Last class

- Hill-climbing search
- Simulated annealing
- Local beam search
- Local search for continuous spaces
- Evolutionary algorithms

Local search

Heuristic Search and Evolutionary Algorithms

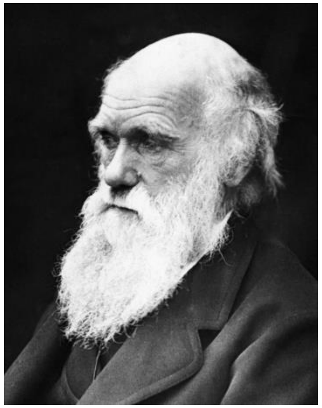
Lecture 5: Evolutionary Algorithms – Origins, Components and Applications

Chao Qian （钱超）

Associate Professor, Nanjing University, China

Email: qianc@nju.edu.cn
Homepage: http://www.lamda.nju.edu.cn/qianc/

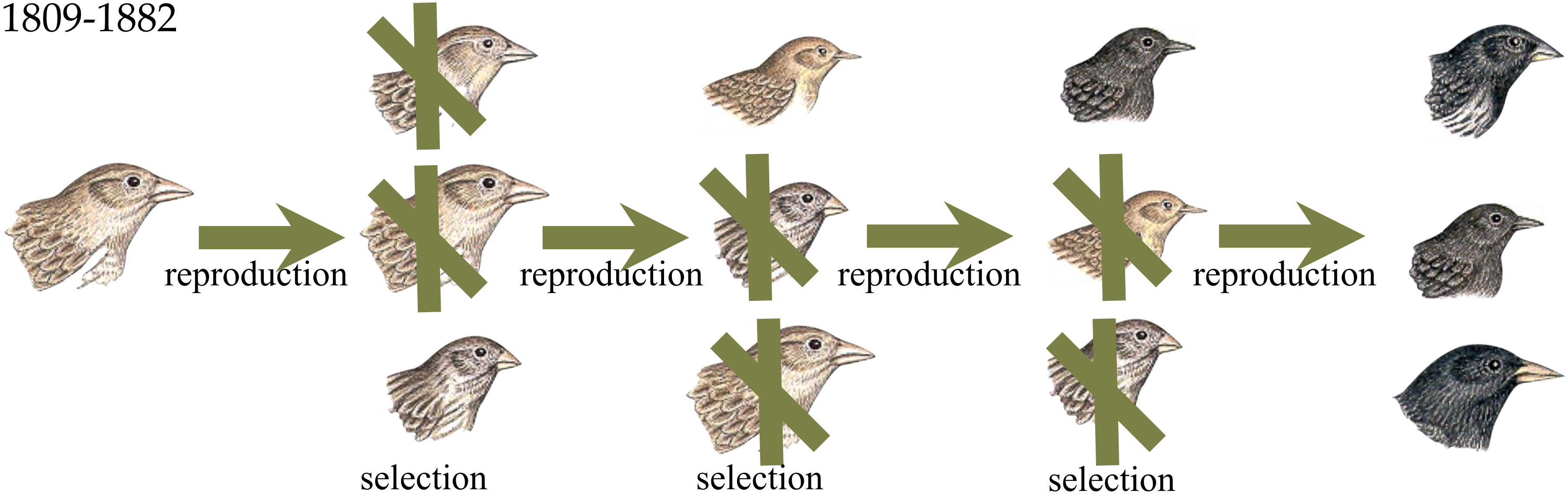
Biological evolution



Charles Darwin
1809-1882

C. Darwin, after collecting abundant evidence, developed a theory about how species evolve

reproduction with variation + nature selection



Optimization

With the development of computing technology

Curious researchers started to implement Darwin's theory of evolution in computer, and found connections to ***optimization***

Optimization:

how to put as much stuff as possible into a fixed size container?

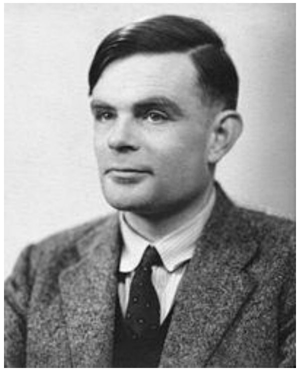
Formally: $\arg\max_{x \in \mathcal{X}} f(x)$ every x is an arrangement of objects

f counts the number of objects in the container

Evolutionary optimization

In 1950, Turing described how evolution might be used for his optimization:

building intelligent machine



Alan Turing
1912-1954

“We have thus divided our problem into two parts. The child programme and the education process. These two remain very closely connected. We cannot expect to find a good child machine at the first attempt. One must experiment with teaching one such machine and see how well it learns. One can then try another and see if it is better or worse. There is an obvious connection between this process and evolution, by the identifications

Structure of the child machine = Hereditary material

Changes of the child machine = Mutations

Judgment of the experimenter = Natural selection” (The last equation swapped)

[A. M. Turing. Computing machinery and intelligence. Mind 49: 433-460, 1950.]

The origins

J. H. Holland
1929-2015

Genetic Algorithms (GA)

for optimization in discrete domains

[J. H. Holland. *Outline for a logical theory of adaptive systems*. JACM, 1962]

University of Michigan

I. Rechenberg
1934-

Evolutionary Strategies (ES)

for optimization in continuous domains

[I. Rechenberg. *Cybernetic solution path of an experimental problem*. 1965]

Technical University of Berlin

L. J. Fogel
1928-2007

Evolutionary Programming (EP)

for optimizing finite state machines (agents)

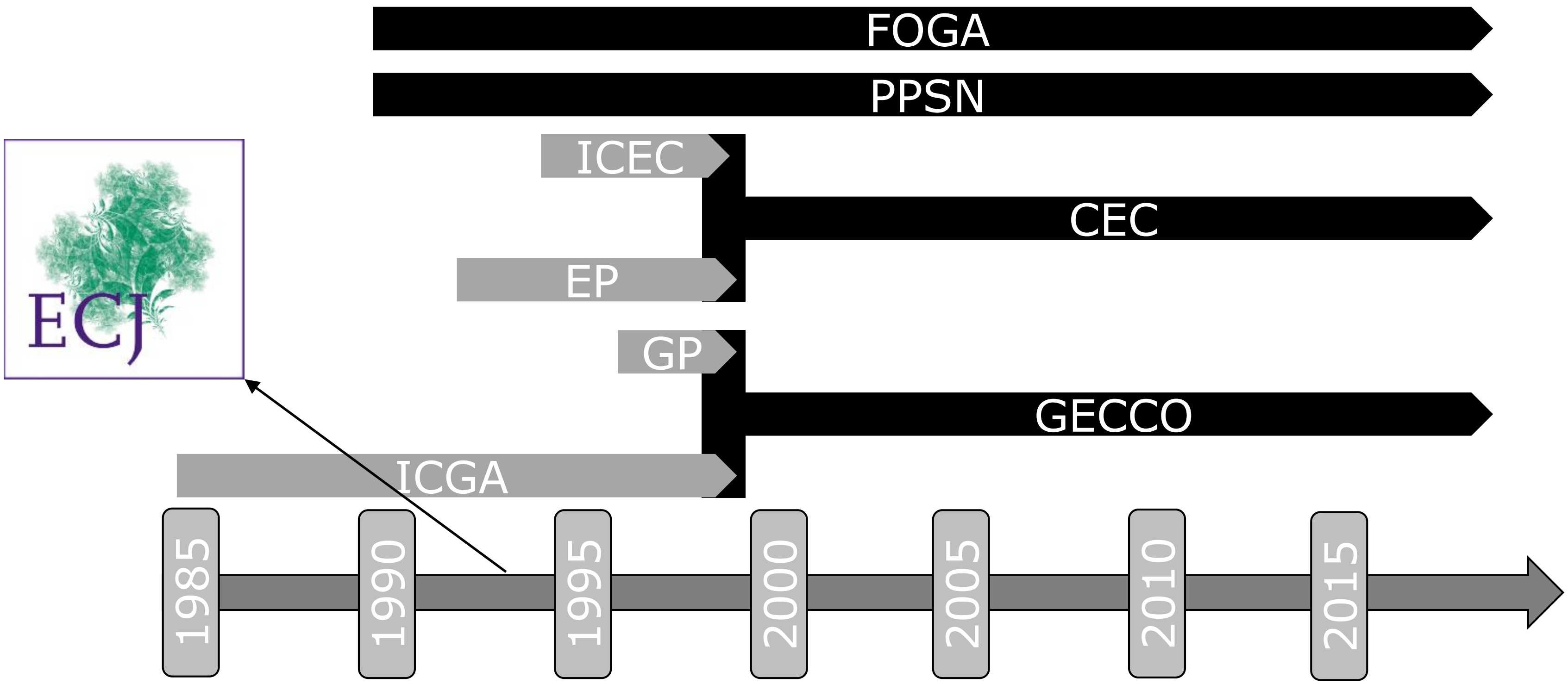
[L. J. Fogel, A. J. Owens, M. J. Walsh. *Artificial Intelligence through Simulated Evolution.* 1966]

University of California, Los Angeles

The origins

The research of GA, ES and EP was done independently from 1960s to 1980s, and unified to one field

"Evolutionary Computation" in 1990s



Main conferences and journals

Four main conferences

- IEEE Congress on Evolutionary Computation (CEC)
- ACM Conference on Genetic and Evolutionary Computation (GECCO)
- International Conference on Parallel Problem Solving from Nature (PPSN)
- ACM Conference on Foundations of Genetic Algorithms (FOGA)

Three main journals

- Evolutionary Computation Journal (ECJ, MIT Press, 1993)
- IEEE Trans. on Evolutionary Computation (TEvC)
- ACM Trans. on Evolutionary Learning and Optimization (TELO)

Evolutionary algorithms

Genetic Algorithms (GA)
for optimization in discrete domains

[J. H. Holland. *Outline for a logical theory of adaptive systems*. JACM, 1962]

Evolutionary Strategies (ES)
for optimization in continuous domains

[I. Rechenberg. *Cybernetic solution path of an experimental problem*. 1965]

Evolutionary Programming (EP)
for optimizing finite state machines

[L. J. Fogel, A. J. Owens, M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. 1966]

