An Evolutionary Model with Turing Machines

G. Feverati

LAPTH, Annecy-le-Vieux

joint work with F. Musso, Universidad de Burgos

Phys. Rev. E **77**, 061901, arXiv:0711.3580 [q-bio.QM]

Summary

- coding/non-coding DNA
- Darwinian evolution
- Turing machines
- our model: in silico evolution
- analysis of the results

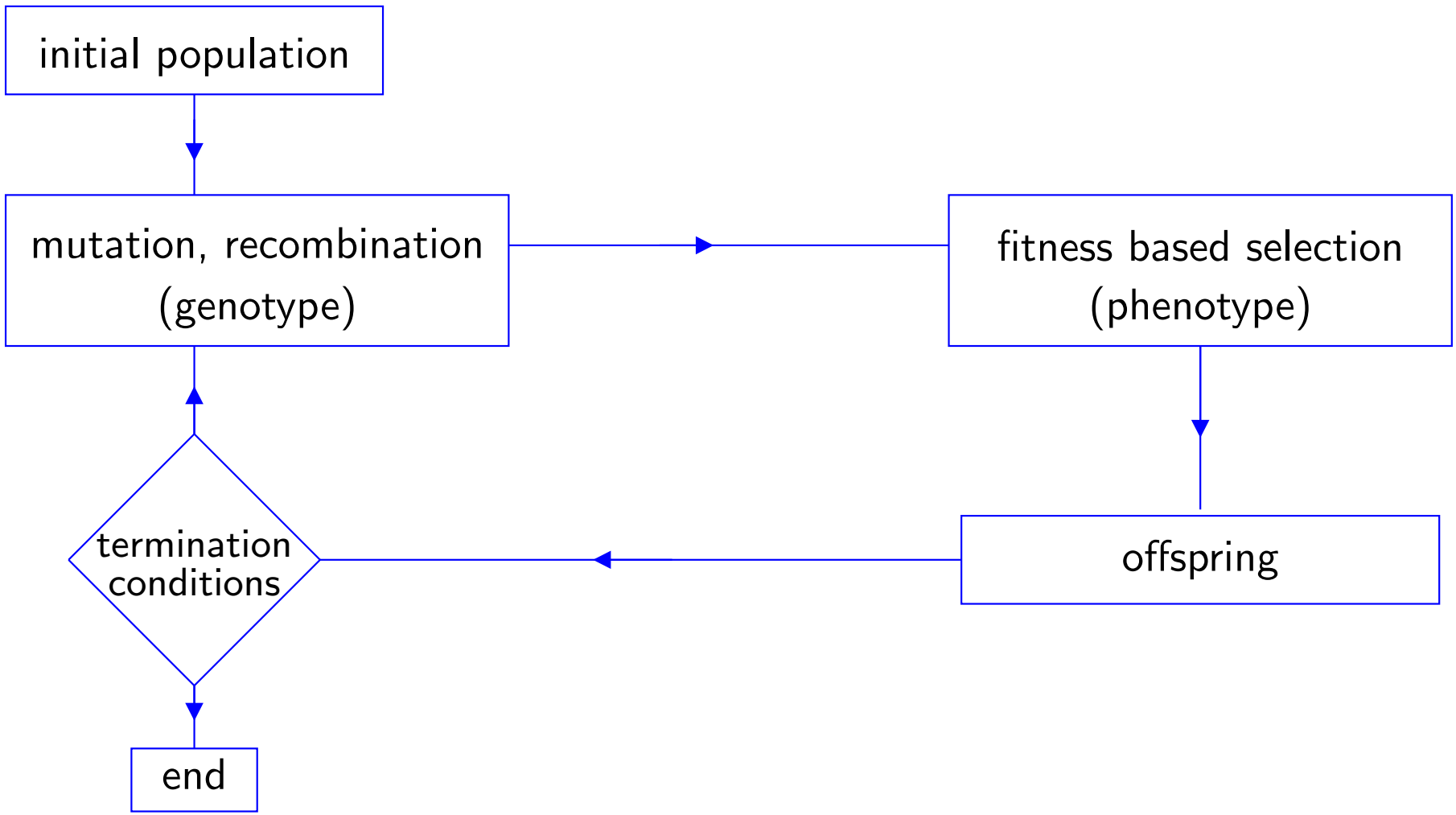
Darwinian evolution

A population (of organisms, objects, agents ...) survives for a limited time and then dies. Some members produce offspring for succeeding generations, the fittest ones tend to produce more.

Over many generations, the make-up of the population changes, in some sense adapts to the environmental conditions.

“In-silico”

- artificial life, evolutionary computations
- population of algorithms: in-silico organisms
- genotype / phenotype



Motivations

- Eukaryotes accumulate non-coding DNA
- Having large non-coding DNA is likely a disvantage for the organism (metabolic cost, DNA enlargement is often destructive)
- If there are parts of DNA not controlled by selection, they can be freely mutated so they can serve as raw material for new genes.

Can this long term advantage balance the short term disadvantage?
(Selection acts on the short term only!)

Abstract model: study of the evolution of a population of algorithms

Aim: Quantify the long term advantage of having useless parts of code.

What is a Turing Machine?

