A Short Course In Nonlinear Optimization

Andreas Wächter

IBM T.J. Watson Research Center

andreasw@us.ibm.com

D.E.I.S.

University of Bologna

March 19-23, 2007

Introduction

Nonlinear optimization problems:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ s.t. \quad & c(x) = 0 \\ & d(x) \geq 0 \end{aligned}$$

x	Variables
$f(x) : \mathbb{R}^n \longrightarrow \mathbb{R}$	Objective function
$c(x) : \mathbb{R}^n \longrightarrow \mathbb{R}^m$	Equality constraints
$d(x) : \mathbb{R}^n \longrightarrow \mathbb{R}^p$	Inequality constraints

- We assume that all functions are C^2, but can be nonlinear
- Interested in large problem ($n = 100, \ldots, 10^7$)

Introduction

Nonlinear optimization problems:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ s.t. \quad & c(x) = 0 \\ & d(x) \geq 0 \end{aligned}$$

x	Variables
$f(x) : \mathbb{R}^n \longrightarrow \mathbb{R}$	Objective function
$c(x) : \mathbb{R}^n \longrightarrow \mathbb{R}^m$	Equality constraints
$d(x) : \mathbb{R}^n \longrightarrow \mathbb{R}^p$	Inequality constraints

- We assume that all functions are C^2, but can be nonlinear
- Interested in large problem ($n = 100, \ldots, 10^7$)
- MINLP: Some of the x variables are required to be integer

Introduction

Nonlinear optimization problems:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ s.t. \quad & c(x) = 0 \\ & d(x) \geq 0 \end{aligned}$$

x	Variables
$f(x) : \mathbb{R}^n \longrightarrow \mathbb{R}$	Objective function
$c(x) : \mathbb{R}^n \longrightarrow \mathbb{R}^m$	Equality constraints
$d(x) : \mathbb{R}^n \longrightarrow \mathbb{R}^p$	Inequality constraints

- We assume that all functions are C^2, but can be nonlinear
- Interested in large problem ($n = 100, \ldots, 10^7$)
- MINLP: Some of the x variables are required to be integer
- Goal of this course: Provide understanding, how
  - non-convex NLPs (software: IPOPT)
  - convex MINLP (software: BONMIN)

  can be solved numerically
- This is not a comprehensive course on (MI)NLP

Outline

1. Unconstrained Optimization

$$\min_{x \in \mathbb{R}^n} f(x)$$

- Optimality conditions
- Newton's method
- Line search

Outline

1. Unconstrained Optimization

$$\min_{x \in \mathbb{R}^n} f(x)$$

- Optimality conditions
- Newton's method
- Line search

2. Equality Constrained Optimization

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ s.t. \quad & c(x) = 0 \end{aligned}$$

- Optimality conditions
- Sequential quadratic programming
- Merit functions and filter methods

Outline

1. Unconstrained Optimization

$$\min_{x \in \mathbb{R}^n} f(x)$$

- Optimality conditions
- Newton's method
- Line search

2. Equality Constrained Optimization

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ s.t. \quad & c(x) = 0 \end{aligned}$$

- Optimality conditions
- Sequential quadratic programming
- Merit functions and filter methods

3. Inequality Constrained Optimization

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ s.t. \quad & c(x) = 0 \\ & d(x) \geq 0 \end{aligned}$$

- Optimality conditions
- Active set methods
- Interior point methods

Outline (cont.)

4. IPOPT: Open source software for non-convex NLP

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ s.t. \quad & c(x) = 0 \\ & x_L \leq x \leq x_U \end{aligned}$$

- Algorithm details
- Implementation
- Adaptive choice of barrier parameter

Outline (cont.)

4. IPOPT: Open source software for non-convex NLP

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ s.t. \quad & c(x) = 0 \\ & x_L \leq x \leq x_U \end{aligned}$$

- Algorithm details
- Implementation
- Adaptive choice of barrier parameter

5. Industrial applications for NLP

- Examples
- Circuit tuning

Outline (cont.)

4. IPOPT: Open source software for non-convex NLP

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ s.t. \quad & c(x) = 0 \\ & x_L \leq x \leq x_U \end{aligned}$$

- Algorithm details
- Implementation
- Adaptive choice of barrier parameter

5. Industrial applications for NLP

- Examples
- Circuit tuning

6. (Convex) Mixed-Integer Nonlinear Programming (MINLP)

$$\begin{aligned} \min_{x \in \mathbb{R}^n, y \in \mathbb{R}^p} \quad & f(x, y) \\ s.t. \quad & c(x, y) = 0 \\ & x_L \leq x \leq x_U \\ & y \in \{0, 1\}^p \end{aligned}$$

- Branch and bound
- Outer approximation
- A new hybrid approach
- The BONMIN package

Unconstrained Optimization

Unconstrained Optimization

$$\min_{x \in \mathbb{R}^n} f(x)$$

For this short course we assume:

- Smoothness:
  - $f(x) : \mathbb{R}^n \longrightarrow \mathbb{R}$ is C^2
- Computational availability of functions and derivatives:
  - $f(x_k)$, $\nabla f(x_k)$ and $\nabla^2 f(x_k)$ can be evaluated for given x_k

Unconstrained Optimization

$$\min_{x \in \mathbb{R}^n} f(x)$$

For this short course we assume:

- Smoothness:
  - $f(x) : \mathbb{R}^n \longrightarrow \mathbb{R}$ is C^2
- Computational availability of functions and derivatives:
  - $f(x_k)$, $\nabla f(x_k)$ and $\nabla^2 f(x_k)$ can be evaluated for given x_k
- Not covered here:
  - Nonsmooth optimization, e.g., $f(x)$ as "kinks"
  - Derivative-Free Optimization
    - $\nabla f(x_k)$ and $\nabla^2 f(x_k)$ cannot be computed
  - Quasi-Newton Methods
    - $\nabla^2 f(x_k)$ cannot be computed
    - approximate $W_k \approx \nabla^2 f(x_k)$

Optimality Conditions

$$\min_{x \in \mathbb{R}^n} f(x)$$

Characterizing "local solutions" x_*: $f(x) \geq f(x_*)$ for x in neighborhood of x_*.

Optimality Conditions

$$\min_{x \in \mathbb{R}^n} f(x)$$

Characterizing "local solutions" x_*: $f(x) \geq f(x_*)$ for x in neighborhood of x_*.

- 1st order necessary condition:

$$x_* \text{ is local solution of } f(x) \quad \Longrightarrow \quad \nabla f(x_*) = 0$$

At x_* there is no "descent direction" (up to first order).
Such x_* is called "stationary point".

Optimality Conditions

$$\min_{x \in \mathbb{R}^n} f(x)$$

Characterizing "local solutions" x_*: $f(x) \geq f(x_*)$ for x in neighborhood of x_*.

- 1st order necessary condition:

  $$x_* \text{ is local solution of } f(x) \quad \Longrightarrow \quad \nabla f(x_*) = 0$$

  At x_* there is no "descent direction" (up to first order).
  Such x_* is called "stationary point".

- 2nd order necessary condition:

  $$\nabla^2 f(x_*) \succeq 0 \qquad \text{(positive semi-definite)}$$

Optimality Conditions

$$\min_{x \in \mathbb{R}^n} f(x)$$

Characterizing "local solutions" x_*: $f(x) \geq f(x_*)$ for x in neighborhood of x_*.

- 1st order necessary condition:

  $$x_* \text{ is local solution of } f(x) \quad \Longrightarrow \quad \nabla f(x_*) = 0$$

  At x_* there is no "descent direction" (up to first order).
  Such x_* is called "stationary point".

- 2nd order necessary condition:

  $$\nabla^2 f(x_*) \succeq 0 \qquad \text{(positive semi-definite)}$$

- 2nd order sufficient condition:

  $$\left.\begin{array}{l} \nabla f(x_*) = 0 \\ \nabla^2 f(x_*) \succ 0 \end{array}\right\} \quad \Longrightarrow \quad x_* \text{ is local solution of } f(x)$$

Comments

- If $f(x)$ is convex:
  - Every stationary point is a global solution
  - There are no maxima or saddle point points
  - 1st order optimality conditions are sufficient for x_* being a global solution

Comments

- If $f(x)$ is convex:
  - Every stationary point is a global solution
  - There are no maxima or saddle point points
  - 1st order optimality conditions are sufficient for x_* being a global solution
- If $f(x)$ is non-convex:
  - Also maxima and saddle points may exist which satisfy $\nabla f(x) = 0$
  - Might have many local solutions x_* with different $f(x_*)$

    (we only try to find local solutions here)

Comments

- If $f(x)$ is convex:
  - Every stationary point is a global solution
  - There are no maxima or saddle point points
  - 1st order optimality conditions are sufficient for x_* being a global solution
- If $f(x)$ is non-convex:
  - Also maxima and saddle points may exist which satisfy $\nabla f(x) = 0$
  - Might have many local solutions x_* with different $f(x_*)$

    (we only try to find local solutions here)

Usually, numerical methods for finding a local solution

- seek point x_* with $\nabla f(x_*) = 0$
- try to avoid convergence to maxima

Newton's Method

- Numerical method for solving nonlinear system of equations

$$F(x_*) = 0$$

- Iterative method:
  - Given starting point x_0
  - Generate sequence $\{x_k\}$ of improved "guesses" of the solution

Newton's Method

- Numerical method for solving nonlinear system of equations

$$F(x_*) = 0$$

- Iterative method:
  - Given starting point x_0
  - Generate sequence $\{x_k\}$ of improved "guesses" of the solution
- Step: Given x_k, set up local linear "model" of $F(x)$

$$m_k(x) = F(x_k) + \nabla F(x_k)^T (x - x_k)$$

From Taylor expansion: $\|m_k(x) - F(x)\| = O(\|x - x_k\|^2)$

Newton's Method

- Numerical method for solving nonlinear system of equations

$$F(x_*) = 0$$

- Iterative method:
  - Given starting point x_0
  - Generate sequence $\{x_k\}$ of improved "guesses" of the solution
- Step: Given x_k, set up local linear "model" of $F(x)$

$$m_k(x) = F(x_k) + \nabla F(x_k)^T (x - x_k)$$

From Taylor expansion: $\|m_k(x) - F(x)\| = O(\|x - x_k\|^2)$

- Find solution $\tilde{x}_{k+1} = x_k + d_k$ of model $m_k(x)$

$$0 = m_k(x) = F(x_k) + \nabla F(x_k)^T (x - x_k)$$

$$\Longrightarrow \qquad d_k = -[\nabla F(x_k)]^{-T} F(x_k)$$

Newton's Method

- Numerical method for solving nonlinear system of equations

$$\boxed{F(x_*) = 0}$$

- Iterative method:
  - Given starting point x_0
  - Generate sequence $\{x_k\}$ of improved "guesses" of the solution
- Step: Given x_k, set up local linear "model" of $F(x)$

$$\boxed{m_k(x) = F(x_k) + \nabla F(x_k)^T (x - x_k)}$$

From Taylor expansion: $\|m_k(x) - F(x)\| = O(\|x - x_k\|^2)$

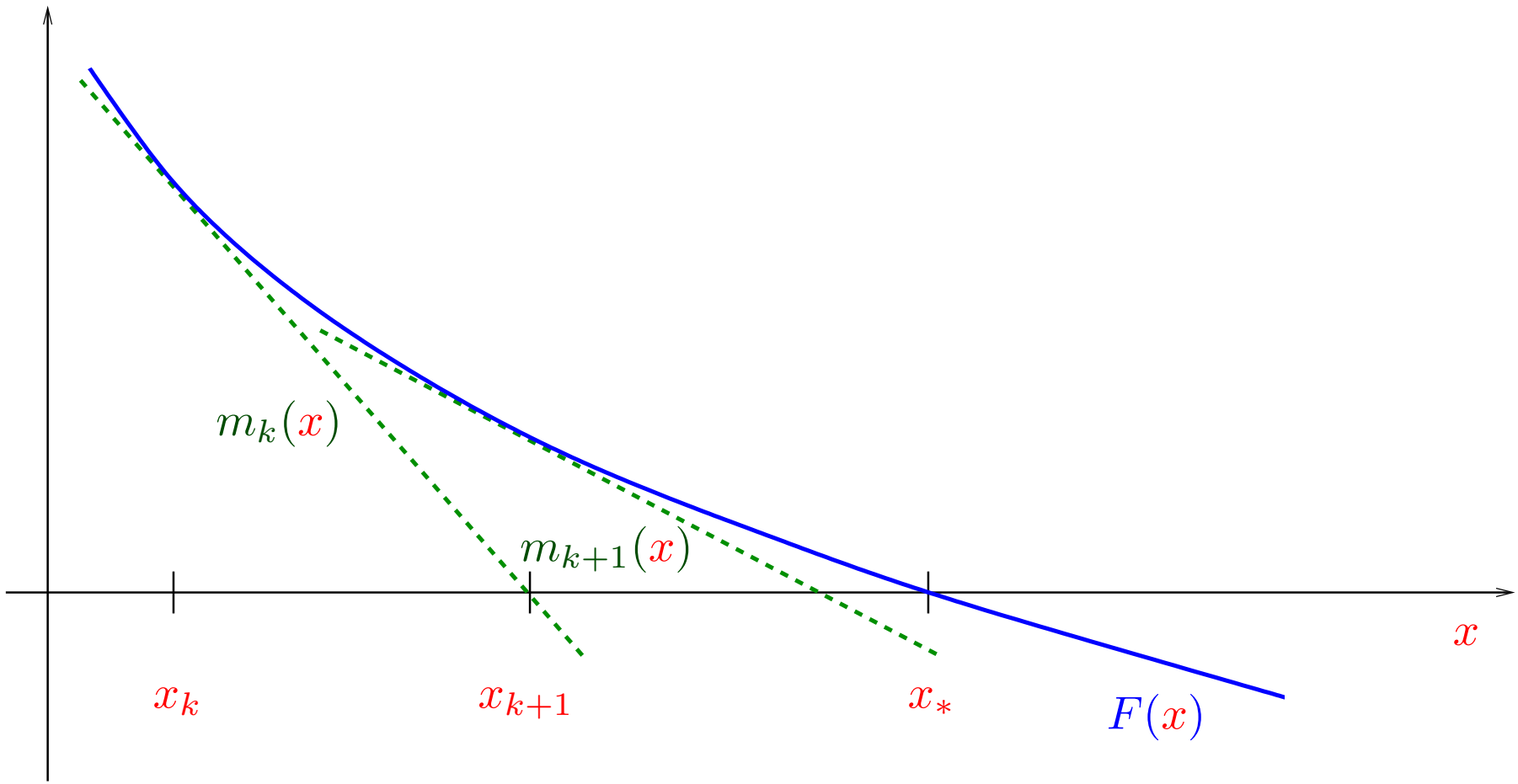
- Find solution $\tilde{x}_{k+1} = x_k + d_k$ of model $m_k(x)$

$$0 = m_k(x) = F(x_k) + \nabla F(x_k)^T (x - x_k)$$

$$\Longrightarrow \qquad d_k = -[\nabla F(x_k)]^{-T} F(x_k)$$

- Set $x_{k+1} = x_k + d_k$

Newton's Method For Equations



Newton's Method For Optimization

- Numerical method for solving nonlinear system of equations

$$F(x_*) = 0$$

- Iterative method:
  - Given starting point x_0
  - Generate sequence $\{x_k\}$ of improved "guesses" of the solution
- Step: Given x_k, set up local linear "model" of $F(x)$

$$m_k(x) = F(x_k) + \nabla F(x_k)^T (x - x_k)$$

From Taylor expansion: $\|m_k(x) - F(x)\| = O(\|x - x_k\|^2)$

- Find solution $\tilde{x}_{k+1} = x_k + d_k$ of model $m_k(x)$

$$0 = m_k(x) = F(x_k) + \nabla F(x_k)^T (x - x_k)$$

$$\Longrightarrow \qquad d_k = -[\nabla(x_k)]^{-T} F(x_k)$$

- Set $x_{k+1} = x_k + d_k$

Newton's Method For Optimization

- Numerical method for solving nonlinear optimization problem

$$\min_{x \in \mathbb{R}^n} f(x) \qquad \nabla f(x_*) = 0$$

- Iterative method:
  - Given starting point x_0
  - Generate sequence $\{x_k\}$ of improved "guesses" of the solution
- Step: Given x_k, set up local quadratic "model" of $f(x)$

$$m_k(x) = f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{1}{2}(x - x_k)^T \nabla^2 f(x_k)(x - x_k)$$

From Taylor expansion: $\|\nabla m_k(x) - \nabla f(x)\| = O(\|x - x_k\|^2)$

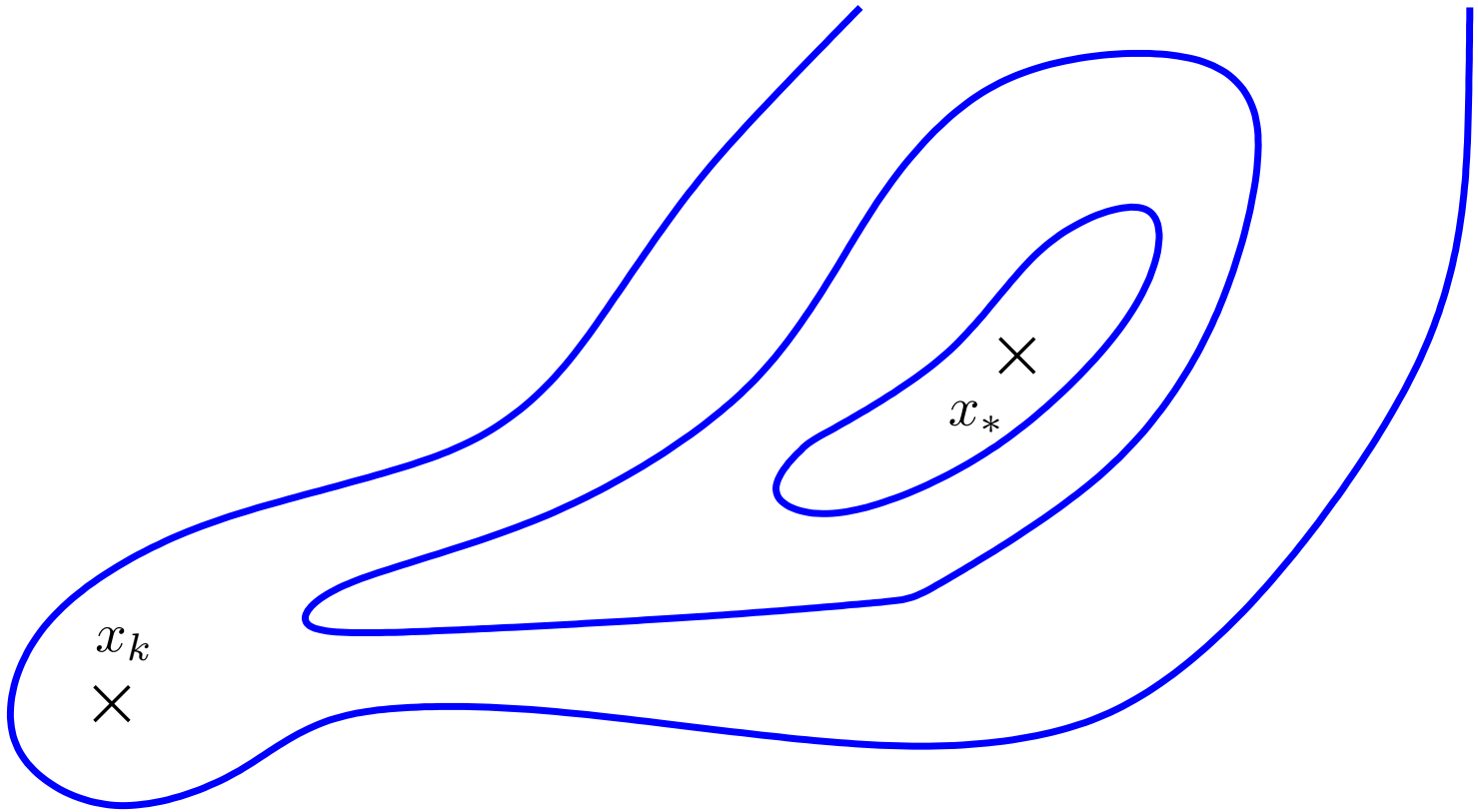
- Find minimizer $\tilde{x}_{k+1} = x_k + d_k$ of model $m_k(x)$ (assume convex)

$$0 = \nabla m_k(x) = \nabla f(x_k) + \nabla^2 f(x_k)(x - x_k)$$

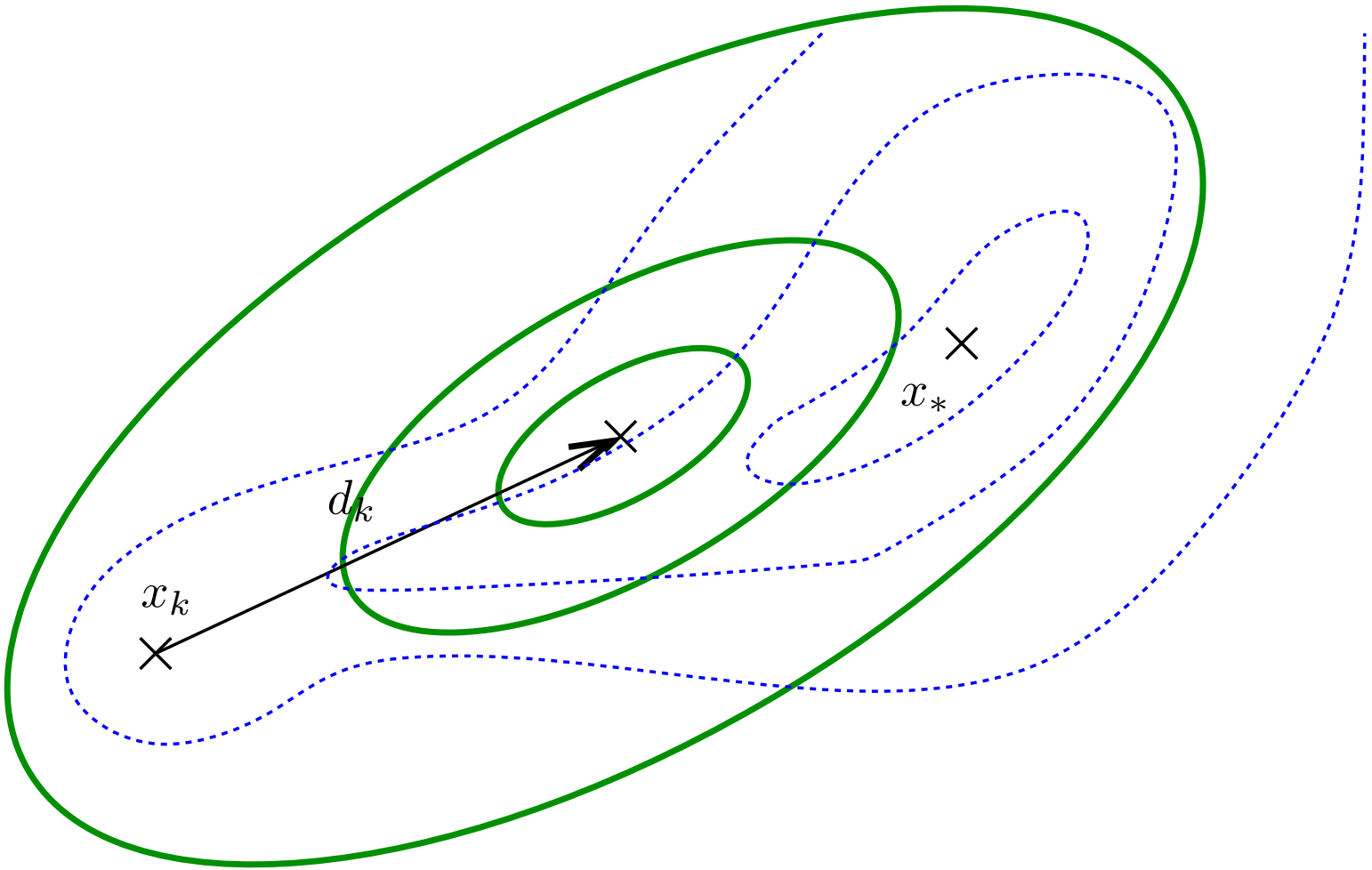
$$\Longrightarrow \qquad d_k = -[\nabla^2 f(x_k)]^{-1} \nabla f(x_k)$$

- Set $x_{k+1} = x_k + d_k$

Newton's Method For Optimization



Newton's Method For Optimization



Fast Congergence of Newton's Method

Assume:

- x_* is second order sufficient solution for $\min_{x \in \mathbb{R}^n} f(x)$
- x_0 is sufficiently close to x_*
- $x_{k+1} = x_k - [\nabla^2 f(x_k)]^{-1} \nabla f(x_k)$

Then:

- $\lim_k x_k = x_*$
- $\{x_k\}$ converges quadratically to x_*, i.e.,

$$\exists\, C > 0 : \quad \|x_{k+1} - x_*\| \leq C \|x_k - x_*\|^2$$

Fast Congergence of Newton's Method

Assume:

- x_* is second order sufficient solution for $\min_{x \in \mathbb{R}^n} f(x)$
- x_0 is sufficiently close to x_*
- $x_{k+1} = x_k - [\nabla^2 f(x_k)]^{-1} \nabla f(x_k)$

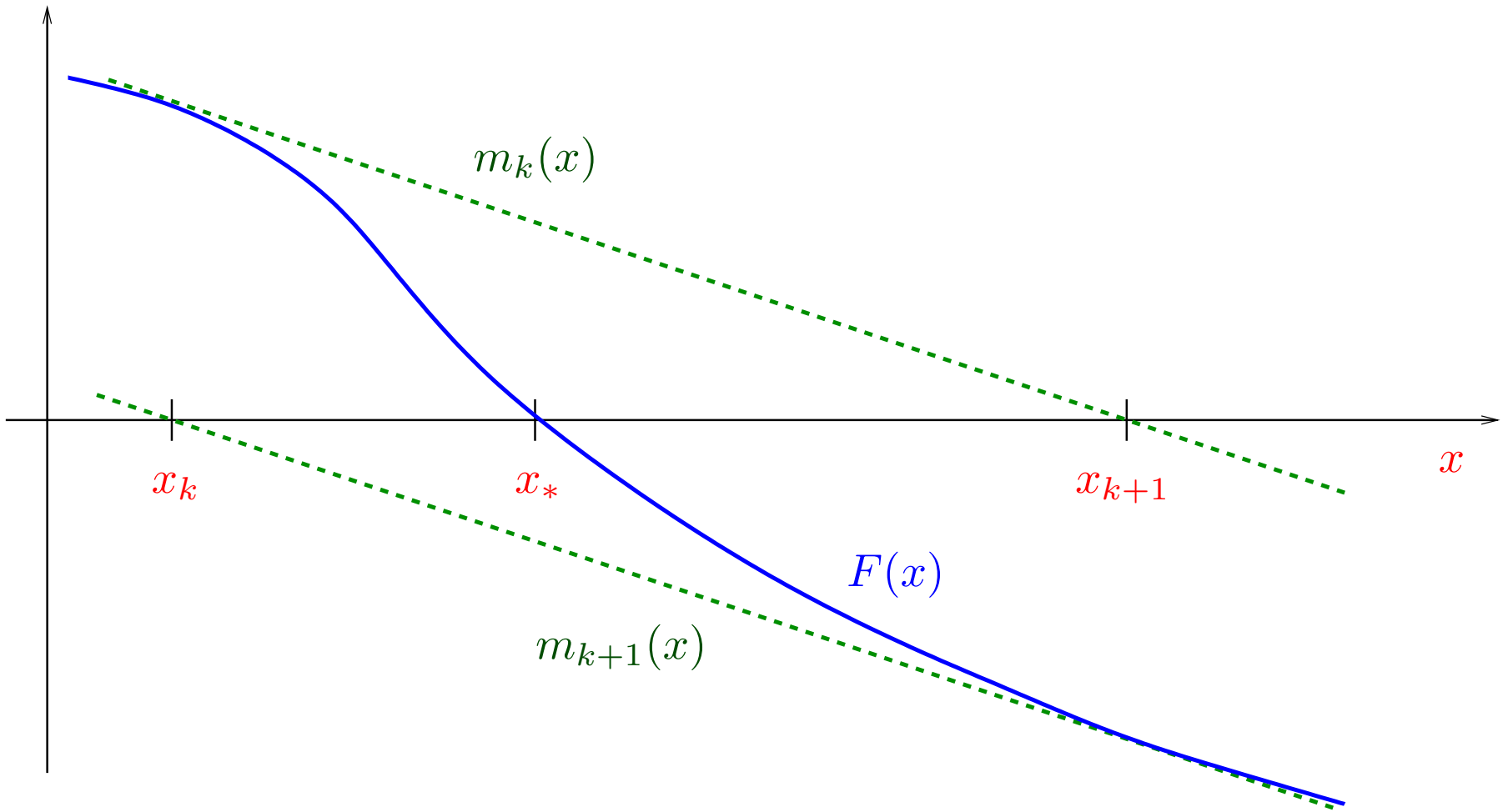
Then:

- $\lim_k x_k = x_*$
- $\{x_k\}$ converges quadratically to x_*, i.e.,

$$\exists\, C > 0 : \quad \|x_{k+1} - x_*\| \leq C \|x_k - x_*\|^2$$

But what happens if x_0 is not close to x_*?

What can go wrong



Line Search

- Want to guarantee "global convergence" (convergence from any x_0)
- Consider Newton direction $d_k^N = -[\nabla^2 f(x_k)]^{-1} \nabla f(x_k)$
- How does $f(x)$ behave in direction d_k^N?

Line Search

- Want to guarantee "global convergence" (convergence from any x_0)
- Consider Newton direction $d_k^N = -[\nabla^2 f(x_k)]^{-1} \nabla f(x_k)$
- How does $f(x)$ behave in direction d_k^N?
  - Define $\Phi_k(\alpha) := f(x_k + \alpha d_k^N)$
  - Then

$$\begin{aligned} \Phi_k'(\alpha) &= \nabla f(x_k)^T d_k^N \\ &= -\nabla f(x_k)^T [\nabla^2 f(x_k)]^{-1} \nabla f(x_k) \\ &< 0 \end{aligned}$$

    if $\nabla^2 f(x_k) \succ 0$ (assume for now).
  - Therefore, d_k^N is a "descent direction" for $f(x)$

Line Search

- Want to guarantee "global convergence" (convergence from any x_0)
- Consider Newton direction $d_k^N = -[\nabla^2 f(x_k)]^{-1} \nabla f(x_k)$
- How does $f(x)$ behave in direction d_k^N?
  - Define $\Phi_k(\alpha) := f(x_k + \alpha d_k^N)$
  - Then

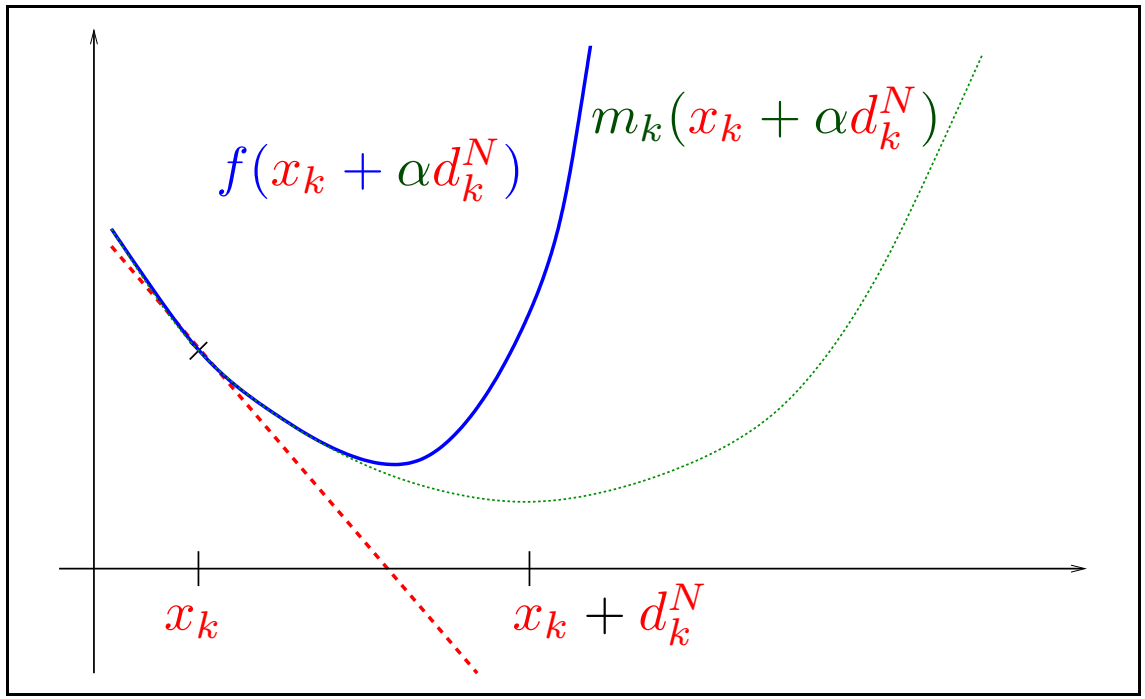
$$\begin{aligned} \Phi_k'(\alpha) &= \nabla f(x_k)^T d_k^N \\ &= -\nabla f(x_k)^T [\nabla^2 f(x_k)]^{-1} \nabla f(x_k) \\ &< 0 \end{aligned}$$

    if $\nabla^2 f(x_k) \succ 0$ (assume for now).
  - Therefore, d_k^N is a "descent direction" for $f(x)$
  - Idea: Choose "step size" $\alpha_k \in (0, 1]$, so that

$$\boxed{x_{k+1} = x_k + \alpha_k d_k^N}$$

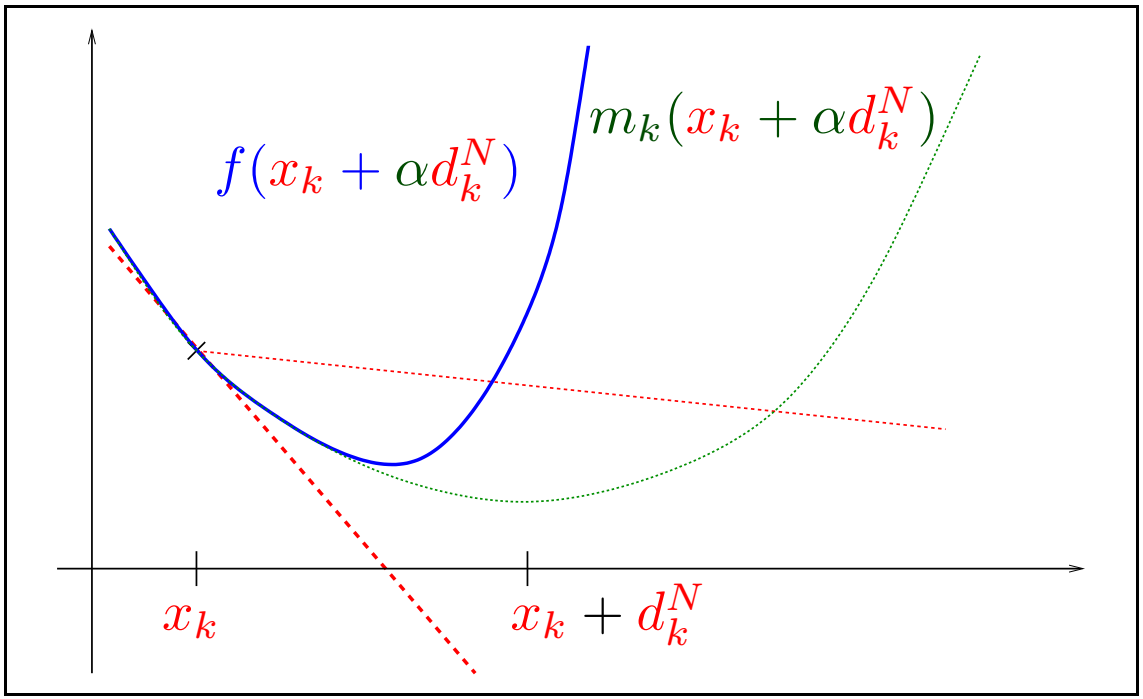
    gives "sufficient decrease" in $f(x)$

Armijo Condition



- We have linearization $f(x_k + \alpha d_k^N) \approx f(x_k) + \alpha \nabla f(x_k)^T d_k^N$

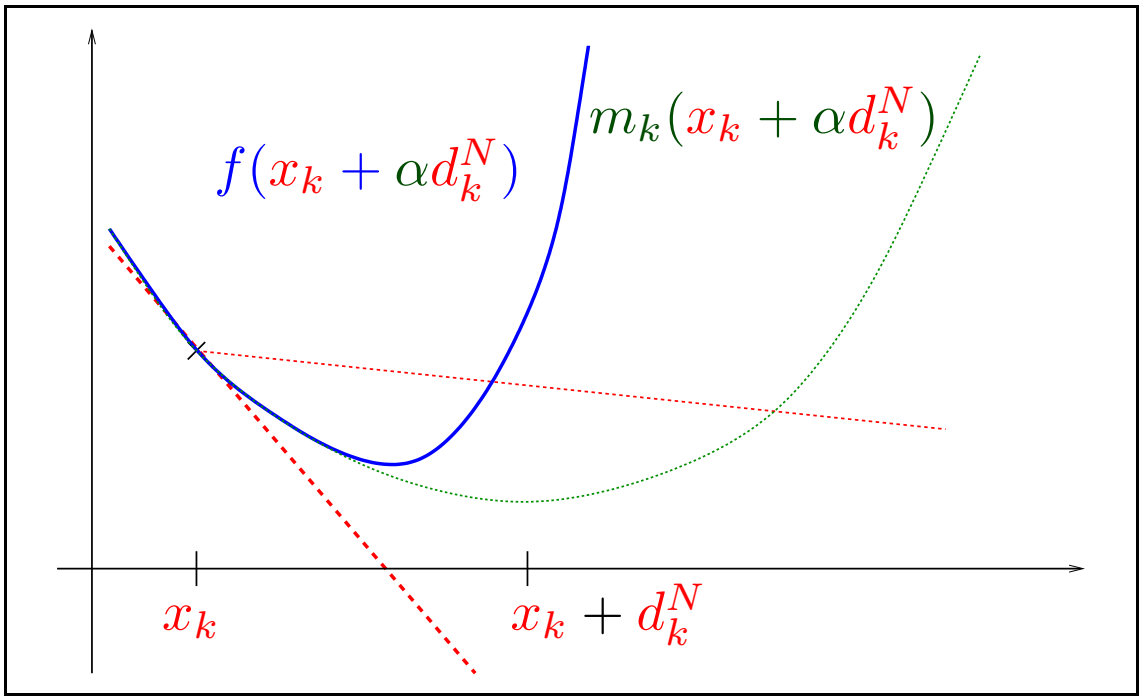
Armijo Condition



- We have linearization $f(x_k + \alpha d_k^N) \approx f(x_k) + \alpha \nabla f(x_k)^T d_k^N$
- Relax slope, and ask for decrease w.r.t. "relaxed linearization"

$$\boxed{f(x_k + \alpha_k d_k^N) \leq f(x_k) + \eta \alpha_k \nabla f(x_k)^T d_k^N} \qquad \eta \in (0, 1)$$

Armijo Condition



- We have linearization $f(x_k + \alpha d_k^N) \approx f(x_k) + \alpha \nabla f(x_k)^T d_k^N$
- Relax slope, and ask for decrease w.r.t. "relaxed linearization"

$$\boxed{f(x_k + \alpha_k d_k^N) \leq f(x_k) + \eta \alpha_k \nabla f(x_k)^T d_k^N} \qquad \eta \in (0,1)$$

- Backtracking line search:
  - Try $\alpha_k^l = \left(\frac{1}{2}\right)^l$ for $l = 0, 1, 2, \ldots$, until Armijo condition holds

Nonconvex Problems

- We need that $d_k = -[\nabla^2 f(x_k)]^{-1} \nabla f(x_k)$ is a descent direction, i.e.,

$$\Phi_k(\alpha)' = -\nabla f(x_k)^T [\nabla^2 f(x_k)]^{-1} \nabla f(x_k) < 0$$

Nonconvex Problems

- We need that $d_k = -W_k^{-1}\nabla f(x_k)$ is a descent direction, i.e.,

$$\Phi_k(\alpha)' = -\nabla f(x_k)^T W_k^{-1}\nabla f(x_k) < 0$$

1. Choose $\delta \geq 0$ so that

$$W_k = \nabla^2 f(x_k) + \delta I \succ 0$$

and perform "trial factorizations" (e.g. Cholesky) until $W_k \succ 0$. ("Bend into directions of steepest descent.")

Nonconvex Problems

- We need that $d_k = -W_k^{-1}\nabla f(x_k)$ is a descent direction, i.e.,

$$\Phi_k(\alpha)' = -\nabla f(x_k)^T W_k^{-1}\nabla f(x_k) < 0$$

1. Choose $\delta \geq 0$ so that

$$W_k = \nabla^2 f(x_k) + \delta I \succ 0$$

   and perform "trial factorizations" (e.g. Cholesky) until $W_k \succ 0$. ("Bend into directions of steepest descent.")

2. "Modified Cholesky factorization" (change non-positive pivots)

Nonconvex Problems

- We need that $d_k = -W_k^{-1} \nabla f(x_k)$ is a descent direction, i.e.,

$$\Phi_k(\alpha)' = -\nabla f(x_k)^T W_k^{-1} \nabla f(x_k) < 0$$

1. Choose $\delta \geq 0$ so that

$$W_k = \nabla^2 f(x_k) + \delta I \succ 0$$

   and perform "trial factorizations" (e.g. Cholesky) until $W_k \succ 0$. ("Bend into directions of steepest descent.")

2. "Modified Cholesky factorization" (change non-positive pivots)

3. Try to identify "directions d_k of negative curvature" with ${d_k}^T \nabla^2 f(x_k) d_k < 0$ and perform line search into that direction

Nonconvex Problems

- We need that $d_k = -W_k^{-1}\nabla f(x_k)$ is a descent direction, i.e.,

$$\Phi_k(\alpha)' = -\nabla f(x_k)^T W_k^{-1}\nabla f(x_k) < 0$$

1. Choose $\delta \geq 0$ so that

$$W_k = \nabla^2 f(x_k) + \delta I \succ 0$$

   and perform "trial factorizations" (e.g. Cholesky) until $W_k \succ 0$. ("Bend into directions of steepest descent.")

2. "Modified Cholesky factorization" (change non-positive pivots)

3. Try to identify "directions d_k of negative curvature" with ${d_k}^T\nabla^2 f(x_k)d_k < 0$ and perform line search into that direction

- Important alternative approach: Trust region methods (not covered)

Global Convergence Proof (1)

Method:

$$d_k = -W_k^{-1} \nabla f(x_k) \qquad f(x_{k+1}) - f(x_k) \leq \eta \alpha_k \nabla f(x_k)^T d_k$$

Global Convergence Proof (1)

Method:

$$d_k = -W_k^{-1} \nabla f(x_k) \qquad f(x_{k+1}) - f(x_k) \leq \eta \alpha_k \nabla f(x_k)^T d_k$$

Assume:

$$\exists \epsilon_1, \epsilon_2 > 0,\ f_{\text{low}},\ M : \epsilon_1 I \prec W_k \prec \epsilon_2 I,\ f(x) \geq f_{\text{low}},\ \|\nabla^2 f(x)\| \leq M$$

Global Convergence Proof (1)

Method:

$$d_k = -W_k^{-1}\nabla f(x_k) \qquad f(x_{k+1}) - f(x_k) \leq \eta\alpha_k \nabla f(x_k)^T d_k$$

Assume:

$$\exists \epsilon_1, \epsilon_2 > 0,\ f_{\text{low}},\ M : \epsilon_1 I \prec W_k \prec \epsilon_2 I,\ f(x) \geq f_{\text{low}},\ \|\nabla^2 f(x)\| \leq M$$

Have:

$$\begin{aligned}
\|d_k\| &\leq \epsilon_1^{-1}\|\nabla f(x_k)\| && (\xi^{\alpha_k} \in [x_k, x_k + d_k]) \\
f(x_k + \alpha d_k) - f(x_k) &= \alpha \nabla f(x_k)^T d_k + \alpha^2 {d_k}^T \nabla^2 f(\xi^{\alpha_k}) d_k \\
&\leq \alpha \nabla f(x_k)^T d_k + \alpha^2 M \|d_k\|^2 \\
\nabla f(x_k)^T d_k &= -\nabla f(x_k)^T W_k^{-1} \nabla f(x_k) \leq -\epsilon_2^{-1}\|\nabla f(x_k)\|^2
\end{aligned}$$

Global Convergence Proof (1)

Method: $\boxed{d_k = -W_k^{-1}\nabla f(x_k) \qquad f(x_{k+1}) - f(x_k) \leq \eta\alpha_k \nabla f(x_k)^T d_k}$

Assume:

$$\exists \epsilon_1, \epsilon_2 > 0,\ f_{\text{low}},\ M : \epsilon_1 I \prec W_k \prec \epsilon_2 I,\ f(x) \geq f_{\text{low}},\ \|\nabla^2 f(x)\| \leq M$$

Have:

$$\begin{aligned} \|d_k\| &\leq \epsilon_1^{-1}\|\nabla f(x_k)\| && (\xi^{\alpha_k} \in [x_k, x_k + d_k]) \\ f(x_k + \alpha d_k) - f(x_k) &= \alpha \nabla f(x_k)^T d_k + \alpha^2 {d_k}^T \nabla^2 f(\xi^{\alpha_k}) d_k \\ &\leq \alpha \nabla f(x_k)^T d_k + \alpha^2 M \|d_k\|^2 \\ \nabla f(x_k)^T d_k &= -\nabla f(x_k)^T W_k^{-1} \nabla f(x_k) \leq -\epsilon_2^{-1}\|\nabla f(x_k)\|^2 \end{aligned}$$

Therefore:

$$\begin{aligned} & f(x_k + \alpha d_k) - f(x_k) - \eta\alpha \nabla f(x_k)^T d_k \\ \leq\ & \alpha(1-\eta)\nabla f(x_k)^T d_k + \alpha^2 M \epsilon_1^{-2}\|\nabla f(x_k)\|^2 \\ \leq\ & -\alpha(1-\eta)\epsilon_2^{-1}\|\nabla f(x_k)\|^2 + \alpha^2 M \epsilon_1^{-2}\|\nabla f(x_k)\|^2 \end{aligned}$$

Global Convergence Proof (1)

Method: $\boxed{d_k = -W_k^{-1}\nabla f(x_k) \qquad f(x_{k+1}) - f(x_k) \leq \eta\alpha_k \nabla f(x_k)^T d_k}$

Assume:

$$\exists \epsilon_1, \epsilon_2 > 0,\ f_{\text{low}},\ M : \epsilon_1 I \prec W_k \prec \epsilon_2 I,\ f(x) \geq f_{\text{low}},\ \|\nabla^2 f(x)\| \leq M$$

Have:

$$\begin{aligned} \|d_k\| &\leq \epsilon_1^{-1}\|\nabla f(x_k)\| && (\xi^{\alpha_k} \in [x_k, x_k + d_k]) \\ f(x_k + \alpha d_k) - f(x_k) &= \alpha \nabla f(x_k)^T d_k + \alpha^2 {d_k}^T \nabla^2 f(\xi^{\alpha_k}) d_k \\ &\leq \alpha \nabla f(x_k)^T d_k + \alpha^2 M \|d_k\|^2 \\ \nabla f(x_k)^T d_k &= -\nabla f(x_k)^T W_k^{-1} \nabla f(x_k) \leq -\epsilon_2^{-1}\|\nabla f(x_k)\|^2 \end{aligned}$$

Therefore:

$$\begin{aligned} & f(x_k + \alpha d_k) - f(x_k) - \eta\alpha \nabla f(x_k)^T d_k \\ \leq\ & \alpha(1-\eta)\nabla f(x_k)^T d_k + \alpha^2 M \epsilon_1^{-2}\|\nabla f(x_k)\|^2 \\ \leq\ & -\alpha(1-\eta)\epsilon_2^{-1}\|\nabla f(x_k)\|^2 + \alpha^2 M \epsilon_1^{-2}\|\nabla f(x_k)\|^2 < 0 \end{aligned}$$

for $\alpha < \frac{\epsilon_1(1-\eta)}{\epsilon_2 M} =: \bar{\alpha}$. Hence $\alpha_k \geq \frac{1}{2}\bar{\alpha} > 0$ from backtracking line search.

Global Convergence Proof (2)

Method: $\boxed{d_k = -W_k^{-1}\nabla f(x_k) \qquad f(x_{k+1}) - f(x_k) \leq \eta\alpha_k \nabla f(x_k)^T d_k}$

Have: $\alpha_k \geq \frac{1}{2}\bar{\alpha} > 0$ and $\nabla f(x_k)^T d_k \leq -\epsilon_2^{-1}\|\nabla f(x_k)\|^2$.

Global Convergence Proof (2)

Method: $\boxed{d_k = -W_k^{-1}\nabla f(x_k) \qquad f(x_{k+1}) - f(x_k) \leq \eta\alpha_k \nabla f(x_k)^T d_k}$

Have: $\alpha_k \geq \frac{1}{2}\bar{\alpha} > 0$ and $\nabla f(x_k)^T d_k \leq -\epsilon_2^{-1}\|\nabla f(x_k)\|^2$.

Therefore:

$$\begin{aligned} f(x_{k+1}) - f(x_k) &\leq \eta\alpha_k \nabla f(x_k)^T d_k \\ &\leq -\eta\alpha_k \epsilon_2^{-1}\|\nabla f(x_k)\|^2 \\ &\leq -\frac{1}{2}\eta\bar{\alpha}\epsilon_2^{-1}\|\nabla f(x_k)\|^2 \end{aligned}$$

Global Convergence Proof (2)

Method: $\boxed{d_k = -W_k^{-1}\nabla f(x_k) \qquad f(x_{k+1}) - f(x_k) \leq \eta\alpha_k \nabla f(x_k)^T d_k}$

Have: $\alpha_k \geq \frac{1}{2}\bar{\alpha} > 0$ and $\nabla f(x_k)^T d_k \leq -\epsilon_2^{-1}\|\nabla f(x_k)\|^2$.

Therefore:

$$\begin{aligned} f(x_{k+1}) - f(x_k) &\leq \eta\alpha_k \nabla f(x_k)^T d_k \\ &\leq -\eta\alpha_k \epsilon_2^{-1}\|\nabla f(x_k)\|^2 \\ &\leq -\frac{1}{2}\eta\bar{\alpha}\epsilon_2^{-1}\|\nabla f(x_k)\|^2 \end{aligned}$$

Since $f(x_k) \geq f_{\text{low}}$, we have global convergence result:

$$\boxed{\lim_{k\to\infty} \|\nabla f(x_k)\| = 0}$$

Local Convergence Of Globalized Method

Assume:

- x_* satisfies sufficient second order optimality conditions.
- x_k gets close to x_*.
- No Hessian modification takes place for large k.
  (Remember: $\nabla^2 f(x_*) \succ 0$)

Local Convergence Of Globalized Method

Assume:

- x_* satisfies sufficient second order optimality conditions.
- x_k gets close to x_*.
- No Hessian modification takes place for large k.
  (Remember: $\nabla^2 f(x_*) \succ 0$)

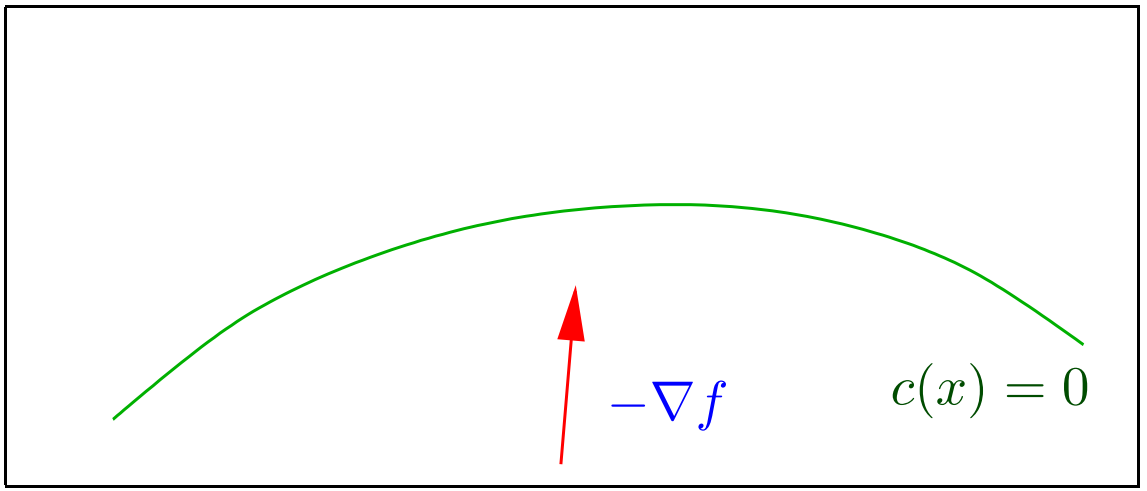
Then, using Taylor's theorem, one can show:

- $\alpha_k = 1$ for large k.
  (i.e., eventually, we take full Newton steps)
- $\{x_k\}$ converges quadratically to x_*.

