Parameter-less genetic algorithm

Fernando Lobo

Overview

- Motivation
- Review parameter setting in GAs
- Parameter-less approach
- Computer experiments

Motivation

- Traditional GAs are hard to use
- User must specify a number of parameters
- We should make life easier for users

Which parameters?

- Population size
- Selection pressure
- Crossover and mutation probabilities

Other design choices

- Representation
- Variation operators

Parameter setting in GAs

- Empirical studies
- Parameter adaptation techniques
- Facetwise theoretical studies

Empirical studies

- De Jong (1975), Schaffer et al. (1989)
  - population size: 50-100
  - prob. crossover: 0.6-0.9
  - prob. mutation: 0.001-0.01

Parameter adaptation techniques

- Parameter values change throughout the search
- Lots of work on operator probabilities
- Very little on population sizing

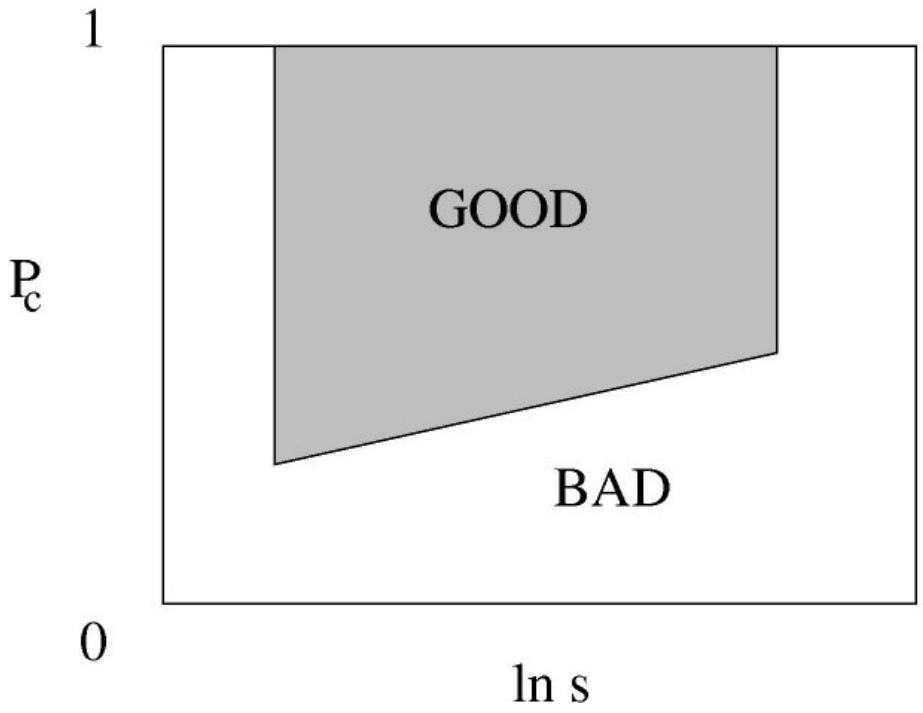
Facetwise theoretical studies

- Selection alone
- Mutation alone
- Population sizing ignoring mutation and assuming perfect mixing
- …

Two critical facetwise studies

- Control maps
- Population sizing

Selection rate and crossover prob.



Goldberg, Deb, & Thierens (1993)

Selection rate and crossover prob.

- Avoid very high and very low selection rates
- Ensure BB growth: $s\ (1\text{-}p_c) > 1$
- s=4, p_c=0.5 => $s\ (1\text{-}p_c) = 2$

Population sizing theory

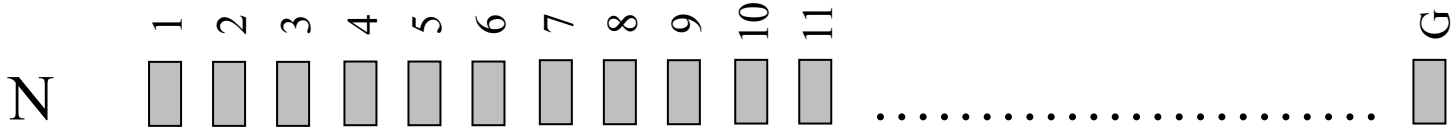
- Goldberg, Deb, Clark (1991)
  Harik, Cantú-Paz, Goldberg, Miller (1997)
- Not easy to apply for real world problems
- But very important to understand the role of the population in GAs

The intuition for population sizing in GAs

- Difficult problems require more processing power than easy problems.
- More processing power => larger population

What happens in practice?

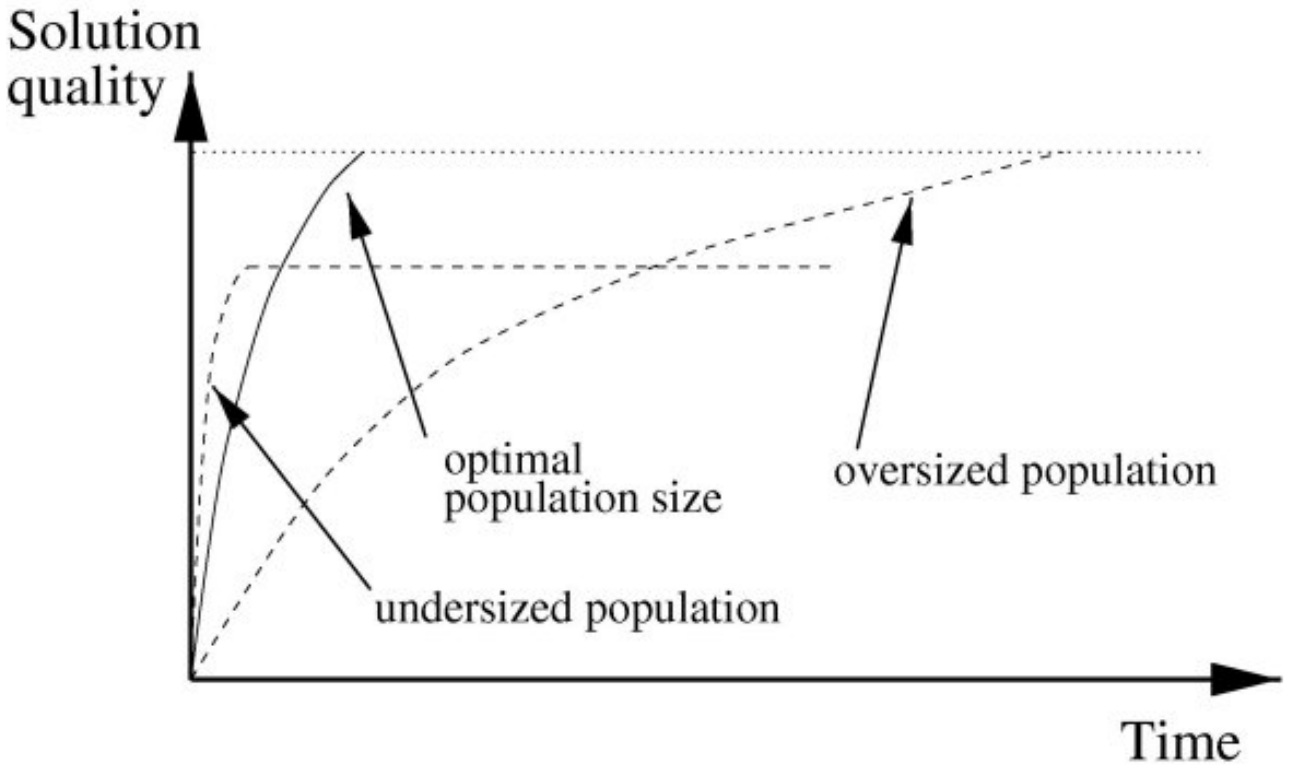
- User guesses a population size (N), and let it run a number of generations (G)



Guessing right is luck

- What if N is too small?
- What if N is too large?

This is the result

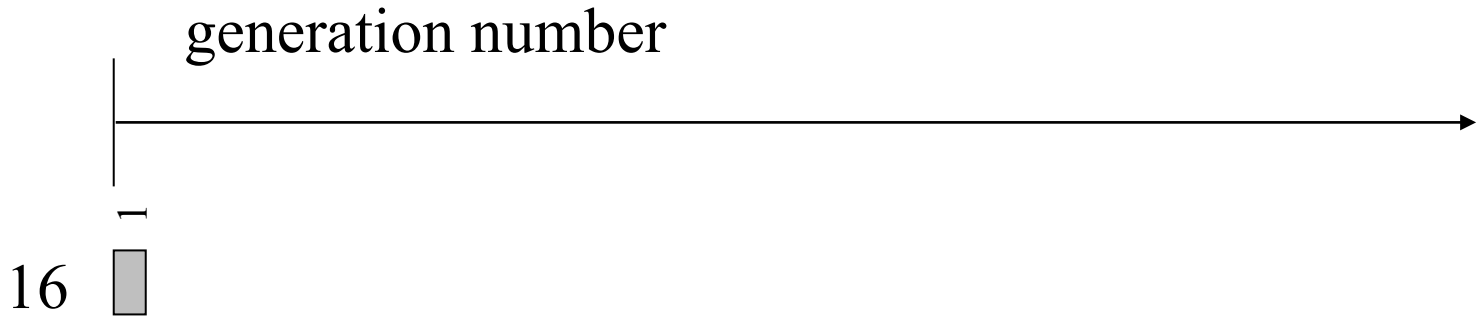


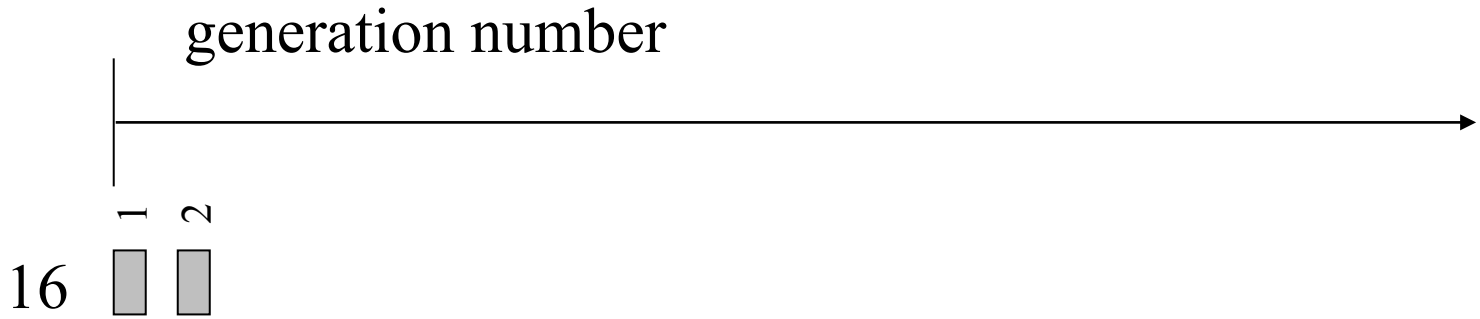
Parameter-less GA approach

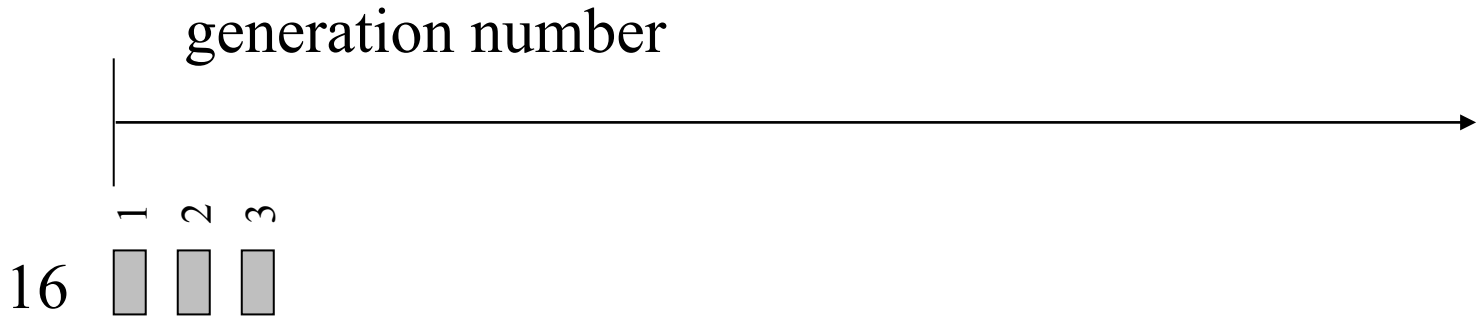
- Start with a small population size and let it run
- After some time, spawn a new population twice as large, and let it run
- And so on