Turing machines are (abstract) symbol-manipulating devices that implement a "one-point" discret evolution law:

$$T(x,t) \xrightarrow{\text{evolution law}} T(x,t+1) \quad \forall \quad x$$

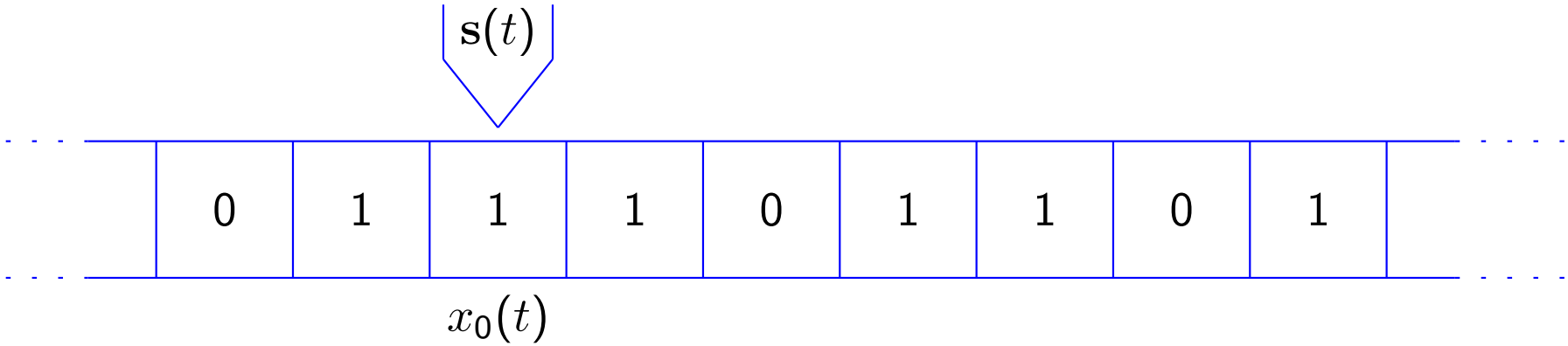
space $x \in \mathbb{Z}$, time $t \in \mathbb{N}$, tape configuration $T(x,t) \in \{0,1\}$

such that

only a single mutated position $x_0(t)$ can exist at each time

$$T(x,t+1) = T(x,t) \quad \forall\, x \neq x_0(t)$$

$$T(x_0(t),t+1) \text{ could be } \neq T(x_0(t),t)$$



- $T(x,t)$ = tape divided into cells (containing the symbols 1 or 0)
- Turing machine (TM) = list of internal states $\{s_i, \forall\ i\}$ = potential actions

At each step, TM performs three actions determined by the value read on the tape at the position $x_0(t)$ and by its internal state $\mathbf{s}(t)$:

1. **write:** write a new symbol at position $x_0(t)$,
2. **displace:** move right or left by one cell or keep still
3. **call:** change its internal state.

"Transition table" of triplets **write-displace-call** for the two possible inputs 0 or 1

Example: sum of two numbers

Turing machine that performs the sum of two positive integers

read \ **state**	**1**	**2**	**3**
0	1 – Right – **2**	0 – Left – **3**	_ – _ – _
1	1 – Right – **1**	1 – Right – **2**	0 – _ – **Halt**

Example: sum of two numbers

Turing machine that performs the sum of two positive integers

read \ state	**1**	**2**	**3**
0	1 – Right – **2**	0 – Left – **3**	_ – _ – _
1	1 – Right – **1**	1 – Right – **2**	0 – _ – **Halt**

1
1 1 1 0 1 1 0 . . . $\xrightarrow{\text{1-Right-}\mathbf{1}}$

Example: sum of two numbers

Turing machine that performs the sum of two positive integers

read \ state	**1**	**2**	**3**
0	1 – Right – **2**	0 – Left – **3**	_ – _ – _
1	1 – Right – **1**	1 – Right – **2**	0 – _ – **Halt**

$$\begin{array}{c}\mathbf{1}\\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ldots\end{array} \xrightarrow{\text{1-Right-}\mathbf{1}} \begin{array}{c}\mathbf{1}\\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ldots\end{array} \xrightarrow{\text{1-Right-}\mathbf{1}}$$

Example: sum of two numbers

Turing machine that performs the sum of two positive integers

read \ state	**1**	**2**	**3**
0	1 – Right – **2**	0 – Left – **3**	_ – _ – _
1	1 – Right – **1**	1 – Right – **2**	0 – _ – **Halt**

$$\begin{array}{c} \mathbf{1} \\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ldots \end{array} \xrightarrow{\text{1-Right-}\mathbf{1}} \begin{array}{c} \mathbf{1} \\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ldots \end{array} \xrightarrow{\text{1-Right-}\mathbf{1}}$$

$$\begin{array}{c} \mathbf{1} \\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ldots \end{array} \xrightarrow{\text{1-Right-}\mathbf{1}}$$

Example: sum of two numbers

Turing machine that performs the sum of two positive integers

read \ state	**1**	**2**	**3**
0	1 – Right – **2**	0 – Left – **3**	_ – _ – _
1	1 – Right – **1**	1 – Right – **2**	0 – _ – **Halt**

$$\mathbf{1}\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ldots \xrightarrow{\text{1-Right-}\mathbf{1}} 1\ \mathbf{1}\ 1\ 1\ 0\ 1\ 1\ 0\ldots \xrightarrow{\text{1-Right-}\mathbf{1}}$$

$$1\ 1\ \mathbf{1}\ 1\ 0\ 1\ 1\ 0\ldots \xrightarrow{\text{1-Right-}\mathbf{1}} 1\ 1\ 1\ \mathbf{1}\ 0\ 1\ 1\ 0\ldots \xrightarrow{\text{1-Right-}\mathbf{2}}$$

