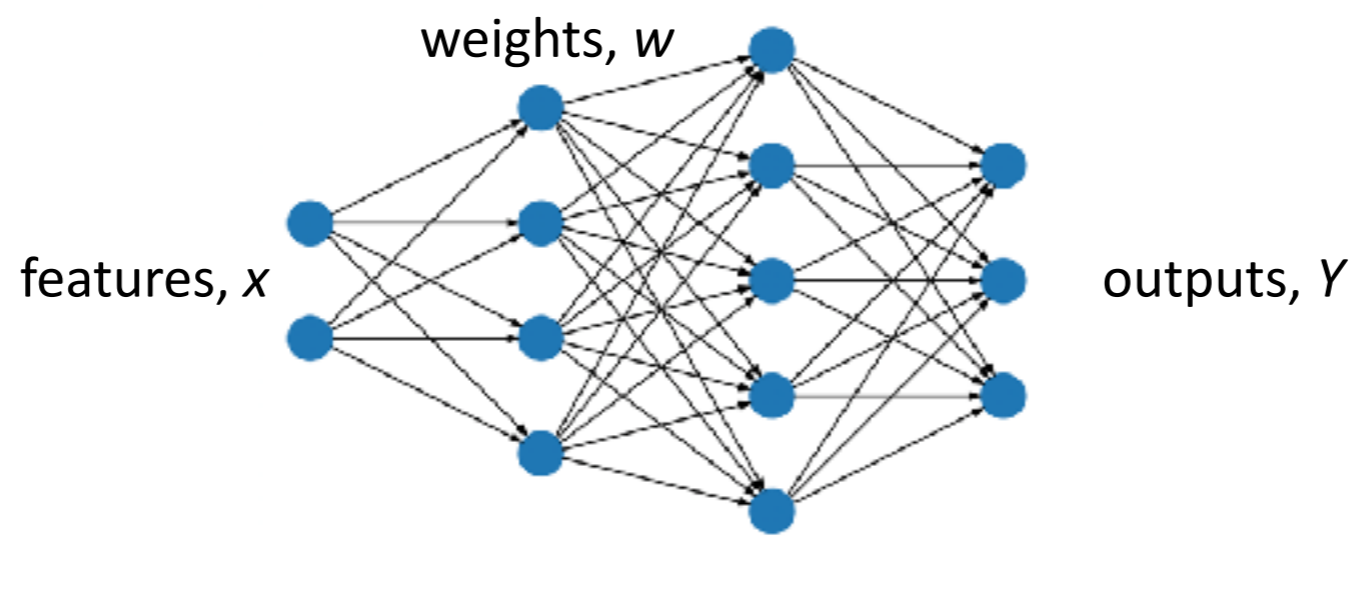
$(3, 8, 8)$	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
50139	1	3	6	10	12	20	23	30	2	5	9	13	17	19	25	27
73206	1	3	4	17	20	25	26	30	5	8	9	10	19	21	28	32
78202	3	4	17	18	19	24	27	30	1	2	9	16	20	22	28	32
85418	2	3	9	16	18	23	31	32	6	10	14	15	24	26	27	30

- smallest of these is a solution in a search space of 10^{24} .

***Application: Completely Quantum
Neural Networks***



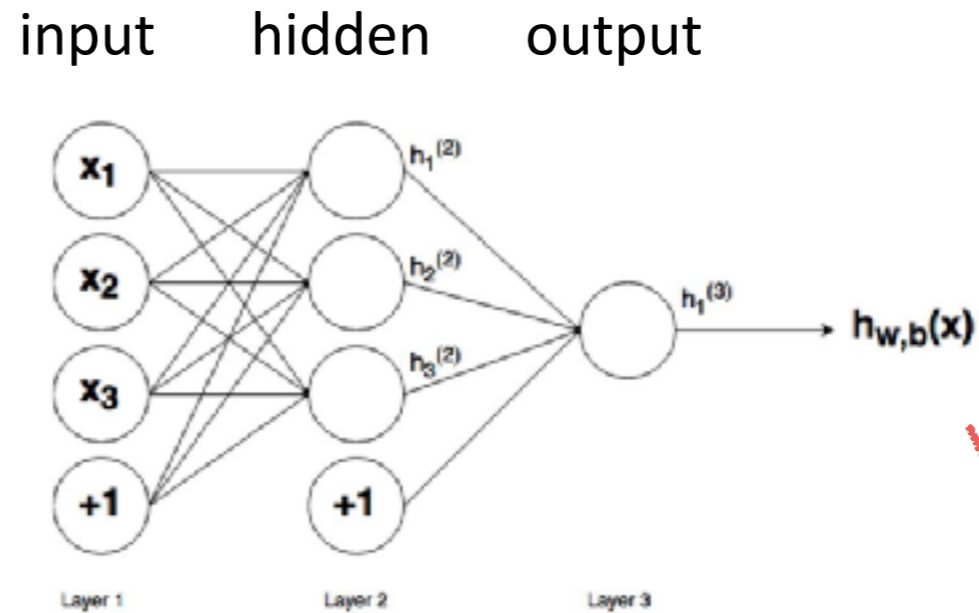
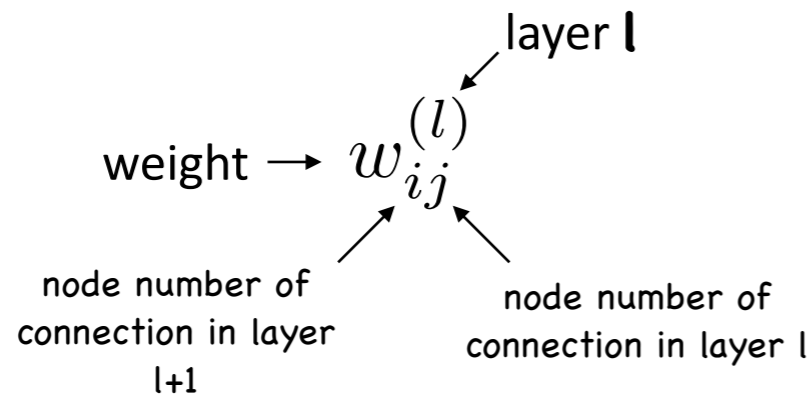
Recap of classical NNs: the AI in your phone consists of a NN that encodes the solution to a class of problems in weights and biases:



NN produces outputs Y by passing inputs x through layers with activation functions g as follows:

$$L_i(x) = g \left(\sum_j w_{ij} x_j + b_i \right) \quad Y = L^{(n)} \circ \dots \circ L^{(0)}$$

In detail it is put together like this ...



bias of layer l is connected to all nodes in layer $l+1$, thus $b_i^{(l)}$

$b_i^{(l)}$ and $w_{ij}^{(l)}$ have to be calculated for a given problem during the learning phase of the NN

activation function

$$h_1^{(2)} = g(w_{11}^{(1)}x_1 + w_{12}^{(1)}x_2 + w_{13}^{(1)}x_3 + b_1^{(1)})$$

$$h_2^{(2)} = g(w_{21}^{(1)}x_1 + w_{22}^{(1)}x_2 + w_{23}^{(1)}x_3 + b_2^{(1)})$$

$$h_3^{(2)} = g(w_{31}^{(1)}x_1 + w_{32}^{(1)}x_2 + w_{33}^{(1)}x_3 + b_3^{(1)})$$

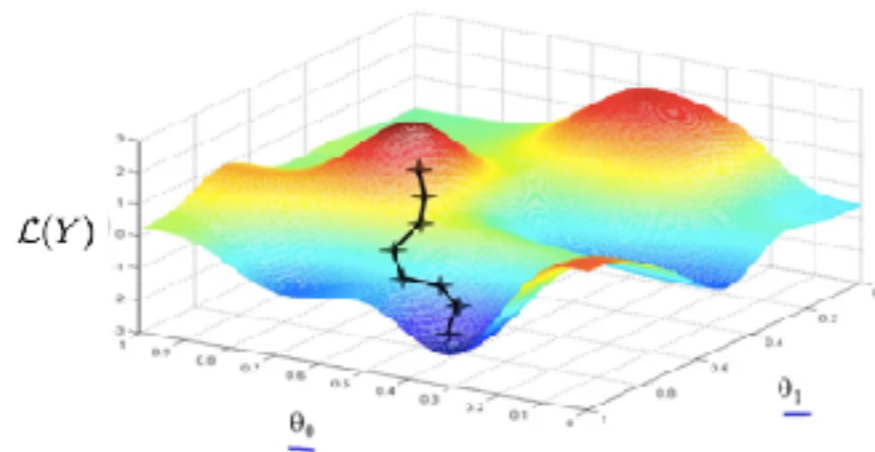
$$h_{w,b}(x) = h_1^{(3)} = g(w_{11}^{(2)}h_1^{(2)} + w_{12}^{(2)}h_2^{(2)} + w_{13}^{(2)}h_3^{(2)} + b_1^{(2)})$$

The feedforward pass:

To make the network learn (in a supervised way), we define a loss function that we minimise for a whole load of previous data to determine all the weights and biases (e.g. for classification with data labelled data by a):

$$\mathcal{L}(Y) = \frac{1}{N_d} \sum_a |y_a - Y(x_a)|^2$$

Classically: minimise \mathcal{L} using gradient descent and backpropagation



The loss function establishes a hypersurface for which we can try to find a minimum using gradient descent

Gradient descent for every weight $w_{ij}^{(l)}$ and every bias $b_i^{(l)}$ in the NN looks like:

$$w_{ij}^{(l)} = w_{ij}^{(l)} - \alpha \frac{\partial}{\partial w_{ij}^{(l)}} \mathcal{L}(w, b)$$

$$b_i^{(l)} = b_i^{(l)} - \alpha \frac{\partial}{\partial b_i^{(l)}} \mathcal{L}(w, b)$$

in short: $w_{new} = w_{old} - \alpha * \nabla error$

where α is the learning rate

Quantum training of NNs: The training process can be one of the lengthiest parts of the process: can we use a quantum annealer to train? (After all it is built to minimise loss functions.)

Three problems must be solved to do this:

If we think about $\mathcal{L}(Y) = \frac{1}{N_d} \sum_a |y_a - Y(x_a)|^2$ we want to avoid having to encode each data point in qubits

We will instead encode the weights and biases in qubits in binary fashion and read off their values. But the resulting Hamiltonian will again be a high order polynomial in sigma's just like before - depending on the number of layers etc. Therefore need reduce the Hamiltonian.

Finally the annealer is not completely connected (couplings are missing). Need to find a minor embedding linking qubits together (libraries available on D-wave but also an NP hard problem in principle).

Example: we took a *single* hidden layer:

$$Y_{v,w}(x_j) = v_i g(w_{ij} x_j) + v_0$$

The activation function must be nonlinear for a NN to work, but it can be simple: $g(x) = (1 + x)^2 / 4$

Then what appears in the loss function $\mathcal{L}(Y) = \frac{1}{N_d} \sum_a |y_a - Y(x_a)|^2$ is

$$Y_{v,w}(x) = y_a - \frac{1}{4} - \frac{1}{2} v_i w_{ij} x_{aj} - \frac{1}{4} v_i w_{ij} w_{ij'} x_{aj} x_{aj'}$$

So we are in the business of encoding something *sextic* in the spins even for this simple activation function, with the weights being encoded as fractional binaries ...

$$\omega = -1 + \frac{1}{1 - 2^{-\beta}} \sum_{\alpha=0}^{\beta} 2^{-\alpha} \tau_{\alpha}^{\omega}$$

e.g. 2D datasets = “circles”, “quadrants”, “bands” and t-tbar yields a classification curve.
(The features for the latter are the highest transverse momentum of a b-jet and the missing energy, in simulated LHC pp collisions.)

Unexpected advantage: our weights and biases are all discretised due to the “qubitisation”. A standard NN cannot be trained very well for discrete weights and biases as it gets stuck.

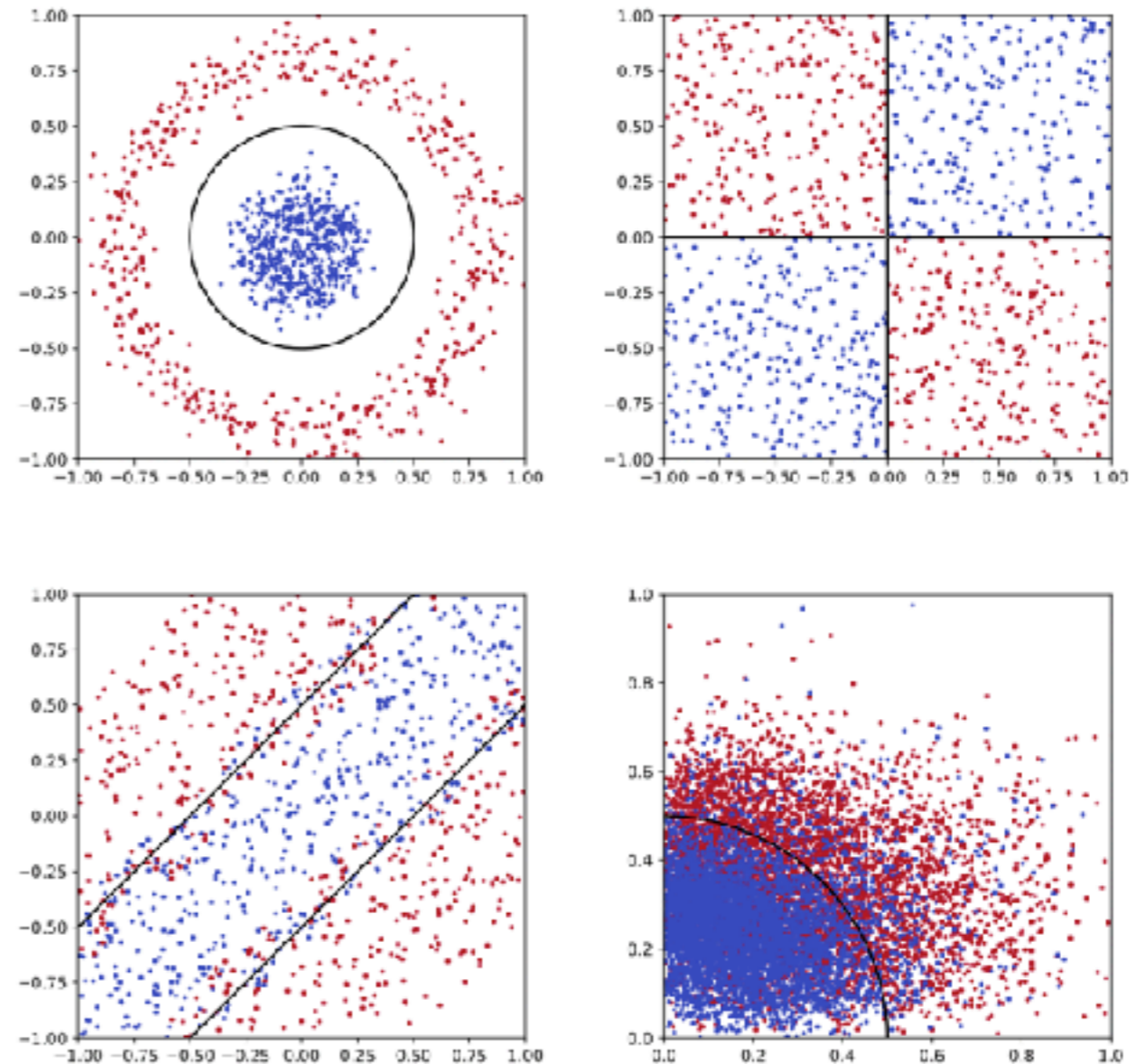


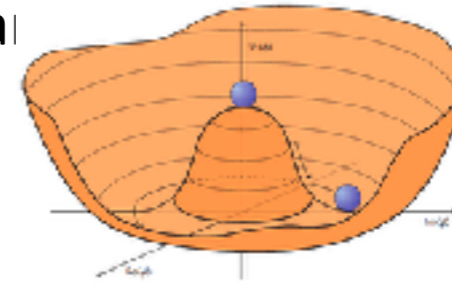
Figure 4: Decision boundary obtained with the quantum NN for each dataset.

***Encoding continuous parameters:
Simulating QFT***

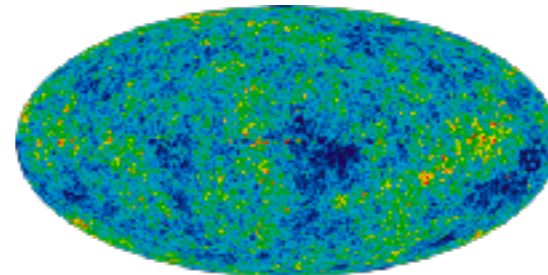


Many interesting QFT problems involve “tunnelling”:

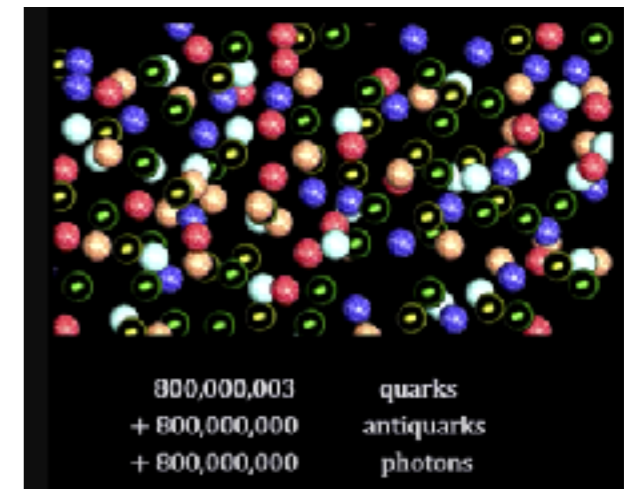
- Electroweak phase transition (Higgs mechanism)



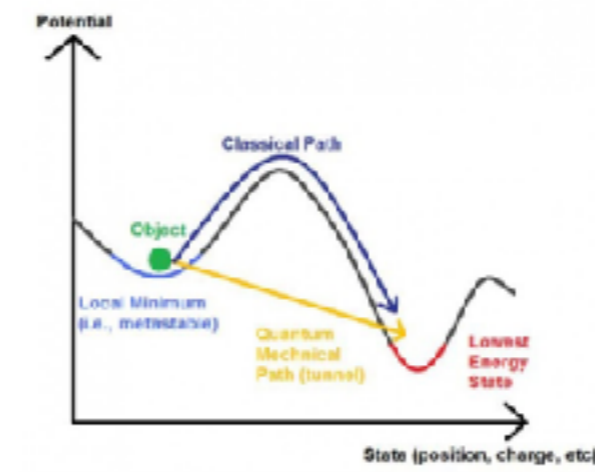
- Inflation



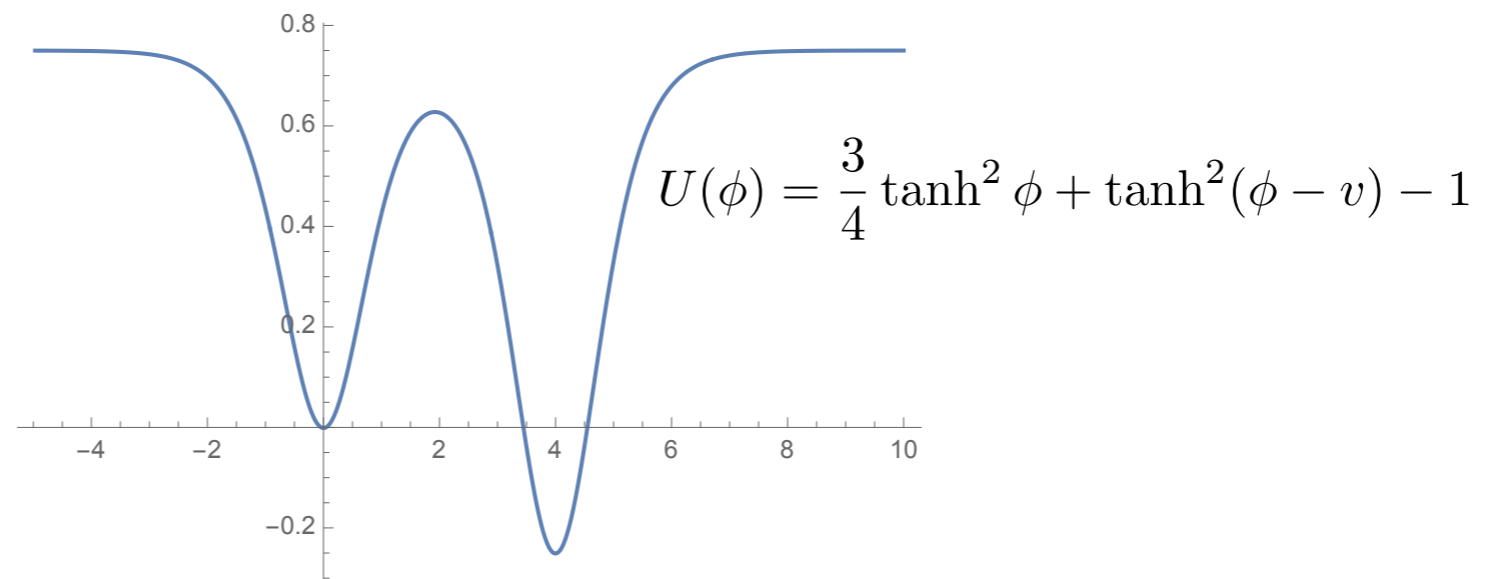
- Baryogenesis (creation of (anti)matter asymmetry)



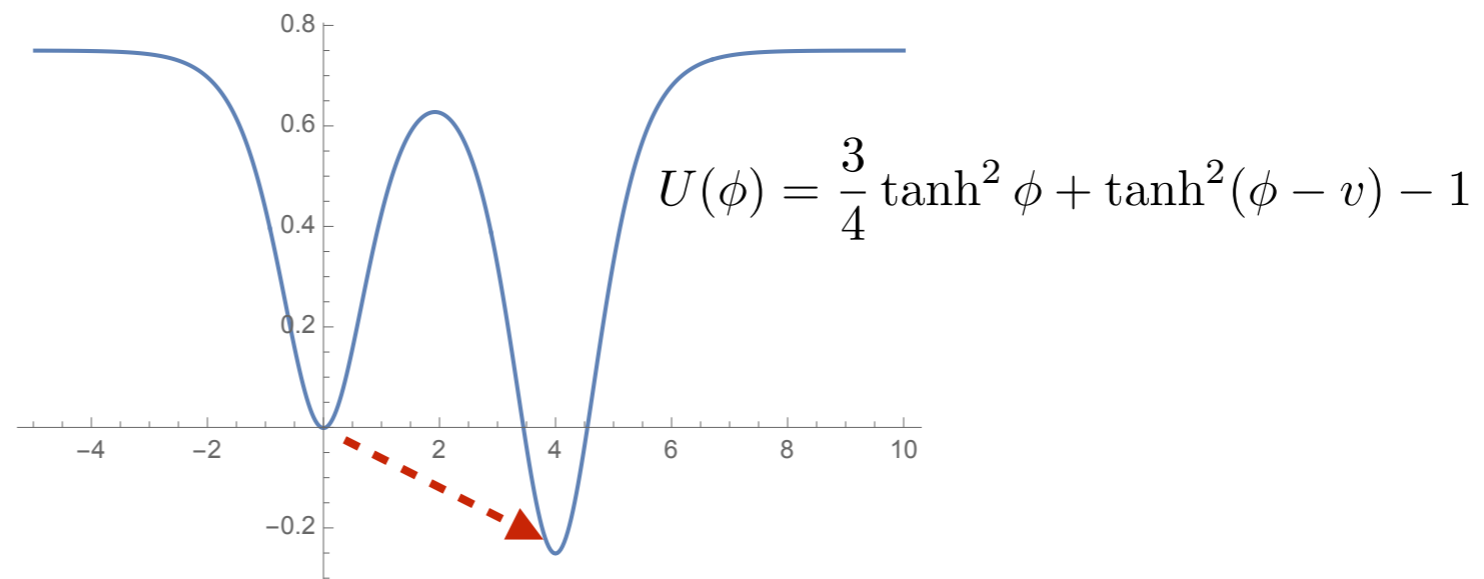
- Instanton processes:



Simple tunnelling problem: *tunnelling out of the false minimum of this potential (where ϕ is the single space coordinate)*



Simple tunnelling problem: *tunnelling out of the false minimum of this potential (where ϕ is the single space coordinate)*



If we begin in the false minimum on the left, the system should be able to tunnel to the lower one on the right.

