The Quest for Optimal Sorting Networks

L. Cruz-Filipe[1] M. Codish[2] P. Schneider-Kamp[1]

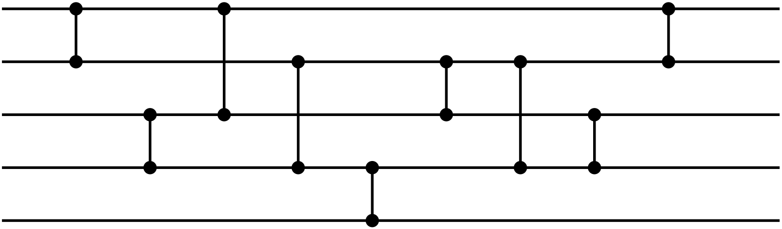
[1]Dept. Mathematics and Computer Science, Univ. Southern Denmark (Denmark)

[2]Ben-Gurion University of the Negev (Israel)

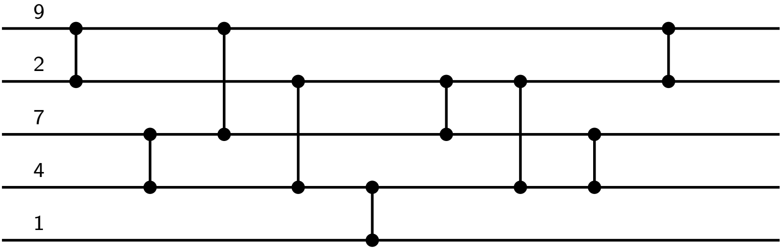
SYNASC
September 23rd, 2014

Outline

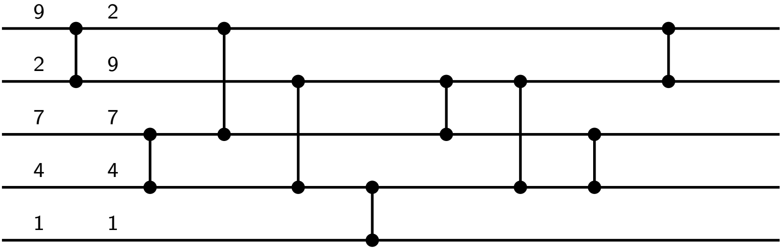
A sorting network



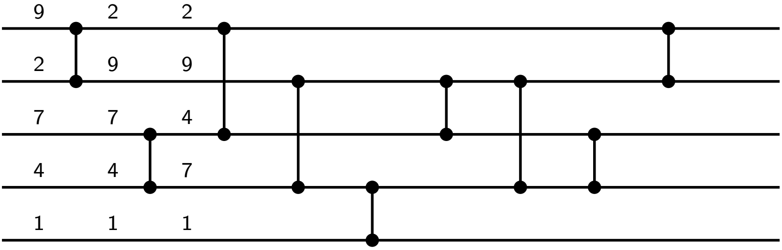
A sorting network



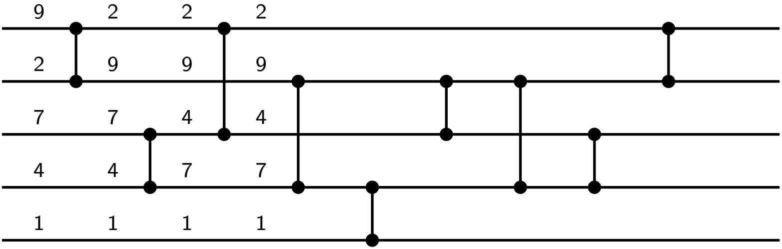
A sorting network



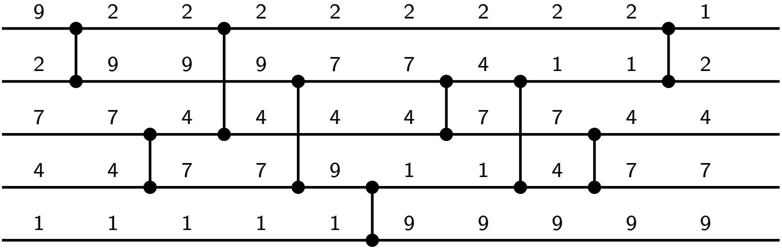
A sorting network



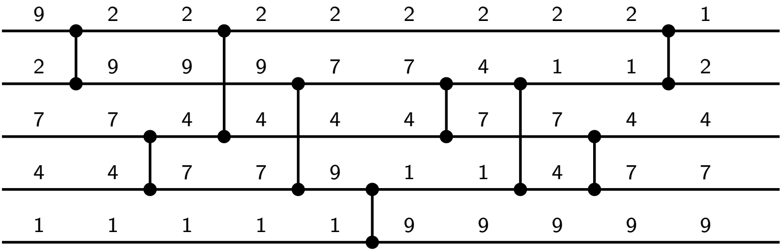
A sorting network



A sorting network



A sorting network



Size

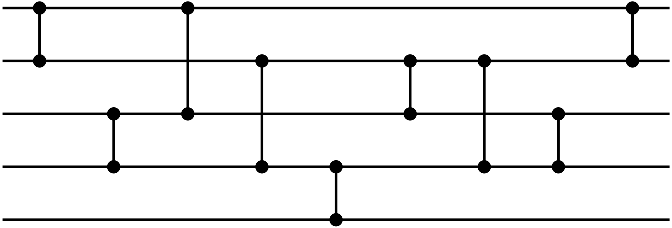
This net has 5 *channels* and 9 *comparators*.

A sorting network

Some of the comparisons may be performed in parallel:

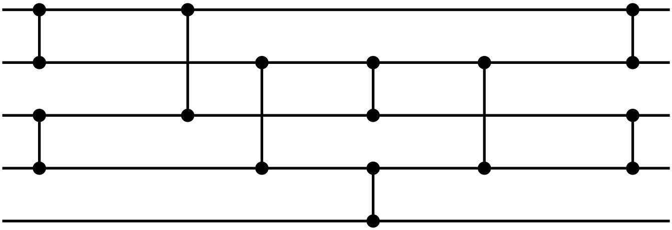
A sorting network

Some of the comparisons may be performed in parallel:



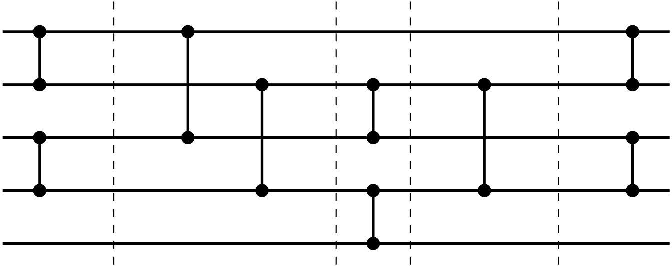
