Evolutionary Algorithms and Their Applications

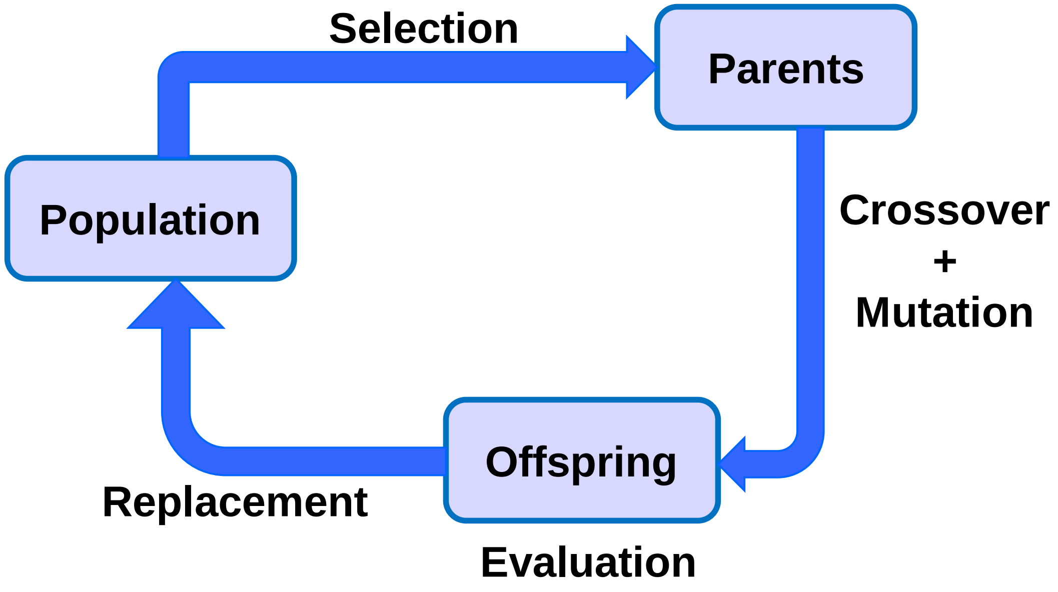
Jiří Kubalík

Czech Institute of Informatics, Robotics, and Cybernetics
CTU in Prague
email: jiri.kubalik@cvut.cz

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.

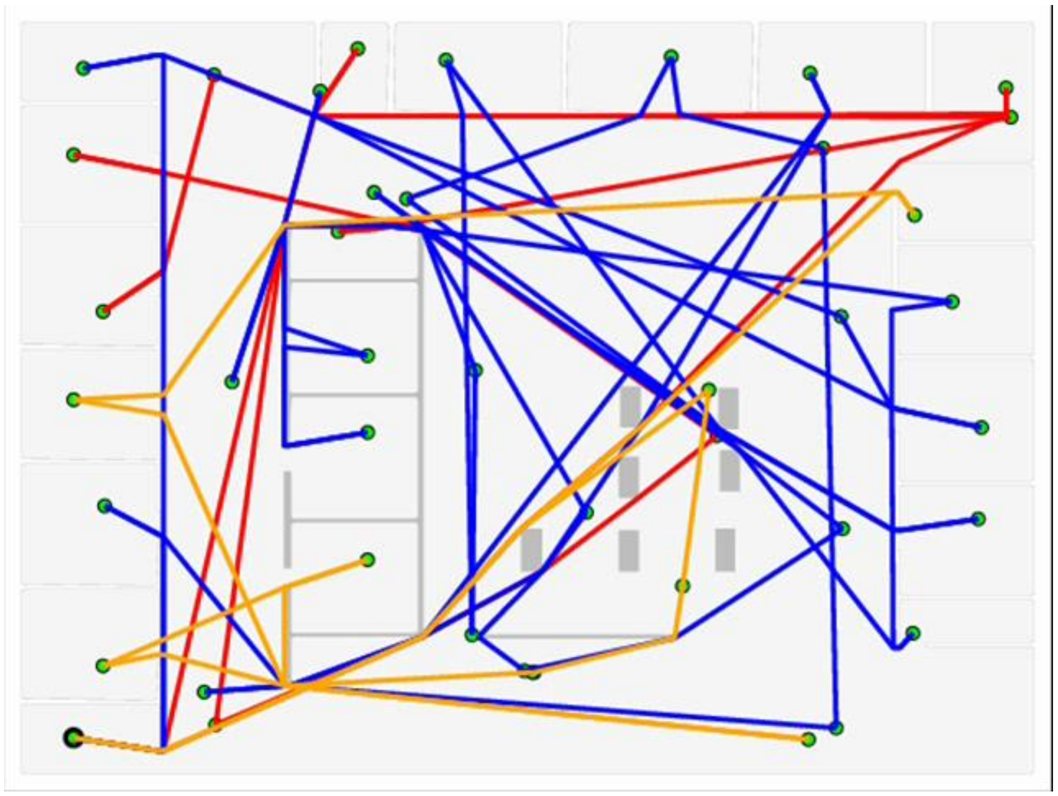
  The problem can involve multiple optimization objectives as well.
- **A typical evolutionary model:**



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.



Evolutionaty Algorithms for Dynamic Opt.

Dynamic optimizations - a class of problems whose specifications commonly termed

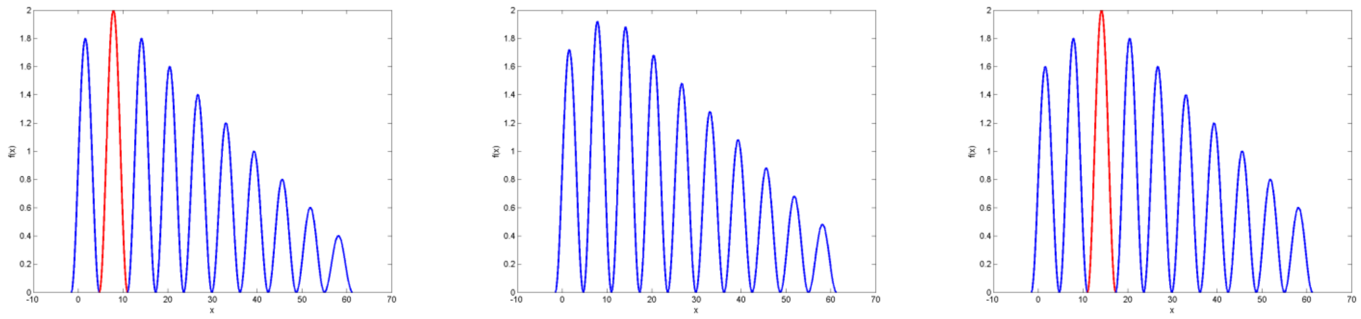
- optimization objectives, and/or
- problem-specific constraints, and/or
- environmental parameters and problem settings

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

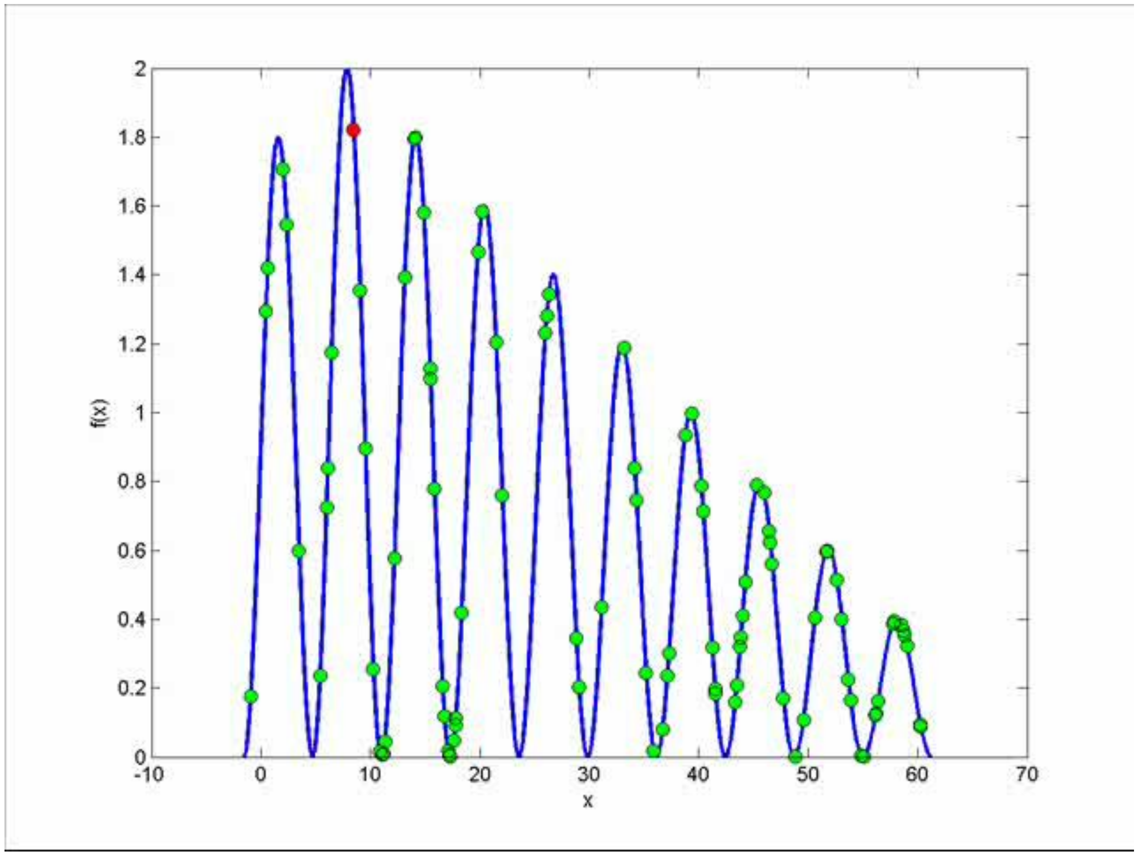
Ex.: Scheduling, manufacturing, trading with stochastic arrival of new tasks, machine faults, climatic change, market fluctuation, economic and financial factors.

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.



Evolutionaty Algorithms for Dynamic Opt.



Application Areas of EAs

EAs are popular for their

- simplicity,
- effectiveness,
- robustness.

Holland: "*It's best used in areas where you don't really have a good idea what the solution might be. And it often surprises you with what you come up with.*"

Well suited for black-box optimization

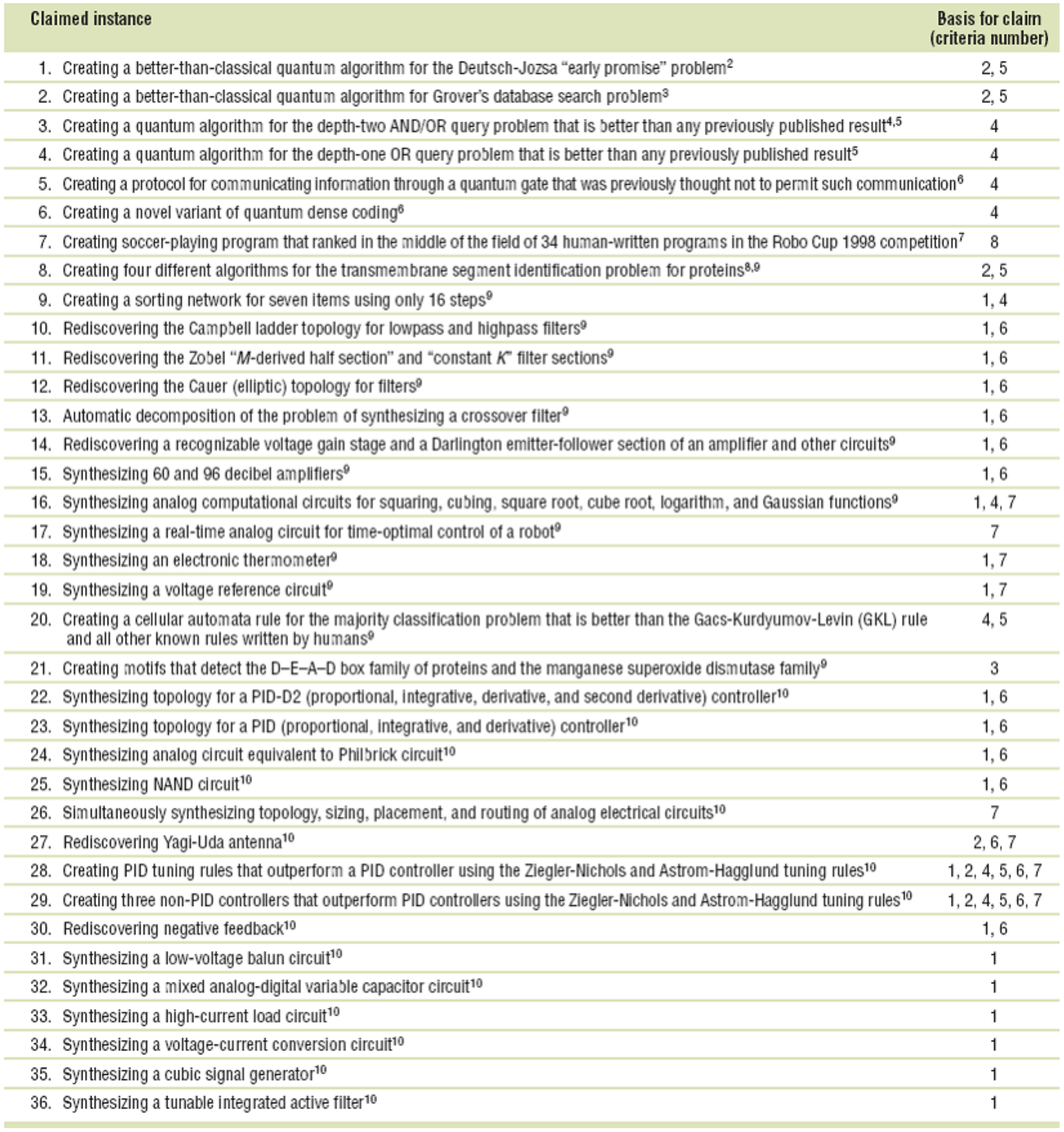
- No information about what the optimal solution looks like, no information about how to go about finding it in a principled way.
- Very little information about what the optimized function looks like.
- Very little heuristic information to go on.
- Brute-force search is out of the question because of the huge search space.

Applications

- control,
- engineering design,
- image processing,
- planning & scheduling,
- VLSI circuit design,
- network optimization & routing problems,
- optimal resource allocation,
- marketing,
- credit scoring & risk assessment,
- and many others.

Human-Competitive Results

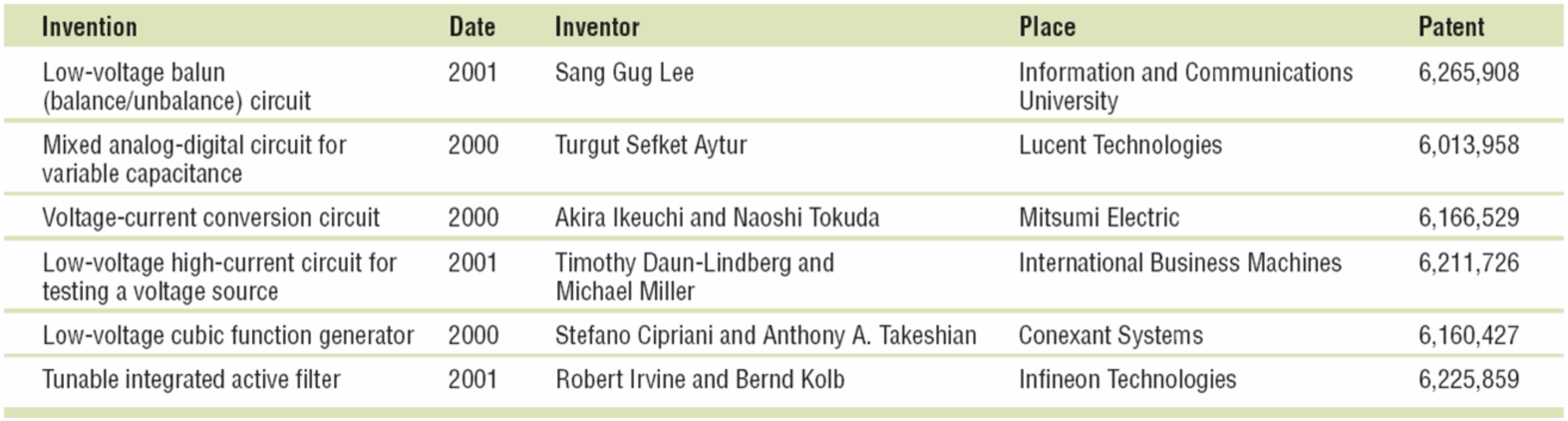
John R. Koza et al.: What's AI Done for Me Lately? Genetic Programming's Human-Competitive Results.



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)

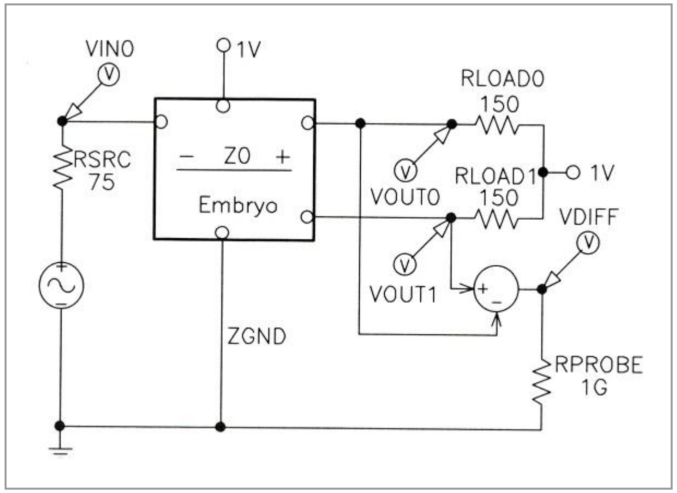
  for circuits that duplicate the patented inventions' functionality.

- 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.

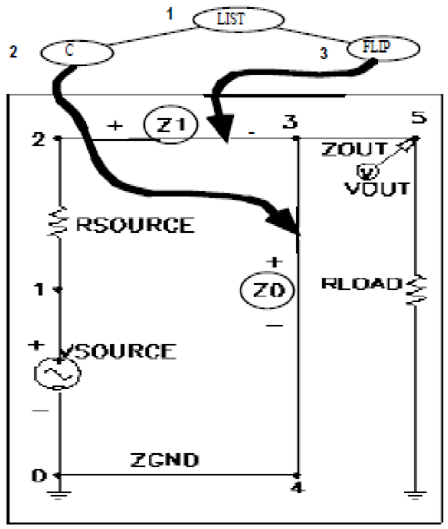


Test fixture

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.



- Component-creating function (inductor-, capacitor-, …creating functions)
- Connection-modifying function – series/parllel division function, delta-shaped composition

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**
- 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**
  - deme size: 10,000
  - 64 demes
  - Migration rate: 2%

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

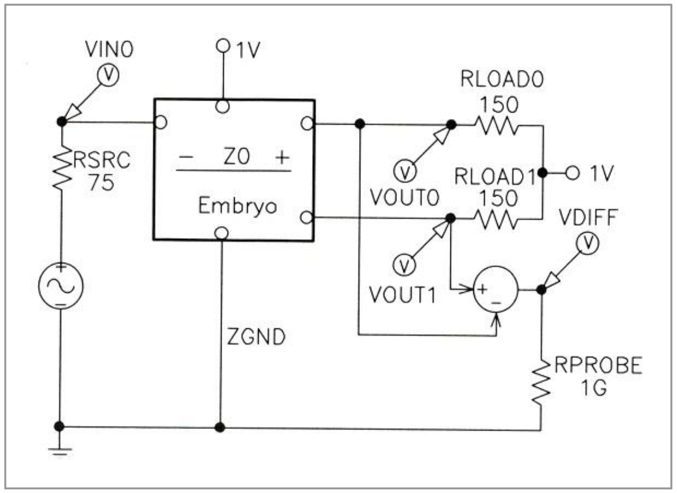
Low-Voltage Balun Circuit

- 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.

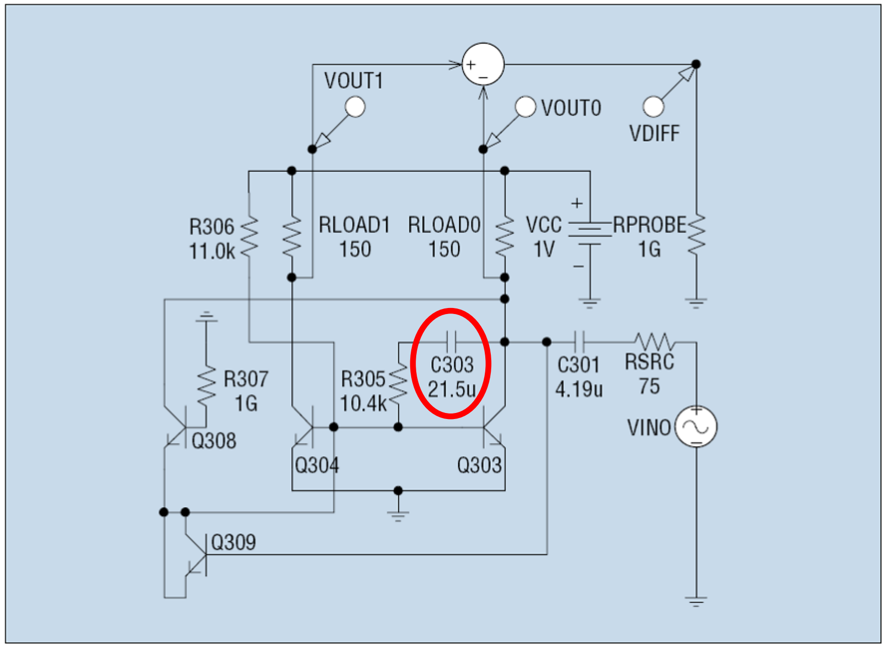
Genetically Evolved Low-Voltage Balun Circuit

- 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



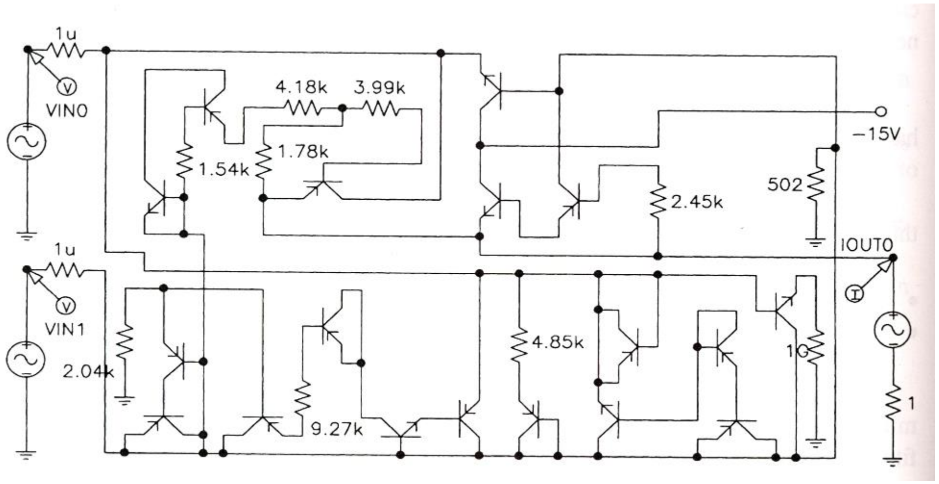
Evolved balun circuit



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**.

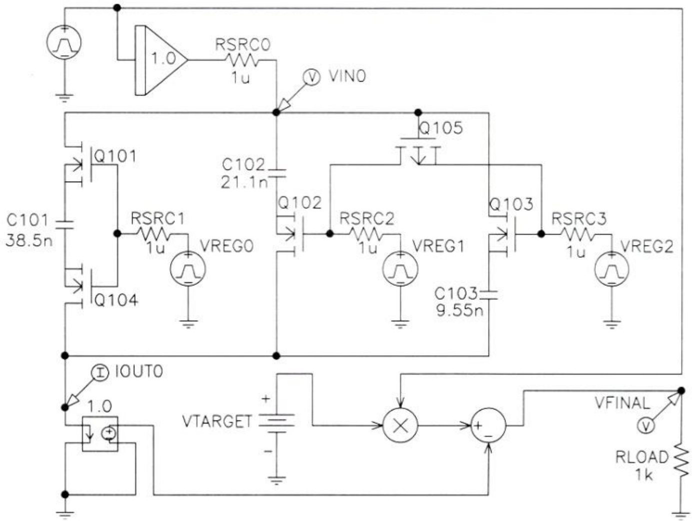


John R. Koza et al.: What's AI Done for Me Lately? Genetic Programming's Human-Competitive Results.

