Simplex with Sum of Infeasibilities for SMT

Tim King[1] Clark Barrett[1] Bruno Dutertre[2]

[1]New York University

[2]SRI International

October 23, 2013

Decision Procedure for QF_LRA

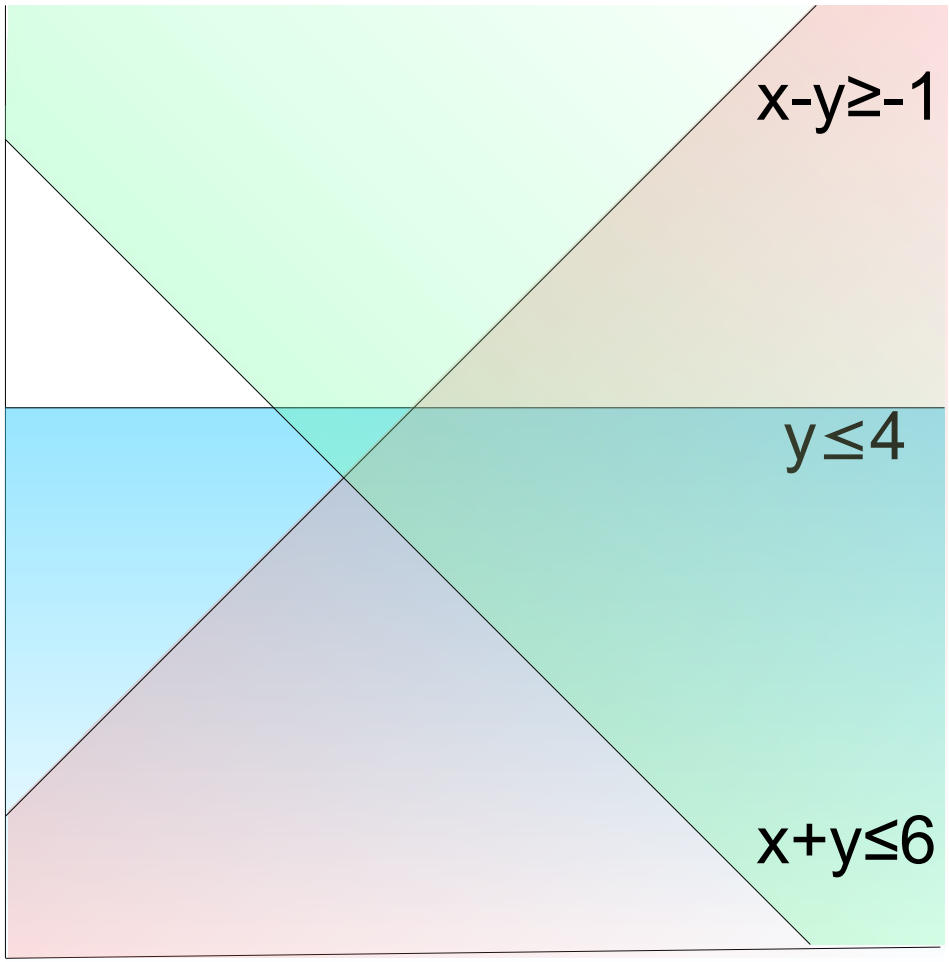
Quantifier Free Linear Real Arithmetic

Is there a satisfying assignment, $a : \mathcal{V} \to \mathbb{R}$, that makes,

$$\begin{array}{rcrcr} x & + & y & \leq & 6 \\ x & - & y & \geq & -1 \\ & & y & \leq & 4 \end{array}$$

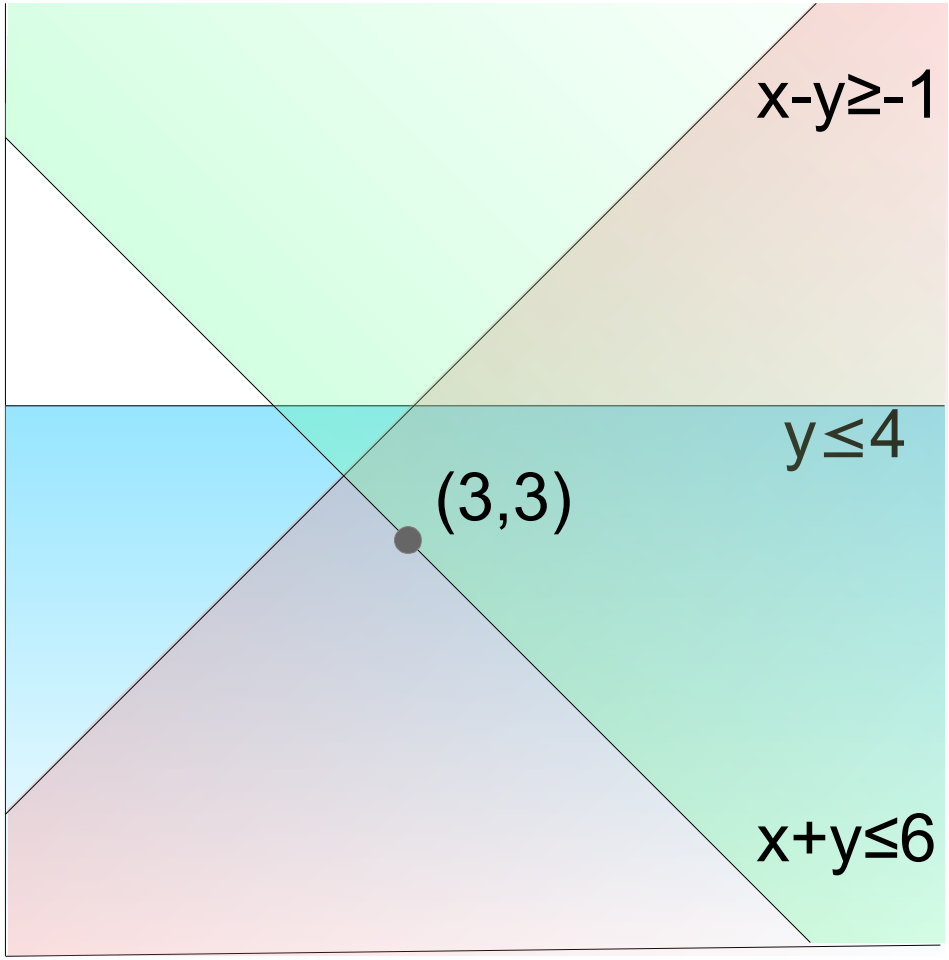
evaluate to true?

Visually



$$\begin{array}{rcrcr} x & + & y & \leq & 6 \\ x & - & y & \geq & -1 \\ & & y & \leq & 4 \end{array}$$

Visually



$$\begin{array}{rcrcr} x & + & y & \leq & 6 \\ x & - & y & \geq & -1 \\ & & y & \leq & 4 \end{array}$$

$$\begin{vmatrix} a(x) \\ a(y) \end{vmatrix} = \begin{vmatrix} 3 \\ 3 \end{vmatrix}$$

How?

Simplex for DPLL(T)

Dutertre and de Moura CAV'06

- Decision Procedure for `QF_LRA`
- De-facto Standard Algorithm:
  Z3, yices[1/2], mathsat5, SMTInterpol, CVC4, etc.
- Good on verification/SMTLIB
- Basis for SOISimplex