Example: sum of two numbers

Turing machine that performs the sum of two positive integers

read \ state	**1**	**2**	**3**
0	1 – Right – **2**	0 – Left – **3**	_ – _ – _
1	1 – Right – **1**	1 – Right – **2**	0 – _ – **Halt**

$$\mathbf{1}\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ldots \xrightarrow{\text{1-Right-}\mathbf{1}} 1\ \mathbf{1}\ 1\ 1\ 0\ 1\ 1\ 0\ldots \xrightarrow{\text{1-Right-}\mathbf{1}}$$

$$1\ 1\ \mathbf{1}\ 1\ 0\ 1\ 1\ 0\ldots \xrightarrow{\text{1-Right-}\mathbf{1}} 1\ 1\ 1\ \mathbf{1}\ 0\ 1\ 1\ 0\ldots \xrightarrow{\text{1-Right-}\mathbf{2}}$$

$$1\ 1\ 1\ 1\ \mathbf{2}\ 1\ 0\ldots \xrightarrow{\text{1-Right-}\mathbf{2}}$$

Example: sum of two numbers

Turing machine that performs the sum of two positive integers

read \ state	**1**	**2**	**3**
0	1 – Right – **2**	0 – Left – **3**	_ – _ – _
1	1 – Right – **1**	1 – Right – **2**	0 – _ – **Halt**

$$\mathbf{1}\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ldots \xrightarrow{\text{1-Right-}\mathbf{1}} 1\ \mathbf{1}\ 1\ 1\ 0\ 1\ 1\ 0\ldots \xrightarrow{\text{1-Right-}\mathbf{1}}$$

$$1\ 1\ \mathbf{1}\ 1\ 0\ 1\ 1\ 0\ldots \xrightarrow{\text{1-Right-}\mathbf{1}} 1\ 1\ 1\ \mathbf{1}\ 0\ 1\ 1\ 0\ldots \xrightarrow{\text{1-Right-}\mathbf{2}}$$

$$1\ 1\ 1\ 1\ \mathbf{2}\ 1\ 1\ 0\ldots \xrightarrow{\text{1-Right-}\mathbf{2}} 1\ 1\ 1\ 1\ 1\ \mathbf{2}\ 1\ 0\ldots \xrightarrow{\text{1-Right-}\mathbf{2}}$$

Example: sum of two numbers

Turing machine that performs the sum of two positive integers

read \ state	**1**	**2**	**3**
0	1 – Right – **2**	0 – Left – **3**	_ – _ – _
1	1 – Right – **1**	1 – Right – **2**	0 – _ – **Halt**

$$\mathbf{1}\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ldots \xrightarrow{\text{1-Right-}\mathbf{1}} 1\ \mathbf{1}\ 1\ 1\ 0\ 1\ 1\ 0\ldots \xrightarrow{\text{1-Right-}\mathbf{1}}$$

$$1\ 1\ \mathbf{1}\ 1\ 0\ 1\ 1\ 0\ldots \xrightarrow{\text{1-Right-}\mathbf{1}} 1\ 1\ 1\ \mathbf{1}\ 0\ 1\ 1\ 0\ldots \xrightarrow{\text{1-Right-}\mathbf{2}}$$

$$1\ 1\ 1\ 1\ \mathbf{2}\ 1\ 1\ 0\ldots \xrightarrow{\text{1-Right-}\mathbf{2}} 1\ 1\ 1\ 1\ 1\ \mathbf{2}\ 1\ 0\ldots \xrightarrow{\text{1-Right-}\mathbf{2}}$$

$$1\ 1\ 1\ 1\ 1\ 1\ \mathbf{2}\ 0\ldots \xrightarrow{\text{0-Left-}\mathbf{3}}$$

Example: sum of two numbers

Turing machine that performs the sum of two positive integers

read \ state	**1**	**2**	**3**
0	1 – Right – **2**	0 – Left – **3**	_ – _ – _
1	1 – Right – **1**	1 – Right – **2**	0 – _ – **Halt**

$$\mathbf{1}\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ldots \xrightarrow{\text{1-Right-}\mathbf{1}} 1\ \mathbf{1}\ 1\ 1\ 0\ 1\ 1\ 0\ldots \xrightarrow{\text{1-Right-}\mathbf{1}}$$

$$1\ 1\ \mathbf{1}\ 1\ 0\ 1\ 1\ 0\ldots \xrightarrow{\text{1-Right-}\mathbf{1}} 1\ 1\ 1\ \mathbf{1}\ 0\ 1\ 1\ 0\ldots \xrightarrow{\text{1-Right-}\mathbf{2}}$$

$$1\ 1\ 1\ 1\ \mathbf{2}\ 1\ 1\ 0\ldots \xrightarrow{\text{1-Right-}\mathbf{2}} 1\ 1\ 1\ 1\ 1\ \mathbf{2}\ 1\ 0\ldots \xrightarrow{\text{1-Right-}\mathbf{2}}$$

$$1\ 1\ 1\ 1\ 1\ 1\ \mathbf{2}\ 0\ldots \xrightarrow{\text{0-Left-}\mathbf{3}} 1\ 1\ 1\ 1\ 1\ \mathbf{3}\ 1\ 0\ldots \xrightarrow{\text{0-\_-}\mathbf{Halt}}$$

