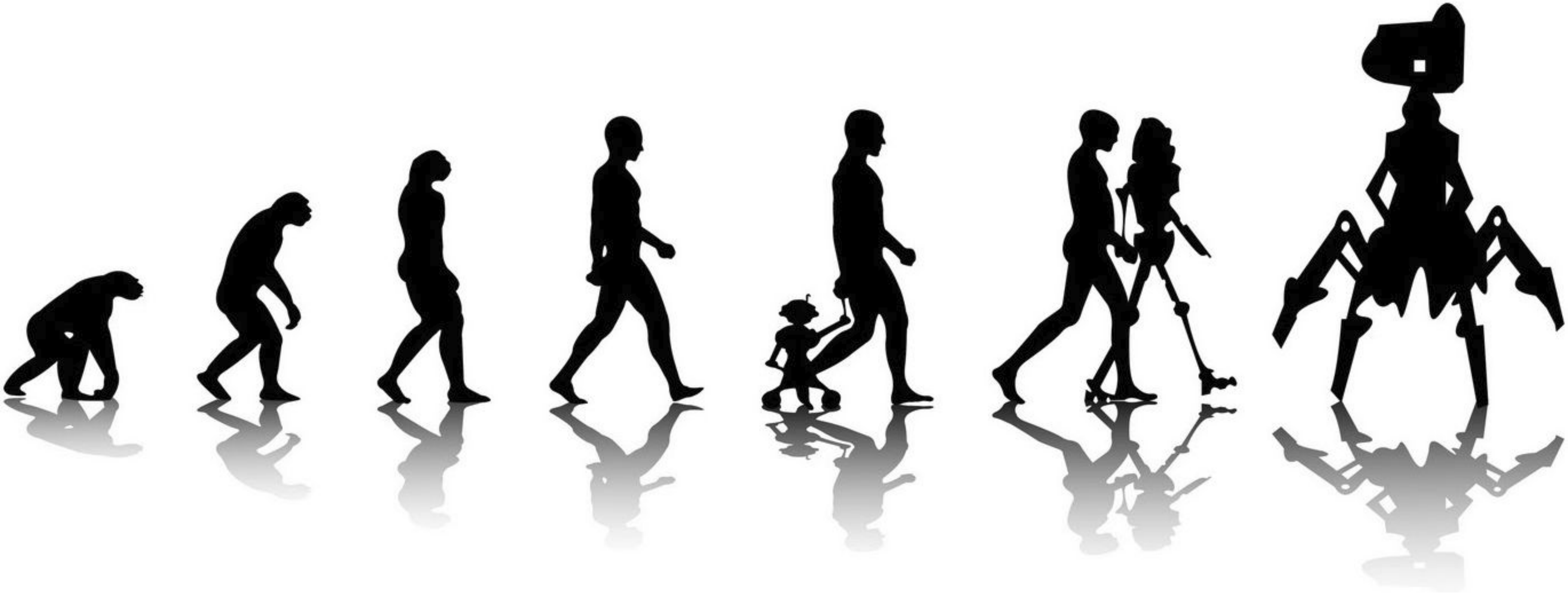


Modern Robotics: Evolutionary Robotics
COSC 4560 / COSC 5560

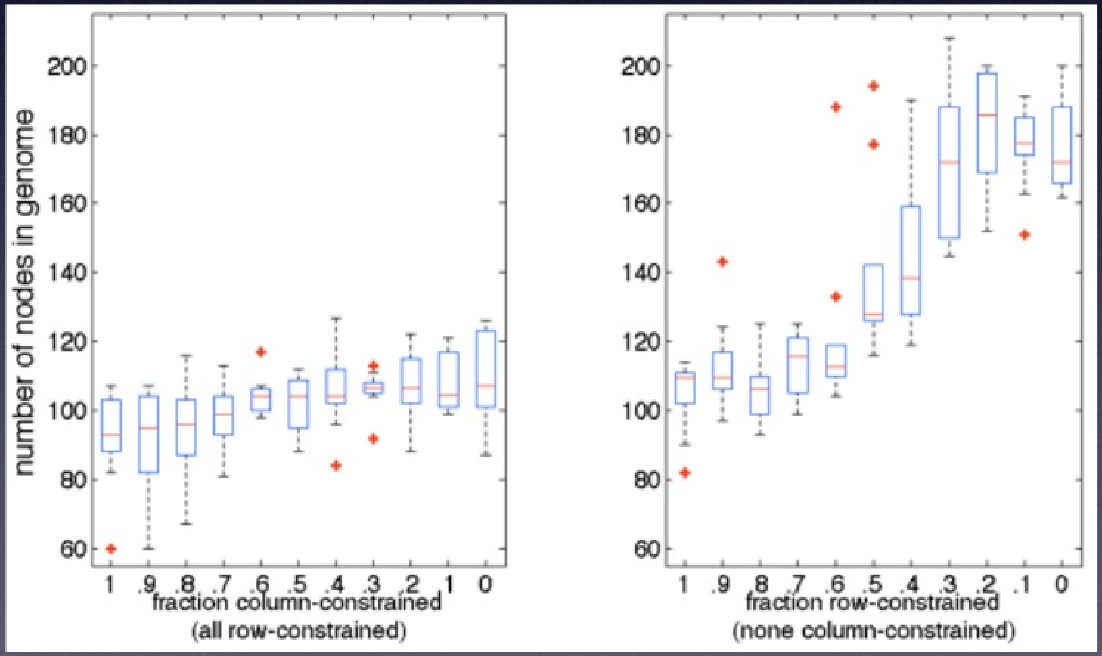
Professor Cheney
2/23/18

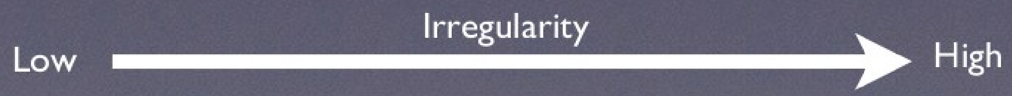
HyperCube NeuroEvolution of Augmenting Topologies (Hyper-NEAT)

HyperNEAT ANNs are More Regular on More Regular Problems

Genome node number indicates compressibility

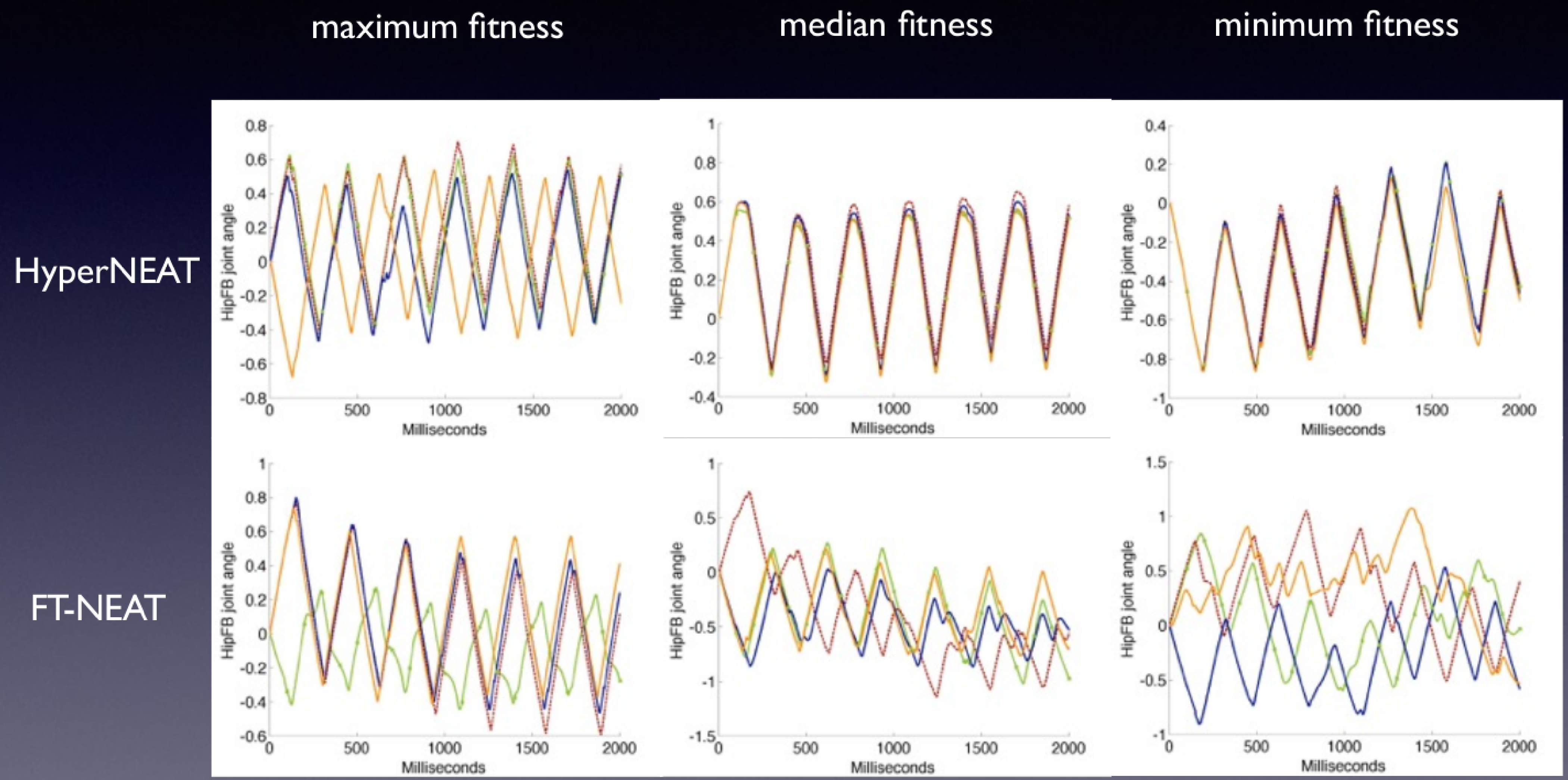
- Target Weights: *r = 0.54 (p = .08)*
- Quadruped: *r = 0.58 (p > .1)*
- Bit Mirroring: *r = 0.91 (p < .001)*





Clune et al. IEEE TEC. 2011

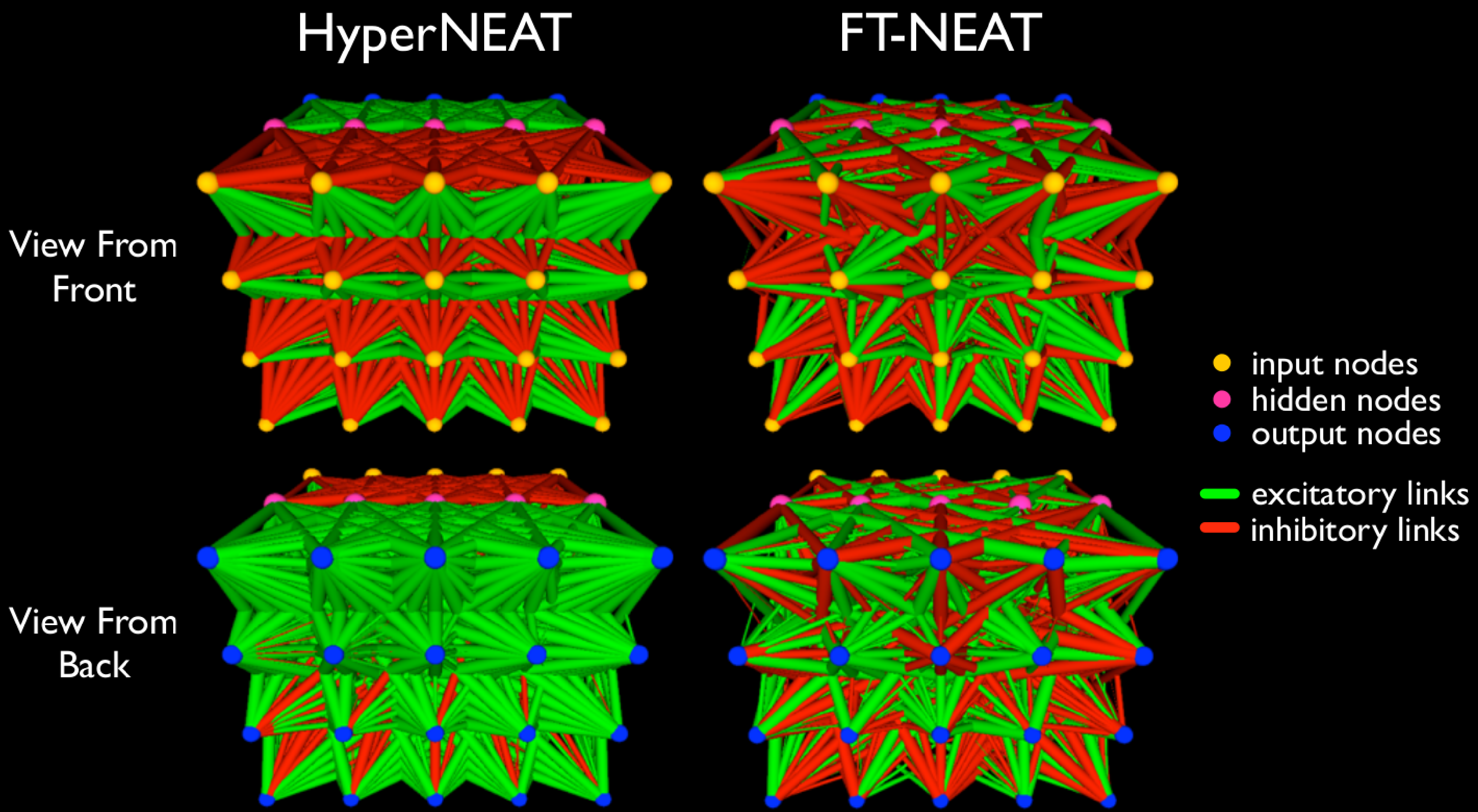
HyperNEAT Controllers are More Coordinated



Clune et al. CEC. 2009

Clune et al. IEEE TEC. 2011

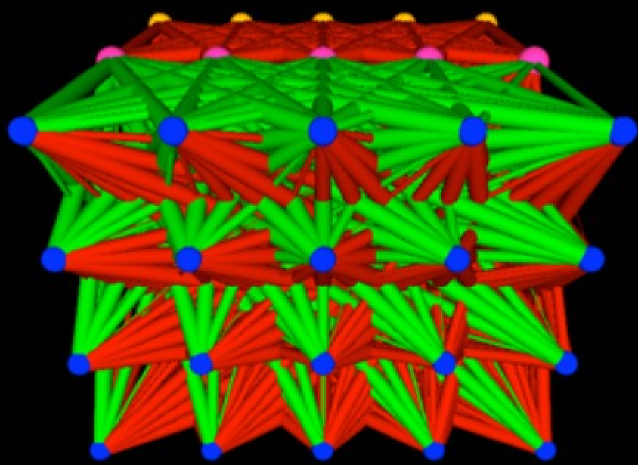
HyperNEAT ANNs are More Regular



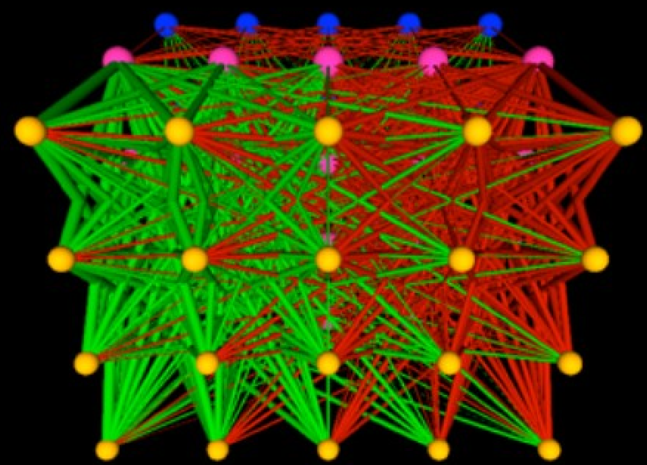
Clune et al. IEEE TEC. 2011

HyperNEAT ANNs have a diverse array of regular patterns

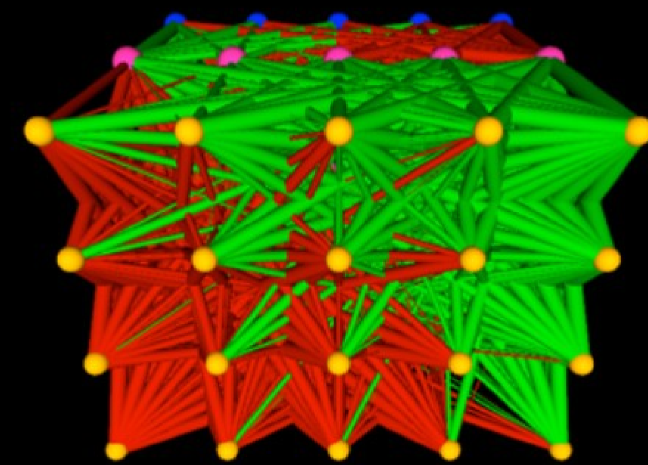
repeat for each node



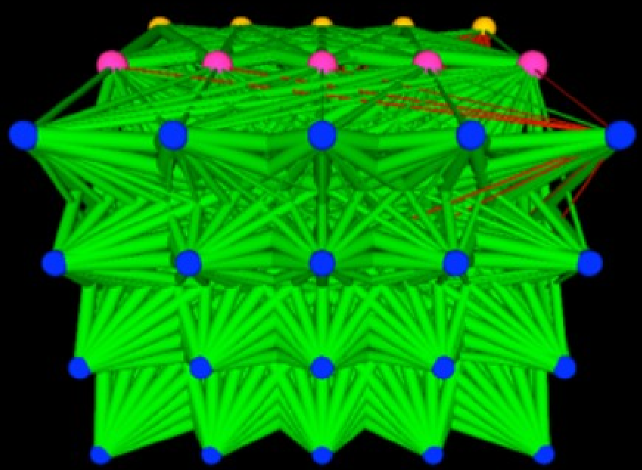
left-right symmetry



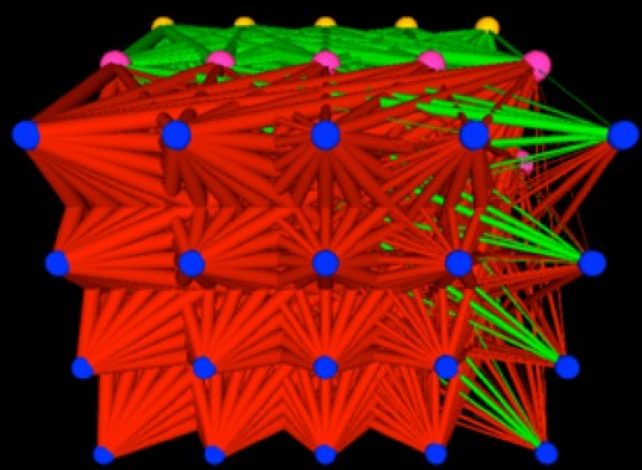
diagonal symmetry



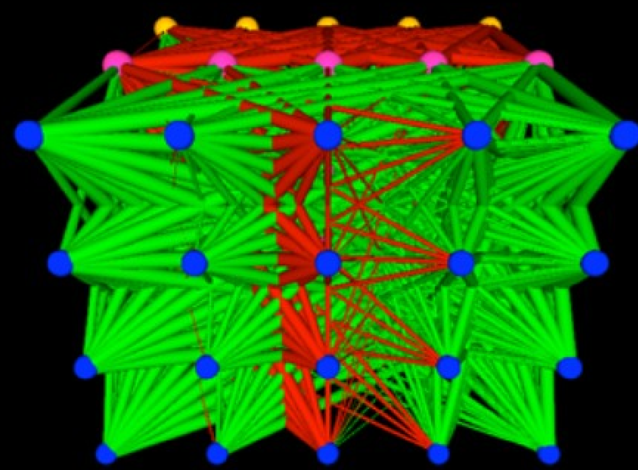
exception for single node



exception for single column



exception in center columns

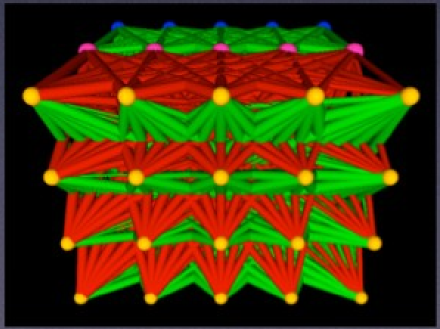


Clune et al. IEEE TEC. 2011

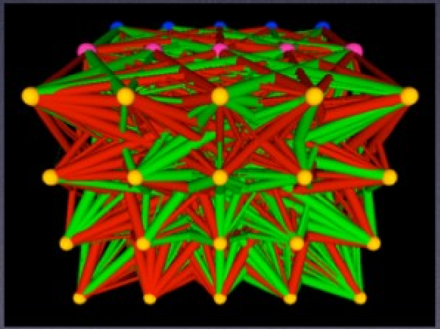
HyperNEAT ANNs are More Regular

Test: Compression (gzip)

