Exploiting Common Subexpressions in Numerical CSPs

Ignacio Araya, Bertrand Neveu, Gilles Trombettoni

INRIA, Université de Nice-Sophia, CERTIS, France

SWIM, June 2008, Montpellier, France

Outline

Common Subexpressions Elimination (CSE)

- The symbolic form of the equations is crucial for interval-based solving techniques.
- **CSE** is an important feature in optimization of code.
- CSE consists in replacing common subexpressions (CS) by auxiliary variables.

For example:

```
a = b * c + g
d = b * c * d
```

It may be worth (in performance) transforming the code to:

```
tmp = b * c
a = tmp + g
d = tmp * d
```

Common Subexpressions Elimination (CSE)

- The symbolic form of the equations is crucial for interval-based solving techniques.
- **CSE** is an important feature in optimization of code.
- CSE consists in replacing common subexpressions (CS) by auxiliary variables.

  For example:

  ```
  a = b * c + g
  d = b * c * d
  ```

  It may be worth (in performance) transforming the code to:

  ```
  tmp = b * c
  a = tmp + g
  d = tmp * d
  ```

Common Subexpressions Elimination (CSE)

- In interval analysis, Schichl and Neumaier proposed a unique DAG to represent a system of equations.
- It is thought that the gain in CPU time due to common subexpressions is only **due to a reduction of the number of operations**.

Contributions

- State when CSE may be useful to improve the performance of interval solvers (bringing *a better contraction/filtering*).
- An algorithm (`I-CSE`) that generates a new system of equations.
  - I-CSE replace CSs only if it may improve the contraction/filtering of solvers.
  - I-CSE is able to find all the maximal CS including so-called conflictive CSs that overlap.
  - I-CSE is not intrusive.

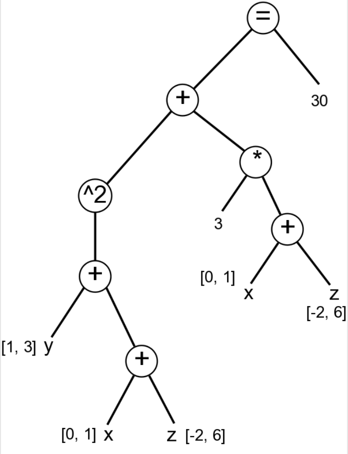
Outline

HC4 Algorithm

- Filtering algorithm (similar to AC3) used in interval solvers. Handles the constraints individually with a procedure HC4-revise.
- The system is represented as a set of binary trees.
- HC4-revise works in two 2 phases: forward or **evaluation phase** and backward or **narrowing phase**.

