advances in sorting networks

luís cruz-filipe

(joint work with michael codish[2], michael frank[2] and peter schneider-kamp[1])

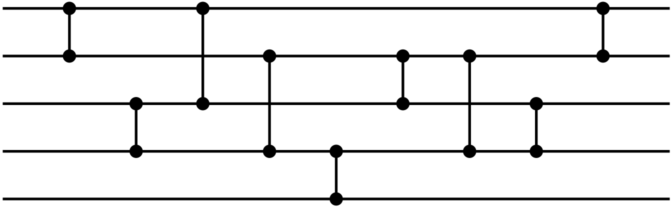
[1]department of mathematics and computer science
university of southern denmark

[2]department of computer science
ben-gurion university of the negev, israel

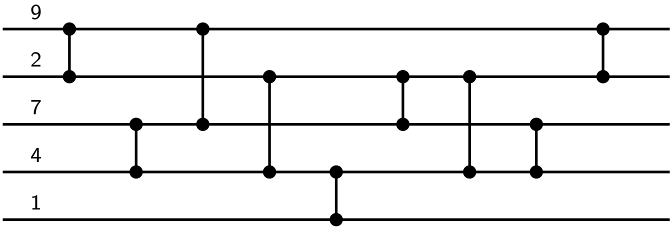
radboud university
april 28th, 2015

outline

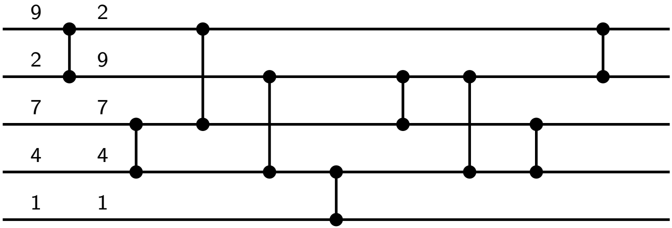
a sorting network



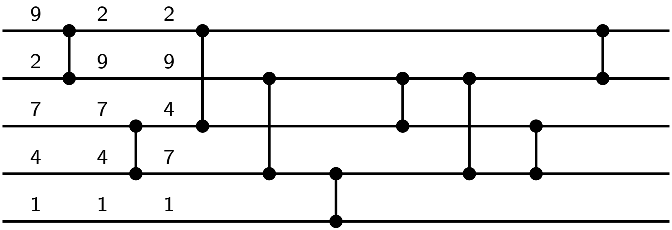
a sorting network



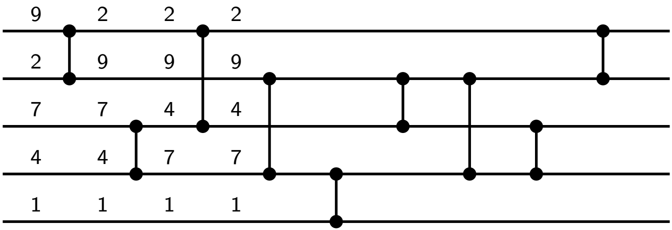
a sorting network



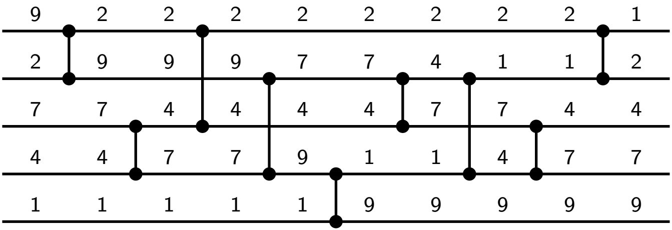
a sorting network



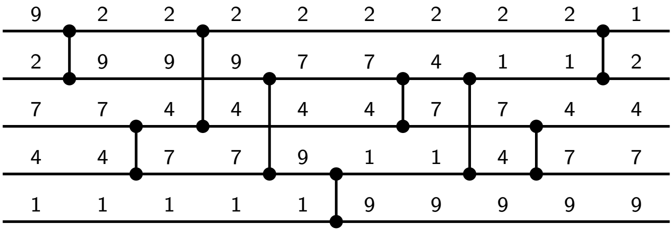
a sorting network



a sorting network

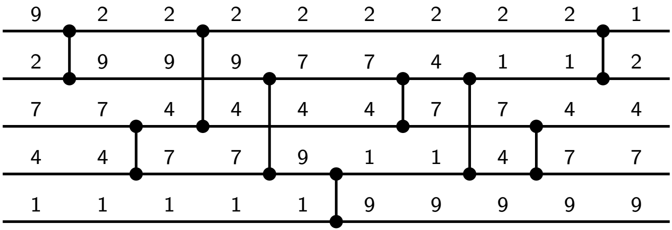


a sorting network



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

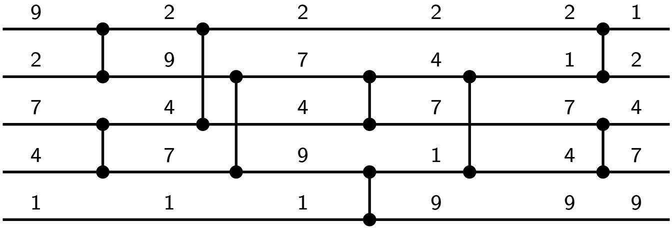
a sorting network



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

some of the comparisons may be performed in parallel

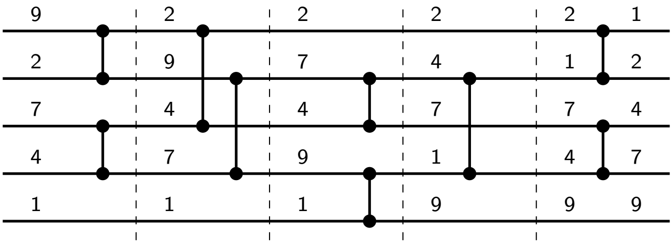
a sorting network



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

some of the comparisons may be performed in parallel

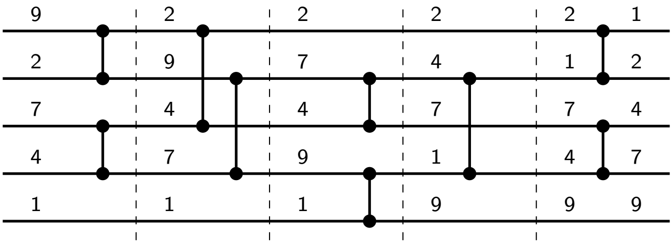
a sorting network



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

some of the comparisons may be performed in parallel

