Associating Narrowing Functions to Constraints

Jorge Cruz

DI/FCT/UNL

2020

Associating Narrowing Functions to Constraints

Narrowing Functions and their Properties

Narrowing Functions for Single Constraints

- Constraint Decomposition Method
- Constraint Newton Method
- Revise Procedures

Narrowing Functions for Systems of Constraints

- Multivariate Interval Newton
- Reformulation-Linearization
- Linear relaxation based on affine arithmetic
- Corner-based Taylor relaxation

Narrowing Functions and their Properties

The pruning of the variable domains according to the constraints of a CCSP is based on narrowing functions

A narrowing function must be able to narrow the domains (contractance) without loosing solutions (correctness)

Narrowing Function. Let $P=(X,D,C)$ be a CCSP. A narrowing function NF associated with a constraint $c=(s,\rho)$ (with $c\in C$) is a mapping between elements of 2^D with the following properties (where A is any element of the domain of NF):

P1) $NF(A)\subseteq A$ (contractance)

P2) $\forall_{d\in A}\; d\notin NF(A) \Rightarrow d[s]\notin\rho$ (correctness) ❑

Monotonicity and *Idempotency* are additional properties common to most of the narrowing functions used in interval constraints

Monotonicity and Idempotency. Let $P=(X,D,C)$ be a CCSP. Let NF be a narrowing function associated with a constraint of C. Let A_1 and A_2 be any elements of the domain of NF. NF is respectively monotonic and idempotent iff the following properties hold:

P3) $A_1\subseteq A_2 \Rightarrow NF(A_1)\subseteq NF(A_2)$ (monotonicity)

P4) $NF(NF(A_1)) = NF(A_1)$ (idempotency) ❑

Narrowing functions from Interval Extension Evaluation

The simplest method to associate a narrowing function to a constraint $f(\mathbf{x}) = 0$ consists in evaluating with interval arithmetic an interval extension F of f with the current box B and check whether 0 belongs to the result. If not, the box may be discarded.

$$NF_{f(x)=0}(B) = \begin{cases} B & if\ 0 \in F(B) \\ \emptyset & otherwise \end{cases}$$

No solution is lost since by the definition of interval extension:

$\forall_{x \in B} f(x) \in F([x])$ and by monotonicity: $[x] \subseteq B \rightarrow F([x]) \subseteq F(B)$

This binary contractor (in the sense that it keeps all or nothing) can be easily extended to inequalities and may be applied with different interval extensions of f or even their intersection.

Narrowing functions from Interval Extension Evaluation

Another method to associate narrowing functions to a multivariate constraint $f(\mathbf{x}) = 0$ is by solving it wrt to each variable and evaluate the expressions in current box B to narrow their domains.

Example: $f(x_1, x_2) = x_1 \exp(x_2) - 1$

$$NF_{f(x_1,x_2)=0}(\langle I_1, I_2 \rangle) = \left\langle I_1 \cap \frac{1}{\exp(I_2)}, I_2 \cap log\left(\frac{1}{I_1}\right) \right\rangle$$

It is not always possible to solve an equation wrt a variable!

Example: $f(x_1, x_2) = x_1 \exp(x_1 x_2) - \sin(x_2)$

Projection Function and its Enclosure

Usually, narrowing functions are associated with a constraint by considering projections with respect to each variable in the scope

A projection function identifies from a box:

all the possible values of a particular variable for which there is a value combination belonging to the constraint relation

Projection Function. Let $P=(X,D,C)$ be a CCSP. The projection function with respect to a constraint $c=(s,\rho)\in C$ and a variable $x_i \in s$, denoted $\pi_{x_i}^{\rho}$, obtains a set of real values from a real box B and is defined by:

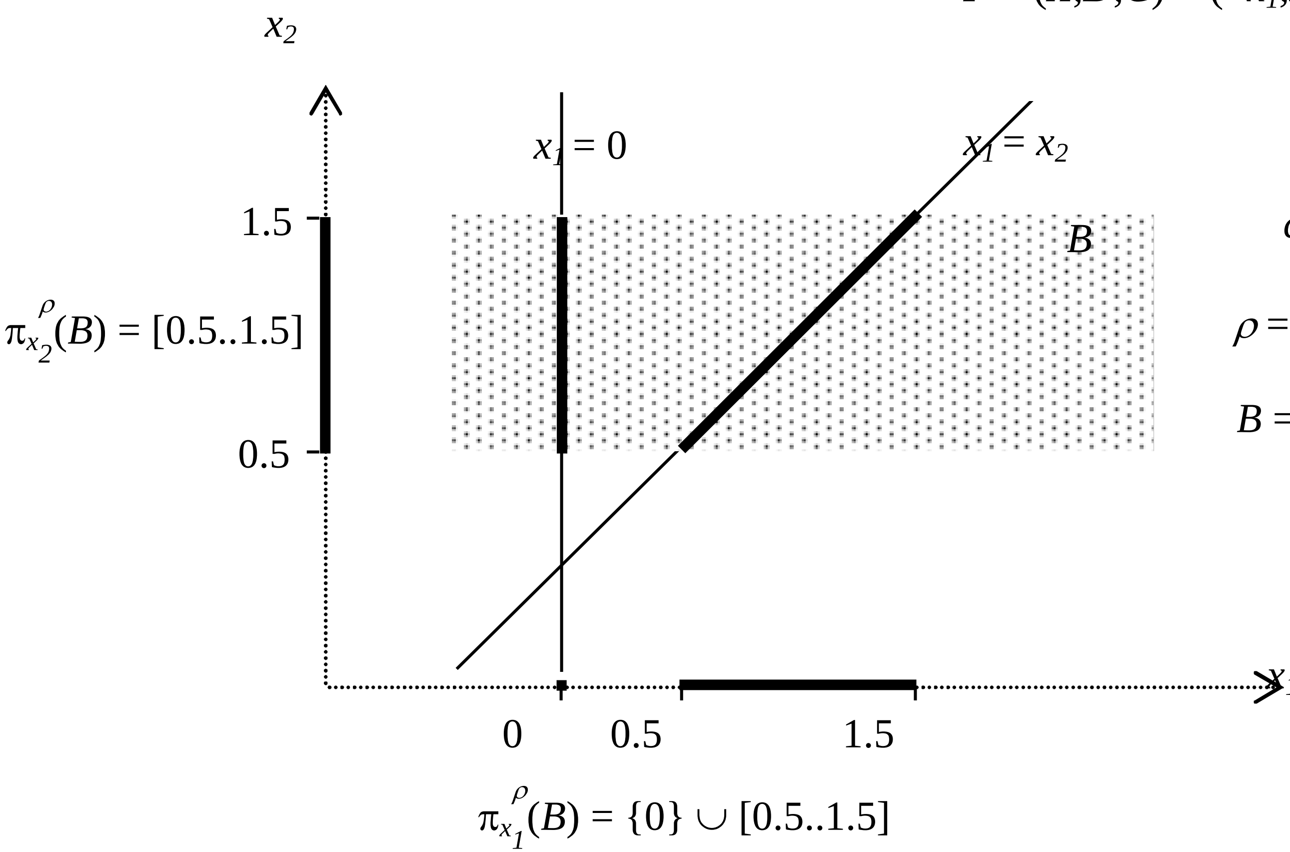
$$\pi_{x_i}^{\rho}(B) = \{ d[x_i] \mid d \in \rho \wedge d \in B \} = (\rho \cap B)[x_i]$$ ❑

All value combinations within B with x_i values outside $\pi_{x_i}^{\rho}(B)$ are outside the relation ρ and so they do not satisfy the constraint c.

Projection Function and its Enclosure

$P = (X,D,C) = (\langle x_1,x_2\rangle, D_1\times D_2, \{c\})$

$c \equiv x_1\times(x_2-x_1) = 0$



$c = (\langle x_1,x_2\rangle,\rho)$

$\rho = \{\ \langle x_1,x_2\rangle \in D \mid x_1\times(x_2-x_1) = 0\ \}$

$B = \langle[-0.5..2.5],[0.5..1.5]\rangle$

All value combinations within B with x_i values outside $\pi_{x_i}^{\rho}(B)$ are outside the relation ρ and so they do not satisfy the constraint c.

Projection Function and its Enclosure

A box-narrowing function narrows the domain of one variable, from a box representing all the variables of the CCSP, eliminating some values that do not belong to a projection function

Box-Narrowing Function. Let $P=(X,D,C)$ be a CCSP (with $X=\langle x_1,\ldots,x_i,\ldots,x_n\rangle$). A box-narrowing function with respect to a constraint $(s,\rho)\in C$ and a variable $x_i\in s$ is a mapping, denoted $\mathrm{BNF}_{x_i}^{\rho}$, that relates any F-box $B=\langle I_{x_1},\ldots, I_{x_i},\ldots, I_{x_n}\rangle$ $(B\subseteq D)$ with the union of m $(1\leq m)$ F-boxes, defined by:

$$\mathrm{BNF}_{x_i}^{\rho}(\langle I_{x_1},\ldots, I_{x_i},\ldots, I_{x_n}\rangle) = \langle I_{x_1},\ldots, I_1,\ldots, I_{x_n}\rangle \cup \ldots \cup \langle I_{x_1},\ldots, I_m,\ldots, I_{x_n}\rangle$$

satisfying the property:

$$\pi_{x_i}^{\rho}(B[s]) \subseteq I_1 \cup \ldots \cup I_m \subseteq I_{x_i}$$ ❑

Contractance follows from $I_1\cup\ldots\cup I_m \subseteq I_{x_i}$ (the only changed domain is smaller than the original)

Correctness follows from $\pi_{x_i}{}^{\rho}(B[s]) \subseteq I_1 \cup \ldots \cup I_m$ (the eliminated combinations have x_i values outside the projection function)

Constraint Decomposition Method

Decomposition of complex constraints into an equivalent set of primitive constraints whose projection functions can be easily computed by inverse functions

Primitive Constraints

Primitive Constraint. Let e_c be a real expression with at most one basic operator and with no multiple occurrences of its variables. Let e_0 be a real constant or a real variable not appearing in e_c. The constraint c is a primitive constraint iff it is expressed as:

$$e_c \diamond e_0 \quad \text{with} \quad \diamond \in \{\leq, =, \geq\}$$ ❑

A set of primitive constraints can be easily obtained from any non-primitive constraint:

A constraint may be decomposed by considering new variables and new equality constraints