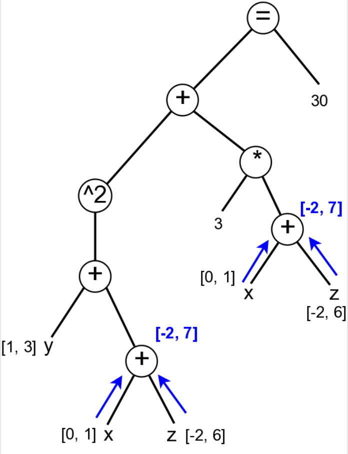
HC4 Algorithm

Example of `HC4-revise`: $(x + y + z)^2 + 3(x + z) = 30$



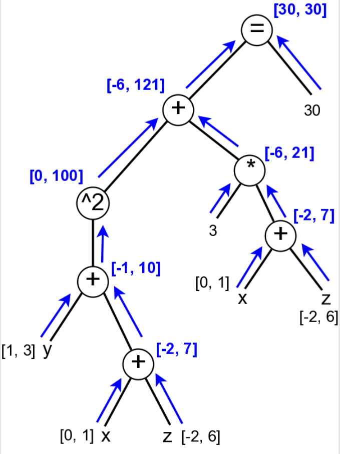
HC4 Algorithm

Example of `HC4-revise`: $(x+y+z)^2+3(x+z)=30$



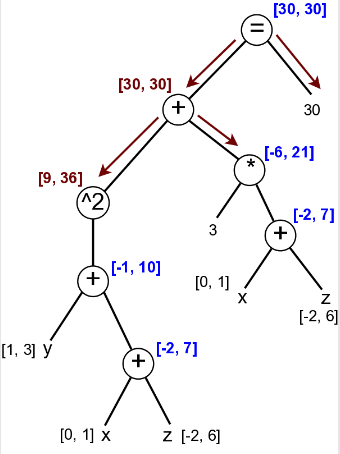
HC4 Algorithm

Example of `HC4-revise`: $(x + y + z)^2 + 3(x + z) = 30$



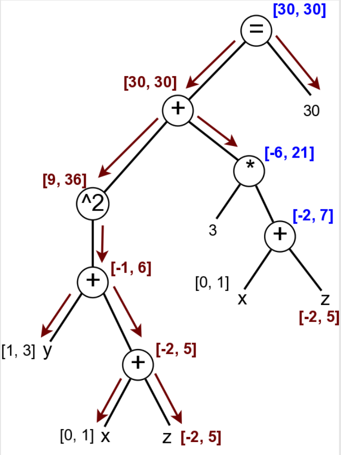
HC4 Algorithm

Example of `HC4-revise`: $(x + y + z)^2 + 3(x + z) = 30$



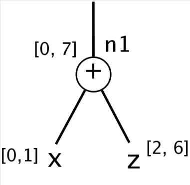
HC4 Algorithm

Example of `HC4-revise`: $(x+y+z)^2 + 3(x+z) = 30$



Over-estimation in HC4 caused by CSs

Example: The sum $x + z$ is shared by two constraints A and B.
$A := ... * (x + z)^3 + ... = 0$ $\quad$ $B := ... + x + z = 0$

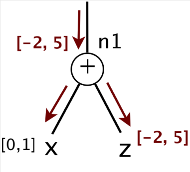


Observation

The contraction obtained by narrowing an expression (n_1) is in general lost in the next evaluation of the same expression (n_2).

Over-estimation in HC4 caused by CSs

Example: The sum $x + z$ is shared by two constraints A and B.
$A := \ldots * (x + z)^3 + \ldots = 0$ $\quad B := \ldots + x + z = 0$

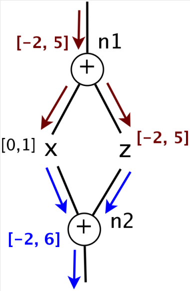


Observation

The contraction obtained by narrowing an expression (n_1) is in general lost in the next evaluation of the same expression (n_2).

Over-estimation in HC4 caused by CSs

Example: The sum $x + z$ is shared by two constraints A and B.
$A := ... * (x + z)^3 + ... = 0$ $B := ... + x + z = 0$

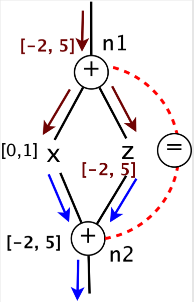


Observation

The contraction obtained by narrowing an expression (n_1) is in general lost in the next evaluation of the same expression (n_2).

Over-estimation in HC4 caused by CSs

Example: The sum $x + z$ is shared by two constraints A and B.
$A := ... * (x + z)^3 + ... = 0 \qquad B := ... + x + z = 0$

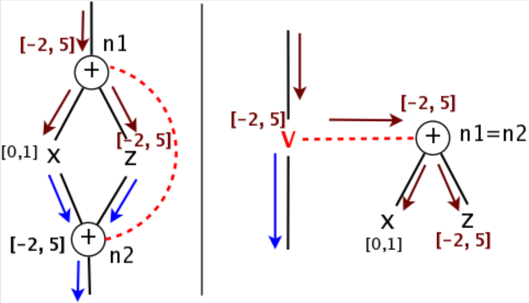


Observation

Adding the new constraint ($n_1 = n_2$) the contraction is not lost.

Over-estimation in HC4 caused by CSs

Example: `HC4-revise` in the new system
$A := \ldots * v^3 + \ldots = 0 \qquad B := \ldots + v = 0$



Solution

Replace n_1 and n_2 by a common variable v, and add a new constraint $v = x + z$.

Some properties

Proposition 1

HC4 obtains a **better or equivalent** filtering in a system modified by CSE.

