Parallel Beam Search for Functionality Partial Matching

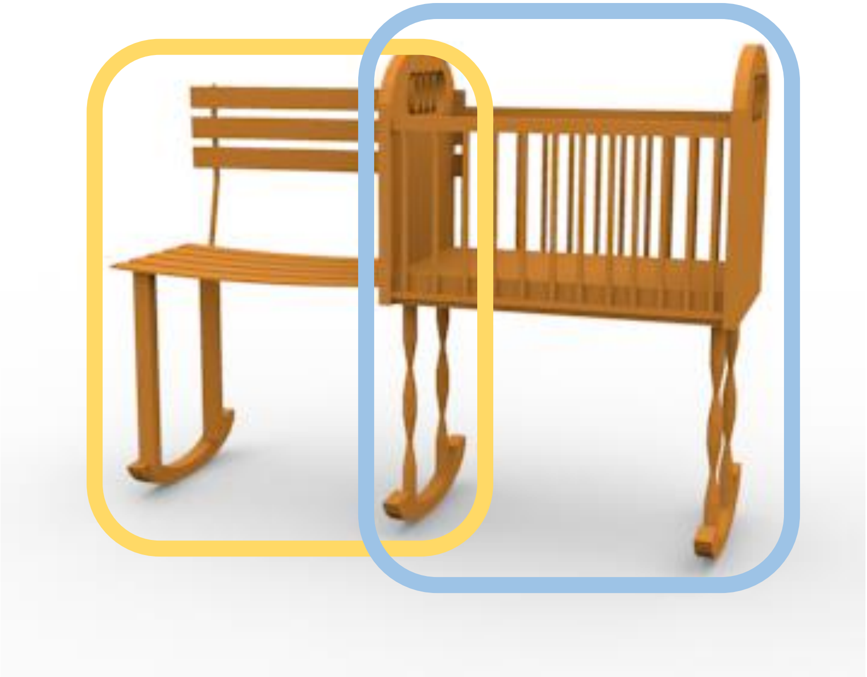
Yanran Guan
yanran.guan@carleton.ca

What is the functionality of an object?

Chair
Sitting + Leaning

Crib
Storage + Rocking

What is functionality partial matching?

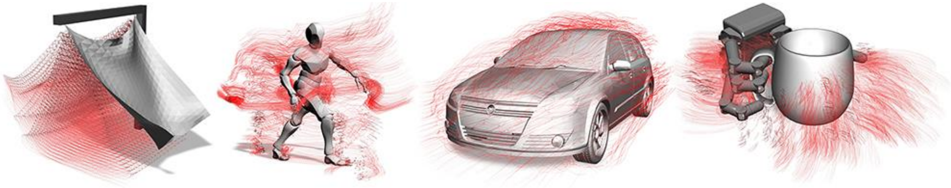


Problem Statement

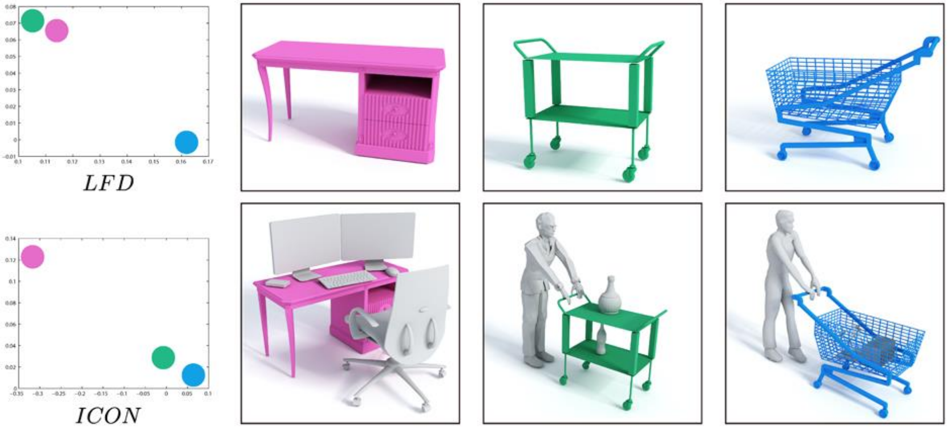
- Develop a parallel beam search algorithm
- Speed up the existing functionality partial matching method

Literature Review

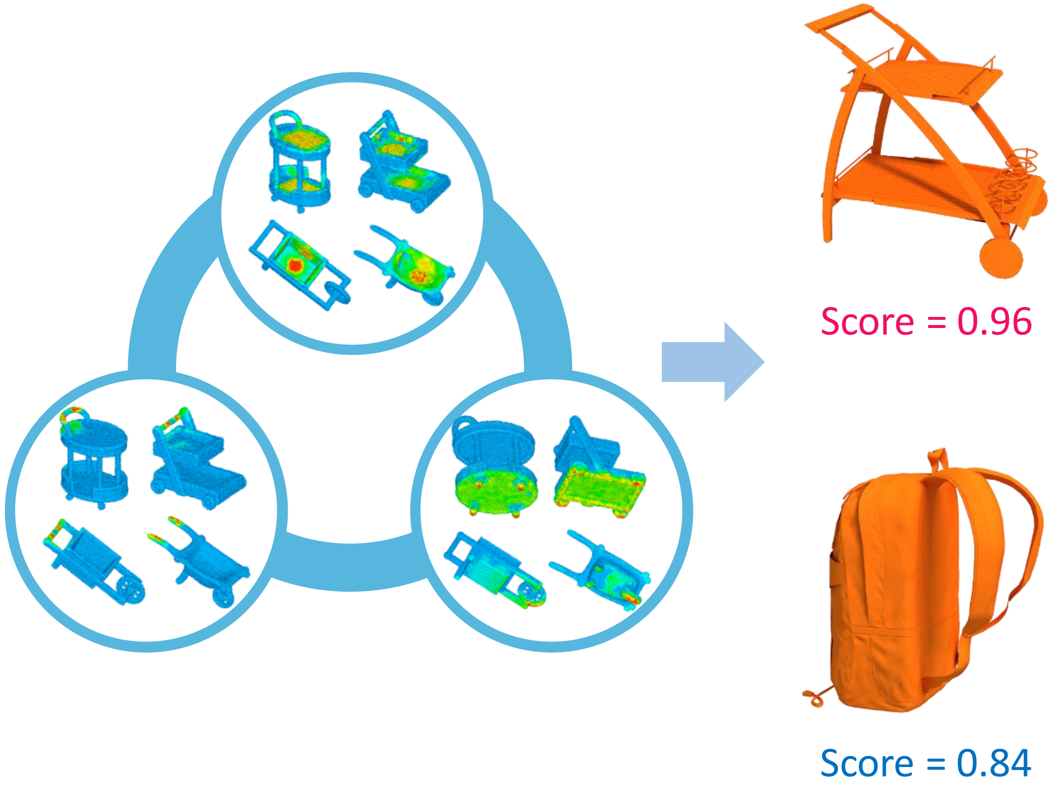
Functionality Analysis in 3D Shape Modeling



