Last class

- Parent selection
- Survival selection
- Population diversity

Heuristic Search and Evolutionary Algorithms

Lecture 8: Popular Variants of Evolutionary Algorithms

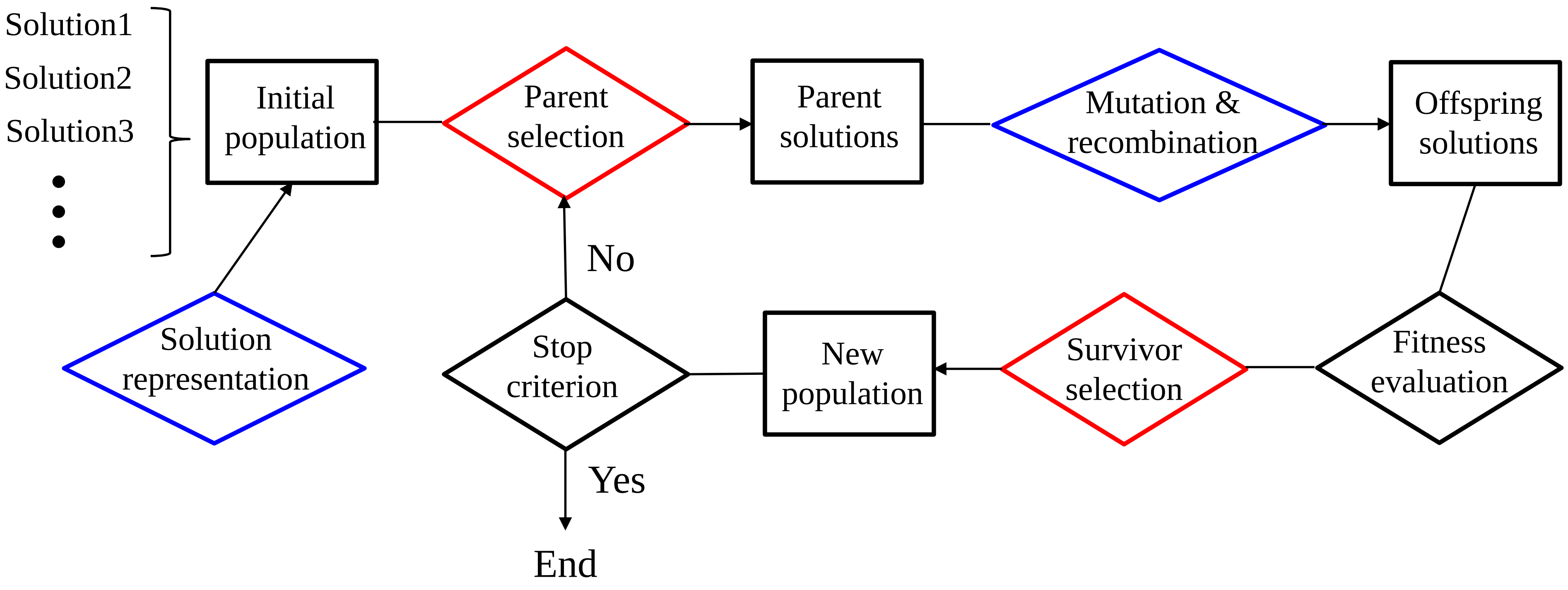
Chao Qian （钱超）

Associate Professor, Nanjing University, China

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

Evolutionary algorithms

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



There have been many popular variants of EAs

Genetic algorithms

J. H. Holland
1929-2015

Genetic Algorithms (GA)

Typically applied to optimization in discrete domains

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

University of Michigan

Simple GA (SGA)

Representation	Binary representation
Recombination	One-point crossover
Mutation	Bit-wise mutation
Parent selection	Fitness proportional selection – implemented by Roulette Wheel
Survivor selection	Generational, i.e, age-based replacement with $\lambda = \mu$

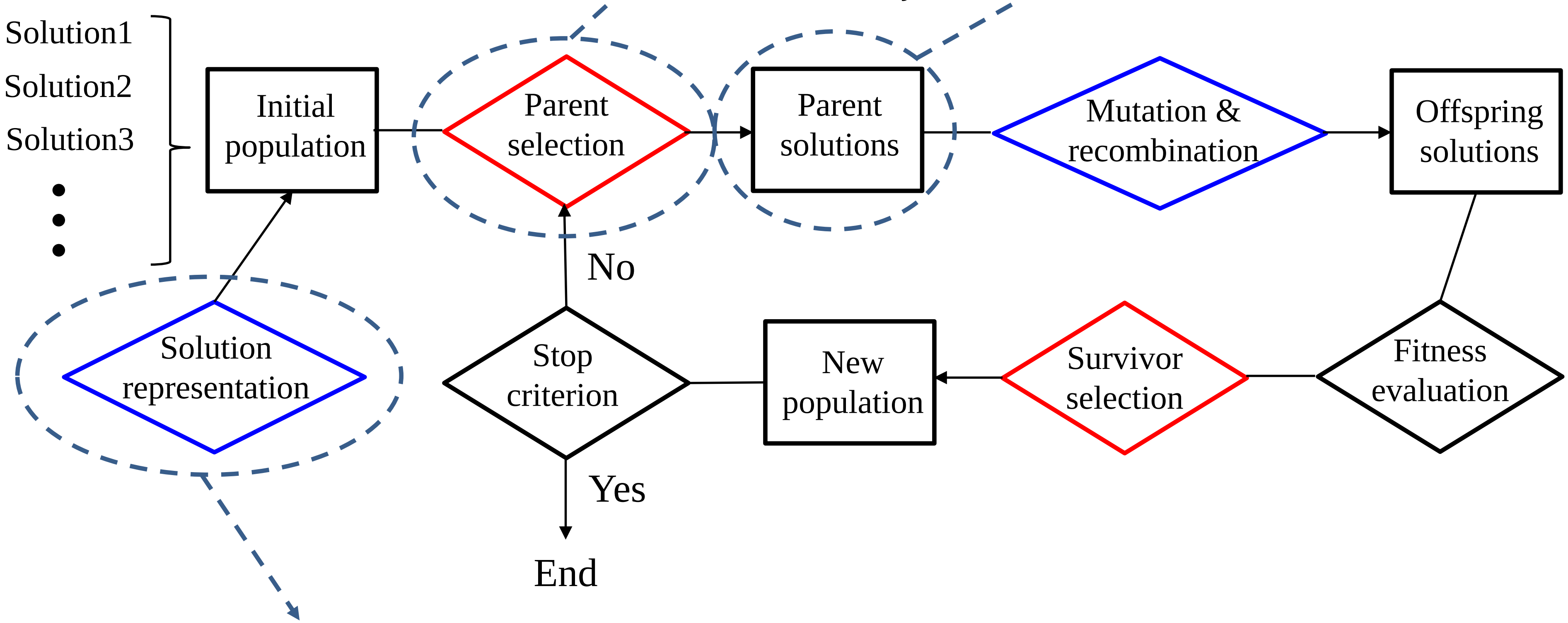
Genetic algorithms

SGA

the probability of selecting the i-th individual is

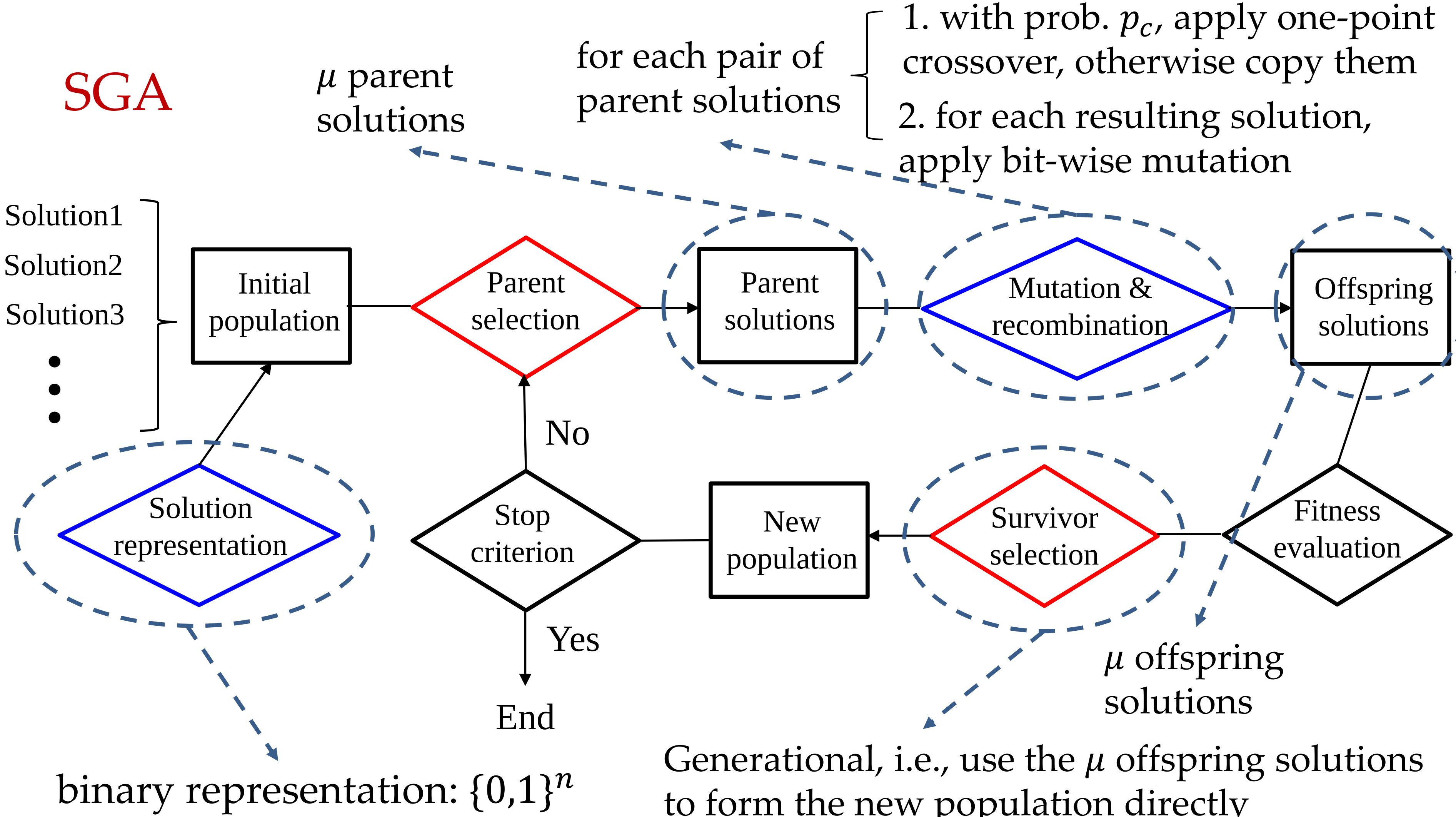
$$P_{FPS}(i) = f_i / \sum_{j=1}^{\mu} f_j$$