Proposition 2 and 3

	x^2	x^3	sin	cos	log	*exp*	$+$	$\times$
Useful	X		X	X			X	X
Useless		X			X	X		

Proposition 4

The lost Δ of a binary sum $x + y$ can be estimated.
$\Delta \leq 2 \times \min(Diam(x), Diam(y))$.

Some properties

Proposition 1

HC4 obtains a **better or equivalent** filtering in a system modified by CSE.

Proposition 2 and 3

	x^2	x^3	sin	cos	log	*exp*	$+$	$\times$
Useful	X		X	X			X	X
Useless		X			X	X		

Proposition 4

The lost Δ of a binary sum $x + y$ can be estimated.
$\Delta \leq 2 \times \min(Diam(x), Diam(y))$.

Some properties

Proposition 1

HC4 obtains a **better or equivalent** filtering in a system modified by CSE.

Proposition 2 and 3

	x^2	x^3	sin	cos	log	*exp*	$+$	$\times$
Useful	X		X	X			X	X
Useless		X			X	X		

Proposition 4

The lost Δ of a binary sum $x + y$ can be estimated.
$\Delta \leq 2 \times \min(Diam(x), Diam(y))$.

Outline

The I-CSE Algorithm

- The novelty of I-CSE lies in the way **additive and multiplicative** CSs are taken into account.
- I-CSE manages **conflictive subexpressions**.
- Algorithm divided into 4 steps.

Step 1: DAG Generation

- Every equation in the system is represented by an n-ary tree.
- The trees are compacted into a DAG, merging together equivalent subtrees.
- Standard bottom-up procedure.

The I-CSE Algorithm

Initial System:

$$
\begin{aligned}
x^2 + y + (y + x^2 + y^3 - 1)^3 + x^3 &= 2 \\
\frac{(y^3 + x^2) \times (x^2 + cos(y)) + 14}{x^2 + cos(y)} &= 8
\end{aligned}
$$

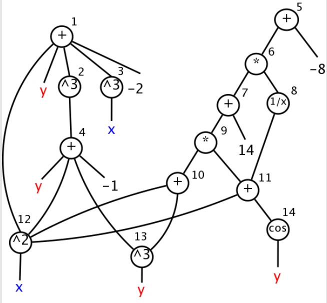
The I-CSE Algorithm

Initial System:

$$\begin{aligned} \mathbf{x^2} + y + (y + \mathbf{x^2} + y^3 - 1)^3 + x^3 &= 2 \\ \frac{(y^3 + \mathbf{x^2}) \times (\mathbf{x^2} + \mathbf{cos(y)}) + \mathbf{14}}{\mathbf{x^2} + \mathbf{cos(y)}} &= 8 \end{aligned}$$

The I-CSE Algorithm

Step 1: DAG Generation



Observations

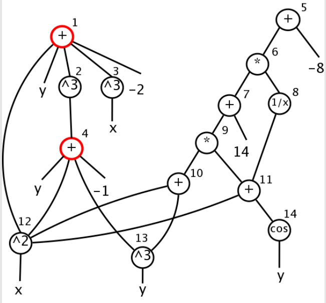
Equivalent nodes are merged together.

Step 2: Pairwise Intersection

- Nodes corresponding to n-ary sums (resp. multiplications) are intersected pairwise, creating **intersection nodes**.

The I-CSE Algorithm

Step 2: Pairwise intersection of nodes

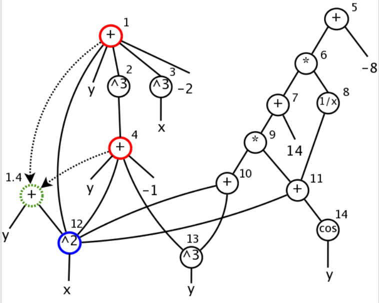


Observations

$+(\mathbf{x^2}, \mathbf{y}, node_2, x^3, -2)_1 \cap +(\mathbf{y}, \mathbf{x^2}, y^3, -1)_4 = +(\mathbf{y}, \mathbf{x^2})$