Equality-Constrained Optimization

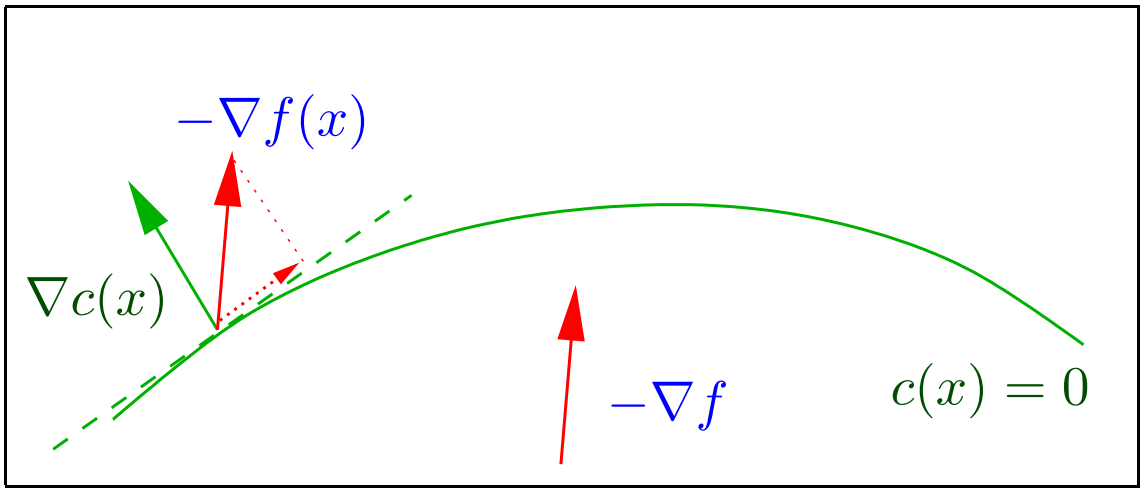
First Order Optimality Conditions Motivation

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ s.t. \quad & c(x) = 0 \end{aligned}$$



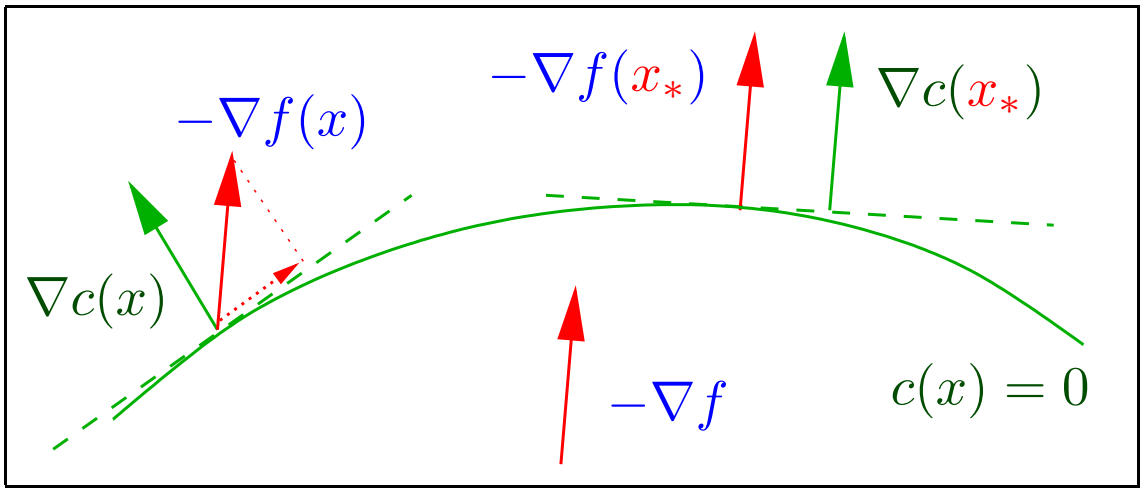
First Order Optimality Conditions Motivation

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ s.t. \quad & c(x) = 0 \end{aligned}$$



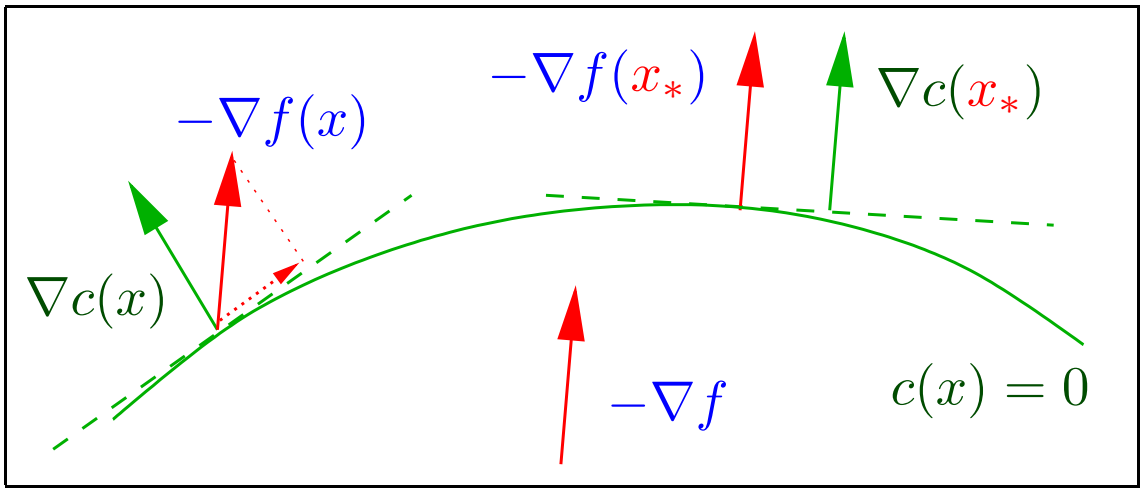
First Order Optimality Conditions Motivation

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ s.t. \quad & c(x) = 0 \end{aligned}$$



First Order Optimality Conditions Motivation

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ s.t. \quad & c(x) = 0 \end{aligned}$$

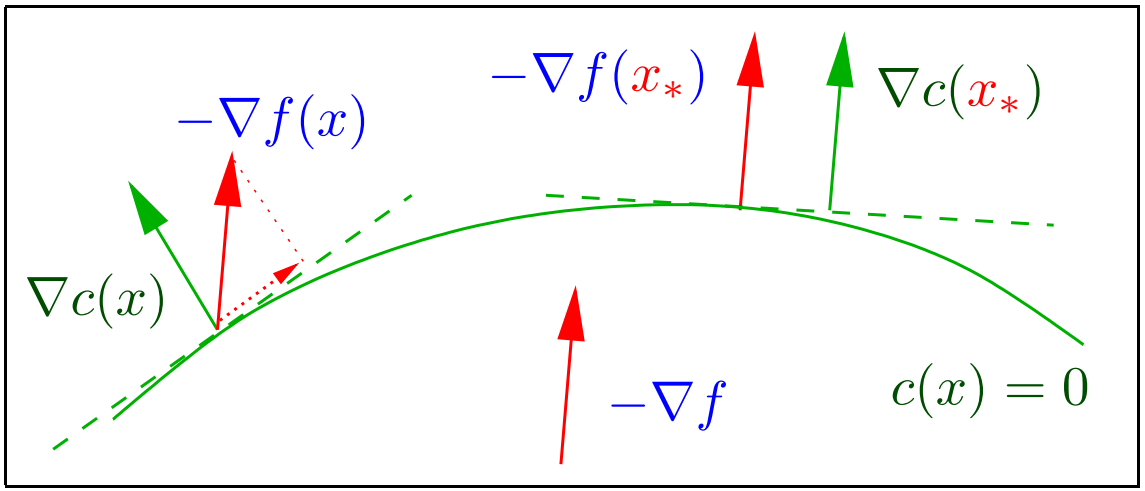


First order optimality conditions:

$$\begin{aligned} \nabla f(x_*) + \nabla c(x_*)\lambda_* &= 0 \\ c(x_*) &= 0 \end{aligned}$$

First Order Optimality Conditions Motivation

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ s.t. \quad & c(x) = 0 \end{aligned}$$



First order optimality conditions:

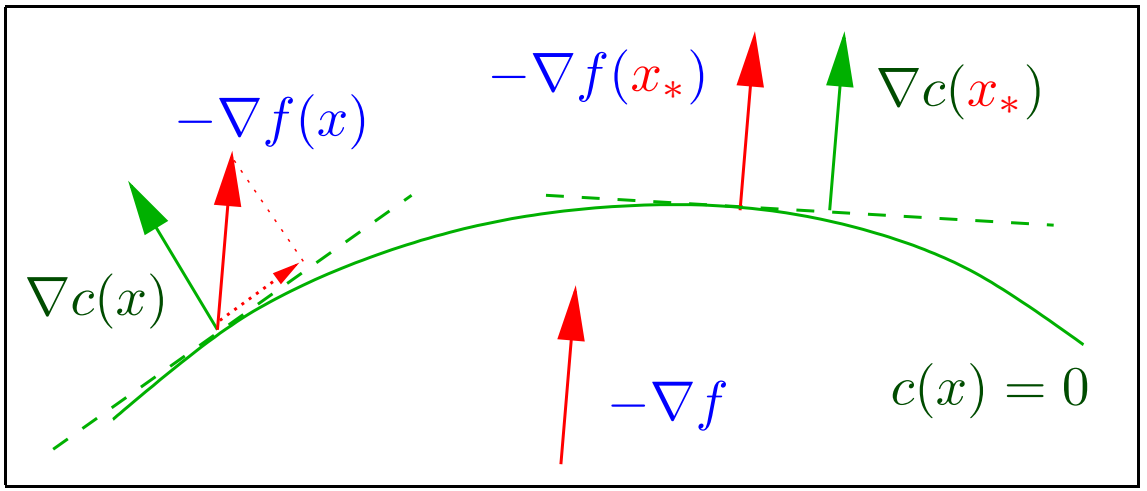
$$\begin{aligned} \nabla f(x_*) + \nabla c(x_*)\lambda_* &= 0 \\ c(x_*) &= 0 \end{aligned}$$

Constraint qualification:

$\nabla c_1(x_*), \ldots, \nabla c_m(x_*)$ linear ind.

First Order Optimality Conditions Motivation

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ s.t. \quad & c(x) = 0 \end{aligned}$$



First order optimality conditions:

$$\begin{aligned} \nabla f(x_*) + \nabla c(x_*)\lambda_* &= 0 \\ c(x_*) &= 0 \end{aligned}$$

Constraint qualification:

$\nabla c_1(x_*), \ldots, \nabla c_m(x_*)$ linear ind.

Lagrangian function:

$$\mathcal{L}(x, \lambda) = f(x) + c(x)^T \lambda$$

λ_* are called optimal (Lagrangian) multipliers.

First Order Optimality Conditions (1)

- Assume:
  - x_* is a local solution, i.e., there exists neighborhood $\mathcal{N}_*$ of x_* with

    $$f(x) \geq f(x_*) \qquad \text{for all } x \in \mathcal{N}_* \text{ with } c(x) = 0$$

  - $\nabla c(x_*)$ has full rank

First Order Optimality Conditions (1)

- Assume:
  - x_* is a local solution, i.e., there exists neighborhood $\mathcal{N}_*$ of x_* with

    $$f(x) \geq f(x_*) \qquad \text{for all } x \in \mathcal{N}_* \text{ with } c(x) = 0$$

  - $\nabla c(x_*)$ has full rank
- Then: There exist multipliers λ_* such that

$$\begin{array}{rcl} \nabla f(x_*) + \nabla c(x_*)\lambda_* & = & 0 \\ c(x_*) & = & 0 \end{array}$$

First Order Optimality Conditions (1)

- Assume:
  - x_* is a local solution, i.e., there exists neighborhood $\mathcal{N}_*$ of x_* with

$$f(x) \geq f(x_*) \qquad \text{for all } x \in \mathcal{N}_* \text{ with } c(x) = 0$$

  - $\nabla c(x_*)$ has full rank
- Then: There exist multipliers λ_* such that

$$\begin{array}{rcl} \nabla f(x_*) + \nabla c(x_*)\lambda_* & = & 0 \\ c(x_*) & = & 0 \end{array}$$

- Condition "$\nabla c(x_*)$ has full rank" is called a "constraint qualification"

First Order Optimality Conditions (1)

- Assume:
  - x_* is a local solution, i.e., there exists neighborhood $\mathcal{N}_*$ of x_* with

$$f(x) \geq f(x_*) \qquad \text{for all } x \in \mathcal{N}_* \text{ with } c(x) = 0$$

  - $\nabla c(x_*)$ has full rank
- Then: There exist multipliers λ_* such that

$$\begin{aligned} \nabla f(x_*) + \nabla c(x_*)\lambda_* &= 0 \\ c(x_*) &= 0 \end{aligned}$$

- Condition "$\nabla c(x_*)$ has full rank" is called a "constraint qualification"

$$\min_{x \in \mathbb{R}^n} (x^{(1)} + 1)^2 + (x^{(2)})^2$$

$$s.t. (x^{(1)})^2 = 0$$

First Order Optimality Conditions (1)

- Assume:
  - x_* is a local solution, i.e., there exists neighborhood $\mathcal{N}_*$ of x_* with

$$f(x) \geq f(x_*) \qquad \text{for all } x \in \mathcal{N}_* \text{ with } c(x) = 0$$

  - $\nabla c(x_*)$ has full rank
- Then: There exist multipliers λ_* such that

$$\begin{aligned} \nabla f(x_*) + \nabla c(x_*)\lambda_* &= 0 \\ c(x_*) &= 0 \end{aligned}$$

- Condition "$\nabla c(x_*)$ has full rank" is called a "constraint qualification"

$$\min_{x \in \mathbb{R}^n} (x^{(1)} + 1)^2 + (x^{(2)})^2 \quad s.t. (x^{(1)})^2 = 0$$

has $x_* = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, but $\nabla f(x_*) = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$, $\nabla c(x_*) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$

First Order Optimality Conditions (2)

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ s.t. \quad & c(x) = 0 \end{aligned}$$

x	Variables
$f(x) : \mathbb{R}^n \longrightarrow \mathbb{R}$	Objective function
$c(x) : \mathbb{R}^n \longrightarrow \mathbb{R}^m$	Equality constraints

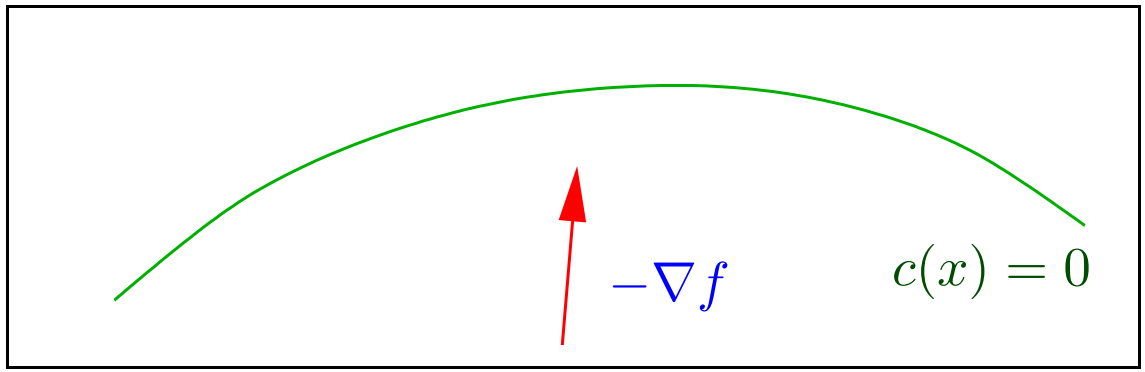
From implicit function theorem:

- Can partition $x = (y, z)$ with
  - $y \in \mathbb{R}^{n-m}$ (independent variables)
  - $z \in \mathbb{R}^m$ (dependent variables)
- Find C^2 function $\Phi(y) : \mathbb{R}^{n-m} \longrightarrow \mathbb{R}^m$, so that

$$\Phi(y_*) = z_* \quad \text{and} \quad c(y, \Phi(y)) = 0 \text{ for all } (y, \Phi(y)) \in \mathcal{N}_*$$

i.e., z is determined by y if variables should satisfy $c(x) = 0$

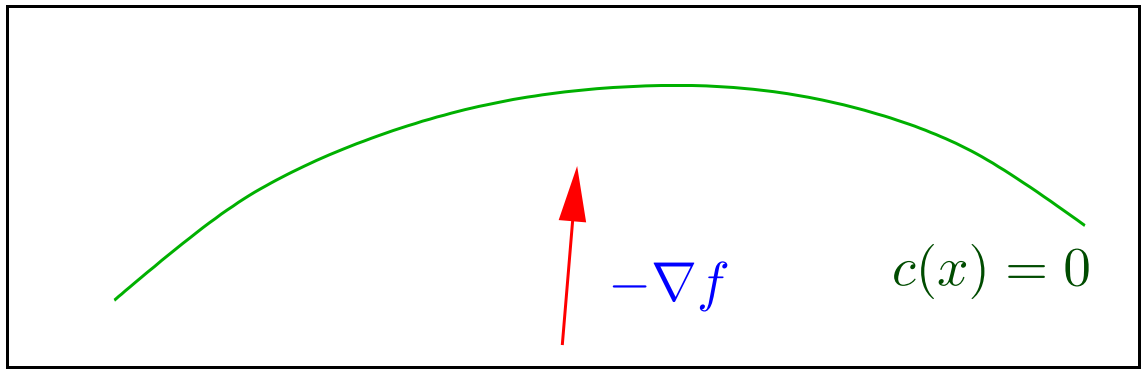
First Order Optimality Conditions (3)



$\Phi(y_*) = z_*$ and $c(y, \Phi(y)) = 0$ for all $(y, \Phi(y)) \in \mathcal{N}_*$

- Φ is a "local parametrization" of the manifold $\mathcal{M} = \{x \in \mathcal{N}_* : c(x) = 0\}$.

First Order Optimality Conditions (3)



$\Phi(y_*) = z_*$ and $c(y, \Phi(y)) = 0$ for all $(y, \Phi(y)) \in \mathcal{N}_*$

- Φ is a "local parametrization" of the manifold $\mathcal{M} = \{x \in \mathcal{N}_* : c(x) = 0\}$.
- Then: x_* is local minimizer of

$$\min f(x), \text{ s.t. } c(x) = 0$$

is equivalent to: $x_* = (y_*, \Phi(y_*))$, where y_* is local minimizer of unconstrained problem

$$\min \Psi(y) := f(y, \Phi(y))$$

First Order Optimality Conditions (4)

Optimality of y_* for

$$\boxed{\min \Psi(y) := f(y, \Phi(y))}$$

implies

$$\begin{aligned} 0 &= \nabla_y \Psi(y_*) = \nabla_y [f(y_*, \Phi(y_*))] \\ &= \nabla_y f(y_*, z_*) + \nabla_y \Phi(y_*) \nabla_z f(y_*, z_*) \end{aligned}$$

First Order Optimality Conditions (4)

Optimality of y_* for

$$\boxed{\min \Psi(y) := f(y, \Phi(y))}$$

implies

$$\begin{aligned} 0 &= \nabla_y \Psi(y_*) = \nabla_y [f(y_*, \Phi(y_*))] \\ &= \nabla_y f(y_*, z_*) + \nabla_y \Phi(y_*) \nabla_z f(y_*, z_*) \\ &= \nabla_y f(x_*) - \nabla_y c(x_*) \left[\nabla_z c(x_*)\right]^{-1} \nabla_z f(x_*) \end{aligned}$$

since from $c(y, \Phi(y)) = 0$

$$0 = \nabla_y [c(y_*, \Phi(y_*))] = \nabla_y c(y_*, z_*) + \nabla_z c(y_*, z_*) \nabla_y \Phi(y_*)$$

First Order Optimality Conditions (4)

Optimality of y_* for

$$\boxed{\min \Psi(y) := f(y, \Phi(y))}$$

implies

$$\begin{aligned} 0 &= \nabla_y \Psi(y_*) = \nabla_y [f(y_*, \Phi(y_*))] \\ &= \nabla_y f(y_*, z_*) + \nabla_y \Phi(y_*) \nabla_z f(y_*, z_*) \\ &= \nabla_y f(x_*) - \nabla_y c(x_*) \left[\nabla_z c(x_*)\right]^{-1} \nabla_z f(x_*) \end{aligned}$$

since from $c(y, \Phi(y)) = 0$

$$0 = \nabla_y [c(y_*, \Phi(y_*))] = \nabla_y c(y_*, z_*) + \nabla_z c(y_*, z_*) \nabla_y \Phi(y_*)$$

Define $\lambda_* := -\left[\nabla_z c(x_*)\right]^{-1} \nabla_z f(x_*)$, then

$$\begin{aligned} 0 &= \nabla_y f(x_*) + \nabla_y c(x_*) \lambda_* \\ 0 &= \nabla_z f(x_*) + \nabla_z c(x_*) \lambda_* \end{aligned}$$

Second Order Optimality Conditions

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ s.t. \quad & c(x) = 0 \end{aligned}$$

x	Variables
$f(x) : \mathbb{R}^n \longrightarrow \mathbb{R}$	Objective function
$c(x) : \mathbb{R}^n \longrightarrow \mathbb{R}^m$	Equality constraints

Lagrangian $\mathcal{L}(x, \lambda) = f(x) + \lambda^T c(x)$; Assumptions as before.

- 2nd order necessary conditions: If x_* is local minimum, then

  for all $d \in \mathbb{R}^n$ with $\nabla c(x_*)^T d = 0$: $\quad d^T \nabla^2_{xx} \mathcal{L}(x_*, \lambda_*)^T d \geq 0$

"Projection of $\nabla^2_{xx} \mathcal{L}(x_*, \lambda_*)$ onto manifold $\mathcal{M}$ is positive semi- definite"

Second Order Optimality Conditions

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ s.t. \quad & c(x) = 0 \end{aligned}$$

x	Variables
$f(x) : \mathbb{R}^n \longrightarrow \mathbb{R}$	Objective function
$c(x) : \mathbb{R}^n \longrightarrow \mathbb{R}^m$	Equality constraints

Lagrangian $\mathcal{L}(x, \lambda) = f(x) + \lambda^T c(x)$; Assumptions as before.

- 2nd order necessary conditions: If x_* is local minimum, then

  for all $d \in \mathbb{R}^n$ with $\nabla c(x_*)^T d = 0$: $\quad d^T \nabla^2_{xx} \mathcal{L}(x_*, \lambda_*)^T d \geq 0$

- 2nd order sufficient conditions:

  If x_* satisfies 1st order necessary conditions and
  for all $d \in \mathbb{R}^n \setminus \{0\}$ with $\nabla c(x_*)^T d = 0$: $\quad d^T \nabla^2_{xx} \mathcal{L}(x_*, \lambda_*)^T d > 0,$

  then x_* is local a solution.

"Projection of $\nabla^2_{xx} \mathcal{L}(x_*, \lambda_*)$ onto manifold $\mathcal{M}$ is positive (semi-)definite"

Newton's Method (again)

- Idea of many algorithms: Apply Newton's method to

$$\begin{aligned} \nabla f(x_*) + \nabla c(x_*)\lambda_* &= 0 \\ c(x_*) &= 0 \end{aligned}$$

- Nonlinear system of equations in (x, λ).
- Fast local convergence for 2nd order sufficient (x_*, λ_*)

Newton's Method (again)

- Idea of many algorithms: Apply Newton's method to

$$\begin{aligned} \nabla f(x_*) + \nabla c(x_*)\lambda_* &= 0 \\ c(x_*) &= 0 \end{aligned}$$

- Nonlinear system of equations in (x, λ).
- Fast local convergence for 2nd order sufficient (x_*, λ_*)
- Newton steps: $(W_k = \nabla^2_{xx}\mathcal{L}(x_k, \lambda_k))$

$$\begin{bmatrix} W_k & \nabla c(x_k) \\ \nabla c(x_k)^T & 0 \end{bmatrix} \begin{pmatrix} d_k^x \\ d_k^\lambda \end{pmatrix} = -\begin{pmatrix} \nabla f(x_k) + \nabla c(x_k)\lambda_k \\ c(x_k) \end{pmatrix}$$

Newton's Method (again)

- Idea of many algorithms: Apply Newton's method to

$$\begin{aligned} \nabla f(x_*) + \nabla c(x_*)\lambda_* &= 0 \\ c(x_*) &= 0 \end{aligned}$$

- Nonlinear system of equations in (x, λ).
- Fast local convergence for 2nd order sufficient (x_*, λ_*)
- Newton steps: $(W_k = \nabla^2_{xx}\mathcal{L}(x_k, \lambda_k))$

$$\begin{bmatrix} W_k & \nabla c(x_k) \\ \nabla c(x_k)^T & 0 \end{bmatrix} \begin{pmatrix} d_k^x \\ d_k^\lambda \end{pmatrix} = -\begin{pmatrix} \nabla f(x_k) + \nabla c(x_k)\lambda_k \\ c(x_k) \end{pmatrix}$$

- "Infeasible" algorithm, since $c(x_k) = 0$ can be violated

Interpretation As Quadratic Program

- Newton step:

$$\begin{bmatrix} W_k & \nabla c(x_k) \\ \nabla c(x_k)^T & 0 \end{bmatrix} \begin{pmatrix} d_k^x \\ d_k^\lambda \end{pmatrix} = - \begin{pmatrix} \nabla f(x_k) + \nabla c(x_k)\lambda_k \\ c(x_k) \end{pmatrix}$$

Interpretation As Quadratic Program

- Newton step:

$$\begin{bmatrix} W_k & \nabla c(x_k) \\ \nabla c(x_k)^T & 0 \end{bmatrix} \begin{pmatrix} d_k^x \\ \lambda_k^+ \end{pmatrix} = - \begin{pmatrix} \nabla f(x_k) \\ c(x_k) \end{pmatrix} \qquad \lambda_k^+ = \lambda_k + d_k^\lambda$$

Interpretation As Quadratic Program

- Newton step:

$$\begin{bmatrix} W_k & \nabla c(x_k) \\ \nabla c(x_k)^T & 0 \end{bmatrix} \begin{pmatrix} d_k^x \\ \lambda_k^+ \end{pmatrix} = - \begin{pmatrix} \nabla f(x_k) \\ c(x_k) \end{pmatrix} \qquad \lambda_k^+ = \lambda_k + d_k^\lambda$$

- These are the optimality conditions for quadratic program (QP)

$$\begin{aligned} \min_{d \in \mathbb{R}^n} \quad & f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T W_k d \\ s.t. \quad & c(x_k) + \nabla c(x_k)^T d = 0 \end{aligned}$$

- Therefore: We can alternatively solve QP to obtain (d_k^x, λ_k^+)
- Can interpret Newton step as solution of “local model” of NLP

Interpretation As Quadratic Program

- Newton step:

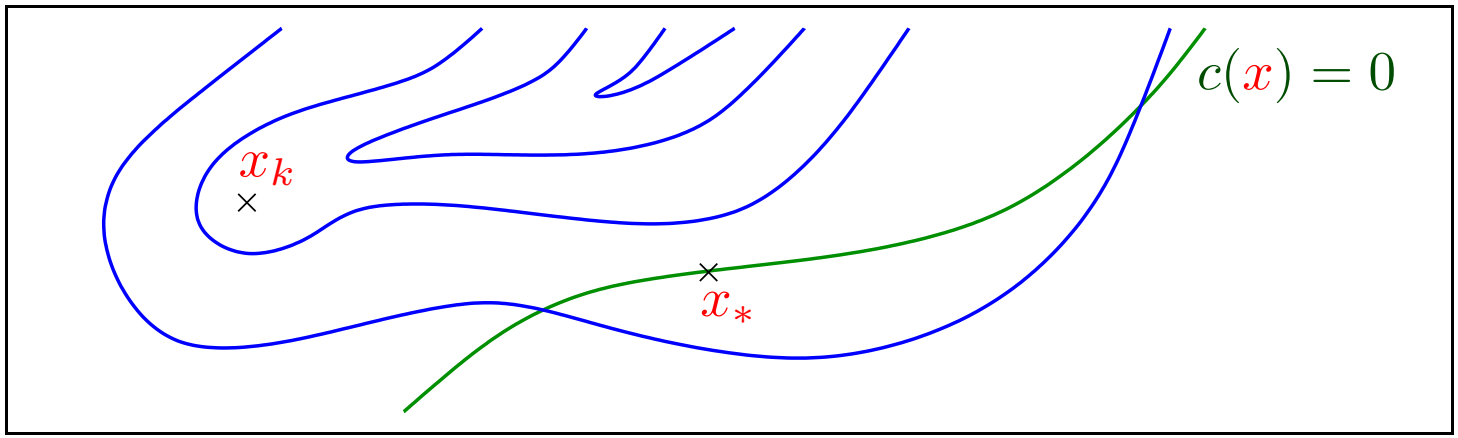
$$\begin{bmatrix} W_k & \nabla c(x_k) \\ \nabla c(x_k)^T & 0 \end{bmatrix} \begin{pmatrix} d_k^x \\ \lambda_k^+ \end{pmatrix} = - \begin{pmatrix} \nabla f(x_k) \\ c(x_k) \end{pmatrix} \qquad \lambda_k^+ = \lambda_k + d_k^\lambda$$

- These are the optimality conditions for quadratic program (QP)

$$\begin{aligned} \min_{d \in \mathbb{R}^n} \quad & f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T W_k d \\ s.t. \quad & c(x_k) + \nabla c(x_k)^T d = 0 \end{aligned}$$

- Therefore: We can alternatively solve QP to obtain (d_k^x, λ_k^+)
- Can interpret Newton step as solution of "local model" of NLP
- Implicitly assumed projection of W_k onto null space of $\nabla c(x_k)^T$ is positive definite; otherwise QP is unbounded.

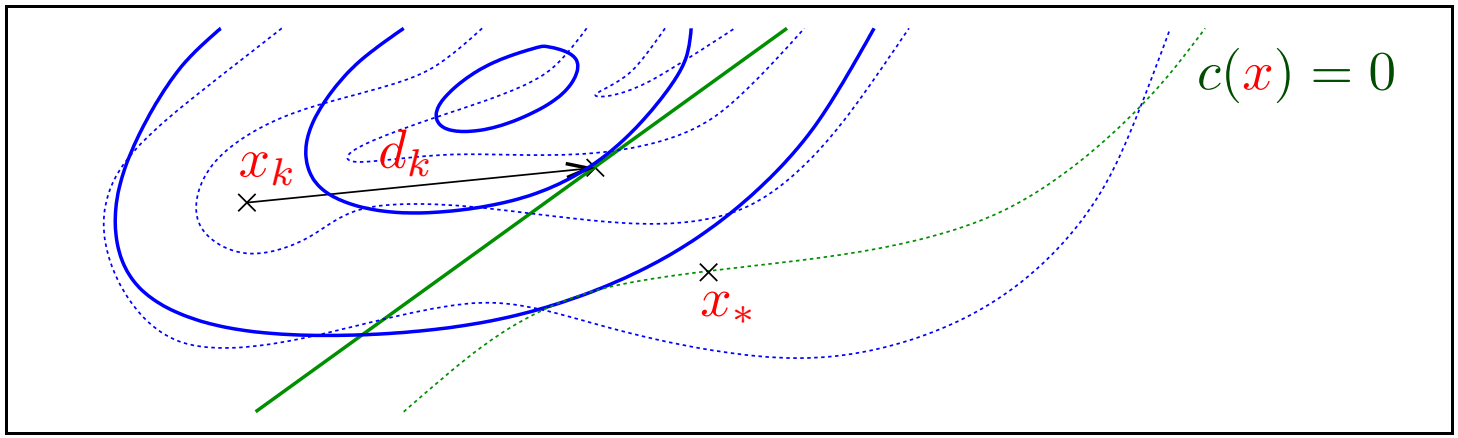
Sequential Quadratic Programming (SQP)



Original Problem

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ s.t. \quad & c(x) = 0 \end{aligned}$$

Sequential Quadratic Programming (SQP)



Quadratic Model

$$\begin{aligned} \min_{d \in \mathbb{R}^n} \quad & f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T W_k d \\ s.t. \quad & c(x_k) + \nabla c(x_k)^T d = 0 \end{aligned}$$

- At each iteration, compute d_k as solution of QP, and $\lambda_{k+1} = \lambda_k^+$
- Nonconvexity: Ensure that W_k pos. def. in null space of $\nabla c(x_k)^T$
- If $W_k = \nabla^2_{xx} \mathcal{L}(x_k, \lambda_k)$, this is equivalent to Newton's method (fast)

Handling Nonconvexities in SQP

- Want: Projection $(P_k^c)^T W_k P_k^c \succ 0$ (with projection $(P_k^c)^T \nabla c(x_k) = 0$)
- Could compute eigenvalues of $(P_k^c)^T W_k P_k^c$ explicitly (expensive!)

Handling Nonconvexities in SQP

- Want: Projection $(P_k^c)^T W_k P_k^c \succ 0$ (with projection $(P_k^c)^T \nabla c(x_k) = 0$)
- Could compute eigenvalues of $(P_k^c)^T W_k P_k^c$ explicitly (expensive!)
- Can show: "$(P_k^c)^T W_k P_k^c \succ 0$" is equivalent to:

$$\begin{matrix} n \\ m \end{matrix} \begin{bmatrix} W_k & \nabla c(x_k) \\ \nabla c(x_k)^T & 0 \end{bmatrix}$$ has n positive, and m negative eigenvalues ("has inertia $(n, m, 0)$")

- Many direct symmetric indefinite linear solvers can easily return inertia
- One way: Perform trial factorizations of $\begin{bmatrix} \nabla^2_{xx} \mathcal{L}(x_k, \lambda_k) + \delta I & \nabla c(x_k) \\ \nabla c(x_k)^T & 0 \end{bmatrix}$ for increasing values of $\delta \geq 0$, until inertia is correct

Handling Nonconvexities in SQP

- Want: Projection $(P_k^c)^T W_k P_k^c \succ 0$ (with projection $(P_k^c)^T \nabla c(x_k) = 0$)
- Could compute eigenvalues of $(P_k^c)^T W_k P_k^c$ explicitly (expensive!)
- Can show: "$(P_k^c)^T W_k P_k^c \succ 0$" is equivalent to:

$$\begin{matrix} n \\ m \end{matrix} \begin{bmatrix} W_k & \nabla c(x_k) \\ \nabla c(x_k)^T & 0 \end{bmatrix}$$ has n positive, and m negative eigenvalues ("has inertia $(n, m, 0)$")

- Many direct symmetric indefinite linear solvers can easily return inertia
- One way: Perform trial factorizations of $\begin{bmatrix} \nabla^2_{xx} \mathcal{L}(x_k, \lambda_k) + \delta I & \nabla c(x_k) \\ \nabla c(x_k)^T & 0 \end{bmatrix}$ for increasing values of $\delta \geq 0$, until inertia is correct
- Other option: "Inertia-controlling" factorization (usually expensive)

Line Search With Merit Function

- Want to ensure convergence to stationary point from any (x_0, λ_0).
- Two (usually conflicting) goals:

$$\boxed{\min f(x)} \qquad \boxed{\min \|c(x)\|}$$

Line Search With Merit Function

- Want to ensure convergence to stationary point from any (x_0, λ_0).
- Two (usually conflicting) goals:

$$\boxed{\min f(x)} \qquad \boxed{\min \|c(x)\|}$$

- Merit function: Combine the two "measures" into one single function
- Exact penality function: $\Phi_\rho(x) = f(x) + \rho\|c(x)\|$
- If (x_*, λ_*) local solution: $\rho > \|\lambda_*\|^{\text{dual}} \implies x_*$ is local minimizer of $\Phi_\rho(x)$
- Idea: Minimize $\Phi_\rho(x)$ (non-differentiable)

Line Search With Merit Function

- Want to ensure convergence to stationary point from any (x_0, λ_0).
- Two (usually conflicting) goals:

$$\boxed{\min f(x)} \qquad \boxed{\min \|c(x)\|}$$

- Merit function: Combine the two "measures" into one single function
- Exact penality function: $\Phi_\rho(x) = f(x) + \rho\|c(x)\|$
- If (x_*, λ_*) local solution: $\rho > \|\lambda_*\|^{\text{dual}} \implies x_*$ is local minimizer of $\Phi_\rho(x)$
- Idea: Minimize $\Phi_\rho(x)$ (non-differentiable)
- Basic algorithm (simplified):
  - Compute d_k (ensure correct inertia)
  - Update $\rho_k \geq \rho_{k-1}$, so that directional derivative $D\Phi_{\rho_k}(x_k; d_k) < 0$
  - Backtracking line search $\alpha_k^l = 2^{-l}$, until Armijo condition satisfied:

$$\Phi_{\rho_k}(x_k + \alpha_k d_k) - \Phi_{\rho_k}(x_k) \leq \eta\alpha_k D\Phi_{\rho_k}(x_k; d_k)$$

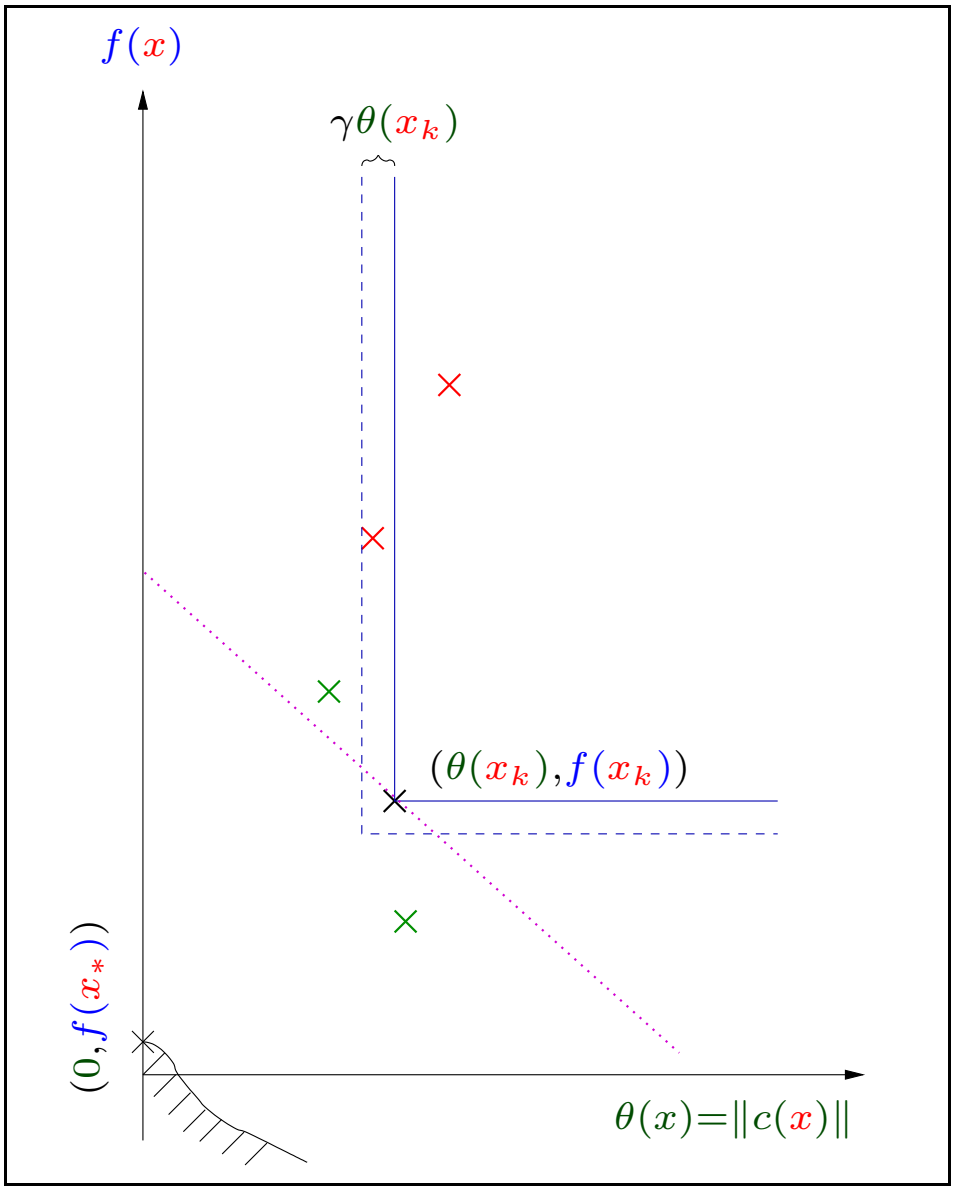
  - Set $x_{k+1} = x_k + \alpha_k d_k$ and $k \leftarrow k + 1$

A Filter Line Search Method

Filter: Alternative to merit function

(Fletcher, Leyffer; 1998)

A Filter Line Search Method



Filter: Alternative to merit function

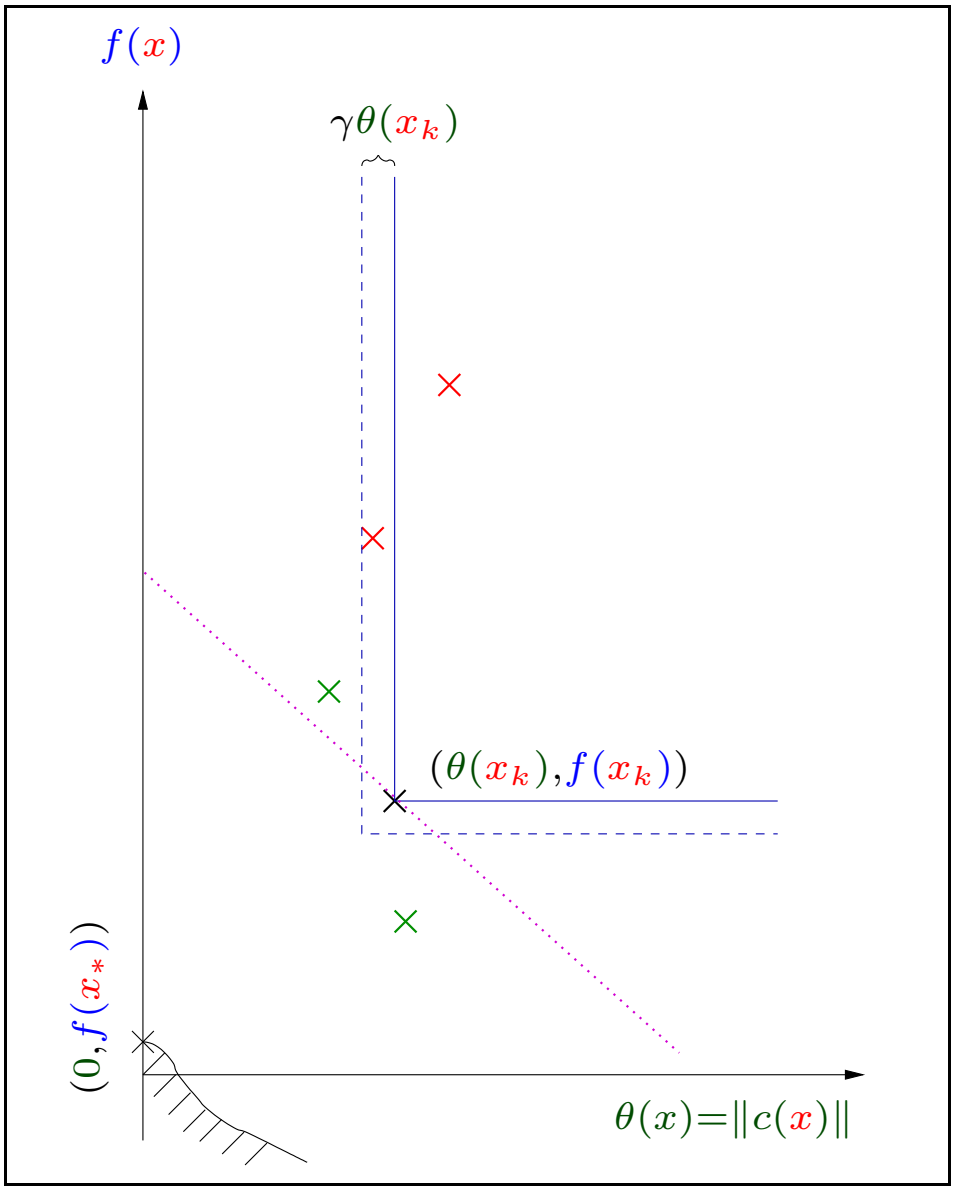
(Fletcher, Leyffer; 1998)

Idea: Bi-objective optimization

$$\begin{array}{ll} \min & f(x) \\ s.t. & c(x) = 0 \end{array}$$

$\min \ \theta(x)$ $\qquad$ $\min \ f(x)$

A Filter Line Search Method



Filter: Alternative to merit function

(Fletcher, Leyffer; 1998)

Idea: Bi-objective optimization

$$\begin{array}{ll} \min & f(x) \\ s.t. & c(x) = 0 \end{array}$$

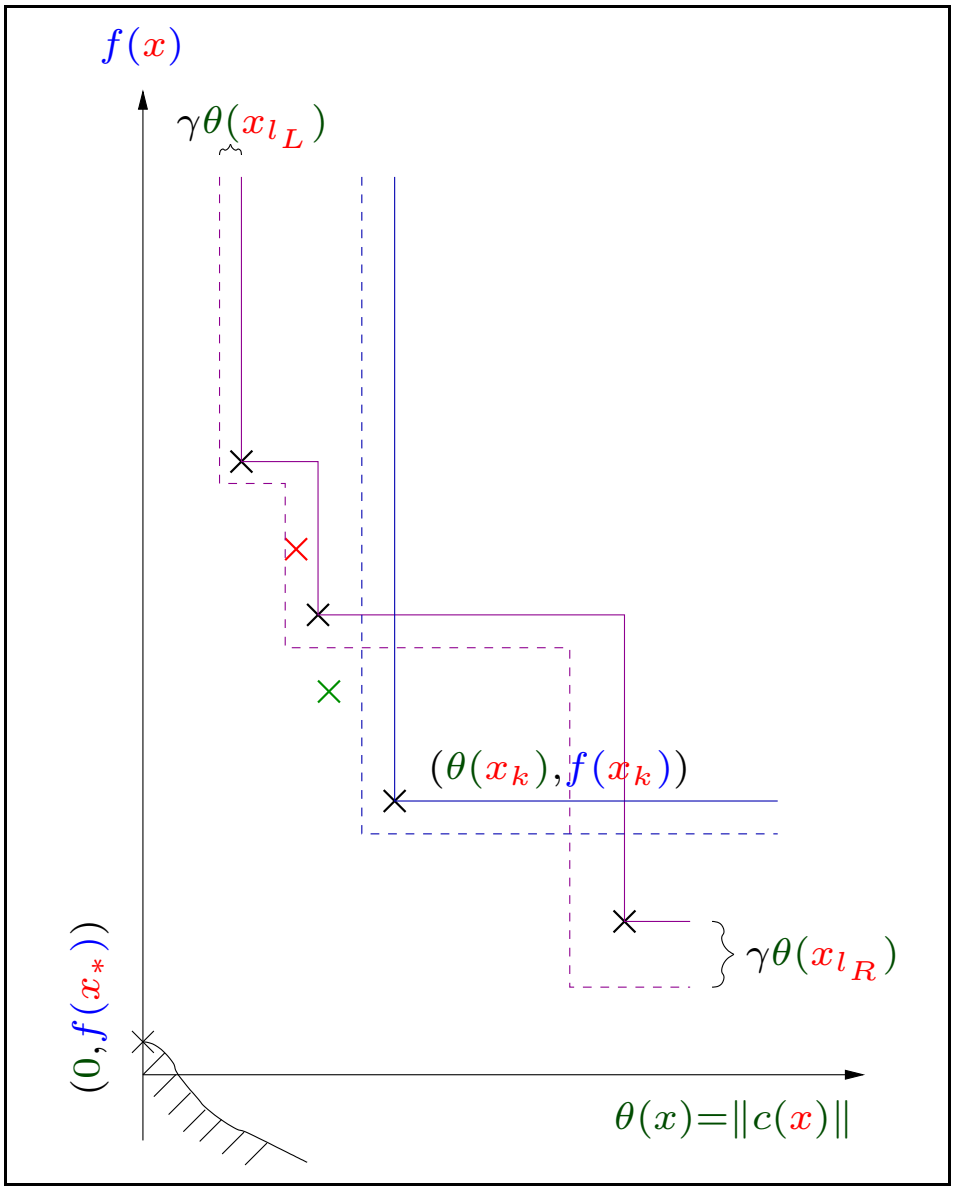
$$\min \quad \theta(x) \qquad\qquad \min \quad f(x)$$

$$x_{\text{tr}} = x_k + \alpha d_k^x$$

Sufficient progress w.r.t. x_k:

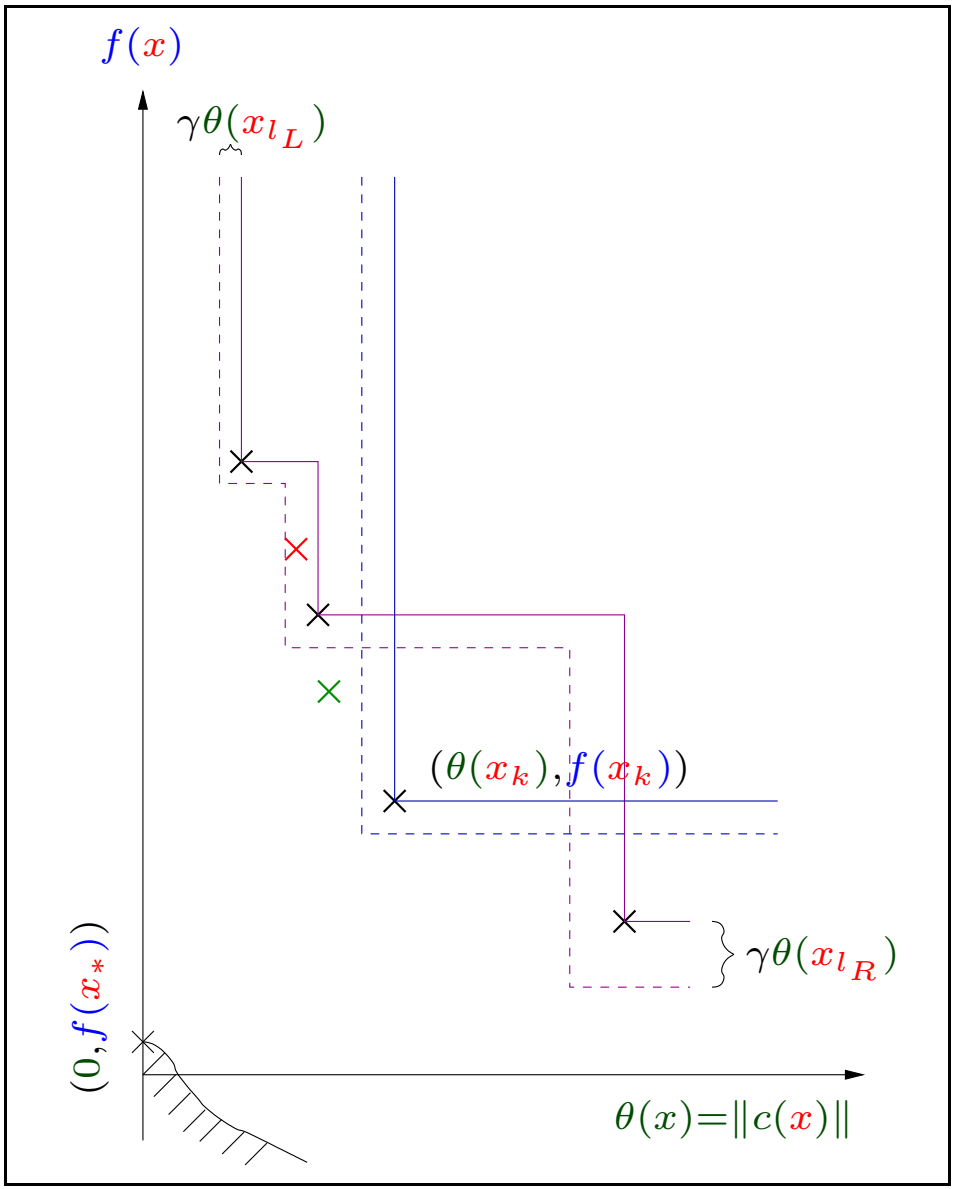
$$\begin{aligned} f(x_{\text{tr}}) &\leq f(x_k) - \gamma_f \theta(x_k) \\ \theta(x_{\text{tr}}) &\leq \theta(x_k) - \gamma_\theta \theta(x_k) \end{aligned}$$

A Filter Line Search Method (Filter)



Need to avoid cycling

A Filter Line Search Method (Filter)



Need to avoid cycling

$\Downarrow$

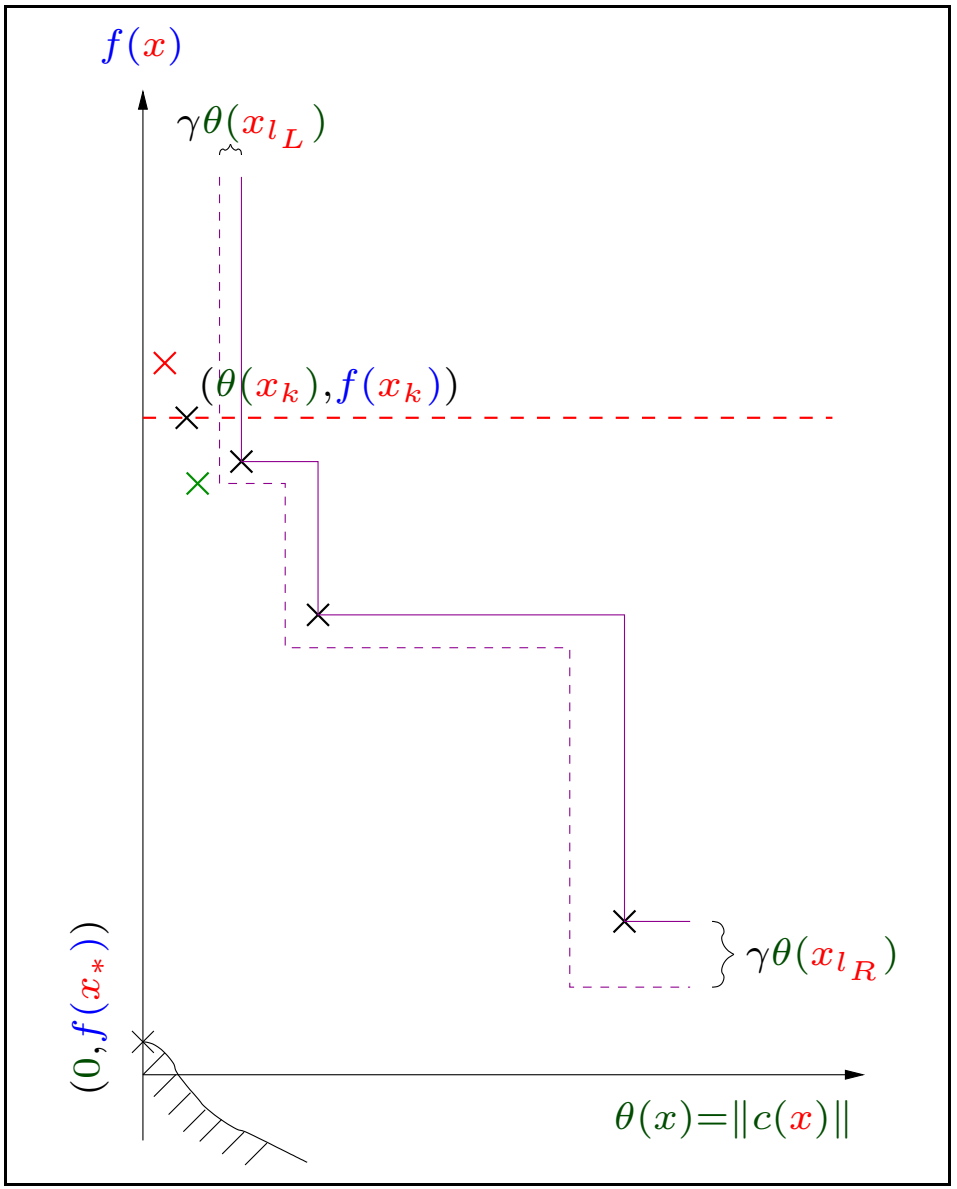
Store some previous $(\theta(x_l), f(x_l))$ pairs in *filter* $\mathcal{F}_k$

Sufficient progress w.r.t. filter:

$$\begin{aligned} f(x_{\text{tr}}) &\leq f(x_l) - \gamma_f \theta(x_l) \\ \theta(x_{\text{tr}}) &\leq \theta(x_l) - \gamma_\theta \theta(x_l) \end{aligned}$$

for $(\theta(x_l), f(x_l)) \in \mathcal{F}_k$

A Filter Line Search Method ("f-type")



If *switching condition*

$$-\alpha \nabla f(x_k)^T d_k^x > \delta \left[\theta(x_k)\right]^{s_\theta}$$

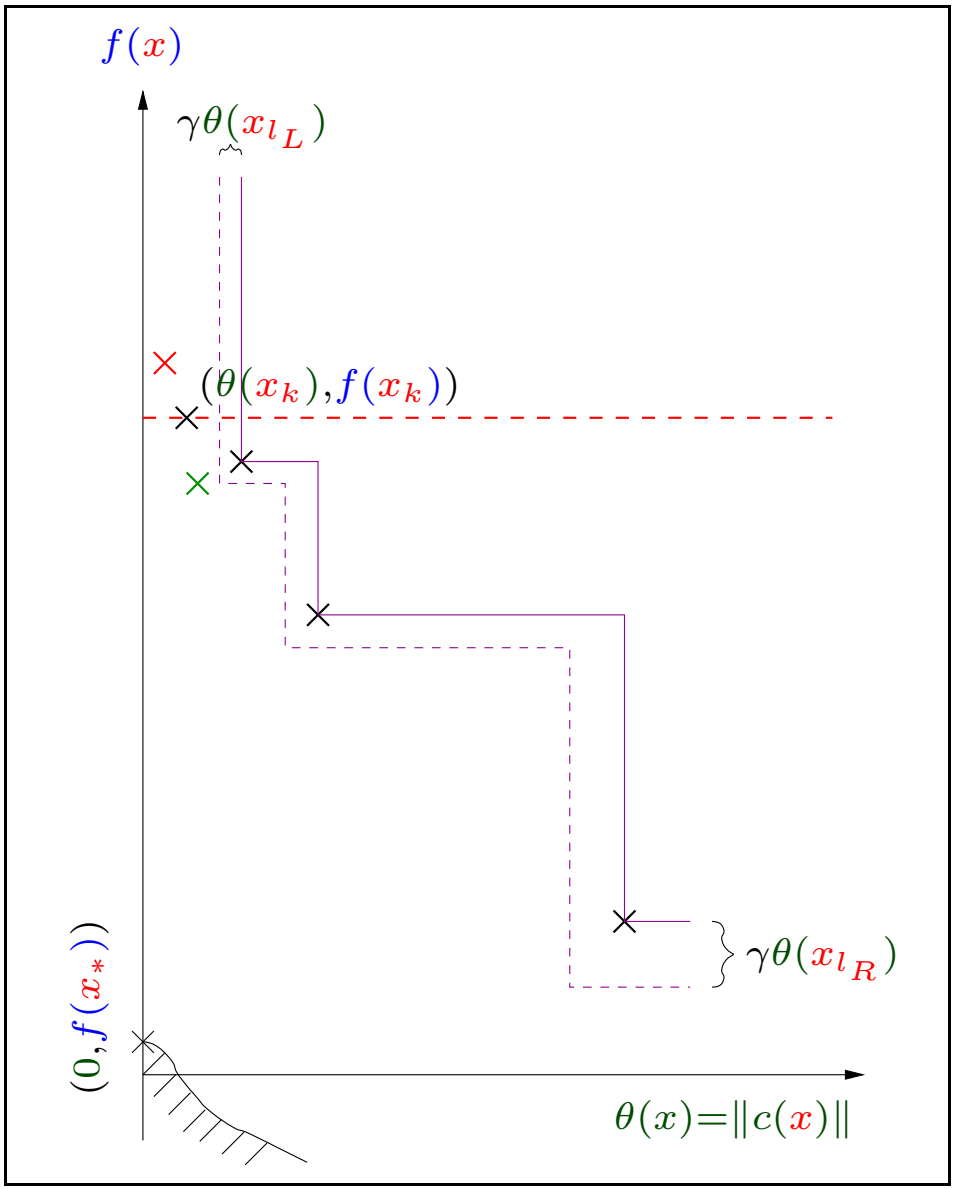
holds ($s_\theta > 1$):

$$\Downarrow$$

Armijo-condition on $f(x)$:

$$f(x_{\text{tr}}) \leq f(x_k) + \eta \alpha \nabla f(x_k)^T d_k^x$$

A Filter Line Search Method ("f-type")