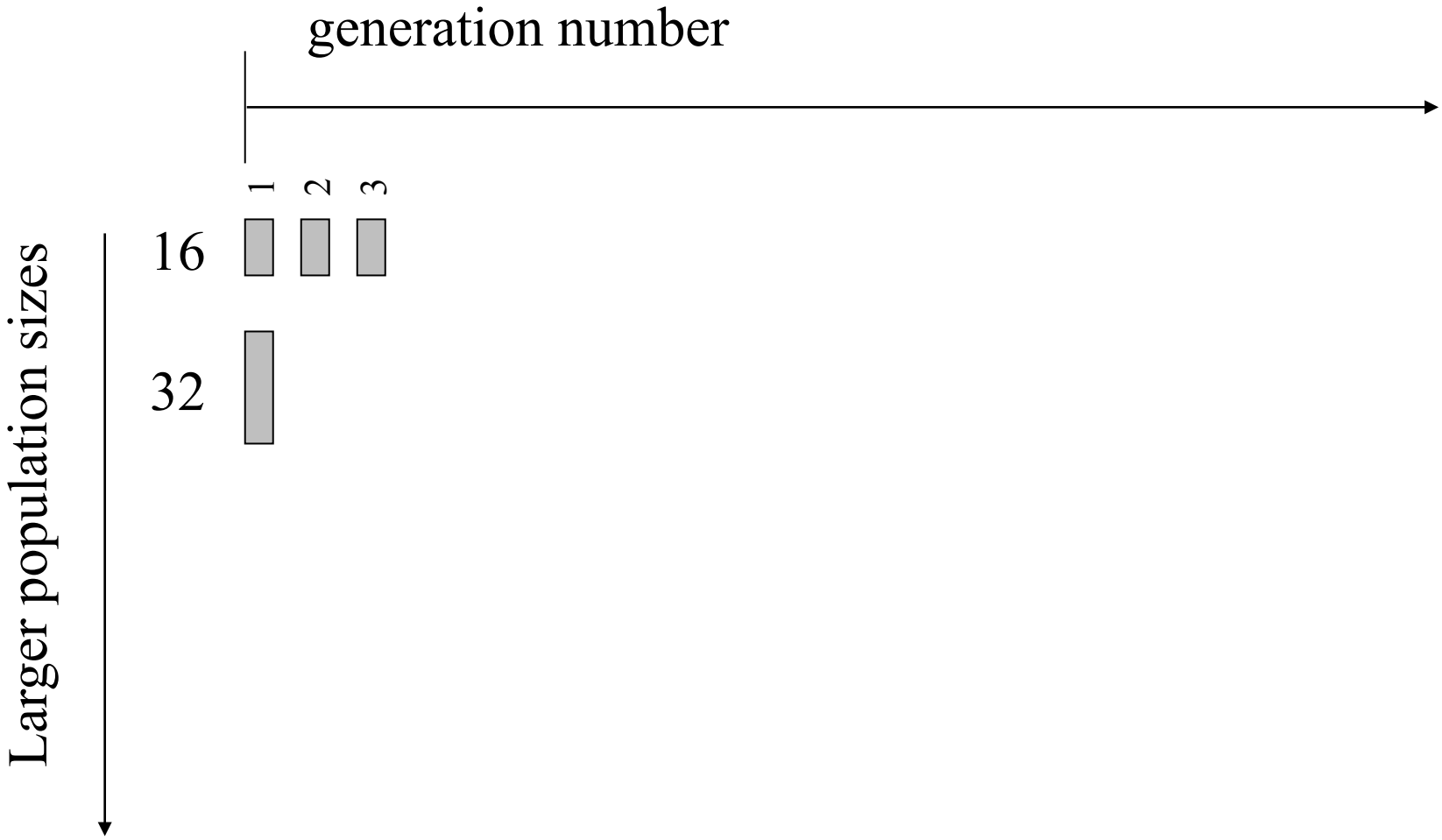
Parameter-less GA approach

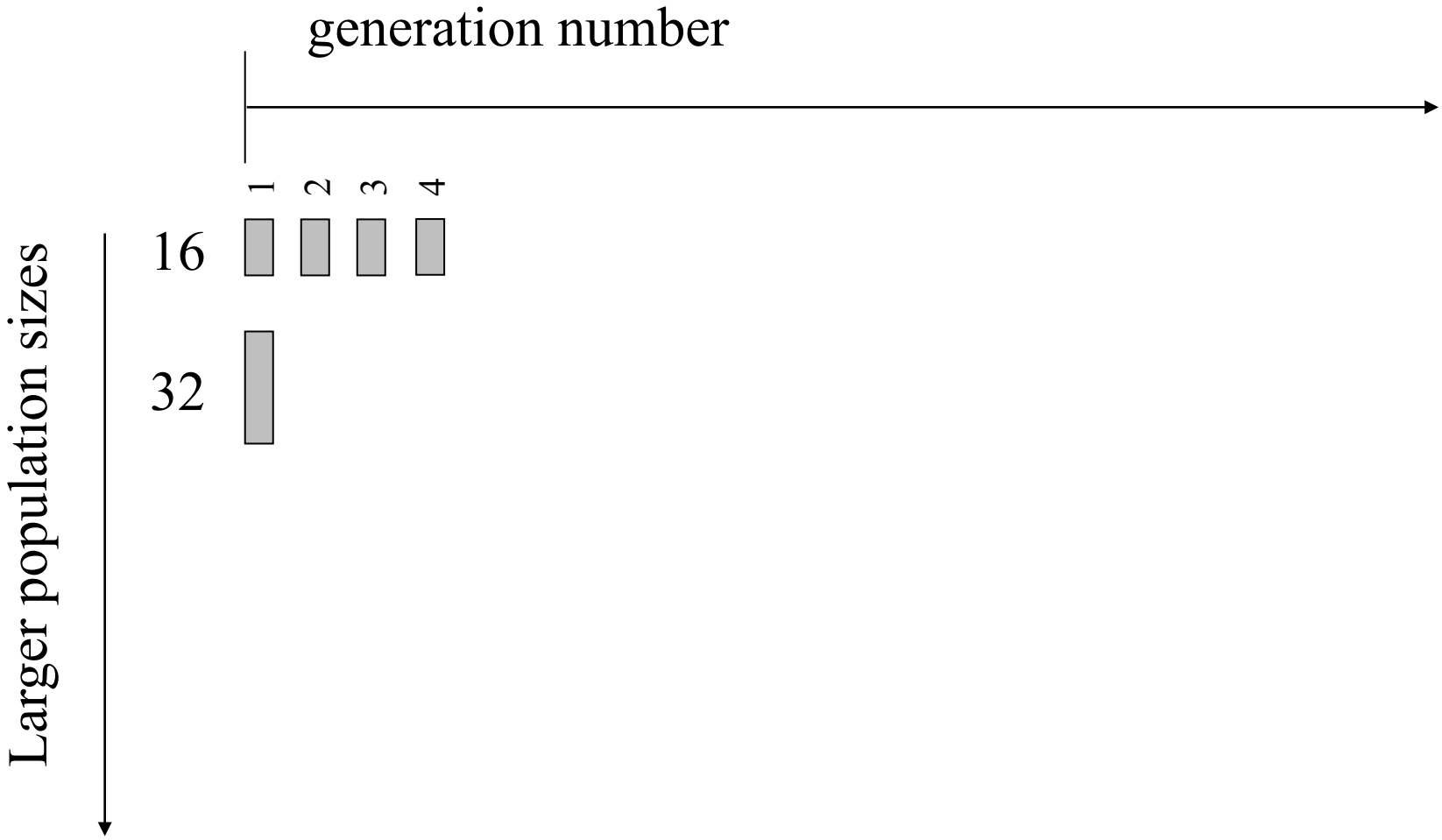
- Establish a race among populations of different sizes
- … giving a head start to the smaller ones

