DIGITBRAIN

Horizon 2020 Programme

Computational Assemblies: Analysis, Design, and Fabrication

Peng Song

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Ziqi Wang

ETH zürich

Marco Livesu

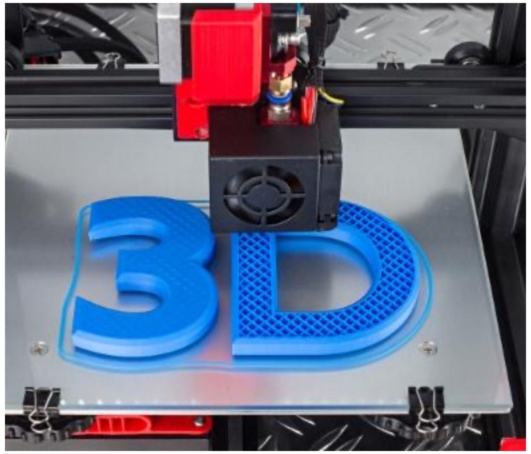
imati

Timetable

		Peng	Ziqi	Marco
Introduction	~20 mins	X		
Computational analysis of assemblies	~50 mins	X		
Computational design of assemblies	~50 mins		X	
Computational fabrication of assemblies	~50 mins			X
Q & A	~10 mins	X	X	X

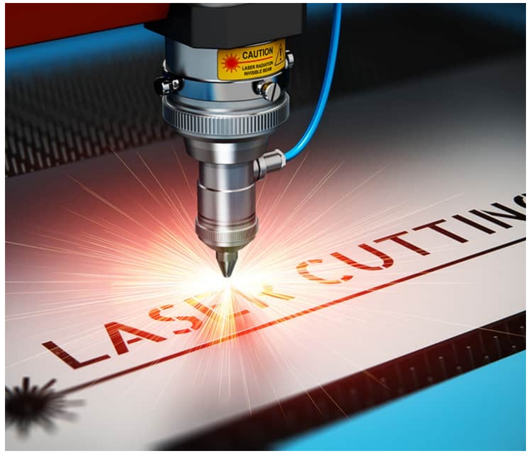
What Is This Part About?

- Shape decomposition as a mean to **overcome the limitations** of digital fabrication hardware

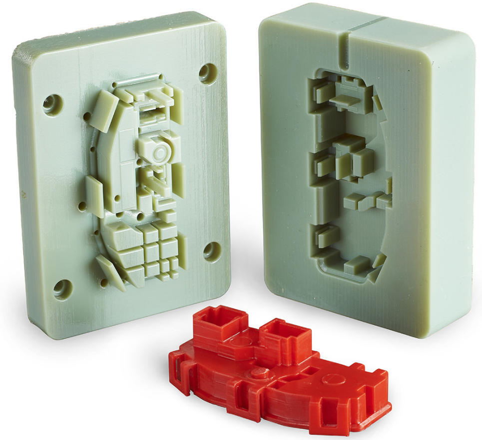


3D PRINTING

CNC MILLING



LASER CUTTING



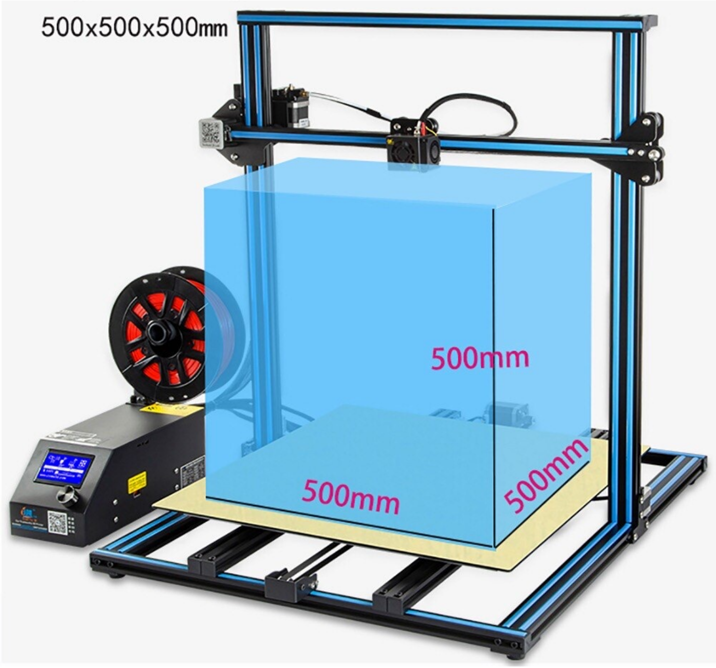
CASTING/MOLDING

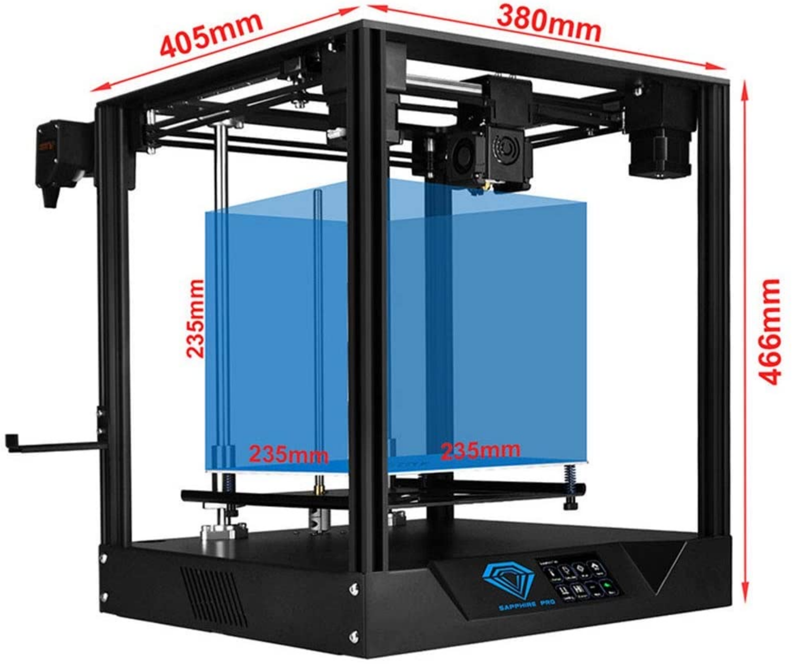
Remark #1

- Unlike previous parts, here assemblies ***are not*** an artistic/functional choice. We split shapes **because we are obliged to!**

- Algorithms must be designed to address **hardware dependent** constraints (e.g. size)

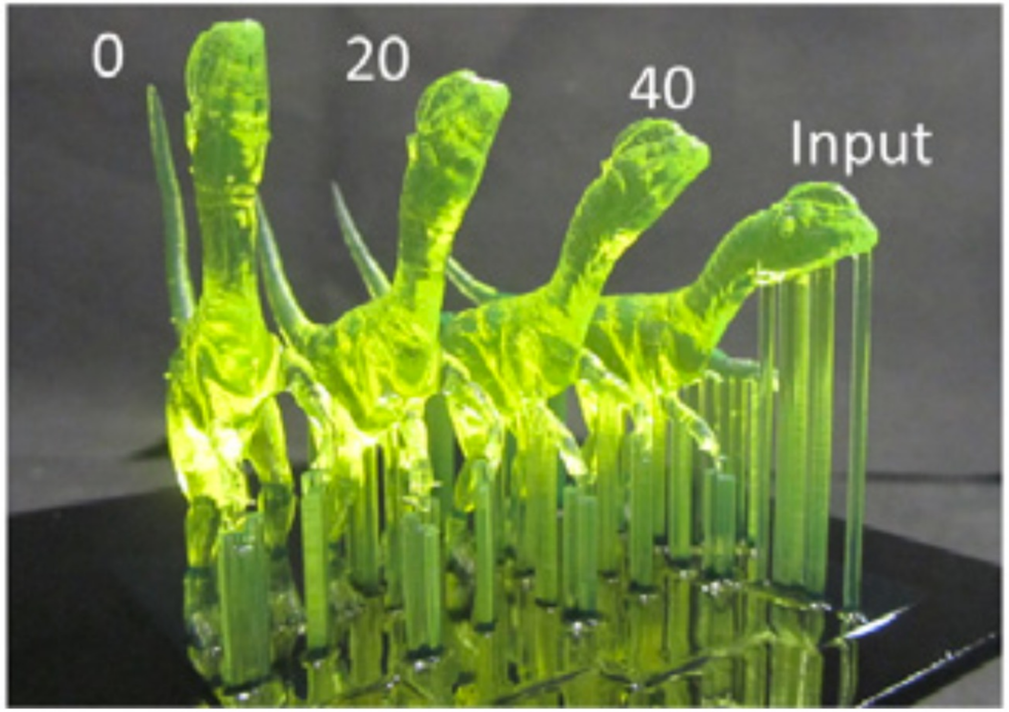
- Constraints can be either **hard** or **soft**

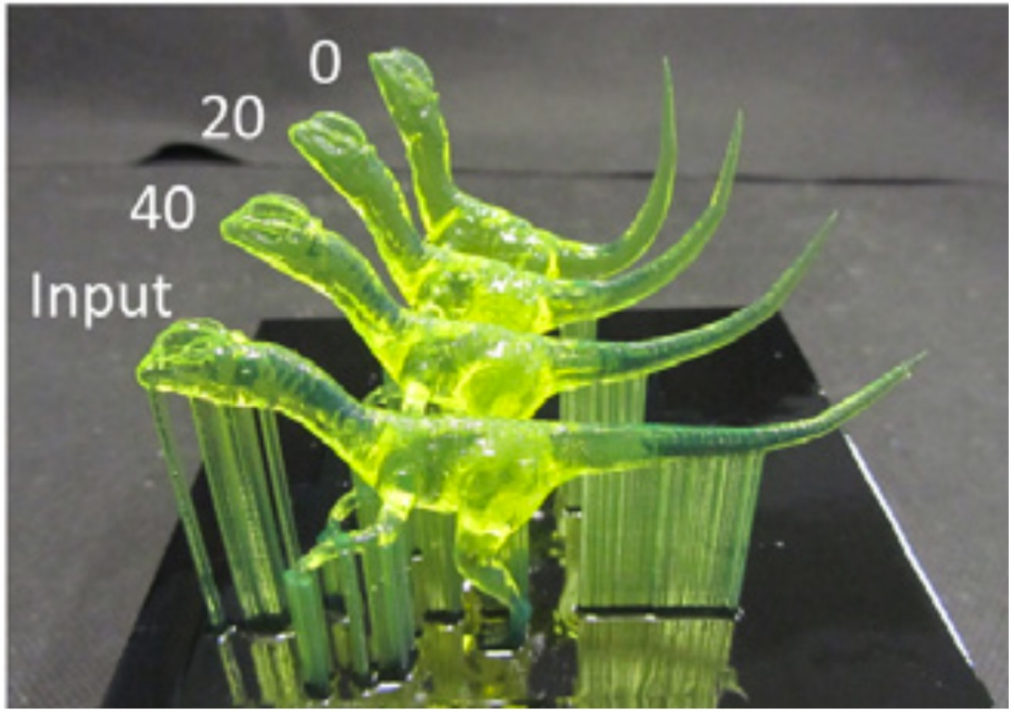




Remark #2

- Splitting is just one possible way to address manufacturing constraints
- Mesh **deformation** is a valid alternative

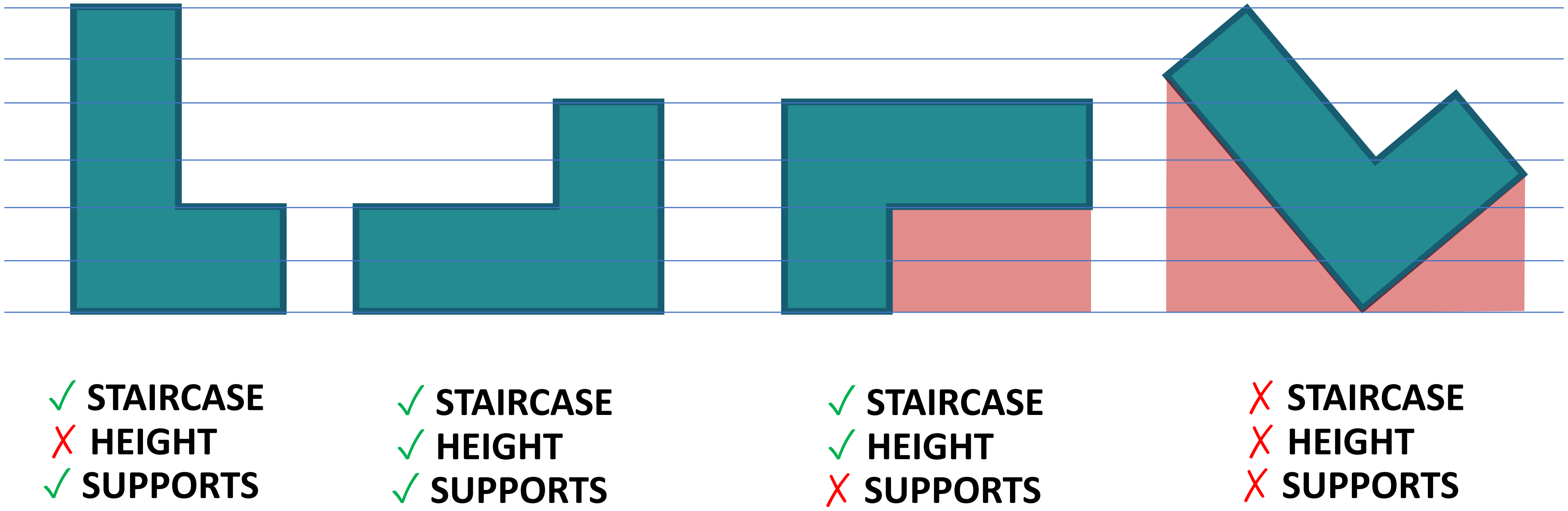




[Hu et al., CAD 2015]

Remarks

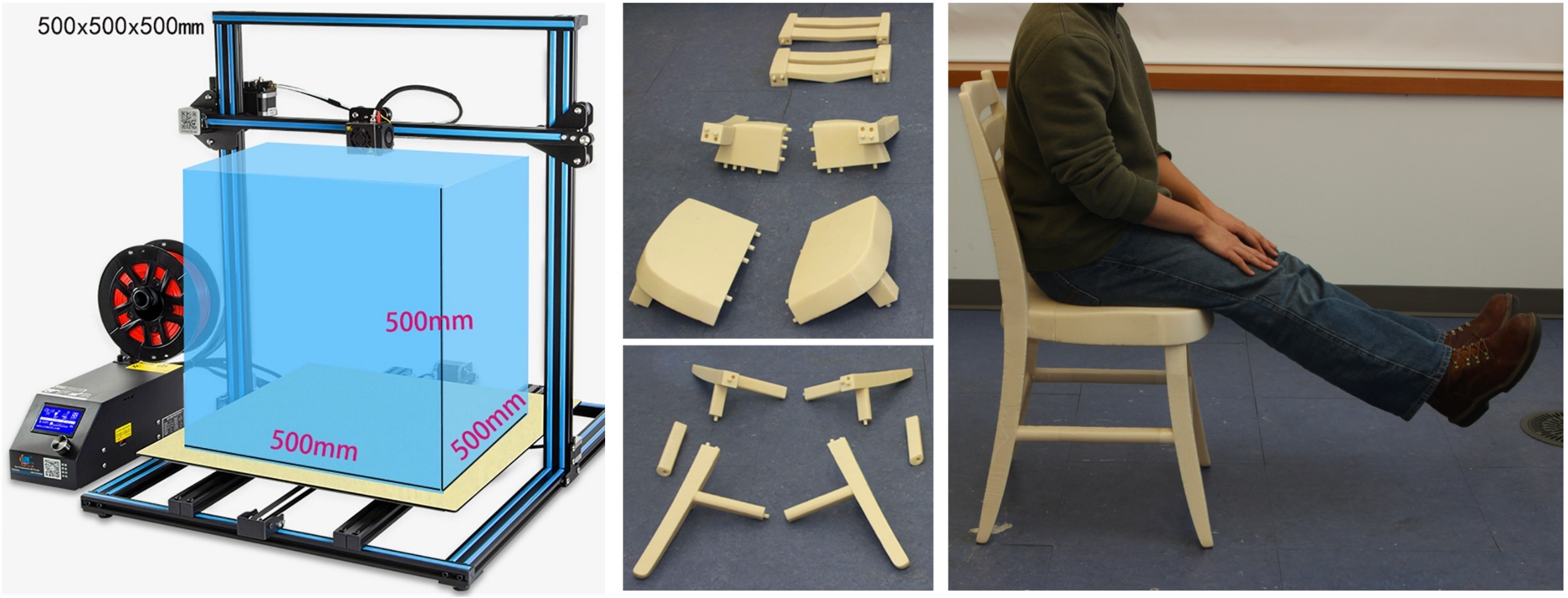
- Splitting is just one possible way to address manufacturing constraints
- Mesh **deformation** is a valid alternative
- Also **re-orientation** may be a valid alternative



[Livesu et al., EG STAR 2017]

Shape Decomposition for Manufacturing

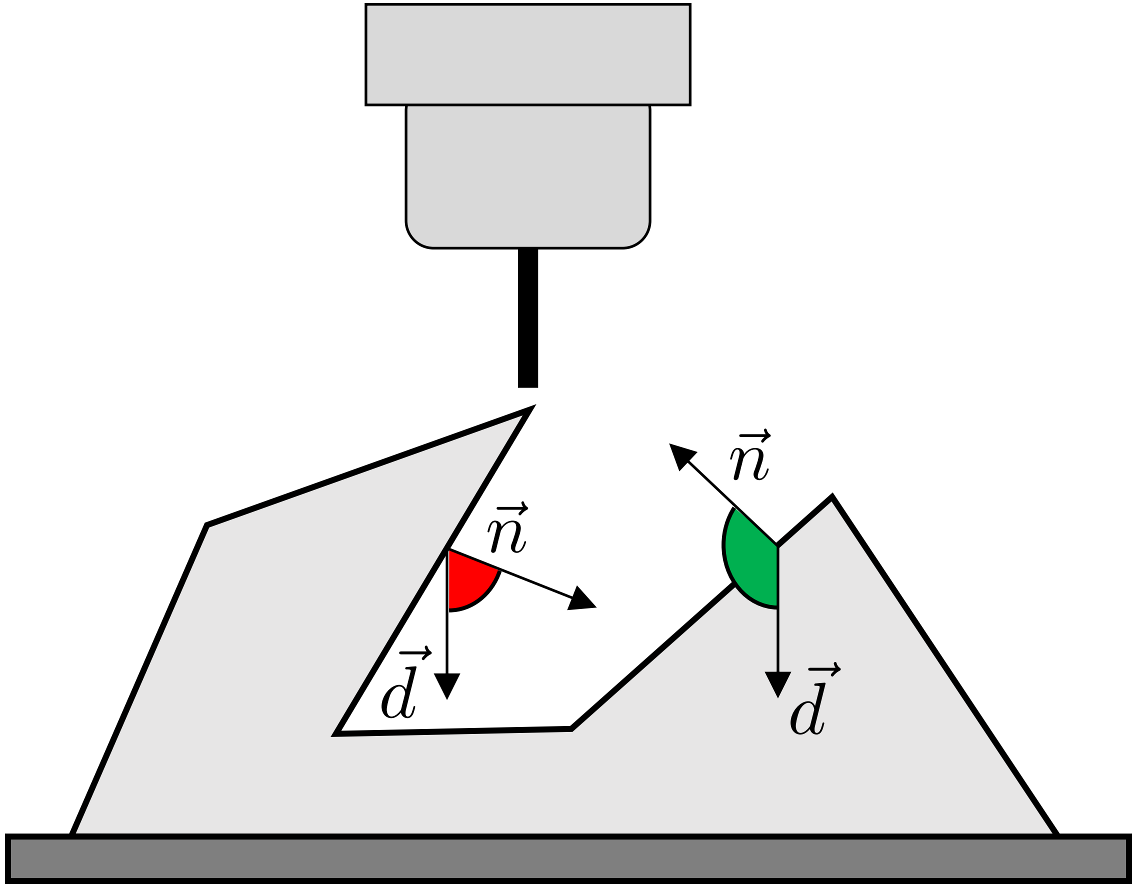
– **size**



[Luo et al., SIGGRAPH 2012]

