Human-Competitive Results

- **Arthur Samuel**: "The aim of AI is … to get machines to exhibit behavior, which if done by humans, would be assumed to involve the use of intelligence".
- 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 1 January 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. As computer time becomes ever more inexpensive, researchers will start to routinely use genetic programming to produce useful new designs, generate patentable new inventions, and engineer around existing patents.

  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 ¾ *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.

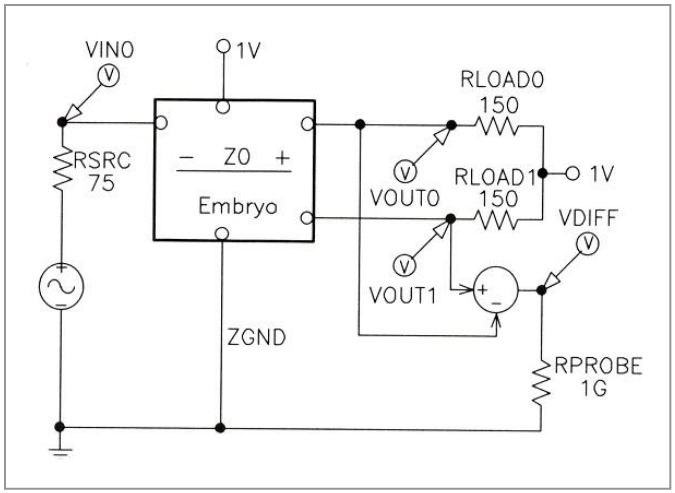
- Genetic programming used to synthesize both
  - the **structure/topology**, and
  - **sizing** (numerical component values)

  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

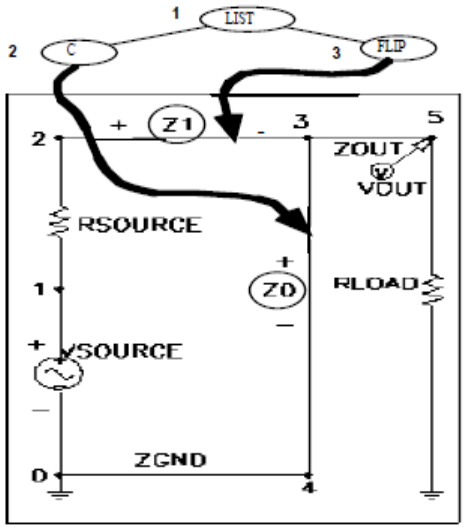
WYWIWYG: Embryonic Electrical Circuit

The Mapping between Electrical Circuits and Program Trees

- 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 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.
- The embryonic circuit used on a particular problem depends on the number of input signals and the number of output signals.

Example for an asymmetric bandpass filter:

- One input,
- One output
- Source and load resistors
- Two modifiable wires (Z0 and Z1) possessing writing heads (indicating components to be modified according to the program tree.) Other elements are fixed for ever.



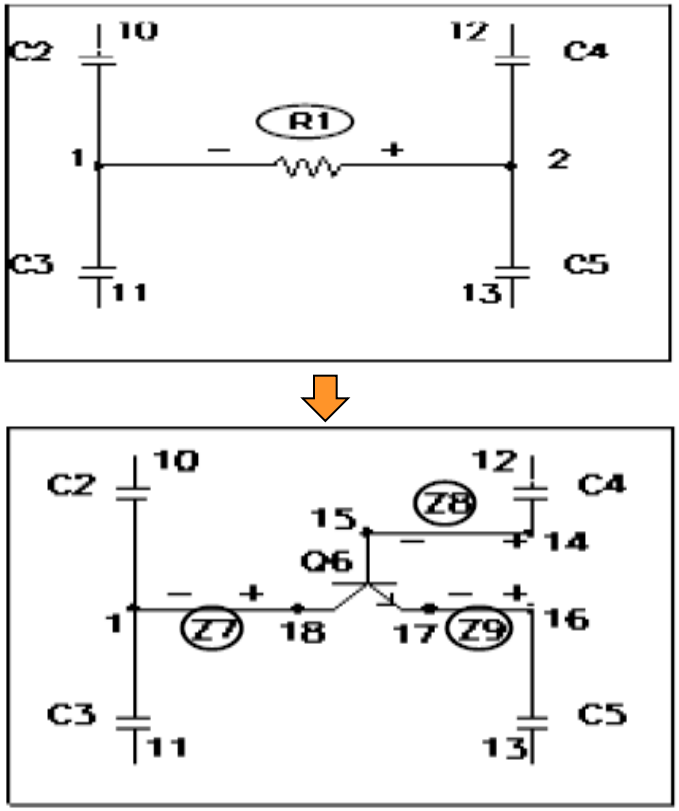
WYWIWYG: Component-Creating Functions

Each component-creating function (inductor-, capacitor-, …creating functions)

- inserts a component into the developing circuit and assigns component value(s)
- points to an associated highlighted component (i.e., a component with a writing head) in the developing circuit and modifies the highlighted component in some way.
- spawns one or more writing heads (through its construction-continuing subtrees)
  - The construction-continuing subtree of each component-creating function points to a successor function or terminal in the circuit-constructing program tree.
  - The arithmetic-performing subtree specifies the numerical value of the component

Example: Application of transistor-creating fn to the resistor R1

- R1 replaced with Q6
- 5 new nodes and 3 heads generated

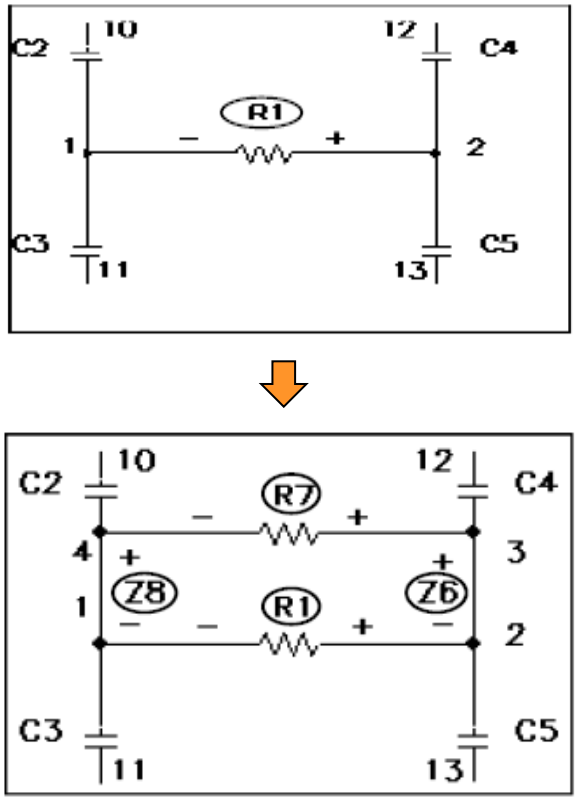


WYWIWYG: Connection-Modifying Functions

Each connection-modifying function

- points to an associated highlighted component and modifies the topology of the developing circuit in some way.
- spawns zero, one, or more writing heads.

- Examples:
  - FLIP - Polarity-reversing function
  - SERIES – series division function
  - PSS – parallel division function
  - DELTA – creates delta-shaped composition



Result of applying parallel division function to resistor R1

WYWIWYG: Function and Terminal Sets

Function set for arithmetic-performing subtree

- F_{aps} = {+, -}

Terminal set for arithmetic-performing subtree

- T_{aps} = {ERC}, floating point random constant

Function set for construction-continuing subtree

- F_{ccs} = {C, L, SERIES, PSS, GND, …}

Terminal set for construction-continuing subtree

- T_{ccs} = {END}

WYWIWYG: Fitness Assignment

- The evaluation of fitness for each individual circuit-constructing program tree 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.
- Each circuit is then simulated using SPICE (an acronym for Simulation Program with Integrated Circuit Emphasis) to determine its behavior.
- The fitness measure may incorporate any calculable 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.

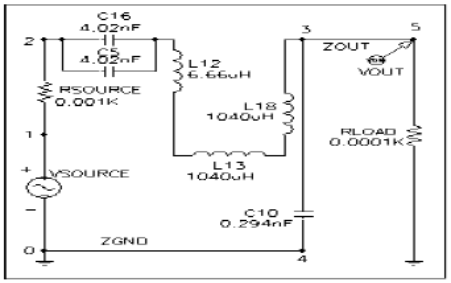
Asymmetric Bandpass Filter: Control Params

- Population size: 640,000
- Pcrossover = 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 with
  - deme size: 10,000
  - 64 demes
  - Migration rate: 2%

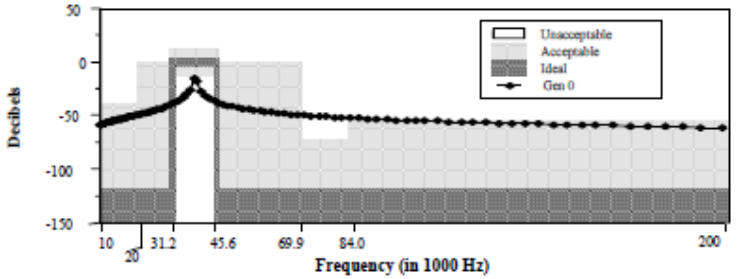
Asymmetric Bandpass Filter: Results

- 100%-complient program

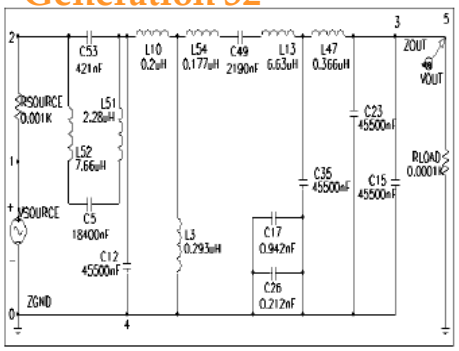
Generation 0



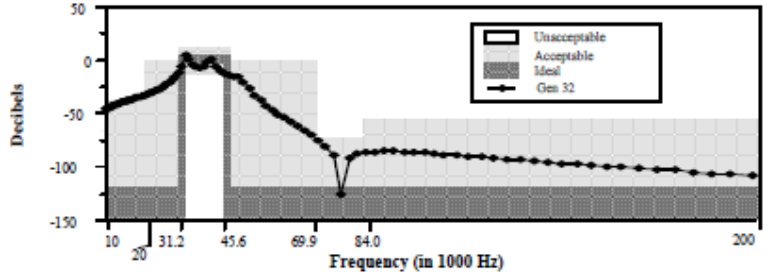
Generation 0



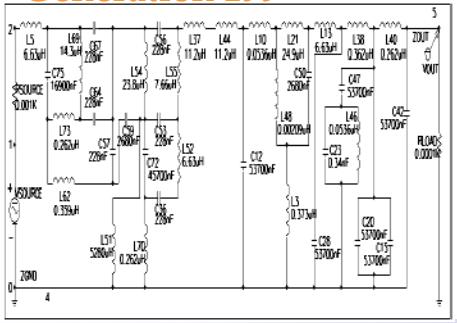
Generation 32



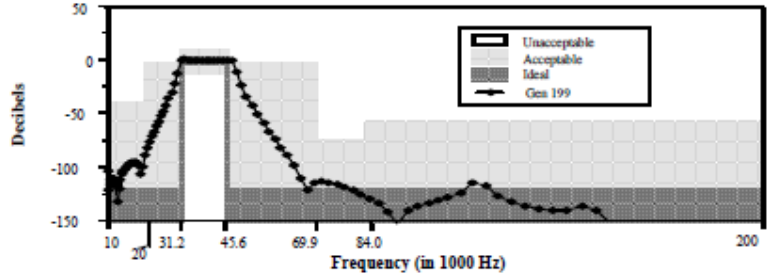
Generation 32



Generation 199



Generation 199



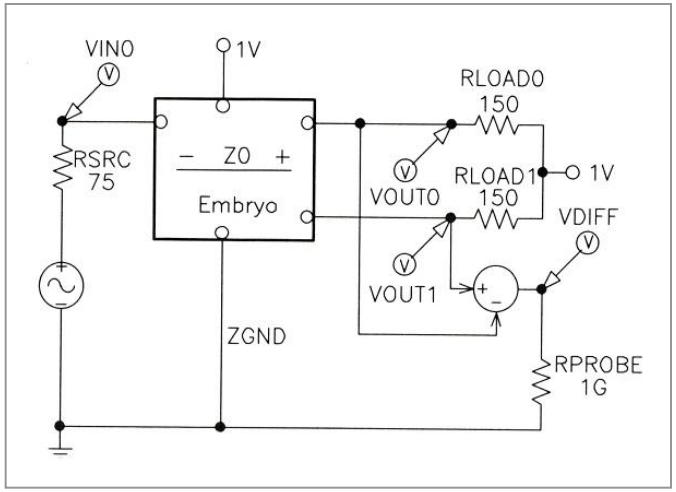
Low-Voltage Balun Circuit

- 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.