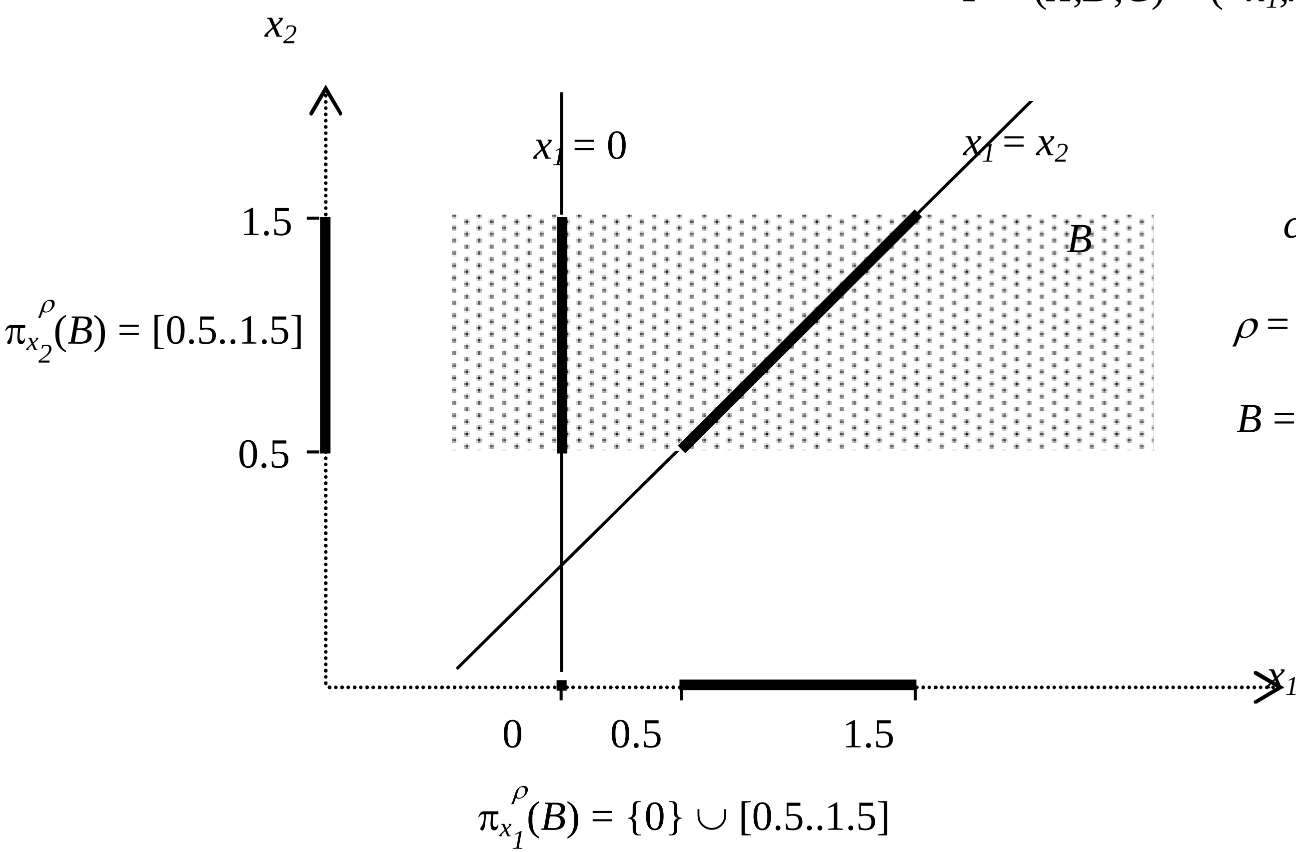
The whole set of primitives may be obtained by repeating this process until all constraints are primitive

Constraint Decomposition Method

Primitive Constraints

$P = (X,D,C) = (<x_1,x_2>,D_1 \times D_2,\{c\})$

$c \equiv x_1 \times (x_2 - x_1) = 0$



$c = (<x_1,x_2>,\rho)$

$\rho = \{ <x_1,x_2> \in D \mid x_1 \times (x_2 - x_1) = 0 \}$

$B =<[-0.5..2.5],[0.5..1.5]>$

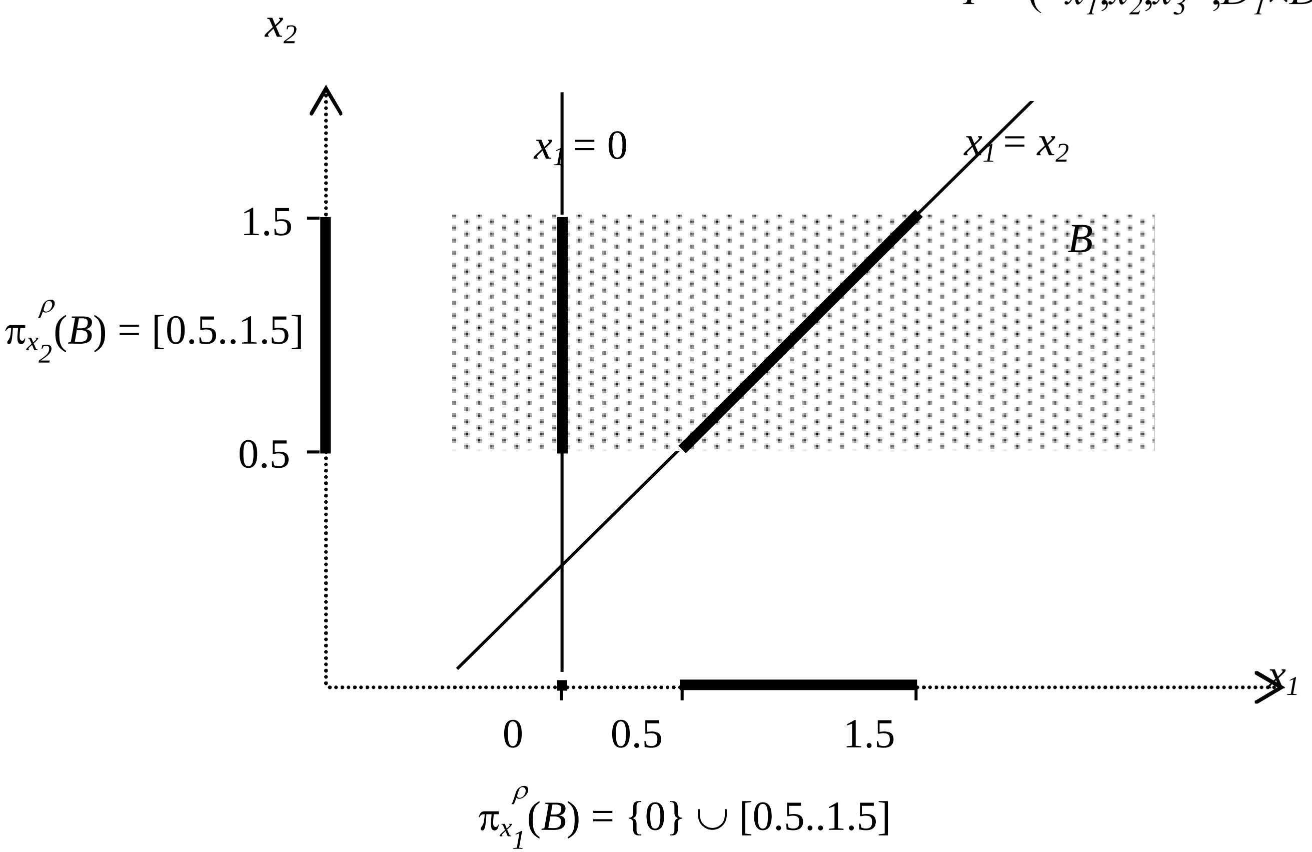
The constraint c is not primitive since it contains two basic arithmetic operators and the variable x_1 occurs twice

Constraint Decomposition Method

Primitive Constraints

$P' = (\langle x_1, x_2, x_3 \rangle, D_1 \times D_2 \times [-\infty..+\infty], \{c_1, c_2\})$

$$c \begin{cases} c_1 \equiv x_1 \times x_3 = 0 \\ c_2 \equiv x_2 - x_1 = x_3 \end{cases}$$



a new variable x_3 is introduced and c is replaced by c_1 and c_2

the domain of x_3 is unbounded defining a new equivalent CCSP P'

Constraint Decomposition Method

Inverse Functions

Inverse Interval Expression. Let c=(s,ρ) be a primitive constraint expressed in the form $e_c \diamond e_0$ where $e_c \equiv e_1$ or $e_c \equiv \Phi(e_1,\ldots,e_m)$ (Φ is an m-ary basic operator and e_i a variable from s or a real constant). The inverse interval expression of c with respect to e_i, denoted Ψe_i, is the natural interval expression of the expression obtained by solving algebraically, wrt e_i, the equality e_c=e_0 if c is an equality or e_c=e_0+k if c is an inequality (with $k \leq 0$ for inequalities of the form $e_c \leq e_0$ and $k \geq 0$ for inequalities of the form $e_c \geq e_0$). ❑

	Ψe_1	Ψe_2	Ψe_3
$e_1+e_2 \diamond e_3$	$(E_3+K)-E_2$	$(E_3+K)-E_1$	$(E_1+E_2)-K$
$e_1-e_2 \diamond e_3$	$(E_3+K)+E_2$	$E_1-(E_3+K)$	$(E_1-E_2)-K$
$e_1 \times e_2 \diamond e_3$	$(E_3+K)/E_2$	$(E_3+K)/E_1$	$(E_1 \times E_2)-K$
$e_1/e_2 \diamond e_3$	$(E_3+K) \times E_2$	$E_1/(E_3+K)$	$(E_1/E_2)-K$
$e_1 \diamond e_2$	(E_2+K)	E_1-K	

$\diamond \in \{\leq,=,\geq\}$

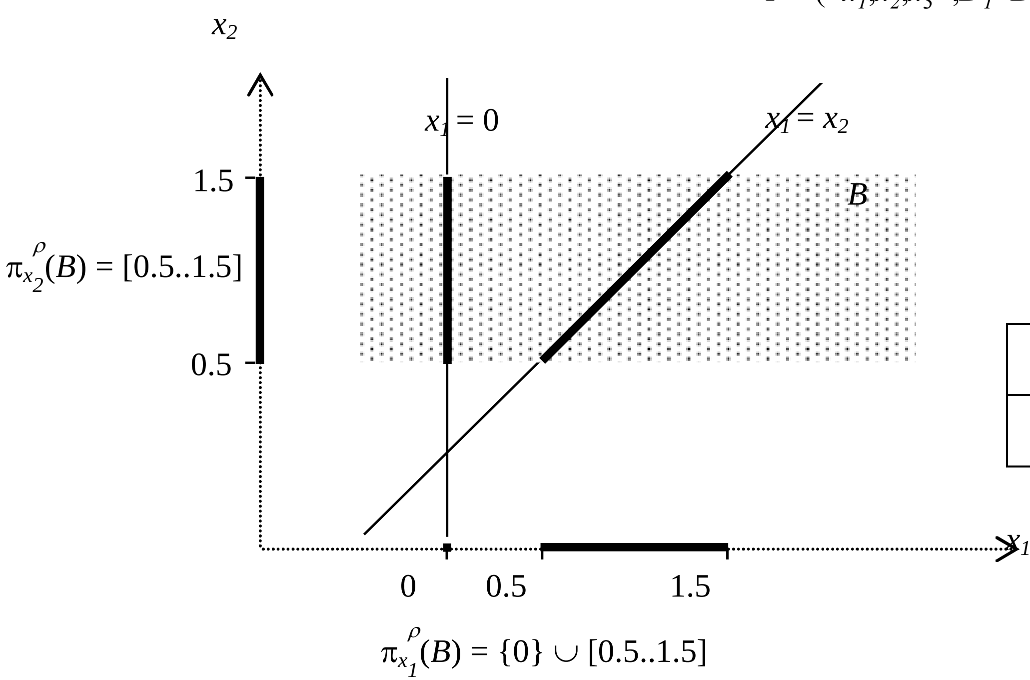
e_i is a real variable or a real constant

E_i is the natural interval extension of e_i

$$K=\begin{cases}[-\infty..0] & \text{if } \diamond \equiv \leq \\ [0..0] & \text{if } \diamond \equiv = \\ [0..+\infty] & \text{if } \diamond \equiv \geq\end{cases}$$

