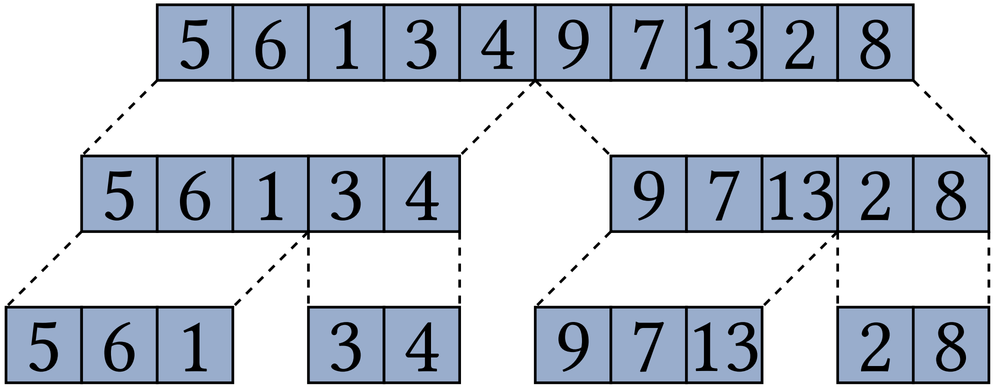
Synthesis of Sorting Kernels

03.03.2025, CGO 2025

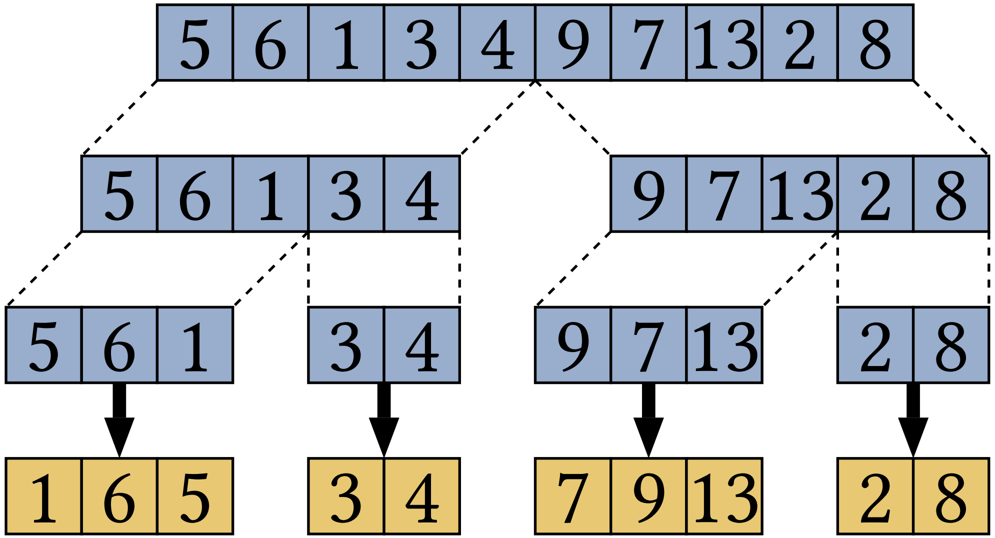
Marcel Ullrich, Sebastian Hack

Saarland University, Saarland Informatics Campus

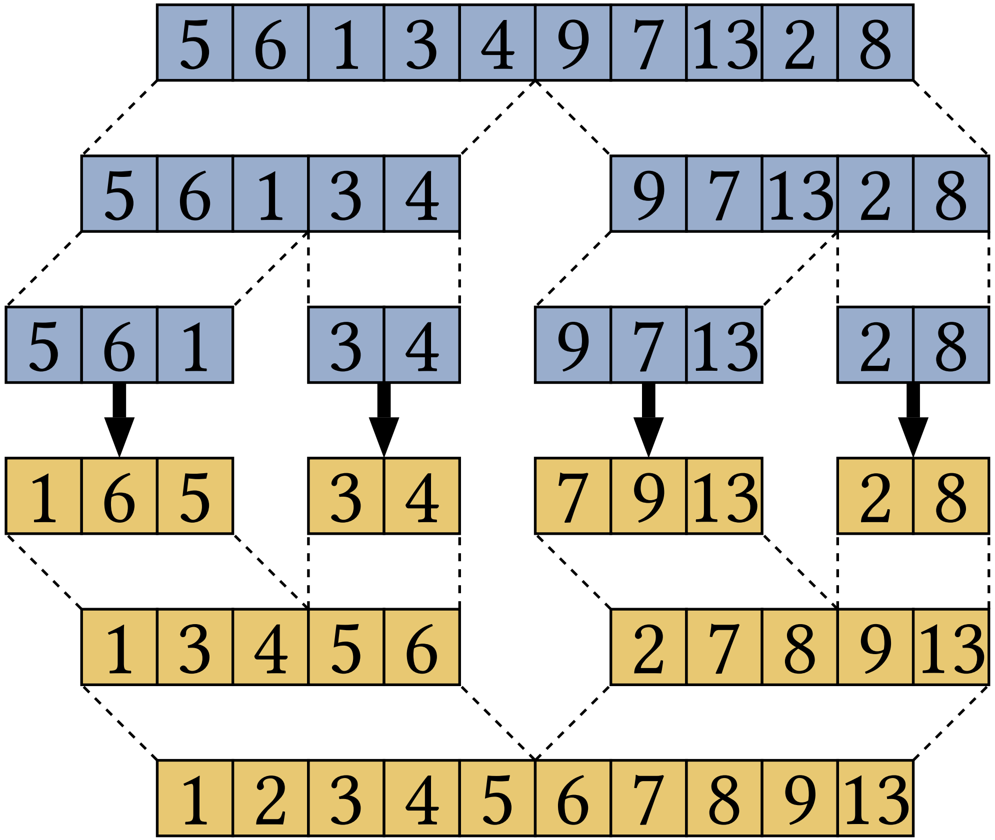
Sorting Kernels



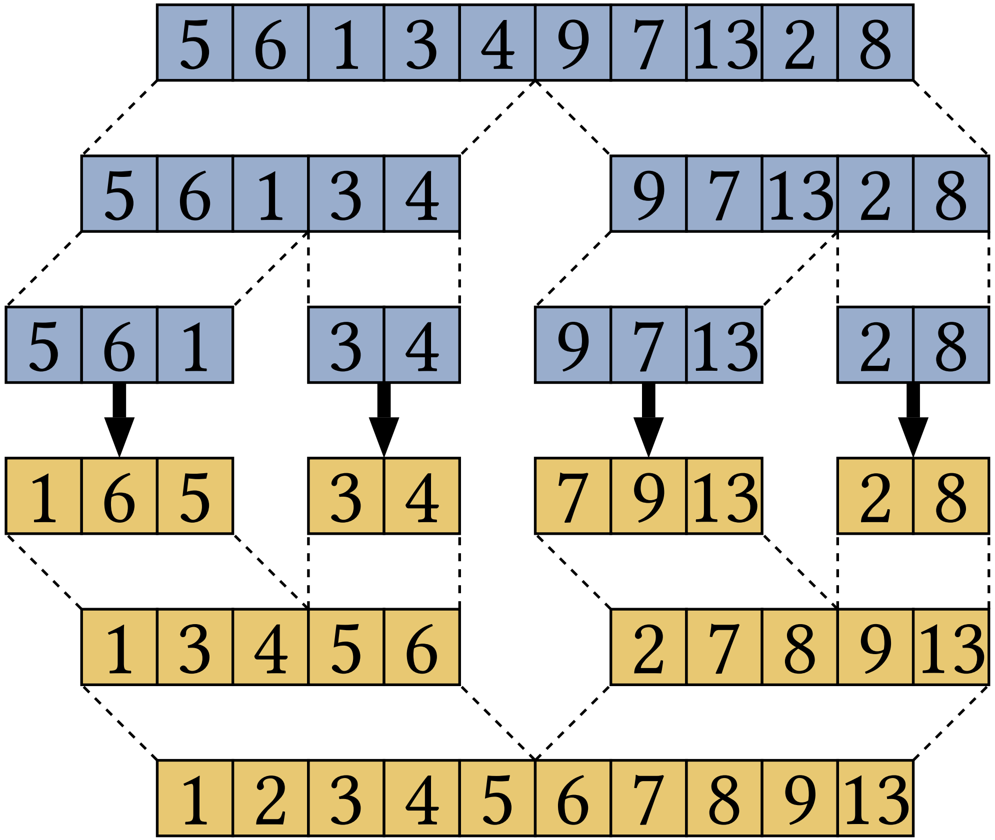
Sorting Kernels

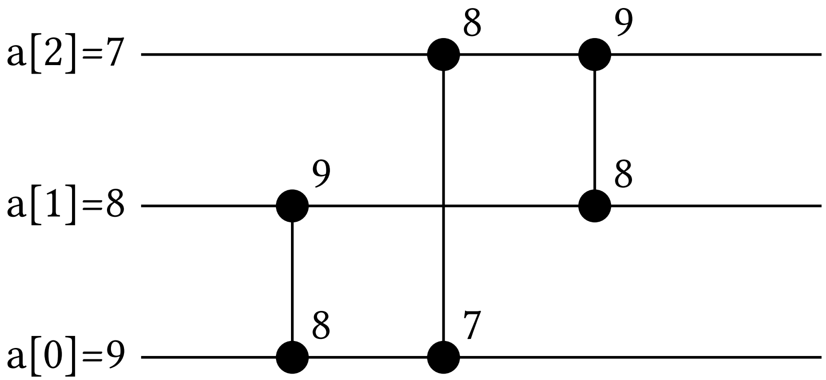


Sorting Kernels

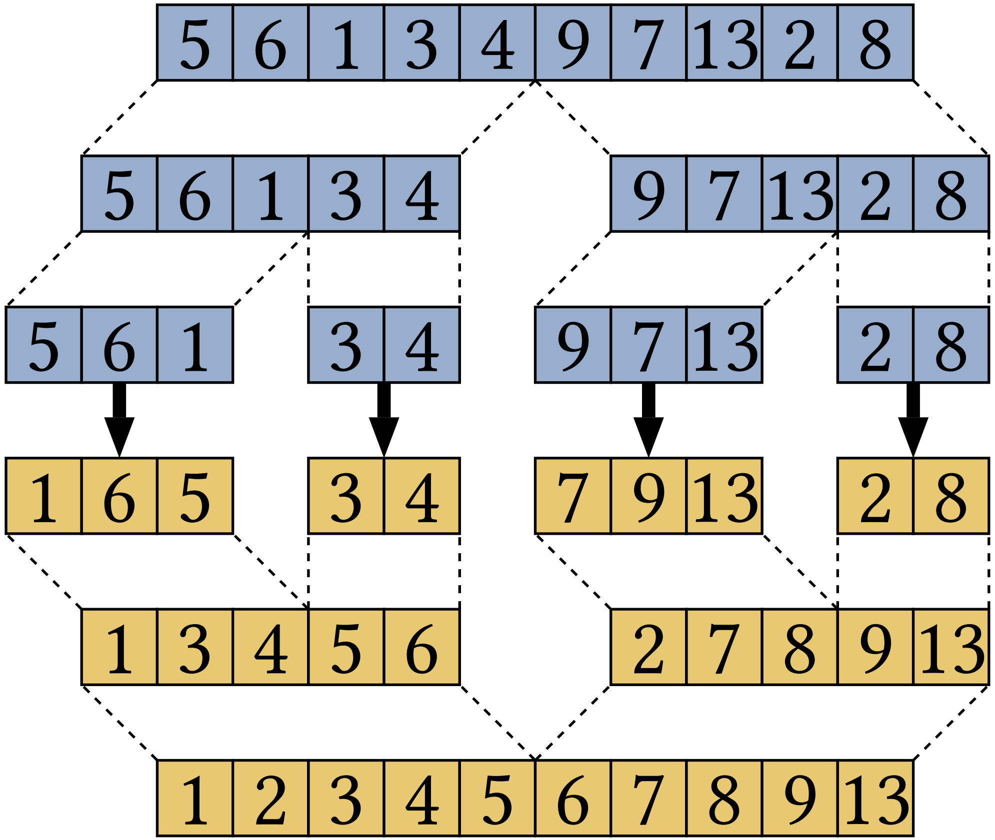


