Mutation-Selection Dynamics and Error Threshold in an Evolutionary Model for Turing Machines

joint work with G. Feverati, LAPTH, Annecy, France

F. Musso

fmusso@ubu.es

Department of Physics

Universidad de Burgos

Spain

In silico evolutionary models

In silico evolutionary models

Basic idea: simulate the evolution of computer algorithms subject to mutation and selection procedures with the aim of finding interesting evolutionary behaviours.

In silico evolutionary models

Basic idea: simulate the evolution of computer algorithms subject to mutation and selection procedures with the aim of finding interesting evolutionary behaviours.

Algorithm: input data $\xrightarrow{\text{effective procedure}}$ output data.

In silico evolutionary models

Basic idea: simulate the evolution of computer algorithms subject to mutation and selection procedures with the aim of finding interesting evolutionary behaviours.

Algorithm: input data $\xrightarrow{\text{effective procedure}}$ output data.

Algorithms can be identified with computable functions between integers and specified in terms of a code.

In silico evolutionary models

Basic idea: simulate the evolution of computer algorithms subject to mutation and selection procedures with the aim of finding interesting evolutionary behaviours.

Algorithm: input data $\xrightarrow{\text{effective procedure}}$ output data.

Algorithms can be identified with computable functions between integers and specified in terms of a code.

Mutation: random variations in the code when copying the algorithm.

In silico evolutionary models

Basic idea: simulate the evolution of computer algorithms subject to mutation and selection procedures with the aim of finding interesting evolutionary behaviours.

Algorithm: input data $\xrightarrow{\text{effective procedure}}$ output data.

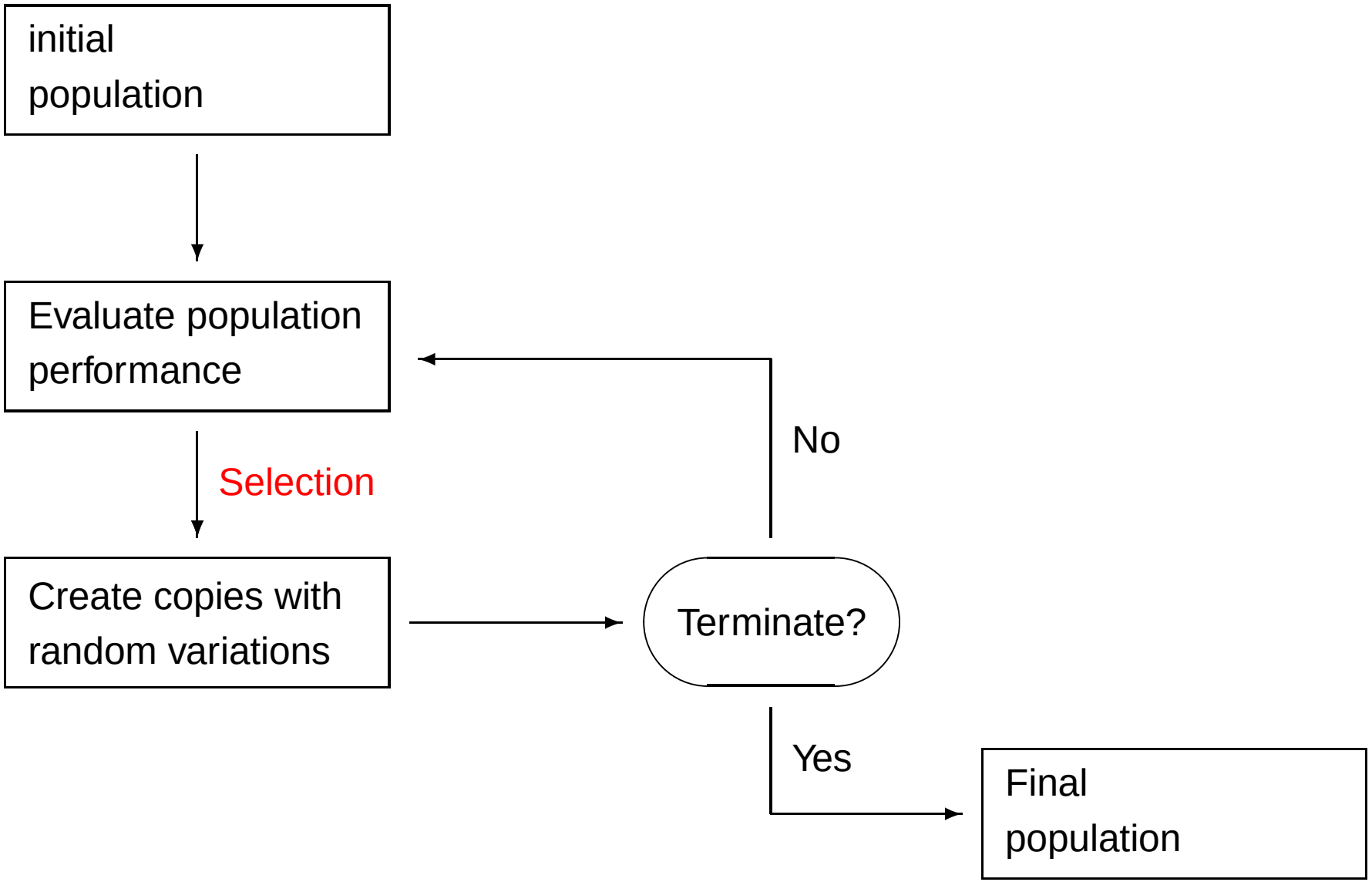
Algorithms can be identified with computable functions between integers and specified in terms of a code.

Mutation: random variations in the code when copying the algorithm.

Selection: the average number of copies (with mutation) of an algorithm at the next generation is an increasing function of its performance.

Performance $\begin{cases} \text{Evaluation of the output vs a goal output} \\ \text{Evaluation of the procedure (i.e. time consumption)} \end{cases}$

Flowchart


initial
population
Evaluate population
performance
Selection
Create copies with
random variations
Terminate?
No
Yes
Final
population

Why?

- Maynard-Smith: “...we badly need a comparative biology. So far, we have been able to study only one evolving system and we cannot wait for interstellar flight to provide us with a second. If we want to discover generalizations about evolving systems, we will have to look at artificial ones.”

Why?

- Maynard-Smith: “...we badly need a comparative biology. So far, we have been able to study only one evolving system and we cannot wait for interstellar flight to provide us with a second. If we want to discover generalizations about evolving systems, we will have to look at artificial ones.”
- Long term evolutionary experiments (even if in a strongly simplified setting):
  - Very fast (50000 generations for 300 TMs last half a day/processor in our model)
  - Unexpensive
  - repeatable
  - Complete control on the experimental apparatus (full reductionistic approach)

Why?

- Maynard-Smith: “...we badly need a comparative biology. So far, we have been able to study only one evolving system and we cannot wait for interstellar flight to provide us with a second. If we want to discover generalizations about evolving systems, we will have to look at artificial ones.”
- Long term evolutionary experiments (even if in a strongly simplified setting):
  - Very fast (50000 generations for 300 TMs last half a day/processor in our model)
  - Unexpensive
  - repeatable
  - Complete control on the experimental apparatus (full reductionistic approach)

But....

Why?

- Maynard-Smith: “...we badly need a comparative biology. So far, we have been able to study only one evolving system and we cannot wait for interstellar flight to provide us with a second. If we want to discover generalizations about evolving systems, we will have to look at artificial ones.”
- Long term evolutionary experiments (even if in a strongly simplified setting):
  - Very fast (50000 generations for 300 TMs last half a day/processor in our model)
  - Unexpensive
  - repeatable
  - Complete control on the experimental apparatus (full reductionistic approach)

But....

Organisms$=$Algorithms

Is it meaningful?

Despite the abismal difference between the evolving objects, could the observed behaviour correspond to general evolutionary phenomena?

Is it meaningful?

Despite the abismal difference between the evolving objects, could the observed behaviour correspond to general evolutionary phenomena?

Is it meaningful?

Despite the abismal difference between the evolving objects, could the observed behaviour correspond to general evolutionary phenomena?
General evolutionary behaviours do emerge if the mutation-selection dynamics have a prominent role on the peculiar characteristics of the evolving organism.

Is it meaningful?

Despite the abismal difference between the evolving objects, could the observed behaviour correspond to general evolutionary phenomena?
General evolutionary behaviours do emerge if the mutation-selection dynamics have a prominent role on the peculiar characteristics of the evolving organism. When this is the case, the observed effects can be reproduced through a population genetic mathematical model.

Is it meaningful?

Despite the abismal difference between the evolving objects, could the observed behaviour correspond to general evolutionary phenomena?

General evolutionary behaviours do emerge if the mutation-selection dynamics have a prominent role on the peculiar characteristics of the evolving organism. When this is the case, the observed effects can be reproduced through a population genetic mathematical model.

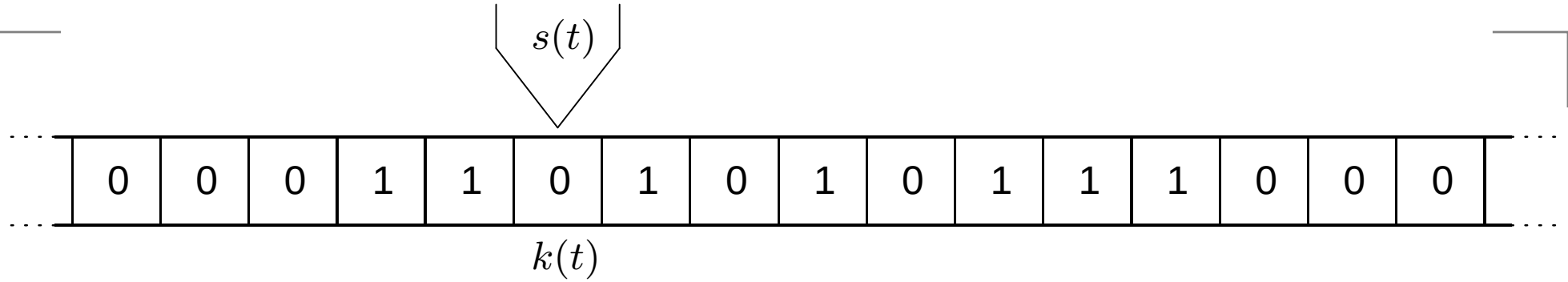
We think that coupling computer simulations with population genetic models is a good strategy to understand the causes underlying the observed phenomena and determine their eventual biological relevance.