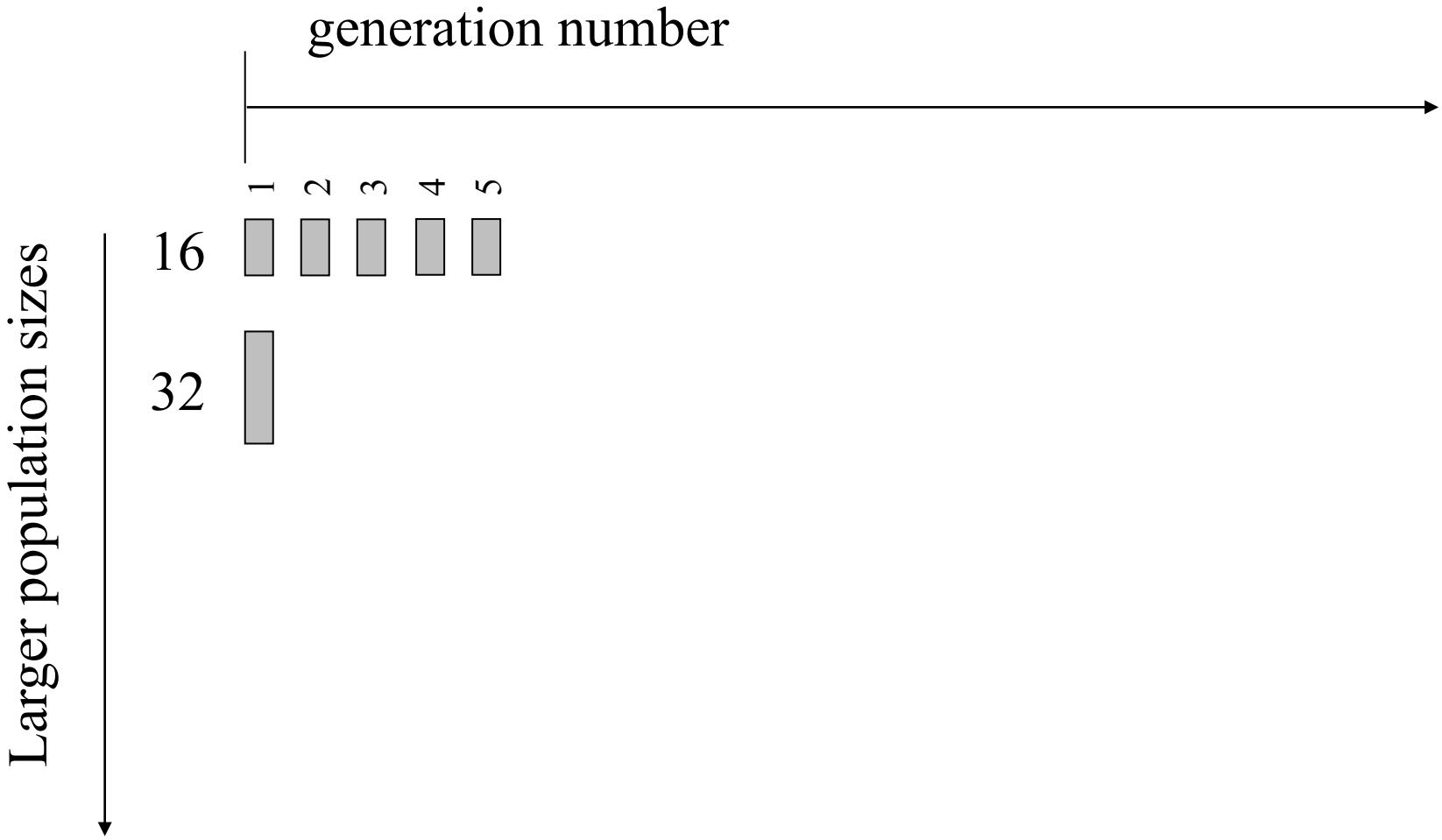

generation number
1
16

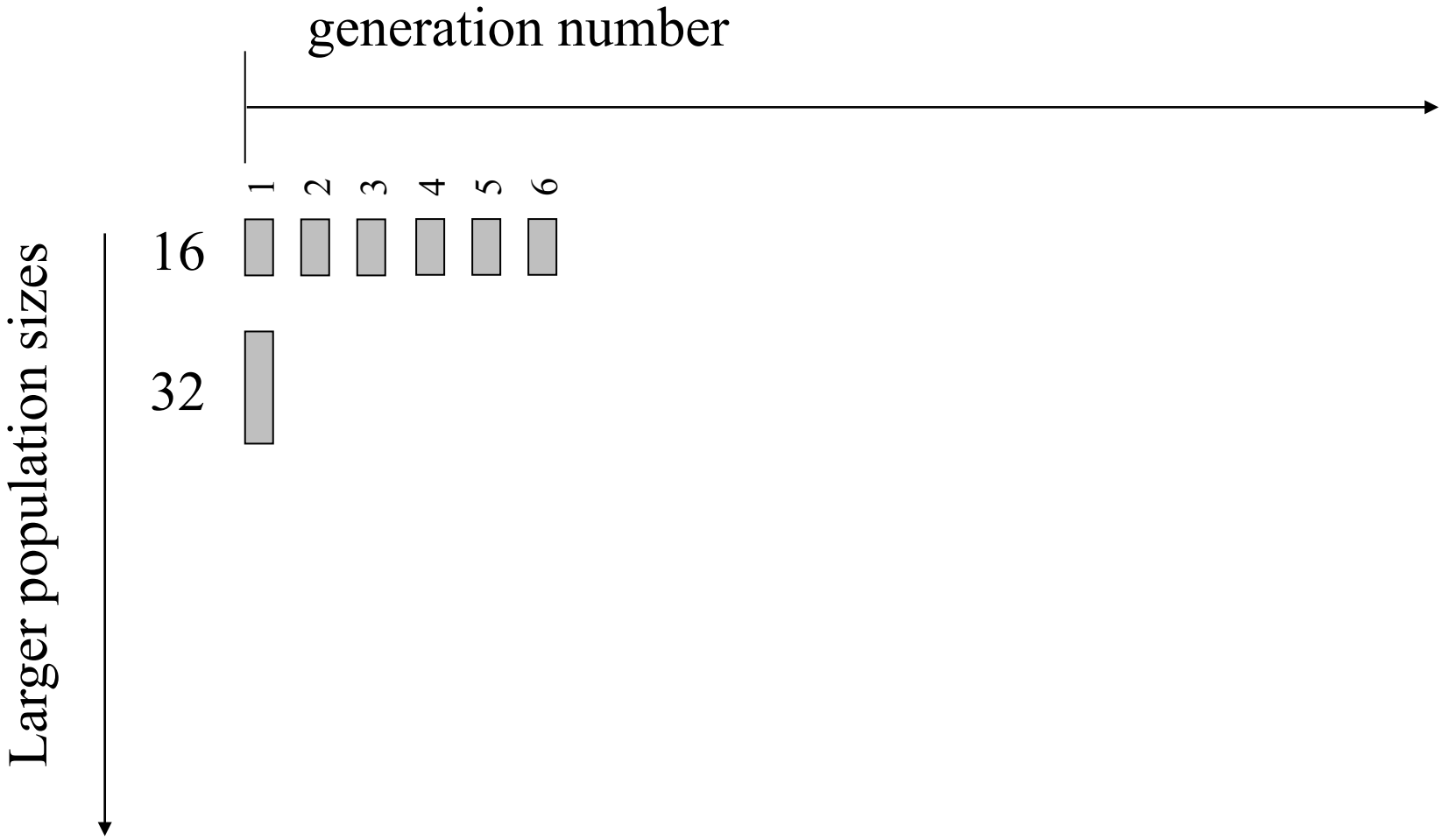

generation number
1
2
16

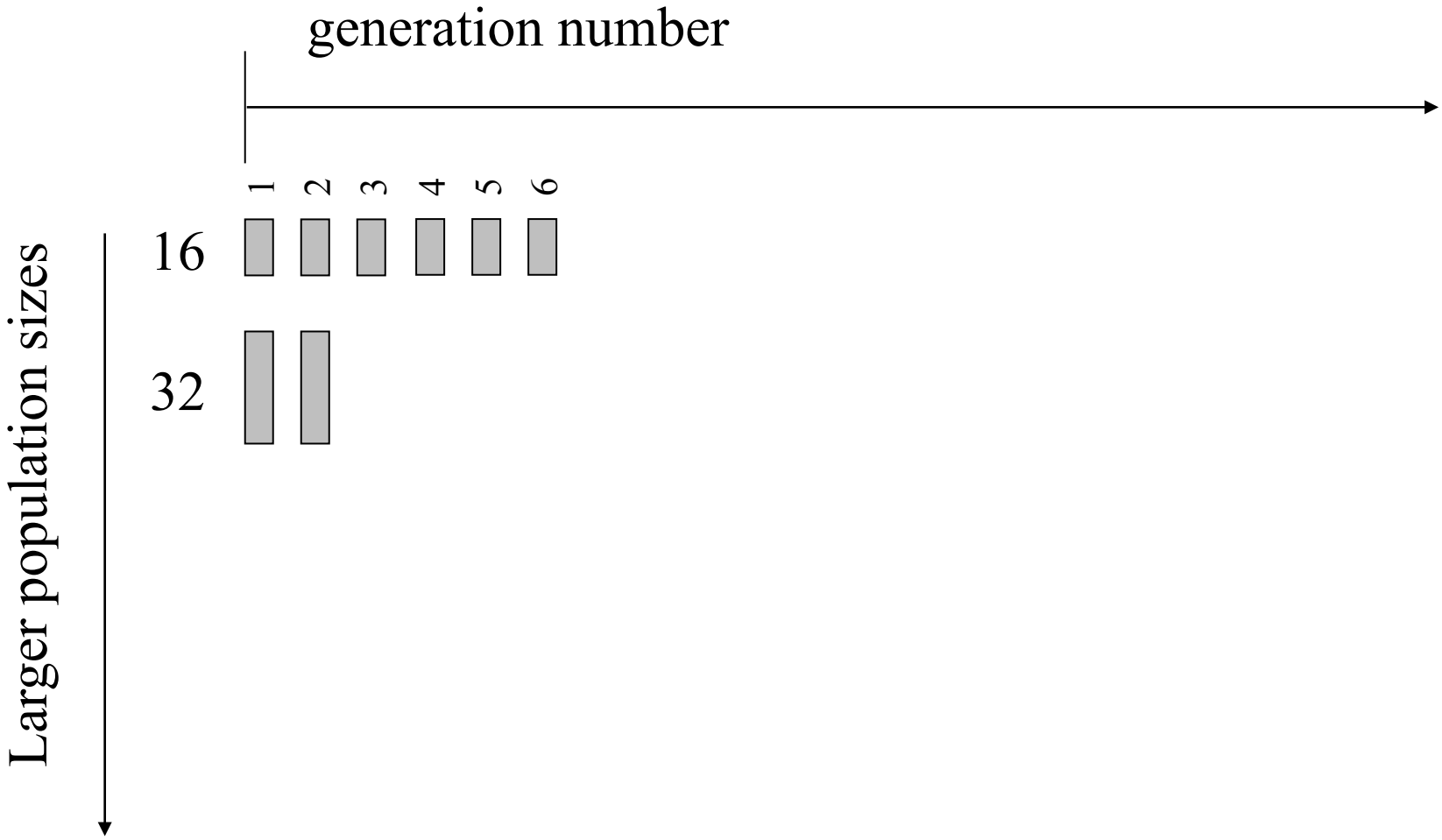

generation number
1
2
3
16

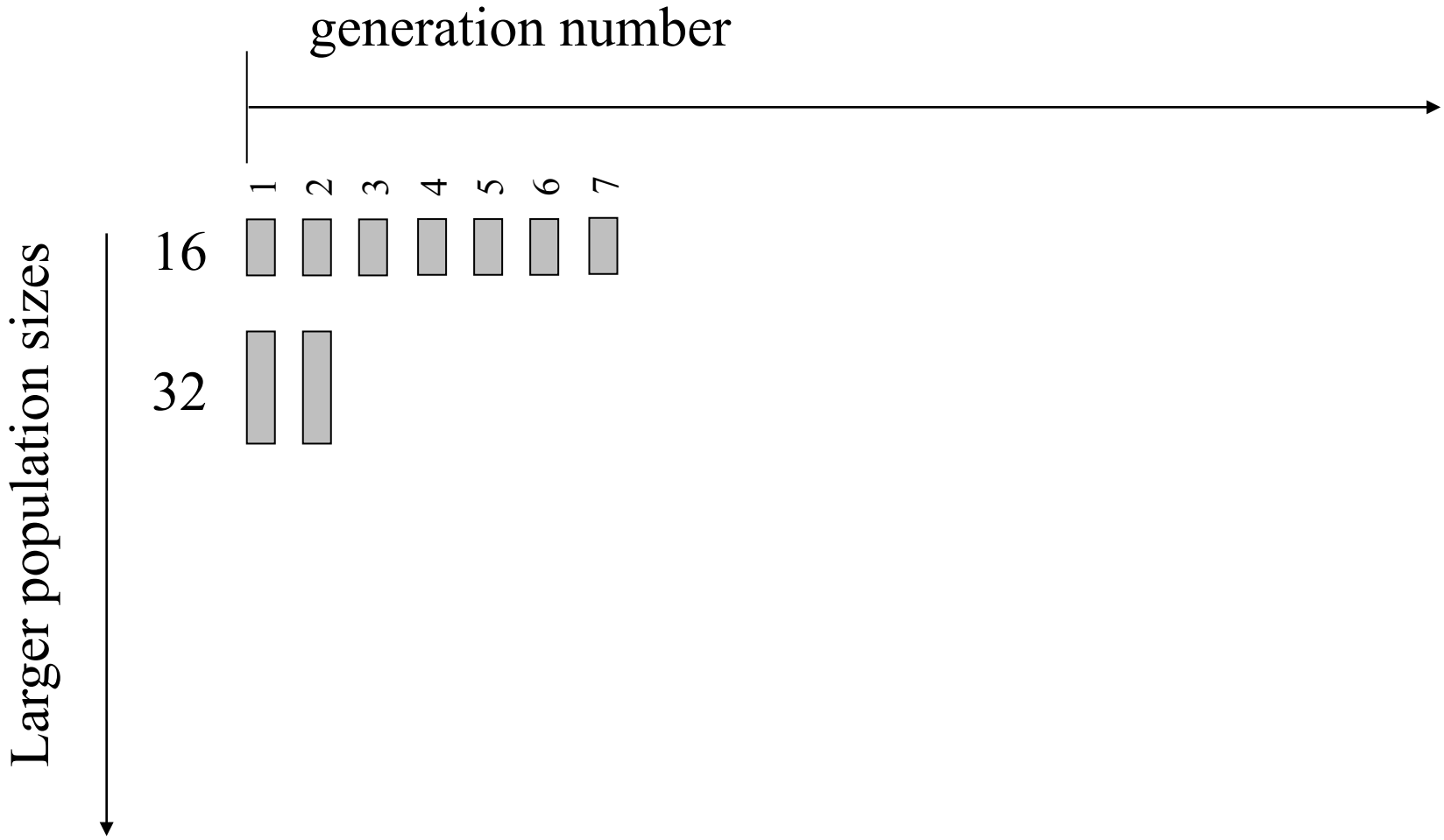

generation number
1
2
3
16
32
Larger population sizes

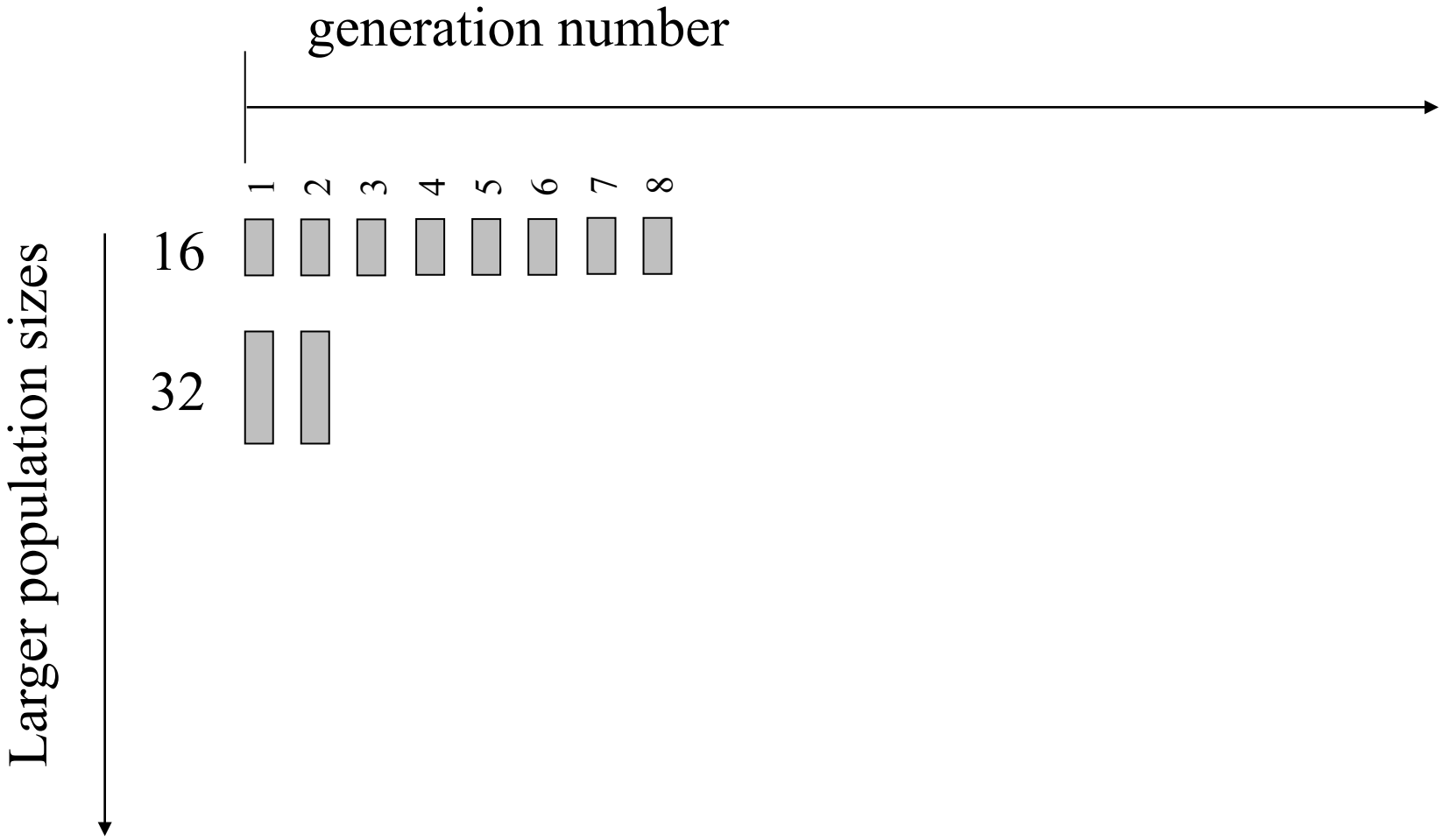

generation number
1
2
3
4
16
32
Larger population sizes

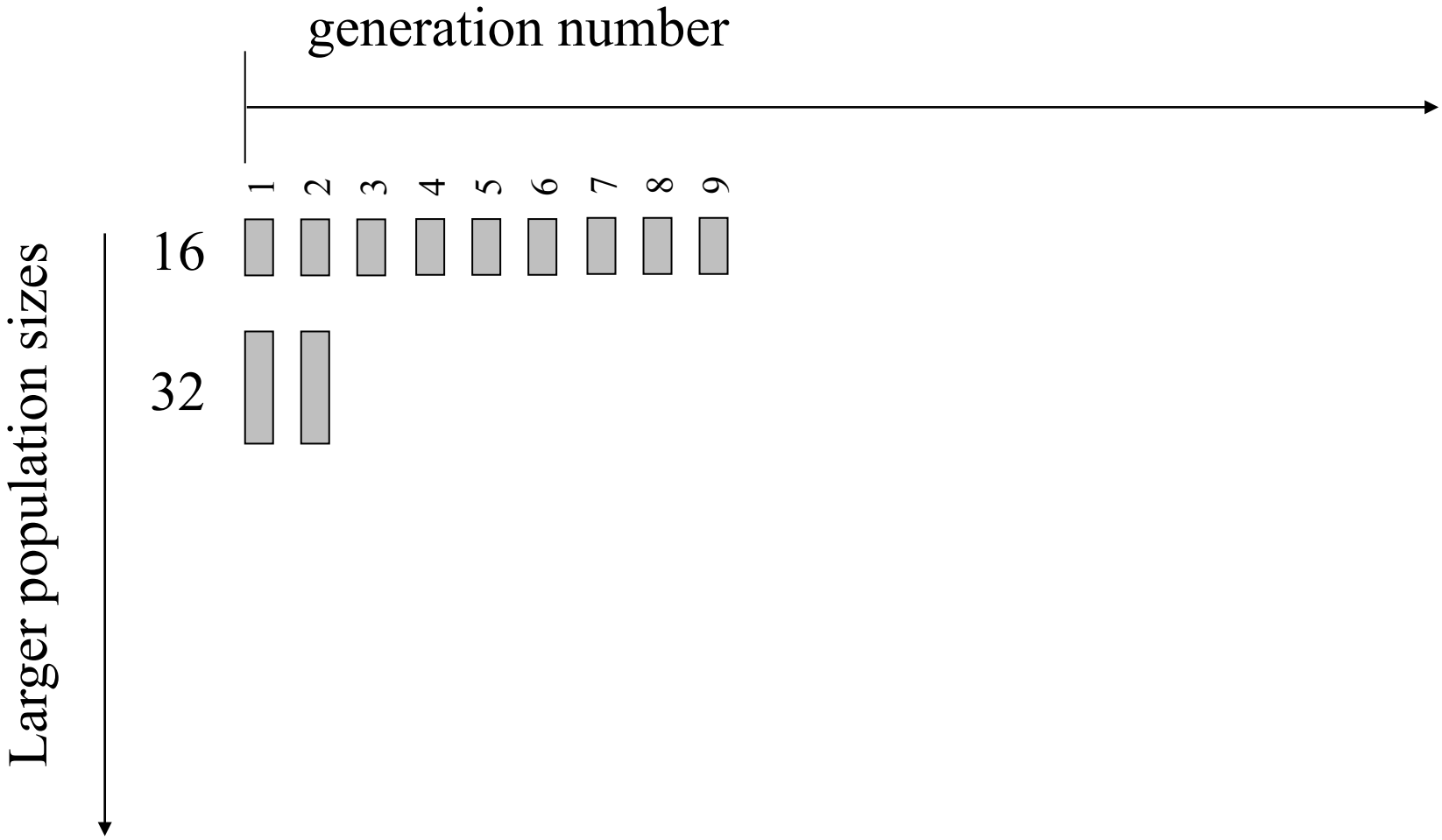

generation number
1
2
3
4
5
16
32
Larger population sizes

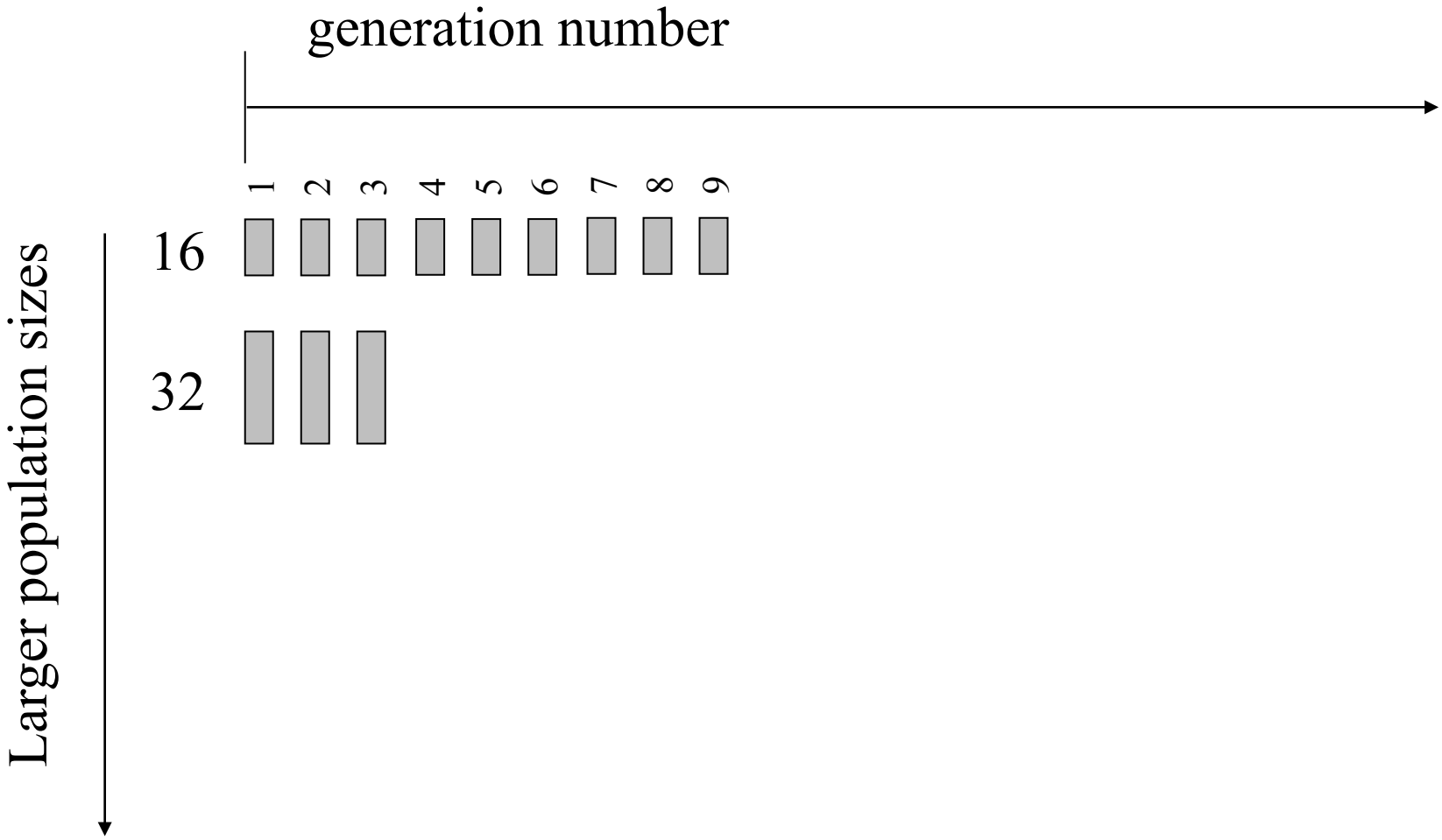

generation number
1
2
3
4
5
6
16
32
Larger population sizes

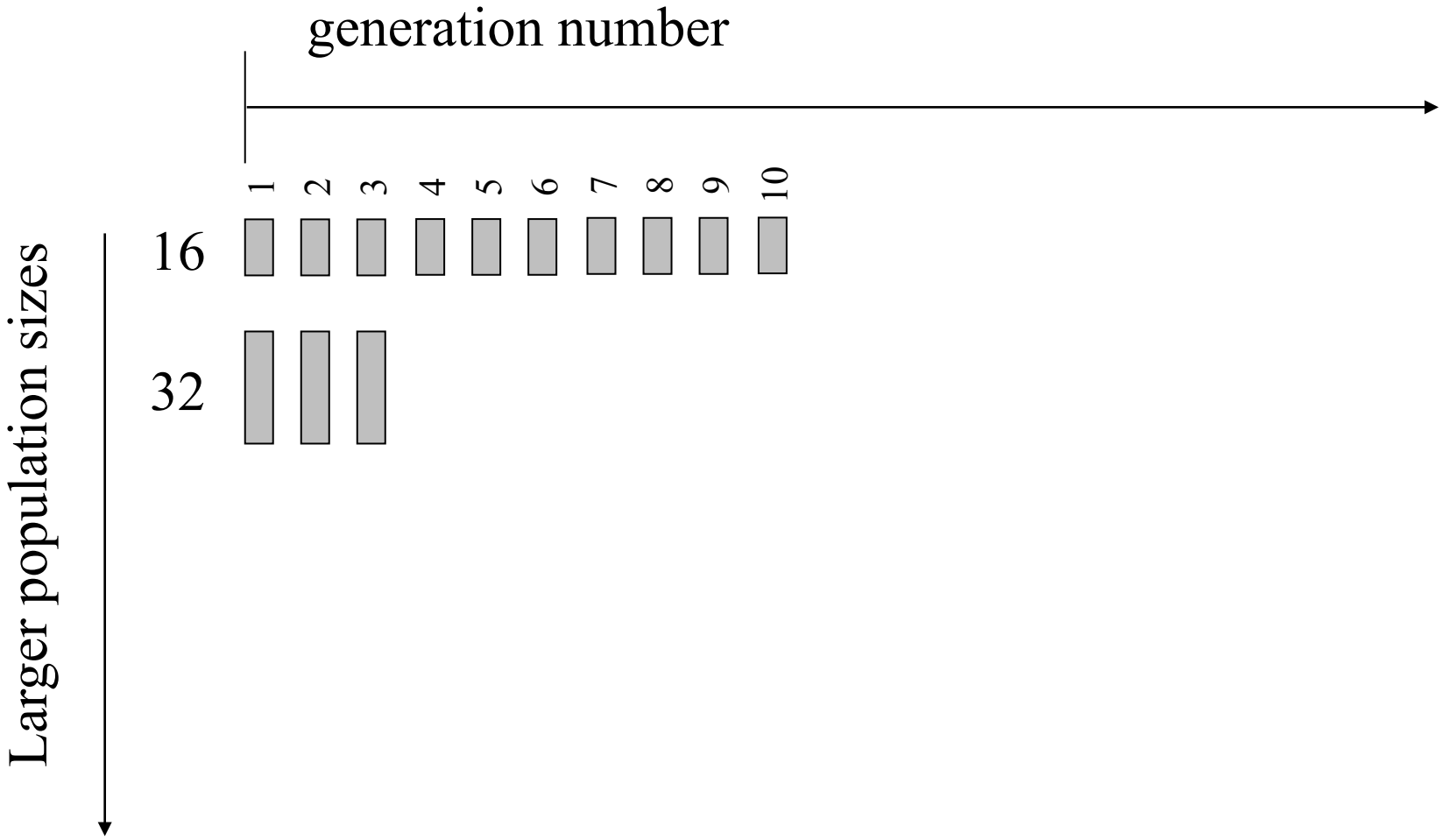

generation number
1
2
3
4
5
6
16
32