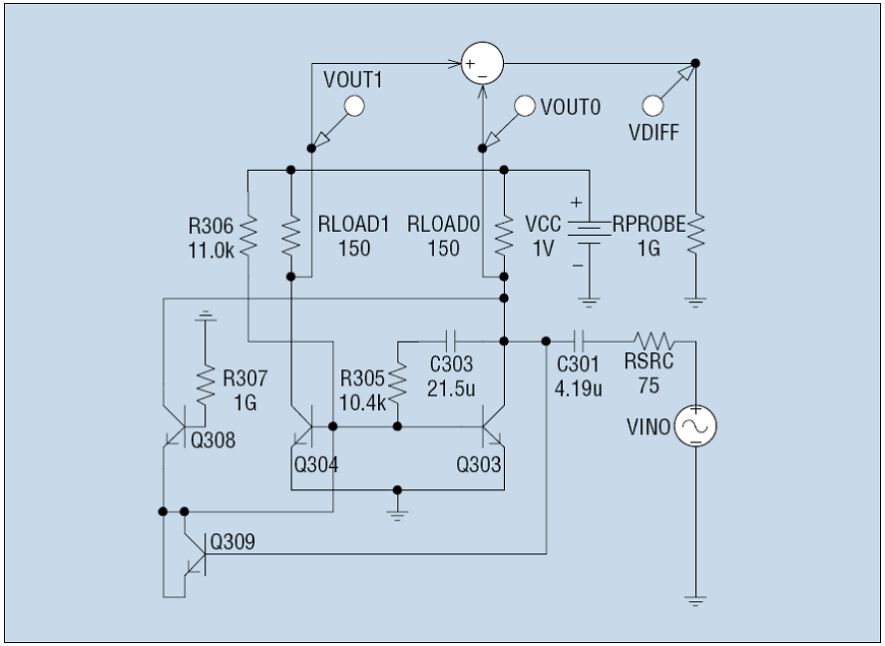
Genetically Evolved Low-Voltage Balun Circuit

- **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



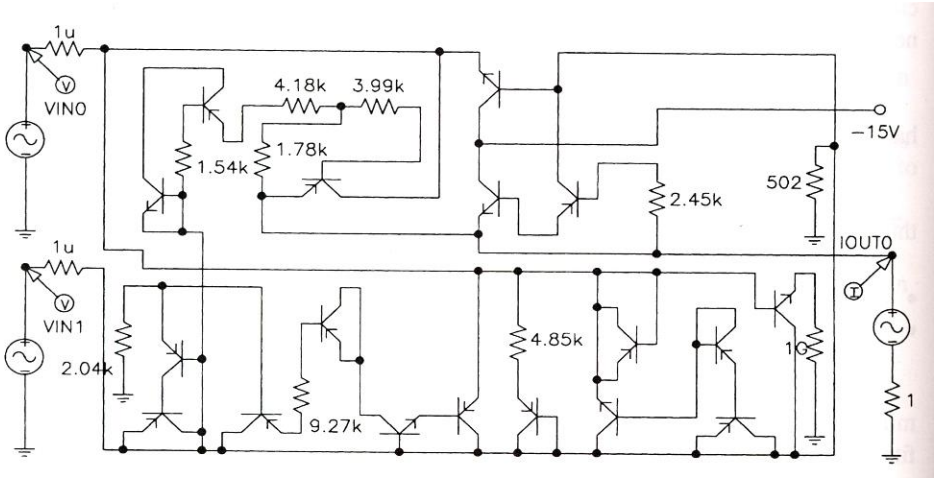
Evolved balun circuit



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

Voltage-Current Conversion Circuit

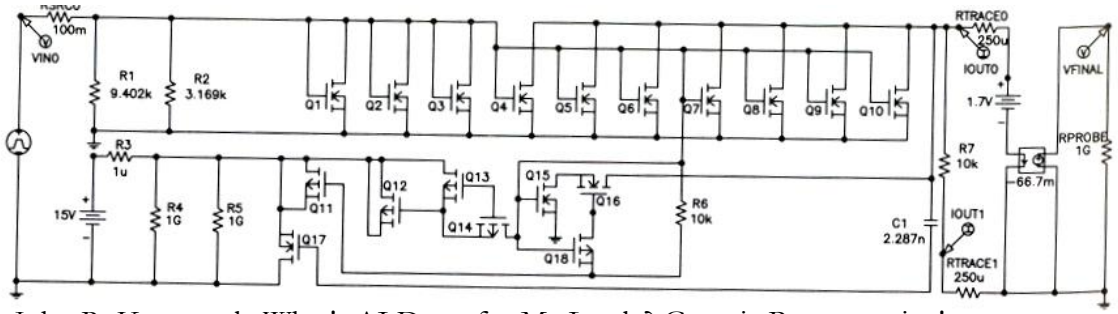
- **Voltage-current conversion circuit**'s purpose is to take two voltages as input and to produce as output a stable current whose magnitude is proportional to the difference between the voltages.
- **Fitness** measure is based on four time-domain input signals.
- **Genetically evolved circuit**
  - 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.

High-Current Load Circuit

- The **genetically evolved circuit shares some features found in the patented solution**
  - a variable, high-current, low-voltage, load circuit for testing a voltage source, comprising … a plurality of high-current transistors having source-to-drain paths connected in parallel between a pair of terminals and a test load.
  - However, the remaining elements of the genetically evolved circuit bear hardly any resemblance to the patented circuit.



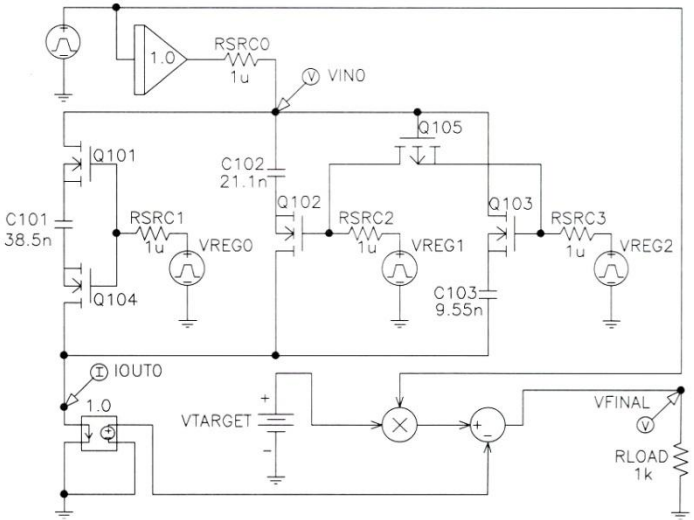
John R. Koza et al.: What's AI Done for Me Lately? Genetic Programming's Human-Competitive Results.

- **GP produced a circuit that duplicates the patented circuit's functionality using a different structure**.

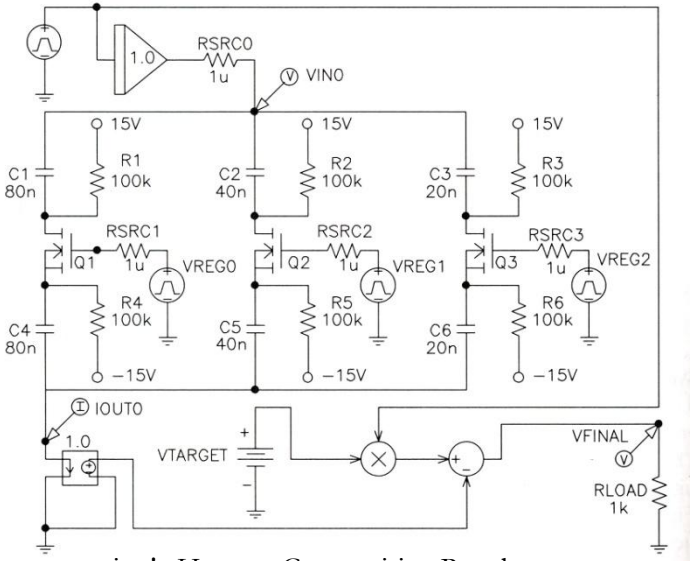
Mixed Analog-Digital Register-Controlled Variable Capacitor

- 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.

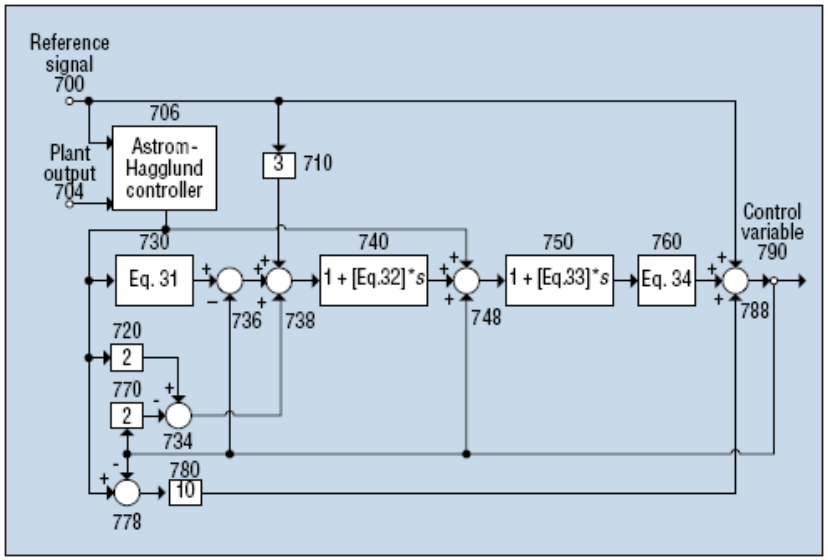
Automatic synthesis of PID controllers

- For the past six decades, most industrial users have relied on the tuning rules that John G.Ziegler and Nathaniel B. Nichols developed in 1942 to select the numerical parameters (for the gain of the controller's proportional, integrative, and derivative blocks and the reference signal's setpoint weighting) for the widely used PID type of controller.
- Karl J. Åström and Tore Hägglund improved upon these rules in 1995.
- **Genetic programming synthesized tuning rules for PID controllers that outperform these existing rules**.

  Koza et al. applied for a patent on these new improved tuning rules and the improved non-PID controller described next. If a patent is granted (as expected), it should be the first one for an invention that genetic programming created.

Automatic synthesis of PID controllers cont.

- Genetic programming can automatically create, in a single run, a general (parameterized) solution to a problem in the form of a graphical structure whose edges represent components and where the parameter values of the components are specified by mathematical expressions containing free variables.
- For example, genetic programming recently synthesized both the topology and sizing of an improved general-purpose non-PID controller that outperforms the general-purpose 1995 Astrom-Hagglund PID controller and the widely used Ziegler-Nichols tuning rules.

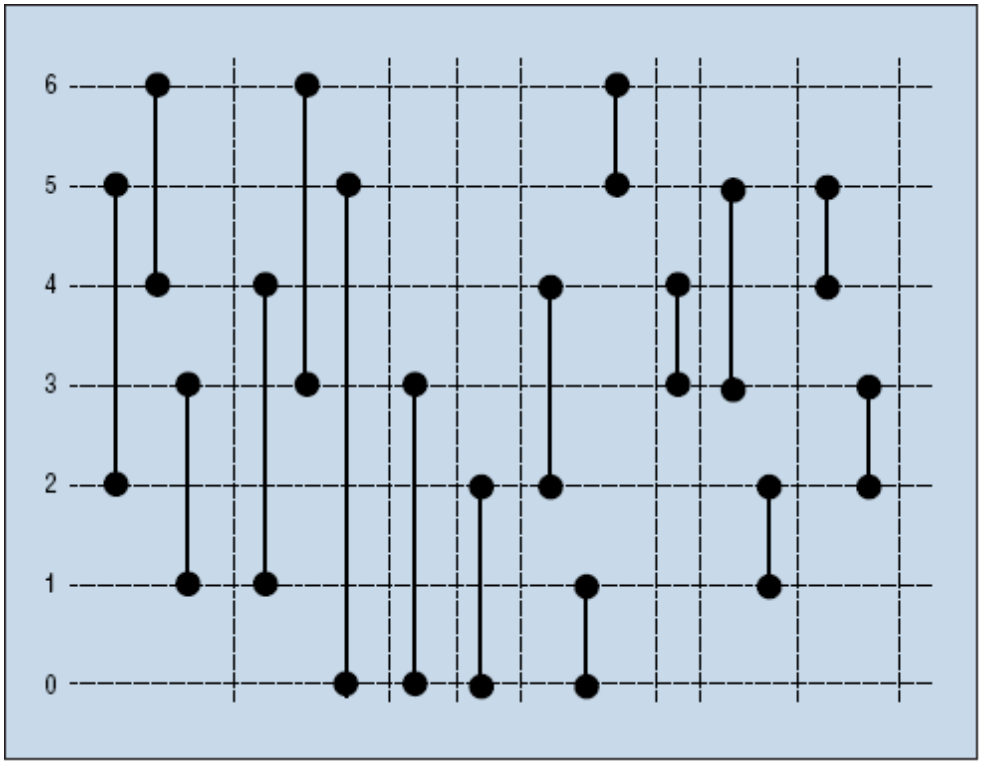


Gain block 730

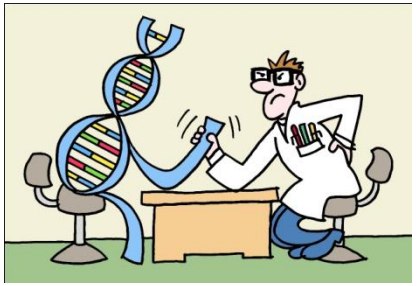
$$\left| \log \left| T_r \,?\, T_u + \log \frac{\log\left(|L|^{L}\right)}{T_u + 1} \right| \right|$$

Genetically Evolved Sorting Network

- **Genetic programming synthesized a 100 percent–correct 16-step sorting network** superior (that is, having fewer compare-swap operations) to the one Daniel G. O'Connor and Raymond J. Nelson presented in their 1962 patent (US patent 3,029,413).



