Design Problems

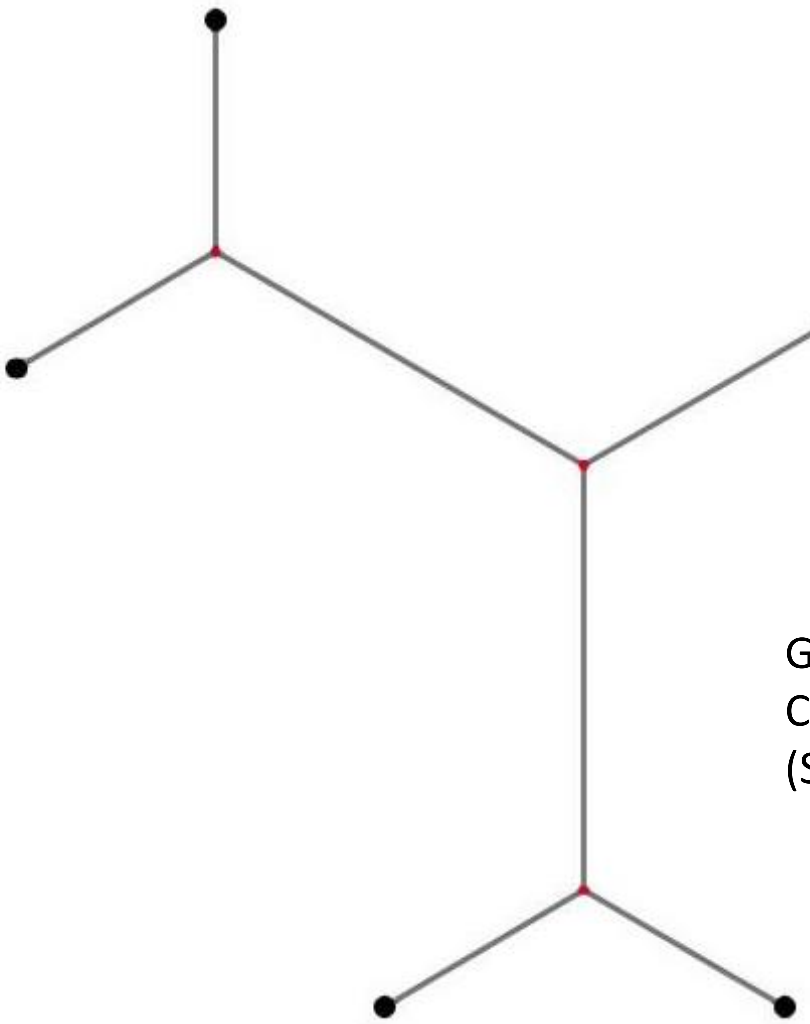
Day 3, Jyvaskyla Summer School 2021 COM3

Michael Emmerich

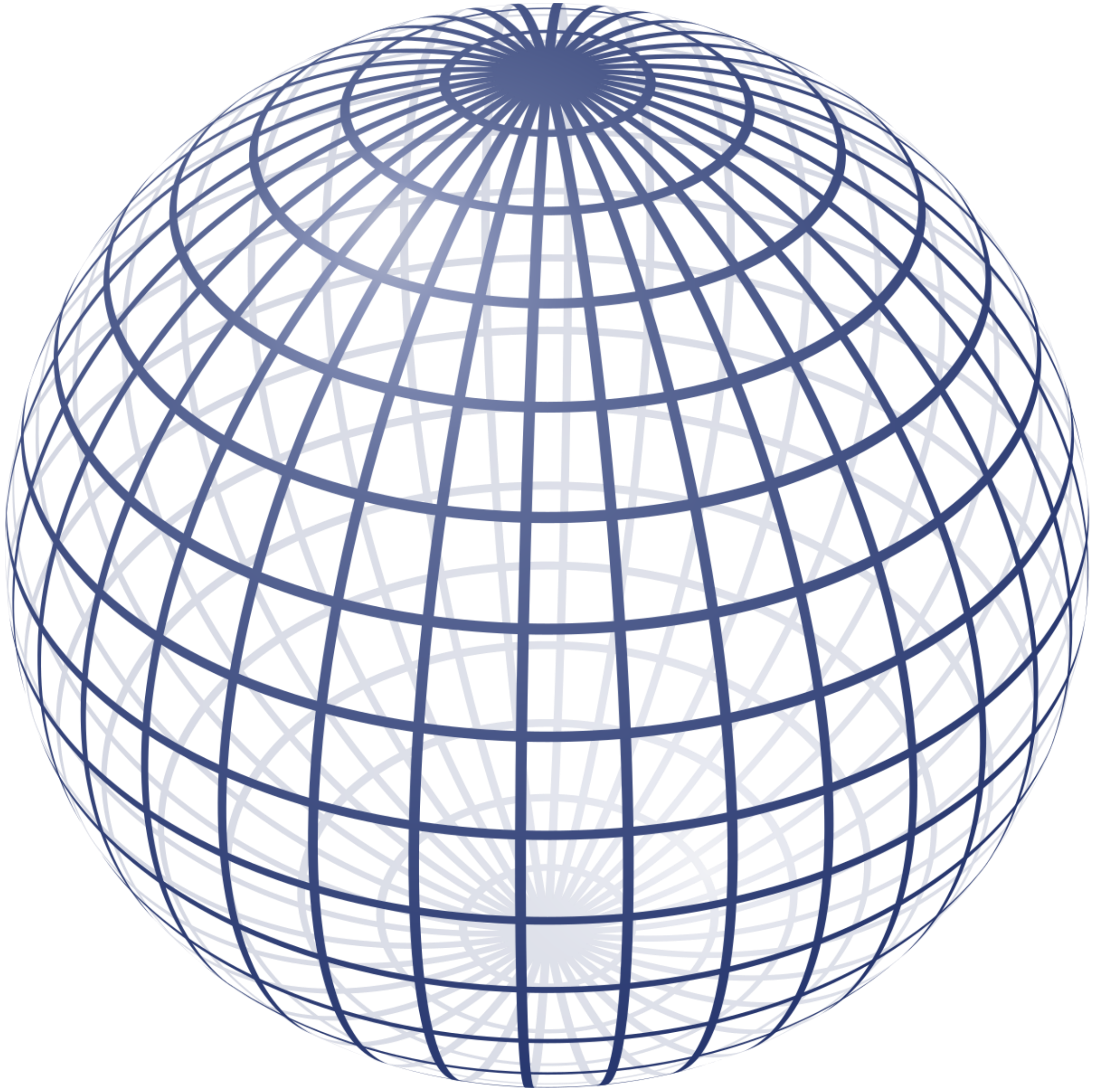
Learning Goals

- Platon’s dream: What are ideal solutions to design problems?
- Design as a discipline