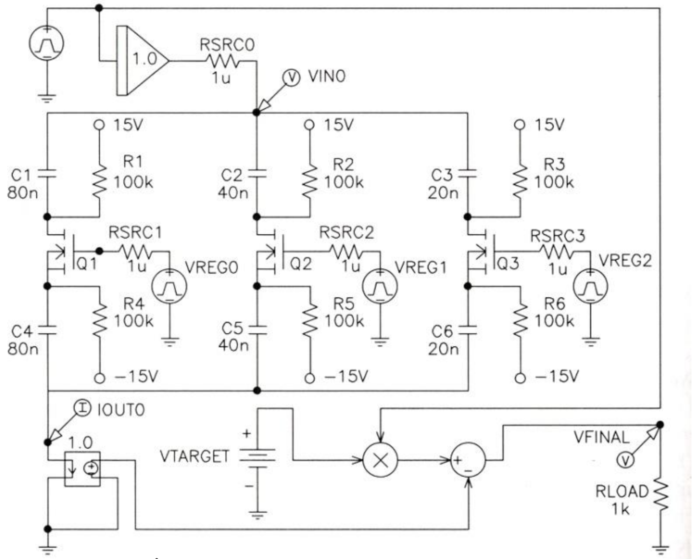
Mixed Analog-Digital Register-Controlled Variable Capacitor

- 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



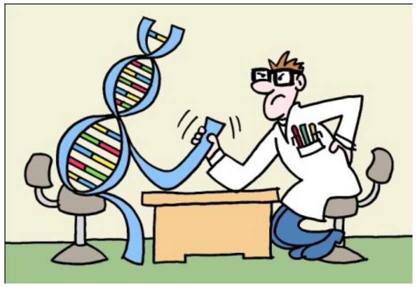
Patented circuit



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	- Game strategies
- Quantum circuit design	- Image processing
- Physics	- Antenna design
- Digital circuits/programs	- Classical optimization
- Chemistry	- …

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

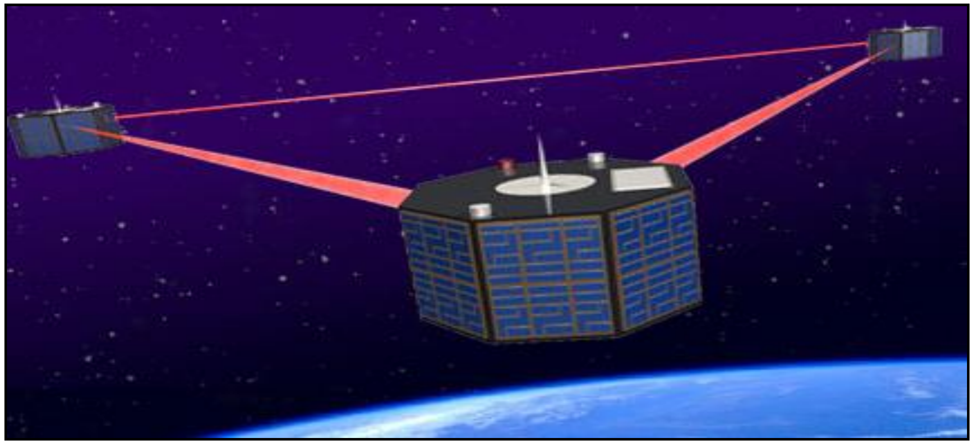
2004 Human-Competitive Awards in Genetic and Evolutionary Computation

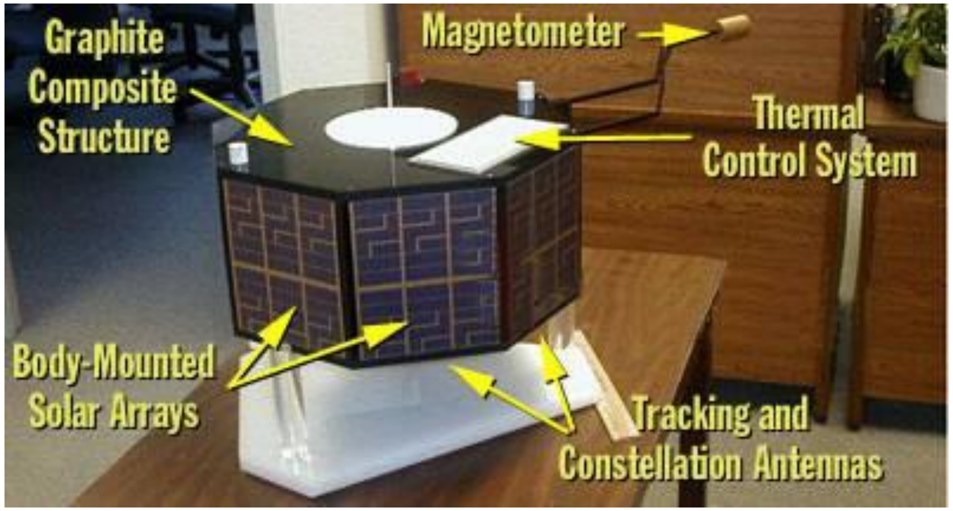
http://www.genetic-programming.org/gecco2004hc.html

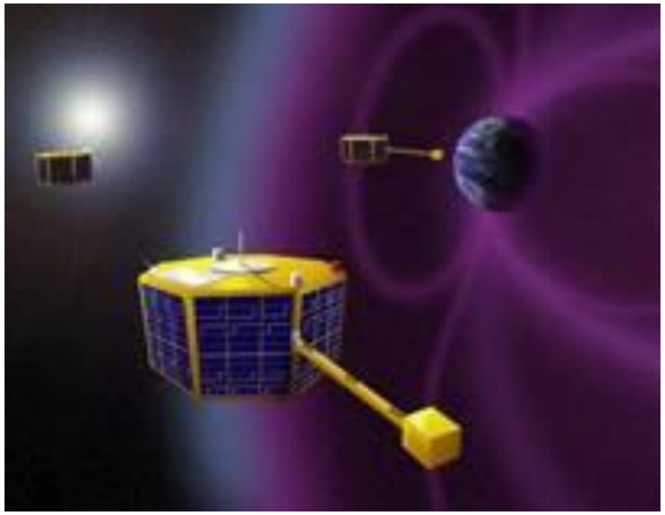
- $1500 – Gold
  - Jason D. Lohn, Gregory S. Hornby, Derek S. Linden: An Evolved Antenna for Deployment on NASA's Space Technology 5 Mission
  - Lee Spector: Automatic Quantum Computer Programming: A Genetic Programming Approach
- $500 – Silver
  - Alex Fukunaga: Evolving Local Search Heuristics for 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.




Graphite Composite Structure
Magnetometer
Thermal Control System
Body-Mounted Solar Arrays
Tracking and Constellation Antennas



© 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

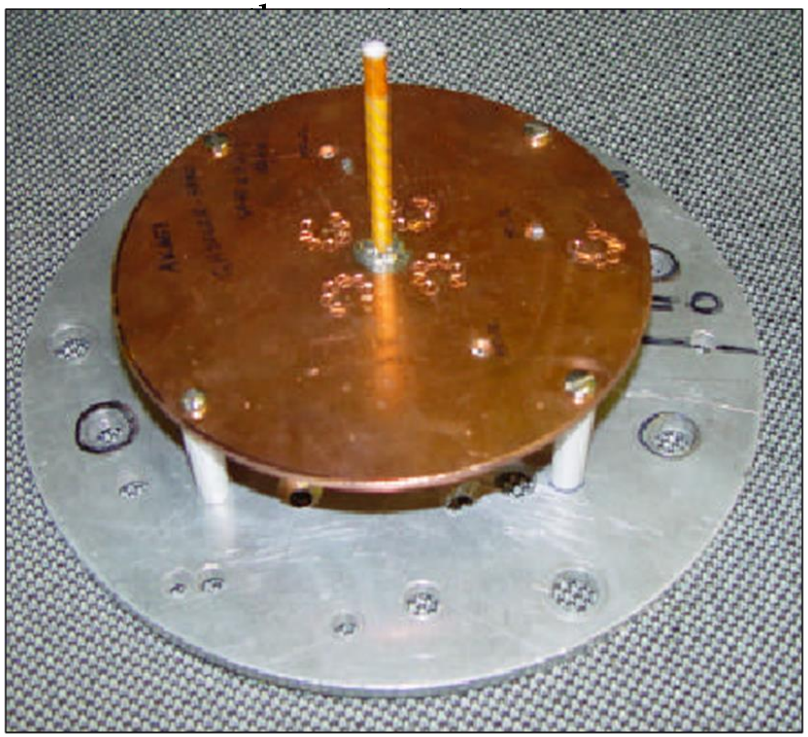
    >= 2dBic, 80 degrees

    >= 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

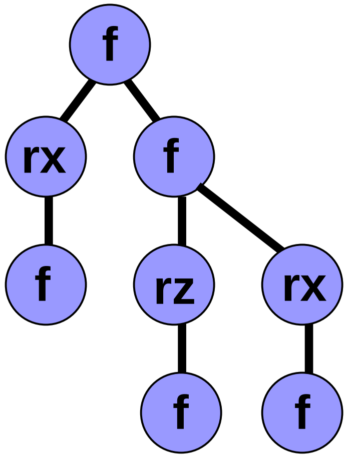
- **ST5 Quadrifilar Helical Antenna**
  - designed by a team of human designers

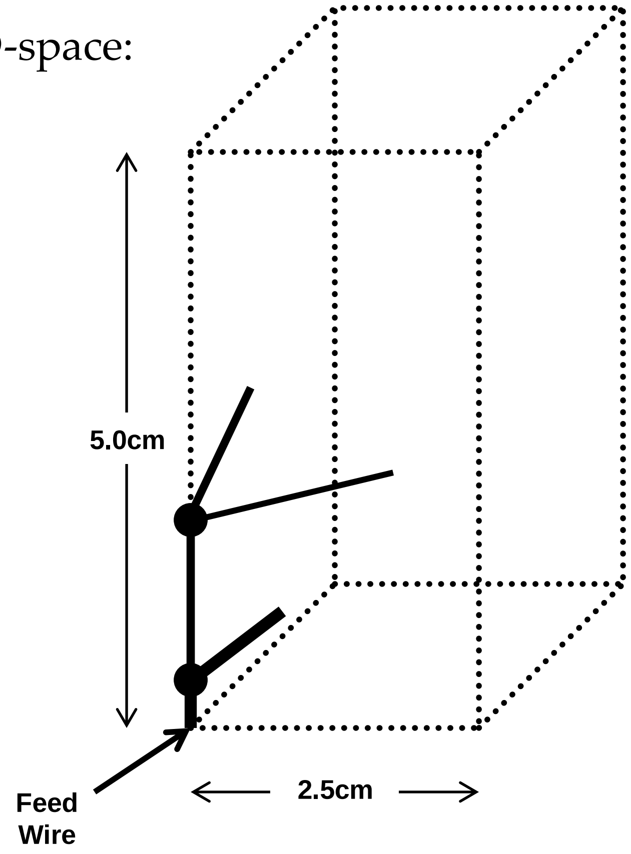


© Jason D. Lohn, Gregory S. Hornby, Derek S. Linden: Human-Competitive Results: Evolved Antennas for Deployment on NASA's ST5 Misson

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 3D-space:



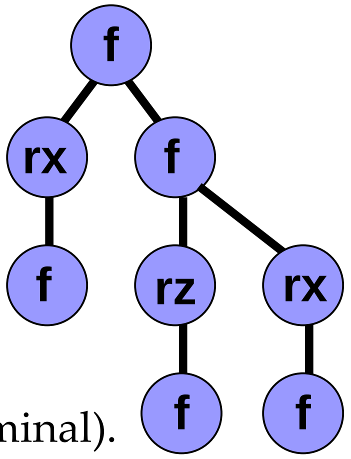


- Branching in genotype results in branching in wire form

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).

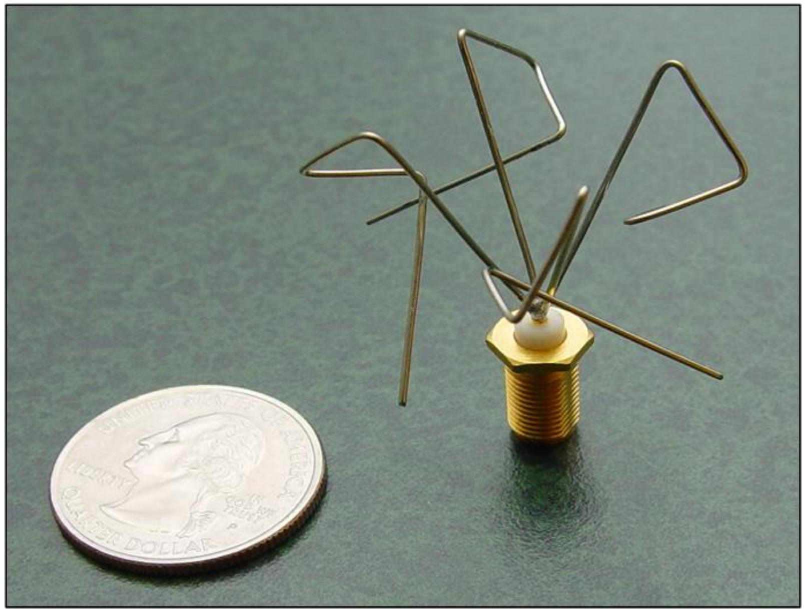


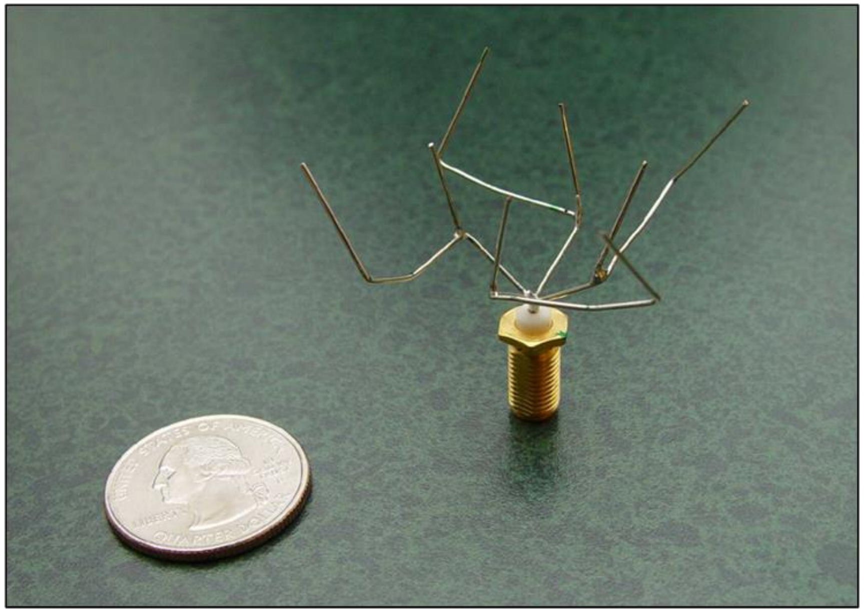
- **Fitness function** (to be minimized):

  F = VSWR_Score * Gain_Score * Penalty_Score

Evolved Antenna for Space Technology 5 mission

- 1st Set of Genetically Evolved Antennas





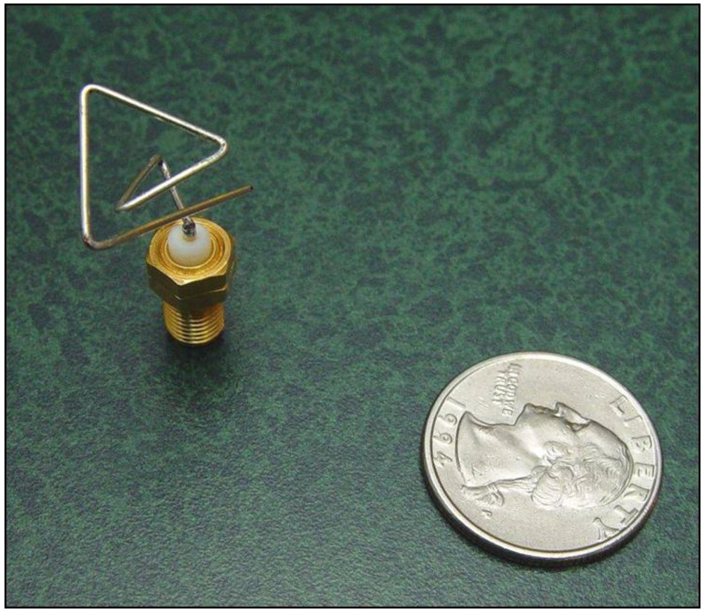
© Jason D. Lohn, Gregory S. Hornby, Derek S. Linden: Human-Competitive Results: Evolved Antennas for Deployment on NASA's Space Technology 5 Misson

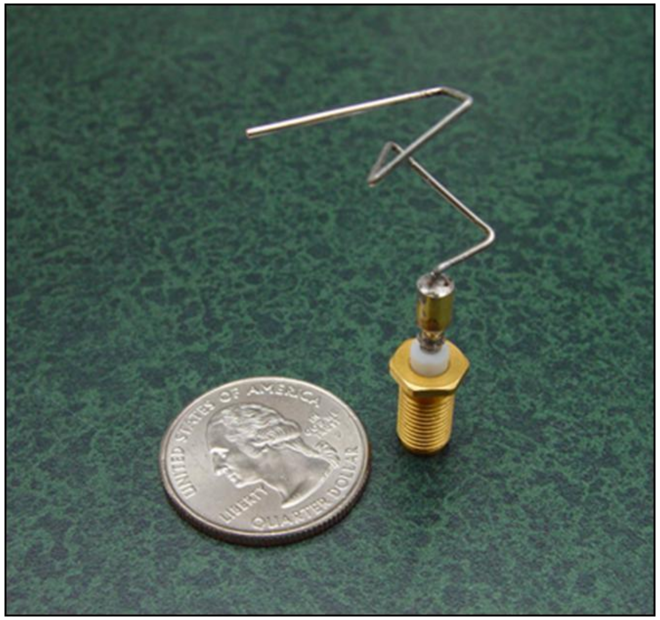
Non-branching:
ST5-4W-03

Branching:
ST5-3-10

Evolved Antenna for Space Technology 5 mission

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





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

EA 1 – Vector of Parameters

EA 2 – Constructive Process

Evolved Antenna for Space Technology 5 mission

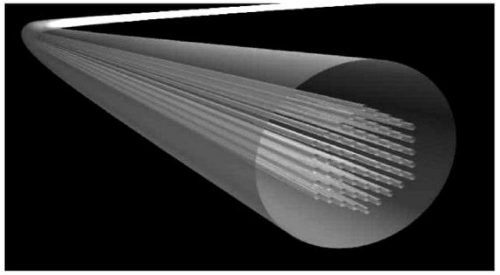
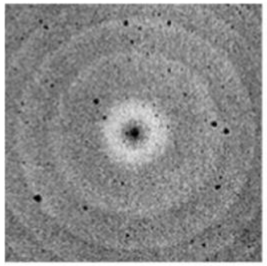
- **Conclusion**
  - Meets mission requirements
  - Better than conventional design
  - Successfully passed space qualification
  - **First Evolved Hardware in Space when mission launched in 2005**
- **Direct competition**: The antenna designed by the contracting team of human designers for the Space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

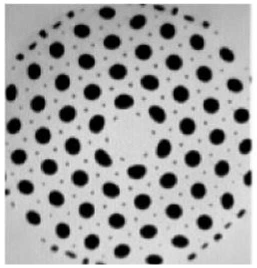
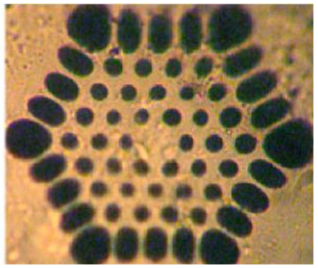
2007 Human-Competitive Awards in Genetic and Evolutionary Computation

- 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

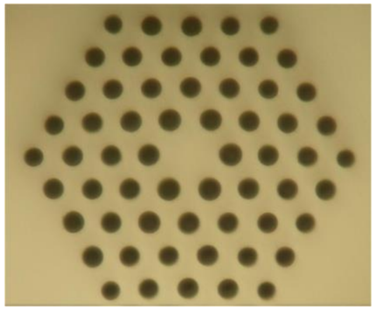
- Steven Manos, Leon Poladian, Maryanne Large: **Evolutionary Design of Microstructured Polymer Optical Fibres using an Artificial Embryogeny Representation**
  reference: http://www.genetic-programming.org/hc2007/cfe2007.html
- Applications of optical fibres
  - Long distance telecommunications
  -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

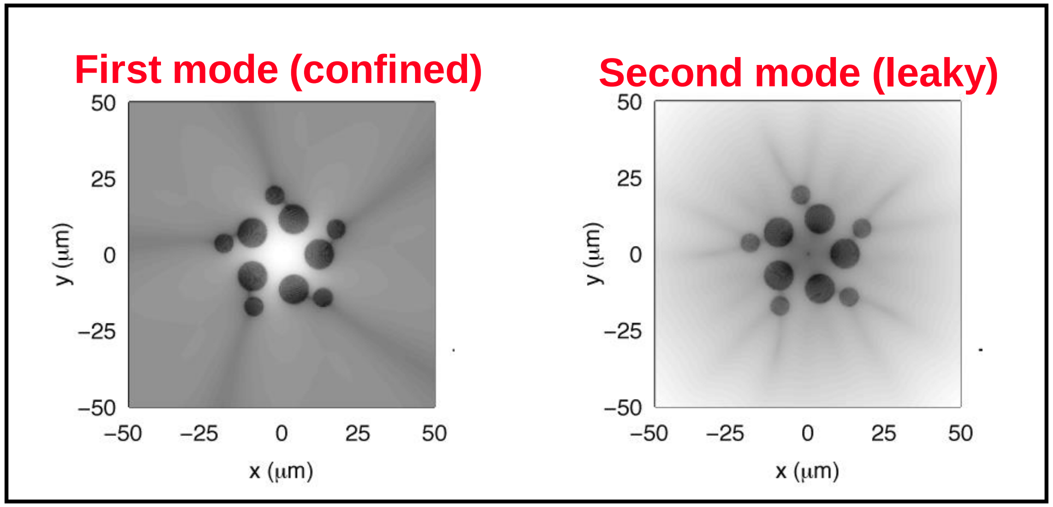



Single-moded fibres

Typical hexagonal design



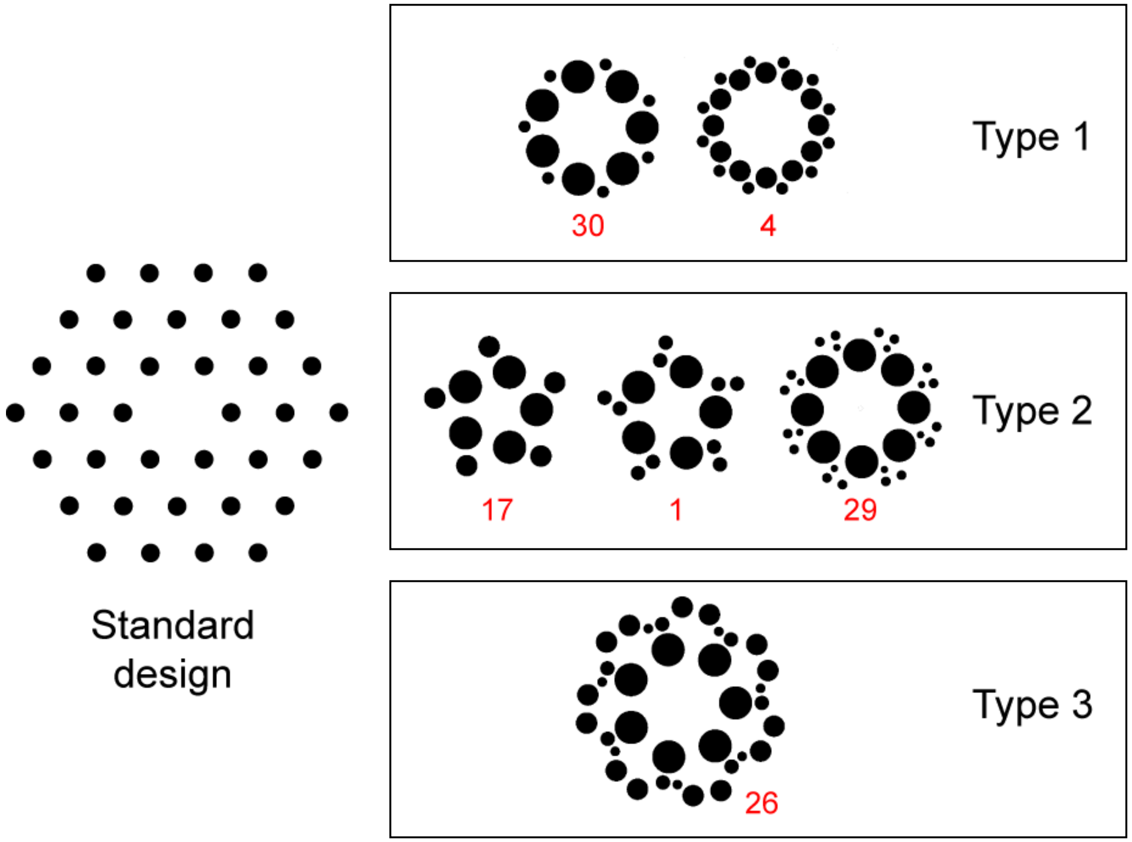
Standard design since the early 1990's



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.