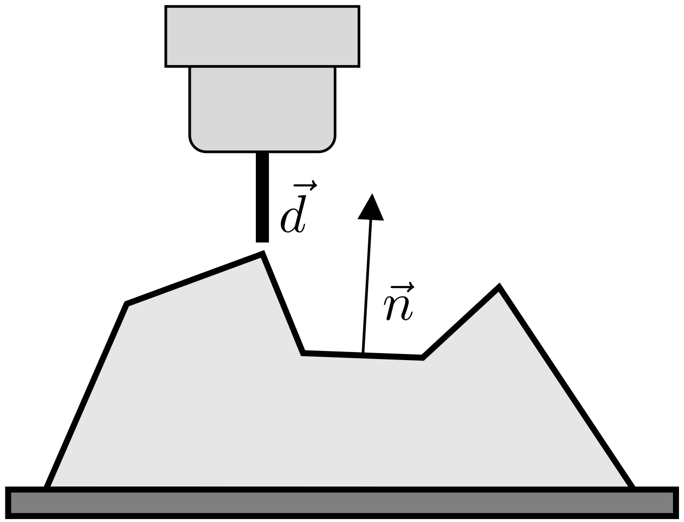
Shape Decomposition for Manufacturing

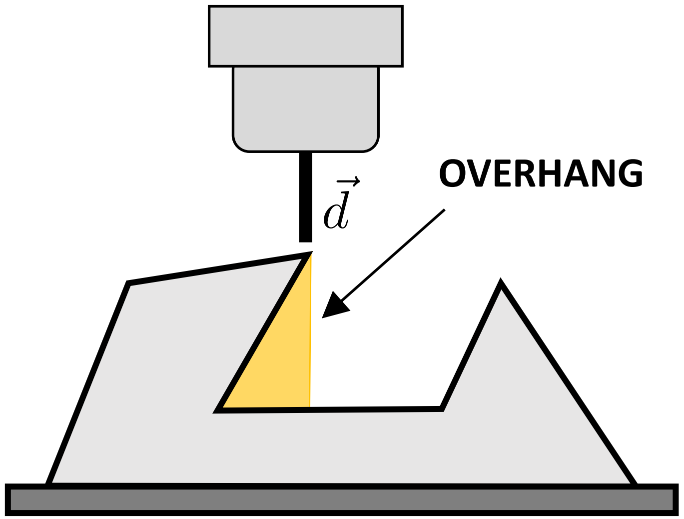
- size
- **geometry**

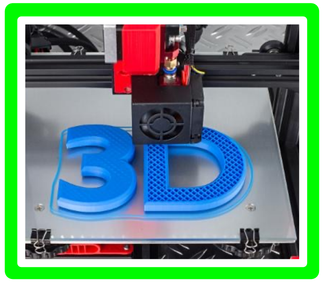


Shape Decomposition for Manufacturing

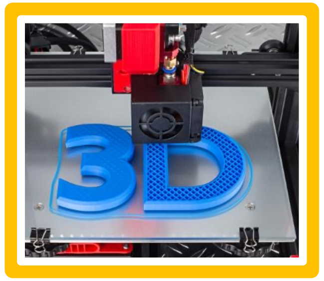
- size
- **geometry**
  - **3d printing**
  - **3 axis milling**







no supports

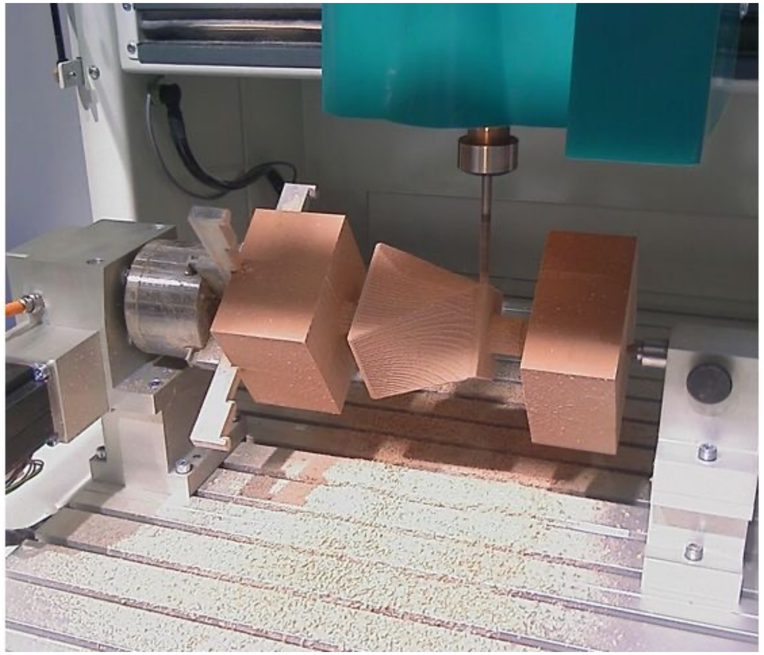


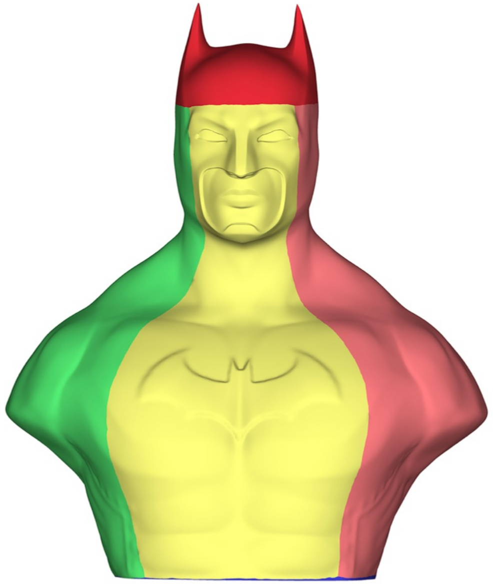
supports needed

not fabricable

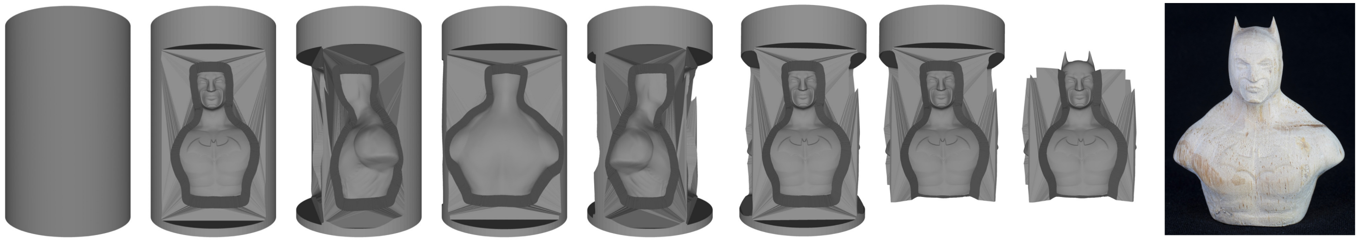
Shape Decomposition for Manufacturing

- size
- **geometry**
  - 3d printing
  - 3 axis milling
  - **4 axis milling**



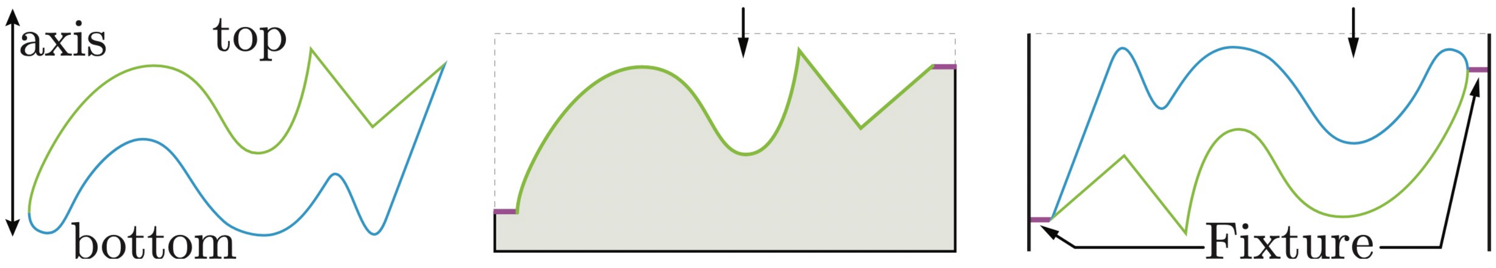


[Nuvoli et al., EG 2021]



Shape Decomposition for Manufacturing

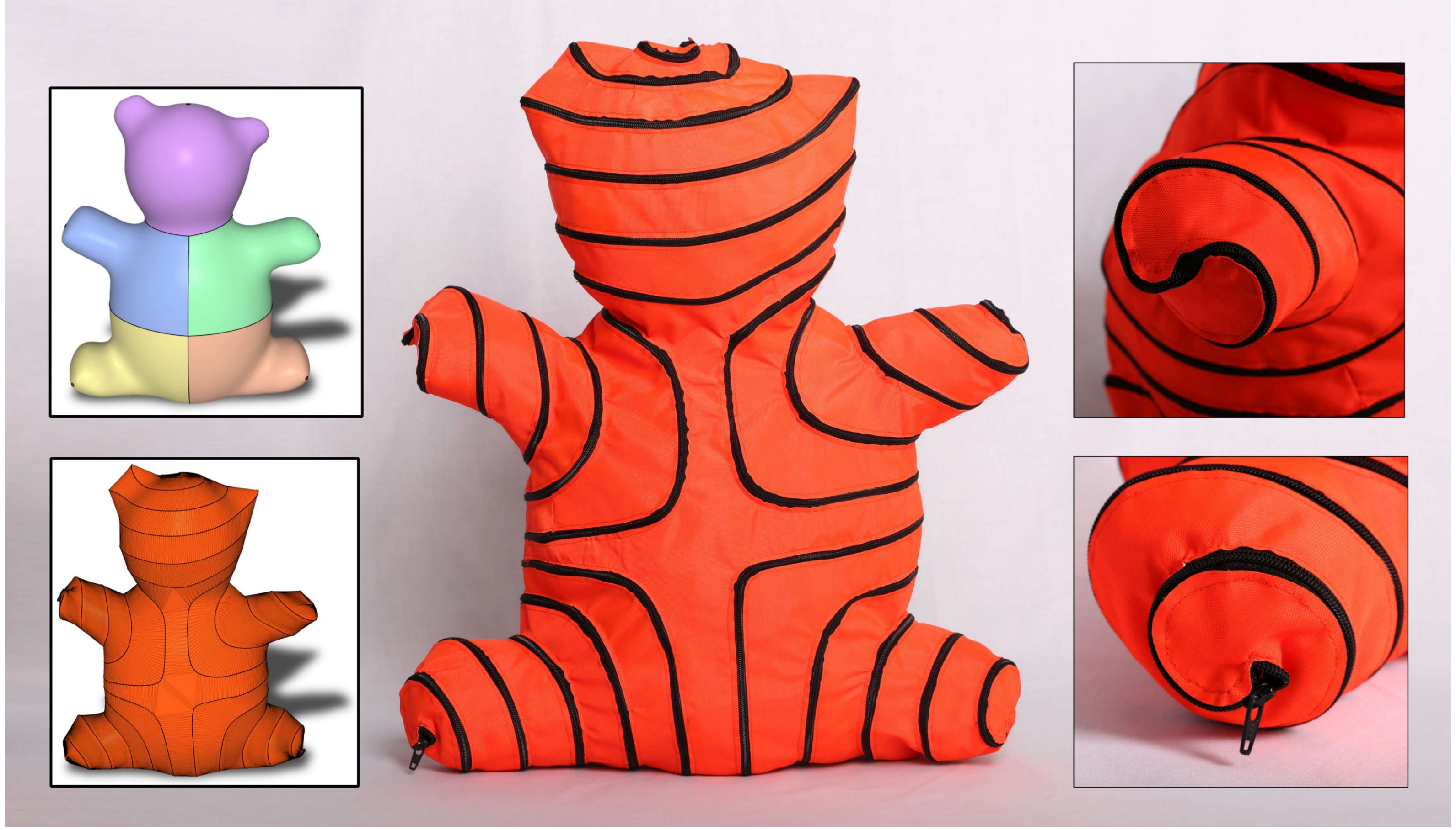
- size
- **geometry**
  - 3d printing
  - 3 axis milling
  - 4 axis milling
  - **DHF**



[Yang et al., SIGGRAPH Asia 2020]

Shape Decomposition for Manufacturing

- size
- **geometry**
  - 3d printing
  - 3 axis milling
  - 4 axis milling
  - DHF
  - **laser cutting**



[Schüller et al., SIGGRAPH 2018]

Shape Decomposition for Manufacturing

- size
- **geometry**
  - 3d printing
  - 3 axis milling
  - 4 axis milling
  - DHF
  - laser cutting
  - **rigid molding**

[Alderighi et al., SIGGRAPH 2021]

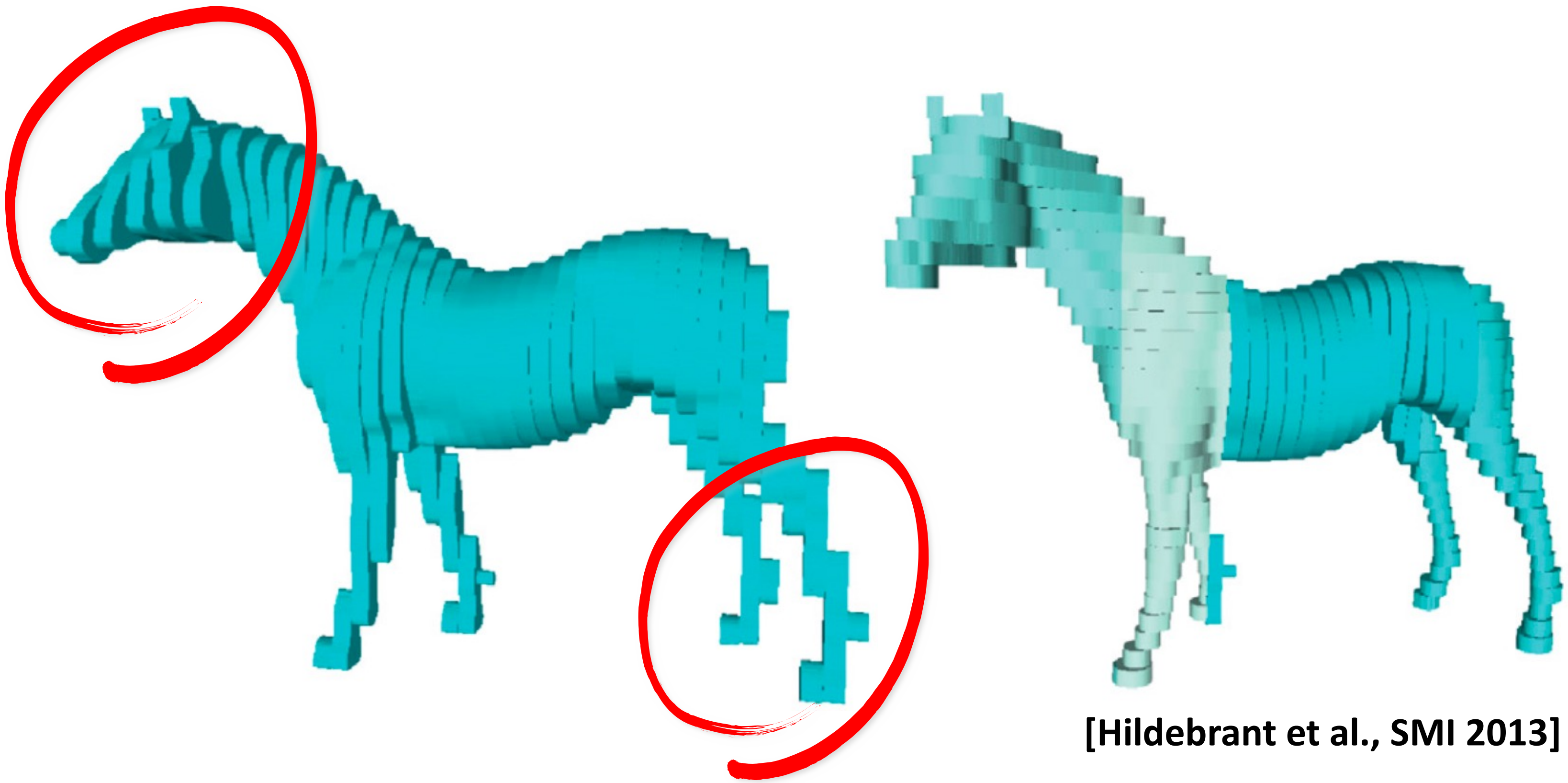
Shape Decomposition for Manufacturing

- size
- **geometry**
  - 3d printing
  - 3 axis milling
  - 4 axis milling
  - DHF
  - laser cutting
  - rigid molding
  - **flexible molding**

[Alderighi et al., SIGGRAPH 2019]

Shape Decomposition for Manufacturing

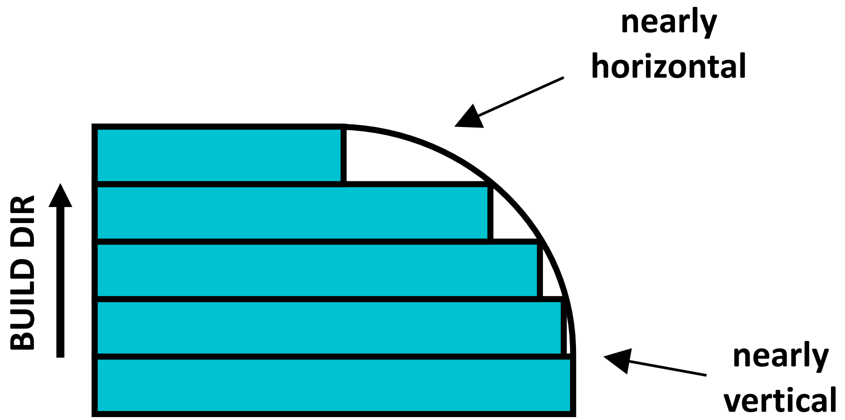
- size
- geometry
- **staircase effect**



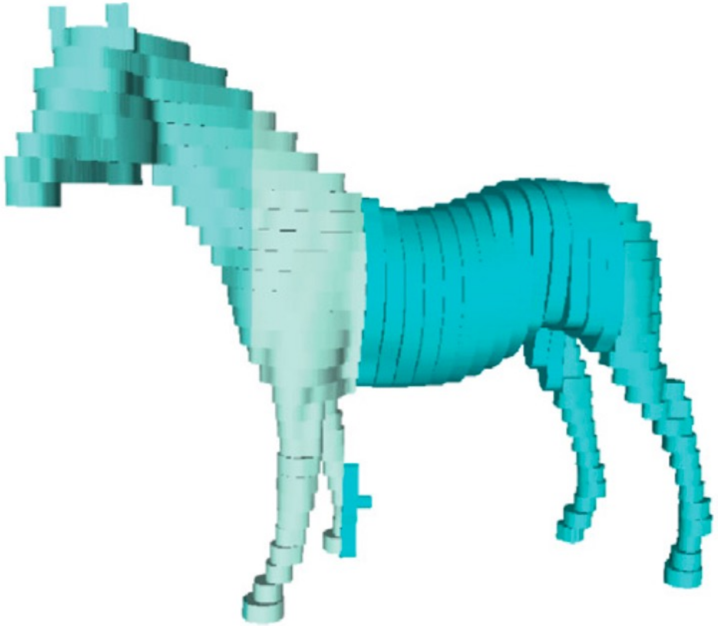
[Hildebrant et al., SMI 2013]

Shape Decomposition for Manufacturing

- size
- geometry
- **staircase effect**



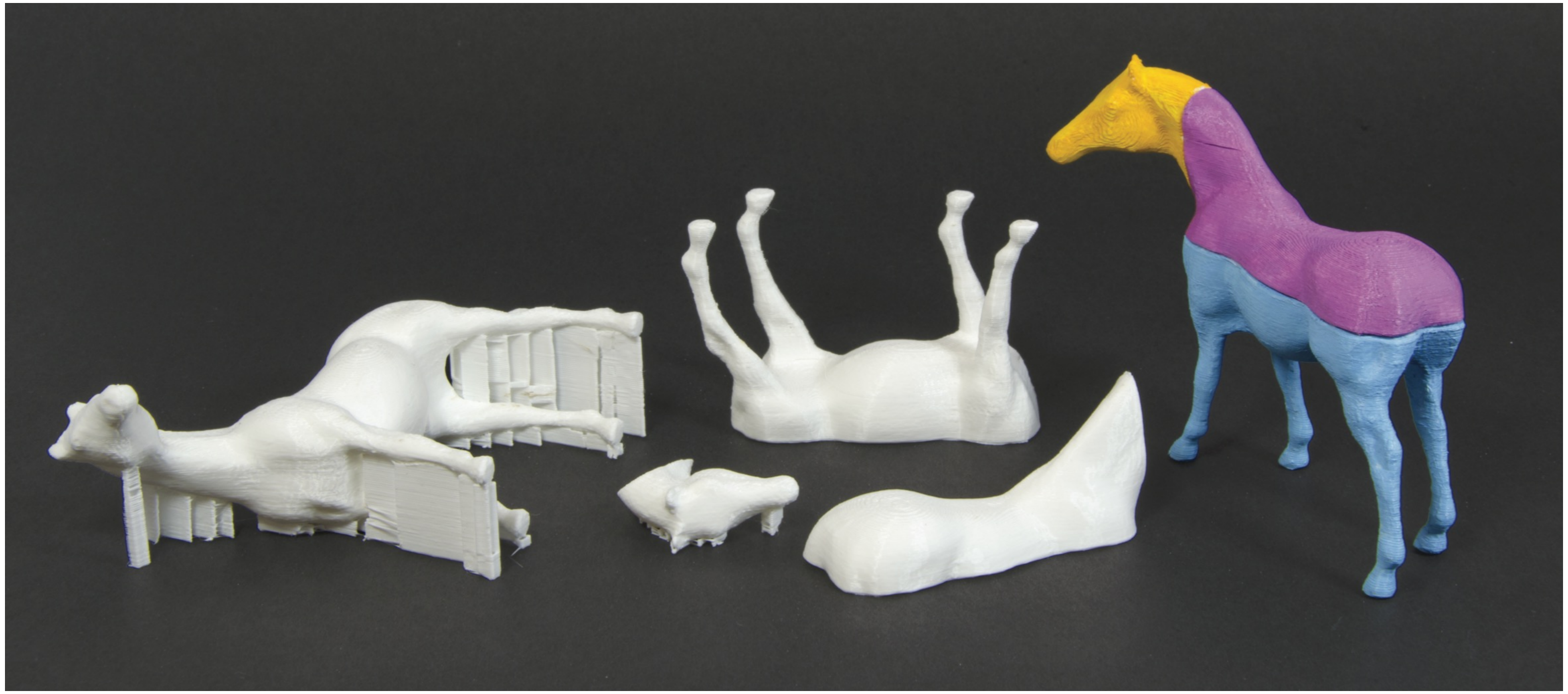
E(nearly horizontal) >> E(nearly vertical)



[Hildebrant et al., SMI 2013]

Shape Decomposition for Manufacturing

- size
- geometry
- staircase effect
- **material consumption**

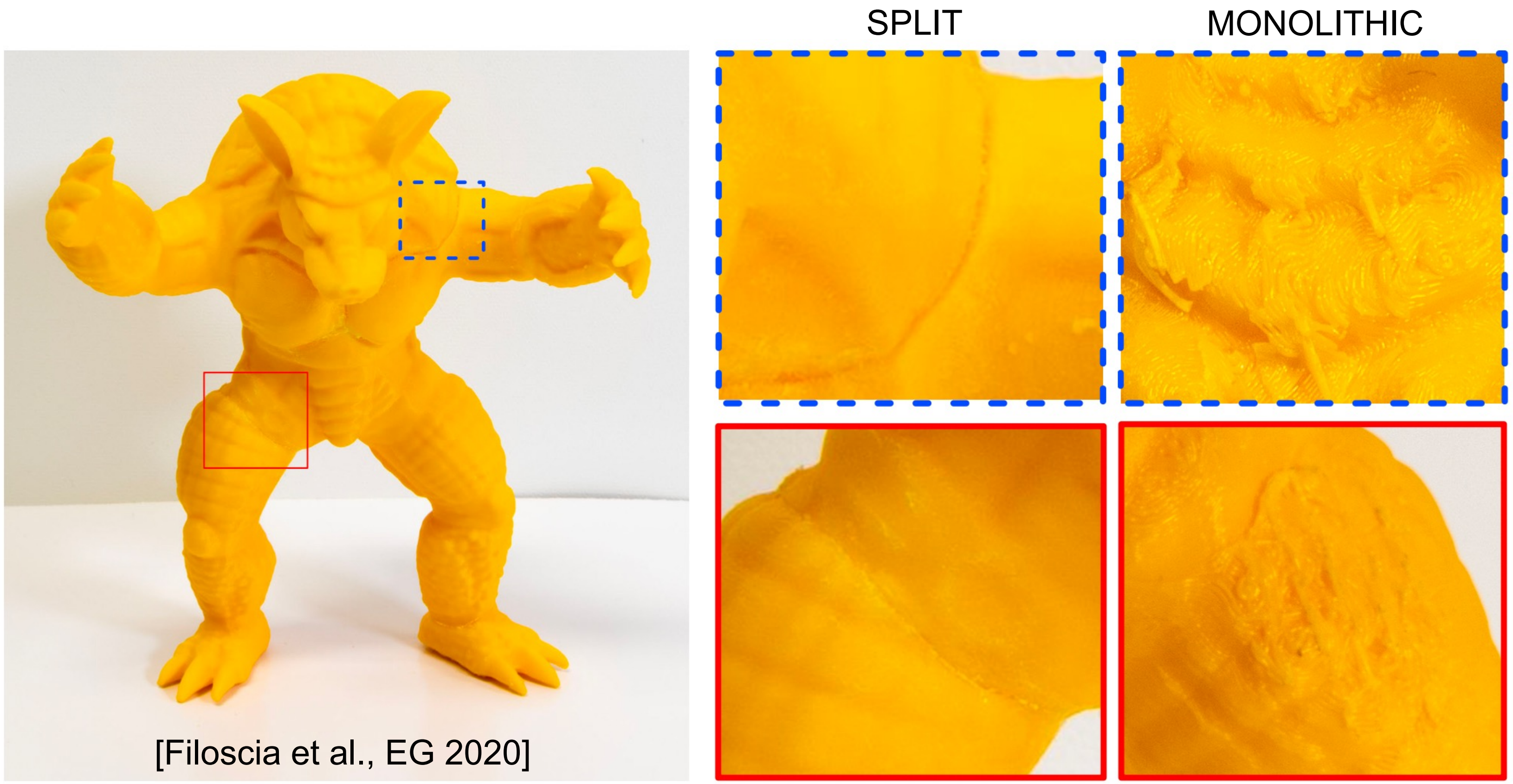


non pyramidal

pyramidal decomposition

Shape Decomposition for Manufacturing

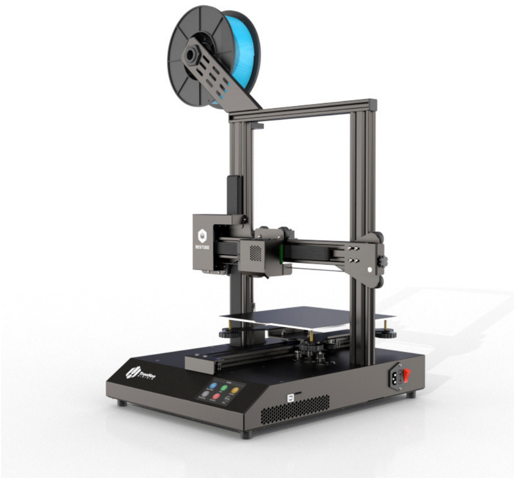
- size
- geometry
- staircase effect
- material consumption
- **support artifacts**