Example: sum of two numbers

Turing machine that performs the sum of two positive integers

read \ state	**1**	**2**	**3**
0	1 – Right – **2**	0 – Left – **3**	_ – _ – _
1	1 – Right – **1**	1 – Right – **2**	0 – _ – **Halt**

$$\mathbf{1}\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ldots \xrightarrow{\text{1-Right-}\mathbf{1}} 1\ \mathbf{1}\ 1\ 1\ 0\ 1\ 1\ 0\ldots \xrightarrow{\text{1-Right-}\mathbf{1}}$$

$$1\ 1\ \mathbf{1}\ 1\ 0\ 1\ 1\ 0\ldots \xrightarrow{\text{1-Right-}\mathbf{1}} 1\ 1\ 1\ \mathbf{1}\ 0\ 1\ 1\ 0\ldots \xrightarrow{\text{1-Right-}\mathbf{2}}$$

$$1\ 1\ 1\ 1\ \mathbf{2}\ 1\ 1\ 0\ldots \xrightarrow{\text{1-Right-}\mathbf{2}} 1\ 1\ 1\ 1\ 1\ \mathbf{2}\ 1\ 0\ldots \xrightarrow{\text{1-Right-}\mathbf{2}}$$

$$1\ 1\ 1\ 1\ 1\ 1\ \mathbf{2}\ 0\ldots \xrightarrow{\text{0-Left-}\mathbf{3}} 1\ 1\ 1\ 1\ 1\ \mathbf{3}\ 1\ 0\ldots \xrightarrow{\text{0-\_-}\mathbf{Halt}}$$

$$1\ 1\ 1\ 1\ 1\ 0\ 0\ldots$$

Our model

Initial population: 300 TM with 1-state, of the form

	0
0	0-Still-Halt
1	1-Still-Halt

and let them evolve for thousands of generations.

At each generation every TM undergoes four processes:

1. States-increase: with a rate p_{i} the TM passes from N to N+1 states by the addition of the further state

	N
0	0-Still-Halt
1	1-Still-Halt

2. Mutation: every value inside the transition tables of every state can be modified with a rate p_{m}.
3. Fitness assignment.
4. Selection and reproduction.

Fitness evaluation: goal tape

Fitness scores the phenotype: measure of the adaptation to the environment namely of the capability to survive and reproduce.

$$\text{input tape} = 0\ 0\ 0\ 0\ 0\ 0\ 0\ \ldots \xrightarrow{\text{TM}} \text{output tape}$$

Halt conditions:

1. the machine enters the halt state,
2. the head goes out of the tape,
3. the machine has run for 4000 time steps

Goal tape (ideal performance):

primes	1 1 1 0 1 0 1 0 0 0 1 0 1 0 0 0 1 0 1 0 ...
$(\pi - 3)_{\text{bin}}$	0 0 1 0 0 1 0 0 0 0 1 1 1 1 1 1 0 1 1 0 ...

Fitness = how close the output tape is to the goal tape:

+1 for every 1 placed on a good position,
−3 for every 1 placed on a bad position.

Selection and reproduction

(tournament selection, rank based)

Two TM are randomly extracted from the old population

$T_1\ T_2$

The fitnessess are compared, the machine which scores higher is copied on the other one that is eliminated (asexual reproduction). If the fitness are equal, nothing happens.

$T_1\ T_1 \quad T_1\ T_2 \quad T_2\ T_2$

The two TM are eliminated from the old population and placed in the new population and the process restarts until exhaustion of the old population

