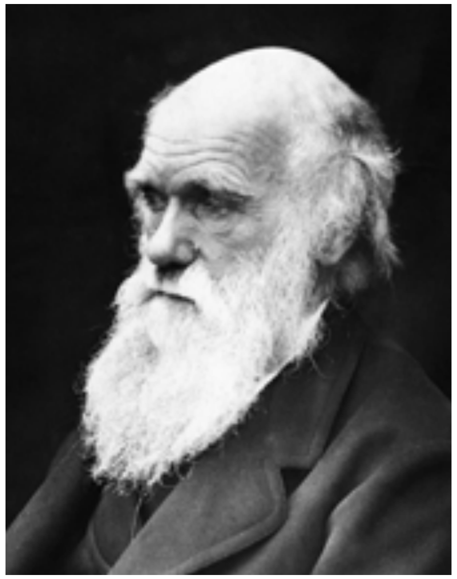
Introduction

Yang Yu and Zhi-Hua Zhou
LAMDA Group
National Key Laboratory for Novel Software Technology
Nanjing University, China

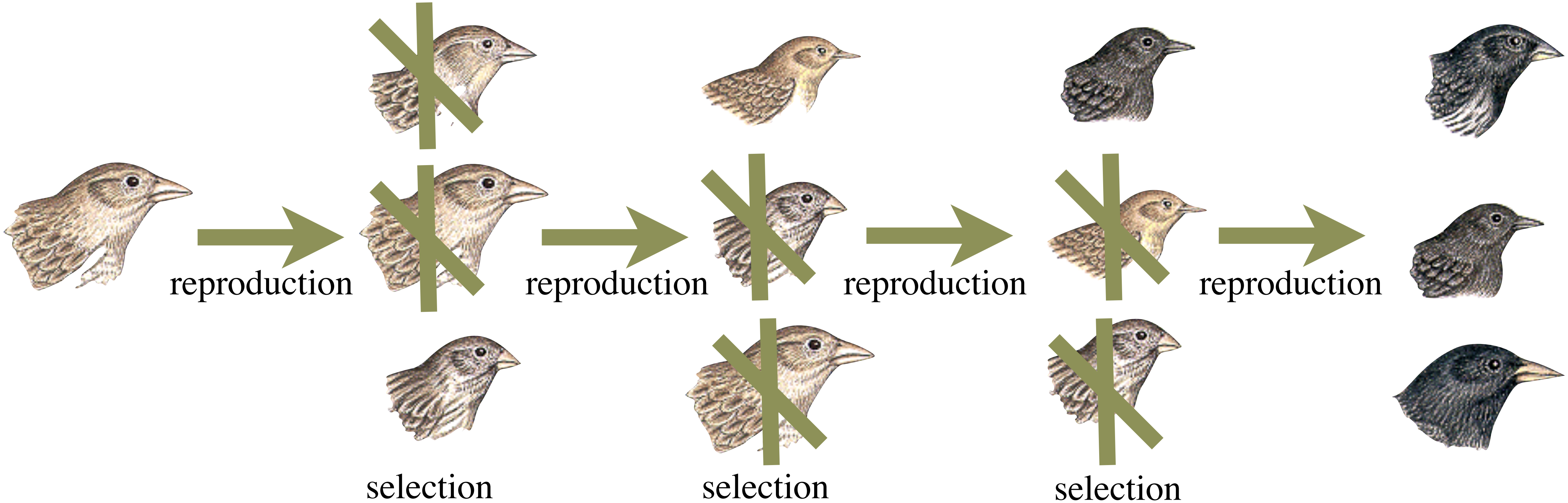
Biological evolution



Charles Darwin
1809-1882

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

reproduction with variation + nature selection



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

Evolution v.s. optimization

Alan Turing
1912-1954

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

building intelligent machine

"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 the sides)

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

Evolutionary algorithms

Genetic Algorithms (GA)
for optimization in discrete domains

[J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.]

Evolutionary Strategies (ES)
for optimization in continuous domains

[I. Rechenberg. *Evolutionstrategie: Optimierung Technisher Systeme nach Prinzipien des Biologischen Evolution*. Fromman-Hozlboog Verlag, Stuttgart, 1973.]

Evolutionary Programming (EP)
structured solutions, e.g., trees, graphs

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

Other variants:
- Genetic Programming
- Differential Evolutionary Algorithm
- ...

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

general Evolutionary Algorithms (EAs)

Evolutionary algorithms

EAs share a common routine

For an optimization problem $\arg\max_{x \in \mathcal{X}} f(x)$

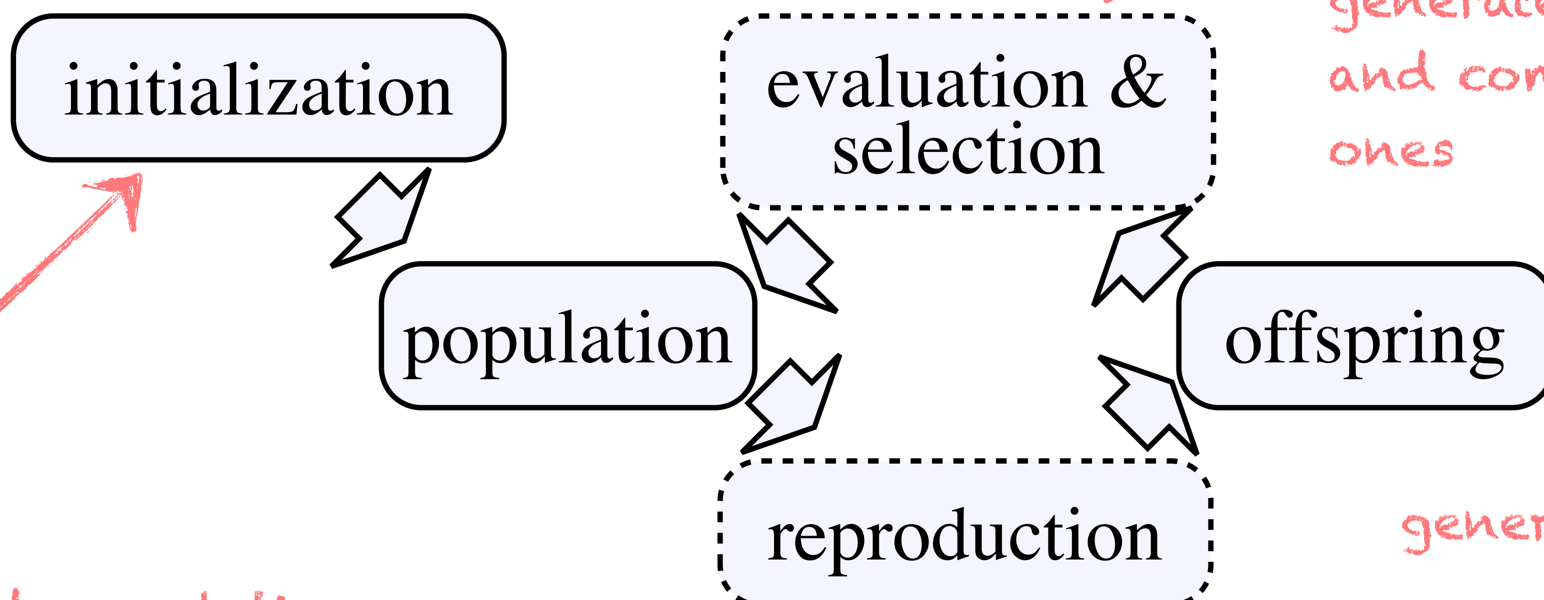
fitness function

First, choose a representation of solutions

binary vectors, real vectors, trees, graphs ...

Then, follow a cycle:

calculate the fitness of newly generated solutions by f(), and competes with the old ones



generated offspring solutions

generate random solutions and evaluate them

manipulate solutions in the current population

A typical EA

Encode a solution as a binary vector, $\mathcal{X} = \{0,1\}^n$

problem independent

```
Pop ← n randomly drawn solutions from X
for t=1,2,... do
    Pop^m ← {mutate(s) | ∀s ∈ Pop}, the mutated solutions
    Pop^c ← {crossover(s_1, s_2) | ∃s_1, s_2 ∈ Pop^m}, the recombined solutions
    evaluate every solution in Pop^c by f(s)(∀s ∈ Pop^c)
    Pop^s ← selected solutions from Pop and Pop^c
    Pop ← Pop^s
    terminate if meets a stopping criterion
end for
```

problem dependent

(stop whenever you want)

$mutate(s)$: flip each bit (0 to 1, or 1 to 0) with probability $1/n$

(1,0,1,1,0,0,0,0,1) $\longrightarrow$ (1,1,1,0,0,1,0,0,0)