[Filoscia et al., EG 2020]

Shape Decomposition for Manufacturing

- size
- geometry
- staircase effect
- material consumption
- support artifacts
- **material/color**



single filament

+

composite object

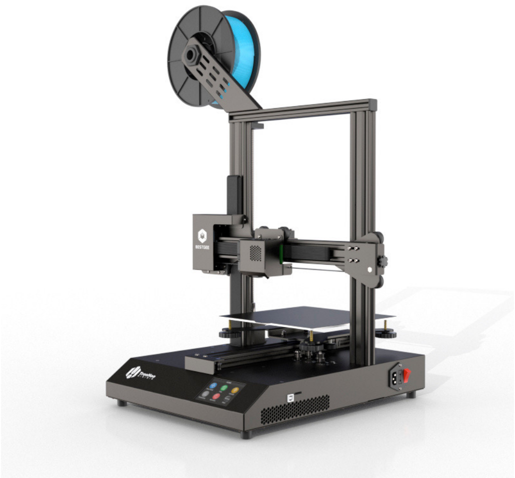
=

homogeneous
assemblable

[Araùjo et al., SIGGRAPH 2019]

Shape Decomposition for Manufacturing

- size
- geometry
- staircase effect
- material consumption
- support artifacts
- **material/color**



single filament

+

composite object

=

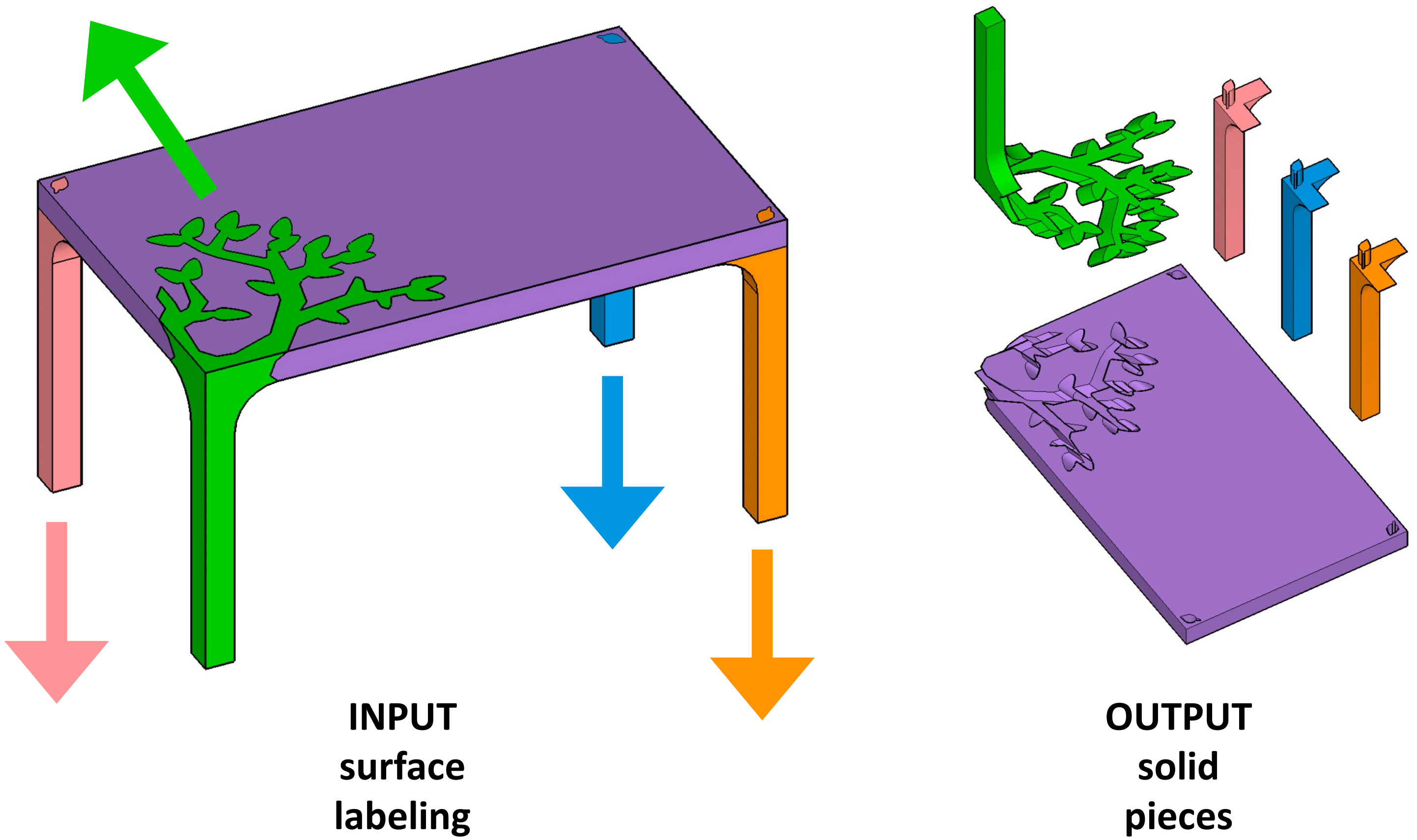
homogeneous
assemblable

[Araùjo et al., SIGGRAPH 2019]

Peng Song Ziqi Wang Marco Livesu

Shape Decomposition for Manufacturing

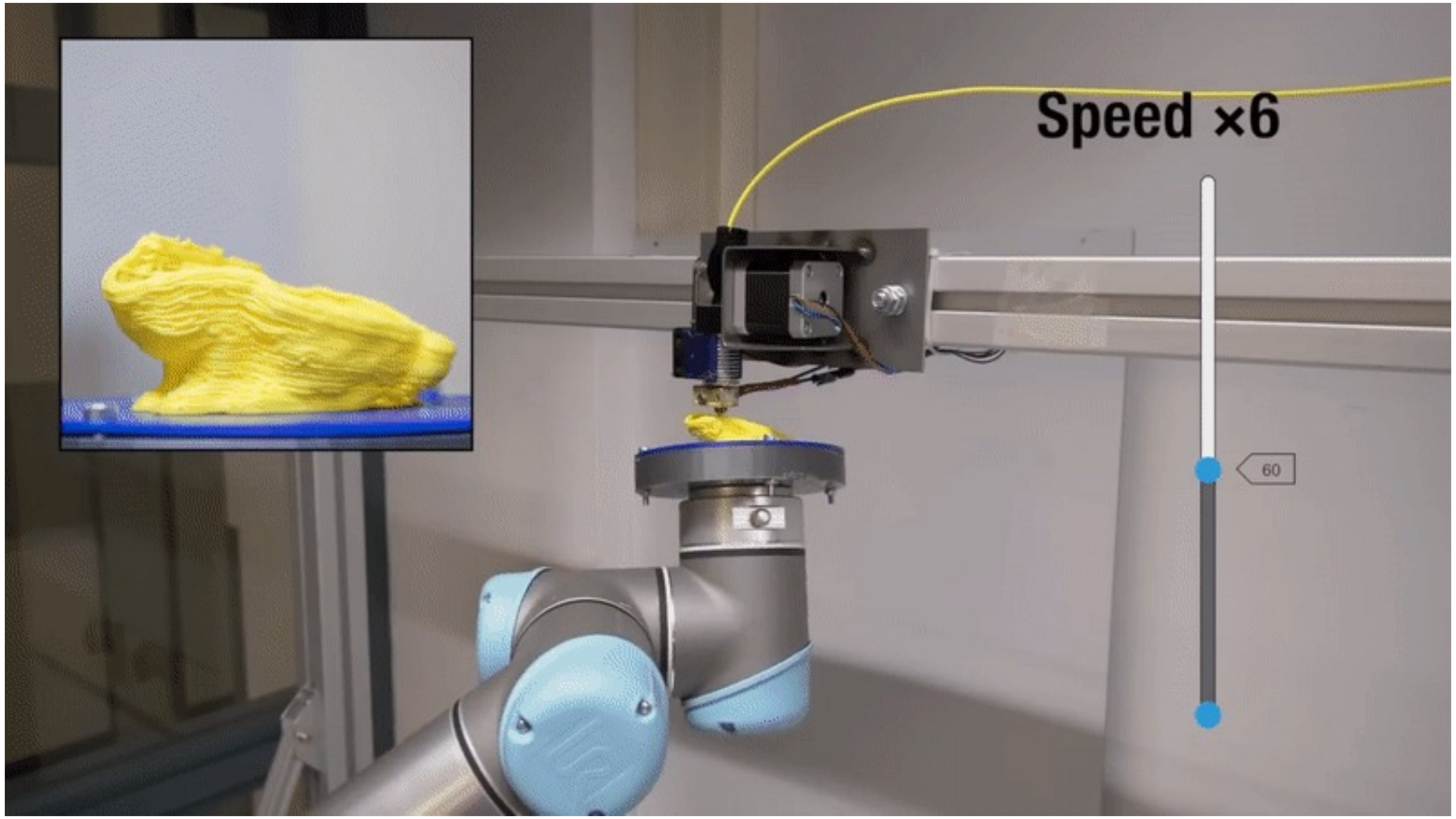
- size
- geometry
- staircase effect
- material consumption
- support artifacts
- material/color
- **assemblability**



[Yao et al., ACM TOG 2017]

Shape Decomposition for Manufacturing

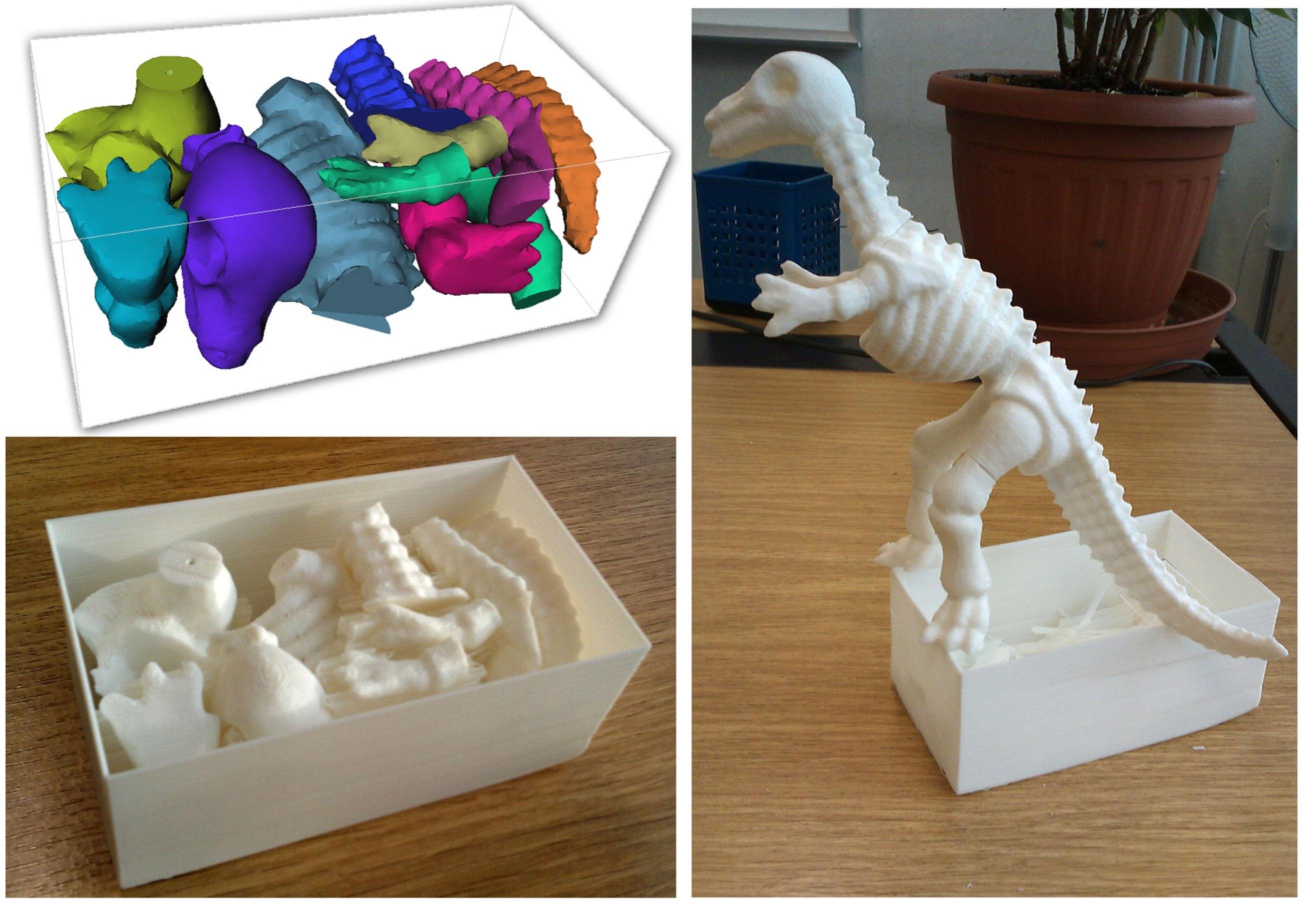
- size
- geometry
- staircase effect
- material consumption
- support artifacts
- material/color
- assemblability
- **collision avoidance**



[Dai et al., SIGGRAPH 2018]

Shape Decomposition for Manufacturing

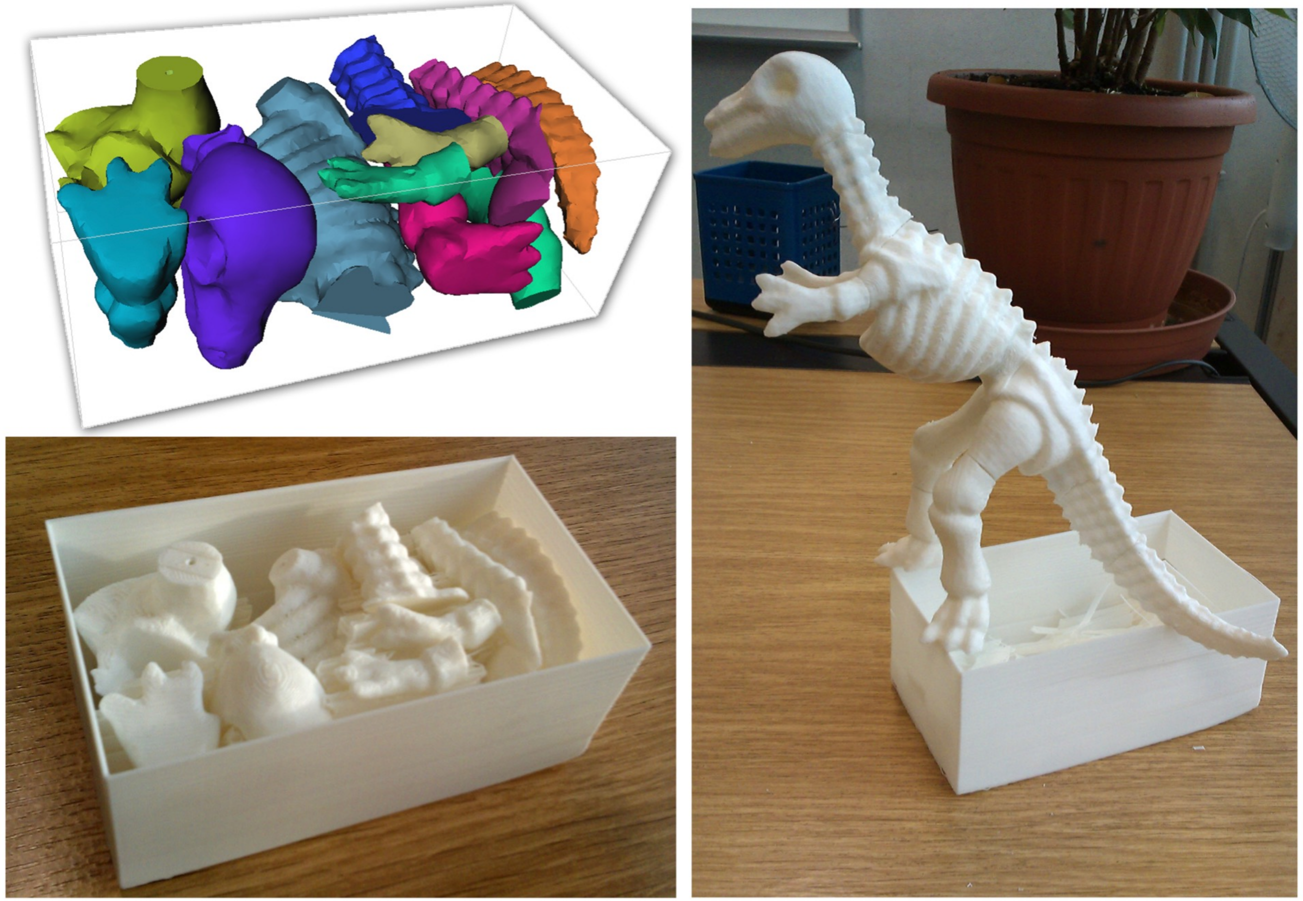
- size
- geometry
- staircase effect
- material consumption
- support artifacts
- material/color
- assemblability
- collision avoidance
- **packing**



[Attene, EG 2015]

Shape Decomposition for Manufacturing

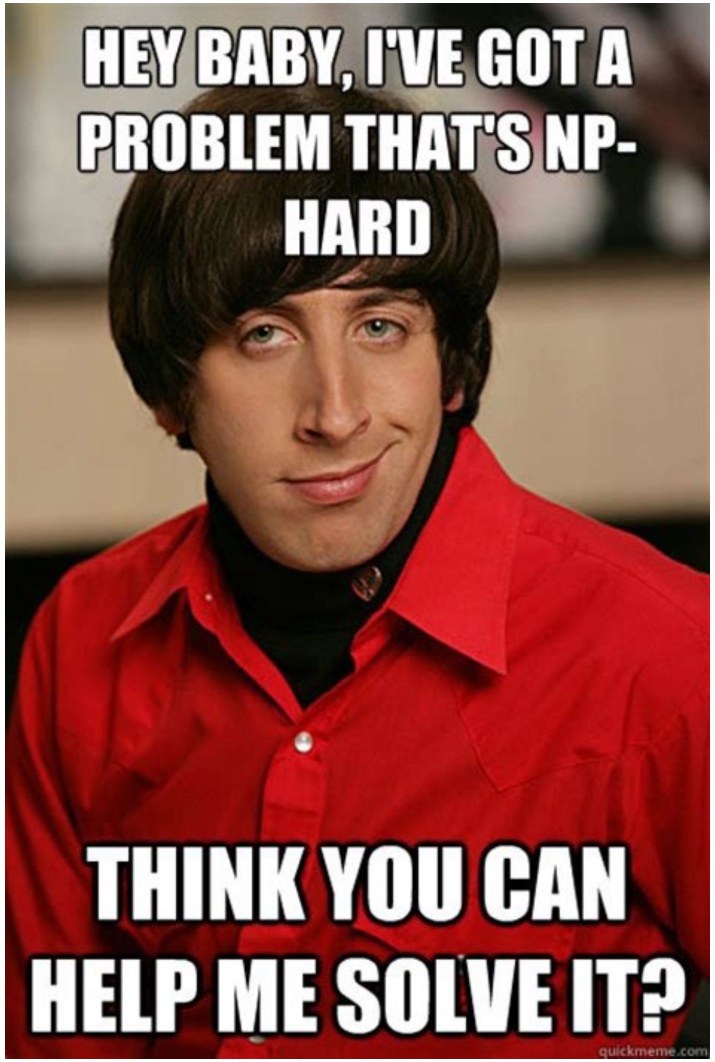
- size
- geometry
- staircase effect
- material consumption
- support artifacts
- material/color
- assemblability
- collision avoidance
- packing
- **… and many others!**



[Attene, EG 2015]

How Hard it is?

