Constraint Programming and Safe Global Optimization

Michel RUEHER

Joint work with Alexandre GOLDSZTEJN, Yahia LEBBAH and Claude MICHEL

Université de Nice Sophia-Antipolis / CNRS, France

SCAN 08, October 2008

Outline

Motivations

Basics on CSP

Using CSP to boost safe OBR

Computing “sharp” upper bounds

Conclusion

The Problem

We consider the continuous global optimisation problem

$$\mathcal{P} \equiv \begin{cases} \min & f(x) \\ \text{s.c.} & g_i(x) = 0,\ j = 1..k \\ & g_j(x) \leq 0,\ j = k+1..m \\ & \underline{\mathbf{b}} \leq x \leq \overline{\mathbf{b}} \end{cases} \tag{1}$$

with

- $\mathbf{B} = [\underline{\mathbf{b}}, \overline{\mathbf{b}}]$: a vector of intervals of R
- $f : R^n \rightarrow R$ and $g_j : R^n \rightarrow R$
- Functions f and g_j: are continuously differentiable on $\mathbf{B}$

Trends in global optimisation

- **Performance**

  Most successful systems (Baron, αBB, ...) use local methods and linear relaxations
  $\rightarrow$ **not rigorous** (work with floats)

- **Rigour**

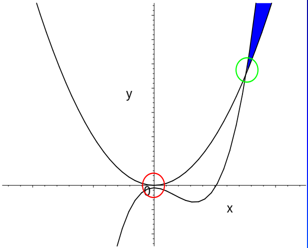
  Mainly rely on interval computation
  ... available systems (e.g., Globsol) are **quite slow**

- **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:

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



Baron 6.0 and Baron 7.2 find 0 as the minimum ...

Basics on CSP

Numeric CSP

- $\mathcal{X} = \{x_1, \ldots, x_n\}$ is a set of variables
- $\mathcal{D} = \{D_{x_1}, \ldots, D_{x_n}\}$ is a set of domains
  (D_{x_i} contains all acceptable values for variable x_i)

  $D_{x_i} = [\underline{\mathbf{x}_i}, \overline{\mathbf{x}_i}]$
- $\mathcal{C} = \{c_1, \ldots, c_m\}$ is a set of constraints

CSP: Overall scheme

A Branch & Prune schema:

1. **Pruning the search space**
2. **Making a choice to generate two (or more) sub-problems**

- The pruning step → **filtering techniques** to reduce the size of the intervals
- The branching step → **splits the intervals** (uses heuristics to choose the variable to split)

Local consistencies (1)

A constraint system $\mathcal{C}$ satisfies a local consistency property if each constraint C holds in **a relaxation of $\mathcal{C}$**

Consider $\mathbf{X} = [\underline{\mathbf{x}}, \overline{\mathbf{x}}]$ and $C(x, x_1, \ldots, x_n) \in \mathcal{C}$:
if $C(x, x_1, \ldots, x_n)$ does not hold for any values $a \in [\underline{\mathbf{x}}, \mathbf{x}']$,
then $\mathbf{X}$ can be shrinked to $\mathbf{X} = [\mathbf{x}', \overline{\mathbf{x}}]$

Local consistencies (2)

- **2B–consistency** only requires to check the Arc–Consistency property **for each bound** of the intervals

  Variable x with $\mathbf{X} = [\underline{\mathbf{x}}, \overline{\mathbf{x}}]$ is 2B–consistent for constraint $f(x, x_1, \ldots, x_n) = 0$ if $\underline{\mathbf{x}}$ and $\overline{\mathbf{x}}$ are the leftmost and the rightmost zero of $f(x, x_1, \ldots, x_n)$

- **Box–consistency** :
  - $\rightarrow$ coarser relaxation of AC than 2B–consistency
  - $\rightarrow$ **better filtering**

  Variable x with $\mathbf{X} = [\underline{\mathbf{x}}, \overline{\mathbf{x}}]$ is Box–Consistent for constraint $f(x, x_1, \ldots, x_n) = 0$ if $\underline{\mathbf{x}}$ and $\overline{\mathbf{x}}$ are the leftmost and the rightmost zero of $\mathbf{F}(\mathbf{X}, \mathbf{X_1}, \ldots, \mathbf{X_n})$, the optimal interval extension of $f(x, x_1, \ldots, x_n)$

