Filtering Algorithms Based on the Word-RAM Model

Claude-Guy Quimper
Philippe Van Kessel

Filtering

- Which values in these domains do not satisfy the constraint $A + B = C$?

$$\text{dom}(A) = \{1, 5\}$$

$$\text{dom}(B) = \{1, 3, 5\}$$

$$\text{dom}(C) = \{2, 3, 6\}$$

Filtering

- Which values in these domains do not satisfy the constraint $A + B = C$?

$$\mathrm{dom}(A) = \{1, 5\}$$

$$\mathrm{dom}(B) = \{1, \quad 5\}$$

$$\mathrm{dom}(C) = \{2, \quad 6\}$$

Filtering

- Which values in these domains do not satisfy the constraint $A + B = C$?

$$\mathrm{dom}(A) = \{1, 5\}$$

$$\mathrm{dom}(B) = \{1, \quad 5\}$$

$$\mathrm{dom}(C) = \{2, \quad 6\}$$

- Each constraint has its own filtering algorithm.

Filtering

- Which values in these domains do not satisfy the constraint $A + B = C$?

$$\mathrm{dom}(A) = \{1, 5\}$$

$$\mathrm{dom}(B) = \{1, \quad 5\}$$

$$\mathrm{dom}(C) = \{2, \quad 6\}$$

- Each constraint has its own filtering algorithm.
- These algorithms are called millions of times during the search process.

Filtering

- Which values in these domains do not satisfy the constraint $A + B = C$?

$$\mathrm{dom}(A) = \{1, 5\}$$

$$\mathrm{dom}(B) = \{1, \quad 5\}$$

$$\mathrm{dom}(C) = \{2, \quad 6\}$$

- Each constraint has its own filtering algorithm.
- These algorithms are called millions of times during the search process.
- It therefore important that the design and the implementation of these algorithms make them as efficient as possible.

Random Access Machine (RAM)

- The running time efficiency of an algorithm is analyzed on a theoretical machine.
- Summary of the RAM: manipulating one bit of memory requires one unit of time.
- Adding two w-bit integers require O(w) time.
- Usually, we simplify by saying that the size of integers are bounded by a constant and therefore arithmetic operations execute in constant time.

Word-RAM

- A Word-RAM is a RAM that can manipulate w bits in a single instruction.
- If each bit represents a piece of input data then the Word-RAM can manipulate w pieces of data in constant time.
- One can hope to solve problems w times faster with a Word-RAM than with RAM.

Word-RAM Instructions

	7	6	5	4	3	2	1	0
x	0	1	0	0	1	1	0	1
y	1	0	0	1	1	0	1	1
x & y	0	0	0	0	1	0	0	1
$x \mid y$	1	1	0	1	1	1	1	1
$x \oplus y$	1	1	0	1	0	1	1	0

Word-RAM Instructions

	7	6	5	4	3	2	1	0
x	0	1	0	0	1	1	0	1
x « 2	0	0	1	1	0	1	0	0
x » 2	0	0	0	1	0	0	1	1
~x	1	0	1	1	0	0	1	0

LSB(x) 0

MSB(x) 6

Word-RAM and Constraint Solvers

- Constraint solvers already use the advantages of a Word-RAM to encode the variable domains.
- A bitset can represent a set of integers and therefore a domain.

dom(X) = {0, 2, 3, 6}

7	6	5	4	3	2	1	0
0	1	0	0	1	1	0	1

- There are two advantages with this encoding:
  - When backtracking, the solver restores 64 values in time O(1).
  - The encoding is compact.

