Zero-Order, Black-Box, Derivative-Free, and Simulation-Based Optimization

Stefan Wild

Argonne National Laboratory
Mathematics and Computer Science Division

August 5, 2016

The Plan

Motivation

Black-Box Optimization

- Direct Search Methods
- Model-Based Methods
- Some Global Optimization

Simulation-Based Optimization and Structure

- NLS=Nonlinear Least Squares
- CNO=Composite Nonsmooth Optimization
- SKP=Some Known Partials
- SCO=Simulation-Constrained Optimization

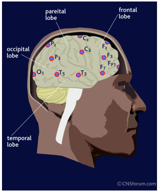
Simulation-Based Optimization

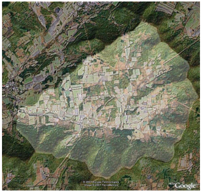
$$\min_{x \in \mathbb{R}^n} \{ f(x) = F[x, S(x)] : c_I[x, S(x)] \le 0,\ c_E[x, S(x)] = 0 \}$$

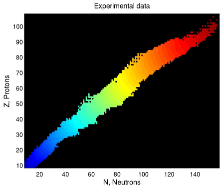
- S (numerical) simulation output, often "noisy" (even when deterministic)
- Derivatives $\nabla_x S$ often unavailable or prohibitively expensive to obtain/approximate directly
- S can contribute to objective and/or constraints
- Single evaluation of S could take seconds/minutes/hours/days

Evaluation is a bottleneck for optimization

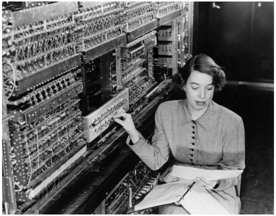
Functions of complex numerical simulations arise everywhere







Computing is Responsible for Pervasiveness of Simulations in Sci&Eng

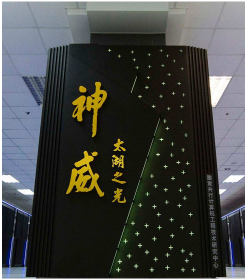


Argonne's AVIDAC

(1953: vacuum tubes)

Argonne's Blue Gene/Q (2012: 786,432 cores)

Currently 6th fastest in the world



Sunway TaihuLight (2016: 11M cores)

Currently fastest in the world

- Parallel/multi-core environments increasingly common
  - Small clusters/multi-core desktops/multi-core laptops pervasive
  - Leadership class machines increasingly parallel
- Simulations (the "forward problem") become faster/more realistic/more complex

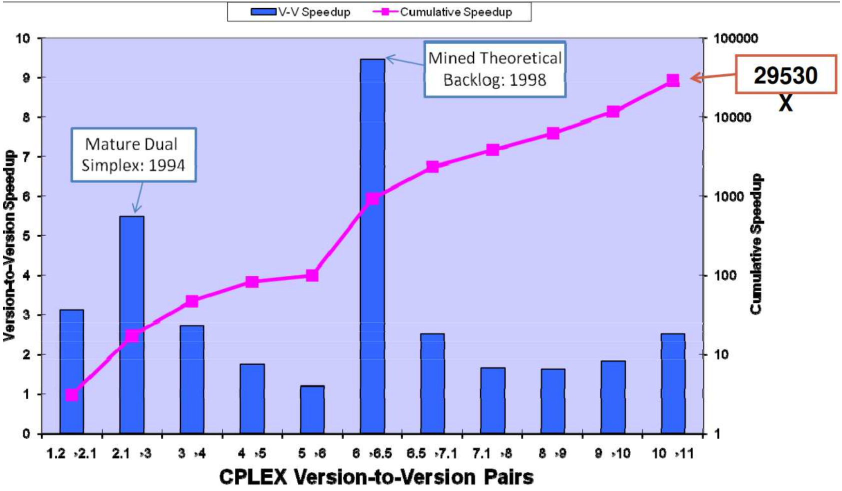
Improvements from Algorithms Can Trump Those From Hardware

Martin Grötschel's production planning benchmark problem (a MIP):

1988	solve time using current computers and LP algorithms:	82 years
2003	solve time using current computers and LP algorithms:	1 minute

- Speed up of 43,000,000X
  10^3X from processor improvements
  10^4X additional from algorithmic improvements

Improvements from Algorithms Can Trump Those From Hardware



1991 (v1.2) to 2007 (v11.0): Moore's Law transistor speedup: ≈ 256X

[Slide from Bixby (CPLEX/GUROBI)]: Solves 1,852 MIPs

Derivative-Free/Zero-Order Optimization

"Some derivatives are unavailable for optimization purposes"

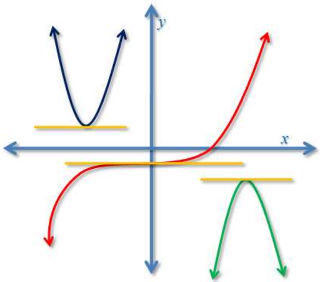
Derivative-Free/Zero-Order Optimization

"Some derivatives are unavailable for optimization purposes"

The Challenge: Optimization is tightly coupled with derivatives

Typical optimality (no noise, smooth functions)

$$\nabla_x f(x^*) + \lambda^T \nabla_x c_E(x^*) = 0, c_E(x^*) = 0$$



(sub)gradients $\nabla_x f$, $\nabla_x c$ enable:

- Faster feasibility
- Faster convergence
  - Guaranteed descent
  - Approximation of nonlinearities
- Better termination
  - Measure of criticality $\|\nabla_x f\|$ or $\|\mathcal{P}_\Omega(\nabla_x f)\|$
- Sensitivity analysis
  - Correlations, standard errors, UQ, ...

Ways to Get Derivatives

(assuming they exist)

Handcoding (HC)

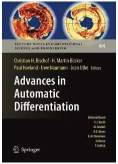
"Army of students/programmers"

- ? Prone to errors/conditioning
- ? Intractable as number of ops increases

Algorithmic/Automatic Differentiation (AD)

"Exact* derivatives!"

- ? No black boxes allowed
- ? Not always automatic/cheap/well-conditioned



Finite Differences (FD)

"Nonintrusive"

- ? Expense grows with n
- ? Sensitive to stepsize choice/noise

→[Moré & W.; SISC 2011], [Moré & W.; TOMS 2012]

. . . then apply derivative-based method (that handles inexact derivatives)

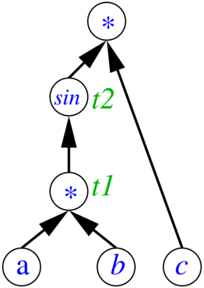
Algorithmic Differentiation

→ [Coleman & Xu; SIAM 2016], [Griewank & Walther; SIAM 2008]

Computational Graph

- $y = sin(a * b) * c$
- Forward and reverse modes
- AD tool provides code for your derivatives

Write codes and formulate problems with AD in mind!

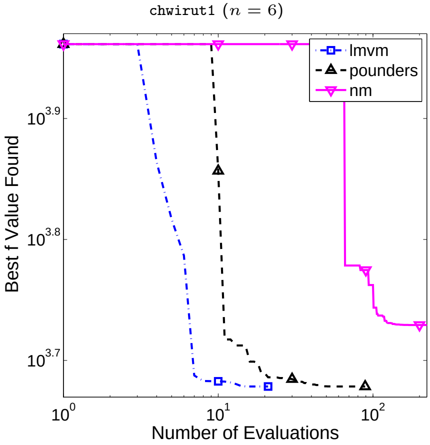


Many tools (see `www.autodiff.org`):

F	`OpenAD`
F/C	`Tapenade`, `Rapsodia`
C/C++	`ADOL-C`, `ADIC`
Matlab	`ADiMat`, `INTLAB`
Python/R	`ADOL-C`

Also done in `AMPL`, `GAMS`, `JULIA`!

The Price of Algorithm Choice: Solvers in PETSc/TAO



Toolkit for Advanced Optimization
[Munson et al.; `mcs.anl.gov/tao`]

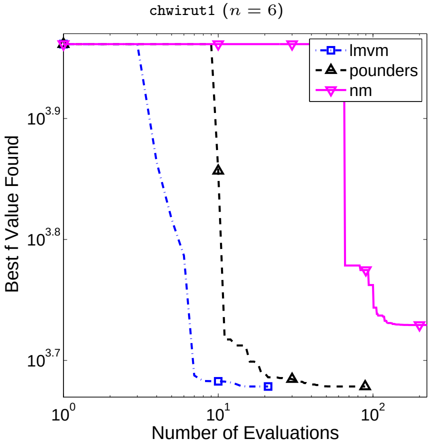
Increasing level of user input:

nm Assumes $\nabla_x f$ unavailable, black box

pounders Assumes $\nabla_x f$ unavailable, exploits problem structure

lmvm Uses available $\nabla_x f$

The Price of Algorithm Choice: Solvers in PETSc/TAO



Toolkit for Advanced Optimization
[Munson et al.; mcs.anl.gov/tao]

Increasing level of user input:

- **nm** Assumes $\nabla_x f$ unavailable, black box
- **pounders** Assumes $\nabla_x f$ unavailable, exploits problem structure
  THIS TALK
- **lmvm** Uses available $\nabla_x f$

DFO methods should be designed to beat finite-difference-based methods

Observe: Constrained by budget on #evals, method limits solution accuracy/problem size

Global Optimization, $\min_{x\in\Omega} f(x)$

Careful:

- Global convergence: Convergence (to a local solution/stationary point) from anywhere in Ω
- Convergence to a global minimizer: Obtain x^* with $f(x^*) \leq f(x) \forall x \in \Omega$

Global Optimization, $\min_{x\in\Omega} f(x)$

Careful:

- Global convergence: Convergence (to a local solution/stationary point) from anywhere in Ω
- Convergence to a global minimizer: Obtain x^* with $f(x^*) \leq f(x) \forall x \in \Omega$

Anyone selling you global solutions when derivatives are unavailable:

either assumes more about your problem (e.g., convex f)

or expects you to wait forever

Törn and Žilinskas: An algorithm converges to the global minimum for any continuous f if and only if the sequence of points visited by the algorithm is dense in Ω.

or cannot be trusted

Global Optimization, $\min_{x \in \Omega} f(x)$

Careful:

- Global convergence: Convergence (to a local solution/stationary point) from anywhere in Ω
- Convergence to a global minimizer: Obtain x^* with $f(x^*) \leq f(x) \, \forall x \in \Omega$

Anyone selling you global solutions when derivatives are unavailable:

either assumes more about your problem (e.g., convex f)

or expects you to wait forever

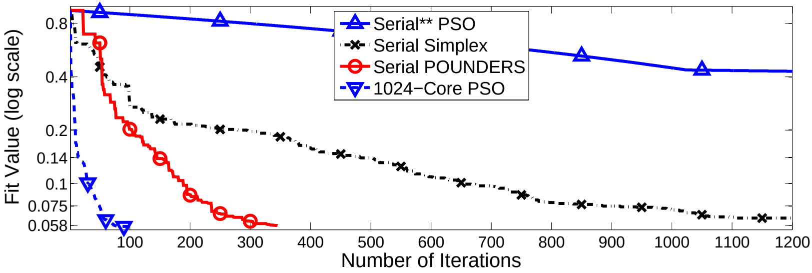
Törn and Žilinskas: An algorithm converges to the global minimum for any continuous f if and only if the sequence of points visited by the algorithm is dense in Ω.

or cannot be trusted

Instead:

- Rapidly find good local solutions and/or be robust to poor solutions
- Consider multistart approaches and/or structure of multimodality

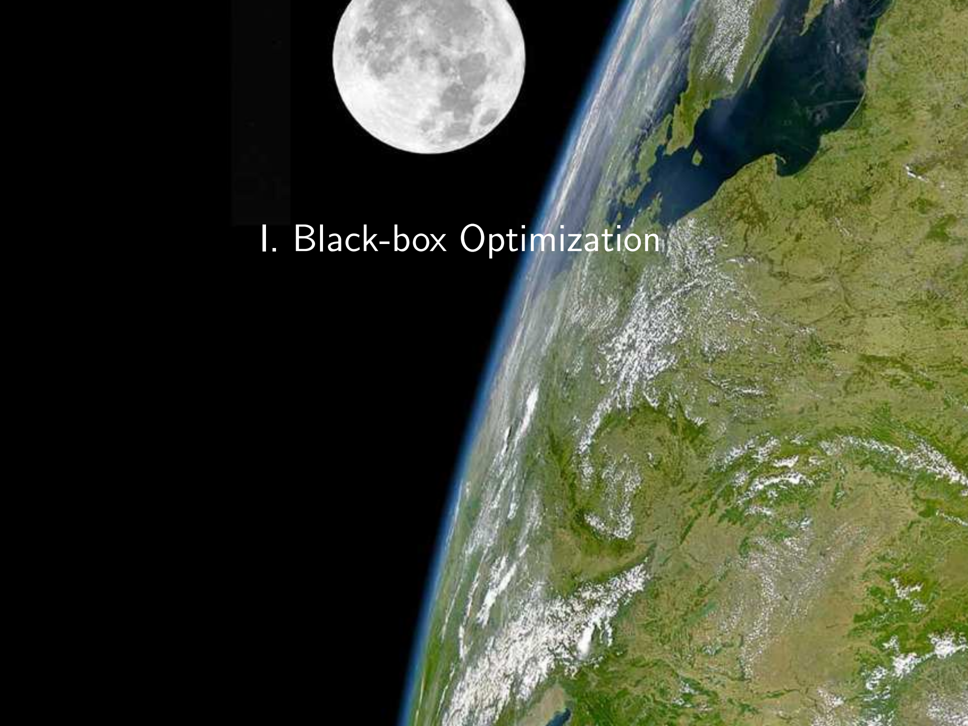
(One Reason) Why We Won't Be Talking About Heuristics



- Heuristics often "embarrassingly/naturally parallel"; PSO= particle swarm method
  - Typically through stochastic sampling/evolution
  - 1024 function evaluations per iteration
- Simplex is Nelder-Mead; POUNDERS is model-based trust-region algorithm
  - one function evaluation per iteration

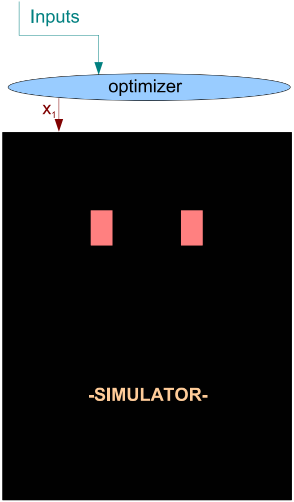
→ Is this an effective use of resources?