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
  - …

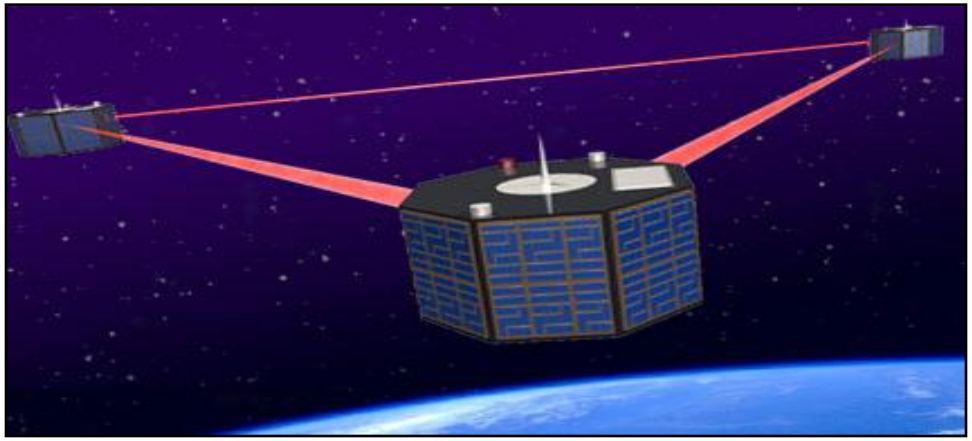
http://www.genetic-programming.org/hc2011/combined.html

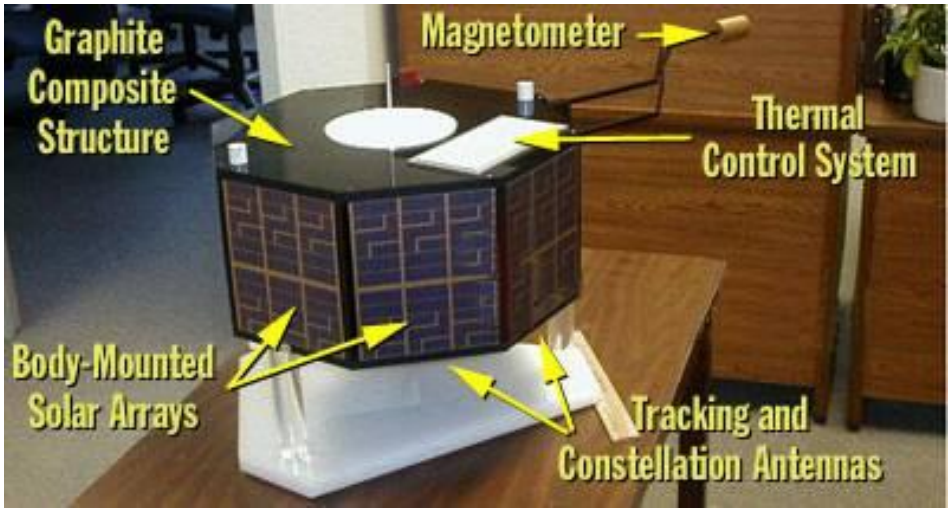
2004 Human-Competitive Awards in Genetic and Evolutionary Computation

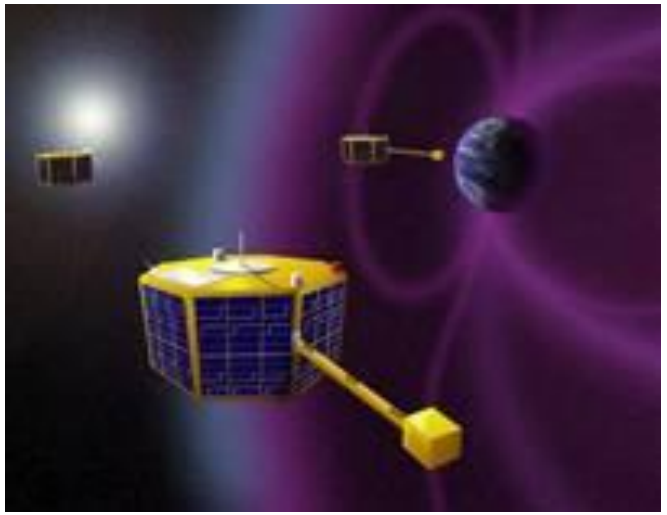
- http://www.genetic-programming.org/gecco2004hc.html
- **$1500 – Gold**
  - Jason D. Lohn, Gregory S. Hornby, Derek S. Linden: **An Evolved Antenna for Deployment on NASA's Space Technology 5 Mission**
  - Lee Spector: **Automatic Quantum Computer Programming: A Genetic Programming Approach**
- **$500 – 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 Misson

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

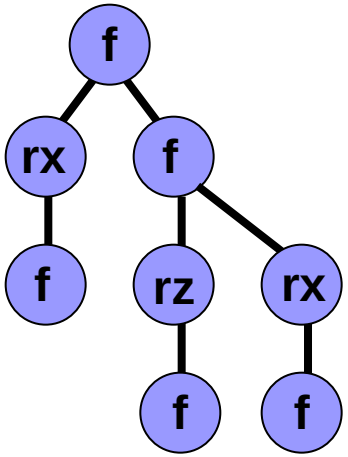
- Original ST5 Antenna Requirements
  - Transmit: 8470 MHz
  - Receive: 7209.125 MHz
  - Gain:
    - >= 0dBic, 40 to 80 degrees
    - >= 2dBic, 80 degrees
    - >= 4dBic, 90 degrees
  - 50 Ohm impedance
  - Voltage Standing Wave Ratio (VSWR):
    - < 1.2 at Transmit Freq
    - < 1.5 at Receive Freq
  - Fit inside a 6" cylinder

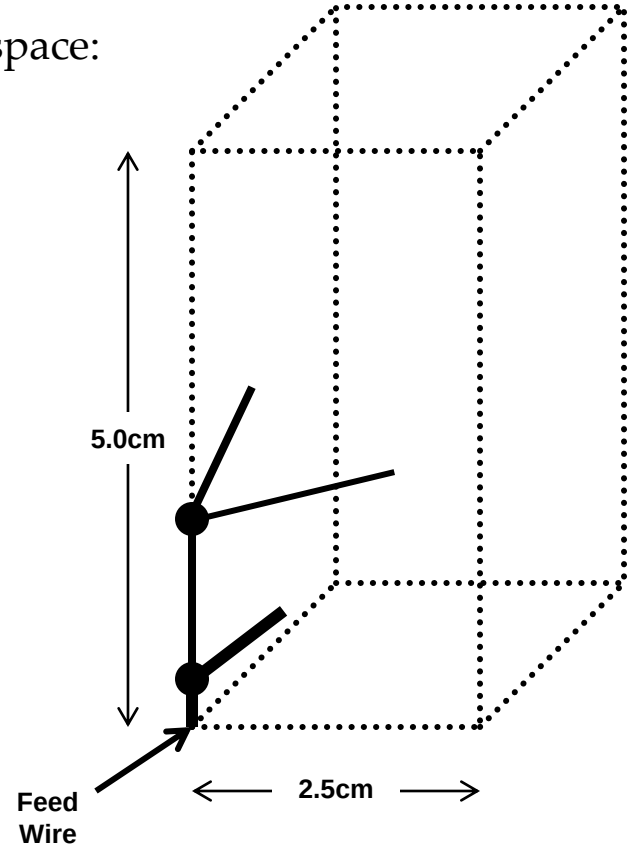
- ST5 Quadrifilar Helical Antenna
  - designed by a team of human designers

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

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 3-space:

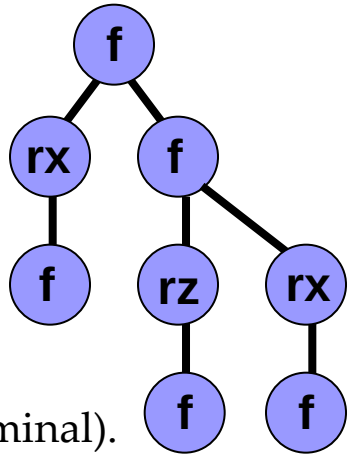




- Branching in genotype results in branching in wire form

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.

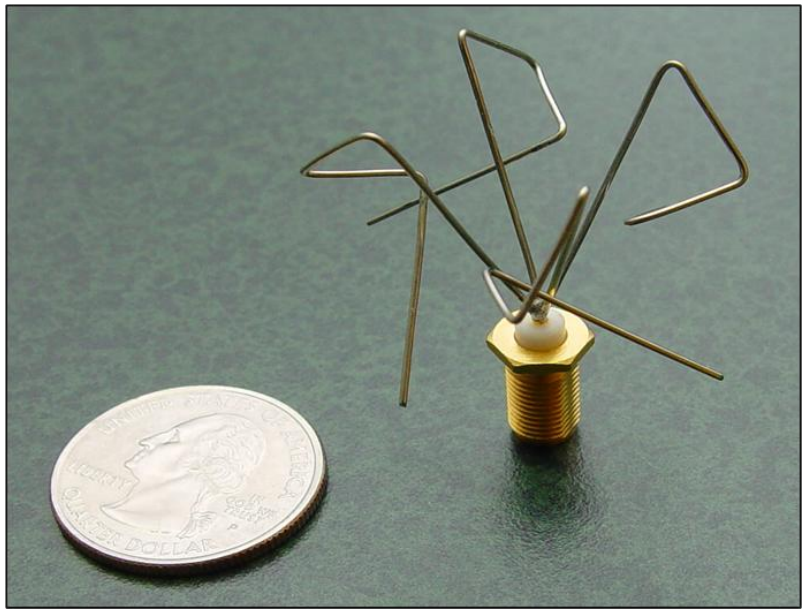
Rotate_x/y/z() commands have exactly 1 child (always non-terminal).

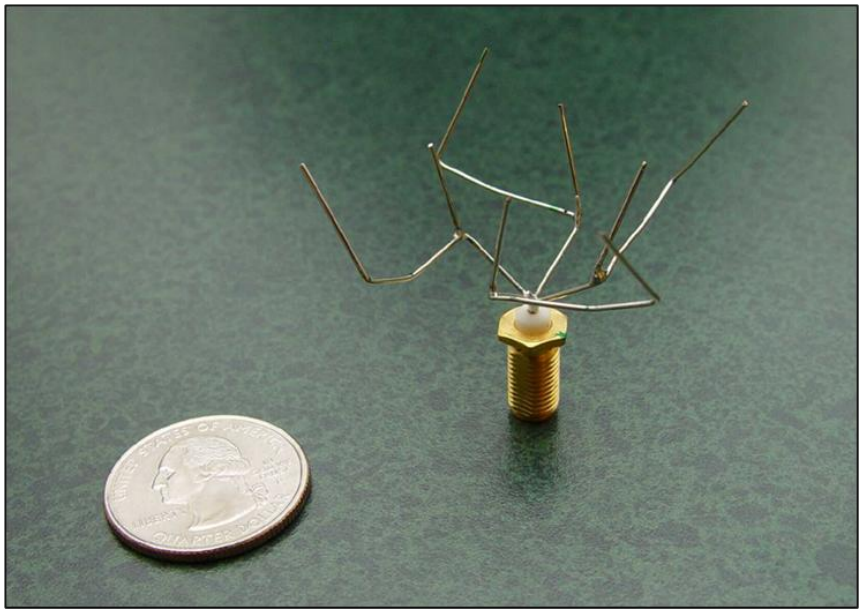
- **Fitness function** (to be minimized):

  F = VSWR_Score * Gain_Score * Penalty_Score

Evolved Antenna for Space Technology 5 mission

- 1st Set of Genetically Evolved Antennas





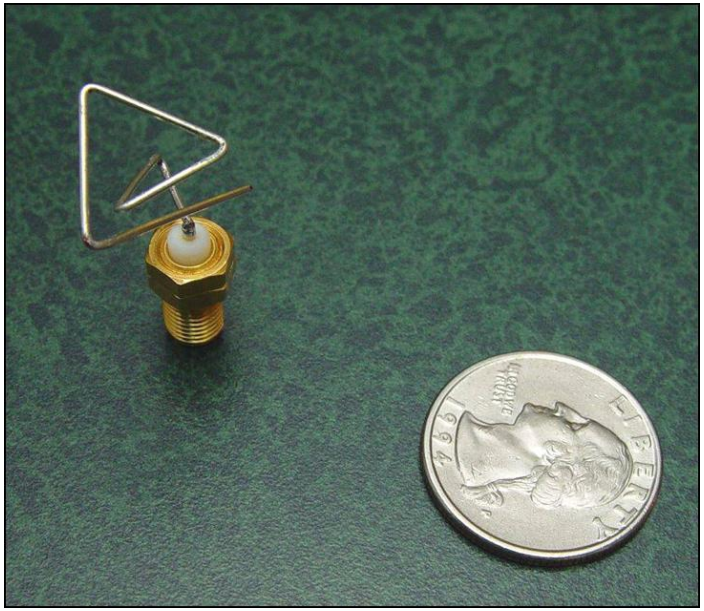
© Jason D. Lohn, Gregory S. Hornby, Derek S. Linden: Human-Competitive Results: Evolved Antennas for Deployment on NASA's Space Technology 5 Misson

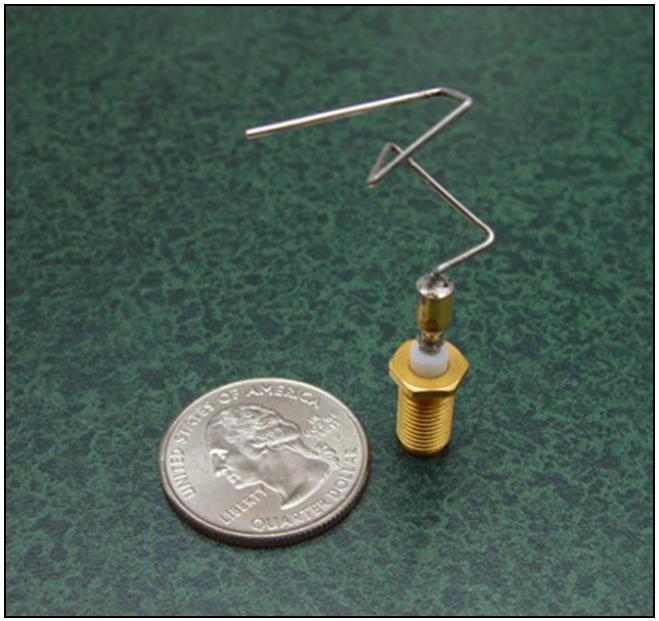
Non-branching:
ST5-4W-03

Branching:
ST5-3-10

Evolved Antenna for Space Technology 5 mission

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





EA 1 – Vector of Parameters

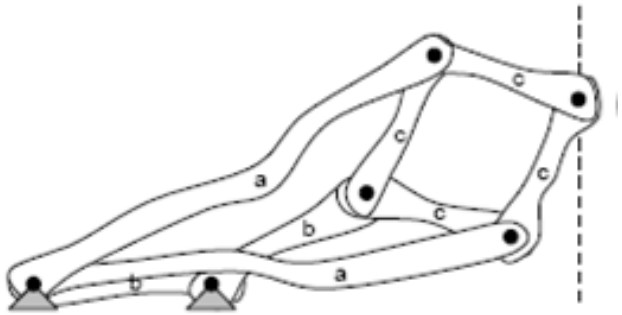
EA 2 – Constructive Process

Evolved Antenna for Space Technology 5 mission

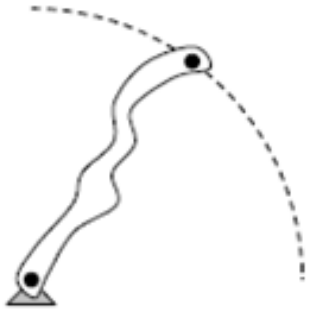
- **Conclusion**
  - Meets mission requirements.
  - Better than conventional design.
  - Successfully passed space qualification.
  - **First Evolved Hardware in Space when mission launched in 2005.**
- **Direct competition**: The antenna designed by the contracting team of human designers for the Space 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!