Larger population sizes

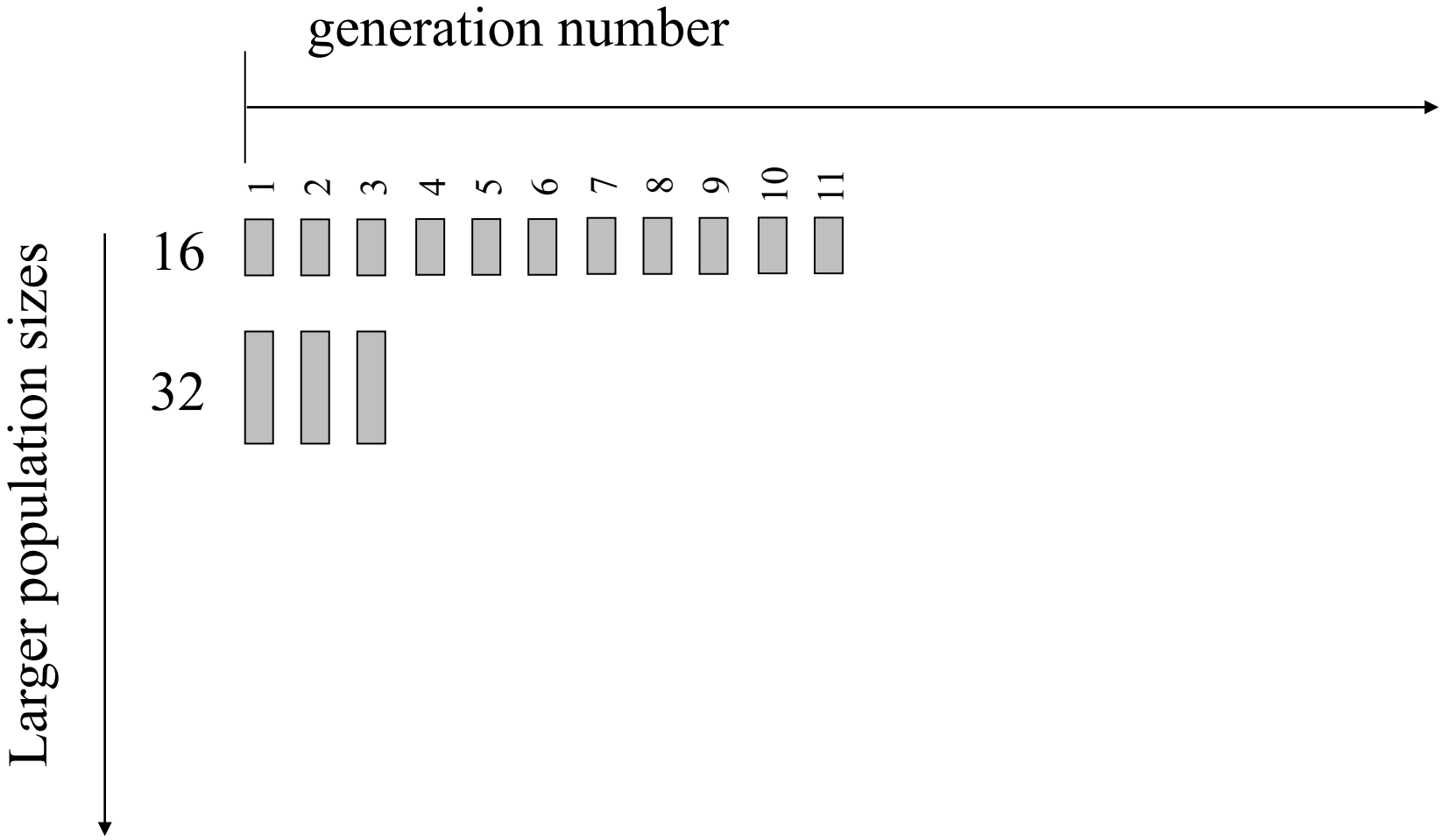

generation number
1
2
3
4
5
6
7
16
32
Larger population sizes


generation number
1
2
3
4
5
6
7
8
16
32
Larger population sizes

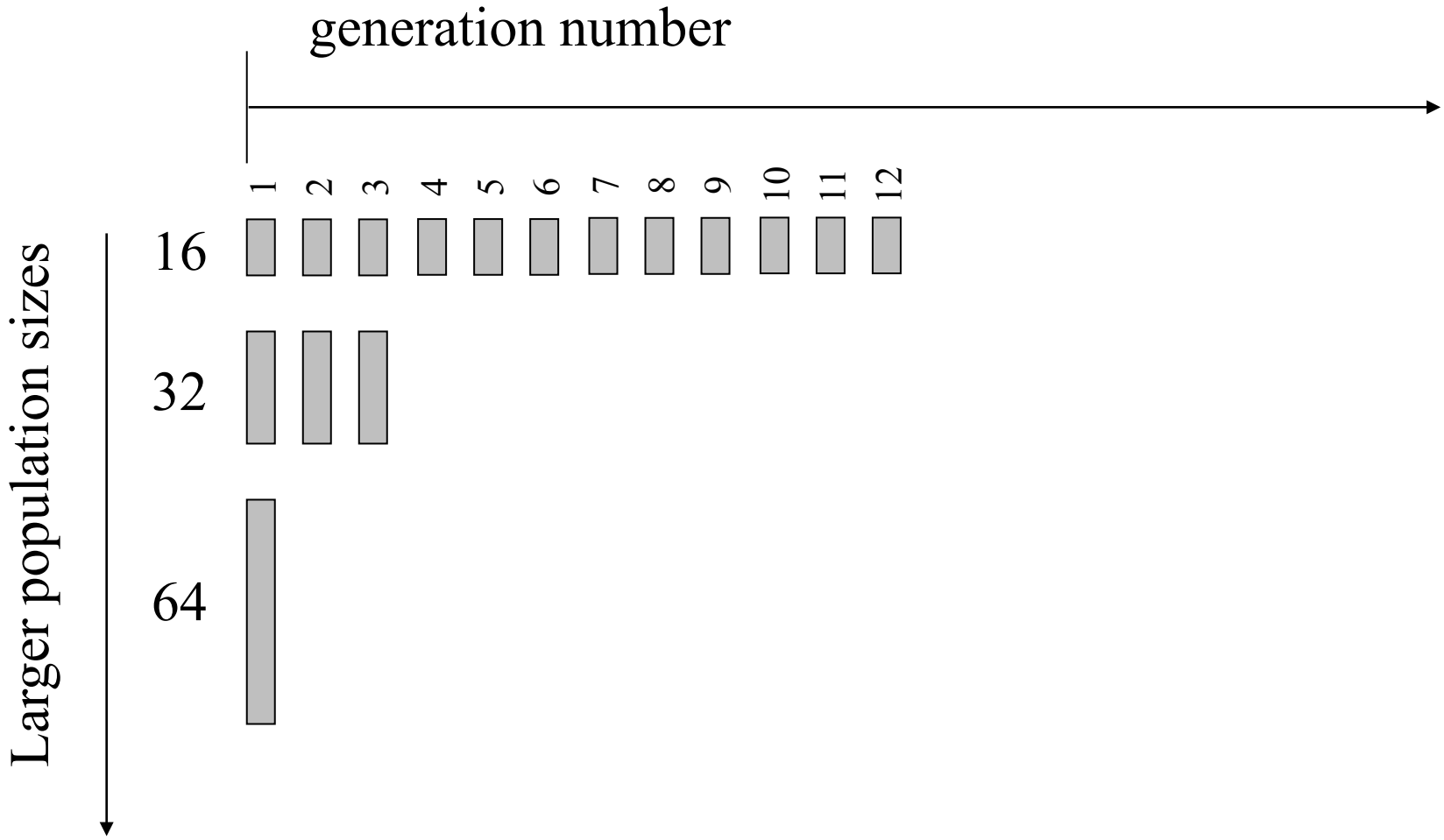

generation number
1
2
3
4
5
6
7
8
9
16
32
Larger population sizes

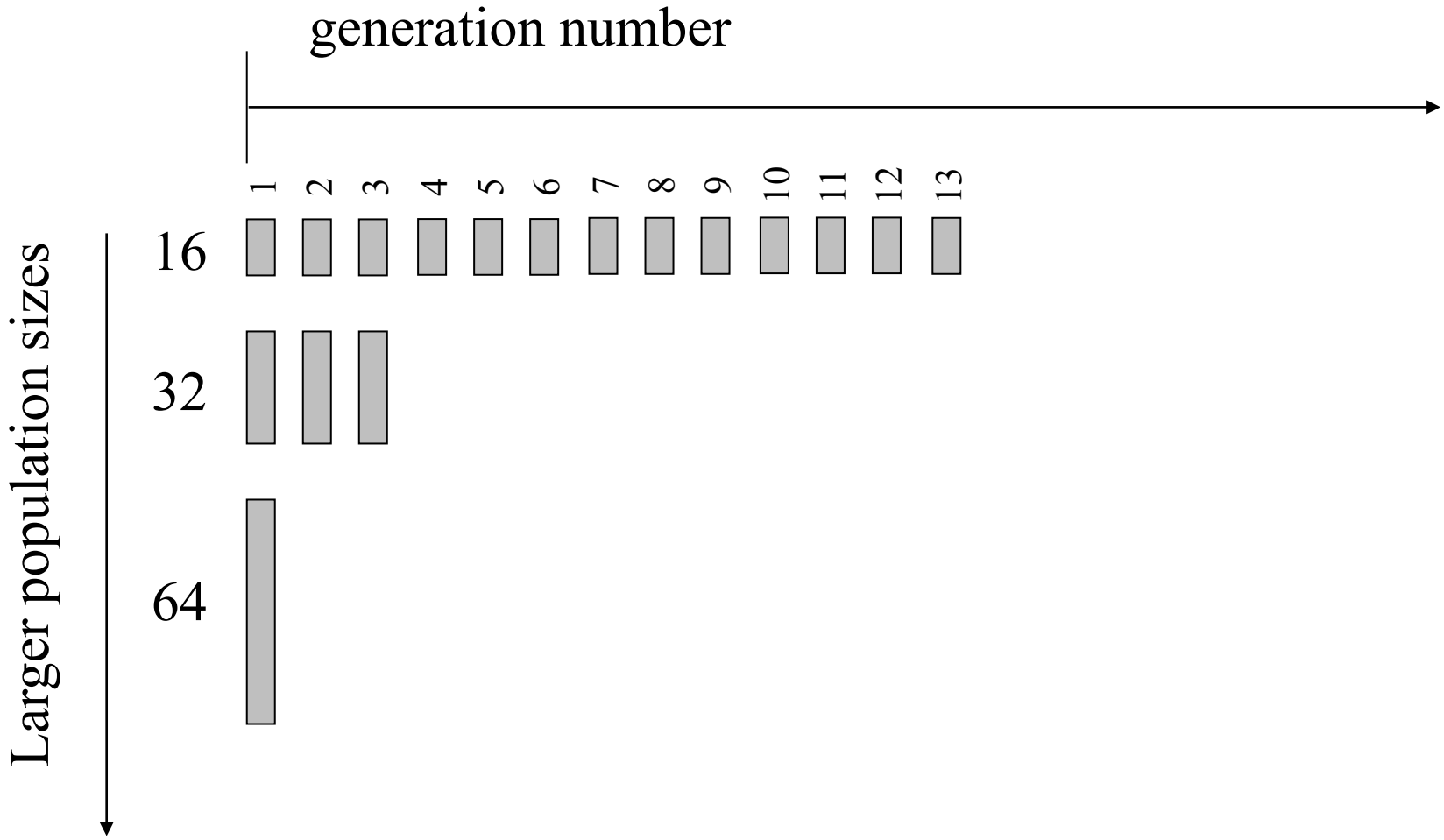

generation number
1
2
3
4
5
6
7
8
9
16
32
Larger population sizes

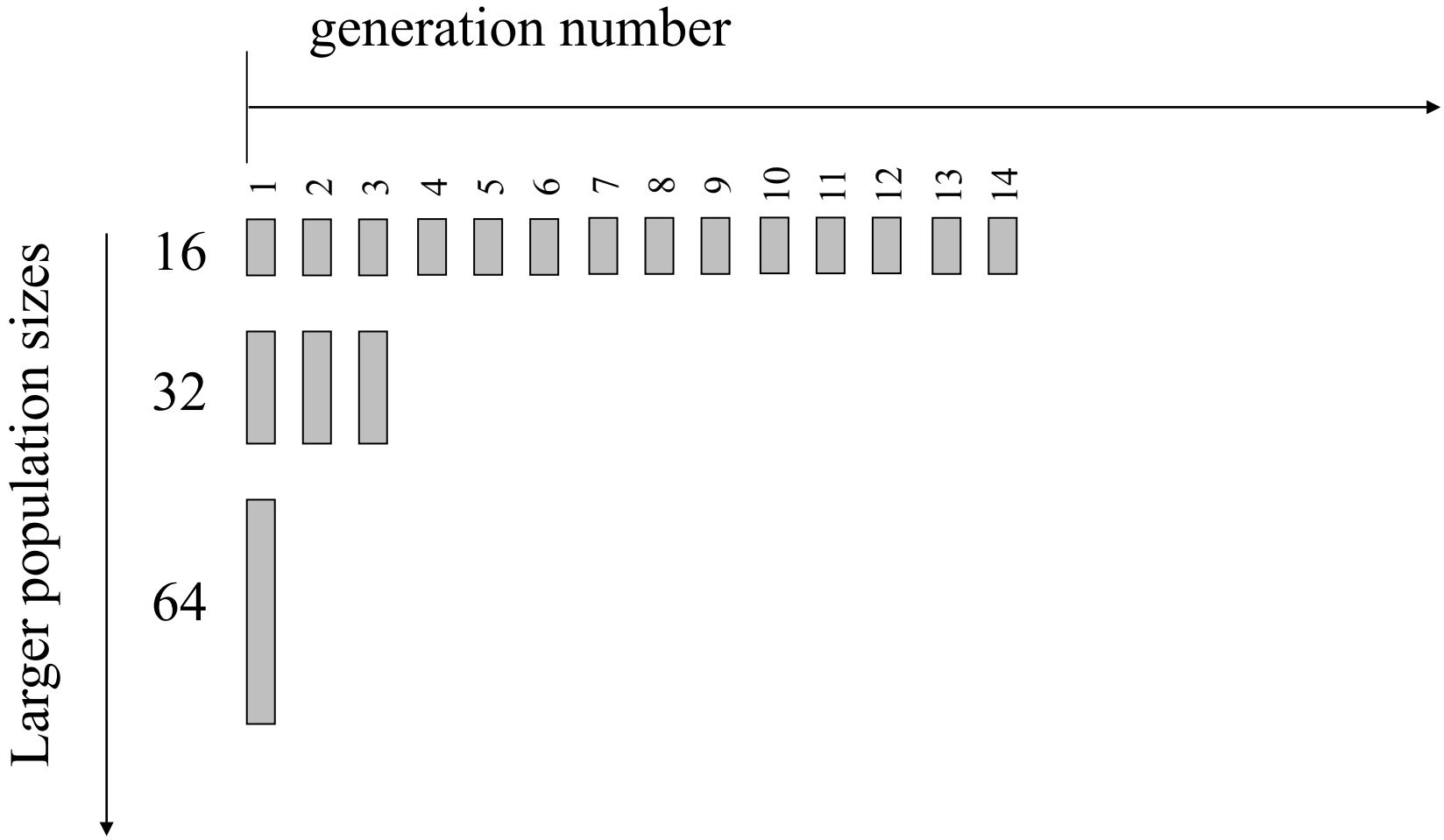

generation number
1
2
3
4
5
6
7
8
9
10
16
32
Larger population sizes

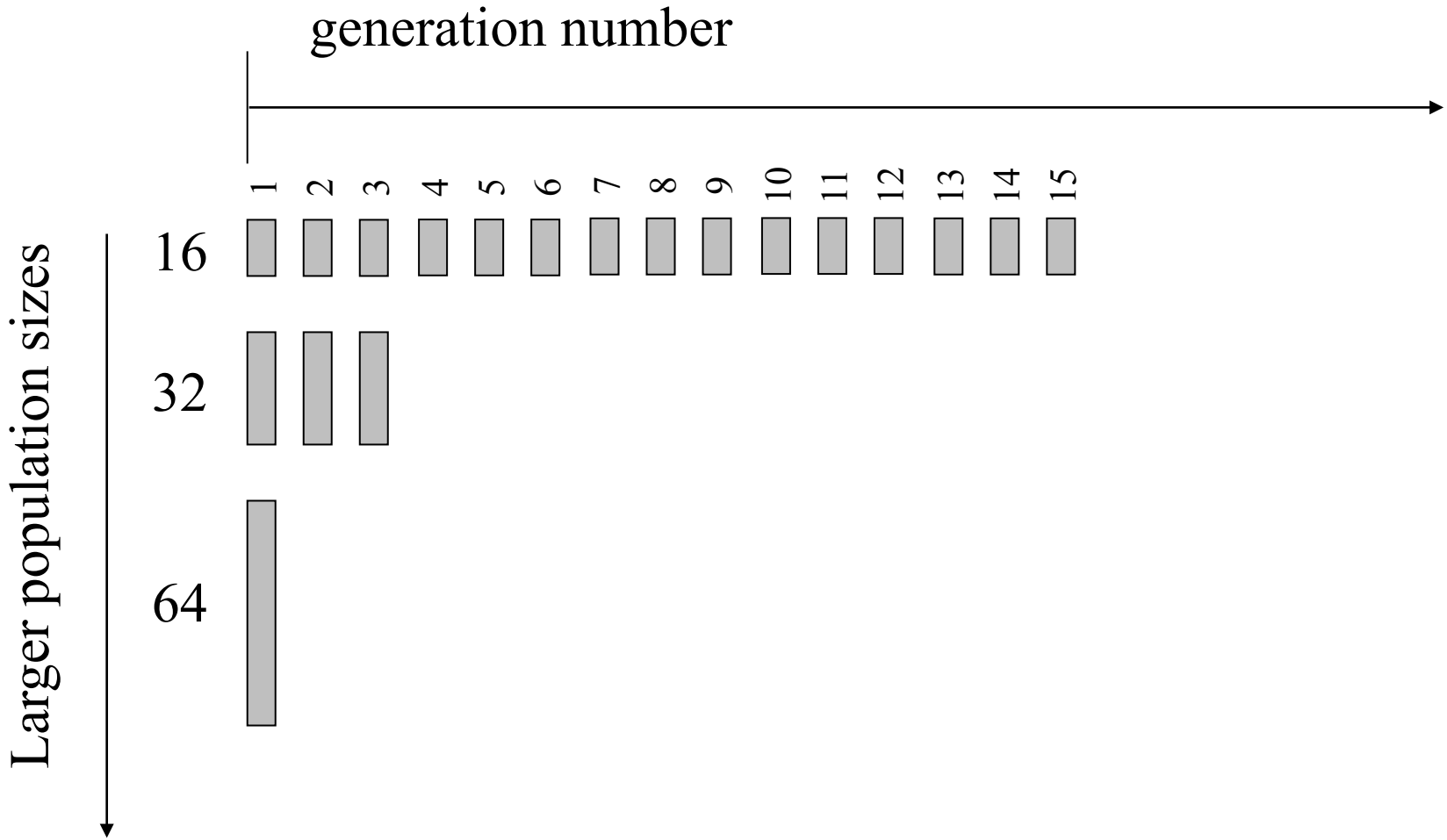

generation number
1
2
3
4
5
6
7
8
9
10
11
16
32
Larger population sizes

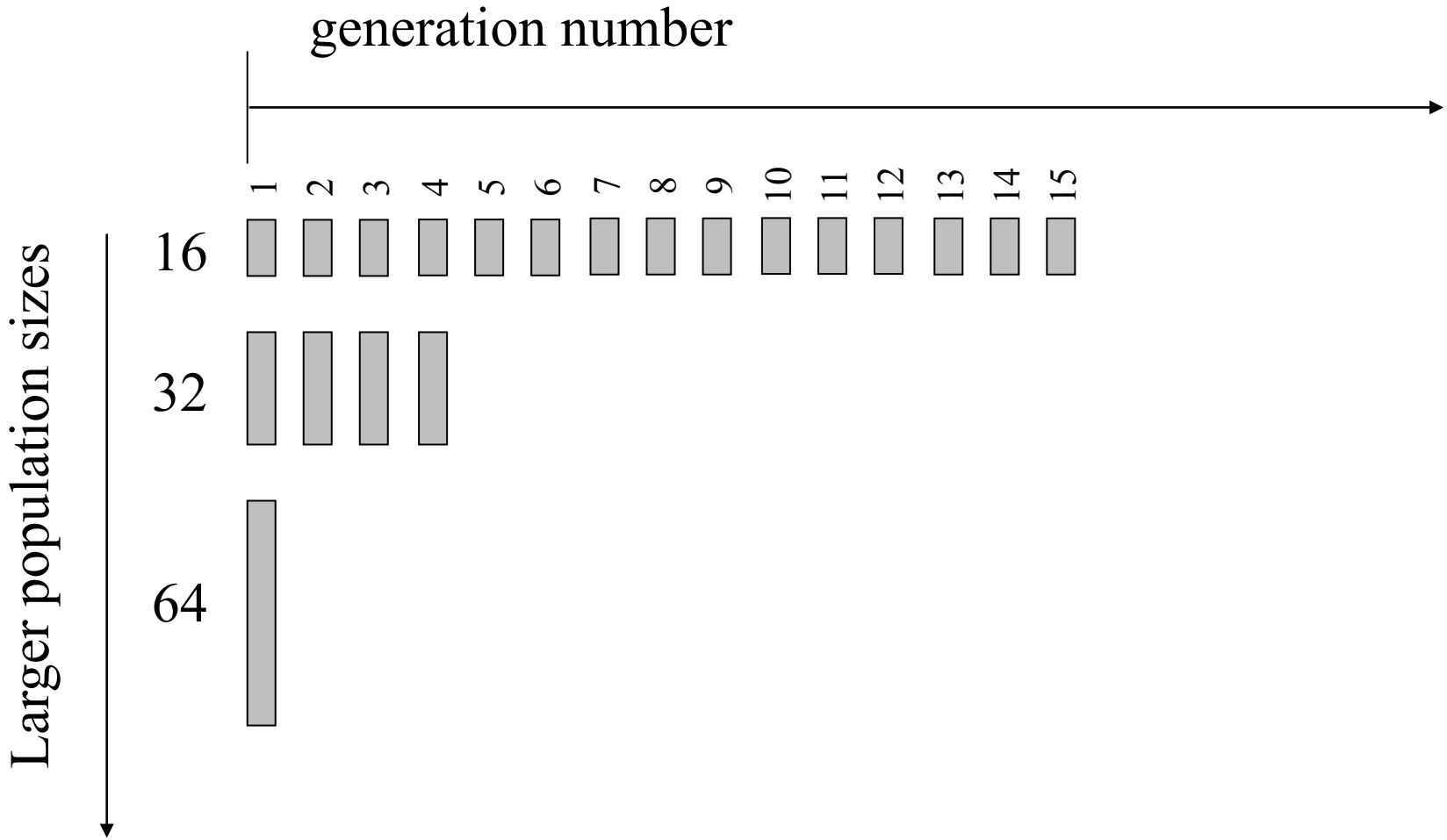
