

Applications of Evolutionary Algorithms

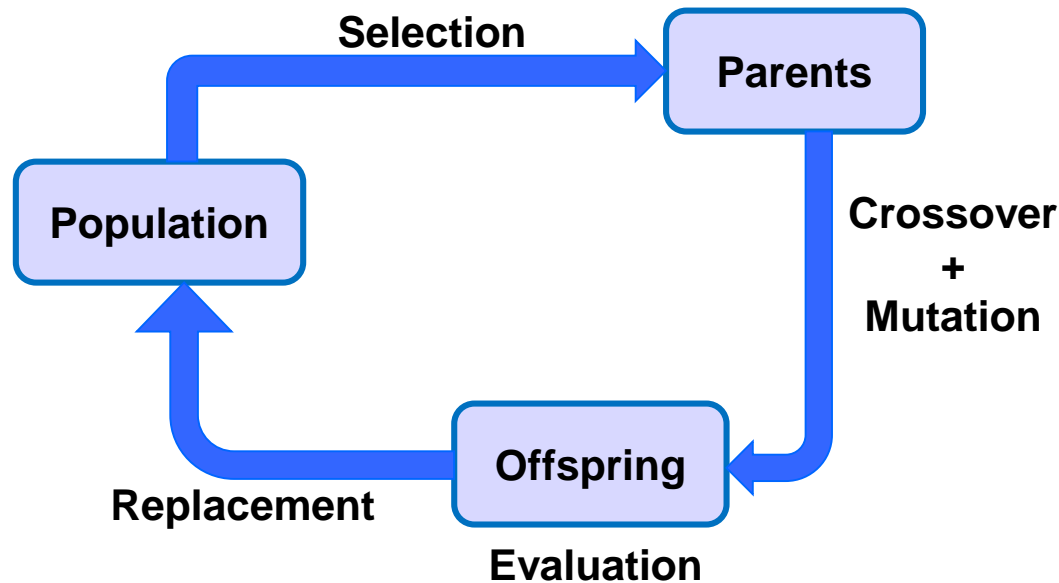
Jiří Kubalík

email: kubalik@ciirc.cvut.cz

Czech Technical University in Prague
Czech Institute of Informatics, Robotics, and Cybernetics
Zikova street 1903/4, 166 36 Praha 6
Czech Republic

Evolutionary Algorithms

- **Evolutionary Algorithms are general-purpose stochastic optimization algorithms.**
- Problem solution can be represented as binary string, real-valued string, string of symbols, tree or graph.
- Optimized objective function can be continuous/discrete, multimodal, nonlinear, multidimensional, noisy.
The problem can involve multiple optimization objectives as well.
- **A typical evolutionary model:**



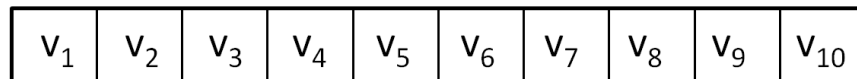
Genetic Programming (GP)

GP shares with GA the philosophy of survival and reproduction of the fittest and the analogy of naturally occurring genetic operators.

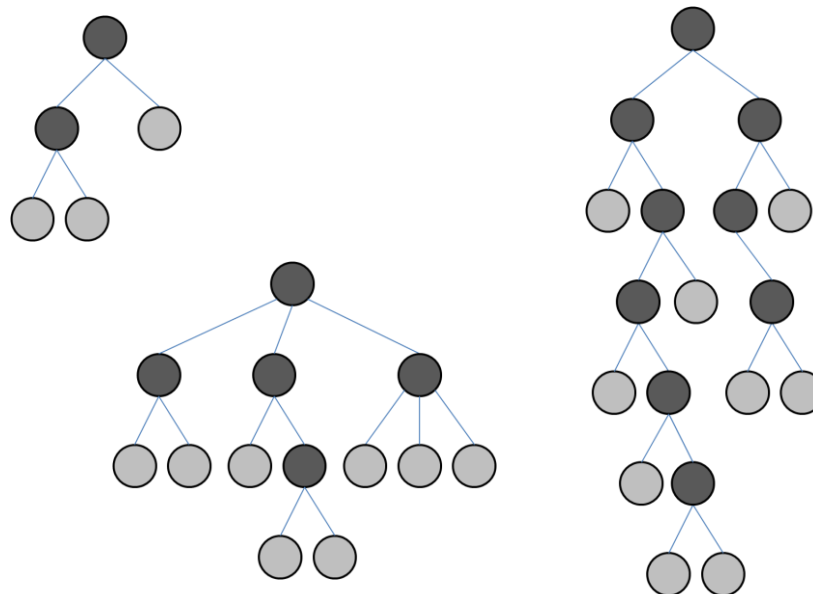
GP differs from GA in a representation, genetic operators and a scope of applications.

Structures evolved in GP are trees of dynamically varying size and shape representing hierarchical computer programs.

GA chromosome of fixed length:



GP trees:

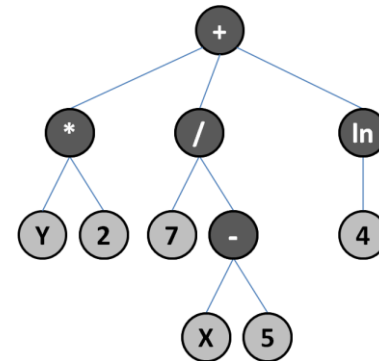


GP: Application Domains

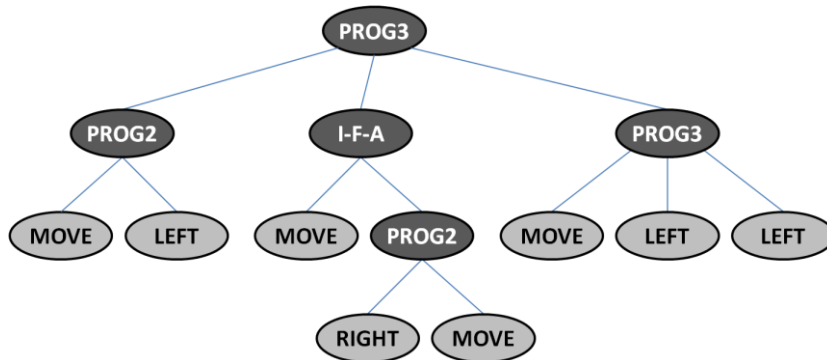
Applications

- learning programs,
- learning decision trees,
- learning rules,
- learning strategies,
- ...

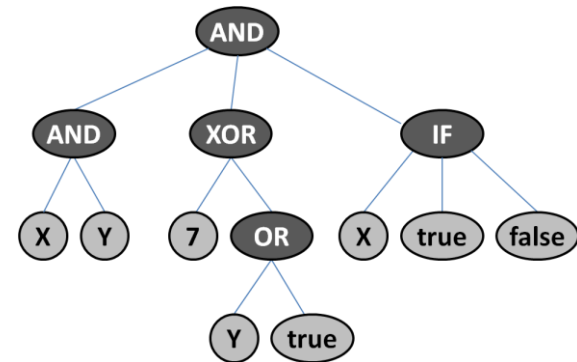
Arithmetic expressions



Artificial ant strategy



Logical expressions



GP: Trigonometric Identity

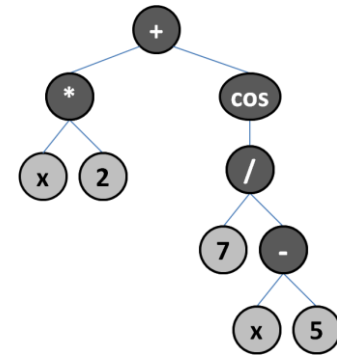
An illustrative example of GP in action (one of many interesting examples from)

- Koza, J.: Genetic Programming: vol. 1 , On the programming of computers by means of natural selection, MIT Press, 1992

GP used to find an equivalent expression to $\cos(2x)$.

GP implementation:

- **Terminal set** $T = \{x, 1.0\}$
- **Function set** $F = \{+, -, *, \%, \sin\}$
- **Training cases:** 20 pairs (x_i, y_i) , where x_i are values evenly distributed in interval $(0, 2\pi)$.
- **Fitness:** Sum of absolute differences between desired y_i and the values returned by generated expressions.
- **Stopping criterion:** A solution found that gives the error less than 0.01.



GP: Trigonometric Identity

1. run, 13th generation

```
(- (- 1 (* (sin x) (sin x))) (* (sin x) (sin x)))
```

which equals (after editing) to $1-2*\sin^2x$.

2. run, 34th generation

```
(- 1 (* (* (sin x) (sin x)) 2))
```

which is just another way of writing the same expression.

GP: Trigonometric Identity

1. run, 13th generation

```
(- (- 1 (* (sin x) (sin x))) (* (sin x) (sin x)))
```

which equals (after editing) to $1-2*\sin^2x$.

2. run, 34th generation

```
(- 1 (* (* (sin x) (sin x)) 2))
```

which is just another way of writing the same expression.

3. run, 30th generation

```
(sin (- (- 2 (* x 2)))
```

```
  (sin (sin (sin (sin (sin (sin (* (sin (sin 1))
```

```
    (sin (sin 1))
```

```
    )))))))
```

GP: Trigonometric Identity

1. run, 13th generation

```
(- (- 1 (* (sin x) (sin x))) (* (sin x) (sin x)))
```

which equals (after editing) to $1-2*\sin^2x$.

2. run, 34th generation

```
(- 1 (* (* (sin x) (sin x)) 2))
```

which is just another way of writing the same expression.

3. run, 30th generation

```
(sin (- (- 2 (* x 2))
```

```
  (sin (sin (sin (sin (sin (sin (* (sin (sin 1))
```

```
    (sin (sin 1))
```

```
    ))))))))
```

2 minus the expression on the 2nd and 3rd rows is almost $\pi/2$, so the discovered identity is $\cos(2x) = \sin(\pi/2 - 2x)$.

Human-Competitive Results

- John R. Koza, Martin A. Keane, Matthew J. Streeter: *What's AI Done for Me Lately? Genetic Programming's Human-Competitive Results*. IEEE Intelligent Systems 18(3): 25-31 (2003)
<http://www.genetic-programming.com/humancompetitive.html>
http://www.cs.bham.ac.uk/~wbl/biblio/cache/http___www.genetic-programming.com_jkpdf_ieee2003intelligent.pdf
 - The automated problem-solving technique of genetic programming has generated at least **36 human-competitive results (21 of which duplicate previously patented inventions, 6 of which duplicate functionality of inventions patented after January 1 2000)**.
 - It also covers two automatically synthesized controllers for which the authors have applied for a patent and includes examples of an automatically synthesized antenna, classifier program, and mathematical algorithm.

More recent survey on human-competitive results produced by GP:

<http://www.springerlink.com/content/92n753376213655k/fulltext.pdf>

Criteria of Human-Competitive Results

- We say that an automatically created **result** is “*human-competitive*” if it satisfies one or more of the eight criteria below.
- 1. The **result was patented as an invention** in the past, **is an improvement over a patented invention**, or would qualify today as a patentable new invention.
- 2. The **result is equal to or better than a result that was accepted as a new scientific result** at the time when it was published in a peer-reviewed scientific journal.
- 3. The result is equal to or better than a result that was placed into a database or archive of results maintained by an internationally recognized panel of scientific experts.
- 4. The result is publishable in its own right as a new scientific result *independent* of the fact that the result was mechanically created.
- 5. The **result is equal to or better than the most recent human-created solution** to a long-standing problem for which there has been a succession of increasingly better human-created solutions.
- 6. The result is equal to or better than a result that was considered an achievement in its field at the time it was first discovered.
- 7. The result solves a problem of indisputable difficulty in its field.
- 8. The **result holds its own or wins a regulated competition involving human contestants** (in the form of either live human players or human-written computer programs).

Human-Competitive Results

John R. Koza et al.: What's AI Done for Me Lately? Genetic Programming's Human-Competitive Results.

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

Six Post-2000 patented analog circuits

- John R. Koza et al.: What's AI Done for Me Lately? Genetic Programming's Human-Competitive Results.

Invention	Date	Inventor	Place	Patent
Low-voltage balun (balance/unbalance) circuit	2001	Sang Gug Lee	Information and Communications University	6,265,908
Mixed analog-digital circuit for variable capacitance	2000	Turgut Sefket Aytur	Lucent Technologies	6,013,958
Voltage-current conversion circuit	2000	Akira Ikeuchi and Naoshi Tokuda	Mitsumi Electric	6,166,529
Low-voltage high-current circuit for testing a voltage source	2001	Timothy Daun-Lindberg and Michael Miller	International Business Machines	6,211,726
Low-voltage cubic function generator	2000	Stefano Cipriani and Anthony A. Takeshian	Conexant Systems	6,160,427
Tunable integrated active filter	2001	Robert Irvine and Bernd Kolb	Infineon Technologies	6,225,859

Automated Design of Electrical Circuits

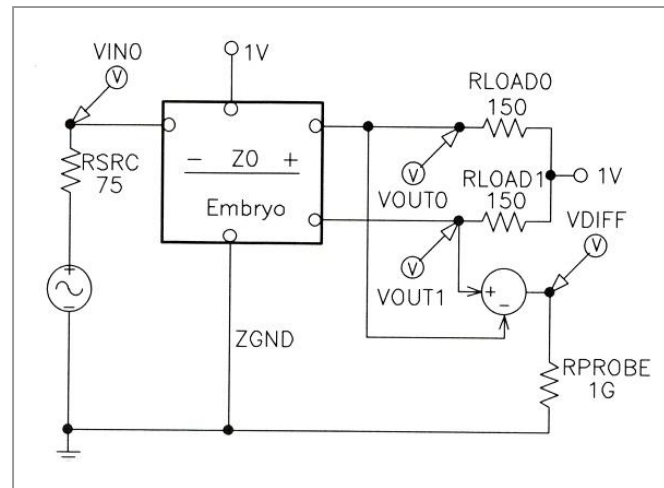
Automated “*What You Want Is What You Get*” process for circuit synthesis.

- **Genetic programming** used to synthesize both
 - the **structure/topology**, and
 - **sizing** (numerical component values)for circuits that duplicate the patented inventions’ functionality.

- Method
 - Starts from a **high-level statement of a circuit’s desired behavior and characteristics** and only minimal knowledge about analogue electrical circuits. Then, a **fitness measure** is created that reflects the invention’s performance and characteristics – it **specifies the desired time- or frequency-domain outputs, given various inputs**.
 - Employs a circuit simulator for analyzing candidate circuits, but **does not rely on domain expertise or knowledge concerning the synthesis of circuits**.

Automated Design of Electrical Circuits

- Method
 - For each problem, a **test fixture** consisting of appropriate hard-wired components (such as a source resistor or load resistor) connected to the input ports and desired output ports is used.

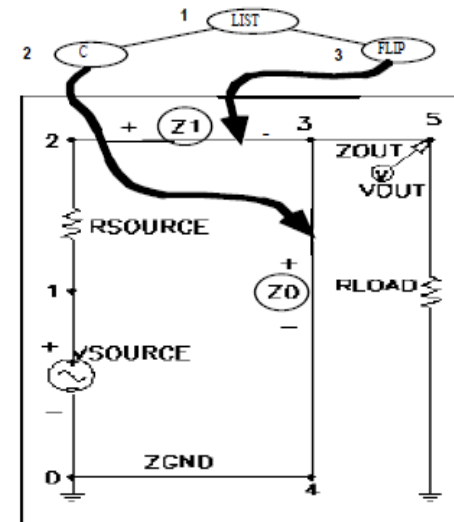


Test fixture

WYWIWYG: Embryonic Electrical Circuit

The Mapping between Program Trees and Electrical Circuits

- The **growth process** used for electrical circuits begins with a **very simple embryonic electrical circuit** and builds a more complex circuit by **progressively executing the functions in a circuit-constructing program tree**.
- The embryonic circuit used on a particular problem depends on the number of input signals and the number of output signals.
- The result of this process is
 - the topology of the circuit,
 - the choice of the types of components that are situated at each location within the topology,
 - and the sizing of the components.

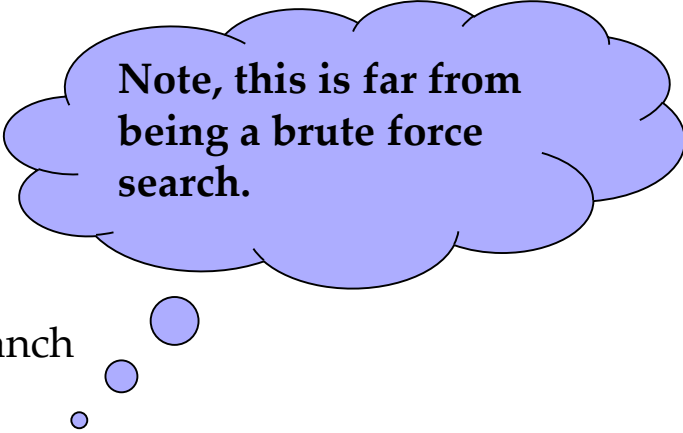


WYWIWYG: Fitness Assignment

- **Circuit-constructing program tree evaluation** in the population begins with its **execution**.
 - This execution applies the functions in the program tree to the very simple embryonic circuit, thereby developing the embryonic circuit into a fully developed circuit.
 - A netlist that identifies each component of the circuit, the nodes to which that component is connected, and the value of that component is then created.
- **Circuit is then simulated using SPICE** (an acronym for Simulation Program with Integrated Circuit Emphasis) to determine its behavior.
- **Fitness measure may incorporate many characteristic** or combination of characteristics of the circuit, including
 - the circuit's behavior in the time domain,
 - its behavior in the frequency domain,
 - its power consumption,
 - or the number, cost, or surface area of its components.

GP Control Parameters Setup

- **Population size: 640,000**
- Prossover = 89%
- Pmutation = 1%
- Preproduction = 10%
- Maximum 200 nodes for each value-producing branch
- Parallel Parsytec computer system
 - 64 x 80 MHz Power PC 601 processors arranged in a toroidal mesh
- **Parallel GA**
 - deme size: 10,000
 - 64 demes
 - Migration rate: 2%



Note, this is far from being a brute force search.

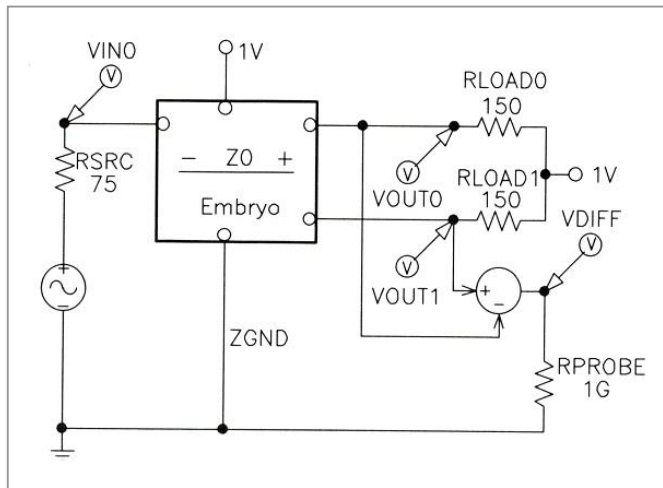
Low-Voltage Balun Circuit

- A **balun (balance/unbalance) circuit's** purpose is to produce two outputs from a single input
 - each having half of the input's amplitude;
 - one output should be in phase with the input while the other should be 180 degrees out of phase with the input, and both should have the same DC offset.
- The **fitness** measure was based on
 - a frequency sweep analysis designed to measure the magnitude and phase of the circuit's two outputs and
 - a Fourier analysis designed to measure harmonic distortion.