Constraint Decomposition Method

Inverse Functions



$P' = (\langle x_1, x_2, x_3 \rangle, D_1 \times D_2 \times [-\infty..+\infty], \{c_1, c_2\})$

$$c \begin{cases} c_1 \equiv x_1 \times x_3 = 0 \\ c_2 \equiv x_2 - x_1 = x_3 \end{cases}$$

	Ψ_{e_1}	Ψ_{e_2}	Ψ_{e_3}
$x_1 \times x_3 = 0$	$0/X_3$	$0/X_1$	$X_1 \times X_3$
$x_2 - x_1 = x_3$	$X_3 + X_1$	$X_2 - X_3$	$X_2 - X_1$

The inverse interval expressions are associated with the primitive constraints of the decomposed CCSP *P'*

Constraint Decomposition Method

Projection Function Enclosure with the Inverse Function

The inverse interval expression wrt a variable allows the definition of the projection function of the constraint wrt to that variable

Projection Function based on the Inverse Interval Expression. Let $P=(X,D,C)$ be a CCSP. Let $c=(s,\rho)\in C$ be an n-ary primitive constraint expressed in the form $e_c \diamond e_0$ where $e_c \equiv e_1$ or $e_c \equiv \Phi(e_1,\ldots,e_m)$ (with Φ an m-ary basic operator and e_i a variable from s or a real constant). Let Ψx_i be the inverse interval expression of c with respect to the variable x_i ($e_i \equiv x_i$). The projection function $\pi_{x_i}^{\rho}$ of the constraint c wrt variable x_i is the mapping:

$\pi_{x_i}^{\rho}(B) = \Psi x_i(B) \cap B[x_i]$ where B is an n-ary real box ❑

$x_1 \times x_3 = 0$
$\pi_{x_1}^{\rho}(\langle I_1,I_3\rangle) = (0/I_3) \cap I_1$
$\pi_{x_3}^{\rho}(\langle I_1,I_3\rangle) = (0/I_1) \cap I_3$

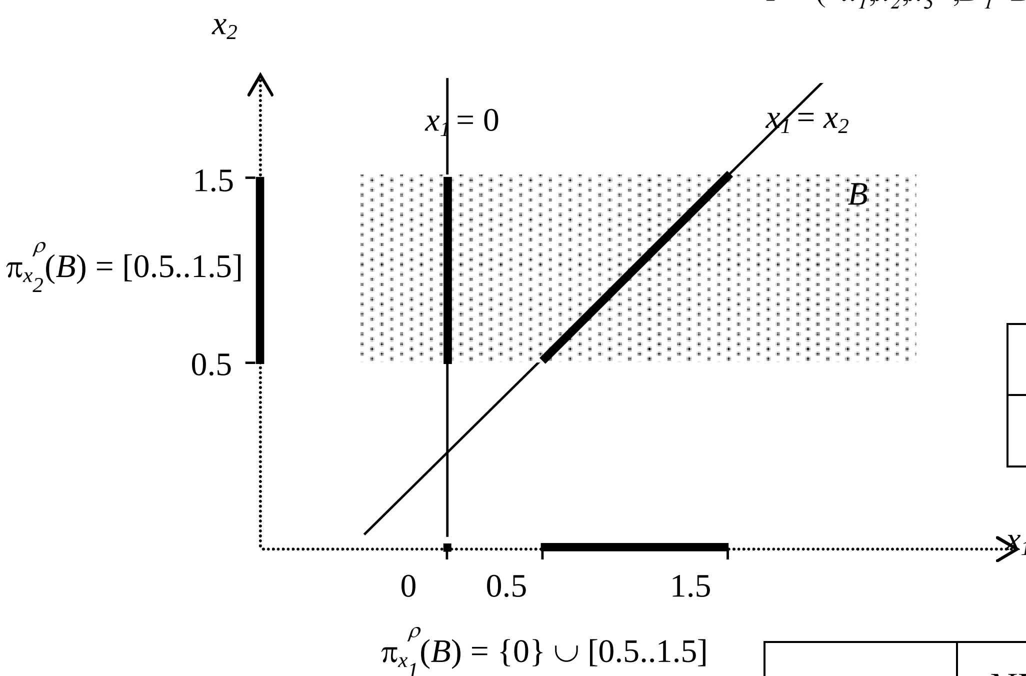
$x_2 - x_1 = x_3$
$\pi_{x_1}^{\rho}(\langle I_1,I_2,I_3\rangle) = (I_2-I_3) \cap I_1$
$\pi_{x_2}^{\rho}(\langle I_1,I_2,I_3\rangle) = (I_3+I_1) \cap I_2$
$\pi_{x_3}^{\rho}(\langle I_1,I_2,I_3\rangle) = (I_2-I_1) \cap I_3$

Constraint Decomposition Method

Projection Function Enclosure with the Inverse Function

$P' = (\langle x_1, x_2, x_3\rangle, D_1 \times D_2 \times [-\infty..+\infty], \{c_1, c_2\})$

$$c \begin{cases} c_1 \equiv x_1 \times x_3 = 0 \\ c_2 \equiv x_2 - x_1 = x_3 \end{cases}$$



	Ψe_1	Ψe_2	Ψe_3
$x_1 \times x_3 = 0$	$0/X_3$	$0/X_1$	$X_1 \times X_3$
$x_2 - x_1 = x_3$	$X_3 + X_1$	$X_2 - X_3$	$X_2 - X_1$

Box-narrowing functions are associated with the decomposed CCSP *P'*

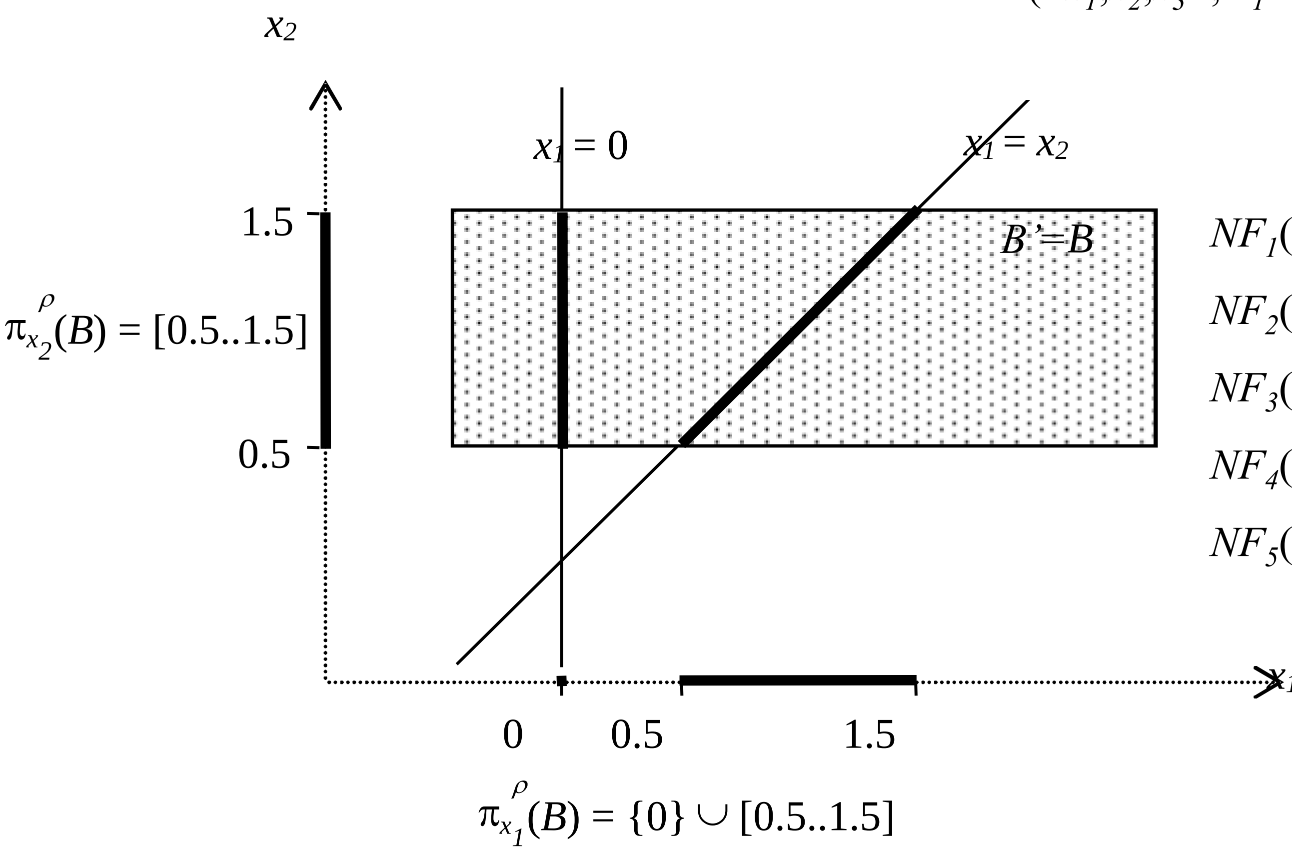
$x_1 \times x_3 = 0$	NF_1	$BNF^{\rho}_{x_1}(\langle I_1, I_2, I_3\rangle) = \langle (0/I_3) \cap I_1, I_2, I_3\rangle$
	NF_2	$BNF^{\rho}_{x_3}(\langle I_1, I_2, I_3\rangle) = \langle I_1, I_2, (0/I_1) \cap I_3\rangle$
$x_2 - x_1 = x_3$	NF_3	$BNF^{\rho}_{x_1}(\langle I_1, I_2, I_3\rangle) = \langle (I_2 - I_3) \cap I_1, I_2, I_3\rangle$
	NF_4	$BNF^{\rho}_{x_2}(\langle I_1, I_2, I_3\rangle) = \langle I_1, (I_3 + I_1) \cap I_2, I_3\rangle$
	NF_5	$BNF^{\rho}_{x_3}(\langle I_1, I_2, I_3\rangle) = \langle I_1, I_2, (I_2 - I_1) \cap I_3\rangle$

Constraint Decomposition Method

Example

$P' = (\langle x_1, x_2, x_3 \rangle, D_1 \times D_2 \times [-\infty..+\infty], \{c_1, c_2\})$

$$c \begin{cases} c_1 \equiv x_1 \times x_3 = 0 \\ c_2 \equiv x_2 - x_1 = x_3 \end{cases}$$



$NF_1(\langle I_1, I_2, I_3 \rangle) = \langle (0/I_3) \cap I_1, I_2, I_3 \rangle$

$NF_2(\langle I_1, I_2, I_3 \rangle) = \langle I_1, I_2, (0/I_1) \cap I_3 \rangle$

$NF_3(\langle I_1, I_2, I_3 \rangle) = \langle (I_2 - I_3) \cap I_1, I_2, I_3 \rangle$

$NF_4(\langle I_1, I_2, I_3 \rangle) = \langle I_1, (I_3 + I_1) \cap I_2, I_3 \rangle$

$NF_5(\langle I_1, I_2, I_3 \rangle) = \langle I_1, I_2, (I_2 - I_1) \cap I_3 \rangle$

with $B = \langle [-0.5, 2.5], [0.5, 1.5] \rangle$ no pruning would be obtained:

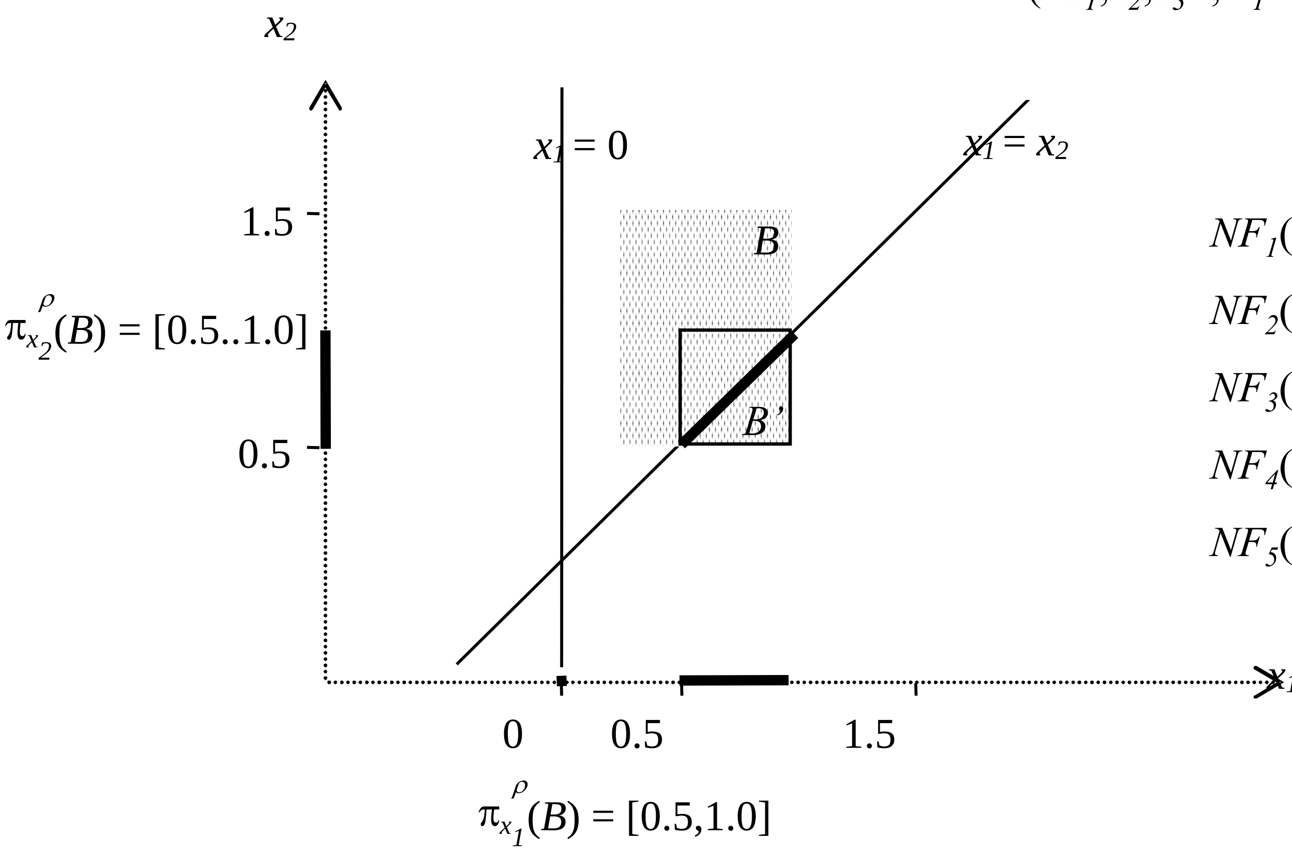
$B' = \langle [-0.5, 2.5], [0.5, 1.5] \rangle$ and $x_3 = [-2.0, 2.0]$

Constraint Decomposition Method

Example

$P' = (\langle x_1, x_2, x_3 \rangle, D_1 \times D_2 \times [-\infty..+\infty], \{c_1, c_2\})$

$$c \begin{cases} c_1 \equiv x_1 \times x_3 = 0 \\ c_2 \equiv x_2 - x_1 = x_3 \end{cases}$$



$NF_1(\langle I_1, I_2, I_3 \rangle) = \langle (0/I_3) \cap I_1, I_2, I_3 \rangle$

$NF_2(\langle I_1, I_2, I_3 \rangle) = \langle I_1, I_2, (0/I_1) \cap I_3 \rangle$

$NF_3(\langle I_1, I_2, I_3 \rangle) = \langle (I_2 - I_3) \cap I_1, I_2, I_3 \rangle$

$NF_4(\langle I_1, I_2, I_3 \rangle) = \langle I_1, (I_3 + I_1) \cap I_2, I_3 \rangle$

$NF_5(\langle I_1, I_2, I_3 \rangle) = \langle I_1, I_2, (I_2 - I_1) \cap I_3 \rangle$

with $B = \langle [0.25, 1.0], [0.5, 1.5] \rangle$ the best narrowing is obtained:

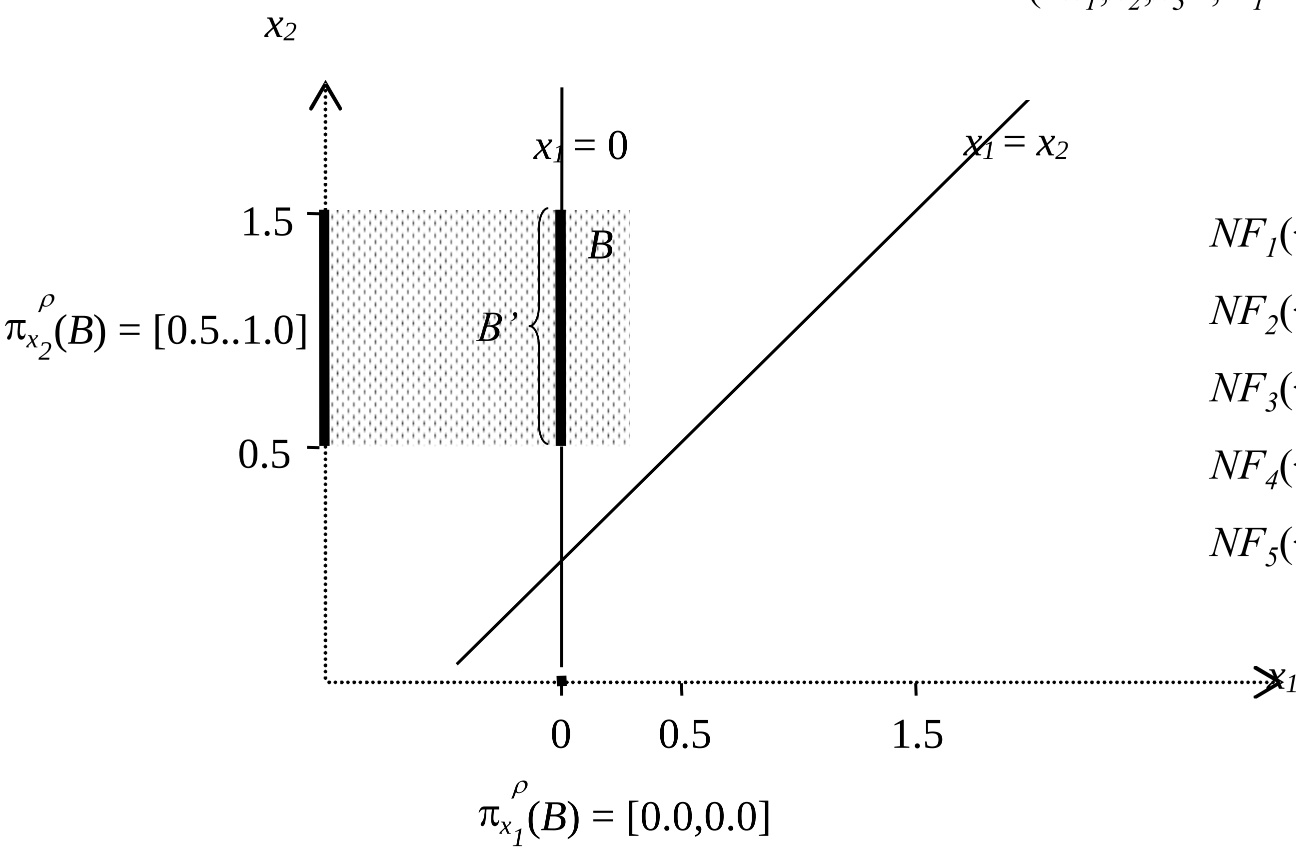
$B' = \langle [0.5, 1.0], [0.5, 1.0] \rangle$ and $x_3 = [0.0, 0.0]$

Constraint Decomposition Method

Example

$P' = (\langle x_1, x_2, x_3 \rangle, D_1 \times D_2 \times [-\infty..+\infty], \{c_1, c_2\})$

$$c \begin{cases} c_1 \equiv x_1 \times x_3 = 0 \\ c_2 \equiv x_2 - x_1 = x_3 \end{cases}$$



$NF_1(\langle I_1, I_2, I_3 \rangle) = \langle (0/I_3) \cap I_1, I_2, I_3 \rangle$

$NF_2(\langle I_1, I_2, I_3 \rangle) = \langle I_1, I_2, (0/I_1) \cap I_3 \rangle$

$NF_3(\langle I_1, I_2, I_3 \rangle) = \langle (I_2 - I_3) \cap I_1, I_2, I_3 \rangle$

$NF_4(\langle I_1, I_2, I_3 \rangle) = \langle I_1, (I_3 + I_1) \cap I_2, I_3 \rangle$

$NF_5(\langle I_1, I_2, I_3 \rangle) = \langle I_1, I_2, (I_2 - I_1) \cap I_3 \rangle$

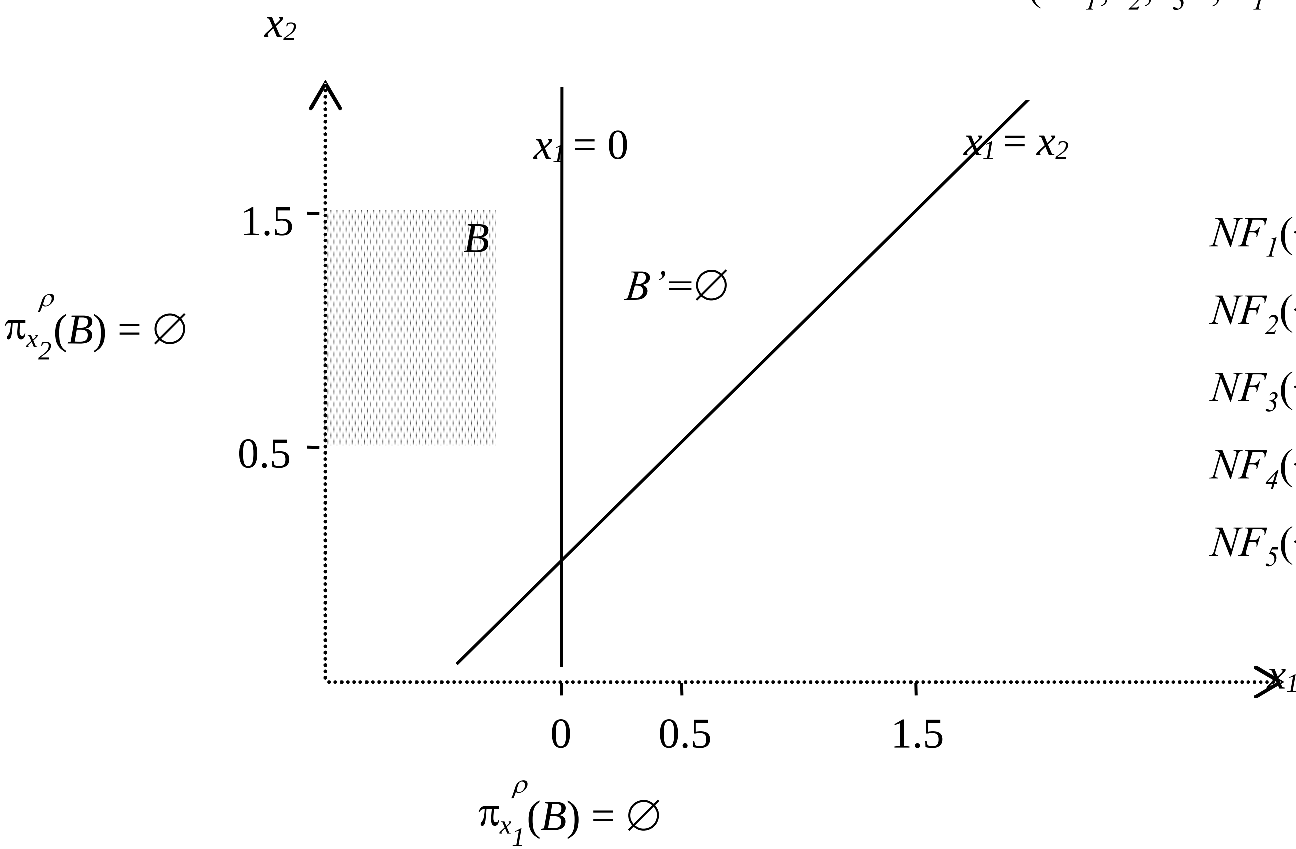
with $B = \langle [-1.0, 0.25], [0.5, 1.5] \rangle$ the best narrowing is also obtained: $B' = \langle [0.0, 0.0], [0.5, 1.5] \rangle$ and $x_3 = [0.5, 1.5]$