generation number
1
2
3
4
5
6
7
8
9
10
11
12
16
32
Larger population sizes

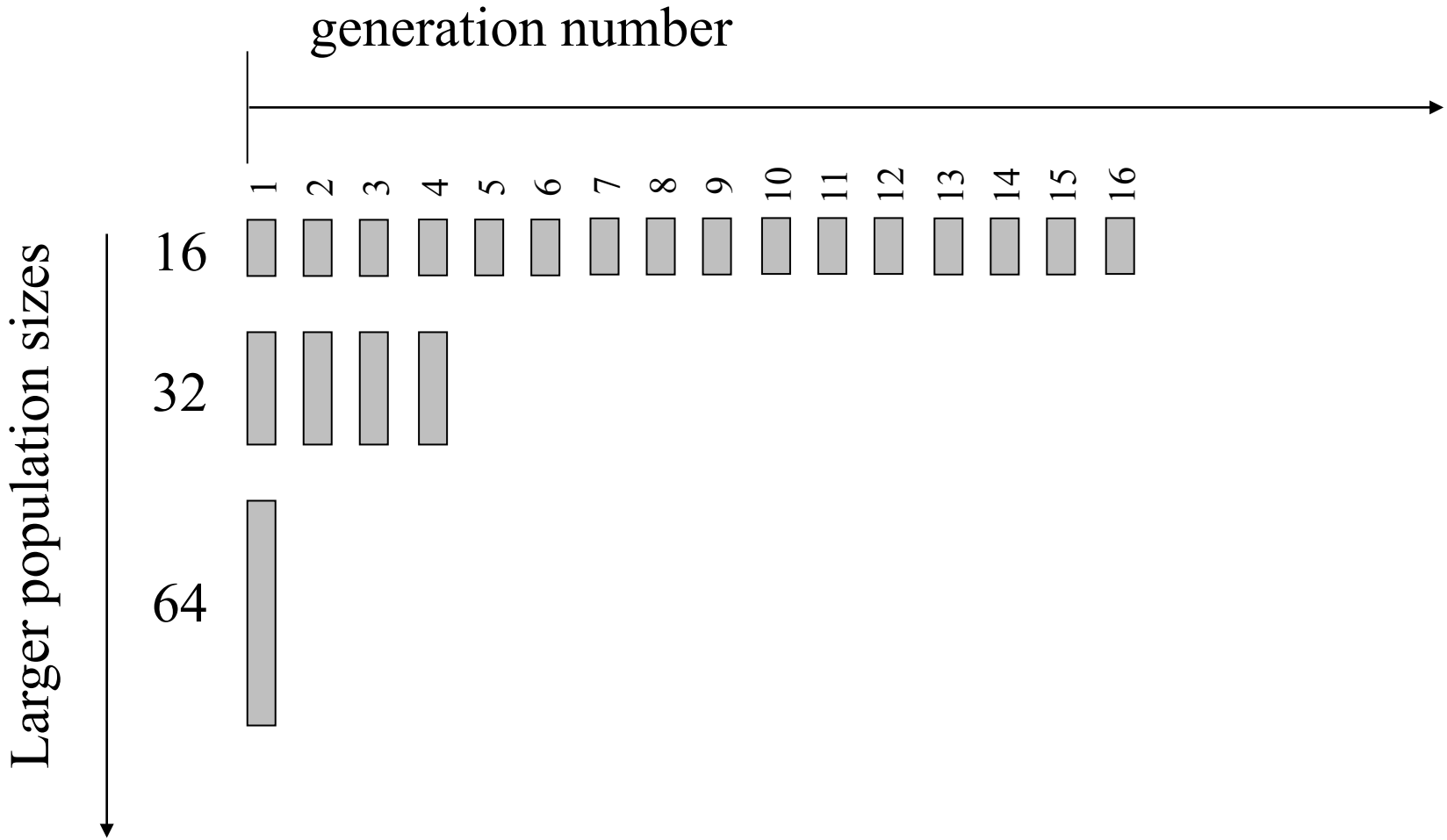

generation number
1
2
3
4
5
6
7
8
9
10
11
12
16
32
64
Larger population sizes


generation number
1
2
3
4
5
6
7
8
9
10
11
12
13
16
32
64
Larger population sizes


generation number
1
2
3
4
5
6
7
8
9
10
11
12
13
14
16
32
64
Larger population sizes


generation number
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
32
64
Larger population sizes


generation number
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
32
64
Larger population sizes


generation number
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
16
32
64
Larger population sizes

