minimizing sorting networks at the sub-comparator level

luís cruz-filipe
(joint work with peter schneider-kamp)

department of mathematics and computer science
university of southern denmark

lpar-25, balaclava
may 27th, 2024

learning from ai
with an example from sorting networks

luís cruz-filipe
(joint work with peter schneider-kamp)

department of mathematics and computer science
university of southern denmark

lpar-25, balaclava
may 27th, 2024

background

sorting networks

- data-oblivious algorithms, readily implementable as hardware circuits
- also used as base cases for software algorithms in libraries, each gate is implemented as four instructions

background

sorting networks

- data-oblivious algorithms, readily implementable as hardware circuits
- also used as base cases for software algorithms in libraries, each gate is implemented as four instructions

new development using deep learning

discovered better implementations of some sorting networks

background

sorting networks

- data-oblivious algorithms, readily implementable as hardware circuits
- also used as base cases for software algorithms in libraries, each gate is implemented as four instructions

new development using deep learning

discovered better implementations of some sorting networks

limitations

- *ad hoc* constructions
- no intuition
- not scalable

our contribution

our analysis

looking closer at the results, a pattern emerges

- all “new best” networks correspond to “old best” networks with some instructions removed
- always the same instruction

our contribution

our analysis

looking closer at the results, a pattern emerges

- all "new best" networks correspond to "old best" networks with some instructions removed
- always the same instruction

obvious (?) question

can this be generalized?

our contribution

our analysis

looking closer at the results, a pattern emerges

- all “new best” networks correspond to “old best” networks with some instructions removed
- always the same instruction

obvious (?) question

can this be generalized?

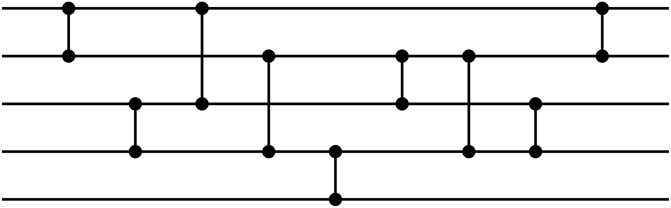
our results

- general construction
- can be reduced to a satisfiability problem
- generalizes to all relevant practical cases

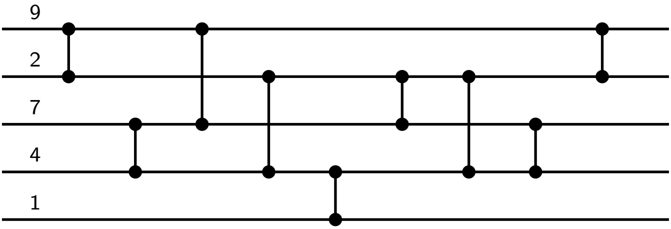
outline of this talk

1. basics of sorting networks
2. standard implementation
3. systematic optimization
4. reduction to sat
5. results
6. conclusions

