



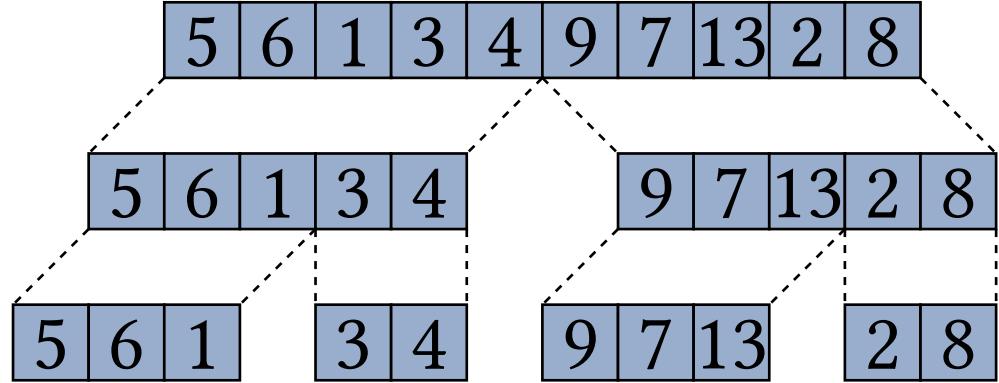
# Synthesis of Sorting Kernels

03.03.2025, CGO 2025

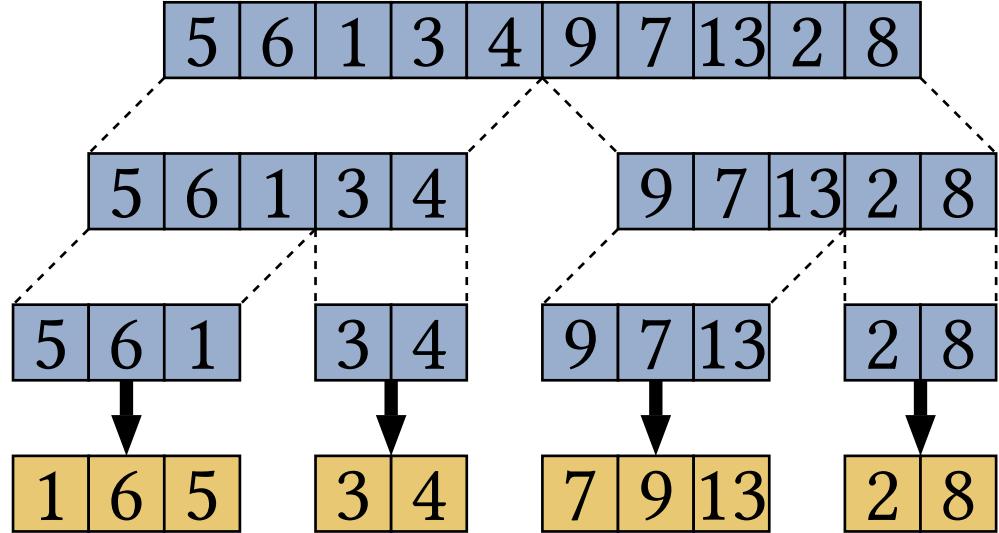
Marcel Ullrich, Sebastian Hack