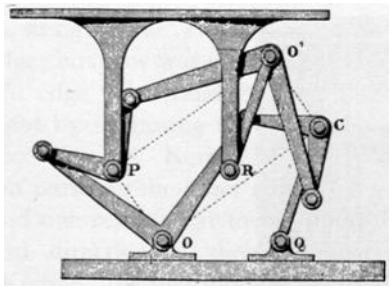
line



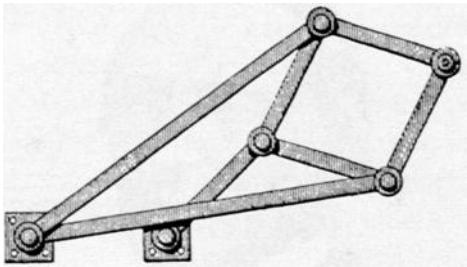
circle

How to Draw a Straight Line Using a GP

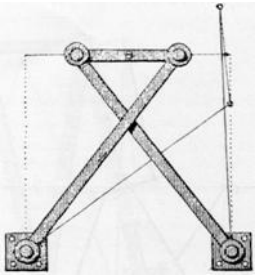
- Some key straight-line mechanisms



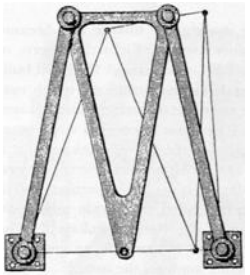
Silverster-Kempe's
(1877)



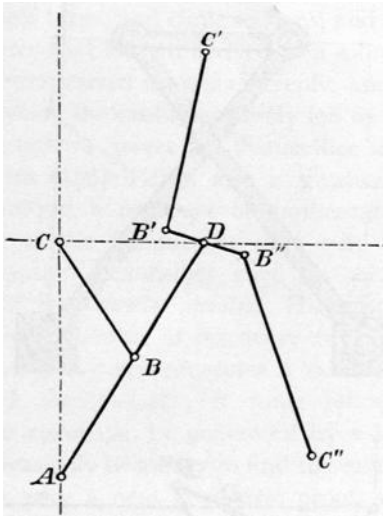
Peaucelier
(1873)



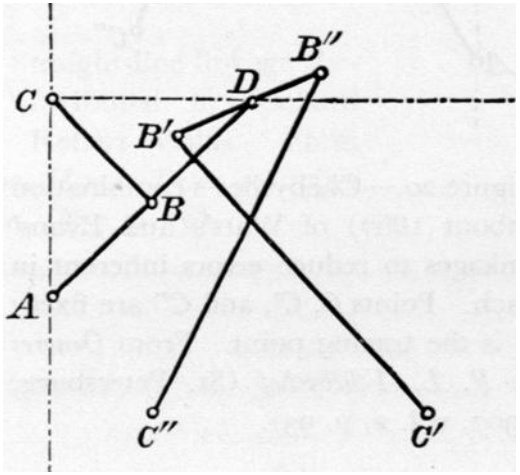
Chebyshev
(1867)



Robert
(1841)



Chebyshev
(1867)

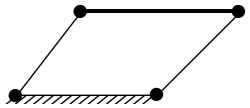


Chebyshev-Evans
(1907)

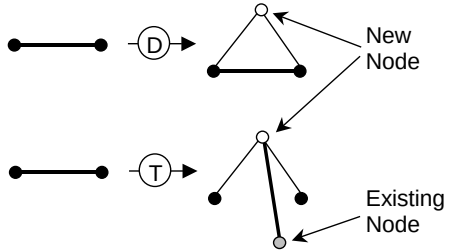
See
http://kmoddl.library.cornell.edu

How to Draw a Straight Line Using a GP

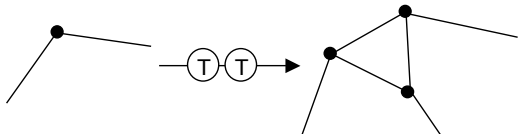
- Top down encoding of a mechanism



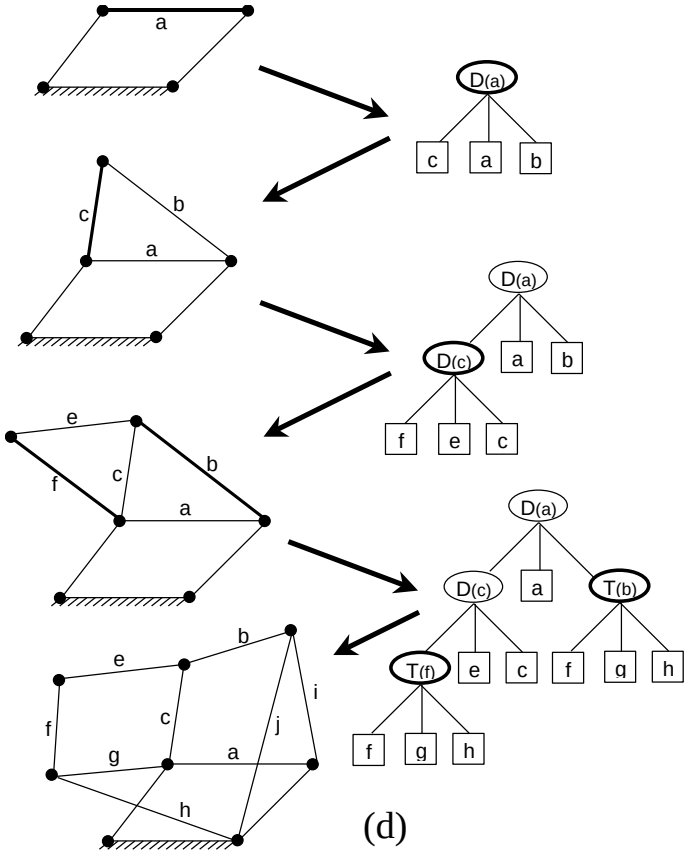
Start with Embryo with desired # of DoF, e.g. a four-bar mechanism (1 DoF)



Two variation operators maintain DoF



E.g. Transform dyad into tryad

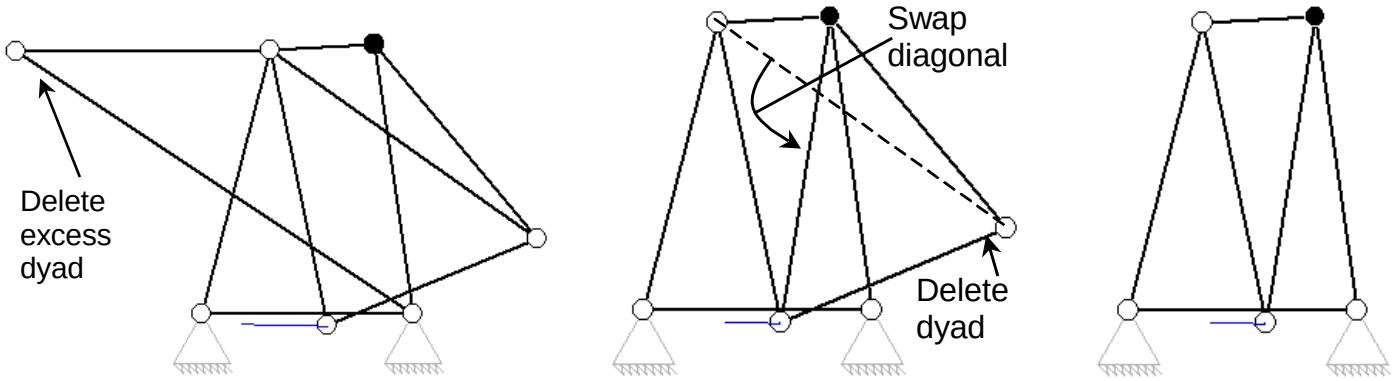


(d)

