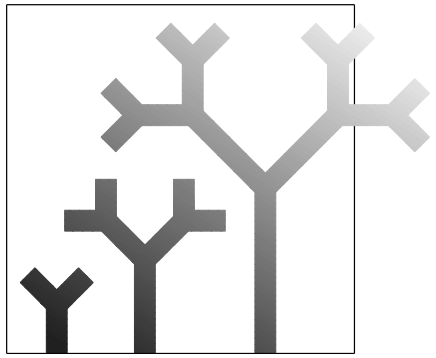
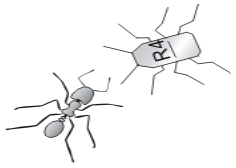
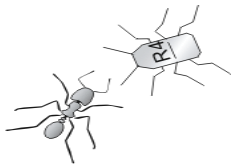
Morphology Representations





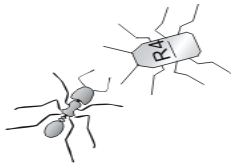
What you will learn in this class

- Direct encoding of body parameters
  - aircraft morphologies
- Rewriting representations
  - fractals
  - plants
  - neural architectures



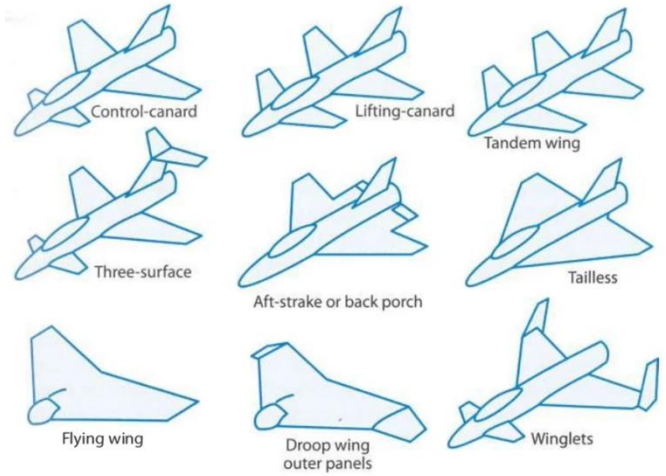
Direct Encoding of Body Parameters

- Start from a body template
- Define parameters of body parts
- Use direct encoding: each parameter is directly encoded in the genotype (i.e., without using a growth process)

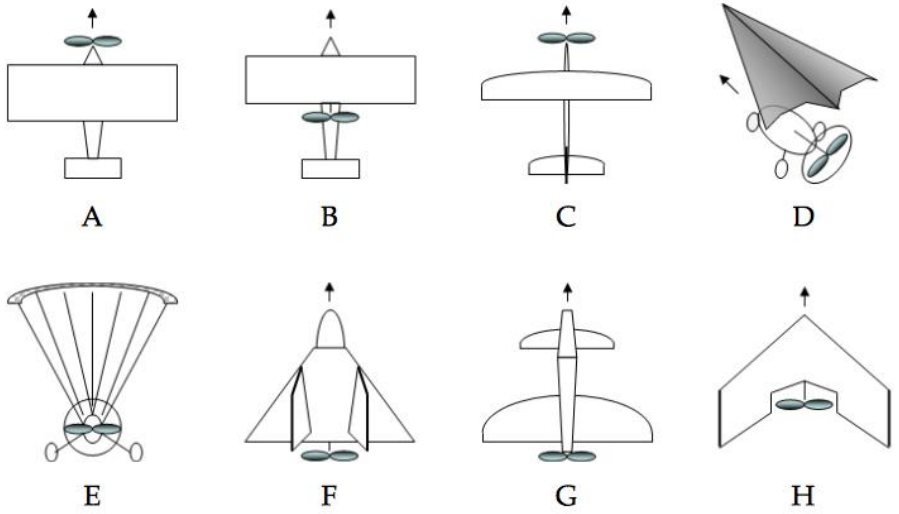


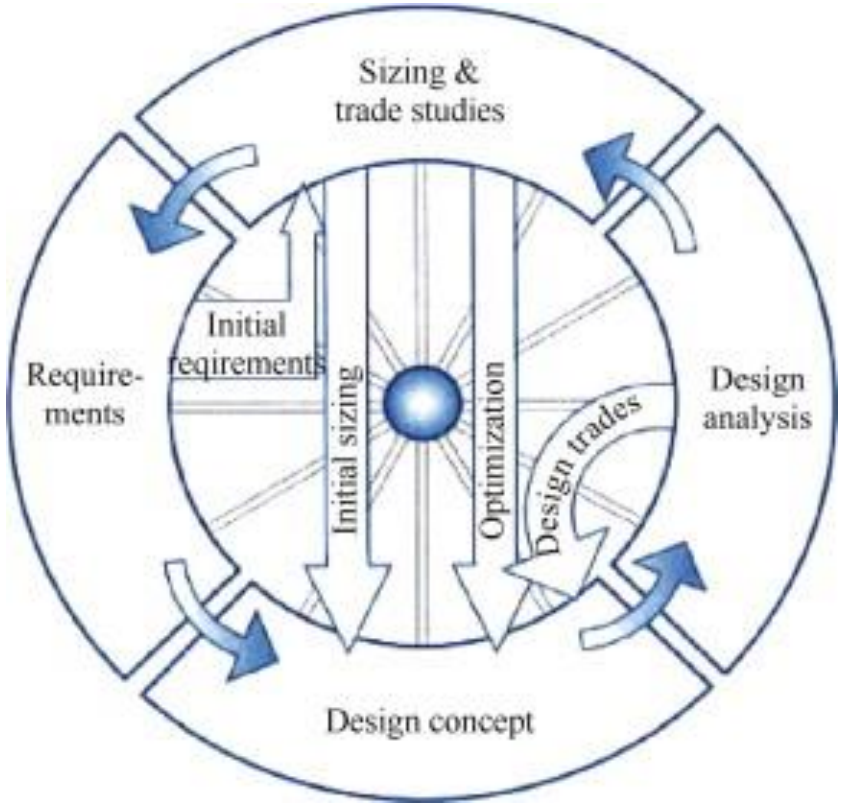
The “art” of aircraft design

Jets: High speed, high payload aircraft



Propellers: Low speed, low payload aircraft and drones





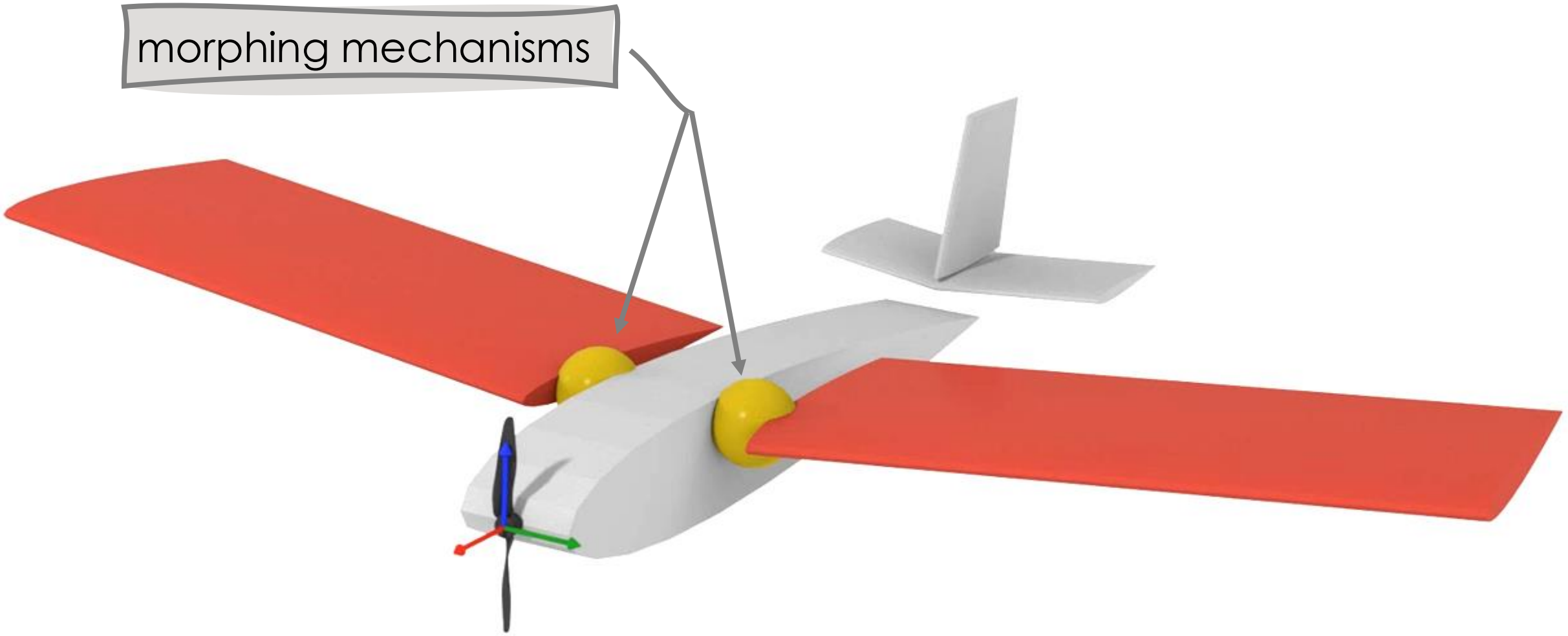
Designing an aircraft, big or small, is an iterative process (Raymer, 2015)

