Implementation and Evaluation of a Compact Table Propagator in Gecode

Linnea Ingmar
<linnea.ingmar.3244@student.uu.se>

The ASTRA Group
on Combinatorial Optimisation

Uppsala University

15th May 2017

Supervisor: Mats Carlsson (SICS)
Reviewer: Pierre Flener

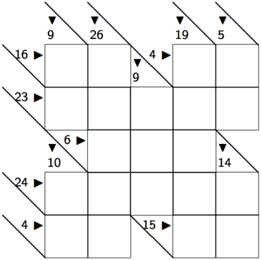
Outline

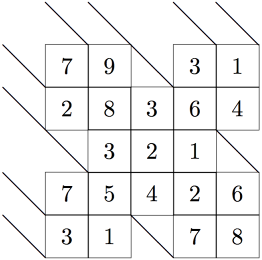
UPPSALA UNIVERSITET

Background
Constraint Programming
Propagators
Gecode
The Compact Table algorithm
Algorithms
Evaluation
Conclusions

Outline

Kakuro puzzle





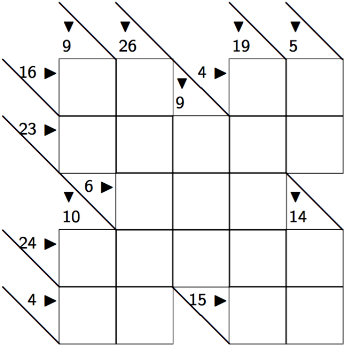
- Rows and columns of cells (*entries*) with prefilled numbers (*clues*).
- Fill in digits from 1 to 9 inclusive into the empty cells so that for each entry, the sum of the digits is equal to the clue of that entry, and so that each digit appears at most once in each entry.

Kakuro puzzle as a constraint problem (1)

Variables One per empty cell.

Domains $\{1...9\}$ for all variables.

Constraints (1) For each entry: variables are *distinct*, and the *sum* of them is equal to the clue.

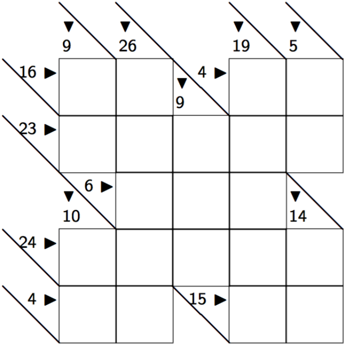


Kakuro puzzle as a constraint problem (2)

Variables One per empty cell.

Domains $\{1...9\}$ for all variables.

Constraints (2) For each entry: state the *possible combinations of values* that the variables can take.



For an entry of size 2 and clue 4: $\langle 1, 3\rangle$ and $\langle 3, 1\rangle$ are the only combinations.

Constraints and TABLE constraints (definitions)

Definition (Constraint)

A **constraint** on a finite sequene of n variables X is a relation, denoted $rel(c)$, that contains the set of allowed n-tuples for X. Each variable $x_i \in X$ has a corresponding *domain* D_i that is the set of possible values that x_i can take.

Definition (TABLE constraints)

A TABLE constraint explicitly lists $rel(c)$ as a sequence of n-tuples.

Example of a TABLE constraint.

Constraint Problems (definition)

Definition (Constraint problem)

A **constraint satisfaction problem (CSP)** is a triple

$$\langle V, D, C\rangle$$

where:

- $V = v_1, \ldots, v_n$ is a finite sequence of variables,
- $D = D_1, \ldots, D_n$ is a finite sequence of domains for the respective variable,
- $C = \{c_1, \ldots, c_m\}$ is a finite set of constraints, each on a subsequence of V.

Constraint Stores (definition)

Definition (Constraint store)

A **constraint store** s is a function mapping a finite set of variables $V = v_1, \ldots, v_n$ to a finite set of domains $D = D_1, \ldots, D_n$:

$$s : \mathit{variables} \mapsto \mathit{domains}$$

Furthermore, a store s...

