An Efficient and Safe Framework for Handling Numerical Constraint Systems and for Solving Optimization Problems

最適化問題を解決する為の安全なフレームワーク

Michel RUEHER

(Joint work with Yahia LEBBAH and Claude MICHEL)

COPRIN Project INRIA–I3S/CNRS
University of Nice – Sophia Antipolis

October 2004

Outline

- Motivations
- Safe use of linear relaxations : the ***QuadSolver*** experience
- Performance of the ***QuadSolver***
- A global optimisation framework
- First experimentations
- Conclusion

Motivations

➢ A constraint is handled as a **black-box** by local consistencies (2B–filtering,BOX–filtering)

- No way to catch the dependencies between constraints
- Splitting is behind the success for small dimensions

➢ Higher consistencies (KB–filtering,Bound–filtering)
$\longrightarrow$ visiting numerous combinations

QuadSolver

- ***safe and rigorous*** **linear relaxations**
- ***a global*** **constraint to handle a tight linear approximation of the constraint system (Simplex)**
- ***local*** **consistencies (2B, Box) and** ***interval*** **methods (Newton)**

The Quad-filtering process

◇ Reformulation

➢ capture the linear part of the problem
→ replace each non linear term by a new variable (eg x^2 by y_i)

◇ Linearisation/relaxation

➢ introduce ***redundant linear constraints***
→ tight approximations of the non-linear terms (RLT)

◇ Computing $\boldsymbol{min(x)} = \underline{x_i}$ and $\boldsymbol{max(x)} = \overline{x_i}$ in LP

Linearisation of x^2

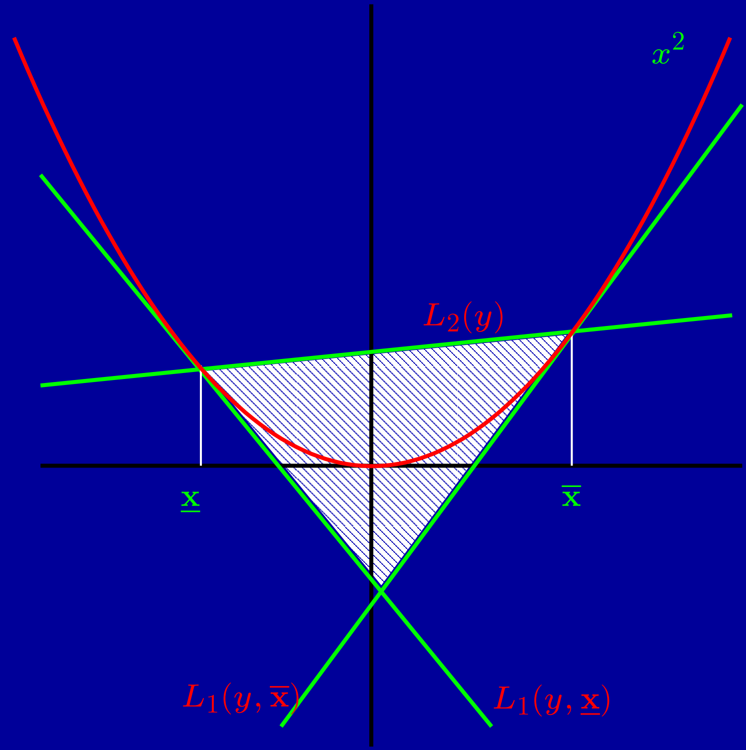
Example : relaxation of $y = x^2$ with $\mathbf{x} = [-4, 5]$

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

$L_1(-4) : y \geq -8x - 16$
$L_1(5) : y \geq 10x - 25$

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

$L_2 : y \leq x + 20$



Linearisation of x^2

- $f(x) = x^2$ with $\underline{x} \leq x \leq \overline{x}$ is approximated by :

$$L_1(\alpha) \equiv [(x-\alpha)^2 \geq 0]_l \text{ where } \alpha \in [\underline{x}, \overline{x}] \quad (1)$$

$$L_2 \equiv (\underline{x} + \overline{x})x - y - \underline{x} * \overline{x} \geq 0 \quad (2)$$

- $[(x-\alpha_i)^2 = 0]_l$ generates the tangents to $y = x^2$ at $x = \alpha_i$ (***QuadSolver*** only computes $L_1(\overline{x})$ and $L_1(\underline{x})$)