Interaction landscapes [8]

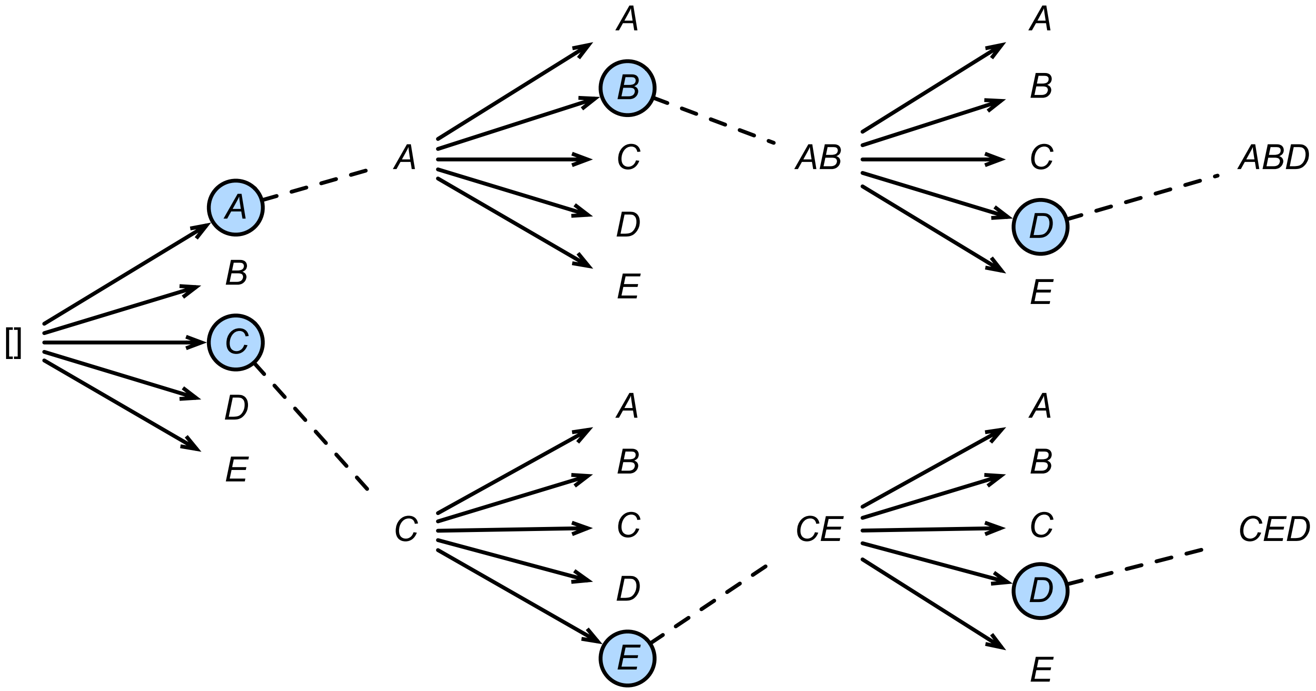


Interaction context (ICON) [5]



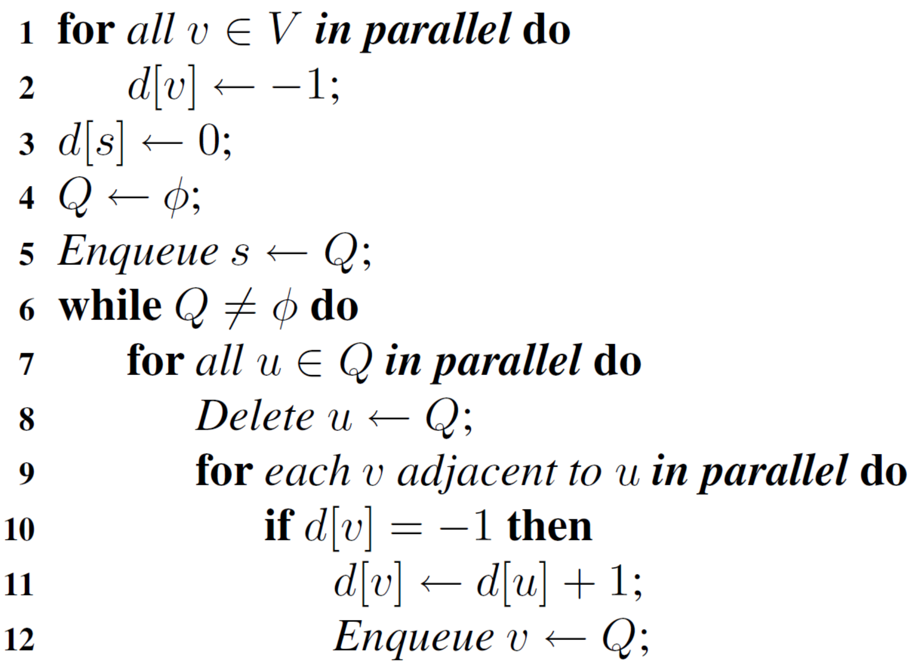
Category functionality model [4]

