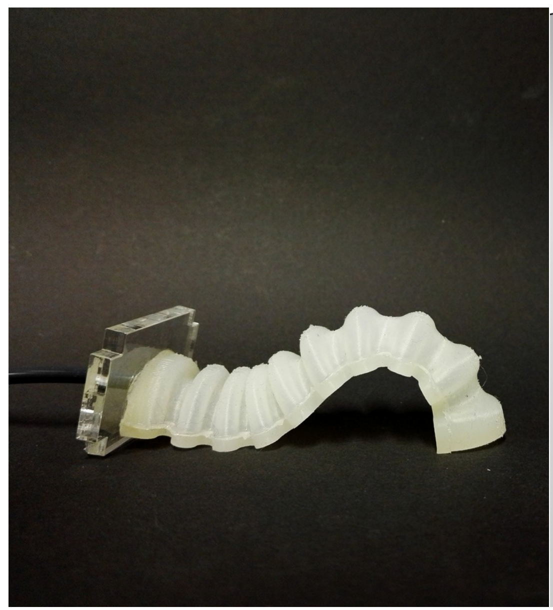
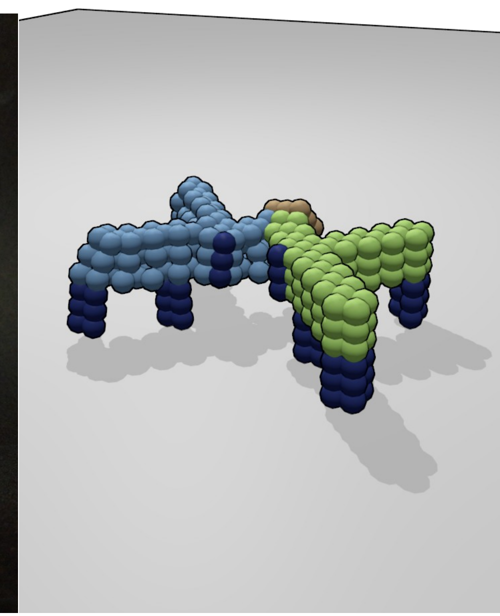
# Computational Design Synthesis of Virtual Locomotive Soft Robots

Merel van Diepen and
Prof. Dr. Kristina Shea

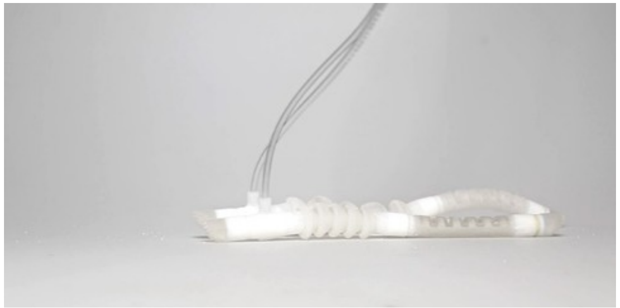
Engineering Design +
Computing Laboratory
ETH Zürich





# Introduction to Soft Robotic Systems

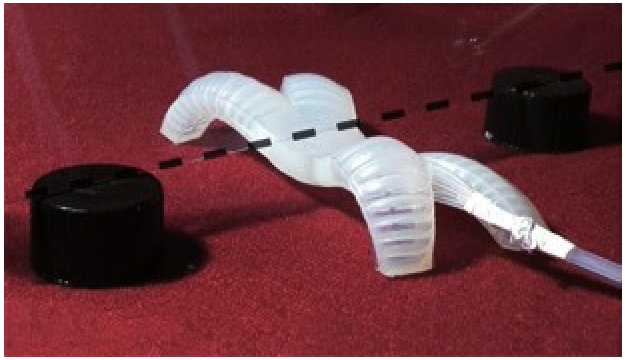
- Compliant materials
- Large number of degrees of freedom
- No joints sensitive to contamination



(Modular pneumatic toolkit, Du Pasquier, 2017)

- Manual-design of soft locomotion robots is challenging
- Locomotion is essential to most robotic tasks

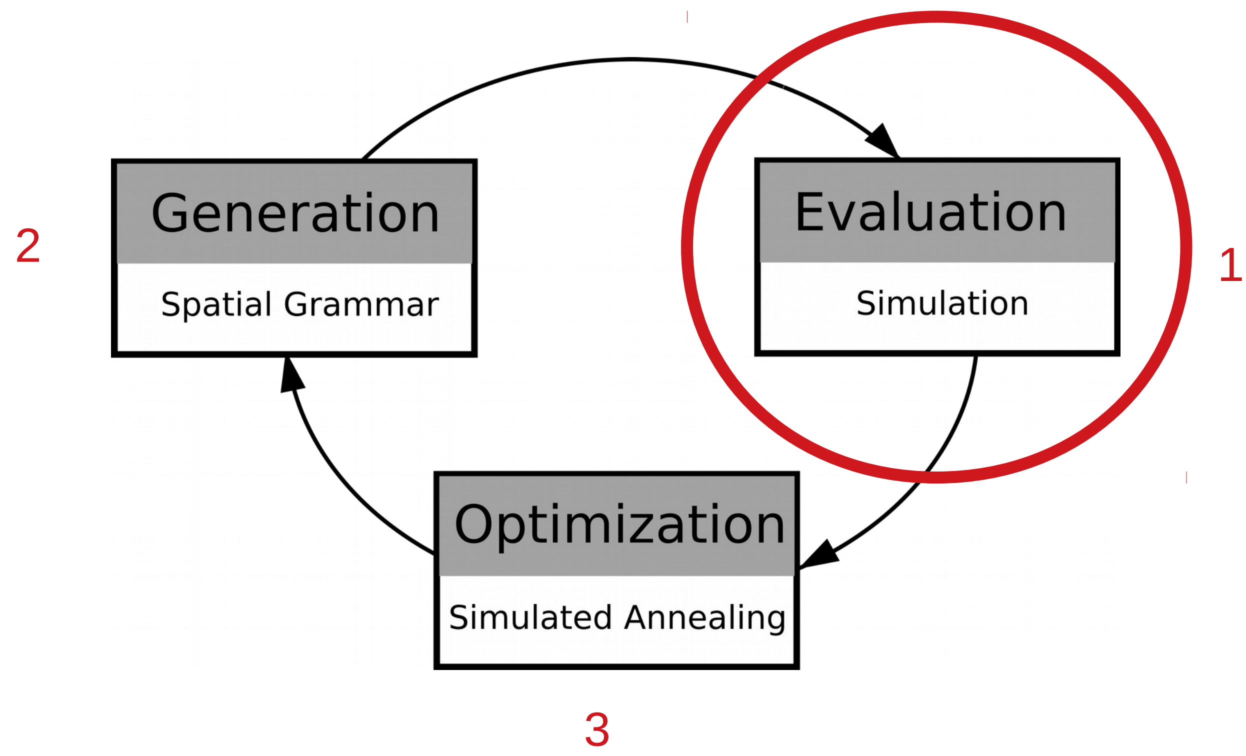
→ Computation Design Synthesis (CDS) of virtual, soft locomotion robots