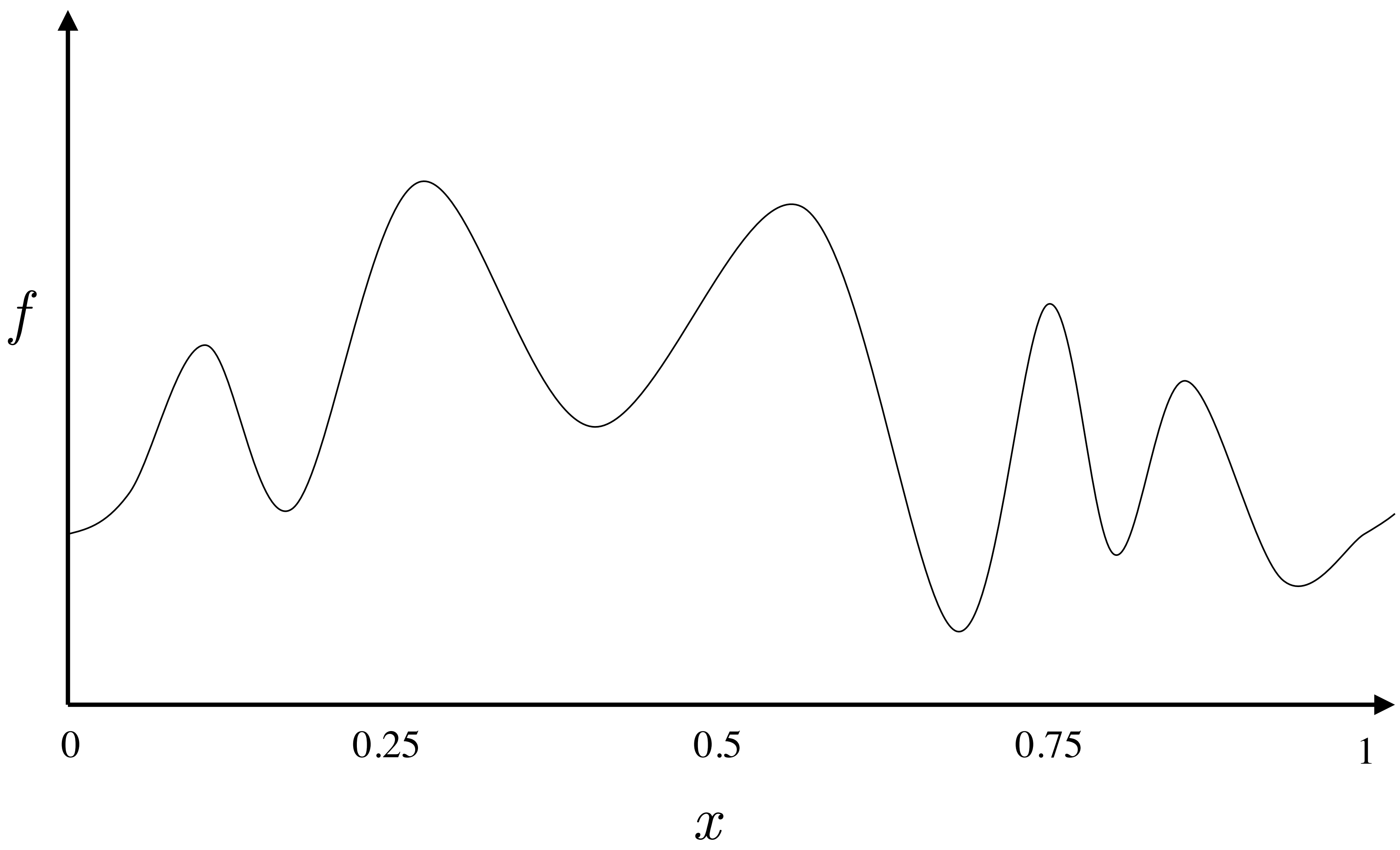
$crossover(s_1, s_2)$: select a position at random, exchange the parts after the position

(1,0,1,1,0,0,0,0,1) + (0,0,1,0,0,1,1,0,1) $\longrightarrow$ (1,0,1,1,0,1,1,0,1) + (0,0,1,0,0,0,0,0,1)