Saarland University, Saarland Informatics Campus

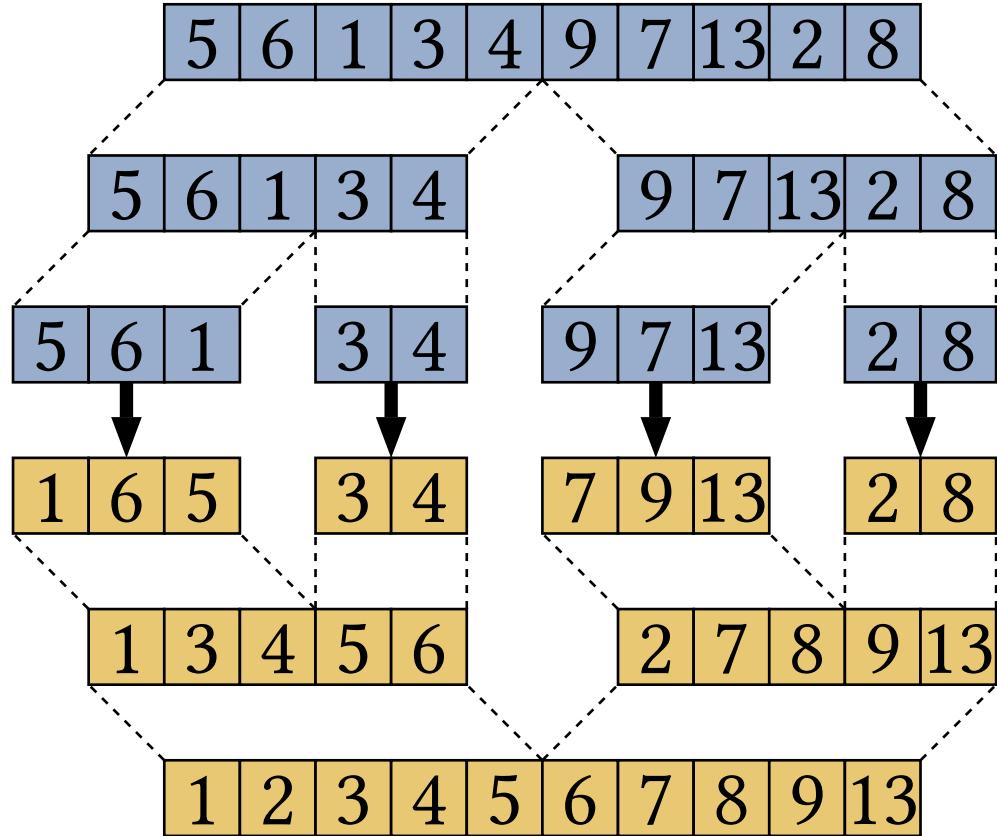
# Sorting Kernels



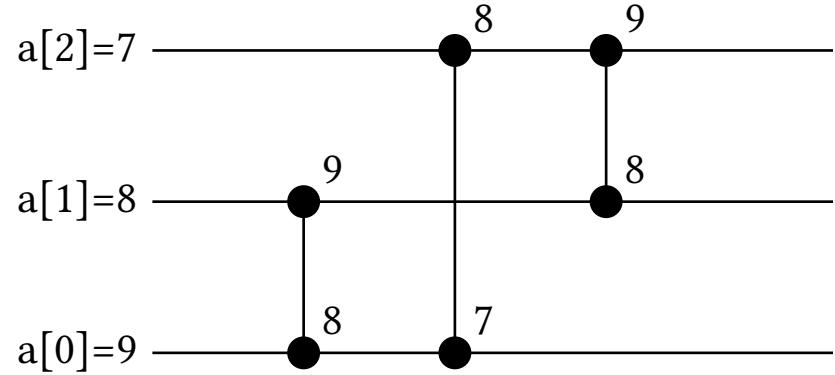