- Computing **a decomposition** is **easy**
- Computing **a constrained decomposition** is **still easy**
  - why? I can split into many tiny pieces!
- Computing **a constrained decomposition** that is *minimal* is **NP-Hard**
  - search space grows exponentially
  - no hope to find the global optimum
  - just strive for a *"good"* local minimum
  - it's all about heuristics

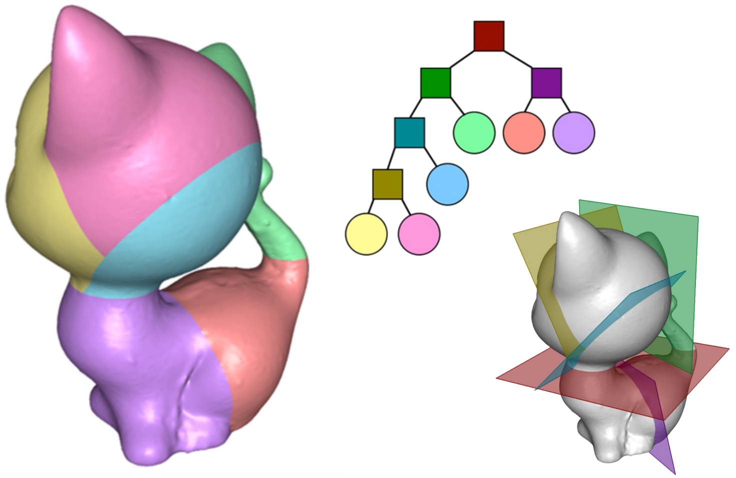


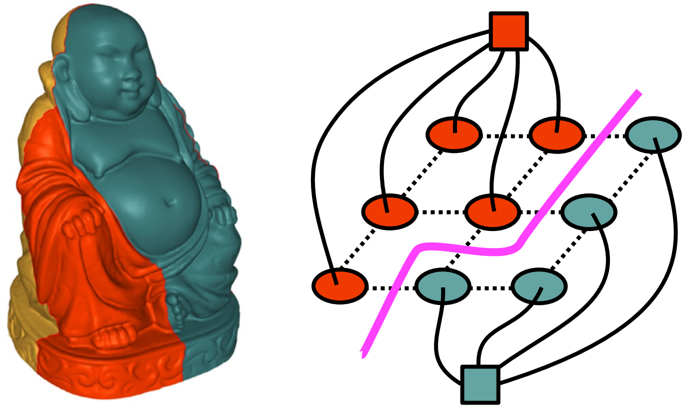
Recurring Approaches

- Despite this variety of goals, manufacturing paradigms and fabrication hardware, all methods

- Aim to control the **same aspects**
  - part size
  - local surface orientation
  - assemblability

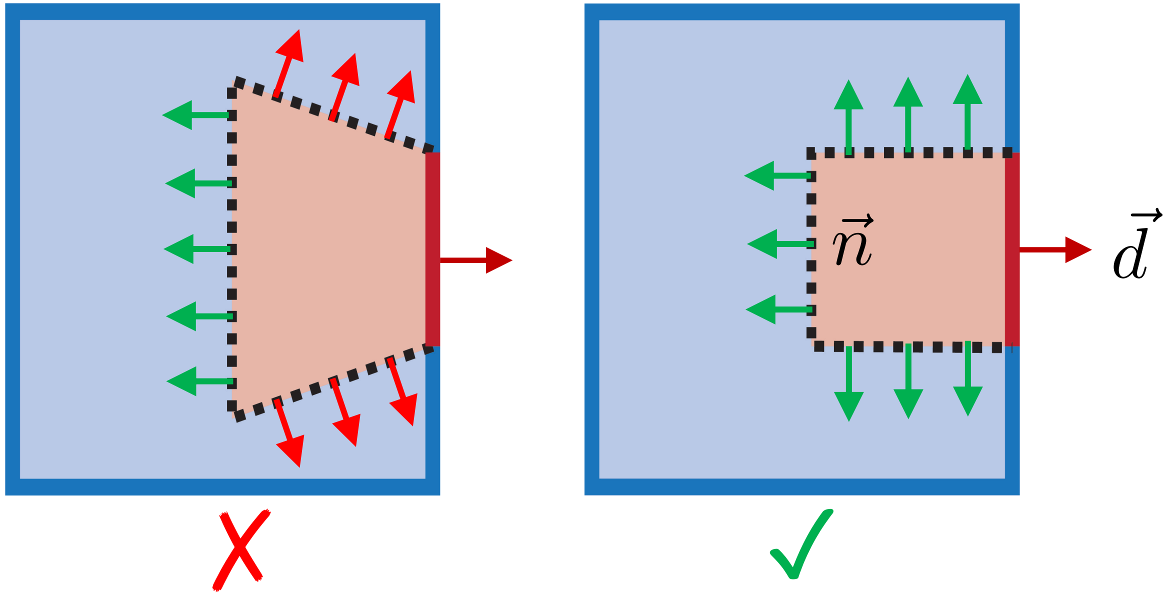
- Mostly exploit **similar techniques**
  - Binary Space Partitions
  - Graph Labeling
  - Mesh Booleans





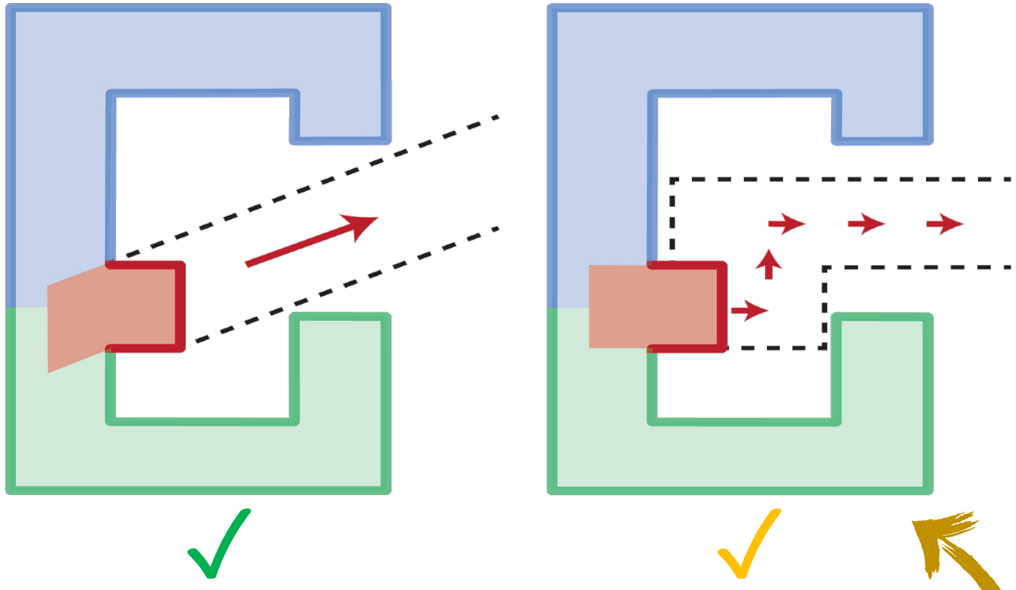
Assemblability – Rigid Bodies

- Parts must fulfill both **local** and **global** requirements



INTERFACES MUST BE **HEIGHT FIELDS**
W.R.T. THE EXTRACTION DIRECTION

$$\vec{d} \cdot \vec{n} \leq 0$$

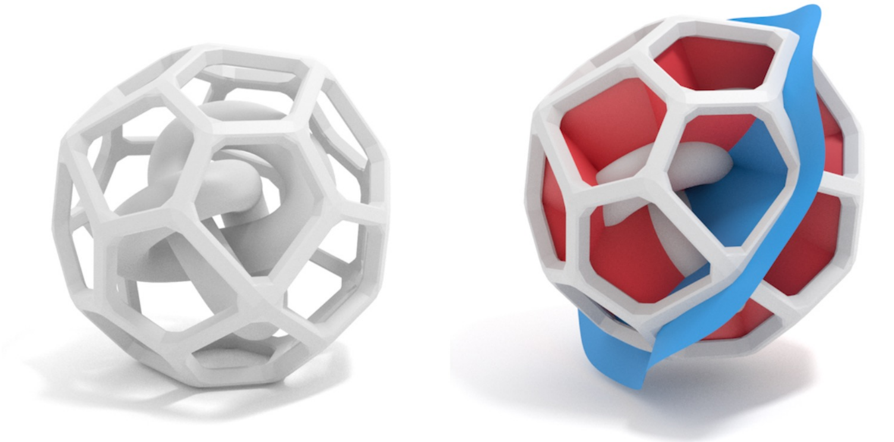


THERE MUST EXIST A **CLEARED**
EXTRACTION PATH

HUGE SEARCH SPACE!

Assemblability – Soft Bodies

- Can be extracted even if they **violate** height fieldness
  - full FEM simulation is overly expensive
    - simulating contact and frictional forces is hard!

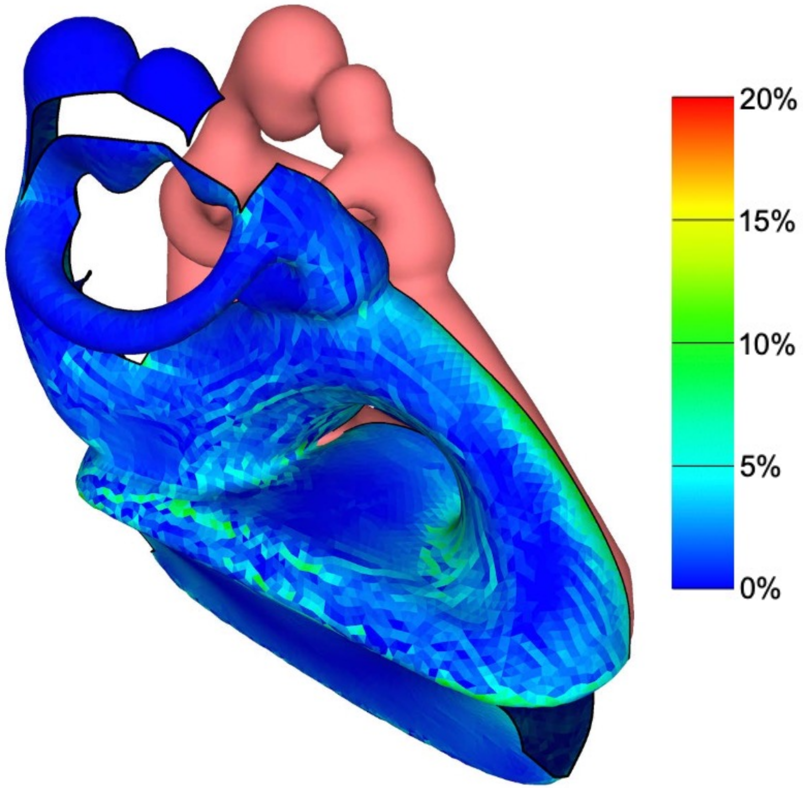


[Alderighi et al., SIGGRAPH 2019]

Assemblability of Deformable Bodies

- Can be extracted even if they **violate** height fieldness
  - full FEM simulation is overly expensive
    - simulating contact and frictional forces is hard!

- **FlexMolds approach**
  - projective dynamics on a thin shell
  - measure per triangle deformation during extraction

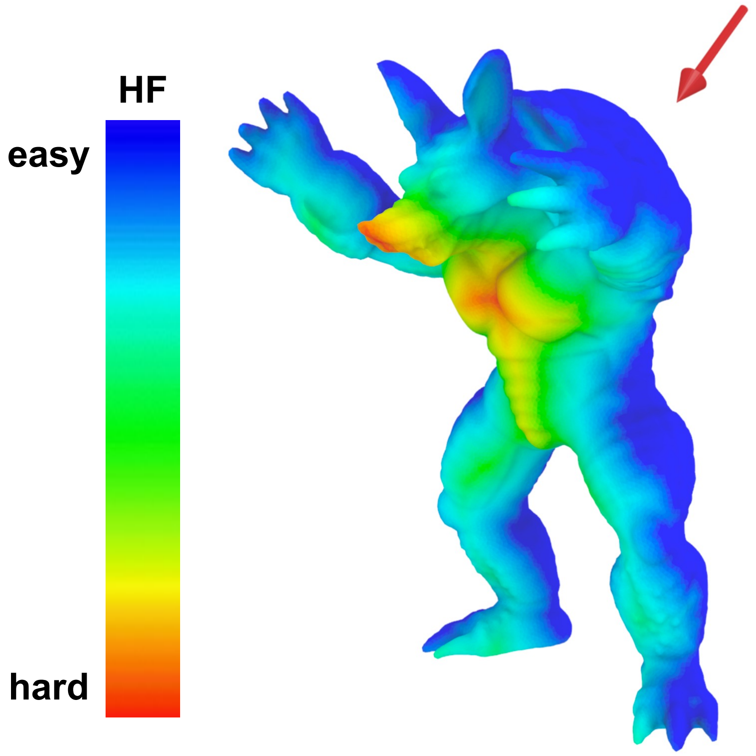


Assemblability of Deformable Bodies

- Can be extracted even if they **violate** height fieldness
  - full FEM simulation is overly expensive
    - simulating contact and frictional forces is hard!

- **FlexMolds approach**
  - projective dynamics on a thin shell
  - measure per triangle deformation during extraction

- **MetaMolds approach**
  - fully geometric
  - geodesic distance from closest HF region
  - works remarkably well in practice!

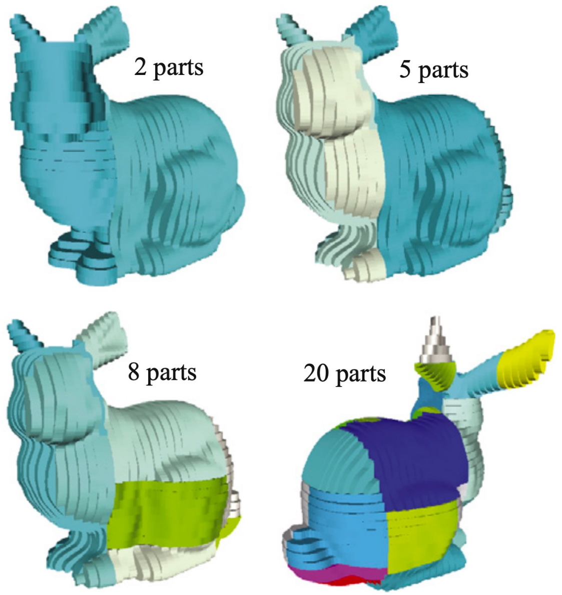


Binary Space Partitions

Size

[Luo et al., SIGGRAPH 2012]

Staircase Effect



[Hildebrand et al., SMI 2013]

Packing

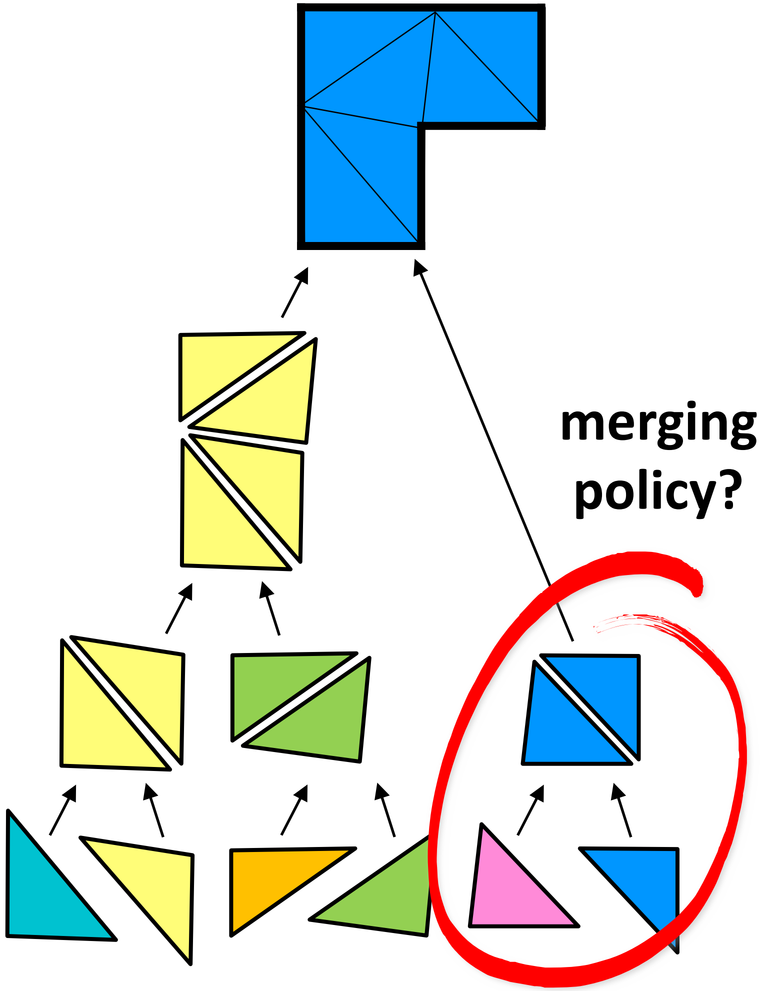
[Chen et al., SIGGRAPH Asia 2015]

[Attene, EG 2015]

Top Down
(with beam search)

Bottom Up
(greedy)

Top Down vs Bottom Up



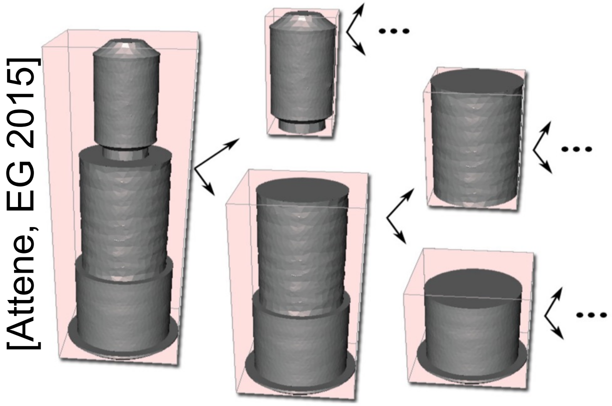
Shapes in a Box:

- **discretize** domain (tetrahedralization)
- minimize absolute aboxiness

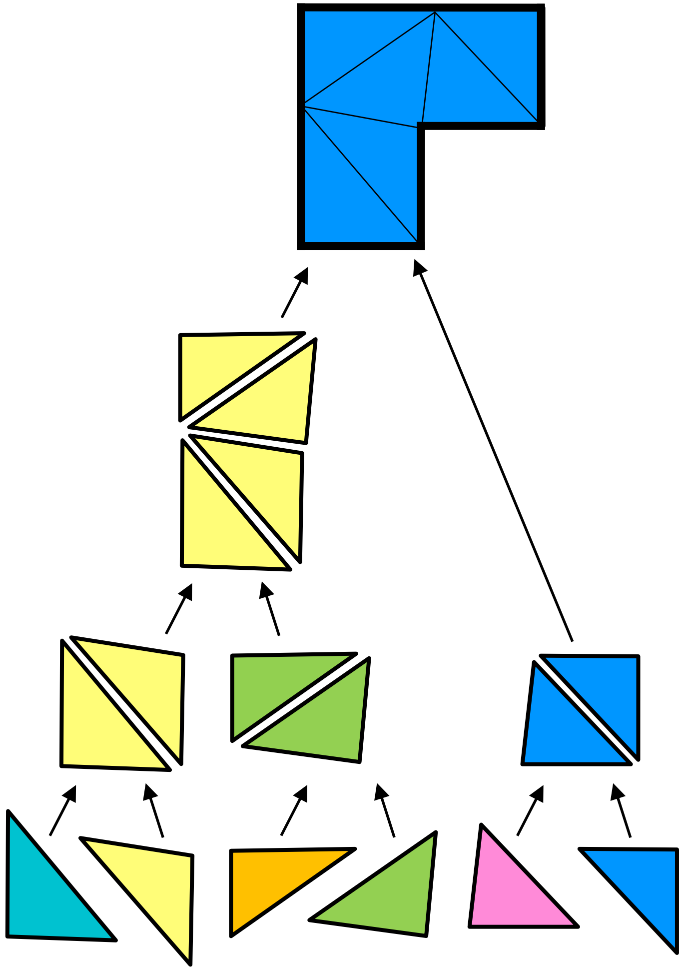
$$A(\mathcal{O}) = VOL(OBB(\mathcal{O})) - VOL(\mathcal{O})$$

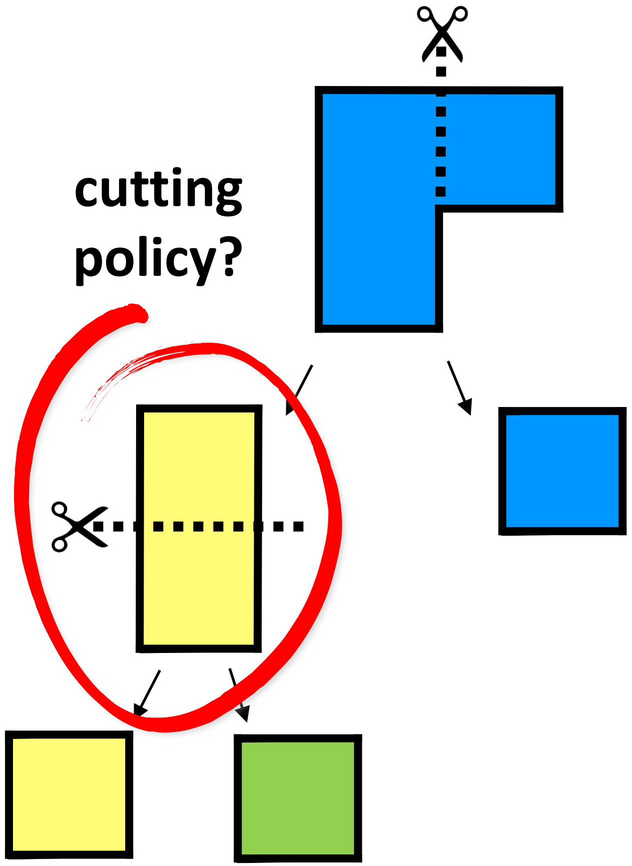
cluster of tetrahedra

minimal oriented bbox



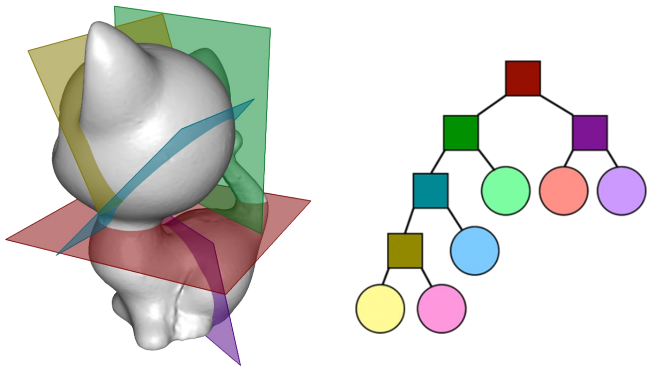
Top Down vs Bottom Up





Chopper:

- **discretize** set of planar cuts
- minimize split metric based on
    - # of parts
    - connectors
    - structure/fragility
    - aesthetics (hide seams)
    - symmetry



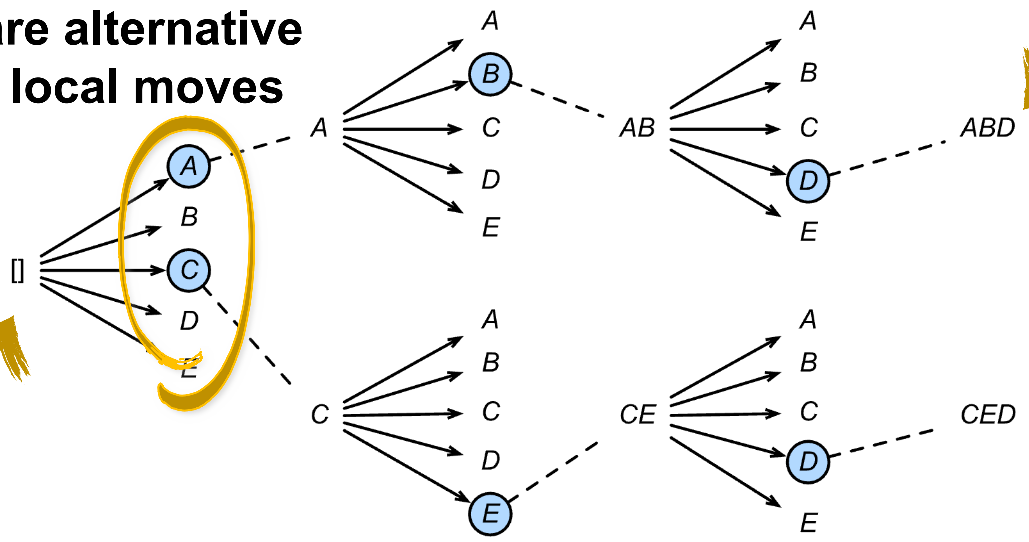
[Luo et al., SIGGRAPH 2012]

Exploring the Space of Solutions

- **Greedy:** at each step pick the best move
  - easy to get stuck at local minima!
- **Beam search** allows to explore a wider portion of the feasible space
  - assumption: partial solutions can be ranked
  - idea: at each level, continue exploring only the N best partial solutions (N is called **beam width**)

outcoming arcs are alternative local moves

this tree is the space of solutions!