selection: select n solutions with the best fitness

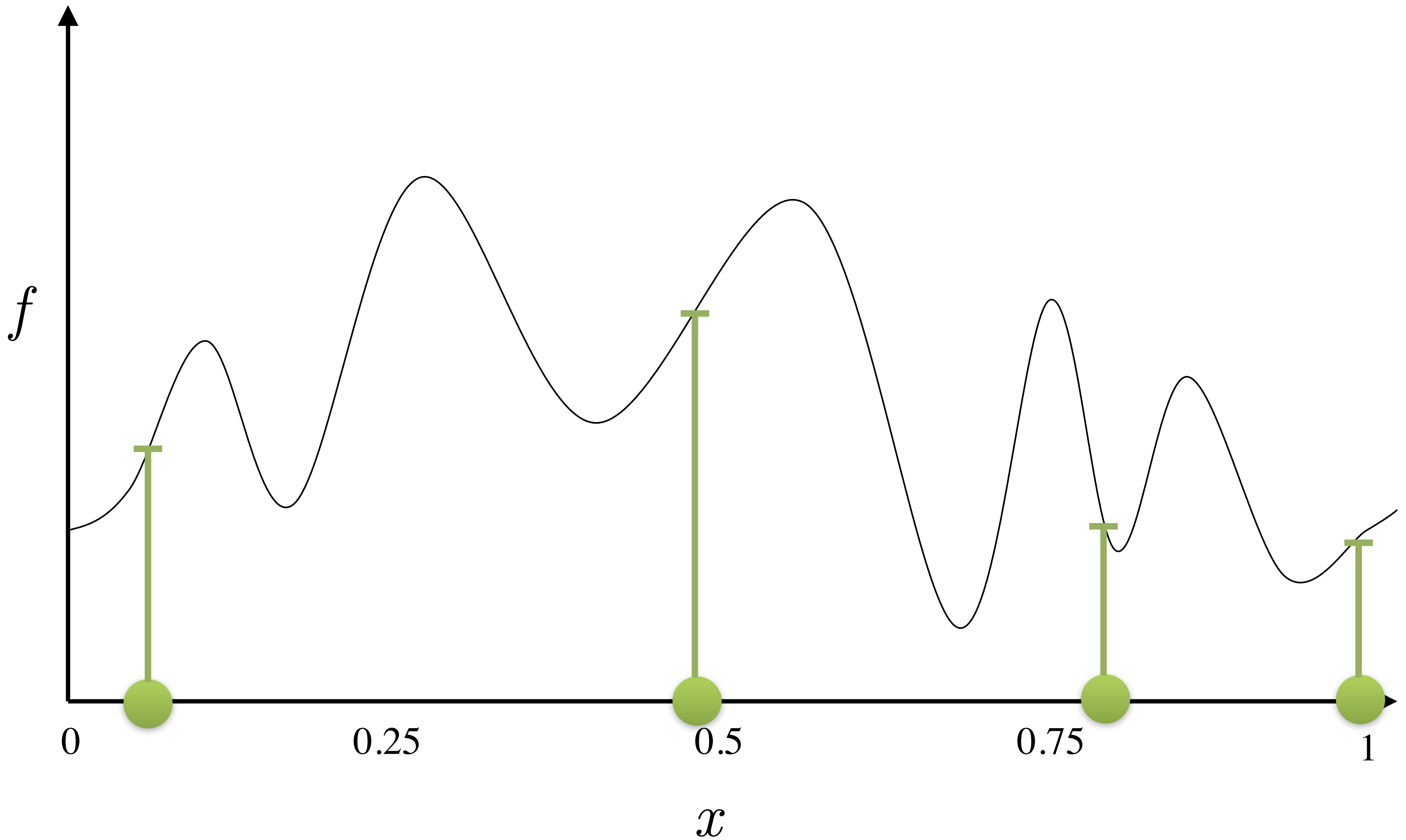
An illustration of running

$\mathcal{X} = [0, 1]$



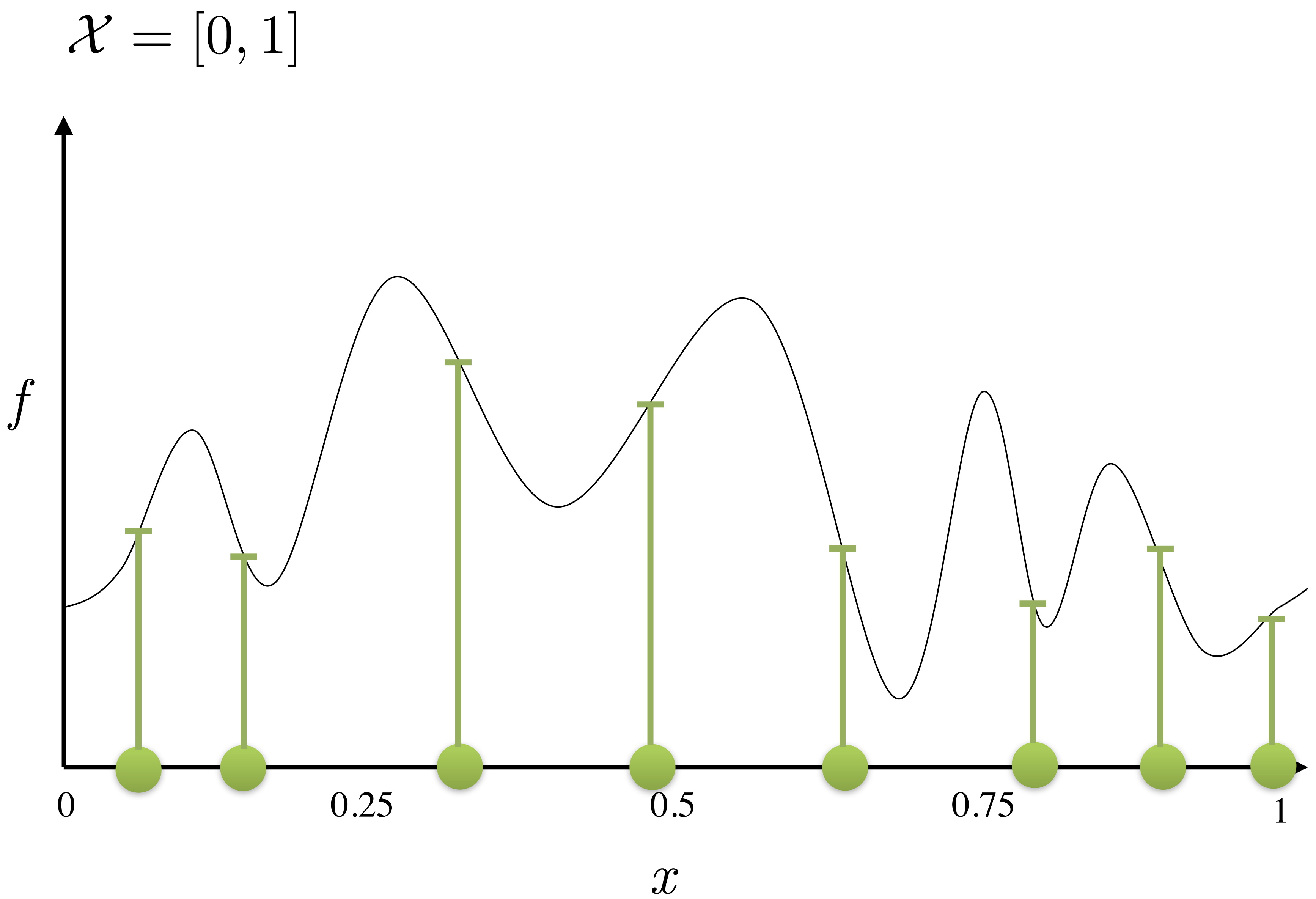
An illustration of running

$\mathcal{X} = [0, 1]$



initialization
evaluation

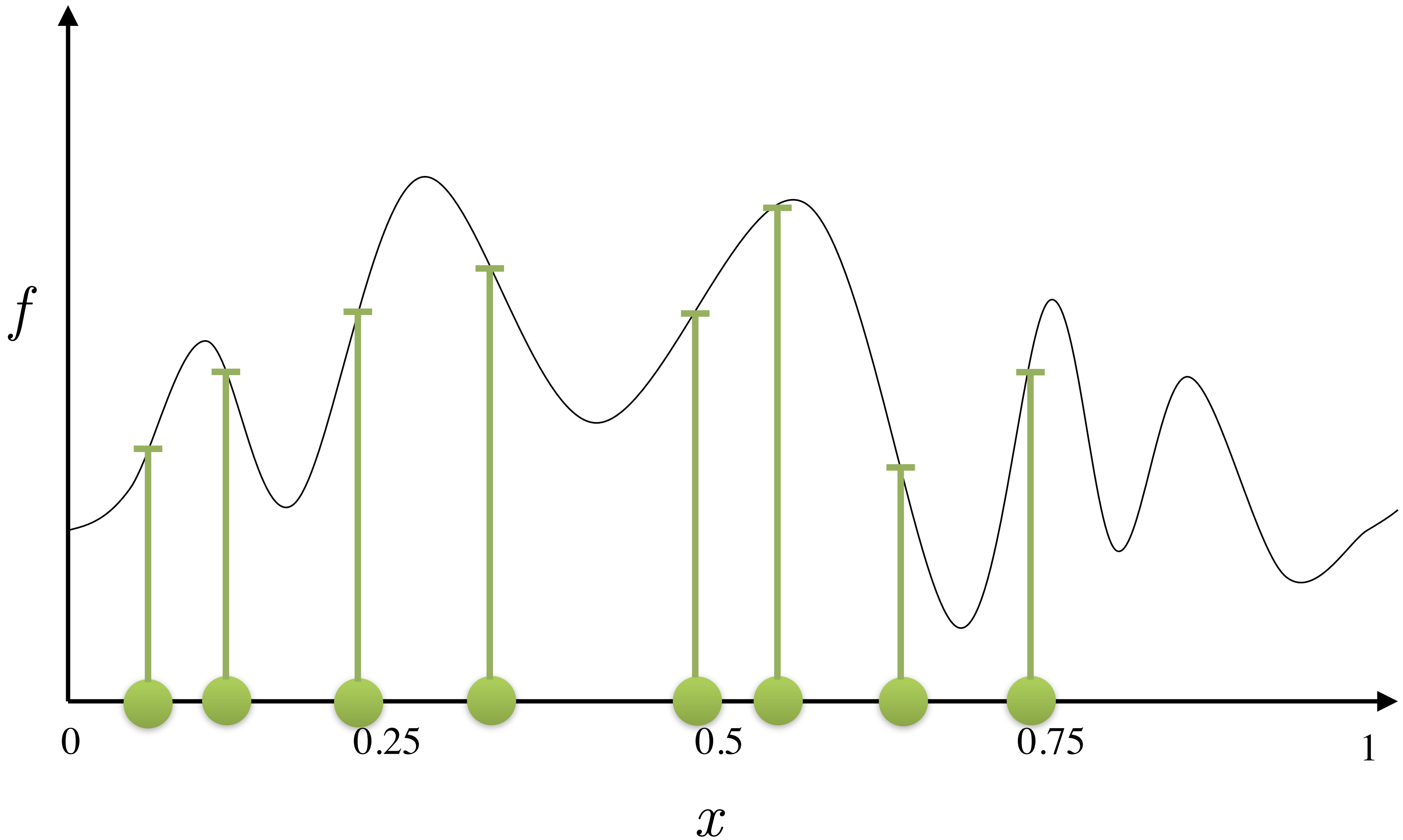
An illustration of running



initialization
evaluation
reproduction
evaluation

An illustration of running

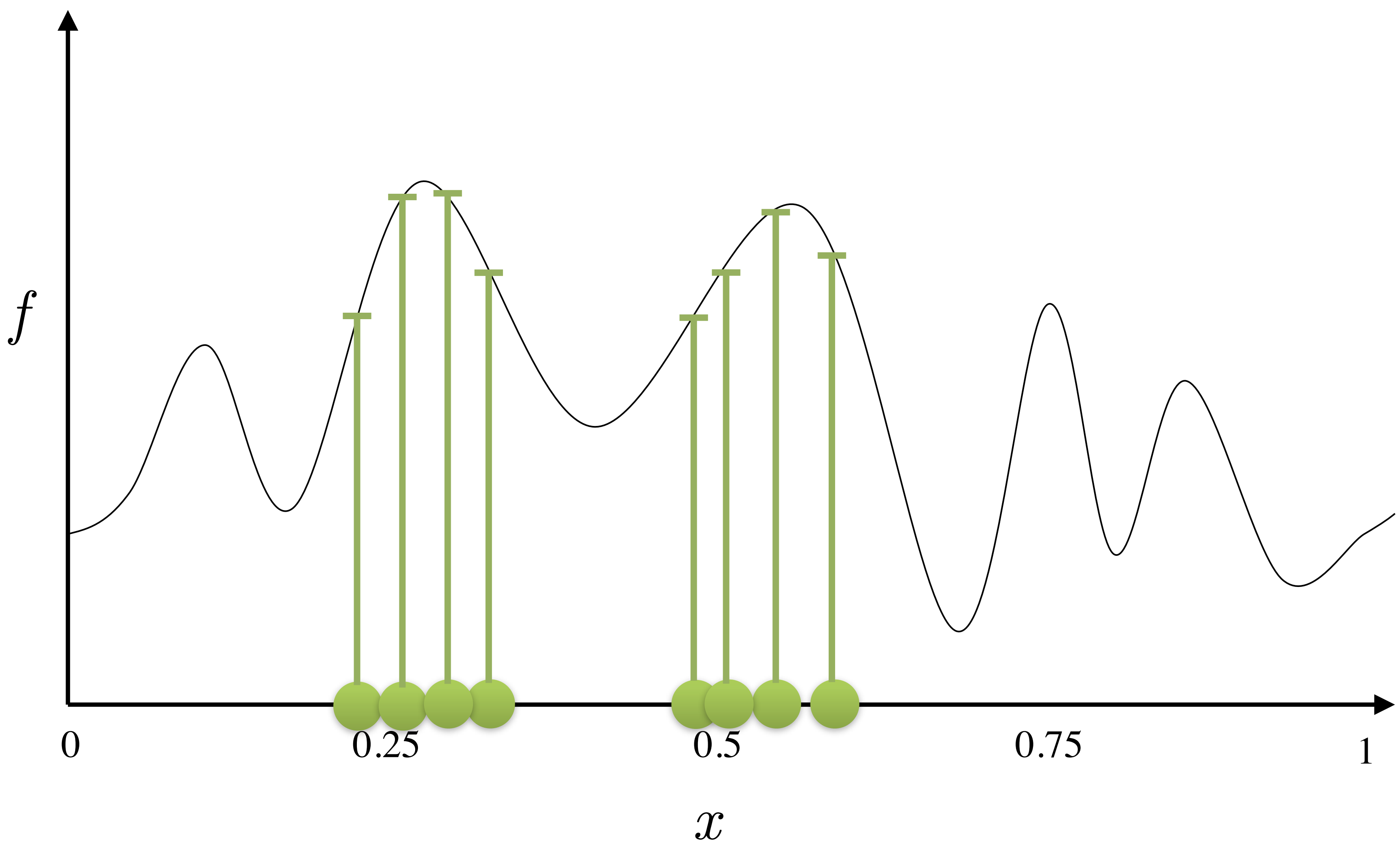
$\mathcal{X} = [0, 1]$



initialization
evaluation