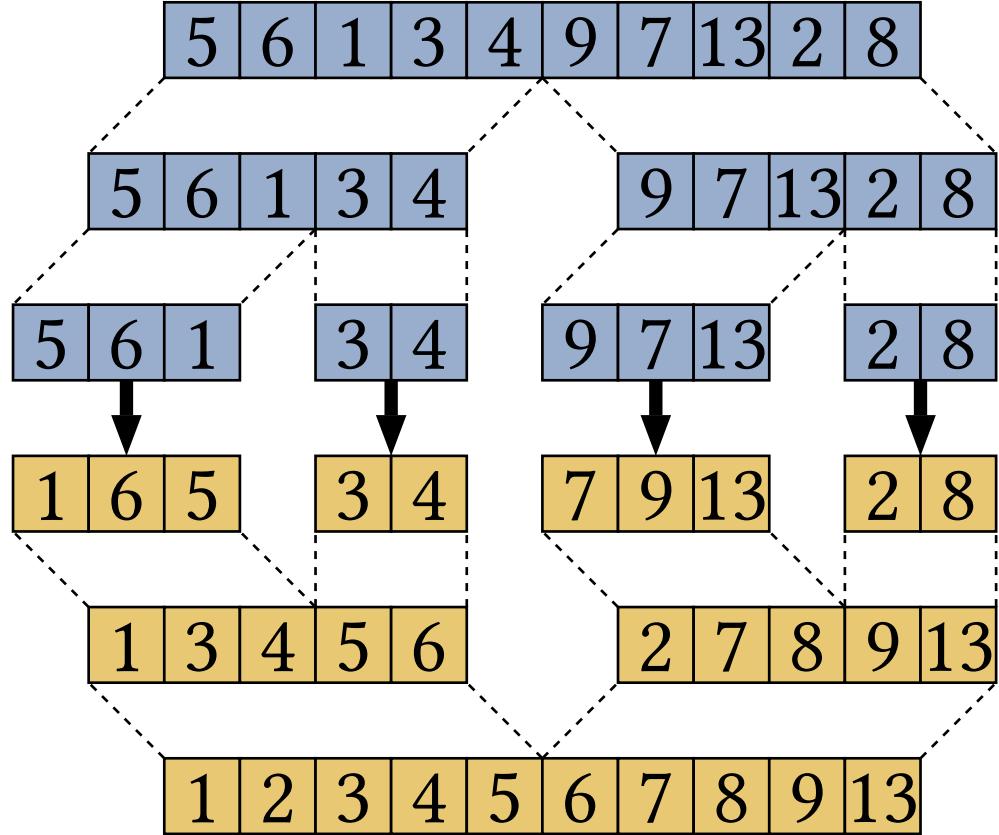
# Sorting Kernels



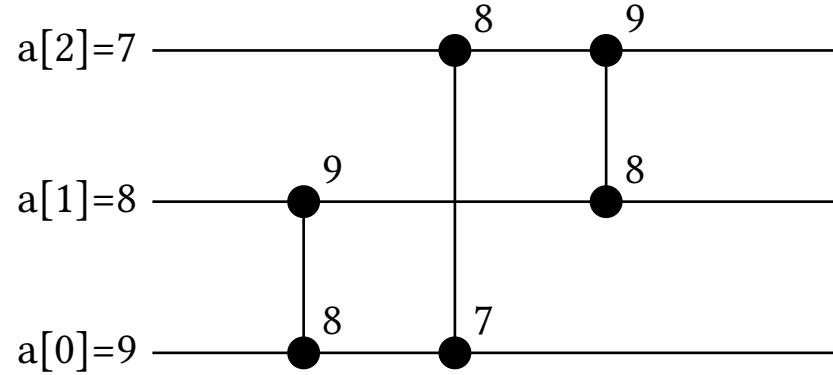
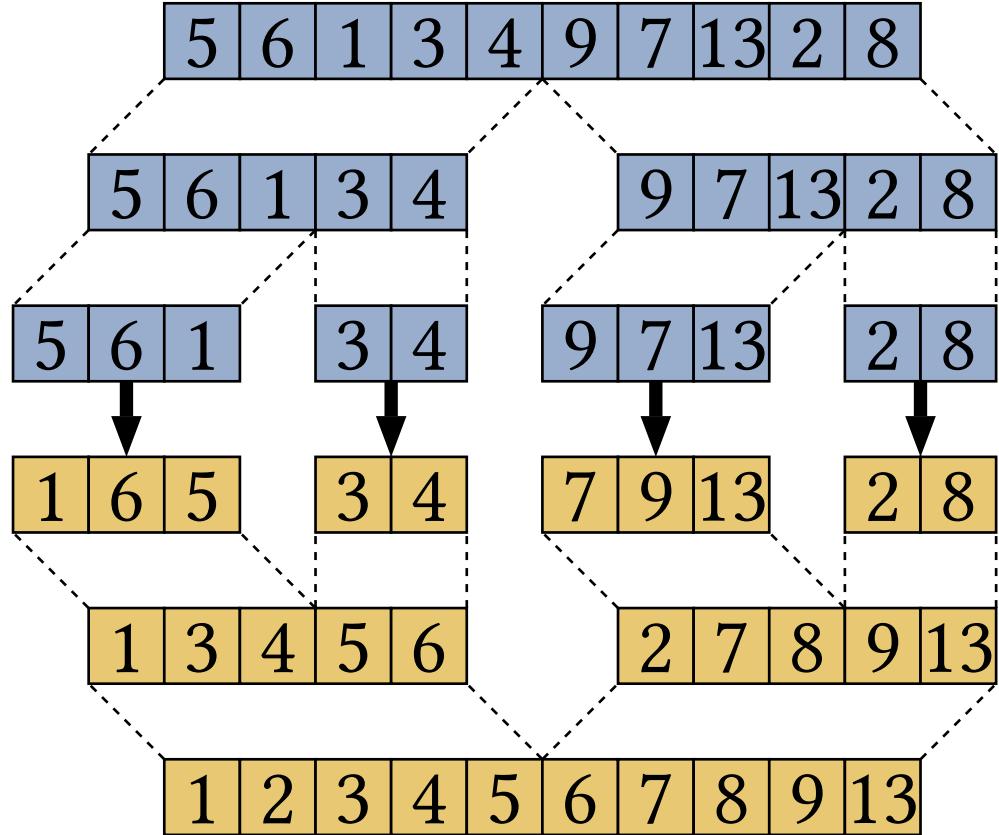
# Sorting Kernels