Implementation

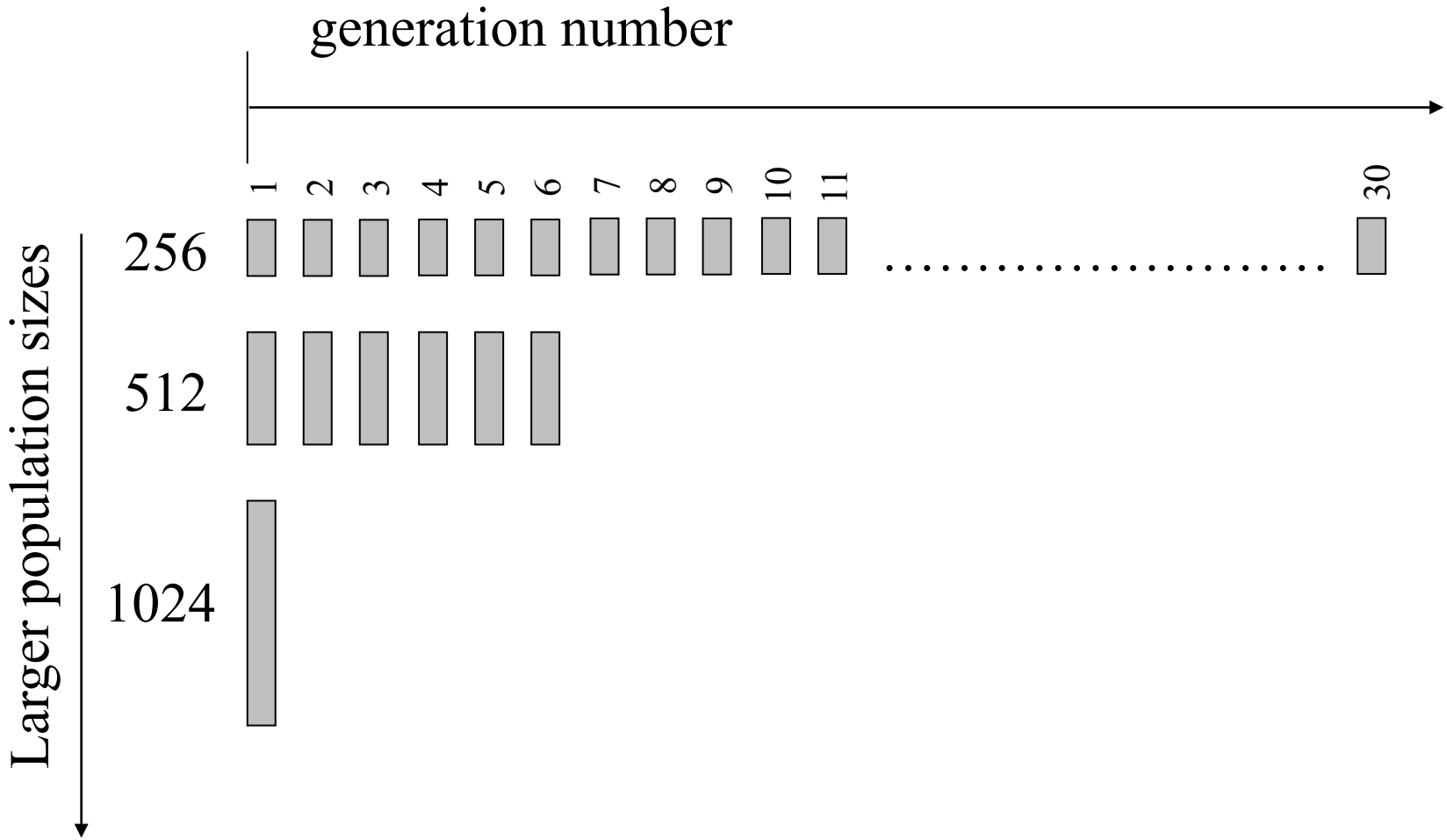
- Use a counter base 4
- Least significant digit changes 4 times more often than the next digit.

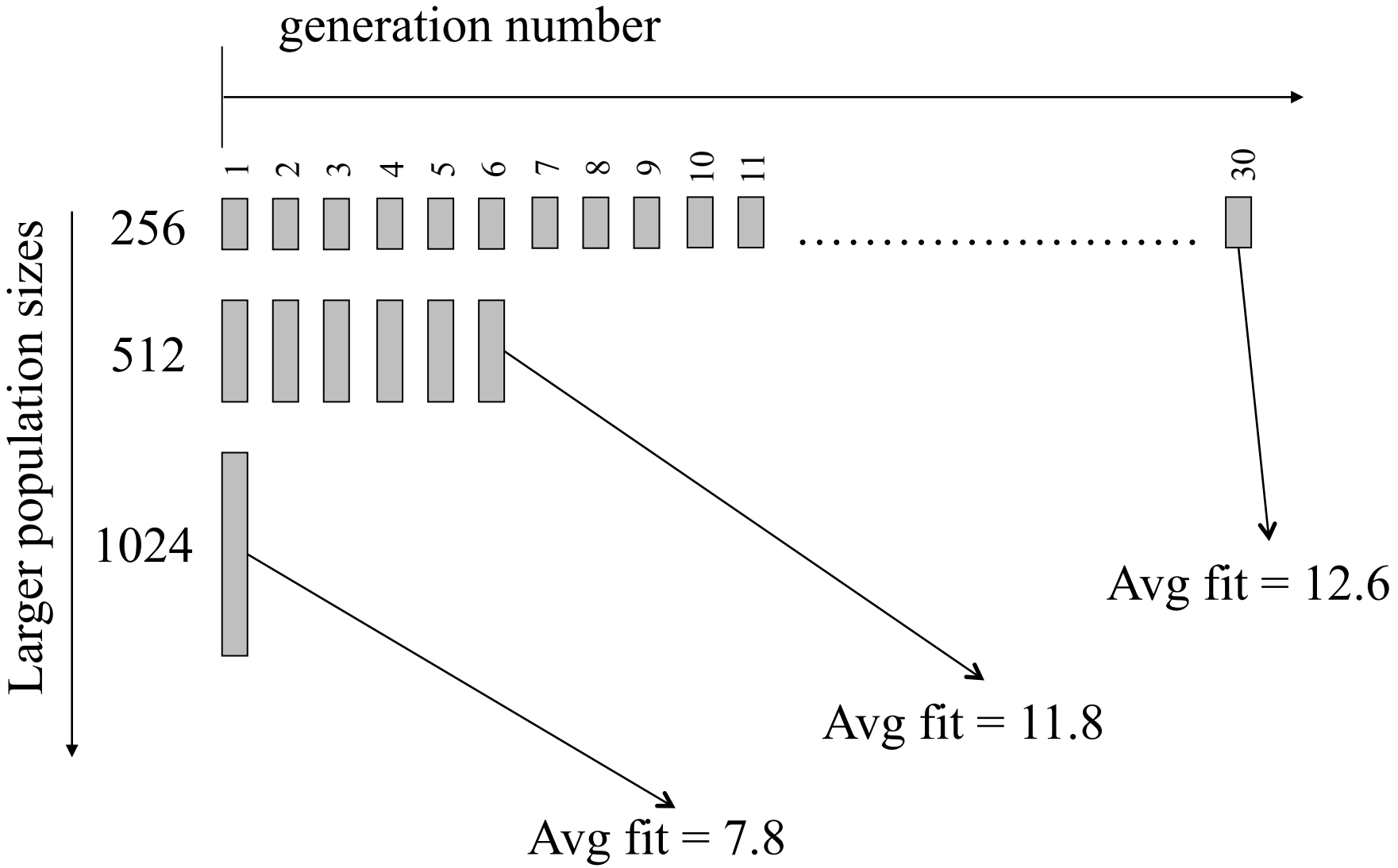
1
2
3
10
11
12
13
20
21
22
23
30
...

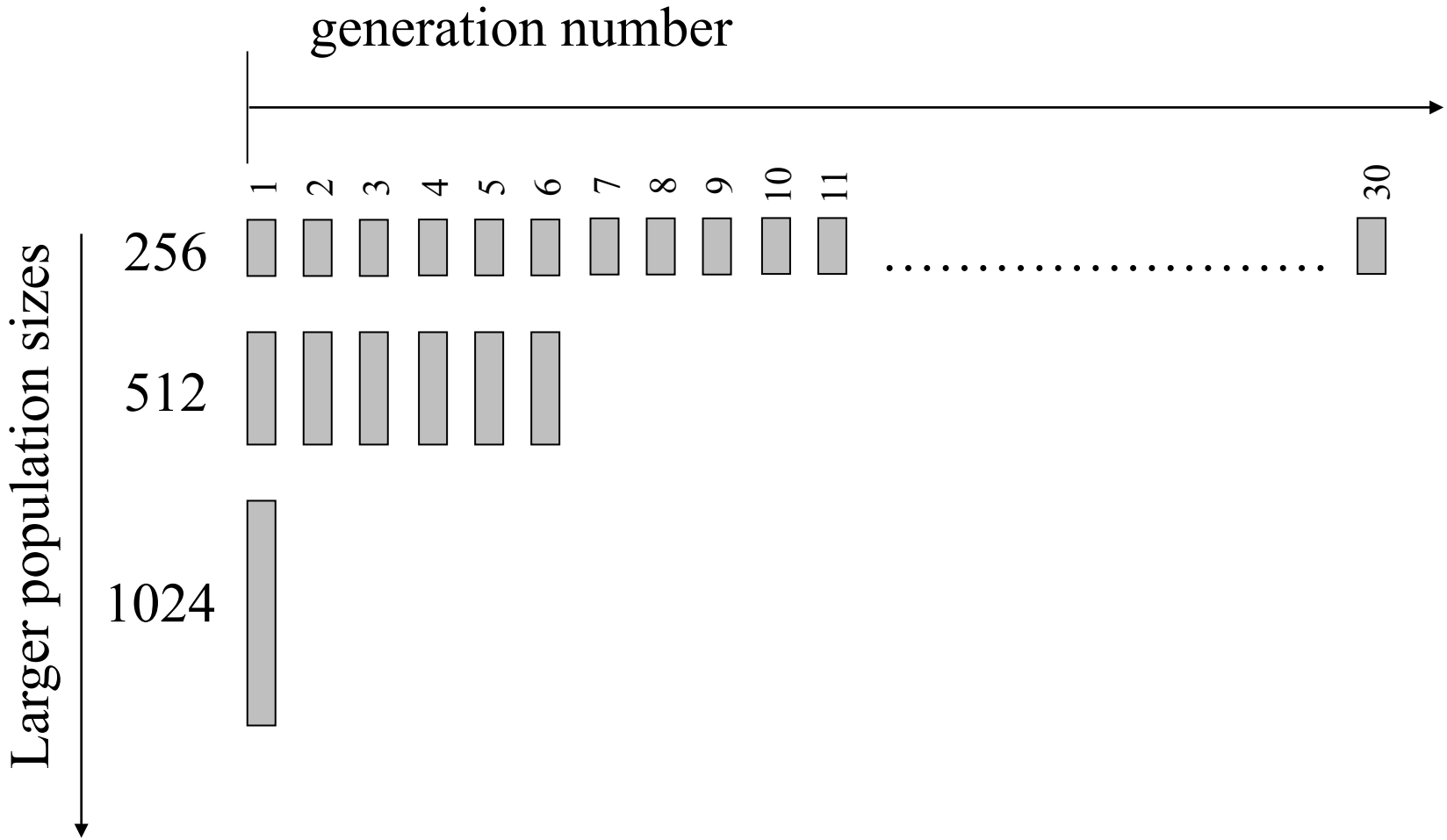
Do we keep running all populations forever?

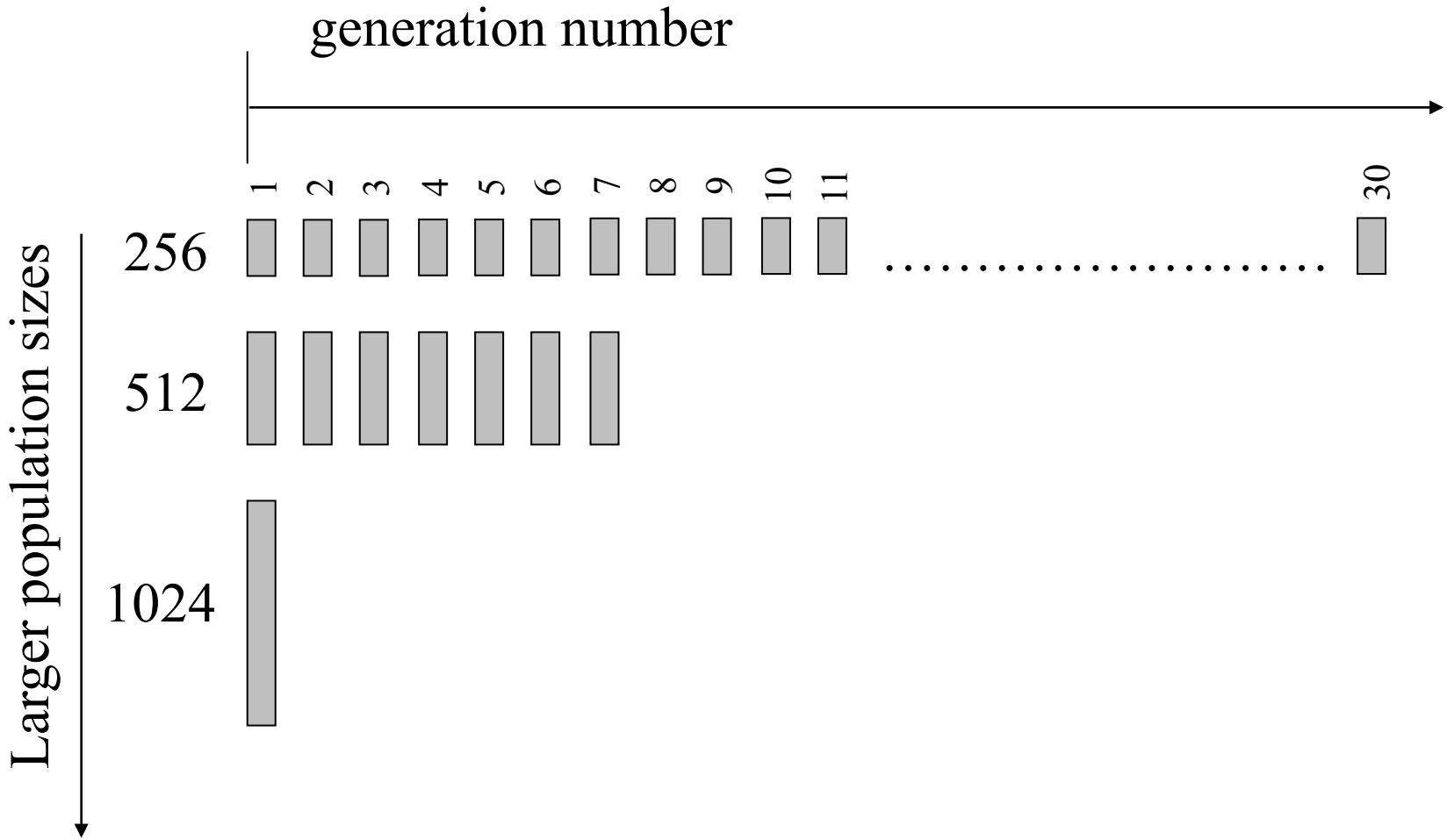
- Answer: No
- Sometimes populations are deleted
- We'll see how in a moment

