sorting networks
the end game

luís cruz-filipe[1] michael codish[2]
peter schneider-kamp[1]

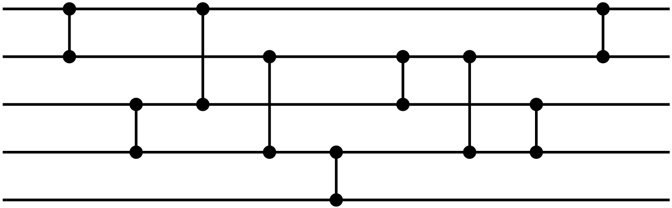
[1]department of mathematics and computer science
university of southern denmark

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

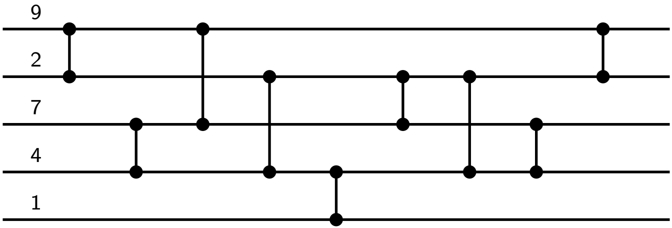
labmag seminar
april 13th, 2015

outline

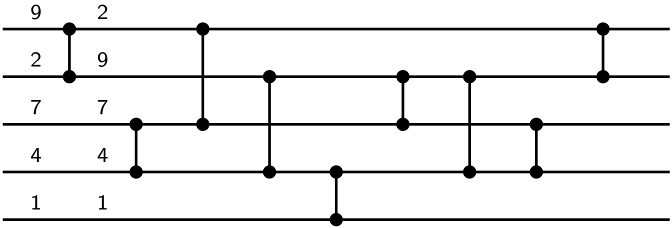
a sorting network



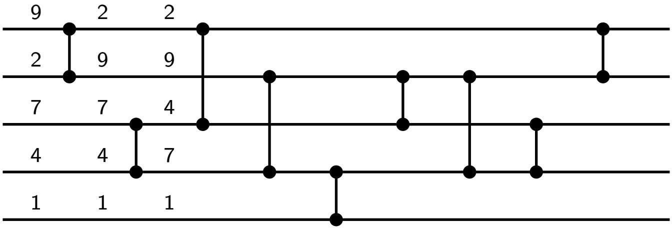
a sorting network



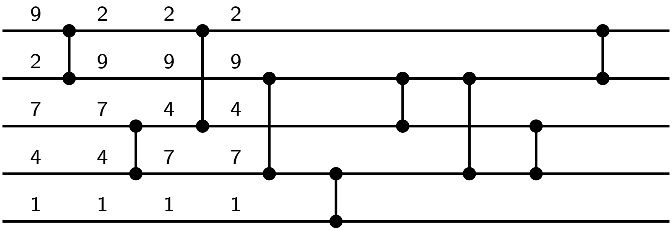
a sorting network



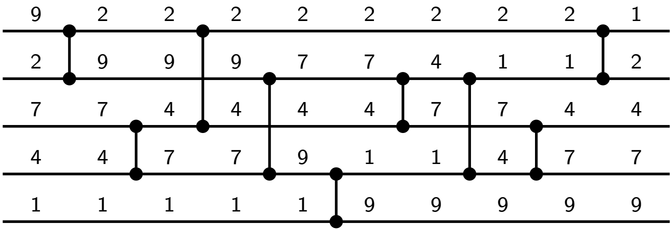
a sorting network



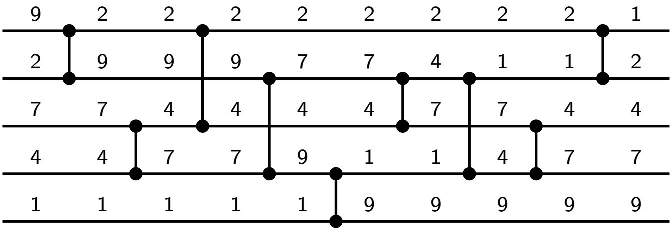
a sorting network



a sorting network

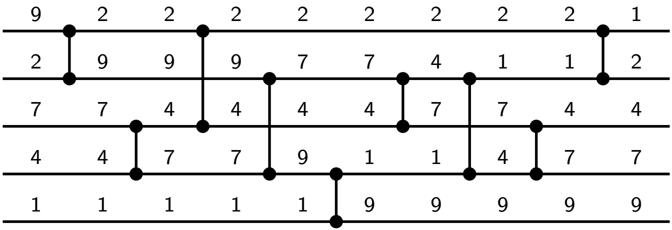


a sorting network



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

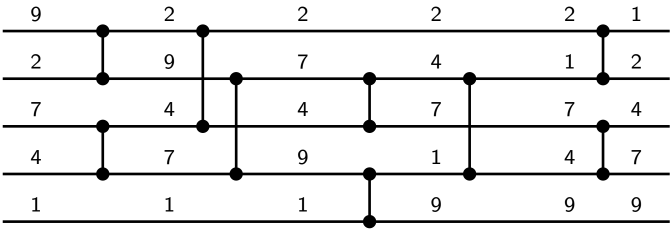
a sorting network



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

some of the comparisons may be performed in parallel

a sorting network

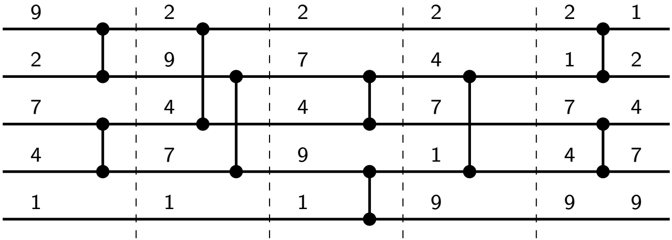


size

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

some of the comparisons may be performed in parallel

a sorting network

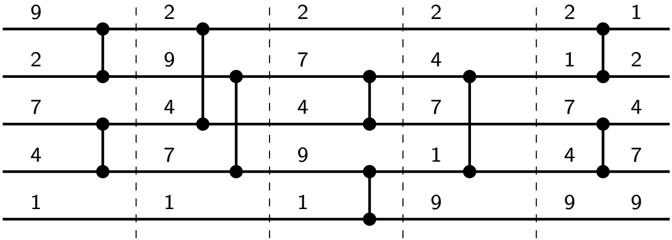


size

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

some of the comparisons may be performed in parallel

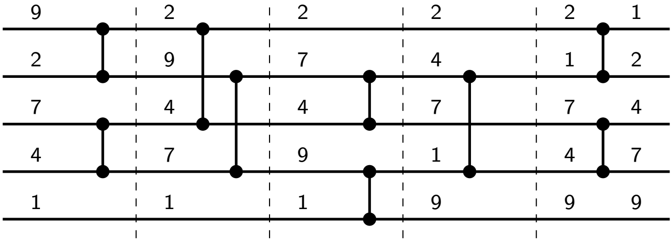
a sorting network



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

the optimization problems

the optimal size problem

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

the optimal depth problem

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

the optimization problems

the optimal size problem

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

the optimal depth problem

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

knuth 1973

n	1	2	3	4	5	6	7	8	9	10
t_n	0	1	3	3	5	5	6	6	7 6	7 6

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

the optimization problems

the optimal size problem

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

the optimal depth problem

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

parberry 1991

n	1	2	3	4	5	6	7	8	9	10
t_n	0	1	3	3	5	5	6	6	**7**	**7**

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

the optimization problems

the optimal size problem

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

the optimal depth problem

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

bundala & závodný 2013

n	1	2	3	4	5	6	7	8	9	10
t_n	0	1	3	3	5	5	6	6	7	7

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

the optimization problems

the optimal size problem

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

the optimal depth problem

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

ehlers & müller 2014

n	1	2	3	4	5	6	7	8	9	10
t_n	0	1	3	3	5	5	6	6	7	7

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

the optimization problems

the optimal size problem

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

the optimal depth problem

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

ehlers & müller 2015

n	1	2	3	4	5	6	7	8	9	10
t_n	0	1	3	3	5	5	6	6	7	7

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

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 $\rightsquigarrow$ fixed first layer
- exhaustive search (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