- ...is a *failed store* iff $s(v_i) = \emptyset$ for some $v_i \in V$.
- ...is an *assignment store* iff $|s(v_i)| = 1$ for all $v_i \in V$.
- ...is a *solution store* for a constraint c iff s is an assignment store that constructs a solution to c.

Propagators (definition)

Definition (Propagators.)

A **propagator** p is a function mapping stores to stores:

$$p : store \mapsto store$$

Furthermore, a propagator p...

- ...is a decreasing function (i.e. can only remove values and not add new values).
- ...is a monotonic function (not a strict obligation).
- ...must faithfully implement its constraint.
- ...signals a *status message*.

Status messages

- FAIL. $p(s)$ is a failed store.
- SUBSUMED. p can never propagate any more values (all the following stores will be fixpoints).
- FIX. $p(s)$ is a fixpoint to p: $p(s) = p(p(s))$
- NOFIX. $p(s)$ is (possibly) not a fixpoint.

Obligations regarding status messages:

- Must signal SUBSUMED on a solution store s.
- Must signal FAIL on an assignment store s that is not a solution store.
- Must not claim FIX or SUBSUMED if it could propagate more.

Always safe to signal NOFIX on stores that are not assignment stores.

UPPSALA UNIVERSITET

Gecode

Gecode (Generic Constraint Development Environment) is...

- ...a constraint solver (a software that solves constraint problems).
- ...written in C++, modular, extensible, and has state-of-the-art performance.
- ...supports the programming of new propagators.
- ...developed at KTH, Sweden.

Two existing propagators for the TABLE constraint, and one for the related constraint REGULAR, that expresses *rel*(*c*) as a DFA.

A new propagation algorithm for the TABLEconstraint. Published in a 2016 paperNo attempt to implement it in Gecode (until now).

UPPSALA UNIVERSITET

Outline

PROCEDURE COMPACTTABLE(s : store) : $\langle StatusMsg, \text{store} \rangle$

```
if the propagator is being posted then
    s ← INITIALISECT(s, T0)
    if s = ∅ then
        return ⟨FAIL, ∅⟩
else
    foreach variable x ∈ s whose domain has changed since last time do
        UPDATETABLE(s, x)
        if validTuples.isEmpty() then
            return ⟨FAIL, ∅⟩
    if validTuples has changed since last time then
        s ← FILTERDOMAINS(s)
if there is at most one unassigned variable left then
    return ⟨SUBSUMED, s⟩
else
    return ⟨FIX, s⟩
```

Algorithm 1: Compact Table Propagator.

UPPSALA UNIVERSITET

Updating `validTuples`

PROCEDURE UPDATETABLE(s: store, x: variable)

`validTuples`.clearMask()
if Δ_x is available $\wedge$ $|\Delta_x| < |s(x)|$ **then**
  foreach $a \in \Delta_x$ **do**
    `validTuples`.addToMask(`supports`$[x, a]$)
  `validTuples`.reverseMask()
else
  foreach $a \in s(x)$ **do**
    `validTuples`.addToMask(`supports`$[x, a]$)
`validTuples`.intersectWithMask()

Filtering out values

PROCEDURE FILTERDOMAINS(s) : store

foreach $x \in s$ such that $|s(x)| > 1$ **do**
    foreach $a \in s(x)$ **do**
        $index \leftarrow \texttt{residues}[x, a]$
        if $\texttt{validTuples}[index]\ \&\ \texttt{supports}[x, a][index] = 0$ **then**
            $index \leftarrow \texttt{validTuples}.\mathrm{intersectIndex}(\texttt{supports}[x, a])$
            if $index \neq -1$ **then**
                $\texttt{residues}[x, a] \leftarrow index$
            else
                $s \leftarrow s[x \mapsto s(x) \setminus \{a\}]$
return s

Outline

Outline