If *switching condition*

$$-\alpha \nabla f(x_k)^T d_k^x > \delta \left[\theta(x_k)\right]^{s_\theta}$$

holds ($s_\theta > 1$):

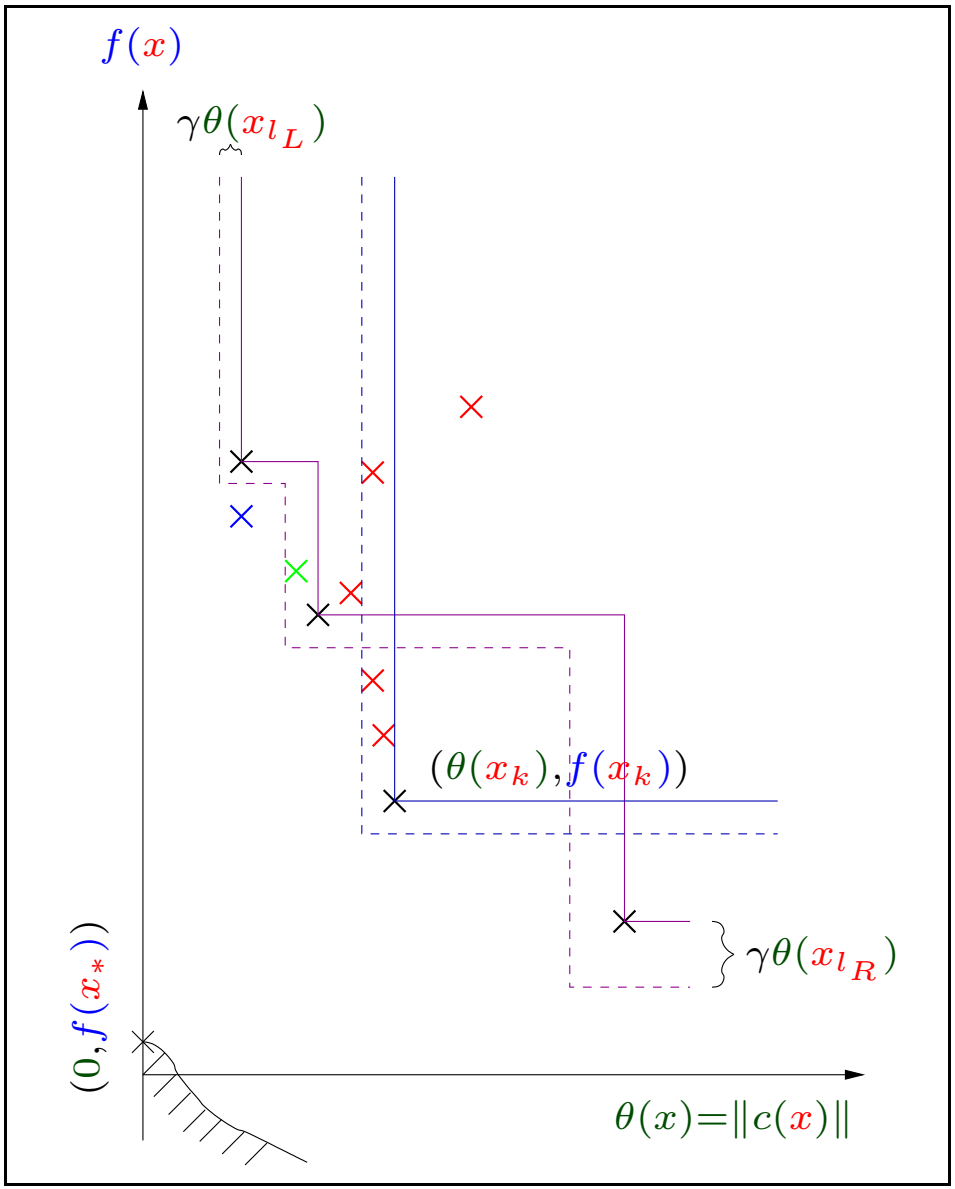
$$\Downarrow$$

Armijo-condition on $f(x)$:

$$f(x_{\text{tr}}) \leq f(x_k) + \eta \alpha \nabla f(x_k)^T d_k^x$$

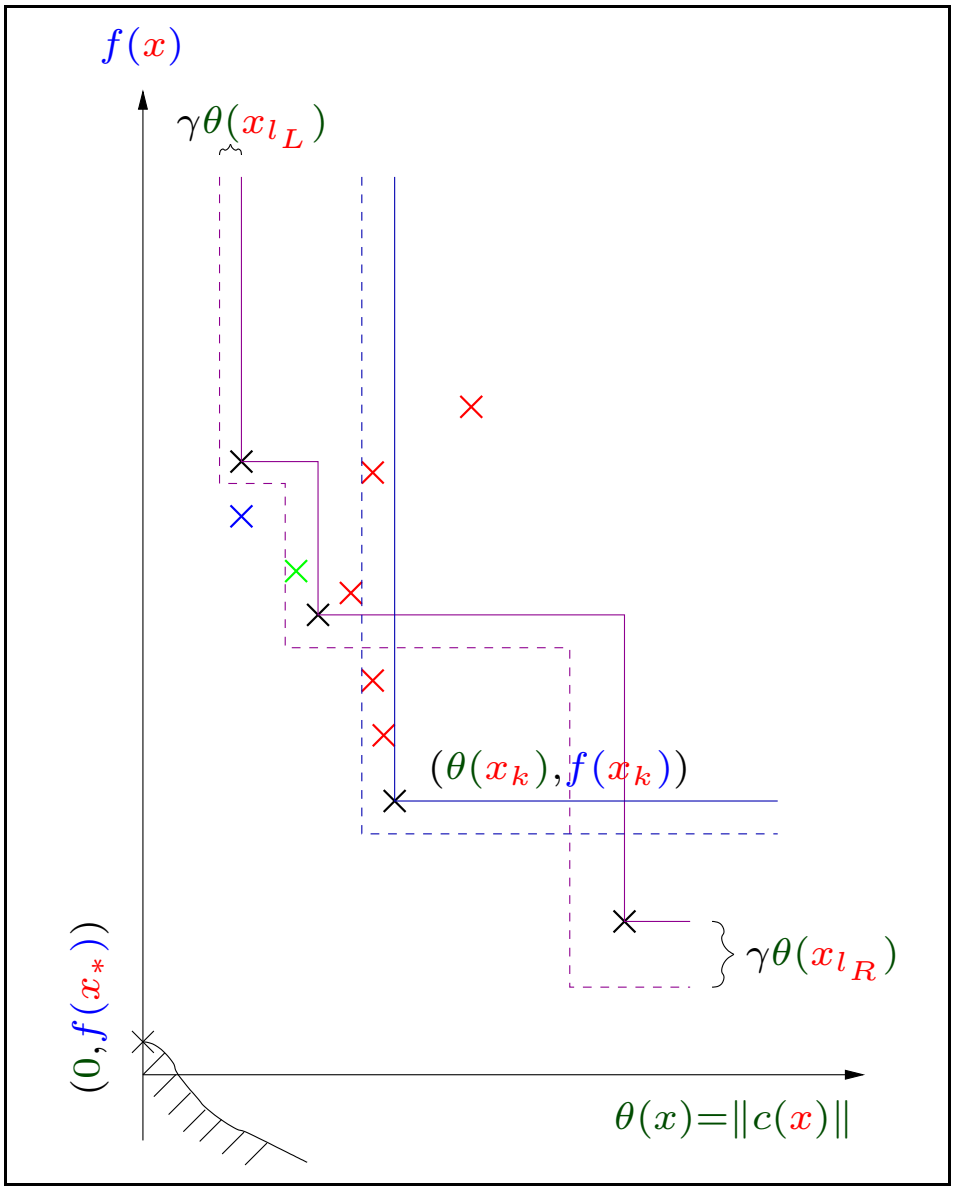
$\Longrightarrow$ Don't augment $\mathcal{F}_k$ in that case

A Filter Line Search Method (Restoration)



If no admissible step size α_k can be found

A Filter Line Search Method (Restoration)



If no admissible step size α_k can be found

$\Downarrow$

Revert to
feasibility restoration phase:

Decrease $\theta(x)$ until

- found acceptable new iterate x_{k+1}, or
- converged to local minimizer of constraint violation

Global Convergence Of Filter Method

- Assume
  - Functions and their derivatives are bounded and Lipschitz continuous
  - Singular values of $\nabla c(x_k)$ are bounded away from zero
  - W_k is bounded, and $(P_k^c)^T W_k P^c \succ \epsilon I$ for $\epsilon > 0$

Global Convergence Of Filter Method

- Assume
  - Functions and their derivatives are bounded and Lipschitz continuous
  - Singular values of $\nabla c(x_k)$ are bounded away from zero
  - W_k is bounded, and $(P_k^c)^T W_k P^c \succ \epsilon I$ for $\epsilon > 0$

- Then
  - $\lim_k \|c(x_k)\| = 0$ (feasibility)
  - $\liminf_k \|(P_k^c)^T \nabla f(x_k)\| = 0$ (optimality)

Local Convergence Of Line Search Method

- If $W_k = \nabla^2_{xx}\mathcal{L}(x_k, \lambda_k)$, then (d_k^x, d_k^λ) is Newton step

  $\Longrightarrow$ Fast local convergence if $\alpha_k = 1$ for large k

Local Convergence Of Line Search Method

- If $W_k = \nabla^2_{xx}\mathcal{L}(x_k, \lambda_k)$, then (d^x_k, d^λ_k) is Newton step

  $\Longrightarrow$ Fast local convergence if $\alpha_k = 1$ for large k

- Maratos effect:
  - Even arbitrarily close to nice local solution x_*, we can have

    $$f(x_k + d^x_k) > f(x_k) \quad \text{and} \quad \|c(x_k + d^x_k)\| > \|c(x_k)\|$$

  - Both $\Phi_\rho(x)$ and filter $\mathcal{F}_k$ globalization will reject $\alpha_k = 1$

Local Convergence Of Line Search Method

- If $W_k = \nabla^2_{xx}\mathcal{L}(x_k, \lambda_k)$, then (d^x_k, d^λ_k) is Newton step

  $\Longrightarrow$ Fast local convergence if $\alpha_k = 1$ for large k

- Maratos effect:
  - Even arbitrarily close to nice local solution x_*, we can have

    $$f(x_k + d^x_k) > f(x_k) \quad \text{and} \quad \|c(x_k + d^x_k)\| > \|c(x_k)\|$$

  - Both $\Phi_\rho(x)$ and filter $\mathcal{F}_k$ globalization will reject $\alpha_k = 1$
- Remedies:
  - Second order correction step: Additional Newton step for "$c(x) = 0$"

    $$\begin{bmatrix} W_k & \nabla c(x_k) \\ \nabla c(x_k)^T & 0 \end{bmatrix} \begin{pmatrix} \tilde{d}^x_k \\ \lambda^+_k \end{pmatrix} = - \begin{pmatrix} \nabla f(x_k) \\ c(x_k) + c(x_k + d^x_k) \end{pmatrix}$$

    Idea: Might accept $\alpha_k = 1$ for $\tilde{d}^x_k$ (which aims for less infeasibility)

Local Convergence Of Line Search Method

- If $W_k = \nabla^2_{xx}\mathcal{L}(x_k, \lambda_k)$, then (d^x_k, d^λ_k) is Newton step

  $\Longrightarrow$ Fast local convergence if $\alpha_k = 1$ for large k

- Maratos effect:
  - Even arbitrarily close to nice local solution x_*, we can have

    $$f(x_k + d^x_k) > f(x_k) \quad \text{and} \quad \|c(x_k + d^x_k)\| > \|c(x_k)\|$$

  - Both $\Phi_\rho(x)$ and filter $\mathcal{F}_k$ globalization will reject $\alpha_k = 1$
- Remedies:
  - Second order correction step: Additional Newton step for "$c(x) = 0$"

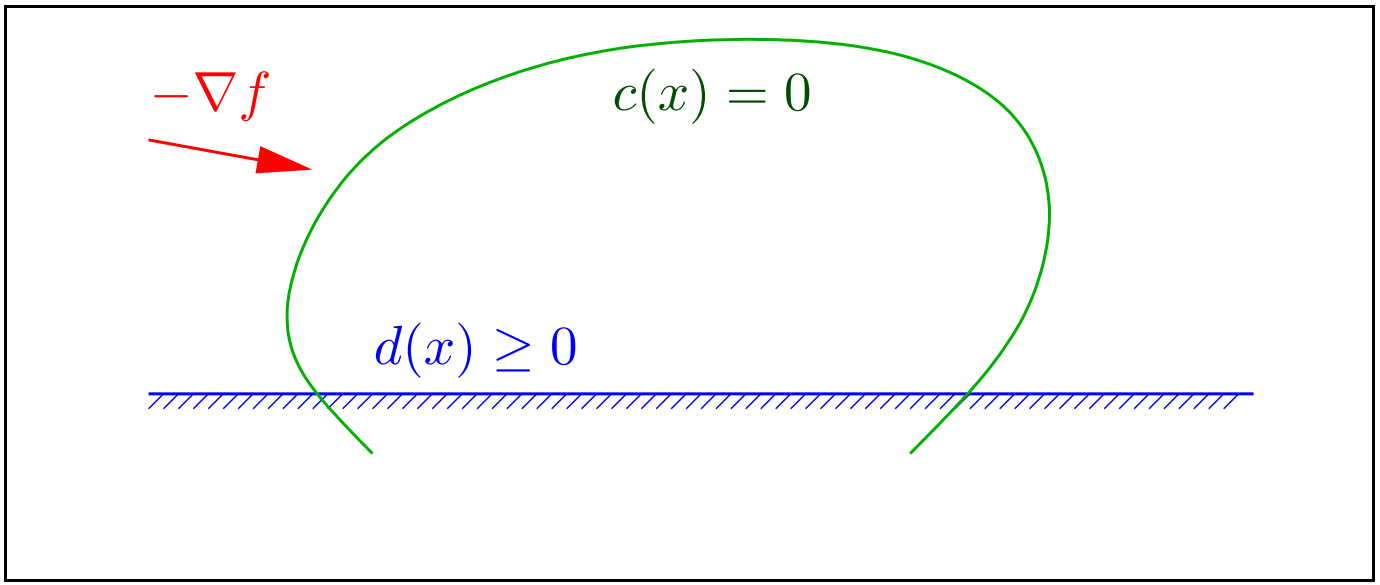
    $$\begin{bmatrix} W_k & \nabla c(x_k) \\ \nabla c(x_k)^T & 0 \end{bmatrix} \begin{pmatrix} \tilde{d}^x_k \\ \lambda^+_k \end{pmatrix} = - \begin{pmatrix} \nabla f(x_k) \\ c(x_k) + c(x_k + d^x_k) \end{pmatrix}$$

    Idea: Might accept $\alpha_k = 1$ for $\tilde{d}^x_k$ (which aims for less infeasibility)
  - Watchdog: Accept d^x_k temporarily and continue. Go back and perform line search with d^x_k if no progress after L iterations

Generally Constrainted Optimization

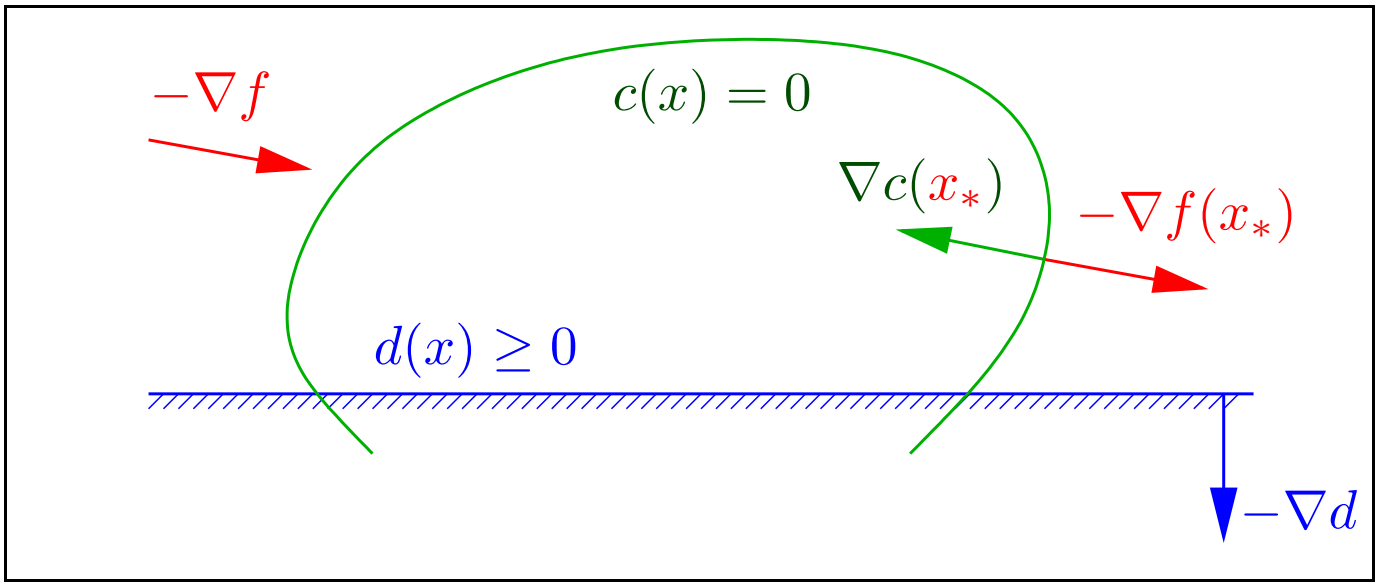
Optimality Conditions (Motivation)

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ s.t. \quad & c(x) = 0 \\ & d(x) \geq 0 \end{aligned}$$



Optimality Conditions (Motivation)

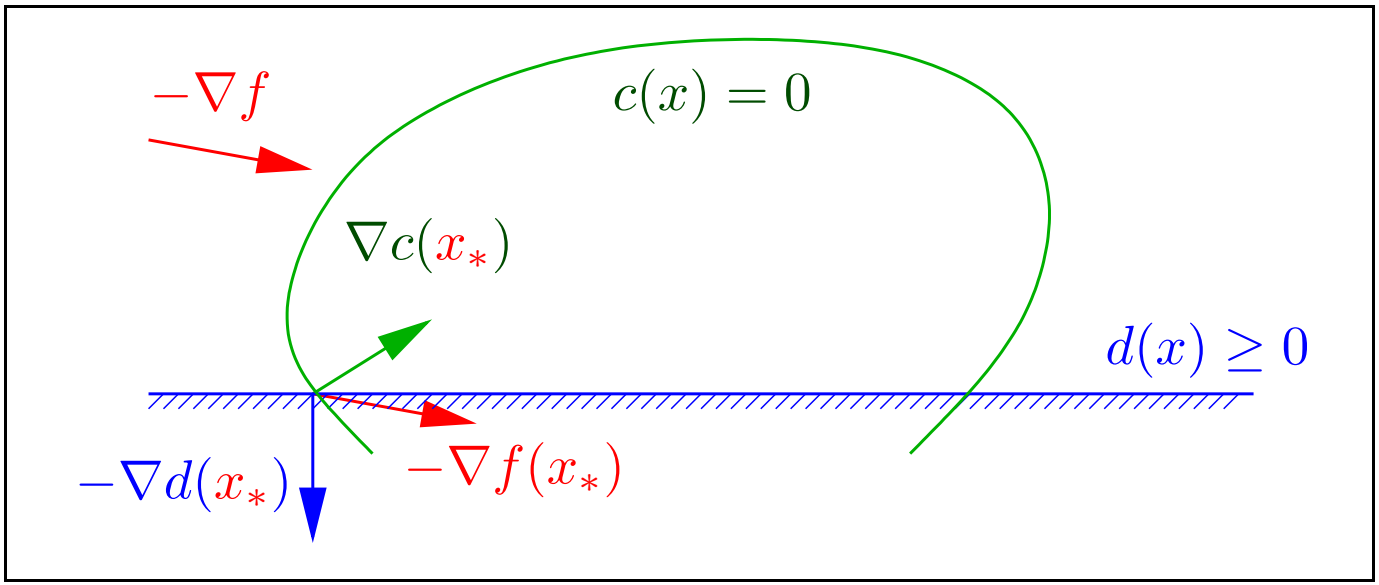
$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ s.t. \quad & c(x) = 0 \\ & d(x) \geq 0 \end{aligned}$$



$$\begin{aligned} -\nabla f(x_*) &= \nabla c(x_*)\lambda_* - \nabla d(x_*) z_* \\ c(x_*) &= 0 \\ d(x_*) &> 0 \\ z_* &= 0 \end{aligned}$$

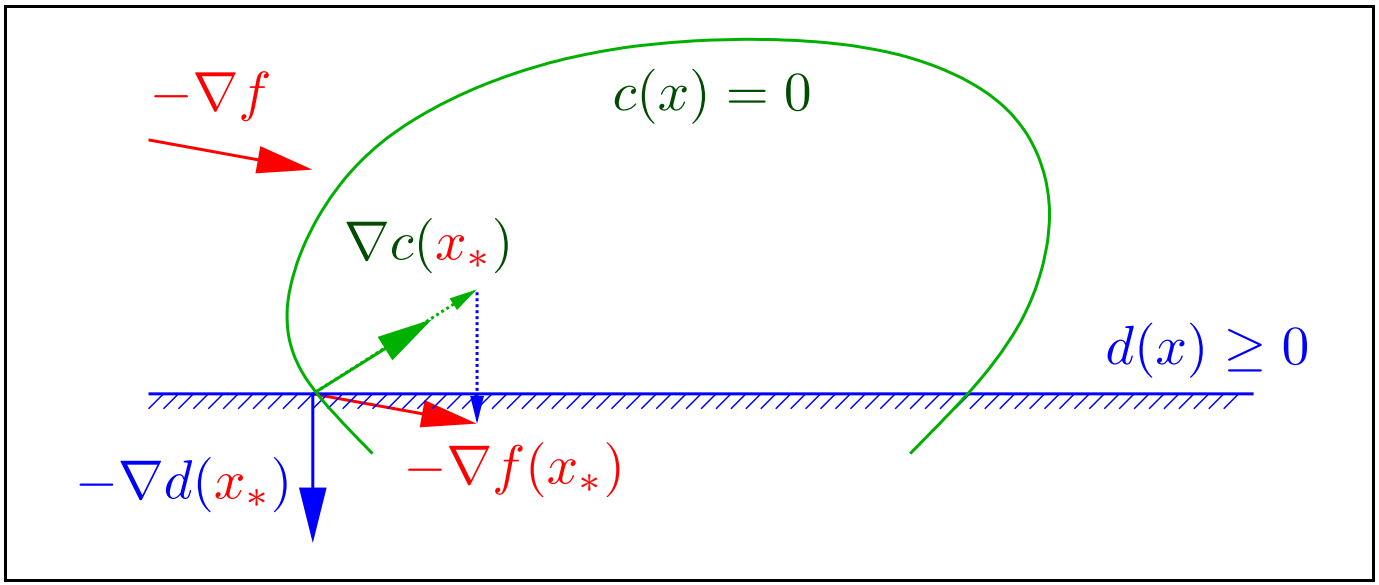
Optimality Conditions (Motivation)

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ s.t. \quad & c(x) = 0 \\ & d(x) \geq 0 \end{aligned}$$



Optimality Conditions (Motivation)

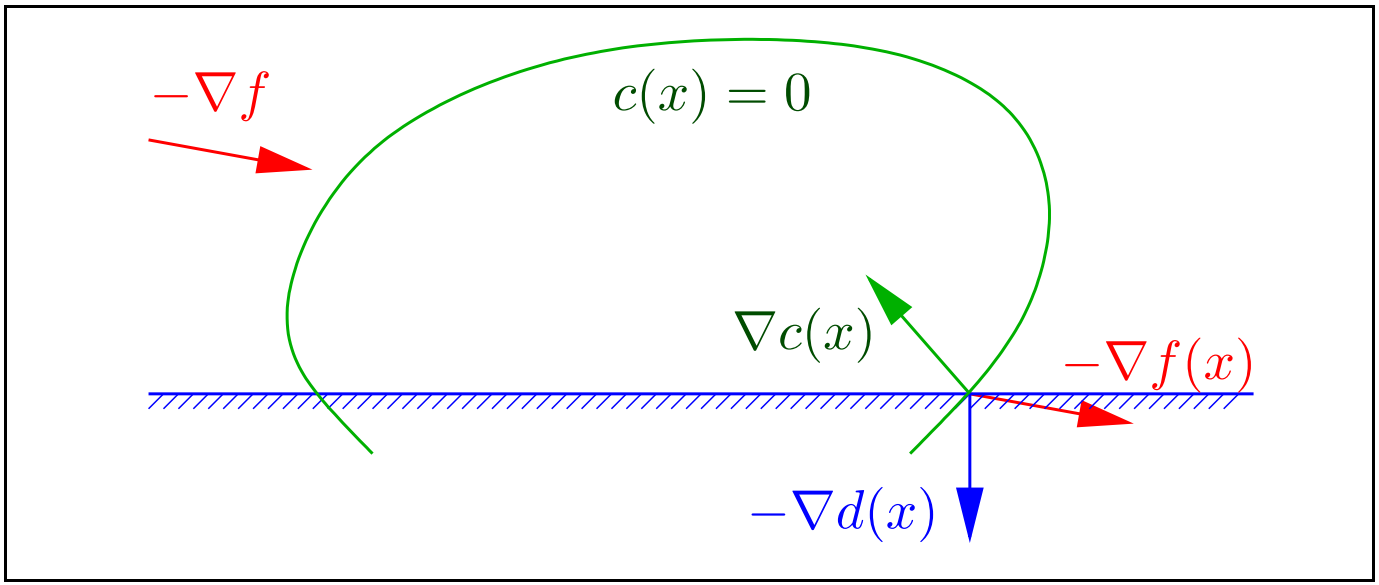
$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ s.t. \quad & c(x) = 0 \\ & d(x) \geq 0 \end{aligned}$$



$$\begin{aligned} -\nabla f(x_*) &= \nabla c(x_*)\lambda_* - \nabla d(x_*) z_* \\ c(x_*) &= 0 \\ d(x_*) &= 0 \\ z_* &> 0 \end{aligned}$$

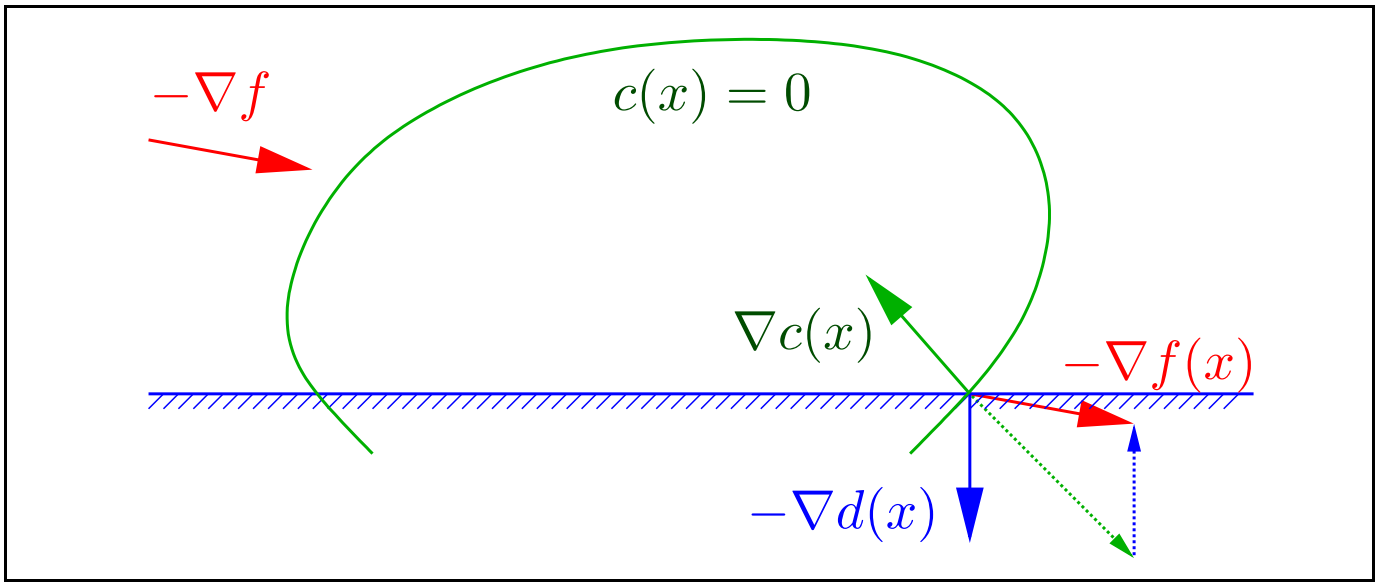
Optimality Conditions (Motivation)

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ s.t. \quad & c(x) = 0 \\ & d(x) \geq 0 \end{aligned}$$



Optimality Conditions (Motivation)

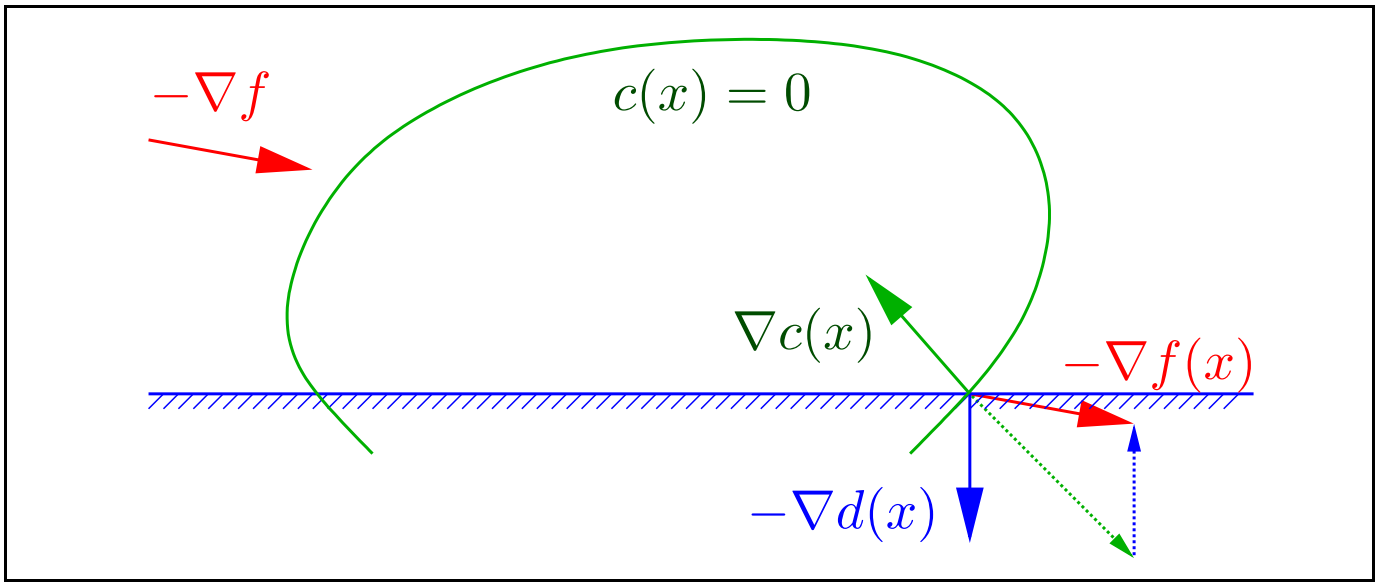
$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ s.t. \quad & c(x) = 0 \\ & d(x) \geq 0 \end{aligned}$$



$$\begin{aligned} -\nabla f(x) &= \nabla c(x)\lambda - \nabla d(x) z \\ c(x) &= 0 \\ d(x) &= 0 \\ z &< 0 \quad \text{Wrong sign!} \end{aligned}$$

Optimality Conditions (Motivation)

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ s.t. \quad & c(x) = 0 \\ & d(x) \geq 0 \end{aligned}$$

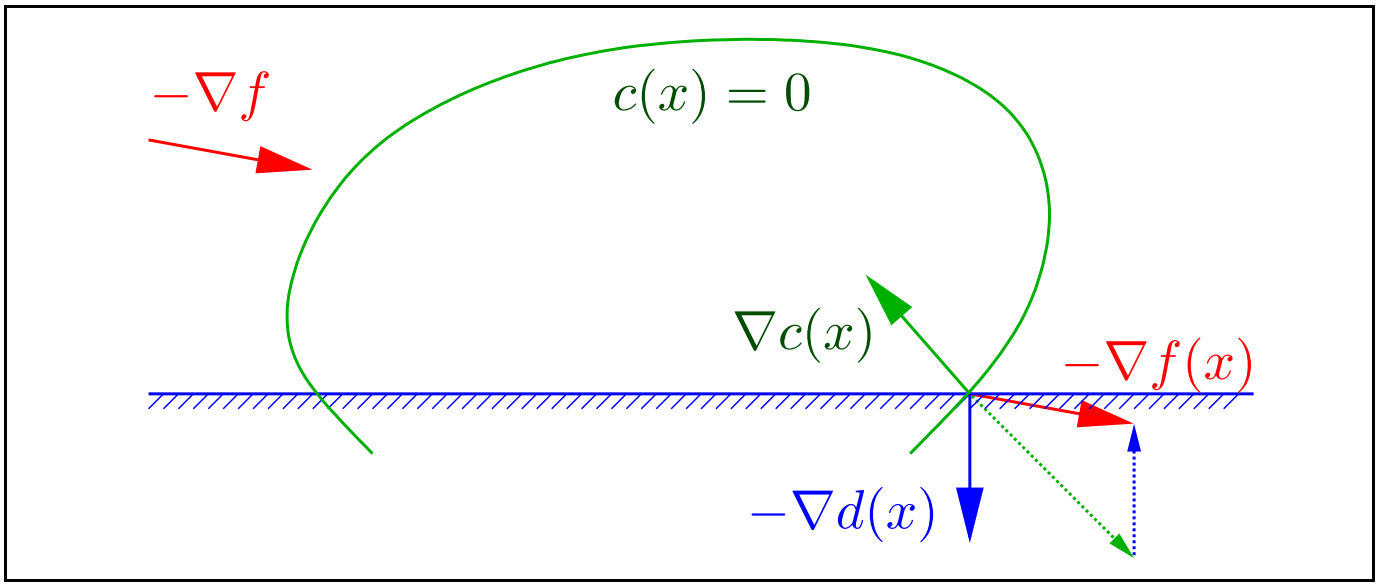


Karush-Kuhn-Tucker (KKT) conditions

$$\begin{aligned} -\nabla f(x) &= \nabla c(x)\lambda - \nabla d(x) z \\ c(x) &= 0 \\ d(x) &\geq 0 \\ z &\geq 0 \\ d(x)^T z &= 0 \end{aligned}$$

Optimality Conditions (Motivation)

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ s.t. \quad & c(x) = 0 \\ & x \geq 0 \end{aligned}$$



Karush-Kuhn-Tucker (KKT) conditions

$$\begin{aligned} -\nabla f(x) &= \nabla c(x)\lambda - z \\ c(x) &= 0 \\ x &\geq 0 \\ z &\geq 0 \\ x^T z &= 0 \end{aligned}$$

Necessary Optimality Conditions

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ s.t. \quad & c(x) = 0 \\ & x \geq 0 \end{aligned}$$

x	Variables
$f(x) : \mathbb{R}^n \longrightarrow \mathbb{R}$	Objective function
$c(x) : \mathbb{R}^n \longrightarrow \mathbb{R}^m$	Equality constraints
	Bound constraints (w.l.o.g)

- If x_* local solution (at which constraint qualification holds), then exist multipliers $\lambda_* \in \mathbb{R}^m$, $z_* \in \mathbb{R}^n$ so that KKT conditions hold:

$$\begin{aligned} \nabla f(x_*) + \nabla c(x_*)\lambda_* - z_* &= 0 && \text{dual feasibily} \\ c(x_*) &= 0 && \text{primal feasibily} \\ X_* z_* &= 0 && \text{complementarity} \\ x_*, z_* &\geq 0 && \text{non-negativity} \end{aligned}$$

- $\nabla^2_{xx} \mathcal{L}(x_*, \lambda_*, z_*)$ positive semi-definite in null space of all *active* constraints (Lagrangian: $\mathcal{L}(x, \lambda, z) = f(x) + \lambda^T c(x) - z$)

Constraint Qualifications

- Conditions that ensure that the KKT conditions hold at local solution
- Linearly-Independent Constraint Qualification (LICQ)
  - The gradients of the active constraints

$$\nabla c^{(1)}(x_*), \ldots, \nabla c^{(m)}(x_*) \quad \text{and} \quad e_i \text{ for } i \in \mathcal{A}(x_*) := \{j : x_*^{(j)} = 0\}$$

  are linearly independent.

Constraint Qualifications

- Conditions that ensure that the KKT conditions hold at local solution
- Linearly-Independent Constraint Qualification (LICQ)
  - The gradients of the active constraints

$$\nabla c^{(1)}(x_*), \ldots, \nabla c^{(m)}(x_*) \quad \text{and} \quad e_i \text{ for } i \in \mathcal{A}(x_*) := \{j : x_*^{(j)} = 0\}$$

  are linearly independent.

- Mangasarian-Fromowitz Constraint Qualification (MFCQ)
  - $\nabla c(x_*)$ has full rank
  - There exists direction $d \in \mathbb{R}^n$ so that

$$\nabla c(x_*)^T d = 0 \quad \text{and} \quad e_i^T d > 0 \text{ for } i \in \mathcal{A}(x_*)$$

Constraint Qualifications

- Conditions that ensure that the KKT conditions hold at local solution
- Linearly-Independent Constraint Qualification (LICQ)
  - The gradients of the active constraints

$$\nabla c^{(1)}(x_*), \ldots, \nabla c^{(m)}(x_*) \quad \text{and} \quad e_i \text{ for } i \in \mathcal{A}(x_*) := \{j : x_*^{(j)} = 0\}$$

  are linearly independent.

- Mangasarian-Fromowitz Constraint Qualification (MFCQ)
  - $\nabla c(x_*)$ has full rank
  - There exists direction $d \in \mathbb{R}^n$ so that

$$\nabla c(x_*)^T d = 0 \quad \text{and} \quad e_i^T d > 0 \text{ for } i \in \mathcal{A}(x_*)$$

- MFCQ is weaker than LICQ

Sufficient Second Order Conditions

- Assume:
  - LICQ holds at x_*
  - There exist $\lambda_* \in \mathbb{R}^m$ and $z_* \in \mathbb{R}^n$ so that the KKT conditions hold
  - $\nabla^2_{xx}\mathcal{L}(x_*, \lambda_*, z_*)$ is positive definite in null space of all *active* constraints
  - Strict complementarity holds:

$$x_*^{(i)} > 0 \quad \text{or} \quad z_*^{(i)} > 0 \qquad \text{for all } i \in \mathcal{A}(x_*)$$

- Then: x_* is a (strict) local solution

Newton's Method (again...?)

$$\begin{aligned} \nabla f(x) + \nabla c(x) - z &= 0 \\ c(x) &= 0 \\ Xz &= 0 \\ x, z &\geq 0 \end{aligned}$$

Newton's Method (again...?)

$$\begin{aligned} \nabla f(x) + \nabla c(x) - z &= 0 \\ c(x) &= 0 \\ Xz &= 0 \\ x, z &\geq 0 \end{aligned}$$

- Complications: Cannot use Newton's method this time because of the non-negativity conditions (might end up at point with wrong sign)

Newton's Method (again...?)

$$\begin{aligned} \nabla f(x) + \nabla c(x) - z &= 0 \\ c(x) &= 0 \\ Xz &= 0 \\ x, z &\geq 0 \end{aligned}$$

- Complications: Cannot use Newton's method this time because of the non-negativity conditions (might end up at point with wrong sign)
- But we can use the SQP idea (with a "local QP model")

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ s.t. \quad & c(x) = 0 \\ & x \geq 0 \end{aligned}$$

$\longrightarrow$

$$\begin{aligned} \min_{d \in \mathbb{R}^n} \quad & f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T W_k d \\ s.t. \quad & c(x_k) + \nabla c(x_k)^T d = 0 \\ & x_k + d \geq 0 \end{aligned}$$

Sequential Quadratic Programming

$$\begin{aligned} \min_{d \in \mathbb{R}^n} \quad & f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T W_k d \\ s.t. \quad & c(x_k) + \nabla c(x_k)^T d = 0 \\ & x_k + d \geq 0 \end{aligned}$$

- If x_* satisfies sufficient second order conditions and $x_k \to x_*$:
  - For large k: $\mathcal{A}^{\mathrm{QP}}(x_k) = \mathcal{A}^{\mathrm{NLP}}(x_*)$

    (The QP "identifies the correct active set")

Sequential Quadratic Programming

$$\begin{aligned} \min_{d \in \mathbb{R}^n} \quad & f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T W_k d \\ s.t. \quad & c(x_k) + \nabla c(x_k)^T d = 0 \\ & x_k + d \geq 0 \end{aligned}$$

- If x_* satisfies sufficient second order conditions and $x_k \to x_*$:
  - For large k: $\mathcal{A}^{\mathrm{QP}}(x_k) = \mathcal{A}^{\mathrm{NLP}}(x_*)$

    (The QP "identifies the correct active set")
  - Then, the steps are Newton steps for the equality-constrained NLP

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ s.t. \quad & c(x) = 0 \\ & x^{(j)} = 0 \qquad \text{for } j \in \mathcal{A}^{\mathrm{NLP}}(x_*) \end{aligned}$$

  - Newton's method! $\Longrightarrow$ Quadratic local convergence

Discussion Of SQP

- Originators: Han (1977), Powell (1978)
- Most popular method for NLP (still. . . ?)
- Many different algorithms and implementations:
  - using 2nd derivatives vs. approximating them with quasi-Newton methods
  - line search vs. trust region
  - different merit functions vs. filter
  - Some codes: DONLP2, FilterSQP, FSQP, NPSOL, Sℓ_1QP, SNOPT, and many more. . .
- Crutial question: How do we solve the QP?

QP Solvers in SQP

- Active set solvers:
  - guess which inequalities are active
  - solve corresponding equality constrained QP
    (requires solution of linear equations)
  - correct guess until solution is found
  - towards the end of SQP algorithm, $\mathcal{A}^{\mathrm{QP}}(x_k)$ stays the same
    (if x_* satisfies 2nd order optimality conditions)
  - QP solver can be "warm-started"
  - for large problems: combinatorial complexity to determine $\mathcal{A}^{\mathrm{QP}}(x_k)$ becomes the bottleneck

QP Solvers in SQP

- Active set solvers:
  - guess which inequalities are active
  - solve corresponding equality constrained QP
    (requires solution of linear equations)
  - correct guess until solution is found
  - towards the end of SQP algorithm, $\mathcal{A}^{\mathrm{QP}}(x_k)$ stays the same
    (if x_* satisfies 2nd order optimality conditions)
  - QP solver can be "warm-started"
  - for large problems: combinatorial complexity to determine $\mathcal{A}^{\mathrm{QP}}(x_k)$ becomes the bottleneck
- Interior point QP solvers
  - avoid combinatorial complexity
  - can tackle large problems
  - but no good warm start possibilities (yet?)

QP Solvers in SQP

- Active set solvers:
  - guess which inequalities are active
  - solve corresponding equality constrained QP
    (requires solution of linear equations)
  - correct guess until solution is found
  - towards the end of SQP algorithm, $\mathcal{A}^{\mathrm{QP}}(x_k)$ stays the same
    (if x_* satisfies 2nd order optimality conditions)
  - QP solver can be "warm-started"
  - for large problems: combinatorial complexity to determine $\mathcal{A}^{\mathrm{QP}}(x_k)$ becomes the bottleneck
- Interior point QP solvers
  - avoid combinatorial complexity
  - can tackle large problems
  - but no good warm start possibilities (yet?)
- Since early '90s: Apply interior point idea directly to NLP

Interior Point Methods

$$\begin{aligned} \nabla f(x) - \nabla c(x)\lambda - z &= 0 \\ c(x) &= 0 \\ XZe &= 0 \\ (x, z &\geq 0) \end{aligned}$$

- Here: $e = (1, \ldots, 1)^T$

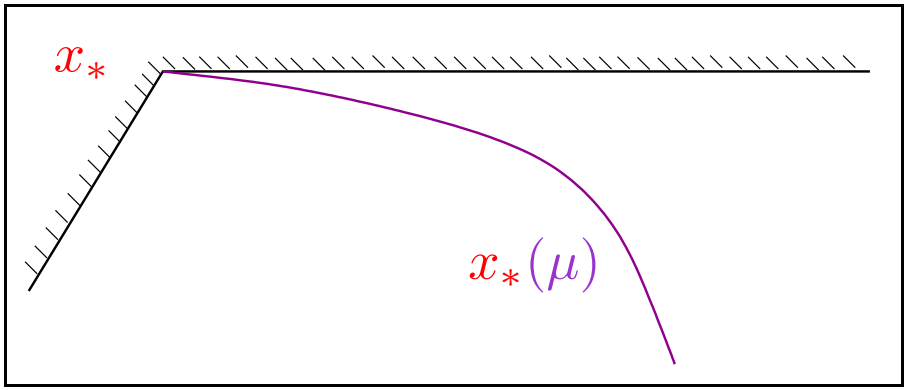
Interior Point Methods

$$\begin{aligned} \nabla f(x) - \nabla c(x)\lambda - z &= 0 \\ c(x) &= 0 \\ XZe &= \mu e \\ (x, z &> 0) \end{aligned}$$

- Here: $e = (1, \ldots, 1)^T$
- Relax complementarity conditions with parameter $\mu > 0$

Interior Point Methods

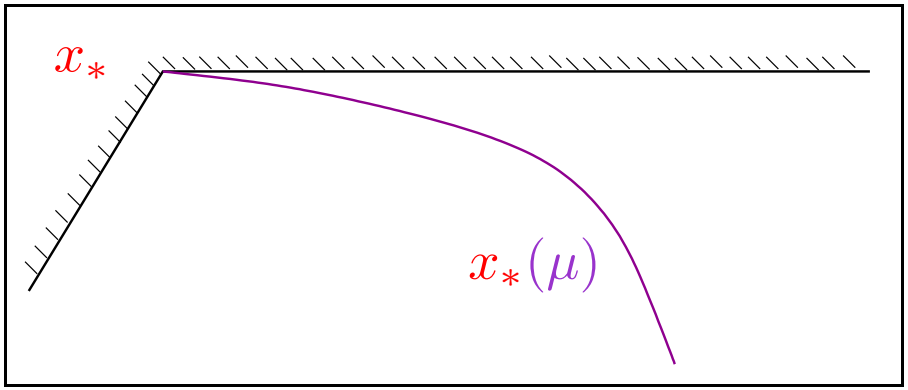
$$\begin{aligned} \nabla f(x) - \nabla c(x)\lambda - z &= 0 \\ c(x) &= 0 \\ XZe &= \mu e \\ (x, z &> 0) \end{aligned}$$



- Here: $e = (1, \ldots, 1)^T$
- Relax complementarity conditions with parameter $\mu > 0$
- Trace solutions $x_*(\mu)$ as function of μ
- If x_* strong local solution: $x_*(\mu) \to x_*$ as $\mu \to 0$

Interior Point Methods

$$\begin{aligned} \nabla f(x) - \nabla c(x)\lambda - z &= 0 \\ c(x) &= 0 \\ XZe &= \mu e \\ (x, z &> 0) \end{aligned}$$



- Here: $e = (1, \ldots, 1)^T$
- Relax complementarity conditions with parameter $\mu > 0$
- Trace solutions $x_*(\mu)$ as function of μ
- If x_* strong local solution: $x_*(\mu) \to x_*$ as $\mu \to 0$
- Homotopy approach: Trace $x_*(\mu_k)$ and drive $\mu_k \to 0$
- Approaching the solution from the interior $x_k > 0$

Newton's Method (yes again!)

$$\begin{aligned} \nabla f(x) - \nabla c(x)\lambda - z &= 0 \\ c(x) &= 0 \\ XZe &= \mu e \\ (x, z &> 0) \end{aligned}$$

Newton steps

$$\begin{bmatrix} W_k & \nabla c(x_k) & -I \\ \nabla c(x_k)^T & 0 & 0 \\ Z_k & 0 & X_k \end{bmatrix} \begin{pmatrix} d_k^x \\ d_k^\lambda \\ d_k^z \end{pmatrix} = - \begin{pmatrix} \nabla f(x_k) + \nabla c(x_k)\lambda_k - z_k \\ c(x_k) \\ X_k Z_k e - \mu e \end{pmatrix}$$

Newton's Method (yes again!)

$$\begin{aligned} \nabla f(x) - \nabla c(x)\lambda - z &= 0 \\ c(x) &= 0 \\ XZe &= \mu e \\ (x, z &> 0) \end{aligned}$$

Newton steps

$$\begin{bmatrix} W_k & \nabla c(x_k) & -I \\ \nabla c(x_k)^T & 0 & 0 \\ Z_k & 0 & X_k \end{bmatrix} \begin{pmatrix} d_k^x \\ d_k^\lambda \\ d_k^z \end{pmatrix} = - \begin{pmatrix} \nabla f(x_k) + \nabla c(x_k)\lambda_k - z_k \\ c(x_k) \\ X_k Z_k e - \mu e \end{pmatrix}$$

- If x_* satisfies sufficient 2nd order optimality conditions:
  - Matrix is nonsingular at solution (x_*, λ_*, z_*)
  - $\Longrightarrow$ Can devise superlinearly convergent method if μ_k is updated in particular ways
- Very established for LP, QP, convex problems (e.g. SDP)

Augmented System

$$\begin{bmatrix} W_k & \nabla c(x_k) & -I \\ \nabla c(x_k)^T & 0 & 0 \\ Z_k & 0 & X_k \end{bmatrix} \begin{pmatrix} d_k^x \\ d_k^\lambda \\ d_k^z \end{pmatrix} = - \begin{pmatrix} \nabla f(x_k) + \nabla c(x_k)\lambda_k - z_k \\ c(x_k) \\ X_k Z_k e - \mu e \end{pmatrix}$$

can be solved equivalently by first solving the "augmented system"

$$\begin{bmatrix} W_k + \Sigma_k & \nabla c(x_k) \\ \nabla c(x_k)^T & 0 \end{bmatrix} \begin{pmatrix} d_k^x \\ d_k^\lambda \end{pmatrix} = - \begin{pmatrix} \nabla f(x_k) - \mu X_k^{-1} e + \nabla c(x_k)\lambda_k \\ c(x_k) \end{pmatrix}$$

with $\Sigma_k = X_k^{-1} Z_k$, and then compute

$$d_k^z = \mu X_k^{-1} e - z_k - \Sigma_k d_k^x$$

Barrier Methods

Barrier Problem (fixed μ)

$$\begin{array}{ll} \min\limits_{x\in\mathbb{R}^n} & \varphi_\mu(x) := f(x) - \mu \sum \ln(x^{(i)}) \\ s.t. & c(x) = 0 \end{array} \qquad (B(\mu))$$

- Parameterized by "barrier parameter" $\mu > 0$
- Fiacco-McCormick (1968)
  1. Choose $\mu_0 > 0$, $x_0 > 0$; set $l \leftarrow 0$
  2. Solve $(B(\mu_l))$ from x_l to relaxed tolerance to obtain x_{l+1}
  3. Update $\mu_{l+1} \in (0, \mu_l)$; set $l \leftarrow l + 1$; go back to 2.

Barrier Methods

Barrier Problem (fixed μ)

$$\begin{array}{ll} \min\limits_{x\in\mathbb{R}^n} & \varphi_\mu(x) := f(x) - \mu \sum \ln(x^{(i)}) \\ s.t. & c(x) = 0 \end{array} \qquad (B(\mu))$$

- Parameterized by "barrier parameter" $\mu > 0$
- Fiacco-McCormick (1968)
  1. Choose $\mu_0 > 0$, $x_0 > 0$; set $l \leftarrow 0$
  2. Solve $(B(\mu_l))$ from x_l to relaxed tolerance to obtain x_{l+1}
  3. Update $\mu_{l+1} \in (0, \mu_l)$; set $l \leftarrow l + 1$; go back to 2.
- Solve a sequence of barrier problems
  - can use known techniques for equality constrianed optimization
  - know how to globalize and handle non-convexity (almost. . .)
  - fast local convergence? (only for fixed μ_l)

Optimality Conditions For $(B(\mu))$

$$\begin{aligned} \nabla\varphi_\mu(x) + \nabla c(x)\lambda &= 0 \\ c(x) &= 0 \\ (x &> 0) \end{aligned}$$

Optimality Conditions For $(B(\mu))$

$$\begin{aligned} \nabla\varphi_\mu(x) + \nabla c(x)\lambda &= 0 \\ c(x) &= 0 \\ (x &> 0) \end{aligned}$$

$$\downarrow$$

$$\begin{aligned} \nabla f(x) - \mu X^{-1} e + \nabla c(x)\lambda &= 0 \\ c(x) &= 0 \\ (x &> 0) \end{aligned}$$

Optimality Conditions For $(B(\mu))$

$$\begin{aligned} \nabla\varphi_\mu(x) + \nabla c(x)\lambda &= 0 \\ c(x) &= 0 \\ (x &> 0) \end{aligned}$$

$$\downarrow$$

$$\begin{aligned} \nabla f(x) - \mu X^{-1} e + \nabla c(x)\lambda &= 0 \\ c(x) &= 0 \\ (x &> 0) \end{aligned}$$

Disadvanatges:

- As $\mu_l \to 0$ and $x_l \to x_*$, optimality conditions not defined
- Newton's method for $(B(\mu_l))$ has increasingly smaller "radius of convergence" as $\mu_l \to 0$.

Newton Method For Barrier Problem

$$\begin{bmatrix} W_k + \mu X_k^{-2} & \nabla c(x_k) \\ \nabla c(x_k)^T & 0 \end{bmatrix} \begin{pmatrix} d_k^x \\ \lambda_k^+ \end{pmatrix} = - \begin{pmatrix} \nabla f(x_k) - \mu X_k^{-1} e + \nabla c(x_k) \lambda_k \\ c(x_k) \end{pmatrix}$$

with $W_k = \nabla^2_{xx} \mathcal{L}(x_k, \lambda_k, z_k) = \nabla^2 f(x_k) + \sum_j \lambda_k^{(j)} \nabla^2 c^{(j)}(x_k)$.

Newton Method For Barrier Problem

$$\begin{bmatrix} W_k + \mu X_k^{-2} & \nabla c(x_k) \\ \nabla c(x_k)^T & 0 \end{bmatrix} \begin{pmatrix} d_k^x \\ \lambda_k^+ \end{pmatrix} = - \begin{pmatrix} \nabla f(x_k) - \mu X_k^{-1} e + \nabla c(x_k) \lambda_k \\ c(x_k) \end{pmatrix}$$

with $W_k = \nabla^2_{xx} \mathcal{L}(x_k, \lambda_k, z_k) = \nabla^2 f(x_k) + \sum_j \lambda_k^{(j)} \nabla^2 c^{(j)}(x_k)$.

Looks familiar. . . ?

Newton Method For Barrier Problem

$$\begin{bmatrix} W_k + \mu X_k^{-2} & \nabla c(x_k) \\ \nabla c(x_k)^T & 0 \end{bmatrix} \begin{pmatrix} d_k^x \\ \lambda_k^+ \end{pmatrix} = - \begin{pmatrix} \nabla f(x_k) - \mu X_k^{-1} e + \nabla c(x_k) \lambda_k \\ c(x_k) \end{pmatrix}$$

with $W_k = \nabla^2_{xx} \mathcal{L}(x_k, \lambda_k, z_k) = \nabla^2 f(x_k) + \sum_j \lambda_k^{(j)} \nabla^2 c^{(j)}(x_k)$.

