Octagon Abstract Domain

Session 6

Thibault Falque

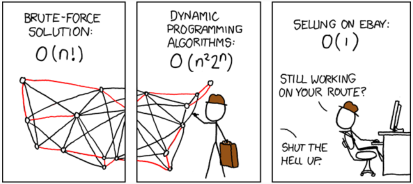
Abstract Interpretation Workshop – 20th June 2024

University of Luxembourg

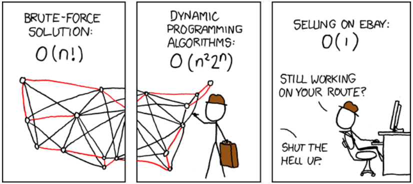
Contents

Introduction

Abstract constraint programming



Abstract constraint programming



Why?

- *A framework for combining constraint solvers*
- *Constraint solving on GPUs*

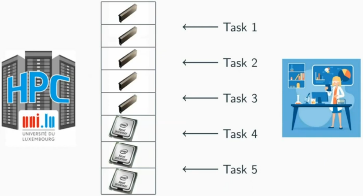
Constraint programming

- Constraint programming: we only specify what should be the solution using relations on variables (declarative programming).

Constraint programming

- Constraint programming: we only specify what should be the solution using relations on variables (declarative programming).
- But we do not program how to compute the solution.

An exemple of constraint problem



- Constraint problem: Tasks have a duration, use resources (#CPU/#GPU), and have precedence relations.
- Goal: Find a minimal schedule of the tasks on the HPC.

Scheduling problem RCPSP

NP-complete optimisation problem:

- T is a set of tasks, $d_i \in \mathbb{N}$ the duration of task i.
- P are the precedences among tasks: $i \ll j \in P$ if i must terminate before j starts.
- R is a set of resources where $k \in R$ has a capacity $c_k \in \mathbb{N}$.
- Each task i uses a quantity $r_{k,i}$ of resources k.

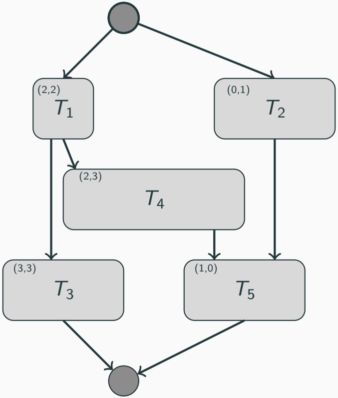
Scheduling problem RCPSP

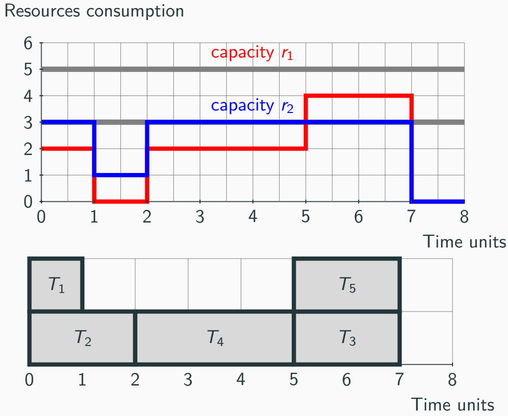
NP-complete optimisation problem:

- T is a set of tasks, $d_i \in \mathbb{N}$ the duration of task i.
- P are the precedences among tasks: $i \ll j \in P$ if i must terminate before j starts.
- R is a set of resources where $k \in R$ has a capacity $c_k \in \mathbb{N}$.
- Each task i uses a quantity $r_{k,i}$ of resources k.

Goal: find a (minimal) planning of tasks T that satisfies precedences in P without exceeding the capacity of available resources.

Example with 5 tasks and 2 resources





CSP & COP

Constraint satisfaction problem

A `constraint satisfaction problem` is a tuple $P = \langle X, D, C \rangle$ where:

- a *finite set of variables*, denoted by X
- $D_i \in D$ the set of values taken by each variable $x_i \in X$
- a *finite set of constraints*, denoted by C, each covering a sub-set of X such as $\forall c \in, \texttt{scope}(c) \subseteq X$.

CSP & COP

Constraint satisfaction problem

A `constraint satisfaction problem` is a tuple $P = \langle X, D, C \rangle$ where:

- a *finite set of variables*, denoted by X
- $D_i \in D$ the set of values taken by each variable $x_i \in X$
- a *finite set of constraints*, denoted by C, each covering a sub-set of X such as $\forall c \in, \texttt{scope}(c) \subseteq X$.

Constraint satisfaction problem

A `constraint optimization problem` is a tuple $P = \langle X, D, C, O \rangle$ where:

- a *finite set of variables*, denoted by X
- $D_i \in D$ the set of values taken by each variable $x_i \in X$
- a *finite set of constraints*, denoted by C, each covering a sub-set of X such as $\forall c \in, \texttt{scope}(c) \subseteq X$.
- an *objective function* $\mathcal{O} = \texttt{obj}(X)$ to be maximized or minimized

Constraints model [Schutt et al.]

- Variables : $s_i \in \{0..h-1\}$ is the starting time of task i.
- Constraints :

$$\forall (i \ll j) \in P,\ s_i + d_i \leq s_j \tag{1}$$

$$\begin{aligned} &\forall j \in [1..n],\ \forall i \in [1..n] \setminus \{j\}, \\ &\qquad b_{i,j} \Leftrightarrow (s_i \leq s_j \wedge s_j < s_i + d_i) \end{aligned} \tag{2}$$

$$\forall j \in [1..n],\ r_{k,j} + \Big(\sum_{i \in [1..n] \setminus \{j\}} r_{k,i} * b_{i,j}\Big) \leq c_k \tag{3}$$

1. Temporal constraints (eq. 1)
2. Resources constraints (eq. 2 and 3): *tasks decomposition* of global constraint `cumulative`.