Our model

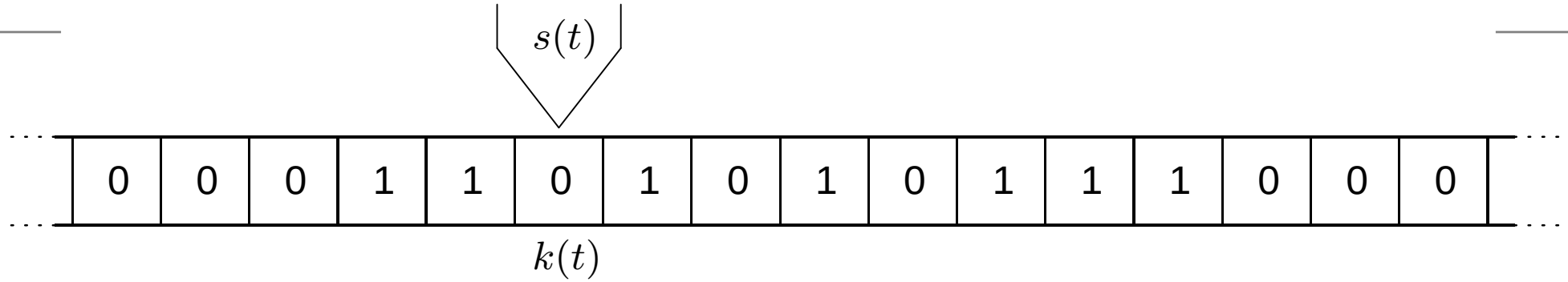
- We use Turing machines (TMs) to encode the algorithms
- Mutation procedure: point mutation and code size increase
- Performance evaluation: comparison between the output tape and a goal tape
- Selection procedure:asexual reproduction determined by tournament selection of size 2 without replacement

What is a Turing Machine?

What is a Turing Machine?

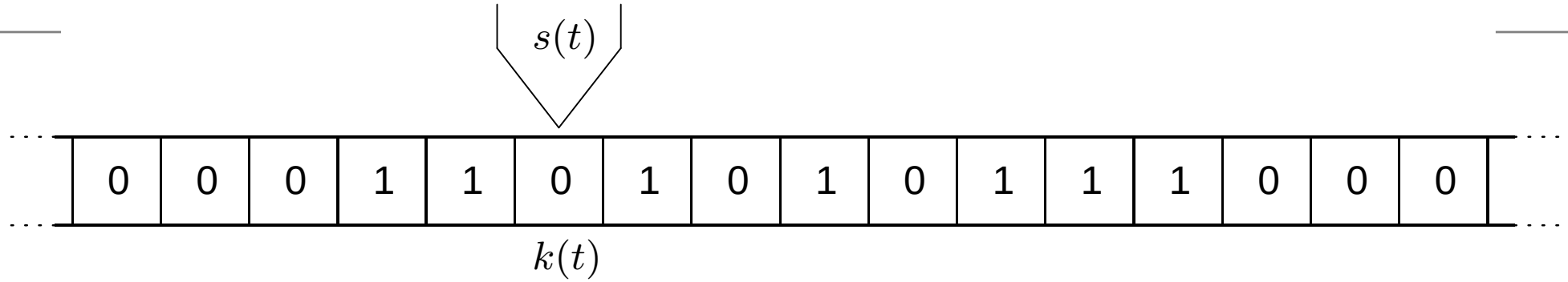


What is a Turing Machine?



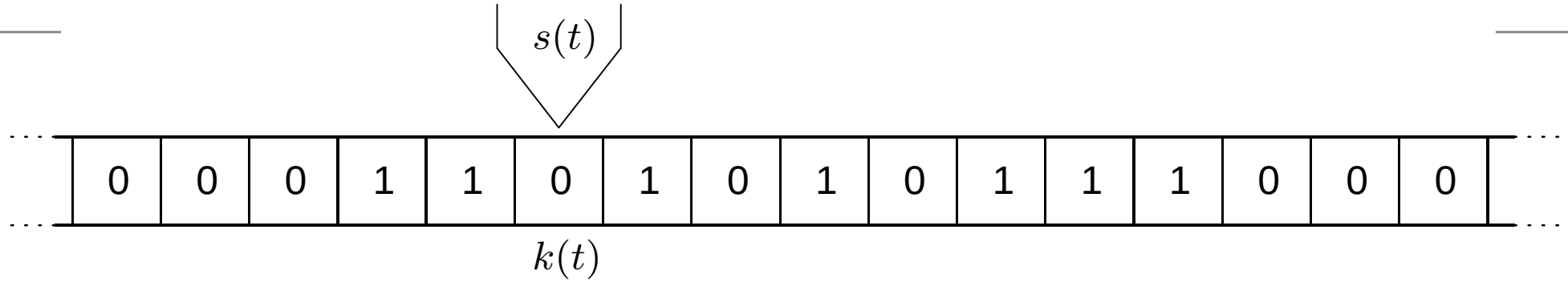
At any time t the head is in a given internal state $\mathbf{s}(t)$ ($\mathbf{s}(t) \in \{H, 1, \ldots, N\}$ and it is located upon a single cell $k(t)$ of the infinite tape $T(t)$.

What is a Turing Machine?



At any time t the head is in a given internal state $\mathbf{s}(t)$ ($\mathbf{s}(t) \in \{H, 1, \ldots, N\}$ and it is located upon a single cell $k(t)$ of the infinite tape $T(t)$.
The internal state $s(t)$, on the basis of the symbol stored inside the $k(t)$ cell, specifies three actions for the head to perform:

What is a Turing Machine?



At any time t the head is in a given internal state $\mathbf{s}(t)$ ($\mathbf{s}(t) \in \{H, 1, \ldots, N\}$ and it is located upon a single cell $k(t)$ of the infinite tape $T(t)$.
The internal state $s(t)$, on the basis of the symbol stored inside the $k(t)$ cell, specifies three actions for the head to perform:

- write: writes a new symbol on the $k(t)$ cell, ($T(t) \mapsto T(t+1)$)
- move: moves right or left ($k(t) \mapsto k(t+1)$),
- call: changes its internal state ($s(t) \mapsto s(t+1)$)

The call to the special state H causes the machine to halt. The tape $T(0)$ is the input tape and the tape $T(\bar{t})$ after the halt state has been called is the output tape.

Example

The following machine performs the sum of two numbers, in the example: $3 + 2$.

read \ state	$\mathbf{1}$	$\mathbf{2}$	$\mathbf{3}$
0	1 – Right – $\mathbf{2}$	0 – Left – $\mathbf{3}$	_ – _ – _
1	1 – Right – $\mathbf{1}$	1 – Right – $\mathbf{2}$	0 – _ – **Halt**

1

… 0 1 1 1 0 1 1 0 …

Example

The following machine performs the sum of two numbers, in the example: $3 + 2$.

state / read	$\mathbf{1}$	$\mathbf{2}$	$\mathbf{3}$
0	1 – Right – $\mathbf{2}$	0 – Left – $\mathbf{3}$	_ – _ – _
1	1 – Right – $\mathbf{1}$	1 – Right – $\mathbf{2}$	0 – _ – **Halt**

1

… 0 1 1 1 0 1 1 0 …

Example

The following machine performs the sum of two numbers, in the example: $3 + 2$.

read \ state	$\mathbf{1}$	$\mathbf{2}$	$\mathbf{3}$
0	$1 -$ Right $- \mathbf{2}$	$0 -$ Left $- \mathbf{3}$	_ $-$ _ $-$ _
1	$1 -$ Right $- \mathbf{1}$	$1 -$ Right $- \mathbf{2}$	$0 -$ _ $-$ **Halt**

1

… 0 1 1 1 0 1 1 0 …

Example

The following machine performs the sum of two numbers, in the example: $3 + 2$.

read \ state	$\mathbf{1}$	$\mathbf{2}$	$\mathbf{3}$
0	1 – Right – $\mathbf{2}$	0 – Left – $\mathbf{3}$	_ – _ – _
1	1 – Right – $\mathbf{1}$	1 – Right – $\mathbf{2}$	0 – _ – **Halt**

1

… 0 1 1 1 0 1 1 0 …

Example

The following machine performs the sum of two numbers, in the example: $3 + 2$.

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

2

… 0 1 1 1 1 1 1 0 …

Example

The following machine performs the sum of two numbers, in the example: $3 + 2$.

read \ state	$\mathbf{1}$	$\mathbf{2}$	$\mathbf{3}$
0	1 – Right – $\mathbf{2}$	0 – Left – $\mathbf{3}$	_ – _ – _
1	1 – Right – $\mathbf{1}$	1 – Right – $\mathbf{2}$	0 – _ – **Halt**

2

… 0 1 1 1 1 1 1 0 …

Example

The following machine performs the sum of two numbers, in the example: $3 + 2$.

read \ state	$\mathbf{1}$	$\mathbf{2}$	$\mathbf{3}$
0	$1 -$ Right $- \mathbf{2}$	$0 -$ Left $- \mathbf{3}$	_ $-$ _ $-$ _
1	$1 -$ Right $- \mathbf{1}$	$1 -$ Right $- \mathbf{2}$	$0 -$ _ $-$ **Halt**

2

… 0 1 1 1 1 1 1 0 …

Example

The following machine performs the sum of two numbers, in the example: $3 + 2$.

read \ state	$\mathbf{1}$	$\mathbf{2}$	$\mathbf{3}$
0	$1 -$ Right $- \mathbf{2}$	$0 -$ Left $- \mathbf{3}$	_ $-$ _ $-$ _
1	$1 -$ Right $- \mathbf{1}$	$1 -$ Right $- \mathbf{2}$	$0 -$ _ $-$ **Halt**

3

… 0 1 1 1 1 1 1 0 …

Example

The following machine performs the sum of two numbers, in the example: $3 + 2$.

read \ state	$\mathbf{1}$	$\mathbf{2}$	$\mathbf{3}$
0	1 – Right – $\mathbf{2}$	0 – Left – $\mathbf{3}$	_ – _ – _
1	1 – Right – $\mathbf{1}$	1 – Right – $\mathbf{2}$	0 – _ – **Halt**

… 0 1 1 1 1 1 0 0 …

Initial population

We begin with a population of 300 1-state TM of the following form:

	1
0	0-R/L-Halt
1	1-R/L-Halt

and let them evolve for 50000 generations (termination condition).

Mutation procedure

Mutation is composed of two steps:

- State increasing: with a probability p_i the TM passes from N to N+1 states by the addition of the further state:

	N+1
0	0-R/L-Halt
1	1-R/L-Halt

Mutation procedure

Mutation is composed of two steps:

- State increasing: with a probability p_i the TM passes from N to N+1 states by the addition of the further state:

	N+1
0	0-R/L-Halt
1	1-R/L-Halt

- Point mutation: every value inside the transition tables of every state is modified with a probability p_m.

Mutation procedure

Mutation is composed of two steps:

- State increasing: with a probability p_i the TM passes from N to N+1 states by the addition of the further state:

	N+1
0	0-R/L-Halt
1	1-R/L-Halt

- Point mutation: every value inside the transition tables of every state is modified with a probability p_m.

Mutation procedure

Mutation is composed of two steps:

- State increasing: with a probability p_i the TM passes from N to N+1 states by the addition of the further state:

	N+1
0	0-R/L-Halt
1	1-R/L-Halt

- Point mutation: every value inside the transition tables of every state is modified with a probability p_m.