Looks familiar. . . ? Remember, for interior point method

$$\begin{bmatrix} W_k + \Sigma_k & \nabla c(x_k) \\ \nabla c(x_k)^T & 0 \end{bmatrix} \begin{pmatrix} d_k^x \\ d_k^\lambda \end{pmatrix} = - \begin{pmatrix} \nabla f(x_k) - \mu X_k^{-1} e + \nabla c(x_k) \lambda_k \\ c(x_k) \end{pmatrix}$$

where $\Sigma_k = X_k^{-1} Z_k$.

Newton Method For Barrier Problem

$$\begin{bmatrix} W_k + \mu X_k^{-2} & \nabla c(x_k) \\ \nabla c(x_k)^T & 0 \end{bmatrix} \begin{pmatrix} d_k^x \\ \lambda_k^+ \end{pmatrix} = - \begin{pmatrix} \nabla f(x_k) - \mu X_k^{-1} e + \nabla c(x_k) \lambda_k \\ c(x_k) \end{pmatrix}$$

with $W_k = \nabla^2_{xx} \mathcal{L}(x_k, \lambda_k, z_k) = \nabla^2 f(x_k) + \sum_j \lambda_k^{(j)} \nabla^2 c^{(j)}(x_k)$.

Looks familiar. . . ? Remember, for interior point method

$$\begin{bmatrix} W_k + \Sigma_k & \nabla c(x_k) \\ \nabla c(x_k)^T & 0 \end{bmatrix} \begin{pmatrix} d_k^x \\ d_k^\lambda \end{pmatrix} = - \begin{pmatrix} \nabla f(x_k) - \mu X_k^{-1} e + \nabla c(x_k) \lambda_k \\ c(x_k) \end{pmatrix}$$

where $\Sigma_k = X_k^{-1} Z_k$.

Only difference in "Hessian for barrier term $-\mu \sum_i \log(x^{(i)})$"

- "primal": $\Sigma_k = \mu X_k^{-2}$
- "primal-dual": $\Sigma_k = X_k^{-1} Z_k$
  (maintain iterates for bound multipliers z_k)

Primal-Dual Interior Point Method

$$\begin{bmatrix} W_k + \Sigma_k & \nabla c(x_k) \\ \nabla c(x_k)^T & 0 \end{bmatrix} \begin{pmatrix} d_k^x \\ d_k^\lambda \end{pmatrix} = - \begin{pmatrix} \nabla f(x_k) - \mu X_k^{-1} e + \nabla c(x_k) \lambda_k \\ c(x_k) \end{pmatrix}$$

- Primal method:
  - $\Sigma_k = \mu X_k^{-2}$
  - have good methods for solving $(B(\mu))$ for each μ
  - gives us global convergence
- Primal-dual method:
  - $\Sigma_k = X_k^{-1} Z_k$
  - can achieve fast local convergence, if μ is updated properly
- Primal-dual approach takes advantage of both and is basis for most modern interior point methods for NLP

A Primal-Dual IPM For NLP

1. Given (x_0, λ_0, z_0) with $x_0, z_0 > 0$; $\mu_0 > 0$; set $k \leftarrow 0$
2. Compute primal-dual search direction $(d_k^x, d_k^\lambda, d_k^z)$
3. Determine largest $\alpha_k^\tau, \alpha_k^{z,\tau} \in (0, 1]$ such that (usually $\tau \approx 0.99$)

$$\begin{array}{rcccl} x_k + \alpha_k^\tau d_k^x & \geq & (1-\tau)x_k & > & 0 \\ z_k + \alpha_k^{z,\tau} d_k^z & \geq & (1-\tau)z_k & > & 0 \end{array}$$

4. Perform backtracking line search to find acceptable $\alpha_{k,l} = 2^{-l}\alpha_k^\tau$, using globalization method for $(B(\mu_k))$
5. Accept new iterate

$$\begin{array}{rcl} (x_{k+1}, \lambda_{k+1}) & = & (x_k, \lambda_k) + \alpha_k \left(d_k^x, d_k^\lambda\right) \\ z_{k+1} & = & z_k + \alpha_k^z d_k^z \end{array}$$

6. If $(B(\mu_k))$ is solved sufficiently well, decrease $\mu_{k+1} \in (0, \mu_k)$; otherwise set $\mu_{k+1} \leftarrow \mu_k$
7. Go to step 2.

Example Problem

$$
\begin{aligned}
\min_{x \in \mathbb{R}^3} \quad & x^{(1)} \\
s.t. \quad & (x^{(1)})^2 - x^{(2)} - 1 = 0 \\
& x^{(1)} - x^{(3)} - 0.5 = 0 \\
& x^{(2)}, x^{(3)} \geq 0
\end{aligned}
$$

Example Problem

$$\begin{aligned} \min_{x \in \mathbb{R}^3} \quad & x^{(1)} \\ s.t. \quad & (x^{(1)})^2 - x^{(2)} - 1 = 0 \\ & x^{(1)} - x^{(3)} - 0.5 = 0 \\ & x^{(2)}, x^{(3)} \geq 0 \end{aligned}$$

- $f(x)$ and $c(x)$ smooth
- $\nabla c(x)$ full rank everywhere
- Only one stationary point:

$$\begin{aligned} x_* &= (1, 0, 0.5)^T \\ \lambda_* &= (-0.5, 0)^T \\ (z_*^{(2)}, z_*^{(3)})^T &= (0.5, 0)^T \end{aligned}$$

- Sufficient 2nd order optimality cond.
- Strict complementarity

Example Problem

$$\begin{aligned} \min_{x \in \mathbb{R}^3} \quad & x^{(1)} \\ s.t. \quad & (x^{(1)})^2 - x^{(2)} - 1 = 0 \\ & x^{(1)} - x^{(3)} - 0.5 = 0 \\ & x^{(2)}, x^{(3)} \geq 0 \end{aligned}$$

- $f(x)$ and $c(x)$ smooth
- $\nabla c(x)$ full rank everywhere
- Only one stationary point:

$$\begin{aligned} x_* &= (1, 0, 0.5)^T \\ \lambda_* &= (-0.5, 0)^T \\ (z_*^{(2)}, z_*^{(3)})^T &= (0.5, 0)^T \end{aligned}$$

- Sufficient 2nd order optimality cond.
- Strict complementarity

This problem is well–posed!

Example Problem

$$\begin{aligned} \min_{x \in \mathbb{R}^3} \quad & x^{(1)} \\ s.t. \quad & (x^{(1)})^2 - x^{(2)} - 1 = 0 \\ & x^{(1)} - x^{(3)} - 0.5 = 0 \\ & x^{(2)}, x^{(3)} \geq 0 \end{aligned}$$

- $f(x)$ and $c(x)$ smooth
- $\nabla c(x)$ full rank everywhere
- Only one stationary point:

$$\begin{aligned} x_* &= (1, 0, 0.5)^T \\ \lambda_* &= (-0.5, 0)^T \\ (z_*^{(2)}, z_*^{(3)})^T &= (0.5, 0)^T \end{aligned}$$

- Sufficient 2nd order optimality cond.
- Strict complementarity

This problem is well–posed!

Let's try starting point $x_0 = (-2, 3, 1)^T \ldots$

But...

ITER	X(1)	X(2)	X(3)
0	-0.20000000D+01	0.30000000D+01	0.10000000D+01
1	-0.18947413D+01	0.25789655D+01	0.10000000D-01
2	-0.12201587D+01	0.25789655D-01	0.23540510D-02
3	-0.12082251D+01	0.25789655D-03	0.93113038D-03
4	-0.12080095D+01	0.25789655D-05	0.15850601D-03
5	-0.12079893D+01	0.14108060D-05	0.15850601D-05
6	-0.12079884D+01	0.14108060D-07	0.10157878D-06
7	-0.12079884D+01	0.14108060D-09	0.22892307D-08
8	-0.12079884D+01	0.21513916D-11	0.22892307D-10
9	-0.12079884D+01	0.21513916D-13	0.25068098D-12
10	-0.12079884D+01	0.21513916D-15	0.25450869D-14

From starting point $x^0 = (-2, 3, 1)^T$, original IPOPT (using line search with primal-dual Newton steps) crashes into bounds!

Different starting points give similar results...

Failure of Global Convergence

$$\begin{aligned} \min_{x \in \mathbb{R}^3} \quad & f(x) \\ s.t. \quad & (x^{(1)})^2 - x^{(2)} - 1 = 0 \\ & x^{(1)} - x^{(3)} - 0.5 = 0 \\ & x^{(2)}, x^{(3)} \geq 0 \end{aligned}$$

Failure of Global Convergence

$$\min_{x \in \mathbb{R}^3} \quad f(x)$$

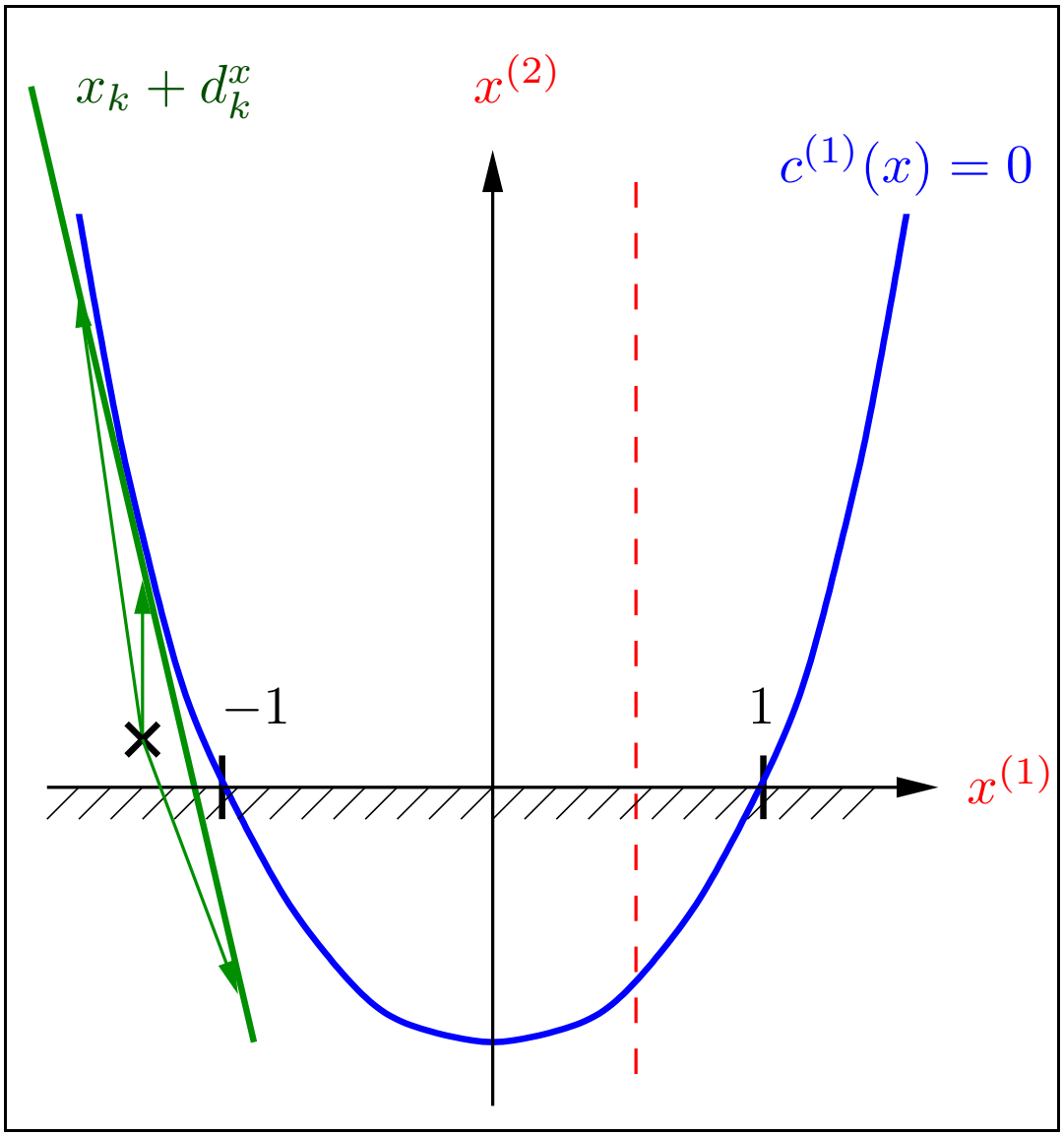
$$s.t. \quad (x^{(1)})^2 - x^{(2)} - 1 = 0$$

$$x^{(1)} - x^{(3)} - 0.5 = 0$$

$$x^{(2)}, x^{(3)} \geq 0$$

Step computation

1. $\nabla c(x_k)^T d_k^x + c(x_k) = 0$
2. ${x_k}^{(i)} + \alpha_k^x (d_k^x)^{(i)} > 0$



Discussion Of Convergence Failure

Step computation

1. $\nabla c(x_k)^T d_k^x + c(x_k) = 0$
2. $x_k^{(i)} + \alpha_k^x (d_k^x)^{(i)} > 0$

- Search direction satisfying linearization of constraints keeps pointing "outside" so that an increasingly smaller fraction of step can be taken

Discussion Of Convergence Failure

Step computation

1. $\nabla c(x_k)^T d_k^x + c(x_k) = 0$
2. $x_k^{(i)} + \alpha_k^x (d_k^x)^{(i)} > 0$

- Search direction satisfying linearization of constraints keeps pointing "outside" so that an increasingly smaller fraction of step can be taken
- Limit point is not a local minimizer of constraint violation
  - This would be a good reason to stop

Discussion Of Convergence Failure

Step computation

1. $\nabla c(x_k)^T d_k^x + c(x_k) = 0$
2. $x_k^{(i)} + \alpha_k^x (d_k^x)^{(i)} > 0$

- Search direction satisfying linearization of constraints keeps pointing "outside" so that an increasingly smaller fraction of step can be taken
- Limit point is not a local minimizer of constraint violation
  - This would be a good reason to stop
- Inconsistency between search direction and step size
  - In SQP framework: QP is infeasible, and algorithm detects problem

Discussion Of Convergence Failure

Step computation

1. $\nabla c(x_k)^T d_k^x + c(x_k) = 0$
2. $x_k^{(i)} + \alpha_k^x (d_k^x)^{(i)} > 0$

- Search direction satisfying linearization of constraints keeps pointing "outside" so that an increasingly smaller fraction of step can be taken
- Limit point is not a local minimizer of constraint violation
  - This would be a good reason to stop
- Inconsistency between search direction and step size
  - In SQP framework: QP is infeasible, and algorithm detects problem
- Global convergence proofs existed for affected IPM:
  - Implicitly assumed that iterates stay away from bounds for $\mu > 0$.

Discussion Of Convergence Failure

Step computation

1. $\nabla c(x_k)^T d_k^x + c(x_k) = 0$
2. $x_k^{(i)} + \alpha_k^x (d_k^x)^{(i)} > 0$

- Search direction satisfying linearization of constraints keeps pointing "outside" so that an increasingly smaller fraction of step can be taken
- Limit point is not a local minimizer of constraint violation
  - This would be a good reason to stop
- Inconsistency between search direction and step size
  - In SQP framework: QP is infeasible, and algorithm detects problem
- Global convergence proofs existed for affected IPM:
  - Implicitly assumed that iterates stay away from bounds for $\mu > 0$.
- In order to overcome convergence problem, we must allow steps that don't satisfy linearization of the constraints

Remedies For Convergence Failure

- Penalizing constraint violation

  Penalty-barrier method:

  - (Fiacco, McCormick)

    (Forsgren, Gill)

$$\min \quad f(x) - \mu \sum_i \ln(x^{(i)}) + \frac{1}{2\mu}\|c(x)\|_2^2$$

Remedies For Convergence Failure

- Penalizing constraint violation

  Penalty-barrier method:

  - (Fiacco, McCormick)
    (Forsgren, Gill)

$$\min \quad f(x) - \mu \sum_i \ln(x^{(i)}) + \frac{1}{2\mu}\|c(x)\|_2^2$$

  Splitting equalities:

  - (Tits, W. et al.)
    (Gould, Orban, Toint)

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) + \rho \sum_j c^{(j)}(x) \\ s.t. \quad & c(x) \geq 0 \\ & x \geq 0 \end{aligned}$$

Remedies For Convergence Failure

- Penalizing constraint violation

  Penalty-barrier method:

  - (Fiacco, McCormick)
    (Forsgren, Gill)

$$\min \quad f(x) - \mu \sum_i \ln(x^{(i)}) + \frac{1}{2\mu}\|c(x)\|_2^2$$

  Splitting equalities:

  - (Tits, W. et al.)
    (Gould, Orban, Toint)

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) + \rho \sum_j c^{(j)}(x) \\ s.t. \quad & c(x) \geq 0 \\ & x \geq 0 \end{aligned}$$

- Step decomposition in trust region framework
  - e.g., Byrd-Omojokun framework (such as in KNITRO)

Remedies For Convergence Failure

- Penalizing constraint violation

  Penalty-barrier method:

  - (Fiacco, McCormick)
    (Forsgren, Gill)

$$\min \quad f(x) - \mu \sum_i \ln(x^{(i)}) + \frac{1}{2\mu}\|c(x)\|_2^2$$

  Splitting equalities:

  - (Tits, W. et al.)
    (Gould, Orban, Toint)

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) + \rho \sum_j c^{(j)}(x) \\ s.t. \quad & c(x) \geq 0 \\ & x \geq 0 \end{aligned}$$

- Step decomposition in trust region framework
  - e.g., Byrd-Omojokun framework (such as in KNITRO)

- Filter methods
  - Restoration phase is called when α_k^x becomes very small
  - Steps in restoration phase aim to reduce infeasibility

Ipopt

An algorithm for large-scale nonlinear nonconvex optimization

- Framework: Barrier method (Fiacco-McCormick)
- Steps: Primal-dual interior point method

Ipopt

An algorithm for large-scale nonlinear nonconvex optimization

- Framework: Barrier method (Fiacco-McCormick)
- Steps: Primal-dual interior point method
- Globalization: Filter method for $(B(\mu))$
  - Restoration phase: Use IPOPT recursively to minimize infeasibility
    $\Longrightarrow$ Fast detection of infeasible problems

Ipopt

An algorithm for large-scale nonlinear nonconvex optimization

- Framework: Barrier method (Fiacco-McCormick)
- Steps: Primal-dual interior point method
- Globalization: Filter method for $(B(\mu))$
  - Restoration phase: Use IPOPT recursively to minimize infeasibility
    $\Longrightarrow$ Fast detection of infeasible problems
- Handling nonconvexities:

$$\begin{bmatrix} W_k + \Sigma_k + \delta I & \nabla c(x_k) \\ \nabla c(x_k)^T & 0 \end{bmatrix}$$

  Trial factorizations for increasing $\delta \geq 0$ until interia is correct

Ipopt

An algorithm for large-scale nonlinear nonconvex optimization

- Framework: Barrier method (Fiacco-McCormick)
- Steps: Primal-dual interior point method
- Globalization: Filter method for $(B(\mu))$
  - Restoration phase: Use IPOPT recursively to minimize infeasibility
    $\Longrightarrow$ Fast detection of infeasible problems
- Handling nonconvexities:

$$\begin{bmatrix} W_k + \Sigma_k + \delta I & \nabla c(x_k) \\ \nabla c(x_k)^T & 0 \end{bmatrix}$$

  Trial factorizations for increasing $\delta \geq 0$ until interia is correct
- Theoretical convergence properties:
  - Global convergence: From filter method for each $(B(\mu))$
  - Local convergence: If μ_k is updated properly, can achieve superlinear convergence (Gould, Orban, Sartenaer, Toint, 2001))

COIN-OR *http://www.coin-or.org*

- COmputational INfrastructure for Operations Research
- Open source repository for optimization and related software
- Nonprofit foundation (hosted at INFORMS)
- Numerous projects:
  - `Bcp` (parallel framework for Branch, Cut and Pricing algorithms)
  - `Cbc` (COIN Branch and Cut algorithm for MILP)
  - `Cgl` (Cut Generation Library)
  - `Clp` (Coin LP Solver)
  - `Osi` (Open Solver Interface) [`Cplex`, `OSL`, `Xpress MP`, `Clp`, `Glpk`]
  - `Ipopt` (Interior Point NLP solver)
  - `DFO`, `NLP-API`, `GAMSlinks`, `SMI`, `SVM-QP`, `CSPD`, `CppAD`, `...`
- Also models/data sets (for benchmarking, allow recreating results from papers), frameworks
- Contribution are WELCOME!

Implementation Of Ipopt

- Code IPOPT available as open source on COIN-OR
- Written in C++ (replaces earlier Fortran version) (with Carl Laird, CMU)
- Interfaced with AMPL, GAMS, CUTEr, (Matlab?); available on NEOS
- Has been used for
  - Optimization of differential-algebraic systems
  - Nonlinear model predictive control
  - Air traffic resolution (MPECs, (Raghunathan, 2004)) ...
- `coin-ipopt` mailing list with 184 subscribers
- Recent developments
  - Adaptive choice of the barrier parameter (Nocedal, W, Waltz, 2005)
  - Interfaces to a number of linear solvers:
    - Harwell, WSMP, MUMPS, Pardiso (Schenk, W, Hagemann, 2005)
    - This is a crutial part in terms of efficiency and robustness
  - Limited memory quasi-Newton approximation of 2nd derivatives

Adaptive Barrier Parameter

Barrier Methods for NLP

$$\begin{array}{ll} \min & f(x) \\ s.t. & c(x) = 0 \\ & x \geq 0 \end{array} \quad \longrightarrow \quad \begin{array}{ll} \min & f(x) - \mu \sum \ln(x^{(i)}) \\ s.t. & c(x) = 0 \end{array} \qquad \mathrm{BP}(\mu)$$

- Fiacco-McCormick approach:
  Solve barrier problem $\mathrm{BP}(\mu_l)$ (approximately) for monotone sequence of barrier parameters $\{\mu_l\} \to 0$
- Many different variants
  (primal-dual, different handling of constraints)

Barrier Methods for NLP

$$\begin{array}{ll} \min & f(x) \\ s.t. & c(x) = 0 \\ & x \geq 0 \end{array} \quad \longrightarrow \quad \begin{array}{ll} \min & f(x) - \mu \sum \ln(x^{(i)}) \\ s.t. & c(x) = 0 \end{array} \qquad \mathrm{BP}(\mu)$$

- Fiacco-McCormick approach:
  Solve barrier problem $\mathrm{BP}(\mu_l)$ (approximately) for monotone sequence of barrier parameters $\{\mu_l\} \to 0$
- Many different variants
  (primal-dual, different handling of constraints)
- Global convergence: Ensure global convergence for each $\mathrm{BP}(\mu_l)$

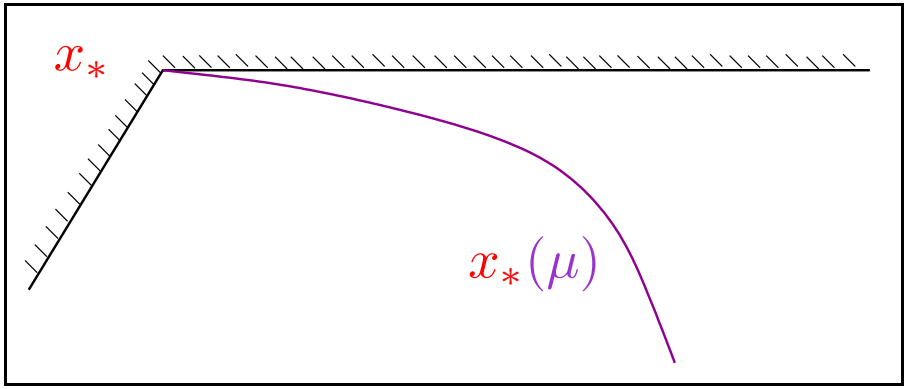
Barrier Methods for NLP

$$\begin{array}{ll} \min & f(x) \\ s.t. & c(x) = 0 \\ & x \geq 0 \end{array} \quad \longrightarrow \quad \begin{array}{ll} \min & f(x) - \mu \sum \ln(x^{(i)}) \\ s.t. & c(x) = 0 \end{array} \qquad \mathrm{BP}(\mu)$$

- Fiacco-McCormick approach:
  Solve barrier problem $\mathrm{BP}(\mu_l)$ (approximately) for monotone sequence of barrier parameters $\{\mu_l\} \to 0$
- Many different variants
  (primal-dual, different handling of constraints)
- Global convergence: Ensure global convergence for each $\mathrm{BP}(\mu_l)$
- Possible problems:
  - For given large μ, problem might be unbounded or difficult
  - Difficult to find good starting value μ_0
  - Fixing μ for several iterations might not be most efficient option
    (don't really want to solve $\mathrm{BP}(\mu_l)$)

Solving Primal-Dual Equations

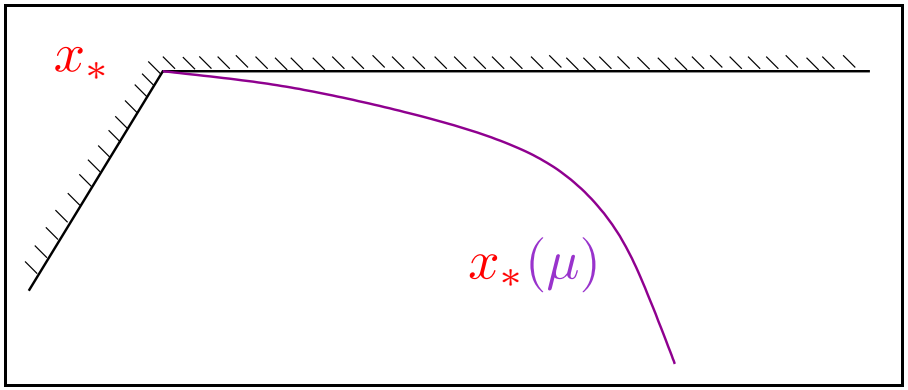
$$\begin{aligned} \nabla f(x) - \nabla c(x)\lambda - z &= 0 \\ c(x) &= 0 \\ XZe &= \mu e \\ x, z &\geq 0 \end{aligned}$$



- Concentrate on solving the primal-dual equations (with $\mu = 0$)
- Follow the central path $x_*(\mu)$ as $\mu \to 0$
- Methods adapt μ non-monotonically in every iteration
- Very effective for LP, QP, convex programming

Solving Primal-Dual Equations

$$\begin{aligned} \nabla f(x) - \nabla c(x)\lambda - z &= 0 \\ c(x) &= 0 \\ XZe &= \mu e \\ x, z &\geq 0 \end{aligned}$$



- Concentrate on solving the primal-dual equations (with $\mu = 0$)
- Follow the central path $x_*(\mu)$ as $\mu \to 0$
- Methods adapt μ non-monotonically in every iteration
- Very effective for LP, QP, convex programming
- Global convergence issues:
  - Solving PD equations good enough for convex problems
  - In nonconvex case: Ignoring $f(x)$ can lead more easily to convergence to saddle points or maxima

Our Goal

Want to combine best of both approaches

1. Globalization method for $\mathrm{BP}(\mu)$, e.g.
   - KNITRO (Byrd, Nocedal, Waltz, et al.)
     - Trust region, merit function
   - IPOPT (W, Biegler)
     - Line search, filter method
   - Your favorite method

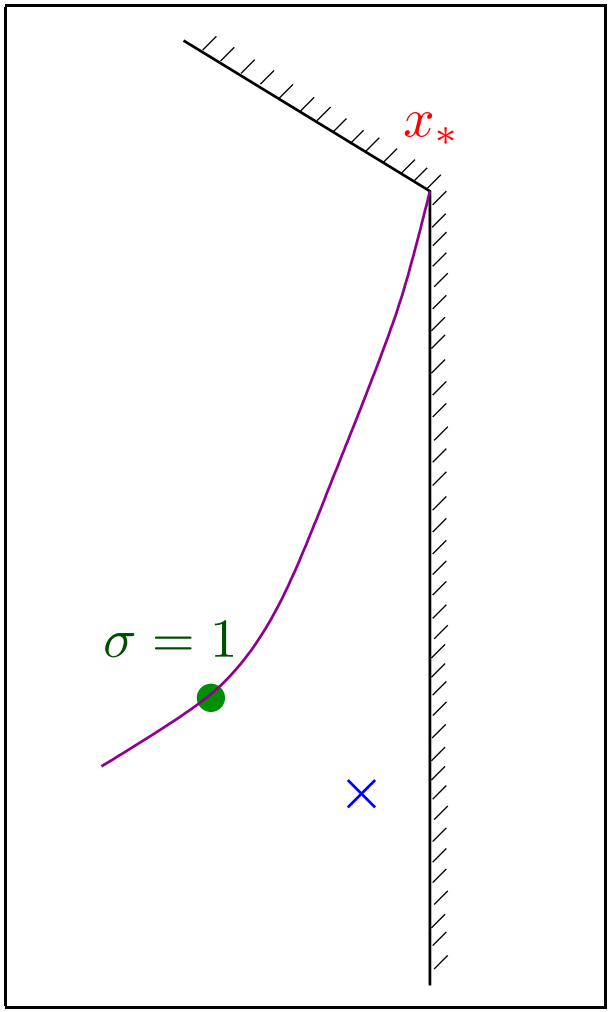
2. Effective adaptive update rules for μ

in an efficient, globally convergent framework

Choosing Adaptive Barrier Parameter

$$\mu = \sigma \frac{x^T z}{n}$$

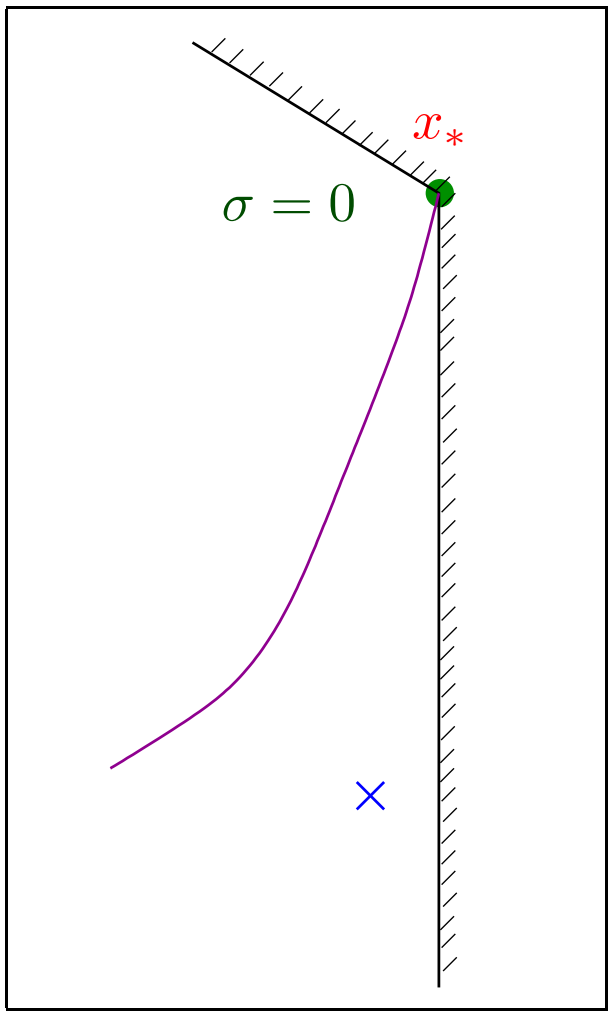
with centering parameter σ



Choosing Adaptive Barrier Parameter

$$\mu = \sigma \frac{x^T z}{n}$$

with centering parameter σ



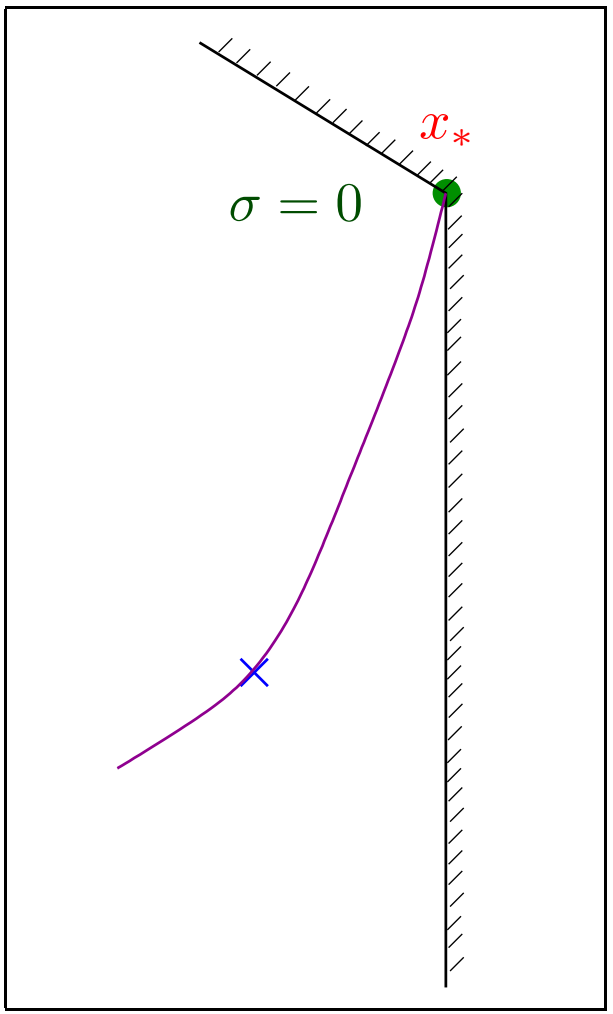
Choosing Adaptive Barrier Parameter

$$\mu = \sigma \frac{x^T z}{n}$$

with centering parameter σ

Methods previously proposed:

1. LOQO centrality rule (formula)
   (Vanderbei, Shanno, 1999)
   - iterate central ($x^{(i)} z^{(i)} \equiv \frac{x^T z}{n}$) $\Longrightarrow \sigma = 0$



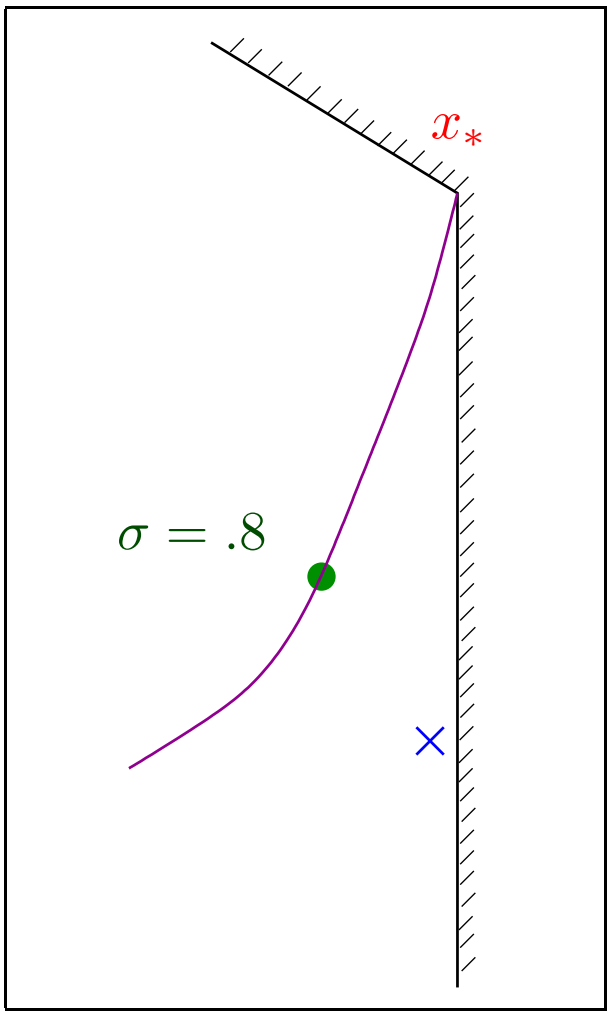
Choosing Adaptive Barrier Parameter

$$\mu = \sigma \frac{x^T z}{n}$$

with centering parameter σ

Methods previously proposed:

1. LOQO centrality rule (formula)
   (Vanderbei, Shanno, 1999)
   - iterate central ($x^{(i)} z^{(i)} \equiv \frac{x^T z}{n}$) $\Longrightarrow \sigma = 0$
   - iterate not central ($x^{(\tilde{i})} z^{(\tilde{i})} \ll \frac{x^T z}{n}$) $\Longrightarrow \sigma = .8$



Choosing Adaptive Barrier Parameter

$$\mu = \sigma \frac{x^T z}{n}$$

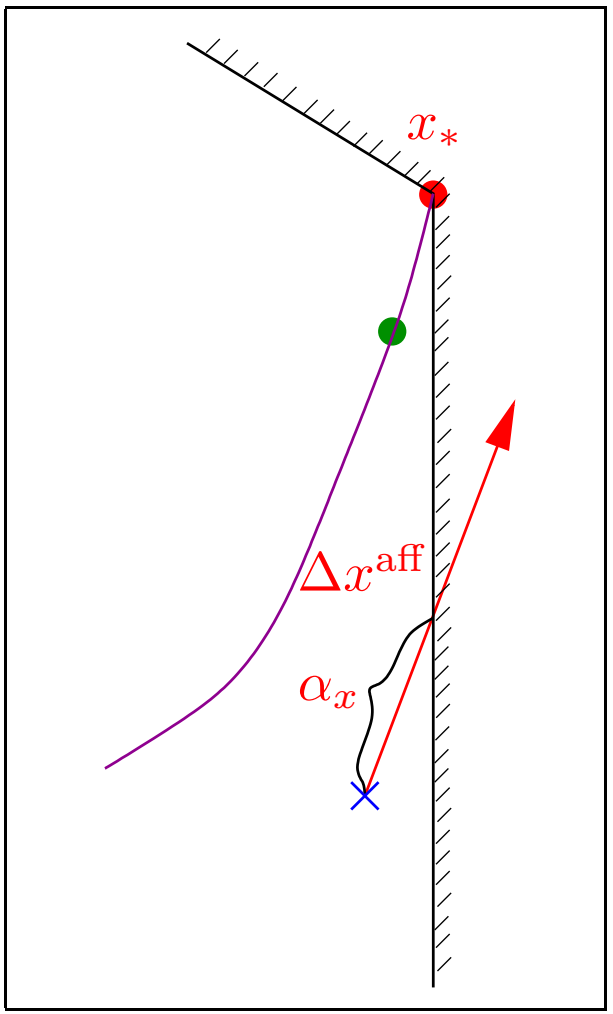
with centering parameter σ

Methods previously proposed:

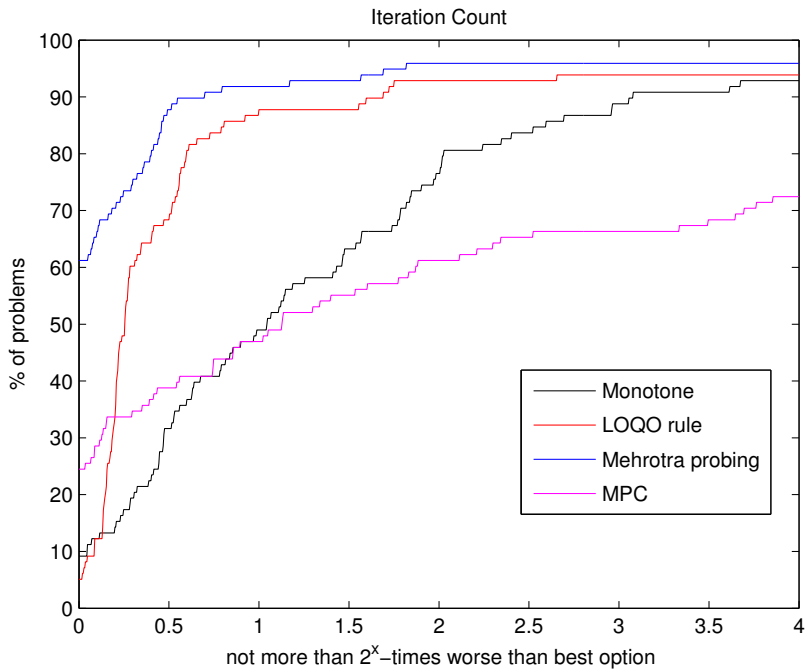
1. LOQO centrality rule (formula)
   (Vanderbei, Shanno, 1999)
   - iterate central ($x^{(i)} z^{(i)} \equiv \frac{x^T z}{n}$) $\Longrightarrow \sigma = 0$
   - iterate not central ($x^{(\tilde{i})} z^{(\tilde{i})} \ll \frac{x^T z}{n}$) $\Longrightarrow \sigma = .8$
2. Mehrotra's probing rule for LP
   (Mehrotra, 1992)

$$\sigma = \left(\frac{(x + \alpha^x \Delta x^{\text{aff}})^T (z + \alpha^z \Delta z^{\text{aff}})}{x^T z} \right)^3$$

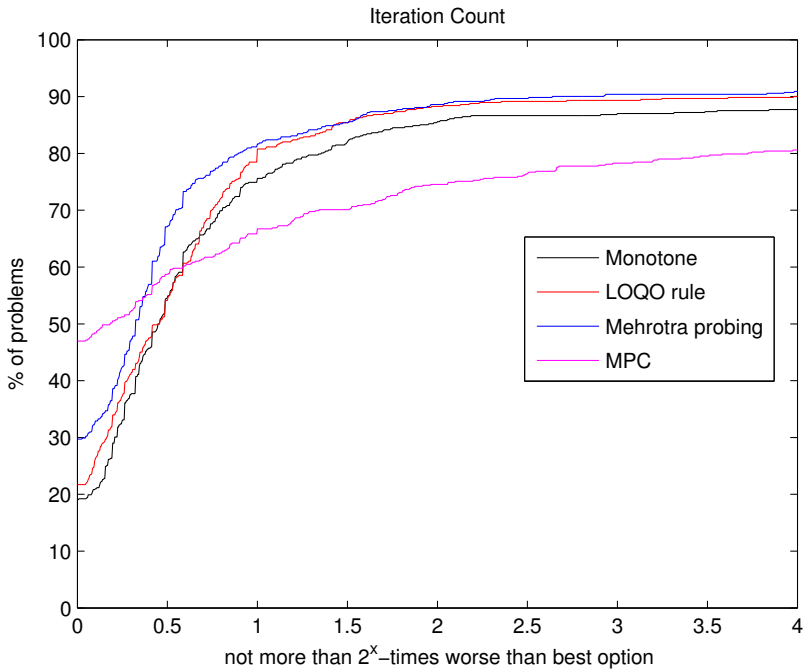
(requires extra step computation)



Adaptive Strategies (Ipopt)



Netlib

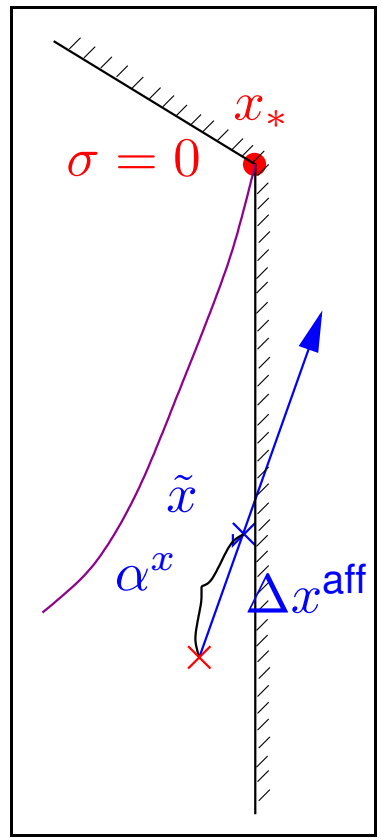


CUTEr

No line search (always fraction-to-the-boundary step size)

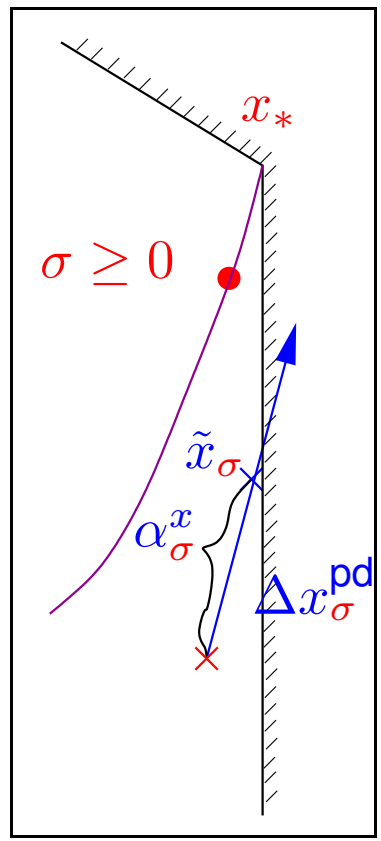
Can We Improve The Probing Strategy?

$$\begin{bmatrix} W & A(x) & -I \\ A(x)^T & 0 & 0 \\ Z & 0 & X \end{bmatrix} \begin{pmatrix} \Delta x^{\text{aff}} \\ \Delta \lambda^{\text{aff}} \\ \Delta z^{\text{aff}} \end{pmatrix} = - \begin{pmatrix} \nabla f(x) + A(x)\lambda - z \\ c(x) \\ XZe \end{pmatrix}$$



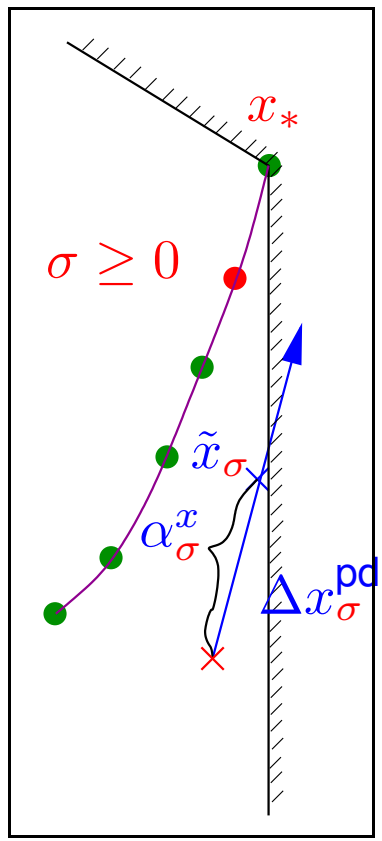
Can We Improve The Probing Strategy?

$$\begin{bmatrix} W & A(x) & -I \\ A(x)^T & 0 & 0 \\ Z & 0 & X \end{bmatrix} \begin{pmatrix} \Delta x_\sigma^{\text{pd}} \\ \Delta \lambda_\sigma^{\text{pd}} \\ \Delta z_\sigma^{\text{pd}} \end{pmatrix} = - \begin{pmatrix} \nabla f(x) + A(x)\lambda - z \\ c(x) \\ XZe - \sigma \frac{x^T z}{n} e \end{pmatrix}$$



Can We Improve The Probing Strategy?

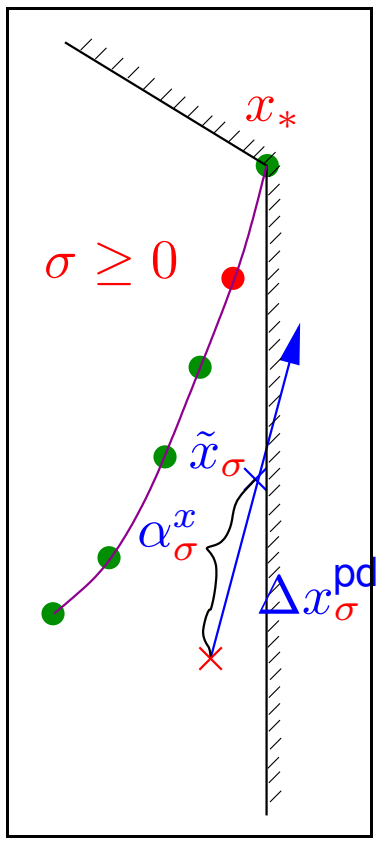
$$\begin{bmatrix} W & A(x) & -I \\ A(x)^T & 0 & 0 \\ Z & 0 & X \end{bmatrix} \begin{pmatrix} \Delta x_\sigma^{\text{pd}} \\ \Delta \lambda_\sigma^{\text{pd}} \\ \Delta z_\sigma^{\text{pd}} \end{pmatrix} = - \begin{pmatrix} \nabla f(x) + A(x)\lambda - z \\ c(x) \\ XZe - \sigma \frac{x^T z}{n} e \end{pmatrix}$$



Can We Improve The Probing Strategy?

$$\begin{bmatrix} W & A(x) & -I \\ A(x)^T & 0 & 0 \\ Z & 0 & X \end{bmatrix} \begin{pmatrix} \Delta x_\sigma^{\text{pd}} \\ \Delta \lambda_\sigma^{\text{pd}} \\ \Delta z_\sigma^{\text{pd}} \end{pmatrix} = - \begin{pmatrix} \nabla f(x) + A(x)\lambda - z \\ c(x) \\ XZe \end{pmatrix} + \sigma \begin{pmatrix} 0 \\ 0 \\ \frac{x^T z}{n} e \end{pmatrix}$$

- Compute Δ^{aff} and Δ^{cen} in two linear solves
- Then, $\Delta_\sigma^{\text{pd}} = \Delta^{\text{aff}} + \sigma \Delta^{\text{cen}}$ cheaply available for all $\sigma \geq 0$



Can We Improve The Probing Strategy?

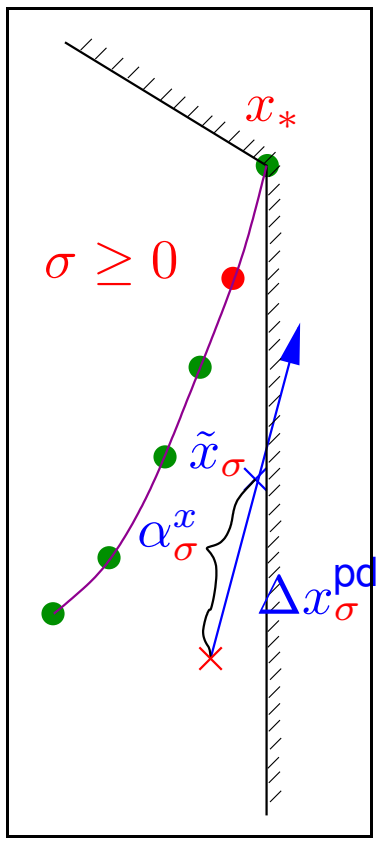
$$\begin{bmatrix} W & A(x) & -I \\ A(x)^T & 0 & 0 \\ Z & 0 & X \end{bmatrix} \begin{pmatrix} \Delta x_\sigma^{\text{pd}} \\ \Delta \lambda_\sigma^{\text{pd}} \\ \Delta z_\sigma^{\text{pd}} \end{pmatrix} = - \begin{pmatrix} \nabla f(x) + A(x)\lambda - z \\ c(x) \\ XZe \end{pmatrix} + \sigma \begin{pmatrix} 0 \\ 0 \\ \frac{x^T z}{n} e \end{pmatrix}$$

- Compute Δ^{aff} and Δ^{cen} in two linear solves
- Then, $\Delta_\sigma^{\text{pd}} = \Delta^{\text{aff}} + \sigma \Delta^{\text{cen}}$ cheaply available for all $\sigma \geq 0$
- Given $\sigma \geq 0$, examine probing points

$$\begin{aligned} \tilde{x}_\sigma &= x + \alpha_\sigma^x \Delta x_\sigma^{\text{pd}} \\ \tilde{\lambda}_\sigma &= \lambda + \alpha_\sigma^z \Delta \lambda_\sigma^{\text{pd}} \\ \tilde{z}_\sigma &= z + \alpha_\sigma^z \Delta z_\sigma^{\text{pd}} \end{aligned}$$

where

$$\begin{aligned} \alpha_\sigma^x &= \max\{\alpha \in (0,1] : x + \alpha \Delta x_\sigma^{\text{pd}} \geq (1-\tau)x\} \\ \alpha_\sigma^z &= \max\{\alpha \in (0,1] : z + \alpha \Delta z_\sigma^{\text{pd}} \geq (1-\tau)z\} \end{aligned}$$



Quality Function For Centering Parameter

- Search for the "best" σ that minimizes some "quality function" $\Phi(\sigma)$
  E.g.,

$$\Phi(\sigma) = \left\| \begin{pmatrix} \nabla f(\tilde{x}_\sigma) + A(\tilde{x}_\sigma)\tilde{\lambda}_\sigma - \tilde{z}_\sigma \\ c(\tilde{x}_\sigma) \\ \tilde{X}_\sigma \tilde{Z}_\sigma e \end{pmatrix} \right\|_2^2$$

Quality Function For Centering Parameter

- Search for the "best" σ that minimizes some "quality function" $\Phi(\sigma)$
  E.g.,

$$\Phi(\sigma) = \left\| \begin{pmatrix} \nabla f(\tilde{x}_\sigma) + A(\tilde{x}_\sigma)\tilde{\lambda}_\sigma - \tilde{z}_\sigma \\ c(\tilde{x}_\sigma) \\ \tilde{X}_\sigma \tilde{Z}_\sigma e \end{pmatrix} \right\|_2^2$$

- Do not want to evaluate functions and gradients

Quality Function For Centering Parameter

- Search for the "best" σ that minimizes some "quality function" $\Phi(\sigma)$
  E.g.,

$$\Phi(\sigma) = \left\| \begin{pmatrix} \nabla f(\tilde{x}_\sigma) + A(\tilde{x}_\sigma)\tilde{\lambda}_\sigma - \tilde{z}_\sigma \\ c(\tilde{x}_\sigma) \\ \tilde{X}_\sigma \tilde{Z}_\sigma e \end{pmatrix} \right\|_2^2$$

- Do not want to evaluate functions and gradients
- "Linear quality function"

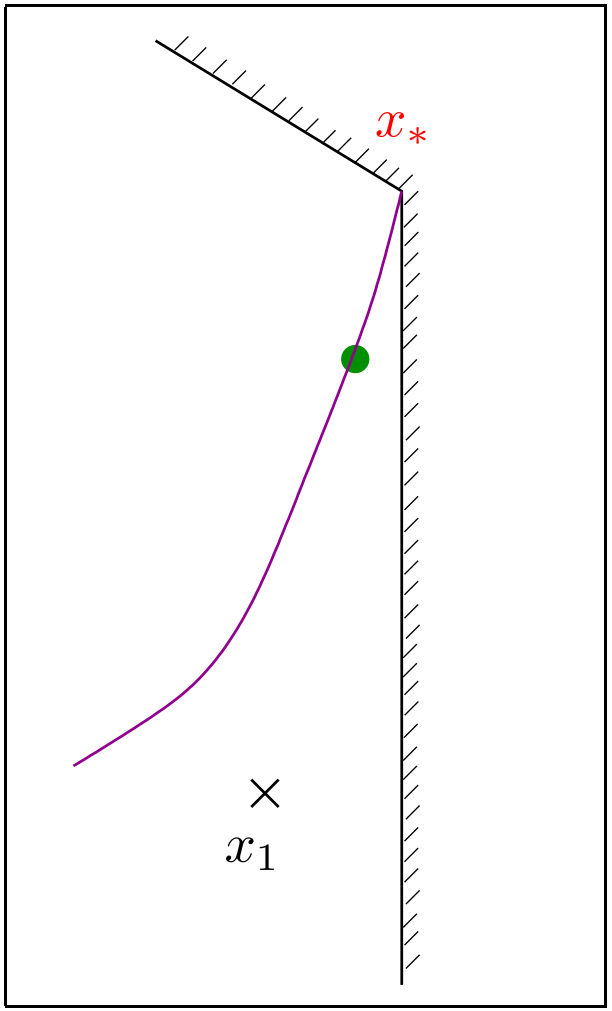
$$\begin{aligned} \Phi(\sigma) \;=\; & (1-\alpha_\sigma^z)^2 \|\nabla f(x_k) + A(x_k)\lambda_k - z_k\|_2^2 + \\ & (1-\alpha_\sigma^x)^2 \|c(x_k)\|_2^2 + \\ & \|\tilde{X}_\sigma \tilde{Z}_\sigma e\|_2^2 \end{aligned}$$

  - Accurately predicts primal-dual error at probing point for LPs
  - Relatively inexpensive to compute (need α_σ^x, α_σ^z, $\tilde{X}_\sigma \tilde{Z}_\sigma e$ for given σ)
  - $\Phi(\sigma)$ is non-smooth, non-convex; use direct search

A General Globalization Framework

For each iteration k:

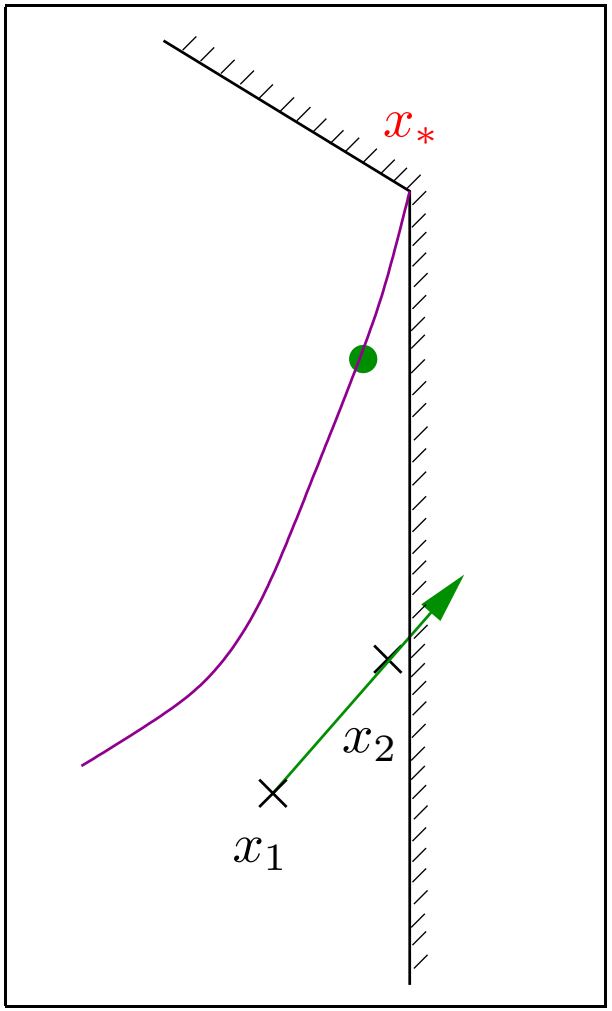
1. Choose μ_k by adaptive rule (*free mode*)



A General Globalization Framework

For each iteration k:

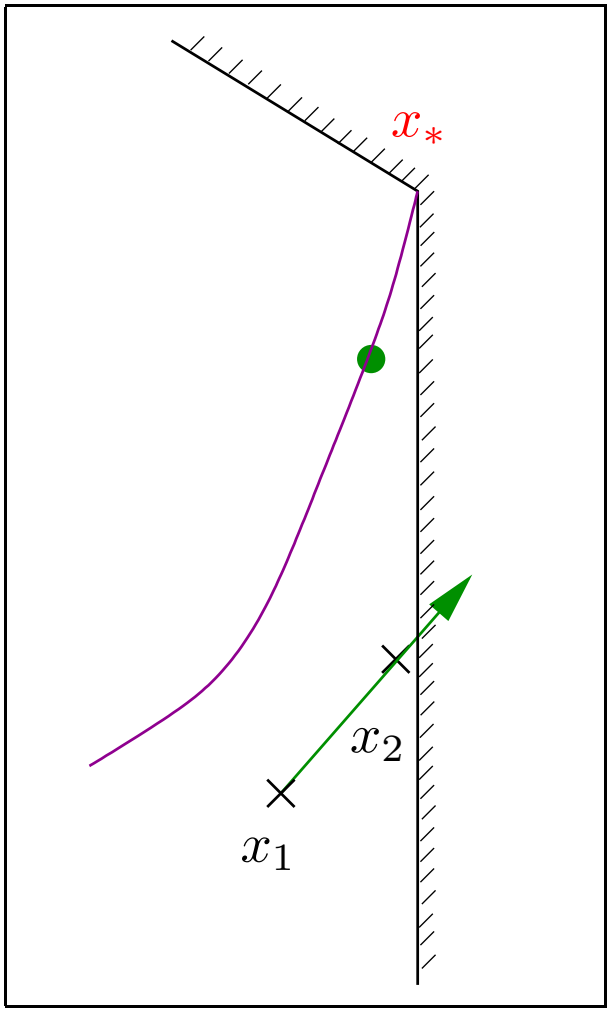
1. Choose μ_k by adaptive rule (*free mode*)
2. Compute step Δ^{pd} for $\mathrm{BP}(\mu_k)$, and x_{k+1} (ensure progress for $\mathrm{BP}(\mu_k)$ w.r.t. x_k)



A General Globalization Framework

For each iteration k:

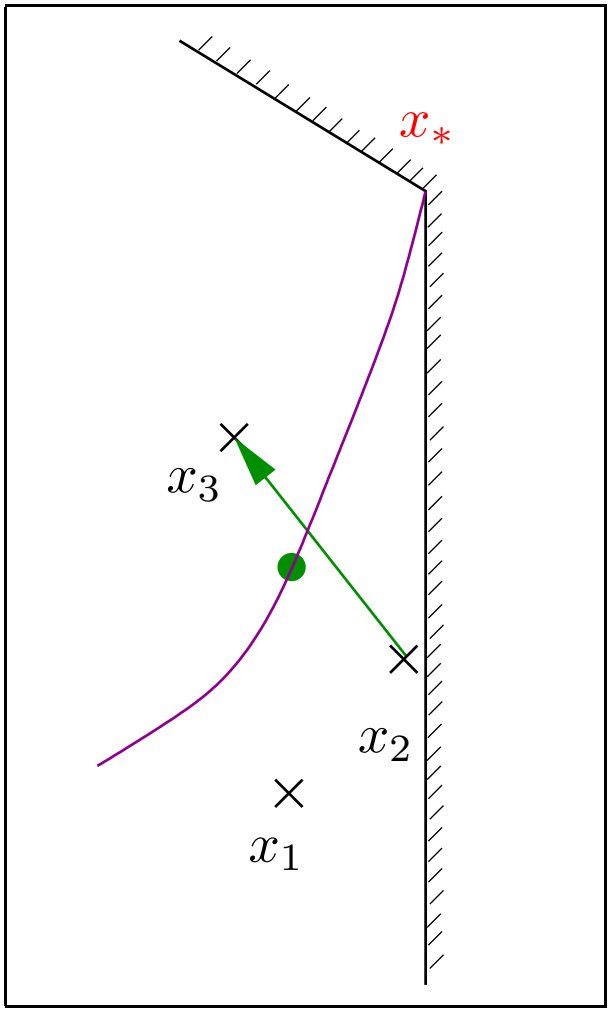
1. Choose μ_k by adaptive rule (*free mode*)
2. Compute step Δ^{pd} for $\mathrm{BP}(\mu_k)$, and x_{k+1} (ensure progress for $\mathrm{BP}(\mu_k)$ w.r.t. x_k)
3. Progress for original NLP? (suff. progress in $\|\mathsf{PD_error}_k\|_2^2$?)