(Multigait Soft Robot, Shepherd et al., 2011)

# Overview

Computational Design Synthesis (CDS) of virtual, soft locomotion robots



# Simulation of Soft robots

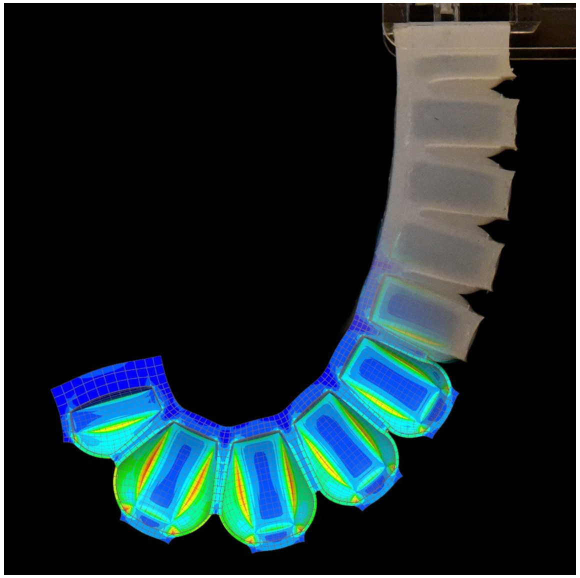
Characteristics of soft robots simulation:

- Highly non-linear materials
- Large displacements
- Collision, self-collision

Methods:

- Finite Elements Analysis
  Unstable, computationally expensive

- Forced-based Soft-body Dynamics
- Position-based Soft-body Dynamics
  (too) unstable (for optimization)



(M. Dreyer, ED+C, ETH Zürich, 2016)

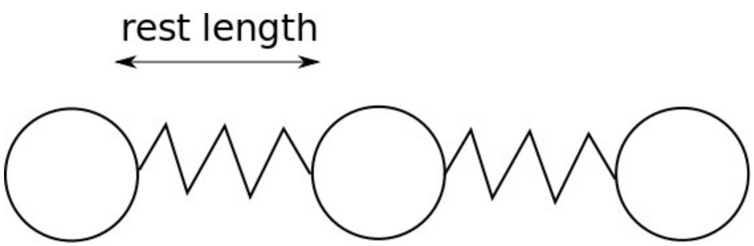
# Simulation Method

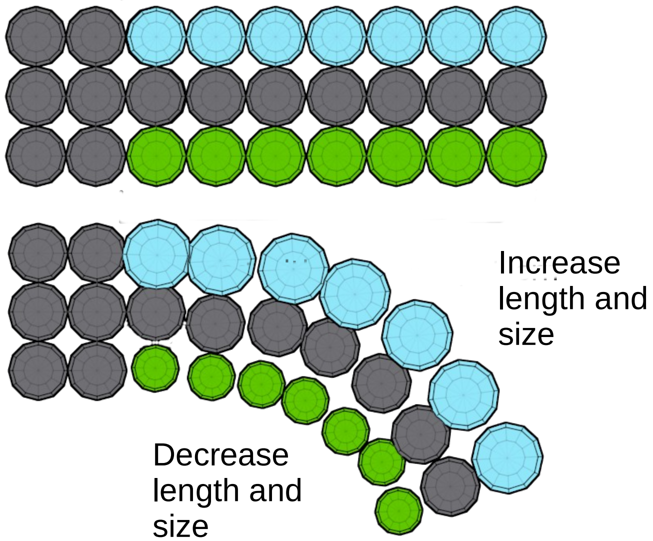
Rigid body approximation

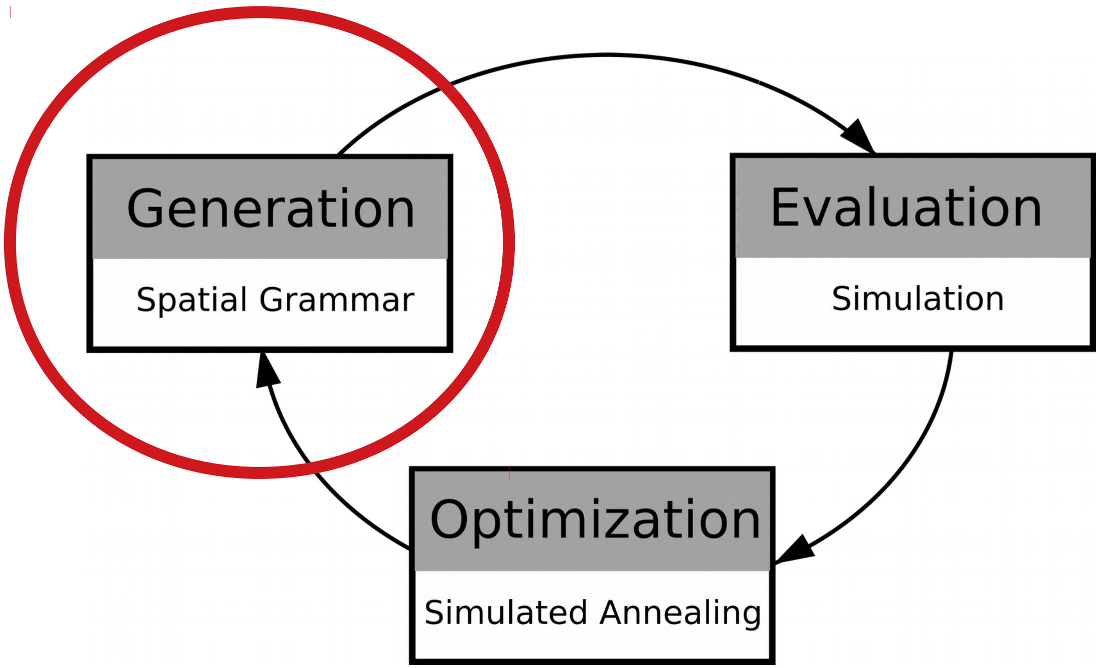
- Bodies connected by springs
- Activate by changing
  - Springs rest-length
  - Body sizes