μ parent solutions



binary representation: $\{0,1\}^n$

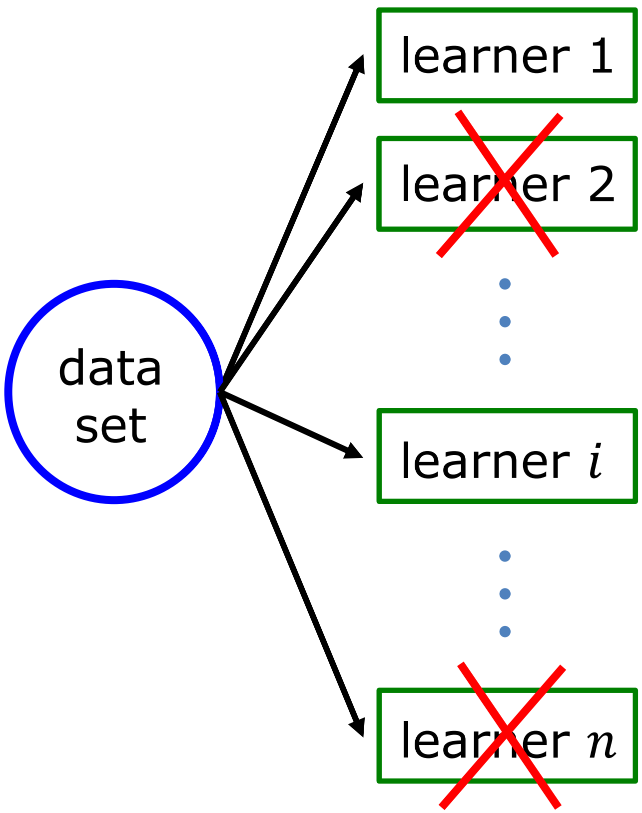
Genetic algorithms



Genetic algorithms: Application

Ensemble learning [Zhou, 2012]

- better performance than a single learner



Selective ensemble (ensemble pruning) [Zhou, 2012]

- better performance than the complete ensemble
- reduce storage and improve efficiency

Two goals

- maximize the generalization performance
- minimize the number of selected learners

Genetic algorithms: Application

PEP [Qian, Yu and Zhou, AAAI'15]: apply GA with uniform parent selection, bit-wise mutation and fitness-based survivor selection to solve the selective ensemble problem

binary representation: $\boldsymbol{x} \in \{0,1\}^n$

⇩

a subset of base learners

$x_i = 1$: the i-th base learner is selected

$x_i = 0$: the i-th base learner is not selected

Genetic algorithms: Application

Pruning bagging base learners with size 100

baseline methods ordering-based methods

		Test Error						
Data set	PEP	Bagging	BI	RE	Kappa	CP	MD	DREP
australian	.144±.020	**.143±.017**	.152±.023•	.144±.020	**.143±.021**	.145±.022	.148±.022	.144±.019
breast-cancer	**.275±.041**	.279±.037	.298±.044•	.277±.031	.287±.037	.282±.043	.295±.044•	**.275±.036**
disorders	**.304±.039**	.327±.047•	.365±.047•	.320±.044•	.326±.042•	.306±.039	.337±.035•	.316±.045
heart-statlog	.197±.037							
house-votes	.045±.019							
ionosphere	.088±.021							
kr-vs-kp	**.010±.003**							
letter-ah	.013±.005							
letter-br	**.046±.008**							
letter-oq	.043±.009	.049±.012•	.078±.017•	.046±.011	.042±.011	.042±.010	.046±.011	**.041±.010**
optdigits	**.035±.006**	.038±.007•	.095±.008•	.036±.006	**.035±.005**	.036±.005	.037±.006•	**.035±.006**
satimage-12v57	**.028±.004**							
satimage-2v5	**.021±.007**							
sick	**.015±.003**							
sonar	**.248±.056**							
spambase	**.065±.006**							
tic-tac-toe	.131±.027	.164±.028•	.212±.028•	.135±.026	.132±.023	.132±.026	.145±.022•	**.129±.026**
vehicle-bo-vs	**.224±.023**	.228±.026	.257±.025•	.226±.022	.233±.024•	.234±.024•	.244±.024•	.234±.026•
vehicle-b-v	**.018±.011**	.027±.014•	.024±.013•	.020±.011	.019±.012	.020±.011	.021±.011•	.019±.013
vote	.044±.018	.047±.018	.046±.016	.044±.017	**.041±.016**	.043±.016	.045±.014	.043±.019
count of the best	12	2	0	2	7	1	0	5
PEP: count of direct win		**17**	**20**	**15.5**	12.5	**17**	**20**	12.5

PEP achieves the smallest error on 60% (12/20) of the data sets, while other methods perform the best on at most 35% (7/20) data

PEP is better than any other method on more than 60% (12.5/20) data sets

PEP is never significantly worse

Genetic algorithms: Application

ordering-based methods

	Ensemble Size					
Data set	PEP	RE	Kappa	CP	MD	DREP
australian	10.6±4.2	12.5±6.0	14.7±12.6	11.0±9.7	**8.5±14.8**	11.7±4.7
breast-cancer	8.4±3.5	8.7±3.6	26.1±21.7•	8.8±12.3	**7.8±15.2**	9.2±3.7
disorders	14.7±4.2	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
heart-statlog	**9.3±2.3**	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
house-votes	**2.9±1.7**	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
ionosphere	**5.2±2.2**	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
kr-vs-kp	**4.2±1.8**	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
letter-ah	**5.0±1.9**	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
letter-br	**10.9±2.6**	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
letter-oq	**12.0±3.7**	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
optdigits	22.7±3.1	25.0±9.3	25.2±8.1	**21.4±7.5**	46.8±23.9•	25.0±8.0
satimage-12v57	**17.1±5.0**	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
satimage-2v5	**5.7±1.7**	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
sick	**6.9±2.8**	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
sonar	11.4±4.2	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
spambase	17.5±4.5	18.5±5.0	20.0±8.1	19.0±9.9	28.8±17.0•	**16.7±4.6**
tic-tac-toe	14.5±3.8	16.1±5.4	17.4±6.5	15.4±6.3	28.0±22.6•	**13.6±3.4**
vehicle-bo-vs	16.5±4.5	15.7±5.7	16.5±8.2	**11.2±5.7∘**	21.6±20.4	13.2±5.0∘
vehicle-b-v	**2.8±1.1**	3.4±2.1	4.5±1.6•	5.3±7.4	**2.8±3.8**	4.0±3.9
vote	**2.7±1.1**	3.2±2.7	5.1±2.6•	5.4±5.2•	6.0±9.8	3.9±2.5•
count of the best	12	2	0	2	3	3
PEP: count of direct win		17	19.5	18	17.5	16

PEP achieves the smallest size on 60% (12/20) of the data sets, while other methods achieve the smallest size on at most 15% (3/20) data

PEP is better than any other method on at least 80% (16/20) data sets

PEP is never significantly worse, except two losses on vehicle-bo-vs

Evolutionary strategies

I. Rechenberg
1934-

Evolutionary Strategies (ES)