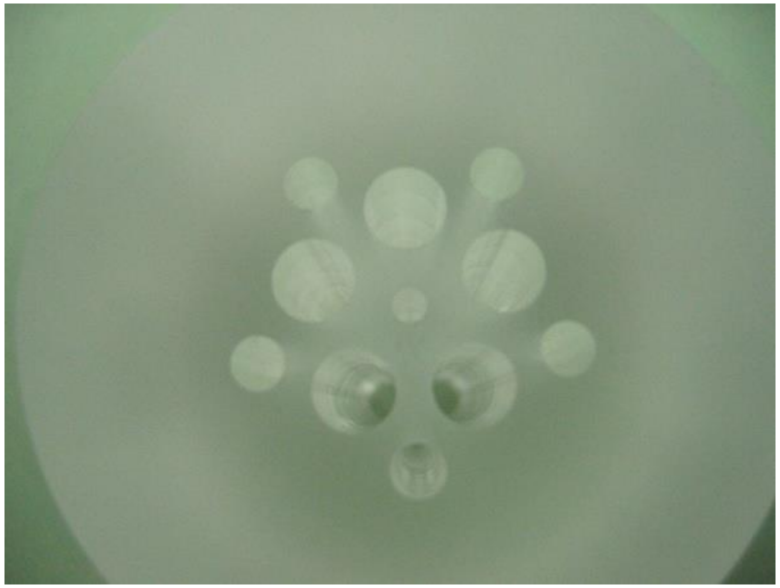
Evolved single-mode designs

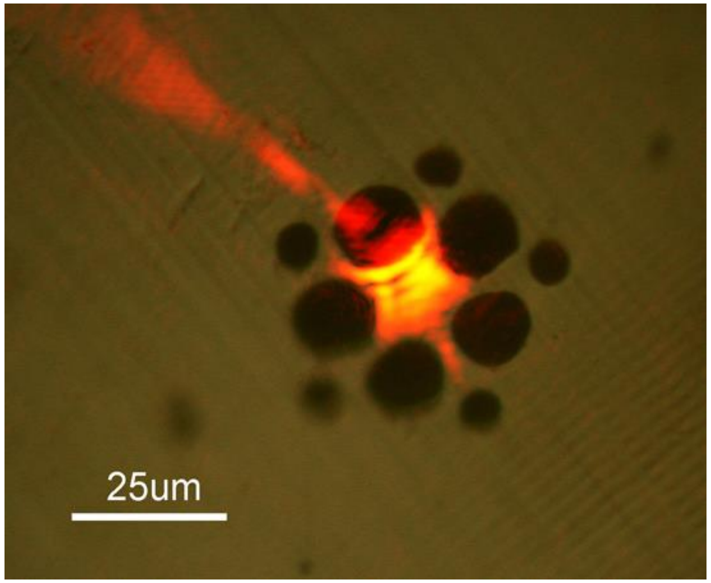


All designs have confined fundamental modes with $l_{c,1} < 10^{-1}$ **dB/m**, with losses more typically being $l_{c,1} < 10^{-3}$ **dB/m**.
The loss of the second mode $l_{c,2} > 10^{4}$ **dB/m** in all cases.

All single-moded, yet phenotypically different.

Manufactured single-mode MPOF



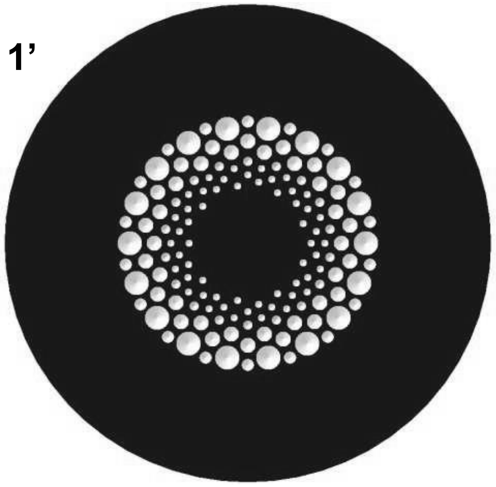


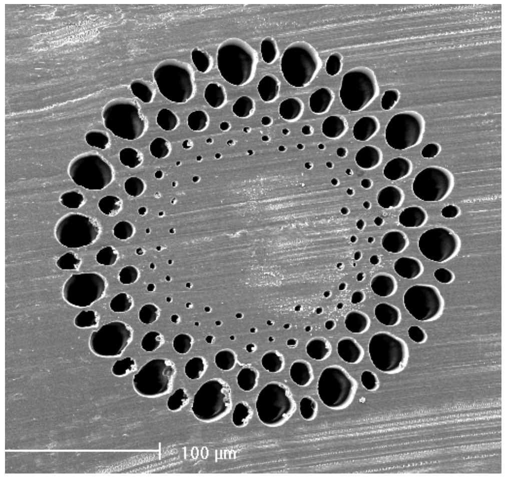
- **Evolved designs are simpler than previous designs, and easier to manufacture.**
- Provided us with a rich set of never before seen single-moded microstructured fibre designs to investigate further.

A different fitness function

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

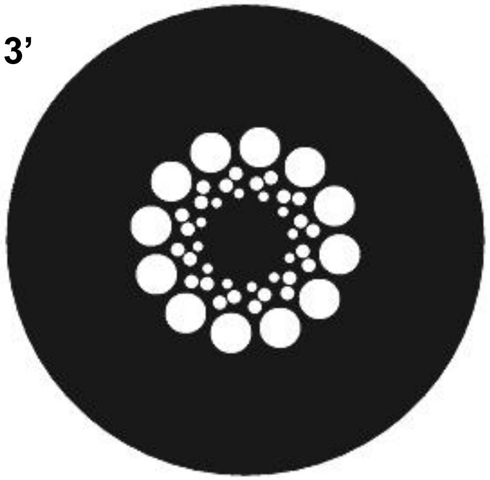
‘GIMP 1’

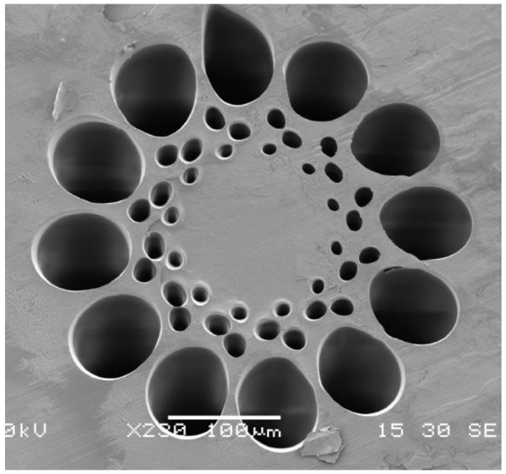




Hand-designed fibre

‘GIMP 3’

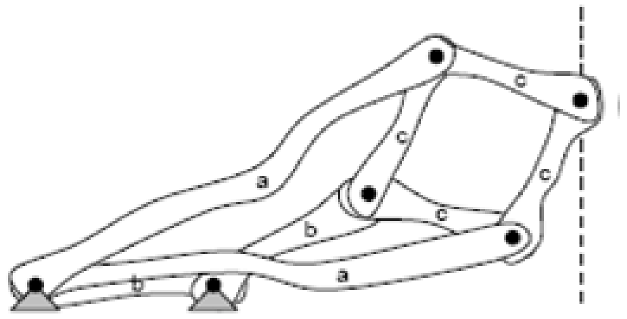




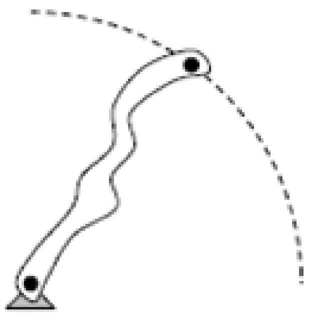
Patented GA-designed fewer holes, easier to manufacture.

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!

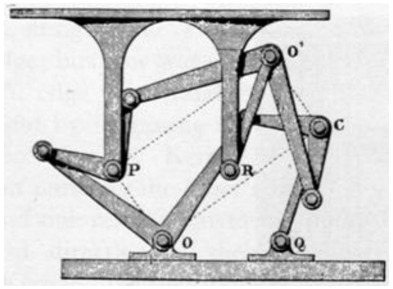


line

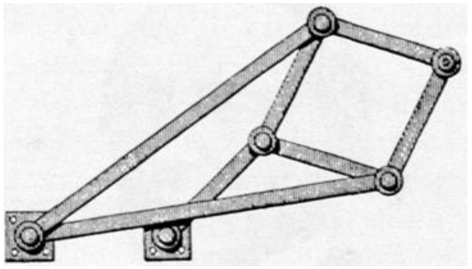

circle

How to Draw a Straight Line Using a GP

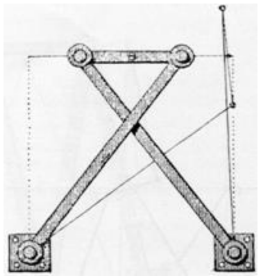
- Some key straight-line mechanisms



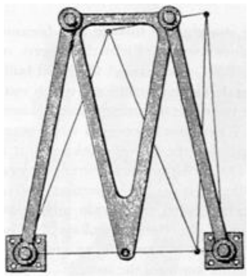
Silverster-Kempe's
(1877)



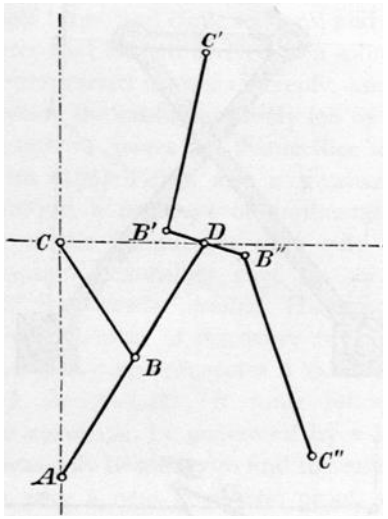
Peaucelier
(1873)



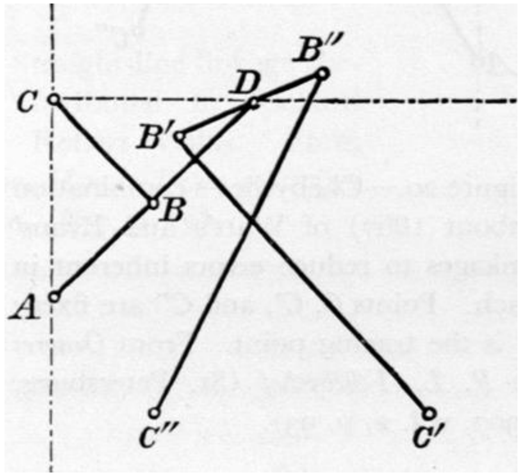
Chebyshev
(1867)



Robert
(1841)



Chebyshev
(1867)

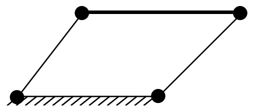


Chebyshev-Evans
(1907)

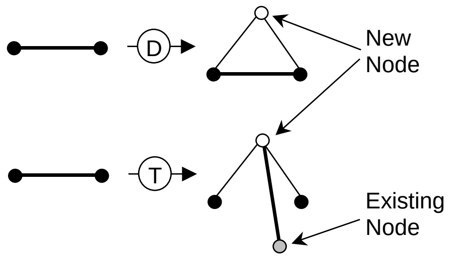
See
http://kmoddl.library.cornell.edu

How to Draw a Straight Line Using a GP

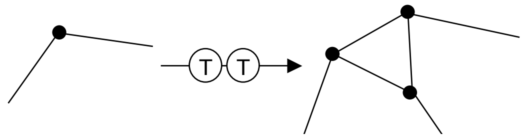
- **Top down encoding of a mechanism**



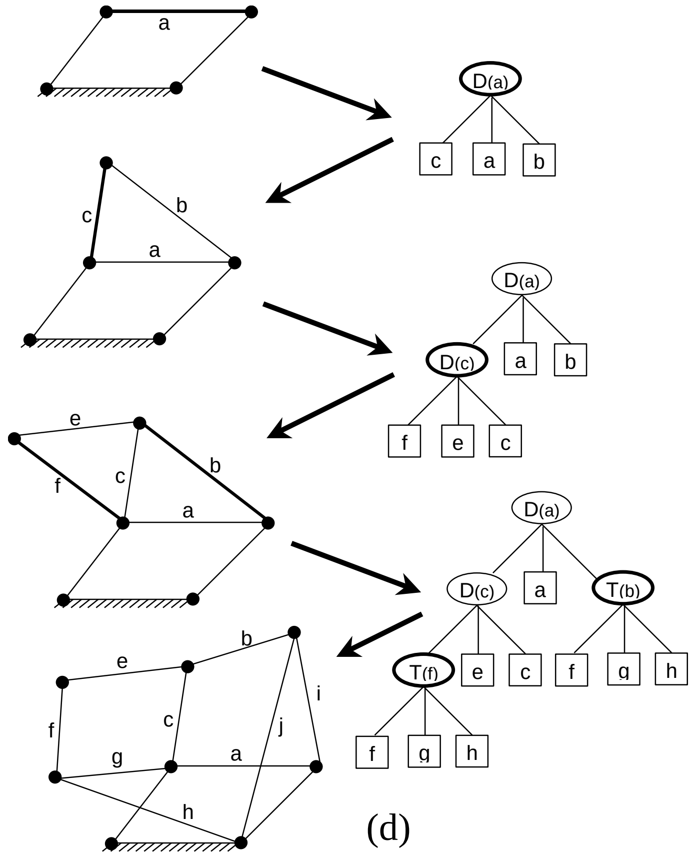
Start with Embryo with desired # of DoF, e.g. a four-bar mechanism (1 DoF)



Two variation operators maintain DoF



E.g. Transform dyad into tryad

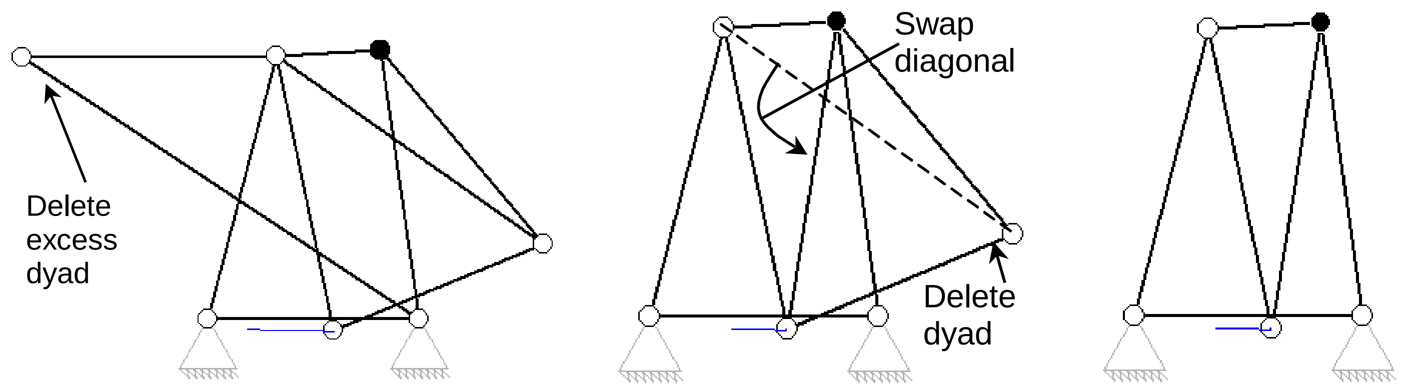


(d)

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

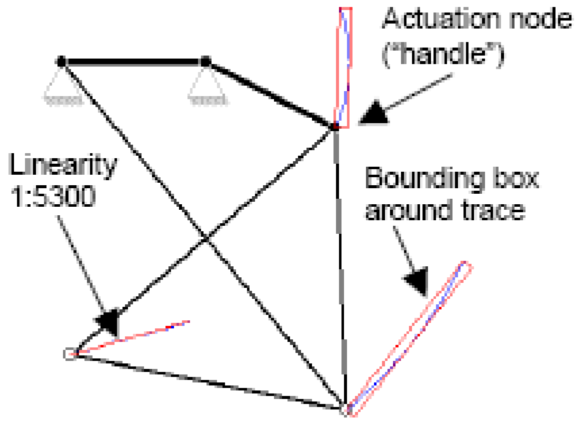
How to Draw a Straight Line Using a GP

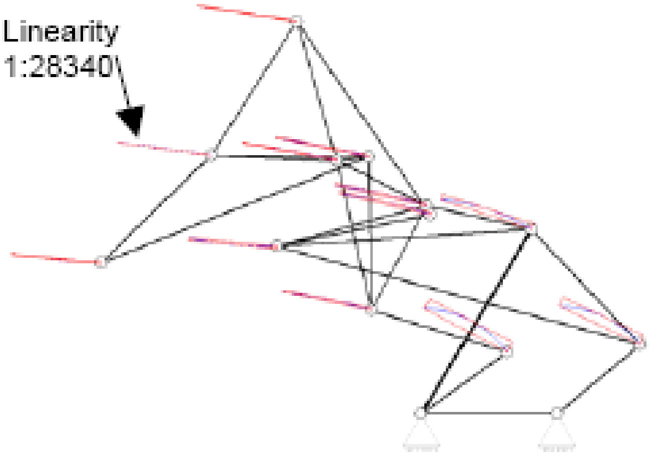
- **Comparison of mechanisms** can be difficult
  - Equivalent mechanisms may appear very different
  - Masked by excess and redundant topology
- **Two transformations** allow moving in "neutral pathways" of mechanisms
  - Rigid diagonal swap
  - Redundant dyad removal/addition



How to Draw a Straight Line Using a GP

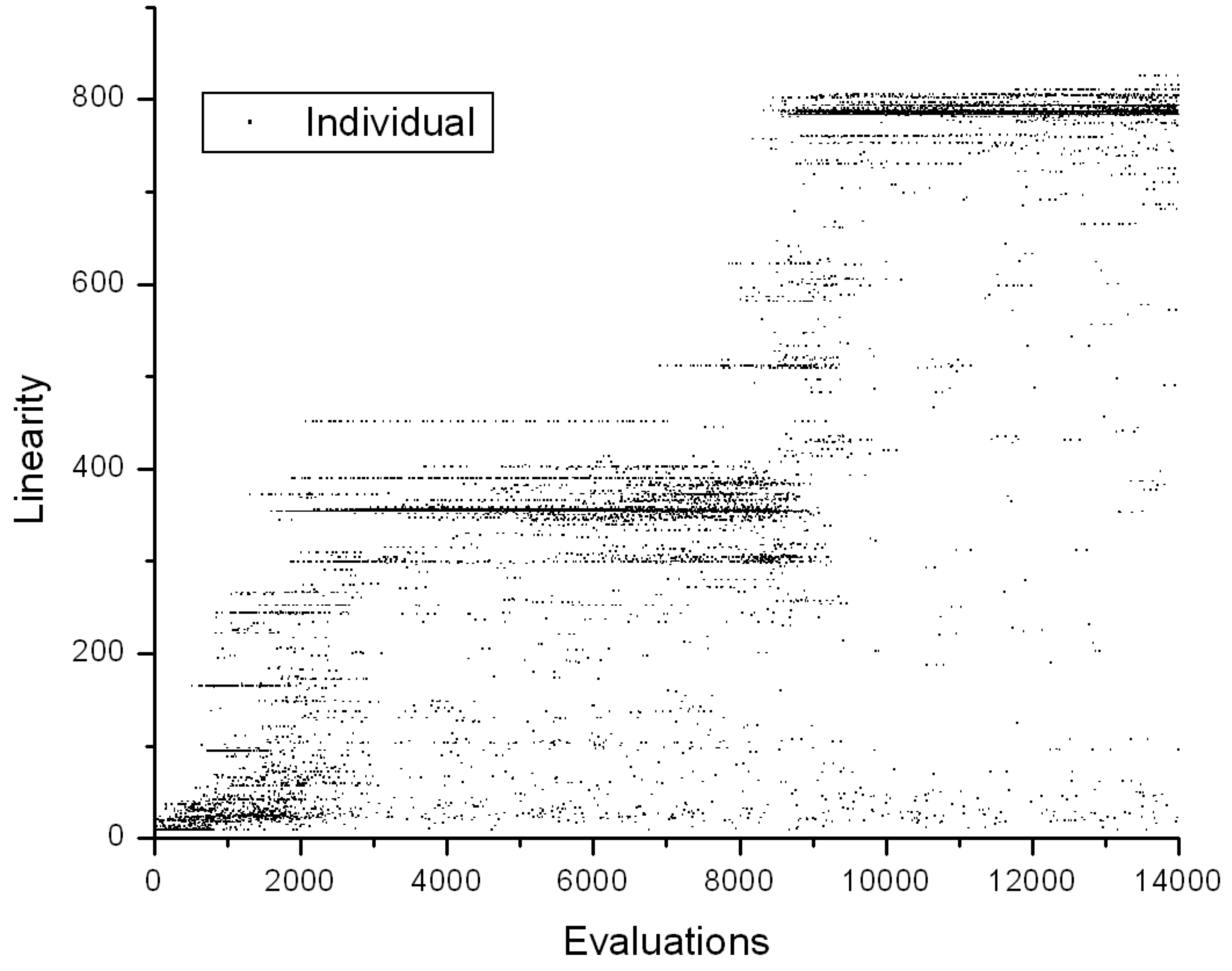
- Used **GP with Top-down tree encoding and 2-bar or 4-bar embryo**
  - Population size: 100
  - Crossover 90%
  - Mutation 10% (Node positions, Operator types)
- Selection: **Stochastic Universal Sampling**
- **Evaluation of an evolved straight-line mechanism**
  - The mechanism is actuated at an arbitrary 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



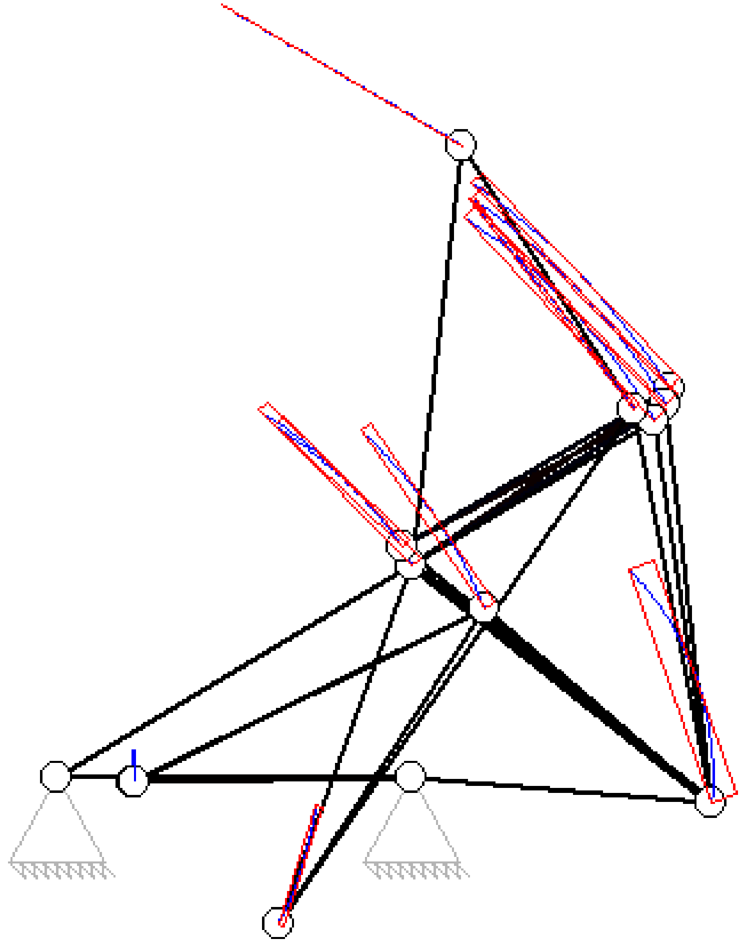


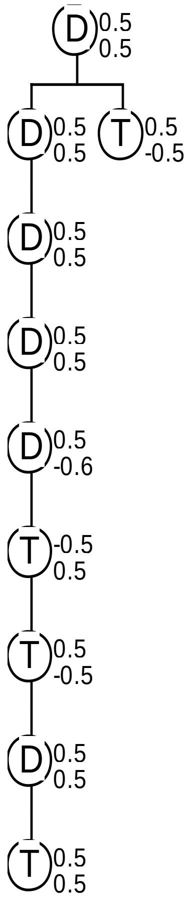
How to Draw a Straight Line Using a GP

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



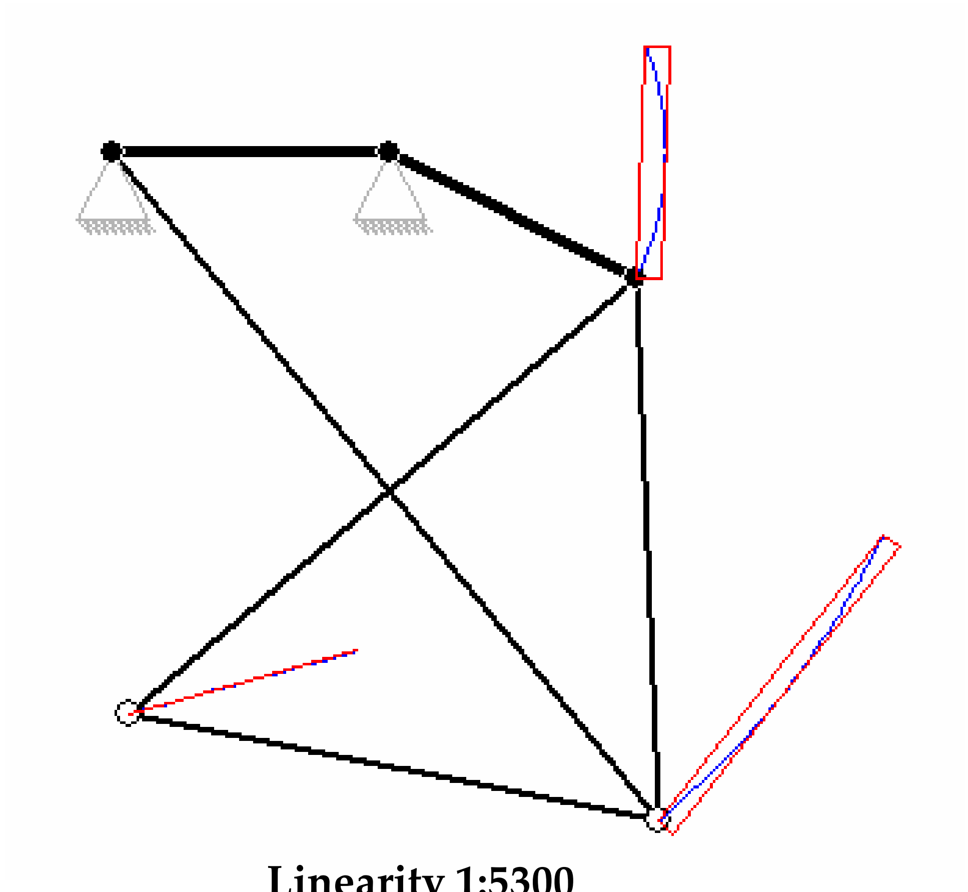
Some results





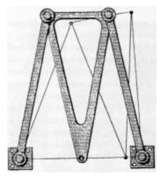
Linearity 1:4979

Some results

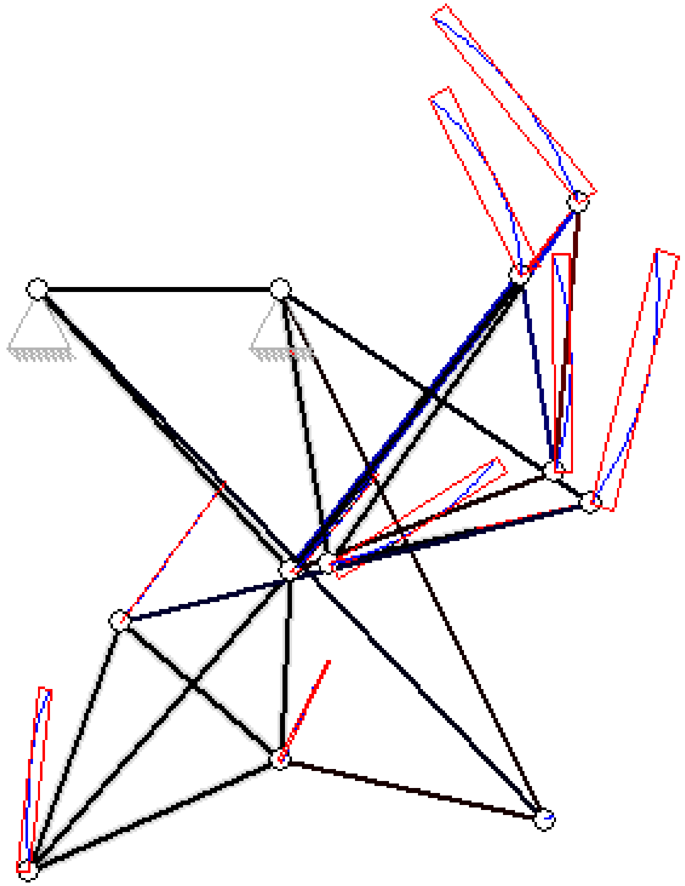


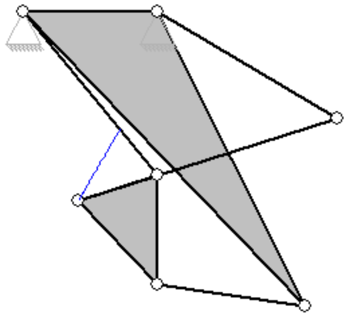
Linearity 1:5300

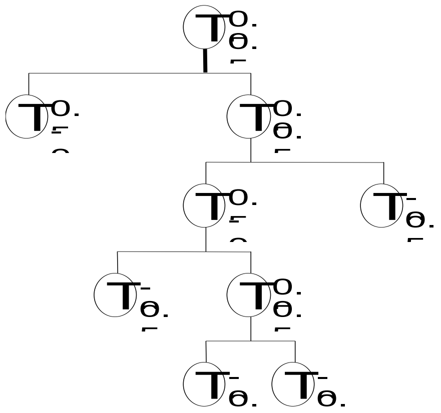
Infringes on Robert's Linkage (1841)
Published: Kempe A. B., (1877), *How To Draw A Straight Line*, London