Filtering

- **2B–filtering Algorithms** $\rightsquigarrow$ **projection functions**
  - → considers that each occurrence is a different new variable
  - → initial constraints are decomposed into "primitive" constraints: amplifies the dependency problem
- **Box–filtering Algorithms** $\rightsquigarrow$ **monovariate version of the interval Newton method**
  - → Monovariate constraints: substituting intervals for all variables but one
  - → Computing bounds: **dichotomy algorithm** (combined with the interval Newton method)

Limits of Local Consistencies

- **A constraint is handled as a black-box** by local consistencies (2B,BOX,...)
  - No way to catch the dependencies between constraints
  - Splitting is behind the success for small dimensions

- **Higher consistencies** (KB–filtering,Bound–filtering)
  $\longrightarrow$ capture some dependencies between constraints
  $\longrightarrow$ visiting numerous combinations

- A **global constraint** to handle a **linear approximation** with LP solvers
  $\longrightarrow$ safe linear relaxations

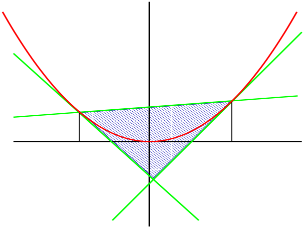
Safe use of Linear Relaxation

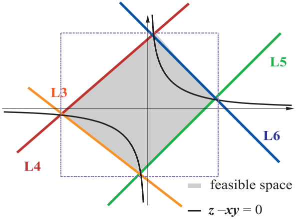
- A **global constraint** to handle an **approximation** of the constraint system with LP

- QUAD_SOLVER
  → **Global constraint** on linear relaxations
  → **local consistencies** (2B, Box) and **interval methods** (Newton)

The QUAD_SOLVER framework

- **Reformulation**
  - capture the linear part
    → replace non linear terms by new variable
    eg x^2 by y_i
- **Linearisation**
  - introduce **redundant linear constraints**
    → tight approximations (RLT)
- **Computing min(X) $= \underline{\mathbf{x}_i}$ and max(X) $= \overline{\mathbf{x}_i}$ in *LP***





Quad_Solver filtering algorithm

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

1. ***Reformulation***
   $\rightarrow$ linear inequalities $[\mathcal{C}]_R$ for the nonlinear terms in $\mathcal{C}$
2. ***Linearisation/relaxation of the whole system*** $[\mathcal{C}]_L$
   $\rightarrow$ 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**
   4.1 Update the coefficients of $[\mathcal{C}]_R$ according to $\mathcal{D}'$
   4.2 Reduce the lower and upper bounds $\underline{\mathbf{x}}'_i$ and $\overline{\mathbf{x}}'_i$ of each ***initial*** variable $x_i \in \mathcal{X}$ computing min and max of $\mathbf{X_i}$ subject to LR with a LP solver

Issues in the use of linear relaxation

- Coefficients of linear relaxations are scalars
  $\Rightarrow$ computed with ***floating point numbers***

- Efficient implementations of the simplex algorithm
  $\Rightarrow$ use ***floating point numbers***

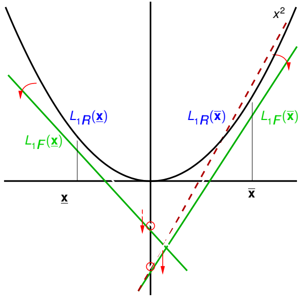
- All the computations with floating point numbers require ***right corrections***

Safe approximations of L_1

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

Effects of rounding:

- rounding of 2α
  - $\Rightarrow$ rotation on y axis
- rounding of α^2
  - $\Rightarrow$ translation on y axis



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 $x_R \in R^n$
- LP solver computes a vector $x_F \in F^n \neq x_R$
- x_F is safe for the objective if $c^T x_R \geq c^T x_F$
- **Neumaier & Shcherbina**

  $\rightarrow$ cheap method to obtain a **rigorous bound** of the objective
  (use of the approximation solution of the dual)

Using constraint techniques to boost safe OBR (optimal based reduction)

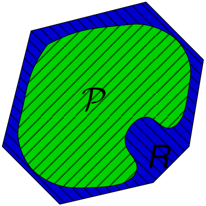
Branch and bound

- **BB Algorithm:**
  While $\mathcal{L} \neq \emptyset$ do %$\mathcal{L}$ initialized with the input box
  - Select a box B from the set of current boxes $\mathcal{L}$
  - Reduction (filtering or tightening) of B
  - Lower bounding of f in box B
  - Upper bounding of f in box B
  - Splitting of B (if not empty)
- **Upper Bounding – Critical issue:**
  to prove the **existence** of a feasible point in a reduced box
- **Lower Bounding – Critical issue:**
  to achieve an **efficient pruning**