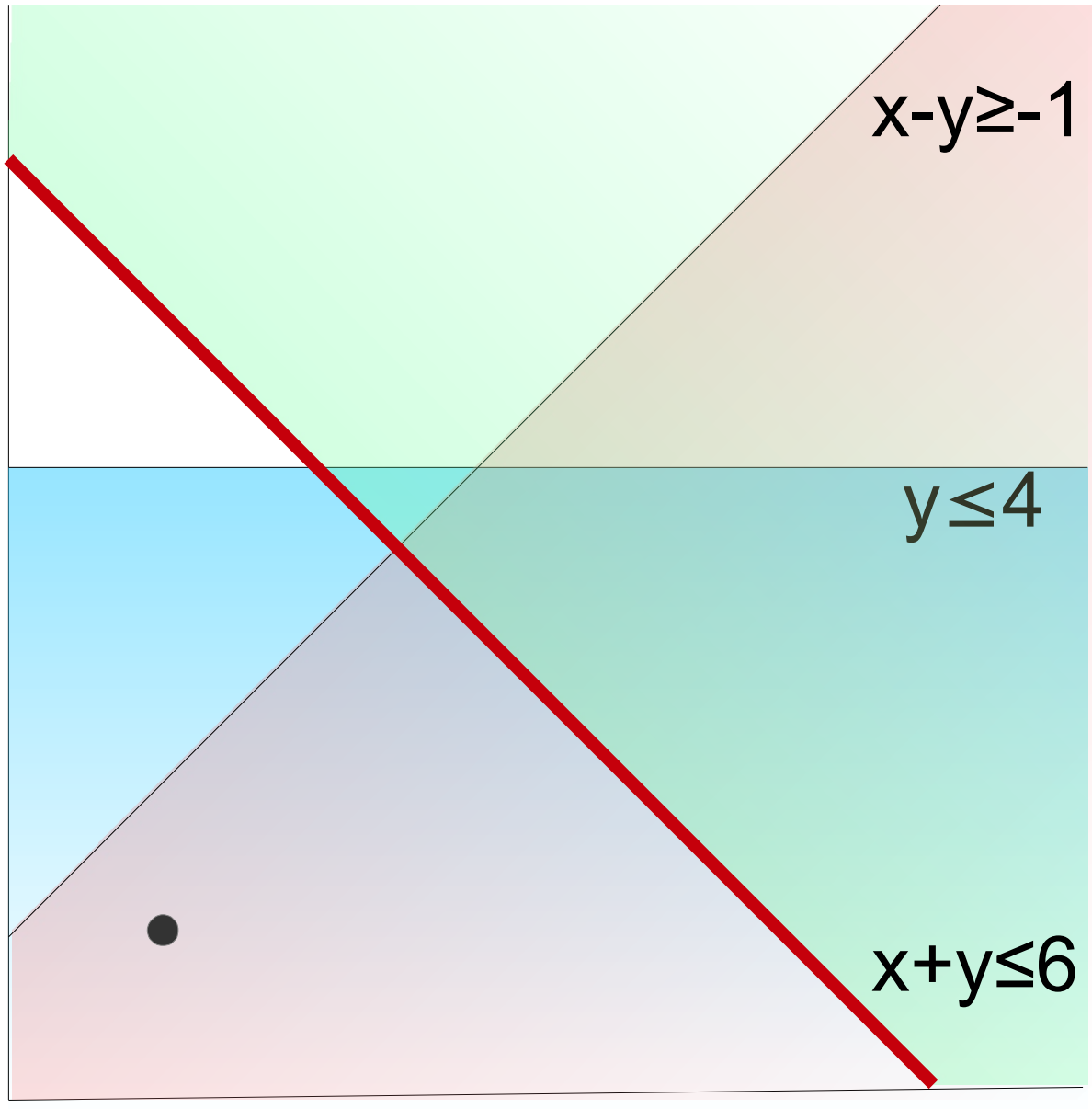
`latendresse` Benchmarks

Examples that started this work

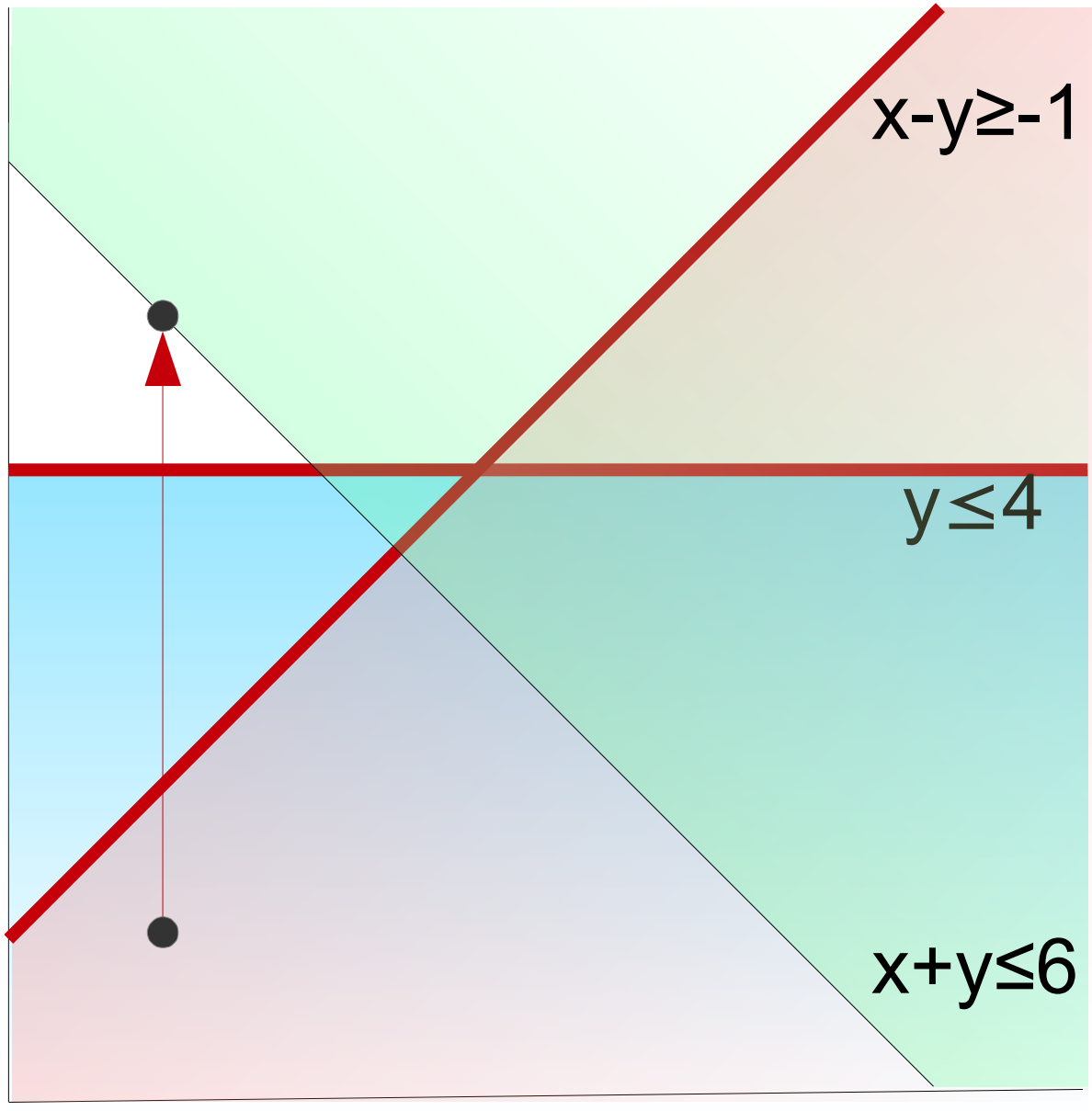
- Use yices to model chemical reactions in bacteria
- SimplexForDPLL chokes right out of the gate (yices, CVC4, Z3, mathsat5, ...)
- Family of 18 benchmarks
- Pick up ~ 1 by fiddling with heuristics (with high timeouts)

SOISimplex tries to add stability to the search by globally minimizing a function.