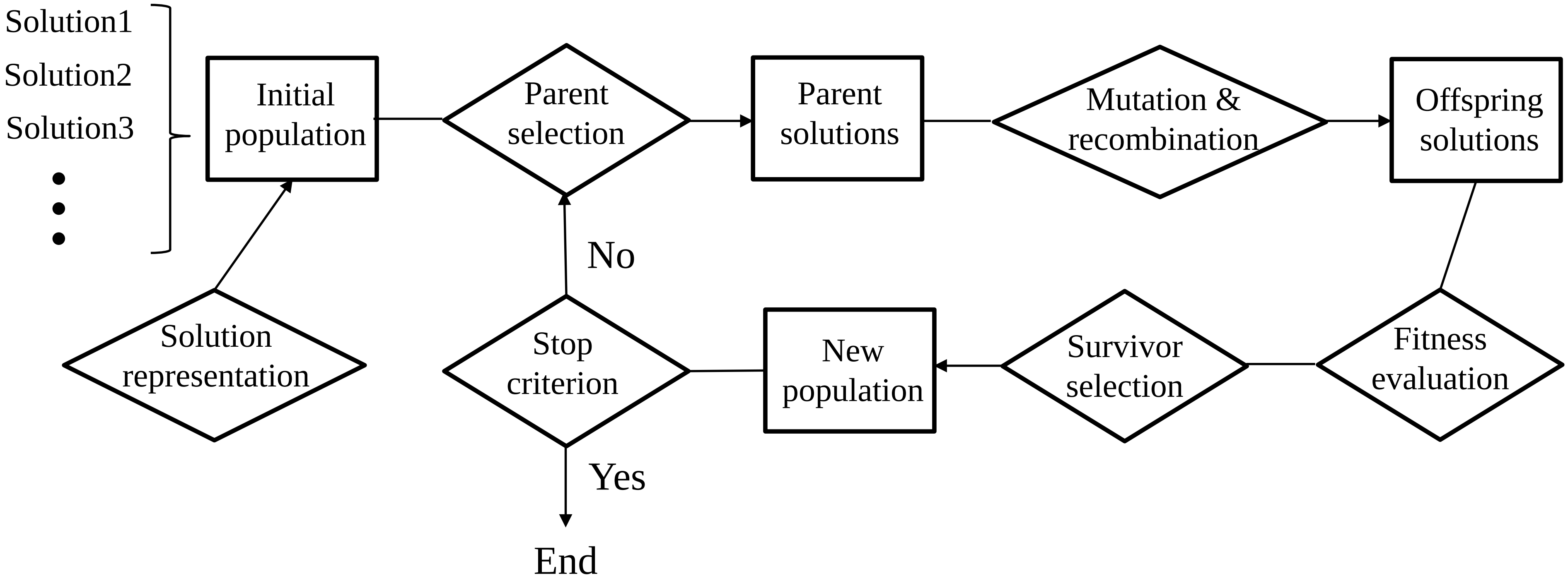
Other variants:
Genetic Programming
Differential Evolution
...

Other heuristics inspired from nature:
Ant Colony Optimization
Particle Swarm Optimization
...

Evolutionary algorithms (EAs)

Evolutionary algorithms

EAs share a common routine for $\arg\max_{x} f(x)$



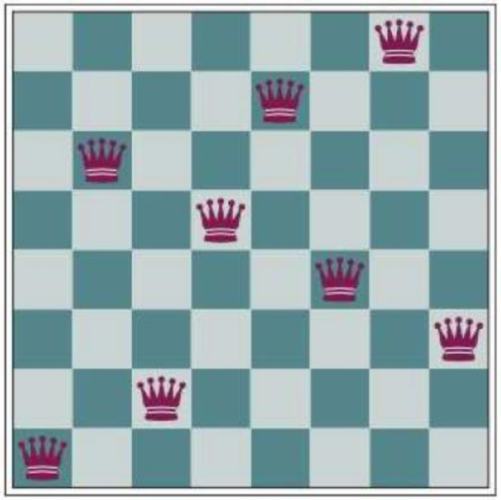
Components - representation

Representation: provides code for candidate solutions that can be manipulated by a computer

encoding & decoding

phenotype:
object in original problem context

genotype:
code to denote that object



Integer vector

1	6	2	5	7	4	8	3

Binary vector

000101001100110011111010

Permutation

1	6	2	5	7	4	8	3

different

Components – fitness

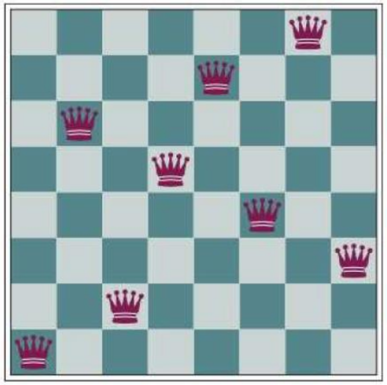
Fitness: represents the task to solve, or the requirements (can be seen as "the environment") to adapt to

Fitness evaluation assigns a single real-value to each phenotype which forms the basis for selection

Example:

$$\arg\max_x x^2$$

Fitness: x^2



Fitness: number of nonattacking pairs of queens

Components - population

Population: holds the candidate solutions of the problem, which is a multiset of genotypes

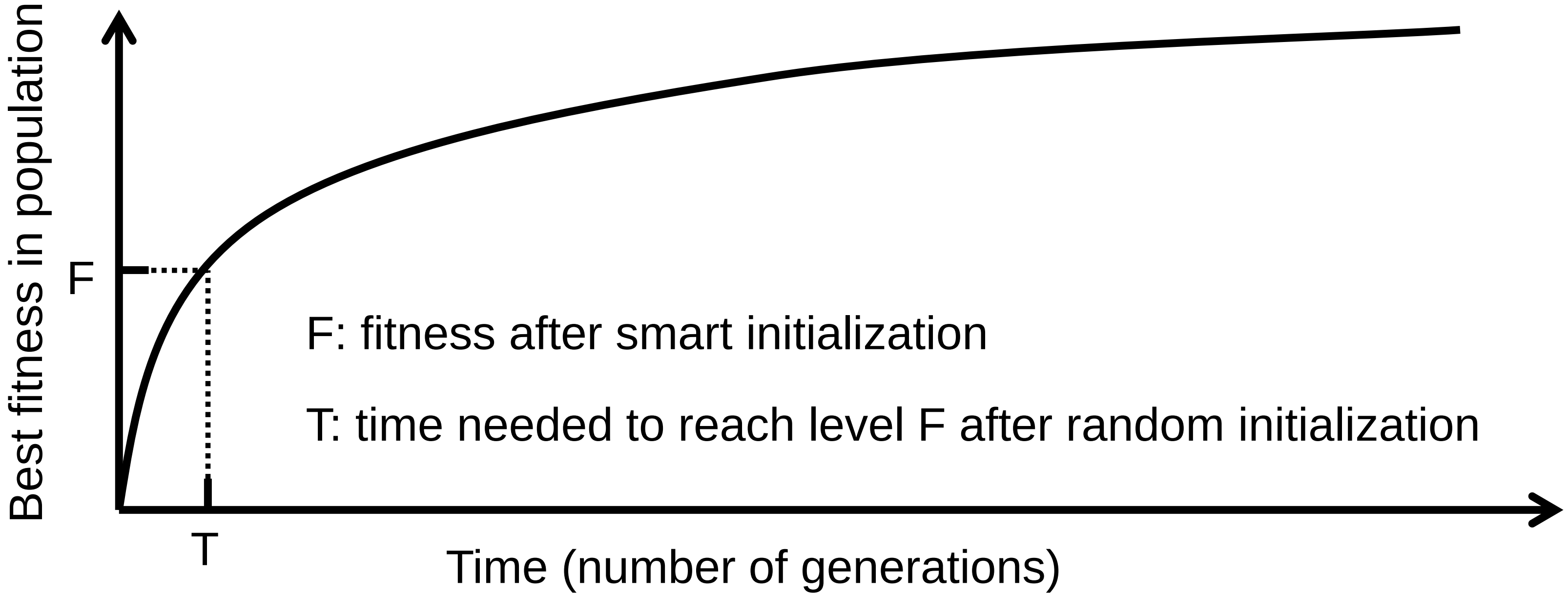
Size of population: the number of contained genotypes

Diversity of population: the number of different fitnesses / phenotypes / genotypes present

Components - initialization

Initialization: generates the genotypes in the initial population

- generates the genotypes randomly
- includes existing solutions, or uses problem-specific heuristics, to seed the population



Components – parent selection

Parent selection: selects genotypes to undergo variation

Usually probabilistic

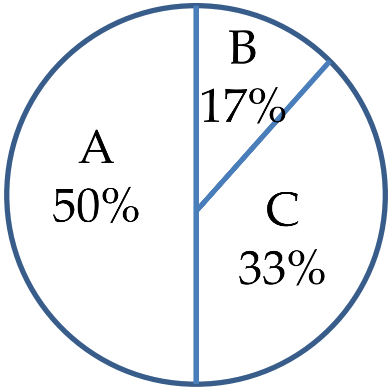
- high quality genotypes more likely to be selected than low quality
- even worst in current population usually has non-zero probability of being selected

Example: fitness proportional selection

fitness(A) = 3

fitness(B) = 1

fitness(C) = 2



Components – variation

Variation: generates new (offspring) genotypes

- Mutation: causes small, random variance of one parent

Parent	1	0	1	1	1	0	0	0
Offspring	1	0	1	1	0	0	0	0

- Recombination/crossover: merges information from parents into offspring

Parent1	1	0	1	1	1	0	0	0
Parent2	0	0	1	0	1	0	1	0
Offspring1	1	0	1	1	1	0	1	0
Offspring2	0	0	1	0	1	0	0	0

Components – survivor selection

Survivor selection: selects genotypes from parents and offspring to form the next population

Often deterministic

- Fitness based : e.g., rank parents and offspring, and select the top segment
- Age based: make as many offspring as parents and delete all parents

Example:

Parents	Offspring	
fitness(A) = 3	fitness(D) = 4	Fitness based: A, C, D
fitness(B) = 1	fitness(E) = 1.5	Age based: D, E, F
fitness(C) = 2	fitness(F) = 1	