Abstract domain

Abstract domain

An abstract domain $\langle Abs, \leq, \sqcup, \top, \gamma, [\![.]\!], refine, split \rangle$ is a lattice such that:

- Abs is a set of elements representable in a machine.
- $\leq$ is a partial order.
- $\sqcup$ performs the *join* of two elements ("union of information").
- $\top$ is the largest element ("initial state").
- $\gamma : A \rightarrow D^\flat$ is a monotone concretization function.
- $\texttt{state} : Abs \rightarrow K$ gives the state of an element ($K = \{$ true, false, unknown $\}$).
- $[\![.]\!] : \Phi \rightarrow Abs$ is a partial interpretation function turning a constraint into an element of the abstract domain.
- $refine : Abs \rightarrow Abs$ is an extensive function, *e.g.*, $a \leq refine(a)$, refining an abstract element ("gain information").
- $split : Abs \rightarrow \mathcal{P}(Abs)$ is an extensive function dividing an abstract element into a set of sub-elements.
- $\models$: $Abs \times \Phi$: $a \models \varphi$ holds whenever $\gamma(a) \subseteq [\![\varphi]\!]^\flat$ the deduction relation, called the 'entailment'.

Interval

Interval

An interval is a pair $(l, u) \in \mathbb{Z}^2$ of the lower and upper bounds, written $[l, u]$.

Lattice of intervals

The lattice of interval $\langle \mathcal{I}, \sqsubseteq, \sqcup, \sqcap, \bot, [-\infty, \infty] \rangle$ is defined as:

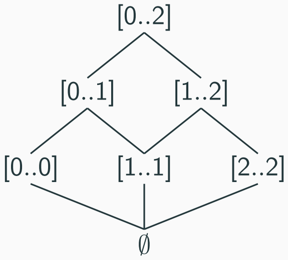
$$\mathcal{I} \triangleq \{[a, b] \mid a \in \mathbb{Z} \cup \{-\infty\}, b \in \mathbb{Z} \cup \{\infty\}, a \sqsubseteq b\} \cup \{\bot\}$$

with the following operations:

- $[a, b] \sqsubseteq [c, d] \Leftrightarrow a \geq c \wedge b \leq d$.
- $[a, b] \sqcup [c, d] \triangleq [\min(a, c), \max(b, d)]$.
- $[a, b] \sqcap [c, d] \triangleq [\max(a, c), \min(b, d)]$.

Example of lattice of intervals

For the set $\{0, 1, 2\}$



Box

Box domain

- Let $\mathcal{I}$ be the lattice of integer intervals, and V a set of variables.
- Then $\text{Box} = [V \nrightarrow \mathcal{I}]$ is the abstract domain of box.

It treats constraints of the form

$$x \leq d \quad x \geq d$$

where $d \in \mathbb{Z}$ is a constant.

Octagon abstract domain

Octagonal constraint

Octagonal constraint

We call `octagagonal constraint` any constraint of the form $\pm x_i - \pm x_j \leq c$ with c is a constant from $\mathbb{Z}$, $\mathbb{Q}$ or $\mathbb{R}$.

We call `octagon` the set of points satisfying a conjuction of octagonal constraints.

Remark

The name `octagon` comes from the fact that, in two dimensions, our sets are `polyhedra` with at most eight sides.

Potential constraints

Potential constraint

We call `potential constraint` any constraint of the form $x_i - x_j \leq c$.

Potential constraints

Potential constraint

We call `potential constraint` any constraint of the form $x_i - x_j \leq c$.

Potential graphs

A conjunction of potential constraints can be represented as a directed graph $\mathcal{G}$ with nodes from $(x_0, \ldots, x_{n-1})$ and value in $\mathbb{Z}$, $\mathbb{Q}$ or $\mathbb{R}$.

For each ordered pair of variables $x_i, x_j \in \mathcal{V}^2$, there will be an arc from x_i to x_j with weight c if the constraint $x_i - x_j \leq c$ is in constraint conjuction.

Difference Bound Matrix

Difference bound matrices

An equivalent representation for potential constraint conjuction is by means of a `Difference Bound Matrix` (DBM).

A DBM m is a $n \times n$ square matrix where n is the number of variables.

The element at line i, column j where $1 \leq i \leq n$, $1 \leq j \leq n$, denoted by m_{ij}, equals to c if there is a constraint of the form $x_i - x_j \leq c$ in our constraint conjunction and $+\infty$ otherwise.

Difference Bound Matrix

Difference bound matrices

An equivalent representation for potential constraint conjuction is by means of a `Difference Bound Matrix` (DBM).

A DBM m is a $n \times n$ square matrix where n is the number of variables.

The element at line i, column j where $1 \leq i \leq n$, $1 \leq j \leq n$, denoted by m_{ij}, equals to c if there is a constraint of the form $x_i - x_j \leq c$ in our constraint conjunction and $+\infty$ otherwise.

Remark

A DBM m can be seen as the adjacency matrix of a potential graph.

Transformation of octagonal constraints

Transformation of octagonal constraints

From the set of variables $\mathcal{V} = (x_0, \ldots, x_{n-1})$ we derive the set $\mathcal{V}' = (x'_0, \ldots, x'_{2n})$.

Each variable $x_i \in \mathcal{V}$ has both a `positive form` x'_{2i}, and a `negative form` x'_{2i+1}.

Transformation of octagonal constraints

Transformation of octagonal constraints

From the set of variables $\mathcal{V} = (x_0, \ldots, x_{n-1})$ we derive the set $\mathcal{V}' = (x'_0, \ldots, x'_{2n})$.

Each variable $x_i \in \mathcal{V}$ has both a `positive form` x'_{2i}, and a `negative form` x'_{2i+1}.

We will encode `octagonal constraints` on $\mathcal{V}$ as `potential constraints` on $\mathcal{V}'$.

Transformation of octagonal constraints

Transformation of octagonal constraints

From the set of variables $\mathcal{V} = (x_0, \ldots, x_{n-1})$ we derive the set $\mathcal{V}' = (x'_0, \ldots, x'_{2n})$.

Each variable $x_i \in \mathcal{V}$ has both a `positive form` x'_{2i}, and a `negative form` x'_{2i+1}.

We will encode `octagonal constraints` *on* $\mathcal{V}$ *as* `potential constraints` *on* $\mathcal{V}'$.

$$
\begin{aligned}
x_i - x_j \leq d &\rightsquigarrow x'_{2i} - x'_{2j} \leq d \wedge x'_{2j+1} - x'_{2i+1} \leq d \\
x_i + x_j \leq d &\rightsquigarrow x'_{2i} - x'_{2j+1} \leq d \wedge x'_{2j} - x'_{2i+1} \leq d \\
-x_i - x_j \leq d &\rightsquigarrow x'_{2i+1} - x'_{2j} \leq d \wedge x'_{2j+1} - x'_{2i} \leq d \\
x_i \leq d &\rightsquigarrow x'_{2i} - x'_{2i+1} \leq 2d \\
-x_i \leq d &\rightsquigarrow x'_{2i+1} - x'_{2i} \leq 2d
\end{aligned}
$$

Transformation of octagonal constraints

- In a `potential constraint` x'_{2i} will represent x_i while x'_{2i+1} will represent $-x_i$.
- A conjuction of `octagonal constraints` on $\mathcal{V}$ can be represented as a DBM of dimension $2 \times n$.