The new entry is randomly chosen among all corresponding permitted values excluding the original one. The permitted values are:

Mutation procedure

Mutation is composed of two steps:

- State increasing: with a probability p_i the TM passes from N to N+1 states by the addition of the further state:

	N+1
0	0-R/L-Halt
1	1-R/L-Halt

- Point mutation: every value inside the transition tables of every state is modified with a probability p_m.

The new entry is randomly chosen among all corresponding permitted values excluding the original one. The permitted values are:

- 0 or 1 for the “write” entries;
- Right, Left for the “move” entries;
- The Halt state or an integer from $\mathbf{1}$ to the number of states $\mathbf{N}$ of the machine for the “call” entries.

Selection and reproduction

- Two TM are randomly extracted from the old population

Selection and reproduction

- Two TM are randomly extracted from the old population
- They are provided with a periodic input tape made of 300 zeroes

Selection and reproduction

- Two TM are randomly extracted from the old population
- They are provided with a periodic input tape made of 300 zeroes
- They run until one of the following things happens:

Selection and reproduction

- Two TM are randomly extracted from the old population
- They are provided with a periodic input tape made of 300 zeroes
- They run until one of the following things happens:

  1. the machine enter the halt state,

Selection and reproduction

- Two TM are randomly extracted from the old population
- They are provided with a periodic input tape made of 300 zeroes
- They run until one of the following things happens:
  1. the machine enter the halt state,
  2. the machine has run for 4000 time steps

Selection and reproduction

- Two TM are randomly extracted from the old population
- They are provided with a periodic input tape made of 300 zeroes
- They run until one of the following things happens:
  1. the machine enter the halt state,
  2. the machine has run for 4000 time steps
- The performance of the machines is computed comparing the output tape with a “goal-tape”: $+1$ for every 1 placed on the right position, -3 for every 1 placed on the wrong position.

Selection and reproduction

- Two TM are randomly extracted from the old population
- They are provided with a periodic input tape made of 300 zeroes
- They run until one of the following things happens:
  1. the machine enter the halt state,
  2. the machine has run for 4000 time steps
- The performance of the machines is computed comparing the output tape with a "goal-tape": $+1$ for every 1 placed on the right position, -3 for every 1 placed on the wrong position.
- The performances are compared, the machine which scores higher is copied on the other one that is eliminated (asexual reproduction). If the performances are equal, nothing happens.

Selection and reproduction

- Two TM are randomly extracted from the old population
- They are provided with a periodic input tape made of 300 zeroes
- They run until one of the following things happens:
  1. the machine enter the halt state,
  2. the machine has run for 4000 time steps
- The performance of the machines is computed comparing the output tape with a "goal-tape": $+1$ for every 1 placed on the right position, -3 for every 1 placed on the wrong position.
- The performances are compared, the machine which scores higher is copied on the other one that is eliminated (asexual reproduction). If the performances are equal, nothing happens.
- The two TM are eliminated from the old population and placed in the new population. And the process restart until exhaustion of the old population

Example

Goal tape: prime numbers

	1	2	3	4	5	6	7	8
0	1-R-4	1-R-5	1-R-6	1-R-3	1-R-7	0-R-2	1-L-7	0-R-8
1	1-R-3	1-L-2	0-L-3	0-L-5	0-L-4	1-L-1	0-L-H	0-R-2

1

0 0 0 0 0 0 0 0 . . .

Example

Goal tape: prime numbers

	1	2	3	4	5	6	7	8
0	1-R-4	1-R-5	1-R-6	1-R-3	1-R-7	0-R-2	1-L-7	0-R-8
1	1-R-3	1-L-2	0-L-3	0-L-5	0-L-4	1-L-1	0-L-H	0-R-2

4

1 0 0 0 0 0 0 0 . . .

Example

Goal tape: prime numbers

	1	2	3	4	5	6	7	8
0	1-R-4	1-R-5	1-R-6	1-R-3	1-R-7	0-R-2	1-L-7	0-R-8
1	1-R-3	1-L-2	0-L-3	0-L-5	0-L-4	1-L-1	0-L-H	0-R-2

3

1 1 0 0 0 0 0 0 . . .

Example

Goal tape: prime numbers

	1	2	3	4	5	6	7	8
0	1-R-4	1-R-5	1-R-6	1-R-3	1-R-7	0-R-2	1-L-7	0-R-8
1	1-R-3	1-L-2	0-L-3	0-L-5	0-L-4	1-L-1	0-L-H	0-R-2

6

1 1 1 0 0 0 0 0 . . .

Example

Goal tape: prime numbers

	1	2	3	4	5	6	7	8
0	1-R-4	1-R-5	1-R-6	1-R-3	1-R-7	0-R-2	1-L-7	0-R-8
1	1-R-3	1-L-2	0-L-3	0-L-5	0-L-4	1-L-1	0-L-H	0-R-2

2

1 1 1 0 0 0 0 0 . . .

Example

Goal tape: prime numbers

	1	2	3	4	5	6	7	8
0	1-R-4	1-R-5	1-R-6	1-R-3	1-R-7	0-R-2	1-L-7	0-R-8
1	1-R-3	1-L-2	0-L-3	0-L-5	0-L-4	1-L-1	0-L-H	0-R-2

5

1 1 1 0 1 0 0 0 . . .

Example

Goal tape: prime numbers

	1	2	3	4	5	6	7	8
0	1-R-4	1-R-5	1-R-6	1-R-3	1-R-7	0-R-2	1-L-7	0-R-8
1	1-R-3	1-L-2	0-L-3	0-L-5	0-L-4	1-L-1	0-L-H	0-R-2

7

1 1 1 0 1 1 0 0 . . .

Example

Goal tape: prime numbers

	1	2	3	4	5	6	7	8
0	1-R-4	1-R-5	1-R-6	1-R-3	1-R-7	0-R-2	1-L-7	0-R-8
1	1-R-3	1-L-2	0-L-3	0-L-5	0-L-4	1-L-1	0-L-H	0-R-2

7

1 1 1 0 1 1 1 0 . . .

Example

Goal tape: prime numbers

	1	2	3	4	5	6	7	8
0	1-R-4	1-R-5	1-R-6	1-R-3	1-R-7	0-R-2	1-L-7	0-R-8
1	1-R-3	1-L-2	0-L-3	0-L-5	0-L-4	1-L-1	0-L-H	0-R-2

H

1 1 1 0 1 0 1 0 . . .

Example

Goal tape: prime numbers

	1	2	3	4	5	6	7	8
0	1-R-4	1-R-5	1-R-6	1-R-3	1-R-7	0-R-2	1-L-7	0-R-8
1	1-R-3	1-L-2	0-L-3	0-L-5	0-L-4	1-L-1	0-L-H	0-R-2

H

1 1 1 0 1 0 1 0 . . .

Performance $= 5$,

Example

Goal tape: prime numbers

	1	2	3	4	5	6	7	8
0	1-R-4	1-R-5	1-R-6	1-R-3	1-R-7	0-R-2	1-L-7	0-R-8
1	1-R-3	1-L-2	0-L-3	0-L-5	0-L-4	1-L-1	0-L-H	0-R-2

H

1 1 1 0 1 0 1 0 . . .

Performance $= 5$,

Blue=Coding Triplets, Total states$= 8$, Coding triplets$= 8$, Non-coding triplets$= 8$.

Why TMs?

- every algorithm can be encoded through a TM (Church-Turing thesis: every function which would naturally be regarded as computable is Turing computable)

Why TMs?

- every algorithm can be encoded through a TM (Church-Turing thesis: every function which would naturally be regarded as computable is Turing computable)
- point mutation always produces legal code (no sintax errors)

Why TMs?

- every algorithm can be encoded through a TM (Church-Turing thesis: every function which would naturally be regarded as computable is Turing computable)
- point mutation always produces legal code (no sintax errors)
- state-increasing is always neutral

Why TMs?

- every algorithm can be encoded through a TM (Church-Turing thesis: every function which would naturally be regarded as computable is Turing computable)
- point mutation always produces legal code (no sintax errors)
- state-increasing is always neutral
- there is a simple mechanism of code activation through mutations

Why TMs?

- every algorithm can be encoded through a TM (Church-Turing thesis: every function which would naturally be regarded as computable is Turing computable)
- point mutation always produces legal code (no sintax errors)
- state-increasing is always neutral
- there is a simple mechanism of code activation through mutations
- can give raise to a very complicated behaviour even with very few states

Why TMs?

- every algorithm can be encoded through a TM (Church-Turing thesis: every function which would naturally be regarded as computable is Turing computable)
- point mutation always produces legal code (no sintax errors)
- state-increasing is always neutral
- there is a simple mechanism of code activation through mutations
- can give raise to a very complicated behaviour even with very few states
- are defined in terms of an atomic instruction: the state.

Simulations

- We used a C-program to simulate the evolution of a population of 300 TMs

Simulations

- We used a C-program to simulate the evolution of a population of 300 TMs
- as a goal tape we used a a tape reproducing the decimal part of π in binary (maximum performance=125)

Simulations

- We used a C-program to simulate the evolution of a population of 300 TMs
- as a goal tape we used a a tape reproducing the decimal part of π in binary (maximum performance=125)
- We let p_{i} and p_{m} vary in the sets:

$$
\begin{aligned}
p_{\mathsf{i}} \quad \in \quad & \{9.26 \cdot 10^{-5}\,; 1.66 \cdot 10^{-4}\,; 3.00 \cdot 10^{-4}\,; 5.40 \cdot 10^{-4}\,; 9.72 \cdot 10^{-4}\,; 1.75 \cdot 10^{-3}\,; \\
& 3.14 \cdot 10^{-3}\,; 5.68 \cdot 10^{-3}\,; 1.02 \cdot 10^{-2}\,; 1.85 \cdot 10^{-2}\,; 3.33 \cdot 10^{-2}; 6.00 \cdot 10^{-2}\,; \\
& 1.08 \cdot 10^{-1}\,; 1.95 \cdot 10^{-1}\,; 3.51 \cdot 10^{-1}\,; 6.33 \cdot 10^{-1}\,; 1.14\} \ . \\
p_{\mathsf{m}} \quad \in \quad & \{4.91 \cdot 10^{-5}\,; 8.10 \cdot 10^{-5}\,; 1.34 \cdot 10^{-4}\,; 2.21 \cdot 10^{-4}\,; 3.64 \cdot 10^{-4}\,; 6.01 \cdot 10^{-4}\,; \\
& 9.91 \cdot 10^{-4}\,; 1.64 \cdot 10^{-3}\,; 2.70 \cdot 10^{-3}\,; 4.44 \cdot 10^{-3}\,; 7.35 \cdot 10^{-3}\ \} \ .
\end{aligned}
$$