- Dynamic simulation
- Stable rigid body collision
- Self-collision handled as normal collision

- Using Bullet Physics Library



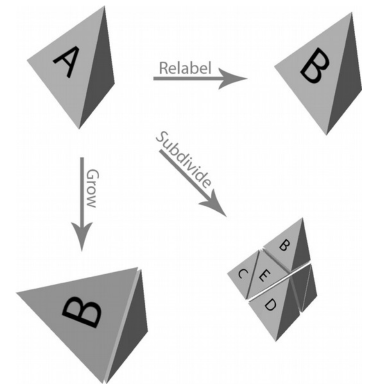




Generation
Spatial Grammar
Evaluation
Simulation
Optimization
Simulated Annealing

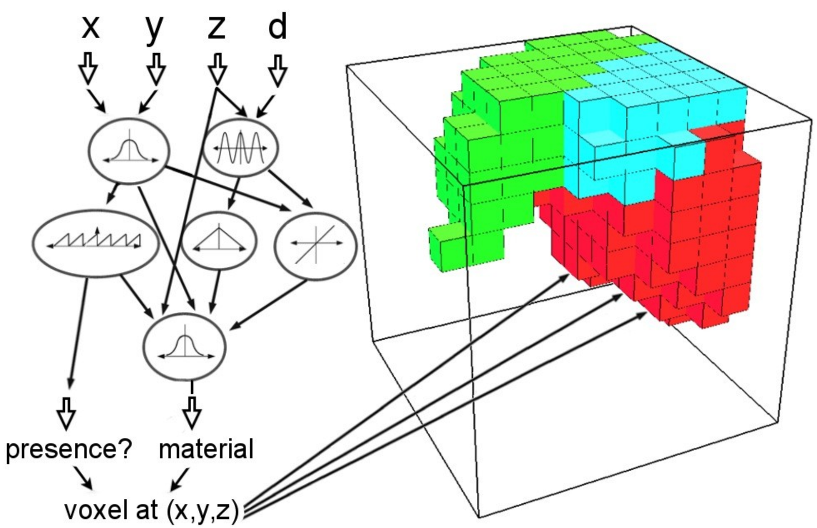
# Background: Generation Methods for Soft Robots

Indirect encoding of designs:

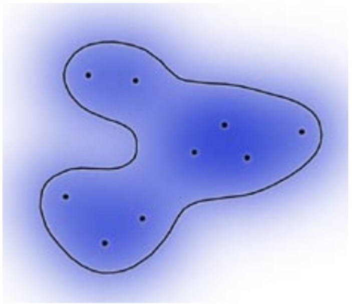
- L-System-like
- Gaussian mixture points
- Composition Pattern Producing Networks




(Growing and Evolving Soft Robots, Rieffel et al., 2013)




(Unshackling Evolution: Evolving Soft Robots with Multiple Materials and a Powerful Generative Encoding, Cheney et al., 2013)



(Evolving Amorphous Robots, Hiller and Lipson, 2010)

# Why Spatial Grammar

Drawbacks of these methods:

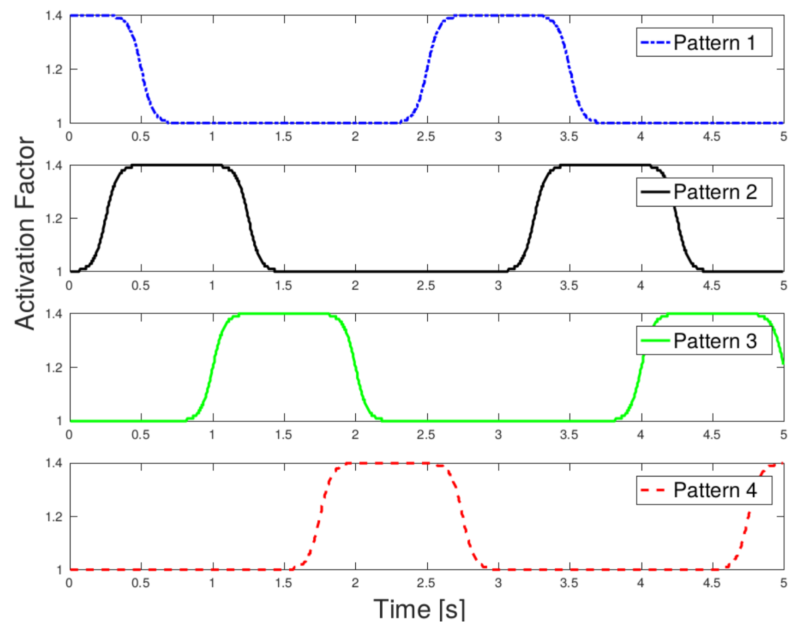
- Black box
- No way to guide the generation towards desired designs

Spatial Grammar:

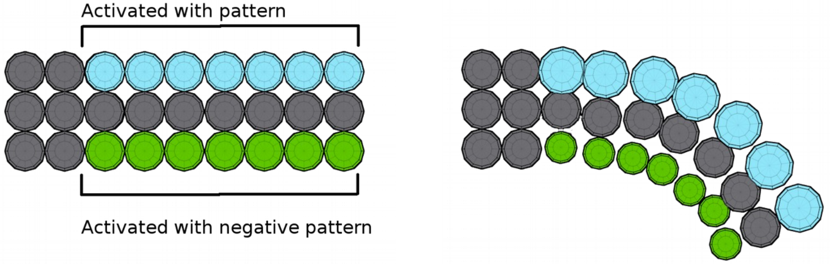
- Generate desired types of designs (e.g. limbs or not)
- Exclude infeasible designs
- Take into account fabrication (constrain to building blocks)
- General requirements and constraints in the generation method instead of checking during evaluation

# Generation Implementation

- Design-space of 15x15x15 empty locations
- Each location can be occupied by a ball
- Balls have
  - A spring-stiffness for connection springs
  - **Fixed** or no activation pattern for spring rest-length and ball size