reproduction
evaluation
selection
reproduction
evaluation

An illustration of running

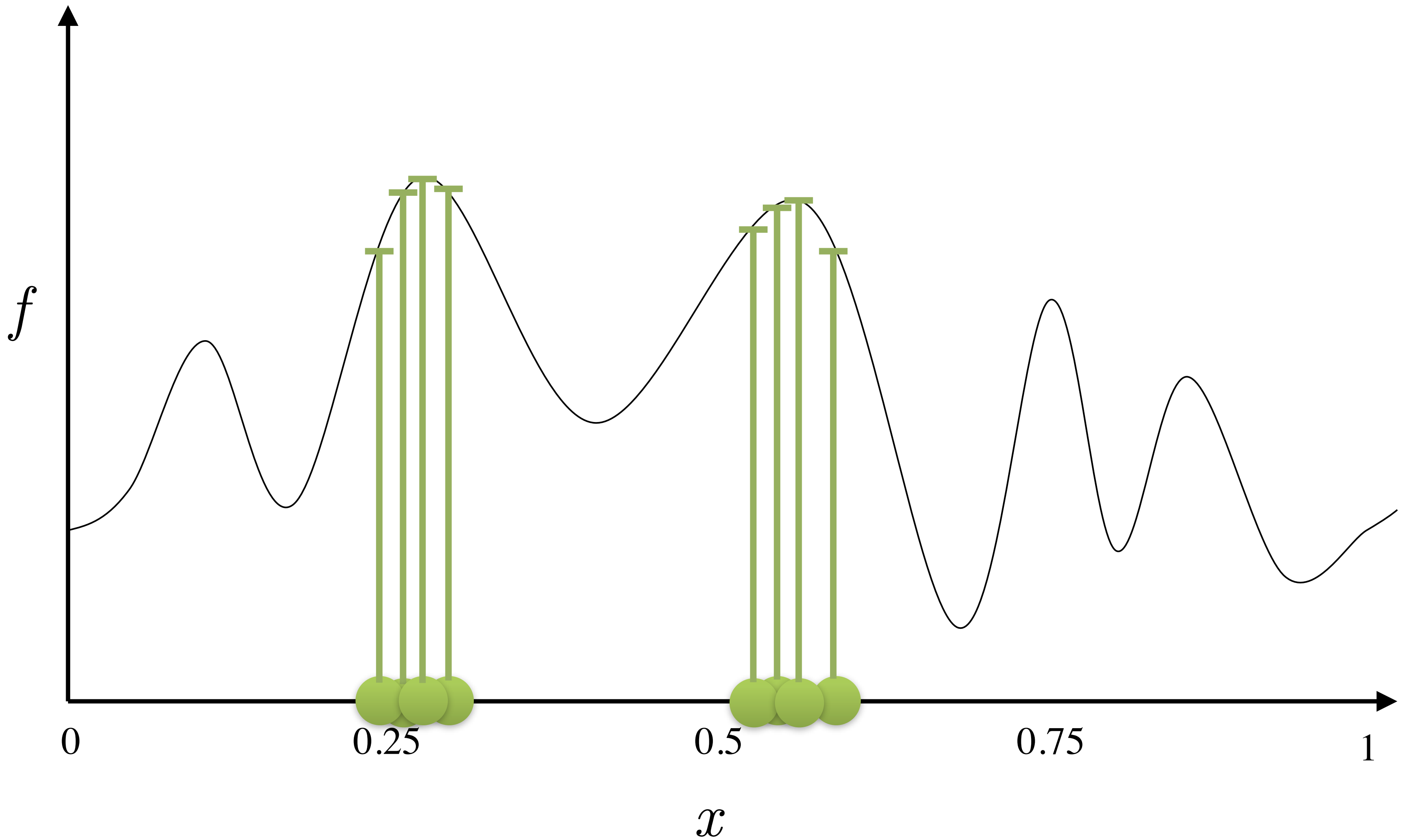
$\mathcal{X} = [0, 1]$



initialization
evaluation
reproduction
evaluation
selection
reproduction
evaluation
selection
reproduction
evaluation

An illustration of running

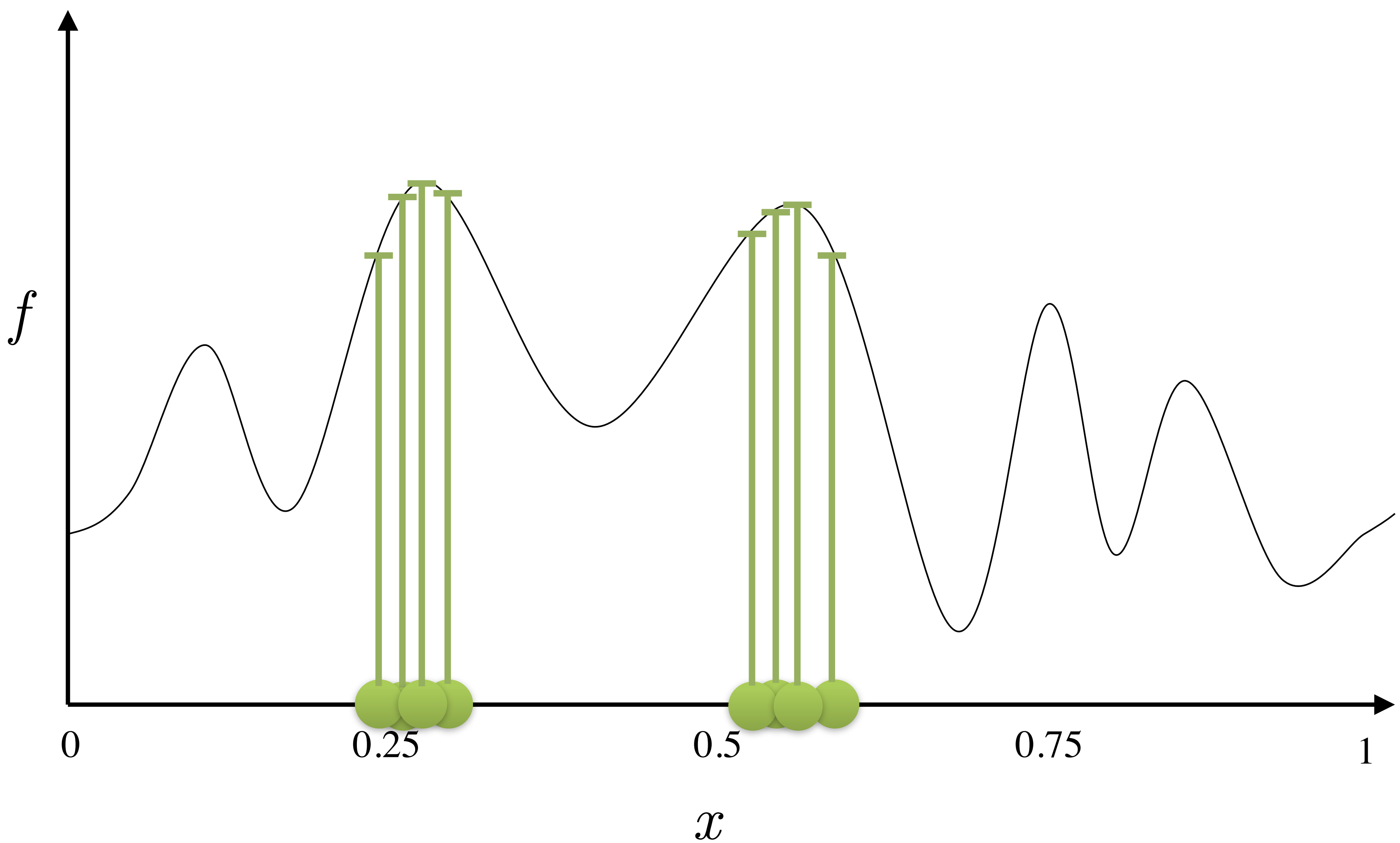
$\mathcal{X} = [0, 1]$



initialization
evaluation
reproduction
evaluation
selection
reproduction
evaluation
selection
reproduction
evaluation
selection
reproduction
evaluation
...