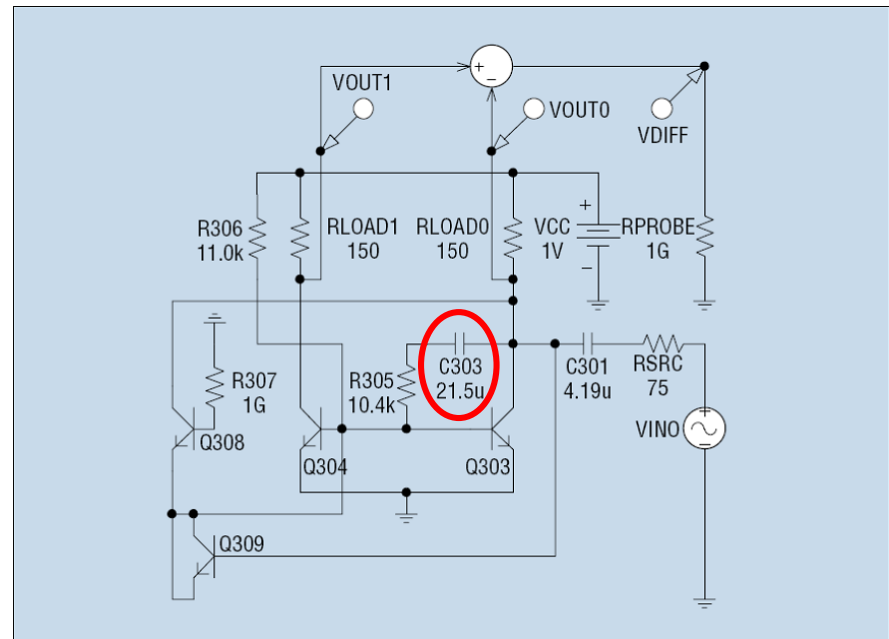
Genetically Evolved Low-Voltage Balun Circuit

- Evolved circuit is **roughly a fourfold improvement over the patented circuit** in terms of the fitness measure.
 - It is superior both in terms of its frequency response and harmonic distortion.

Test fixture



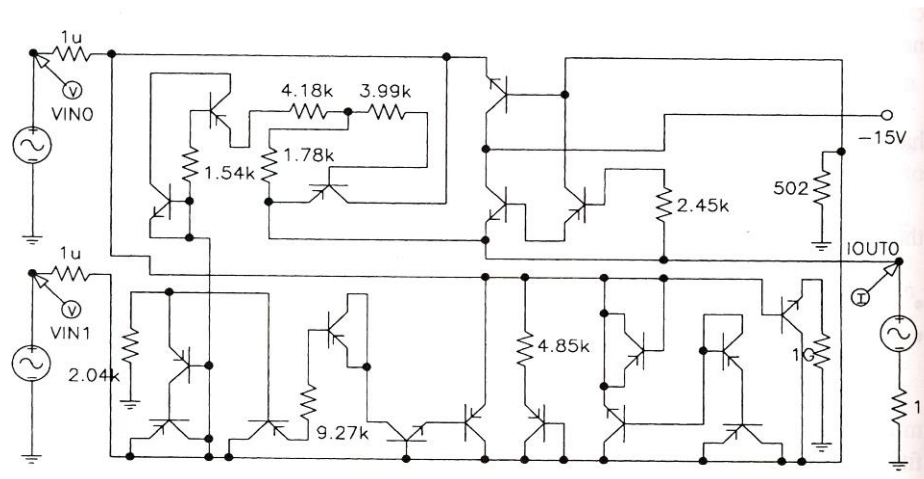
Evolved balun circuit



John R. Koza et al.: What's AI Done for Me Lately? Genetic Programming's Human-Competitive Results.

Voltage-Current Conversion Circuit

- **Voltage-current conversion circuit's** purpose is to take two voltages as input and to produce as output a stable current whose magnitude is proportional to the difference between the voltages.
- **Fitness** measure is based on four time-domain input signals.
- Genetically evolved circuit (**entirely different than the patented circuit**)
 - has roughly **62 percent of the average (weighted) error of the patented circuit** and
 - **outperformed the patented circuit on additional previously unseen test cases.**

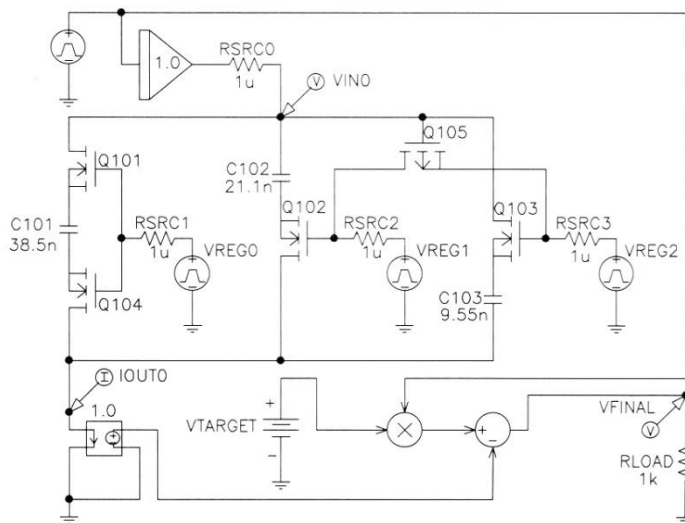


John R. Koza et al.: What's AI Done for Me Lately? Genetic Programming's Human-Competitive Results.

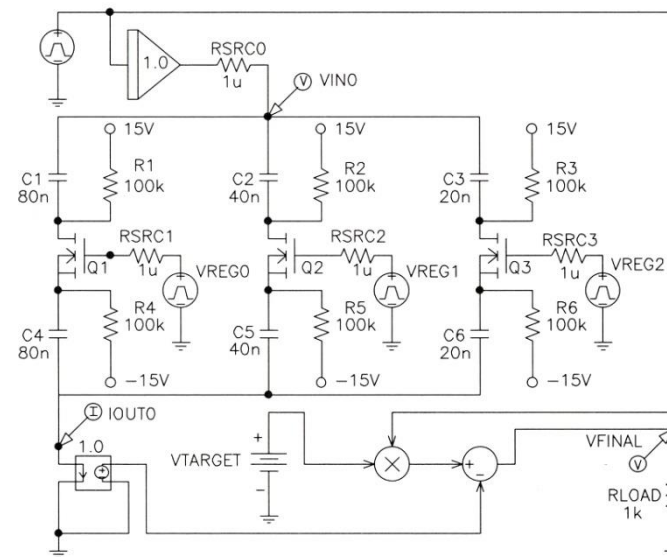
Mixed Analog-Digital Register-Controlled Variable Capacitor

- Mixed analog-digital variable capacitor circuit has a capacitance controlled by the value stored in a digital register.
- **Fitness measure** was based on the error accumulated by 16 combinations of time-domain test signals ranging over all eight possible values of a 3-bit digital register for two different analog input signals.
- **The evolved circuit performs as well as the patented circuit.**

Evolved circuit



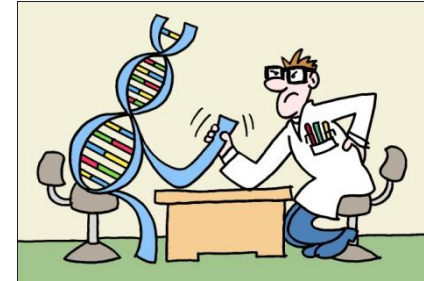
Patented circuit



John R. Koza et al.: What's AI Done for Me Lately? Genetic Programming's Human-Competitive Results.

HUMIES

- Annual “**HUMIES**” awards for human-competitive results produced by genetic and evolutionary computation held at the Genetic and Evolutionary Computation Conference (GECCO)



- Entries present **human-competitive results** that have been **produced by any form of genetic and evolutionary computation** (including, but not limited to genetic algorithms, genetic programming, evolution strategies, evolutionary programming, learning classifier systems, grammatical evolution, gene expression programming, differential evolution, etc.) and that have been published in the open literature.
- Human-competitive results awarded in areas:
 - Analog circuit design
 - Quantum circuit design
 - Physics
 - Digital circuits/programs
 - Chemistry
 - Game strategies
 - Image processing
 - Antenna design
 - Classical optimization
 - ...

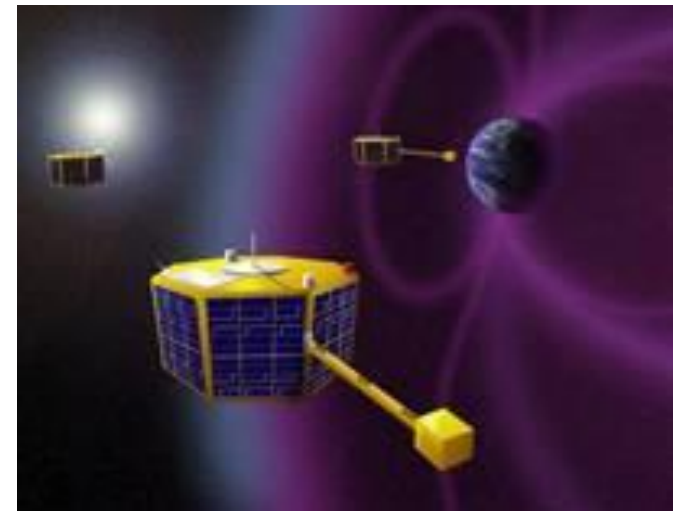
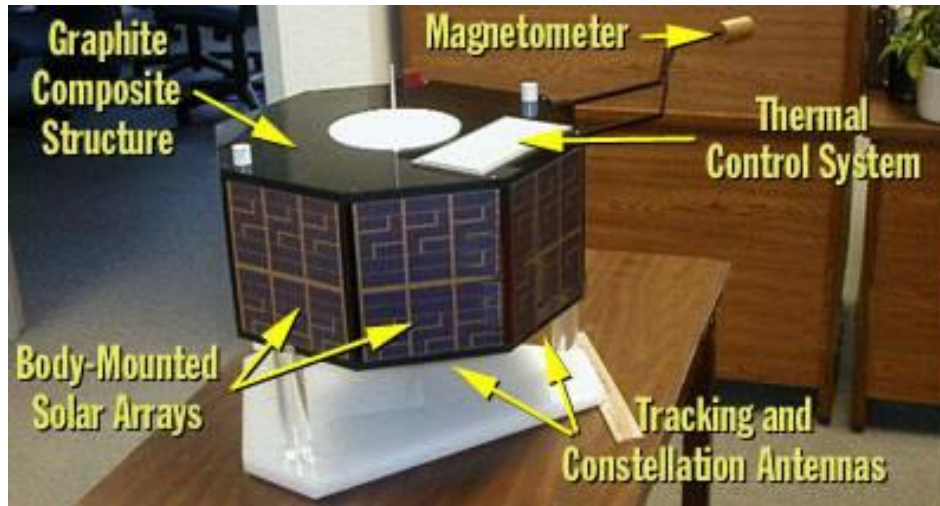
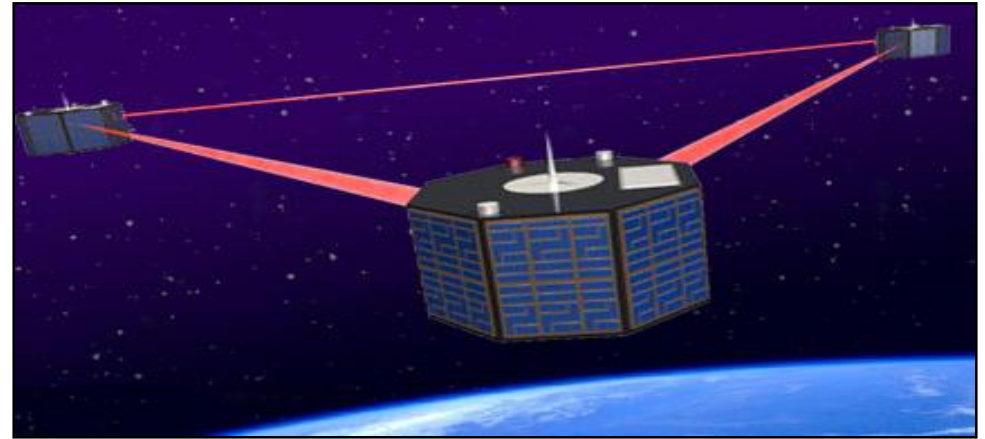
<http://www.genetic-programming.org/combined.html>

2004 Human-Competitive Awards in Genetic and Evolutionary Computation

- <http://www.genetic-programming.org/gecco2004hc.html>
- **\$1500 – Gold**
 - Jason D. Lohn, Gregory S. Hornby, Derek S. Linden: **An Evolved Antenna for Deployment on NASA's Space Technology 5 Mission**
 - Lee Spector: **Automatic Quantum Computer Programming: A Genetic Programming Approach**
- **\$500 – Silver**
 - Alex Fukunaga: **Evolving Local Search Heuristics for SAT Using GP**
 - Hod Lipson: **How to Draw a Straight Line Using a GP: Benchmarking Evolutionary Design Against 19th Century Kinematic Synthesis**
 - Bijan Khosraviani, Raymond E. Levitt, John R. Koza: **Organization Design Optimization Using Genetic Programming**
- **\$500 – Bronze**
 - Adrian Stoica, Ricardo Zebulum, Didier Keymeulen, Michael Ian Ferguson, Vu Duong, Xin Guo: **Taking evolutionary circuit design from experimentation to implementation: some useful techniques and a silicon demonstration**

The winner of Humies 2004

- Three nanosats (20in diameter).
- Measure effect of solar activity on the Earth's magnetosphere.



© Jason D. Lohn, Gregory S. Hornby, Derek S. Linden: Human-Competitive Results: Evolved Antennas for Deployment on NASA's Space Technology 5 Mission

Evolved Antennas for Deployment on NASA's Space Technology 5 Mission

■ Original ST5 Antenna Requirements

- Transmit: 8470 MHz
- Receive: 7209.125 MHz
- Gain:
 - >= 0dBic, 40 to 80 degrees
 - >= 2dBic, 80 degrees
 - >= 4dBic, 90 degrees
- 50 Ohm impedance
- Voltage Standing Wave Ratio (VSWR):
 - < 1.2 at Transmit Freq
 - < 1.5 at Receive Freq
- Fit inside a 6" cylinder

■ ST5 Quadrifilar Helical Antenna

- designed by a team of human designers

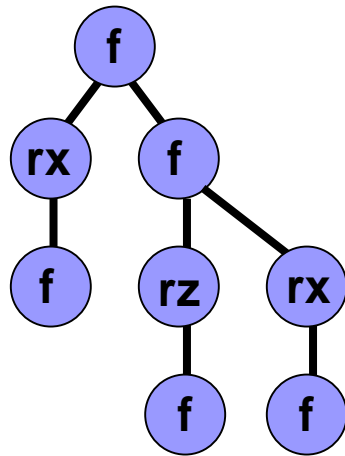


© Jason D. Lohn, Gregory S. Hornby, Derek S. Linden: Human-Competitive Results: Evolved Antennas for Deployment on NASA's ST5 Mission

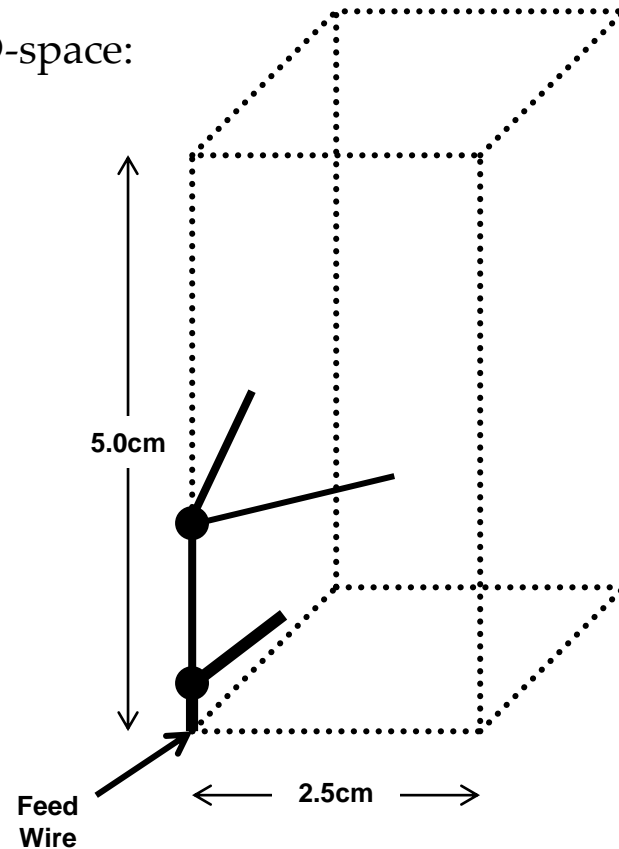
Evolved Antenna for Space Technology 5 mission

■ Branching EA: Antenna Genotype

- Genotype is a tree-structured encoding that specifies the construction of a wire form
- Genotype specifies design of 1 arm in 3D-space:



- Branching in genotype results in branching in wire form



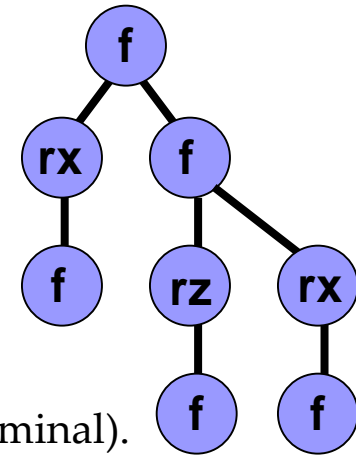
Evolved Antenna for Space Technology 5 mission

■ Branching EA: Antenna Construction Commands

- forward(length radius)
- rotate_x(angle)
- rotate_y(angle)
- rotate_z(angle)

Forward() command can have 0,1,2, or 3 children.

Rotate_x/y/z() commands have exactly 1 child (always non-terminal).