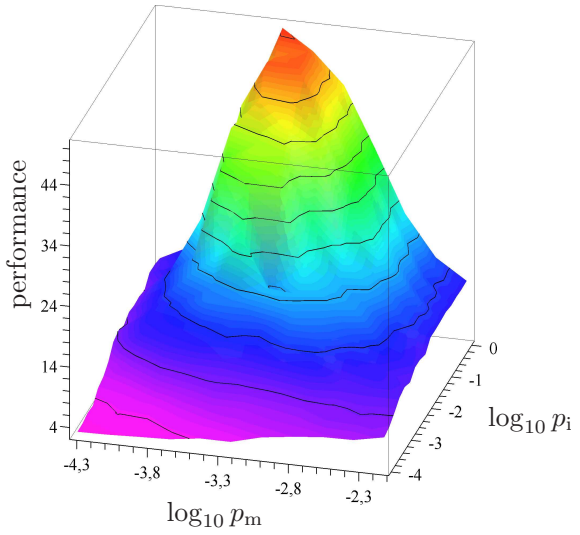
Simulations

- We used a C-program to simulate the evolution of a population of 300 TMs
- as a goal tape we used a a tape reproducing the decimal part of π in binary (maximum performance=125)
- We let p_{i} and p_{m} vary in the sets:

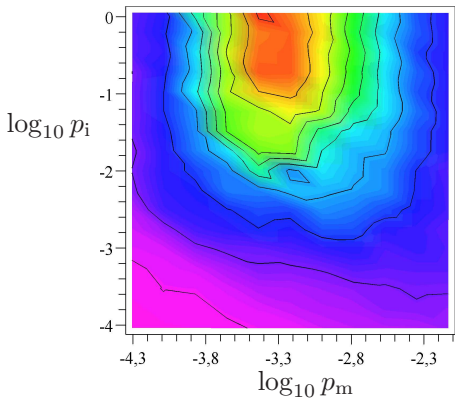
$$
\begin{aligned}
p_{\mathrm{i}} \quad \in \quad & \{9.26 \cdot 10^{-5}\,; 1.66 \cdot 10^{-4}\,; 3.00 \cdot 10^{-4}\,; 5.40 \cdot 10^{-4}\,; 9.72 \cdot 10^{-4}\,; 1.75 \cdot 10^{-3}\,; \\
& 3.14 \cdot 10^{-3}\,; 5.68 \cdot 10^{-3}\,; 1.02 \cdot 10^{-2}\,; 1.85 \cdot 10^{-2}\,; 3.33 \cdot 10^{-2}; 6.00 \cdot 10^{-2}\,; \\
& 1.08 \cdot 10^{-1}\,; 1.95 \cdot 10^{-1}\,; 3.51 \cdot 10^{-1}\,; 6.33 \cdot 10^{-1}\,; 1.14\} \ . \\
p_{\mathrm{m}} \quad \in \quad & \{4.91 \cdot 10^{-5}\,; 8.10 \cdot 10^{-5}\,; 1.34 \cdot 10^{-4}\,; 2.21 \cdot 10^{-4}\,; 3.64 \cdot 10^{-4}\,; 6.01 \cdot 10^{-4}\,; \\
& 9.91 \cdot 10^{-4}\,; 1.64 \cdot 10^{-3}\,; 2.70 \cdot 10^{-3}\,; 4.44 \cdot 10^{-3}\,; 7.35 \cdot 10^{-3}\ \} \ .
\end{aligned}
$$

- For any pair p_m, p_i we performed 20 simulations of 50000 generations varying the seed of the random number generator

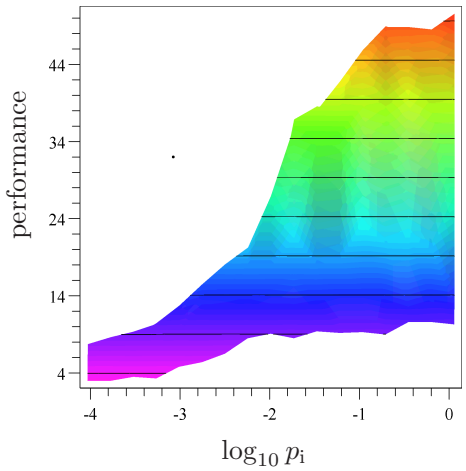
Results



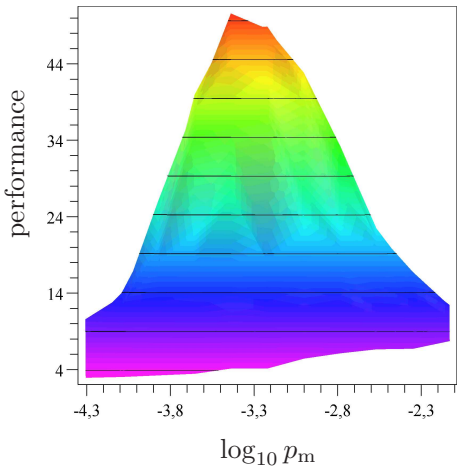
(a)



(b)



(c)

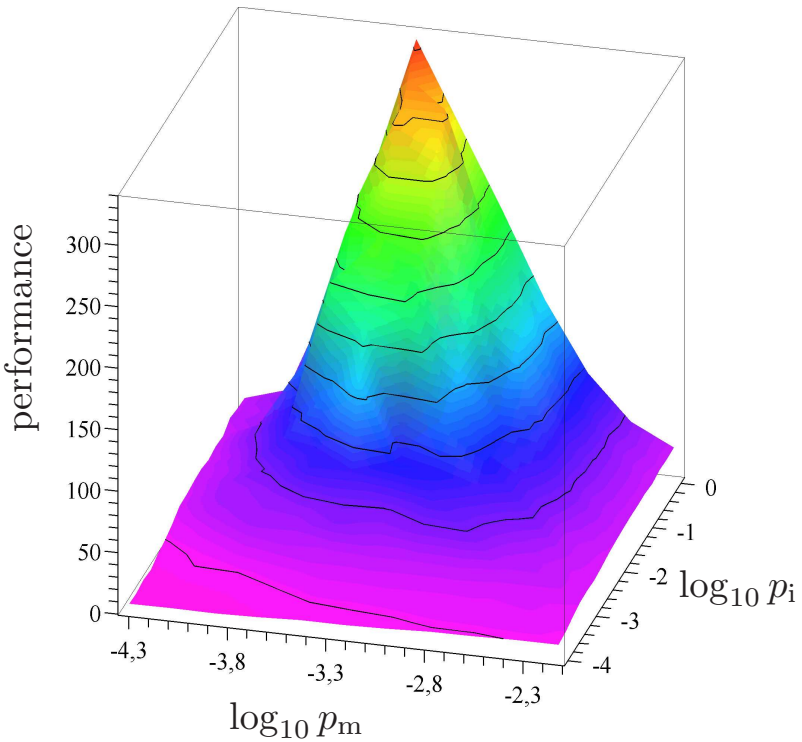


(d)

Best performance value in the population at the last generation.

The best performance is averaged on the twenty different seeds and plotted as a function of the states-increase rate p_{i} and of the mutation rate p_{m}

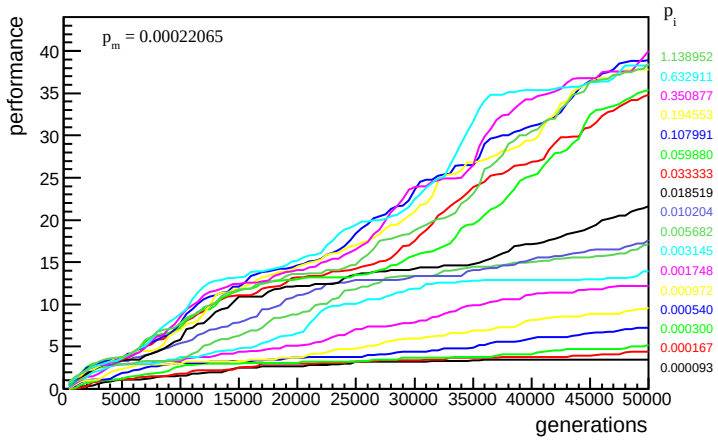
Coding Triplets



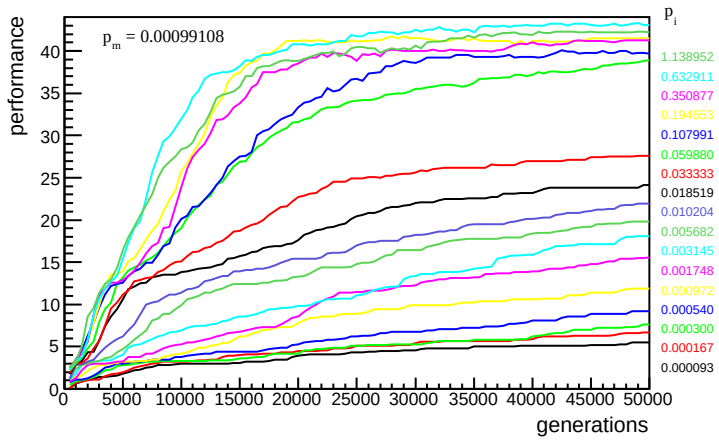
Number of coding triplets in the population at the last generation averaged on the best machines and on the seeds.

There is a positive correlation between the performance and the number of coding triplets ($r = 0.95$).

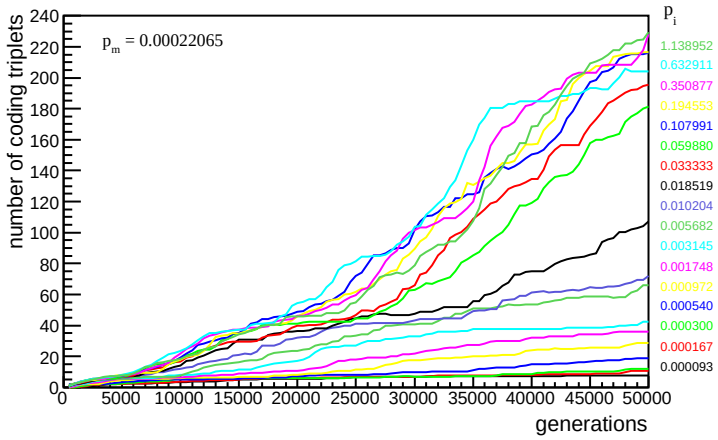
Accumulation of coding triplets happens in a gradual way, so that it is slower for low values of p_{m}; high values of p_{m} limit the number of coding triplets.



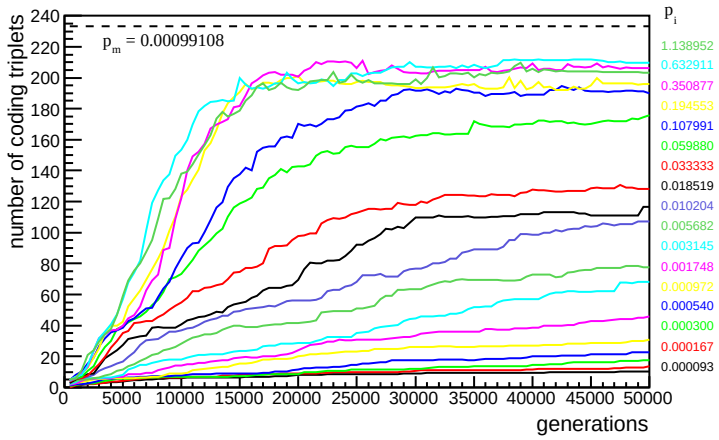
(a)



(b)



(c)



(d)

Data along the generations. Data are sampled every 100 generations and averaged on the seeds.

