Ibexopt-sip: an optimizer for semi-infinite problems

Antoine Marendet

Équipe OGRE

École Centrale de Nantes

antoine.marendet@gmail.com

Ibexdays, Nantes April 23, 2019

ibexopt-sip

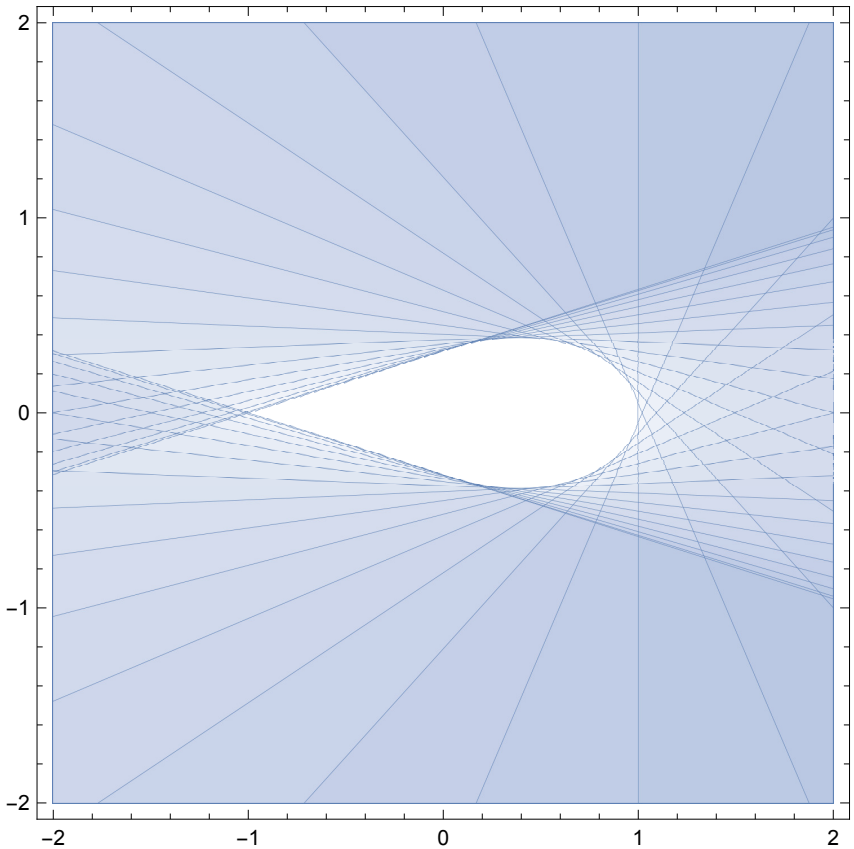
- ibexopt-sip: ibexopt equivalent for semi-infinite problems
- Fully develop with Ibex
- Ibex plugin: on the website (outdated), and on GitHub
- Main algorithm based on IbexOpt

Semi-infinite problems

- NLP: $\min f(x)$ s.t. $g(x) \leq 0$ and $h(x) = 0$
- At least one semi-infinite constraint (SIC)

$$g_i(x) = \max_{a \in A_i} \tilde{g}_i(x, a) \tag{1}$$

- Equivalently: $\forall a \in A_i,\ \tilde{g}_i(x, a) \leq 0$ (robust optimization)
- Note: $A_i \subseteq R^m$



- Feasible set for the SIC
  $\forall a \in [-\pi, \pi]$,
  $x_1 \cos(a) + x_2(\sin(a) + a) - 1 \leq 0$
- Infinitely many linear inequalities

Branch-and-Bound for NLP

- Branch-and-Bound for NLP:
  - filtering: constraint propagation, relaxations (lower bounding)
  - find feasible points: restrictions, local search
  - bisect variable domains
- Goal: extend this framework to SIP

Integration into Ibex

- Treat SIC just as non-SIC constraints
- Transparent to the B&B, smoothly integrates into Ibex
- Main components of Ibex B&B:
  - Interval evaluation of constraint and gradient
  - Constraint propagation
  - Linear relaxations
  - Linear restrictions
  - + Line search to find feasible points