$T_1'\ \ T_2'$

Example

	0	1	2	3	4	5	6	7
0	1-R-3	1-R-4	1-R-5	1-R-2	1-R-6	0-R-1	1-L-6	0-S-7
1	1-S-2	1-S-1	0-S-2	0-S-4	0-L-3	1-L-0	0-L-H	0-R-1

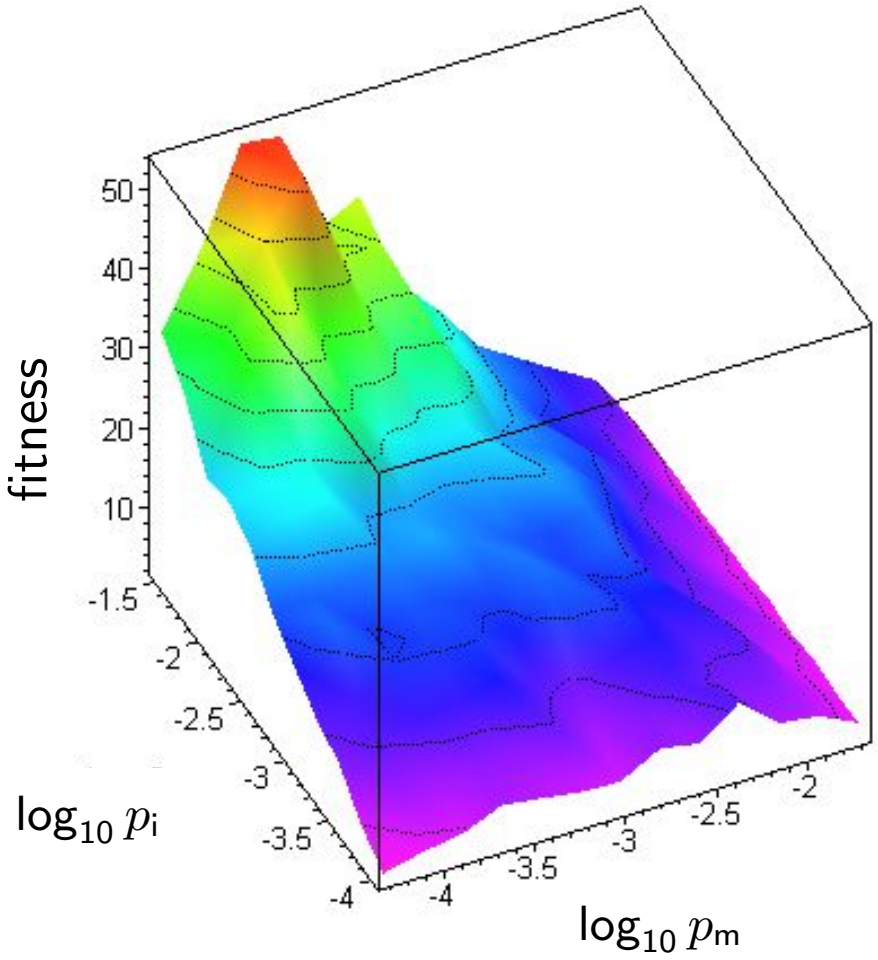
H

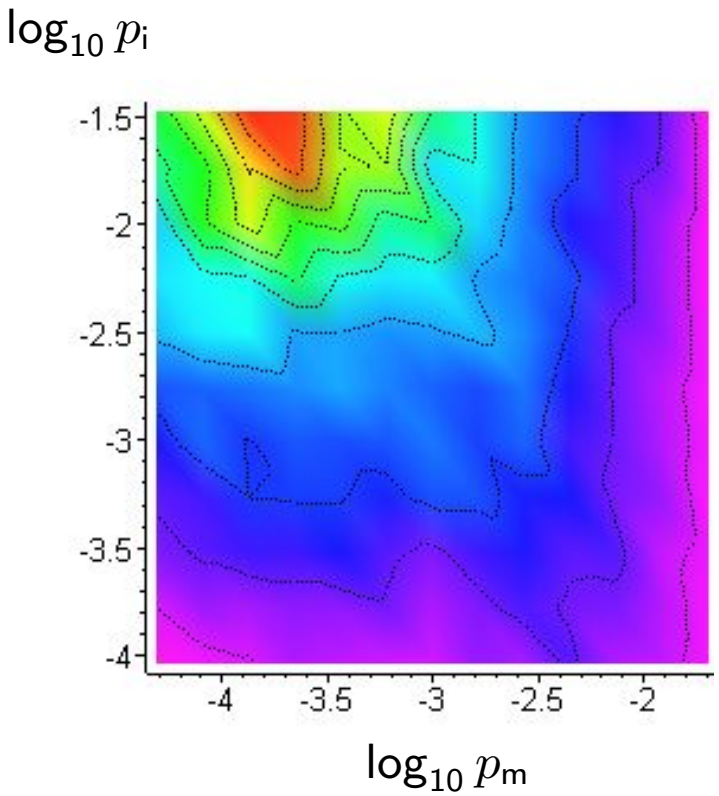
1 1 1 0 1 0 1 0 . . .

Fitness $= 5$, Red $=$ coding triplets, N. states $= 8$, N. coding triplets $= 8$, N. non-coding triplets $= 8$.
Obtained with $p_1 = 1/1009$, $p_2 = 1/3333$ after 20000 generations.