Components – stop criterion

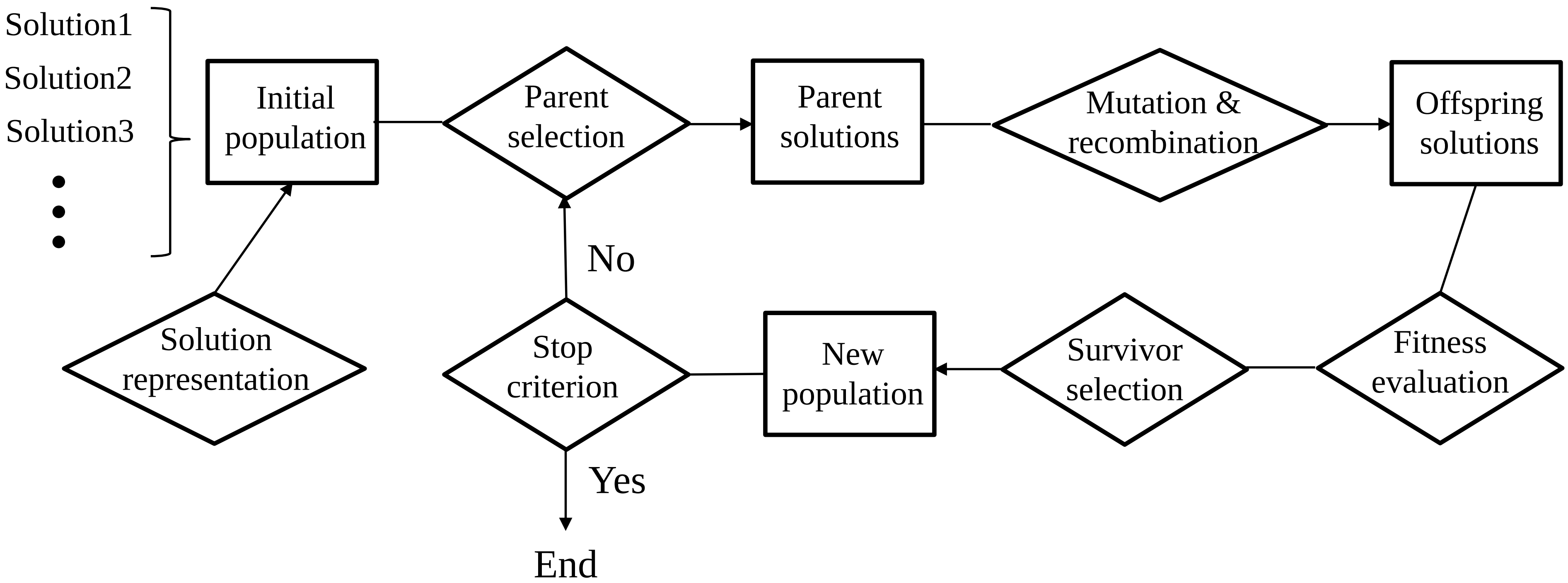
Anytime behavior of EAs

Stop criteria:

- Reaching some (known/hoped for) fitness
- Reaching some maximum allowed number of generations
- Reaching some specified number of generations without fitness improvement
- Reaching some minimum level of population diversity

Evolutionary algorithms

EAs share a common routine for $\arg\max_x f(x)$



Need to design each component of EAs

Evolutionary algorithms

Genetic Algorithms (GA)
for optimization in discrete domains

[J. H. Holland. *Outline for a logical theory of adaptive systems*. JACM, 1962]

Binary representation

Evolutionary Strategies (ES)
for optimization in continuous domains

[I. Rechenberg. *Cybernetic solution path of an experimental problem*. 1965]

Evolutionary Programming (EP)
for optimizing finite state machines

[L. J. Fogel, A. J. Owens, M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. 1966]

Real-valued representation

Genetic Programming (GP)
for optimizing computer programs

[J. R. Koza. *Genetic Programming*.1992]

Tree representation

Example illustration - max x^2

The problem: $\arg max_{x\in\{0,1,\ldots,31\}}\ x^2$ Fitness function f: x^2

Solution representation: binary vector of length 5

For example, $x = 15$ can be represented by 01111

Genotype no.	Initial population	x value	Fitness $f(x) = x^2$	Selection prob. p_i	Expected count	Actual count
1	0 1 1 0 1	13	169	0.14	0.58	1
2	1 1 0 0 0	24	576	0.49	1.97	2
3	0 1 0 0 0	8	64	0.06	0.22	0
4	1 0 0 1 1	19	361	0.31	1.23	1

Population size = 4, randomly generated

Parent selection: $p_i = f(i)/\sum_{j\in P} f(j)$

Parent solutions

Example illustration - max x^2

Genotype no.	Parent solutions	Crossover point	Offspring after xover	Flipped bits	Offspring after mutation
1	0 1 1 0|1	4	0 1 1 0 0	1	1 1 1 0 0
2	1 1 0 0|0	4	1 1 0 0 1	none	1 1 0 0 1
2	1 1|0 0 0	2	1 1 0 1 1	none	1 1 0 1 1
4	1 0|0 1 1	2	1 0 0 0 0	3	1 0 1 0 0

One-point crossover:

Select one point randomly, and exchange the parts after the point

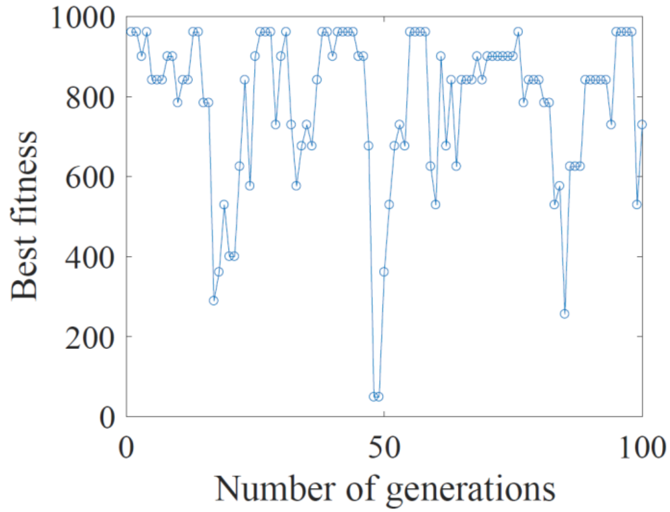
Bit-wise mutation:

Flip each bit of a solution with prob. $1/n$ where $n = 5$

Example illustration - max x^2

Initial population	x value	Fitness $f(x) = x^2$	Offspring after mutation	x value	Fitness $f(x) = x^2$	Next population
0 1 1 0 1	13	169	1 1 1 0 0	26	676	1 1 1 0 0
1 1 0 0 0	24	576	1 1 0 0 1	25	625	1 1 0 0 1
0 1 0 0 0	8	64	1 1 0 1 1	27	729	1 1 0 1 1
1 0 0 1 1	19	361	1 0 1 0 0	18	324	1 0 1 0 0

Curve change of the best fitness



Fitness evaluation

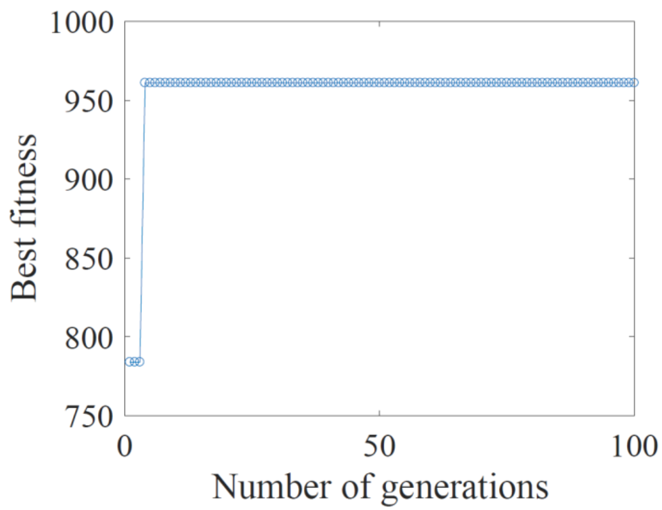
Age based survival selection:

Use the offspring directly to form the next population

Example illustration - max x^2

Initial population	x value	Fitness $f(x) = x^2$	Offspring after mutation	x value	Fitness $f(x) = x^2$	Next population
0 1 1 0 1	13	169	1 1 1 0 0	26	676	1 1 1 0 0
1 1 0 0 0	24	576	1 1 0 0 1	25	625	1 1 0 0 1
0 1 0 0 0	8	64	1 1 0 1 1	27	729	1 1 0 1 1
1 0 0 1 1	19	361	1 0 1 0 0	18	324	1 1 0 0 0

Curve change of the best fitness



Fitness based survival selection:

Select the best four genotypes from the current population and offspring

Example illustration - knapsack

Knapsack problem: given n items, each with a weight w_i and a value v_i, to select a subset of items maximizing the sum of values while keeping the summed weights within some capacity W_{max}

$$\arg max_{x\in\{0,1\}^n} \sum_{i=1}^{n} v_i x_i \quad s.t. \sum_{i=1}^{n} w_i x_i \leq W_{max}$$

$x_i = 1$: the i-th item is included

Solution representation

Genotype: binary vector of length n

1	1	0	1	1	0	1	1

Phenotype: binary vector of length n

1	1	0	1	1	0	1	1

Decoding: scan from left to right, and keep the value 1 if the summed weight does not exceed W_{max}

Example illustration - knapsack

Knapsack: $\arg max_{x \in \{0,1\}^n} \sum_{i=1}^{n} v_i x_i \quad s.t. \sum_{i=1}^{n} w_i x_i \leq W_{max}$

Solution representation

$x_i = 1$: the i-th item is included

Genotype: binary vector of length n

1	1	0	1	1	0	1	1

Phenotype: binary vector of length n

1	1	0	1	1	0	1	1

Decoding: scan from left to right, and keep the value 1 if the summed weight does not exceed W_{max}