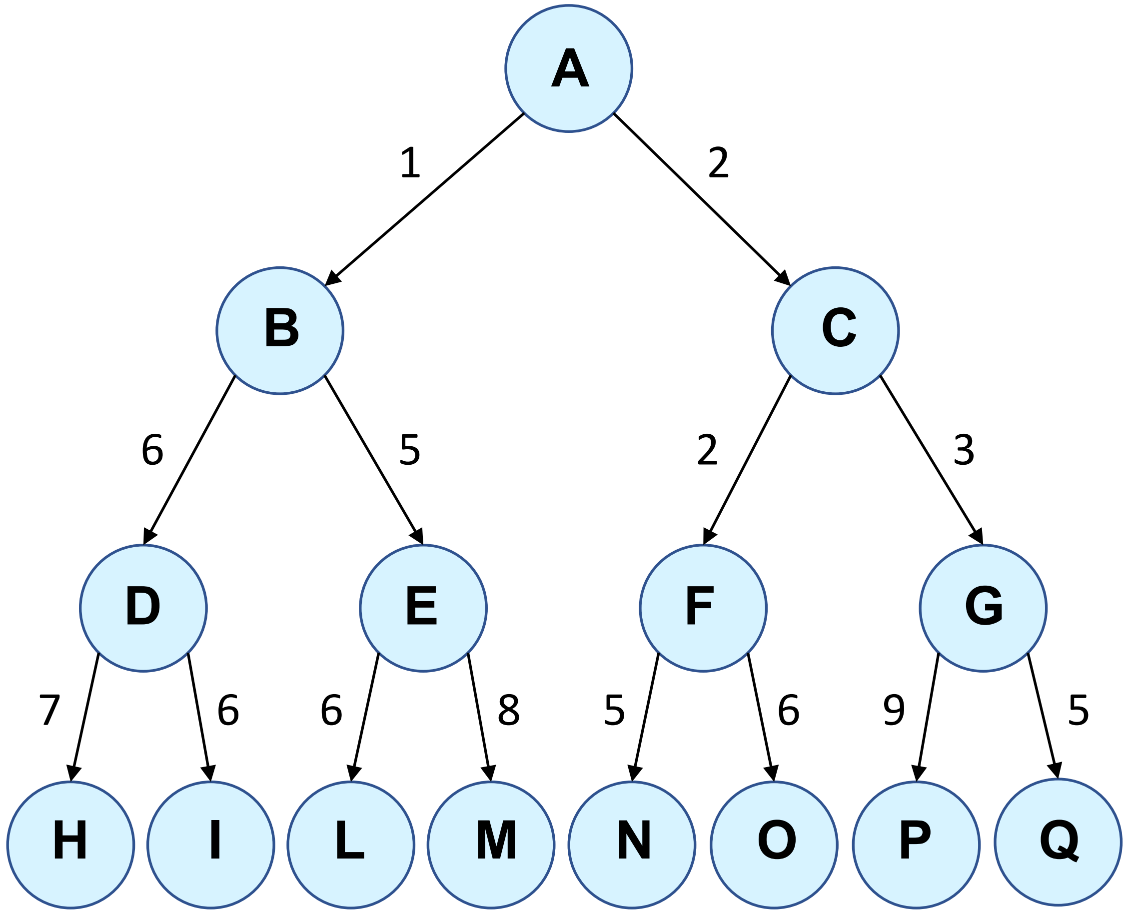


each node is an alternative partition tree

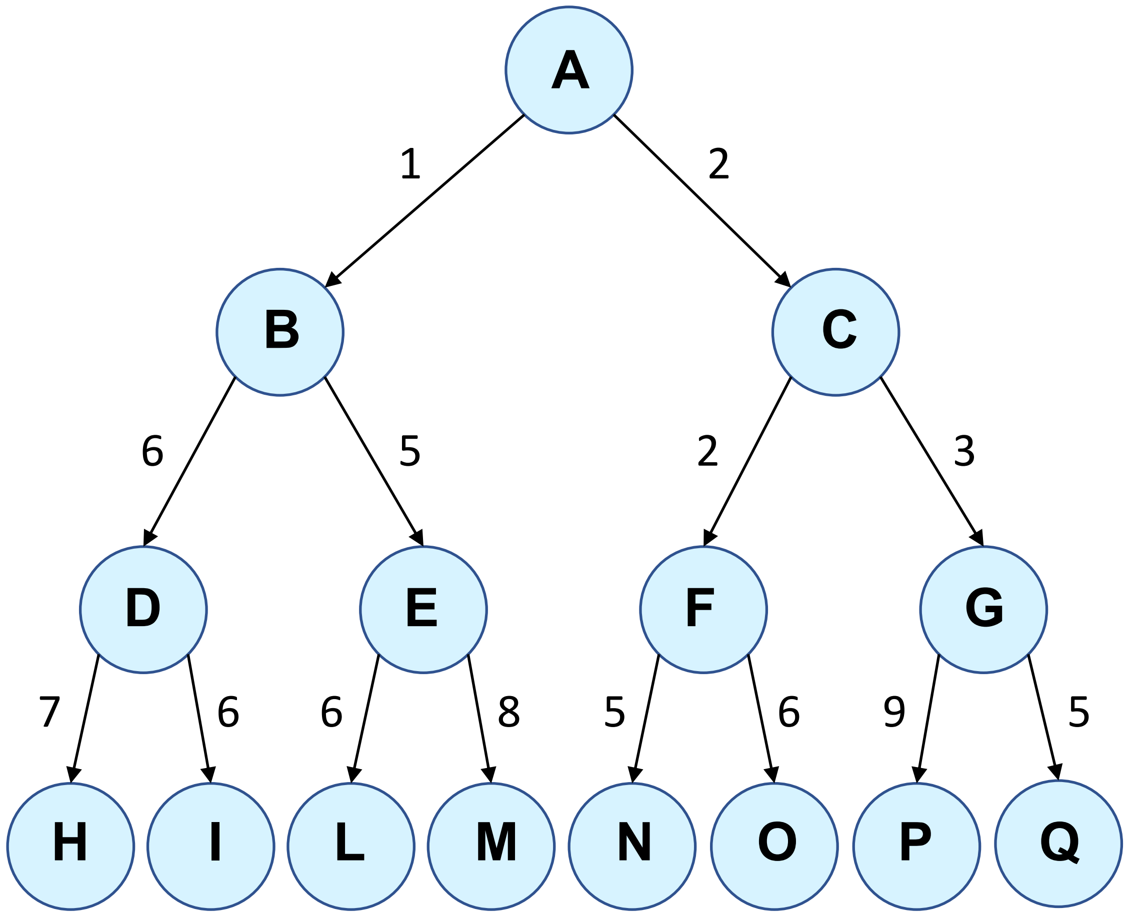
(in this example beam width is 2)