# Sorting Kernels



# Sorting Kernels



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

# State of the Art

- sorting network 
- 

---

<sup>1</sup> Mankowitz, Daniel J., et al. “Faster sorting algorithms discovered using deep reinforcement learning.” *Nature* 618.7953 (2023): 257-263.



# State of the Art

- handoptimized 
- 

---

<sup>2</sup> Mankowitz, Daniel J., et al. “Faster sorting algorithms discovered using deep reinforcement learning.” *Nature* 618.7953 (2023): 257-263.



# State of the Art

- handoptimized 
- 2024 AlphaDev<sup>3</sup>
  - ▶  $n = 3$ : 6min
  - ▶  $n = 4$ : 30min
  - ▶  $n = 5$ : 17.5h

---

<sup>3</sup> Mankowitz, Daniel J., et al. “Faster sorting algorithms discovered using deep reinforcement learning.” Nature 618.7953 (2023): 257-263.



# State of the Art

- handoptimized 
  - ▶  $n = 3$ : ~~6min~~ 97ms
  - ▶  $n = 4$ : ~~30min~~ 2.4s
  - ▶  $n = 5$ : ~~17.5h~~ 11min
-  faster synthesis

# State of the Art

- handoptimized 
- - ▶  $n = 3$ : **6min 97ms**
  - ▶  $n = 4$ : **30min 2.4s**
  - ▶  $n = 5$ : **17.5h 11min**
-  faster synthesis
-  faster sorting kernels

# State of the Art

- handoptimized 
- - ▶  $n = 3$ : ~~6min~~ 97ms
  - ▶  $n = 4$ : ~~30min~~ 2.4s
  - ▶  $n = 5$ : ~~17.5h~~ 11min
-  faster synthesis
-  faster sorting kernels
-  minimality proof

# Model

			register	swap	lt	gt
mov	s1	r2	2	1	-	-
cmp	r1	r2	2	1	1	-
cmovg	r2	r1	2	1	1	-
cmovg	r1	s1	2	2	1	-
			1	2	1	-