An illustration of running

$\mathcal{X} = [0, 1]$



initialization
evaluation
reproduction
evaluation
selection
reproduction
evaluation
selection
reproduction
evaluation
selection
reproduction
evaluation
...

EAs only need to evaluate solutions

can be applied without knowledge of the problem

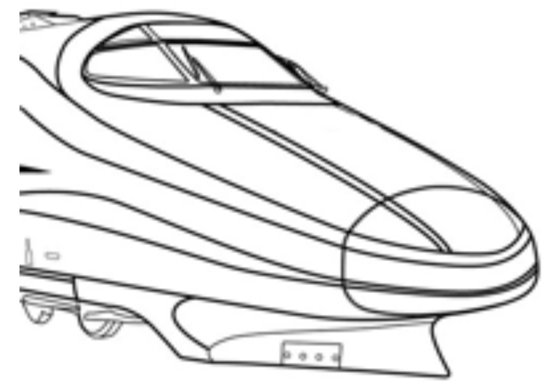
Applications: High-speed train head design

Problem: optimize the efficiency of the train head

extremely hard to apply traditional optimization methods

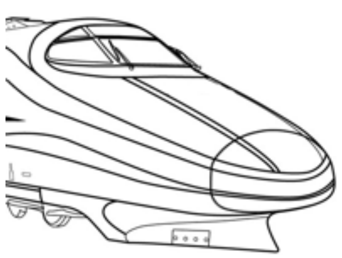
Representation:

parameterize

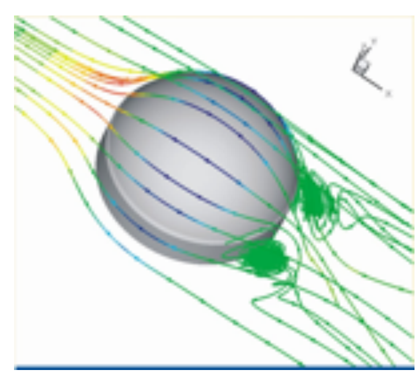


represented as a vector of parameters

Fitness:



x_i



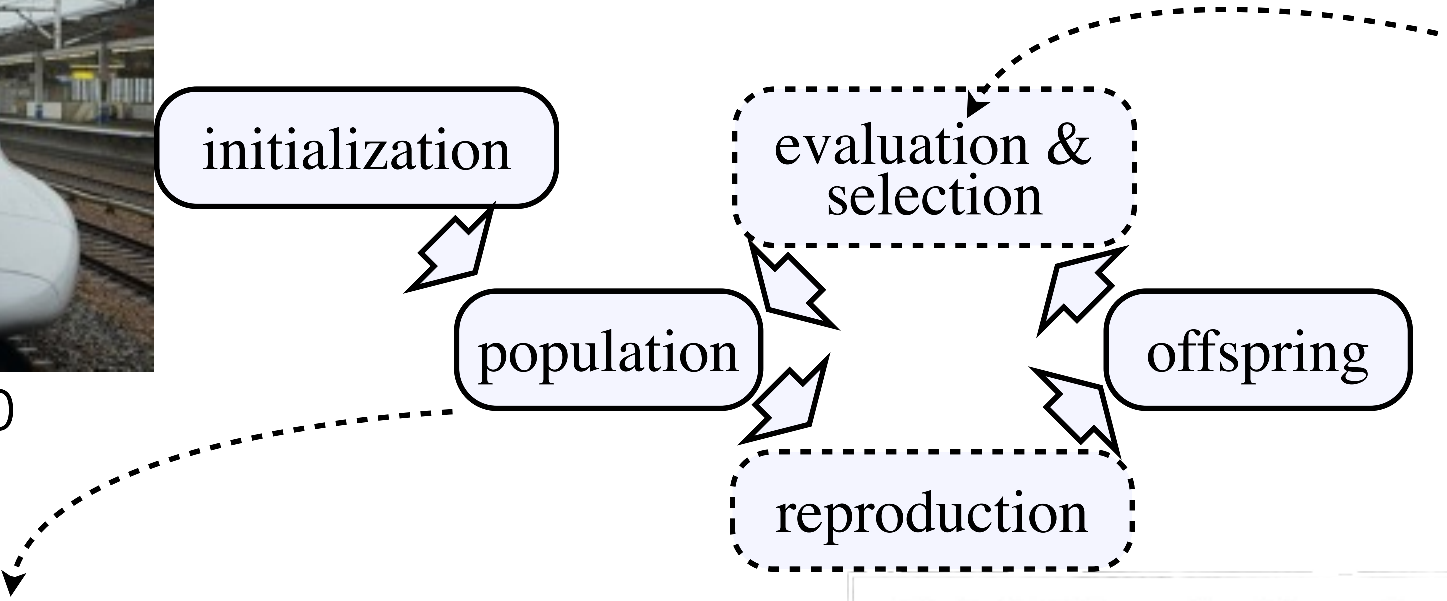
test by simulation

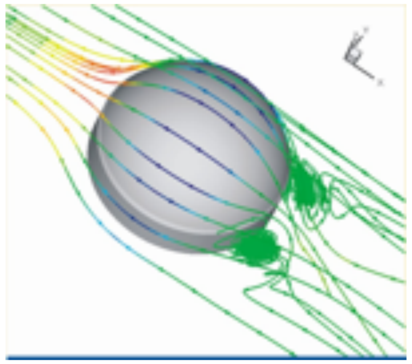
$f(x_i)$

but easy to test a given solution

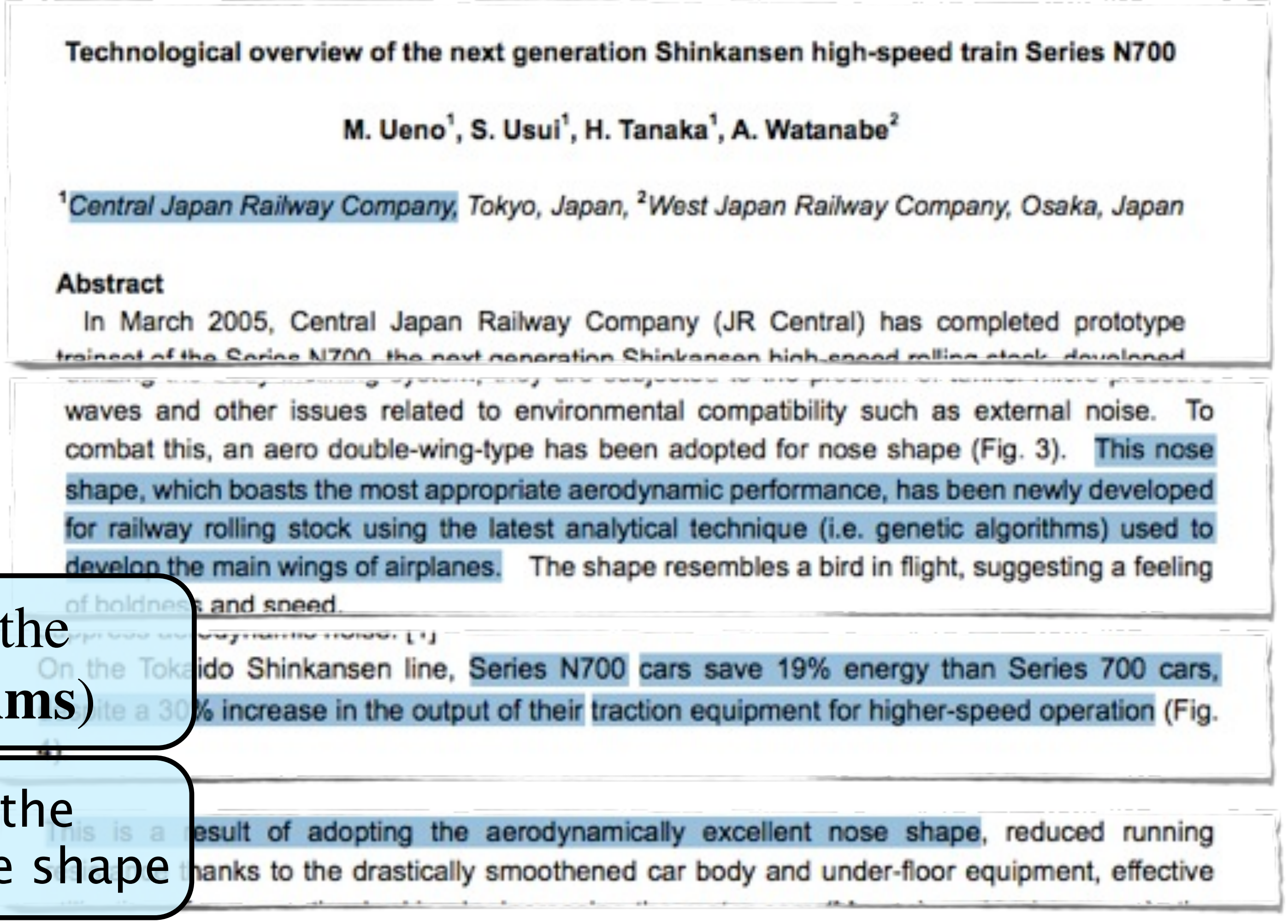
Applications: High-speed train head design

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.