- HyperNEAT compress more than FT-NEAT ($p < .05$)
  - HyperNEAT: mean 4488 bytes ± 710
  - FT-NEAT: mean 3349 bytes ± 37
- This quantitative measure supports the descriptive evidence



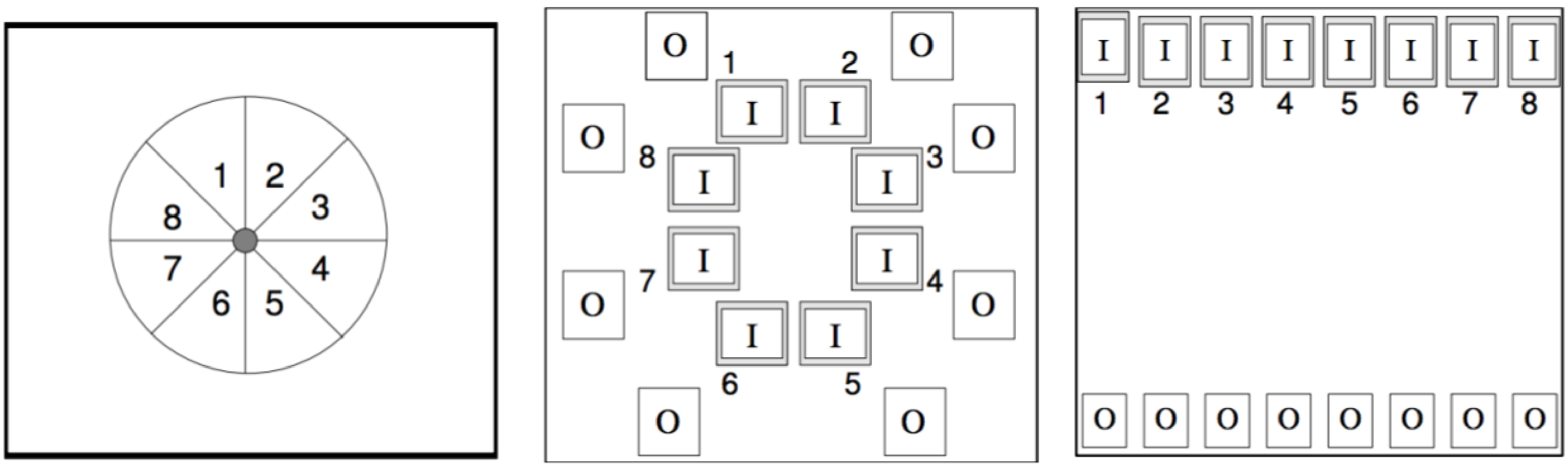
HyperNEAT



FT-NEAT

Clune et al. IEEE TEC. 2011

HyperNEAT: Geometrically Aware

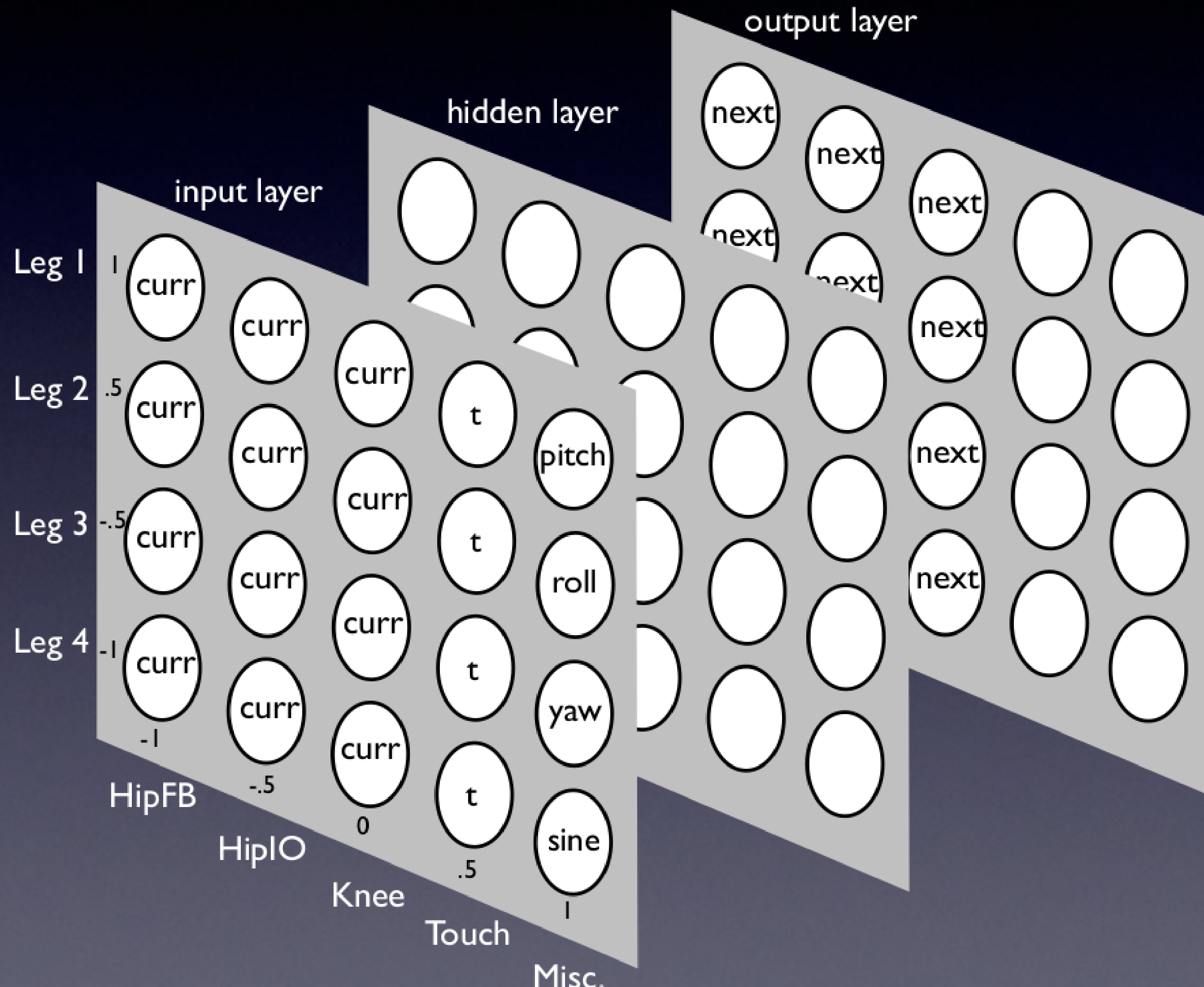


(a) Robot (b) Concentric (c) Parallel

Stanley et al. Alife. 2009

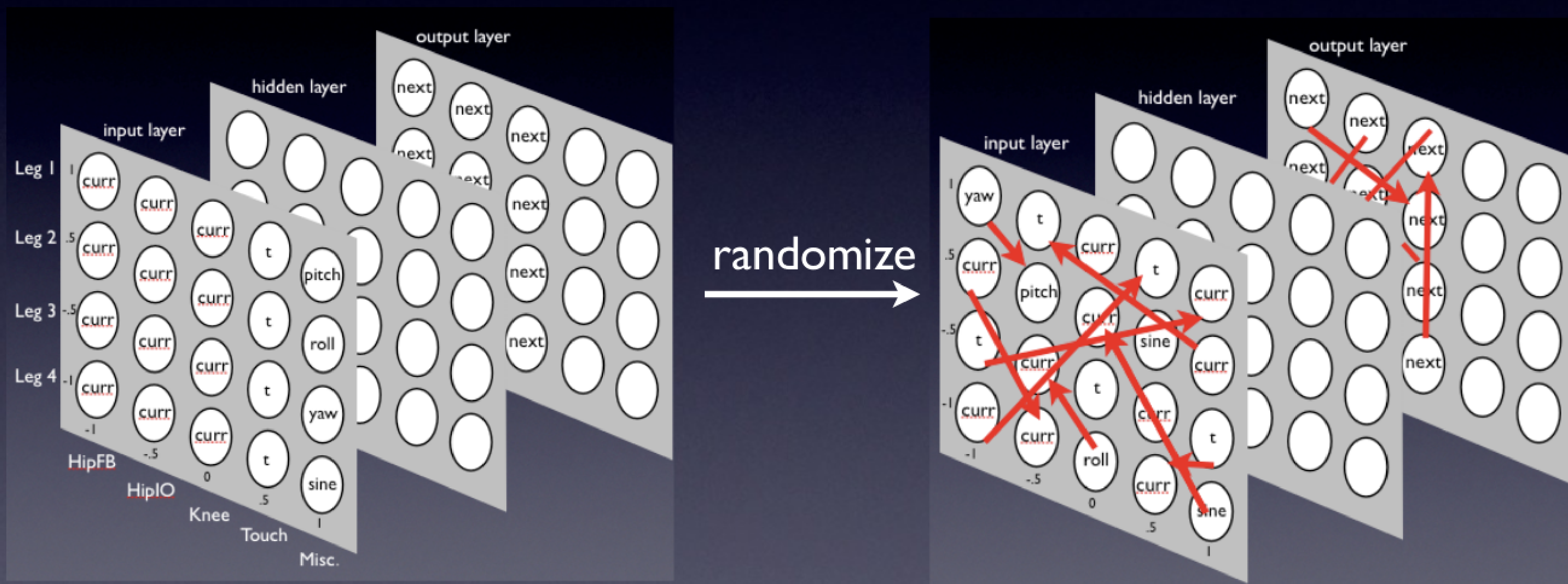
Engineered Representation

ANN Controller

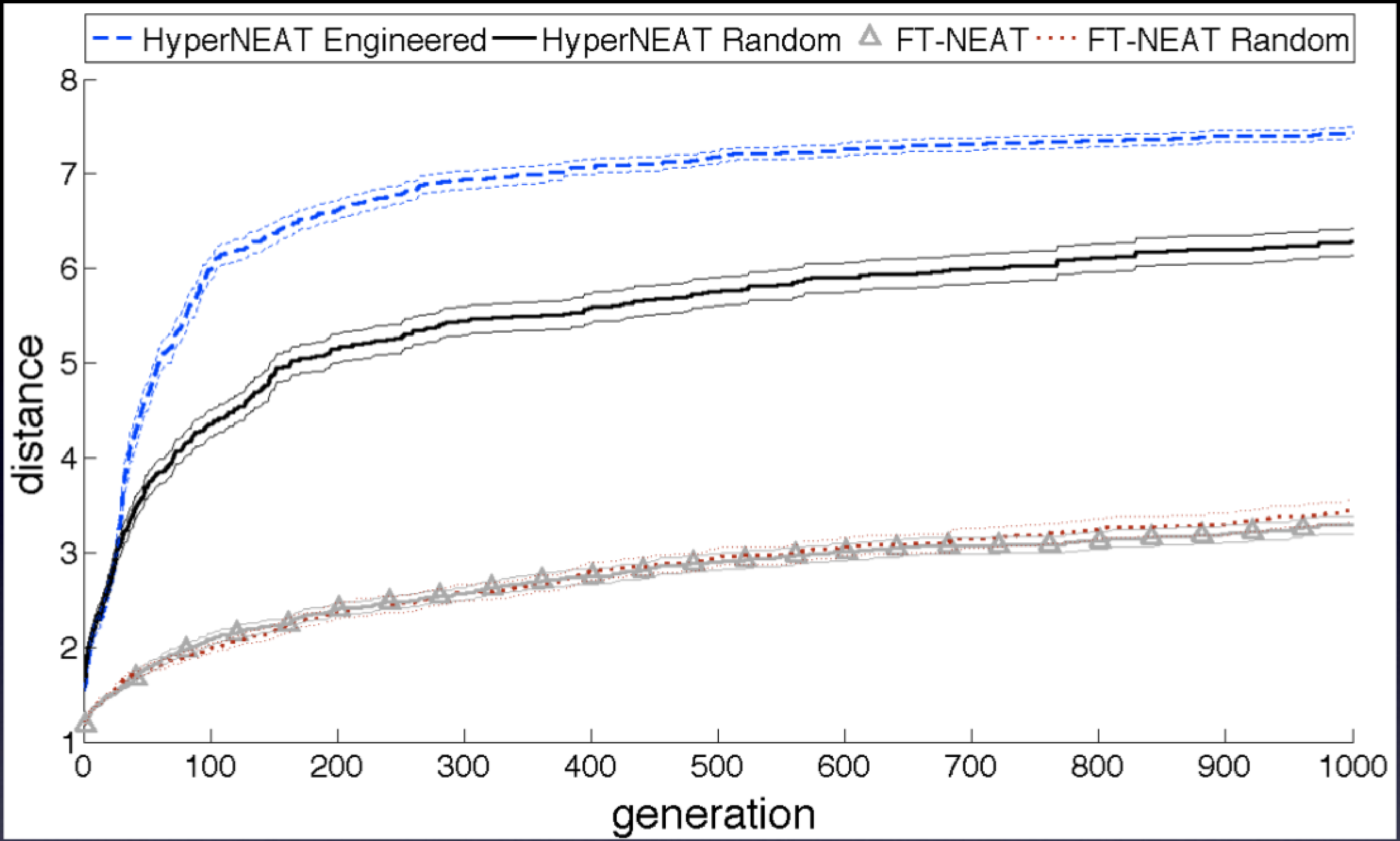


Clune et al. CEC 2008

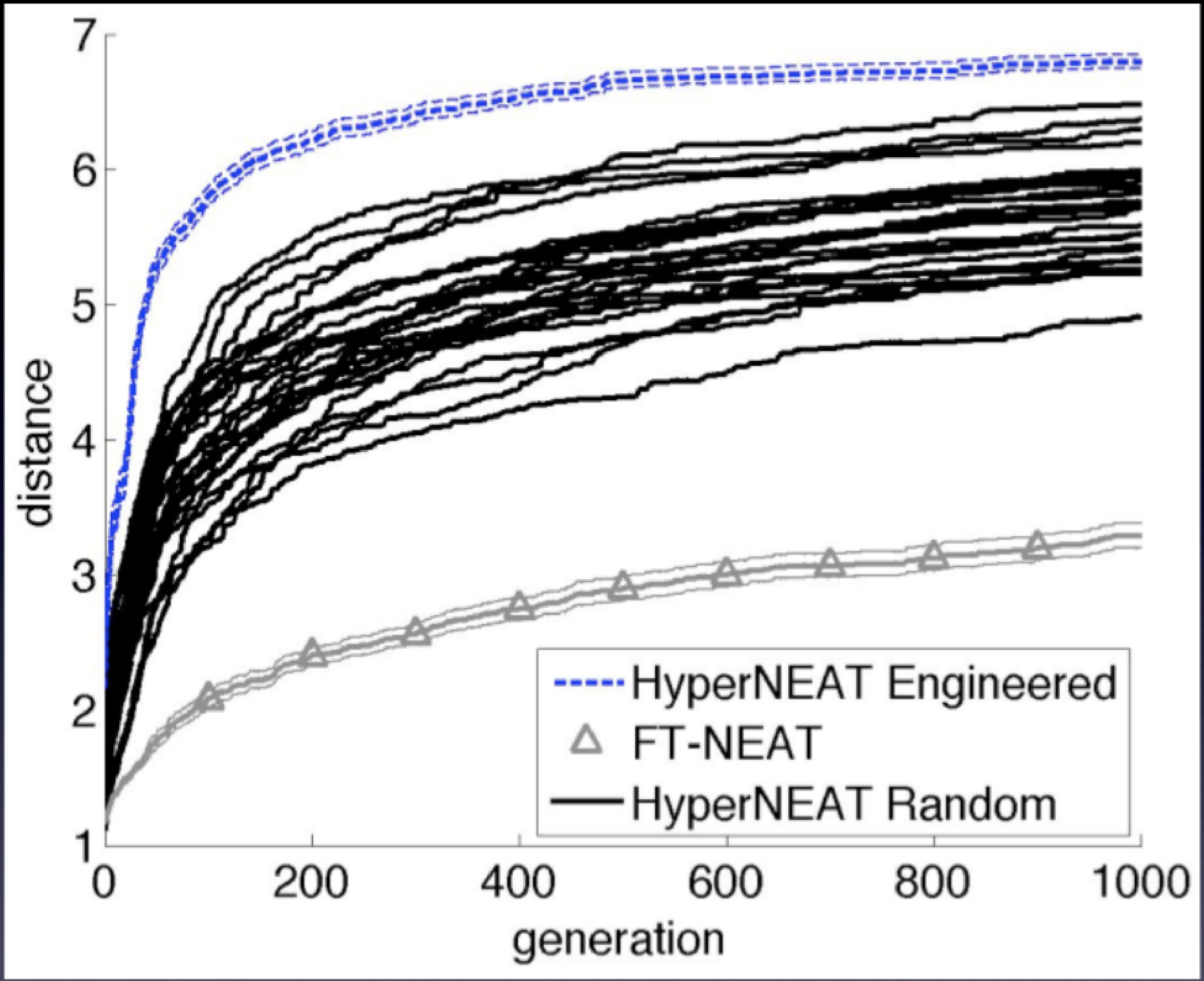
Evaluating Representation



- Each of the 50 trials had a different randomized representation for the entire run