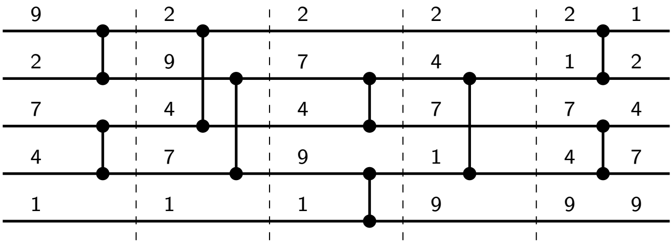
a sorting network



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

depth this net has 5 *layers*

a sorting network



size	this net has 5 *channels* and 9 *comparators*
depth	this net has 5 *layers*
more info	see d.e. knuth, *the art of computer programming*, vol. 3

outline

the optimization problems

the optimal size problem

what is the minimal number of *comparators* in a sorting network on n channels (s_n)?

the optimal depth problem

what is the minimal number of *layers* in a sorting network on n channels (t_n)?

the optimization problems

the optimal size problem

what is the minimal number of *comparators* in a sorting network on n channels (s_n)?

the optimal depth problem

what is the minimal number of *layers* in a sorting network on n channels (t_n)?

knuth 1973

n	1	2	3	4	5	6	7	8	9	10
s_n	0	1	3	5	9	12	16	19	25 23	29 27
t_n	0	1	3	3	5	5	6	6	7 6	7 6

n	11	12	13	14	15	16	17
s_n	35 31	39 35	45 39	51 43	56 47	60 51	73 56
t_n	8 7	8 7	9 7	9 7	9 7	9 7	11 7

the optimal depth problem

optimal depth

t_n: minimal number of *steps* to sort n inputs

knuth 1973

n	8	9	10	11	12	13	14	15	16	17
t_n	6	7	7	8	8	9	9	9	9	11
		6	6	7	7	7	7	7	7	7

- upper bounds obtained by concrete examples (1960s)
- lower bounds obtained by mathematical arguments
- exhaustive analysis of space state