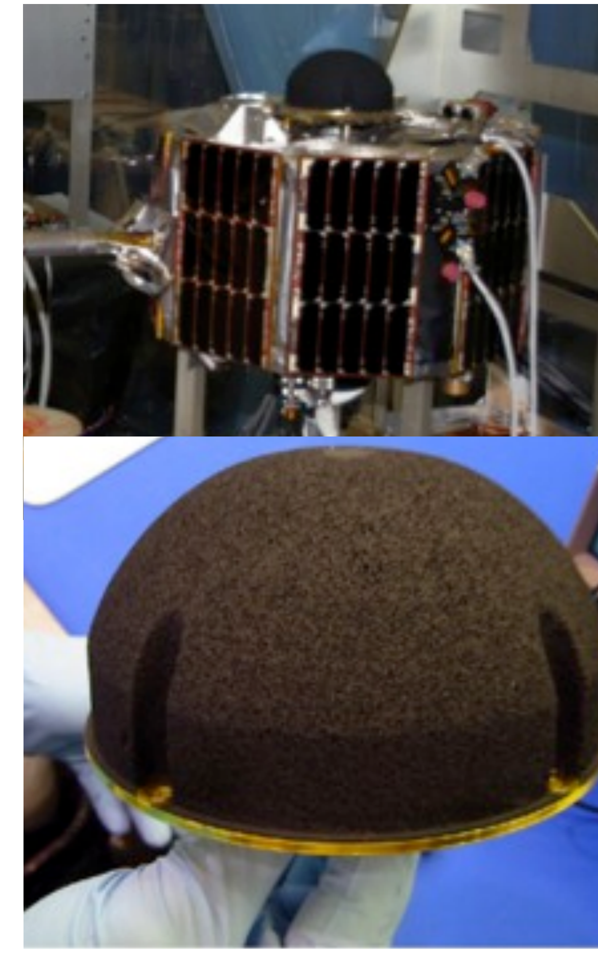
On the Tokaido Shinkansen line, Series N700 cars save 19% energy than Series 700 cars, despite a 30% increase in the output of their traction equipment for higher-speed operation (Fig. 4).

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

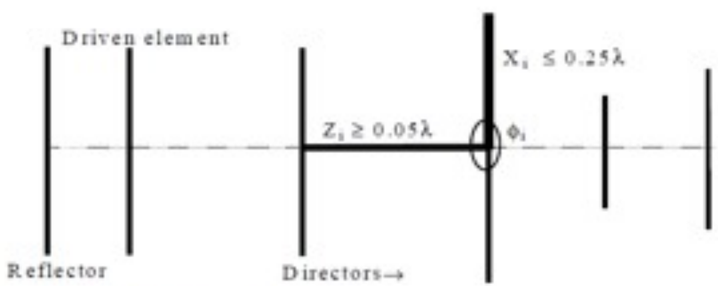
Applications: Antenna design



Problem: optimize the efficiency of the antenna

extremely hard to apply traditional optimization methods

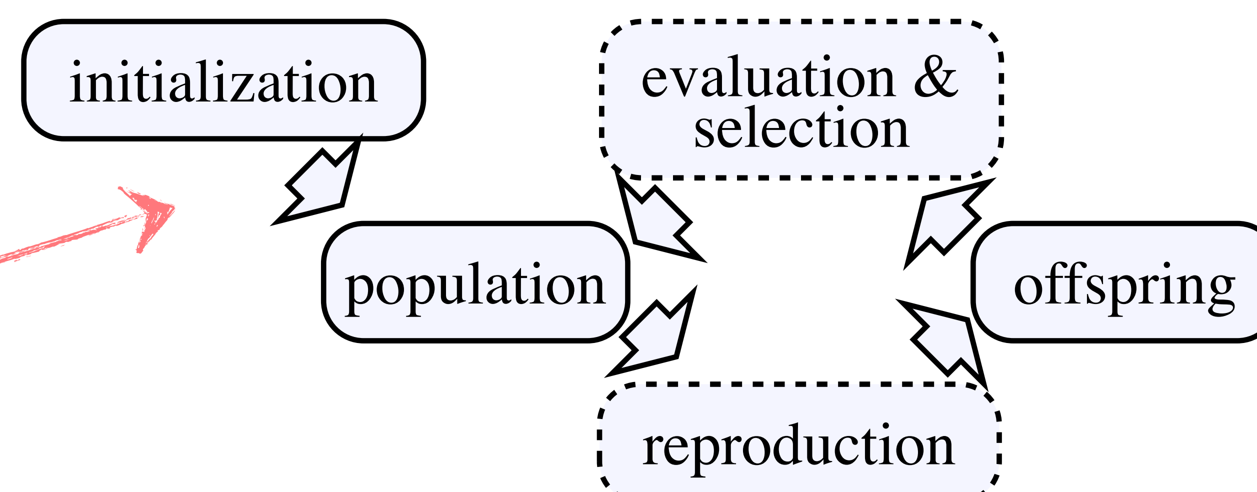
Representation:



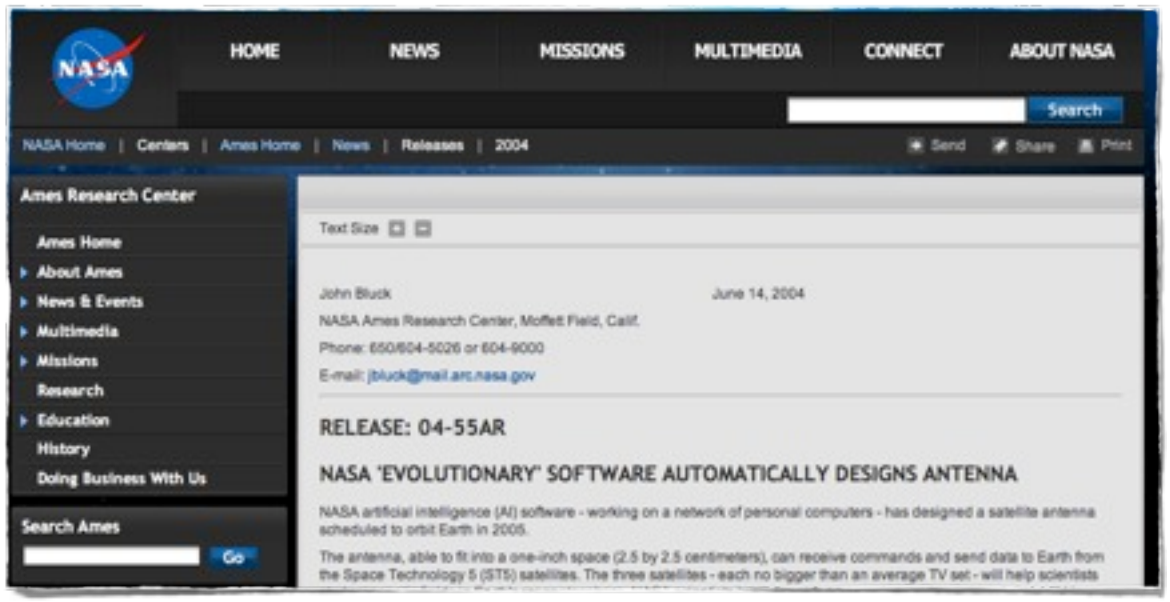
a sequence of operators
forward, rotate-x
rotate-y, rotate-z

Fitness by simulation test

easy to test a given solution
use EAs!