Rows of cells form bending actuators



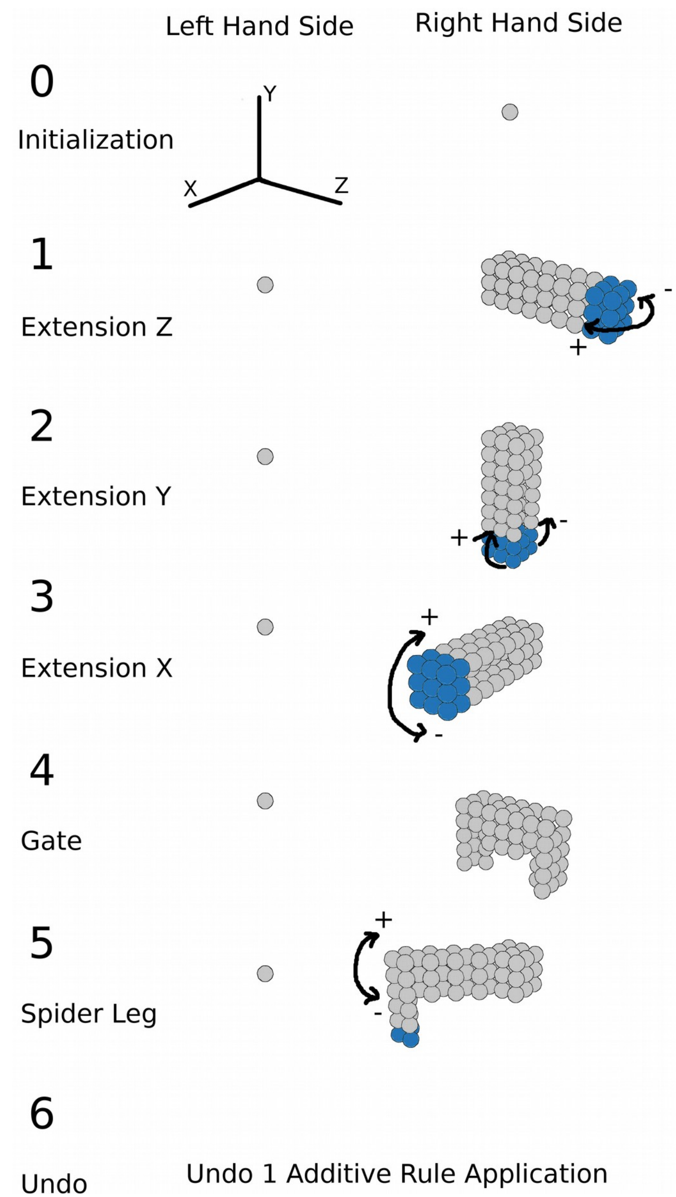
# Spatial Grammar Rules

Goal: generate crawling, hopping and walking
- Rules with useful sub-assemblies

- Rule parameters:
  - Location of application
  - Orientation
  - Connecting spring stiffness
  - Activation pattern
  - Activation positive or negative

- Entire grammar can be used with plane symmetry

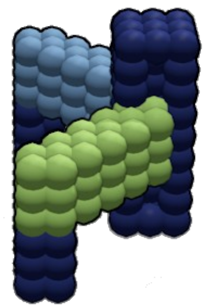
Rule sub-assemblies live



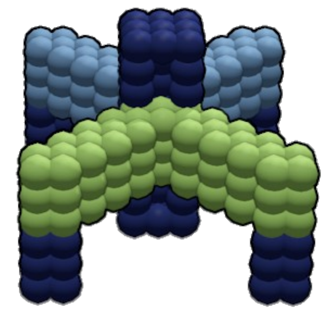
# Generation Example

Example of 3 rules application with plane symmetry.

Rule: Extension Y (2)
Material: Stiff
Activation: None
Offset : 3 in Y-direction

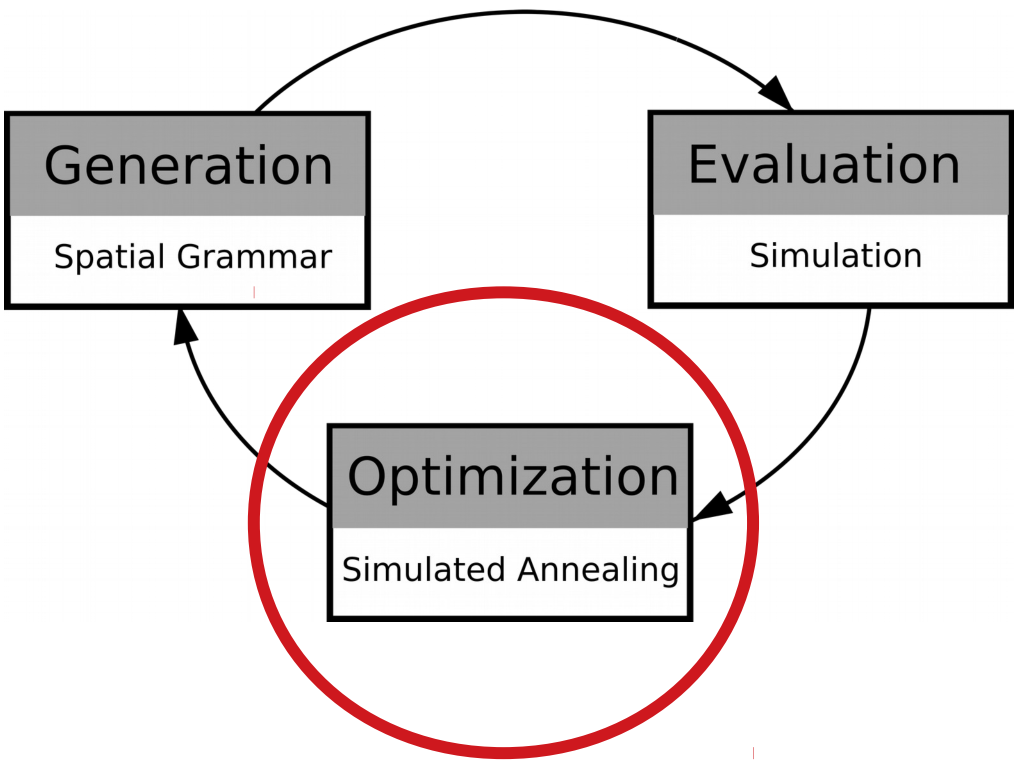


Rule: Spiderleg (5)
Material: Soft
Activation: Pattern 1
Offset : None
Activation Direction: + Back