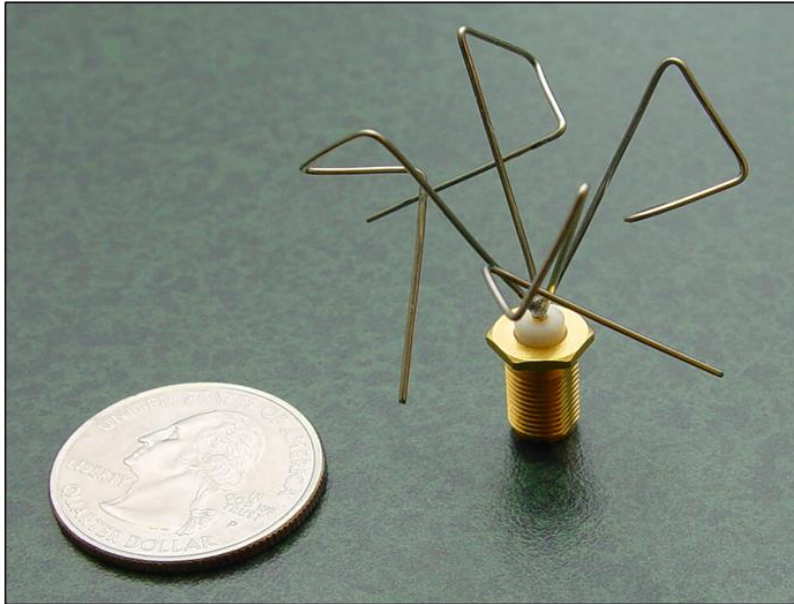


■ Fitness function (to be minimized):

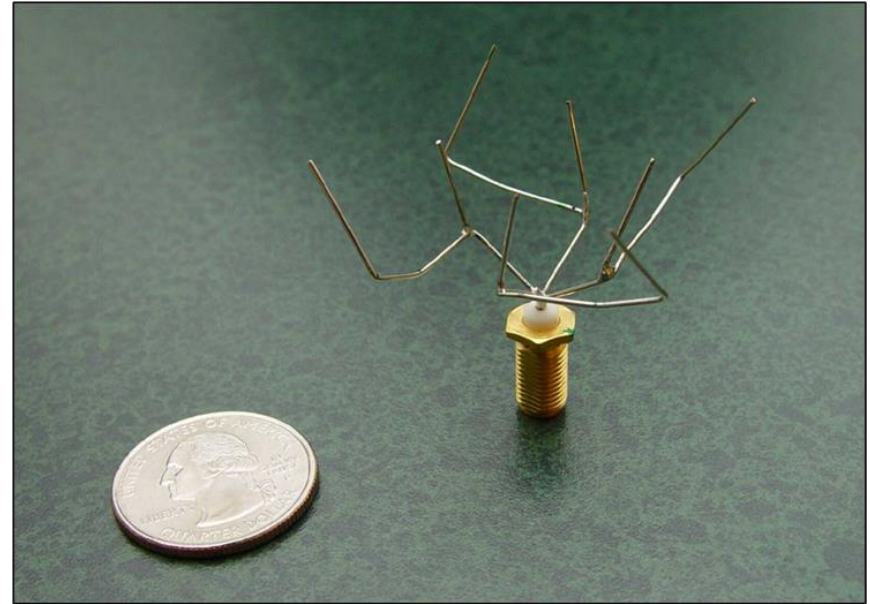
$$F = \text{VSWR_Score} * \text{Gain_Score} * \text{Penalty_Score}$$

Evolved Antenna for Space Technology 5 mission

- 1st Set of Genetically Evolved Antennas



Non-branching:
ST5-4W-03

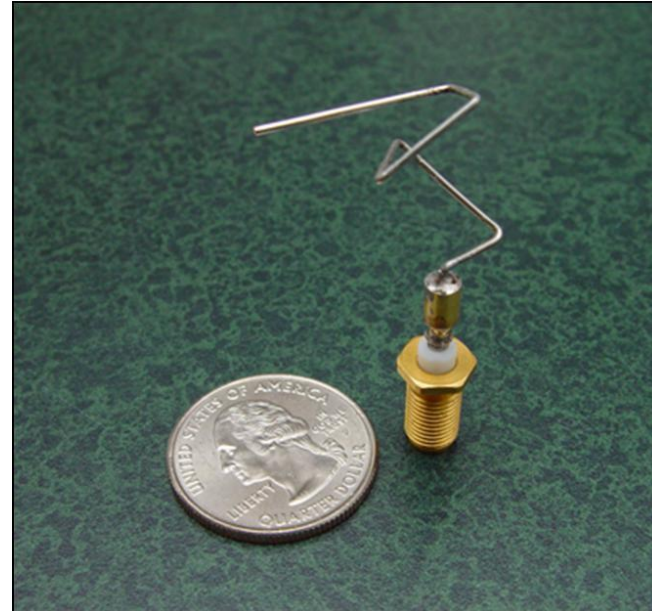
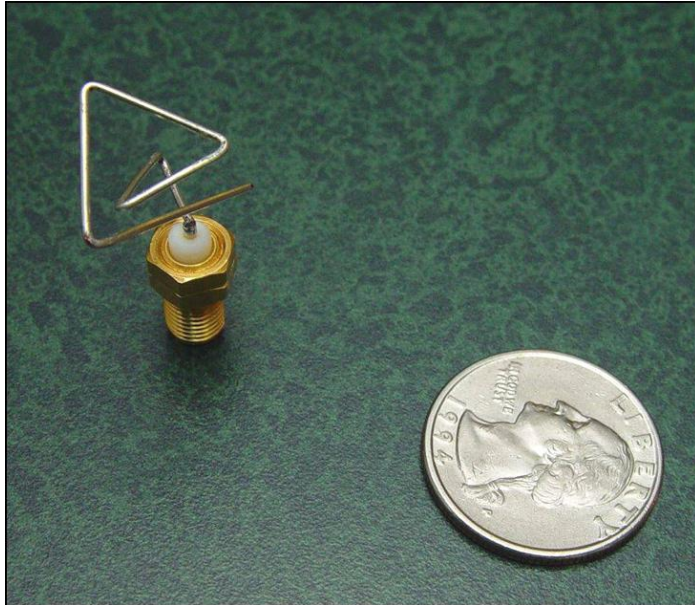


Branching:
ST5-3-10

© Jason D. Lohn, Gregory S. Hornby, Derek S. Linden: Human-Competitive Results: Evolved Antennas for Deployment on NASA's Space Technology 5 Mission

Evolved Antenna for Space Technology 5 mission

- 2nd Set of genetically evolved antennas for new mission requirements



© Jason D. Lohn, Gregory S. Hornby, Derek S. Linden: Human-Competitive Results: Evolved Antennas for Deployment on NASA's Space Technology 5 Mission

EA 1 – Vector of Parameters

EA 2 – Constructive Process

Evolved Antenna for Space Technology 5 mission

■ Conclusion

- Meets mission requirements
- Better than conventional design
- Successfully passed space qualification
- **First Evolved Hardware in Space when mission launched in 2005**

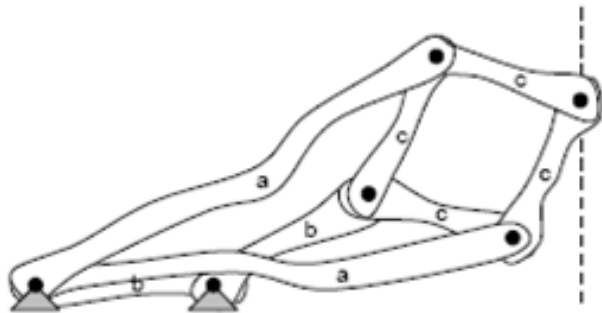
- **Direct competition:** The antenna designed by the contracting team of human designers for the Space Technology 5 mission - which won the bid against several competing organizations to supply the antenna - did not meet the mission requirements while the evolved antennas did meet these requirements.

■ Evolutionary design:

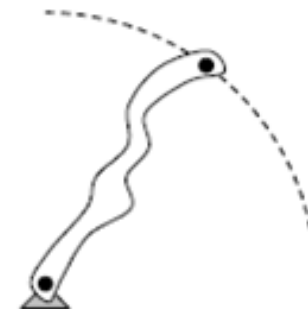
- Fast design cycles save time/money (**4 weeks from start-to-first-hardware**)
- Fast design cycles allow iterative “what-if”
- Can rapidly respond to changing requirements
- Can produce new types of designs
- May be able to produce designs of previously unachievable performance

How to Draw a Straight Line Using a GP

- Hod Lipson: How to Draw a Straight Line Using a GP: Benchmarking Evolutionary Design Against 19th Century Kinematic Synthesis
 - This entry presents the application of genetic programming to the synthesis of compound 2D kinematic mechanisms, and benchmarks the results against one of the classical kinematic challenges of 19th century mechanical design.
- Test Case: **The Straight Line Problem**
 - The straight-line problem seeks a kinematic mechanism that traces a straight line without reference to an existing straight line.
 - For example, a circle is easy, a line is a challenge!



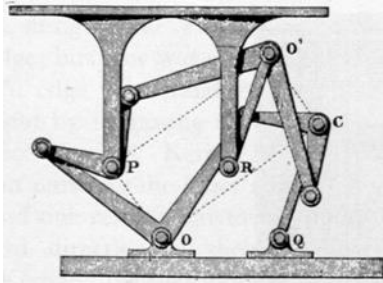
line



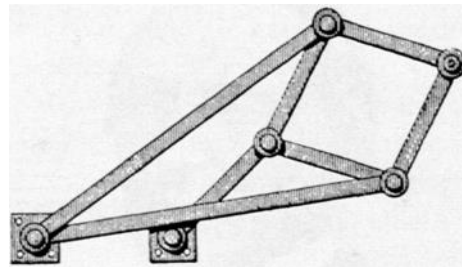
circle

How to Draw a Straight Line Using a GP

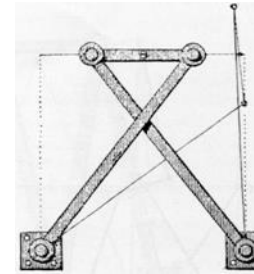
- Some key straight-line mechanisms



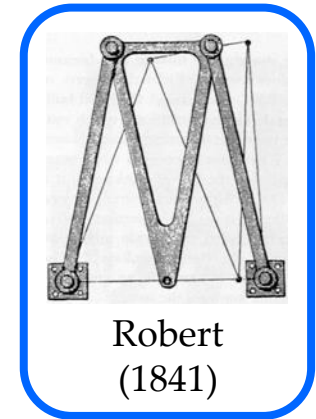
Silverster-Kempe's
(1877)



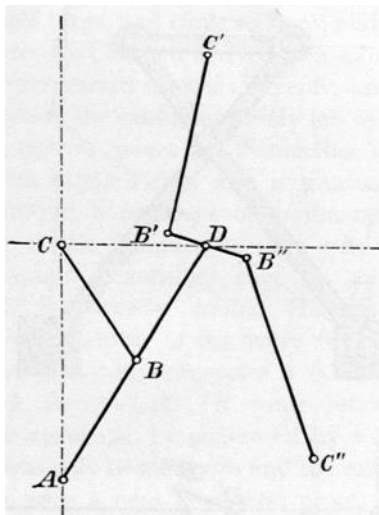
Peaucellier
(1873)



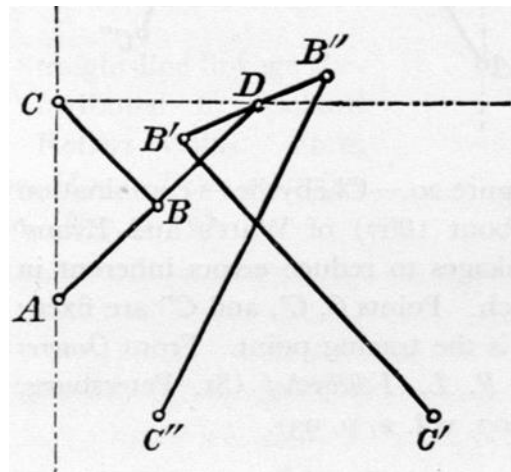
Chebyshev
(1867)



Robert
(1841)



Chebyshev
(1867)

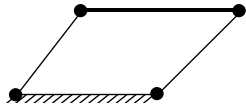


Chebyshev-Evans
(1907)

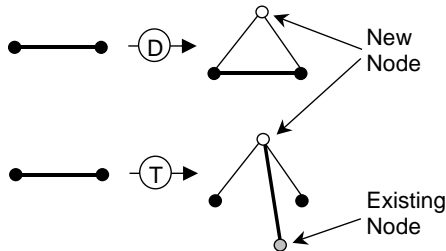
See
<http://kmoddl.library.cornell.edu>

How to Draw a Straight Line Using a GP

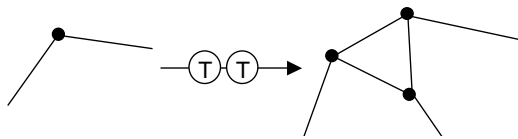
■ Top down encoding of a mechanism



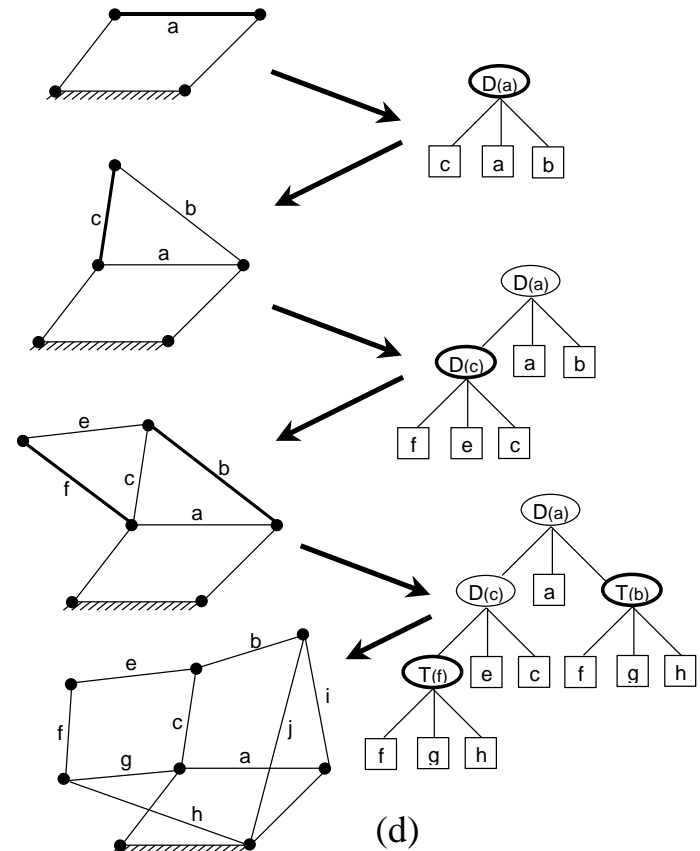
Start with Embryo with desired # of DoF, e.g. a four-bar mechanism (1 DoF)



Two variation operators maintain DoF



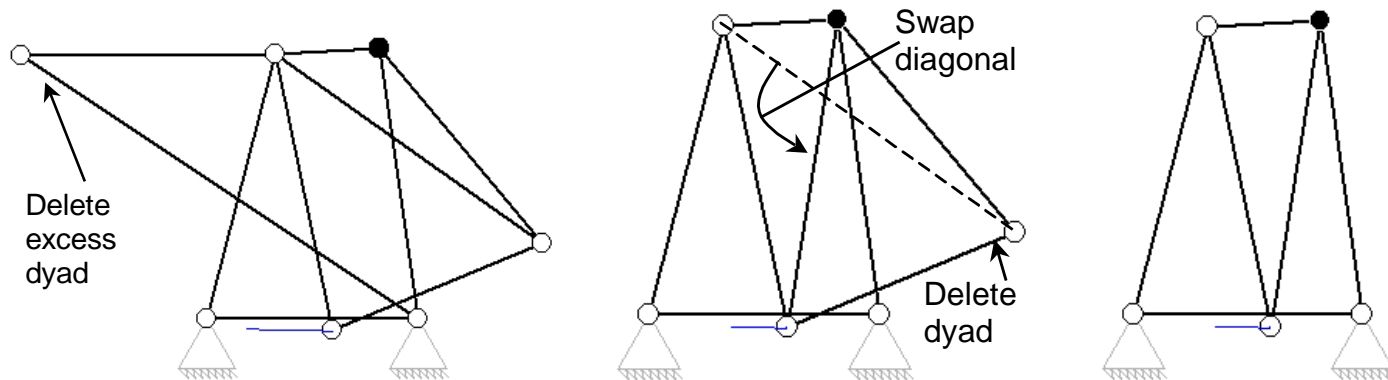
E.g. Transform dyad into tryad



Example: A tree that constructs this 1-DoF compound mechanism

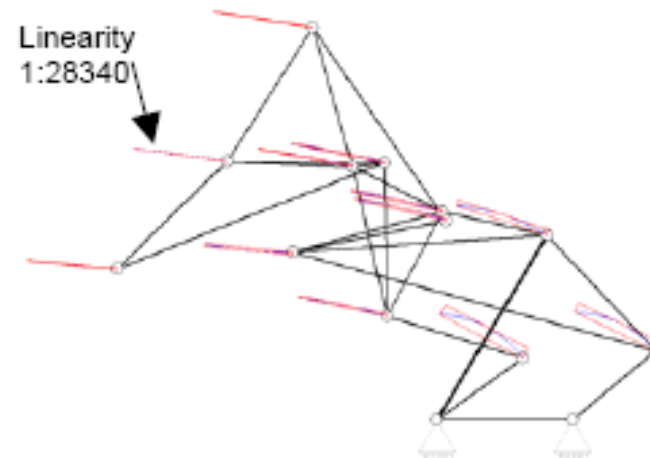
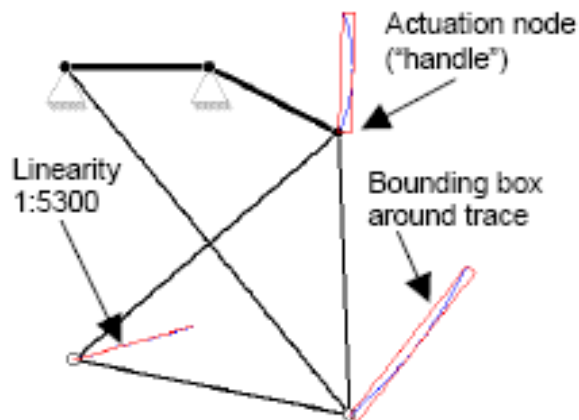
How to Draw a Straight Line Using a GP

- **Comparison of mechanisms** can be difficult
 - Equivalent mechanisms may appear very different
 - Masked by excess and redundant topology
- **Two transformations** allow moving in “neutral pathways” of mechanisms
 - Rigid diagonal swap
 - Redundant dyad removal/addition



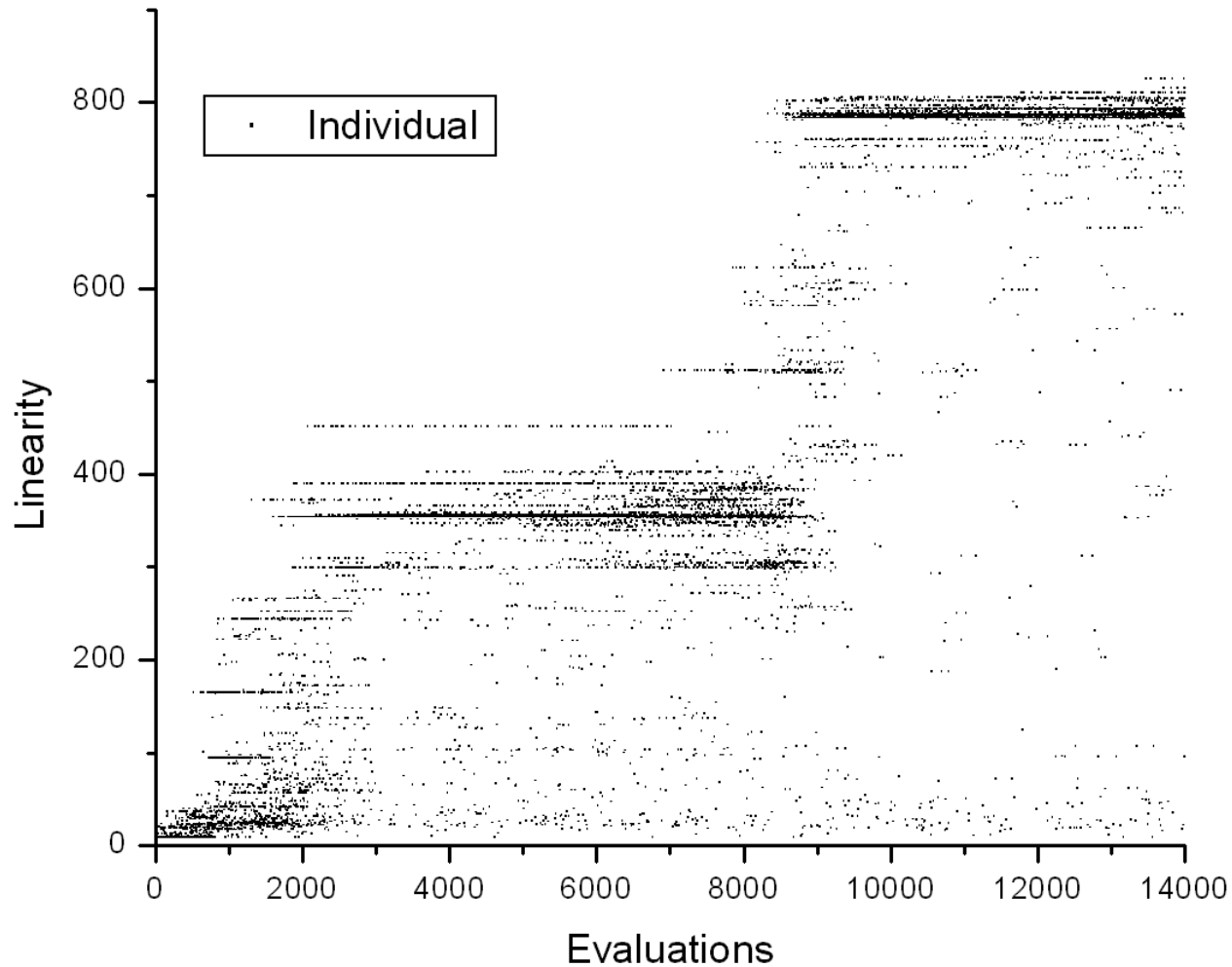
How to Draw a Straight Line Using a GP

- Used **GP with Top-down tree encoding and 2-bar or 4-bar embryo**
 - Population size: 100
 - Crossover 90%
 - Mutation 10% (Node positions, Operator types)
- Selection: **Stochastic Universal Sampling**
- Evaluation of an evolved straight-line mechanism
 - The mechanism is actuated at an arbitrary handle and the aspect ratios of bounding boxes of node trajectories are measured.
 - One node of the evolved machine on the left traces a curve that is linear to 1:5300 accuracy.
 - The evolved mechanism on the right traces a curve that is linear to 1:28340 accuracy