Typically applied to optimization in continuous domains

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

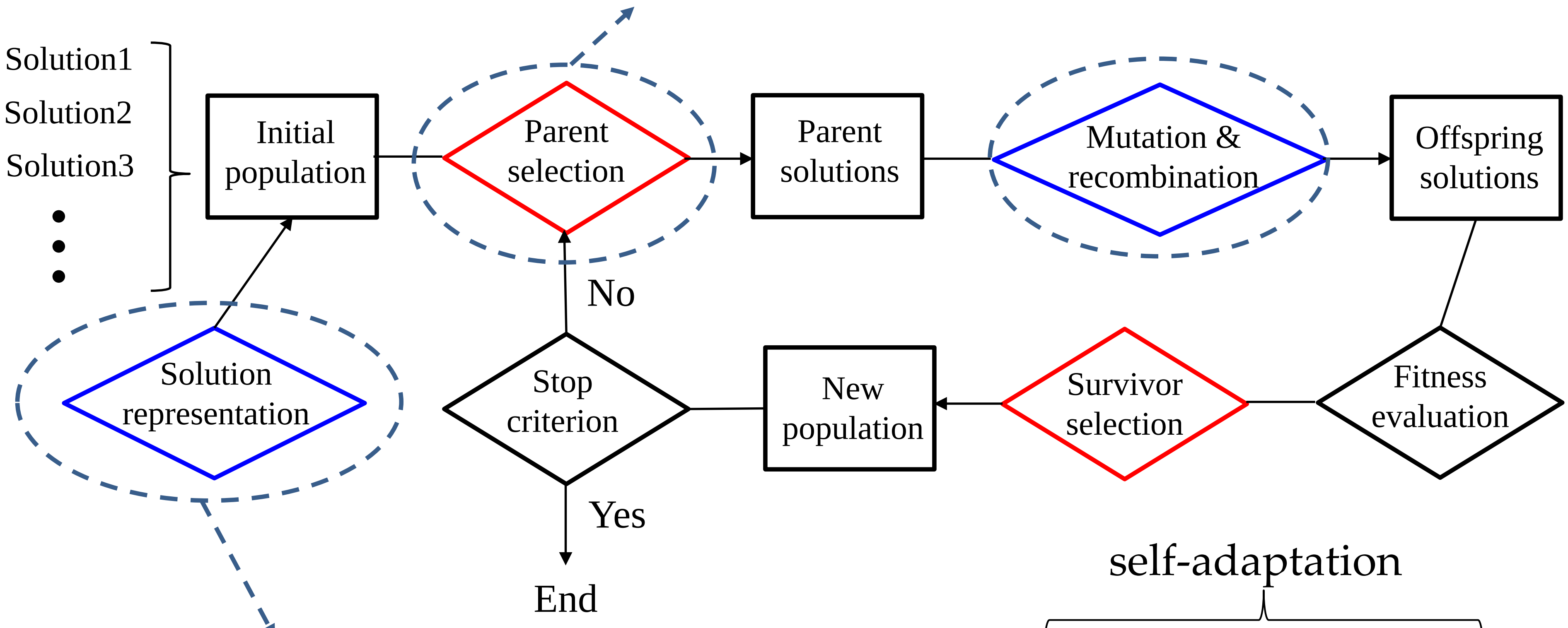
Technical University of Berlin

Representation	Real-valued representation
Recombination	Discrete or arithmetic
Mutation	Gaussian perturbation
Parent selection	Uniform random
Survivor selection	Fitness-based replacement by (μ, λ) or $(\mu + \lambda)$
Speciality	Self-adaptation of mutation step sizes

Evolutionary strategies

ES

the probability of selecting each individual is $1/\mu$



Real-valued representation: R^n $\Rightarrow$ $(x_1, \ldots, x_n, \overbrace{\sigma_1, \ldots, \sigma_n, \alpha_1, \ldots, \alpha_{n(n-1)/2}}^{\text{self-adaptation}})$

Evolutionary strategies

Local recombination:

Select two parents uniformly at random

$$\left(x_1, \ldots, x_n, \sigma_1, \ldots, \sigma_n, \alpha_1, \ldots, \alpha_{n(n-1)/2}\right)$$

$$\left(y_1, \ldots, y_n, \sigma'_1, \ldots, \sigma'_n, \alpha'_1, \ldots, \alpha'_{n(n-1)/2}\right)$$

$$\left(z_1, \ldots, z_n, \sigma''_1, \ldots, \sigma''_n, \alpha''_1, \ldots, \alpha''_{n(n-1)/2}\right)$$

Discrete: z_i is chosen from x_i and y_i uniformly at random

Arithmetic: $\sigma''_i = \sigma_i/2 + \sigma'_i/2 \quad \alpha''_i = \alpha_i/2 + \alpha'_i/2$

Global recombination: the two parents are selected uniformly at random for each position

Evolutionary strategies

Local recombination:

Select two parents uniformly at random $\Rightarrow$

$$\left(x_1, \ldots, x_n, \sigma_1, \ldots, \sigma_n, \alpha_1, \ldots, \alpha_{n(n-1)/2}\right)$$

$$\left(y_1, \ldots, y_n, \sigma'_1, \ldots, \sigma'_n, \alpha'_1, \ldots, \alpha'_{n(n-1)/2}\right)$$

$$\left(z_1, \ldots, z_n, \sigma''_1, \ldots, \sigma''_n, \alpha''_1, \ldots, \alpha''_{n(n-1)/2}\right)$$

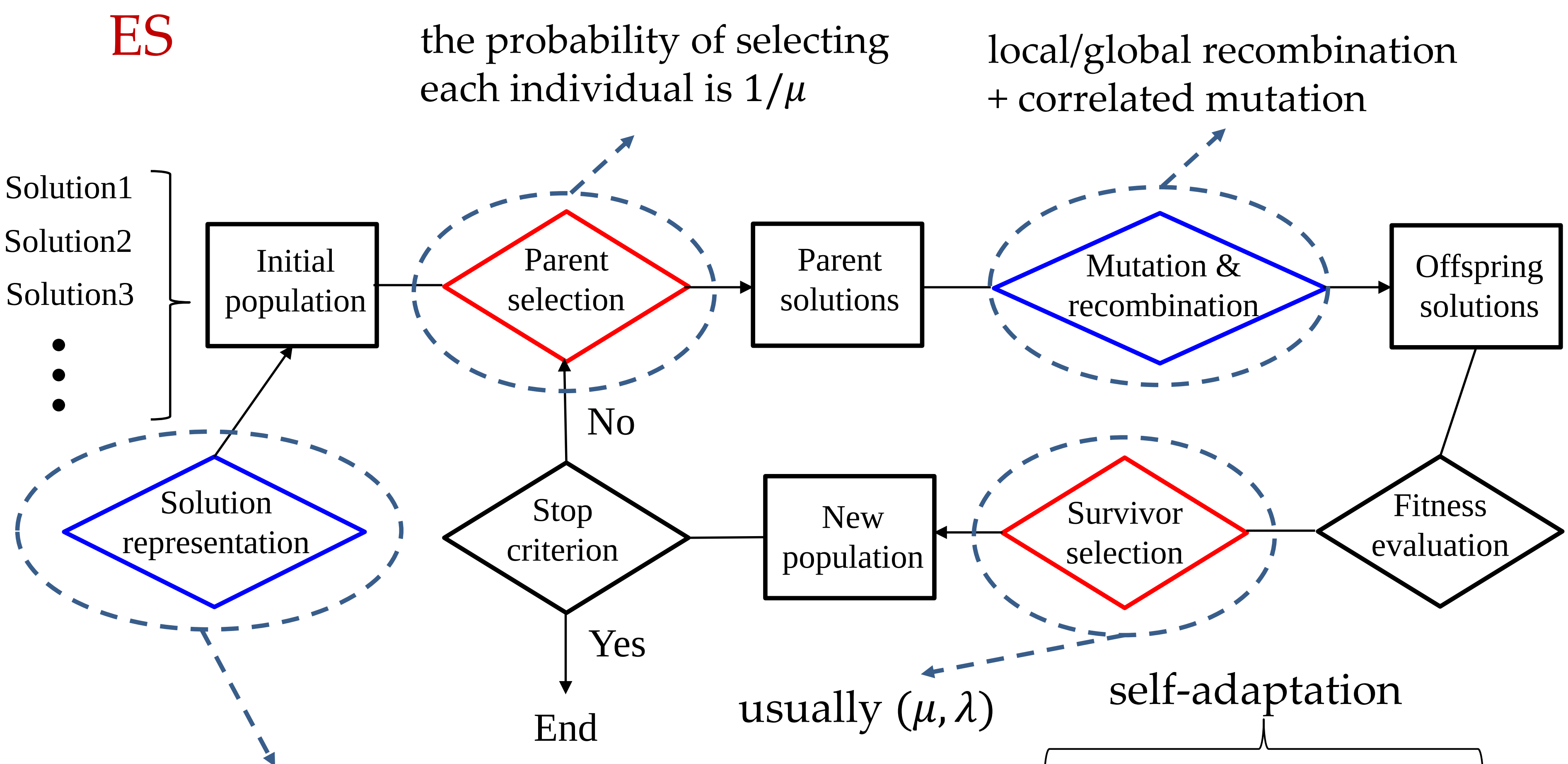
Correlated mutation:

$$\left(w_1, \ldots, w_n, \delta_1, \ldots, \delta_n, \beta_1, \ldots, \beta_{n(n-1)/2}\right)$$

Self-adaptation $\quad \delta_i = \sigma''_i \cdot e^{\tau' \cdot N(0,1) + \tau \cdot N_i(0,1)} \quad \beta_j = \alpha''_j + \beta \cdot N_j(0,1)$

$$\boldsymbol{w} = \boldsymbol{z} + N(\mathbf{0}, \mathbf{C}')$$

Evolutionary strategies



Real-valued representation: R^n ⇨ $\left(x_1, \dots, x_n, \overbrace{\sigma_1, \dots, \sigma_n, \alpha_1, \dots, \alpha_{n(n-1)/2}}^{\text{self-adaptation}}\right)$