To encode U , first encode ϕ by discretising its value using N qubits:

$$\phi = \phi_0 + j \xi = \phi_0 + \xi \dots \phi_0 + N\xi$$

Represent it as a point on a spin chain == domain wall encoding (Chancellor):

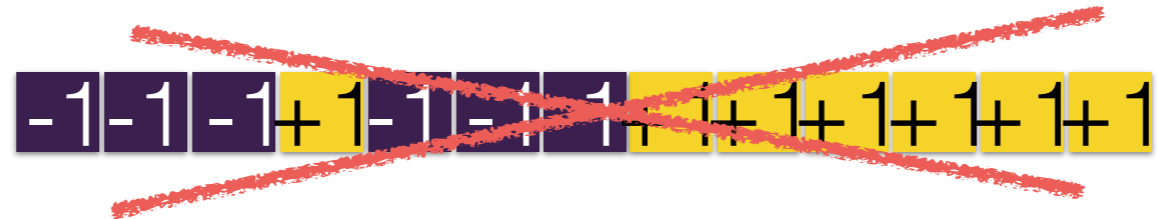


Position j

We can translate any spin chain back to the corresponding field value using

$$\phi = \phi_0 + \frac{\xi}{2} \sum_{i=1}^N (1 - \sigma_i^Z)$$

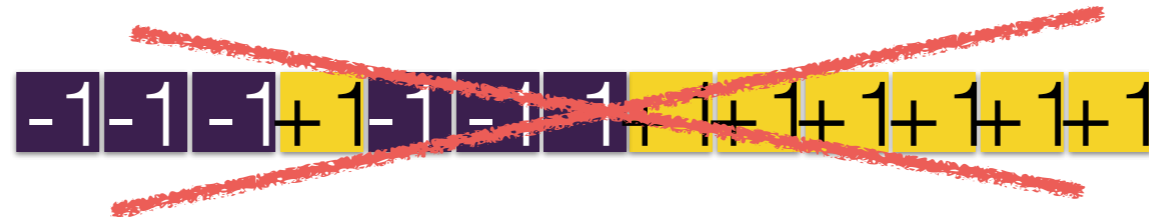
For this to work as a consistent encoding we have to avoid e.g.



This is the domain-wall encoding. Begin in the Ising model with a ferromagnetic interaction that favours as few flips as possible, but frustrate at least one by having the endpoints pinned at -1 ... +1.

$$\mathcal{H}^{(\text{chain})} = \Lambda \left(\sigma_1^Z - \sigma_N^Z - \sum_i^{N-1} \sigma_i^Z \sigma_{i+1}^Z \right)$$

For this to work as a consistent encoding we have to avoid e.g.



This is the domain-wall encoding. Begin in the Ising model with a ferromagnetic interaction that favours as few flips as possible, but frustrate at least one by having the endpoints pinned at $-1 \dots +1$.

$$\mathcal{H}^{(\text{chain})} = \Lambda \left(\sigma_1^Z - \sigma_N^Z - \sum_i^{N-1} \sigma_i^Z \sigma_{i+1}^Z \right)$$



Pins the end spins at opposing values

$$h_j^{(\text{chain})} = \Lambda (\delta_{j1} - \delta_{jN})$$

penalty for different adjacent spin

$$J_{ij}^{(\text{chain})} = -\frac{\Lambda}{2} \begin{pmatrix} 0 & 1 & & & & & \\ 1 & 0 & 1 & & & & \\ & 1 & 0 & & & & \\ & & & \ddots & & & \\ & & & & 0 & 1 & \\ & & & & 1 & 0 & \end{pmatrix}_{ij}$$

To add the potential we then add a contribution to the linear h couplings

-1 -1 -1 -1 -1 -1 -1 + 1 + 1 + 1 + 1 + 1 + 1

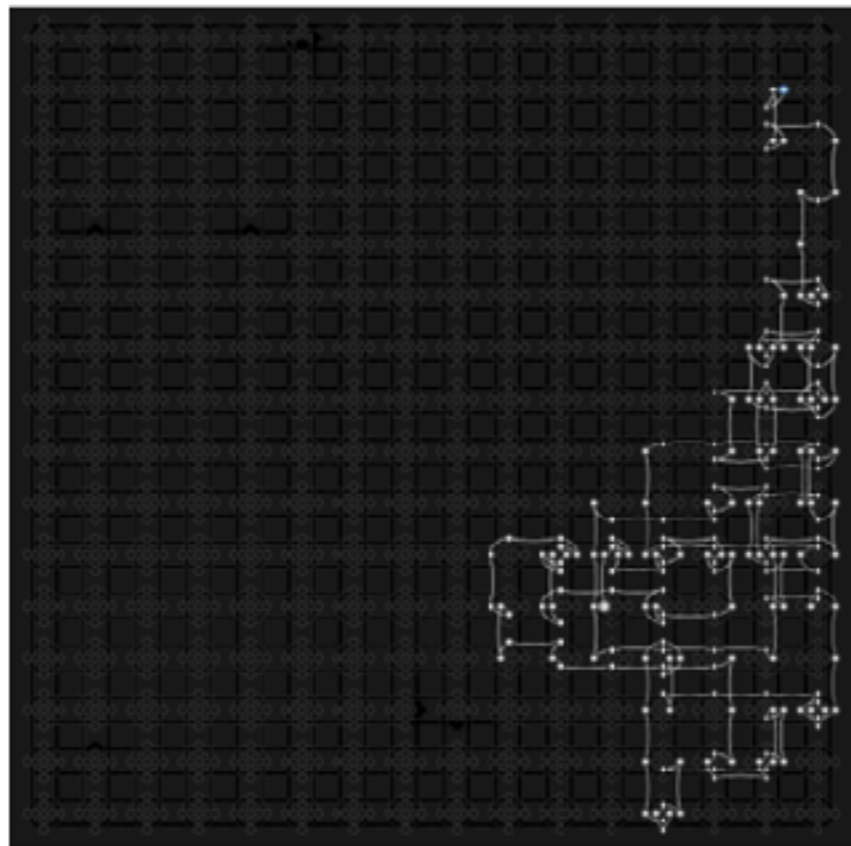
only the
frustrated link
contributes

$$\begin{aligned}
 U(\phi) &= \frac{1}{2} \sum_i^{N-1} U(\phi_0 + j\xi) (\sigma_{j+1}^Z - \sigma_j^Z) \\
 &\equiv -\frac{\xi}{2} \sum_i^{N-1} U'(\phi_0 + j\xi) \sigma_j^Z
 \end{aligned}$$

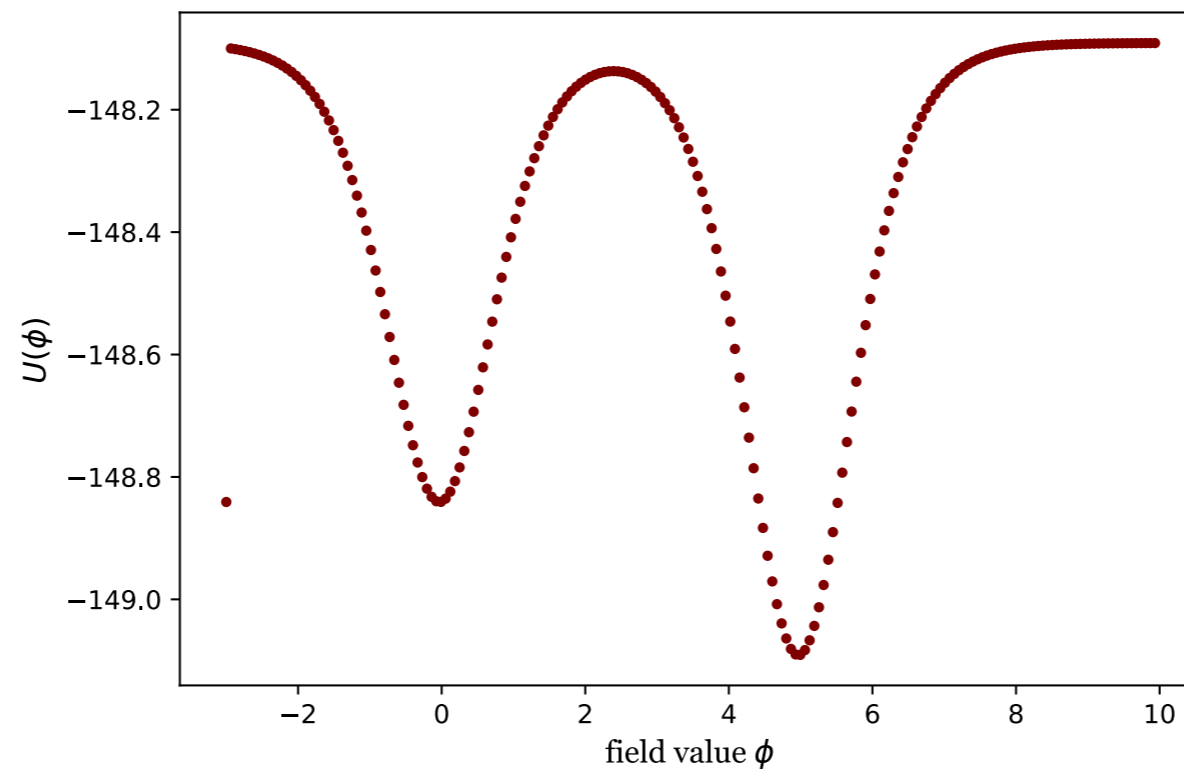
Finally add everything together (use $N=200$ qubits):

$$\mathcal{H} = \mathcal{H}^{\text{chain}} + \mathcal{H}^U$$

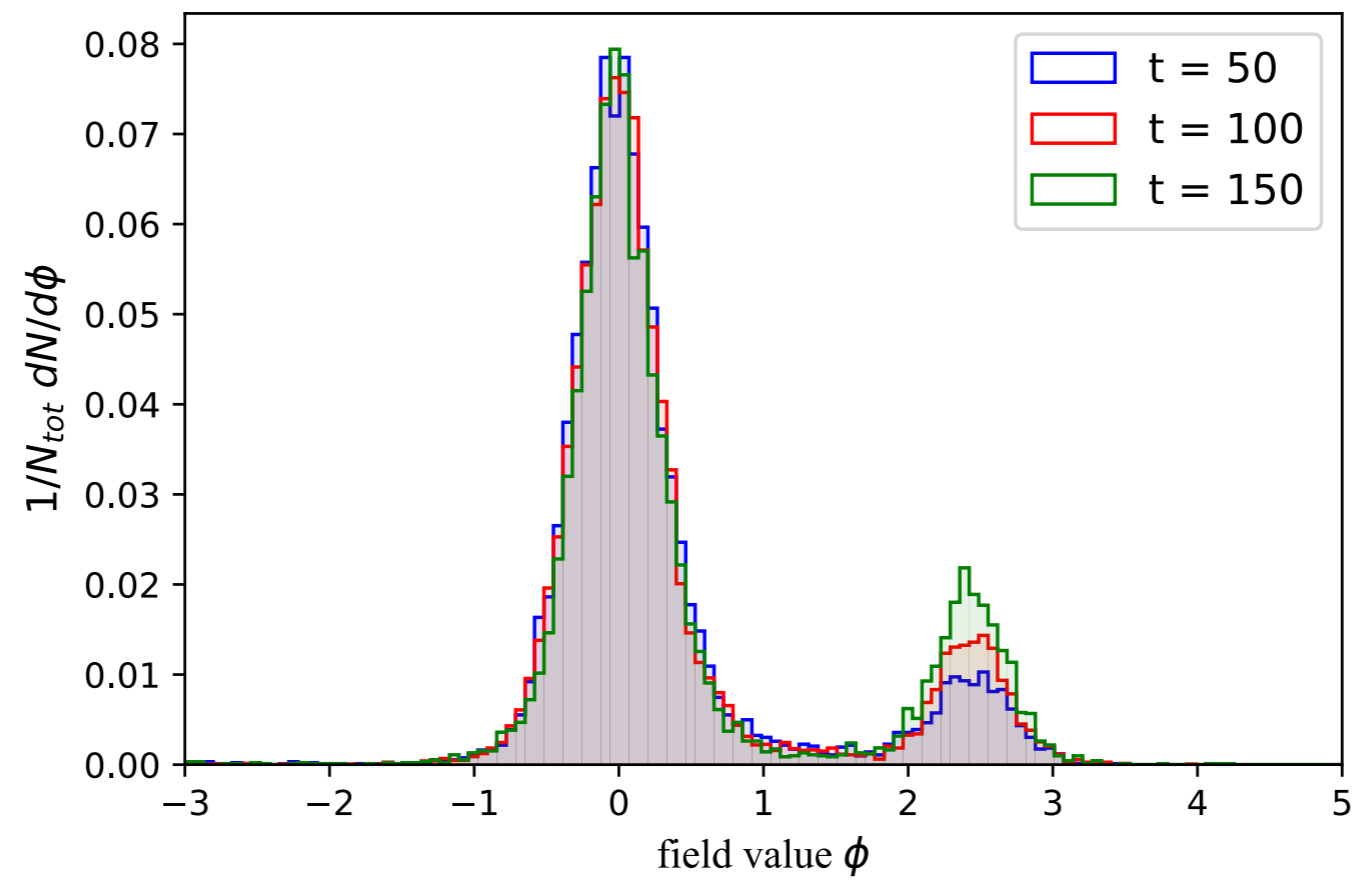
Typical embedd



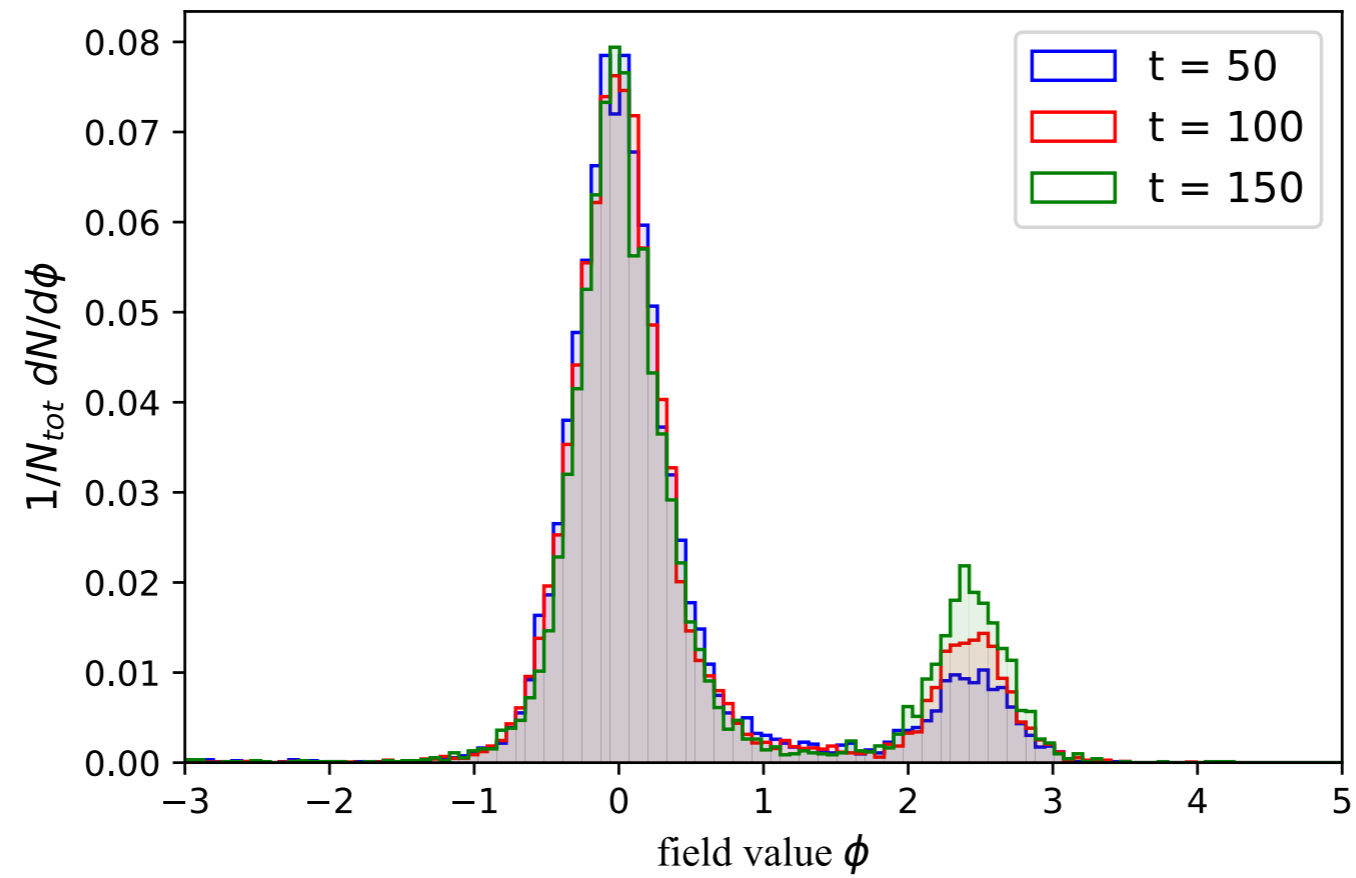
This is what the quantum annealer sees (i.e. taken directly from the couplings we pass to it)



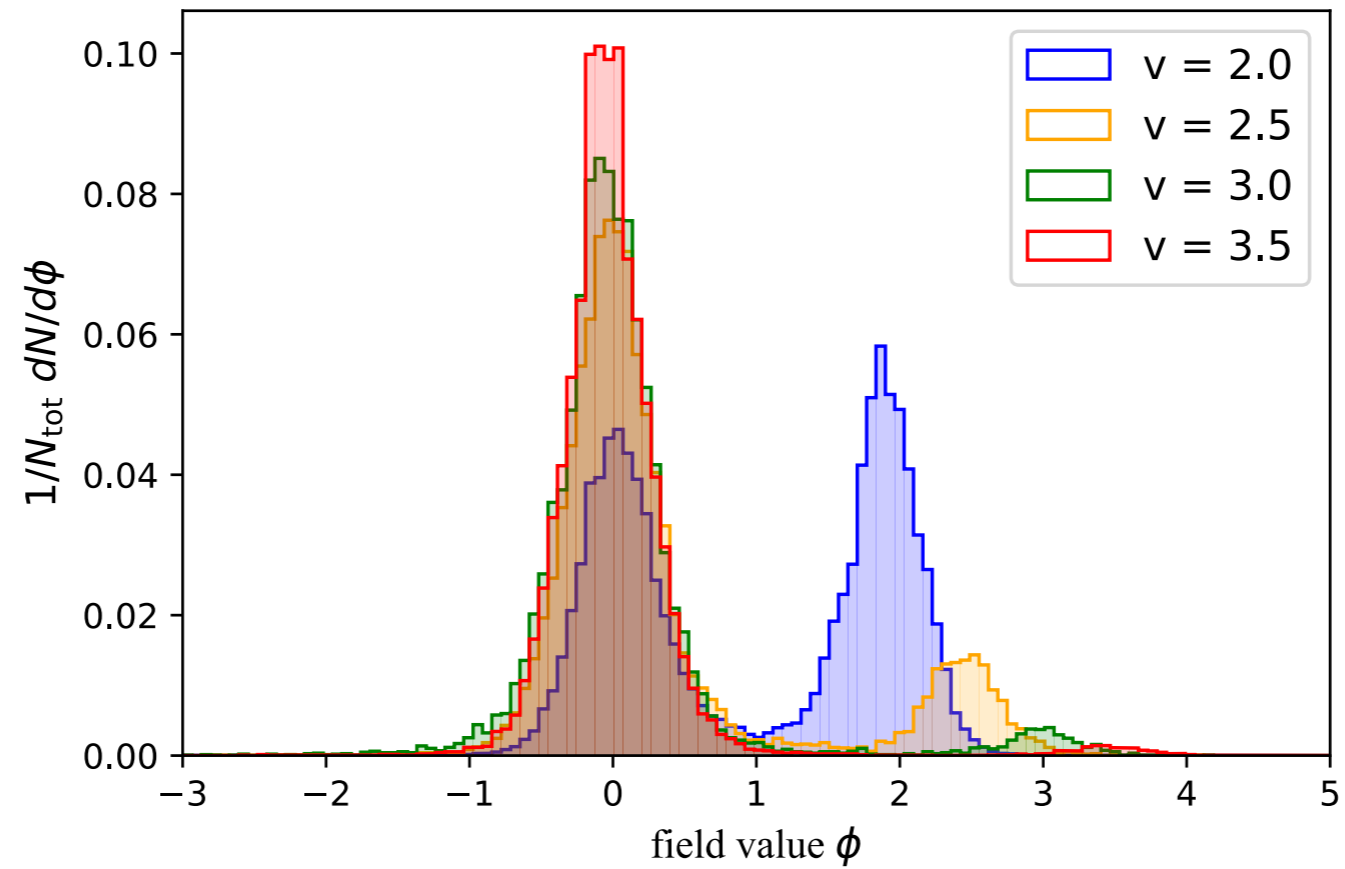
Results: (reverse anneal with 200 qubits) we see tunnelling — e.g. at $v=2.5$