Constraint Decomposition Method

Example

$$P' = (\langle x_1, x_2, x_3 \rangle, D_1 \times D_2 \times [-\infty..+\infty], \{c_1, c_2\})$$

$$c \begin{cases} c_1 \equiv x_1 \times x_3 = 0 \\ c_2 \equiv x_2 - x_1 = x_3 \end{cases}$$



$$NF_1(\langle I_1, I_2, I_3 \rangle) = \langle (0/I_3) \cap I_1, I_2, I_3 \rangle$$

$$NF_2(\langle I_1, I_2, I_3 \rangle) = \langle I_1, I_2, (0/I_1) \cap I_3 \rangle$$

$$NF_3(\langle I_1, I_2, I_3 \rangle) = \langle (I_2 - I_3) \cap I_1, I_2, I_3 \rangle$$

$$NF_4(\langle I_1, I_2, I_3 \rangle) = \langle I_1, (I_3 + I_1) \cap I_2, I_3 \rangle$$

$$NF_5(\langle I_1, I_2, I_3 \rangle) = \langle I_1, I_2, (I_2 - I_1) \cap I_3 \rangle$$

with $B = \langle [-1.0, -0.25], [0.5, 1.5] \rangle$ inconsistency is proved:

$B' = \varnothing$

Constraint Newton Method

Complex constraints are handled without decomposition using a technique based on the interval Newton method for searching the zeros of univariate functions

Interval Projections

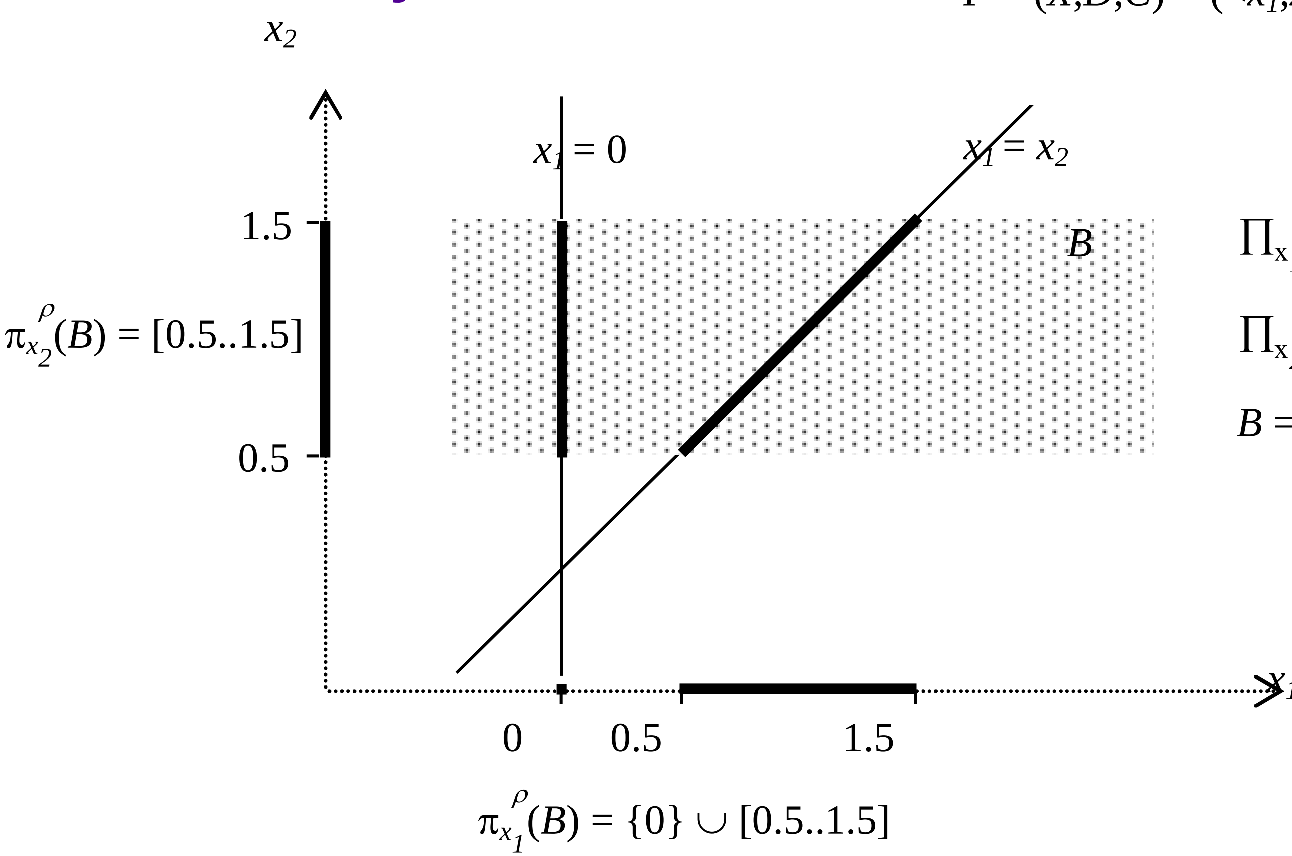
Interval Projection. Let P=(X,D,C) be a CCSP. Let c=(s,ρ)$\in C$ be an n-ary constraint expressed in the form $e_c \diamond 0$ (with $\diamond \in \{\leq,=,\geq\}$ and e_c a real expression). Let B be an n-ary F-box. The interval projection of c wrt $x_i \in s$ and B is the function, denoted $\prod_{x_i}^{\rho B}$, represented by the expression obtained by replacing in e_c each real variable x_j ($x_j \neq x_i$) by the interval constant $B[x_j]$. ❑

Constraint Newton Method

Interval Projections

$P = (X,D,C) = (<x_1,x_2>, D_1 \times D_2, \{c\})$

$c \equiv x_1 \times (x_2 - x_1) = 0$



$\prod_{x_1}^{\rho B} \equiv x_1 \times ([0.5..1.5] - x_1)$

$\prod_{x_2}^{\rho B} \equiv [-0.5..2.5] \times (x_2 - [-0.5..2.5])$

$B = <[-0.5..2.5],[0.5..1.5]>$

Constraint Newton Method

Properties of an Interval Projection

From the properties of the interval projections, a strategy is devised for obtaining an enclosure of the projection function

Properties of the Interval Projection. Let P=(X,D,C) be a CCSP. Let c=(s,ρ)$\in C$ be an n-ary constraint expressed in the form $e_c \diamond 0$ (with $\diamond \in \{\leq,=,\geq\}$ and e_c a real expression) and B an n-ary F-box. Let $\prod_{x_i}^{\rho B}$ be the interval projection of c wrt variable $x_i \in s$ and B. The following property is necessarily satisfied:

$$\forall_{r \in B[x_i]}\ r \in \pi_{x_i}^{\rho}(B) \Rightarrow \exists v \in \prod_{x_i}^{\rho B}([r]):\ v \diamond 0$$

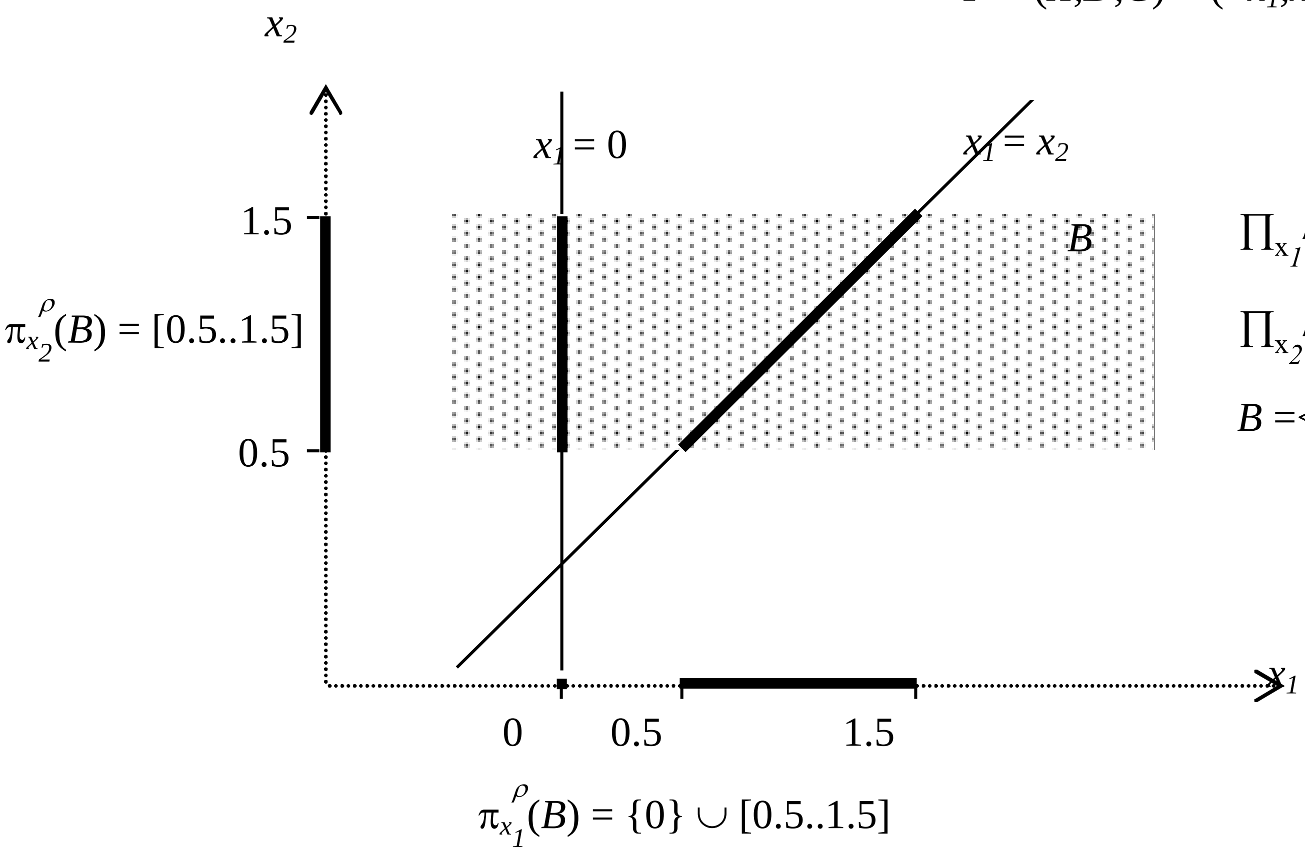
We will say that a real value r satisfies the interval projection condition if the right side of the implication is satisfied. ❑