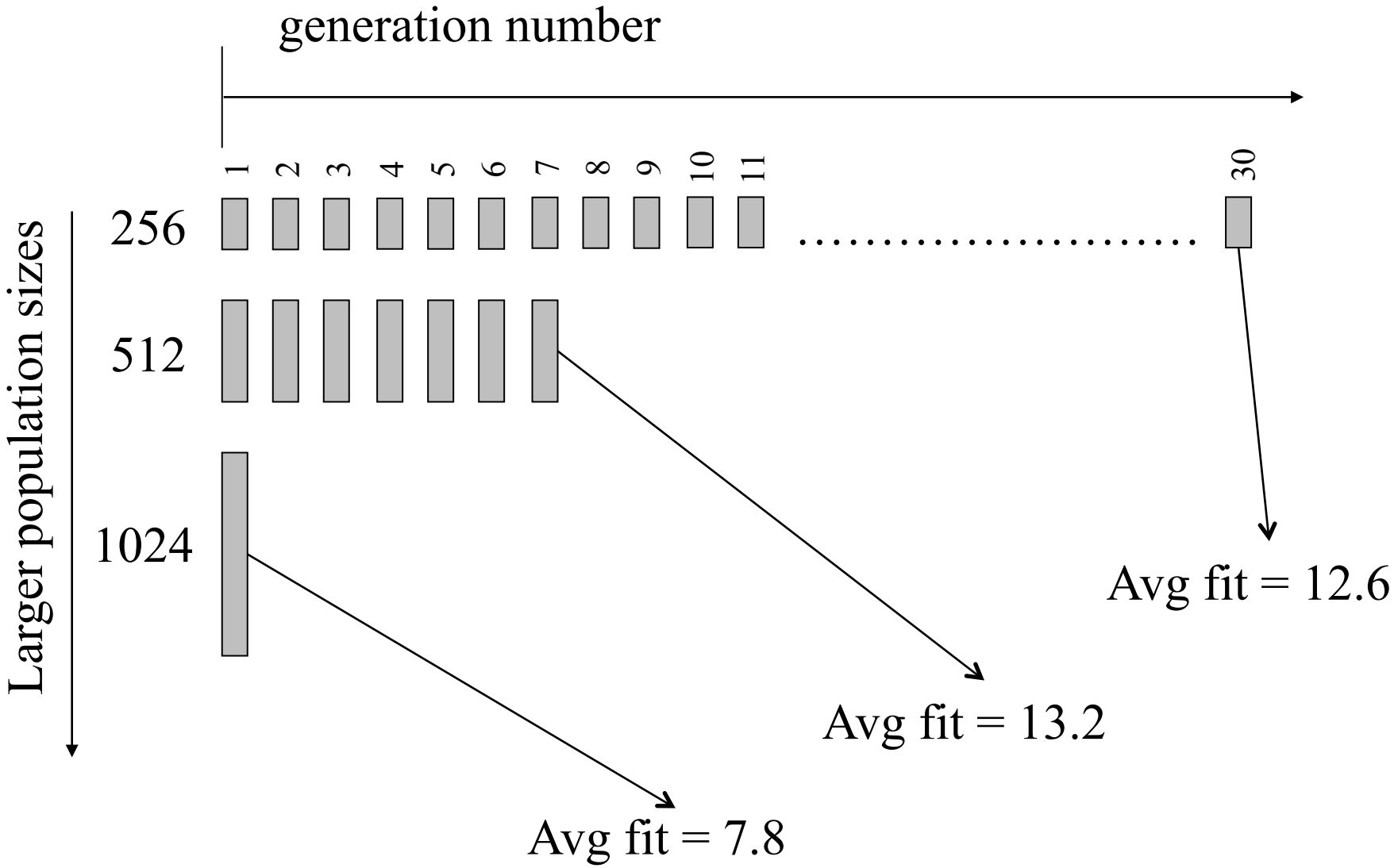
After some time…


generation number
1
2
3
4
5
6
7
8
9
10
11
30
256
512
1024
Larger population sizes

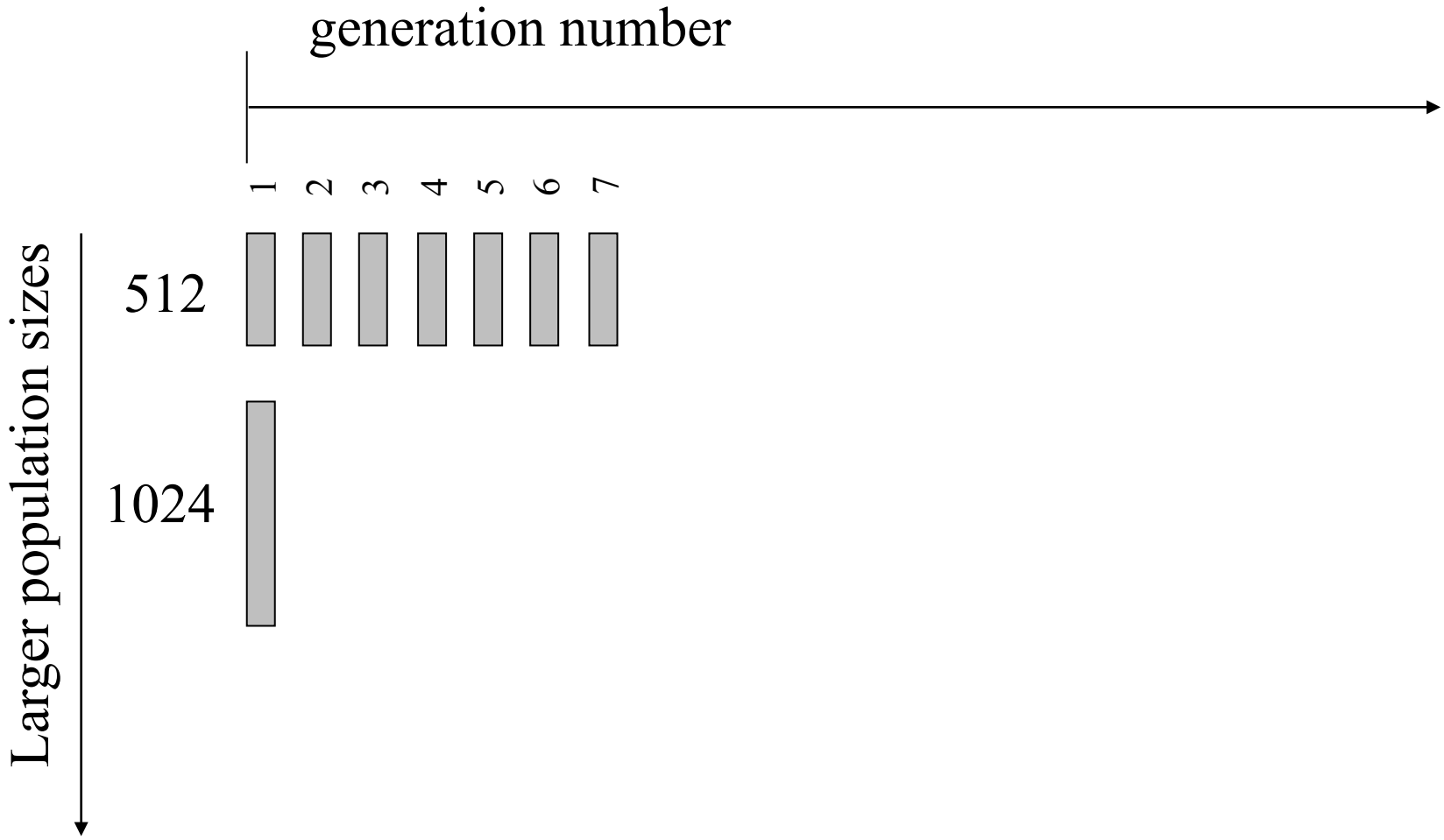

generation number
1
2
3
4
5
6
7
8
9
10
11
30
256
512
1024
Larger population sizes
Avg fit = 12.6
Avg fit = 11.8
Avg fit = 7.8


generation number
1
2
3
4
5
6
7
8
9
10
11
30
256
512
1024
Larger population sizes

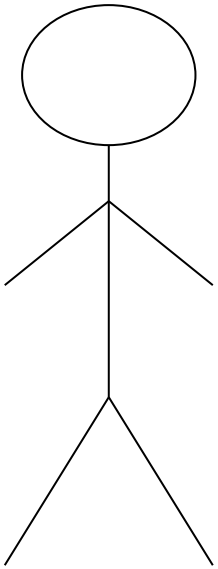

generation number
1
2
3
4
5
6
7
8
9
10
11
30
256
512
1024
Larger population sizes


generation number
1
2
3
4
5
6
7
8
9
10
11
30
256
512
1024
Larger population sizes
Avg fit = 12.6
Avg fit = 13.2
Avg fit = 7.8

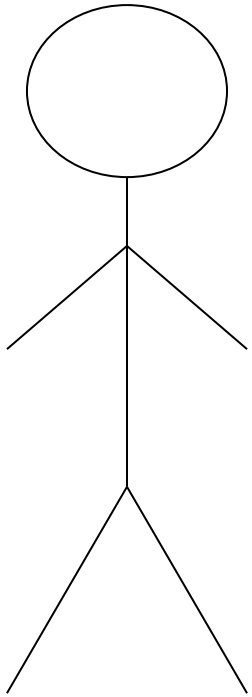
Delete population size with 256
and keep going…


generation number
1
2
3
4
5
6
7
Larger population sizes
512
1024

Why?



12 year old

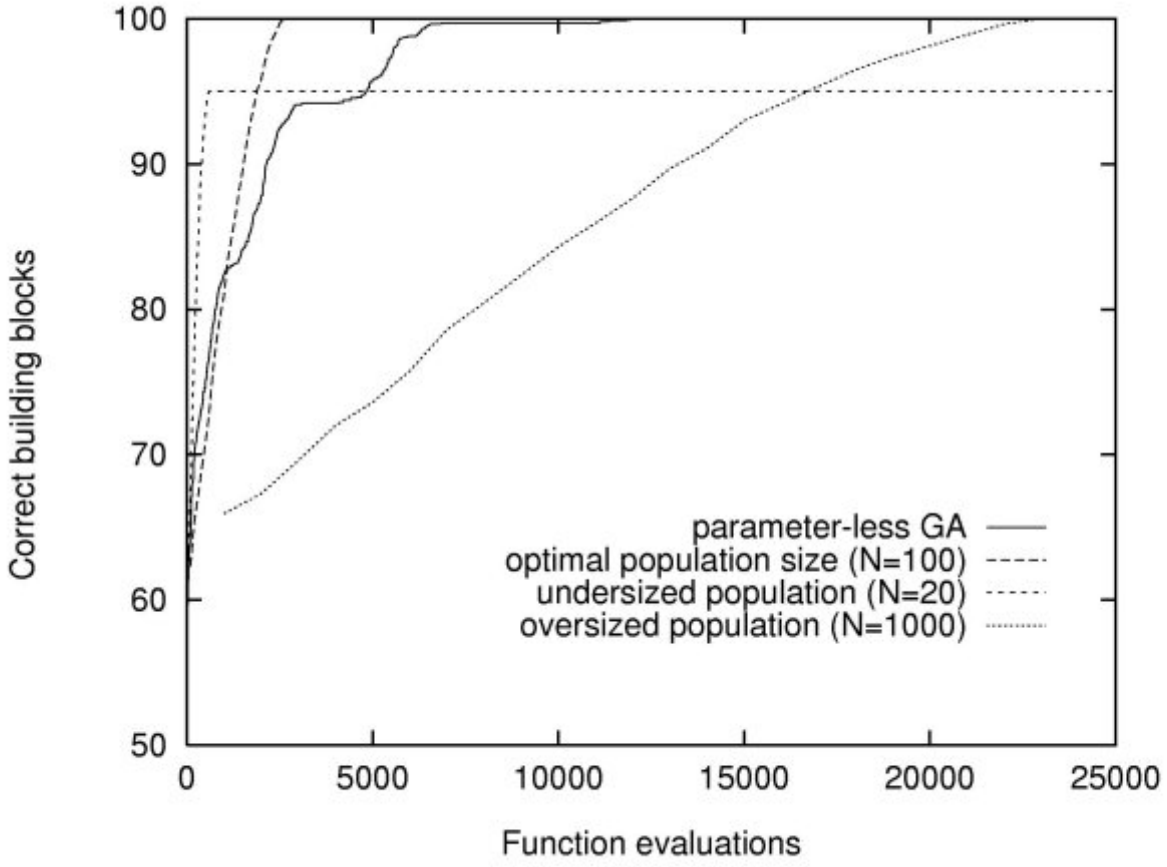


6 year old

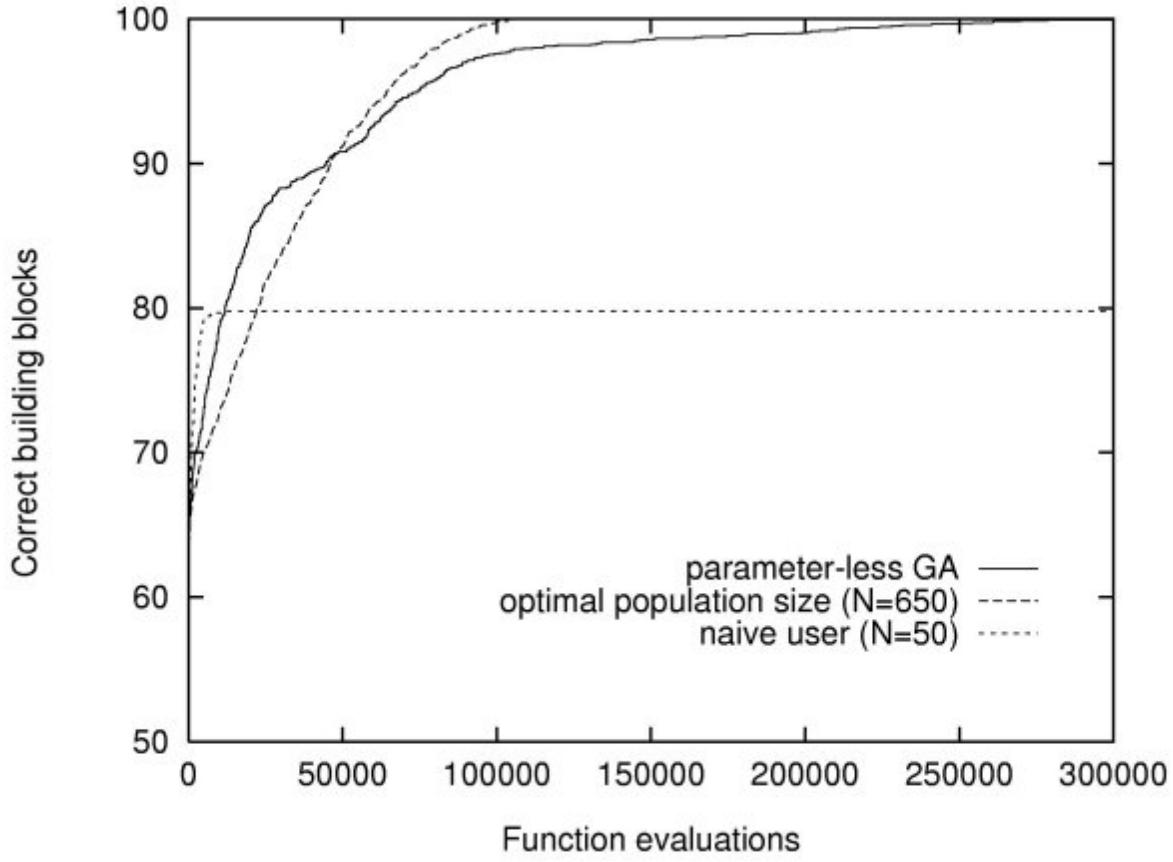
When to delete populations?