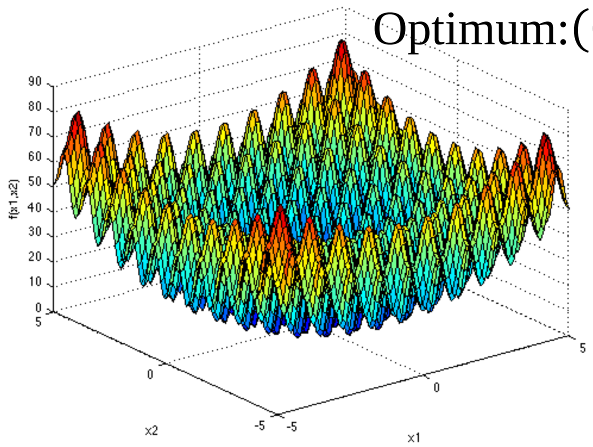
Evolutionary strategies: Application

Optimum:(0,0), 0

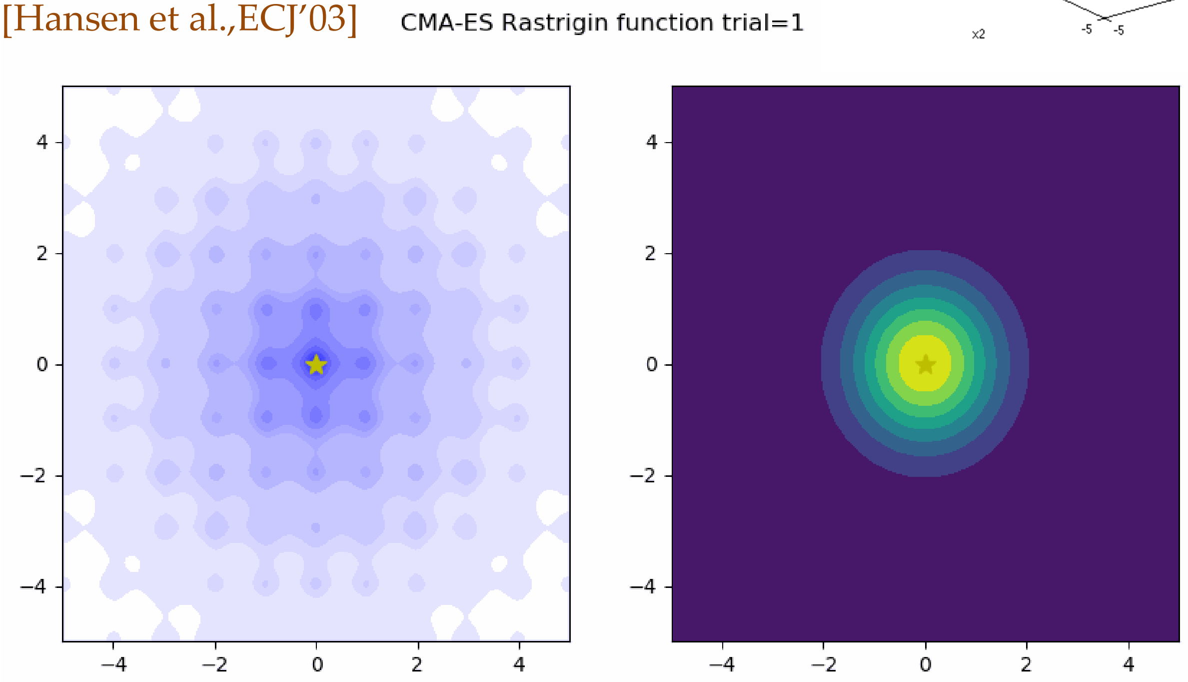
Rastrigin function

$d = 2$

$$f(\boldsymbol{x}) = 10d + \sum_{i=1}^{d} [x_i^2 - 10\cos(2\pi x_i)]$$



CMA-ES [Hansen et al.,ECJ'03]



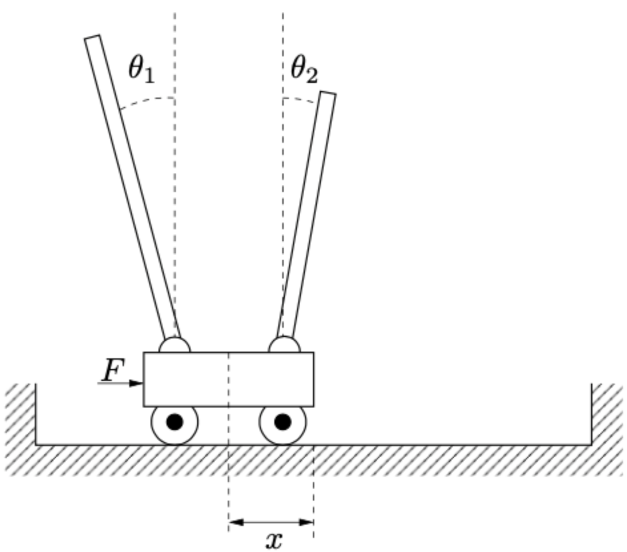
Evolutionary strategies: Application

Reinforcement learning

- learn how to take actions in an environment in order to maximize the cumulative reward

Example: double pole with velocities problem

Goal: Learn an optimal policy to keep the angles of the poles in the range $[-36^\circ, 36^\circ]$ for 10^5 time steps, where each step corresponds to 0.02s



velocity

State: vector $(x, \boxed{\dot{x}}, \theta_1, \boxed{\dot{\theta}_1}, \theta_2, \boxed{\dot{\theta}_2})$

Action: exert forces either left or right on the cart

Reward: -1 when balancing fails (any of the poles out of range $[-36^\circ, 36^\circ]$)

Finite length track

Evolutionary strategies: Application

[Igel, CEC'03] uses CMA-ES with average arithmetic recombination, Gaussian perturbation with self-adaptation, and $(\mu + \lambda)$ survivor selection to solve the double pole with velocities problem

real-valued representation: $\boldsymbol{x} \in \mathrm{R}^n$

$\Downarrow$

neural network weights

$\Downarrow$

a policy π

Evolutionary strategies: Application

CMA-ES	n_{hidden}	bias	n_{weights}	evaluations	failures
(3/3, 13)	4	no	28	**884** / 3142	0 / 1
(3/3, 13)	4	yes	33	2929 / 25853	0 / 12
(3/3, 13)	6	no	42	**895**	0
(3/3, 13)	6	yes	49	2672 / 13464	0 / 5
(4/4, 16)	8	no	56	1119	0
(4/4, 16)	8	yes	65	2493	0
(4/4, 16)	10	no	70	1003	0
(4/4, 16)	10	yes	81	2494	0
(4/4, 16)	12	no	84	1143	0
(4/4, 16)	12	yes	97	2216	0
(4/4, 16)	14	no	98	1021	0
(4/4, 16)	14	yes	113	2391	0
(4/4, 16)	16	no	112	967	0
(4/4, 16)	16	yes	129	2146	0

CMA-ES can find an optimal policy even with a small population size

method	evaluations	population size
NE (Wieland, 1991)	$\approx$ 307200	2048
EP (Saravanan and Fogel, 1995)	$\approx$ 80000	100
SANE (Moriarty and Miikulainen, 1996)	12600	200
ESP (Gomez and Miikulainen, 1999)	3800	200
NEAT (Stanley and Miikkulainen, 2002)	3600	150

CMA-ES is almost four times faster than the best previous algorithm

Evolutionary programming

L. J. Fogel
1928-2007

Evolutionary Programming (EP)

Originally 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

Now typically applied to optimization in continuous domains, and almost merged with ES

Representation	Real-valued representation
Recombination	None
Mutation	Gaussian perturbation
Parent selection	Deterministic (each parent generates one offspring)
Survivor selection	Round-robin tournament
Speciality	Self-adaptation of mutation step sizes

difference

Genetic programming

J. R. Koza
1944-

Genetic Programming (GP)

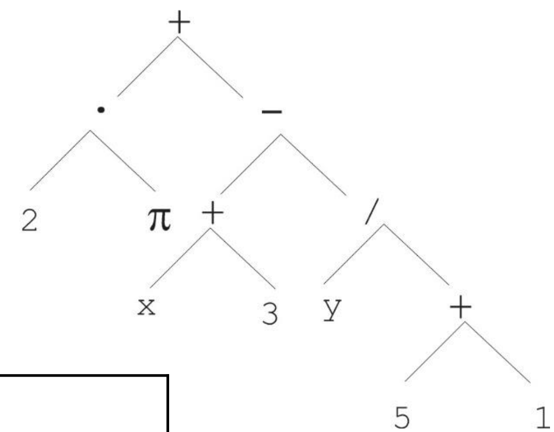
Typically for optimizing computer programs

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

Stanford University

Arithmetic formula:

$$2 \cdot \pi + \left((x+3) - \frac{y}{5+1} \right)$$

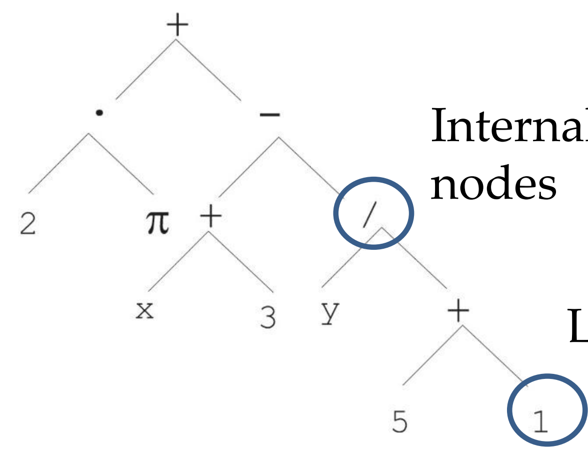


Representation	Tree representation
Recombination	Exchange of subtrees
Mutation	Random change in trees
Parent selection	Fitness proportional
Survivor selection	Generational replacement