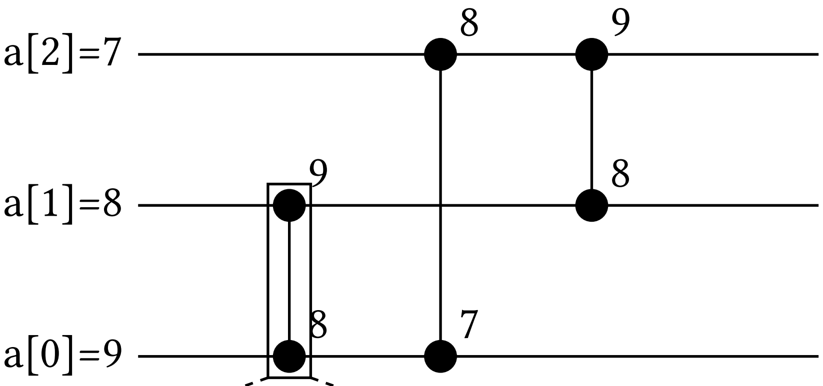
Sorting Kernels





Sorting Kernels






```
mov rdi, rax
cmp rbx, rax
cmovl rax, rbx
cmovl rbx, rdi
```

Model

```
mov s1 r2
cmp r1 r2
cmovg r2 r1
cmovg r1 s1
```

sample from
{mov, cmovl, cmovg, cmp}

register		swap	lt	gt
13	9	-	-	-
13	9	9	-	-
13	9	9	-	>
13	13	9	-	>
9	13	9	-	>

Search Space

```
mov s1 r2
cmp r1 r2
cmovg r2 r1
cmovg r1 s1
```

sample from