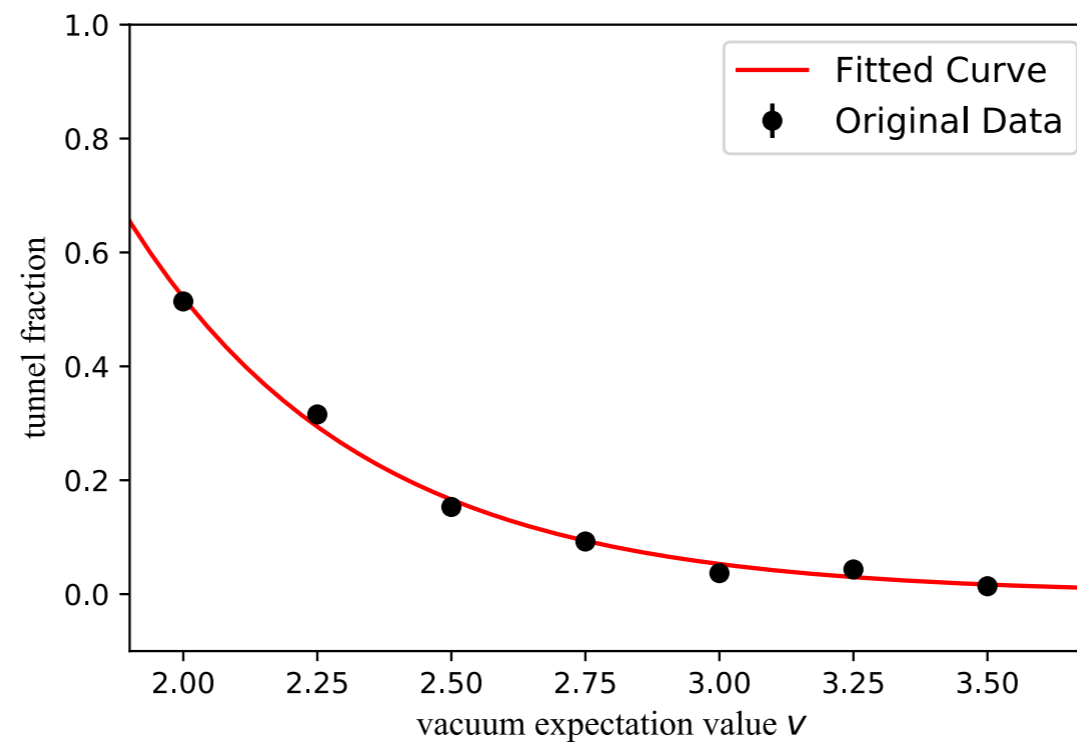
***Note that this is in principle an experimental measurement
of the tunnelling rate!***



Results: it appears to decrease exponentially with v as expected (WKB approximation):



Results: it appears to decrease exponentially with v as expected (WKB approximation):

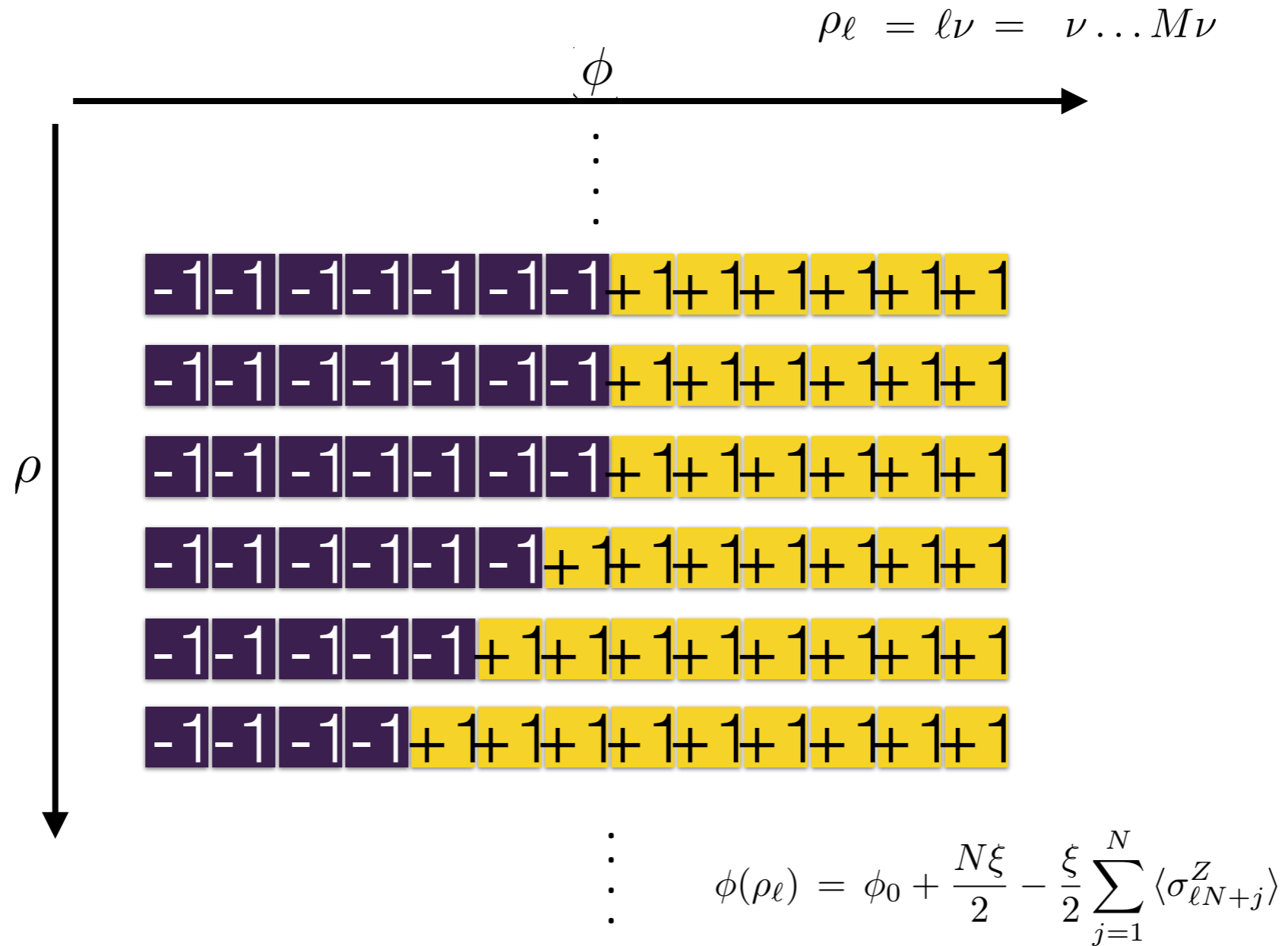


Theory:

$$\log \Gamma = 3.0 \times (1.66 - v)$$
$$\log \Gamma = 2.29 \times (1.71 - v)$$

Encoding space derivatives

Back to the domain wall encoding : add the discretised spacetime coordinates:



Everything done so far is then trivially extended in the / spacetime index:

$$h_{\ell N+j}^{(\text{chain})} = \Lambda (\delta_{j1} - \delta_{jN}) \quad J_{\ell N+i, m N+j}^{(\text{chain})} = -\frac{\Lambda}{2} \delta_{\ell m} \begin{pmatrix} 0 & 1 & & & & \\ 1 & 0 & 1 & & & \\ & 1 & 0 & & & \\ & & & \ddots & & \\ & & & & 0 & 1 \\ & & & & 1 & 0 \end{pmatrix}_{ij}$$

$$h_{N\ell+j}^{(\text{QFT})} = \begin{cases} -\frac{\nu\xi}{2} U'(\phi_0 + j\xi); & j < N \\ \frac{\nu}{2} U(\phi_0 + (N-1)\xi); & j = N \end{cases}$$

Then kinetic terms are as follows:

$$S_{KE} \equiv \int_0^{\Delta\rho} d\rho \frac{1}{2} \dot{\phi}^2 = \lim_{M \rightarrow \infty} \sum_{\ell=1}^{M-1} \frac{1}{2\nu} (\phi(\rho_{\ell+1}) - \phi(\rho_\ell))^2$$

$$= \sum_{\ell=1}^{M-1} \sum_{ij} \frac{\xi^2}{8\nu} \left[\sigma_{(\ell+1)N+i}^Z - \sigma_{\ell N+i}^Z \right] \times$$

$$\left[\sigma_{(\ell+1)N+j}^Z - \sigma_{\ell N+j}^Z \right]$$

Everything done so far is then trivially extended in the / spacetime index:

$$h_{\ell N+j}^{(\text{chain})} = \Lambda (\delta_{j1} - \delta_{jN}) \quad J_{\ell N+i, m N+j}^{(\text{chain})} = -\frac{\Lambda}{2} \delta_{\ell m} \begin{pmatrix} 0 & 1 & & & \\ 1 & 0 & 1 & & \\ & 1 & 0 & & \\ & & & \ddots & \\ & & & & 0 & 1 \\ & & & & 1 & 0 \end{pmatrix}_{ij}$$

$$h_{N\ell+j}^{(\text{QFT})} = \begin{cases} -\frac{\nu\xi}{2} U'(\phi_0 + j\xi) ; & j < N \\ \frac{\nu}{2} U(\phi_0 + (N-1)\xi) ; & j = N \end{cases}$$

Then kinetic terms are as follows:

$$J_{\ell N+i, m N+j}^{(\text{QFT})} = \frac{\xi^2}{8\nu} (2\delta_{\ell m} - \delta_{\ell(m+1)} - \delta_{(\ell+1)m})$$

May also need to impose physical boundary conditions: e.g. Z2 domain wall

$$\mathcal{H}^{(BC)} = \frac{\Lambda'}{2} (\phi(0) + v)^2 + \frac{\Lambda'}{2} (\phi(\rho_M) - v)^2$$

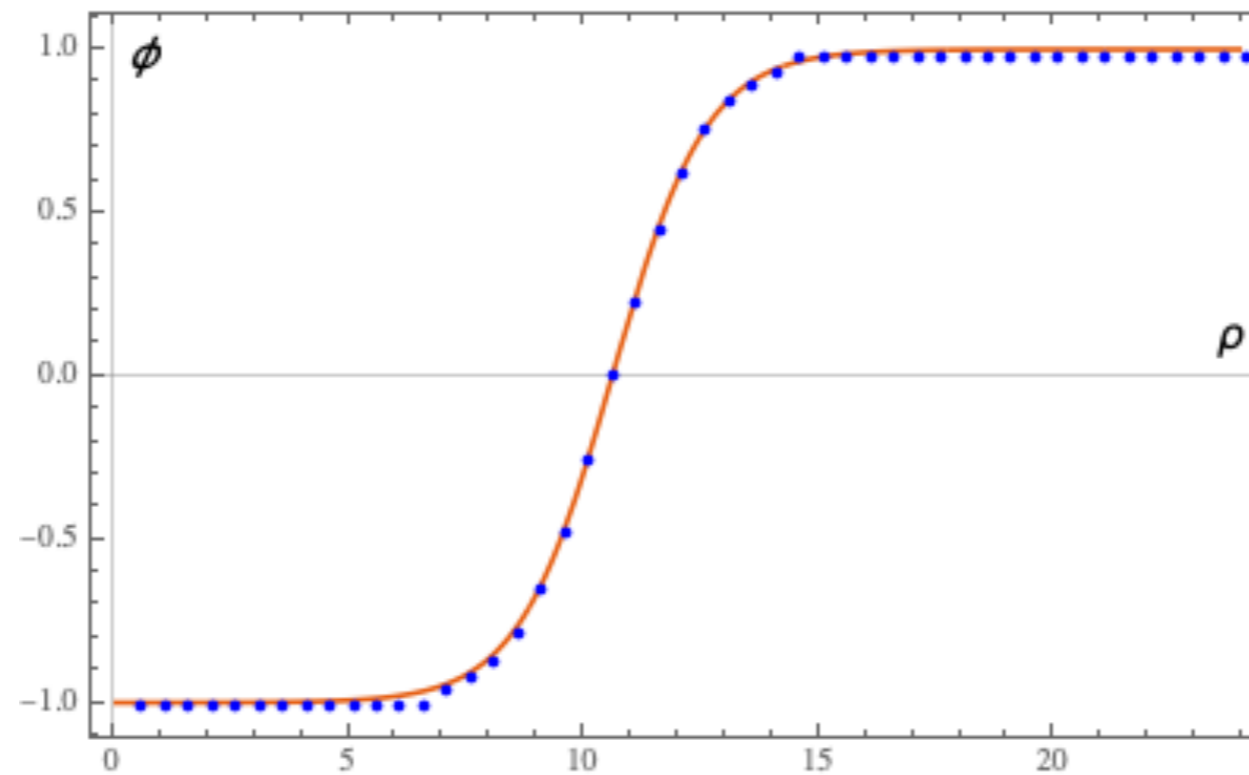
We can think of these as just boundary mass-term potentials in U :

$$h_{N\ell+j}^{(BC)} = \begin{cases} -\Lambda'(\phi_0 + j\xi + v) ; & \ell = 1, \forall j \\ -\Lambda'(\phi_0 + j\xi - v) ; & \ell = M - 1, \forall j \end{cases}$$

Add everything together:

$$\mathcal{H} = \mathcal{H}^{(\text{chain})} + \mathcal{H}^{(\text{QFT})} + \mathcal{H}^{(\text{BC})}.$$

Example Z2 domain wall potential in one space dimension on Dwave



So we can encode a pure field theory potential on the annealer, and can experiment with QFT tunnelling

Adiabatic Quantum Computing versus Quantum Annealing and QIBO



Can implement similar scheme on gate quantum computers using Qibo: <https://qibo.science/>

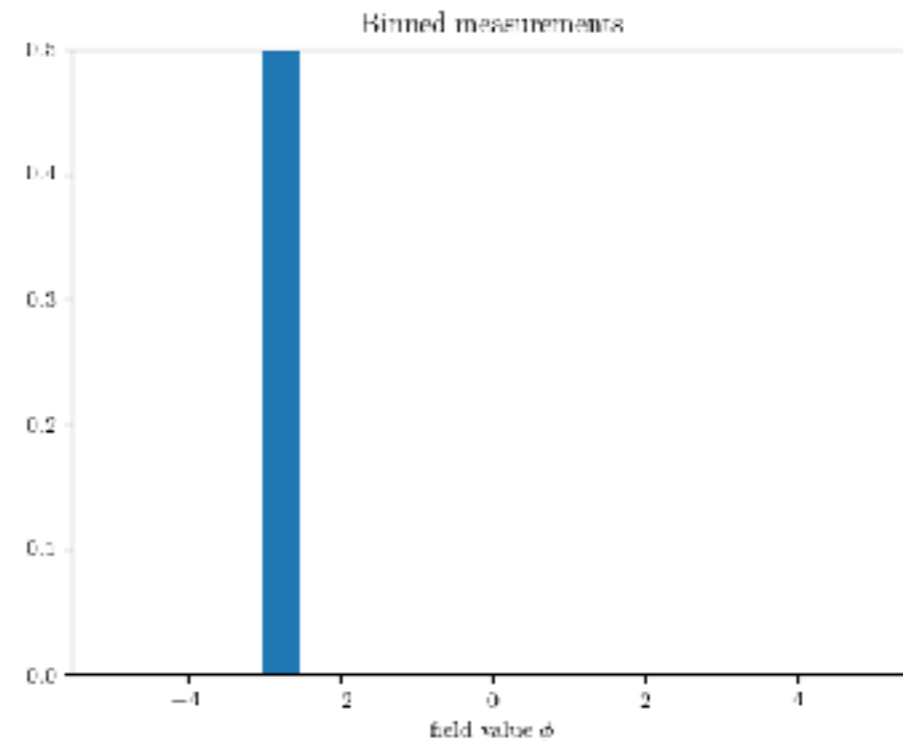
```
# Define Hamiltonian using Qibo symbols
# ZZ terms
symbolic_ham = sum( sum(J[i][j]* Z(i) * Z(j) for i in range(N)) for j in range(N))
# X terms
symbolic_ham += sum(h[i]*Z(i) for i in range(N))
symbolic_ham += sum( X(i) for i in range(N))

# Define a Hamiltonian using the above form
ham = hamiltonians.SymbolicHamiltonian(symbolic_ham)
# This Hamiltonian is memory efficient as it does not construct the full matrix
```

```
evolve = models.StateEvolution(ham, dt=0.01)

def final_state(t):
    return evolve(final_time=t, initial_state=initial_state)
```

Here is a wave-packet in the SHO potential:

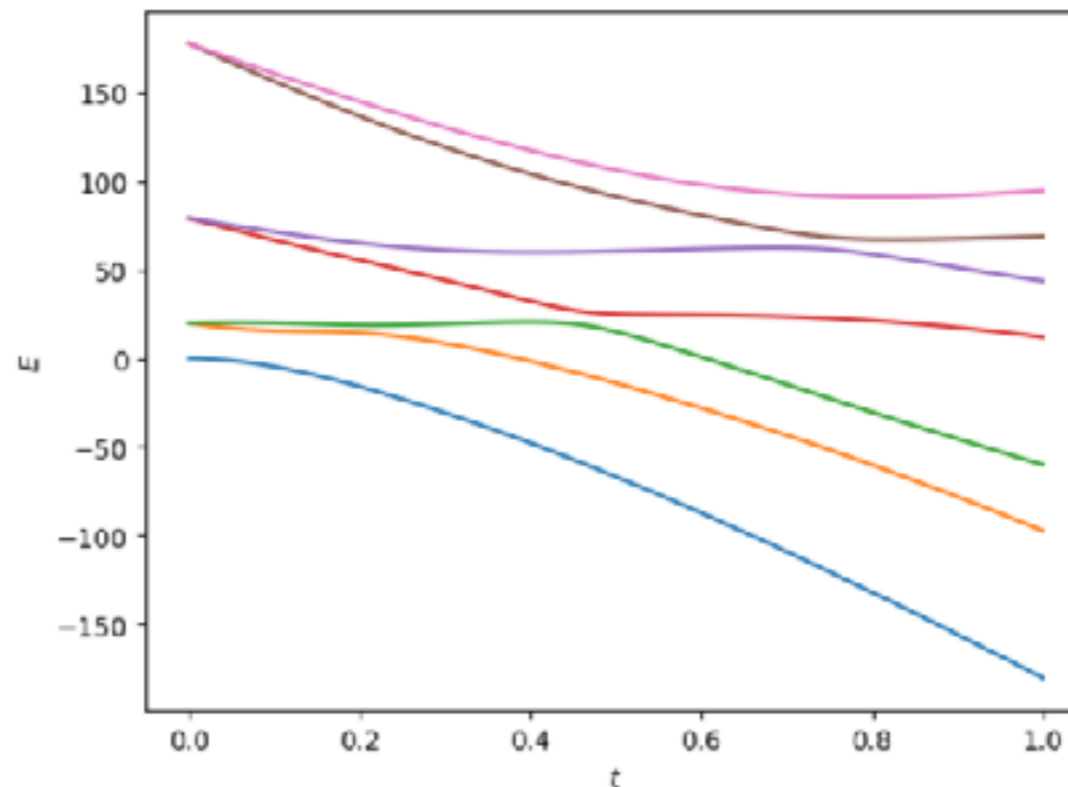


Quantum annealers like D-wave's are dissipative — they lose energy (and coherence) but are great for finding ground states by tunnelling.

However the idea of AQC was a bit different. To find the ground state of a complicated system we begin in the ground state of the pure transverse term which is trivial.

Then the *original idea* of quantum adiabatic computing was to remain in the ground state as we adjust the Hamiltonian to end up in the difficult Hamiltonian. (Farhi, Goldstone, Gutmann, Sipser)

i.e. The evolution of the spectrum should look something like ...