Beam Search vs Greedy

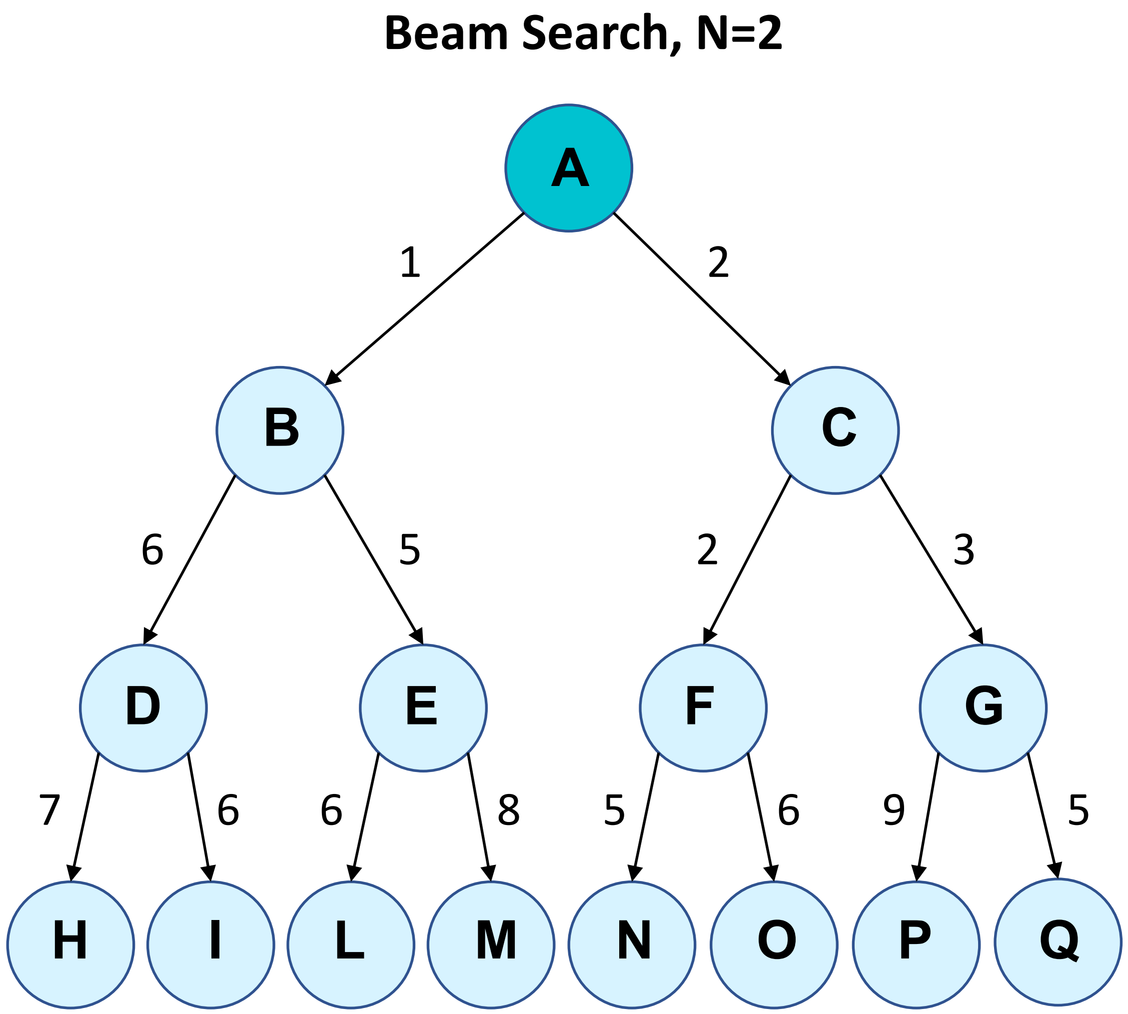
Beam Search, N=2

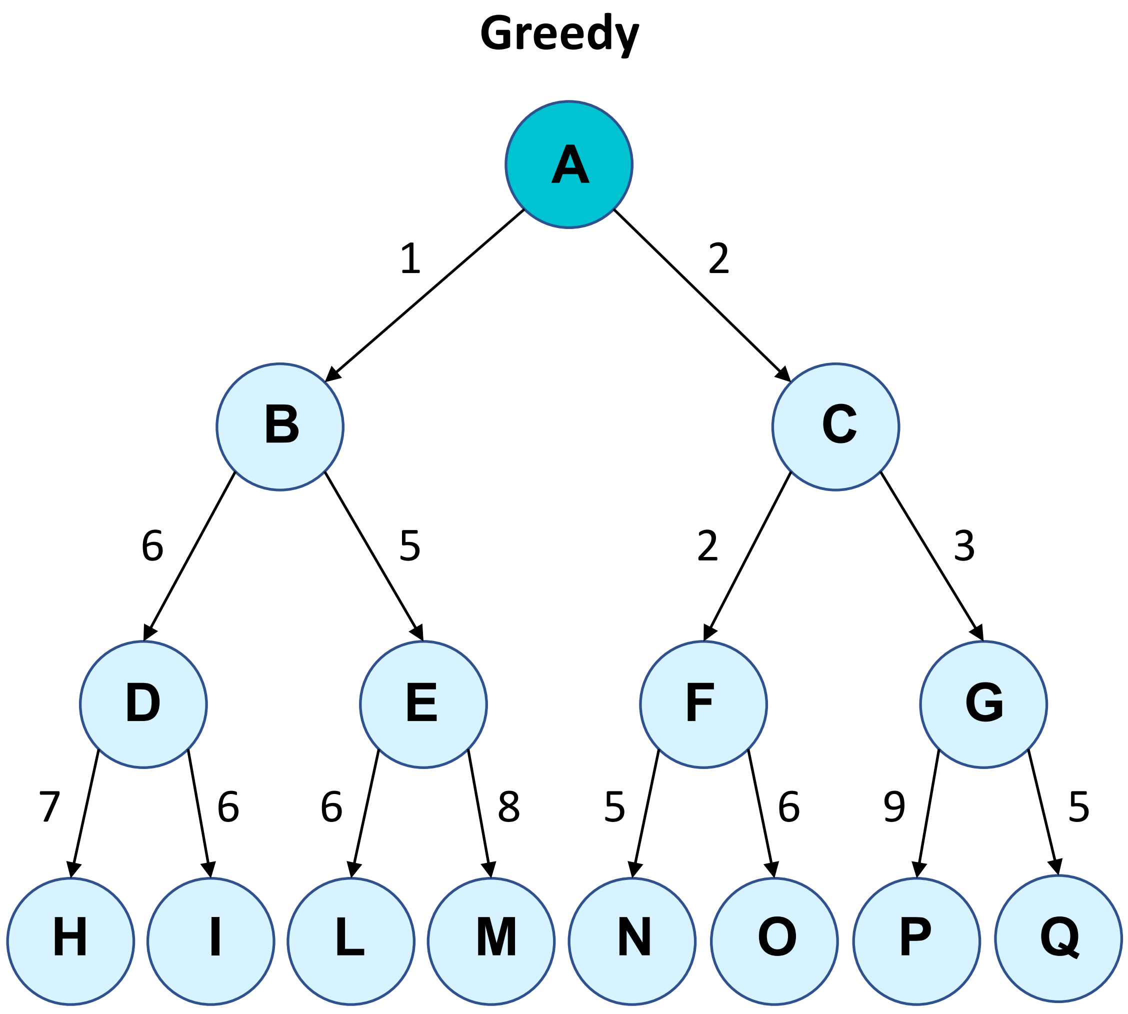


Greedy



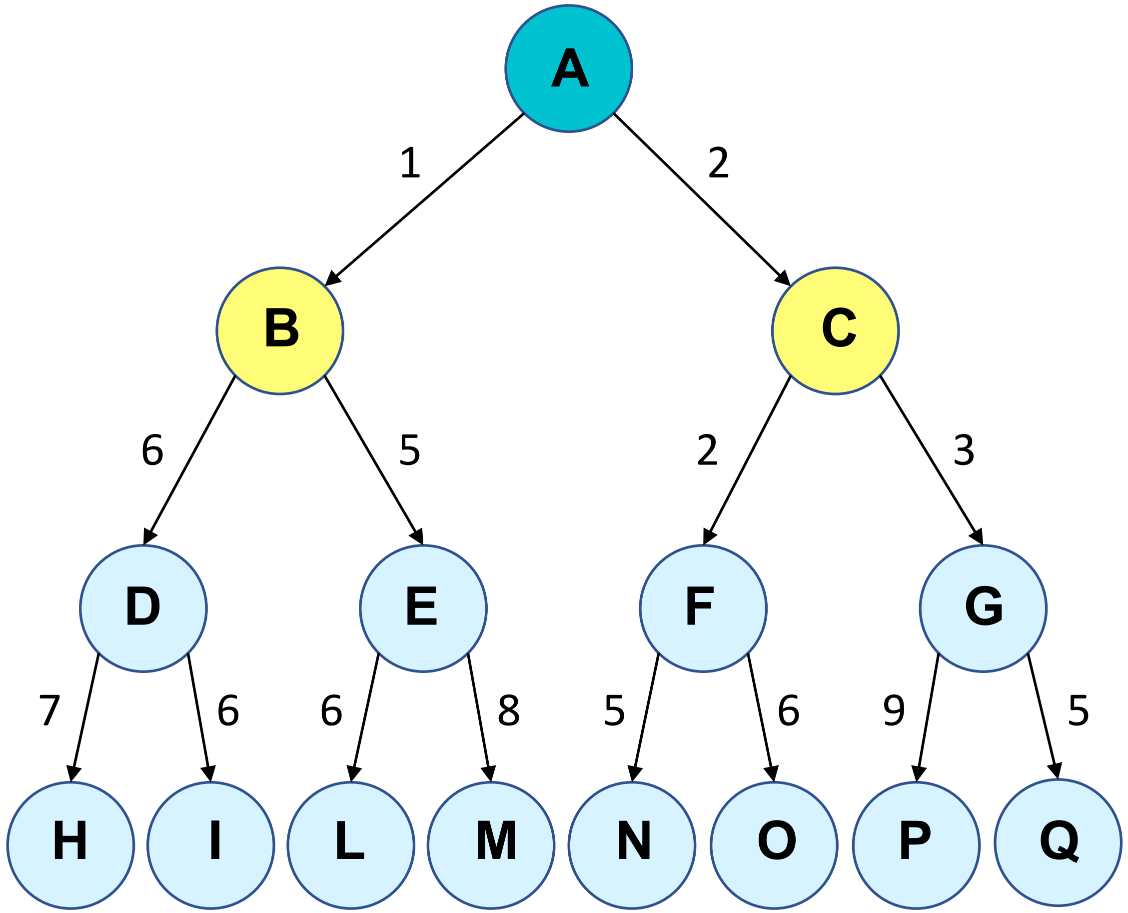
Beam Search vs Greedy



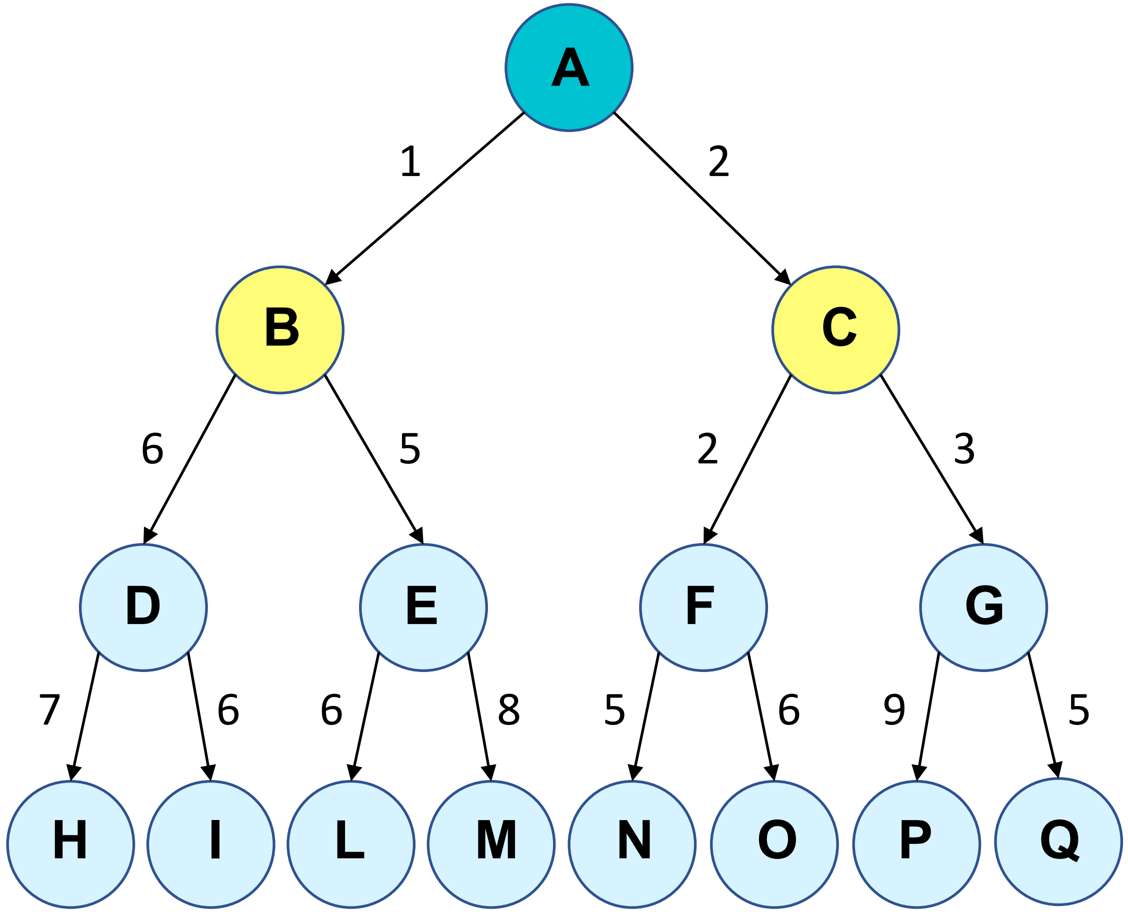


Beam Search vs Greedy

Beam Search, N=2

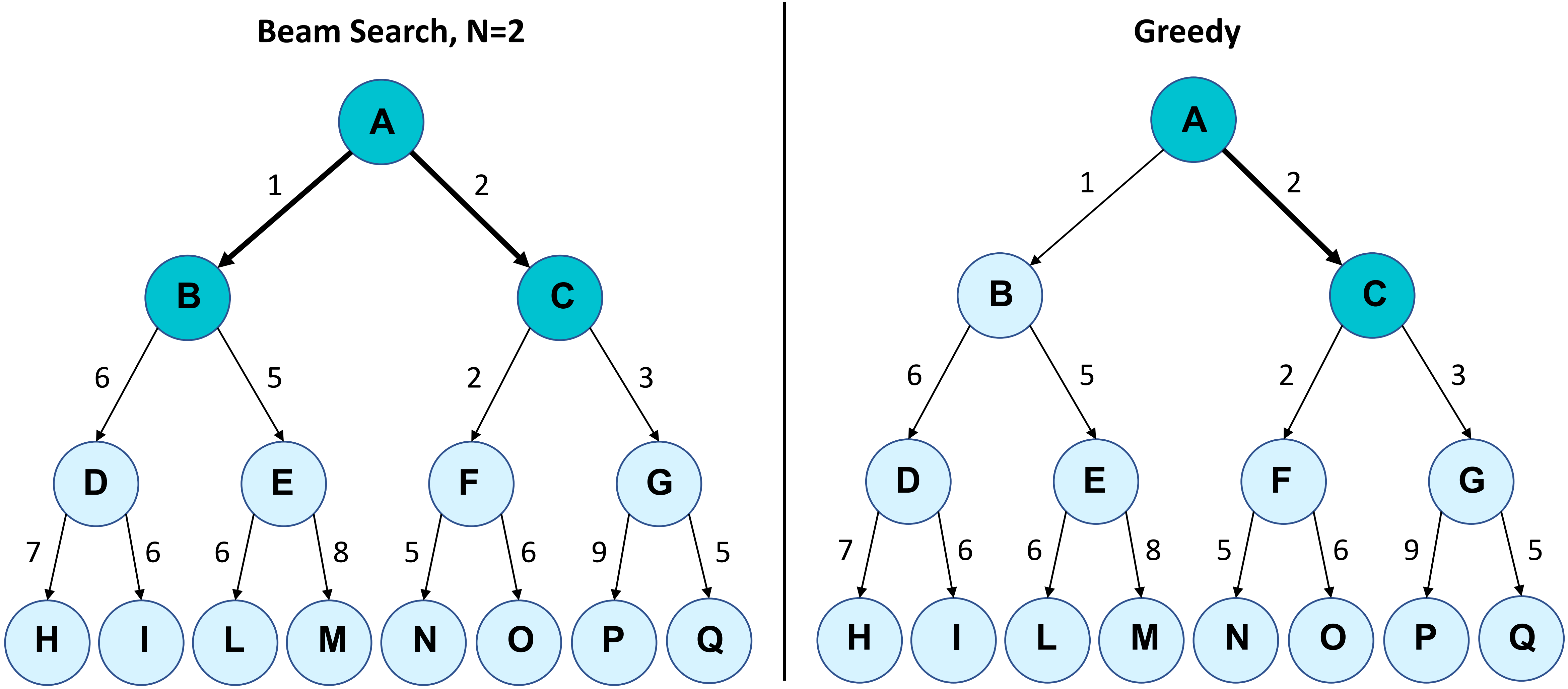


Greedy



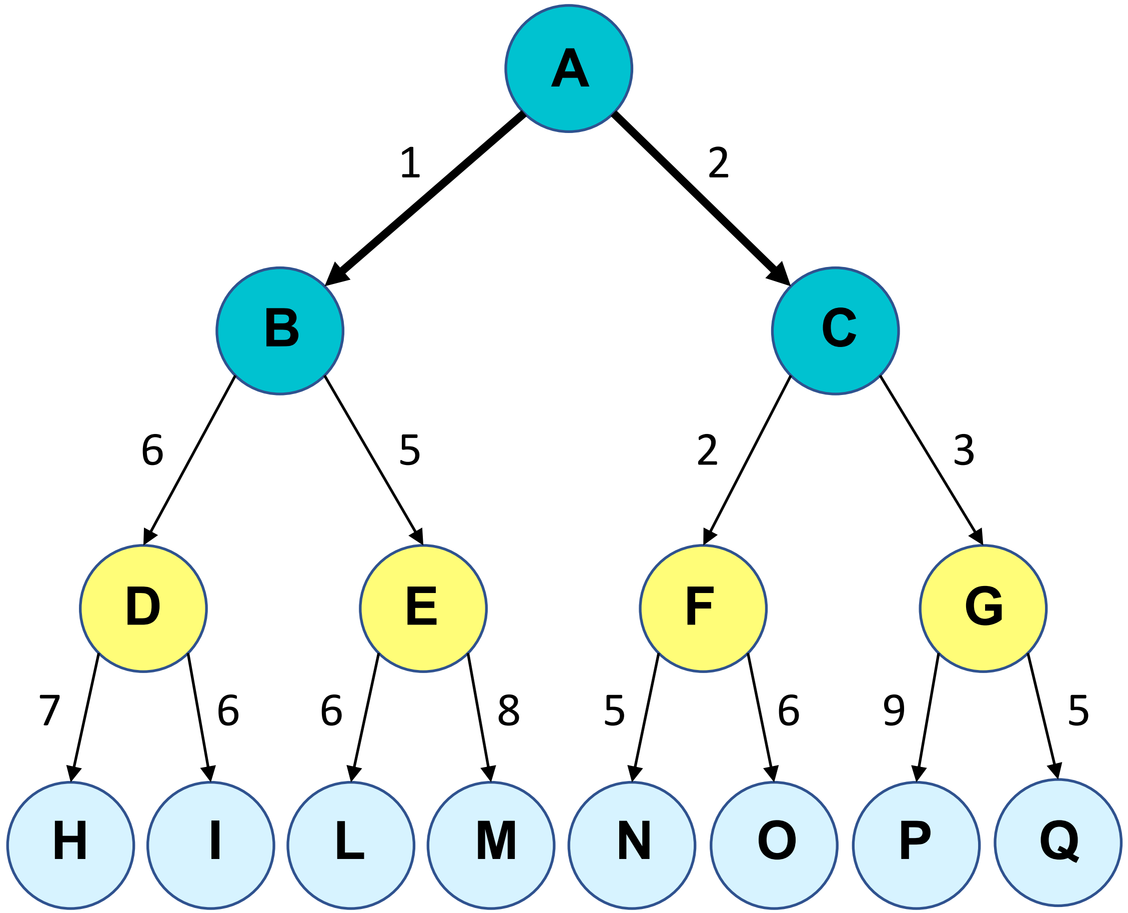
Peng Song Ziqi Wang Marco Livesu

Beam Search vs Greedy

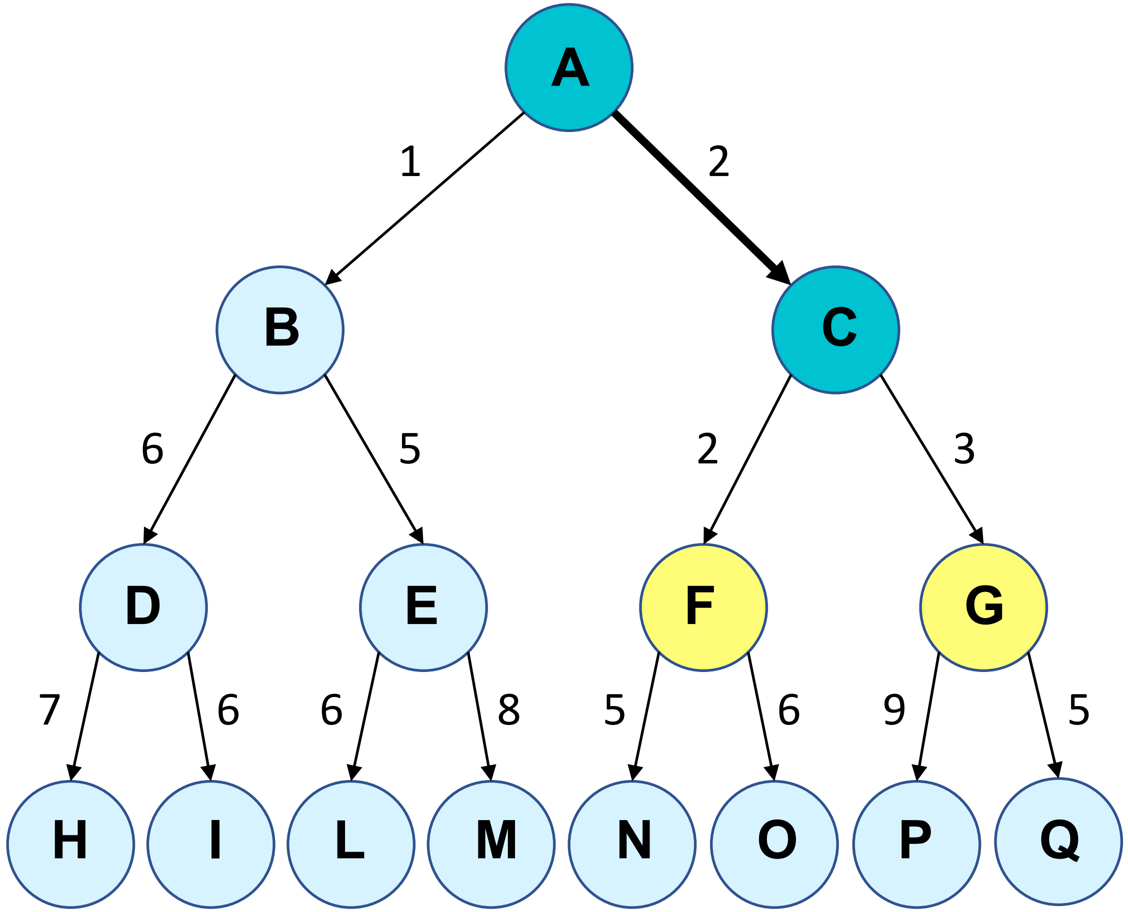


Beam Search vs Greedy

Beam Search, N=2



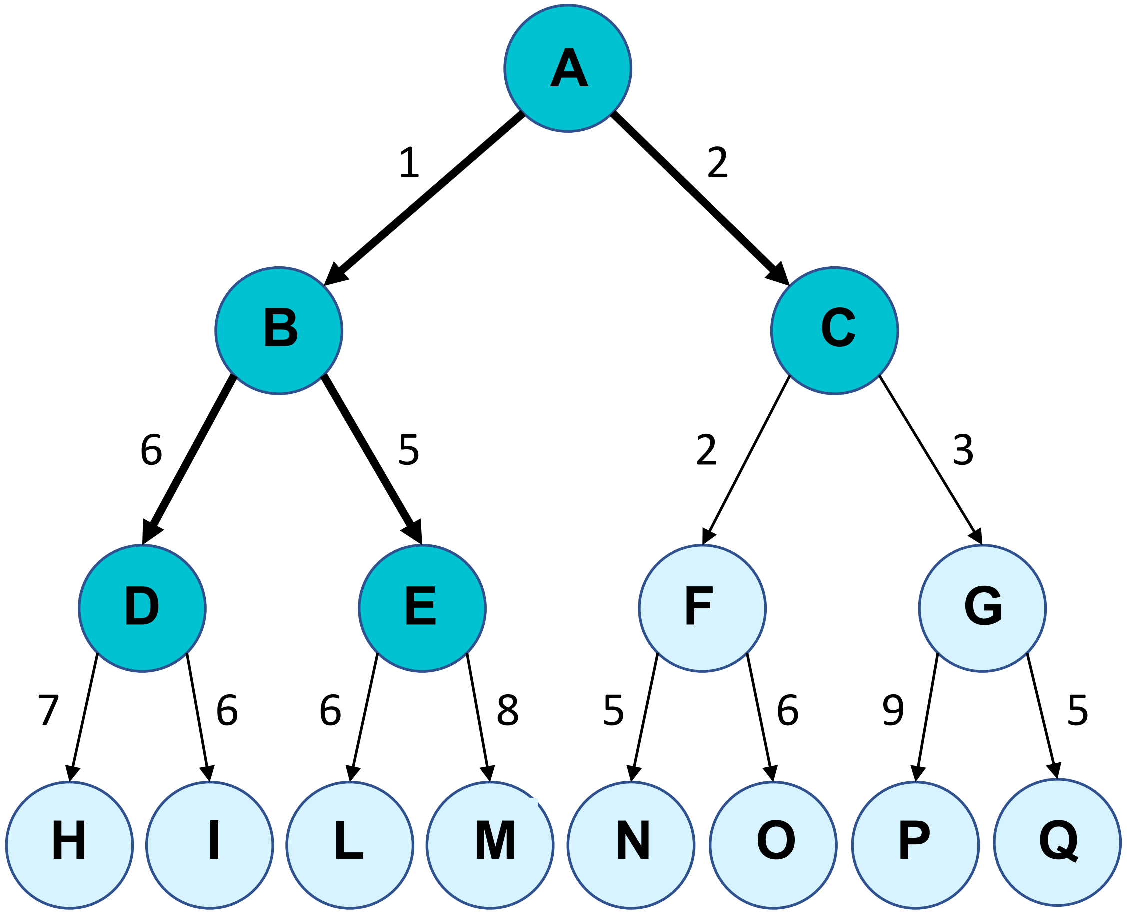
Greedy



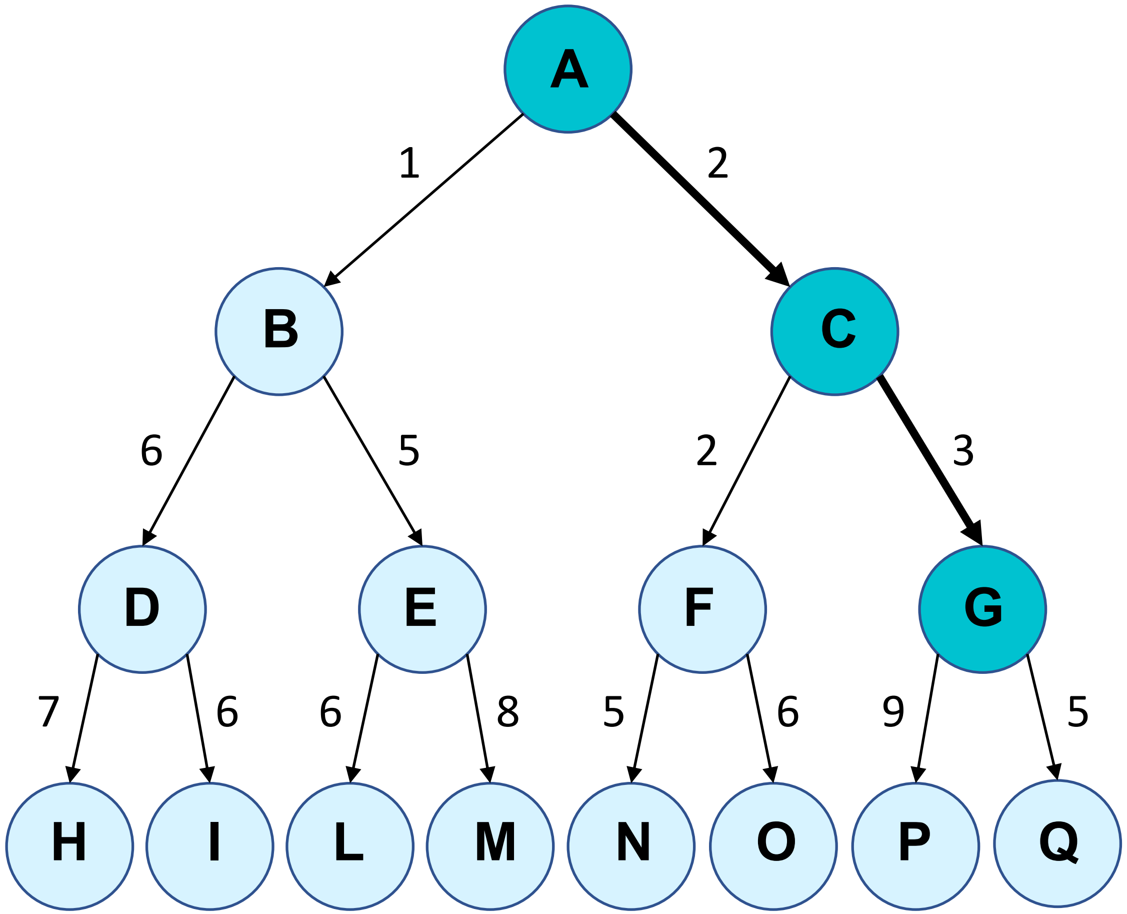
Peng Song Ziqi Wang Marco Livesu

Beam Search vs Greedy

Beam Search, N=2

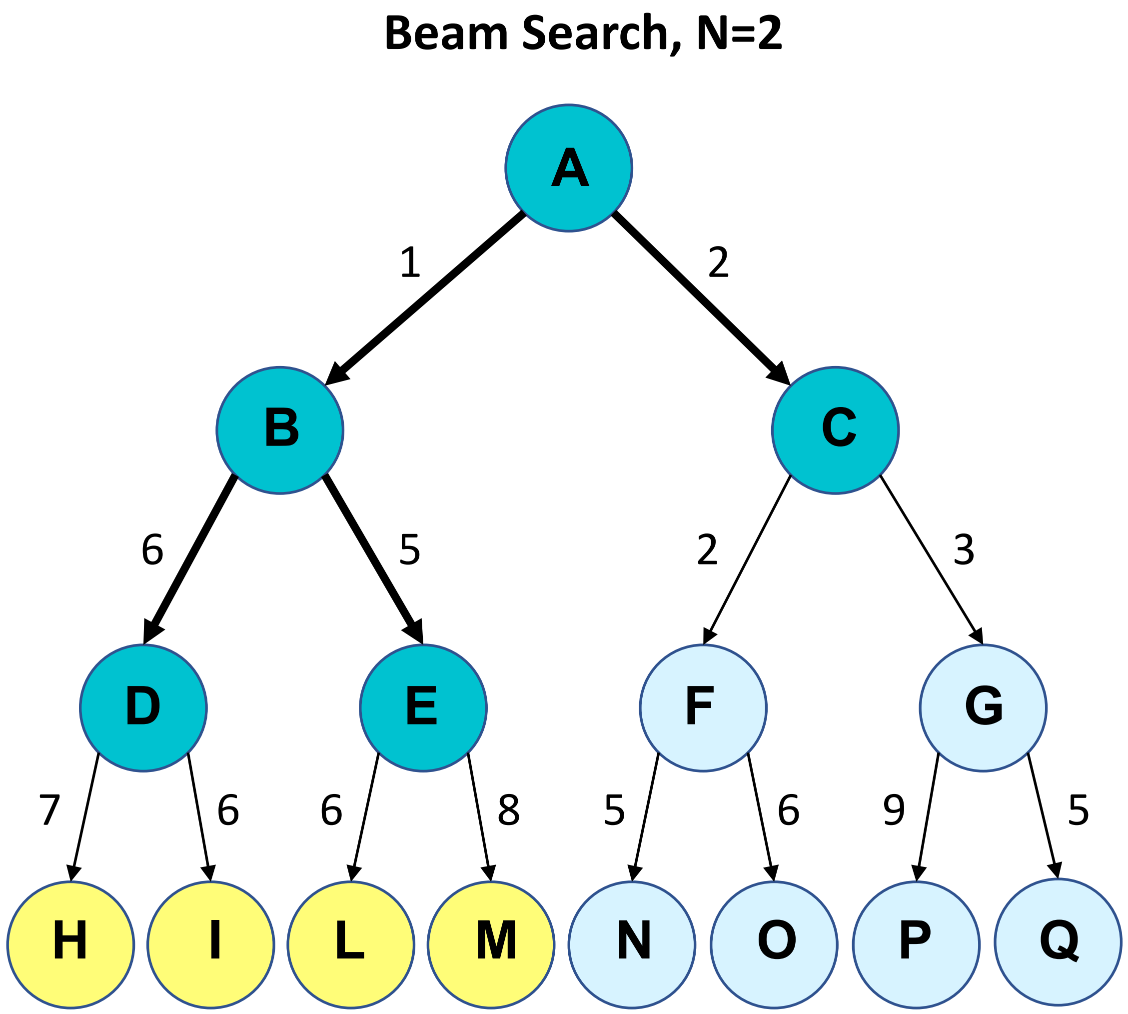


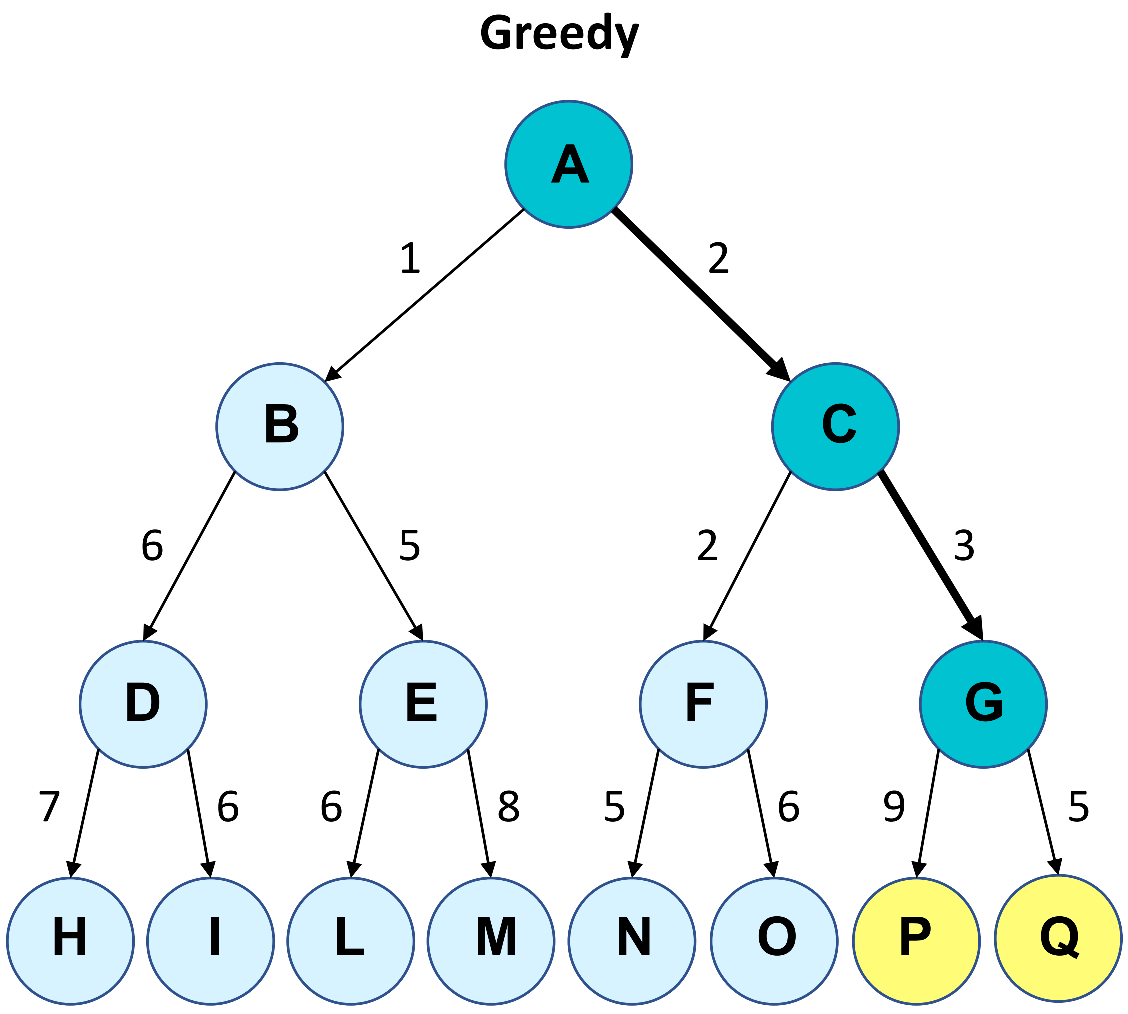
Greedy



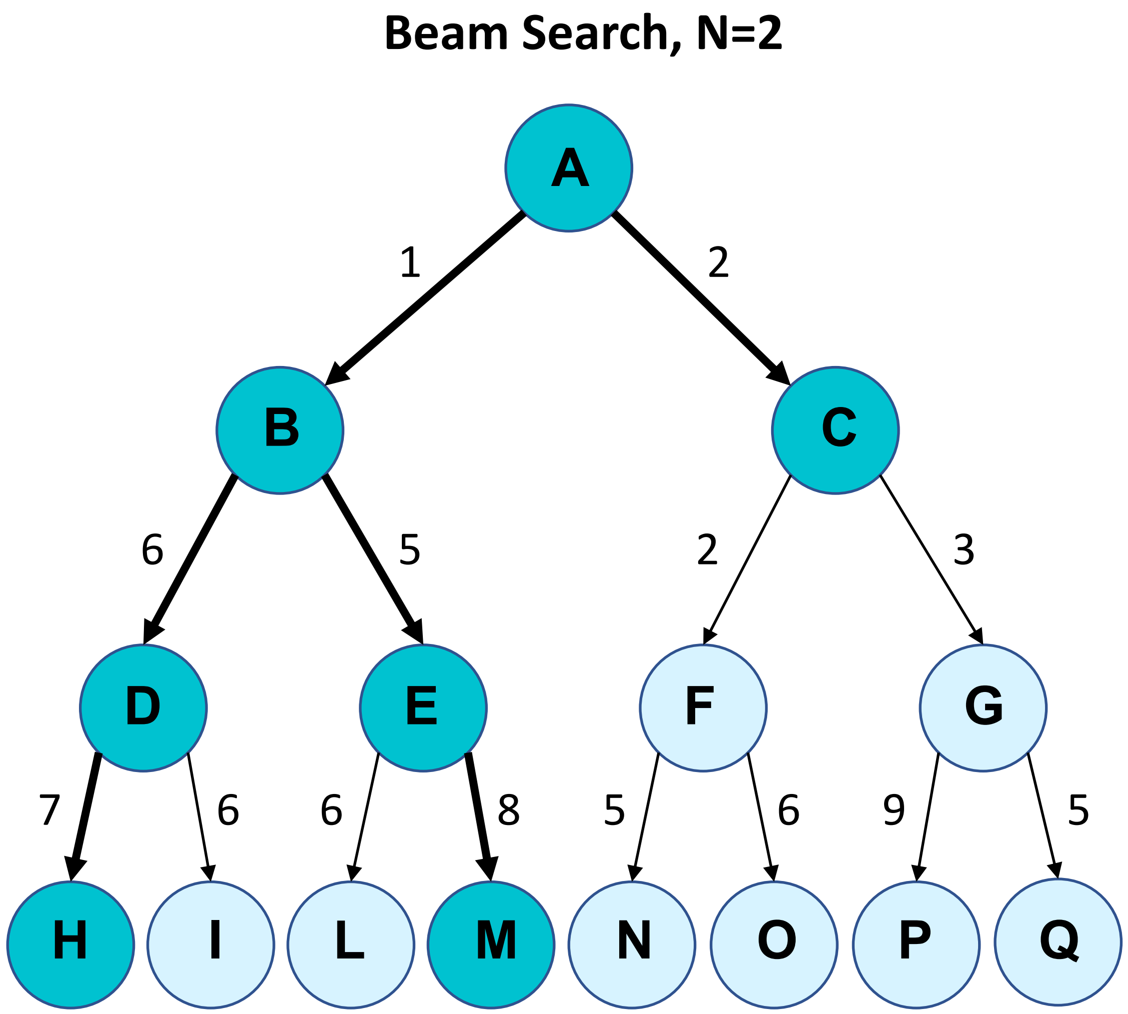
Peng Song Ziqi Wang Marco Livesu

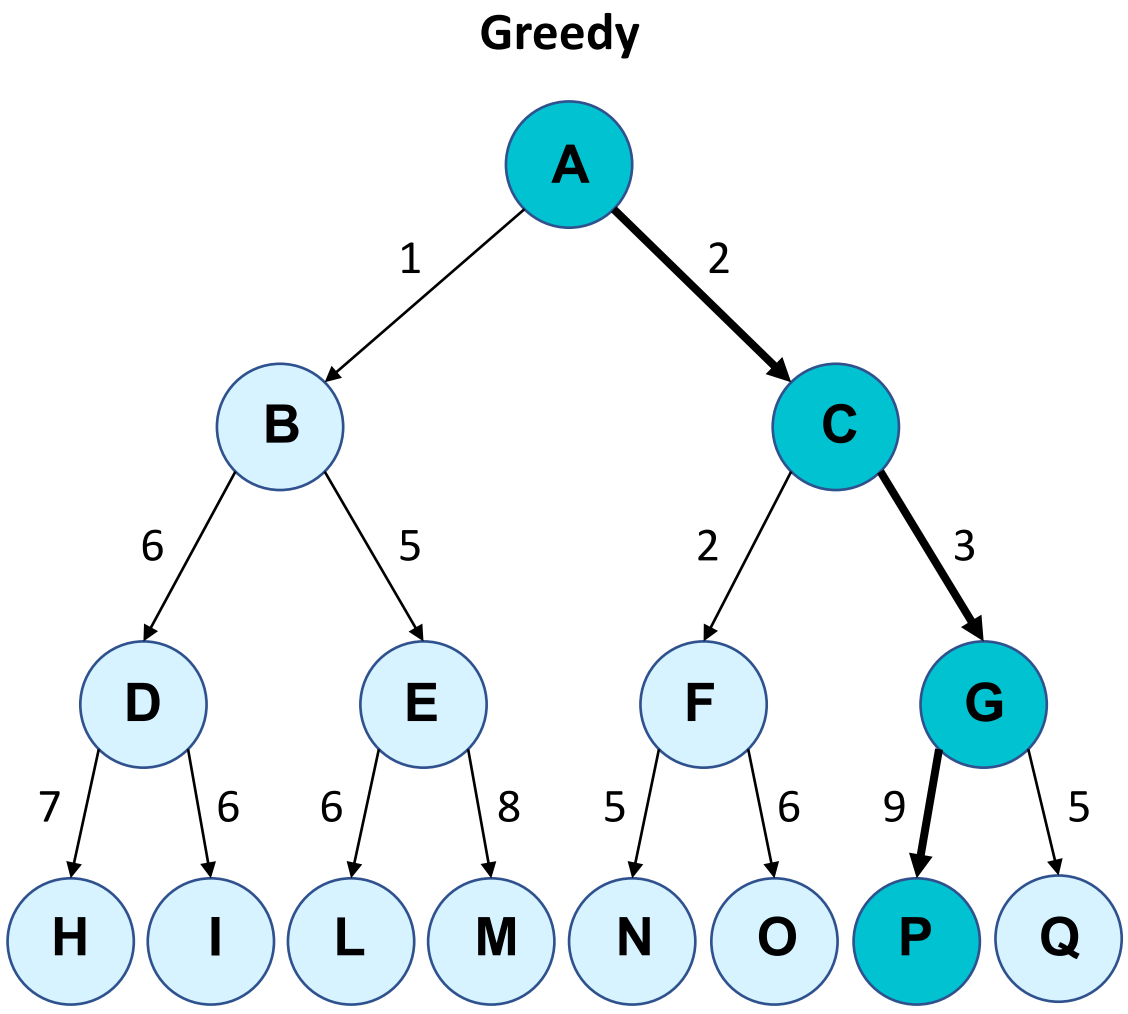
Beam Search vs Greedy



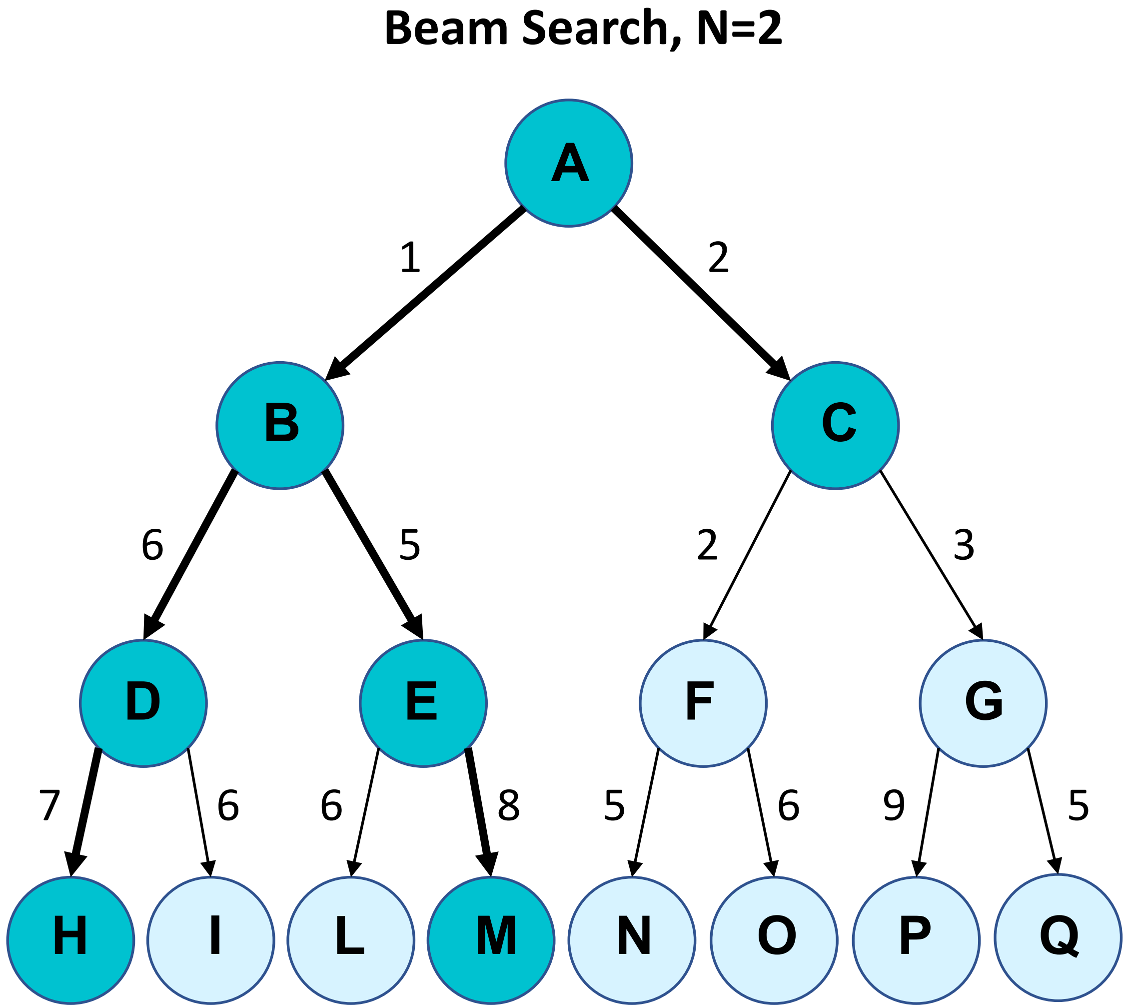


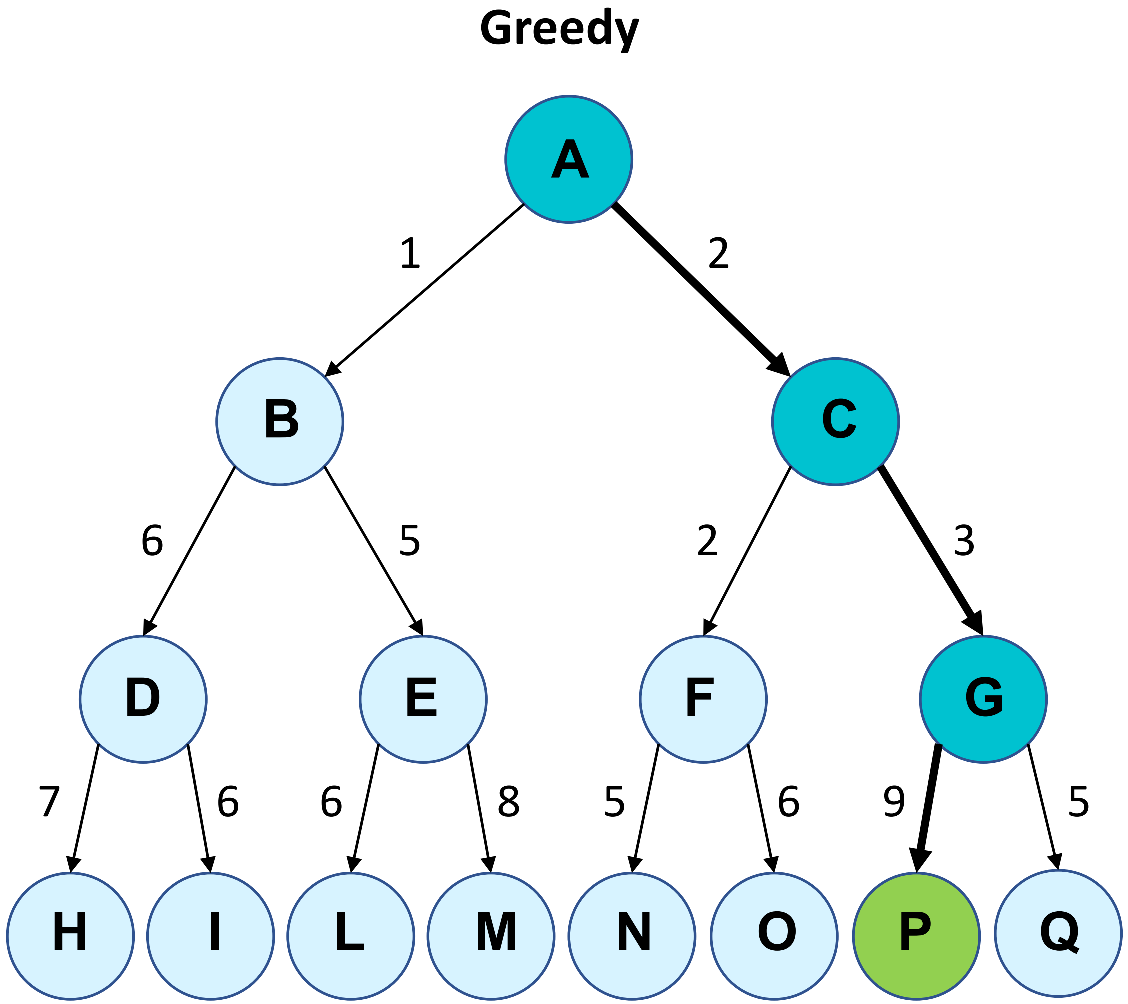
Beam Search vs Greedy



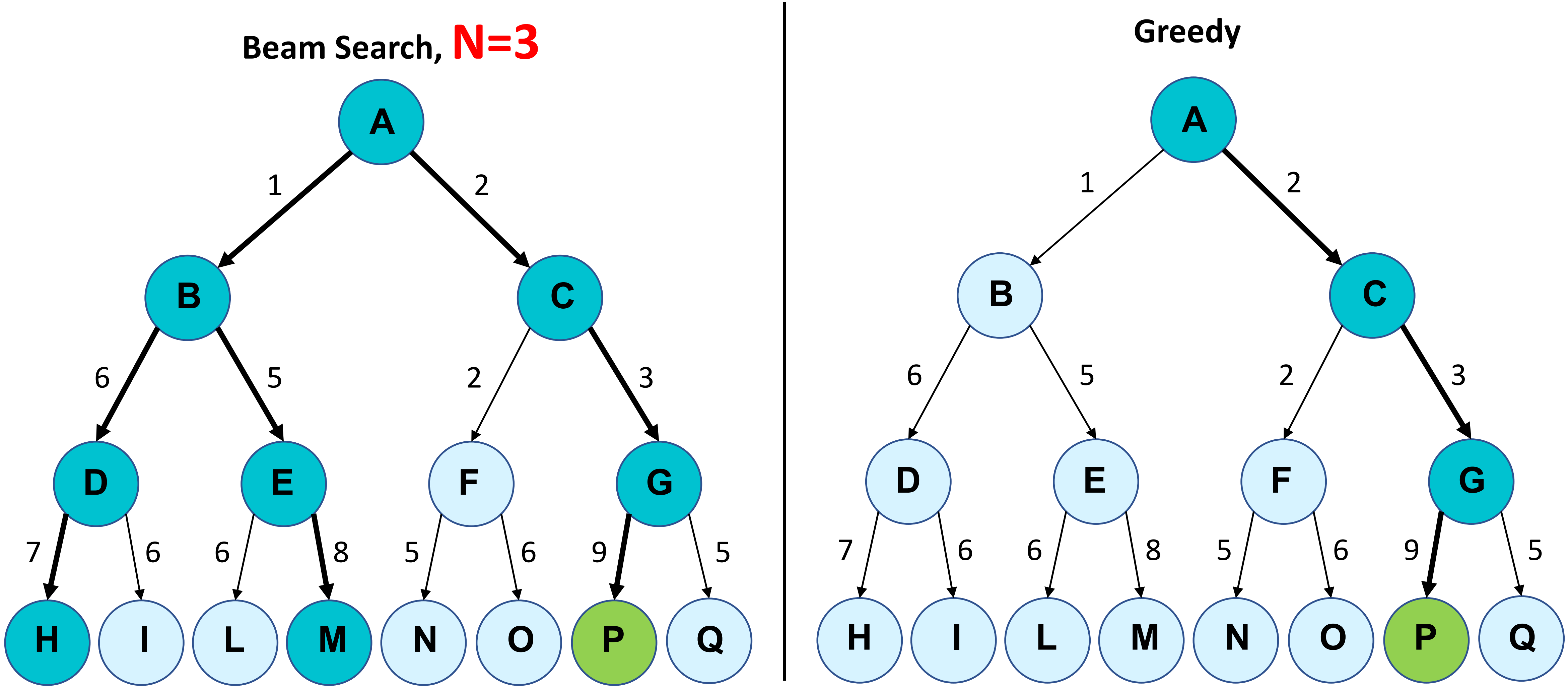


Beam Search vs Greedy





Beam Search vs Greedy



Labeling

[Herholz et al., EG 2015]

ILP, GraphCut

[Alderighi et al., SIGGRAPH 2019]

Greedy

[Filoscia et al., EG 2020]

Over segmentation + region merging

Graph Cuts

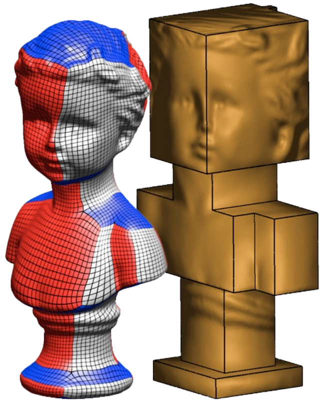
- Solves a multi-labeling problem on a generic graph *G(N,A)* by minimizing

$$L = \arg\min \sum_{i \in N} C_i(l) + \sum_{ij \in A} C_{ij}(l_i, l_j)$$

DATA TERM
cost of assigning
label l to node i

SMOOTH TERM
cost of assigning
labels l_i,l_j to
adjacent nodes i, j

- The problem is NP-Complete
  - finds a local minimum
  - depends on initialization and processing order
  - heavily used in Vision/Graphics
  - it works remarkably well in practice!



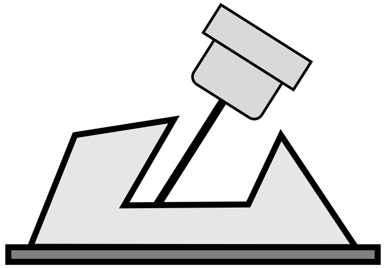
[PolyCut, SIG Asia 2013]

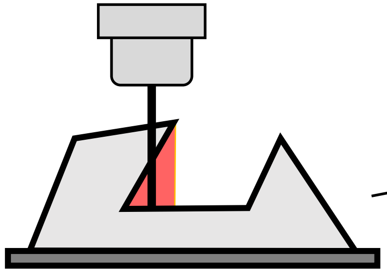
[GrabCut, SIGGRAPH 2004]