Each design is a compromise of requirements, efficiency, and control

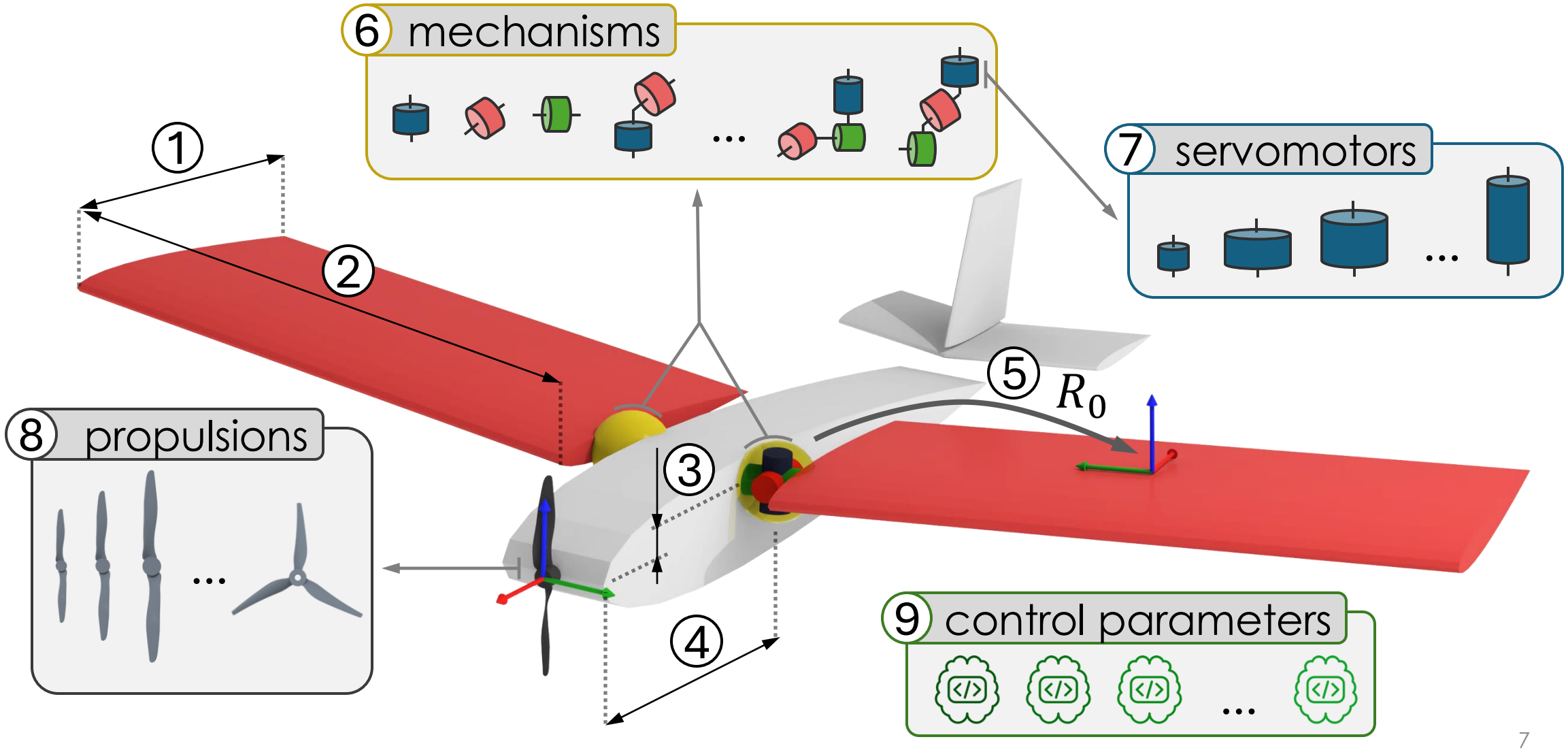
Cruising flight

E. Ajanic et al., “Bioinspired wing and tail morphing extends drone flight capabilities,” Science Robotics, 2020.

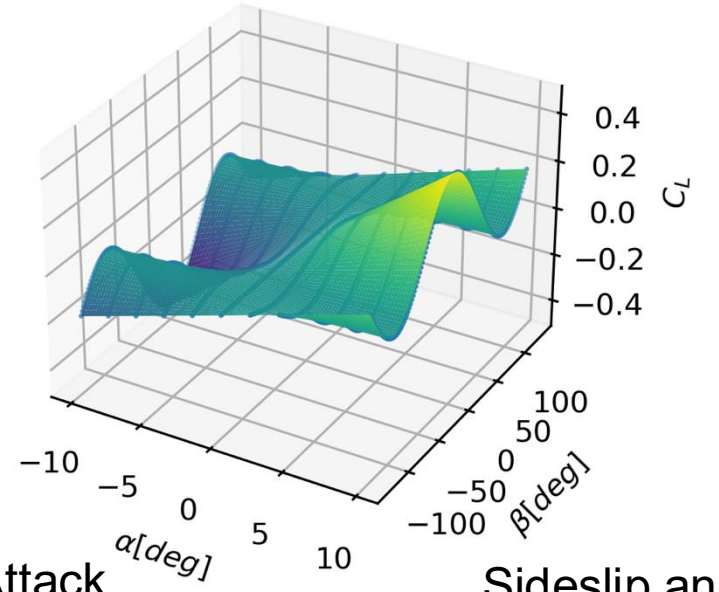
Morphological template



Genetically-encoded Body Parameters



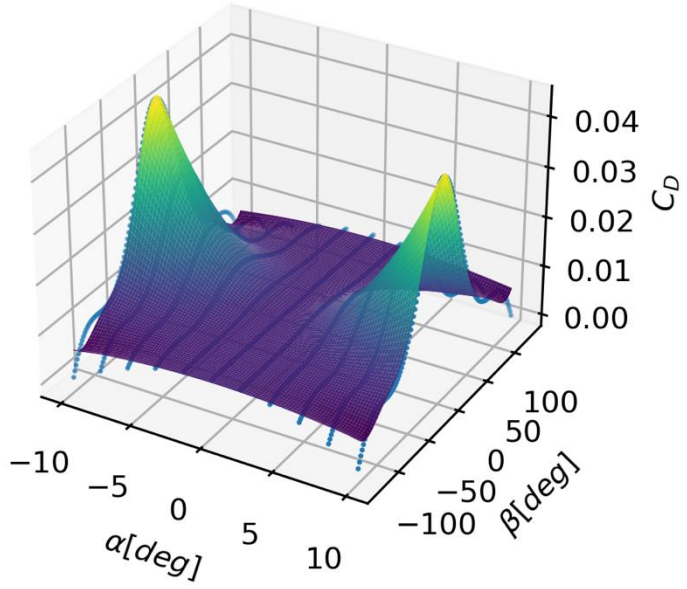
Aerodynamic Model



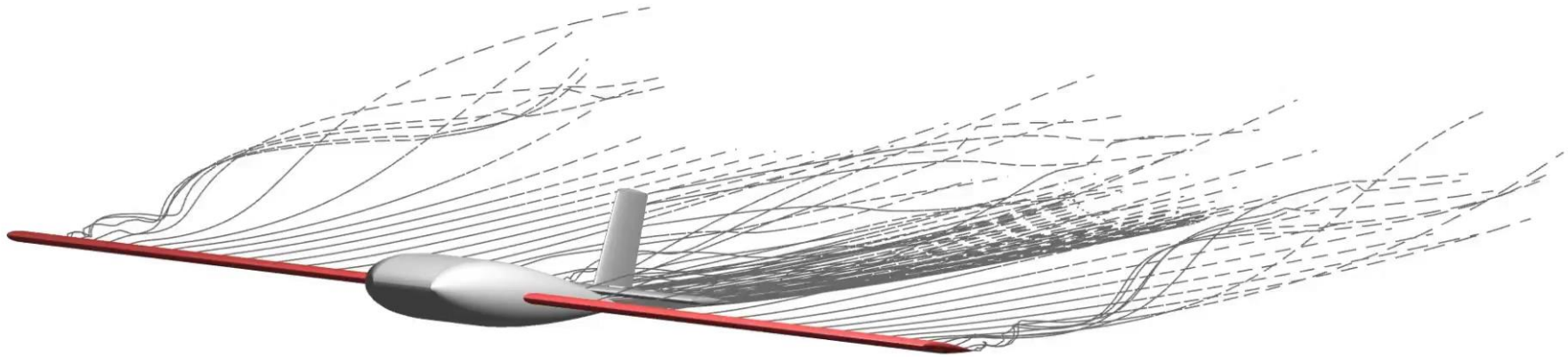
Lift

Angle of Attack

Sideslip angle

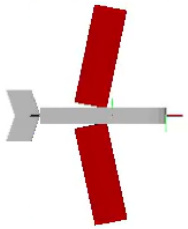


Drag

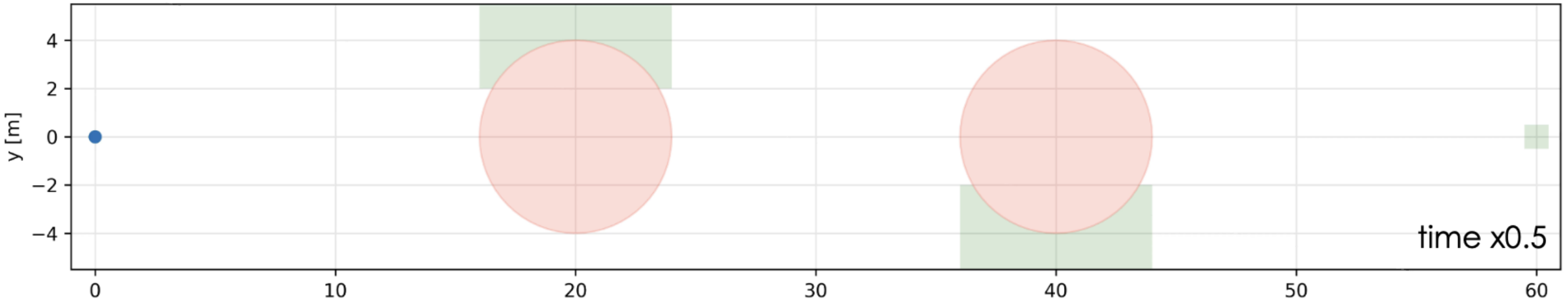


Trajectory Optimisation in Simulation

Identify time-varying control inputs for optimal trajectory with constraints



Mission definition: Collision-Free Flight



Minimise:

Energy Consumption

Flight Time

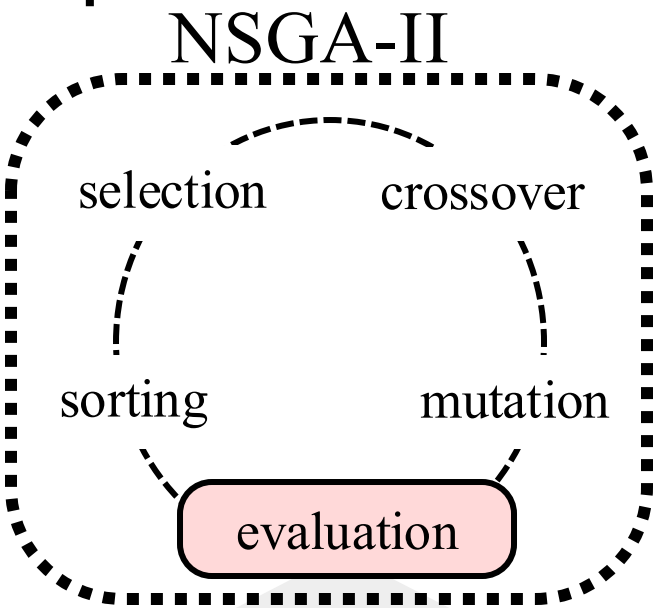
Constraints:

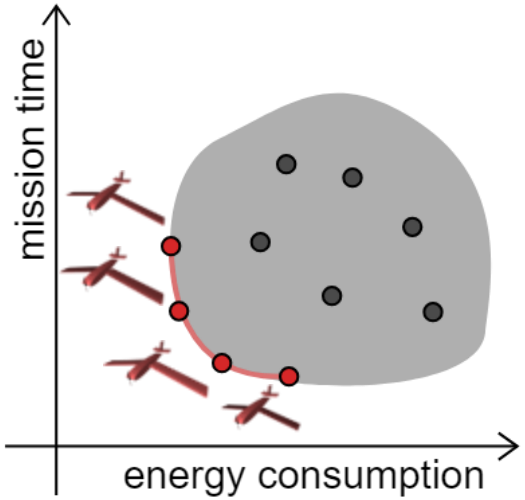
1) start at blue location;
2) pass through green sides
3) arrive to green target, stay within mechanical limits of aircraft

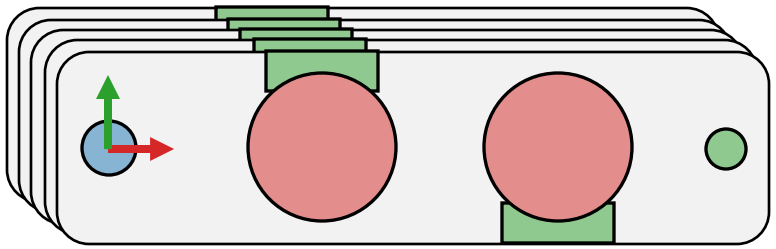
Evolutionary Loop

Mission scenarios

NSGA-II







fitness functions:

$$\mathcal{L}_1 = \sum_{i=1}^{n_s} \frac{(\text{energy})_i}{n_s}$$

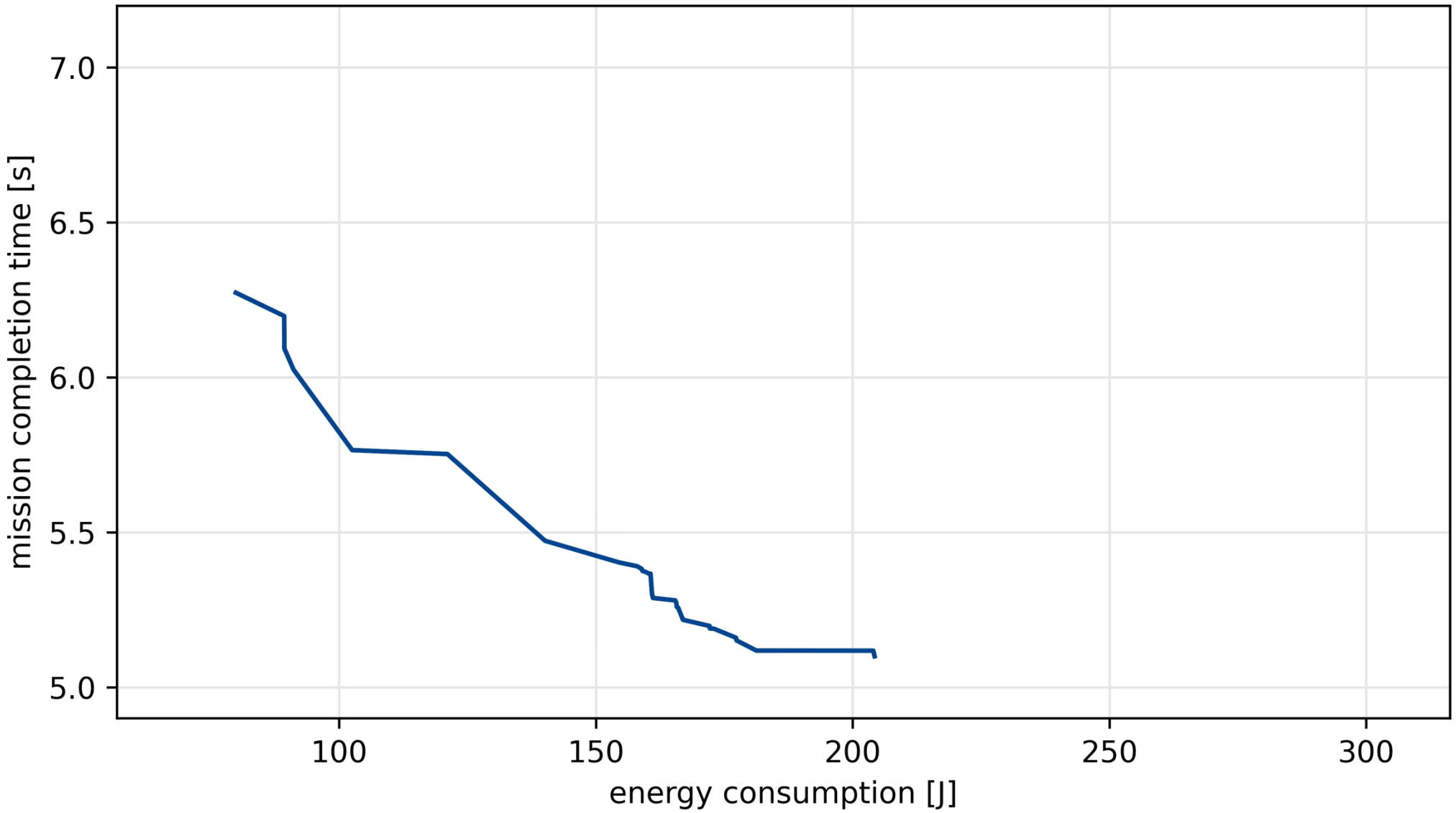
$$\mathcal{L}_2 = \sum_{i=1}^{n_s} \frac{(\text{time})_i}{n_s}$$