TMs increase their performance by increasing their number of coding triplets.

Error threshold

Given a mutation probability, what is the maximum amount of coding triplets?

How near do the TMs come to this maximum amount?

Error threshold

Given a mutation probability, what is the maximum amount of coding triplets?
How near do the TMs come to this maximum amount?
M. Eigen, studying RNA replication at the origin of life, developed a mathematical model to answer this question.

Error threshold

Given a mutation probability, what is the maximum amount of coding triplets?
How near do the TMs come to this maximum amount?
M. Eigen, studying RNA replication at the origin of life, developed a mathematical model to answer this question.
From its model he derived the Eigen's paradox:

- Without error correction machinery, the maximum size of a naked (RNA) gene replicator is about 100 bases.
- In order for a genome to specify an error correction machinery, it must be substantially larger than 100 bases.

Error threshold

Given a mutation probability, what is the maximum amount of coding triplets?
How near do the TMs come to this maximum amount?
M. Eigen, studying RNA replication at the origin of life, developed a mathematical model to answer this question.
From its model he derived the Eigen's paradox:

- Without error correction machinery, the maximum size of a naked (RNA) gene replicator is about 100 bases.
- In order for a genome to specify an error correction machinery, it must be substantially larger than 100 bases.

The selection mechanism used in the Eigen model is different from our, so that we need a different mathematical model.

Deterministic model

Let us consider a population of fixed size N.

Deterministic model

Let us consider a population of fixed size N.

Let us suppose that do exist M possible performance classes and let us denote with n_i the number of individuals inside the $i-$th class.

Deterministic model

Let us consider a population of fixed size N.
Let us suppose that do exist M possible performance classes and let us denote with n_i the number of individuals inside the $i-$th class.
We generalize our selection procedure by allowing the individual with the highest performance to make two copies of itself with probability f or one copy with probability $1-f$ (TM case corresponds to $f=1$), $0 \leq f \leq 1$.

Deterministic model

Let us consider a population of fixed size N.
Let us suppose that do exist M possible performance classes and let us denote with n_i the number of individuals inside the $i-$th class.
We generalize our selection procedure by allowing the individual with the highest performance to make two copies of itself with probability f or one copy with probability $1-f$ (TM case corresponds to $f=1$), $0 \leq f \leq 1$.
With this mechanism, the probabilities of producing $2, 1$ or 0 offsprings for an individual belonging to the $i-$th performance class are, respectively

$$
\begin{aligned}
P_2 &= f\frac{\sum_{j<i} n_j}{N-1} \\
P_1 &= \frac{1}{N-1}\left((1-f)\sum_{j<i} n_j + n_i - 1 + (1-f)\sum_{j>i} n_j\right) \\
P_0 &= \frac{f}{N-1}\sum_{j>i} n_j
\end{aligned}
$$

It follows that the expected number n_i' of individuals in the ith performance class after selection is given by:

$$n_i' = n_i \left[1 + \frac{f}{N-1} \left(\sum_{j<i} n_j - \sum_{j>i} n_j \right) \right]$$

It follows that the expected number n_i' of individuals in the ith performance class after selection is given by:

$$n_i' = n_i \left[1 + \frac{f}{N-1} \left(\sum_{j<i} n_j - \sum_{j>i} n_j \right) \right]$$

Let Q_i be the probability of neutral or no mutation and g_{ij} the probability to pass from class j to class i because of mutation ($g_{ii} = 0$), then the number of individuals inside the $i-$th class after the mutation step will be:

$$n_i'' = n_i' Q_i + \sum_{j=1}^{M} (1 - Q_j) n_j' g_{ij}$$

Let s be the best occupied performance class at a given time $n_s > 0$, $n_i = 0,\ i > s$ and let us suppose that $g_{ij} \ll 1$ if $i > j$, then

$$n_s'' \simeq n_s Q_s \left[1 + \frac{f}{N-1} \left(\sum_{j<s} n_j \right) \right] = n_s Q_s \left[1 + \frac{f}{N-1} \left(N - n_s \right) \right]$$

Let s be the best occupied performance class at a given time $n_s > 0$, $n_i = 0,\ i > s$ and let us suppose that $g_{ij} \ll 1$ if $i > j$, then

$$n_s'' \simeq n_s Q_s \left[1 + \frac{f}{N-1}\left(\sum_{j<s} n_j\right)\right] = n_s Q_s \left[1 + \frac{f}{N-1}\left(N - n_s\right)\right]$$

Imposing the existence of a stable equilibrium larger than zero $n_s'' = n_s > 0$, we get

$$Q_s > \frac{1}{1 + f\frac{N}{N-1}} = \frac{1}{1+f} + O\left(\frac{1}{N}\right) \doteq \bar{Q}$$

Let s be the best occupied performance class at a given time $n_s > 0$, $n_i = 0,\ i > s$ and let us suppose that $g_{ij} \ll 1$ if $i > j$, then

$$n_s'' \simeq n_s Q_s \left[1 + \frac{f}{N-1}\left(\sum_{j<s} n_j\right)\right] = n_s Q_s \left[1 + \frac{f}{N-1}\left(N - n_s\right)\right]$$

Imposing the existence of a stable equilibrium larger than zero $n_s'' = n_s > 0$, we get

$$Q_s > \frac{1}{1 + f\frac{N}{N-1}} = \frac{1}{1+f} + O\left(\frac{1}{N}\right) \doteq \bar{Q}$$

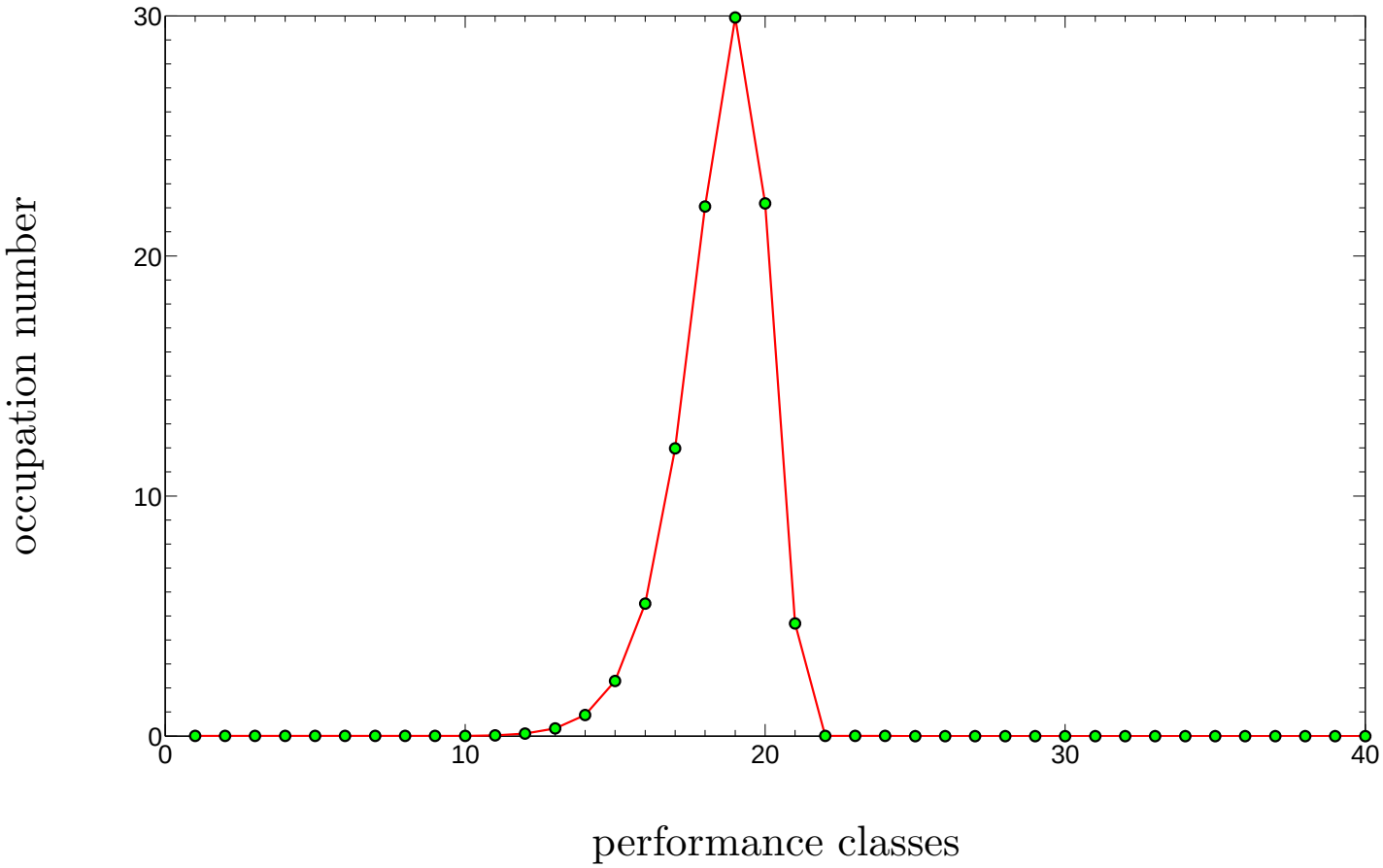
For example, if $M = 40$, $N = 100$, $f = 10^{-3}$,

$$g_{ij} = (1 - 10^{-6})(\delta_{i,1}\delta_{j,1} + \delta_{j,i+1}) + 10^{-6}(\delta_{i,40}\delta_{j,40} + \delta_{j,i-1}),$$

$$Q_i = (1 - 10^{-5})^{\sqrt{i^3}}, \qquad i = 1, \ldots, 40,$$

the last occupied class will be the $21-$st.

Numerical simulation



It can be shown that under these mutation-selection mechanisms, after an infinite number of generations, the population reaches the error threshold or the maximum performance class.