Beam Search and Its Applications



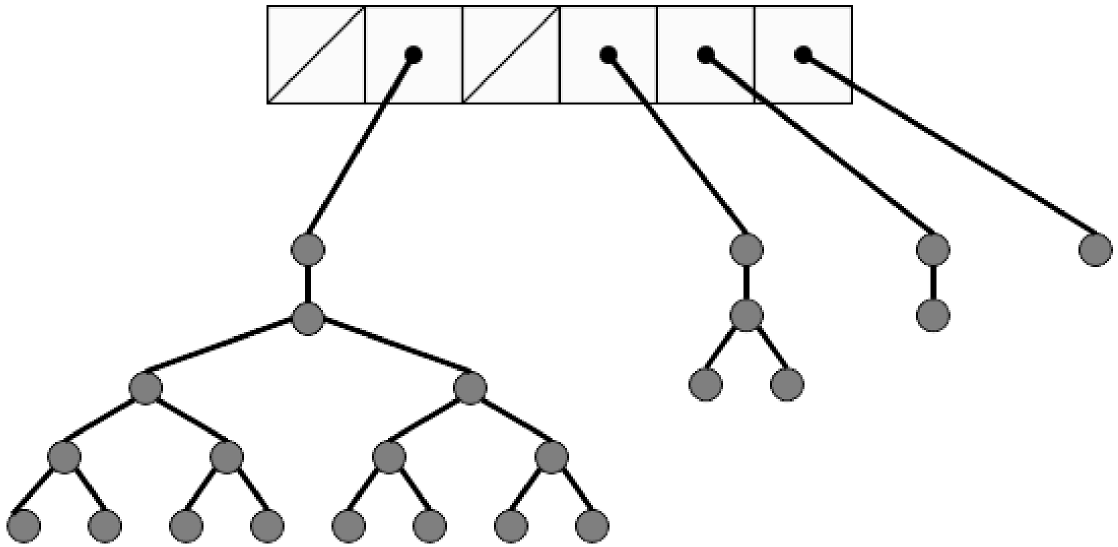
Applied in Speech recognition systems [7, 2], machine translation systems [10], sequence-to-sequence models [9], and so on…

Parallel Breadth-First Search (BFS) Algorithms



```
for all $v \in V$ in parallel do
    $d[v] \leftarrow -1$;
$d[s] \leftarrow 0$;
$Q \leftarrow \phi$;
Enqueue $s \leftarrow Q$;
while $Q \neq \phi$ do
    for all $u \in Q$ in parallel do
        Delete $u \leftarrow Q$;
        for each $v$ adjacent to $u$ in parallel do
            if $d[v] = -1$ then
                $d[v] \leftarrow d[u] + 1$;
                Enqueue $v \leftarrow Q$;
```

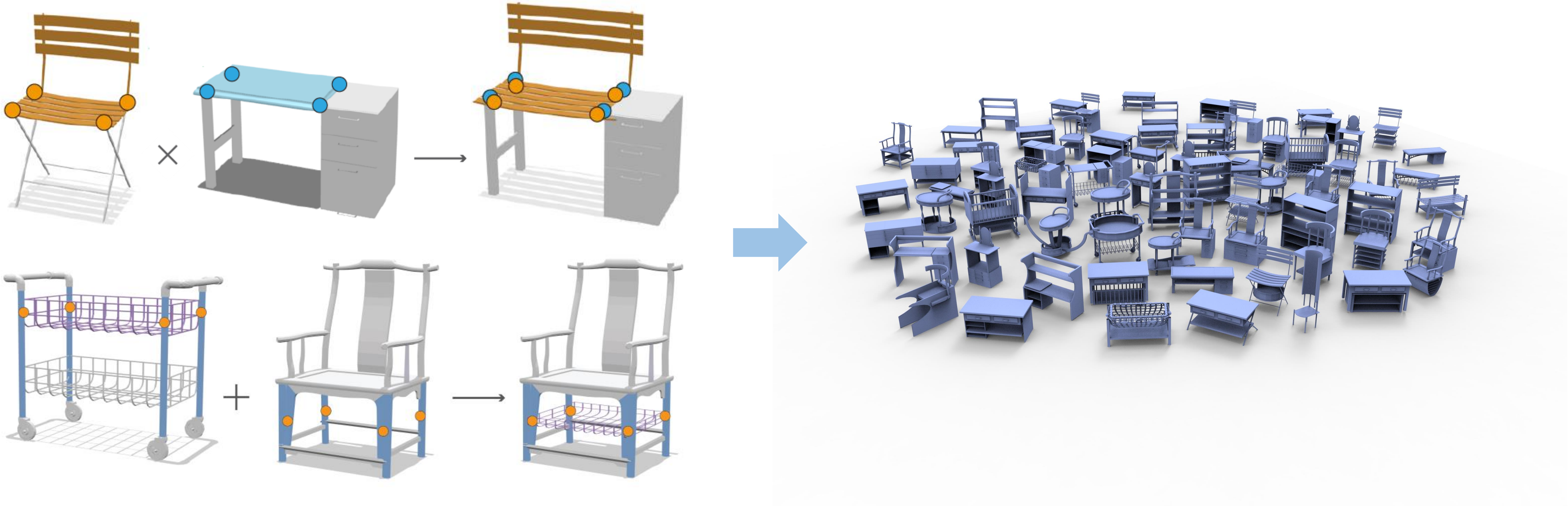
Level synchronization [1, 11]



Multi-set data structure [6]