Example: v_i: 4,2,6,10,4,3,7,2; w_i: 2,3,3,8,6,5,7,1; $W_{max} = 25$

Genotype: 11011011 Phenotype: 11011001

Fitness function f: the sum of values, i.e., $\sum_{i=1}^{n} v_i x_i$

Example illustration - knapsack

Population size	500
Initialization	Random
Parent selection	Tournament selection with size 2
Recombination	One-point crossover
Recombination prob.	70%
Mutation	Bit-wise mutation
Mutation prob.	$1/n$
Number of offspring	500
Survival selection	Age based
Termination condition	No improvement in last 25 generations

Select two solutions from the population uniformly at random, and choose the better one as a parent solution

Example illustration - knapsack

Population size	500
Initialization	Random
Parent selection	Tournament selection with size 2
Recombination	One-point crossover
Recombination prob.	70%
Mutation	Bit-wise mutation
Mutation prob.	$1/n$
Number of offspring	500
Survival selection	Age based
Termination condition	No improvement in last 25 generations

Select one point randomly, and exchange the parts of the parents after that point

Example illustration - knapsack

Population size	500
Initialization	Random
Parent selection	Tournament selection with size 2
Recombination	One-point crossover
Recombination prob.	70%
Mutation	Bit-wise mutation
Mutation prob.	$1/n$
Number of offspring	500
Survival selection	Age based
Termination condition	No improvement in last 25 generations

Flip each bit of a solution with probability $1/n$

Example illustration - knapsack

Population size	500
Initialization	Random
Parent selection	Tournament selection with size 2
Recombination	One-point crossover
Recombination prob.	70%
Mutation	Bit-wise mutation
Mutation prob.	$1/n$
Number of offspring	500
Survival selection	Age based
Termination condition	No improvement in last 25 generations

The 500 offspring form the next population directly

Example illustration - knapsack

Population size	500
Initialization	Random
Parent selection	Tournament selection with size 2
Recombination	One-point crossover
Recombination prob.	70%
Mutation	Bit-wise mutation
Mutation prob.	$1/n$
Number of offspring	500
Survival selection	Age based
Termination condition	No improvement in last 25 generations

Example illustration - knapsack

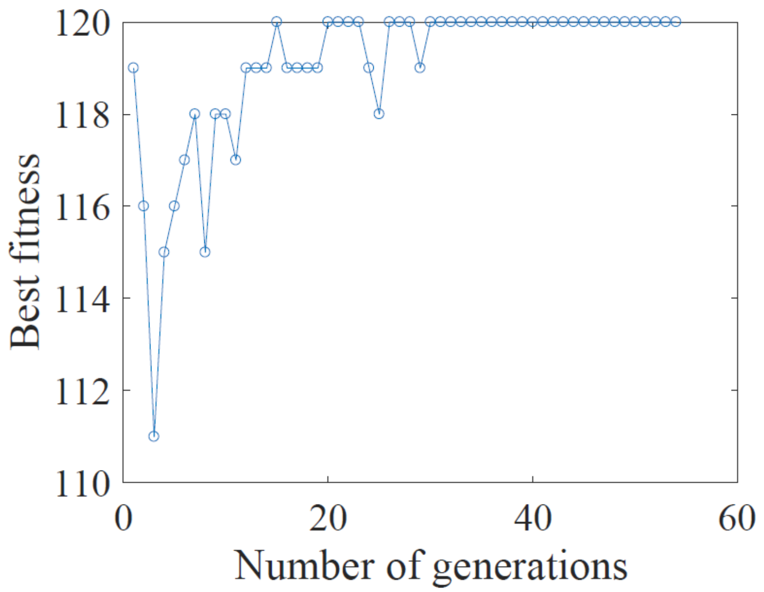
Example: $n = 20, W_{max} = 100$

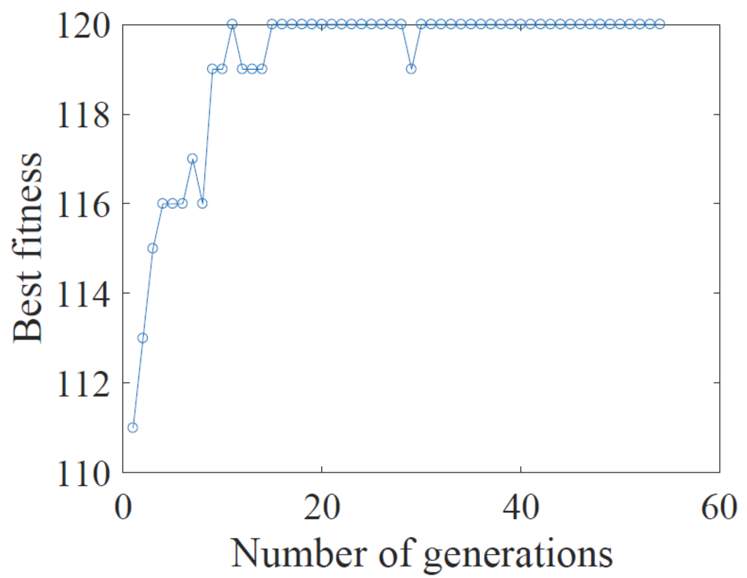
v	4	18	1	16	5	9	3	19	7	13	10	6	5	1	2	17	12	12	2	15
w	6	11	6	12	16	14	4	16	11	18	2	3	7	7	19	16	12	12	9	18

Run 1:

Randomized

Run 2:

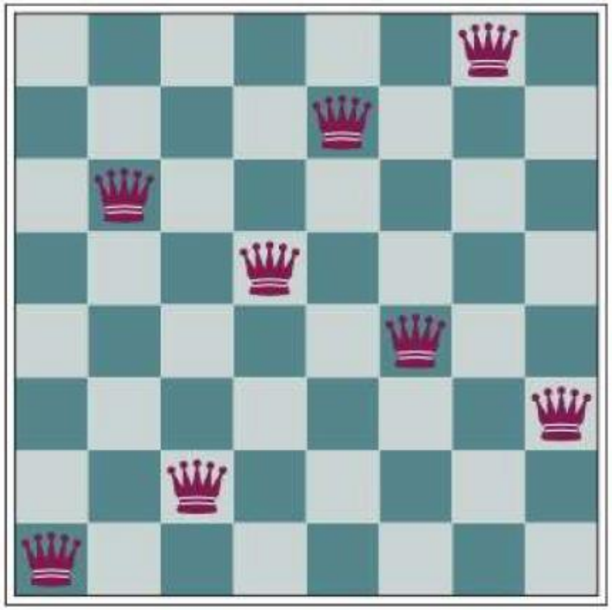




Example illustration - 8-queens

8-queens problem: to place eight queens on a chessboard such that no queen attacks any other

Fitness function f: number of nonattacking pairs of queens



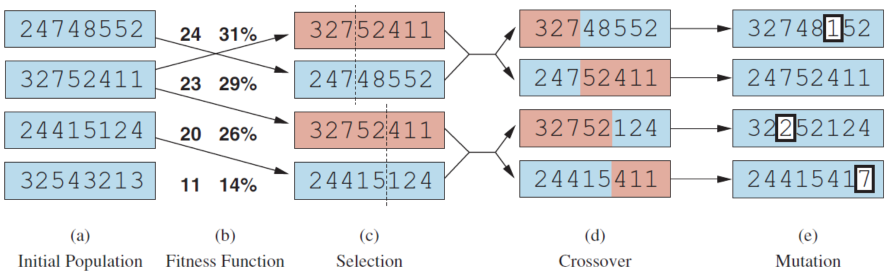
Solution representation

Integer vector

1	6	2	5	7	4	8	3

position of the queen on each column

Example illustration - 8-queens



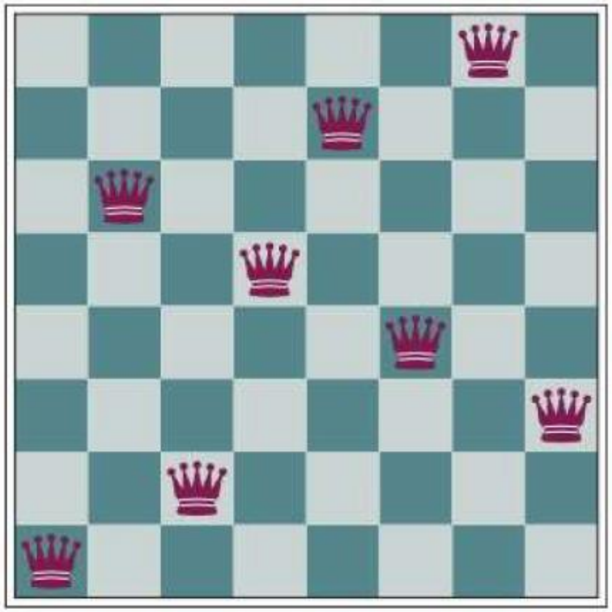
How about another setup?