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

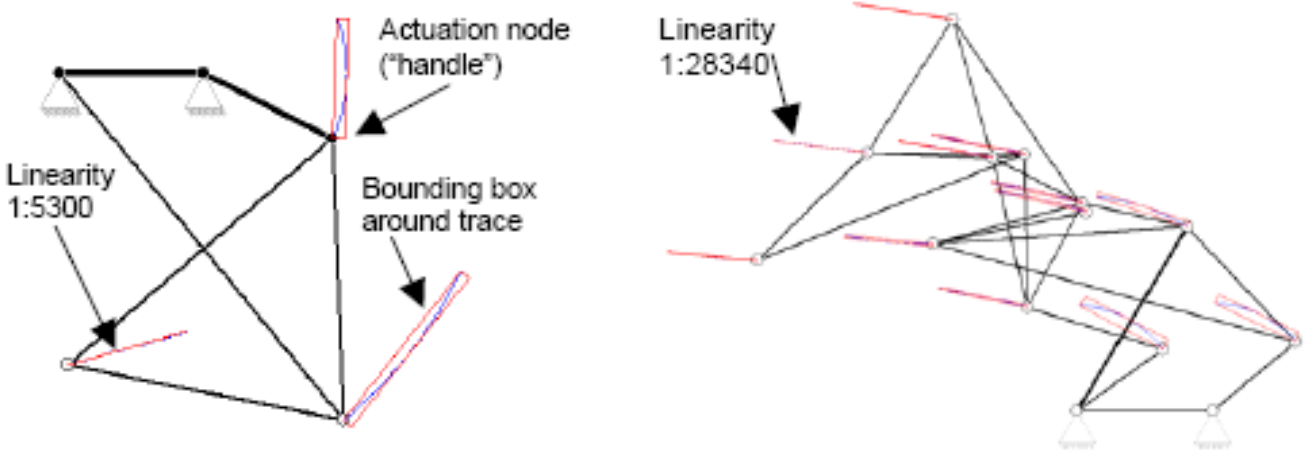
How to Draw a Straight Line Using a GP

- **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



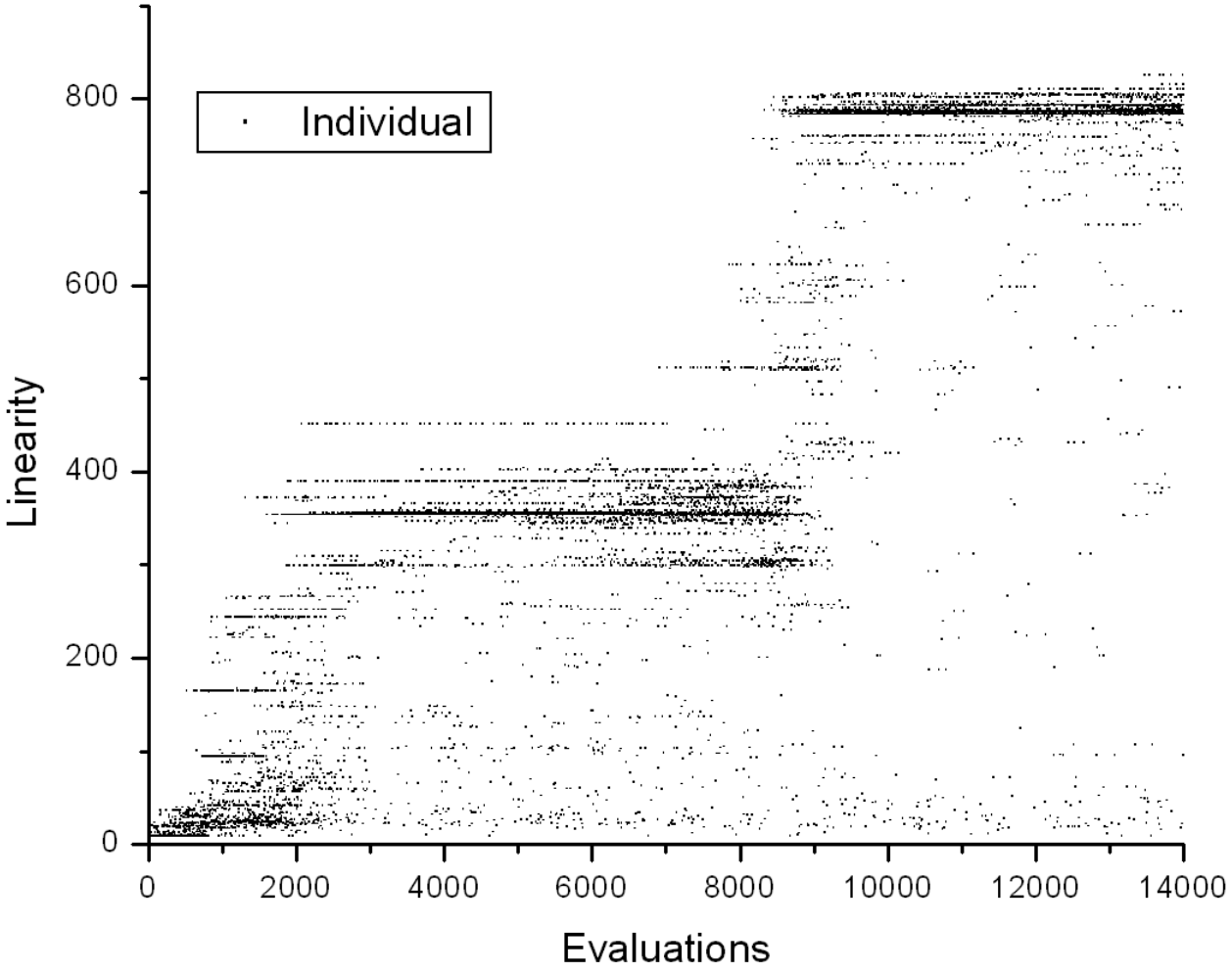
How to Draw a Straight Line Using a GP

- 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

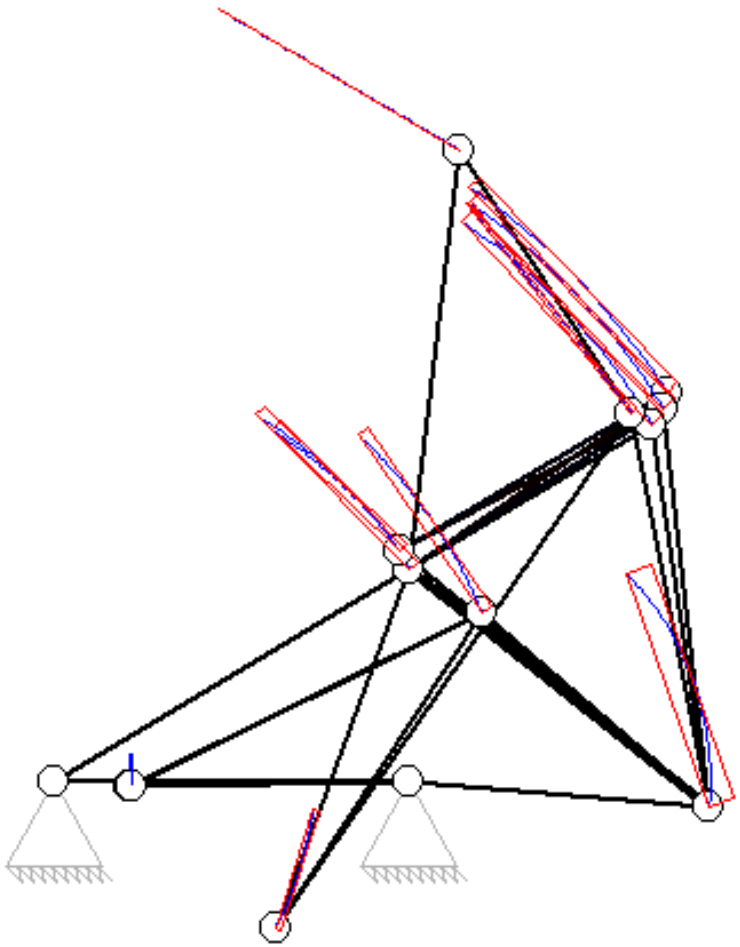


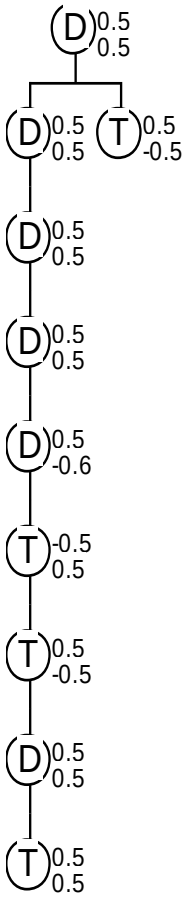
How to Draw a Straight Line Using a GP

- **A typical run** – each dot represents an evaluated individual



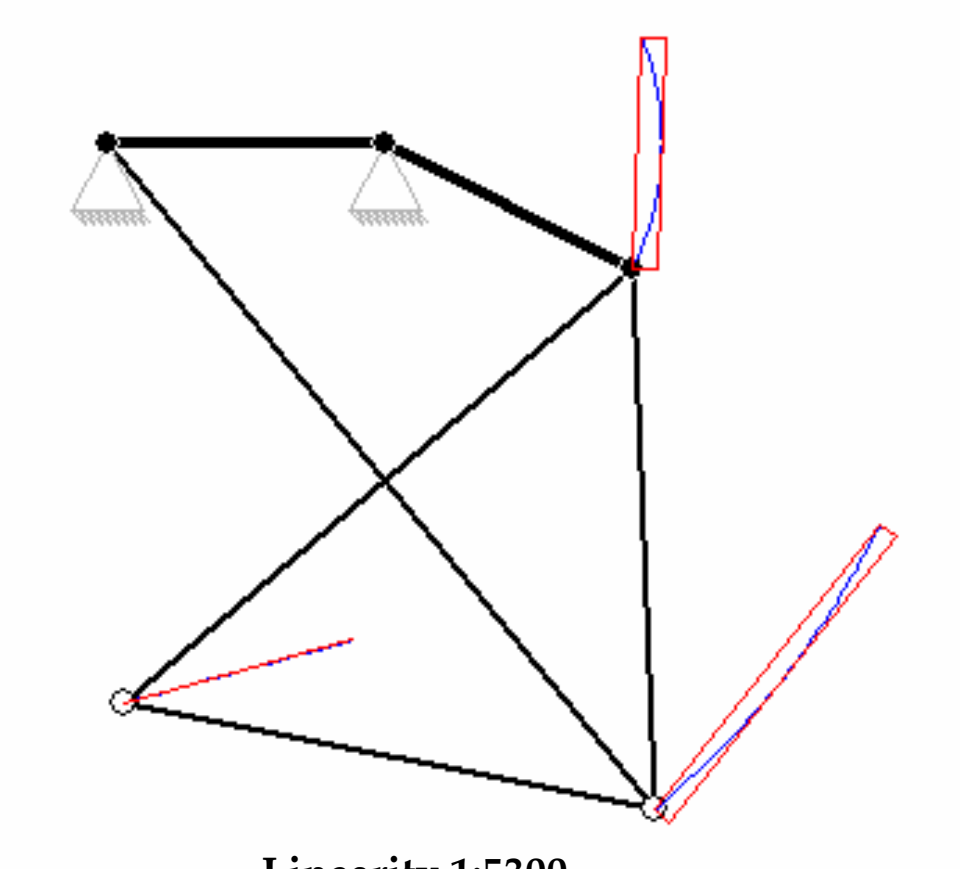
Some results





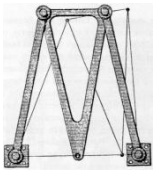
Linearity 1:4979

Some results

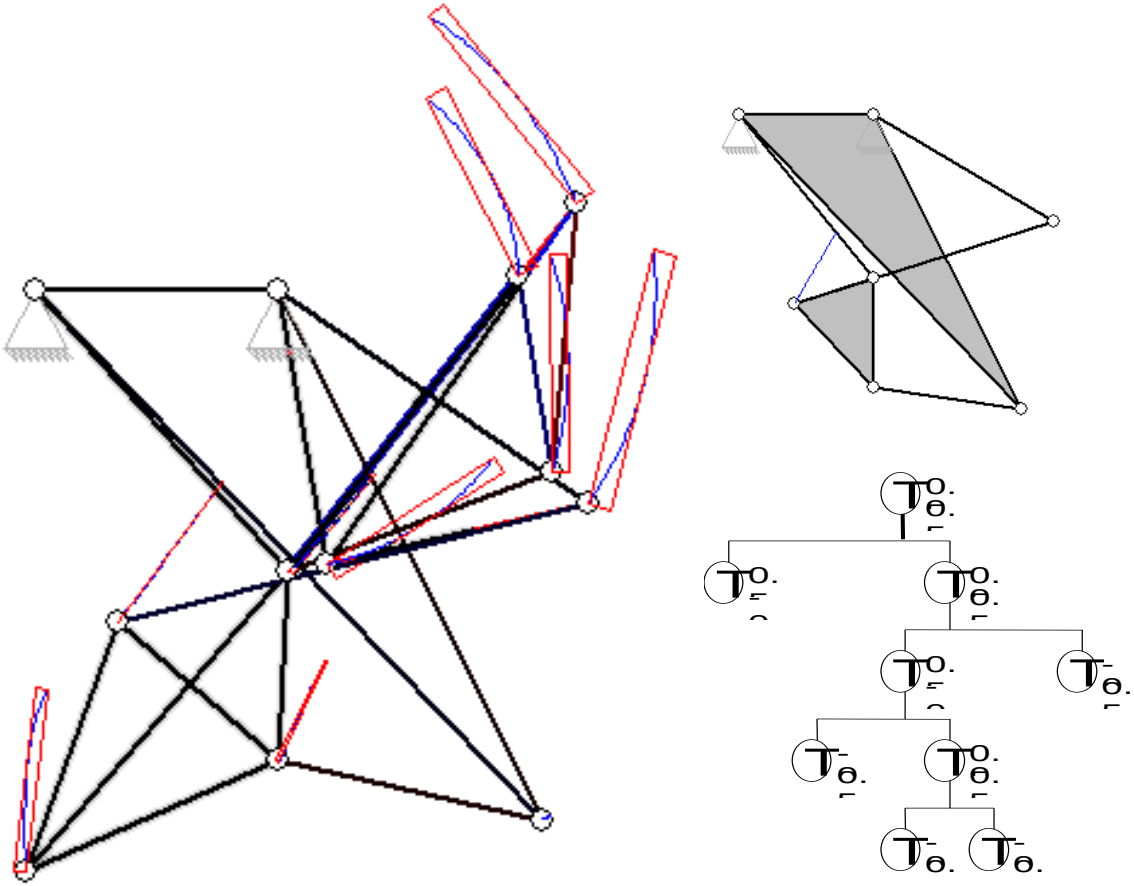


Linearity 1:5300

Infringes on Robert's Linkage (1841)
Published: Kempe A. B., (1877), *How To Draw A Straight Line*, London

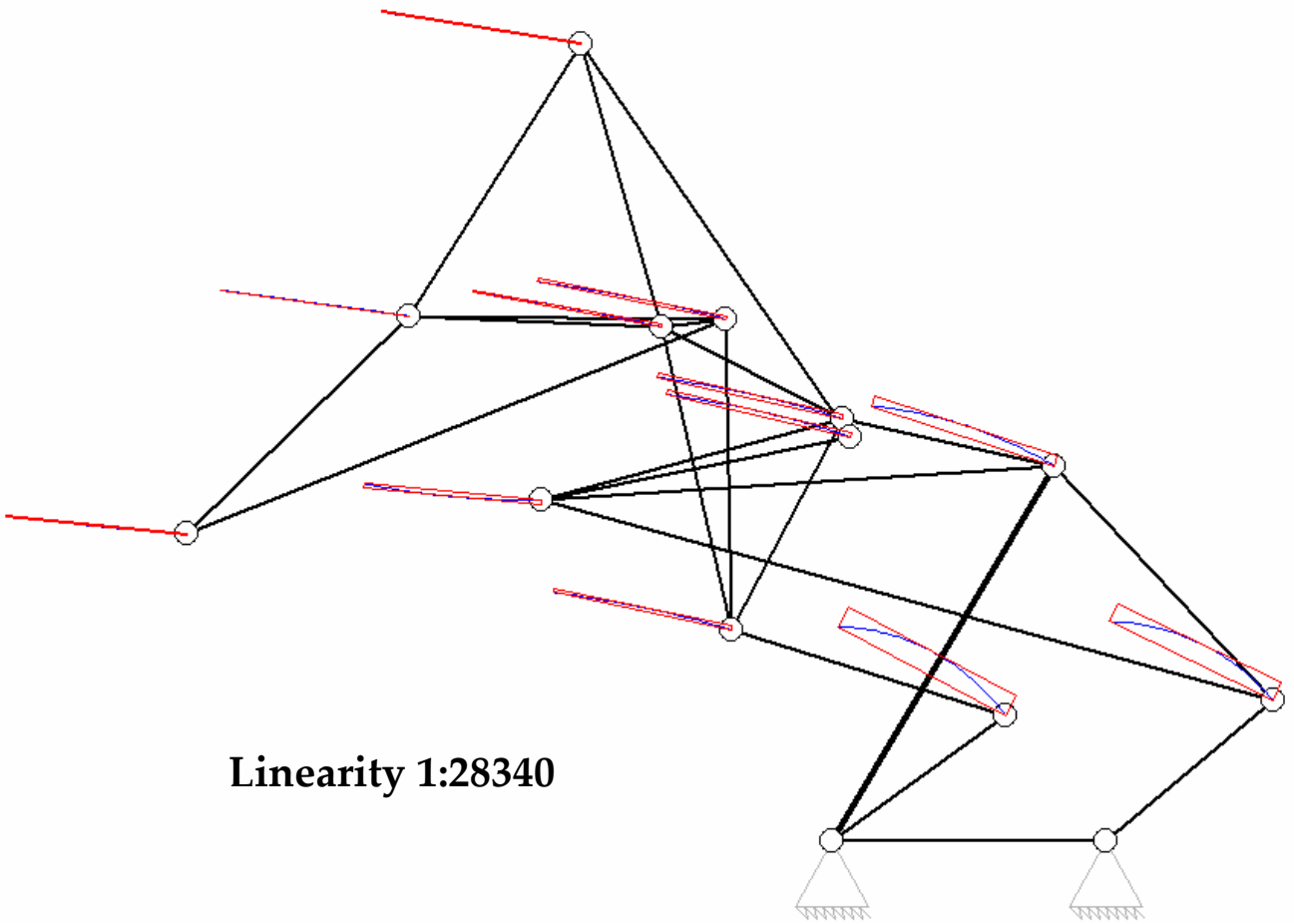


Some results



Linearity 1:12819

Some results



2005 Human-Competitive Awards in Genetic and Evolutionary Computation

- http://www.genetic-programming.org/hc2005/cfe2005.html
- **$2500 – Gold**
  - Randy Bartels et al.:

    (1) Learning from Learning Algorithms: Applications to attosecond dynamics of high-harmonic generation,

    (2) Shaped-pulse optimization of coherent soft-x-rays.