A sorting network

Some of the comparisons may be performed in parallel:



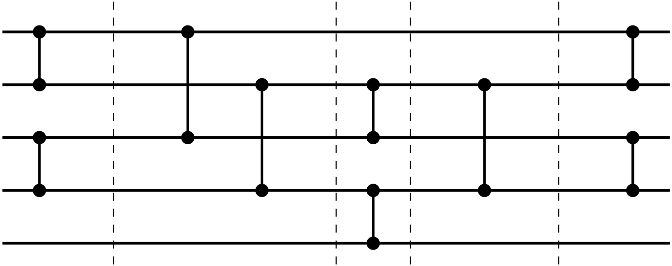
A sorting network

Some of the comparisons may be performed in parallel:



A sorting network

Some of the comparisons may be performed in parallel:

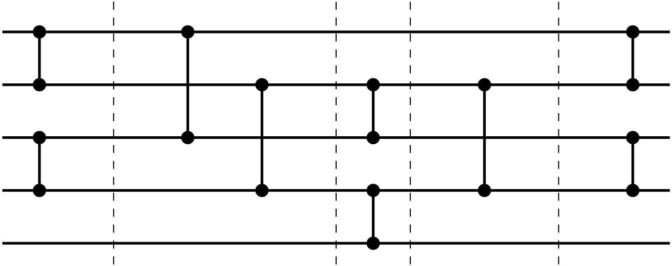


Depth

This net has 5 *layers*.

A sorting network

Some of the comparisons may be performed in parallel:



Depth

This net has 5 *layers*.

See Donald E. Knuth, *The Art of Computer Programming*, vol. 3 for more details

The optimization problems

The size problem

What is the minimal number of *comparators* on a sorting network on n channels (S_n)?

The depth problem

What is the minimal number of *layers* on a sorting network on n channels (T_n)?

The optimization problems

The size problem

What is the minimal number of *comparators* on a sorting network on n channels (S_n)?

The depth problem

What is the minimal number of *layers* on a sorting network on n channels (T_n)?