- $L_1(\overline{x})$ and $L_1(\underline{x})$: underestimations of y
  L_2 : overestimation of y

The Quad filtering algorithm

Function `Quad_filtering(IN:` $\mathcal{X}$, $\mathcal{D}$, $\mathcal{C}$, ϵ) **return** $\mathcal{D}'$

1. ***Reformulation***
   $\longrightarrow$ linear inequalities $[\mathcal{C}]_R$ for the nonlinear terms in $\mathcal{C}$

2. ***Linearisation/relaxation of the whole system*** $[\mathcal{C}]_L$
   $\longrightarrow$ a linear system $LR = [\mathcal{C}]_L \cup [\mathcal{C}]_R$

3. $\mathcal{D}' := \mathcal{D}$

4. ***Pruning***:
   While amount of reduction of some bound $> \epsilon$ **and** $\emptyset \notin \mathcal{D}'$ **Do**

   (a) Update the coefficients of $[\mathcal{C}]_R$ according to $\mathcal{D}'$
   (b) Reduce the lower and upper bounds $\underline{x}'_i$ and $\overline{x}'_i$ of each ***initial*** variable $x_i \in \mathcal{X}$ (computing min and max of x_i subject to LR with a LP solver)

Issues in the use of linear relaxation

◇ Coefficients of linear relaxations are scalars

➔ computed with ***floating point numbers***

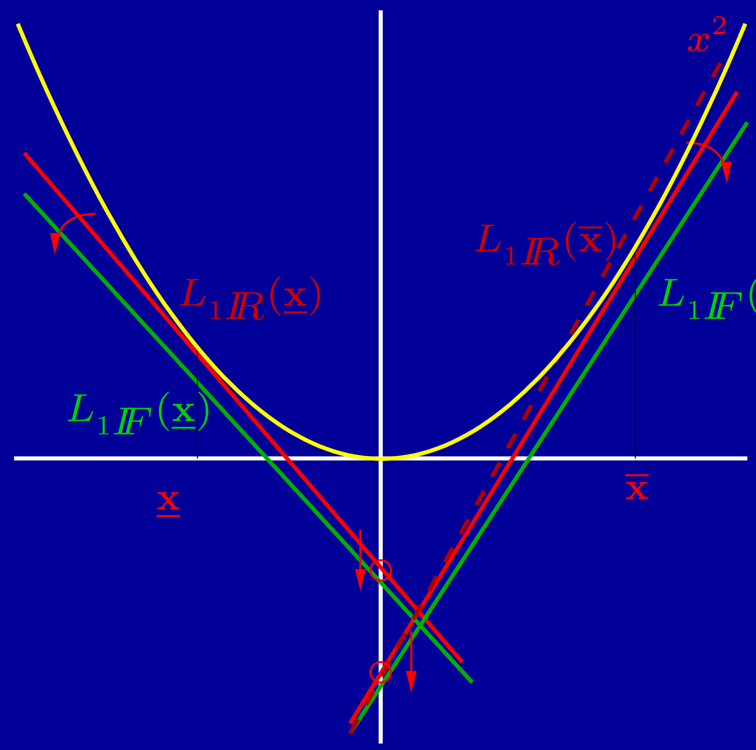
◇ Efficient implementations of the simplex algorithm

➔ use ***floating point numbers***

➢ All the computations with floating point numbers require ***right corrections***

Safe approximations of L_1

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



Effects of rounding:

◇ rounding of 2α

➔ rotations on $L_1(\alpha)$

◇ rounding of α^2

➔ translation on y axis

➢ **intersection with x^2**

Safe approximations of L_1

$L_{1\mathbb{F}}(\alpha)$ approximations
Let $\alpha \in \mathbb{F}$ and

$$L_{1\mathbb{F}}(\alpha) \equiv \begin{cases} y - \lfloor 2\alpha \rfloor x + \lceil \alpha^2 \rceil \geq 0 & \text{iff } \alpha \geq 0 \\ y - \lceil 2\alpha \rceil x + \lceil \alpha^2 \rceil \geq 0 & \text{iff } \alpha < 0 \end{cases}$$

$\forall x \in \mathbf{x}$, and $y \in [0, max\{\underline{\mathbf{x}}^2, \overline{\mathbf{x}}^2\}]$,
if $L_1(\alpha)$ holds, then $L_{1\mathbb{F}}(\alpha)$ holds too