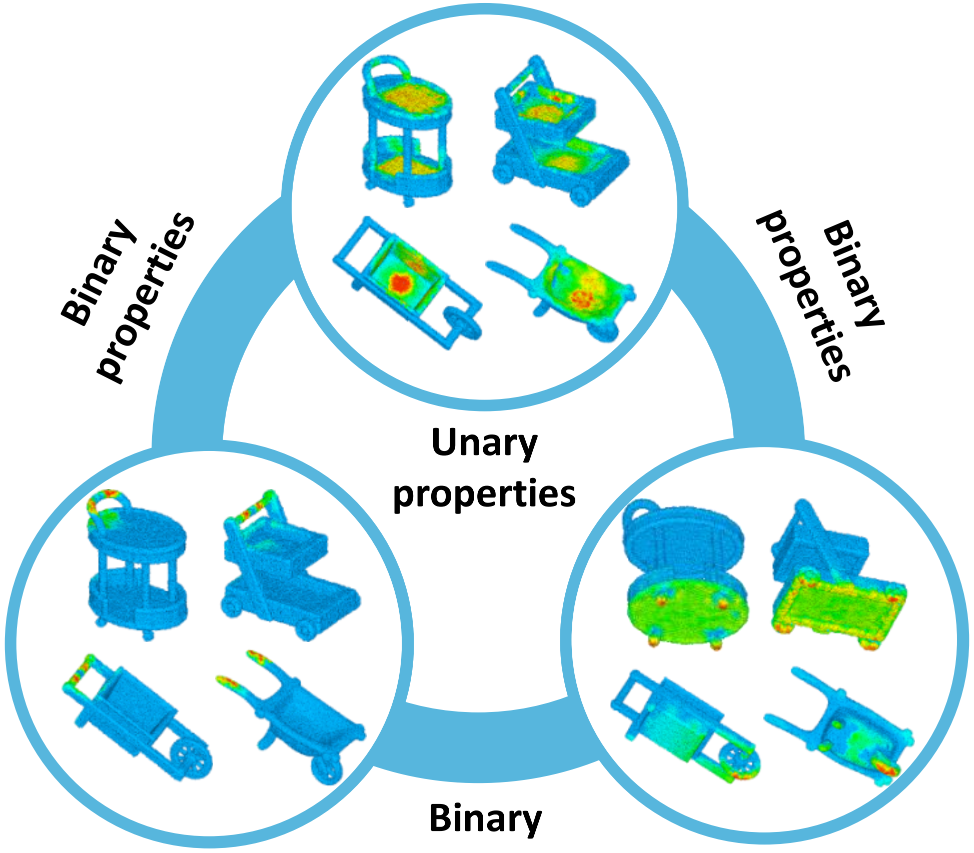
Our Method

Input: Hybrid Shapes



Shape hybridization via evolution [3]