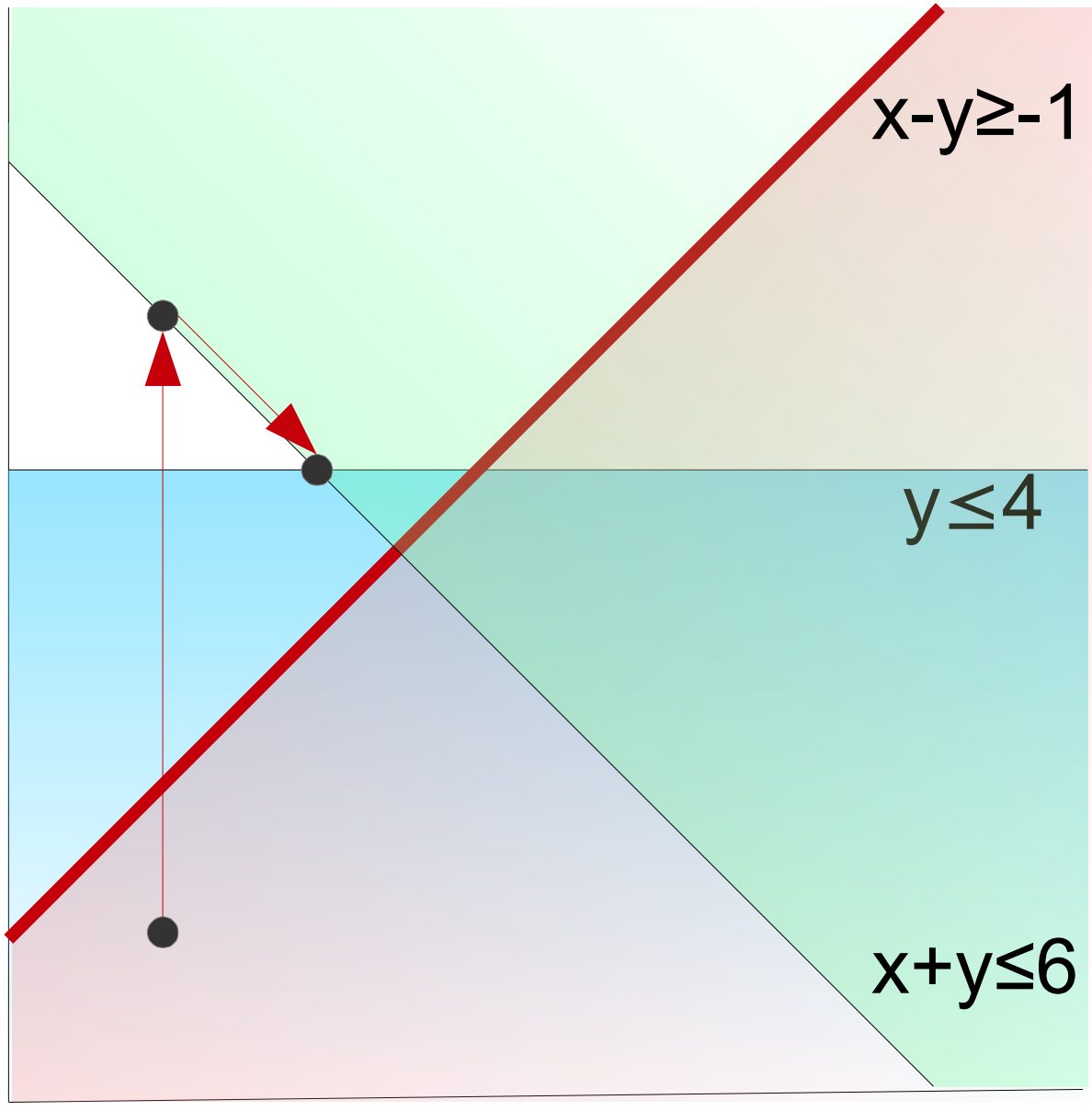
Simplex for DPLL(T) Search



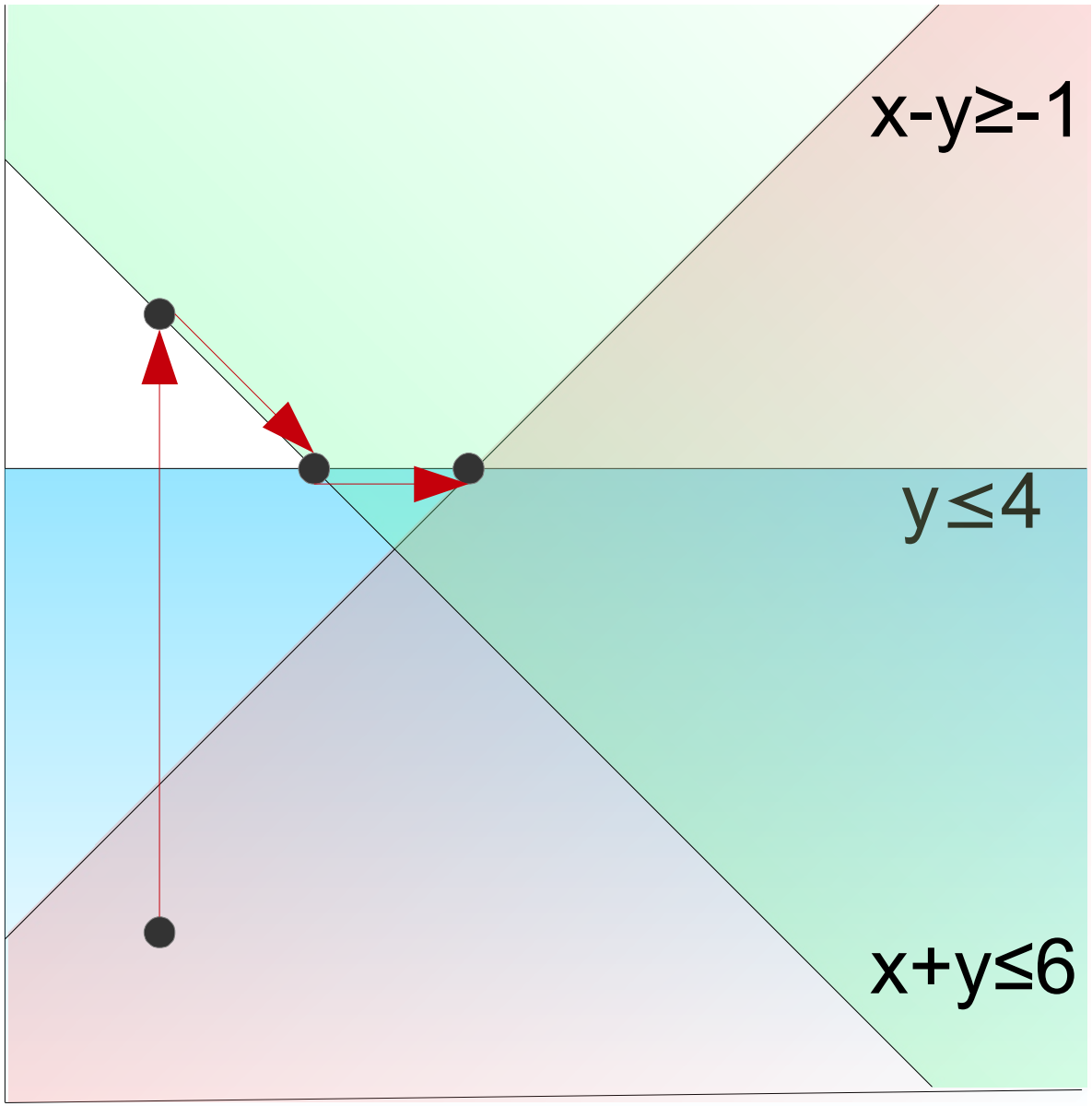
Simplex for DPLL(T) Search



Simplex for DPLL(T) Search



Simplex for DPLL(T) Search



Preprocessing

- Introduce a fresh b_i for each $\sum t_{i,j} \cdot x_j$
- Literals are of the form:

$$\bigwedge (b_i = \sum_{x_j} t_{i,j} \cdot x_j) \wedge \bigwedge l(x) \leq x \leq u(x)$$

  and b_i appears in exactly 1 equality.

- Collect into:

$$T\vec{\mathcal{V}} = 0 \qquad \vec{l} \leq \vec{\mathcal{V}} \leq \vec{u}$$

Preprocessing

- Introduce a fresh b_i for each $\sum t_{i,j} \cdot x_j$
- Literals are of the form:

$$\bigwedge (b_i = \sum_{x_j} t_{i,j} \cdot x_j) \wedge \bigwedge l(x) \leq x \leq u(x)$$

  and b_i appears in exactly 1 equality.

- Collect into:

$$T\vec{\mathcal{V}} = 0 \qquad \vec{l} \leq \vec{\mathcal{V}} \leq \vec{u}$$

Preprocessing

- Introduce a fresh b_i for each $\sum t_{i,j} \cdot x_j$
- Literals are of the form:

$$\bigwedge (b_i = \sum_{x_j} t_{i,j} \cdot x_j) \wedge \bigwedge l(x) \leq x \leq u(x)$$

  and b_i appears in exactly 1 equality.

- Collect into:

$$T\vec{\mathcal{V}} = 0 \qquad \vec{l} \leq \vec{\mathcal{V}} \leq \vec{u}$$

Basic, Nonbasic, & Tableau

- Every row in T is solved for a basic variable b

$$b = \sum_{x \in \mathcal{N}} t_{b,x} x$$

- Not solved for variables are "nonbasic" ($\mathcal{N}$)
- Changing $\mathcal{B}$ changes T

Tableau Example

$$\begin{array}{rcrcr} x & + & y & \leq & 6 \\ x & - & y & \geq & -1 \\ & & y & \leq & 4 \end{array}$$

Tableau Example

$$\begin{aligned} u &= x + y \\ v &= x - y \end{aligned}$$

$$u \geq 6 \wedge v \geq -1 \wedge y \leq 4$$

Tableau Example

$$\begin{aligned} u &= x + y \\ v &= x - y \end{aligned}$$

$$u \geq 6 \wedge v \geq -1 \wedge y \leq 4$$

$$T\vec{\mathcal{V}} = \begin{bmatrix} -1 & 0 & 1 & 1 \\ 0 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} u \\ v \\ x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\mathcal{B} = \{u, v\}, \mathcal{N} = \{x, y\}$$

Tableau Example

$$\begin{aligned} u &= x + y \\ v &= x - y \end{aligned}$$

$$u \geq 6 \wedge v \geq -1 \wedge y \leq 4$$

$$T\vec{\mathcal{V}} = \begin{bmatrix} -1 & 0 & 1 & 1 \\ 0 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} u \\ v \\ x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\mathcal{B} = \{u, v\}, \mathcal{N} = \{x, y\}$$

Tableau Example

$$\begin{aligned} u &= x + y \\ v &= x - y \end{aligned}$$

$$u \geq 6 \wedge v \geq -1 \wedge y \leq 4$$

$$T\vec{\mathcal{V}} = \begin{bmatrix} -1 & 0 & 1 & 1 \\ 0 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} u \\ v \\ x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\mathcal{B} = \{u, v\}, \mathcal{N} = \{x, y\}$$

Simplex for DPLL(T)