The I-CSE Algorithm

Step 2: Pairwise intersection of nodes

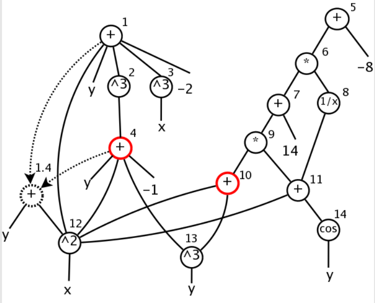


Observations

$+(\mathbf{x^2}, \mathbf{y}, node_2, x^3, -2)_1 \cap +(\mathbf{y}, \mathbf{x^2}, y^3, -1)_4 = +(\mathbf{y}, \mathbf{x^2})_{1.4}$

The I-CSE Algorithm

Step 2: Pairwise intersection of nodes

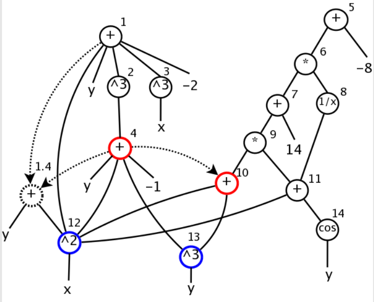


Observations

The set of intersection nodes correspond to all the maximal CSs between every pair of expressions.

The I-CSE Algorithm

Step 2: Pairwise intersection of nodes



Observations

The set of intersection nodes correspond to all the maximal CSs between every pair of expressions.

Step 3: Integrating CSs into the DAG

- The CSs are integrated into the DAG
- CSs of a node sharing terms are called **conflictive**.
- To attach conflictive CSs it is necessary to create redundant nodes.

  For example:

  $n_4 := y + \mathbf{x^2} + y^3 - 1$

  $n_{1.4} := y + \mathbf{x^2}$

  $n_{10} := \mathbf{x^2} + y^3$

  $n_4 := n_{1.4} + y^3 - 1$

  $n_{4B} := y + n_{10} - 1$

Step 3: Integrating CSs into the DAG

- The CSs are integrated into the DAG
- CSs of a node sharing terms are called **conflictive**.
- To attach conflictive CSs it is necessary to create redundant nodes.
  For example:
  $n_4 := y + \mathbf{x^2} + y^3 - 1$
  $n_{1.4} := y + \mathbf{x^2}$
  $n_{10} := \mathbf{x^2} + y^3$
  $n_4 := n_{1.4} + y^3 - 1$
  $n_{4B} := y + n_{10} - 1$

Step 3: Integrating CSs into the DAG

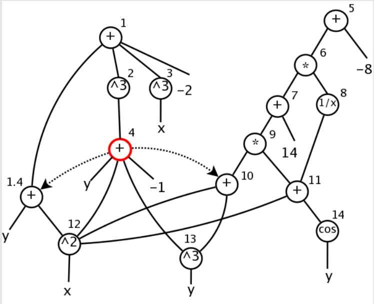
- The CSs are integrated into the DAG
- CSs of a node sharing terms are called **conflictive**.
- To attach conflictive CSs it is necessary to create redundant nodes.
  For example:
  $n_4 := y + \mathbf{x^2} + y^3 - 1$
  $n_{1.4} := y + \mathbf{x^2}$
  $n_{10} := \mathbf{x^2} + y^3$
  $n_4 := n_{1.4} + y^3 - 1$
  $n_{4B} := y + n_{10} - 1$

Step 3: Integrating CSs into the DAG

- The CSs are integrated into the DAG
- CSs of a node sharing terms are called **conflictive**.
- To attach conflictive CSs it is necessary to create redundant nodes.
  For example:
  $n_4 := y + \mathbf{x^2} + y^3 - 1$
  $n_{1.4} := y + \mathbf{x^2}$
  $n_{10} := \mathbf{x^2} + y^3$
  $n_4 := n_{1.4} + y^3 - 1$
  $n_{4B} := y + n_{10} - 1$

The I-CSE Algorithm

Step 3: Integrating intersection nodes (CSs) into the DAG

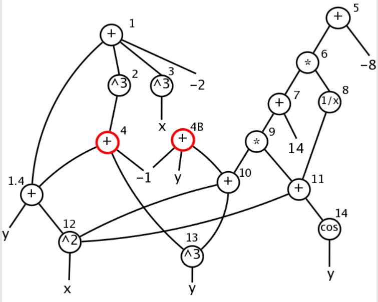


Observations

A redundant node is generated (4B) to attach coflictive CSs.