Set Operations

Operator	Equivalence

Set Operations

Operator	Equivalence
$A \mathbin{\&} B$	$A \cap B$

Set Operations

Operator	Equivalence
$A \,\&\, B$	$A \cap B$
$A \mid B$	$A \cup B$

Set Operations

Operator	Equivalence
$A \mathbin{\&} B$	$A \cap B$
$A \mid B$	$A \cup B$
$A \ll k$	$\{a + k \mid a \in A, a + k < w\}$

Set Operations

A = {0, 2, 3, 6}

7	6	5	4	3	2	1	0
0	1	0	0	1	1	0	1

A « 2 = {2, 4, 5}

7	6	5	4	3	2	1	0
0	0	1	1	0	1	0	0

$A \mid B$	$A \cup B$
$A \ll k$	$\{a + k \mid a \in A, a + k < w\}$

Set Operations

Operator	Equivalence
$A \mathbin{\&} B$	$A \cap B$
$A \mid B$	$A \cup B$
$A \ll k$	$\{a + k \mid a \in A, a + k < w\}$

Set Operations

Operator	Equivalence
$A \mathbin{\&} B$	$A \cap B$
$A \mid B$	$A \cup B$
$A \ll k$	$\{a + k \mid a \in A, a + k < w\}$
$A \gg k$	$\{a - k \mid a \in A, a \geq k\}$

Set Operations

A = {0, 2, 3, 6}

7	6	5	4	3	2	1	0
0	1	0	0	1	1	0	1

A » 2 = {0, 1, 4}

7	6	5	4	3	2	1	0
0	0	0	1	0	0	1	1

$A \gg k$	$\{a - k \mid a \in A, a \geq k\}$

Set Operations

Operator	Equivalence
$A \mathbin{\&} B$	$A \cap B$
$A \mid B$	$A \cup B$
$A \ll k$	$\{a + k \mid a \in A, a + k < w\}$
$A \gg k$	$\{a - k \mid a \in A, a \geq k\}$

Set Operations

Operator	Equivalence
$A \mathbin{\&} B$	$A \cap B$
$A \mid B$	$A \cup B$
$A \ll k$	$\{a + k \mid a \in A, a + k < w\}$
$A \gg k$	$\{a - k \mid a \in A, a \geq k\}$
$\mathrm{LSB}(A)$	$\min(A)$

Set Operations

Operator	Equivalence
$A \,\&\, B$	$A \cap B$
$A \mid B$	$A \cup B$
$A \ll k$	$\{a + k \mid a \in A, a + k < w\}$
$A \gg k$	$\{a - k \mid a \in A, a \geq k\}$
$\mathrm{LSB}(A)$	$\min(A)$
$\mathrm{MSB}(A)$	$\max(A)$

Set Operations

Operator	Equivalence
$A \,\&\, B$	$A \cap B$
$A \mid B$	$A \cup B$
$A \ll k$	$\{a + k \mid a \in A, a + k < w\}$
$A \gg k$	$\{a - k \mid a \in A, a \geq k\}$
$\mathrm{LSB}(A)$	$\min(A)$
$\mathrm{MSB}(A)$	$\max(A)$
$\sim A$	$\overline{A}$

The SUM constraint

- Which values in these domains do not satisfy the constraint $A + B = C$?

$$\mathrm{dom}(A) = \{1, 5\}$$

$$\mathrm{dom}(B) = \{1, 3, 5\}$$

$$\mathrm{dom}(C) = \{2, 3, 6\}$$

- Assuming the cardinality of these three domains is *n*, the best known algorithm to solve this problem runs in time $O(n^2)$ on a RAM.
- Can we do better with a Word-RAM?

Filtering Algorithm

$A + B = C$

$dom(A) = \{1, 5\}$, $dom(B) = \{1, 3, 5\}$, $dom(C) = \{2, 3, 6\}$

Filtering Algorithm

A + B = C

dom(A) = {1, 5}, dom(B) = {1, 3, 5}, dom(C) = {2, 3, 6}

$$\mathrm{dom}(C) \leftarrow \mathrm{dom}(C) \cap \{a + b \mid a \in \mathrm{dom}(A), b \in \mathrm{dom}(B)\}$$