Genetic programming

Arithmetic formula:

$$2 \cdot \pi + \left((x+3) - \frac{y}{5+1} \right)$$



Tree representation

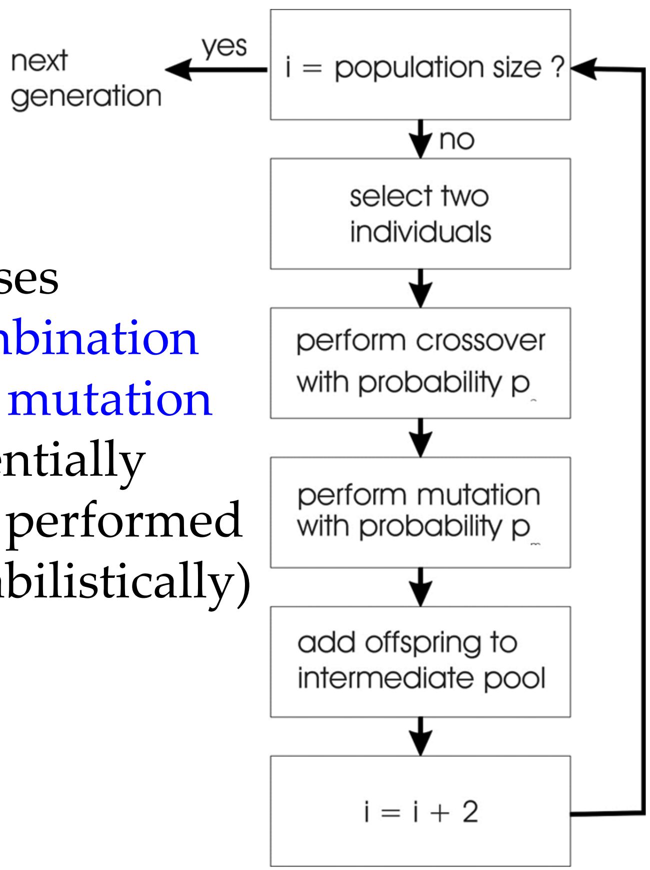
Internal nodes → Function set F

Leaves → Terminal set T

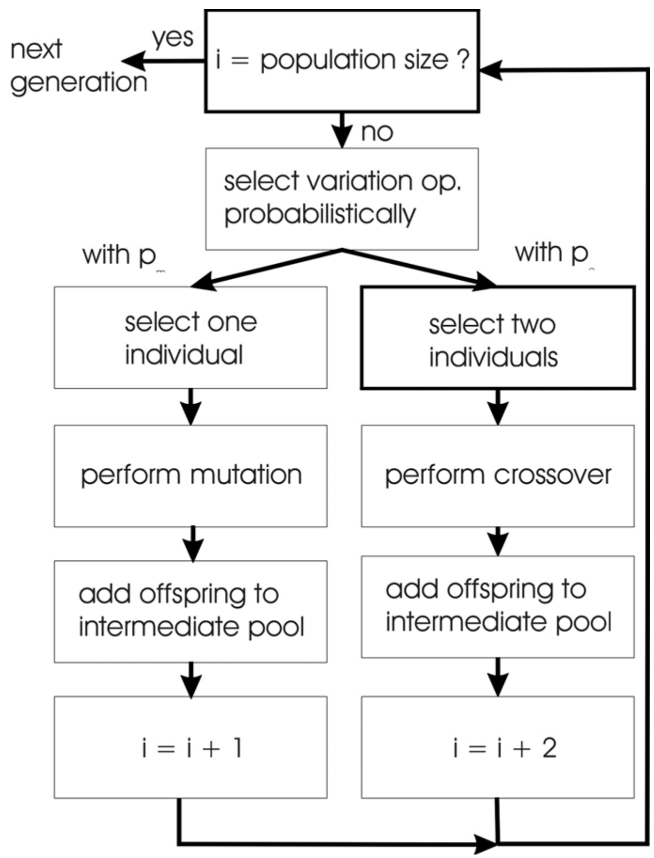
- Initial tree construction (maximum initial depth d_{max})
  - ➢ Full method (each branch has depth $= d_{max}$):
    - nodes at depth $< d_{max}$ are randomly chosen from F
    - nodes at depth d_{max} are randomly chosen from T
  - ➢ Grow method (each branch has depth $\leq d_{max}$):
    - nodes at depth $< d_{max}$ are randomly chosen from $F \cup T$
    - nodes at depth d_{max} are randomly chosen from T

Genetic programming

GA uses recombination AND mutation sequentially (each performed probabilistically)



GA flowchart



GP flowchart

GP uses recombination OR mutation (chosen probabilistically)

Genetic programming

- Bloat: average tree sizes tend to grow over time
  - Prohibiting variation operators that would generate "too big" offspring
  - Parsimony pressure: penalty for being oversized

- Parent selection
  - Typically fitness proportional selection
  - Over-selection for very large population sizes

    - rank population by fitness and divide it into two groups: group 1: best x% of population, group 2: other $(100-x)$%

    - 80% of selection chooses from group 1, 20% from group 2

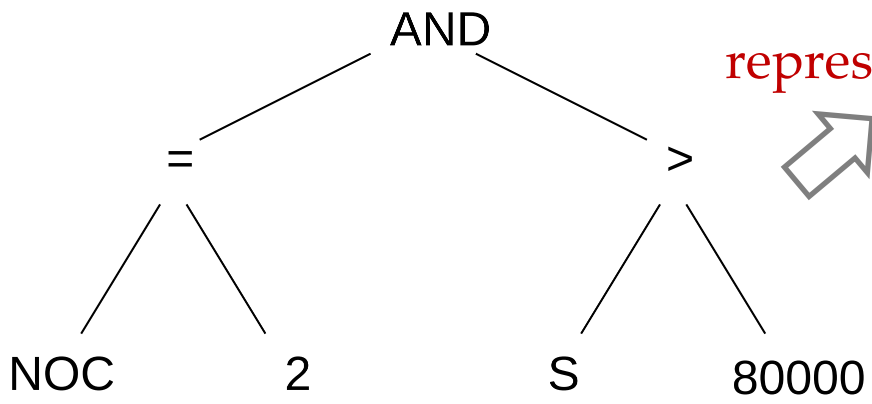
    - for pop. size = $1000, 2000, 4000, 8000$, $x = 32\%, 16\%, 8\%, 4\%$

Selection pressure increases with the population size

Genetic programming: Application

Task: learn a rule to distinguish good from bad loan applicants

ID	No of children	Salary	Marital status	Good?
ID-1	2	45000	Married	0
ID-2	0	30000	Single	1
ID-3	1	40000	Divorced	1
…				



represent

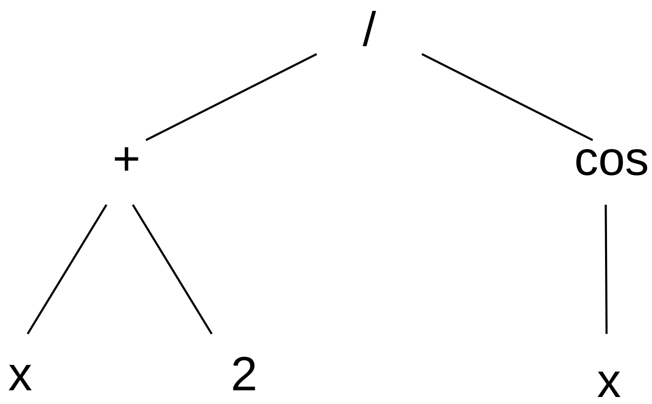
IF (NOC = 2) AND (S > 80000) THEN good ELSE bad