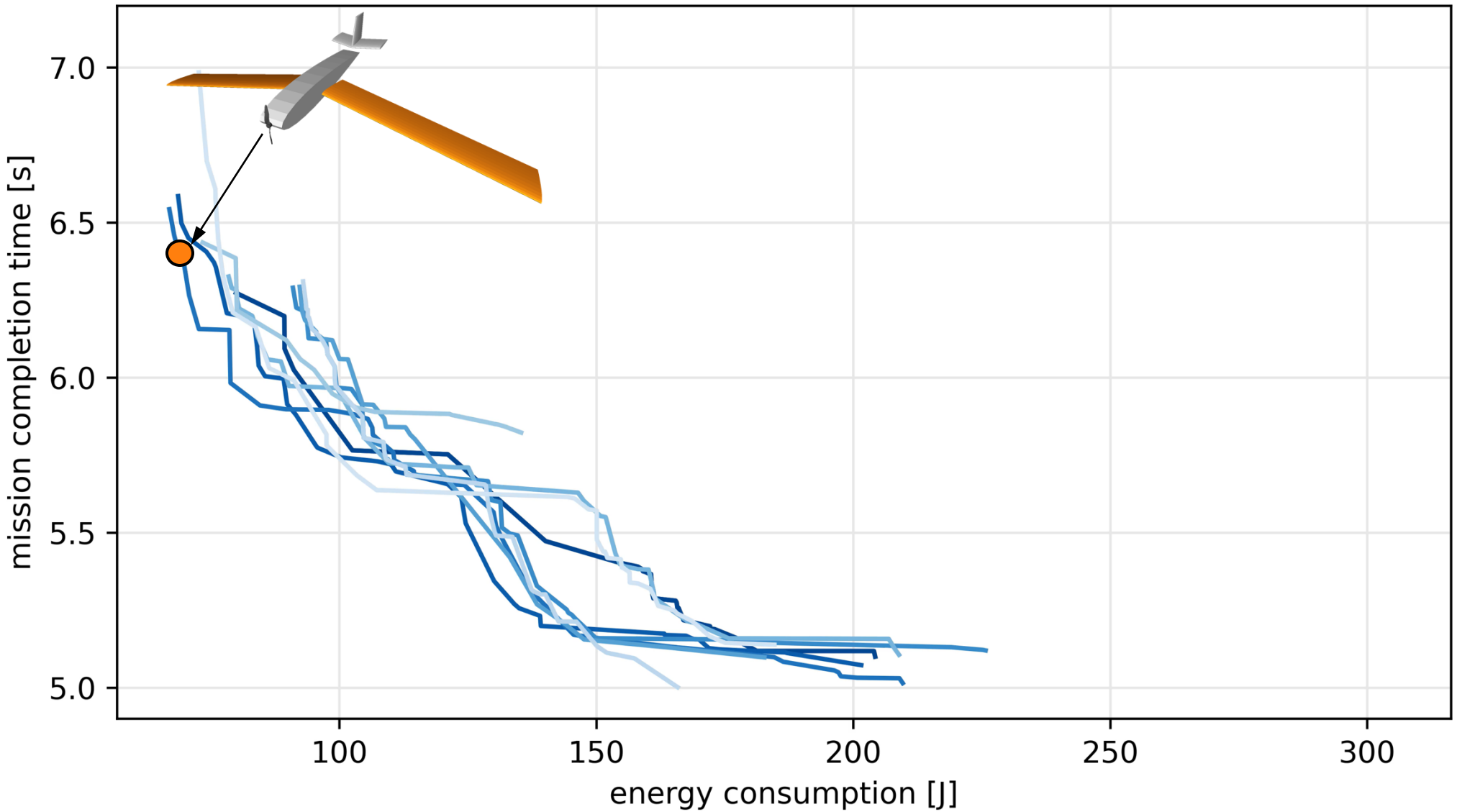
scenario 1 ⇨ modelling ⇨ traj.opt.

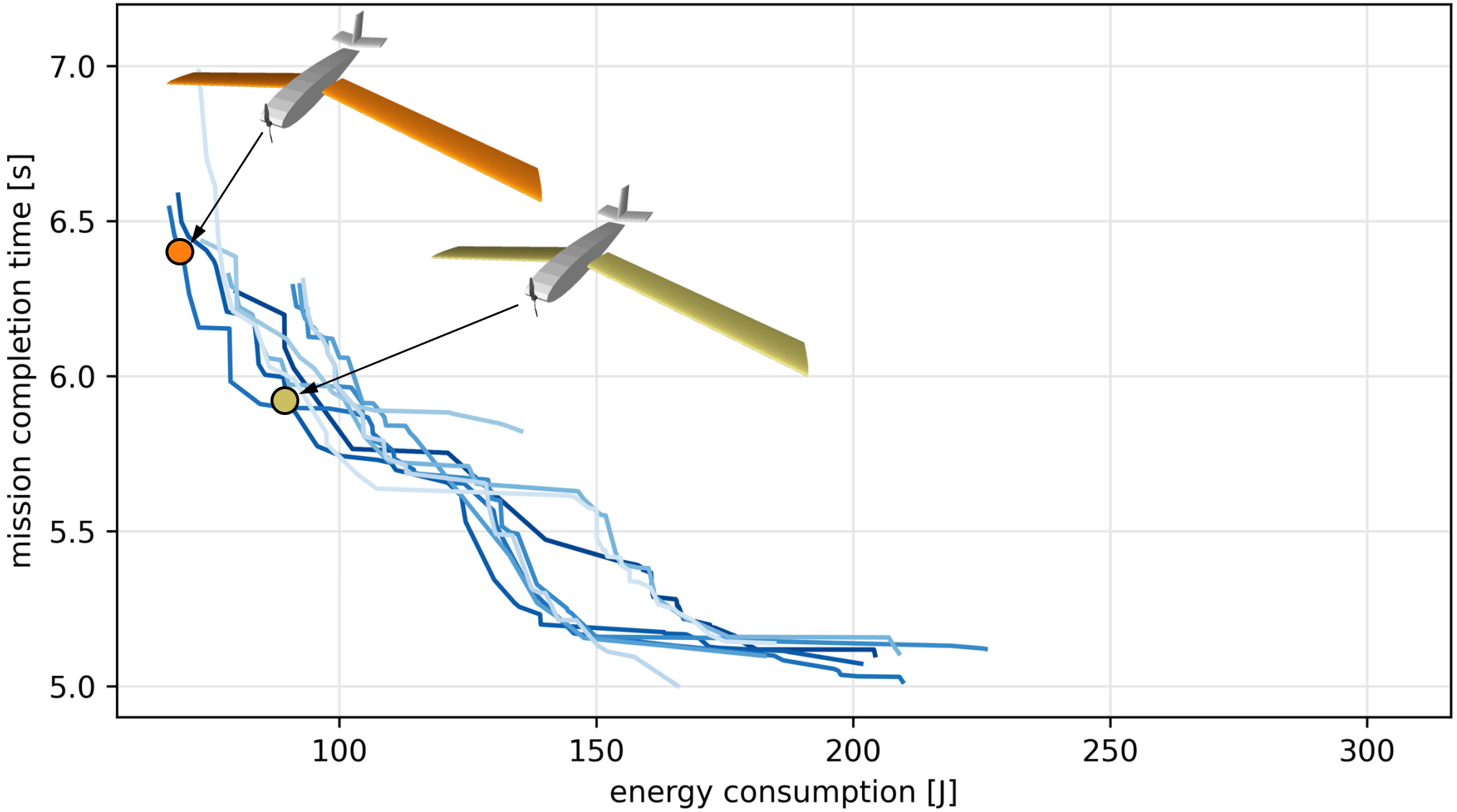
⋮ ⋮ ⋮

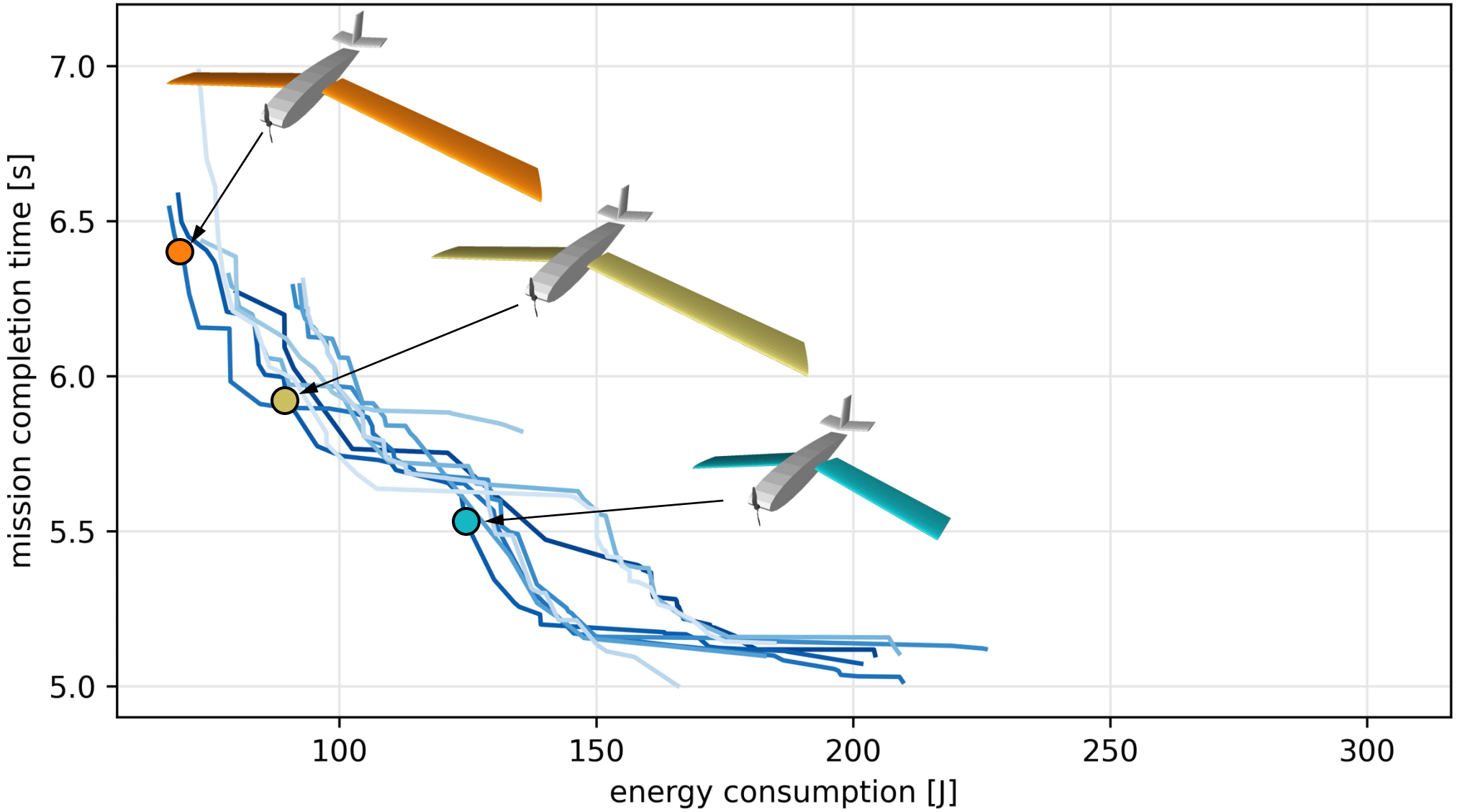
scenario n_s ⇨ modelling ⇨ traj.opt.