If we assume that almost all mutations inside a coding triplet are deleterious, then the fidelity rate of a TM with N_c coding triplets is:

$$Q = (1 - p_{\mathsf{m}})^{3N_c}$$

If we assume that almost all mutations inside a coding triplet are deleterious, then the fidelity rate of a TM with N_c coding triplets is:

$$Q = (1 - p_{\mathsf{m}})^{3N_c}$$

In our case $f = 1$ implies that the fidelity threshold is

$$\bar{Q} = \frac{1}{2}.$$

If we assume that almost all mutations inside a coding triplet are deleterious, then the fidelity rate of a TM with N_c coding triplets is:

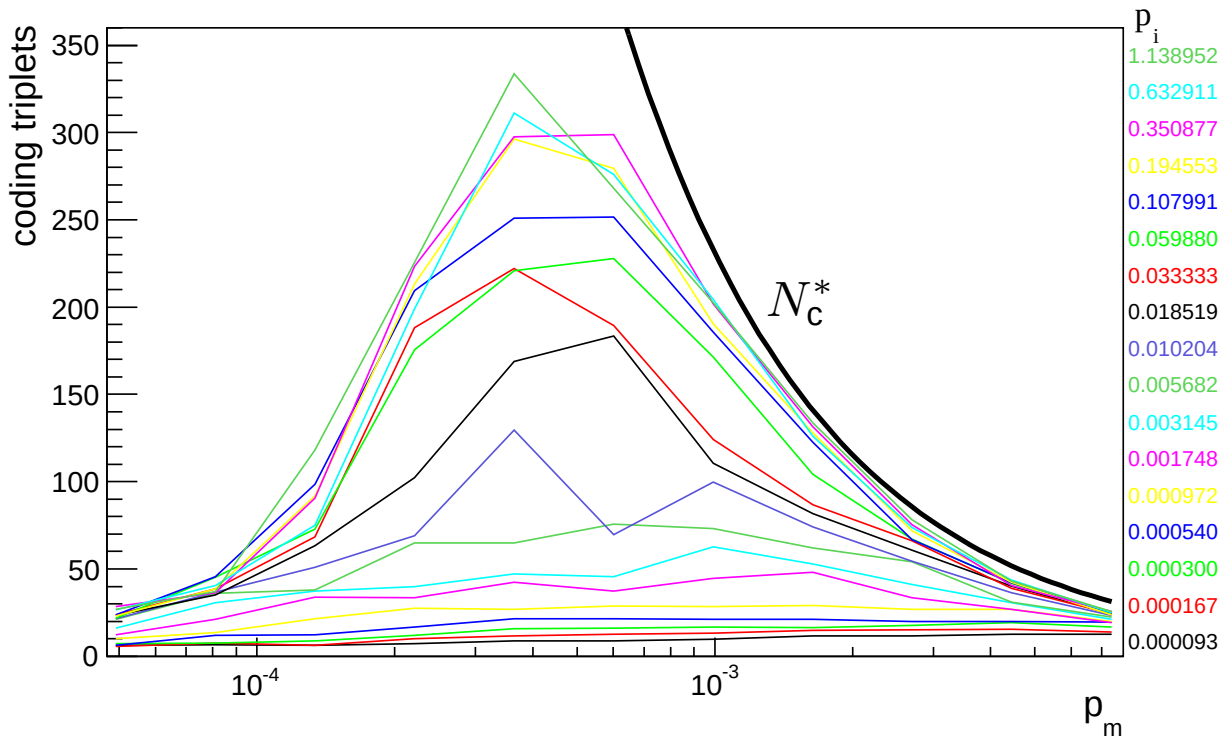
$$Q = (1 - p_{\mathsf{m}})^{3N_c}$$

In our case $f = 1$ implies that the fidelity threshold is

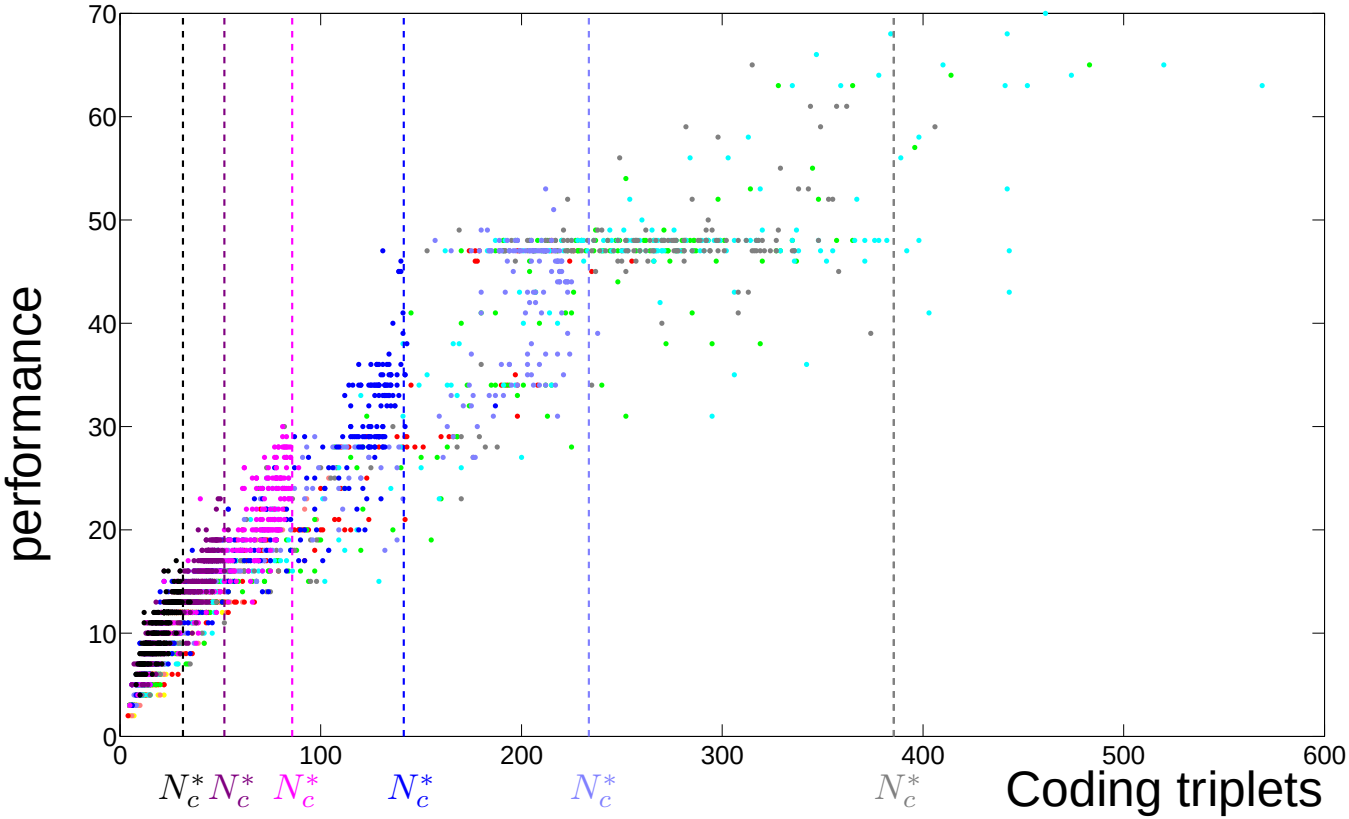
$$\bar{Q} = \frac{1}{2}.$$

So, the maximum number of coding triplets is

$$N_{\mathsf{c}}^{*} = -\frac{\ln(2)}{3\ln(1 - p_{\mathsf{m}})}$$



Plot of the number of coding triplets for the best machine in the population. The number of coding triplets after 50000 generations, averaged on the seeds, is shown as a function of p_{m}, for all the values of p_{i}. The black thick line on the right represents the critical number of coding triplets.



p_{m} : $4.91 \cdot 10^{-5}$, $8.10 \cdot 10^{-5}$, $1.34 \cdot 10^{-4}$, $2.21 \cdot 10^{-4}$, $3.64 \cdot 10^{-4}$, $6.01 \cdot 10^{-4}$, $9.91 \cdot 10^{-4}$, $1.64 \cdot 10^{-3}$, $2.70 \cdot 10^{-3}$, $4.44 \cdot 10^{-3}$, $7.35 \cdot 10^{-3}$

Performance versus the number of coding triplets. The performance is shown for the best performant TMs at generation 50000 for the 3740 runs of our simulations.

The stochastic model

The deterministic model works for infinite populations.

The stochastic model

The deterministic model works for infinite populations.

When the population is finite there are stochastic effects that can be kept into account through a Markov model.

The stochastic model

The deterministic model works for infinite populations.
When the population is finite there are stochastic effects that can be kept into account through a Markov model.
The i, j entry of the Markov matrix gives the probability of passing from the $i-$th state of the system to the $j-$th one.

The stochastic model

The deterministic model works for infinite populations.
When the population is finite there are stochastic effects that can be kept into account through a Markov model.
The i, j entry of the Markov matrix gives the probability of passing from the $i-$th state of the system to the $j-$th one.
We will consider the evolution in the number n_s of the best individuals only.

The stochastic model

The deterministic model works for infinite populations.
When the population is finite there are stochastic effects that can be kept into account through a Markov model.
The i, j entry of the Markov matrix gives the probability of passing from the $i-$th state of the system to the $j-$th one.
We will consider the evolution in the number n_s of the best individuals only.
For the selection step we have:

$$P_{rip}(n_s \to n'_s) = \begin{cases} 0 & \text{if } n'_s < n_s \text{ or } n'_s > \min(2n_s, N) \\ 0 & \text{if } n'_s \text{ odd} \\ 2^{(n'_s - n_s)} \begin{pmatrix} \frac{N}{2} \\ \frac{2n_s - n'_s}{2} \end{pmatrix} \begin{pmatrix} \frac{N - 2n_s + n'_s}{2} \\ n'_s - n_s \end{pmatrix} \Big/ \begin{pmatrix} N \\ n_s \end{pmatrix} & \text{otherwise.} \end{cases}$$

The mutation step

We use the two simplifying assumptions that we already used for the deterministic model, namely:

- TMs in the best performance class have the same number of coding triplets N_{c}.

The mutation step

We use the two simplifying assumptions that we already used for the deterministic model, namely:

- TMs in the best performance class have the same number of coding triplets N_{c}.
- Mutations in coding triplets are (almost) always deleterious.

The mutation step

We use the two simplifying assumptions that we already used for the deterministic model, namely:

- TMs in the best performance class have the same number of coding triplets N_{c}.
- Mutations in coding triplets are (almost) always deleterious.

The mutation step

We use the two simplifying assumptions that we already used for the deterministic model, namely:

- TMs in the best performance class have the same number of coding triplets N_{c}.
- Mutations in coding triplets are (almost) always deleterious.

We have:

$$P_{mut}(n_s' \to n_s'' = n_s' - k) = \left(\begin{array}{c} n_s' \\ k \end{array} \right) P^k (1-P)^{n_s'-k},$$

where P is the probability that an individual in the best performance class will undergo at least one mutation into a coding triplet

$$P = 1 - (1 - p_{\mathsf{m}})^{3N_{\mathsf{c}}}.$$

The Markov matrix

In our case the state of the system is labelled by the number n_s of individuals into the best performance class and the entries of M will be given by

$$M_{n_s+1,n_s''+1} = \sum_{n_s'=0}^{N} P_{rip}(n_s \to n_s') P_{mut}(n_s' \to n_s''), \qquad n_s, n_s'' = 0, \ldots, N$$

The Markov matrix

In our case the state of the system is labelled by the number n_s of individuals into the best performance class and the entries of M will be given by

$$M_{n_s+1,n_s''+1} = \sum_{n_s'=0}^{N} P_{rip}(n_s \to n_s') P_{mut}(n_s' \to n_s''), \qquad n_s, n_s'' = 0, \ldots, N$$

The state $n_s'' = 0$ is the only absorbing state for M.