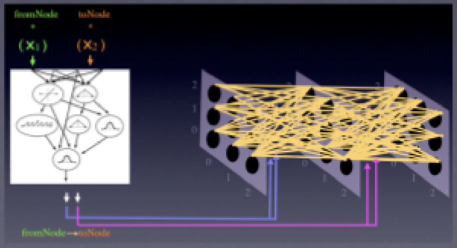
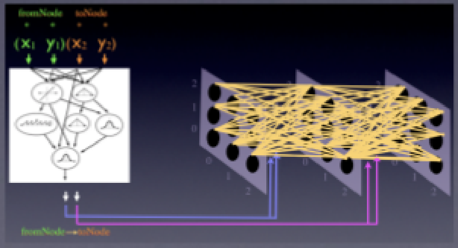
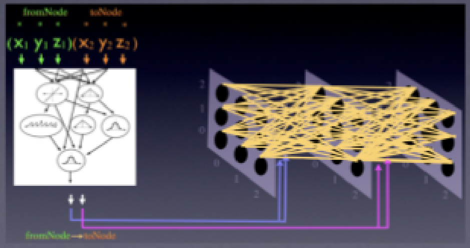
- Engineered rep. beats Randomized rep. ($p < .05$)
  - Representations can make a difference
  - Human intuitions help
- But Randomized rep. beats direct encoding control ($p < .001$)
  - HyperNEAT can outperform direct encoding without an 'intelligent' rep.
  - Could be b/c of its generative nature, or via exploiting randomly generated regularity

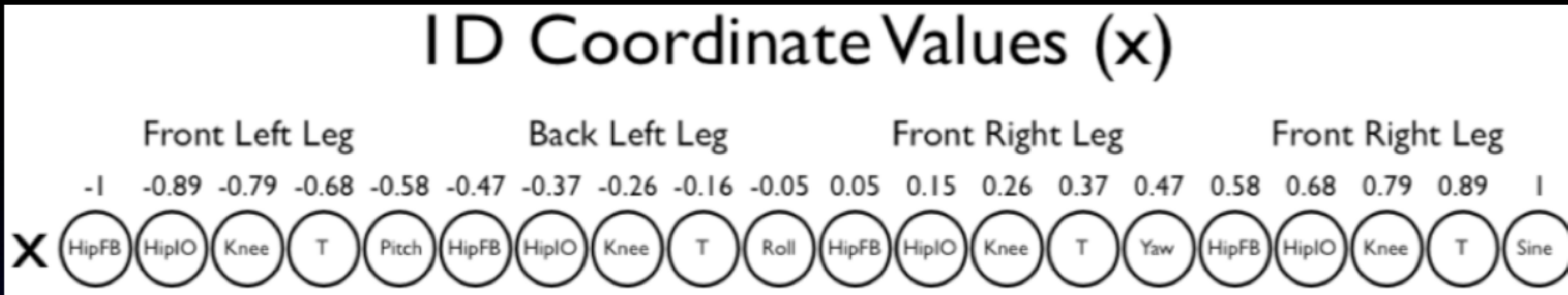


- The performance between the best and worst randomized rep. was significant ($p < .001$)
- Engineered HyperNEAT < Randomized HyperNEAT < FT-NEAT ($p < .01$)
  - HyperNEAT outperforms FT-NEAT on regular problems (Clune et al. PPSN 2008)

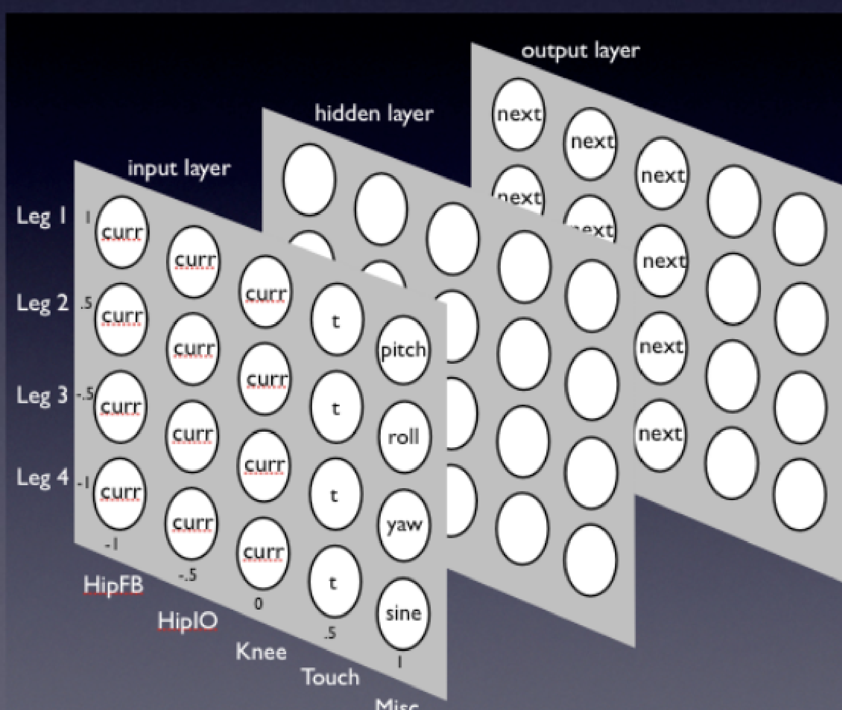
Different Dimensions

- same problem with representations of different dimensions
  - not tested before
  - 1, 2 & 3 dimensional representations
  - extra dimensions could help
    - can make certain distinctions easier
    - the true geometry of this problem is 3D
  - ... or hurt
    - extra inputs to the CPPN

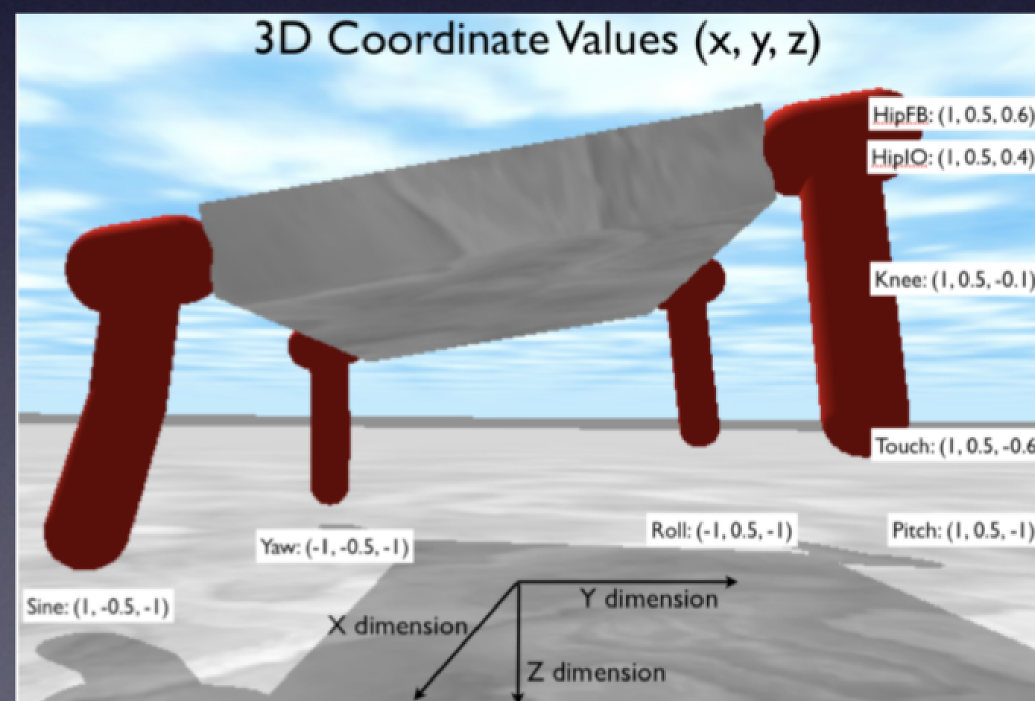





1D Coordinate Values (x)
Front Left Leg
Back Left Leg
Front Right Leg
Front Right Leg
-1 -0.89 -0.79 -0.68 -0.58 -0.47 -0.37 -0.26 -0.16 -0.05 0.05 0.15 0.26 0.37 0.47 0.58 0.68 0.79 0.89 1
x
HipFB HipIO Knee T Pitch HipFB HipIO Knee T Roll HipFB HipIO Knee T Yaw HipFB HipIO Knee T Sine

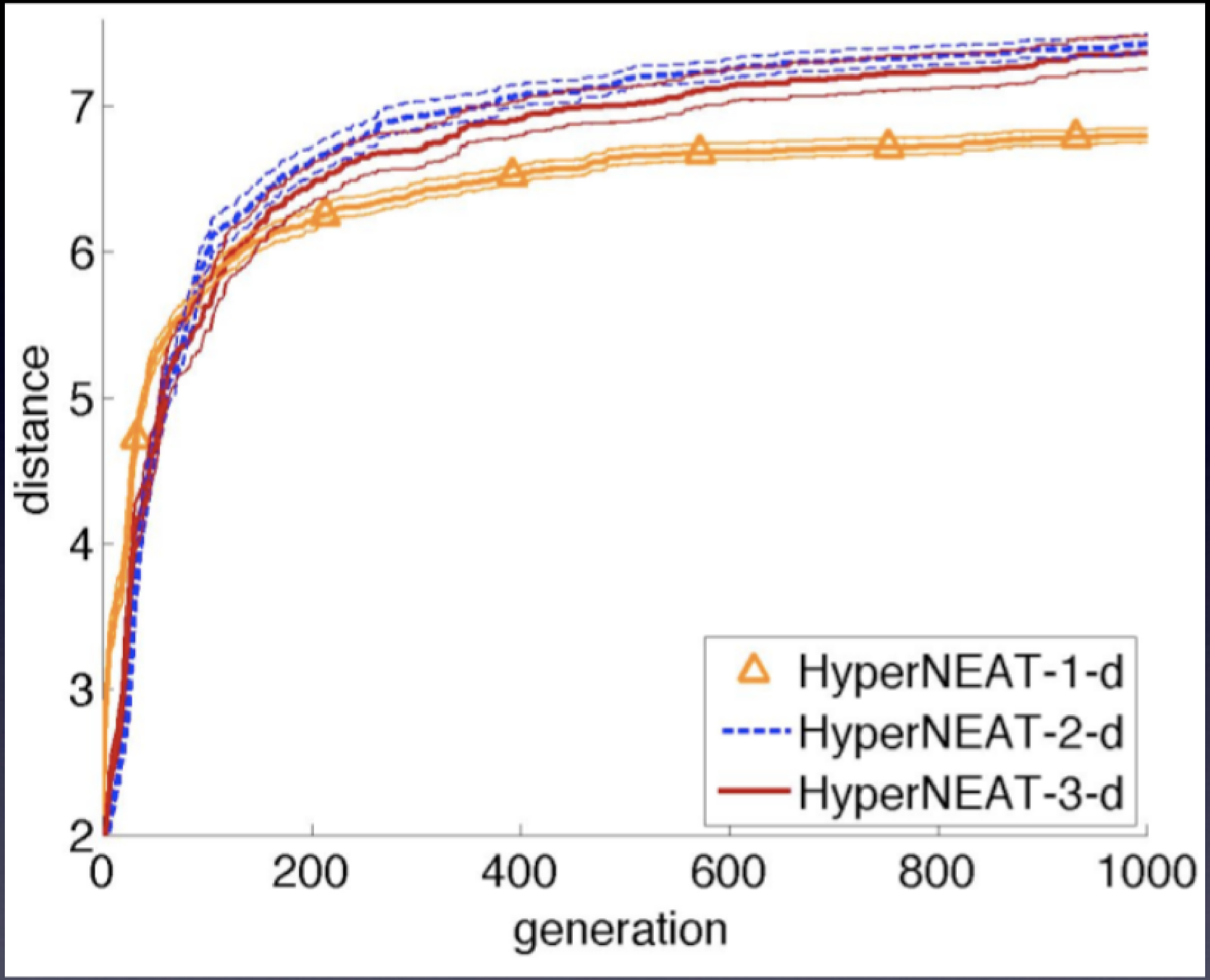
2D (X, Y)


output layer
hidden layer
input layer
Leg 1
Leg 2
Leg 3
Leg 4
curr
t
pitch
roll
yaw
sine
next
HipFB
HipIO
Knee
Touch
Misc.

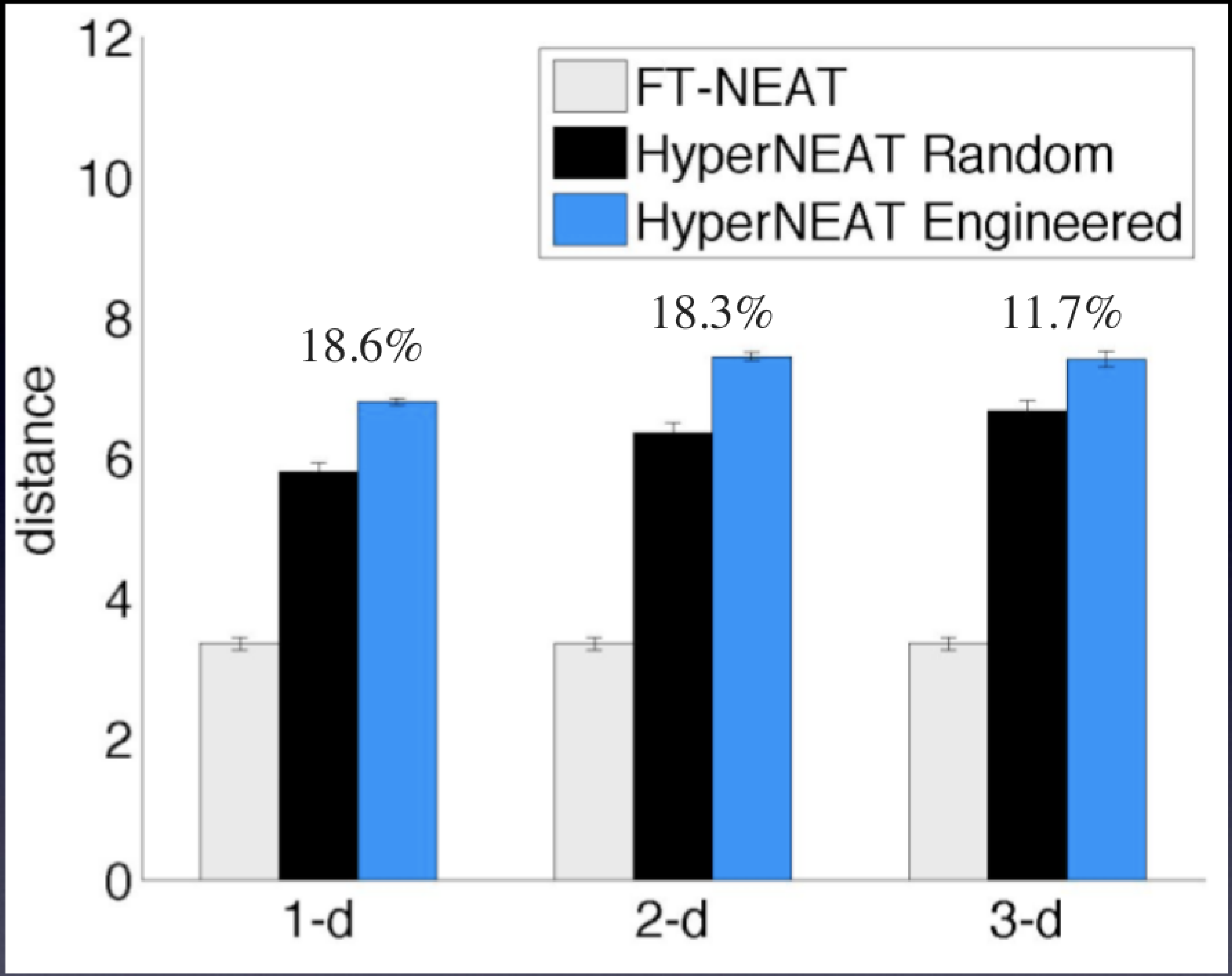
3D (X, Y, Z)


3D Coordinate Values (x, y, z)
HipFB: (1, 0.5, 0.6)
HipIO: (1, 0.5, 0.4)
Knee: (1, 0.5, -0.1)
Touch: (1, 0.5, -0.6)
Roll: (-1, 0.5, -1)
Pitch: (1, 0.5, -1)
Yaw: (-1, -0.5, -1)
Sine: (1, -0.5, -1)
Y dimension
X dimension
Z dimension

Different Dimensions



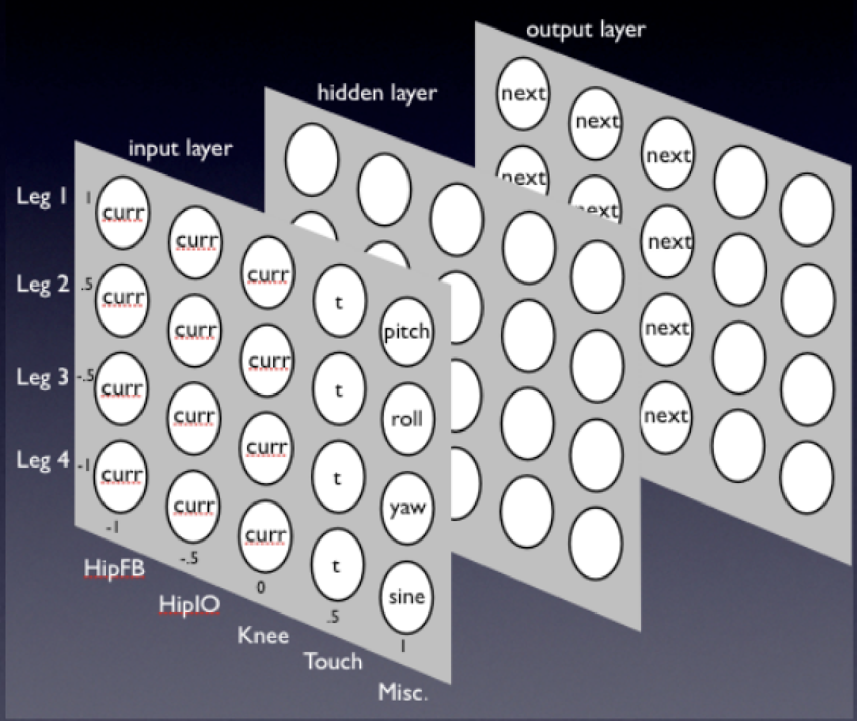
- 1D outperforms 2D & 3D gens 1-58 ($p < .05$), but underperforms after gen. 170 ($p < .05$)
  - 1D simpler, but less powerful?
  - 1D less accurate with respect to true problem geometry? (e.g. few symmetries)
- 2D & 3D treatments statistically indistinguishable ($p > .05$)