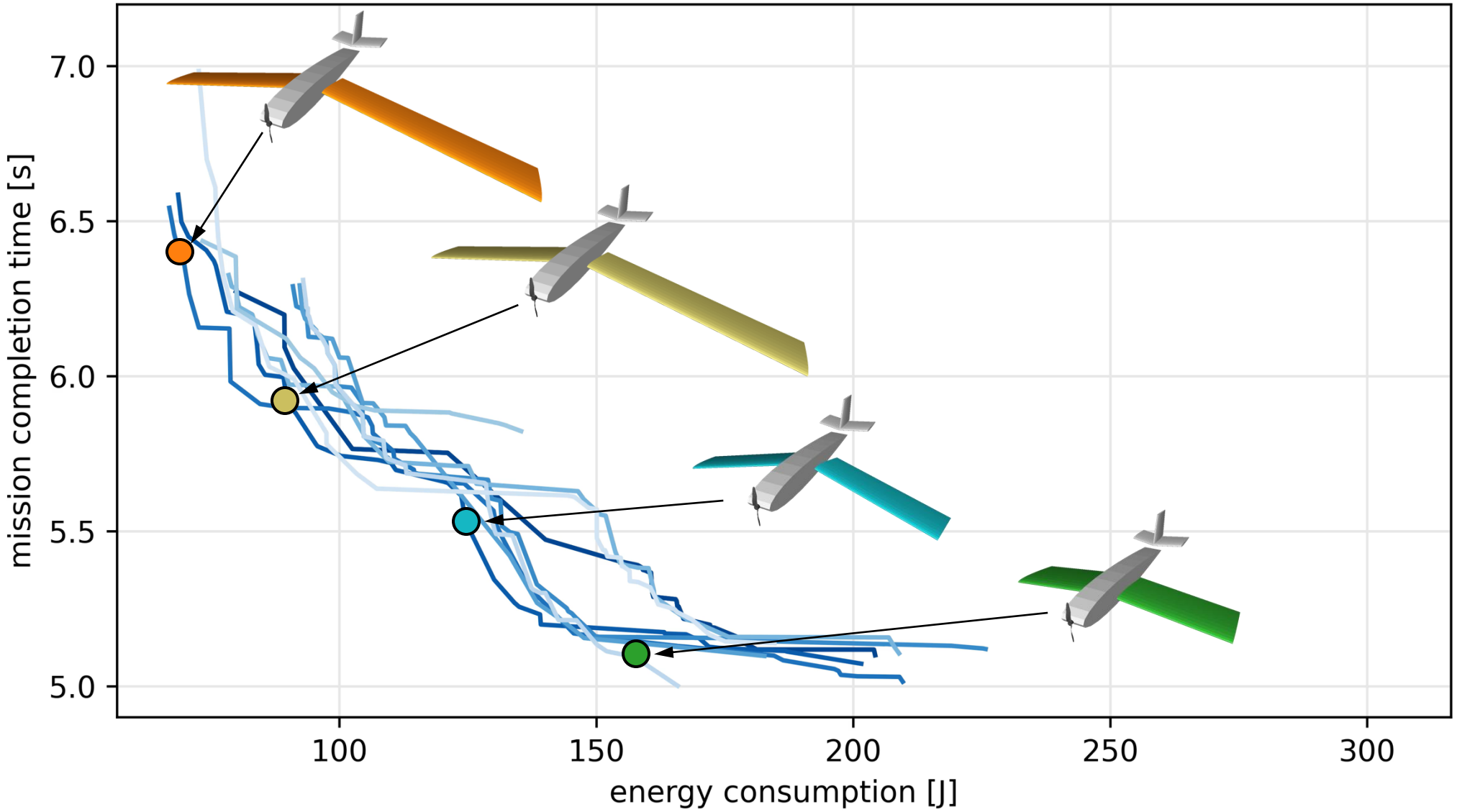
Results from 9 runs of NSGA-II



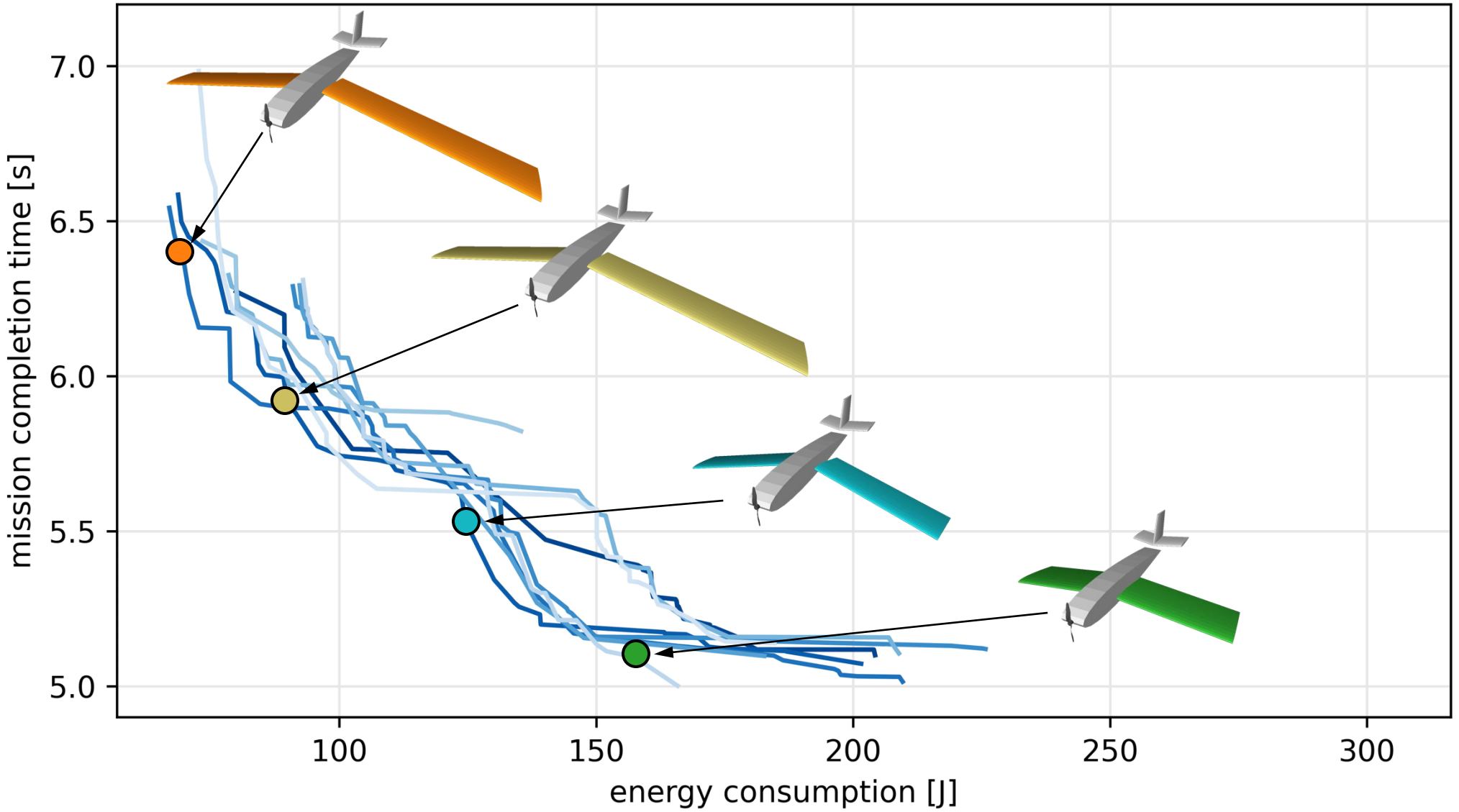

7.0
6.5
6.0
5.5
5.0
mission completion time [s]
100
150
200
250
300
energy consumption [J]


7.0
6.5
6.0
5.5
5.0
mission completion time [s]
100
150
200
250
300
energy consumption [J]

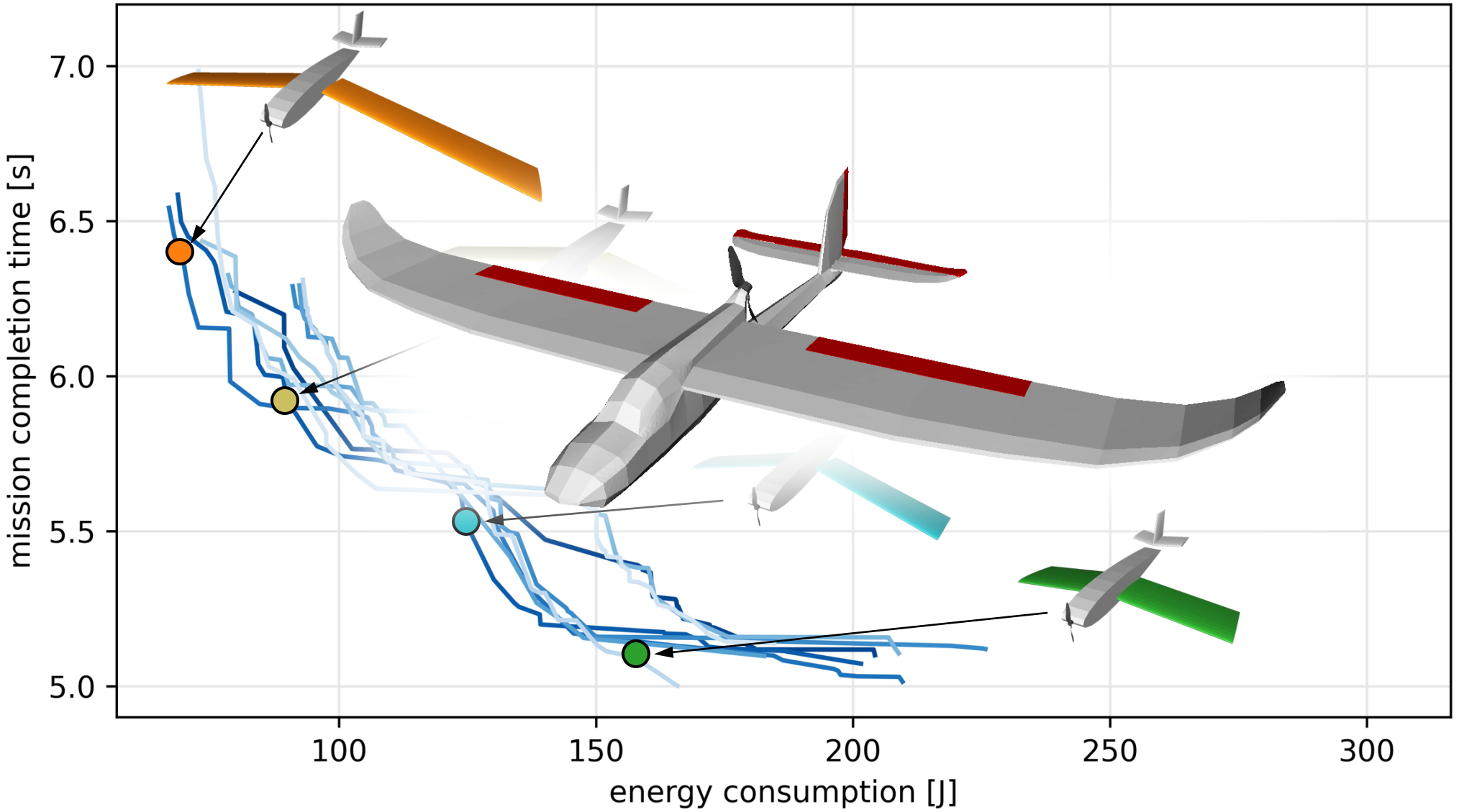

7.0
6.5
6.0
5.5
5.0
mission completion time [s]
100
150
200
250
300
energy consumption [J]


7.0
6.5
6.0
5.5
5.0
mission completion time [s]
100
150
200
250
300
energy consumption [J]