- At convergence (if no mutation is used) or
- when a population is overtaken fitness-wise by a larger population

Experiment #1: onemax

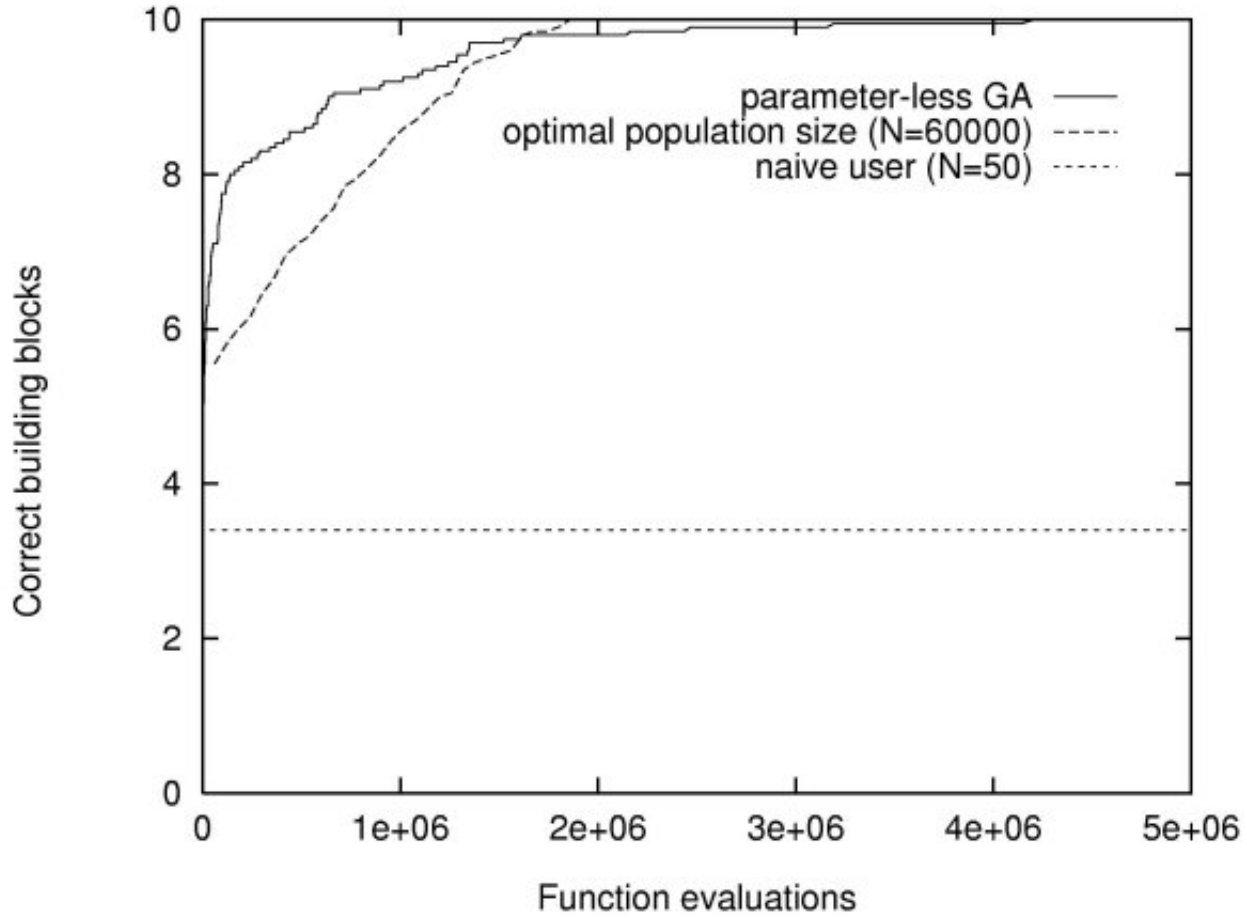


Experiment #2: noisy onemax



Experiment #3: trap functions

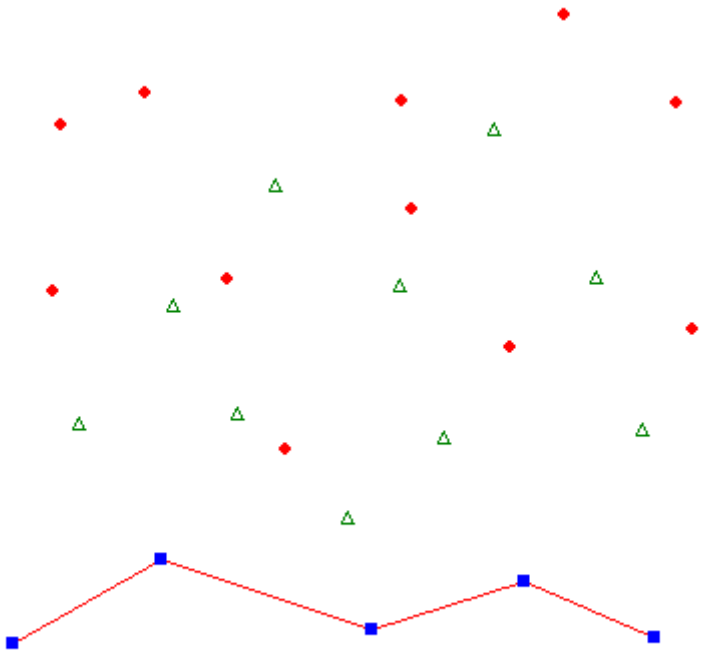
(with unform crossover)



Application to a network expansion problem

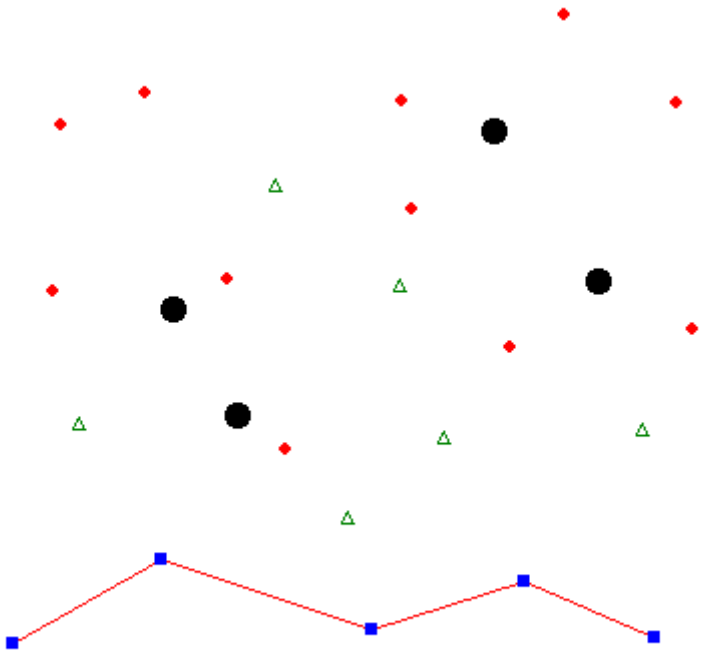
- Goal: illustrate the techniques in non-artificial problems
- Case study: a simplified version of a utility network expansion problem

Network expansion problem



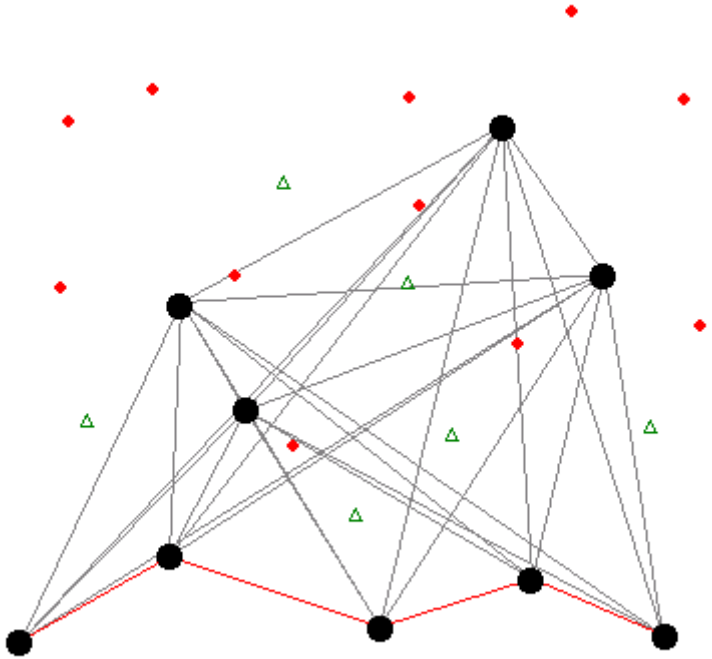
A 10-bit problem

The encoding

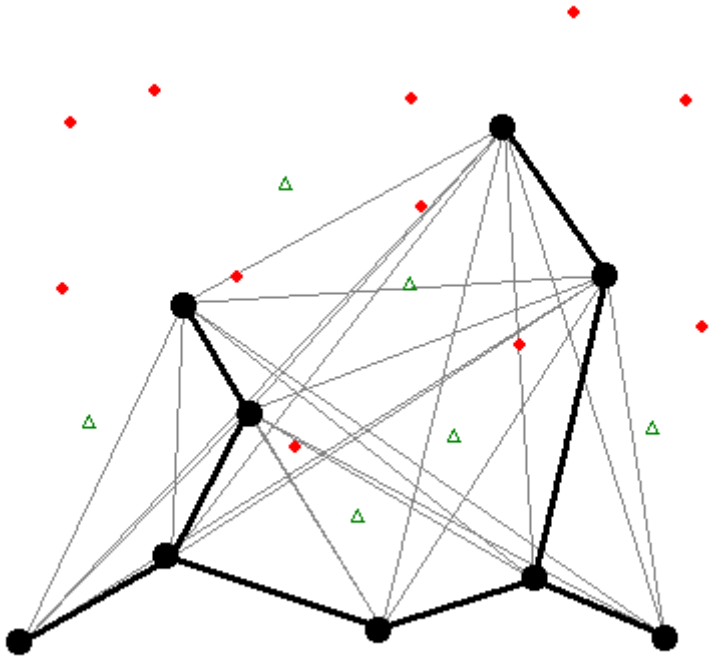


Solution: 0110000110

Obj. function in 3 steps: Step #1

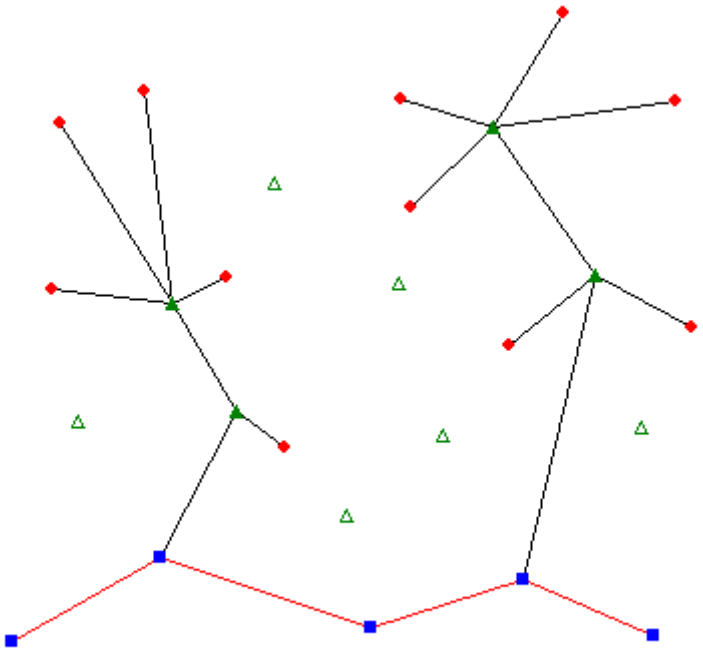


Step #2



Construct a minimum spanning tree

Step #3



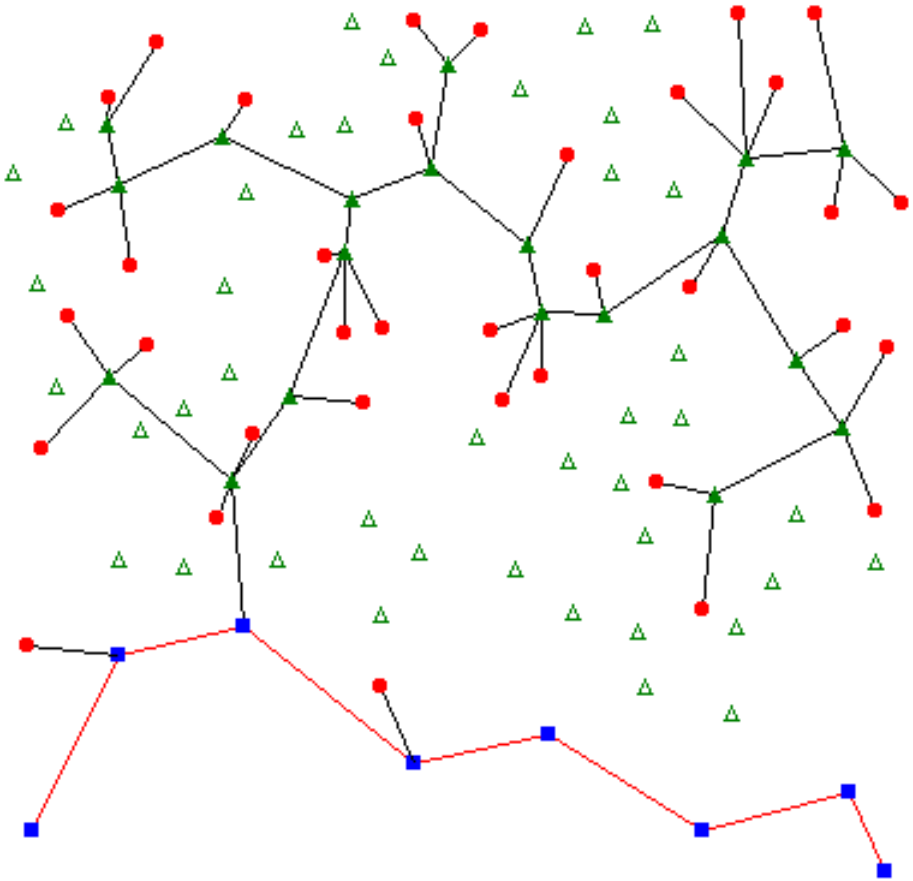
Network corresponding to solution 0110000110

60-bit network problem

Pop. size	Sol. quality
16	2131.79
32	2109.33
64	2004.93
128	1990.25
256	1971.50
512	1984.76
1024	1986.49
2048	1967.21
4096	1967.21
...	...
65536	1967.21

Takes on average 100-200 thousand f.e. to reach the target

Best solution found



60-bit network problem: “standard settings”

- population size = 100, p_c=0.7, p_m=1/l
- … couldn’t reach the target solution after a million function evaluations

60-bit network problem

- It's unlikely that a user would guess 2000 as the right population sizing for this problem
- Not even a GA expert would guess it right
- Use parameter-less GA

Another good thing about it

- Independent of the GA to be used with
- Can have parameter-less versions of other type of GAs

Conclusions

- It is possible to eliminate the parameters of the GA
- And keep a good performance across a broad range of problems

Want to know more?

- Read the papers:
  - *A parameter-less genetic algorithm* (1999), by Harik and Lobo.
  - *The parameter-less genetic algorithm in practice* (2003), by Lobo and Goldberg.