- For all:
  - Engineered rep. beats Randomized rep ($p < .05$)
  - Randomized rep. beats direct encoding ($p < .001$)

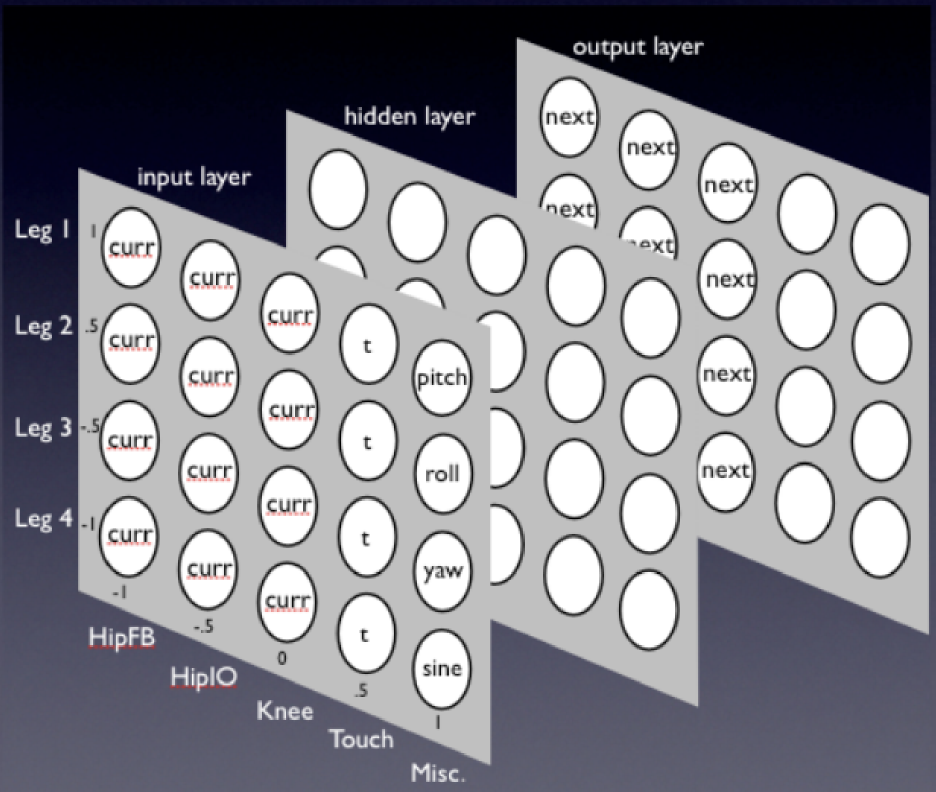
Different Engineered Configurations

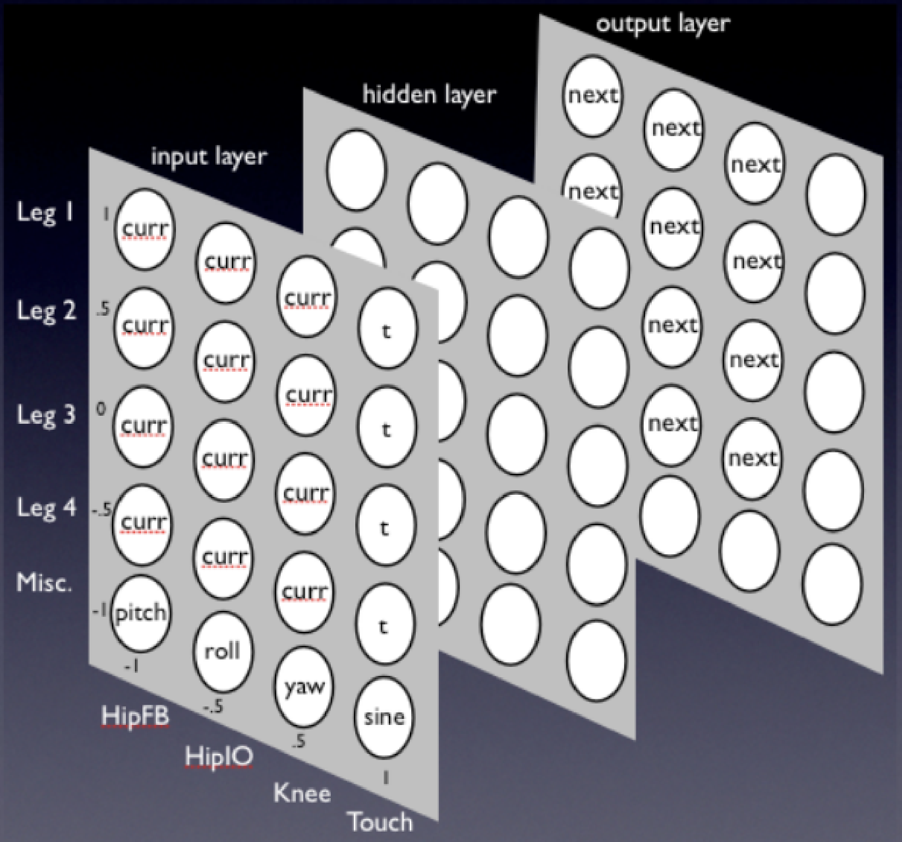
- Some arbitrary choices
- Some difficult choices
- Do they matter?



Arbitrary Design Decisions

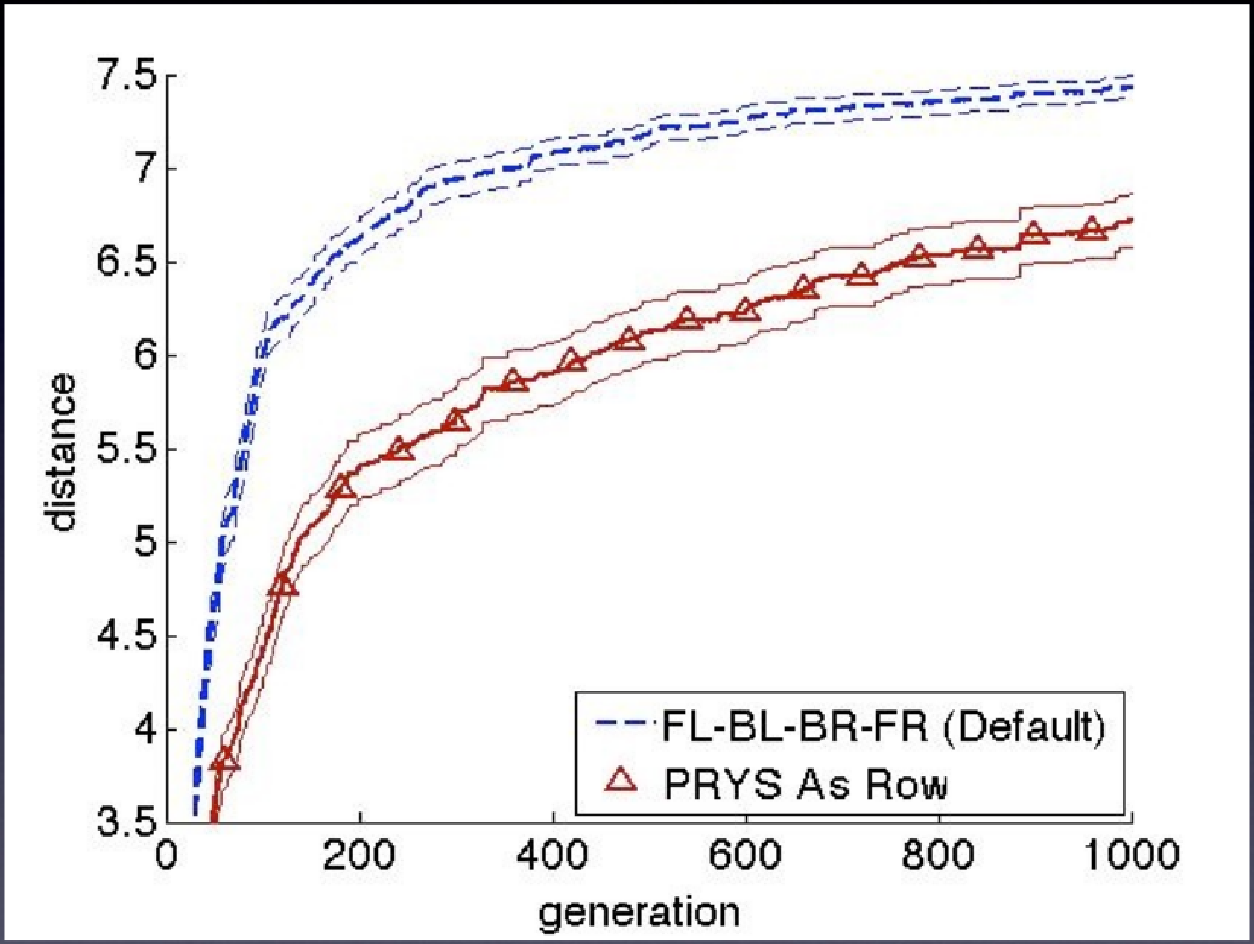
- Ideally it makes no difference



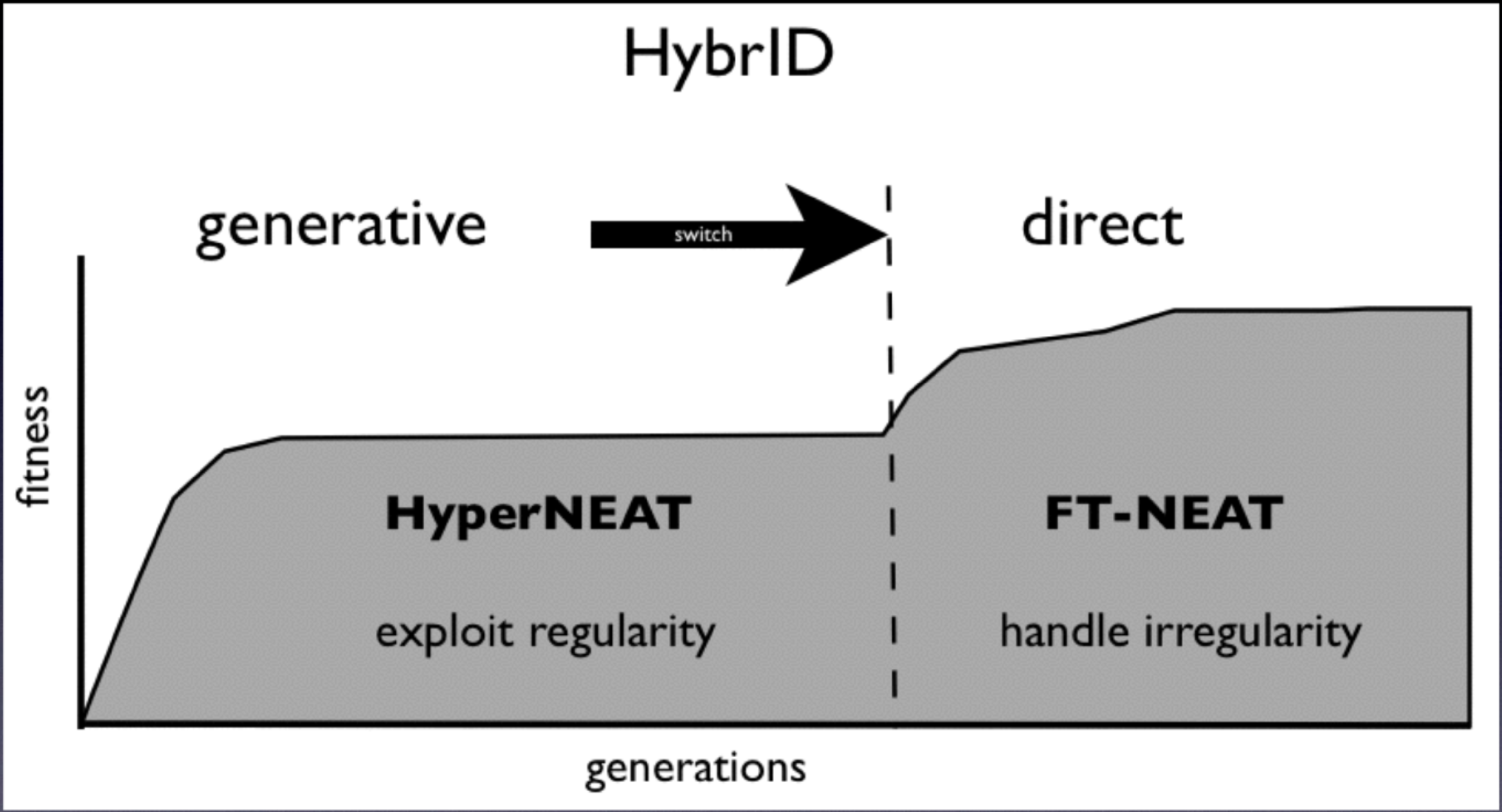


Default (2D Engineered) vs. PRYS as Row

Arbitrary Design Decisions

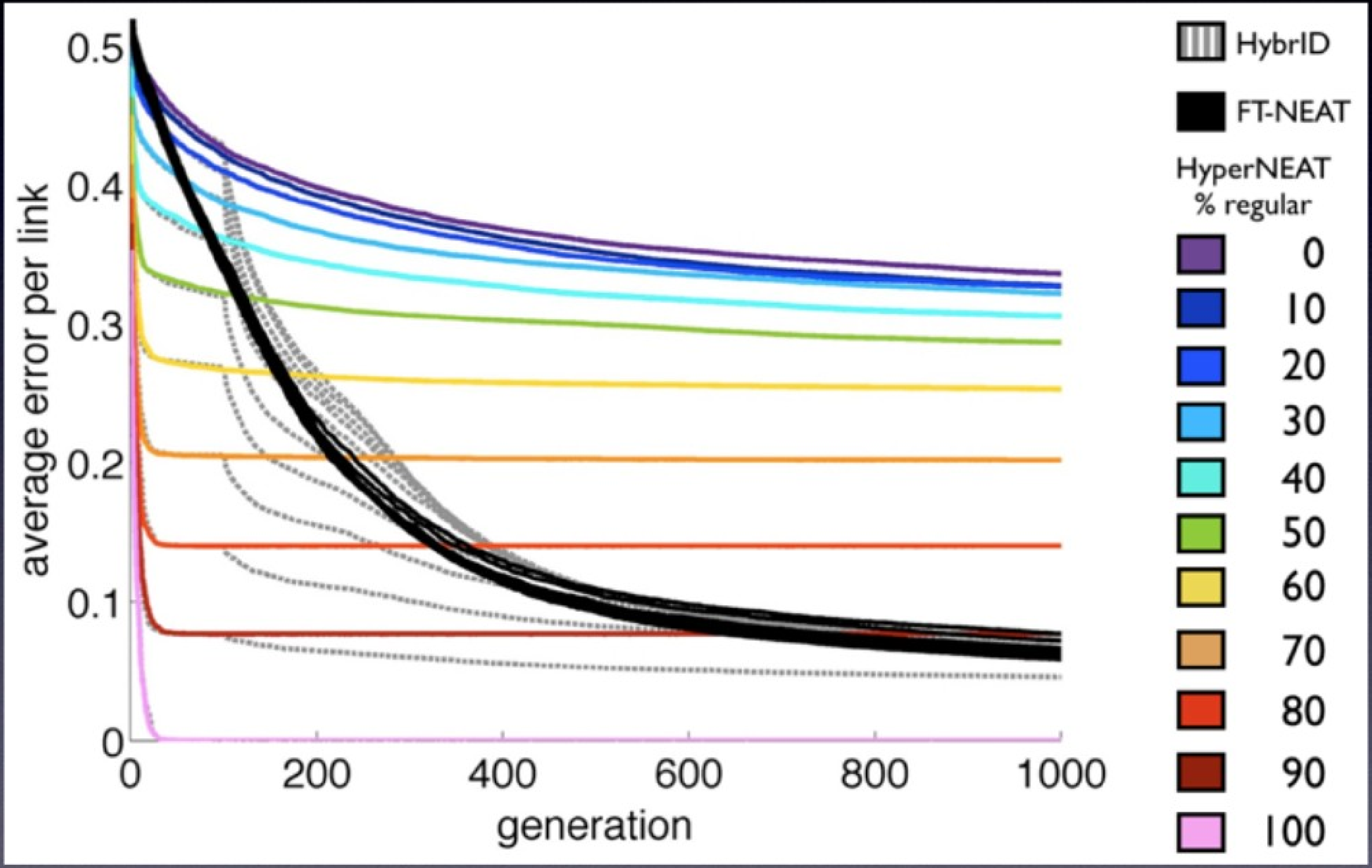


- PRYS as Row does 9.7% worse ($p < .001$)
- Demonstrates that seemingly arbitrary decisions can have a significant impact