Some results

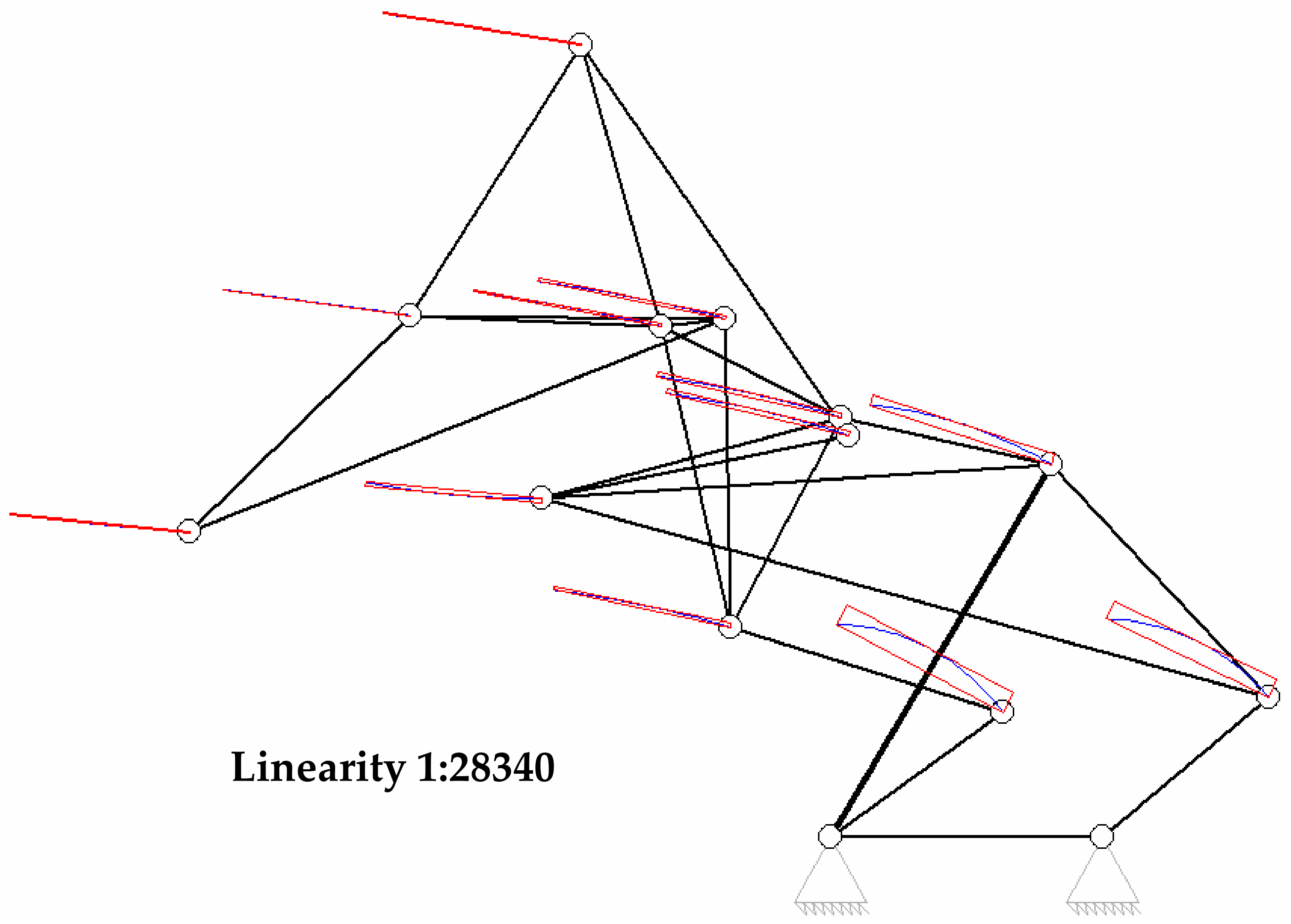






Linearity 1:12819

Some results



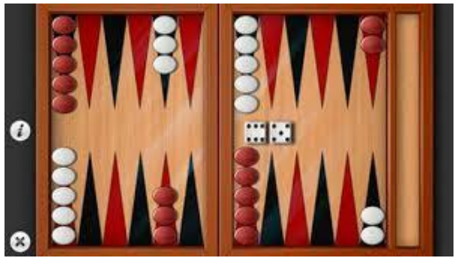
2005 Human-Competitive Awards in Genetic and Evolutionary Computation cont.

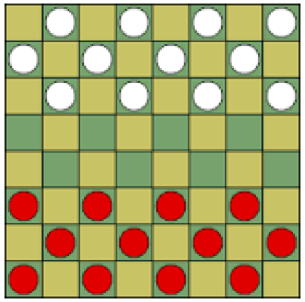
- $1000 – Silver
  - Richard J. Terrile et al.:
    - **Evolutionary Computation Technologies for the Automatic Design of Space Systems,**
    - **Evolutionary Computation applied to the Tuning of MEMS gyroscopes,**
    - **Multi-Objective Evolutionary Algorithms for Low-Thrust Orbit Transfer Optimization**
- $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 Grasemann, Risto Miikkulainen: **Effective Image Compression using Evolved Wavelets**

Moshe Sipper: Evolved to Win

Board games

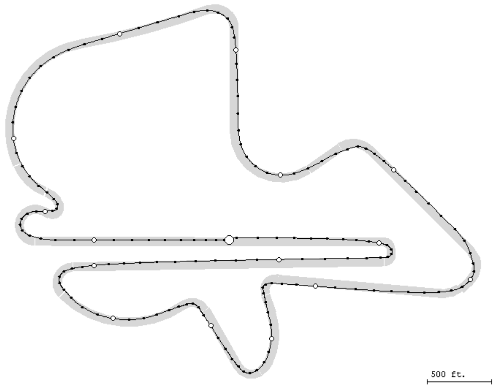
- Checkers
- Chess endgames
- Backgammon





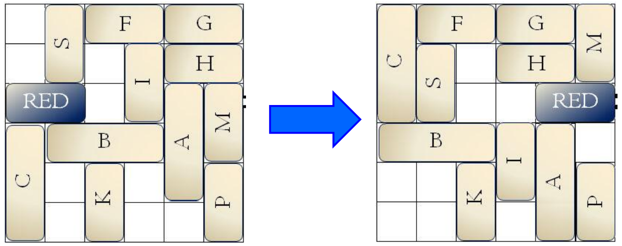
Simulation games

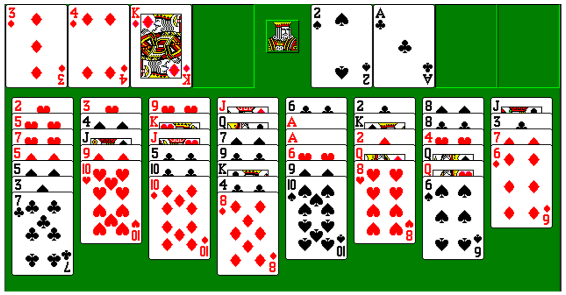
- Robocode
- Robot Auto Racing Simulator



Puzzles

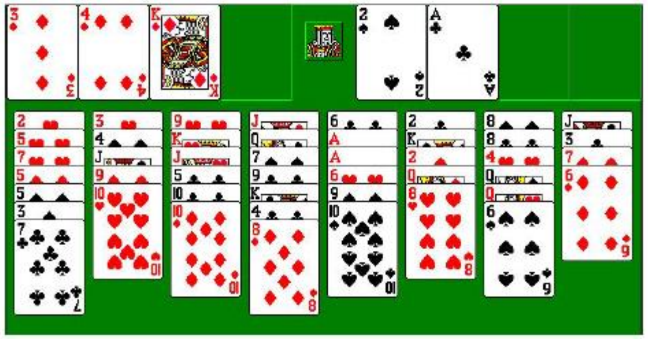
- Rush hour
- FreeCell





```
(% (% (% (% (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.

Source: Elyasaf, A. at all.: Evolutionary Design of FreeCell Solvers. 2013

Automatically Finding Patches Using GP

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 occurences in the tescases.
- 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

```c
/* requires: a >= 0, b >= 0 */
void gcd(int a, int b) {
  if (a == 0) {
    printf("%d", b);
  }
  while (b != 0)
    if (a > b)
      a = a - b;
    else
      b = b - a;
  printf("%d", a);
  exit(0);
}
```

Automatically Finding Patches Using GP

- Example: Euclid's greatest common divisor

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

Original program

```c
/* requires: a >= 0, b >= 0 */
void gcd(int a, int b) {
  if (a == 0) {
    printf("%d", b);
  }
  while (b != 0)
    if (a > b)
      a = a - b;
    else
      b = b - a;
  printf("%d", a);
  exit(0);
}
```

Primary repair

```c
void gcd_3(int a, int b) {
  if (a == 0) {
    printf("%d", b);
    exit(0);                  // inserted
    a = a - b;                // inserted
  }
  while (b != 0)
    if (a > b)
      a = a - b;
    else
      b = b - a;
  printf("%d", a);
  exit(0);
}
```

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