Category Functionality Model



Binary properties

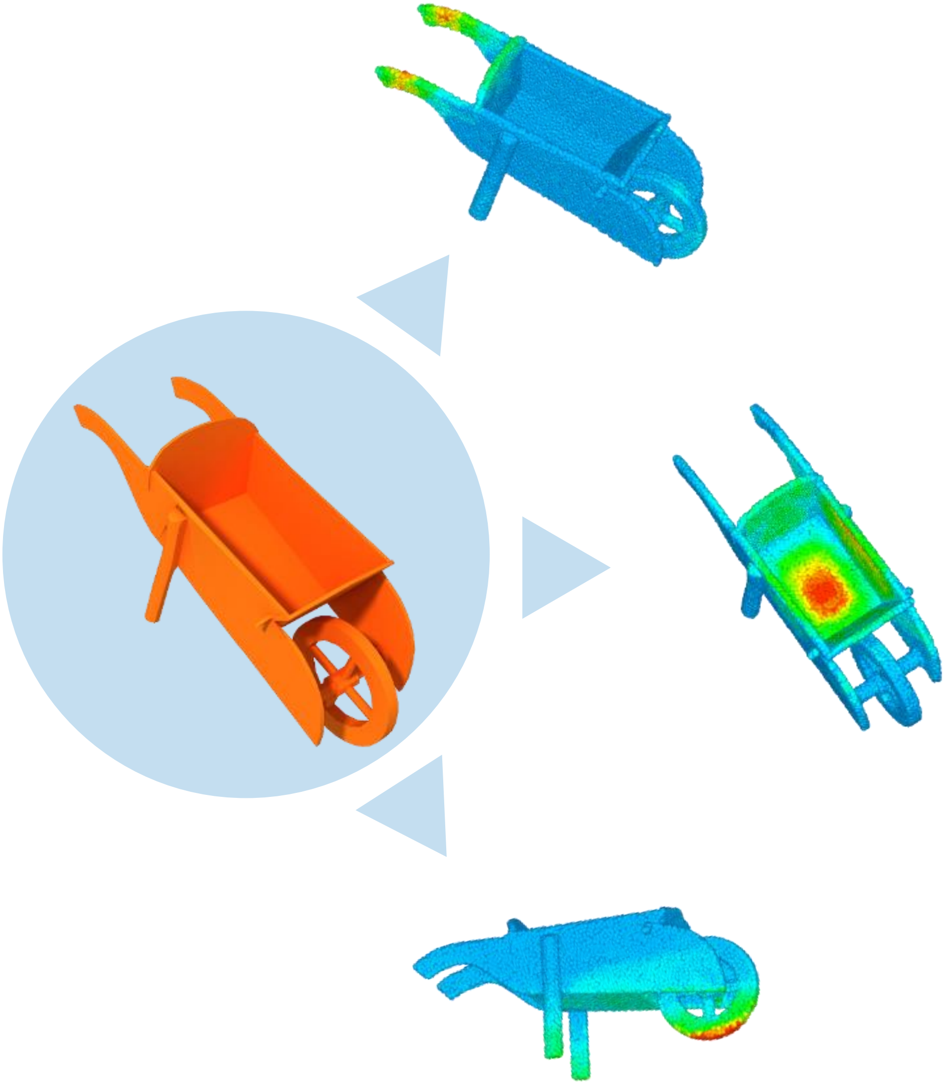
Unary properties

Binary properties

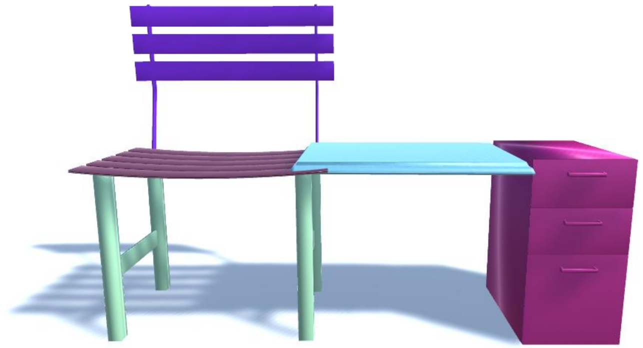
Unary properties

Binary properties

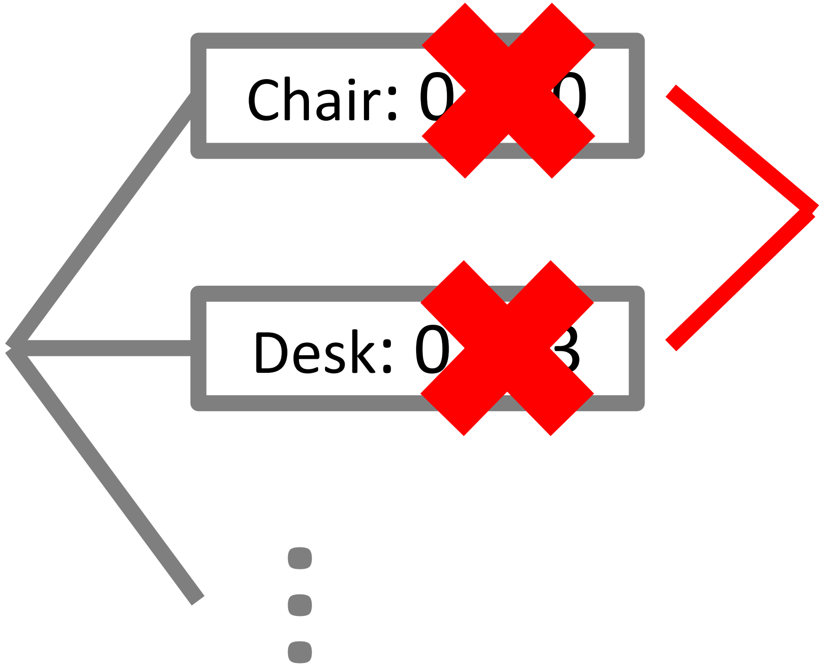
Unary properties



Category Functionality Model



Hybrid shape



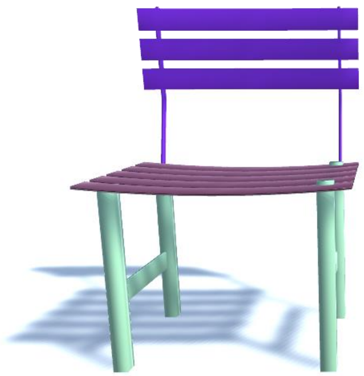
Chair: 0 0

Desk: 0 3

category scores

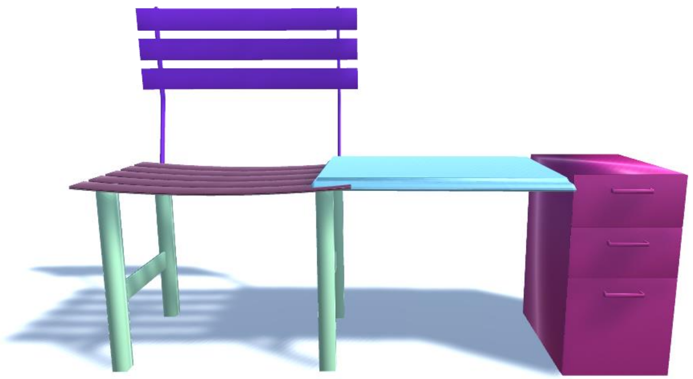
Category models [4]