Pseudocode

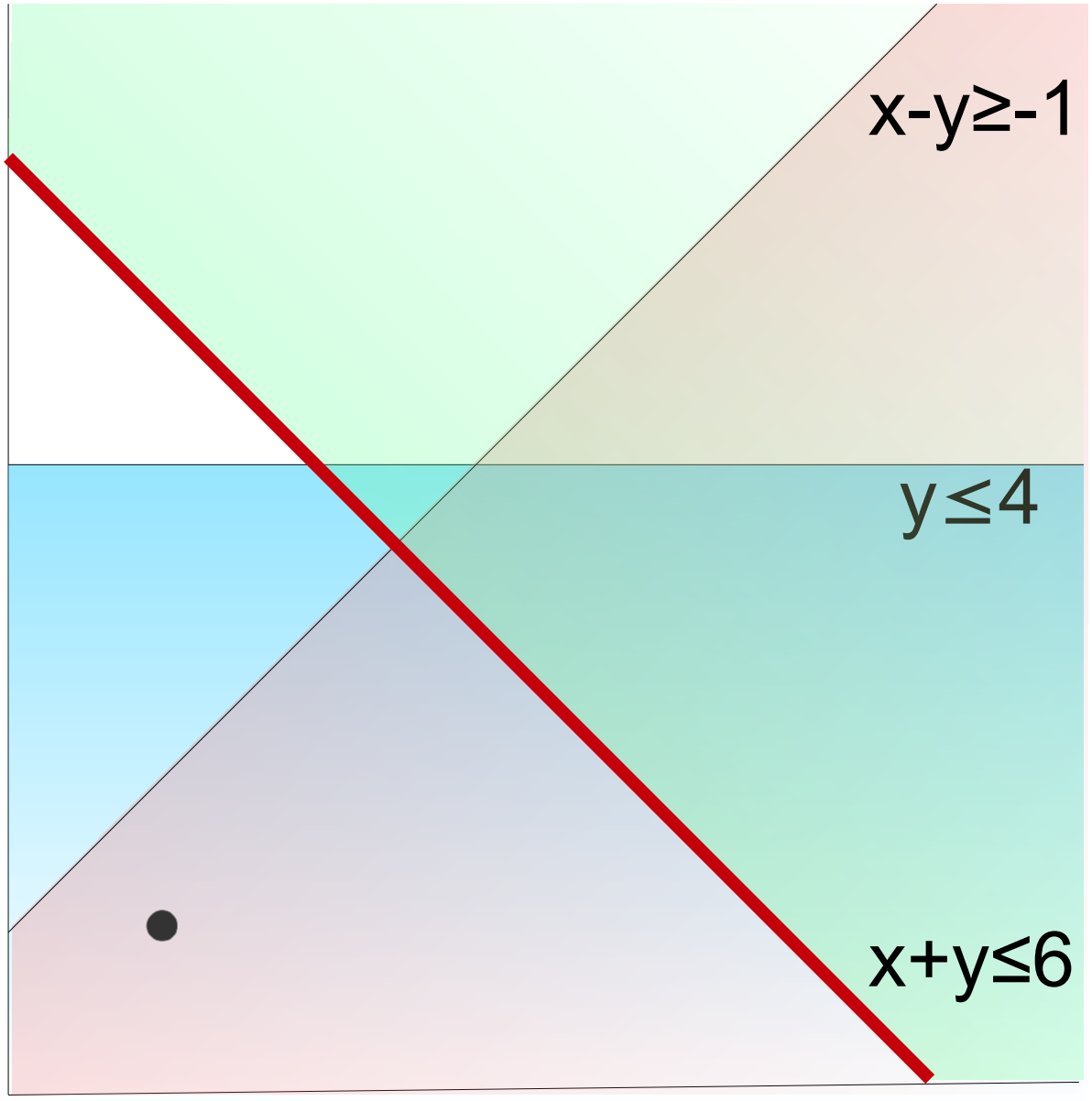
procedure SimplexForDPLL
  while $b \in \mathcal{B}$ s.t. $a(b) > u(b)$ or ... **do**
    select some $b = \sum t_{b,x} \cdot x$
    if $\sum c \cdot x$ is at a minimum **then**
      return a row conflict
    else
      Select x in $\sum c \cdot x$
      Change the assignment of x s.t. $a(b) \leftarrow u(b)$
      Pivot(b, x)

Simplex for DPLL(T)

Pseudocode

procedure SIMPLEXFORDPLL
  while $b \in \mathcal{B}$ s.t. $a(b) > u(b)$ or ... **do**
    select some $b = \sum t_{b,x} \cdot x$
    if $\sum c \cdot x$ is at a minimum **then**
      return a row conflict
    else
      Select x in $\sum c \cdot x$
      Change the assignment of x s.t. $a(b) \leftarrow u(b)$
      PIVOT(b, x) ▷ $O(|T|)$

Simplex for DPLL(T) Search



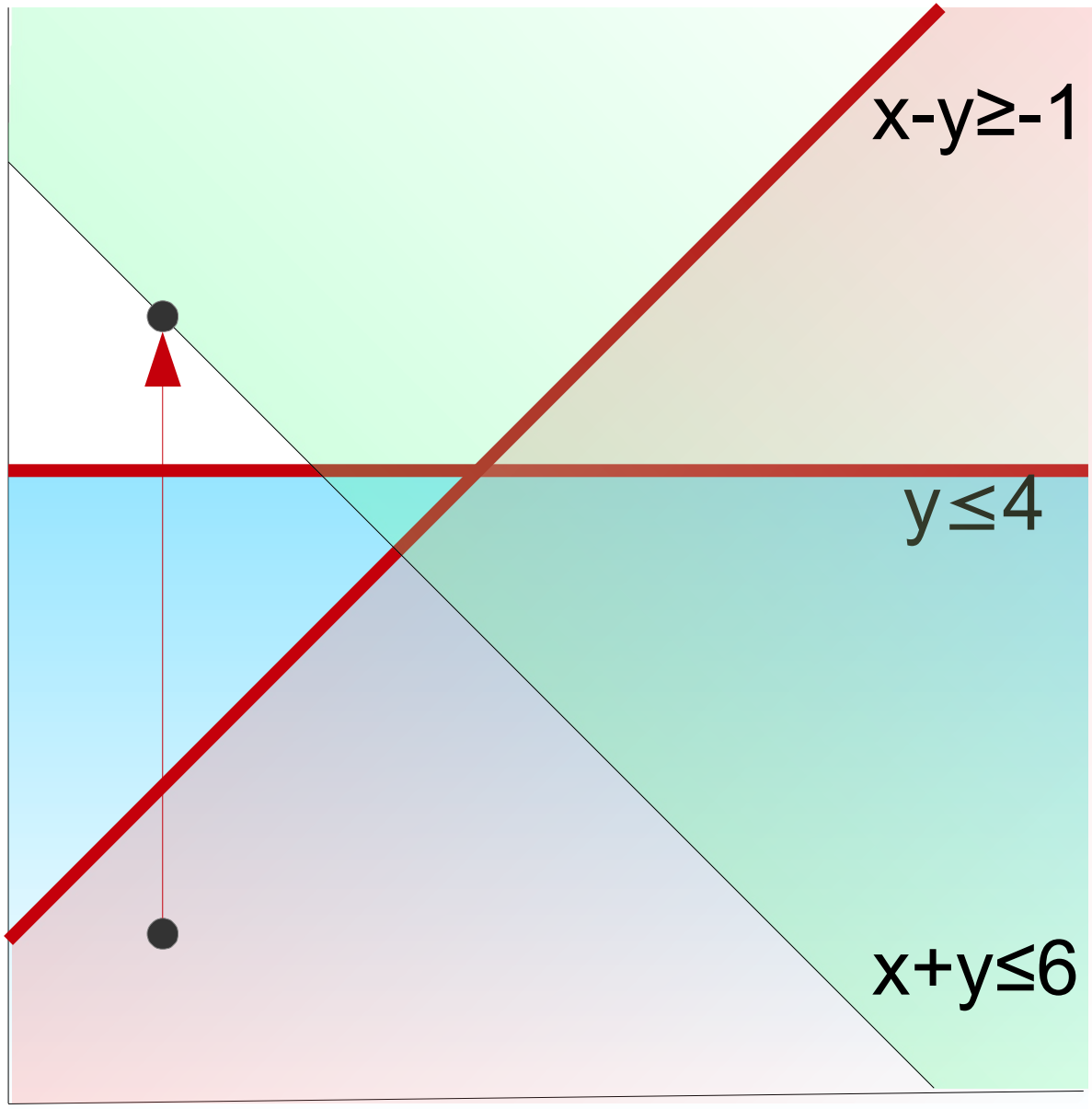
Greedily optimize

$$x + y$$

ignoring

$$x - y \geq -1, \quad y \leq 4$$

Simplex for DPLL(T) Search



Greedily optimize

$$x + y$$

ignoring

$$x - y \geq -1, \quad y \leq 4$$

Sum Of Infeasibilities

- Infeasibility of x is how much x violates its bounds.

$$\text{Vio}(x) = \begin{cases} a(x) - u(x) & a(x) > u(x) \\ 0 & l(x) \leq a(x) \leq u(x) \\ l(x) - a(x) & a(x) < l(x) \end{cases}$$

- Sum of infeasibilities:

$$\text{Vio}(\mathcal{V}) = \sum_{x \in \mathcal{V}} \text{Vio}(x)$$

- SOISimplex minimizes $\text{Vio}(\mathcal{V})$ every round

Sum Of Infeasibilities

- Infeasibility of x is how much x violates its bounds.