```
/* requires: a >= 0, b >= 0 */
void gcd(int a, int b) {
  if (a == 0) {
    printf("%d", b);
  }
  while (b != 0)
    if (a > b)
      a = a - b;
    else
      b = b - a;
  printf("%d", a);
  exit(0);
}
```

Primary repair

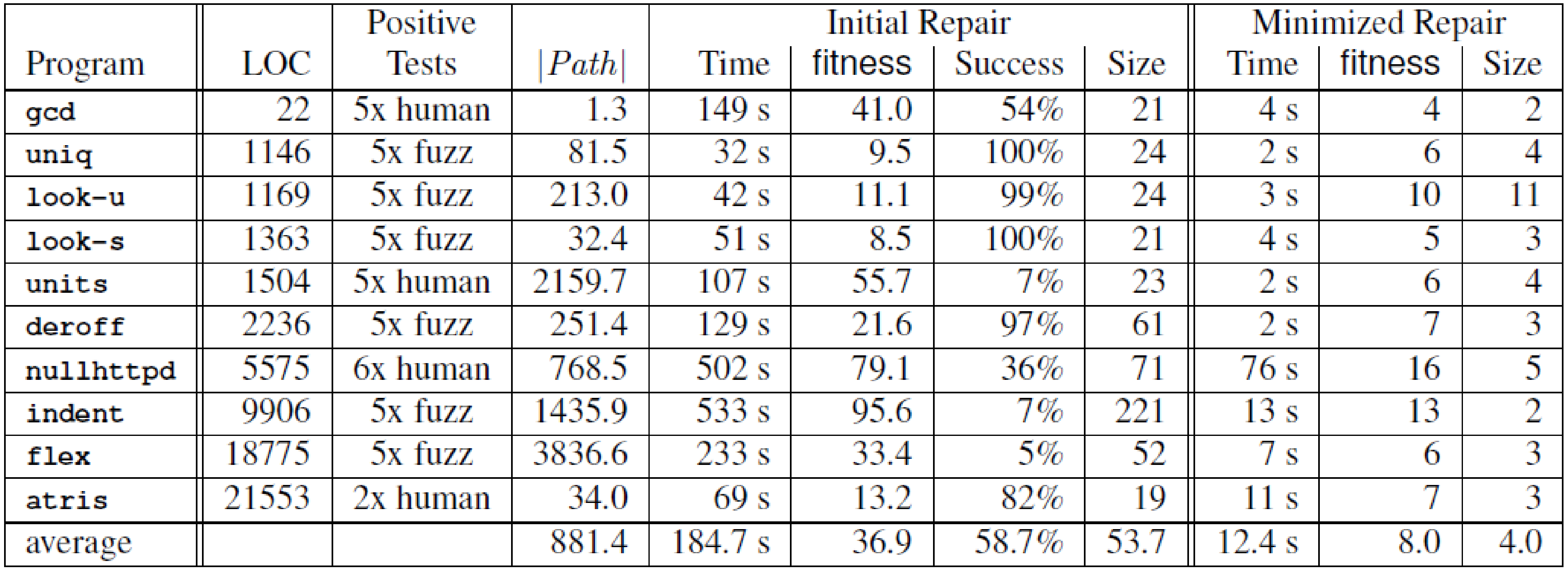
```
void gcd_3(int a, int b) {
  if (a == 0) {
    printf("%d", b);
    exit(0);                  // inserted
    ~~a = a - b;                // inserted~~
  }
  while (b != 0)
    if (a > b)
      a = a - b;
    else
      b = b - a;
  printf("%d", a);
  exit(0);
}
```

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 Time	Initial Repair fitness	Initial Repair Success	Initial Repair Size	Minimized Repair Time	Minimized Repair fitness	Minimized Repair 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

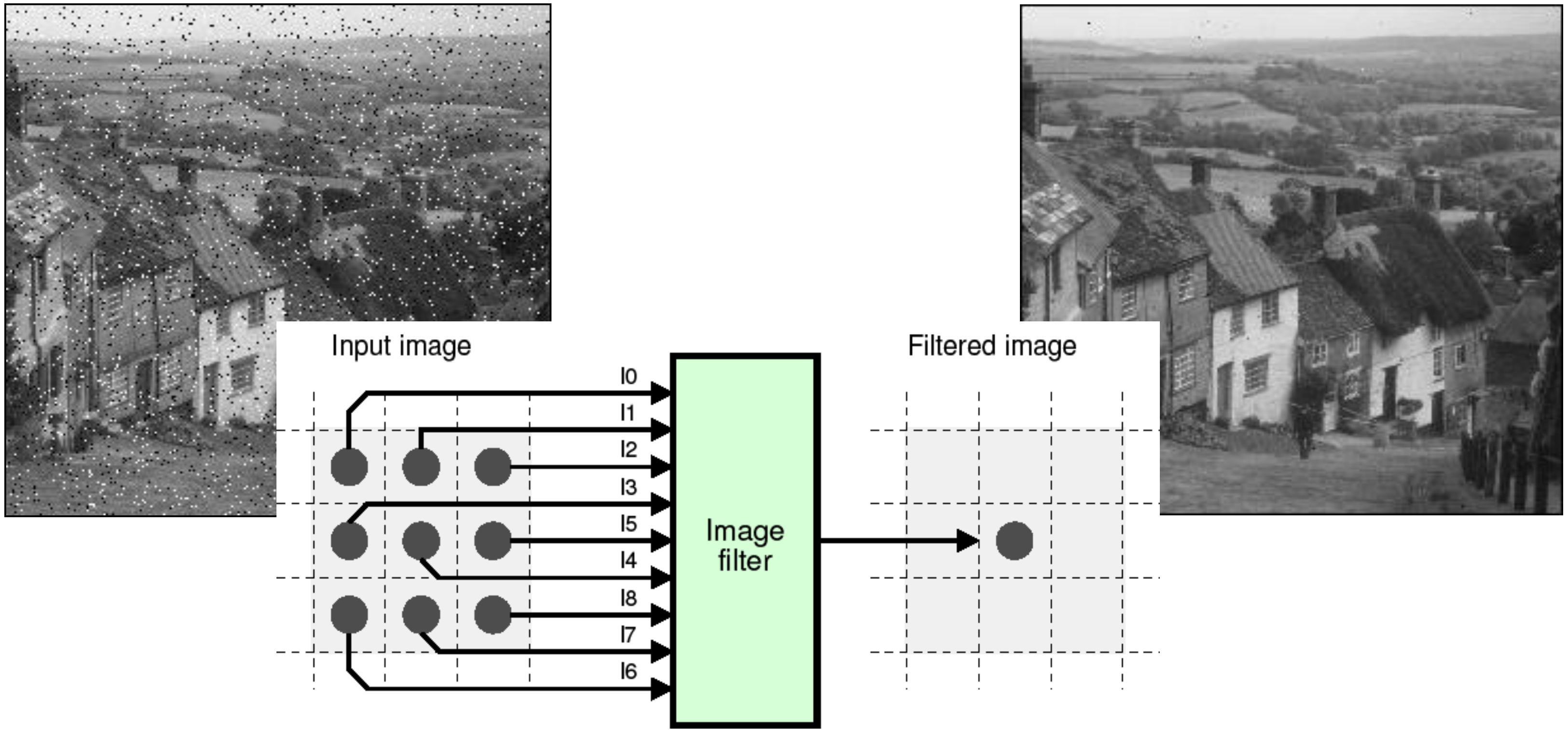
Weimer W. et al.: Automatically Finding Patches Using Genetic Programming
Forrest S. et al.: A Genetic Programming Approach to Automated Software Repair, GECCO 2009

BRNO UNIVERSITY OF TECHNOLOGY

FACULTY OF INFORMATION TECHNOLOGY

Evolutionary Design of Image Filters

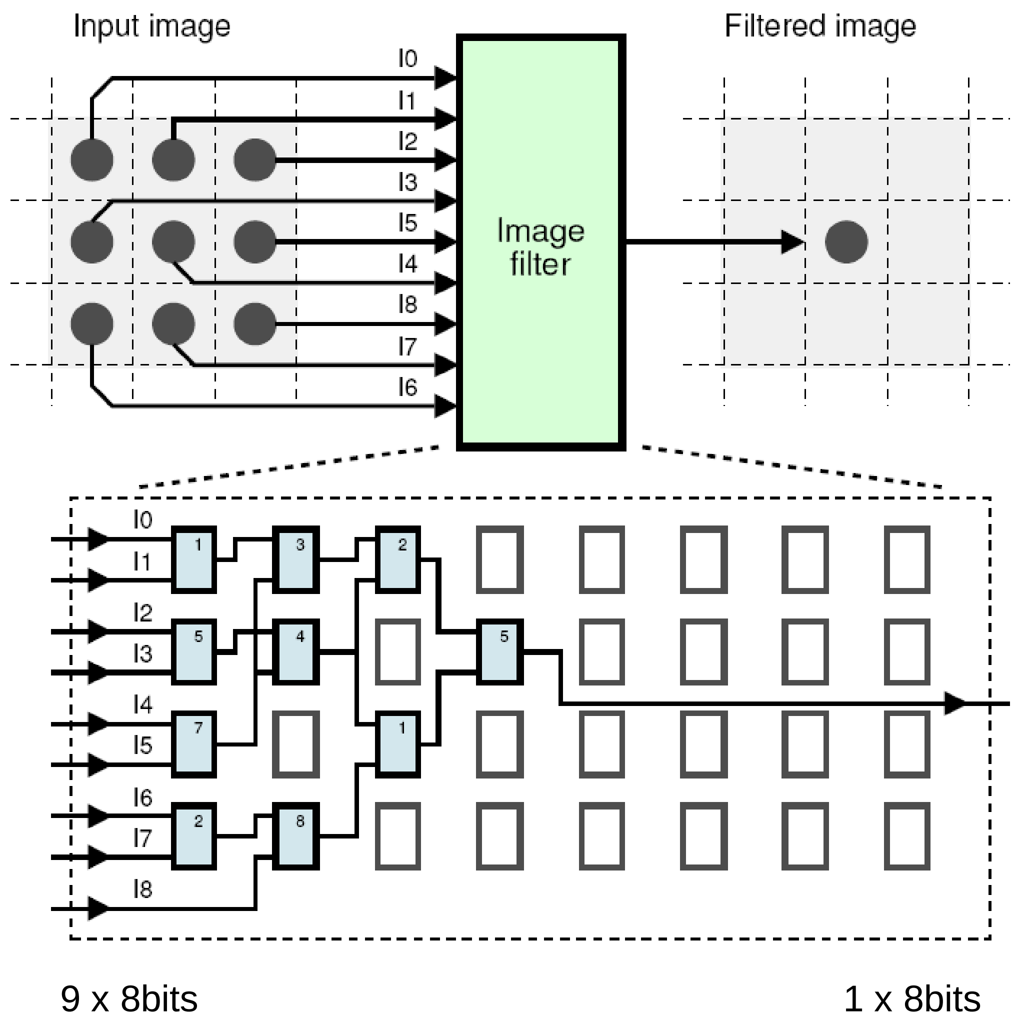
Evolutionary design of image filters



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.

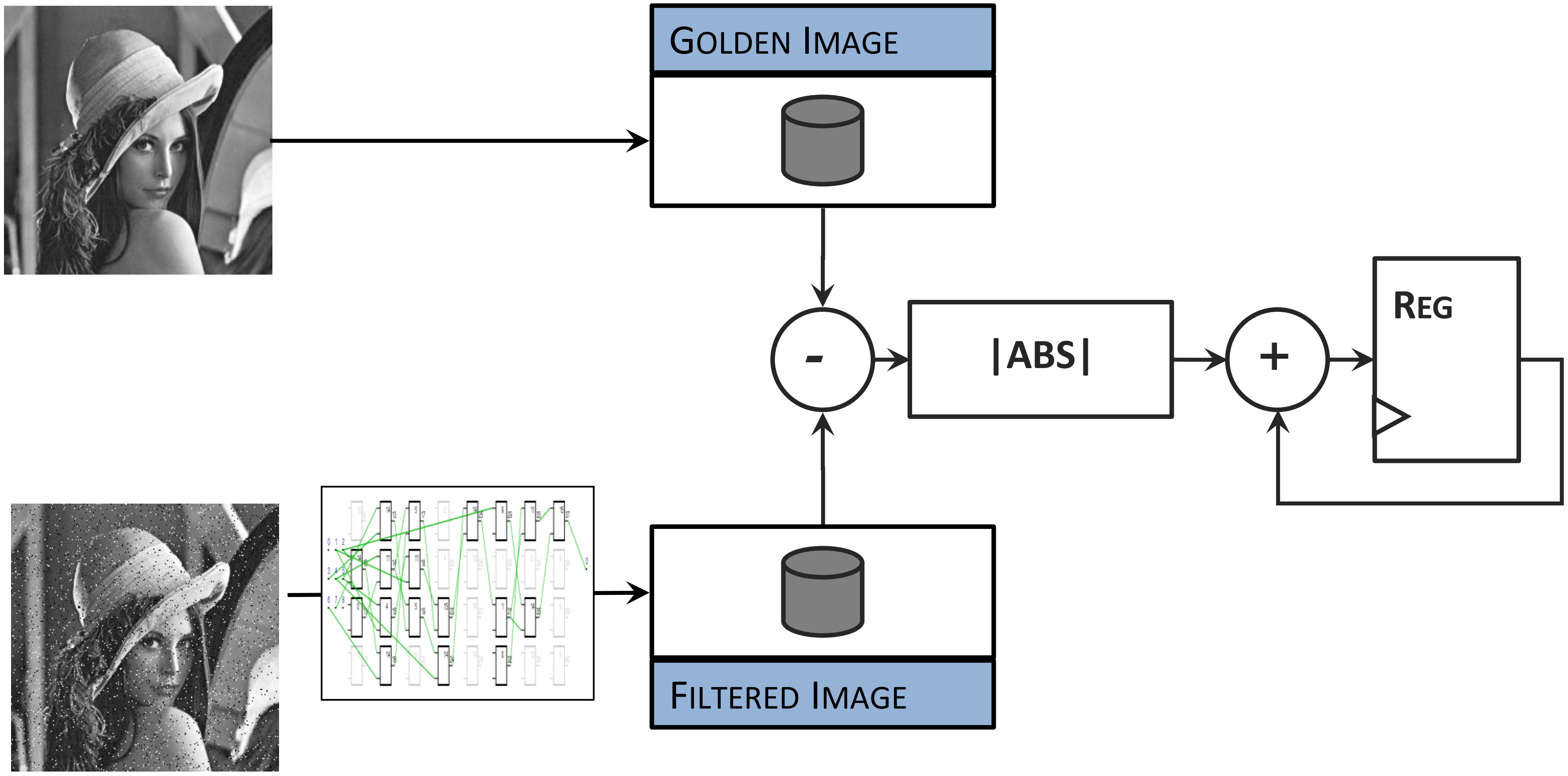
- PE over 8 bits:
  - Minimum
  - Maximum
  - Average
  - Constants
  - logic operators
  - shift

Sekanina L.: EvoAISP 2002

Fitness function

$$MAE = \frac{1}{RC}\sum_{r=0}^{R-1}\sum_{c=0}^{C-1}\left|I(r,c) - K(r,c)\right|$$

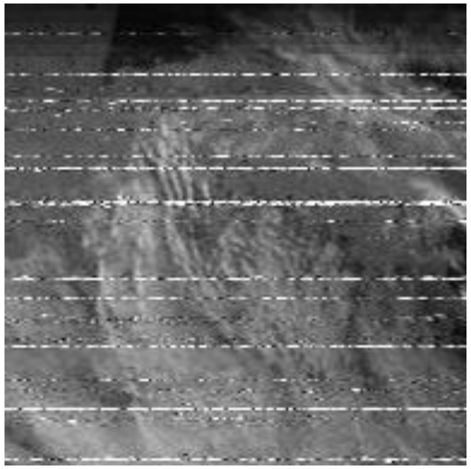
I: golden image
K: filtered image
R: rows
C: columns

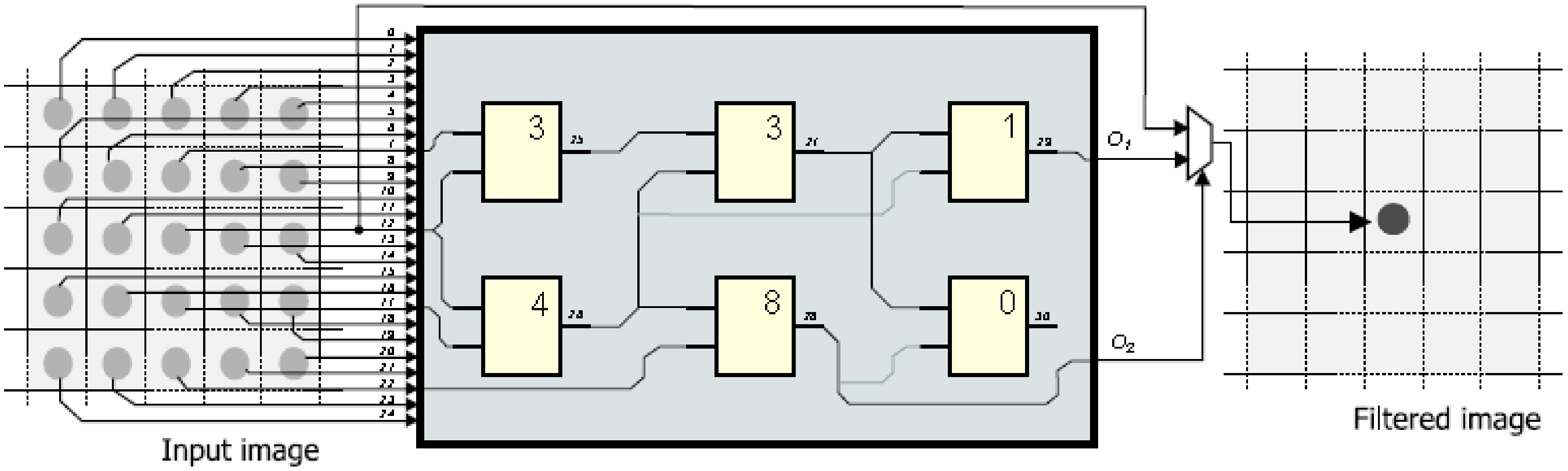


BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Burst Noise Filtering

- switching image filters
- 5 x 5 filtering window





Burst Noise Filtering

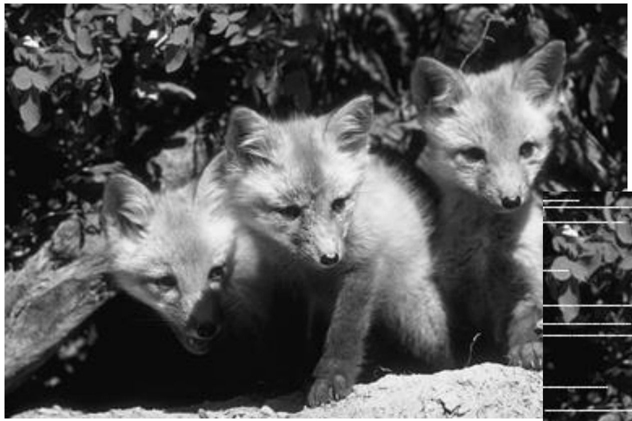
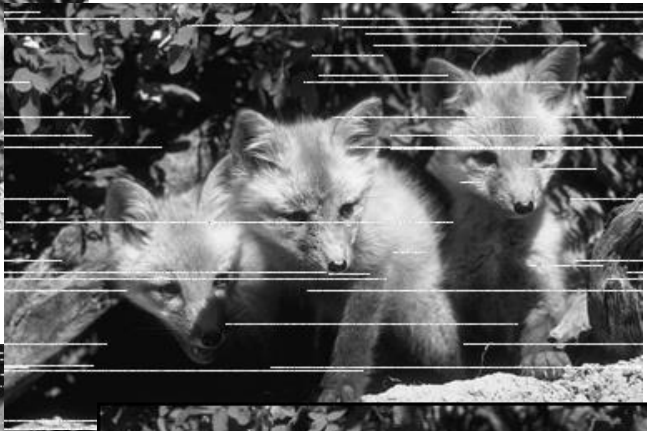
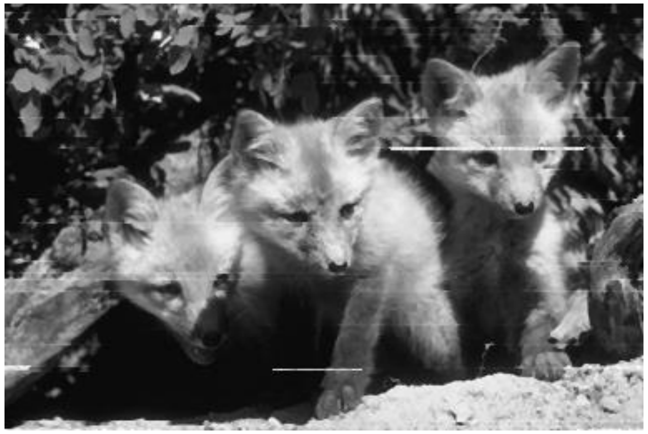
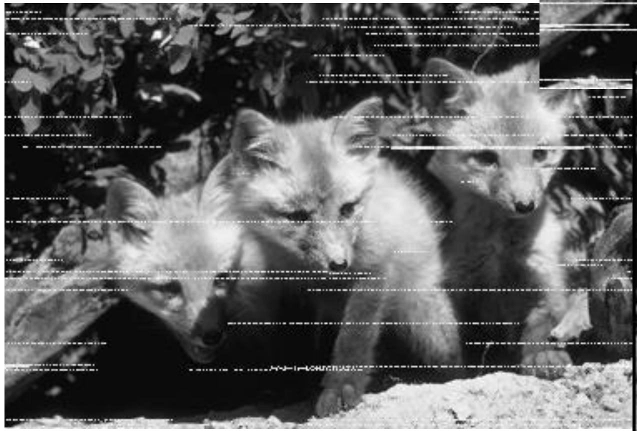


Image corrupted by 5% impulse bursts noise.

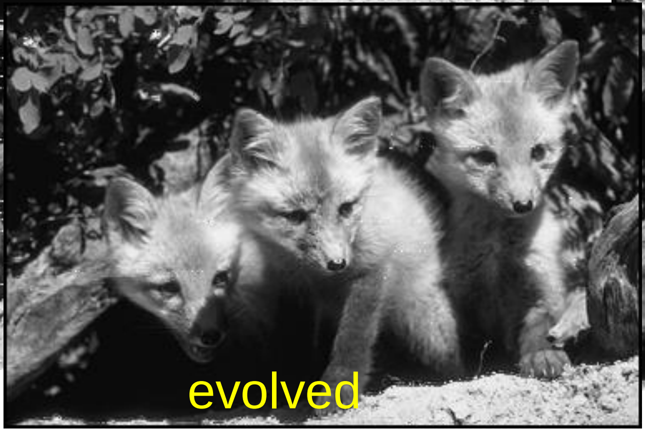




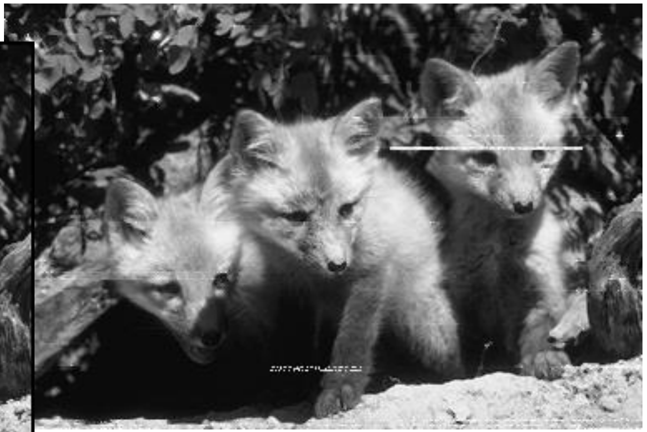
CWMF



AMF



evolved



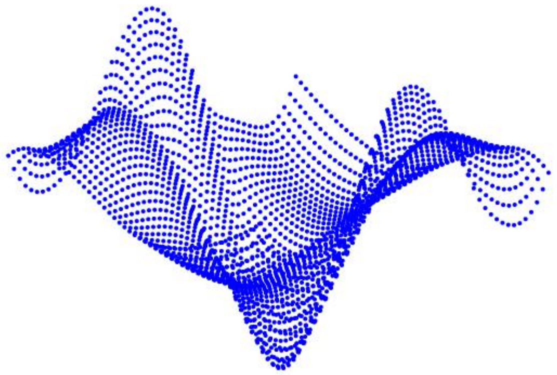
PWMAD

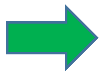
VAŠÍČEK, BIDLO, SEKANINA: Evolution of efficient real-time non-linear image filters for FPGAs. Soft Computing. 17(11), 2013

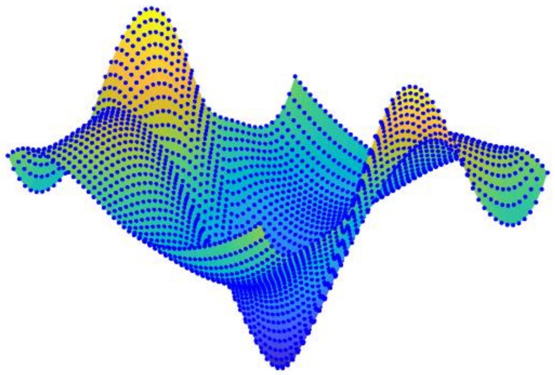
Symbolic Regression for RL

Symbolic Regression

- Fitting models in the form of mathematical expressions to a set of discrete data points







-3.141592654	-30	-23.34719731
-2.932153143	-30	-22.67195916
-2.722713633	-30	-22.07798667
-2.513274123	-30	-21.63117778
-2.303834613	-30	-21.2992009
...	...	...

f = -15.42978401 + 2.42980826 * ((x1 – (x1 * -1.49416733 + x2 * 0.51196778 + 0.00000756)) + (sqrt(power((x1 – (x1 * -1.49416733 + x2 * 0.51196778 + 0.00000756)), 2) + 1) – 1) / 2) ...

Reinforcement Learning Framework

- Dynamic system of interest is described by the state **transition function**

  $x_{k+1} = f(x_k, u_k)$

  with

  state: $x_k, x_{k+1} \in X \subset R^n$

  action: $u_k \in U \quad R^m$

  as a generative model given by a numerical simulation of complex differential equations.

- Control goal is specified through a reward function which assigns a scalar reward $r_{k+1} \in R$ to each state transition from x_k to x_{k+1}:

  $r_{k+1} = \rho(x_k, u_k, x_{k+1})$

  This function typically calculates the reward based on the distance of the current state x_k from a given reference state x_r that should be attained.

- Goal of RL is to find an optimal control policy π which maximizes

$$V^{\pi}(x) = E\left\{\sum_{i=0}^{\infty} \gamma^i \rho(x_i, \pi(x_i), x_{i+1}) \middle| x_0 = x, \pi\right\} \text{, for all } x \in X.$$

Optimal V-function computed by solving the Bellman optimality equation

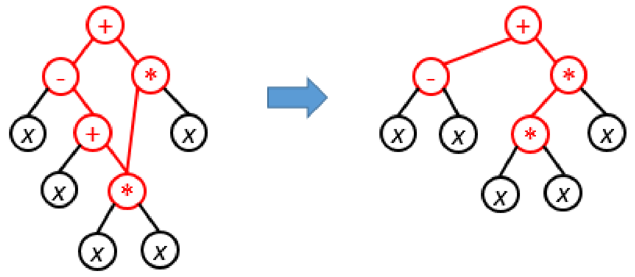
$$\hat{V}^*(x) = \max_{u \in \mathcal{U}} \left[\rho\big(x, \pi(x), f(x,u)\big) + \gamma \hat{V}^*\big(f(x,u)\big)\right]$$

Single Node Genetic Programming (SNGP)

- Graph-based GP technique
- Evolves a linear population of individuals, each representing a single program node
- Program node types
    - **Terminals** – variables, constants
    - **Functions**

	0	1	2	3	4	5	6	7	8	9	10	11
u	*x*	+	-	*****	**+**	/	*****	+	**-**	/	**+**	*
Succ	-	0,0	0,1	**0,0**	**0,3**	1,3	**0,3**	0,4	**0,~~4~~**	5,6	**6,8**	2,8
Pred	-	2,5	11	**4**,5,**6**	7,**8**	9	9,**10**	-	**10**,11	-	**-**	-