Knuth 1973

n	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
T_n	3	3	5	5	6	6	7	7	8	8	9	9	9	9	11
							6	6	6	6	6	6	6	6	6

The optimization problems

The size problem

What is the minimal number of *comparators* on a sorting network on n channels (S_n)?

The depth problem

What is the minimal number of *layers* on a sorting network on n channels (T_n)?

Parberry 1991

n	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
T_n	3	3	5	5	6	6	**7**	**7**	8 **7**	8 **7**	9 **7**	9 **7**	9 **7**	9 **7**	11 **7**

The optimization problems

The size problem

What is the minimal number of *comparators* on a sorting network on n channels (S_n)?

The depth problem

What is the minimal number of *layers* on a sorting network on n channels (T_n)?

Bundala & Závodný 2013

n	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
T_n	3	3	5	5	6	6	7	7	**8**	**8**	**9**	**9**	**9**	**9**	11 **9**

An exponential explosion

- Upper bounds obtained by concrete examples (1960s)
- Lower bounds obtained by mathematical arguments
- HUGE number of nets

An exponential explosion

- Upper bounds obtained by concrete examples (1960s)
- Lower bounds obtained by mathematical arguments
- HUGE number of nets
- Parberry (1991)
  - exploration of symmetries
  - fixed first layer
  - 200 hours of computation

An exponential explosion

- Upper bounds obtained by concrete examples (1960s)
- Lower bounds obtained by mathematical arguments
- HUGE number of nets
- Parberry (1991)
- Bundala & Závodný (2013)
  - exploration of symmetries
  - reduced set of two-layer prefixes
  - intensive SAT-solving

An exponential explosion

- Upper bounds obtained by concrete examples (1960s)
- Lower bounds obtained by mathematical arguments
- HUGE number of nets
- Parberry (1991)
- Bundala & Závodný (2013)
- These techniques do not scale for T_{17}
  $\approx 211 \times 10^6$ possibilities for each layer when $n = 17$
- These techniques are not directly applicable to the size problem
  36 possibilities for each comparator when $n = 9$, so
  $36^{24} \approx 2.2 \times 10^{37}$ 24-comparator nets

Outline

Comparator networks

A *comparator network* C on n wires is a sequence of *comparators* (i, j) with $1 \leq i < j \leq n$.

The *output* of C on a sequence $\vec{x} = x_1 \ldots x_n$ is denoted $C(\vec{x})$.

The set of binary outputs of C is
$\text{outputs}(C) = \{C(\vec{x}) \mid x \in \{0, 1\}^n\}$.

A comparator network C is a *sorting network* if $C(\vec{x})$ is sorted for every input $\vec{x}$.

Well-known results

0–1 lemma (Knuth 1973)

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

Well-known 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).

Well-known 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)

Let C and C' be comparator networks such that $\text{outputs}(C) \subseteq \text{outputs}(C')$. If $C'; N$ is a sorting network, then so is $C; N$.

Well-known 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)

Let C and C' be comparator networks such that $\mathrm{outputs}(C) \subseteq \mathrm{outputs}(C')$. If $C'; N$ is a sorting network, then so is $C; N$.

Proof

$$
\begin{array}{ccccc}
\{0,1\}^n & \xrightarrow{C} & X & & \\
 & & \rotatebox{90}{$\subseteq$} & & \\
\{0,1\}^n & \xrightarrow{C'} & X' & \xrightarrow{N} & S
\end{array}
$$

Well-known 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)

Let C and C' be comparator networks such that $\text{outputs}(C) \subseteq \text{outputs}(C')$. If $C'; N$ is a sorting network, then so is $C; N$.

Corollary

There is a minimal-depth sorting network on n channels whose first layer F_n contains the comparators $(1,2)$, $(3,4)$, $(5,6)$, &c.

Finding the value of T_{13}

The strategy

1. Generate all saturated two-layer networks with first layer F_{13}.
   Saturated: syntactic notion we can impose to reduce candidates.

Finding the value of T_{13}

The strategy

1. Generate all saturated two-layer networks with first layer F_{13}.
2. Remove equivalent nets.
   Equivalent: up to “renaming” of channels.

Finding the value of T_{13}

The strategy

1. Generate all saturated two-layer networks with first layer F_{13}.
2. Remove equivalent nets.
3. Remove some more nets.
   Semantic criteria can be used to eliminate candidates.