Generalisation to n-ary linearisations

Let $\sum_{i=1}^{n} a_i x_i + b \geq 0$

then $\forall x_i \in \mathbf{x}_i$:

$$\sum_{i=1}^{n} \overline{a}_i x_i + \sup(\overline{b} + \sum_{i=1}^{n} \sup(\sup(\mathbf{a}_i \underline{x}_i) - \overline{a}_i \underline{x}_i)) \geq \sum_{i=1}^{n} a_i x_i + b \geq 0$$

Extensions :

- Borodaile and Van Hentenryck
- Hongthong and Kearfott

Correction of the Simplex algorithm

Consider the following LP :

$$\text{minimise } c^T x$$

$$\text{subject to } \underline{\mathbf{b}} \leq Ax \leq \overline{\mathbf{b}}$$

- Solution = vector $x_{\mathbb{R}} \in \mathbb{R}^n$
- CPLEX computes a vector $x_{\mathbb{F}} \in \mathbb{F}^n \neq x_{\mathbb{R}}$.
- $x_{\mathbb{F}}$ is safe for the objective if $c^T x_{\mathbb{R}} \geq c^T x_{\mathbb{F}}$.

➢ Neumaier and Shcherbina

→ ***cheap method to obtain a rigorous bound of the objective***

→ ***rigorous computation of the certificate of infeasibility***

Performance of the *QuadSolver*

			QuadSolver		IlogSolver (Box)		Realpaver
Name	n	δ	$Ksplits$	$T(s)$	$Ksplits$	$T(s)$	$T(s)$
assur44	8	3	**0.1**	**49.5**	15.8	72.5	72.6
katsura5	6	2	**0.1**	9.9	8.2	12.7	**6.7**
katsura6	7	2	**0.5**	**121.9**	136.6	281.4	191.8
kin2	8	2	**0.0**	6.2	3.5	19.3	**2.6**
tangents2	6	2	**0.1**	17.5	**14.1**	27.9	16.5
camera1s	6	2	**1.0**	**28.9**	11820.3	—	—
didrit	9	2	**0.1**	**14.7**	51.3	132.9	94.6
geneig	6	3	**0.8**	**39.1**	290.7	868.6	475.6
kinema	9	2	**0.2**	**19.9**	244.0	572.4	268.4
katsura7	8	2	**1.7**	**686.9**	1858.5	11104.1	4671.1
lee	9	2	**0.5**	**43.3**	8286.3	—	—
reimer5	5	6	**0.1**	**53.0**	2230.2	2892.5	733.9
stewgou40	9	4	**1.6**	**924.0**	3128.6	—	—
yama194	16	3	**0.0**	**11.1**	1842.1	—	—
yama195	60	3	**0.0**	**106.1**	19.6	—	—
yama196	30	1	**0.0**	**6.7**	816.7	—	—

A global optimisation framework

We consider the continuous global optimisation problem $\mathcal{P}$

$$\begin{array}{ll} \textbf{minimise} & f(X) \\ \textbf{subject to} & g_i(X) = 0, \quad i = 1..k \\ & g_j(X) \leq 0, \quad j = k+1..m \end{array} \qquad \textbf{(3)}$$

with $X \in \mathbf{X}$; $f : I\!R^n \rightarrow I\!R$ and $g_{1\ldots m} : I\!R^n \rightarrow I\!R$.
Functions f and $g_{1\ldots m}$ are continuously differentiable on $\mathbf{X}$,
where $\mathbf{X}$ denotes a vector of intervals of $I\!R$.

Trends in global optimisation

◇ **Performance**

Most successful systems (Baron, αBB, . . .) use linear relaxations
$\longrightarrow$ complete methods, but **not rigorous**