Constraint Newton Method

Properties of an Interval Projection

$P = (X,D,C) = (<x_1,x_2>,D_1 \times D_2,\{c\})$

$c \equiv x_1 \times (x_2 - x_1) = 0$



$\prod_{x_1}^{\rho B} \equiv x_1 \times ([0.5..1.5] - x_1)$

$\prod_{x_2}^{\rho B} \equiv [-0.5..2.5] \times (x_2 - [-0.5..2.5])$

$B = <[-0.5..2.5],[0.5..1.5]>$

$\forall_{r \in [-0.5,2.5]}\ r \in \pi_{x_1}{}^{\rho}(B) \Rightarrow \exists v \in [r] \times ([0.5..1.5] - [r]):\ v = 0$

$\forall_{r \in [0.5,1.5]}\ r \in \pi_{x_2}{}^{\rho}(B) \Rightarrow \exists v \in [-0.5..2.5] \times ([r] - [-0.5..2.5]):\ v = 0$

Constraint Newton Method

Projection Function Enclosure with the Interval Projection

The strategy used in the constraint Newton method is to search for the leftmost and the rightmost elements of the original variable domain satisfying the interval projection condition

Projection Function Enclosure based on the Interval Projection. Let P=(X,D,C) be a CCSP. Let c=(s,ρ)$\in C$ be an n-ary constraint, B an n-ary F-box and x_i an element of s. Let a and b be respectively the leftmost and the rightmost elements of $B[x_i]$ satisfying the interval projection condition. The following property necessarily holds:

$$\pi_{x_i}^{\rho}(B) \subseteq [a..b]$$ ❑

What is needed is a function, denoted *narrowBounds*, with the following property:

$$\pi_{x_i}^{\rho}(B) \subseteq [a..b] \subseteq \mathit{narrowBounds}(B[x_i])$$

Constraint Newton Method

Projection Function Enclosure with the Interval Projection

To obtain a new bound, the projection condition is firstly verified in the extreme of the original domain and only in case of failure the leftmost (rightmost) zero of the interval projection is searched

function *narrowBounds*(an F-interval $[a..b]$)

  if $a = b$ **then if** *intervalProjCond*($[a]$) **then return** $[a]$ **else return** $\varnothing$; **end if**; **end if**;
  if not *intervalProjCond*($[a..a^+]$) **then** $a \leftarrow$ *searchLeft*($[a^+..b]$);
  if $a = \varnothing$ **then return** $\varnothing$;
  if $a = b$ **then return** $[b]$;
  if not *intervalProjCond*($[b^-..b]$) **then** $b \leftarrow$ *searchRight*($[a..b^-]$);
  return $[a..b]$;

end function

In case of failure of an inequality condition, it assumes that the leftmost (rightmost) element satisfying the interval projection condition must be a zero of the interval projection

Constraint Newton Method

Projection Function Enclosure with the Interval Projection

The verification if the interval projection condition is satisfied in a canonical interval is straightforward

function *intervalProjCond*(a canonical F-interval I)

(1) $[a..b] \leftarrow \prod_{x_i}^{\rho B}(I)$;

(2) **case** $\diamond$ **of**

(3) "$=$": **return** $0 \in [a..b]$;

(4) "$\leq$": **return** $a \leq 0$;

(5) "$\geq$": **return** $b \geq 0$;

(6) **end case**;

end function

Constraint Newton Method

Projection Function Enclosure with the Interval Projection

The algorithm for searching for the leftmost zero of an interval projection uses a Newton Narrowing function (*NN*) associated with the interval projection for reducing the search space

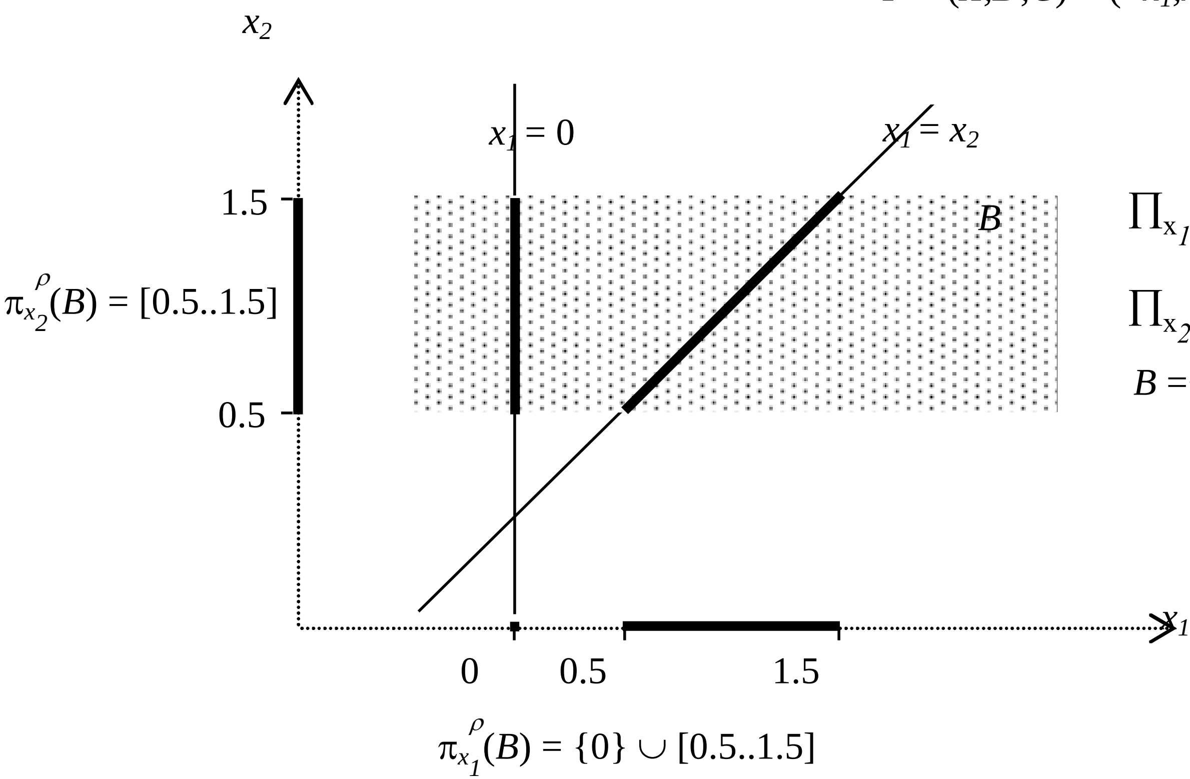
function *searchLeft*(an F-interval I)
  $Q \leftarrow \{I\}$;
  while $Q \neq \emptyset$ **do**
    choose $I_1 \in Q$ with the smallest left bound ($\forall_{I \in Q}\ left(I_1) \leq left(I)$);
    $Q \leftarrow Q \setminus \{I_1\}$;
    if $0 \in \prod_{x_i}^{\rho A}(I_1)$ **then**
      $I_1 \leftarrow NN(I_1)$;
      if $I_1 \neq \emptyset$ **then**
        $I_0 \leftarrow cleft(I_1)$; $I_1 \leftarrow [right(I_0)..right(I_1)]$;
        if $0 \in \prod_{x_i}^{\rho B}(I_0)$ **then return** $left(I_0)$;
        else $Q \leftarrow Q \cup \{[left(I_1)..\lfloor center(I_1)\rfloor], [\lfloor center(I_1)\rfloor..right(I_1)]\}$; **end if**;
      end if;
    end if;
  end while;
  return $\emptyset$;
end function

Constraint Newton Method

Example

$P = (X,D,C) = (<x_1,x_2>, D_1 \times D_2, \{c\})$

$c \equiv x_1 \times (x_2 - x_1) = 0$



$\prod_{x_1}{}^{\rho B} \equiv x_1 \times ([0.5..1.5] - x_1)$

$\prod_{x_2}{}^{\rho B} \equiv [-0.5..2.5] \times (x_2 - [-0.5..2.5])$

$B = <[-0.5..2.5],[0.5..1.5]>$

Narrowing the domain of variable x_1: narrowBounds([-0.5,2.5])

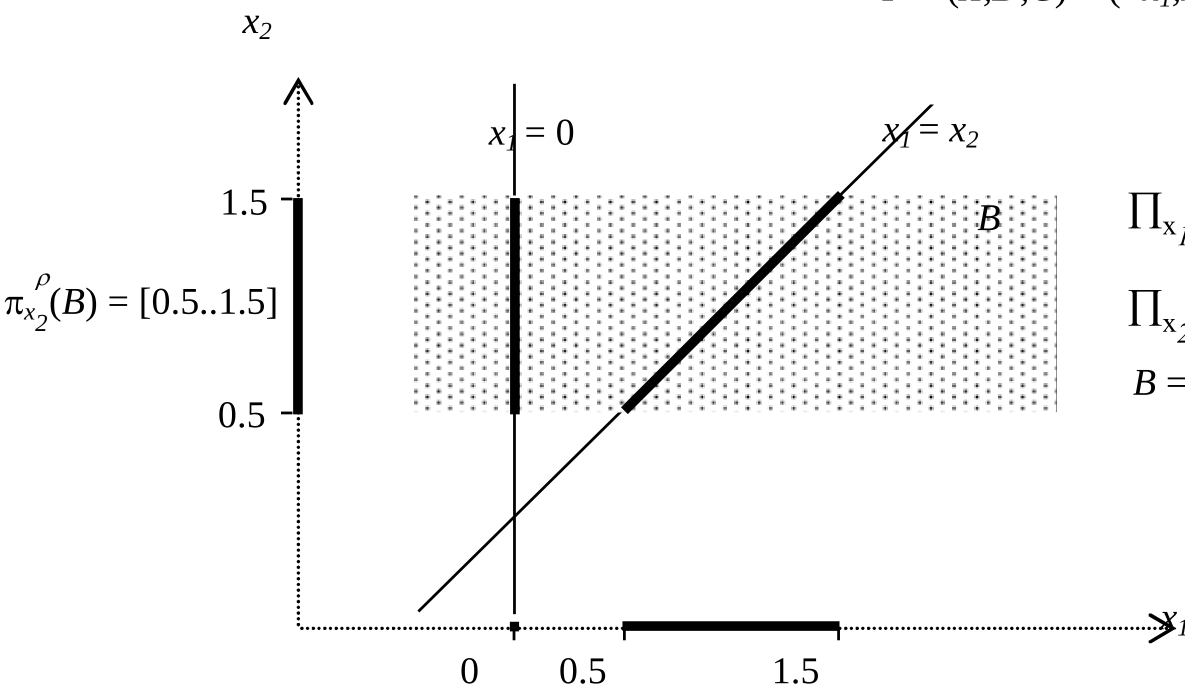
intervalProjCond([-0.5,-0.499]) $\rightarrow$ False $0 \notin \Pi_{x_1}{}^{\rho B}([-0.5,-0.499]) = [-1,-0.499]$