{mov, cmovl, cmovg, cmp}

register		swap	lt	gt
13	9	-	-	-
13	9	9	-	-
13	9	9	-	>
13	13	9	-	>
9	13	9	-	>

```
min(a,min(b,c) =
min(min(max(c,b),a),min(b,c))
```

Search Space

```
  mov s1 r2
  cmp r1 r2
cmovg r2 r1
cmovg r1 s1
```

sample from
{mov, cmovl, cmovg, cmp}

register		swap	lt	gt
13	9	-	-	-
13	9	9	-	-
13	9	9	-	>
13	13	9	-	>
9	13	9	-	>

n	Program Length	Search Space
3	11	$10^{19.9}$
4	20	10^{40}
5	≈ 33	$10^{71.2}$
6	≈ 45	$10^{108.4}$

5602 solutions for $n = 3$

```
min(a,min(b,c) =
min(min(max(c,b),a),min(b,c))
```

Search Space

```
  mov s1 r2
  cmp r1 r2
cmovg r2 r1
cmovg r1 s1
```

sample from {mov, cmovl, cmovg, cmp}

register		swap	lt	gt
13	9	-	-	-
13	9	9	-	-
13	9	9	-	>
13	13	9	-	>
9	13	9	-	>

TSNE-embedding of solutions

n	Program Length	Search Space
3	11	$10^{19.9}$
4	20	10^{40}
5	≈ 33	$10^{71.2}$
6	≈ 45	$10^{108.4}$