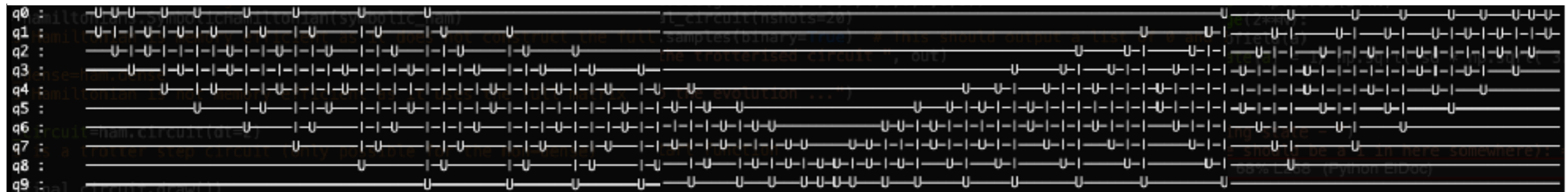


Qibo gives us a way to evolve the system using Trotterized circuits rather than the fixed Ising structure of the annealer.

e.g. the commands for a quadratic potential look like this ...

```
# Define Hamiltonian using Qibo symbols
# ZZ terms
symbolic_ham = sum( sum( c2* deltaphi* deltaphi* 2**(i+j - 2 * (N+1)) * (1+Z(i)) * (1+Z(j)) for i in range(N)) for j in range(N))
symbolic_ham += sum( 2* c2* phimin * deltaphi* 2**(i - (N+1)) * (1+Z(i)) for i in range(N))
symbolic_ham += c2* phimin * phimin
# X terms
symbolic_ham += sum( 0.1*(1+X(i)) for i in range(N))
```

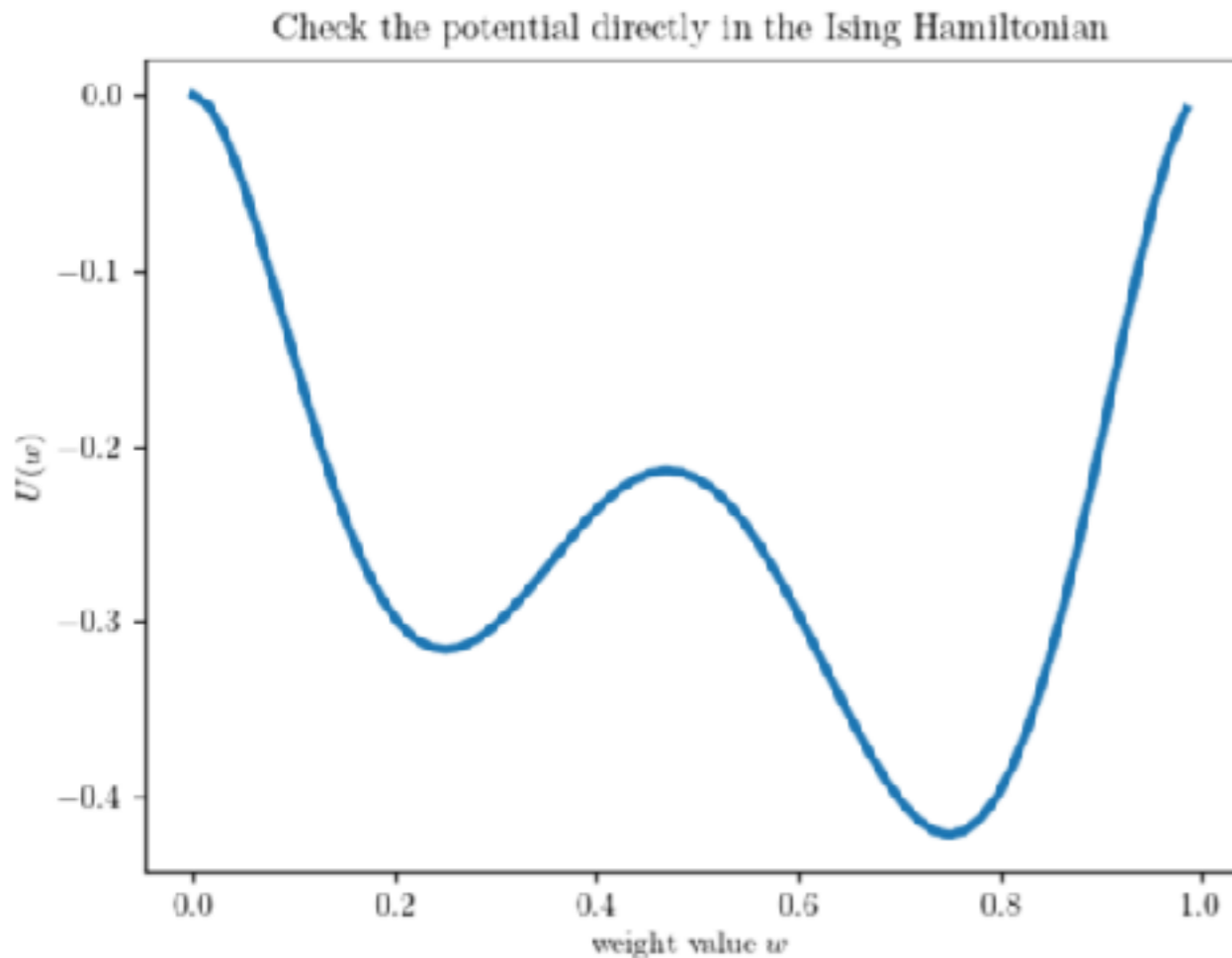
Trade-off between #qubits versus #gates: the circuit looks like this ...



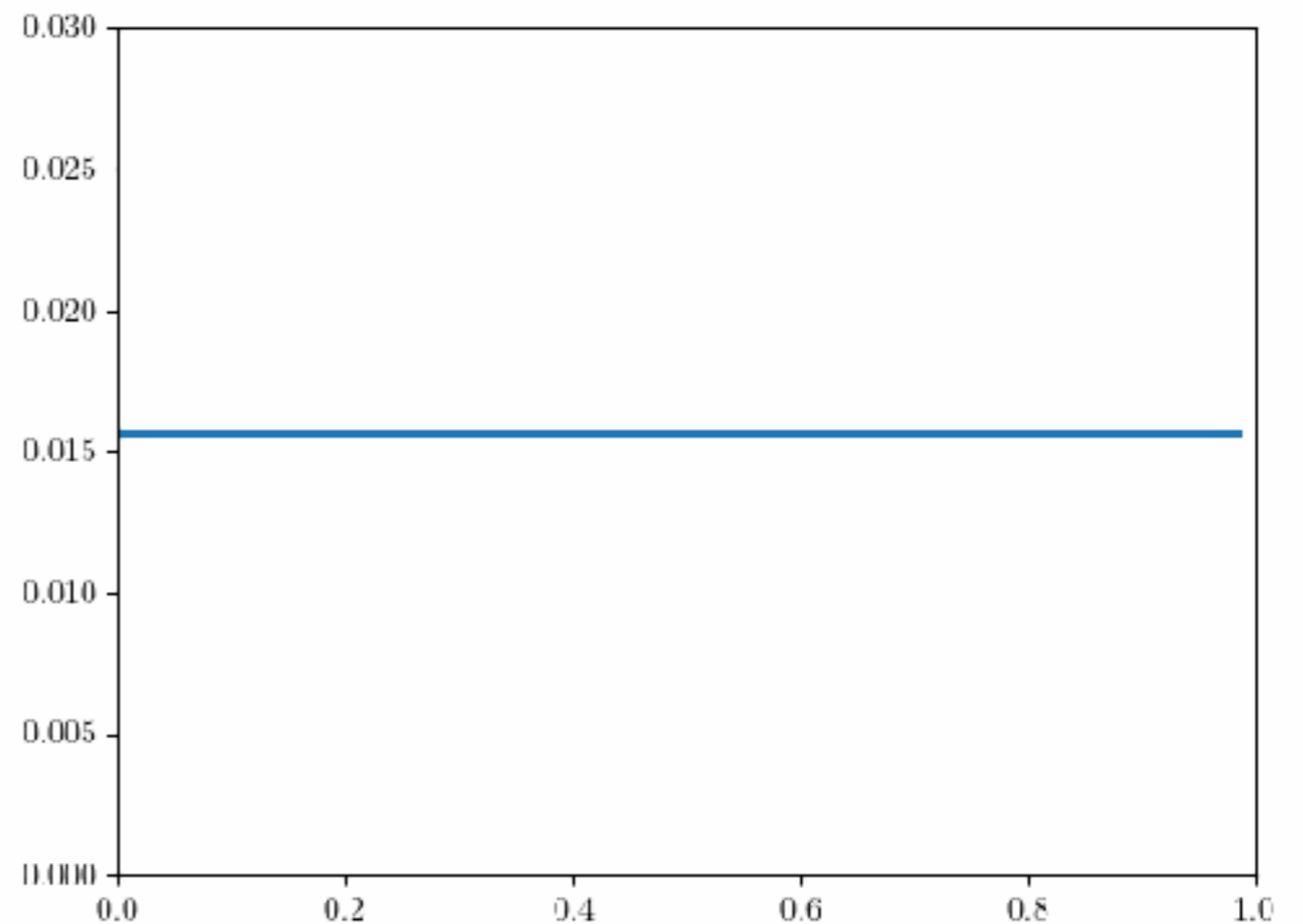
On the plus side the Hamiltonian does not need to be quadratic but can be high order.

For a binary encoded potential in H (*this is a septic in binaries encoded directly*) we can indeed find the ground state of the whole system:

```
evolve = models.AdiabaticEvolution(H0, H1, s, dt=dt, track_state=True)
```



This is what happens to the ground state during the anneal:



At the moment this is just a simulation
(~ 25 qubits possible to simulate)

Summary

- Annealers (both quantum and simulated) are useful tools for solving both discrete and continuous optimisation problems
- Most available *annealers* require reduction to quadratic Hamiltonian — leads to large qubit number (1000s)
- Nevertheless they can be used to train NNs and solve anomaly constraints
- We have seen how the general Ising model can be used to encode QFT
- Observe and measure genuine tunnelling out of false vacua (d=1 QFT)
- Important distinction between quantum annealing versus AQC. The latter can avoid need for reduction, but leads to large gate depth.

QA tutorial problems:

Tutor: Luca Nutricati

Reduction: reduce the following

$$H = \tau_1\tau_2\tau_3 + \tau_2\tau_3\tau_4 + \tau_1\tau_2\tau_3\tau_4.$$

where recall we can substitute for pairs with auxiliary qubits ...

$$Q(\tau_{12}; \tau_1, \tau_2) = \Lambda(\tau_1\tau_2 - 2\tau_{12}(\tau_1 + \tau_2) + 3\tau_{12})$$

thus ...

$$H \equiv \tau_{12}\tau_3 + \tau_2\tau_{34} + \tau_{12}\tau_{34} \\ + Q(\tau_{12}; \tau_1, \tau_2) + Q(\tau_{34}; \tau_3, \tau_4)$$

QA tutorial problems:

Tutor: Luca Nutricati

Factorising an integer X: let the factors be

$$w_i = 1 + 2\tau_i^1 + 4\tau_i^2$$

Let $H = (X - w_1 w_2)^2$

Then reduce the product:

$$w_1 w_2 = (1 + 2\tau_1^1 + 4\tau_1^2)(1 + 2\tau_2^1 + 4\tau_2^2)$$

$$= 1 + 2\tau_1^1 + 4\tau_1^2 + 2\tau_2^1 + 4\tau_2^2 + 4\tau_1^1\tau_2^1 + 16\tau_1^2\tau_2^2 + 8\tau_1^2\tau_2^1 + 8\tau_1^1\tau_2^2$$

$$\{w_1 w_2\} = \tau_1^1 + 4\tau_1^2 + 2\tau_2^1 + 4\tau_2^2 + 4\tau_{\tau_1^1\tau_2^1} + 16\tau_{\tau_1^2\tau_2^2} + 8\tau_{\tau_1^2\tau_2^1} + 8\tau_{\tau_1^1\tau_2^2}$$

Then

$$H_{reduced} = (X - \{w_1 w_2\})^2 + \sum_{i,j,a,b} Q(\tau_{\tau_i^a\tau_j^b}; \tau_i^a, \tau_j^b)$$

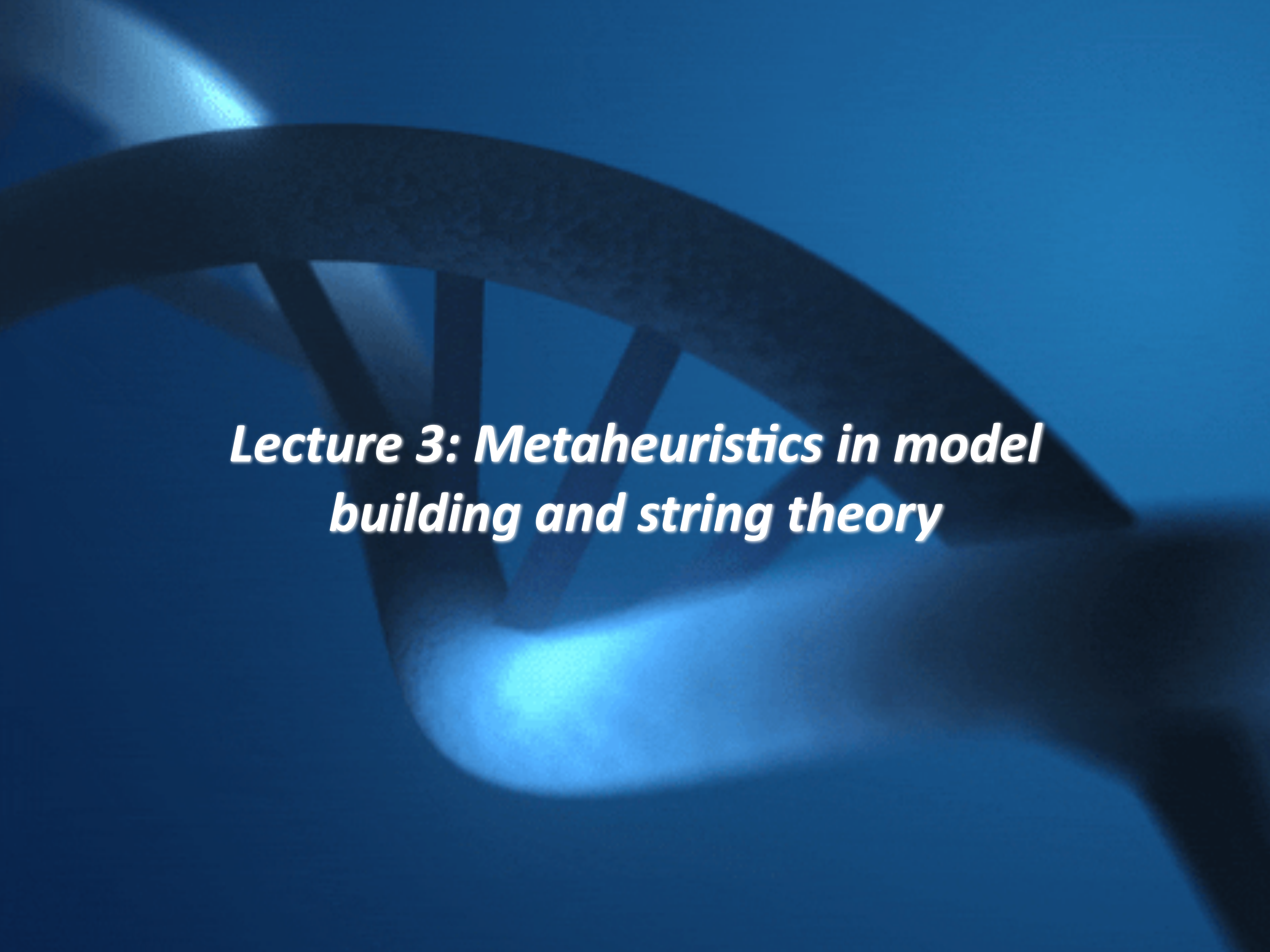
QA tutorial problems:

Tutor: Luca Nutricati

```
pip install dwave-neal
```

To check it in the drop-box you will find “Test Code.ipynb” which contains ...

```
import neal
sampler = neal.SimulatedAnnealingSampler()
h = {'a': 0.0, 'b': 0.0, 'c': 0.0}
J = {('a', 'b'): 1.0, ('b', 'c'): 1.0, ('a', 'c'): 1.0}
sampleset = sampler.sample_ising(h, J, num_reads=10)
if sampleset.first.energy == -1.0:
    print("You are ready for the tutorial!")
else:
    print("Ops, something went wrong!")
```

***Lecture 3: Metaheuristics in model
building and string theory***

GAs in particle physics ...

- Yamaguchi and H. Nakajima (2000)
- Allanach, Grellscheid, Quevedo (2004)
- Akrami, Scott, Edsjo, Conrad and Bergstrom (2009)
- Bl'aba'ck, Danielsson and Dibitetto, (2013)
- SAA, Rizos (2014)
- Ruehle (2017)
- SAA, Cerdeno, Robles (2018)
- Cole, Schachner, Shiu (2019)
- AbdusSalam, SAA, Cicoli, Quevedo, Shukla (2020)
- Bena, Bl'aba'ck, Grana, Luest (2021)
- SAA, Constantin, Lukas, Harvey (2021,23)
- Loges, Shiu (2021)
- Cole, Krippendorf, Schachner, Shiu (2021)
- Rawash, Turton (2022)
- SAA, Constantin, Lukas, Harvey, Nutricati (2023)
- Berglund, He, Heyes, Hirst, Jejjala, Lukas (2023)

Overview ...

- GA in early string work
- QA for particle model building and example stringy QA application
- GAs in heterotic line-bundle models
- GAs versus reinforcement learning
- Genetic Quantum Annealing

A dark blue background with a glowing blue DNA double helix structure. The helix is rendered in a semi-transparent, glowing style, with the two strands and the connecting rungs visible. The lighting is soft and focused on the central part of the helix, creating a sense of depth and highlighting the molecular structure.

Early GA string application

First string example

SAA+Rizos, 2014

- Find a phenomenologically attractive Pati-Salam model.
- We will consider the “fermionic string construction”. These are general 4D models in which the world sheet degrees of freedom are fermions. [Kawai, Lewellyn, Tye](#); [Antoniadis, Bachas, Kounnas](#)
- PS Models are defined in terms of a set of basis vectors [Faraggi, Kounnas, Nooij, Rizos](#)

$$v_1 = \mathbb{1} = \{ \psi^\mu, \chi^{1,\dots,6}, y^{1,\dots,6}, \omega^{1,\dots,6} | \bar{y}^{1,\dots,6}, \bar{\omega}^{1,\dots,6}, \bar{\eta}^{1,2,3}, \bar{\psi}^{1,\dots,5}, \bar{\phi}^{1,\dots,8} \}$$

$$v_2 = S = \{ \psi^\mu, \chi^{1,\dots,6} \}$$

$$v_{2+i} = e_i = \{ y^i, \omega^i | \bar{y}^i, \bar{\omega}^i \}, \quad i = 1, \dots, 6$$

$$v_9 = b_1 = \{ \chi^{34}, \chi^{56}, y^{34}, y^{56} | \bar{y}^{34}, \bar{y}^{56}, \bar{\eta}^1, \bar{\psi}^{1,\dots,5} \}$$

$$v_{10} = b_2 = \{ \chi^{12}, \chi^{56}, y^{12}, y^{56} | \bar{y}^{12}, \bar{y}^{56}, \bar{\eta}^2, \bar{\psi}^{1,\dots,5} \}$$

$$v_{11} = z_1 = \{ \bar{\phi}^{1,\dots,4} \}$$

$$v_{12} = z_2 = \{ \bar{\phi}^{5,\dots,8} \}$$

$$v_{13} = \alpha = \{ \bar{\psi}^{45}, \bar{y}^{1,2} \} .$$

- in addition to a set of GSO projection phases $c \begin{bmatrix} v_i \\ v_j \end{bmatrix}, i, j = 1, \dots, n$ which have to satisfy certain conditions (later)