Filtering Algorithm

A + B = C

dom(A) = {1, 5}, dom(B) = {1, 3, 5}, dom(C) = {2, 3, 6}

$$\mathrm{dom}(C) \leftarrow \mathrm{dom}(C) \cap \{a + b \mid a \in \mathrm{dom}(A), b \in \mathrm{dom}(B)\}$$

$$\mathrm{dom}(C) \leftarrow \mathrm{dom}(C) \cap \bigcup_{a \in \mathrm{dom}(A)} \{a + b \mid b \in \mathrm{dom}(B)\}$$

Filtering Algorithm

A + B = C

dom(A) = {1, 5}, dom(B) = {1, 3, 5}, dom(C) = {2, 3, 6}

$$\mathrm{dom}(C) \leftarrow \mathrm{dom}(C) \cap \{a + b \mid a \in \mathrm{dom}(A), b \in \mathrm{dom}(B)\}$$

$$\mathrm{dom}(C) \leftarrow \mathrm{dom}(C) \cap \bigcup_{a \in \mathrm{dom}(A)} \{a + b \mid b \in \mathrm{dom}(B)\}$$

$$\mathrm{dom}(C) \leftarrow \mathrm{dom}(C) \cap \bigcup_{a \in \mathrm{dom}(A)} \mathrm{dom}(B) \ll a$$

Filtering Algorithm

A + B = C

dom(A) = {1, 5}, dom(B) = {1, 3, 5}, dom(C) = {2, 3, 6}

$$\mathrm{dom}(C) \leftarrow \mathrm{dom}(C) \cap \{a + b \mid a \in \mathrm{dom}(A), b \in \mathrm{dom}(B)\}$$

$$\mathrm{dom}(C) \leftarrow \mathrm{dom}(C) \cap \bigcup_{a \in \mathrm{dom}(A)} \{a + b \mid b \in \mathrm{dom}(B)\}$$

$$\mathrm{dom}(C) \leftarrow \mathrm{dom}(C) \cap \bigcup_{a \in \mathrm{dom}(A)} \mathrm{dom}(B) \ll a$$

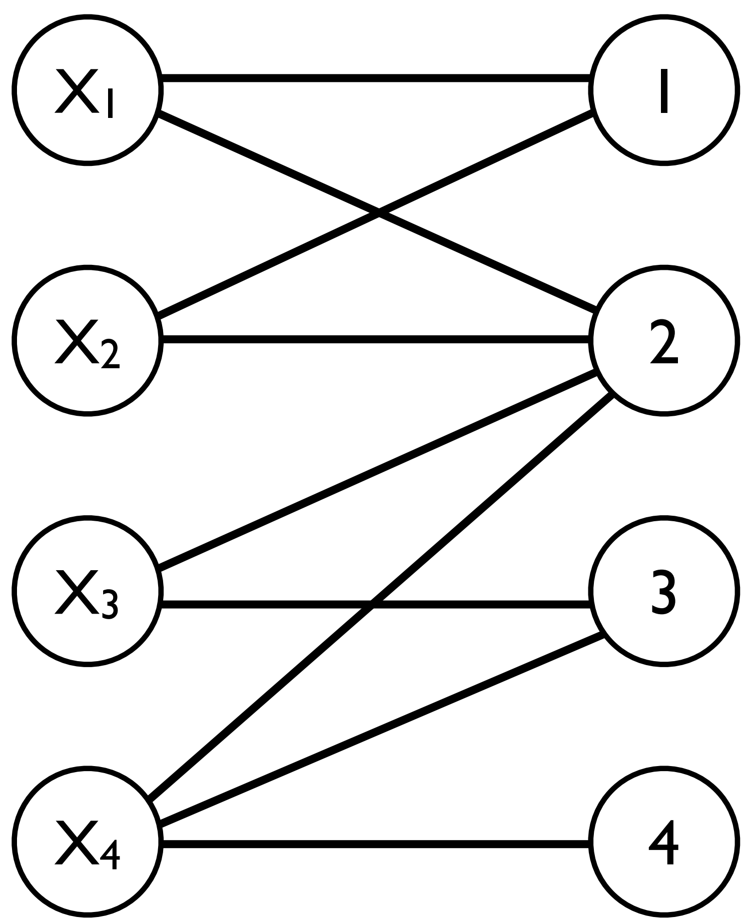
$$\mathrm{dom}(C) \leftarrow \mathrm{dom}(C) \& \left(\mathrm{dom}(B) \ll a_1 \mid \ldots \mid \mathrm{dom}(B) \ll a_{|\,\mathrm{dom}(A)|}\right)$$