5602 solutions for $n = 3$

State of the Art

- sorting network 🐌
-
-

State of the Art

- sorting network 🐌
- handoptimized 🐛
-

State of the Art

- sorting network 🐌
- handoptimized 🐛
- 2024 AlphaDev[1]
  - $n = 3$: 6min
  - $n = 4$: 30min
  - $n = 5$: 17.5h

[1] Mankowitz, Daniel J., et al. "Faster sorting algorithms discovered using deep reinforcement learning." Nature 618.7964 (2023): 257-263.

State of the Art

- sorting network 🐌
- handoptimized 🐛
- ~~2024 AlphaDev~~
  - $n = 3$: ~~6min~~ 97ms
  - $n = 4$: ~~30min~~ 2.4s
  - $n = 5$: ~~17.5h~~ 11min

faster synthesis

State of the Art

- sorting network 🐌
- handoptimized 🐛
- ~~2024 AlphaDev~~
  - $n = 3$: ~~6min~~ 97ms
  - $n = 4$: ~~30min~~ 2.4s
  - $n = 5$: ~~17.5h~~ 11min

faster synthesis

faster sorting kernels

State of the Art

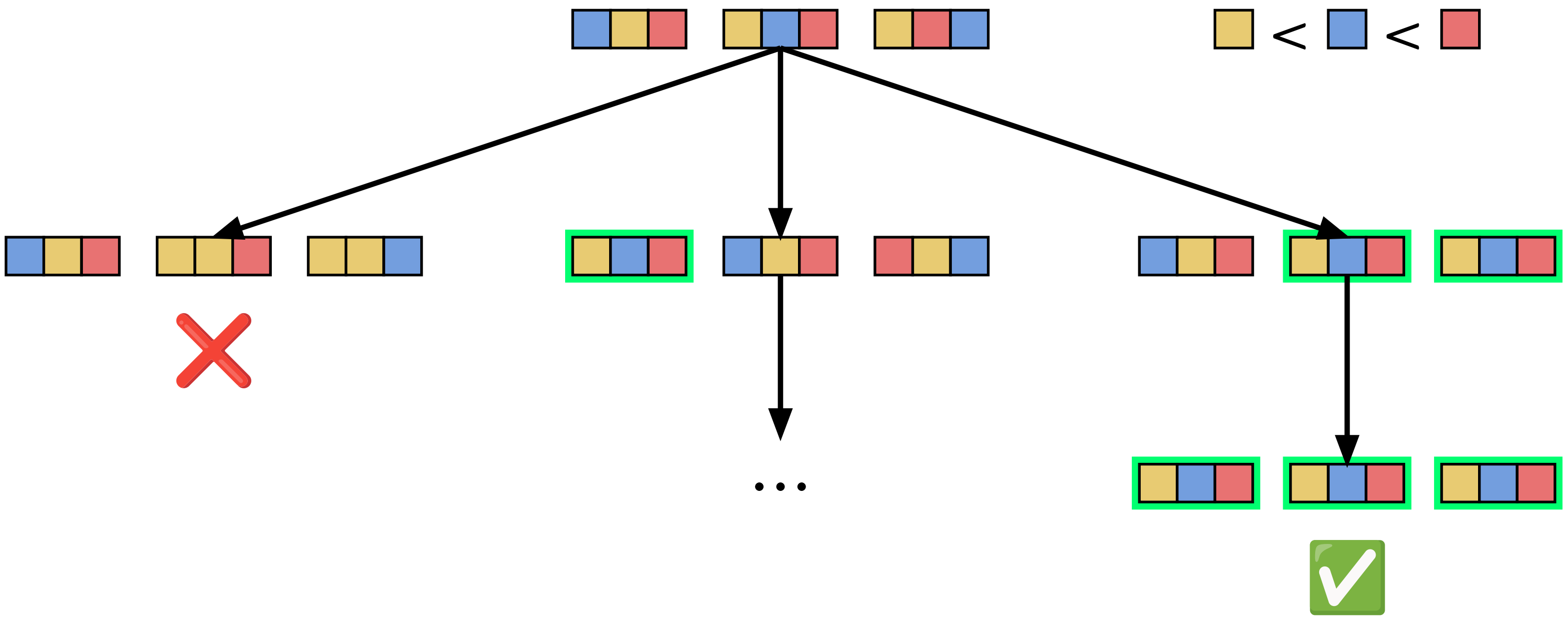
- sorting network 🐌
- handoptimized 🐛
- ~~2024 AlphaDev~~
  - $n = 3$: ~~6min~~ 97ms
  - $n = 4$: ~~30min~~ 2.4s
  - $n = 5$: ~~17.5h~~ 11min