Perfect Structures

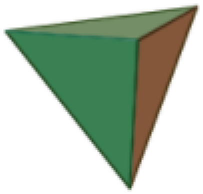
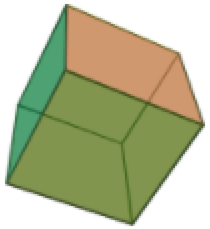
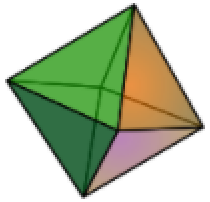
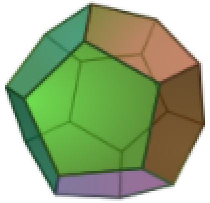
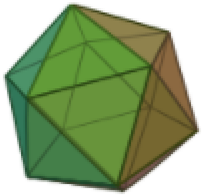


Given points
Connect with min path
(Steiner tree)



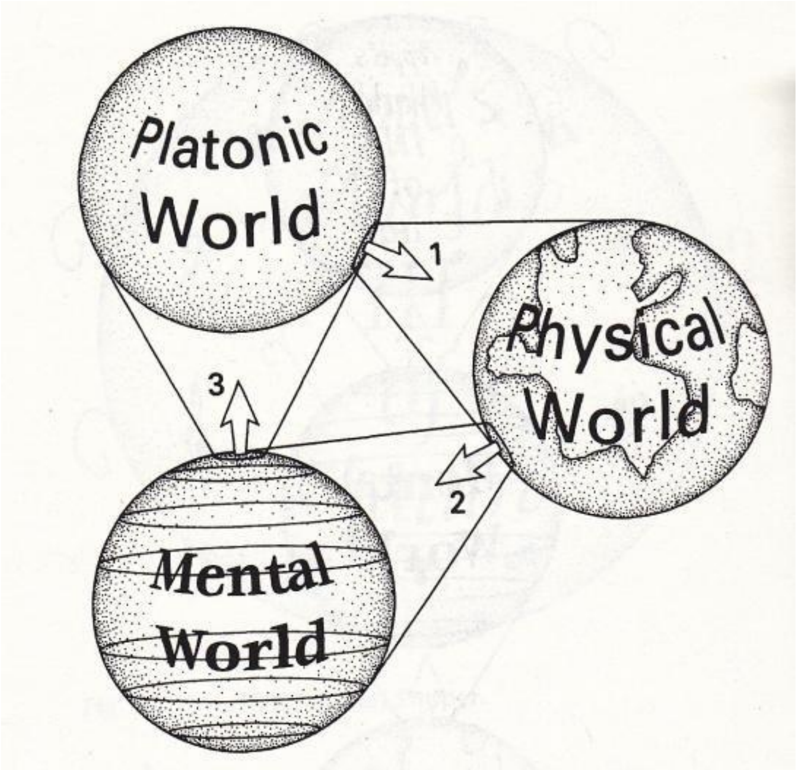
Min Surface, Volume = const

Platonic bodies

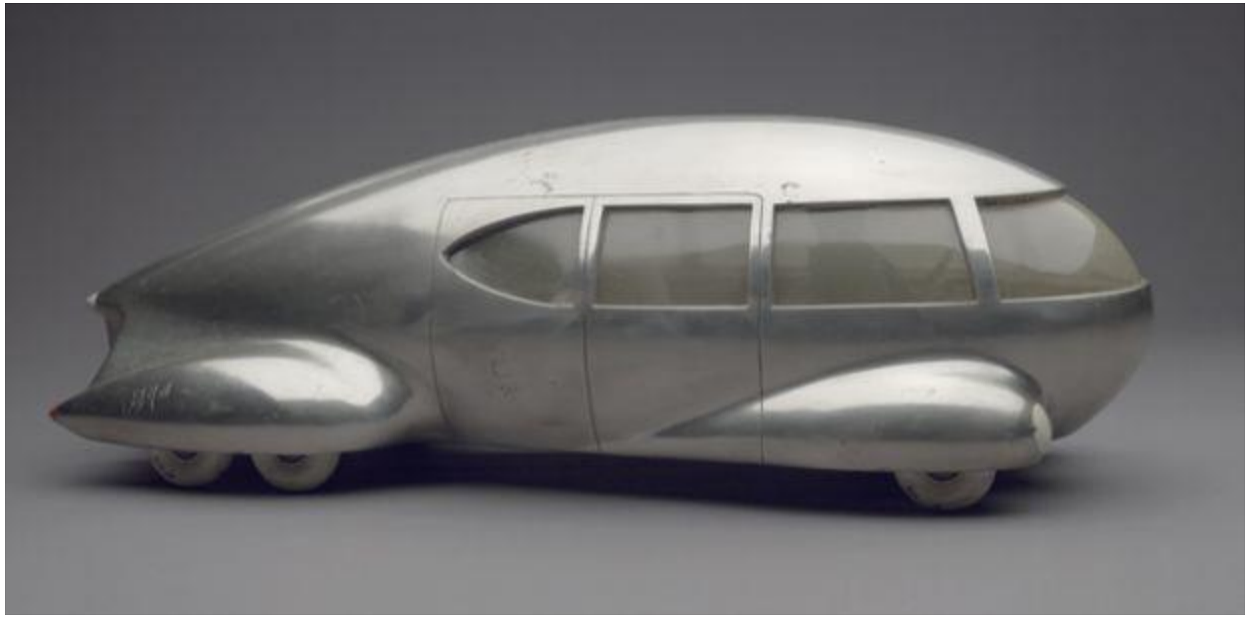
Tetrahedron	Cube	Octahedron	Dodecahedron	Icosahedron
Four faces	Six faces	Eight faces	Twelve faces	Twenty faces
				
(Animation)	(Animation)	(Animation)	(Animation)	(Animation)
(3D model)	(3D model)	(3D model)	(3D model)	(3D model)