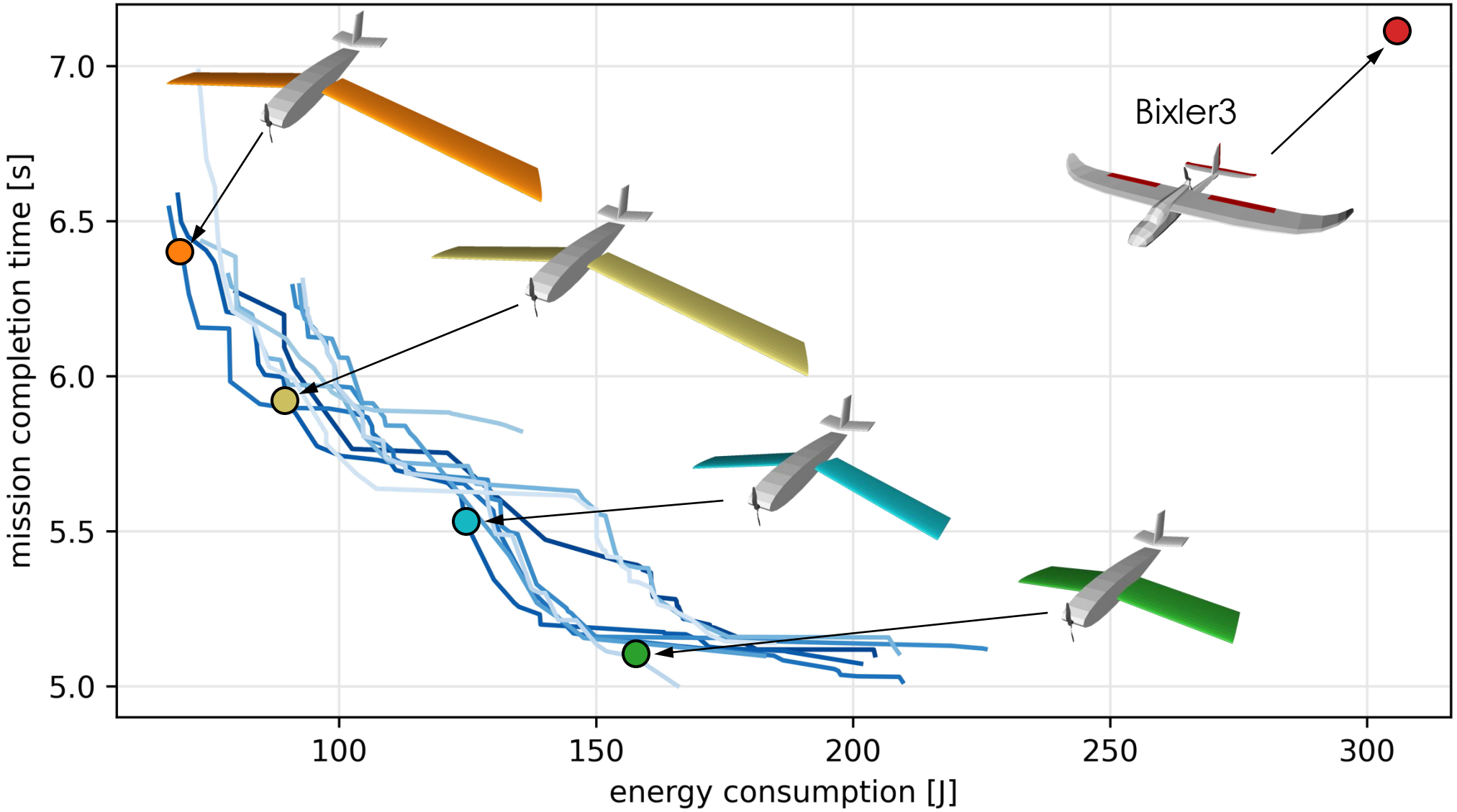
comparing results with a commercial fixed-winged (Bixler3)



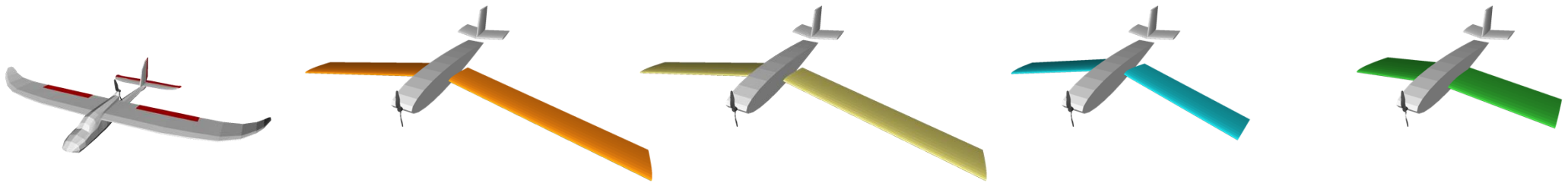
comparing results with a commercial fixed-winged (Bixler3)



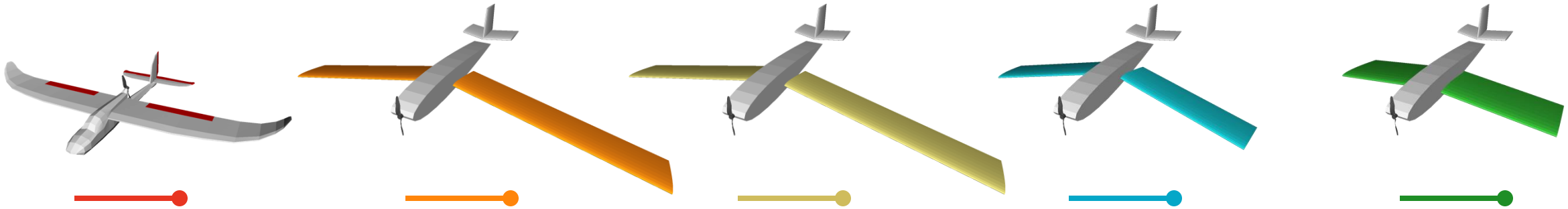
Comparing results with a commercial fixed-winged (Bixler3)

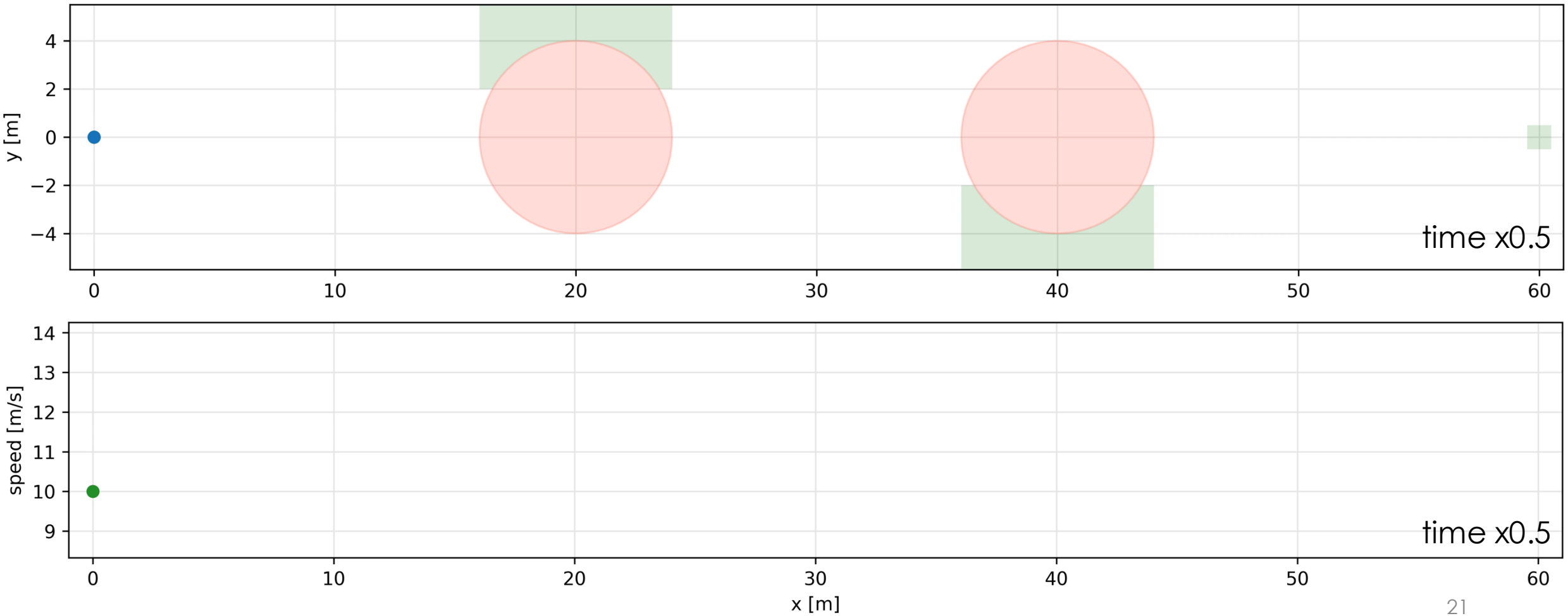


Technical Specifications



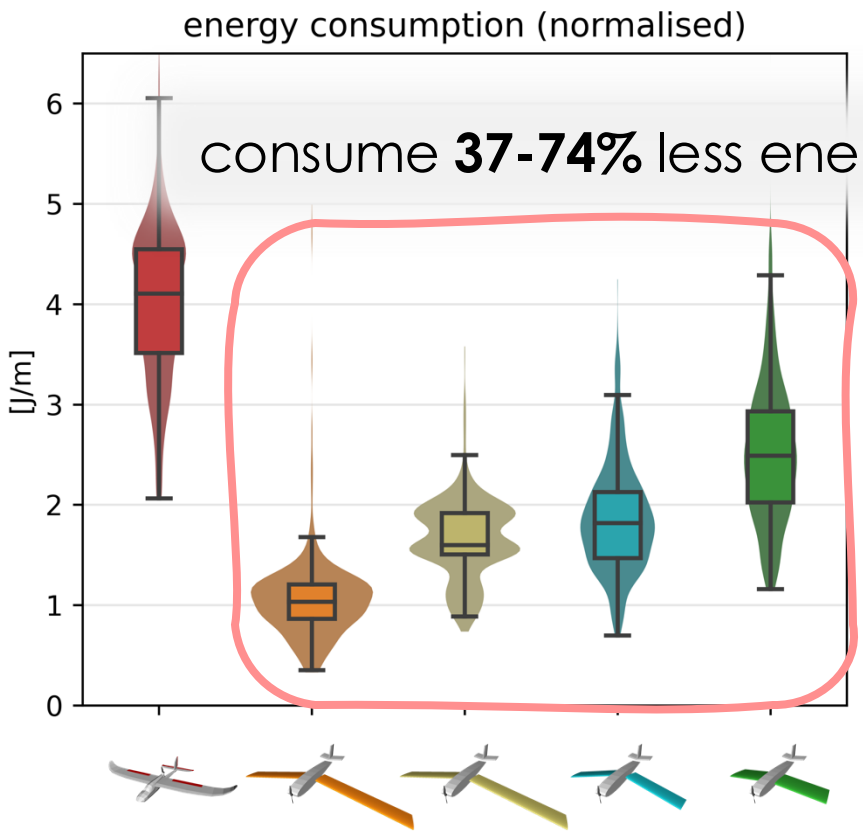
name	bix3	opt1	Opt2	opt3	Opt4
mass	1012 g	760 g	800 g	920 g	940 g
max thrust	7.0 N	4.1 N	4.1 N	4.1 N	10.7 N
wingspan	1.55 m	1.86 m	1.86 m	1.24 m	1.10 m
morphing strategies	-	incidence	sweep incidence	incidence sweep dihedral	dihedral sweep incidence