Constraint Newton Method

Example

$P = (X,D,C) = (\langle x_1,x_2\rangle, D_1 \times D_2, \{c\})$

$c \equiv x_1 \times (x_2 - x_1) = 0$



$\prod_{x_1}^{\rho B} \equiv x_1 \times ([0.5..1.5] - x_1)$

$\prod_{x_2}^{\rho B} \equiv [-0.5..2.5] \times (x_2 - [-0.5..2.5])$

$B = \langle [-0.5..2.5],[0.5..1.5] \rangle$

searchLeft([-0.499..2.5])			
$Q=\{I_1,\ldots,I_n\}$	$0 \in \prod_{x_i}^{\rho B}(I_1)$	$NN(I_1)$	$0 \in \prod_{x_i}^{\rho B}(I_0)$
{[-0.499..2.5]}	0∈[-5..4.998]	[-0.499..2.5]	0∉[-0.998.. -0.497]
{[-0.498..1.001],[1.001..2.5]}	0∈[-0.995..2]	[-0.498..1.001]	0∉[-0.997.. -0.496]
{[-0.497..0.252],[0.252..1.001],[1.001..2.5]}	0∈[-0.992..0.504]	[0..0.001]	0∈[0..0.002]

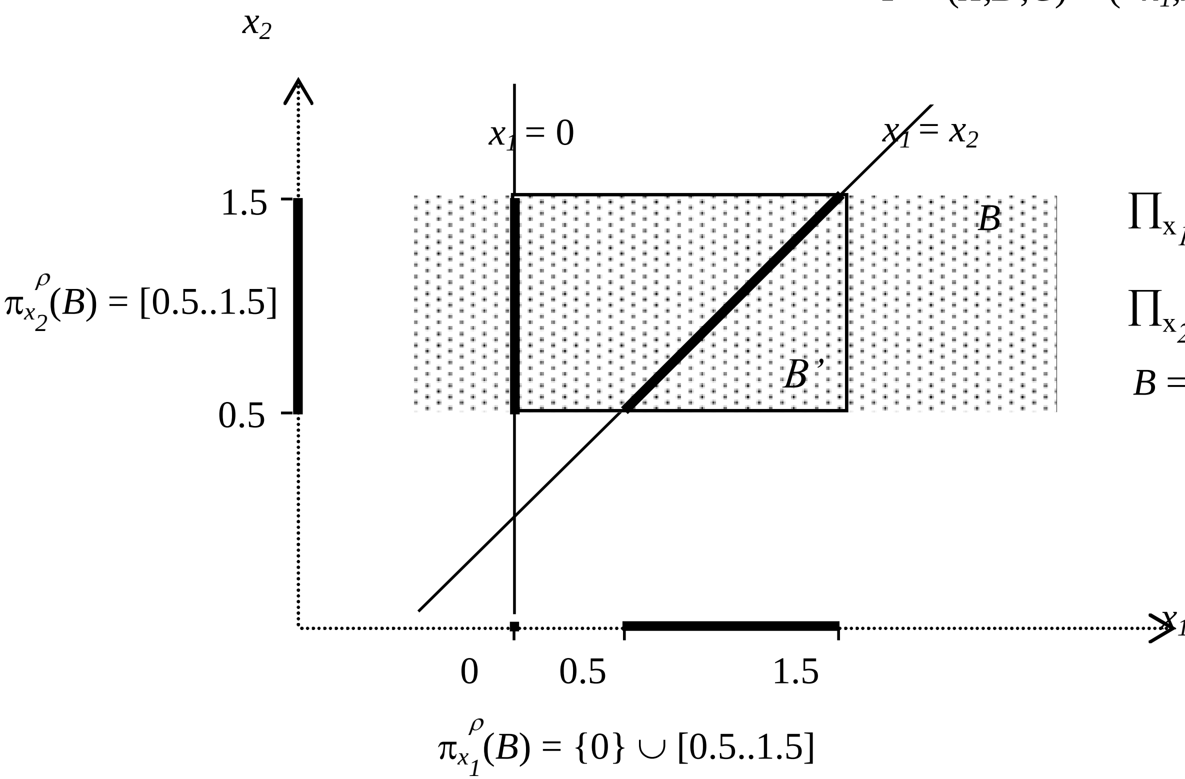
return 0

Constraint Newton Method

Example

$P = (X,D,C) = (\langle x_1,x_2\rangle, D_1\times D_2, \{c\})$

$c \equiv x_1\times(x_2-x_1) = 0$



$\prod_{x_1}^{\rho B} \equiv x_1\times([0.5..1.5]-x_1)$

$\prod_{x_2}^{\rho B} \equiv [-0.5..2.5]\times(x_2-[-0.5..2.5])$

$B = \langle[-0.5..2.5],[0.5..1.5]\rangle$

Proceeding similarly for the upper bound of x_1, the best narrowing is obtained:

$B' = \langle[0.0,1.501],[0.5,1.0]\rangle$

Revise Procedures

The revise procedures are algorithms that from one constraint contract concurrently the domains of all the variables of its scope

These are efficient implementations and variations of the generic methods presented before and are the basic building blocks of any continuous constraint solver library

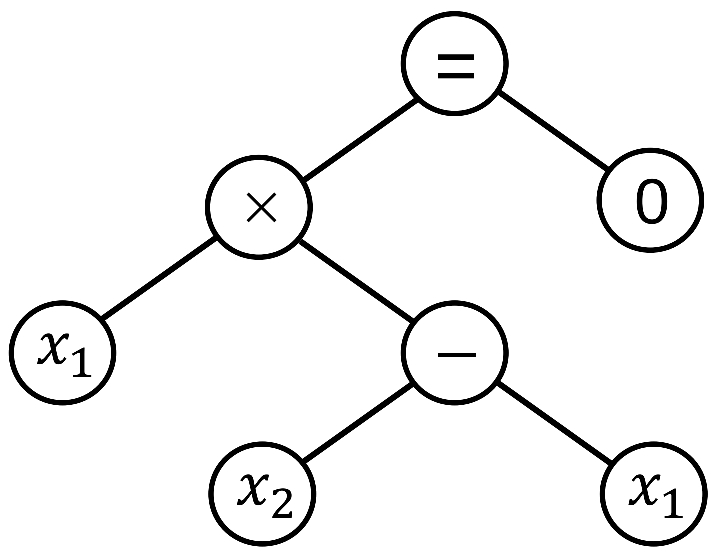
- HC3-Revise implements the decomposition method
- BC3-Revise implements the constraint Newton method
- HC4-Revise is an algorithm that obtains the results of the decomposition method considering the original constraint rather than primitives generated by decomposition
- Mohc-Revise is an algorithm that exploits monotonicity of functions to improve contraction using monotonic versions of HC4-Revise and BC3-Revise.

Revise Procedures

HC4-Revise

A constraint is represented by a tree where the root node contains the relation symbol, and terms are composed of nodes containing either a variable, a constant, or an operation symbol.

Example: $x_1 \times (x_2 - x_1) = 0$



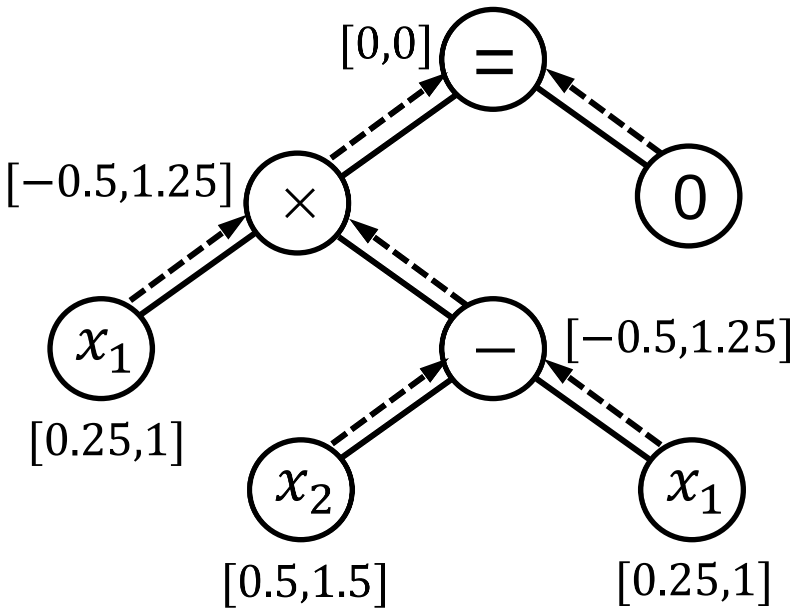
Revise Procedures

HC4-Revise

The algorithm proceeds in two consecutive stages:

1. Forward evaluation is a traversal of the terms from leaves to roots to evaluate the natural interval extension of every sub-term

Example: $x_1 \times (x_2 - x_1) = 0$ with: $B = [0.25,1] \times [0.5,1.5]$



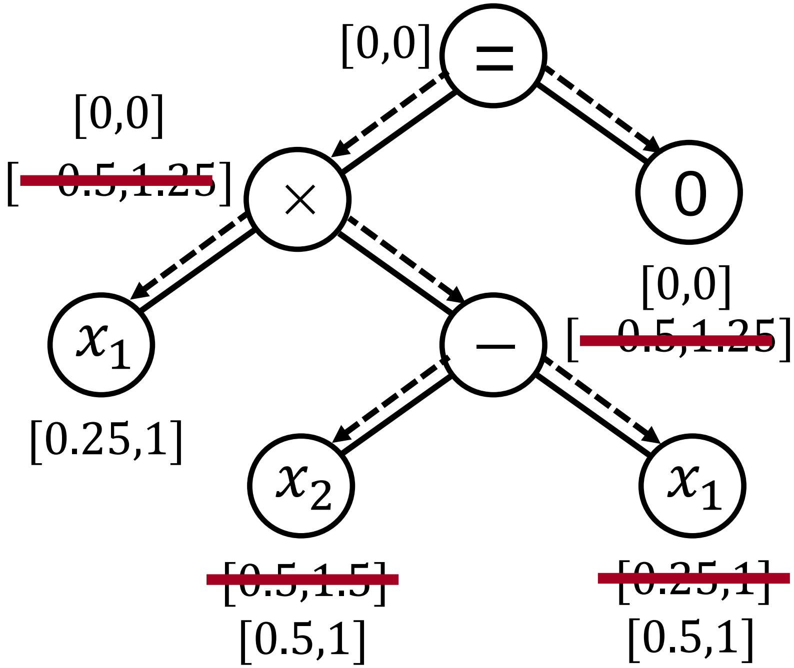
Revise Procedures

HC4-Revise

The algorithm proceeds in two consecutive stages:

2. Backward propagation is a traversal from root to leaves to evaluate at each node a projection narrowing operator associated its father