Shapes of maximally even (equal) surfaces at their boundary

Penrose: Three worlds



Perfect Designs, $c_w = \min$

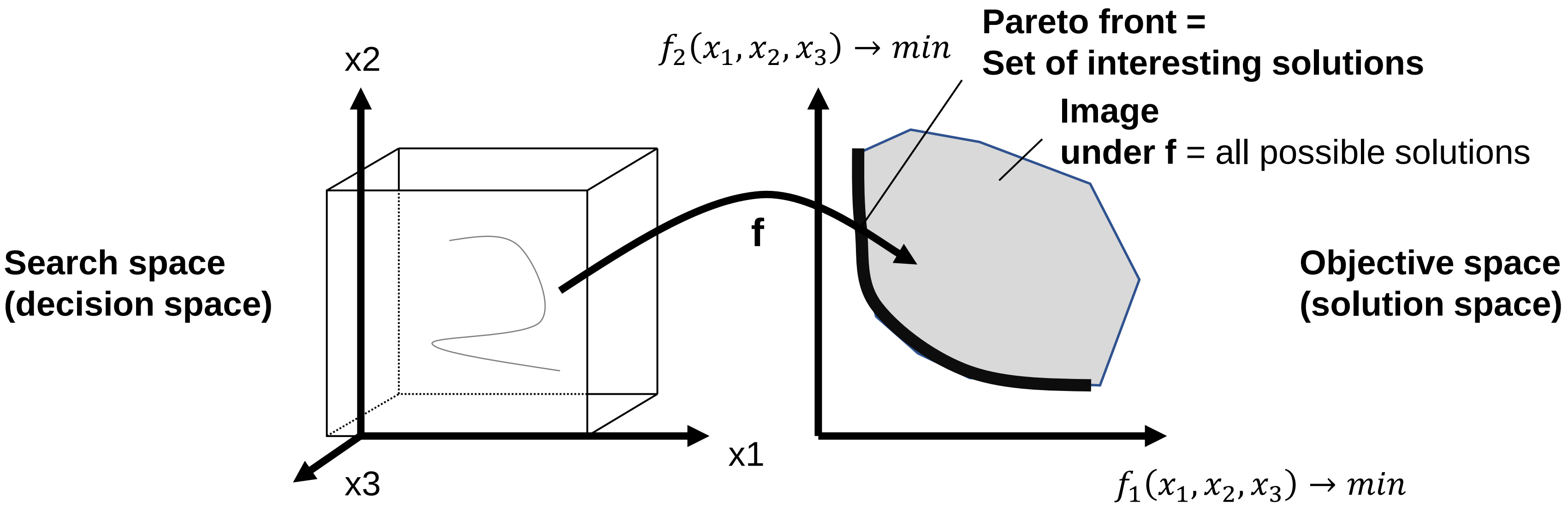


Norman Bel Geddes, "Motor Car No. 9 (without tail fin)"

Aalto shape (beauty is hard to measure …)

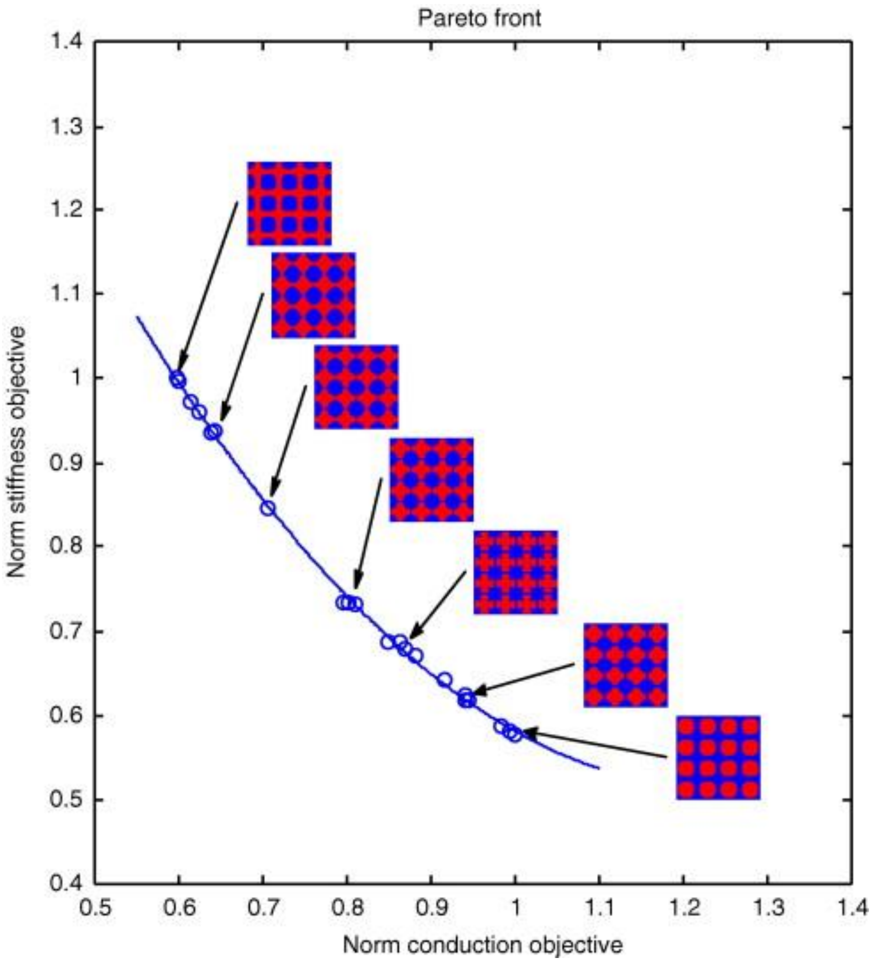
Think of examples of beautiful shapes in nature …

Pareto optimality



Research questions

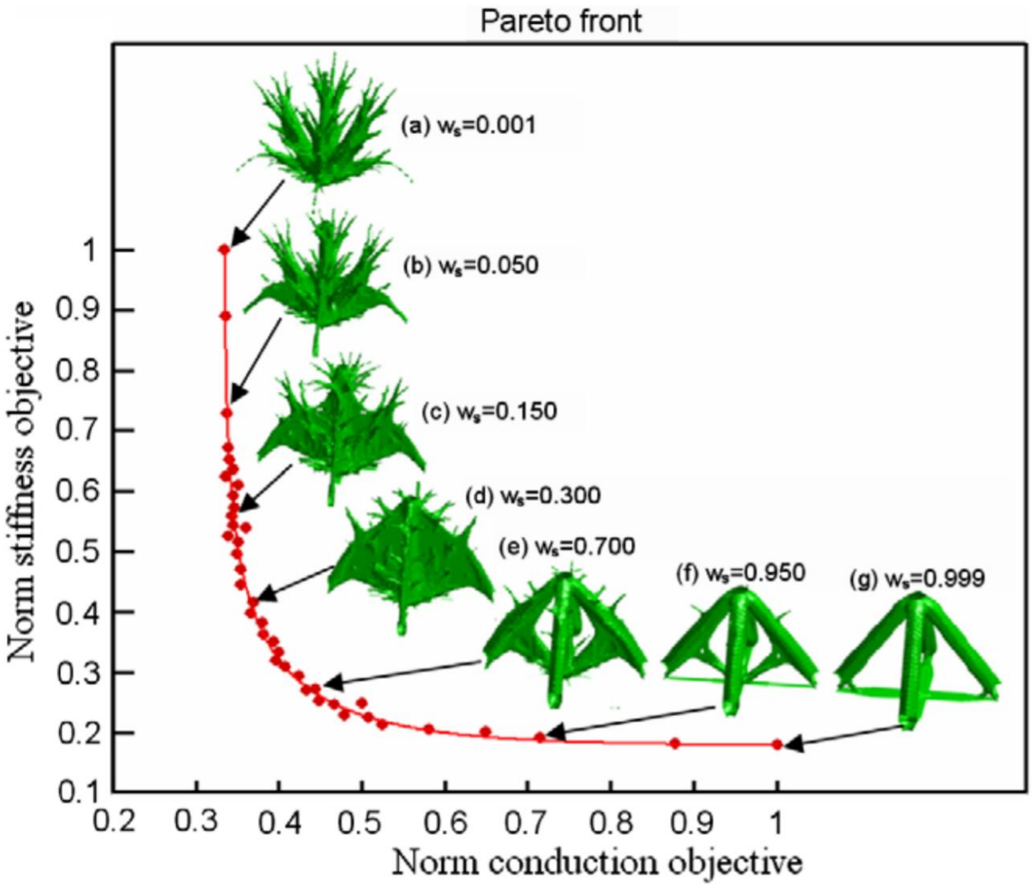
- Find ‘Pareto perfect’ structures:
    - Micro-
    - Macro-
- Efficiency
- Precision & Coverage