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

bundala & závodný 2013

- 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

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

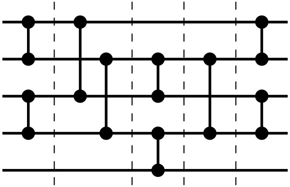
bundala & závodný 2013

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

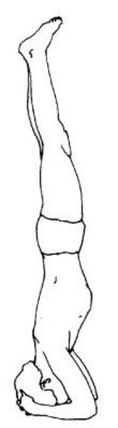
however... these techniques do not scale for t_{17}

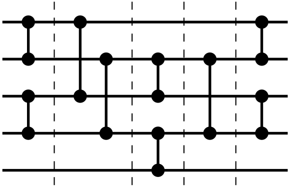
- sat-solvers cannot handle two-layer prefixes
- too many possibilities for third layer

inspirational sources

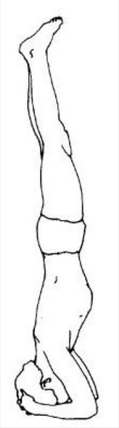


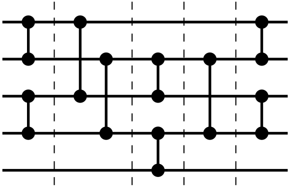
inspirational sources





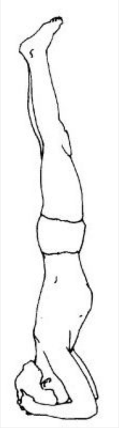
inspirational sources

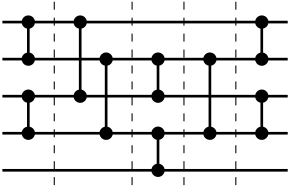




main idea study the properties of the *last* layers of sorting networks

inspirational sources





main idea study the properties of the *last* layers of sorting networks

- (surprisingly) never done before
- very different problem

outline

redundancy

redundant comparator

let $C; (i, j); C'$ be a comparator network
the comparator (i, j) is *redundant* if $x_i \leq x_j$ for all sequences $x_1 \ldots x_n \in \text{outputs}(C)$

lemma

if D and D' only differ in redundant comparators, then D is a sorting network iff D' is a sorting network

redundancy

redundant comparator
let $C; (i,j); C'$ be a comparator network
the comparator (i,j) is *redundant* if $x_i \leq x_j$ for all sequences $x_1 \ldots x_n \in \text{outputs}(C)$

lemma
if D and D' only differ in redundant comparators, then D is a sorting network iff D' is a sorting network

goal
restrict the search space by disallowing redundant comparators

problem
redundancy is a semantic property
$\rightsquigarrow$ not easily encodable in sat

the last layer

lemma all comparators in the last layer of a non-redundant sorting network are of the form $(i, i+1)$

the last layer

lemma all comparators in the last layer of a non-redundant sorting network are of the form $(i, i+1)$

theorem there are $f_{n+1} - 1$ possible last layers in an n-channel sorting network with no redundancy

fibonacci sequence $f_1 = f_2 = 1,\ f_{n+2} = f_{n+1} + f_n$

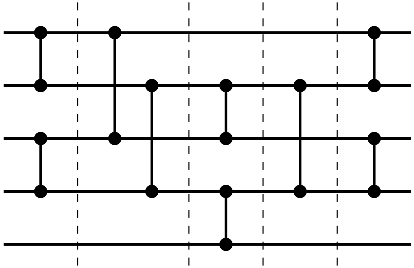
$\rightsquigarrow$ this reduces the number of possible last layers on 17 channels from 211,799,312 to just 2,583

blocks i/ii

k-block a k-block in a sorting network is a set of channels that are connected after layer k