Graph Cuts for Digital Manufacturing

- The **graph** is the dual mesh
  - one node per triangle / tetrahedron / voxel
- The **labels** are candidate machining / extraction directions

$$L = \arg\min \sum_{i \in N} C_i(l) + \sum_{ij \in A} C_{ij}(l_i, l_j)$$





DATA TERM

$$\begin{cases} f_i(l) & \text{if feasible} \\ \infty & \text{otherwise} \end{cases}$$

SMOOTH TERM

$$\begin{cases} 0 & \text{if } l_i = l_j \\ f_{ij}(l_i, l_j) & \text{otherwise} \end{cases}$$

Peng Song Ziqi Wang Marco Livesu

Graph Cuts in Digital Manufacturing

- The **graph** is the dual mesh
  - one node per triangle / tetrahedron / voxel
- The **labels** are candidate machining / extraction directions

Surface2Volume

G: dual tetmesh
L: extraction directions

[Araùjo et al., SIGGRAPH 2019]

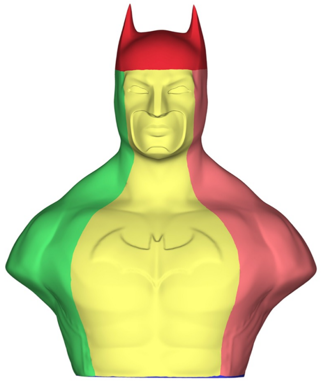
HF Decomp

G: dual trimesh
L: HF directions

[Herholz et al., EG 2015]

4 Axis Milling

G: dual trimesh
L: milling directions



[Nuvoli et al., EG 2021]

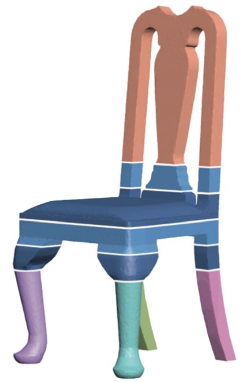
Rigid Molding

G: dual tetmesh
L: molding directions

[Alderighi et al., SIG Asia 2021]

DHF Slicer

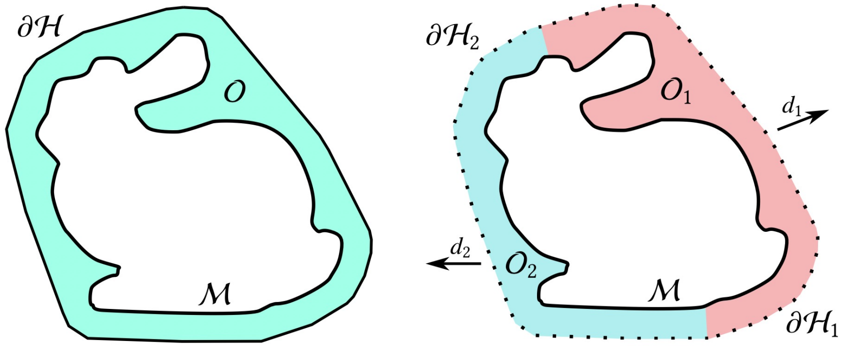
G: dual trimesh
L: DHF directions

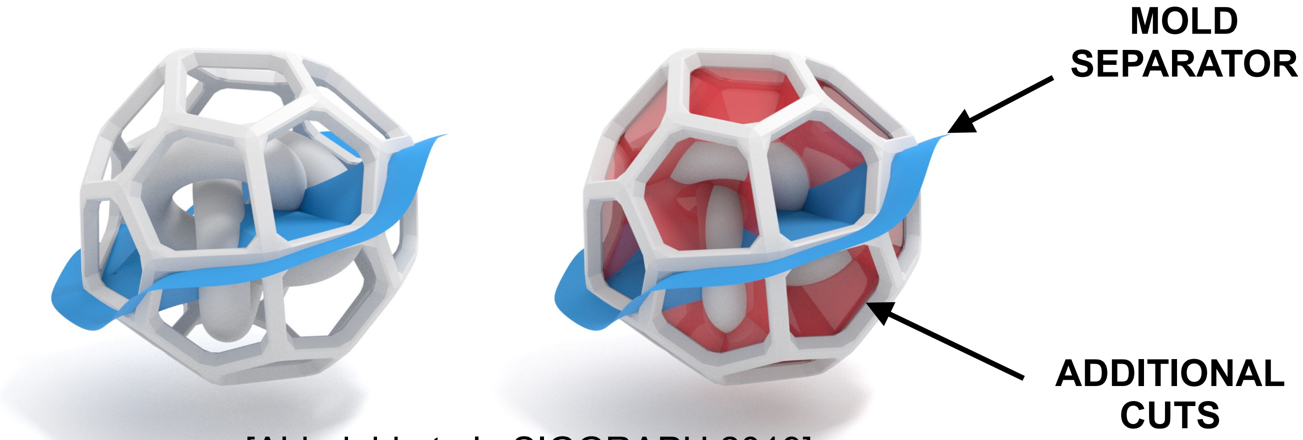


[Yang et al., SIG Asia 2020]

Greedy Labeling for Volume-Aware molding

- Greedy labeling does not typically work as well as graph cuts, but when you have the **right idea**, it becomes extremely powerful!
  - bipartition the **outer shell** finding two mold directions that maximize visual coverage
  - propagate the labeling **inside the volume**, also defining additional cuts

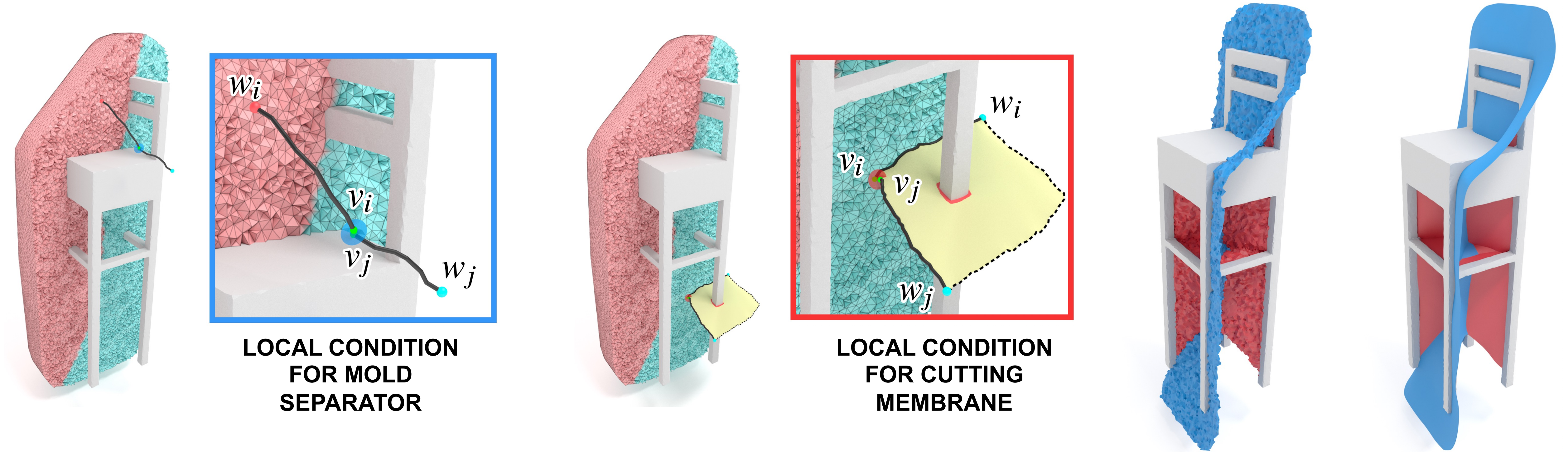




[Alderighi et al., SIGGRAPH 2019]

Greedy Labeling for Volume-Aware molding

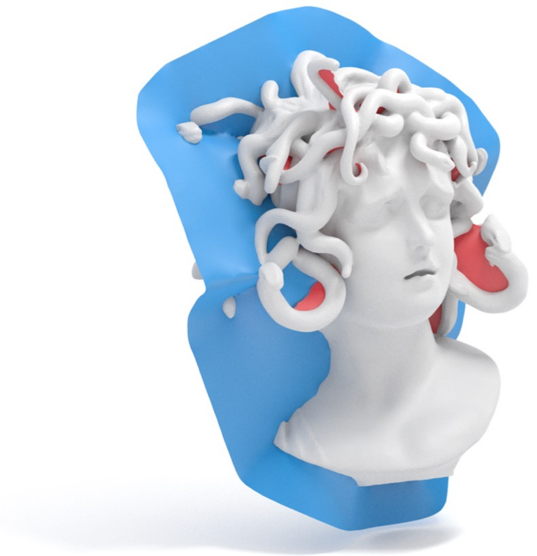
- Greedy labeling does not typically work as well as graph cuts, but when you have the **right idea**, it becomes extremely powerful!