Functionality Partial Matching



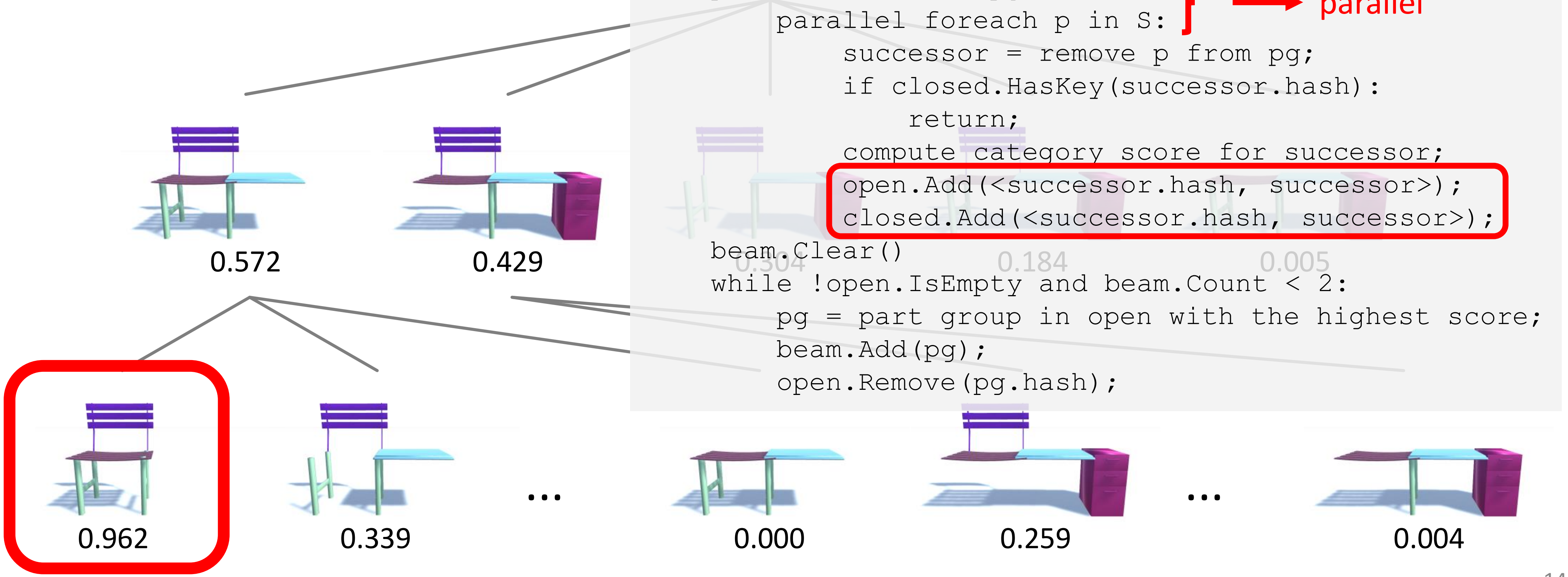
Chair: 0.962

Desk: 0.988



Final score: 0.988

Parallel Beam Search

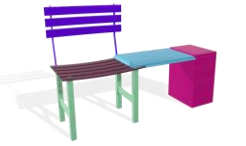
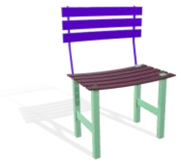
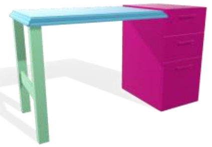
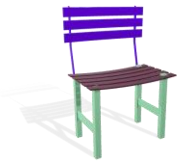
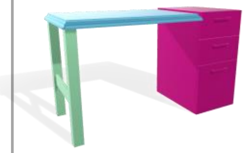
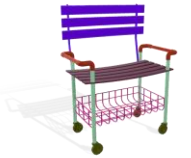
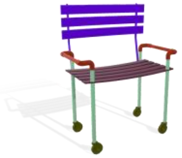
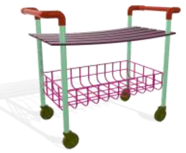
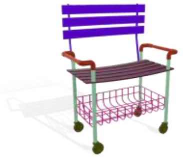
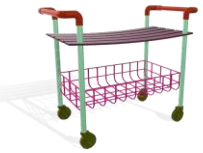
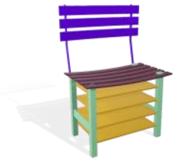
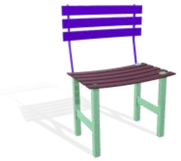
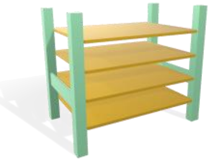
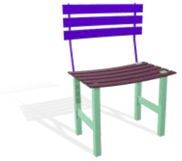
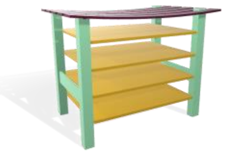
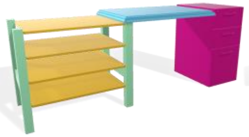
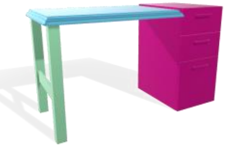
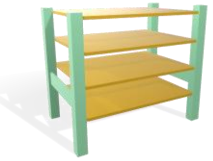
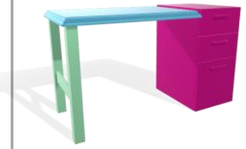
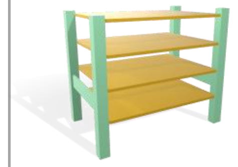
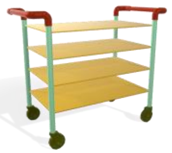
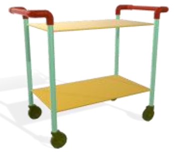
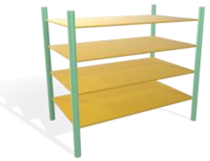
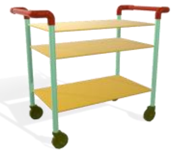
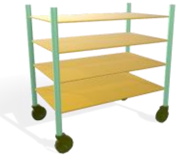


```
beam = {S};
closed = {<S.hash, S>};
while beam.Count > 0:
    open = {};
    parallel foreach pg in beam:
        parallel foreach p in S:
            successor = remove p from pg;
            if closed.HasKey(successor.hash):
                return;
            compute category score for successor;
            open.Add(<successor.hash, successor>);
            closed.Add(<successor.hash, successor>);
    beam.Clear()
    while !open.IsEmpty and beam.Count < 2:
        pg = part group in open with the highest score;
        beam.Add(pg);
        open.Remove(pg.hash);
```