- huge number of networks

the optimal depth problem

optimal depth

t_n: minimal number of *steps* to sort n inputs

parberry 1991

n	8	9	10	11	12	13	14	15	16	17
t_n	6	**7**	**7**	8 7	8 7	9 7	9 7	9 7	9 7	11 7

- exploration of symmetries $\rightsquigarrow$ fixed first layer
- exhaustive search (200 hours of computation)

the optimal depth problem

optimal depth t_n: minimal number of *steps* to sort n inputs

parberry 1991

n	8	9	10	11	12	13	14	15	16	17
t_n	6	**7**	**7**	8	8	9	9	9	9	11
				7	7	7	7	7	7	7

- exploration of symmetries $\rightsquigarrow$ fixed first layer
- exhaustive search (200 hours of computation)

update (2015) now takes only 12 seconds

- still does not finish for $n = 11$

the optimal depth problem

optimal depth

bundala & závodný 2013

t_n: minimal number of *steps* to sort n inputs

n	8	9	10	11	12	13	14	15	16	17
t_n	6	7	7	**8**	**8**	**9**	**9**	**9**	**9**	11 **9**

- reduced set of two-layer prefixes
- intensive sat-solving

the optimal depth problem

optimal depth

bundala & závodný 2013

t_n: minimal number of *steps* to sort n inputs

n	8	9	10	11	12	13	14	15	16	17
t_n	6	7	7	**8**	**8**	**9**	**9**	**9**	**9**	11 **9**

- reduced set of two-layer prefixes
- intensive sat-solving

- prefixes very expensive to compute
- cannot handle $n = 14$ directly

the optimal depth problem

optimal depth

ehlers & müller 2014

t_n: minimal number of *steps* to sort n inputs

n	8	9	10	11	12	13	14	15	16	17
t_n	6	7	7	8	8	9	9	9	9	**10** 9

- heuristics to reduce search space
- intensive sat-solving

the optimal depth problem

optimal depth

ehlers & müller 2015

t_n: minimal number of *steps* to sort n inputs

n	8	9	10	11	12	13	14	15	16	17
t_n	6	7	7	8	8	9	9	9	9	**10**

- reduced set of two-layer prefixes*
- constraints on last layers*
- iterative sat-solving
- optimal arrangement of prefixes

(*results from yours truly)

the optimal size problem

optimal size

s_n: minimal number of *comparisons* to sort n inputs

knuth 1973

n	1	2	3	4	5	6	7	8	9	10
s_n	0	1	3	5	9	12	16	19	25 23	29 27

n	11	12	13	14	15	16	17
s_n	35 31	39 35	45 39	51 43	56 47	60 51	73 56

- values for $n \leq 4$ from information theory
- values for $n = 5$ and $n = 7$ by exhaustive case analysis

knuth $s_{n+1} \geq s_n + 3$ $\rightsquigarrow$ values for $n = 6, 8$

van voorhis $s_{n+1} \geq s_n + \lg(n)$ $\rightsquigarrow$ other lower bounds

the optimal size problem

optimal size

s_n: minimal number of *comparisons* to sort n inputs

n	1	2	3	4	5	6	7	8	9	10
s_n	0	1	3	5	9	12	16	19	**25**	**29**

yours truly 2014

n	11	12	13	14	15	16	17
s_n	35 **33**	39 **37**	45 **41**	51 **45**	56 **49**	60 **53**	73 **58**

- generate-and-prune algorithm
- intensive parallel computing
- $\sim$ 16 years of cpu time to compute s_9

the optimal size problem

optimal size

yours truly 2014

s_n: minimal number of *comparisons* to sort n inputs

n	1	2	3	4	5	6	7	8	9	10
s_n	0	1	3	5	9	12	16	19	**25**	**29**

n	11	12	13	14	15	16	17
s_n	35 **33**	39 **37**	45 **41**	51 **45**	56 **49**	60 **53**	73 **58**

- generate-and-prune algorithm
- intensive parallel computing
- $\sim$ 16 years of cpu time to compute s_9

- first formally verified proof of values for $n \leq 9$
- more info at types 2015

outline

sorting networks in practice

sorting networks in practice

origins

military & airspace research (1950s)

sorting networks in practice

origins military & airspace research (1950s)

different compared to usual sorting

- *oblivious* sorting algorithms
- suitable for *hardware* implementations

sorting networks in practice

origins military & airspace research (1950s)

different compared to usual sorting

- *oblivious* sorting algorithms
- suitable for *hardware* implementations

interpretation the optimization problems

- *optimal size* = lowest production cost
- *optimal size* = lowest energy consumption
- *optimal depth* = lowest execution time

sorting networks in software

case study the quicksort implementation in the standard c library

- usual recursive procedure
- reverts to insertion sort for $n \leq 4$

sorting networks in software

case study the quicksort implementation in the standard c library

- usual recursive procedure
- reverts to insertion sort for $n \leq 4$

our experiments obtain improvements by

- full unrolling (no cycle)
- (oblivious) conditional move (no branching)
- compressing (maximum parallelization)