A General Globalization Framework

For each iteration k:

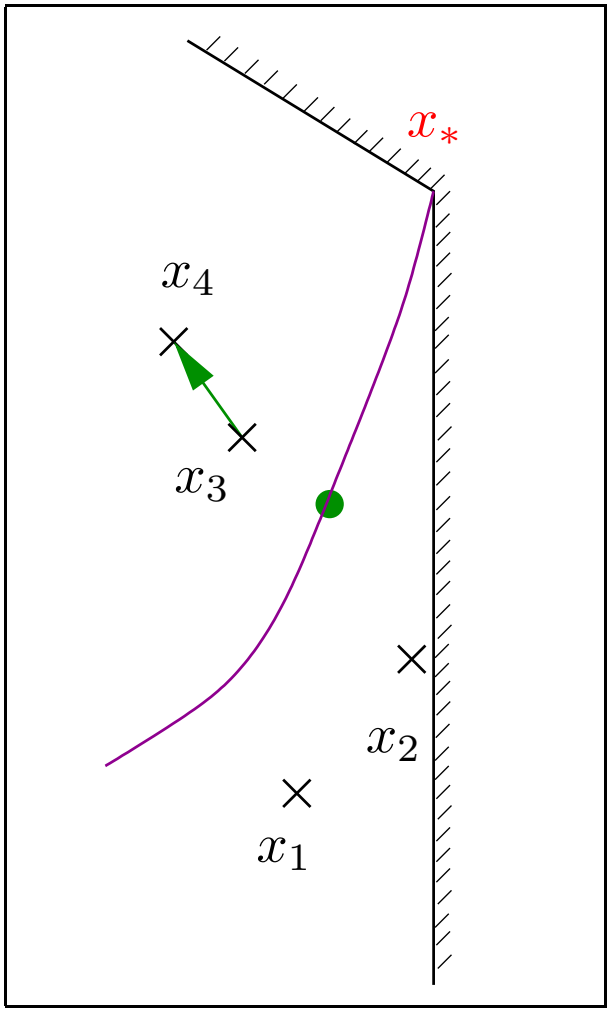
1. Choose μ_k by adaptive rule (*free mode*)
2. Compute step Δ^{pd} for $\mathrm{BP}(\mu_k)$, and x_{k+1} (ensure progress for $\mathrm{BP}(\mu_k)$ w.r.t. x_k)
3. Progress for original NLP? (suff. progress in $\|\mathsf{PD_error}_k\|_2^2$?)
   - Yes: Continue with next iteration



A General Globalization Framework

For each iteration k:

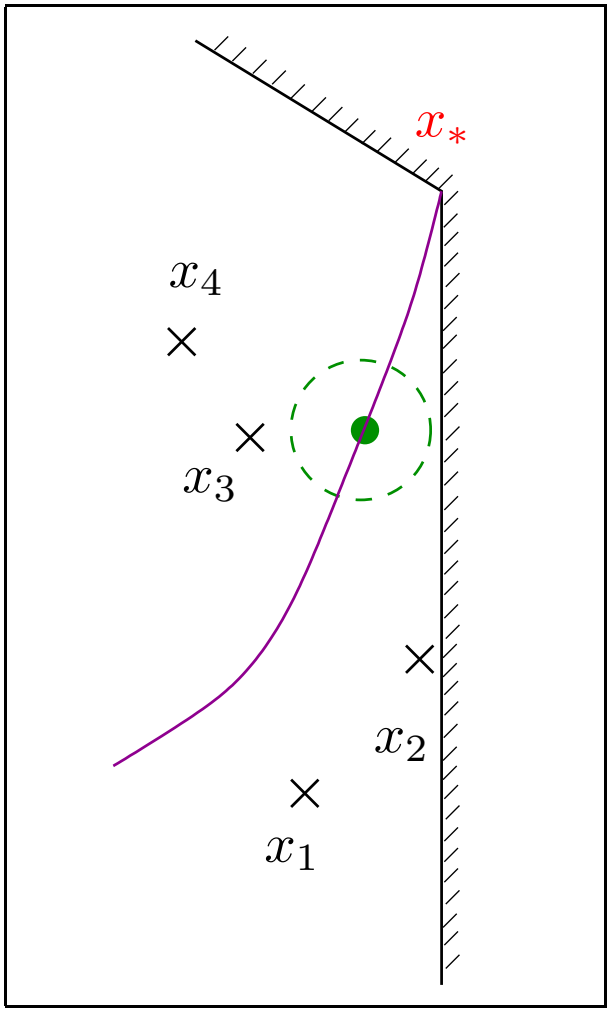
1. Choose μ_k by adaptive rule (*free mode*)
2. Compute step Δ^{pd} for $\mathrm{BP}(\mu_k)$, and x_{k+1} (ensure progress for $\mathrm{BP}(\mu_k)$ w.r.t. x_k)
3. Progress for original NLP? (suff. progress in $\|\mathsf{PD_error}_k\|_2^2$?)
   - Yes: Continue with next iteration
   - No:



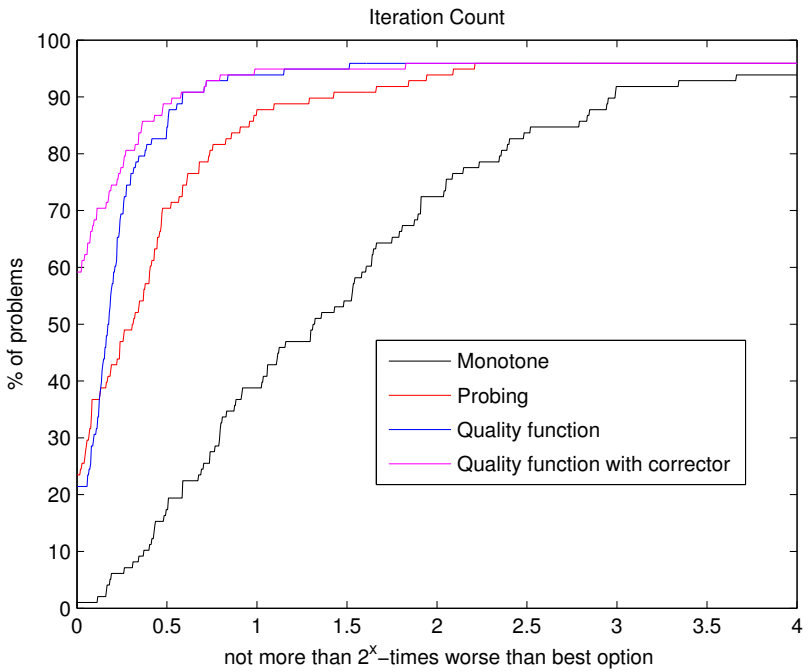
A General Globalization Framework

For each iteration k:

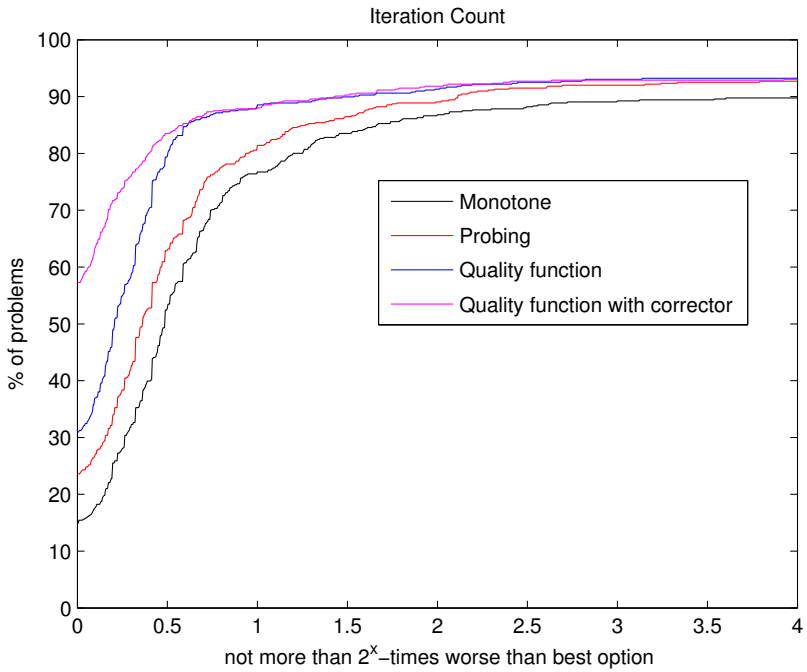
1. Choose μ_k by adaptive rule (*free mode*)
2. Compute step Δ^{pd} for $\mathrm{BP}(\mu_k)$, and x_{k+1} (ensure progress for $\mathrm{BP}(\mu_k)$ w.r.t. x_k)
3. Progress for original NLP? (suff. progress in $\|\mathsf{PD_error}_k\|_2^2$?)
   - Yes: Continue with next iteration
   - No: (*monotone mode*)
     (a) execute robust, monotone Fiacco-McCormick algorithm (approx. solve $\mathrm{BP}(\bar{\mu})$ problems)
     (b) if suff. progress for original NLP, return to *free mode*



Globalization Strategy (Ipopt)



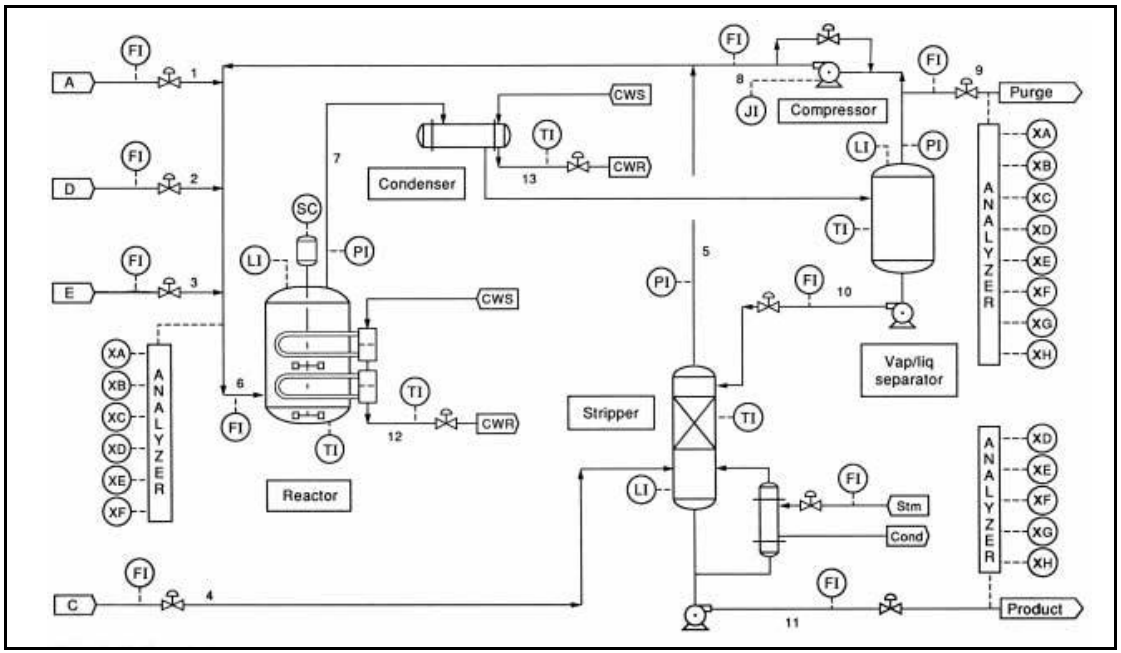
Netlib



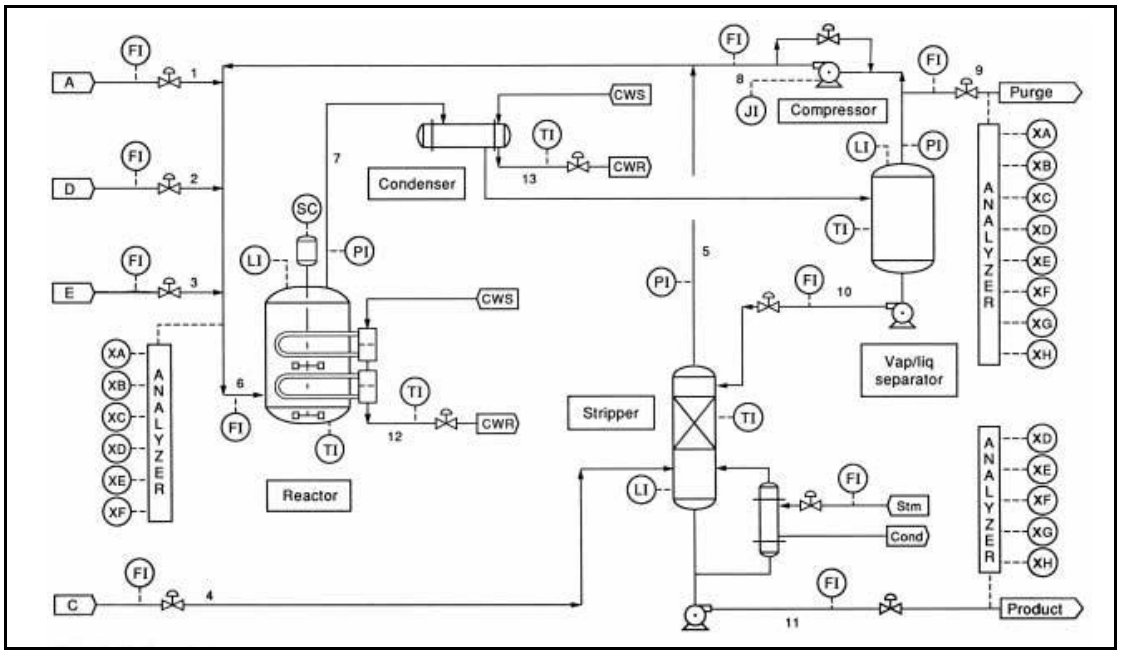
CUTEr

Nonlinear Optimization Applications

Design and Operation of Chemical Plant

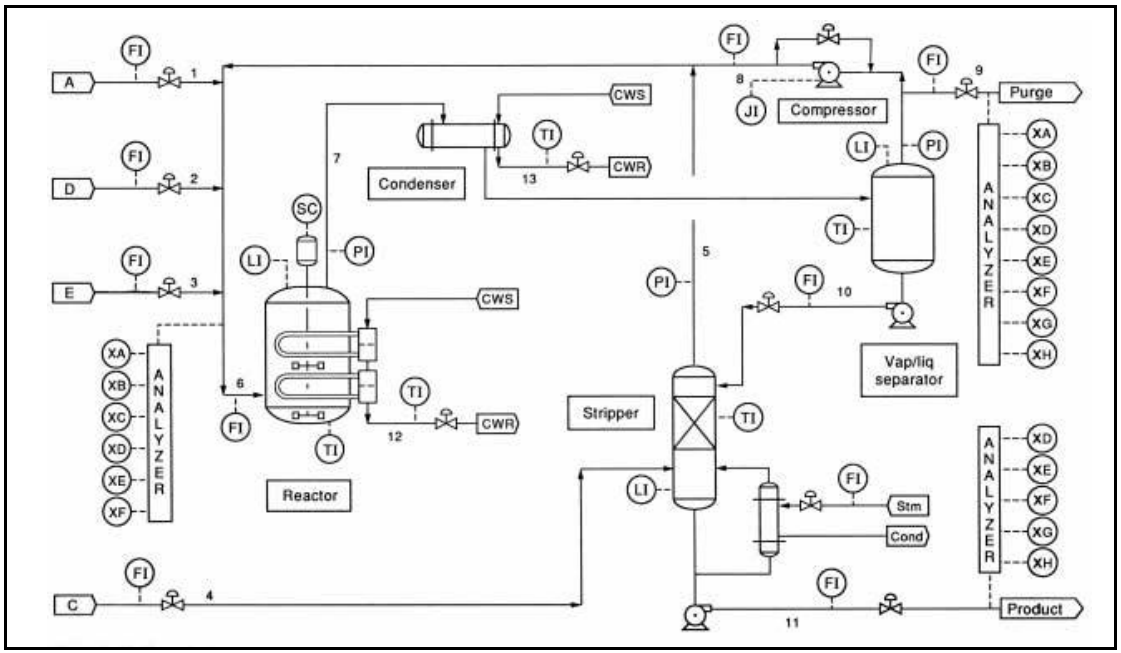


Design and Operation of Chemical Plant



- Minimize "Costs $-$ Profit"
- Variables: Physical quantities
- Constraints: physical relationships
  (conservation laws; themodyn. rel.)
- Limits (physical and operational)
- $< 10^5$ variables; few degrees of freedom

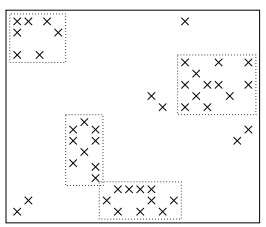
Design and Operation of Chemical Plant



- Minimize "Costs $-$ Profit"
- Variables: Physical quantities
- Constraints: physical relationships
  (conservation laws; themodyn. rel.)
- Limits (physical and operational)
- $< 10^5$ variables; few degrees of freedom

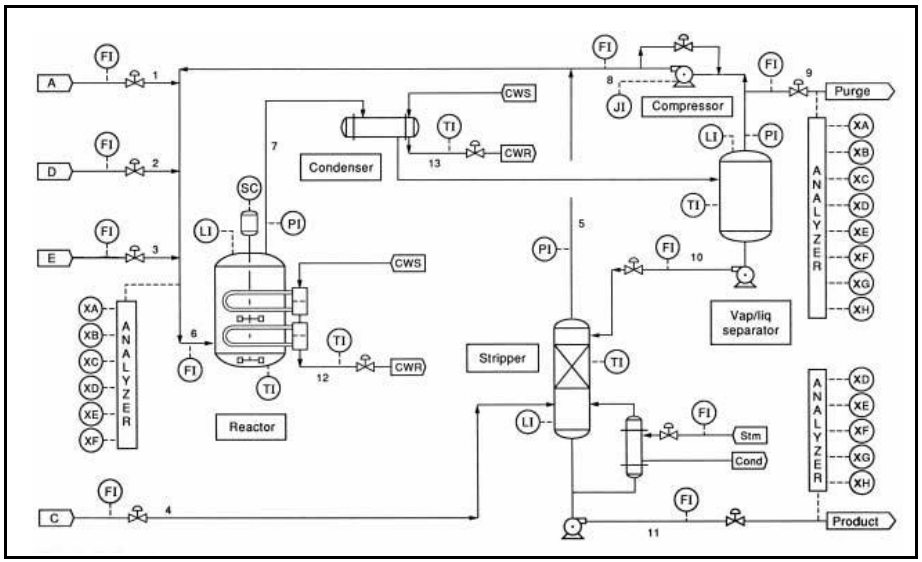
Constraint Jacobian

$$\nabla c(x)^T =$$

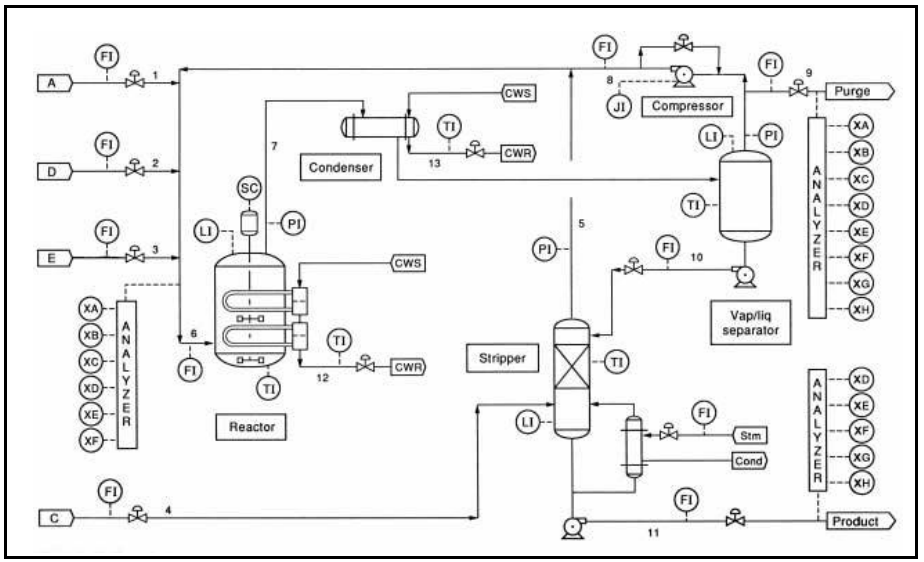


is sparse and structured

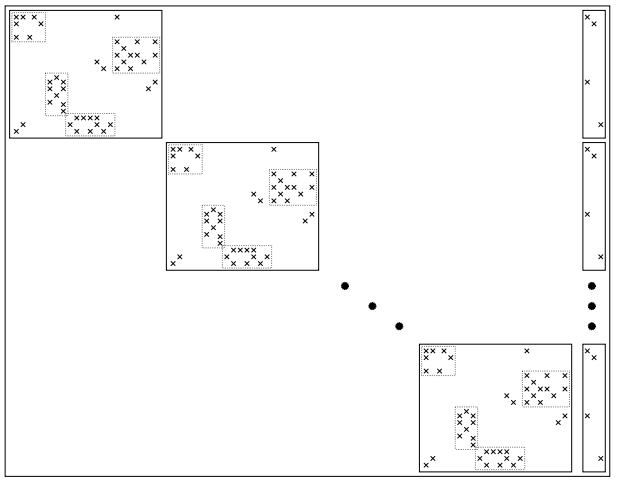
Design Under Uncertainty



Design Under Uncertainty

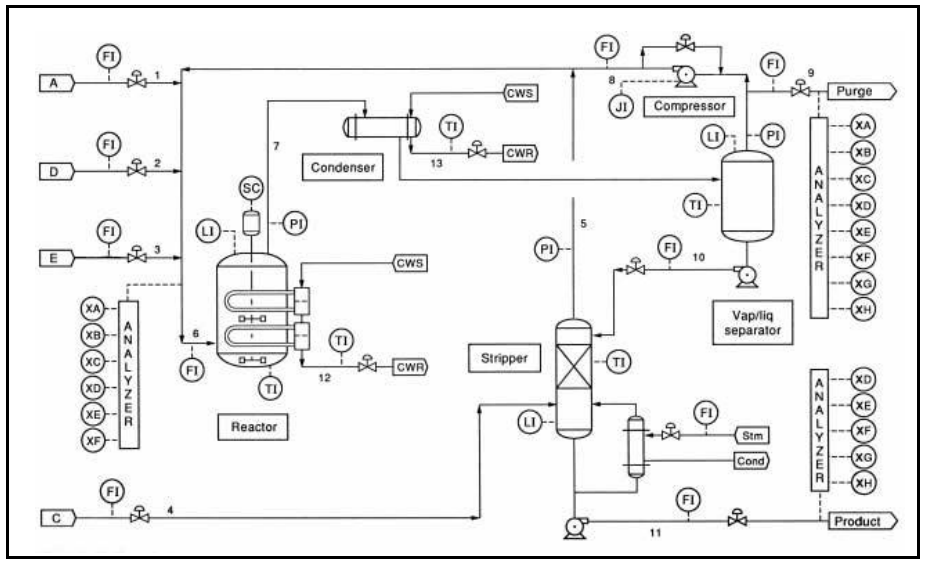


$$\nabla c(x)^T =$$



- Scenario parameters: $F^l_{in}, x^l_{in}, T^l_{env}, p^l_{env}, \ldots$ (given)
  (defining scenarios $l = 1, \ldots, L$)
- Design variables: $V_{react}, D_{dist}, h_{tank}, \ldots$
- Control variables: $Q^l_{heat}, r^l_{refl}, v^l_{valve}, \ldots$
- State variables: $x^l_{str}, F^l_{str}, L^l_{tank}, T^l, p^l, \ldots$

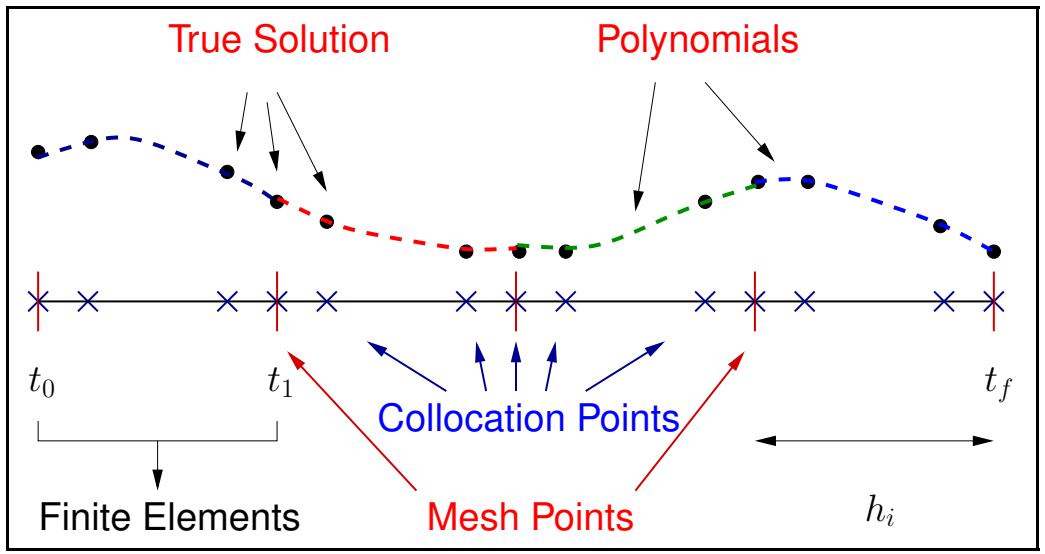
Optimal Control / Dynamic Optimization

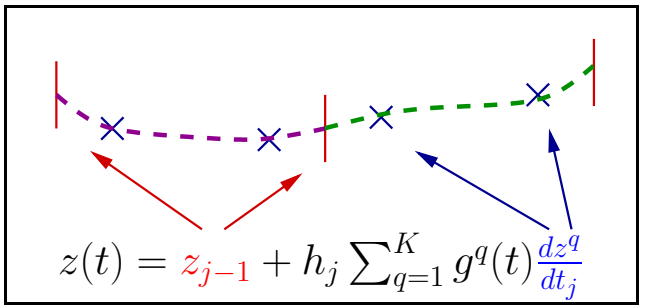


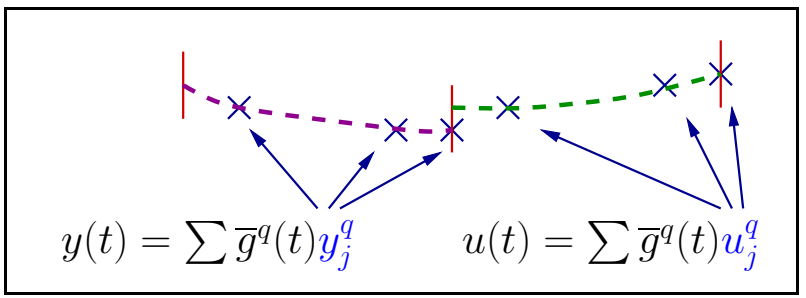
$$\begin{aligned} \min_{z,y,u,p} \quad & f(z(t_f), y(t_f), u(t_f), p) \\ s.t. \quad & F(\dot{z}(t), z(t), y(t), u(t), p) = 0 \\ & G(z(t), y(t), u(t), p) = 0 \\ & z(0) = z_{init} \\ & \textit{bound constraints} \end{aligned}$$

$u : [0, t_f] \rightarrow \mathbb{R}^{n_u}$	control variables
$z : [0, t_f] \rightarrow \mathbb{R}^{n_z}$	differentiable state variables
$y : [0, t_f] \rightarrow \mathbb{R}^{n_y}$	algebraic state variables
$p \in \mathbb{R}^{n_p}$	time-independent parameters
z_{init}	initial conditions
t_f	final time

Discretization of DAEs



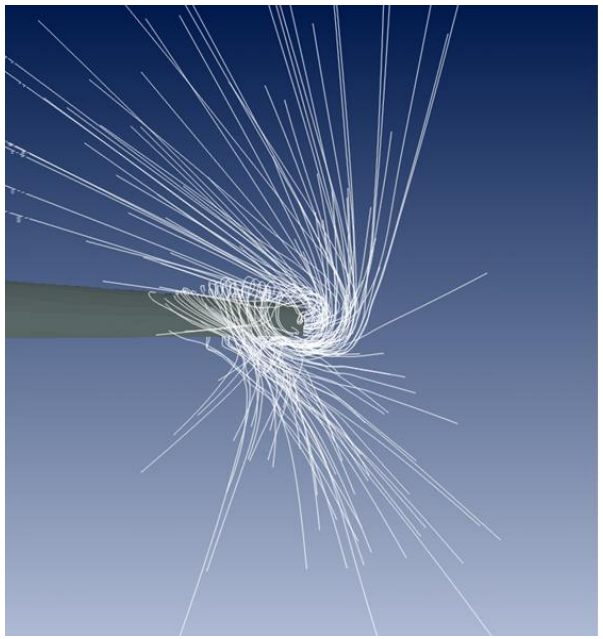




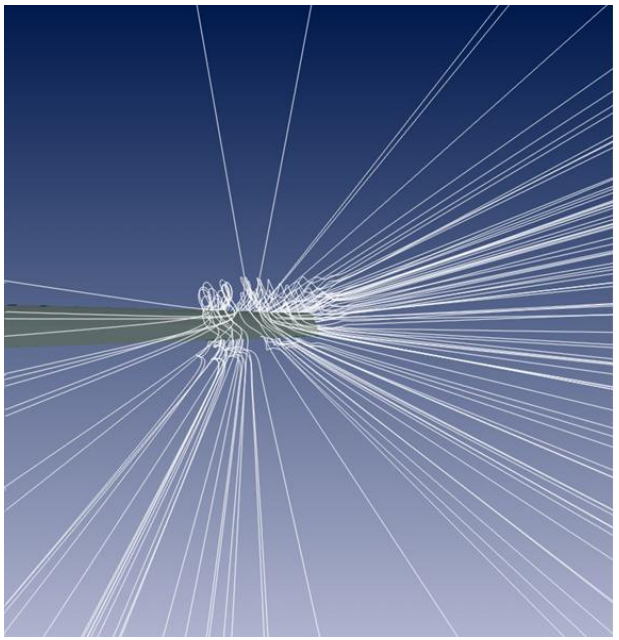
- x: Coefficients in polynomial representation
- c: DAE at collocation points / continuity equations
- Many variables $n = 1,000 - 10,000,000$
- Relatively few degrees of freedom ($n - m = 10 - 10,000$)
- Jacobian sparse and structured (almost block-diagonal)
- Finer discretization gives better approximation

Problems with PDEs

Uncontrolled



Controlled



- Constraints: discretized PDEs
- Very large problems
- Iterative linear solvers (preconditioners)
- Require parallel computing

- Applications:
  - Design (plains, cars)
  - Inverse Problems (geological data)
  - Hypothermia cancer treatment

Computationally Challenging Problems

- Large-Scale NLPs

Analytic functions

- e.g. discretized DAEs/PDEs
- 1st derivatives available
- 2nd deriv. available or approx.

Many variables (n up to $10,000,000$)

Exploit structure

- Sparse, structured $\nabla c(x)^T$
- PDE preconditioners
- Few vs. many degrees of freedom

The Other Extreme

- Computation of functions very expensive
- Function evaluation might include
  - Simulation of complicated system
  - Laboratory experiment (car crash test)
- Functions might be “noisy”
- Computation of gradients impossible or difficult
- Applications
  - Design of cars/planes (helicopter blades; jet engines)
  - Energy network optimization
- Solution methods:
  - Surrogate models (derivative-free optimization)
  - Pattern search (PDS, Nelder-Meade)
- Number of variables usually very small (n up few 100)
  (the relevant quantities determined by statisticians)

Computationally Challenging Problems

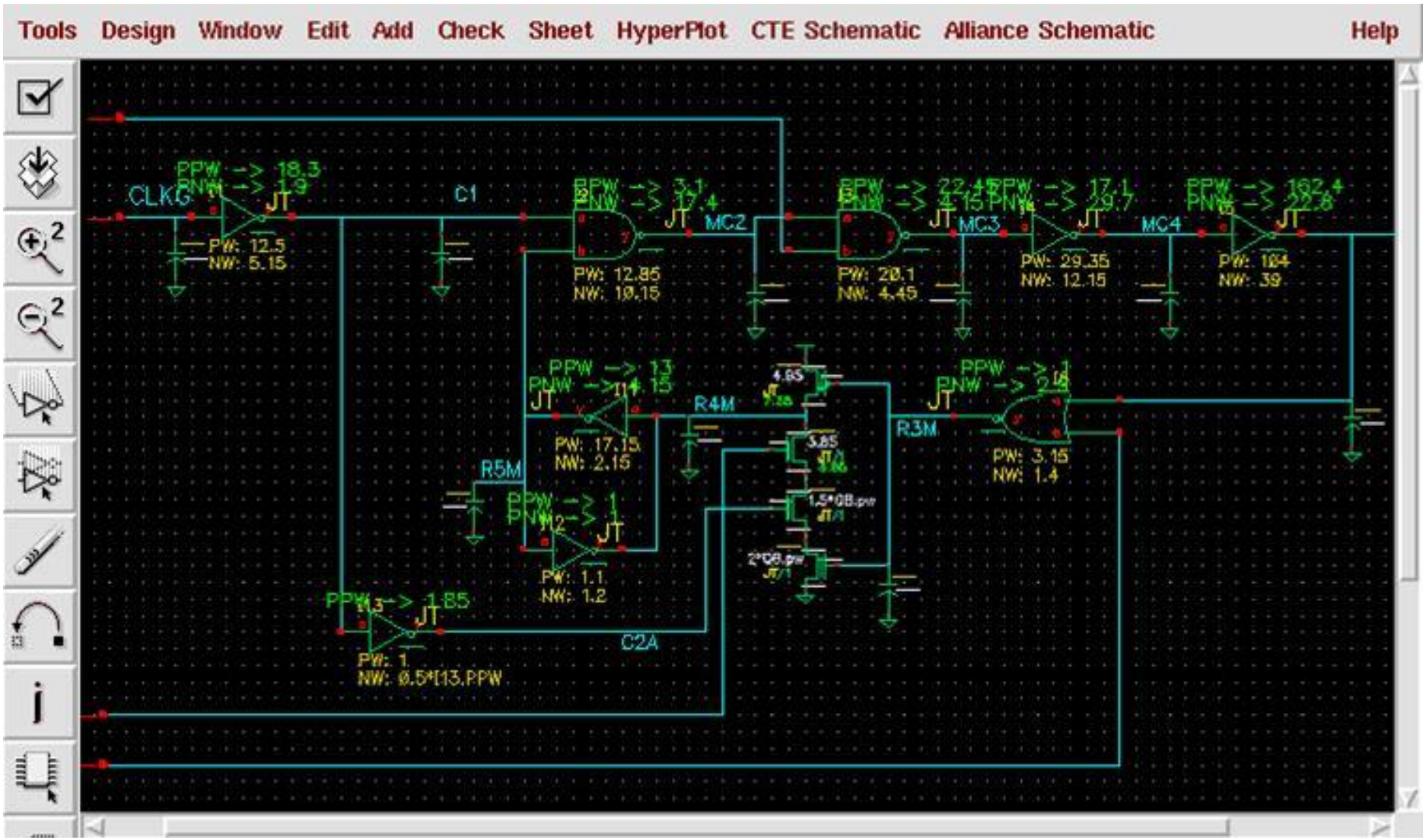
Large-Scale NLPs	Small NLPs
Analytic functions	Expensive function evaluations
– e.g. discretized DAEs/PDEs	– e.g. simulation/experiments
– 1st derivatives available	– no derivatives available
– 2nd deriv. available or approx.	– numerical noise
Many variables (n up to $10,000,000$)	Few variables (n up to 100)
Exploit structure	Methods
– Sparse, structured $\nabla c(x)^T$	– Surrogate models (DFO)
– PDE preconditioners	– Pattern search
– Few vs. many degrees of freedom	

Computationally Challenging Problems

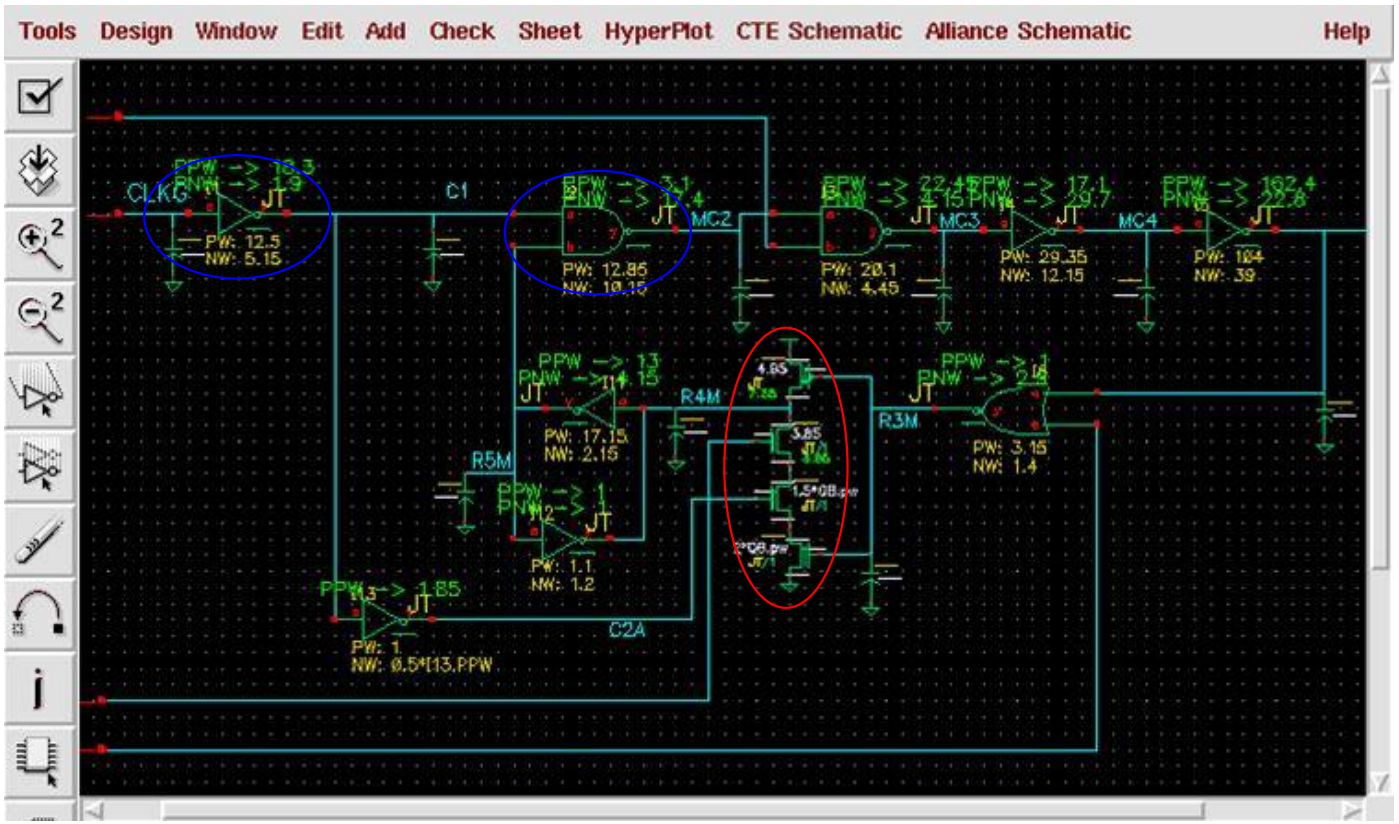
Large-Scale NLPs	Small NLPs
Analytic functions	Expensive function evaluations
– e.g. discretized DAEs/PDEs	– e.g. simulation/experiments
– 1st derivatives available	– no derivatives available
– 2nd deriv. available or approx.	– numerical noise
Many variables (n up to $10,000,000$)	Few variables (n up to 100)
Exploit structure	Methods
– Sparse, structured $\nabla c(x)^T$	– Surrogate models (DFO)
– PDE preconditioners	– Pattern search
– Few vs. many degrees of freedom	

Circuit Tuning

Circuit Tuning: Basic Concepts

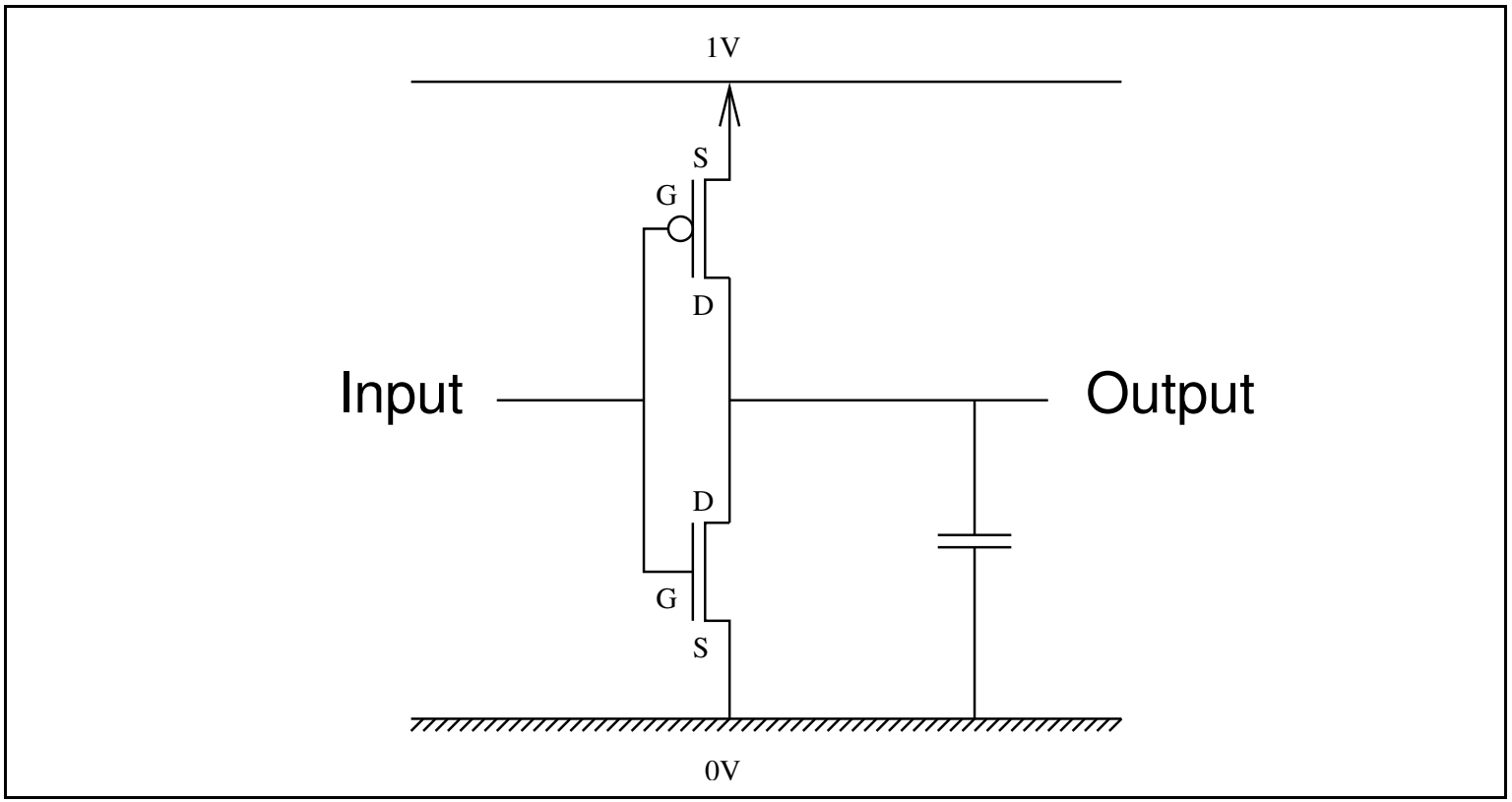


Circuit Tuning: Basic Concepts



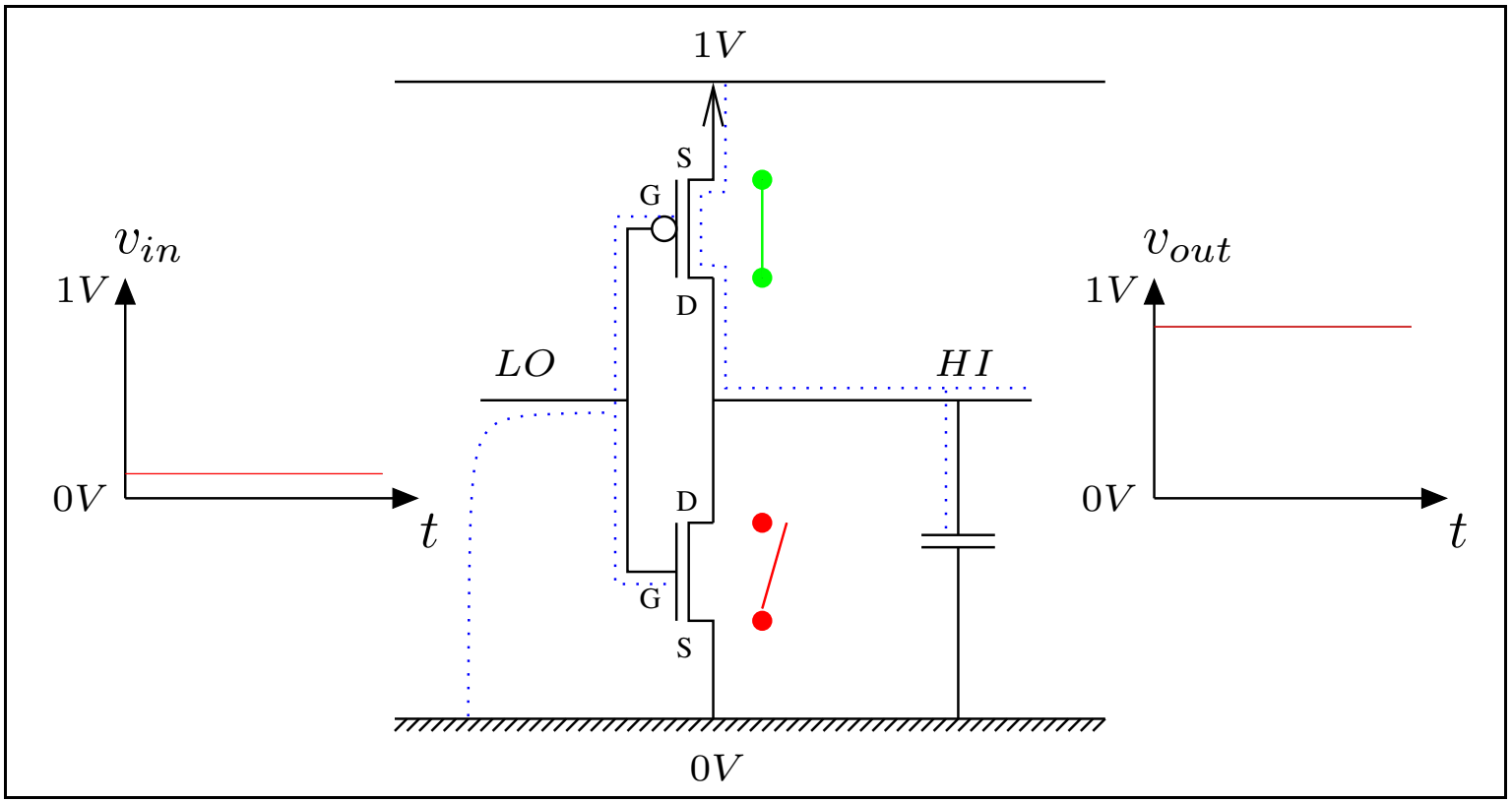
- Basic building blocks:
  - Transistors (switches); Gates (logical units)
- Connected by wires

A Simple Example: Inverter (NOT-Gate)



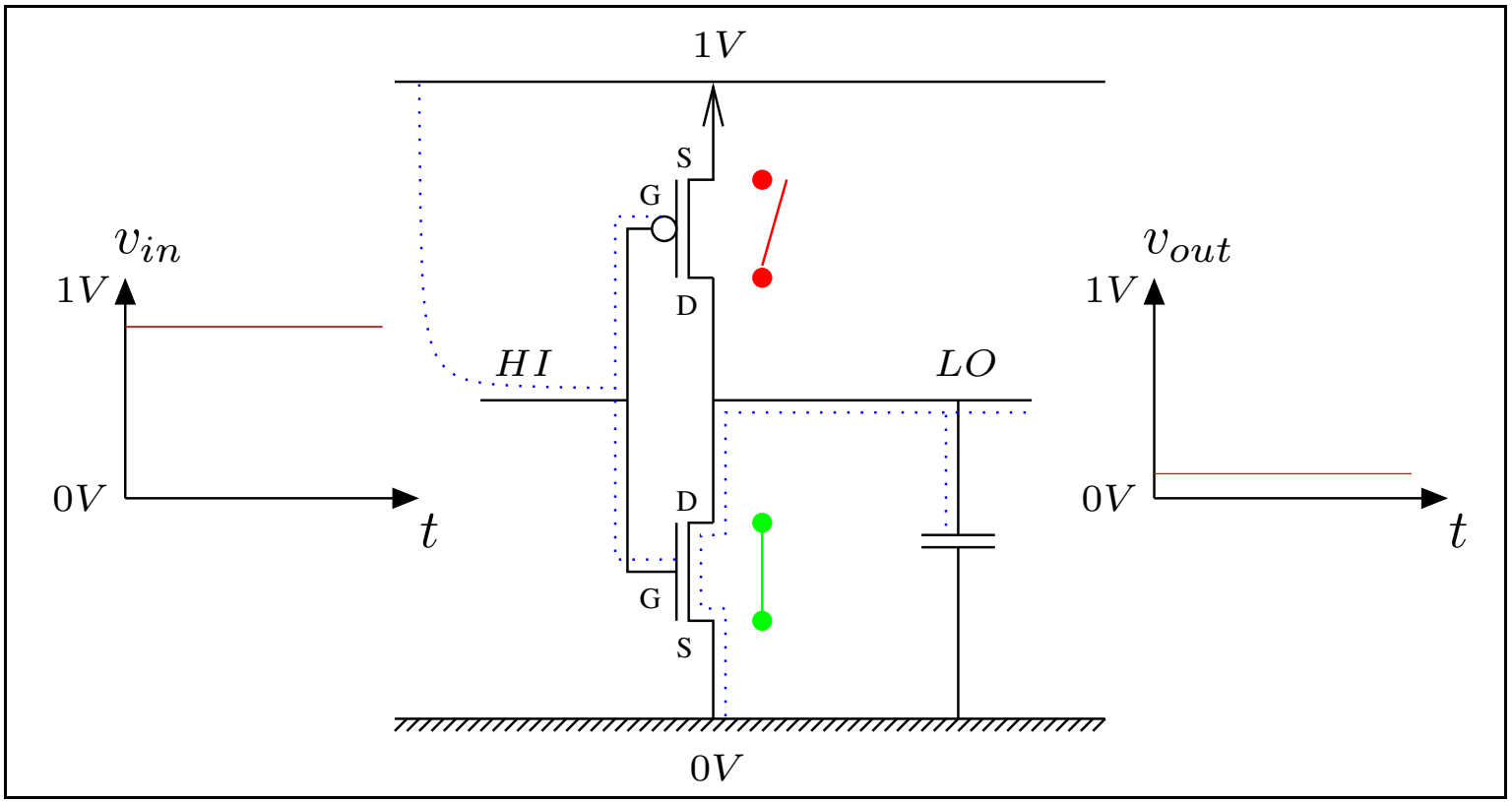
- Two transistors (P-FET and N-FET)