Representation	Integer vector
Population size	4
Initialization	Random
Parent selection	Fitness proportional
Recombination	One-point crossover
Mutation	Bit-wise mutation
Mutation prob.	$1/n$
Number of offspring	4
Survival selection	fitness based
Termination condition	Reach the best fitness

Example illustration - 8-queens

8-queens problem: to place eight queens on a chessboard such that no queen attacks any other

Fitness function f: number of nonattacking pairs of queens



Solution representation

Permutation

1	6	2	5	7	4	8	3

position of the queen on each column

Genotype space is smaller than that of integer representation, but still contains the optimum

Example illustration - 8-queens

Representation	Permutation
Population size	100
Initialization	Random
Parent selection	Best 2 out of random 5
Recombination	Cut-and-crossfill crossover
Mutation	Swap
Mutation prob.	80%
Number of offspring	2
Survival selection	Fitness based
Termination condition	Reach the best fitness or 10,000 fitness evaluations

Select five solutions from the population uniformly at random, and choose the best two as the parent solutions

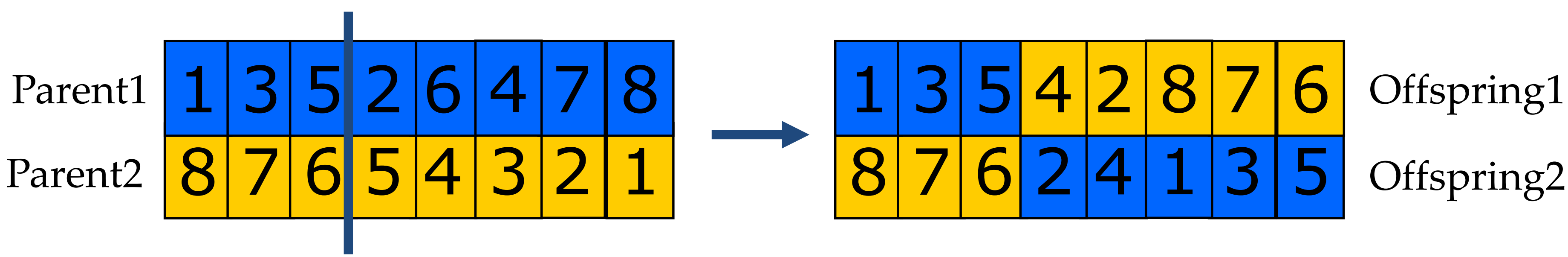
Example illustration - 8-queens

Representation	Permutation
Population size	100
Initialization	Random
Parent selection	Best 2 out of random 5
Recombination	Cut-and-crossfill crossover
Mutation	Swap
Mutation prob.	80%
Number of offspring	2
Survival selection	Fitness based
Termination condition	Reach the best fitness or 10,000 fitness evaluations

Example illustration - 8-queens

Cut-and-crossfill crossover:

1. Select a crossover point randomly;
2. Cut both parents into two segments at this point;
3. Copy the first segment of parent 1 into offspring 1 and the first segment of parent 2 into offspring 2;
4. Scan parent 2 after the crossover point and fill the second segment of offspring 1 with values from parent 2, skipping those that it already contains
5. Do the same for parent 1 and offspring 2



Example illustration - 8-queens

Representation	Permutation
Population size	100
Initialization	Random
Parent selection	Best 2 out of random 5
Recombination	Cut-and-crossfill crossover
Mutation	Swap
Mutation prob.	80%
Number of offspring	2
Survival selection	Fitness based
Termination condition	Reach the best fitness or 10,000 fitness evaluations

Swap values of two randomly chosen positions

1	3	5	2	6	4	7	8	→	1	3	7	2	6	4	5	8

Example illustration - 8-queens

Representation	Permutation
Population size	100
Initialization	Random
Parent selection	Best 2 out of random 5
Recombination	Cut-and-crossfill crossover
Mutation	Swap
Mutation prob.	80%
Number of offspring	2
Survival selection	Fitness based
Termination condition	Reach the best fitness or 10,000 fitness evaluations

Remove the worst two from the population and two offspring

Example illustration - 8-queens

Representation	Permutation
Population size	100
Initialization	Random
Parent selection	Best 2 out of random 5
Recombination	Cut-and-crossfill crossover
Mutation	Swap
Mutation prob.	80%
Number of offspring	2
Survival selection	Fitness based
Termination condition	Reach the best fitness or 10,000 fitness evaluations

Example illustration - 8-queens

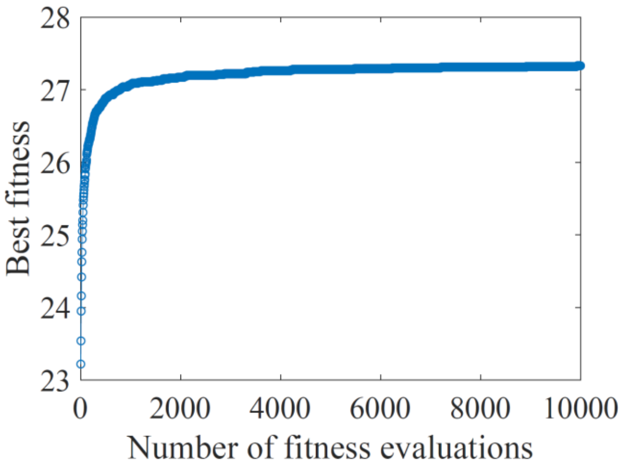
Average of 100 independent runs

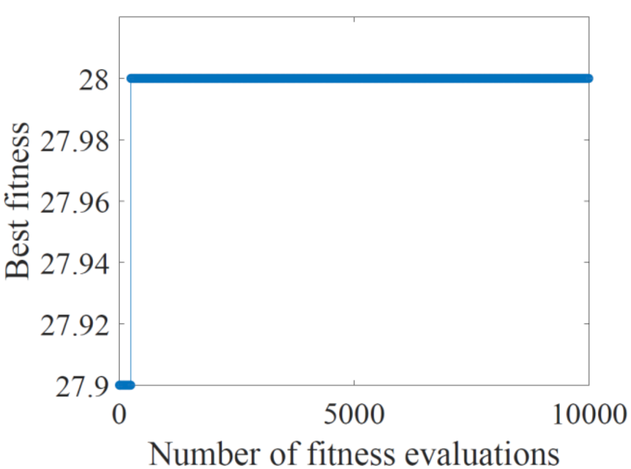
Setup 1: Setup 2:

The number of fitness evaluations

7094 13

Curve change of the best fitness





The setup of components has a large influence on the performance

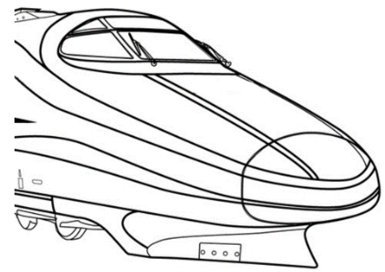
Application: High-speed train head design

Problem: optimize the efficiency of the train head

extremely hard to apply traditional optimization methods

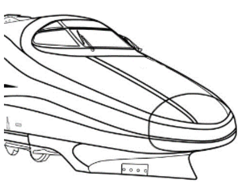
Representation:

parameterize

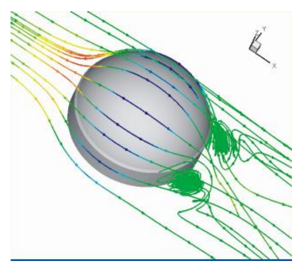


represented as a vector of parameters

Fitness:



x_i



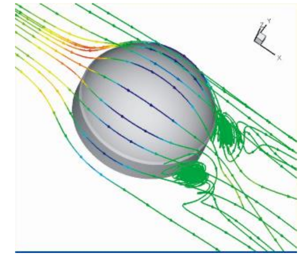
test by simulation

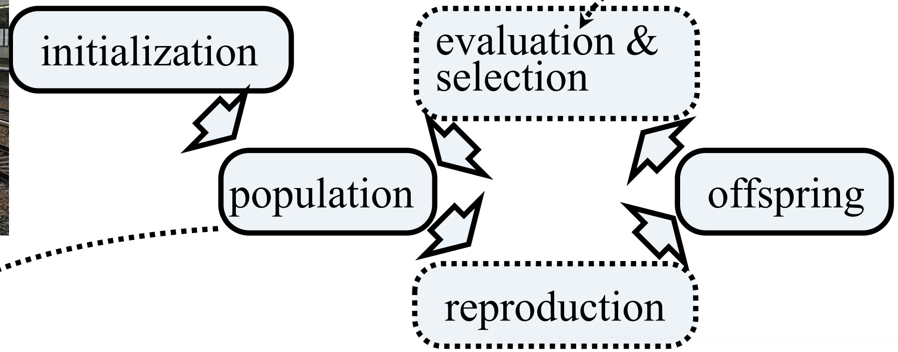
$f(x_i)$

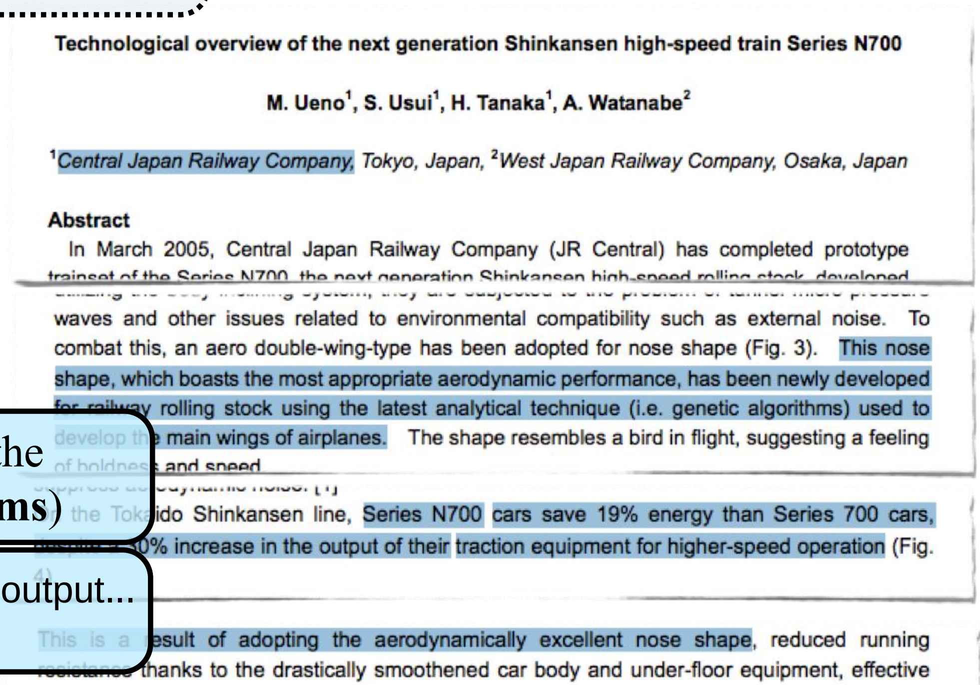
Application: High-speed train head design

Series 700

Series N700







Technological overview of the next generation Shinkansen high-speed train Series N700

M. Ueno[1], S. Usui[1], H. Tanaka[1], A. Watanabe[2]

[1]Central Japan Railway Company, Tokyo, Japan, [2]West Japan Railway Company, Osaka, Japan

Abstract

In March 2005, Central Japan Railway Company (JR Central) has completed prototype trainset of the Series N700, the next generation Shinkansen high-speed rolling stock, developed

... waves and other issues related to environmental compatibility such as external noise. To combat this, an aero double-wing-type has been adopted for nose shape (Fig. 3). This nose shape, which boasts the most appropriate aerodynamic performance, has been newly developed for railway rolling stock using the latest analytical technique (i.e. genetic algorithms) used to develop the main wings of airplanes. The shape resembles a bird in flight, suggesting a feeling of boldness and speed.