Our genotype will be these phases (think of the string construction as a black box that turns these numbers into phenomenological model)

$$c_{ij} = \begin{pmatrix} \mathbb{1} & S & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & b_1 & b_2 & z_1 & z_2 & \alpha \\ \mathbb{1} & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ S & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ e_1 & 1 & 1 & 0 & \ell_{26} & \ell_{27} & \ell_{28} & \ell_{29} & \ell_{30} & \ell_6 & 0 & \ell_{14} & \ell_{20} & \ell_{41} \\ e_2 & 1 & 1 & \ell_{26} & 0 & \ell_{31} & \ell_{32} & \ell_{33} & \ell_{34} & \ell_7 & 0 & \ell_{15} & \ell_{21} & \ell_{42} \\ e_3 & 1 & 1 & \ell_{27} & \ell_{31} & 0 & \ell_{35} & \ell_{36} & \ell_{37} & 0 & \ell_{10} & \ell_{16} & \ell_{22} & \ell_{43} \\ e_4 & 1 & 1 & \ell_{28} & \ell_{32} & \ell_{35} & 0 & \ell_{38} & \ell_{39} & 0 & \ell_{11} & \ell_{17} & \ell_{23} & \ell_{44} \\ e_5 & 1 & 1 & \ell_{29} & \ell_{33} & \ell_{36} & \ell_{38} & 0 & \ell_{40} & \ell_8 & \ell_{12} & \ell_{18} & \ell_{24} & \ell_{45} \\ e_6 & 1 & 1 & \ell_{30} & \ell_{34} & \ell_{37} & \ell_{39} & \ell_{40} & 0 & \ell_9 & \ell_{13} & \ell_{19} & \ell_{25} & \ell_{46} \\ b_1 & 0 & 0 & \ell_6 & \ell_7 & 0 & 0 & \ell_8 & \ell_9 & 1 & 0 & \ell_2 & \ell_4 & \ell_{47} \\ b_2 & 0 & 0 & 0 & 0 & \ell_{10} & \ell_{11} & \ell_{12} & \ell_{13} & 0 & 1 & \ell_3 & \ell_5 & \ell_{48} \\ z_1 & 1 & 1 & \ell_{14} & \ell_{15} & \ell_{16} & \ell_{17} & \ell_{18} & \ell_{19} & \ell_2 & \ell_3 & 1 & \ell_1 & \ell_{49} \\ z_2 & 1 & 1 & \ell_{20} & \ell_{21} & \ell_{22} & \ell_{23} & \ell_{24} & \ell_{25} & \ell_4 & \ell_5 & \ell_1 & 1 & \ell_{50} \\ \alpha & 1 & 1 & \ell_{41} & \ell_{42} & \ell_{43} & \ell_{44} & \ell_{45} & \ell_{46} & \ell_{47} + 1 & \ell_{48} + 1 & \ell_{49} + 1 & \ell_{50} & \ell_{51} \end{pmatrix} \text{ mod } 2$$

$$c \begin{bmatrix} v_i \\ v_j \end{bmatrix}, i, j = 1, \dots, n$$

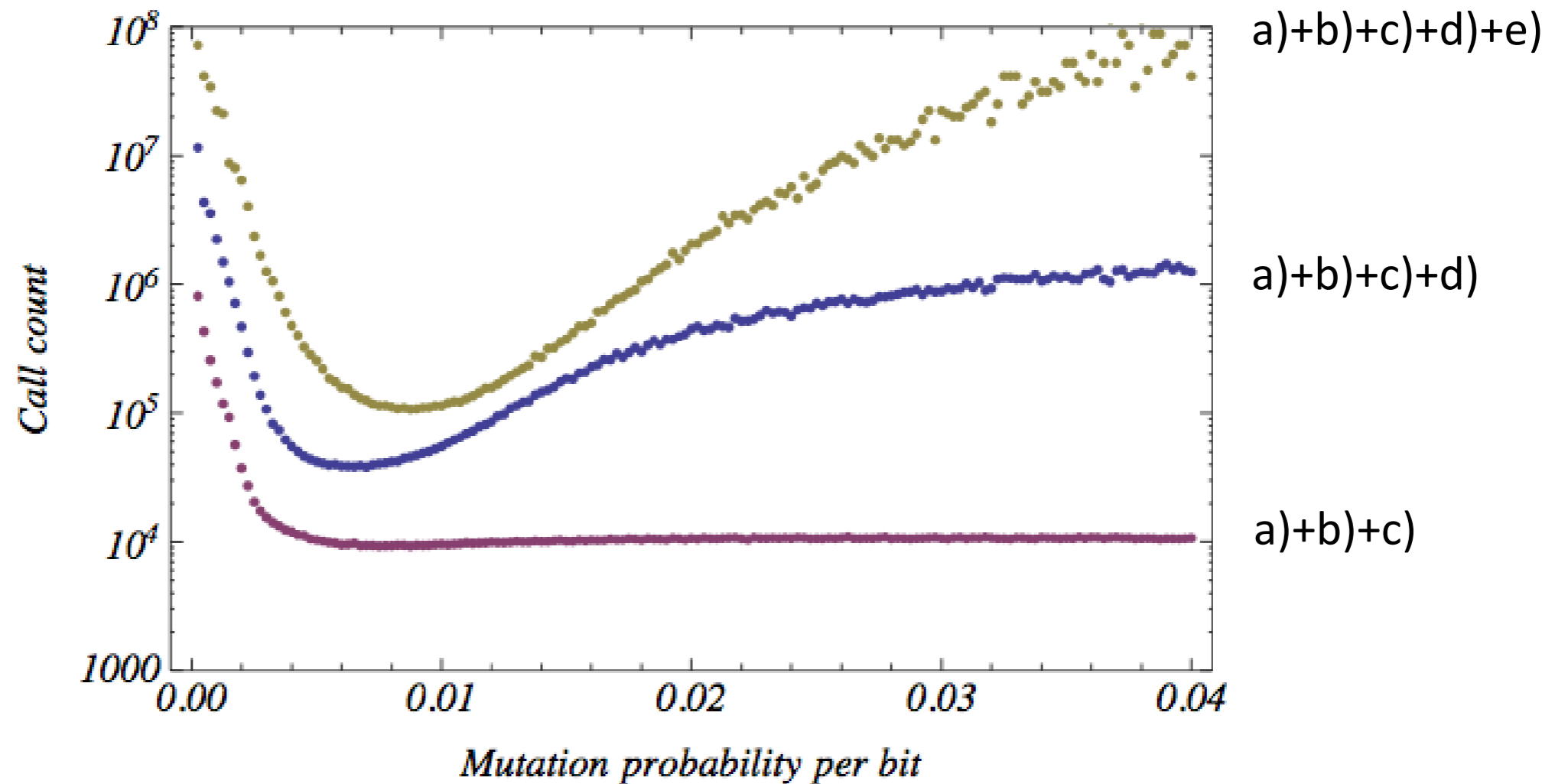
51 independent phases in these models: hence search space is $2^{51} = 2 \times 10^{15}$

This search space is (just about) searchable deterministically so we can compare the two methods.

The phases determine the characteristics of the models

- (a) 3 complete family generations, $n_g = 3$
- (b) Existence of PS breaking Higgs, $k_R \geq 1$
- (c) Existence of SM Higgs doublets, $n_h \geq 1$
- (d) Absence of exotic fractional charge states, $n_e = 0$
- (e) Existence of top Yukawa coupling

- a)+b)+c) = 1 : 10,000
- a)+b)+c)+d) = 1 : 2,500,000
- a)+b)+c)+d)+e) = 1 : 10,000,000,000
- deterministically we would expect to have to construct 10 billion models to find an example of the latter



- As in toy examples optimum mutation rate => genetic algorithm is working as expected
- GA's do not confer much advantage when the search is "easy"
- They work best when there are many criteria and the search is difficult



QAs for direct model building

Warm-up: solving anomaly conditions

SAA, Nutricati

An example TaxiCab-like problem that we can take as a “prototypical” consistency condition. Consider U(1) extension to the SM: need to solve ...

$$\begin{aligned}\sum_{i=1}^3 (6Q_i + 3U_i + 3D_i + 2L_i + E_i + N_i) &= 0, \\ \sum_{i=1}^3 (3Q_i + L_i) &= 0, \\ \sum_{i=1}^3 (2Q_i + U_i + D_i) &= 0, \\ \sum_{i=1}^3 (Q_i + 8U_i + 2D_i + 3L_i + 6E_i) &= 0, \\ \sum_{i=1}^3 (Q_i^2 - 2U_i^2 + D_i^2 - L_i^2 + E_i^2) &= 0, \\ \sum_{i=1}^3 (6Q_i^3 + 3U_i^3 + 3D_i^3 + 2L_i^3 + E_i^3 + N_i^3) &= 0.\end{aligned}$$

for some reasonable fractional charges.

Recall: quite tricky to do because we have to encode numbers on qubits as binaries, and we have to encode a sextic spin-polynomial so we use the “reduction method”: where we trade successive pairs of qubits for auxiliary qubits.

Derive a list of suitable anomaly-free extensions (numerators of) with this Ising encoding:

Q_1	Q_2	Q_3	U_1	U_2	U_3	D_1	D_2	D_3	L_1	L_2	L_3	E_1	E_2	E_3	N_1	N_2	N_3
-1	0	1	-1	0	1	1	-1	0	1	0	-1	0	-1	1	1	0	-1
0	-2	2	1	-1	2	-2	0	0	0	1	-1	0	-1	-1	0	1	1
3	-1	-2	-1	-2	3	-4	2	2	0	-3	3	2	-2	0	2	-3	1
3	-2	-1	1	-3	3	-4	3	0	-1	0	1	-1	0	0	3	-3	1
-1	1	0	-2	-1	4	-5	4	0	-2	-1	3	0	2	-3	1	-2	2
1	-1	0	0	-2	5	-6	4	-1	-1	0	1	0	-1	-2	0	-2	5
1	0	0	-1	-2	6	-7	4	-2	-3	0	0	2	1	-4	0	0	7
2	-1	-1	2	-3	4	-6	2	1	0	0	0	-3	1	-1	-3	-2	8
2	-2	-2	2	1	2	-2	-1	2	6	-1	1	-3	-1	-5	-10	-2	9
1	-3	0	1	5	2	-2	1	-3	2	1	3	-5	-3	-4	-13	0	13

This particular search is a space of size $26^{18} \sim 3 \times 10^{25}$

Towards direct QA string model building

SAA, Nutricati+Rizos, 2023

Morally can actually be a simpler task as long as we don't ask the annealer to "count" generations: e.g. for the fermionic string the GSO projections for the spinorials (matter) in a particular $SO(10) \times U(1)^3 \times SO(8)^2$ model is possible ...

- These models are defined in terms of the following 12 basis vectors

$$\beta_1 = \mathbb{1} = \{\psi^\mu, x^{1,\dots,6}, y^{1,\dots,6}, \omega^{1,\dots,6}; \\ \bar{y}^{1,\dots,6}, \bar{\omega}^{1,\dots,6}, \bar{\psi}^{1,\dots,5}, \bar{\eta}^{1,2,3}, \bar{\phi}^{1,\dots,4}, \bar{\phi}^{5,\dots,8}\},$$

$$\beta_2 = S = \{\psi^\mu, x^{1,\dots,6}\},$$

$$\beta_{2+i} = e_i = \{y^i \omega^i; \bar{y}^i, \bar{\omega}^i\}, i = 1, \dots, 6,$$

$$\beta_9 = b_1 = \{x^{34}, x^{56}, y^{3,4}, y^{5,6}; \bar{y}^{3,4}, \bar{y}^{5,6}, \bar{\psi}^{1,\dots,5}, \bar{\eta}^1\},$$

$$\beta_{10} = b_2 = \{x^{12}, x^{56}, y^{1,2}, y^{5,6}; \bar{y}^{1,2}, \bar{y}^{5,6}, \bar{\psi}^{1,\dots,5}, \bar{\eta}^2\},$$

$$\beta_{11} = z_1 = \{\bar{\phi}^{1,2,3,4}\},$$

$$\beta_{12} = z_2 = \{\bar{\phi}^{5,6,7,8}\}$$

- In a particular sector the GSO constraints turn into constraints on the Lorentz dot products of the vectors. e.g. for 3 matter spinorials we end up looking for solutions to

$$\Delta^I U_s^I = Y_s^I, \quad I = 1, 2, 3,$$

where the U's entries are 0,1 denoting the sector (in terms of basis vectors), and all the GSO phases are contained in a bunch of phases defined by the basis vectors: $(a|b) \equiv e^{i\pi a \cdot b}$

$$\Delta^1 = \begin{pmatrix} (e_1|e_3) & (e_1|e_4) & (e_1|e_5) & (e_1|e_6) \\ (e_2|e_3) & (e_2|e_4) & (e_2|e_5) & (e_2|e_6) \\ (z_1|e_3) & (z_1|e_4) & (z_1|e_5) & (z_1|e_6) \\ (z_2|e_3) & (z_2|e_4) & (z_2|e_5) & (z_2|e_6) \end{pmatrix}, \quad Y_s^1 = \begin{pmatrix} (e_1|b_1) \\ (e_2|b_1) \\ (z_1|b_1) \\ (z_2|b_1) \end{pmatrix}, \quad Y_s^2 = \begin{pmatrix} (e_3|b_2) \\ (e_4|b_2) \\ (z_1|b_2) \\ (z_2|b_2) \end{pmatrix}, \quad Y_s^3 = \begin{pmatrix} (e_5|b_3) \\ (e_6|b_3) \\ (z_1|b_3) \\ (z_2|b_3) \end{pmatrix}$$

$$\Delta^2 = \begin{pmatrix} (e_3|e_1) & (e_3|e_2) & (e_3|e_5) & (e_3|e_6) \\ (e_4|e_1) & (e_4|e_2) & (e_4|e_5) & (e_4|e_6) \\ (z_1|e_1) & (z_1|e_2) & (z_1|e_5) & (z_1|e_6) \\ (z_2|e_1) & (z_2|e_2) & (z_2|e_5) & (z_2|e_6) \end{pmatrix}, \quad U_s^1 = \begin{pmatrix} p_s^1 \\ q_s^1 \\ r_s^1 \\ s_s^1 \end{pmatrix}, \quad U_s^2 = \begin{pmatrix} p_s^2 \\ q_s^2 \\ r_s^2 \\ s_s^2 \end{pmatrix}, \quad U_s^3 = \begin{pmatrix} p_s^3 \\ q_s^3 \\ r_s^3 \\ s_s^3 \end{pmatrix}$$

$$\Delta^3 = \begin{pmatrix} (e_5|e_1) & (e_5|e_2) & (e_5|e_3) & (e_5|e_4) \\ (e_6|e_1) & (e_6|e_2) & (e_6|e_3) & (e_6|e_4) \\ (z_1|e_1) & (z_1|e_2) & (z_1|e_3) & (z_1|e_4) \\ (z_2|e_1) & (z_2|e_2) & (z_2|e_3) & (z_2|e_4) \end{pmatrix}$$

- Similar for PS Higgs scalars etc. The main point is if we choose sectors (preprocessing) we can encode the entire set of equivalent equations for matter, Higgs, and also top Yukawa coupling.

- Finally have to impose chirality constraints by hand (post-processing) which thanks to our pre-processing is a constraint on the third T2 compactification plane only ...

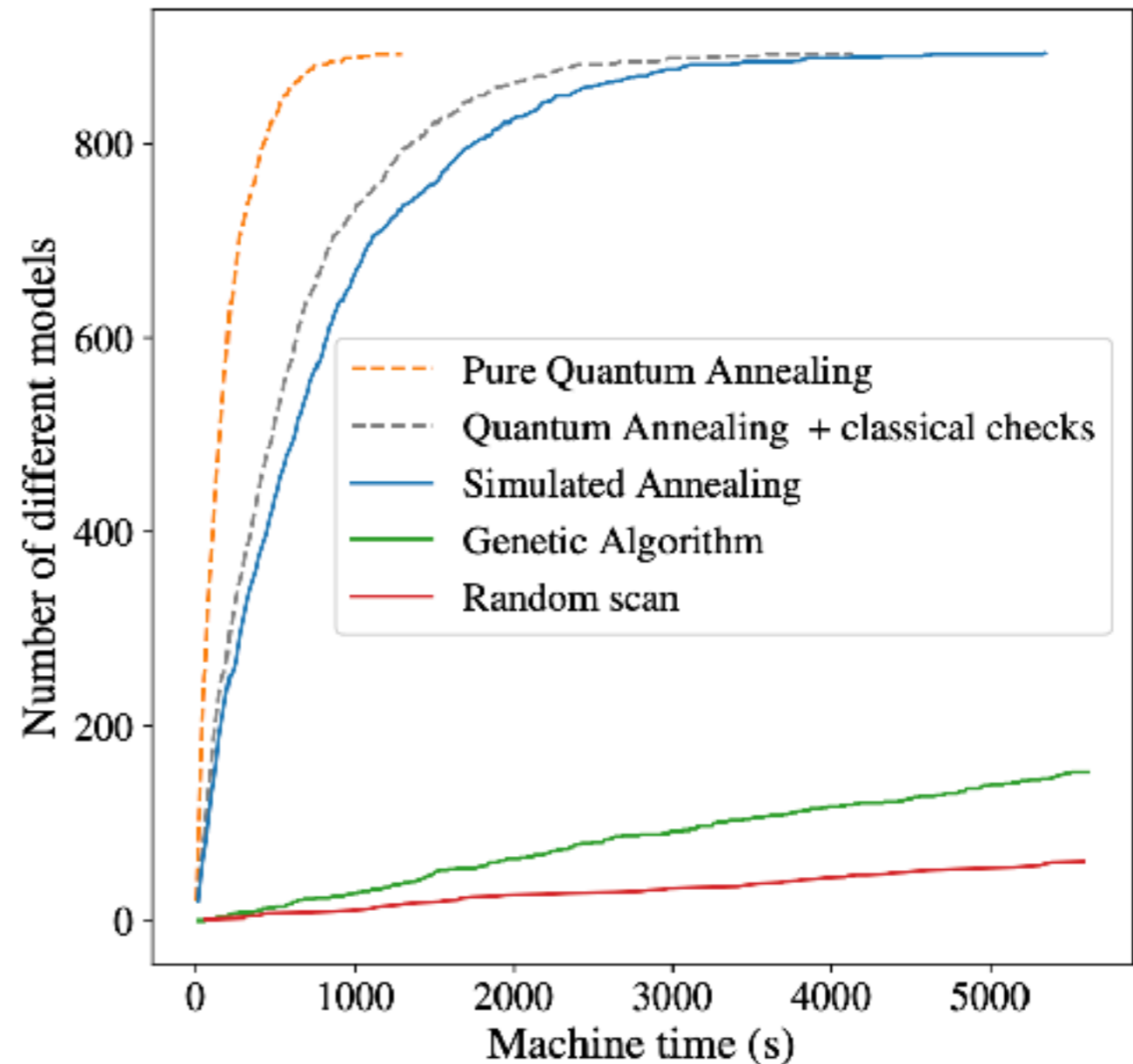
$$N_F = \sum_{I=1}^3 \sum_{p,q,r,s \in \Xi_s^I} X_{pqrs}^{(I)} = 3$$

$$X_{pqrs}^{(I)} = \exp(i\pi\chi_{pqrs}^{(I)})$$

where the exponent is another function of the phases.

In general we note three things:

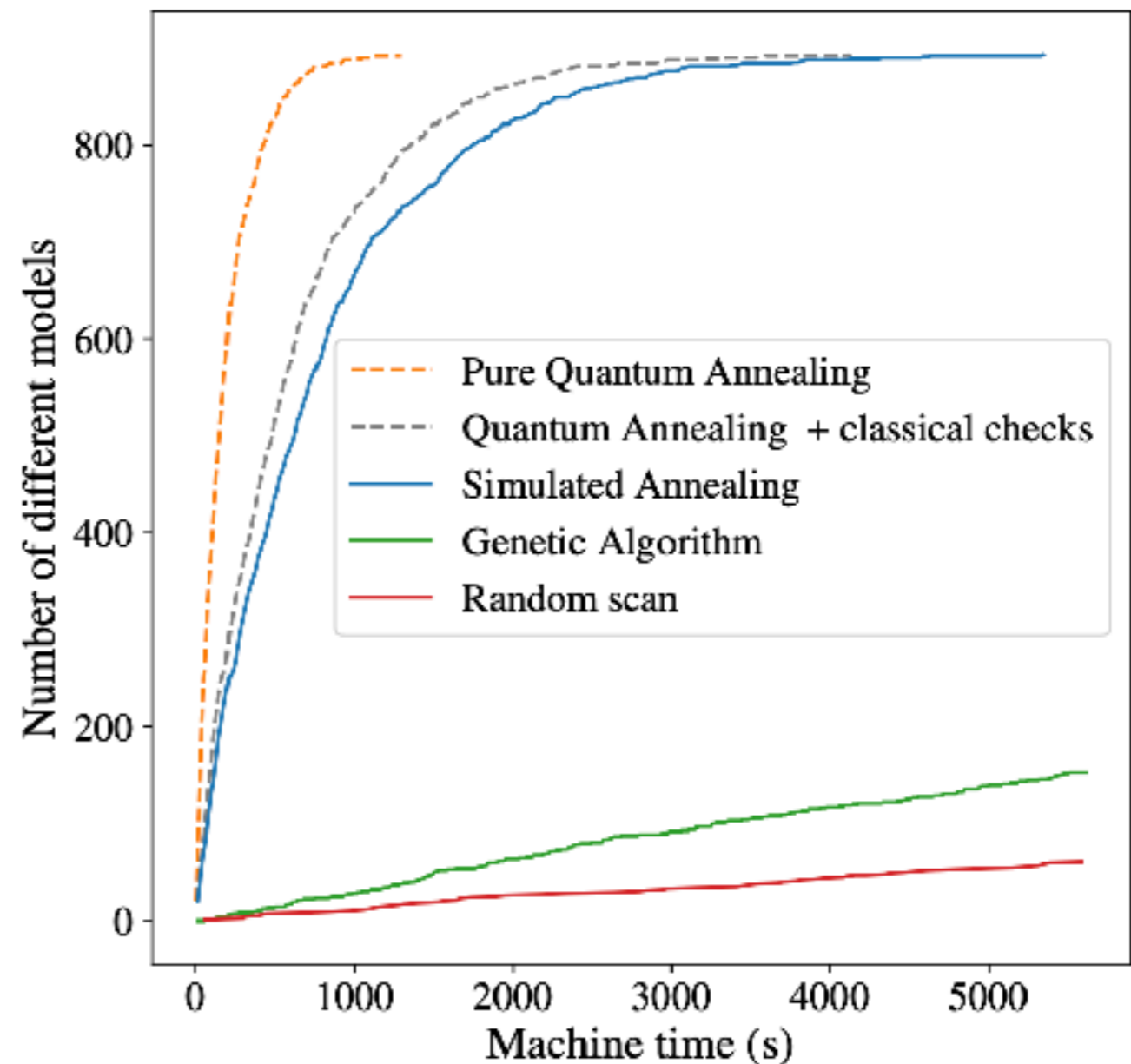
- The encoding of the system has to be done in detail - (unlike the GA where your friend John comes along and you tack your GA code on to his model code)
- Certain things are hard to encode: counting numbers of generations for example
- However for this problem we find ...



The search space for these models has 28 free +/-1 phases choices. Hence ...

We know there are about 1:10000 good models in the search space. We saturate at 1500 for a particular choice of matter sectors after 0.5M reads. In other words only 300 reads required to find a good model instead of 10^4 .

Caveat: arguably this is “GA-easy” so we would not expect huge GA improvement over the scan.