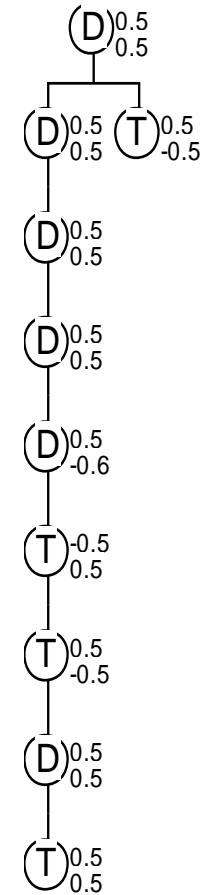
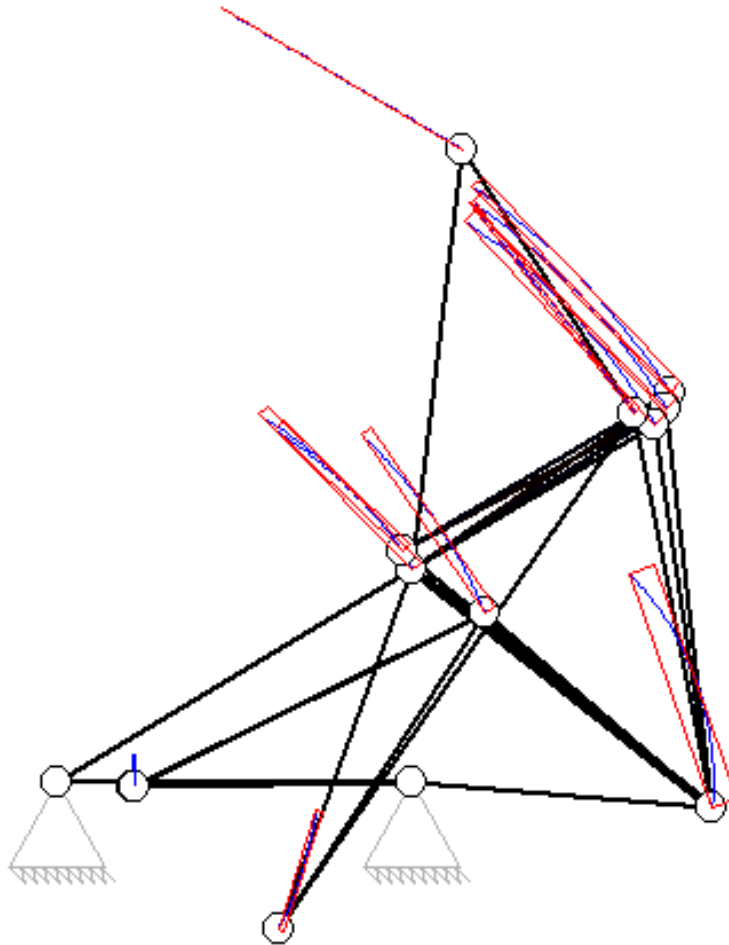


How to Draw a Straight Line Using a GP

- A typical run – each dot represents an evaluated individual

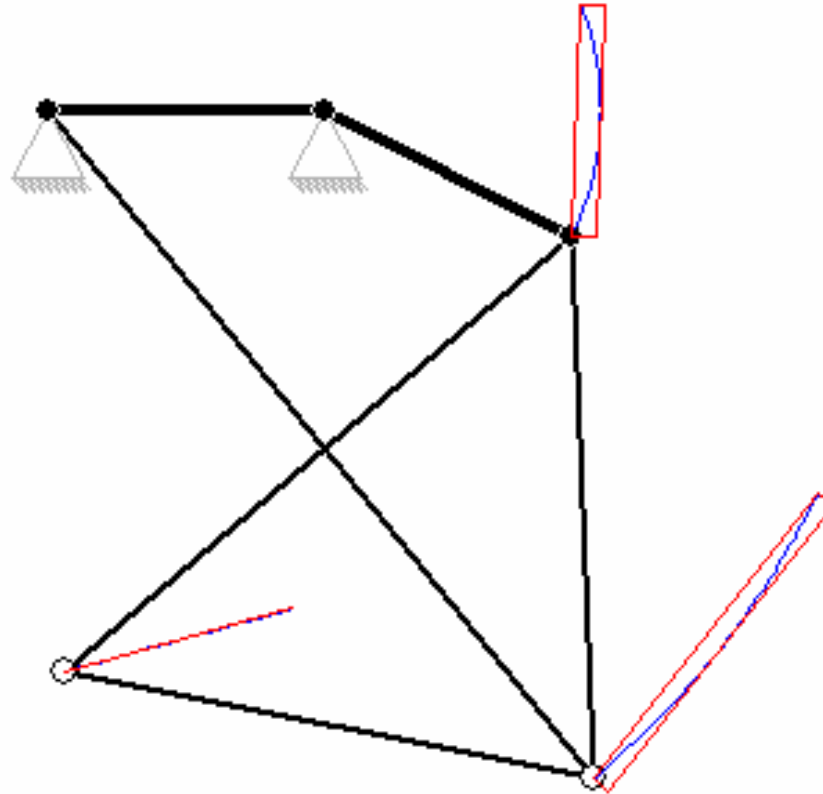


Some results



Linearity 1:4979

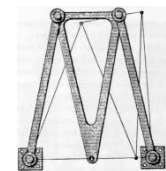
Some results



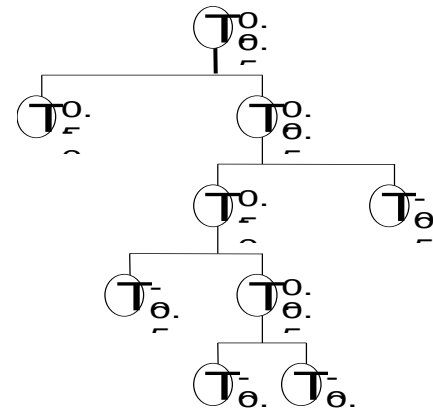
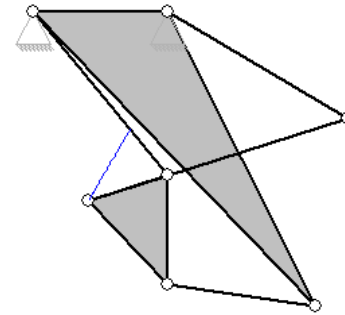
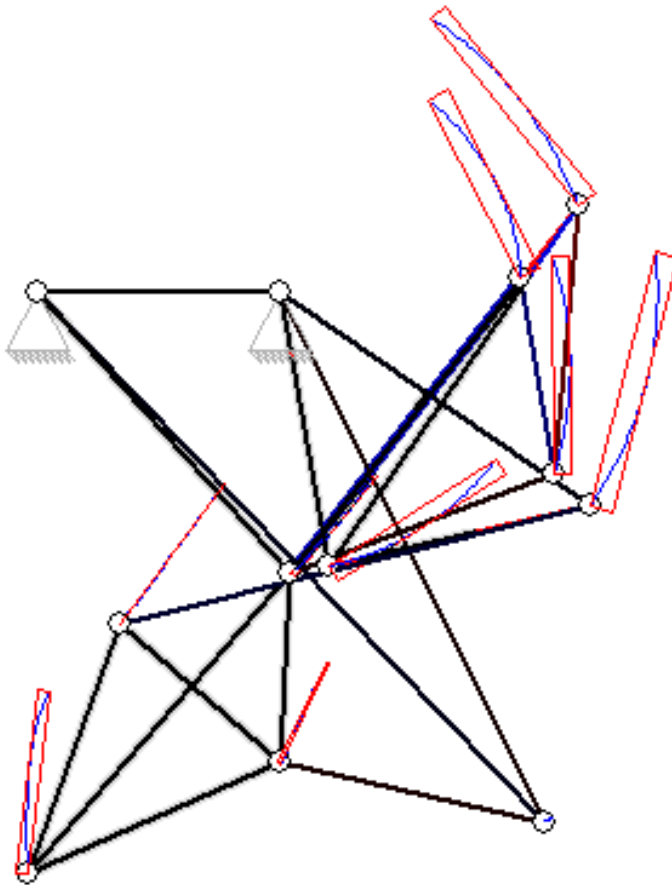
Linearity 1:5300

Infringes on Robert's Linkage (1841)

Published: Kempe A. B., (1877), *How To Draw A Straight Line*, London



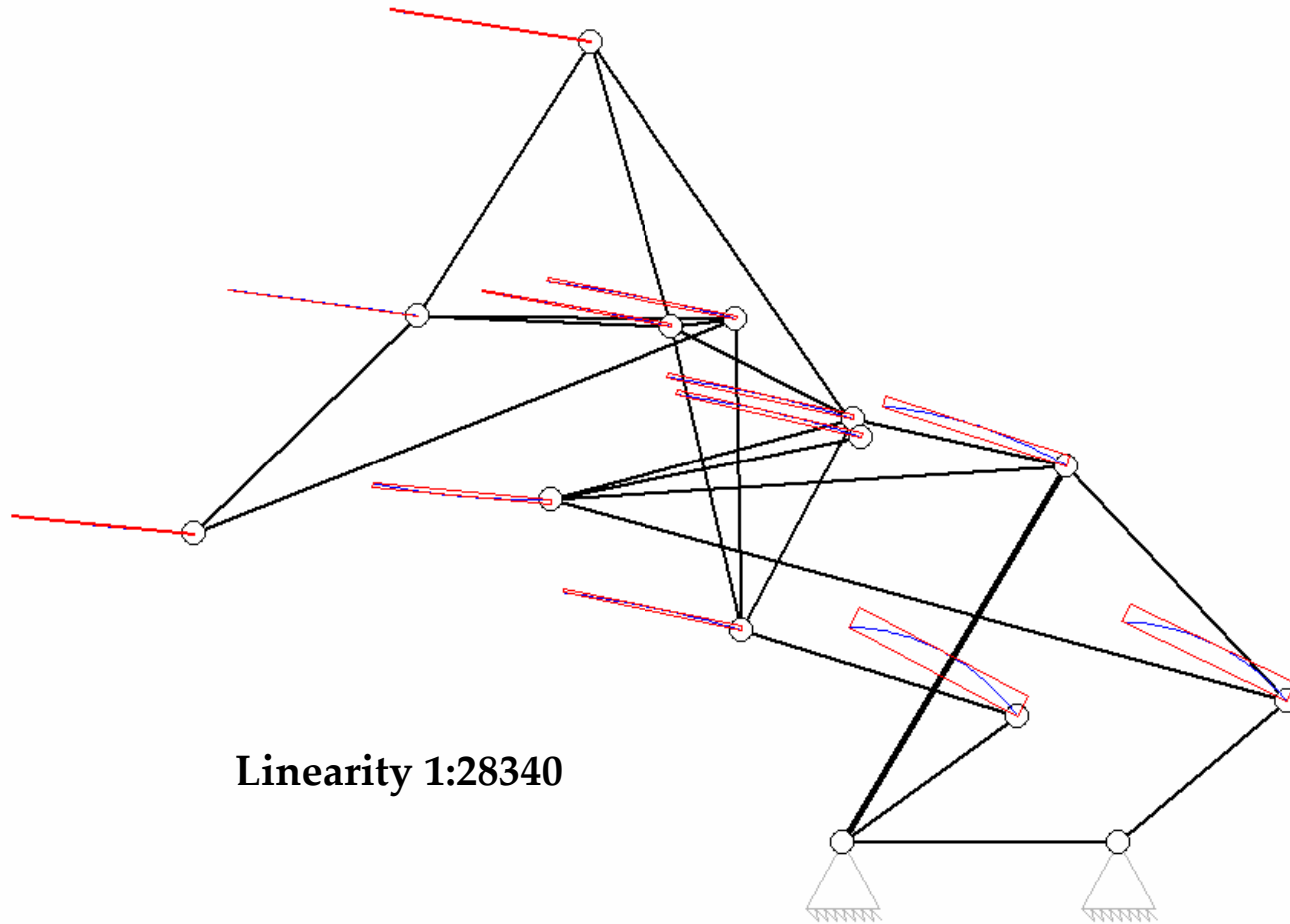
Some results



Linearity 1:12819



Some results



Linearity 1:28340

2005 Human-Competitive Awards in Genetic and Evolutionary Computation cont.

■ \$1000 – Silver

- Richard J. Terrile et al.:
 - Evolutionary Computation Technologies for the Automatic Design of Space Systems,
 - Evolutionary Computation applied to the Tuning of MEMS gyroscopes,
 - Multi-Objective Evolutionary Algorithms for Low-Thrust Orbit Transfer Optimization

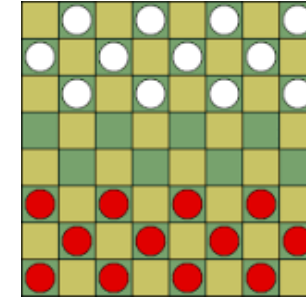
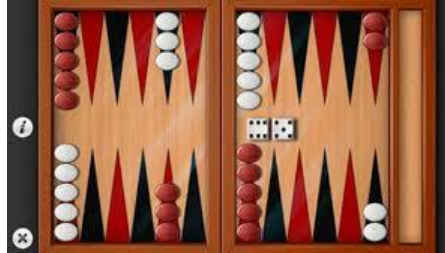
■ \$500 – Bronze

- Moshe Sipper et al.: Attaining Human-Competitive Game Playing with Genetic Programming (Backgammon Players, Robocode Players, Chess Endgame)
Moshe Sipper: Evolved to win
(<http://www.moshesipper.com/evolved-to-win.html>)
- Uli Grasmann, Risto Miikkulainen: Effective Image Compression using Evolved Wavelets

Moshe Sipper: Evolved to Win

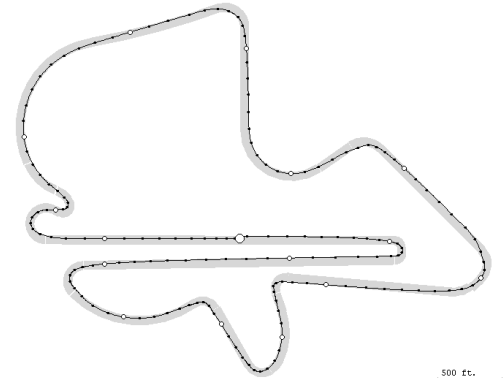
Board games

- Checkers
- Chess endgames
- Backgammon



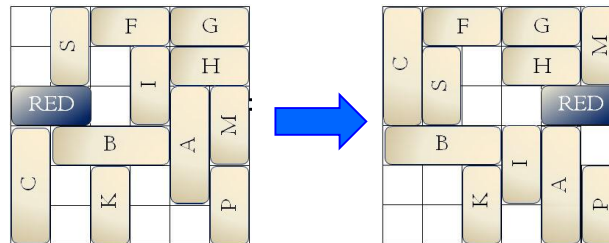
Simulation games

- Robocode
- Robot Auto Racing Simulator

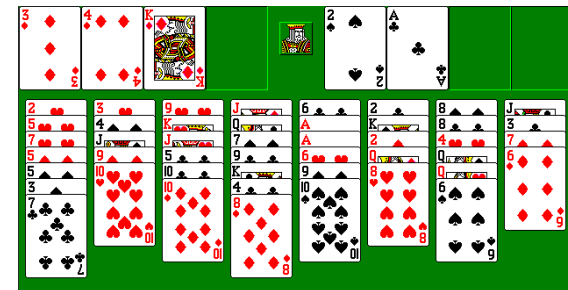


Puzzles

- Rush hour
- FreeCell



```
(% (% (% (% (IFG 0.702 AH AA (* NV -0.985)) (- AH (neg AH))) (-
(% 1.0 (% V AH)) (neg AH))) (- (- (* NV (neg NV)) (neg AH)) (neg
AH))) (- (% 1.0 (% V AH)) (neg (% (% 1.0 (% V AH)) (% V AH))))
```



Learning Game Strategies: FreeCell

GP used to evolve heuristics to guide staged deepening search for the hard game of FreeCell.

Trained and tested on **32,000 problems**—known as Microsoft 32K—all solvable but one.

FreeCell requires an enormous amount of search, due both to long solutions and to large branching factors.

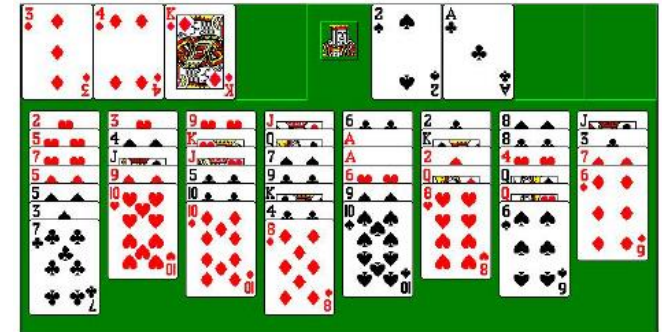
It remains out of reach for optimal heuristic search algorithms, such as variants of A*.

FreeCell remains intractable even when powerful enhancement techniques are employed, such as transposition tables and macro moves.

The previous top gun is the **Heineman's FreeCell solver**

- Heineman's staged deepening algorithm, based on a hybrid A* / hillclimbing search
- Heineman's heuristic

solved 96% of Microsoft 32K.