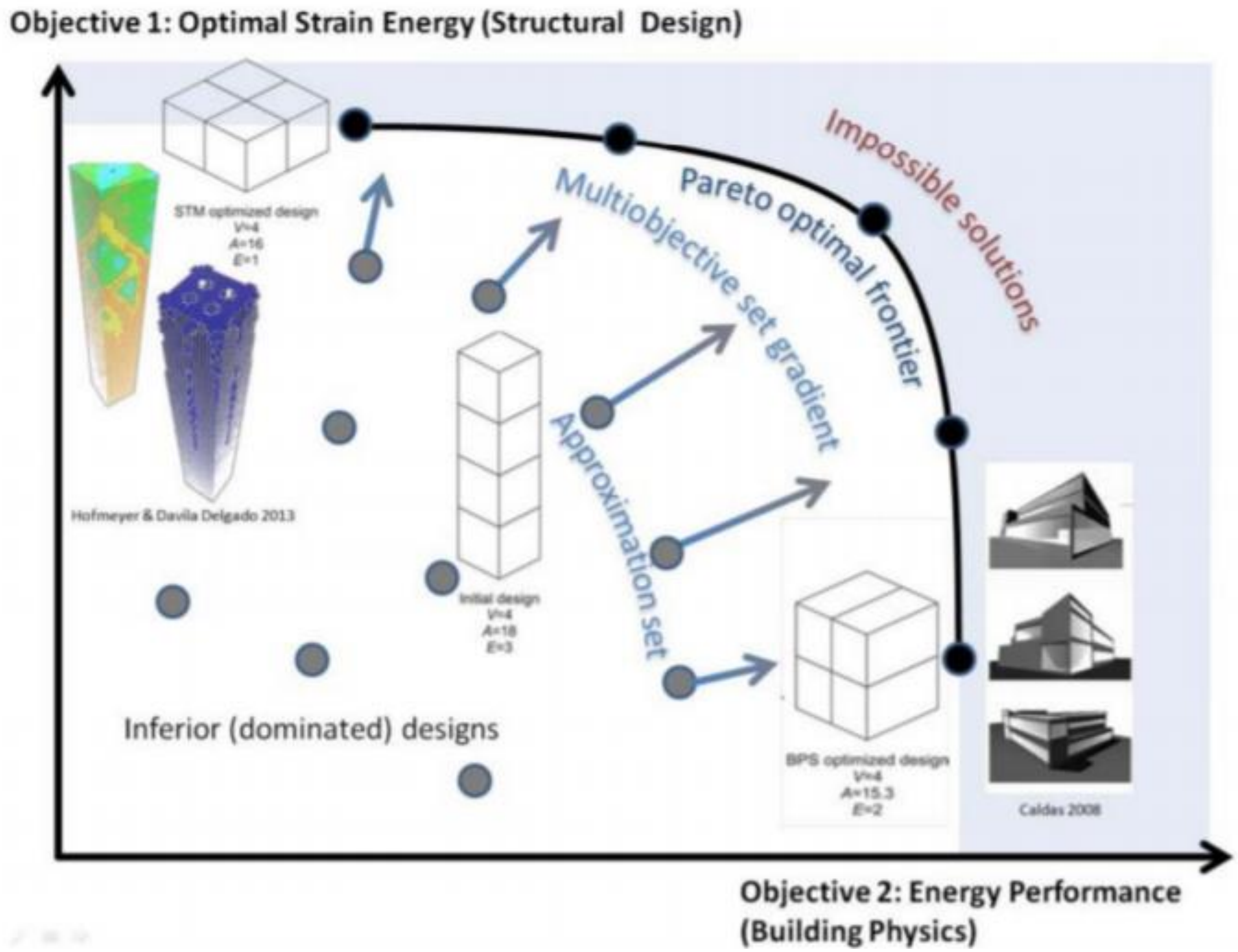


Niek de Kruijf, Shiwei Zhou, Qing Li, Yiu-Wing Mai,Topological design of structures and composite materials with multiobjectives, International Journal of Solids and Structures, Volume 44, Issues 22-23, 2007, Pages 7092-7109

‘Pareto morphing’ along the Pareto front

Chen, Yuhang, Shiwei Zhou, and Qing Li. "Multiobjective topology optimization for finite periodic structures." *Computers & Structures* 88.11-12 (2010): 806-811.




Objective 1: Optimal Strain Energy (Structural Design)
STM optimized design
V=4
A=16
E=1
Hofmeyer & Davila Delgado 2013
Impossible solutions
Pareto optimal frontier
Multiobjective set gradient
Approximation set
Initial design
V=4
A=18
E=3
Inferior (dominated) designs
BPS optimized design
V=4
A=15.3
E=2
Caldas 2008
Objective 2: Energy Performance
(Building Physics)

Question

- How can we express shapes by means of decision variables?
- How can we make sure that constraints are kept?