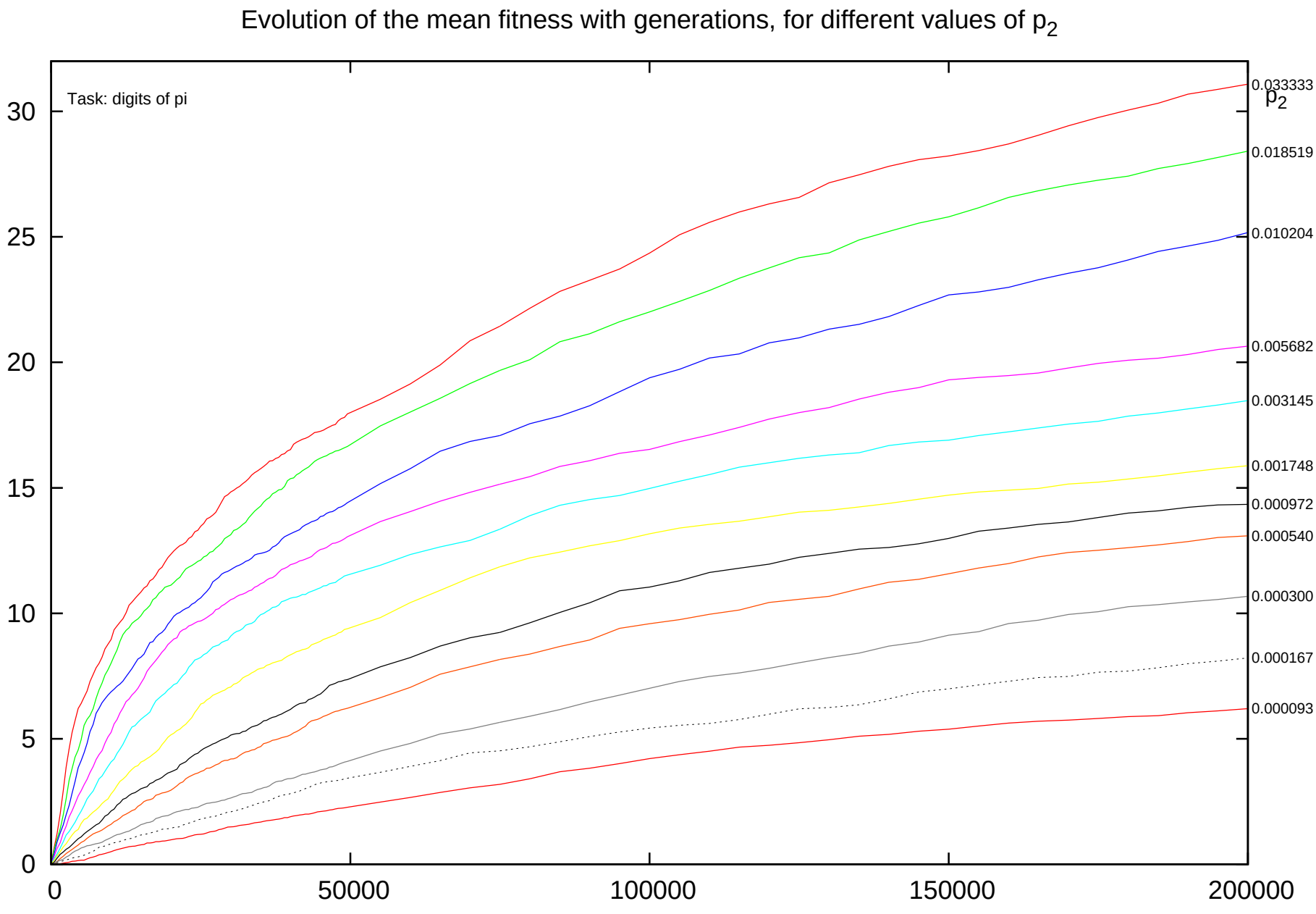
Results

Goal tape: decimal digits of π



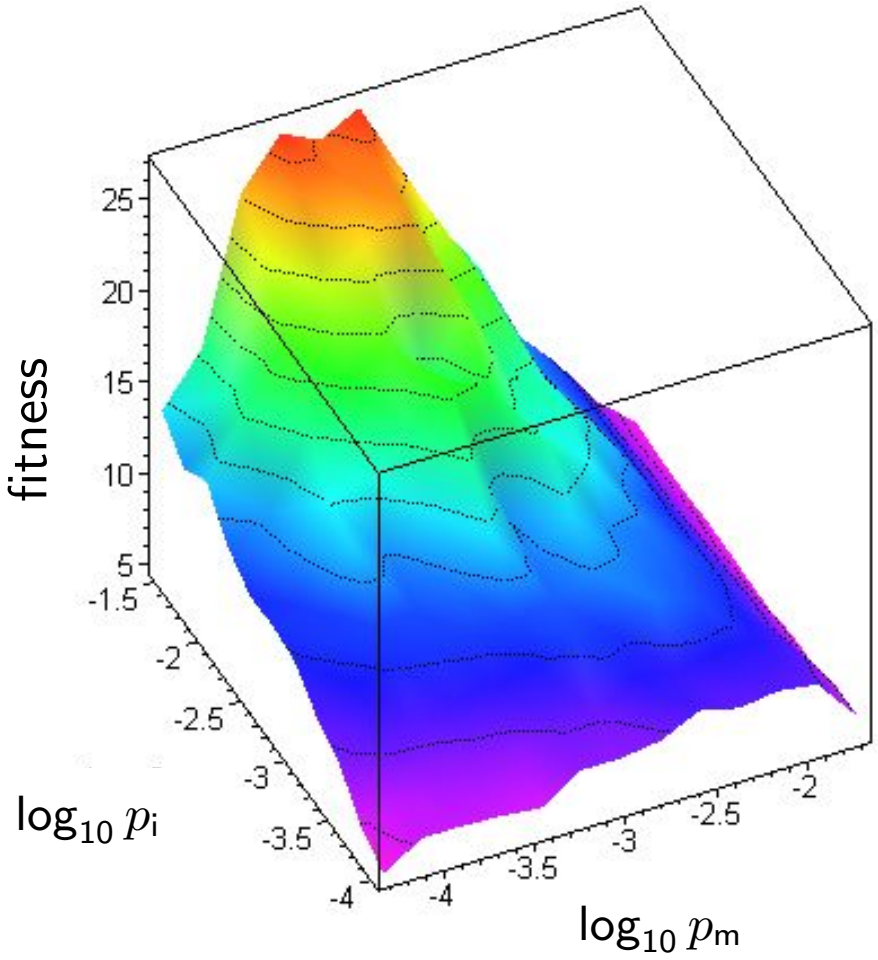


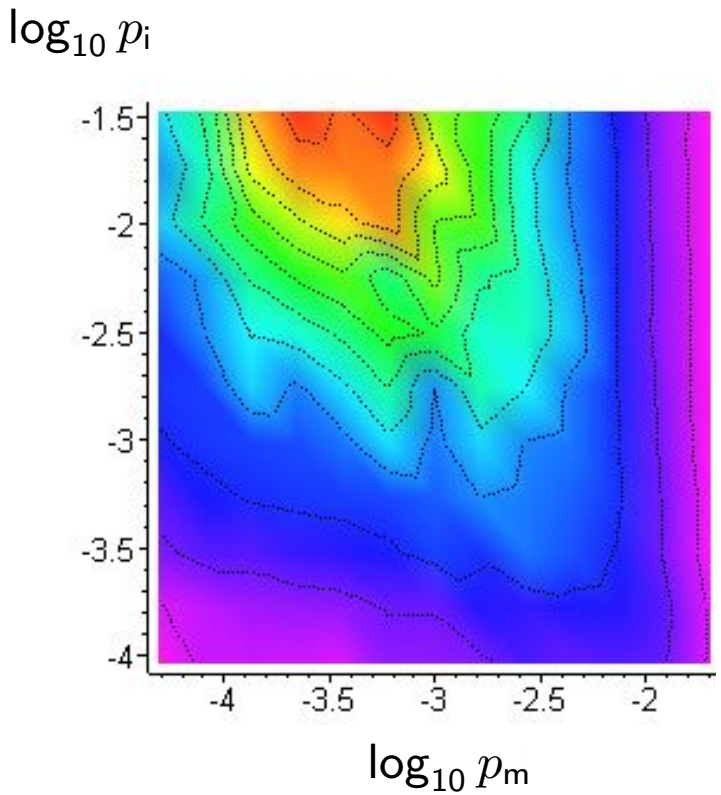
Average fitness (10 different values of seed) versus mutation rate (p_m) and states-increase rate (p_i).
200000 generations.


Evolution of the mean fitness with generations, for different values of p_2
Task: digits of pi
p_2
0.033333
0.018519
0.010204
0.005682
0.003145
0.001748
0.000972
0.000540
0.000300
0.000167
0.000093
0
5
10
15
20
25
30
0
50000
100000
150000
200000