# Search Space

			register	swap	lt	gt
mov	s1	r2	2	1	-	-
cmp	r1	r2	2	1	1	-
cmoveg	r2	r1	2	1	1	-
cmoveg	r1	s1	2	2	1	-
			1	2	1	-

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



# Search Space

mov	s1	r2
cmp	r1	r2
cmove	r2	r1
cmove	r1	s1

	register	swap	lt	gt
2	1	-	-	-
2	1	1	-	-
2	1	1	-	>
2	2	1	-	>
1	2	1	-	>

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

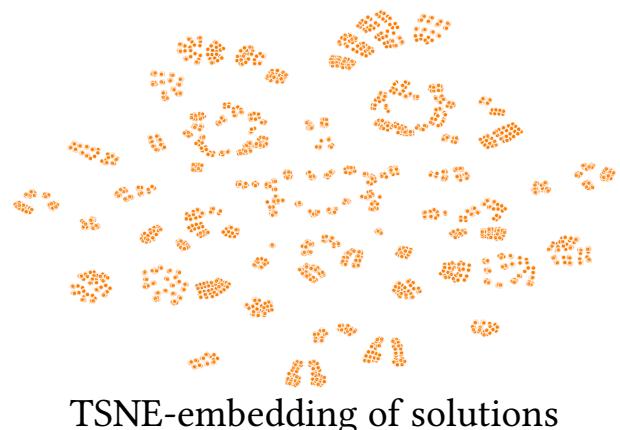
$$\min(a, \min(b, c)) = \min(\min(\max(c, b), a), \min(b, c))$$

5602 solutions for  $n = 3$

# Search Space

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

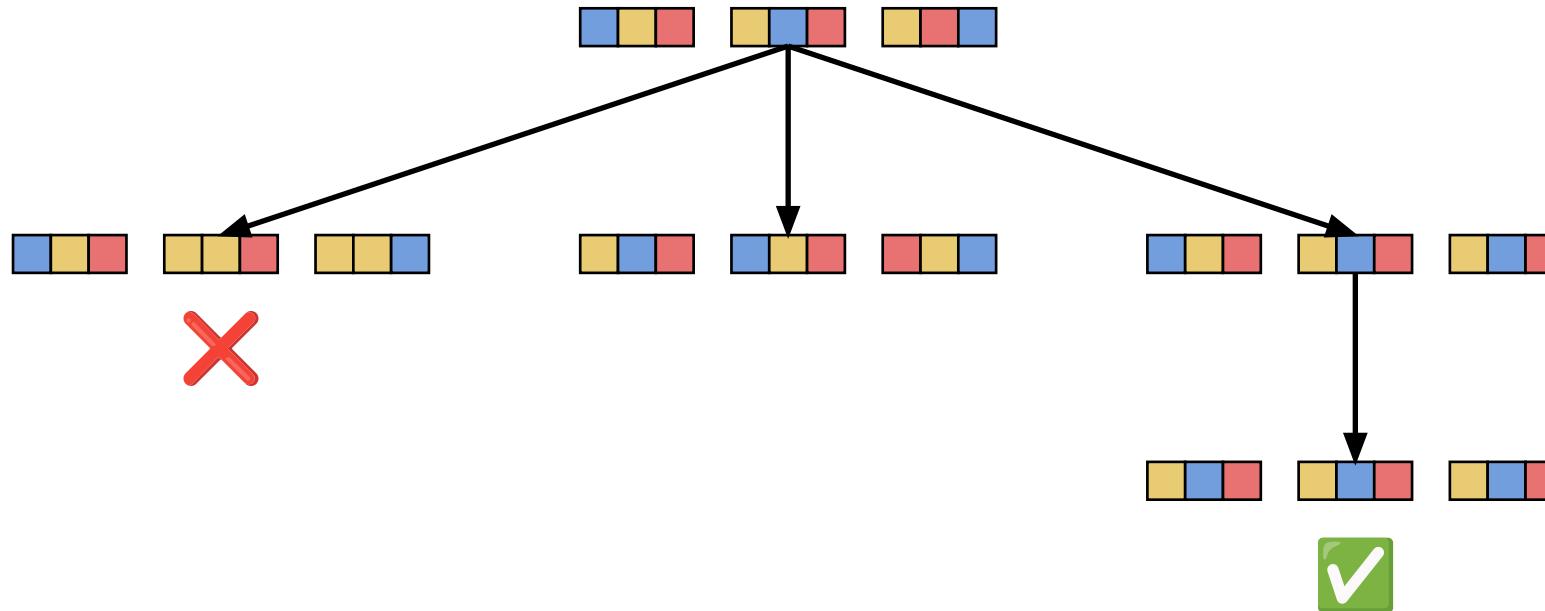
	register	swap	lt	gt
mov	2	1	-	-
cmp	2	1	1	-
cmovg	2	1	1	- >
cmovg	2	2	1	- >
	1	2	1	- >