Example of transformation of octagonal system to potential constraint system

$$x_0 \leq 3$$
$$x_1 \leq 2$$
$$x_0 + x_1 \leq 6$$
$$-x_0 - x_1 \leq 5$$
$$-x_0 \leq 3$$

Example of transformation of octagonal system to potential constraint system

$$x_0 \leq 3$$
$$x_1 \leq 2$$
$$x_0 + x_1 \leq 6$$
$$-x_0 - x_1 \leq 5$$
$$-x_0 \leq 3$$

How to translate this octagonal system and fill the DBM ?

Example of transformation of octagonal system to potential constraint system

$$x_0 \leq 3$$
$$x_1 \leq 2$$
$$x_0 + x_1 \leq 6$$
$$-x_0 - x_1 \leq 5$$
$$-x_0 \leq 3$$

	x'_0	x'_1	x'_2	x'_3
x'_0	∞	∞	∞	∞
x'_1	∞	∞	∞	∞
x'_2	∞	∞	∞	∞
x'_3	∞	∞	∞	∞

Example of transformation of octagonal system to potential constraint system

$x_0 \leq 3$

$x'_0 - x'_1 \leq 6$

	x'_0	x'_1	x'_2	x'_3
x'_0	∞	∞	∞	∞
x'_1	∞	∞	∞	∞
x'_2	∞	∞	∞	∞
x'_3	∞	∞	∞	∞

We apply $x_i \leq d \rightsquigarrow x'_{2i} - x'_{2i+1} \leq 2d$

Example of transformation of octagonal system to potential constraint system

$x_0 \leq 3$ $\qquad$ $x'_0 - x'_1 \leq 6$

	x'_0	x'_1	x'_2	x'_3
x'_0	∞	6	∞	∞
x'_1	∞	∞	∞	∞
x'_2	∞	∞	∞	∞
x'_3	∞	∞	∞	∞

We apply $x_i \leq d \rightsquigarrow x'_{2i} - x'_{2i+1} \leq 2d$

Example of transformation of octagonal system to potential constraint system

$x_0 \leq 3$	$x'_0 - x'_1 \leq 6$
$x_1 \leq 2$	$x'_2 - x'_3 \leq 4$

	x'_0	x'_1	x'_2	x'_3
x'_0	∞	6	∞	∞
x'_1	∞	∞	∞	∞
x'_2	∞	∞	∞	∞
x'_3	∞	∞	∞	∞

We apply $x_i \leq d \rightsquigarrow x'_{2i} - x'_{2i+1} \leq 2d$

Example of transformation of octagonal system to potential constraint system

$x_0 \leq 3$	$x'_0 - x'_1 \leq 6$
$x_1 \leq 2$	$x'_2 - x'_3 \leq 4$

	x'_0	x'_1	x'_2	x'_3
x'_0	∞	6	∞	∞
x'_1	∞	∞	∞	∞
x'_2	∞	∞	∞	4
x'_3	∞	∞	∞	∞

We apply $x_i \leq d \rightsquigarrow x'_{2i} - x'_{2i+1} \leq 2d$

Example of transformation of octagonal system to potential constraint system

$x_0 \leq 3$	$x'_0 - x'_1 \leq 6$
$x_1 \leq 2$	$x'_2 - x'_3 \leq 4$
$x_0 + x_1 \leq 6$	$x'_0 - x'_3 \leq 6$, $x'_2 - x'_1 \leq 6$

	x'_0	x'_1	x'_2	x'_3
x'_0	∞	6	∞	∞
x'_1	∞	∞	∞	∞
x'_2	∞	∞	∞	4
x'_3	∞	∞	∞	∞

We apply $x_i + x_j \leq d \rightsquigarrow x'_{2i} - x'_{2j+1} \leq d \wedge x'_{2j} - x'_{2i+1} \leq d$

Example of transformation of octagonal system to potential constraint system

$x_0 \leq 3$	$x'_0 - x'_1 \leq 6$
$x_1 \leq 2$	$x'_2 - x'_3 \leq 4$
$x_0 + x_1 \leq 6$	$x'_0 - x'_3 \leq 6$, $x'_2 - x'_1 \leq 6$

	x'_0	x'_1	x'_2	x'_3
x'_0	∞	6	∞	6
x'_1	∞	∞	∞	∞
x'_2	∞	6	∞	4
x'_3	∞	∞	∞	∞

We apply $x_i + x_j \leq d \rightsquigarrow x'_{2i} - x'_{2j+1} \leq d \wedge x'_{2j} - x'_{2i+1} \leq d$

Example of transformation of octagonal system to potential constraint system

$x_0 \leq 3$	$x'_0 - x'_1 \leq 6$
$x_1 \leq 2$	$x'_2 - x'_3 \leq 4$
$x_0 + x_1 \leq 6$	$x'_0 - x'_3 \leq 6,\ x'_2 - x'_1 \leq 6$
$-x_0 - x_1 \leq 5$	$x'_1 - x'_2 \leq 5,\ x'_3 - x'_0 \leq 5$

	x'_0	x'_1	x'_2	x'_3
x'_0	∞	6	∞	6
x'_1	∞	∞	∞	∞
x'_2	∞	6	∞	4
x'_3	∞	∞	∞	∞

We apply $-x_i - x_j \leq d \rightsquigarrow x'_{2i+1} - x'_{2j} \leq d \wedge x'_{2j+1} - x'_{2i} \leq d$