faster synthesis

faster sorting kernels

minimality proof

Enumerative Synthesis



Enumerative Synthesis

1. Select open state
2. Apply Instruction
3. Check for viability
4. Check for solution
5. Cut non-promising
6. Deduplicate states

A⋆ with heuristics:

- permutations
- permutations + scratch register
- delete-relaxed (maximum per permutation)

Enumerative Synthesis

1. Select open state
2. Apply Instruction
3. Check for viability
4. Check for solution
5. Cut non-promising
6. Deduplicate states

Remove redundant/non-sensical:

- `cmp r1 r2;cmp r1 r3`
- `cmp r1 r1`

Restrict to beneficial:

- delete-relaxed
- `cmp r2 r1` $\longrightarrow$ `cmp r1 r2`

Enumerative Synthesis

1. Select open state
2. Apply Instruction
3. Check for viability
4. Check for solution
5. Cut non-promising
6. Deduplicate states

Cut programs:

- number eliminated
- longer than bound/solution
- can not be completed in time

Enumerative Synthesis

1. Select open state
2. Apply Instruction
3. Check for viability
4. Check for solution
5. Cut non-promising
6. Deduplicate states

All permutations already sorted

Enumerative Synthesis

1. Select open state
2. Apply Instruction
3. Check for viability
4. Check for solution
5. Cut non-promising
6. Deduplicate states

Cut if
permutation count > k × best

Cut	Solutions
$k = 1$	222
$k = 1.5$	838
$k = 2$	5602
$k = \infty$	5602

Enumerative Synthesis

1. Select open state
2. Apply Instruction
3. Check for viability
4. Check for solution
5. Cut non-promising
6. Deduplicate states

Hashset-based deduplication of states

Solver-Based Techniques

$$\forall r : P(r) = o \to$$

$$\underbrace{(\forall 1 \le i \le |r| : o_i \le o_{i+1})}_{\text{ascending}} \land$$

$$\underbrace{(\forall x : |\{i : r_i = x\}| = |\{i : o_i = x\}|)}_{\text{same elements}}$$

Solver-Based Techniques

$$r \in \mathrm{Perm}(1..n)$$

$$\forall r : P(r) = o \rightarrow \forall 1 \leq i \leq r : o_i = i$$

Solver-Based Techniques

$$\bigwedge_{r \in \text{Perm}(1..n)} \bigwedge_{1 \leq i \leq n} P(r)_i = i$$

Solver-Based Techniques

$$\bigwedge_{r \in \mathrm{Perm}(1..n)} \bigwedge_{1 \leq i \leq n} P(r)_i = i$$

Heuristics:

- `cmp r1 r2; cmp r2 r3` $\rightarrow$ `cmp r2 r3`
- `cmp r1 r1` $\rightarrow$ `noop`
- `cmp r3 r2` $\rightarrow$ `cmp r2 r3`
- only read initialized
- do not make uncompleteable

Solver-Based Synthesis $n = 3$

SMT	Approach
97min	CEGIS, arbitrary inputs
25min	CEGIS, 1..n
44min	all permutations
–	SyGuS (CVC5, Metalift)