Example: $x_1 \times (x_2 - x_1) = 0$ with: $B = [0.25,1] \times [0.5,1.5]$



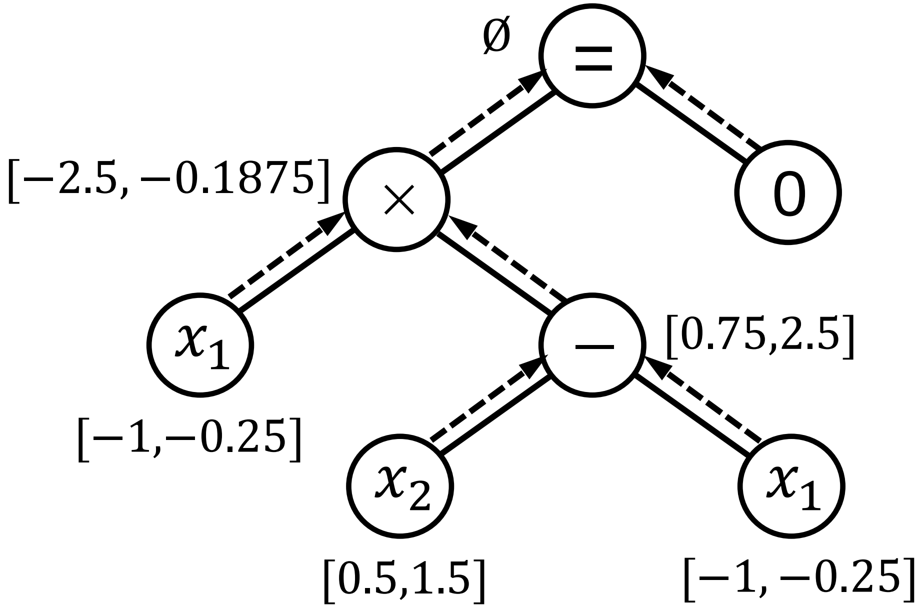
Revise Procedures

HC4-Revise

The algorithm proceeds in two consecutive stages:

1. Forward evaluation is a traversal of the terms from leaves to roots to evaluate the natural interval extension of every sub-term

Example: $x_1 \times (x_2 - x_1) = 0$ with: $B = [-1, -0.25] \times [0.5,1.5]$



Narrowing functions for systems of constraints

Instead of associating narrowing functions with each single constraint it is possible to associate it with multiple constraints

Multivariate Interval Newton

One possibility is the direct application of the interval Newton method to a square system of equations (see lecture3)

Acts like a global constraint that performs powerful contraction when the domains become small enough

Can prove rigorously the existence of a solution of a well constrained system of equations

Box-k Revise: The method is applied to a subsystems of k CCSP equations (to make it square)

Alternatively, variations of the method exist for non-square systems and for inequality constraints

Narrowing functions for systems of constraints

Some techniques linearize the nonlinear system and use the efficient linear algorithms (e.g. Simplex) to narrow the domains

Reformulation-Linearization

Is a technique developed for methods dedicated to quadratic non-convex problems

Each non linear term is replaced by a new variable and redundant linear constraints are introduced

Narrowing functions for systems of constraints

Some techniques linearize the nonlinear system and use the efficient linear algorithms (e.g. Simplex) to narrow the domains

Reformulation-Linearization

Example 1: relaxation of x^2 with $x \in [-4, 5]$

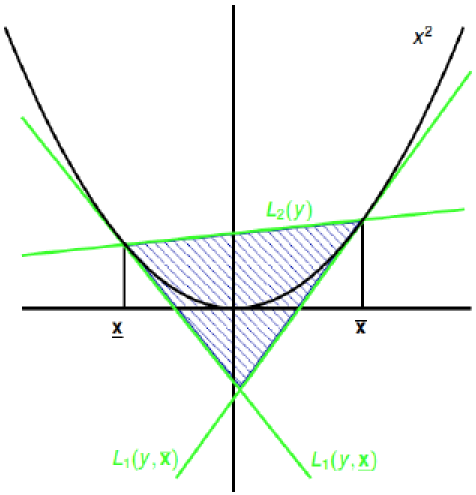
- $L_1(y, \alpha) \equiv y \geq 2\alpha x - \alpha^2$

  $L_1(y, -4) : y \geq -8x - 16$

  $L_1(y, 5) : y \geq 10x - 25$

- $L_2(y) \equiv y \leq (\underline{x} + \overline{x})x - \underline{x} * \overline{x}$

  $L_2(y) : y \leq x + 20$



Narrowing functions for systems of constraints

Some techniques linearize the nonlinear system and use the efficient linear algorithms (e.g. Simplex) to narrow the domains

Reformulation-Linearization

Is a technique developed for methods dedicated to quadratic non-convex problems

Each non linear term is replaced by a new variable and redundant linear constraints are introduced

Use of a Linear Programming algorithm to narrow the domain of each variable and update the coefficients of these linear constraints

Quad-algorithm: works on the relaxations of the nonlinear terms of the constraint system whereas a Box-consistency algorithm works on the initial constraint system

Narrowing functions for systems of constraints

Some techniques linearize the nonlinear system and use the efficient linear algorithms (e.g. Simplex) to narrow the domains

Linear relaxation based on affine arithmetic

Is a technique that uses affine arithmetic (an extension of interval arithmetic) to generate linear relaxations.

It produces a polytope by replacing in the constraint expressions every basic operator by specific affine forms

All real numbers are represented by an affine form $\widehat{x}$

$$\widehat{x} = x_0 + \sum_{i=1}^{n} x_i \varepsilon_i ,$$

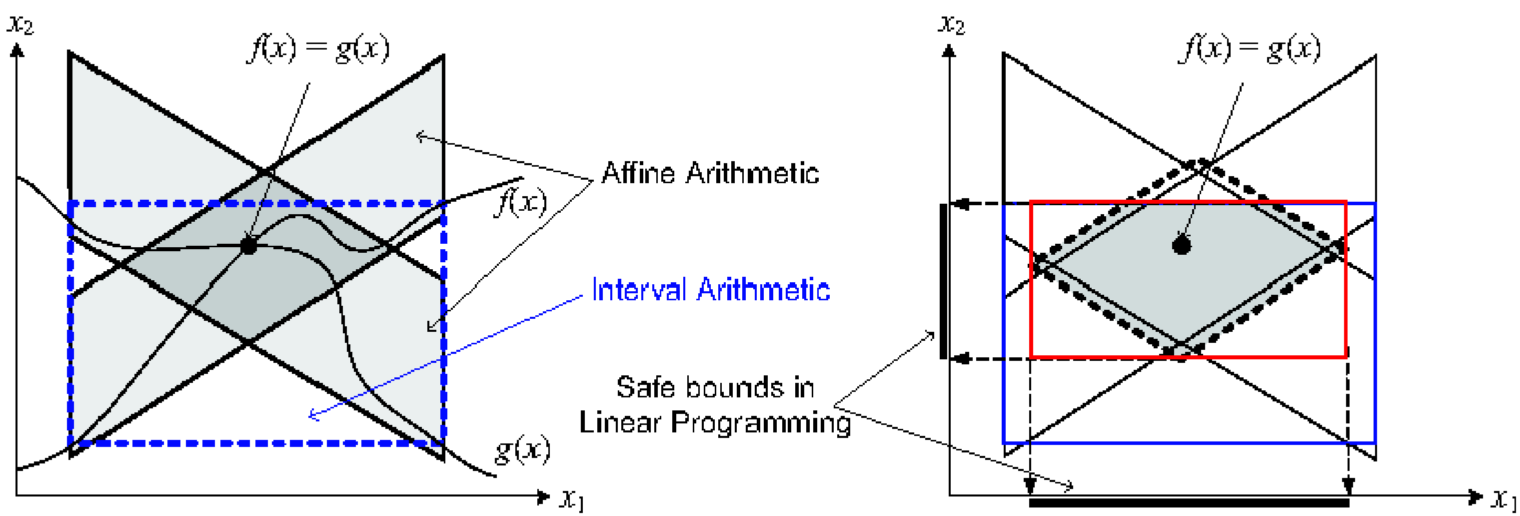
$$\textit{with } \forall i \in [1; n], x_i \in \mathbb{R} \textit{ and } \varepsilon_i = [-1, 1].$$

Narrowing functions for systems of constraints

Some techniques linearize the nonlinear system and use the efficient linear algorithms (e.g. Simplex) to narrow the domains

Linear relaxation based on affine arithmetic

Is a technique that uses affine arithmetic (an extension of interval arithmetic) to generate linear relaxations.



Narrowing functions for systems of constraints

Some techniques linearize the nonlinear system and use the efficient linear algorithms (e.g. Simplex) to narrow the domains

Linear relaxation based on affine arithmetic

Is a technique that uses affine arithmetic (an extension of interval arithmetic) to generate linear relaxations.

It produces a polytope by replacing in the constraint expressions every basic operator by specific affine forms

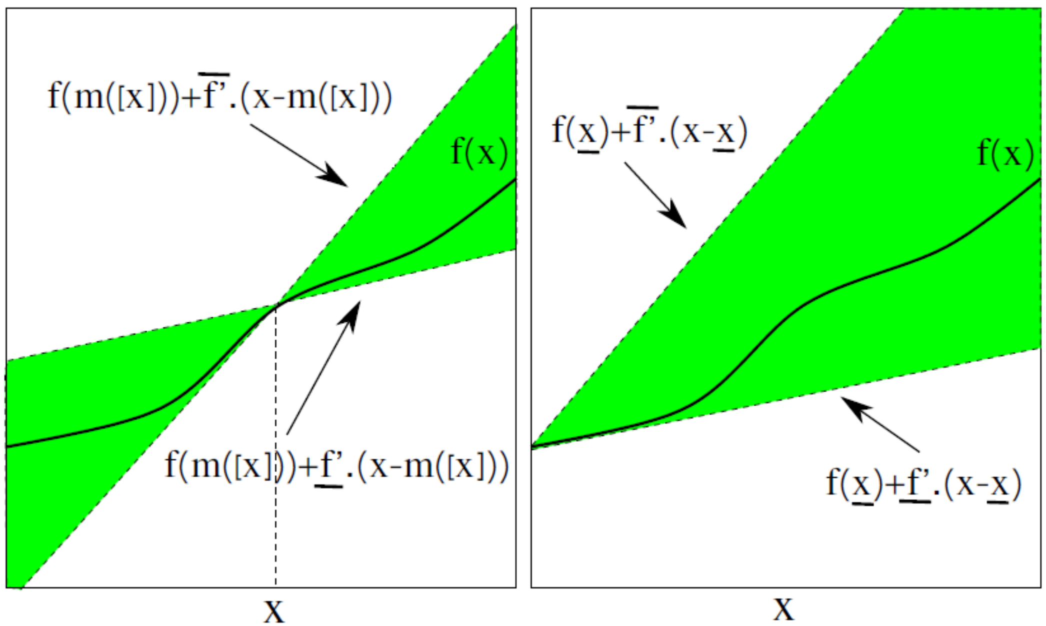
IBBA-algorithm: An interval Branch and Bound algorithm where the technique is integrated with other pruning methods

Narrowing functions for systems of constraints

Some techniques linearize the nonlinear system and use the efficient linear algorithms (e.g. Simplex) to narrow the domains

Corner-based Taylor relaxation

Is a technique that produces a polytope by selecting the two corners of the interval Taylor form instead of the usual midpoint



Narrowing functions for systems of constraints

Some techniques linearize the nonlinear system and use the efficient linear algorithms (e.g. Simplex) to narrow the domains

Corner-based Taylor relaxation

Is a technique that produces a polytope by selecting the two corners of the interval Taylor form instead of the usual midpoint

It uses two opposite corners of the domain for every constraint

Polytope-Hull: Is a contractor that with two calls to an LP solver computes the minimum and maximum values in this polytope for each of the variables

X-Newton: Is a contractor based on this technique that can treat well-constrained systems as well as under-constrained ones (with fewer equations than variables and with inequalities)