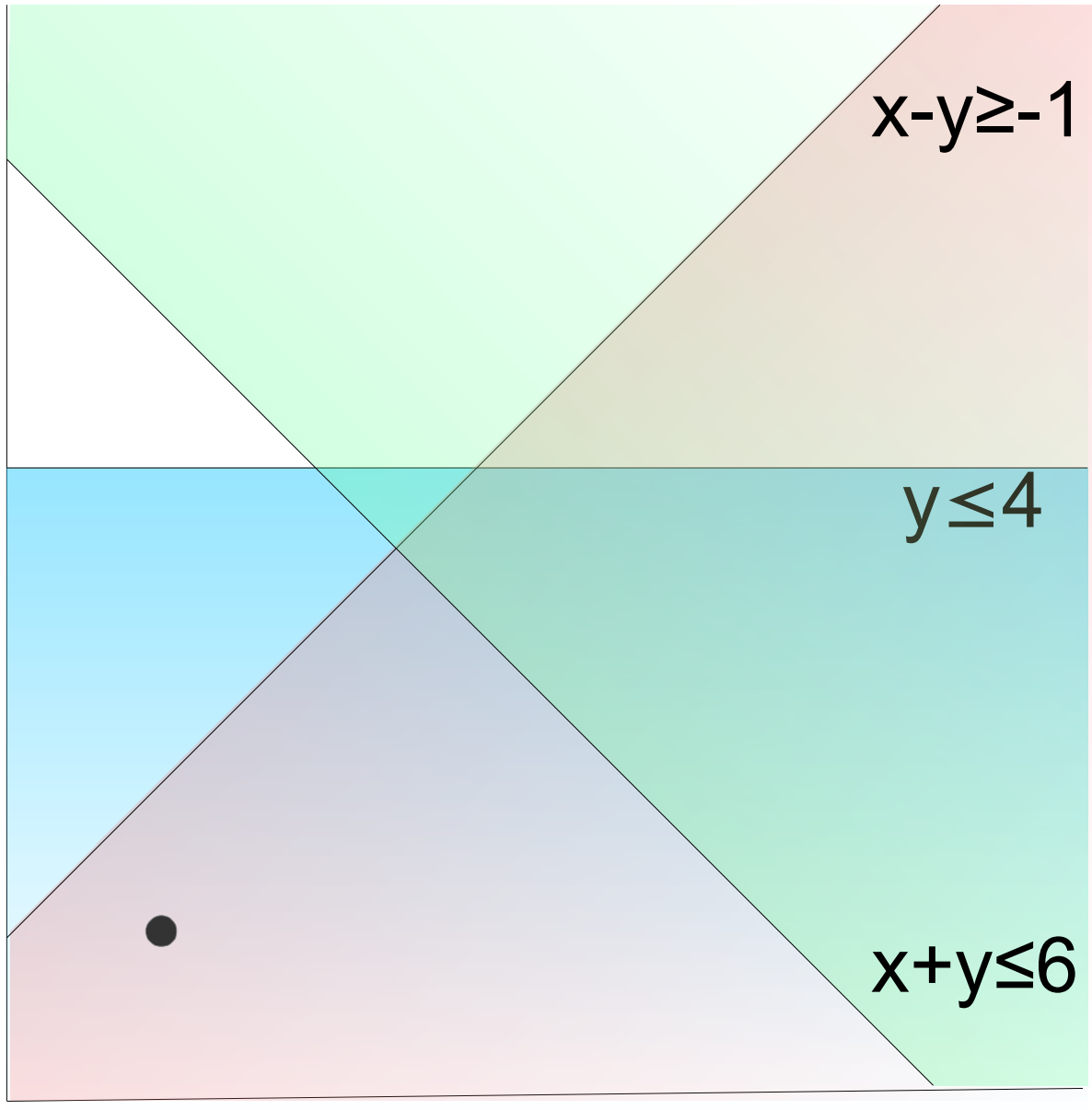
$$\text{Vio}(x) = \begin{cases} a(x) - u(x) & a(x) > u(x) \\ 0 & l(x) \leq a(x) \leq u(x) \\ l(x) - a(x) & a(x) < l(x) \end{cases}$$

- Sum of infeasibilities:

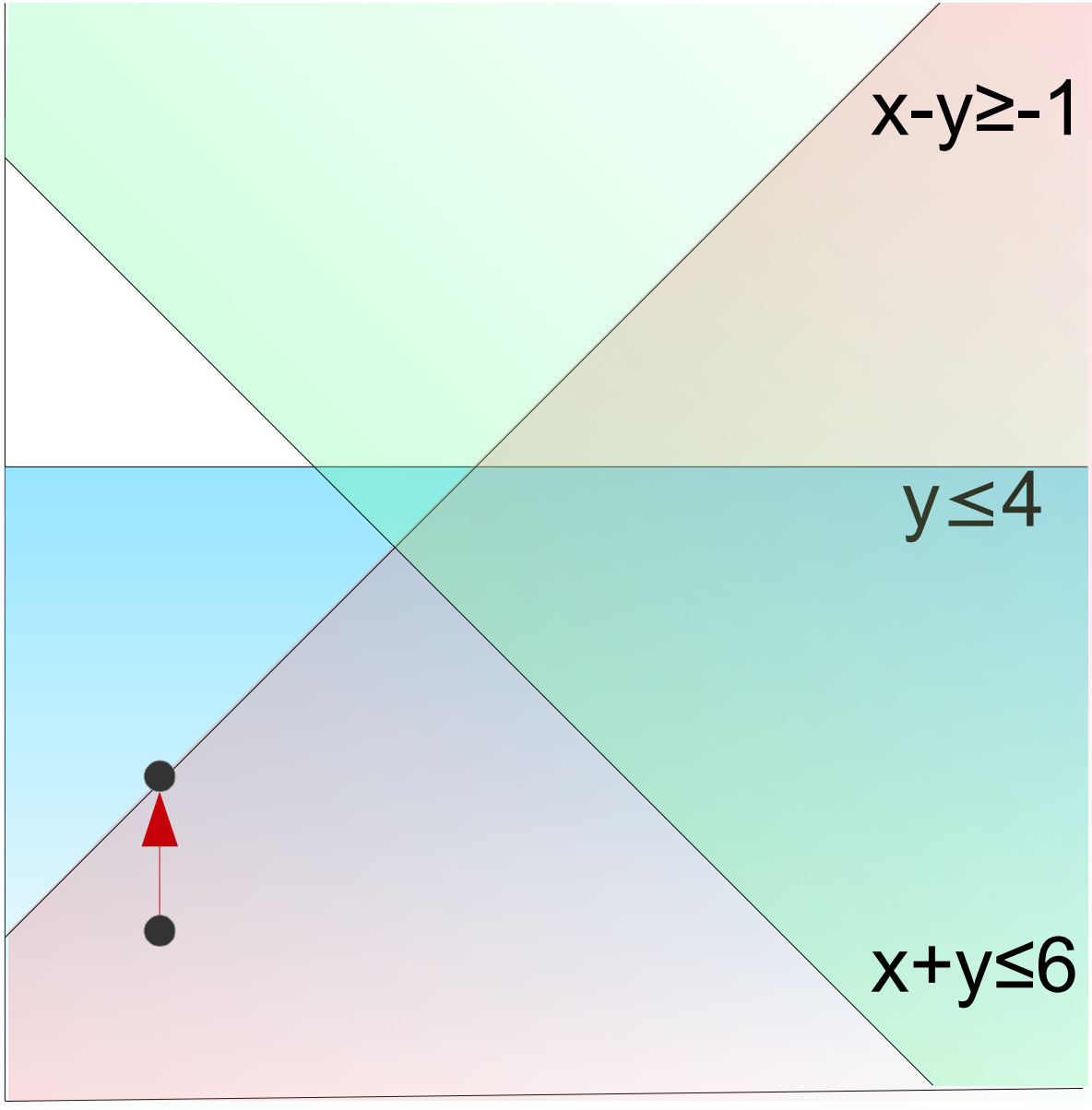
$$\text{Vio}(\mathcal{V}) = \sum_{x \in \mathcal{V}} \text{Vio}(x)$$

- SOISimplex minimizes $\text{Vio}(\mathcal{V})$ every round
  - Well known in optimization
  - New for SMT