Example of transformation of octagonal system to potential constraint system

$x_0 \leq 3$	$x'_0 - x'_1 \leq 6$
$x_1 \leq 2$	$x'_2 - x'_3 \leq 4$
$x_0 + x_1 \leq 6$	$x'_0 - x'_3 \leq 6,\ x'_2 - x'_1 \leq 6$
$-x_0 - x_1 \leq 5$	$x'_1 - x'_2 \leq 5,\ x'_3 - x'_0 \leq 5$

	x'_0	x'_1	x'_2	x'_3
x'_0	∞	6	∞	6
x'_1	∞	∞	5	∞
x'_2	∞	6	∞	4
x'_3	5	∞	∞	∞

We apply $-x_i - x_j \leq d \rightsquigarrow x'_{2i+1} - x'_{2j} \leq d \wedge x'_{2j+1} - x'_{2i} \leq d$

Example of transformation of octagonal system to potential constraint system

$x_0 \leq 3$	$x'_0 - x'_1 \leq 6$
$x_1 \leq 2$	$x'_2 - x'_3 \leq 4$
$x_0 + x_1 \leq 6$	$x'_0 - x'_3 \leq 6,\ x'_2 - x'_1 \leq 6$
$-x_0 - x_1 \leq 5$	$x'_1 - x'_2 \leq 5,\ x'_3 - x'_0 \leq 5$
$-x_0 \leq 3$	$x'_1 - x'_0 \leq 6$

	x'_0	x'_1	x'_2	x'_3
x'_0	∞	6	∞	6
x'_1	∞	∞	5	∞
x'_2	∞	6	∞	4
x'_3	5	∞	∞	∞

We apply $-x_i \leq d \rightsquigarrow x'_{2i+1} - x'_{2i} \leq 2d$

Example of transformation of octagonal system to potential constraint system

$x_0 \leq 3$	$x'_0 - x'_1 \leq 6$
$x_1 \leq 2$	$x'_2 - x'_3 \leq 4$
$x_0 + x_1 \leq 6$	$x'_0 - x'_3 \leq 6,\ x'_2 - x'_1 \leq 6$
$-x_0 - x_1 \leq 5$	$x'_1 - x'_2 \leq 5,\ x'_3 - x'_0 \leq 5$
$-x_0 \leq 3$	$x'_1 - x'_0 \leq 6$

	x'_0	x'_1	x'_2	x'_3
x'_0	∞	6	∞	6
x'_1	6	∞	5	∞
x'_2	∞	6	∞	4
x'_3	5	∞	∞	∞

We apply $-x_i \leq d \rightsquigarrow x'_{2i+1} - x'_{2i} \leq 2d$

Example - Potential graph

What about the graph representation ?

$$x_0' - x_1' \leq 6$$
$$x_2' - x_3' \leq 4$$
$$x_0' - x_3' \leq 6,\ x_2' - x_1' \leq 6$$
$$x_1' - x_2' \leq 5,\ x_3' - x_0' \leq 5$$
$$x_1' - x_0' \leq 6$$

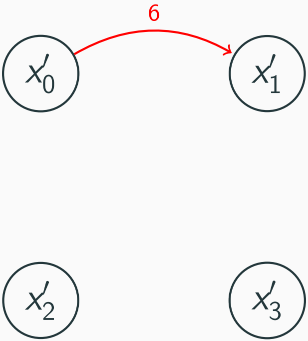
	x_0'	x_1'	x_2'	x_3'
x_0'	∞	6	∞	6
x_1'	6	∞	5	∞
x_2'	∞	6	∞	4
x_3'	5	∞	∞	∞

Example - Potential graph

What about the graph representation ?

$$x'_0 - x'_1 \leq 6$$
$$x'_2 - x'_3 \leq 4$$
$$x'_0 - x'_3 \leq 6,\ x'_2 - x'_1 \leq 6$$
$$x'_1 - x'_2 \leq 5,\ x'_3 - x'_0 \leq 5$$
$$x'_1 - x'_0 \leq 6$$

	x'_0	x'_1	x'_2	x'_3
x'_0	∞	6	∞	6
x'_1	6	∞	5	∞
x'_2	∞	6	∞	4
x'_3	5	∞	∞	∞

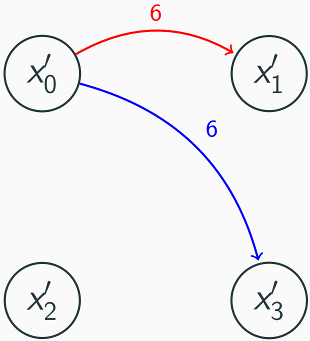


Example - Potential graph

What about the graph representation ?

$$x'_0 - x'_1 \leq 6$$
$$x'_2 - x'_3 \leq 4$$
$$x'_0 - x'_3 \leq 6,\ x'_2 - x'_1 \leq 6$$
$$x'_1 - x'_2 \leq 5,\ x'_3 - x'_0 \leq 5$$
$$x'_1 - x'_0 \leq 6$$

	x'_0	x'_1	x'_2	x'_3
x'_0	∞	6	∞	6
x'_1	6	∞	5	∞
x'_2	∞	6	∞	4
x'_3	5	∞	∞	∞

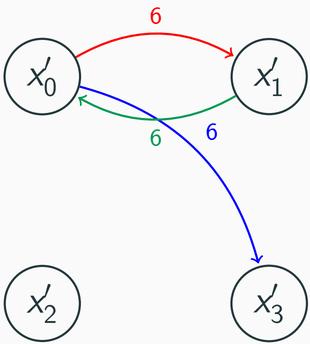


Example - Potential graph

What about the graph representation ?

$$x'_0 - x'_1 \leq 6$$
$$x'_2 - x'_3 \leq 4$$
$$x'_0 - x'_3 \leq 6,\ x'_2 - x'_1 \leq 6$$
$$x'_1 - x'_2 \leq 5,\ x'_3 - x'_0 \leq 5$$
$$x'_1 - x'_0 \leq 6$$

	x'_0	x'_1	x'_2	x'_3
x'_0	∞	6	∞	6
x'_1	6	∞	5	∞
x'_2	∞	6	∞	4
x'_3	5	∞	∞	∞

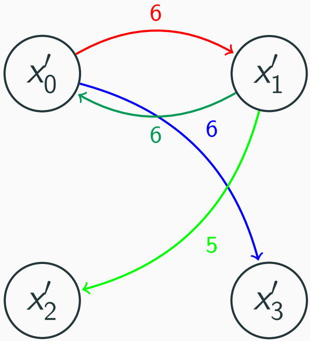


Example - Potential graph

What about the graph representation ?