→ How many cores would have sufficed?



I. Black-box Optimization

Black-box Optimization Problems



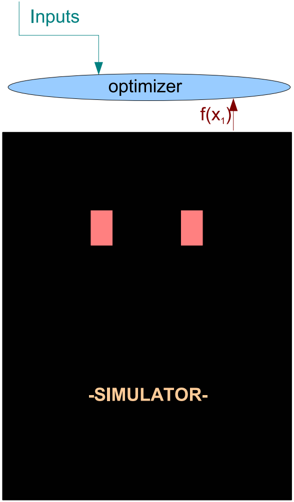
Only knowledge about f is obtained by sampling

- $f = S$ a black box (running some executable-only code or performing an experiment in the lab)
- Only give a single output (no derivatives $\nabla_x S(x)$)

Good solutions guaranteed in the limit, but:

- Usually have computational budget (due to scheduling, finances, deadlines)
- Limited number of evaluations

Black-box Optimization Problems



Only knowledge about f is obtained by sampling

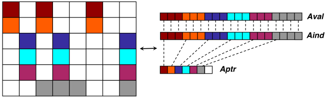
- $f = S$ a black box (running some executable-only code or performing an experiment in the lab)
- Only give a single output (no derivatives $\nabla_x S(x)$)

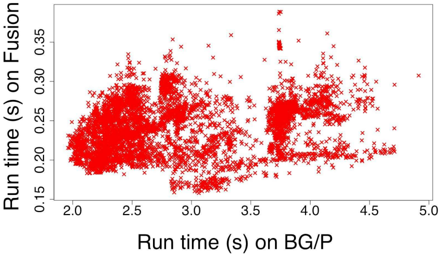
Good solutions guaranteed in the limit, but:

- Usually have computational budget (due to scheduling, finances, deadlines)
- Limited number of evaluations

A Black Box: Automating Empirical Performance Tuning

Given semantically equivalent codes $x^1, x^2, \ldots$, minimize run time subject to energy consumption





$$\min \{ f(x) : (x_{\mathcal{C}}, x_{\mathcal{I}}, x_{\mathcal{B}}) \in \Omega_{\mathcal{C}} \times \Omega_{\mathcal{I}} \times \Omega_{\mathcal{B}} \}$$

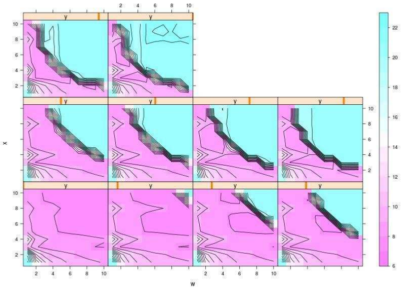
- x multidimensional parameterization (compiler type, compiler flags, unroll/tiling factors, internal tolerances, ...)
- Ω search domain (feasible transformation, no errors)
- f quantifiable performance objective (requires a run)

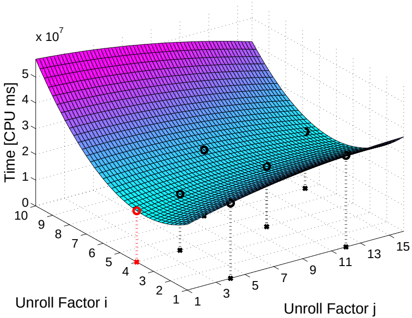
→ [Audet & Orban; SIOPT 2006], [Balaprakash, W., Hovland; ICCS 2011], [Porcelli & Toint; 2016]

Numerical Linear Algebra → [N. Higham; SIMAX 1993], ...

Optimization for Automatic Tuning of HPC Codes

Evaluation of f requires:
transforming source, compilation, (repeated?) execution, checking for correctness





Challenges:

- Evaluating $f(\Omega)$ prohibitively expensive (e.g., 10^{19} discrete decisions)
- f noisy

- Discrete x unrelaxable
- $\nabla_x f$ unavailable/nonexistent
- Many distinct/local solutions

Black-box Algorithms

Solve general problems $\min\{f(x) : x \in \mathbb{R}^n\}$:

- Only require function values (no $\nabla f(x)$)
- Don't rely on finite-difference approximations to $\nabla f(x)$
- Seek greedy and rapid decrease of function value
- Have asymptotic convergence guarantees
- Assume parallel resources are used <u>within</u> function evaluation

Main styles of DFO algorithms

- Randomized methods (later?)
- Direct search methods (pattern search, Nelder-Mead, ...)
- Model-based methods (quadratics, radial basis functions, ...)

Black-Box Algorithms: Stochastic Methods

Random search

Repeat:

1. Randomly generate direction $d^k \in \mathbb{R}^n$
2. Evaluate "gradient-free oracle" $g(x^k; h_k) = \frac{f(x^k + h_k d^k) - f(x^k)}{h_k} d^k$ ($\approx$ directional derivative)
3. Compute $x^{k+1} = x^k - \delta_k g(x^k; h_k)$, evaluate $f(x^{k+1})$

Convergence (for different types of f) tends to be probabilistic

[Kiefer & Wolfowitz; AnnMS 1952], [Polyak; 1987], [Ghadimi & Lan; SIOPT 2013], [Nesterov & Spokoiny; FoCM 2015], . . .

Black-Box Algorithms: Stochastic Methods

Random search

Repeat:

1. Randomly generate direction $d^k \in \mathbb{R}^n$
2. Evaluate "gradient-free oracle" $g(x^k; h_k) = \frac{f(x^k + h_k d^k) - f(x^k)}{h_k} d^k$ ($\approx$ directional derivative)
3. Compute $x^{k+1} = x^k - \delta_k g(x^k; h_k)$, evaluate $f(x^{k+1})$

Convergence (for different types of f) tends to be probabilistic

[Kiefer & Wolfowitz; AnnMS 1952], [Polyak; 1987], [Ghadimi & Lan; SIOPT 2013], [Nesterov & Spokoiny; FoCM 2015], . . .

Stochastic heuristics (nature-inspired methods, etc.)

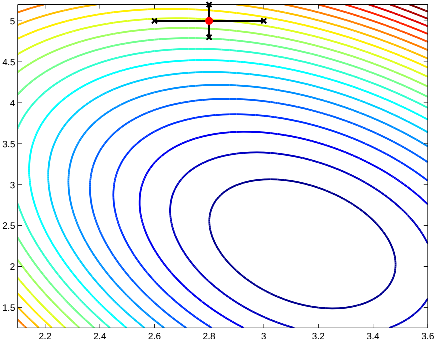
- Popular in practice, especially in engineering
- Typically global in nature
- Require many f evaluations

Pattern Search And Its Variants

Choose a set of directions (pattern or mesh) $\mathcal{D}^k$

Ex.- $\pm$ coordinate directions ($2n$ directions)

Ex.- any minimal positive spanning set ($[e_1, \cdots, e_n, -\sum e_i]$)



Basic iteration ($k \geq 0$):

- Evaluate $f(x^k + \Delta_k d^j)$, $j = 1, \ldots, |\mathcal{D}^k|$
- If $\left[f(x^k + \Delta_k d^j) < f(x^k)\right]$,
  move to $x^{k+1} = x^k + \Delta_k d^j$

  Otherwise shrink Δ_k
- Update $\mathcal{D}^k$

This is an indicator function, does not say anything about the magnitude of f values, just the ordering

Survey $\rightarrow$ [Kolda, Lewis, Torczon; SIREV 2003]

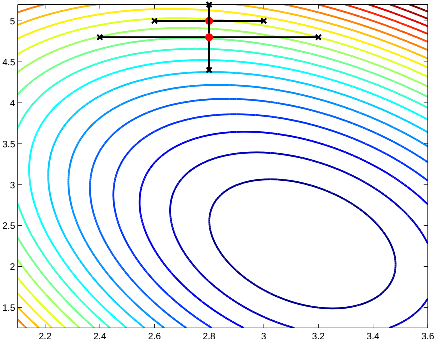
Tools $\rightarrow$ `DFL` [Liuzzi et al.], `NOMAD` [Audet et al.], . . .

Pattern Search And Its Variants

Choose a set of directions (pattern or mesh) $\mathcal{D}^k$

Ex.- $\pm$ coordinate directions ($2n$ directions)

Ex.- any minimal positive spanning set ($[e_1, \cdots, e_n, -\sum e_i]$)



Basic iteration ($k \geq 0$):

- Evaluate $f(x^k + \Delta_k d^j)$, $j = 1, \ldots, |\mathcal{D}^k|$
- If $\left[f(x^k + \Delta_k d^j) < f(x^k)\right]$,
  move to $x^{k+1} = x^k + \Delta_k d^j$

  Otherwise shrink Δ_k
- Update $\mathcal{D}^k$

This is an indicator function, does not say anything about the magnitude of f values, just the ordering

Survey $\rightarrow$ [Kolda, Lewis, Torczon; SIREV 2003]

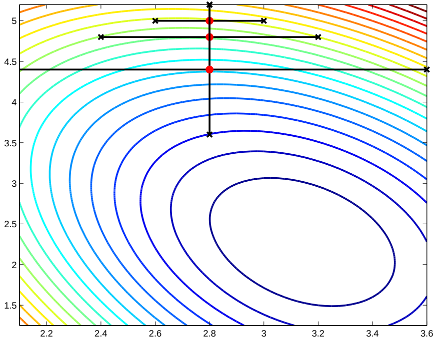
Tools $\rightarrow$ `DFL` [Liuzzi et al.], `NOMAD` [Audet et al.], . . .

Pattern Search And Its Variants

Choose a set of directions (pattern or mesh) $\mathcal{D}^k$

Ex.- $\pm$ coordinate directions ($2n$ directions)

Ex.- any minimal positive spanning set ($[e_1, \cdots, e_n, -\sum e_i]$)



Basic iteration ($k \geq 0$):

- Evaluate $f(x^k + \Delta_k d^j)$, $j = 1, \ldots, |\mathcal{D}^k|$
- If $\left[f(x^k + \Delta_k d^j) < f(x^k)\right]$,
  move to $x^{k+1} = x^k + \Delta_k d^j$

  Otherwise shrink Δ_k
- Update $\mathcal{D}^k$

This is an indicator function, does not say anything about the magnitude of f values, just the ordering

Survey $\rightarrow$ [Kolda, Lewis, Torczon; SIREV 2003]

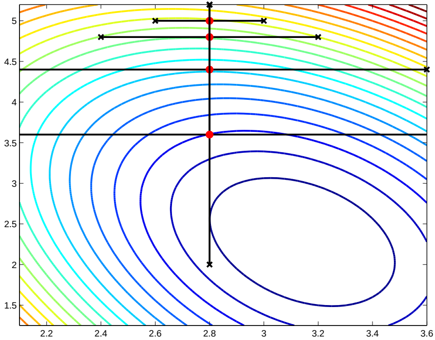
Tools $\rightarrow$ `DFL` [Liuzzi et al.], `NOMAD` [Audet et al.], . . .

Pattern Search And Its Variants

Choose a set of directions (pattern or mesh) $\mathcal{D}^k$

Ex.- $\pm$ coordinate directions ($2n$ directions)

Ex.- any minimal positive spanning set ($[e_1, \cdots, e_n, -\sum e_i]$)



Basic iteration ($k \geq 0$):

- Evaluate $f(x^k + \Delta_k d^j)$, $j = 1, \ldots, |\mathcal{D}^k|$
- If $\left[f(x^k + \Delta_k d^j) < f(x^k)\right]$,
  move to $x^{k+1} = x^k + \Delta_k d^j$

  Otherwise shrink Δ_k
- Update $\mathcal{D}^k$

This is an indicator function, does not say anything about the magnitude of f values, just the ordering

Survey $\rightarrow$ [Kolda, Lewis, Torczon; SIREV 2003]

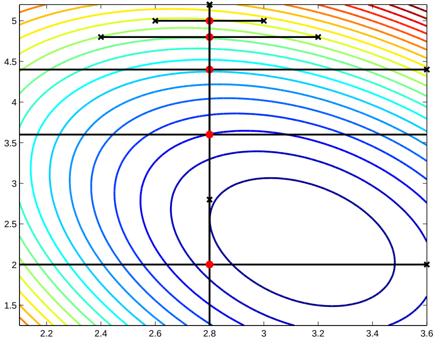
Tools $\rightarrow$ `DFL` [Liuzzi et al.], `NOMAD` [Audet et al.], . . .