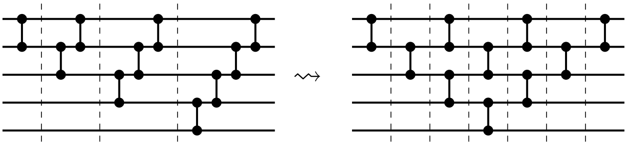
sorting networks in software

case study — the quicksort implementation in the standard c library

- usual recursive procedure
- reverts to insertion sort for $n \leq 4$

our experiments — obtain improvements by

- full unrolling (no cycle)
- (oblivious) conditional move (no branching)
- compressing (maximum parallelization)



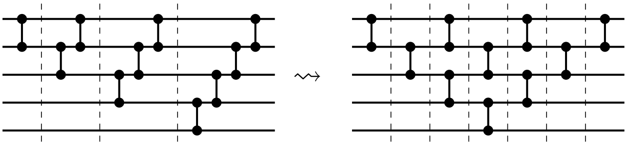
sorting networks in software

case study the quicksort implementation in the standard c library

- usual recursive procedure
- reverts to insertion sort for $n \leq 4$

our experiments obtain improvements by

- full unrolling (no cycle)
- (oblivious) conditional move (no branching)
- compressing (maximum parallelization)



$\rightsquigarrow$ so why not use a better sorting network?

results & directions

so far...

- switching to sorting networks on $n \leq 32$
- up to $1.5\times$ speedup on large instances
- using systematic constructions of non-optimal networks

results & directions

so far...

- switching to sorting networks on $n \leq 32$
- up to $1.5\times$ speedup on large instances
- using systematic constructions of non-optimal networks

$n = 9$ used for benchmarking and testing

- size matters
- depth matters
- too much parallelism does not help

results & directions

so far…

- switching to sorting networks on $n \leq 32$
- up to $1.5\times$ speedup on large instances
- using systematic constructions of non-optimal networks

$n = 9$ used for benchmarking and testing

- size matters
- depth matters
- too much parallelism does not help

mysteries not always working as expected

- we can predict *bad* performance
- we cannot predict *good* performance
- need better understanding of cpu parallelism

outline

comparator networks

comparator network	a *comparator network* C on n channels is a sequence of *comparators* (i, j) with $1 \leq i < j \leq n$
output	$C(\vec{x})$ denotes the *output* of C on $\vec{x} = x_1 \dots x_n$
binary outputs	the set of binary outputs of C is $\text{outputs}(C) = \{C(\vec{x}) \mid x \in \{0, 1\}^n\}$
sorting network	a comparator network C is a *sorting network* if $C(\vec{x})$ is sorted for every input $\vec{x}$

two essential results

0/1 lemma (knuth 1973)

C is a sorting network on n channels iff C sorts all inputs in $\{0,1\}^n$

two essential results

0/1 lemma (knuth 1973)

C is a sorting network on n channels iff C sorts all inputs in $\{0,1\}^n$

"C is a sorting network on n channels" is co-NP (complete)

two essential results

0/1 lemma (knuth 1973)

C is a sorting network on n channels iff C sorts all inputs in $\{0,1\}^n$

output lemma (parberry 1991)

if outputs(C) $\subseteq$ outputs(C') and $C'; N$ is a sorting network, then $C; N$ is a sorting network

two essential results

0/1 lemma (knuth 1973)

C is a sorting network on n channels iff C sorts all inputs in $\{0,1\}^n$

output lemma (parberry 1991)

if outputs$(C) \subseteq$ outputs(C') and $C'; N$ is a sorting network, then $C; N$ is a sorting network

proof

$$\begin{tikzcd}
\{0,1\}^n \arrow[r, "C"] & X \arrow[d, phantom, sloped, "\subseteq"] & \\
\{0,1\}^n \arrow[r, "C'"] & X' \arrow[r, "N"] & S
\end{tikzcd}$$

two essential results

0/1 lemma (knuth 1973)