◇ **Rigour**

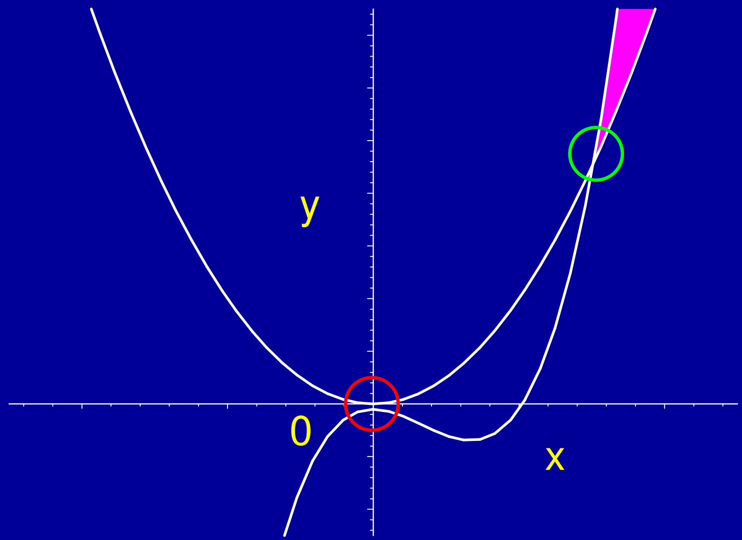
Mainly rely on interval computation
. . . available systems (e.g., Globsol) are **rather slow**

➢ **Challenge:** to combine the advantages of both approaches in an efficient and **rigorous** global optimisation framework

Example of flaw due to a lack of rigour

Consider the following optimisation problem:

$$\begin{array}{ll} \text{min} & x \\ \text{s. t.} & y - x^2 \geq 0 \\ & y - x^2 * (x - 2) + 10^{-5} \leq 0 \\ & x, y \in [-10, +10] \end{array}$$

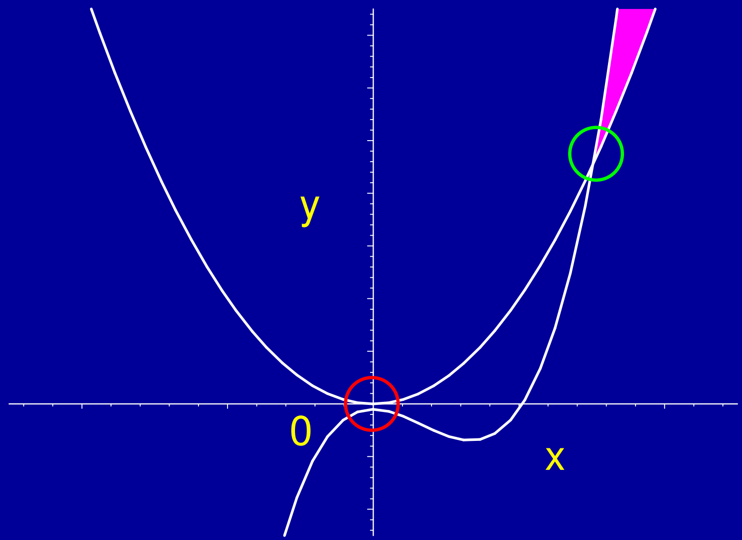


`Baron 7.2` finds **0** as the minimum . . .

Example of flaw due to a lack of rigour

Consider the following optimisation problem:

$$\begin{array}{ll} \text{min} & x \\ \text{s. t.} & y - x^2 \geq 0 \\ & y - x^2 * (x-2) + 10^{-5} \leq 0 \\ & x, y \in [-10, +10] \end{array}$$



`Baron 7.2` finds $\mathbf{0}$ as the minimum . . .

`QuadOpt :`

$\longrightarrow$ lower bound of the objective function is $\mathbf{2.96}$

$\longrightarrow$ upper bound of the objective at the feasible point $\mathbf{(3, 9)}$ is $\mathbf{3.00}$

From *QuadSolver* to global optimisation

QuadSolver offers a safe and efficient framework to solve non-linear constraint systems

- Switch to global optimisation

  → ***Add constraint*** $\mathbf{f}^* = f(X)$

- Key of success

  → ***Adapt the search tree***

General schema of the QuadOpt solver

Algorithm QuadOpt(IN $\mathcal{P}$, ϵ; OUT $\mathcal{S}$, $\mathbf{f}^*$)

Use ***QuadSolver*** to reduce($\mathcal{D}$, $\mathbf{f}^*$)
$\mathcal{L} := \mathcal{D} \quad ; \quad S := \emptyset$
while $w(\mathbf{f}^*) > \epsilon$ **do**
    $LB :=$ **LowerBound**($\mathcal{P}$, $[\underline{f^*}, m(\mathbf{f}^*)]$)
    $(UB, F_p) :=$ **UpperBox**($\mathcal{P}$, $[LB, m(\mathbf{f}^*)]$, $\mathcal{L}$)
    if $[LB, UB] = \emptyset$
        then $\underline{f^*} := m(\mathbf{f}^*) \quad ; \quad \mathcal{L} := \mathcal{D}$
        else $\mathbf{f}^* := [LB, UB] \quad ; \quad \mathcal{S} := \mathcal{S} \cup \{F_p\}$
    endif
endwhile