blocks i/ii

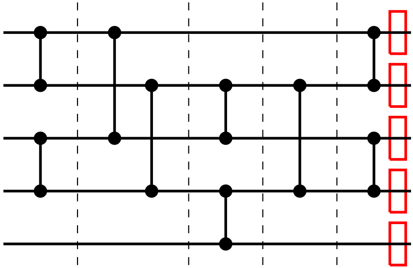
k-block a k-block in a sorting network is a set of channels that are connected after layer k



blocks i/ii

k-block

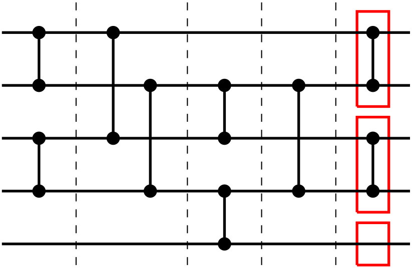
a *k*-block in a sorting network is a set of channels that are connected after layer *k*



5-blocks

blocks i/ii

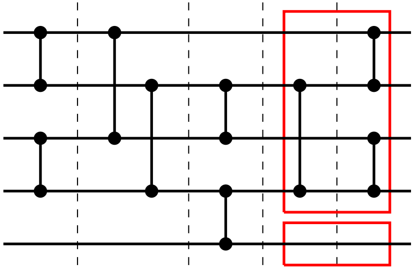
k-block a *k*-block in a sorting network is a set of channels that are connected after layer *k*



4-blocks

blocks i/ii

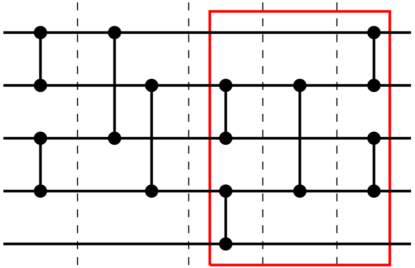
k-block a k-block in a sorting network is a set of channels that are connected after layer k



3-blocks

blocks i/ii

k-block a k-block in a sorting network is a set of channels that are connected after layer k

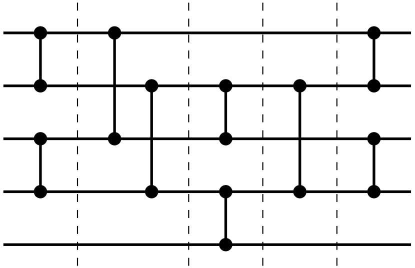


2-blocks

blocks i/ii

k-block

a k-block in a sorting network is a set of channels that are connected after layer k

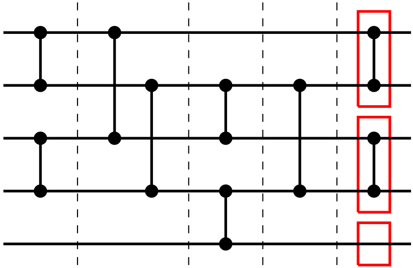


lemma

for every input $\bar{x} \in \{0,1\}^n$, there is at most one k-block that receives both 0s and 1s as inputs

blocks i/ii

k-block a k-block in a sorting network is a set of channels that are connected after layer k



4-blocks

lemma for every input $\bar{x} \in \{0,1\}^n$, there is at most one k-block that receives both 0s and 1s as inputs

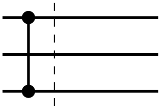
blocks ii/ii

theorem every comparator at layer k of a non-redundant sorting network connects adjacent k-blocks

blocks ii/ii

theorem every comparator at layer k of a non-redundant sorting network connects adjacent k-blocks

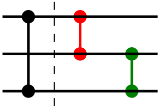
corollary restrictions on the last two layers



blocks ii/ii

theorem every comparator at layer k of a non-redundant sorting network connects adjacent k-blocks

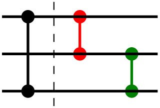
corollary restrictions on the last two layers