    Bartels, R. et al.: Learning from Learning Algorithms: Applications to attosecond dynamics of high-harmonic generation. Physical Review A. 70(1):1-5 (2004)
  - Stefan Preble, Hod Lipson, Michal Lipson: **Two-dimensional photonic crystals designed by evolutionary algorithms**
- **$1000 – Silver**
  - John Koza, Sameer Al-Sakran, Lee Jones: **Automated Re-Invention of Six Patented Optical Lens Systems using Genetic Programming**
  - Paul Massey, John A Clark, Susan Stepney: **Evolution of a Human-Competitive Quantum Fourier Transform Algorithm Using Genetic Programming**
  - Fulvio Corno, Edgar Ernesto, Sanchez Sanchez, Giovanni Squillero: **Evolving Assembly Programs: How Games Help Microprocessor Validation**

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)**
  - Uli Grasemann, Risto Miikkulainen: **Effective Image Compression using Evolved Wavelets**

Exploring different quantum behaviors in an fast automated fashion

- Bartels et al. use **an evolutionary optimization algorithm to shape laser pulses** to distort molecules in specific ways to catalyze chemical reactions, with the ultimate goal of manipulating large molecules, for example proteins and enzymes, in a biological cell.
- The method has also been used to create **new quantum behaviors** at the atomic level.
- The learning component is a relatively **simple evolution strategy**. A deformable mirror pulse shaper and an evolutionary strategy algorithm were then used to
  - selectively optimize the 27th harmonic,
  - while simultaneously suppressing the 25th and 29th orders.

  The optimum pulse shape is found after about 100 iterations of the learning algorithm, each of which consists of about 100 trials.
- Human can probe quantum systems, but are not capable of exploring different quantum behaviors in an fast automated fashion. In a sense, **the results are beyond human competitive**.

Exploring different quantum behaviors in an fast automated fashion

- **There is also no doubt that this is a major scientific result which was achieved using evolutionary computation.**
- The **results are completely unexpected and amazing from a physical point of view: behaviors are being evolved that were not known to be physically possible**. This includes anti-correlated attosecond harmonics in quantum systems. The ability to move beyond the nanoscale to the attoscale is a major breakthrough, and the potential applications of controlling the behavior of materials at atomic level are enormous.

2006 Human-Competitive Awards in Genetic and Evolutionary Computation

- http://www.genetic-programming.org/hc2006/cfe2006.html
- **$5000 – Gold**
  - Varun Aggarwal et al.: **Catalogue of Variable Frequency and Single-Resistance-Controlled Oscillators Employing A Single Differential Difference Complementary Current Conveyor**
- **$3000 – Silver**
  - Kumara Sastry et al.: **Multiobjective Genetic Algorithms for Multiscaling Excited-State Direct Dynamics in Photochemistry**
- **$1000 – Bronze**
  - Jie Yao et al.: **A multi-population genetic algorithm for robust and fast ellipse detection**
  - Gustavo Olague, Leonardo Trujillo:
    - **Using Evolution to Learn How to Perform Interest Point Detection**
    - **Synthesis of Interest Point Detectors Through Genetic Programming**

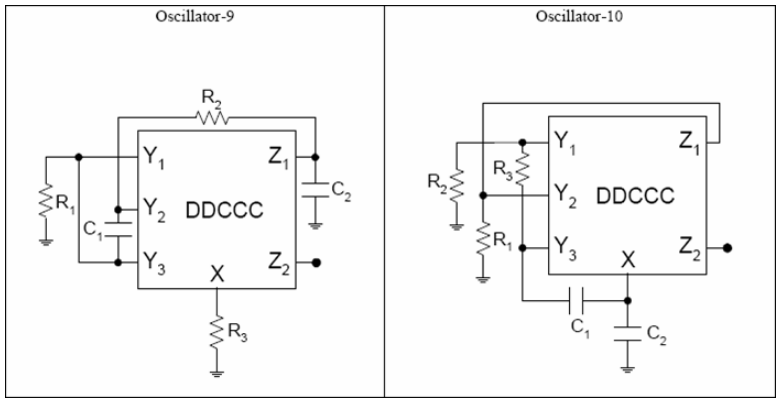
The winner of Humies 2006

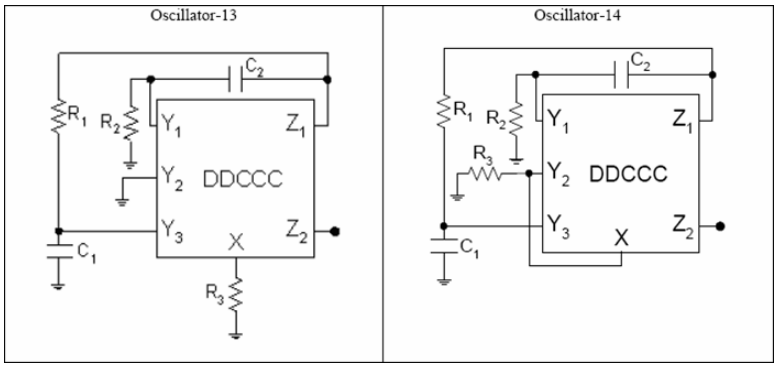
- Selcuk Kilinc et al: **Catalogue of Variable Frequency and Single-Resistance-Controlled Oscillators Employing a Single Differential Difference Complementary Current Conveyor**

  reference: http://web.mit.edu/varun_ag/www/hc-2006.html
- Sine-wave oscillators have various applications:
  - Communication systems,
  - Control and measurement,
  - Signal processing, etc.
- Interest in design of inductorless low power and low area oscillators
- Oscillator Design: How?
  - Human designed: Adhoc, intuition, analysis.
  - Exhaustive approaches: Mathematically rigorous. Infeasible with size/topological constraints.
  - Explore a small topological space manually
    - Fixed connection and different elements
    - Use of specific design principle

Oscillators

- **Genetic Algorithm:**
  - Push button approach to oscillator synthesis, Automatic, No human designer, No mathematics, Looks at complete search space, Scaleable.
  - **Can it find new design ideas?**
- GA evolved a catalogue of sinusoidal oscillators with single resistor frequency control, that use only a single active building block (ABB) and minimum number of passive components.





- **Value of result**
  - Design beyond expectation, new design principles
  - Human interpretable and SPICE validated
  - Improvement over the state-of-art: Lower in power and area.

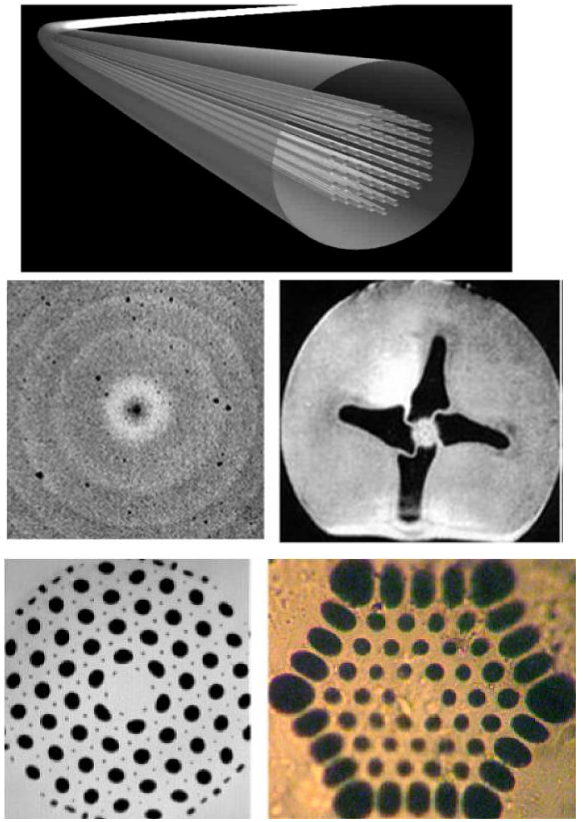
2007 Human-Competitive Awards in Genetic and Evolutionary Computation

- http://www.genetic-programming.org/hc2007/cfe2007.html
- **$5000 – Gold**
  - Steven Manos et al.: **Evolutionary Design of Single-Mode Microstructured Polymer Optical Fibres using an Artificial Embryogeny Representation**
- **$3000 – Silver**
  - Ami Hauptman, Moshe Sipper: **Evolution of an Efficient Search Algorithm for the Mate-In-N Problem in Chess**
- **$1000 – Bronze**
  - Jaume Bacardit et al.: **Automated Alphabet Reduction Method with Evolutionary Algorithms for Protein Structure Prediction**
  - Xavier Llorà et al.: **Towards Better than Human Capability in Diagnosing Prostate Cancer Using Infrared Spectroscopic Imaging**

Evolutionary Design of Single-Mode Microstructured Polymer Optical Fibres

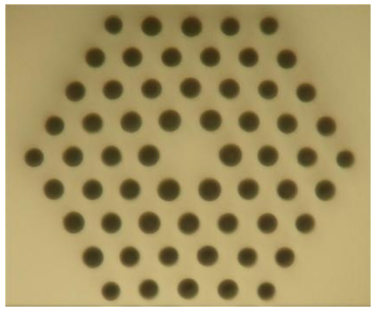
- Steven Manos, Leon Poladian, Maryanne Large: **Evolutionary Design of Microstructured Polymer Optical Fibres using an Artificial Embryogeny Representation**
  reference: http://www.genetic-programming.org/hc2007/cfe2007.html
- Applications of optical fibres
  - Long distance telecommunications
  - 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?

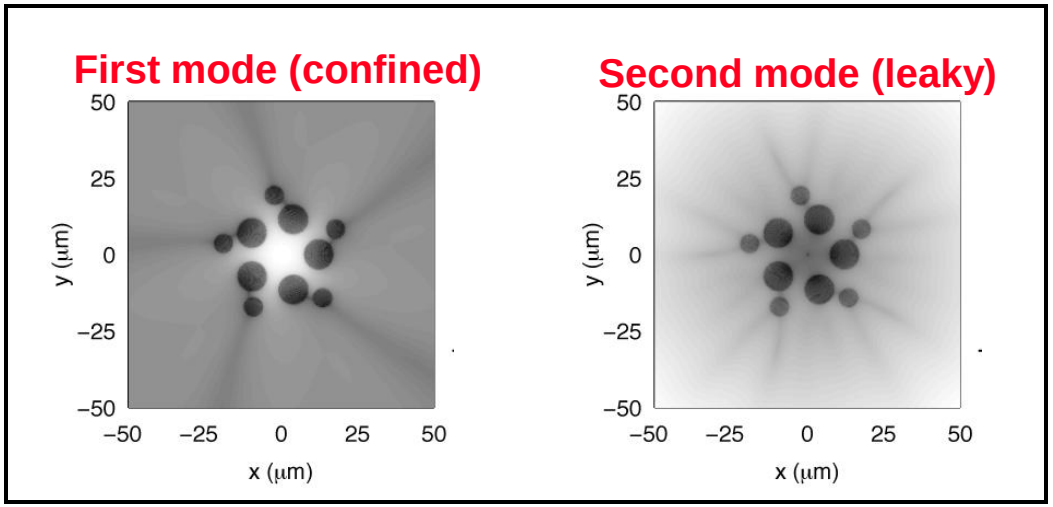


Single-moded fibres

Typical hexagonal design



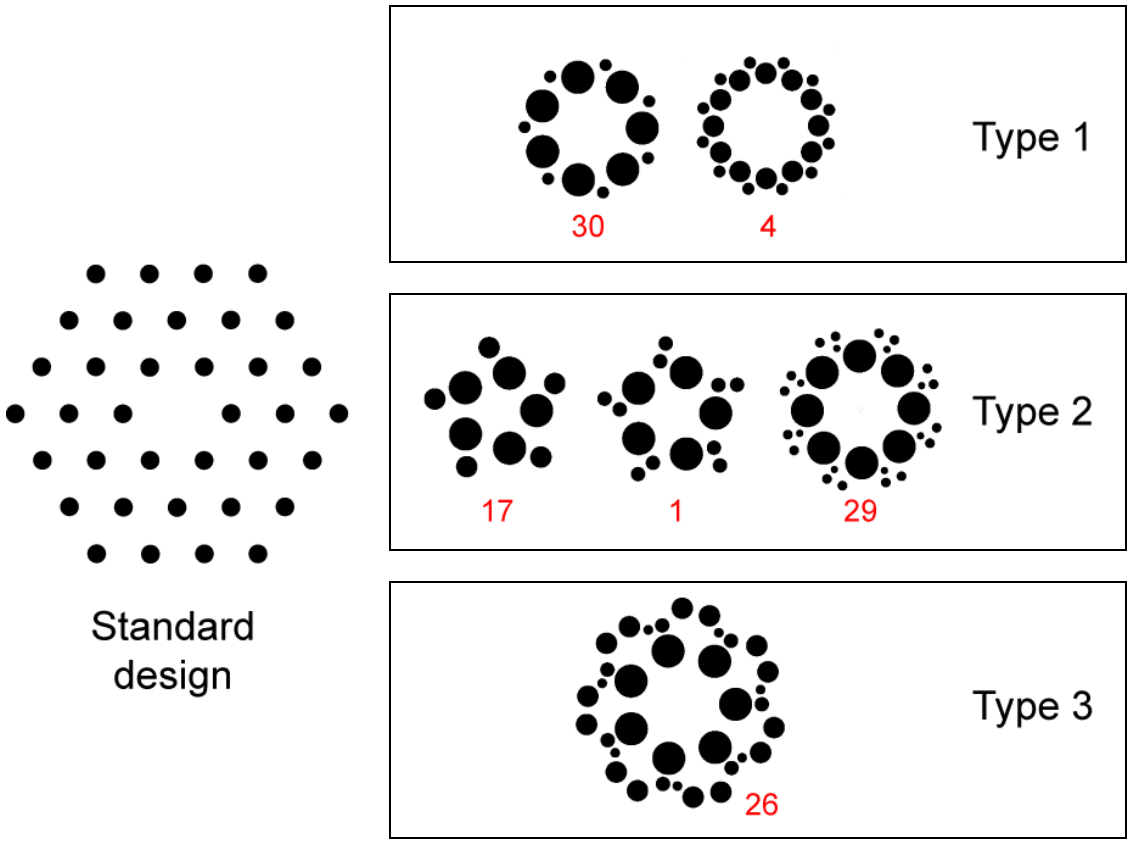
Standard design since the early 1990's



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.