The Markov matrix

In our case the state of the system is labelled by the number n_s of individuals into the best performance class and the entries of M will be given by

$$M_{n_s+1,n_s''+1} = \sum_{n_s'=0}^{N} P_{rip}(n_s \to n_s') P_{mut}(n_s' \to n_s''), \qquad n_s, n_s'' = 0, \ldots, N$$

The state $n_s'' = 0$ is the only absorbing state for M. We can use M to compute the expected number of generations τ for reaching it.

The Markov matrix

In our case the state of the system is labelled by the number n_s of individuals into the best performance class and the entries of M will be given by

$$M_{n_s+1,n_s''+1} = \sum_{n_s'=0}^{N} P_{rip}(n_s \to n_s') P_{mut}(n_s' \to n_s''), \qquad n_s, n_s'' = 0, \ldots, N$$

The state $n_s'' = 0$ is the only absorbing state for M. We can use M to compute the expected number of generations τ for reaching it. Let S be the matrix that one obtains by removing the first row and the first column corresponding to the only absorbing state and let $\mathbf{c}$ be a $N-$dimensional vector whose entries are all one. The matrix $\mathbb{I} - S$, where $\mathbb{I}$ denotes the identity matrix, is invertible.

The Markov matrix

In our case the state of the system is labelled by the number n_s of individuals into the best performance class and the entries of M will be given by

$$M_{n_s+1,n_s''+1} = \sum_{n_s'=0}^{N} P_{rip}(n_s \to n_s') P_{mut}(n_s' \to n_s''), \qquad n_s, n_s'' = 0, \ldots, N$$

The state $n_s'' = 0$ is the only absorbing state for M. We can use M to compute the expected number of generations τ for reaching it. Let S be the matrix that one obtains by removing the first row and the first column corresponding to the only absorbing state and let $\mathbf{c}$ be a $N-$dimensional vector whose entries are all one. The matrix $\mathbb{I} - S$, where $\mathbb{I}$ denotes the identity matrix, is invertible.
If the Markov process begins in the state i, then the expected number of generations before extinction will be given by:

$$\tau = \left[(\mathbb{I} - S)^{-1} \mathbf{c} \right]_i .$$

The Markov matrix

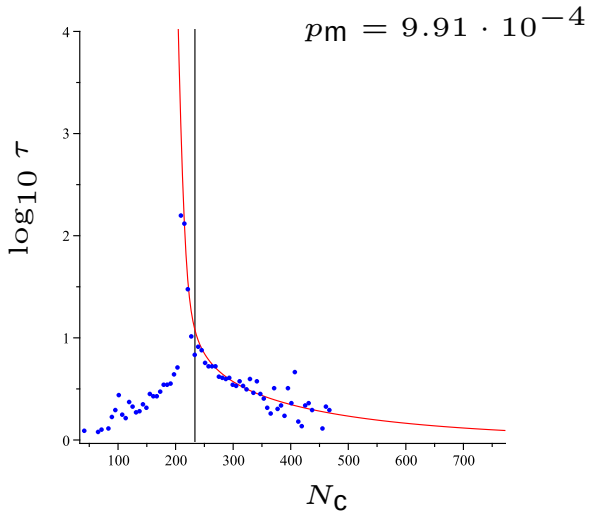
In our case the state of the system is labelled by the number n_s of individuals into the best performance class and the entries of M will be given by

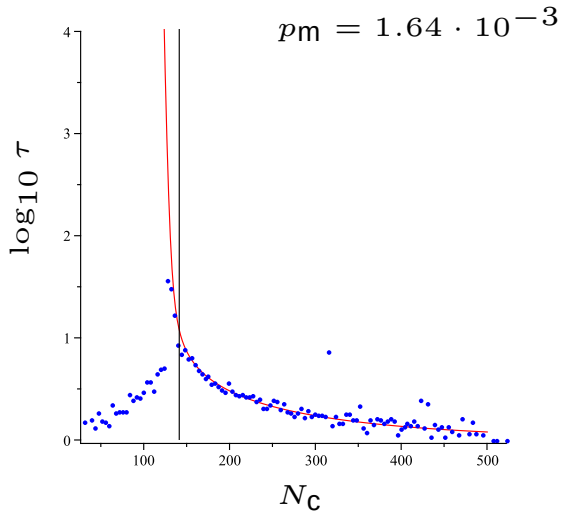
$$M_{n_s+1,n_s''+1} = \sum_{n_s'=0}^{N} P_{rip}(n_s \to n_s') P_{mut}(n_s' \to n_s''), \qquad n_s, n_s'' = 0, \ldots, N$$

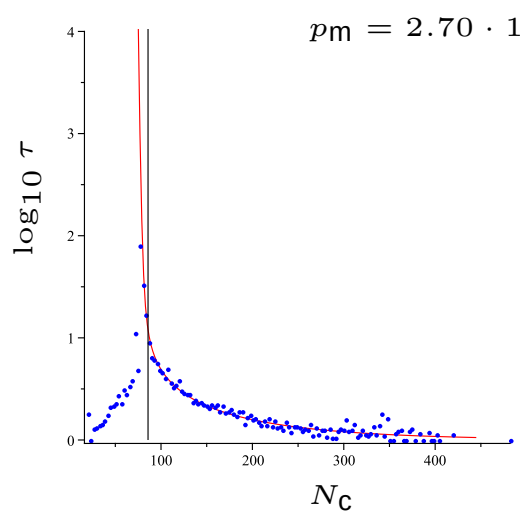
The state $n_s'' = 0$ is the only absorbing state for M. We can use M to compute the expected number of generations τ for reaching it. Let S be the matrix that one obtains by removing the first row and the first column corresponding to the only absorbing state and let $\mathbf{c}$ be a $N-$dimensional vector whose entries are all one. The matrix $\mathbb{I} - S$, where $\mathbb{I}$ denotes the identity matrix, is invertible.
If the Markov process begins in the state i, then the expected number of generations before extinction will be given by:

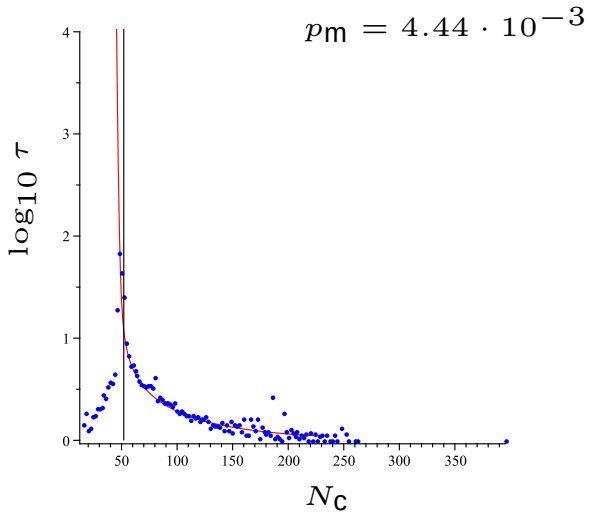
$$\tau = \left[(\mathbb{I} - S)^{-1} \mathbf{c} \right]_i .$$

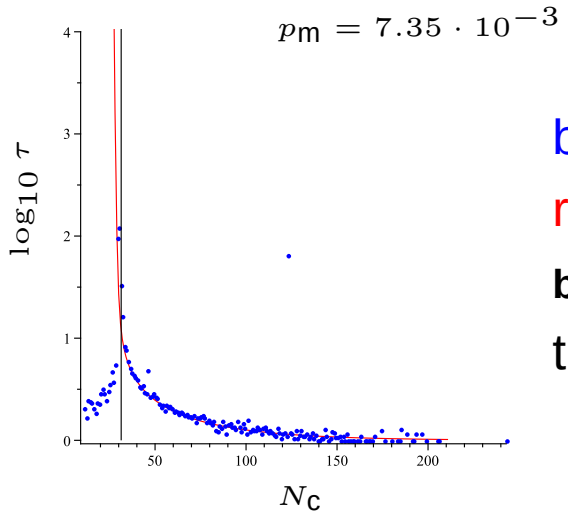
This equation assumes an infinite number of generations.