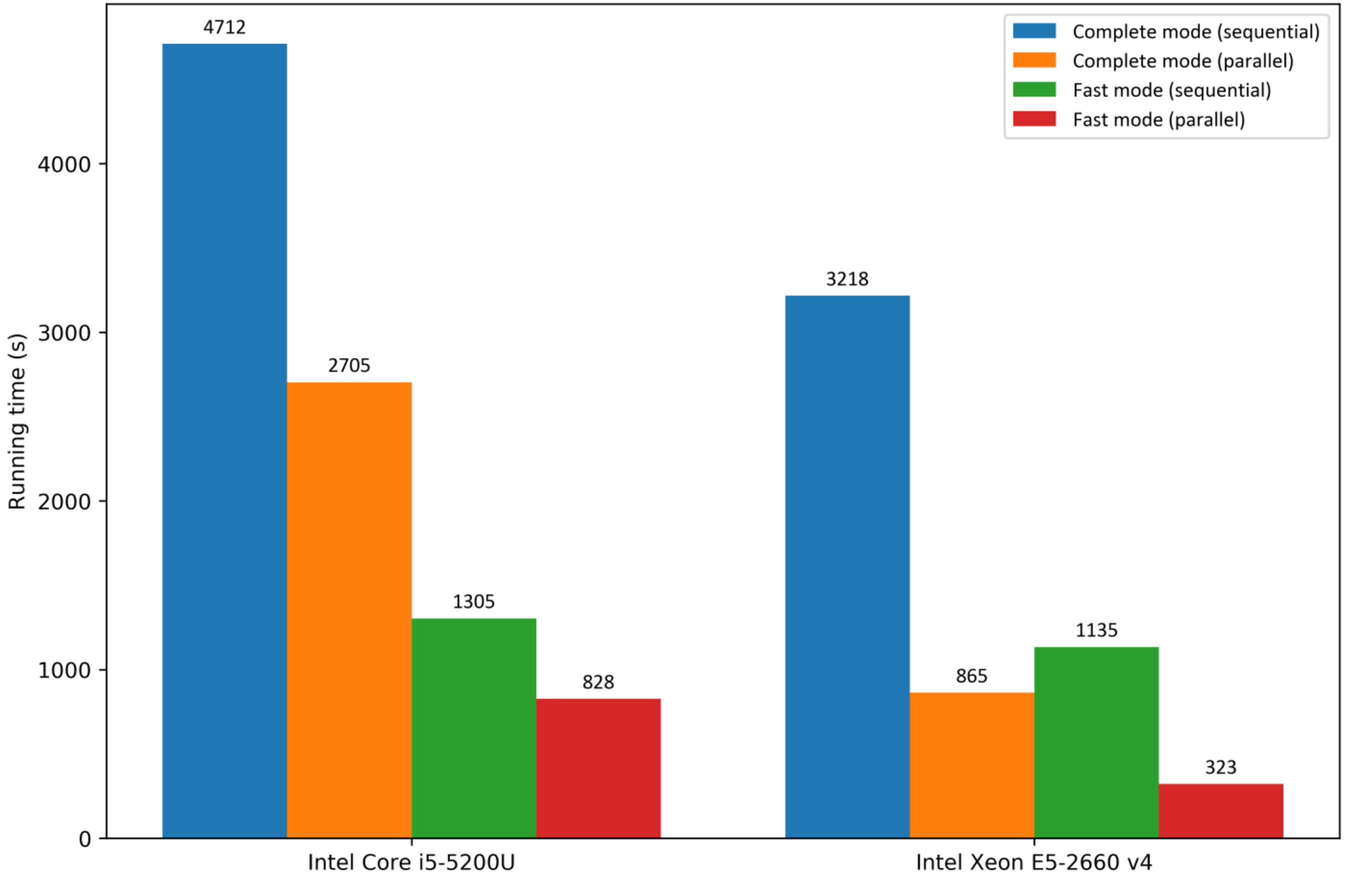
sequential

parallel

Results and Applications

Shape	Combinatorial search		Beam search		Time
	 $s_{\text{chair}} = 0.96$	 $s_{\text{desk}} = 0.98$	 $s_{\text{chair}} = 0.96$	 $s_{\text{desk}} = 0.98$	722s
	 $s_{\text{chair}} = 0.93$	 $s_{\text{handcart}} = 0.98$	 $s_{\text{chair}} = 0.90$	 $s_{\text{handcart}} = 0.98$	1161s
	 $s_{\text{chair}} = 0.96$	 $s_{\text{shelf}} = 0.94$	 $s_{\text{chair}} = 0.96$	 $s_{\text{shelf}} = 0.93$	232s
	 $s_{\text{desk}} = 0.98$	 $s_{\text{shelf}} = 0.96$	 $s_{\text{desk}} = 0.98$	 $s_{\text{shelf}} = 0.96$	1258s
	 $s_{\text{handcart}} = 0.96$	 $s_{\text{shelf}} = 0.97$	 $s_{\text{handcart}} = 0.92$	 $s_{\text{shelf}} = 0.96$	765s