Fitness: percentage of correctly classified cases

Genetic programming: Application

Task: find a function $f(x)$ to fit the observed data

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$$



represent

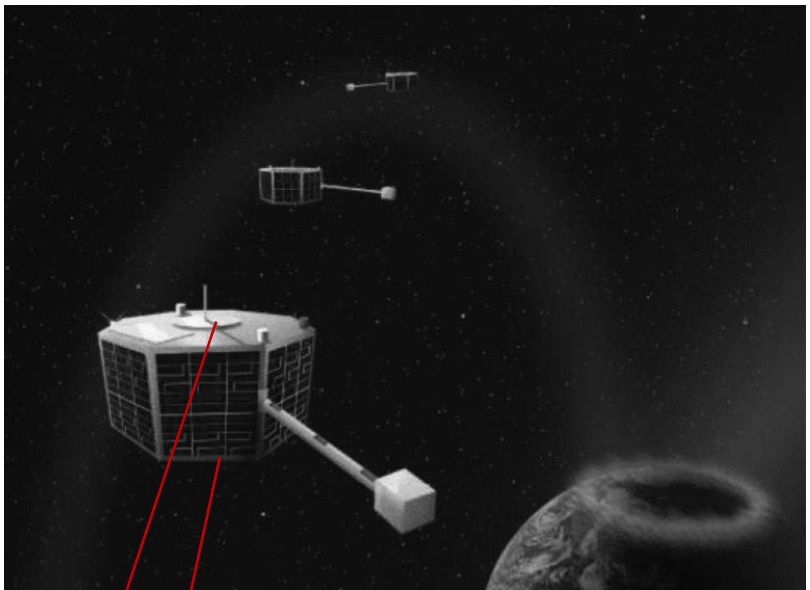
$$f(x) = (x+2)/\cos x$$

Fitness: the error

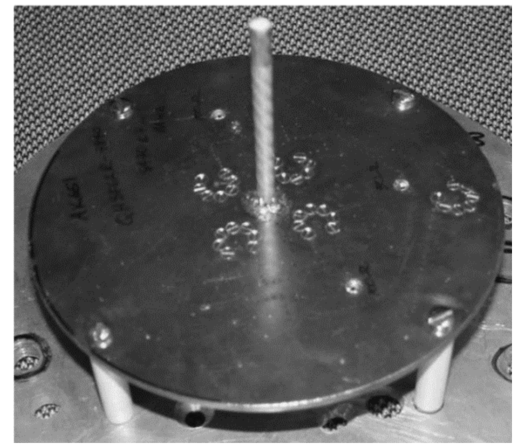
$$\sum_{i=1}^{n}(f(x_i) - y_i)^2$$

Genetic programming: Application

Task: antenna design in NASA's Space Technology 5 (ST5) mission

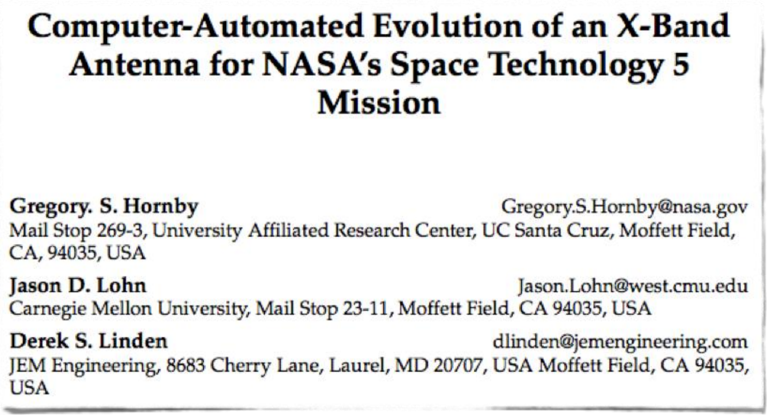


Two antennas centered on the top and bottom of each spacecraft



Quadrifilar helical antenna designed by human experts

Genetic programming: Application



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

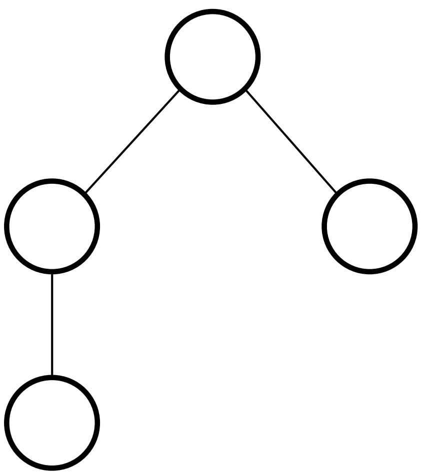
Jason D. Lohn Jason.Lohn@west.cmu.edu
Carnegie Mellon University, Mail Stop 23-11, Moffett Field, CA 94035, USA

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

use GP to design antenna automatically

Fitness: efficiency and gain evaluated by simulation

tree representation



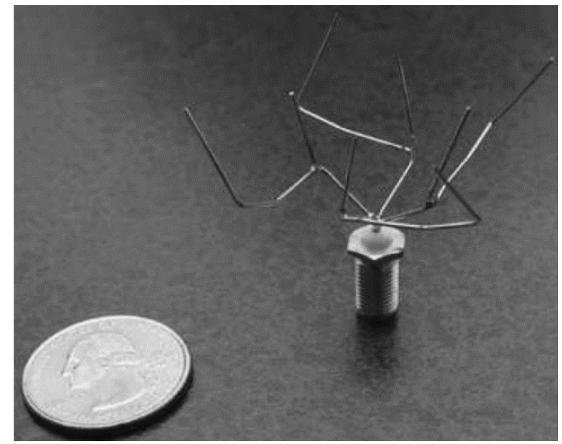
- forward(length, radius)
- rotate-x(angle)
- rotate-y(angle)
- rotate-z(angle)

Execute the operators by preorder traversal

an antenna

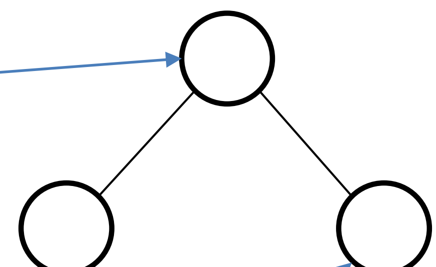
Genetic programming: Application

```
rotate-z(1.984442) 1 [ rotate-x(2.251165) 1 [
rotate-x(0.062240) 1 [ rotate-x(0.083665) 1 [
rotate-y(-2.449035) 1 [ rotate-z(-0.894357) 1 [
rotate-y(-2.057702) 1 [ rotate-y(0.661755) 1 [
rotate-x(0.740703) 1 [ rotate-y(2.057436) 1 [
forward(0.013292,0.000283) 2 [ rotate-z(-1.796822) 1 [
rotate-x(-1.651348) 1 [ rotate-y(-2.940880) 1 [
rotate-x(0.095209) 1 [ rotate-z(1.248723) 1 [
forward(0.003815,0.000363) 1 [
forward(0.008289,0.000355) 1 [
forward(0.008413,0.000369) 1 [ rotate-x(-0.006494) 1 [
rotate-x(-0.592854) 1 [ rotate-z(-2.085023) 1 [
rotate-z(1.735374) 1 [ rotate-z(-2.045125) 1 [
rotate-z(0.203076) 1 [ rotate-z(1.750799) 1 [
rotate-z(-2.038688) 1 [ rotate-z(1.725007) 1 [
rotate-y(1.478109) 1 [ rotate-x(2.477117) 1 [
rotate-x(-2.441858) 1 [ forward(0.015082,0.000223) ] ]
] ] ] ] ] ] ] ] ] ] ] ] ] ] ] ] ] ] rotate-y(2.335438)
1 [ rotate-y(-1.042201) 1 [ rotate-y(-1.761594) 1 [
rotate-x(2.518405) 1 [ rotate-z(-0.739608) 1 [
rotate-x(0.426553) 1 [ rotate-z(-0.291483) 1 [
rotate-x(2.152738) 1 [ forward(0.013190,0.000414) ] ] ]
] ] ] ] ] ] ] ] ] ] ] ] ] ] ] ]
```



Evolved antenna ST5-3-10

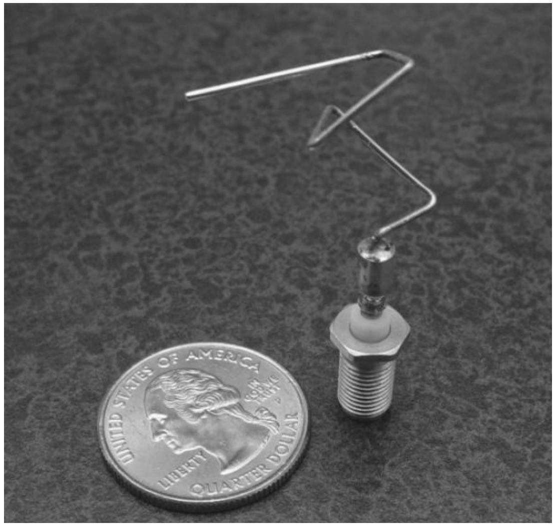
operator1   2   [subtree-1   subtree-2]



Genetic programming: Application

The change of the launch vehicle for the ST5 spacecraft leads to new requirements for the antenna

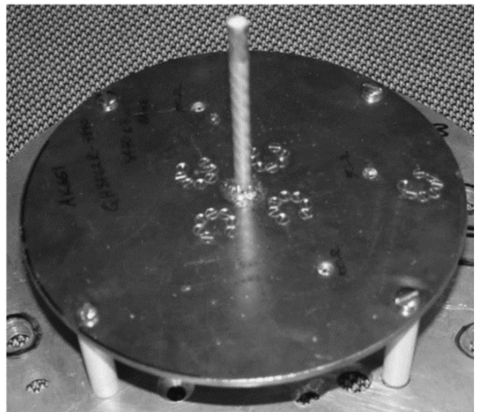
```
rotate-z(0.723536) 1 [ rotate-x(2.628787) 1 [
rotate-z(1.145415) 1 [ rotate-x(1.930810) 1 [
rotate-z(2.069497) 1 [ rotate-x(1.822537) 1 [
forward(0.007343,0.000406) 1 [ rotate-z(1.901507) 1 [
forward(0.013581,0.000406) 1 [ rotate-x(1.909851) 1 [
rotate-y(2.345316) 1 [ rotate-y(0.308043) 1 [
rotate-y(2.890265) 1 [ rotate-x(0.409742) 1 [
rotate-y(2.397507) 1 [ forward(0.011671,0.000406) 1 [
rotate-x(2.187298) 1 [ rotate-y(2.497974) 1 [
rotate-y(0.235619) 1 [ rotate-x(0.611508) 1 [
rotate-y(2.713447) 1 [ rotate-y(2.631141) 1 [
forward(0.011597,0.000406) 1 [ rotate-y(1.573367) 1 [
forward(0.007000,0.000406) 1 [ rotate-x(-0.974118) 1 [
rotate-y(2.890265) 1 [ rotate-z(1.482916) 1 [
forward(0.019955,0.000406) ] ] ] ] ] ] ] ] ] ] ] ] ] ] ] ] ] ]
] ] ] ] ] ] ] ] ] ]
```