***GAs for heterotic line bundle
moduli***

Heterotic CY line-bundle set-up

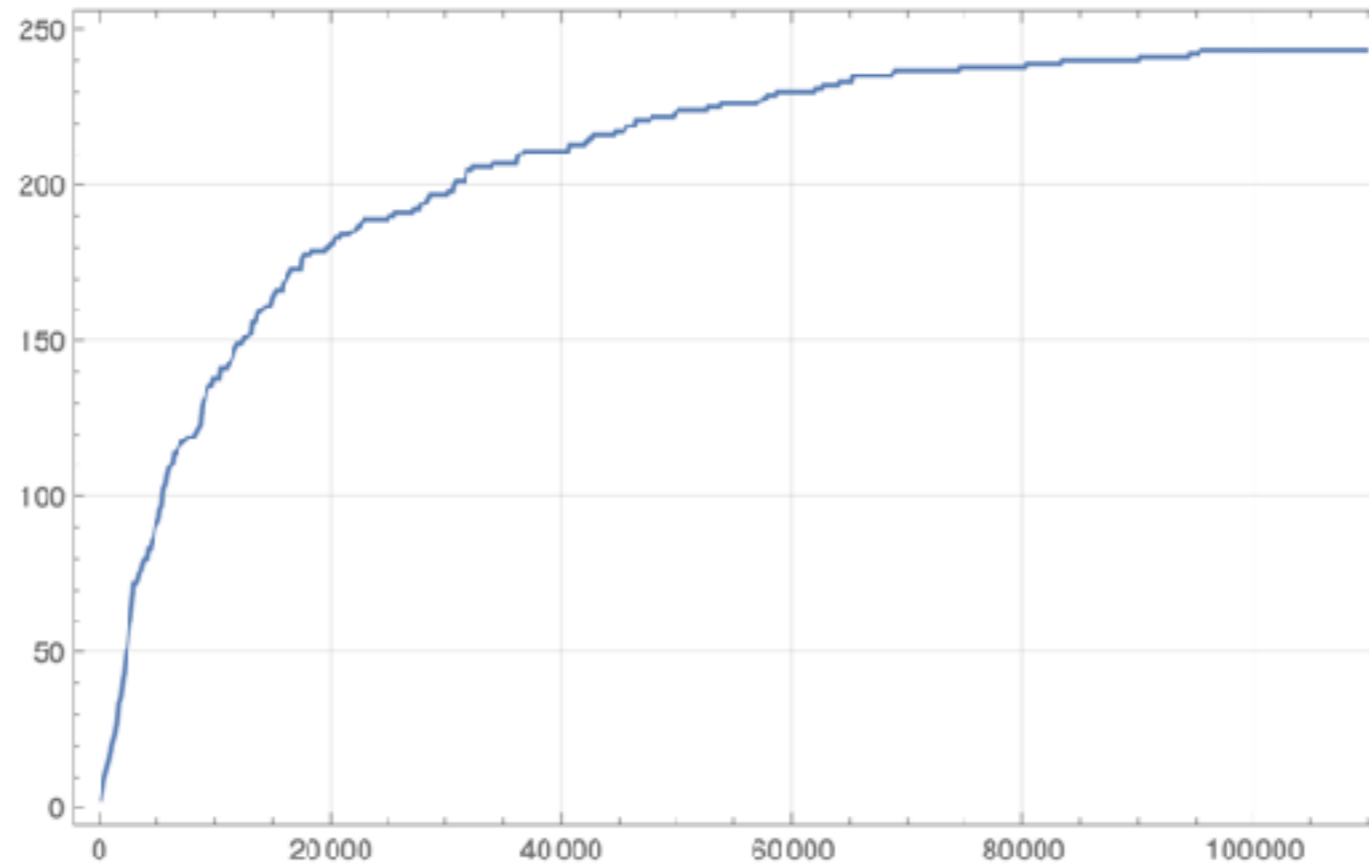
SAA, Constantin, Lukas, Harvey, Nutricati

- CICY threefold X with line-bundles $V = \bigoplus_{a=1}^5 L_a$ to break the $E_8 \times E_8$ gauge symmetry to the Standard Model gauge group or to one of its grand unification embeddings.
- Distinct pairs (X, V) that can serve as compactification data virtually unbounded [Anderson, Constantin, Gray, Lukas, Palti; Buchbinder, Constantin, Lukas](#)
- Considered the following CYs with configuration matrices

$$\begin{aligned}
 X_{7862}^{(4,68)} &= \begin{matrix} \mathbb{P}^1 \\ \mathbb{P}^1 \\ \mathbb{P}^1 \\ \mathbb{P}^1 \end{matrix} \begin{bmatrix} 2 \\ 2 \\ 2 \\ 2 \end{bmatrix}, & X_{7447}^{(5,45)} &= \begin{matrix} \mathbb{P}^1 \\ \mathbb{P}^1 \\ \mathbb{P}^1 \\ \mathbb{P}^1 \\ \mathbb{P}^1 \end{matrix} \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \\
 X_{5302}^{(6,30)} &= \begin{matrix} \mathbb{P}^1 \\ \mathbb{P}^1 \\ \mathbb{P}^1 \\ \mathbb{P}^1 \\ \mathbb{P}^1 \\ \mathbb{P}^1 \end{matrix} \begin{bmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}, & X_{4071}^{(7,27)} &= \begin{matrix} \mathbb{P}^1 \\ \mathbb{P}^2 \\ \mathbb{P}^1 \\ \mathbb{P}^1 \\ \mathbb{P}^1 \\ \mathbb{P}^2 \\ \mathbb{P}^3 \end{matrix} \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}
 \end{aligned}$$

- First Chern class $c_1(L_a) = k_a^i J_i$, where k_a^i are the components of the 5 integer vectors $k_a \in \mathbb{Z}^h$ and (J_1, \dots, J_h) is a basis of $H^2(X, \mathbb{Z})$.
- Assumed Wilson line breaking to SM on X/Γ
- Apply line-bundle cohomology formulae for the defining integers to satisfy various constraints (E_8 embedding/vanishing C1, anomaly cancellation, SUSY/polystability, spectrum, equivariance): [Constantin, Lukas; Klaewer, Schlechter; Larfors, Schneider; Brodie, Constantin, Lukas](#)
- e.g. E_8 embedding $\Rightarrow c_1(V) = \sum_{a=1}^5 k_a \stackrel{!}{=} 0$. fixes one set of k 's so $4h$ integers left
- Then e.g. Spectrum: cohomology dimensions must satisfy ...
 10-multiplets: $h^1(X, V) = 3|\Gamma|$; no $\bar{10}$ -multiplets: $h^2(X, V) = 0$
 5-multiplets: $h^1(X, \wedge^2 V) = 3|\Gamma| + nh$, $nh > 0$; Higgs: $h^2(X, \wedge^2 V) = nh$;
 Chiral spectrum: $\chi(X, V) = \chi(X, \wedge V) = 3|\Gamma|$
- Take $k_a^i \in \{-2^n + 1, \dots, 2^n\}$; with $n=3$ for all except $n=2$ for X_{4071} .
- Search space is roughly $2^{4h(n+1)}$ so e.g. for 5302 this is 10^{29}

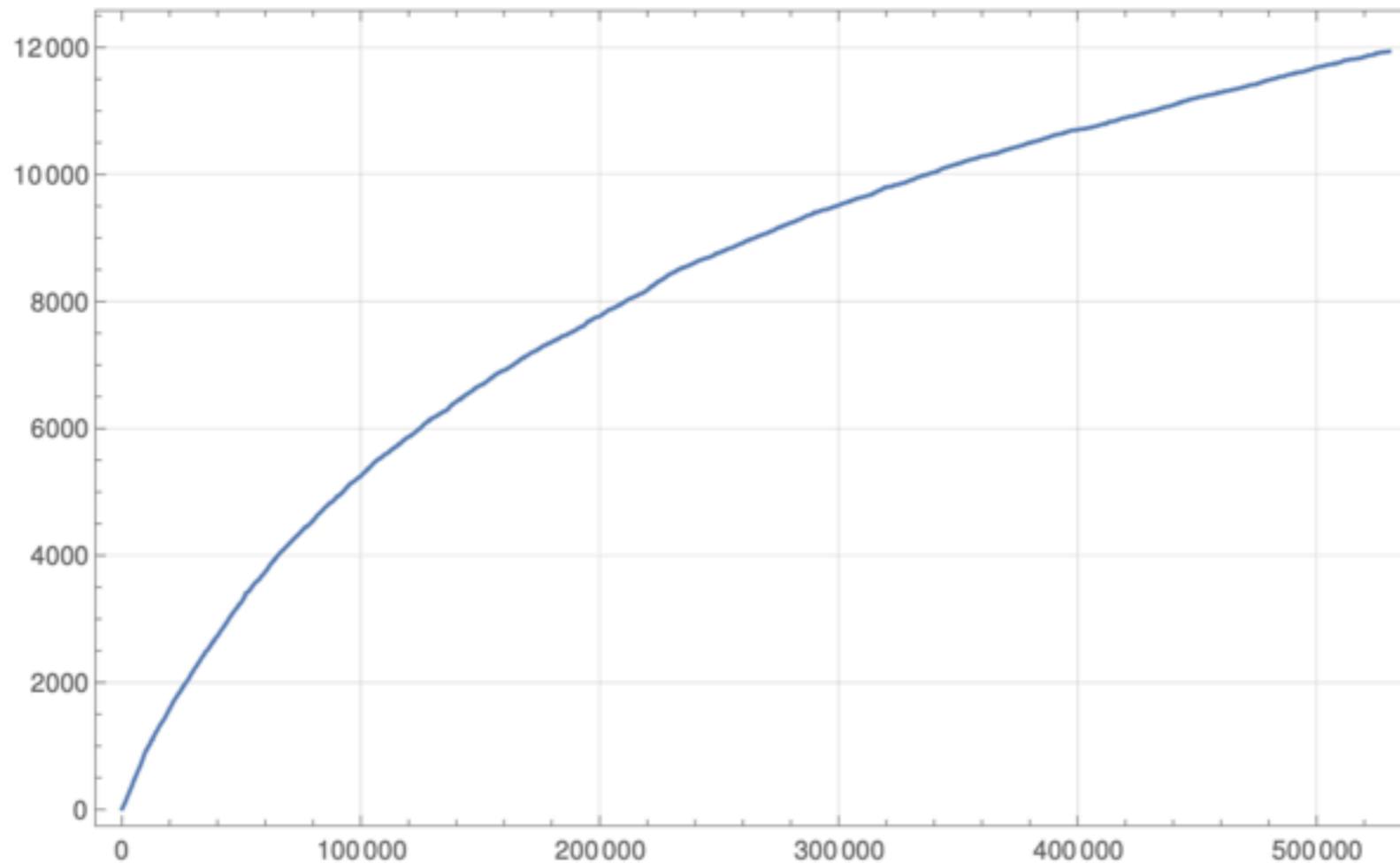
Searched for $N = 1$ supersymmetric $SU(5)$ GUTs with four additional Green-Schwarz anomalous $U(1)$ s, three $(\bar{5}, 10)$ generations, no exotic $\bar{10}$ multiplets and at least one vector-like $5-\bar{5}$ pair to account for the Higgs fields.



$$|\Gamma| = 4 \text{ and } h^{1,1}(X_{7447}) = 5$$

- Horizontal axis represents the number of genetic episodes, in each episode a number of 90,000 states being visited

Searched for $N = 1$ supersymmetric $SU(5)$ GUTs with four additional Green- Schwarz anomalous $U(1)$ s, three $(\bar{5}, 10)$ generations, no exotic $\bar{10}$ multiplets and at least one vector-like $5-\bar{5}$ pair to account for the Higgs fields.



X_{4071} with $|\Gamma| = 2$ and $h^{1,1}(X_{4071}) = 7$

Searched for $N = 1$ supersymmetric $SU(5)$ GUTs with four additional Green-Schwarz anomalous $U(1)$ s, three $(\bar{5}, 10)$ generations, no exotic $\bar{10}$ multiplets and at least one vector-like $5-\bar{5}$ pair to account for the Higgs fields.

Manifold	h	$ \Gamma $	Range	GA	Scan	Found	Explored
7862	4	2	$[-7, 8]$	5	5	100%	10^{-10}
7862	4	4	$[-7, 8]$	30	31	97%	10^{-10}
7447	5	2	$[-7, 8]$	38	38	100%	10^{-14}
7447	5	4	$[-7, 8]$	139	154	90%	10^{-14}
5302	6	2	$[-7, 8]$	403	442	93%	10^{-19}
5302	6	4	$[-7, 8]$	722	897	80%	10^{-19}
4071	7	2	$[-3, 4]$	11,937	N/A	N/A	10^{-14}

- fraction of environment explored is tiny



***GAs in CYs and GAs versus
reinforcement learning***

GAs versus reinforcement learning

SAA, Constantin, Lukas, Harvey

- First comparison of GA versus RL in string context (NB techniques both work with “environment”)
- Consider monad bundles on Complete Intersection Calabi Yaus. [Kachru; Anderson; Anderson, He, Lukas; Anderson, Gray, He, Lukas; He, Lee, Lukas](#)
- Considered the following two kinds of CICY (*bi-cubic* and *triple trilinear* respectively) with configuration matrices, where indices are h11, h21, and Euler number:

$$\left[\begin{array}{c|c} \mathbb{P}^2 & 3 \\ \mathbb{P}^2 & 3 \end{array} \right]_{-162}^{2,83}, \quad \left[\begin{array}{c|ccc} \mathbb{P}^2 & 1 & 1 & 1 \\ \mathbb{P}^2 & 1 & 1 & 1 \\ \mathbb{P}^2 & 1 & 1 & 1 \end{array} \right]_{-90}^{3,48}$$

- Models constructed by monad bundles on the CICY defining the E8xE8 background: constructed from two line-bundle sums, B and C : in the end boils down to matrix of integers (where $k=1,\dots,h11$):

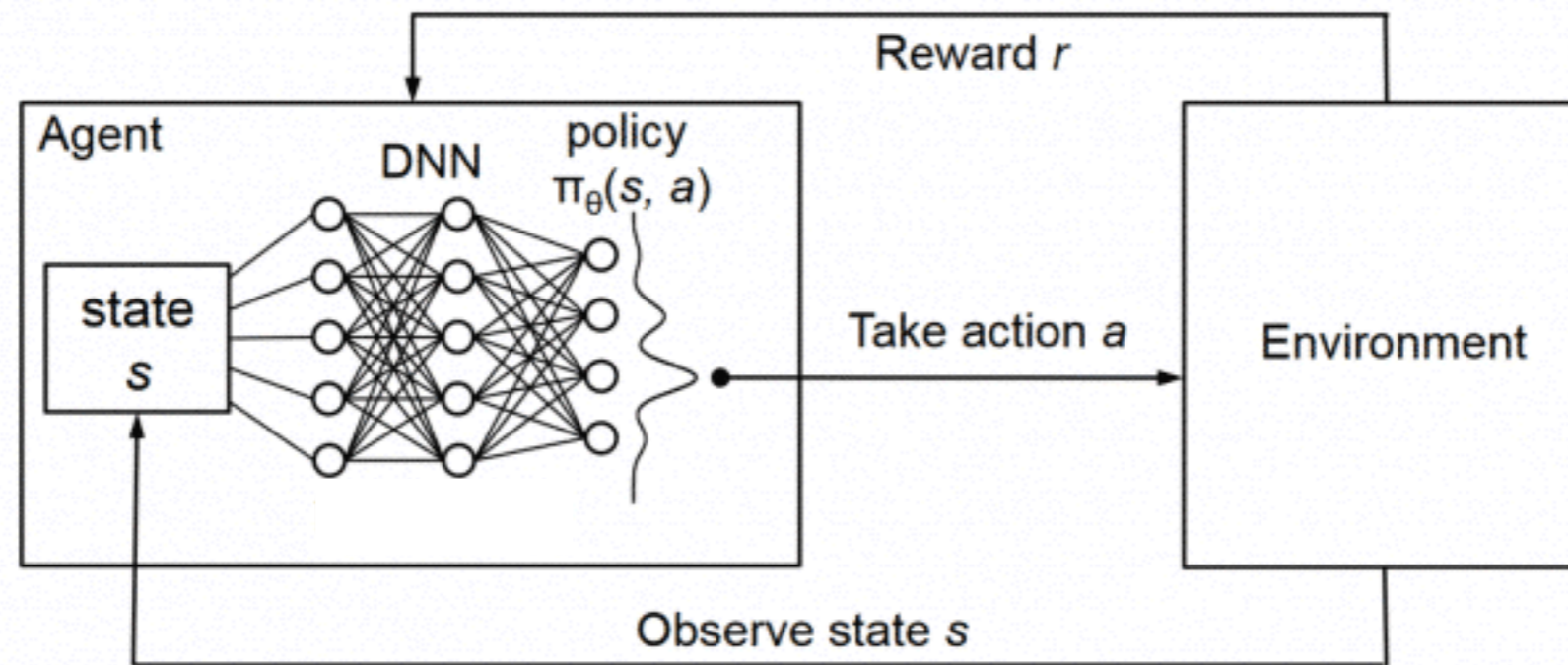
$$(b_1^k, \dots, b_{r_B}^k, c_1^k, \dots, c_{r_C}^k)$$

- Similarly all the phenomenological properties (e.g. number of generations) determined by these numbers via (several) index theorems. *Constantin, Lukas, Harvey*
- Search for “perfect-models” (aka “terminal states”): require SM-like theories (i.e. SO(10) GUT from broken E8, with 3 generations).
- *So what is the size of search space?* If we take $b_{\min} \leq b_i^k \leq b_{\max}$, $c_{\min} \leq c_a^k \leq c_{\max}$,
- Allowing say 10 values per entry, that is $10^{h^{1,1}(r_B+r_C-1)}$ with say $h^{1,1}=3$ it again becomes huge very quickly!
- For the GA we simply encode these integers as a single binary string and operate as before. Used quite large population = 250.
- In both RL and GA we use the same function to stand for the reward / fitness, based on the number of criteria that are satisfied.

Reinforcement learning vs GAs for these models

These models were already shown to be amenable to RL. Constantin, Lukas, Harvey

using REINFORCE ...



Reinforcement learning vs GAs for these models

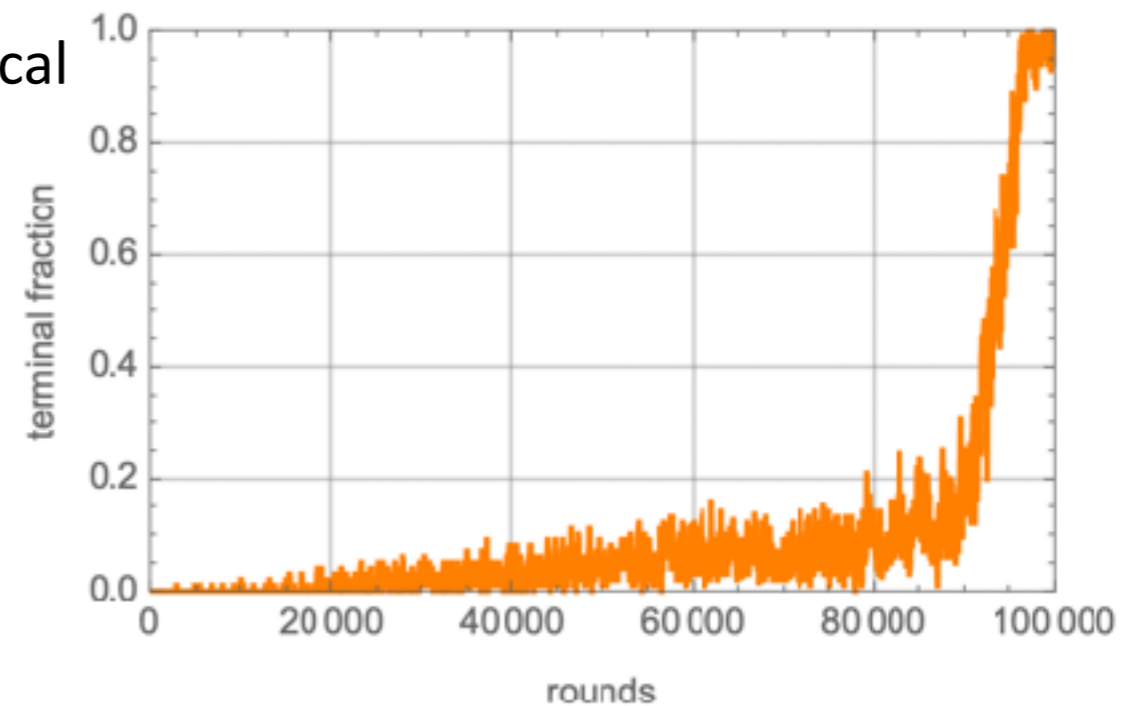
These models were already shown to be amenable to RL.