Rule: Spiderleg (5)
Material: Soft
Activation: Pattern 1
Offset : None
Activation Direction: + Front

- Rules are picked randomly
- Undo used to limit design size

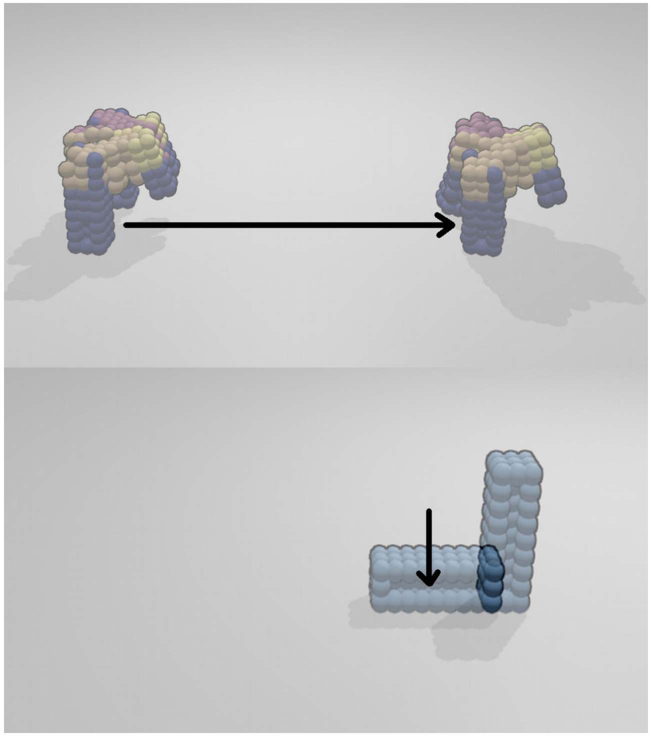

Generation
Spatial Grammar
Evaluation
Simulation
Optimization
Simulated Annealing

# Optimization Problem

The unconstrained optimization problem is given by:

$$\underset{\mathcal{D}}{\text{maximize}} \left( \min_i |\mathbf{x}_i^{t_0} - \mathbf{x}_i^{t_{end}}| \right), \; i = \quad , N \qquad (1)$$

$$\underset{\mathcal{D}}{\text{minimize}} \left( \max \left( \frac{1}{N} \sum_{i=1}^{N} (y_i^{t_0} - y_i^{t_{end}}), 0 \right) \right) \qquad (2)$$



Rewritten as a weighted sum **maximization** problem:

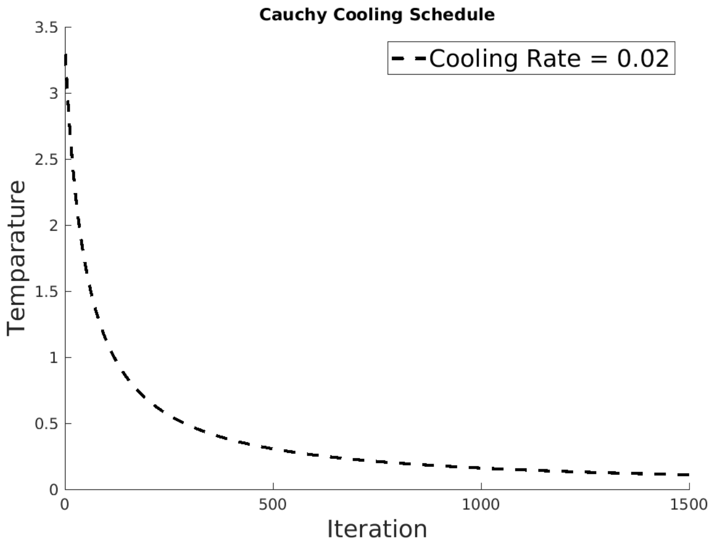
$$f(\mathcal{D}) = \min_i |\mathbf{x}_i^{t_0} - \mathbf{x}_i^{t_{end}}| - \max \left( \frac{1}{N} \sum_{i=1}^{N} y_i^{t_0} - y_i^{t_{end}}, 0 \right)$$

# Optimization Method

## Simulated Annealing

- Stochastic search method
- Accept inferior solutions → escape local minimum
- Probability of acceptance depends on temperature T
- Cauchy temperature schedule

```
for i iterations:
        for j moves:
                apply 1 random rule (or undo if large)
                evaluate
                accept or reject design (depends also on T)
        update T
```