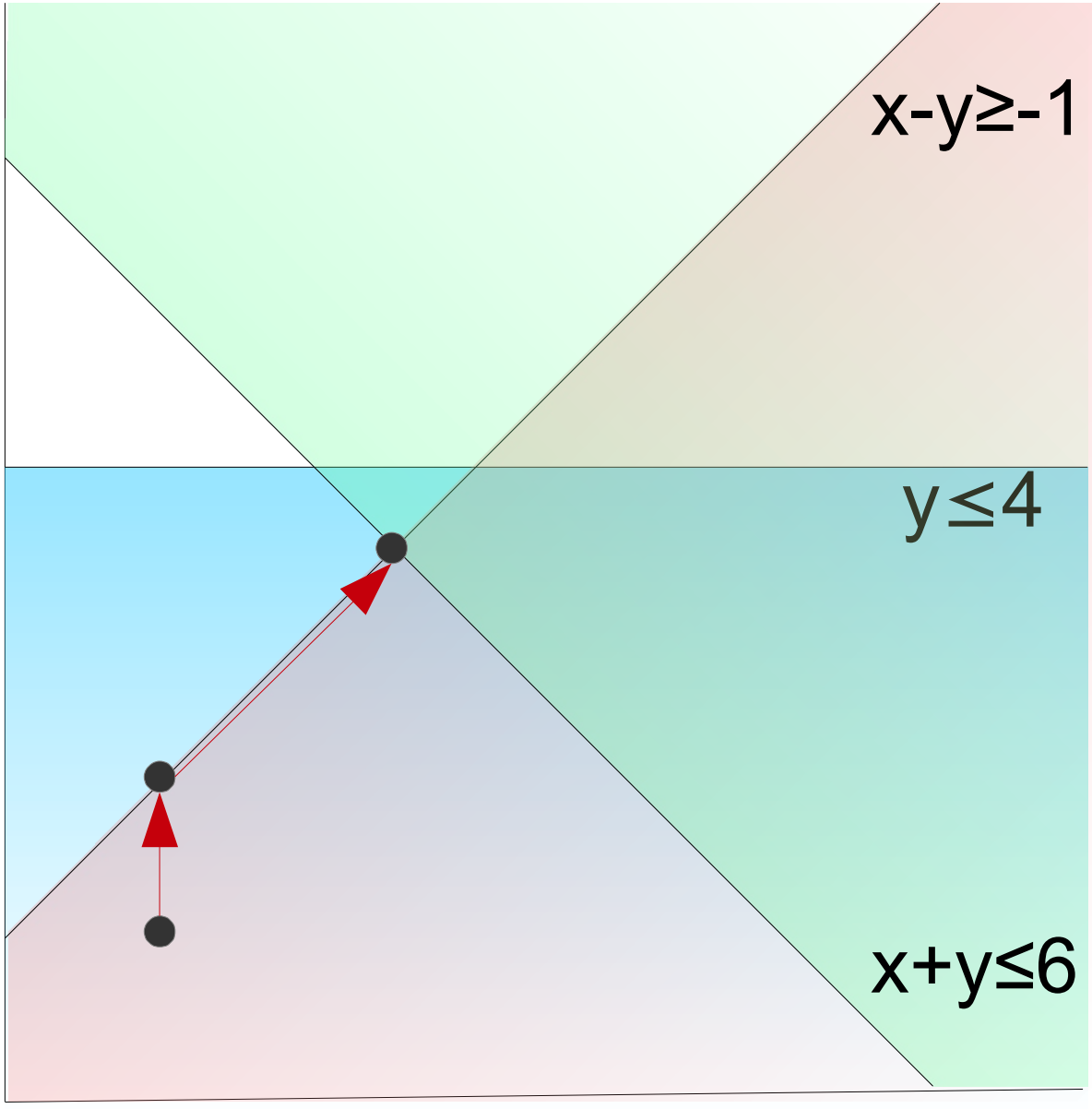
SOISimplex Search



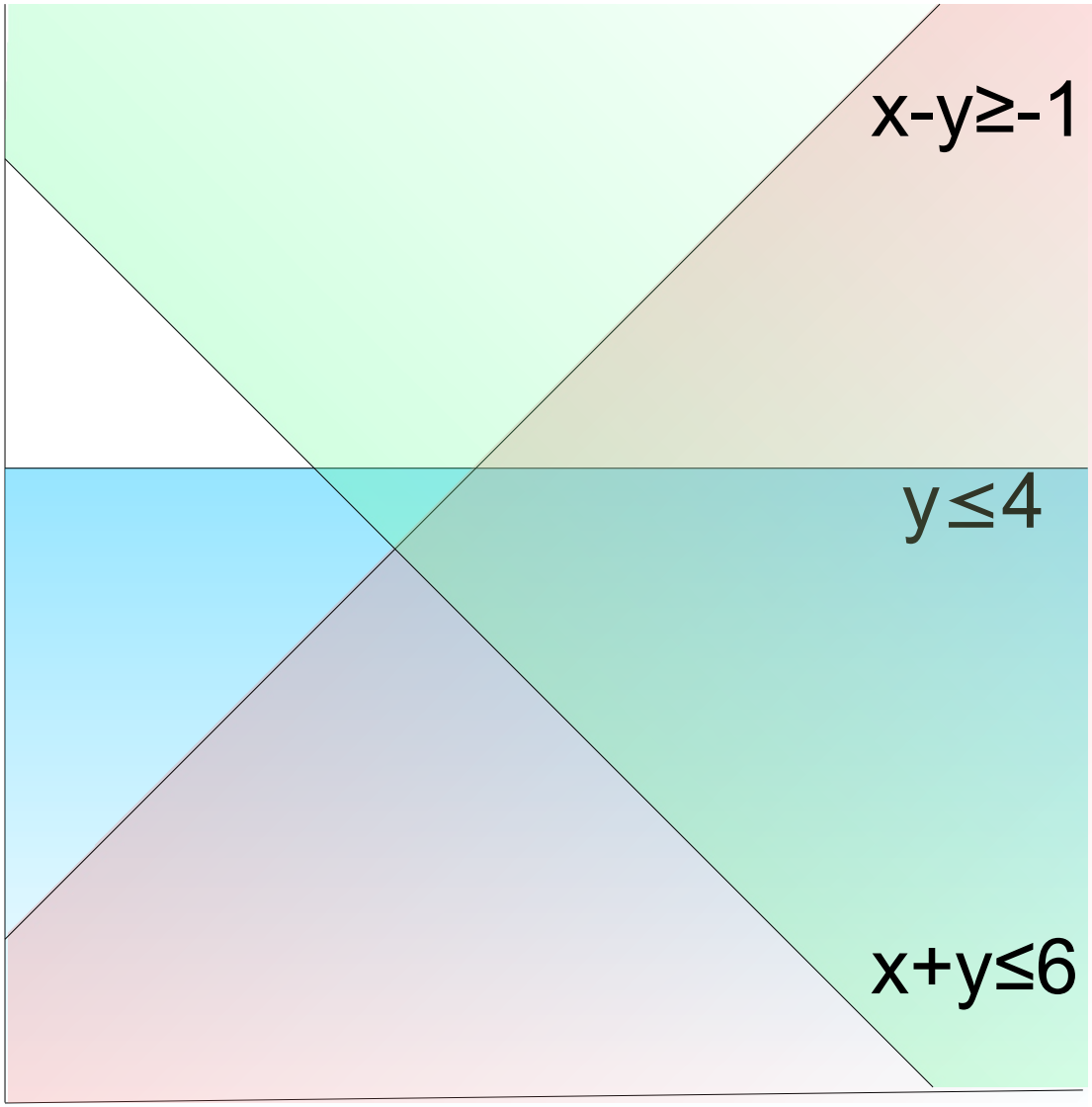
SOISimplex Search



SOISimplex Search

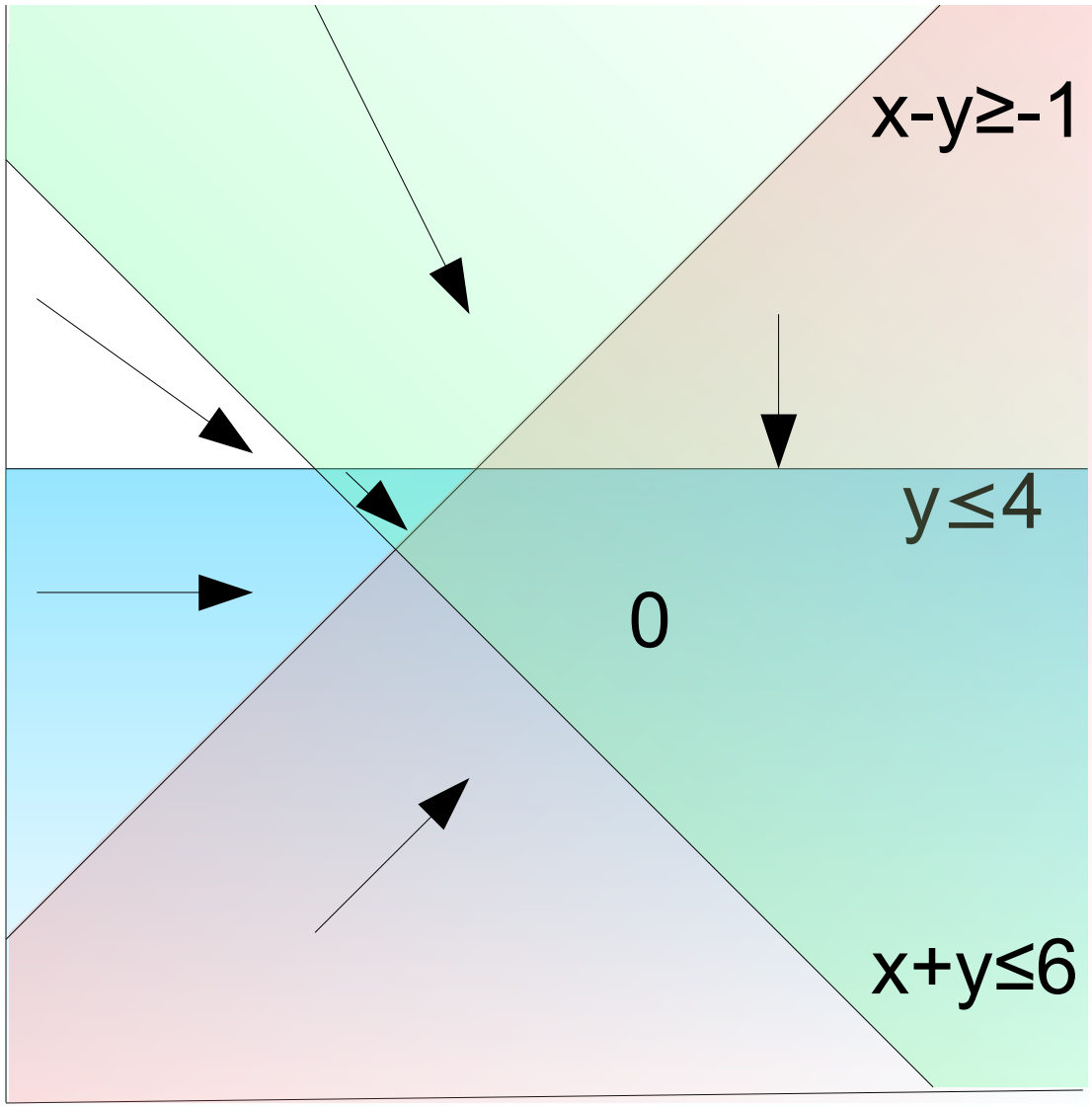


Direction of Vio($\mathcal{V}$)



$$\begin{cases} a(x)\ldots & a(x) > u(x) \\ 0 & \text{otherwise} \\ \ldots - a(x) & a(x) < l(x) \end{cases}$$