Solver-Based Synthesis $n = 3$

SMT	Approach
97min	CEGIS, arbitrary inputs
25min	CEGIS, 1..n
44min	all permutations
—	SyGuS (CVC5, Metalift)

CP	Approach
—	ILP, MIP
—	CP (MiniZinc other)
232s	chuffed, no heuristic
70s	chuffed, h, $= 1..n$
30s	chuffed, h, $\leq$,1..3

Solver-Based Synthesis $n = 3$

SMT	Approach
97min	CEGIS, arbitrary inputs
25min	CEGIS, 1..n
44min	all permutations
—	SyGuS (CVC5, Metalift)

Planning	Approach
679s	Scorpion planner
216s	Lama planner grounded
3.54s	Lama Planner

CP	Approach
—	ILP, MIP
—	CP (MiniZinc other)
232s	chuffed, no heuristic
70s	chuffed, h, $= 1..n$
30s	chuffed, h, $\leq$,1..3

9

Solver-Based Synthesis $n = 3$

SMT	Approach
97min	CEGIS, arbitrary inputs
25min	CEGIS, 1..n
44min	all permutations
—	SyGuS (CVC5, Metalift)

Planning	Approach
679s	Scorpion planner
216s	Lama planner grounded
3.54s	Lama Planner

CP	Approach
—	ILP, MIP
—	CP (MiniZinc other)
232s	chuffed, no heuristic
70s	chuffed, h, $= 1..n$
30s	chuffed, h, $\leq$,1..3