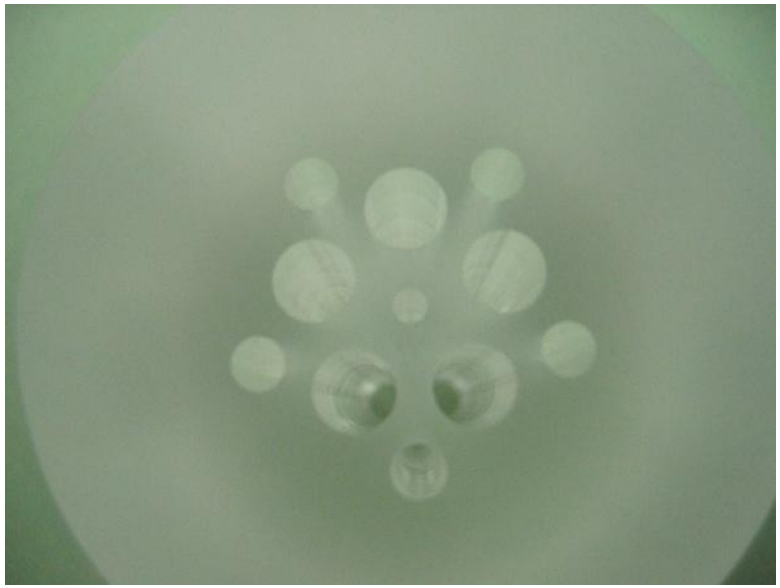
Evolved single-mode designs

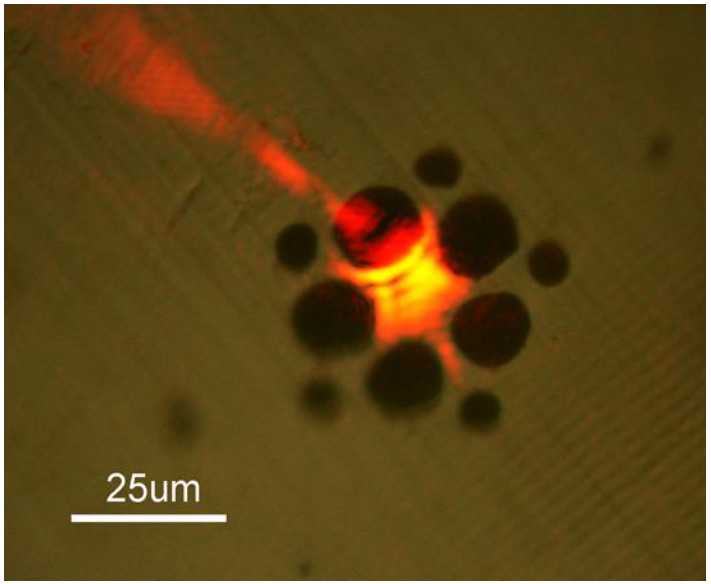


All designs have confined fundamental modes with $l_{c,1} < 10^{-1}$ **dB/m**, with losses more typically being $l_{c,1} < 10^{-3}$ **dB/m**.
The loss of the second mode $l_{c,2} > 10^{4}$ **dB/m** in all cases.

All single-moded, yet phenotypically different.

Manufactured single-mode MPOF



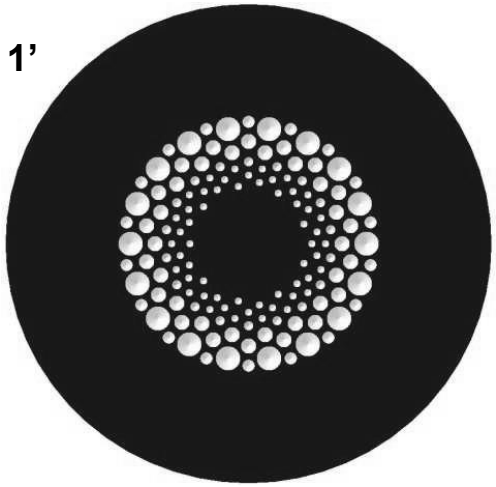


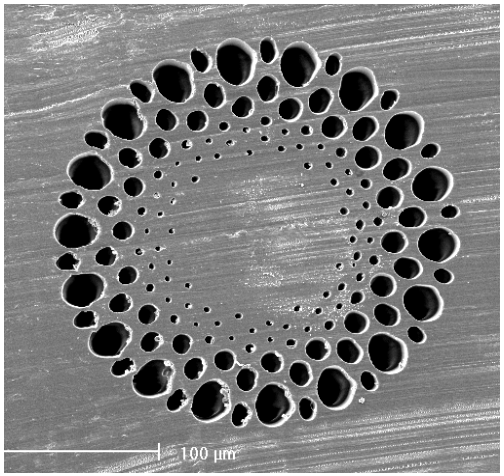
- **Evolved designs are simpler than previous designs, and easier to manufacture.**
- Provided us with a rich set of never before seen single-moded microstructured fibre designs to investigate further.

A different fitness function

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

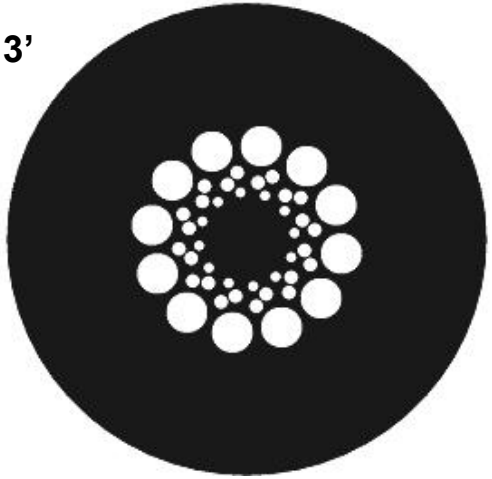
'GIMP 1'

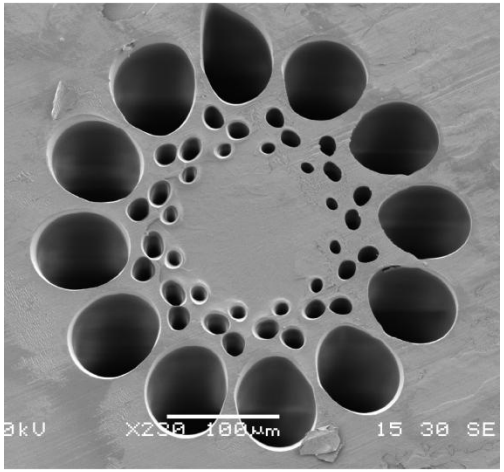




Hand-designed fibre

'GIMP 3'





Patented GA-designed fewer holes, easier to manufacture.

2008 Human-Competitive Awards in Genetic and Evolutionary Computation

- http://www.genetic-programming.org/hc2008/cfe2008.html
- **$5000 – Gold**
  - L. Spector, D. M. Clark, I. Lindsay, B. Barr, J. Klein**: Genetic Programming for Finite Algebras**
- **$3000 – Silver**
  - T. Pecenka, Z. Kotasek, L. Sekanina**: Evolution of Synthetic RTL Benchmark Circuits with Predefined Testability**
- **$1000 – Bronze**
  - A. Glazer, M. Sipper**: Evolving an automatic defect classification tool**

2009 Human-Competitive Awards in Genetic and Evolutionary Computation

- http://www.genetic-programming.org/hc2009/cfe2009.html
- **$5000 – Gold**
  - S. Forrest, C. Le Goues, T. Nguyen, W. Weimer**: A Genetic Programming Approach to Automated Software Repair**
- **$3000 – Silver**
  - M. Shahzad, S. Zahid, M. Farooq, S. Ali Khayam**: A Hybrid GA-PSO Fuzzy System for User Identification on Smart Phones**
- **$1000 – Bronze**
  - A. Hauptpman, A. Elyasaf, M. Sipper, A. Karmon**: GP-Rush: Using Genetic Programming to Evolve Solvers for the Rush Hour Puzzle**
  - C. B. Perez, G. Olague**: Learning Invariant Region Descriptor Operators with Genetic Programming and the F-measure**

Automatically Finding Patches Using GP

- Fully automated 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** 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 occurences in the tescases.
  - **Program locations visited when executing the negative testcases** are favored to program locations visited while executing the positive 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

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

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

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

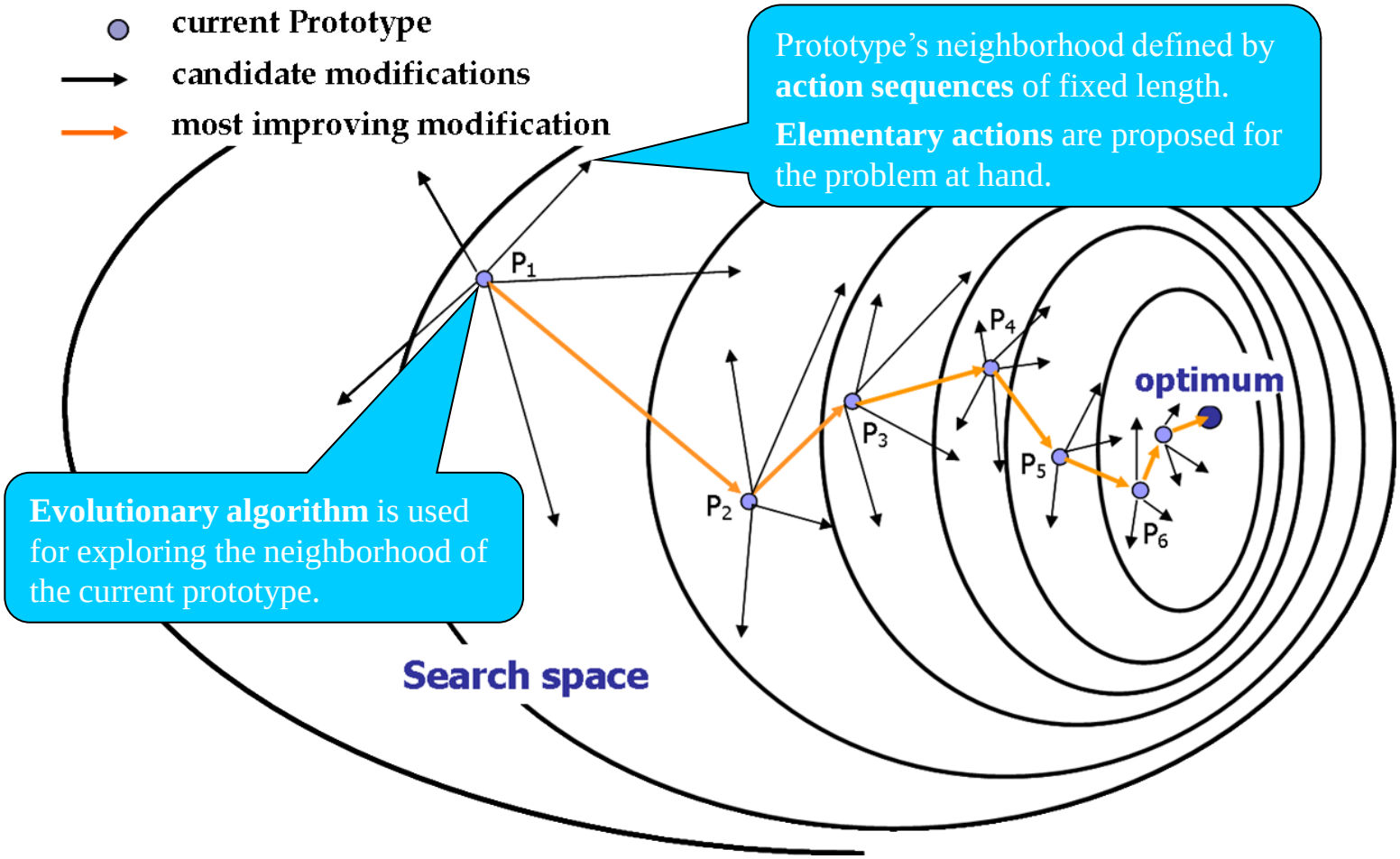
generated given the bias towards modifying lines that are involved in producing the faults and the preference for insertions similar to existing code.

Automatically Finding Patches Using GP

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

Program	LOC	Positive Tests	$\|Path\|$	Initial Repair				Minimized Repair		
				Time	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

Prototype Optimization with Evolved Improvement Steps (POEMS)



POEMS Characteristics

- **POEMS** takes the best of the two worlds of the **single-state and population-based** metaheuristics:
  - **Contrary to single-state metaheuristics**, that define the neighborhood by a single variation operator, **POEMS searches much bigger neighborhood**.

    Therefore, the exploration capability of POEMS is better than the standard single-state techniques.
  - **The prototype's neighborhood is effectively searched by means of an evolutionary algorithm** (EA).

    The space of prototype's neighbors is smaller than a space of all candidate solutions to the given problem, thus, moderate resources (i.e. small population size and small number of generations) can be sufficient to effectively search the prototype's neighborhood.