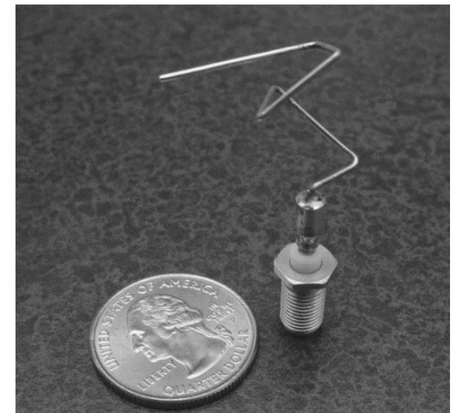
Re-evolved antenna ST5-33.142.7

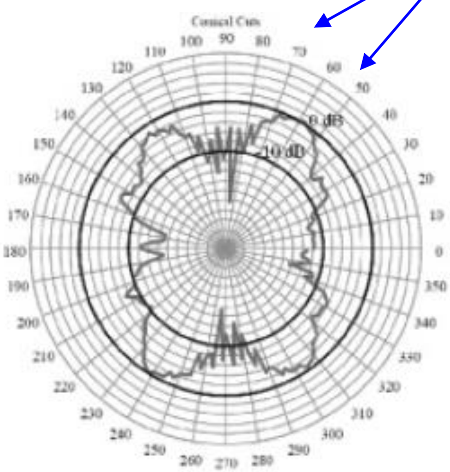
Genetic programming: Application

Quadrifilar helical antenna designed by human experts

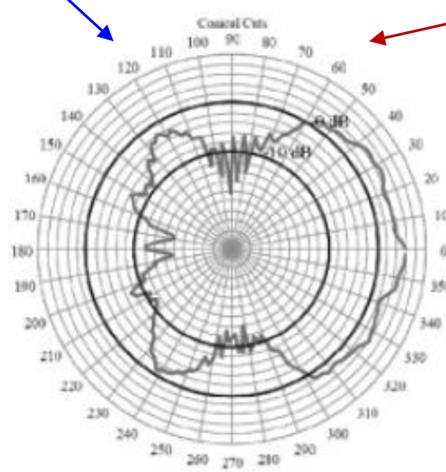


Re-evolved antenna ST5-33.142.7

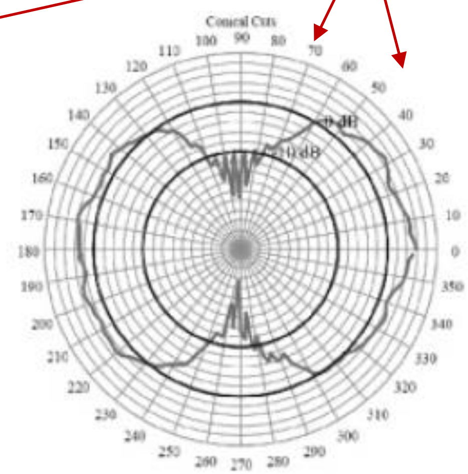




38% efficiency

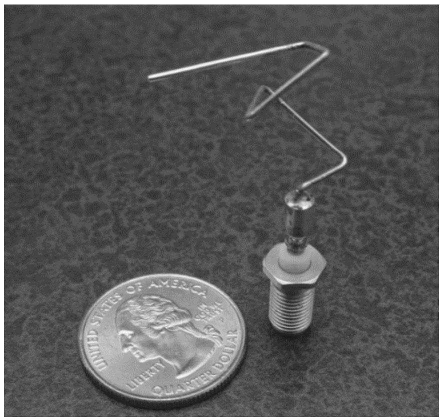


80% efficiency



93% efficiency

Genetic programming: Application



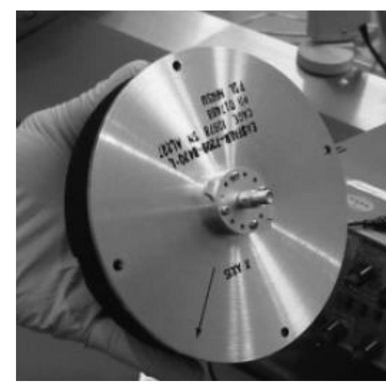
Re-evolved antenna ST5-33.142.7

February 25, 2005

Delivered to Goddard Space Flight Center to undergo tests

April 8, 2005

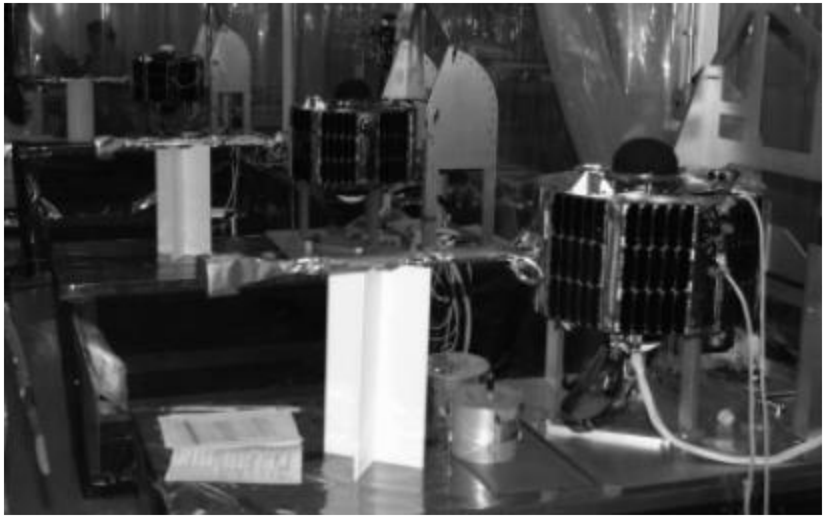
Complete the tests



March 22, 2006

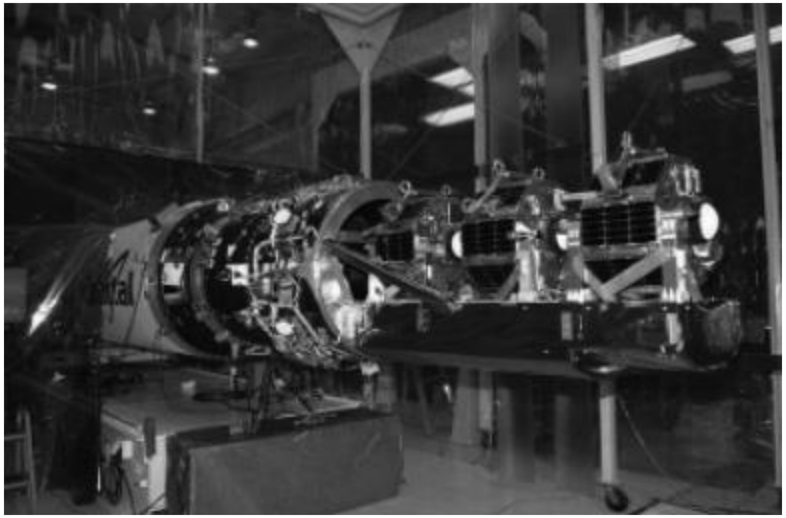
Launched from Vandenberg Air Force Base, California on a Pegasus XL rocket

Genetic programming: Application



Three ST5 spacecraft with the black radomes on top containing an evolved antenna, ST5-33.142.7

Three ST5 spacecraft mounted for launch on a Pegasus XL rocket



The first computer-evolved hardware in space

Genetic programming: Application

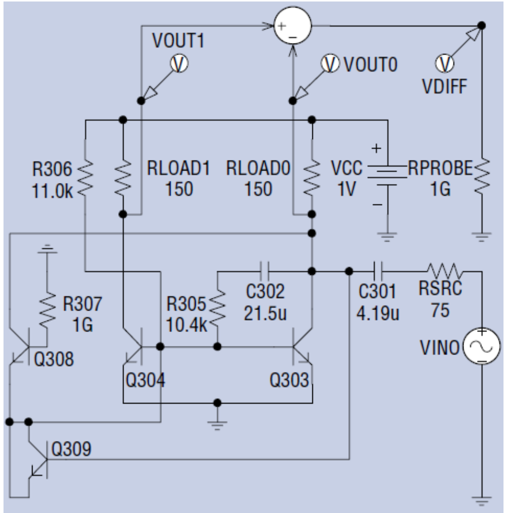
[J. R. Koza, et al. What's AI Done for Me Lately? Genetic Programming's Human-Competitive Results. IEEE Intelligent Systems, 18(3): 25-31, 2003.]

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 fourfold improvement 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."



Differential evolution

R. Storn

Differential Evolution (DE)

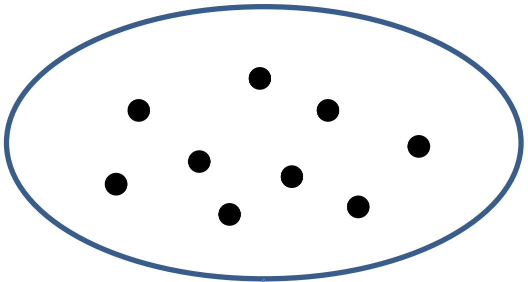
Typically applied to nonlinear and nondifferentiable continuous optimization

[R. Storn, K. Price. *Differential Evolution – A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces*. 1995]

International Computer Science Institute in Berkeley, USA

Representation	Real-valued representation
Recombination	Uniform crossover
Mutation	Differential mutation
Parent selection	Uniform random selection
Survivor selection	Deterministic elitist replacement (parent vs. offspring)

Differential evolution



Randomly select three parent solutions $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$

