# Filtering Algorithms Based on the Word-RAM Model

Philippe Van Kessel and Claude-Guy Quimper
Université Laval, Québec, Canada
phil.vank@gmail.com, claude-guy.quimper@ift.ulaval.ca

## Abstract

The Word-RAM is a model of computation that takes into account the capacity of a computer to manipulate a word of bits $w$ with a single instruction. Many modern constraint solvers use a bitset data structure to encode the values contained in the variable domains. Using the algorithmic techniques developed for the Word-RAM, we propose new filtering algorithms that can prune $O(w)$ values from a domain in a single instruction. Experiments show that on a 64-bit processor, the new filtering algorithms that enforce domain consistency on the constraints $A+B=C$, $|A-B|=C$, and All-Different can offer a speed up of a factor 10.

## Computational Models

The running time efficiency of an algorithm is analyzed on a theoretical machine.

**RAM** (Random Access Machine): Manipulating one bit of memory requires one unit of time.

*Example*: Adding two $w$-bit integers require $O(w)$ time.

**Word-RAM**: Manipulating a word of $w$ bits requires one unit of time.

- If each bit represents a piece of data, then a 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 Operators

| | 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 ⊕ y | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 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 0

MSB 6

## Word-RAM in Constraint Solvers

A bitset can represent a set of integers and therefore a domain.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| dom(x) = {0, 2, 3, 6} | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |

This representation is more compact and allows to manipulate multiple values in a domain with a single operation.

## Set Operators

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

## Constraint A + B = C

Enforcing domain consistency on A + B = C removes the value 3 from the domains of B and C.

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

Best algorithm on a RAM: $O(n^2)$.

The operations to filter the domain of $C$ can be encoded as follows.

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

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

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

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

Using the Word-RAM operators, the constraint is filtered in time $O(n^2/w)$.

## Constraint |A - B| = C

This constraint can be rewritten as $A = B \pm C$ and be updated by the following rule.

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

$$\mathrm{dom}(A) \leftarrow \bigcup_{v \in \mathrm{dom}(C)} \mathrm{dom}(B) \ll v \mid \mathrm{dom}(B) \gg v$$

Time complexity: $O(n^2/w)$.

## Increasing and Bijective Functions

We consider the constraint $B = f(A)$ where $f$ is an increasing function. For instance: $B = 2A$.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| dom($A$) | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| dom($B$) | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

Each bit in dom($A$) is paired to a bit in dom($B$) to form a solution. These bits preserve their relative positions.

An operation of *compression* moves to the right the bits in a bitset B that are flagged in a mask $m$.

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| B | $b_7$ | $b_6$ | $b_5$ | $b_4$ | $b_3$ | $b_2$ | $b_1$ | $b_0$ |
| m | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| Compression(B, m) | 0 | 0 | 0 | $b_7$ | $b_5$ | $b_4$ | $b_2$ | $b_1$ |

The compression of one word is computed in O(log $w$) Word-RAM operators. The domain of $A$ is the compression of the domain of $B$. For domains of size $n$, one can filter the constraint in $O($ ($n$ log $n$) / $w$). A similar result applies for any bijective function $f$.

## All-Different

Régin's filtering algorithm for the All-Different constraint proceeds in two steps:

- It computes a matching in a bipartite graph;
- It computes the strongly connected components in the residual graph.

(Cheriyan & Mehlhorn 1996) show how to compute a matching and how to find strongly connected components in time $O(n^{2.5}/w)$ and $O(n^2/w)$. These algorithms rely on a Depth-First-Search in a graph.

## Depth-First-Search

**DFS designed for a RAM.**

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

**DFS designed for a Word-RAM**

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

## Experiments

*n*-queen

| $n$ | bt | ALL-DIFFERENT SUM$_{Table}$ | ALL-DIFFERENT SUM$_{WordRam}$ | ALL-DIFFERENT$_{WordRam}$ SUM$_{Table}$ | ALL-DIFFERENT$_{WordRam}$ 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 |

Magic Square

| $n$ | bt | ALL-DIFFERENT SUM$_{BruteForce}$ | ALL-DIFFERENT SUM$_{WordRam}$ | ALL-DIFFERENT$_{WordRam}$ SUM$_{BruteForce}$ | ALL-DIFFERENT$_{WordRam}$ 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 |

Golomb Ruler

| $n$ | bt | ALL-DIFFERENT SUM$_{BruteForce}$ | ALL-DIFFERENT SUM$_{WordRam}$ | ALL-DIFFERENT$_{WordRam}$ SUM$_{BruteForce}$ | ALL-DIFFERENT$_{WordRam}$ SUM$_{WordRam}$ |
|---|---|---|---|---|---|
| 6 | 39 | 8 | 4 | 8 | 2 |
| 7 | 207 | 44 | 22 | 37 | 16 |
| 8 | 1284 | 275 | 175 | 228 | 127 |
| 9 | 5980 | 1823 | 1286 | 1538 | 990 |
| 10 | 33318 | 12976 | 10380 | 11037 | 8484 |
| 11 | 553793 | 309715 | 275332 | 276345 | 241827 |

All-Interval

| $n$ | bt | ALL-DIFFERENT ABS$_{Table}$ | ALL-DIFFERENT ABS$_{WordRam}$ | ALL-DIFFERENT$_{WordRam}$ ABS$_{Table}$ | ALL-DIFFERENT$_{WordRam}$ ABS$_{WordRam}$ |
|---|---|---|---|---|---|
| 9 | 855 | 24 | 14 | 18 | 10 |
| 10 | 2903 | 93 | 56 | 69 | 37 |
| 11 | 10335 | 366 | 216 | 268 | 140 |
| 12 | 39270 | 1555 | 891 | 1131 | 578 |
| 13 | 155792 | 6823 | 3838 | 4857 | 2480 |
| 14 | 656435 | 31116 | 17351 | 22443 | 10967 |
| 15 | 2886750 | 146681 | 80817 | 105740 | 50960 |
| 16 | 13447418 | - | 402566 | 522795 | 251246 |

## Conclusion

The bitset encoding of the variable domains offers an opportunity to design new filtering algorithms adapted to a Word-RAM leading to substantial gains in performance.