Clune et al. ECAL 2009

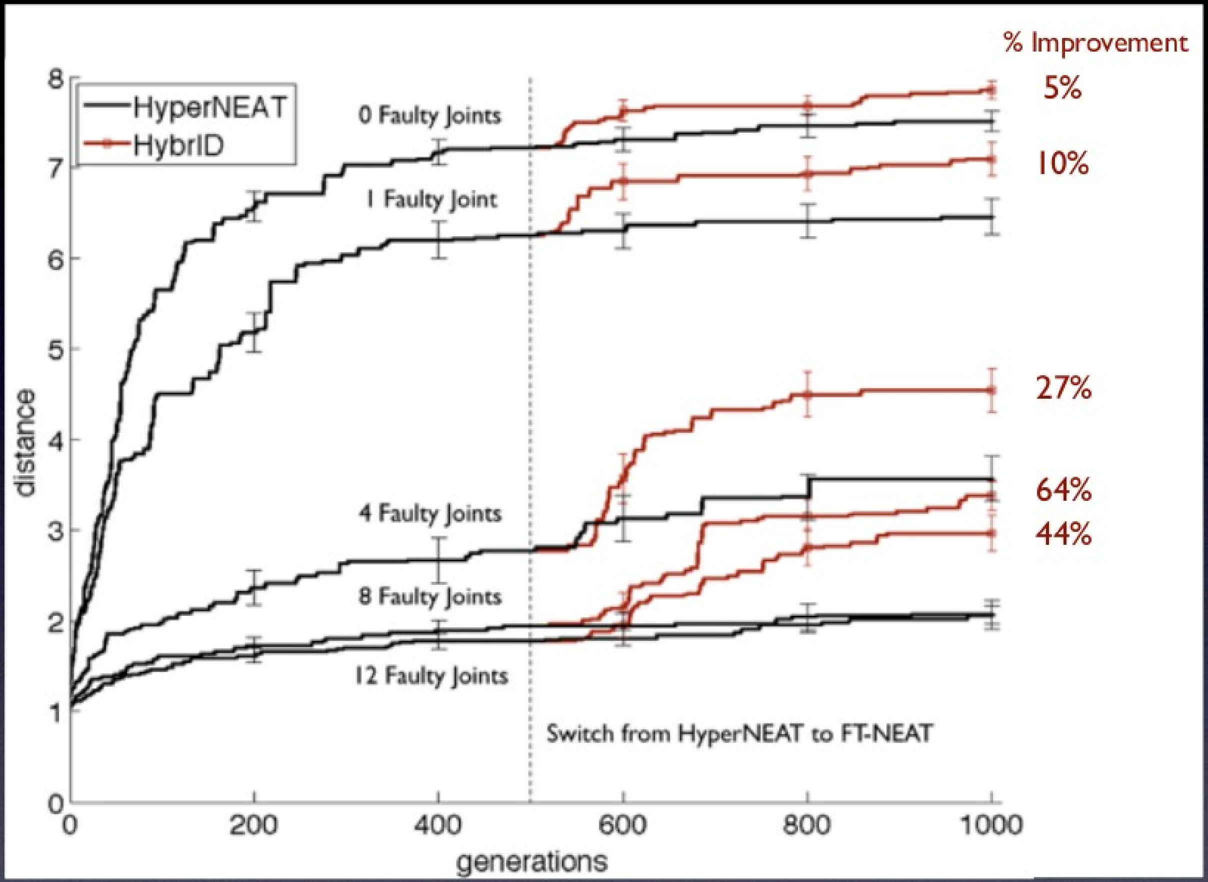
HybrID > HyperNEAT & FT-NEAT on Target Weights



HybrID > HyperNEAT & FT-NEAT at generation 250 on 70%, 80%, and 90% *($p < .01$)*

Clune et al. ECAL 2009

HybrID > HyperNEAT on Quadruped Controller



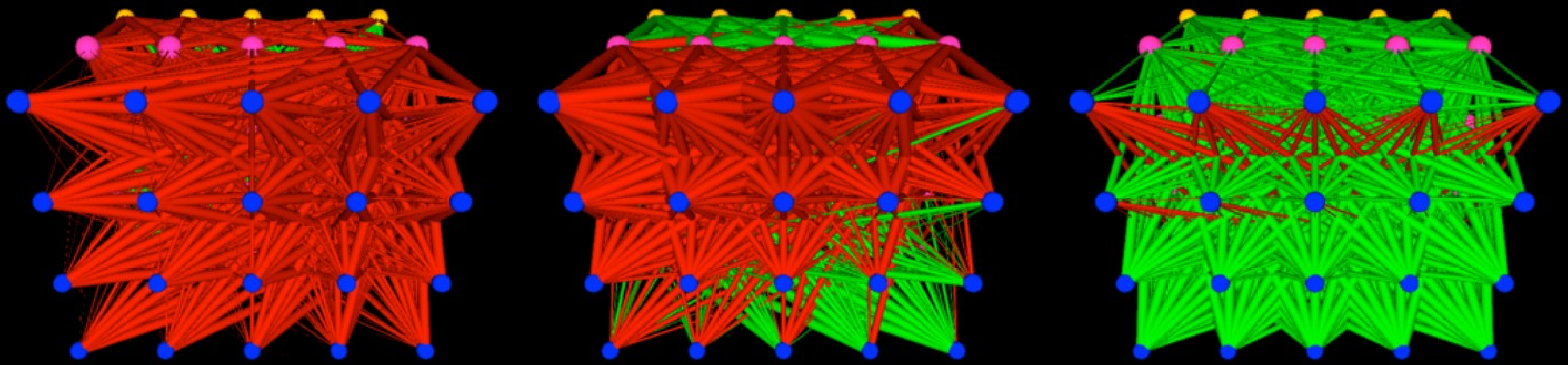
HybrID > HyperNEAT on all *($p < .001$)*

Clune et al. CEC. 2009

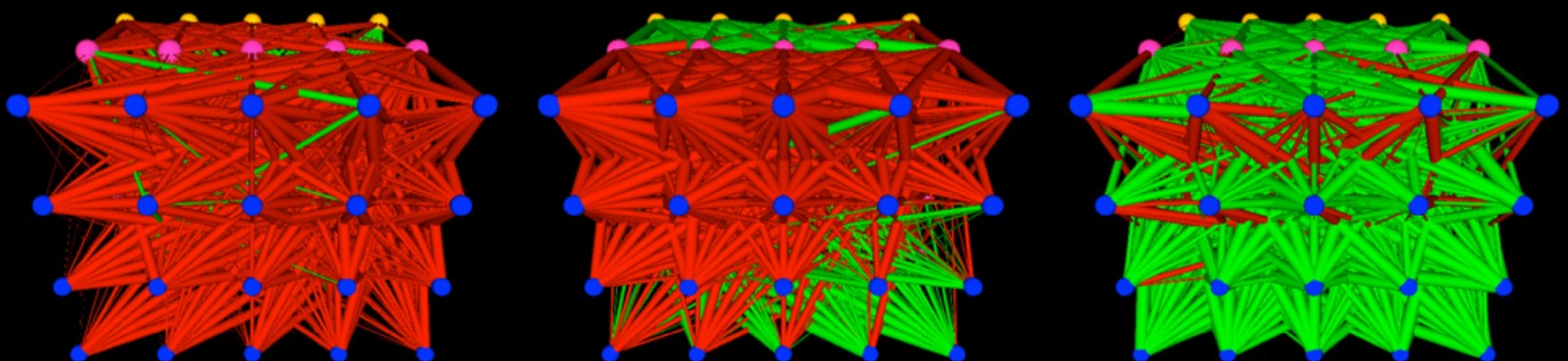
Clune et al. IEEE TEC. 2011

HybrID Changes

After HyperNEAT Phase



After FT-NEAT Phase



Clune et al. IEEE TEC. 2011

HybrID implications

- HybrID >= HyperNEAT on all problems
- Suggests generative encodings have difficulty adjusting patterns in irregular ways
- HybrID offers path forward: a process of refinement
    - generative encoding + direct encoding
    - generative encoding + lifetime learning

Evolving Morphologies with Lindenmayer Systems

Grammar (alphabet, axiom, production rules)

alphabet – set of possible symbols

axiom – initial state of the system

production rules – definition of symbol replacements to grow the system

Fractal (binary) tree

- **variables** : 0, 1
- **constants**: [,]
- **axiom** : 0
- **rules** : (1 → 11), (0 → 1[0]0)

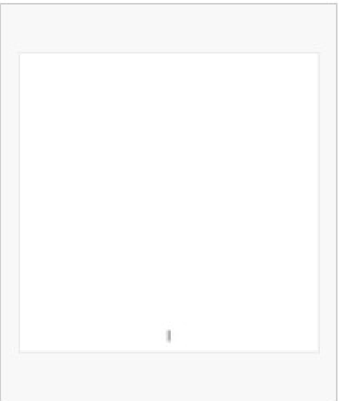
- 0: draw a line segment ending in a leaf
- 1: draw a line segment
- [: push position and angle, turn left 45 degrees
-]: pop position and angle, turn right 45 degrees

axiom: 0

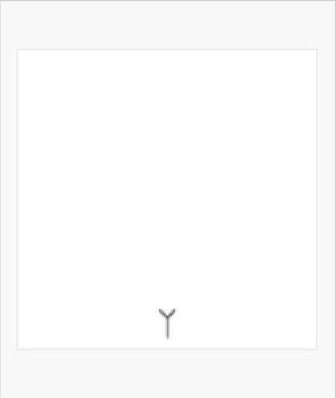
1st recursion: 1[0]0

2nd recursion: 11[1[0]0]1[0]0

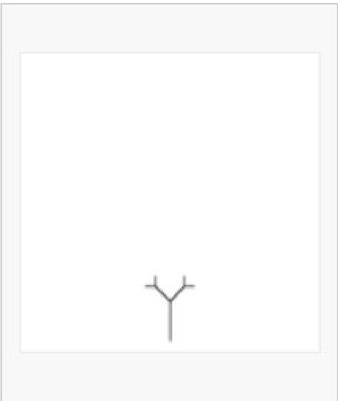
3rd recursion: 1111[11[1[0]0]1[0]0]11[1[0]0]1[0]0



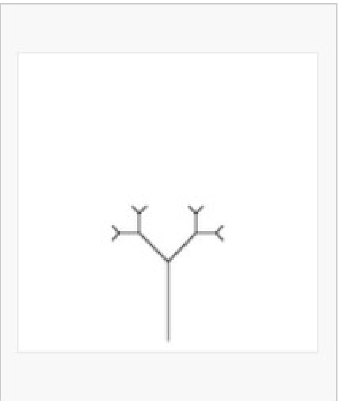
Axiom



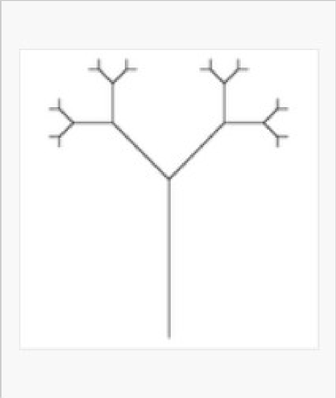
First recursion



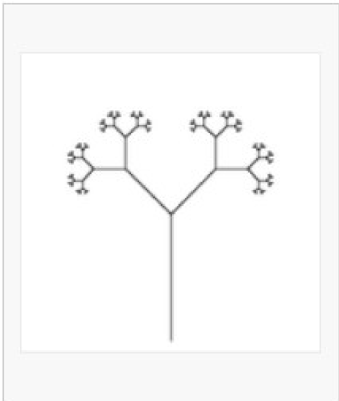
Second recursion



Third recursion



Fourth recursion



Seventh recursion, scaled down ten times

Cantor set

variables : A B

constants : none

start : A {starting character string}

rules : (A → ABA), (B → BBB)

Let *A* mean "draw forward" and *B* mean "move forward".

Koch curve

variables : F

constants : + −

start : F

rules : (F → F+F−F−F+F)

F means "draw forward",

+ means "turn left 90°",

− means "turn right 90°"

n = 0:

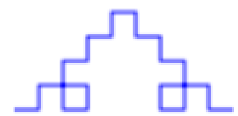
F

n = 1:

F+F−F−F+F

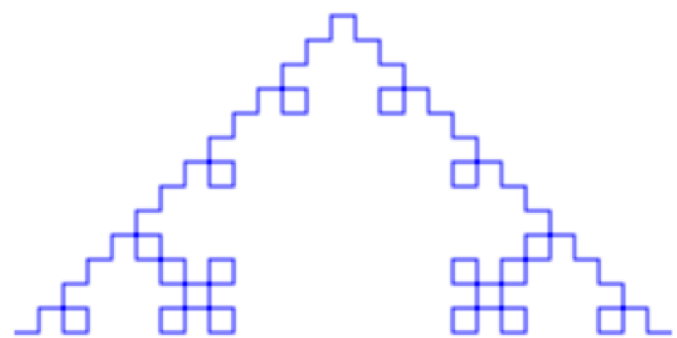
n = 2:

F+F−F−F+F + F+F−F−F+F − F+F−F−F+F − F+F−F−F+F + F+F−F−F+F



n = 3:

F+F−F−F+F+F+F−F−F+F−F+F−F−F+F−F+F−F−F+F+F+F−F−F+F +
F+F−F−F+F+F+F−F−F+F−F+F−F−F+F−F+F−F−F+F+F+F−F−F+F −
F+F−F−F+F+F+F−F−F+F−F+F−F−F+F−F+F−F−F+F+F+F−F−F+F −
F+F−F−F+F+F+F−F−F+F−F+F−F−F+F−F+F−F−F+F+F+F−F−F+F +
F+F−F−F+F+F+F−F−F+F−F+F−F−F+F−F+F−F−F+F+F+F−F−F+F



Sierpinski triangle

variables : F G

constants : + −

start : F−G−G

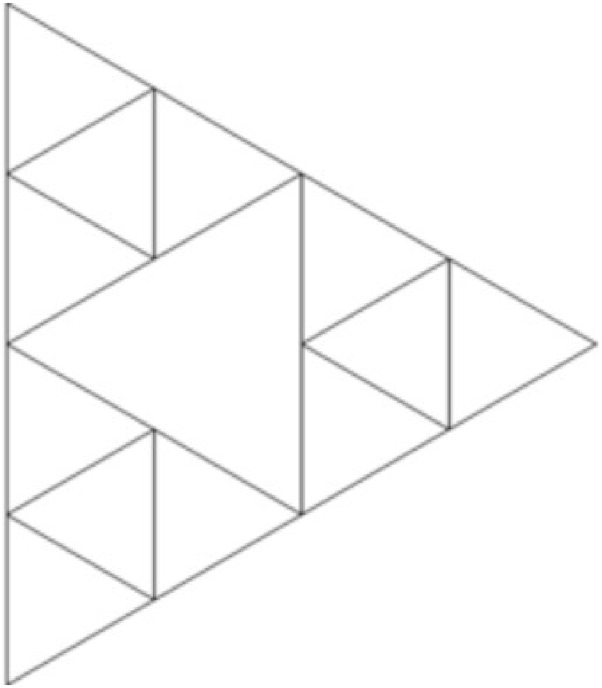
rules : (F → F−G+F+G−F), (G → GG)

angle : 120°

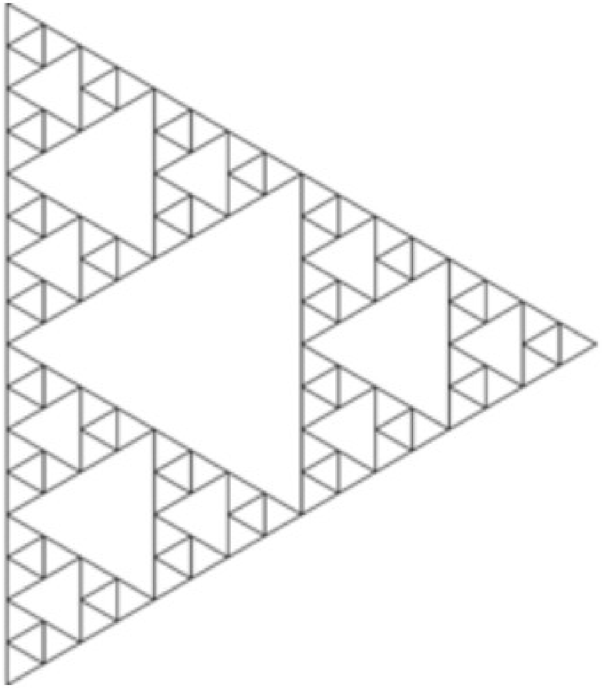
F and G both mean "draw forward",

+ means "turn left by angle",

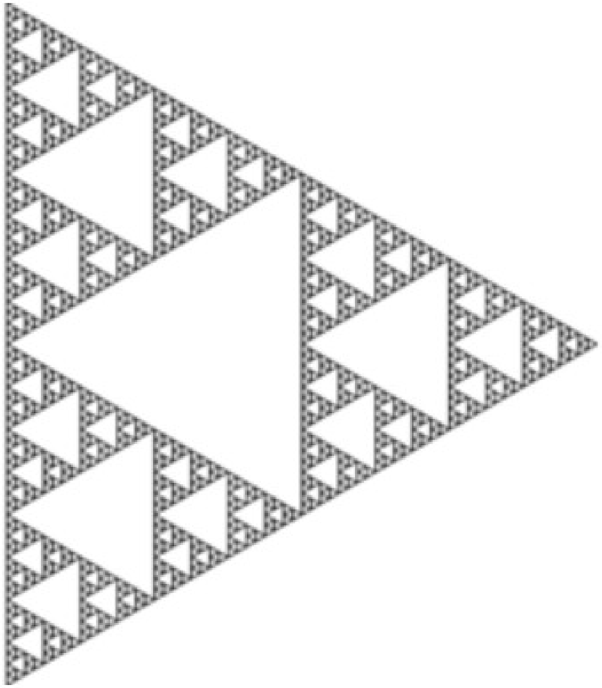
− means "turn right by angle".



n = 2



n = 4



n = 6

Sierpinski triangle

variables : A B

constants : + −

start : A

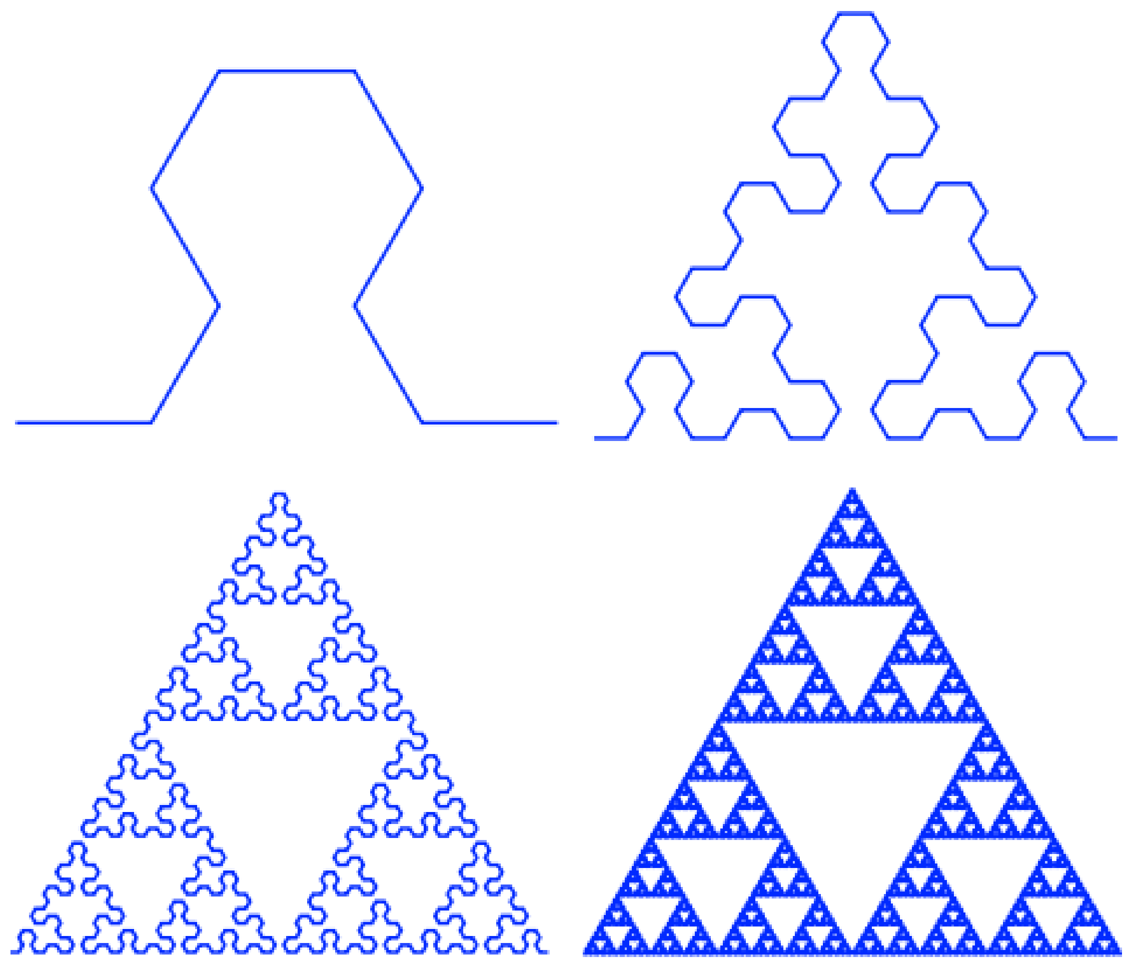
rules : (A → B−A−B), (B → A+B+A)

angle : 60°

F and G both mean "draw forward",

\+ means "turn left by angle",

− means "turn right by angle".