Parameter space paving

$$g(x) = \max_{a \in A} \tilde{g}(x, a) \tag{2}$$

- Only relevant values of A are $a_x^\star = \text{argmax}_{a \in A}\, \tilde{g}(x, a)$
- We store the paving $\mathcal{P}$ such that $a_x^\star \in \mathcal{P}$
- $\mathcal{P}$ is refined by bisection and filtered during the search
- $[g]([x])$ and generalized gradient $\partial[g]([x])$ computed using $\mathcal{P}$

Example $\tilde{g}(x, a) = x_1 \cos(a) + x_2(\sin(a) + a) - 1$ and $a \in [-\pi, \pi]$

$$[g]^1([1, 1.2], [0.5, 0.7]) = [-5.10, 3.10] \tag{3}$$
$$[g]^{100}([1, 1.2], [0.5, 0.7]) = [0.41, 1.01] \tag{4}$$

- Interval evaluations $\Rightarrow$ outer box rejection, inner box proving, first-order rejection test,
- Convergent paving $\Rightarrow$ B&B convergence

Filtering

- Goal: filter infeasible and non-optimal points
- SIC: $\forall a \in A,\ \tilde{g}(x,a) \leq 0$
- discretize: $\tilde{a} \in A \rightarrow$ relaxation of $g(x) \leq 0$
- Ibex contractors can be used *as is* on the discretization constraints (HC4, PolytopeHull)
- Choice of $\tilde{a}$ critical for efficiency

$\rightarrow$ $\mathcal{P}$ helps relaxing on relevant parameter values

Feasible points

- Goal: find improving feasible points
- Linear restrictions similar to non-SIC case :

$$g(\tilde{x}) + [D_x g]([x])(x - \tilde{x}) \leq 0 \tag{5}$$

- Extension to SIC: $[\tilde{g}](\tilde{x}, [a]) + [D_x \tilde{g}]([x], [a])(x - \tilde{x}) \leq 0$
- One linear restriction for each $[a] \in \mathcal{P}$

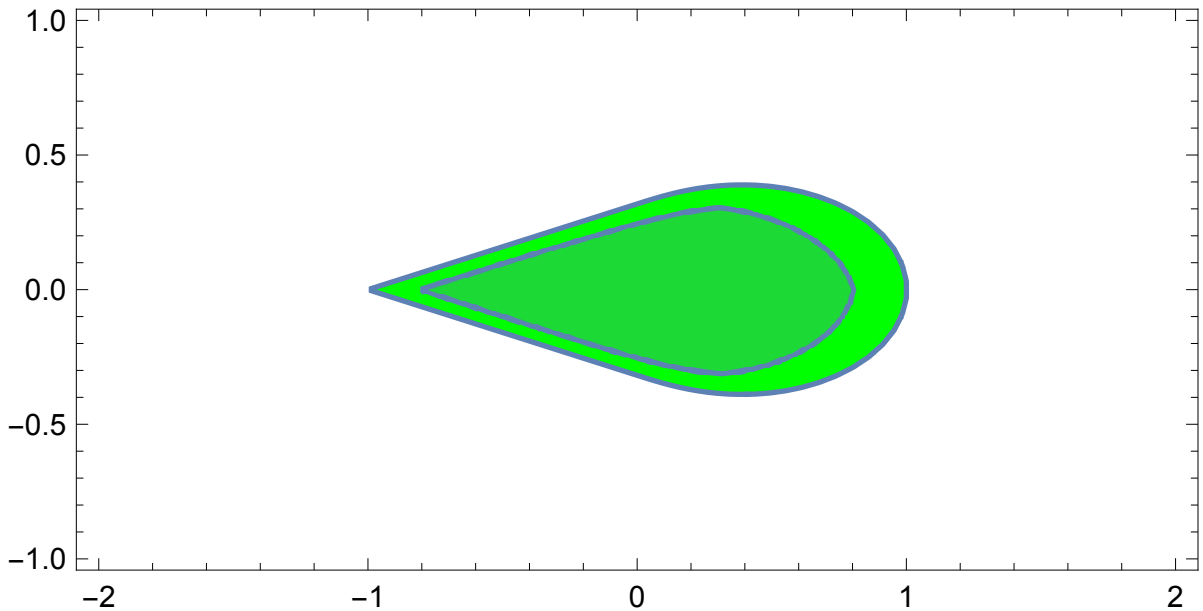
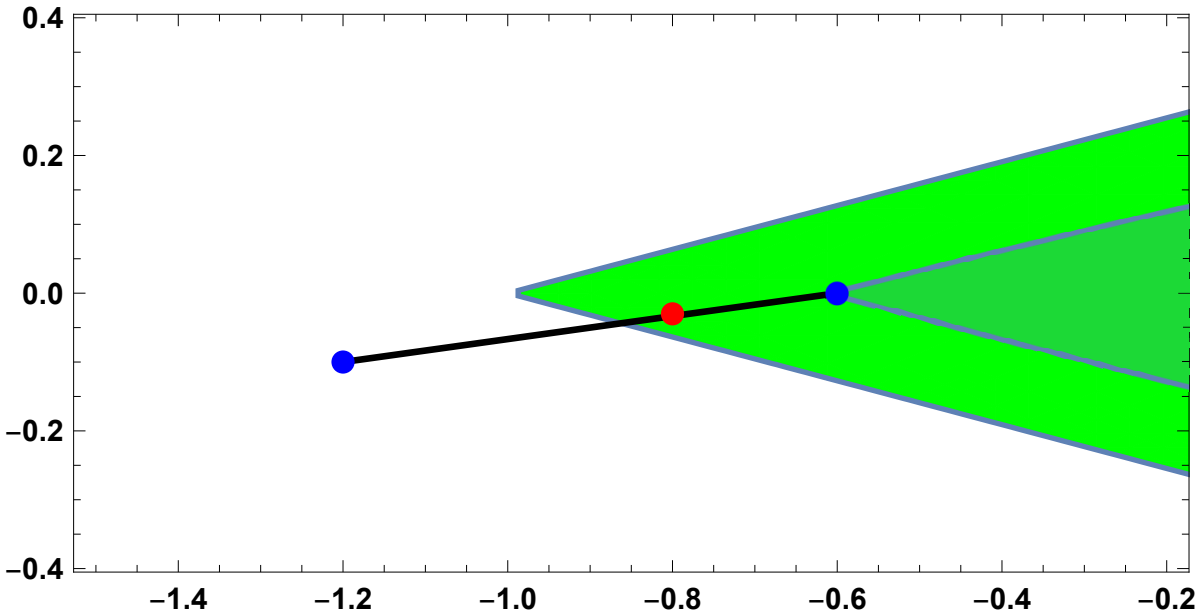