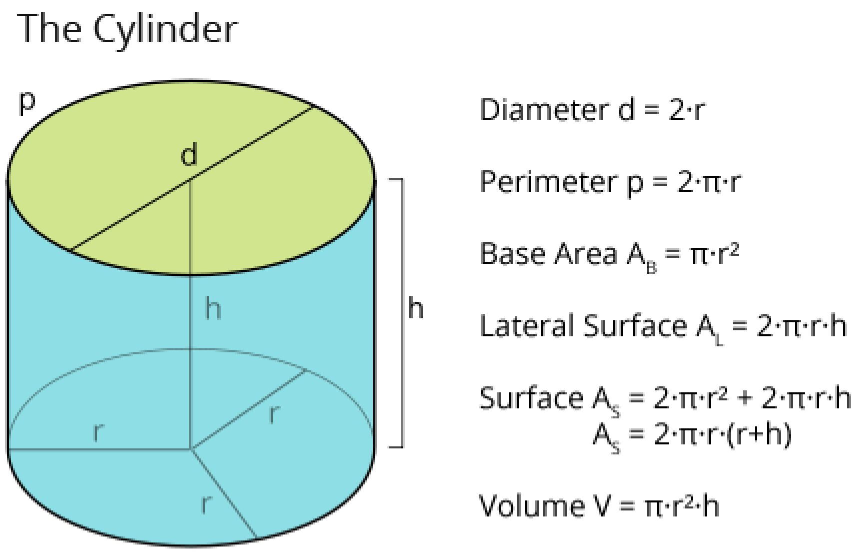
Example: Cylinder (tin)

Volume → max

Surface → min

Parameters: r, h

Constraints: r > 0, h > 0



Solution

$$V(r,h) = \pi r^2 h = \epsilon \Rightarrow h = \frac{\epsilon}{\pi r^2}$$

$$A(r,h) = 2\pi r^2 + 2\pi r h$$

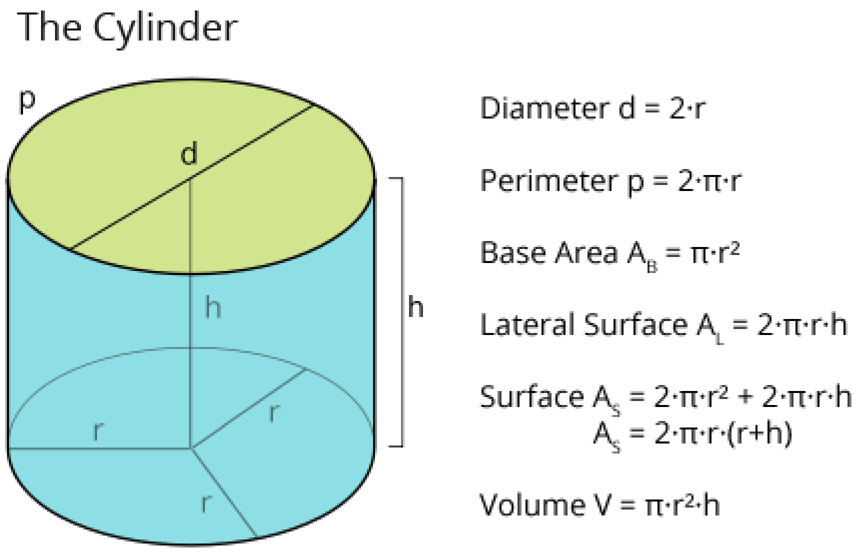
$$= 2\pi r^2 + \frac{2\pi r}{\pi r^2} = 2\pi r^2 + \frac{2}{r} \rightarrow min$$

$$\nabla A(r) = \frac{dA(r)}{dr} = 4\pi r - \frac{2\epsilon}{r^2} = 0$$

$$\Leftrightarrow r^3 = \frac{2\epsilon}{4\pi} \Leftrightarrow r = \sqrt[3]{\frac{\epsilon}{2\pi}}$$

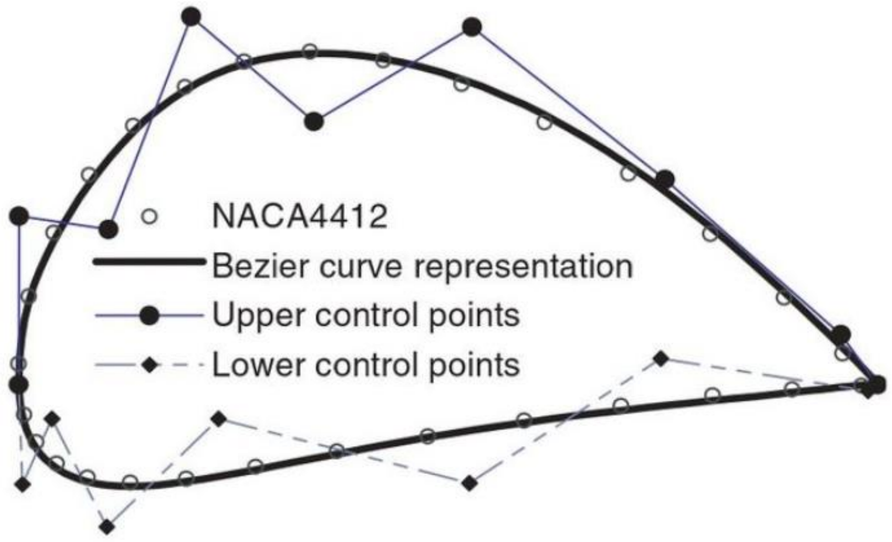
Efficient Set:

$$\mathrm{X_e} = \{(\sqrt[3]{\frac{\epsilon}{2\pi}}, \frac{\epsilon}{\pi r^2}) | \epsilon \in [0, \infty]\}$$



Design Parameterization

- Design Parameterization is the problem of describing a geometrical shape by means of continuous parameter vectors
- Examples: Bezier points, superstructures