Applications: Antenna design



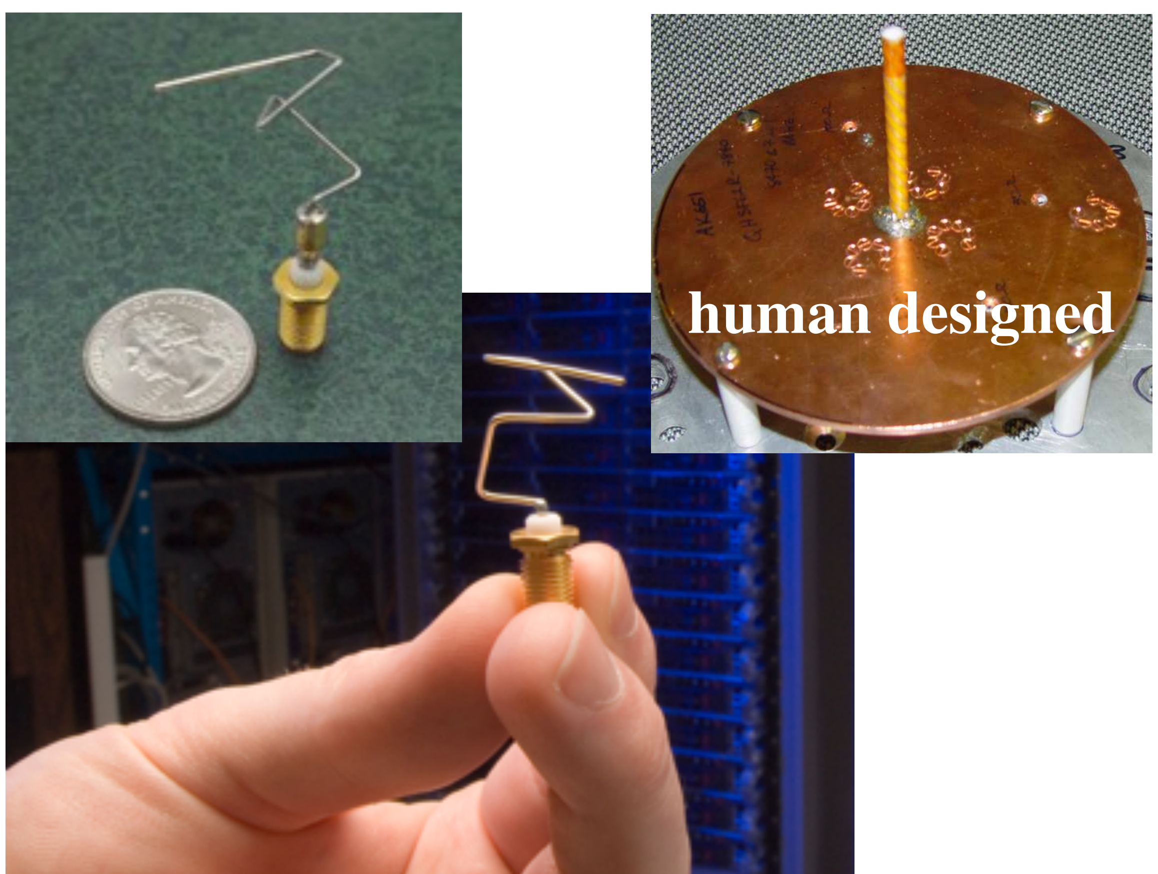
Computer-Automated Evolution of an X-Band Antenna for NASA's Space Technology 5 Mission

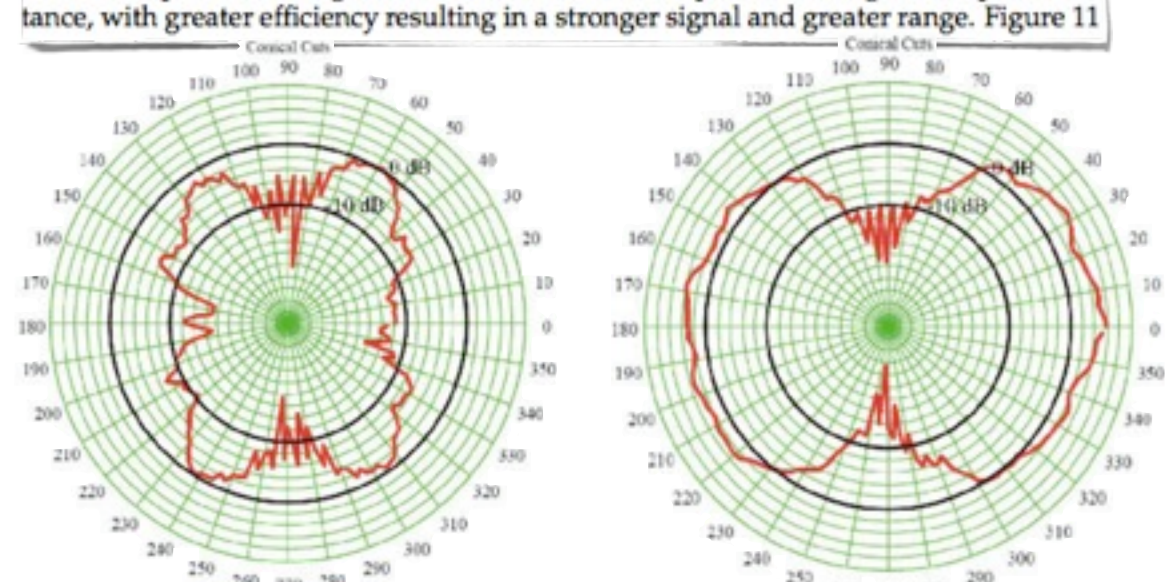
Gregory. S. Hornby Gregory.S.Hornby@nasa.gov
Mail Stop 269-3, University Affiliated Research Center, UC Santa Cruz, Moffett Field, CA, 94035, USA

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

Applications: More designs

[J. R. Koza, et al. What's AI Done for Me Lately? Genetic Programming's Human-Competitive Results. IEEE Intelligent Systems, 18(3): 25-31, 2003.]

Table 1. Human-competitive results produced by genetic programming.

Claimed instance	Basis for claim (criteria number)
1. Creating a better-than-classical quantum algorithm for the Deutsch-Jozsa "early promise" problem[2]	2, 5
2. Creating a better-than-classical quantum algorithm for Grover's database search problem[3]	2, 5
3. Creating a quantum algorithm for the depth-two AND/OR query problem that is better than any previously published result[4,5]	4
4. Creating a quantum algorithm for the depth-one OR query problem that is better than any previously published result[5]	4
5. Creating a protocol for communicating information through a quantum gate that was previously thought not to permit such communication[6]	4
6. Creating a novel variant of quantum dense coding[6]	4
7. Creating soccer-playing program that ranked in the middle of the field of 34 human-written programs in the Robo Cup 1998 competition[7]	8
8. Creating four different algorithms for the transmembrane segment identification problem for proteins[8,9]	2, 5
9. Creating a sorting network for seven items using only 16 steps[9]	1, 4
10. Rediscovering the Campbell ladder topology for lowpass and highpass filters[9]	1, 6
11. Rediscovering the Zobel "*M*-derived half section" and "constant *K*" filter sections[9]	1, 6
12. Rediscovering the Cauer (elliptic) topology for filters[9]	1, 6
13. Automatic decomposition of the problem of synthesizing a crossover filter[9]	1, 6
14. Rediscovering a recognizable voltage gain stage and a Darlington emitter-follower section of an amplifier and other circuits[9]	1, 6
15. Synthesizing 60 and 96 decibel amplifiers[9]	1, 6
16. Synthesizing analog computational circuits for squaring, cubing, square root, cube root, logarithm, and Gaussian functions[9]	1, 4, 7
17. Synthesizing a real-time analog circuit for time-optimal control of a robot[9]	7
18. Synthesizing an electronic thermometer[9]	1, 7
19. Synthesizing a voltage reference circuit[9]	1, 7
20. Creating a cellular automata rule for the majority classification problem that is better than the Gacs-Kurdyumov-Levin (GKL) rule and all other known rules written by humans[9]	4, 5
21. Creating motifs that detect the D–E–A–D box family of proteins and the manganese superoxide dismutase family[9]	3
22. Synthesizing topology for a PID-D2 (proportional, integrative, derivative, and second derivative) controller[10]	1, 6
23. Synthesizing topology for a PID (proportional, integrative, and derivative) controller[10]	1, 6
24. Synthesizing analog circuit equivalent to Philbrick circuit[10]	1, 6
25. Synthesizing NAND circuit[10]	1, 6
26. Simultaneously synthesizing topology, sizing, placement, and routing of analog electrical circuits[10]	7
27. Rediscovering Yagi-Uda antenna[10]	2, 6, 7
28. Creating PID tuning rules that outperform a PID controller using the Ziegler-Nichols and Astrom-Hagglund tuning rules[10]	1, 2, 4, 5, 6, 7
29. Creating three non-PID controllers that outperform PID controllers using the Ziegler-Nichols and Astrom-Hagglund tuning rules[10]	1, 2, 4, 5, 6, 7
30. Rediscovering negative feedback[10]	1, 6
31. Synthesizing a low-voltage balun circuit[10]	1
32. Synthesizing a mixed analog-digital variable capacitor circuit[10]	1
33. Synthesizing a high-current load circuit[10]	1
34. Synthesizing a voltage-current conversion circuit[10]	1
35. Synthesizing a cubic signal generator[10]	1
36. Synthesizing a tunable integrated active filter[10]	1

e.g.: design low-voltage balun circuit

"The best-of-run evolved circuit (see Figure 1) is roughly a <u>fourfold improvement</u> over the patented circuit in terms of our fitness measure. The evolved circuit is superior both in terms of its frequency response and harmonic distortion."