Pattern Search And Its Variants

Choose a set of directions (pattern or mesh) $\mathcal{D}^k$

Ex.- $\pm$ coordinate directions ($2n$ directions)

Ex.- any minimal positive spanning set ($[e_1, \cdots, e_n, -\sum e_i]$)



Basic iteration ($k \geq 0$):

- Evaluate $f(x^k + \Delta_k d^j)$, $j = 1, \ldots, |\mathcal{D}^k|$
- If $\left[f(x^k + \Delta_k d^j) < f(x^k)\right]$,
  move to $x^{k+1} = x^k + \Delta_k d^j$

  Otherwise shrink Δ_k
- Update $\mathcal{D}^k$

This is an indicator function, does not say anything about the magnitude of f values, just the ordering

Survey $\rightarrow$ [Kolda, Lewis, Torczon; SIREV 2003]

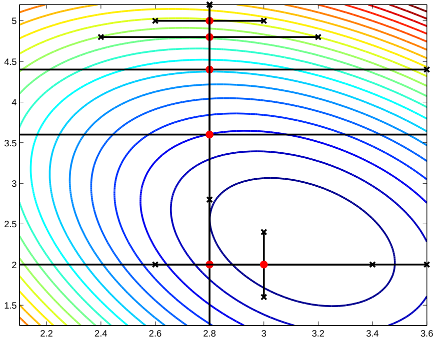
Tools $\rightarrow$ `DFL` [Liuzzi et al.], `NOMAD` [Audet et al.], . . .

Pattern Search And Its Variants

Choose a set of directions (pattern or mesh) $\mathcal{D}^k$

Ex.- $\pm$ coordinate directions ($2n$ directions)

Ex.- any minimal positive spanning set ($[e_1, \cdots, e_n, -\sum e_i]$)



Basic iteration ($k \geq 0$):

- Evaluate $f(x^k + \Delta_k d^j)$, $j = 1, \ldots, |\mathcal{D}^k|$
- If $\left[f(x^k + \Delta_k d^j) < f(x^k)\right]$,
  move to $x^{k+1} = x^k + \Delta_k d^j$

  Otherwise shrink Δ_k
- Update $\mathcal{D}^k$

This is an indicator function, does not say anything about the magnitude of f values, just the ordering

Survey $\rightarrow$ [Kolda, Lewis, Torczon; SIREV 2003]

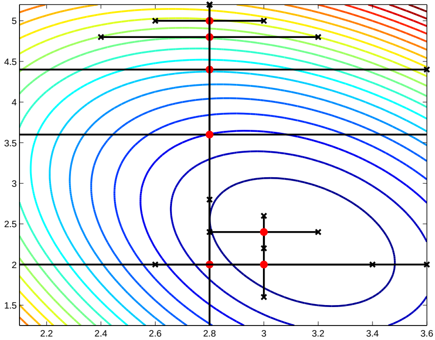
Tools $\rightarrow$ `DFL` [Liuzzi et al.], `NOMAD` [Audet et al.], . . .

Pattern Search And Its Variants

Choose a set of directions (pattern or mesh) $\mathcal{D}^k$

Ex.- $\pm$ coordinate directions ($2n$ directions)

Ex.- any minimal positive spanning set ($[e_1, \cdots, e_n, -\sum e_i]$)



Basic iteration ($k \geq 0$):

- Evaluate $f(x^k + \Delta_k d^j)$, $j = 1, \ldots, |\mathcal{D}^k|$
- If $\left[f(x^k + \Delta_k d^j) < f(x^k)\right]$,
  move to $x^{k+1} = x^k + \Delta_k d^j$

  Otherwise shrink Δ_k
- Update $\mathcal{D}^k$

This is an indicator function, does not say anything about the magnitude of f values, just the ordering

$\rightarrow$ Lends itself well to doing concurrent function evaluations
$\rightarrow$ See also mesh-adaptive direct search methods
$\rightarrow$ Can establish convergence for nonsmooth f

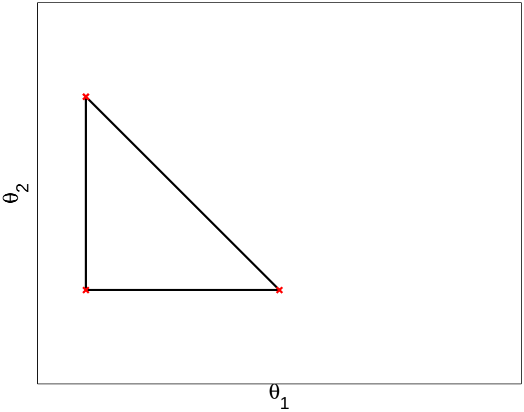
Survey $\rightarrow$ [Kolda, Lewis, Torczon; SIREV 2003]

Tools $\rightarrow$ `DFL` [Liuzzi et al.], `NOMAD` [Audet et al.], . . .

The Nelder-Mead Method [1965]

Basic iteration ($k \geq 0$):

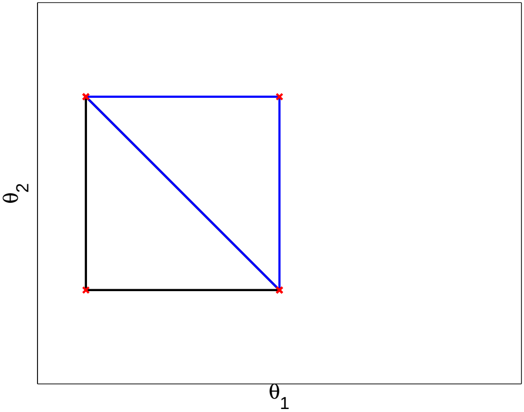
- Evaluate f on the $n+1$ vertices of the simplex $x^k + \Delta_k \mathcal{S}^{(k)}$
- Reflect worst vertex about the best face
- Shrink, contract, or expand $\Delta_k \mathcal{S}^{(k)}$



The Nelder-Mead Method [1965]

Basic iteration ($k \geq 0$):

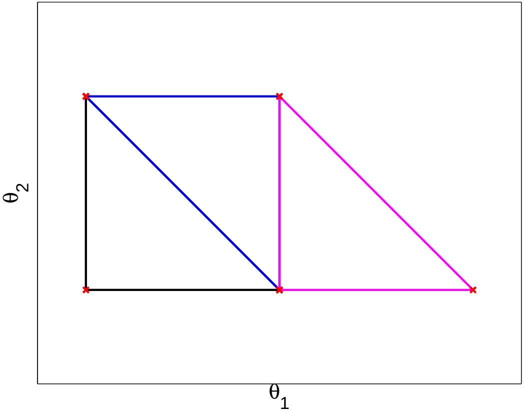
- Evaluate f on the $n+1$ vertices of the simplex $x^k + \Delta_k \mathcal{S}^{(k)}$
- Reflect worst vertex about the best face
- Shrink, contract, or expand $\Delta_k \mathcal{S}^{(k)}$



The Nelder-Mead Method [1965]

Basic iteration ($k \geq 0$):

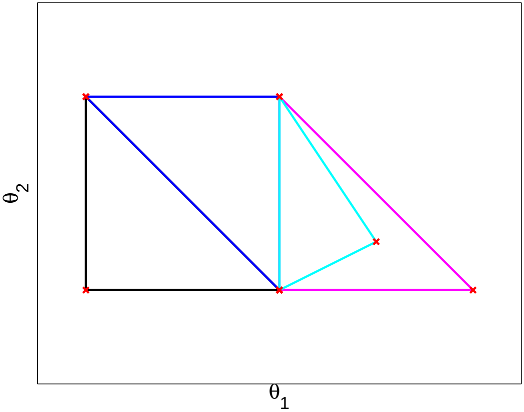
- Evaluate f on the $n+1$ vertices of the simplex $x^k + \Delta_k \mathcal{S}^{(k)}$
- Reflect worst vertex about the best face
- Shrink, contract, or expand $\Delta_k \mathcal{S}^{(k)}$



The Nelder-Mead Method [1965]

Basic iteration ($k \geq 0$):

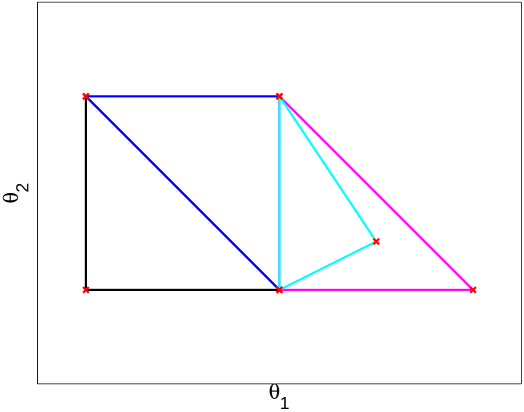
- Evaluate f on the $n+1$ vertices of the simplex $x^k + \Delta_k \mathcal{S}^{(k)}$
- Reflect worst vertex about the best face
- Shrink, contract, or expand $\Delta_k \mathcal{S}^{(k)}$



The Nelder-Mead Method [1965]

Basic iteration ($k \geq 0$):

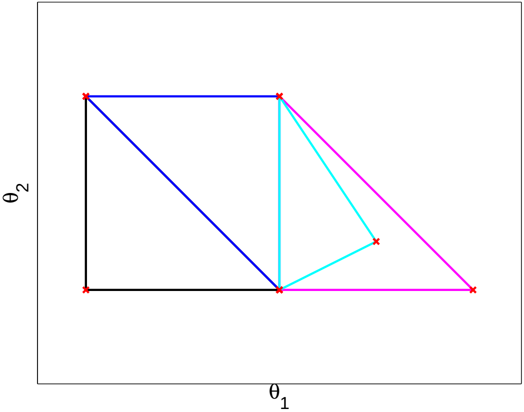
- Evaluate f on the $n+1$ vertices of the simplex $x^k + \Delta_k \mathcal{S}^{(k)}$
- Reflect worst vertex about the best face
- Shrink, contract, or expand $\Delta_k \mathcal{S}^{(k)}$



The Nelder-Mead Method [1965]

Basic iteration ($k \geq 0$):

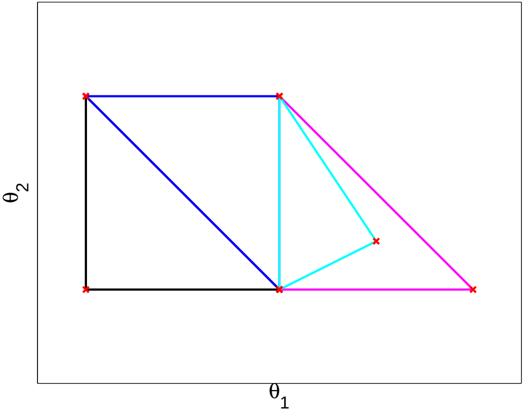
- Evaluate f on the $n+1$ vertices of the simplex $x^k + \Delta_k \mathcal{S}^{(k)}$
- Reflect worst vertex about the best face
- Shrink, contract, or expand $\Delta_k \mathcal{S}^{(k)}$



The Nelder-Mead Method [1965]

Basic iteration ($k \geq 0$):

- Evaluate f on the $n+1$ vertices of the simplex $x^k + \Delta_k \mathcal{S}^{(k)}$
- Reflect worst vertex about the best face
- Shrink, contract, or expand $\Delta_k \mathcal{S}^{(k)}$



Only the order of the function values matter:
$f(\hat{x}) = 1,\ f(\tilde{x}) = 1.0001$ is the same as $f(\hat{x}) = 1,\ f(\tilde{x}) = 10000$.

$\rightarrow$ A very popular (in Numerical Recipes), robust first choice
. . . with nontrivial convergence   Newer NM $\rightarrow$ [Lagarias, Poonen, Wright; SIOPT 2012]

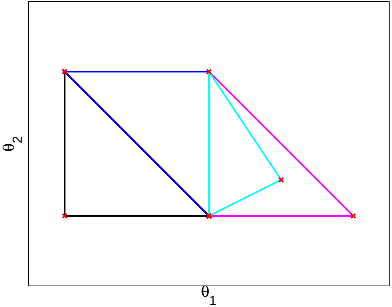
What Are We Missing?