Lower bounding

- **Relaxing the problem**
  - linear relaxation R of $\mathcal{P}$

$$\begin{array}{ll} min & d^T x \\ s.t. & Ax \leq b \end{array} \qquad (2)$$



  - LP solver $\rightarrow \underline{\mathbf{f}}^*$

$\rightarrow$ numerous splitting

- **OBR is a way to speed up the reduction process**

Optimality Base Reduction

- Introduced by **Ryoo and Sahinidis**
  - to take advantage of the **known bounds of the objective function** to reduce the size of the domains
  - uses a well known property of the **saddle point** to compute new bounds for the domains taking into account the known bounds of the objective function

Theorems of OBR

- Let $[L, U]$ be the domain of f. If the constraint $x_i - \overline{\mathbf{x}}_i \leq 0$ is active at the optimal solution of R and has a corresponding multiplier $\lambda_i^* > 0$ (λ^* is the optimal solution of the dual of R), then

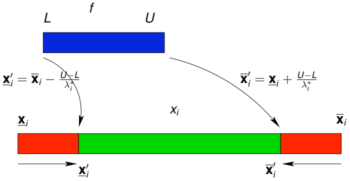
$$x_i \geq \underline{\mathbf{x}}'_i \text{ with } \underline{\mathbf{x}}'_i = \overline{\mathbf{x}}_i - \frac{U - L}{\lambda_i^*}. \qquad (3)$$

  if $\underline{\mathbf{x}}'_i > \underline{\mathbf{x}}_i$, the domain of x_i can be shrinked to $[\underline{\mathbf{x}}'_i, \overline{\mathbf{x}}_i]$
  without loss of any global optima

- similar theorems for $\underline{\mathbf{x}}_i - x_i \leq 0$ and $g_i(x) \leq 0$.

OBR: intuitions

- Ryoo & Sahinidis 96



$$x_i \geq \underline{\mathbf{x}}_i' \text{ with } \underline{\mathbf{x}}_i' = \overline{\mathbf{x}}_i - \frac{U-L}{\lambda_i^*} \qquad (4)$$

- **does not modify the very branch and bound process**
- **almost for free !**

OBR Issues

- **Critical issue: basic OBR algorithm is unsafe**
  - it uses the dual solution of the linear relaxation
  - Efficient LP solvers work with floats $\longrightarrow$
    the available dual solution λ^* is an **approximation**
    if used in OBR ...
    ... $\rightarrow$ **OBR may remove actual optimum !**

- **Solutions:** two ways to take advantage of OBR
  1. **prove dual solution** (Kearfott): combininig the dual of linear relaxation with the Kuhn-Tucker conditions
  2. **validate the reduction** proposed by OBR with CP !

CP approach: intuition

- **Essential observation**: if the constraint system

$$
\begin{array}{l}
L \leq f(x) \leq U \\
g_i(x) = 0, \; i = 1..k \\
g_j(x) \leq 0, \; j = k+1..m
\end{array}
$$

  has **no solution** when the domain of x is set to $[\underline{\mathbf{x}}_i, \underline{\mathbf{x}}'_i]$, then the reduction computed by OBR is valid

- Try to reject $[\underline{\mathbf{x}}_i, \underline{\mathbf{x}}'_i]$ with classical filtering techniques, otherwise add this box to the list of boxes to process

CP algorithm

$\mathcal{L}_r := \emptyset$ % set of potential non-solution boxes

for each variable x_i **do**
  Apply OBR
    and add the generated potential non-solution boxes to $\mathcal{L}_r$

for each box $\mathbf{B_i}$ in $\mathcal{L}_r$ **do**
  $\mathbf{B_i'} :=$ 2*B*-filtering($\mathbf{B_i}$)
  if $\mathbf{B_i'} = \emptyset$ **then** reduce the domain of x_i
  else $\mathbf{B_i''} :=$ QUAD_SOLVER-filtering($\mathbf{B_i'}$)
    if $\mathbf{B_i''} = \emptyset$ **then** reduce the domain of x_i
    else add $\mathbf{B_i}$ to global list of box to be handled **endif**
  endif

[Compute $\underline{f}$ with QUAD_SOLVER in $\mathbf{X}$]

Experiments

- Compares 4 versions of the branch and bound algorithm:
  - without OBR
  - with unsafe OBR
  - with safe OBR based on Kearfott's approach
  - with safe OBR based on CP techniques

  implemented with **Icos using Coin/CLP and Coin/IpOpt**

- On 78 **benches** (from Ryoo & Sahinidis 1995, Audet thesis and the coconut library)

- All experiments have been done on PC-Notebook/1Ghz.