Finding the value of T_{13}

The strategy

1. Generate all saturated two-layer networks with first layer F_{13}.
2. Remove equivalent nets.
3. Remove some more nets.
4. Use a SAT-solver to find out if the remaining nets can be extended to a sorting network.

Finding the value of T_{13}

The strategy

1. Generate all saturated two-layer networks with first layer F_{13}.
2. Remove equivalent nets.
3. Remove some more nets.
4. Use a SAT-solver to find out if the remaining nets can be extended to a sorting network.

n	5	6	7	8	9	10	11	12	13
$\lvert G_n \rvert$	26	76	232	764	2620	9496	35696	140152	568504
$\lvert S_n \rvert$	10	51	74	513	700	6345	8174	93255	113008
$\lvert G_n/\approx \rvert$	18	28	74	101	295	350	1134	1236	4288
$\lvert S_n/\approx \rvert$	8		29		100		341		1155
red.	6	6	14	15	37	27	88	70	212
$\lvert R_n \rvert$	4	5	8	12	22	21	28	50	118

And for higher n?

This approach does not scale.

Computing equivalence of nets is very expensive (and not working correctly).

Reducing the set of candidates requires iterating over 2^n outputs and $n!$ permutations.

Furthermore, $T_{13} = T_{14} = T_{15} = T_{16}$.

To go beyond these values, we need different techniques.

Outline

Word representation for two-layer networks

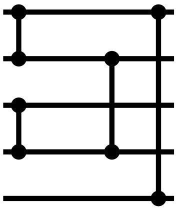
General idea

Represent two layer networks abstracting from the channel names.

Word representation for two-layer networks

General idea

Represent two layer networks abstracting from the channel names.

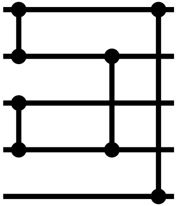


Head word:
01221

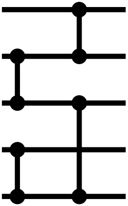
Word representation for two-layer networks

General idea

Represent two layer networks abstracting from the channel names.



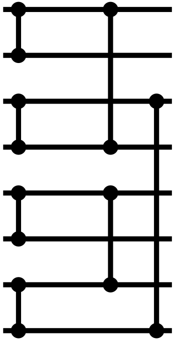
Head word:
01221



Word representation for two-layer networks

General idea

Represent two layer networks abstracting from the channel names.

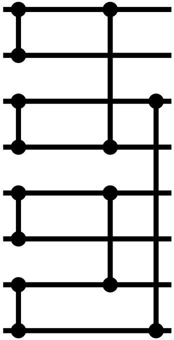


Stick word:
21121212
21212112

Word representation for two-layer networks

General idea

Represent two layer networks abstracting from the channel names.

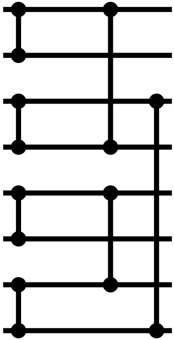


Stick word:
21121212
21212112

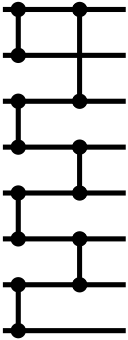
Word representation for two-layer networks

General idea

Represent two layer networks abstracting from the channel names.



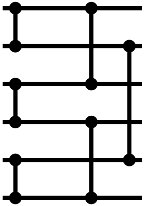
Stick word:
21121212
21212112



Word representation for two-layer networks

General idea

Represent two layer networks abstracting from the channel names.

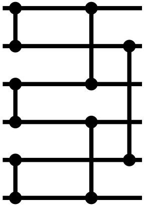


Cycle word:
121221
122121
122112

Word representation for two-layer networks

General idea

Represent two layer networks abstracting from the channel names.



Cycle word:

121221

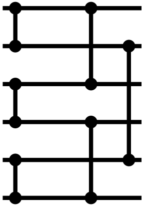
122121

122112

Word representation for two-layer networks

General idea

Represent two layer networks abstracting from the channel names.



Cycle word:

121221

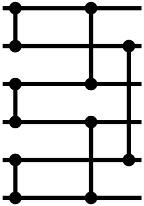
122121

122112

Word representation for two-layer networks

General idea

Represent two layer networks abstracting from the channel names.