$$x'_0 - x'_1 \leq 6$$
$$x'_2 - x'_3 \leq 4$$
$$x'_0 - x'_3 \leq 6, \; x'_2 - x'_1 \leq 6$$
$$x'_1 - x'_2 \leq 5, \; x'_3 - x'_0 \leq 5$$
$$x'_1 - x'_0 \leq 6$$

	x'_0	x'_1	x'_2	x'_3
x'_0	∞	6	∞	6
x'_1	6	∞	5	∞
x'_2	∞	6	∞	4
x'_3	5	∞	∞	∞

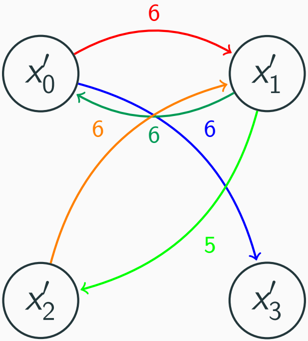


Example - Potential graph

What about the graph representation ?

$$x'_0 - x'_1 \leq 6$$
$$x'_2 - x'_3 \leq 4$$
$$x'_0 - x'_3 \leq 6,\; x'_2 - x'_1 \leq 6$$
$$x'_1 - x'_2 \leq 5,\; x'_3 - x'_0 \leq 5$$
$$x'_1 - x'_0 \leq 6$$

	x'_0	x'_1	x'_2	x'_3
x'_0	∞	6	∞	6
x'_1	6	∞	5	∞
x'_2	∞	6	∞	4
x'_3	5	∞	∞	∞

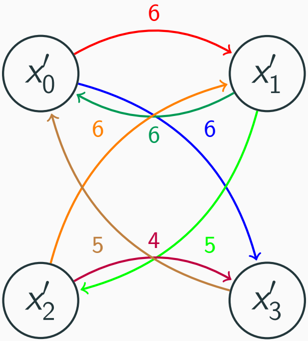


Example - Potential graph

What about the graph representation ?

$$x'_0 - x'_1 \leq 6$$
$$x'_2 - x'_3 \leq 4$$
$$x'_0 - x'_3 \leq 6,\ x'_2 - x'_1 \leq 6$$
$$x'_1 - x'_2 \leq 5,\ x'_3 - x'_0 \leq 5$$
$$x'_1 - x'_0 \leq 6$$

	x'_0	x'_1	x'_2	x'_3
x'_0	∞	6	∞	6
x'_1	6	∞	5	∞
x'_2	∞	6	∞	4
x'_3	5	∞	∞	∞



Some remarks about DBM

	x'_0	x'_1	x'_2	x'_3
x'_0	∞	6	∞	6
x'_1	6	∞	5	∞
x'_2	∞	6	∞	4
x'_3	5	∞	∞	∞

- $m_{0,3} = m_{2,1} = 6.$
- $x'_0 - x'_3 \leq 6, \quad x'_2 - x'_1 \leq 6$
- $x_0 + x_1 \leq 6, \quad x_1 + x_0 \leq 6$
- DBM operations should keep entries equal.

Coherence and Consistency

Coherence

A DBM $\mathbf{m}$ is coherent iff $\forall i.j.\mathbf{m}_{i,j} = \mathbf{m}_{\bar{\jmath},\bar{\imath}}$ where $\bar{\imath} = i + 1$ if i is even and $i - 1$ otherwise.

Bar operator

The bar operation can be realised without a branch using $\bar{\imath} = i \oplus 1$.

Coherence and Consistency

Coherence

A DBM **m** is coherent iff $\forall i.j.\mathbf{m}_{i,j} = \mathbf{m}_{\bar{\jmath},\bar{\imath}}$ where $\bar{\imath} = i + 1$ if i is even and $i - 1$ otherwise.

Bar operator

The bar operation can be realised without a branch using $\bar{\imath} = i \oplus 1$.

Consistency

A DBM **m** is consistent iff $\forall i.\mathbf{m}_{i,i} \geq 0$.

Negative cycle

Intuitively, consistency means that there is not `negative cycle` in the DBM, which corresponds to unsatisfiability.

Order

Partial order

Let m and m' two matrices of size N from two potential sets we can define the `order` operator, denoted $\leq$, as

$$m \leq m' \text{ iff } m_{i,j} \leq m'_{i,j} \quad \forall i, j \in N$$

Link with CP

The `order` allows for the removal of redundant constraints.

Join

Join $\sqcup$

Let m and m' two matrices of size N from two potential sets we can define the `join` operator as

$$m \sqcup m' = \left\{ \max\left(m_{i,j}, m'_{i,j}\right)^{i,j} \mid i, j \in N \right\}$$

Link with CP

$\sqcup$ can be seen as the `disjunction` of constraints of the form $x_0 + x_1 \leq d$.

Join - Example

Let m the matrix represented the constraint $x_0 + x_1 \leq 5$ and m' the matrix represented the constraint $x_0 + x_1 \leq 7$.

	x'_0	x'_1	x'_2	x'_3
x'_0	∞	∞	∞	5
x'_1	∞	∞	∞	∞
x'_2	∞	5	∞	∞
x'_3	∞	∞	∞	∞

$\sqcup$

	x'_0	x'_1	x'_2	x'_3
x'_0	∞	∞	∞	7
x'_1	∞	∞	∞	∞
x'_2	∞	7	∞	∞
x'_3	∞	∞	∞	∞

$=$

	x'_0	x'_1	x'_2	x'_3
x'_0	∞	∞	∞	7
x'_1	∞	∞	∞	∞
x'_2	∞	7	∞	∞
x'_3	∞	∞	∞	∞

Meet

Meet $\sqcap$

Let m and m' two matrices of size N from two potential sets we can define the `meet` operator as

$$m \sqcap m' = \left\{ \min \left(m_{i,j}, m'_{i,j} \right)^{i,j} \mid i, j \in N \right\}$$

Link with CP

$\sqcap$ can be seen as the `conjunction` of constraints of the form $x_0 + x_1 \leq d$.