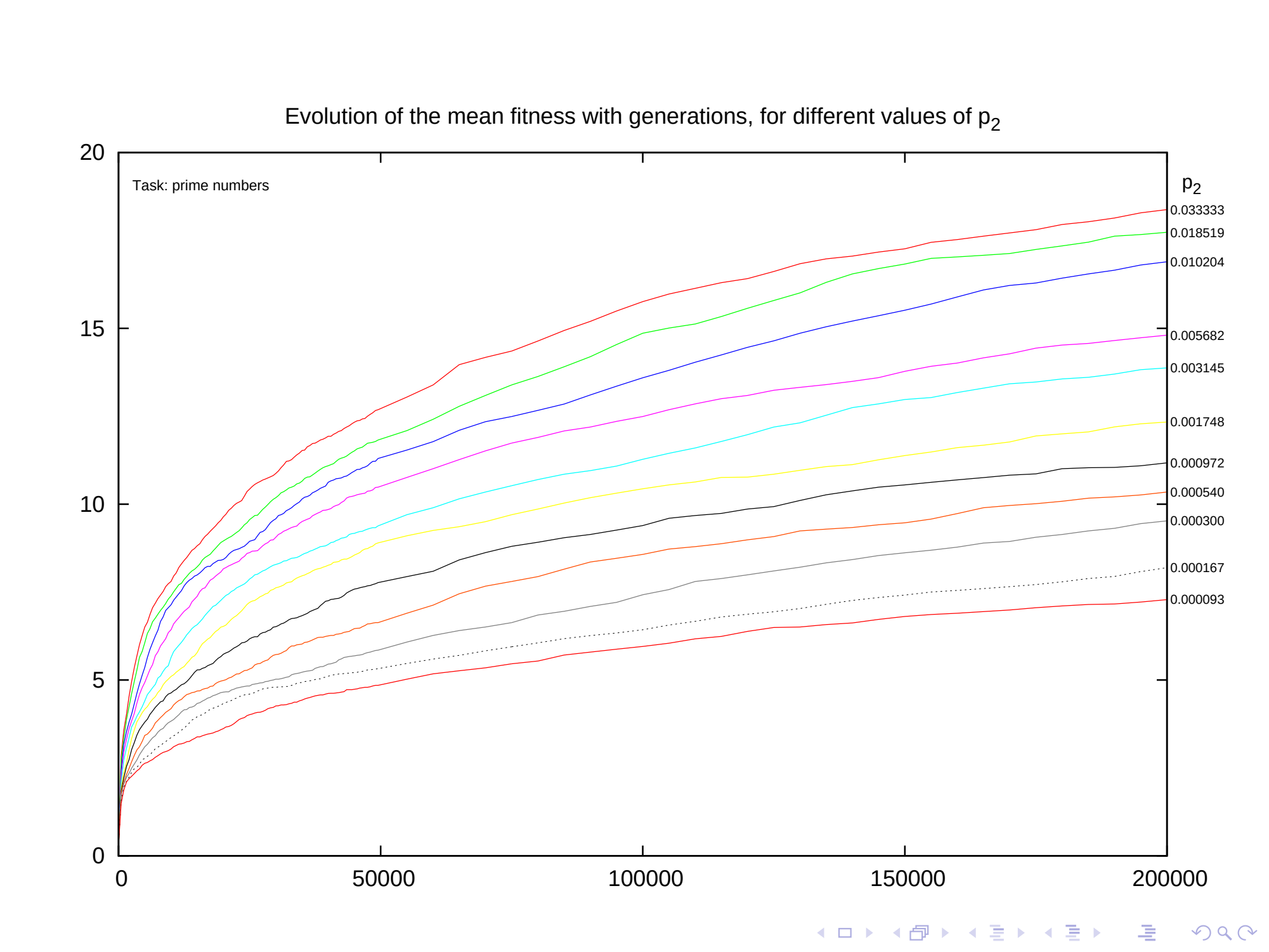
Results

Goal tape: prime numbers





Average fitness (10 different values of seed) versus mutation rate (p_m) and states-increase rate (p_i).
200000 generations.


Evolution of the mean fitness with generations, for different values of p_2
Task: prime numbers
p_2
0.033333
0.018519
0.010204
0.005682
0.003145
0.001748
0.000972
0.000540
0.000300
0.000167
0.000093
0
5
10
15
20
0
50000
100000
150000
200000

Coding and non-coding sequences

total number of states $\sim 200000\ p_{\mathsf{i}} + 1$

observed number of coding states $\frac{N_{\mathsf{c}}}{N_{\mathsf{t}}}$

task	$\frac{1}{10800}$	$\frac{1}{6000}$	$\frac{1}{3333}$	$\frac{1}{1852}$	$\frac{1}{1029}$	$\frac{1}{572}$	$\frac{1}{318}$	$\frac{1}{176}$	$\frac{1}{98}$	$\frac{1}{54}$	$\frac{1}{30}$
primes	0.262	0.255	0.183	0.139	0.091	0.098	0.041	0.044	0.030	0.021	0.019
π	0.403	0.243	0.258	0.231	0.154	0.077	0.081	0.066	0.054	0.032	0.026

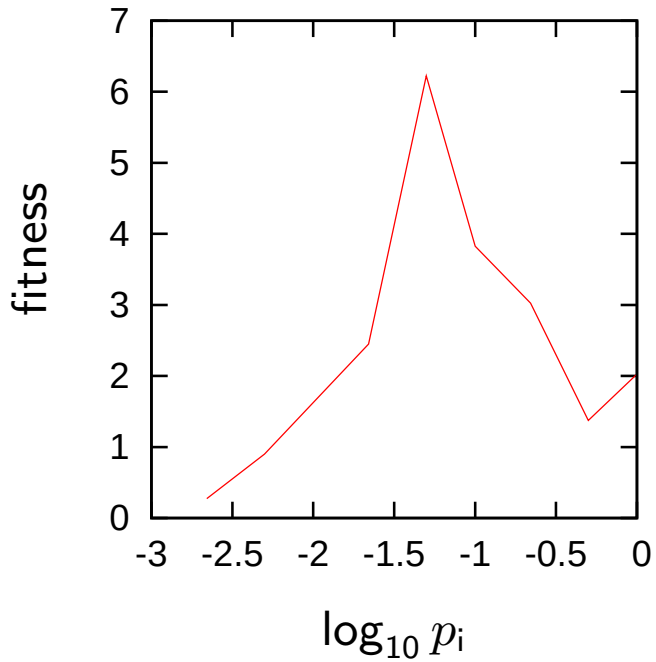
p_{i}

Number of states is indirectly selected

task	p_i	$\frac{1}{10800}$	$\frac{1}{6000}$	$\frac{1}{3333}$	$\frac{1}{1852}$	$\frac{1}{1029}$	$\frac{1}{572}$	$\frac{1}{318}$	$\frac{1}{176}$	$\frac{1}{98}$	$\frac{1}{54}$	$\frac{1}{30}$
	$\bar{N}_{\text{exp}}$	19.5	34.3	61.0	109.0	195.4	350.7	629.9	1137.4	2041.8	3704.7	6667.7
primes	$\bar{N}_{\text{obs}}$	23.5	37.9	64.2	113.5	199.4	355.1	633.0	1144.2	2045.0	3701.2	6670.1
	% diff	**20.48**	**10.49**	**5.18**	**4.12**	2.05	1.27	0.49	0.60	0.15	−0.09	0.04
π	$\bar{N}_{\text{obs}}$	21.8	36.2	63.3	111.1	197.5	356.4	631.0	1134.6	2045.4	3710.0	6665.6
	% diff	**11.66**	**5.48**	**3.71**	1.96	1.10	1.63	0.17	−0.25	0.17	0.14	−0.03

Survival of the "fattest"

A comparative test



Goal tape: multiples of 5

Non-coding triplets: indirect advantage and indirect cost

Conclusions

We started with a population of TMs completely unsuited to a given task and observed that TMs generated with high values of p_i

- experiment a faster evolution
- have a larger portion of non-coding triplets
- on the whole range of generations.

Returning to the original question:
Can the long term advantage of accumulating non-coding DNA balance the short term disadvantage?

Our model suggests that the answer is positive.
Biology: more potential genes are tested

Work in progress: a too large number of non-coding states slows down evolution.
Biology: cost and time

Other