Fractal plant

variables : X F

constants : + − [ ]

start : X

rules : (X → F[−X][X]F[−X]+FX), (F → FF)

angle : 25°

F means "draw forward",

− means "turn left 25°",

+ means "turn right 25°"

X does not correspond to any drawing action

[: push position and angle,

]: pop position and angle

Fractal plant for *n* = 6

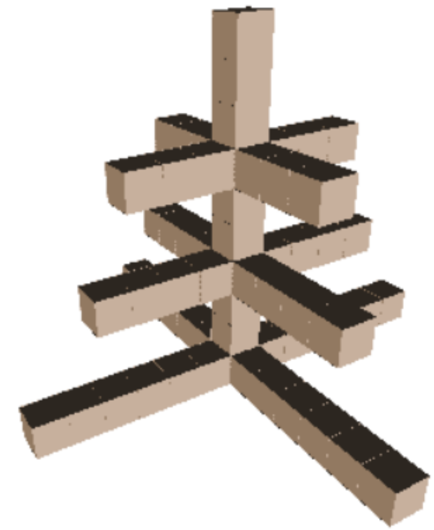
The Advantages of Generative Grammatical Encodings for Physical Design

Gregory S. Hornby
415 South Street
DEMO Lab
Brandeis University
Waltham, MA 02454
hornby@cs.brandeis.edu

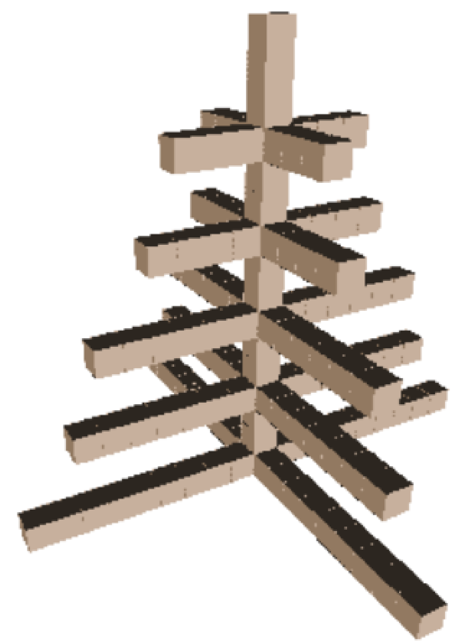
Jordan B. Pollack
415 South Street
DEMO Lab
Brandeis University
Waltham, MA 02454
pollack@cs.brandeis.edu

Table 1: Design Language

Command	Description	Symbol
[ ]	push/pop orientation to stack	[ ]
{ *block* }(n)	repeat enclosed block n times	{ }
forward(n)	move in the turtle's positive X direction n units	f
backward(n)	move in the turtle's negative X direction n units	b
up(n)	rotate heading $n \times 90°$ about the turtle's Z axis	∧
down(n)	rotate heading $n \times -90°$ about the turtle's Z axis	v
left(n)	rotate heading $n \times 90°$ about the turtle's Y axis	<
right(n)	rotate heading $n \times -90°$ about the turtle's Y axis	>
clockwise(n)	rotate heading $n \times 90°$ about the turtle's X axis	/
counter-clockwise(n)	rotate heading $n \times -90°$ about the turtle's X axis	\



[{ [forward(6)] left(1) }(4)] up(1) forward(3) down(1) [{ [forward(4.5)] left(1) }(4)] up(1) forward(3) down(1) [{ [forward(3)] left(1) }(4)] up(1) forward(3) down(1) forward(3)



Mutation

For example, if the production $P0$ is selected to be mutated,

$$\begin{aligned} P0(n0,n1): \quad & n0 > 5.0 \rightarrow \{\ a(1.0)\ b(2.0)\ \}(n1) \\ & n0 > 2.0 \rightarrow [\ P1(n1-1.0, n0/2.0)\ ] \end{aligned}$$

some of the possible mutations are,
Mutate the condition:

$$\begin{aligned} P0(n0,n1): \quad & n0 > \mathbf{5.0} \rightarrow \{\ a(1.0)\ b(2.0)\ \}(n1) \\ & n0 > 2.0 \rightarrow [\ P1(n1-1.0, n0/2.0)\ ] \end{aligned}$$

Mutate an argument:

$$\begin{aligned} P0(n0,n1): \quad & n0 > 5.0 \rightarrow \{\ a(1.0)\ b(2.0)\ \}(n1) \\ & n0 > 2.0 \rightarrow [\ P1(\mathbf{n1-2.0}, n0/2.0)\ ] \end{aligned}$$

Mutate a symbol:

$$\begin{aligned} P0(n0,n1): \quad & n0 > 5.0 \rightarrow \{\ \mathbf{c(1.0)}\ b(2.0)\ \}(n1) \\ & n0 > 2.0 \rightarrow [\ P1(n1-1.0, n0/2.0)\ ] \end{aligned}$$

Delete random character(s):

$$\begin{aligned} P0(n0,n1): \quad & n0 > 5.0 \rightarrow \{\ a(1.0)\ \}(n1) \\ & n0 > 2.0 \rightarrow [\ P1(n1-1.0, n0/2.0)\ ] \end{aligned}$$

Insert a random sequence of character(s):

$$\begin{aligned} P0(n0,n1): \quad & n0 > 5.0 \rightarrow \{\ a(1.0)\ b(2.0)\ \}(n1)\ \mathbf{c(3.0)} \\ & n0 > 2.0 \rightarrow [\ P1(n1-1.0, n0/2.0)\ ] \end{aligned}$$

Encapsulate a block of characters:

$$\begin{aligned} P0(n0,n1): \quad & n0 > 5.0 \rightarrow \{\ \mathbf{P2(n0,n1)}\ \}(n1) \\ & n0 > 2.0 \rightarrow [\ P1(n1-1.0, n0/2.0)\ ] \\ \mathbf{P2(n0,n1)}: \quad & \mathbf{n0 > 5.0 \rightarrow a(1.0)\ b(2.0)} \\ & \mathbf{n0 > 2.0 \rightarrow a(1.0)\ b(2.0)} \end{aligned}$$

Recombination

For example if parent 1 has the following rule,

$$P3(n0,n1): \quad \begin{array}{l} n0 > 5.0 \rightarrow \{\ a(1.0)\ b(2.0)\ \}(n1) \\ n0 > 2.0 \rightarrow [\ P1(n1 - 1.0, n0/2.0)\ ] \end{array}$$

and parent 2 has the following rule,

$$P3(n0,n1): \quad \begin{array}{l} n1 > 3.0 \rightarrow b(3.0)\ a(2.0) \\ n0 > 1.0 \rightarrow P1(n1 - 1.0, n1 - 2.0) \end{array}$$

Then some of the possible results of a recombination on successor P3 are:
Replace an entire condition-successor pair:

$$P3(n0,n1): \quad \begin{array}{l} \mathbf{n1 > 3.0 \rightarrow b(3.0)\ a(2.0)} \\ n0 > 2.0 \rightarrow [\ P1(n1 - 1.0, n0/2.0)\ ] \end{array}$$

Replace just a successor:

$$P3(n0,n1): \quad \begin{array}{l} n0 > 5.0 \rightarrow \{\ a(1.0)\ b(2.0)\ \}(n1) \\ n0 > 2.0 \rightarrow \mathbf{P1(n1 - 1.0, n1 - 2.0)} \end{array}$$

Replace one block with another:

$$P3(n0,n1): \quad \begin{array}{l} n0 > 5.0 \rightarrow \{\ a(1.0)\ b(2.0)\ \}(n1) \\ n0 > 2.0 \rightarrow [\ \mathbf{b(3.0)\ a(2.0)}\ ] \end{array}$$

$$
\begin{aligned}
f_{height} &= \text{the height of the highest voxel, } Y_{max}. \\
f_{surface} &= \text{the number of voxels at } Y_{max}. \\
f_{stability} &= \sum_{y=0}^{Y_{max}-1} f_{area}(y) \\
f_{area}(y) &= \text{area in the convex hull at height y.} \\
f_{excess} &= \text{number of voxels not on the surface.}
\end{aligned}
$$

For these experiments we combine these measures into a single function [1],

$$
\text{fitness} = f_{height} \times f_{surface} \times f_{stability} / f_{excess} \quad (1)
$$

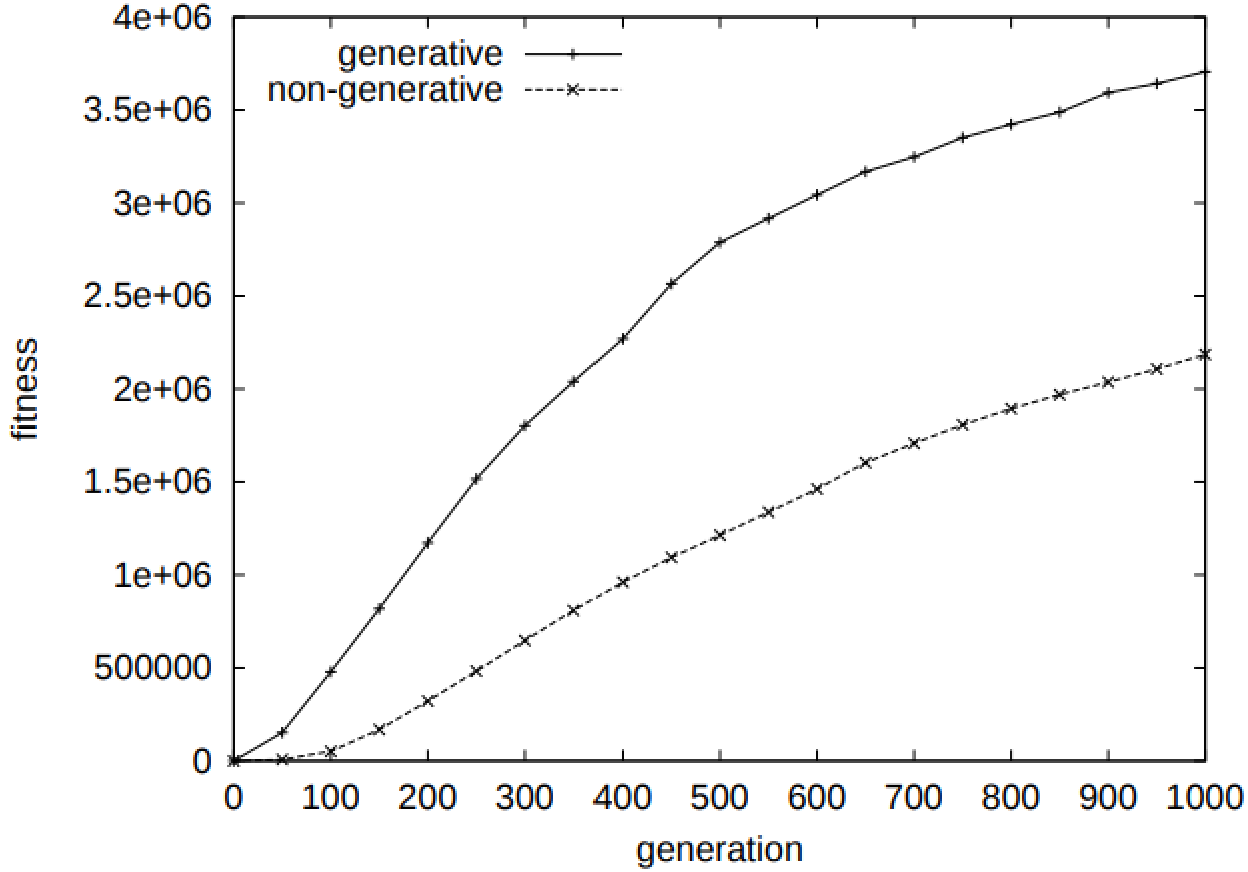
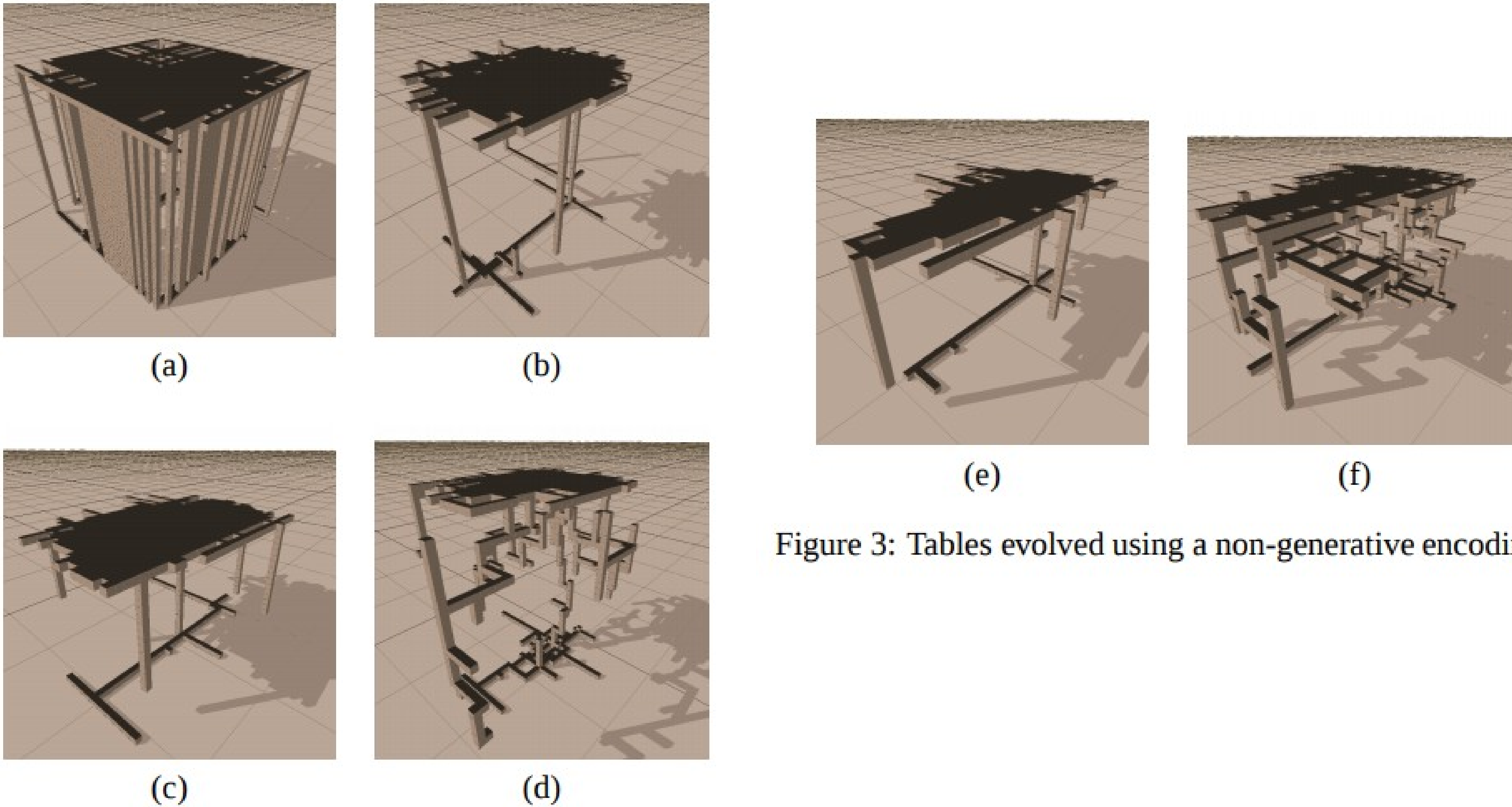


Figure 2: Performance comparison between the non-generative encoding and the P0L-system generative encoding.



(a) (b) (c) (d) (e) (f)

Figure 3: Tables evolved using a non-generative encoding.