These methods will (eventually) find a local solution
Overview: → [Kolda, Lewis, Torczon, SIREV 2003]

Each evaluation of f is expensive (valuable)

N-M:

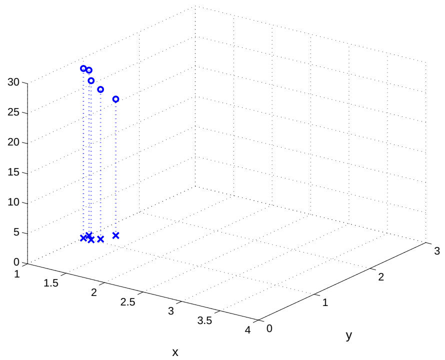
1. Only remembers the last $n + 1$ evaluations
2. Neglects the magnitudes of the function values (order only)
3. Doesn't take into account the special (LS) problem structure



→ This is the reason many direct search methods use a search phase on top of the usual poll phase

Making the Most of Little Information About Smooth f

- f is expensive $\Rightarrow$ can afford to make better use of points
- Overhead of the optimization routine is minimal (negligible?) relative to cost of evaluating simulation



Bank of data, $\{x^i, f(x^i)\}_{i=1}^{k}$:

= Points (& function values) evaluated so far

= Everything known about f

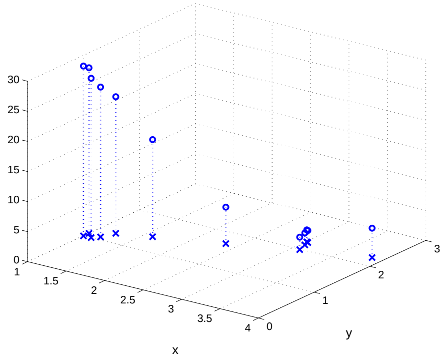
Idea:

- Make use of growing Bank as optimization progresses
- Limit unnecessary evaluations

(geometry/approximation)

Making the Most of Little Information About Smooth f

- f is expensive $\Rightarrow$ can afford to make better use of points
- Overhead of the optimization routine is minimal (negligible?) relative to cost of evaluating simulation



Bank of data, $\{x^i, f(x^i)\}_{i=1}^{k}$:

= Points (& function values) evaluated so far

= Everything known about f

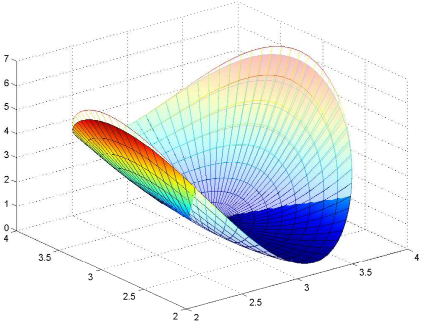
Idea:

- Make use of growing Bank as optimization progresses
- Limit unnecessary evaluations

(geometry/approximation)

Trust-Region Methods Use Models Instead of f

To reduce the number of expensive f evaluations
$\rightarrow$ Replace difficult optimization problem $\min f(x)$
with a much simpler one $\min \{m(x) : x \in \mathcal{B}\}$

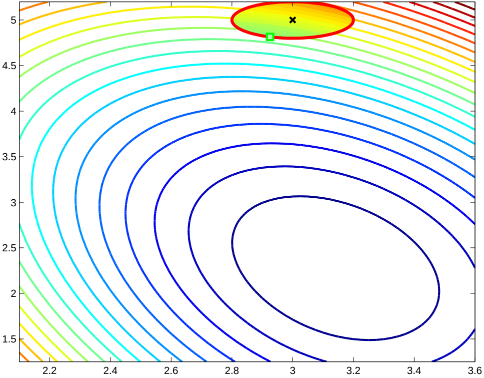


Classic NLP Technique:

- f Original function: computationally expensive, no derivatives
- m Surrogate model: computationally attractive, analytic derivatives

Basic Trust-Region Idea

Use a model $m(x)$ in place of the unwieldy $f(x)$



Optimize over m to avoid expense of f:

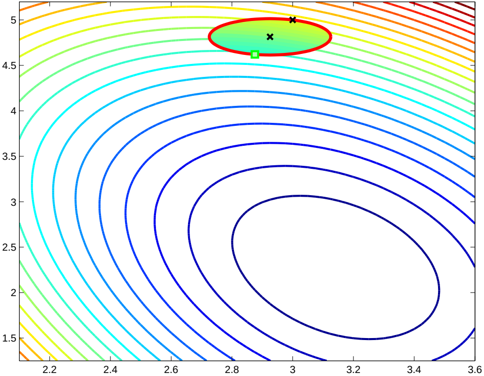
- Trust m to approximate f within $\mathcal{B}_k = \{x \in \mathbb{R}^n : \|x - x^k\| \le \Delta_k\}$,
- Obtain next point from $\min \left\{ m(x^k + s) : x^k + s \in \mathcal{B}_k \right\}$,
- Evaluate function and update (x^k, Δ_k) based on how good the model's prediction was:

$$\rho_k = \frac{f(x^k) - f(x^k + s^k)}{m(x^k) - m(x^k + s^k)}$$

[Conn, Gould, Toint; SIAM, 2000]

Basic Trust-Region Idea

Use a model $m(x)$ in place of the unwieldy $f(x)$



Optimize over m to avoid expense of f:

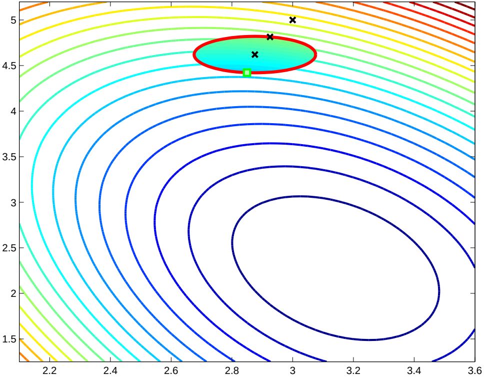
- Trust m to approximate f within $\mathcal{B}_k = \{x \in \mathbb{R}^n : \|x - x^k\| \leq \Delta_k\}$,
- Obtain next point from $\min \left\{m(x^k + s) : x^k + s \in \mathcal{B}_k\right\}$,
- Evaluate function and update (x^k, Δ_k) based on how good the model's prediction was:

$$\rho_k = \frac{f(x^k) - f(x^k + s^k)}{m(x^k) - m(x^k + s^k)}$$

[Conn, Gould, Toint; SIAM, 2000]

Basic Trust-Region Idea

Use a model $m(x)$ in place of the unwieldy $f(x)$



Optimize over m to avoid expense of f:

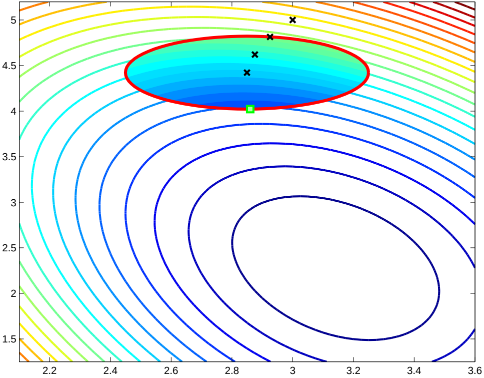
- Trust m to approximate f within $\mathcal{B}_k = \{x \in \mathbb{R}^n : \|x - x^k\| \leq \Delta_k\}$,
- Obtain next point from $\min \left\{m(x^k + s) : x^k + s \in \mathcal{B}_k\right\}$,
- Evaluate function and update (x^k, Δ_k) based on how good the model's prediction was:

$$\rho_k = \frac{f(x^k) - f(x^k + s^k)}{m(x^k) - m(x^k + s^k)}$$

[Conn, Gould, Toint; SIAM, 2000]

Basic Trust-Region Idea

Use a model $m(x)$ in place of the unwieldy $f(x)$



Optimize over m to avoid expense of f:

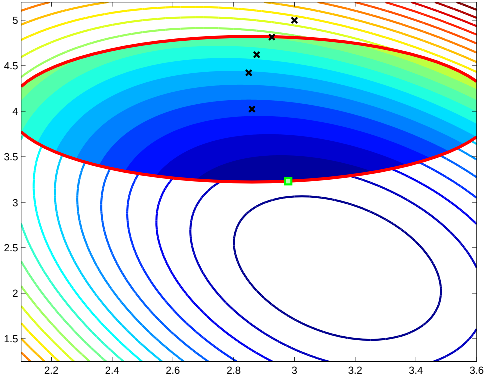
- Trust m to approximate f within $\mathcal{B}_k = \{x \in \mathbb{R}^n : \|x - x^k\| \le \Delta_k\}$,
- Obtain next point from $\min\left\{m(x^k + s) : x^k + s \in \mathcal{B}_k\right\}$,
- Evaluate function and update (x^k, Δ_k) based on how good the model's prediction was:

$$\rho_k = \frac{f(x^k) - f(x^k + s^k)}{m(x^k) - m(x^k + s^k)}$$

[Conn, Gould, Toint; SIAM, 2000]

Basic Trust-Region Idea

Use a model $m(x)$ in place of the unwieldy $f(x)$



Optimize over m to avoid expense of f:

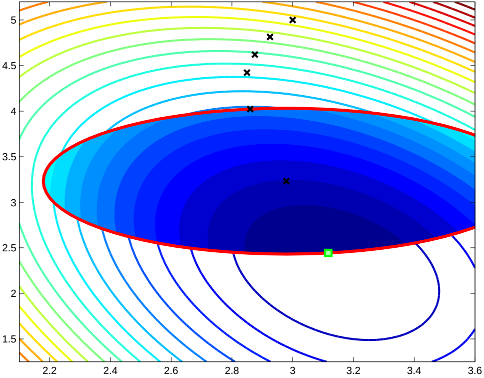
- Trust m to approximate f within $\mathcal{B}_k = \{x \in \mathbb{R}^n : \|x - x^k\| \le \Delta_k\}$,
- Obtain next point from $\min\left\{m(x^k + s) : x^k + s \in \mathcal{B}_k\right\}$,
- Evaluate function and update (x^k, Δ_k) based on how good the model's prediction was:

$$\rho_k = \frac{f(x^k) - f(x^k + s^k)}{m(x^k) - m(x^k + s^k)}$$

[Conn, Gould, Toint; SIAM, 2000]

Basic Trust-Region Idea

Use a model $m(x)$ in place of the unwieldy $f(x)$



Optimize over m to avoid expense of f:

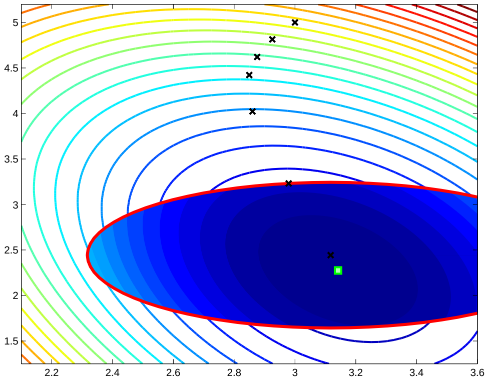
- Trust m to approximate f within $\mathcal{B}_k = \{x \in \mathbb{R}^n : \|x - x^k\| \leq \Delta_k\}$,
- Obtain next point from $\min \left\{ m(x^k + s) : x^k + s \in \mathcal{B}_k \right\}$,
- Evaluate function and update (x^k, Δ_k) based on how good the model's prediction was:

$$\rho_k = \frac{f(x^k) - f(x^k + s^k)}{m(x^k) - m(x^k + s^k)}$$

[Conn, Gould, Toint; SIAM, 2000]

Basic Trust-Region Idea

Use a model $m(x)$ in place of the unwieldy $f(x)$



Optimize over m to avoid expense of f:

- Trust m to approximate f within $\mathcal{B}_k = \{x \in \mathbb{R}^n : \|x - x^k\| \leq \Delta_k\}$,
- Obtain next point from $\min \left\{m(x^k + s) : x^k + s \in \mathcal{B}_k\right\}$,
- Evaluate function and update (x^k, Δ_k) based on how good the model's prediction was:

$$\rho_k = \frac{f(x^k) - f(x^k + s^k)}{m(x^k) - m(x^k + s^k)}$$

[Conn, Gould, Toint; SIAM, 2000]

Where Does the Model Come From?

When derivatives are available:

Taylor-based model $m(x^k + s) = c + (g^k)^T s + \frac{1}{2} s^T H^k s$

- $g^k = \nabla_x f(x^k)$
- $H^k \approx \nabla^2_{x,x} f(x^k)$

Where Does the Model Come From?

When derivatives are available:

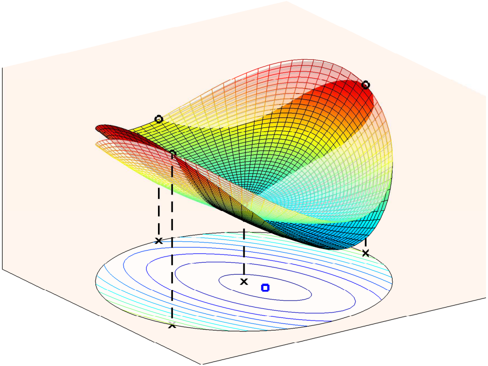
Taylor-based model $m(x^k + s) = c + (g^k)^T s + \frac{1}{2} s^T H^k s$

- $g^k = \nabla_x f(x^k)$
- $H^k \approx \nabla^2_{x,x} f(x^k)$

Without derivatives

- Interpolation-based models
- Regression-based models
- Stochastic/randomized models

Interpolation-Based Quadratic Models



An interpolating quadratic in $\mathbb{R}^2$

$m(x^k + s) = c + g^T s + \frac{1}{2} s^T H s$:

Get the model parameters $c, g, H = H^T$ by demanding interpolation:

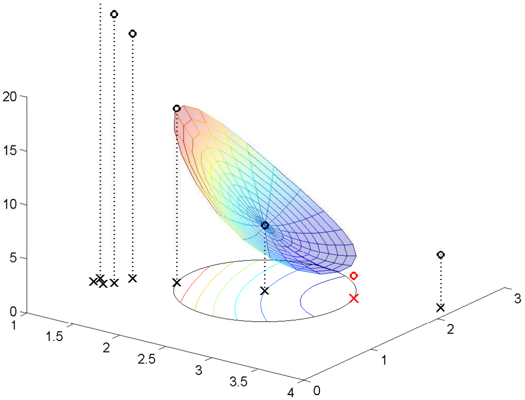
$$m(x^k + y^i) = f(x^k + y^i)$$

for all $y^i \in \mathcal{Y}$ =interpolation set

Main difficulty is $\mathcal{Y}$:

- Use prior function evaluations,
- m well-defined and approximates f locally.