Experimental Results (2): Synthesis

Synthesis of the results:

	$\Sigma_t(s)$	*%saving*
no OBR	2384.36	-
unsafe OBR	881.51	63.03%
safe OBR Kearfott	1975.95	17.13%
safe OBR CP	**454.73**	**80.93%**

(with a timeout of 500s)

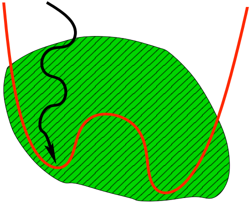
Safe CP-based OBR faster than unsafe OBR !

... because wrong domains reductions prevent the upper-bounding process from improving the current upper bound !!

Computing “sharp” upper bounds

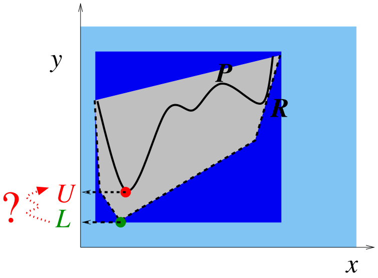
Statement of the problem

- **Upper bounding**
  - **local search**
    $\longrightarrow$ approximate feasible point x_{approx}
  - **epsilon inflation process and proof**
    $\longrightarrow$ provide a feasible box x_{proved}
  - compute $\bar{\mathbf{f}}^* = min(\bar{\mathbf{f}}(x_{proved}), \bar{\mathbf{f}}^*)$



- **Critical issue**: to prove the existence of a feasible point in a reduced box
  - Singularities
  - Guess point too far from a feasible region (local search works with floats)

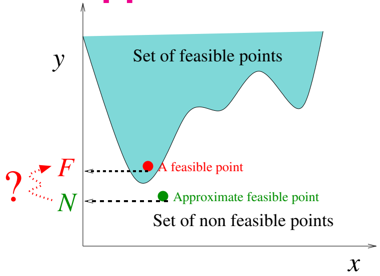
Using the lower bound to get an upper-bound



Branch&Bound step where *P* is the set of feasible points and *R* is the linear relaxation

Idea: modify the lower bound ... to get an upper-bound !

Lower bound: a good starting point to find a feasible upper-bound ?



N, optimal solution of R, not a feasible point of P but (may be) **a good starting point**:

- BB splits the domains at each iteration:
  smaller box $\rightsquigarrow$ N nearest from the optima of P
- Proof process inflates a box around the guess point
  $\rightsquigarrow$ compensate the distance from the feasible region

Method

- Correction procedure to **get a better feasible point** from a given approximate feasible point

  → to exploit **Newton-Raphson for under-constrained systems** of equations (and Moore-Penrose inverse)

  good convergence when the starting point is nearly feasible

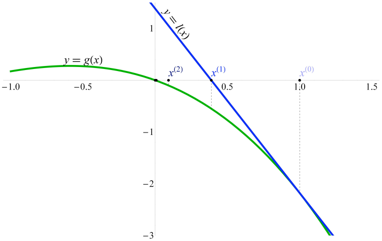
Handling square systems of equations

- $g = (g_1, \ldots, g_m) : R^n \longrightarrow R^m$ **($n = m$)**

  $\rightarrow$ Newton-Raphson step:

  $$x^{(i+1)} = x^{(i)} - J_g^{-1}(x^{(i)})g(x^{(i)})$$

  Converges well if the exact solution to be approximated is **not singular**



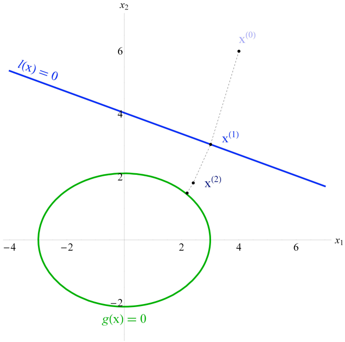
Handling under-constrained systems of equations

Manifold of solutions
→ linear system $l(x) = 0$ is under-constrained
→ **Choose a solution** $x^{(1)}$ of $l(x) = 0$

Best choice:
Solution of $l(x) = 0$ close to $x^{(0)}$
Can easily be computed with the **Moore-Penrose inverse**:

$$x^{(i+1)} = x^{(i)} - A_g^+(x^{(i)})g(x^{(i)})$$

$A_g^+ \in R^{n\times m}$ is the Moore-Penrose inverse of A_g, solution of the equation which minimizes $||x^{(1)} - x^{(0)}||$)