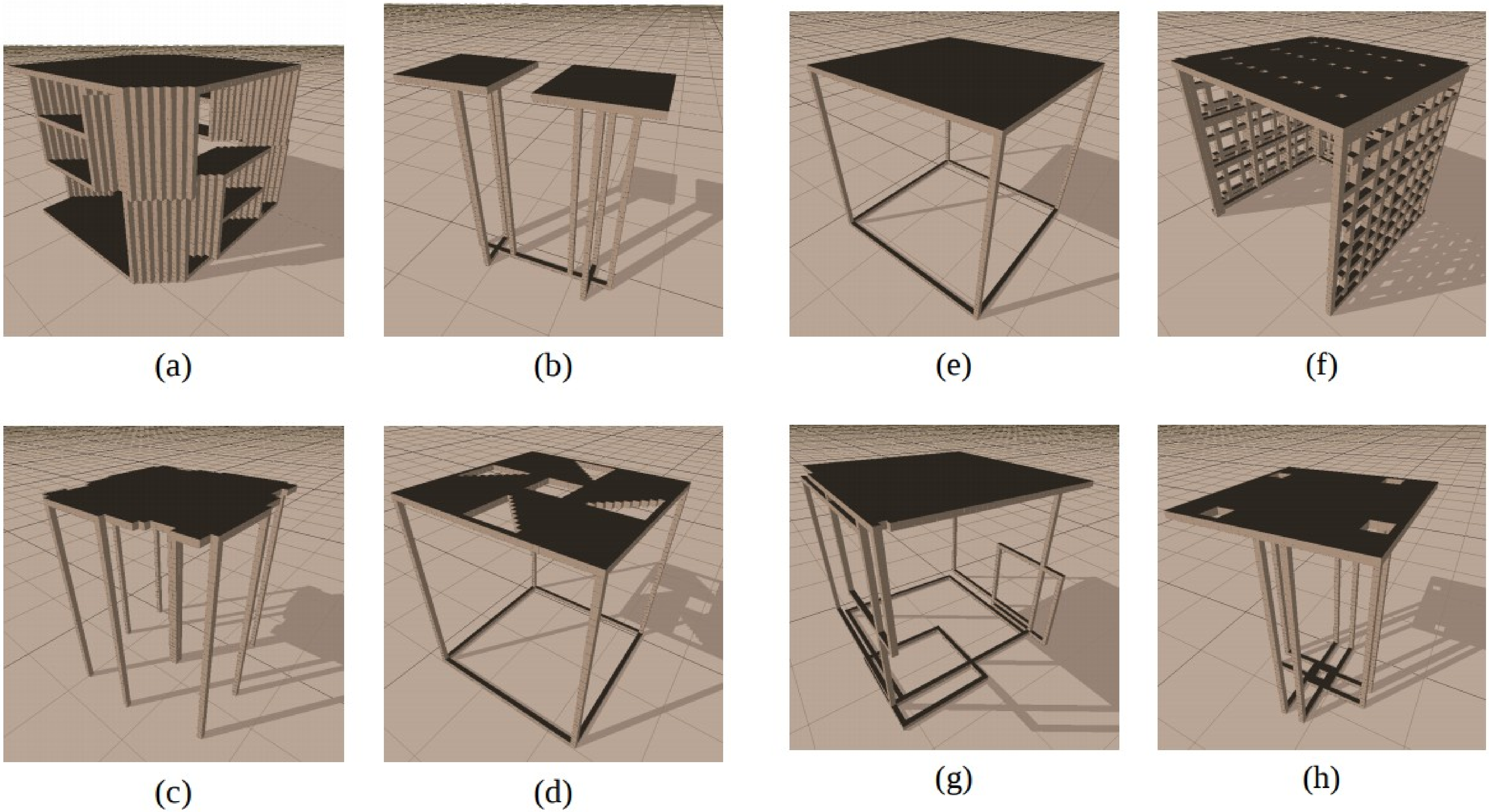
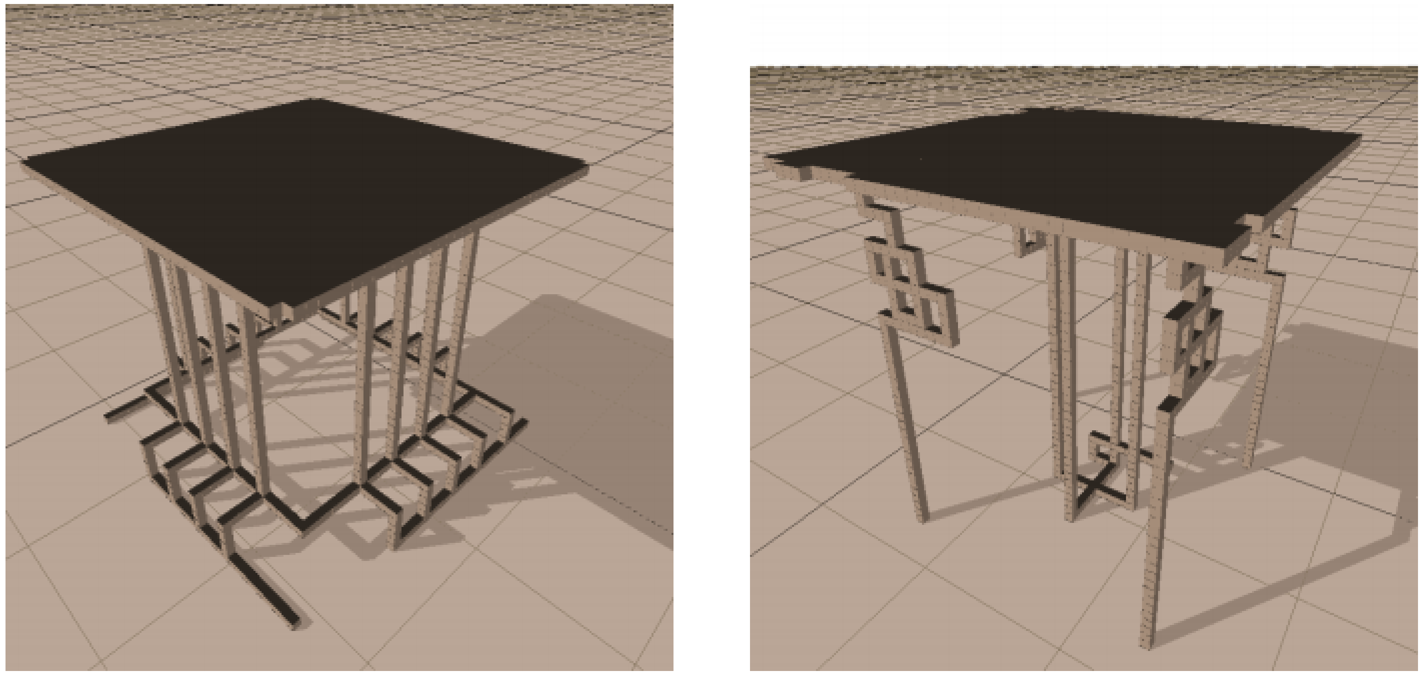


Figure 4: Tables evolved using the P0L-system as a generative encoding.



(a) (b)

Figure 5: Tables evolved using: a, a non-L-system, generative encoding with block replication; and b, a P0L-system encoding without block replication.

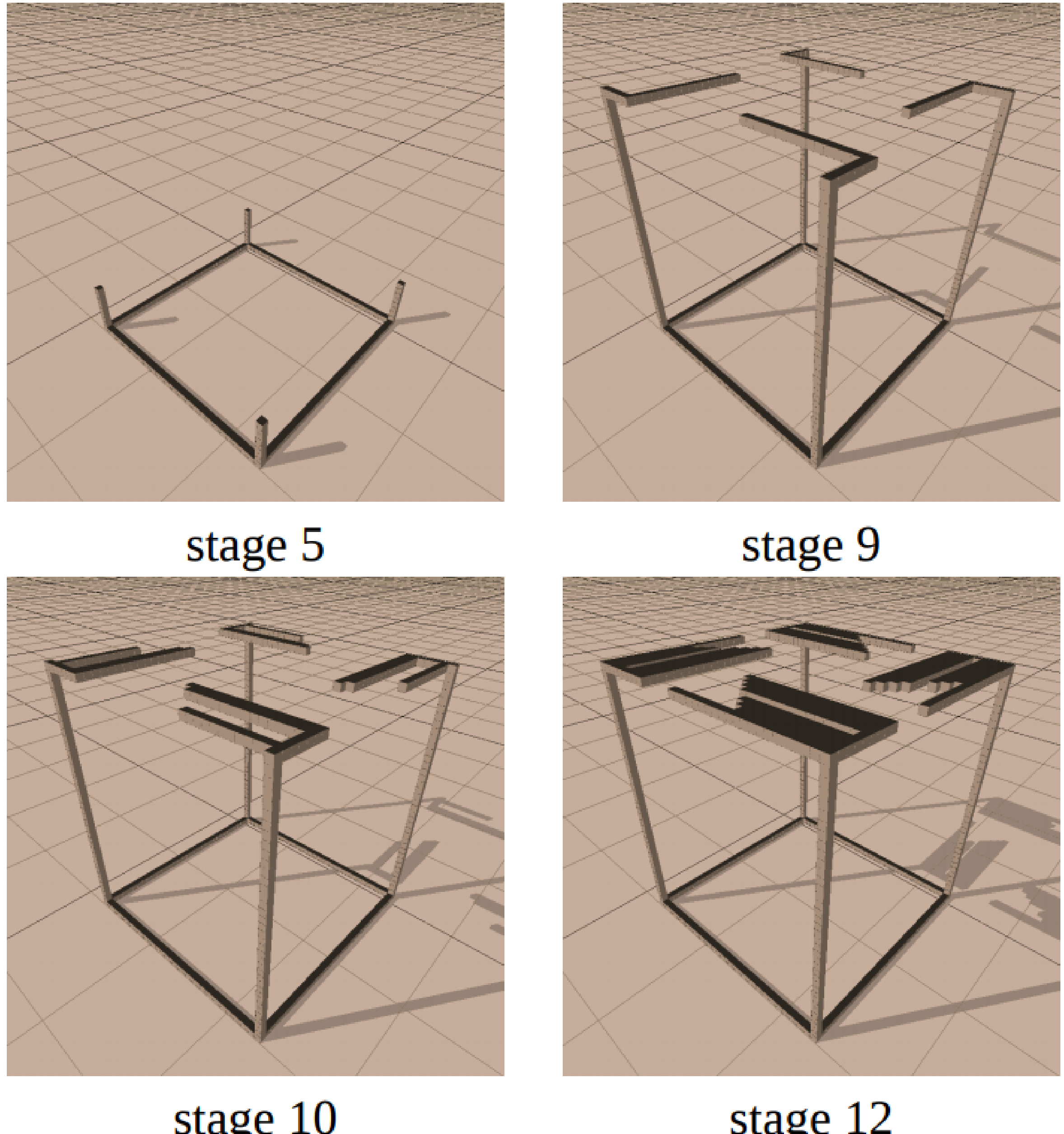


Figure 7: Growth of a table.

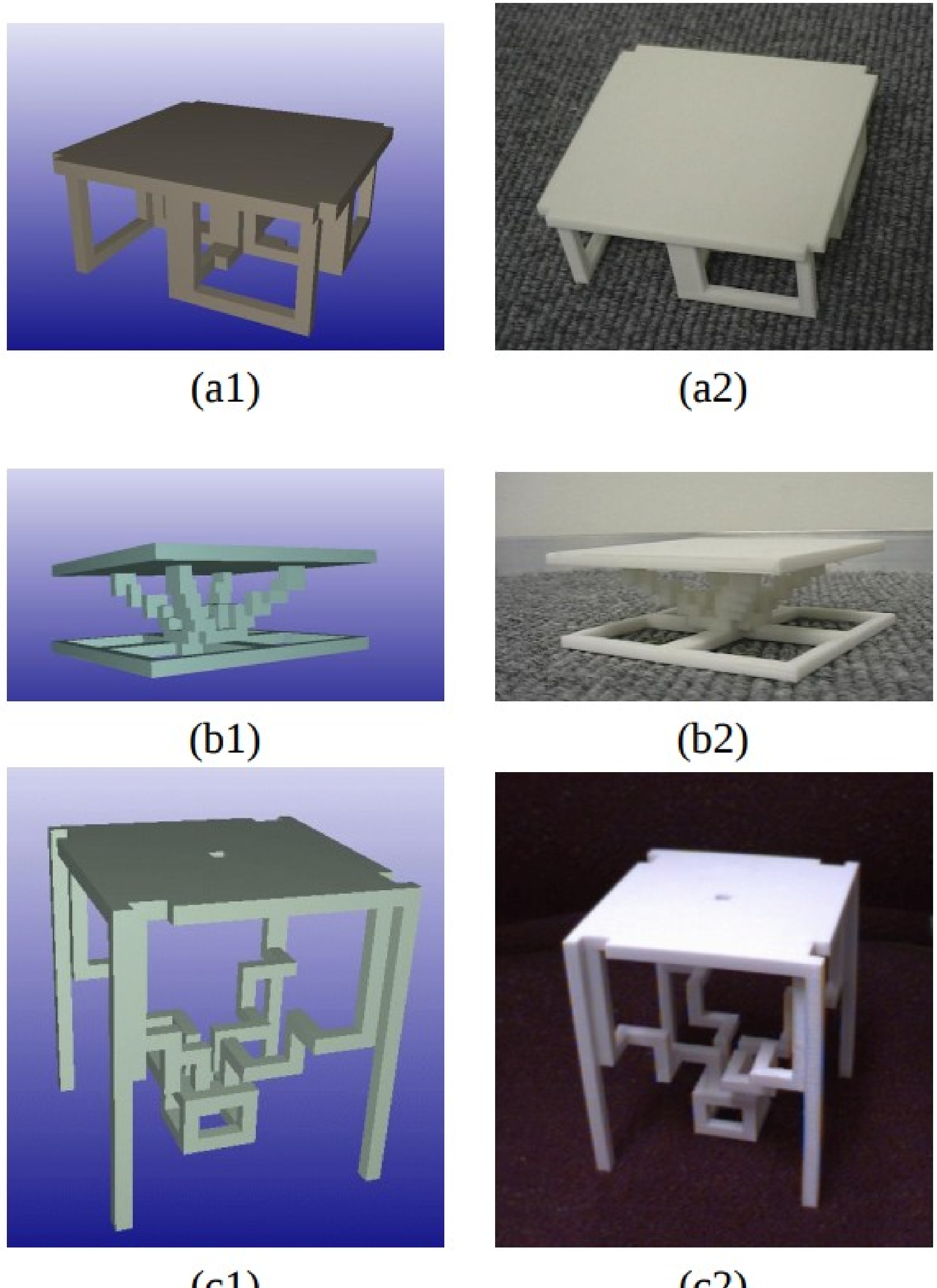


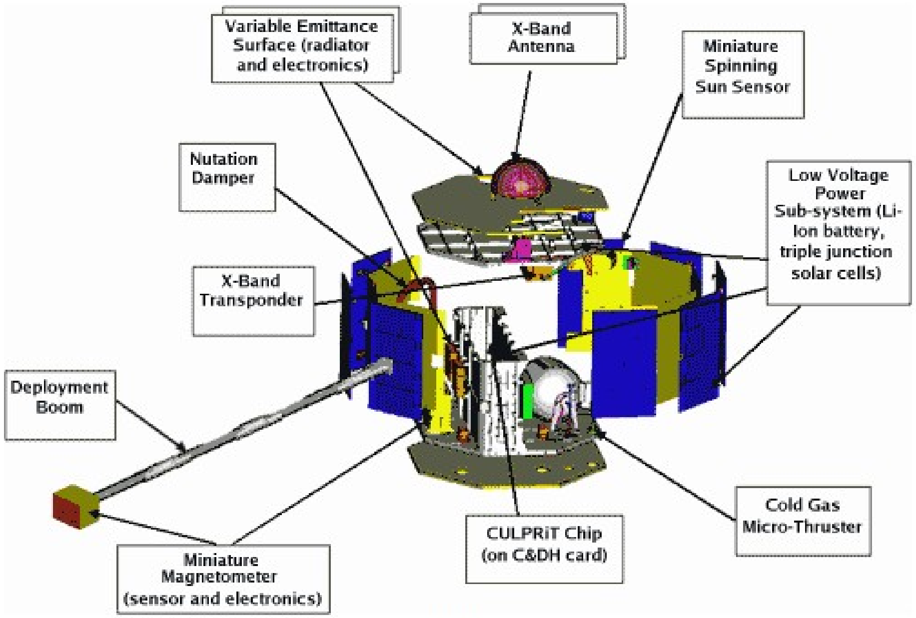
Figure 6: Manufacture tables shown both in simulation (left) and reality (right).

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

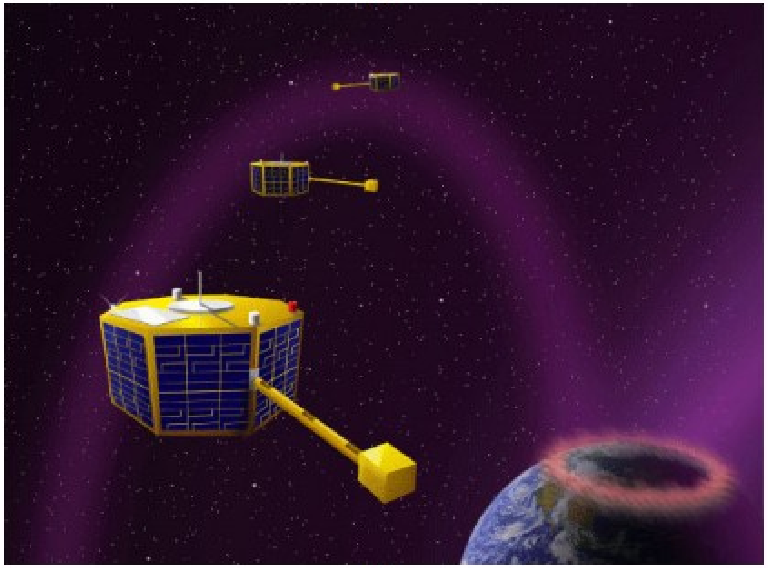
Gregory. S. Hornby Gregory.S.Hornby@nasa.gov
Mail Stop 269-3, University Affiliated Research Center, UC Santa Cruz, Moffett Field, CA, 94035, USA

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

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

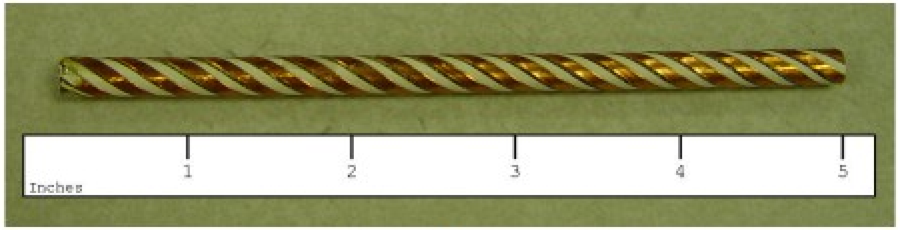


(a)

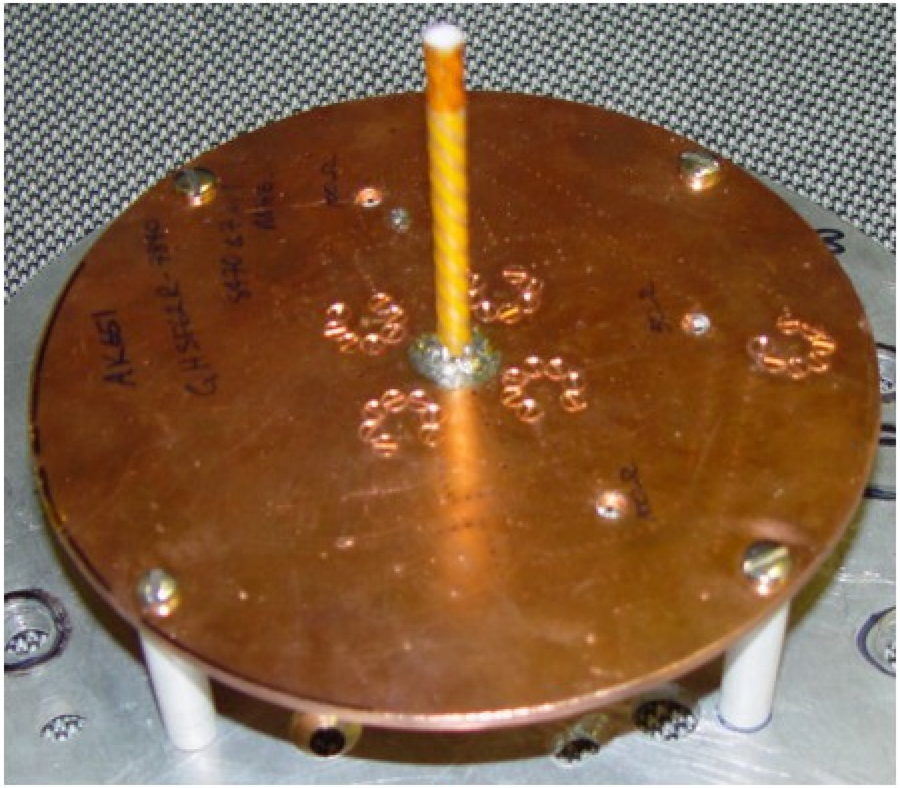


(b)

Figure 1: Artist's depiction of: (a) the spacecraft model showing the different spacecraft components, and (b) the ST5 mission with the three spacecraft in their string of pearls orbit.