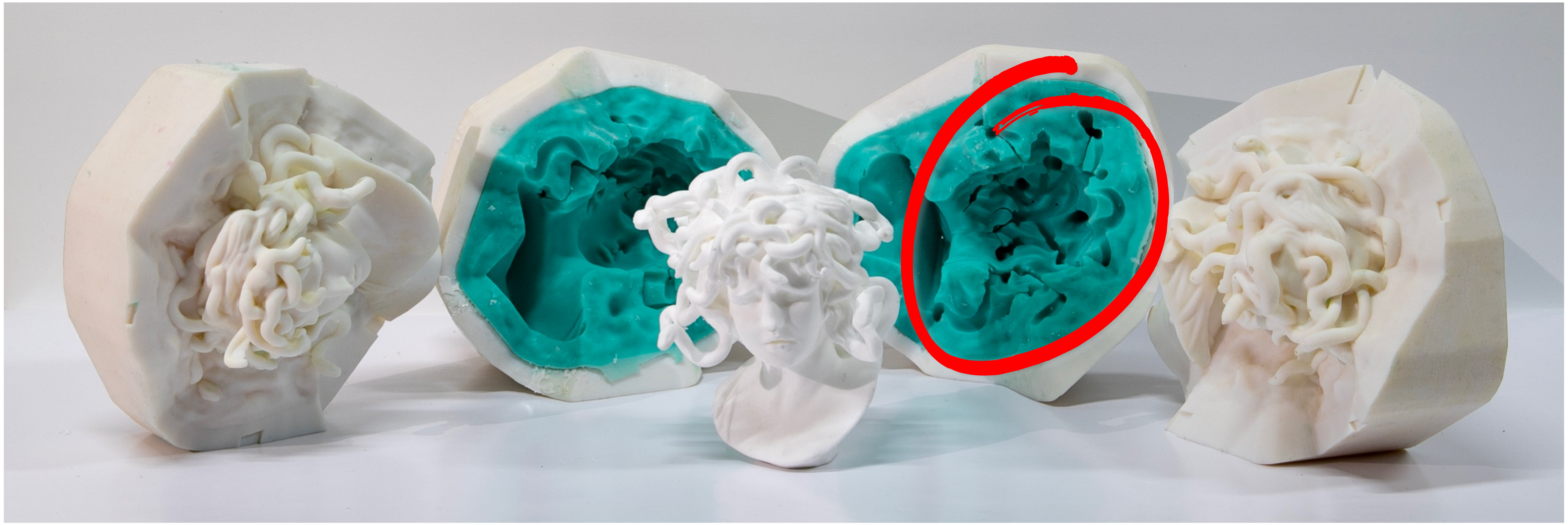


[Alderighi et al., SIGGRAPH 2019]

Greedy Labeling for Volume-Aware molding

- Greedy labeling does not typically work as well as graph cuts, but when you have the **right idea**, it becomes extremely powerful!

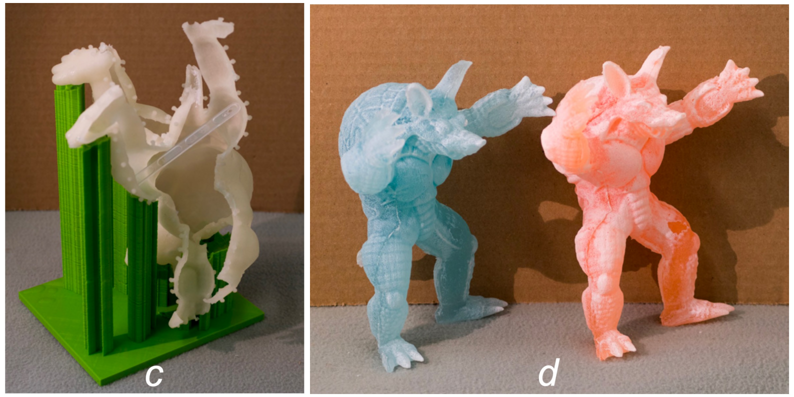


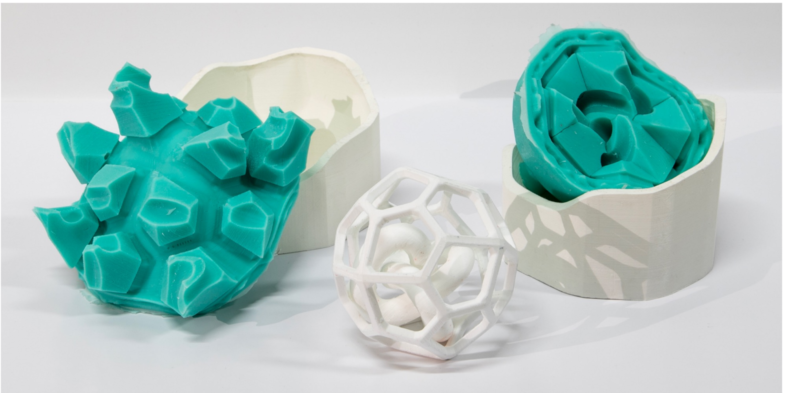


[Alderighi et al., SIGGRAPH 2019]

Greedy Labeling for Volume-Aware molding

- Greedy labeling does not typically work as well as graph cuts, but when you have the **right idea**, it becomes extremely powerful!





time

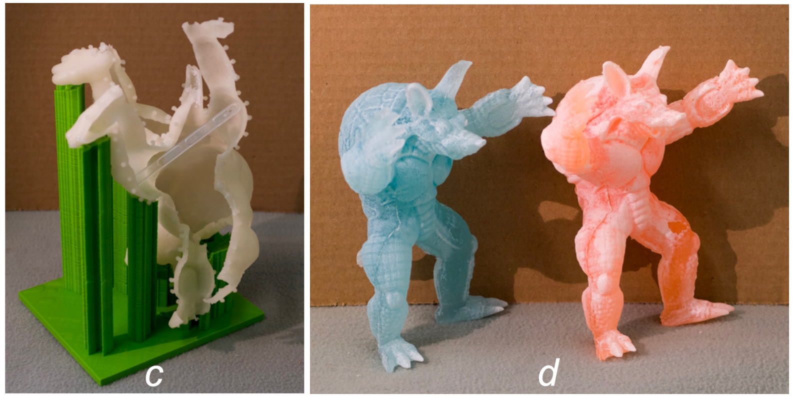
SIGGRAPH Asia 2016
Flex-Molds

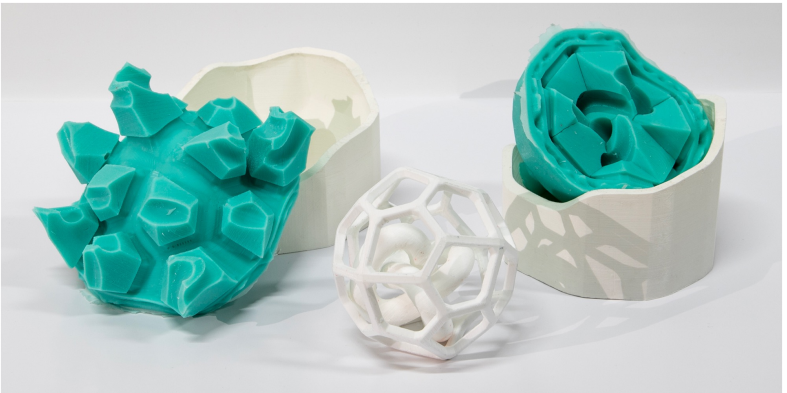
SIGGRAPH 2018
Meta-Molds

SIGGRAPH 2019
Volume-Aware Molds

Greedy Labeling for Volume-Aware molding

- Greedy labeling does not typically work as well as graph cuts, but when you have the **right idea**, it becomes extremely powerful!





time

results

SIGGRAPH Asia 2016
Flex-Molds

SIGGRAPH 2018
Meta-Molds

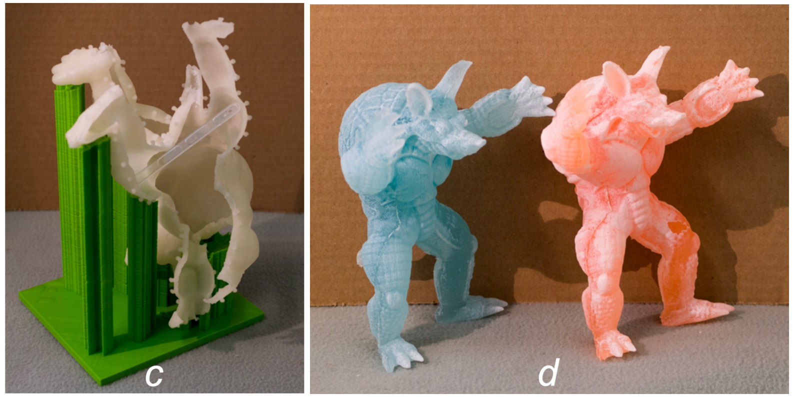
SIGGRAPH 2019
Volume-Aware Molds

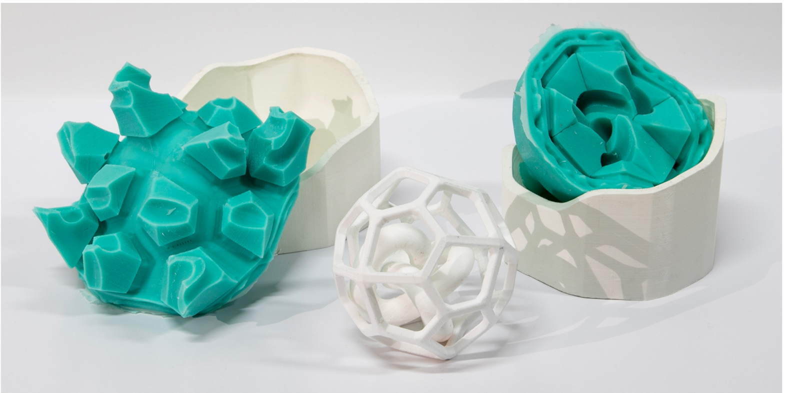
ETH zürich imati

Peng Song Ziqi Wang Marco Livesu

Greedy Labeling for Volume-Aware molding

- Greedy labeling does not typically work as well as graph cuts, but when you have the **right idea**, it becomes extremely powerful!





time

results

complexity

SIGGRAPH Asia 2016
Flex-Molds

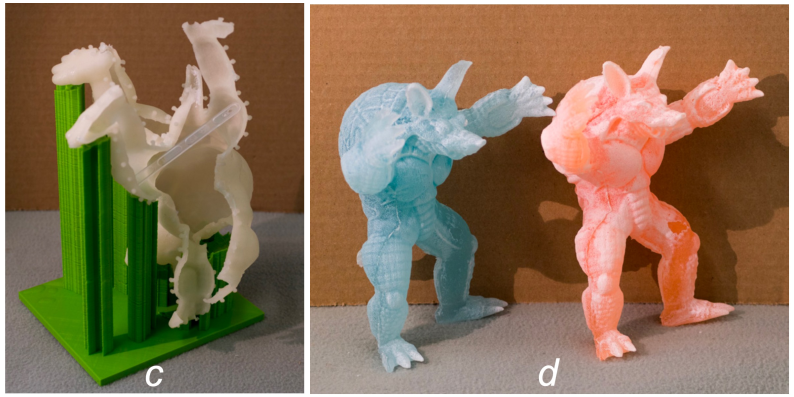
SIGGRAPH 2018
Meta-Molds

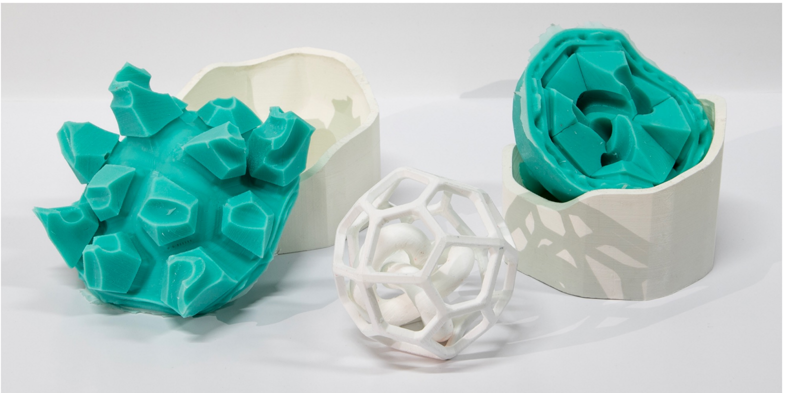
SIGGRAPH 2019
Volume-Aware Molds

Peng Song Ziqi Wang Marco Livesu

Greedy Labeling for Volume-Aware molding

- Greedy labeling does not typically work as well as graph cuts, but when you have the **right idea**, it becomes extremely powerful!





time →

results →

complexity ←

“Simplicity is complexity resolved”

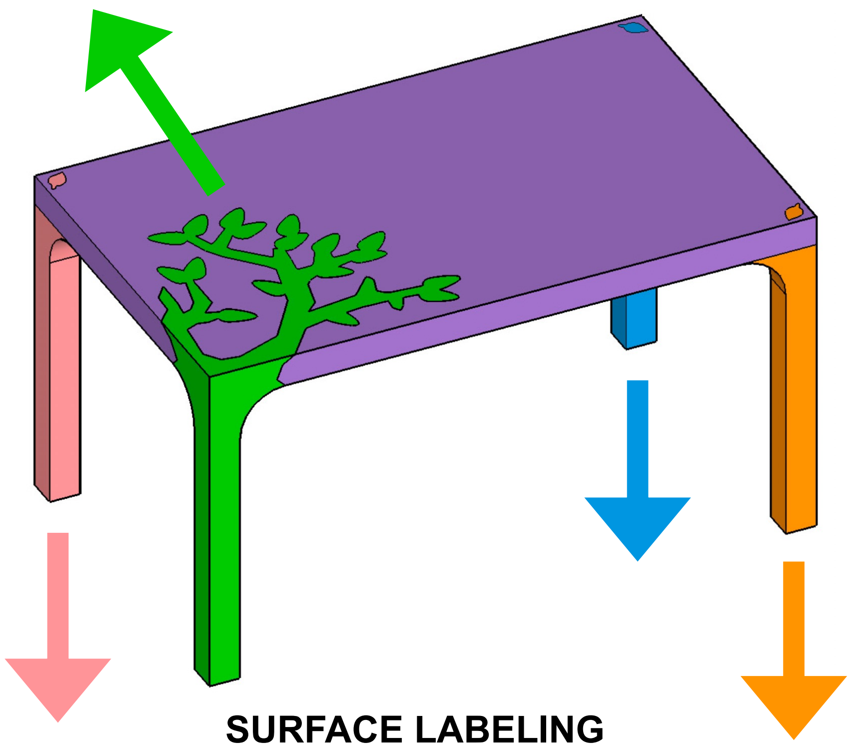
SIGGRAPH Asia 2016
Flex-Molds

SIGGRAPH 2018
Meta-Molds

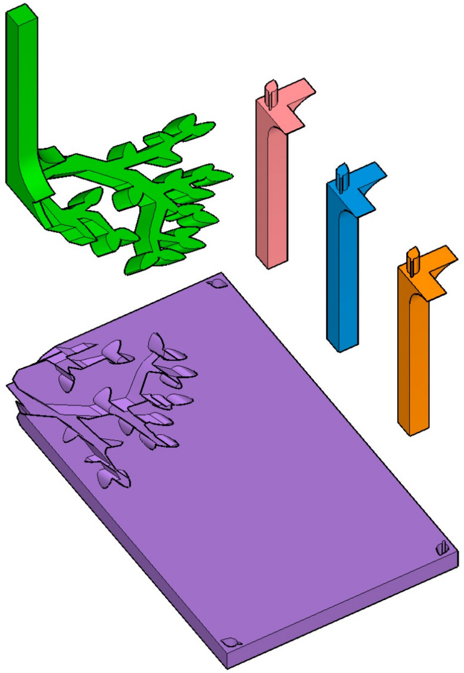
SIGGRAPH 2019
Volume-Aware Molds

Surface Mesh Booleans

- **Extrusion** of surface patches along a feasible extraction direction



SURFACE LABELING



SOLID PIECES (EXTRUDED)

[Yao et al., ACM TOG 2017]

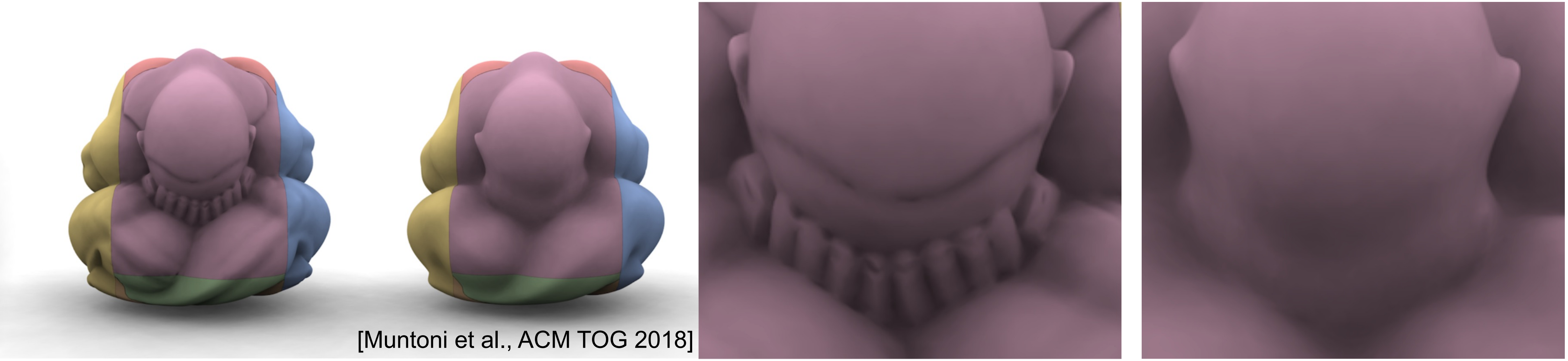
Surface Mesh Booleans

- **Intersection** with an axis aligned box bounding a height field

[Muntoni et al., ACM TOG 2018]

Defeaturing

- Removing small scale features in pre-processing helps reducing part count



[Muntoni et al., ACM TOG 2018]

Input → **Defeature** → **Decompose** → **Restore features**

***without violating constraints**

Defeaturing

- Removing small scale features in pre-processing helps reducing part count

i) Store differential coordinates

$$\delta_i = \frac{1}{|N_i|} \sum_{v_i \in N_i} (v_i - v_j)$$

ii) Apply Smoothing

$$\min \sum_i \|\Delta v_i - \delta_i\|^2$$

subject to
fabrication constraints

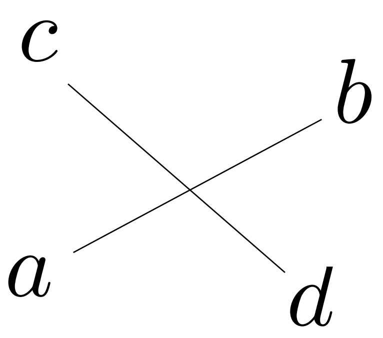
Input → **Defeature** → **Decompose** → **Restore features**

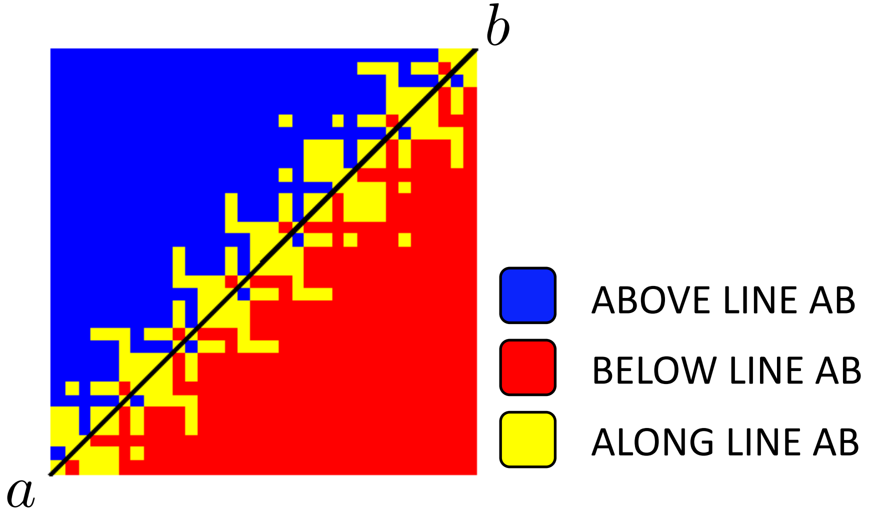
***without violating constraints**

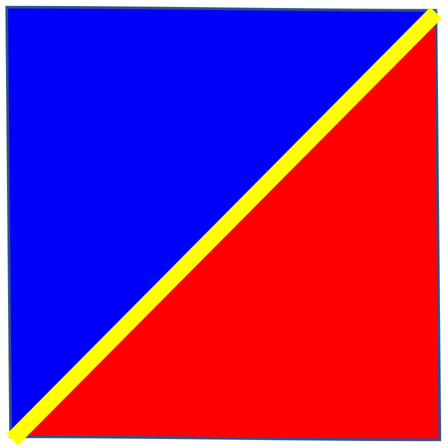
Peng Song Ziqi Wang Marco Livesu

Mesh Booleans: Robustness

- Mesh booleans and planar cuts require finding intersections between mesh elements. This computation is **not robust** in floating point!







Indirect Predicates [Attene 2020]
Predicates Construction Kit [Levy 2015]
Shewchuk Predicates [Shewchuk 1997]
GMP/ CGAL

- Rational numbers or exact predicates are robust, but **slow!**
- Labeling (with interface smoothing) is **float friendly**

To Conclude

- Shape Decomposition for Fabrication comes in **many flavours** and is useful for a variety of different things

- It all boils down to control **two basic ingredients**
  - local surface orientation
  - part size (either static or in motion, for assemblability)

- Always a **hard** problem, but we have good heuristics
  - greedy is not a bad word
  - when the idea is good, greedy algorithms become fast, easy to code and powerful!