Differential mutation

$$\boldsymbol{v} = \boldsymbol{x} + F \cdot (\boldsymbol{y} - \boldsymbol{z})$$

$\boldsymbol{v}_i$

1	0	1	1	1	0	0	0

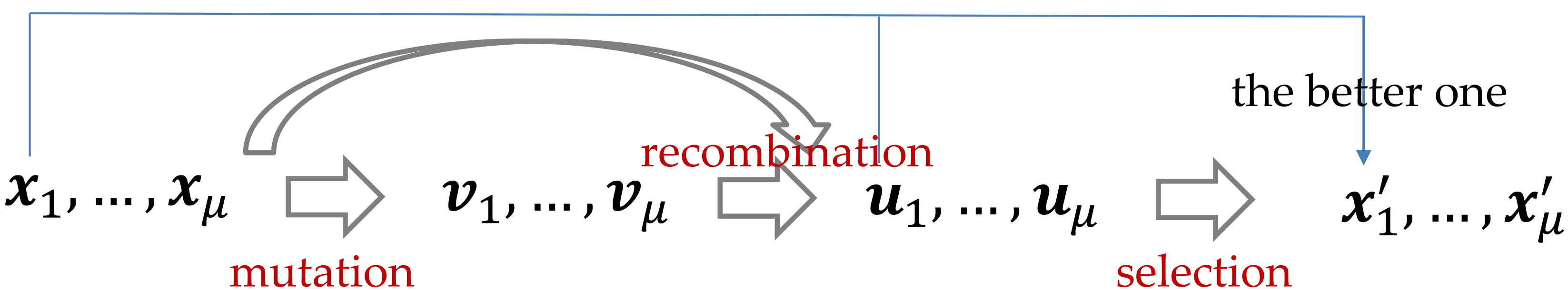
$\boldsymbol{x}_i$

0	0	1	0	1	0	1	0

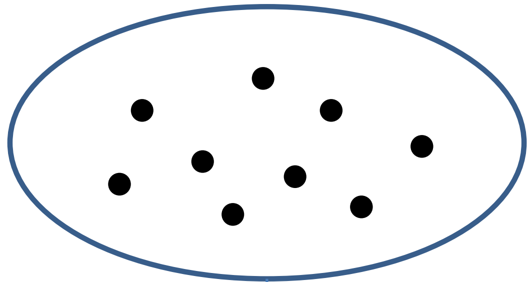
Select the value from $\boldsymbol{v}_i$ with prob. p for each position

Uniform crossover $\boldsymbol{u}_i$

1	0	1	1	1	0	1	0



Differential evolution



Randomly select three parent solutions $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}$

Differential mutation

$$\boldsymbol{v} = \boldsymbol{x} + F \cdot (\boldsymbol{y} - \boldsymbol{z})$$

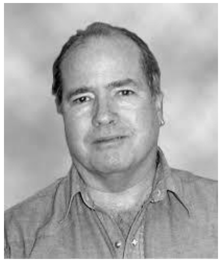
base vector $\begin{cases} \text{random} \\ \text{best} \end{cases}$

perturbation vector $\begin{cases} \boldsymbol{y} - \boldsymbol{z} & \text{randomly select two} \\ (\boldsymbol{y} - \boldsymbol{z}) + (\boldsymbol{y}' - \boldsymbol{z}') & \text{randomly select four} \end{cases}$

Variants of DE: DE/a/b/c

- a is the base vector (rand or best)
- b is the number of different vectors to define perturbation vector
- c denotes the crossover scheme ("bin" is uniform crossover)

Particle swarm optimization




J. Kennedy R. Eberhart

Particle Swarm Optimization (PSO)

Typically applied to nonlinear optimization

[J. Kennedy, R. Eberhart. *Particle Swarm Optimization*. 1995]

fish school


bird flock

Representation	Real-valued representation
Recombination	None
Mutation	Adding velocity vector
Parent selection	Deterministic (each parent creates one offspring via mutation)
Survivor selection	Generational (offspring replaces parents)

Particle swarm optimization

Each member in the population (a list):

$(\boldsymbol{x}, \boldsymbol{v})$

solution, named position

perturbation vector, named velocity

particle

inertia weight

learning rate for the personal influence

learning rate for the social influence

random matrices

Mutation

$$\boldsymbol{v}' = w \cdot \boldsymbol{v} + \phi_1 \mathbf{U}_1 \cdot (\boldsymbol{y} - \boldsymbol{x}) + \phi_2 \mathbf{U}_2 \cdot (\boldsymbol{z} - \boldsymbol{x})$$

the best position the member ever had

the best position the population ever had

$$\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{v}'$$

Particle swarm optimization

The i-th member in the population (a list):

$(\boldsymbol{x}_i, \boldsymbol{v}_i, \boldsymbol{y}_i)$

solution, named position

perturbation vector, named velocity

particle

personal best of the i-th population member

global best of the population

Mutation
$$\begin{cases} \boldsymbol{v}_i' = w \cdot \boldsymbol{v}_i + \phi_1 \mathbf{U_1} \cdot (\boldsymbol{y}_i - \boldsymbol{x}_i) + \phi_2 \mathbf{U_2} \cdot (\boldsymbol{z} - \boldsymbol{x}_i) \\ \boldsymbol{x}_i' = \boldsymbol{x}_i + \boldsymbol{v}_i' \end{cases}$$

$$\boldsymbol{y}_i' = \begin{cases} \boldsymbol{x}_i' & \text{if } f(\boldsymbol{x}_i') < f(\boldsymbol{y}_i) \\ \boldsymbol{y}_i & \text{Otherwise} \end{cases}$$

The global best $\boldsymbol{z}$ is updated if $\min\{f(\boldsymbol{x}_1'), \dots, f(\boldsymbol{x}_\mu')\} < f(\boldsymbol{z})$

Ant colony optimization

M. Dorigo

Ant Colony Optimization (ACO)

Typically applied to find good paths through graphs

[M. Dorigo. *Optimization, Learning and Natural Algorithms*. 1992]

Ants find the shortest path between their nest and a good source using **pheromone trails**

Solution representation	path on a graph

Solution construction
An ant moves on the graph according to the pheromone and length of each edge

Pheromone update
The pheromone of each edge is updated according to the number of ants traversing it and the lengths of constructed paths

Ant colony optimization

Solution construction
An ant moves on the graph according to the pheromone and length of each edge

For an ant k, if the current vertex is i, the probability of selecting j as the next vertex is

$$p_k(i,j) = \begin{cases} \dfrac{(\tau(i,j))^\alpha (\eta(i,j))^\beta}{\sum_{u \in J_k(i)} (\tau(i,u))^\alpha (\eta(i,u))^\beta}, & \text{if } j \in J_k(i) \\ 0, & \text{otherwise} \end{cases}$$

pheromone

usually $1/d(i,j)$, where $d(i,j)$ is the distance between i and j

vertices which are connected to i and unvisited by the ant k

Ant colony optimization

Pheromone update
The pheromone of each edge is updated according to the number of ants traversing it and the lengths of constructed paths

After the ants construct the paths, the pheromone is updated by

$$\tau(i,j) = (1-\rho)\cdot\tau(i,j) + \sum_{k=1}^{m} \Delta\tau_k(i,j)$$

number of ants, i.e., population size

evaporation factor

pheromone density laid on edge (i,j) by the ant k

$$\Delta\tau_k(i,j) = \begin{cases} \frac{1}{C_k}, & \text{if } (i,j) \in R^k \\ 0, & \text{otherwise} \end{cases}$$

length of the path constructed by the ant k

edge set traversed by the ant k

Estimation of distribution algorithms

S. Baluja

Estimation of Distribution Algorithms (EDA)

Applied to diverse optimization

[S. Baluja. *Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning*. 1994]

Carnegie Mellon University

EDA guide the search for the optimum by building and sampling explicit probabilistic models of promising candidate solutions

Select the fittest subset of sampled solutions

Model building

Model sampling

Estimation of distribution algorithms

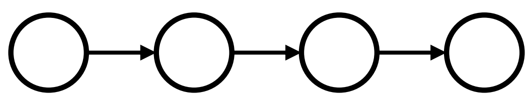
Model building

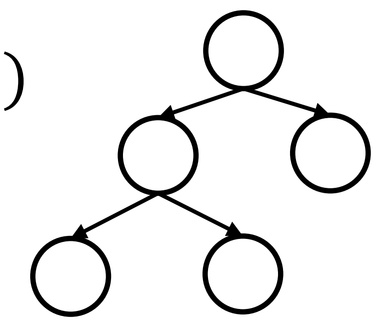
Model sampling

Probabilistic model $P(x_1, x_2, \ldots, x_n)$

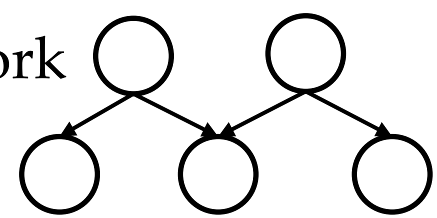
Univariate: $P(x_1) \cdot P(x_2) \cdot \cdots \cdot P(x_n)$

Bivariate: $\prod_{i=1}^{n} P(x_i \mid pa_i)$





Multivariate: Bayesian network



Summary

- Genetic algorithms
- Evolutionary strategies
- Evolutionary programming
- Genetic programming

Historical EA variants

- Differential evolution
- Particle swarm optimization
- Ant colony optimization
- Estimation of distribution algorithms

Recent EA variants

References

- A. E. Eiben and J. E. Smith. Introduction to Evolutionary Computing. Chapter 6.
- J. R. Koza, M. A. Keane and M. J. Streeter. What's AI done for me lately? Genetic programming's human-competitive results. *IEEE Intelligent Systems*, 2003, 18(3): 25-31.
- C. Igel. Neuroevolution for reinforcement learning using evolution strategies. In: *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'03)*, 2003, pages 2588-2595, Canberra, Australia.
- M. Dorigo and T. Stützle. Ant Colony Optimization. 2004, MIT Press.
- G. S. Hornby, J. D. Lohn and D. S. Linden. Computer-automated evolution of an X-band antenna for NASA's space technology 5 mission. *Evolutionary Computation*, 2011, 19(1): 1-23.
- C. Qian, Y. Yu and Z.-H. Zhou. Pareto ensemble pruning. In: *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI'15)*, 2015, pages 2935-2941, Austin, TX.