The I-CSE Algorithm

Step 3: Integrating intersection nodes (CSs) into the DAG

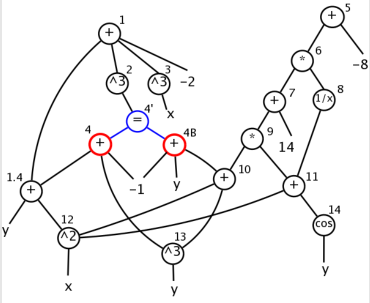


Observations

A redundant node is generated (4B) to attach coflictive CSs.

The I-CSE Algorithm

Step 3: Integrating intersection nodes (CSs) into the DAG



Observations

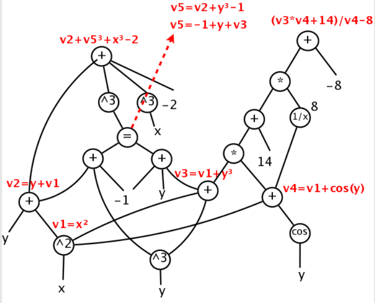
An equal node is generated to incorporate redundant nodes into the DAG.

Step 4: Generation of the new system

- It is possible to use directly the DAG obtained at step 3 (Vu et al.). But it implies modifying the propagation algorithm.
- To continue using classical algorithms, I-CSE generates a new system.
  - *Useful CSs* and equal nodes generate new variables and equations.

The I-CSE Algorithm

Step 4: Generation of the new system



Observations

The new equations are generated.

Step 4: Generation of the new system

The new generated system is:

$$\begin{aligned} v_2 + v_5^3 + x^3 - 2 &= 0 \\ \frac{v_3 \times v_4 + 14}{v_4} - 8 &= 0 \end{aligned}$$

$$\begin{aligned} v_1 &= x^2 & \qquad v_4 &= v_1 + cos(y) \\ v_2 &= y + v_1 & \qquad v_5 &= v_2 + y^3 - 1 \\ v_3 &= v_1 + y^3 & \qquad v_5 &= -1 + y + v_3 \end{aligned}$$

Outline

Implementation of I-CSE

- I-CSE has been implemented using Mathematica.
- Two variants that find fewer CSs: I-CSE-B and I-CSE-NC.
- Solving algorithms have been developed in Ibex library (C++).
- Branch and prune process.

Selected Benchmarks

Benchmark			I-CSE-B	ICSE-NC	I-CSE		Benchmark			I-CSE-B	ICSE-NC	I-CSE	
	#s	N	#cs	#cs	#cs	#rc		#s	N	#cs	#cs	#cs	#rc
6body	5	6	2	3	3	0	Kin1	16	6	13	13	19	3
Bellido	8	9	0	1	1	0	Pramanik	2	8	0	15	15	0
Brown-7	3	7	3	7	21	24	Prolog	0	21	0	7	7	0
Brown-7*	3	7	3	1	1	0	Rose	16	3	5	5	5	0
Brown-30	**2**	**30**	**26**	**53**	**435**	**783**	Trigexp1-30	1	30	29	29	29	0
BroyBand-20	1	20	22	37	97	73	Trigexp1-50	1	50	49	49	49	0
BroyBand-100	1	100	102	119	479	473	Trigexp2-11	0	11	15	15	15	0
Caprasse	18	4	6	7	11	2	Trigexp2-19	0	19	27	27	27	0
Design	1	9	3	3	3	0	Trigonom-5	2	5	7	9	20	14
Dis-Integral-6	1	6	4	6	18	9	Trigonom-5*	2	5	7	6	6	0
Dis-Integral-20	3	20	18	34	207	171	Trigonom-10	24	10	15	15	26	15
Eco9	16	8	0	3	7	1	Trigonom-10*	24	10	15	12	12	0
EqCombustion	4	5	7	8	11	1	Yamamura-8	7	8	5	10	36	48
ExtendWood-4	3	4	2	2	2	0	Yamamura-8*	7	8	5	1	1	0
Geneig	10	6	11	14	14	0	Yamamura-10	9	12	7	14	55	79
Hayes	1	8	9	8	8	0	Yamamura-10*	9	12	7	1	1	0
I5	30	10	3	4	10	5	Yamamura-12	9	12	9	18	78	119
Katsura-19	5	20	81	81	81	0	Yamamura-12*	9	12	9	1	1	0
Katsura-20	7	21	90	90	90	0	Yamamura-16	9	16	13	26	136	224