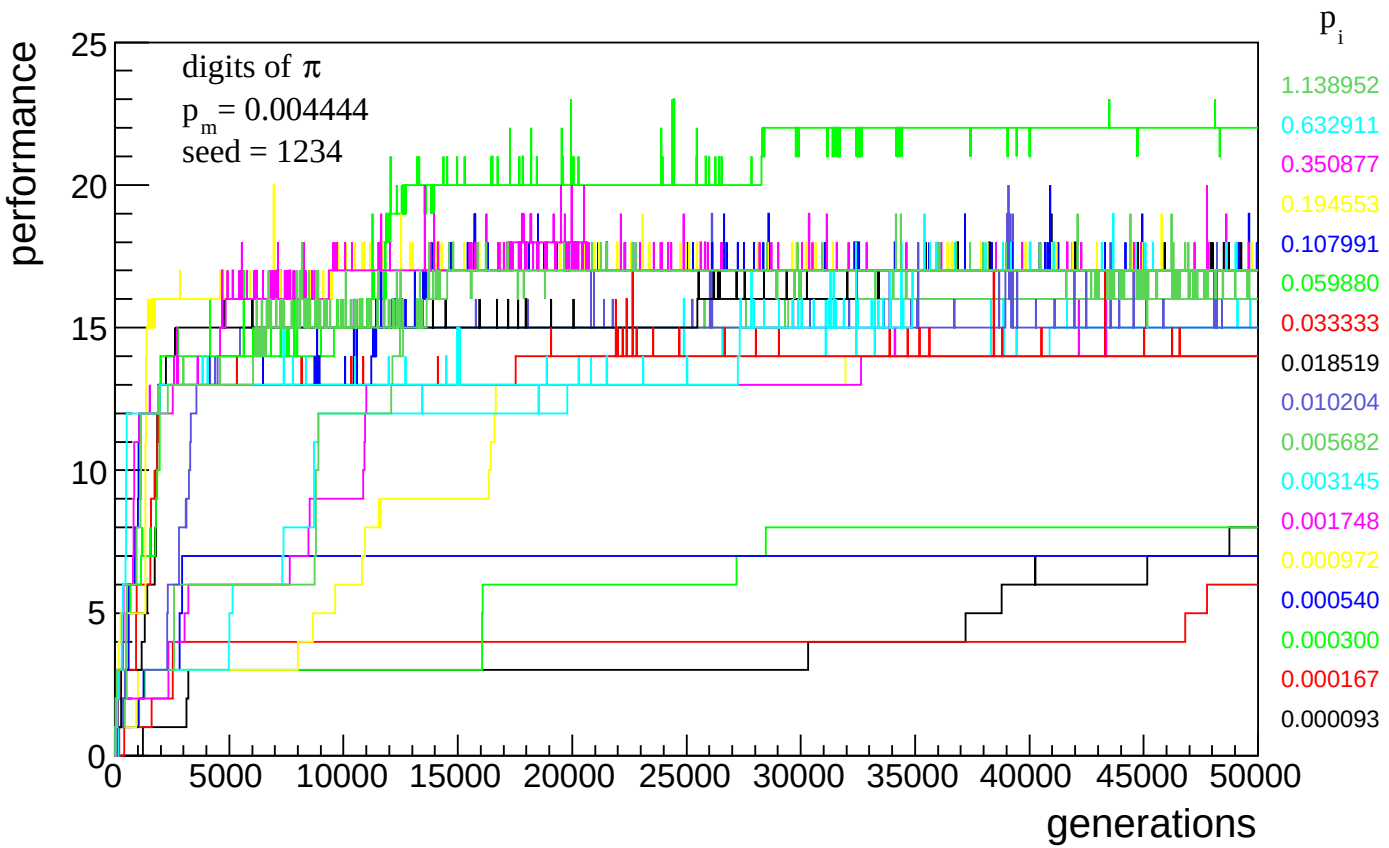


blue points: simulations data

red line: stochastic model

black line: deterministic error threshold

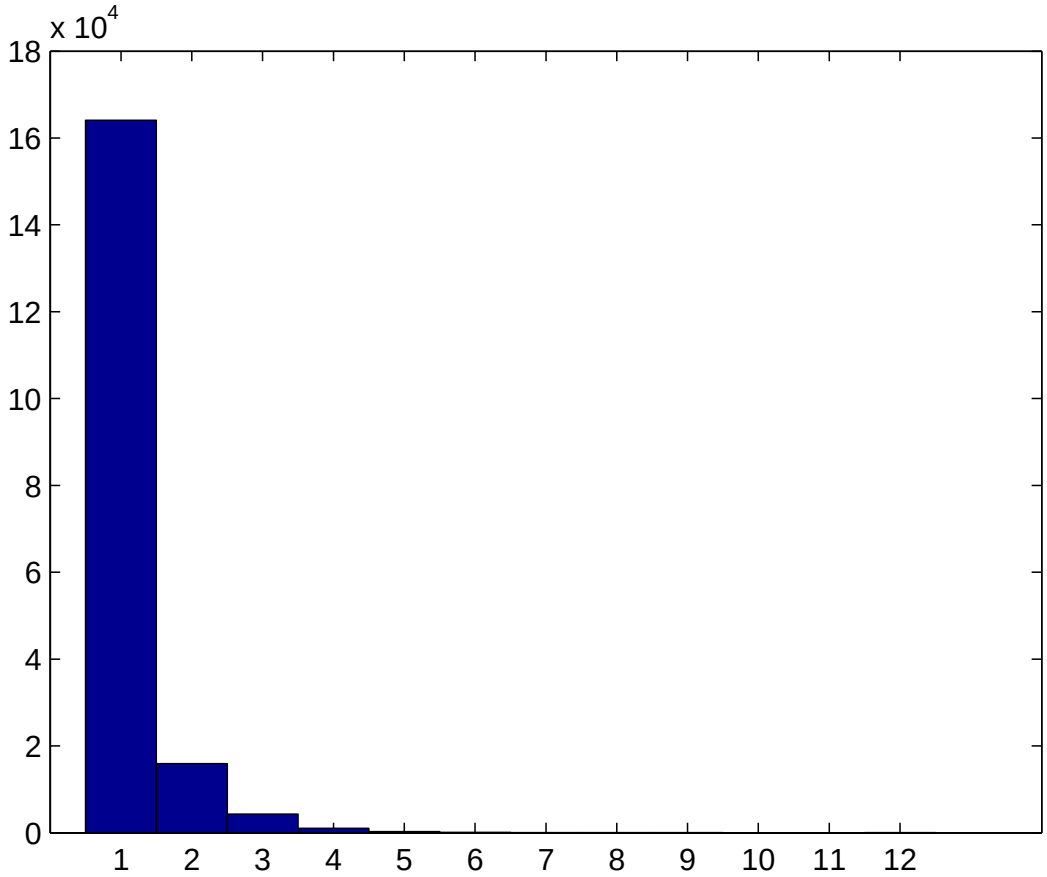
Life near the error threshold



For $p_{\mathrm{m}} = 0.0044$ and certain values of p_{i}, TMs reach the error threshold.

From there on, a typical oscillatory pattern emerges.

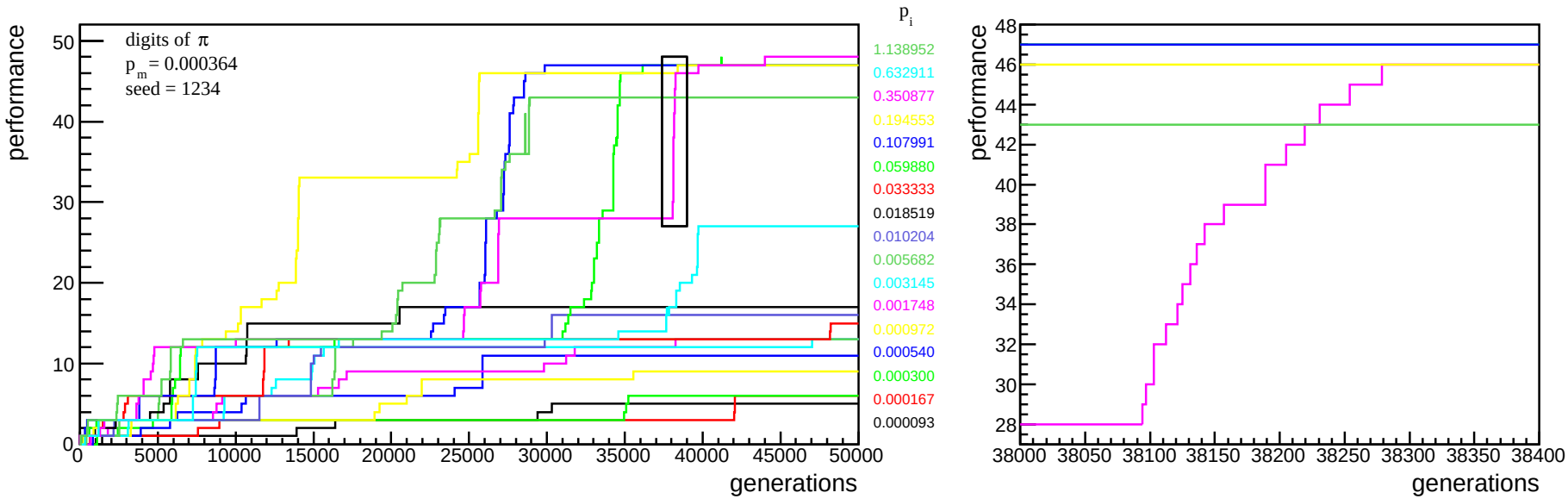
Performance jumps



This histogram shows the number of increases in the performance versus their amplitude.

Performance shows a gradualistic evolution.

Punctuated equilibria



We observe the presence of long stasis periods alternated by short periods of fast evolution.

The small black rectangle is zoomed on in the right part of the figure to show the actual jumps in the performance.

Conclusions

Simulations of our evolutionary model for TMs showed that they evolve toward the error threshold.

Conclusions

Simulations of our evolutionary model for TMs showed that they evolve toward the error threshold.

Mathematical models show that this is due to the mutation-selection dynamics and to the fact that TMs satisfy three basic assumptions:

Conclusions

Simulations of our evolutionary model for TMs showed that they evolve toward the error threshold.
Mathematical models show that this is due to the mutation-selection dynamics and to the fact that TMs satisfy three basic assumptions:

- there is no perfect replicator
- fidelity rate and performance are negatively correlated
- the probability of a beneficial mutation is small but never exceedingly small

Conclusions

Simulations of our evolutionary model for TMs showed that they evolve toward the error threshold.
Mathematical models show that this is due to the mutation-selection dynamics and to the fact that TMs satisfy three basic assumptions:

- there is no perfect replicator
- fidelity rate and performance are negatively correlated
- the probability of a beneficial mutation is small but never exceedingly small

The fact that this behaviour could be much more general than the present model only is suggested by the fact that it emerges also in a completely different evolutionary model (Knibbe et al 2007) that uses completely different algorithms coding, mutation and selection operators.

RNA virus have error rates (per genome per replication) near to one and have been suggested to replicate near the error threshold (Eigen 2000) in accordance with the behaviour we observe in our model.

RNA virus have error rates (per genome per replication) near to one and have been suggested to replicate near the error threshold (Eigen 2000) in accordance with the behaviour we observe in our model.
DNA based organisms have a much smaller but remarkably constant mutation rate $\simeq 10^{-2},\ 10^{-3}$ (Drake 1998).

RNA virus have error rates (per genome per replication) near to one and have been suggested to replicate near the error threshold (Eigen 2000) in accordance with the behaviour we observe in our model.

DNA based organisms have a much smaller but remarkably constant mutation rate $\simeq 10^{-2},\ 10^{-3}$ (Drake 1998).

Eigen suggested that this constancy could be due to the fact that also DNA based organisms do reproduce near the error threshold. The smaller error rate could be due to a larger number of neutrals in DNA organisms and/or to the dissymmetry between the two daughter strands error rates.

RNA virus have error rates (per genome per replication) near to one and have been suggested to replicate near the error threshold (Eigen 2000) in accordance with the behaviour we observe in our model.

DNA based organisms have a much smaller but remarkably constant mutation rate $\simeq 10^{-2},\ 10^{-3}$ (Drake 1998).

Eigen suggested that this constancy could be due to the fact that also DNA based organisms do reproduce near the error threshold. The smaller error rate could be due to a larger number of neutrals in DNA organisms and/or to the dissymmetry between the two daughter strands error rates.

While our results support this view, we didn't consider error correction mechanisms.

RNA virus have error rates (per genome per replication) near to one and have been suggested to replicate near the error threshold (Eigen 2000) in accordance with the behaviour we observe in our model.

DNA based organisms have a much smaller but remarkably constant mutation rate $\simeq 10^{-2},\ 10^{-3}$ (Drake 1998).

Eigen suggested that this constancy could be due to the fact that also DNA based organisms do reproduce near the error threshold. The smaller error rate could be due to a larger number of neutrals in DNA organisms and/or to the dissymmetry between the two daughter strands error rates.

While our results support this view, we didn't consider error correction mechanisms.

Approaching the error threshold put a selective pressure on the development of error correction mechanisms. The error rate could emerge as a balance between this selective pressure and the metabolic costs.

RNA virus have error rates (per genome per replication) near to one and have been suggested to replicate near the error threshold (Eigen 2000) in accordance with the behaviour we observe in our model.

DNA based organisms have a much smaller but remarkably constant mutation rate $\simeq 10^{-2},\ 10^{-3}$ (Drake 1998).

Eigen suggested that this constancy could be due to the fact that also DNA based organisms do reproduce near the error threshold. The smaller error rate could be due to a larger number of neutrals in DNA organisms and/or to the dissymmetry between the two daughter strands error rates.

While our results support this view, we didn't consider error correction mechanisms.

Approaching the error threshold put a selective pressure on the development of error correction mechanisms. The error rate could emerge as a balance between this selective pressure and the metabolic costs.