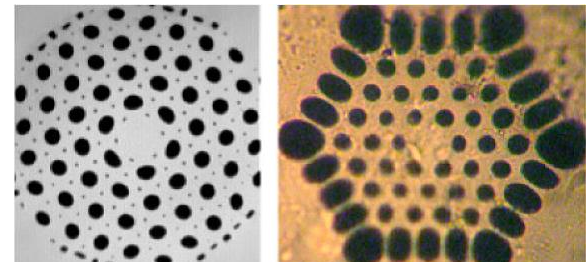
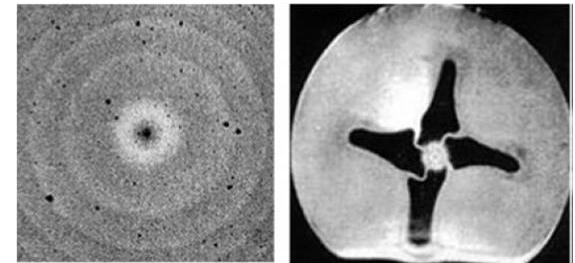
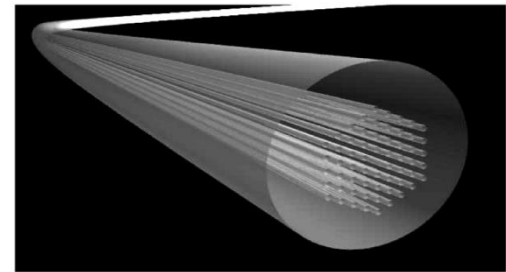
Evolutionary Design of Single-Mode Microstructured Polymer Optical Fibres

- Steven Manos, Leon Poladian, Maryanne Large: **Evolutionary Design of Microstructured Polymer Optical Fibres using an Artificial Embryogeny Representation**

reference: <http://www.genetic-programming.org/hc2007/cfe2007.html>

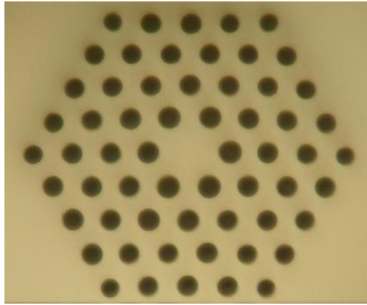
- Applications of optical fibres
 - Long distance telecommunications
 - Computer networks
 - Automotive and aeronautical
 - Electrical current measurement
 - Temperature and strain sensing
 - Medical (lasers and endoscopy)
- The behaviour of light depends on this internal structure

New functionality = more complex designs?

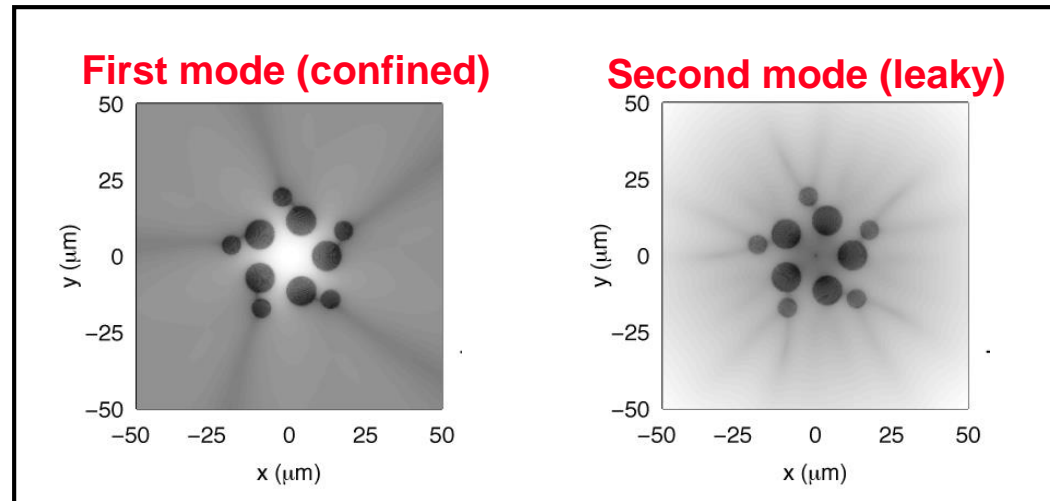


Single-moded fibres

Typical hexagonal design



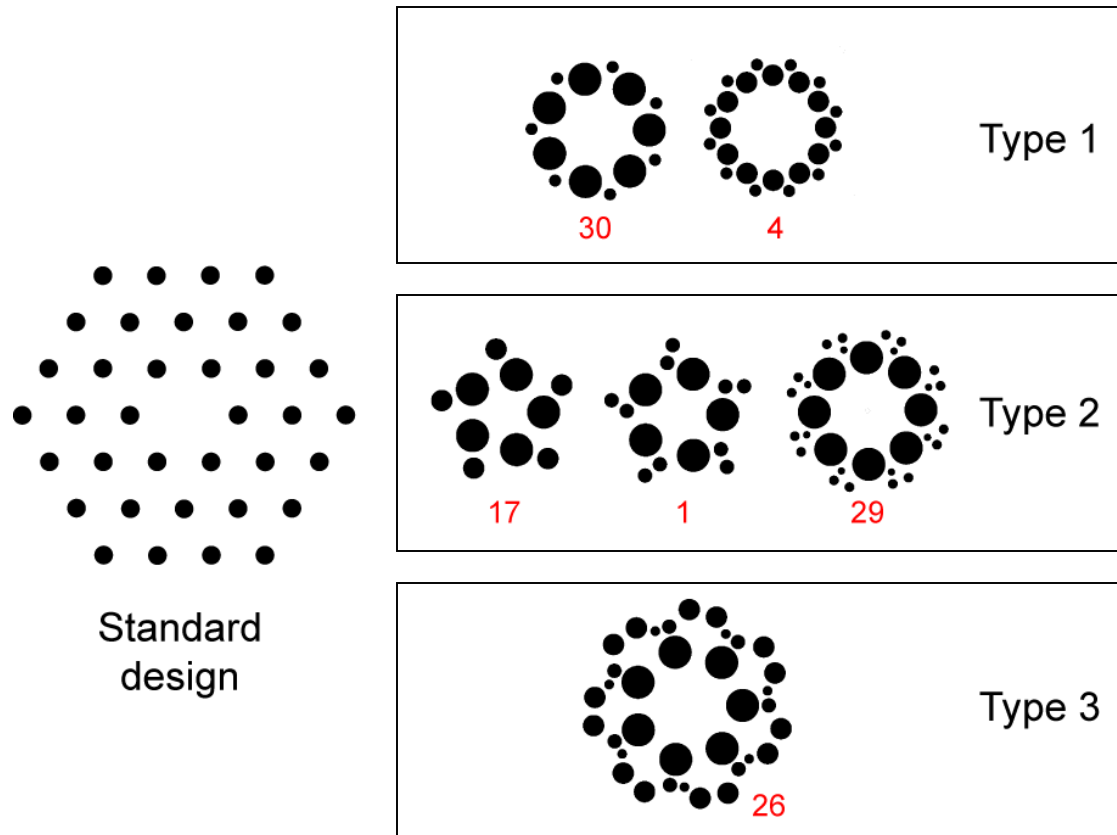
**Standard design since
the early 1990's**



Single-moded operation

- Single-moded fibres support the propagation of only the fundamental mode.
- These fibres are important in applications such as high-bandwidth communications, temperature sensing and strain sensing.
- By discovering fibres that don't have a typical hexagonal design, we can start doing more interesting things with them.

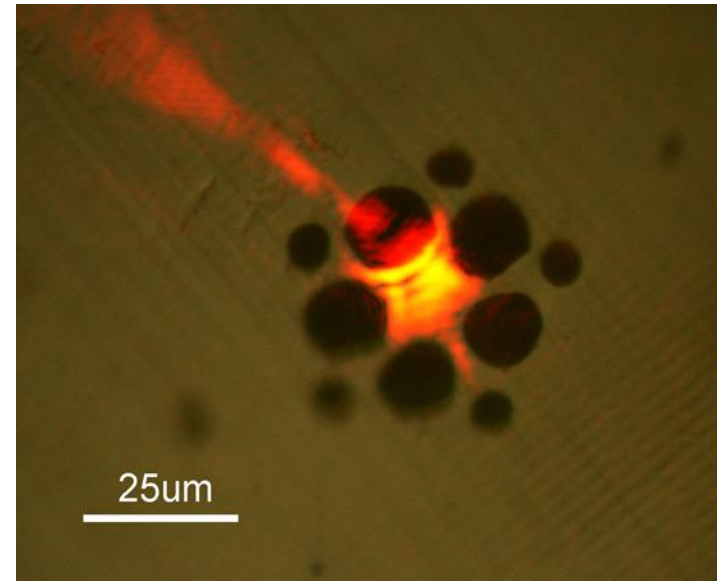
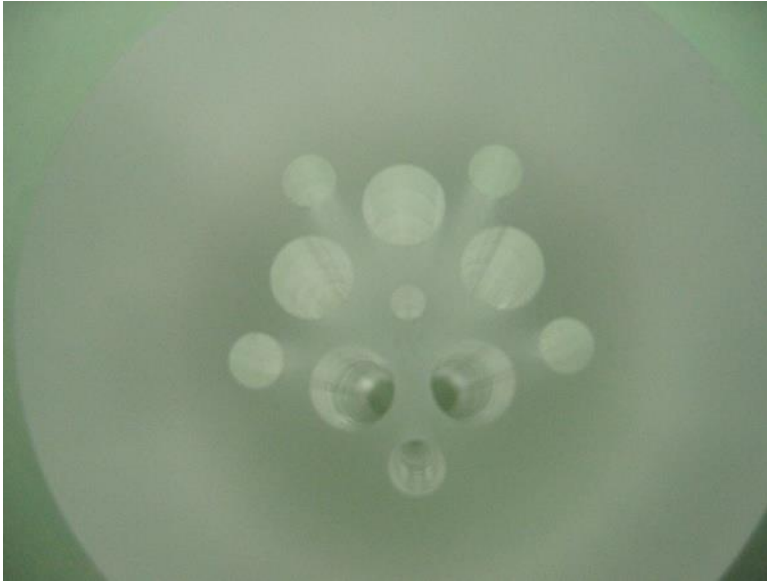
Evolved single-mode designs



All designs have confined fundamental modes with $l_{c,1} < 10^{-1}$ dB/m, with losses more typically being $l_{c,1} < 10^{-3}$ dB/m.
The loss of the second mode $l_{c,2} > 10^4$ dB/m in all cases.

All single-moded, yet phenotypically different.

Manufactured single-mode MPOF

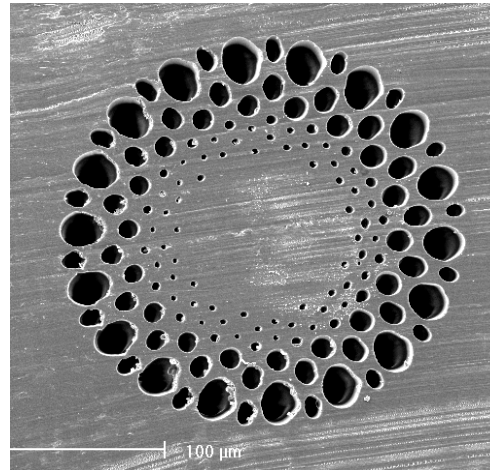
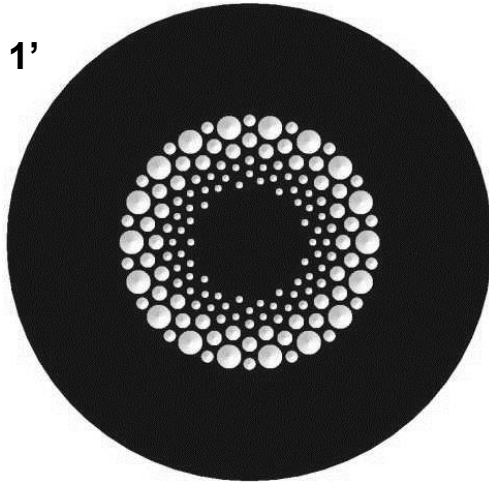


- Evolved designs are simpler than previous designs, and easier to manufacture.
- Provided us with a rich set of never before seen single-moded microstructured fibre designs to investigate further.

A different fitness function

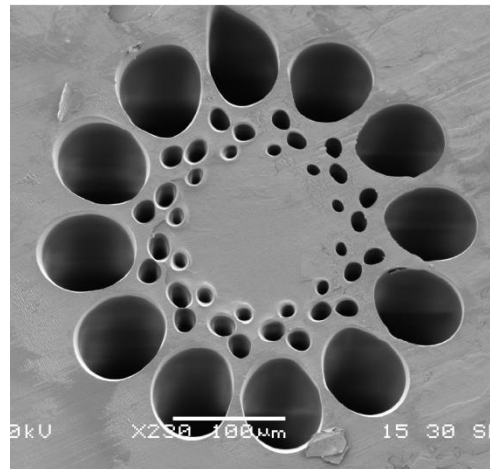
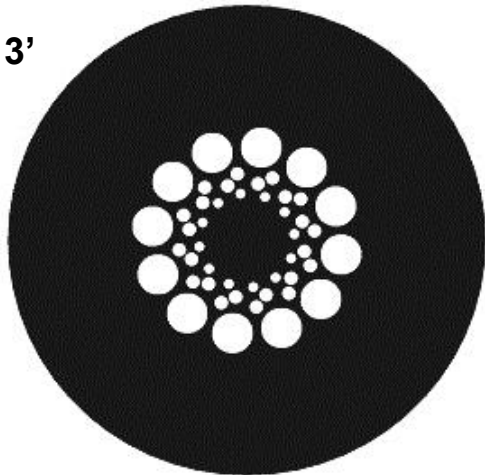
- **Highly multi-moded fibres** designed for use in LANs and other short-distance high-bandwidth applications.

'GIMP 1'



Hand-designed fibre

'GIMP 3'



Patented GA-designed
fewer holes, easier to
manufacture.

Automatically Finding Patches Using GP

Fully automated **GP-based** method for locating and repairing bugs in software

- Set of **testcases** consists of both
 - a set of **negative testcases** – that characterize a fault
 - A set of **positive testcases** that encode functionality requirements.
- Special GP representation of evolved repaired programs.
 - An **abstract syntax tree (AST)** including all of the statements in the program (CIL toolkit for manipulating C programs)
 - A **weighted path** through the program – a list of pairs [statement, weight] where the weight is based on that statement's occurrences in the testcases.
- Genetic operators are **restricted to AST nodes visited when executing the negative testcases**.
- Genetic operators realize **insertion, deletion, and swapping program statements and control flow**.
Insertions **based on the existing program structures** are favored.
- After a **primary repair that passes all negative and positive testcases** has been found, it is further **minimized w.r.t. the number of differences** between the original and repair program.

Automatically Finding Patches Using GP

- Example: Euclid's greatest common divisor

Weimer, W. et al.: Automatically Finding Patches Using Genetic Programming

Original program

```
1  /* requires: a >= 0, b >= 0 */
2  void gcd(int a, int b) {
3      if (a == 0) {
4          printf("%d", b);
5      }
6      while (b != 0)
7          if (a > b)
8              a = a - b;
9          else
10             b = b - a;
11     printf("%d", a);
12     exit(0);
13 }
```

Automatically Finding Patches Using GP

■ Example: Euclid's greatest common divisor

Weimer, W. et al.: Automatically Finding Patches Using Genetic Programming

Original program

```
1  /* requires: a >= 0, b >= 0 */
2  void gcd(int a, int b) {
3      if (a == 0) {
4          printf("%d", b);
5      }
6      while (b != 0)
7          if (a > b)
8              a = a - b;
9          else
10             b = b - a;
11     printf("%d", a);
12     exit(0);
13 }
```

Primary repair

```
1  void gcd_3(int a, int b) {
2      if (a == 0) {
3          printf("%d", b);
4          exit(0);           // inserted
5          a = a - b;        // inserted
6      }
7      while (b != 0)
8          if (a > b)
9              a = a - b;
10         else
11             b = b - a;
12     printf("%d", a);
13     exit(0);
14 }
```

generated given the bias towards modifying lines that are involved in producing the faults and the preference for insertions similar to existing code.

Automatically Finding Patches Using GP

■ Example: Euclid's greatest common divisor

Weimer, W. et al.: Automatically Finding Patches Using Genetic Programming

Original program

```
1  /* requires: a >= 0, b >= 0 */
2  void gcd(int a, int b) {
3      if (a == 0) {
4          printf("%d", b);
5      }
6      while (b != 0)
7          if (a > b)
8              a = a - b;
9          else
10             b = b - a;
11     printf("%d", a);
12     exit(0);
13 }
```

Primary repair

```
1  void gcd_3(int a, int b) {
2      if (a == 0) {
3          printf("%d", b);
4          exit(0);          // inserted
5          a = a - b;      // inserted
6      }
7      while (b != 0)
8          if (a > b)
9              a = a - b;
10         else
11             b = b - a;
12     printf("%d", a);
13     exit(0);
14 }
```

After repair minimization

generated given the bias towards modifying lines that are involved in producing the faults and the preference for insertions similar to existing code.

Automatically Finding Patches Using GP

- 10 different C programs of different size totaling 63,000 lines of code (LOC)

Program	LOC	Positive Tests	Path	Time	Initial Repair			Minimized Repair		
					fitness	Success	Size	Time	fitness	Size
gcd	22	5x human	1.3	149 s	41.0	54%	21	4 s	4	2
uniq	1146	5x fuzz	81.5	32 s	9.5	100%	24	2 s	6	4
look-u	1169	5x fuzz	213.0	42 s	11.1	99%	24	3 s	10	11
look-s	1363	5x fuzz	32.4	51 s	8.5	100%	21	4 s	5	3
units	1504	5x human	2159.7	107 s	55.7	7%	23	2 s	6	4
deroff	2236	5x fuzz	251.4	129 s	21.6	97%	61	2 s	7	3
nullhttpd	5575	6x human	768.5	502 s	79.1	36%	71	76 s	16	5
indent	9906	5x fuzz	1435.9	533 s	95.6	7%	221	13 s	13	2
flex	18775	5x fuzz	3836.6	233 s	33.4	5%	52	7 s	6	3
atris	21553	2x human	34.0	69 s	13.2	82%	19	11 s	7	3
average			881.4	184.7 s	36.9	58.7%	53.7	12.4 s	8.0	4.0

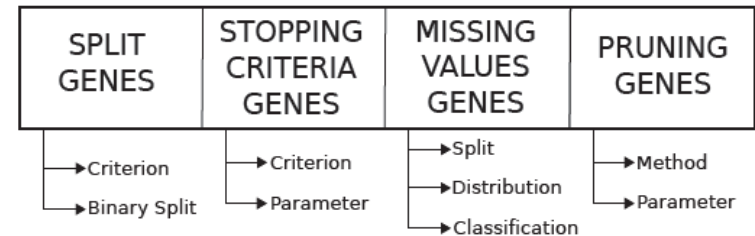
Weimer W. et al.: Automatically Finding Patches Using Genetic Programming

Forrest S. et al.: A Genetic Programming Approach to Automated Software Repair, GECCO 2009

Automatically Designing DT Algorithms

Hyper-heuristic Evolutionary Algorithm for Designing DT algorithms (HEAD-DT)

- automatically seeks for the best combination of the DT components
 - **Split**
 - Criterion – index of one of **15** splitting criteria (information gain, Gini, Mantaras, DCSCM, ...)
 - Binary split - indicates whether the splits of a DT are only binary or multi-way
 - **Stopping criteria**
 - Criterion – index of one of **5** strategies
 - Parameter of the selected strategy
 - **Missing values**
 - Split criterion evaluation – **4** strategies
 - Distribution – **7** strategies
 - Classification – **3** strategies
 - **Pruning**
 - Method – **5** strategies
 - Parameter



Automatically Designing DT Algorithms

Evolutionary Algorithm

- Population size: 100, Generations: 100, Tournament selection: $t = 2$, 1-point crossover: $pc = 0.9$, Uniform mutation: $pm = 0.1$, Elitism: 10

Example - algorithm designed for the Semeion data set – 1593 inst., 265 att.