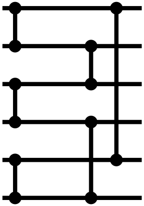


Cycle word:

121221

122121

122112



Word representation for two-layer networks

General idea

Represent two layer networks abstracting from the channel names.

Every net generates a unique word, and every well-formed word generates a unique net. The functions net-to-word and word-to-net form an adjunction.

A regular language for words

$$
\begin{aligned}
\text{Word} &::= \text{Head} \mid \text{Stick} \mid \text{Cycle} \\
\text{Head} &::= 0(12+21)^* \\
\text{Stick} &::= (12+21)^+ \\
\text{Cycle} &::= 12(12+21)^*(1+2)
\end{aligned}
$$

A regular language for words

$$
\begin{aligned}
\text{Word} &::= \text{Head} \mid \text{Stick} \mid \text{Cycle} \\
\text{Head} &::= 0(12 + 21)^* \\
\text{Stick} &::= (12 + 21)^+ \\
\text{Cycle} &::= 12(12 + 21)^*(1 + 2)
\end{aligned}
$$

Generating all words and filtering to obtain only the lexicographically smallest is very easy for the relevant values of n.

A regular language for words

$$\begin{aligned}
\text{Word} &::= \text{Head} \mid \text{Stick} \mid \text{Cycle} \\
\text{Head} &::= 0(12+21)^* \\
\text{Stick} &::= (12+21)^+ \\
\text{Cycle} &::= 12(12+21)^*(1+2)
\end{aligned}$$

Generating all words and filtering to obtain only the lexicographically smallest is very easy for the relevant values of n. Two-layer comparator networks can be represented by multi-sets of words. By choosing a canonical representation of multi-sets, we can easily generate exactly one representative for all two-layer networks with first layer F_n modulo equivalence.

Saturation

Saturation

A comparator network C is *redundant* if there exists a network C' obtained from C by removing a comparator such that $\text{outputs}(C') = \text{outputs}(C)$.
A network C is *saturated* if it is non-redundant and every network C' obtained by adding a comparator to the last layer of C satisfies $\text{outputs}(C') \not\subseteq \text{outputs}(C)$.

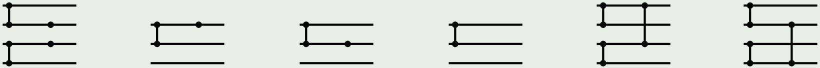
Saturation

Saturation

A comparator network C is *redundant* if there exists a network C' obtained from C by removing a comparator such that $\text{outputs}(C') = \text{outputs}(C)$.
A network C is *saturated* if it is non-redundant and every network C' obtained by adding a comparator to the last layer of C satisfies $\text{outputs}(C') \not\subseteq \text{outputs}(C)$.

Saturation theorem

Let C be a two-layer network. Then C is saturated iff C contains none of the following two-layer patterns.



Optimizations

It is easy to restrict the grammar above to generate only multi-sets of words corresponding to saturated nets.

We can again generate all saturated networks very efficiently for n up to 40.

A similar technique encodes other syntactic criteria.

Some numerology

n	5	6	7	8	9	10	11	12	13
$\lvert G_n\rvert$	26	76	232	764	2,620	9,496	35,696	140,152	568,504
$\lvert S_n\rvert$	10	28	70	230	676	2,456	7,916	31,374	109,856
$\lvert R(G_n)\rvert$	16	20	52	61	165	152	482	414	1,378
$\lvert R(S_n)\rvert$	6	6	14	15	37	27	88	70	212
$\lvert R_n\rvert$	4	5	8	12	22	21	28	50	117

n	14	15	16	17	18
$\lvert G_n\rvert$	2,390,480	10,349,536	46,206,736	211,799,312	997,313,824
$\lvert S_n\rvert$	467,716	1,759,422	7,968,204	31,922,840	152,664,200
$\lvert R(G_n)\rvert$	1,024	3,780	2,627	10,187	6,422
$\lvert R(S_n)\rvert$	136	494	323	1,149	651
$\lvert R_n\rvert$	94	262	211	609	411

n	19	20	25	30	35	40
$\lvert R(S_n)\rvert$	2,632	1,478	30,312	64,168	1,604,790	2,792,966
$\lvert R_n\rvert$	1,367	894	15,469	34,486	806,710	1,429,836

Outline

Results

- Efficient generation of two-layer prefixes for comparator networks
- Representation can capture different important semantic properties
- Identified relevant sets of networks for open cases
- Bottleneck is now processing each relevant network