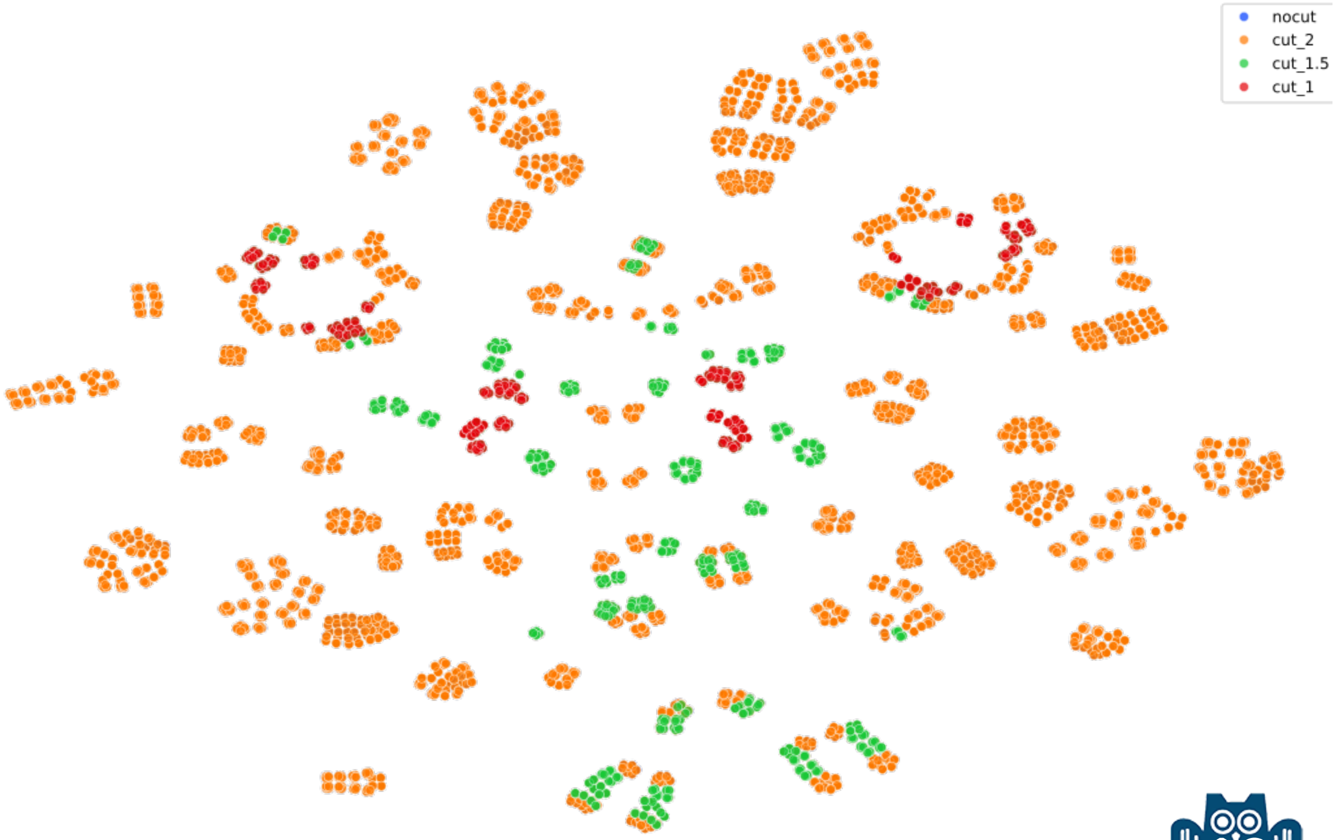
Enum: 97ms

Evaluation Enumeration $n = 3$

Approach	Time
Dijkstra	56s
Dijkstra parallel	17s
Dedup, viable	8.6s
Dedup, A$\star$	1.7s
+viable, instr	0.7s
+cut $k = 1$	0.1s

Evaluation Enumeration $n = 3$

Approach	Time
Dijkstra	56s
Dijkstra parallel	17s
Dedup, viable	8.6s
Dedup, A$\star$	1.7s
+viable, instr	0.7s
+cut $k = 1$	0.1s



Evaluation Enumeration $n \geq 3$

Approach	$l = 11$ $n = 3$	$l = 20$ $n = 4$	$l = 33$ $n = 5$
Enumeration	97ms	2.4s	11min
AlphaDev-RL	6min	30min	17.5h
AlphaDev-S	0.4s	0.6s	5.75h

- All solutions for $n = 3$: 10min
- Optimality for $n = 4$: 2weeks

Evaluation Kernels

Kernel	$n = 3$	$n = 4$	$n = 5$
Enumeration	5.8ms	9.4ms	14.8ms
Mimicry[2]	8.0ms	8.8ms	—
AlphaDev	6.7ms	10.4ms	16.2ms
Sorting Network (Cmp)	7.1ms	14.8ms	19.4ms

[2] Mimicry. 2023. Faster Sorting Beyond DeepMind's AlphaDev. https://www.mimicry.ai/faster-sorting-beyond-deepminds-alphadev Accessed: 2023-09-20

Evaluation Kernels MinMax

Kernel	$n = 3$	$n = 4$	$n = 5$
Enumeration	5.8ms	9.4ms	14.8ms
Mimicry[3]	8.0ms	8.8ms	—
AlphaDev	6.7ms	10.4ms	16.2ms
Sorting Network (Cmp)	7.1ms	14.8ms	19.4ms
MinMax	4.6ms	7.0ms	10.7ms
Sorting Network	5.3ms	8.1ms	12.2ms

```
movdqa %xmm1, %xmm3
pminud %xmm2, %xmm1
pmaxud %xmm3, %xmm2
movdqa %xmm0, %xmm3
pminud %xmm2, %xmm3
pmaxud %xmm0, %xmm2
pminud %xmm1, %xmm0
pmaxud %xmm3, %xmm1
```

[3] Mimicry. 2023. Faster Sorting Beyond DeepMind's AlphaDev. https://www.mimicry.ai/faster-sorting-beyond-deepminds-alphadev Accessed: 2023-09-20

Conclusion

- faster synthesis
- faster sorting kernels
- minimality proof

⇧ Project on GitHub