-
- 1 Recursively split nodes with the Chandra-Varghese criterion;
 - 2 Aggregate nominal splits in binary subsets;
 - 3 Perform step 1 until class-homogeneity or the minimum number of 5 instances is reached;
 - 4 Perform MEP pruning with $m = 10$;

When dealing with missing values:

- 1 Calculate the split of missing values by performing unsupervised imputation;
 - 2 Distribute missing values by assigning the instance to all partitions;
 - 3 For classifying an instance with missing values, explore all branches and combine the results.
-

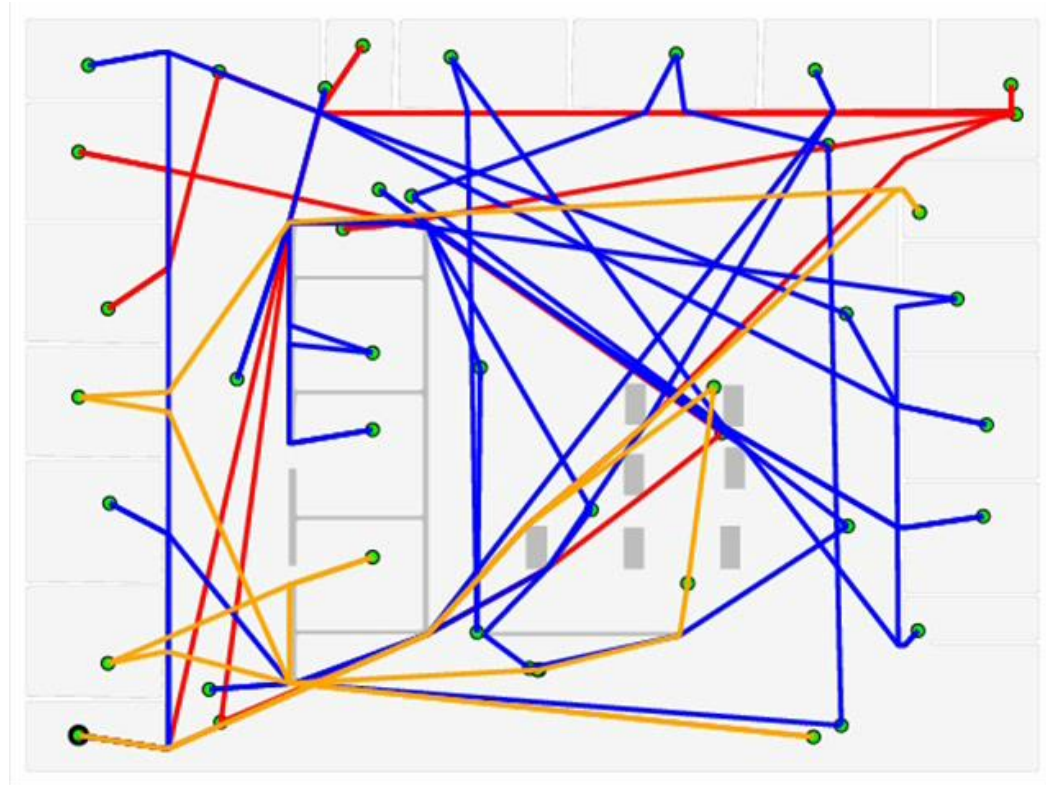
	CART	C4.5	HEAD-DT
Accuracy	0.94 ± 0.01	0.95 ± 0.02	1.0 ± 0.0
F-measure	0.93 ± 0.01	0.95 ± 0.02	1.0 ± 0.0
Tree size	34 ± 12.3	55 ± 8.27	19 ± 0.0

Very rare

Multiple Traveling Salesman Problem

MTSP – rescue operations planning

- Given N cities and K agents, find an optimal tour for each agent so that every city is visited exactly once.
- The optimization objective is to **minimize the overall time** spent by the squad (i.e. the slowest team member) during the environment exploration.



VIDEO

Evolutionary Algorithms for Dynamic Opt.

Dynamic optimizations - a class of problems whose specifications commonly termed

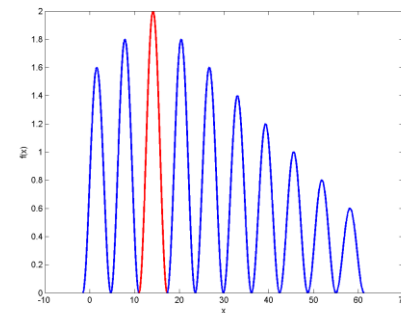
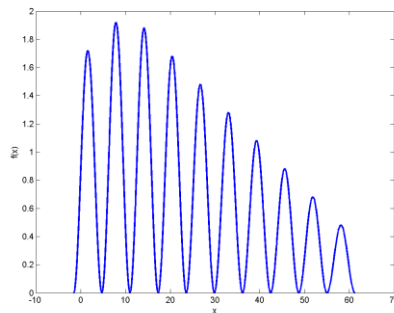
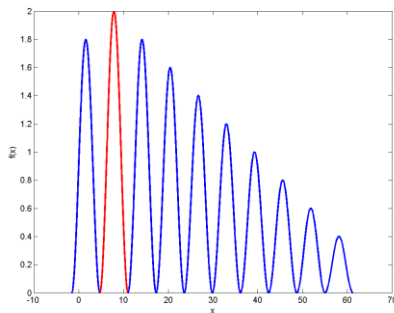
- optimization objectives, and/or
- problem-specific constraints, and/or
- environmental parameters and problem settings

change over time, resulting in **continuously moving optima**.

Ex.: Scheduling, manufacturing, trading with stochastic arrival of new tasks, machine faults, climatic change, market fluctuation, economic and financial factors.

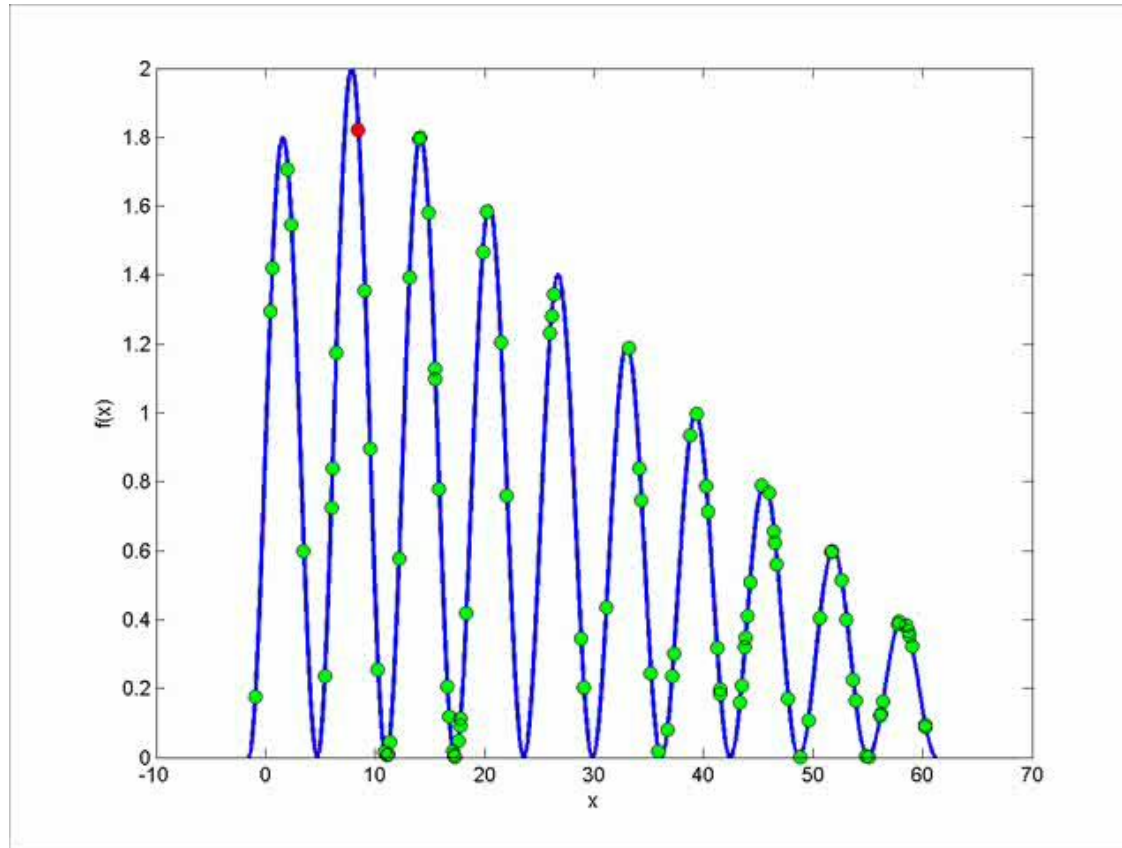
The goal of optimization in dynamic environment is to

- continuously track the optimum, or
- find robust solutions that perform satisfactorily even when the environmental parameters change.



Evolutionary Algorithms for Dynamic Opt.

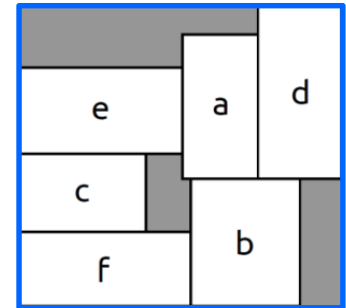
VIDEO



Floorplanning

- **Problem:** Floorplanning also known as 2D rectangle packing problem.
 - Given: A set of N unoriented blocks (rectangles) with fixed dimensions.
 - Goal: To place all blocks on a plane so that there is no overlap between any pair of rectangles and the bounding rectangular area is minimal (or the dead space is minimal).

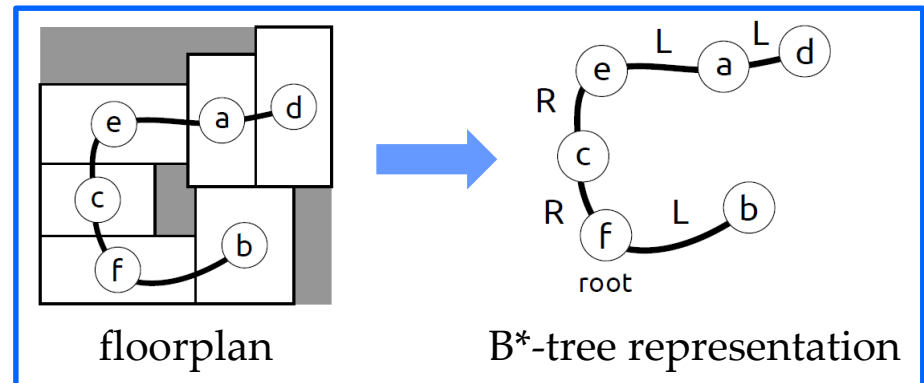
Ex.: Blocks {a, b, c, d, e, f} should be placed in a rectangular area so that the bounding rectangular area is minimal (or the dead space, shown in gray, is minimal).



- **Prototype** solution (floorplan) is encoded by B*-Tree non-slicing representation.

Each **tree** is expressed by a **linear string** of symbols in a prefix not.

Optimization works on linear structs.



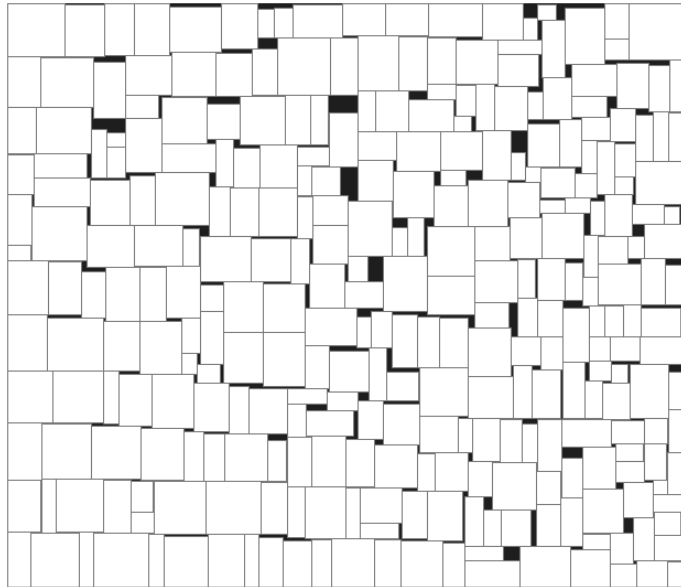
Floorplanning

- **Actions** (variation operators used as building blocks of the action sequences)
 - *Flip* – rotates a block of the specified tree node.
 - *Mirror* – swaps the left and right child of the specified tree node.
 - *Rotate* – recursively flips and mirrors child nodes of the specific node.
 - *ExchangeNodes* – exchanges blocks between two specified tree nodes.
 - *ExchangeSubtrees* – exchanges sub-trees rooted in two specified tree nodes.
 - *MoveSubtree* – places a specified node with its sub-tree to a new position in the tree.
- **Compared algorithms**
 - CompaSS - commercially available program.
 - B*-Tree/l , B*-Tree/SA – approaches using the same B*-tree representation searching the space of all possible floorplans by means of local search algorithm and simulated annealing, respectively.
- **Main achievements**
 - POEMS generates floorplans comparable and better than the ones generated by B*-Tree/l and B-Tree/SA algorithms.
 - POEMS clearly outperforms CompaSS algorithm w.r.t. the quality of the generated floorplans.

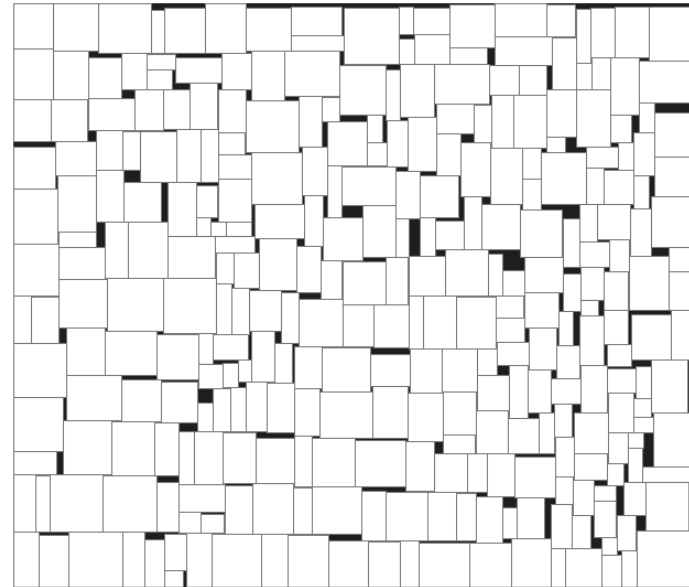
Floorplanning

- ▣ Examples of results obtained for 300 blocks.

3.8% dead space



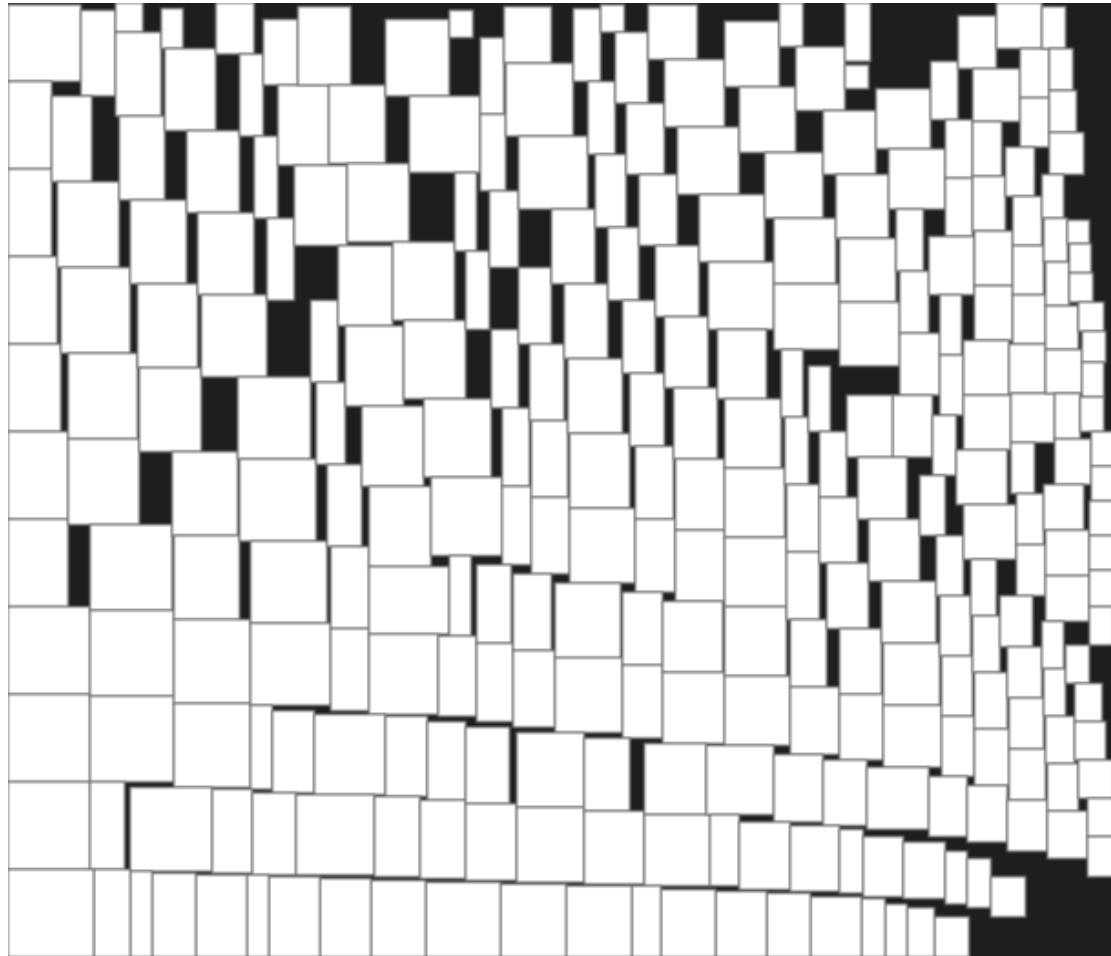
4.6% dead space



Hordějčuk, V.: Optimisation of Rectangular Shapes Placement by Means of Evolutionary Algorithms. Master thesis, Czech Technical University in Prague, Faculty of Electrical Engineering, 2011.

Floorplanning

Visualization of a POEMS run on data with 300 blocks.

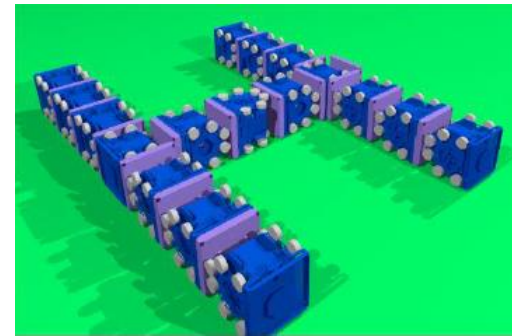
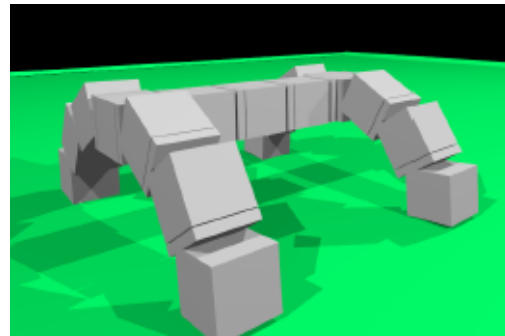
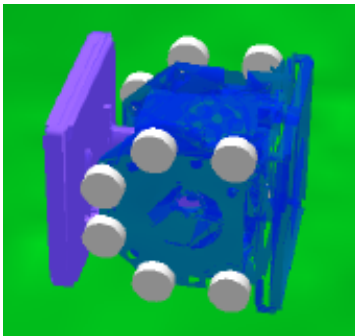


VIDEO

By V. Hordějčuk.