Direction of Vio($\mathcal{V}$)



$$\begin{cases} a(x)\ldots & a(x) > u(x) \\ 0 & \text{otherwise} \\ \ldots - a(x) & a(x) < l(x) \end{cases}$$

$\text{Vio}(\mathcal{V}) = 0$ iff a is sat

SOISimplex Highlevel

Rough Sketch

procedure SOISIMPLEX
  while Vio($\mathcal{V}$) is not at a minimum **do**
    select a variable x
    update x s.t. Vio($\mathcal{V}$) decreases and
      $a(b) \leftarrow u(x) \ldots$ for some $b \in \mathcal{B}$
    PIVOT(b, x)

Updating Nonbasics

Changing the assignment to $x \in \mathcal{N}$ is easy:

- $a(x) \mathrel{+}= \delta$
- for all $b \in \mathcal{B}$:
  $$a(b) \mathrel{+}= t_{b,x} \cdot \delta.$$

Updating Nonbasics

Changing the assignment to $x \in \mathcal{N}$ is easy:

- $a(x) \mathrel{+}= \delta$
- for all $b \in \mathcal{B}$:
  $$a(b) \mathrel{+}= t_{b,x} \cdot \delta.$$

Add the Invariant

The nonbasic variables satisfy their bounds.

PIVOT(b, x)

Move Variables In/Out of $\mathcal{B}$

Preconditions

Given b basic, x nonbasic, and $t_{b,x} \neq 0$,
PIVOT(b, x) makes b nonbasic and x basic.

Pivot(b, x)

Move Variables In/Out of $\mathcal{B}$

Preconditions

Given b basic, x nonbasic, and $t_{b,x} \neq 0$,
Pivot(b, x) makes b nonbasic and x basic.

- Take b's row

$$b = t_{b,x} \cdot x + \sum t_{b,z} \cdot z$$

- Solve for x

$$x = \frac{1}{t_{b,x}} \cdot b + \sum -\frac{t_{b,z}}{t_{b,x}} \cdot z$$

- Replace x everywhere else in T

Pivot(b, x)

Move Variables In/Out of $\mathcal{B}$

Preconditions

Given b basic, x nonbasic, and $t_{b,x} \neq 0$,
Pivot(b, x) makes b nonbasic and x basic.

- Take b's row

$$b = t_{b,x} \cdot x + \sum t_{b,z} \cdot z$$

- Solve for x

$$x = \frac{1}{t_{b,x}} \cdot b + \sum -\frac{t_{b,z}}{t_{b,x}} \cdot z$$

- Replace x everywhere else in T

Preserves Linear Subspace

Pivot(b, x) preserves $Ta = 0$.

Is $a(b)$ minimal?

Simplex's Core Proof Rule

$$b = \sum_{c>0} c \cdot x + \sum_{d<0} d \cdot y$$

Suppose each $a(x) = l(x)$ and $a(y) = u(x)$ (*)

$$b = \sum_{c>0} c \cdot x + \sum_{d<0} d \cdot y$$

Is $a(b)$ minimal?

Simplex's Core Proof Rule

$$b = \sum_{c>0} c \cdot x + \sum_{d<0} d \cdot y$$

Suppose each $a(x) = l(x)$ and $a(y) = u(x)$ (*)

$$b = \sum_{c>0} c \cdot x + \sum_{d<0} d \cdot y$$

$$\geq \sum_{c>0} c \cdot l(x) + \sum_{d<0} d \cdot u(x)$$

Is $a(b)$ minimal?

Simplex's Core Proof Rule

$$b = \sum_{c>0} c \cdot x + \sum_{d<0} d \cdot y$$

Suppose each $a(x) = l(x)$ and $a(y) = u(x)$ (*)

$$\begin{aligned} b &= \sum_{c>0} c \cdot x + \sum_{d<0} d \cdot y \\ &\geq \sum_{c>0} c \cdot l(x) + \sum_{d<0} d \cdot u(x) \\ &= \sum_{c>0} c \cdot a(x) + \sum_{d<0} d \cdot a(x) \end{aligned}$$

Is $a(b)$ minimal?

Simplex's Core Proof Rule

$$b = \sum_{c>0} c \cdot x + \sum_{d<0} d \cdot y$$

Suppose each $a(x) = l(x)$ and $a(y) = u(x)$ (*)

$$\begin{aligned} b &= \sum_{c>0} c \cdot x + \sum_{d<0} d \cdot y \\ &\geq \sum_{c>0} c \cdot l(x) + \sum_{d<0} d \cdot u(x) \\ &= \sum_{c>0} c \cdot a(x) + \sum_{d<0} d \cdot a(x) \\ &= a(b) \end{aligned}$$

Is $a(b)$ minimal?

Simplex's Core Proof Rule

$$b = \sum_{c>0} c \cdot x + \sum_{d<0} d \cdot y$$

Suppose each $a(x) = l(x)$ and $a(y) = u(x)$ (*)

$$\begin{aligned} b &= \sum_{c>0} c \cdot x + \sum_{d<0} d \cdot y \\ &\geq \sum_{c>0} c \cdot l(x) + \sum_{d<0} d \cdot u(x) \\ &= \sum_{c>0} c \cdot a(x) + \sum_{d<0} d \cdot a(x) \\ &= a(b) \\ b &\geq a(b) \end{aligned}$$

Is $a(b)$ minimal?

Simplex's Core Proof Rule

$$b = \sum_{c>0} c \cdot x + \sum_{d<0} d \cdot y$$

Suppose each $a(x) = l(x)$ and $a(y) = u(x)$ (*)

$$\begin{aligned} b &= \sum_{c>0} c \cdot x + \sum_{d<0} d \cdot y \\ &\geq \sum_{c>0} c \cdot l(x) + \sum_{d<0} d \cdot u(x) \\ &= \sum_{c>0} c \cdot a(x) + \sum_{d<0} d \cdot a(x) \\ &= a(b) \\ b &\geq a(b) \end{aligned}$$

If $a(b) > u(b)$, and (*) is true,

$$\{x \geq l(x)\} \cup \{y \leq u(x)\} \cup \{b \leq u(b)\} \models \bot$$

SOISimplex

Filling in the Sketch

procedure SOISIMPLEX
  while Vio($\mathcal{V}$) is not at a minimum **do**
    $\langle x, \delta, b \rangle \leftarrow$ SOISELECT()
    UPDATE(x, δ)
    PIVOT(b, x)
    Checks every row for a conflict
  if Vio($\mathcal{V}$) $= 0$ **then**
    return sat
  else
    return SoiQE()

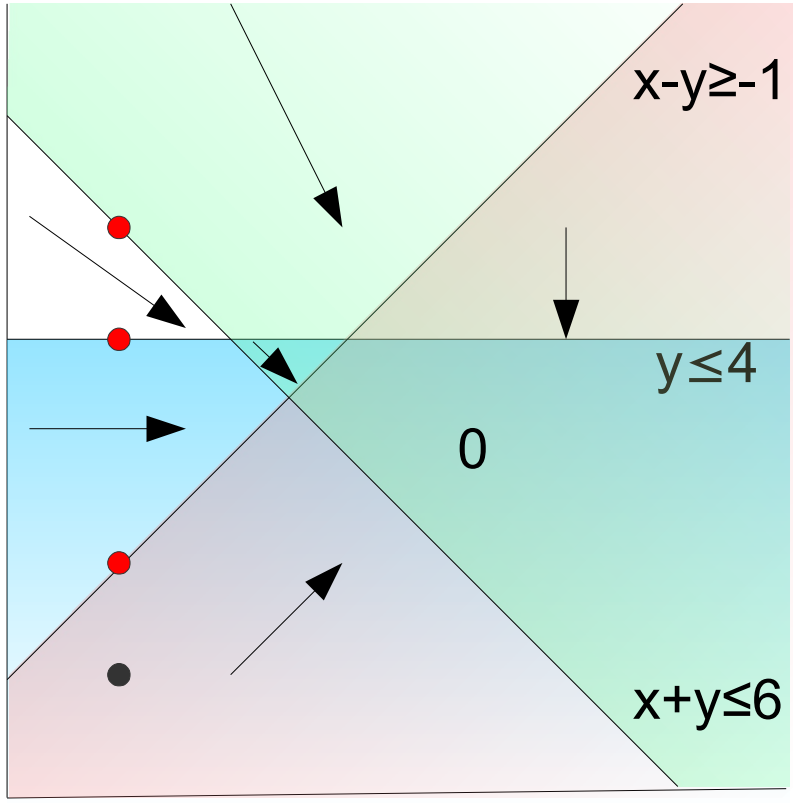
SOISimplex

Filling in the Sketch

procedure SOISIMPLEX
  while Vio($\mathcal{V}$) is not at a minimum **do**
    $\langle x, \delta, b \rangle \leftarrow$ SOISELECT()
    UPDATE(x, δ)
    PIVOT(b, x)
    Checks every row for a conflict ▷ Amortized $O(1)$
  if Vio($\mathcal{V}$) $= 0$ **then**
    return sat
  else
    return SoiQE()