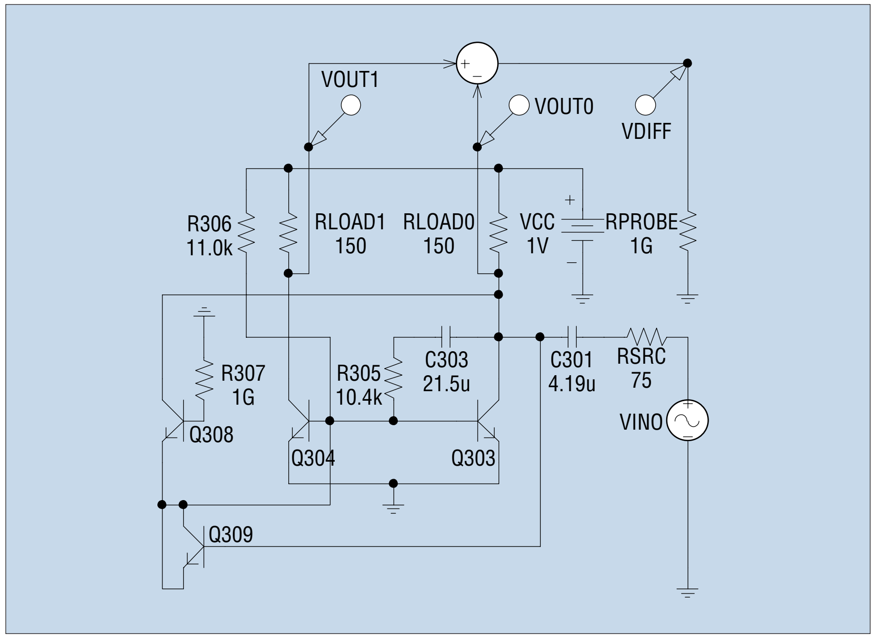
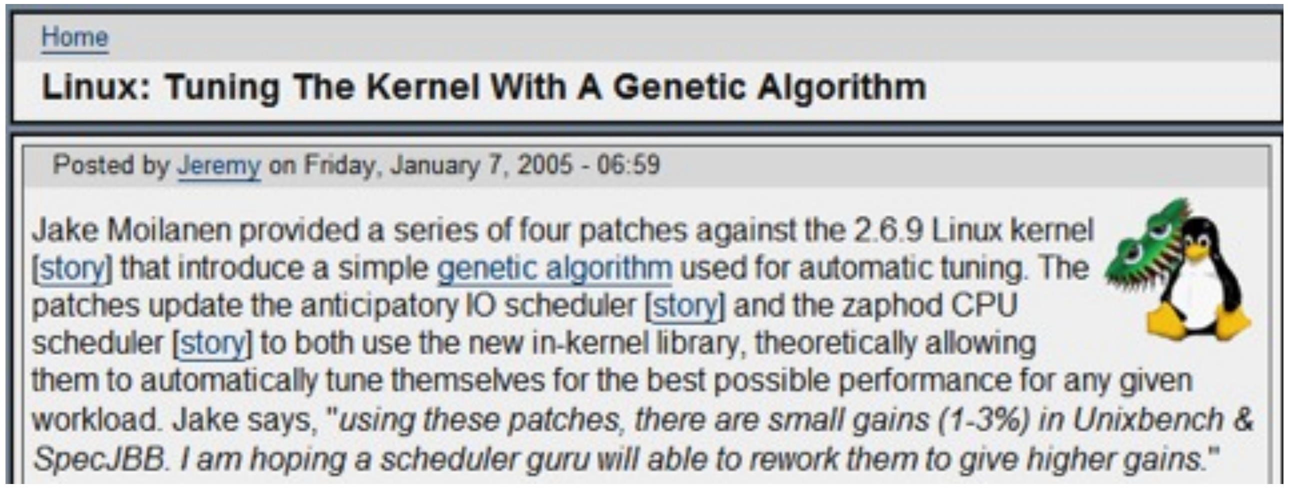


Figure 1. Genetically evolved low-voltage balun (balance/unbalance) circuit.

And more ...

optimizing operating systems:



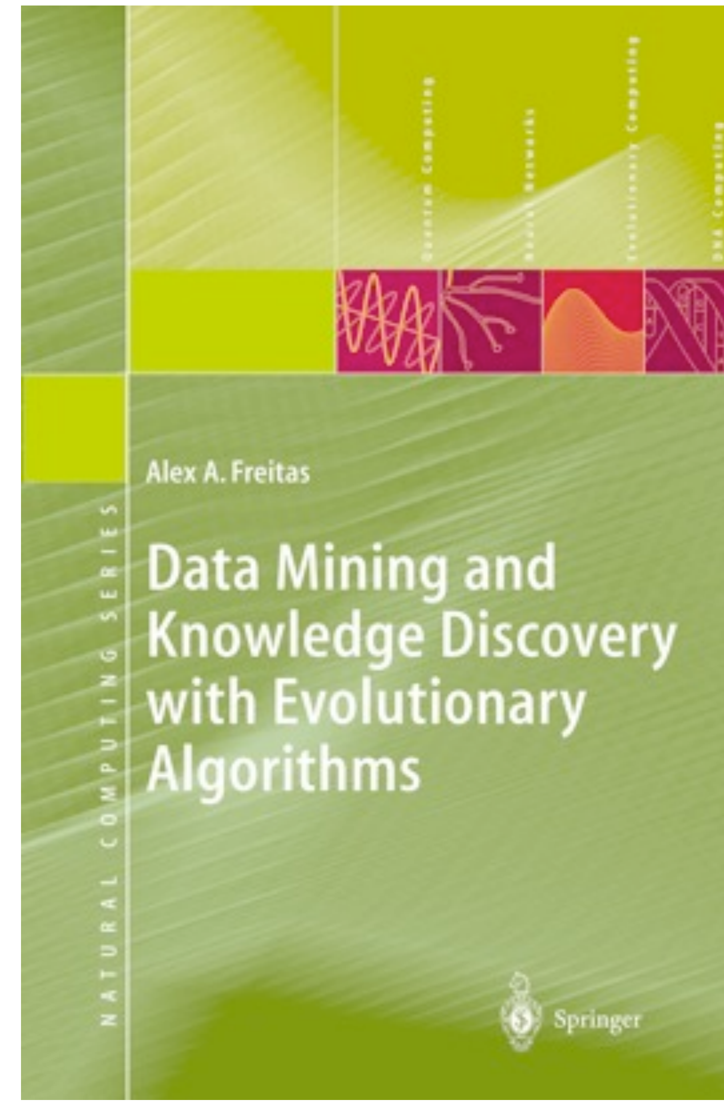
Home

Linux: Tuning The Kernel With A Genetic Algorithm

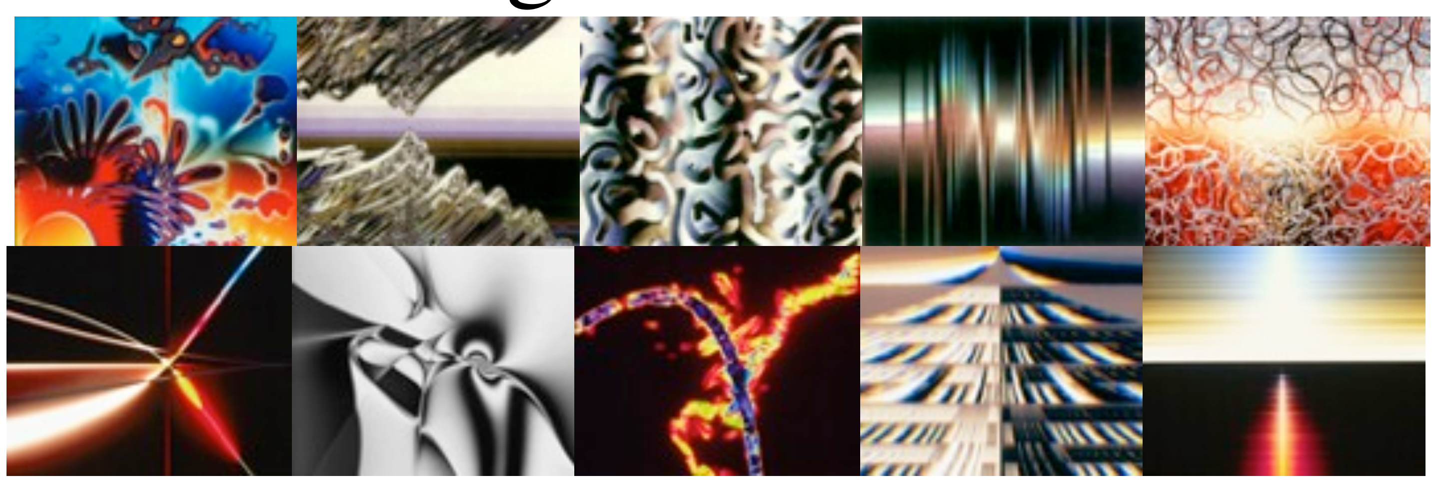
Posted by Jeremy on Friday, January 7, 2005 - 06:59

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.*"

data mining:



interactive art design:



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