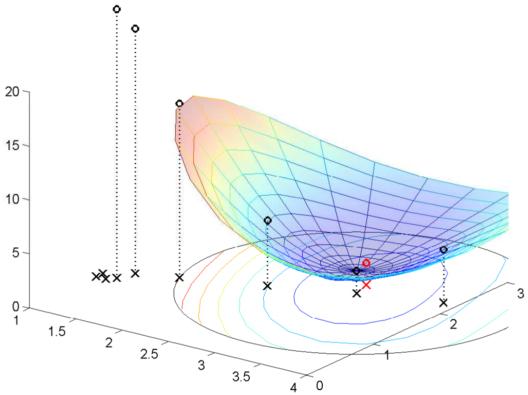
Interpolation-Based Trust-Region Methods



Iteration k:

- Build a model m_k interpolating f on $\mathcal{Y}$
- Trust m_k within region $\mathcal{B}_k$
- Minimize m_k within $\mathcal{B}_k$ to obtain next point for evaluation
- Do expensive evaluation
- Update m_k and $\mathcal{B}_k$ based on how good model prediction was

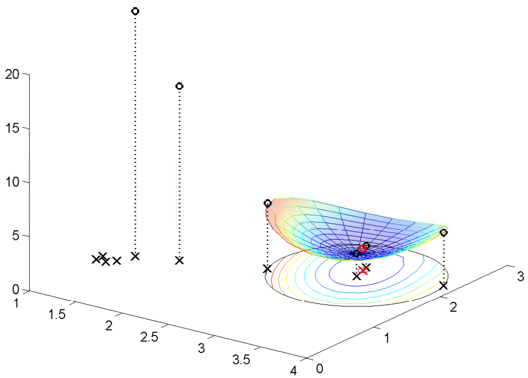
Interpolation-Based Trust-Region Methods



Iteration k:

- Build a model m_k interpolating f on $\mathcal{Y}$
- Trust m_k within region $\mathcal{B}_k$
- Minimize m_k within $\mathcal{B}_k$ to obtain next point for evaluation
- Do expensive evaluation
- Update m_k and $\mathcal{B}_k$ based on how good model prediction was

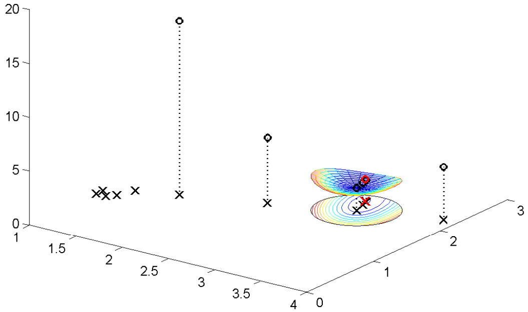
Interpolation-Based Trust-Region Methods



Iteration k:

- Build a model m_k interpolating f on $\mathcal{Y}$
- Trust m_k within region $\mathcal{B}_k$
- Minimize m_k within $\mathcal{B}_k$ to obtain next point for evaluation
- Do expensive evaluation
- Update m_k and $\mathcal{B}_k$ based on how good model prediction was

Interpolation-Based Trust-Region Methods



Iteration k:

- Build a model m_k interpolating f on $\mathcal{Y}$
- Trust m_k within region $\mathcal{B}_k$
- Minimize m_k within $\mathcal{B}_k$ to obtain next point for evaluation
- Do expensive evaluation
- Update m_k and $\mathcal{B}_k$ based on how good model prediction was

Quick Diversion: Polynomial Bases

- Let ϕ denote a basis for some space of polynomials of n variables
  - Linear:
$$\phi(x) = [1,\ x_1,\ \cdots\ ,\ x_n]$$

Quick Diversion: Polynomial Bases

- Let ϕ denote a basis for some space of polynomials of n variables
  - Linear:
  $$\phi(x) = [1,\ x_1,\ \cdots,\ x_n]$$
  - Full quadratics:
  $$\phi(x) = \left[1,\ x_1,\ \cdots,\ x_n\ x_1^2,\ \cdots,\ x_n^2\ x_1x_2,\ \cdots,\ x_{n-1}x_n\right]$$

Quick Diversion: Polynomial Bases

- Let ϕ denote a basis for some space of polynomials of n variables
  - Linear:

$$\phi(x) = [1,\ x_1,\ \cdots,\ x_n]$$

  - Full quadratics:

$$\phi(x) = \left[1,\ x_1,\ \cdots,\ x_n\ x_1^2,\ \cdots,\ x_n^2\ x_1 x_2,\ \cdots,\ x_{n-1}x_n\right]$$

- Given a collection of $p = |\mathcal{Y}|$ points $\mathcal{Y} = \{y^1, \cdots, y^p\}$:

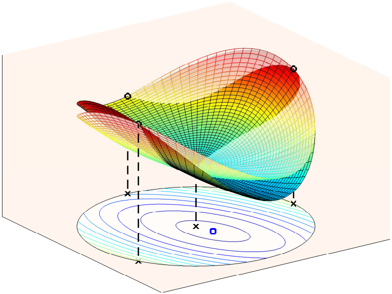
$$\Phi(\mathcal{Y}) = \begin{bmatrix} 1 & y_1^1 & \cdots & y_n^1 & (y_1^1)^2 & \cdots & (y_n^1)^2 & y_1^1 y_2^1 & \cdots & y_{n-1}^1 y_n^1 \\ \vdots & & & & & & & & & \vdots \\ 1 & y_1^p & \cdots & y_n^p & (y_1^p)^2 & \cdots & (y_n^p)^2 & y_1^p y_2^p & \cdots & y_{n-1}^p y_n^p \end{bmatrix}$$

This is a matrix of size $p \times \frac{(n+1)(n+2)}{2}$

Building Models Without Derivatives

Given data $(\mathcal{Y}^k, f(\mathcal{Y}^k))$ and basis Φ, "solve"

$$\Phi(\mathcal{Y}^k)z = \begin{bmatrix} \Phi_c & \Phi_g & \Phi_H \end{bmatrix} \begin{bmatrix} z_c \\ z_g \\ z_H \end{bmatrix} = \underline{\mathsf{f}} = f\left(\mathcal{Y}^k\right)$$



$n = 2, |\mathcal{Y}^k| = 4$

Full quadratics, $|\mathcal{Y}^k| = \frac{(n+1)(n+2)}{2}$

- Geometric conditions on points in $\mathcal{Y}^k$

Undetermined interpolation, $|\mathcal{Y}^k| < \frac{(n+1)(n+2)}{2}$

- Use (Powell) Hessian updates

$$\begin{aligned} \min_{g^k, H^k} \quad & \|H^k - H^{k-1}\|_F^2 \\ \text{s.t.} \quad & q_k = \underline{\mathsf{f}} \text{ on } \mathcal{Y}^k \end{aligned}$$

Regression, $|\mathcal{Y}^k| > \frac{(n+1)(n+2)}{2}$

- Solve $\min_z \|\Phi z - \underline{\mathsf{f}}\|$

Multivariate (Scattered Data) Interpolation is a Different Kind of Animal

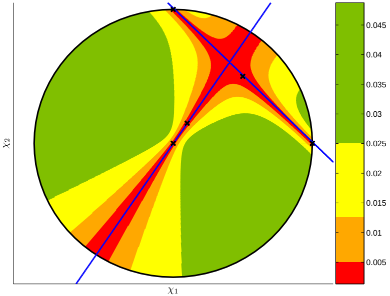
$$m(x^k + y^i) = f(x^k + y^i) \qquad \forall y^i \in \mathcal{Y}$$

$n = 1$ Given p distinct points, can find a unique degree $p-1$ polynomial m

$n > 1$ Not true! (see Mairhuber-Curtis Theorem)

For quadratic models in $\mathbb{R}^n$:

- $\frac{(n+1)(n+2)}{2}$ coefficients
- Unique interpolant may not exist, even when $|\mathcal{Y}| = \frac{(n+1)(n+2)}{2}$
- Locations of the points in $\mathcal{Y}$ must satisfy additional geometric conditions (has nothing to do with f values)



$\longrightarrow$ [Wendland; Cambridge University Press, 2010]

Notions of Nonlinear Model Quality

"Taylor-like" Error Bounds

1. Assuming underlying f is sufficiently smooth
   $=$ derivatives of f exist but are unavailable
2. A model m_k is locally fully linear if:

   For all $x \in \mathcal{B}_k = \{x \in \Omega : \|x - x^k\| \leq \Delta_k\}$
   - $|m_k(x) - f(x)| \leq \kappa_1 \Delta_k^2$
   - $\|\nabla m_k(x) - \nabla f(x)\| \leq \kappa_2 \Delta_k$

   for constants κ_i independent of x and Δ_k.

→[Conn, Scheinberg, Vicente; SIAM 2009]

Notions of Nonlinear Model Quality

"Taylor-like" Error Bounds

1. Assuming underlying f is sufficiently smooth
2. A model m_k is locally fully quadratic if:

   For all $x \in \mathcal{B}_k = \{x \in \Omega : \|x - x^k\| \le \Delta_k\}$

   - $|m_k(x) - f(x)| \le \kappa_1 \Delta_k^3$
   - $\|\nabla m_k(x) - \nabla f(x)\| \le \kappa_2 \Delta_k^2$
   - $\|\nabla^2 m_k(x) - \nabla^2 f(x)\| \le \kappa_3 \Delta_k$

   for constants κ_i independent of x and Δ_k.

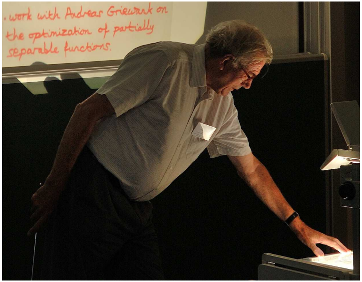
$\longrightarrow$[Conn, Scheinberg, Vicente; SIAM 2009]

Ingredients for Convergence to Stationary Points

$\lim_{k\to\infty} \nabla f(x^k) = 0$ provided:

0. f is sufficiently smooth and regular (e.g., bounded level sets)
1. Control $\mathcal{B}_k$ based on model quality
2. (Occasional) approximation within $\mathcal{B}_k$

   Our quadratics satisfy
   - $|q_k(x) - f(x)| \leq \kappa_1(\gamma_f + \|H^k\|)\Delta_k^2, \qquad x \in \mathcal{B}_k$
   - $\|g^k + H^k(x - x^k) - \nabla f(x)\| \leq \kappa_2(\gamma_f + \|H^k\|)\Delta_k, \qquad x \in \mathcal{B}_k$
3. Sufficient decrease



Michael J.D. Powell, 1936-2015

Survey →[Conn, Scheinberg, Vicente; SIAM 2009]

Methods →[Powell: COBYLA, UOBYQA, NEWUOA, BOBYQA, LINCOA], ...

Line search methods also work →[Kelley et al; IFFCO]

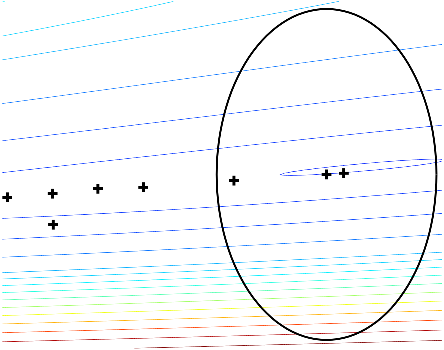
RBF models also work →[W. & Shoemaker; SIREV 2013]

Probabilistic models→[Bandeira, Scheinberg, Vicente; SIOPT 2014]

Greed. Alone. Can. Hurt.