Results I-CSE

- I-CSE time < 1 second.

Results with HC4 and Interval Newton

Benchmark	TIME in second				TIME(Osys) / TIME			#Boxes		
	Osys	`ICSE-B`	`ICSE-NC`	`I-CSE`	`ICSE-B`	`ICSE-NC`	`I-CSE`	Osys	`ICSE-NC`	`I-CSE`
EqCombustion	>3600	26.1	0.35	**0.14**	>137	>10000	**>25000**	>1e+08	3967	**1095**
Rose	>3600	500	**101**	**101**	>7.2	>**35**	>**35**	>3e+07	**865099**	**865099**
Hayes	141	51.9	**15.7**	**15.7**	2.7	**9**	**9**	550489	**44563**	**44563**
6-body	0.22	**0.07**	**0.07**	**0.07**	**3.1**	**3.1**	**3.1**	4985	**495**	**495**
Design	176	65.2	**63.2**	**63.2**	2.7	**2.8**	**2.8**	425153	**122851**	**122851**
I5	>3600	>3600	**1534**	1565	?	>**2.3**	>2.3	>3e+07	**7e+06**	7e+06
Geneig	3323	2910	2722	2722	1.14	**1.22**	**1.22**	7e+08	4e+08	4e+08
Kin1	8.52	8.32	8.32	**8.01**	1.02	1.02	**1.06**	**905**	909	**905**
Pramanik	89.3	92.1	**84.9**	**84.9**	0.97	**1.05**	**1.05**	487255	**378879**	**378879**
Bellido	15.7	15.9	**15.6**	**15.6**	0.99	**1.01**	**1.01**	29759	**29319**	**29319**
Eco9	**23.9**	**23.9**	24	24.1	**1.00**	**1.00**	0.99	126047	117075	**110885**
Caprasse	**1.56**	1.81	1.68	2.16	0.86	0.93	0.72	8521	7793	**7491**
Brown-7*	500	350	**0.01**	**0.01**	1.42	**49500**	**49500**	6e+06	**95**	**95**
Dis-Integral-6	201	0.46	1.3	**0.03**	437	155	**6700**	653035	4157	**47**
ExtendWood-4	29.9	**0.03**	**0.03**	**0.03**	**997**	**997**	**997**	422705	**353**	**353**
Brown-7	500	350	30.7	**1.49**	1.42	16.1	**332**	6e+06	258601	**3681**
Trigexp2-11	1118	208	**56.2**	**56.2**	5.38	**19.9**	**19.9**	1e+06	**316049**	**316049**
Yamamura-8*	13	13.3	**0.75**	**0.75**	0.98	**17.3**	**17.3**	29615	**2161**	**2161**
Broy-Banded-20	778	759	261	**58.1**	1.03	2.98	**13.4**	172959	46761	**12623**
Trigonometric-5*	15.8	12.3	**1.49**	**1.49**	1.28	**10.6**	**10.6**	10531	**1503**	**1503**
Trigonometric-5	15.8	12.3	8.94	**6.97**	1.28	1.77	**2.27**	10531	7369	**5307**
Yamamura-8	13	13.3	44.6	**10.8**	0.98	0.3	**1.20**	29615	115211	**13211**
Katsura-19	**1430**	1583	1583	1583	0.90	0.90	0.90	**145839**	153193	153193
Trigexp1-30	**2465**	3244	3244	3244	0.76	0.76	0.76	1e+07	1e+07	1e+07