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

5602 solutions for  $n = 3$

# 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 $\star$  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 → 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  $> k \times \text{best}$

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

$$\begin{aligned} \forall r : P(r) = o \rightarrow \\ \underbrace{\left( \forall 1 \leq i \leq |r| : o_i \leq o_{i+1} \right)}_{\text{ascending}} \wedge \\ \underbrace{\left( \forall x : |\{i : r_i = x\}| = |\{i : o_i = x\}| \right)}_{\text{same elements}} \end{aligned}$$



# Solver-Based Techniques

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

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



# Solver-Based Techniques

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



# Solver-Based Techniques

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

Heuristics:

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



# Solver-Based Synthesis

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



# Solver-Based Synthesis

SMT	Approach	CP	Approach
97min	CEGIS, arbitrary inputs	—	ILP, MIP
25min	CEGIS, 1..n	—	CP (MiniZinc other)
44min	all permutations	232s	chuffed, no heuristic
—	SyGuS (CVC5, Metalift)	70s	chuffed, h, = 1..n
		30s	chuffed, h, $\leq$ , 1..3
		0.9s	+init



# Solver-Based Synthesis

SMT	Approach	CP	Approach
97min	CEGIS, arbitrary inputs	—	ILP, MIP
25min	CEGIS, 1..n	—	CP (MiniZinc other)
44min	all permutations	232s	chuffed, no heuristic
—	SyGuS (CVC5, Metalift)	70s	chuffed, $h, = 1..n$
Planning		30s	chuffed, $h, \leq, 1..3$
		0.9s	+init
679s	Scorpion planner		
216s	Lama planner grounded		
3.54s	Lama Planner		