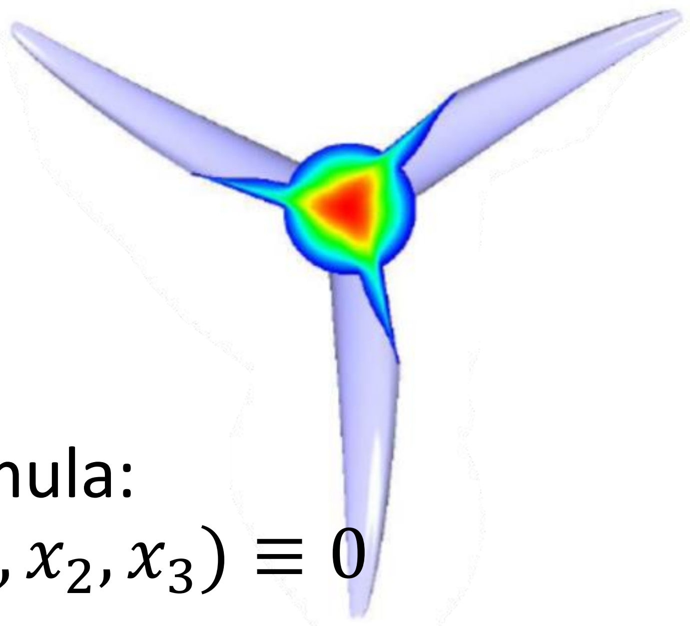


Figure

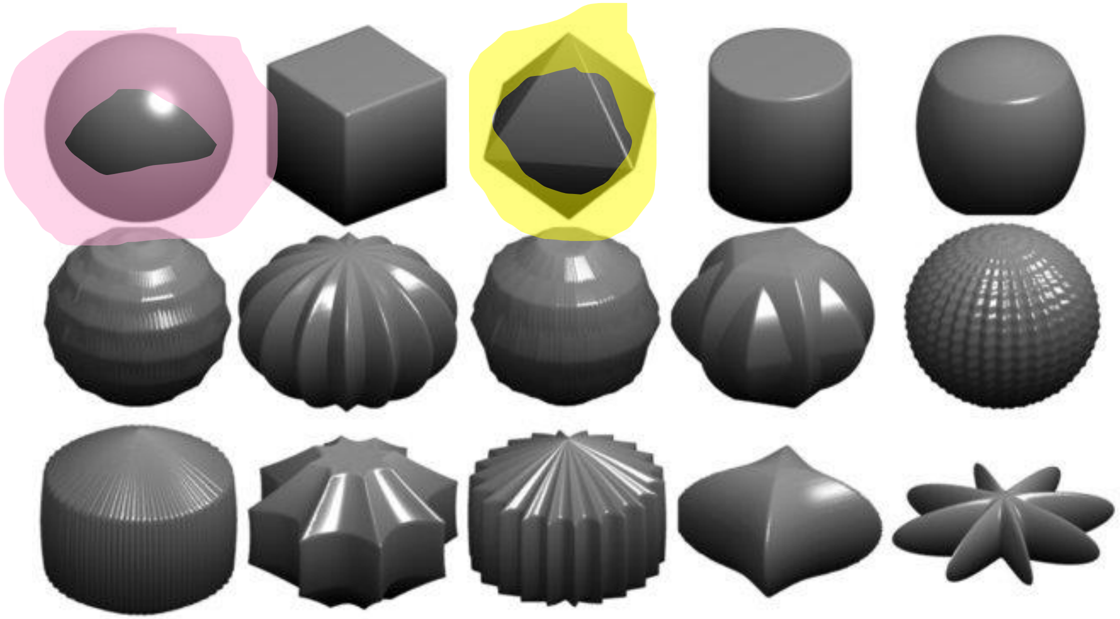
Caption

Fig. 1.8: Bezier curves representation of NACA4412 airfoil [36]

https://arxiv.org/pdf/1204.4107.pdf



Gielis' superformula



$$r(\varphi) = \left(\left| \frac{\cos\left(\frac{m\varphi}{4}\right)}{a} \right|^{n_2} + \left| \frac{\sin\left(\frac{m\varphi}{4}\right)}{b} \right|^{n_3} \right)^{-\frac{1}{n_1}}.$$

- The superformula: f$(p_1, \ldots, p_n, x_1, x_2, x_3) \equiv 0$
- $x_1, \ldots x_n$: variables
- $p_1, \ldots, p_n$: parameters

- Example

$$f(x_1, x_2, x_3) = \sum |x_i|^{p_i}$$

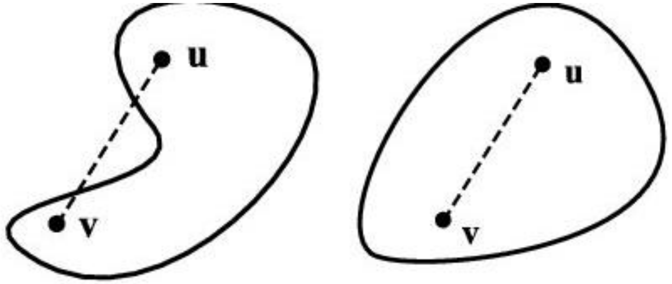
$$p_i = 1, i = 1, \ldots, 3$$

$$p_i = 2, i = 1, \ldots, 3$$

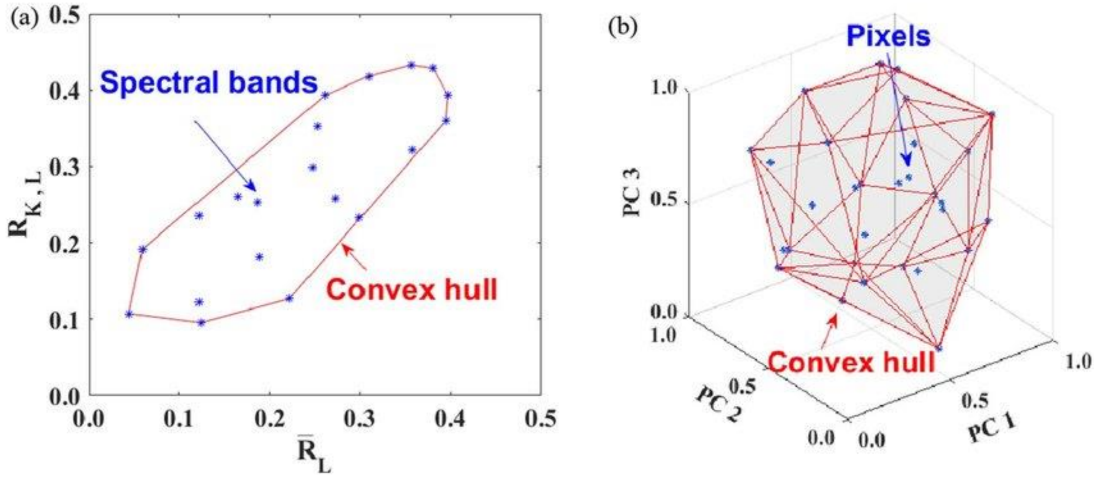
Convexity

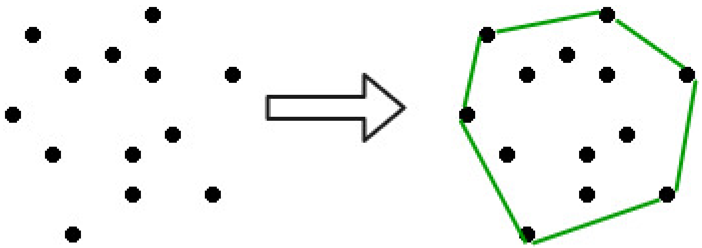
Idea
Represent
buildings as
convex shapes

- A set $S \subseteq \mathbb{R}^d$ is called a convex set if it for any two points, $u \in S$ and $\mathrm{v} \in S$ the line-segment connecting u and v is fully contained in S.
- These shapes have no-cavities (this is why buildings are often convex, excepting those with 'swimming pools' on the roof)



