Constraint Programming

Connections with HPC

Jean-Guillaume FAGES, PhD
COSLING CEO

COSLING

Outline

1. What is Constraint-Programming?
2. How does it work?
3. How does HPC contributes to Constraint-Programming?

What is Constraint-Programming?

A Paradigm

COSLING

Constraint-Programming

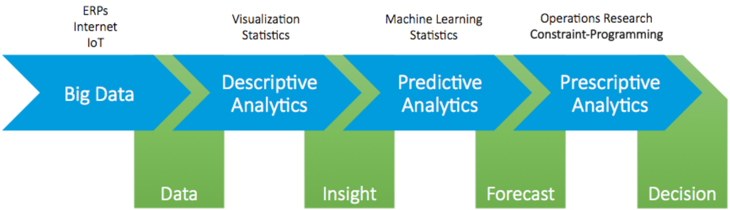
A programming paradigm between AI and OR to solve constrained decision problems in a generic way.

A Paradigm

COSLING

Constraint-Programming

A programming paradigm between AI and OR to solve constrained decision problems in a generic way.



Problem oriented paradigm

COSLING

Focus

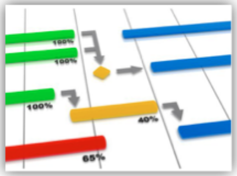
- It is not about Data (input)
- It is about solving problems
- Models are problem-specific

$\rightarrow$ CP is a generic toolbox to design algorithms

Applications

Applications

Applications



Applications

COSLING

How does it work?

Definition

Constraint Satisfaction Problem (CSP)

Defined by

- a set of variables $X = \{x_1, .., x_n\}$ (unknowns)
- a set of domains $D = \{d_1, .., d_n\}$ (possible values)
- a set of constraints $C = \{c_1, .., c_m\}$ (restrictions)

Definition

Constraint Satisfaction Problem (CSP)

Defined by

- a set of variables $X = \{x_1, .., x_n\}$ (unknowns)
- a set of domains $D = \{d_1, .., d_n\}$ (possible values)
- a set of constraints $C = \{c_1, .., c_m\}$ (restrictions)

Variable	Domain	Example
Binary	two integers	$\{0, 1\}$
Integer (bounded)	integer interval	$[\![3, 11]\!]$
Integer (enum)	integer set	$\{-2, ..., 3, 12\}$
Set	(int) set interval	$[\{2\}, \{0, 2, 3, 5\}]$
Graph	graph interval	
Real	real interval	$[3.14, 12.7]$

Definition

COSLING

Constraint Satisfaction Problem (CSP)

Defined by

- a set of variables $X = \{x_1, .., x_n\}$ (unknowns)
- a set of domains $D = \{d_1, .., d_n\}$ (possible values)
- a set of constraints $C = \{c_1, .., c_m\}$ (restrictions)

Constraints	Examples
Arithmetic	$+, -, \times, \div, =, >, \leq, \neq$
Logical	$\vee, \wedge, \Rightarrow, \Leftrightarrow$
Set	$\cap, \cup, \subseteq, card, partition$
Global	$AllDifferent, Cumulative, Regular, Circuit...$
Graph	$Degrees, NCliques, NSCC, Tree$

No restrictions!

Example: TSP

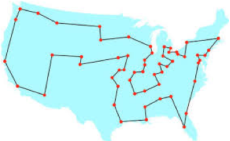
COSLING

```java
// --- Circuit: successors[i] = k <=> once at client i, travel to client k
IntVar[] successors = model.intVarArray("succ",n, 0, n - 1, false);
model.circuit(successors).post();

// --- Unit travel cost
IntVar[] travelDistances = model.intVarArray(n, min, max, true);
for(int i=0; i<n ; i++){
    model.element(travelDistances[i], distances[i], successors[i]).post();
}

// --- Objective function
IntVar totalDistance = model.intVar("total distance",min*n, max*n, true);
model.sum(travelDistances, "=", totalDistance).post();
model.setObjective(Model.MINIMIZE, totalDistance);

// --- Solving
Solver s = model.getSolver();
while (s.solve()){
    System.out.println(totalDistance);
}
```



The Holy Grail

Constraint Programming represents one of the closest approaches computer science has yet made to the Holy Grail of programming: the user states the problem, the computer solves it. [Eugene Freuder]

Main algorithm

COSLING

Solving loop

- **Filtering**
  - Search space reduction (inconsistent values removal)
  - Propagating deductions until **fixpoint**
- **Search**
  - Search space exploration (hypothesis)
  - Applying branching **heuristics** to compute **decisions**

Until all variables are fixed to a value (solution)

Main algorithm

Solving loop

- **Filtering**
  - Search space reduction (inconsistent values removal)
  - Propagating deductions until **fixpoint**
- **Search**
  - Search space exploration (hypothesis)
  - Applying branching **heuristics** to compute **decisions**