... the Tokaido Shinkansen line, Series N700 cars save 19% energy than Series 700 cars, despite a 30% increase in the output of their traction equipment for higher-speed operation (Fig. 4).

This is a result of adopting the aerodynamically excellent nose shape, reduced running resistance thanks to the drastically smoothened car body and under-floor equipment, effective

this nose ... has been newly developed ... using the latest analytical technique (i.e. **genetic algorithms**)

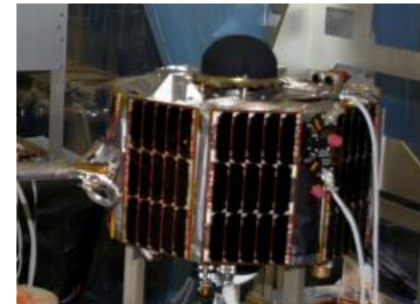
N700 cars save **19%** energy ... **30%** increase in the output...
This is a result of adopting the ... nose shape

http://www.lamda.nju.edu.cn/qianc/

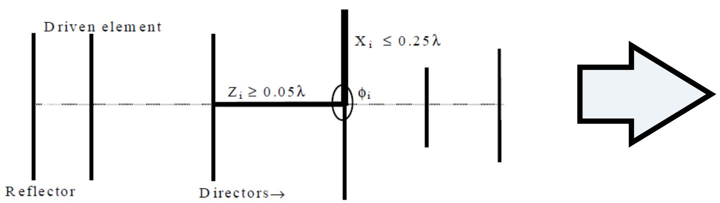
Application: Antenna design

Problem: optimize the efficiency of the antenna

extremely hard to apply traditional optimization methods

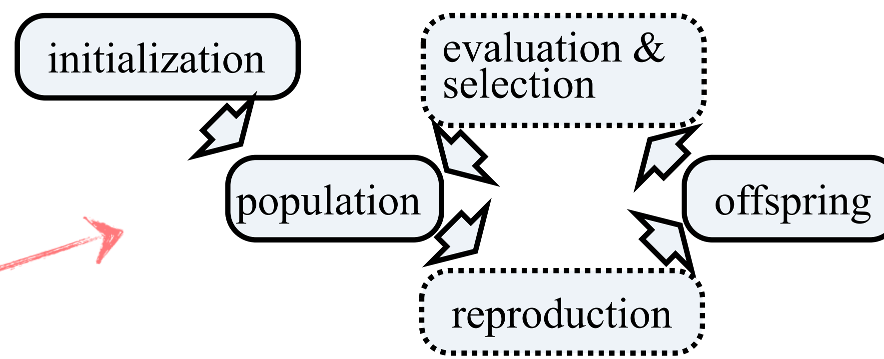


Representation:



a sequence of operators
forward, rotate-x
rotate-y, rotate-z

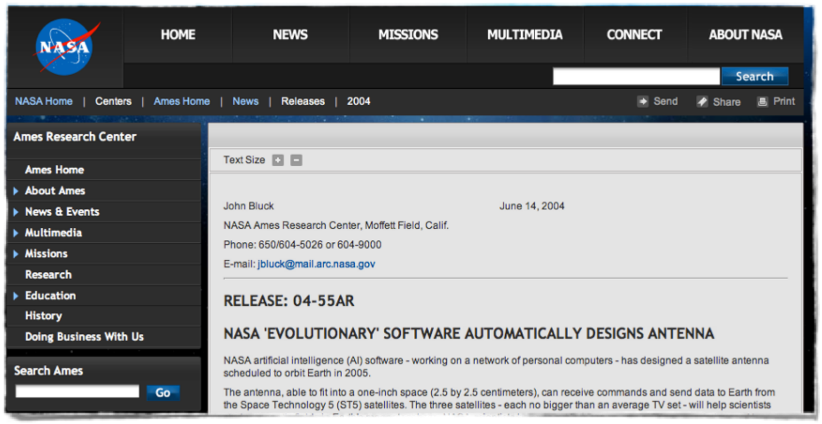
Fitness by simulation test

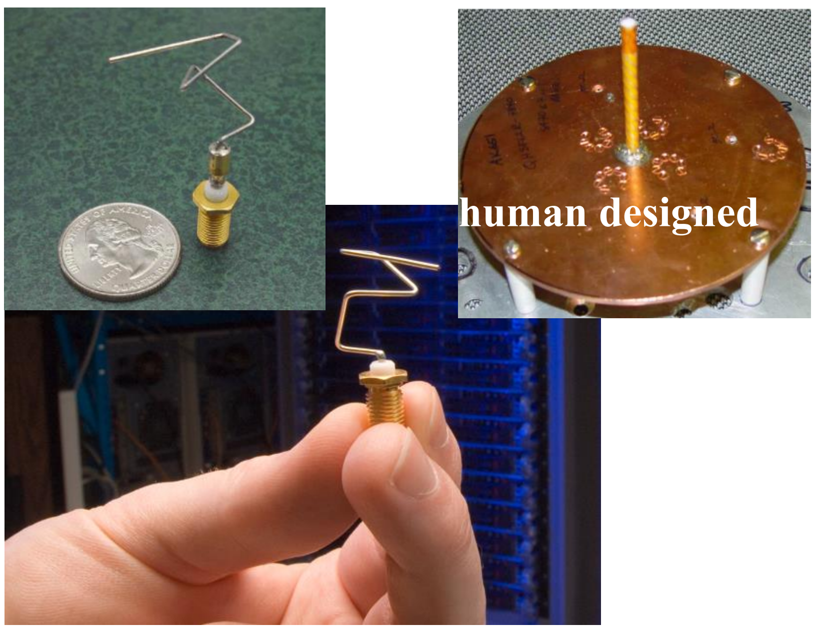


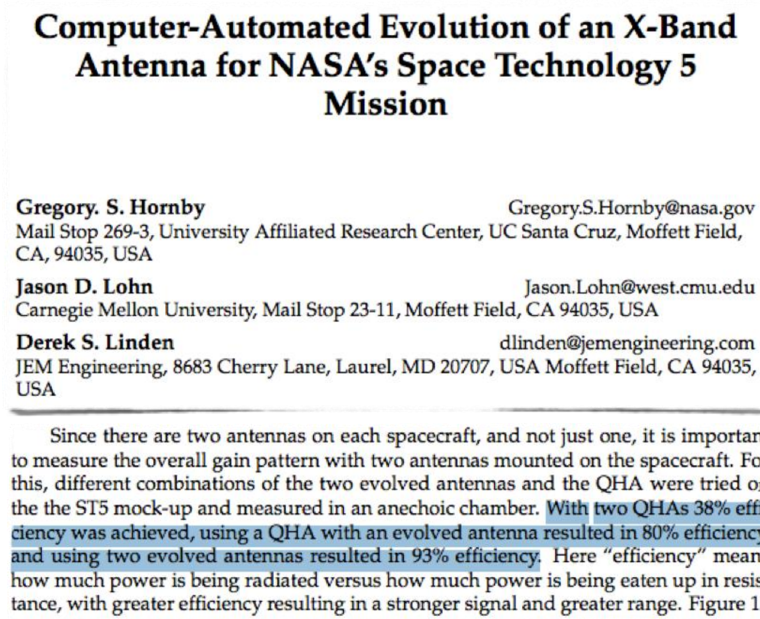
easy to test a given solution

use EAs!

Application: Antenna design







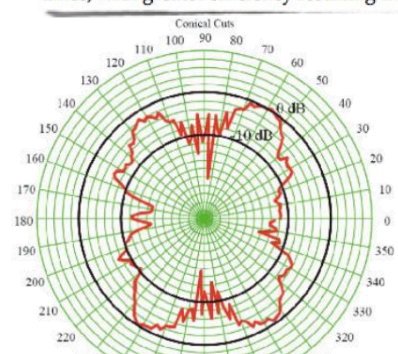
Computer-Automated Evolution of an X-Band Antenna for NASA's Space Technology 5 Mission

Gregory. S. Hornby Gregory.S.Hornby@nasa.gov
Mail Stop 269-3, University Affiliated Research Center, UC Santa Cruz, Moffett Field, CA, 94035, USA

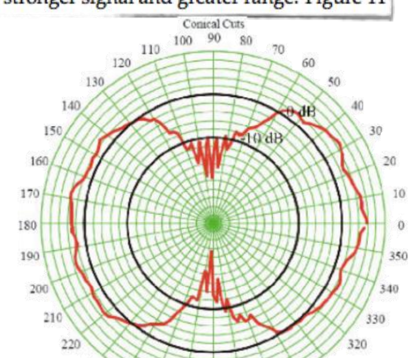
Jason D. Lohn Jason.Lohn@west.cmu.edu
Carnegie Mellon University, Mail Stop 23-11, Moffett Field, CA 94035, USA

Derek S. Linden dlinden@jemengineering.com
JEM Engineering, 8683 Cherry Lane, Laurel, MD 20707, USA Moffett Field, CA 94035, USA

Since there are two antennas on each spacecraft, and not just one, it is important to measure the overall gain pattern with two antennas mounted on the spacecraft. For this, different combinations of the two evolved antennas and the QHA were tried on the the ST5 mock-up and measured in an anechoic chamber. With two QHAs 38% efficiency was achieved, using a QHA with an evolved antenna resulted in 80% efficiency, and using two evolved antennas resulted in 93% efficiency. Here "efficiency" means how much power is being radiated versus how much power is being eaten up in resistance, with greater efficiency resulting in a stronger signal and greater range. Figure 11