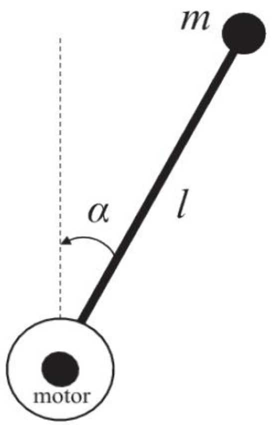
	0	1	2	3	4	5	6	7	8	9	10	11
u	*x*	+	-	*****	+	/	*****	+	**-**	/	**+**	*
Succ	-	0,0	0,1	**0,0**	0,3	1,3	**0,3**	0,4	**0, 0**	5,6	**6,8**	2,8
Pred	-	2,5	11	**4**,5,**6**	7,8	9	9,**10**	-	**10**,11	-	**-**	-

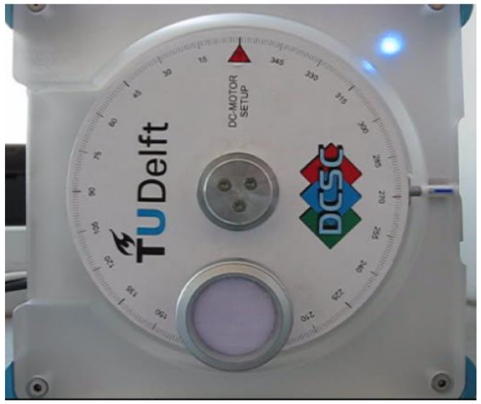


Learning Transition Function

Inverted Pendulum

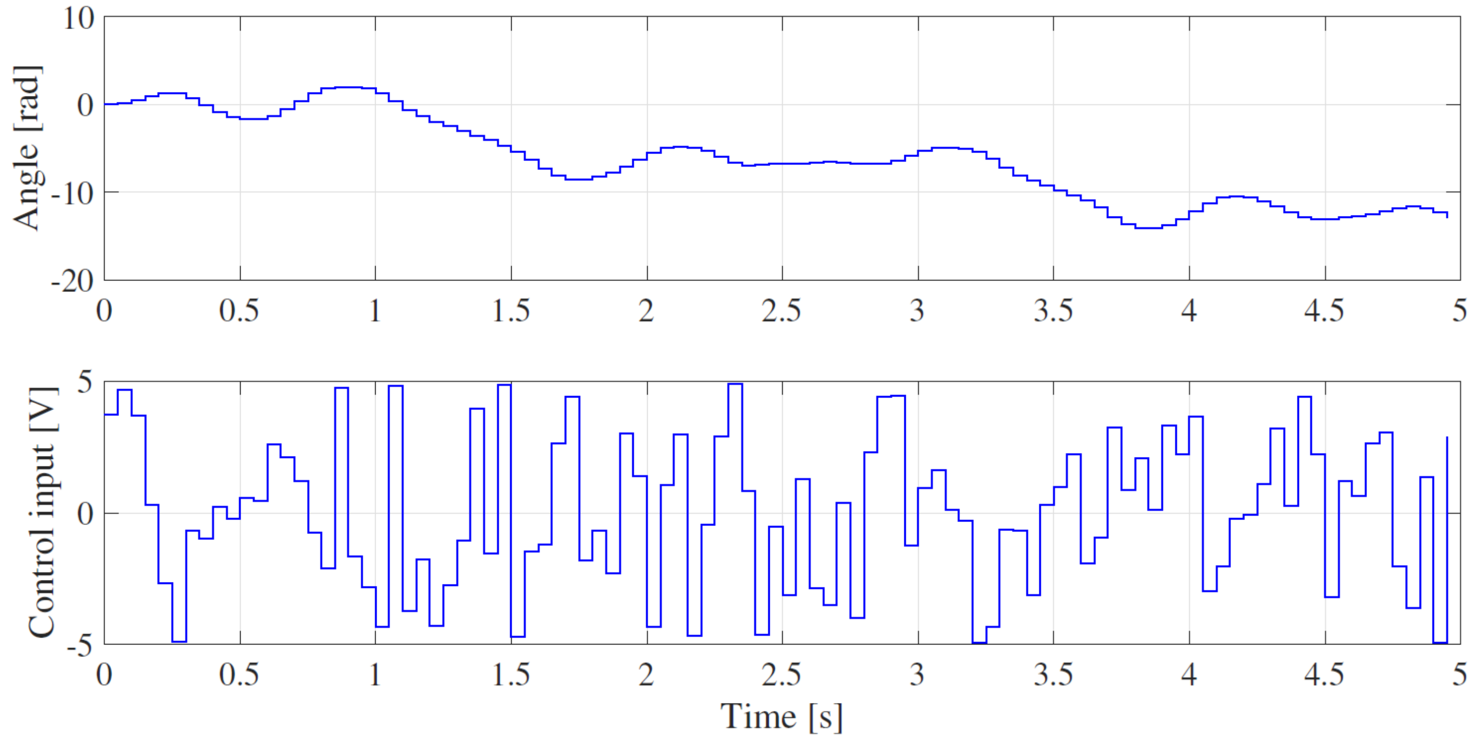
- The pendulum system consists of a weight of mass m attached to an actuated link that rotates in a vertical plane.
- The state vector is $x = [\alpha, \dot{\alpha}]^T$, where α is the angle and $\dot{\alpha}$ is the angular velocity.
- The control input is the voltage u.
- The control goal is to stabilize the pendulum in the unstable equilibrium $x_{des} = [\pi, 0]^T$.





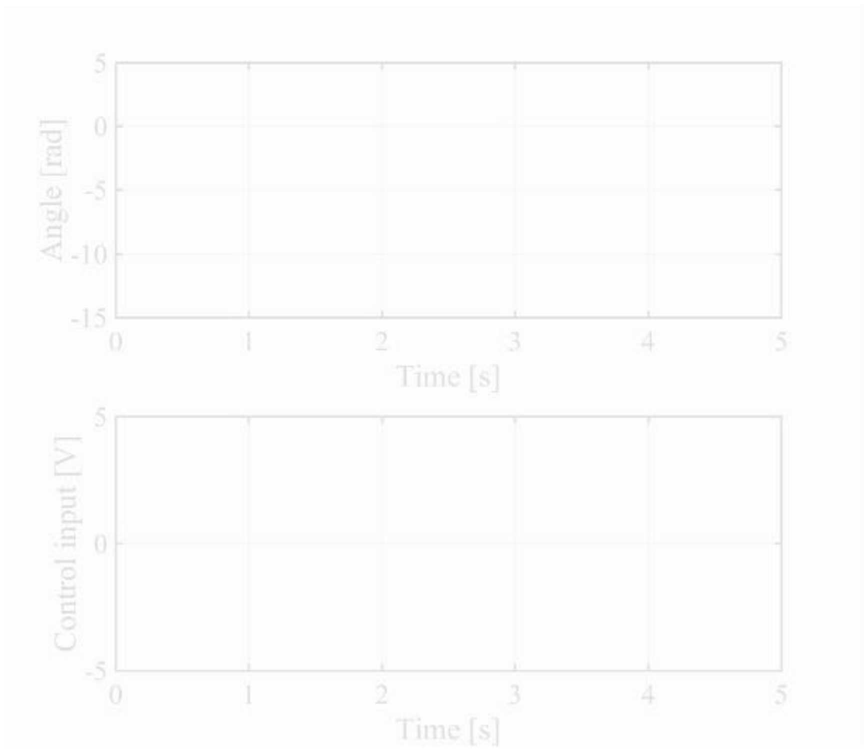
Experiments with Real Inverted Pendulum

Data were collected while applying random input to the system.



Experiments with Real Inverted Pendulum

Collection of training data: random input



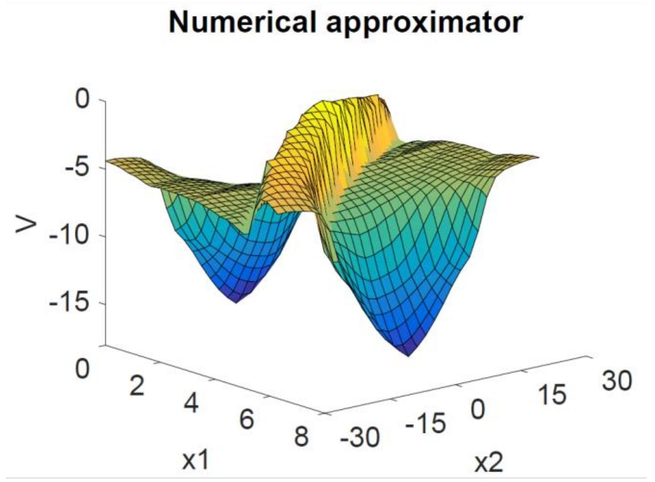
Experiments with Real Inverted Pendulum

Example of the refined model

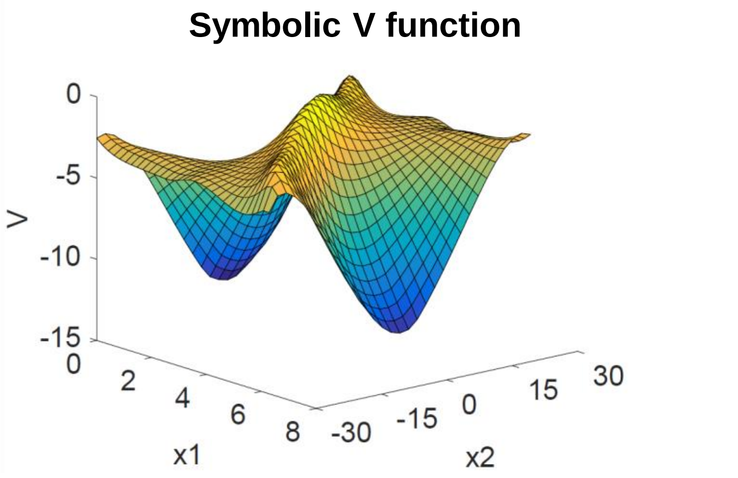
$$
\begin{aligned}
\alpha_{k+1} = {} & 0.99997\,\alpha_k + 0.04872\,\dot{\alpha}_k + 0.03432\,u_k - \\
& - 0.00293\,\mathrm{sign}(\dot{\alpha}_k - 0.50985) + 0.00293\,\cos(\alpha_k\,u_k) + \\
& + 0.00293\,\cos(\alpha_k) + 0.01192\,\mathrm{sign}(\dot{\alpha}_k) - 0.14159\,\sin(\alpha_k) + \\
& + 0.00041\,(\mathrm{sign}(\dot{\alpha}_k) + 2.8284)\,(\mathrm{sign}(\alpha_k\,u_k) + \alpha_k\,\sin(\alpha_k) - 38.199) - \\
& - 0.00008\,\cos(u_k + 25.0)\,(u_k - 1.0\,\cos(\alpha_k) + (\alpha_k - 2.6069)^2 + 25.0) - \\
& - 0.00517\,\sin(2.25\,u_k + 56.25)\,\cos(\dot{\alpha}_k - 0.79209\,\sin(\dot{\alpha}_k)) - \\
& - 0.00241\,\dot{\alpha}_k\,\cos(\alpha_k) + 0.00001\,\cos(2.0\,\alpha_k\,u_k)\,(\alpha_k - 2.6069)^4 + \\
& + 0.04182, \\
\dot{\alpha}_{k+1} = {} & 0.8041\,\dot{\alpha}_k - 0.01419\,\alpha_k + 1.2414\,u_k + 0.10012\,\mathrm{sign}(\cos(\alpha_k)) - \\
& - 0.24194\,\mathrm{sign}(1.0\,\mathrm{sign}(u_k + 1.118) - \dot{\alpha}_k - \cos(\alpha_k) + 1.0\,\mathrm{sign}(\dot{\alpha}_k)) + \\
& + 0.00162\,\sin(6.9282\,\alpha_k + 1.1995\,\sin(u_k)) + 0.10012\,\cos(\dot{\alpha}_k) - \\
& - 0.06341\,\cos(u_k) - 0.1117\,\sin(\sin(\cos(\alpha_k)) + \alpha_k\,\sin(u_k)) + \\
& + 0.06341\,\mathrm{sign}(u_k) - 5.1187\,\sin(\alpha_k) + 0.06341\,\sin(\dot{\alpha}_k) - \\
& - 0.06341\,\cos(\sin(\dot{\alpha}_k))\,(\sin(\alpha_k\,\sin(u_k)) + \mathrm{sign}(\dot{\alpha}_k)) + \\
& + 0.1117\,\dot{\alpha}_k\,(\cos(\sin(\cos(\alpha_k))) - 1.0\,\sin(\sin(\cos(\alpha_k)))) + \\
& + 0.08627\,(\cos(\alpha_k) - 1.0)\,(\sin(u_k) - 0.9093) - \\
& - 0.22817\,\sin(6.9282\,\alpha_k + 1.1995\,\sin(u_k))\,(\cos(u_k)^2 + \cos(\alpha_k) - \\
& - 1.0) + 0.00163\,\dot{\alpha}_k\,\sin(\alpha_k)\,(\dot{\alpha}_k - 1.0) + 0.01791
\end{aligned}
$$

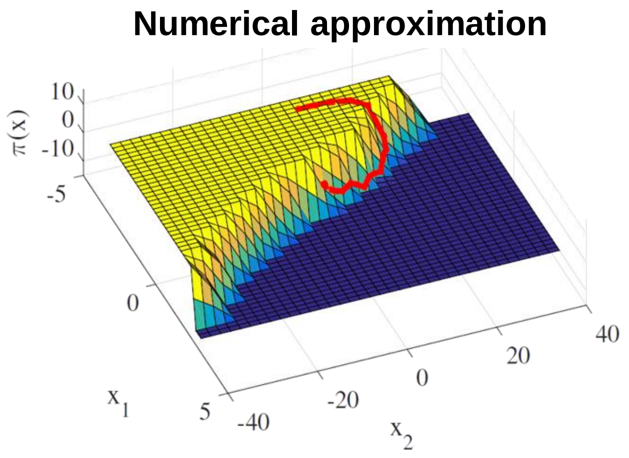
Fitting V and Policy function

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

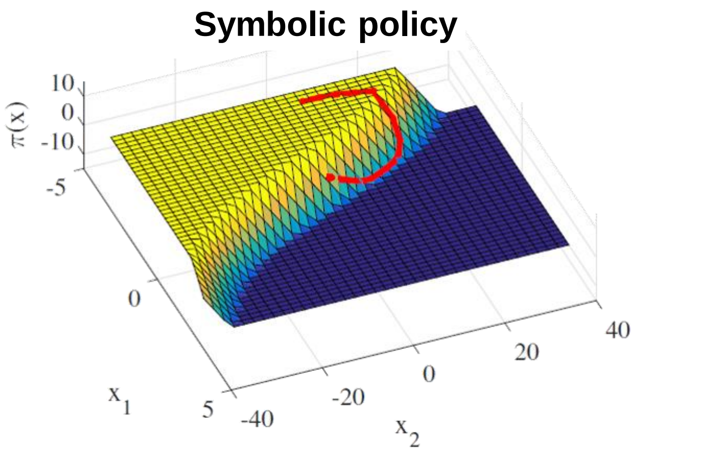


vs.

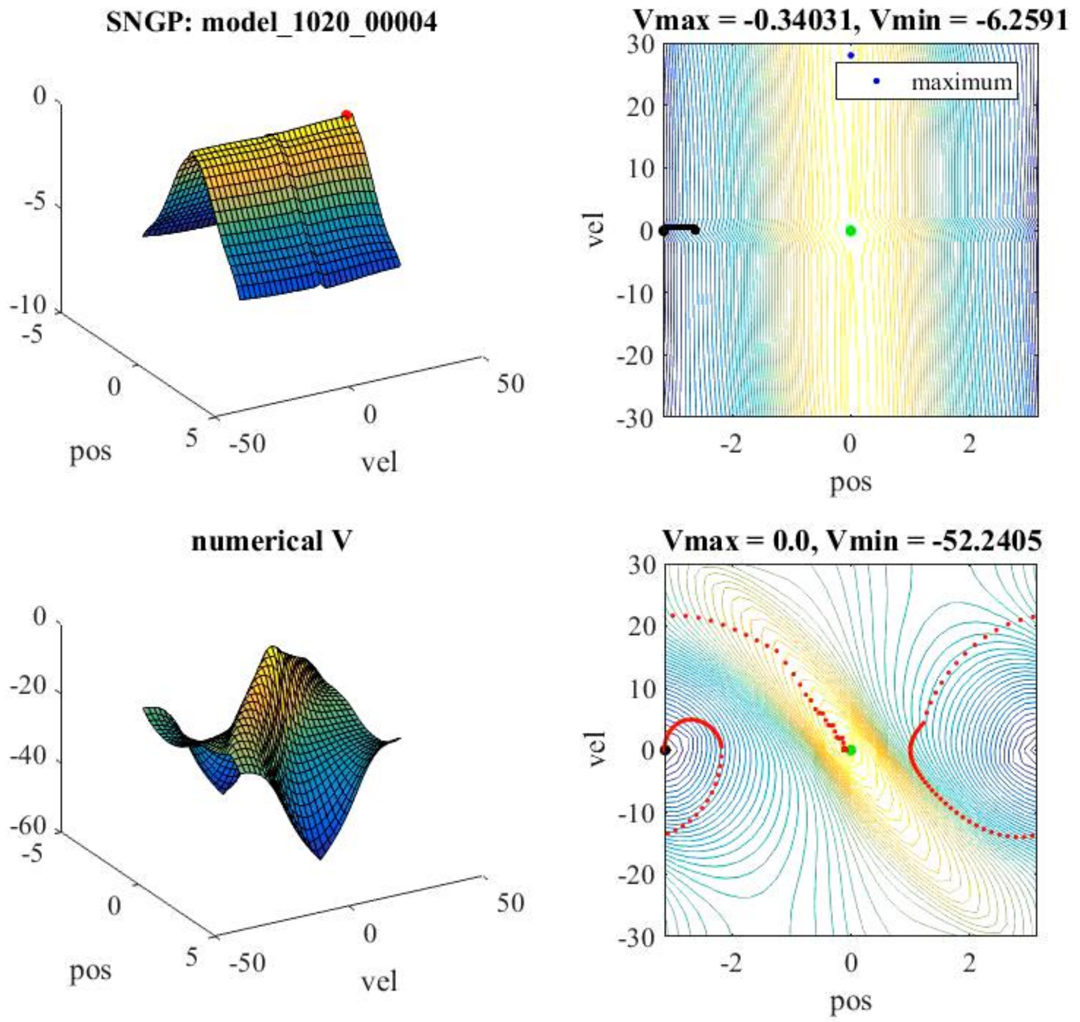




vs.



Learning V function



EAs for Agents Clustering

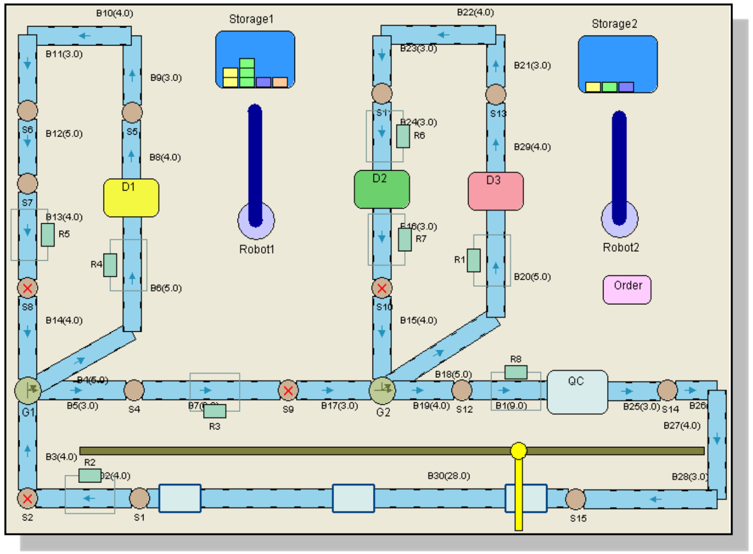
Problem: Static agents clustering within multi-agent system from messages sent and received between every pair of agents.

Goal: To find a partitioning of agents into *N* clusters such that

- **communication within clusters is maximized** and **inter-cluster communication is minimized** and
- **the clusters are of similar size** in order to efficiently distribute agents across multiple execution units.

Example: MAS, where agents are used for control of the transportation of products or discrete materials on the factory's shop floor using a network of conveyor belts.

- partitioning of **67 agents into 4 clusters**

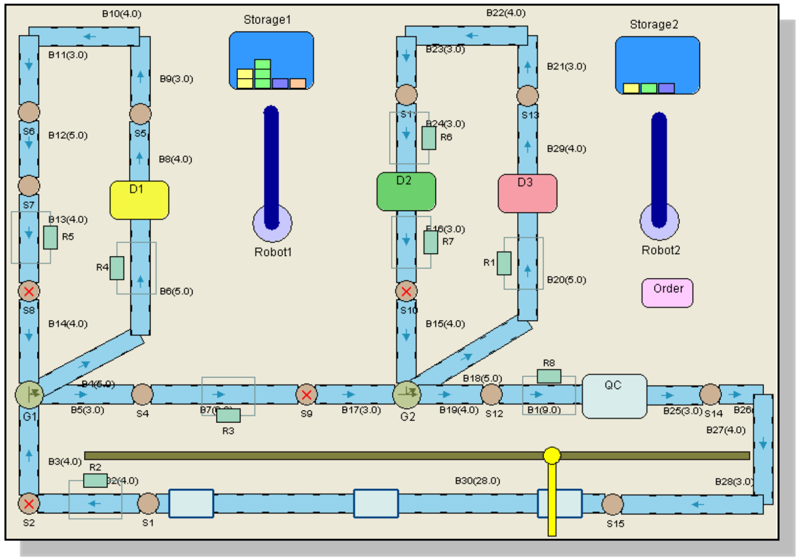


EAs for Agents Clustering

- Example: MAS, where agents are used for control of the transportation of products or discrete materials on the factory's shop floor using a network of conveyor belts.

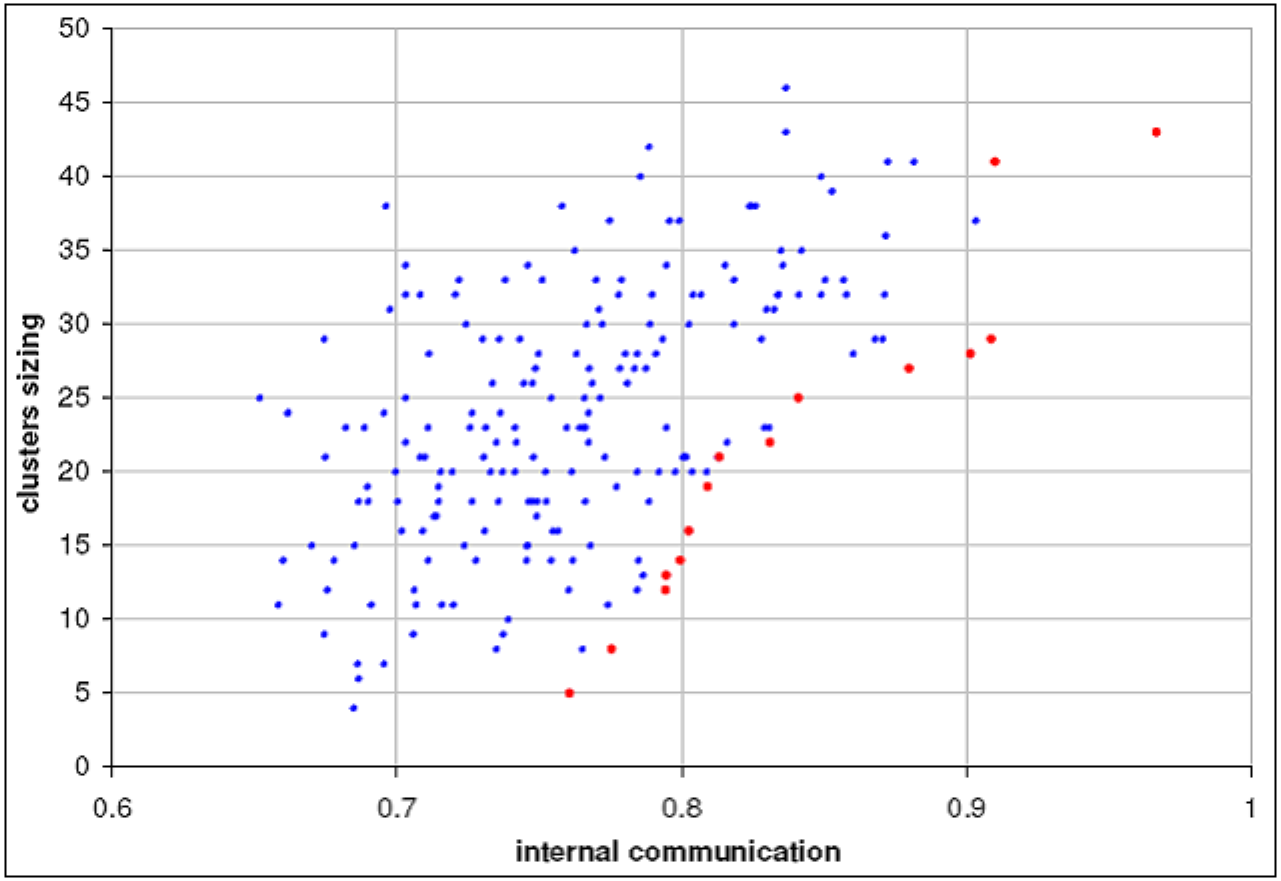
  Goal is to find an optimal partitioning of **67 agents into 4 clusters**.

MAS: The material-handling application

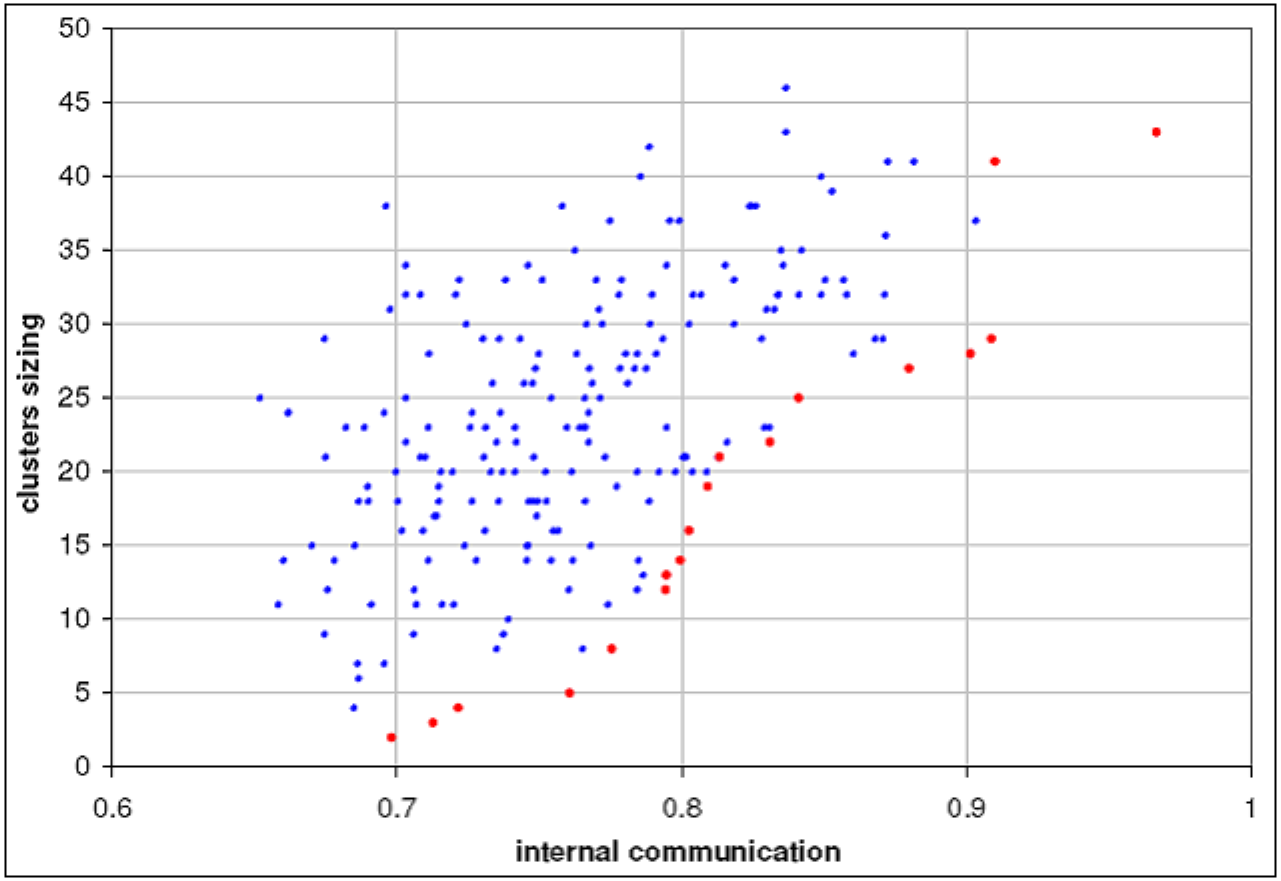


Copyright © 2007 Rockwell Automation, Inc. All rights reserved

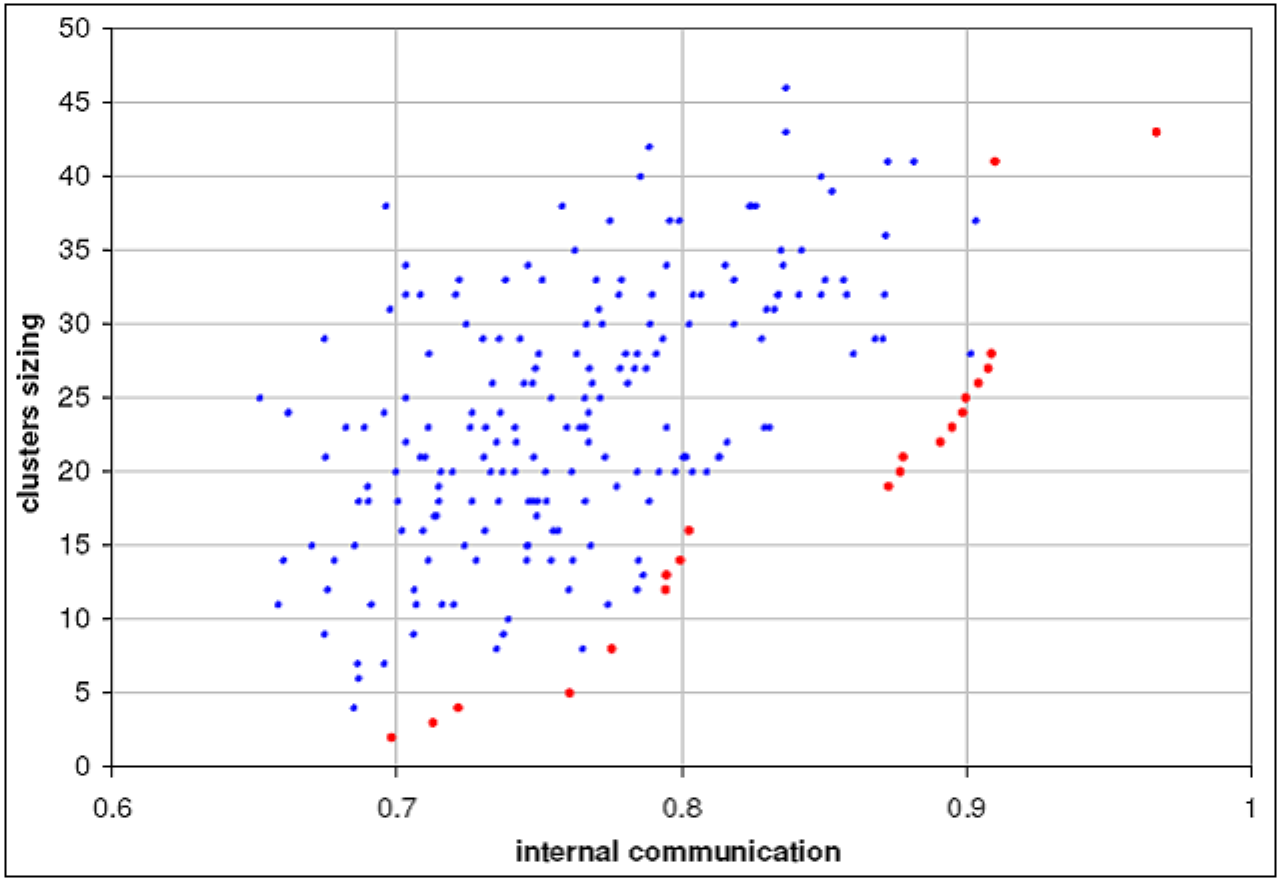
mPOEMS: Initialization of SB



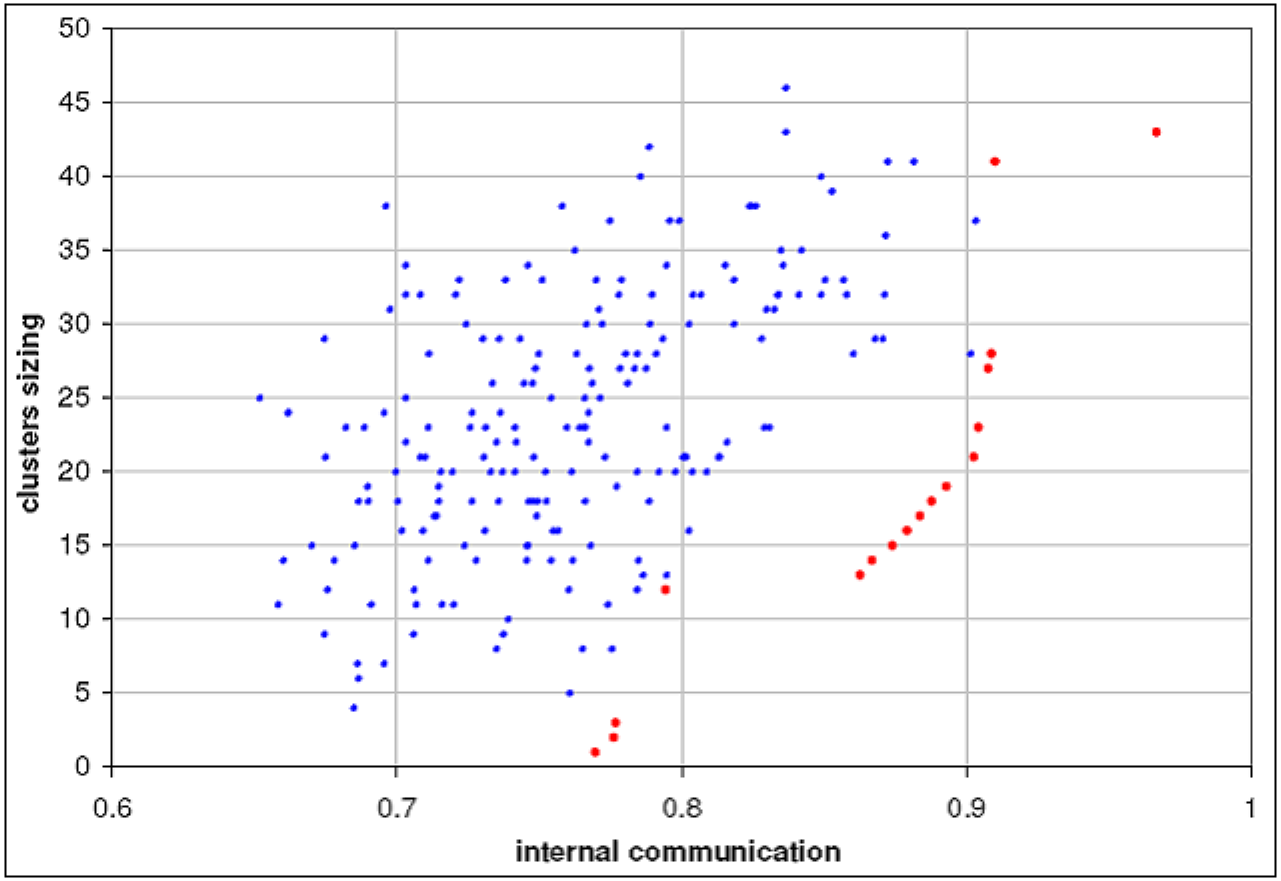
mPOEMS: After 1st iteration