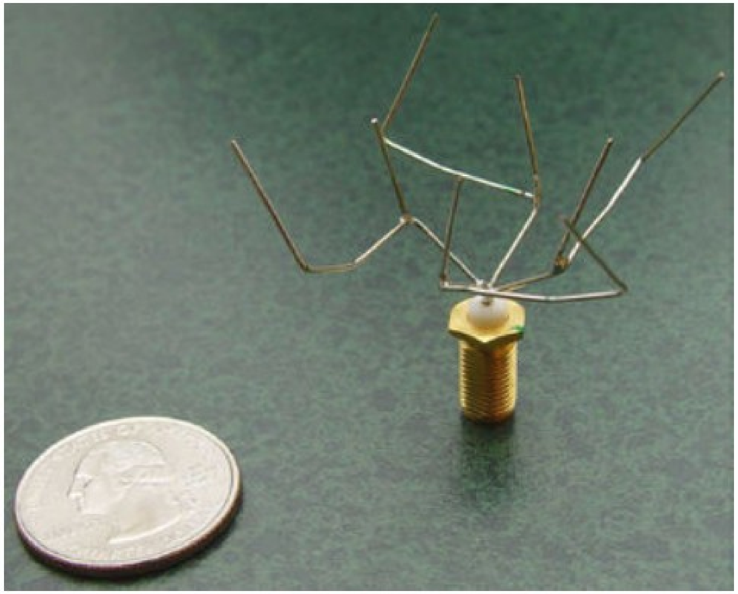


(a)

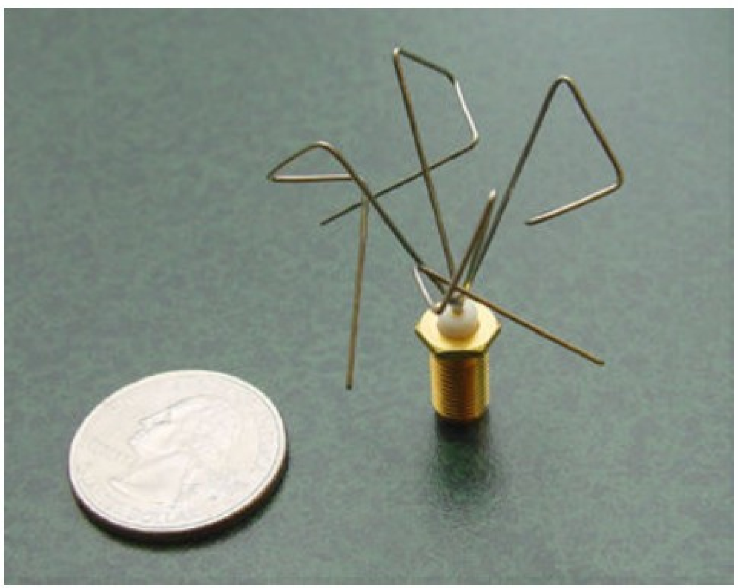


(b)

Figure 2: Conventionally-designed quadrifilar helical antenna: (a) radiator; and (b) radiator mounted on a ground plane.

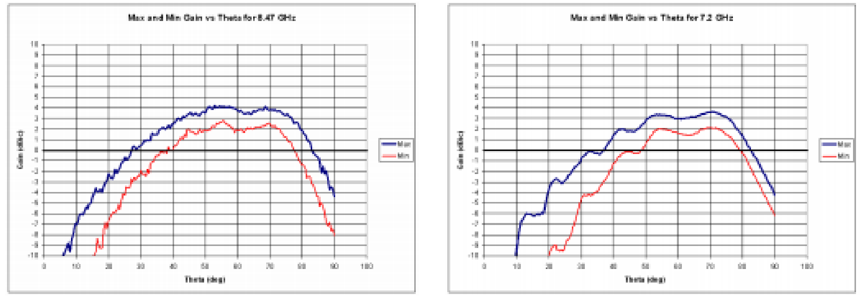


(a)



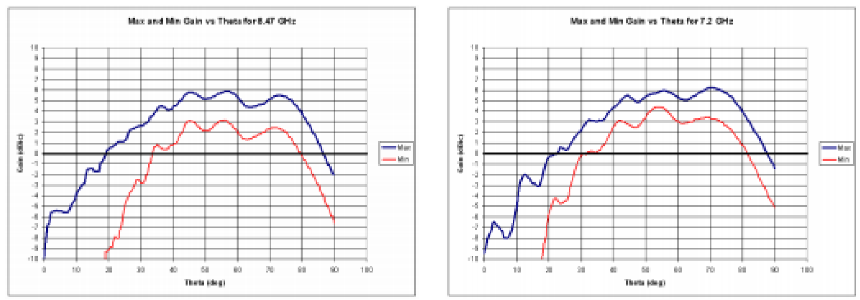
(b)

Figure 5: Photographs of prototype evolved antennas: (a) ST5-3-10; (b) ST5-4W-03



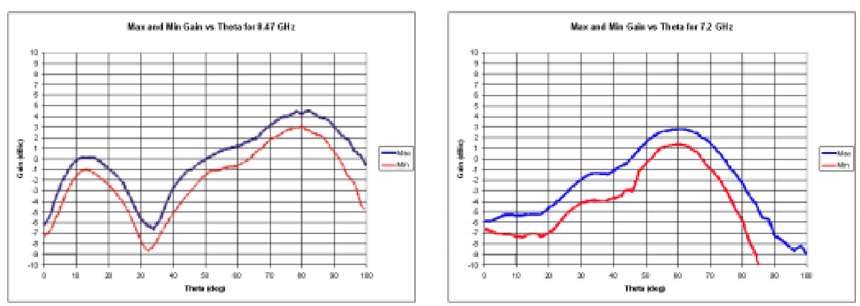
(a) (b)

Figure 6: Maximum and minimum gain for antenna ST5-4W-03, as measured in an anechoic test chamber at NASA Goddard Space Flight Center, at: (a) 8.47 GHz; and (b) 7.2GHz. This antenna was evolved with the parameterized EA.



(a) (b)

Figure 7: Maximum and minimum gain for antenna ST5-3-10, as measured in an anechoic test chamber at NASA Goddard Space Flight Center, at: (a) 8.47 GHz; and (b) 7.2GHz. This antenna was evolved with the open-ended EA which allowed branching.

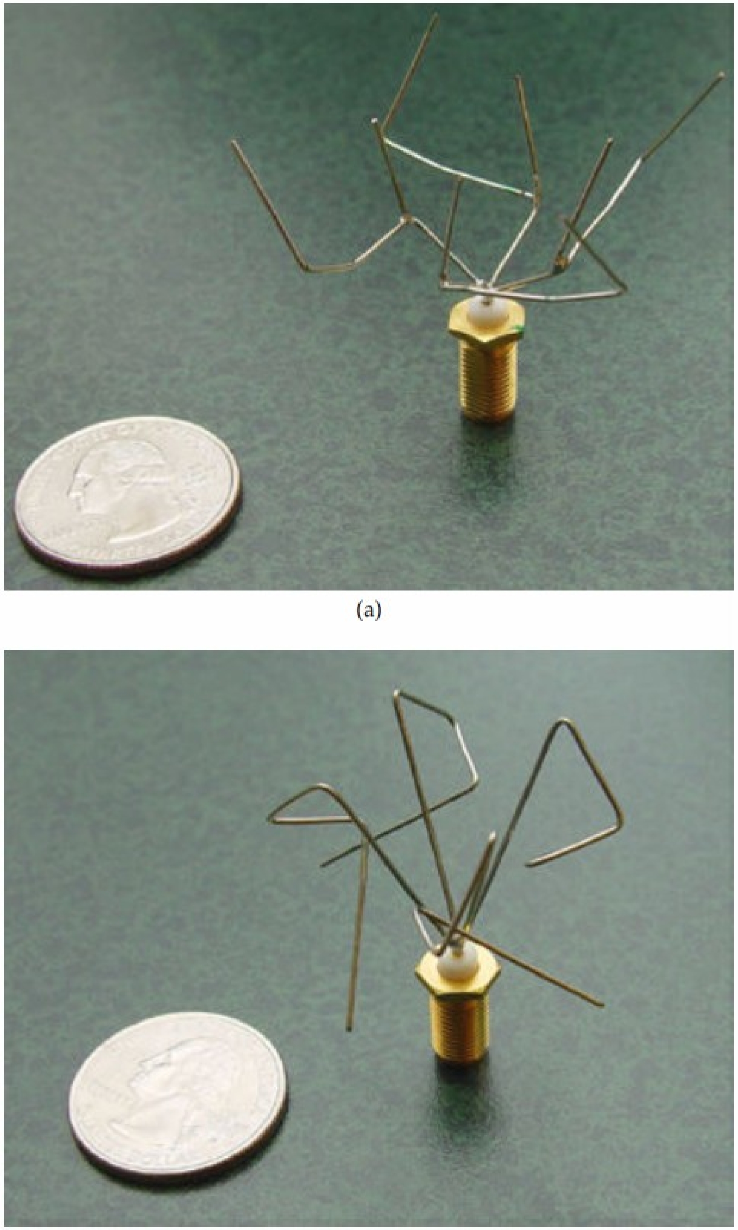


(a) (b)

Figure 8: Maximum and minimum gain for the traditionally designed QHA at: (a) 8.47 GHz; and (b) 7.2GHz.

5.1 Revised Design Space

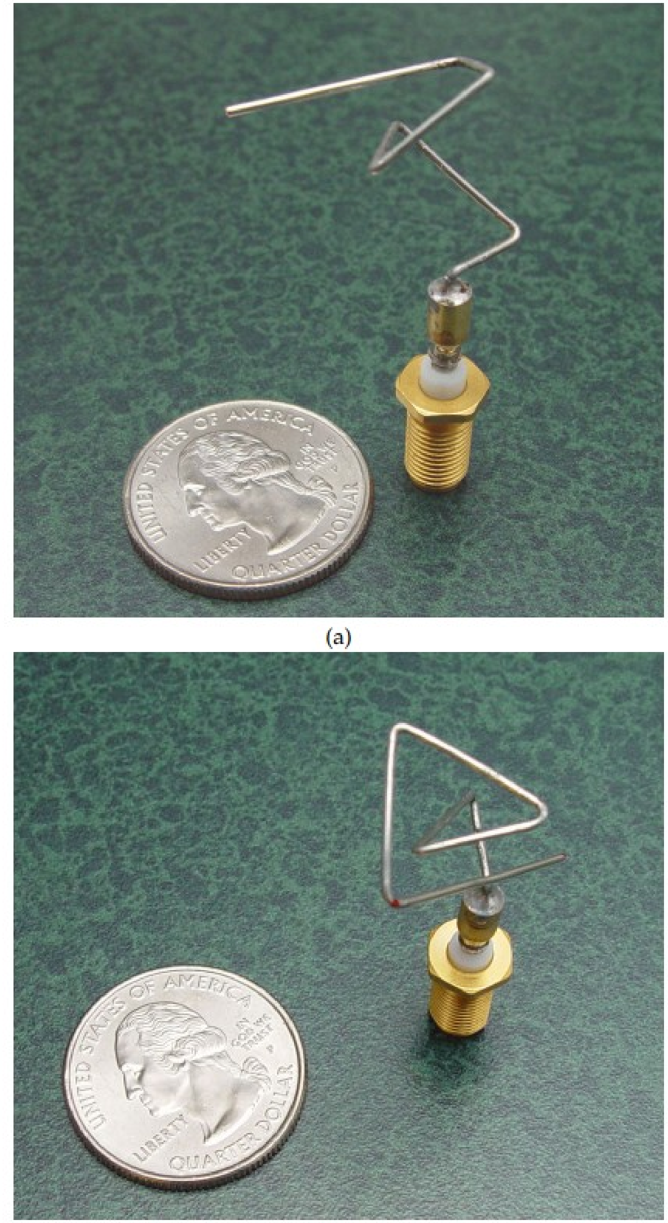
As a result of the new mission requirements, we needed to modify both the type of antenna being evolved and the fitness function (Lohn et al., 2005). The original antennas evolved for the ST5 mission were constrained to monopole wire antennas with four identical arms, with each arm rotated 90° from its neighbors. With these antennas the EA evolved genotypes that specified the design for one arm and the phenotype consisted of four copies of the evolved arm. Because of symmetry, this four-arm design has a null at zenith that is built into the design and is unacceptable for the revised mission. To achieve an antenna that meets the new mission requirements, we decided to search the space of single-arm antennas. In addition, because of our concerns in meeting space-qualification standards in the joints of a branching antenna, we constrained our antenna designs to non-branching ones. Producing a single-arm antenna to meet the mission requirements is a very challenging problem since the satellite is spinning at roughly 40 RPM and it is important that the antennas have uniform gain patterns in the azimuth. This criteria is difficult to meet with a single-arm antenna, because it is inherently asymmetric. In the remainder of this section we describe how we modified our two evolutionary algorithms to address these new requirements.



(a)

(b)

Figure 5: Photographs of prototype evolved antennas: (a) ST5-3-10; (b) ST5-4W-03



(a)

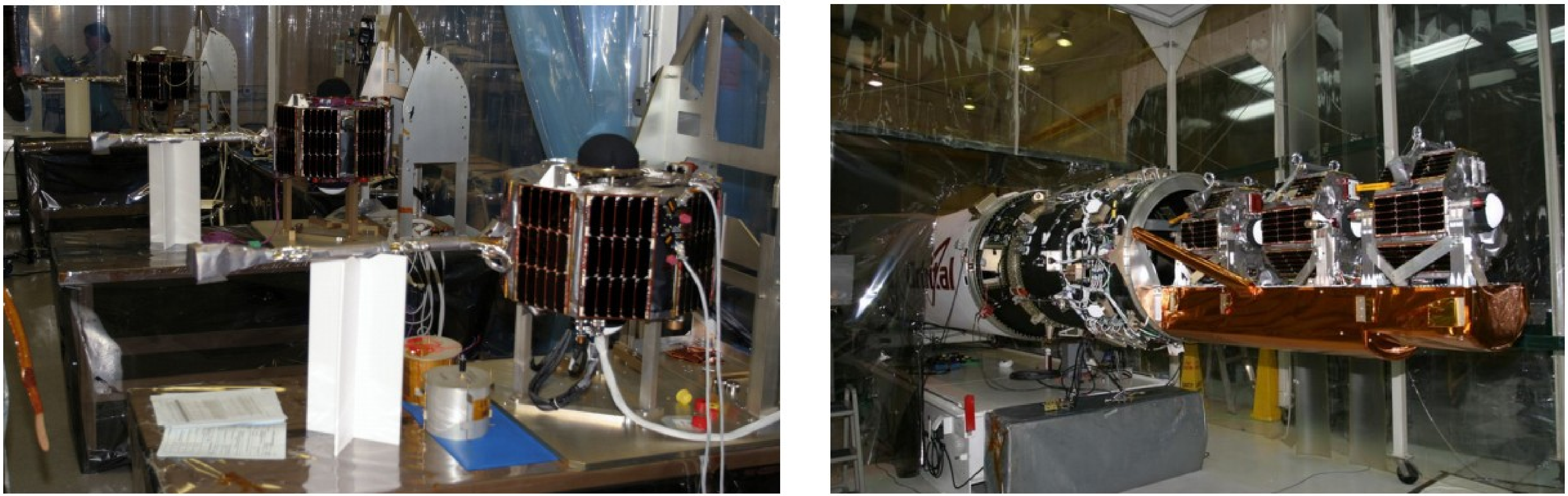
(b)

Figure 9: Evolved antenna designs: (a) evolved using a constructive process, named ST5-33.142.7; and (b) evolved using a vector of parameters, named ST5-104.33.

7 First Computer-Evolved Hardware in Space

(a) (b)

Figure 12: Images of a completed, flight antenna: (a) a flight unit after it has been coated; and (c), the underside of a flight unit.



(a) (b)

Figure 13: Images of the ST5 spacecraft: (a) the three ST5 spacecraft with the black radomes on top containing an evolved antenna, ST5-33.142.7; and (b) the three ST5 Spacecraft mounted for launch on a Pegasus XL rocket.

In addition to being the first evolved hardware in space, the evolved antennas demonstrate several advantages over the conventionally designed antenna and over manual design in general. The evolutionary algorithms used were not limited to variations of previously developed antenna shapes but generated and tested thousands of completely new types of designs, many of which have unusual structures that expert antenna designers would not be likely to produce. By exploring such a wide range of designs EAs may be able to produce designs of previously unachievable performance. For example, the best antennas that were evolved achieve high gain across a wider range of elevation angles, which allows a broader range of angles over which maximum data throughput can be achieved and may require less power from the solar array and batteries. In addition, antenna ST5-33.142.7 has a very uniform pattern with small ripples in the elevations of greatest interest (40° to 80°) which allows for reliable performance as elevation angle relative to the ground changes. With the evolutionary design approach it took approximately 3 person-months of work to generate the initial evolved antennas versus 5 person-months for the conventionally designed antenna and, when the mission orbit changed, with the evolutionary approach we were able to modify our algorithms and re-evolve new antennas specifically designed for the new orbit and prototype hardware in 4 weeks. The faster design cycles of an evolutionary approach results in less development costs and allows for an iterative "what-if" design and test approach for different scenarios. This ability to rapidly respond to changing requirements is of great use to NASA since NASA mission requirements frequently change. As computer hardware becomes increasingly more powerful and as computer modeling packages become better at simulating different design domains we expect evolutionary design systems to become more useful in a wider range of design problems and gain wider acceptance and industrial usage.