Breakpoints

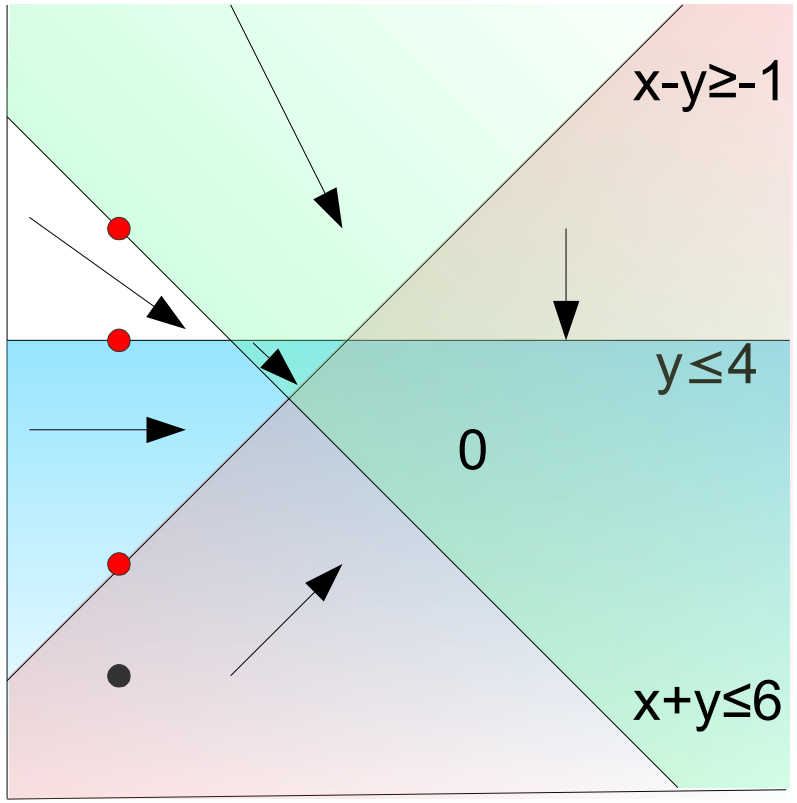
Where Vio($\mathcal{V}$) changes



$$\delta \in \left\{ \frac{a(b) - u(y)}{t_{b,y}}, \frac{a(b) - l(y)}{t_{b,y}}, \ldots, a(y) - u(y), a(y) - l(y) \right\}$$

Breakpoints

Where Vio($\mathcal{V}$) changes



$$\delta \in \{1, 3, 4\}$$

SOISelect()

procedure SOISelect

Select x on Vio($\mathcal{V}$) not at its bound

Compute breakpoints $\{\delta\}$

Compute Vio($\mathcal{V}$) post Update(x, δ) for each δ

return the δ and corresponding b with

the lowest Vio($\mathcal{V}$) post Update(x, δ)

SOISelect()

procedure SOISelect

Select x on Vio($\mathcal{V}$) not at its bound

Compute breakpoints $\{\delta\}$

Compute Vio($\mathcal{V}$) post Update(x, δ) for each δ

return the δ and corresponding b with

the lowest Vio($\mathcal{V}$) post Update(x, δ)

Vio($\mathcal{V}$) monotonically decreases!

SOISimplex()

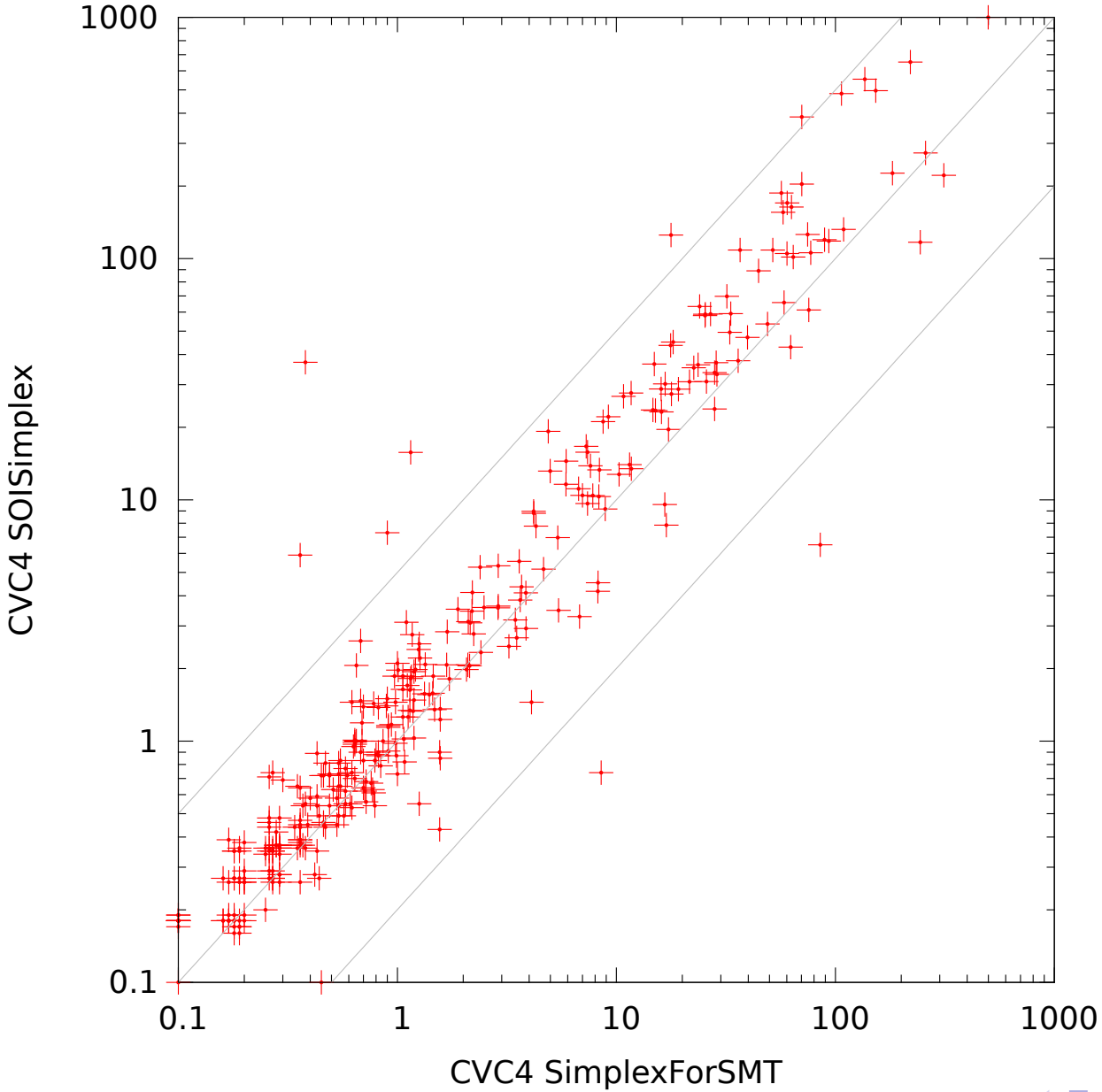
procedure SOISimplex
  while Vio($\mathcal{V}$) is not at a minimum **do**
    $tuplex, \delta, b \leftarrow$ SOISelect()
    Update(x, δ)
    Pivot(b, x)
    Checks rows for conflicts
  if Vio($\mathcal{V}$) $= 0$ **then**
    return sat
  else
    return SoiQE()

SoiQE()

- Proof a single Vio(b) is at a minimum generalizes to sets S
- Start with $S = \mathcal{V}$
- Greedily minimize S modulu this rule applying
- Conflict clause may not be minimal due to cancellations
- $|S| \leq 2 \implies$ minimal
- $|S| > 2$ is rare

SOISimplex versus SimplexForDPLL

SMTLIB QF_LRA



Below $x = y$ means SOISimplex is faster

Number Solved

	SOI	Z3	yices2	mathsat
QFLRA (634)	618	620	619	608
`latendresse` (18)	**18**	8	10	10

Conclusions

SOISimplex

- Minimize $\mathrm{Vio}(\mathcal{V})$
- More expensive analysis
- Fewer rounds on average*

SimplexForDPLL

- Greedily fixes $\mathrm{Vio}(b) > 0$
- Cheaper analysis
- Faster on easy instances

Thanks for your attention.

The Structure of the Problem

- $\sim 95\%$ of calls to theory solver need 0 simplex round
- $\sim 1.8\%$ of calls to the theory solver need 1 simplex round
- $\sim 2.5\%$ of calls to the theory solver need $[2 - 10]$ simplex rounds
- This is about 50% of the simplex rounds in total