C is a sorting network on n channels iff C sorts all inputs in $\{0,1\}^n$

output lemma (parberry 1991)

if outputs(C) $\subseteq$ outputs(C') and $C'; N$ is a sorting network, then $C; N$ is a sorting network

proof

$$\begin{array}{ccccc} \{0,1\}^n & \xrightarrow{C} & X & \xrightarrow{N} & S \\ & & \cap\!\! \mid & & \\ \{0,1\}^n & \xrightarrow{C'} & X' & \xrightarrow{N} & S \end{array}$$

two essential results

0/1 lemma (knuth 1973)
C is a sorting network on n channels iff C sorts all inputs in $\{0,1\}^n$

output lemma (parberry 1991)
if $\text{outputs}(C) \subseteq \text{outputs}(C')$ and $C'; N$ is a sorting network, then $C; N$ is a sorting network

corollary
there is a minimal-depth sorting network on n channels whose first layer contains $\left\lfloor \frac{n}{2} \right\rfloor$ comparators

two essential results

0/1 lemma (knuth 1973): C is a sorting network on n channels iff C sorts all inputs in $\{0,1\}^n$

output lemma (parberry 1991): if $\mathrm{outputs}(C) \subseteq \mathrm{outputs}(C')$ and $C'; N$ is a sorting network, then $C; N$ is a sorting network

corollary: there is a minimal-depth sorting network on n channels whose first layer contains $\left\lfloor \frac{n}{2} \right\rfloor$ comparators

permuted output lemma: if there is a permutation π such that $\pi(\mathrm{outputs}(C)) \subseteq \mathrm{outputs}(C')$ and C' extends to a sorting network, then C extends to a sorting network

two essential results

0/1 lemma (knuth 1973)

C is a sorting network on n channels iff C sorts all inputs in $\{0,1\}^n$

output lemma (parberry 1991)

if outputs(C) $\subseteq$ outputs(C') and $C'; N$ is a sorting network, then $C; N$ is a sorting network

corollary

there is a minimal-depth sorting network on n channels whose first layer contains $\left\lfloor \frac{n}{2} \right\rfloor$ comparators

permuted output lemma

if there is a permutation π such that $\pi(\text{outputs}(C)) \subseteq \text{outputs}(C')$ and C' extends to a sorting network, then C extends to a sorting network

proof

$$\begin{tikzcd}
\{0,1\}^n \arrow[r, "C"] & X \arrow[d, "\pi"] & \\
\{0,1\}^n \arrow[r, "C'"] & X' \arrow[r, "N"] & S
\end{tikzcd}$$

two essential results

0/1 lemma (knuth 1973)
C is a sorting network on n channels iff C sorts all inputs in $\{0,1\}^n$

output lemma (parberry 1991)
if $\text{outputs}(C) \subseteq \text{outputs}(C')$ and $C'; N$ is a sorting network, then $C; N$ is a sorting network

corollary
there is a minimal-depth sorting network on n channels whose first layer contains $\left\lfloor \frac{n}{2} \right\rfloor$ comparators

permuted output lemma
if there is a permutation π such that $\pi(\text{outputs}(C)) \subseteq \text{outputs}(C')$ and C' extends to a sorting network, then C extends to a sorting network

proof

$$\begin{tikzcd}
\{0,1\}^n \arrow[r, "C"] & X \arrow[r, "\pi^{-1}(N)"] \arrow[d, "\pi"] & \pi^{-1}(S) \\
\{0,1\}^n \arrow[r, "C'"] & X' \arrow[r, "N"] & S
\end{tikzcd}$$

two essential results

0/1 lemma (knuth 1973): C is a sorting network on n channels iff C sorts all inputs in $\{0,1\}^n$

output lemma (parberry 1991): if outputs$(C) \subseteq$ outputs(C') and $C'; N$ is a sorting network, then $C; N$ is a sorting network

corollary: there is a minimal-depth sorting network on n channels whose first layer contains $\left\lfloor \frac{n}{2} \right\rfloor$ comparators

permuted output lemma: if there is a permutation π such that $\pi(\text{outputs}(C)) \subseteq \text{outputs}(C')$ and C' extends to a sorting network, then C extends to a sorting network

proof:

$$\begin{tikzcd}
\{0,1\}^n \arrow[r, "C"] & X \arrow[r, "t(\pi^{-1}(N))"] \arrow[d, "\pi"] & S \\
\{0,1\}^n \arrow[r, "C'"] & X' \arrow[r, "N"] & S
\end{tikzcd}$$

two essential results

0/1 lemma (knuth 1973)	C is a sorting network on n channels iff C sorts all inputs in $\{0,1\}^n$
output lemma (parberry 1991)	if outputs$(C) \subseteq$ outputs(C') and $C'; N$ is a sorting network, then $C; N$ is a sorting network
corollary	there is a minimal-depth sorting network on n channels whose first layer contains $\left\lfloor \frac{n}{2} \right\rfloor$ comparators
permuted output lemma	if there is a permutation π such that $\pi(\text{outputs}(C)) \subseteq$ outputs(C') and C' extends to a sorting network, then C extends to a sorting network
corollary	there is a minimal-depth sorting network on n channels whose first layer contains $(1,2)$, $(3,4)$, $(5,6)$, &c

generate-and-prune

subsumption

$C \preceq_\pi C'$ if $\pi(\mathsf{outputs}(C)) \subseteq \mathsf{outputs}(C')$
$C \preceq C'$ if $C \preceq_\pi$ for some permutation π

generate-and-prune

subsumption

$C \preceq_\pi C'$ if $\pi(\text{outputs}(C)) \subseteq \text{outputs}(C')$
$C \preceq C'$ if $C \preceq_\pi$ for some permutation π

$\rightsquigarrow$ subsumption is reflexive and transitive

generate-and-prune

subsumption $C \preceq_\pi C'$ if $\pi(\text{outputs}(C)) \subseteq \text{outputs}(C')$
$C \preceq C'$ if $C \preceq_\pi$ for some permutation π

init set $R_0^n = \{\emptyset\}$ and $k = 0$

repeat until $k > 1$ and $|R_k^n| = 1$

generate N_{k+1}^n extend each net in R_k^n by one comparator in all possible ways

prune to R_{k+1}^n keep only one element from each minimal equivalence class w.r.t. $\preceq^T$

step increase k

generate-and-prune

subsumption $C \preceq_\pi C'$ if $\pi(\text{outputs}(C)) \subseteq \text{outputs}(C')$
$C \preceq C'$ if $C \preceq_\pi$ for some permutation π

init set $R_0^n = \{\emptyset\}$ and $k = 0$

repeat until $k > 1$ and $|R_k^n| = 1$

generate N_{k+1}^n extend each net in R_k^n by one comparator in all possible ways

prune to R_{k+1}^n keep only one element from each minimal equivalence class w.r.t. $\preceq^T$

step increase k

termination condition if C is a sorting network on n channels of size k, then $|R_k^n| = 1$

optimizations

- only generate networks when the extra comparator does something
- prove and implement criteria for when subsumption will fail
- restrict the search space of possible permutations
- optimize data structures
- parallelize to 288 nodes

some numerology

n	s_n	max $\lvert N_k^n \rvert$	max $\lvert R_k^n \rvert$	execution time
3	3	2	2	$\sim$0
4	5	12	4	$\sim$0
5	9	65	11	$\sim$0
6	12	380	53	2 sec
7	16	7,438	678	2 min
8	19	253,243	16,095	6 hours
9	25	18,420,674	914,444	16 years

some numerology

n	s_n	max $\|N_k^n\|$	max $\|R_k^n\|$	execution time
3	3	2	2	$\sim$0
4	5	12	4	$\sim$0
5	9	65	11	$\sim$0
6	12	380	53	2 sec
7	16	7,438	678	2 min
8	19	253,243	16,095	6 hours
9	25	18,420,674	914,444	16 years

parallel runtime for $n = 9$: 3 weeks

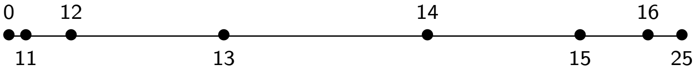
some numerology

n	s_n	max $\lvert N_k^n \rvert$	max $\lvert R_k^n \rvert$	execution time
3	3	2	2	$\sim$0
4	5	12	4	$\sim$0
5	9	65	11	$\sim$0
6	12	380	53	2 sec
7	16	7,438	678	2 min
8	19	253,243	16,095	6 hours
9	25	18,420,674	914,444	16 years

parallel runtime for $n = 9$: 3 weeks

the hard part

going "over the peak" consumes most execution time



why should we trust this result?