Remark

The order $m \leq m'$ is equivalent to $m \sqcap m' = m$ and $m \leq m'$ is equivalent to $m \sqcup m' = m$

Meet - Example

Let m the matrix representing the constraint $x_0 + x_1 \leq 5$ and m' the matrix representing the constraint $x_0 + x_1 \leq 7$.

	x_0'	x_1'	x_2'	x_3'
x_0'	∞	∞	∞	5
x_1'	∞	∞	∞	∞
x_2'	∞	5	∞	∞
x_3'	∞	∞	∞	∞

$\sqcap$

	x_0'	x_1'	x_2'	x_3'
x_0'	∞	∞	∞	7
x_1'	∞	∞	∞	∞
x_2'	∞	7	∞	∞
x_3'	∞	∞	∞	∞

$=$

	x_0'	x_1'	x_2'	x_3'
x_0'	∞	∞	∞	5
x_1'	∞	∞	∞	∞
x_2'	∞	5	∞	∞
x_3'	∞	∞	∞	∞

Closure

Closure

A DBM m is `closed`, and denoted by m^* iff

- $\forall i.\mathbf{m}_{i,i} = 0$
- $\forall i, j, k.\mathbf{m}_{i,j} \leq \mathbf{m}_{i,k} + \mathbf{m}_{k,j}$

Closure

Closure

A DBM m is `closed`, and denoted by m^* iff

- $\forall i.\mathbf{m}_{i,i} = 0$
- $\forall i, j, k.\mathbf{m}_{i,j} \leq \mathbf{m}_{i,k} + \mathbf{m}_{k,j}$

- `Floyd-Warshall algorithm`
- Complexity of n^3 where n is the number of variables

Closure

Closure

A DBM m is `closed`, and denoted by m^* iff

- $\forall i.\mathbf{m}_{i,i} = 0$
- $\forall i, j, k.\mathbf{m}_{i,j} \leq \mathbf{m}_{i,k} + \mathbf{m}_{k,j}$

- `Floyd-Warshall algorithm`
- Complexity of n^3 where n is the number of variables

It is basically a loop computing n matrices, m^1 to m^n, as follows

$$\begin{cases} m^0 \stackrel{\text{def}}{=} m \\ m^k_{i,j} \stackrel{\text{def}}{=} \min(m^{k-1}_{i,j}, m^{k-1}_{i,k} + m^{k-1}_{k,j}), \quad \text{if } 1 \leq i, j, k \leq n \\ m^*_{i,j} \stackrel{\text{def}}{=} \begin{cases} m^n_{i,j}, & \text{if } i \neq j \\ 0, & \text{if } i = j \end{cases} \end{cases}$$

Example of application of the closure operator

```
function CLOSE(m)
    for k ∈ {0, ..., 2n − 1} do
        for i ∈ {0, ..., 2n − 1} do
            for j ∈ {0, ..., 2n − 1} do
                m'_{i,j} ← min(m_{i,j}, m_{i,k} + m_{k,j})
            end for
        end for
    end for
    return m'
end function
```

Figure 1: Floyd-Warshall algorithm for computing closure of a DBM.

Example of application of the closure operator

```
function CLOSE(m)
    for k ∈ {0, ..., 2n − 1} do
        for i ∈ {0, ..., 2n − 1} do
            for j ∈ {0, ..., 2n − 1} do
                m'_{i,j} ← min(m_{i,j}, m_{i,k} + m_{k,j})
            end for
        end for
    end for
    return m'
end function
```

Figure 1: Floyd-Warshall algorithm for computing closure of a DBM.

	x'_0	x'_1	x'_2	x'_3
x'_0	∞	6	∞	6
x'_1	6	∞	5	∞
x'_2	∞	6	∞	4
x'_3	5	∞	∞	11

Example of application of the closure operator

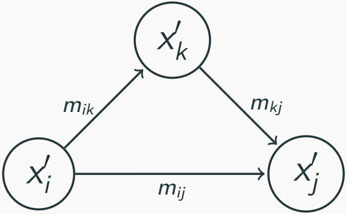
```
function CLOSE(m)
    for k ∈ {0, ..., 2n − 1} do
        for i ∈ {0, ..., 2n − 1} do
            for j ∈ {0, ..., 2n − 1} do
                m'_{i,j} ← min(m_{i,j}, m_{i,k} + m_{k,j})
            end for
        end for
    end for
    return m'
end function
```

Figure 1: Floyd-Warshall algorithm for computing closure of a DBM.

	x'_0	x'_1	x'_2	x'_3
x'_0	0	6	11	6
x'_1	6	0	5	9
x'_2	9	6	0	4
x'_3	5	11	16	0

Implicit constraints

For each node x_k in turn, it checks, for all pairs (x_i, x_j), whether it would be shorter to pass through x_k instead of taking the direct arc from x_i to x_j.



This also corresponds to adding the constraints

$$x_i - x_k \leq c \wedge x_k - x_j \leq d$$

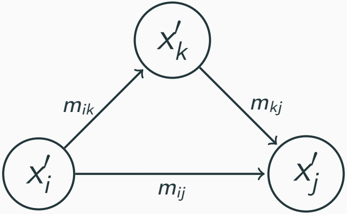
to derive the constraint (called `implicit constraint`*)*

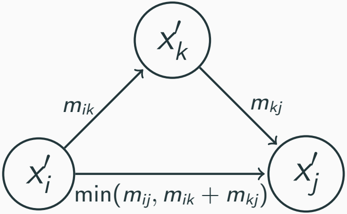
$$x_i - x_j \leq c + d$$

The closure makes all `implicit` *constraints* `explicit`*.*

Implicit constraints

For each node x_k in turn, it checks, for all pairs (x_i, x_j), whether it would be shorter to pass through x_k instead of taking the direct arc from x_i to x_j.





This also corresponds to adding the constraints

$$x_i - x_k \leq c \wedge x_k - x_j \leq d$$

to derive the constraint (called *implicit constraint)*

$$x_i - x_j \leq c + d$$

The closure makes all *implicit* *constraints* *explicit*.

Strong closure

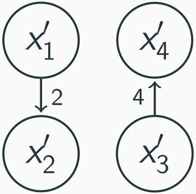
Closure is not sufficient for obtaining canonical form of the DBM.

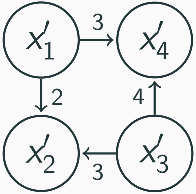
- Same octagon $x_1 \leq 1 \wedge x_2 \leq 2$
- $x_1 + x_2 \leq 3$

Strong closure

Closure is not sufficient for obtaining canonical form of the DBM.