Model-improvement may be needed when:

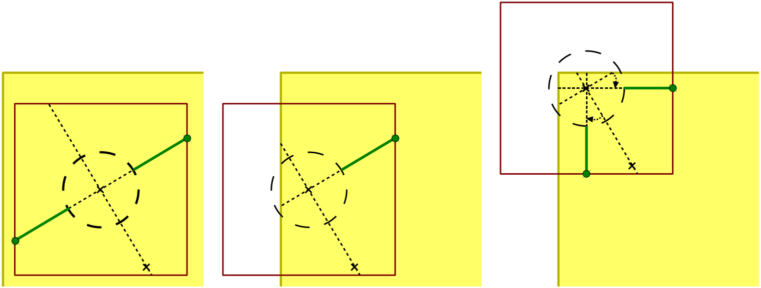
- Nearby points line up
- May not have enough points to ensure model quality in all directions



→ May need n additional evaluations

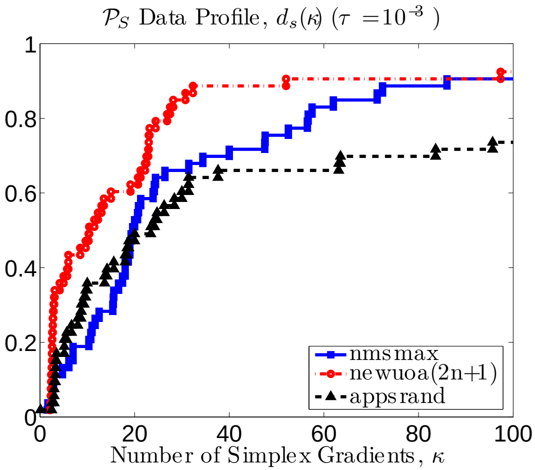
Constraints and Model Quality

Constraints complicate matters
... if one does not allow evaluation of infeasible points



→ May need directions normal to nearby constraints

Performance Comparisons on Test Functions

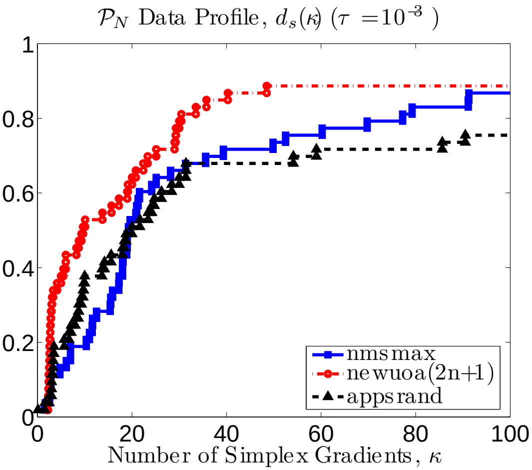


Smooth problems

- When evaluations are sequential, model-based methods (NEWUOA) regularly outperform direct search methods without a search phase (nmsmax, **appsrand**)

→ [Moré & W., SIOPT 2009]

Performance Comparisons on Test Functions



Noisy problems

- When evaluations are sequential, model-based methods (NEWUOA) regularly outperform direct search methods without a search phase (nmsmax, **appsrand**)

→ [Moré & W., SIOPT 2009]

Many Practical Details In Implementations

- Choice of interpolation points $\mathcal{Y}^k$
- Updating of trust region $\mathcal{B}_k$
- Improvement of models

Many Practical Details In Implementations

- Choice of interpolation points $\mathcal{Y}^k$
- Updating of trust region $\mathcal{B}_k$
- Improvement of models

`BOBYQA` [Powell], `DFO` [Scheinberg], `POUNDer` [W.]

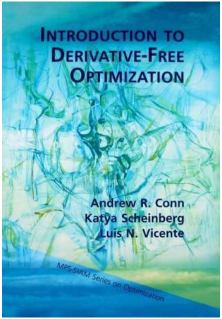
Initialization $p = |\mathcal{Y}^k|$ structured evaluations
Based on input, $\approx 2n + 1$
Based on input, no more than $n + 1$

Interpolation Set $p = |\mathcal{Y}^k|, \forall k$
Bootstrap to $|\mathcal{Y}^k| = \frac{(n+1)(n+2)}{2}$, then fixed
Varies in $\{n + 1, \cdots, \frac{(n+1)(n+2)}{2}\}$ based on available points

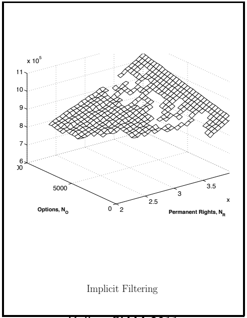
Linear Algebra If $p = \mathcal{O}(n)$, model formation costs only $\mathcal{O}(n^2)$
Expensive
Expensive

Growing, Recent Body of Tools and Resources for Local DFO

? What to use on problems with characteristics X, Y, and Z ?



Conn, Scheinberg, Vicente; SIAM 2009



Kelley; SIAM 2011

Many solvers

Sample considered by Rios & Sahinidis, 2010:

ASA,	BOBYQA,
CMA-ES,	DFO,
DAKOTA/*,	TOMLAB/*,
FMINSEARCH,	GLOBAL,
HOPSPACK,	IMFIL,
MCS,	NEWUOA,
NOMAD,	PSWARM,
SID-PSM,	SNOBFIT

Toward Global Optimization

A quick sketch of a multistart methods and some practical details

- ◇ useful in derivative-based and derivative-free cases
- ◇ obtain a list of distinct minimizers (for post-processing, etc.)
- ◇ simple to get started
- ! simple to abuse/misuse ("I found all minimizers")

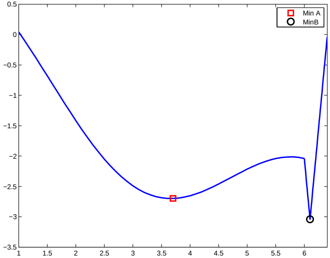
Why Multistart?

Multiple local minima are often of interest in practice:

Design: Multiple objectives (or even constraints) might later be of interest

Simulation Errors: Could have spurious local minima from anomalies in the simulator

Uncertainty: Some minima are more sensitive to perturbations than others (gentle valleys versus steep cliffs)



Global Optimization Multistart Methods

Two phase iterative method

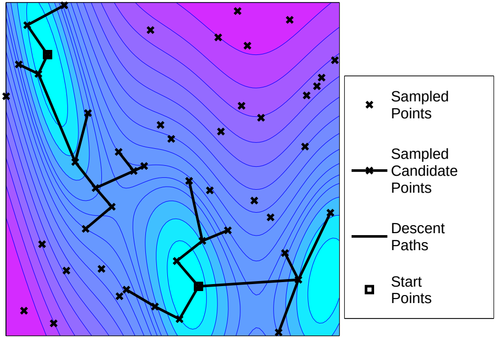
Global Exploration: Sample N points in $\mathcal{D}$. $\leftarrow$ Guarantees convergence

Local Refinement: Start a local minimization algorithm $\mathcal{A}$ from some promising subset of the sample points.

Want to find many (good) local minima while avoiding repeatedly finding the same local minima.

Multi Level Single Linkage (MLSL) Clustering Procedure

Where to start $\mathcal{A}$ in kth iteration [Rinnooy Kan & Timmer, Math. Programming 1987]



Ex.: It. 1 Exploration

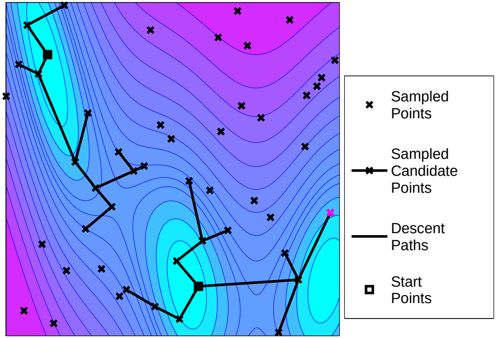
Start $\mathcal{A}$ at each sample point x^i provided:

- $\mathcal{A}$ has not been started from x^i, and
- no other sample point x^j with $f(x^j) < f(x^i)$ is within a distance

$$r_k = \frac{1}{\sqrt{\pi}} \sqrt[n]{\text{vol}(\mathcal{D}) \frac{5\Gamma\left(1+\frac{n}{2}\right)\log(kN)}{kN}},$$

Multi Level Single Linkage (MLSL) Clustering Procedure

Where to start $\mathcal{A}$ in kth iteration [Rinnooy Kan & Timmer, Math. Programming 1987]



Ex.: It. 1 Exploration

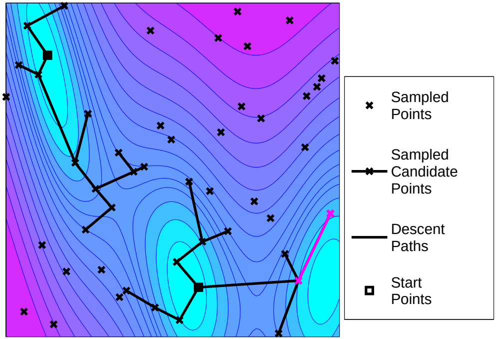
Start $\mathcal{A}$ at each sample point x^i provided:

- $\mathcal{A}$ has not been started from x^i, and
- no other sample point x^j with $f(x^j) < f(x^i)$ is within a distance

$$r_k = \frac{1}{\sqrt{\pi}} \sqrt[n]{\mathrm{vol}(\mathcal{D}) \frac{5\Gamma\left(1+\frac{n}{2}\right)\log(kN)}{kN}},$$

Multi Level Single Linkage (MLSL) Clustering Procedure

Where to start $\mathcal{A}$ in kth iteration [Rinnooy Kan & Timmer, Math. Programming 1987]



Ex.: It. 1 Exploration

Start $\mathcal{A}$ at each sample point x^i provided:

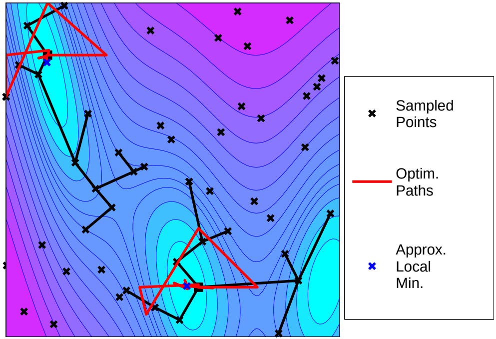
- $\mathcal{A}$ has not been started from x^i, and
- no other sample point x^j with $f(x^j) < f(x^i)$ is within a distance

$$r_k = \frac{1}{\sqrt{\pi}} \sqrt[n]{\text{vol}(\mathcal{D}) \frac{5\Gamma\left(1+\frac{n}{2}\right)\log(kN)}{kN}},$$

Thm [RK-T]- Will start finitely many local runs with probability 1.

Multi Level Single Linkage (MLSL) Clustering Procedure

Where to start $\mathcal{A}$ in kth iteration [Rinnooy Kan & Timmer, Math. Programming 1987]



Ex.: It. 1 Refinement

Start $\mathcal{A}$ at each sample point x^i provided:

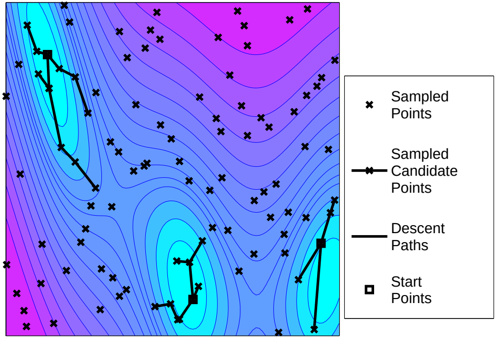
- $\mathcal{A}$ has not been started from x^i, and
- no other sample point x^j with $f(x^j) < f(x^i)$ is within a distance

$$r_k = \frac{1}{\sqrt{\pi}} \sqrt[n]{\text{vol}(\mathcal{D}) \frac{5\Gamma\left(1 + \frac{n}{2}\right) \log(kN)}{kN}},$$

Thm [RK-T]- Will start finitely many local runs with probability 1.

Multi Level Single Linkage (MLSL) Clustering Procedure

Where to start $\mathcal{A}$ in kth iteration [Rinnooy Kan & Timmer, Math. Programming 1987]



Start $\mathcal{A}$ at each sample point x^i provided:

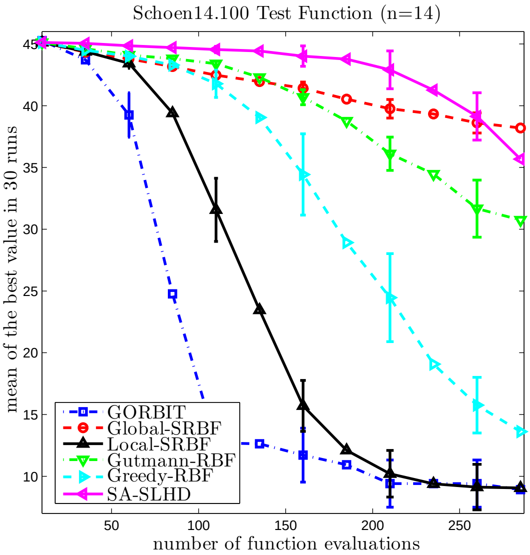
- $\mathcal{A}$ has not been started from x^i, and
- no other sample point x^j with $f(x^j) < f(x^i)$ is within a distance

$$r_k = \frac{1}{\sqrt{\pi}} \sqrt[n]{\mathrm{vol}(\mathcal{D}) \frac{5\Gamma\left(1 + \frac{n}{2}\right) \log(kN)}{kN}},$$

Ex.: It. 2 Exploration

Thm [RK-T]- Will start finitely many local runs with probability 1.