Evolution of Modular Robot Gaits

- ▣ Design of an effective and efficient evolutionary-based system for automated generating of modular robot gaits:
 - robots composed of a number of simple cubic-shaped robotic blocks,
 - each block is endowed with slots (three of them on the main body and one is on the movable arm) that enable them to connect to each other and form more complex robots,



- ▣ Approaches:
 - Co-evolution of a single leg motion pattern and a coordination strategy
 - HyperGP – neuroevolutionary-based approach
 - GP with automatically defined functions

Černý, J. and Kubalík, J.: Analysis of Co-evolutionary Approach for Robotic Gait Generation. In ECTA 2013.

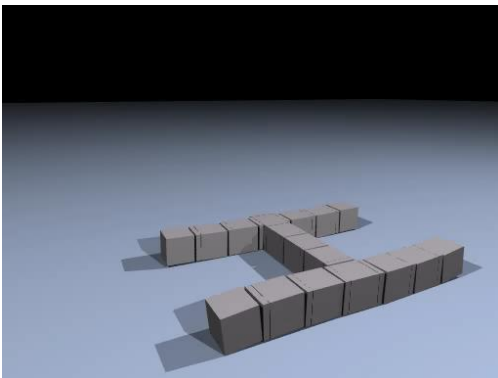
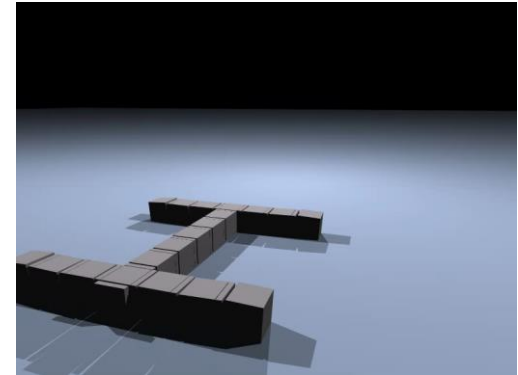
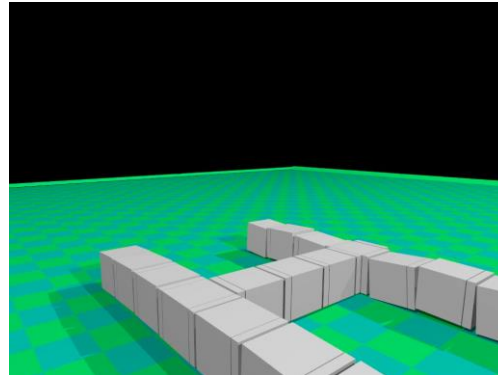
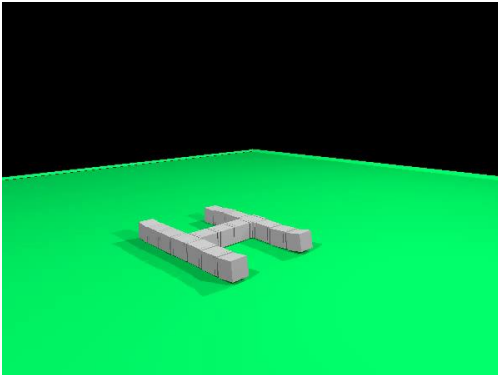
Černý, J. and Kubalík, J.: Co-evolutionary Approach to Design of Robotic Gait. In EvoRobot 2013.

Blvský, T.: Robustní genetické programování pro návrh řídicí strategie robotu. Diploma Thesis, ČVUT FEL, 2013

Dvorský, J.: Neuroevolutionary design of control strategy of a multi-legged robot. Bachelor Thesis, ČVUT FEL, 2013

Evolution of Modular Robot Gaits

- Design of an effective and efficient evolutionary-based system for automated generating of modular robot gaits:

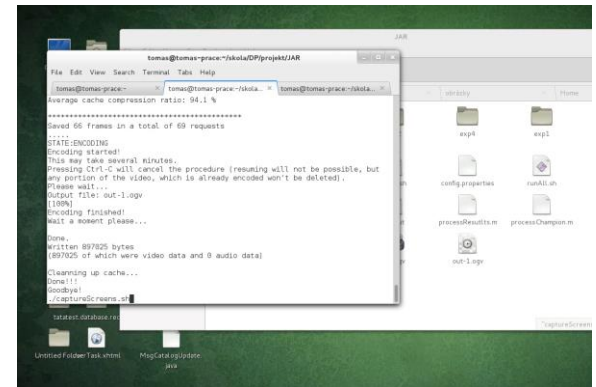


```
File Edit View Search Terminal Tabs Help
tomas@tomas-prace:~/kukla/DP/project$
rd: -1:1:par1:par2:par3:
add:mul:sub:div:sign:abs:adf3:

rd:col:par1:par2:par3:
add:mul:sub:div:sign:abs:adf3:

#####
0.88130342413095876
viewrfd.cpp:149::ln111: 0 -358.004 0
viewrfd.cpp:150::ln111: 0 0 0
viewrfd.cpp:151::ln111: 0 0 1
kuk.cpp:253::ln111:
38.0
#####
[tomas@tomas-prace:~/kukla$
Cached 5 MB, from 124 MB that were received.
Average cache compression ratio: 95.9 %
#####
Saved 34 frames in a total of 37 requests
Shutting down.....
STATE:ENCODING
Encoding started!
This may take several minutes.
Pressing Ctrl-C will cancel the procedure (resuming will not be possible, but
any portion of the video, which is already encoded won't be deleted).
Please wait...
Output file: out-2.ogv
[100%]
Encoding finished!
Wait a moment please...

Done.
Written 305414 bytes
(18544 of which were video data and 0 audio data)
Cleaning up cache...
Done!!
Goodbye!
```



GP for RL: Benchmark

1-DoF pendulum swing-up

- A weight of mass m attached to an actuated link that rotates in a vertical plane. The available torque is insufficient to push the pendulum up in a single rotation from the majority of initial states. It needs to be swung back and forth to gather energy, prior to being pushed up and stabilized.
- **State vector** is $[\alpha, \dot{\alpha}]$ – position and angular velocity
Angle α varies in $[-\pi, \pi]$
Control action u is limited to $[-2, 2]$ V
- The **continuous-time model** of the pendulum dynamics is

$$\ddot{\alpha} = \frac{1}{J} \cdot \left[mgl \sin(\alpha) - b\dot{\alpha} - \frac{K^2}{R} \dot{\alpha} + \frac{K}{R} u \right]$$

where

$$J = 1.91 \cdot 10^{-4} \text{ kgm}^2, m = 0.055 \text{ kg}, g = 9.81 \text{ ms}^{-2}, l = 0.042 \text{ m}$$

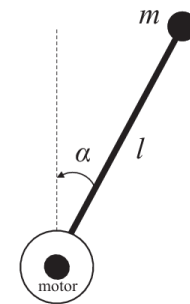
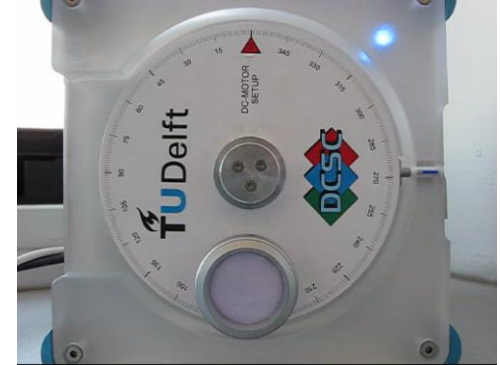
$$b = 3 \cdot 10^{-6} \text{ Nms/rad}, K = 0.0536 \text{ Nm/A}, R = 9.5 \Omega$$

The sampling period is $T_s = 0.01$ second.

The **discrete-time transitions** are obtained by numerically integrating the continuous-time dynamics using the fourth-order Runge-Kutta method.

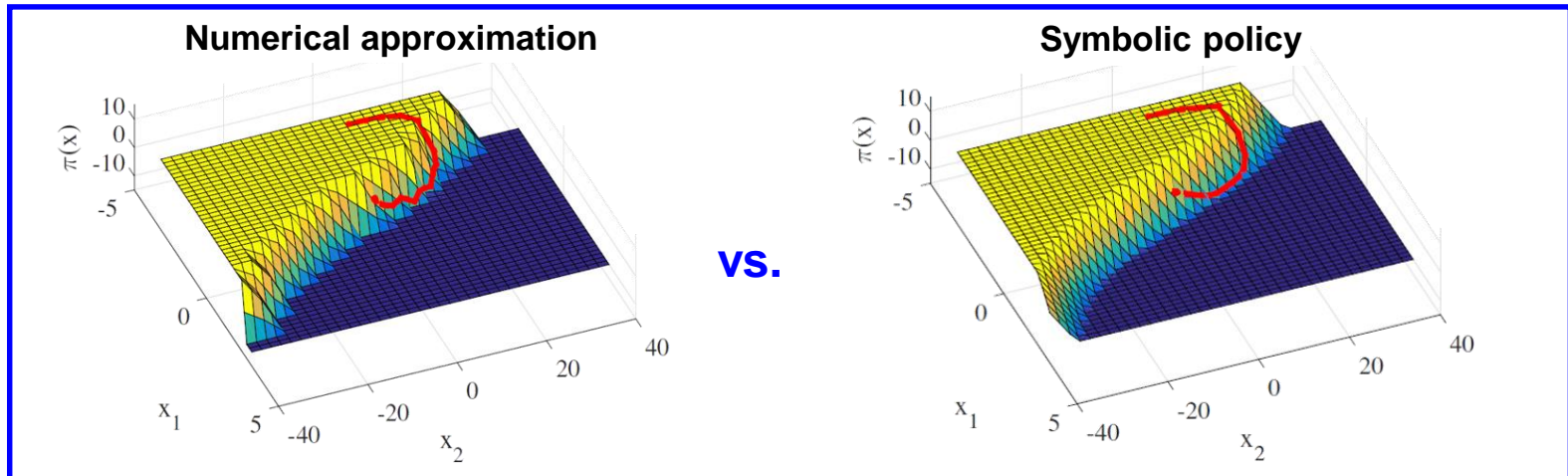
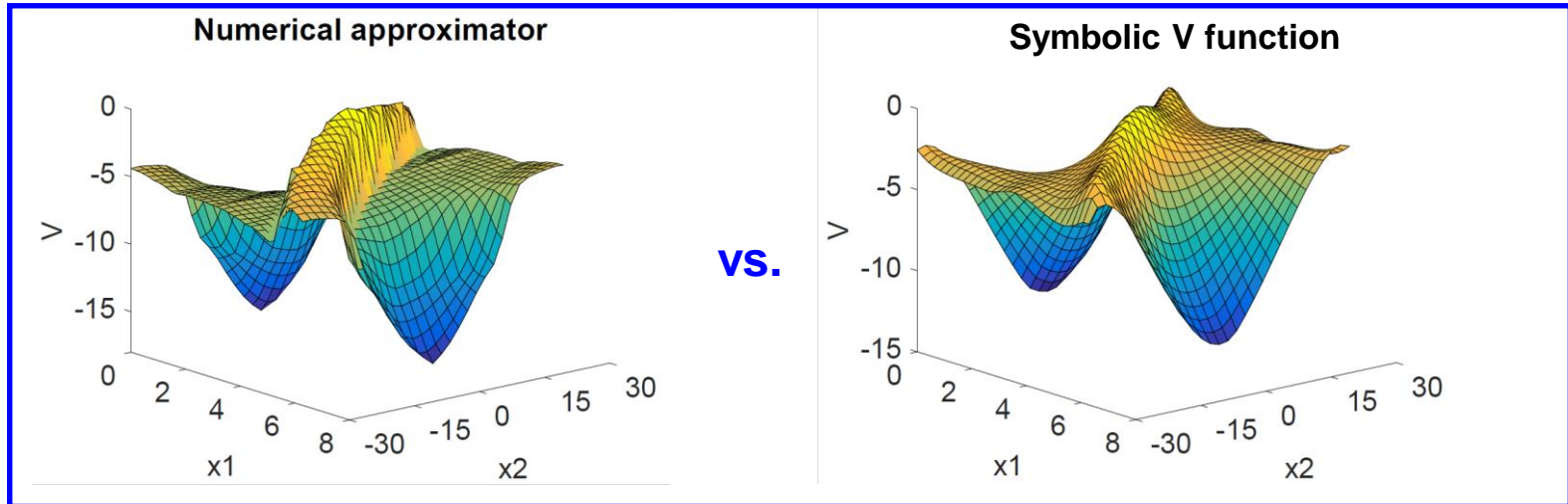
The **control goal** is to stabilize the pendulum in the unstable equilibrium

$$\alpha = \dot{\alpha} = 0$$



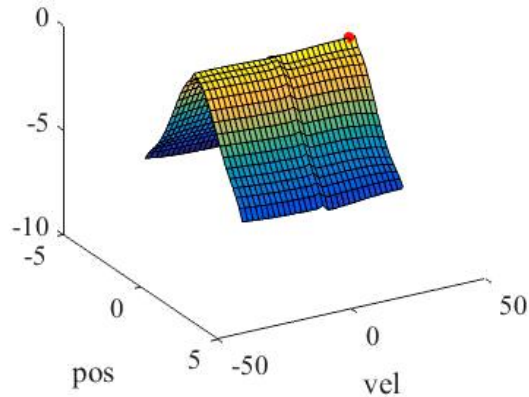
GP for RL: V function, policy

Task: Finding a **symbolic model from a set of discrete samples** of known numerical approximation of the V and policy functions.