Until all variables are fixed to a value (solution)

Optimization

- Compute solution
- Post $z < c$ (cost of previous solution)

Until no better solution exists

Filtering

COSLING

From constraints to filtering

Every constraint comes with one or many **filtering algorithms** to remove inconsistent values (domain modifications)

Filtering

COSLING

From constraints to filtering

Every constraint comes with one or many **filtering algorithms** to remove inconsistent values (domain modifications)

Example

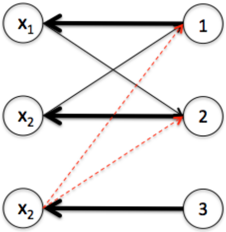
- $x = \{0, 1, 2, 3, 5, 6\}$ and $y = \{2, 3, 5, 6, 10\}$
- constraint: $x > y$
- Filtering $x \rightarrow y$: $y = \{2, 3, 5, \cancel{6}, \cancel{10}\}$
- Filtering $x \leftarrow y$: $x = \{\cancel{0}, \cancel{1}, \cancel{2}, 3, 5, 6\}$

Filtering

COSLING

From constraints to filtering

Every constraint comes with one or many **filtering algorithms** to remove inconsistent values (domain modifications)



Example: *AllDifferent*

Let $X = \{x_1, x_2, x_3\}$
with $x_1 \in \{1, 2\}$, $x_2 \in \{1, 2\}$ and $x_3 \in \{1, 2, 3\}$
such that $x_1 \neq x_2$, $x_1 \neq x_3$, $x_2 \neq x_3$

$x_1 \in \{1, 2\}$, $x_2 \in \{1, 2\}$ and $x_3 \in \{\cancel{1}, \cancel{2}, 3\}$

Filtering

COSLING

From constraints to filtering

Every constraint comes with one or many **filtering algorithms** to remove inconsistent values (domain modifications)

Scope

- Each filtering algorithm is **local** to a constraint
- Filtering algorithms are called **incrementally** until reaching a fix point

Search

COSLING

Search

Heuristic domain modification to drive the solving process

- Triggers propagation and filtering
- Guides how to reach a solution

Binary branching

At each node a **decision** (var, op, val) is taken

- Left child: apply decision (guess)
- Right child: negate decision (deduction)

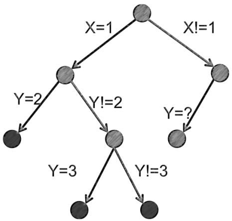
Search

COSLING

Search heuristics

Which decision (var, op, val) to generate?

- Huge impact on results

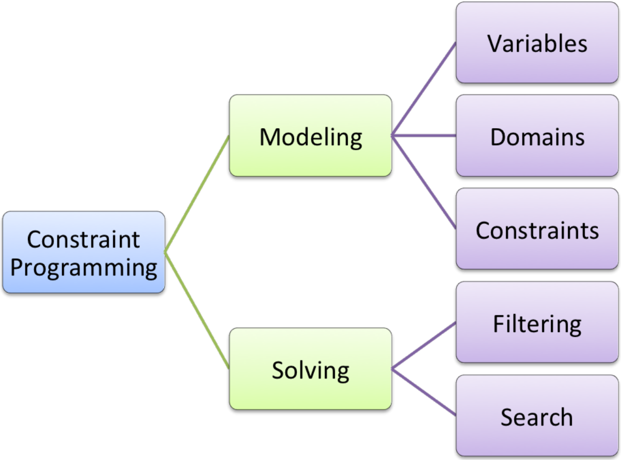


Many strategies

- Black-box v.s. User-defined strategy
- Best-First v.s. Fail-First principle

Summary

COSLING



How does HPC contributes to Constraint-Programming?

Parallelism

COSLING

Parallelism

Can we achieve significant gains using parallel computing?

Parallelism

COSLING

Parallelism

Can we achieve significant gains using parallel computing?

Filtering parallelization

Filtering algorithms could be called in parallel, but:

- Domain cloning overhead
- Unit propagation is (usually) fast
- Propagation is an iterative process

$\Rightarrow$ Not used in practice

Parallelism

COSLING

Parallelism

Can we achieve significant gains using parallel computing?