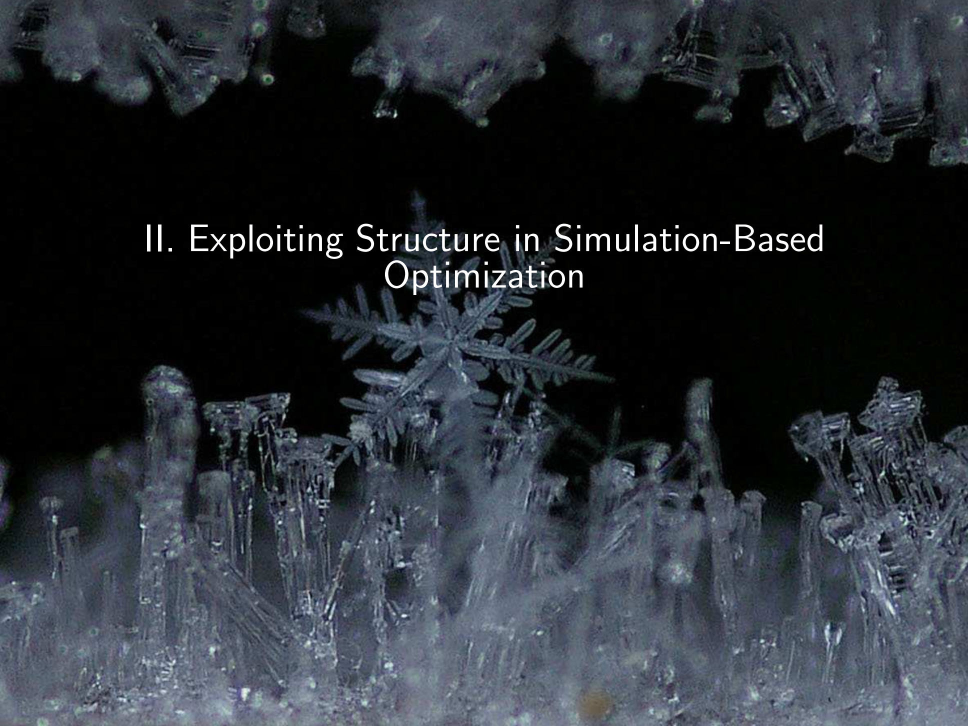
Performance Comparisons on Test Functions



- GORBIT is multistart with RBF model-based method
- SA-SLHD is a heuristic (simulated annealing with a symmetric Latin hypercube design as initialization)

→ [W., Cornell University, 2009]

II. Exploiting Structure in Simulation-Based Optimization



Beyond the Black Box

$$\min f(x) = F[S(x)]$$

So far, $f = S$

Beyond the Black Box

$$\min f(x) = F[S(x)]$$

So far, $f = S$

Your problems are not black-box problems

Beyond the Black Box

$$\min f(x) = F[S(x)]$$

So far, $f = S$

Your problems are not black-box problems

You formulated the problem

$\Rightarrow$ You know more than nothing

Structure in Simulation-Based Optimization, $\min f(x) = F[x, S(x)]$

f is often not a black box S

NLS Nonlinear least squares

$$f(x) = \sum_i (S_i(x) - d_i)^2$$

CNO Composite (nonsmooth) optimization

$$f(x) = h(S(x))$$

SKP Not all variables enter simulation

$$f(x) = g(x_I, x_J) + h(S(x_J))$$

SCO Only some constraints depend on simulation

$$\min\{f(x) : c_1(x) = 0, c_S(x) = 0\}$$

+ Slack variables

$$\Omega_S = \{(x_I, x_J) : S(x_J) + x_I = 0, x_I \geq 0\}$$

...

Model-based methods offer one way to exploit such structure

General Setting – Modeling Smooth $S_1(x), S_2(x), \ldots, S_p(x)$

Assume:

- each S_i is continuously differentiable, available
- each ∇S_i is Lipschitz continuous, unavailable

General Setting – Modeling Smooth $S_1(x), S_2(x), \ldots, S_p(x)$

Assume:

- each S_i is continuously differentiable, available
- each ∇S_i is Lipschitz continuous, unavailable

$m^{S_i} : \mathbb{R}^n \to \mathbb{R}$ approximates S_i on $\mathcal{B}(x, \Delta)$ $\qquad i = 1, \ldots, p$

Fully Linear Models

m^{S_i} fully linear on $\mathcal{B}(x, \Delta)$ if there exist constants $\kappa_{i,\mathrm{ef}}$ and $\kappa_{i,\mathrm{eg}}$ independent of x and Δ so that

$$\begin{aligned} |S_i(x+s) - m^{S_i}(x+s)| &\leq \kappa_{i,\mathrm{ef}}\Delta^2 \quad && \forall s \in \mathcal{B}(0, \Delta) \\ \|\nabla S_i(x+s) - \nabla m^{S_i}(x+s)\| &\leq \kappa_{i,\mathrm{eg}}\Delta \quad && \forall s \in \mathcal{B}(0, \Delta) \end{aligned}$$

NLS– Nonlinear Least Squares $f(x) = \frac{1}{2}\sum_i R_i(x)^2$

Obtain a vector of output $R_1(x), \ldots, R_p(x)$

- Model each R_i

$$R_i(x) \approx m_k^{R_i}(x) = R_i(x^k) + (x - x^k)^\top g_k^{(i)} + \frac{1}{2}(x - x^k)^\top H_k^{(i)}(x - x^k)$$

- Approximate:

$$\begin{aligned}
\nabla f(x) &= \sum_i \nabla \mathbf{R_i(x)} R_i(x) \qquad \longrightarrow \sum_i \nabla m_k^{R_i}(x) R_i(x) \\
\nabla^2 f(x) &= \sum_i \nabla \mathbf{R_i(x)} \nabla \mathbf{R_i(x)}^\top + \sum_i R_i(x) \nabla^2 \mathbf{R_i(x)} \\
&\longrightarrow \sum_i \nabla m_k^{R_i}(x) \nabla m_k^{R_i}(x)^\top + \sum_i R_i(x) \nabla^2 m_k^{R_i}(x)
\end{aligned}$$

- Model f via Gauss-Newton or similar

regularized Hessians $\rightarrow$`DFLS` [Zhang, Conn, Scheinberg]

full Newton $\rightarrow$`POUNDERS` [W., Moré]

NLS– Consequences for $f(x) = \frac{1}{2}\sum_i R_i(x)^2$

Pay a (negligible for expensive S) price in terms of p models

- Save linear algebra using interpolation set $\mathcal{Y}^k$ common to all models
  - Single system solve, multiple right hand sides

$$\Phi(\mathcal{Y}^k)\left[\begin{array}{ccc} z^{(1)} & \cdots & z^{(p)} \end{array}\right] = \left[\begin{array}{ccc} \underline{\mathsf{R}}_1 & \cdots & \underline{\mathsf{R}}_p \end{array}\right]$$

  - m^{R_1} quality $\Rightarrow$ quality of all m^{R_i}

+ (nearly) exact gradients for R_i (nearly) linear
- No longer interpolate function at data points

$$\begin{aligned} m(x^k+\delta) = \quad & f(x^k) \\ & +\delta^\top \textstyle\sum_i g_k^{(i)} R_i(x^k) \\ & +\tfrac{1}{2}\delta^\top \textstyle\sum_i \left(g_k^{(i)}(g_k^{(i)})^\top + R_i(x^k)H_k^{(i)}\right)\delta \\ & + \text{missing h.o. terms} \end{aligned}$$

NLS– POUNDERS in Practice: DFT Calibration/MLE

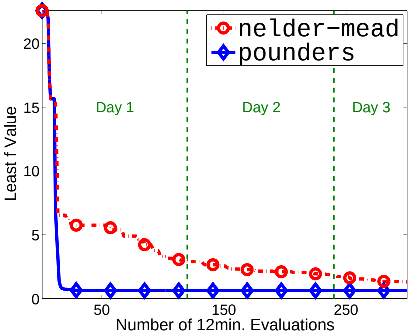
$$\min_x \sum_{i=1}^{p} w_i \left(S_i(x) - d_i\right)^2$$

$S_i(x)$ Simulated (DFT) nucleus property

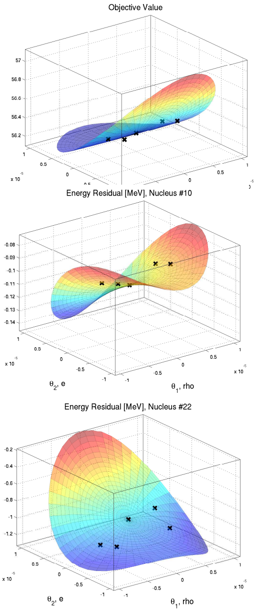
d_i Experimental data i

w_i Weight for data type i

p Parallel simulations (12 wallclock mins)



→ *[Kortelainen et al., PhysRevC 2010]*

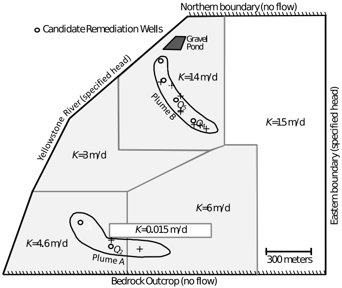


CNO– Composite Nonsmooth Optimization Examples

Ex.- Groundwater remediation

Determine rates x for extraction/injection wells

- Regulator's simulator returns flow $S_i(x)$ in/out of cell i
- Minimize plume fluxes (e.g., regulatory \$ penalties) $f(x) = \sum_i |S_i(x)|$



Lockwood Solvent Ground Water Plume Site (LSGPS)

Ex.- Particle accelerator design

Minimize particle losses: $f(x) = \max_{t_i \in \mathcal{T}_1} S(x;t_i) - \min_{t_i \in \mathcal{T}_2(x)} S(x;t_i)$

CNO– Some Generic Ideas For $f(x) = \sum_{i=1}^{p} |F_i(x)|$

Model-based Approaches:

pounder Ignore structure, model f as usual

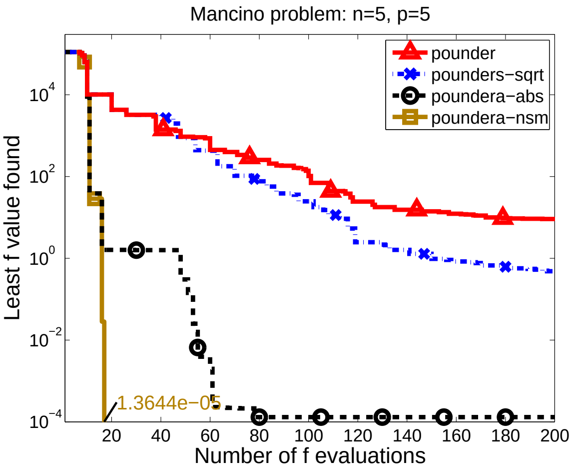
pounders-sqrt $f = \sum_{i=1}^{p} \sqrt{|F_i|}^2$,
model $\sqrt{|F_i|}$ by Q_i
subproblem $\min \sum_{i=1}^{p} \tilde{Q}_i(x)^2$

poundera-abs $f = \sum_{i=1}^{p} |F_i|$,
model $|F_i|$ by Q_i
subproblem $\min \sum_{i=1}^{p} Q_i(x)$

poundera-nsm $f = \sum_{i=1}^{p} |F_i|$,
model F_i by Q_i
subproblem $\min \sum_{i=1}^{p} |Q_i(x)|$

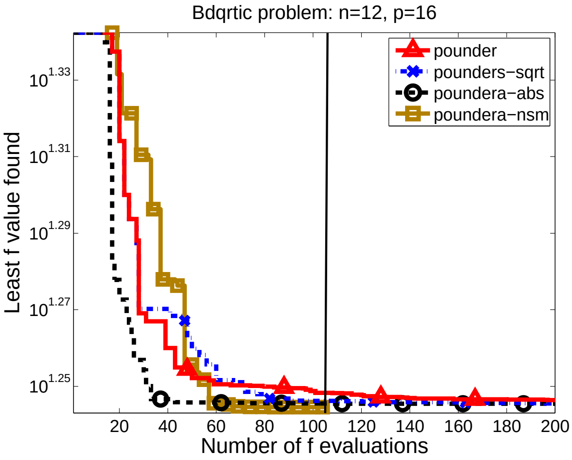
CNO– Results for Generic Ideas, $\min \sum_{i=1}^{p} |F_i(x)|$

pounder	black-box
pounders-sqrt	$\sum_{i=1}^{p} \tilde{Q}_i(x)^2$
poundera-abs	$\sum_{i=1}^{p} Q_i(x)$
poundera-nsm	$\sum_{i=1}^{p} \lvert Q_i(x) \rvert$



CNO– Results for Generic Ideas, $\min \sum_{i=1}^{p} |F_i(x)|$

pounder	black-box
pounders-sqrt	$\sum_{i=1}^{p} \tilde{Q}_i(x)^2$
poundera-abs	$\sum_{i=1}^{p} Q_i(x)$
poundera-nsm	$\sum_{i=1}^{p} \lvert Q_i(x) \rvert$



CNO– Composite Nonsmooth Optimization $f(x) = h(S(x); x)$

nonsmooth (algebraically available) function $h : \mathbb{R}^p \times \mathbb{R}^n \to \mathbb{R}$
of a smooth (blackbox) mapping $S : \mathbb{R}^n \to \mathbb{R}^p$

CNO– Composite Nonsmooth Optimization $f(x) = h(S(x); x)$

nonsmooth (algebraically available) function $h : \mathbb{R}^p \times \mathbb{R}^n \to \mathbb{R}$
of a smooth (blackbox) mapping $S : \mathbb{R}^n \to \mathbb{R}^p$

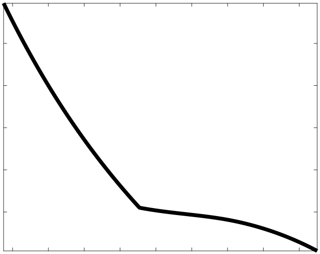
Basic Idea: Knowledge of vector $S(x^k)$ & potential nondifferentiability at $S(x^k)$ should enhance (theoretical and practical) progress to a stationary point

Ex.- $f^1(x) = \|S(x)\|_1 = \sum_{i=1}^p |S_i(x)|$

$$\partial f^1(x) = \sum_{i:S_i(x)\neq 0} \operatorname{sgn}(S_i(x))\nabla S_i(x) + \sum_{i:S_i(x)=0} \mathbf{co}\{-\nabla S_i(x), \nabla S_i(x)\}$$

- $\mathcal{D}^c = \{x : \exists i \text{ with } S_i(x) = 0, \nabla S_i(x) \neq 0\}$