Analysis

- Suppose all domains are contained in the interval [0, n).
- Suppose that a single word is sufficient to contain all domains
  - We have at most n sets to compute and to unite.
  - We have one intersection to compute.
  - Running time complexity: O(n)
- If domains cannot be encoded in a single word (n>w)
  - Each operation has a running time of $O\left(\frac{n}{w}\right)$.
  - Total running time: $O\left(\frac{n^2}{w}\right)$.

All-Different

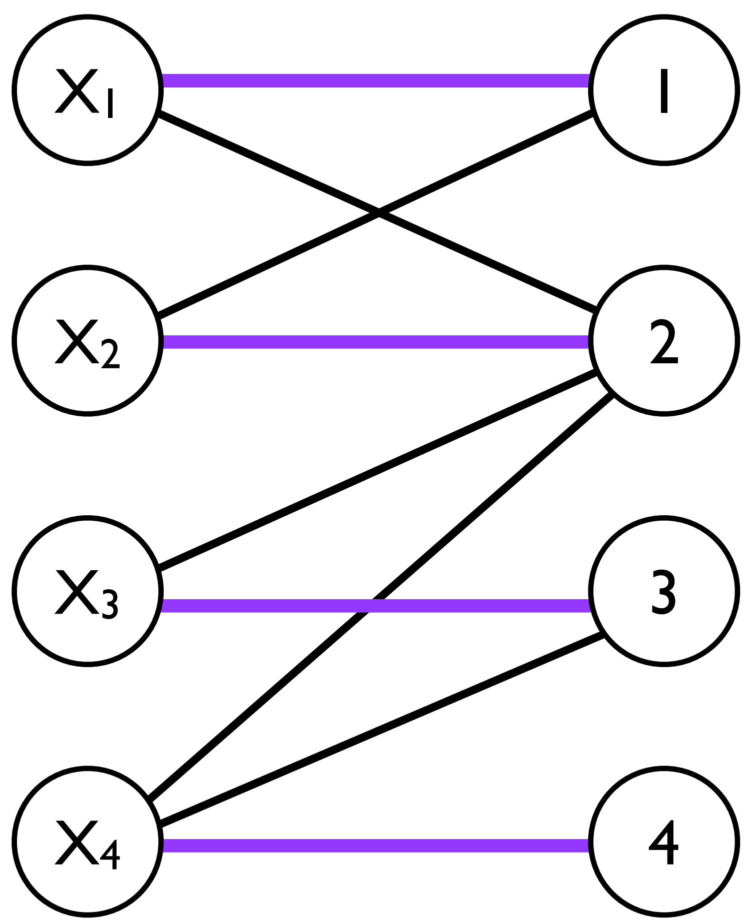
- The constraint All-Different(X_1, ..., X_n) is satisfied when all variables have distinct values.
- This constraint is satisfied when there exists a matching of cardinality *n* in the *value graph*.



$\mathrm{dom}(X_1) = \{1, 2\}$
$\mathrm{dom}(X_2) = \{1, 2\}$
$\mathrm{dom}(X_3) = \{2, 3\}$
$\mathrm{dom}(X_4) = \{2, 3, 4\}$

All-Different

- The constraint All-Different($X_1, ..., X_n$) is satisfied when all variables have distinct values.
- This constraint is satisfied when there exists a matching of cardinality *n* in the *value graph*.