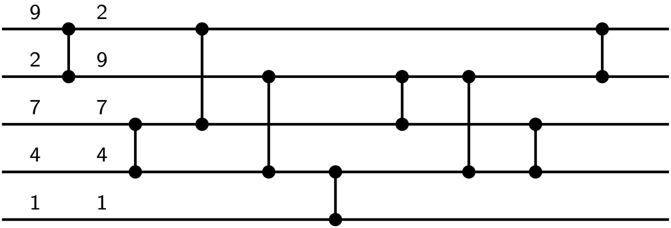
a sorting network



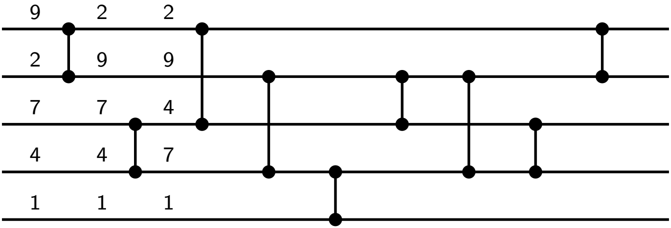
a sorting network



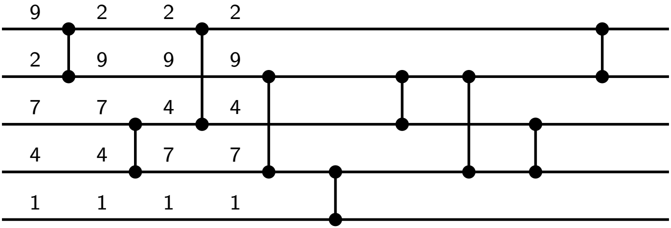
a sorting network



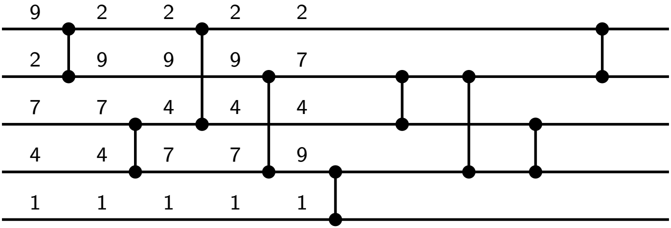
a sorting network



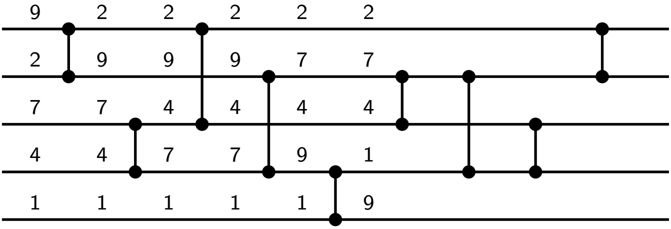
a sorting network



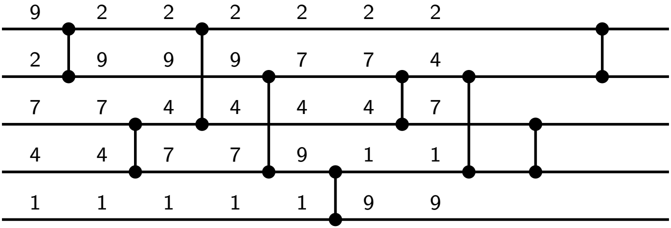
a sorting network



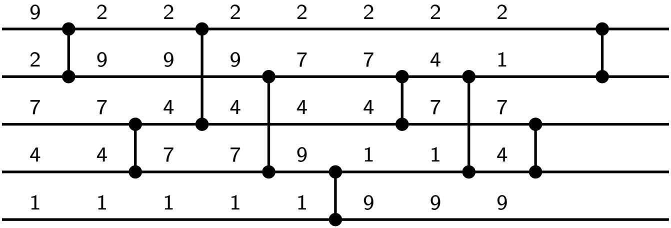
a sorting network



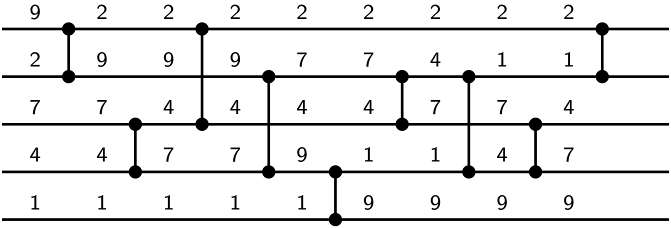
a sorting network



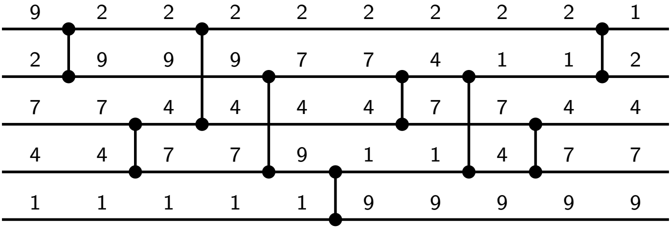
a sorting network



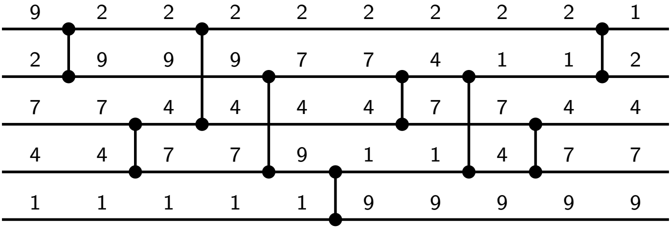
a sorting network



a sorting network



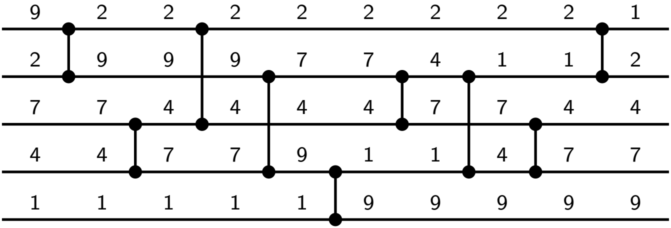
a sorting network



size

this net has 5 *channels* and 9 *comparators*

a sorting network



size

this net has 5 *channels* and 9 *comparators*

more info

see d.e. knuth, *the art of computer programming*, vol. 3

implementing comparators

comparators as conditional swaps

naive implementation of a comparator (i, j), assuming the starting values for channels i and j are in registers r_i and r_j:

1. copy r_i to a new register r_k
2. if $r_i > r_j$, then copy r_j to r_k
3. if $r_i > r_j$, then copy r_i to r_j

the final values are stored in r_k and r_j, while r_i is discarded

implementing comparators

implementation using conditional moves

1. MOV r_i r_k
2. CMP $r_i > r_j$
3. CMOVGE r_j r_k
4. CMOVGE r_i to r_j

more efficient: only one comparison

$\rightsquigarrow$ CMOVGE checks the result of the last CMP operation and performs the copy if it is GE

here comes ai

AlphaDev

- machine-learning system
- trained using deep reinforcement learning
- learned to discover implementations of sorting networks

here comes ai

AlphaDev

- machine-learning system
- trained using deep reinforcement learning
- learned to discover implementations of sorting networks

results

- can generate best known networks for up to 8 input
- in some cases, saves 1–3 instructions
- in *all* cases, the instruction removed is the first one

here comes ai

AlphaDev

- machine-learning system
- trained using deep reinforcement learning
- learned to discover implementations of sorting networks

results

- can generate best known networks for up to 8 input
- in some cases, saves 1–3 instructions
- in *all* cases, the instruction removed is the first one

human abstraction

when can the first instruction be safely removed?

redundant auxiliary registers?

when can the first instruction be safely removed?