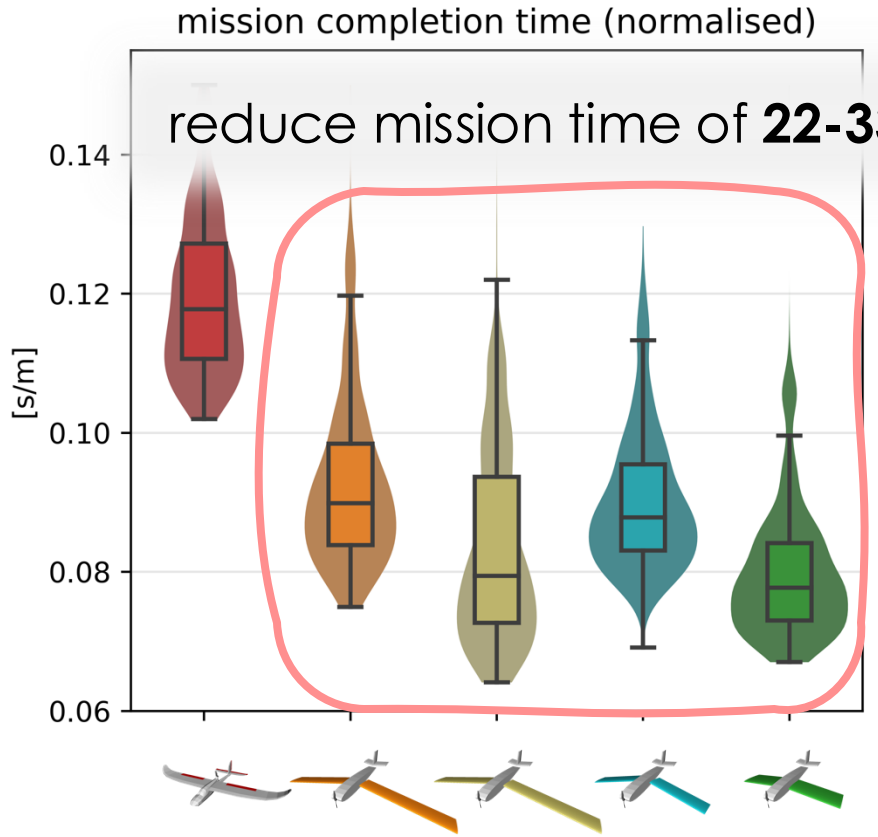


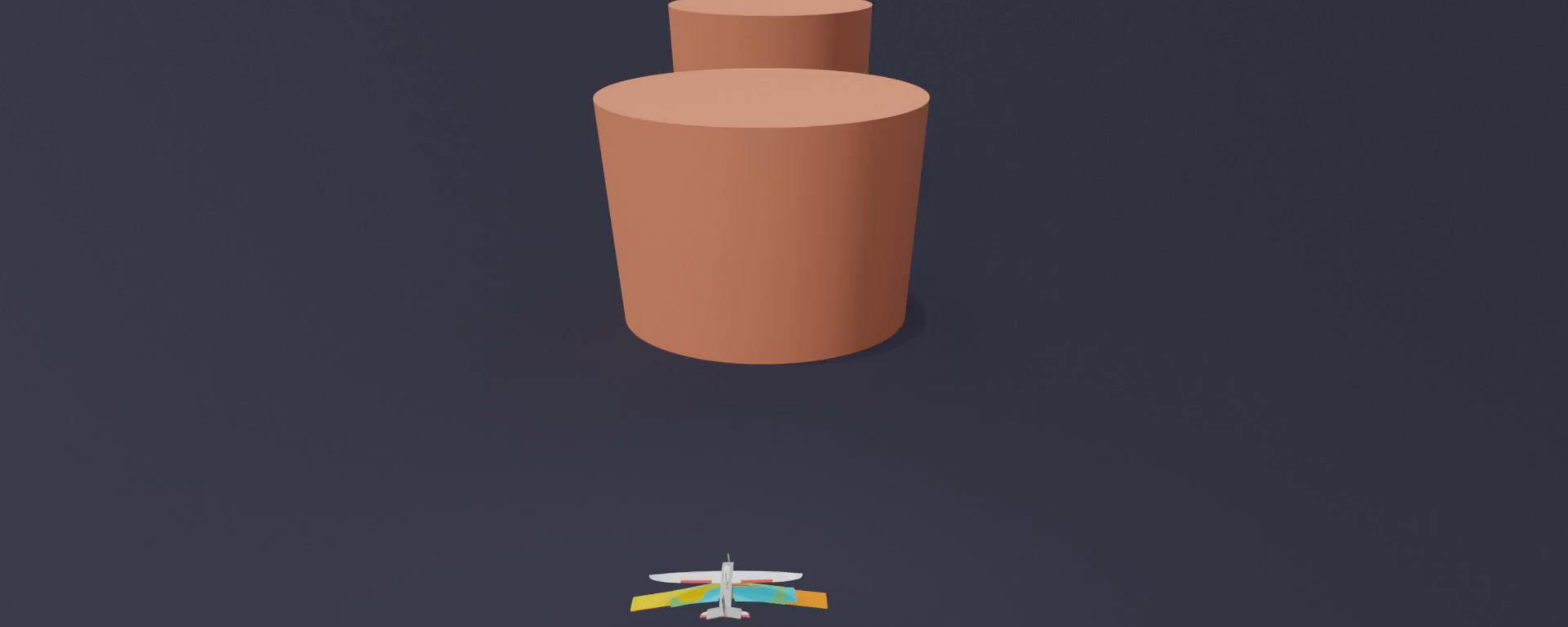

y [m]
4
2
0
−2
−4
0
10
20
30
40
50
60
time x0.5
speed [m/s]
14
13
12
11
10
9
0
10
20
30
40
50
60
x [m]
time x0.5

Validation

Test the drones in 972 scenarios (different from those used for evolution)





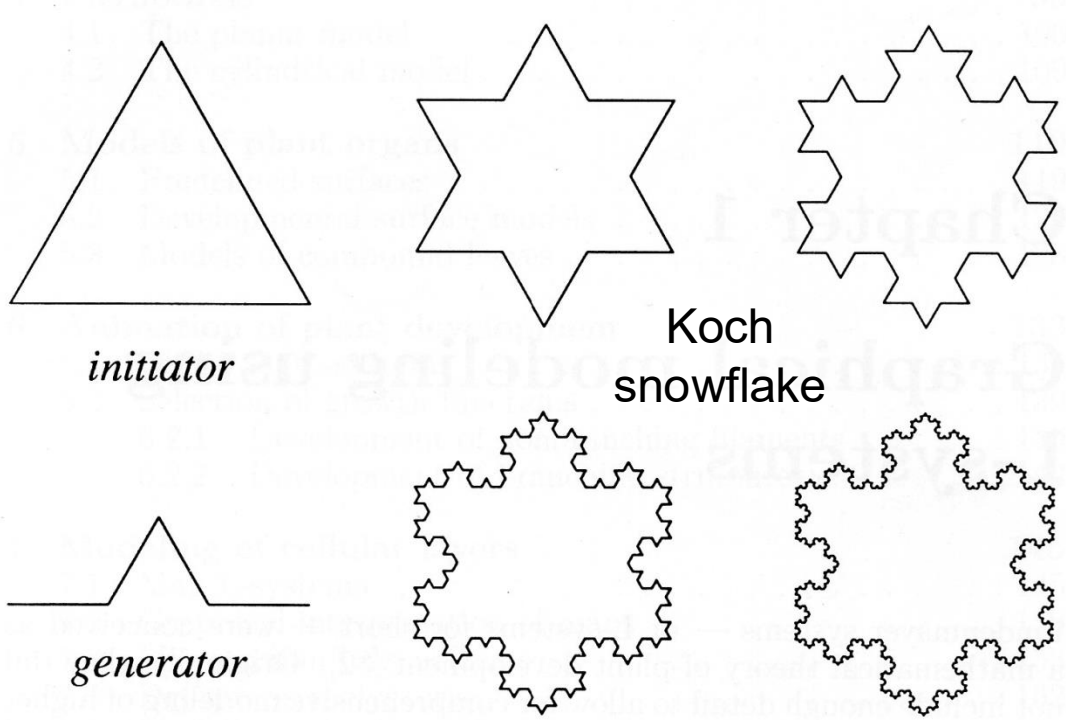


Bergonti, Nava, Wüest, Paolino, L’Erario, Pucci, Floreano (2024) Proceedings of ICRA

Rewriting representations

Rewriting System: recursively replace a sub-component with another sub-component

Fractals: *Replace edges of a polygon with open polygons and rescale at each iteration [von Koch, 1905]*



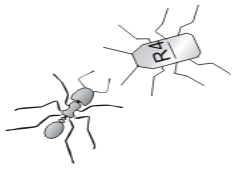
Several types of rewriting systems have been developed. For example:

L-systems (plants)

Cellular automata (anything)

Language systems (language)

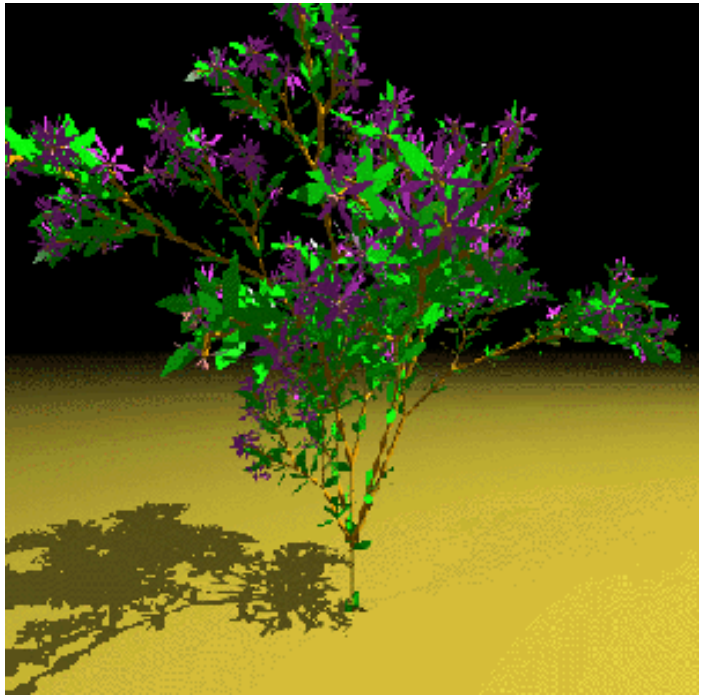
Matrix rewriting (neural networks)



L-systems [Lindenmayer, 1968]

Lindenmayer systems, or L-systems for short, are mathematical models to describe biological morphologies through a growth process. They were originally applied to model growth of plants.

Aristid Lindenmayer



Artificially generated tree

http://local.wasp.uwa.edu.au/~pbourke

L-system: Definition

L-systems are rewriting systems that operate on symbol strings.

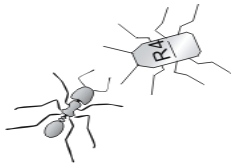
An L-system is composed of:

1. A set of symbols s forming an *alphabet* A
2. An *axiom* ω (initial string of symbols) $s_k, s_z, s_{v,\ldots}$
3. A set $\pi = \{p_i\}$ of *production rules* $p_i : s_k \rightarrow s_z$.

The following assumptions hold:

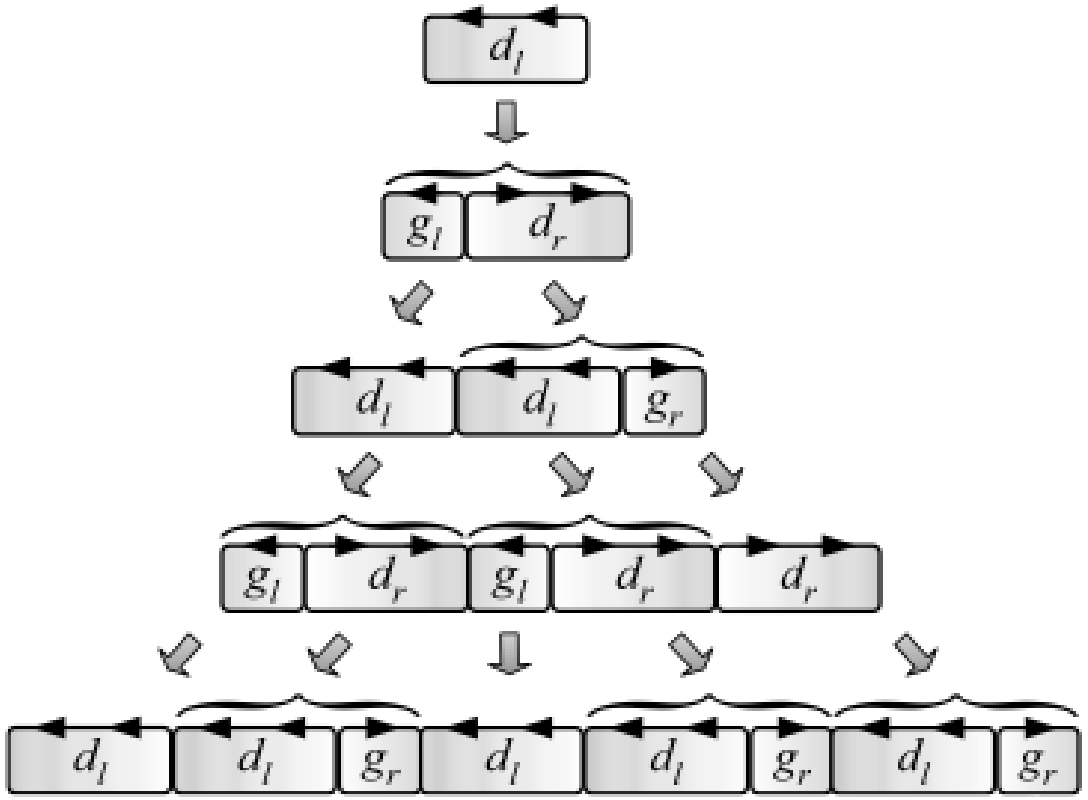
1.Production rules are applied in parallel and replace recursively all symbols in the string.

2.If no production rule is specified for a symbol s, then we assume the identity production rule $p_o : s_k \rightarrow s_k$



L-system: 1D Example

Development of a multicellular filament of blue-green bacteria *Anabaena catenula* [Lindenmayer 1968]



Cells can be in a "growing" state g or in a "dividing" state d with left or right polarity

$$A = \{g_r, g_l, d_r, d_l\}$$

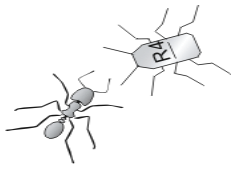
$$\omega = d_l$$

$$p_1 = d_r \rightarrow d_l\, g_r$$

$$p_2 = d_l \rightarrow g_l\, d_r$$

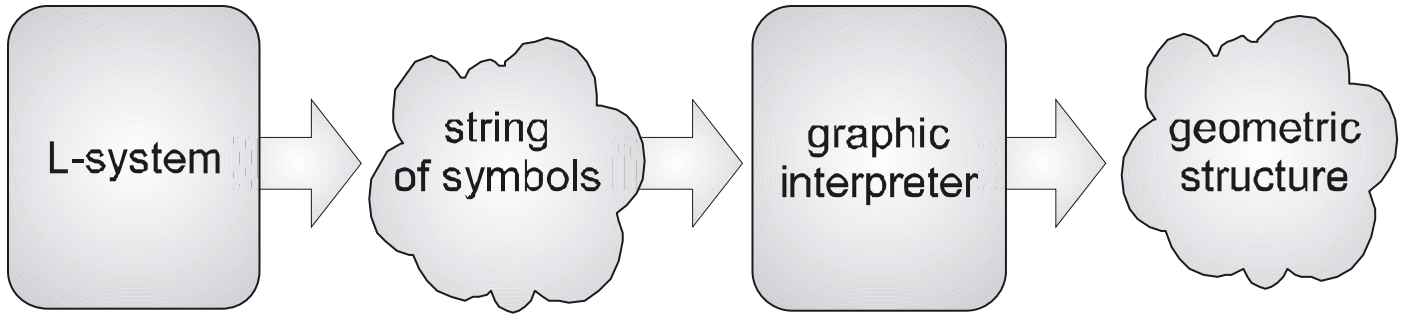
$$p_3 = g_r \rightarrow d_r$$

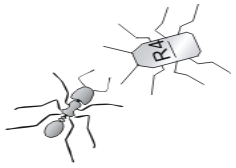
$$p_4 = g_l \rightarrow d_l$$



Graphics Interpretation

- Using symbols that represent directly geometric entities such as 1D or 2D cells becomes rapidly impractical.
- We can increase the graphic potential of L-systems by following the phase of production of strings of symbols with a phase of graphic interpretation of the strings





Turtle Graphics Interpretation

In 2D, the turtle (printer) state is defined by the triplet x, y, α where the Cartesian coordinates (x, y) represent the turtle's position and the angle α, also known as heading, represents the facing direction.