Computing the lower bound

- use of ***QuadSolver***

  → **safe bound**

- ***"light" version of QuadSolver***

  → **linear relaxations (RLT) +** $2B$**-consistency**

  Applied to $f(X) \wedge g_{1..m}(X)$

Computing the upper box

➢ ***to find a box that contains at least one solution***

- **local search**: interior point algorithm (`COIN-IPOPT`), sequential quadratic programming
  → **to find a "candidate" solution quickly**

- **existence proof:** Hansen's heuristic (separation of active and inactive constraints to obtain a square linear system, Gauss Seidel iteration to test feasibility)

- ***specific search tree process***
  → **to update "promising" boxes**

Computing the upper box (2)

Algorithm `UpperBox` (`IN:` $\mathcal{P}$, $\mathbf{D}_{Obj}$; `INOUT:`$\mathcal{L}$; `OUT:` $(UB, \mathcal{S})$)
$S := \emptyset$; MayBe := False;
while $S = \emptyset \wedge \mathcal{L} \neq \emptyset$ **do**
  select and remove some $\mathbf{D}'$ from $\mathcal{L}$
  $\mathbf{D}'' := Prune(\mathbf{D}')$
  if $\mathbf{D}'' \neq \emptyset$ **then**
    if $w(\mathbf{D}'') < \epsilon$ **then** ProveFeasible($\mathbf{D}''$, S,MayBe);
    else if LocalFind($\mathbf{D}''$, $\mathcal{L}_{\mathcal{FP}}$) **then** ProveFeasible($\mathcal{L}_{\mathcal{FP}}$, $\mathcal{S}$,MayBe)
      else split$(\mathbf{D}'', \{\mathbf{D}_1, \mathbf{D}_2\})$; $\mathcal{L} := \mathcal{L} \cup \{\mathbf{D}_1, \mathbf{D}_2\}$; **endif**
    endif
  endif
endwhile
if $S = \emptyset$ **then if** MayBe **then return** $(\overline{\mathbf{D}_{Obj}}, \emptyset)$ **else return** $(-\infty, \emptyset)$
else return $(\overline{\mathbf{f}(S)}, S)$ **endif**

Experimentations (1)

		QuadOpt		GLOBSOL	
Name	(n, m)	*Safe*	$T(s)$	*Safe*	$T(s)$
TP16	(2,2)	$*$	0.03	$*$	0.03
TP220	(2,1)	$*$	**0.02**	$*$	0.06
TP265	(4,2)	$*$	**0.03**	$-$	8.51
TP33	(3,2)	$*$	0.1	$*$	**0.08**
TP54	(6,1)	$*$	0.7	$*$	**0.47**
TP55	(6,6)	$*$	**0.49**	$-$	1.64
Murtagh	(5,3)	$*$	5.35	$*$	**4.69**
Audet140a	(5,4)	$*$	**0.24**	$*$	4.50
Audet140b	(4,2)	$*$	0.18	$*$	0.18
Audet141	(6,4)	$*$	**0.55**	$*$	2.52
Audet145	(7,8)	$-$	**30.65**	$*$	48.57

Experimentations (2)

		QuadOpt		BARON	
Name	(n, m)	*Safe*	$T(s)$	*Safe*	$T(s)$
TP16	(2,2)	$*$	0.03	?	0.02
TP220	(2,1)	$*$	0.02	?	0.00
TP265	(4,2)	$*$	0.03	?	0.02
TP33	(3,2)	$*$	0.1	?	0.03
TP54	(6,1)	$*$	0.7	?	0.04
TP55	(6,6)	$*$	0.49	?	0.02
Murtagh	(5,3)	$*$	5.35	?	0.39
Audet140a	(5,4)	$*$	0.24	?	0.06
Audet140b	(4,2)	$*$	0.18	?	0.04
Audet141	(6,4)	$*$	0.55	?	0.12
Audet145	(7,8)	$-$	30.65	?	0.10

Conclusion and future works

- ***Contribution:*** **a new safe and efficient framework**
- ***Future works***

  … ***improve performances***