1. MOV r_i r_k
2. CMP $r_i > r_j$
3. CMOVGE r_j r_k
4. CMOVGE r_i to r_j

redundant auxiliary registers?

when can the first instruction be safely removed?

1. MOV r_i r_k
2. CMP $r_i > r_j$
3. CMOVGE r_j r_k
4. CMOVGE r_i to r_j

- when the auxiliary register already contains the correct value

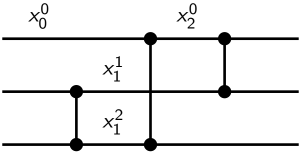
redundant auxiliary registers?

when can the first instruction be safely removed?

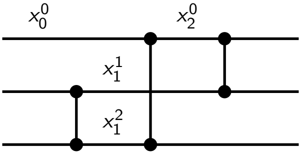
1. MOV r_i r_k
2. CMP $r_i > r_j$
3. CMOVGE r_j r_k
4. CMOVGE r_i to r_j

- when the auxiliary register already contains the correct value
- and no swaps are performed

an example



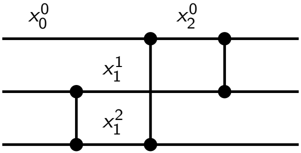
an example



x_2^0 has the right value

- $x_2^0 < x_1^1$ (no swap)
- $x_2^0 = \min(x_0^0, x_1^2)$ (second comparator)
- $x_1^1 \leq x_1^2$ (first comparator)

an example



x^0_2 has the right value

- $x^0_2 < x^1_1$ (no swap)
- $x^0_2 = \min(x^0_0, x^2_1)$ (second comparator)
- $x^1_1 \leq x^2_1$ (first comparator)

yes!

these conditions imply that $x^0_0 = x^0_2$!

the general case

- the reason for removing an assignment may be more complicated than the pattern shown earlier
- in general, we may need several comparators and information about min and max values
- all potentially relevant constraints are collected by a forward pass through the network
- for each comparator, an smt-solver determines whether the set of constraints given implies that the assignment instruction is redundant

optimizing the optimization

the potential problem

smt solvers cannot deal with larger networks

optimizing the optimization

the potential problem

smt solvers cannot deal with larger networks

(maybe this is a theoretical issue, but we're theoreticians)

optimizing the optimization

the potential problem

smt solvers cannot deal with larger networks

can we use sat solvers?

in all case where AlphaDev managed to optimize the network implementation, there is an additional property

- the register being reused was previously at the min-end of a comparator

optimizing the optimization

the potential problem

smt solvers cannot deal with larger networks

can we use sat solvers?

in all case where AlphaDev managed to optimize the network implementation, there is an additional property

- the register being reused was previously at the min-end of a comparator

yes, we can!

with this additional restriction, the satisfiability problems over $\mathbb{Z}$ and $\{0, 1\}$ are equivalent

what can we do?

- we can replicate all AlphaDev's optimizations but one
- we improve on AlphaDev's result for 7 inputs
- we can apply our construction to networks with up to 128 inputs (those relevant in practical applications)
- sat solvers solve the problems quicker than smt, and increasingly faster

the extra assumption

- we tested whether our extra assumption prevents some optimizations
- concretely, if the sat solver returned UNSAT and the set of constraints did not include the relevant one, we ran an smt solver on top
- every time the smt solver was called, it disagreed with the sat solver (so no additional assignments were removed)

the extra assumption

- we tested whether our extra assumption prevents some optimizations
- concretely, if the sat solver returned UNSAT and the set of constraints did not include the relevant one, we ran an smt solver on top
- every time the smt solver was called, it disagreed with the sat solver (so no additional assignments were removed)

conjecture

our additional assumption is a necessary condition for being able to remove an assignment

the extra assumption

- we tested whether our extra assumption prevents some optimizations
- concretely, if the sat solver returned UNSAT and the set of constraints did not include the relevant one, we ran an smt solver on top
- every time the smt solver was called, it disagreed with the sat solver (so no additional assignments were removed)

conjecture

our additional assumption is a necessary condition for being able to remove an assignment

- no idea how to prove it
  (like so much else related to sorting networks. . .)

the moral of the story

man vs machine

data-driven ai cannot abstract

- it can find cool new results. . .

the moral of the story

man vs machine

data-driven ai cannot abstract

- it can find cool new results...
- ...but humans are the ones who can understand them and generalize them

thank you!