A Simple Example: Inverter (NOT-Gate)



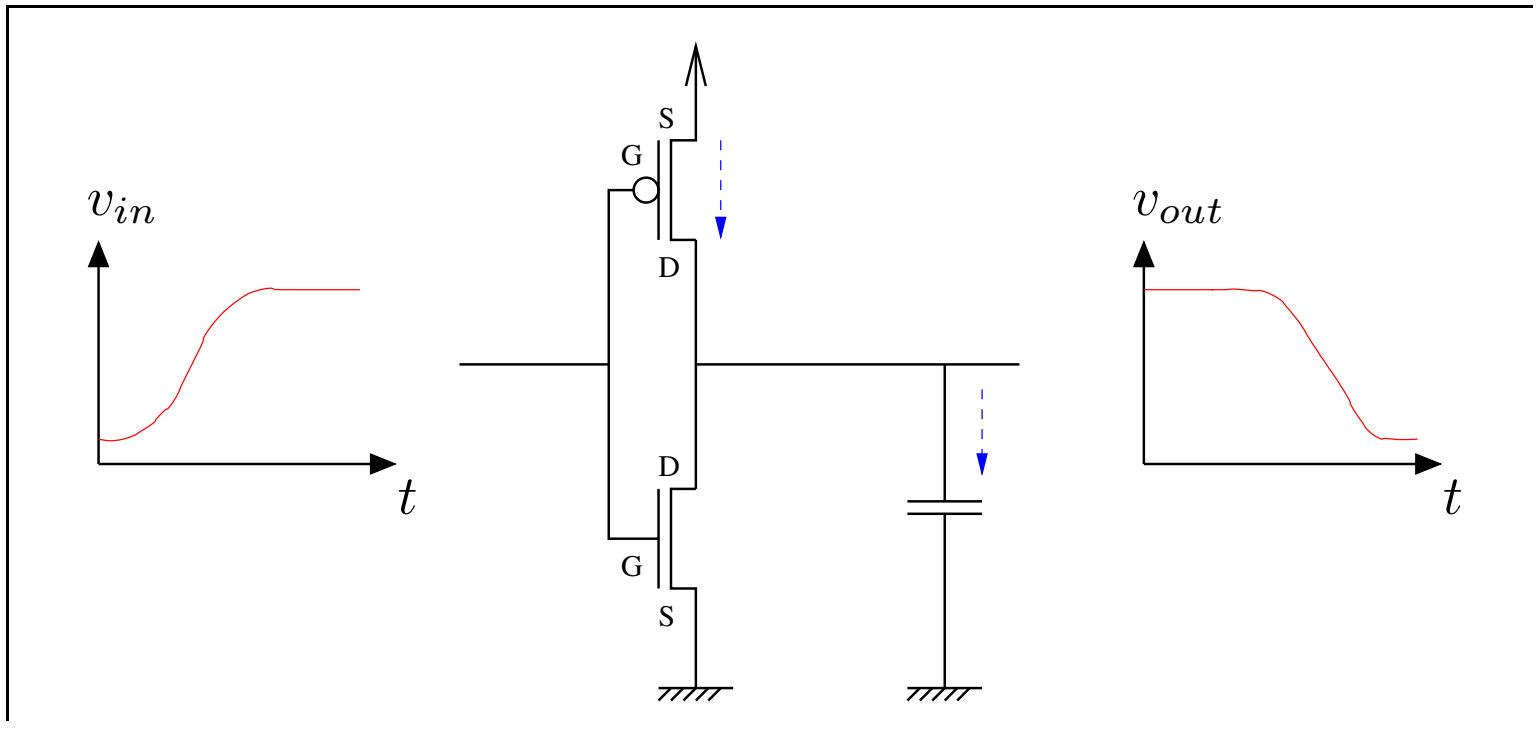
- Two transistors (P-FET and N-FET)
  - Input Lo $\longrightarrow$ Output Hi

A Simple Example: Inverter (NOT-Gate)

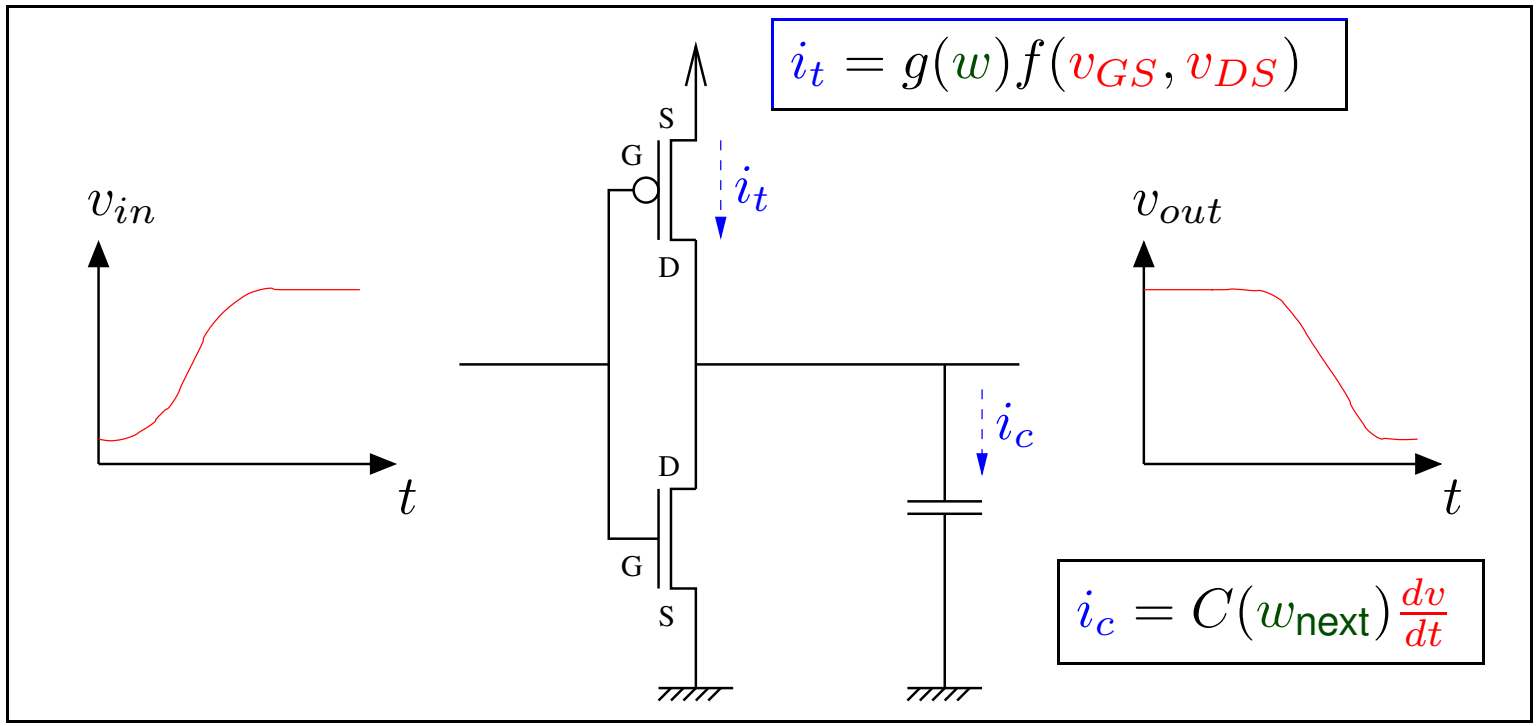


- Two transistors (P-FET and N-FET)
  - Input Lo $\longrightarrow$ Output Hi
  - Input Hi $\longrightarrow$ Output Lo

Dynamic Behavior of a Gate

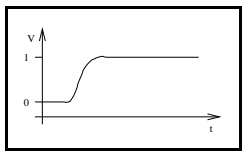


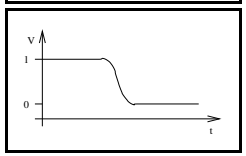
Dynamic Behavior of a Gate

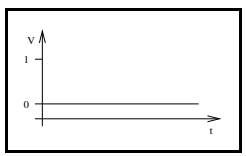


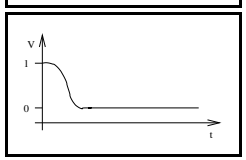
- System of differential and algebraic equations:
  - Conservation laws
  - (Parasitic) capacitors: Include dv/dt terms
  - I-V characteristics for conducting devices

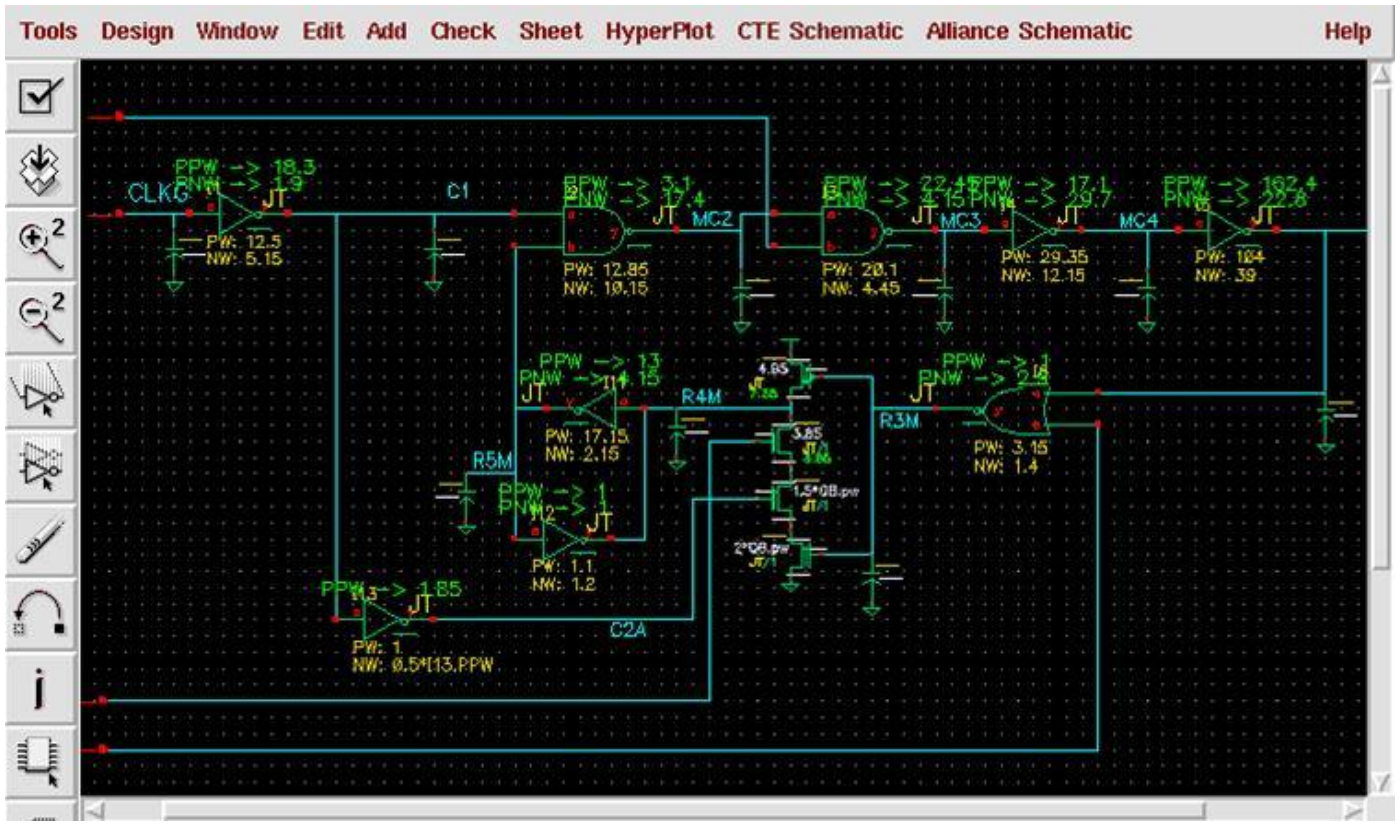
Circuit Tuning: Basic Concepts

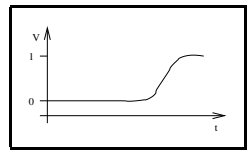






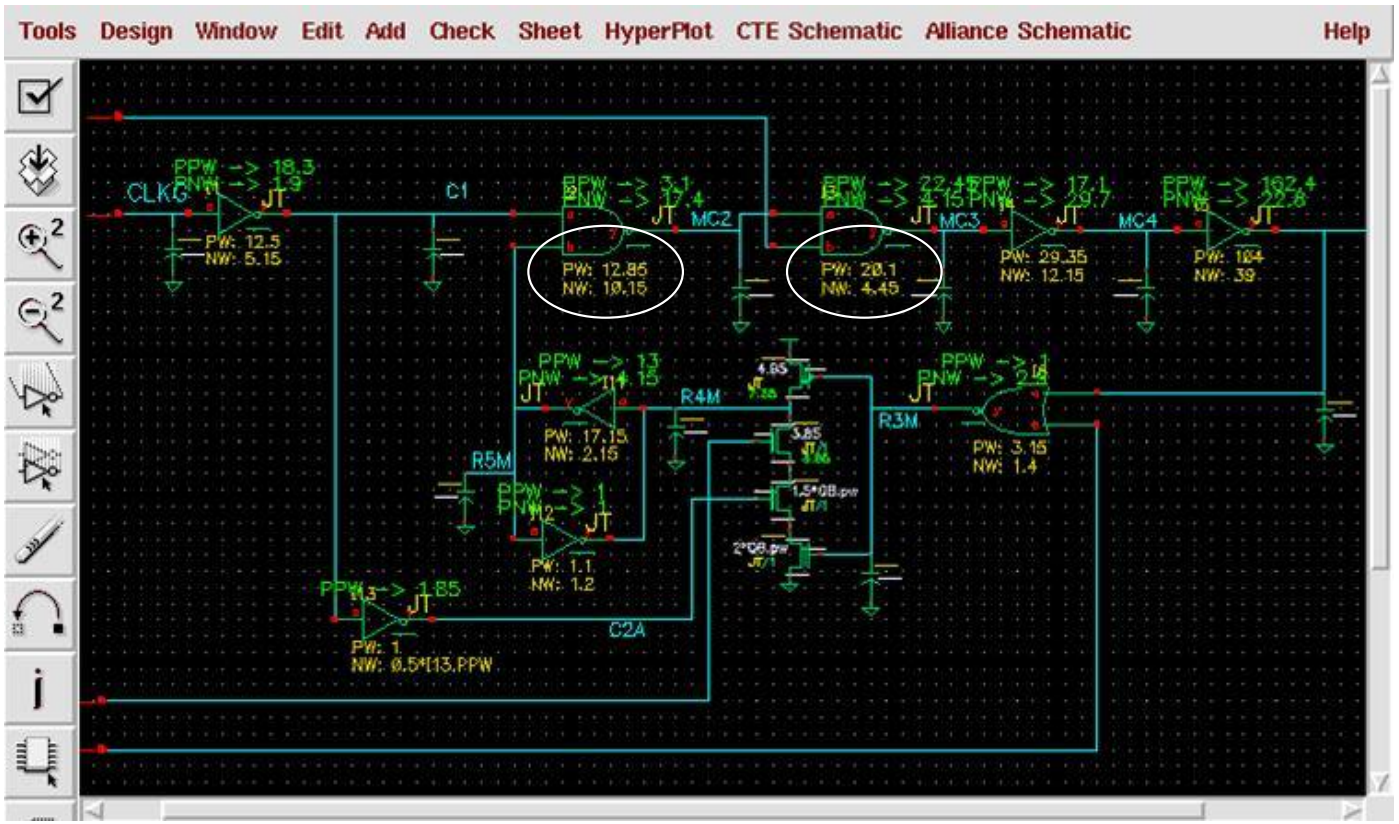






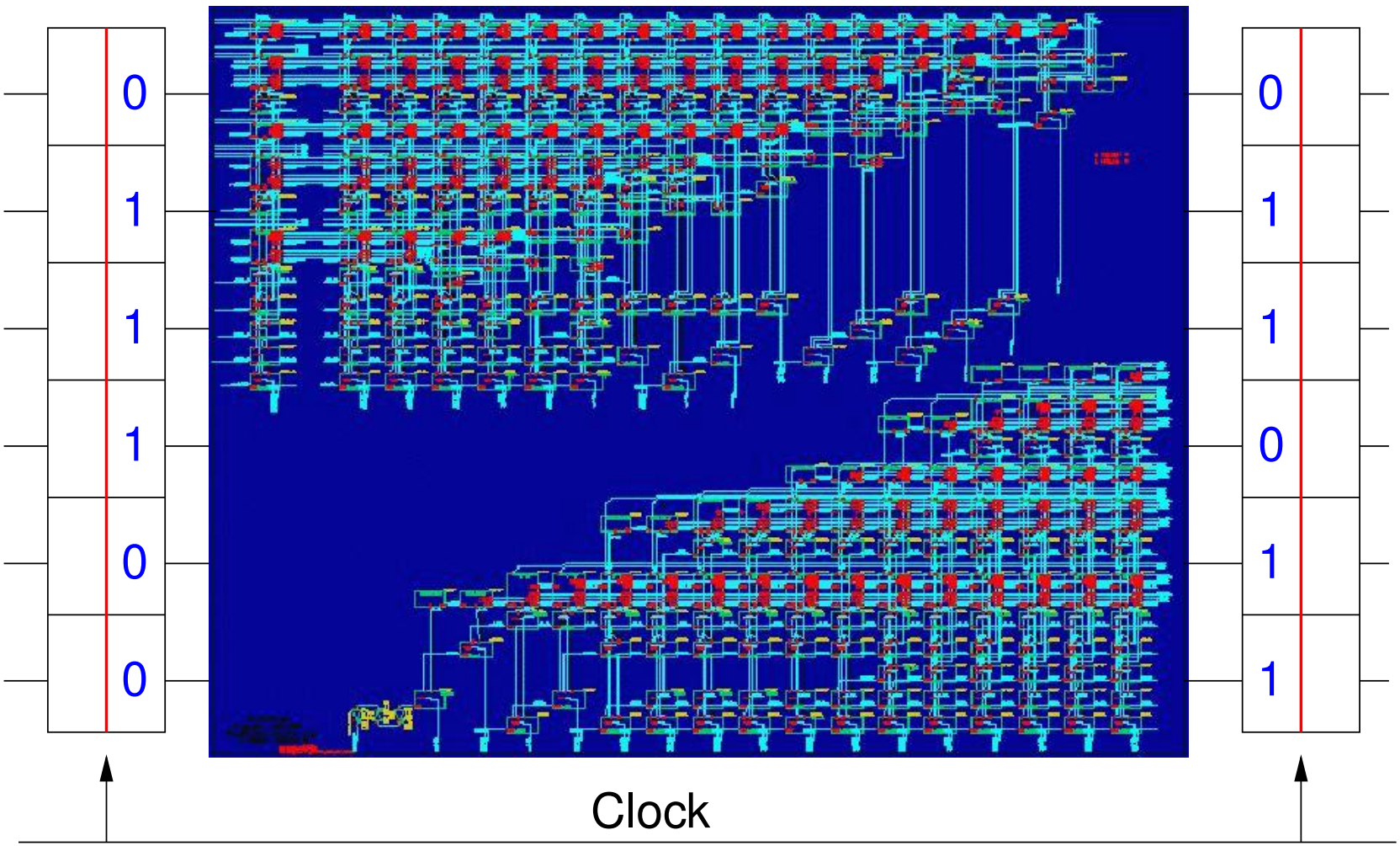
- Signals arrive at the inputs, pass through the circuit, and leave at the outputs (switches at gates on order of 10^{-12} seconds)

Circuit Tuning: Basic Concepts

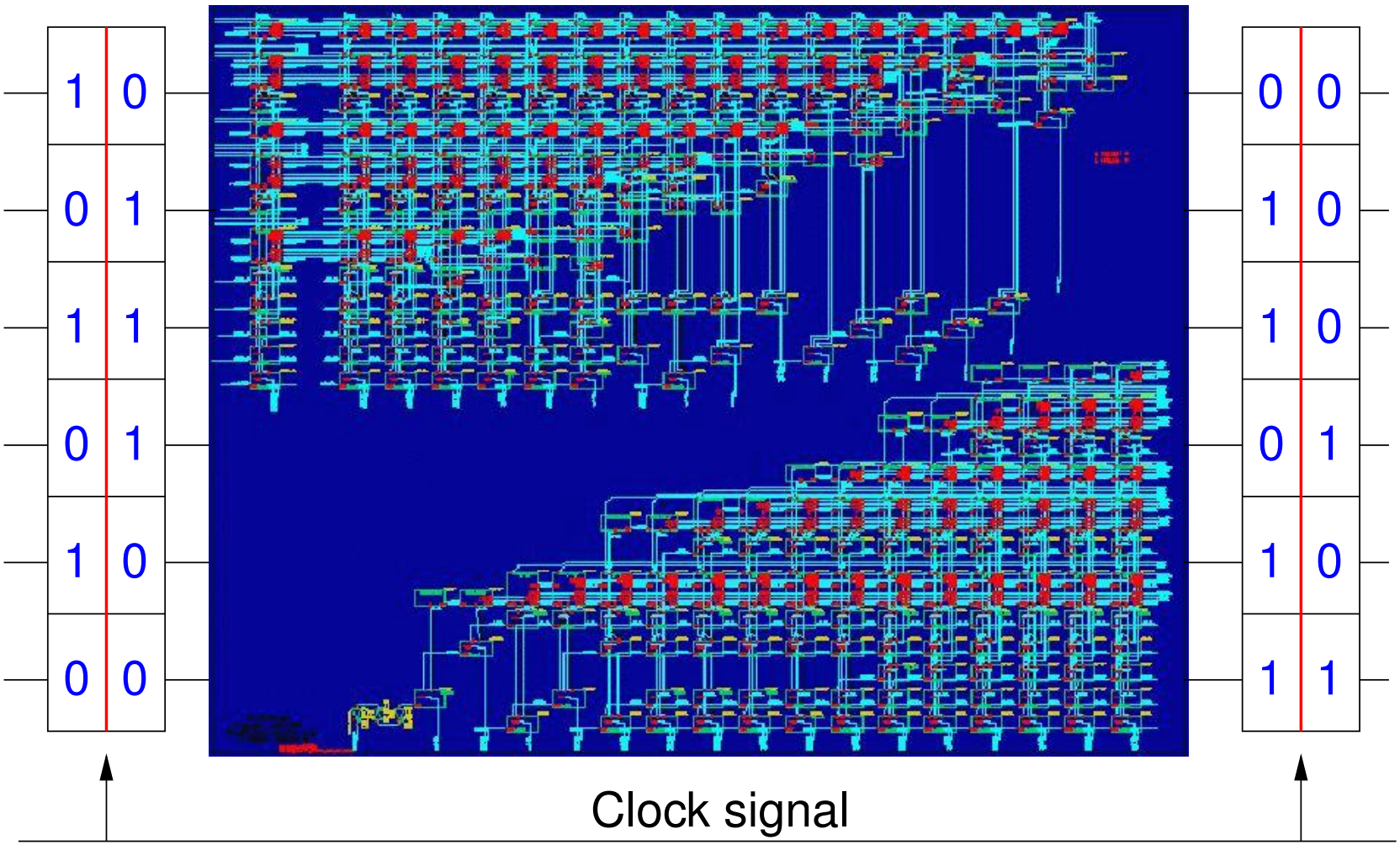


- Can change the “speed” of a gate by changing the widths of its transistors (PFETs and NFETs)

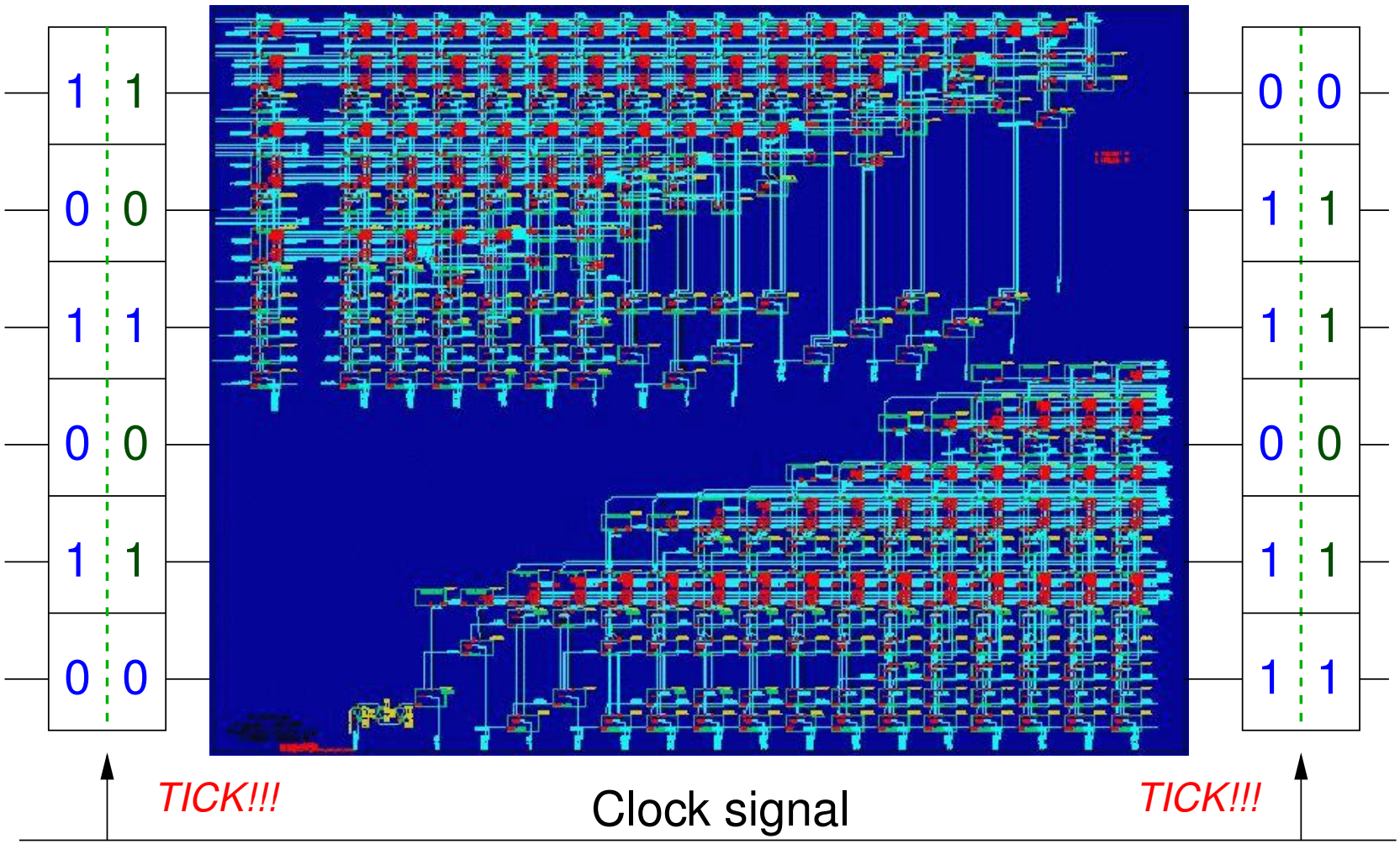
Proper Timing is essential!



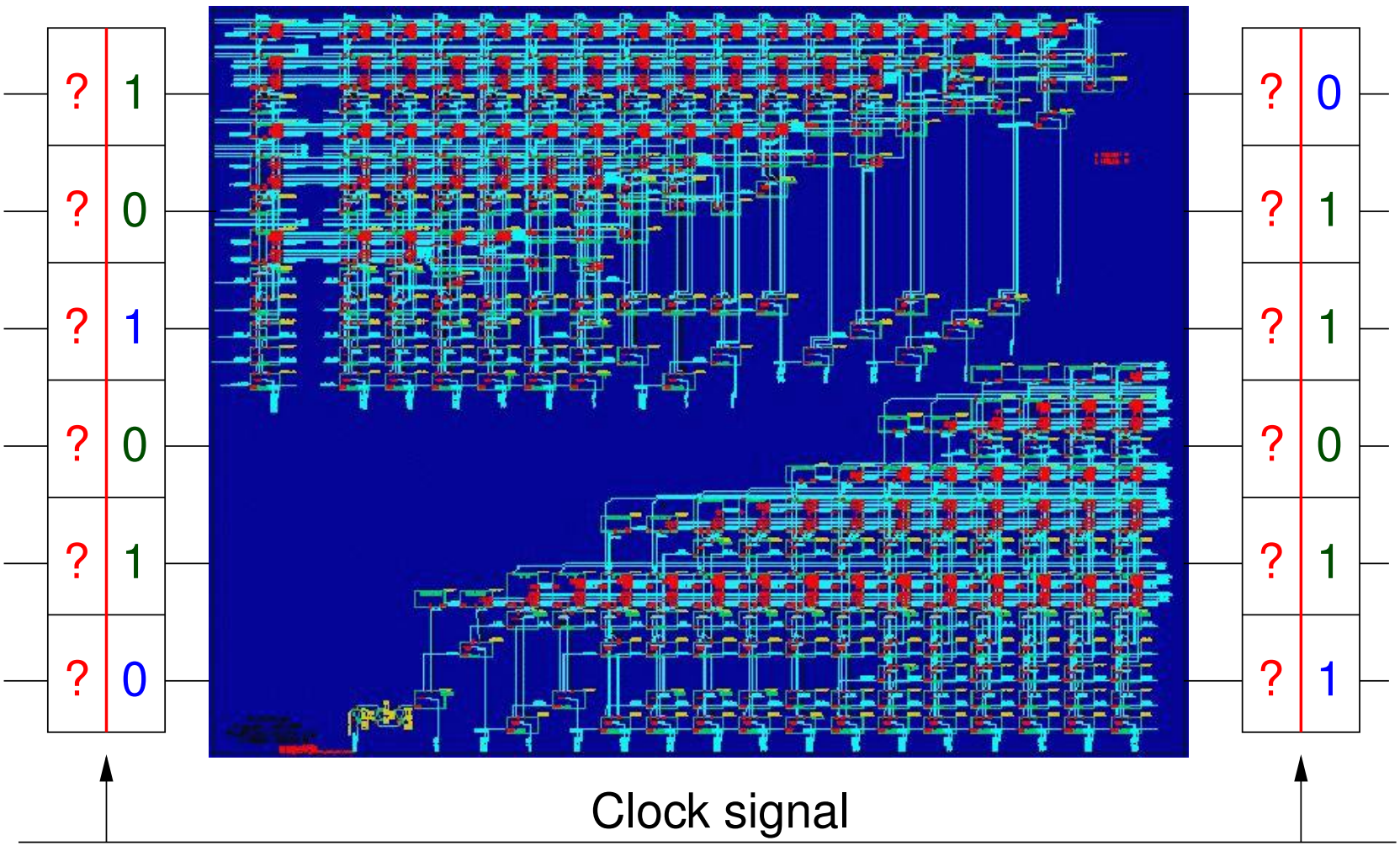
Proper Timing is essential!



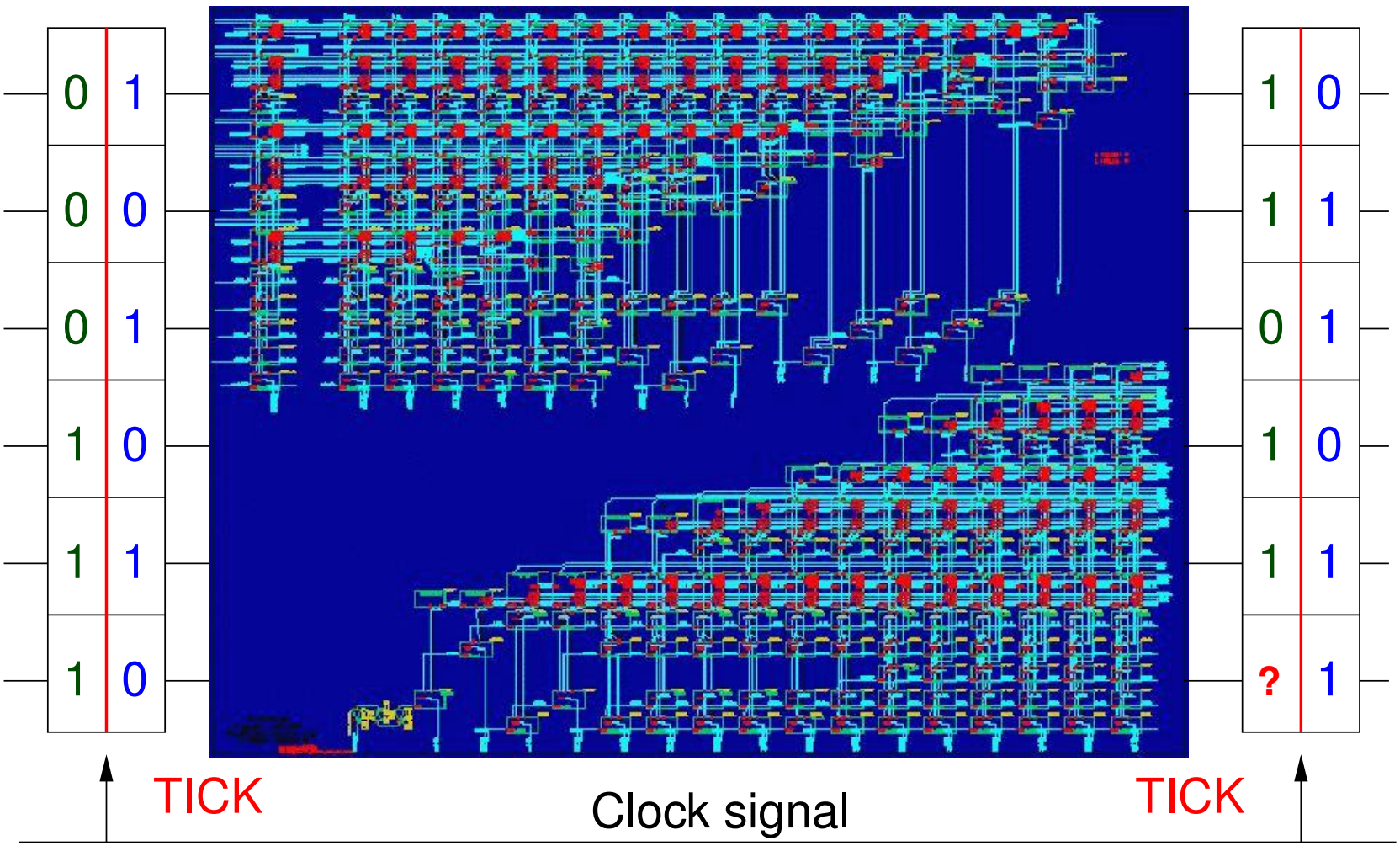
Proper Timing is essential!



Proper Timing is essential!



Proper Timing is essential!



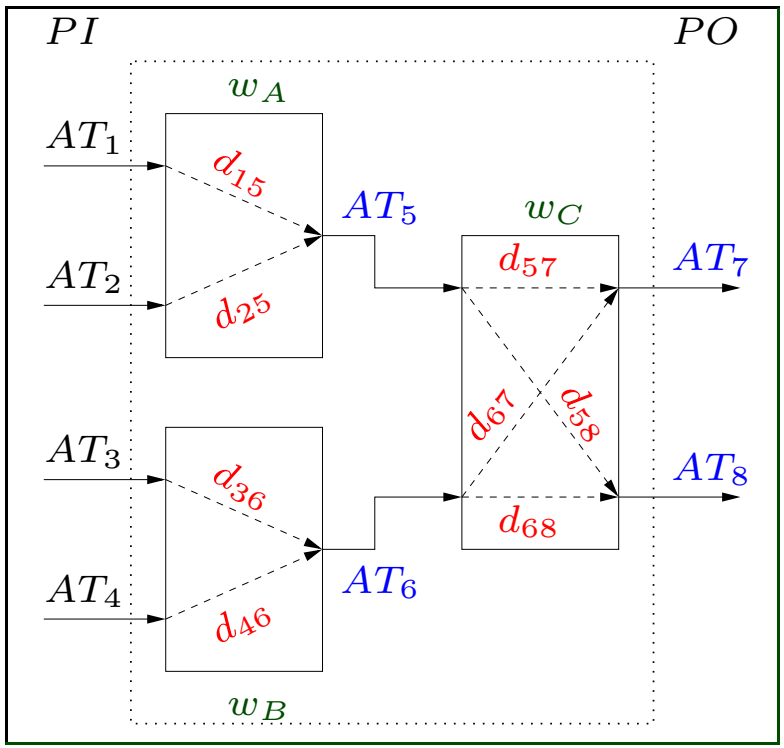
Circuit Tuning

- Stages in Chip Design (iterative)
  - Logic/Functional (boolean)
  - Schematic tuning
  - Physical design (placement/routing)
  - Extracted tuning
- Want to optimize aspects of circuit
  - Delay of signals
  - Area consumption
  - Power consumption

  by changing sizes of transistors
- Often, overall circuit too large
- Simplify circuit (smaller; fix some transistors) $\longrightarrow$ suboptimal solutions
- Currently, tuning circuits with up to 71,022 transistors (19,576 ind.)
- Strong incentive to be able to tune larger circuits in less time

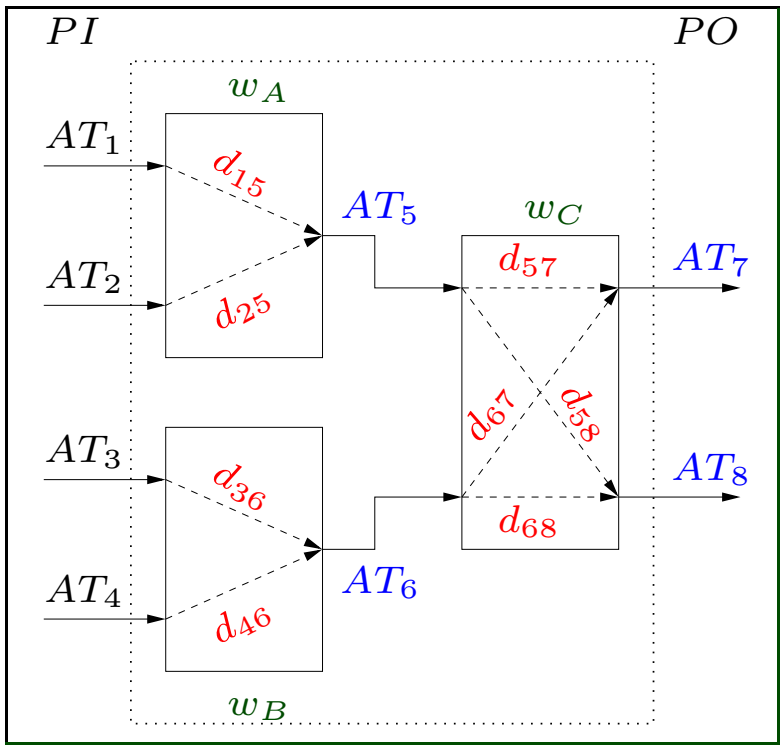
Mathematical Formulation (Static Timing)

Circuit



Mathematical Formulation (Static Timing)

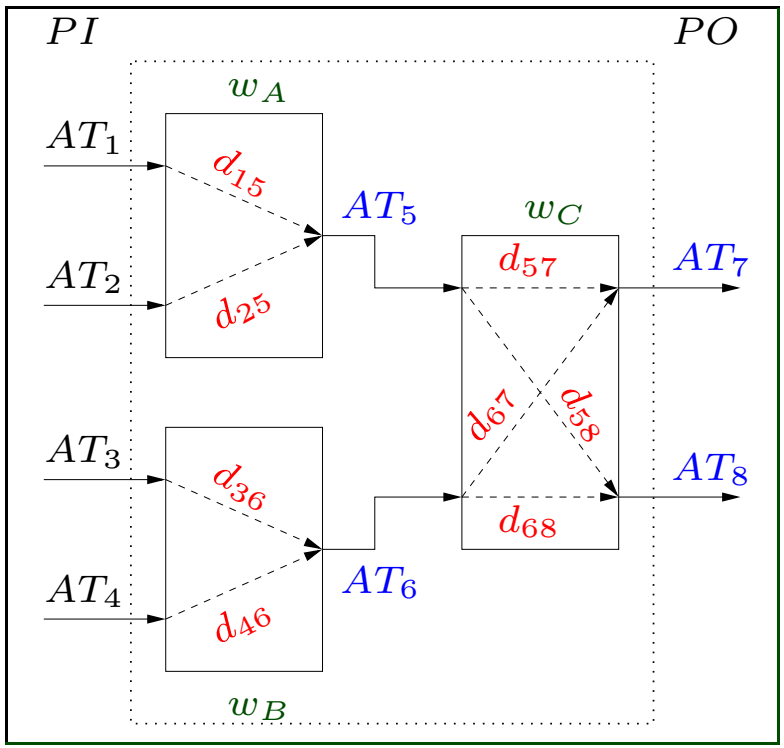
Circuit



$$
\begin{aligned}
AT_5 &= \max\{AT_1 + d_{15}, AT_2 + d_{25}\} \\
AT_6 &= \max\{AT_3 + d_{36}, AT_4 + d_{46}\} \\
AT_7 &= \max\{AT_5 + d_{57}, AT_6 + d_{67}\} \\
AT_8 &= \max\{AT_5 + d_{58}, AT_6 + d_{68}\}
\end{aligned}
$$

Mathematical Formulation (Static Timing)

Circuit

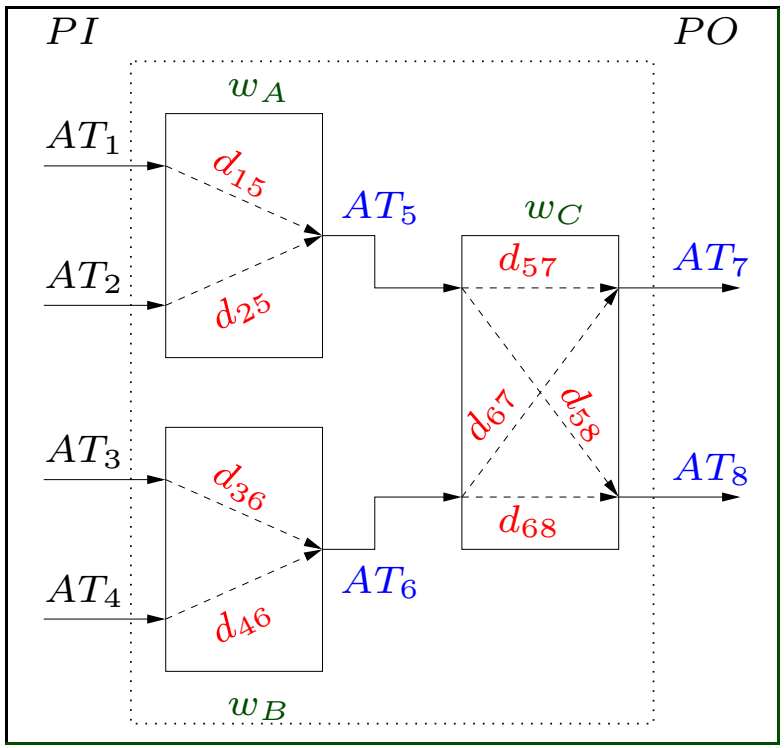


$$
\begin{aligned}
AT_5 &= \max\{AT_1 + d_{15}, AT_2 + d_{25}\} \\
AT_6 &= \max\{AT_3 + d_{36}, AT_4 + d_{46}\} \\
AT_7 &= \max\{AT_5 + d_{57}, AT_6 + d_{67}\} \\
AT_8 &= \max\{AT_5 + d_{58}, AT_6 + d_{68}\}
\end{aligned}
$$

$$AT_j = \max\{AT_i + d_{ij} : i \in \mathsf{input}(j)\}$$

Mathematical Formulation (Static Timing)

Circuit



$$
\begin{aligned}
AT_5 &= \max\{AT_1 + d_{15}, AT_2 + d_{25}\} \\
AT_6 &= \max\{AT_3 + d_{36}, AT_4 + d_{46}\} \\
AT_7 &= \max\{AT_5 + d_{57}, AT_6 + d_{67}\} \\
AT_8 &= \max\{AT_5 + d_{58}, AT_6 + d_{68}\}
\end{aligned}
$$

$$AT_j = \max\{AT_i + d_{ij} : i \in \text{input}(j)\}$$

Delays are functions of transistor widths:

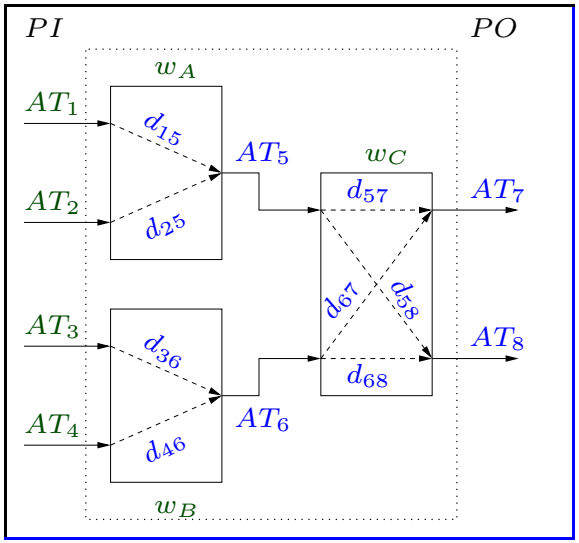
$$d_{ij} = d_{ij}(w_j, w_{\text{next}(j)})$$

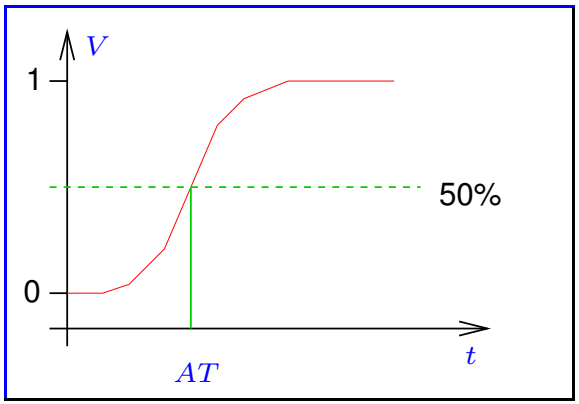
Optimization Problem (Delay Minimization)

$$\min_{AT,w} \quad \max\{AT_i \ : \ i \in PO\}$$

$$s.t. \quad AT_j = \max\{AT_i + d_{ij}(w_j, w_{\text{next}(j)}) \ : \ i \in \text{input}(j)\}$$

$$w_{\min} \leq w \leq w_{\max}$$



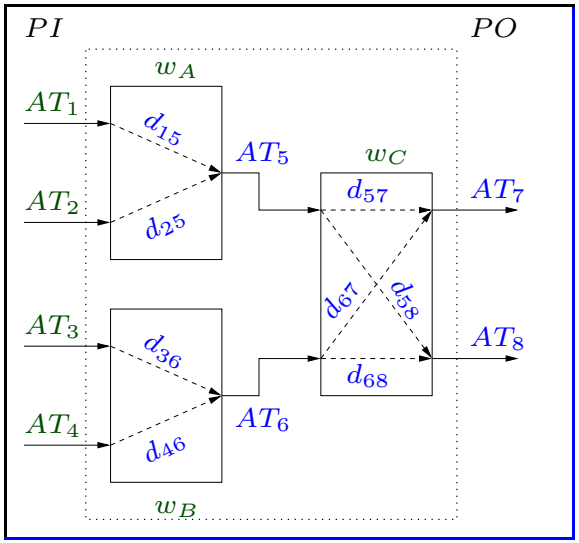


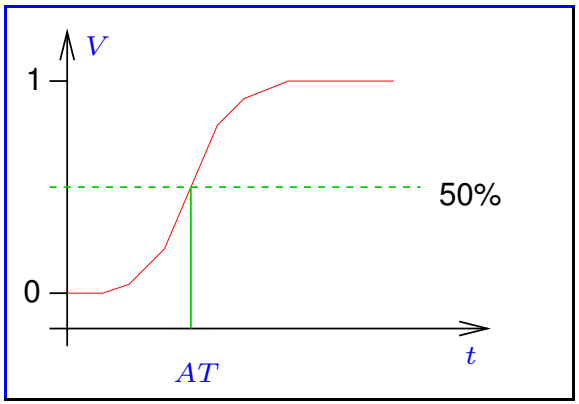
Optimization Problem (Delay Minimization)

$$\min_{AT,w} \quad \max\{AT_i \ : \ i \in PO\}$$

$$s.t. \quad AT_j = \max\{AT_i + d_{ij}(w_j, w_{\text{next}(j)}) \ : \ i \in \text{input}(j)\}$$

$$w_{\min} \leq w \leq w_{\max}$$





$$\min_{x} \quad \max_{i} f_i(x)$$

$\longrightarrow$

$$\min_{x,z} \quad z$$

$$s.t. \quad z \geq f_i(x) \quad \forall i$$

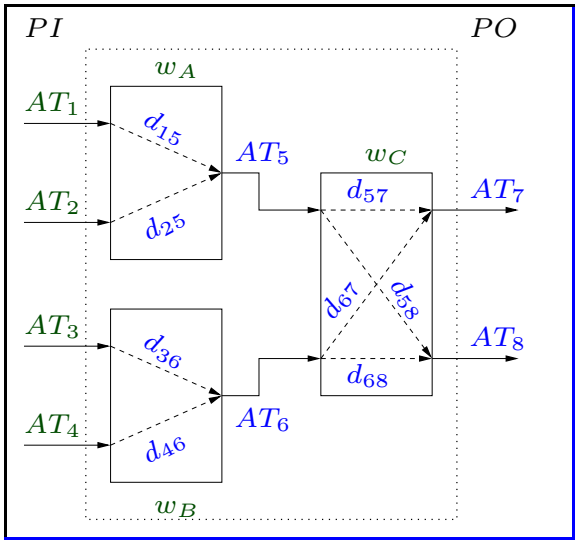
Optimization Problem (Delay Minimization)

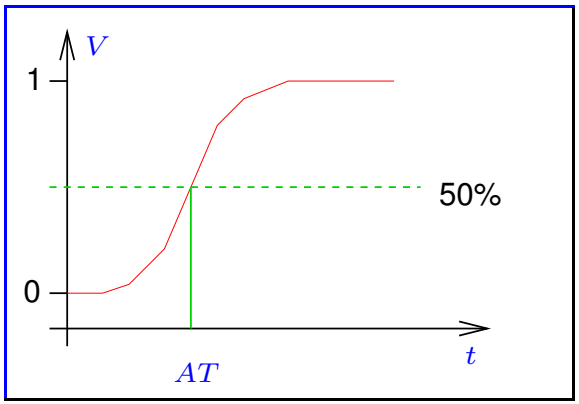
$$\min_{AT,w\ ,z} \quad z$$

$$s.t. \quad z \geq AT_i \qquad i \in PO$$

$$AT_j \geq AT_i + d_{ij}(w_j, w_{\mathsf{next}(j)}\ ) \qquad i \in \mathsf{input}(j)$$

$$w_{\mathsf{min}} \leq w \leq w_{\mathsf{max}}$$





Optimization Problem (Delay Minimization)

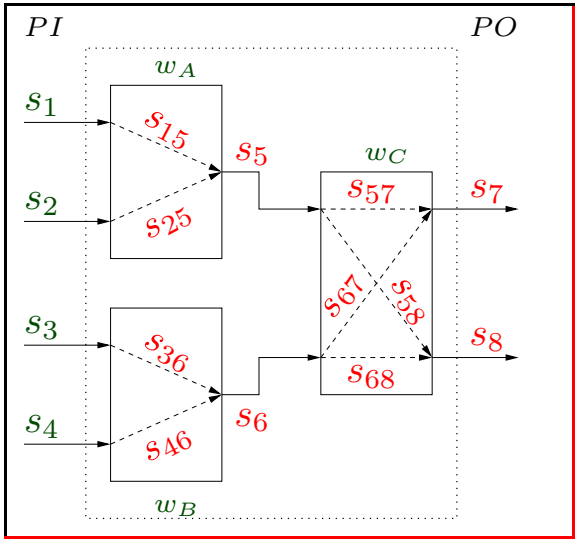
$$\min_{AT,w,s,z} \quad z$$

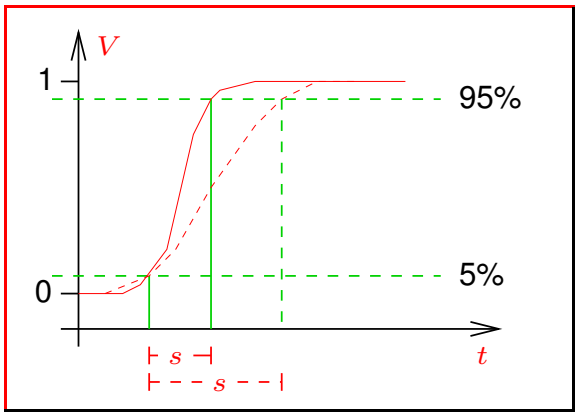
$$s.t. \quad z \geq AT_i \qquad i \in PO$$

$$AT_j \geq AT_i + d_{ij}(w_j, w_{\text{next}(j)}, s_i) \qquad i \in \text{input}(j)$$

$$s_j \geq s_{ij}(w_j, w_{\text{next}(j)}, s_i) \qquad i \in \text{input}(j)$$

$$w_{\min} \leq w \leq w_{\max}, \qquad s_{\min} \leq s \leq s_{\max}$$





Optimization Problem (Delay Minimization)

$$\min_{AT,w,s,z} \quad z$$

$$s.t. \quad z \geq AT_i \qquad i \in PO$$

$$AT_j \geq AT_i + d_{ij}(w_j, w_{\text{next}(j)}, s_i) \qquad i \in \text{input}(j)$$

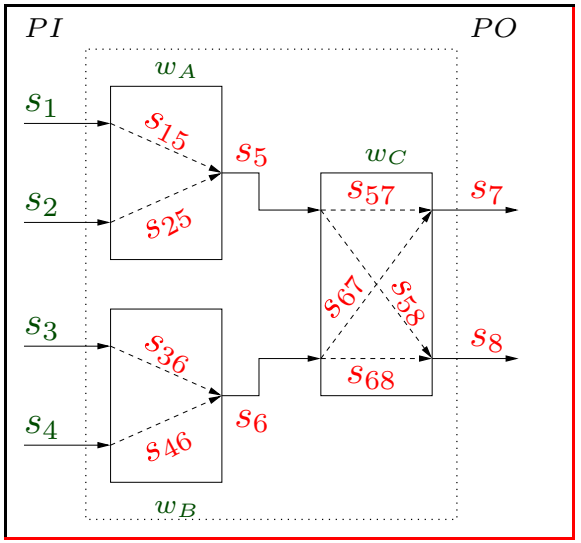
$$s_j \geq s_{ij}(w_j, w_{\text{next}(j)}, s_i) \qquad i \in \text{input}(j)$$

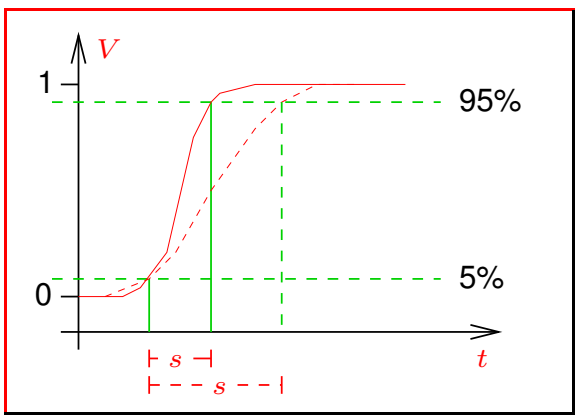
$$\sum A_i w_i \leq A_{\max}$$

$$C_i(w) \leq C_i^{\max} \qquad i \in PI$$

$$\beta_i^{\min} \leq \frac{w_i^{\text{PFET}}}{w_i^{\text{NFET}}} \leq \beta_i^{\max} \qquad i \in G$$

$$w_{\min} \leq w \leq w_{\max}, \qquad s_{\min} \leq s \leq s_{\max}$$





Properties of Optimization Problem

- The nonlinear functions $d_{ij}(w_i, w_{\text{next}(i)}, s_i)$ and $s_{ij}(w_i, w_{\text{next}(i)}, s_i)$ are obtained by SPECS (event driven circuit simulator) (Visweswariah)
  - Very fast compared to standard circuit simulators
  - First derivatives available
  - Computationally expensive part in optimization
  - Numerical noise (mostly from integration)

Properties of Optimization Problem

- The nonlinear functions $d_{ij}(w_i, w_{\text{next}(i)}, s_i)$ and $s_{ij}(w_i, w_{\text{next}(i)}, s_i)$ are obtained by SPECS (event driven circuit simulator) (Visweswariah)
  - Very fast compared to standard circuit simulators
  - First derivatives available
  - Computationally expensive part in optimization
  - Numerical noise (mostly from integration)
- After optimization, transistor widths are snapped to grid
  - Don't need highly accurate solution

Properties of Optimization Problem

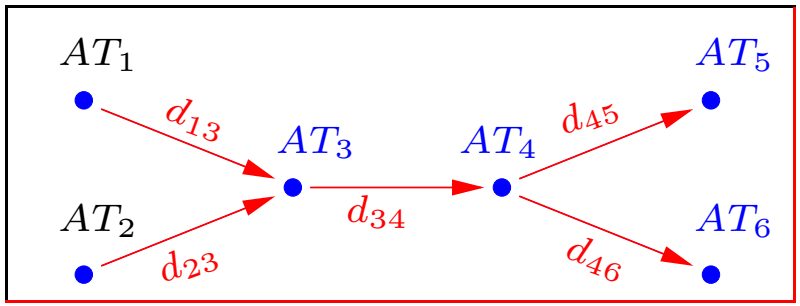
- The nonlinear functions $d_{ij}(w_i, w_{\text{next}(i)}, s_i)$ and $s_{ij}(w_i, w_{\text{next}(i)}, s_i)$ are obtained by SPECS (event driven circuit simulator) (Visweswariah)
  - Very fast compared to standard circuit simulators
  - First derivatives available
  - Computationally expensive part in optimization
  - Numerical noise (mostly from integration)
- After optimization, transistor widths are snapped to grid
  - Don't need highly accurate solution
- Many degenerate variables (AT and s in non-critical paths)
- Many variables and many degrees of freedom (n up to $736,000$)
- Computation time can be up to several days

EinsTuner

- IBM-internal software tool for designers of custom digital circuits
- Previously: Nonlinear optimization problem solved with Lancelot
  (Conn, Gould, Toint)
- Had to be customized:
  - Noise considerations
  - Made aggressive
- Preprocessing (Pruning)

Pruning

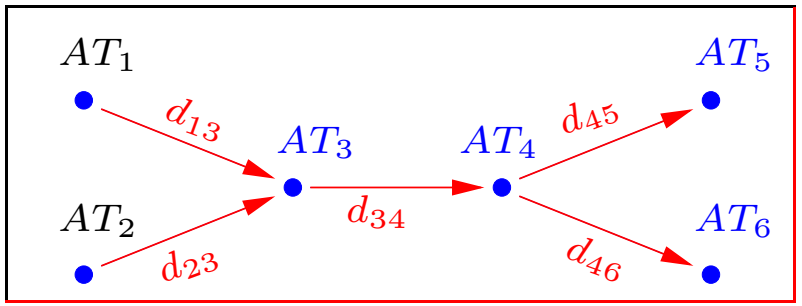
1.



$$
\begin{aligned}
AT_3 &\geq AT_1 + d_{13} \\
AT_3 &\geq AT_2 + d_{23} \\
AT_4 &\geq AT_3 + d_{34} \\
AT_5 &\geq AT_4 + d_{45} \\
AT_6 &\geq AT_4 + d_{46}
\end{aligned}
$$

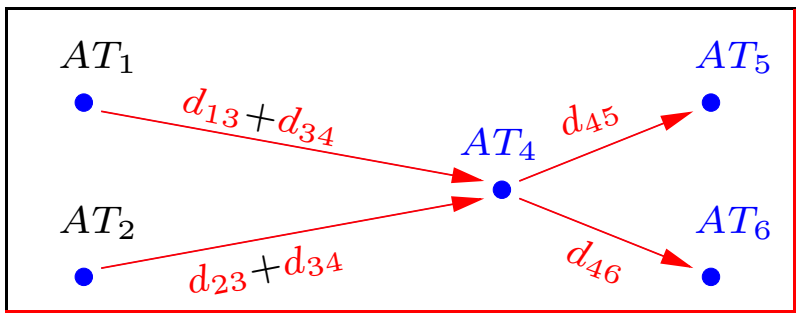
Pruning

1.



$$
\begin{aligned}
AT_3 &\geq AT_1 + d_{13} \\
AT_3 &\geq AT_2 + d_{23} \\
AT_4 &\geq AT_3 + d_{34} \\
AT_5 &\geq AT_4 + d_{45} \\
AT_6 &\geq AT_4 + d_{46}
\end{aligned}
$$

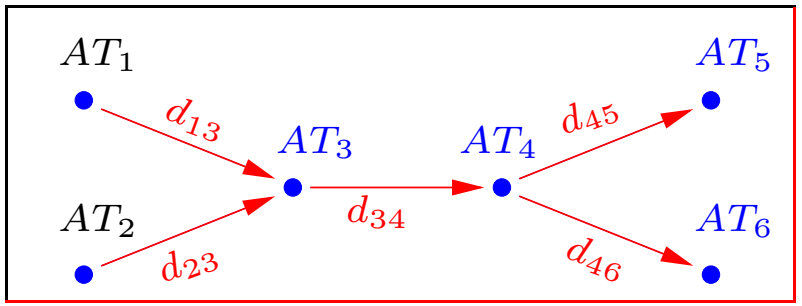
2.



$$
\begin{aligned}
AT_4 &\geq AT_1 + d_{13} + d_{34} \\
AT_4 &\geq AT_2 + d_{23} + d_{34} \\
AT_5 &\geq AT_4 + d_{45} \\
AT_6 &\geq AT_4 + d_{46}
\end{aligned}
$$

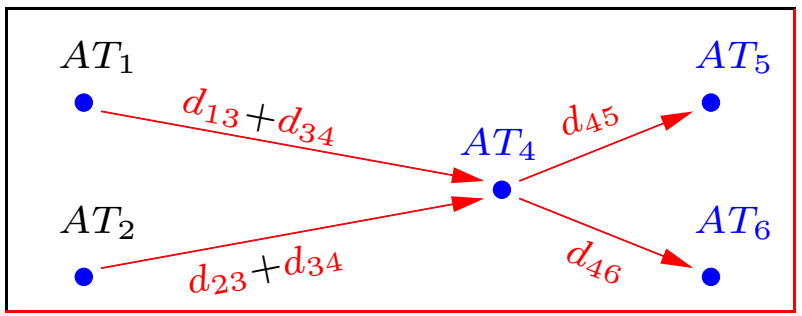
Pruning

1.