Constantin, Lukas, Harvey

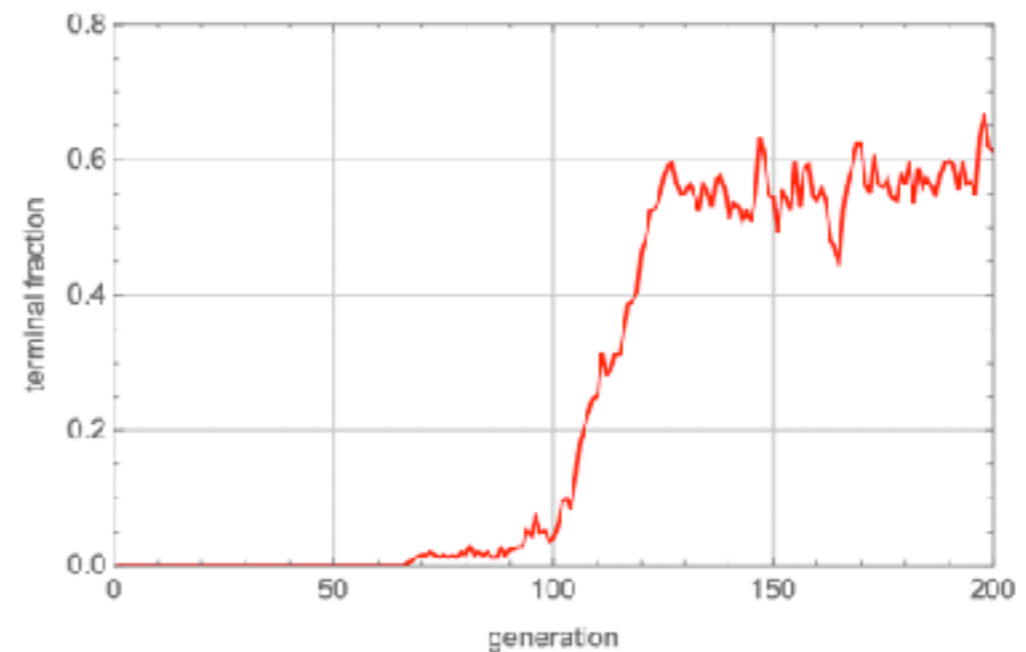
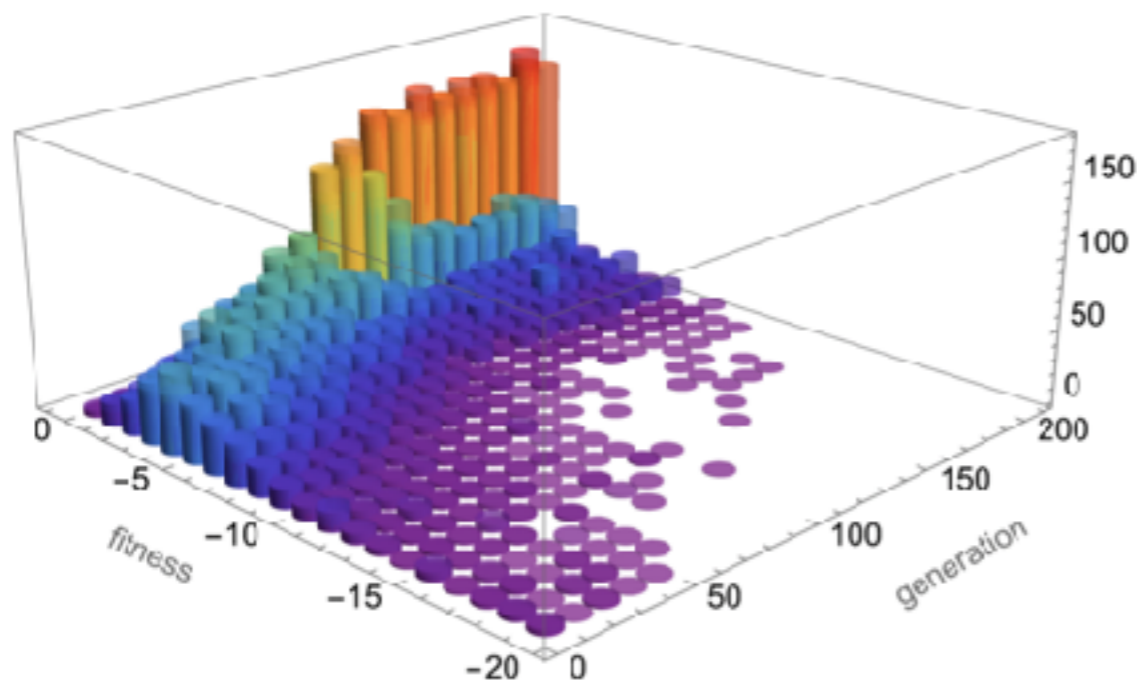
Find good performance after a long training time: typical run on the (6,2) bi-cubic with

$$b_{\min} = -3, c_{\min} = 0 \quad b_{\max} = 4, c_{\max} = 7.$$

for which the search space is $\simeq 4.4 \times 10^{12}$

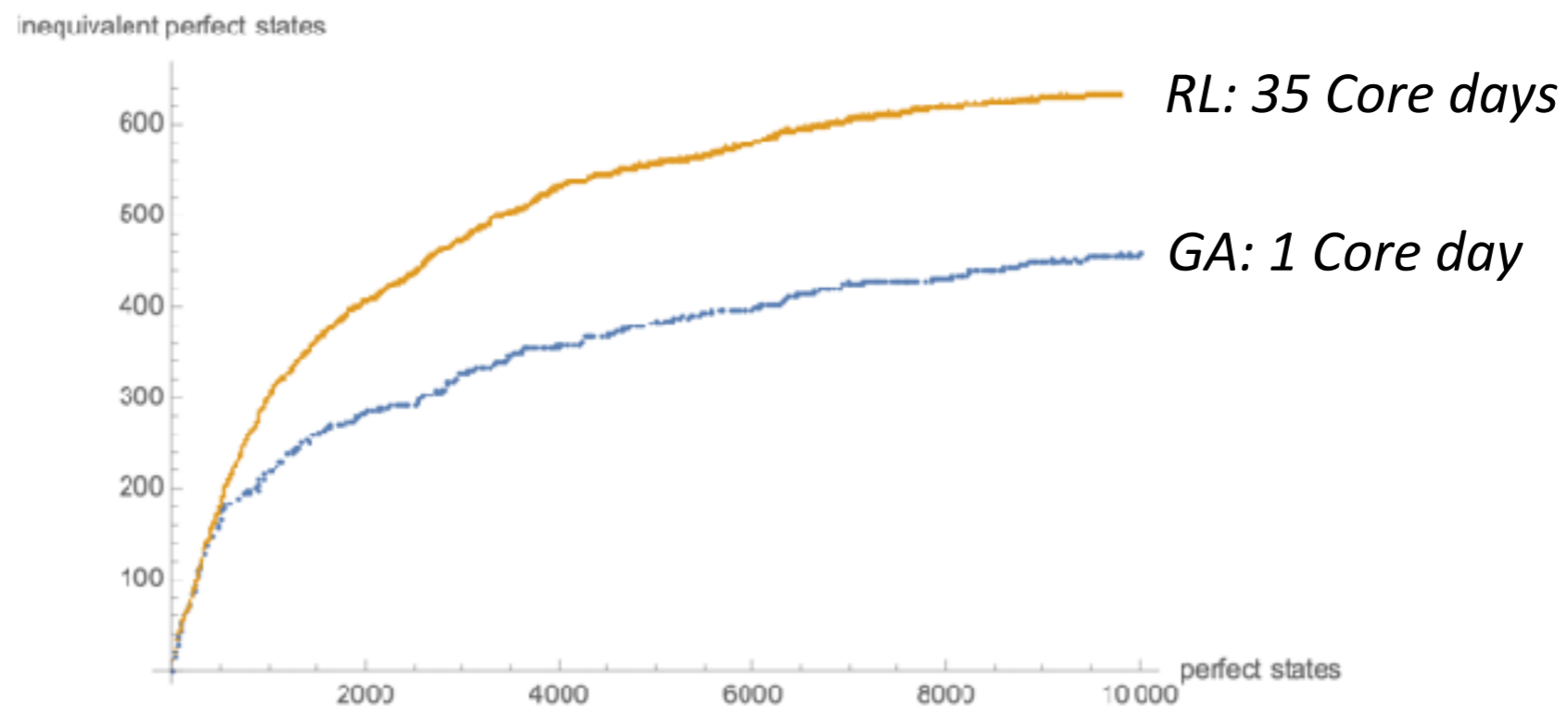


The GA is much faster to the first solutions! Note only 50K states visited:



Reinforcement learning vs GAs for these models

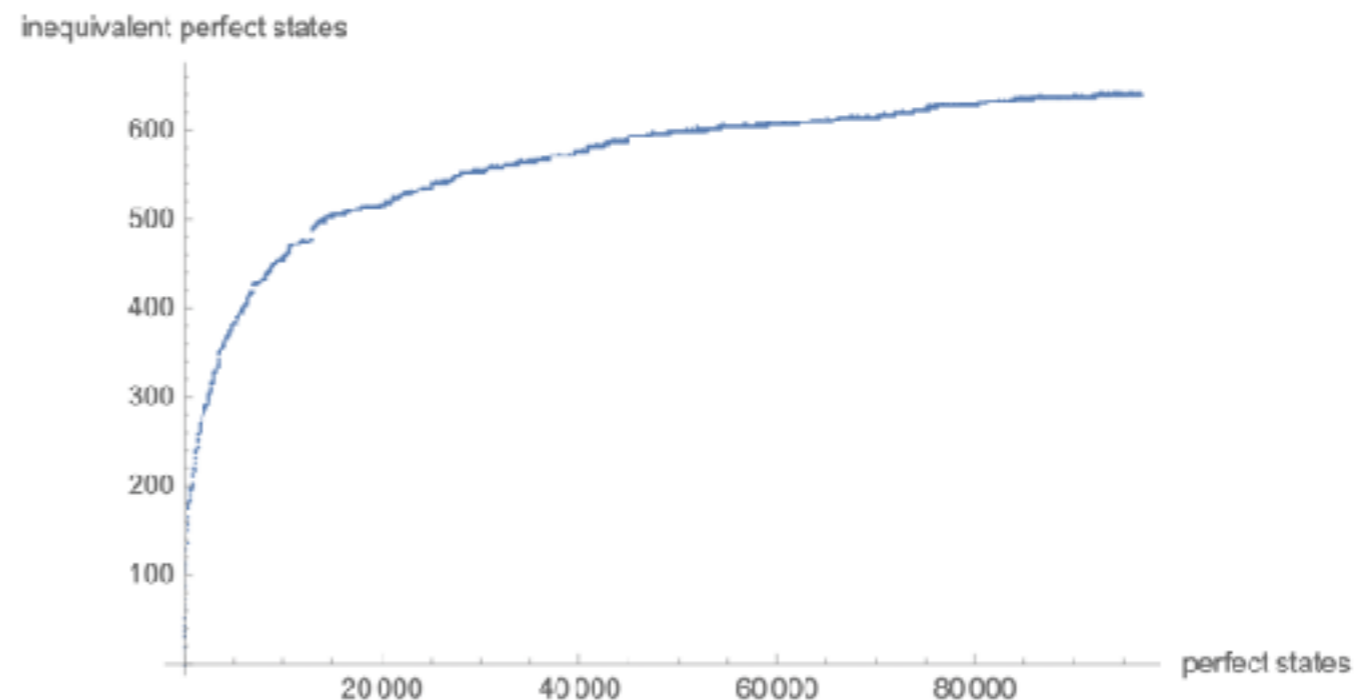
Redundancy: The methods behave differently. GA's tend to produce a lot of redundancy (equivalent perfect states) due to convergence, but are still more efficient:



Reinforcement learning vs GAs for these models

Saturation: after 35 core days the RL produced 643 inequivalent perfect states. After 10 core days the GA saturated at 639 inequivalent perfect states.

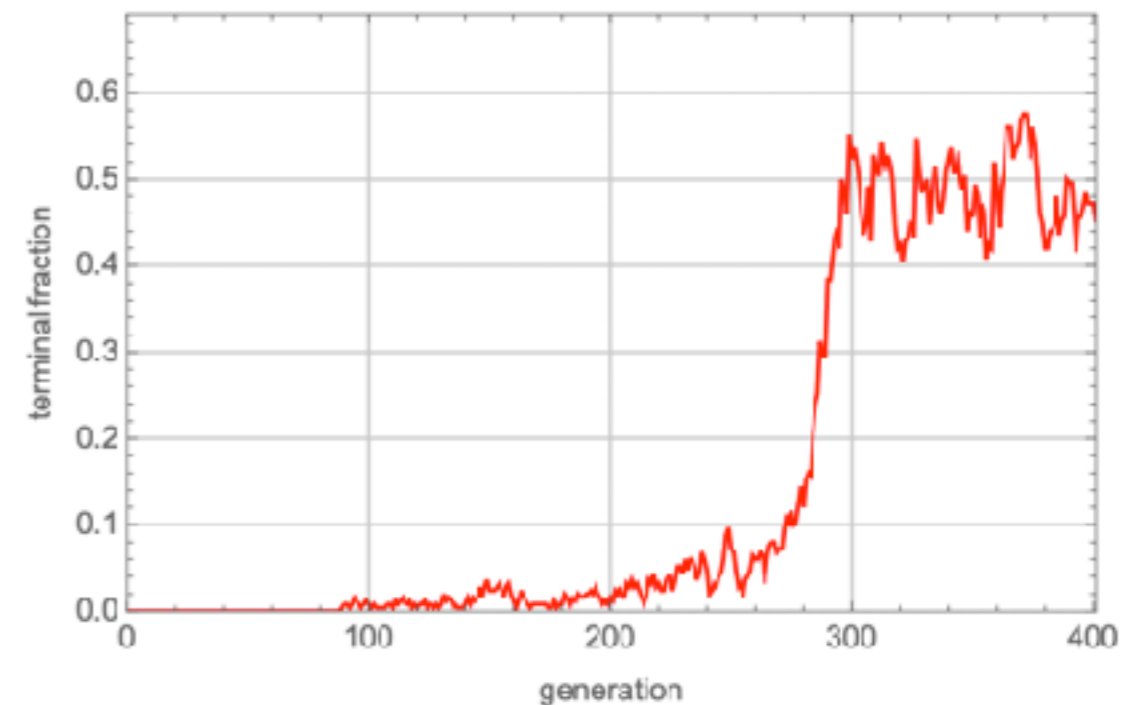
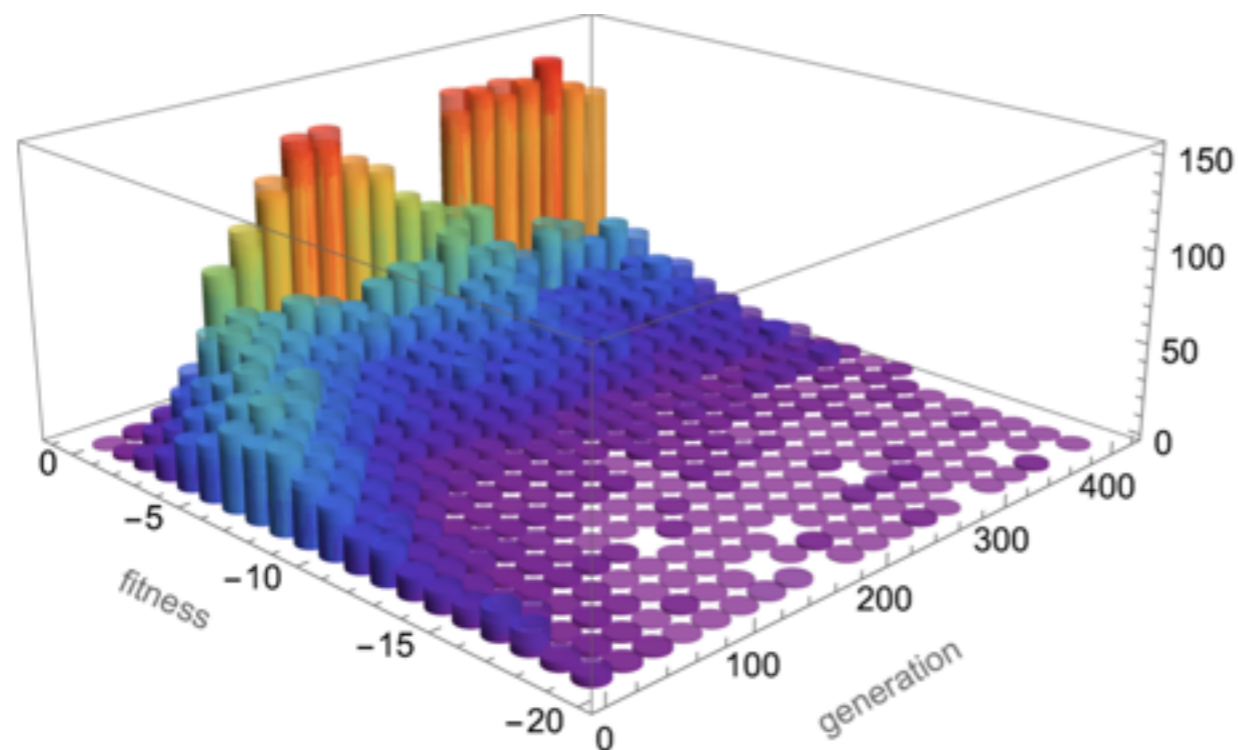
About 50 models in complement (i.e. 689 models in total)



NB: at the beginning they cover different regions (Sammon mapping), so an important side-effect is that we have evidence almost all possibilities are saturated, for this choice of hyper-parameters .

Reinforcement learning vs GAs for these models

(6,2) triple trilinear. Keep same domains of defining integers, but now $h_{11}=3$ gives search space $8^{21} \simeq 10^{19}$ is seven orders of magnitude larger.



GA in a given run takes only twice as many generations to reach the saturated fitness.

A glowing blue DNA double helix structure is shown against a dark blue background. The helix is composed of two thick, curved strands connected by several vertical rungs. The strands and rungs are illuminated from within, creating a bright blue glow that fades into the darker background. The overall effect is futuristic and scientific.

Genetic Quantum Annealing

Genetic quantum annealing (GQA)

Recall flow-chart for Genetic Algorithms

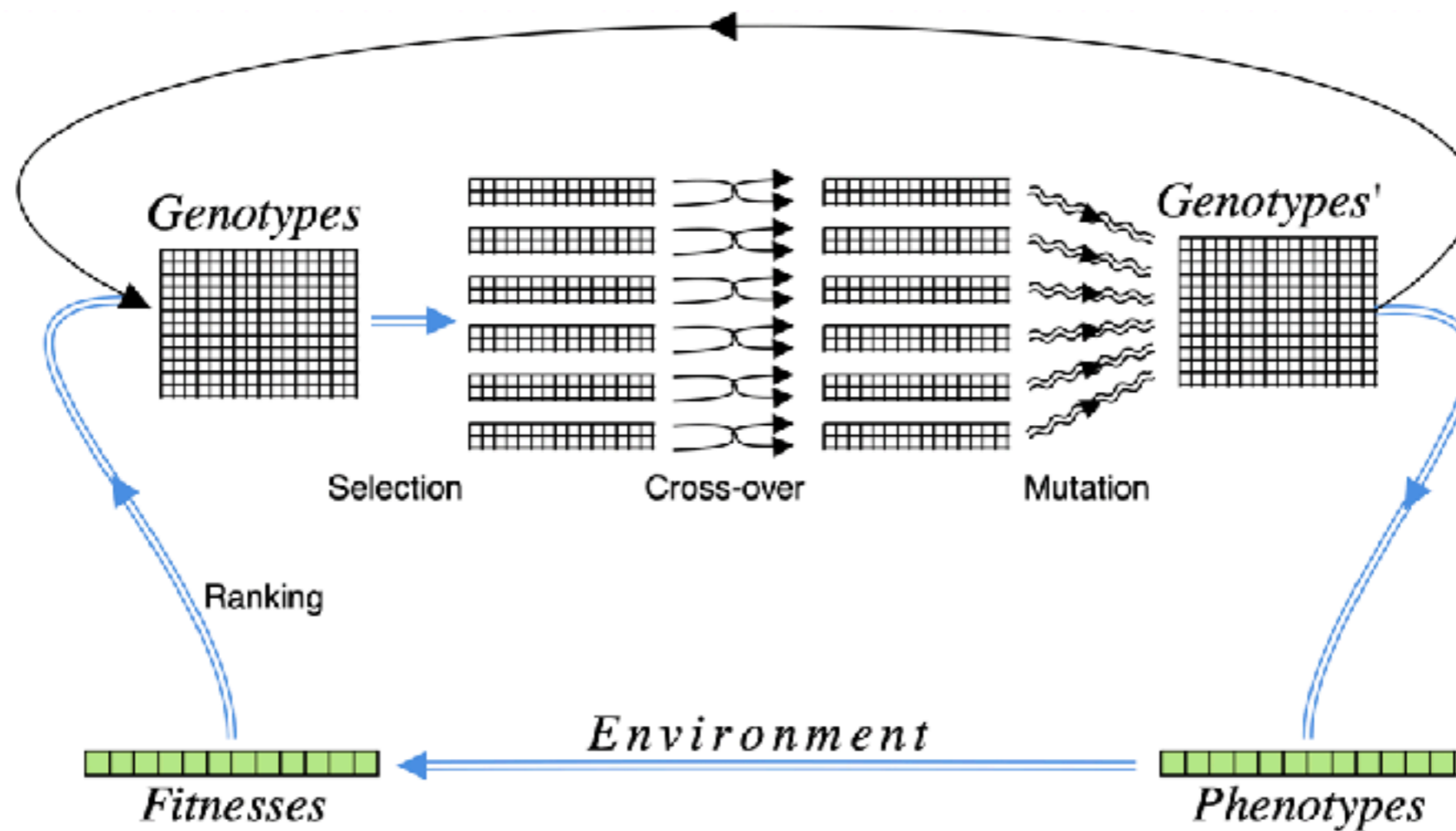


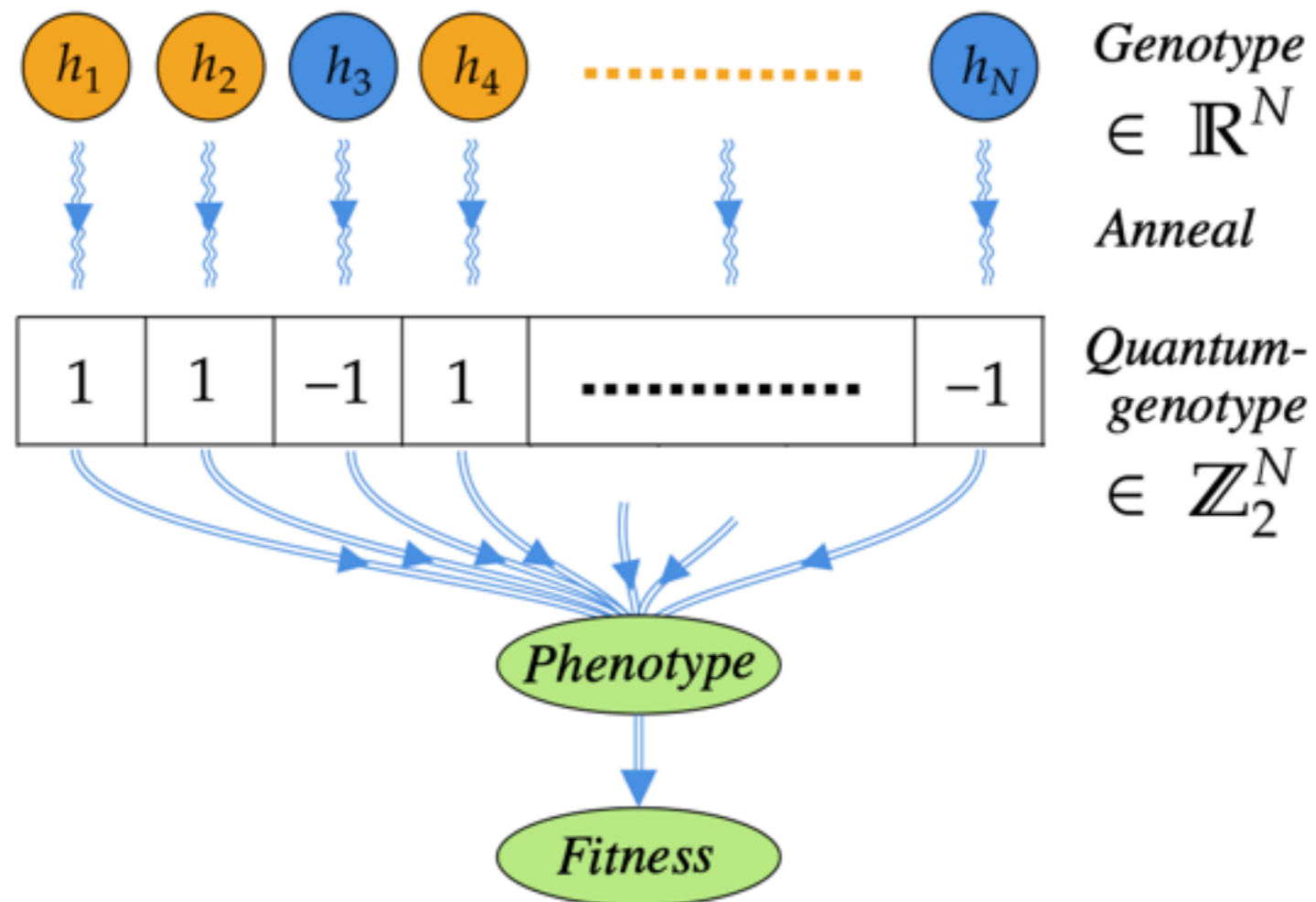
Diagram for classical GA.