QHAs (human designed)
38% efficiency



evolved antennas
93% efficiency

Application: Biological evolution

南京大學 NANJING UNIVERSITY | 新闻网 NJU NEWS 南京大学新闻中心主办

新闻关键字搜索 理论园地 南京大学报

首 页 综合新闻 专题新闻 理论园地 讲话与部署 南雍号 媒体传真 学术动态 影像南大 校园动态 学人视点 南大人

首页 - 综合新闻 2020-01-17 作者：地球科学与工程学院 来源：地球科学与工程学院

《Science》刊登南京大学地球科学与工程学院研究成果：大数据和超算揭秘古生代海洋生物多样性演化

北京时间1月17日，国际权威期刊《Science》以研究长文的形式在线发表了南京大学、中国科学院南京地质古生物所樊隽轩教授、沈树忠院士等的论文“A high-resolution summary of Cambrian to Early Triassic marine invertebrate biodiversity”。该研究利用古生物大数据、超算和遗传算法等全新的方法和手段，基于化石记录重现了生命演化历史，改变了当前对古生代海洋生物多样性演化的认知。

最近更新

如何让专利“活”起来？全国百所高校聚...
2020.10.15

扬子江生态文明创新中心首届理事会召开...
2020.10.15

电子科学与工程学院启动“星火培优”学...
2020.10.15

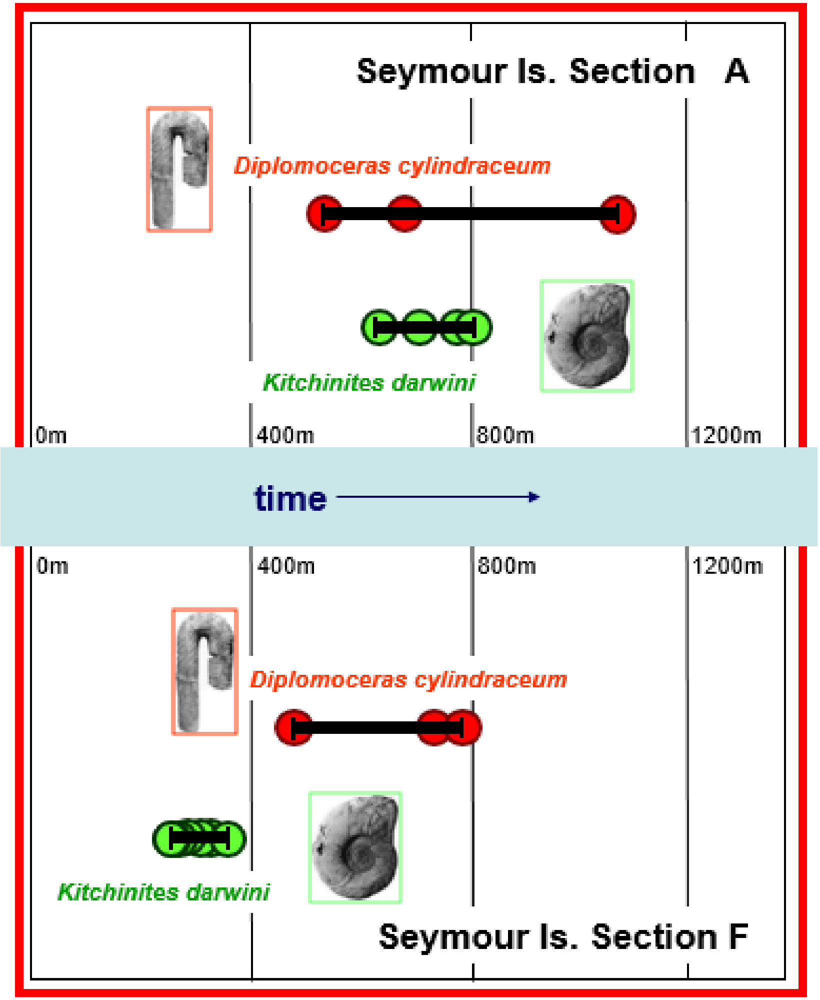
我校举行“墨子杯”兵棋推演大赛校内选...

Application: Biological evolution

Each section contains many taxa

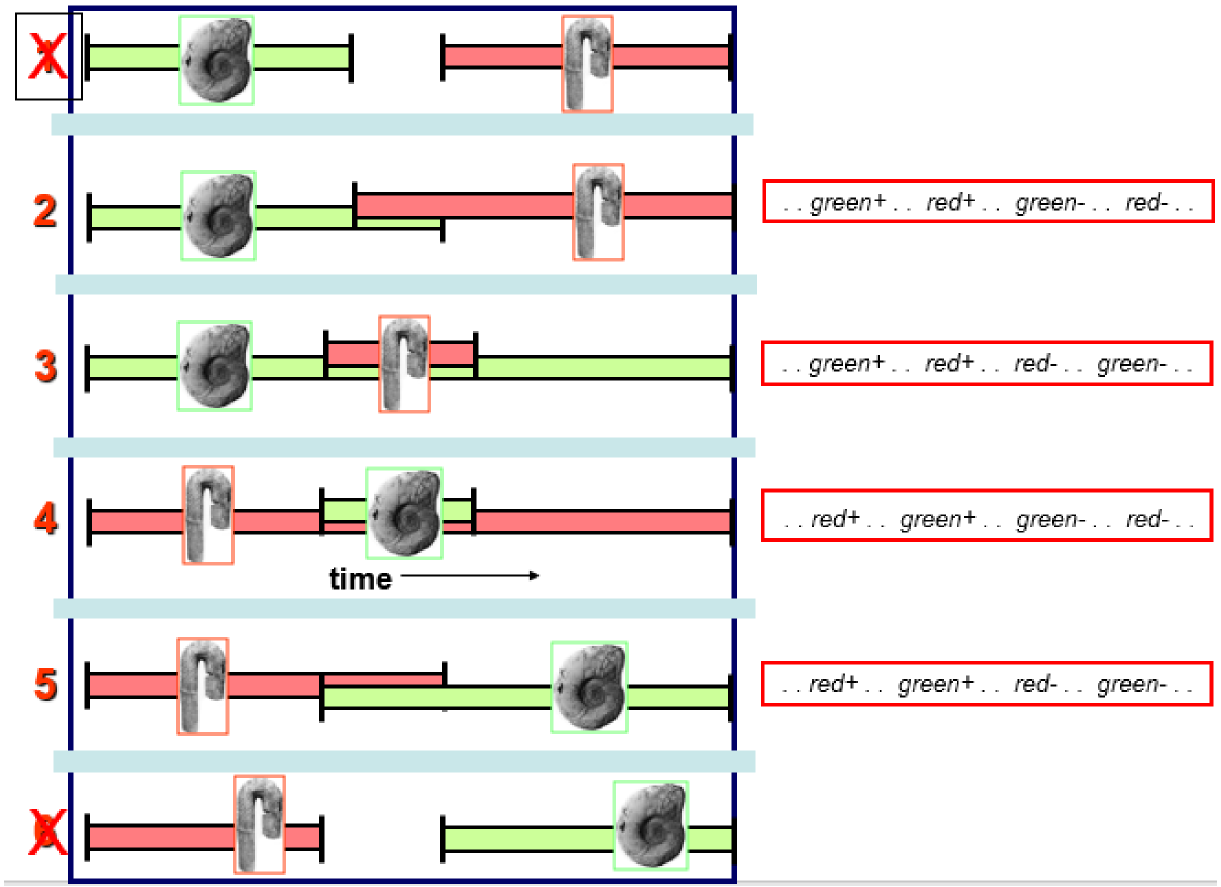
Each taxon on each section has two biological events: FAD, LAD

Problem: to find a sequence of biological events of taxa, which fits the observed biological events best

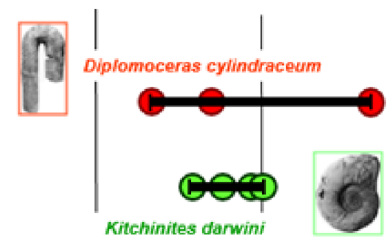


Application: Biological evolution

Two taxa: 6 possible sequences of biological events



Symbiotic constraints

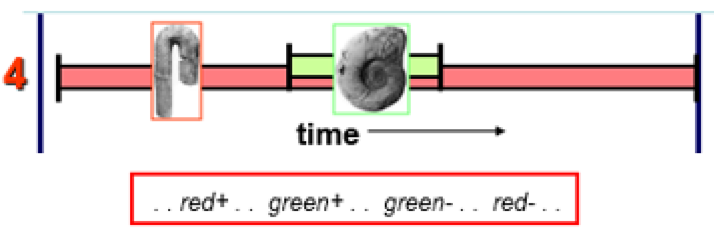


⇩

Sequences 1 and 6 are infeasible

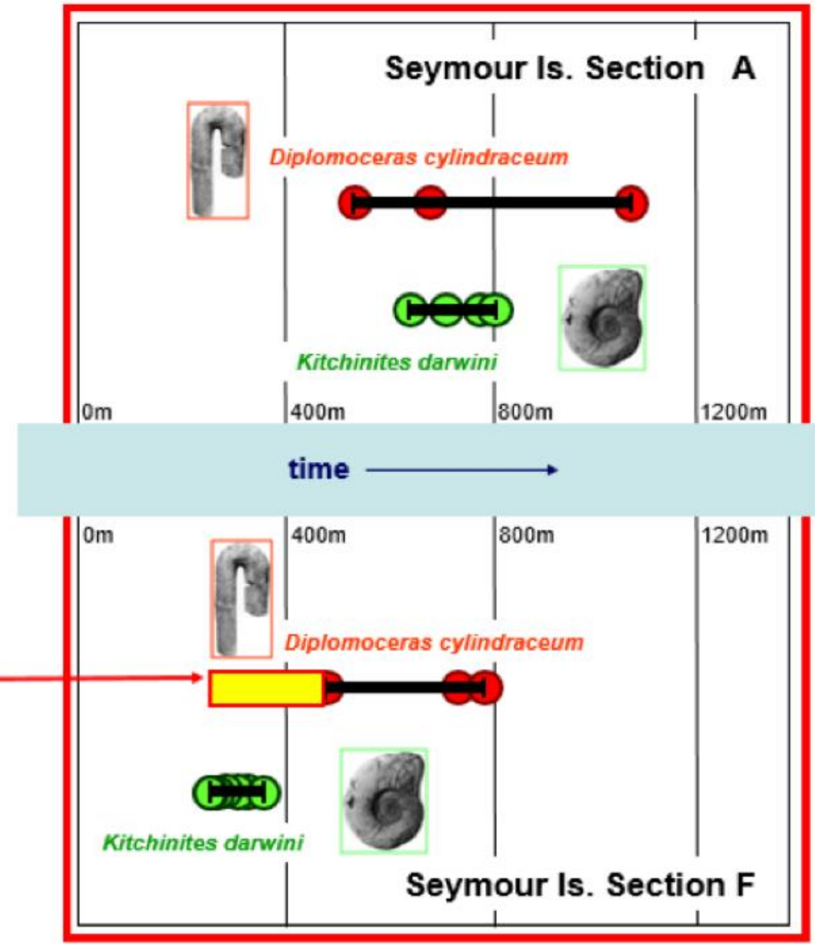
Application: Biological evolution

Objective function: total range extensions to make the selected sequence and the observed biological events consistent