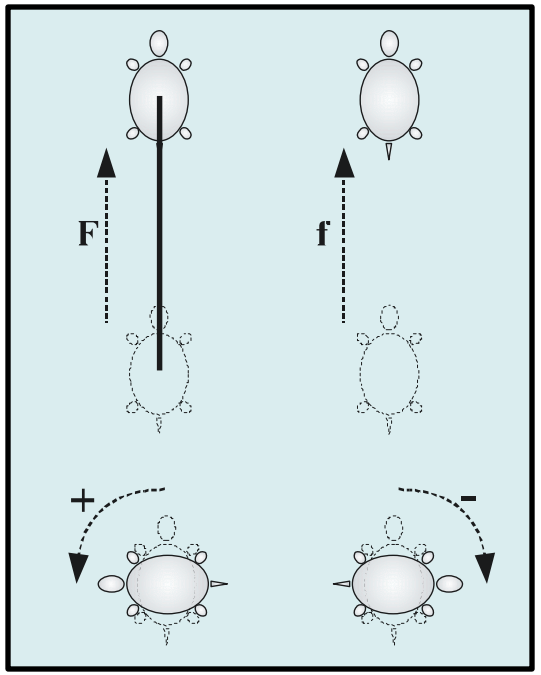
Given the step size d and the angle increment δ, the turtle can respond to the following commands:

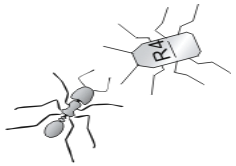
F : move forward by a step while drawing a line.

f : move forward by a step without drawing a line.

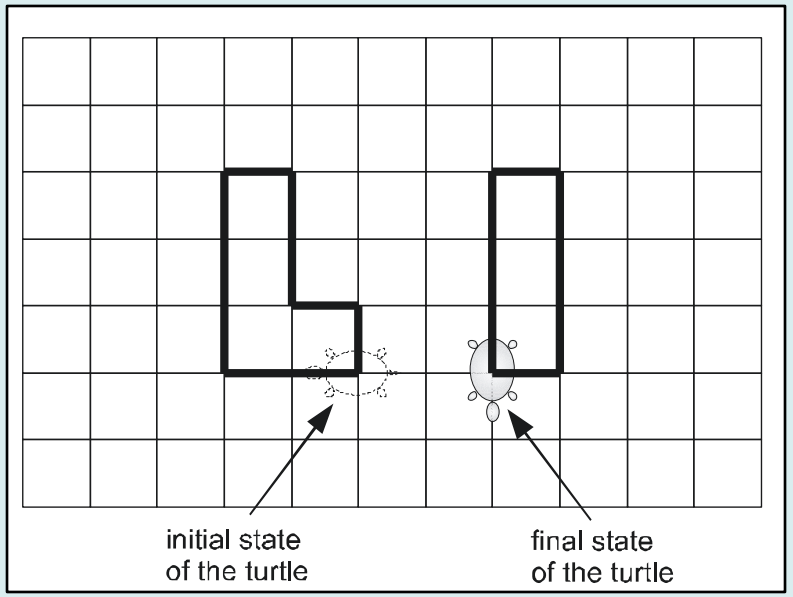
+ : turn left (counterclockwise) by angle δ.

– : turn right (clockwise) by angle δ.





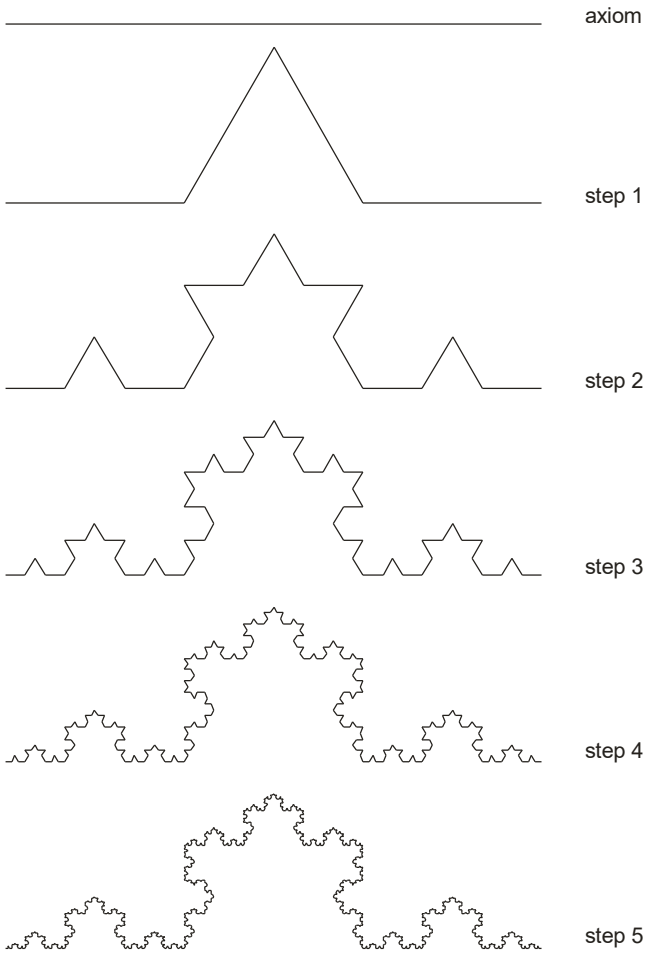
Examples



$\delta = 90^\circ$, A={ $F, f, +, -$ }

$\omega = FF - FFF - F - FF + F -$

$F + ff F + FFF + F + FFF$



$\delta = 60^\circ$, A={ $F, f, +, -$ }, $\omega = F$

$p = F \rightarrow F+F--F+F$

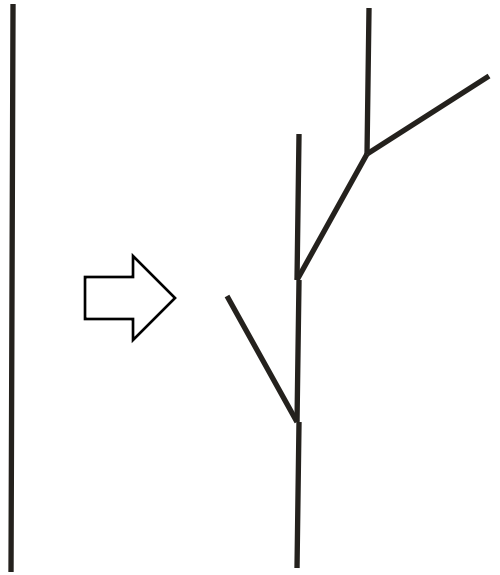
Bracketed L-systems

In drawing branching structures using the turtle interpreter it is necessary to reposition the turtle at the base of a branch after the drawing of the branch itself

- Two new symbols:

[Save current state of the turtle (position, orientation, color, thickness, etc.).

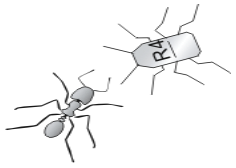
] Restore the state of the turtle using the last saved state (no line is drawn).

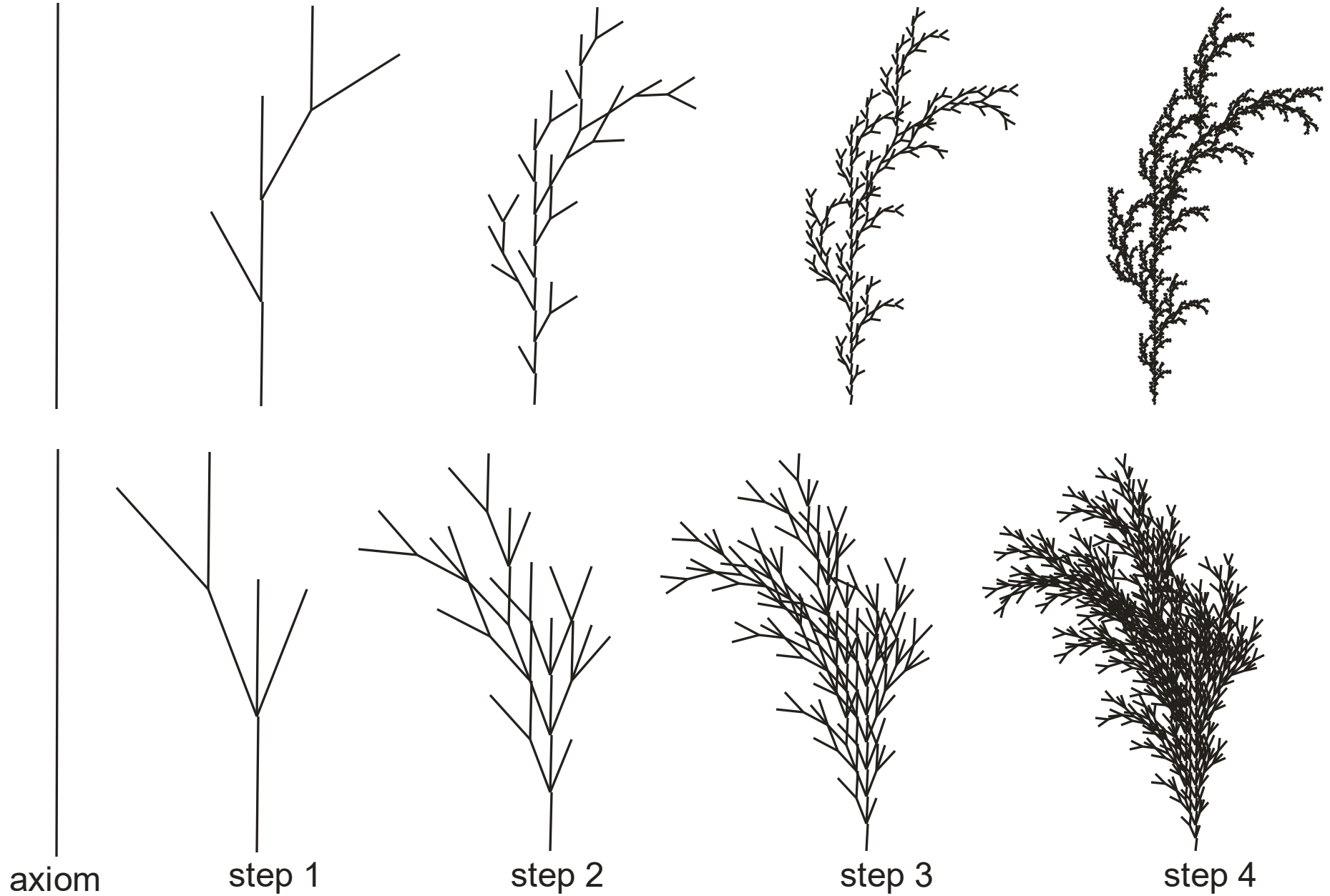


$\delta \quad \delta = 29^\circ \;,\; \mathrm{A} = \{ F, +, -, [,] \}$

$\omega = F$

$p = F \rightarrow F\,[+F]F\,[-F\,[+F][-F]]F$




axiom
step 1
step 2
step 3
step 4

Stochastic L-systems

- In nature individuals of the same species are not identical.
- Specimen variability can be modeled by associating probabilities to production rules
- The sum of all probabilities over the same symbol must be 1

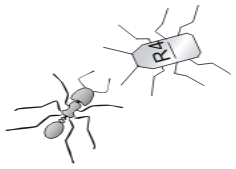
$\delta \quad \delta = 29^\circ \quad , \quad \mathrm{A} = \{ F, +, -, [,] \}$

$\omega = F$

$p_1 = F \xrightarrow{1/3} F[+F]F[-F]F$

$p_2 = \mathrm{F} \xrightarrow{1/3} F[-F]F[+F]F$

$p_3 = \mathrm{F} \xrightarrow{1/3} F[-FF-F]F$



Application to computer graphics

http://gug.sunsite.dk/

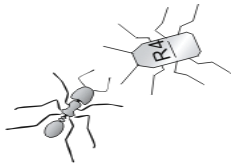
http://www.ii.uib.no/~knute

http://www.uweb.ucsb.edu/~svetlin

How to identify rewriting rules?