- Same octagon $x_1 \leq 1 \wedge x_2 \leq 2$
- $x_1 + x_2 \leq 3$

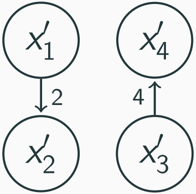


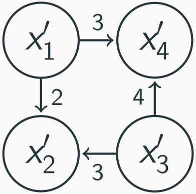


Strong closure

Closure is not sufficient for obtaining canonical form of the DBM.

- Same octagon $x_1 \leq 1 \wedge x_2 \leq 2$
- $x_1 + x_2 \leq 3$





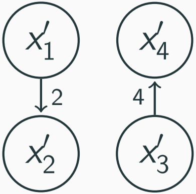
	x'_1	x'_2	x'_3	x'_4
x'_1	0	2	∞	∞
x'_2	∞	0	∞	∞
x'_3	∞	∞	0	4
x'_4	∞	∞	∞	0

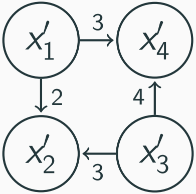
	x'_1	x'_2	x'_3	x'_4
x'_1	∞	2	∞	3
x'_2	∞	∞	∞	∞
x'_3	∞	3	∞	4
x'_4	∞	∞	∞	∞

Strong closure

Closure is not sufficient for obtaining canonical form of the DBM.

- Same octagon $x_1 \leq 1 \wedge x_2 \leq 2$
- $x_1 + x_2 \leq 3$





	x'_1	x'_2	x'_3	x'_4
x'_1	0	2	∞	∞
x'_2	∞	0	∞	∞
x'_3	∞	∞	0	4
x'_4	∞	∞	∞	0

	x'_1	x'_2	x'_3	x'_4
x'_1	0	2	∞	3
x'_2	∞	0	∞	∞
x'_3	∞	3	0	4
x'_4	∞	∞	∞	0

Strong closure - Intuition

As explained before, Floyd-Warshall algorithm as performing local constraints propagations of the form

$$x'_i - x'_k \leq c \wedge x'_k - x'_j \leq d \implies x'_i - x'_j \leq c + d$$

on $\mathcal{V}'$ until no further propagation can be done.

Strong closure - Intuition

As explained before, Floyd-Warshall algorithm as performing local constraints propagations of the form

$$x'_i - x'_k \leq c \wedge x'_k - x'_j \leq d \implies x'_i - x'_j \leq c + d$$

on $\mathcal{V}'$ until no further propagation can be done.

The idea of `strong closure` is to add another step of local constraints propagation.

$$x'_i - x'_{\bar{\imath}} \leq c \wedge x'_{\bar{\jmath}} - x'_j \leq d \implies x'_i - x'_j \leq (c + d)/2$$

such that $x'_i = -x'_{\bar{\imath}}$

so $m_{i,j}$ is replacing with $\min(m_{i,j}, (m_{i,\bar{\imath}} + m_{\bar{\jmath},j})/2)$.

Strong closure

Strong Closure

A DBM m is `strongly closed` iff

- $\mathbf{m}$ is closed
- $\forall i, j \cdot \mathbf{m}_{i,j} \leq \mathbf{m}_{i,\bar{\imath}}/2 + \mathbf{m}_{\bar{\jmath},j}/2$

Strong closure

Strong Closure

A DBM m is `strongly closed` iff

- $\mathbf{m}$ is closed
- $\forall i, j \cdot \mathbf{m}_{i,j} \leq \mathbf{m}_{i,\bar{\imath}}/2 + \mathbf{m}_{\bar{\jmath},j}/2$

	x'_1	x'_2	x'_3	x'_4
x'_1	∞	2	∞	∞
x'_2	∞	∞	∞	∞
x'_3	∞	∞	∞	4
x'_4	∞	∞	∞	∞

	x'_1	x'_2	x'_3	x'_4
x'_1	∞	2	∞	3
x'_2	∞	∞	∞	∞
x'_3	∞	3	∞	4
x'_4	∞	∞	∞	∞

Strong closure

Strong Closure

A DBM m is `strongly closed` iff

- $\mathbf{m}$ is closed
- $\forall i, j \cdot \mathbf{m}_{i,j} \leq \mathbf{m}_{i,\bar{\imath}}/2 + \mathbf{m}_{\bar{\jmath},j}/2$

	x'_1	x'_2	x'_3	x'_4
x'_1	0	2	∞	0
x'_2	∞	0	∞	∞
x'_3	∞	3	0	4
x'_4	∞	∞	∞	0

	x'_1	x'_2	x'_3	x'_4
x'_1	∞	2	∞	3
x'_2	∞	∞	∞	∞
x'_3	∞	3	∞	4
x'_4	∞	∞	∞	∞

Strong closure

Strong Closure

A DBM m is `strongly closed` iff

- $\mathbf{m}$ is closed
- $\forall i, j \cdot \mathbf{m}_{i,j} \leq \mathbf{m}_{i,\bar{\imath}}/2 + \mathbf{m}_{\bar{\jmath},j}/2$

	x'_1	x'_2	x'_3	x'_4
x'_1	0	2	∞	0
x'_2	∞	0	∞	∞
x'_3	∞	3	0	4
x'_4	∞	∞	∞	0

	x'_1	x'_2	x'_3	x'_4
x'_1	0	2	∞	3
x'_2	∞	0	∞	∞
x'_3	∞	3	0	4
x'_4	∞	∞	∞	0

Product of abstract domain

Back to our example: three kinds of constraints in RCPSP

- octagonal constraints treated by octagon abstract domain.
- equivalence constraints treated in a specialized reduced product.
- interval constraints treated by the PP abstract domain.

$$\forall (i \ll j) \in P, s_i + d_i \leq s_j$$

$$\forall j \in [1..n], \forall i \in [1..n] \setminus \{j\}, b_{i,j} \Leftrightarrow (s_i \leq s_j \wedge s_j < s_i + d_i)$$

$$\forall j \in [1..n], r_{k,j} + \left(\sum_{i \in [1..n] \setminus \{j\}} r_{k,i} * b_{i,j} \right) \leq c_k$$