$\text{dom}(X_1) = \{1, 2\}$
$\text{dom}(X_2) = \{1, 2\}$
$\text{dom}(X_3) = \{2, 3\}$
$\text{dom}(X_4) = \{2, 3, 4\}$

Graph Traversal

- Régin's filtering algorithm for the All-Different constraint requires to compute matching in the value graph and find the strongly connected components of the residual graph.
- These operations require to perform multiple traversals of a graph.
  - Depth-first search.
  - Breadth-first search

Depth-First Search

```
Visit(Graph, s)
  Mark the node s as visited
  For all v in Neighbors(s)
    If v has not been visited
      Visit(Graph, v)
```

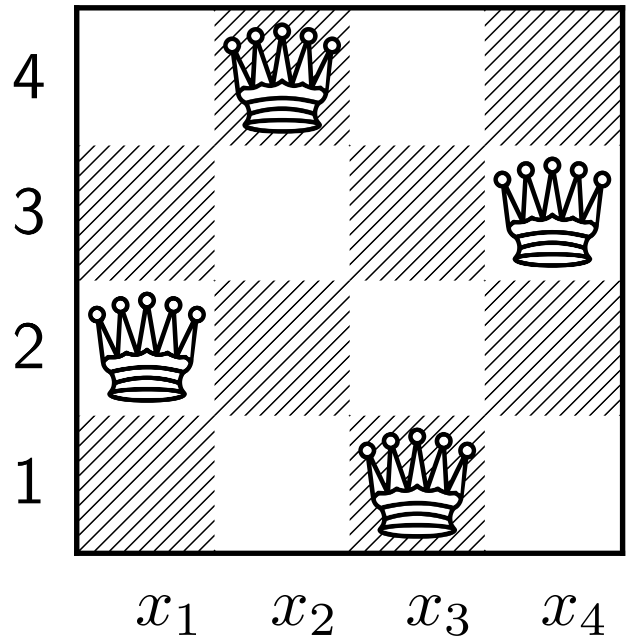
- Running time complexity on a RAM: $O(n + m)$
- Running time complexity on a Word-RAM: $O(n^2/w)$

Depth-First Search

```
Visit(Graph, s)
  Mark the node s as visited
  For all v in Neighbors(s)
    If v has not been visited
      Visit(Graph, v)

VisitWordRAM(Graph, s)
  V = V | (1 « s )
  While Neighbors(s) & ~V != 0
      Visit(Graph, LSB(Neighbors(s) & ~V))
```

N-Queen Problem



$$A_i = X_i + i$$

$$B_i = X_i - i$$

$$\text{All-Different}(X_1, \ldots, X_n)$$

$$\text{All-Different}(A_1, \ldots, A_n)$$

$$\text{All-Different}(B_1, \ldots, B_n)$$

n-queen

		All-Different		All-Different$_{WordRam}$	
n	bt	SUM$_{Table}$	SUM$_{WordRam}$	SUM$_{Table}$	SUM$_{WordRam}$
9	208	14	11	11	8
10	686	48	38	37	26
11	2940	210	163	157	112
12	13450	972	759	737	526
13	65677	4827	3782	3657	2610
14	344179	25842	20199	19464	13798
15	1948481	149567	116567	111822	80002

Times are in milliseconds

Magic Squares

2	7	6
9	5	1
4	3	8

Magic Square					
		ALL-DIFFERENT		ALL-DIFFERENT$_{WordRam}$	
n	bt	SUM$_{BruteForce}$	SUM$_{WordRam}$	SUM$_{BruteForce}$	SUM$_{WordRam}$
5	782	613	89	603	61
6	1535	2953	330	2931	238
7	2584	10654	1003	10551	748
8	4336	36301	3501	36220	2844
9	8211	119710	11228	117856	8849
10	23902	596705	46818	587675	37781
11	41857	-	109521	-	90062

Times are in milliseconds

Conclusion

- Using a different theoretical machine leads to a different running time analysis.
- The way one analyses an algorithm changes the way one designs that algorithm.
- The Word-RAM model can lead to substantial gains in execution time.