Search parallelization

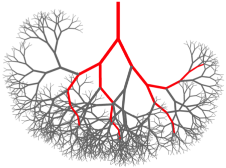
- Solver portfolios
- Distributed search
  - Work stealing
  - Embarrassingly Parallel Search

Portfolio approaches

COSLING

Portfolio

- Several instances of **same model** but **Different search strategies** run in parallel
- Satisfaction: no communication at all (luck)
- Optimization: Bound sharing (collaboration)

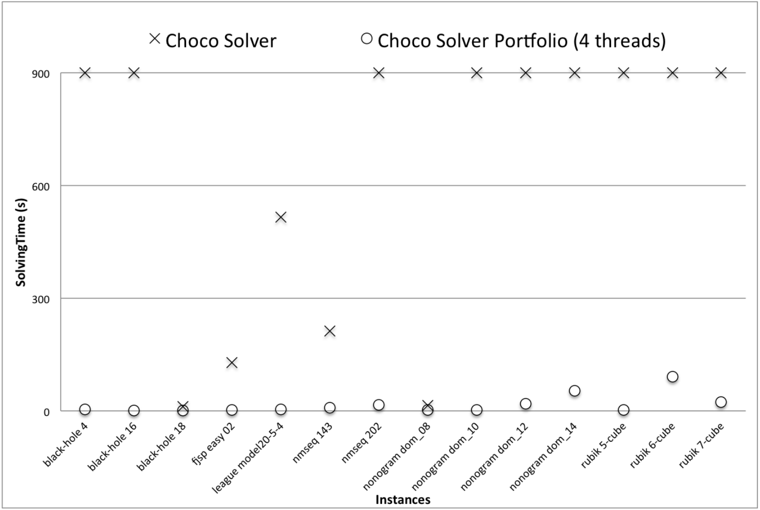


Many strategies

- Exponential speedup
- State-of-the-art for few cores

Portfolio approaches

COSLING

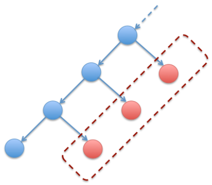


Distributed DFS

COSLING

Work stealing [Jaffar et. al.]

- Master records solutions & bounds
- Workers (dynamical)
  - Pop or steal node
  - Explore node
  - Push right child in stack



Issue

- Communication cost
- Hard to find a balanced distribution
- Hard to scale beyond 100 cores

Distributed DFS

COSLING

Embarrassingly Parallel Search [Regin et. al.]

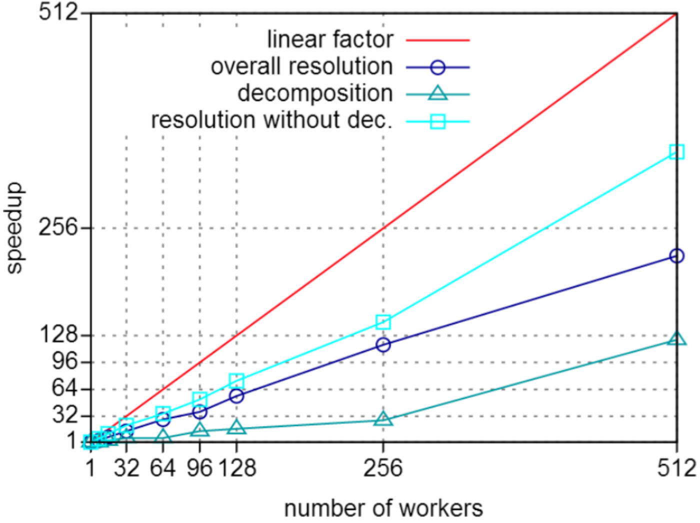
- **Static** decomposition into **consistent** subproblems
- Based on depth-bounded DFS
- Communicate with master only when subproblem is solved
- ≈ 40 subproblems per workers $\rightarrow$ balanced workload

Issue

- Almost no communication cost but decomposition cost
- State of the art over large number of cores

EPS speedup

COSLING



Distributed DFS

COSLING

Insight

- Independent subproblems are efficiently solved in parallel
- Automated decomposition may be costly/intricate

Distributed DFS

COSLING

Insight

- Independent subproblems are efficiently solved in parallel
- Automated decomposition may be costly/intricate

Good news!

It is often possible to decompose a problem

- Stochastic optimization based on different scenarios
- Optimization of different orders (logistics)
- Routing optimization of different areas (transportation)
- Frequent user request presolving (web-app)

Conclusion

COSLING

HPC

- Simple and affordable
- Pushes forward the limits of Constraint Programming
- Opens real-time application opportunities

Thank you for your attention

COSLING

Contact

- Jean-Guillaume FAGES, PhD
- jg.fages@cosling.com
- www.cosling.com
- +33683311966

References

COSLING

Main references

- Handbook of Constraint Programming
- J.-C. Regin, A Filtering Algorithm for Constraints of Difference in CSPs
- Global Constraint Catalogue: http://sofdem.github.io/gccat/
- A. Malapert, J-C. Regin, M. Rezgui: Embarrassingly Parallel Search in Constraint Programming

Journals & conferences

- Artificial Intelligence, Constraints
- CP, CPAIOR, AAAI, ECAI, JFPC