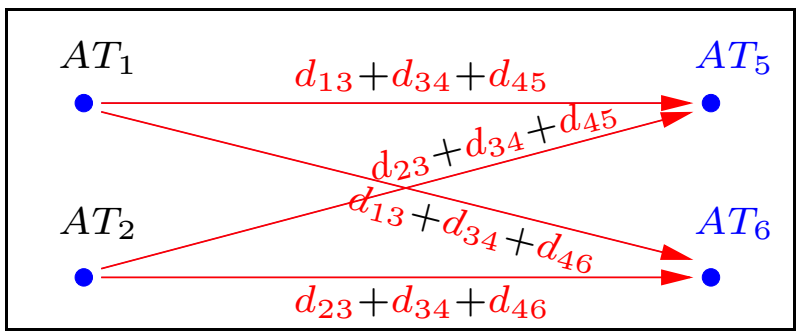
$$
\begin{aligned}
AT_3 &\geq AT_1 + d_{13} \\
AT_3 &\geq AT_2 + d_{23} \\
AT_4 &\geq AT_3 + d_{34} \\
AT_5 &\geq AT_4 + d_{45} \\
AT_6 &\geq AT_4 + d_{46}
\end{aligned}
$$

2.



$$
\begin{aligned}
AT_4 &\geq AT_1 + d_{13} + d_{34} \\
AT_4 &\geq AT_2 + d_{23} + d_{34} \\
AT_5 &\geq AT_4 + d_{45} \\
AT_6 &\geq AT_4 + d_{46}
\end{aligned}
$$

3.



$$
\begin{aligned}
AT_5 &\geq AT_1 + d_{45} + d_{13} + d_{34} \\
AT_5 &\geq AT_2 + d_{45} + d_{23} + d_{34} \\
AT_6 &\geq AT_1 + d_{46} + d_{13} + d_{34} \\
AT_6 &\geq AT_2 + d_{46} + d_{23} + d_{34}
\end{aligned}
$$

EinsTuner

- IBM-internal software tool for designers of custom digital circuits
- Previously: Nonlinear optimization problem solved with Lancelot
  (Conn, Gould, Toint)
- Had to be customized:
  - Noise considerations
  - Made aggressive
- Preprocessing (Pruning)
  - *Equivalent* formulation
  - Reduced number of constraints and (degenerate) variables
  - Usual reduction on large problems:
    33%↓ vars, 22%↓ cons, 1.7-4.1x↓ CPU time (Lancelot)

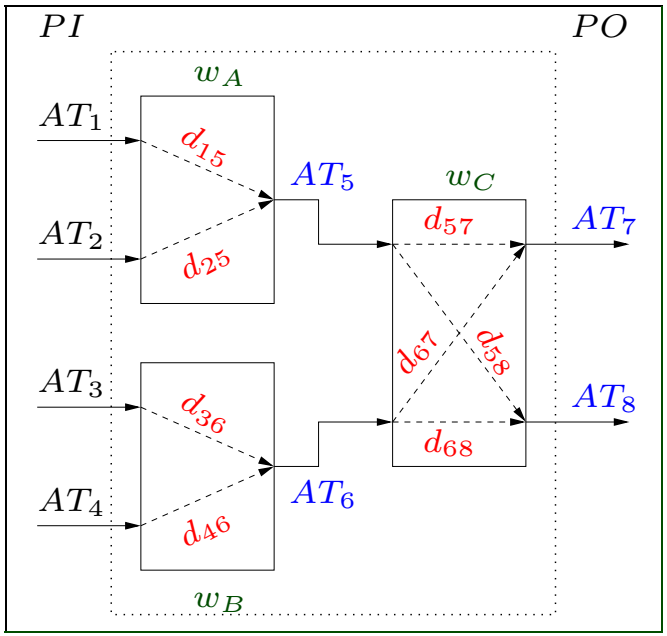
EinsTuner

- IBM-internal software tool for designers of custom digital circuits
- Previously: Nonlinear optimization problem solved with Lancelot
  (Conn, Gould, Toint)
- Had to be customized:
  - Noise considerations
  - Made aggressive
- Preprocessing (Pruning)
  - *Equivalent* formulation
  - Reduced number of constraints and (degenerate) variables
  - Usual reduction on large problems:
    33%↓ vars, 22%↓ cons, 1.7-4.1x↓ CPU time (Lancelot)
- Used for the design of every custom digital chip in IBM
  - 15% gain in speed over carefully hand-tuned circuits

EinsTuner

- IBM-internal software tool for designers of custom digital circuits
- Previously: Nonlinear optimization problem solved with Lancelot
  (Conn, Gould, Toint)
- Had to be customized:
  - Noise considerations
  - Made aggressive
- Preprocessing (Pruning)
  - *Equivalent* formulation
  - Reduced number of constraints and (degenerate) variables
  - Usual reduction on large problems:
    33%↓ vars, 22%↓ cons, 1.7-4.1x↓ CPU time (Lancelot)
- Used for the design of every custom digital chip in IBM
  - 15% gain in speed over carefully hand-tuned circuits
- Integration of IPOPT optimization code

Integration of IPOPT in EinsTuner



$$\min_{AT,w} \quad AT_{\text{latest}}$$

$$s.t. \quad AT_{\text{latest}} \geq AT_i$$

$$AT_j \geq AT_i + d_{ij}(w_i)$$

$$\sum_i \text{Area}_i w_i \leq \text{Area}_{\text{max}}$$

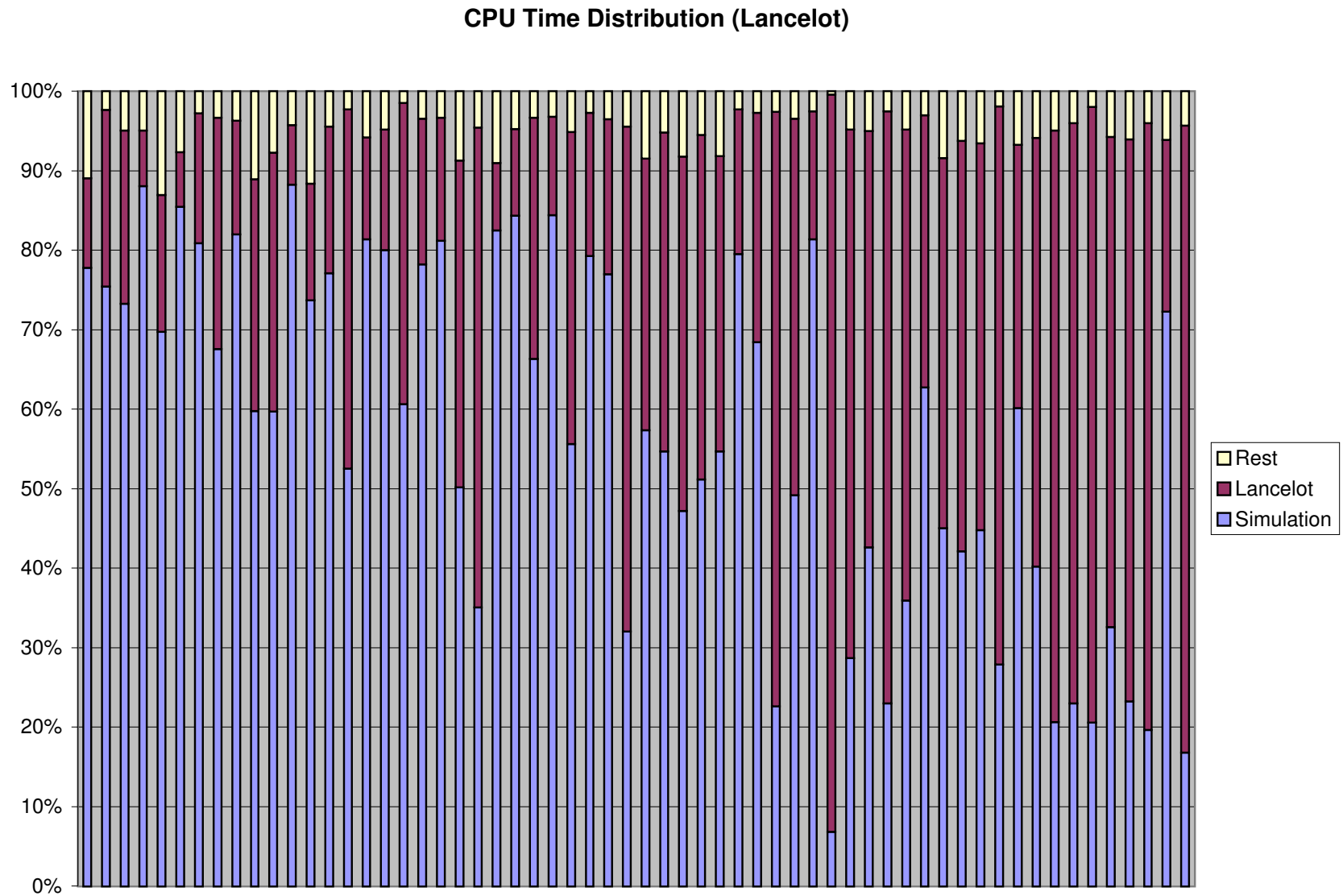
$$w_{\text{min}} \leq w \leq w_{\text{max}}$$

- Approximate 2nd derivatives with limited-memory BFGS
- Factorize linear system with WSMP (Gupta), a sparse direct parallel solver
- Overall faster and more robust than previous optimization engine
- Used extensively in production
- Since recently using adaptive barrier parameter strategy

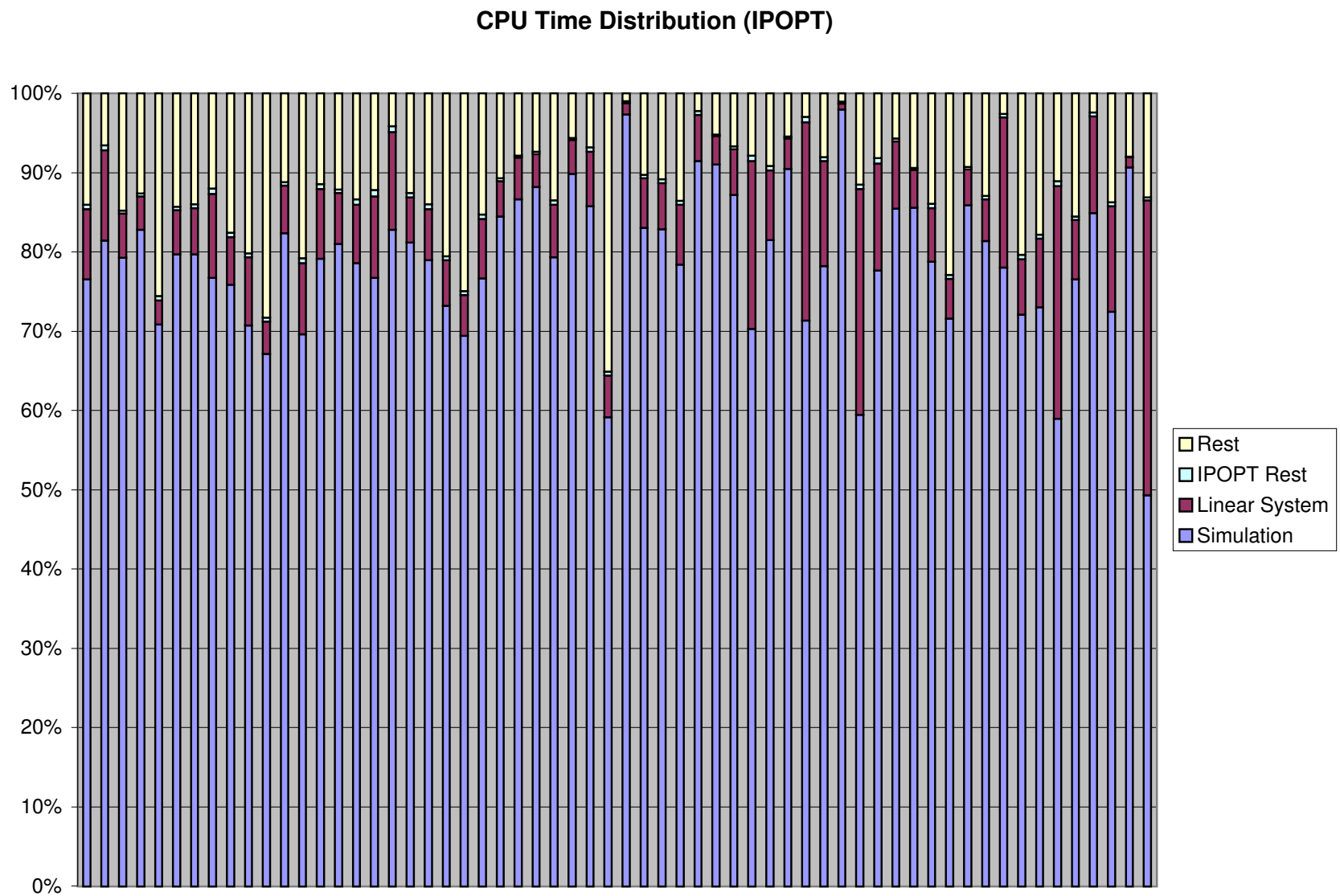
Next slide:

- 51 benchmark problems (n=1,261,. . . ,161,701)

Breakdown of CPU time — Lancelot



Breakdown of CPU time — Ipopt



Mixed-Integer Nonlinear Optimization

The CMU–IBM Bonmin Collaboration

- Goals:
  - Develop MINLP algorithms
  - Implement as open course software (as BONMIN in COIN-OR)
  - Release publicly available MINLP test set

- The Gang:

CMU

- Pietro Belotti
- Larry T. Biegler
- Gérard Cornuéjols
- Ignacio E. Grossmann
- Carl D. Laird
- François Margot
- Nick Sawaya

IBM

- Pierre Bonami
- Andrew R. Conn
- John J. Forrest
- Oktay Günlük
- Laci Ladanyi
- Jon Lee
- Andrea Lodi (U Bologna)
- Andreas Wächter

Outline

- Algorithms implemented in BONMIN
  - Branch-and-bound
  - Outer approximation
  - A hybrid approach
- Numerical comparison

- Improving branching rules for branch-and-bound
  - Variants of strong branching
- Numerical comparison

- Related and future work

Mixed Integer Nonlinear Programming

$$(P) \quad \boxed{\begin{array}{ll} \min & f(x,y) \\ s.t. & c(x,y) \leq 0 \\ & x \in \{0,1\}^n, \quad y \in \mathbb{R}^p \end{array}}$$

- If f and c linear $\Longrightarrow$ MILP
- If f and c nonlinear $\Longrightarrow$ MINLP
- If f and c convex $\Longrightarrow$ convex MINLP (focus here)

Mixed Integer Nonlinear Programming

$$(P) \quad \begin{array}{ll} \min & f(x,y) \\ s.t. & c(x,y) \leq 0 \\ & x \in \{0,1\}^n, \quad y \in \mathbb{R}^p \end{array}$$

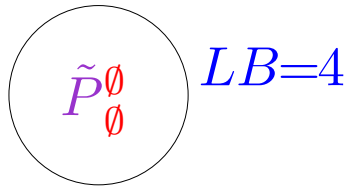
- If f and c linear $\implies$ MILP
- If f and c nonlinear $\implies$ MINLP
- If f and c convex $\implies$ convex MINLP (focus here)

$$(\tilde{P}^{\mathcal{U}}_{\mathcal{D}}) \quad \begin{array}{ll} \min\limits_{x\in[0,1]^n, y\in\mathbb{R}^p} & f(x,y) \\ s.t. & c(x,y) \leq 0 \\ & x_{\mathcal{D}} = 0, \quad x_{\mathcal{U}} = 1 \end{array}$$

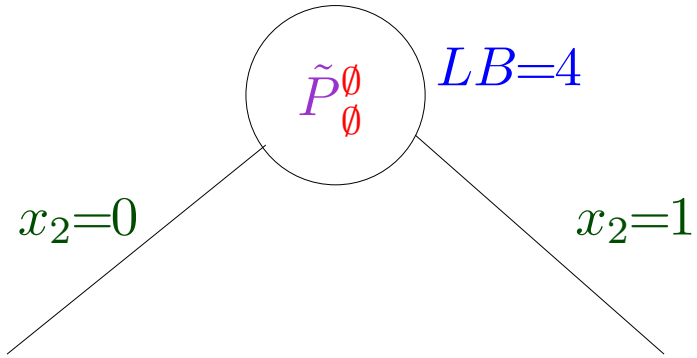
$\mathcal{D},\mathcal{U} \subseteq \{1,\ldots,n\}$ with $\mathcal{D} \cap \mathcal{U} = \emptyset$ (partially relaxed/fixed continuous subproblem)

$$\mathcal{D} \subseteq \bar{\mathcal{D}}, \mathcal{U} \subseteq \bar{\mathcal{U}} \implies OPT(\tilde{P}^{\mathcal{U}}_{\mathcal{D}}) \leq OPT(\tilde{P}^{\bar{\mathcal{U}}}_{\bar{\mathcal{D}}})$$

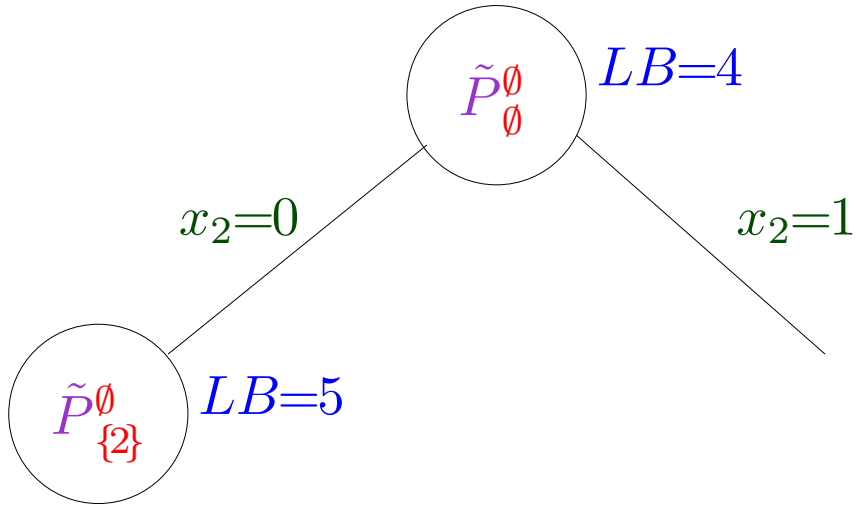
Branch & Bound Procedure



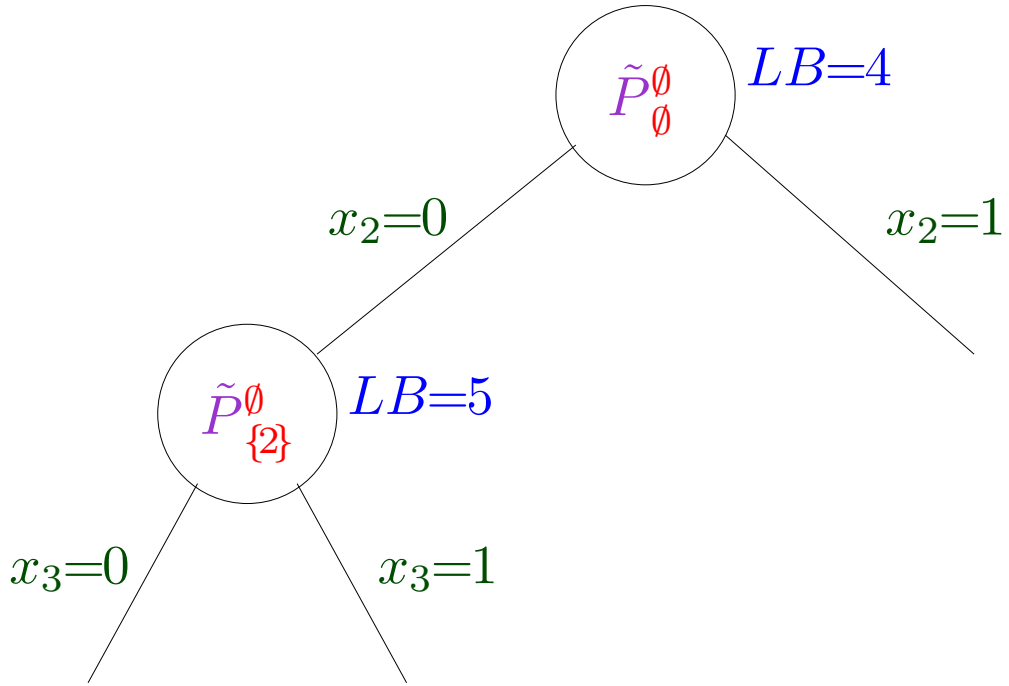
Branch & Bound Procedure



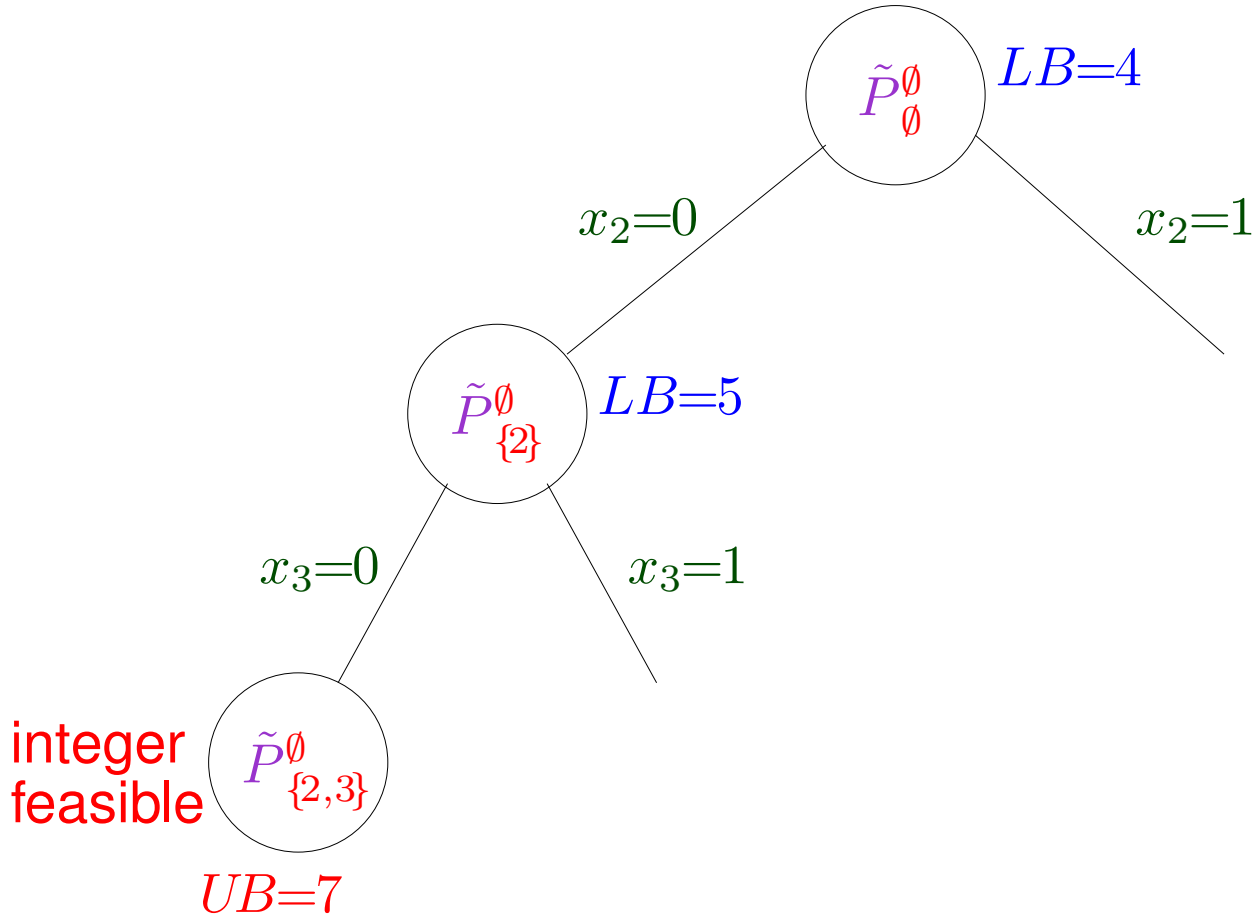
Branch & Bound Procedure



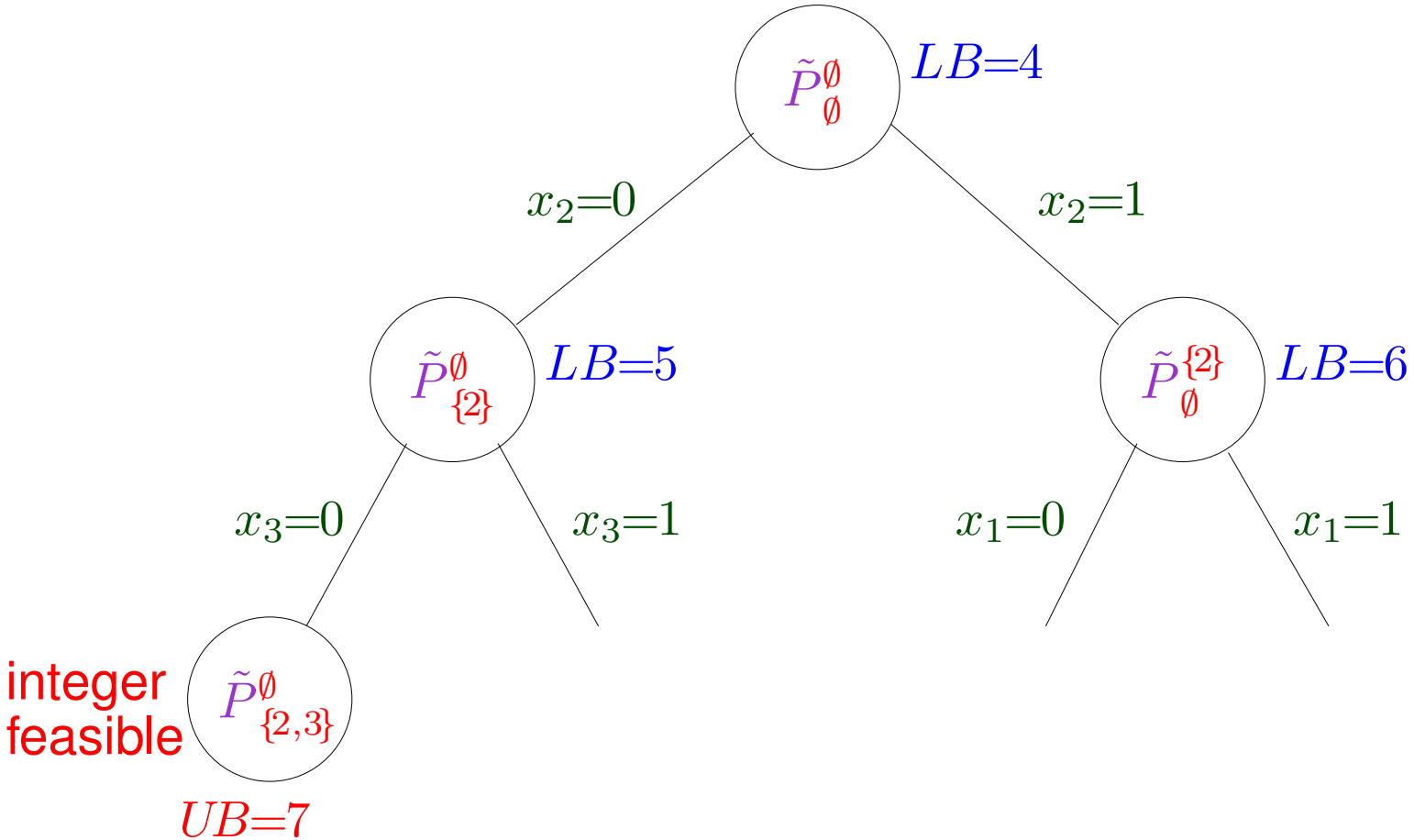
Branch & Bound Procedure



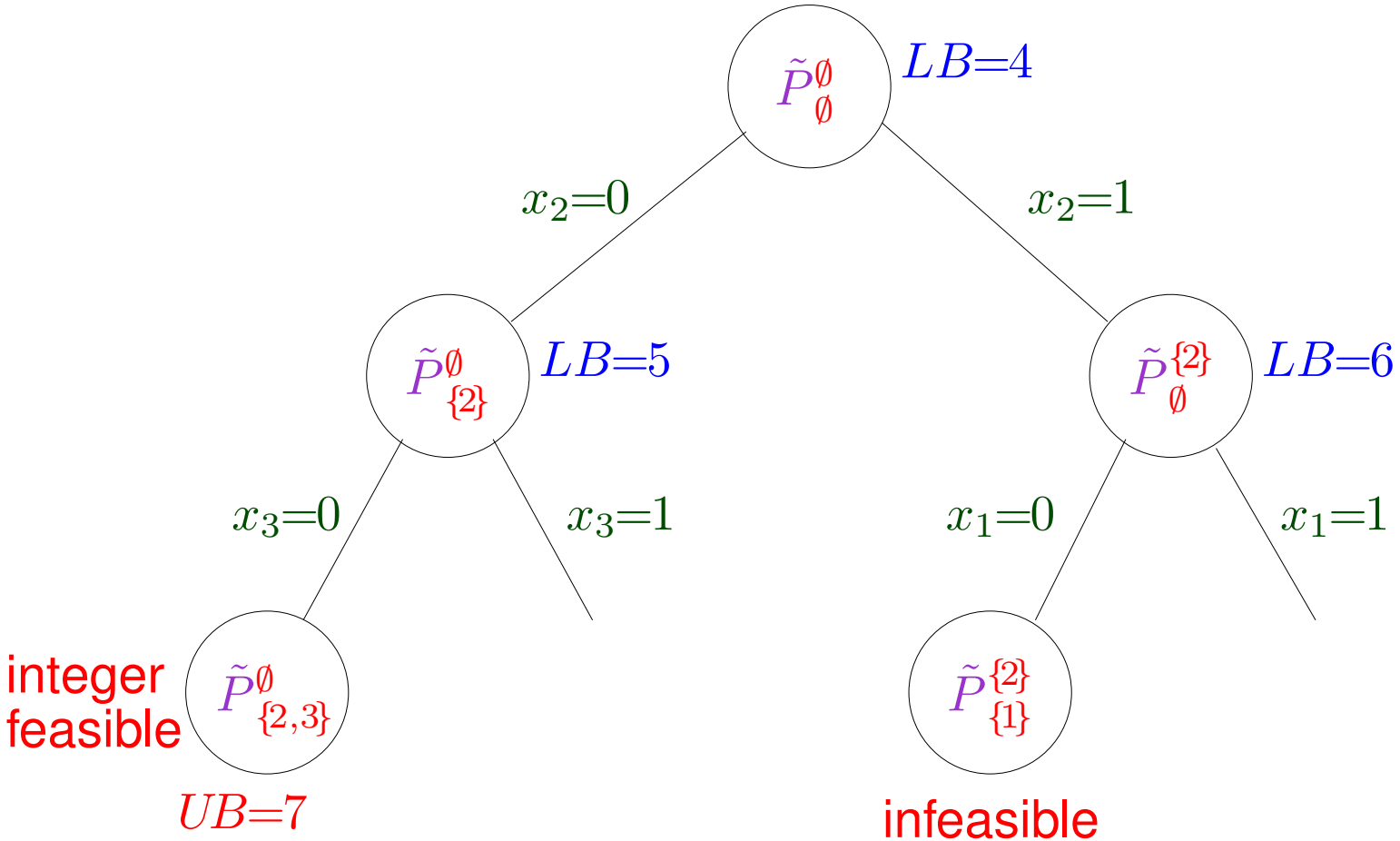
Branch & Bound Procedure



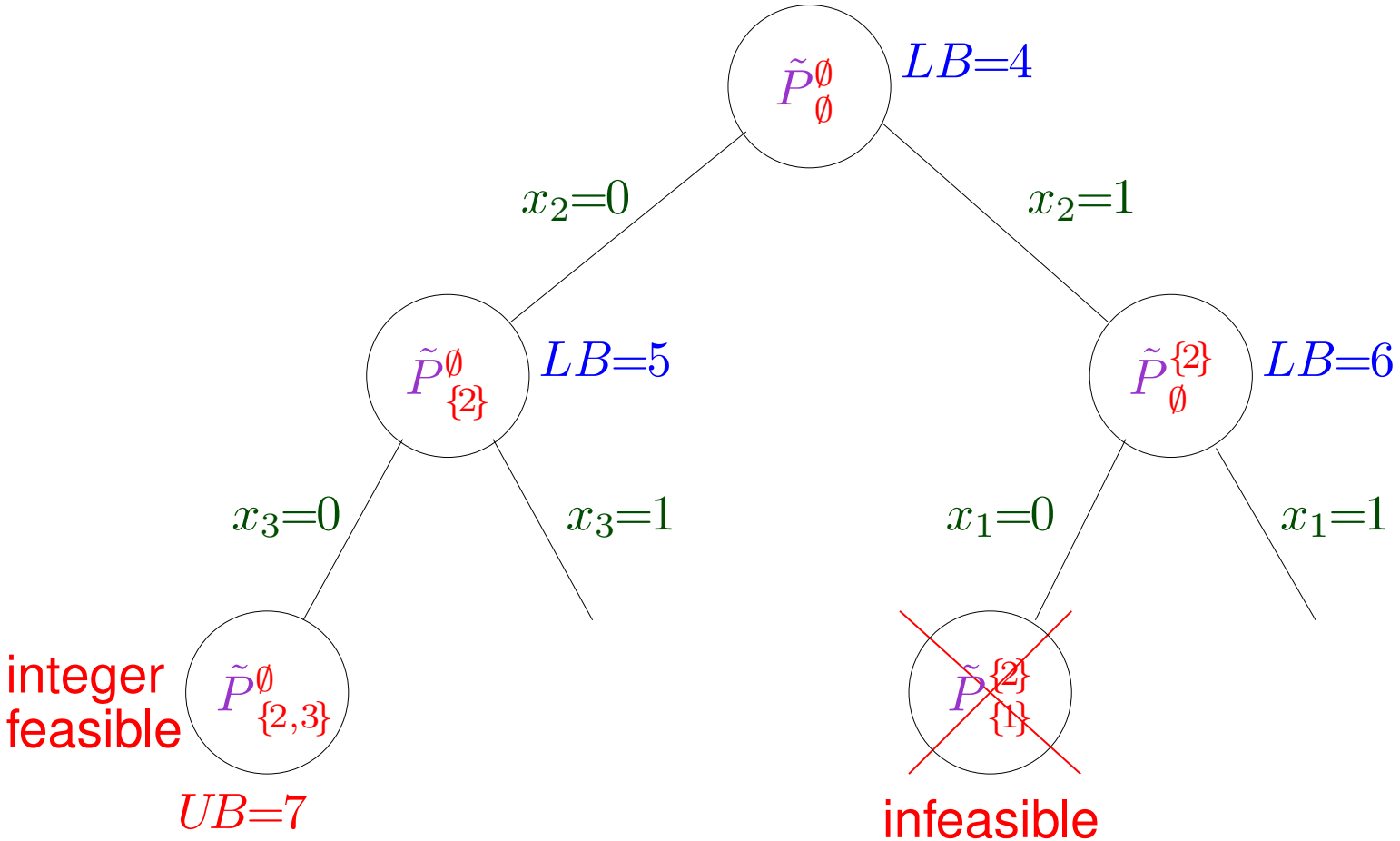
Branch & Bound Procedure



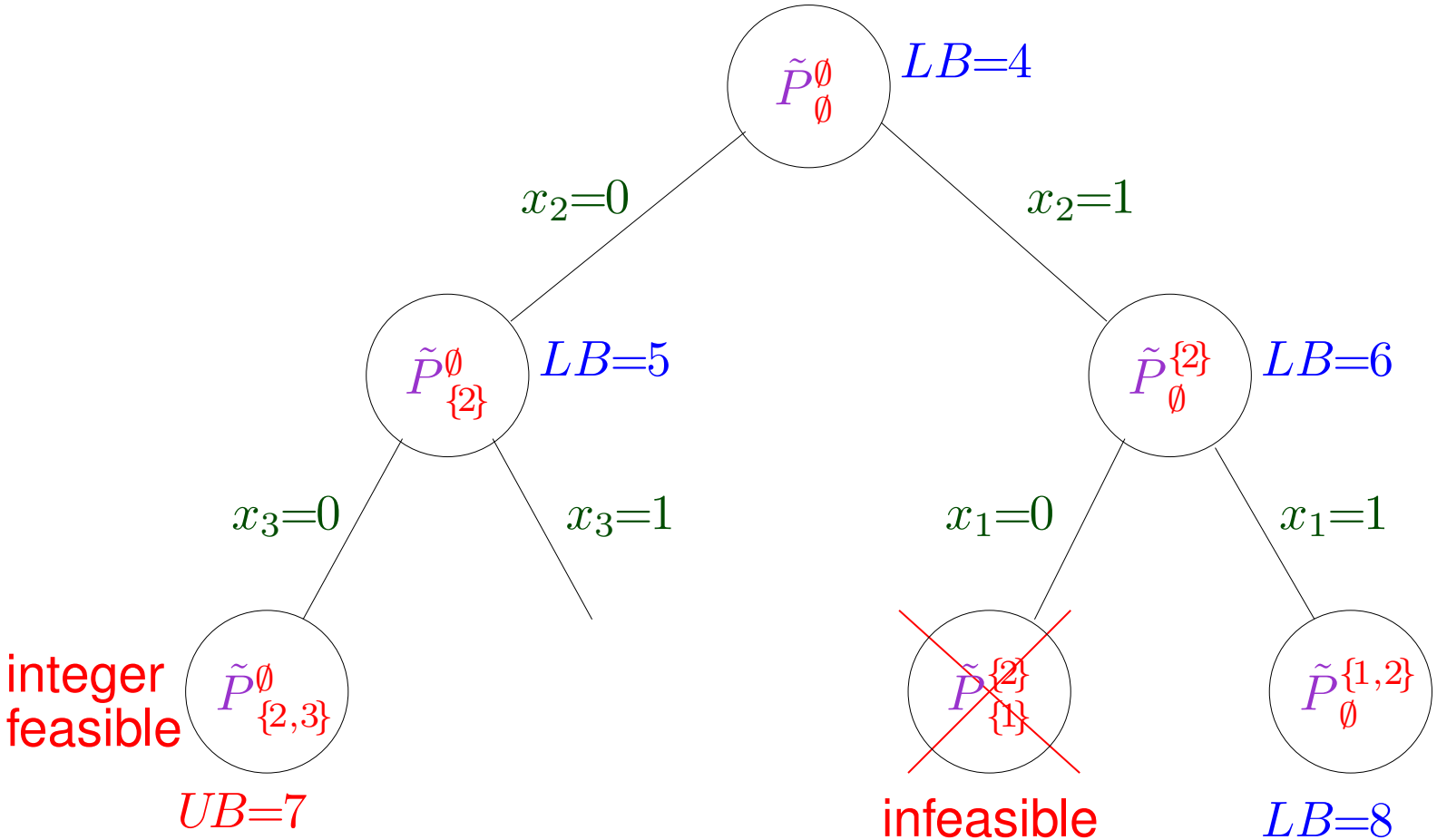
Branch & Bound Procedure



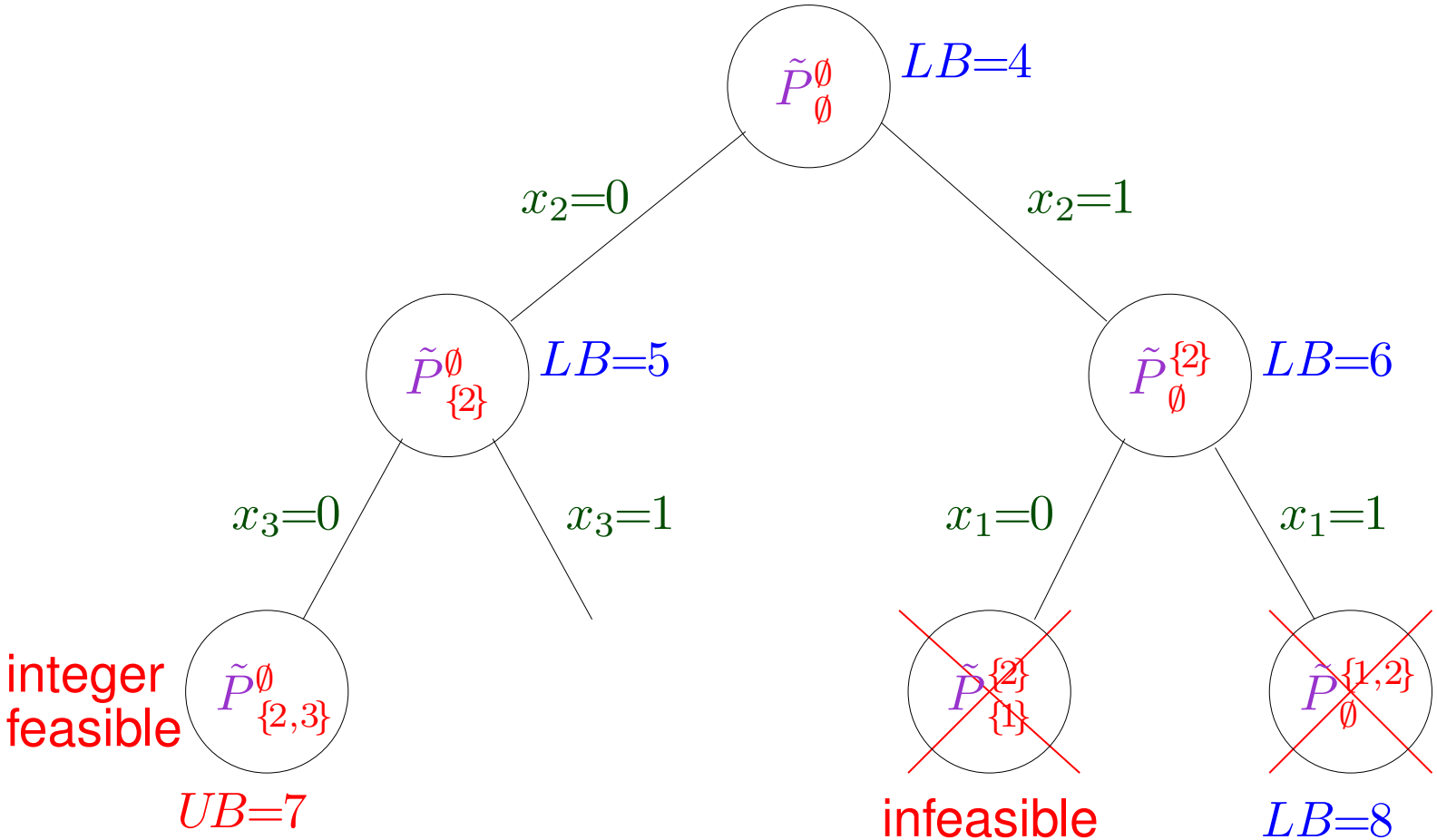
Branch & Bound Procedure



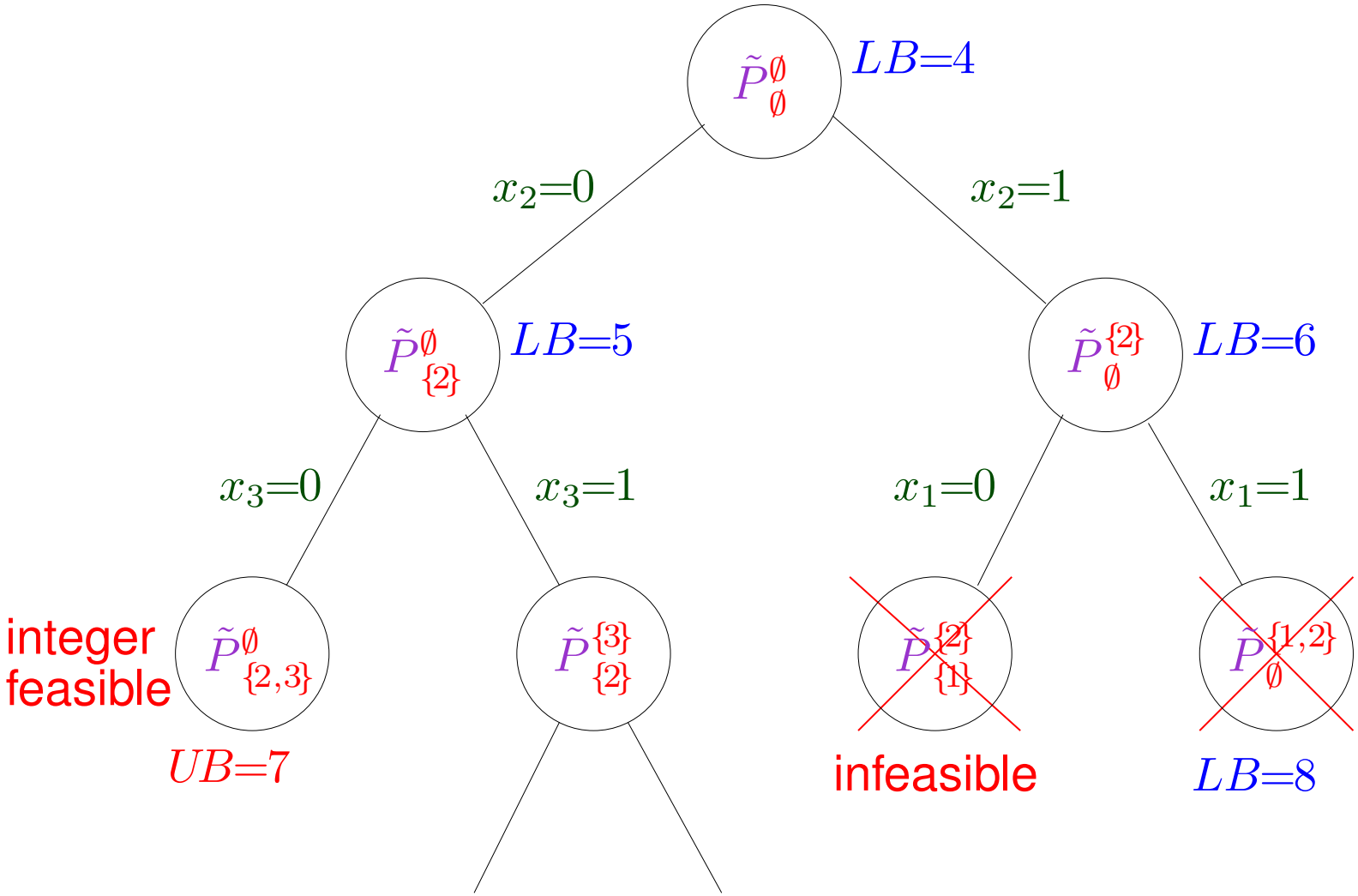
Branch & Bound Procedure



Branch & Bound Procedure



Branch & Bound Procedure



Branch & Bound

- Basic framework for many MILP methods
- For MILP, very efficient solution of node subproblems possible
  - Use optimal solution from parent node
  - Simplex LP algorithm can be hot-started
    (re-use factorization; matrices don't change)

Branch & Bound

- Basic framework for many MILP methods
- For MILP, very efficient solution of node subproblems possible
  - Use optimal solution from parent node
  - Simplex LP algorithm can be hot-started
    (re-use factorization; matrices don't change)
- Different branching rules
  - Most integer-infeasible variable
  - Pseudo-costs (based on average past effect on objective function)
  - Strong branching (later)
  - ...
- Different node selection strategies
  - Depth first (quickly gives upper bounds)
  - Best bound (minimum number of nodes, wide tree)
  - ...