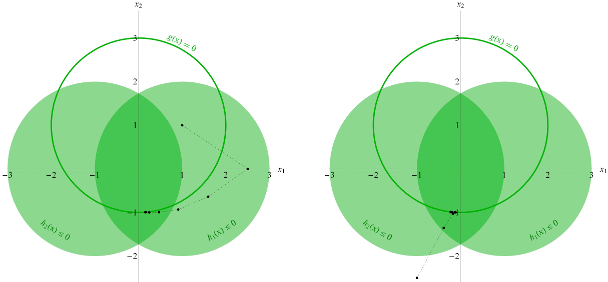


Handling under-constrained systems of equations and inequalities

- Under-constrained systems of equations and **inequalities**
  $\rightsquigarrow$ introduce **slack variables**

- **Initial values** for the slack variables have to be provided

  Slightly positive value
  $\longrightarrow$ to break the symmetry
  $\longrightarrow$ good convergence



A new upper bounding strategie

Function `UpperBounding`(`IN` $\mathbf{x}$, x^*_{LP}; `OUT` $\mathcal{S}'$)

% $\mathcal{S}'$: list of proven feasible boxes
% x^*_{LP}: the optimal solution of the LP relaxation of $\mathcal{P}(\mathbf{x})$
$\mathcal{S}'$:= $\emptyset$
x^*_{corr} := FeasibilityCorrection(x^*_{LP}) % Improving x^*_{LP} feasibility
$\mathbf{x}_p$:= InflateAndProve(x^*_{corr}, $\mathbf{x}$)
if $\mathbf{x}_p \neq \emptyset$ **then**
    $\mathcal{S}'$:= $\mathcal{S}' \cup \mathbf{x}_p$
endif
return $\mathcal{S}'$

Experiments (1)

- Significant set of benchmarks of the COCONUT project
- Selection of 35 benchmarks where Icos did find the global minimum while relying on an unsafe local search
- 31 benchmarks are solved within a 30s time out
- Almost all benchmarks are solved in **much less time** and with **much more proven solutions**

Experiments (2)

Name	(n,m)	LS: $t(s)$	UB/LB: $t(s)$
alkyl	(14, 7)	-	1.54
circle	(3, 10)	1.98	0.84
ex14_1_2	(6, 9)	-	1.74
ex14_1_3	(3, 4)	-	0.42
ex14_1_6	(9, 15)	-	12.44
ex14_1_8	(3, 4)	-	-
ex2_1_1	(5, 1)	0.09	0.04
ex2_1_2	(6, 2)	-	0.24
ex2_1_3	(13, 9)	-	1.32
ex2_1_4	(6, 5)	0.52	0.43
ex2_1_6	(10, 5)	1.61	0.35
ex3_1_3	(6, 6)	1.03	0.29
ex3_1_4	(3, 3)	6.51	0.14
ex4_1_2	(1, 0)	18.84	17.03
ex4_1_6	(1, 0)	0.11	14.28
ex4_1_7	(1, 0)	0.07	0.01
ex5_4_2	(8, 6)	-	18.15
ex6_1_2	(4, 3)	0.51	0.52
ex6_1_4	(6, 4)	7.45	8.92
ex7_3_5	(13, 15)	-	-
ex8_1_6	(2, 0)	-	0.39
ex9_1_1	(13, 12)	-	-
ex9_1_10	(14, 12)	-	3.76
ex9_1_4	(10, 9)	-	0.49
ex9_1_5	(13, 12)	-	2.68
ex9_1_8	(14, 12)	-	3.76
ex9_2_1	(10, 9)	-	0.68
ex9_2_4	(8, 7)	2.94	0.69
ex9_2_5	(8, 7)	-	-
ex9_2_7	(10, 9)	-	0.68
ex9_2_8	(6, 5)	-	0.53
house	(8, 8)	-	0.90
nemhaus	(5, 5)	0.02	0.01

Conclusion

- **CSP refutation techniques**
  - allow a **safe** and **efficient** implementation of OBR
  - can **outperform standard mathematical methods**
  - might be suitable for other unsafe method
- **Safe global constraints**
  - provide an efficient alternative to local search: $\longrightarrow$ good starting point for a Newton method $\rightsquigarrow$ feasible region
  - **drastically improve the performances** of the upper-bounding process