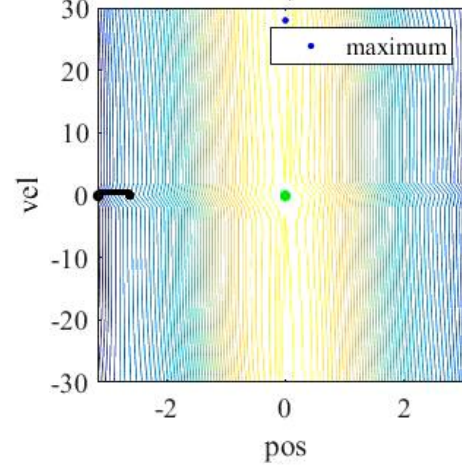


GP for RL: Learning V function

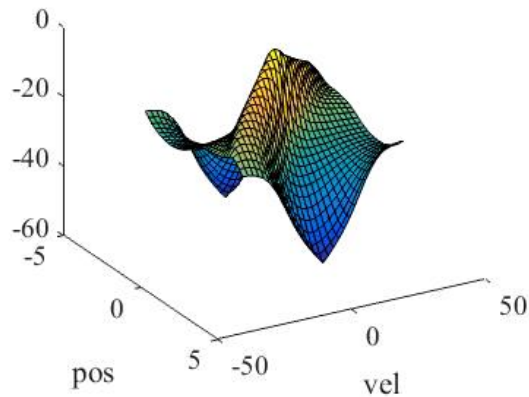
SNGP: model_1020_00004



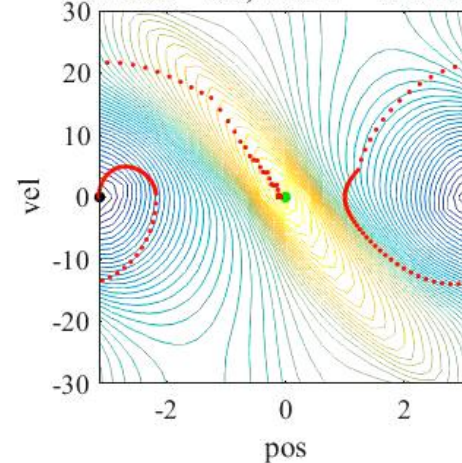
Vmax = -0.34031, Vmin = -6.2591



numerical V

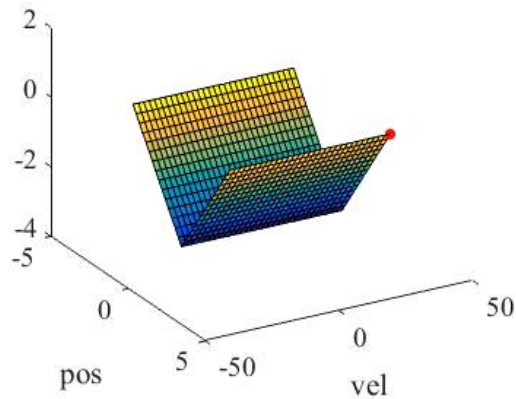


Vmax = 0.0, Vmin = -52.2405

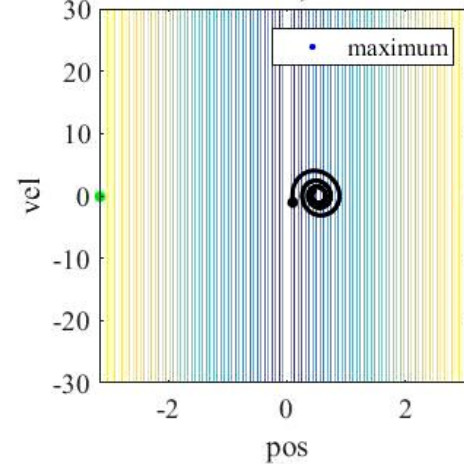


GP for RL: Learning V function

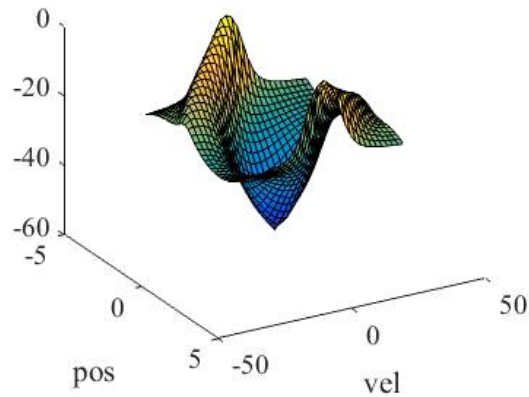
SNGP: model_2004_00002



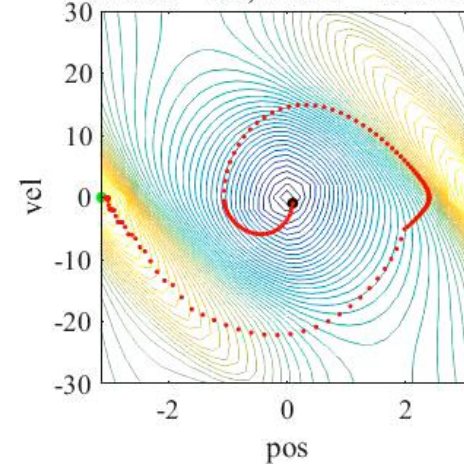
Vmax = 0.0001934, Vmin = -3.1405



numerical V



Vmax = 0.0, Vmin = -52.2405



GP for RL: Fitting Transition Model

Data-driven Construction of Symbolic Process Models for Reinforcement Learning

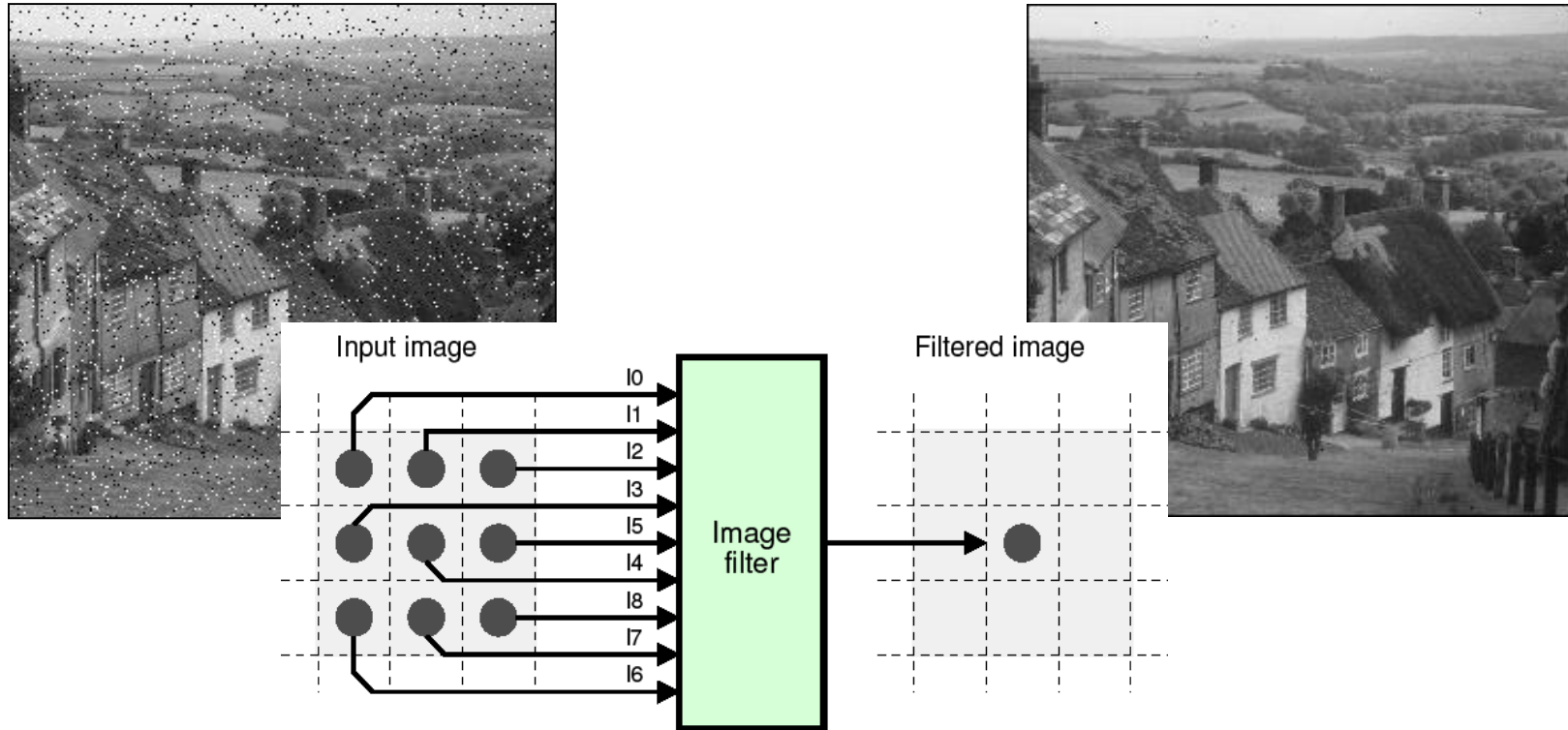
Pendulum Swing-up Experiments

Erik Derner, Jiří Kubalík & Robert Babuška

r.babuska@tudelft.nl



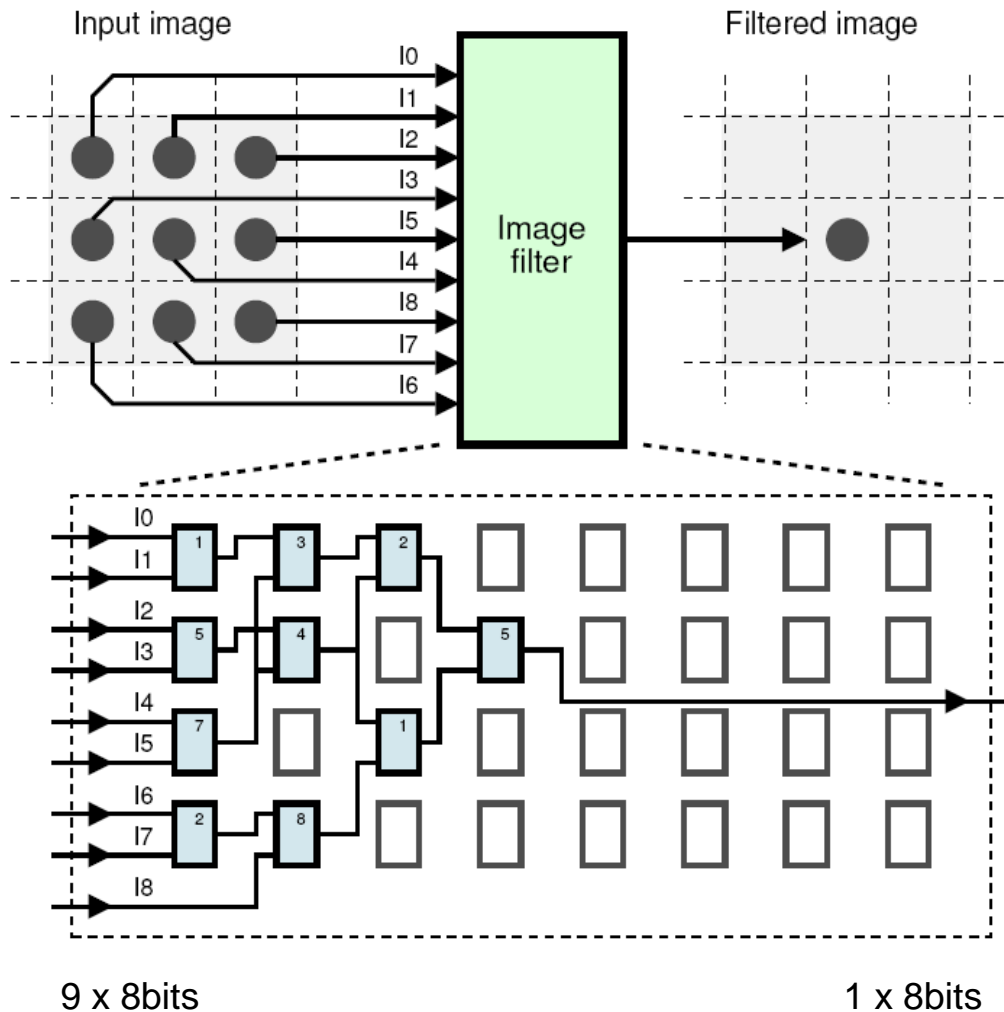
Evolutionary design of image filters



Can EA design an image filter which exhibits better filtering properties and lower implementation cost w.r.t. conventional solutions?

Target domain: filters suppressing shot noise, Gaussian noise, burst noise, edge detectors, ...

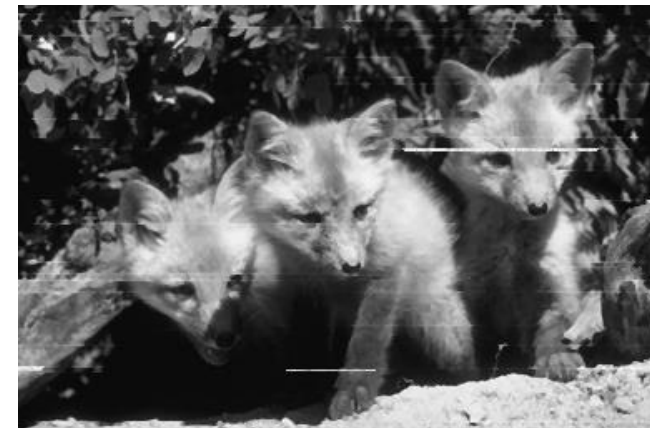
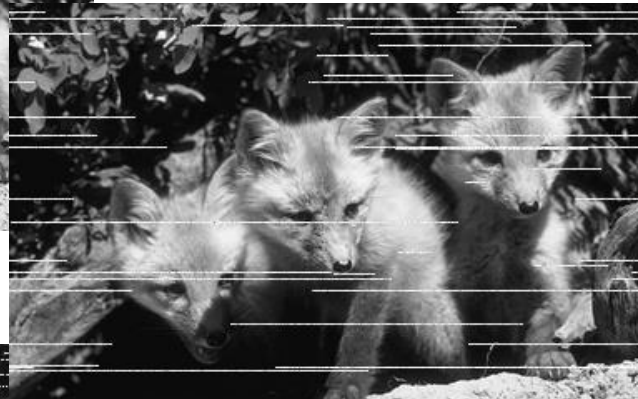
Image filter in CGP



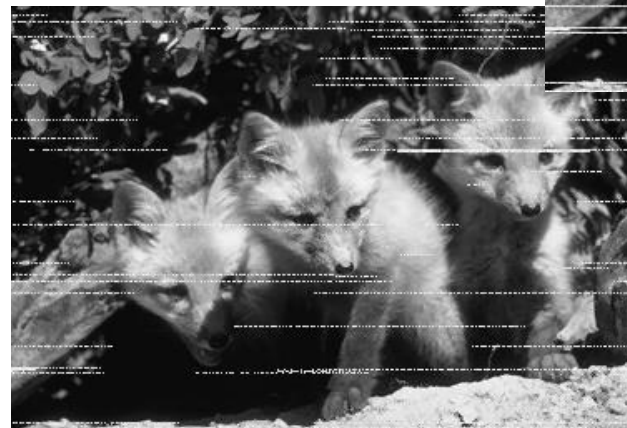
- Array of programmable elements (PE).
- No feedbacks.
- All I/O and connections on 8 bits.
- PE over 8 bits:
 - Minimum
 - Maximum
 - Average
 - Constants
 - logic operators
 - shift

Burst Noise Filtering

Image corrupted by 5% impulse bursts noise.



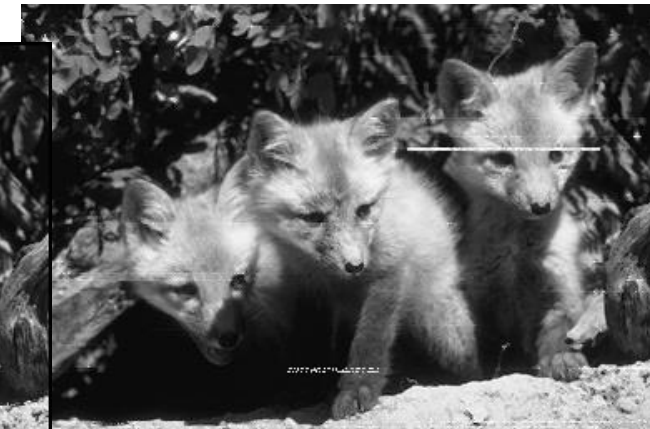
CWMF



AMF



evolved



PWMAD

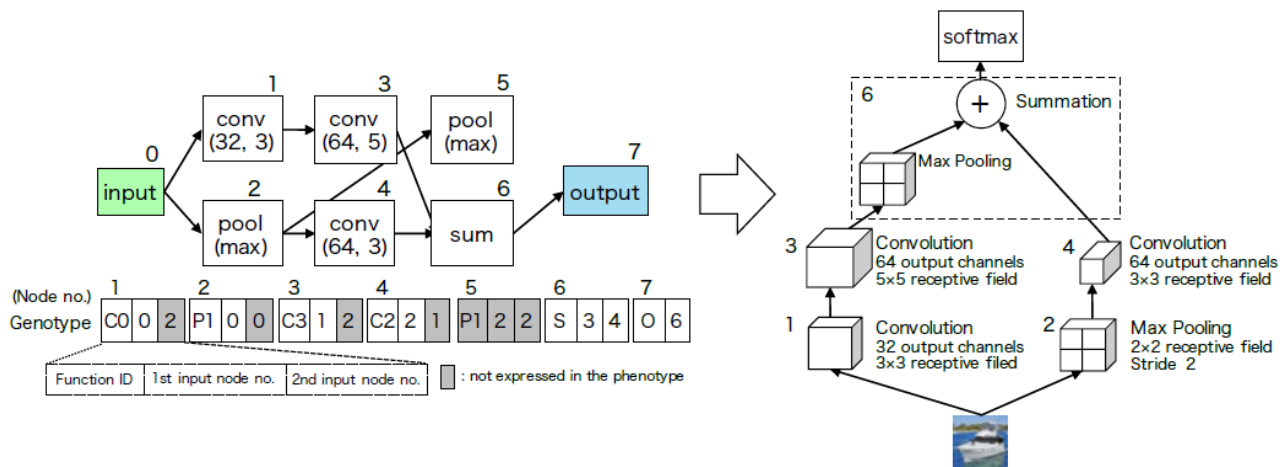
GP for CNN Architectures Design

Cartesian genetic programming (CGP) for automatic construction of **CNN architectures** for an image classification.

High-level functional modules as the SNGP nodes

- **ConvBlock** consists of standard convolution processing with a stride of 1 followed by batch normalization and rectified linear units (ReLU)
- **ResBlock** is composed of a convolution processing, batch normalization, ReLU, and tensor summation
- **max and average poolings**
- **concatenation** function concatenates two feature maps in the channel dimension
- **summation** performs the element-wise addition of two feature maps, channel by channel

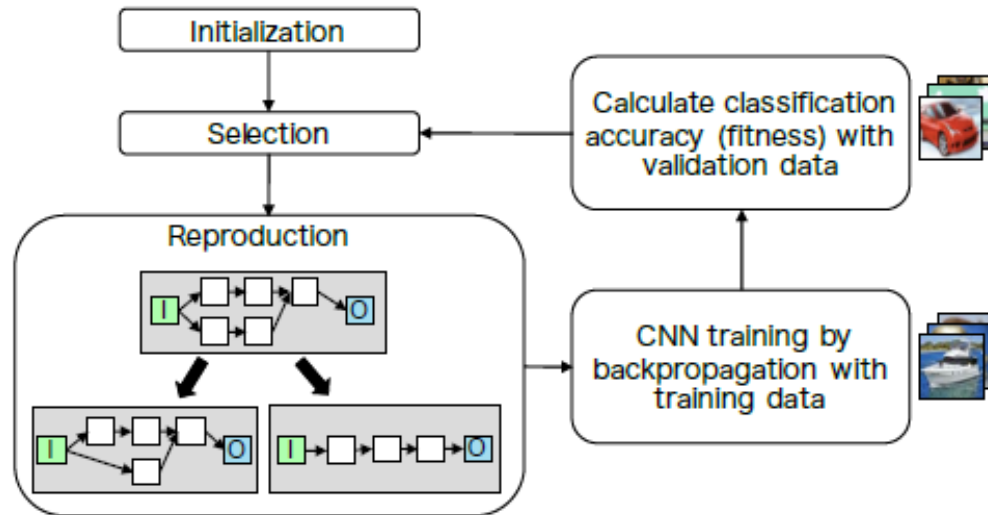
Output node represents the **softmax** function with the number of classes.



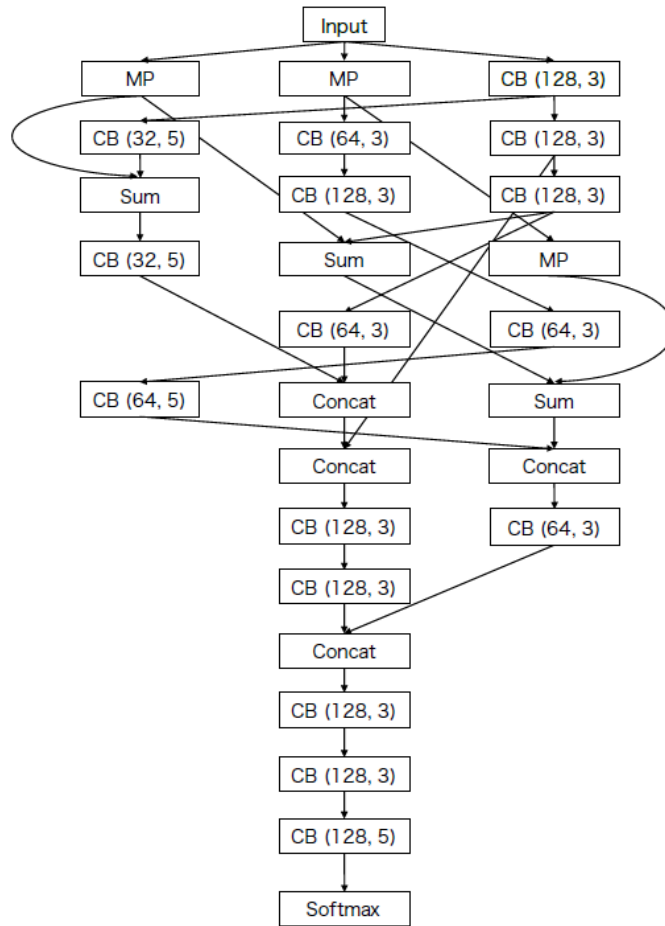
GP for CNN Architectures Design

Outline of the method

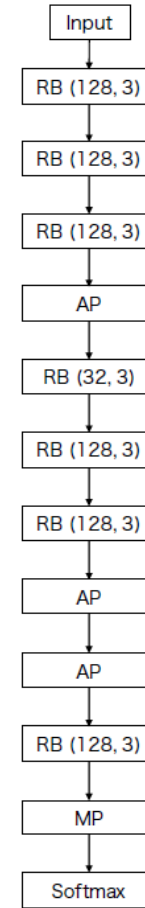
- CNN architecture are trained on training set
- and assigned the **validation accuracy** of the trained model **as the fitness**



GP for CNN Design: Examples



CGP-CNN (ConvSet)



CGP-CNN (ResSet)

Surprising Creativity of Digital Evolution

Elbow Walking, Cully et al. (2015)

- an algorithm that enables damaged robots to successfully adapt to damage
- how the evolution **solves the case where all six feet touch the ground 0% of the time**



Source: Lehman, J. et al.: The Surprising Creativity of Digital Evolution: A Collection of Anecdotes from the Evolutionary Computation and Artificial Life Research Communities. 2018

Surprising Creativity of Digital Evolution

Evolution of Muscles and Bones, Cheney et al. [68]

- evolution to discover from scratch the benefit of **complementary (opposing) muscle groups**, similar to such muscle pairs in humans, e.g. biceps and triceps – and also to place them in a functional way

Ever wonder what it would be like
to see evolution happening
right before your eyes?

Source: Lehman, J. at all.: The Surprising Creativity of Digital Evolution: A Collection of Anecdotes from the Evolutionary Computation and Artificial Life Research Communities. 2018

Surprising Creativity of Digital Evolution

Re-enabling Disabled Appendages, Ecarlat and colleagues [85]

- The goal was to accumulate a wide variety of controllers, to move the cube onto the table, to grasp the cube, to launch it into a basket in front of the robot, ...
- Then the robot's gripper was crippled, preventing it from opening and closing, ...



Source: Lehman, J. at all.: The Surprising Creativity of Digital Evolution: A Collection of Anecdotes from the Evolutionary Computation and Artificial Life Research Communities. 2018