Difficulties in MINLP

- Algorithms in MILP well-developed
- Algorithms in NLP well-developed
- But: Much less research in MINLP

Difficulties in MINLP

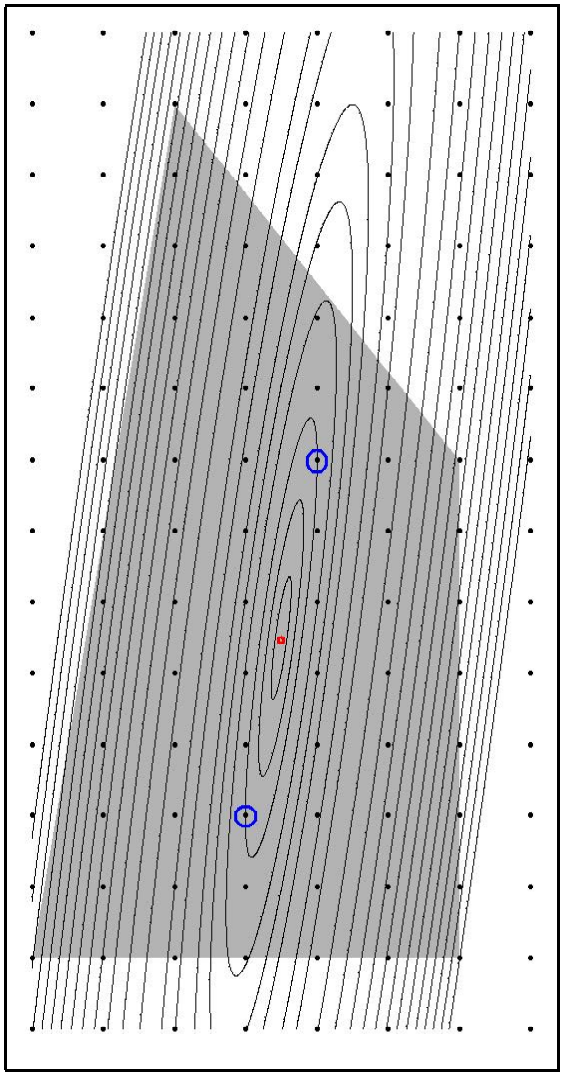
- Algorithms in MILP well-developed
- Algorithms in NLP well-developed
- But: Much less research in MINLP
- Branch & bound framework can be applied to MINLP (as in Fletcher/Leyffer's `MINLP` code)

Difficulties in MINLP

- Algorithms in MILP well-developed
- Algorithms in NLP well-developed
- But: Much less research in MINLP
- Branch & bound framework can be applied to MINLP (as in Fletcher/Leyffer's `MINLP` code)
- Complications:
  - If the problem is nonconvex, local solutions give incorrect bounds
    (global optimization is difficult)

Difficulties in MINLP

- Algorithms in MILP well-developed
- Algorithms in NLP well-developed
- But: Much less research in MINLP
- Branch & bound framework can be applied to MINLP (as in Fletcher/Leyffer's `MINLP` code)
- Complications:
  - If the problem is nonconvex, local solutions give incorrect bounds
    (global optimization is difficult)
  - Even in convex case:
    optimal solution can lie in the interior;
    finding convex hull of integer feasible points not sufficient
  - MINLP solution not obtained by rounding NLP solution
  - NLP solver cannot be hot-started



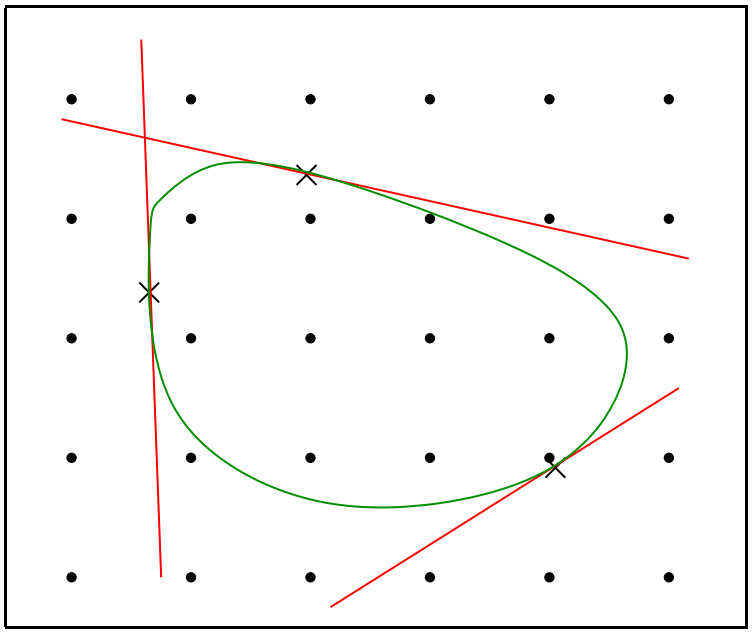
Outer Approximation

$$
\begin{array}{lll}
\min & z & \text{(linear objective)} \\
s.t. & f(x,y) \leq z & \\
 & c(x,y) \leq 0 & \\
 & x \in \{0,1\}^n, y \in \mathbb{R}^p, z \in \mathbb{R} &
\end{array}
$$

Outer Approximation

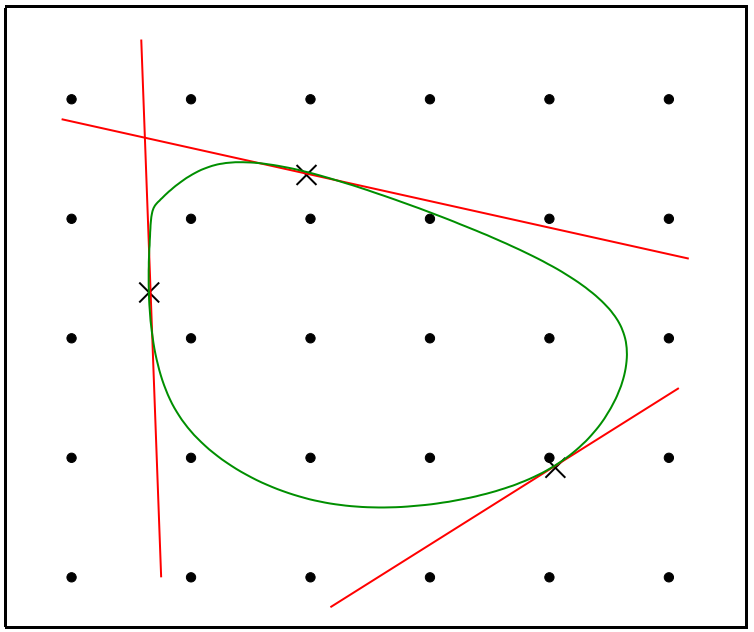
$$
\begin{array}{lll}
\min & z & \text{(linear objective)} \\
s.t. & f(x,y) \leq z & \\
 & c(x,y) \leq 0 & \\
 & x \in \{0,1\}^n, y \in \mathbb{R}^p, z \in \mathbb{R} &
\end{array}
$$

Idea: approximate by MILP (hyperplanes)



Outer Approximation

$$\begin{aligned} \min \quad & z && \text{(linear objective)} \\ s.t. \quad & f(x,y) \le z \\ & c(x,y) \le 0 \\ & x \in \{0,1\}^n, y \in \mathbb{R}^p, z \in \mathbb{R} \end{aligned}$$



Idea: approximate by MILP (hyperplanes)

$$\begin{aligned} \min \quad & z \\ s.t. \quad & \nabla f(x^k,y^k)^T \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} + f(x^k,y^k) \le z \\ & \nabla c(x^k,y^k)^T \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} + c(x^k,y^k) \le 0 \\ & \qquad\qquad \text{for all } (x^k,y^k) \in \mathcal{T} \\ & x \in \{0,1\}^n, y \in \mathbb{R}^p, z \in \mathbb{R} \end{aligned}$$

$\mathcal{T}$ contains linearization points (augmented during algorithm)

Outer Approximation Algorithm

relaxed MINLP

$$
\begin{aligned}
\min \quad & z \\
s.t. \quad & f(x,y) \leq z \\
& c(x,y) \leq 0 \\
& x \in [0,1]^n, y \in \mathbb{R}^p, z \in \mathbb{R}
\end{aligned}
$$

Outer Approximation Algorithm

relaxed MINLP

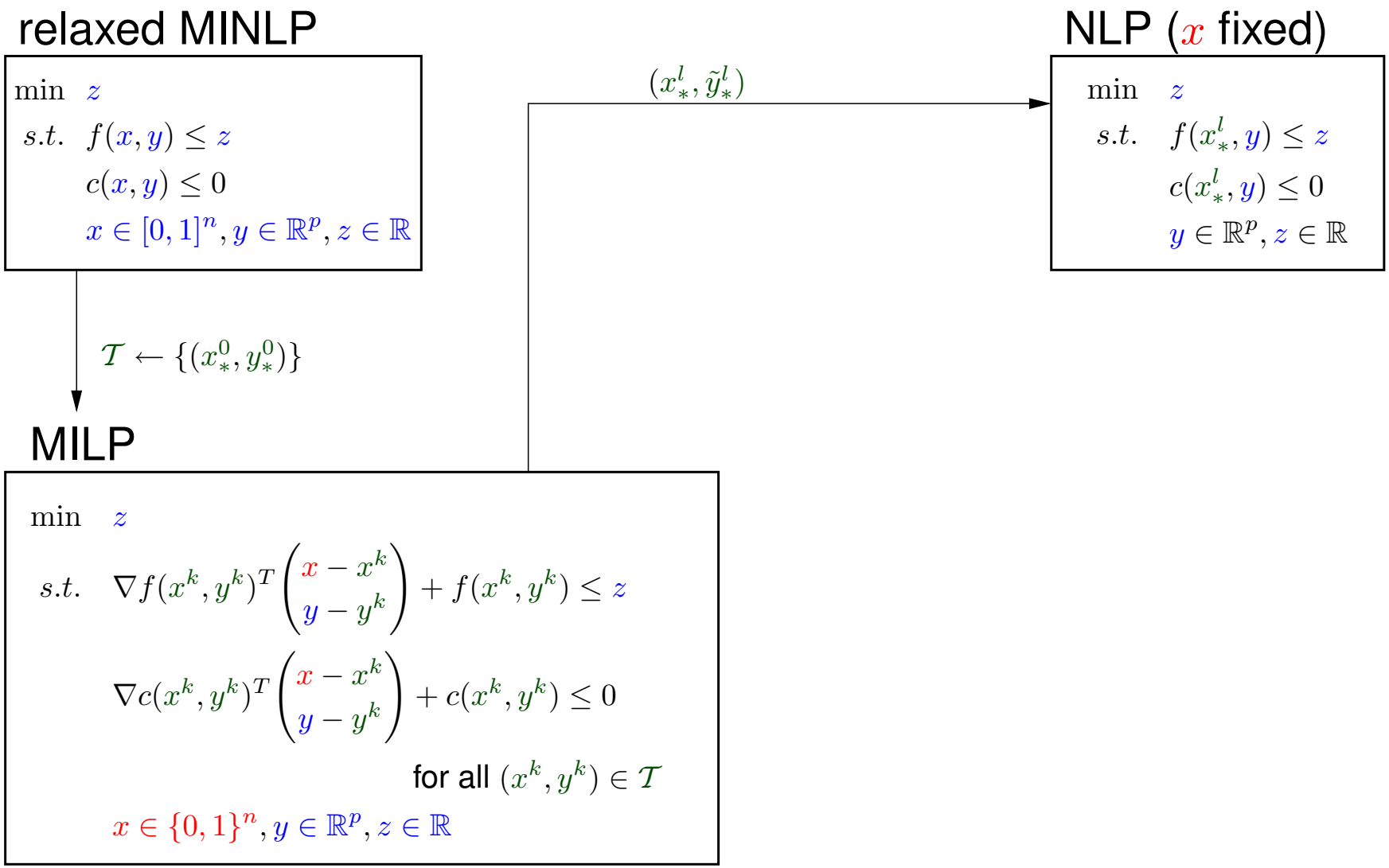
$$
\begin{aligned}
\min \quad & z \\
s.t. \quad & f(x,y) \le z \\
& c(x,y) \le 0 \\
& x \in [0,1]^n, y \in \mathbb{R}^p, z \in \mathbb{R}
\end{aligned}
$$

$\mathcal{T} \leftarrow \{(x^0_*, y^0_*)\}$

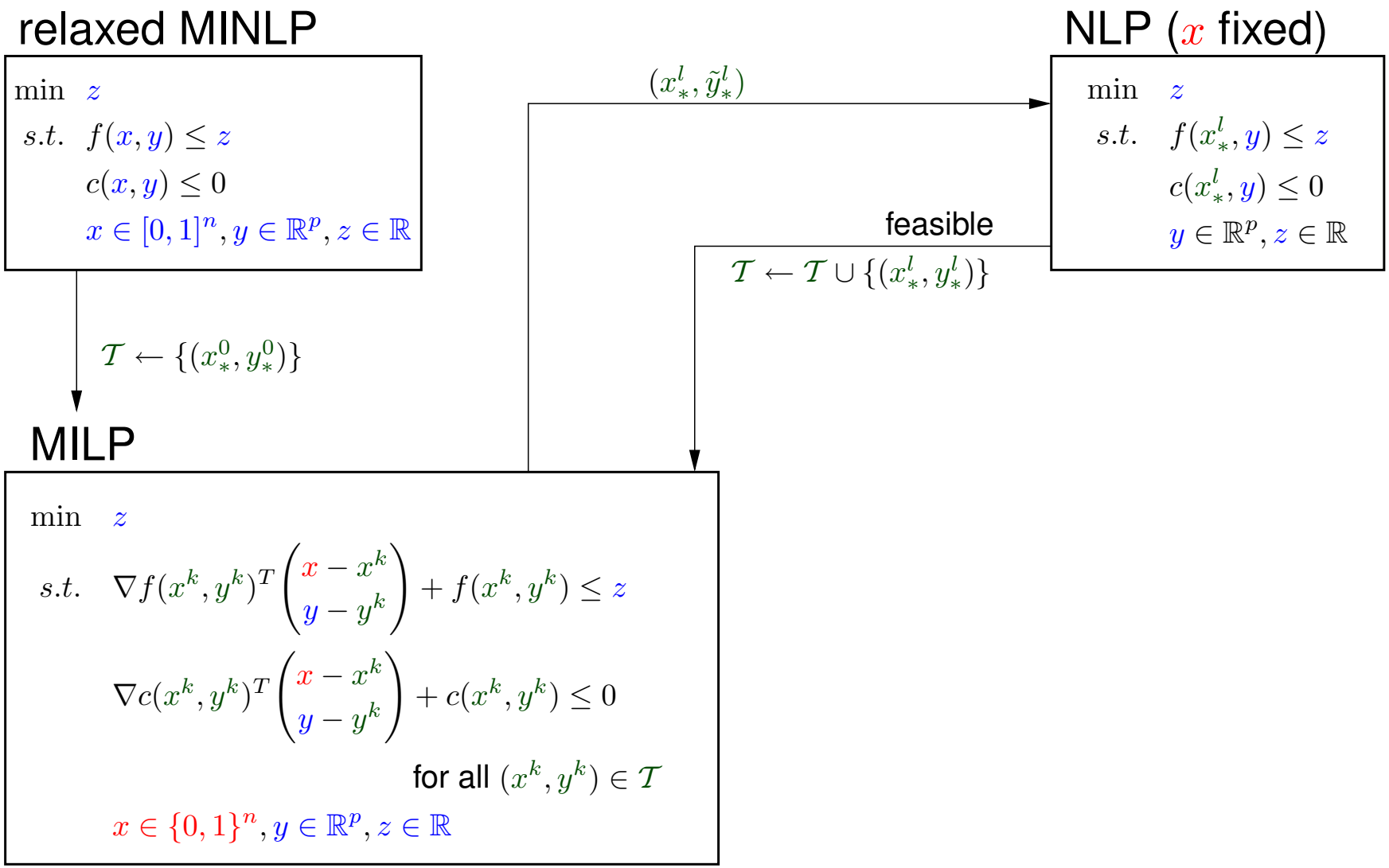
MILP

$$
\begin{aligned}
\min \quad & z \\
s.t. \quad & \nabla f(x^k,y^k)^T \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} + f(x^k,y^k) \le z \\
& \nabla c(x^k,y^k)^T \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} + c(x^k,y^k) \le 0 \\
& \qquad\qquad \text{for all } (x^k,y^k) \in \mathcal{T} \\
& x \in \{0,1\}^n, y \in \mathbb{R}^p, z \in \mathbb{R}
\end{aligned}
$$

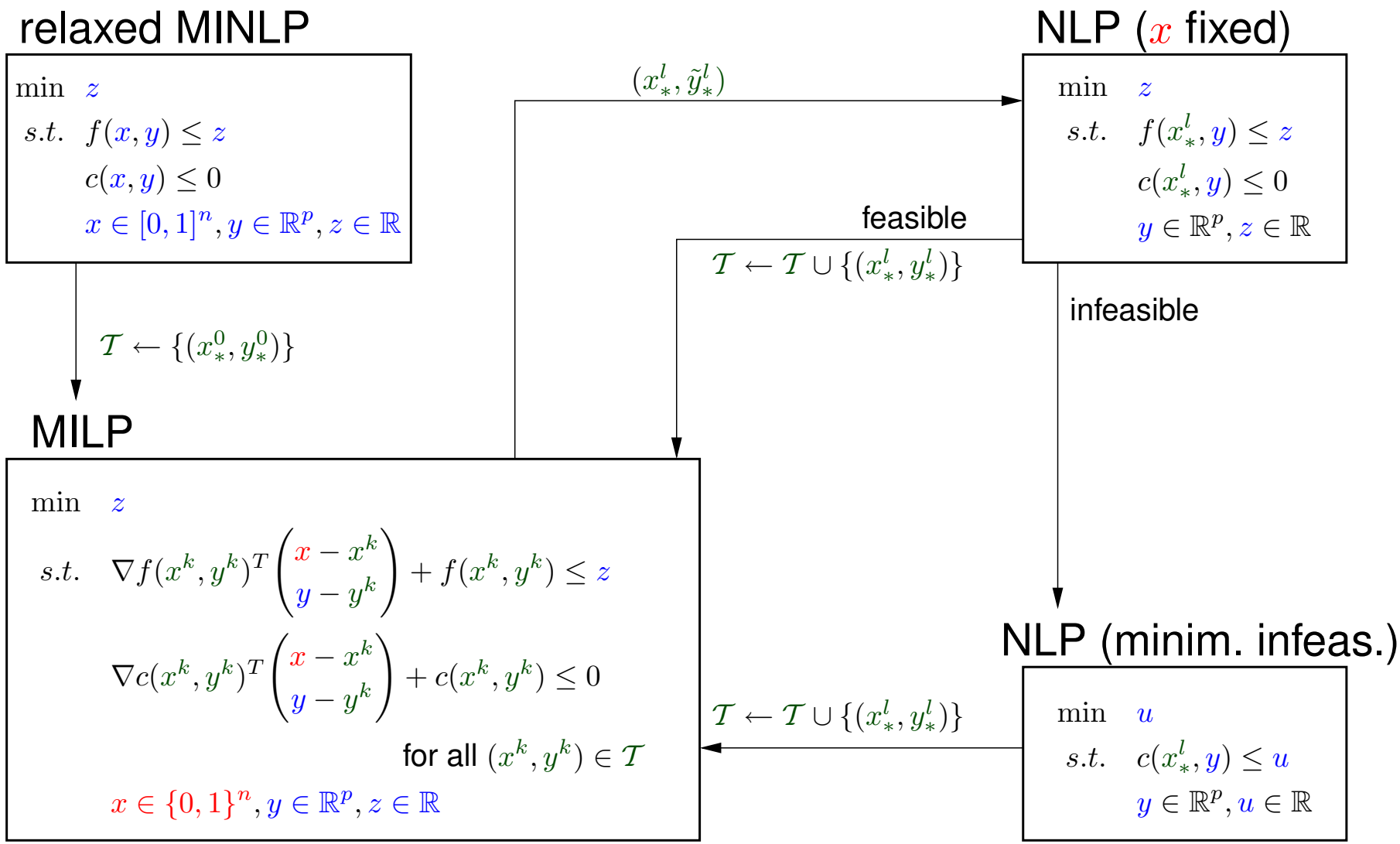
Outer Approximation Algorithm



Outer Approximation Algorithm



Outer Approximation Algorithm



Outer Approximation Discussion

- Alternatingly solve MILPs and NLPs
- Finite termination
  - Finished when MILP returns same integer x_*^l solution twice

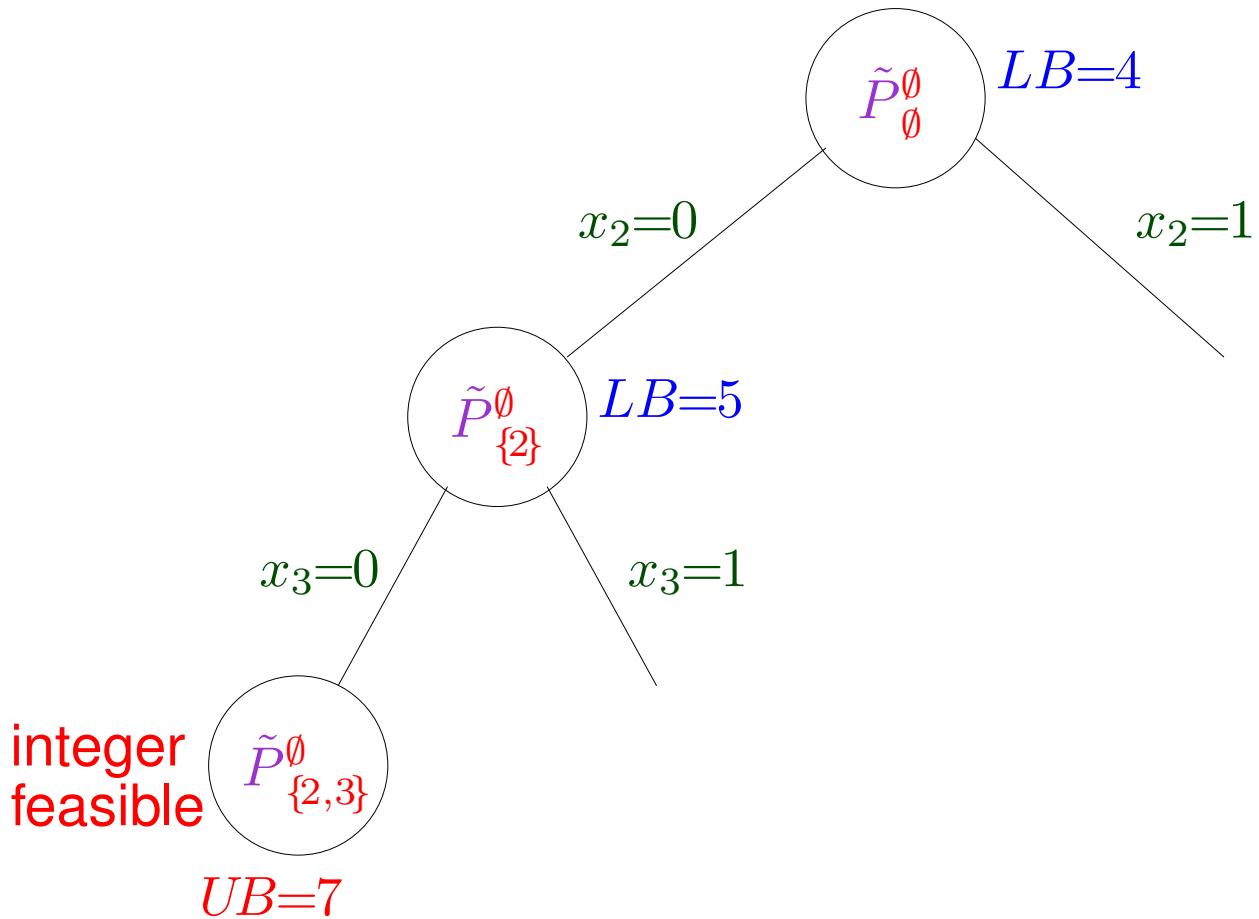
Outer Approximation Discussion

- Alternatingly solve MILPs and NLPs
- Finite termination
  - Finished when MILP returns same integer x_*^l solution twice
- Disadvantage: Need to solve every MILP from scratch

Outer Approximation Discussion

- Alternatingly solve MILPs and NLPs
- Finite termination
  - Finished when MILP returns same integer x_*^l solution twice
- Disadvantage: Need to solve every MILP from scratch
- Improvement (Quesada, Grossmann, 1992):
  - Build only one MILP enumeration tree
  - Solve NLP with fixed x_*^l for every MILP integer feasible solution
  - Then add new cuts to current MILP formulation
    ("branch-and-*cut*" procedure, new cuts added to open nodes)

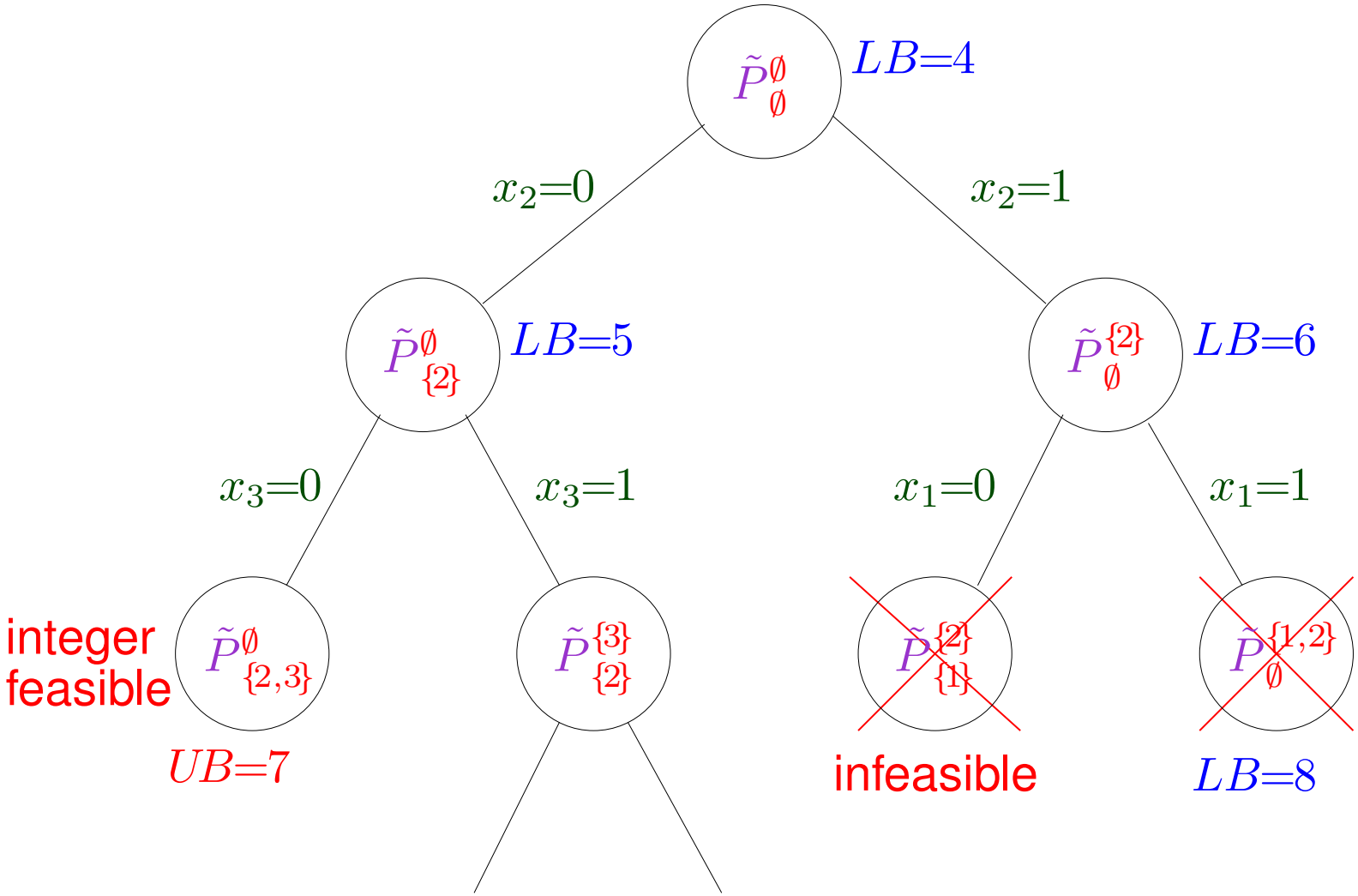
Branch & Bound Procedure



Outer Approximation Discussion

- Alternatingly solve MILPs and NLPs
- Finite termination
  - Finished when MILP returns same integer x_*^l solution twice
- Disadvantage: Need to solve every MILP from scratch
- Improvement (Quesada, Grossmann, 1992):
  - Build only one MILP enumeration tree
  - Solve NLP with fixed x_*^l for every MILP integer feasible solution
  - Then add new cuts to current MILP formulation
    ("branch and cut" procedure, new cuts added to open nodes)
- A "hybrid" approach:
  - Solve NLPs also at nodes where only some x_i are integer
  - For example, solve partially relaxed MINLP every 10th MILP node
    - Includes information about nonlinear geometry more quickly
    - Requires solution of more NLPs
  - Includes outer approximation and branch & bound as two extremes

Branch & Bound Procedure



Bonmin

- Basic Open-source Nonlinear Mixed-INteger programming
- Implementation of MINLP algorithms
- Available on COIN-OR:
  https://projects.coin-or.org/Bonmin
- Written in C++; can be used from AMPL, GAMS, and C++ code

Bonmin

- Basic Open-source Nonlinear Mixed-INteger programming
- Implementation of MINLP algorithms
- Available on COIN-OR:
  https://projects.coin-or.org/Bonmin
- Written in C++; can be used from AMPL, GAMS, and C++ code
- Uses COIN-OR components:
  - `Bcp`: Branch, cut and price (MILP)
  - `Cbc`: Coin branch and cut (MILP)
  - `Clp`: Coin LP solver
  - `Cgl`: Coin cut generation library
  - `Ipopt`: Interior point NLP solver
    - primal-dual warm start option
    - "expect infeasible problem" option
    - adaptive barrier parameter strategy used
  - Essential for fast development: availability of open source

Bonmin

- Basic Open-source Nonlinear Mixed-INteger programming
- Implementation of MINLP algorithms
- Available on COIN-OR:
  https://projects.coin-or.org/Bonmin
- Written in C++; can be used from AMPL, GAMS, and C++ code
- Uses COIN-OR components:
  - Bcp: Branch, cut and price (MILP)
  - Cbc: Coin branch and cut (MILP)
  - Clp: Coin LP solver
  - Cgl: Coin cut generation library
  - Ipopt: Interior point NLP solver
    - primal-dual warm start option
    - "expect infeasible problem" option
    - adaptive barrier parameter strategy used
  - Essential for fast development: availability of open source
- Can also use Cplex as (MI)LP solver, FilterSQP as NLP solver

I-BB

- Branch and bound for MINLP
- For numerical experiments (other options implemented):
  - Branching rule: Most integer-infeasible variable
  - Search strategy: Best bound
  - Use root node solution as warm start point in nodes
    - using parent node solution didn't seem to pay off (with `Ipopt`)
    - storing parent node solutions requires a lot of memory
- If NLP solver fails
  - Try different starting point
  - Continue branching

I-OA, I-QG, and I-Hyb

- I-OA
  - Standard outer approximation algorithm
  - MILP solver: I-OA: `Cbc` ; I-OA_Cpx: `Cplex`
- I-QG
  - Quesada-Grossmann algorithm
- I-Hyb
  - Hybrid algorithm
  - Start with I-OA algorithm for 30 secs
  - Solve NLP every 10th node and at MILP integer feasible nodes

I-OA, I-QG, and I-Hyb

- I-OA
  - Standard outer approximation algorithm
  - MILP solver: I-OA: `Cbc` ; I-OA_Cpx: `Cplex`
- I-QG
  - Quesada-Grossmann algorithm
- I-Hyb
  - Hybrid algorithm
  - Start with I-OA algorithm for 30 secs
  - Solve NLP every 10th node and at MILP integer feasible nodes
- Branching rule: pseudo costs with strong branching initialization
- Use cuts from `Cgl` to strengthen MILP formulation
  - Gomory, Cover, Mixed-integer rounding cuts
- Use root node solution as warm start point in NLP nodes

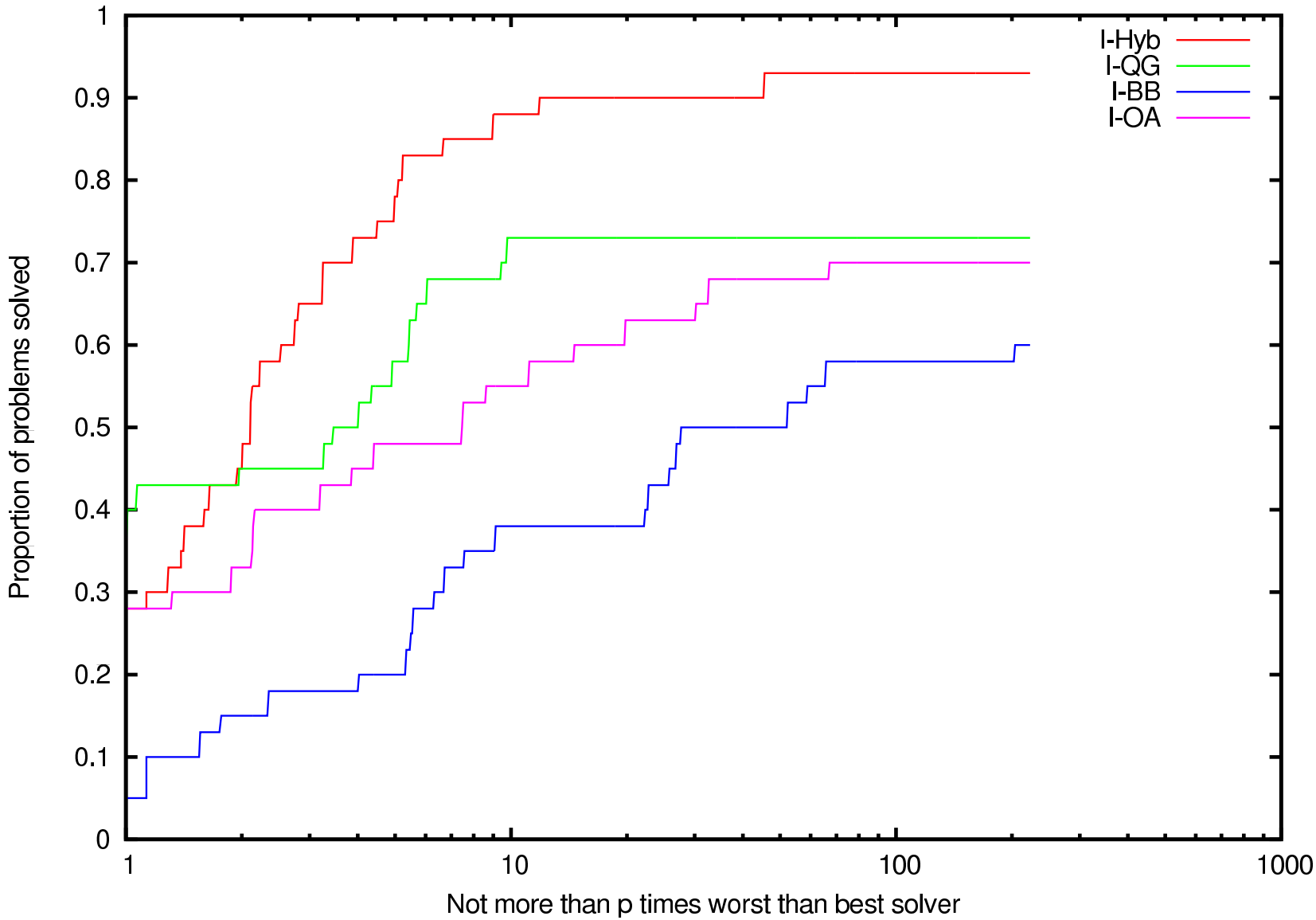
MINLP Test Set

- A library of currently 152 convex MINLPs
  - Existing problem from literature (layout and trimloss problems) (T. Westerlund et al.)
  - Disjunctive problems from chemical engineering formulated both with big-M and convex-hull formulation (N. Sawaya)
  - Water network problems (C. Laird) [many cont. vars]
- Available at `http://egon.cheme.cmu.edu/ibm/page.htm`

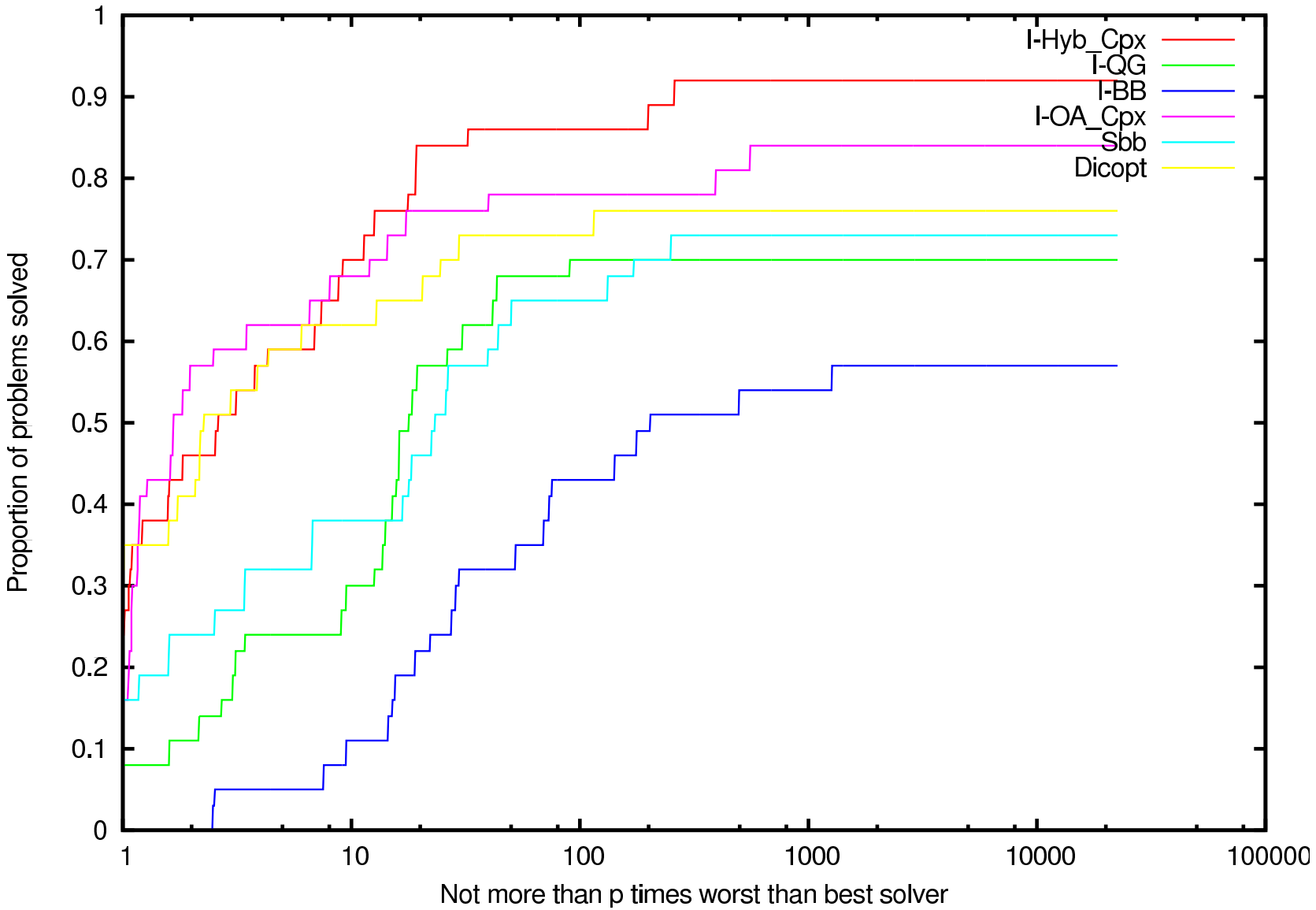
MINLP Test Set

- A library of currently 152 convex MINLPs
  - Existing problem from literature (layout and trimloss problems) (T. Westerlund et al.)
  - Disjunctive problems from chemical engineering formulated both with big-M and convex-hull formulation (N. Sawaya)
  - Water network problems (C. Laird) [many cont. vars]
- Available at `http://egon.cheme.cmu.edu/ibm/page.htm`
- Numerical results for 38 problems (with `Ipopt`)
  - # cont. var: 33–2720 # disc. var: 18–576 # constr: 65–4980
  - 3 hrs. CPU time limit
- Comparison of I-BB, I-OA, I-Hyb with commercial solvers
  - `SBB` (Arki Consulting & Development): Nonlinear branch and bound
    NLPs solved with `Conopt`
  - `Dicopt` (GAMS Corporation): Outer approximation
    MILPs solved with `Cplex`, NLPs solved with `Conopt`

Comparing Bonmin *Algorithms*



Comparing All Solvers



Numerical Results

- I-Hyb is most robust algorithm within time limit
- Only I-Hyb can solve `trimloss4` in 3 hrs.
- I-Hyb almost always faster than I-BB and I-OA
- I-Hyb has usually smaller enumeration tree than I-BB
- Time in `Ipopt` below 5% for I-OA, can be dominant part for I-Hyb
- Average time for NLP node: `SBB` 0.27 secs, I-BB 0.21 secs
  - Using interior point solver does not necessarily seem to be disadvantageous
- Wall clock time for `SBB` is much larger than CPU time (lots of I/O)

Improving Branch and Bound

Motivation:

- Some problems are solved better by I-BB than I-Hyb
  (Pierre's portfolio optimization)

Improving Branch and Bound

Motivation:

- Some problems are solved better by I-BB than I-Hyb
  (Pierre's portfolio optimization)
- Outer-approximation based algorithms difficult to modify for *nonconvex* problems
  - Linearization may cut off feasible points
  - I-BB still exact for pseudo-convex problems

Improving Branch and Bound

Motivation:

- Some problems are solved better by I-BB than I-Hyb
  (Pierre's portfolio optimization)
- Outer-approximation based algorithms difficult to modify for *nonconvex* problems
  - Linearization may cut off feasible points
  - I-BB still exact for pseudo-convex problems
- Simple I-BB heuristics for nonconvex MINLP:
  - Solve nodes for different, randomly generated starting points
  - Negative cut-off value (continue even if lower bound is larger than current best integer feasible solution)
  - Continue branching even if node is infeasible

Explore different branching rules

Most-Fractional:

- Choose most integer-infeasible variable x_i

Explore different branching rules

Most-Fractional:

- Choose most integer-infeasible variable x_i

Strong Branching:

- Get list of k possible candidates $x_i \in (0,1)$
  (e.g., k most integer-infeasible)
- For each variable x_i in list:
  - Solve corresponding relaxed MINLPs $\tilde{P}_{\mathcal{D}_i}^{\mathcal{U}_i}$ (approximately)
    for $x_i = 0$ and $x_i = 1$
  - Obtain (estimate of) change in objective function
- Choose most promising candidate
  - Most change in objective function (weighted)
  - (Most integer feasible non-fixed variables)
  - If NLP for one candidate proven to be infeasible, pick this candidate.

How to solve strong branching problems

- Solve NLP $\tilde{P}^{\mathcal{U}_i}_{\mathcal{D}_i}$ for fixed candidates
  - Analog to LP strong branching (expensive; no hot-start possible)

How to solve strong branching problems

- Solve NLP $\tilde{P}^{\mathcal{U}_i}_{\mathcal{D}_i}$ for fixed candidates
  - Analog to LP strong branching (expensive; no hot-start possible)
- Solve QP approximation
  - Use QP of last SQP iteration in node (potential for hot-start?)

How to solve strong branching problems

- Solve NLP $\tilde{P}^{\mathcal{U}_i}_{\mathcal{D}_i}$ for fixed candidates
  - Analog to LP strong branching (expensive; no hot-start possible)
- Solve QP approximation
  - Use QP of last SQP iteration in node (potential for hot-start?)
- Solve LP approximation
  - Linearize each function at solution (use LP solver's hot-start)

How to solve strong branching problems

- Solve NLP $\tilde{P}^{\mathcal{U}_i}_{\mathcal{D}_i}$ for fixed candidates
  - Analog to LP strong branching (expensive; no hot-start possible)
- Solve QP approximation
  - Use QP of last SQP iteration in node (potential for hot-start?)
- Solve LP approximation
  - Linearize each function at solution (use LP solver's hot-start)
- Solve improved LP approximation
  - Extended Cutting Plane method:
    Collect linearizations at a number of points (e.g., 5)
  - Use LP hot-starts

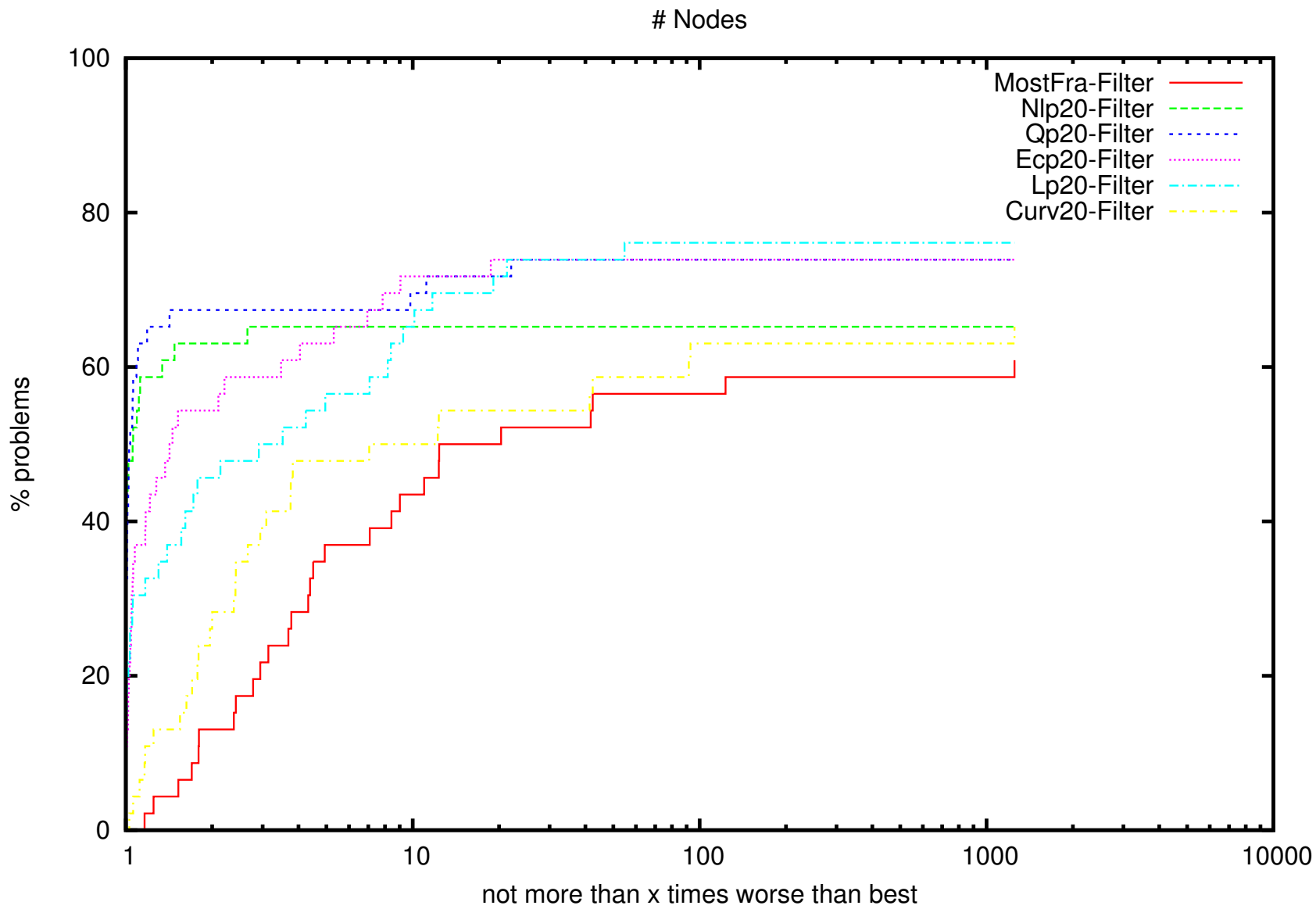
How to solve strong branching problems

- Solve NLP $\tilde{P}^{\mathcal{U}_i}_{\mathcal{D}_i}$ for fixed candidates
  - Analog to LP strong branching (expensive; no hot-start possible)
- Solve QP approximation
  - Use QP of last SQP iteration in node (potential for hot-start?)
- Solve LP approximation
  - Linearize each function at solution (use LP solver's hot-start)
- Solve improved LP approximation
  - Extended Cutting Plane method:
    Collect linearizations at a number of points (e.g., 5)
  - Use LP hot-starts
- Curvature estimator
  - Guess activities; use EQP model to predict objective changes
  - Difficult to guess QP activities, but very cheap

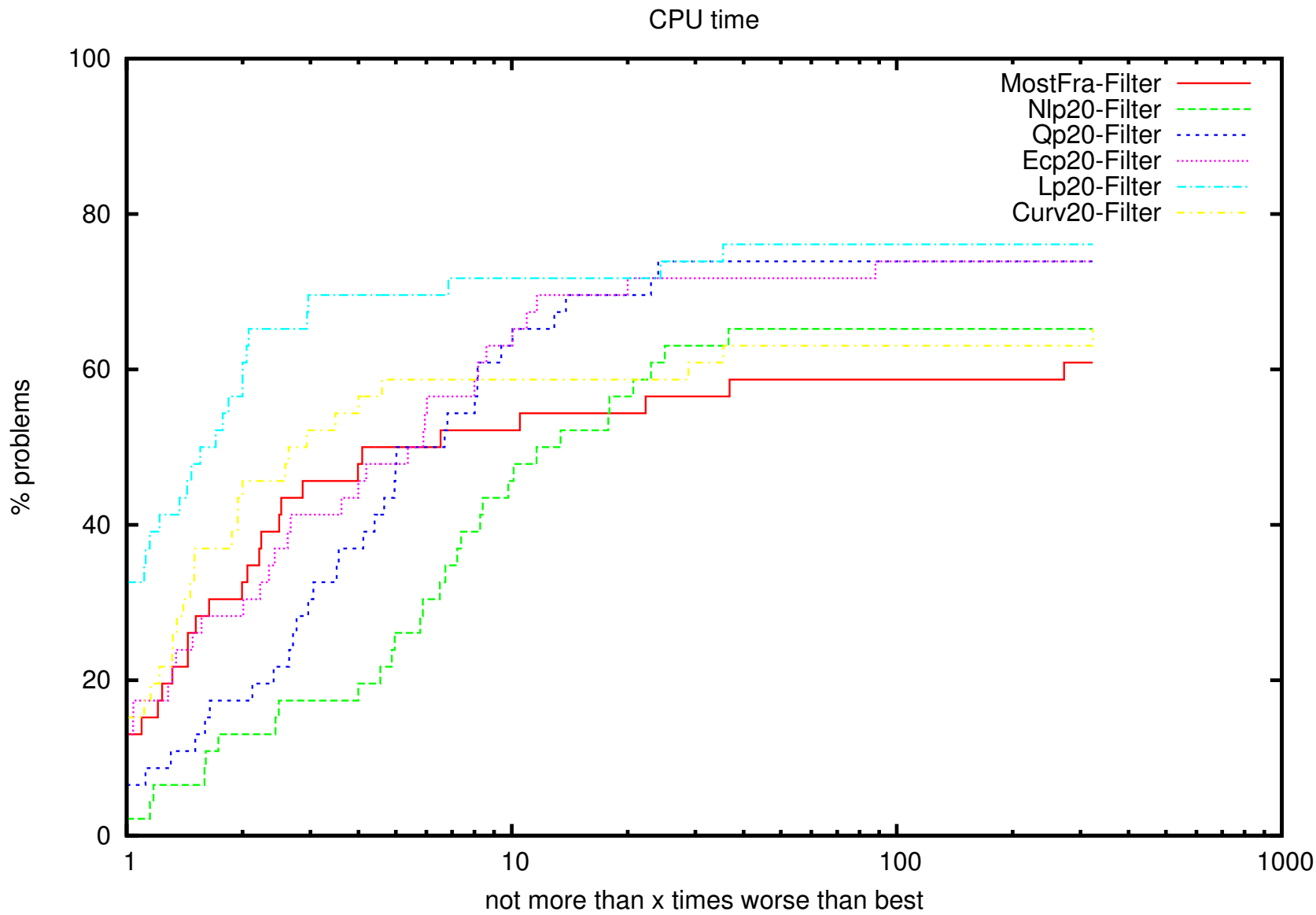
How to solve strong branching problems

- Solve NLP $\tilde{P}^{\mathcal{U}_i}_{\mathcal{D}_i}$ for fixed candidates
  - Analog to LP strong branching (expensive; no hot-start possible)
- Solve QP approximation
  - Use QP of last SQP iteration in node (potential for hot-start?)
- Solve LP approximation
  - Linearize each function at solution (use LP solver's hot-start)
- Solve improved LP approximation
  - Extended Cutting Plane method:
    Collect linearizations at a number of points (e.g., 5)
  - Use LP hot-starts
- Curvature estimator
  - Guess activities; use EQP model to predict objective changes
  - Difficult to guess QP activities, but very cheap
- Comparing options for 50 difficult problems (using `FilterSQP`)

Comparison Branching Options (Nodes)



Comparison Branching Options (CPU*)



Related And Future Work For Bonmin

- Nonlinear feasibility pump (Bonami, Cornuéjols, Lodi, Margot)
  - Heuristic for quickly finding MINLP integer-feasible solutions
  - Able to find feasible solutions not found by other method (e.g., problems trimloss5-12)

Related And Future Work For Bonmin

- Nonlinear feasibility pump (Bonami, Cornuéjols, Lodi, Margot)
  - Heuristic for quickly finding MINLP integer-feasible solutions
  - Able to find feasible solutions not found by other method
    (e.g., problems trimloss5-12)

- Parallel implementation
  - Using COIN-OR's `Bcp` (Ladanyi)
  - So far, implementation of I-BB on BlueGene
  - For `UFLX60_180` (11040 cont., 60 bin. vars; 11160 constr;
    about 1,000,000 nodes; using IPOPT):
    - (almost) linear speed-up for up to 2048 CPUs
  - Not I/O bound (works well on Linux cluster)

Related And Future Work For Bonmin

- Nonlinear feasibility pump (Bonami, Cornuéjols, Lodi, Margot)
  - Heuristic for quickly finding MINLP integer-feasible solutions
  - Able to find feasible solutions not found by other method
    (e.g., problems trimloss5-12)

- Parallel implementation
  - Using COIN-OR's `Bcp` (Ladanyi)
  - So far, implementation of I-BB on BlueGene
  - For `UFLX60_180` (11040 cont., 60 bin. vars; 11160 constr;
    about 1,000,000 nodes; using IPOPT):
    - (almost) linear speed-up for up to 2048 CPUs
  - Not I/O bound (works well on Linux cluster)

- Research for nonconvex MINLP
  - P. Belotti working on implementation of automatic convexification
  - Will be used in BONMIN for global optimization

What Did We Learn?

Basic Tools For Nonlinear Optimization

- Newton's method
  - works with local model of problem (based on Taylor expansion)
  - provides fast local convergence

Basic Tools For Nonlinear Optimization

- Newton's method
  - works with local model of problem (based on Taylor expansion)
  - provides fast local convergence
- Line search
  - guarantees global convergence
  - helps to avoid convergence to non-minimizers
  - need to ensure descent properties of the steps
  - one option to measure acceptability of trial points are filter methods
  - an alternative to line search methods are trust region methods (not covered)

Basic Tools For Nonlinear Optimization

- Newton's method
  - works with local model of problem (based on Taylor expansion)
  - provides fast local convergence
- Line search
  - guarantees global convergence
  - helps to avoid convergence to non-minimizers
  - need to ensure descent properties of the steps
  - one option to measure acceptability of trial points are filter methods
  - an alternative to line search methods are trust region methods (not covered)
- Handling inequality constraints
  - (active set) SQP methods; have trouble for large problems
  - interior point methods overcome combinatorial complexity
  - primal-dual methods combine advantages of Newton's method and globalization schemes

Summary continued

- There are important applications for NLP
  - circuit tuning
  - many other design problems
  - control applications (cancer treatment)

Summary continued

- There are important applications for NLP
  - circuit tuning
  - many other design problems
  - control applications (cancer treatment)
- Mixed Integer Nonlinear Programming
  - a new attractive research area
  - different approaches: Branch & bound, outer approximation
  - often it is good to combine advantages of different methods
    (hybrid approach in BONMIN)
  - lots of research opportunities
    - improve branching decisions
    - generate good (nonlinear?) cuts
    - include global optimization

Summary continued

- There are important applications for NLP
  - circuit tuning
  - many other design problems
  - control applications (cancer treatment)
- Mixed Integer Nonlinear Programming
  - a new attractive research area
  - different approaches: Branch & bound, outer approximation
  - often it is good to combine advantages of different methods
    (hybrid approach in BONMIN)
  - lots of research opportunities
    - improve branching decisions
    - generate good (nonlinear?) cuts
    - include global optimization
- No reason to get bored yet. . .

Literature

- Books on nonlinear optimization
  - J. Nocedal and S. J. Wright: *Numerical Optimization (2nd edition)*
  - R. Fletcher: *Practical Method of Optimization*
  - S. G. Nash and A. Sofer: *Linear and Nonlinear Programming*
- Websites with optimization codes:
  - COIN-OR: `http://www.coin-or.org`
  - NEOS Guide: `http://www-fp.mcs.anl.gov/otc/Guide`
- IPOPT paper:
  - A. Wächter and L. T. Biegler: *On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming*
- BONMIN paper:
  - P. Bonami *et al.*: *An Algorithmic Framework For Convex Mixed Integer Nonlinear Programs*