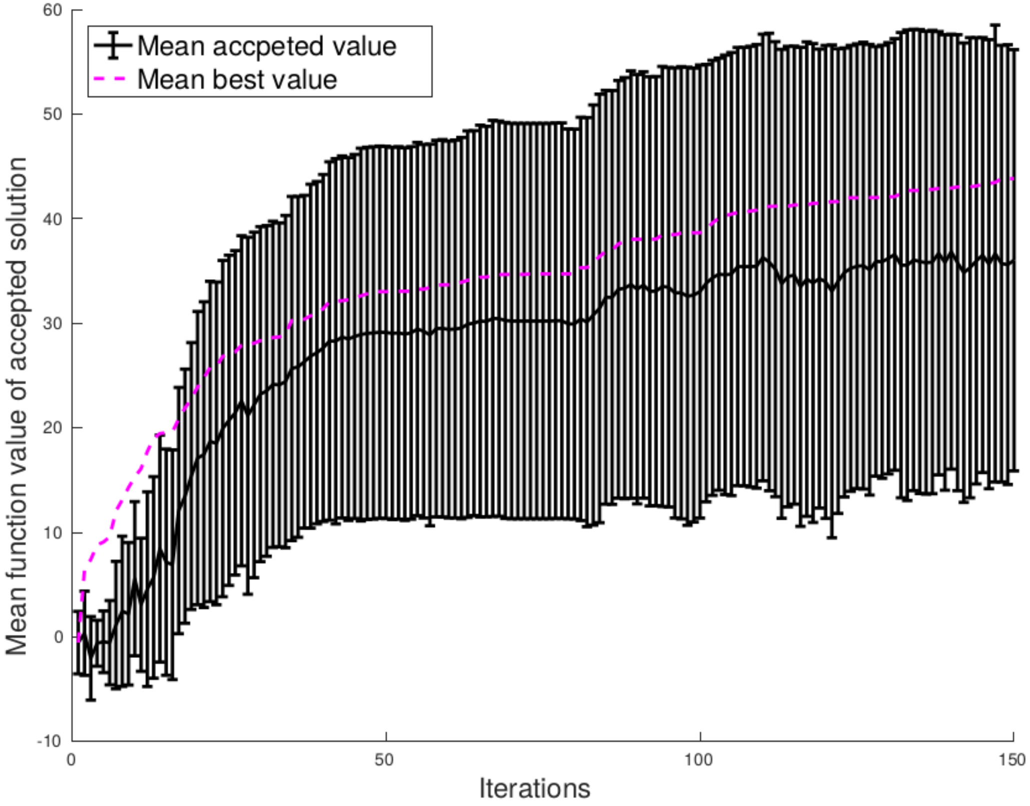


# Resulting Designs

Resulting Designs

# Optimization Result

- 150 iterations, 50 moves
- Mean and standard deviation of 24 runs

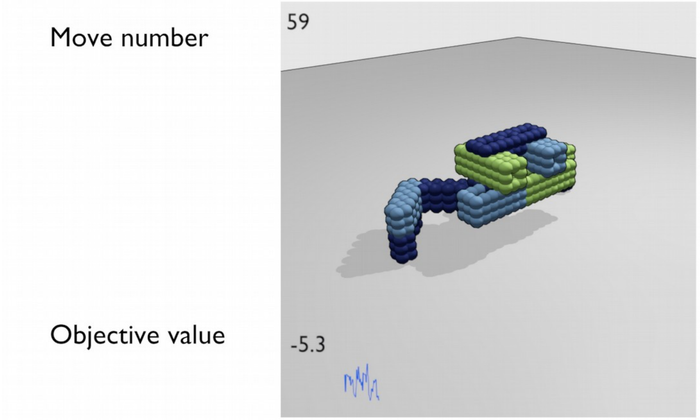



- Large standard deviation
- Converge to local minima

Reheating

# Why Poor Convergence

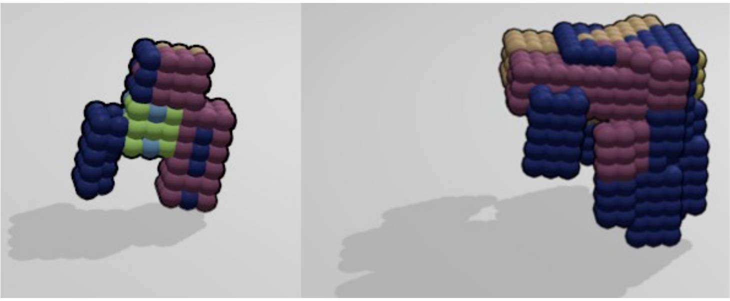
- Many local minima
- Coarse rule-set
  - Large design changes
  - Large objective function value changes




CDS run

- Tuning of the cooling schedule: escaping a local minimum is not the same for different design sizes.

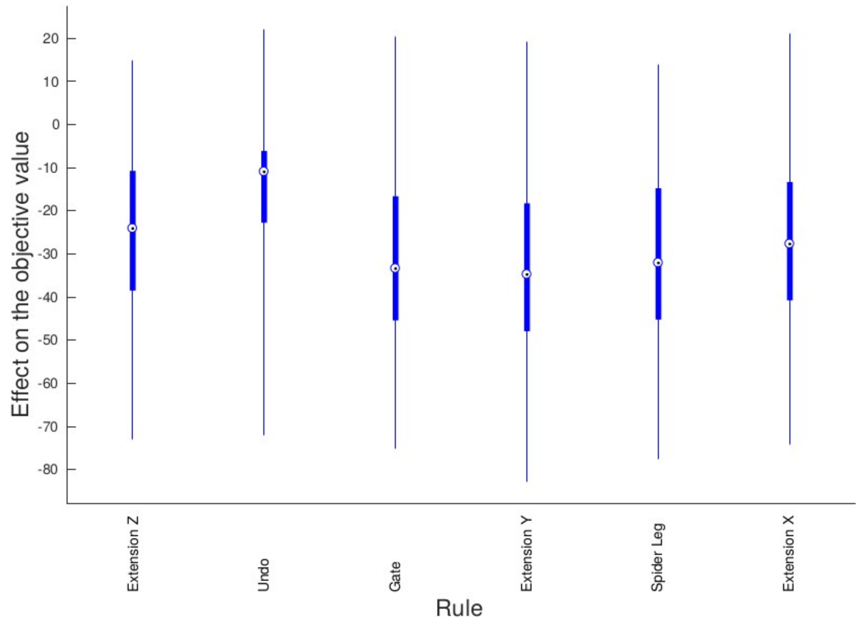
Few rule applications to change behavior

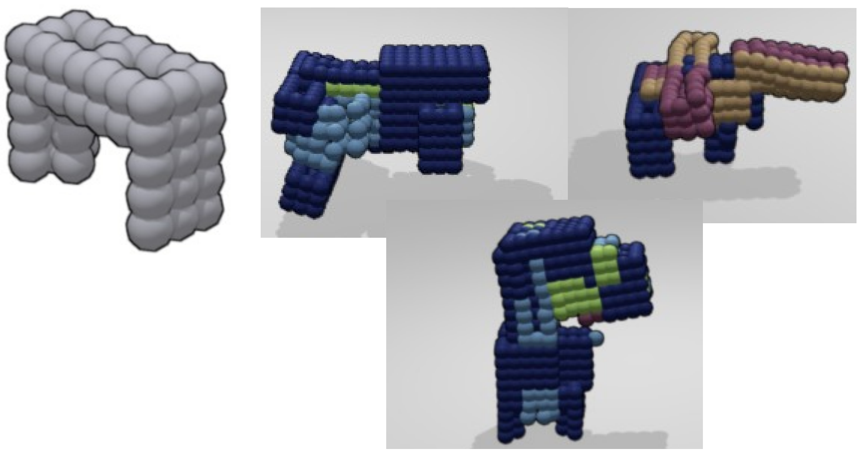


Many rule applications to change behavior

# Spatial Grammar Rule Performance

- Short-term effect of a rule application
- Long-term effect of a rule application
- Anticipated behavior of sub-assemblies