+ Compact $\partial f(x)$
- $\mathcal{D}^c$ depends on $\nabla S_i(x)$



CNO– The Nuisance Set, $\mathcal{N}$

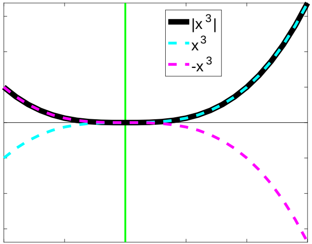
Relaxation $\mathcal{N} \subseteq \mathcal{D}^c$ using only zero-order information

f^1:

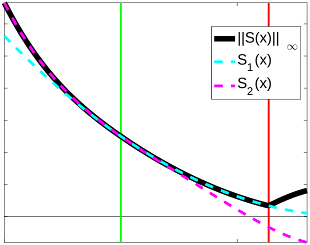
$$\mathcal{N} = \{x : \exists i \text{ with } S_i(x) = 0\}$$

f^∞:

$$\mathcal{N} = \left\{ x : f^\infty(x) = 0 \text{ or } \left| \arg\max_i |S_i(x)| \right| > 1 \right\}$$



$f^1(x) = |x^3|$



$f^\infty(x) = \max\{|S_1(x)|, |S_2(x)|\}$

CNO– The Nuisance Set, $\mathcal{N}$

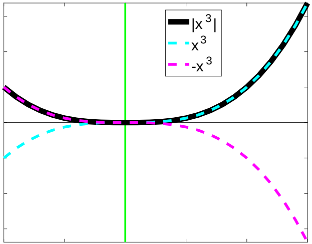
Relaxation $\mathcal{N} \subseteq \mathcal{D}^c$ using only zero-order information

f^1:

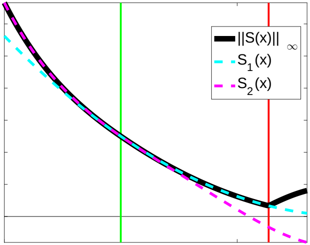
$$\mathcal{N} = \{x : \exists i \text{ with } S_i(x) = 0\}$$

f^∞:

$$\mathcal{N} = \left\{ x : f^\infty(x) = 0 \text{ or } \left| \arg\max_i |S_i(x)| \right| > 1 \right\}$$



$f^1(x) = |x^3|$



$f^\infty(x) = \max\{|S_1(x)|, |S_2(x)|\}$

Observe

When $x^k \notin \mathcal{N}$,

$$\begin{aligned} \partial f(x^k) &= \nabla f(x^k) \\ &= \nabla_x S(x^k)^\top \nabla_S h(S(x^k)) \\ &\approx \nabla_x M(x^k)^\top \nabla_S h(S(x^k)) \end{aligned}$$

and smooth approximation is justified

CNO– Subdifferential Approximation

- $x^k \in \mathcal{N}$, we build a set of generators $\mathcal{G}(x^k)$ based on $\partial_S h(S(x^k))$.
  - $\mathbf{co}\left\{\mathcal{G}(x^k)\right\}$ approximates $\partial f(x^k)$

Ex.- $f^1(x) = \|S(x)\|_1$

$$\mathcal{G}(x^k) = \nabla M(x^k)^\top \left\{ \mathrm{sgn}(S(x^k)) + \bigcup_{i:S_i(x^k)=0} \{-e_i, 0, e_i\} \right\}$$

CNO– Subdifferential Approximation

- $x^k \in \mathcal{N}$, we build a set of generators $\mathcal{G}(x^k)$ based on $\partial_S h(S(x^k))$.
  - $\textbf{co}\left\{\mathcal{G}(x^k)\right\}$ approximates $\partial f(x^k)$

Ex.- $f^1(x) = \|S(x)\|_1$

$$\mathcal{G}(x^k) = \nabla M(x^k)^\top \left\{ \operatorname{sgn}(S(x^k)) + \bigcup_{i:S_i(x^k)=0} \{-e_i, 0, e_i\} \right\}$$

Nearby data $\mathcal{Y} \subset \mathcal{B}(x^k, \Delta_k)$ informs models $M = m^S$ and generator set

- Manifold sampling method uses manifold(s) of $\mathcal{Y}$

$$\nabla M(x^k)^\top \bigcup_{y^i \in \mathcal{Y}} \textbf{mani}\left(S(y^i)\right)$$

- Traditional gradient sampling → [Burke, Lewis, Overton; SIOPT 2005]

$$\bigcup_{y^i \in \mathcal{Y}} \nabla M(y^i)^\top \textbf{mani}\left(S(y^i)\right)$$

CNO– Smooth Trust-Region Subproblem

Smooth master model from minimum-norm element

$$m^f(x^k+s) = f(x^k) + \left\langle s, \mathbf{proj}\left(0, \mathbf{co}\left\{\mathcal{G}(x^k)\right\}\right)\right\rangle + \cdots$$

CNO– Smooth Trust-Region Subproblem

Smooth master model from minimum-norm element

$$m^f(x^k+s) = f(x^k) + \left\langle s, \mathbf{proj}\left(0, \mathbf{co}\left\{\mathcal{G}(x^k)\right\}\right)\right\rangle + \cdots$$

$\Rightarrow$ smooth subproblems

$$\min\left\{m^f(x^k+s) : s \in \mathcal{B}(0,\Delta_k)\right\}$$

vs.

Nonsmooth subproblems

$$\min\left\{h\left(M\left(x^k+s\right)\right) : s \in \mathcal{B}(0,\Delta_k)\right\}$$

- Convex h (e.g., $\|S(x)\|_1$) and ∇S_i is Lipschitz
  $\Rightarrow$ every cluster point of $\{x^k\}_k$ is Clarke stationary
  $\longrightarrow$ [Larson, Menickelly, W.; Preprint 2016]
- OK to sample at $x^k \in \mathcal{D}^C$
- More general (piecewise differentiable f) results:
  $\longrightarrow$ [Larson, Khan, W.; in prog. 2016]

- Requires convex h

$\longrightarrow$ [Fletcher; MathProgStudy 1982]

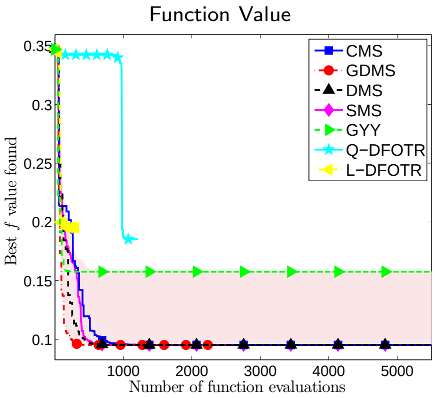
$\longrightarrow$ [Grapiglia, Yuan, Yuan; C&A Math. 2016]

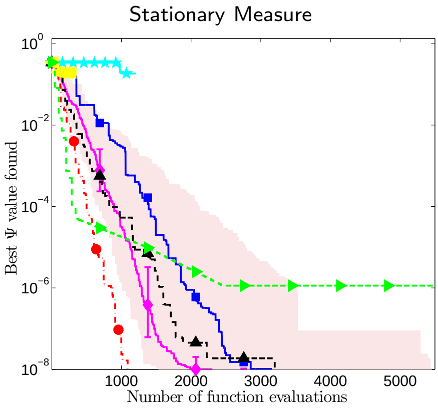
Complexity results

$\longrightarrow$ [Garmanjani, Júdice, Vicente; SIOPT 2016]

Roger Fletcher

CNO– Example Performance on L_1 Test Problems





Smooth black-box methods can fail in practice, even when $\mathcal{D}^C$ has measure zero

Numerical tests: $\longrightarrow$ [Larson, Menickelly, W.; Preprint 2016]

SKP– Some Known Partials Example

Ex.- Bi-level model calibration structure

$$\min_x \left\{ f(x) = \sum_{i=1}^{p} (S_i(x) - d_i)^2 \right\}$$

$S_i(x)$ solution to lower-level problem depending only on x_J

$$\begin{aligned} S_i(x) &= g_i(x) + \min_y \{h_i(x_J; y) : y \in \mathcal{D}_i\} \\ &= g_i(x) + h_i(x_J; y_{i,*}[x_J]) \end{aligned}$$

For $x = (x_I, x_J)$

- $\nabla_{x_I} S_i(x_I, x_J)$ available
- $\nabla_{x_J} S_i(x) \approx \nabla_{x_J} g_i(x) + \nabla_{x_J} m^{\tilde{S}_i}(x_J)$
- $S_i(x)$ continuous and smooth in x_I
- $g_i(x)$ cheap to compute!
- No noise/errors introduced in $g_i(x)$

General bi-level $\longrightarrow$[Conn & Vicente, OMS 2012]

SKP– Some Known Partials

$x = (x_I, x_J)$; have $\frac{\partial f}{\partial x_I}$ but not $\frac{\partial f}{\partial x_J}$

"Solve"

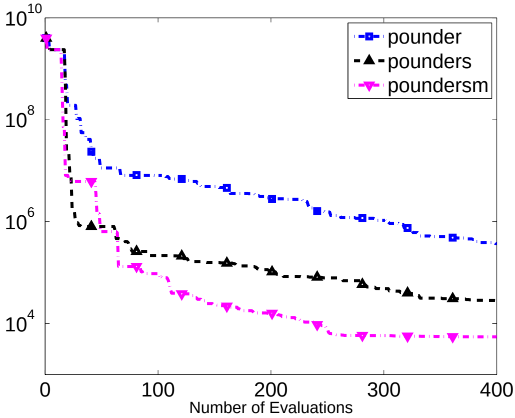
$$\Phi z = \underline{\mathsf{f}}$$

with known $z_{g,I}, z_{H,I}$

$$\begin{bmatrix} \Phi_c & \Phi_{g,J} & \Phi_{H,J} \end{bmatrix} \begin{bmatrix} z_c \\ z_{g,J} \\ z_{H,J} \end{bmatrix} = \underline{\mathsf{f}} - \Phi_{g,I} z_{g,I} - \Phi_{H,I} z_{H,I}$$

- Still have interpolation where required
- Effectively lowers dimension to $|J| = n - |I|$ for
  - approximation
  - model-improving evaluations
  - linear algebra
- $\lim_{k\to\infty} \nabla f(x^k) = 0$ as before:
  - Guaranteed descent in some directions

SKP– Numerical Results With Some Partials



Three approaches:

- \- black box
- s exploit least squares
- m use ∇_{x_I} derivatives

- $n = 16, |I| = 3$
- 5-10 secs/evaluation

Same algorithmic framework, performance advantages from exploiting structure

$\longrightarrow$[Bertolli, Papenbrock, W., PRC 2012]

SCO- General Constraints

$\min\{f(x) : c_1(x) = 0, c_S(x) = 0\}$

- Lagrangian (key to optimality conditions):

$$
\begin{aligned}
\nabla L &= \nabla f + \lambda_1^\top \nabla c_1 + \lambda_2^\top \nabla \mathbf{c_S} \\
&\to \nabla f + \lambda_1^\top \nabla c_1 + \lambda_2^\top \nabla m
\end{aligned}
$$

- Use favorite method: filters, augmented Lagrangian, . . .
- Slack variables
  - Do not increase effective dimension
  - Subproblems can treat separately
  - Know derivatives

→[Lewis & Torczon; 2010]

Modified AL methods →[Diniz-Ehrhardt, Martínez, Pedroso; C&A Math. 2011]

SBO constraints have unique properties →[Le Digabel & W.; ANL/MCS-P5350-0515 2016]

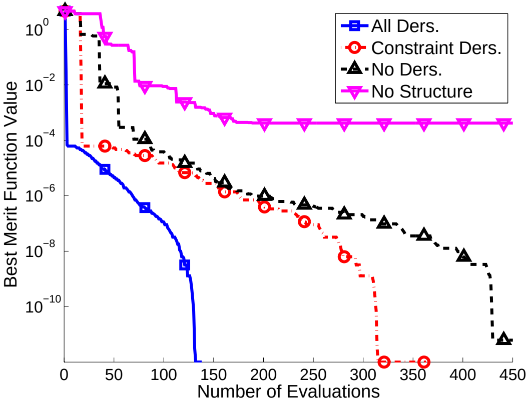
SCO– What Constraint Derivatives Buy You

Ex.- Augmented Lagrangian methods, $L_A(x,\lambda;\mu) = f(x) - \lambda^\top c(x) + \frac{1}{\mu}\|c(x)\|^2$

$\min_x \{f(x) : c(x) = 0\}$

Four approaches:

1. Penalize constraints
2. Treat c and f both as (separate) black boxes
3. Work with f and $\nabla_x c$
4. Have both $\nabla_x f$ and $\nabla_x c$



$n = 15$, 11 constraints

with no explicit internal vars, 15 var, 11 cons

So You Want To Solve A Hard Optimization Problem?

Mathematically unwrap problems to expose the deepest black boxes!

- It is easy to get started with derivative-free methods
- You should strive to obtain derivatives & apply methods from every other lecture

- Model-based methods can make use of expensive function values
- Structure is everywhere, even in "black-box" / legacy code-driven optimization problems
- By exploiting structure, optimization can solve grand-challenge problems in ⟨`insert your field here`⟩:
  - Model residuals $\{r_i(x)\}_i$, not $\|r(x)\|$
  - Model constraints $\{c_i(x)\}_i$, not a penalty $P(c(x))$
  - Explicitly handle nonsmoothness (and noise)

Send me your structured SBO problems!

→ `www.mcs.anl.gov/~wild`