延限延展量

RANGE EXTENSIONS



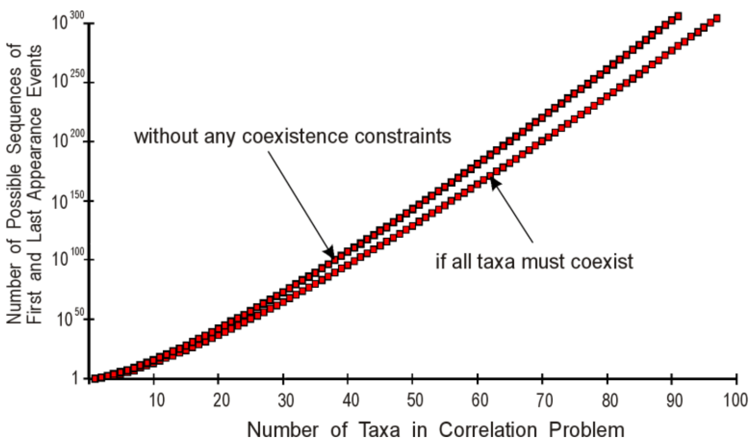
The smaller, the better

Application: Biological evolution

The minimization problem:

- Solution: sequence of biological events of taxa
- Objective function: total range extensions to make the selected sequence and the observed biological events consistent
- Constraints: symbiotic constraints; FAD-LAD constraints

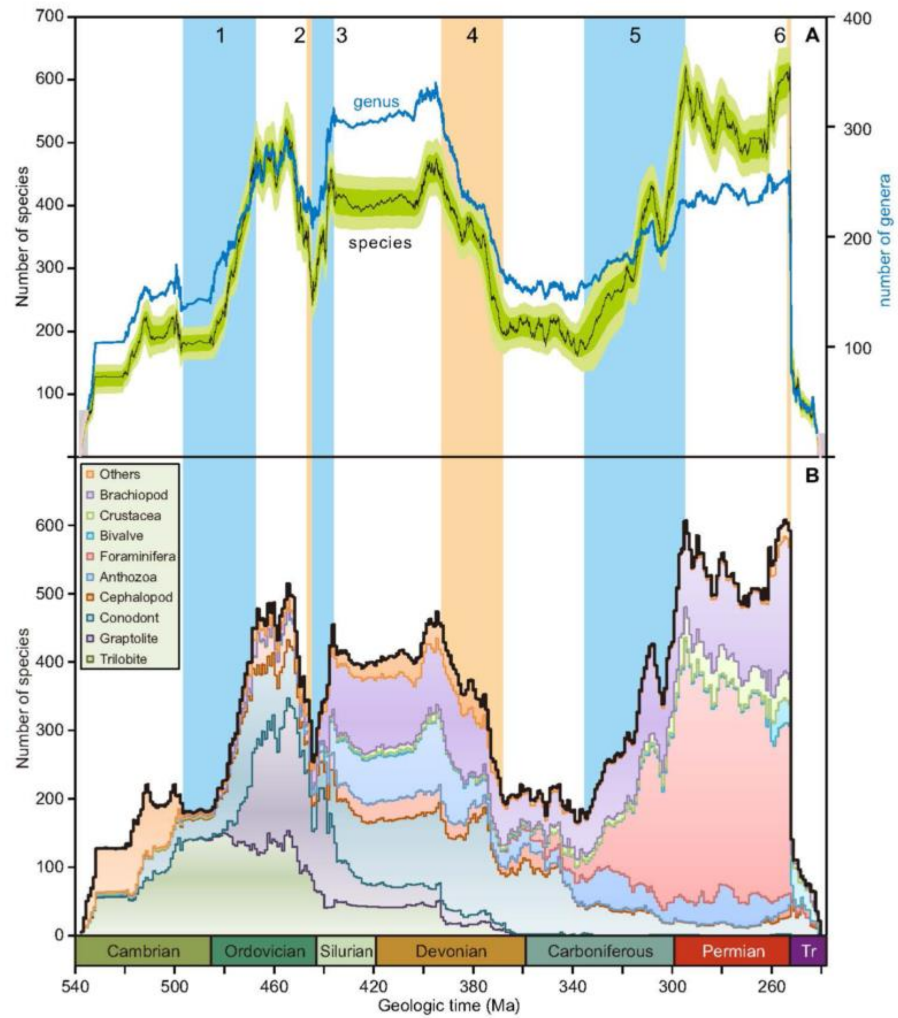
Taxa	1	2	3	4	5	6	7
Sequences	1	6	90	2,520	113,400	7,484,400	681,080,400



A very difficult optimization problem!

Application: Biological evolution

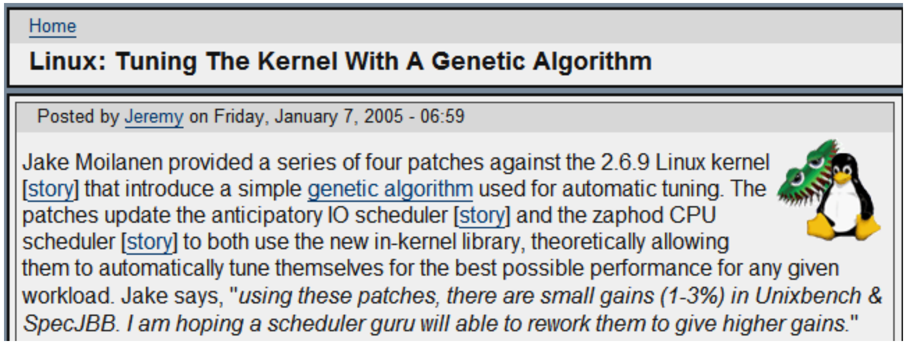
“利用古生物大数据、超算和遗传算法等”



Thanks J. Fan and X. Hou for providing the figures

And more

optimizing operating systems:

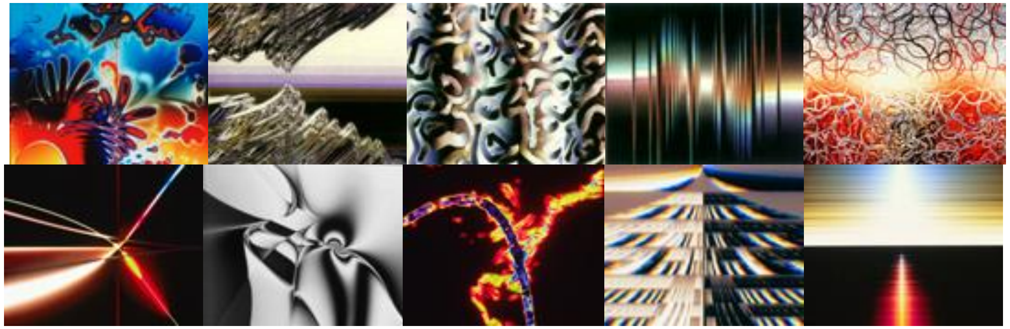

Home

Linux: Tuning The Kernel With A Genetic Algorithm

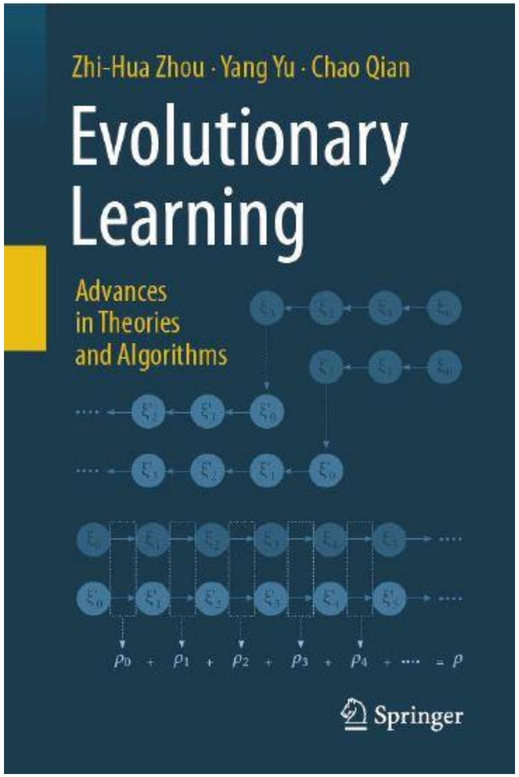
Posted by Jeremy on Friday, January 7, 2005 - 06:59

Jake Moilanen provided a series of four patches against the 2.6.9 Linux kernel [story] that introduce a simple genetic algorithm used for automatic tuning. The patches update the anticipatory IO scheduler [story] and the zaphod CPU scheduler [story] to both use the new in-kernel library, theoretically allowing them to automatically tune themselves for the best possible performance for any given workload. Jake says, "*using these patches, there are small gains (1-3%) in Unixbench & SpecJBB. I am hoping a scheduler guru will able to rework them to give higher gains.*"

interactive art design:



machine learning:



As long as solutions can be evaluated, EAs can be applied

Summary

- Evolutionary algorithms: Origins
- Evolutionary algorithms: Components
- Evolutionary algorithms: Applications

References

- K. A. De Jong. Evolutionary Computation – A Unified Approach. Chapter 2.

- A. E. Eiben and J. E. Smith. Introduction to Evolutionary Computing. Chapters 2-3.