Basic product

We can define a direct product over $PP \times Oct$ as follows:

$$(p, o) \sqcup (p', o') = (p \sqcup_{PP} p', o \sqcup_{Oct} o')$$

$$[\![\varphi]\!] = \begin{cases} ([\![\varphi]\!]_{PP}, [\![\varphi]\!]_{Oct}) & \\ ([\![\varphi]\!]_{PP}, \bot_{Oct}) & \text{if } [\![\varphi]\!]_{Oct} \text{ is not defined} \\ (\bot_{PP}, [\![\varphi]\!]_{Oct}) & \text{if } [\![\varphi]\!]_{PP} \text{ is not defined} \end{cases}$$

$$refine((p, o)) = (refine(p), refine(o))$$

Basic product

We can define a direct product over $PP \times Oct$ as follows:

$$(p, o) \sqcup (p', o') = (p \sqcup_{PP} p', o \sqcup_{Oct} o')$$

$$[\![\varphi]\!] = \begin{cases} ([\![\varphi]\!]_{PP}, [\![\varphi]\!]_{Oct}) & \\ ([\![\varphi]\!]_{PP}, \bot_{Oct}) & \text{if } [\![\varphi]\!]_{Oct} \text{ is not defined} \\ (\bot_{PP}, [\![\varphi]\!]_{Oct}) & \text{if } [\![\varphi]\!]_{PP} \text{ is not defined} \end{cases}$$

$$refine((p, o)) = (refine(p), refine(o))$$

Issue: domains do not exchange information.

Reduced product via equivalence constraints [Talbot et al.]

We can improve the refinement operator of the direct product by connecting constraints from both domains via equivalence constraints.

- Let $\varphi_1 \Leftrightarrow \varphi_2$ be an equivalence constraint where $[\![\varphi_1]\!]_{PP}$ and $[\![\varphi_2]\!]_{Oct}$ are defined, then we have:

$$
\begin{aligned}
&\mathit{prop}_{\Leftrightarrow}(p, o, \varphi_1 \Leftrightarrow \varphi_2) \triangleq \\
&\begin{cases}
p \models_{PP} \varphi_1 \implies (p, o \sqcup [\![\varphi_2]\!]_{Oct}) \\
p \models_{PP} \neg\varphi_1 \implies (p, o \sqcup [\![\neg\varphi_2]\!]_{Oct}) \\
o \models_{Oct} \varphi_2 \implies (p \sqcup [\![\varphi_1]\!]_{PP}, o) \\
o \models_{Oct} \neg\varphi_2 \implies (p \sqcup [\![\neg\varphi_1]\!]_{PP}, o) \\
(p, o) \text{ otherwise}
\end{cases}
\end{aligned}
$$

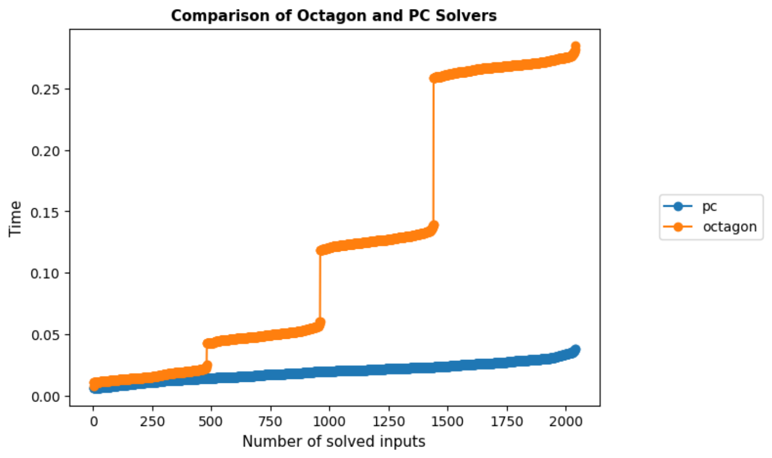
- Result: A generic reduced product to combine abstract domains with disjoint set of variables.

Experiments

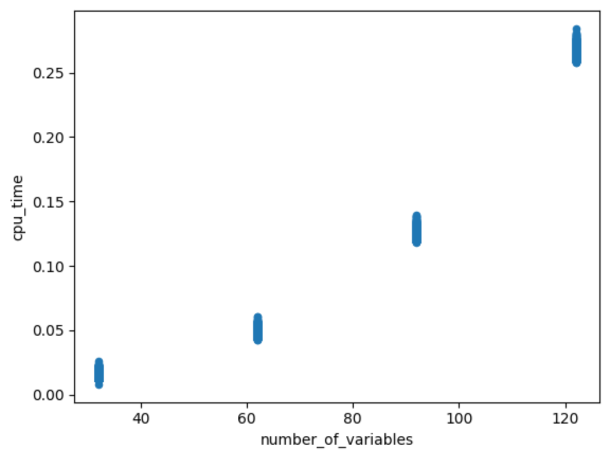
Instances & Environment

- 2040 instances
  - from XCSP3 world
- STP instances (so RCPSP with only the precedence constraints)
- 120 variables
- Precision 7780 13th Gen Intel(R) Core(TM) i9-13950HX
- Timeout of 20 seconds

Octagon vs PC



Octagon vs PC



Conclusion

Conclusion

- RCPSP problem and this modelization
- Different abstract domains $\rightarrow$ Octagon abstract domain with these operators and this representation
- Based on these concepts, we model the RCPSP problem using abstract domains.
- Some experiments

Octagon Abstract Domain

Session 6

Thibault Falque

Abstract Interpretation Workshop – 20th June 2024

University of Luxembourg

UNIVERSITÉ DU LUXEMBOURG

Bibliography

Andreas Schutt, Thibaut Feydy, Peter J. Stuckey, and Mark G. Wallace. Why Cumulative Decomposition Is Not as Bad as It Sounds. In Ian P. Gent, editor, *Principles and Practice of Constraint Programming - CP 2009*, volume 5732, pages 746–761. Springer Berlin Heidelberg. ISBN 978-3-642-04243-0 978-3-642-04244-7. doi: 10.1007/978-3-642-04244-7_58. URL http://link.springer.com/10.1007/978-3-642-04244-7_58.

Pierre Talbot, David Cachera, Eric Monfroy, and Charlotte Truchet. Combining Constraint Languages via Abstract Interpretation. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 50–58. IEEE. ISBN 978-1-72813-798-8. doi: 10.1109/ICTAI.2019.00016. URL https://ieeexplore.ieee.org/document/8995453/.