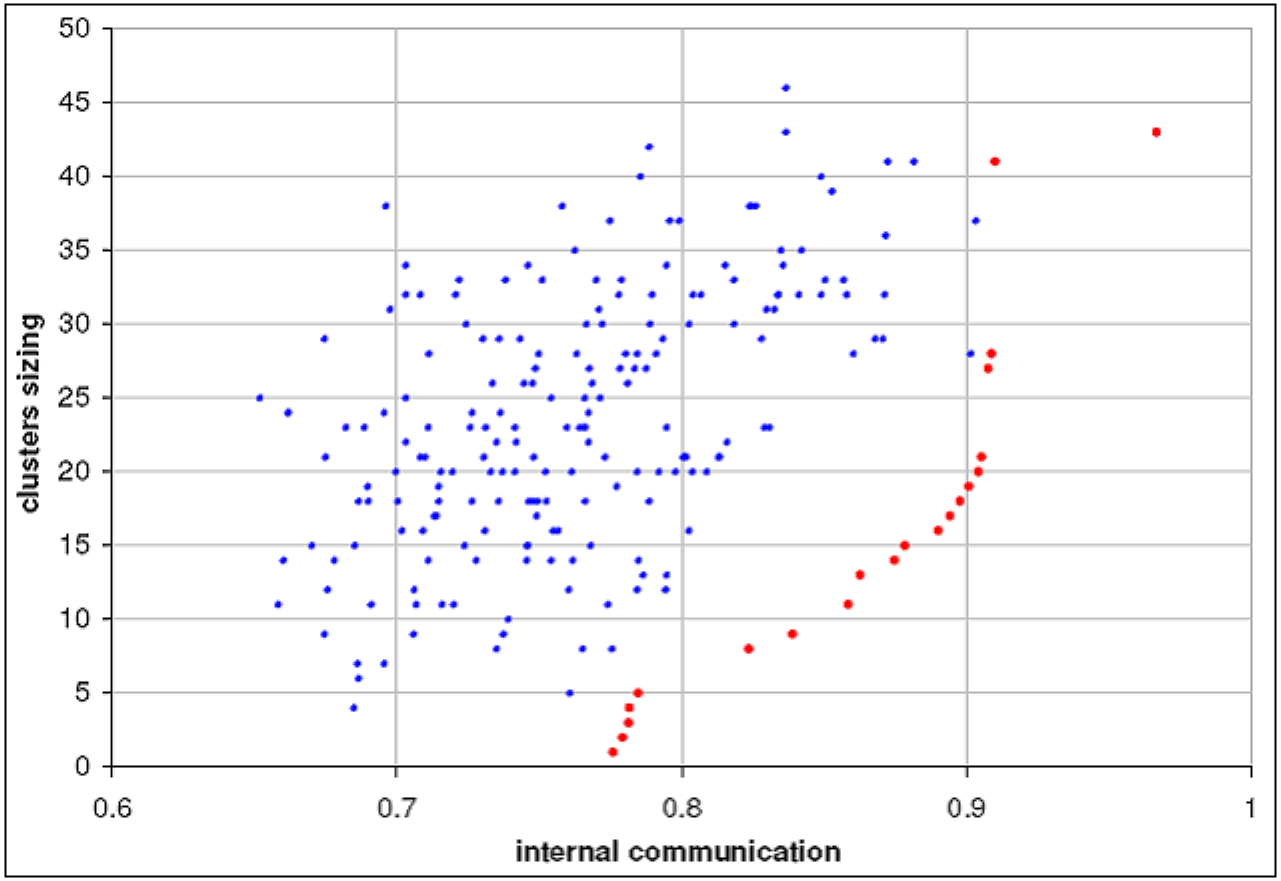
mPOEMS: After 2nd iteration



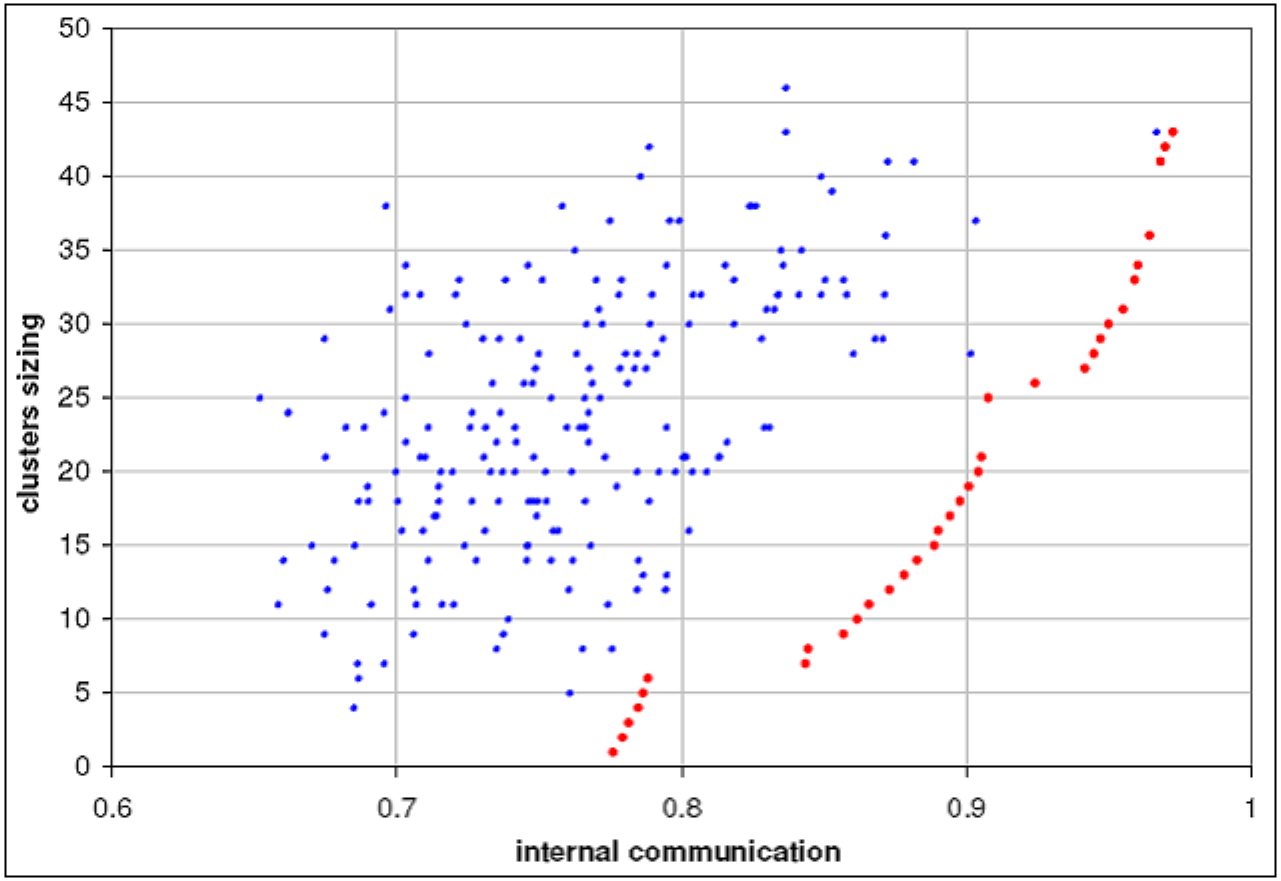
mPOEMS: After 5th iteration



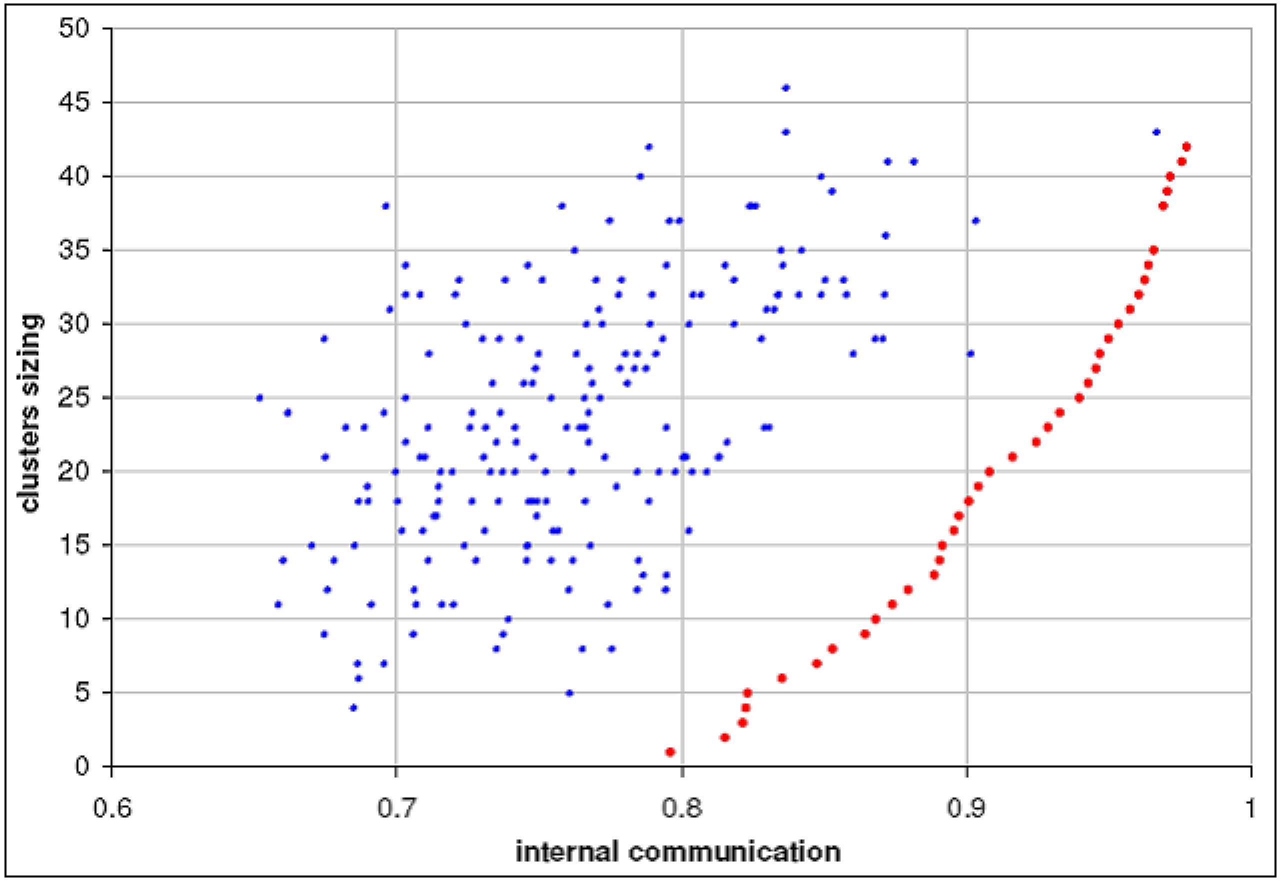
mPOEMS: After 10th iteration



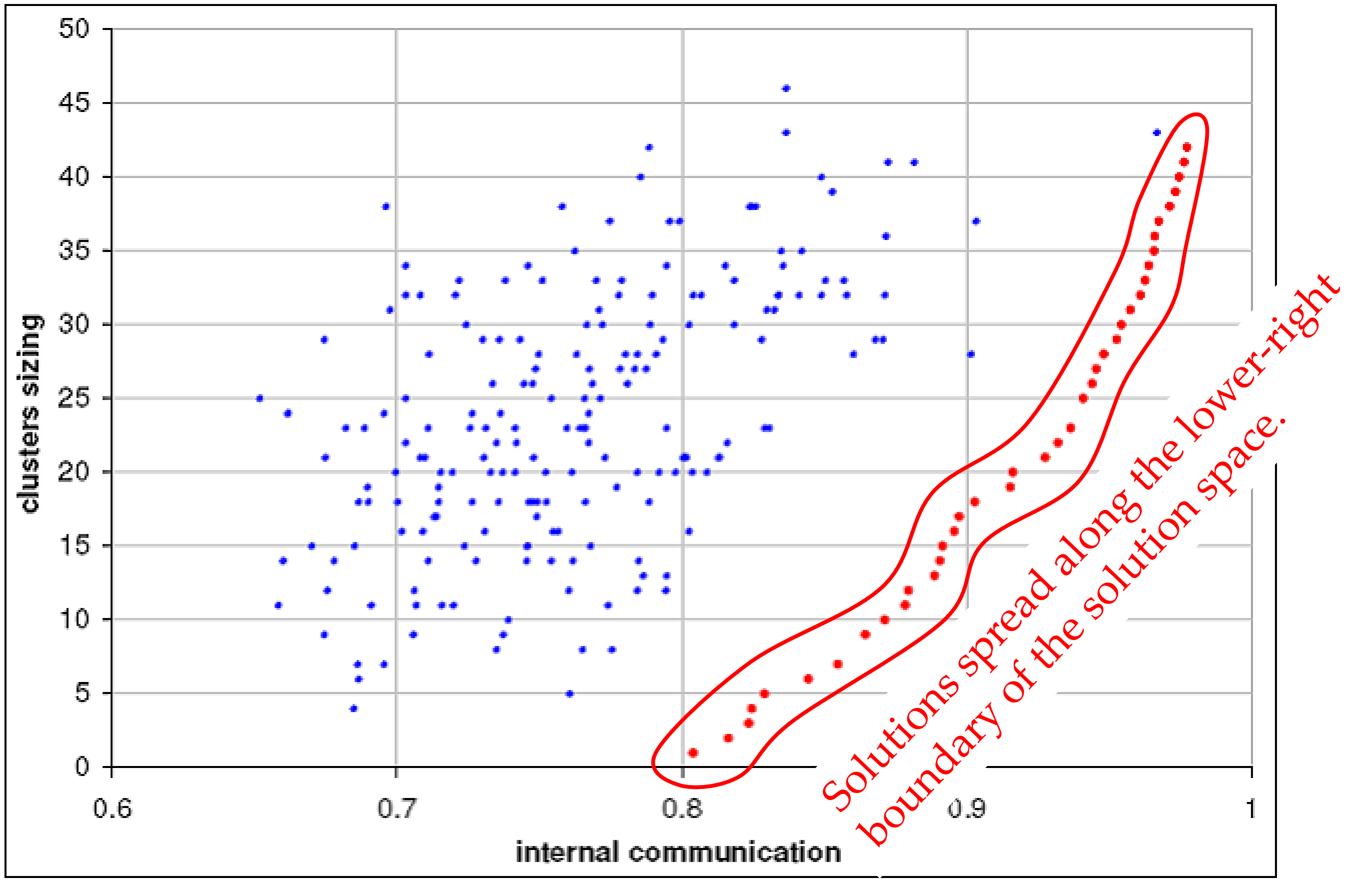
mPOEMS: After 20th iteration



mPOEMS: Final SB after 100 iterations



mPOEMS: Final SB



Facility Layout Design Support Tool

PROBLEM

Facility Layout Optimization

Goal – find a layout optimal w.r.t. the given objective function

- minimize a size of the hall partition occupied by production lines
- minimize overall length of communications among connected production lines

Floorplanning – generally an NP-complete problem

- very hard solve to optimality
- computational complexity grows exponentially with the size of the problem

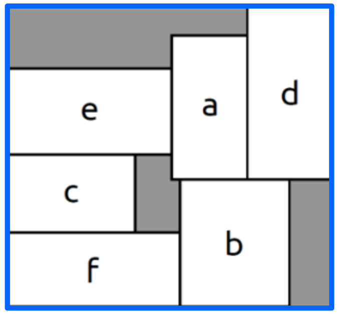
(meta)heurististics needed

- Local search
- Evolutionary algorithms
- Hybrid approaches = EA + LS
- NSGA-II

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).

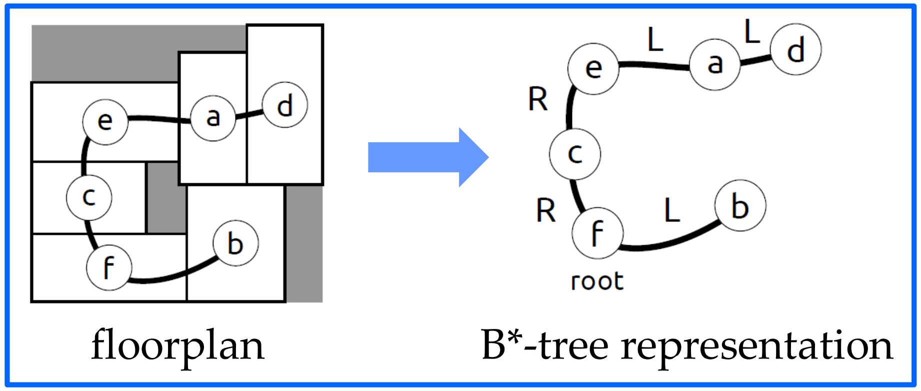
    Ex.: Blocks {a, b, c, d, e, f} should be placed in a rectangular area so that the bounding rectangular area is minimal (or the dead space, shown in gray, is minimal).



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

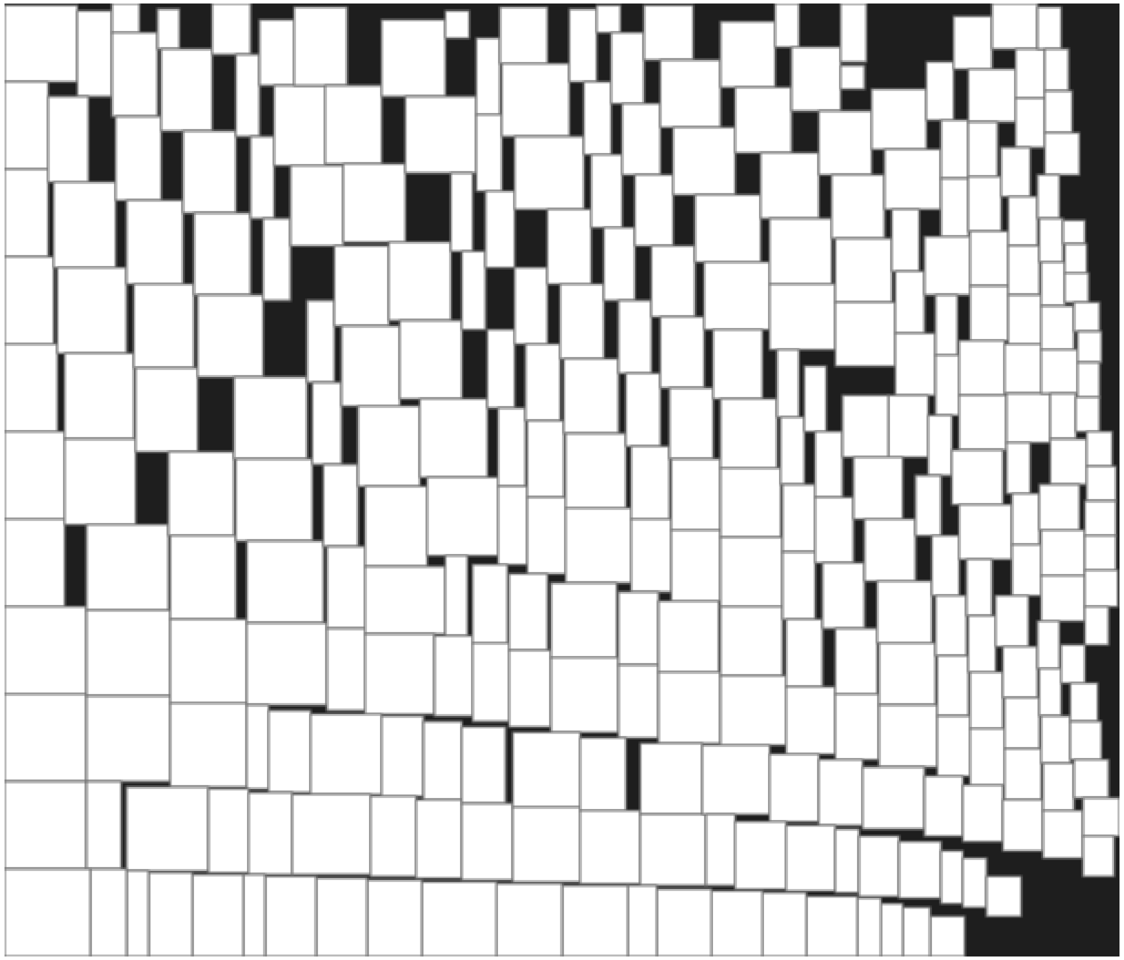
  Each **tree is expressed by a linear string** of symbols in a prefix not.

  Optimization works on linear structs.



Floorplanning

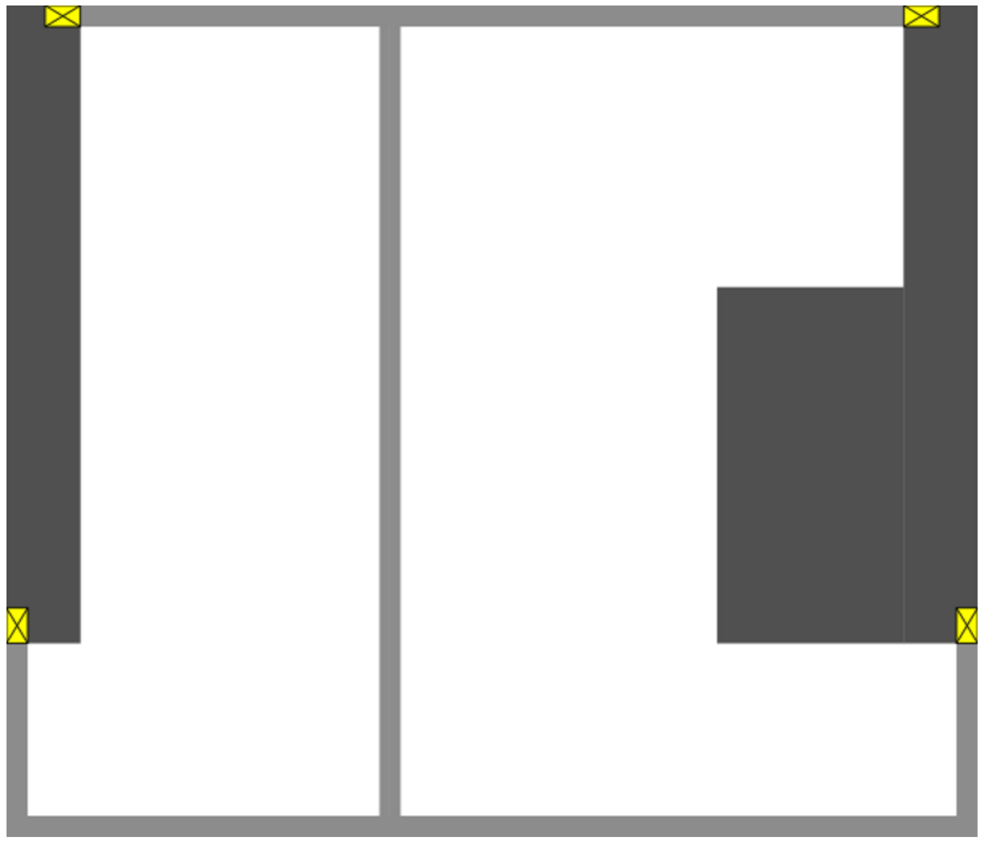
Visualization of a POEMS run on data with 300 blocks.



By V. Hordějčuk.

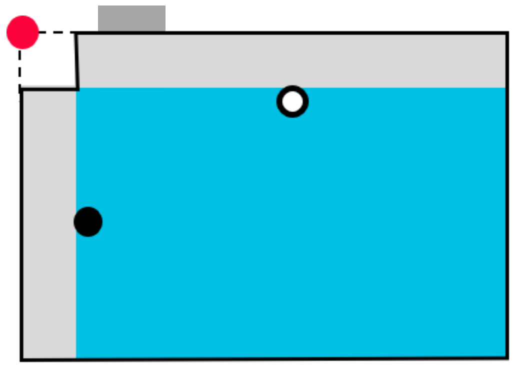
VIDEO

Input: Hall



■	Unavailable space
▨	Handling space
☒	Hall gate

Input: Workstations



🟦	Working area
⬜	Handling area
○	Input point
●	Output point
🔴	Pivot

Mobility

- **Free** – position and rotation randomly initialized, both the position and rotation can change during the optimization process
- **Limited** – optimization starts from recommended position, rotation allowed
- **Pinned** – fixed position, rotation allowed
- **Fixed** – both the position and rotation are fixed for the whole optimization process

Indirect Representation

Indirect representation – a priority list as a sequence of triples <id, r, h>

- id – workstation identifier
- r – workstation rotation
- h – constructive heuristic

expresses the order in which workstations will be inserted into the developed layout and the heuristic used to process each workstation.

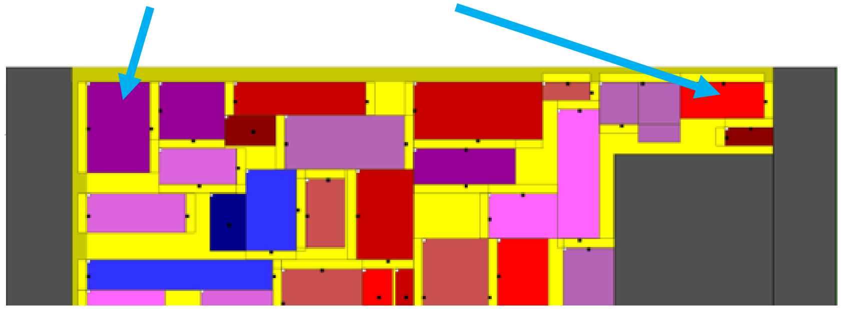
Rotation

- 0°, ±90°, 180°, horizontal/vertical mirroring

Constructive heuristics

- h1, h2

Example: [<ws6, 90°, h1>, <ws6, 0°, h2>, <ws6, -90°, h1>, … , <ws3, 180°, h2>]



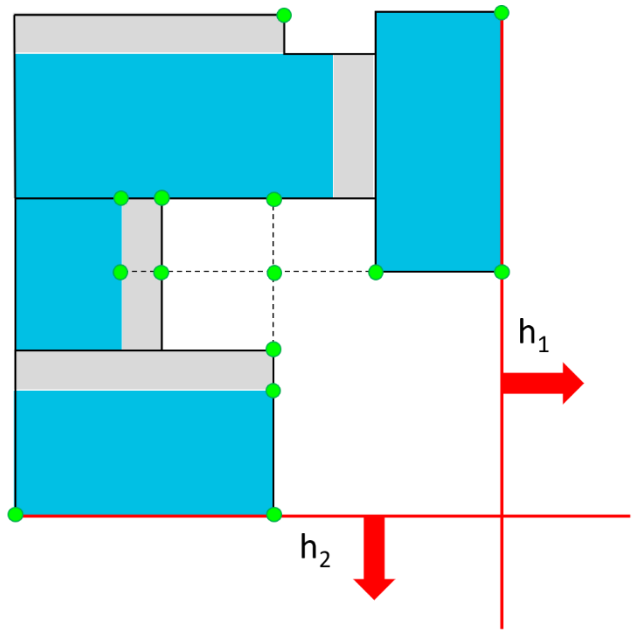
Placement Heuristics

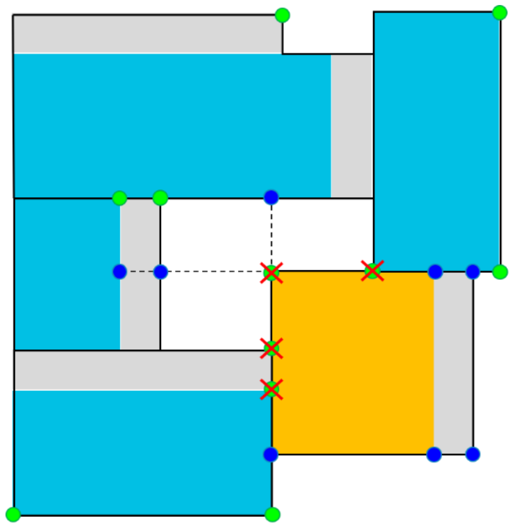
heuristic 1

1. Inside the bounding box.
2. To the right of the bounding box while minimizing the width of the bounding box.

heuristic 2

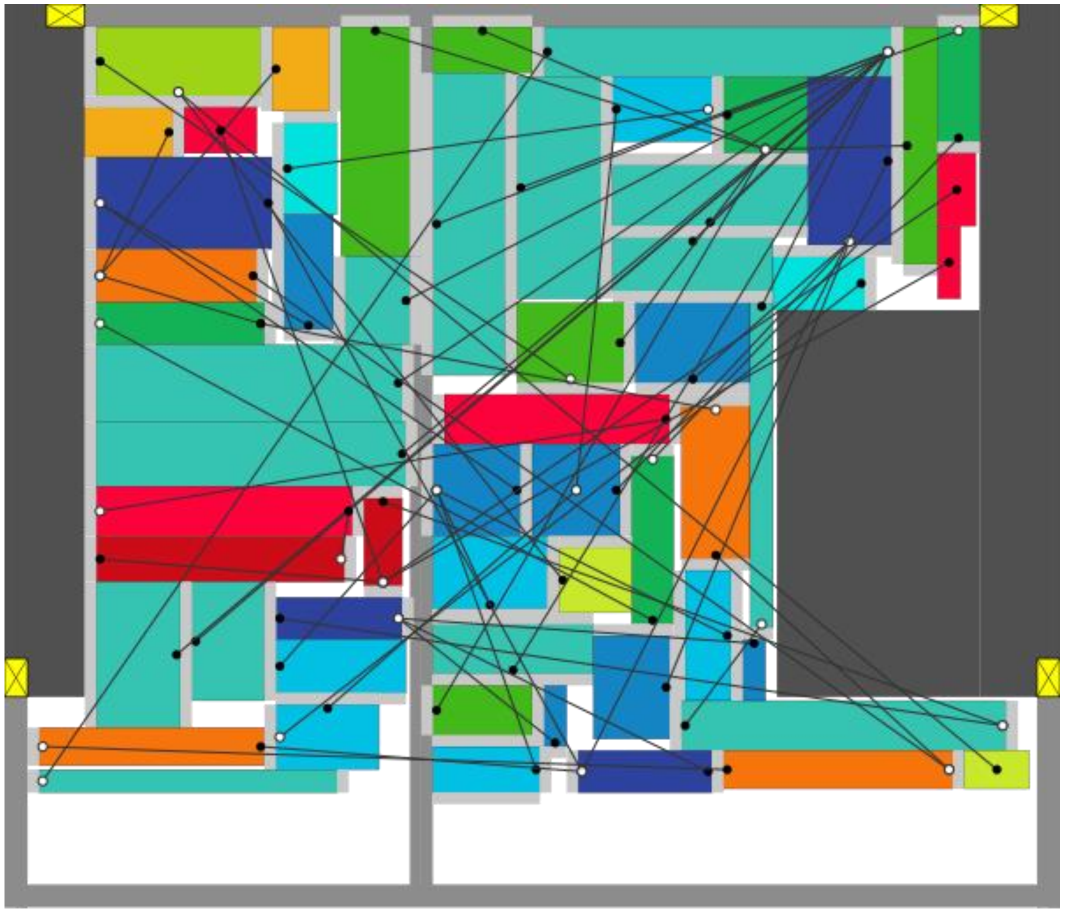
1. Inside the bounding box.
2. Below the bounding box while minimizing the height of the bounding box.





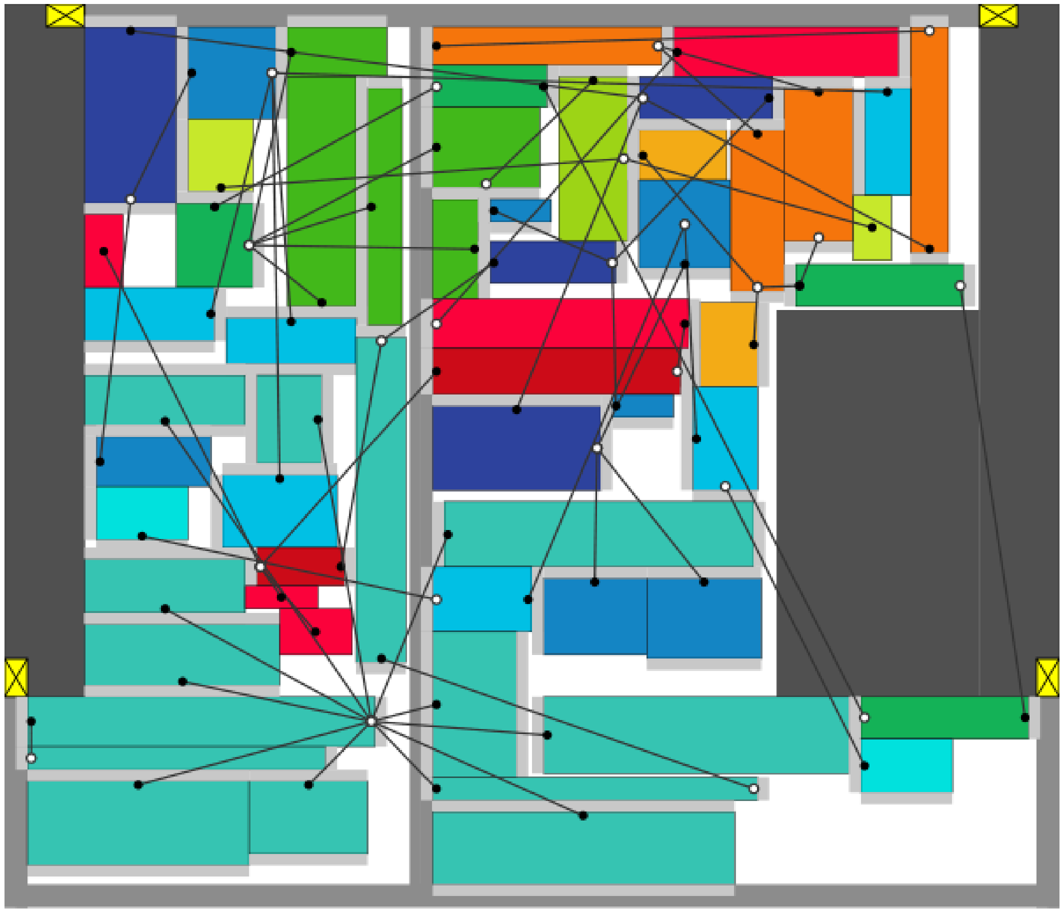
Objective: Minimize Space Used

Minimize a portion of the hall occupied by production lines (maximize available space).



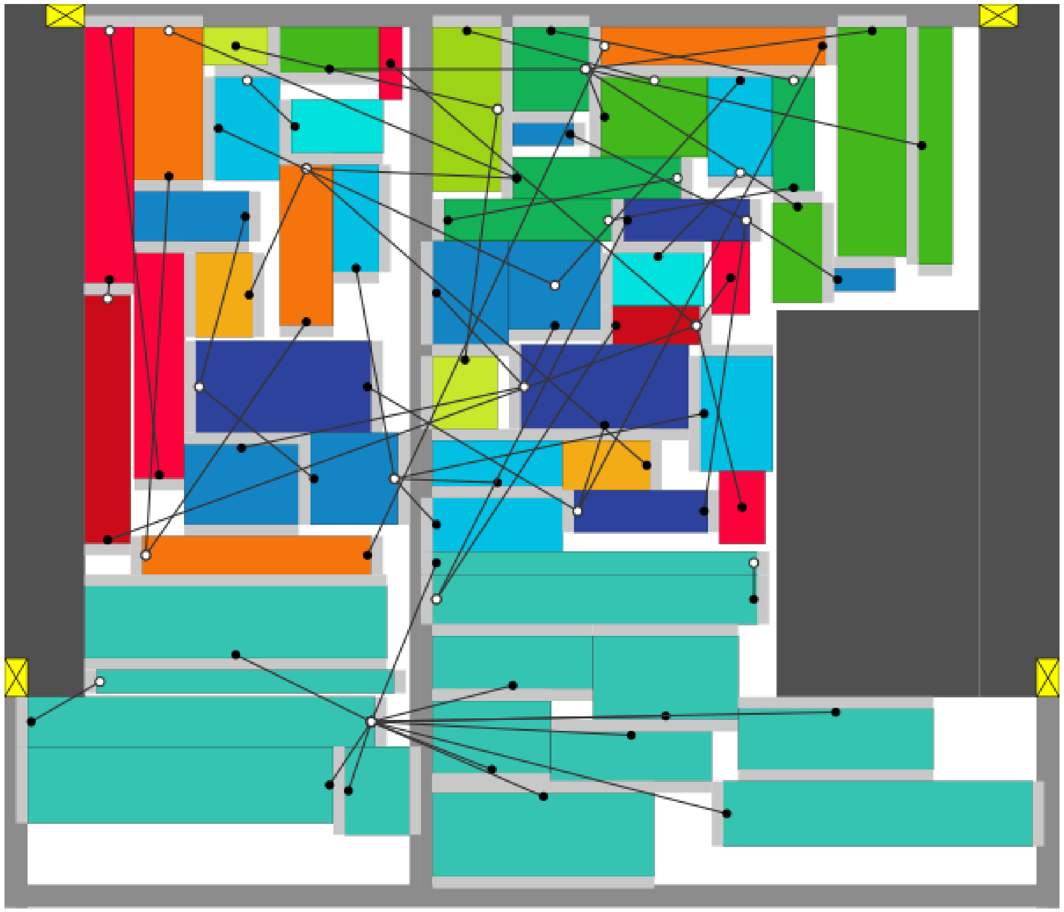
Objective: Minimize Connections

Minimize sum of the communication links among connected production lines (Euclidean distance, Hamming distance, …).



Multi-Objective Optimization

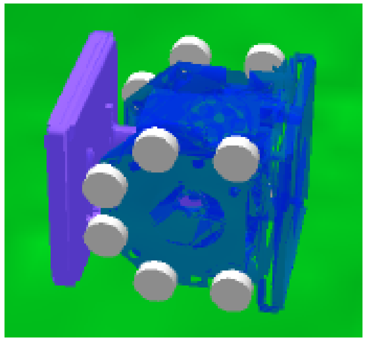
Optimize both criteria simultaneously.

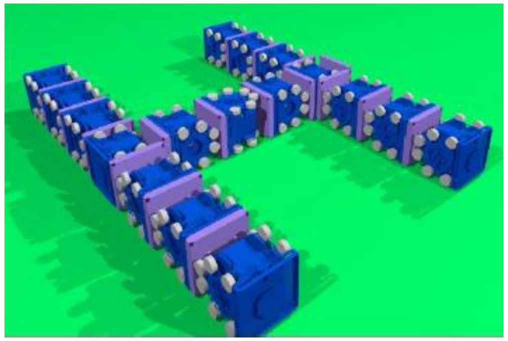


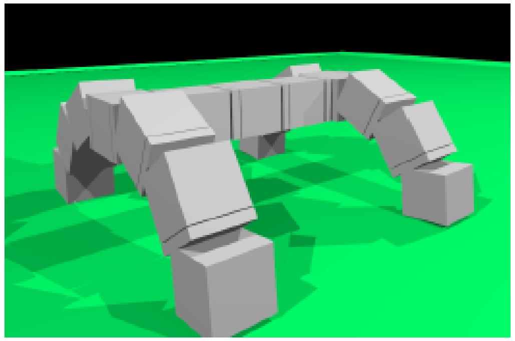
Evolution of Modular Robot Gaits

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

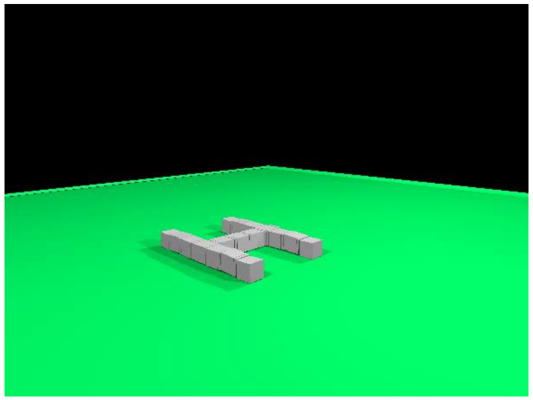
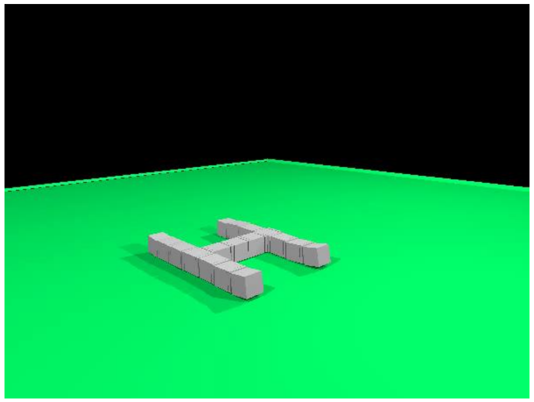
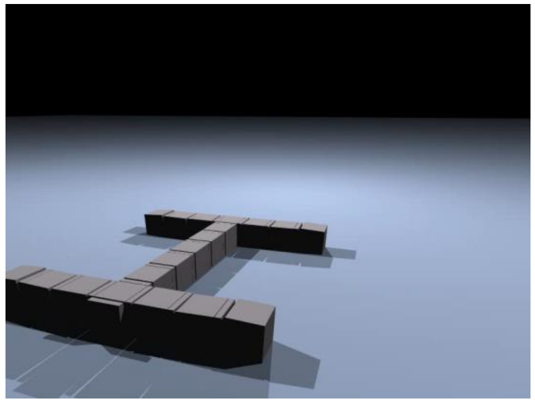
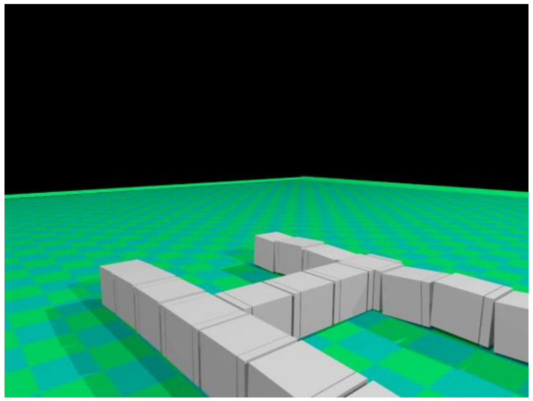
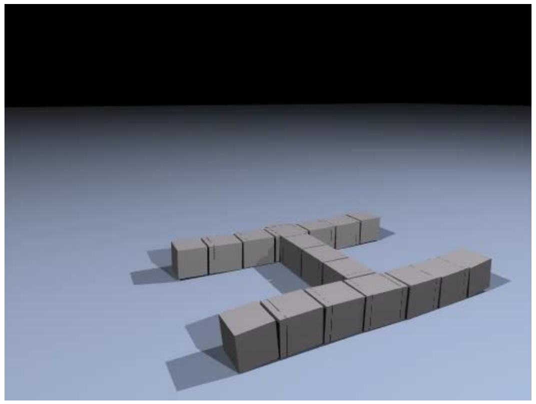
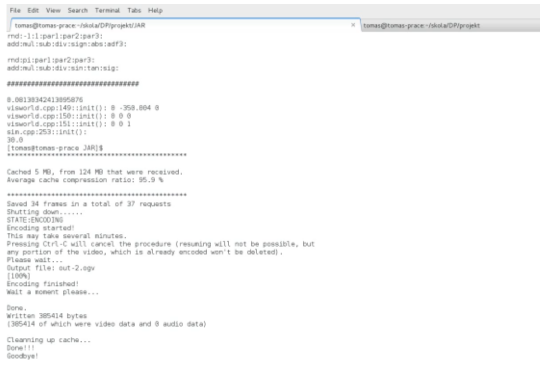

module


modular robot


simulation

- **Approaches:**
  - Co-evolution of a single leg motion pattern and a coordination strategy
  - HyperGP – HyperNEAT with CPPN replaced with GP
  - GP with automatically defined functions

Evolution of Modular Robot Gaits







File Edit View Search Terminal Tabs Help
tomas@tomas-prace:~/skola/DP/projekt/JAR
tomas@tomas-prace:~/skola/DP/projekt

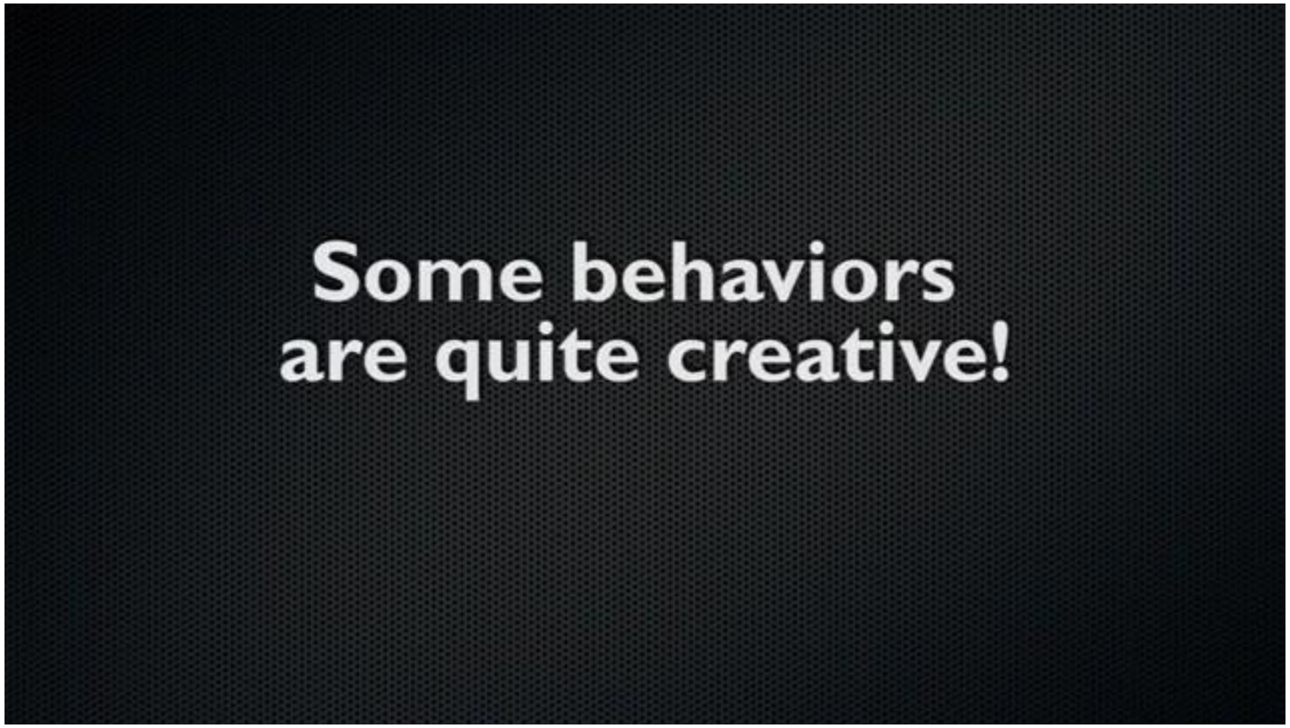
```
rnd:-1:1:par1:par2:par3:
add:mul:sub:div:sign:abs:adf3:

rnd:pi:par1:par2:par3:
add:mul:sub:div:sin:tan:sig:

#################################

0.0813834241389587б
visworld.cpp:149::init(): 0 -350.804 0
visworld.cpp:150::init(): 0 0 0
visworld.cpp:151::init(): 0 0 1
sim.cpp:253::init():
30.0
[tomas@tomas-prace JAR]$
********************************************

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 385414 bytes
(385414 of which were video data and 0 audio data)

Cleanning up cache...
Done!!!
Goodbye!
```

Surprising Creativity of Digital Evolution

Surprising Creativity of Digital Evolution

Elbow Walking, Cully et al. (2015)

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

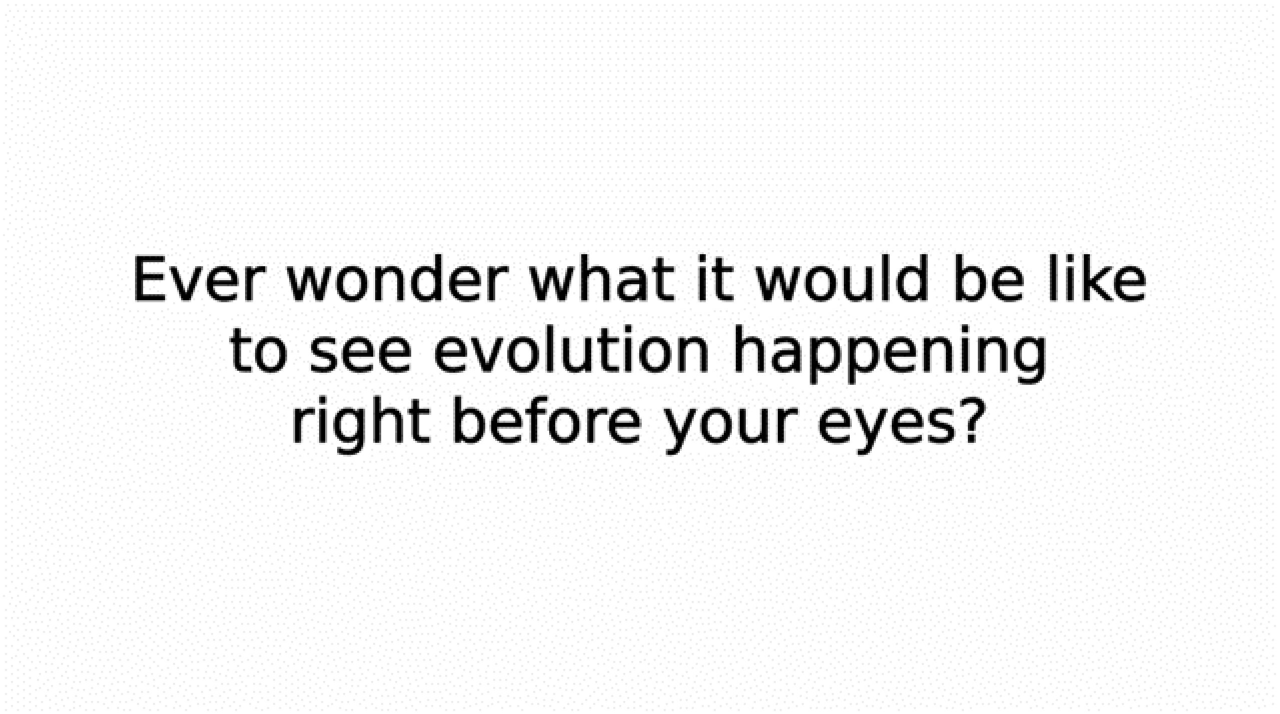

Some behaviors are quite creative!

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

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 Appendenges, 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