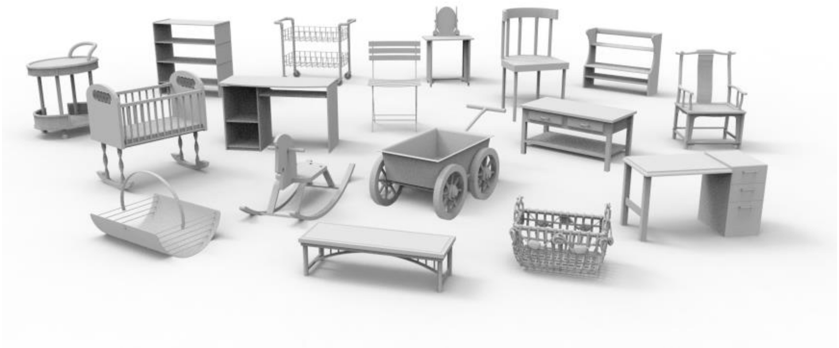
Performance and Scalability



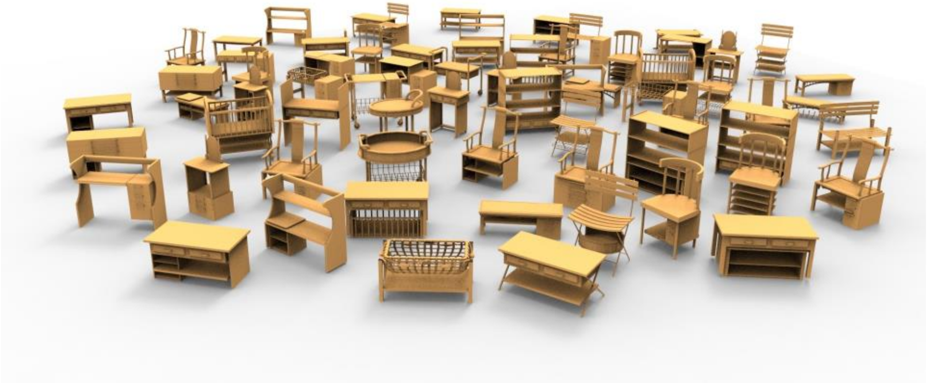
Applications

Functionally plausible

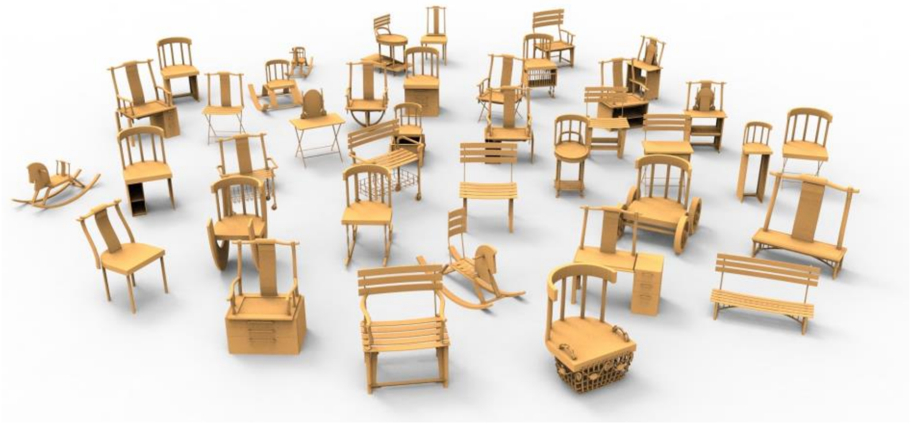
Applications



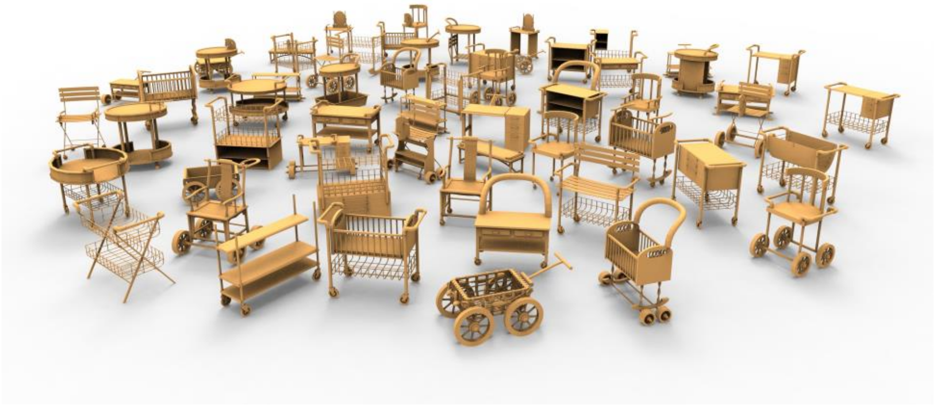
Initial population



Placement + Storage



Sitting + Leaning



Grasping + Rolling

Conclusion and Future Work

- A parallel beam search can largely speed up the process of functionality partial matching
- The implementation of the category functionality model can be further optimized

References

[1] David A. Bader and Kamesh Madduri. Designing multithreaded algorithms for breadth-first search and st-connectivity on the Cray MTA-2. In *ICPP*, 2006.

[2] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *ICASSP*, 2013.

[3] Yanran Guan, Han Liu, Kun Liu, Kangxue Yin, Ruizhen Hu, Oliver van Kaick, Yan Zhang, Ersin Yumer, Nathan Carr, Radomir Mech, and Hao Zhang. FAME: 3D shape generation via functionality-aware model evolution. *IEEE Trans. Vis. Comput. Graph.*, to appear.

[4] Ruizhen Hu, Oliver van Kaick, Bojian Wu, Hui Huang, Ariel Shamir, and Hao Zhang. Learning how objects function via co-analysis of interactions. *ACM Trans. Graph.*, 35(4):47:1–47:12, 2016.

[5] Ruizhen Hu, Chenyang Zhu, Oliver van Kaick, Ligang Liu, Ariel Shamir, and Hao Zhang. Interaction context (ICON): Towards a geometric functionality descriptor. *ACM Trans. Graph.*, 34(4):83:1–83:12, 2015.

[6] Charles E. Leiserson and Tao B. Schardl. A work-efficient parallel breadth-first search algorithm (or how to cope with the nondeterminism of reducers). In *SPAA*, 2010.

[7] Bruce T. Lowerre. *The HARPY Speech Recognition System*. PhD thesis, Carnegie Mellon University, 1976.

[8] Sören Pirk, Vojtech Krs, Kaimo Hu, Suren Deepak Rajasekaran, Hao Kang, Yusuke Yoshiyasu, Bedrich Benes, and Leonidas J. Guibas. Understanding and exploiting object interaction landscapes. *ACM Trans. Graph.*, 36(3):31:1–31:14, 2017.

[9] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.

[10] Christoph Tillmann and Hermann Ney. Word reordering and a dynamic programming beam search algorithm for statistical machine translation. *Comput. Linguist.*, 29(1):97–133, 2003.

[11] Yang Zhang and Eric A. Hansen. Parallel breadth-first heuristic search on a shared-memory architecture. In *AAAI Workshops*, 2006.

Questions

- Why do we need functionality partial matching?
- What is the key idea of parallelizing a beam search?
- Why do we need concurrent open and closed lists in the parallel beam search?