Occurrence in accepted designs

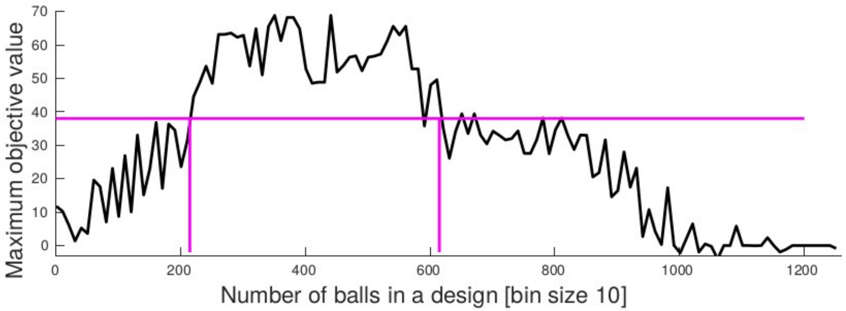
- Undo is used for size control

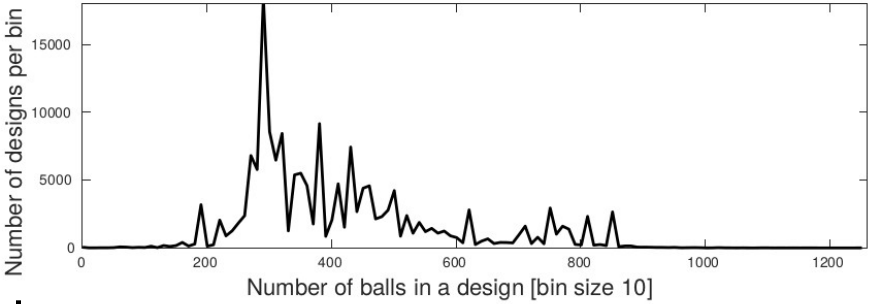
| Rule | Occurrence % |
|---|---|
| Extension z | 9.99 |
| Undo | 56.28 |
| Gate | 8.62 |
| Extension y | 7.28 |
| Spider Leg | 8.59 |
| Extension x | 9.23 |

# Design Sizes for this Grammar+Simulation

Use Undo-rule to limit design size

- What sizes result in good performance?




Lower bound:

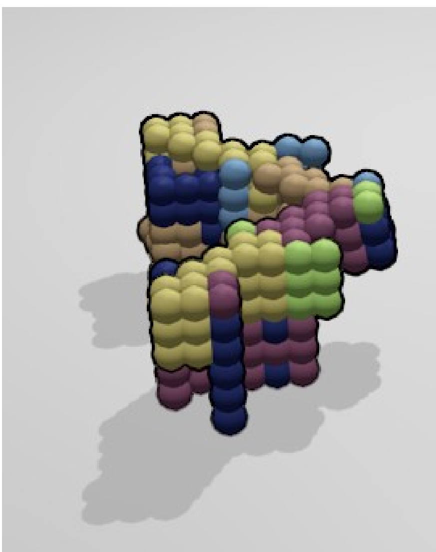
- At least 3 additive rule applications needed

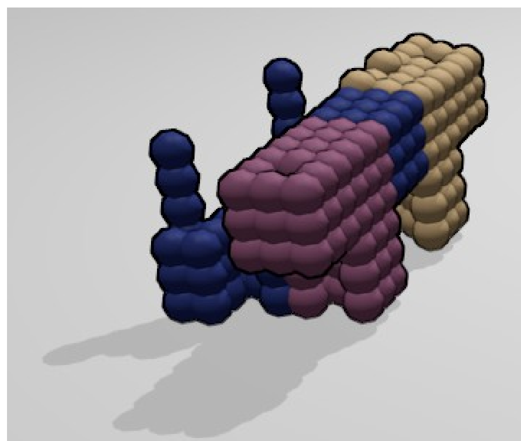
Higher bound:

- Ratio of mass to actuator strength
- More difficult to escaping local minima

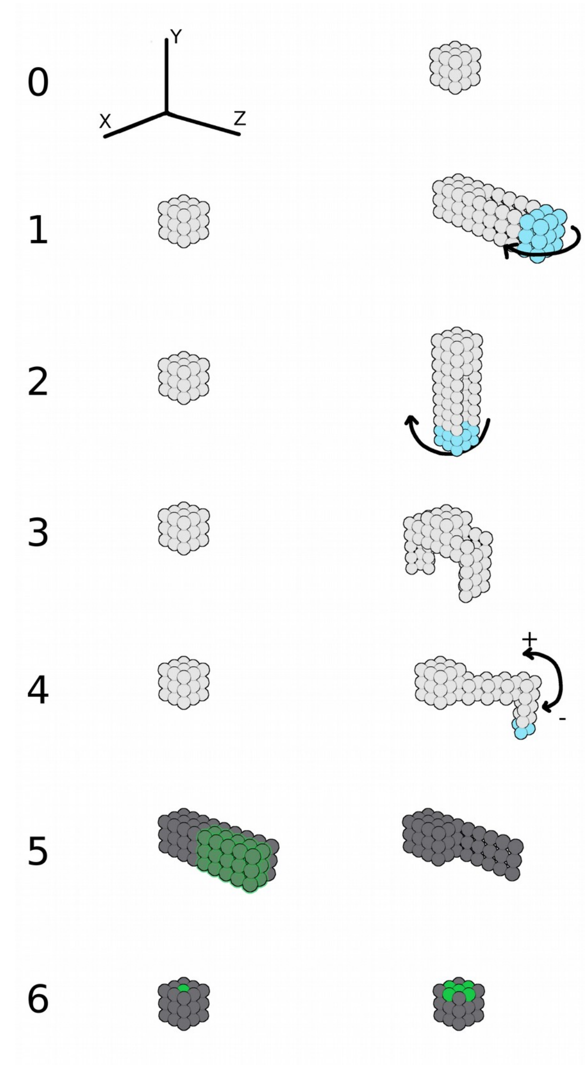
# Alternative Spatial Grammar

- Older version of the presented grammar
- Largest difference: breaking up the sub-assemblies