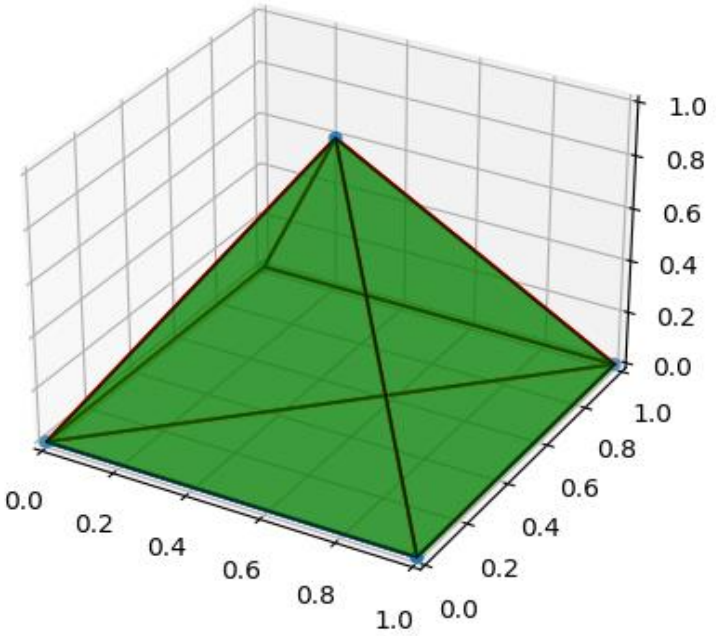
Convex hull representation

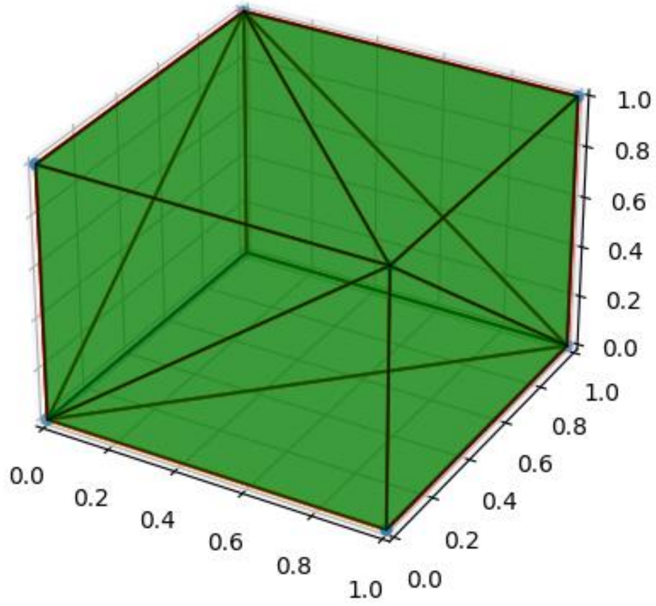




- The convex hull is the smallest convex set that contains all points
- In 2-D you can imagine a rubber-band around the set
- Convex hulls can be used to represent the set of all convex shapes by means of point sets
- ‘Active’ points form corners of this sets. Inactive points are redundant and can be removed.

Design optimization – 3D Shapes



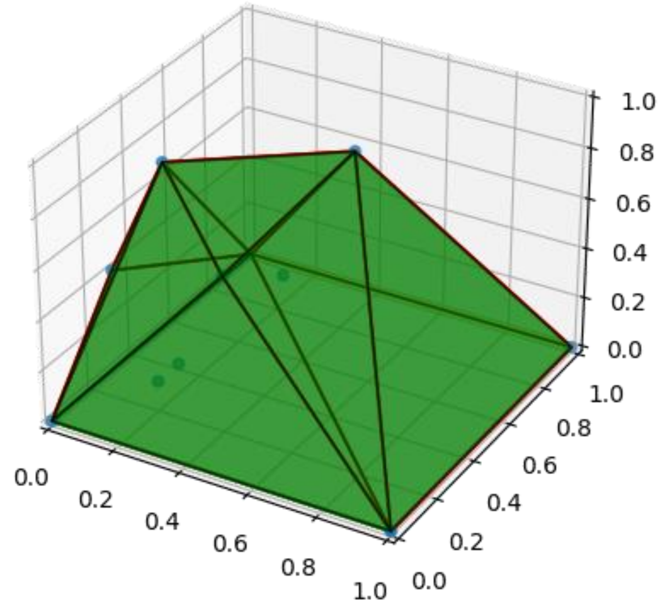



```
# Pyramid example
print("Making a pyramid")
# Define the points first
pyramid_points = np.array([
    [0,0,0], [1,0,0], [0,1,0], [1,1,0], # floor corners
    [.5, .5, 1] # pyramidion / capstoneS
])
```

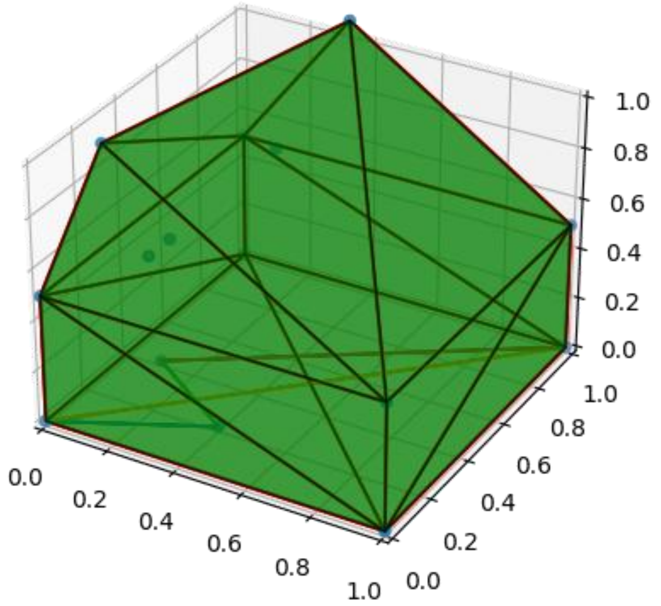
```
# Box example:
print("Making a box")
# Define the points
box_points = np.array([
    [0,0,0], [1,0,0], [0,1,0], [1,1,0], # floor corners
    [0,0,1], [1,0,1], [0,1,1], [1,1,1] # Ceiling/roof corners
])
```

Walls, floors, and roofs

```
random_points = np.array([
    [0,0,0], [0,1,0],[1,1,0], [1,0,0], # floor
    [0, 0.3, 0.4+0.5], [0.4, 0.2, 0.6+0.5], [0.1, 0.4, 0.8],
    [0.4, 0.5, 0.4], [.5, 0.7, 0.8],
    [.2, 0.3, 0.1], [.2, 0.2, 0.1]
])
box3 = Tent(random_points)
box3.plot()
```




```
random_points = np.array([
    [0,0,0], [0,1,0],[1,1,0], [1,0,0], # floor
    [0, 0, 0.5], [0, 1, 0.5], [1, 1, 0.5],[1, 0, 0.5],  # floor+0.5
    [0, 0.3, 0.4+0.5], [0.4, 0.2, 0.6+0.5], [0.1, 0.4, 0.8+0.5],
    [0.4, 0.5, 0.4+0.5], [.5, 0.7, 0.8+0.5],
    [.2, 0.3, 0.1+0.5], [.2, 0.2, 0.1+0.5]
])
box3 = Tent(random_points)
box3.plot()
```