- By hand
  - When the rewriting rules are explicitly given (e.g., fractal curve)
  - When the rewriting rules can be easily deduced from the description of the developmental process (e.g., development of bacteria filaments and moss leaves)
  - When the resulting morphologies have only an aesthetic function

- With evolutionary algorithms
  - When the morphology serves a function that can be measured: neural network, a gene regulatory network, an electronic circuit, a robotic body, etc.



Neural architecture by matrix rewriting

A *rewriting system* [Kitano, 1990] that encodes a grammar to represent network topologies

Genome encodes the rewriting (grammar) rules, such as: *ABCD adaa cbba baac abad 0001 1000 0010 0100*

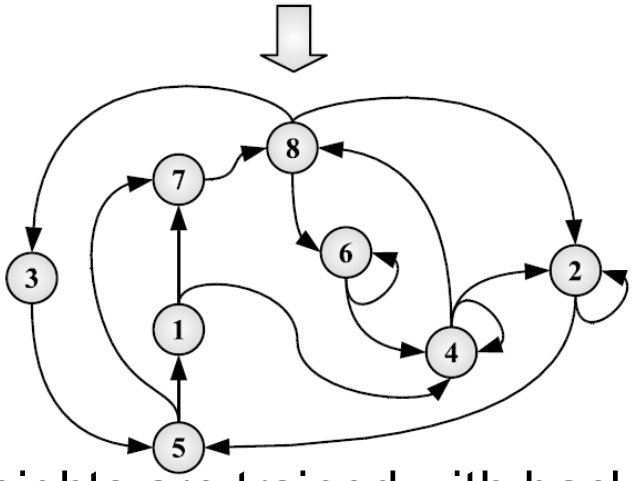
$$\mathcal{E} \rightarrow \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

$$A \rightarrow \begin{bmatrix} a & d \\ a & a \end{bmatrix} \quad B \rightarrow \begin{bmatrix} c & b \\ b & a \end{bmatrix} \quad C \rightarrow \begin{bmatrix} b & a \\ a & c \end{bmatrix} \quad D \rightarrow \begin{bmatrix} a & b \\ a & d \end{bmatrix}$$

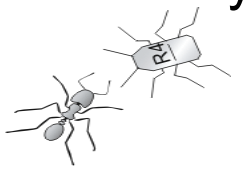
$$a \rightarrow \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \quad b \rightarrow \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \quad c \rightarrow \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \quad d \rightarrow \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$\mathcal{E} \Rightarrow \begin{bmatrix} A & B \\ C & D \end{bmatrix} \Rightarrow \left[\begin{array}{cc|cc} a & d & c & b \\ a & a & b & a \\ \hline b & a & a & b \\ a & c & a & d \end{array}\right] \Rightarrow \left[\begin{array}{cc|cc|cc|cc} 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{array}\right]$$

	1	2	3	4	5	6	7	8
1	0	0	0	1	0	0	1	0
2	0	1	0	0	1	0	0	0
3	0	0	0	0	1	0	0	0
4	0	1	0	1	0	0	0	1
5	1	0	0	0	0	0	1	0
6	0	0	0	1	0	1	0	0
7	0	0	0	0	0	0	0	1
8	0	1	1	0	0	1	0	0

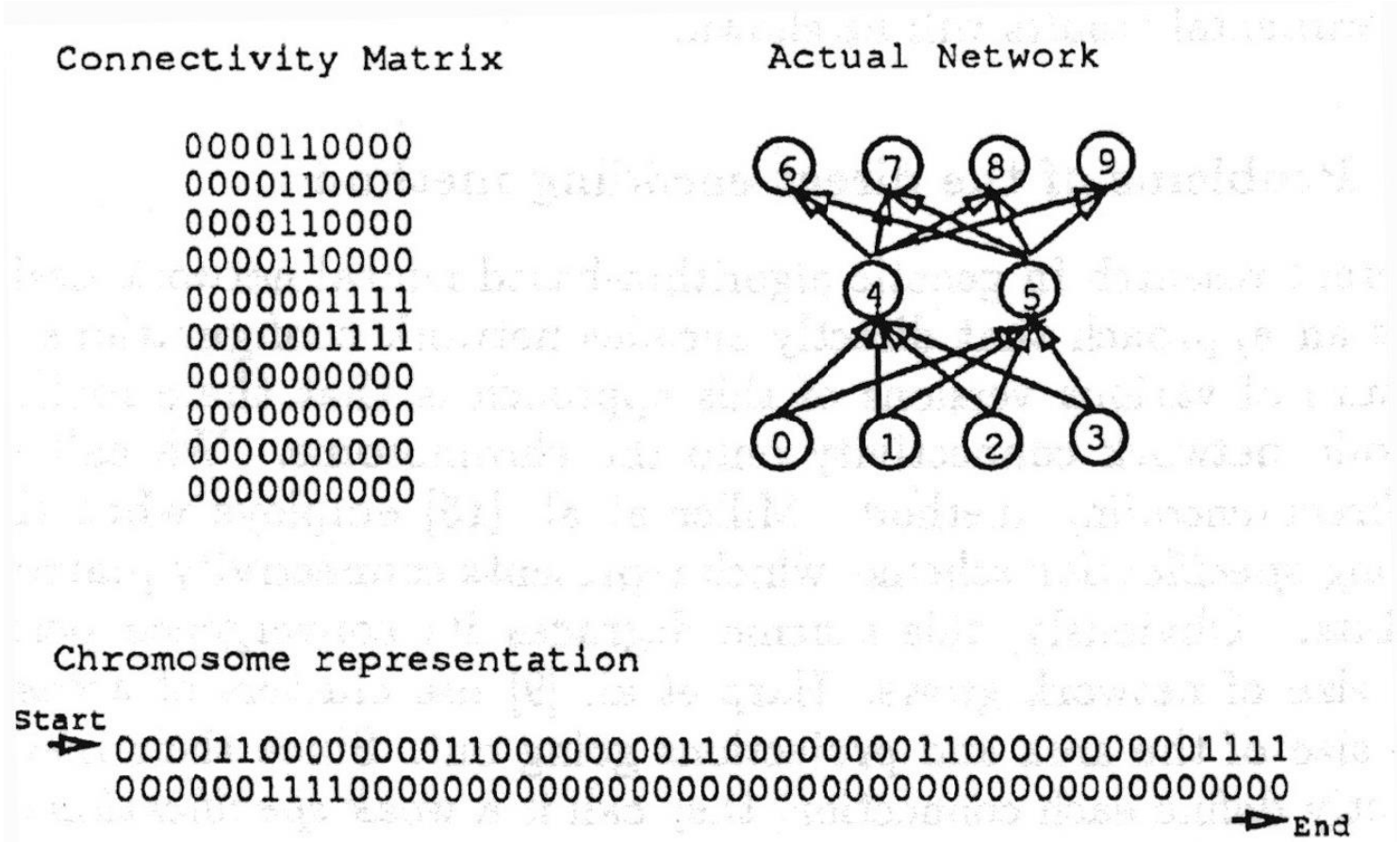


Only the presence/absence of connections is evolved. Weights are trained with backpropagation.



Direct Encoding of Network Topology

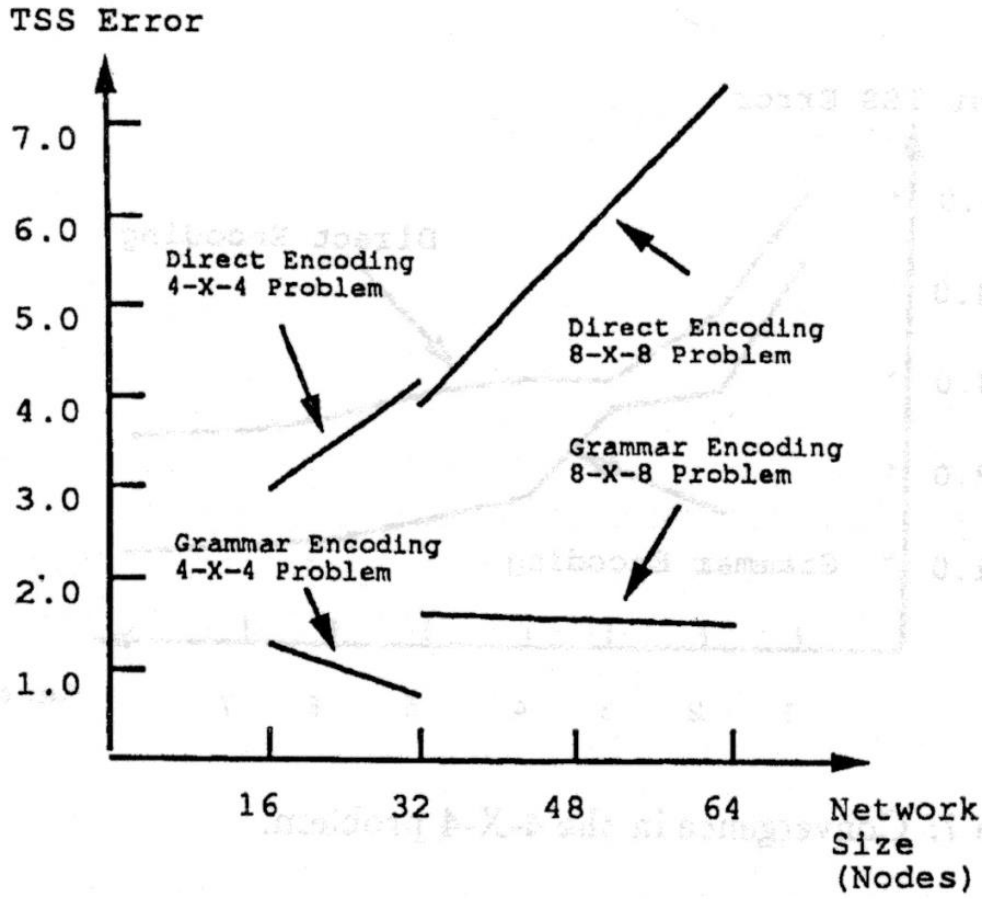
Length of genetic code is proportional to number of neurons in the network



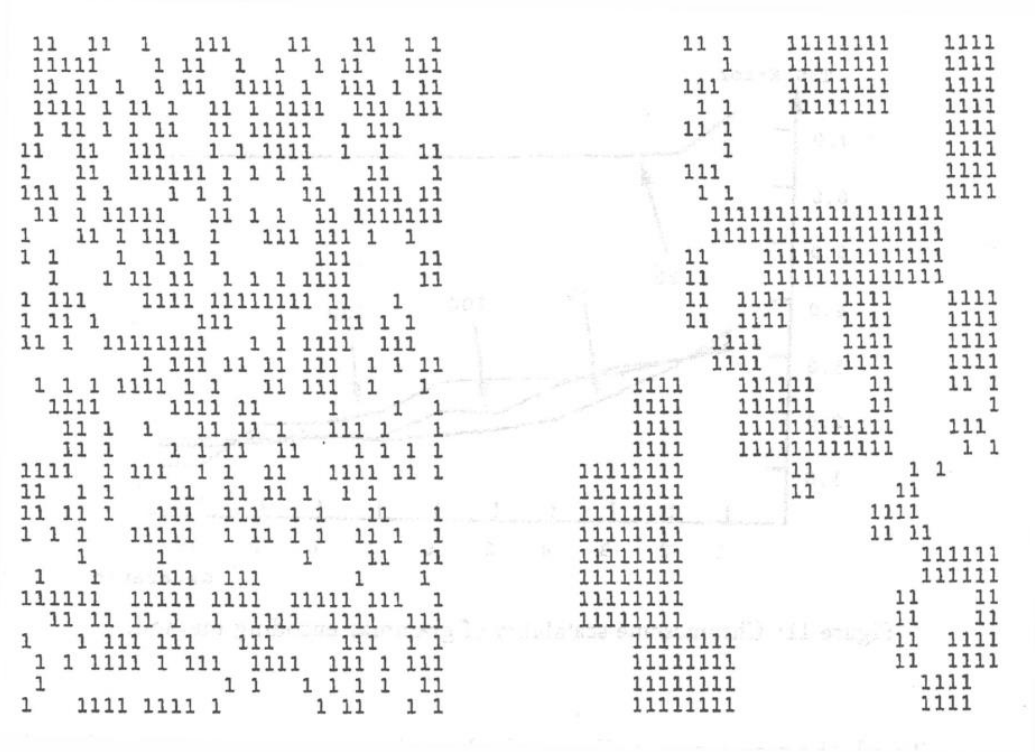
Evolving autoencoder architectures

- Autoencoder performance is worse with direct encoding and worsens with network size
- Topologies evolved with grammar encoding are more regular (good for spatial information processing, such as convolutional neural networks).

Performance comparison



Architecture comparison



Direct encoding

Grammar encoding