- Increasingly non-homogeneous
- Left-overs possible

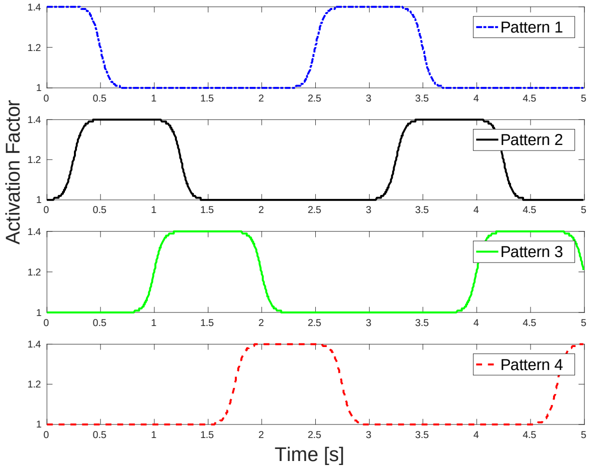


# Conclusion CDS with Fixed Control

- Generates a large variety of gaits
- Guides the generation process towards feasible designs
- Sufficiently accurate for conceptual design

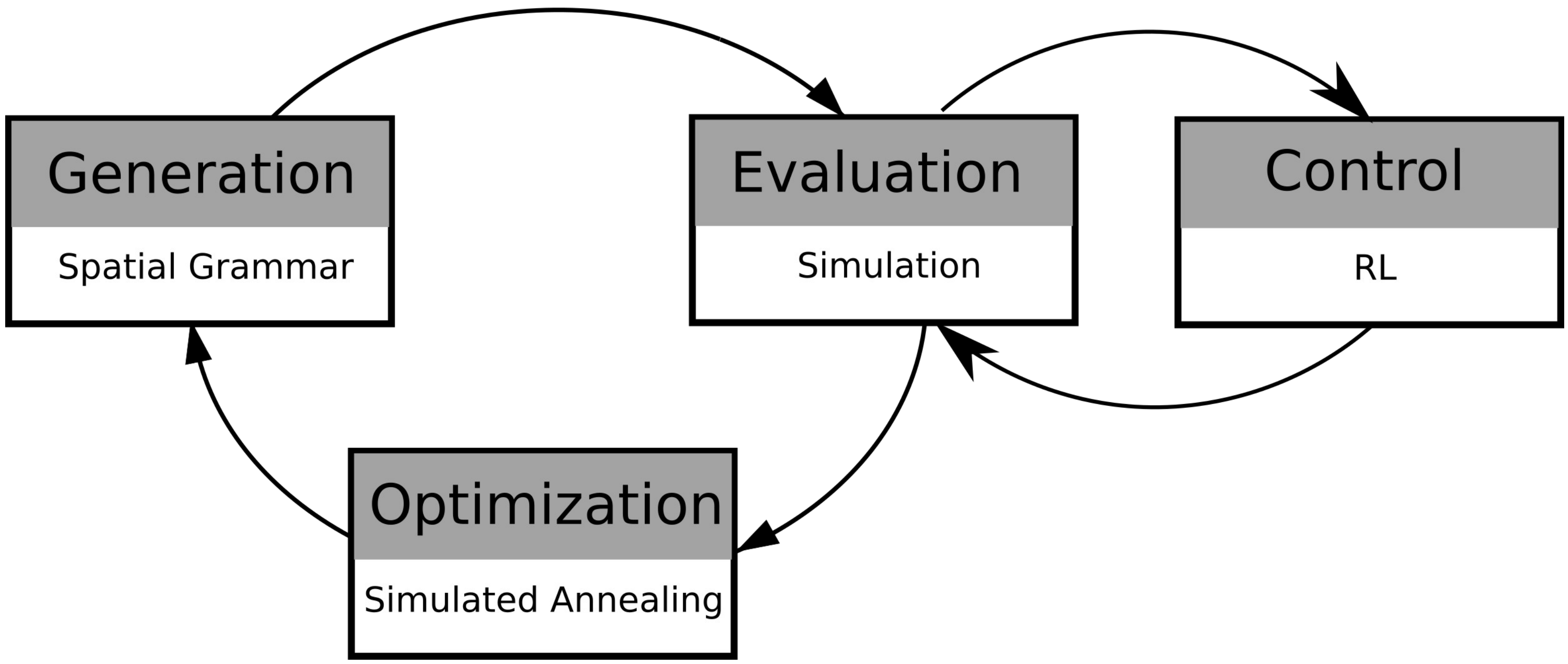
- Grammar, simulation and optimization methods are highly intertwined

Fixed control is a limiting factor.




Good morphology but only with the right control: Worm

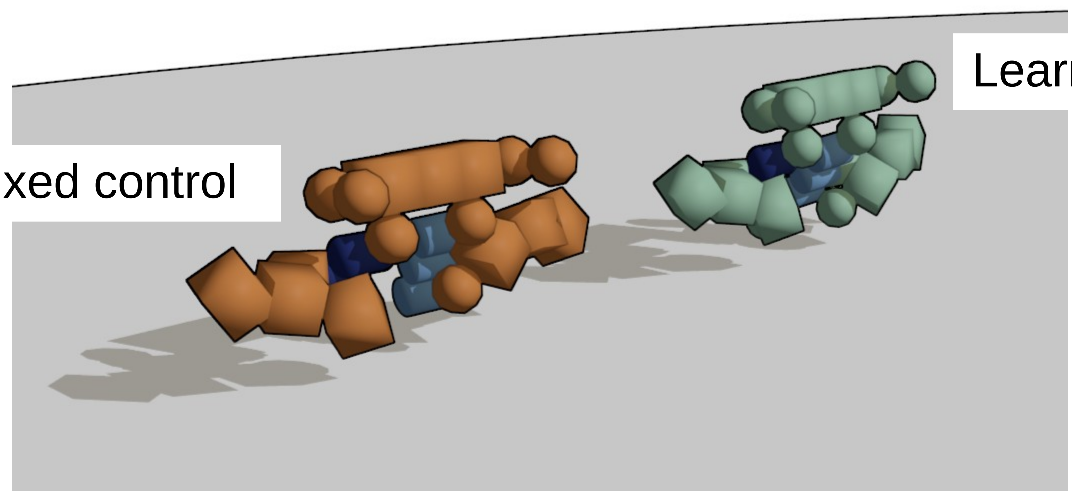
# Outlook: Adding Control to the Loop



# Reinforcement Learning

First step:

- Take result from CDS with fixed control
- Learn a better control for it




fixed VS learned