Two bar truss problem

- Decision variables: height (H), diameter (d), thickness (t), distance (B), modulus of elasticity (E), and material density (ρ)
- Objective functions (minimize):

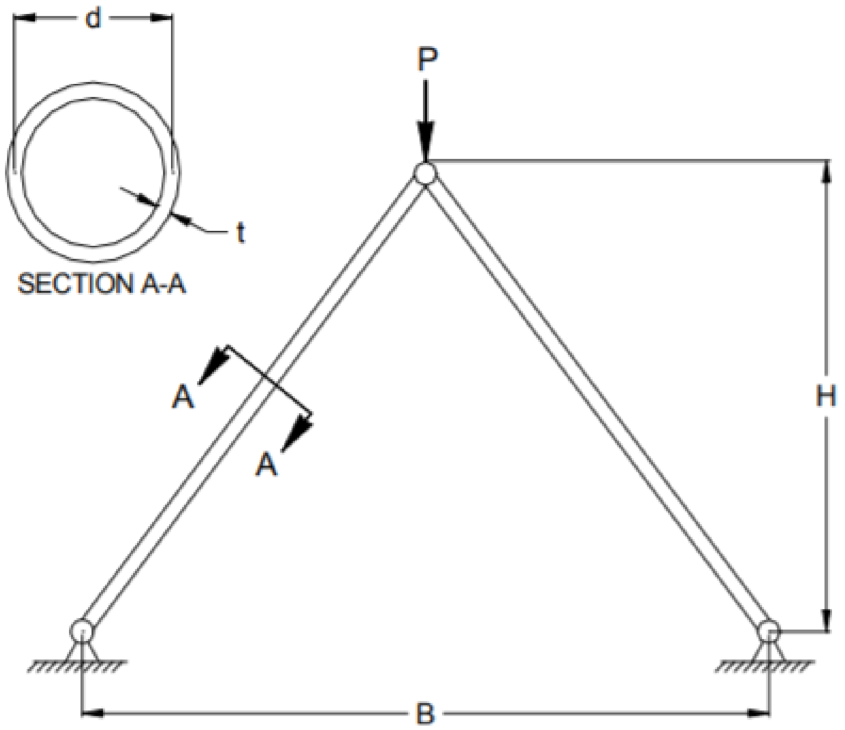


Fig. 1.1 - Layout for the Two-bar truss model.

$$Weight = \rho \cdot 2 \cdot \pi \cdot d \cdot t \sqrt{\left(\frac{B}{2}\right)^2 + H^2}$$

$$Buckling\ Stress = \frac{\pi^2 E(d^2 + t^2)}{8[\left(\frac{B}{2}\right)^2 + H^2]}$$

$$Stress = \frac{P \cdot \sqrt{\left(\frac{B}{2}\right)^2 + H^2}}{2 \cdot t \cdot \pi \cdot d \cdot H}$$

$$Deflection = \frac{P \cdot \left[\left(\frac{B}{2}\right)^2 + H^2\right]^{(3/2)}}{2 \cdot t \cdot \pi \cdot d \cdot H^2 \cdot E}$$

More data (example point)

Analysis Variables	Value
Height, H (in)	30.
Diameter, d (in)	3.
Thickness, t (in)	0.15
Separation distance, B (inches)	60.
Modulus of elasticity (1000 lbs/in^2)	30,000
Density, ρ (lbs/in^3)	0.3
Load (1000 lbs)	66
Analysis Functions	**Value**
Weight (lbs)	35.98
Stress (ksi)	33.01
Buckling stress (ksi)	185.5
Deflection (in)	0.066

```python
# Defining (decision) variables
# Height, diameter, thickness, Seperation distance, modulus of elasticity, density, load
var_names = ["H", "d", "t", "B", "E", "p"]
var_count = len(var_names)

initial_values = np.array([30.0, 3.0, 0.15, 60.0, 30000., 0.3])

# set lower bounds for each variable
lower_bounds = np.array([20.0, 0.5, 0.01, 20.0, 25000., 0.01])

# set upper bounds for each variable
upper_bounds = np.array([60.0, 5.0, 1.0, 100.0, 40000., 0.5])

# Trying to minimize everything so no need to define minimize array

# Create a list of Variables for MOProblem class
variables = variable_builder(var_names, initial_values, lower_bounds, upper_bounds)

# Define objectives
obj1 = _ScalarObjective("Weight", weight)
obj2 = _ScalarObjective("Stress", stress)
obj3 = _ScalarObjective("Buckling stress", buckling_stress)
obj4 = _ScalarObjective("Deflection", deflection)

# In this example we are minimizing all the objectives subject to buckling stress < 250 and deflection < 1

# List of objectives for MOProblem class
objectives = np.array([obj1, obj2, obj3, obj4])[obj_mask]
objectives_count = len(objectives)
```

The python examples …

```python
# a constant load value for the problem
load = 66
# Which objectives do you wish to optimize
# weight, stress, buckling stress and deflection
obj = np.array([
    True, True, True, True, # Optimizing all
])
# Approximate ideal and nadir for a problem with no constraints
# nadir = 573, 2950, 535, 9
# ideal = 0.01, 2.1, 1.5, 0.01

# Set constraint for objectives, [lower, upper]
# If no constraint then set it to None
# Notice that breaking constraints will result in a penalty and therefore we might get results that break the cons
constraints = np.array([
    [10, 100], #  10 < weight < 100
    [15, None], # stress > 15
    [None, 100], # buckling < 100
    [None, None], # deflection no constraint
])

# To create the problem we can call the create_problem method with the parameters defined earlier
# The method returns a MOProblem and a scalarmethod instance which can be passed to different Desdeo objects
problem, method = create_problem(load, obj, constraints)

# Example on solving the pareto front : This might take some time so feel free to comment this out.

# We will use the solve_pareto_front_representation method but one can change this to something else.
# The method takes the problem instance and a step size array

# The method will create reference points from nadir to ideal with these step sizes
# large step sizes => less solutions but faster calculation
# The create_problem method below will print approximate values of the nadir and ideal
# This might help you set the step sizes to fit the problem.
step_sizes = np.array([100, 177, 100, 4])[obj]

# The method returns the decision vectors and corresponding objective vectors
var, obj = solve_pareto_front_representation(problem, step_sizes)

# save the solution if you wish, make sure to change the name to not accidentally overwrite an existing solution.
# Saved solutions can be used later to visualize it
# The solution will be saved to modules/DataAndVisualization/'name'
save("tbExample", obj, var, problem.nadir, problem.ideal)
```

- The problem needs to put constraints on the objectives that are not used in the Pareto optimization
- We have additionally created a script that allows to visualize the Pareto front

PySummerschool\modules\TwoBarTruss\problem.py

PySummerschool\createTwoBartrussProblem.py

PySummerschool\visualizeTwoBarTruss.py

To be continued …