de bruijn criterium

non-trivial “trusted code” kept to a minimum: subsumption check

- manual verification of kernel (12 lines of prolog code)
- very simple structure of remaining code
- optimizations are safe!

why should we trust this result?

de bruijn criterium

non-trivial "trusted code" kept to a minimum: subsumption check

- manual verification of kernel (12 lines of prolog code)
- very simple structure of remaining code
- optimizations are safe!

grounds for skepticism

still, humans make mistakes...

why should we trust this result?

de bruijn criterium

non-trivial "trusted code" kept to a minimum: subsumption check

- manual verification of kernel (12 lines of prolog code)
- very simple structure of remaining code
- optimizations are safe!

grounds for skepticism

still, humans make mistakes. . .

independent verifications

subsequent validations of the code using logged witnesses for successful pruning steps

- sat-based verification (uses R^9_{14})
- independent (skeptic) java verifier
- coq checker using an offline oracle

outline

subsumption i/ii

equivalence if C and C' can be obtained from each other by renaming of channels, then $C \preceq C'$ and $C' \preceq C$

strategy bundala & závodný, 2013

- generate all two-layer prefixes
- represent them as graphs
- use graph isomorphism tool to select representatives

but...

- incomplete method (due to encoding)
- does not scale beyond $n = 13$

subsumption i/ii

equivalence if C and C' can be obtained from each other by renaming of channels, then $C \preceq C'$ and $C' \preceq C$

strategy bundala & závodný, 2013

- generate all two-layer prefixes
- represent them as graphs
- use graph isomorphism tool to select representatives

but...

- incomplete method (due to encoding)
- does not scale beyond $n = 13$

yours truly symbolic representation based on paths

- bipartite graphs (easy)
- generate representatives: regular grammar + simple test

subsumption ii/ii

saturation

C is not saturated if $C;(i,j) \preceq C$ for some i,j
(new network still has depth 2)
$\rightsquigarrow$ not true in general, see knuth

subsumption ii/ii

saturation C is not saturated if $C;(i,j) \preceq C$ for some i, j (new network still has depth 2)

strategy bundala & závodný, 2013

- syntactic criteria necessary for saturation
- generate only saturated two-layer prefixes

subsumption ii/ii

saturation C is not saturated if $C;(i,j) \preceq C$ for some i,j (new network still has depth 2)

strategy bundala & závodný, 2013

- syntactic criteria necessary for saturation
- generate only saturated two-layer prefixes

yours truly full syntactic characterization of saturation

- can encode in upgraded grammar
- eliminates need for subsumption

subsumption ii/ii

saturation C is not saturated if $C;(i,j) \preceq C$ for some i,j (new network still has depth 2)

strategy bundala & závodný, 2013

- syntactic criteria necessary for saturation
- generate only saturated two-layer prefixes

yours truly full syntactic characterization of saturation

- can encode in upgraded grammar
- eliminates need for subsumption

reflection observation by knuth to b&z

- also encodable in grammar
- not captured by subsumption

last-layer constraints

new insight necessary conditions on shape of comparators

- insight on semantics of sorting networks
- significant reduction of search space
- on 17 channels:
  - 2583 last layers, was: 211 million

last-layer constraints

new insight

- necessary conditions on shape of comparators
- insight on semantics of sorting networks
- significant reduction of search space

dual notions

- add redundancy where possible
- co-saturation (syntactic criterium)
- on 17 channels:
  - 89 last layers, was: 211 million
  - $\sim$ 40000 possibilities for two last layers, was: 4×10^{16}

last-layer constraints

new insight

necessary conditions on shape of comparators

- insight on semantics of sorting networks
- significant reduction of search space

dual notions

- add redundancy where possible
- co-saturation (syntactic criterium)

trivia

- fibonacci and padovan numbers
- can re-use regular grammar (but pointless)
- very sat-encodable
- partly applies to optimal size

outline

results

optimal depth

- efficient generation of two-layer prefixes
- last-layer constraints

optimal size

- exact values of s_9 and s_{10}
- formal verification

applications

- experiments with quicksort
- analysis of relevant measure to optimize

on the next episodes

optimal depth

- formalization
- proof of saturation conjecture

optimal size

- exact values of s_{11} (and $s_{12}\ldots$)
- generalization of van voorhis' lower bound

applications

- better implementation of quicksort
- optimal sorting networks for base cases

thank you!