Results with 3BCID (using HC4) and Interval Newton

Benchmark	TIME in second				TIME(Init) / TIME			#Boxes		
	Init	ICSE-B	ICSE-NC	I-CSE	ICSE-B	ICSE-NC	I-CSE	**Init**	ICSE-NC	I-CSE
Rose	2882	5.17	**4.04**	**4.04**	557	**713**	**713**	4e+06	**5711**	**5711**
Prolog	38.5	60	**0.14**	**0.14**	0.64	**275**	**275**	4647	**11**	**11**
EqCombustion	0.42	0.37	**0.06**	**0.06**	1.35	7	7	427	**23**	**23**
Hayes	32.6	27.2	**5.67**	**5.67**	1.13	**5.7**	**5.7**	17455	**1675**	**1675**
Design	52	17.9	**13.3**	**13.3**	2.9	**3.9**	**3.9**	16359	**4401**	**4401**
I5	33.5	41.1	**17.9**	**17.8**	0.81	1.9	**1.9**	10619	4387	**4281**
6-body	0.14	**0.08**	0.1	0.1	**1.75**	1.4	1.4	173	**51**	**51**
Kin1	1.66	2.66	1.76	**1.23**	0.62	0.94	**1.35**	**85**	161	197
Bellido	10.3	10.4	**9.98**	**9.98**	1	**1.03**	**1.03**	4487	**4341**	**4341**
Eco9	**11.6**	**11.6**	12.4	13.2	**1**	0.94	0.88	6205	6045	**5749**
Pramanik	**73.8**	114	96.8	96.8	0.65	0.76	0.76	124663	**95305**	**95305**
Caprasse	1.96	2.51	2.5	2.92	0.74	0.78	0.67	1285	1311	**1219**
Geneig	**696**	1050	1050	1050	0.66	0.66	0.66	362225	**362045**	**362045**
Trigexp2-19	2308	2.23	**0.03**	**0.03**	1035	**77000**	**77000**	250178	**7**	**7**
Brown-7*	600	318	**0.01**	**0.01**	1.88	**60000**	**60000**	662415	**9**	**9**
ExtendWood-4	185	**0.03**	**0.03**	**0.03**	6167	6167	**6167**	669485	**35**	**35**
Dis-Integral-6	135	0.18	0.51	**0.03**	750	264	**4500**	86487	185	**7**
Brown-7	600	318	4.75	**0.22**	1.88	126	**2700**	**662415**	**2035**	**23**
Yamamura-12*	1751	1842	**1.01**	**1.01**	0.95	**1700**	**1700**	364105	**307**	**307**
Yamamura-12	1751	1842	31.1	**8.72**	0.95	56.3	**200**	364105	5647	**445**
Trigono-10*	1344	506	**19.4**	**19.4**	2.67	**69**	**69**	140512	**2033**	**2033**
Trigono-10	1344	506	156	**49.6**	2.67	8.62	**27**	140512	19883	**3339**
Broy-Banded-100	9.96	20.3	14.8	**8.21**	0.49	0.67	**1.21**	13	23	**11**
Trigexp1-50	**0.15**	0.19	0.17	0.17	0.79	0.88	0.88	**1**	**1**	**1**
Katsura20	**3457**	5919	5919	5919	0.58	0.58	0.58	**62451**	120929	120929
Brown-30	>3600	>3600	>3600	**22.9**	?	?	**>150**	>210021	>151527	**31**
Dis-Integral-20	>3600	>3600	>3600	**1.12**	?	?	**>3200**	>111512	>75640	**39**
Yamamura-16	>3600	>3600	681	**35.6**	?	>5	**>100**	>522300	96341	**919**

Conclusion

- CSs can bring significant gains in filtering and not only a decrease in the number of operations.
- Gains in filtering can only be expected when the CSs do not correspond to monotonic and continuous unary operators (x^3, *log*).
- Gains of several orders of magnitude.
- Future work: Interval Newton.

Exploiting Common Subexpressions in Numerical CSPs

Ignacio Araya, Bertrand Neveu, Gilles Trombettoni

INRIA, Université de Nice-Sophia, CERTIS, France

SWIM, June 2008, Montpellier, France