Kubalik, J. and Faigl, J.: Iterative prototype optimisation with evolved improvement steps. In: P. Collet, M. Tomassini, M. Ebner, A. Ekart, S. Gustafson (Eds.), Proceedings of the 9th European Conference on Genetic Programming, EuroGP 2006, Springer, pp. 154–165, 2006.

Multiobjective POEMS

```
generate(SolutionBase)
repeat
    Prototype ← choose_prototype(SolutionBase)
    ActionSequences ← run_MOEA(Prototype, SolutionBase)
    NewSolutions ← apply_to(ActionSequences, Prototype)
    SolutionBase ← merge(NewSolutions, SolutionBase)
until(mPOEMS termination condition is fulfilled)
return SolutionBase
```

- mPOEMS is an optimization algorithm based on the **dominance concept.**
- **SolutionBase** is a set of best solutions found so far.
  Best-of-run solutions are the non-dominated solutions of the SolutionBase.
- Function **choose_prototype()** chooses a prototype as one of the non-dominated solutions of the SolutionBase.
- **MOEA** takes into account **quality** and **uniqueness** of generated neighbours of the current prototype.

Kubalik, J., Mordinyi, R., Biffl, S.: Multiobjective prototype optimization with evolved improvement steps. In: EvoCOP 2008, Springer, ISBN 978-3-540-78603-0, pp. 218–229, 2008.

Application Areas of (m)POEMS

POEMS is a general-purpose optimization approach for hard combinatorial, discrete, and real-valued optimization problems with huge search space.

It is well suited for solving problems of the following classes:

- **Problems involving a refinement** of an initial existing solution, representatives of which are re-planning and re-scheduling.
- **Black-box** optimization problems where no information about the optimized function is available (such as gradients, properties of the fitness landscape, etc.).
- Problems where standard evolutionary algorithms fail due to **a large solution representation** (long chromosome, large solution graph or network) and for which it is **hard to design an efficient crossover** operator.
- **Design of hyper-heuristics** - described as "heuristics to choose heuristics", are search methods or learning mechanisms for selecting or generating heuristics to solve computational search problems.
- **Dynamic optimizations** - are a class of problems whose specifications, evaluation function and/or problem-specific constraints, change over time, resulting in continuously moving optima.

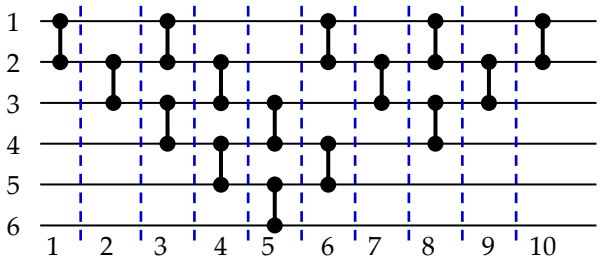
Sorting Networks Design

- **Problem:**
  - Given: **Sorting Network** is a network of n wires and k comparators arranged such that for any input sequence of numbers, the output sequence is monotonically sorted.

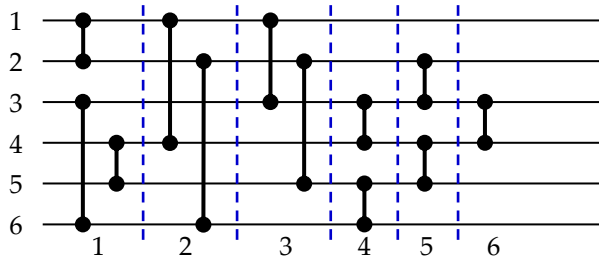
    Each **comparator** performs a compare-swap operation; it connects two wires and outputs the two input values in desired order.

  - Goal: To find a perfect sorting network for the given number of inputs that is optimal w.r.t. the given efficiency measure – such as a size (number of comparators) or a number of parallel layers.

Ex.:



Size: 15; #layers: 10



Size: 12; #layers: 6

Sorting Networks Design

- **Actions** (variation operators used as building blocks of the action sequences)
  - *InsertComparator, RemoveComparator*
  - *SwapComparators , MoveComparator, ModifyComparator*
  - *MoveBlockOfComparators*

- **Compared algorithms**
  - (μ+λ)-Evolution strategy,
  - (1+λ)-Evolution strategy.

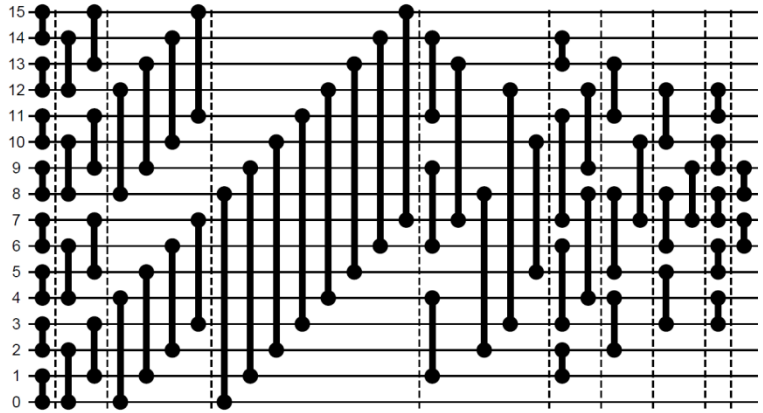
  In both cases, random structured mutations composed of elementary actions designed for the POEMS were used.

- **Main achievements**
  - **POEMS found sorting networks of minimal known size up to 16 inputs** (the 16-input sorting network is the largest one documented in literature).
  - This was achieved **without any problem-specific heuristic or local search** routines.

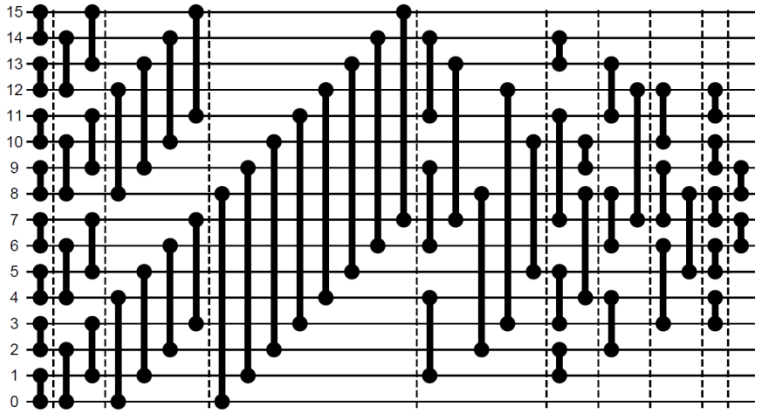
Kubalik, J.: Solving the sorting network problem using iterative optimization with evolved hypermutations. In: Genetic and Evolutionary Computation Conference 2009 [CD-ROM], New York, ACM, pp. 301–308, 2009.

Evolved 16-input SNs

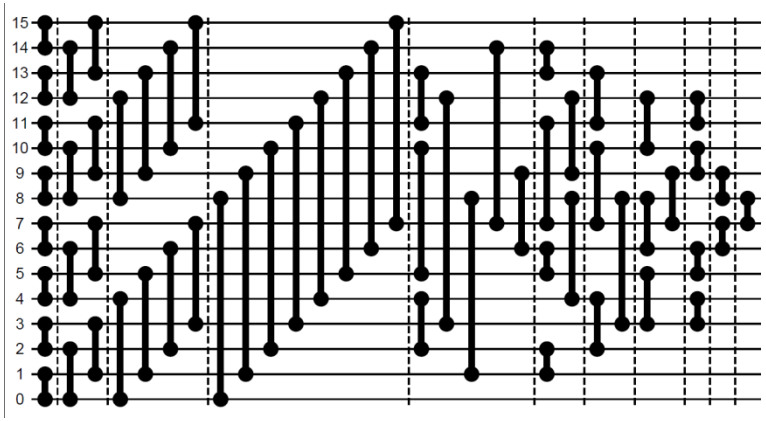
1. 60 comparators in 10 parallel layers



2. 60 comparators in 10 parallel layers



3. 60 comparators in 11 parallel layers



Sorting network with 60 comparators is the least known 16-input sorting network!!!

Shortest Common Supersequence

- **Problem:**
  - Given: A set L of strings s_i composed of symbols from a certain alphabet Σ.
  - Goal: To find a minimal-length sequence S of symbols from Σ such that all strings in set L can be embedded in S.

Ex.:

s_1: ca ag cca cc ta cat c a

s_2: c gag ccat ccgtaaa g tt g

s_3: aga acc tgc taaatgc t a ga

Supersequence *S*: cagagaccatgccgtaaatgcattacga

length=28

- **Actions** (variation operators used as building blocks of the action sequences)
  - *InsertLetter* - inserts new letter into the developed supersequence at given position,
  - *DeleteLetter* - deletes a letter at given position of the supersequence,
  - *MoveLetter* – moves given letter from its original position to specified new one.

Shortest Common Supersequence

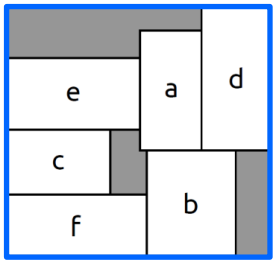
- **Compared algorithms** for moderate size synthetic and real biological data sets – up to 10 sequences of app. length 1000.
  - Majority Merge (MM),
  - Probabilistic Beam Search (PBS),
  - Hybrid Memetic Algorithm – Beam Search (MA-BS) - current state-of-the-art metaheuristic technique for the SCS problem.
- **Compared algorithm** for large scale data sets – up to 500 sequences of length 1000 and 1000 sequences of length 500.
  - Deposition and reduction algorithm (DR).
- **Main achievements**
  - POEMS beats MM and PBS and is competitive to MA-BS on moderate size data.
  - POEMS produced significantly better solutions than the baseline DR algorithm on the challenging large data sets.

Kubalik, J.: Evolutionary-Based Iterative Local Search Algorithm for the Shortest Common Supersequence Problem. In: Genetic and Evolutionary Computation Conference 2011, ACM, ISBN: 978-1-4503-0557-0, pp. 315–322, 2011.

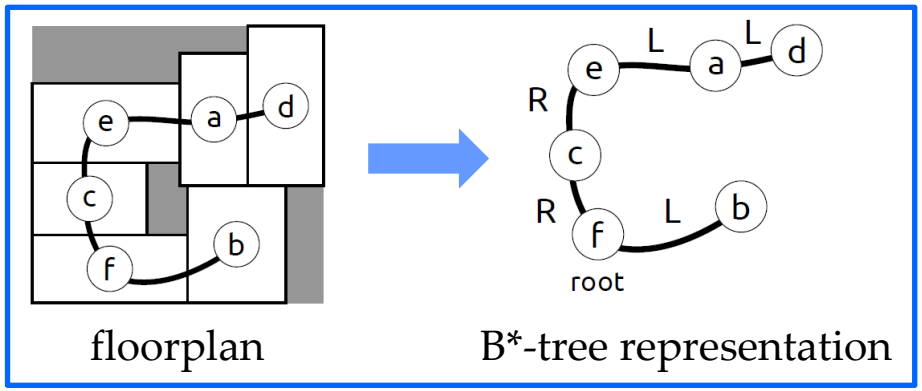
Floorplanning

- **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.



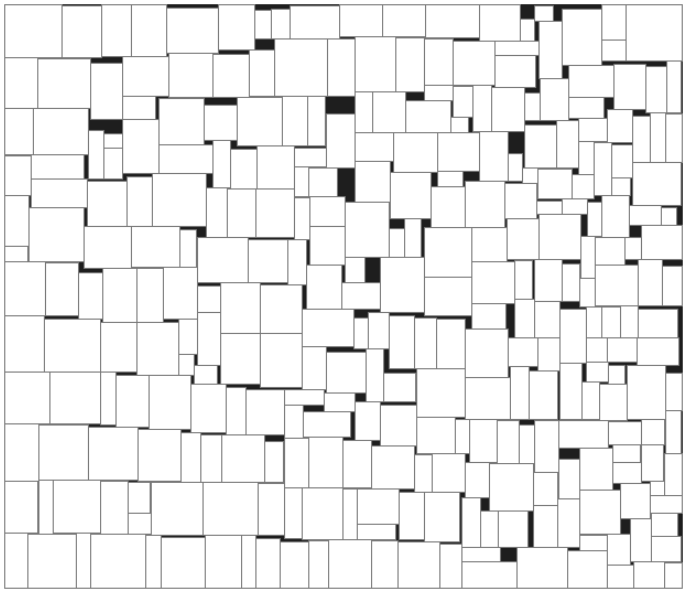
Floorplanning

- **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.

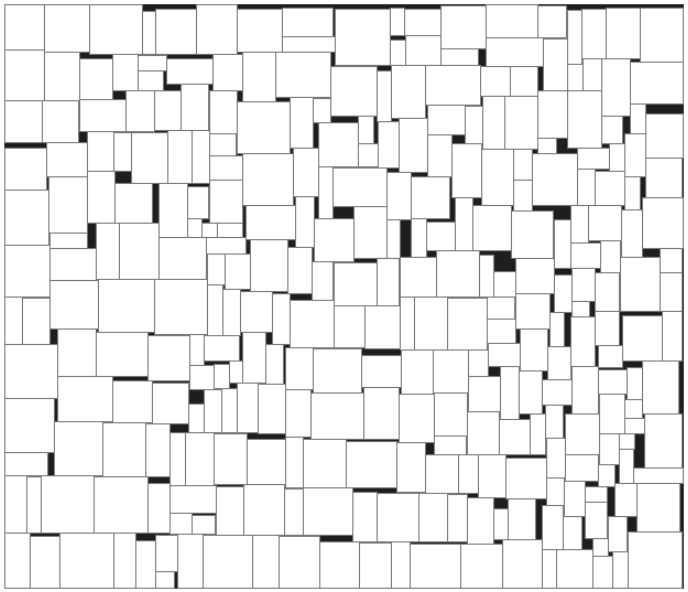
Floorplanning

- 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.