How can we take advantage of Quantum Annealing without Ising encoding the entire environment?

SAA, Nutricati, Spannowsky, quant-ph [2209.07455](#)

Annealer allows us to instead define a population of continuous genotypes in the h :

“Nepotism”



Work with continuous genotypes in the h :

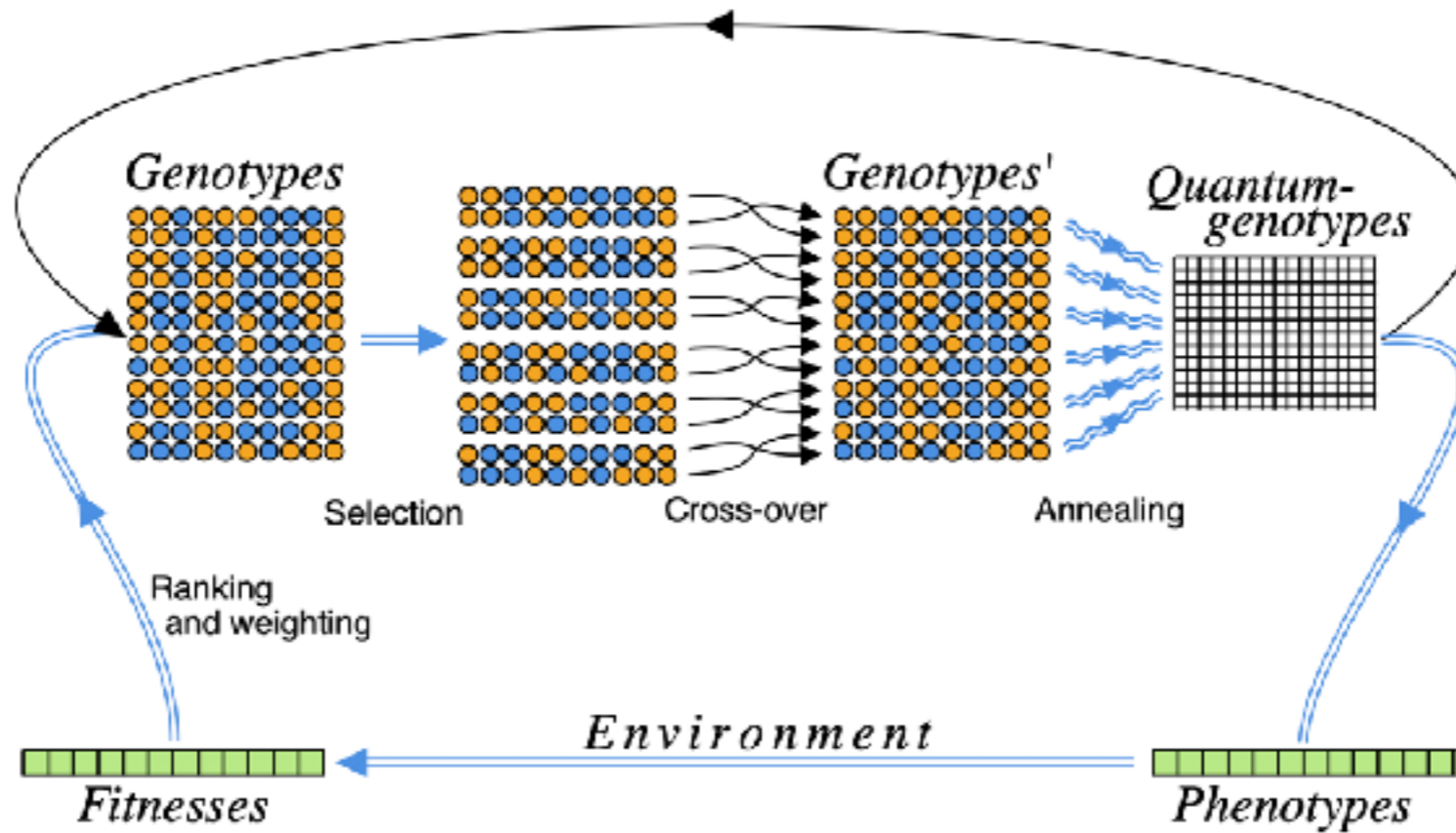


Diagram for GQAA.

Work with continuous genotypes in the h :

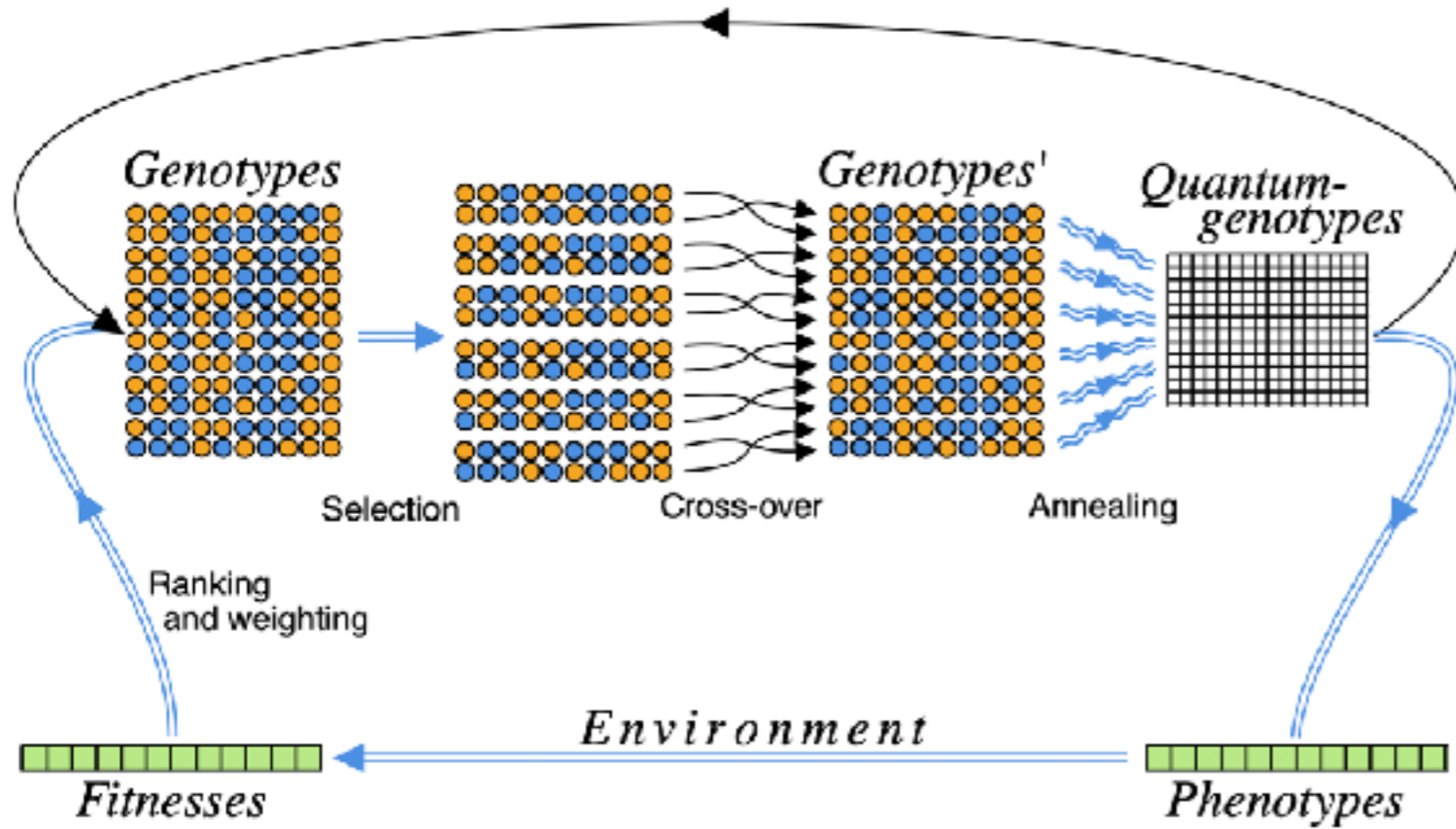
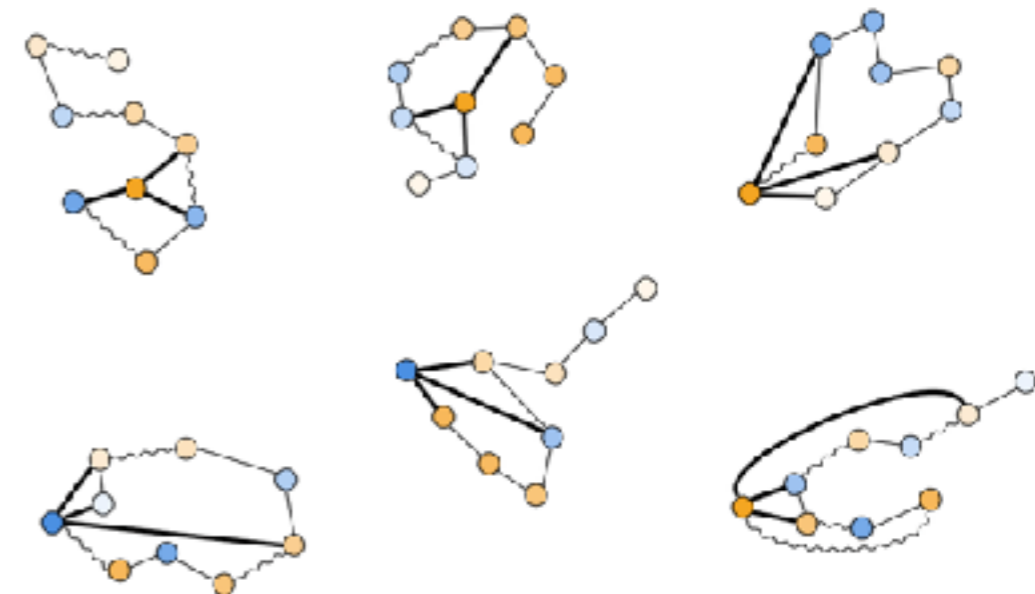


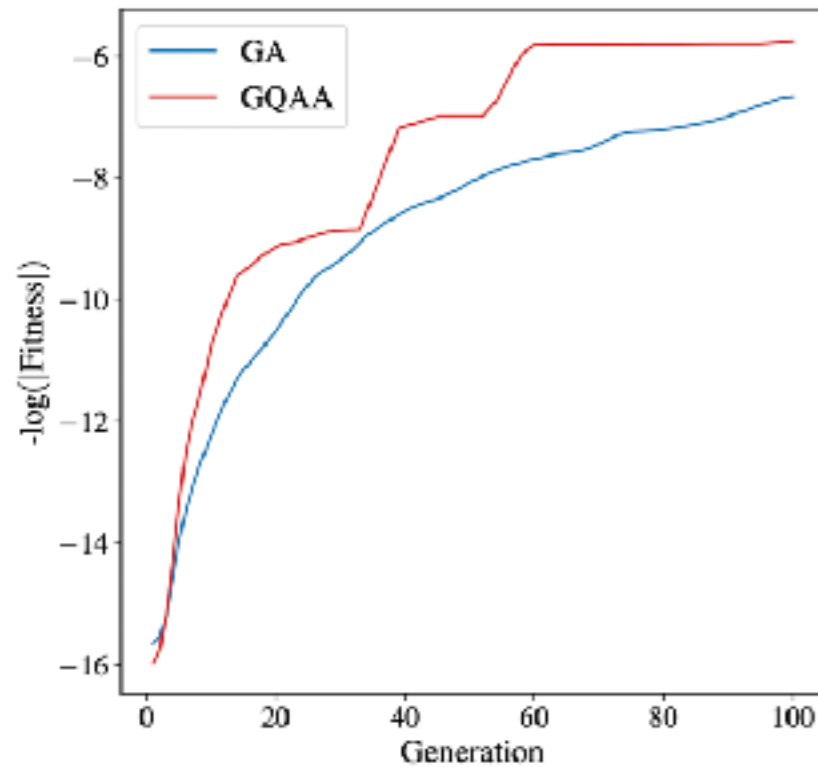
Diagram for GQAA.

Connect the population together using the J couplings: example topology

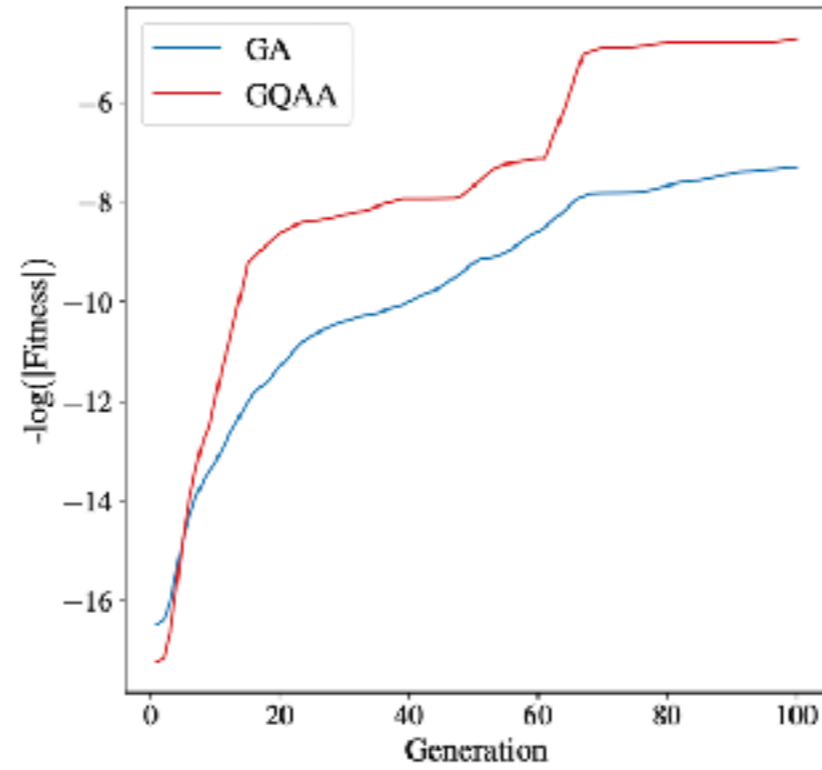


"Polyandry"

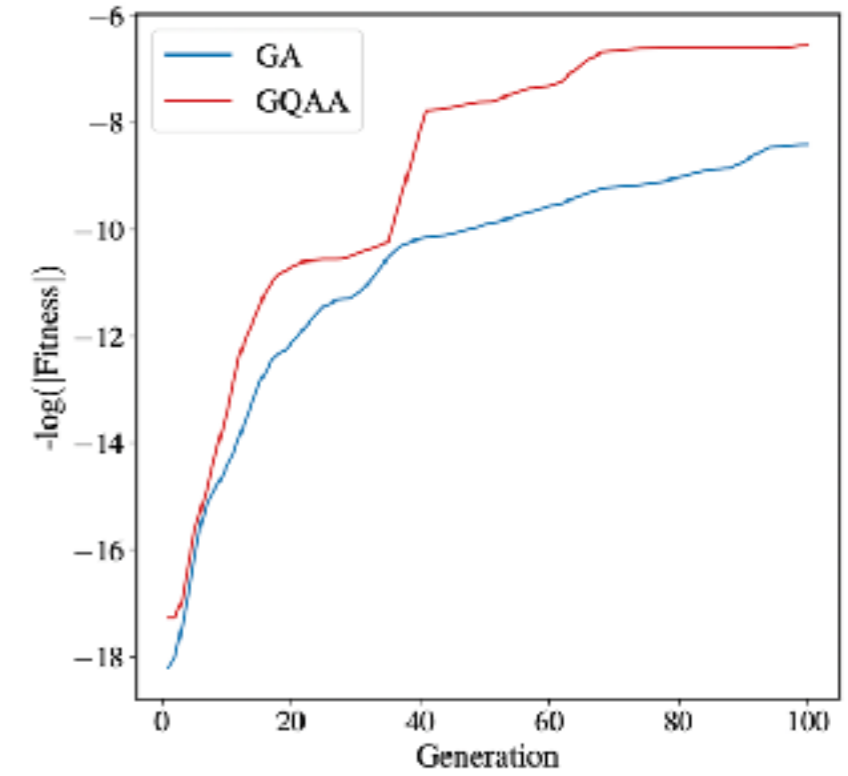
Results



(a) Taxicab (3,6,6)



(b) Taxicab (3,7,7)



(c) Taxicab (3,8,8)

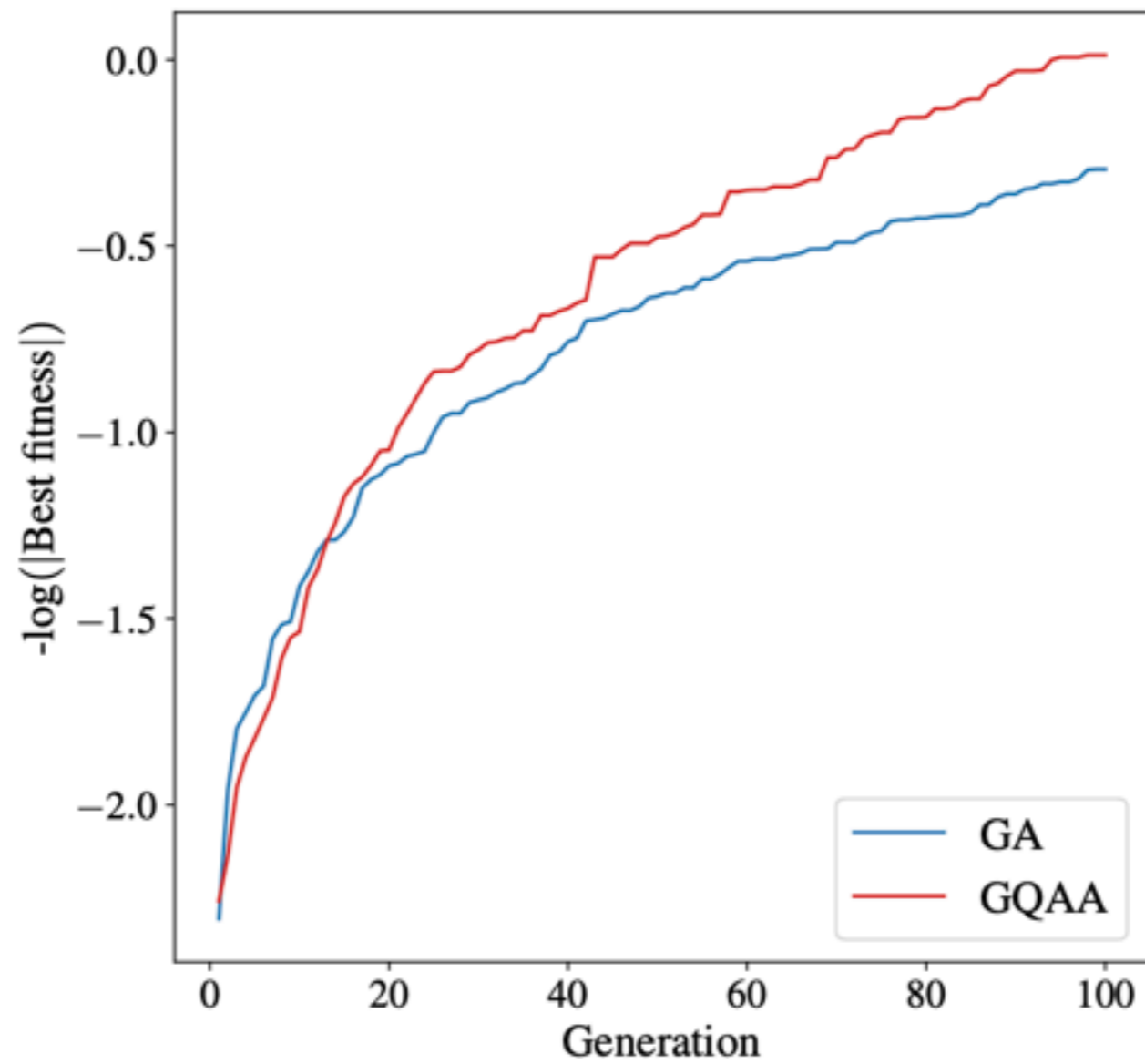
Notation: $(k, m, n) \equiv x_1^k + \dots + x_m^k = y_1^k + \dots + y_n^k$

SAA, Nutricati, Spannowsky, 2022

Results for line bundle models

SAA, Constantin, Lukas, Harvey, Nutricati

X_{5302}



Conclusions ...

- GA's are a strikingly effective search tool for finding favourable string vacua
- Note we did not yet work hard to optimise the GA in the RL/GA study. (Just single point cross-over, not particularly optimising mutation rate, no creep mutation etc)
- Results suggest estimates of string-landscape size are less meaningful than fitness-distance correlation (i.e. SM is not a needle in a haystack)
- QAs (and also simulated annealing) are strikingly similar to string configurations.
- However imposing phenomenological constraints directly can be more tricky and require post-pre-processing for them
- Combined approaches, GA+Clustering+Nelder Mead or QGAA hold promise.
- Adiabatic Quantum Computing has not yet seen much application