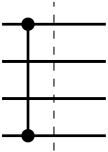


blocks ii/ii

theorem every comparator at layer k of a non-redundant sorting network connects adjacent k-blocks

corollary restrictions on the last two layers

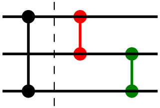


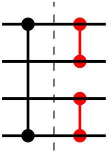


blocks ii/ii

theorem every comparator at layer k of a non-redundant sorting network connects adjacent k-blocks

corollary restrictions on the last two layers

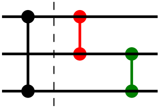


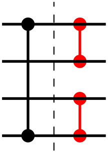


blocks ii/ii

theorem every comparator at layer k of a non-redundant sorting network connects adjacent k-blocks

corollary restrictions on the last two layers



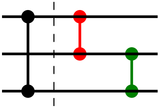


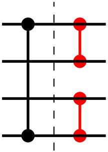
this substantially reduces the number of possibilities for the two last layers

blocks ii/ii

theorem every comparator at layer k of a non-redundant sorting network connects adjacent k-blocks

corollary restrictions on the last two layers





this substantially reduces the number of possibilities for the two last layers

. . . but it is not enough

outline

new idea

we can reduce the search state even more by *adding* redundant comparators!

revisiting the last layer

new idea we can reduce the search state even more by *adding* redundant comparators!

llnf a sorting network is in *last layer normal form* if

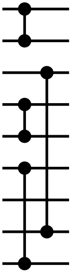
- its last layer only contains comparators between adjacent channels
- its last layer does not contain adjacent unused channels

revisiting the last layer

new idea we can reduce the search state even more by *adding* redundant comparators!

llnf a sorting network is in *last layer normal form* if

- its last layer only contains comparators between adjacent channels
- its last layer does not contain adjacent unused channels

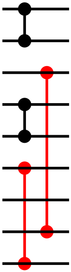


revisiting the last layer

new idea we can reduce the search state even more by *adding* redundant comparators!

llnf a sorting network is in *last layer normal form* if

- its last layer only contains comparators between adjacent channels
- its last layer does not contain adjacent unused channels

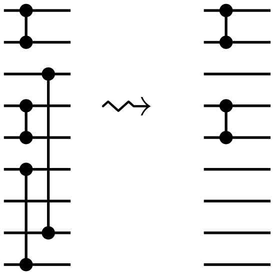


revisiting the last layer

new idea we can reduce the search state even more by *adding* redundant comparators!

llnf a sorting network is in *last layer normal form* if

- its last layer only contains comparators between adjacent channels
- its last layer does not contain adjacent unused channels

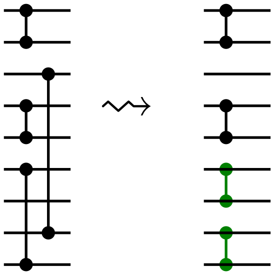


revisiting the last layer

new idea we can reduce the search state even more by *adding* redundant comparators!

llnf a sorting network is in *last layer normal form* if

- its last layer only contains comparators between adjacent channels
- its last layer does not contain adjacent unused channels

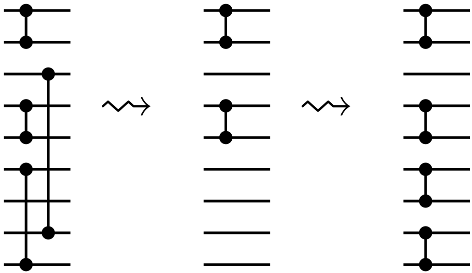


revisiting the last layer

new idea we can reduce the search state even more by *adding* redundant comparators!

llnf a sorting network is in *last layer normal form* if

- its last layer only contains comparators between adjacent channels
- its last layer does not contain adjacent unused channels



some more numerology

llnf a sorting network is in *last layer normal form* if

- its last layer only contains comparators between adjacent channels
- its last layer does not contain adjacent unused channels

some more numerology

llnf a sorting network is in *last layer normal form* if

- its last layer only contains comparators between adjacent channels
- its last layer does not contain adjacent unused channels

theorem there are p_{n+5} last layers in llnf on n channels

padovan sequence $p_0 = 1,\ p_1 = p_2 = 0,\ p_{n+3} = p_{n+1} + p_n$

some more numerology

llnf a sorting network is in *last layer normal form* if

- its last layer only contains comparators between adjacent channels
- its last layer does not contain adjacent unused channels