Figure: Linear restriction with 100 subdivisions

Feasible points

- Addition to ibexopt: line search
- Search a improving feasible point from a relaxation point in a direction pointing towards the feasible set
- Various strategies to find the direction: gradients on the relaxation point, linear restrictions point, midpoint...
- Cheap and effective for SIP, where linear restrictions are sometimes not enough to find good feasible points



Bisection

- Bisection on decision variables: same bisectors as Ibex
- Bisection of parameter variables: custom strategy, probably needs improvements

Implementation

- Class `SIPSystem` similar to IbexOpt's `System`
- New classes `NormalConstraints` and `SIConstraints` to wrap constraints
- `SIPOptimizer` with same interface as IbexOpt's `Optimizer` (and almost same code)
- specialized `Ctc`s, `LoupFinder`s and `Linearizer`s
- `Bxp` class `BxpNodeData` storing paving $\mathcal{P}$ for each `SIConstraints` in B&B nodes

Example in Minibex

```
//!--universal="a"
/* Universal takes an egrep-style regular expression */
Variables
    x1 in [-2, 2];
    x2 in [-2, 2];
    a in [-pi, pi];

Minimize
    x1;

Constraints
    x1*cos(a) + x2*(sin(a) + a) - 1 <= 0;
end
```

Invocation:

```
$ ibexopt-sip exemple.mbx -a1e-3
```

```
$ ibexopt-sip exemple.mbx --universal="a" -a1e-3
```

Numerical results

- Comparison with state-of-the-art Mitsos-Djelassi algorithm [DM16] on Mitsos SIP test set [Mit09]
- Good results: 6.26s vs 114s

References I

[DM16] Hatim Djelassi and Alexander Mitsos.
A hybrid discretization algorithm with guaranteed feasibility for the global solution of semi-infinite programs.
Journal of Global Optimization, pages 1–27, 2016.

[Mit09] Alexander Mitsos.
Test set for semi-infinite programs.
Technical report, Massachusetts Institute of Technology, http://web.mit.edu/mitsos/www/pubs/siptestset.pdf, 2009.