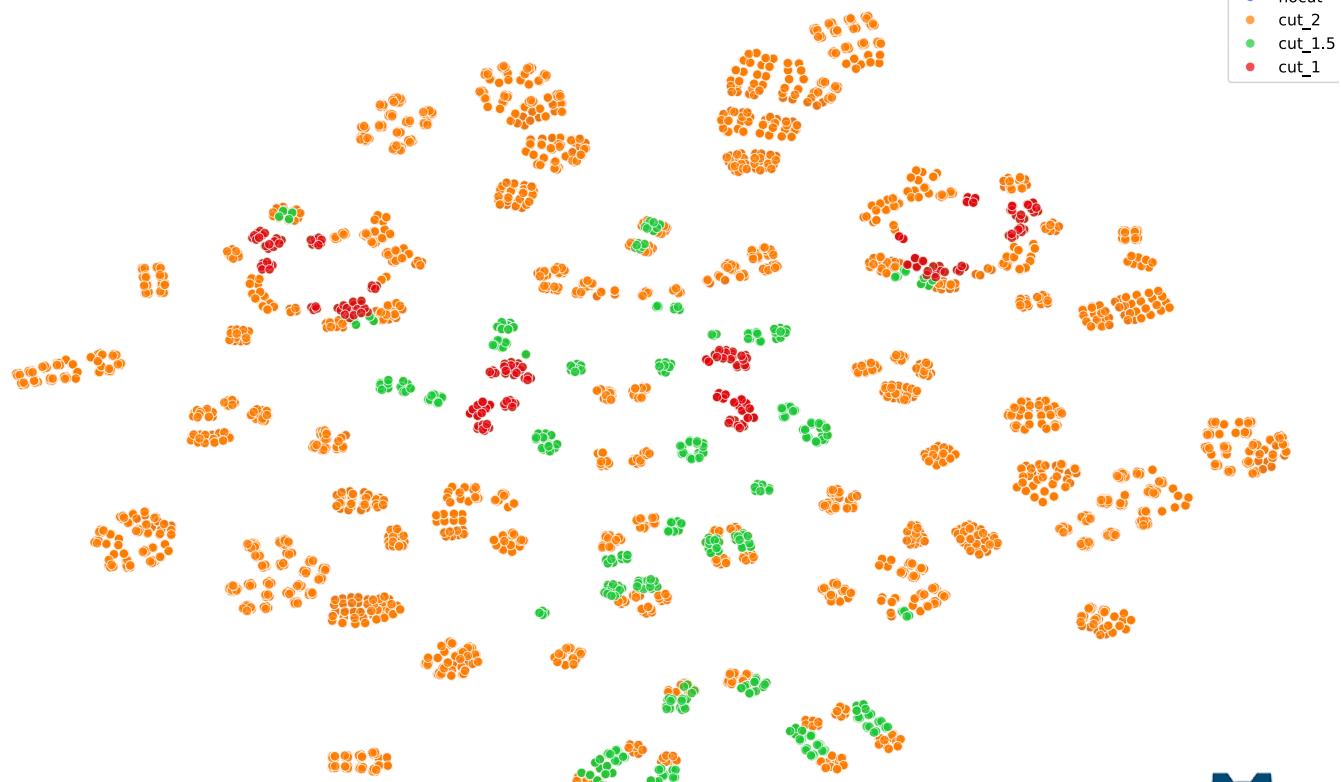


# Evaluation Enumeration $n = 3$

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*	1.7s
+viable, instr	0.7s
+cut $k = 1$	0.1s



# Evaluation Enumeration $n \geq 3$

	$l = 11$	$l = 20$	$l = 33$
Approach	$n = 3$	$n = 4$	$n = 5$
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 <sup>4</sup>	—	8.8ms	—
AlphaDev	6.7ms	10.4ms	16.2ms
Sorting Network (Cmp)	7.1ms	14.8ms	19.4ms

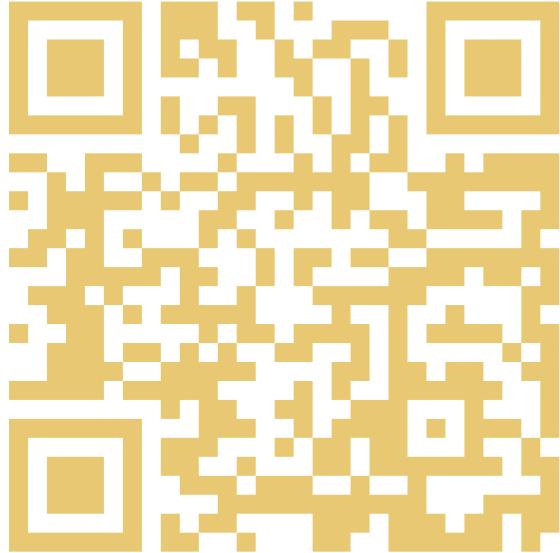
<sup>4</sup>Mimicry. 2023. Faster Sorting Beyond DeepMind's AlphaDev. <https://www.mimicry.ai/faster-sorting-beyond-deepminds-alphadev> Accessed: 2023-09-20

# Evaluation Kernels MinMax

Kernel	$n = 3$	$n = 4$	$n = 5$
Enumeration	5.8ms	9.4ms	14.8ms
Mimicry <sup>5</sup>	—	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

<sup>5</sup>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