theorem there are p_{n+5} last layers in llnf on n channels

padovan sequence $p_0 = 1,\ p_1 = p_2 = 0,\ p_{n+3} = p_{n+1} + p_n$

$\rightsquigarrow$ this further reduces the number of possible last layers on 17 channels from 2,583 to only 86

co-saturation i/ii

generalization

we can apply the same reasoning to previous layers

co-saturation i/ii

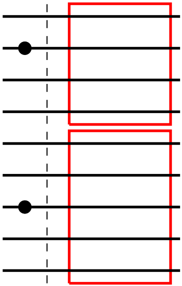
generalization we can apply the same reasoning to previous layers

lemma if $i < j$ are two channels unused in layer k of a sorting network belonging to different k-blocks, then the comparator (i, j) in layer k is redundant

co-saturation i/ii

generalization we can apply the same reasoning to previous layers

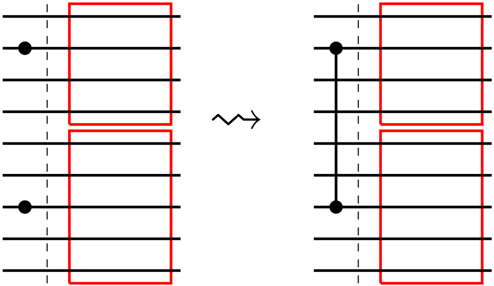
lemma if $i < j$ are two channels unused in layer k of a sorting network belonging to different k-blocks, then the comparator (i, j) in layer k is redundant



co-saturation i/ii

generalization we can apply the same reasoning to previous layers

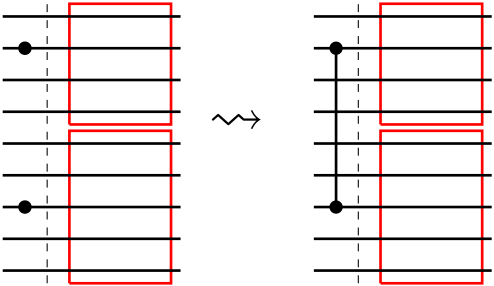
lemma if $i < j$ are two channels unused in layer k of a sorting network belonging to different k-blocks, then the comparator (i, j) in layer k is redundant



co-saturation i/ii

generalization we can apply the same reasoning to previous layers

lemma if $i < j$ are two channels unused in layer k of a sorting network belonging to different k-blocks, then the comparator (i, j) in layer k is redundant



lemma (some stuff about "sliding" comparators)

co-saturation ii/ii

co-saturation

we can characterize the networks resulting from applying these transformations to the two last layers

co-saturation ii/ii

co-saturation — we can characterize the networks resulting from applying these transformations to the two last layers

co-saturation theorem — if there is a sorting network on n channels, then there is a co-saturated sorting network on n channels with the same depth

co-saturation ii/ii

co-saturation we can characterize the networks resulting from applying these transformations to the two last layers

co-saturation theorem if there is a sorting network on n channels, then there is a co-saturated sorting network on n channels with the same depth

$\rightsquigarrow$ for $n = 17$, there are only 45,664 possibilities for the last two layers of a co-saturated sorting network

practical impact

the good news

we can encode co-saturation in sat

		unrestricted last two layers			
		slowest instance			total
n	#cases	#clauses	#vars	time	time
15	262	278,312	18,217	754.74	**130,551.42**
16	211	453,810	27,007	1,779.14	**156,883.21**

		co-saturated last two layers			
		slowest instance			total
n	#cases	#clauses	#vars	time	time
15	262	335,823	25,209	148.35	**19,029.26**
16	211	314,921	22,901	300.07	**24,604.53**

outline

results

- *necessary* conditions on last two layers
(was: *sufficient* conditions on first two layers)
- co-saturation
- 6× speedup on optimal depth problem
- similar techniques give 4× speedup on optimal size problem
- can find 10-layer sorting network on 17 channels in one hour
- key ingredient in computing exact value of t_{17}

thank you!